Efficient and Formally Proved Polyhedral Computations

Quentin Canu

Supervisors: Xavier Allamigeon, Georges Gonthier & Pierre-Yves Strub

21 April 2021

Outline

Outline

Formally Proved Polyhedral Computing

Polyhedra

- Expressivity & Algorithmic Complexity
- Multiple Problems, from Mathematics (Combinatorics Problems) to Computer Science (Program Verification)
- Computational Proofs

On the needs of computational formal Proofs

	x_1	x_2	x_3	x_4	x_5
1^+	18	0	0	0	1
2^+	−18	0	0	0	1
3^+	0	18	0	0	1
4^+	0	−18	0	0	1
5^+	0	0	45	0	1
6^+	0	0	−45	0	1
7^+	0	0	0	45	1
8^+	0	0	0	−45	1
9^+	15	15	0	0	1
10^+	−15	15	0	0	1
11^+	15	−15	0	0	1
12^+	−15	−15	0	0	1
13^+	0	0	30	30	1
14^+	0	0	−30	30	1
15^+	0	0	30	−30	1
16^+	0	0	−30	−30	1
17^+	0	10	40	0	1
18^+	0	−10	40	0	1
19^+	0	10	−40	0	1
20^+	0	−10	−40	0	1
21^+	10	0	0	40	1
22^+	−10	0	0	40	1
23^+	10	0	0	−40	1
24^+	−10	0	0	−40	1

	x_1	x_2	x_3	x_4	x_5
1^-	0	0	0	18	−1
2^-	0	0	0	−18	−1
3^-	0	0	18	0	−1
4^-	0	0	−18	0	−1
5^-	45	0	0	0	−1
6^-	−45	0	0	0	−1
7^-	0	45	0	0	−1
8^-	0	−45	0	0	−1
9^-	0	0	15	15	−1
10^-	0	0	15	−15	−1
11^-	0	0	−15	15	−1
12^-	0	0	−15	−15	−1
13^-	30	30	0	0	−1
14^-	−30	30	0	0	−1
15^-	30	−30	0	0	−1
16^-	−30	−30	0	0	−1
17^-	40	0	10	0	−1
18^-	40	0	−10	0	−1
19^-	−40	0	10	0	−1
20^-	−40	0	−10	0	−1
21^-	0	40	0	10	−1
22^-	0	40	0	−10	−1
23^-	0	−40	0	10	−1
24^-	0	−40	0	−10	−1

Table 1: The 48 vertices of a 5-prismatoid without the d-step property

From Santos' *A counterexample to the Hirsch conjecture*

Coq-Polyhedra

- A Coq Library on polyhedral computing
- Formalizing the Simplex Method, Face Lattice of Polyhedra...
- Makes use of Mathematical Components Library ...

Outline

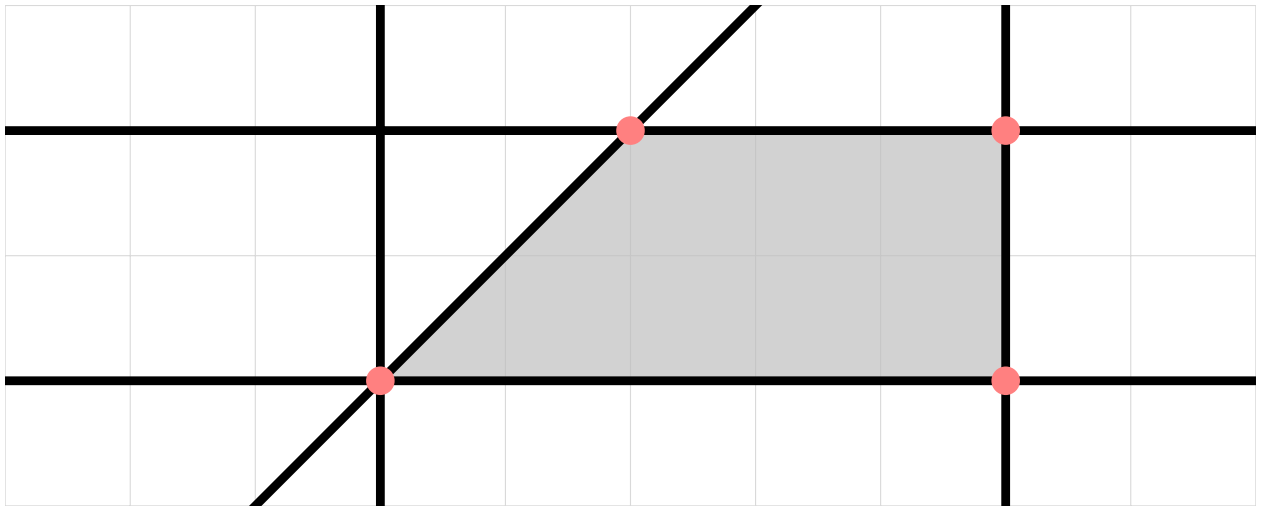
Polyhedra and Polytopes

Definition

A (convex) *polyhedron* of $\mathbb{R}^n$ is the set of the solutions of a finite system of linear inequations. More formally, if $m, n \in \mathbb{N}$, $A \in \mathcal{M}_{m,n}(\mathbb{R})$ and $b \in \mathbb{R}^m$, then a polyhedron has the form $\{x \in \mathbb{R}^n \mid Ax \geq b\}$.

Definition

A bounded polyhedron is called a *polytope*.



Vertices of Polytopes

Let $\mathcal{P}$ a polytope of $\mathbb{R}^n$ defined by the set $\{x \in \mathbb{R}^n | Ax \geq b\}$

Definition

Let I a subset of $\mathbb{N}_m$, and A_I, b_I the matrices extracted from A and b of the lines indexed by I. Then I is called a *basis* iff $\#I = n$ and $rank(A_I) = n$

Proposition

$x \in \mathbb{R}^n$ is a vertex of $\mathcal{P}$ iff

- *x is feasible, i.e. $x \in \mathcal{P}$*
- *there exists a basis I such that $A_I x = b_I$ (I is a feasible basis)*

Vertices of Polytopes

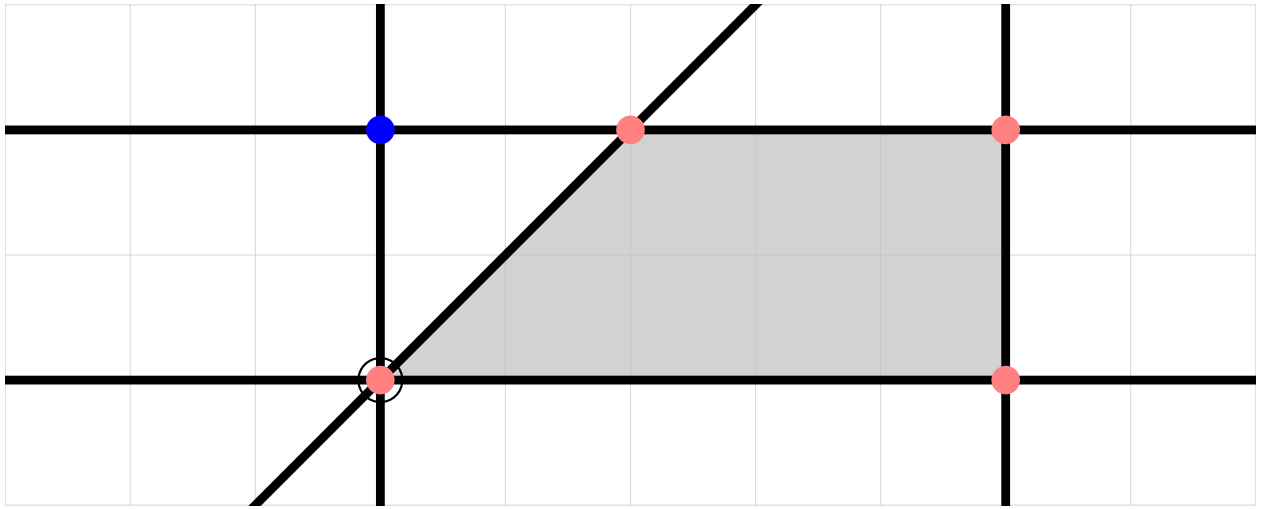


Figure: Red points are vertices, blue point is not

The circled point is associated to three different bases. These bases are called $degenerate$.

Linear Programming

Definition

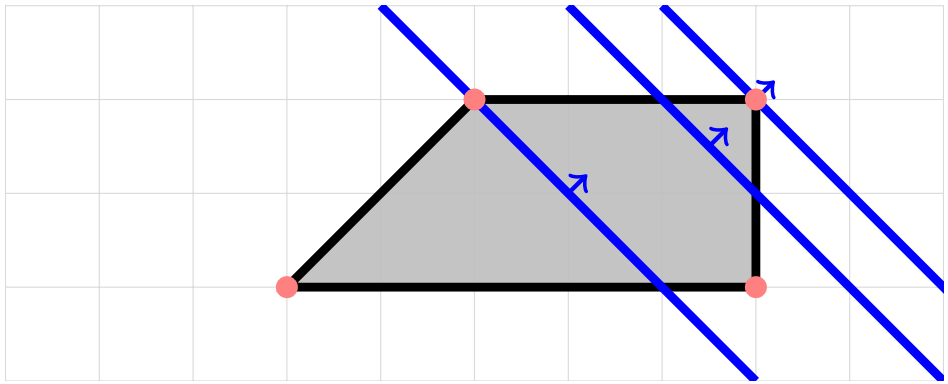
A *Linear Program* is an optimization problem that can be expressed as

$$\min\{c^T \cdot x \mid x \in \mathcal{P}\}$$

where $\mathcal{P}$ is a polyhedron.

Proposition

If there is an optimum, it is reached on a vertex.



The Simplex Algorithm

Let I a feasible basis, and x^I its associated vertex.

Definition

The *reduced cost* is the vector, denoted u^I, such that u^I is the only solution of

$$A^T u = c$$

Proposition

- *If* $u^I \geq 0$, *then* x^I *is an optimal solution of the linear program.*
- *If, for some* $i \in I$, $u^I_i < 0$, *then the linear program decreases on the half-line* $\{x + \lambda d \mid \lambda \in \mathbb{R}^+\}$, *where* d *is the only solution of*

$$A_I d = \mathbb{1}_i$$

The Simplex Algorithm : Pivoting

In the case of $i \in I$ such that $u_i^I < 0$

Proposition

If $Ad \geq 0$, *then* $\forall \lambda \in \mathbb{R}^+, x^I + \lambda d \in \mathcal{P}$

Otherwise, if $(Ad)_j < 0$ for some j, then the half-line is not included in the $\mathcal{P}$, so we can consider $\lambda_m := \sup\{\lambda \mid x^I + \lambda d \in \mathcal{P}\}$

Proposition

$$\lambda_m = \min_{\{j|(Ad)_j<0\}} \frac{b_j - A_j x^I}{(Ad)_j}$$

Let j *reaching this minimum, then* $I' = I \setminus \{i\} \cup \{j\}$ *is a feasible basis, and* $c^T \cdot x^{I'} \leq c^T \cdot x^I$

The Simplex Algorithm

The Simplex Algorithm consists in two phases

- *Phase 1* : Pick an initial feasible basis I_0.
- *Phase 2* : Use the previous procedure to construct feasible bases by pivoting, until we know we have an optimum.

In case of degenerate bases, the simplex algorithm can pivot two bases associated to the same vertex. The choice of the pivot is critical.

Outline

Vertices Validation: The Initial Problem

Given the system $(Ax \geq b)$ of a polytope $\mathcal{P}$ of $\mathbb{R}^n$, and a list L of $\mathbb{R}^n$ points, is L the exact list of the vertices of $\mathcal{P}$?

- Validation $\neq$ Computation
- Informally Provided Input
- A Need of Efficiency
- Formally Proved Algorithm

Sketch of the Algorithm

Let S the set of vertices of $\mathcal{P}$.

"$L \subseteq S$" : for each point $x \in L$

1. x is feasible (it checks $Ax \geq b$)
2. if $I \subseteq \mathbb{N}_m$ such that $A_I x = b_I$, then it verifies $rank(A_I) = n$

"$L = S$" : it verifies that L is closed for the adjacency relation

- for each $x \in L$, if $y \in \mathbb{R}^n$ is adjacent to x, then $y \in L$

Adjacency Relation

Proposition

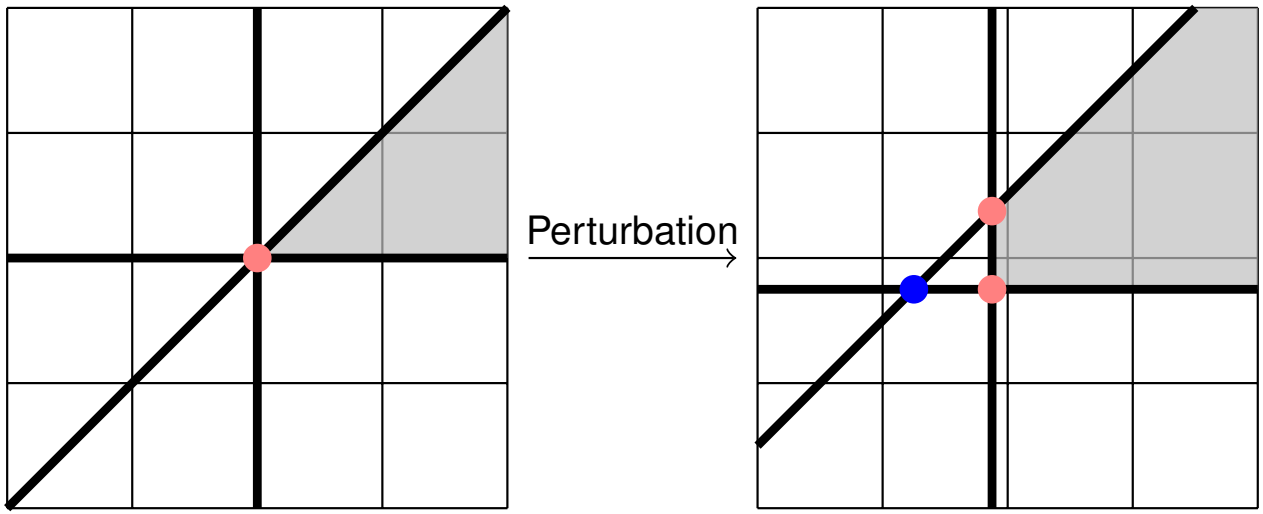
Two distinct vertices x and y are adjacent iff there exists one basis I associated to x and one basis J associated to y such that $\#(I \cap J) = n - 1$

Proposition

If $\mathcal{P}$ has no degenerate bases, then each vertex is adjacent to exactly n vertices.

In that case, to check the closure of the adjacency relation is equivalent to check the regularity of the relation

Perturbed Polyhedron



Perturbed Polyhedron

Definition

Given $\mathcal{P} = \{x \in \mathbb{R}^n | Ax \geq b\}$, we define $\tilde{\mathcal{P}}(\epsilon) = \{x \in \mathbb{R}^n \mid Ax \geq \tilde{b}(\epsilon)\}$ with $\epsilon > 0$ and $\tilde{b}(\epsilon) = (b_i - \epsilon^i)_{1 \leq i \leq m}$.

Theorem

If ϵ is small enough, then $\tilde{\mathcal{P}}(\epsilon)$ has no degenerate bases. Moreover, if $x(\epsilon)$ is a vertex of $\tilde{\mathcal{P}}(\epsilon)$, then $x(0)$ is a vertex of $\mathcal{P}$

This is equivalent to consider the polyhedron $\mathcal{P}_{lex} = \{x \in \mathcal{M}_{n,m+1}(\mathbb{R}) \mid Ax \geq_{lex} (b| - I_m)\}$, where $\leq_{lex}$ is the row-wise order deduced from the lexicographic order between vectors of $\mathbb{R}^{m+1}$

Evolution

1. Adjacency relation is basis wise : L is a list of couples (basis, point)
2. Lexicographic feasible basis : switch to $\mathcal{P}_{lex}$
3. Adjacency relation is n-regular : L is structured into a graph

Finally : the Input

Let G be the graph of the vertices of $\mathcal{P}$, and G_{lex} the one of $\mathcal{P}_{lex}$

Input

- $g_{lex} = (V_{lex}, E_{lex})$, each vertex of V_{lex} are labelled by one set $I \subseteq \mathbb{N}_m$ and one point $x \in \mathcal{M}_{n,m+1}(\mathbb{R})$
- $g = (V, E)$, each vertex of V is labelled by a point $x \in \mathbb{R}^n$

Finally: The Algorithm

"g_{lex} *is a subgraph of* G_{lex}": for each $(I, x) \in V_{lex}$

- x is feasible (checks if $Ax \geq_{lex} (b|I_m)$)
- checks if $A_I x = (b| - I_m)_I$

"$g_{lex} = G_{lex}$"

- for each $((I, x), (J, y)) \in E_{lex}$, checks $\#(I \cap J) = n - 1$
- for each $(I, x) \in V_{lex}$, checks that (I, x) has exactly n successors

"$g = G$"

- checks that g is the image of g_{lex} (first column projection)

Image of a Graph

Definition

Let $G = (V, E)$ a graph, and $f : V \mapsto V'$. Then the graph $f(G) = (f(V), E')$ is called the *image of* G *by* f, and is defined by

$$(v', w') \in E' \Leftrightarrow \exists v, \exists w, \begin{cases} v' \neq w' \\ f(v) = v' \\ f(w) = w' \\ (v, w) \in E \end{cases}$$

We want to show that the graph $G = (V, E)$ of $\mathcal{P}$ is the image of the graph of its lexicographic bases $G_{lex} = (V_{lex}, E_{lex})$.

Let $f : V_{lex} \mapsto V$ associating a lexicographic feasible basis I to the first column of x^I. We will prove that $G = f(G_{lex})$.

Graph Vertices

$f(V_{lex}) \subseteq V$:

- if I is a lexicographic feasible basis of $\mathcal{P}_{lex}$, then I is a feasible basis of $\mathcal{P}$ ...

Graph Vertices

$V \subseteq f(V_{lex})$:

- Let x a vertex of $\mathcal{P}$. Then $\exists c$ such that $\{x\}$ is the solution of the PL $\min\{c^T \cdot y \mid y \in \mathcal{P}\}$.
- We uses the simplex algorithm on the PL $\min\{c^T \cdot y \mid y \in \mathcal{P}_{lex}\}$. Let $X^{opt} \in \mathcal{M}_{(n,m+1)}$ its output.
- Let $X = (x \mid 0)$, then

$$
\begin{aligned}
c^T \cdot X^{opt} &\leq_{lex} c^T \cdot X \\
\Rightarrow c^T \cdot X_1^{opt} &\leq c^T \cdot x \\
\Rightarrow c^T \cdot X_1^{opt} &= c^T \cdot x \\
\Rightarrow X_1^{opt} &= x
\end{aligned}
$$

Graph Edges

We have $f(E_{lex}) \subseteq E$ easily ...

Graph Edges

Let x, y adjacent in $\mathcal{P}$, then $\exists c$ such that $[x; y]$ is the solution of the PL $\min\{c^T \cdot z \mid z \in \mathcal{P}\}$.

Let X^{opt} the output of the simplex algorithm applied on the PL $\min_{lex}\{c^T \cdot z \mid z \in \mathcal{P}_{lex}\}$. Then $c^T \cdot X_1^{opt} = c^T \cdot x = c^T \cdot y$.

- X_1^{opt} is in the segment $[x; y]$.
- X_1^{opt} is a vertex.

$X_1^{opt} \in \{x, y\}$.

Graph Edges

Let assume that $X_1^{opt} = x$.

We will find a PL on which we will apply the simplex algorithm. Let Y^{opt} its output. It will verify:

- Y_1^{opt} = y
- For all bases I encountered by the simplex, $X_1^I \in \{x, y\}$

Graph Edges

Let k, such that y activates $A_k z \geq b_k$, but not x.

Let $c' = c - \epsilon A_k^T$, with ϵ "small enough".

- $c'^T \cdot X^{pivot(I)} \leq_{lex} c'^T \cdot X^I \Rightarrow forall I \in Simplex, X_1^I \in \{x, y\}$.
- $c'^T \cdot y < c'^T \cdot x \Rightarrow Y_1^{opt} = y$.

Proposition

$$G = f(G_{lex})$$

Outline

Intuitionistic Logic

Is there two $a, b \in \mathbb{R} \setminus \mathbb{Q}$ such that $a^b \in \mathbb{Q}$?

- If $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$, then $a = b = \sqrt{2}$.
- If $\sqrt{2}^{\sqrt{2}} \in \mathbb{R} \setminus \mathbb{Q}$, then $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$.

In this proof, we don't know whether $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$ or not.

In intuitionistic logic, we no longer consider truth values, but constructive proofs.

Curry-Howard Correspondence

Proof of $A \Rightarrow B$: "Assume A (the hypothesis) and conclude B (the conclusion)."

$\Leftrightarrow$

Function of Type $A \to B$: "A function taking a proof p_A of A and constructing a proof p_B of B"

Calculus Of Inductive Constructions

Every term t of Coq is associated to a type T. We denote it $t : T$.

Types:

- Data: bool, $\mathbb{N}$, $\mathbb{R}$ . . . (e.g $n : \mathbb{N}$ means "n is a natural number")
- Functions : $f : A -> B$ means "f is a function taking $x : A$ and returning $(f\,x) : B$"
- Propositions : $h : \forall n : \mathbb{N}, n \geq 0$ means "h is a proof that every natural number is bigger than 0"
- Types: *Prop*, *Type* are the type of types. We call them *Sorts*.

Calculus of Inductive Constructions

```
Inductive and (A B : Prop) : Prop :=
  |conj : A -> B -> and A B.

Inductive or (A B : Prop) : Prop :=
  |or_introl: A -> or A B
  |or_intror: B -> or A B.
```

conj, *or_introl* and *or_intror* are called constructors

Calculus of Inductive Constructions

```
Inductive nat : Set :=
    |O : nat
    |S : nat -> nat.
```

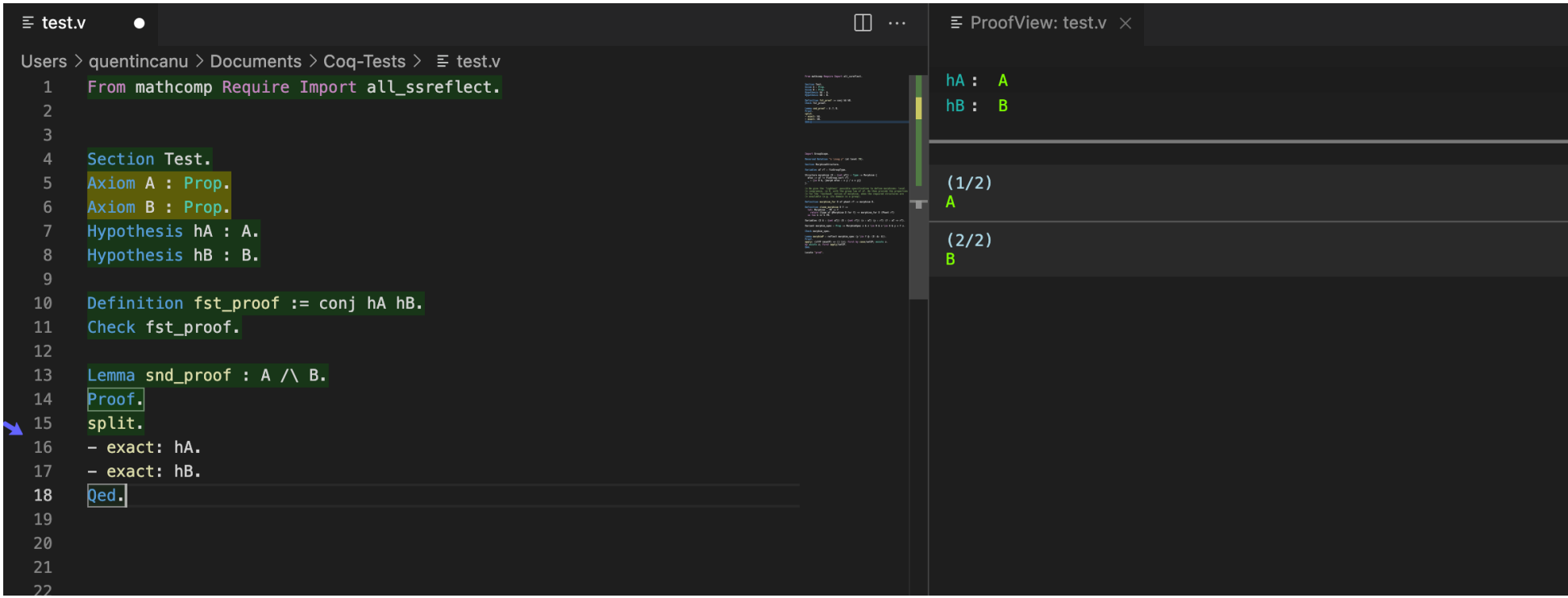
The type of *O*, *(S O)*, *(S (S O))* is *nat*.

”Polyhedral” Example

```
Inductive prebasis : predArgType :=
    |Prebasis :
        (I : {set 'I_m}) -> (_ : (#|I| == n)%N) -> prebasis.

Inductive basis : predArgType :=
    |Basis :
        (bas: prebasis) -> (_ : is_basis bas) -> basis.
```

Tactics


test.v
Users > quentincanu > Documents > Coq-Tests > test.v
1 From mathcomp Require Import all_ssreflect.
2
3
4 Section Test.
5 Axiom A : Prop.
6 Axiom B : Prop.
7 Hypothesis hA : A.
8 Hypothesis hB : B.
9
10 Definition fst_proof := conj hA hB.
11 Check fst_proof.
12
13 Lemma snd_proof : A /\ B.
14 Proof.
15 split.
16 - exact: hA.
17 - exact: hB.
18 Qed.
19
20
21
22
ProofView: test.v
hA : A
hB : B
(1/2)
A
(2/2)
B

Two Levels of Data Types

Low Level

- Used to computations
- Focused on efficiency

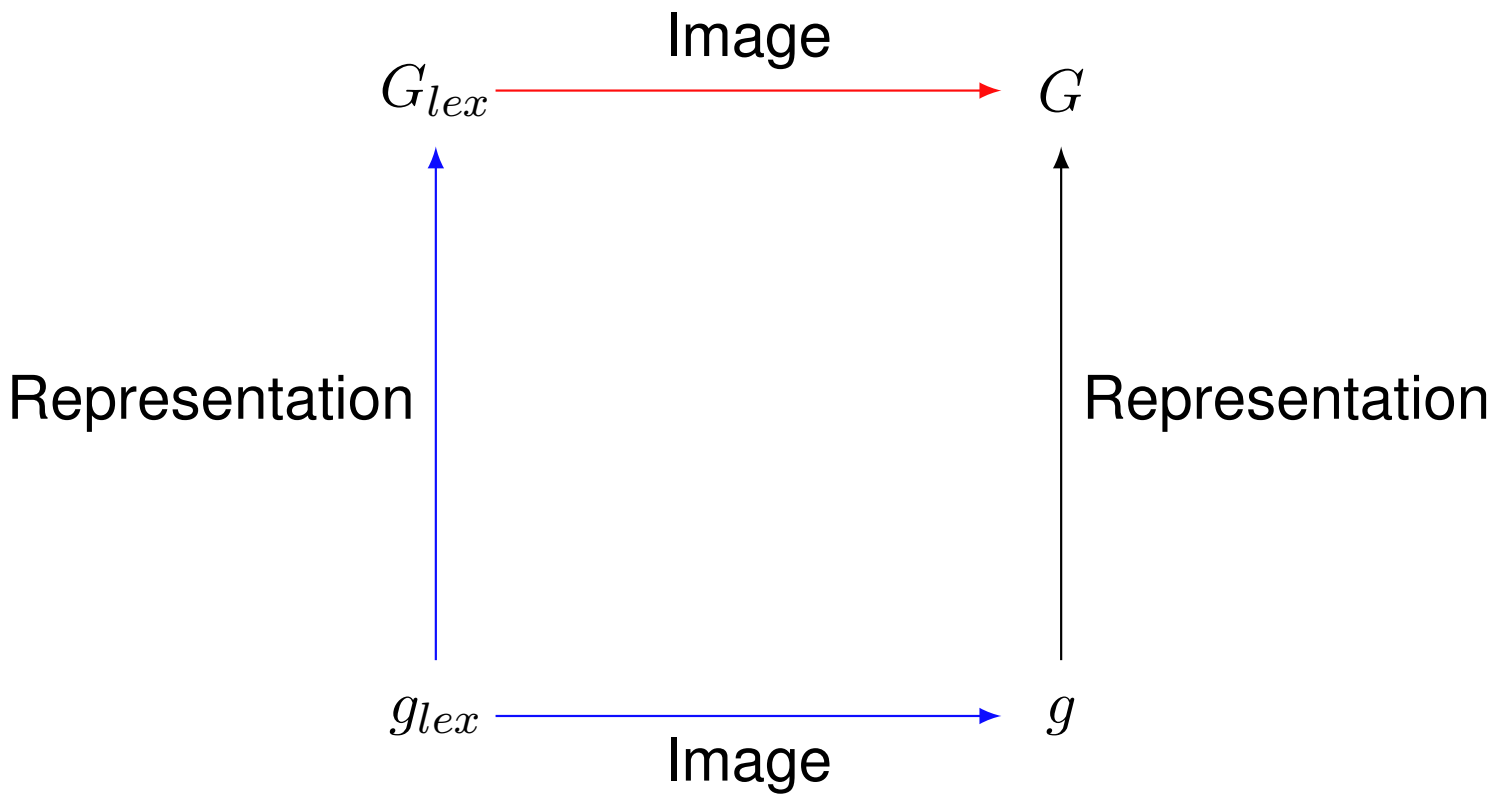
High Level

- Abstract
- Close to the mathematical concepts
- Used to prove the correctness of the algorithm

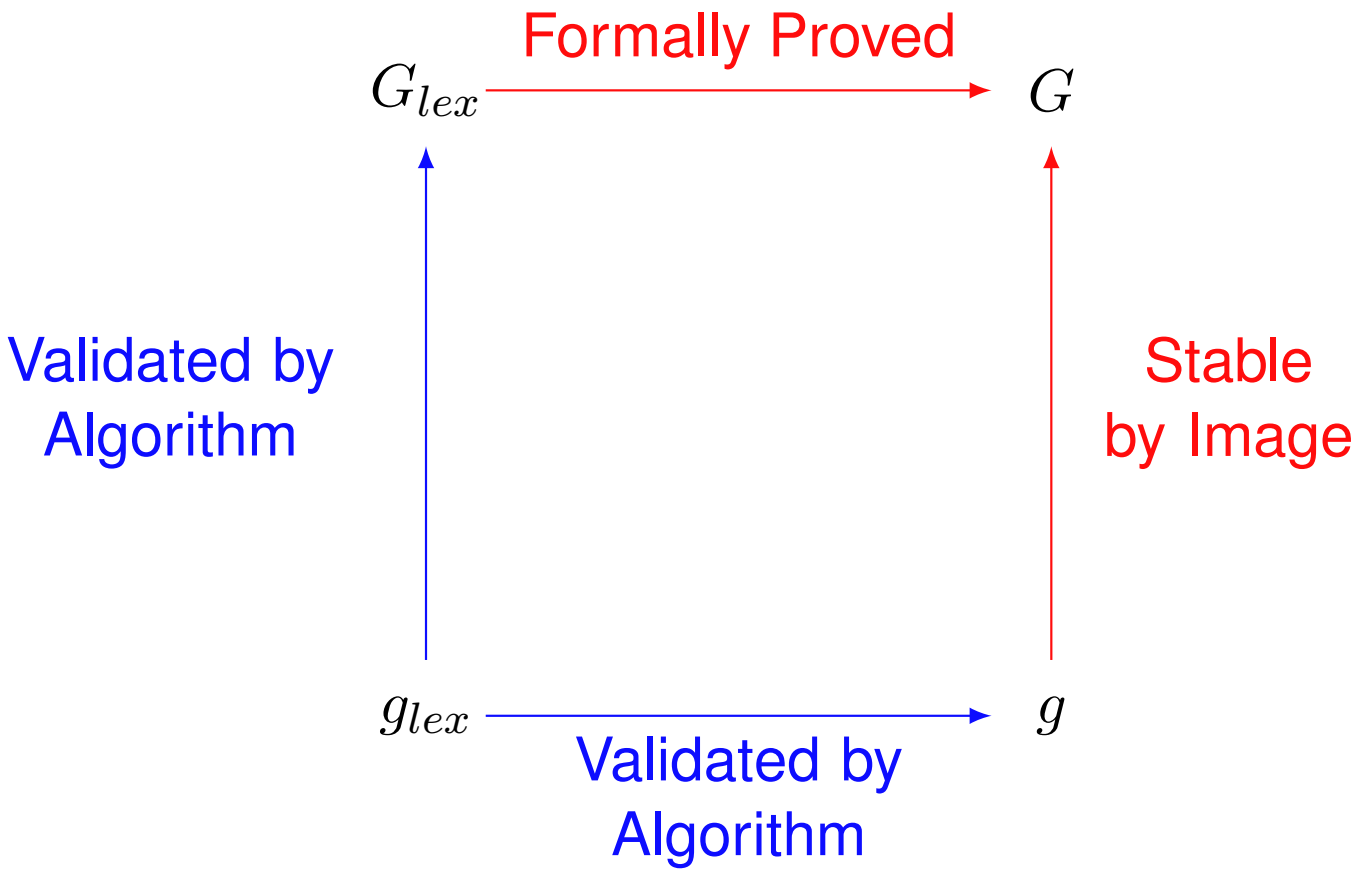
Refinement between Data Types

Relations between types are defined to maintain equivalence between low-level and high-level

So far...



So far...


Formally Proved
G_{lex}
G
Validated by Algorithm
Stable by Image
g_{lex}
g
Validated by Algorithm

Exemple of Implementation : The Simplex Algorithm

```
Inductive phase2_final_result :=
| Phase2_res_unbounded (bas: lex_feasible_basis) of 'I
| Phase2_res_optimal_basis of lex_feasible_basis.

Inductive phase2_intermediate_result :=
| Phase2_final of phase2_final_result
| Phase2_next_basis of lex_feasible_basis.
```

Exemple of Implementation : The Simplex Algorithm

```
Definition basic_step bas :=
  let u := reduced_cost_of_basis c bas in
  if [pick i | u i 0 < 0] is Some i then
    let d := direction i in
    if (@idPn (feasible_dir A d)) is
        ReflectT infeas_dir then
      Phase2_next_basis (lex_rule_fbasis infeas_dir)
    else Phase2_final (Phase2_res_unbounded i)
  else
    Phase2_final (Phase2_res_optimal_basis bas).
```

Use Case : Santos Counter-Example

Polyhedron disproving Hirsch Conjecture

- Dimension : 20
- Number of half-spaces : 40
- Number of vertices : 36425

Validation of lexicographic feasible basis takes $\simeq 186$ minutes