Implementation and Certification of a CAD algorithm

Assia Mahboubi

INRIA Sophia-Antipolis (Marelle)

Proofs and computations

- More and more *mathematical* proofs involve *machine* computations
  Examples :
  - graph combinatorics : the Four Colour Theorem (1973, K.Appel, W.Haken)
  - sphere packing density : proof of the Kepler Conjecture (1997, T.Hales, S. Ferguson)
- Huge collaborative proofs may be difficult to gather in a coherent, comprehensive document.
  Example : Finite Groups Classification ...

$\Rightarrow$ Need for formal verifications of computations, and machine check of mathematical statements.

The Coq system

A proof assistant based on the
Calculus of Inductive Constructions (extension of type theory)
Proofs:

- A property is represented by the set of its proofs
- Proving a property = Showing that the set of its proofs is not empty
- System of types
- Every term of the language has a type
- A proposition is represented by a type
- Proving a proposition is providing a term of the required type

Programs : A functional language with a (strong) type system

The Coq system

Further consequences of the Curry-de Bruijn-Howard isomorphism

- Notion of computation (typed programming language, compiler)
- Specification of objects
- Proof objects, interactively built by *tactics*
- Proof checking is type-checking
- Intuitionistic (unless you add axioms...)

A strong need for automation

Proofs are completed by:

- Application of inference rules
- Computation steps ($2 + 2 = 4$ for natural numbers)
- Rewriting of axioms $(\forall x, y \in \mathbb{N},\ x + y = y + x)$ or lemmas

«Hand-made» proofs are soon very tedious!

Automation is mandatory

- Computation of normal forms (e.g. ring equalities)
- (Proof producing) decision procedures (e.g. Presburger arithmetic)

A decision procedure for real numbers

- δ-ϵ lemmas in the following style:

$$\forall x \in \mathbb{R},\ \forall \epsilon > 0,\ \exists \delta > 0, |x_0 - x| < \delta \Rightarrow |{x_0}^3 - x^3| < \frac{\epsilon}{2\sqrt{2}}$$

- ... and more generally all the formulas of the language of a real closed field

can be decided (Tarski 1948).

Existential problems

Input :

$P_1, \ldots, P_s$ polynomials in $\mathbb{Q}[X_1, \ldots, X_n]$

$\sharp_1, \ldots, \sharp_s \in \{<, >, =\}$ sign conditions

Output :

Existence of a solution in $\mathbb{R}^n$ for the system :

$$\begin{cases} P_1 & \sharp_1 & 0 \\ \vdots & & \\ P_s & \sharp_s & 0 \end{cases}$$

An autarkic approach

For a skeptical approach (cf. the talk of M.Micaela Mayero):

- the problem may be difficult to solve...
- ... but the solution is “easy” to check (algebraic certificates...)
- trust a skillful oracle to guess the solution
- check its answer

Here:

- the solution (positive or negative) is not easy to check...
- the introduction of a compiler to the Coq system makes computations feasible

The design of the tactic

Design a tool (a Coq tactic) deciding and proving automatically formulas of the language of the real closed field of real numbers.
Sketch of the method:

- A decision procedure is programmed using Coq as a programming language
- The correction theorem of this function is fomally proved within the Coq system
- The user states a first order formula in the theory of real numbers
- The decision function *computes* the answer, using Coq's reduction machine (and compiler)
- The output proof is the instanciation of the correctness proof for the user's problem.

The decision procedure

- One variable : root isolation (rational approximation of roots) with Bernstein polynomials
- Several variables : Cylindrical Algebraic Decomposition (Collins, 1975) with subresultant elimination
  - Elimination of variables by a projection operator
  - Root isolation over polynomials with *algebraic* coefficients

One dimension problems:Root isolation

The method :

- Isolate the roots of $P_1, \ldots, P_s$ by disjoint intervals with rational endpoints
- Compute the signs at the roots (possibly using gcd computations)
- Compute the signs outside the isolating intervals (between the roots)

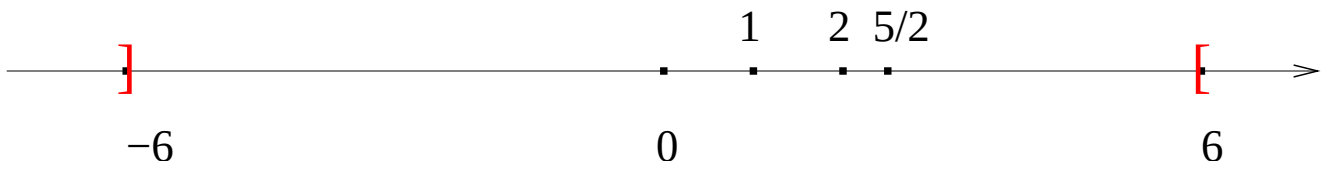
One dimension problems:Root isolation

- Bound the roots and compute the finite length relevant interval (Cauchy formulas)
- Use the following blackbox:
  - P a polynomial in $\mathbb{Q}[X]$, $c, d \in \mathbb{Q}$,
  - $magic(P, c, d) = 0 \quad \Leftrightarrow \quad$ no root for P in $]c, d[$
  - $magic(P, c, d) = 1 \quad \Leftrightarrow \quad$ exactly one root for P in $]c, d[$
  - $magic(P, c, d) = \perp \quad \Rightarrow \quad$ we don't know (but try again)

Example

$P_1 = (X-1)(X-2)$

don't know



Example

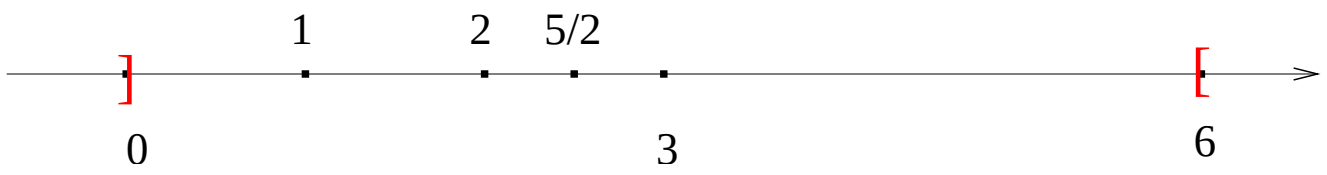
$$P_1 = (X-1)(X-2)$$

no root

Example

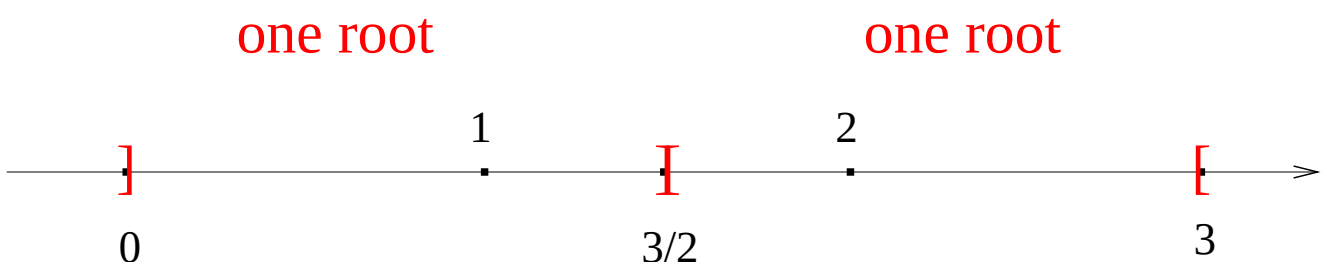
$$P_1 = (X-1)(X-2)$$

don't know



Example

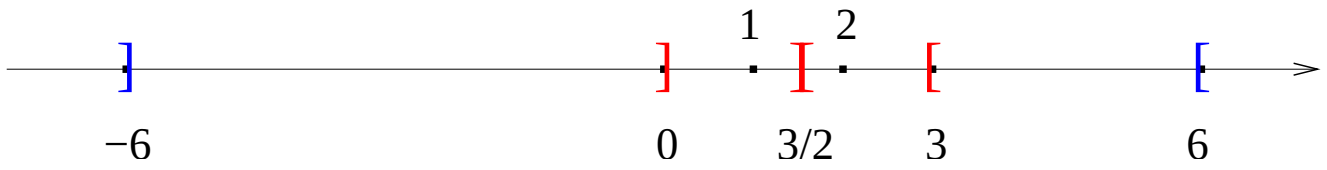
$$P_1 = (X-1)(X-2)$$



Example

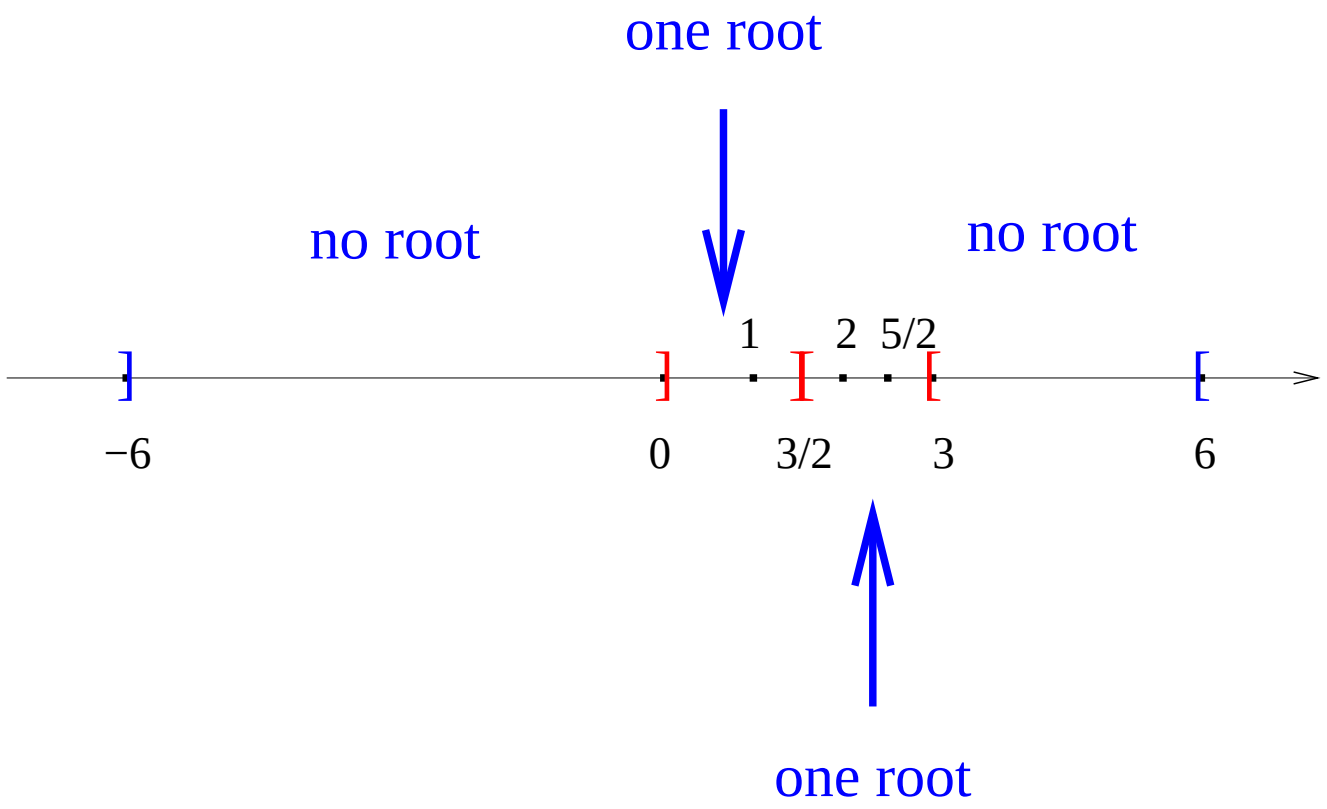
$P_1 = (X-1)(X-2)$, $P_2 = (X-1)(X-\frac{5}{2})$

don't know



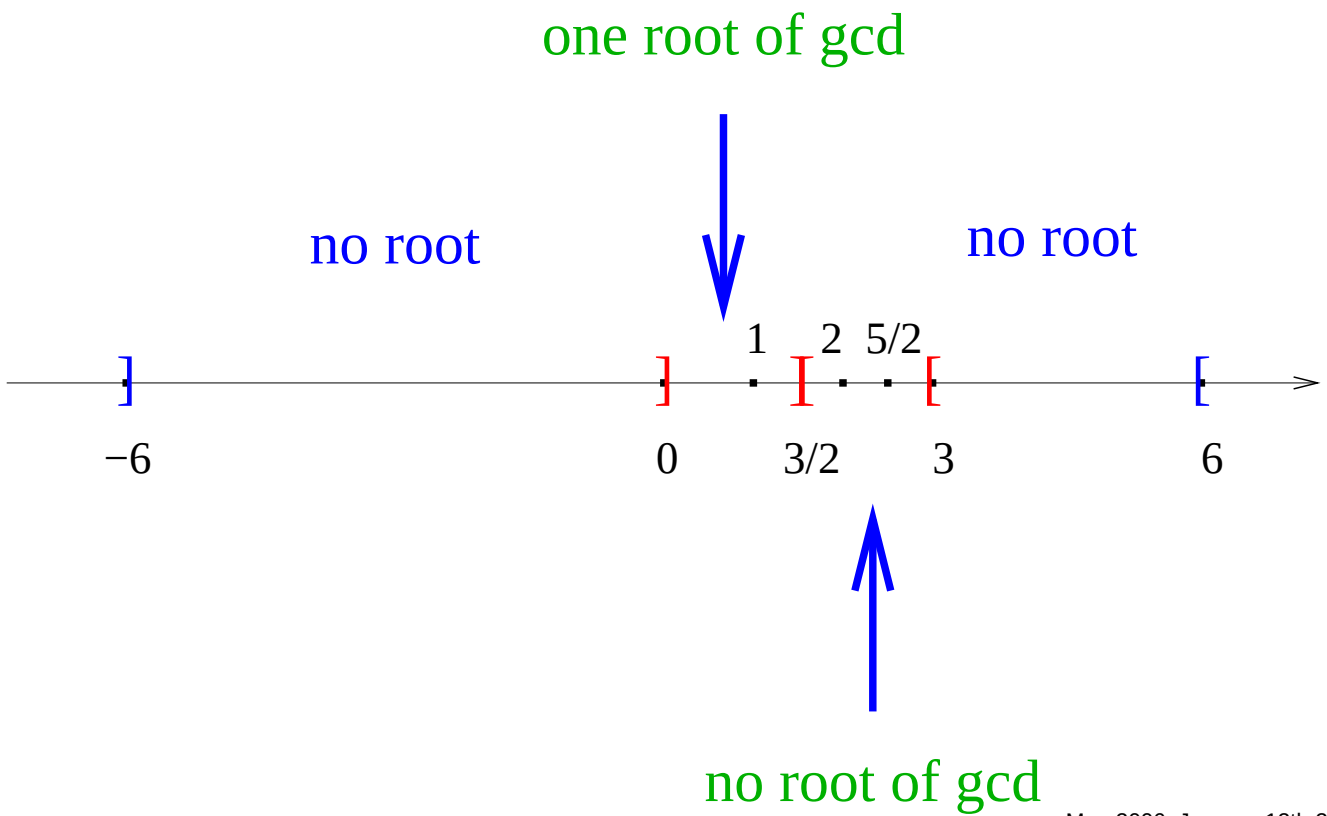
Example

$P_1 = (X-1)(X-2)$, $P_2 = (X-1)(X-\frac{5}{2})$



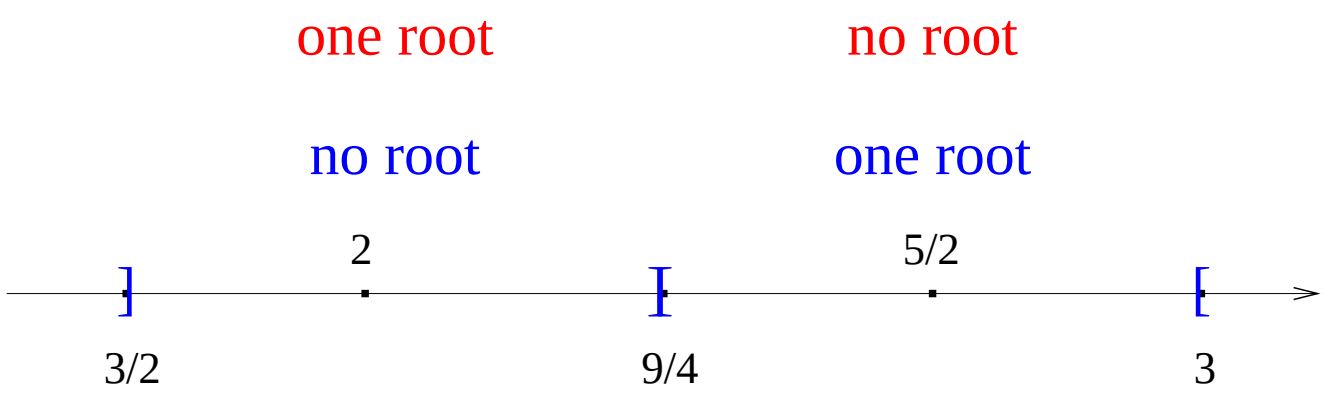
Example

$$P_1 = (X-1)(X-2),\ P_2 = (X-1)(X-\tfrac{5}{2})\ P_1 \wedge P_2 = (X-1)$$



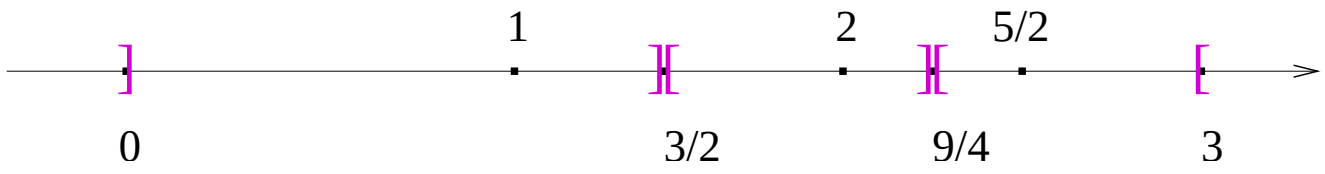
Example

$P_1 = (X-1)(X-2)$, $P_2 = (X-1)(X-\frac{5}{2})$



Example

$P_1 = (X-1)(X-2)$, $P_2 = (X-1)(X-\frac{5}{2})$



The blackbox : Bernstein polynomials

Definition :

$$B_{p,i}(c,d) = \binom{p}{i} \frac{(X-c)^i (d-X)^{p-i}}{(d-c)^p} \quad i \leq p \quad c,d \in \mathbb{Q}$$

Properties :

- A basis for polynomials of degree $\leq p$
- Let P be a polynomial in $\mathbb{Q}[x]$,
- Let $c, d \in \mathbb{Q}$
- Let b the list of the coefficients of P in the Bernstein basis for p, c, d.
- $magic(P, c, d) =$ the number of sign changes in b

Upper dimension : Cylindrical Algebraic Decompositio

$$\mathbb{R}[X_1, \ldots, X_n] \qquad\qquad \mathbb{R}[X_1, \ldots, X_{n-1}]$$

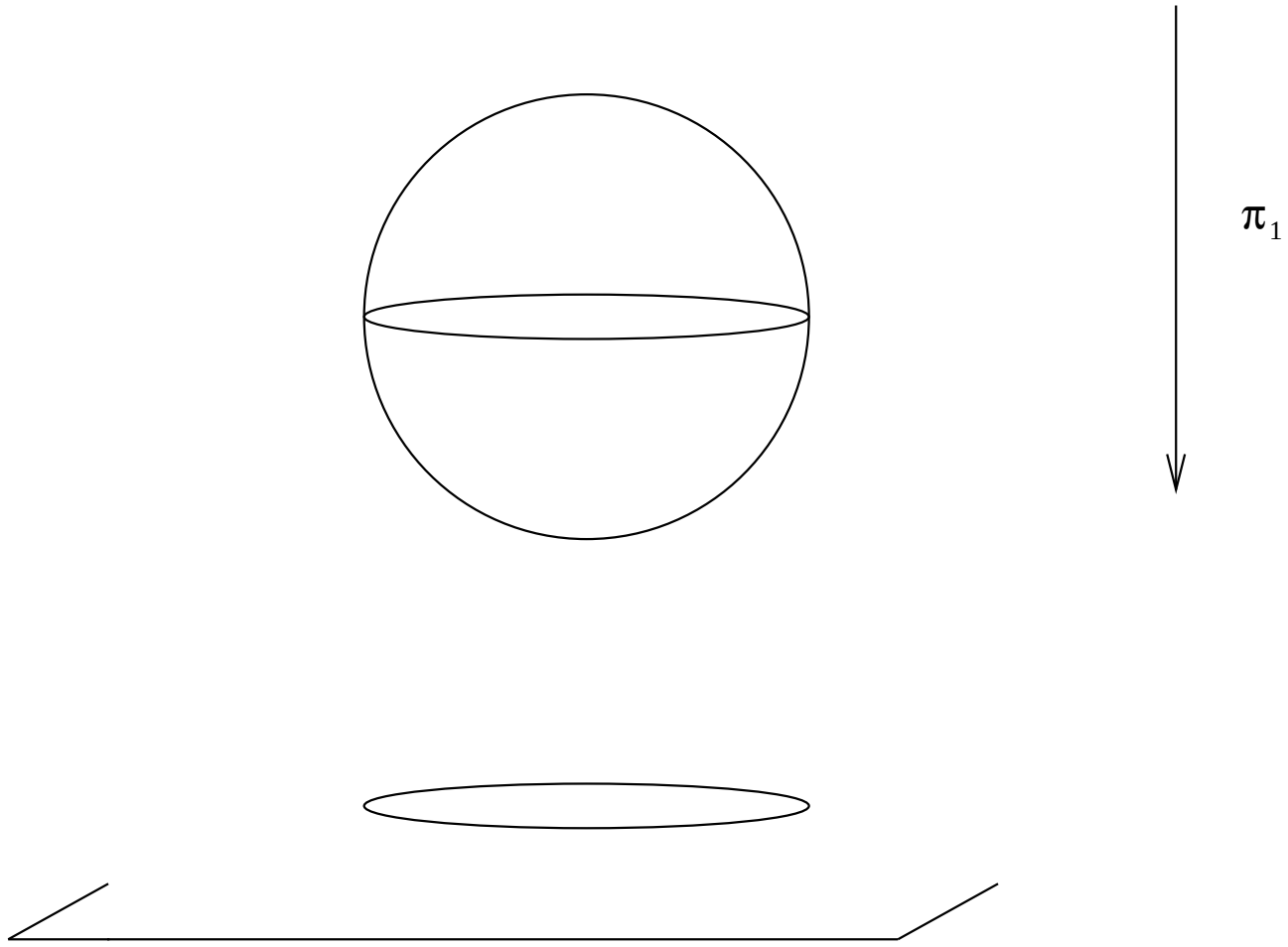
$$\mathcal{P} = P_1, \ldots, P_s \xrightarrow[operator]{projection} \mathcal{Q} = Q_1, \ldots, Q_t$$

$$\text{CAD and signs for } \mathcal{P} \xleftarrow[phase]{lifting} \text{CAD and signs for } \mathcal{Q}$$

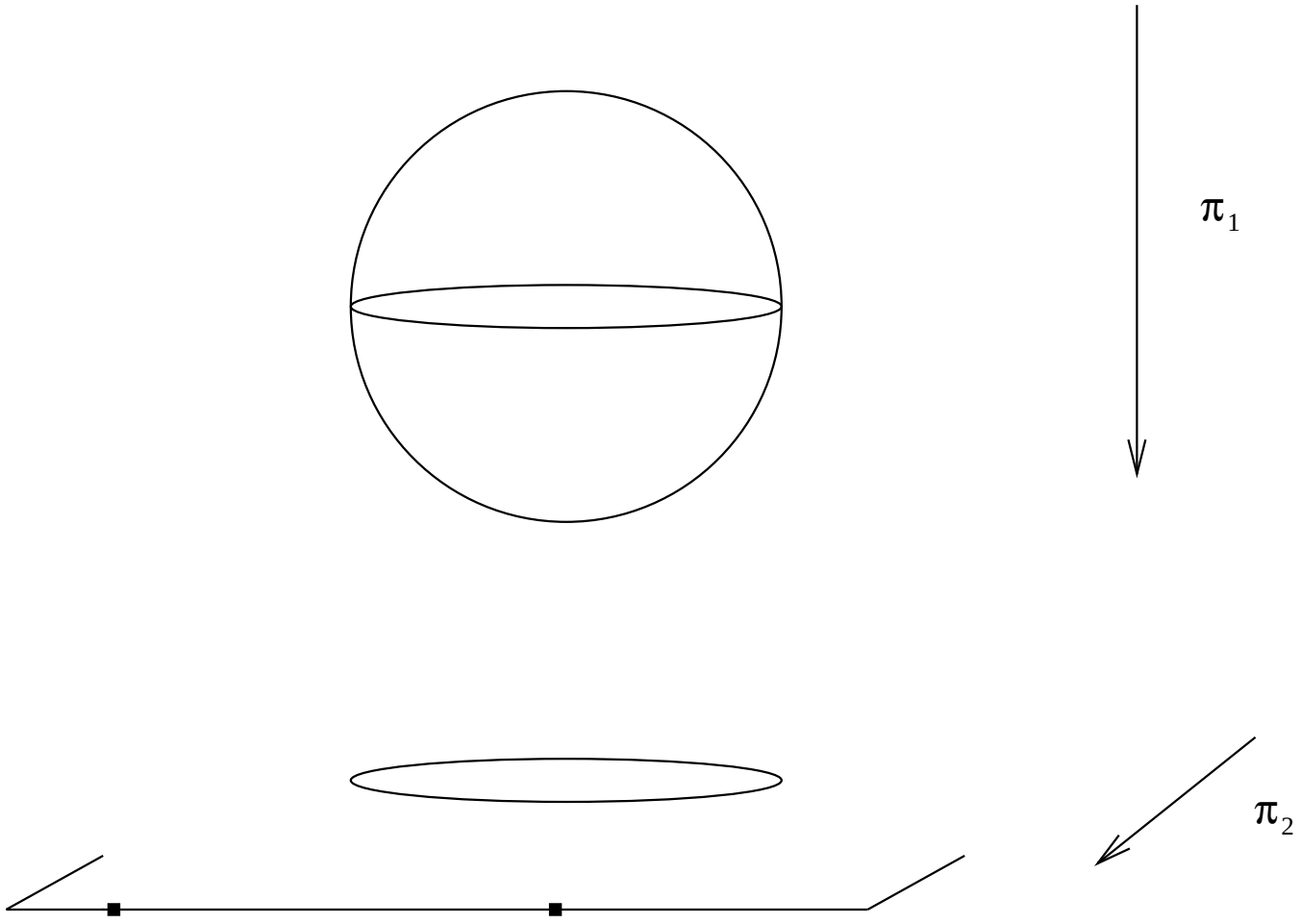
$$\mathbb{R}^n \qquad\qquad \mathbb{R}^{n-1}$$

- Cells for the CAD adapted to $\mathcal{P}$:
  - take a cell C of the CAD adapted to $\mathcal{Q}$
  - consider the cylinder $C \times \mathbb{R}$ above C
  - cut it *in a polynomial way*

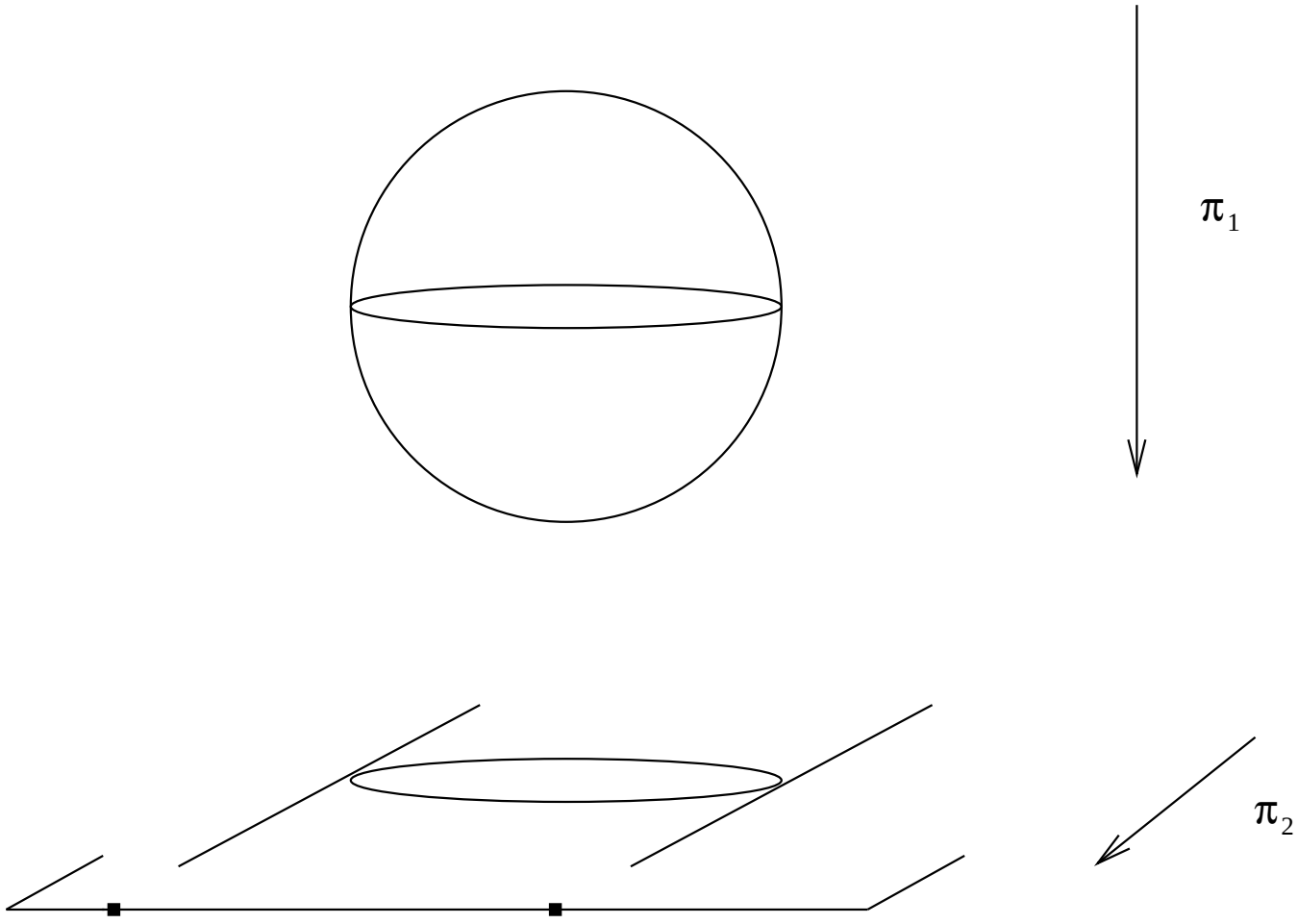
Example



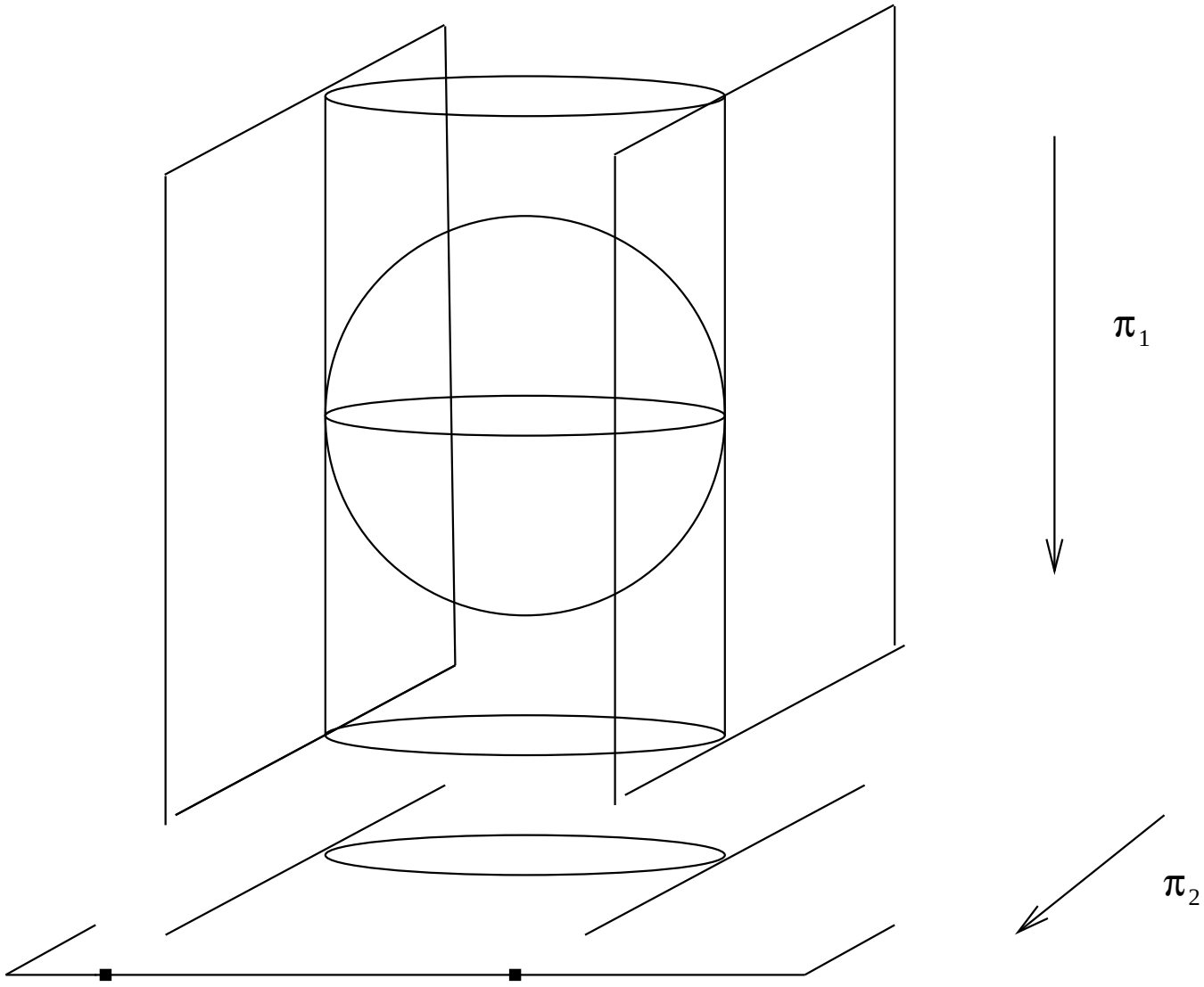
Example



Example



Example



General decision problem

- First transform into prenex form
- Then search into the tree of realizable sign conditions
- Need for a non computational version of the trichotomy (thanks to the Prop/Set distinction)
- A formula is proved $\Leftrightarrow$ there is no classical counter-example

What do we need to implement

- A library on polynomial arithmetic (ring operations, gcd,...) on integral domains
- Root isolation for univariate polynomials with rational coefficients
- Root isolation for univariate polynomials with algebraic coefficients
- Projector operator
- Recursion settings
- Extra : an abstract representation for rational numbers

Architecture of the (computational) developpement

- Need for abstraction to avoid code duplication
- Coq enjoys a module mechanism
  - A nice way to abstract the representation of rational coefficients
  - Not powerful enough for the recursive treatment
- Use tricks with name sharing and files loading

Architecture of the (proofs) developpement

- No extraction : proofs of the functions previously implemented in Coq
- Need for abstraction to avoid proof duplication (polynomials with coefficients in an integral domains)
- Need for automation to treat tedious, low-level proofs (ring identities, arithmetic on binary integers...)
- $\hookrightarrow$ Implementation of symbolic tactics and term rewriting strategies

Current State

- Decision procedure is programmed in Coq following the algorithms described in:
  Algorithms in Real Algebraic Geometry,
  S.Basu, R. Pollack, M.-F. Roy
- Several more general tools as byproducts (improved tactic for solving ring equalities)
- A toy-CAS programmed in Coq, with efficiency concerns (representation of polynomials, algorithms)
- Example previously proved with hours of work (see J.Avigad) are now proved in a few minutes

Ongoing work

The formal proof of this procedure is a huge work and will necessarily be collaborative (joint work with the Marelle team)

- Proofs are in progress :
  - polynomial toolbox
  - correction of Bernstein polynomials properties
  - correction of the subresultant algorithm (technical part of the projection operator, and gcd algorithm)
- Find “good” orders for elimination
- Build heuristics on to of the complete method