First steps in the formalization of convex polyhedra in Coq

"Solvers Principles and Architectures" Seminar, Rennes

Xavier Allamigeon[1] Ricardo D. Katz[2]

[1] INRIA and Ecole Polytechnique
[2] CIFACIS – CONICET

October 10th, 2018

Formal proving

Informal definition

Formal proving = "checking" proof using computers
… by writing the proof in a **proof assistant**

Formal proving

Informal definition

Formal proving = "checking" proof using computers
… by writing the proof in a **proof assistant**

Many proof assistants

Agda, Coq, Isabelle, HOL-Light, Lean, Mizar, PVS, etc

- underlying logic
- design
- automation
- libraries

Formal proving

Informal definition

Formal proving = "checking" proof using computers
… by writing the proof in a **proof assistant**

Many proof assistants

Agda, Coq, Isabelle, HOL-Light, Lean, Mizar, PVS, etc

- underlying logic
- design
- automation
- libraries

Recent achievements

- completion of the proof of Kepler conjecture (Hales et al., 2017)
- Feit–Thompson theorem (Gonthier et al., 2013)

Convex polyhedra

Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Convex polyhedra

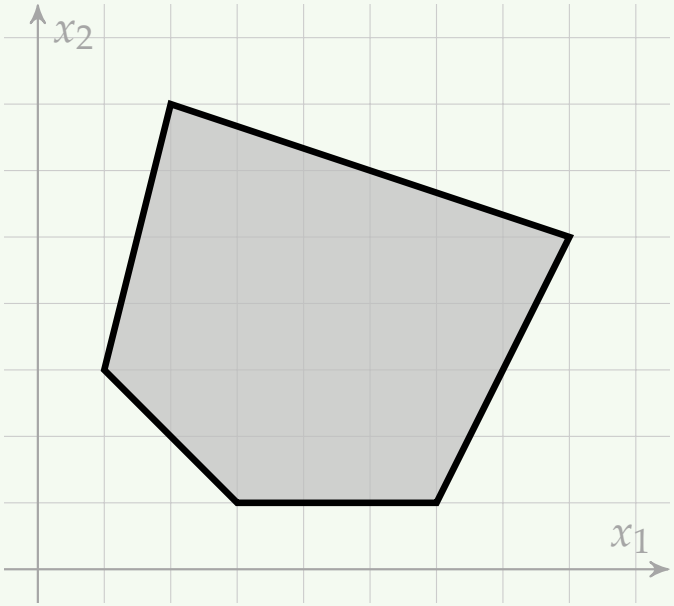
Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Example



$$\begin{pmatrix} 1 & 1 \\ 4 & -1 \\ -1 & -3 \\ -2 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 4 \\ 1 \\ -23 \\ -11 \end{pmatrix}$$

Convex polyhedra

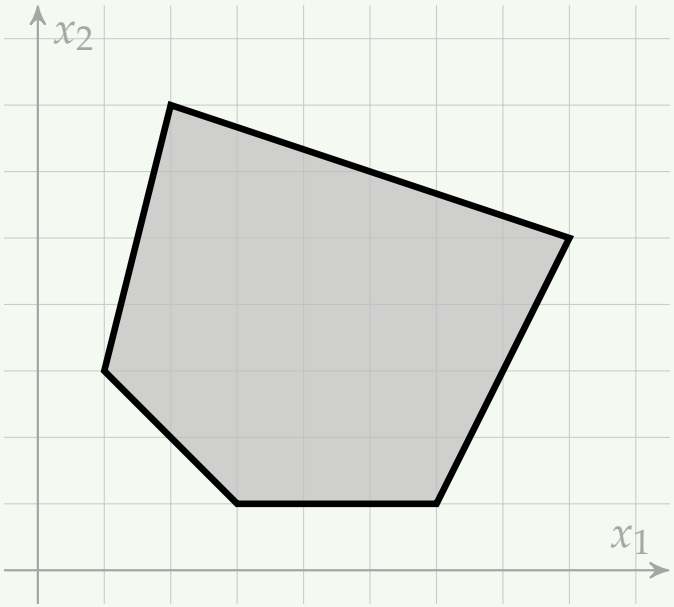
Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Example



$$\begin{aligned} x_1 + x_2 &\geq 4 \\ 4x_1 - x_2 &\geq 1 \\ -x_1 - 3x_2 &\geq -23 \\ -2x_1 + x_2 &\geq -11 \\ x_2 &\geq 1 \end{aligned}$$

Convex polyhedra

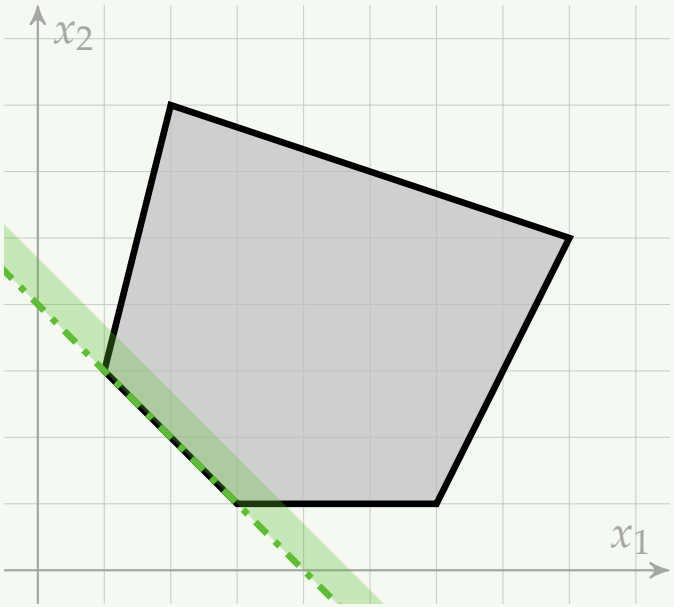
Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Example



$$\begin{aligned} x_1 + x_2 &\geq 4 \\ 4x_1 - x_2 &\geq 1 \\ -x_1 - 3x_2 &\geq -23 \\ -2x_1 + x_2 &\geq -11 \\ x_2 &\geq 1 \end{aligned}$$

Convex polyhedra

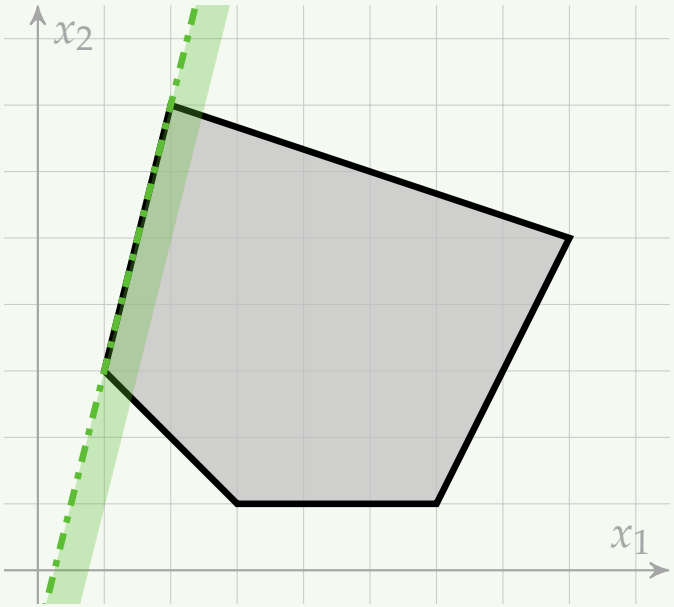
Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Example



$$\begin{aligned} x_1 + x_2 &\geq 4 \\ 4x_1 - x_2 &\geq 1 \\ -x_1 - 3x_2 &\geq -23 \\ -2x_1 + x_2 &\geq -11 \\ x_2 &\geq 1 \end{aligned}$$

Convex polyhedra

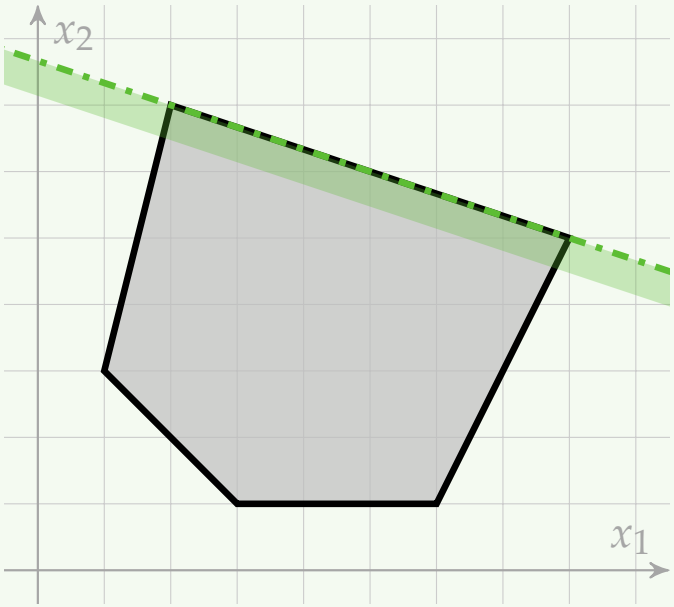
Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Example



$$\begin{aligned} x_1 + x_2 &\geq 4 \\ 4x_1 - x_2 &\geq 1 \\ -x_1 - 3x_2 &\geq -23 \\ -2x_1 + x_2 &\geq -11 \\ x_2 &\geq 1 \end{aligned}$$

Convex polyhedra

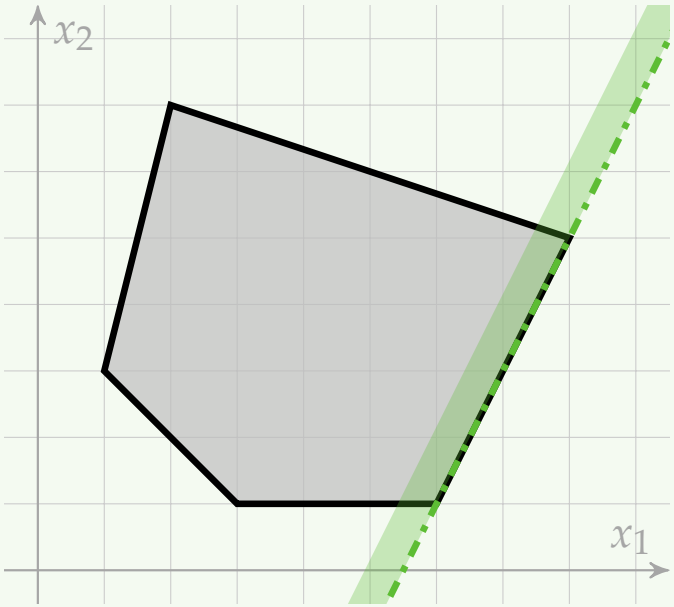
Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Example



$$\begin{aligned} x_1 + x_2 &\geq 4 \\ 4x_1 - x_2 &\geq 1 \\ -x_1 - 3x_2 &\geq -23 \\ -2x_1 + x_2 &\geq -11 \\ x_2 &\geq 1 \end{aligned}$$

Convex polyhedra

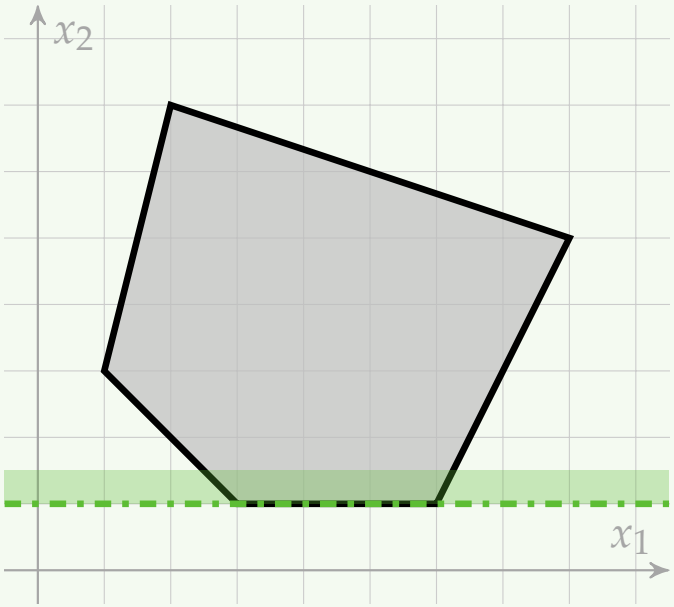
Definition

A **(convex) polyhedron** is the set of the solutions $x \in \mathbb{R}^n$ of a system of linear (affine) inequalities:

$$Ax \geq b$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Example



$$\begin{aligned} x_1 + x_2 &\geq 4 \\ 4x_1 - x_2 &\geq 1 \\ -x_1 - 3x_2 &\geq -23 \\ -2x_1 + x_2 &\geq -11 \\ x_2 &\geq 1 \end{aligned}$$

Why formalizing convex polyhedra?

Pure maths	discrete mathematics, combinatorics, algebraic geometry
Applied maths	optimization, operations research, control theory
CS	computational geometry, software verification, compilation and program optimization, constraint solving

Why formalizing convex polyhedra?

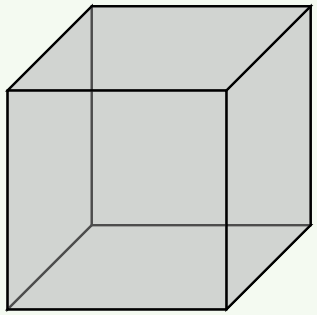
Pure maths	discrete mathematics, **combinatorics**, algebraic geometry
Applied maths	**optimization**, operations research, control theory
CS	computational geometry, software verification, compilation and program optimization, constraint solving

Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube

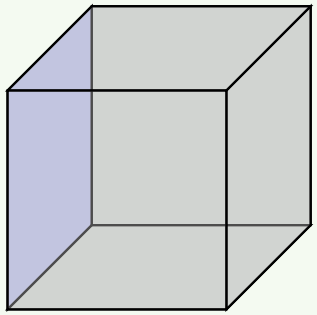


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube

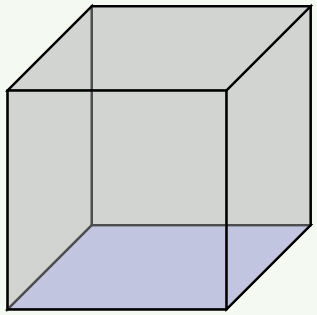


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube

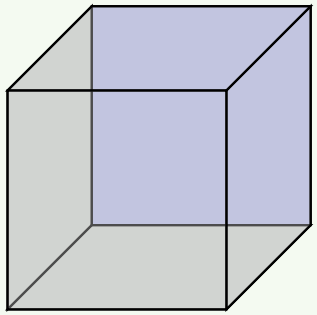


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube

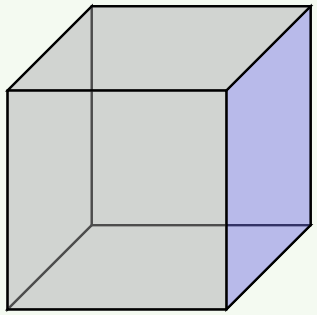


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube

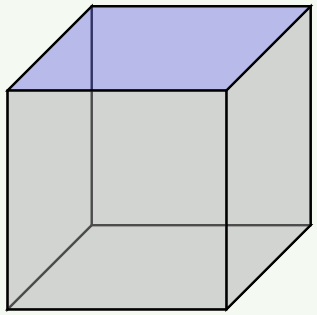


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube

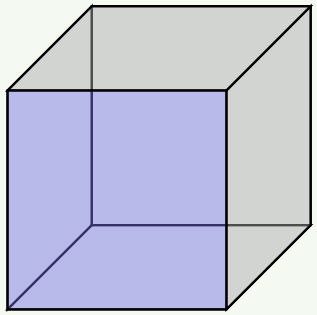


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube

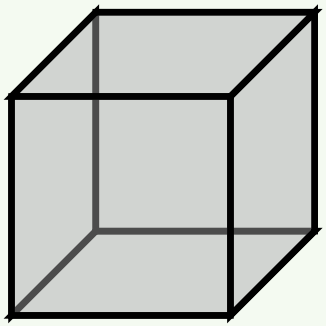


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, **edges**, vertices

The faces of a cube

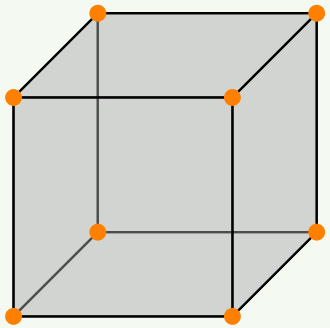


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, **vertices**

The faces of a cube

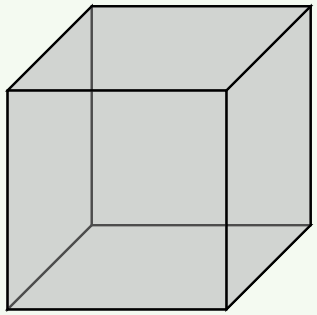


Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube



Diameter of a polyhedron

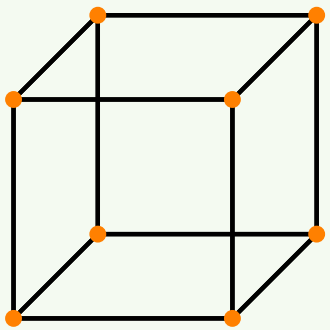
= diameter of the vertex-**edge** graph

Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube



Diameter of a polyhedron

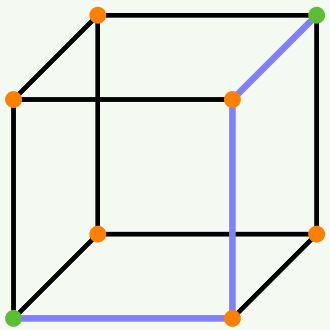
= diameter of the vertex-**edge** graph

Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube



Diameter of a polyhedron

= diameter of the vertex-**edge** graph

Example

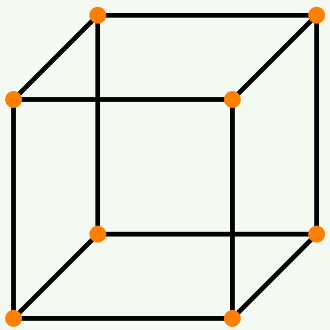
diameter of a 3D-cube $= 3$.

Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube



Diameter of a polyhedron

= diameter of the vertex-**edge** graph

Example

diameter of a 3D-cube $= 3$.

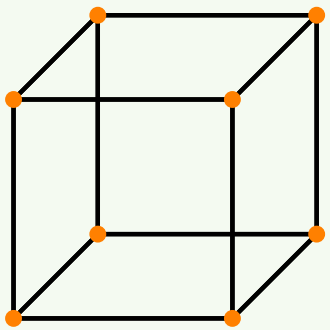
$D(n,f) :=$ maximum diameter of a n-dim. (compact) polyhedron with f facets

Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube



Diameter of a polyhedron

= diameter of the vertex-**edge** graph

Example

diameter of a 3D-cube $= 3$.

$D(n,f) :=$ maximum diameter of a n-dim. (compact) polyhedron with f facets

Hirsch conjecture (1957)

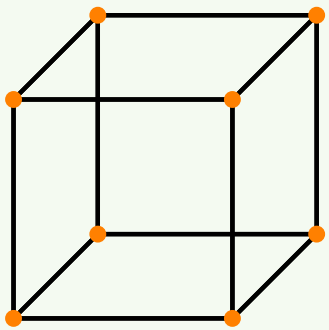
$$D(n,f) \leq f - n$$

Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube



Diameter of a polyhedron

= diameter of the vertex-**edge** graph

Example

diameter of a 3D-cube $= 3$.

$D(n,f) :=$ maximum diameter of a n-dim. (compact) polyhedron with f facets

Hirsch conjecture (1957)

$$D(n,f) \leq f - n$$

…disproved by Santos (2009)

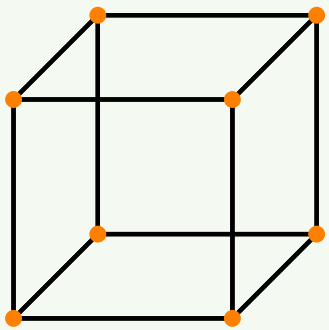
$$D(43,86) \geq 86 - 43 + \mathbf{1}$$

Why convex polyhedra matter… in combinatorics

Convex polyhedra have a rich combinatorial flavor via their faces:

facets, …, edges, vertices

The faces of a cube



Diameter of a polyhedron

= diameter of the vertex-**edge** graph

Example

diameter of a 3D-cube $= 3$.

$D(n,f) :=$ maximum diameter of a n-dim. (compact) polyhedron with f facets

Hirsch conjecture (1957)

$$D(n,f) \leq f - n$$

…disproved by Santos (2009)

$$D(43,86) \geq 86 - 43 + \mathbf{1}$$

Open problem

Is there a **polynomial upper bound** on the diameter?

Why convex polyhedra matter… in optimization

Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Why convex polyhedra matter… in optimization

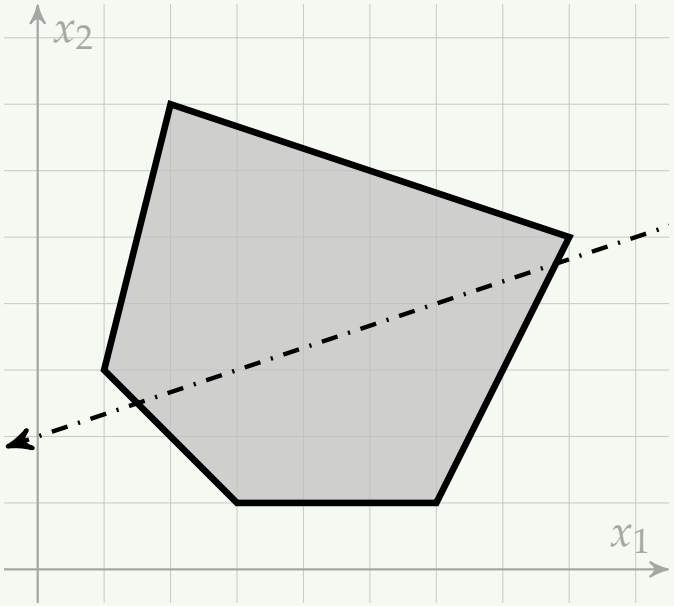
Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Example



$$\begin{aligned} &\text{minimize} && 3x_1 + x_2 \\ &\text{subject to} && x_1 + x_2 \geq 4 \\ & && -x_1 - 3x_2 \geq -23 \\ & && 4x_1 - x_2 \geq 1 \\ & && -2x_1 + x_2 \geq -11 \\ & && x_2 \geq 1 \end{aligned}$$

Why convex polyhedra matter… in optimization

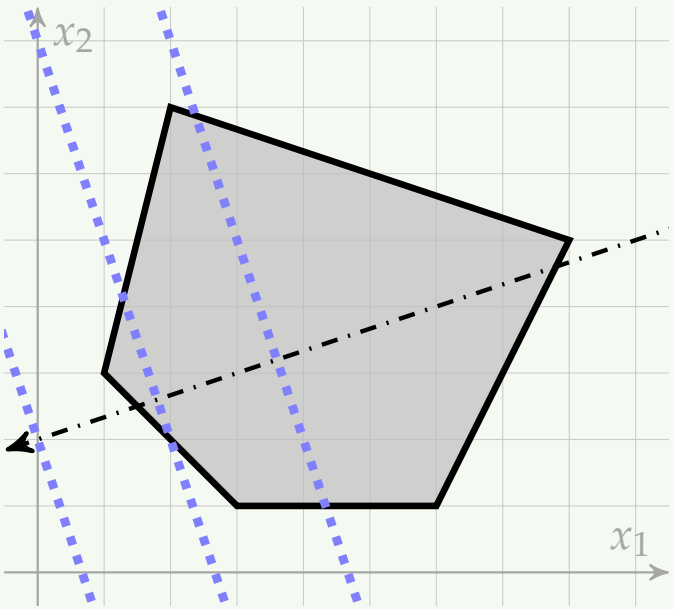
Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Example



$$\begin{aligned} &\text{minimize} && 3x_1 + x_2 \\ &\text{subject to} && x_1 + x_2 \geq 4 \\ & && -x_1 - 3x_2 \geq -23 \\ & && 4x_1 - x_2 \geq 1 \\ & && -2x_1 + x_2 \geq -11 \\ & && x_2 \geq 1 \end{aligned}$$

Why convex polyhedra matter… in optimization

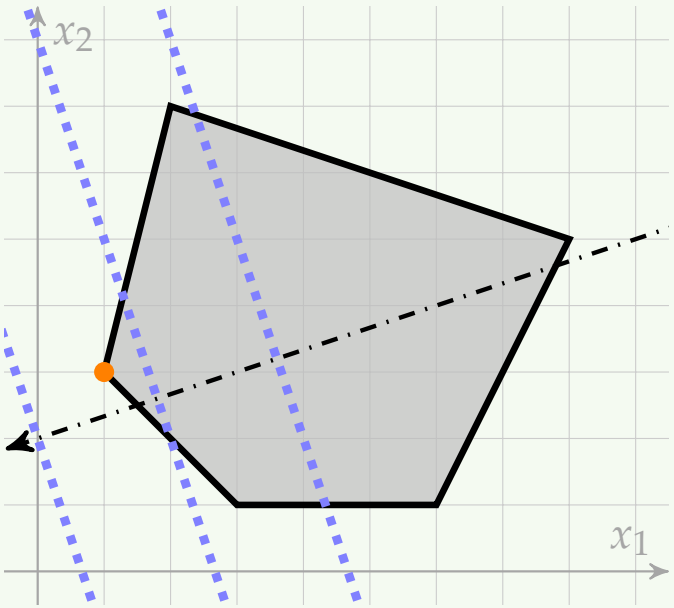
Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Example



$$\begin{aligned} &\text{minimize} && 3x_1 + x_2 \\ &\text{subject to} && x_1 + x_2 \geq 4 \\ & && -x_1 - 3x_2 \geq -23 \\ & && 4x_1 - x_2 \geq 1 \\ & && -2x_1 + x_2 \geq -11 \\ & && x_2 \geq 1 \end{aligned}$$

Why convex polyhedra matter… in optimization

Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Linear programming is **widely** used…

Why convex polyhedra matter… in optimization

Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Linear programming is **widely** used…

because it can be solved in **polynomial time**!!! (Khachiyan, 1980)

Why convex polyhedra matter… in optimization

Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Linear programming is **widely** used…

because it can be solved in **polynomial time**!!! (Khachiyan, 1980)

9th Smale's Problem (Smale, 1998)

Can we solve linear programming in **strongly** polynomial time?

Why convex polyhedra matter… in optimization

Linear programming

= optimizing a linear function over a polyhedron

$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.

Linear programming is **widely** used…

because it can be solved in **polynomial time**!!! (Khachiyan, 1980)

9th Smale's Problem (Smale, 1998)

Can we solve linear programming in **strongly** polynomial time?

- polynomial time
- number of arithmetic operations bounded by a polynomial in the **dimension** of the problem, i.e. $\sim m \times n$.

Why formalizing convex polyhedra?

Pure maths discrete mathematics, combinatorics, algebraic geometry

Applied maths optimization, operations research, control theory

CS computational geometry, software verification, compilation and program optimization, constraint solving

Why formalizing convex polyhedra?

Pure maths	discrete mathematics, combinatorics, algebraic geometry
Applied maths	optimization, operations research, control theory
CS	computational geometry, software verification, compilation and program optimization, constraint solving

Some reasons to formalize convex polyhedra in a proof assistant

- critical applications
  $\Longrightarrow$ increase the level of trust in polyhedral computation

Why formalizing convex polyhedra?

Pure maths	discrete mathematics, combinatorics, algebraic geometry
Applied maths	optimization, operations research, control theory
CS	computational geometry, software verification, compilation and program optimization, constraint solving

Some reasons to formalize convex polyhedra in a proof assistant

- critical applications
  $\Longrightarrow$ increase the level of trust in polyhedral computation
- polyhedra are subject to many open problems

Why formalizing convex polyhedra?

Pure maths	discrete mathematics, combinatorics, algebraic geometry
Applied maths	optimization, operations research, control theory
CS	computational geometry, software verification, compilation and program optimization, constraint solving

Some reasons to formalize convex polyhedra in a proof assistant

- critical applications
  $\Longrightarrow$ increase the level of trust in polyhedral computation
- polyhedra are subject to many open problems
- provide **rigorous** proof of theorems relying on **informal computations**
  $\Longrightarrow$ "formal experimental maths"

Why formalizing… in Coq?

1. Coq is known for its **computational abilities**
   + computing that $a = b$ in Coq is a proof of $a = b$.

Why formalizing… in Coq?

1. Coq is known for its **computational abilities**
   + computing that $a = b$ in Coq is a proof of $a = b$.

A computational motivation

Counterexample of Matschke, Santos, and Weibel (2015) to Hirsch conjecture:
a polytope in dimension 20, with 40 facets and… 36 425 vertices

Why formalizing… in Coq?

1. Coq is known for its **computational abilities**
   $+$ computing that $a = b$ in Coq is a proof of $a = b$.

2. Coq has the **Mathematical Components** library (Gonthier et al., 2016)

MathComp library

- **large hierarchy** of mathematical objets: linear algebra, finite groups, algebraic numbers, representation theory, etc
- introduce **proven methodologies** for formalization of maths in Coq

Why formalizing… in Coq?

1. Coq is known for its **computational abilities**
   $+$ computing that $a = b$ in Coq is a proof of $a = b$.
2. Coq has the **Mathematical Components** library (Gonthier et al., 2016)
3. Challenge: the logic of Coq is intuitionistic, i.e. **constructive**

Why formalizing… in Coq?

1. Coq is known for its **computational abilities**
   + computing that $a = b$ in Coq is a proof of $a = b$.
2. Coq has the **Mathematical Components** library (Gonthier et al., 2016)
3. Challenge: the logic of Coq is intuitionistic, i.e. **constructive**

Constructive logic

- no excluded middle law $P \vee \neg P$

Why formalizing… in Coq?

1. Coq is known for its **computational abilities**
   $+$ computing that $a = b$ in Coq is a proof of $a = b$.
2. Coq has the **Mathematical Components** library (Gonthier et al., 2016)
3. Challenge: the logic of Coq is intuitionistic, i.e. **constructive**

Constructive logic

- no excluded middle law $P \vee \neg P$
- no double negation elimination $\neg\neg P \Rightarrow P$, no proof by contradiction

Why formalizing… in Coq?

1. Coq is known for its **computational abilities**
   $+$ computing that $a = b$ in Coq is a proof of $a = b$.
2. Coq has the **Mathematical Components** library (Gonthier et al., 2016)
3. Challenge: the logic of Coq is intuitionistic, i.e. **constructive**

Constructive logic

- no excluded middle law $P \vee \neg P$
- no double negation elimination $\neg\neg P \Rightarrow P$, no proof by contradiction
- to show $\exists x.\, P(x)$, you need to **construct** an x such that $P(x)$ holds.

This talk

First steps of the formalization of the theory of convex polyhedra, in Coq.

This talk

First steps of the formalization of the theory of convex polyhedra, in Coq.

Main characteristics

It is carried out in an **effective** way:

1. relies on a complete formalization of the simplex method in Coq
2. basic properties of polyhedra (emptiness, boundedness, etc) are defined by means of **Coq programs**, calling the simplex method

This talk

First steps of the formalization of the theory of convex polyhedra, in Coq.

Main characteristics

It is carried out in an **effective** way:

1. relies on a complete formalization of the simplex method in Coq
2. basic properties of polyhedra (emptiness, boundedness, etc) are defined by means of **Coq programs**, calling the simplex method

Outcome

- bypass the intuitionistic nature of Coq

This talk

First steps of the formalization of the theory of convex polyhedra, in Coq.

Main characteristics

It is carried out in an **effective** way:

1. relies on a complete formalization of the simplex method in Coq
2. basic properties of polyhedra (emptiness, boundedness, etc) are defined by means of **Coq programs**, calling the simplex method

Outcome

- bypass the intuitionistic nature of Coq
- this easily provides several essential results on polyhedra:
  Farkas, Minkowski, strong duality, etc

This talk

First steps of the formalization of the theory of convex polyhedra, in Coq.

Main characteristics

It is carried out in an **effective** way:

1. relies on a complete formalization of the simplex method in Coq
2. basic properties of polyhedra (emptiness, boundedness, etc) are defined by means of **Coq programs**, calling the simplex method

Outcome

- bypass the intuitionistic nature of Coq
- this easily provides several essential results on polyhedra:
  Farkas, Minkowski, strong duality, etc

Implementation

The development is gathered in the library **Coq-Polyhedra**:

`github.com/nhojem/Coq-Polyhedra`

Outline of the talk

1. The simplex method: base block of the theory of convex polyhedra
2. Formalization of the simplex method
3. Concluding remarks and perspectives

Outline of the talk

Overcoming the intuitionistic nature of Coq

Formulas of Coq live in the sort `Prop`.

	`Prop`
	`True, False`
conjunction	`A /\ B`
disjunction	`A \/ B`
negation	`~ A`

Overcoming the intuitionistic nature of Coq

Formulas of Coq live in the sort `Prop`.

	`Prop`
	`True, False`
conjunction	`A /\ B`
disjunction	`A \/ B`
negation	`~ A`

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

Overcoming the intuitionistic nature of Coq

Formulas of Coq live in the sort `Prop`.

	`Prop`
	`True, False`
conjunction	`A /\ B`
disjunction	`A \/ B`
negation	`~ A`

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

```
Definition empty A b := ~ (exists x, A *m x >=m b) : Prop.
```

Overcoming the intuitionistic nature of Coq

Formulas of Coq live in the sort `Prop`.

	`Prop`
	`True, False`
conjunction	`A /\ B`
disjunction	`A \/ B`
negation	`~ A`

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

```
Definition empty A b := ~ (exists x, A *m x >=m b) : Prop.
```

Problems

- a polyhedron is empty, or non-empty, or what?
- the formula `~(empty A b)` does not provide a point `x` s.t. `A *m x >=m b`

Overcoming the intuitionistic nature of Coq (2)

Formulas of Coq live in the sort `Prop`.

+ Boolean algebra, i.e., the type `bool := true | false`

	`Prop`	`bool`
	`True, False`	`true, false`
conjunction	`A /\ B`	`a && b`
disjunction	`A \/ B`	`a || b`
negation	`~ A`	`~~ a`

Overcoming the intuitionistic nature of Coq (2)

Formulas of Coq live in the sort `Prop`.

\+ Boolean algebra, i.e., the type `bool := true | false`

	`Prop`	`bool`
	`True, False`	`true, false`
conjunction	`A /\ B`	`a && b`
disjunction	`A \/ B`	`a || b`
negation	`~ A`	`~~ a`

Remark

`bool` behaves like the **classical logic**!

Overcoming the intuitionistic nature of Coq (2)

Formulas of Coq live in the sort `Prop`.
+ Boolean algebra, i.e., the type `bool := true | false`

	`Prop`	`bool`
	`True, False`	`true, false`
conjunction	`A /\ B`	`a && b`
disjunction	`A \/ B`	`a || b`
negation	`~ A`	`~~ a`

Remark

`bool` behaves like the **classical logic**!

Boolean reflection methodology (Gonthier et al., 2016)

`reflect P b` means that `P : Prop` and `b : bool` are equivalent:

- either `P` holds and `b = true`,
- or `~P` holds and `b = false`.

Overcoming the intuitionistic nature of Coq (2)

Formulas of Coq live in the sort `Prop`.
+ Boolean algebra, i.e., the type `bool := true | false`

	`Prop`	`bool`
	`True, False`	`true, false`
conjunction	`A /\ B`	`a && b`
disjunction	`A \/ B`	`a || b`
negation	`~ A`	`~~ a`

Remark

`bool` behaves like the **classical logic**!

Boolean reflection methodology (Gonthier et al., 2016)

`reflect P b` means that `P : Prop` and `b : bool` are equivalent:

- either `P` holds and `b = true`,
- or `~P` holds and `b = false`.

$\Longrightarrow$ The Coq formula `P` can be manipulated like in classical logic! case analysis, etc

Overcoming the intuitionistic nature of Coq (3)

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

```
Definition empty A b := (...) : Prop.
```

Overcoming the intuitionistic nature of Coq (3)

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

```
Definition empty A b := (...) : bool.
```

Overcoming the intuitionistic nature of Coq (3)

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

```
Definition empty A b := (...) : bool.

Theorem emptyPn : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Overcoming the intuitionistic nature of Coq (3)

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

```
Definition empty A b := (...) : bool.

Theorem emptyPn : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Remark

Booleans are **computed** by functions written in the language of Coq!

Overcoming the intuitionistic nature of Coq (3)

Assume that $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.
How to define that the polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty?

```
Definition empty A b := (...) : bool.

Theorem emptyPn : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Remark

Booleans are **computed** by functions written in the language of Coq!

Solution

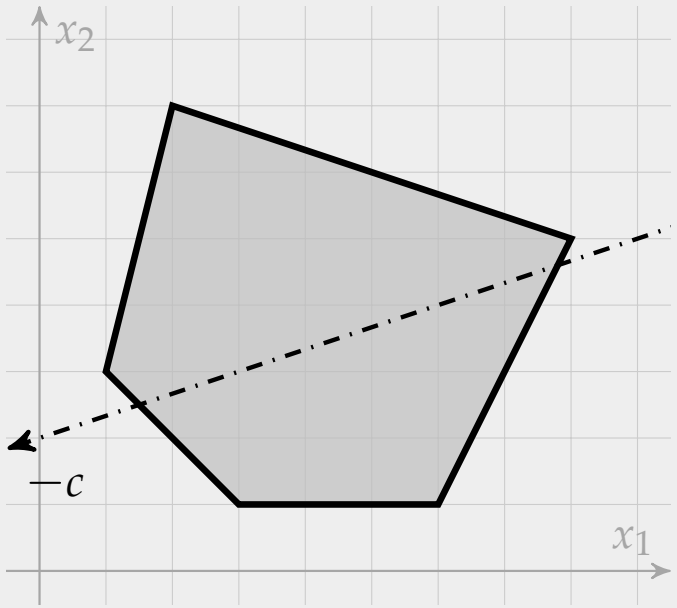
Implement **decision procedures** for emptiness, boundedness, membership, etc
$\Longrightarrow$ one building block: the simplex method in Coq

The simplex method

Linear programming

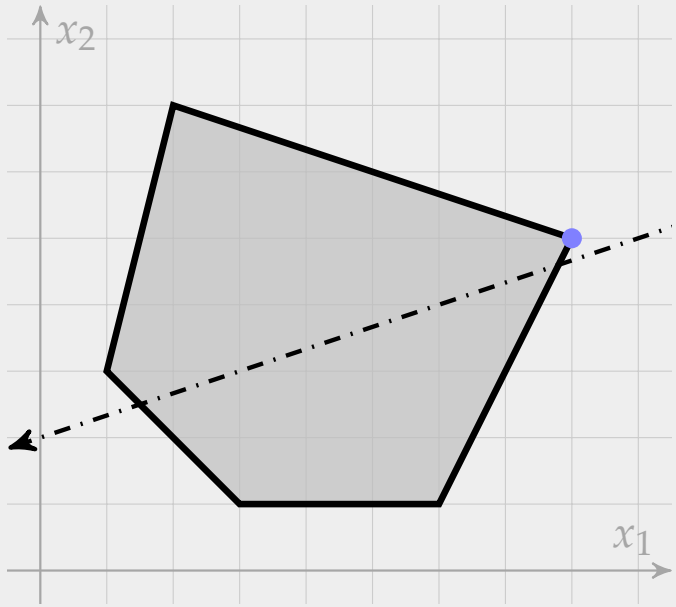
$$\begin{aligned} &\text{minimize} && \langle c, x \rangle \\ &\text{subject to} && Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, and $\langle c, x \rangle := \sum_{i=1}^{n} c_i x_i$.



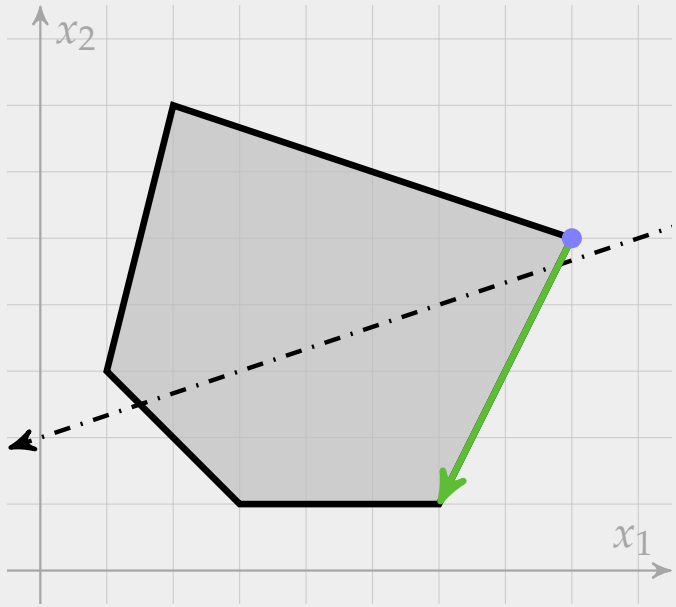
The simplex method (2)

1. starting from an initial vertex



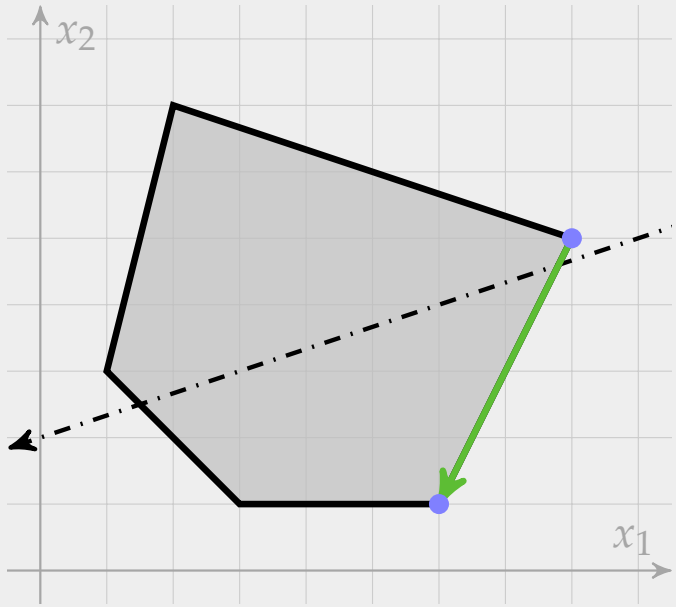
The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function



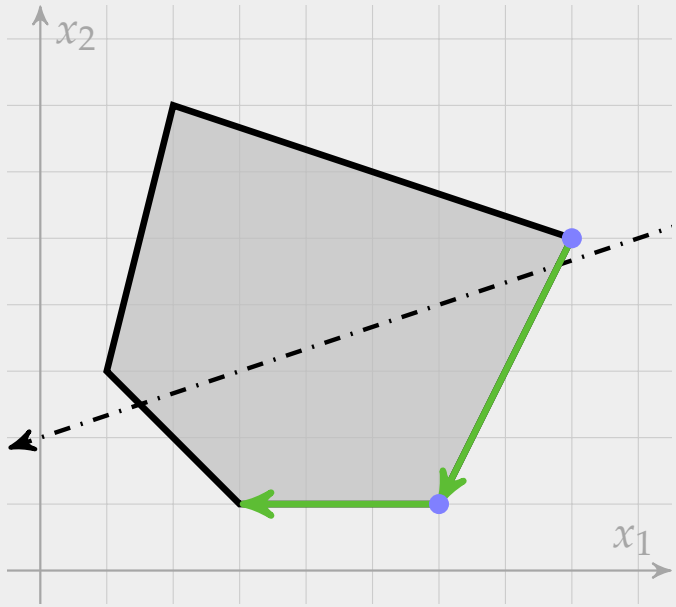
The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function



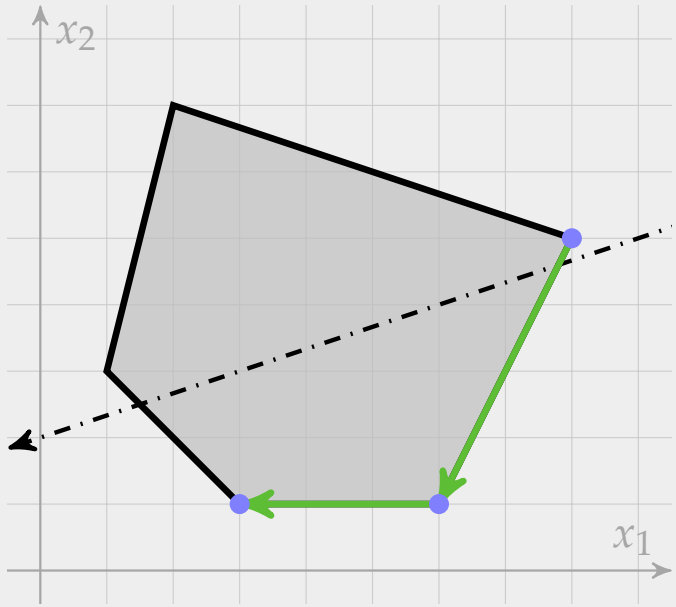
The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function



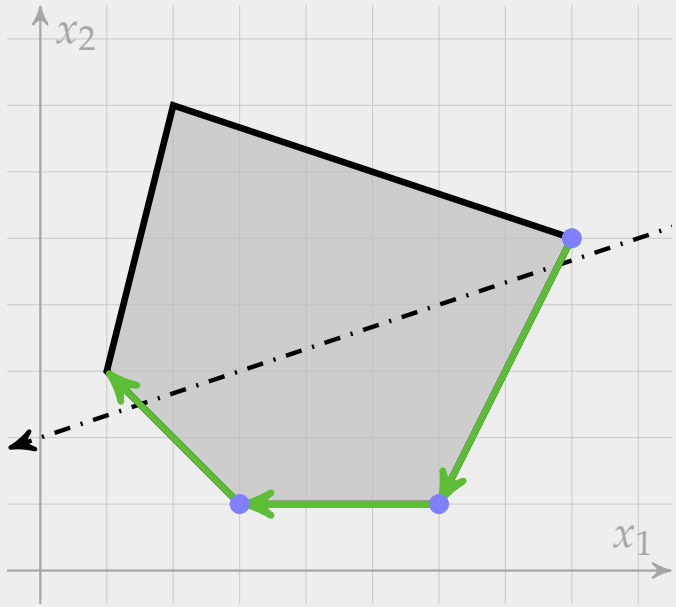
The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function



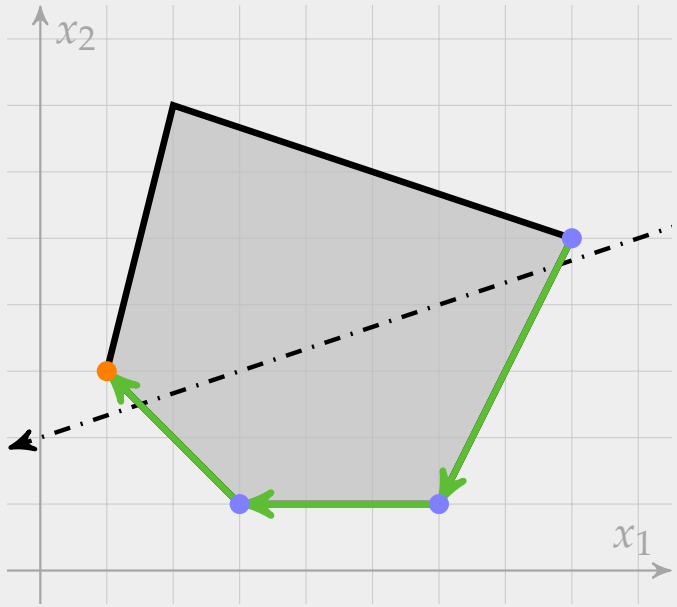
The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function



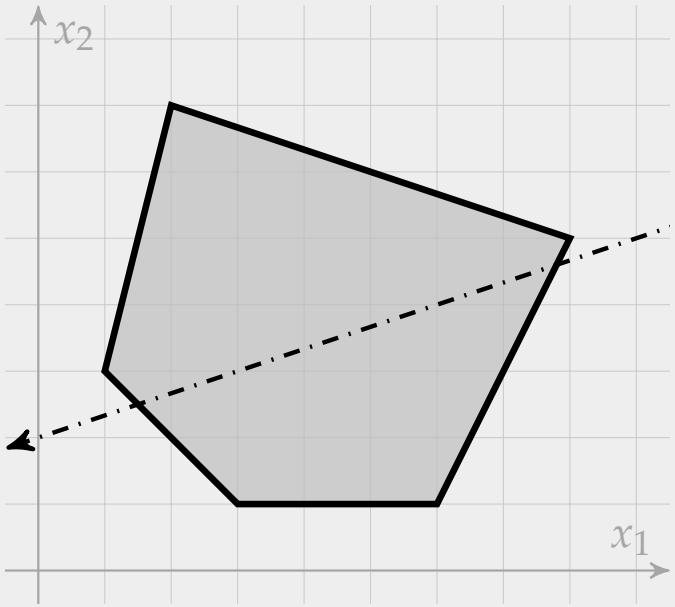
The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function
3. up to finding an optimal vertex



The simplex method (2)

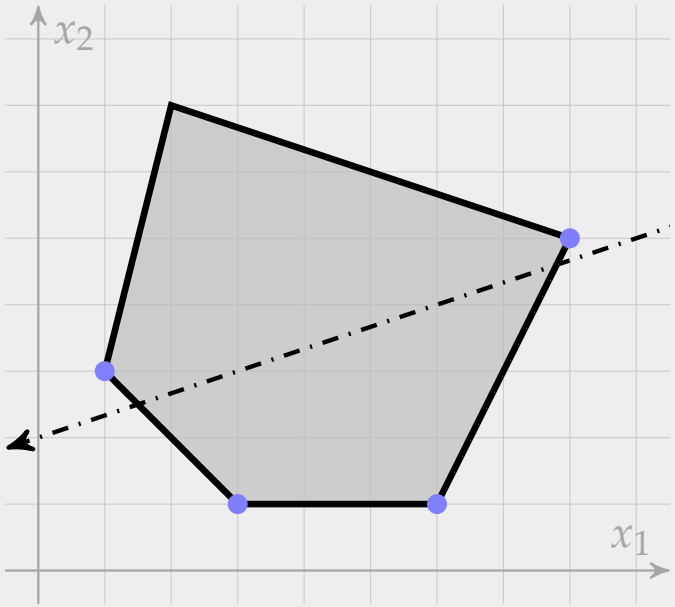
1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function
3. up to finding an optimal vertex



Three ingredients

The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function
3. up to finding an optimal vertex



Three ingredients

- bases encode vertices

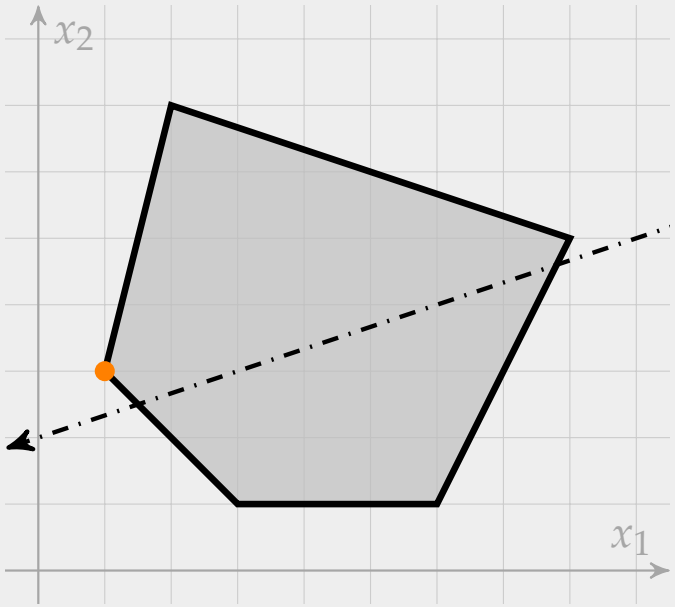
Definition

A **basis** is a subset $I \subset [m]$ of cardinality n such that the rows $(A_i)_{i \in I}$ are linearly independent.
The associated **basic point** is defined as:

$$x^I := A_I^{-1} b_I \,.$$

The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function
3. up to finding an optimal vertex



Three ingredients

- bases encode vertices
- reduced costs determine optimality

Reduced cost vector

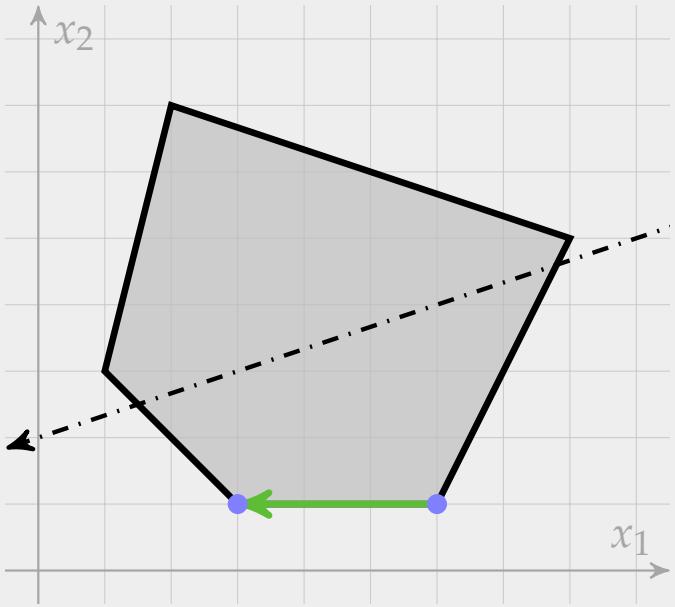
The **reduced cost vector** at basis I is defined as $u^I := A_I^{-T} c_I$.
When **non-negative**, it provides a feasible element of the dual LP:

$$\text{maximize} \quad \langle b, u \rangle \quad \text{subject to} \quad A^T u = c\,,\ u \geq 0\,,\ u \in \mathbb{R}^m$$

which satisfies $\langle b, u^I \rangle = \langle c, x^I \rangle$.

The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function
3. up to finding an optimal vertex

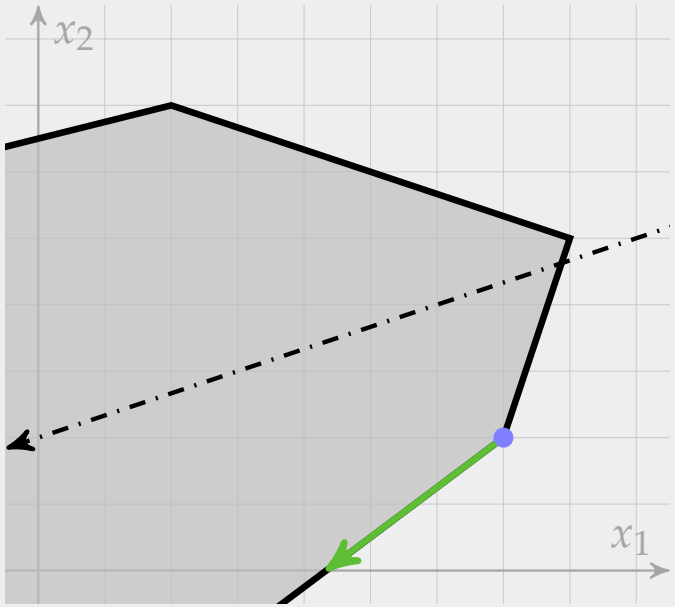


Three ingredients

- bases encode vertices
- reduced costs determine optimality
- pivoting switches from a vertex to another

The simplex method (2)

1. starting from an initial vertex
2. iterate over the vertex-edge graph while decreasing the objective function
3. up to finding an optimal vertex



Three ingredients

- bases encode vertices
- reduced costs determine optimality
- pivoting switches from a vertex to another **or** determines if the LP is unbounded

Emptiness as a Boolean predicate

Emptiness as a Boolean predicate

Farkas Lemma

The polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty if, and only if,

- there exists $d \in \mathbb{R}^m$ such that $A^T d = 0$ and $\langle b, d \rangle > 0$

Emptiness as a Boolean predicate

Farkas Lemma

The polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty if, and only if,

- there exists $d \in \mathbb{R}^m$ such that $A^T d = 0$ and $\langle b, d \rangle > 0$
- or, equivalently, the following LP is unbounded:

$$\text{maximize} \quad \langle b, u \rangle \quad \text{subject to} \quad A^T u = 0\,,\ u \geq 0\,,\ u \in \mathbb{R}^m \tag{1}$$

Emptiness as a Boolean predicate

Farkas Lemma

The polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty if, and only if,

- there exists $d \in \mathbb{R}^m$ such that $A^T d = 0$ and $\langle b, d \rangle > 0$
- or, equivalently, the following LP is unbounded:

$$\text{maximize} \quad \langle b, u \rangle \quad \text{subject to} \quad A^T u = 0\,,\ u \geq 0\,,\ u \in \mathbb{R}^m \tag{1}$$

Suppose that we have formalized the simplex method in Coq:

```
Definition empty A b :=
  match simplex [A^T -A^T 1:%M] 0 (-b) with
  | Unbounded d => true
  | Optimal_basis I => false
  end.
```

Emptiness as a Boolean predicate

Farkas Lemma

The polyhedron $\{x \in \mathbb{R}^n : Ax \geq b\}$ is empty if, and only if,

- there exists $d \in \mathbb{R}^m$ such that $A^T d = 0$ and $\langle b, d \rangle > 0$
- or, equivalently, the following LP is unbounded:

$$\text{maximize} \quad \langle b, u \rangle \quad \text{subject to} \quad A^T u = 0\,,\ u \geq 0\,,\ u \in \mathbb{R}^m \tag{1}$$

Suppose that we have formalized the simplex method in Coq:

```
Definition empty A b :=
  match simplex [A^T -A^T 1:%M] 0 (-b) with
  | Unbounded d => true
  | Optimal_basis I => false
  end.
```

We get Farkas Lemma for free:

```
Theorem emptyP A b :
  reflect (exists d, A^T *m d = 0 /\ '[b,d] > 0) (empty A b).
```

Emptiness as a Boolean predicate (2)

```
Definition empty A b :=
  match phase2 [A^T -A^T 1:%M] ... with
  | Unbounded d => true
  | Optimal_basis I => false
  end.
```

It remains to prove:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Emptiness as a Boolean predicate (2)

```
Definition empty A b :=
  match phase2 [A^T -A^T 1:%M] ... with
  | Unbounded d => true
  | Optimal_basis I => false
  end.
```

It remains to prove:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Proof sketch

Follows from the correctness of the simplex method:

if `phase2` returns `Optimal_basis I`, then `reduced_cost I >=m 0`

Emptiness as a Boolean predicate (2)

```
Definition empty A b :=
  match phase2 [A^T -A^T 1:%M] ... with
  | Unbounded d => true
  | Optimal_basis I => false
  end.
```

It remains to prove:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Proof sketch

Follows from the correctness of the simplex method:

if `phase2` returns `Optimal_basis I`, then `reduced_cost I >=m 0`

$\Longrightarrow$ the point `x` is built from `reduced_cost I`, as a feasible point of

the dual of maximize $\langle b, u \rangle$ subject to $A^T u = 0\,,\ u \geq 0\,,\ u \in \mathbb{R}^m$

Emptiness as a Boolean predicate (2)

```
Definition empty A b :=
  match phase2 [A^T -A^T 1:%M] ... with
  | Unbounded d => true
  | Optimal_basis I => false
  end.
```

It remains to prove:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Proof sketch

Follows from the correctness of the simplex method:

if `phase2` returns `Optimal_basis I`, then `reduced_cost I >=m 0`

$\Longrightarrow$ the point `x` is built from `reduced_cost I`, as a feasible point of

$$\text{the dual of} \quad \text{maximize} \quad \langle b, u \rangle \quad \text{subject to} \quad A^T u = 0\,,\ u \geq 0\,,\ u \in \mathbb{R}^m$$
$$\equiv \quad \text{minimize} \quad 0 \quad \text{subject to} \quad Ax \geq b\,,\ x \in \mathbb{R}^n$$

```
Qed.
```

The simplex method is a constructive Swiss knife

Primal LP

$$\begin{aligned} \text{minimize} \quad & \langle c, x \rangle \\ \text{subject to} \quad & Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

Dual LP

$$\begin{aligned} \text{maximize} \quad & \langle b, u \rangle \\ \text{subject to} \quad & A^T u = c\,,\ u \geq 0\,,\ u \in \mathbb{R}^m \end{aligned}$$

The simplex method is a constructive Swiss knife

Primal LP

$$\begin{aligned} \text{minimize} \quad & \langle c, x \rangle \\ \text{subject to} \quad & Ax \geq b\,,\ x \in \mathbb{R}^n \end{aligned}$$

Dual LP

$$\begin{aligned} \text{maximize} \quad & \langle b, u \rangle \\ \text{subject to} \quad & A^T u = c\,,\ u \geq 0\,,\ u \in \mathbb{R}^m \end{aligned}$$

Theorem (Strong duality)

If both LP are feasible, there exists optimal solutions x^* and u^* of the primal and dual LP such that:

$$\langle c, x^* \rangle = \langle b, u^* \rangle$$

The simplex method is a constructive Swiss knife

Primal LP		**Dual LP**	
minimize	$\langle c, x \rangle$	maximize	$\langle b, u \rangle$
subject to	$Ax \geq b\,,\ x \in \mathbb{R}^n$	subject to	$A^T u = c\,,\ u \geq 0\,,\ u \in \mathbb{R}^m$

Theorem (Strong duality)

If both LP are feasible, there exists optimal solutions x^* and u^* of the primal and dual LP such that:

$$\langle c, x^* \rangle = \langle b, u^* \rangle$$

Proof sketch

The simplex returns an optimal basis I:

- `point_of_basis I` provides an optimal point of the primal;
- `reduced_cost I >=m 0` provides an optimal point of the dual.

The simplex method is a constructive Swiss knife (2)

Other properties can be defined as Boolean predicate thanks to the simplex method: boundedness, membership to the convex hull, …

The simplex method is a constructive Swiss knife (2)

Other properties can be defined as Boolean predicate thanks to the simplex method: boundedness, **membership to the convex hull**, …

Definition

A point $x \in \mathbb{R}^n$ belongs to the convex hull of the set $V = \{v^1, \ldots, v^p\}$ if

$$\exists \lambda \in \mathbb{R}^p\,, \quad x = \sum_{i=1}^{p} \lambda_i v^i \quad \text{where } \lambda \geq 0\,,\ \sum_{i=1}^{p} \lambda_i = 1\,.$$

The simplex method is a constructive Swiss knife (2)

Other properties can be defined as Boolean predicate thanks to the simplex method: boundedness, **membership to the convex hull**, …

Definition

A point $x \in \mathbb{R}^n$ belongs to the convex hull of the set $V = \{v^1, \ldots, v^p\}$ if

$$\exists \lambda \in \mathbb{R}^p, \quad x = \sum_{i=1}^{p} \lambda_i v^i \quad \text{where } \lambda \geq 0, \ \sum_{i=1}^{p} \lambda_i = 1.$$

$\Longrightarrow$ membership amounts to the non-emptiness of a polyhedron in $\lambda \in \mathbb{R}^p$ (parametrized by x and V)

```
Definition is_in_convex_hull (x:'cV_n) :=
  let Ax := [V -V e^T -e^T 1%:M] in
  let bx := [x -x 1 -1 0] in
  ~~ (empty Ax bx) : bool.
```

The simplex method is a constructive Swiss knife (2)

Other properties can be defined as Boolean predicate thanks to the simplex method: boundedness, **membership to the convex hull**, …

Definition

A point $x \in \mathbb{R}^n$ belongs to the convex hull of the set $V = \{v^1, \ldots, v^p\}$ if

$$\exists \lambda \in \mathbb{R}^p, \quad x = \sum_{i=1}^{p} \lambda_i v^i \quad \text{where } \lambda \geq 0, \ \sum_{i=1}^{p} \lambda_i = 1 .$$

$\Longrightarrow$ membership amounts to the non-emptiness of a polyhedron in $\lambda \in \mathbb{R}^p$ (parametrized by x and V)

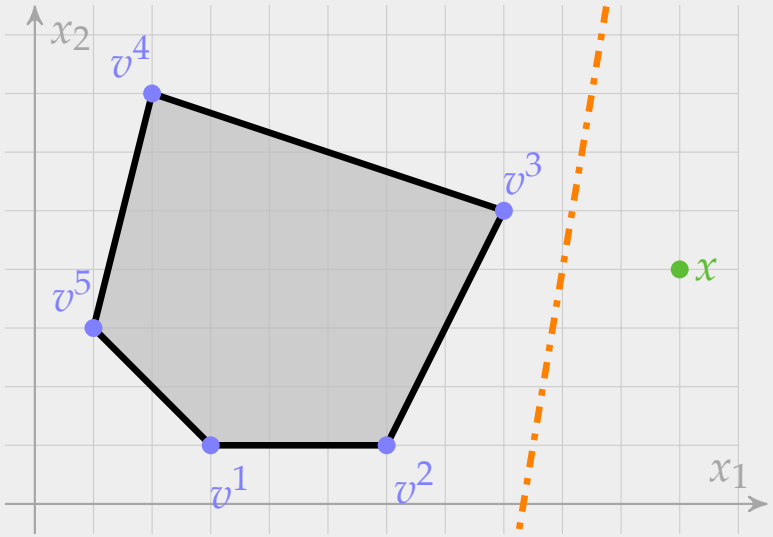
```
Definition is_in_convex_hull (x:'cV_n) :=
  let Ax := [V -V e^T -e^T 1%:M] in
  let bx := [x -x 1 -1 0] in
  ~~ (empty Ax bx) : bool.

Lemma is_in_convex_hullP (x:'cV_n) :
  reflect
    (exists lambda,
      [/\ (lambda >=m 0), '[e, lambda] = 1 & x = V *m lambda])
    (is_in_convex_hull x).
```

The simplex method is a constructive Swiss knife (3)

Theorem (Separation result)

*The point x **does not** belong to the convex hull of V iff there exists $c \in \mathbb{R}^n$ such that*

$$\langle c, v^i \rangle > \langle c, x \rangle \ , \quad i = 1, \ldots, p$$

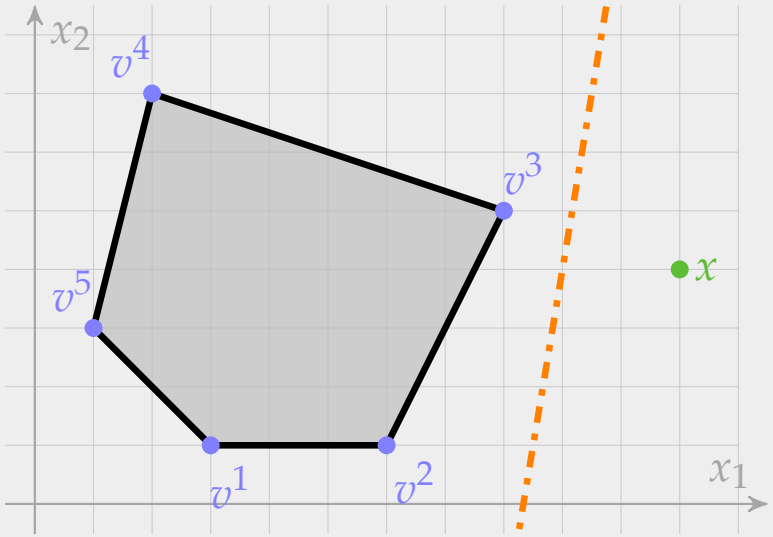


The simplex method is a constructive Swiss knife (3)

Theorem (Separation result)

*The point x **does not** belong to the convex hull of V iff there exists $c \in \mathbb{R}^n$ such that*

$$\langle c, v^i \rangle > \langle c, x \rangle \ , \quad i = 1, \ldots, p$$




```
Theorem separation x :
  reflect (exists c, forall i, '[c, col i V] > '[c, x])
          ~~ (is_in_convex_hull x).
```

Proof sketch

The normal vector `c` is built from the (Farkas) emptiness certificate `d` of the polyhedron over $\lambda \in \mathbb{R}^p$.

The simplex method is a constructive Swiss knife (4)

Theorem (Minkowski)

Every bounded polyhedron is the convex hull of finitely many points.

The simplex method is a constructive Swiss knife (4)

Theorem (Minkowski)

Every bounded polyhedron is the convex hull of finitely many points.

```
Theorem minkowski : bounded_polyhedron A b ->
  polyhedron A b =i is_in_convex_hull matrix_of_points.
```

where:

- `=i` is the extensional equality
- `matrix_of_points` is the matrix of the feasible basic points

The simplex method is a constructive Swiss knife (4)

Theorem (Minkowski)

Every bounded polyhedron is the convex hull of finitely many points.

```
Theorem minkowski : bounded_polyhedron A b ->
  polyhedron A b =i is_in_convex_hull matrix_of_points.
```

where:

- `=i` is the extensional equality
- `matrix_of_points` is the matrix of the feasible basic points

Proof sketch

Suppose that x lies in the polyhedron.

If x does not belong to the convex hull of the basic points, there exists c such that

$$\langle c, x \rangle < \langle c, z \rangle \quad \text{for all feasible basic point } z$$

The simplex method is a constructive Swiss knife (4)

Theorem (Minkowski)

Every bounded polyhedron is the convex hull of finitely many points.

```
Theorem minkowski : bounded_polyhedron A b ->
  polyhedron A b =i is_in_convex_hull matrix_of_points.
```

where:

- `=i` is the extensional equality
- `matrix_of_points` is the matrix of the feasible basic points

Proof sketch

Suppose that x lies in the polyhedron.

If x does not belong to the convex hull of the basic points, there exists c such that

$$\langle c, z^* \rangle \leq \langle c, x \rangle < \langle c, z \rangle \quad \text{for all feasible basic point } z$$

where z^* is the basic point found by the simplex method.

`Qed.`

Outline of the talk

Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.

Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.

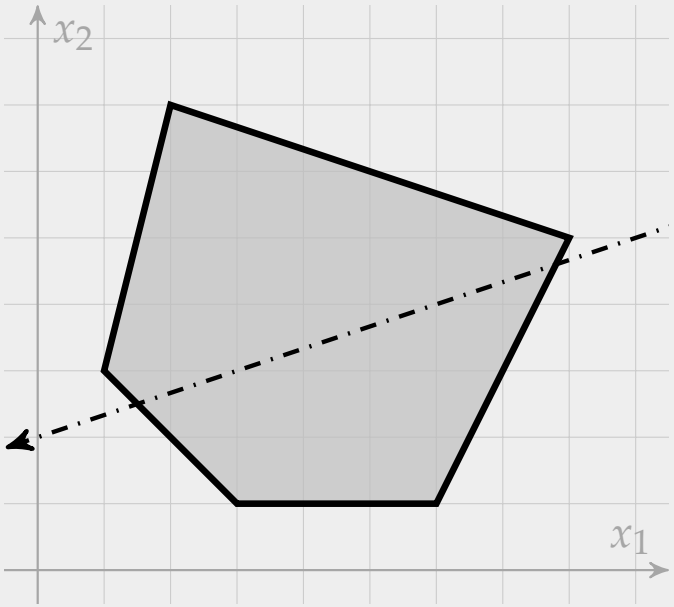
Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.



$$\begin{aligned}
\text{minimize} \quad & 3x_1 + x_2 \\
\text{subject to} \quad & x_1 + x_2 \geq 4 && \text{①} \\
& -x_1 - 3x_2 \geq -23 && \text{②} \\
& 4x_1 - x_2 \geq 1 && \text{③} \\
& -2x_1 + x_2 \geq -11 && \text{④} \\
& x_2 \geq 1 && \text{⑤}
\end{aligned}$$

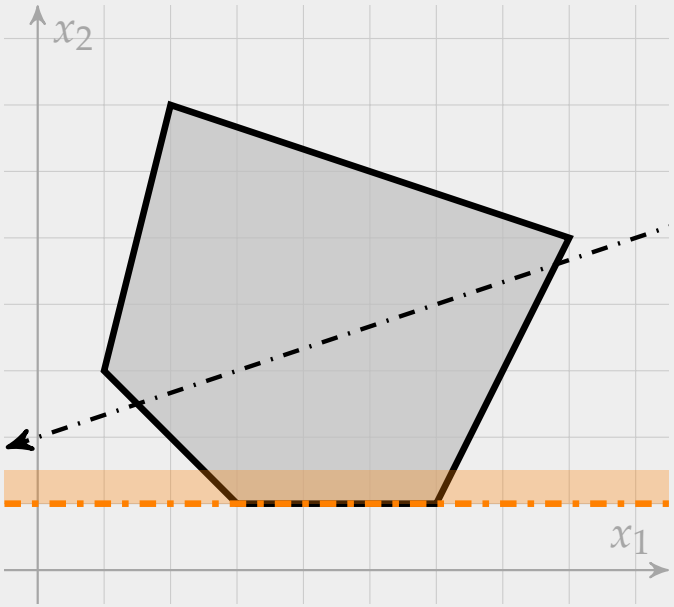
Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.



$$\begin{aligned}
\text{minimize} \quad & 3x_1 + x_2 \\
\text{subject to} \quad & x_1 + x_2 \geq 4 && \text{①} \\
& -x_1 - 3x_2 \geq -23 && \text{②} \\
& 4x_1 - x_2 \geq 1 && \text{③} \\
& -2x_1 + x_2 \geq -11 && \text{④} \\
& x_2 \geq 1 && \text{⑤}
\end{aligned}$$

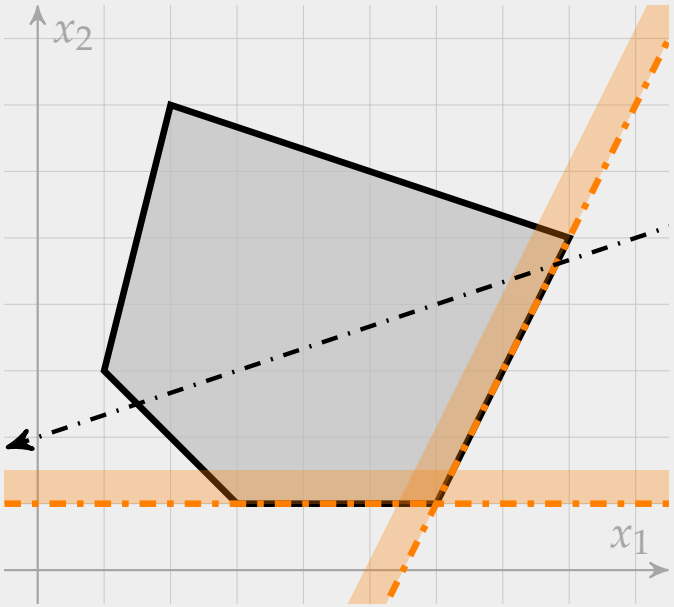
Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.



$$\begin{aligned}
\text{minimize} \quad & 3x_1 + x_2 \\
\text{subject to} \quad & x_1 + x_2 \geq 4 && \text{①} \\
& -x_1 - 3x_2 \geq -23 && \text{②} \\
& 4x_1 - x_2 \geq 1 && \text{③} \\
& -2x_1 + x_2 \geq -11 && \text{④} \\
& x_2 \geq 1 && \text{⑤}
\end{aligned}$$

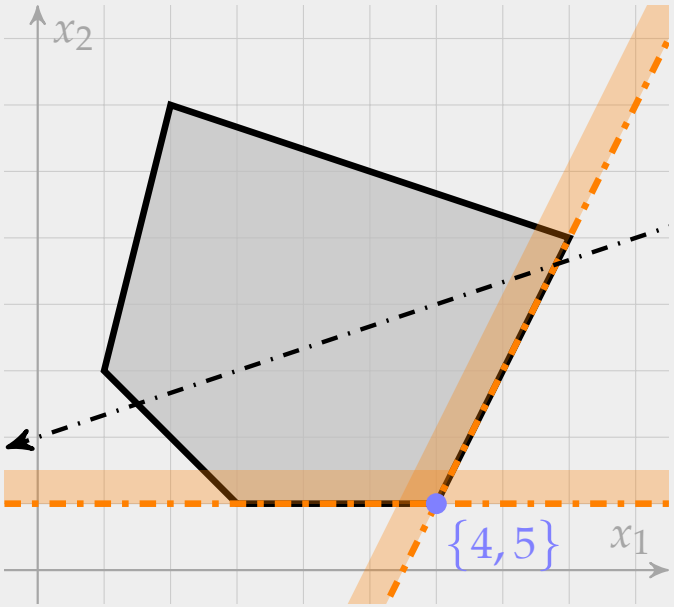
Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.



$$\begin{aligned}
\text{minimize} \quad & 3x_1 + x_2 \\
\text{subject to} \quad & x_1 + x_2 \geq 4 && \text{①} \\
& -x_1 - 3x_2 \geq -23 && \text{②} \\
& 4x_1 - x_2 \geq 1 && \text{③} \\
& -2x_1 + x_2 \geq -11 && \text{④} \\
& x_2 \geq 1 && \text{⑤}
\end{aligned}$$

Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.

Bases are formalized via three layers of types:

Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.

Bases are formalized via three layers of types:

```
Inductive prebasis := Prebasis (I: {set 'I_m}) of (#|I| == n).
```

Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.

Bases are formalized via three layers of types:

```
Inductive prebasis := Prebasis (I: {set 'I_m}) of (#|I| == n).

Inductive basis :=
  Basis (I:prebasis) of row_free (row_submx A I).
```

= submatrix A_I

Bases

Definition

A **basis** is a subset $I \subset \{1, \ldots, m\}$ of cardinality n such that the system

$$A_i x = b_i\,, \quad i \in I$$

has a unique solution, called the **basic point**.
The basis is **feasible** when the basic point belongs to the polyhedron.

Bases are formalized via three layers of types:

```
Inductive prebasis := Prebasis (I: {set 'I_m}) of (#|I| == n).

Inductive basis :=
  Basis (I:prebasis) of row_free (row_submx A I).

Inductive feasible_basis :=
  FeasibleBasis (I:basis) of point_of_basis I \in polyhedron A b
```

where we have defined:

```
Definition point_of_basis (I:basis) :=
  (qinvmx [...] (row_submx A I)) *m (row_submx b I).
```

Reduced costs: optimality certificate

```
Variable I : feasible_basis.
```

Reduced costs: optimality certificate

```
Variable I : feasible_basis.
```

Definition

The **reduced cost vector** at basis I is defined as the unique solution $u \in \mathbb{R}^I$ of the system

$$(A_I)^T u = c\,.$$

Reduced costs: optimality certificate

```
Variable I : feasible_basis.
```

Definition

The **reduced cost vector** at basis I is defined as the unique solution $u \in \mathbb{R}^I$ of the system

$$(A_I)^T u = c\,.$$

In Coq, this simply writes as:

```
Definition reduced_cost :=
  (qinvmx [...] (row_submx A I))^T *m c.
```

Reduced costs: optimality certificate

```
Variable I : feasible_basis.
```

Definition

The **reduced cost vector** at basis I is defined as the unique solution $u \in \mathbb{R}^I$ of the system

$$(A_I)^T u = c\,.$$

In Coq, this simply writes as:

```
Definition reduced_cost :=
  (qinvmx [...] (row_submx A I))^T *m c.

Lemma optimality_certificate :
  reduced_cost >=m 0 ->
    forall x, x \in polyhedron A b ->
      '[c, point_of_basis I] <= '[c, x].
```

Reduced costs: optimality certificate

```
Variable I : feasible_basis.
```

Definition

The **reduced cost vector** at basis I is defined as the unique solution $u \in \mathbb{R}^I$ of the system

$$(A_I)^T u = c\,.$$

In Coq, this simply writes as:

```
Definition reduced_cost :=
  (qinvmx [...] (row_submx A I))^T *m c.

Lemma optimality_certificate :
  reduced_cost >=m 0 ->
    forall x, x \in polyhedron A b ->
      '[c, point_of_basis I] <= '[c, x].
```

Reduced costs: optimality certificate

```
Variable I : feasible_basis.
```

Definition

The **reduced cost vector** at basis I is defined as the unique solution $u \in \mathbb{R}^I$ of the system

$$(A_I)^T u = c\,.$$

In Coq, this simply writes as:

```
Definition reduced_cost :=
  (qinvmx [...] (row_submx A I))^T *m c.

Lemma optimality_certificate :
  reduced_cost >=m 0 ->
    forall x, x \in polyhedron A b ->
      '[c, point_of_basis I] <= '[c, x].
```

Proof sketch

$$\langle c, x\rangle = \langle u, A_I x\rangle = \langle \underbrace{u}_{\geq 0}, \underbrace{A_I x - b_I}_{\geq 0}\rangle + \langle u, b_I\rangle$$

Reduced costs: optimality certificate

```
Variable I : feasible_basis.
```

Definition

The **reduced cost vector** at basis I is defined as the unique solution $u \in \mathbb{R}^I$ of the system

$$(A_I)^T u = c\,.$$

In Coq, this simply writes as:

```
Definition reduced_cost :=
  (qinvmx [...] (row_submx A I))^T *m c.

Lemma optimality_certificate :
  reduced_cost >=m 0 ->
    forall x, x \in polyhedron A b ->
      '[c, point_of_basis I] <= '[c, x].
```

Proof sketch

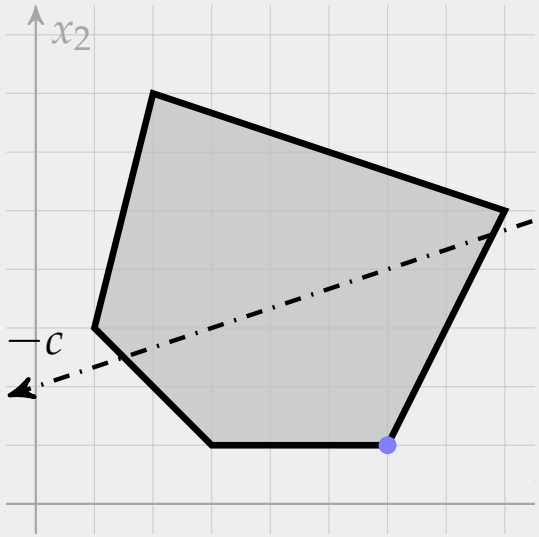
$$\langle c, x\rangle = \langle u, A_I x\rangle = \langle \underbrace{u}_{\geq 0}, \underbrace{A_I x - b_I}_{\geq 0}\rangle + \langle u, b_I\rangle \geq \langle u, b_I\rangle = \langle c, x^I\rangle$$

```
Qed.
```

Pivoting

If the reduced cost vector has some negative entries:

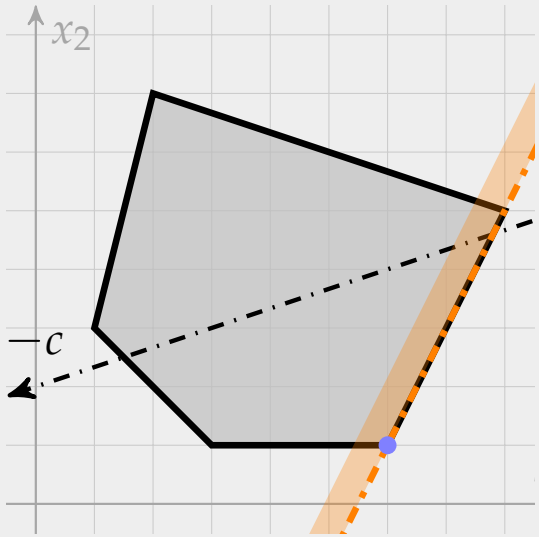
```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```



Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```



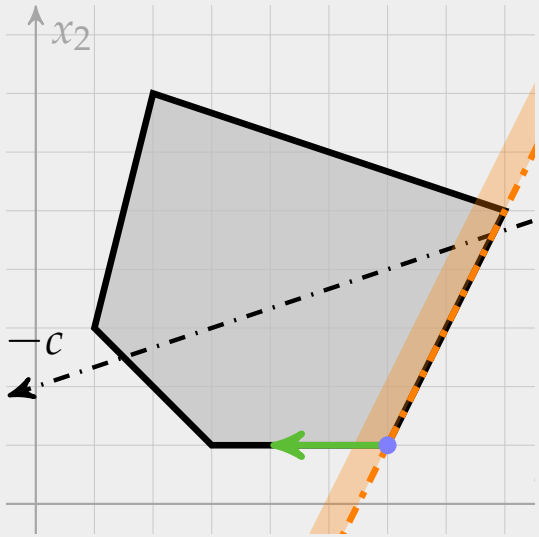
Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```

we can build a direction vector which

- follows an incident edge
- decreases the objective function



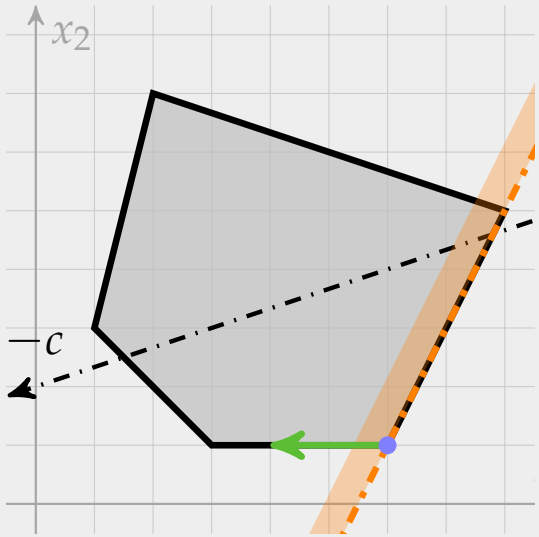
Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```

we can build a direction vector which

- follows an incident edge
- decreases the objective function




```
Definition direction :=
  let: ei := (delta_mx i 0) in
  (qinvmx [...] (row_submx A I)) *m ei.

Lemma direction_improvement :
  '[c, direction] < 0.
```

Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```

we can build a direction vector which

- follows an incident edge
- decreases the objective function

At this stage, two possibilities:

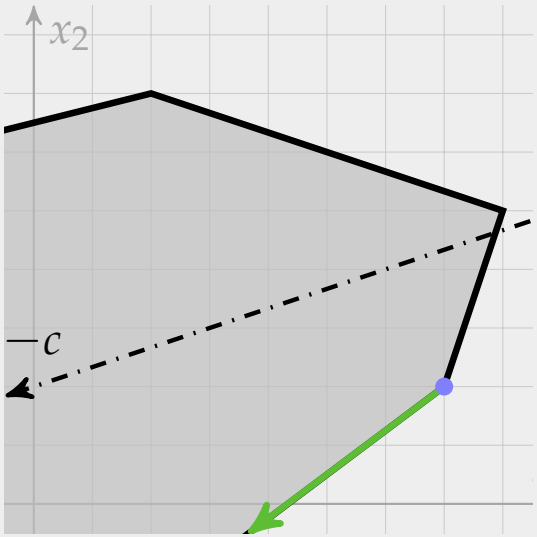
Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```

we can build a direction vector which

- follows an incident edge
- decreases the objective function



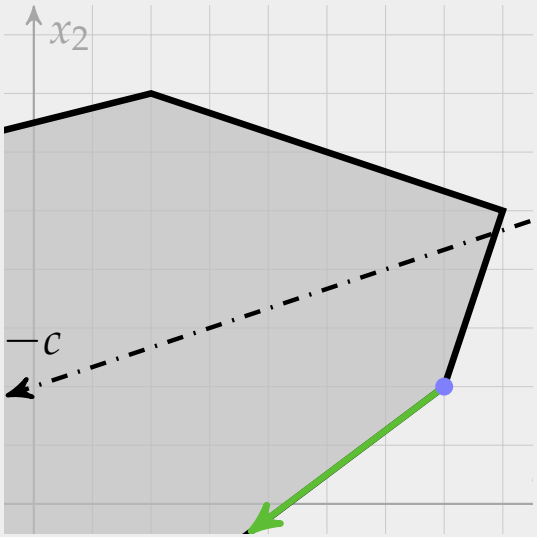
At this stage, two possibilities:

- the direction is **feasible**: `(A *m direction) >=m 0`
  i.e., the halfline is contained in the polyhedron
  $\Longrightarrow$ the LP is unbounded

Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```



we can build a direction vector which

- follows an incident edge
- decreases the objective function

At this stage, two possibilities:

- the direction is **feasible**: `(A *m direction) >=m 0`
  i.e., the halfline is contained in the polyhedron
  $\Longrightarrow$ the LP is unbounded

```
Lemma unbounded_certificate : (A *m direction) >=m 0 ->
  forall M, exists x, (x \in polyhedron A b) /\ ('[c,x] < M)
```

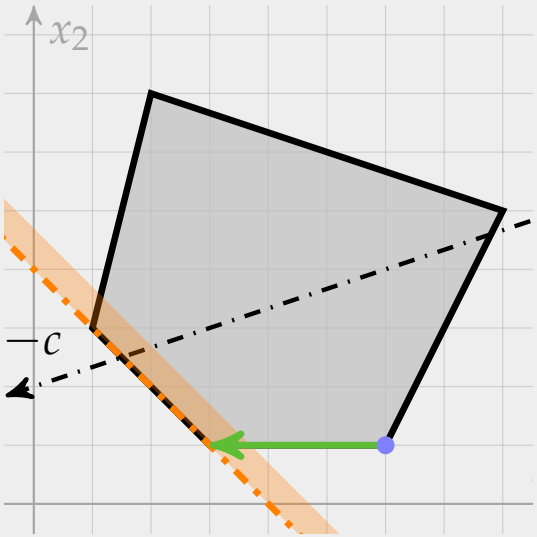
Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```

we can build a direction vector which

- follows an incident edge
- decreases the objective function



At this stage, two possibilities:

- the direction is **feasible**: `(A *m direction) >=m 0`
  $\Longrightarrow$ the LP is unbounded
- or, the halfline hits the boundary of a new halfspace

```
Definition new_halfspace := [...]
```

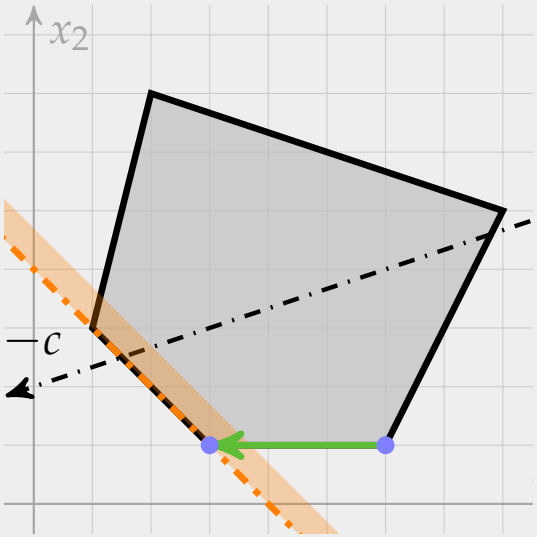
Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```

we can build a direction vector which

- follows an incident edge
- decreases the objective function



At this stage, two possibilities:

- the direction is **feasible**: `(A *m direction) >=m 0`
  $\Longrightarrow$ the LP is unbounded
- or, the halfline hits the boundary of a new halfspace

```
Definition new_halfspace := [...]
```

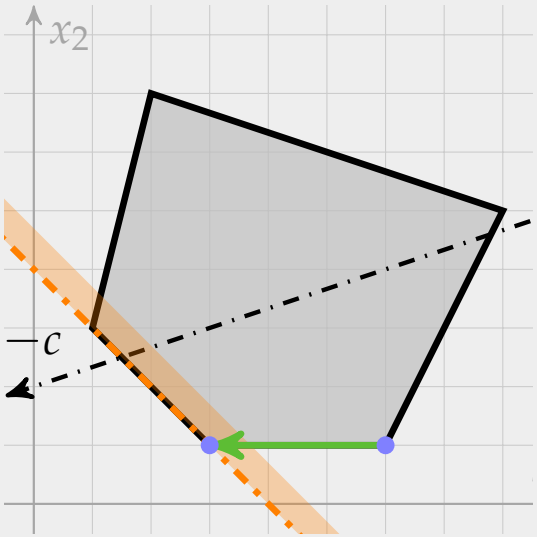
$\Longrightarrow$ the index `new_halfspace` is used to build the next basis:

```
Definition next_I :=
  new_halfspace |: (I :\ (enum_val [...] i)).
```

Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```



we can build a direction vector which

- follows an incident edge
- decreases the objective function

At this stage, two possibilities:

- the direction is **feasible**: `(A *m direction) >=m 0`
  $\Longrightarrow$ the LP is unbounded
- or, the halfline hits the boundary of a new halfspace

```
Definition new_halfspace := [...]
```

$\Longrightarrow$ the index `new_halfspace` is used to build the next basis:

```
Definition next_I :=
  new_halfspace |: (I :\ (enum_val [...] i)).
```

$=$ removes `i` from `I`

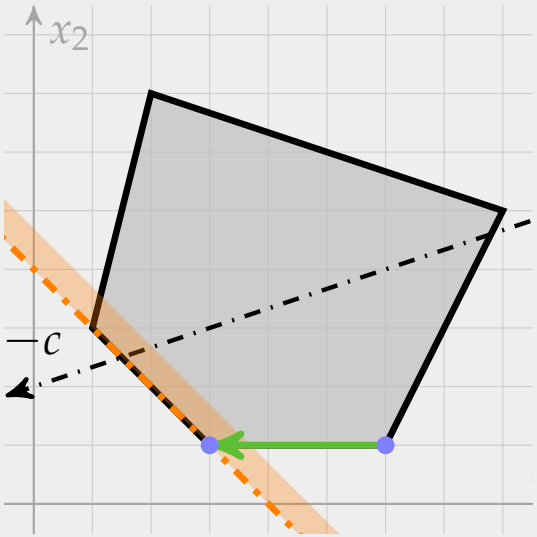
Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```

we can build a direction vector which

- follows an incident edge
- decreases the objective function



At this stage, two possibilities:

- the direction is **feasible**: `(A *m direction) >=m 0`
  $\Longrightarrow$ the LP is unbounded
- or, the halfline hits the boundary of a new halfspace

```
Definition new_halfspace := [...]
```

$\Longrightarrow$ the index `new_halfspace` is used to build the next basis:

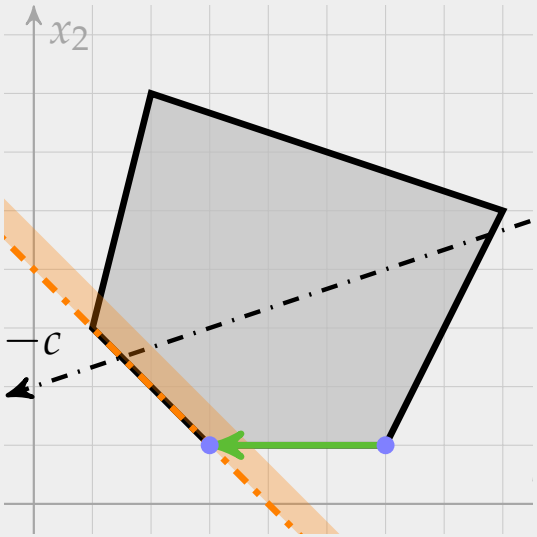
```
Definition next_I :=
  new_halfspace |: (I :\ (enum_val [...] i)).
```

= adds `new_halfspace`

Pivoting

If the reduced cost vector has some negative entries:

```
Variable i : 'I_#|I|.
Hypothesis [...] : reduced_cost i 0 < 0.
```



we can build a direction vector which

- follows an incident edge
- decreases the objective function

At this stage, two possibilities:

- the direction is **feasible**: `(A *m direction) >=m 0`
  $\Longrightarrow$ the LP is unbounded
- or, the halfline hits the boundary of a new halfspace

```
Definition new_halfspace := [...]
```

$\Longrightarrow$ the index `new_halfspace` is used to build the next basis:

```
Definition next_I :=
  new_halfspace |: (I :\ (enum_val [...] i)).

Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

Dealing with termination

The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

Dealing with termination

The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] < '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

Dealing with termination

The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] < '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.
Function simplex_phase2 I {measure basis_height I} := [...].
```

Dealing with termination

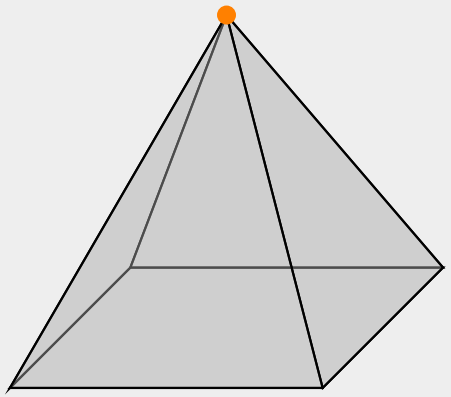
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dealing with termination

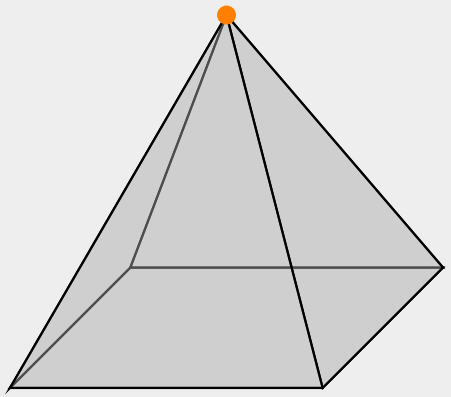
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dealing with termination

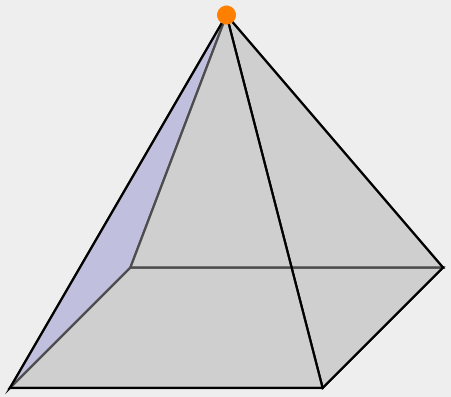
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dealing with termination

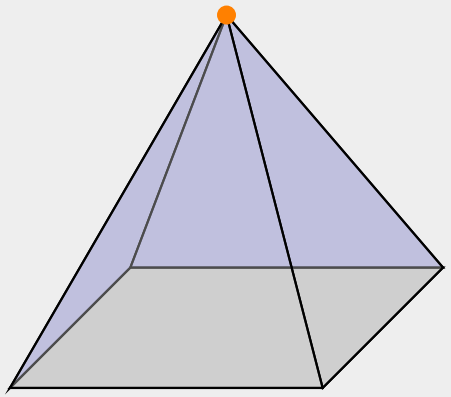
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dealing with termination

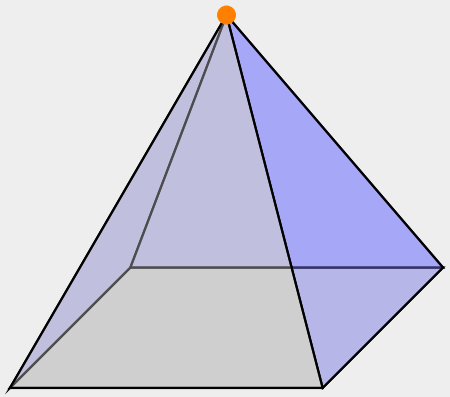
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dealing with termination

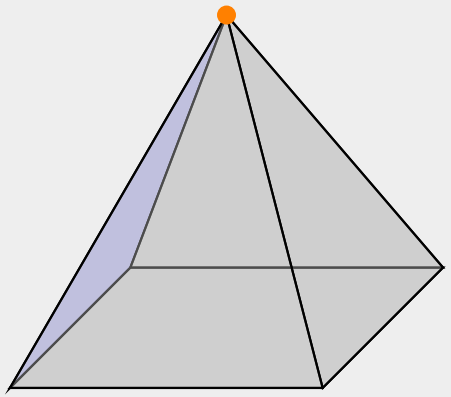
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dealing with termination

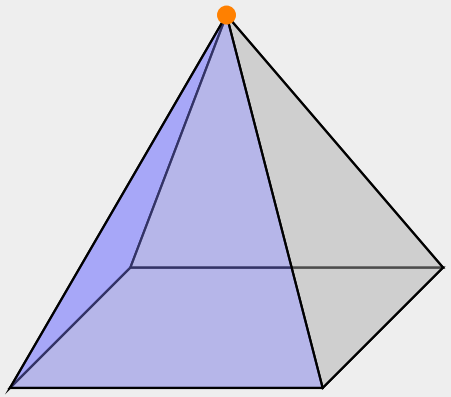
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dealing with termination

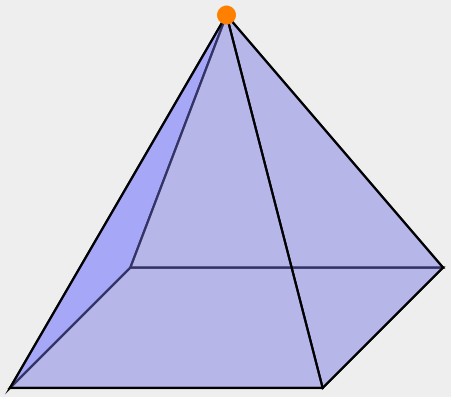
The simplex methods iterates over the basic points with decreasing value:

```
Fact [...]:
  '[c, point_of_basis next_I] <= '[c, point_of_basis I].
```

If at every step the inequality is **strict**, termination follows from the fact that there are finitely many bases:

```
Definition basis_height I :=
  #|[ set J: feasible_bases |
     '[c, point_of_basis I] > '[c, point_of_basis J] ]|.

Function simplex_phase2 I {measure basis_height I} := [...].
```

The inequality may not be strict because of **degenerate bases**
= several bases correspond to the same basic point.



Dantzig's lexicographic rule

Degenerate bases disappear if we consider a slightly **perturbed** LP:

$$\text{minimize} \quad \langle c, x \rangle \quad \text{subject to} \quad Ax \geq \tilde{b}\,, \quad \text{with} \quad \tilde{b}_i := b_i - \varepsilon^i$$

where $0 < \varepsilon \ll 1$.

Dantzig's lexicographic rule

Degenerate bases disappear if we consider a slightly **perturbed** LP:

$$\text{minimize} \quad \langle c, x \rangle \quad \text{subject to} \quad Ax \geq \tilde{b}\,, \quad \text{with} \quad \tilde{b}_i := b_i - \varepsilon^i$$

where $0 < \varepsilon \ll 1$.

Symbolic perturbation scheme

- every real v is now a polynomial in ε of degree $\leq m$

  $$v + v_1\varepsilon + v_2\varepsilon^2 + \cdots + v_m\varepsilon^m$$

  encoded as a row vector $(v, v_1, \ldots, v_m)$.

Dantzig's lexicographic rule

Degenerate bases disappear if we consider a slightly **perturbed** LP:

$$\text{minimize} \quad \langle c, x \rangle \quad \text{subject to} \quad Ax \geq \tilde{b}\,, \quad \text{with} \quad \tilde{b}_i := b_i - \varepsilon^i$$

where $0 < \varepsilon \ll 1$.

Symbolic perturbation scheme

- every real v is now a polynomial in ε of degree $\leq m$

  $$v + v_1\varepsilon + v_2\varepsilon^2 + \cdots + v_m\varepsilon^m$$

  encoded as a row vector $(v, v_1, \ldots, v_m)$.
- the usual order is replaced by the lexicographic order.

$$(v, v_1, \ldots, v_m) \leq_{\text{lex}} (w, w_1, \ldots, w_m)$$
$$\iff v + v_1\varepsilon + v_2\varepsilon^2 + \cdots + v_m\varepsilon^m \leq w + w_1\varepsilon + w_2\varepsilon^2 + \cdots + w_m\varepsilon^m$$
$$\text{for all } 0 < \varepsilon \ll 1\,.$$

Dantzig's lexicographic rule

Degenerate bases disappear if we consider a slightly **perturbed** LP:

$$\text{minimize} \quad \langle c, x \rangle \quad \text{subject to} \quad Ax \geq \tilde{b}, \quad \text{with} \quad \tilde{b}_i := b_i - \varepsilon^i$$

where $0 < \varepsilon \ll 1$.

Symbolic perturbation scheme

- every real v is now a polynomial in ε of degree $\leq m$

  $$v + v_1\varepsilon + v_2\varepsilon^2 + \cdots + v_m\varepsilon^m$$

  encoded as a row vector $(v, v_1, \ldots, v_m)$.
- the usual order is replaced by the lexicographic order.
- still a $\mathbb{R}$-vector space.

Dantzig's lexicographic rule

Degenerate bases disappear if we consider a slightly **perturbed** LP:

$$\text{minimize} \quad \langle c, x \rangle \quad \text{subject to} \quad Ax \geq \tilde{b}\,, \quad \text{with} \quad \tilde{b}_i := b_i - \varepsilon^i$$

where $0 < \varepsilon \ll 1$.

Symbolic perturbation scheme

- every real v is now a polynomial in ε of degree $\leq m$

  $$v + v_1\varepsilon + v_2\varepsilon^2 + \cdots + v_m\varepsilon^m$$

  encoded as a row vector $(v, v_1, \ldots, v_m)$.
- the usual order is replaced by the lexicographic order.
- still a $\mathbb{R}$-vector space.

Observation

The perturbed polyhedron $\{x \in \mathbb{R}^n \mid Ax \geq \tilde{b}\}$ can be encoded by the set

$$\left\{x \in \mathbb{R}^{n\times(1+m)} \mid Ax \geq_{\text{lex}} \begin{pmatrix} b & -I_n \end{pmatrix}\right\}$$

Dantzig's lexicographic rule (2)

Remark: A and c are not perturbed

$\Longrightarrow$ bases and reduced cost vectors are left unchanged.

Dantzig's lexicographic rule (2)

Remark: A and c are not perturbed

$\Longrightarrow$ bases and reduced cost vectors are left unchanged.

Definition

A basis I is said to be **lex-feasible** if the perturbed basic point

$$\tilde{x}^I := (A_I)^{-1}\tilde{b}_I$$

belongs to the perturbed polyhedron.

Dantzig's lexicographic rule (2)

Remark: A and c are not perturbed

$\Longrightarrow$ bases and reduced cost vectors are left unchanged.

Definition

A basis I is said to be **lex-feasible** if the perturbed basic point

$$\tilde{x}^I := (A_I)^{-1} \begin{pmatrix} b_I & (-\delta_{ij})_{(i,j)\in I\times[m]} \end{pmatrix}$$

satisfies $A\tilde{x}^I \geq_{\text{lex}} \begin{pmatrix} b & -I_n \end{pmatrix}$.

Dantzig's lexicographic rule (2)

Remark: A and c are not perturbed

$\Longrightarrow$ bases and reduced cost vectors are left unchanged.

Definition

A basis I is said to be **lex-feasible** if the perturbed basic point

$$\tilde{x}^I := (A_I)^{-1}\begin{pmatrix} b_I & (-\delta_{ij})_{(i,j)\in I\times[m]} \end{pmatrix}$$

satisfies $A\tilde{x}^I \geq_{\text{lex}} \begin{pmatrix} b & -I_n \end{pmatrix}$.

Proposition

Lex-feasible bases form a subset of feasible bases.

Proof sketch

The point $\tilde{x}^I$ is a perturbation of the basic point x^I:

```
Lemma [...] : point_of_basis I = col 0 (point_of_basis_pert I).
+ forall u v, u <=lex v -> u 0 0 <= v 0 0.
Qed.
```

Dantzig's lexicographic rule (3)

```
Lemma no_degenerate_basis (I I' : basis) :
  point_of_basis_pert I = point_of_basis_pert I' -> I = I'.
```

Dantzig's lexicographic rule (3)

```
Lemma no_degenerate_basis (I I' : basis) :
  point_of_basis_pert I = point_of_basis_pert I' -> I = I'.
```

Proof sketch

```
Fact [...] : col (1+j) (point_of_basis_pert I) != 0 <-> (j \in I).
```

Dantzig's lexicographic rule (3)

```
Lemma no_degenerate_basis (I I' : basis) :
  point_of_basis_pert I = point_of_basis_pert I' -> I = I'.
```

Outcome

We arrive at a function `phase2 : lex_feasible_basis -> result`

- iterate over the lex-feasible bases;
- **terminates** in a finite number of steps;
- either finds an optimal lex-feasible basis,
  or determines that the LP is unbounded.

Dantzig's lexicographic rule (3)

```
Lemma no_degenerate_basis (I I' : basis) :
  point_of_basis_pert I = point_of_basis_pert I' -> I = I'.
```

Outcome

We arrive at a function `phase2 : lex_feasible_basis -> result`

- iterate over the lex-feasible bases;
- **terminates** in a finite number of steps;
- either finds an optimal lex-feasible basis,
  or determines that the LP is unbounded.

```
Inductive result :=
| Optimal_basis of lex_feasible_basis.
| Unbounded_cert (I: lex_feasible_basis) of 'I_#|I|
```

```
Lemma phase2P : forall I_0, phase2_spec (phase2 I_0).
```

where

```
Inductive phase2_spec : result -> Type :=
| Optimal (I: lex_feasible_basis) of
  (reduced_cost I) >=m 0 : phase2_spec (Optimal_basis I).
| Unbounded (I: lex_feasible_basis) (i: 'I_#|I|) of
  (reduced_cost I) i 0 < 0 /\ (A *m direction I i) >=m 0 :
    phase2_spec (Unbounded_cert i).
```

Phase I simplex method : finding an initial feasible basis

Usually handled by solving an auxiliary LP:

- with a trivial feasible basis;
- on which we call Phase II;
- which determines whether the LP is feasible or not.

Phase I Linear Program

$$\begin{aligned} \text{minimize} \quad & \langle e, y - A_K x \rangle \\ \text{subject to} \quad & A_K x \leq b_K + y\,,\ A_L x \geq b_L \\ & y \geq 0\,,\ (x, y) \in \mathbb{R}^{n+p} \end{aligned}$$

where K and L are built from an (arbitrary) basis I:

$$K := \{i \in [m] \colon A_i x^I < b_i\}\,, \qquad L := \{i \in [m] \colon A_i x^I \geq b_i\}\,.$$

Phase I simplex method : finding an initial feasible basis

Usually handled by solving an auxiliary LP:

- with a trivial feasible basis;
- on which we call Phase II;
- which determines whether the LP is feasible or not.

Phase I Linear Program

$$\begin{aligned} \text{minimize} \quad & \langle e, y - A_K x \rangle \\ \text{subject to} \quad & A_K x \leq b_K + y\,,\ A_L x \geq b_L \\ & y \geq 0\,,\ (x, y) \in \mathbb{R}^{n+p} \end{aligned}$$

where K and L are built from an (arbitrary) basis I:

$$K := \{i \in [m] : A_i x^I < b_i\}\,, \qquad L := \{i \in [m] : A_i x^I \geq b_i\}\,.$$

Problem

Lots of **painful** and **boring** manipulations of block matrices:

block structure reflected on bases and submatrices.

Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

Find a (lex-)feasible basis from the feasible point `x`!

Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

Build a (lex-)feasible basis from the feasible point `x`!

Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

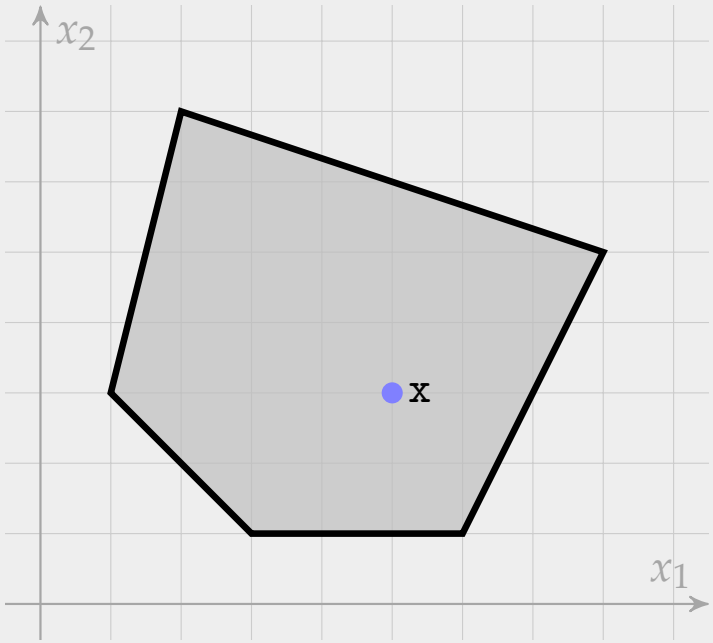
Idea

Build a (lex-)feasible basis from the feasible point `x`!

A technique attributed to Schrijver

Let $I(x) := \{i \in [m] \mid A_i x = b_i\}$.

While $\mathsf{rank}\, A_{I(x)} < n$, do



Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

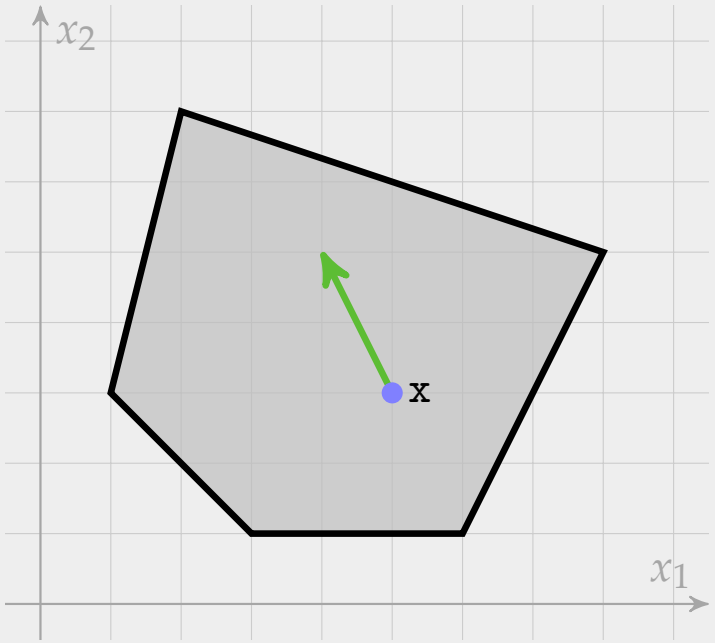
Build a (lex-)feasible basis from the feasible point `x`!

A technique attributed to Schrijver

Let $I(x) := \{i \in [m] \mid A_i x = b_i\}$.

While rank $A_{I(x)} < n$, do

- take a direction d



Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

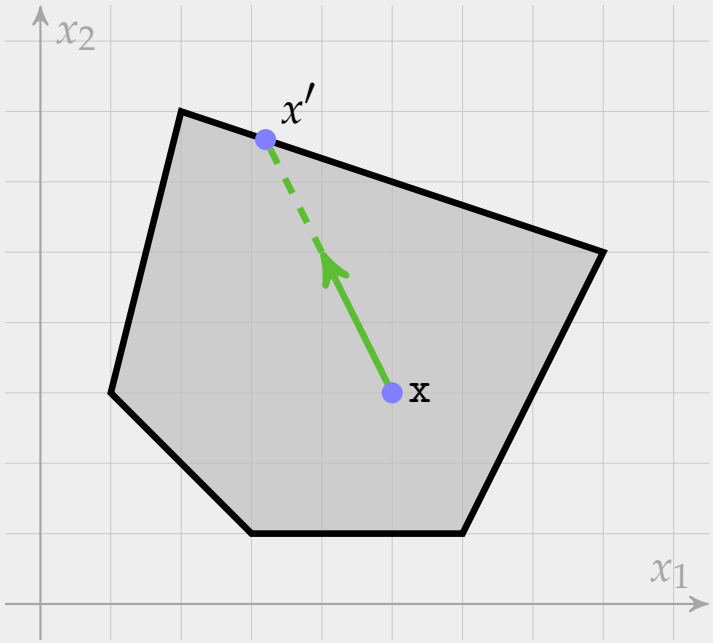
Build a (lex-)feasible basis from the feasible point `x`!

A technique attributed to Schrijver

Let $I(x) := \{i \in [m] \mid A_i x = b_i\}$.

While $\mathsf{rank}\, A_{I(x)} < n$, do

- take a direction d
- move along d as much as possible
  $\Longrightarrow$ new point x'



Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

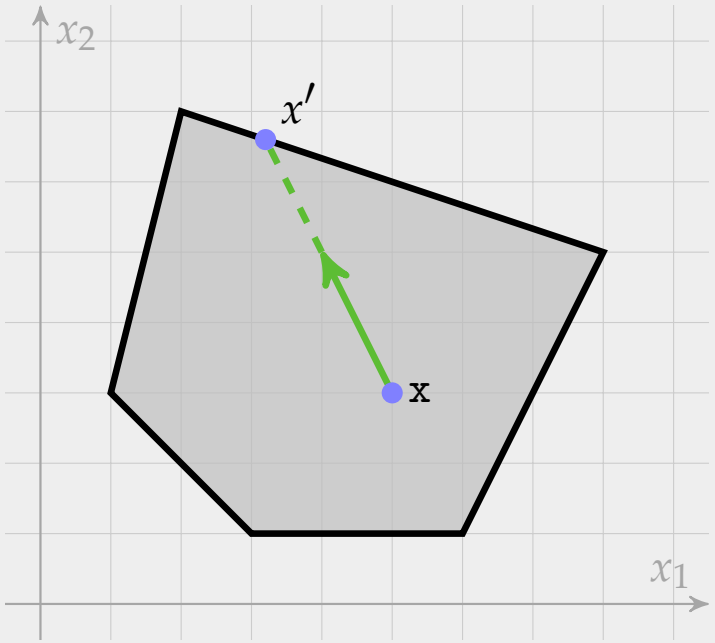
Build a (lex-)feasible basis from the feasible point `x`!

A technique attributed to Schrijver

Let $I(x) := \{i \in [m] \mid A_i x = b_i\}$.

While $\mathsf{rank}\, A_{I(x)} < n$, do

- take a direction d
- move along d as much as possible
  $\Longrightarrow$ new point x' verifies:
  $$\mathsf{rank}\, A_{I(x')} > \mathsf{rank}\, A_{I(x)}\,.$$
- $x := x'$.



Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

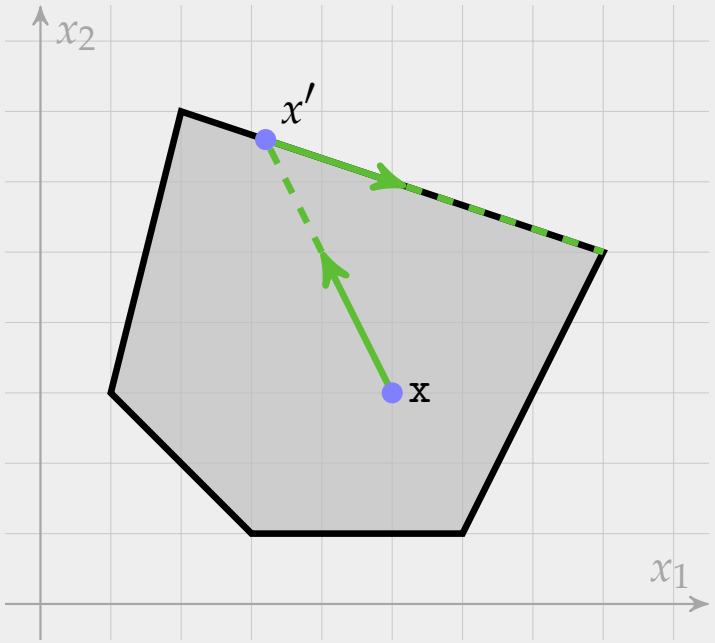
Build a (lex-)feasible basis from the feasible point `x`!

A technique attributed to Schrijver

Let $I(x) := \{i \in [m] \mid A_i x = b_i\}$.

While $\mathsf{rank}\, A_{I(x)} < n$, do

- take a direction $d \in \mathsf{Ker} A_{I(x)}$
- move along d as much as possible
  $\Longrightarrow$ new point x' verifies:
  $$\mathsf{rank}\, A_{I(x')} > \mathsf{rank}\, A_{I(x)}\ .$$
- $x := x'$.



Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

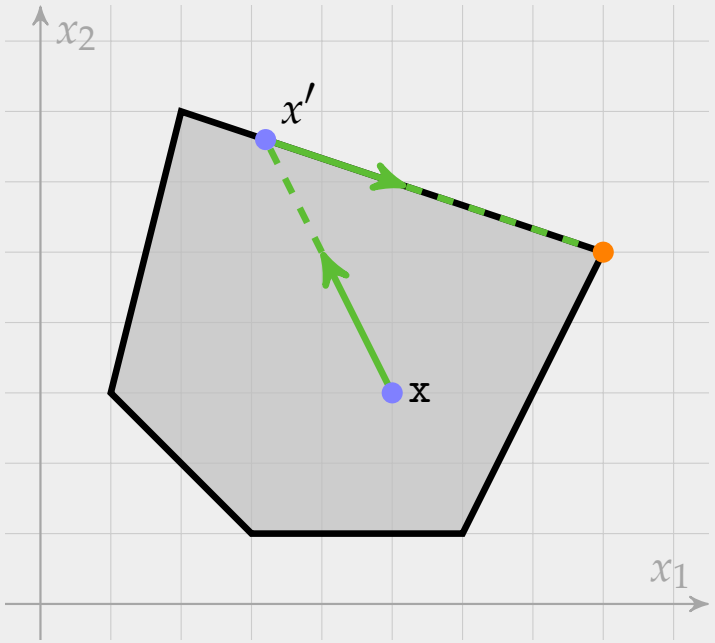
Build a (lex-)feasible basis from the feasible point `x`!

A technique attributed to Schrijver

Let $I(x) := \{i \in [m] \mid A_i x = b_i\}$.

While $\mathsf{rank}\, A_{I(x)} < n$, do

- take a direction $d \in \mathsf{Ker} A_{I(x)}$
- move along d as much as possible
  $\Longrightarrow$ new point x' verifies:
  $$\mathsf{rank}\, A_{I(x')} > \mathsf{rank}\, A_{I(x)}\ .$$
- $x := x'$.



Alternative to Phase I

Our definition of emptiness predicate provides a witness point:

```
Theorem emptyPn A b : reflect (exists x, A *m x >=m b) (~~ empty A b).
```

Idea

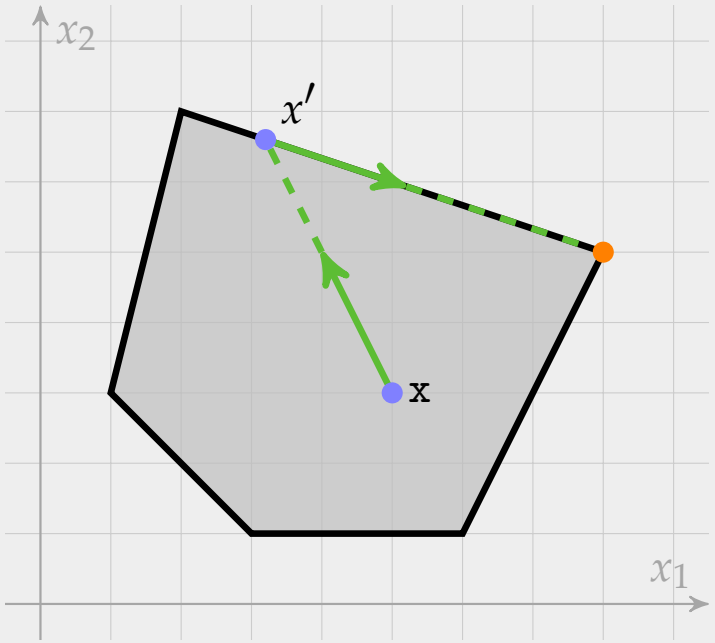
Build a (lex-)feasible basis from the feasible point `x`!

A technique attributed to Schrijver

Let $I(x) := \{i \in [m] \mid A_i x = b_i\}$.

While $\mathsf{rank}\, A_{I(x)} < n$, do

- take a direction $d \in \mathsf{Ker} A_{I(x)}$
- move along d as much as possible
  $\Longrightarrow$ new point x' verifies:
  $$\mathsf{rank}\, A_{I(x')} > \mathsf{rank}\, A_{I(x)}\ .$$
- $x := x'$.



Benefits

- relies on the same ingredients as pivoting
- applies to the symbolic perturbation setting of lexicographic rule

Outline of the talk

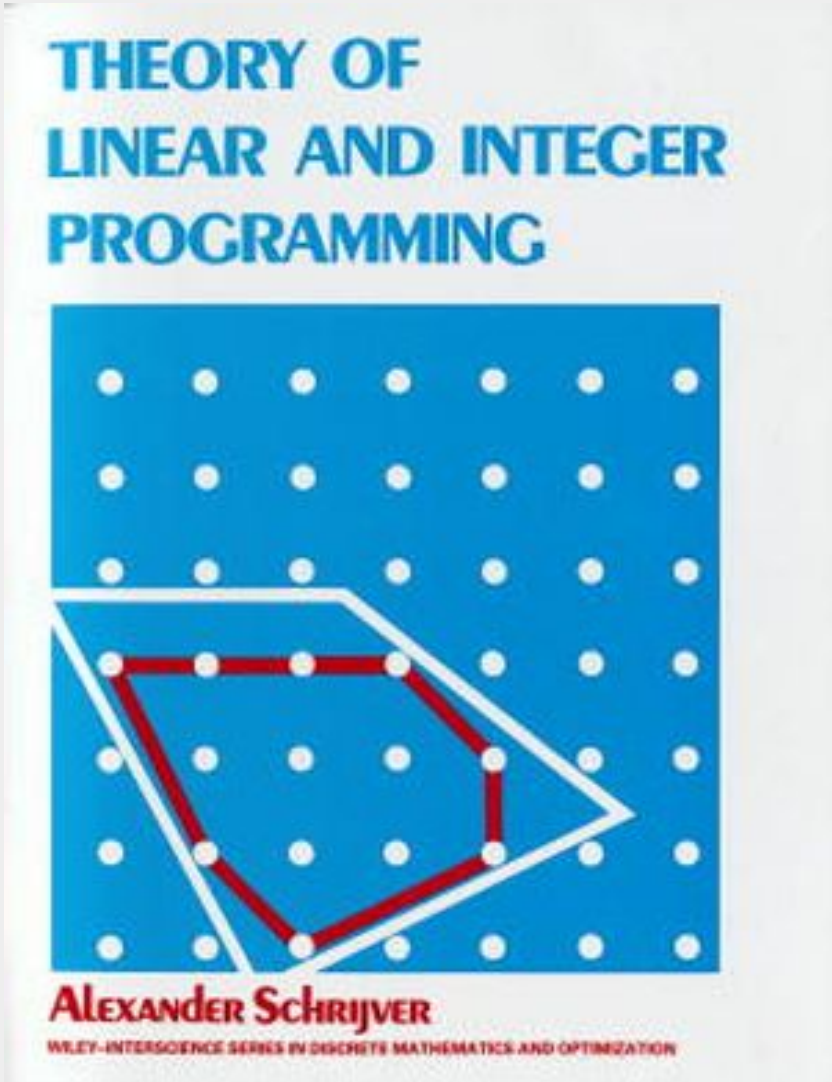
Concluding remarks

Contribution

Foundations of the formalization of convex polyhedra in Coq

- extensive use of Boolean reflection methodology
  $\Longrightarrow$ **constructive** formalization
  $+$ benefit from the Mathematical Components library
- formalization of the simplex method
- lexicographic rule yields a rather simple termination proof
  $+$ opens perspectives to enumerate the vertices (basic points)
- pragmatic way to deal with Phase I
- many essential results on polyhedra follows from the simplex method (with a constructive proof!)

Concluding remarks (2)



7

Fundamental concepts and results on polyhedra, linear inequalities, and linear programming

In this chapter we first state a fundamental theorem on linear inequalities (Section 7.1), and next we derive as consequences some other important results, like the Finite basis theorem for cones and polytopes, the Decomposition theorem for polyhedra (Section 7.2), Farkas' lemma (Section 7.3), the Duality theorem of linear programming (Section 7.4), an affine form of Farkas' lemma (Section 7.6), Carathéodory's theorem (Section 7.7), and results for strict inequalities (Section 7.8). In Section 7.5 we give a geometrical interpretation of LP-duality. In Section 7.9 we study the phenomenon of *complementary slackness*.

Each of the results in this chapter holds both in real spaces and in rational spaces. In the latter case, all numbers occurring (like matrix and vector entries, variables) are restricted to the rationals.

7.1. THE FUNDAMENTAL THEOREM OF LINEAR INEQUALITIES

The fundamental theorem is due to Farkas [1894, 1898a] and Minkowski [1896], with sharpenings by Carathéodory [1911] and Weyl [1935]. Its geometric content is easily understood in three dimensions.

Theorem 7.1 (Fundamental theorem of linear inequalities). *Let $a_1, \ldots, a_m$, b be vectors in n-dimensional space. Then:*

either 1. *b is a nonnegative linear combination of linearly independent vectors from $a_1, \ldots, a_m$;*

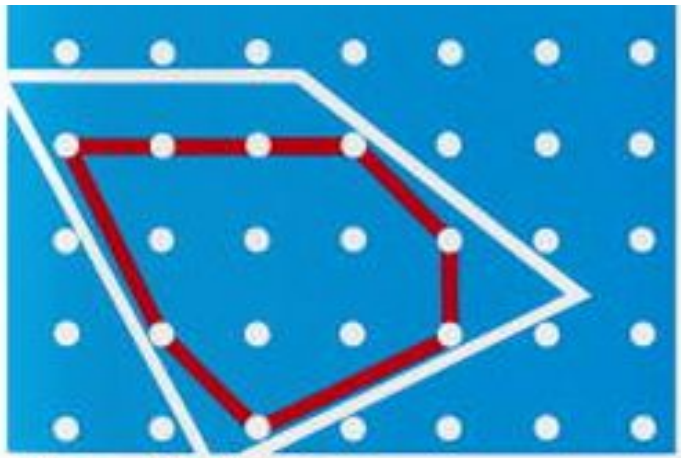
85

Concluding remarks (2)

THEORY OF

7

In this chapter we first state a fundamental theorem on linear inequalities (Section 7.1), and next we derive as consequences some other important results, like the Finite basis theorem for cones and polytopes, the Decomposition theorem for polyhedra (Section 7.2), Farkas' lemma (Section 7.3), the Duality theorem of linear programming (Section 7.4), an affine form of Farkas' lemma (Section 7.6), Carathéodory's theorem (Section 7.7), and results for strict inequalities (Section 7.8). In Section 7.5 we give a geometrical interpretation of LP-duality. In Section 7.9 we study the phenomenon of *complementary slackness*.



ALEXANDER SCHRIJVER

WILEY-INTERSCIENCE SERIES IN DISCRETE MATHEMATICS AND OPTIMIZATION

In this chapter we first state a fundamental theorem on linear inequalities (Section 7.1), and next we derive as consequences some other important results, like the Finite basis theorem for cones and polytopes, the Decomposition theorem for polyhedra (Section 7.2), Farkas' lemma (Section 7.3), the Duality theorem of linear programming (Section 7.4), an affine form of Farkas' lemma (Section 7.6), Carathéodory's theorem (Section 7.7), and results for strict inequalities (Section 7.8). In Section 7.5 we give a geometrical interpretation of LP-duality. In Section 7.9 we study the phenomenon of *complementary slackness*.

Each of the results in this chapter holds both in real spaces and in rational spaces. In the latter case, all numbers occurring (like matrix and vector entries, variables) are restricted to the rationals.

7.1. THE FUNDAMENTAL THEOREM OF LINEAR INEQUALITIES

The fundamental theorem is due to Farkas [1894, 1898a] and Minkowski [1896], with sharpenings by Carathéodory [1911] and Weyl [1935]. Its geometric content is easily understood in three dimensions.

Theorem 7.1 (Fundamental theorem of linear inequalities). *Let $a_1, \ldots, a_m$, b be vectors in n-dimensional space. Then:*

either 1. *b is a nonnegative linear combination of linearly independent vectors from $a_1, \ldots, a_m$;*

85

Concluding remarks (2)

THEORY OF

7

In this chapter we first state a fundamental theorem on linear inequalities (Section 7.1), and next we derive as consequences some other important results, like the Finite basis theorem for cones and polytopes, the Decomposition theorem for polyhedra (Section 7.2), Farkas' lemma (Section 7.3), the Duality theorem of linear programming (Section 7.4), an affine form of Farkas' lemma (Section 7.6), Carathéodory's theorem (Section 7.7), and results for strict inequalities (Section 7.8). In Section 7.5 we give a geometrical interpretation of LP-duality. In Section 7.9 we study the phenomenon of *complementary slackness*.

In this chapter we first state a fundamental theorem on linear inequalities (Section 7.1), and next we derive as consequences some other important results, like the Finite basis theorem

The above proof of this fundamental theorem also gives a fundamental algorithm: it is a disguised form of the famous *simplex method*, with *Bland's rule* incorporated- -see Chapter 11 (see Debreu [1964] for a similar proof, but with a lexicographic rule).

ALEXANDER SCHRIJVER

WILEY-INTERSCIENCE SERIES IN DISCRETE MATHEMATICS AND OPTIMIZATION

Theorem 7.1 (Fundamental theorem of linear inequalities). *Let $a_1, \ldots, a_m$, b be vectors in n-dimensional space. Then:*

either 1. *b is a nonnegative linear combination of linearly independent vectors from $a_1, \ldots, a_m$;*

85

Concluding remarks (3)

Stats

```
> coqwc theories/*.v
     spec    proof comments
      128      239        1 theories/extra_matrix.v
       57      105        0 theories/extra_misc.v
       57       77       10 theories/inner_product.v
      251      469        2 theories/vector_order.v
      173      440        0 theories/row_submx.v
      667      994       57 theories/simplex.v
       25       38        0 theories/duality.v
       36      106        5 theories/minkowski.v
```

Concluding remarks (3)

Stats

```
> coqwc theories/*.v
     spec    proof comments
      128      239        1 theories/extra_matrix.v
       57      105        0 theories/extra_misc.v
       57       77       10 theories/inner_product.v
      251      469        2 theories/vector_order.v
      173      440        0 theories/row_submx.v
      667      994       57 theories/simplex.v
       25       38        0 theories/duality.v
       36      106        5 theories/minkowski.v
```

Lessons learned

- steep learning curve

Concluding remarks (3)

Stats

```
> coqwc theories/*.v
     spec    proof comments
      128      239        1 theories/extra_matrix.v
       57      105        0 theories/extra_misc.v
       57       77       10 theories/inner_product.v
      251      469        2 theories/vector_order.v
      173      440        0 theories/row_submx.v
      667      994       57 theories/simplex.v
       25       38        0 theories/duality.v
       36      106        5 theories/minkowski.v
```

Lessons learned

- steep learning curve
- find the right representation

Concluding remarks (3)

Stats

```
> coqwc theories/*.v
     spec    proof comments
      128      239        1 theories/extra_matrix.v
       57      105        0 theories/extra_misc.v
       57       77       10 theories/inner_product.v
      251      469        2 theories/vector_order.v
      173      440        0 theories/row_submx.v
      667      994       57 theories/simplex.v
       25       38        0 theories/duality.v
       36      106        5 theories/minkowski.v
```

Lessons learned

- steep learning curve
- find the right representation
- write “good” proof: maintenance, factorization

Concluding remarks (3)

Stats

```
> coqwc theories/*.v
     spec    proof comments
      128      239        1 theories/extra_matrix.v
       57      105        0 theories/extra_misc.v
       57       77       10 theories/inner_product.v
      251      469        2 theories/vector_order.v
      173      440        0 theories/row_submx.v
      667      994       57 theories/simplex.v
       25       38        0 theories/duality.v
       36      106        5 theories/minkowski.v
```

Lessons learned

- steep learning curve
- find the right representation
- write “good” proof: maintenance, factorization
- scaling up to large projects requires a lot of rewriting

Concluding remarks (3)

Stats

```
> coqwc theories/*.v
     spec    proof comments
      128      239        1 theories/extra_matrix.v
       57      105        0 theories/extra_misc.v
       57       77       10 theories/inner_product.v
      251      469        2 theories/vector_order.v
      173      440        0 theories/row_submx.v
      667      994       57 theories/simplex.v
       25       38        0 theories/duality.v
       36      106        5 theories/minkowski.v
```

Lessons learned

- steep learning curve
- find the right representation
- write “good” proof: maintenance, factorization
- scaling up to large projects requires a lot of rewriting
- a lot of fun

Perspectives

Right now

Define a **type** for polyhedra: quotient external representations w.r.t. set equality

- equivalence is decidable
- but no canonical representation!

+ introduce the notion of faces, intrinsically related with the representation issue.

Perspectives

Right now

Define a **type** for polyhedra: quotient external representations w.r.t. set equality

- equivalence is decidable
- but no canonical representation!

+ introduce the notion of faces, intrinsically related with the representation issue.

Really soon: the computational side of this work

- MathComp data structures are not made for computing
- Coq is equipped with very efficient data structures
- transfer the statements on proof oriented types to computation oriented types.

Perspectives

Right now

Define a **type** for polyhedra: quotient external representations w.r.t. set equality

- equivalence is decidable
- but no canonical representation!

+ introduce the notion of faces, intrinsically related with the representation issue.

Really soon: the computational side of this work

- MathComp data structures are not made for computing
- Coq is equipped with very efficient data structures
- transfer the statements on proof oriented types to computation oriented types.

Next

- many core properties to be formalized
- apply to **real** computational math problems
- interface with (informal) math software

Perspectives

Right now

Define a **type** for polyhedra: quotient external representations w.r.t. set equality

- equivalence is decidable
- but no canonical representation!

+ introduce the notion of faces, intrinsically related with the representation issue.

Really soon: the computational side of this work

- MathComp data structures are not made for computing
- Coq is equipped with very efficient data structures
- transfer the statements on proof oriented types to computation oriented types.

Next

- many core properties to be formalized
- apply to **real** computational math problems
- interface with (informal) math software

Master internship / PhD: contact me if you're interested!

Thank you!

 `github.com/nhojem/Coq-Polyhedra`

arXiv:1706.10269

Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the odd order theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, **Interactive Theorem Proving**, pages 163–179, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39634-2.

Georges Gonthier, Assia Mahboubi, and Enrico Tassi. A Small Scale Reflection Extension for the Coq system. Research Report RR-6455, Inria Saclay Ile de France, 2016.

Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, and et al. A formal proof of the Kepler conjecture. **Forum of Mathematics, Pi**, 5:e2, 2017. doi: 10.1017/fmp.2017.1.

L.G. Khachiyan. Polynomial algorithms in linear programming. **USSR Computational Mathematics and Mathematical Physics**, 20(1), 1980. ISSN 0041-5553. doi: http://dx.doi.org/10.1016/0041-5553(80)90061-0.

Benjamin Matschke, Francisco Santos, and Christophe Weibel. The width of five-dimensional prismatoids. **Proceedings of the London Mathematical Society**, 110(3):647–672, 2015. ISSN 1460-244X. doi: 10.1112/plms/pdu064. URL `http://dx.doi.org/10.1112/plms/pdu064`.

Steve Smale. Mathematical problems for the next century. **Mathematical Intelligencer**, 20, 1998.