Can the P vs NP question be independent of the axioms of mathematical reasoning?

Shai Ben-David

School of Computer Science

University of Waterloo

What is this talk NOT about?

- *Disclaimer;* I am not going to tell you if SAT can be solved in polytime. Nor am I going to provide any clues towards the answer.

What is this talk about?

- When faced with a hard math problem, there is always the temptation to think:

"maybe this problem is inherently irresolvable. Maybe the reason we fail to find the answer is not our lack of wisdom, but rather that no such answer (=proof) exists?"

The goal of this research

- Why do we fail to resolve basic computational complexity questions?

- Could it be that the P vs NP issue is "*un-resolvable*"?

- More concretely:

  Is it likely that the tools of our mathematical reasoning are inherently too weak to determine relationships between complexity classes?

- Should we direct our efforts to answering these logic oriented questions, rather than struggle with the computational complexity issues themselves?

Insolvability or "Independence" results
Background

- ***Hilbert's Program (1920)*** – develop formal methods that will resolve all mathematical questions.

- ***Godel's Incompleteness results (1931)*** - Hilbert's plan is bound to fail; Every reasonable mathematical framework has irresolvable questions.

  Terminology: *A statement s is independent of a theory T, if T cannot prove s and T cannot prove ¬s.*

  Godel's Incompleteness Theorem:
  If T is a sound and consistent theory then Con(T) is independent of T.
  (In particular, any consistent theory cannot prove its own consistency).

- Is it relevant to "real" mathematical questions?
(or are all independent statements "self-referential" or logic-oriented)?

Towards "Real" independence results
Background –Set Theory and Arithmetic

- ***Set Theory - ZFC (Zermelo, Fraelkel, Skolem 1908-1922)*** –
  a formal theory that defines what is a mathematical proof.
  All of standard mathematics can be based on this axiom system.

- ***Peano Arithmetic - PA (1908 ?)*** – A formal theory for reasoning about natural numbers.
  Equivalent to ZFC minus the axiom stating that there exists an infinite set.

- ZFC proves Con(PA), so it is stronger than PA even w.r.t. statements about natural numbers.

“Real” independence results
Cohen’s Forcing technique

- ***Paul Cohen (1960) –*** Introduces the forcing techniques and proves the first independence of Set Theory result for a “real” question. Namely, *the continuum hypothesis is independent of ZFC*.

- ***More independence-of-Set-Theory results –***
  in cardinal arithmetic, infinite combinatorics, group theory, topology, functional analysis and even machine learning.

Independence results for computational complexity

- Independence of oracle classes w.r.t. any theory

  Hartmanis-Hopcroft (1976) :

  Given any theory T, construct a Turing machine M, s.t.
  “$P^{L(M)}$ vs $NP^{L(M)}$” *is independent of T.*

- Independence w.r.t. weak fragments of PA:
  - Artificial fragments ***(DeMillo- Lipton 1979, Sazanov 1980).***
  - Bounded Arithmetic and conditional independence results ***(Razborov 1995).***

- Limitations of proof techniques – Relativising proofs (***Baker Gill Solovay, 1975***), Natural proofs ***(Razborov-Rudich 1997),*** Algebrizing proofs ***(Aaronson-Wigderson, 2008) .***

Can we prove the independence of P vs NP from set theory?

There are inherent limitations to forcing:

in particular, forcing cannot show the independence of any statement that involves only natural numbers (or finite sets).

P vs *NP* is such a statement:

"For every code p of a Turing machine,

and every k,

there is a propositional formula x

so that the machine that runs p for $|x|^k$ steps

fails to determine the satisfiability of x."

What can we hope to prove?

Non-provability w.r.t. PA

- The weaker a theory, the easier it should be to find statements that it fails to prove.

- Independence w.r.t. PA should be quite

  satisfactory, since there is no reason to assume that one needs the *existence of an infinite set* to resolve the complexity of SAT.

Independence from PA of real mathematical statements

- ***Paris Harrington (1977)*** – a version of the finite Ramsey theorem is true (i.e. provable from ZFC), but cannot be proven from PA.
- Similar results proven later for a variety of statements about natural numbers (*Hercules and the Hydra*, *Goodstein sequences* and more).
- The structure of the PH statement is similar that of P≠NP:

“for all x there exist y such that φ(x,y)”.

(where ϕ(x,y) is quantifier-free)

The conclusions of this work

1. If SAT can be solved by an "almost polynomial" time algorithm then T fails to prove P ≠ NP.

(This holds for any theory T,

where the precise meaning of

"almost polynomial" depends on T).

The conclusions of this work

2. If T is sufficiently strong then the reverse statement holds as well. Namely, the only possible reason for the failure of T to prove P ≠ NP is that SAT can be solved by an almost polytime algorithm.

- Loosely stated –

  proving that mathematics cannot prove P≠NP, amounts to proving that P≈NP.

What do we mean by "almost polynomial"?

We would consider algorithms that run in time $n^{f(n)}$ where, $f(n)$ grows very very slowly.

In other words, the running time is constant on huge stretches on input lengths.

Such functions, $f(n)$, are the inverses of fast growing functions.

The basic tool –
Fast growing functions

The Wainer Hierarchy:

-

- Note that the Ackerman function is F_ω in this sequence.

Fast growing functions - example

- The Wainer Hierarchy:

Very very fast growing functions

- Let ε_0 be the first ordinal α s.t. $\omega^\alpha=\alpha$.

  This is the limit of the sequence

  (this is ordinal exponentiation, so all these ordinal are countable).

We will be interested in ***$F_{\varepsilon 0}$***

Approximation rate and complexity

- The approximation rate of a language by a complexity class:

For a language L and a complexity class C,

let M_1, M_2, … be some canonical enumeration of C,

$$R_L^C(i) = \max_{j<i}\{\min\{|x| : L(x) \neq M_j(x)\}\}$$

Note: R_L^C is a total function if and only of $L \notin C$

Furthermore, the faster R_L^C grows, the closer is L to the class C.

SAT and fast growing functions

Let $M_1, M_2, \ldots$ enumerate of all *P*-time machines such that

- the mapping from i to (a code of) M_i can be computed in linear time, and
- for all i, the running time of M_i is bounded by $n^{\log(i)}$

Then, for any easily computable function g that bounds R^{-1}

(where R is the approximation rate of SAT by P),

SAT is in $$\mathrm{DTIME}(n^{1+\log(g(n))} \times g(n))$$

Corollary: *If the approximation rate of SAT by P is a fast growing function, then SAT has "almost-polynomial" algorithms.*

Where are we at this point?

- At this point, we have two ingredients of our line of reasoning – fast growing functions, and their relation to SAT and the class P.
- *The next (and final) ingredient we need, is relating it to provability (and in particular,*

  to the provability of $P \neq NP$).

Provably recursive functions

- *A function* is ***provably recursive*** *w.r.t. a theory T,* if it is recursive, and the theory can prove that it is a total function.

I.e., the function is computable by some algorithm A s.t.

T proves that A halts on every input.

- *The provably recursive functions w.r.t. PA and PA_1*

are the same (hence we'll just call them "provably recursive").

Fast growing functions and provability

- ***Wainer's theorem*** :

  1. *F_α is provably recursive for every $\alpha < \varepsilon_0$*
  2. *If a function is provably recursive, then it is dominated by F_α for some $\alpha < \varepsilon_0$*

- ***Corollary***:

  If a function grows very fast (i.e., no F_α dominates it), then PA, as well PA_1 cannot prove that it is total.

The relation to independence results

- The Paris Harrington independent statement

  "for all **x** there is **y** such that **ϕ(x,y)**"

  can be viewed as stating the totality of some recursive function.

- The proof of independence from **PA** amounts to showing that this function grows so fast that it is not dominated by the Wainer functions.

  (This basic structure repeats in most other

  Independence-w.r.t.-**PA** proofs)

Conclusion - a sufficient condition for the non-provability of P≠NP

- If the approximation rate of *SAT* by *P* is a very fast growing function, then

  PA cannot prove that *P≠NP*.

- Corollary: If *SAT* can be solved by almost polynomial algorithms then *PA* cannot prove that *P≠NP*.
- In fact, the only effect of *PA* on this result is for *quantifying* the meaning of "almost polynomial" algorithms.

Inverses of fast growing functions

- Let g be a monotone increasing function that is not dominated by the Wainer hierarchy.

  For every monotone provably recursive f,

  there are infinitely many n's such that

  "for every m between n and A(n), $g^{-1}(m) < f^{--1}(m)$

  and $f(m) > g(m)$"

  where A(n) is the Ackermann function

  (or any of your favorite fast growing p.r. functions)

- It follows that if R is fast growing, there is an

  algorithm for SAT whose run time is a fixed polynomial on infinitely many VERY long intervals, [f(m), f(m+1)).

The next step:

Showing that a fast growing R is the only potential reason for non-provability of P≠NP

Our Approach – Strengthening PA

- We argue that proving independence of *P vs NP* is almost equivalent to discovering the actual answer.

- The stronger the theory, the stronger the consequences of being independent of the theory.

- We consider a strong extension of *PA*, PA_1.

  In a sense, it is unrealistically strong – it is not a recursive theory.

- Yet, no currently known technique can separate the two.

  All independence-w.r.t.-*PA* results (of "real mathematical statements") are, in fact, independence-w.r.t.- PA_1

(more on PA_1 later in the talk).

The Theory PA_1

- A first order formula (in the language of arithmetic) is a π_1 formula if it has the form

$$\forall x \varphi(x)$$

  where ϕ has only bounded quantifiers.

- PA_1 is the proof system that has

  $PA \cup \{\Psi: \Psi$ is a π_1 formula that is true in the standard model of Arithmetic$\}$

  as its set of axioms.

Some properties of the theory PA_1

- It is not a recursive theory….

- Representation independence:

  If two (codes of) Turing machines compute the same language, then this equivalence is provable in PA_1 (in fact it is the minimal extension of PA with this property).

- If P=NP then PA_1 proves it

The necessary condition

Theorem: PA_1 proves $P \neq NP$ if and only if

for some $\alpha < \varepsilon_0$, F_α dominates the approximation rate of *SAT* by *P*.

Proof Idea (of the left-to-right direction):

If *R* is dominated by a provably recursive *F*, then $P \neq NP$ is equivalent to

"every *P* machine M_i fails to compute *SAT* on some input of length $< F(i)$"

which is a true π_1 formula.

Corollary: *If PA_1 fails to prove $P \neq NP$, then SAT has almost polytime algorithms.*

More on the significance of PA_1

- The generic way to prove that a theory *T* does not prove some statement Ψ, is to build a model for $T \cup \{\neg\Psi\}$.
- We do that, by starting with a model M for T, and constructing a sub-model $M' \subseteq M$

  s.t. $M' \models T \cup \{\neg\Psi\}$.
- In that case, *M'* satisfies the π_1 theory of *M*.

- *Applying this paradigms to models of PA, yields the independence of the statement Ψ from PA_1.*

The Bottom line

- If it is provable (by any method known

  today) that P≠NP is not provable in *PA*,

  then SAT is in DTIME($n^{g(n)}$) where g^{-1}

  is a very fast growing function (i.e., not dominated by the Wainer hierarchy).

Similar results for circuit complexity follow by these arguments

Related Open Questions

- Can *SAT* be easy for arbitrarily long stretches of inputs and yet by worst-case hard?
- Can we find a recursive sub-theory of **PA_1** that suffices for our result?

  (we mean a theory that we can prove is a subset of **PA_1**, not *PA+”P=NP”* …).