Proof Complexity of Practical Integer Programming

Noah Fleming

Based on

- On the power and Limitations of Branch and Cut
  Fleming, Göös, Impagliazzo, Pitassi, Robere, Wigderson

In This Talk...

- Algorithm analysis from proof complexity
- The proof complexity of integer programming
  - → Cutting planes & Cutting Planes
  - → Branch-and-cut & Stabbing Planes
- Stabbing Planes — a "query" view of Cutting Planes
- Short proofs of $\mathbb{F}_q$ linear equations

Algorithm Analysis from Proofs

Idea: Formalize the techniques used in a class of algorithms A as a proof system P

Algorithm Analysis from Proofs

Idea: Formalize the techniques used in a class of algorithms A as a proof system P

- Hides practical details of algorithms

Algorithm Analysis from Proofs

Idea: Formalize the techniques used in a class of algorithms A as a proof system P

- Hides practical details of algorithms
- Lower bounds on P-proofs → lower bounds on runtime of A

Algorithm Analysis from Proofs

Idea: Formalize the techniques used in a class of algorithms A as a proof system P

- e.g. Algorithms for SAT
  - CDCL and Resolution

Algorithm Analysis from Proofs

Idea: Formalize the techniques used in a class of algorithms A as a proof system P

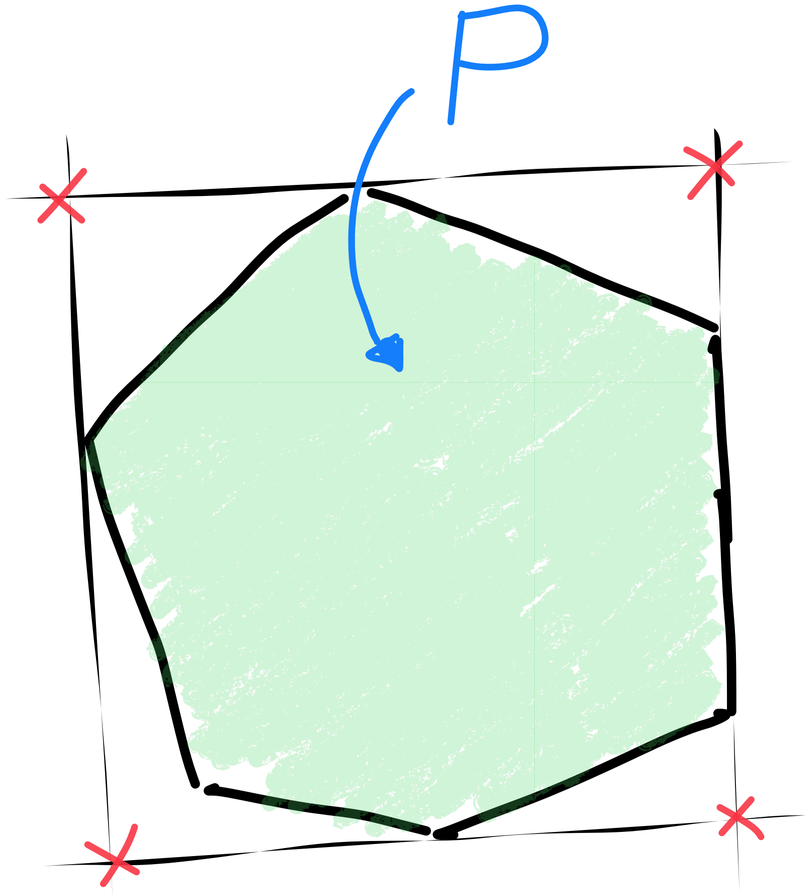
- e.g. Algorithms for SAT
  - CDCL and Resolution
- e.g. Algorithms for Integer Programming
  - Chvátal-Gomory Cutting Planes and Cutting Planes

Integer Programming

Integer-programming: Given $A x \geq b$ find $x \in \mathbb{Z}^n, A x \geq b$

Integer Programming

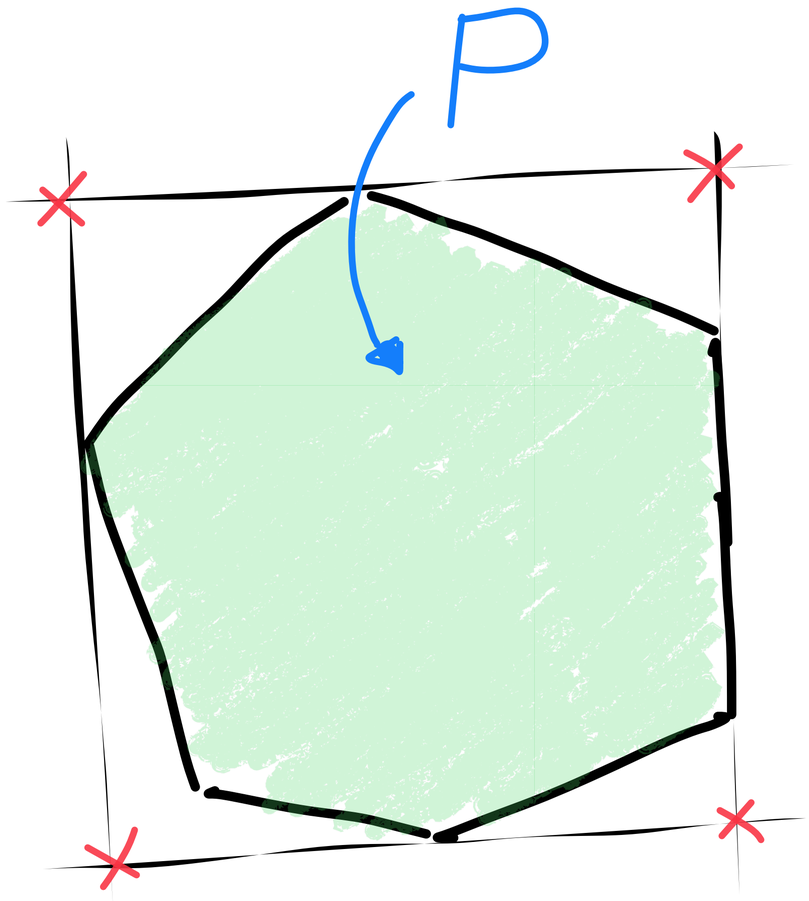
Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$



Integer Programming

Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$

A Classic approach: Chvátal-Gomory Cutting Planes

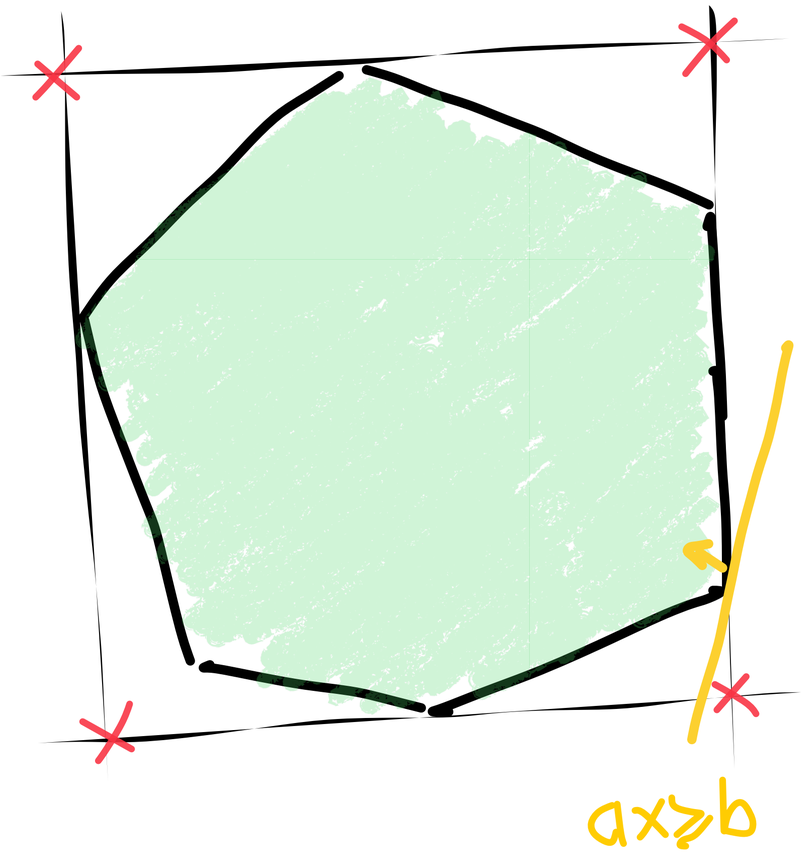


Chvátal-Gomory Cutting Planes

Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$

A Classic approach: Chvátal-Gomory Cutting Planes

CG-Cut: If $ax \geq b$ is valid for P

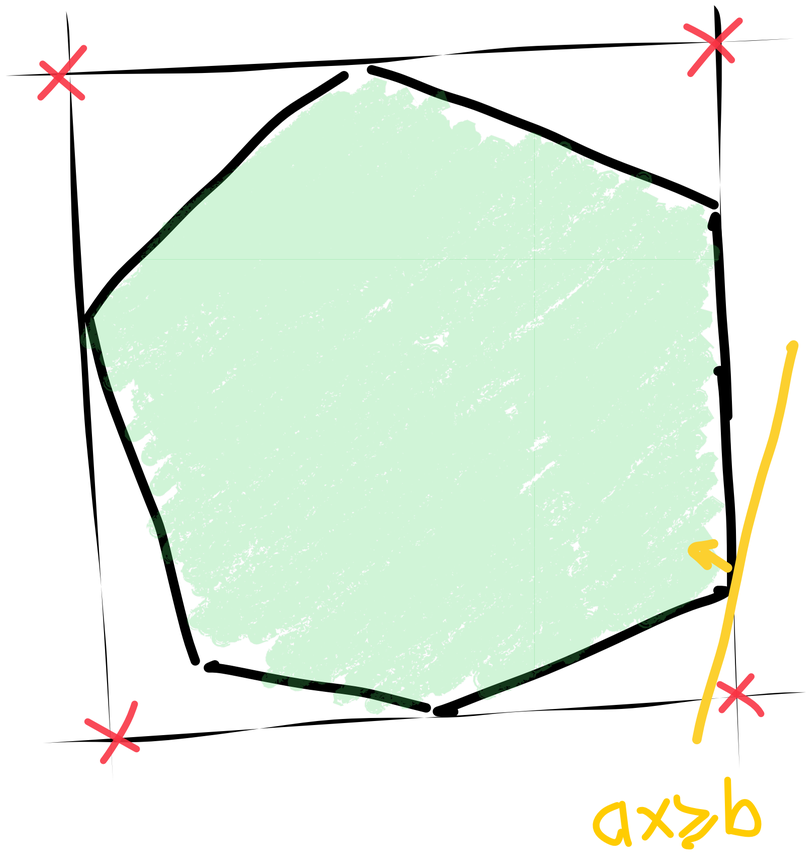


Chvátal-Gomory Cutting Planes

Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$

A Classic approach: Chvátal-Gomory Cutting Planes

CG-Cut: If $ax \geq b$ is valid for P ($a \in \mathbb{Z}^n$, $b \in \mathbb{R}$)

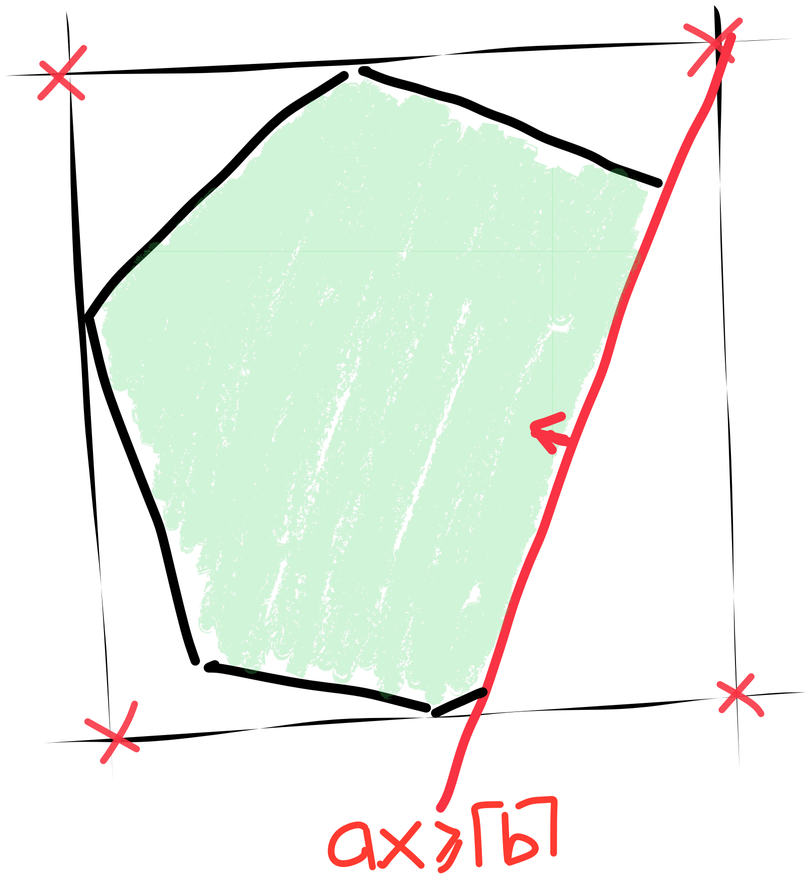


Chvátal-Gomory Cutting Planes

Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$

A Classic approach: **Chvátal-Gomory Cutting Planes**

CG-Cut: If $ax \geq b$ ($a \in \mathbb{Z}^n$, $b \in \mathbb{R}$) is valid for P then $ax \geq \lceil b \rceil$ is a CG-cut



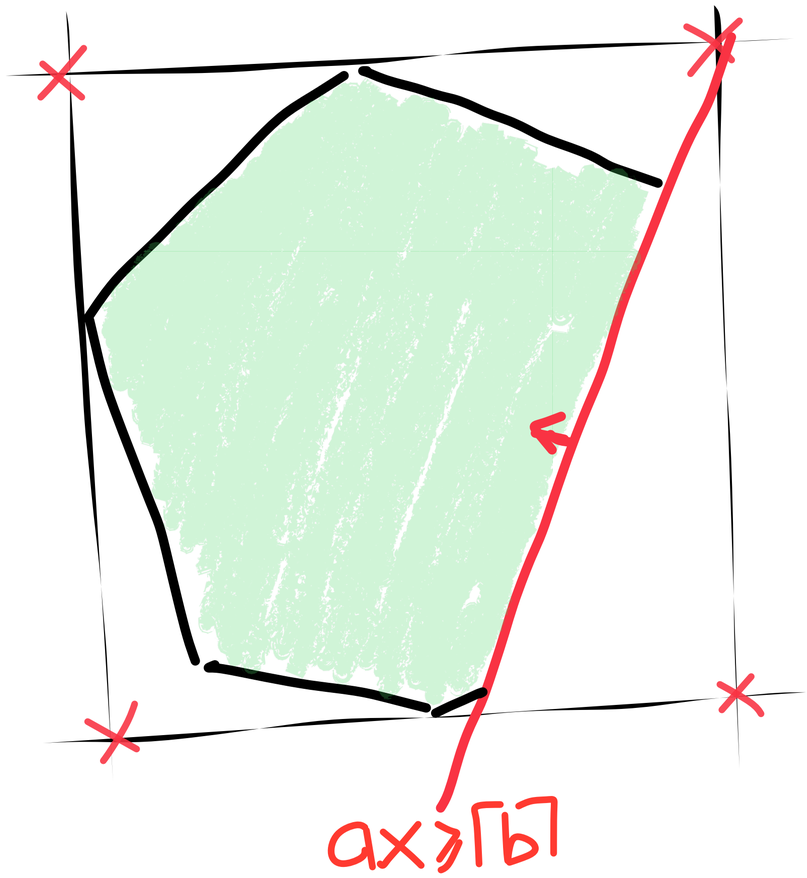
Chvátal-Gomory Cutting Planes

Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$

A Classic approach: Chvátal-Gomory Cutting Planes

CG-Cut: If $ax \geq b$ ($a \in \mathbb{Z}^n$, $b \in \mathbb{R}$) is valid for P
then $ax \geq \lceil b \rceil$ is a CG-cut

→ Preserves integer solutions to P



Chvátal-Gomory Cutting Planes

Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$

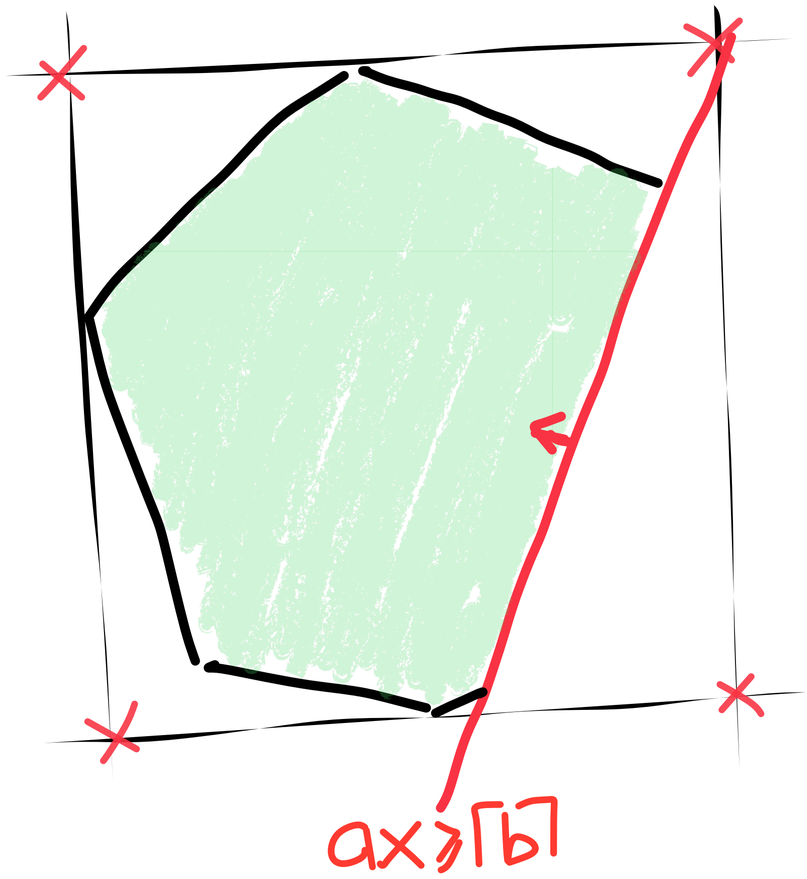
CG-Cutting Planes

Heuristically add CG-cuts to P

CG-Cut: If $ax \geq b$ is valid for P

then $ax \geq \lceil b \rceil$ is a CG-cut

→ Preserves integer solutions to P



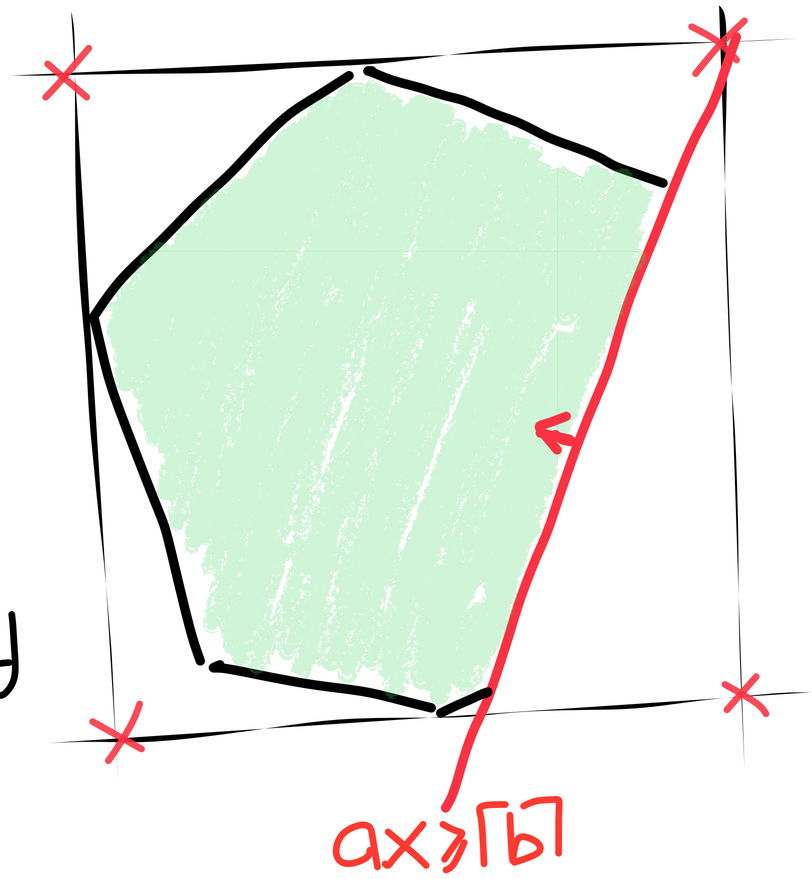
Chvátal-Gomory Cutting Planes

Integer-programming: Given $P = \{x : Ax \geq b\}$ find $x \in P \cap \mathbb{Z}^n$

CG-Cutting Planes

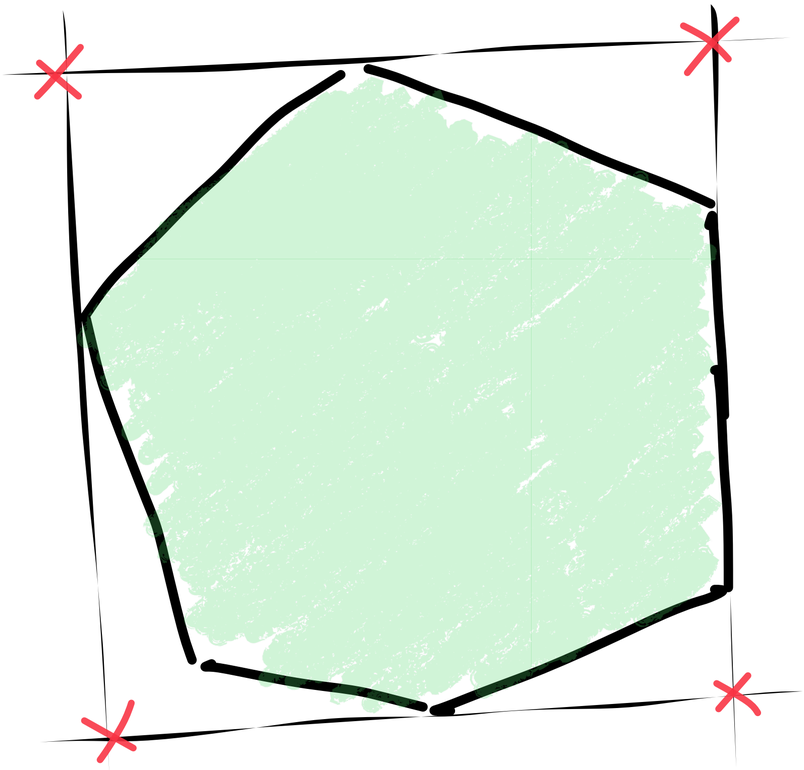
Heuristically add CG-cuts to P until:

- an integer solution is found
- the empty polytope is deduced



Cutting Planes [CCT87]

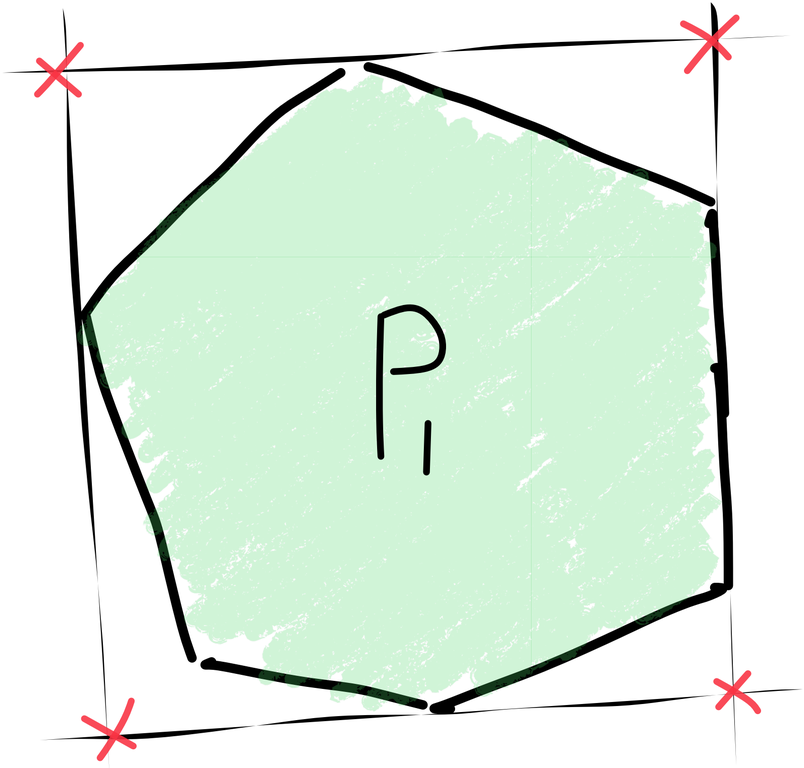
Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

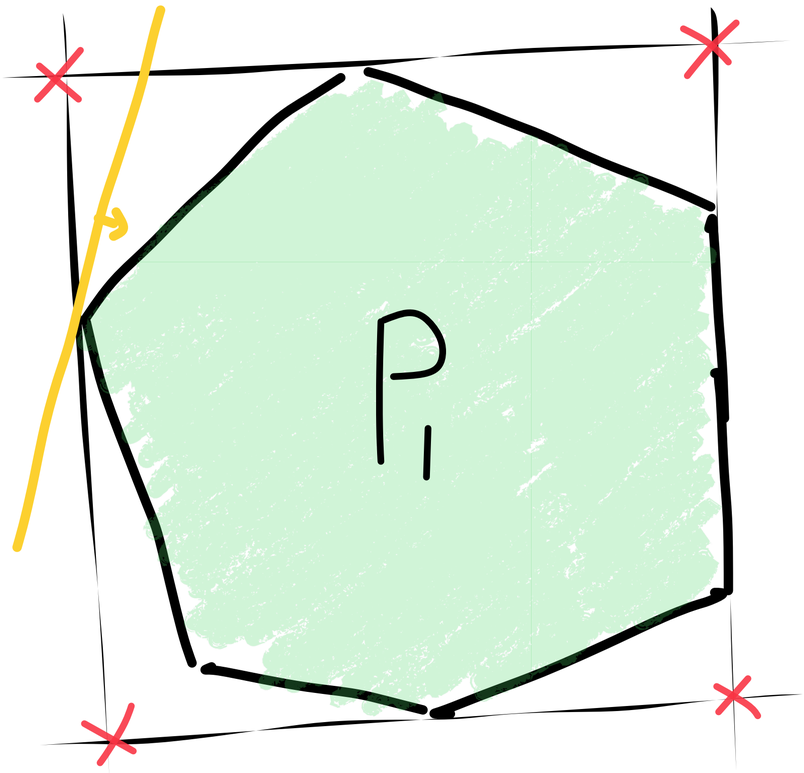
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

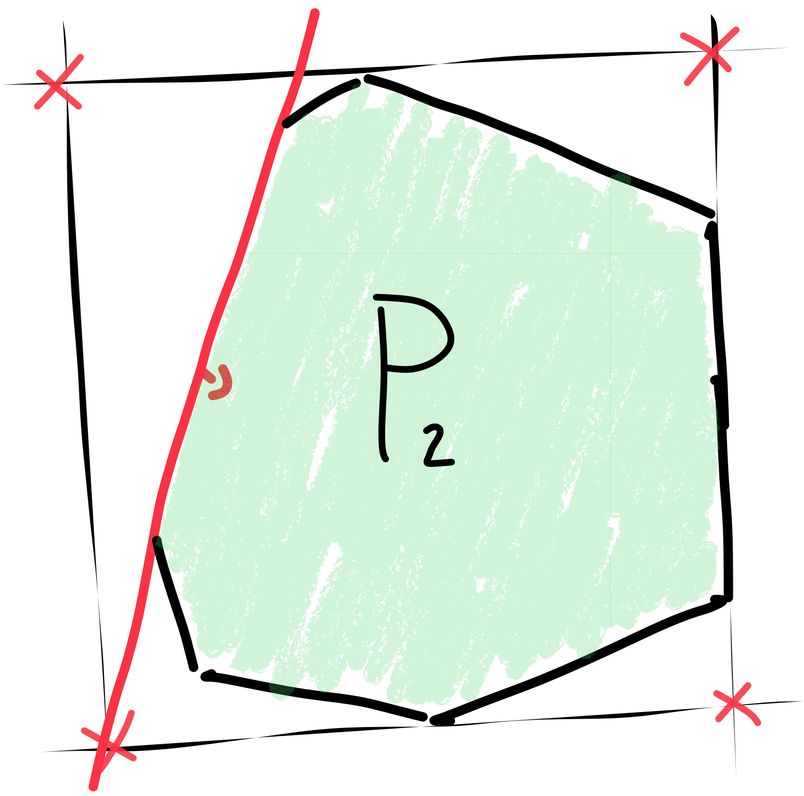
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

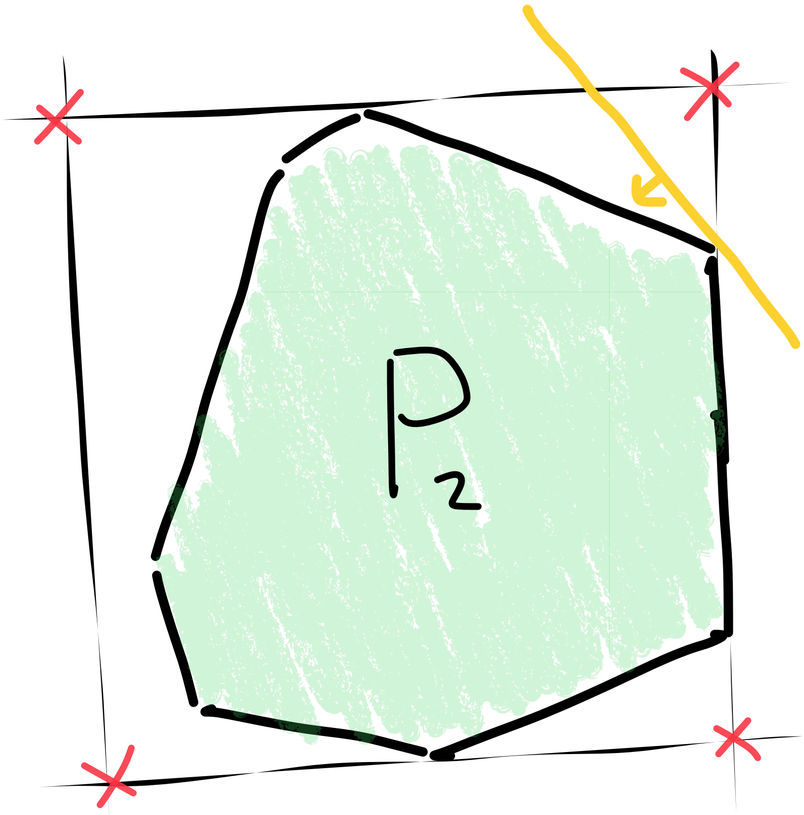
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

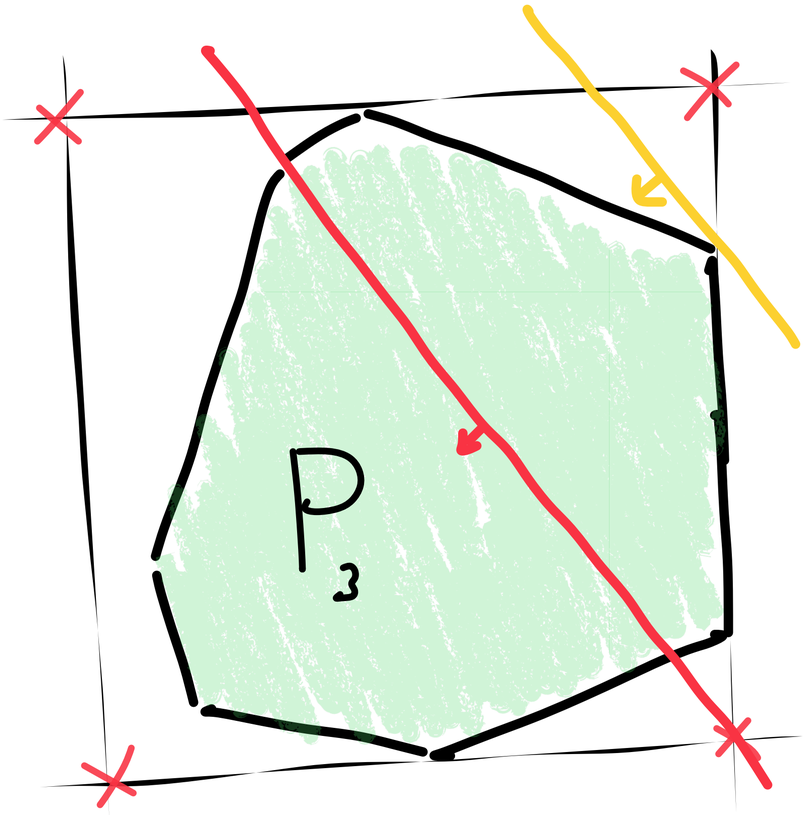
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

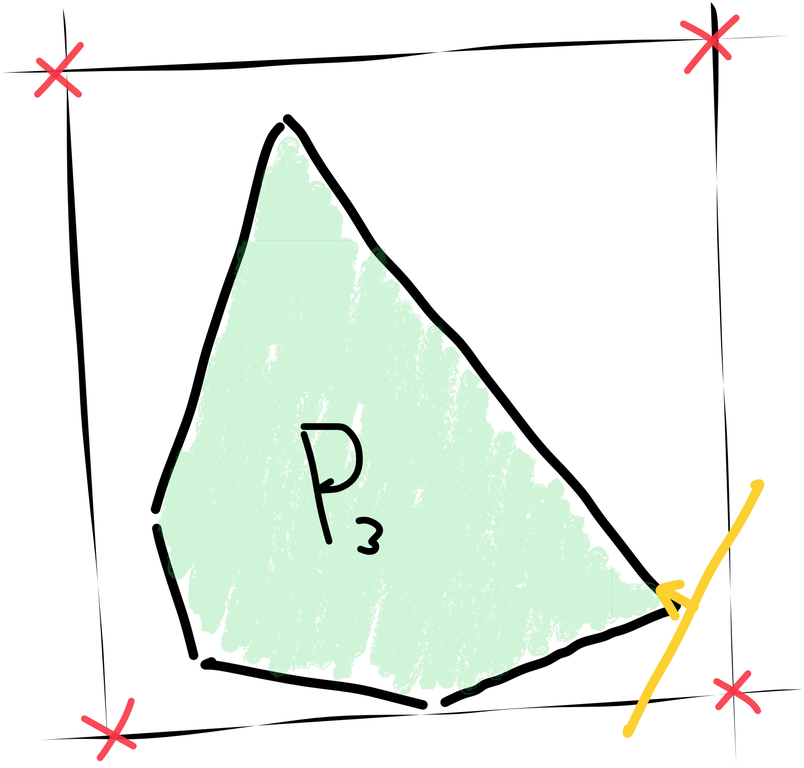
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

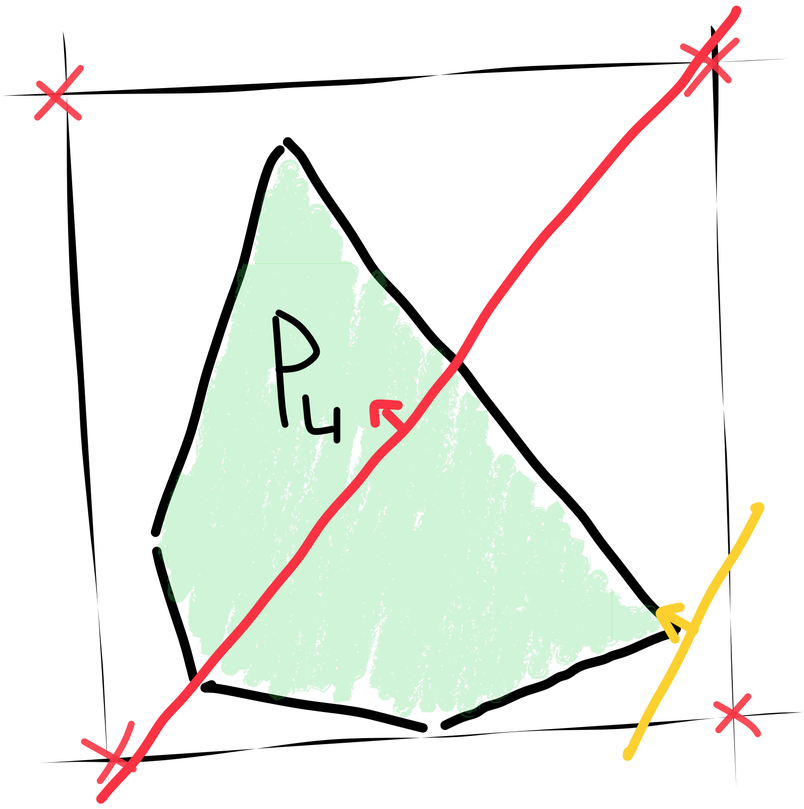
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

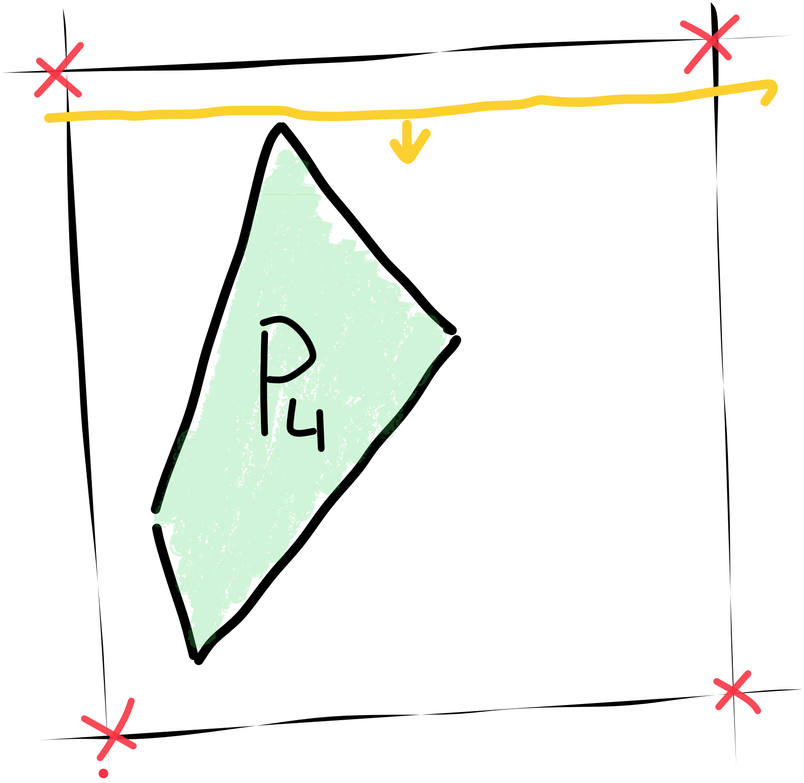
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

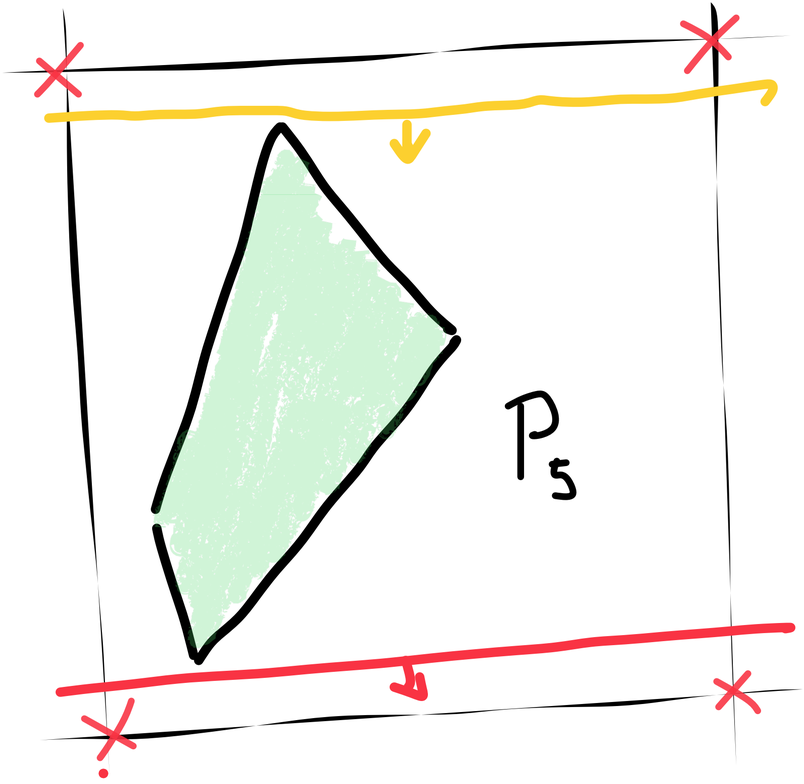
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

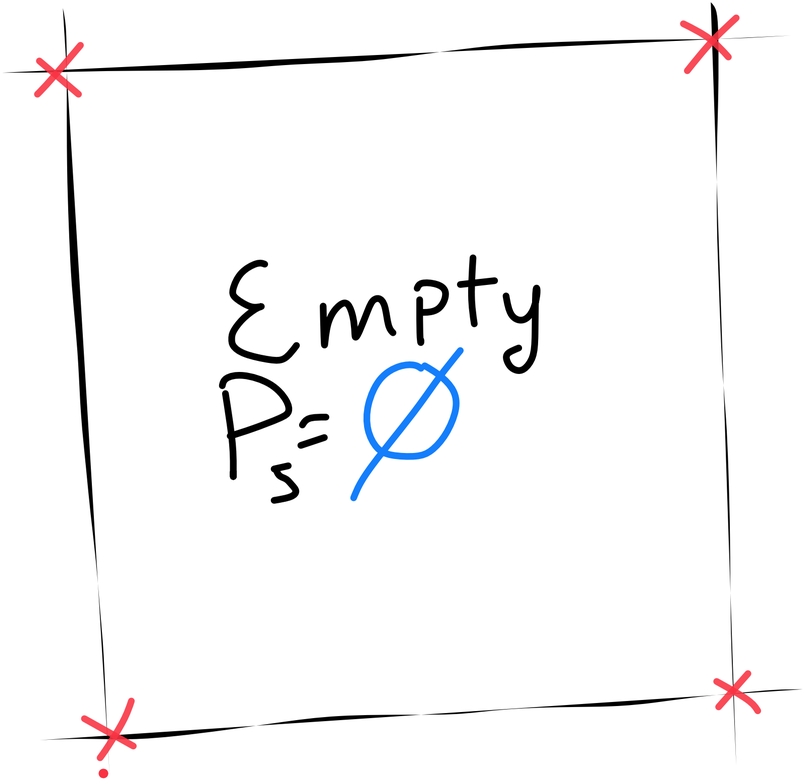
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut



Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut

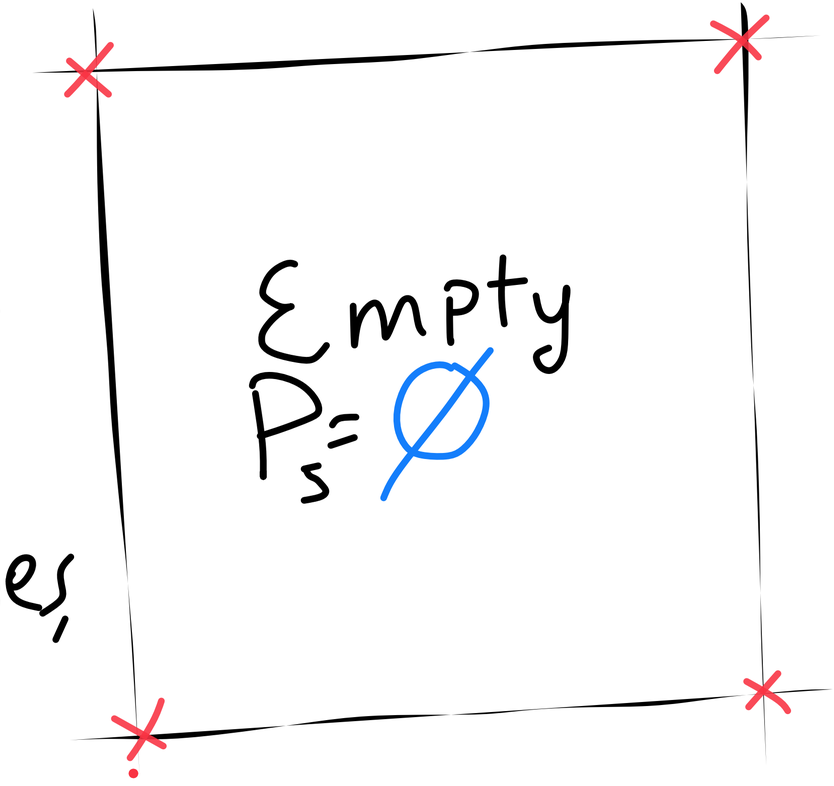


Cutting Planes

Let $P = \{Ax \geq b\}$ be such that $P \cap \mathbb{Z}^n = \emptyset$

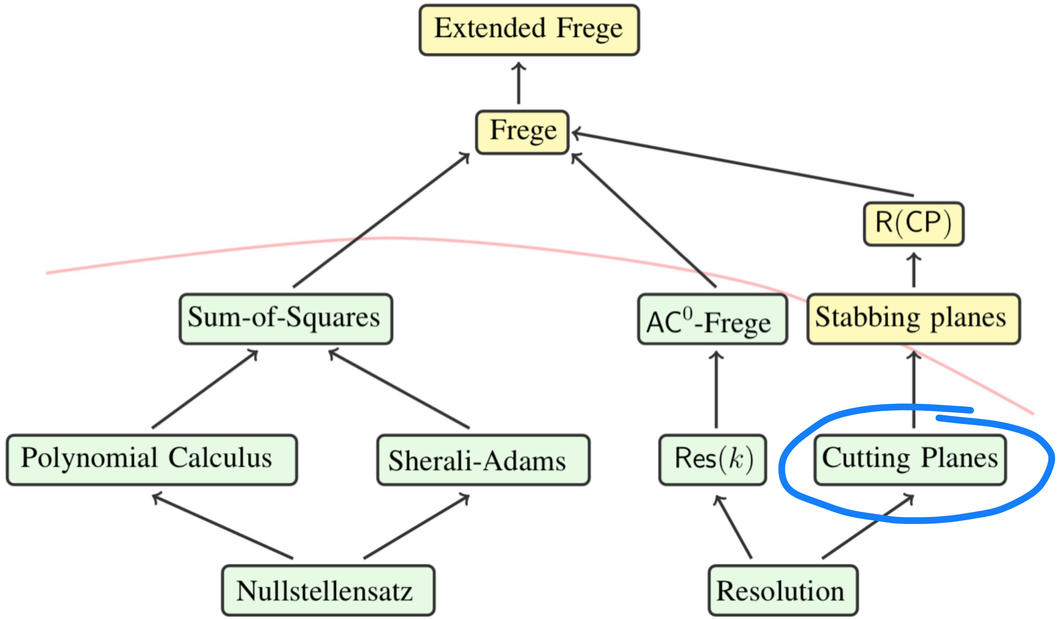
A CP proof that $P \cap \mathbb{Z}^n = \emptyset$ is a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ s.t. P_{i+1} is deduced from P_i by a CG-cut

Proof Size: the number of polytopes, s



Cutting Planes

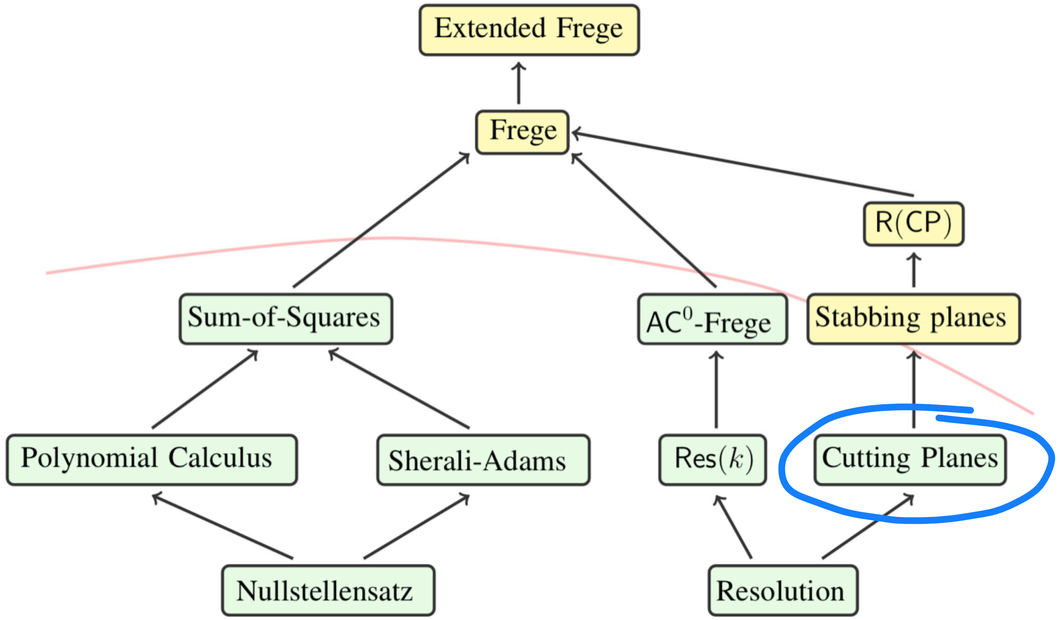
Powerful & Well-studied algebraic proof system



Cutting Planes

Powerful & Well-studied algebraic proof system

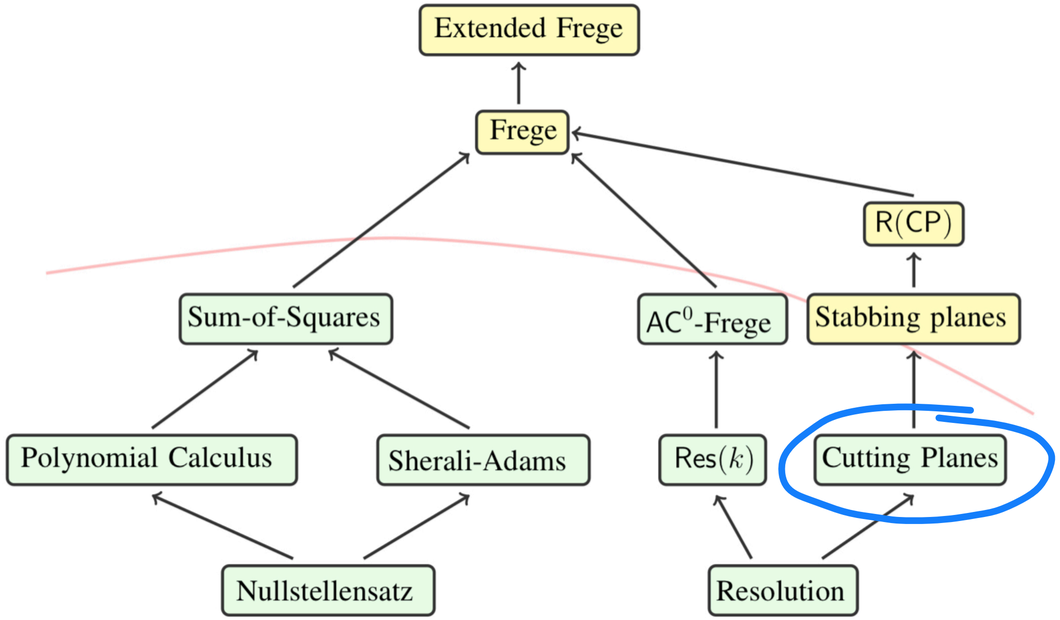
- Short proofs of PHP



Cutting Planes

Powerful & Well-studied algebraic proof system

- Short proofs of PHP
- Exponential lowerbounds [Pud97]



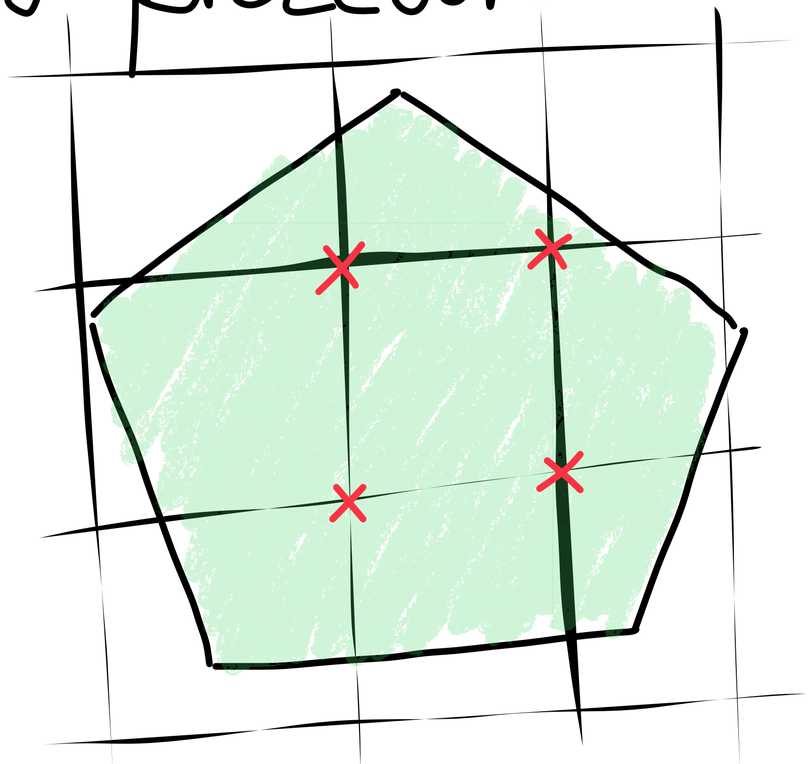
Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut on input P,

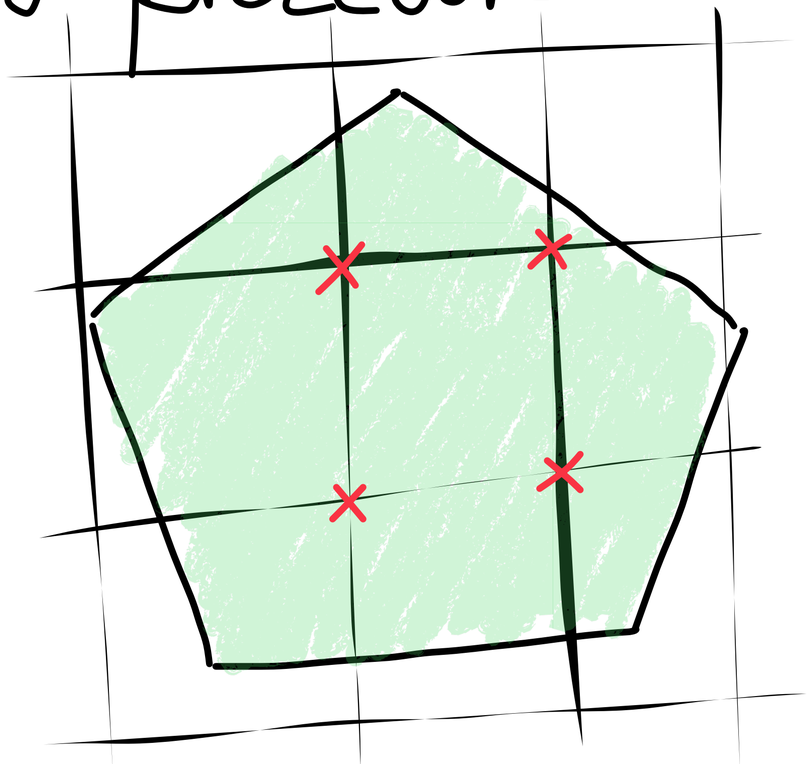


Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut on input P,

Branch: break P into $P_1, \ldots, P_k$ s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \ldots \cup P_k$

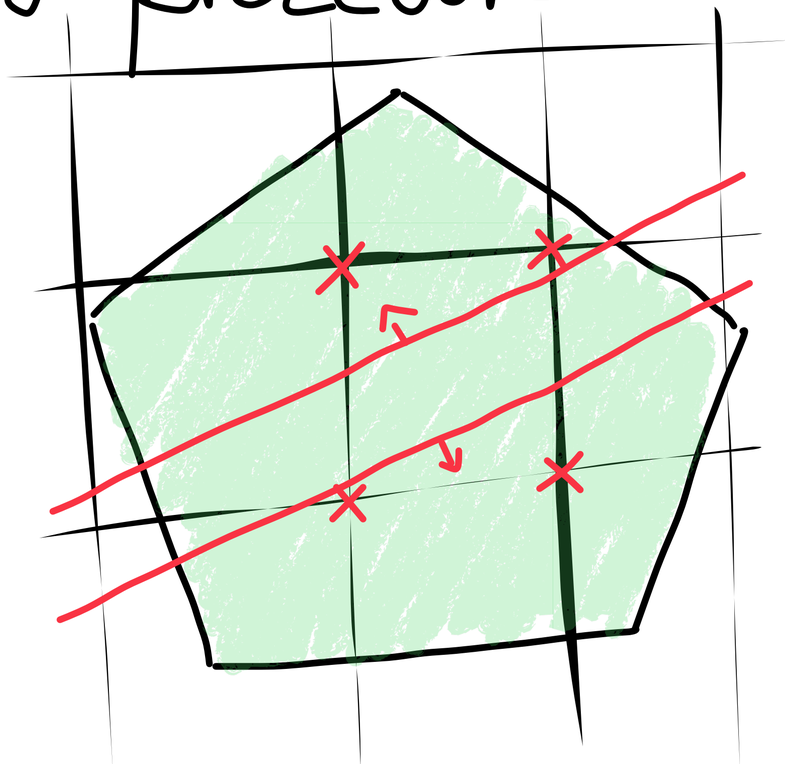


Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut On input P,

Branch: break P into $P_1, \ldots, P_k$ s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \ldots \cup P_k$

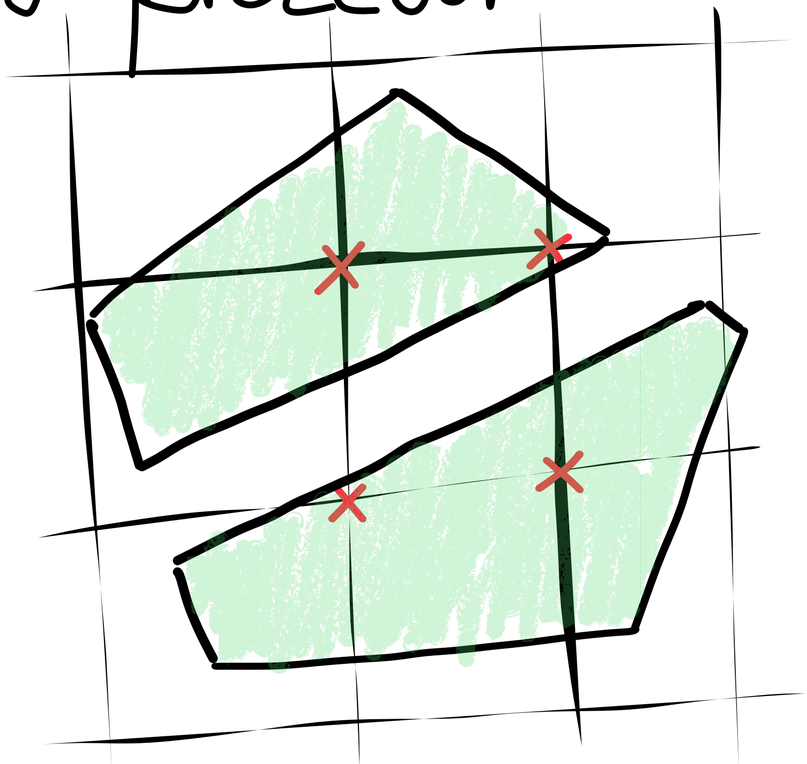


Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut On input P,

Branch: break P into $P_1, \ldots, P_k$
s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \ldots \cup P_k$



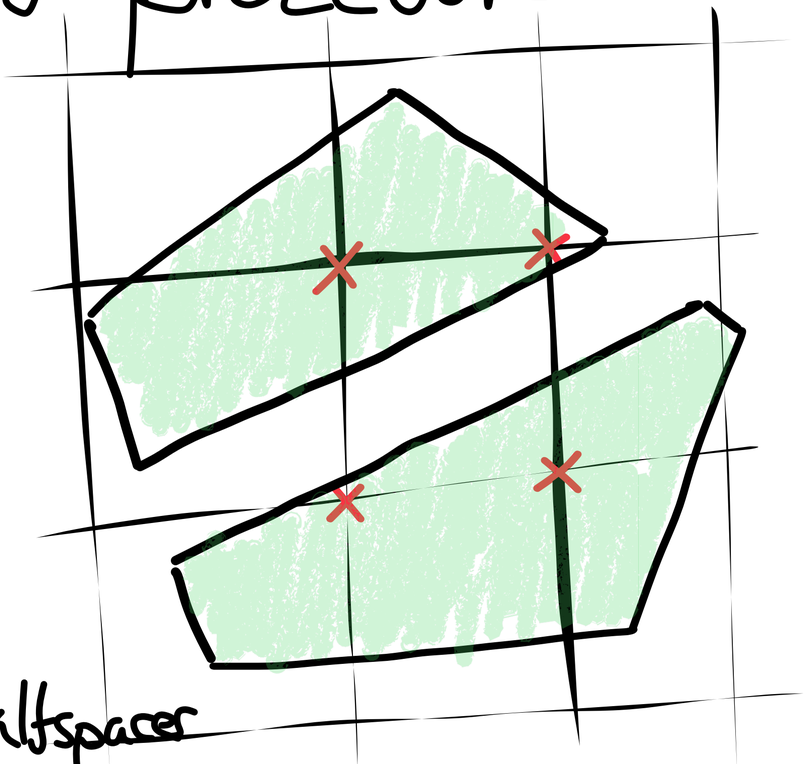
Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut on input P,

Branch: break P into $P_1, \dots, P_k$ s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \dots \cup P_k$

- In practice, P is broken into $P \cap \{x : ax \geq b\}$ and $P \cap \{x : ax \leq b-1\}$ for some class of halfspaces



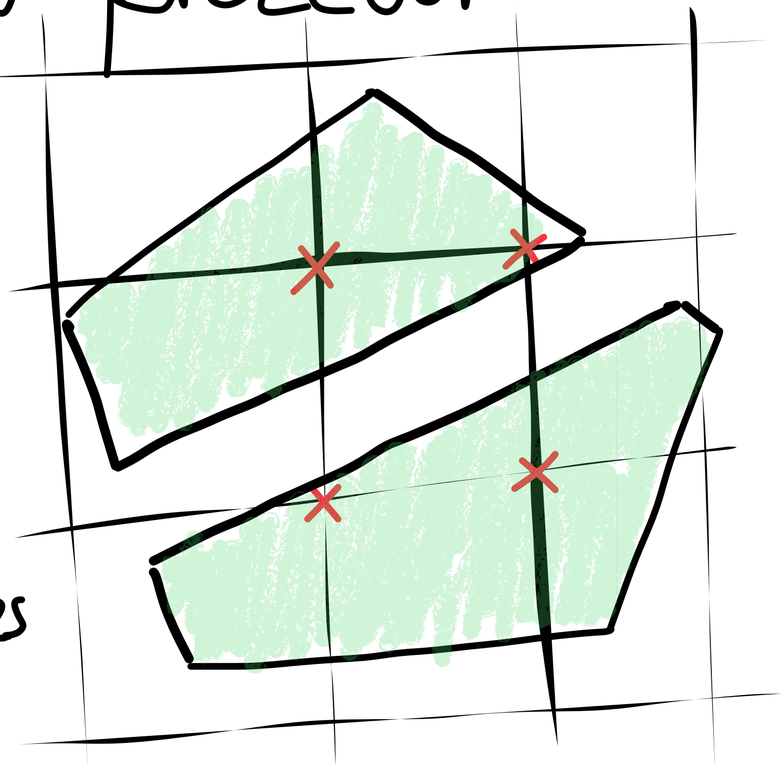
Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut on input P,

Branch: break P into $P_1, \dots, P_k$ s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \dots \cup P_k$

Cut: Refine $P_1, \dots, P_k$ with additional cutting planes



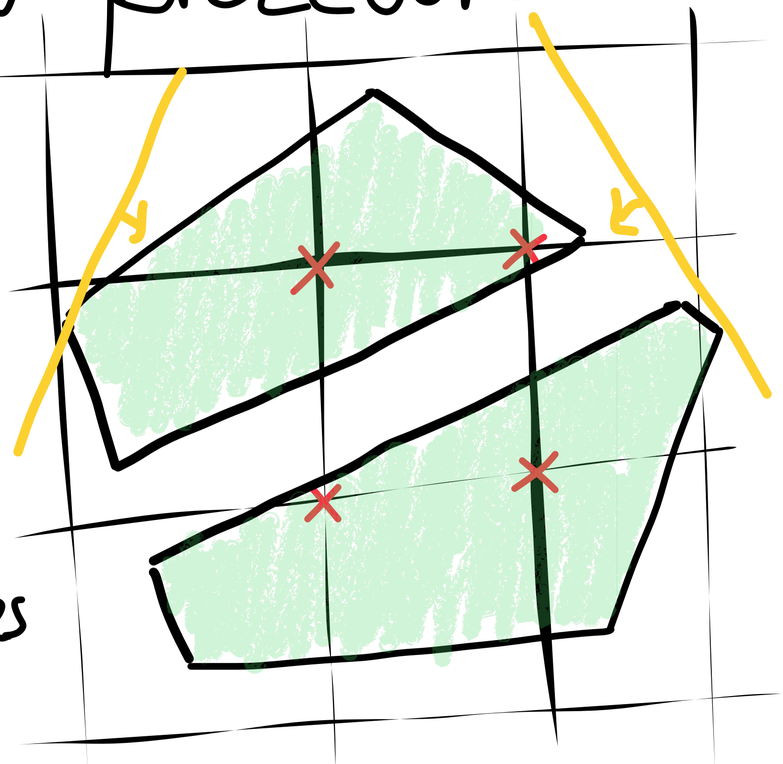
Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut on input P,

Branch: break P into $P_1, \dots, P_k$ s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \dots \cup P_k$

Cut: Refine $P_1, \dots, P_k$ with additional cutting planes



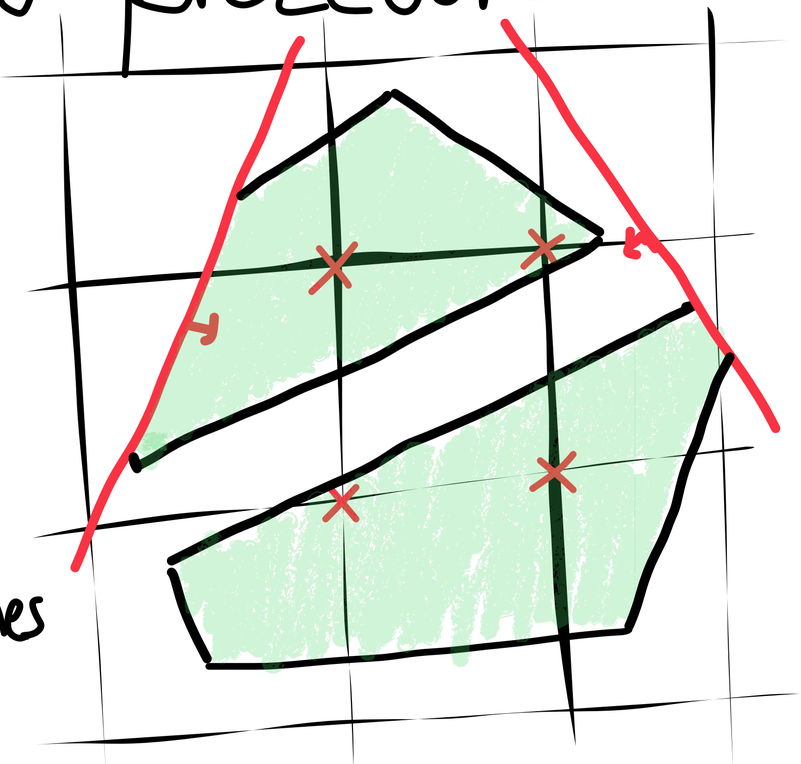
Integer Programming

Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut On input P,

Branch: break P into $P_1, \dots, P_k$ s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \dots \cup P_k$

Cut: Refine $P_1, \dots, P_k$ with additional cutting planes



Integer Programming

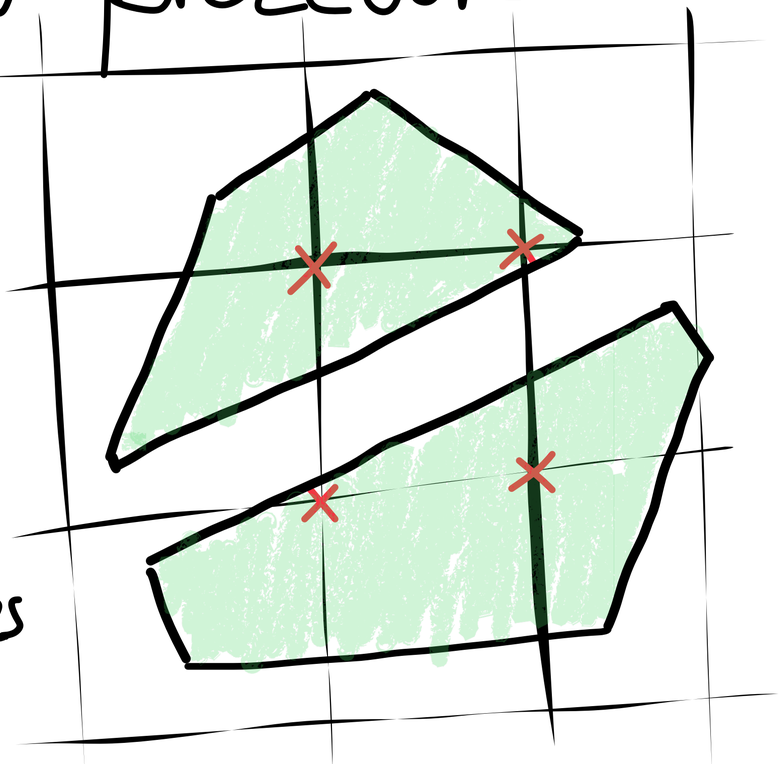
Modern IP Algorithms combine cutting planes with a branch-and-bound procedure

Branch-and-Cut On input P,

Branch: break P into $P_1, \ldots, P_k$ s.t. $P \cap \mathbb{Z}^n \subseteq P_1 \cup \ldots \cup P_k$

Cut: Refine $P_1, \ldots, P_k$ with additional cutting planes

Repeat



Stabbing Planes [BFI+18]

- Formalizes practical branch-and-cut as a proof system

Stabbing Planes [BFI+18]

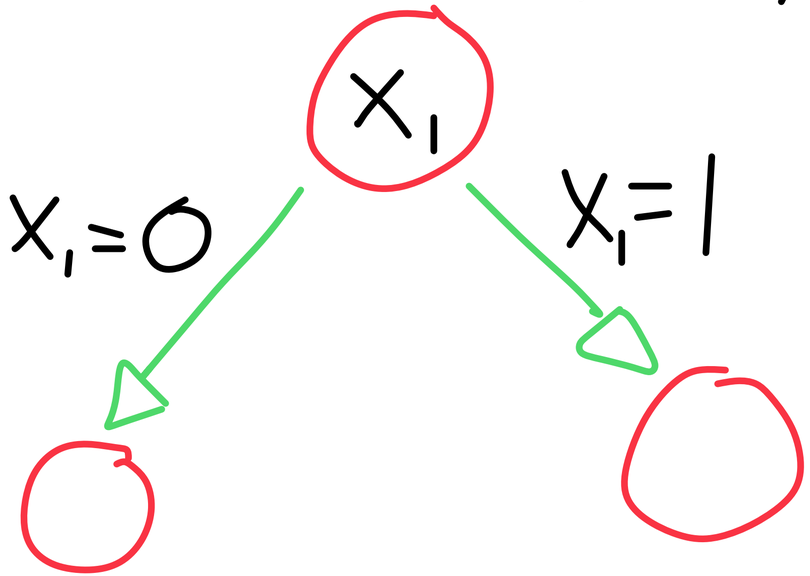
- Formalizes practical branch-and-cut as a proof system
- Extends DPLL to reason about linear inequalities

DPLL Refutation

$$\{x_1 \vee x_2,\ \bar{x}_1 \vee x_2,\ x_1 \vee \bar{x}_2,\ \bar{x}_1 \vee \bar{x}_2\}$$

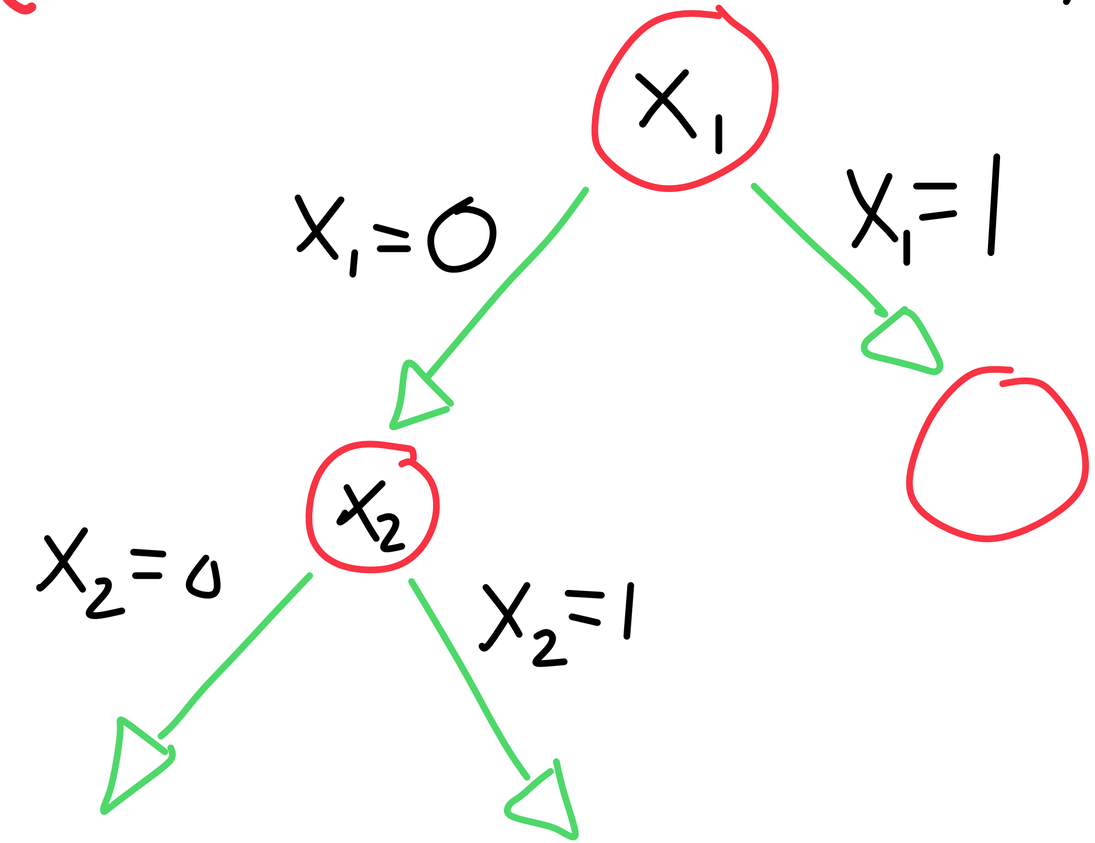
DPLL Refutation

$$\{x_1 \vee x_2,\ \bar{x}_1 \vee x_2,\ x_1 \vee \bar{x}_2,\ \bar{x}_1 \vee \bar{x}_2\}$$



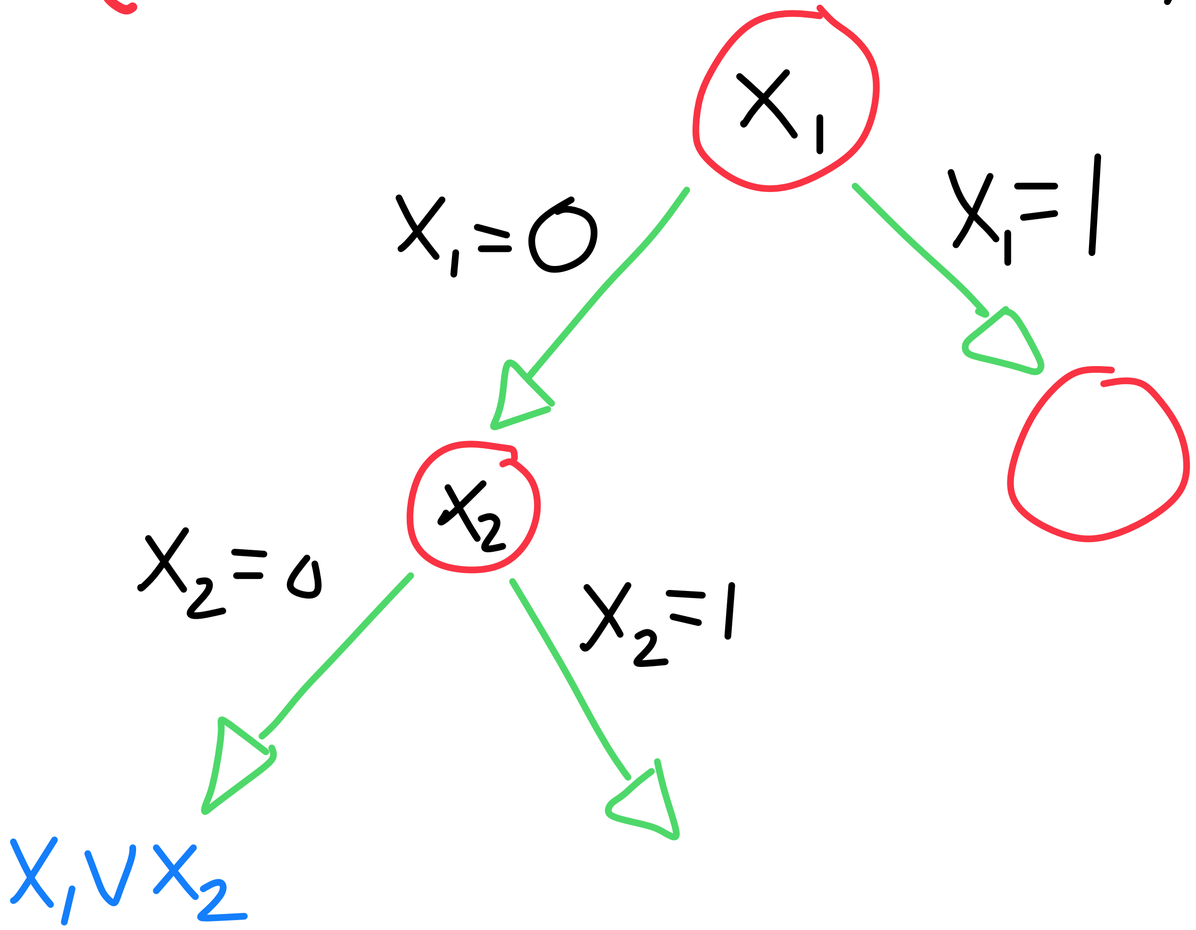
DPLL Refutation

$\{x_1 \vee x_2,\ \bar{x}_1 \vee x_2,\ x_1 \vee \bar{x}_2,\ \bar{x}_1 \vee \bar{x}_2\}$



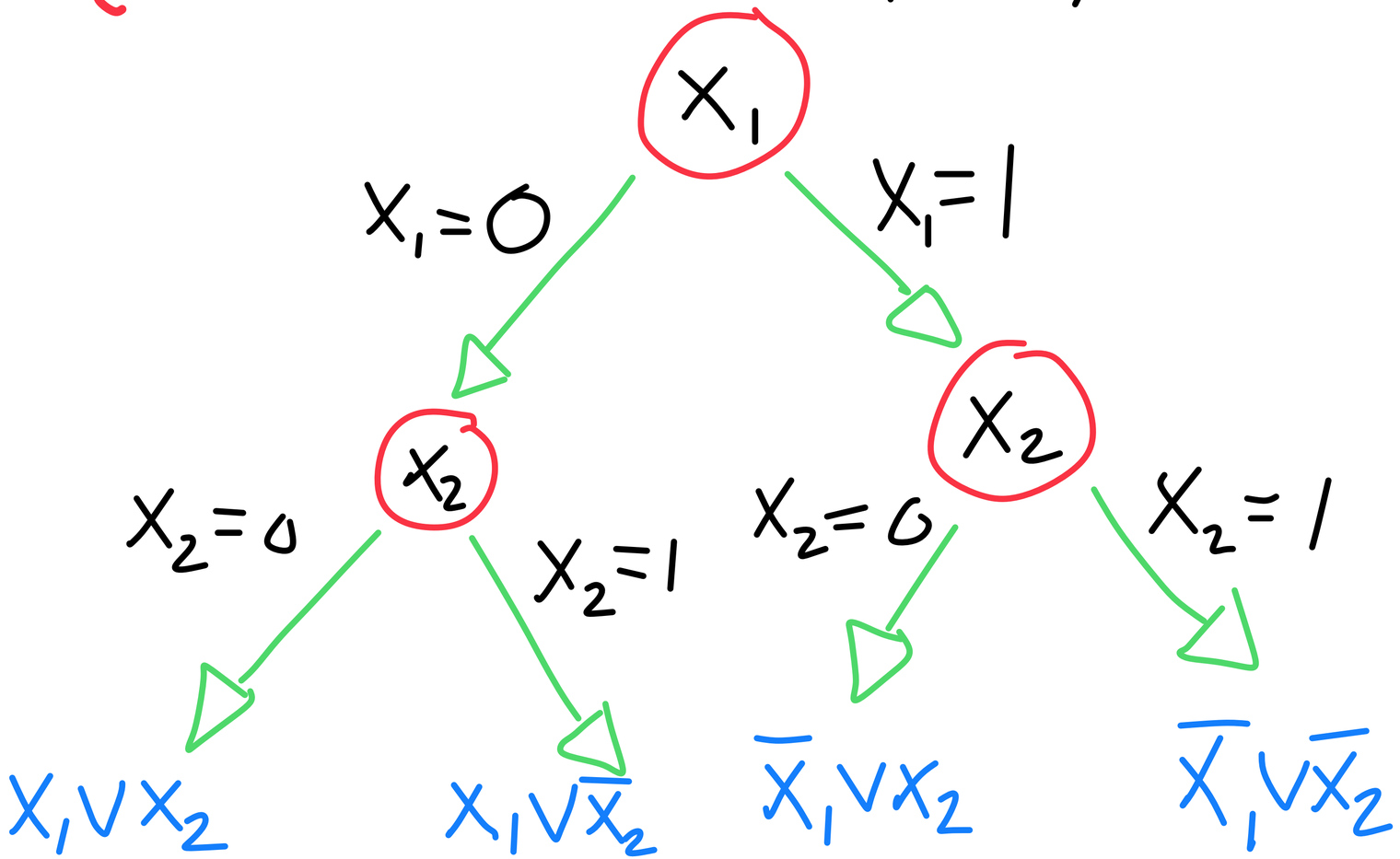
DPLL Refutation

$\{x_1 \vee x_2,\ \bar{x}_1 \vee x_2,\ x_1 \vee \bar{x}_2,\ \bar{x}_1 \vee \bar{x}_2\}$



DPLL Refutation

$\{x_1 \vee x_2,\ \bar{x}_1 \vee x_2,\ x_1 \vee \bar{x}_2,\ \bar{x}_1 \vee \bar{x}_2\}$



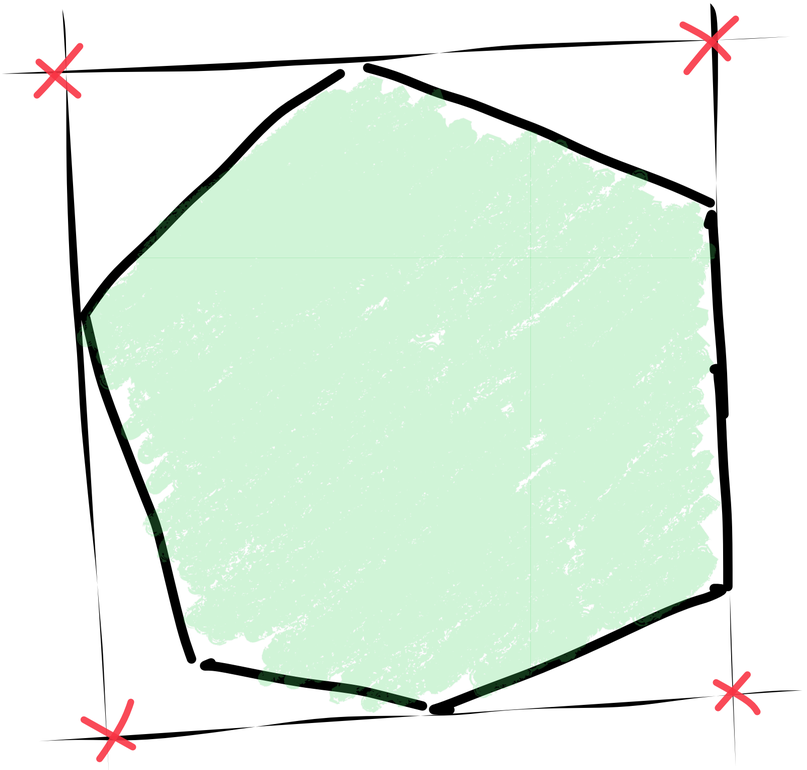
DPLL as Polytopes

$$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$$

DPLL as Polytopes

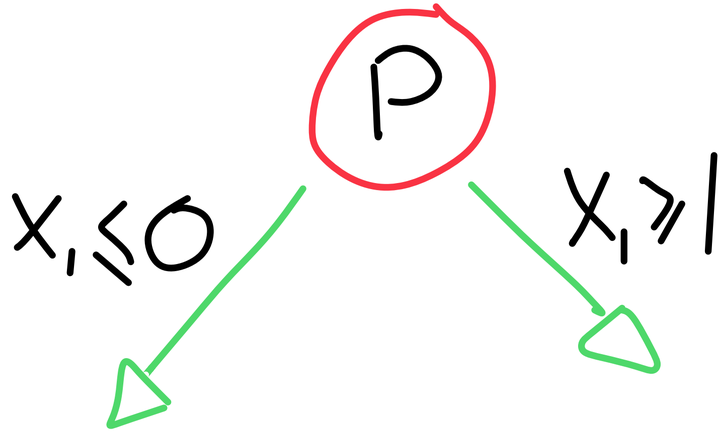
$$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$$

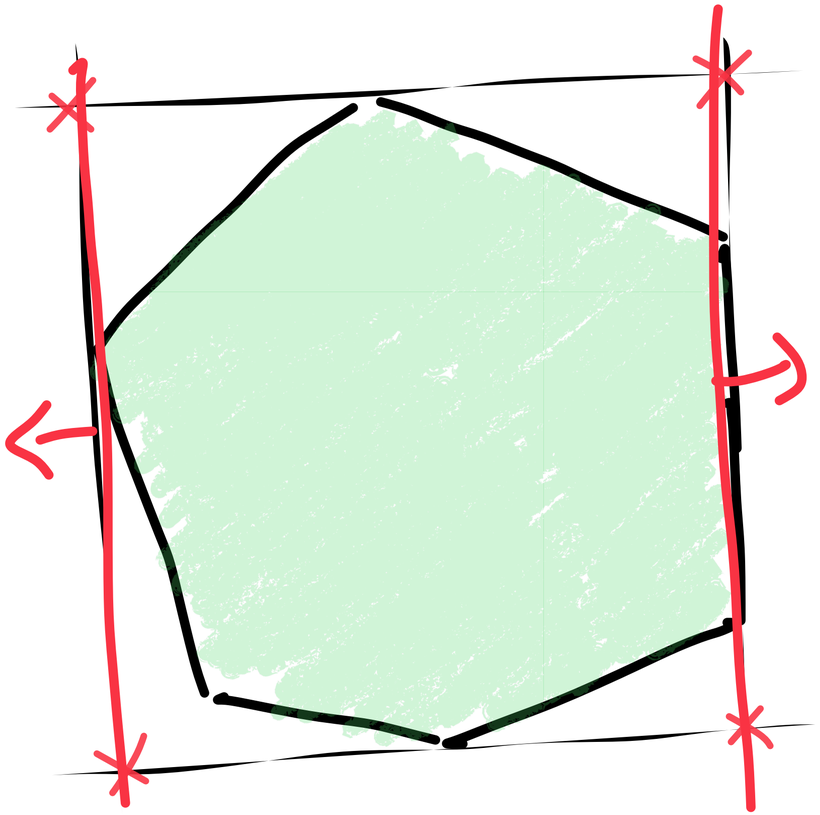
P



DPLL as Polytopes

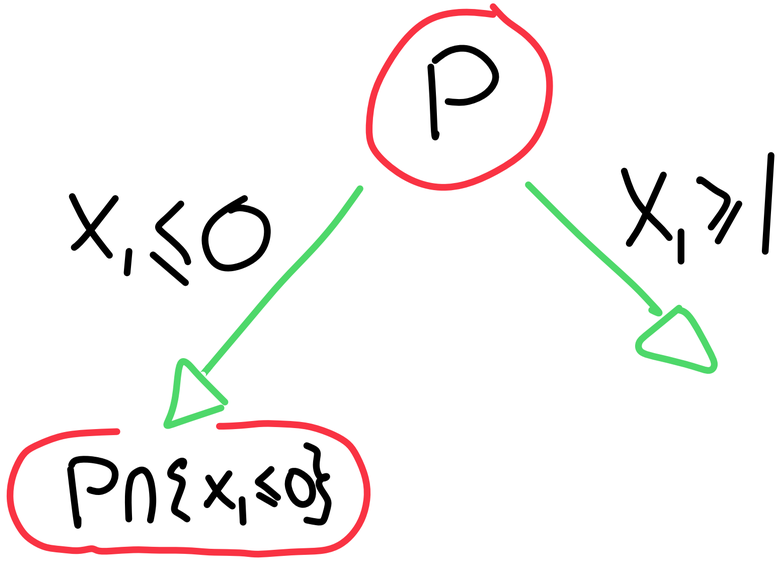
$$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$$

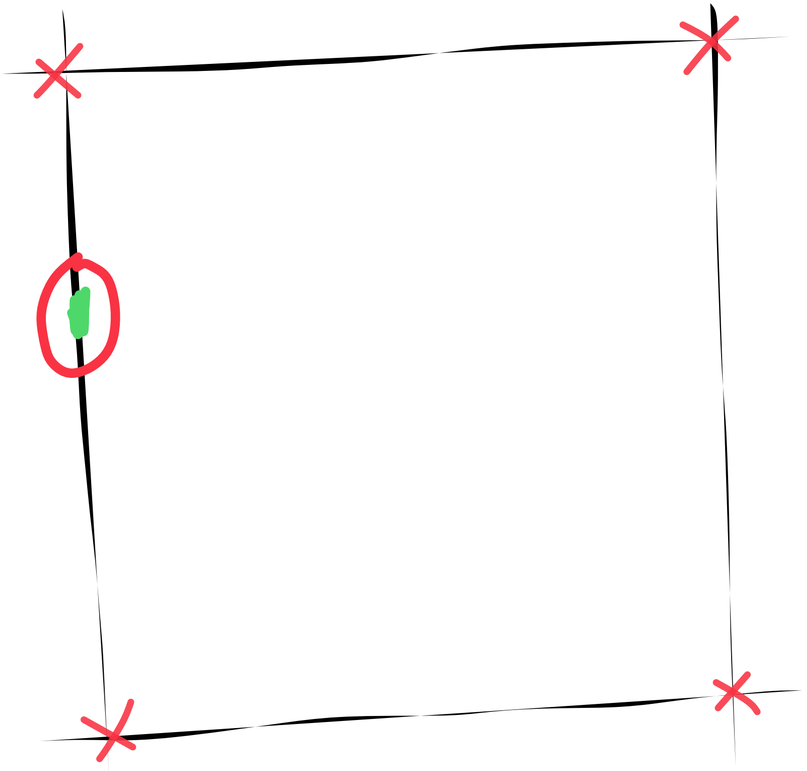




DPLL as Polytopes

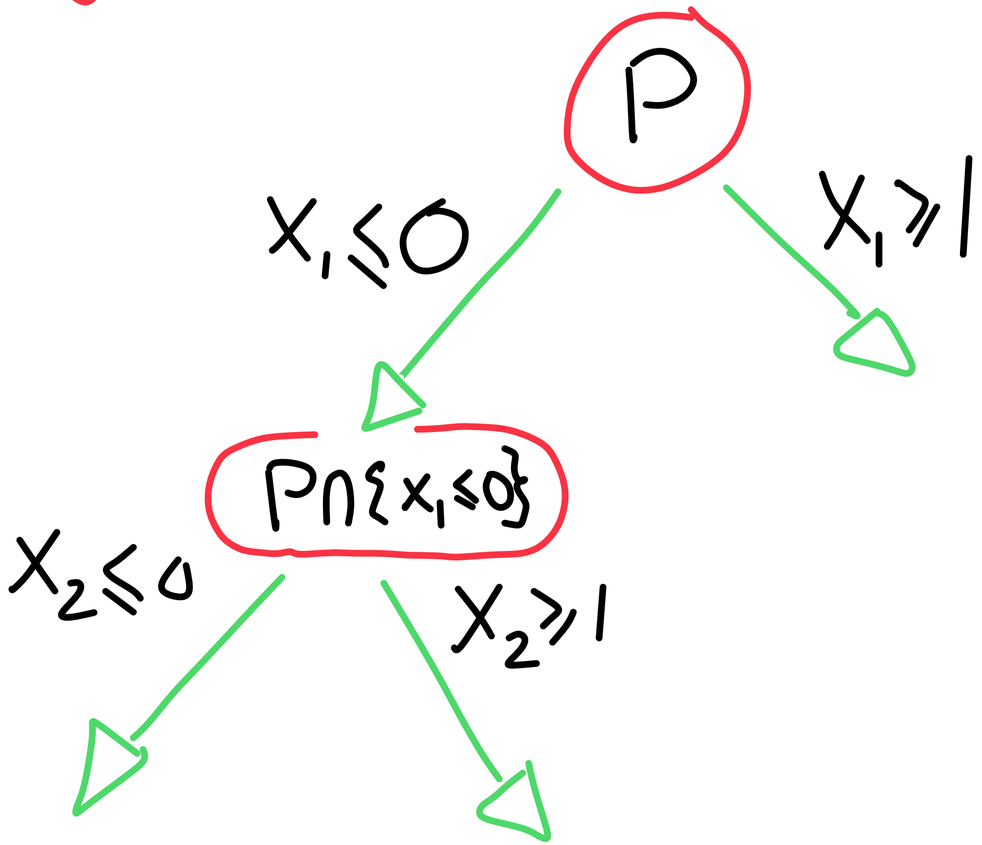
$$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$$

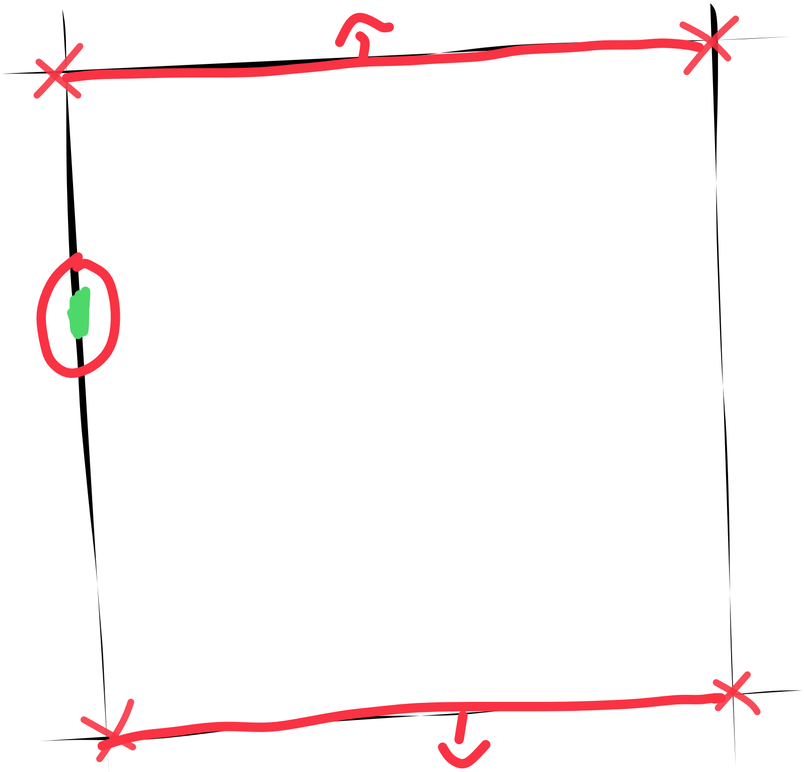




DPLL as Polytopes

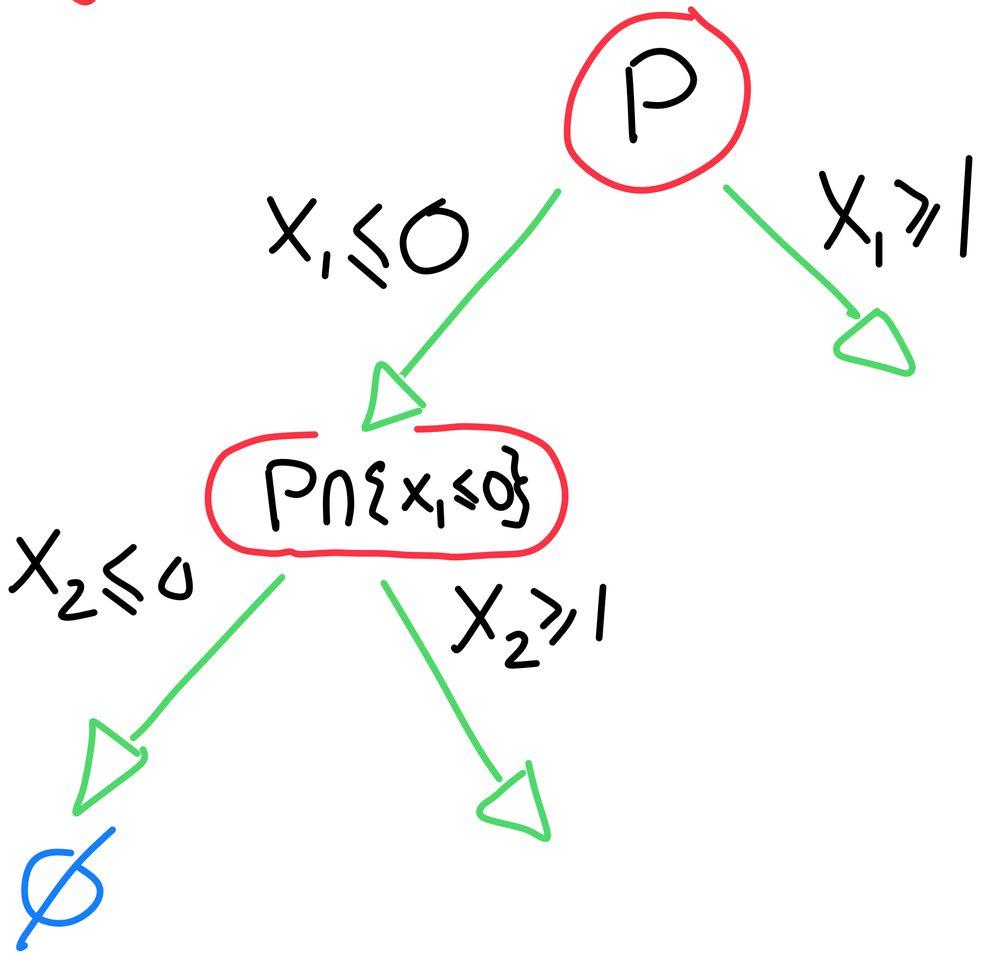
$$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$$

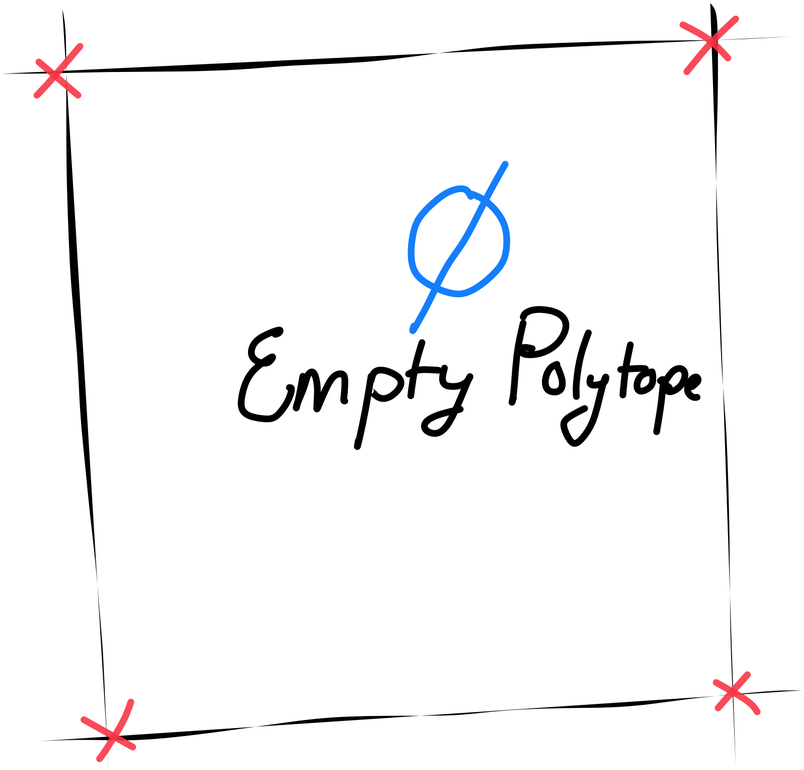




DPLL as Polytopes

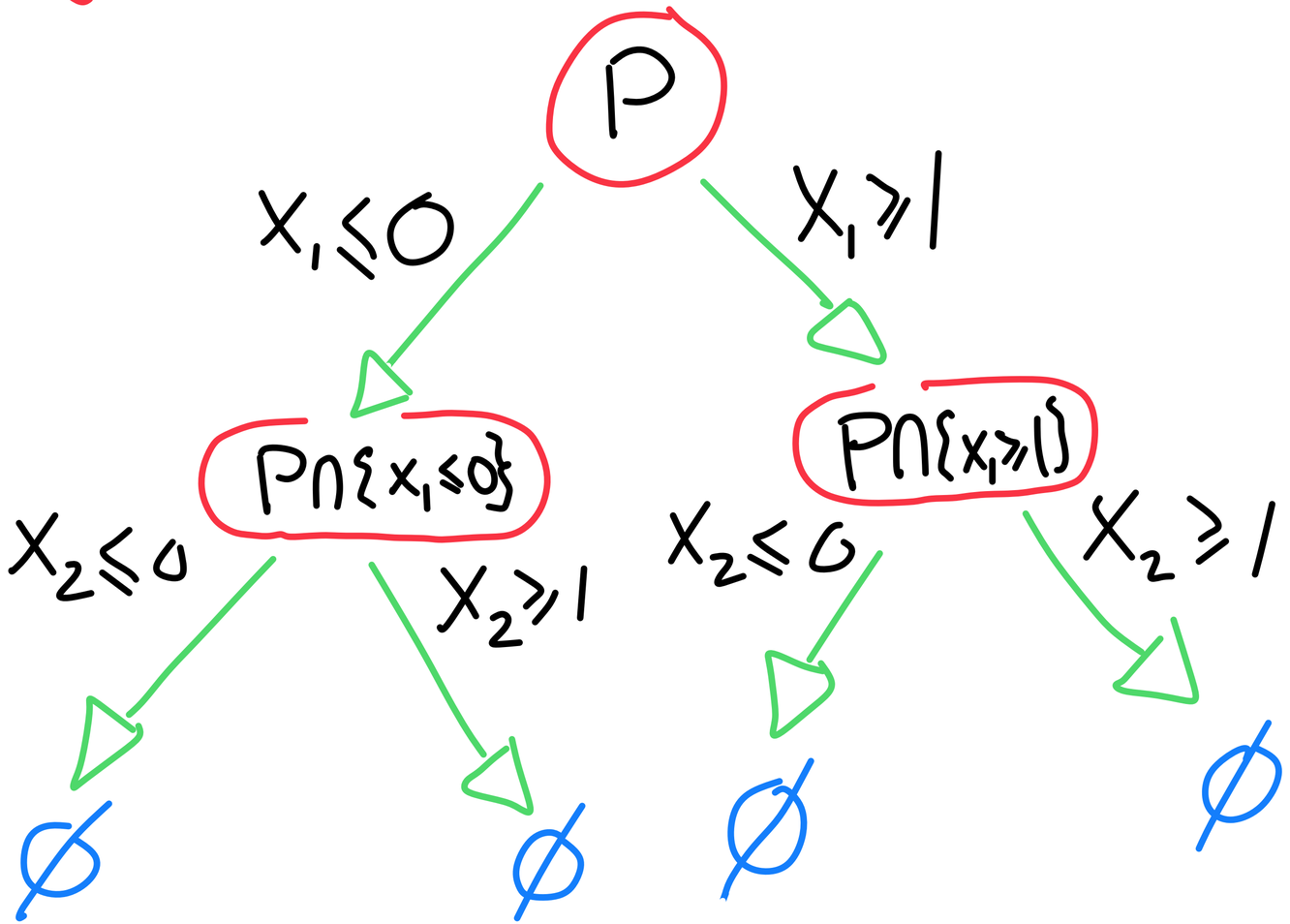
$$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$$

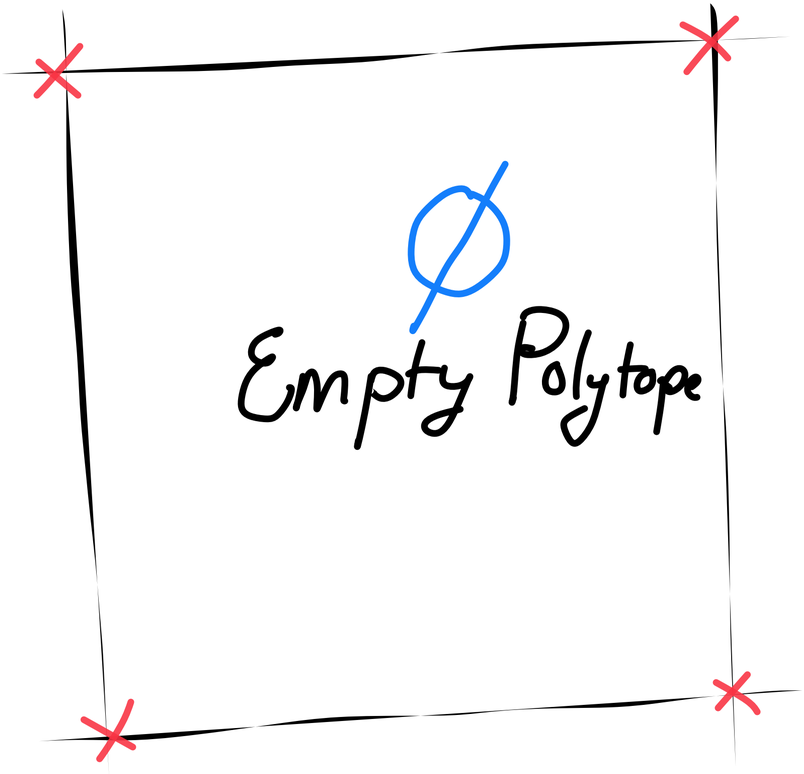




DPLL as Polytopes

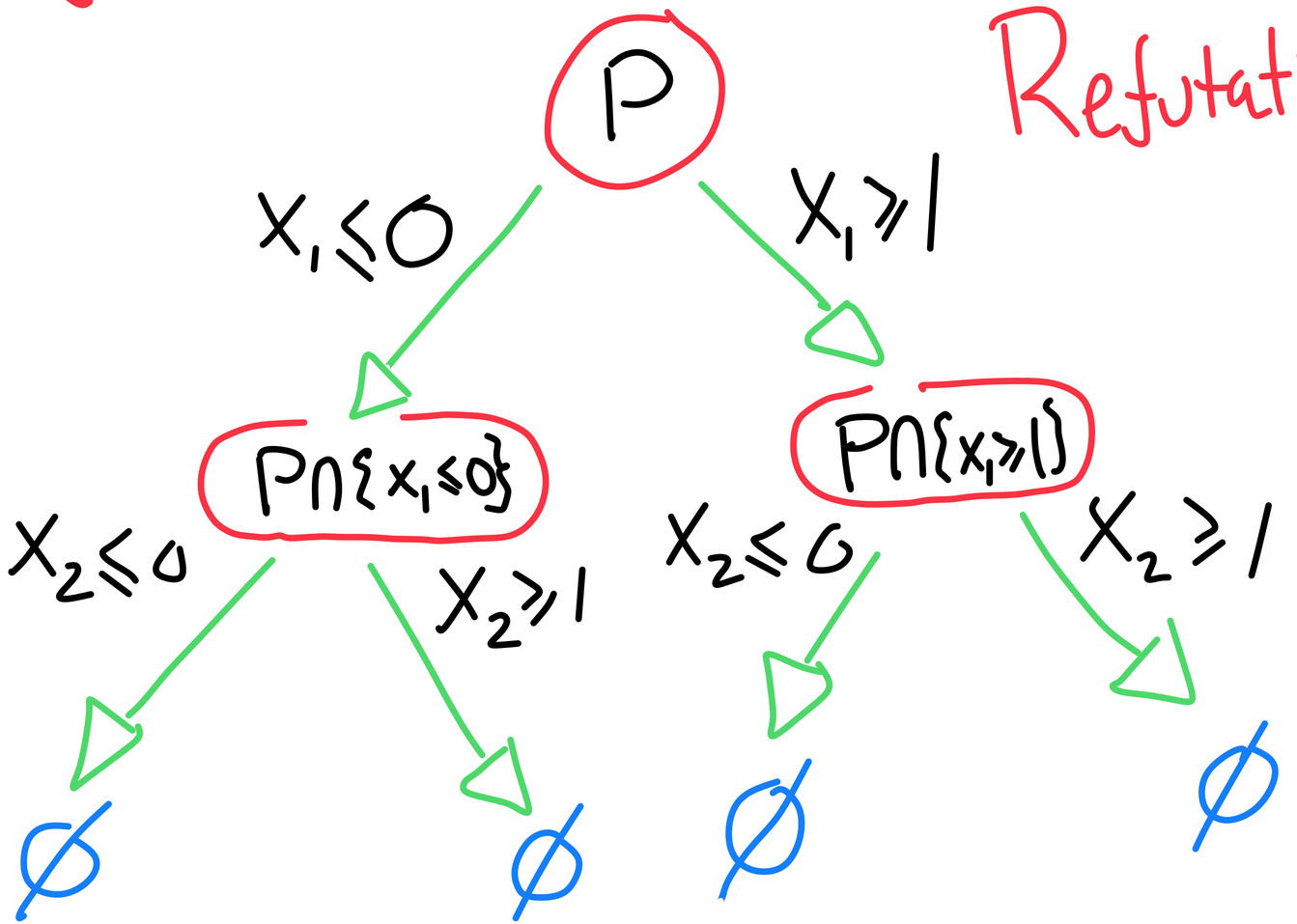
$$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$$



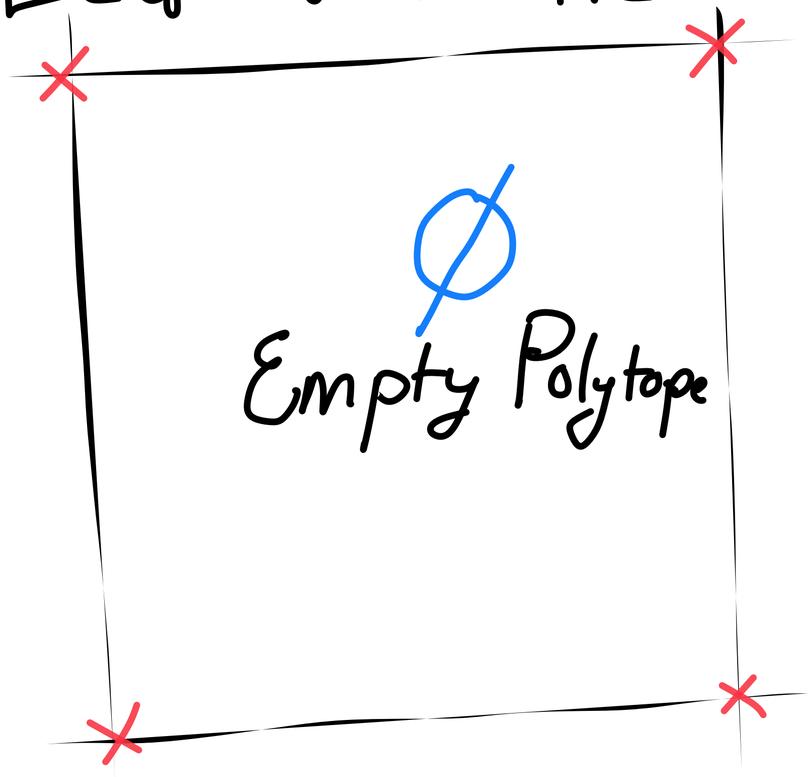


DPLL as Polytopes

$P = \{ x_1 + x_2 \geq 1,\ x_1 - x_2 \geq 0,\ x_2 - x_1 \geq 0,\ -x_1 - x_2 \geq -1,\ 0 \leq x_i \leq 1 \}$



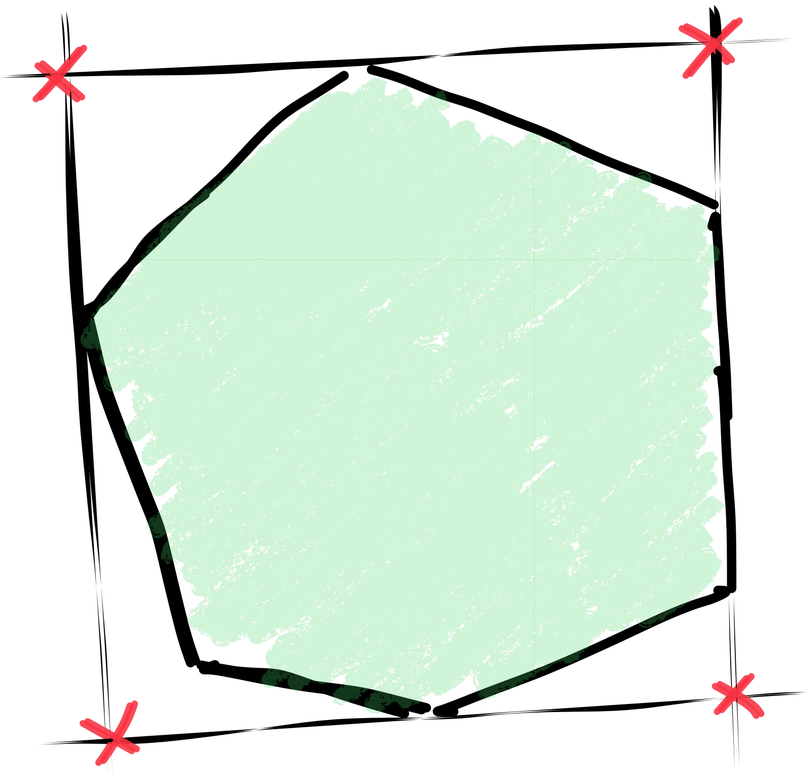
Refutation! $\emptyset$ Derived at every Leaf of the tree



Stabbing Planes

$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.

(P)

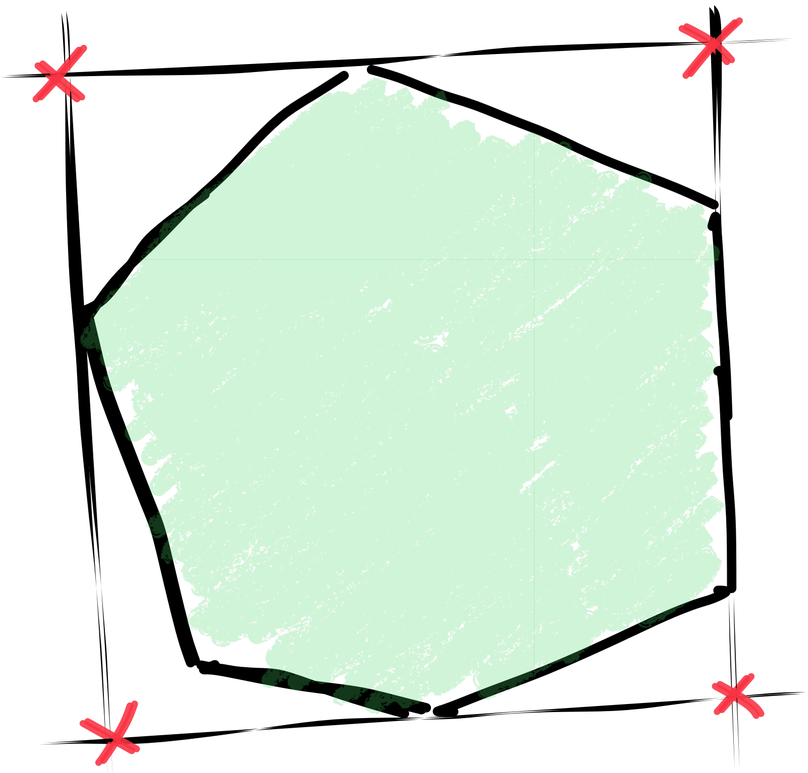


Stabbing Planes

$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.

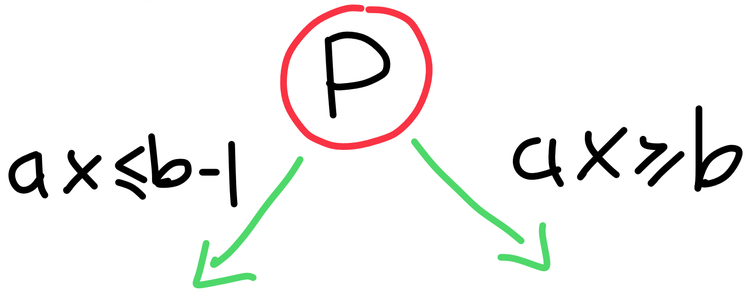
(P)

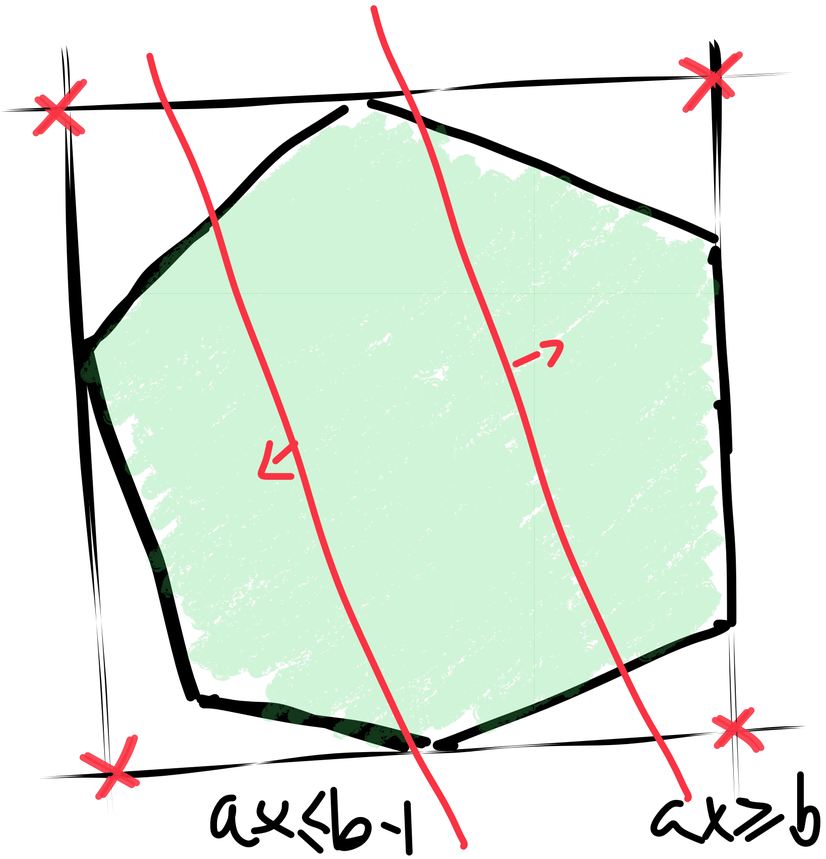
"query arbitrary linear inequalities"



Stabbing Planes

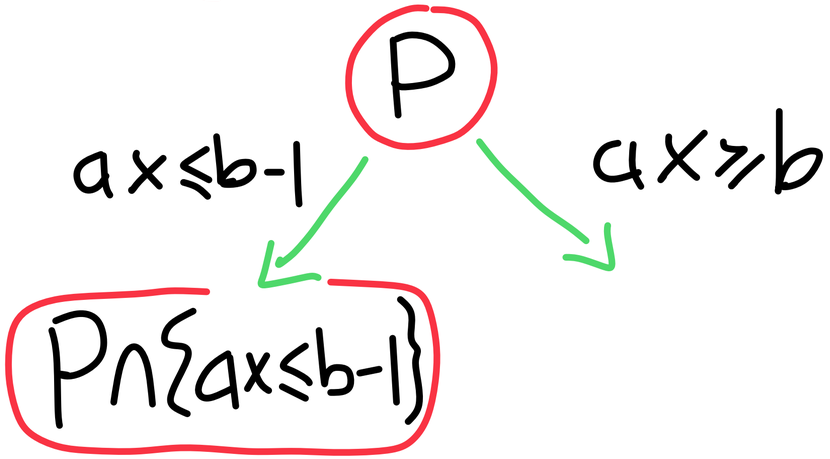
$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.

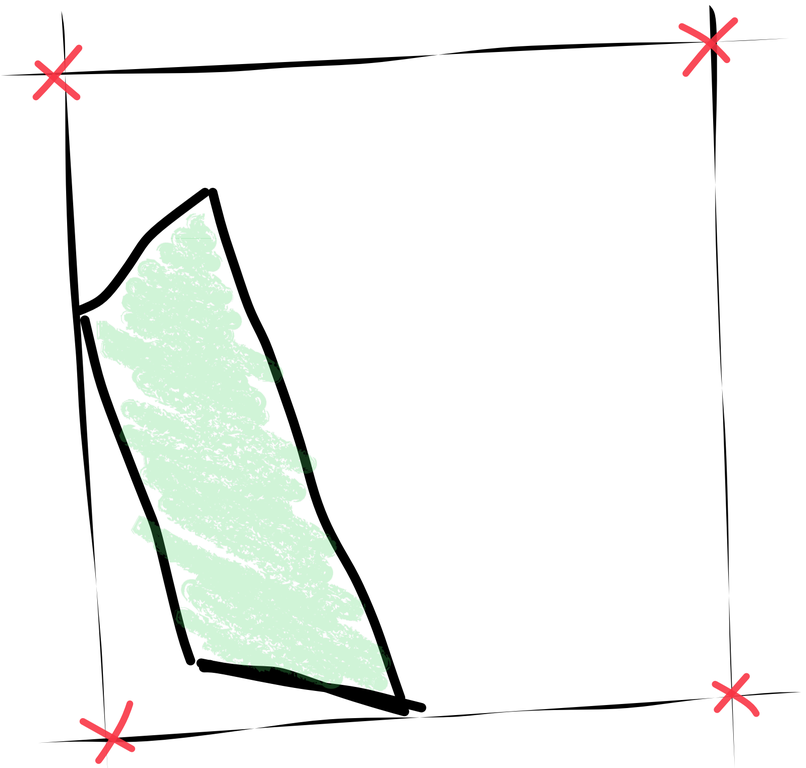




Stabbing Planes

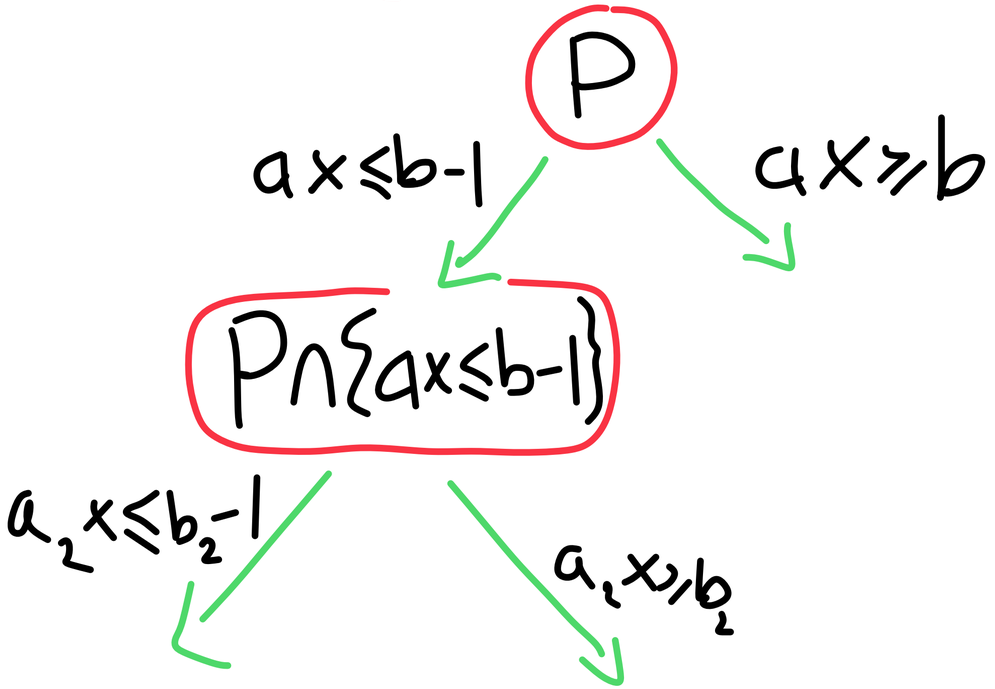
$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.

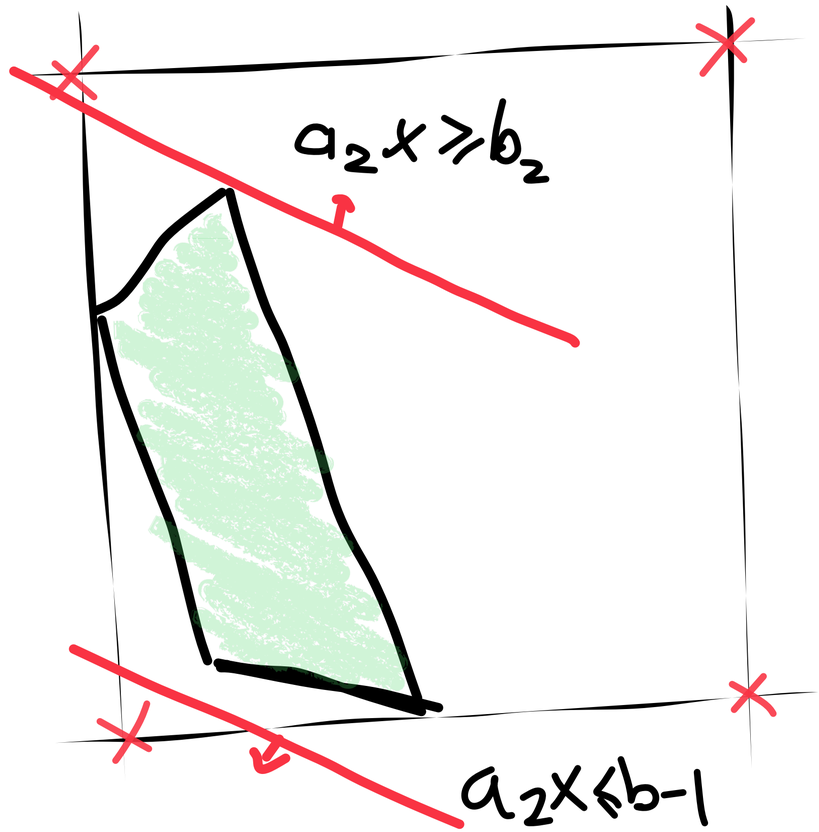




Stabbing Planes

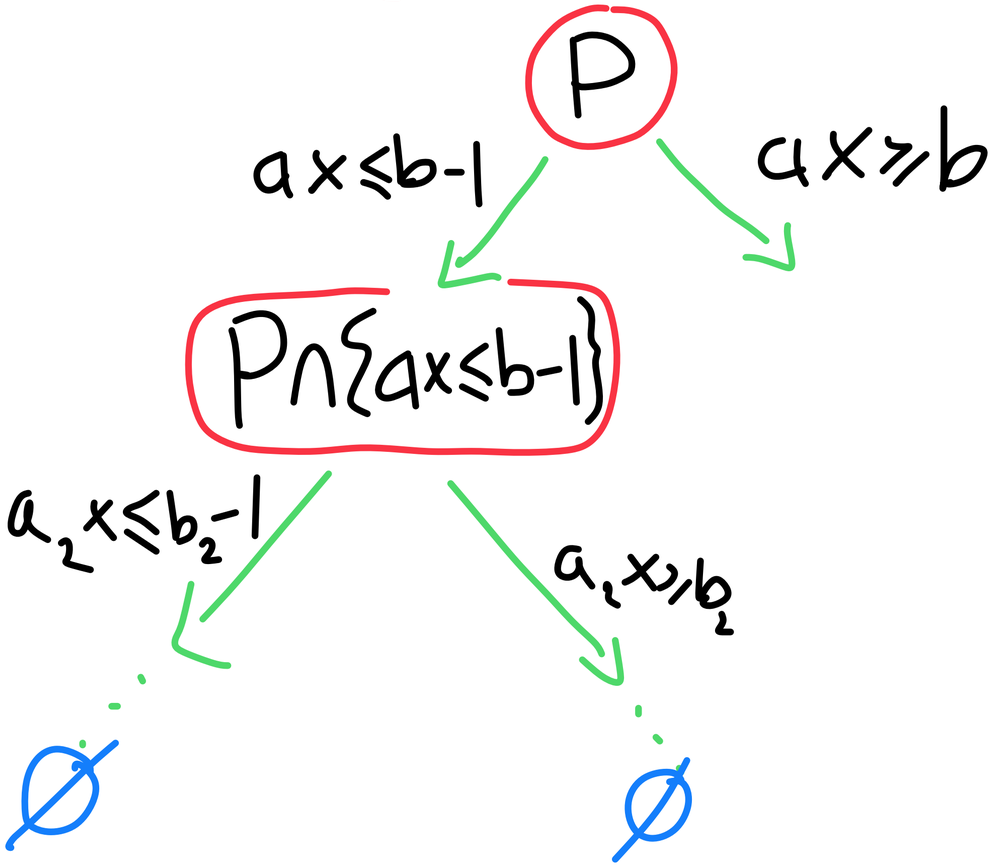
$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.

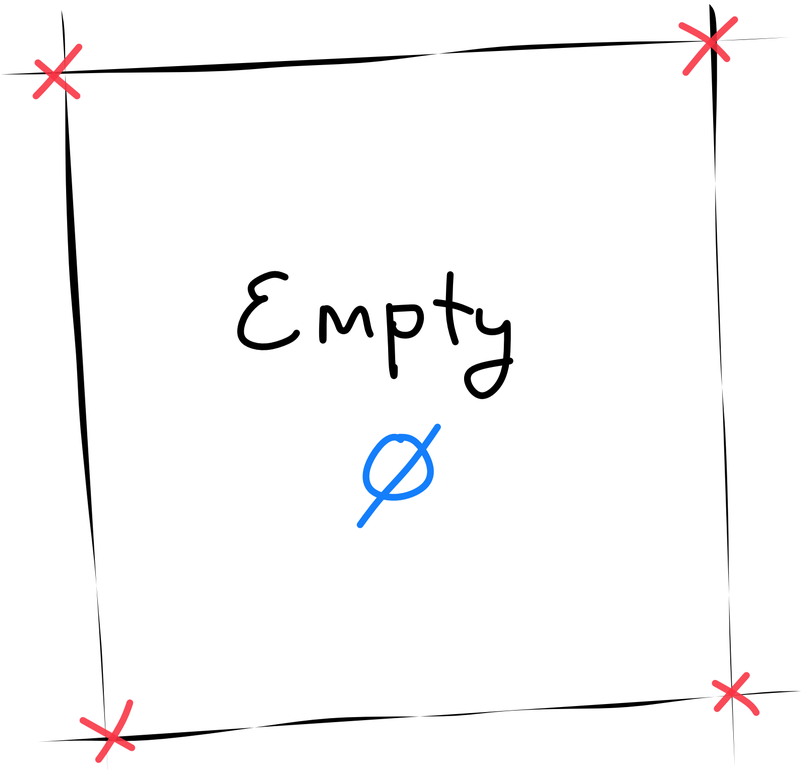




Stabbing Planes

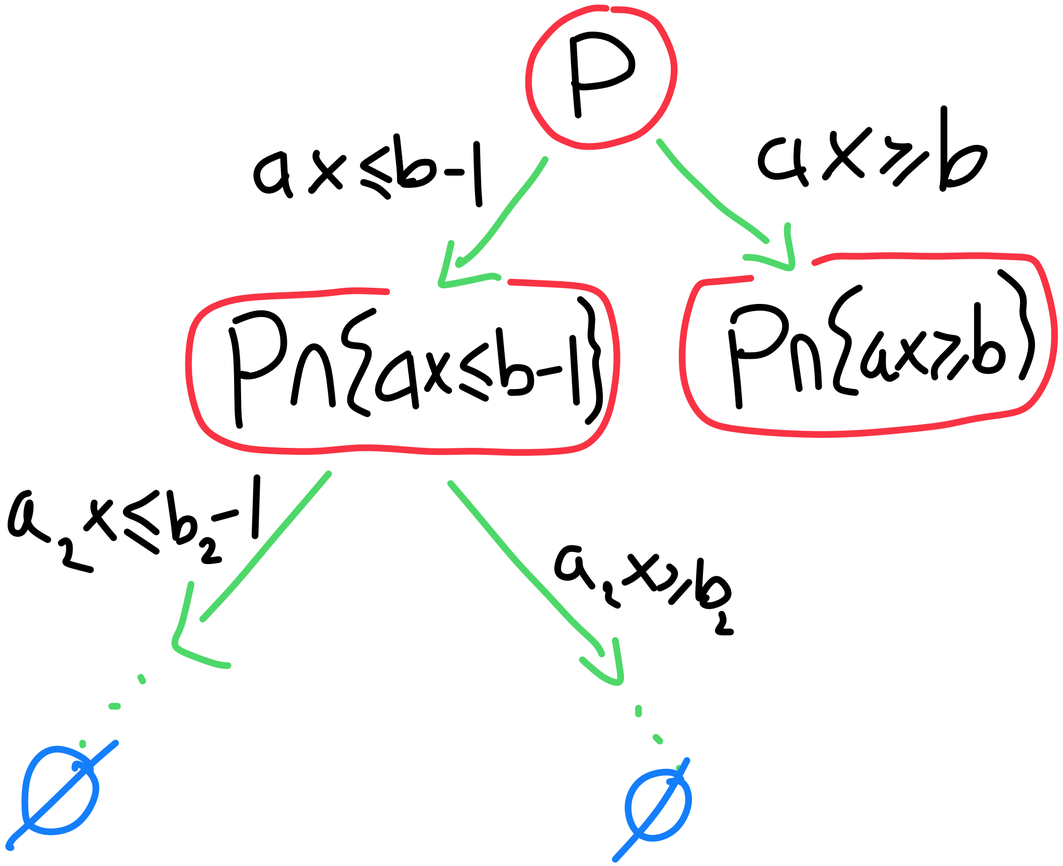
$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.

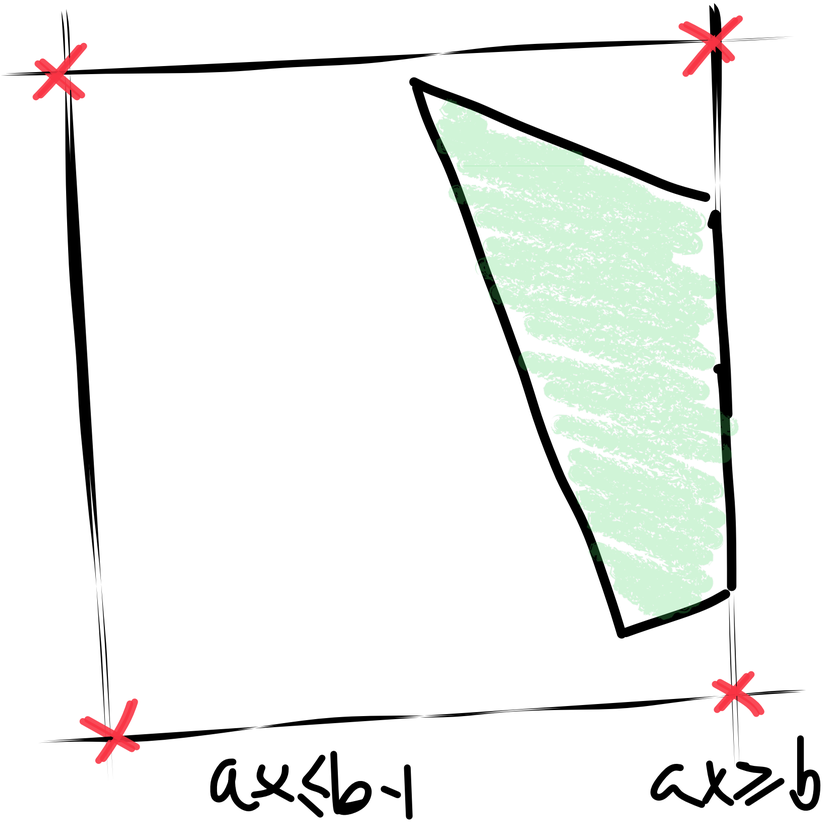




Stabbing Planes

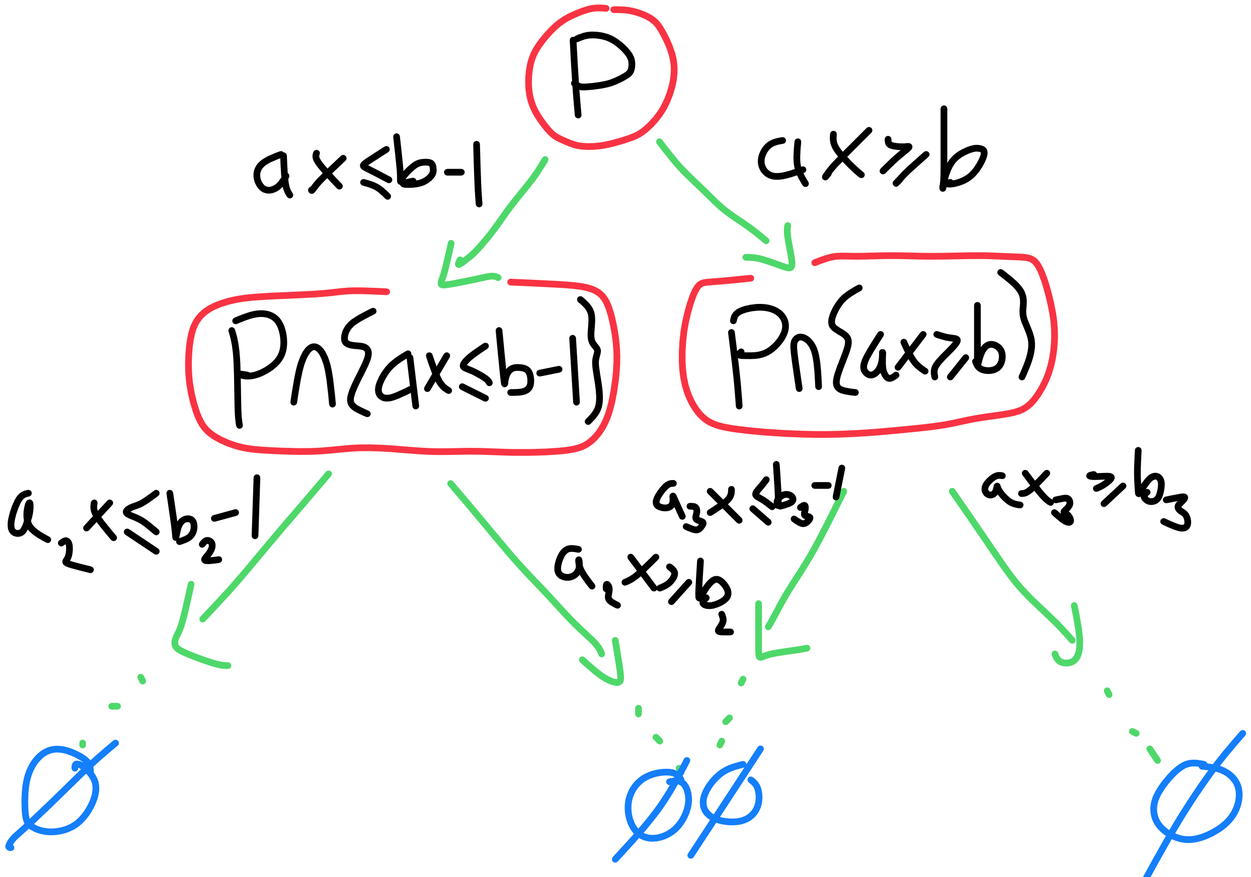
$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.





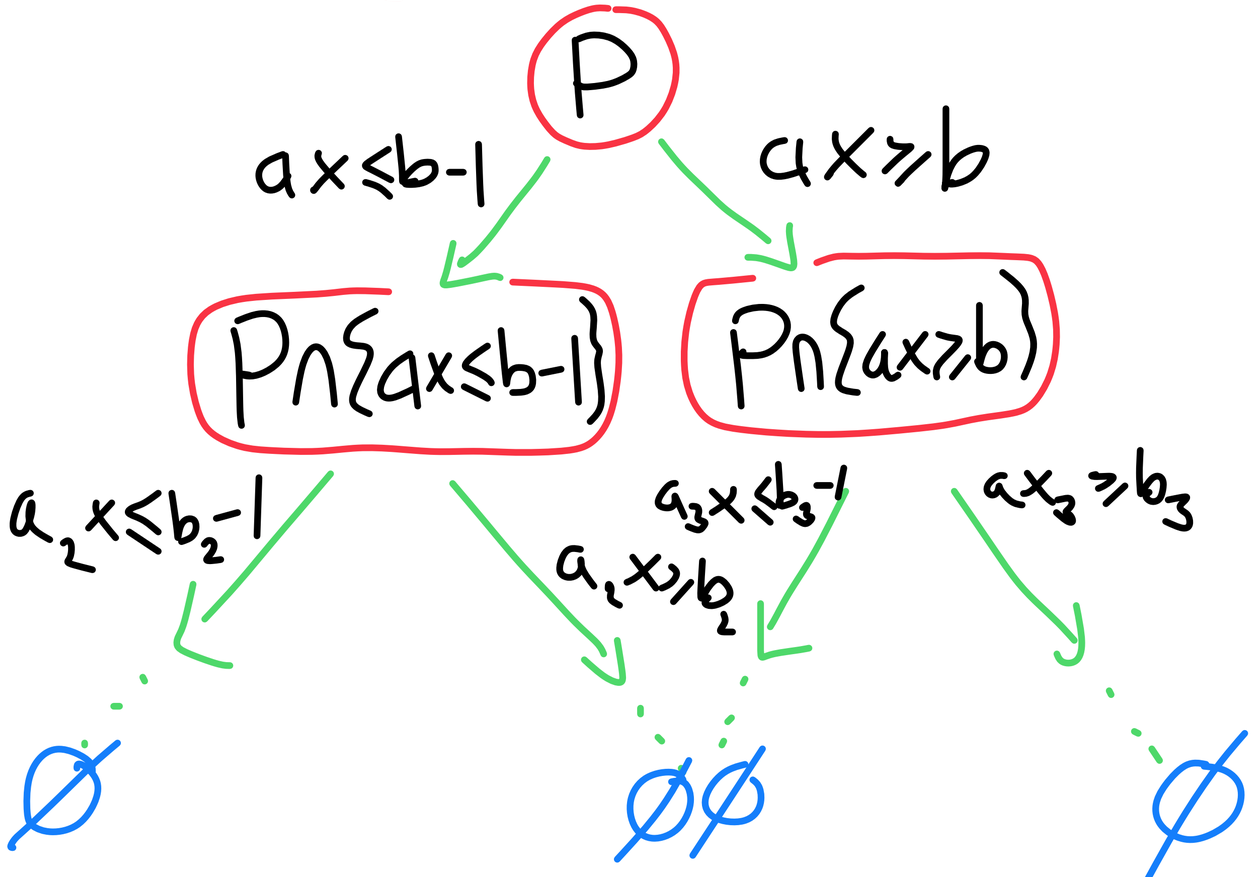
Stabbing Planes

$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.



Stabbing Planes

$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.

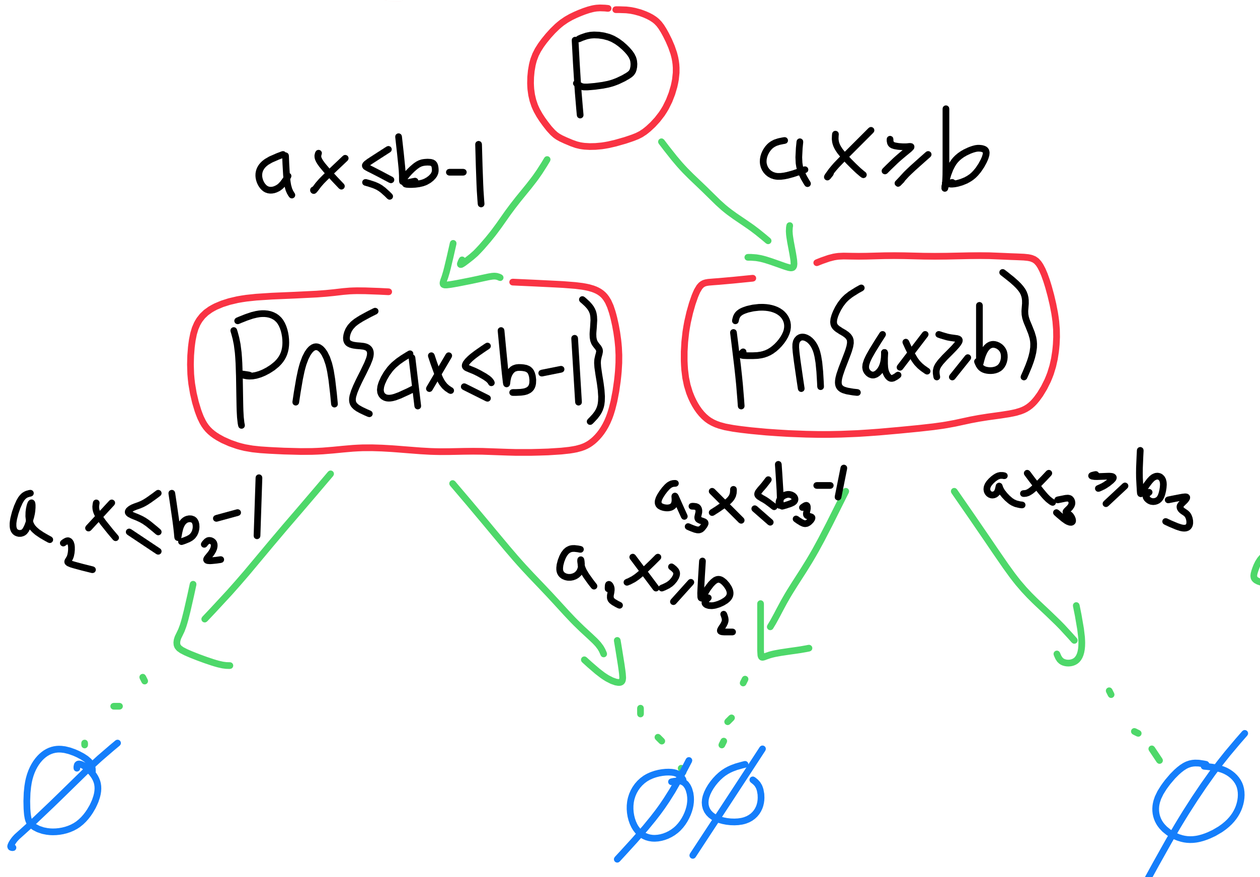


Refutation

Empty polytope $\emptyset$ deduced at every leaf proves $P \cap \mathbb{Z}^n = \emptyset$

Stabbing Planes

$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.



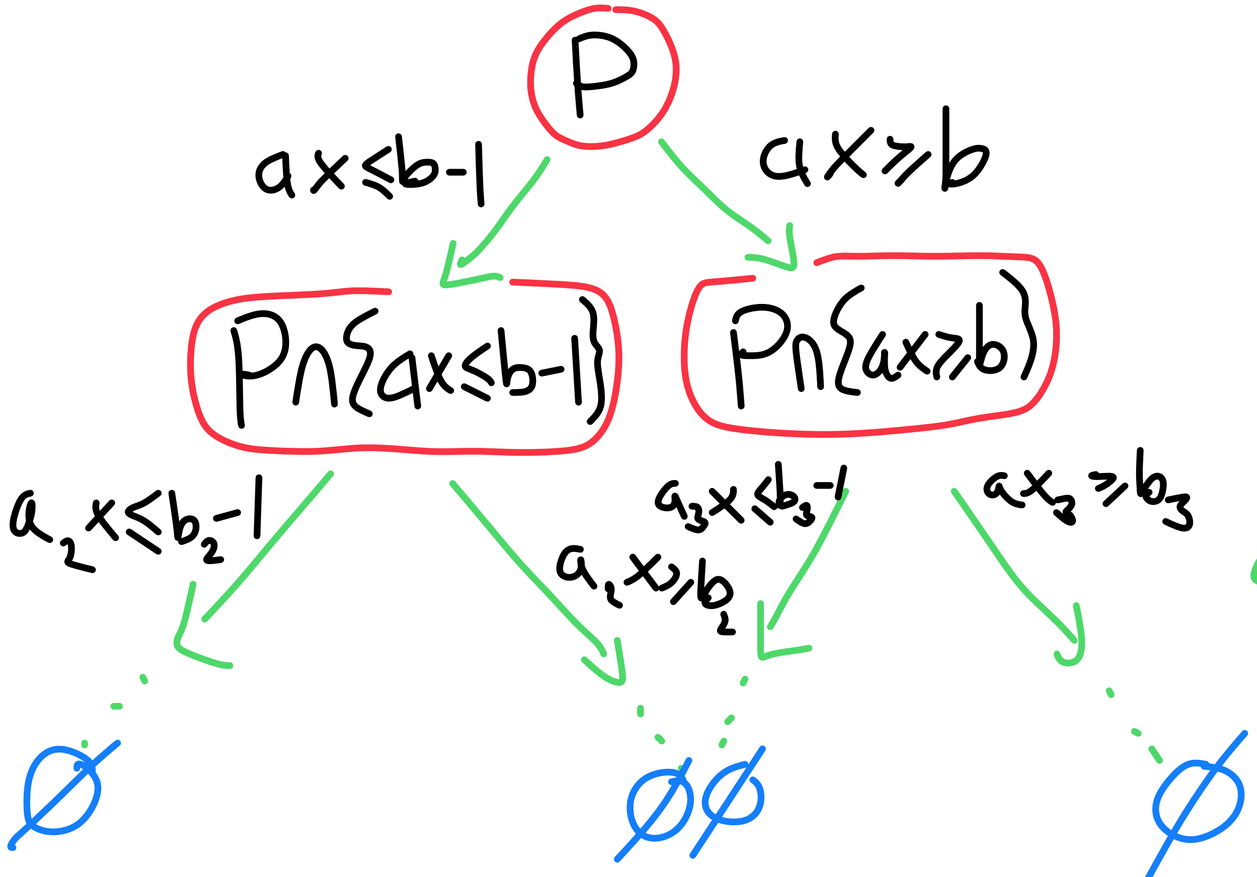
Refutation

Empty polytope $\emptyset$ deduced at every leaf proves $P \cap \mathbb{Z}^n = \emptyset$

Size: # of queries

Stabbing Planes

$P = \{Ax \geq b\}$ s.t. $P \cap \mathbb{Z}^n = \emptyset$.



Refutation

Empty polytope $\emptyset$ deduced at every leaf proves $P \cap \mathbb{Z}^n = \emptyset$

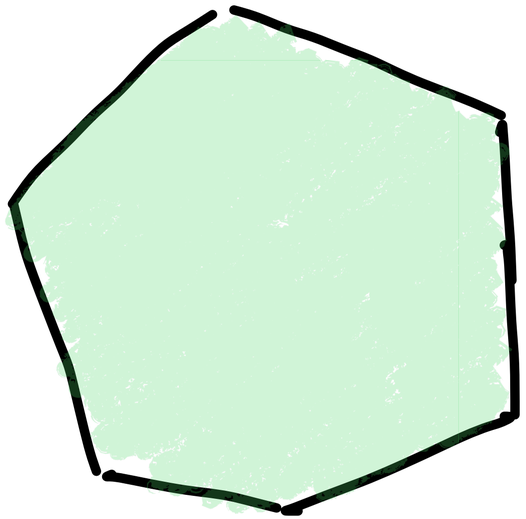
Size: # of queries $\approx$ bit-size [DT20]

Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions

Stubbing Planes vs Cutting Planes

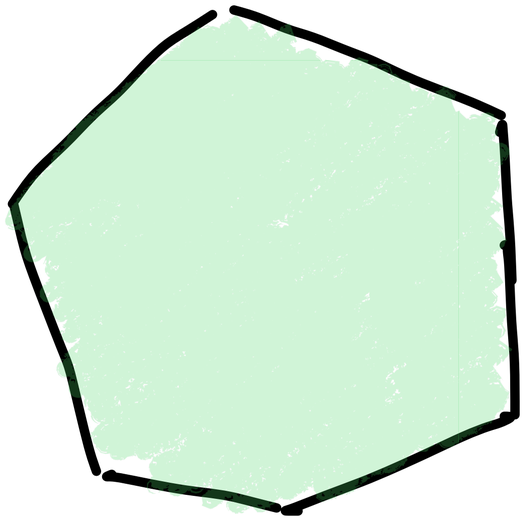
- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP



Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

Clm: path like SP = CP

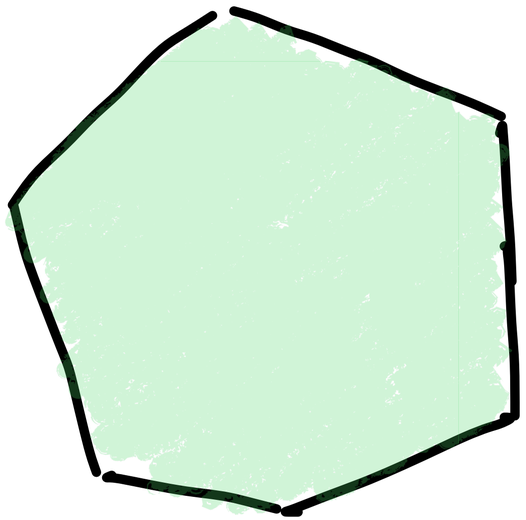


Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

Pathlike SP: A SP query $(ax \le b-1,\ ax \ge b)$ at P is pathlike if $P \cap \{ax \le b-1\} = \emptyset$ or $P \cap \{ax \ge b\} = \emptyset$

Clm: pathlike SP = CP

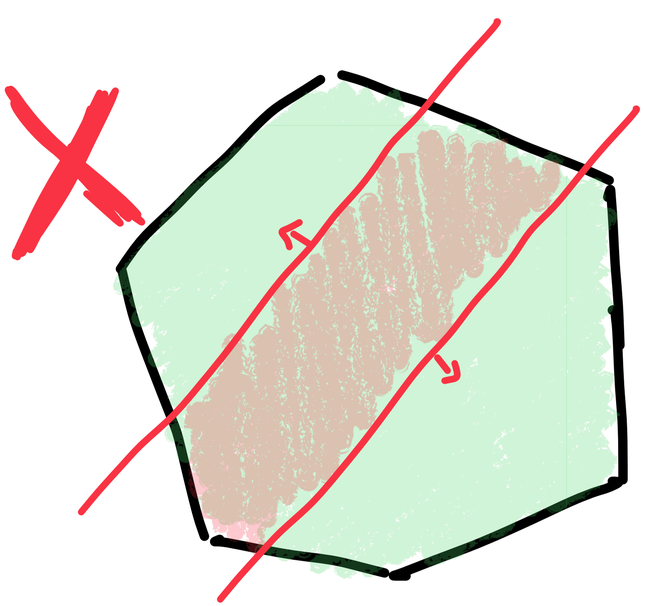


Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

Pathlike SP: A SP query $(ax \le b-1,\ ax \ge b)$ at P is pathlike if $P \cap \{ax \le b-1\} = \emptyset$ or $P \cap \{ax \ge b\} = \emptyset$

Clm: pathlike SP = CP

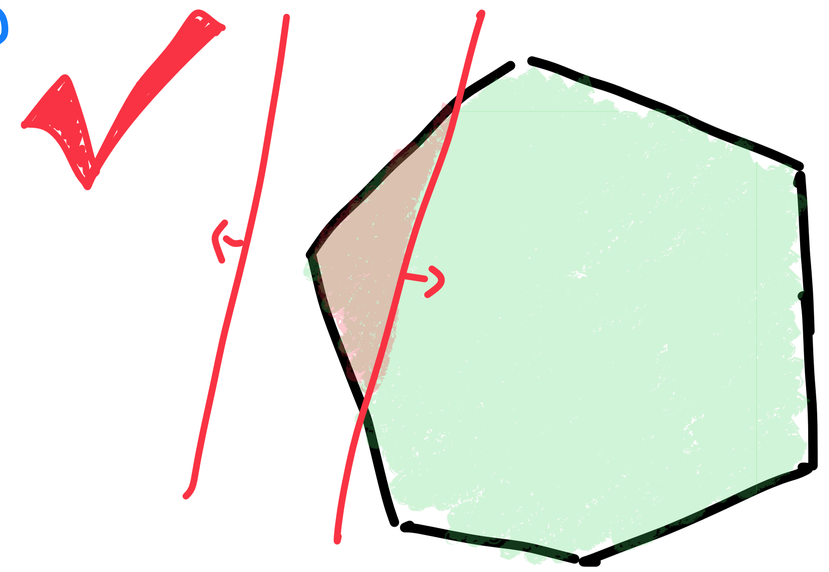


Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

Pathlike SP: A SP query $(ax \le b-1,\ ax \ge b)$ at P is pathlike if $P \cap \{ax \le b-1\} = \emptyset$ or $P \cap \{ax \ge b\} = \emptyset$

Clm: pathlike SP = CP



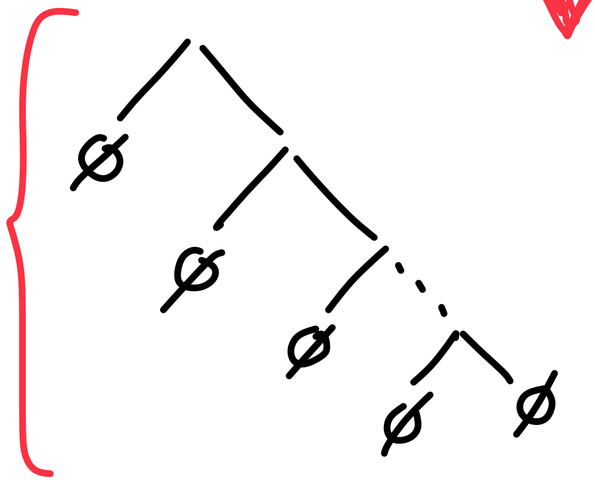
Stubbing Planes vs Cutting Planes

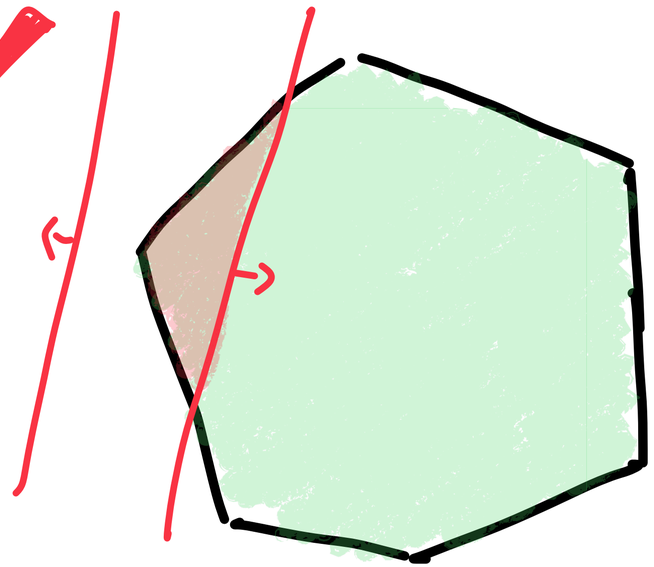
- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

Pathlike SP: A SP query $(ax \le b-1,\ ax \ge b)$ at P is pathlike if $P \cap \{ax \le b-1\} = \emptyset$ or $P \cap \{ax \ge b\} = \emptyset$

Clm: pathlike SP = CP

Pathlike SP Proof





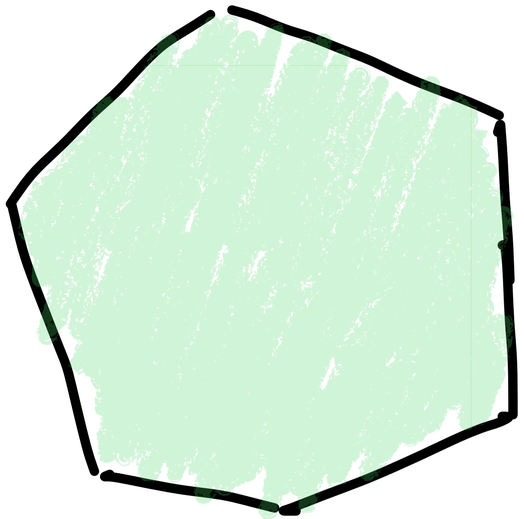
Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

Pathlike SP: A SP query $(ax \leq b-1,\ ax \geq b)$ at P is pathlike if $P \cap \{ax \leq b-1\} = \emptyset$ or $P \cap \{ax \geq b\} = \emptyset$

Clm: pathlike SP = CP

Pf: Show each CG-cut is a pathlike query



Stubbing Planes vs Cutting Planes

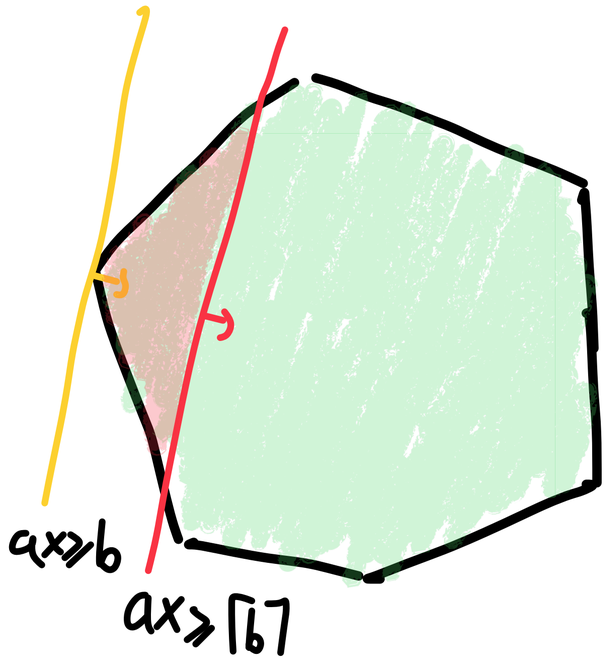
- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

Pathlike SP: A SP query $(ax \le b-1,\ ax \ge b)$ at P is pathlike if $P \cap \{ax \le b-1\} = \emptyset$ or $P \cap \{ax \ge b\} = \emptyset$

Clm: pathlike SP = CP

Pf: Show each CG-cut is a pathlike query

$ax \ge \lceil b \rceil$ is a CG-cut for P if $ax \ge b$ is valid for P



Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

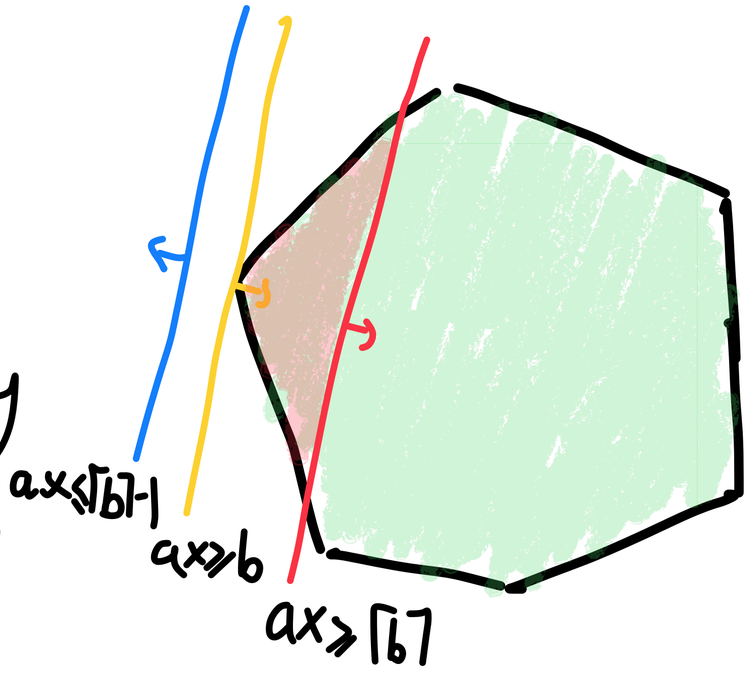
Pathlike SP: A SP query $(ax \leq b-1,\ ax \geq b)$ at P is pathlike if $P \cap \{ax \leq b-1\} = \emptyset$ or $P \cap \{ax \geq b\} = \emptyset$

Clm: pathlike SP = CP

Pf: Show each CG-cut is a pathlike query

$ax \geq \lceil b \rceil$ is a CG-cut for P if $ax \geq b$ is valid for P

$\Rightarrow P \cap \{ax \leq \lceil b \rceil - 1\} = \emptyset$



Stubbing Planes vs Cutting Planes

- Branch-and-cut allows Cutting planes deductions
  - → Superfluous as SP can simulate CP

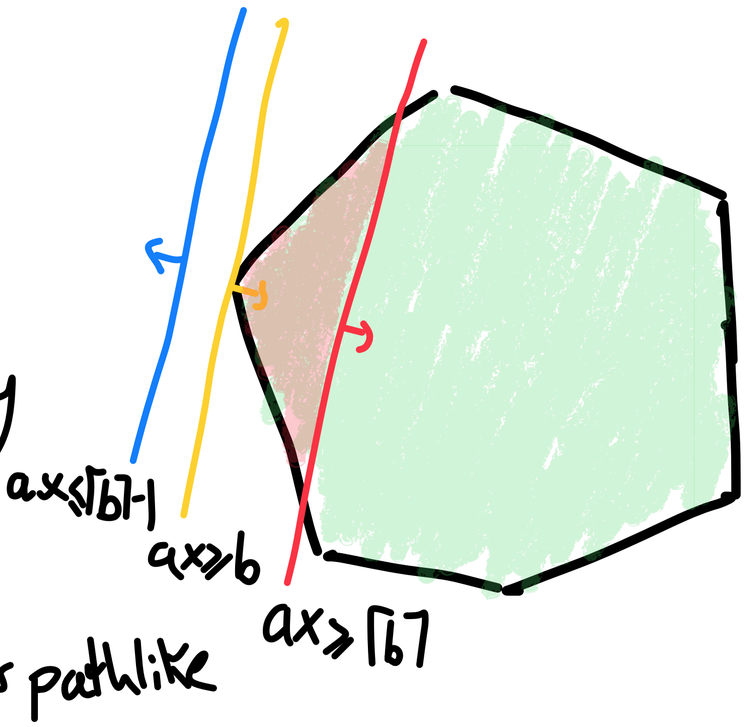
Pathlike SP: A SP query $(ax \le b-1,\ ax \ge b)$ at P is pathlike if $P \cap \{ax \le b-1\} = \emptyset$ or $P \cap \{ax \ge b\} = \emptyset$

Clm: pathlike SP = CP

Pf: Show each CG-cut is a pathlike query

$ax \ge \lceil b \rceil$ is a CG-cut for P if $ax \ge b$ is valid for P

$\Rightarrow P \cap \{ax \le \lceil b \rceil - 1\} = \emptyset$ & $(ax \le \lceil b \rceil - 1,\ ax \ge \lceil b \rceil)$ is pathlike



Stubbing Planes vs Cutting Planes

Q Can we separate CP from SP?

Stubbing Planes vs Cutting Planes

Q Can we separate CP from SP?

- Candidate: Tseitin formulas

Stubbing Planes vs Cutting Planes

Q Can we separate CP from SP?

- Candidate: Tseitin formulas
  - → quasipoly-size SP proofs of Tseitin [BFI+18]

Stubbing Planes vs Cutting Planes

Q Can we separate CP from SP?

- Candidate: Tseitin formulas
  - → quasipoly-size SP proofs of Tseitin [BFI+18]
  - → Conjectured to be hard for CP

Stubbing Planes vs Cutting Planes

Q Can we separate CP from SP?

- Candidate: Tseitin formulas
  - → quasipoly-size SP proofs of Tseitin [BFI+18]
  - → Conjectured to be hard for CP

[DT20] There are quasi-poly size CP proofs of Tseitin

Stubbing Planes vs Cutting Planes

Q Can we separate CP from SP?

- Candidate: Tseitin formulas
  - → quasipoly-size SP proofs of Tseitin [BFI+18]
  - → Conjectured to be hard for CP

[DT20] There are quasi-poly size CP proofs of Tseitin

- Translate the SP proof of Tseitin into CP

Stubbing Planes vs Cutting Planes

Q Can we separate CP from SP?

- Candidate: Tseitin formulas
  - → quasipoly-size SP proofs of Tseitin [BFI+18]
  - → Conjectured to be hard for CP

[DT20] There are quasi-poly size CP proofs of Tseitin

- Translate the SP proof of Tseitin into CP

Q Can every SP proof be translated into CP?

Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

Coefficients in queries are quasi-poly bounded

Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

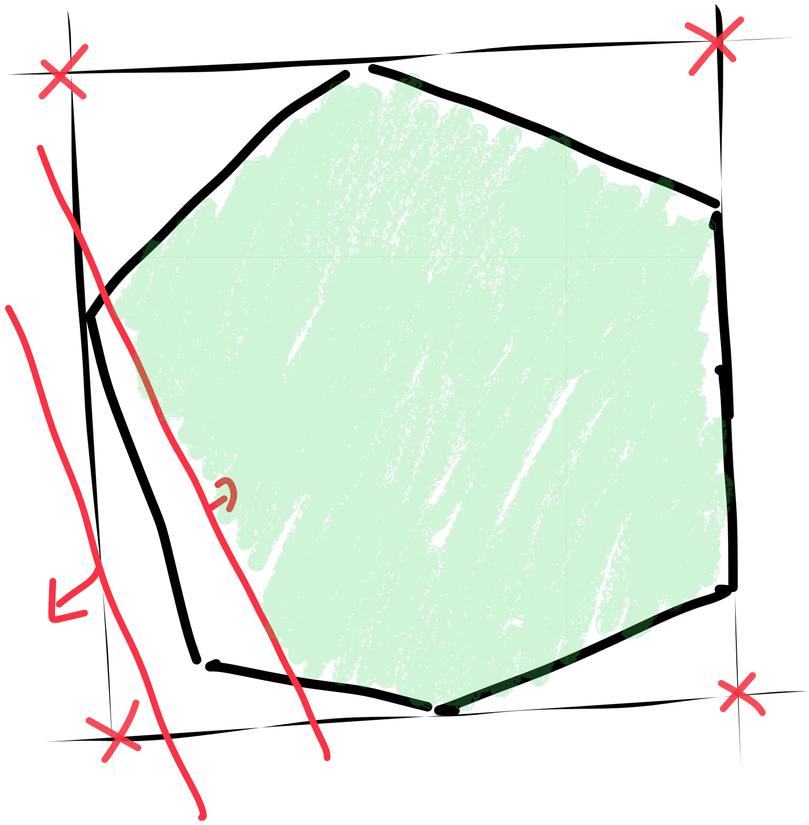
Pf: a) CP = pathlike SP = facelike SP

Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

Pf: a) CP = pathlike SP = facelike SP

Pathlike SP: A query $(ax \leq b-1, ax \geq b)$ is pathlike if $P \cap \{ax \leq b-1\} = \emptyset$ or $P \cap \{ax \geq b\} = \emptyset$



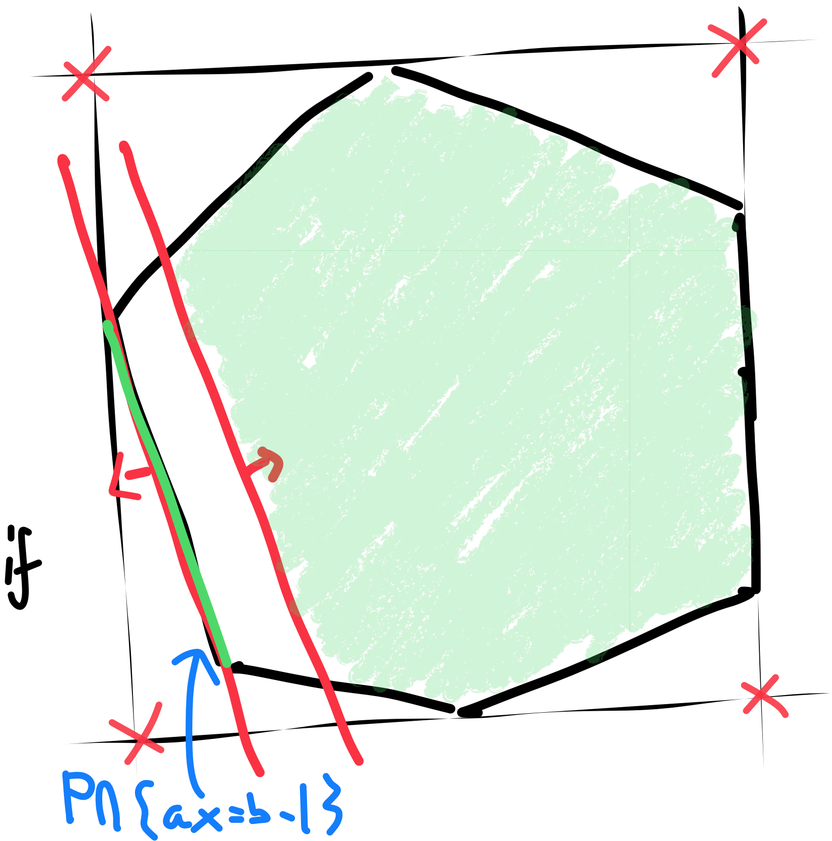
Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

Pf: a) CP = pathlike SP = facelike SP

Pathlike SP: A query $(ax \leq b-1, ax \geq b)$ is pathlike if $P \cap \{ax \leq b-1\} = \emptyset$ or $P \cap \{ax \geq b\} = \emptyset$

Facelike SP: A query $(ax \leq b-1, ax \geq b)$ is facelike if $P \cap \{ax < b-1\} = \emptyset$ or $P \cap \{ax > b\} = \emptyset$



Stubbing Planes vs Cutting Planes

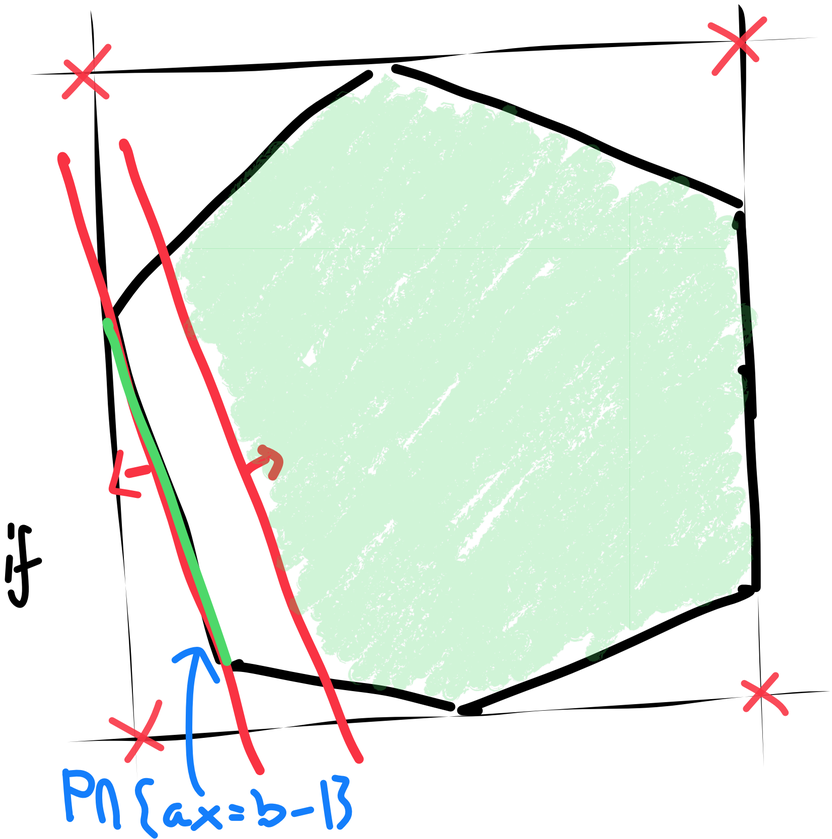
Thm: Every SP^* proof can be quasipolynomially translated into CP

Pf: a) CP = pathlike SP = facelike SP

Pathlike SP: A query $(ax \leq b-1, ax \geq b)$ is pathlike if $P \cap \{ax \leq b-1\} = \emptyset$ or $P \cap \{ax \geq b\} = \emptyset$

Facelike SP: A query $(ax \leq b-1, ax \geq b)$ is facelike if $P \cap \{ax < b-1\} = \emptyset$ or $P \cap \{ax > b\} = \emptyset$

ie $P \cap \{ax \leq b-1\}$ or $P \cap \{ax \geq b\}$ is a face.



Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

Pf: a) CP = pathlike SP = facelike SP

b) Any SP^* proof can be made facelike with a quasi-poly blowup in size.

Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

→ SP^* is a "query" proof system (like DPLL) for CP

Stubbing Planes vs Cutting Planes

Thm: Every SP^* proof can be quasipolynomially translated into CP

→ SP^* is a "query" proof system (like DPLL) for CP

Cor: Exponential lower bounds on SP^*

Refuting Systems of Equations in CP

Cor: Every unsat system of linear equations over a finite field has a quasi-poly refutation in CP.

Refuting Systems of Equations in CP

Cor: Every unsat system of linear equations over a finite field has a quasi-poly refutation in CP.

Step 1. A general algorithm for refuting systems of equations

Refuting Systems of Equations in CP

Cor: Every unsat system of linear equations over a finite field has a quasi-poly refutation in CP.

Step1. A general algorithm for refuting systems of equations

Step2. Implement in SP

Refuting Systems of Equations in CP

Cor: Every unsat system of linear equations over a finite field has a quasi-poly refutation in CP.

Step1. A general algorithm for refuting systems of equations

Step2. Implement in SP

Ex $\mathbb{F}_2$ linear equations

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

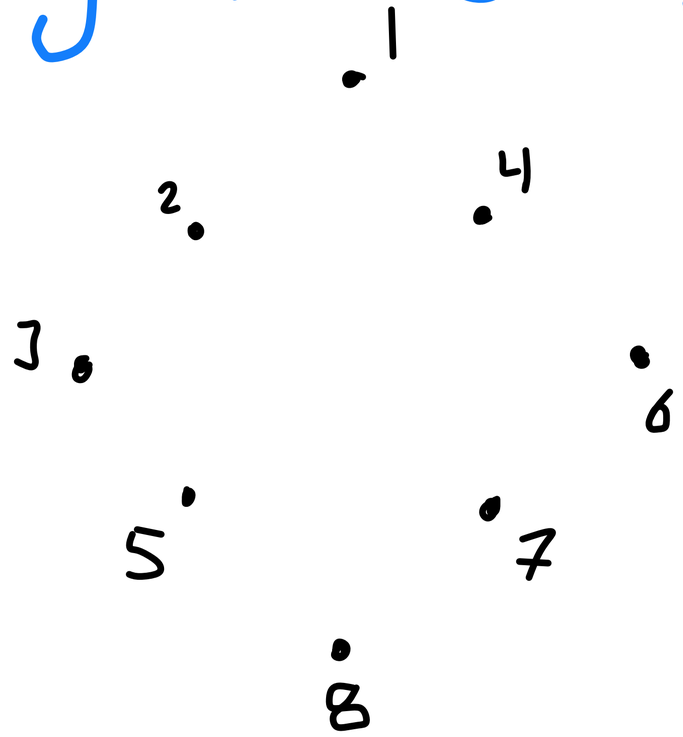
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



Ex $\mathbb{F}_2$ linear equations

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

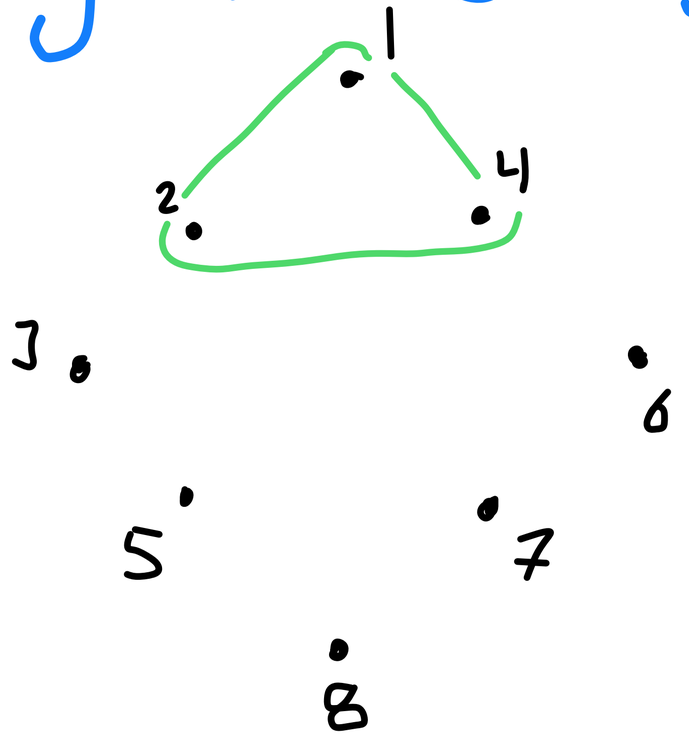
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

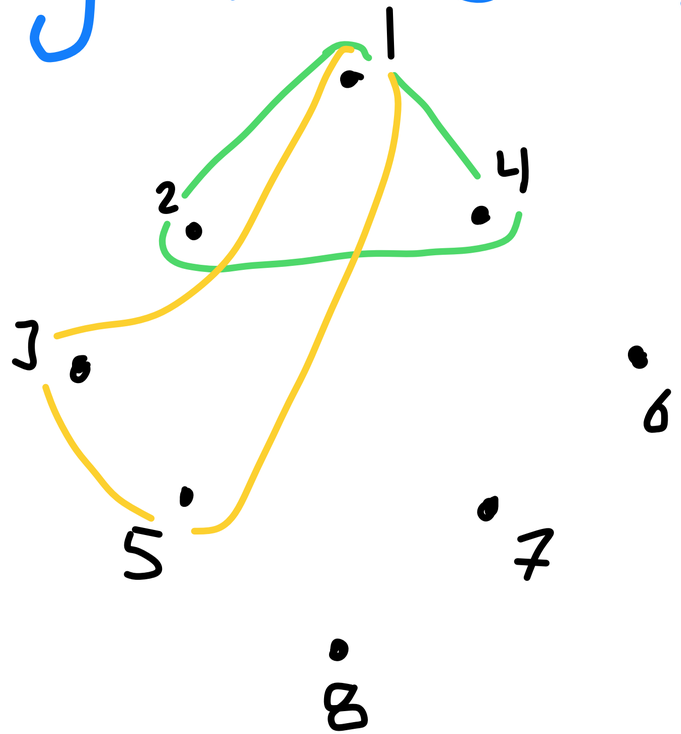
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

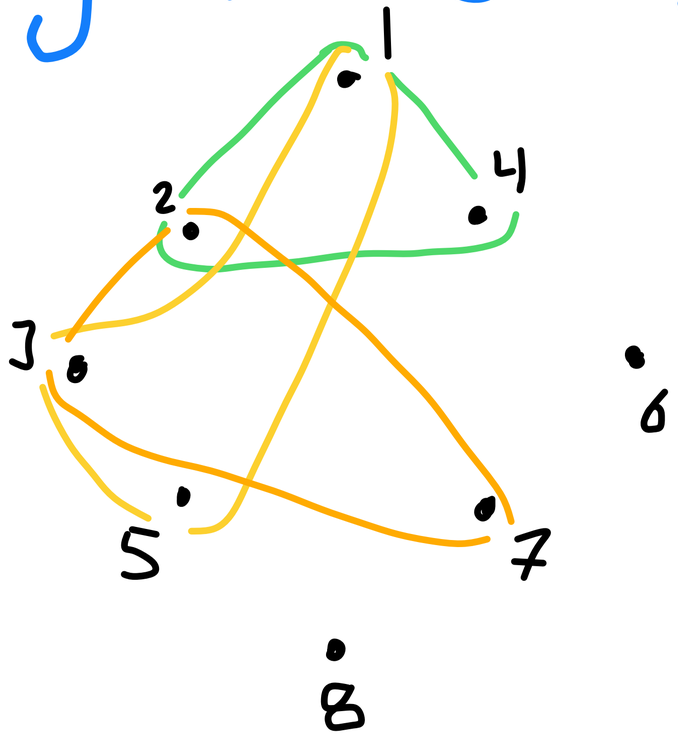
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

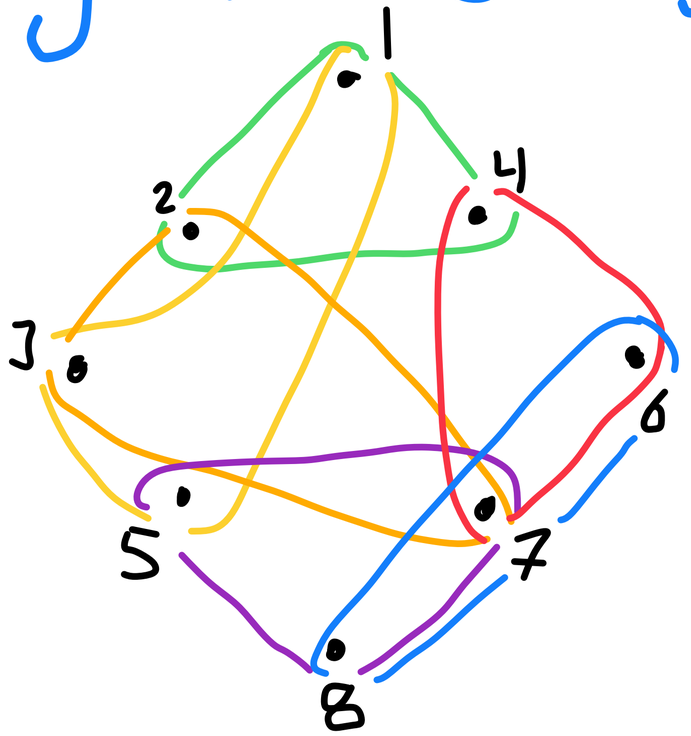
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



E= set of edges

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

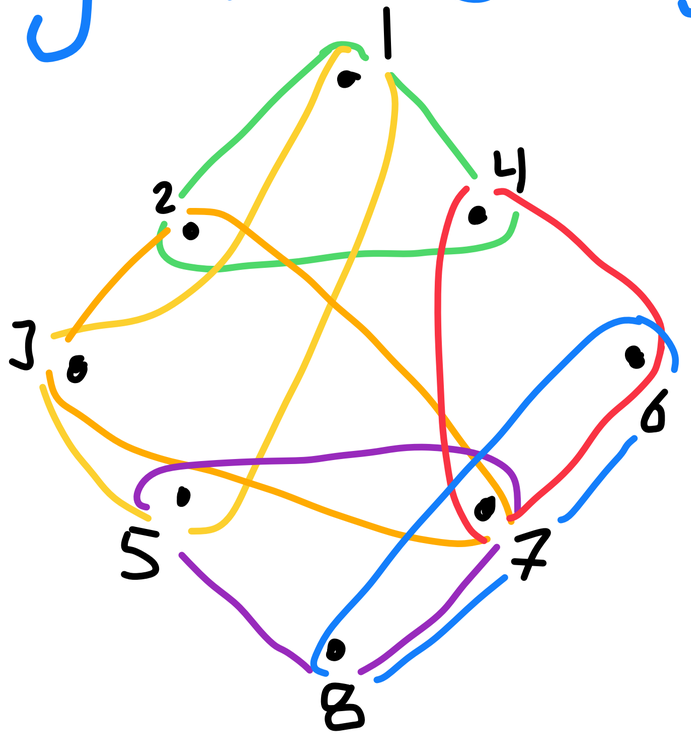
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



E = set of edges

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

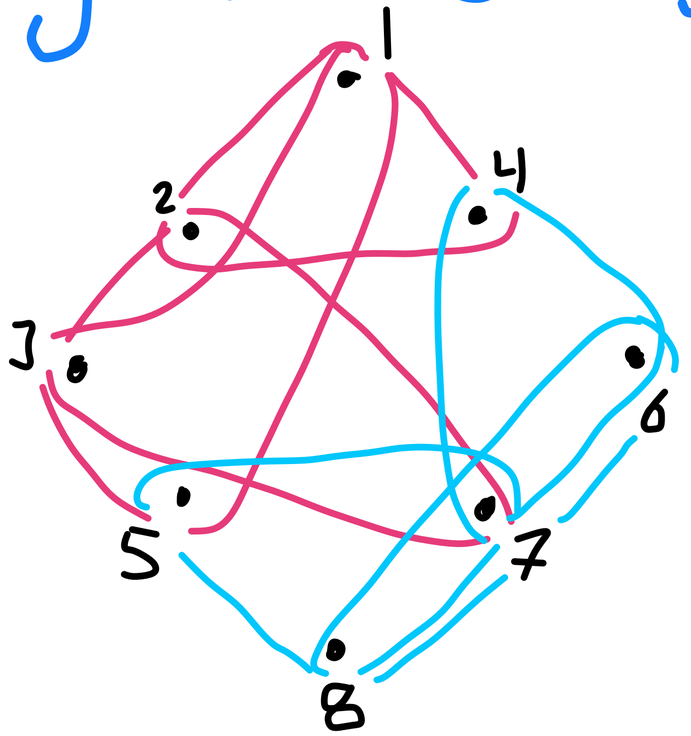
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

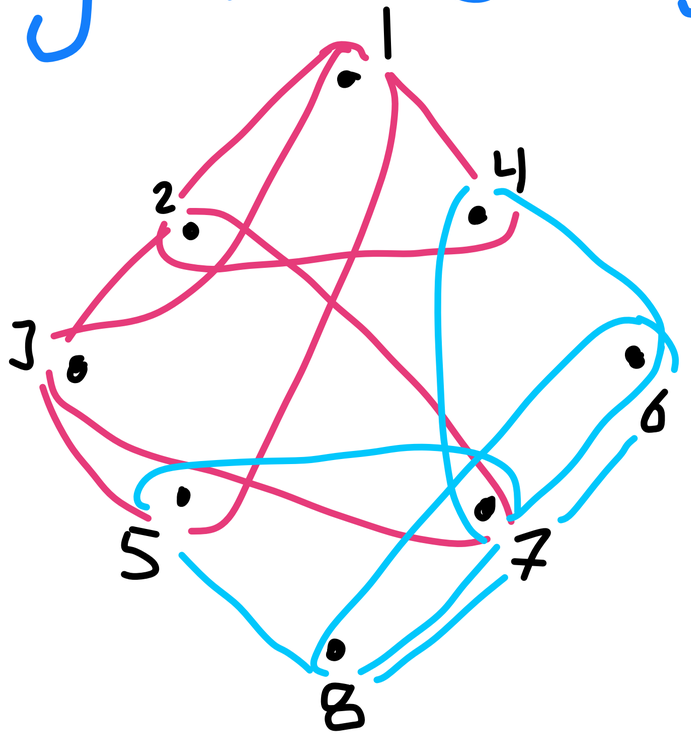
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_j) = \{x_i \mid x_i$ occurs an odd number of times in $E_j\}$

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

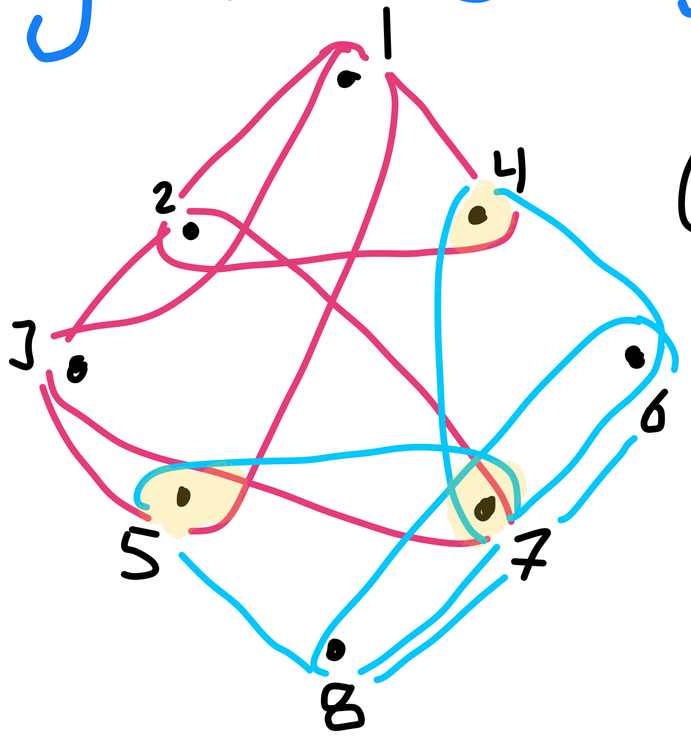
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



$Cut_2(E_1) = Cut_2(E_2) = \{x_4, x_5, x_7\}$

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_j) = \{x_i \mid x_i \text{ occurs an odd number of times in } E_j\}$

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

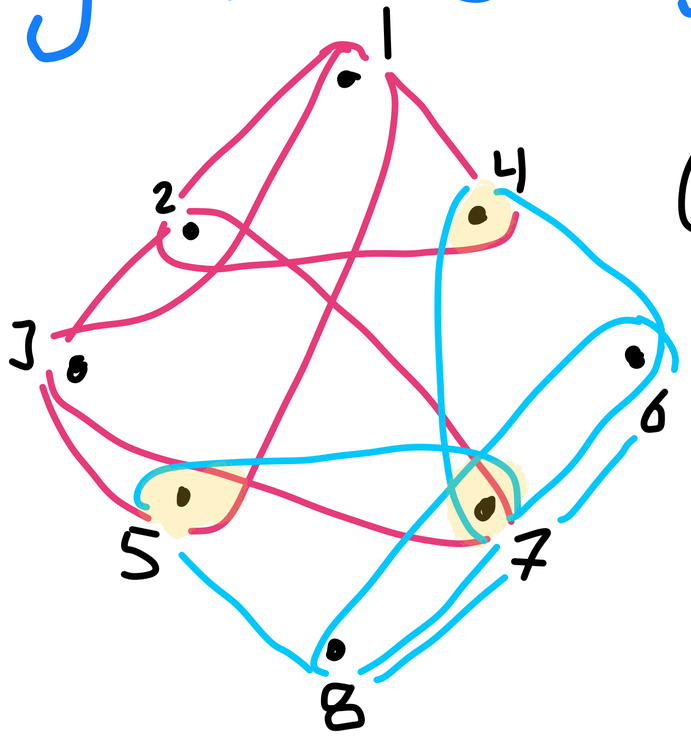
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



$Cut_2(E_1) = Cut_2(E_2) = \{x_4, x_5, x_7\}$

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_j) = \{x_i \mid x_i \text{ occurs an odd number of times in } E_j\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

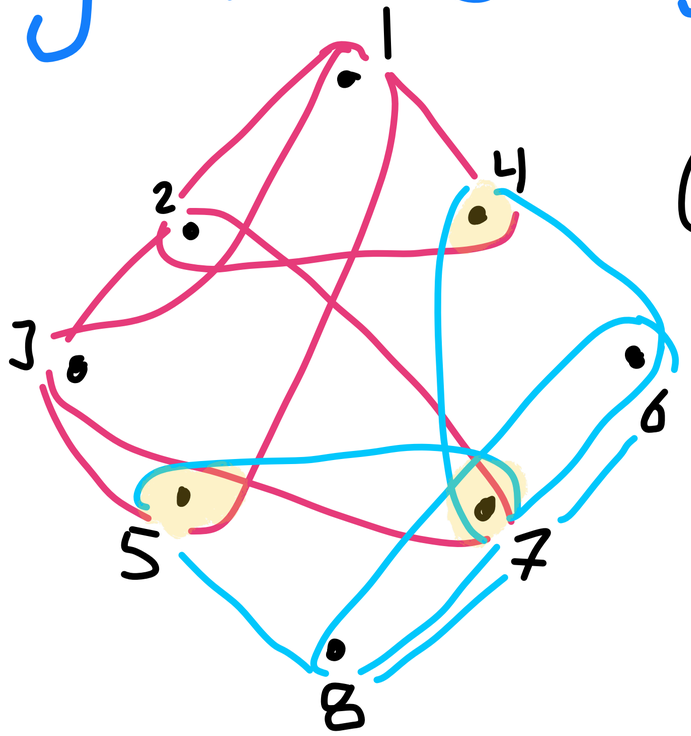
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



$Cut_2(E_1) = Cut_2(E_2) = \{x_4, x_5, x_7\}$

Suppose $x_4 + x_5 + x_7 \equiv 0 \bmod 2$

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

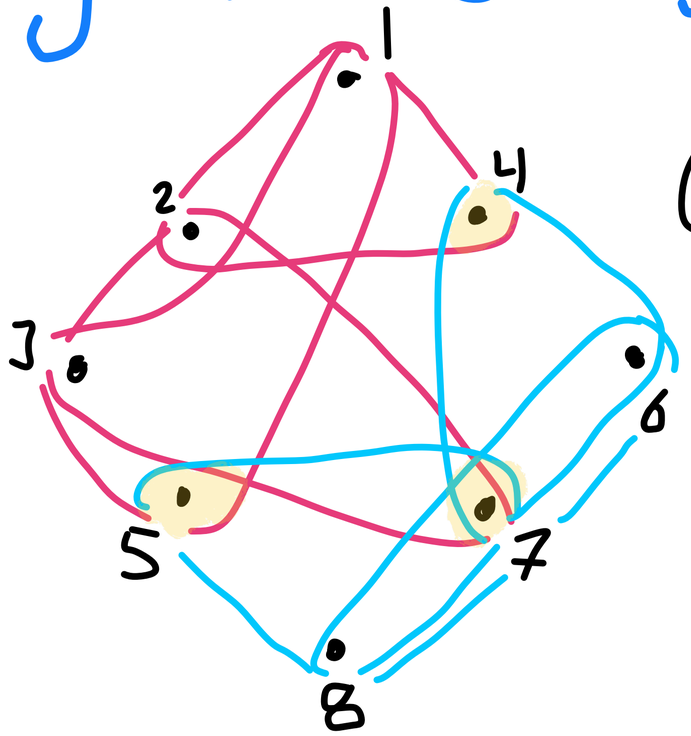
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



$Cut_2(E_1) = Cut_2(E_2) = \{x_4, x_5, x_7\}$

Suppose $x_4 + x_5 + x_7 \equiv 0 \bmod 2$

→ E_1 is unsat

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

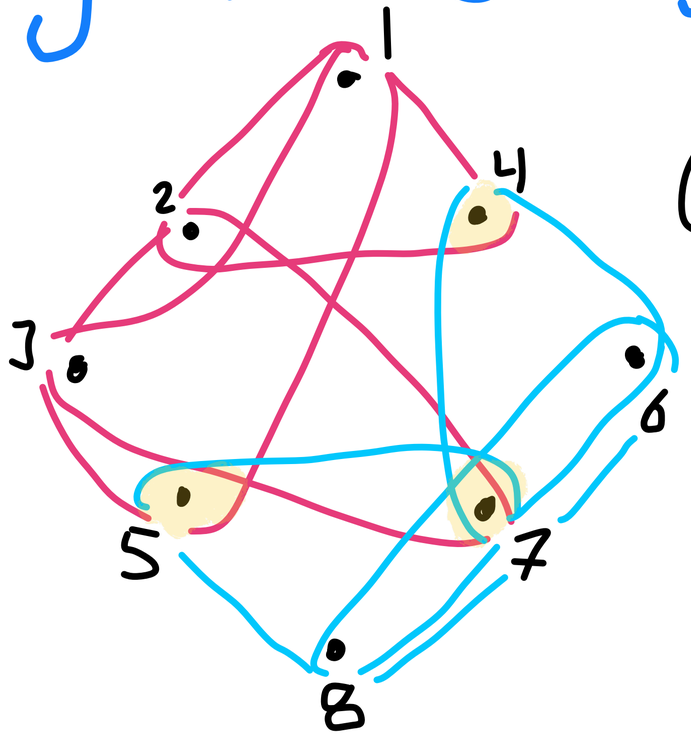
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



$Cut_2(E_1) = Cut_2(E_2) = \{x_4, x_5, x_7\}$

Suppose $x_4 + x_5 + x_7 \equiv 0 \bmod 2$

→ E_1 is unsat

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

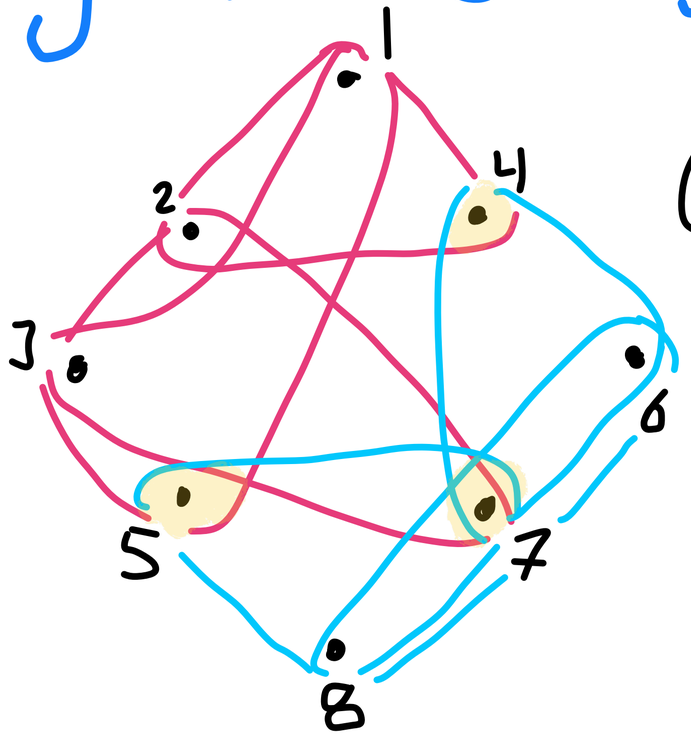
$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$

$x_4 + x_6 + x_7 = 0$

$x_5 + x_7 + x_8 = 0$

$x_6 + x_7 + x_8 = 0$



$Cut_2(E_1) = Cut_2(E_2) = \{x_4, x_5, x_7\}$

Suppose $x_4 + x_5 + x_7 \equiv 0 \bmod 2$

$\rightarrow E_1$ is unsat

Recurse on E_1

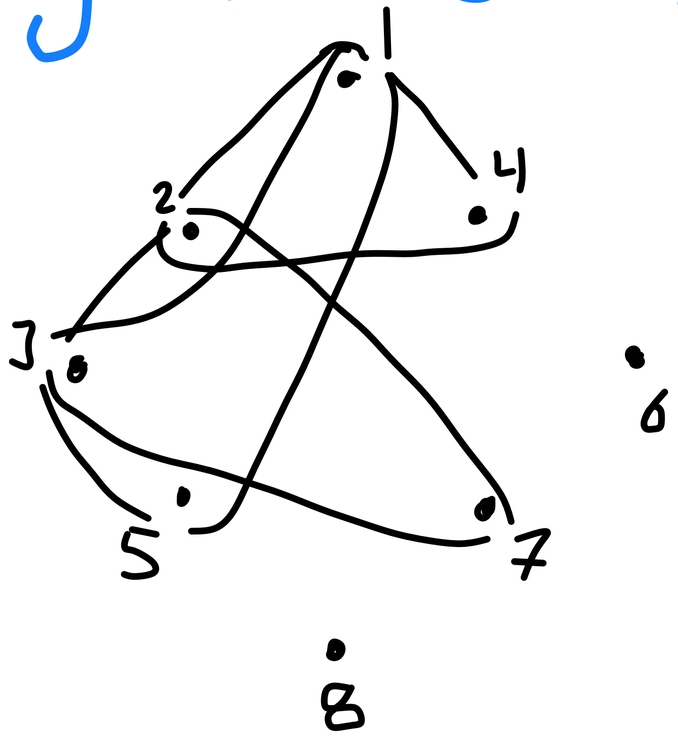
1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$



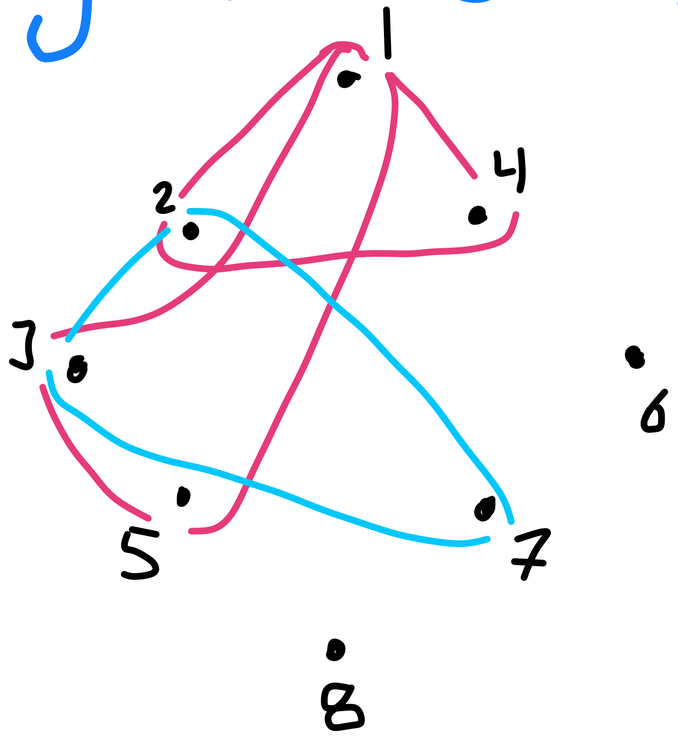
1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$



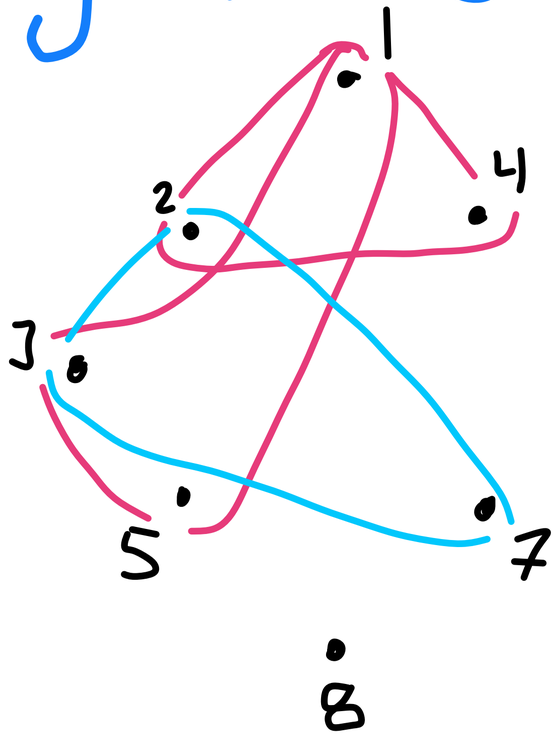
1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$



$Cut_2(E_1) = \{x_2, x_3, x_4, x_5\}$

$Cut_2(E_2) = \{x_2, x_3, x_7\}$

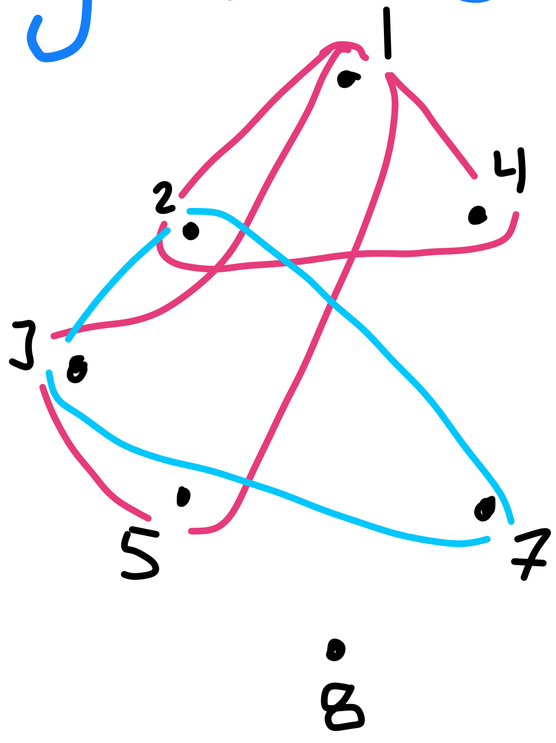
1. Partition E into E_1, E_2 so that $|E_1| \leq |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i \text{ occurs an odd number of times in } E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$



$Cut_2(E_1) = \{x_2, x_3, x_4, x_5\}$

Suppose $x_2 + x_3 + x_4 + x_5 \equiv 1 \bmod 2$

$Cut_2(E_2) = \{x_2, x_3, x_7\}$

Suppose $x_2 + x_3 + x_7 \equiv 1 \bmod 2$

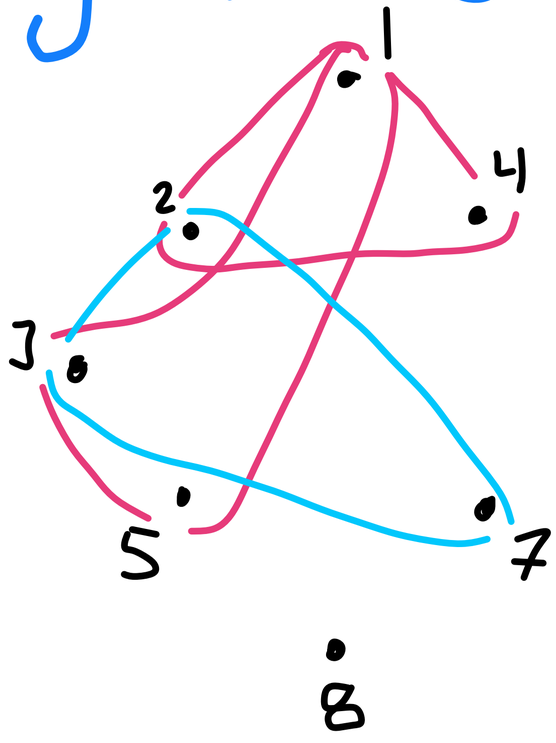
1. Partition E into E_1, E_2 so that $|E_1| \leq |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$

$x_2 + x_3 + x_7 = 1$



$Cut_2(E_1) = \{x_2, x_3, x_4, x_5\}$

Suppose $x_2 + x_3 + x_4 + x_5 \equiv 1 \bmod 2$

$Cut_2(E_2) = \{x_2, x_3, x_7\}$

Suppose $x_2 + x_3 + x_7 \equiv 1 \bmod 2$

→ E_1 is unsat

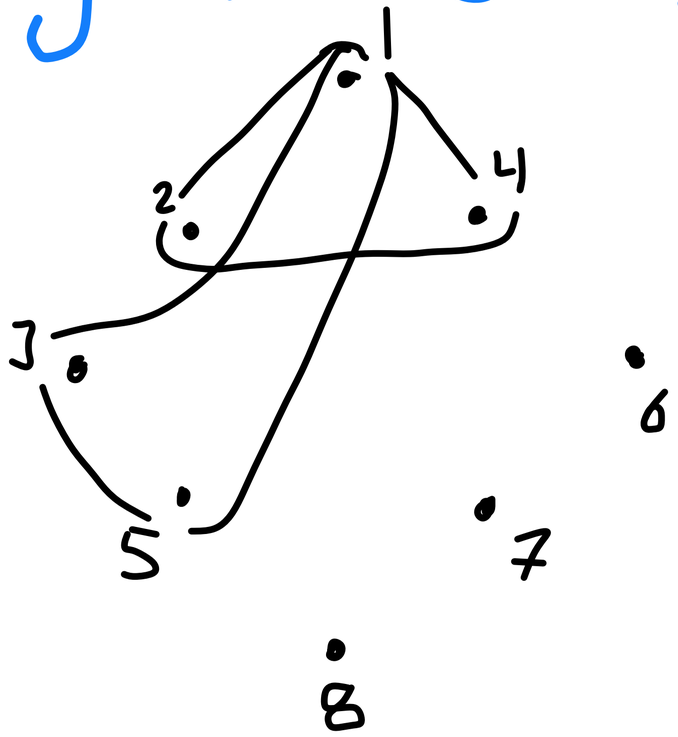
Recurse on E_1

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$

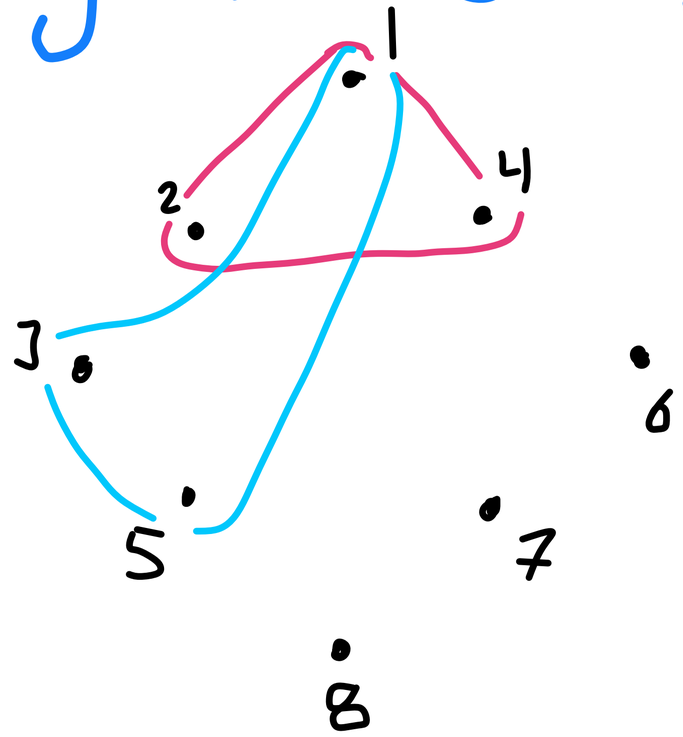


1. Partition E into E_1, E_2 so that $|E_1| \leq |E_2|$
2. Let $Cut_2(E_j) = \{x_i \mid x_i \text{ occurs an odd number of times in } E_j\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$

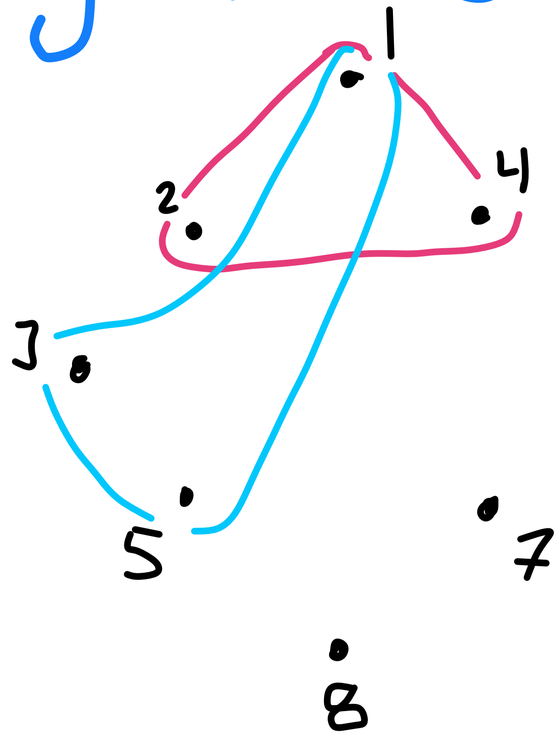


1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_j) = \{x_i \mid x_i \text{ occurs an odd number of times in } E_j\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$



$Cut_2(E_1) = \{x_1, x_2, x_4\}$

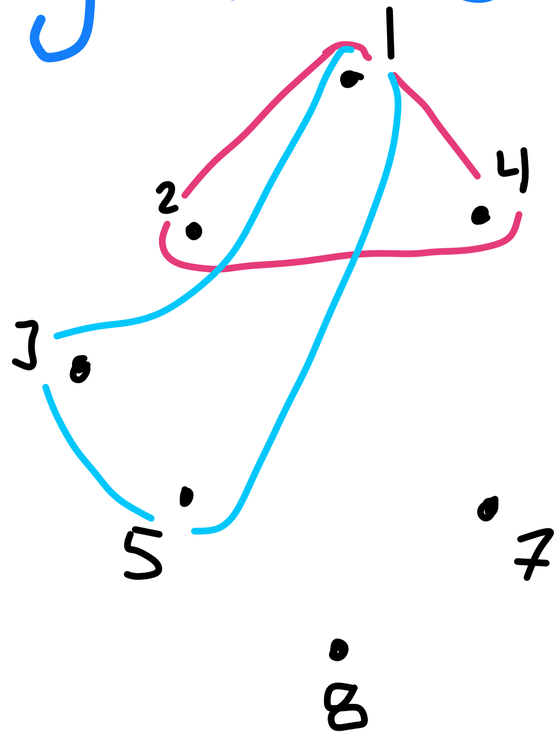
$Cut_2(E_2) = \{x_1, x_3, x_5\}$

1. Partition E into E_1, E_2 so that $|E_1| \leq |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$



$Cut_2(E_1) = \{x_1, x_2, x_4\}$

Suppose $x_1 + x_2 + x_4 \equiv 1 \bmod 2$

$Cut_2(E_2) = \{x_1, x_3, x_5\}$

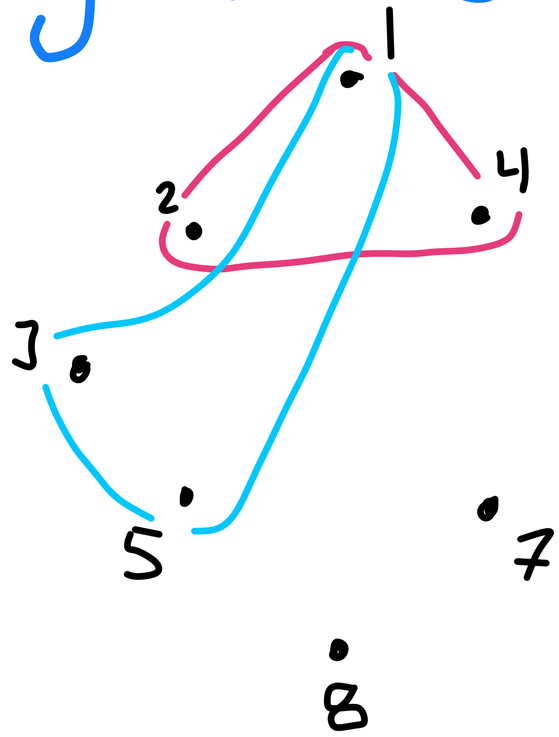
Suppose $x_1 + x_3 + x_5 \equiv 0 \bmod 2$

1. Partition E into E_1, E_2 so that $|E_1| \leq |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 + x_5 = 1$



$Cut_2(E_1) = \{x_1, x_2, x_4\}$

Suppose $x_1 + x_2 + x_4 \equiv 1 \bmod 2$

$Cut_2(E_2) = \{x_1, x_3, x_5\}$

Suppose $x_1 + x_3 + x_5 \equiv 0 \bmod 2$

Violates $x_1 + x_3 + x_5 \equiv 1$

1. Partition E into E_1, E_2 so that $|E_1| \approx |E_2|$
2. Let $Cut_2(E_i) = \{x_i \mid x_i$ occurs an odd number of times in $E_i\}$
3. Branch on the parity of the sum of $Cut_2(E_1)$ and the parity of the sum of $Cut_2(E_2)$
4. Recurse on the unsat half.

Refuting Systems of Equations in SP

To implement in SP:

- Must be able to determine the parity of $Cut_2(E)$

Refuting Systems of Equations In SP

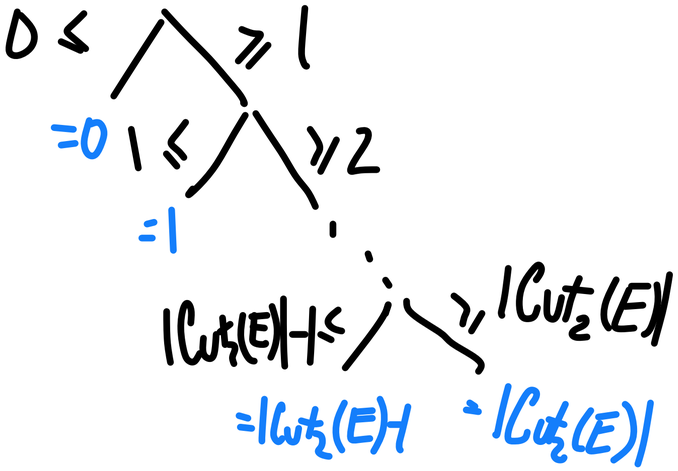
To implement in SP:

- Must be able to determine the parity of $Cut_2(E)$
  - Query $\left(\sum_{i \in Cut_2(E)} x_i \leq t, \sum_{i \in Cut_2(E)} x_i \geq t+1\right)$ for $t = 0, \ldots, |Cut_2(E)|$

Refuting Systems of Equations In SP

To implement in SP:

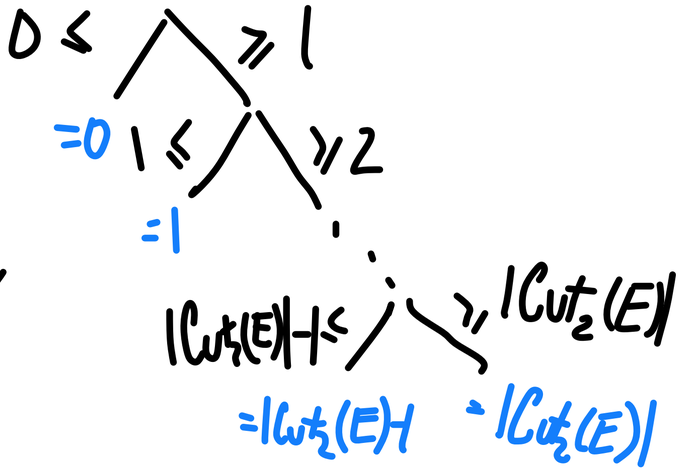
- Must be able to determine the parity of $Cut_2(E)$
  - Query $\left(\sum_{i \in Cut_2(E)} x_i \leq t, \sum_{i \in Cut_2(E)} x_i \geq t+1\right)$ for $t = 0, \ldots, |Cut_2(E)|$



Refuting Systems of Equations In SP

To implement in SP:

- Must be able to determine the parity of $Cut_2(E)$
  - Query $\left(\sum_{i \in Cut_2(E)} x_i \leq t, \sum_{i \in Cut_2(E)} x_i \geq t+1\right)$ for $t = 0, \ldots, |Cut_2(E)|$



  - Once parity of $Cut_2(E_1)$ and of $Cut_2(E_2)$ determined, recurse on unsat side

Refuting Systems of Equations InSP

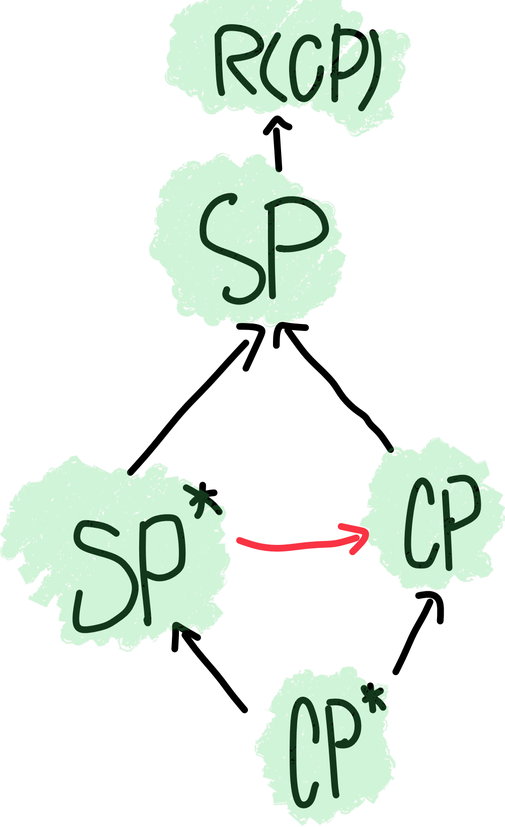
Complexity: $O(\log n)$ recursive rounds.

At most $|Cut_2(E_1)| \cdot |Cut_2(E_2)| \leq n^2$ queries per round

$\therefore n^{O(\log n)}$ size

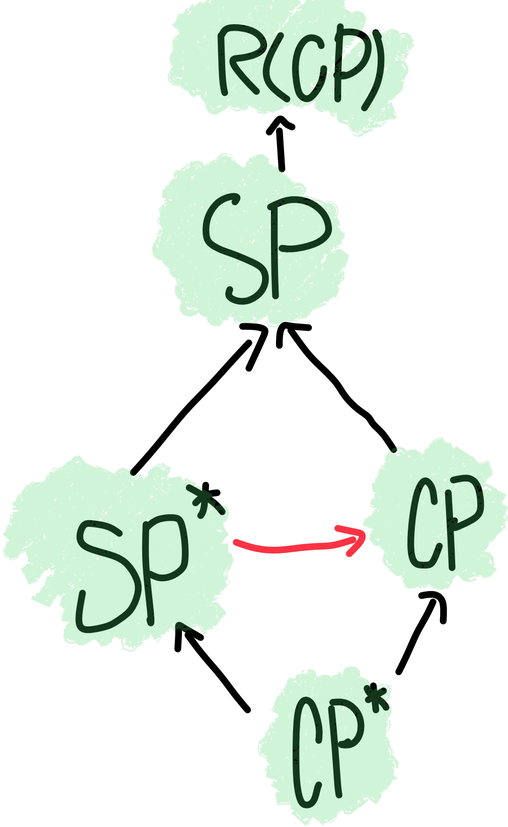
Open Problems

- Can CP p-simulate SP*?



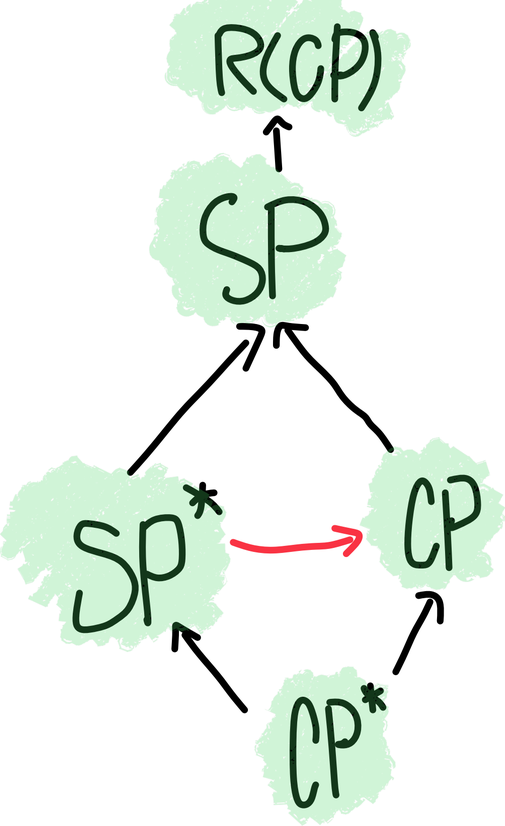
Open Problems

- Can CP p-simulate SP^*?
- Can CP (quasipolynomially) simulate SP?



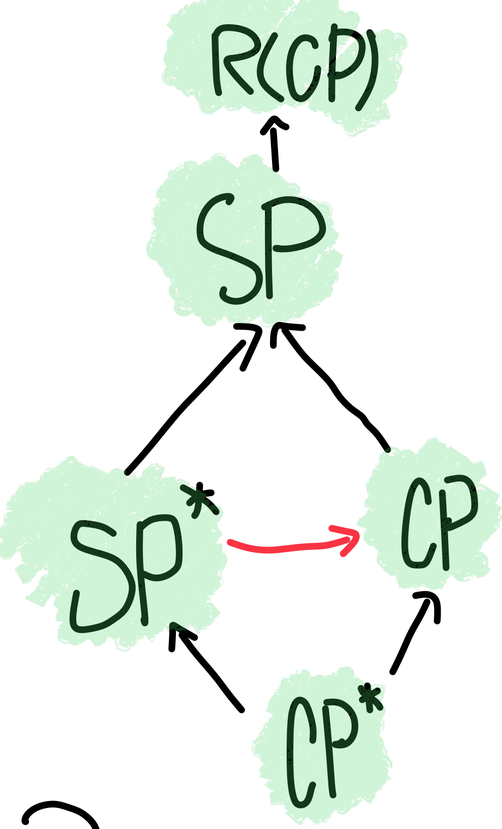
Open Problems

- Can CP p-simulate SP^*?
- Can CP (quasipolynomially) simulate SP?
- Can CP^* (quasipolynomially) simulate SP^*?



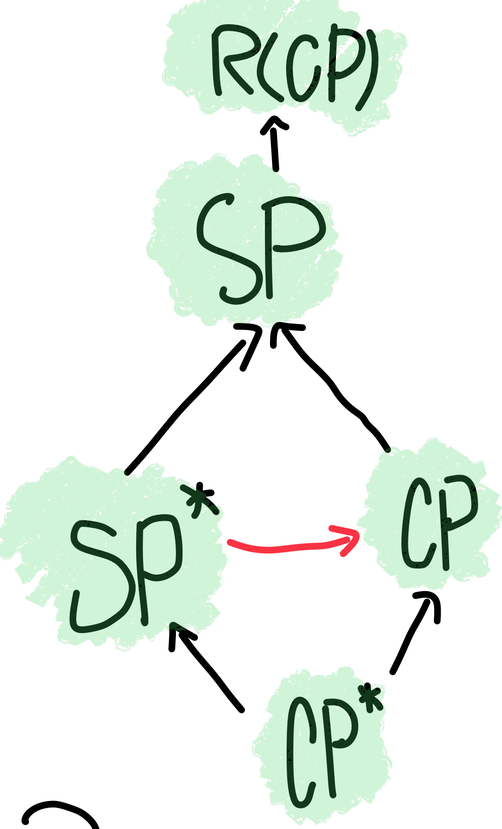
Open Problems

- Can CP p-simulate SP*?
- Can CP (quasipolynomially) simulate SP?
- Can CP* (quasipolynomially) simulate SP*?
- Can SP or CP simulate dag-like SP (R(CP))?



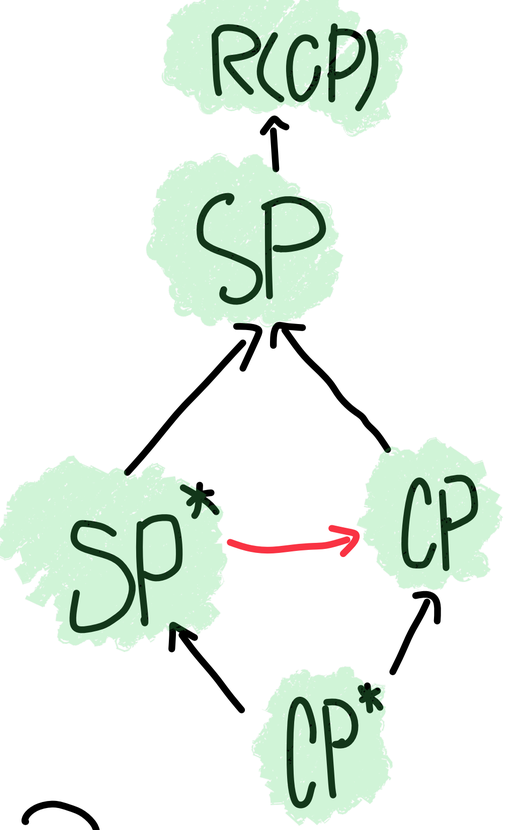
Open Problems

- Can CP p-simulate SP*?
- Can CP (quasipolynomially) simulate SP?
- Can CP* (quasipolynomially) simulate SP*?
- Can SP or CP simulate dag-like SP (R(CP))?
  - → CP cannot p-simulate R(CP) [ABEO2]



Open Problems

- Can CP p-simulate SP*?
- Can CP (quasipolynomially) simulate SP?
- Can CP* (quasipolynomially) simulate SP*?
- Can SP or CP simulate dag-like SP (R(CP))?
  - → CP cannot p-simulate R(CP) [ABEO2]
- Can treelike CP refute systems of $\mathbb{F}_q$ linear equations?



Open Problems

- Can CP p-simulate SP*?
- Can CP (quasipolynomially) simulate SP?
- Can CP* (quasipolynomially) simulate SP*?
- Can SP or CP simulate dag-like SP (R(CP))?
  - → CP cannot p-simulate R(CP) [ABEO2]
- Can treelike CP refute systems of $\mathbb{F}_q$ linear equations?
- SP based solvers?