Proof Complexity and Computational Hardness

Sam Buss
Department of Mathematics
U.C. San Diego

Swansea, Wales
July 2006

Tutorial Outline

- Day I. Proof Complexity and Feasible Computation Classes.
- Day II. Bounded Arithmetic and Propositional Proofs
- Day III. On the (Lack of) Progress Towards "P versus NP".

Proof Complexity and Computational Hardness

I. Proof Complexity and Feasible Computation Classes

Sam Buss
Department of Mathematics
U.C. San Diego

Swansea, Wales
July 3, 2006

Lecture I. Talk Outline

- Propositional formulas: tautologies and proofs.
- Connections to non-determinism and P versus NP.
- A taxonomy of proof systems.
- Proof search and automatizability.
- Anti-cut elimination theorem for Frege systems.
- Interpolation.

Propositional Logic

Boolean connectives: $\wedge$ (AND); $\vee$ (OR); $\neg$ (NOT); etc.

Boolean variables: $p, q, r, \ldots, x, y, z, \ldots,$ range over $\{T, F\}$.

Tautology: A valid (=always true) formula.

Combinatorial principles as tautologies.

Example 1: The pigeonhole principle. For $n \geq 1$, $0 \leq i \leq n$ and $0 \leq j < n$,

$$\neg\left[\bigwedge_i \bigvee_j p_{i,j} \;\wedge\; \bigwedge_{i<i'} \bigwedge_j \neg(p_{i,j} \wedge p_{i',j})\right].$$

Example 2: Primality tautologies. Fix $a > 1$ a prime number. Let a have n bits in its binary representation. $a = (a_{n-1}, \ldots, a_0)_2$.

$$\neg \left[Product(\vec{p}, \vec{q}, \vec{a}) \wedge (\bigvee_{i>1} p_i) \wedge (\bigvee_{i>1} q_i) \right].$$

Here "$Product$" expresses base-2 multiplication. a_i's are constants.

Example 3. Partial consistency. Fix $n > 0$. Encode proofs in ZF set theory as strings of bits in some natural way. Consider a formula:

$$\neg Encodes\_Contradiction\_In\_ZF(p_0, \ldots, p_n).$$

This statement can be encoded with a polynomial size formula. With introduction of extra "helper" variables, it can even be encoded as the negation of a CNF formula. These are tautologies if set theory is consistent!

P and NP

Def'n: P is the class of predicates (decision problems) that are decidable in polynomial time in the length n of the input.

NP is the class of predicates for which "Yes" answers are verifiable in polynomial time.

Examples in P. Given integer x, is x perfect square? Given integers x and y, is the middle bit of the product $x \cdot y$ a '1' ?

Examples in NP: Is x a composite?

Much less obviously, is x a prime? [Pratt, 1975].
Very much less obviously, the set of primes is in P. [Agarwal et al., 2002].

Open Conjecture: Integer Factorization is not in P.
This conjecture and related ones are the basis of the theory of public key cryptography.

Examples of NP problems

Hamiltonian cycle. Given graph G, does it have a Hamiltonian cycle?

SAT (Satisfiability): Given a propositional formula, over connectives AND ($\wedge$), OR ($\vee$), NOT ($\neg$), and with variables $p, q, r, \ldots$, does it have a satisfying assignment? That is, can the variables be set so as to make the formula true?

k-**Provability.** Given a formula (a theorem), does it have a proof of $\leq k$ symbols in some given formal proof system?

All three of these are NP-complete (the third at least for certain proof systems).

Def'n A problem is *co*-NP if its complement is NP. I.e., its "No" answers are verifiable in polynomial time.

Example 1: The set of primes is obviously in co-NP.

Example 2: The set of tautologies is co-NP-complete.

Note that φ is a tautology iff $\neg\varphi \in \mathrm{SAT}$.

Methods for *proving* tautologies: (a) Method of truth tables, (b) decision trees, (c) Give a proof in a formal system (e.g., a Frege system). (d) Etc.

(a),(b) require exponential size proofs. For (c), it open, firstly, whether polynomial size proofs exist and, secondly, whether proofs can be found efficiently.

Cook's Program for the P-NP problem

Def'n. A *Frege proof* system, $\mathcal{F}$, is a proof system for propositional logic, say using $\land$, $\lor$, $\neg$, $\rightarrow$, based on a finite set of axiom schemes such as $A \rightarrow (B \rightarrow A)$, and based on a finite set of inference rules, such as modus ponens:

$$\frac{A \qquad A \rightarrow B}{B}$$

A Frege system is sound and complete (implicationally).

These are the usual "textbook" proof systems.

Open: Find good upper bounds on the lengths of tautologies. I.e., find slow-growing function f such that every tautology of length n, has an $\mathcal{F}$-proof of $\le f(n)$ symbols.

$f(n) = 2^{O(n)}$ suffices. Can $f(n)$ be polynomial, $n^{O(1)}$?

Thm: [Cook'75]. If Frege proof lengths can be polynomially bounded, then $NP =$ co-NP

Pf. The tautologies are co-NP-complete. If they have polynomial size $\mathcal{F}$-proofs, they would be in NP (by simply guessing the proof). From this $NP =$ co-NP would follow. q.e.d.

Cook's program Starting with weak proof systems for propositional logic, prove superpolynomial lower bounds on the size of proofs. Work up to superpolynomial lower bounds on stronger systems such as Frege systems, eventually to all proof systems. This would prove $NP \neq$ co-NP, hence $P \neq NP$.

This has been carried out only for restricted proof systems.

Some Proof Systems

Truth tables	–
Resolution	Clauses
Cutting planes	Linear integer inequalities
Nullstellensatz	Finite field identities
$\star$ Constant-depth (cd) Frege	–
cd-Frege with counting axioms	–
....	....
cd-Frege with counting gates	–
$\star$ Frege systems	Poly size formulas
$\star$ Extended Frege systems	Poly size circuits
Quantified Frege systems	Non-uniform Polynomial Hierarchy
Set theory	–

Superpolynomial lower bounds are known for systems above the dotted line.

Extended Frege systems

An extended Frege system is a Frege system augmented with the ability to introduce abbreviations on the fly with the *extension rule:*

$$q \leftrightarrow A$$

where A is any formula, and q must be a "new" variable that does appear yet in the proof, in A, or in the formula to be proved. This introduces q as an abbreviation for A.

Introducing abbreviations allows proof length to be shorter (well, this is open), since long formulas can be replaced by abbreviations.

Equivalently: extended Frege systems are Frege systems that use Boolean circuits instead of formulas.
Also equivalent: Extended Frege systems are Frege systems with proof length measured in terms of the number of inferences in the proof.

The pigeonhole principle (PHP) as a tautology

Let $[n] = \{0, \ldots, n\}$.

The PHP states there is no 1-1 function $f : [n] \rightarrow [n-1]$. To encode this as a tautology, use propositional variables $p_{i,j}$ which express the truth of $f(i) = j$.

The PHP_n^{n+1} tautology is:

$$\neg\left[\bigwedge_i \bigvee_j p_{i,j} \;\wedge\; \bigwedge_{i \neq i'} \bigwedge_j \neg(p_{i,j} \wedge p_{i',j})\right].$$

Let's give a proof of this by contradiction ....

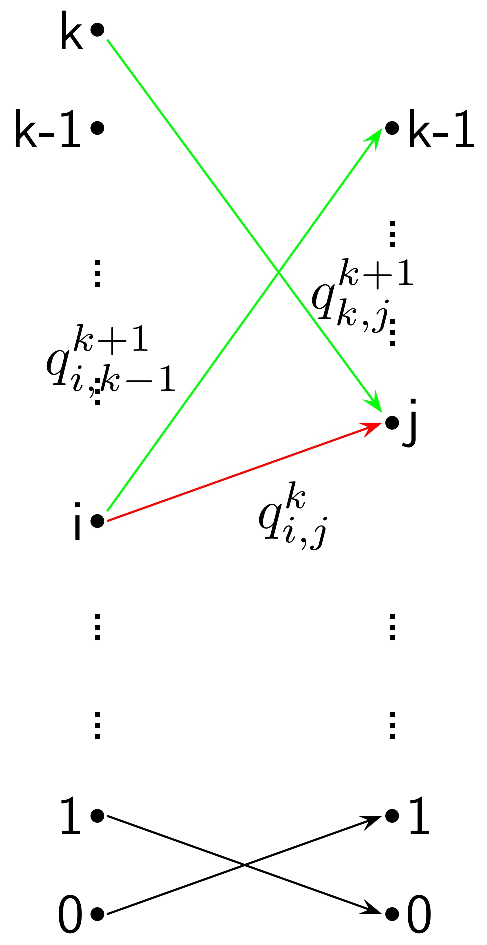
Phase 1: Define variables $q^k_{i,j}$ that define a violation of PHP^{k+1}_k. For this, let $q^n_{i,j} = p_{i,j}$ and define

$$q^k_{i,j} \leftrightarrow q^{k+1}_{i,j} \vee (q^{k+1}_{i,k-1} \wedge q^{k+1}_{k,j}).$$

Phase 2: Prove that if q^{k+1}'s violate the PHP, then so do the q^k's.

Phase 3: Proof is done, the PHP^{n+1}_n implies PHP^2_1, which is impossible.

QED



This gives poly size extended Frege proof, but not poly size Frege proof.

Thm. The PHP_n^{n+1} tautologies have polynomial size Frege proofs.

Thm. [Haken] The PHP_n^{n+1} tautologies require exponential size resolution proofs. (Actually, refutations.)

Def'n The *depth* of a propositional formula is the number of alternations of $\wedge$'s and $\vee$'s. For this count, implications are replaced and negations are pushed to the variables.

A constant depth Frege proof is a (family) of proofs in which the depth of formulas are bounded by a constant.

Thm. [PBI,KPW] The PHP_n^{n+1} tautologies require exponential size constant-depth Frege proofs.

Proof used an extension of the Hastad switching lemma.

Automatizability and Proof Search

Cook's program concerned only the *existence* of proofs. For practical application, proof *search* is at least as important.

Def'n A proof system T is *automatizable* provided there is an algorithm $f(x)$ and a constant $c > 0$ such that, whenever $T \vdash \varphi$ with a proof of k symbols, then $f(\varphi)$ runs for less than k^c steps and outputs a T-proof of φ.

Automatizability means there is a polynomial time algorithm for proof search.

Def'n A product of two primes congruent to 3 mod 4 is called a *Blum integer*.

Thm [Bonet-Pitassi-Raz] If Frege proof systems are automatizable, then there is a probabilistic polynomial algorithm for factoring Blum integers.

(Same holds for TC^0-Frege and weaker results for constant depth Frege.)

Proof - outline. The proof is based on a reduction from automatizability to Craig interpolation using Diffie-Hellman cryptographic hardness.

For P a Blum integer, $g \in \mathbb{Z}_P^*$, $i = 0, 1$, the formulas $A_i(P, g, X, Y, a, b, \ldots)$ state:

$$g^a \equiv X \bmod P \text{ and } g^b \equiv Y \bmod P \text{ and } ((g^{ab} \bmod P) \bmod 2) = i.$$

Intuitively, $\langle P, g, X, Y \rangle$ encode bit value "i" in the Diffie-Hellman encoding. The '. . . ' means extra variables present that depend only P, g, X, Y and help the encoding. The extra variables are polynomial time computable.

Proof - continued

Claim: There are poly size Frege proofs of

$$\neg A_0(P, g, X, Y, a, b) \vee \neg A_1(P, g, X, Y, c, d).$$

Pf of Claim: This is the correctness of Diffie-Hellman: Namely, if both $A_0(P, g, X, Y, a, b)$ and $A_1(P, g, X, Y, c, d)$ then

$$g^{ab} \equiv g^{cd} \bmod P,$$

which is a contradiction. This proof, with suitable extra variables, can be carried out with poly-size Frege proofs.

Proof - continued

If $i = 0$ is correct, then for suitable choice of a_0, b_0 there is a poly-size Frege proof of $A_0(P, g, X, Y, a_0, b_0)$ — a_0, b_0 are constants. By the claim, there is then a short proof of $\neg A_1(P, g, X, Y, c, d)$ where c, d are left as variables.

Dually, if $i = 1$ is correct, there is a short proof of $\neg A_0(P, g, X, Y, a, b)$.

These short proofs can be found by the automatizability algorithm. This Frege is proof is evidence that the bit 0 (resp., 1) is encoded by $\langle P, g, X, Y \rangle$.

This algorithm would solve the Diffie-Hellman cryptographic scheme polynomial time and this is known to allow factorization of Blum integers in probabilistic polynomial time or with polynomial-size circuits [Biham-Boneh-Reingold'97]. qed.

Effective Craig Interpolation

The above proof used a kind of Craig interpolation property. Namely, it would be sufficient to assume that, if we are given a Frege proof of

$$\neg A(\vec{q}) \vee \neg B(\vec{r}),$$

then we can obtain in polynomial time either a proof of $\neg A(\vec{q})$ or a proof of $\neg B(\vec{r})$.

Some proof systems are known to have polynomial time Craig interpolation:

Resolution, cut-free LK, cutting planes,
intuitionistic propositional logic, nullstellensatz.

Often, polynomial time Craig interpolation properties leads to lower bounds on proof size.

An "anti-cut-elimination" theorem.

However, assuming Blum integer factorization is hard, Frege systems do not have feasible Craig interpolation. In fact, this implies an "anti-cut-elimination" or "anti-subformula" property for propositional logic:

Thm. Assume the non-feasibility of Blum integer factorization. Then, there is no feasible algorithm for transforming a Frege proof of

$$\neg A(\vec{q}) \vee \neg B(\vec{r}),$$

into a Frege proof of either $\neg A(\vec{q})$ or $\neg B(\vec{r})$.

As a corollary, there cannot be a polynomial time algorithm for creating proofs that enjoy an analogue of the subformula property (in terms of keeping $\vec{q}$ and $\vec{r}$ separated).

Some “big” open problems on the frontier of research

- Lower bounds on constant-depth Frege systems with counting gates.
- Separate the depth d Frege systems from the depth $d+1$ Frege systems with respect to lengths of proofs of low depth tautologies.
- Superpolynomial lower bounds on proofs in intuitionistic propositional logic.
- Better understanding of automatizability and interpolation.
- Better proof search algorithms for resolution and stronger systems.

Proof Complexity and Computational Hardness

II. Bounded Arithmetic and Propositional Complexity

Sam Buss
Department of Mathematics
U.C. San Diego

Swansea, Wales
July 4, 2006

Lecture II. Talk Outline

- Weak induction axioms and bounded arithmetic.
- Translations to propositional proofs.
- The witnessing theorems for S_2^1 and T_2^1.

Bounded Arithmetic Theories

Language: $0,$, S, $+$, $\cdot$, $\#$, $\lfloor \frac{1}{2}x \rfloor$, $|x|$, $\leq$, Gödel β function, $\langle\rangle$, $*$ (last three for sequence coding operations).
Smash: $x\#y = 2^{|x|\cdot|y|}$ - polynomial growth rate.

Bounded Quantifiers: $(\forall x \leq t)$, $(\exists x \leq t)$.

Sharply Bounded Quantifiers: $(\forall x \leq |t|)$, $(\exists x \leq |t|)$.
Intuition: sharply bounded quantifiers are feasible (poly time) quantifiers.

Σ_i^b**- and** Π_i^b**-formulas** . Count alternations of bounded quantifiers, ignoring sharply bounded quantifiers.
The sets definable by Σ_i^p-formulas are precisely the Σ_i^b-predicates.

The most important case is $i = 1$.
Here Σ_1^b formulas define exactly the NP-predicates.

Def'n. Theories S_2^1 and T_2^i are arithmetic are defined with axioms defining the non-logical symbols (the "basic" axioms) and with

Σ_i^b-**PIND induction axioms** For S_2^i:

$$A(0) \land (\forall x)(A(\lfloor \tfrac{1}{2}x \rfloor) \to A(x)) \to (\forall x)A(x).$$

Σ_i^b-**IND induction axioms** For T_2^i:

$$A(0) \land (\forall x)(A(x) \to A(x+1)) \to (\forall x)A(x).$$

$S_2 = \cup_i S_2^i$, and $T_2 = \cup_i T_2^i$.

Intuition: PIND is a "feasible" (poly time) induction scheme.

All these theories can Σ_1^b-define the polynomial time functions, so w.l.o.g., all polynomial time functions may be added to the theories.

Thm: $S_2^1 \subseteq T_2^1 \preccurlyeq S_2^2 \subseteq T_2^2 \preccurlyeq S_2^3 \subseteq \cdots$.

The Witnessing Theorems for Bounded Arithmetic Fragments

Def'n. A function is Σ_1^b-defined by a theory R provided $R \vdash \forall x(\exists y \leq t)A_f(x,y)$, where A_f is a function defining the graph of f. W.l.o.g., $R \vdash (\forall x)(\exists! y)A_f(x,y)$.

Fact. For R one of the theories S_2^i, T_2^i, $i \geq 1$, the function f may be added as a defined function symbol and used freely in induction axioms.

"Main Thm" for S_2^1, T_2^1.

- The Σ_1^b-definable functions of S_2^1 are precisely the polynomial time functions. [Buss]
- The Σ_1^b-definable functions of T_2^1 are precisely the PLS functions. [Buss-Krajíček]

Analogous theorems hold for $i > 1$ for S_2^i and T_2^i for computability at higher levels of the polynomial time hierarchy.

The intuition for the Witnessing Theorem for S^1_2 is that Σ^b_1-PIND axioms can be "unwound" in polynomial time. Some care is needed to define the notion of witnessing correctly, since the induction is on NP-predicates.

A similar intuition applies to T^1_2 and PLS.

However, we present a proof based on the Paris-Wilkie translation from bounded arithmetic to constant-depth propositional proofs.

Remark: There is a second important method of translating from bounded arithmetic to propositional logic, due to Cook. — This is not covered in this talk.

Some simplifying technical conditions

A formula is *form restricted* Σ_i^b if it is:

$$(\exists y_1 \le t_1)(\forall y_2 \le t_2) \cdots (Qy_i \le t_i)(\overline{Q}z \le |r|)B,$$

where B is quantifier-free. Quantifiers alternate between $\exists$ and $\forall$. Every Σ_i^b-formula is equivalent to a form restricted one, provably in S_2^i.

Restricted by parameter variables. Let P be a proof. The free variables in the endsequent, $\vec{a}$, are called *parameter variables*. A proof is *restricted by parameter variables* iff
(a) every quantifier is bounded by a term that involves only parameter variables,
(b) every induction term involves only parameter variables. and
(c) every sequent which contains a non-parameter b contains a formula $b \le t(\vec{a})$ in its antecedent.

Theorem 1. *Let R be S_2^i or T_2^i, $i \geq 1$. If A is a form restricted Σ_i^b-formula and $R \vdash A$, then there is an R-proof of A which is restricted by parameter variables and in which every formula is form restricted Σ_i^b.*

Such proofs are called *restricted*-Σ_i^b. These proofs are conveniently formed for translation into propositional logic.

Pf: Uses free-cut elimination heavily.

Constant depth propositional LK proofs

Syntax: Tait-style calculus. Variables: p. Literals: p, $\overline{p}$.

Unbounded fanin OR's and AND's: $\bigvee$ and $\bigwedge$.

A *cedent* Γ is set of formulas; intended meaning is the disjunction $\bigvee \Gamma$.

Axioms: *Neg:* $p, \overline{p}$ *Taut:* Γ , where Γ is a tautology.

Rules of inference:

$$\bigvee\text{:}\ \frac{\Gamma, \varphi_{i_0}}{\Gamma, \bigvee_{i\in\mathcal{I}} \varphi_i}\text{, where } i_0 \in \mathcal{I} \qquad \bigwedge\text{:}\ \frac{\Gamma, \varphi_i \quad \text{for all } i \in \mathcal{I}}{\Gamma, \bigwedge_{i\in\mathcal{I}} \varphi_i}$$

$$\textit{Weakening:}\ \frac{\Gamma}{\Gamma, \Delta} \qquad \textit{Cut:}\ \frac{\Gamma, \varphi \quad \Gamma, \overline{\varphi}}{\Gamma}$$

Σ'-depth of LK formulas and proofs

Definition Let S be a proof size parameter (size upper bound). The formulas that have Σ'-depth d *with respect to* S are inductively defined as follows:

a. If φ has size $\leq \log S$, then φ has Σ'-depth 0.

b. If each φ_i has Σ'-depth d, then $\bigvee_{i\in\mathcal{I}} \varphi_i$ and $\bigwedge_{i\in\mathcal{I}} \varphi_i$ have Σ'-depth $(d+1)$.

Definition Let S be a size parameter. An LK-proof P is a Σ'-depth d proof of size S provided:

a. P has $\leq S$ symbols,

b. Every formula in P has Σ'-depth d,

c. Every *Taut* axiom has size at most $\log S$. (Only small tautologies allowed).

Similar definitions: Krajíček['94] of Σ-depth; Beckmann-Buss['03] of Θ-depth.

Conversion from S^i_2, T^i_2 to LK

Let $d \geq 1$ and R be one of S^d_2 or T^d_2. Suppose $A(x)$ is form restricted Σ^b_d and $R \vdash A$. We describe how to transform a restricted proof of A into a Σ'-depth d LK proof. W.l.o.g., x is the only parameter variable.

First step: choose an arbitrary value $n \in \mathbb{N}$. The translation $[\![A]\!]_n$ is a propositional formula stating that $A(x)$ is true for all x such that $|x| \leq n$. The free variables of $[\![A]\!]_n$ are variables $p_{x,i}$ representing the i-th bit of the binary representation of x.

For quantifier-free formulas φ, the formula $[\![\varphi]\!]$ is defined with any polynomial size formula that expresses the value of φ. (All function and relation symbols are describable with polynomial size formulas.) Because we have the *Taut* axioms, the choice of translation formula $[\![\varphi]\!]$ is unimportant. Note $[\![\varphi]\!]$ has size only $m^{O(1)}$ if the free variables of φ are integers of length $\leq m$. We will have $m = n^{O(1)}$ and $m < \log S(n)$.

Consider a sharply bounded formula $(\forall y \le |s|)B$ or $(\exists y \le |s|)B$.
Because the term s contains only parameter variables as variables, and since the parameter variables have at most n bits, we can find a bound $n_y = n^{O(1)}$ such that $|s| \le n_y$. Then,

$$[\![(\forall y \le |s|)B]\!] \;=\; \bigwedge_{i=0}^{n_y} [\![y \le |s| \to B]\!]/(y \mapsto i). \tag{1}$$

The notation "$\psi/(y \mapsto i)$" means replace each $p_{y,j}$ by the (constant) jth bit of the integer i. $[\![(\forall y \le |s|)B]\!]$ has size only $n^{O(1)}$. Thus, it has Σ'-depth 0 for suitable $S(n) = 2^{n^{O(1)}}$.

General bounded quantifiers translated by exactly the same construction, but have bigger size: $2^{n^{O(1)}}$.

A Σ^b_d-formula becomes a Σ'-depth d formula for suitable $S(n) = 2^{n^{O(1)}}$.

To translate a cedent Γ, view it as a Tait-style cedent by moving all formulas to right of the sequent. All non-parameter variables $y_1, \ldots, y_k$ are restricted by parameter variables. So $|y_j| \le n_j$ for some $n_j = n^{O(1)}$.

Γ is translated into a set of cedents, one cedent for each choice of $i_1, \ldots, i_k$ with each $|i_j| < n_j$. The cedents are just

$$[\![\Gamma]\!]/(y_1 \mapsto i_1, \ldots, y_k \mapsto i_k),$$

where the translation is applied individually to each formula. Note: the only variables left are $p_{x,i}$.

As the next theorem states, the translated cedents Γ can be pieced together into a valid proof.

Paris-Wilkie translation theorem

Theorem 2. *Let* $i \geq 1$. *Suppose* $A(x) \in \Sigma^b_i$. *Let* $[\![A]\!]_n$ *denote the propositional translation of* A; $[\![A]\!]_n$ *has free variables* $p_{x,i}$, *for* $i < n$.

a. *Suppose* $S^i_2 \vdash A$. *Then there is a function* $S(n) = 2^{n^{O(1)}}$ *such that, for all* n, $[\![A]\!]_n$ *has a* Σ'*-depth* i *proof of size* $S(n)$. *This proof*

- **i.** *has height* $O(\log\log S(n))$, *and*
- **ii.** *contains only* $O(1)$ *many formulas in each cedent.*

b. *Suppose* $T^i_2 \vdash A$. *Then there is a function* $S(n) = 2^{n^{O(1)}}$ *such that, for all* n, $[\![A]\!]_n$ *has a* Σ'*-depth* i *proof of size* $S(n)$. *This proof*

- **i.** *has height* $O(\log S(n))$, *and*
- **ii.** *contains only* $O(1)$ *many formulas per cedent.*

[Rk: The original W-P translation applies to $S^i_2(\alpha)$ and $T^i_2(\alpha)$.]

One case of the proof: translation of $\wedge$:right inference

An $\wedge$:*right* inference

$$\frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}$$

translates to

$$\dfrac{[\![\Gamma]\!], [\![\varphi]\!] \qquad \dfrac{[\![\Gamma]\!], [\![\psi]\!] \qquad \dfrac{[\![\psi \wedge \varphi]\!], \overline{[\![\varphi]\!]}, \overline{[\![\psi]\!]}}{[\![\Gamma]\!], [\![\psi \wedge \varphi]\!], \overline{[\![\varphi]\!]}, \overline{[\![\psi]\!]}}\,\textit{Weakening}}{[\![\Gamma]\!], [\![\psi \wedge \varphi]\!], \overline{[\![\varphi]\!]}}\,\textit{Cut}}{[\![\Gamma]\!], [\![\psi \wedge \varphi]\!]}\,\textit{Cut}$$

Note that the upper right sequent is a *Taut* axiom.

Another case of the proof: induction rule

Consider an induction inference in P. This translates into m *Cut* inferences in the LK proof, where m is the "length" of the induction. By balancing the tree of cuts, the height (maximal number of cedents along any branch) is only $O(\log m)$. (The induction bound t involves only parameter variables.)

If R is S^i_2, the induction inference translates into $|t| = n^{O(1)}$ many cuts, so the height is $O(\log n)$.

If R is T^i_2, the induction inference translates into $t = 2^{n^{O(1)}}$ many cuts, so the height is $O(n^{O(1)})$. $\square$

Important fact: Although the LK-proofs given by Theorem 2 are exponentially big, they are also polynomial time uniform.

Main Theorem for S^1_2

Theorem 3. (Buss ['85]) *Suppose* $A(x,y) \in \Sigma^b_1$ *and that* S^1_2 *proves* $(\forall x)(\exists y)A(x,y)$. *Then there is a polynomial time function* $f(x) = y$ *such that for all* $x \in \mathbb{N}$, $A(x, f(x))$ *holds.*

Proof By Parikh, $S^1_2 \vdash (\exists y \leq s(x))A(x,y)$. x is the parameter variable. Applying Theorem 2(a) yields a Σ'-depth 1 proof; adding a *Cut* to the end of this proof turns the proof into a refutation R of

$$[\![\forall y \leq s(x))\neg A(x,y)]\!]. \tag{2}$$

We give a polynomial time procedure that is has as input a particular value for x, and traverses the refutation R until it arrives at a false initial cedent. Of necessity this false initial cedent is the cedent (2), and when it is reached, the procedure will know a value y that falsifies the cedent. This value for y will be the value of $f(x)$.

The polynomial time procedure acts as follows: it starts at the root of the proof and traverses the proof upward, backtracking as needed as described below. The root is labeled with the empty sequent. At each stage, the procedure is at some cedent Γ in the proof that it believes to be false. In particular, every Σ'-depth 0 formula in Γ is *False*. (Recall that the variables $p_{x,i}$ are the only variables in R, and the procedure has values for these.) Furthermore, for any formula in Γ which is a conjunction of Σ'-depth 0 formulas, a particular conjunct is known to be false. For the formulas which are a disjunction of Σ'-depth 1 formulas, the procedure does not know for sure that they are false, it merely tentatively assumes they are false.

At the beginning, the procedure is at the endsequent of R, which is the empty cedent.

Other cases are *Cut* inference, $\bigwedge$ inference, and $\bigvee$ inference....

If the procedure is at the lower cedent of a cut inference

$$\frac{\Gamma, \overline{\varphi} \qquad \Gamma, \varphi}{\Gamma}$$

If φ is Σ'-depth 0, then it can be evaluated as being either *True* or *False*. If it is true, the procedure proceeds to the left upper cedent, otherwise, it proceeds to the right upper cedent. Otherwise, φ is w.l.o.g. a disjunction, and the algorithm proceeds to the right upper cedent.

If the procedure is at the lower cedent of a $\bigwedge$-inference:

$$\frac{\Gamma, \psi_i \qquad \text{, for } i \in \mathcal{I}}{\Gamma, \bigwedge_{i \in \mathcal{I}} \psi_i}$$

the algorithm acts as follows. By assumption, the procedure knows a value i_0 such that the conjunct ψ_{i_0} is false. The algorithm proceeds to the upper cedent Γ, ψ_{i_0} where $i = i_0$.

If the procedure is at the lower cedent of a $\bigvee$-inference:

$$\frac{\Gamma, \psi_{i_0}}{\Gamma, \bigvee_{i \in \mathcal{I}} \psi_i}$$

the algorithm acts as follows. If ψ_{i_0} is false, it proceeds to the upper cedent. However, if it is true, the algorithm has discovered a disjunct of $\varphi = \bigvee_{i \in \mathcal{I}} \psi_i$ which is true, contradicting the tentative assumption that φ was false. The procedure then backtracks down the path towards the root until it finds the *Cut* inference where the formula φ was added to the cedent. It then proceeds to the other upper cedent of the *Cut*, and saves the information about which conjunct of $\overline{\varphi}$ is false.

The run time is $O(n^{O(1)})$, because there are only this many *Cut*'s. It thus can terminate only at the cedent (2). When it reaches this, it knows a value for y that falsifies it. This value of y satisfies $A(x, y)$. $\square$

The Main Theorem for T_2^i

PLS $=$ Polynomial Local Search [Johnson, Papadimitriou, Yanakakis, '88].

A PLS problem is a multivalued function $f(x)$ defined by:

- A polynomial bound, $t(x) = 2^{|x|^{O(1)}}$, on possible solutions.
- A polynomial time cost function $c(s,x)$ – cost of solution s.
- A polynomial time function $N(s,x)$.

such that $f(x) = y$ where $y < t$ and $c(y) \le c(N(s,x),x)$.

Theorem 4. (Buss, Krajíček, '94]) *Suppose* $A(x,y) \in \Sigma_1^b$ *and that* T_2^1 *proves* $(\forall x)(\exists y)A(x,y)$. *Then there is a Polynomial Local Search (*PLS*) function* $f(x) = y$ *such that for all* $x \in \mathbb{N}$, $A(x, f(x))$ *holds.*

The proof is identical to before, based on exactly the same procedure. Now the procedure may need $2^{n^{O(1)}}$ steps, instead of $n^{O(1)}$. Use the position in the proof to define a decreasing cost function. $\square$

Remark: Theorems 3 and 4 both hold if all true Π_1^b-formulas are added as axioms (no change to proof needed).

Proof Complexity and Computational Hardness

III. On the (Lack of) Progress Towards "P versus NP"

Sam Buss
Department of Mathematics
U.C. San Diego

Swansea, Wales
July 5, 2006

Lecture III. Talk Outline

- Logical intuitions for $P \neq NP$.
- Oracle results.
- Connections with bounded arithmetic.
- Randomization, one-way hardness.
- The state of the art: Independence via inability to count.

A *logical* reason for $P \neq NP$?

[Rk: "logical" reason as compared to "combinatorial" reason.]

G. Kreisel, Symp. on Automatic Deduction, LNM #125, 1968:

> "Suppose we ... have a proof system; ... the 'faith' is that in a natural way this will yield a *feasible* proof procedure for feasible theorems.
>
> "*Conjecture:* Under reasonable conditions on feasibility, there is an analogue to Gödel's second incompleteness theorem, that is the article of faith above is unjustified."

Another Expert Opinion…

Here is R. Solovay's sought for proof of "$P \neq NP$" :

Proof:

Another Expert Opinion...

Here is R. Solovay's sought for proof of "$P \neq NP$" :

Proof: Let M be a nonstandard model of $Th(\mathbb{N})$.

Another Expert Opinion...

Here is R. Solovay's sought for proof of "$P \neq NP$":

Proof: Let M be a nonstandard model of $Th(\mathbb{N})$.

Therefore $P \neq NP$. **q.e.d.**

Another Expert Opinion...

Here is R. Solovay's sought for proof of "$P \neq NP$" :

Proof: Let M be a nonstandard model of $Th(\mathbb{N})$.

(fill in the details of proof here)

Therefore $P \neq NP$. **q.e.d.**

What "logical" reasons are there for believing $P{\neq}NP$? One is that $P{=}NP$ would make the practice of mathematics too easy: The process of proof search could be automated (automatized) by formalizing mathematical questions completely and then blindly search for proofs of conjectured statements. If $P = NP$, this process could succeed whenever proofs are not too large. This would be a major change in the practice of mathematics.

Gödel [1956 letter to von Neumann]: "... consequences of the greatest importance. ... The mental work of a mathematician concerning Yes-No questions could be completely replaced by a machine."
See [Buss 1995, in Feas. Math. II] for a detailed discussion.

A related objection is that it would mean mathematics would become completely formal, with little room left for human intuition and understanding.

The conservative viewpoint is that this would be undesirable and thus it is deemed unlikely.

Off-the-record comments go here.

From Kreisel's suggestion, one might think to try diagonal or self-referential statements. E.g.,

> "I do not have a polynomial size proof in theory R".
>
> "I do not have a feasible proof."
>
> "My propositional translation does not have a poly size Frege proof."

Etc.

These self-referential statements can be formulated, but none of these seem to imply anything about $P \neq NP$.

Another idea would be to find co-NP predicates that lack polynomial size proofs.

Example: **Partial Consistency Statements.** Let R be a theory, let $k > 0$. Define

$Con(k, R) \Leftrightarrow$ "There is no R-proof of a contradiction of length $\leq k$ symbols."

However, we have the following theorems:

- $PA \overset{poly}{\vdash} Con(k, PA)$. [H. Friedman, Pudlák]
- $S^1_2 \overset{poly}{\vdash} Con(k, S^1_2)$.
- $e\mathcal{F} \overset{poly}{\vdash} Con(k, e\mathcal{F})$. [Cook.]
- $\mathcal{F} \overset{poly}{\vdash} Con(k, \mathcal{F})$. [Buss.]

The idea behind the above proofs of partial consistency statement is that it is possible to define polynomial size partial truth definitions (that is, truth definitions for formulas of length $\leq k$).

[Buss'85] suggests a different approach: Let J_R be the **jump** of the theory R and be defined to be the theory R plus a truth predicate for R-formulas and allowing the truth definition to be used in induction formulas.

Question: $S_2^1 \overset{poly}{\vdash} Con(k, J_{S_2^1})$?

If not, S_2^1 cannot prove $P =$ co-NP.

Related Question: $S_2^1 \overset{poly}{\vdash} Con(k, ZF)$?

Independence of P vs NP?

Traditional "independence" results include:

a. Oracle results [Baker-Gill-Solovay]

b. Natural proofs. [Razborov-Rudich]

Most present day proof methods for lower bounds fall into one of these categories.

What about independence from a formal theory, e.g., bounded arithmetic? Potentially could give concrete measurement of the hardness of proving $P \neq NP$.

Independence of P vs NP?

There are several ways to formalize whether bounded arithmetic can prove $P = NP$, or the collapse of the polynomial hierarchy ($PH \downarrow$).

(1). Does S_2 prove each bounded formula is expressible in Σ_i^b, for some fixed i?
Notation: $S_2 \vdash (PH \downarrow)$

Thm.
(a) [KPT] If $S_2^i \prec_{\Sigma_i^b} T_2^i$ then $PH \downarrow$. Thus, if $S_2 \downarrow$, then $PH \downarrow$.
(b) [B,Z] $S_2 \vdash (PH \downarrow)$ iff $S_2 \downarrow$.

Notation: $S_2 \downarrow$ means S_2 is finitely axiomatized, or equivalently, that the hierarchy S_2^i collapses.

Unfortunately we have no idea how to show $\neg(S_2 \downarrow)$. If we *could*, this would say something about the logical difficulty of proving $P \neq NP$.

(2) Another approach: Show S_2 cannot prove super-polynomial lower bounds on circuit size....

One simple idea for independence from S^1_2: Let C denote a function computed by a Boolean circuit, φ a proposition formula, τ a truth assignment. Define

$$SAT(\varphi) :\Leftrightarrow (\exists \tau \leq \varphi) TRU(\varphi, \tau).$$

Assuming $P \neq NP$ is it possible to prove that

$$S^1_2 \nvdash \forall C \exists \varphi \exists \tau [TRU(\varphi, \tau) \wedge \neg SAT(\varphi, C(\varphi))]\,?$$

Motivation: given C, it might be hard to find φ and τ; contradicting the polynomial time witnessing theorem for S^1_2.

But this is *not* likely to work. Standard cryptographic assumptions about the existence of pseudorandom number generators imply that it is easy to find hard instances of satisfiability.

Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a hard one-way function.

Let $\varphi(x,y)$ express $f(x) = y$. For a fixed $y_0 = f(x_0)$, $\varphi(x) := \varphi(x, y_0)$ is satisfiable (by $x = x_0$).

But under commonly accepted cryptographic conjectures there are polynomial time computable functions f for which there is no feasible algorithm (or polynomial size circuit) that computes x from y.

Given C, chose x_0 at random, and set $\varphi = \varphi(x_0, y)$ and $\tau = x_0$. This witnesses the formula

$$\exists\varphi\exists\tau[TRU(\varphi,\tau) \wedge \neg SAT(\varphi, C(\varphi))].$$

A related cryptographic assumption is:

Strong pseudo-random number generator (SPRNG) conjecture. There are polynomial time computable functions $f : \{0,1\}^n \to \{0,1\}^{2n}$ such that the values $f(x)$ are computationally indistinguishable from random values from $\{0,1\}^n$.

Thm [Razborov] Fix any polynomial hierarchy predicate $A(x)$. Assume that a strong pseudo-random number generator (SPRNG) conjecture holds. Then $S^2_2(\alpha)$ cannot prove any superpolynomial lower bound on the size of a circuit (coded by α) for $A(x)$.

The predicate α serves to encode a circuit for $A(x)$. Note $S^2_2(\alpha)$ can use IND induction on the size of the circuit, but not its Gödel number.

Proof idea was to the use conservativity of $S^2_2(\alpha)$ over $T^1_2(\alpha)$, witnessing in PLS, interpolation, and then natural proof independence of Razborov-Rudich which depends on SPRNG.

However, a subsequent idea by Razborov and an independence result of Krajíček give a much strong result (and a less satisfying result).

Thm [essentially Razborov, Krajíček] (No SPRNG assumption.) $S^2_2(\alpha)$ cannot prove any superpolynomial lower bound on the size of a circuit for $A(x)$.

Proof idea:

Part 1. **Claim:** [K] Let $S^2_2(h) + \neg PHP(h)$ be the theory S^2_2 augmented with a new function symbol $h(x)$ plus an axiom that states $h(x)$ is a one-one mapping from $[2^n]$ onto $[n^{\omega(1)}]$. Then $S^2_2(h)$ is consistent. This is true even if there a function symbol for h^{-1} as well.

Pf. Based on the conservativity of S^2_2 over T^1_2 and the PLS witnessing theorem for T^1_2. A diagonal type of argument that thwarts any PLS algorithm seeking an explicit violation of the pigeonhole principle.

Part 2, [R] Use the inability to count. Given the trivial CNF circuit for satisfiability, which is exponentially big, of size 2^n, use h to convert it to a circuit of size $n^{\omega(1)}$. Hence it is consistent with $S_2^2(\alpha)$ that there are circuits of size $n^{\omega(1)}$ for satisfiability.

The size $n^{\omega(1)}$ is barely superpolynomial and can be smaller than any particular superpolynomial bound.

[This proof idea was first used by Razborov to prove the independence of superpolynomial circuit size from resolution.]

One last idea [Krajíček, ABRW]:

Let R be a proof system, e.g., R is the Frege proof system.

Let $f : \{0,1\}^n \to \{0,1\}^{2n}$ be a suitably strong pseudo-random number generator.

Conjecture. Choose y at random. Then

$$\neg(\exists x)(f(x) = y)$$

does not (always) have polynomial size proofs.

Conclusions

Our best lower bound results for separating P from NP amount to exploiting theories that are so weak that they cannot count well enough to separate exponential size from near polynomial time.

One more idea: return to Razborov-Rudich natural proofs. These in effect say that a strong version of $P \neq NP$ (the SPRNG conjecture) implies the hardness of finding proof that $P \neq NP$. Perhaps this can be extended to give better results. (??? — One can always hope!)