

Feasible Proofs and Computations

Disclaimers

1. I wanted to give (slightly) more serious/technical talk...



2. Please help
Simon Fraser U
November 20, 2006

3. The talk is **not** very homogeneous or connected

What Mathematicians and Society thinks

- **Mathematical proof:** *the cogency of evidence that is used to show the truth of a mathematical assertion. In modern mathematics, a proof begins with one or more statements called premises and demonstrates, using the rules of logic, that if the premises are true then a particular conclusion must also be true.*
- **Computation:** *the act or action of computing; the use or operation of a **computer**... machine that performs tasks, such as calculations or electronic communication, under the control of a set of instructions called a program.*

In earlier mathematics, proofs and computations of Empirical Algorithms were not strictly separated

- $3^2+4^2=5^2$ (constructing the right angle in Egypt, well before the Pythagorean Theorem)
- solving algebraic equations of 3rd and 4th degrees (16th century, no justification then)
- “practical” integrating and differentiating, (17th century, well before the rigorous theory of reals and limits)
- number-theoretical algorithms (Fermat, Gauss etc.)



In today's world...

- **Landau to his students:** *I am interested if someone can ~~integrate an equation~~ build the right angle.*
Mathematical lyrics [rigorous theorems] is of no relevance
- **Computer-aided mathematics:** (well, almost) as experimental as any natural science

The last one completely accepts the basic rules of the game... today we will see one more ``attack'' from within.

Feasible proofs and computations

Feasible usually stands for “resources are bounded by a polynomial in the input size”

Proofs are not always “Hilbert-style” (axioms, inference rules, lemmas, theorems etc.), computations are not always “programs”, and they are not always strictly separated

(Advanced) example

MIP=NEXPTIME

(Babai, Fortnow, Lund 91)

Multi-interactive **proofs** = non-deterministic
exponential time **computations**



Overview of this talk

- Introduction: feasible proofs
- Bounded Arithmetic
- Propositional proof complexity
- Feasible (un)provability of $P \neq NP$
- Non-determinism
- Interactive proofs
- Probabilistically checkable proofs (PCP)
- Hardness of approximations
- Natural proofs

The selection of topics is highly personal

What is a “feasible” (classical) proof?

Feasible usually stands for “polynomial resources”

It is quite natural to assume that all concepts involved in the **statement** must be **computationally** feasible.

Does this always suffice? Or there are statements that involve only computationally feasible objects with proofs requiring non-feasible ones?

Fermat's little theorem

$$\exists x \in [1, p-1] (x^{p-1} \neq 1 \pmod{p}) \Rightarrow$$

p is composite

Meet our guides

Prover

Verifier

$$\exists x \in [1, p-1] (x^{p-1} \neq 1 \pmod p) \Rightarrow p \text{ is composite}$$

Take the cyclic subgroup in $GF(p)^*$ generated by x . It is finite...

What??? I do not know **anything** about infinity...

$$\exists a (x^a = 1 \pmod p)$$



Well, you probably mean this a must be **feasible** (= small). Anyway, why?

$$\exists a(x^a = 1 \pmod p)$$

Well, you probably mean this a must be **feasible** (small). Anyway, why does it exist?

Let me see... Yeah, a is feasible. Anyway, consider the sequence x^1, x^2, x^3, \dots . By the pigeon-hole principle...

By what??? Not even to mention that (unlike a) your sequence has exponential length, and that's not feasible to me...



Bounded Arithmetic

- **Bounded Arithmetic** is a generic name for a collection of weak (well below $I\Delta_1$) arithmetical theories capturing feasible **arguments**
- customary look: open axioms describing non-logical symbols + schemes of limited induction (+ comprehension, collection etc. in the second-order case)
- main characteristic feature: focus on **bounded** quantifiers $(\exists x \leq t), (\forall x \leq t)$, as opposed to unbounded.

S_2^1 is the most popular system in this family

Witnessing theorems. Every statement in a S_2^1 proof can be “witnessed” by a computationally feasible object.

Example (Buss, 1986): whenever we see the statement $(\exists x \leq t) A(x, y)$, we associate to it a polynomial-time computable function $f(y)$ such that $A(f(y), y)$.

Corollary. If **FACTORING** is hard then Fermat's little theorem is not provable in S_2^1

Classical feasible proofs = proofs in S_2^1

Fun Corollary: two ways to destroy the world economy.

- Quantum Computer
- Prove Fermat's little theorem in S_2^1

Moving to the propositional world

Or on relative complexity of syntax and semantics

	Provable/valid	Unprovable/not valid
First-order formulas (quantifiers)	First-order proof: finite object <	Model: countable and in general very complicated
Propositional formulas (no quantifiers)	Propositional proof: (believed to be) in general exponentially long >	Falsifying assignment: absolutely feasible

Propositional Proof Complexity

(Cook, Reckhow 73)

- All about (the complexity of) showing that a propositional formula φ is a tautology (identically true)
- Ordinary (sound and complete) textbook propositional calculus = **Frege proof system**
- We are interested in the (optimal) **length** of propositional proofs
- A proof is **feasible** if this length is polynomial in the length of φ

Some (random) remarks

- The two notions of feasibility (Bounded Arithmetic and Propositional Proof Systems) are virtually the same: “almost open” arithmetic formulae can be translated to propositional ones.
- **Big, big** conjecture: no propositional proof system possesses **feasible** proofs of **any** tautology.
- Propositional proof complexity has many things in common with Boolean (circuit) complexity: both are about understanding what and how can be represented by short formulas.

Some “combinatorial” results

(or what I would have talked about if Valentine would not have forbidden me)

- **Pigeon-hole principle** – m pigeons can not be seated into n holes ($m > n$).
- **Bounded-depth Frege** – fragment of Frege in which the number of alternation of quantifiers in every formula is bounded by an absolute constant.
- **Resolution** – operates with **clauses** like
$$\neg x_1 \vee x_3 \vee x_7$$

- **Theorem.** (Ajtai, Bellantoni, Pitassi, Urquhart, Beame, Impagliazzo, Krajíček, Pudlák, Woods 88-92) There's no feasible proof of the pigeon-hole principle with $n+1$ pigeons and n holes in bounded-depth Frege...
- but there's one with n^2 (or even $2n$) pigeons.
(Paris, Wilkie, Woods, Maciel, Pitassi, 88-00)
- **Theorem.** (Raz, Razborov 02) There's no feasible proof of the pigeon-hole principle with n^2 pigeons and n holes in Resolution.

Feasible provability of $P \neq NP$

Circuit complexity: trying to prove that $P \neq NP$ by proving “lower bounds” for circuits

All known proofs in circuit complexity are feasible (thesis, Razborov 1995)

Propositional proof complexity has many things in common with Boolean (circuit) complexity: both are about understanding what and how can be represented by short formulas.

Fires back

Realistic hope and fascinating (but very hard) project: show that there is no feasible proof of $P \neq NP$ modulo some standard complexity assumptions (like **FACTORING** is hard)

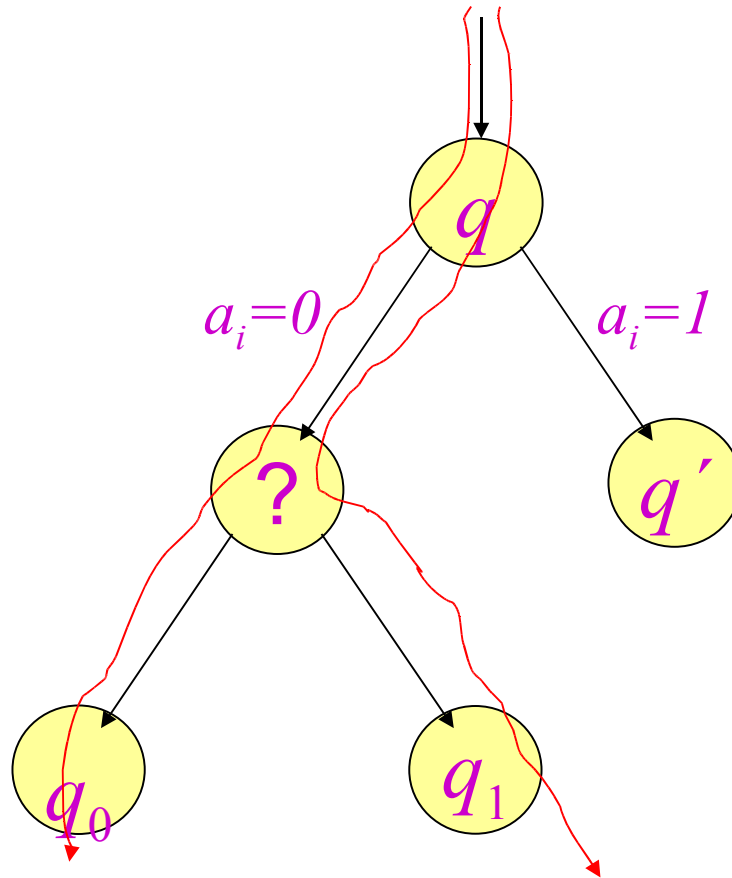
Non-determinism

Pandora box or “what if”?

- How to prove that φ is **not** a tautology or (equivalently) that $\neg\varphi$ is **satisfiable**?
- Trivial proof system for this task: “proofs” are satisfying assignments.
- **Verification** of a proof is feasible (and easy)
- **Search** for one, however, is entirely different matter and does not appear to be feasible (e.g. brute-search involves checking exponentially many possibilities)

S. Cook (1971), *The complexity of theorem
proving procedures*

Proof search also becomes computationally
feasible if we allow **non-determinism**



- A non-deterministic machine **accepts** if and only if there exists **at least one** accepting computational path and **rejects** otherwise.
- SATISFIABILITY: non-deterministically **guess** a satisfying assignment and then (deterministically) **verify** it.

Non-deterministic computations = search for efficient proofs

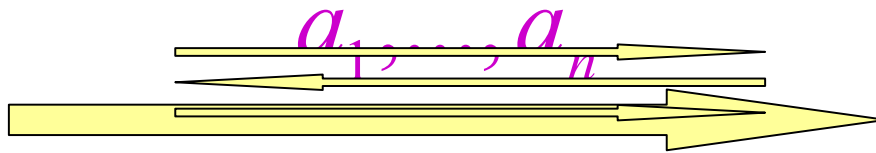
Interactive Proofs

?

$\phi \in L_n$

Prover

Verifier



- ϕ is satisfiable \implies Prover can make Verifier accept (if he plays optimally)
- ϕ is not satisfiable \implies Verifier always rejects (no matter what Prover is doing)



Goldwasser, Micali, Rackoff; Babai 1985

$\varphi \in L$? +

Prover

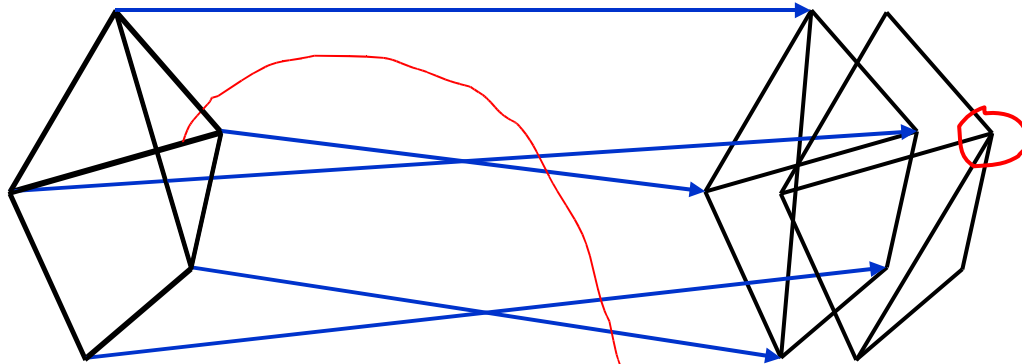
Verifier

~~Still only NP languages!~~

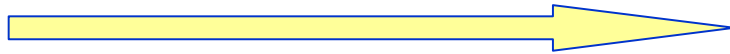
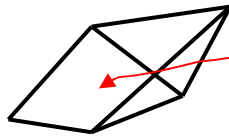
(hint: Isaac Asimov, Foundation Series)

- φ is in L \Rightarrow Prover can make Verifier accept **always** (with probability 1)
- φ is not in L \Rightarrow Verifier rejects (**any** prover) **with high probability**

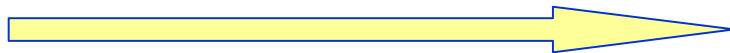
Graph non-isomorphism



Picks at random one of the two graphs and randomly permutes it



Left!!!



- **PCP Theorem**. (Feige, Goldwasser, Lovász, Safra, Szegedy, Arora, Lund, Motwani, Sudan 1992) There is a proof system for **SATISFIABILITY** in which Verifier flips $O(\log n)$ coins and check only **constantly many bits** in the submitted proof.
- The proof is (surprisingly) technically involved and uses non-trivial mathematics (multi-linear polynomials).
- An entirely new proof related to yet another major result **SymmetricLogSpace=Logspace** (Reingold 2004) was recently given by **Dinur (2005)**

Hardness of approximation

(or back to Earth again)

- **NP** problems are formulated as decision (yes/no) questions.
- Optimization problems are their relaxed versions (“how many”), and solutions are also allowed to be approximate.
- **Prototypical example**: what is the maximal number of clauses in a 3-CNF that can be simultaneously satisfied?
- PCP theorem shows that if we can even **approximate** this number sufficiently well (say, within a factor of 2) then we can solve any problem in **NP exactly**.
- Rich theory built since that allows us to draw similar conclusions for a large variety of optimization problems.

Natural Proofs (of $P \neq NP$)

- *Realistic hope and fascinating (but very hard) project:*
The goal is the same: identify a class of arguments that includes (virtually) all known lower bounds in standard complexity assumptions (like **FACTORING** is circuit complexity).
show that there is no feasible proof of $P \neq NP$ modulo some
- **Natural Proofs** (Razborov, Rudich 1995) approach this goal by **postulating** that every proof gives rise to a predicate on Boolean functions (*property*) that possesses some natural computational and combinatorial properties.
- Natural and feasible proofs are *very* close in spirit but seem to be formally incomparable.
- For Natural Proofs we do have the result anticipated (but still elusive) for feasible proofs.

Razborov, Rudich 1995:

If one-way functions exist (e.g. **FACTORING** is hard), then there is no Natural Proof of **P \neq NP**

- The strongest propositional proof systems for which we have a similar conclusion are **Res($\epsilon \log n$)** (operating with **($\epsilon \log n$)-DNF**) [**Razborov 03**] and Polynomial Calculus [**R 98**].
- Both are still very, very weak.

Summary (sort of)

- Even the most boring things like propositional logic begin to shine when looked at through the prism of feasibility. Plenty of unsolved problems, hard, important and mysterious.
- The classical concept of a “purely mathematical” proof is challenged and is brought closer to the everyday concept of “compelling evidence”. Still, **metamathematics** remains completely classical: e.g. every statement about the existence (or non-existence) of Interactive Proofs is a “normal” mathematical theorem.

Summary (cntd.)

- Pursuing the most abstract, **philosophical** questions (like what is the combined power of non-determinism, interaction and randomization), quite concrete **mathematical** concepts (e.g. multi-linear polynomials) are used.