Non-linear Invariants for Control-Command Systems

Pierre Roux

ONERA, Toulouse, France

September 6th 2019

Control-Command Systems

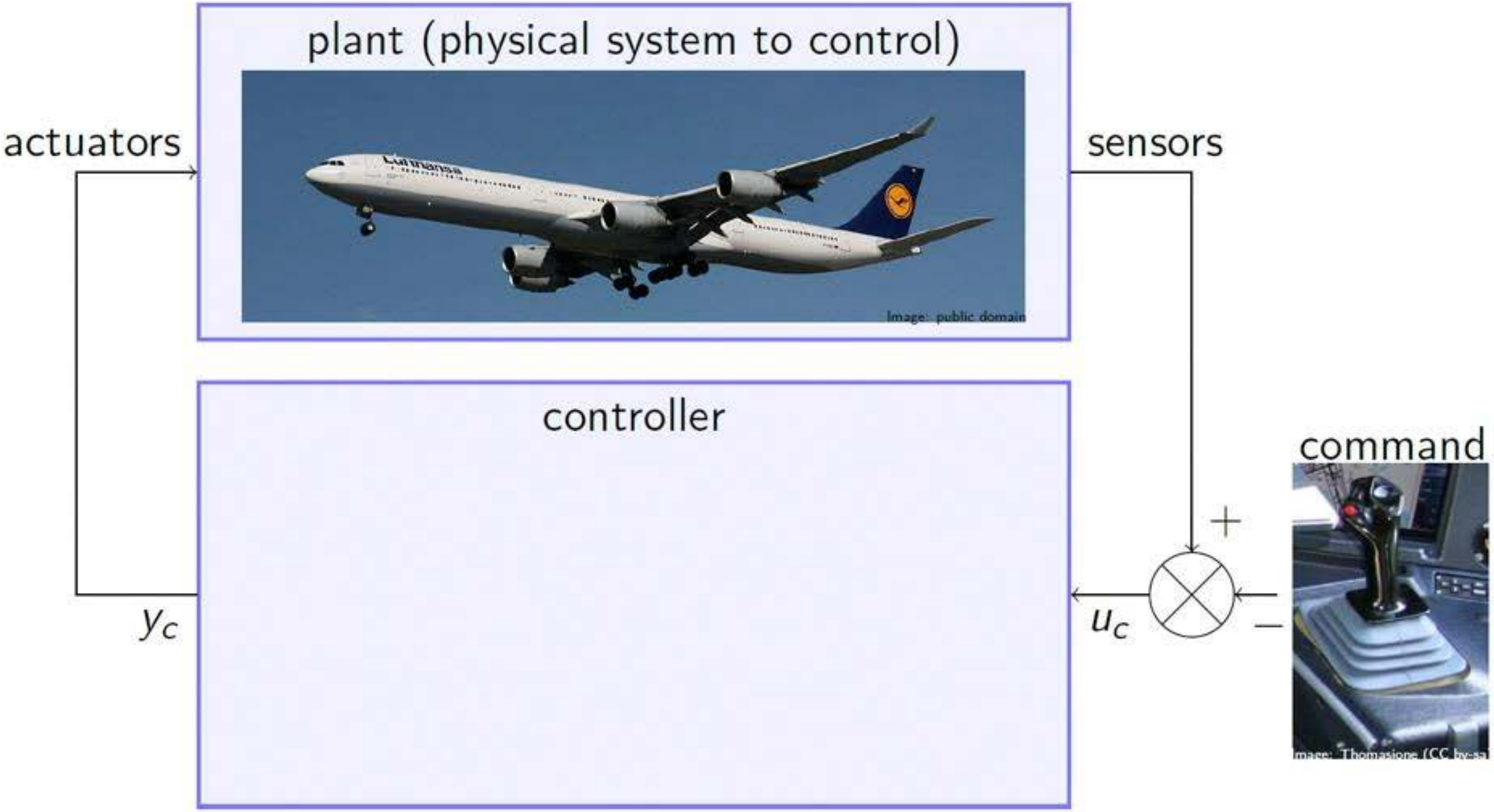
plant (physical system to control)

Image: public domain

command

Image: Thomasione (CC by-sa)

Control-Command Systems



Control-Command Systems

plant (physical system to control)

Image: public domain

actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

u_c

+

−

command

mage: Thomasione (CC by-sa

Control-Command Systems

plant (physical system to control)

actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

u_c

$+$

$-$

command

Control-Command Systems

plant (physical system to control)

actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

y_c

u_c

+

−

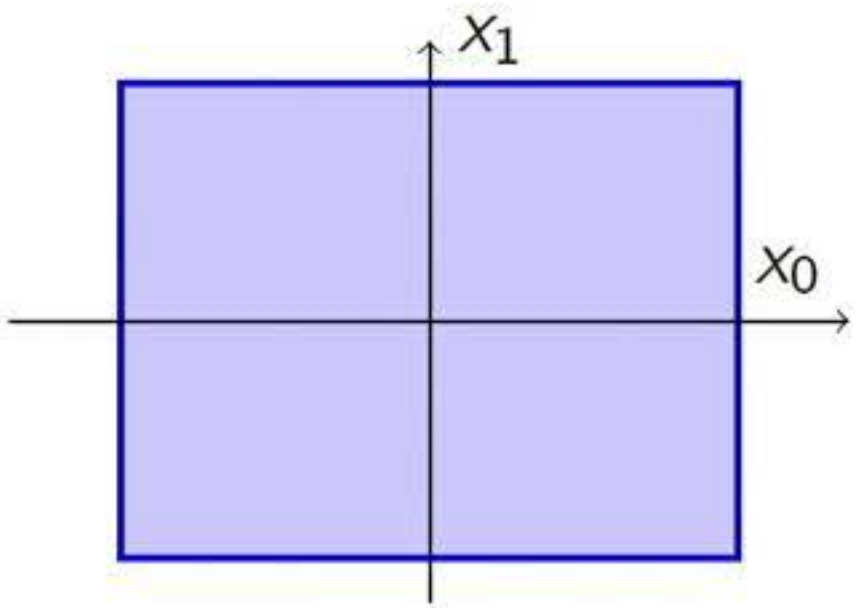
command

Quadratic invariants

- *Linear invariants* commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.

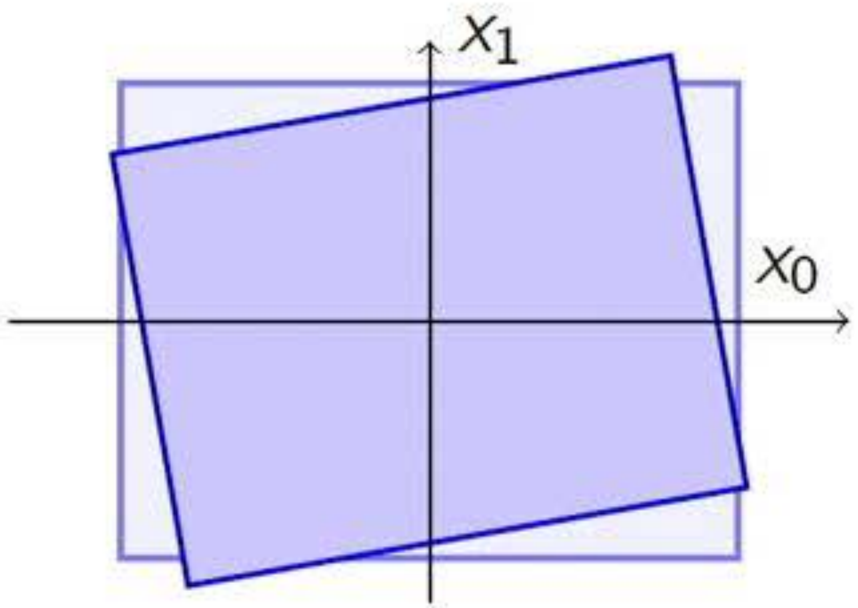
Quadratic invariants

- *Linear invariants* commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



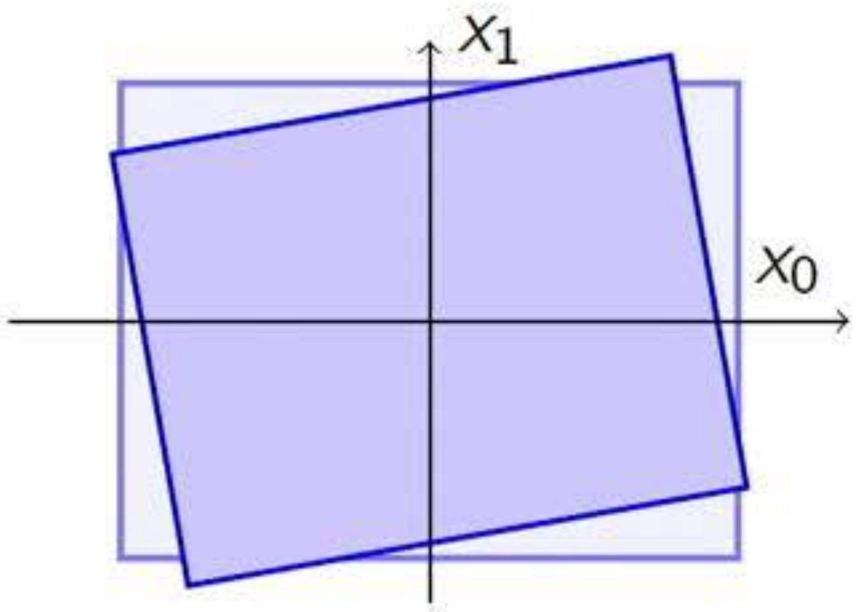
Quadratic invariants

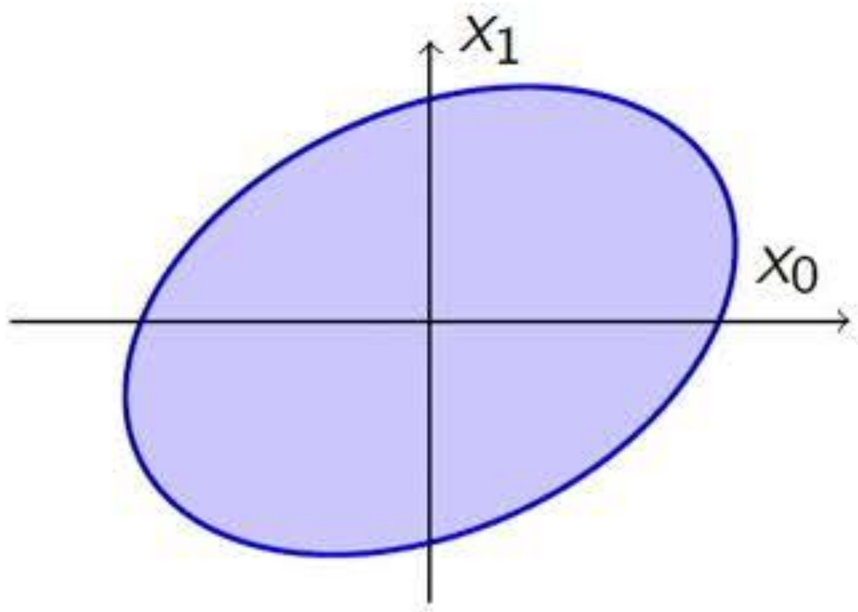
- *Linear invariants* commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



Quadratic invariants

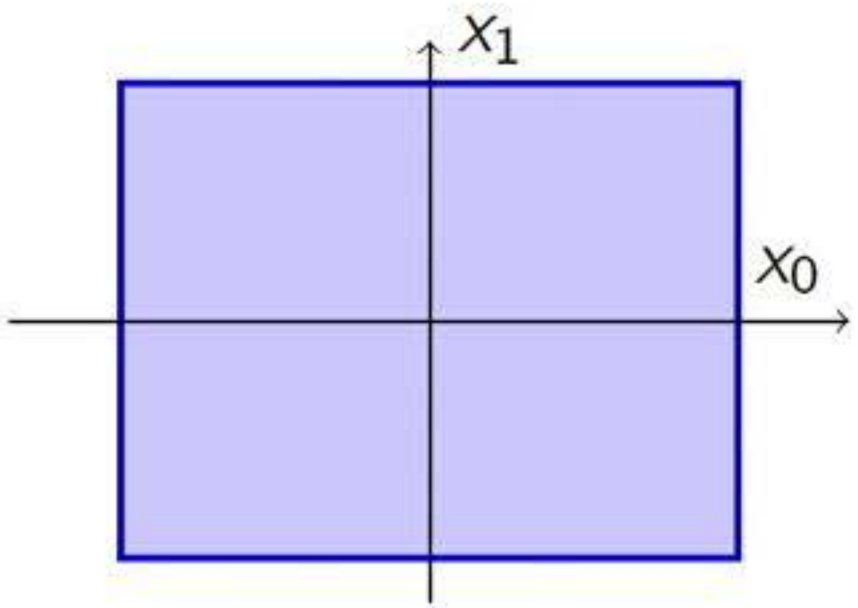
- *Linear invariants* commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.
- Control theorists know for long that *quadratic invariants* are a good fit for linear systems.





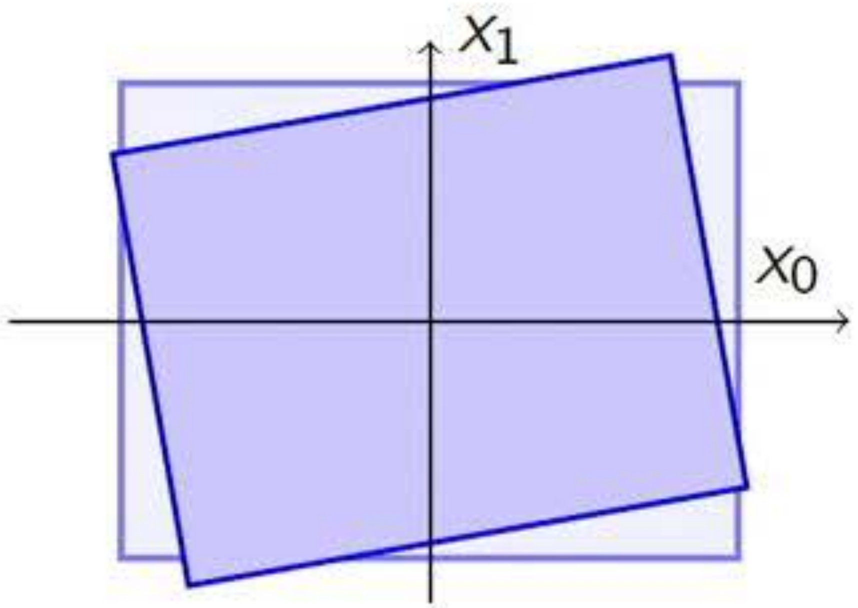
Quadratic invariants

- *Linear invariants* commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



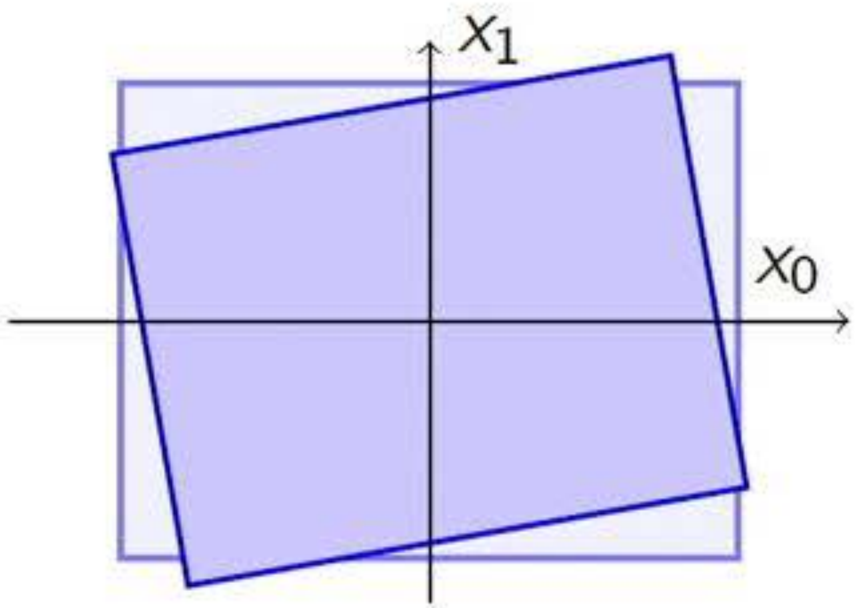
Quadratic invariants

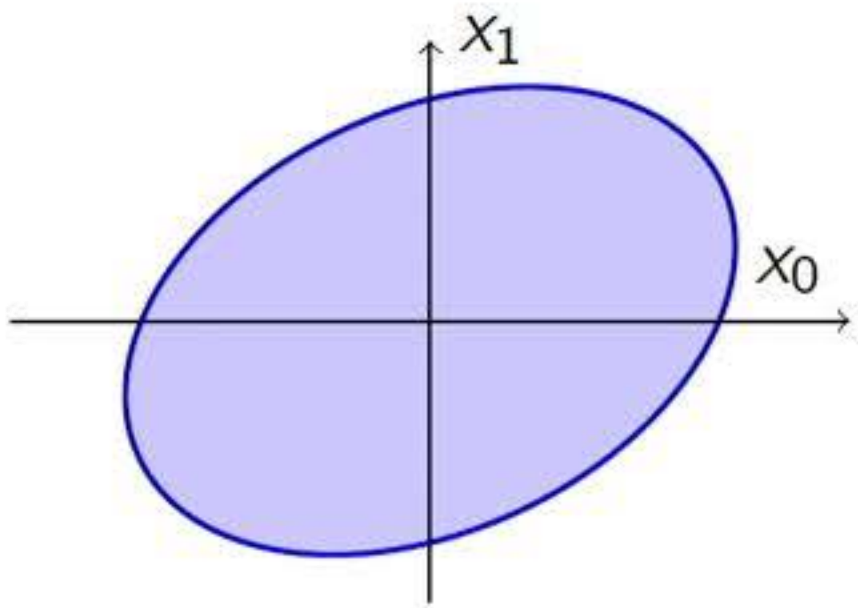
- *Linear invariants* commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.



Quadratic invariants

- *Linear invariants* commonly used in static analysis are not well suited:
  - at best costly;
  - at worst no result.
- Control theorists know for long that *quadratic invariants* are a good fit for linear systems.





Example

SMT solvers have a hard time with non-linear numerical problems.

Demo

```
typedef struct { double x0, x1, x2; } state;

/*@ predicate inv(state *s) =
  @   6.04 * s->x0 * s->x0 - 9.65 * s->x0 * s->x1
  @   - 2.26 * s->x0 * s->x2 + 11.36 * s->x1 * s->x1
  @   + 2.67 * s->x1 * s->x2 + 3.76 * s->x2 * s->x2 <= 1; */

/*@ requires \valid(s) && inv(s) && -1 <= in0 <= 1;
  @ ensures inv(s); */
void step(state *s, double in0) {
  double pre_x0 = s->x0, pre_x1 = s->x1, pre_x2 = s->x2;

  s->x0 = 0.9379*pre_x0 - 0.0381*pre_x1 - 0.0414*pre_x2 + 0.0237*in0;
  s->x1 = -0.0404*pre_x0 + 0.968*pre_x1 - 0.0179*pre_x2 + 0.0143*in0;
  s->x2 = 0.0142*pre_x0 - 0.0197*pre_x1 + 0.9823*pre_x2 + 0.0077*in0;
}
```

Example (Demo)

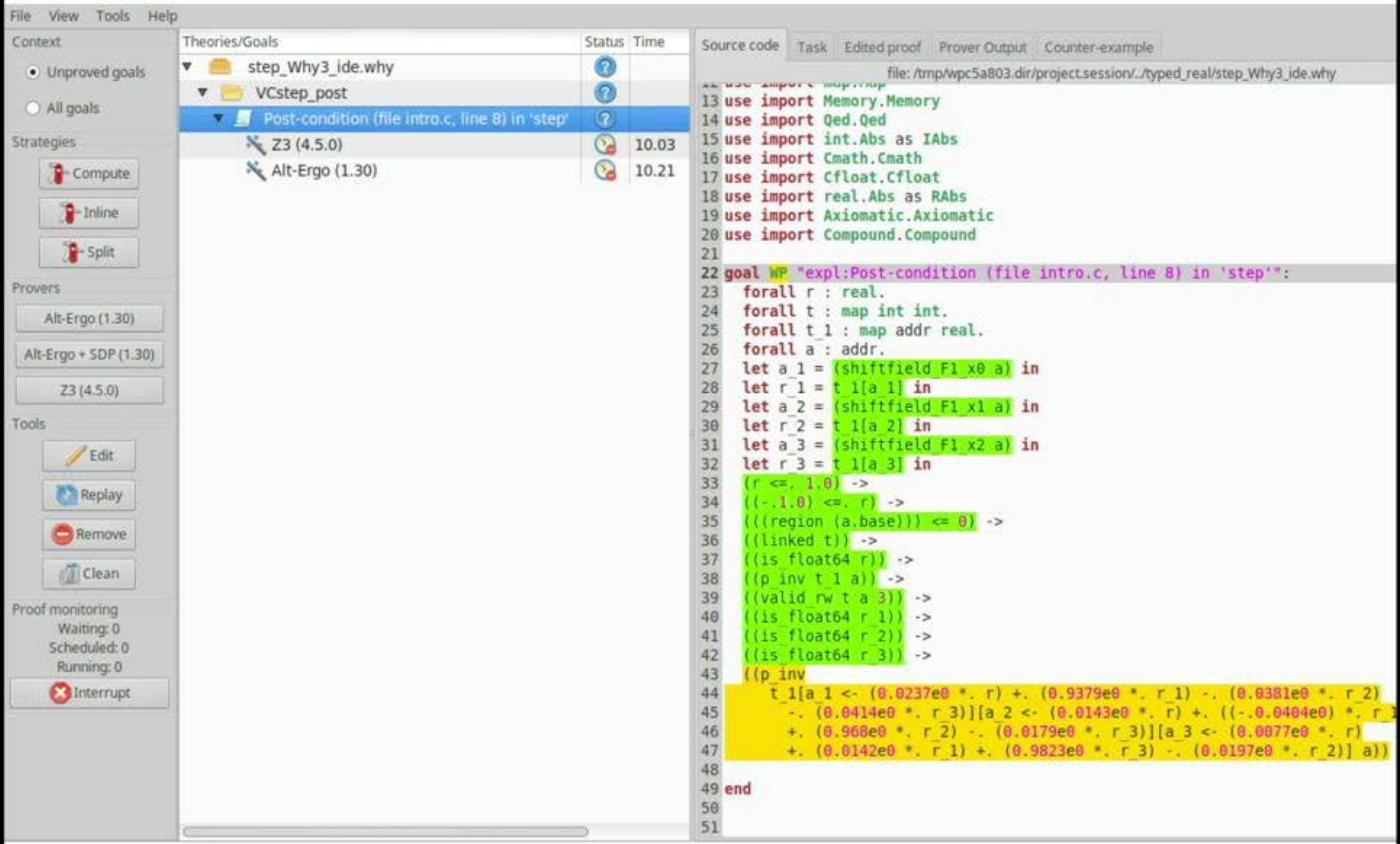
```
┌─(pierre@machine ~/slides)
└─% cat intro.c
typedef struct { double x0, x1, x2; } state;

/*@ predicate inv(state *s) = 6.04 * s->x0 * s->x0 - 9.65 * s->x0 * s
  @   - 2.26 * s->x0 * s->x2 + 11.36 * s->x1 * s->x1
  @   + 2.67 * s->x1 * s->x2 + 3.76 * s->x2 * s->x2 <= 1; */

/*@ requires \valid(s) && inv(s) && -1 <= in0 <= 1;
  @ ensures inv(s); */
void step(state *s, double in0) {
  double pre_x0 = s->x0, pre_x1 = s->x1, pre_x2 = s->x2;

  s->x0 = 0.9379 * pre_x0 - 0.0381 * pre_x1 - 0.0414 * pre_x2 + 0.023
0;
  s->x1 = -0.0404 * pre_x0 + 0.968 * pre_x1 - 0.0179 * pre_x2 + 0.014
0;
  s->x2 = 0.0142 * pre_x0 - 0.0197 * pre_x1 + 0.9823 * pre_x2 + 0.007
0;
}
┌─(pierre@machine ~/slides)
└─% frama-c -wp -wp-model real -wp-prover why3ide intro.c
```

Example (Demo)



Synthesizing Polynomial Invariants

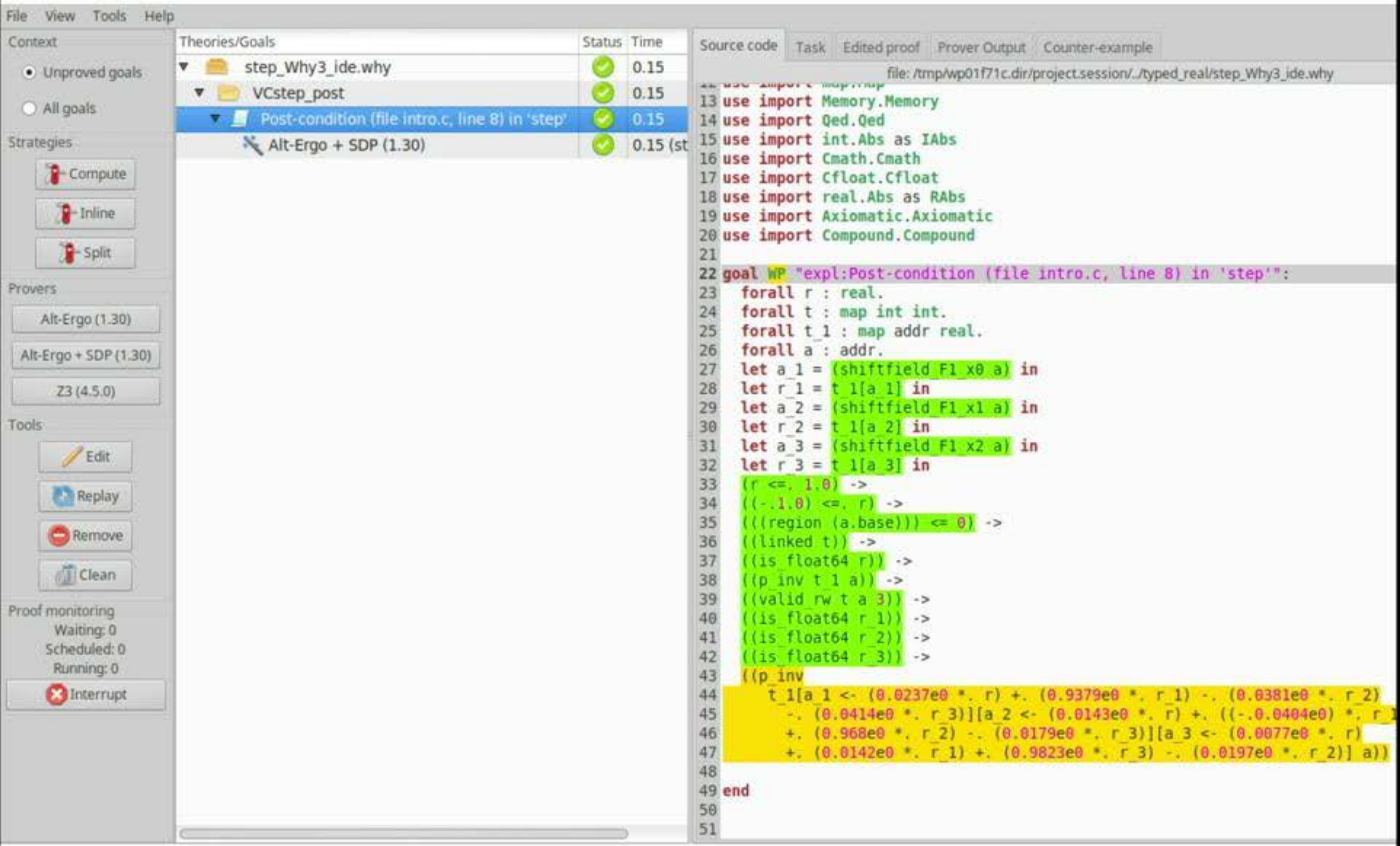
Ensuring Soundness

Bound Floating-Point Rounding Errors

Integration into a SMT Solver

Formalized Proofs with Coq

Example (Demo)



Synthesizing Polynomial Invariants

Ensuring Soundness

Bound Floating-Point Rounding Errors

Integration into a SMT Solver

Formalized Proofs with Coq

Synthesizing Polynomial Invariants

Ensuring Soundness

Bound Floating-Point Rounding Errors

Integration into a SMT Solver

Formalized Proofs with Coq

Polynomial Encoding

Consider the program

```
x = x0;
while (1) {
  in = input();   /* ∈ [−1, 1] */
  x = f(x, in);
}
```

When a polynomial p satisfies

$$
\begin{aligned}
& p(x_0) \geqslant 0 && \text{initial condition} \\
& p \circ f - p - \sigma \left(1 - in^2\right) \geqslant 0 && \text{inductiveness} \\
& \sigma \geqslant 0 && (p(x) \geqslant 0 \text{ implies } p(f(x)) \geqslant 0)
\end{aligned}
$$

Then $p \geqslant 0$ is an invariant.

Need to solve polynomial positivity problems.

Sum of Squares (SOS) Polynomials

Definition (SOS Polynomial)

A polynomial p is SOS if there are polynomials $q_1, \ldots, q_m$ s.t.

$$p = \sum_i q_i^2.$$

- If p SOS then $p \geqslant 0$

1

Sum of Squares (SOS) Polynomials

Definition (SOS Polynomial)

A polynomial p is SOS if there are polynomials $q_1, \ldots, q_m$ s.t.

$$p = \sum_i q_i^2.$$

- If p SOS then $p \geqslant 0$
- p SOS iff there exist $z := \left[1, x_1, x_2, x_1 x_2, \ldots, x_n^d\right]$ and[1] $Q \succeq 0$

$$p = z^T Q \, z.$$

$\Rightarrow$ SOS can be encoded as semidefinite programming (SDP).

[1] $Q \succeq 0$ means Q *positive semidefinite*: $\forall x, x^T Q \, x \geqslant 0$

SOS: Example

Example

Is $p(x, y) := 2x^4 + 2x^3y - x^2y^2 + 5y^4$ SOS ?

$$p(x, y) = \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}$$

that is

$$p(x, y) = q_{11}x^4 + 2q_{13}x^3y + 2q_{23}xy^3 + (2q_{12} + q_{33})x^2y^2 + q_{22}y^4$$

SOS: Example

Example

Is $p(x, y) := 2x^4 + 2x^3y - x^2y^2 + 5y^4$ SOS ?

$$p(x,y) = \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}$$

that is

$p(x, y) = q_{11}x^4 + 2q_{13}x^3y + 2q_{23}xy^3 + (2q_{12} + q_{33})x^2y^2 + q_{22}y^4$

hence $q_{11} = 2$, $2q_{13} = 2$, $2q_{23} = 0$, $2q_{12} + q_{33} = -1$, $q_{22} = 5$.

For instance

$$Q = \begin{bmatrix} 2 & -3 & 1 \\ -3 & 5 & 0 \\ 1 & 0 & 5 \end{bmatrix} = R^T R \qquad R = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$$

hence $p(x, y) = \frac{1}{2}\left(2x^2 - 3y^2 + xy\right)^2 + \frac{1}{2}\left(y^2 + 3xy\right)^2.$

Synthesizing Polynomial Invariants

Ensuring Soundness

Bound Floating-Point Rounding Errors

Integration into a SMT Solver

Formalized Proofs with Coq

Polynomial Invariants

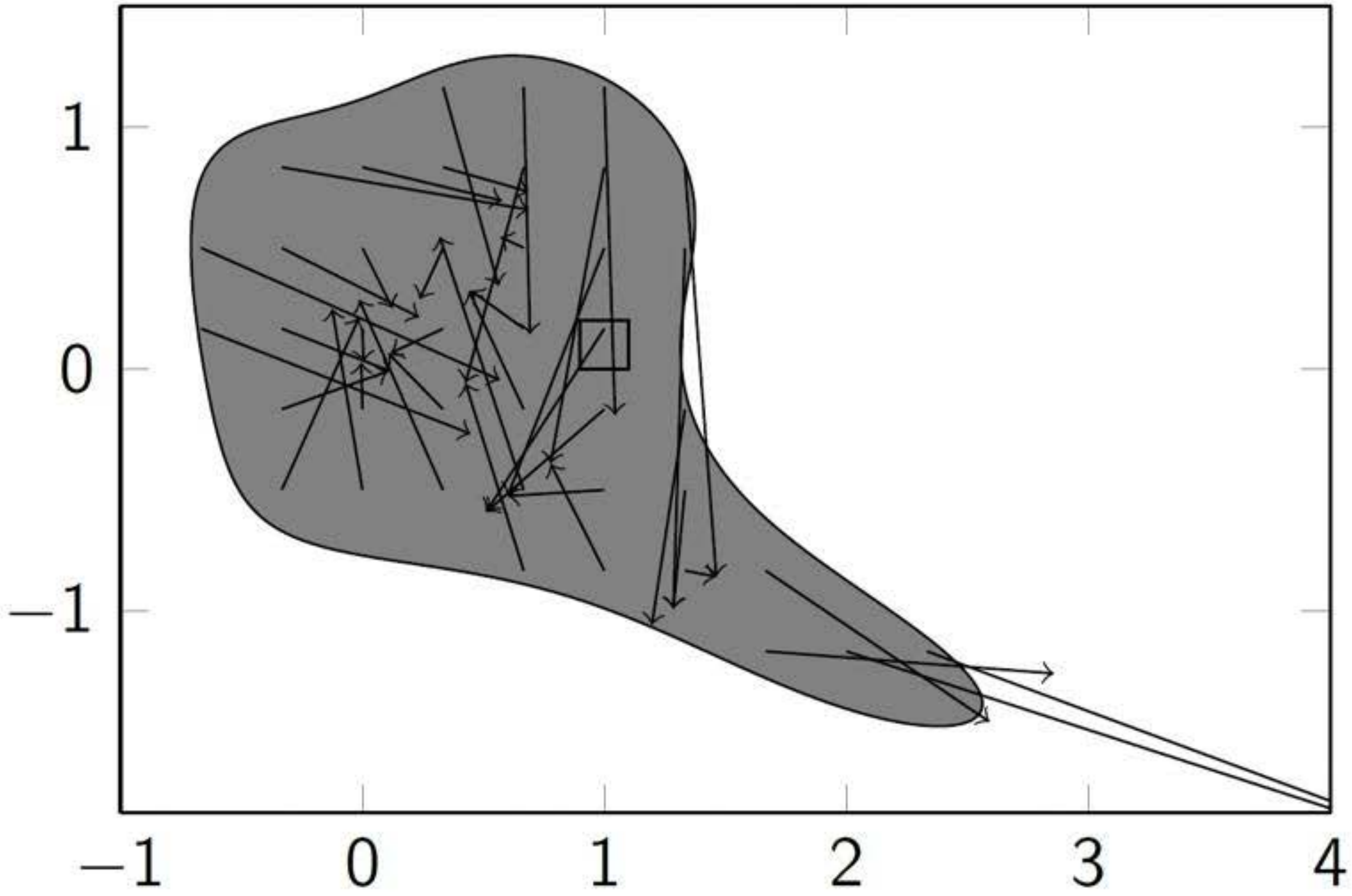
In a very nice SAS'15 paper, authors offer for

```
(x1, x2) ∈ [0.9, 1.1] × [0, 0.2]
while (1) {
  pre_x1 = x1; pre_x2 = x2;
  if (x1^2 + x2^2 <= 1) {
    x1 = pre_x1^2 + pre_x2^3;
    x2 = pre_x1^3 + pre_x2^2;
  } else {
    x1 = 0.5 * pre_x1^3 + 0.4 * pre_x2^2;
    x2 = -0.6 * pre_x1^2 + 0.3 * pre_x2^2;
  }
}
```

the inductive invariant $2.510902467 + 0.0050x_1 + 0.0148x_2 - 3.0998x_1^2 + 0.8037x_2^3 + 3.0297x_1^3 - 2.5924x_2^2 - 1.5266x_1x_2 + 1.9133x_1^2x_2 + 1.8122x_1x_2^2 - 1.6042x_1^4 - 0.0512x_1^3x_2 + 4.4430x_1^2x_2^2 + 1.8926x_1x_2^3 - 0.5464x_2^4 + 0.2084x_1^5 - 0.5866x_1^4x_2 - 2.2410x_1^3x_2^2 - 1.5714x_1^2x_2^3 + 0.0890x_1x_2^4 + 0.9656x_2^5 - 0.0098x_1^6 + 0.0320x_1^5x_2 + 0.0232x_1^4x_2^2 - 0.2660x_1^3x_2^3 - 0.7746x_1^2x_2^4 - 0.9200x_1x_2^5 - 0.6411x_2^6 \geqslant 0.$

Should we trust such results ?

- Some are correct (we'll prove it formally).
- Others aren't (previous degree 6 polynomial)



Polynomial Invariants

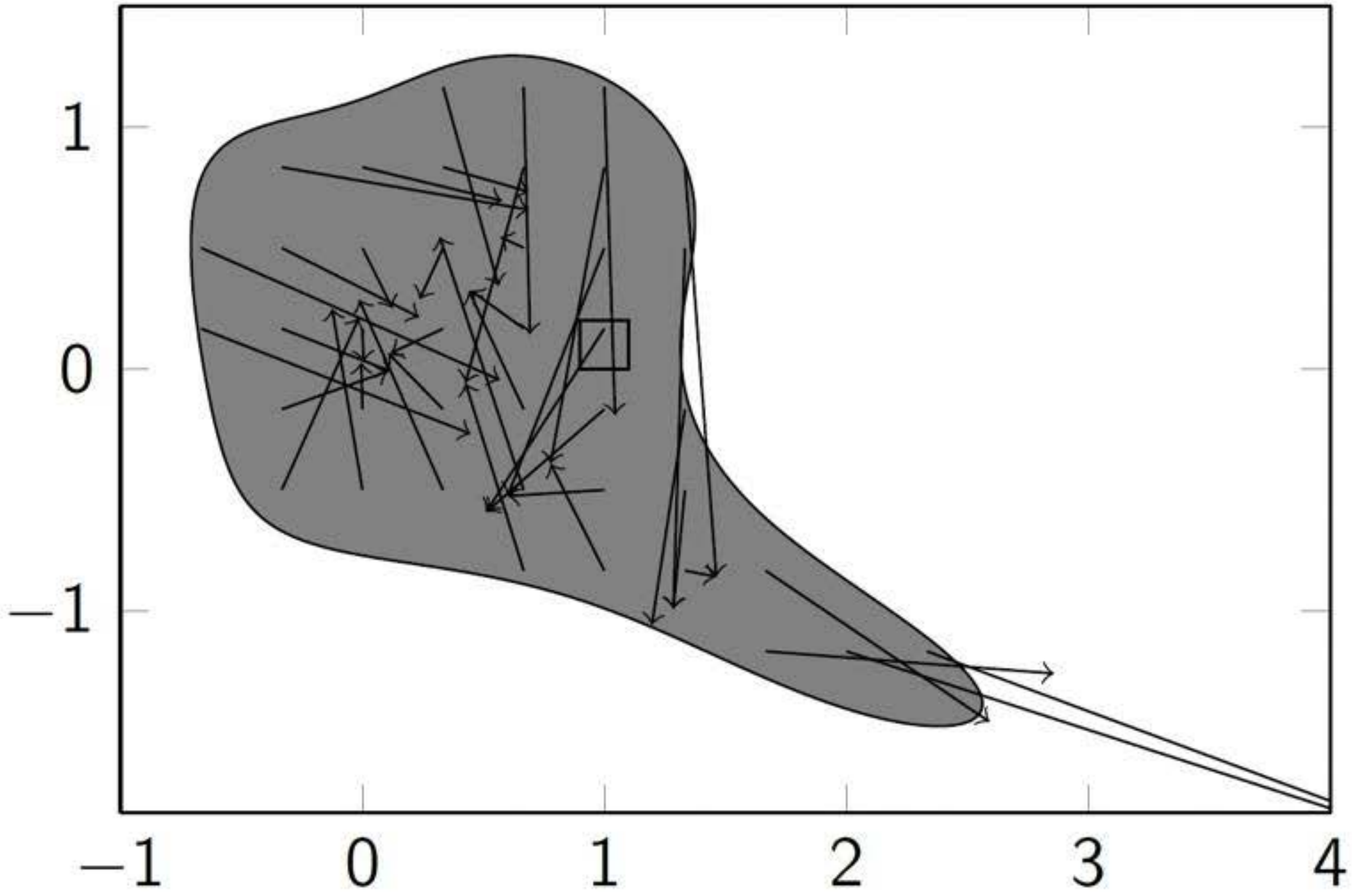
In a very nice SAS'15 paper, authors offer for

```
(x1, x2) ∈ [0.9, 1.1] × [0, 0.2]
while (1) {
  pre_x1 = x1; pre_x2 = x2;
  if (x1^2 + x2^2 <= 1) {
    x1 = pre_x1^2 + pre_x2^3;
    x2 = pre_x1^3 + pre_x2^2;
  } else {
    x1 = 0.5 * pre_x1^3 + 0.4 * pre_x2^2;
    x2 = -0.6 * pre_x1^2 + 0.3 * pre_x2^2;
  }
}
```

the inductive invariant $2.510902467 + 0.0050x_1 + 0.0148x_2 - 3.0998x_1^2 + 0.8037x_2^3 + 3.0297x_1^3 - 2.5924x_2^2 - 1.5266x_1x_2 + 1.9133x_1^2x_2 + 1.8122x_1x_2^2 - 1.6042x_1^4 - 0.0512x_1^3x_2 + 4.4430x_1^2x_2^2 + 1.8926x_1x_2^3 - 0.5464x_2^4 + 0.2084x_1^5 - 0.5866x_1^4x_2 - 2.2410x_1^3x_2^2 - 1.5714x_1^2x_2^3 + 0.0890x_1x_2^4 + 0.9656x_2^5 - 0.0098x_1^6 + 0.0320x_1^5x_2 + 0.0232x_1^4x_2^2 - 0.2660x_1^3x_2^3 - 0.7746x_1^2x_2^4 - 0.9200x_1x_2^5 - 0.6411x_2^6 \geqslant 0$.

Should we trust such results ?

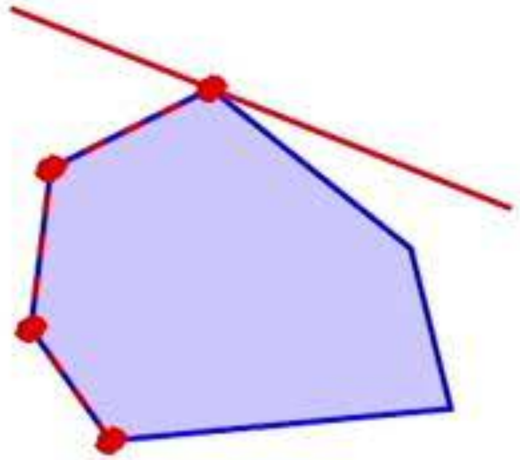
- Some are correct (we'll prove it formally).
- Others aren't (previous degree 6 polynomial)



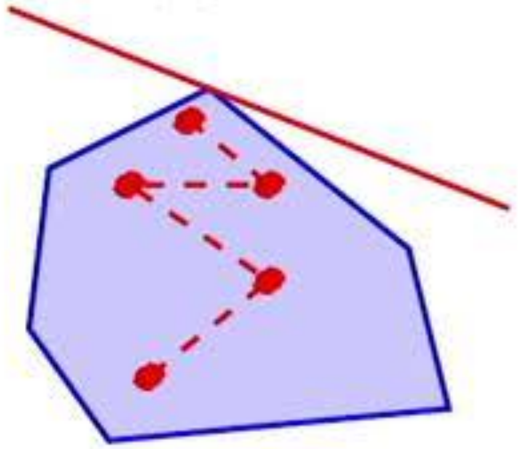
SDP solvers yield approximate solutions

- Linear programming

simplex: exact solution



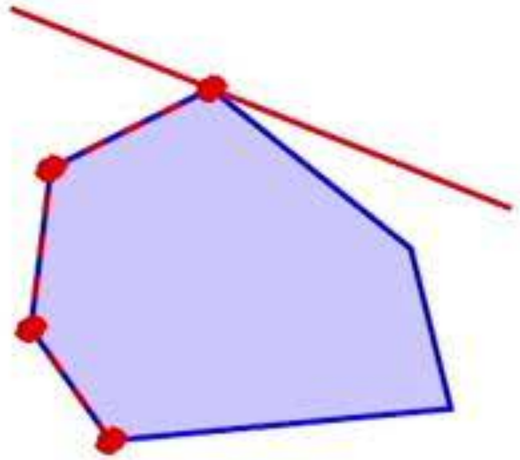
interior-point: approximate solution



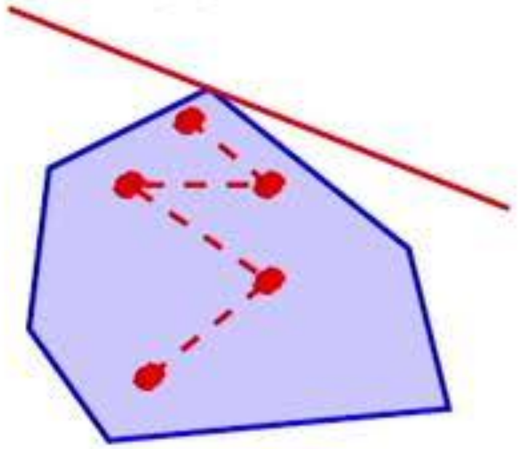
SDP solvers yield approximate solutions

- Linear programming

simplex: exact solution

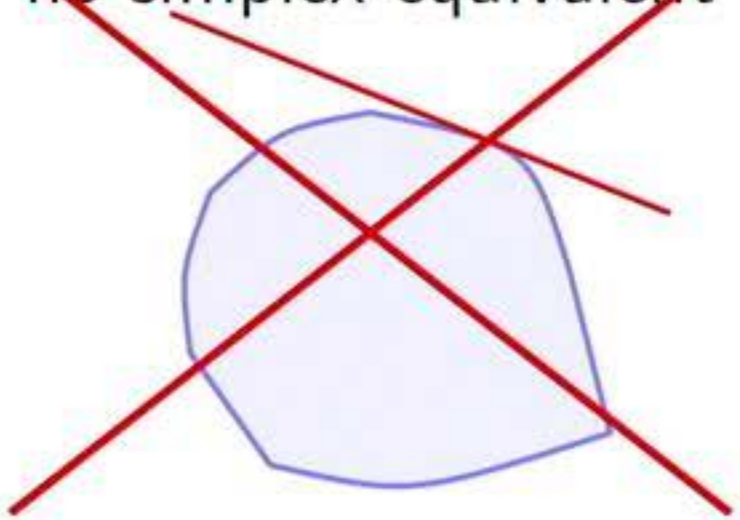


interior-point: approximate solution

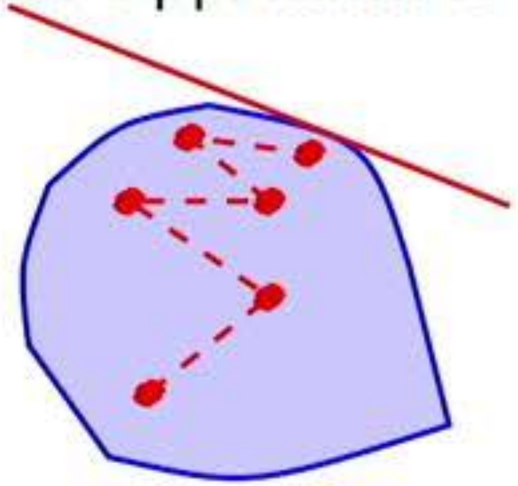


- Semidefinite programming

no simplex equivalent



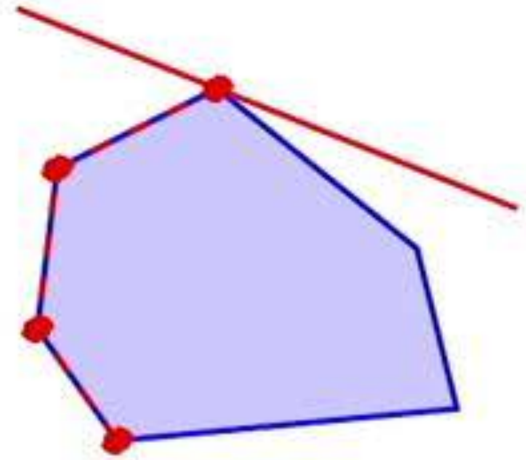
interior-point: approximate solution



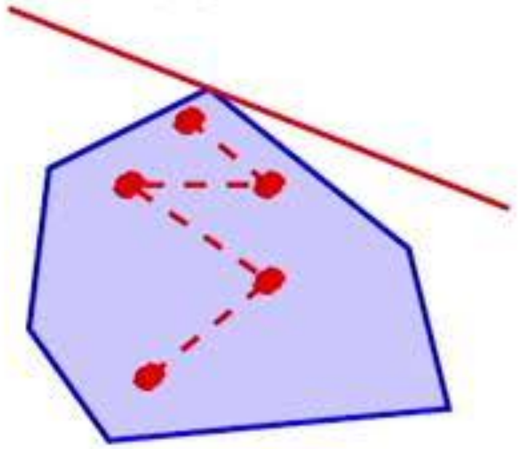
SDP solvers yield approximate solutions

- Linear programming

simplex: exact solution

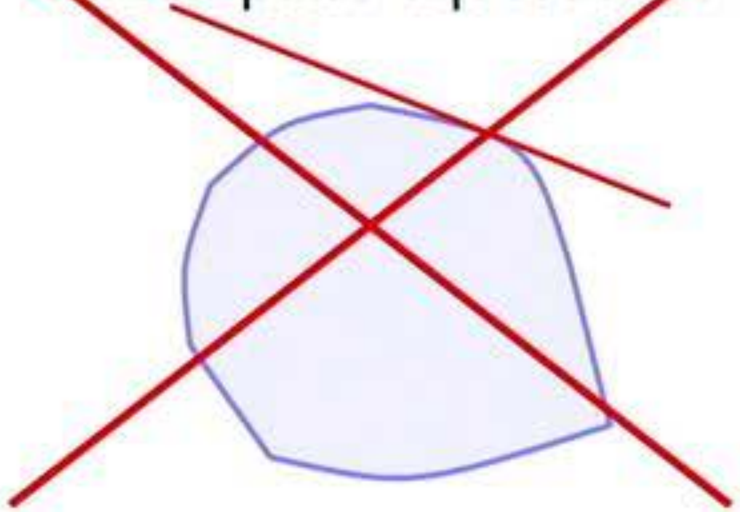


interior-point: approximate solution

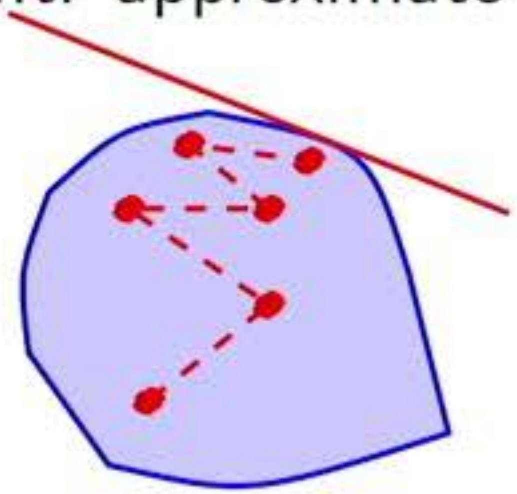


- Semidefinite programming

no simplex equivalent



interior-point: approximate solution



⇒ incompleteness, soundness requires care

SOS: Using approximate SDP solvers

Results from SDP solvers will only satisfy equality constraints up to some ϵ

$$p = z^T Q z + z^T E z, \qquad |E_{i,j}| \leqslant \epsilon.$$

SOS: Using approximate SDP solvers

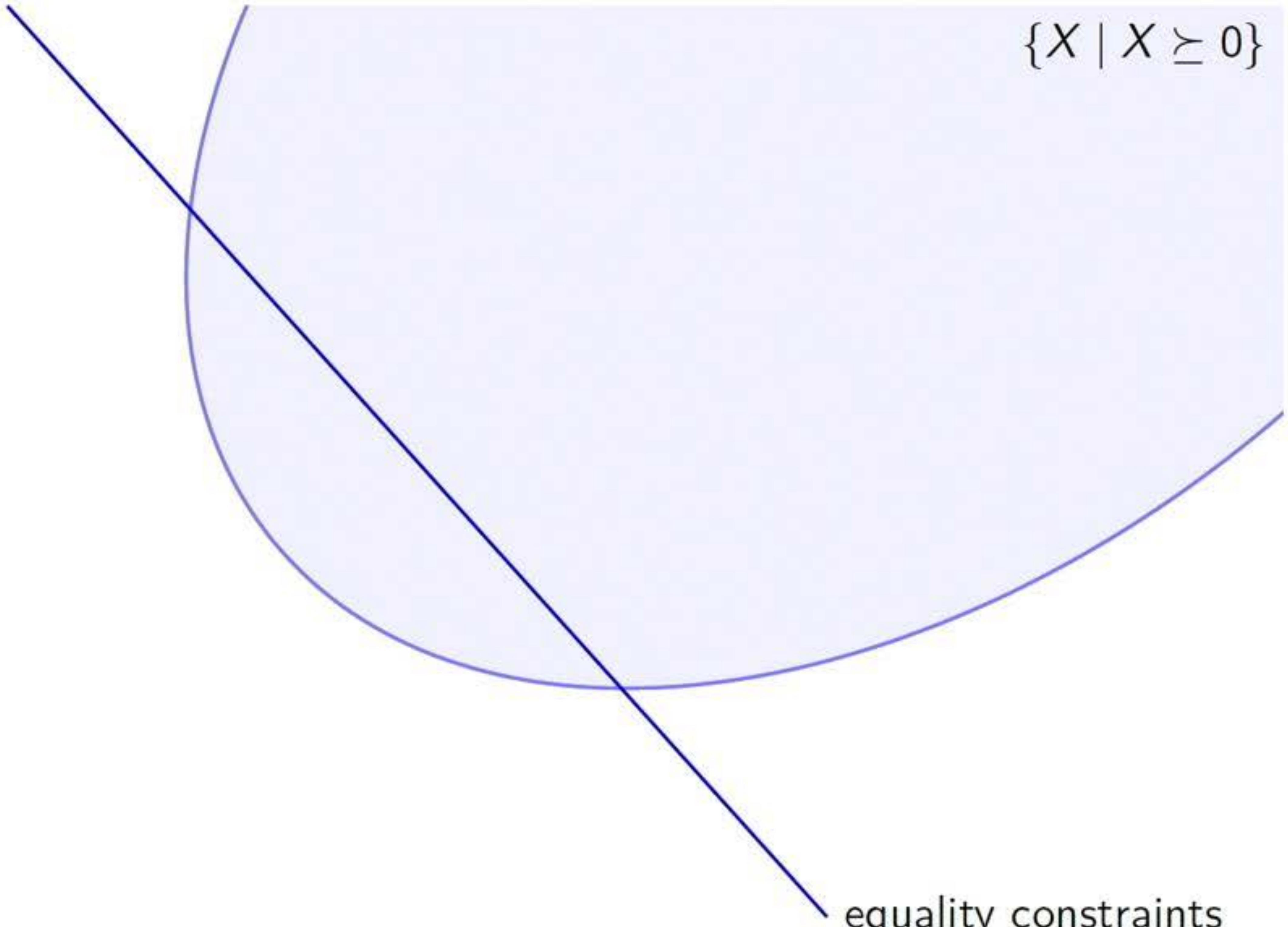
Results from SDP solvers will only satisfy equality constraints up to some ϵ

$$p = z^T Q z + z^T E z, \qquad |E_{i,j}| \leqslant \epsilon.$$

Two validation methods in the literature

- Round Q to an exact solution $\widetilde{Q}$ s.t. $p = z^T \widetilde{Q} z$ and check $\widetilde{Q} \succeq 0$
  - rounding is heuristic
  - check done with rational arithmetic (expensive)
- Check that for any $|E_{i,j}| \leqslant \epsilon$, $Q + E \succeq 0$
  - entirely with floating-point arithmetic (more tricky but fast)

Intuitively, Proving Existence of a Nearby Solution



SOS: Using approximate SDP solvers

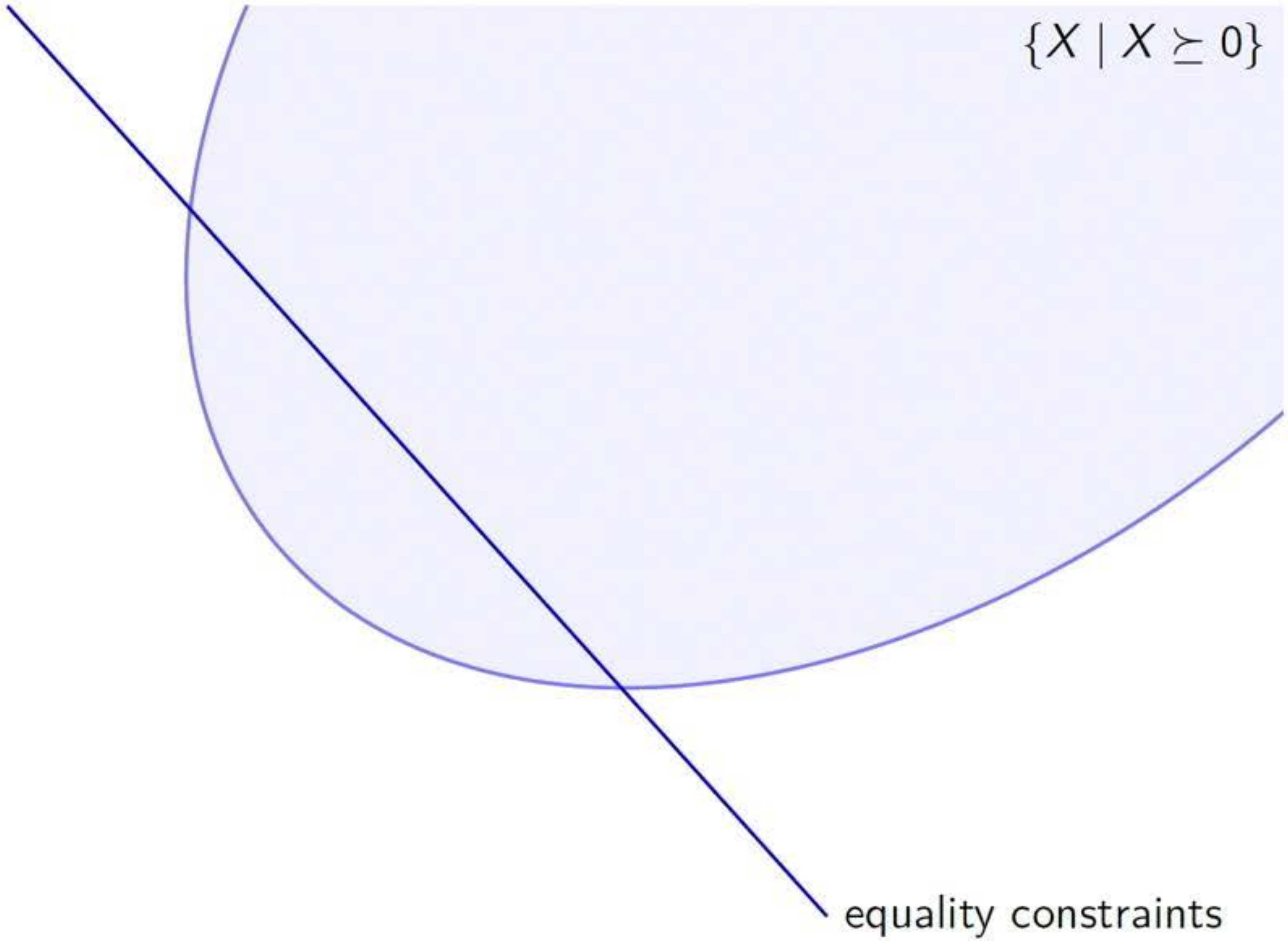
Results from SDP solvers will only satisfy equality constraints up to some ϵ

$$p = z^T Q z + z^T E z, \qquad |E_{i,j}| \leqslant \epsilon.$$

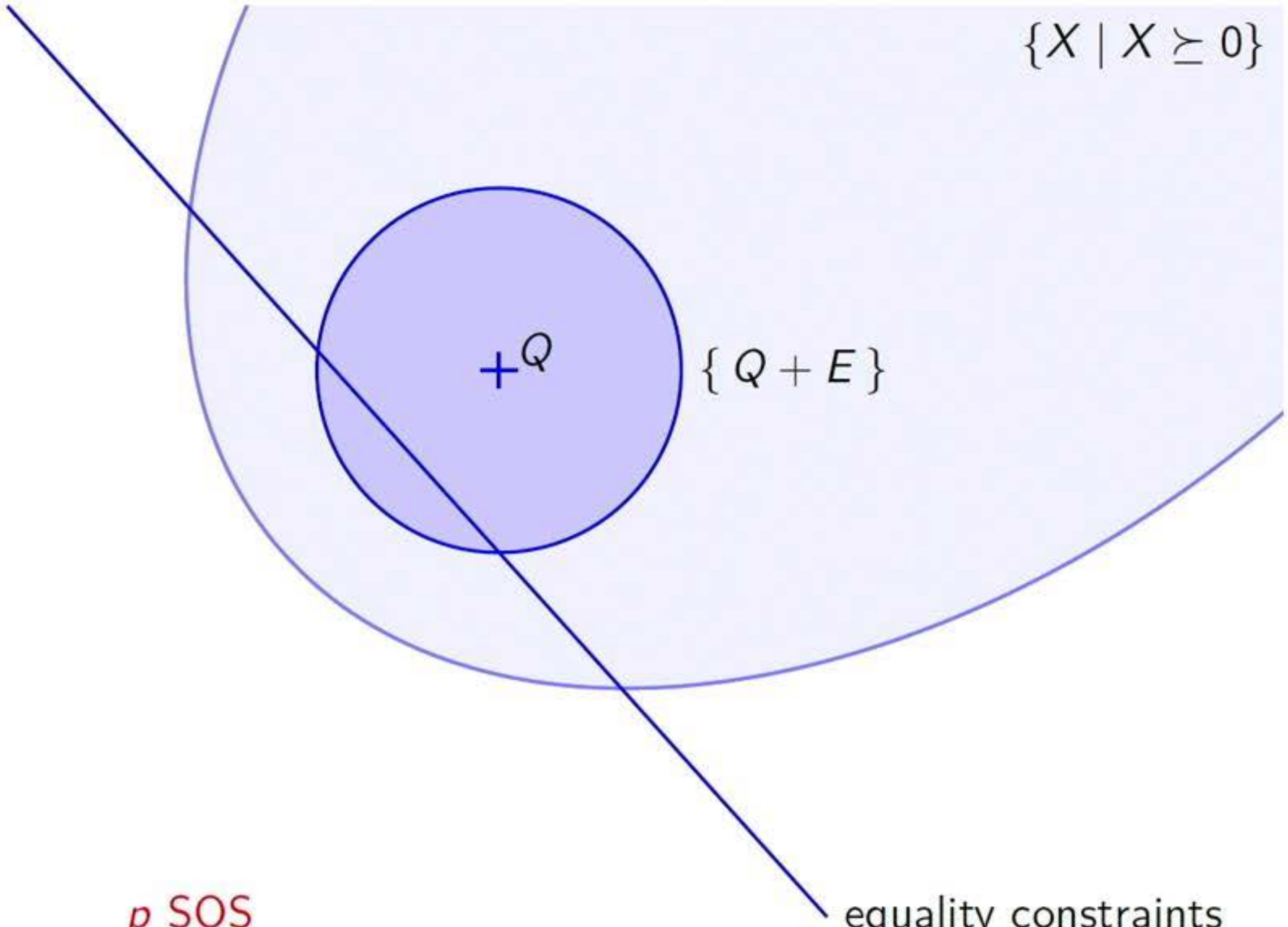
Two validation methods in the literature

- Round Q to an exact solution $\widetilde{Q}$ s.t. $p = z^T \widetilde{Q} z$ and check $\widetilde{Q} \succeq 0$
  - rounding is heuristic
  - check done with rational arithmetic (expensive)
- Check that for any $|E_{i,j}| \leqslant \epsilon$, $Q + E \succeq 0$
  - entirely with floating-point arithmetic (more tricky but fast)

Intuitively, Proving Existence of a Nearby Solution

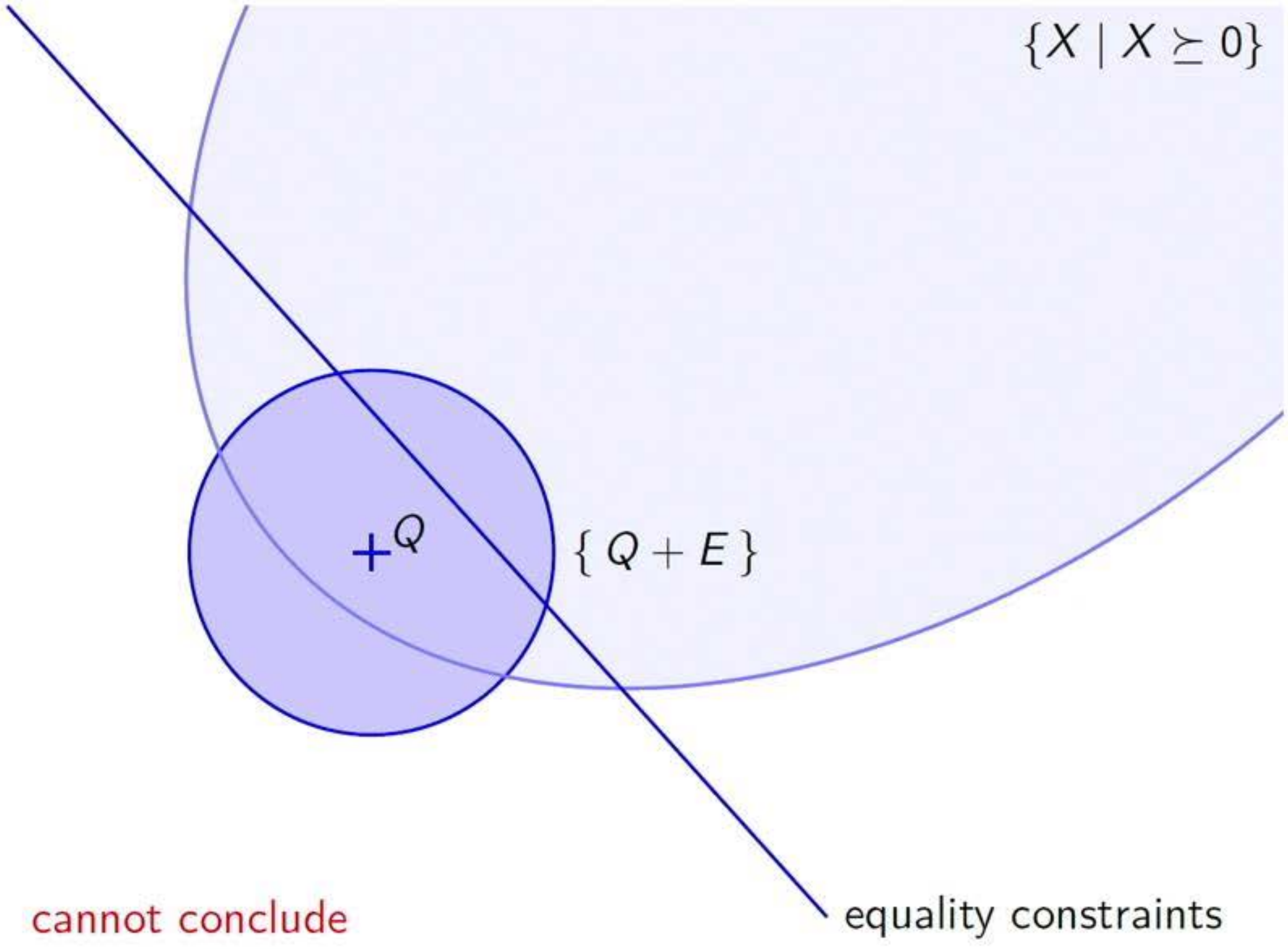


Intuitively, Proving Existence of a Nearby Solution

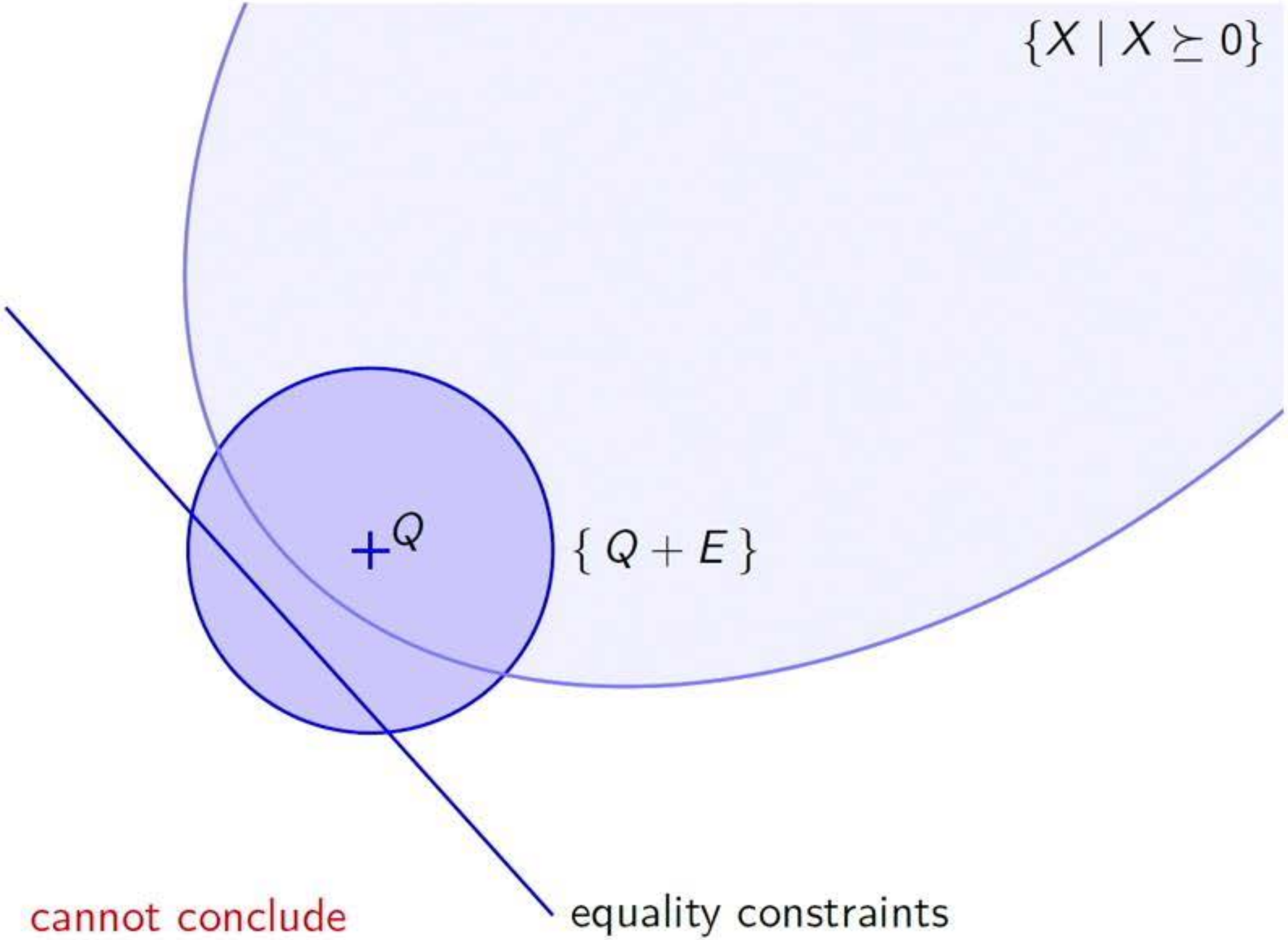


p SOS

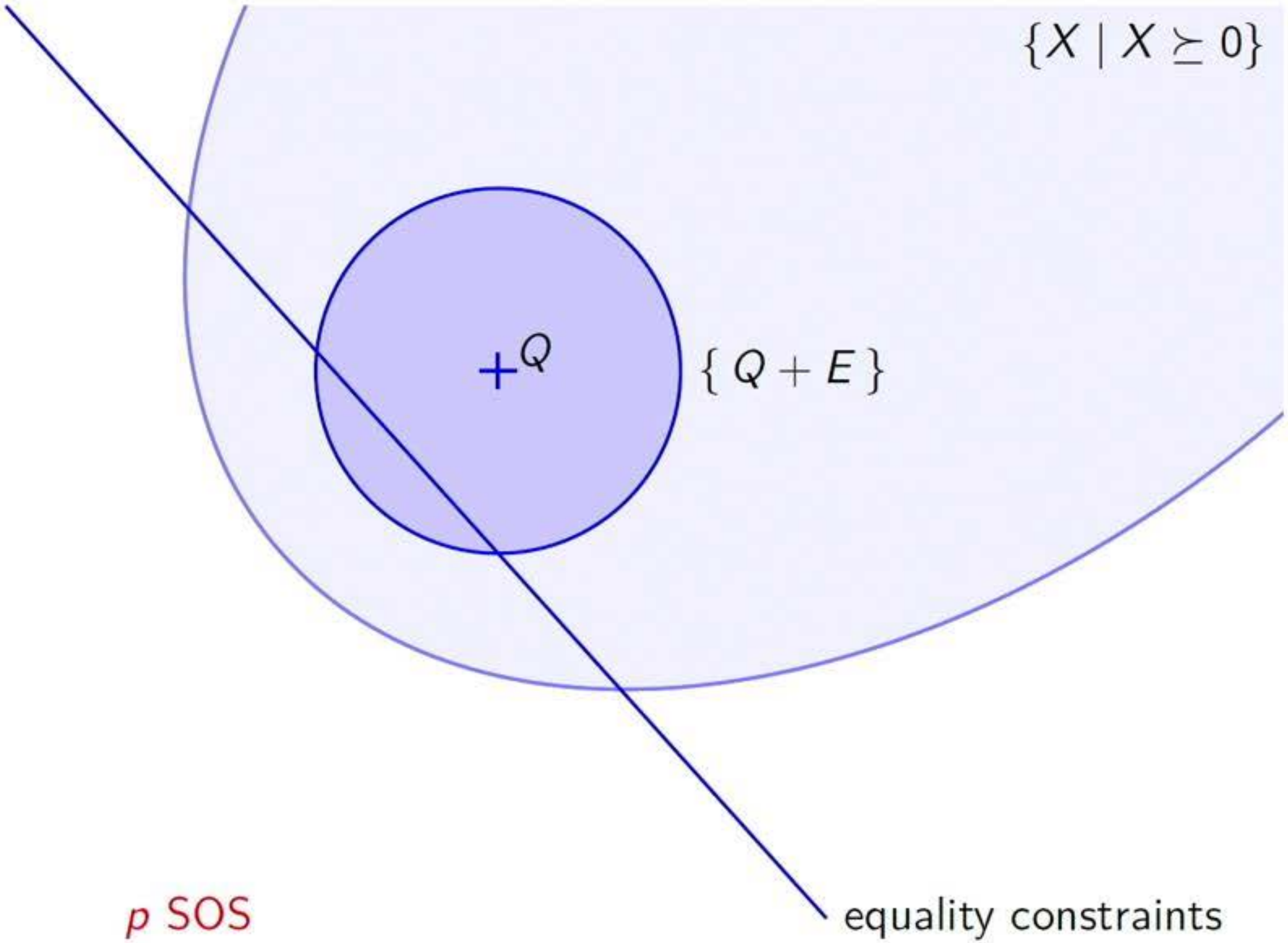
Intuitively, Proving Existence of a Nearby Solution



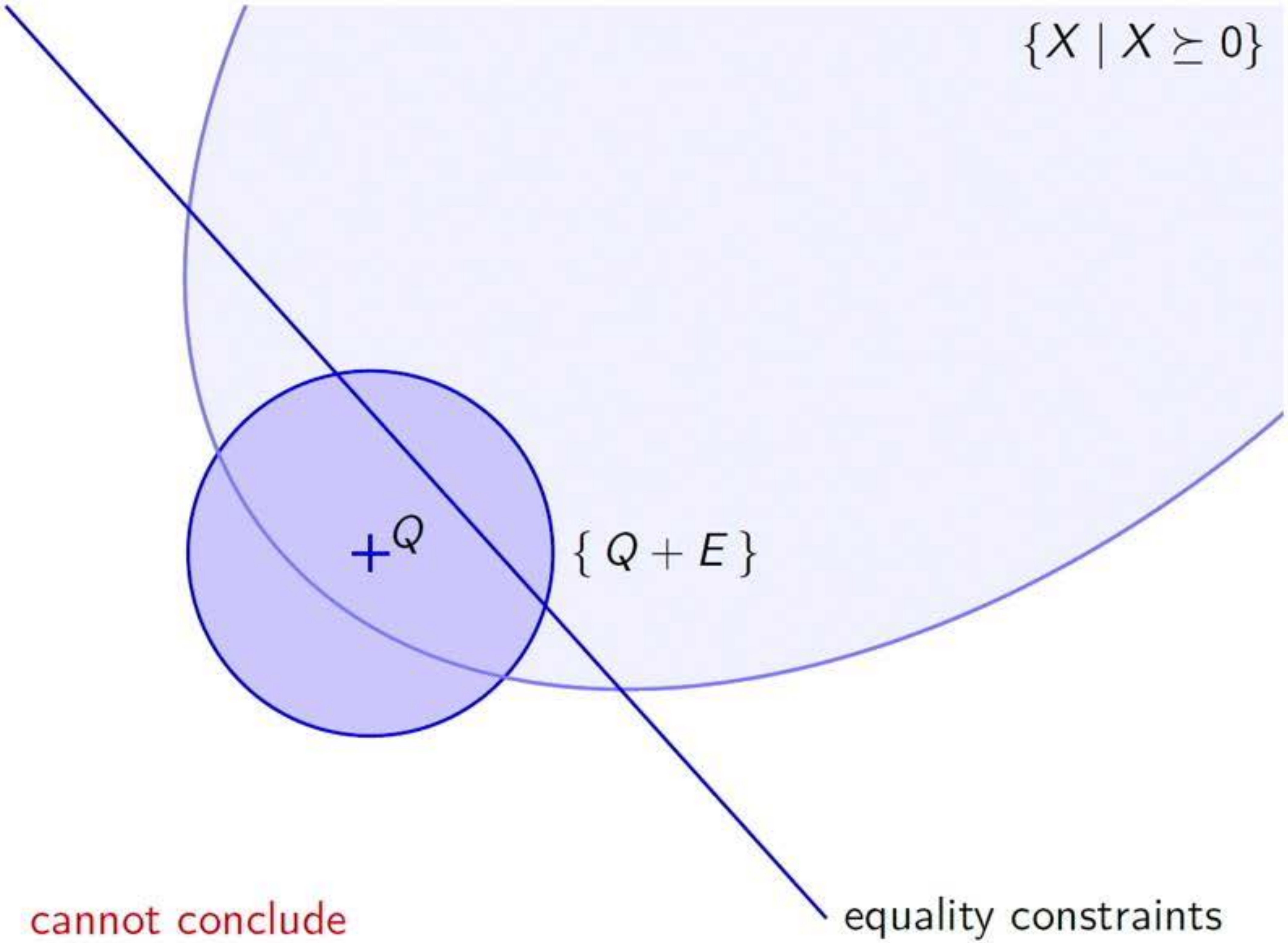
Intuitively, Proving Existence of a Nearby Solution



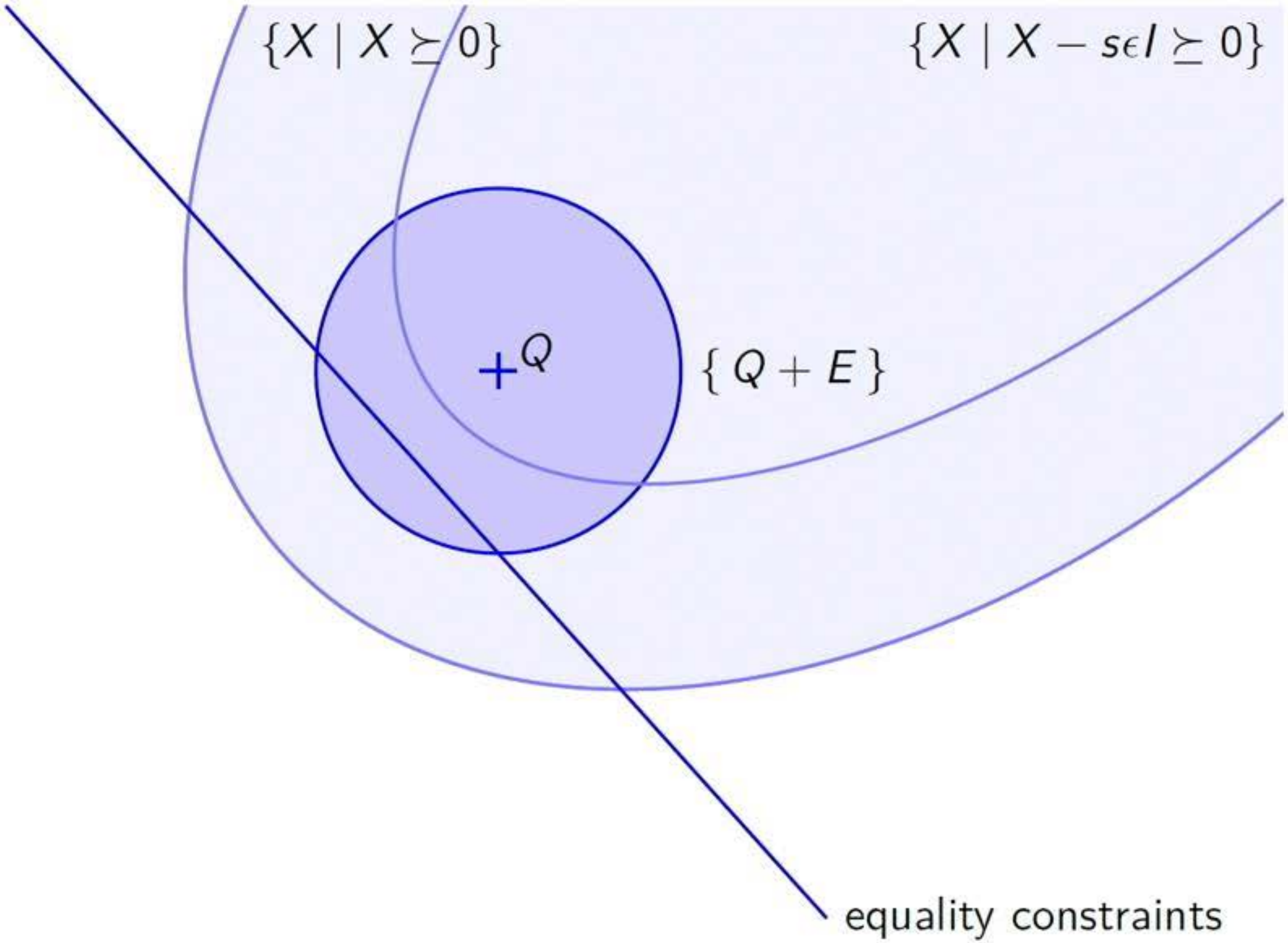
Intuitively, Proving Existence of a Nearby Solution



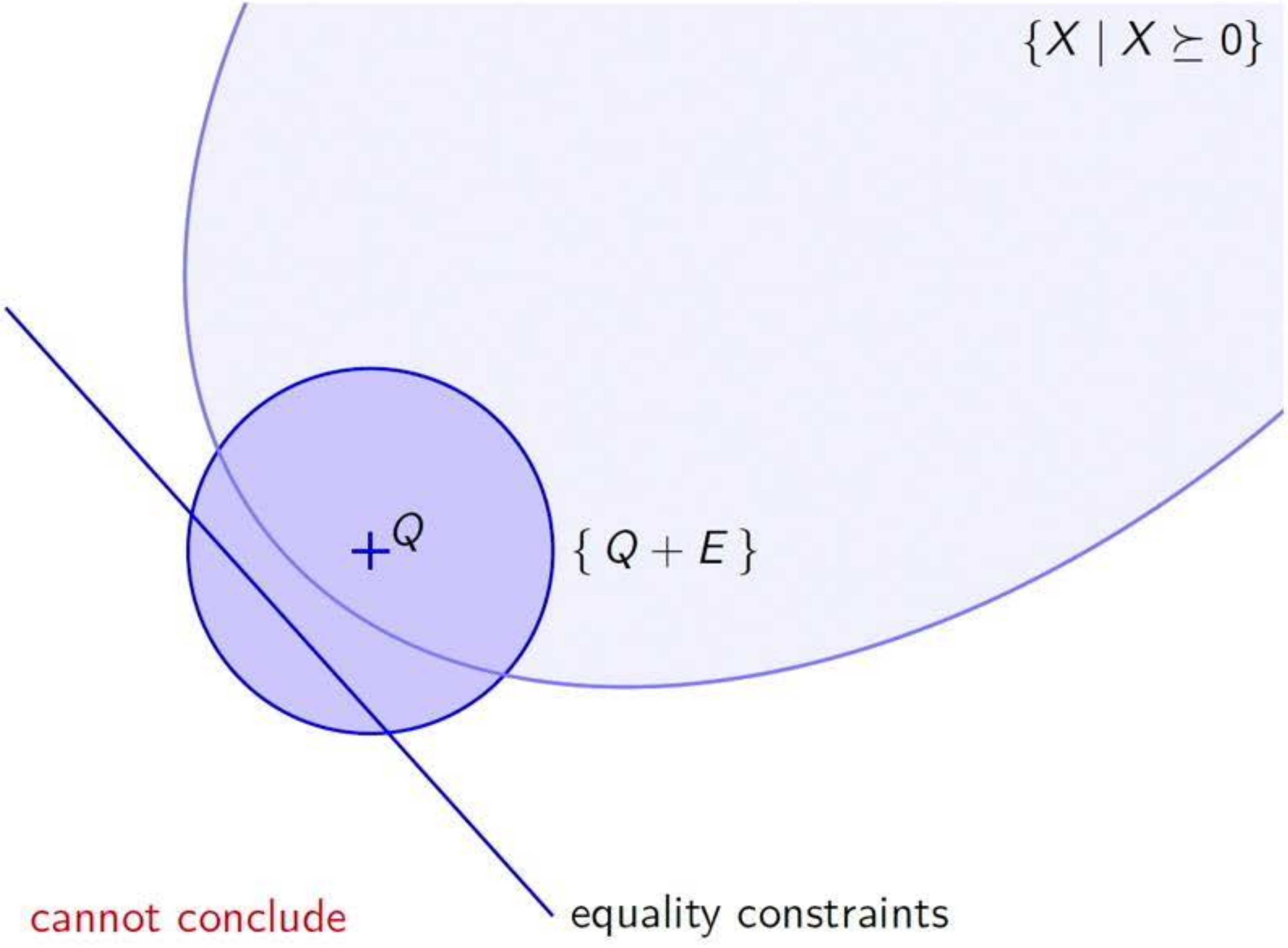
Intuitively, Proving Existence of a Nearby Solution



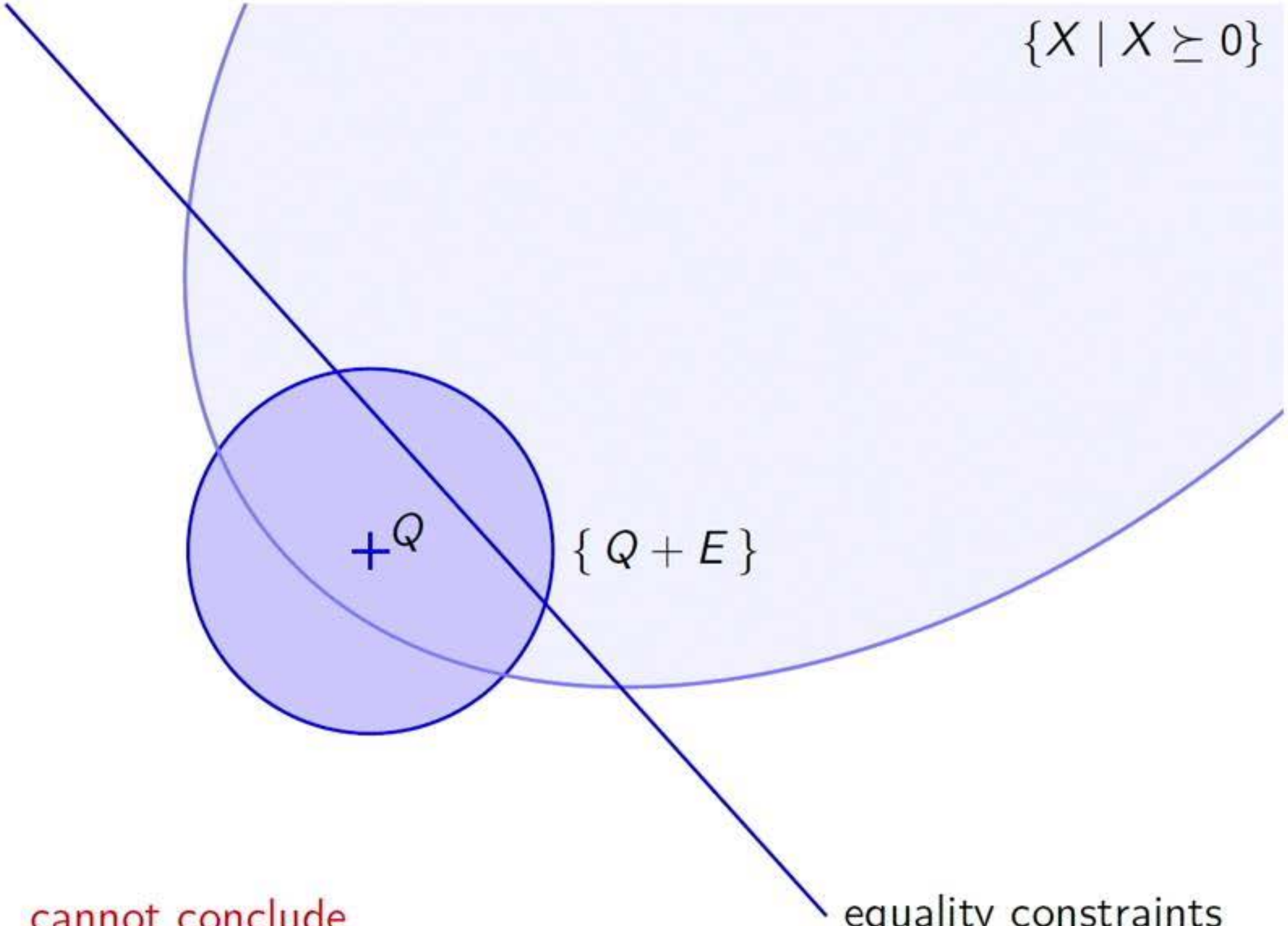
Padding


$\{X \mid X \succeq 0\}$
$\{X \mid X - s\epsilon I \succeq 0\}$
Q
$\{\, Q + E \,\}$
equality constraints

Intuitively, Proving Existence of a Nearby Solution

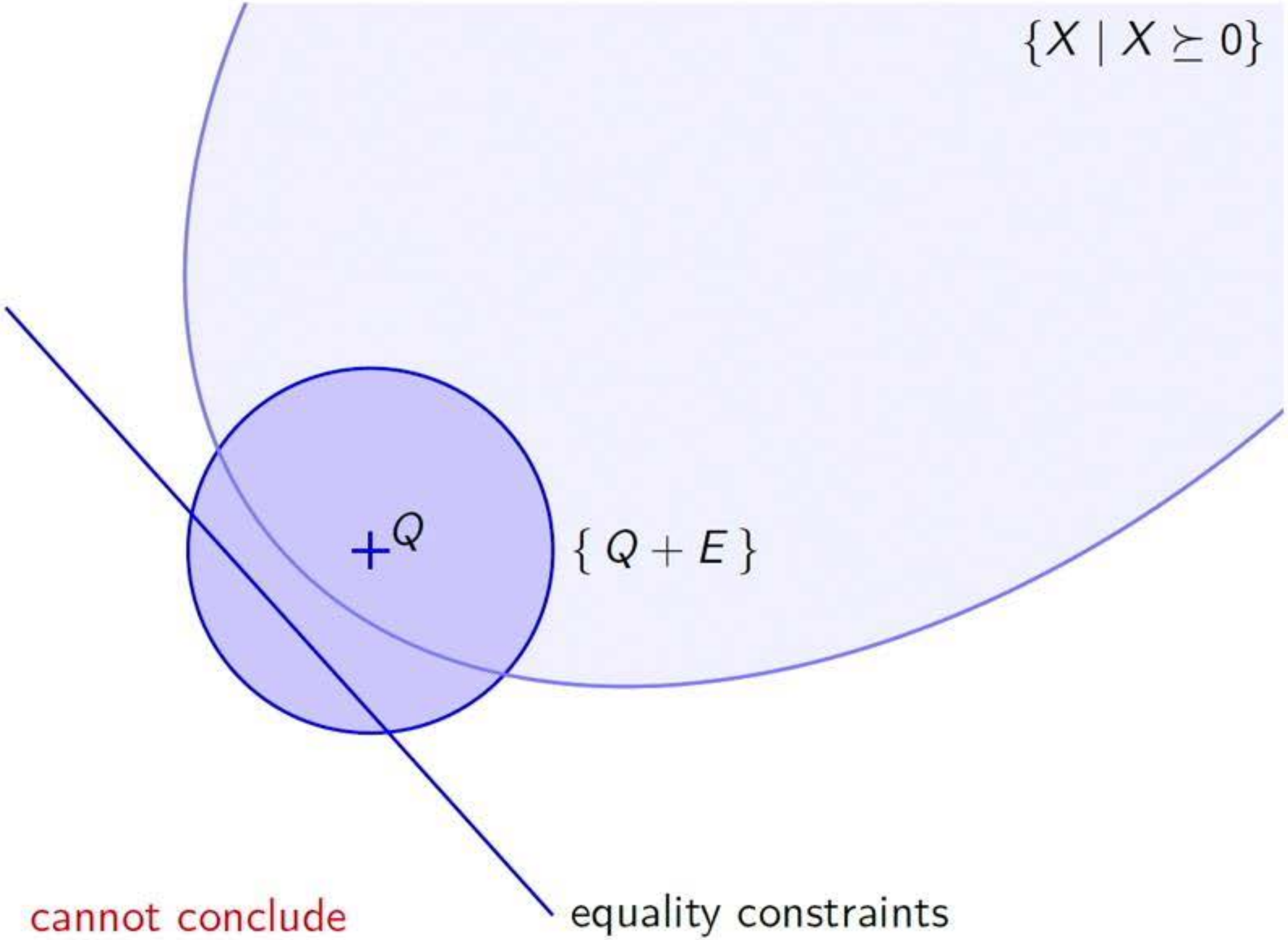


Intuitively, Proving Existence of a Nearby Solution



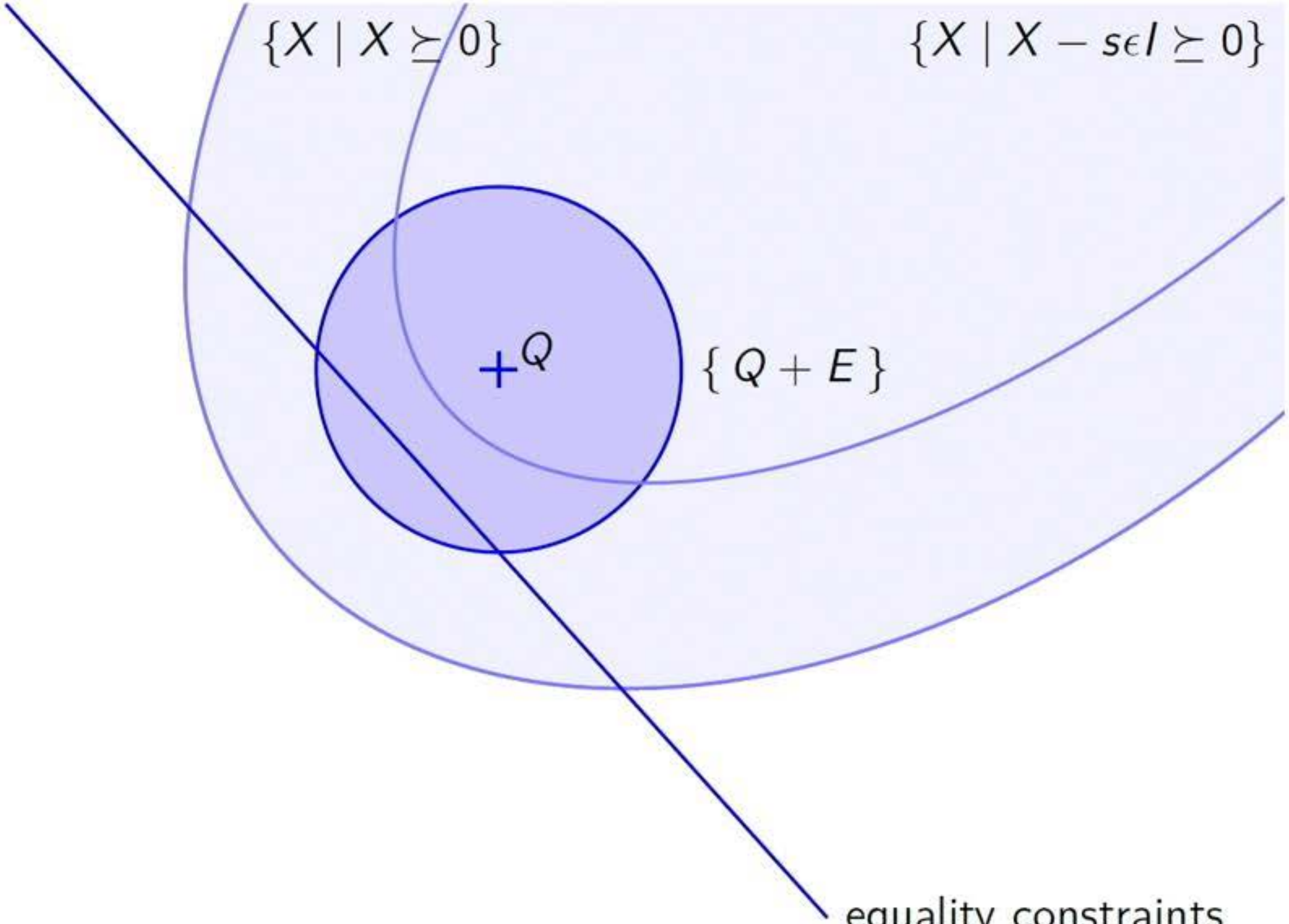
cannot conclude

Intuitively, Proving Existence of a Nearby Solution



cannot conclude

Padding


$\{X \mid X \succeq 0\}$
$\{X \mid X - s\epsilon I \succeq 0\}$
Q
$\{\, Q + E \,\}$
equality constraints

Synthesizing Polynomial Invariants

Ensuring Soundness

Bound Floating-Point Rounding Errors

Integration into a SMT Solver

Formalized Proofs with Coq

Floating-Point Values

Definition

A floating-point format $\mathbb{F}$ is a subset of $\mathbb{R}$. $x \in \mathbb{F}$ when

$$x = m\beta^e$$

for some $m, e \in \mathbb{Z}$, $|m| < \beta^p$ and $e \geqslant e_{min}$.

Floating-Point Values

Definition

A floating-point format $\mathbb{F}$ is a subset of $\mathbb{R}$. $x \in \mathbb{F}$ when

$$x = m\beta^e$$

for some $m, e \in \mathbb{Z}$, $|m| < \beta^p$ and $e \geqslant e_{min}$.

- m: *mantissa* of x
- e: *exponent* of x
- β: *radix* of $\mathbb{F}$
- p: *precision* of $\mathbb{F}$
- e_{min}: minimal exponent of $\mathbb{F}$

Floating-Point Values

Definition

A floating-point format $\mathbb{F}$ is a subset of $\mathbb{R}$. $x \in \mathbb{F}$ when

$$x = m\beta^e$$

for some $m, e \in \mathbb{Z}$, $|m| < \beta^p$ and $e \geqslant e_{min}$.

- m: *mantissa* of x
- e: *exponent* of x
- β: *radix* of $\mathbb{F}$
- p: *precision* of $\mathbb{F}$
- e_{min}: minimal exponent of $\mathbb{F}$

Two kind of numbers

- *normalized*: encoded with p figures ($|m| \geqslant \beta^{p-1}$)
- *denormalized*: tiny values ($e = e_{min}$, $|m| < \beta^{p-1}$)

Standard Model of Floating-Point Arithmetic

Definition

$\mathrm{fl}(e)$: floating-point evaluation of expression e (from left to right).

For $\diamond \in \left\{+, -, \sqrt{}\right\}$:

$$\exists\delta, |\delta| \leqslant \frac{\varepsilon}{1+\varepsilon} \wedge \qquad \mathrm{fl}(x \diamond y) = (1+\delta)(x \diamond y).$$

Standard Model of Floating-Point Arithmetic

Definition

$\mathrm{fl}(e)$: floating-point evaluation of expression e (from left to right).

For $\diamond \in \left\{+, -, \sqrt{}\right\}$:

$$\exists \delta, |\delta| \leqslant \frac{\varepsilon}{1+\varepsilon} \wedge \qquad \mathrm{fl}(x \diamond y) = (1+\delta)(x \diamond y).$$

For $\diamond \in \{\times, \backslash\}$:

$$\exists \delta, \omega, |\delta| \leqslant \frac{\varepsilon}{1+\varepsilon} \wedge |\omega| \leqslant \eta \wedge \qquad \mathrm{fl}(x \diamond y) = (1+\delta)(x \diamond y) + \omega.$$

Standard Model of Floating-Point Arithmetic

Definition

$\mathrm{fl}(e)$: floating-point evaluation of expression e (from left to right).

For $\diamond \in \left\{+, -, \sqrt{}\right\}$:

$$\exists \delta, |\delta| \leqslant \frac{\varepsilon}{1+\varepsilon} \wedge \qquad \mathrm{fl}(x \diamond y) = (1+\delta)(x \diamond y).$$

For $\diamond \in \{\times, \backslash\}$:

$$\exists \delta, \omega, |\delta| \leqslant \frac{\varepsilon}{1+\varepsilon} \wedge |\omega| \leqslant \eta \wedge \qquad \mathrm{fl}(x \diamond y) = (1+\delta)(x \diamond y) + \omega.$$

Example

$\varepsilon = 2^{-53}$ ($\simeq 10^{-16}$) and $\eta = 2^{-1075}$ ($\simeq 10^{-323}$)
for binary64 format (`double` in C) and rounding to nearest.

Example: Summation

Bounds can be combined:

Theorem

For all $x \in \mathbb{R}^n$

$$\left| \mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} x_i \right| \leqslant n\,\varepsilon \sum_{i=1}^{n} |x_i| + (1 + n\,\varepsilon) n\,\eta$$

Proved in Coq (`https://github.com/validsdp/validsdp/`).

Floating-Point arithmetic model from the Flocq library (`http://flocq.gforge.inria.fr/`).

Cholesky Decomposition

- To prove that $a \in \mathbb{R}$ is non negative,
  we can exhibit r such that $a = r^2$ (typically $r = \sqrt{a}$).
- To prove that a matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite
  we can similarly expose R such that $A = R^T R$
  (since $x^T \left(R^T R\right) x = (Rx)^T (Rx) = \|Rx\|_2^2 \geqslant 0$).
- The Cholesky decomposition computes such a matrix R:

R **:=** 0;
for j **from** 1 **to** n **do**
  for i **from** 1 **to** $j-1$ **do**

$$R_{i,j} := \left(A_{i,j} - \sum_{k=1}^{i-1} R_{k,i} R_{k,j}\right) / R_{i,i};$$

  od

$$R_{j,j} := \sqrt{M_{j,j} - \sum_{k=1}^{j-1} {R_{k,j}}^2};$$

od

- If it succeeds (no $\sqrt{}$ of negative or div. by 0) then $A \succeq 0$.

Cholesky Decomposition (end)

With rounding errors $A \neq R^T R$, Cholesky can succeed while $A \nsucceq 0$.

But error is bounded and for some (tiny) $c \in \mathbb{R}$:
if Cholesky succeeds on A then $A + c\,I \succeq 0$.

Hence:

Theorem

If Cholesky succeeds on $A - c\,I$ then $A \succeq 0$

holds for any $c \geqslant \frac{(s+1)\varepsilon}{1-(s+1)\varepsilon}\mathrm{tr}(A) + 4\,s\left(2(s+1) + \max_i(A_{i,i})\right)\eta$

Proved in Coq (paper proof: 6 pages, Coq: 5.1 kloc)

Synthesizing Polynomial Invariants

Ensuring Soundness

Bound Floating-Point Rounding Errors

Integration into a SMT Solver

Formalized Proofs with Coq

Cholesky Decomposition (end)

With rounding errors $A \neq R^T R$, Cholesky can succeed while $A \not\succeq 0$.

But error is bounded and for some (tiny) $c \in \mathbb{R}$:
if Cholesky succeeds on A then $A + c\,I \succeq 0$.

Hence:

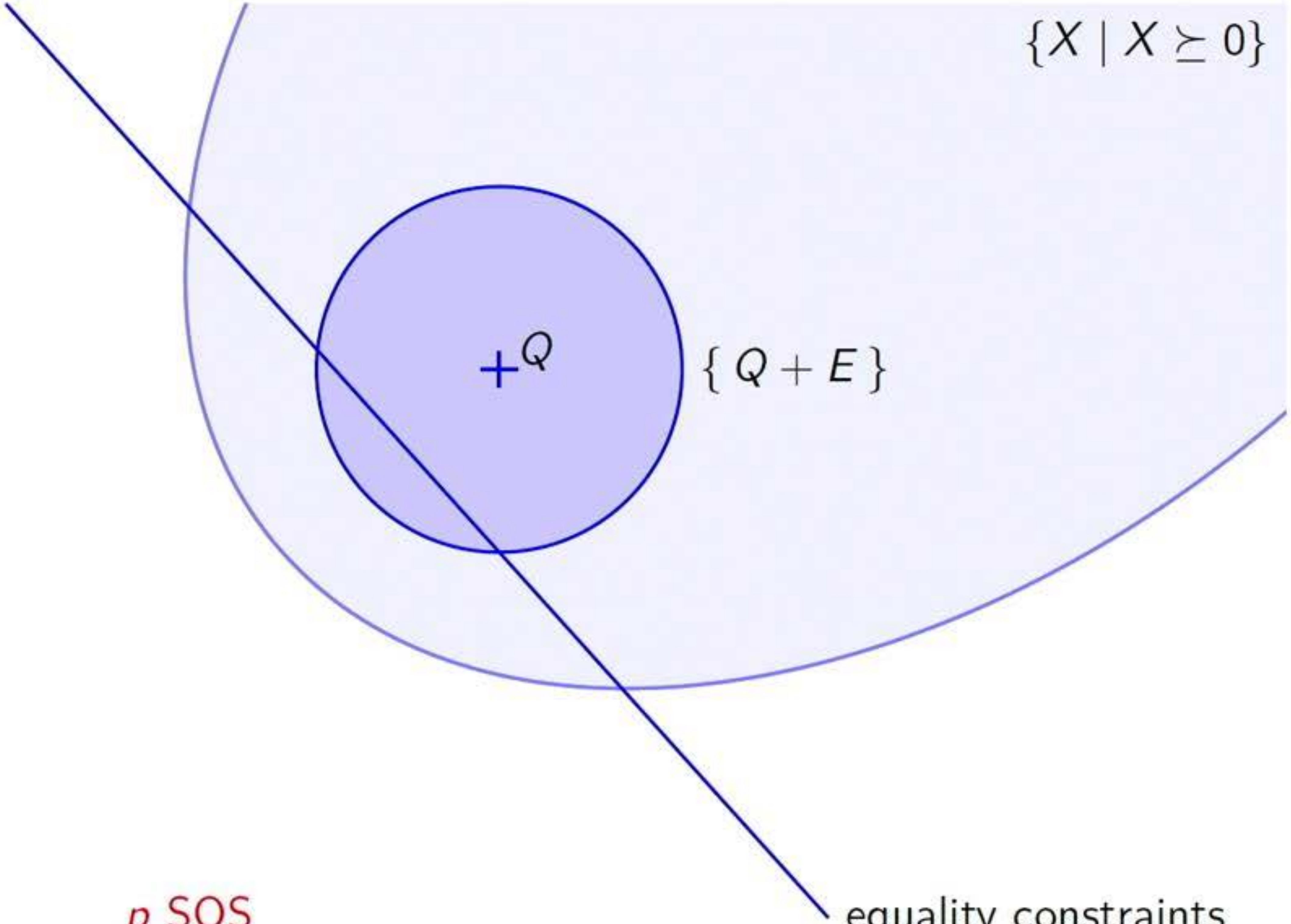
Theorem

If Cholesky succeeds on $A - c\,I$ then $A \succeq 0$

holds for any $c \geqslant \frac{(s+1)\varepsilon}{1-(s+1)\varepsilon}\mathrm{tr}(A) + 4\,s\left(2(s+1) + \max_i(A_{i,i})\right)\eta$

Proved in Coq (paper proof: 6 pages, Coq: 5.1 kloc)

Intuitively, Proving Existence of a Nearby Solution



p SOS

Cholesky Decomposition (end)

With rounding errors $A \neq R^T R$, Cholesky can succeed while $A \not\succeq 0$.

But error is bounded and for some (tiny) $c \in \mathbb{R}$:
if Cholesky succeeds on A then $A + c\, I \succeq 0$.

Hence:

Theorem

If Cholesky succeeds on $A - c\, I$ then $A \succeq 0$

holds for any $c \geqslant \frac{(s+1)\varepsilon}{1-(s+1)\varepsilon}\operatorname{tr}(A) + 4\, s \left(2(s+1) + \max_i(A_{i,i})\right) \eta$

Proved in Coq (paper proof: 6 pages, Coq: 5.1 kloc)

Synthesizing Polynomial Invariants

Ensuring Soundness

Bound Floating-Point Rounding Errors

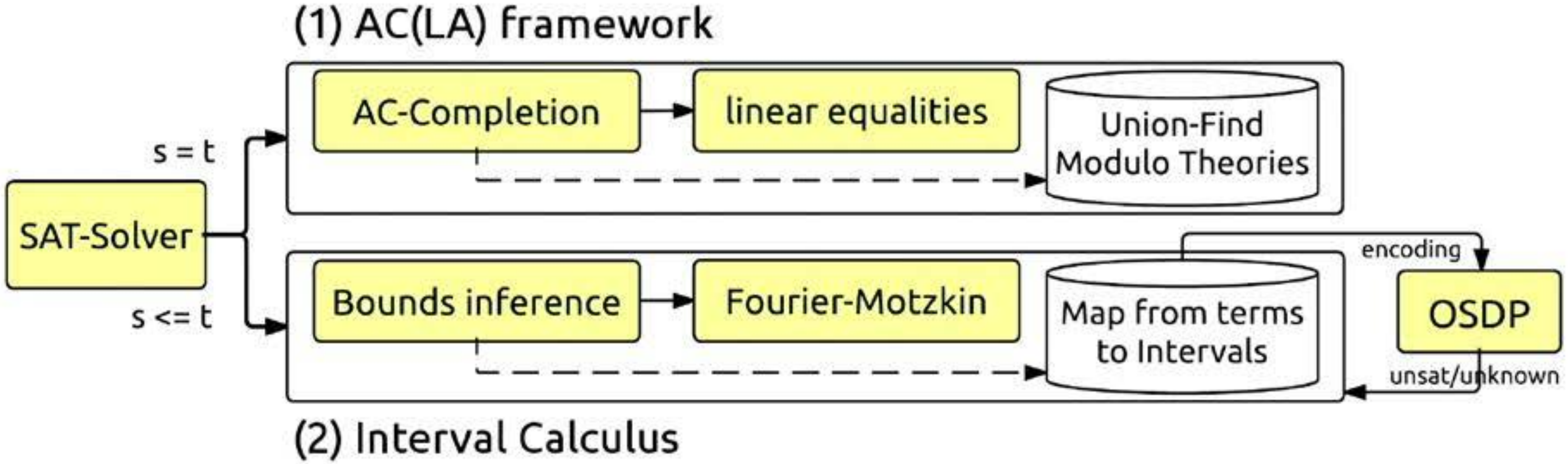
Integration into a SMT Solver

Formalized Proofs with Coq

Integration in Alt-Ergo

Joint work with Mohamed Iguernlala and Sylvain Conchon

- Integrated into Alt-Ergo 2



- Unfortunately no tight collaboration:
  - one shot, no incrementality
  - mostly a boolean result
- available at
  `https://github.com/OCamlPro/alt-ergo/pull/124`

Experimental Results (1/3)

Benchmarks QF_NIA from SMT-LIB.

	AE		AESDP		AESDPap		AESDPex	
	unsat	time	unsat	time	unsat	time	unsat	time
AProVE (746)	103	7387	319	23968	359	7664	318	22701
calypto (97)	92	357	88	679	88	489	89	816
LassoRanker (102)	57	9	62	959	64	274	63	878
LCTES (2)	0	0	0	0	0	0	0	0
leipzig (5)	0	0	0	0	0	0	0	0
mcm (161)	0	0	0	0	0	0	0	0
UltimateAutom (7)	1	0.35	7	0.73	7	0.62	7	0.69
UltimateLasso (26)	26	118	26	212	26	126	26	215
total (1146)	279	7872	502	25818	544	8553	503	24611

	CVC4		Smtrat		Yices2		Z3	
	unsat	time	unsat	time	unsat	time	unsat	time
AProVE (746)	586	10821	185	3879	**709**	**1982**	252	5156
calypto (97)	87	7	89	754	**97**	**409**	95	613
LassoRanker (102)	72	27	20	12	**84**	**595**	84	2538
LCTES (2)	**1**	**0**	0	0	0	0	0	0
leipzig (5)	0	0	0	0	**1**	**0**	0	0
mcm (161)	**4**	**2489**	0	0	0	0	4	2527
UltimateAutom (7)	6	0.03	1	7.22	**7**	**0.04**	7	0.31
UltimateLasso (26)	4	66	26	177	**26**	**6**	26	21
total (1146)	780	13411	321	4829	**924**	**2993**	468	10855

On Intel Xeon 2.3 GHz, time limits 900 s and memory limits 2 GB.

Experimental Results (2/3)

Benchmarks QF_NRA from SMT-LIB.

	AE		AESDP		AESDPap		AESDPex	
	unsat	time	unsat	time	unsat	time	unsat	time
Sturm-MBO (300)	155	12950	155	13075	155	13053	155	12973
hong (20)	1	0	20	28	20	24	20	27
hycomp (2494)	1285	15351	1266	15857	1271	16080	1265	14909
keymaera (320)	261	36	291	356	278	97	291	360
LassoRanker (627)	0	0	0	0	0	0	0	0
meti-tarski (2615)	1882	10	2273	91	2267	65	2241	73
UltimateAutom (13)	0	0	0	0	0	0	0	0
zankl (85)	14	1.00	24	15.46	24	16.09	24	15.67
total (6549)	3571	28348	4029	29423	4015	29334	3996	28357

	CVC4		Smtrat		Yices2		Z3	
	unsat	time	unsat	time	unsat	time	unsat	time
Sturm-MBO (300)	285	1403	**285**	**620**	2	0	47	21
hong (20)	20	1	**20**	**0**	8	240	9	6
hycomp (2494)	2184	208	1588	13784	2182	1241	2201	4498
keymaera (320)	249	4	307	13	270	359	**318**	**2**
LassoRanker (627)	**441**	**32786**	0	0	236	30835	119	1733
meti-tarski (2615)	1643	804	2520	3345	2578	2027	**2611**	**337**
UltimateAutom (13)	5	0.52	0	0	12	57.19	**13**	**19.23**
zankl (85)	24	9.40	19	13.47	**32**	**7.22**	27	0.43
total (6549)	4853	35239	4740	17775	5331	36849	**5355**	**6658**

On Intel Xeon 2.3 GHz, time limits 900 s and memory limits 2 GB.

Experimental Results (1/3)

Benchmarks QF_NIA from SMT-LIB.

	AE		AESDP		AESDPap		AESDPex	
	unsat	time	unsat	time	unsat	time	unsat	time
AProVE (746)	103	7387	319	23968	359	7664	318	22701
calypto (97)	92	357	88	679	88	489	89	816
LassoRanker (102)	57	9	62	959	64	274	63	878
LCTES (2)	0	0	0	0	0	0	0	0
leipzig (5)	0	0	0	0	0	0	0	0
mcm (161)	0	0	0	0	0	0	0	0
UltimateAutom (7)	1	0.35	7	0.73	7	0.62	7	0.69
UltimateLasso (26)	26	118	26	212	26	126	26	215
total (1146)	279	7872	502	25818	544	8553	503	24611

	CVC4		Smtrat		Yices2		Z3	
	unsat	time	unsat	time	unsat	time	unsat	time
AProVE (746)	586	10821	185	3879	**709**	**1982**	252	5156
calypto (97)	87	7	89	754	**97**	**409**	95	613
LassoRanker (102)	72	27	20	12	**84**	**595**	84	2538
LCTES (2)	**1**	**0**	0	0	0	0	0	0
leipzig (5)	0	0	0	0	**1**	**0**	0	0
mcm (161)	**4**	**2489**	0	0	0	0	4	2527
UltimateAutom (7)	6	0.03	1	7.22	**7**	**0.04**	7	0.31
UltimateLasso (26)	4	66	26	177	**26**	**6**	26	21
total (1146)	780	13411	321	4829	**924**	**2993**	468	10855

On Intel Xeon 2.3 GHz, time limits 900 s and memory limits 2 GB.

Experimental Results (3/3)

More numerical benchmarks (incl. control-command programs).

	AE		AESDP		AESDPap		AESDPex	
	unsat	time	unsat	time	unsat	time	unsat	time
C (67)	11	0.05	**63**	**39.78**	63	40.01	13	1.18
quadratic (67)	13	0.06	**67**	**14.68**	67	15.44	15	0.08
flyspeck (20)	1	0.00	**19**	**26.35**	19	26.62	3	0.01
global-opt (14)	2	0.01	**14**	**8.72**	14	8.83	5	0.20
total (168)	27	0.12	**163**	**89.53**	163	90.90	36	1.47

	CVC4		Smtrat		Yices2		Z3	
	unsat	time	unsat	time	unsat	time	unsat	time
C (67)	0	0	0	0	0	0	0	0
quadratic (67)	14	2.46	18	1.26	25	357.20	25	257.39
flyspeck (20)	6	695.59	9	36.54	10	0.05	9	0.05
global-opt (14)	5	0.12	12	41.18	12	0.16	13	683.45
total (168)	25	698.17	39	78.98	47	357.41	47	940.89

On Intel Xeon 2.3 GHz, time limits 900 s and memory limits 2 GB.
All times are in seconds.

Experimental Results (2/3)

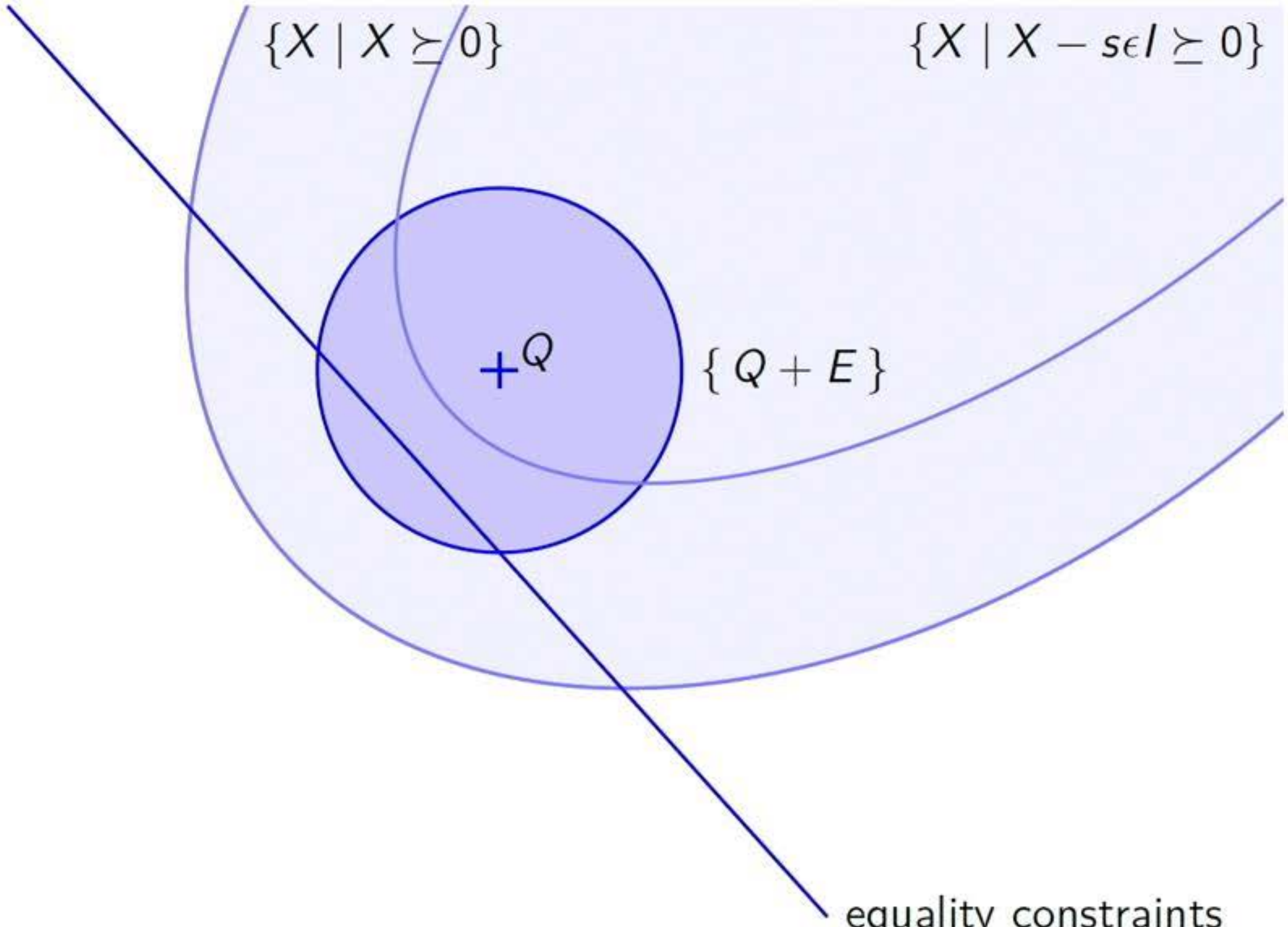
Benchmarks QF_NRA from SMT-LIB.

	AE		AESDP		AESDPap		AESDPex	
	unsat	time	unsat	time	unsat	time	unsat	time
Sturm-MBO (300)	155	12950	155	13075	155	13053	155	12973
hong (20)	1	0	20	28	20	24	20	27
hycomp (2494)	1285	15351	1266	15857	1271	16080	1265	14909
keymaera (320)	261	36	291	356	278	97	291	360
LassoRanker (627)	0	0	0	0	0	0	0	0
meti-tarski (2615)	1882	10	2273	91	2267	65	2241	73
UltimateAutom (13)	0	0	0	0	0	0	0	0
zankl (85)	14	1.00	24	15.46	24	16.09	24	15.67
total (6549)	3571	28348	4029	29423	4015	29334	3996	28357

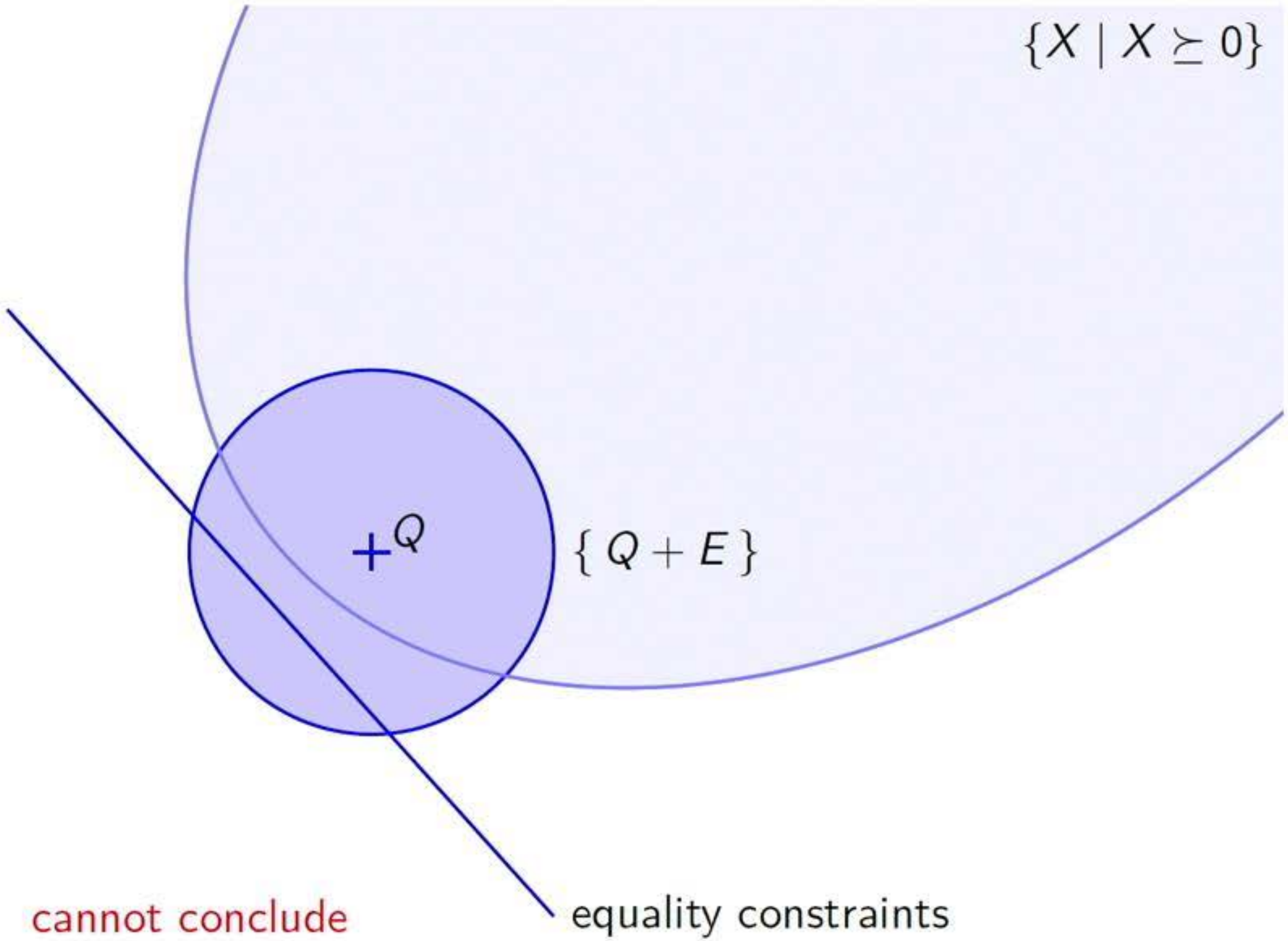
	CVC4		Smtrat		Yices2		Z3	
	unsat	time	unsat	time	unsat	time	unsat	time
Sturm-MBO (300)	285	1403	**285**	**620**	2	0	47	21
hong (20)	20	1	**20**	**0**	8	240	9	6
hycomp (2494)	2184	208	1588	13784	2182	1241	2201	4498
keymaera (320)	249	4	307	13	270	359	**318**	**2**
LassoRanker (627)	**441**	**32786**	0	0	236	30835	119	1733
meti-tarski (2615)	1643	804	2520	3345	2578	2027	**2611**	**337**
UltimateAutom (13)	5	0.52	0	0	12	57.19	**13**	**19.23**
zankl (85)	24	9.40	19	13.47	**32**	**7.22**	27	0.43
total (6549)	4853	35239	4740	17775	5331	36849	**5355**	**6658**

On Intel Xeon 2.3 GHz, time limits 900 s and memory limits 2 GB.

Padding


$\{X \mid X \succeq 0\}$
$\{X \mid X - s\epsilon I \succeq 0\}$
Q
$\{\, Q + E \,\}$
equality constraints

Intuitively, Proving Existence of a Nearby Solution



Experimental Results (1/3)

Benchmarks QF_NIA from SMT-LIB.

	AE		AESDP		AESDPap		AESDPex	
	unsat	time	unsat	time	unsat	time	unsat	time
AProVE (746)	103	7387	319	23968	359	7664	318	22701
calypto (97)	92	357	88	679	88	489	89	816
LassoRanker (102)	57	9	62	959	64	274	63	878
LCTES (2)	0	0	0	0	0	0	0	0
leipzig (5)	0	0	0	0	0	0	0	0
mcm (161)	0	0	0	0	0	0	0	0
UltimateAutom (7)	1	0.35	7	0.73	7	0.62	7	0.69
UltimateLasso (26)	26	118	26	212	26	126	26	215
total (1146)	279	7872	502	25818	544	8553	503	24611

	CVC4		Smtrat		Yices2		Z3	
	unsat	time	unsat	time	unsat	time	unsat	time
AProVE (746)	586	10821	185	3879	**709**	**1982**	252	5156
calypto (97)	87	7	89	754	**97**	**409**	95	613
LassoRanker (102)	72	27	20	12	**84**	**595**	84	2538
LCTES (2)	**1**	**0**	0	0	0	0	0	0
leipzig (5)	0	0	0	0	**1**	**0**	0	0
mcm (161)	**4**	**2489**	0	0	0	0	4	2527
UltimateAutom (7)	6	0.03	1	7.22	**7**	**0.04**	7	0.31
UltimateLasso (26)	4	66	26	177	**26**	**6**	26	21
total (1146)	780	13411	321	4829	**924**	**2993**	468	10855

On Intel Xeon 2.3 GHz, time limits 900 s and memory limits 2 GB.

Experimental Results (3/3)

More numerical benchmarks (incl. control-command programs).

	AE		AESDP		AESDPap		AESDPex	
	unsat	time	unsat	time	unsat	time	unsat	time
C (67)	11	0.05	**63**	**39.78**	63	40.01	13	1.18
quadratic (67)	13	0.06	**67**	**14.68**	67	15.44	15	0.08
flyspeck (20)	1	0.00	**19**	**26.35**	19	26.62	3	0.01
global-opt (14)	2	0.01	**14**	**8.72**	14	8.83	5	0.20
total (168)	27	0.12	**163**	**89.53**	163	90.90	36	1.47

	CVC4		Smtrat		Yices2		Z3	
	unsat	time	unsat	time	unsat	time	unsat	time
C (67)	0	0	0	0	0	0	0	0
quadratic (67)	14	2.46	18	1.26	25	357.20	25	257.39
flyspeck (20)	6	695.59	9	36.54	10	0.05	9	0.05
global-opt (14)	5	0.12	12	41.18	12	0.16	13	683.45
total (168)	25	698.17	39	78.98	47	357.41	47	940.89

On Intel Xeon 2.3 GHz, time limits 900 s and memory limits 2 GB.
All times are in seconds.

Synthesizing Polynomial Invariants

Ensuring Soundness

Bound Floating-Point Rounding Errors

Integration into a SMT Solver

Formalized Proofs with Coq

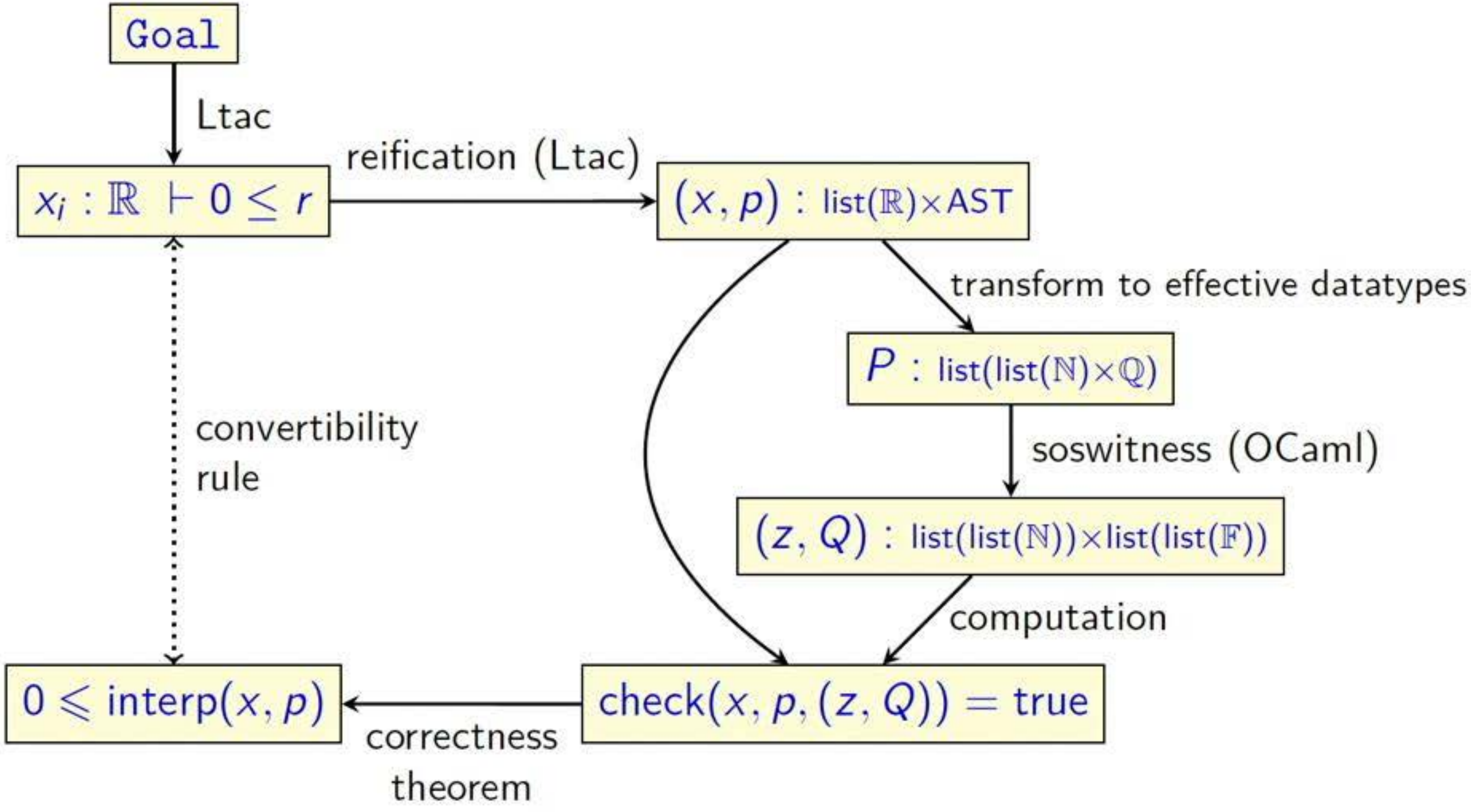
Coq Implementation

Joint work Érik Martin-Dorel

- Available at `https://sourcesup.renater.fr/validsdp/`
- LGPL license
- uses libraries
  - CoqEAL [Cano, Cohen, Dénès, Mörtberg, Rouhling, Siles] for refinement proofs (based on SSReflect and MathComp [Gonthier et al.])
  - SSrMultinomials [Strub] for multivariate polynomials
  - CoqInterval [Melquiond] and Flocq [Boldo, Melquiond] for floating-point numbers
- 15 kloc of Coq + 0.3 kloc of OCaml code

The validsdp tactic – the big picture

Joint work Érik Martin-Dorel



Benchmarks (1/2)

Problem	n	d	OSDP (not verified)	MonniauxC11 (not verified)	NLCertify (not verified)	QEPCAD (not verified)	ValidSDP	PVS/Bernstein	NLCertify	HOL Light/ Taylor
adaptativeLV	4	4	**0.75**	2.67	1.12	3.97	5.16	14.93	**2.61**	12.31
butcher	6	4	1.58	—	**1.05**	—	9.40	48.44	**8.36**	15.62
caprasse	4	4	**0.41**	1.82	0.88	5.74	5.19	25.89	**2.63**	17.68
heart	8	4	**3.18**	268.75	—	—	**16.67**	131.13	—	26.15
magnetism	7	2	**1.11**	2.04	1.64	4.61	**5.18**	245.52	14.50	16.07
reaction	3	2	0.81	1.56	**0.24**	4.38	4.33	11.48	**1.96**	12.41
schwefel	3	4	**0.95**	2.45	2.76	4.17	**3.70**	14.72	56.13	17.46
fs260	6	4	**1.25**	—	—	—	**5.99**	—	—	—
fs461	6	4	**0.70**	11.18	0.87	—	**5.18**	621.06	7.46	22.70
fs491	6	4	**0.54**	21.81	—	—	**5.38**	—	—	—
fs745	6	4	0.98	11.74	**0.94**	—	**5.55**	623.17	6.90	22.48
fs752	6	2	**0.35**	1.81	0.90	—	**3.80**	54.52	7.88	13.34
fs8	6	2	**0.43**	1.53	1.48	—	**3.93**	52.63	6.62	13.40
fs859	6	8	—	—	—	—	—	—	—	—
fs860	6	4	1.21	10.53	**1.11**	—	**6.08**	73.65	7.34	14.28
fs861	6	4	**1.09**	10.48	1.20	—	**5.15**	69.74	7.87	14.28
fs862	6	4	1.27	79.25	**1.25**	—	**5.37**	73.54	7.58	14.14
fs863	6	2	**0.94**	1.50	—	—	**3.85**	—	—	13.85
fs864	6	2	**0.56**	2.05	—	—	**4.05**	—	—	13.28
fs865	6	2	**0.76**	2.11	—	—	**3.68**	—	—	13.76
fs867	6	2	**0.21**	2.09	1.74	—	**4.22**	—	8.04	—

On Intel Core i5 2.9 GHz, time limits 900 s. All times in seconds.

Benchmarks (2/2)

Problem	n	d	OSDP (not verified)	MonniauxC11 (not verified)	NLCertify (not verified)	QEPCAD (not verified)	ValidSDP	PVS/Bernstein	NLCertify	HOL Light/Taylor
fs868	6	4	**0.94**	—	—	—	**6.05**	—	—	—
fs884	6	4	—	—	—	—	—	—	—	—
fs890	6	4	—	**7.78**	—	—	—	—	—	—
ex4_d4	2	12	—	—	—	—	—	—	—	—
ex4_d6	2	18	—	—	—	—	—	—	—	—
ex4_d8	2	24	**16.99**	—	—	—	**82.89**	—	—	—
ex4_d10	2	30	—	—	—	—	—	—	—	—
ex5_d4	3	8	**1.67**	—	—	—	**13.63**	—	—	—
ex5_d6	3	12	**16.10**	—	—	—	**66.82**	—	—	—
ex5_d8	3	16	**203.06**	—	—	—	**353.70**	—	—	—
ex5_d10	3	20	—	—	—	—	—	—	—	—
ex6_d4	4	8	**16.82**	—	—	—	**44.99**	—	—	—
ex6_d6	4	12	—	—	—	—	—	—	—	—
ex7_d4	2	12	—	—	—	—	—	—	—	—
ex7_d6	2	18	**1.50**	—	—	—	**26.78**	—	—	—
ex7_d8	2	24	**15.38**	—	—	—	**83.47**	—	—	—
ex7_d10	2	30	—	—	—	—	—	—	—	—
ex8_d4	2	8	**0.87**	15.72	—	73.75	**7.52**	—	—	—
ex8_d6	2	12	—	—	—	—	—	—	—	—
ex8_d8	2	16	—	—	—	—	—	—	—	—
ex8_d10	2	20	—	—	—	—	—	—	—	—

On Intel Core i5 2.9 GHz, time limits 900 s. All times in seconds.

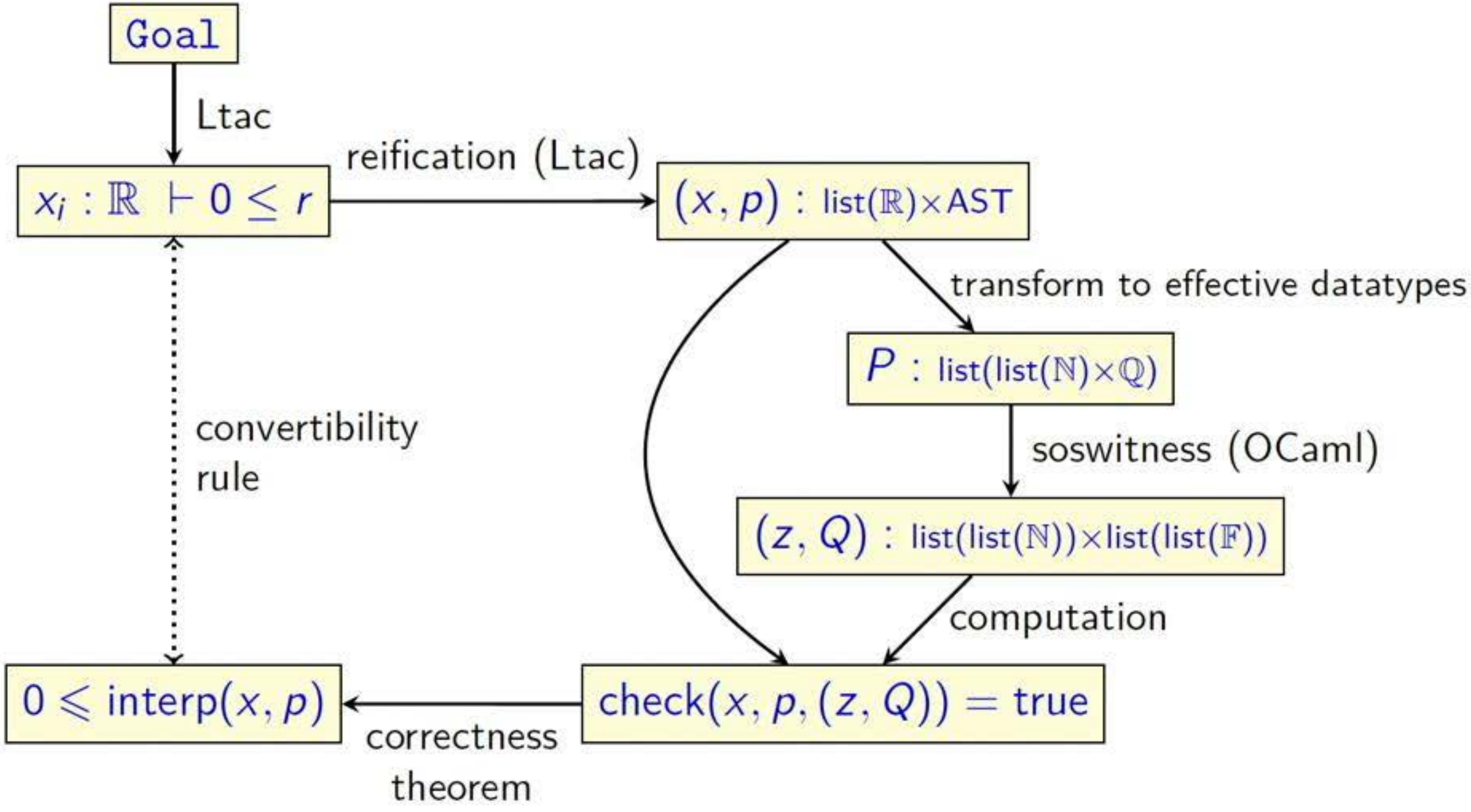
Primitive Floats in Coq

Joint work Guillaume Bertholon and Érik Martin-Dorel

- Coq offers efficient machine integers
- Enables effective floating-point computation by emulating floats with integers
- But slow (x1000 compared to OCaml)

The validsdp tactic – the big picture

Joint work Érik Martin-Dorel



Primitive Floats in Coq

Joint work Guillaume Bertholon and Érik Martin-Dorel

- Coq offers efficient machine integers
- Enables effective floating-point computation by emulating floats with integers
- But slow (x1000 compared to OCaml)

Primitive Floats in Coq

Joint work Guillaume Bertholon and Érik Martin-Dorel

- Coq offers efficient machine integers
- Enables effective floating-point computation by emulating floats with integers
- But slow ($x1000$ compared to OCaml)
- Add sound access to machine floating-point in Coq
- `https://github.com/coq/coq/pull/9867`
- Presentation at ITP next week