Analysis and synthesis: a complexity perspective

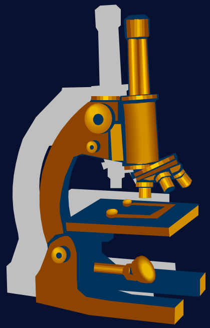
Pablo A. Parrilo
ETH Zürich

control.ee.ethz.ch/~parrilo

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Outline

- System analysis/design
- Formal and informal methods
- SOS/SDP techniques and applications
- Why we think this is a good idea
- A view on complexity
- Connections with other approaches
- Limitations, challenges, and perspectives



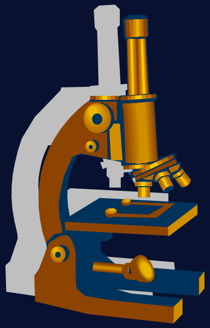
System analysis

Analysis: establish properties of systems

System descriptions (models), not reality.

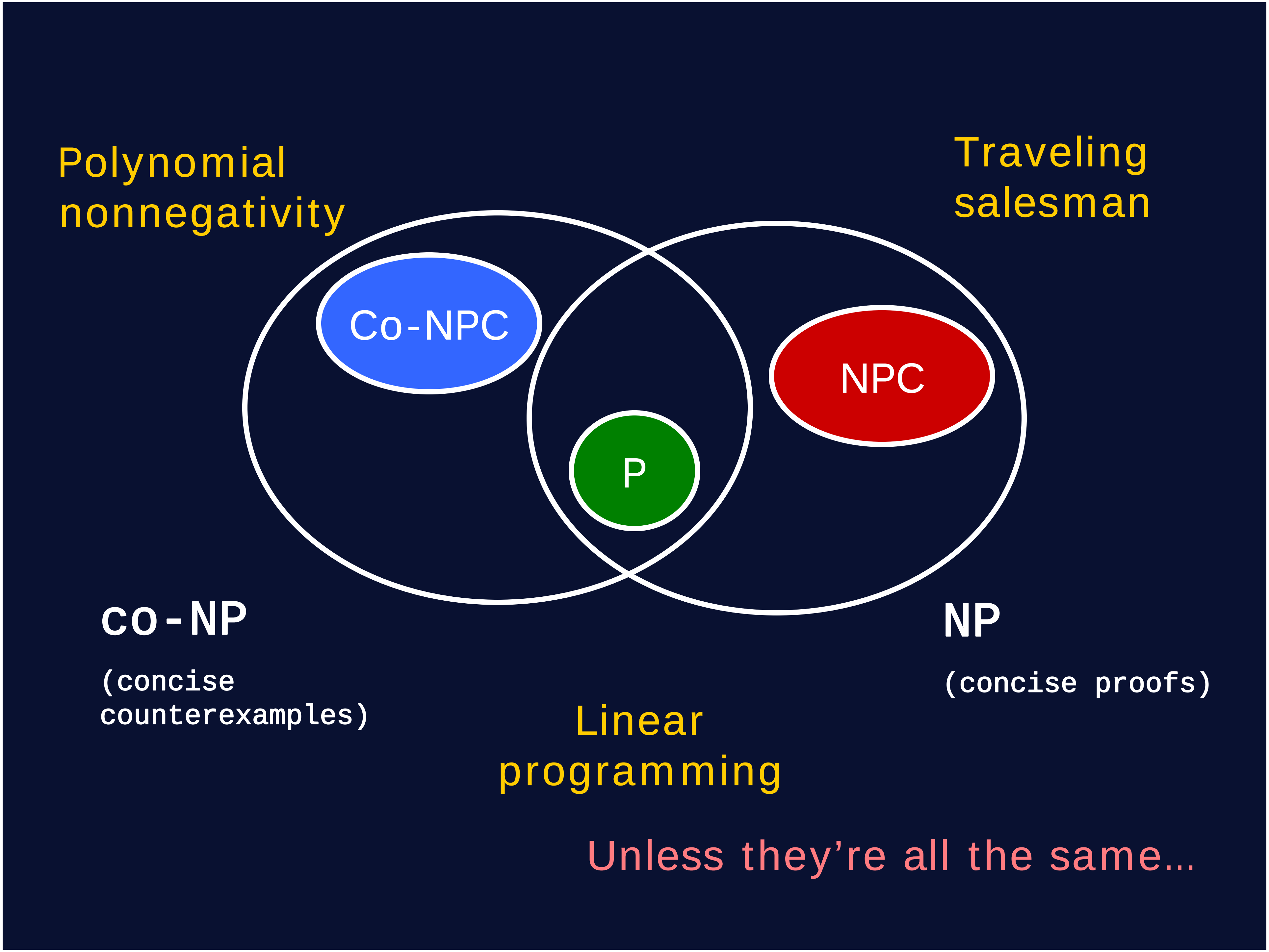
- Informal: reasoning, analogy, intuition, design rules, simulation, extensive testing, etc.
- Formal: Mathematically correct proofs. Guarantees can be deterministic or in probability.

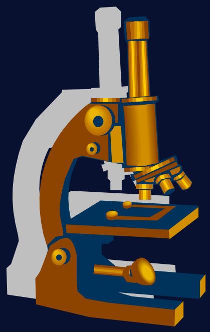
Many recent advances in formal methods (hardware/software design, robust control, etc)



Complexity issues

- What are the barriers to fully automated design/synthesis?
- How to quantify the computational resources needed for these tasks?
- How do they scale with problem size?
- If a system has a certain property, can we concisely explain why? (a scientist’s nightmare)
- Does the existence of a simple proof guarantee that we can efficiently find it?


Polynomial nonnegativity
Traveling salesman
Co-NPC
NPC
P
co-NP
(concise counterexamples)
NP
(concise proofs)
Linear programming
Unless they're all the same...



Analysis vs. synthesis

- So far, analysis or verification.

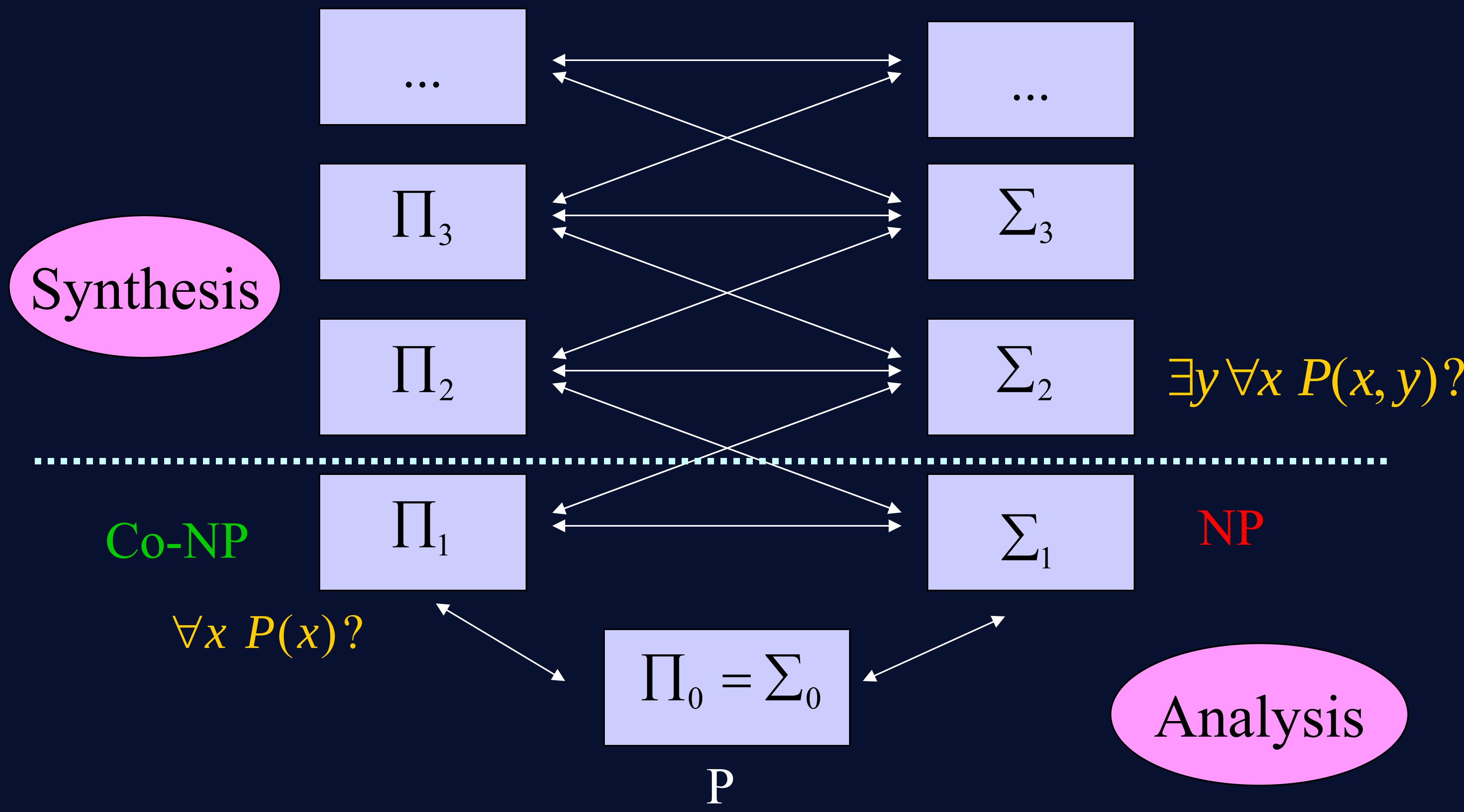
$$\forall x \; P(x)?$$

- Synthesis (design), a much more complicated beast.

$$\exists y \, \forall x \; P(x, y)?$$

- In general, a higher complexity class
- Optimization vs. games, minimax, robustness, etc
- Alternating quantifiers, relativized Turing machines: the polynomial time hierarchy.

Polynomial time hierarchy



A possible way out?

$$\exists y \forall x \; P(x, y) \geq 0$$

- In general is Π_2-hard.
- Really bad. No hope of solving this efficiently.
- But when P(x,y) is quadratic in x and affine in y…
- This is exactly semidefinite programming (SDP)
- Drops two levels to P, polynomial time !
- A reason behind the ubiquity of SDP-based methods

- Synthesis results depend on hand-crafted “tricks” that we don’t fully understand yet.
- Until recently we could say the same about analysis, where custom techniques abound.
- For analysis, there’s a method in the madness, earlier results unified and expanded.

Semialgebraic modeling

- Many problems in different domains can be modeled by polynomial inequalities

$$f_i(x) \geq 0,\ g_i(x) = 0$$

- Continuous, discrete, hybrid
- NP-hard in general
- Tons of examples: spin glasses, dynamical systems, robustness analysis, SAT, quantum systems, etc.

How to *prove* things about them, in an algorithmic, certified and efficient way?

Proving vs. disproving

- Really, it's automatic theorem proving
- Big difference: finding counterexamples vs. producing proofs (NP vs. co-NP)

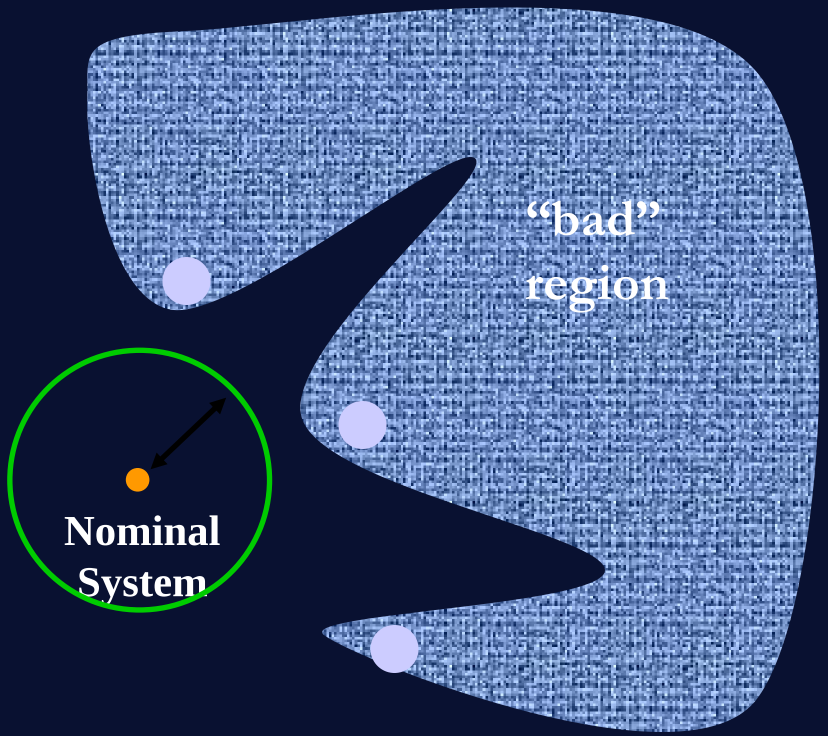
- Find bad events

(e.g. protocol deadlock, death)

or...

- Safety guarantees

(e.g. Lyapunov, barriers, certificates)



Bad events are easy to describe (NP)

Safety proofs could potentially be long (co-NP)

Proving vs. disproving

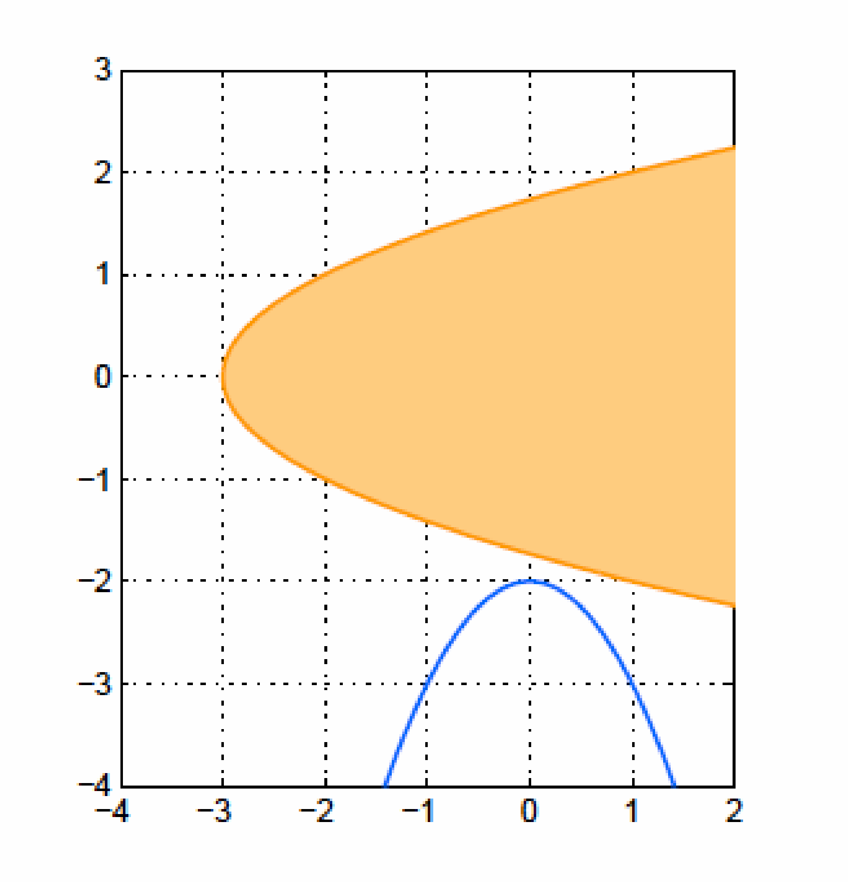
- Big difference: finding counterexamples vs. producing proofs (NP vs. co-NP)
- Decision theory exists (Tarski-Seidenberg, etc), practical performance is quite poor
- Want unconditionally valid proofs, but may fail to get them sometimes
- Rather, we use a particular proof system from real algebra: the Positivstellensatz

An example

$$\{(x,y) \mid f := x - y^2 + 3 \geq 0,\ \ g := y + x^2 + 2 = 0\}$$

Is the set described by these inequalities empty?

How to certify this?

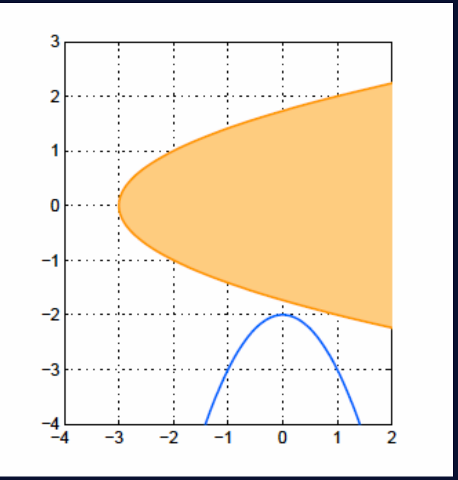


Example (continued)

$$\{(x,y) \mid f := x - y^2 + 3 \geq 0,\ \ g := y + x^2 + 2 = 0\}$$

Is empty, since

$$\underbrace{s_1 + s_2 f}_{\text{nonnegative}} + \underbrace{t_1 g}_{\text{zero}} = -1$$



with

$$s_1 = \tfrac{1}{3} + 2\left(y + \tfrac{3}{2}\right)^2 + 6\left(x - \tfrac{1}{6}\right)^2,\ \ s_2 = 2,\ \ t_1 = -6$$

Reason: evaluate on candidate feasible points

What is this? How to generalize it?

$$\{x \in R^n : \; f_i(x) \geq 0, \; g_i(x) = 0\}$$

Define two algebraic objects:

- The cone generated by the inequalities

$$\mathrm{cone}(f_i) := s_0 + \sum_i s_i f_i + \sum_{i,j} s_{ij} f_i f_j + \cdots$$

The polynomials s_α are sums of squares

- The ideal generated by the equalities

$$\mathrm{ideal}(g_i) := \sum_i t_i g_i$$

Sums of squares (SOS)

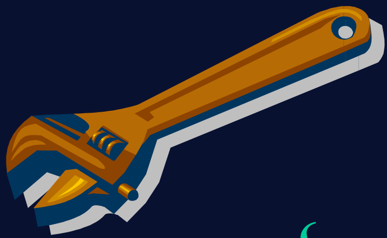
A sufficient condition for nonnegativity:

$$\exists f_i(x):\; p(x) = \sum_i f_i^2(x)\ ?$$

- Convex condition
- Efficiently checked using SDP

Write: $$p(x) = z^T Q z, \qquad Q \geq 0$$

where z is a vector of monomials. Expanding and equating sides, obtain linear constraints among the Q_{ij}.
Finding a PSD Q subject to these conditions is exactly a semidefinite program (LMI).



Positivstellensatz (Real Nullstellensatz)

$$\{x \in R^n : \ f_i(x) \geq 0, \ g_i(x) = 0\} \ \text{is empty}$$

if and only if

$$\exists \ f \in \text{cone}(f_i), \ g \in \text{ideal}(g_i) : \ \ f + g = -1$$

- Infeasibility certificates for polynomial systems over the reals.
- Sums of squares (SOS) are essential
- Conditions are convex in f,g
- Bounded degree solutions can be computed!
- A convex optimization problem.
- Furthermore, it's a semidefinite program (SDP)

P-satz proofs

- Proofs are given by algebraic identities
- Extremely easy to verify
- Use convex optimization to search for them

- Convexity, hence a duality structure:
  - On the primal, simple proofs.
  - On the dual, weaker models (liftings, etc)

- General algorithmic construction
- Based on the axioms of formally real fields
- Techniques for exploiting problem structure

Modeling

Robustness barriers

Polynomial inequalities

→

Analysis

Real algebraic geometry

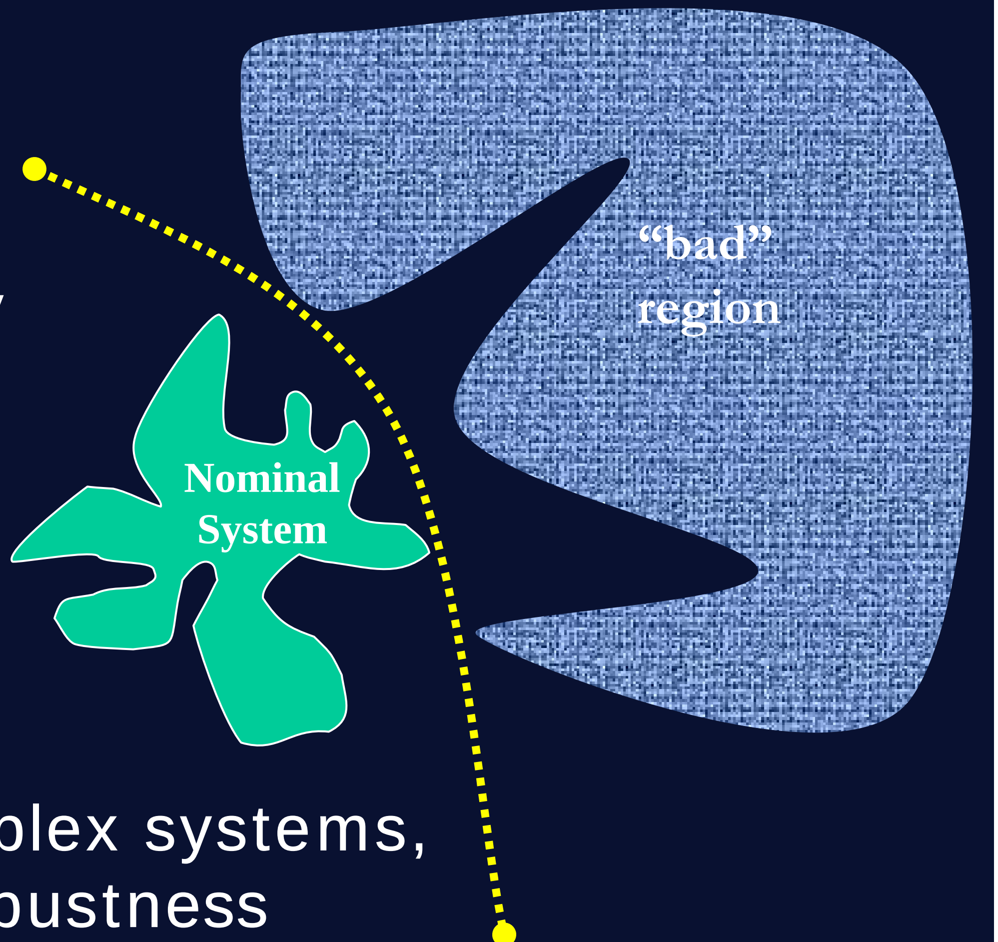
Duality

SDP/SOS

- A formal, complete proof system
- Very effective in a wide variety of areas
- Look for short (bounded-depth) proofs first, according to resources

System Analysis

- Want to decouple
  - System complexity
  - Complexity of verification.



• Even for extremely complex systems, there may exist simple robustness proofs. Try to look for those first…

Special cases

Generalizes well-known methods:

- Linear programming duality
- S-procedure
- SDP relaxations for QP
- LMI techniques for linear systems
- Structured singular value
- Spectral bounds for graphs
- Custom heuristics (e.g. NPP)

A few sample applications

- Continuous and combinatorial optimization
- Graph properties: stability numbers, cuts, ...
- Dynamical systems: Lyapunov and Bendixson-Dulac functions
- Bounds for linear PDEs (Hamilton-Jacobi, etc)
- Robustness analysis, model validation
- Reachability analysis: set mappings, ...
- Hybrid and time-delay systems
- Data/model consistency in biological systems
- Geometric theorem proving
- Quantum information theory

DS applications: Bendixson-Dulac

Does a dynamical system have periodic solutions? How to rule out oscillations?

- In 2D, a well-known criterion: Bendixson-Dulac
- Higher dimensional generalizations (Rantzer)
  - Weaker stability criterion than Lyapunov (allowing a zero-measure set of divergent trajectories).
  - Convexity for synthesis.
- How to search for ρ ?

$$\nabla \cdot (\rho f) > 0$$

Bendixson-Dulac

- Restrict to polynomial (or rationals), use SOS.
- As for Lyapunov, now a fully algorithmic procedure.

Given: $\dot{x} = y$

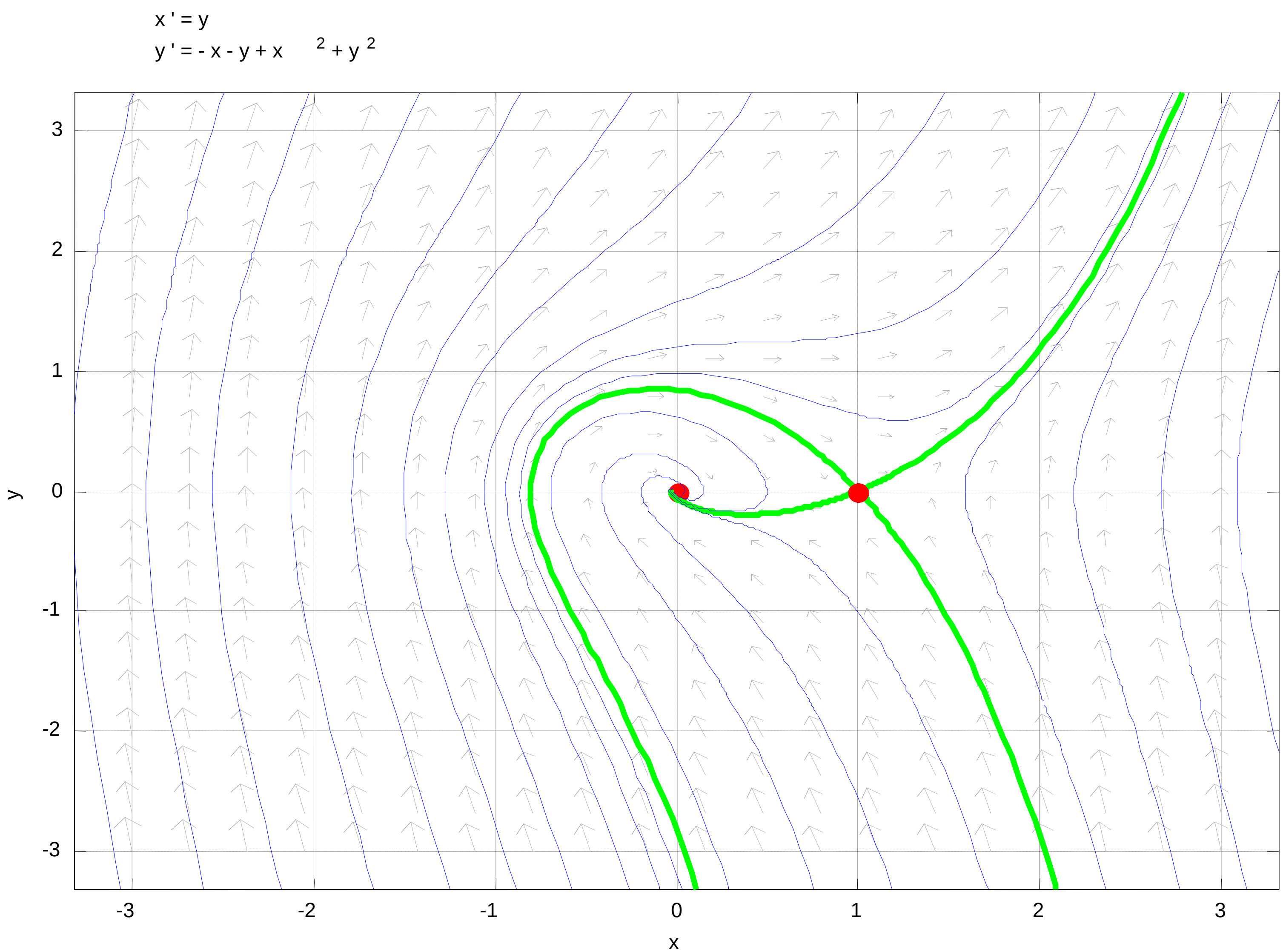
$\dot{y} = -x - y + x^2 + y^2$

Propose: $\rho = a + bx + cy$

After optimization: $a = -\frac{1}{2} - \sqrt{3}, \quad b = \sqrt{3}, \quad c = 1$

$$\nabla \cdot (\rho f) = 3\left(y + \frac{1}{3}\sqrt{3}\, x - \frac{1}{6}\sqrt{3} - \frac{1}{2}\right)^2 - \frac{1}{2} + \frac{1}{2}\sqrt{3} \quad > 0$$

Conclusion: a **certificate of the inexistence of periodic orbits**



Example: Lyapunov stability

$$\dot{V} = \nabla V \cdot f < 0$$

$$V \geq 0$$

- Ubiquitous, fundamental problem
- Algorithmic solution
- Extends to uncertain, hybrid, etc.

Given:

$$\dot{x} = -2y + 3x^2 - x^3$$
$$\dot{y} = 6x - 2y$$

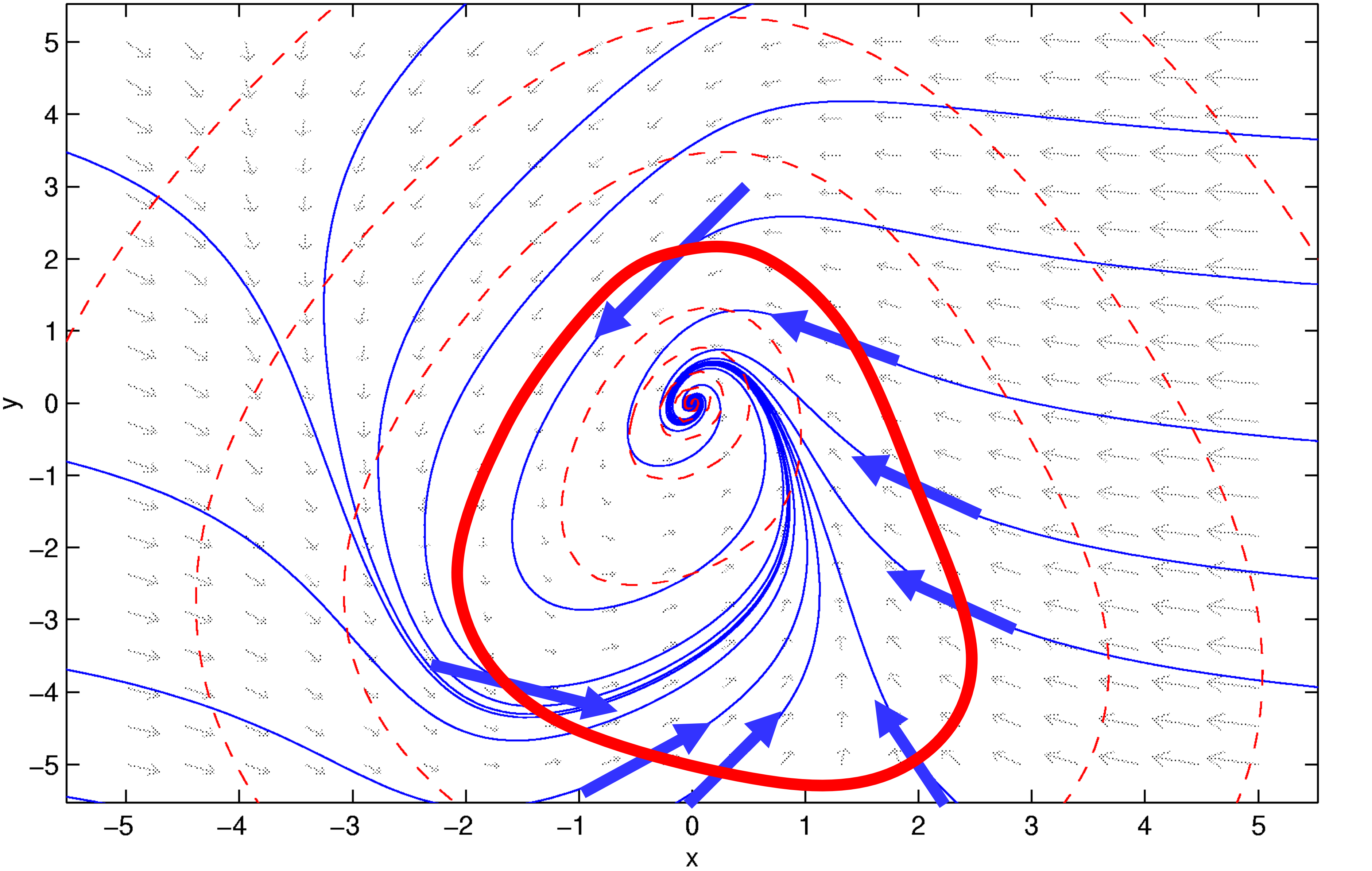
Propose:

$$V(x, y) = \sum_{i+j \leq 4} c_{ij} x^i y^j$$

After optimization: coefficients of V.

A Lyapunov function V, that proves stability.

Conclusion: a **certificate of global stability**



Why do we like these methods?

- Very powerful!
- For several problems, best available techniques
- In simplified special cases, reduce to well-known successful methods
- Reproduce domain-specific results
- Very effective in “well-posed” instances
- Rich algebraic/geometric structure
- Convexity guarantees tractability
- Efficient computation

Complexity

- Traditional view: worst-case over classes of instances
- Rather, an instance-dependent notion: proof length

- Our claim: this makes more sense for systems designed to be *robust*
- Our hope: also holds for biology

Things to think about

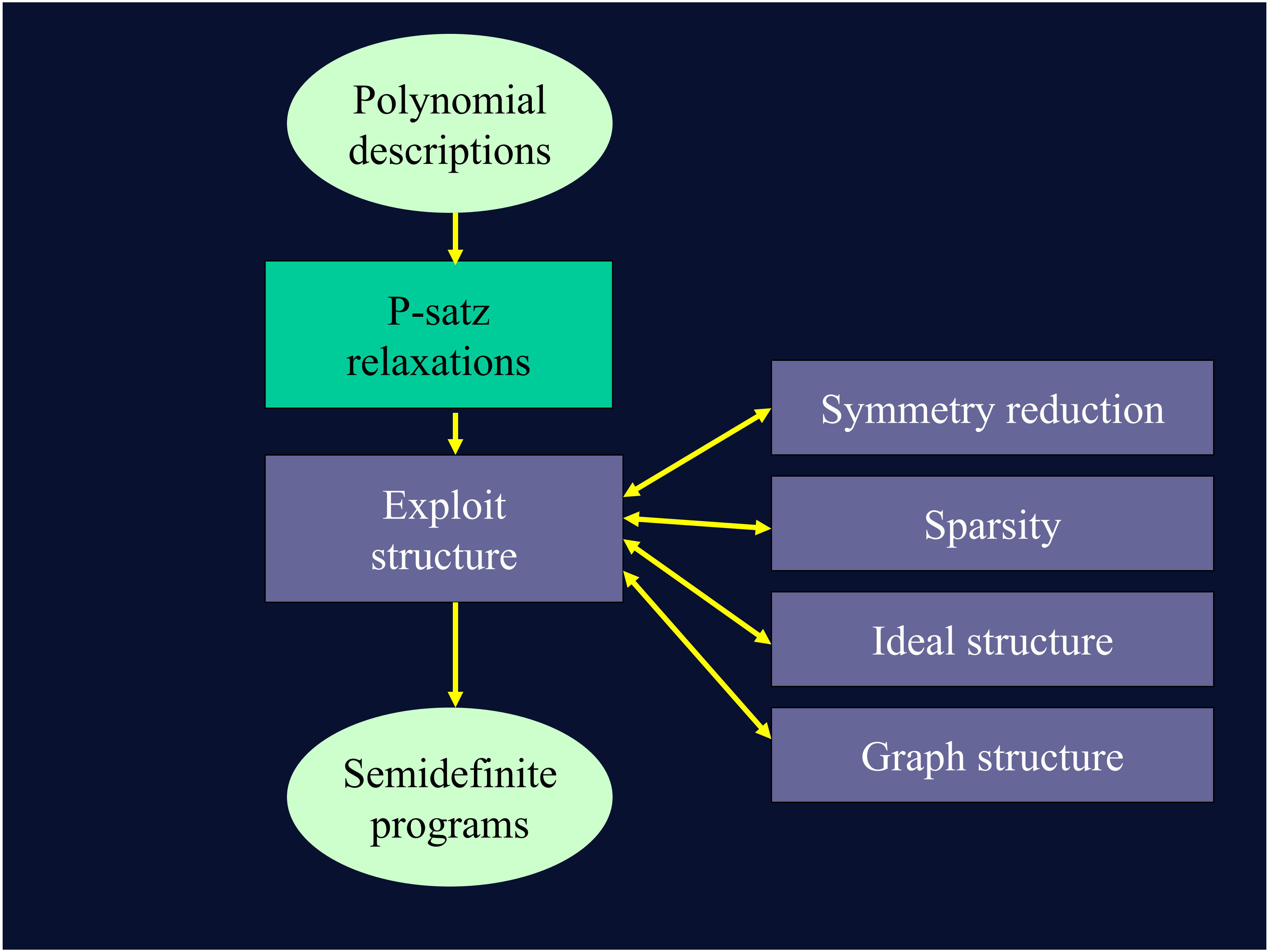
- Correct notion of proof length?
  - Degree? Straight-line programs?
- “Smart” proof structures?
- Proof strategies affect proof length
  - P-satz proofs are global
  - For some problems, branching is better
- Decomposition strategies
  - (Re)use of abstractions

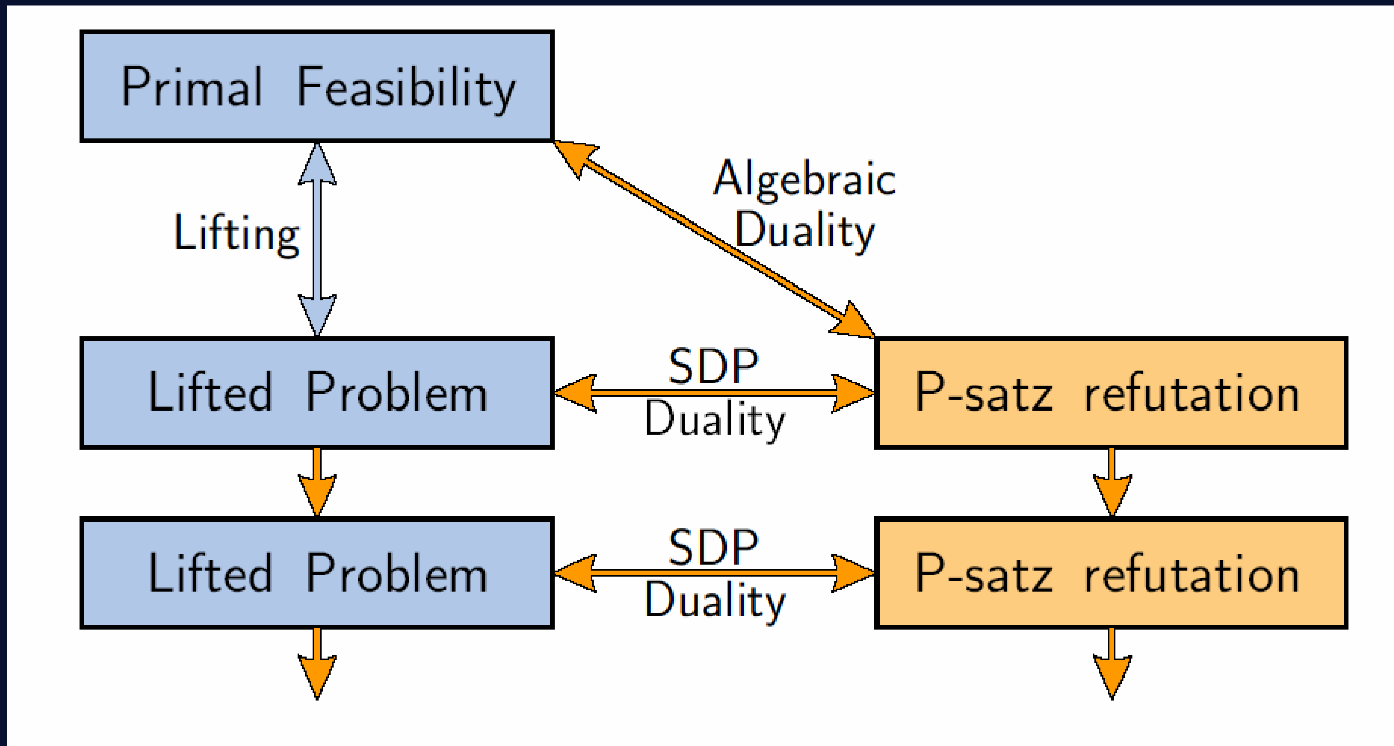
Exploiting structure

Isolate algebraic properties!

- Symmetry reduction: invariance under a group
- Sparsity: Few nonzeros, Newton polytopes
- Ideal structure: Equalities, quotient ring
- Graph structure: use the dependency graph to simplify the SDPs

Methods (mostly) commute, can mix and match

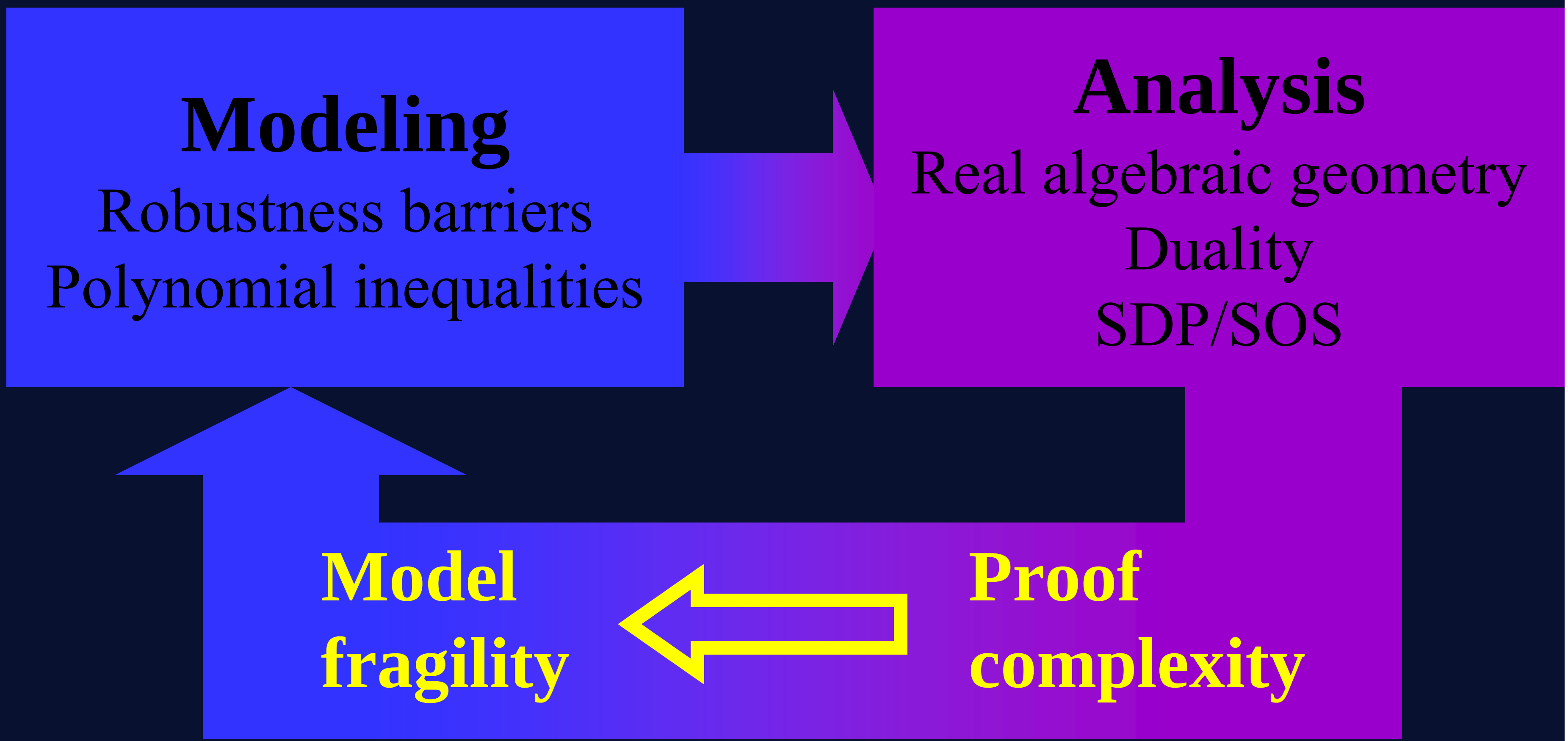

Polynomial descriptions
P-satz relaxations
Exploit structure
Semidefinite programs
Symmetry reduction
Sparsity
Ideal structure
Graph structure



A convexity-based scheme has dual interpretations

Want to feedback information from the dual

For instance, attempting to proving emptiness,
we may obtain a feasible point in the set.



Use dual information to get info on primal fragility

Numerical issues

- SDPs can essentially be solved in polynomial time
- Implementation: SOSTOOLS (Prajna, Papachristodoulou, P.)
- Good results with general-purpose solvers. But, we need to do much better:
  - Reliability, conditioning, stiffness
  - Problem size
  - Speed
- Currently working on customized solvers

Future challenges

- Structure: we know a lot, can we do more?
- A good algorithmic use of abstractions, modularization, and randomization.
- Reuse/parametrization of known tautologies
- Infinite # of variables? Possible, but not too nice computationally. PSD integral operators, discretizations, etc.
- Incorporate stochastics
- Other kinds of structure to exploit?
- Algorithmics: alternatives to interior point?
- Do proofs need domain-specific interpretations?

Summary

- New mathematical tools
- Algorithmic construction of P-satz relaxations
- Generalization of many earlier schemes
- Very powerful in practice
- Done properly, can fully exploit structure
- Customized solvers in the horizon

- Lots of applications, many more to come!