Abraham Ladha

Relativization

20.

First, we begin with a review of diagonalization. I will reprove the halting problem, but this time using a functional notation.

Let $\varphi_1, \varphi_2, \ldots$ be an enumeration of the recursive functions. (These correspond to Turing machines which are allowed to loop). We prove there is no total computable function h

$$h(x,y) = \begin{cases} 1 & \varphi_x(y) \text{ halts} \\ 0 & \varphi_x(y) \text{ loops} \end{cases}$$

Assume to the contrary h is total and computable. Consider the function d:

$$d(x) = \begin{cases} 1 & \text{if } h(x,x) = 0 \\ \text{loops} & \text{if } h(x,x) = 1 \end{cases}$$

certainly d exists since h does. d is a recursive function, so there must exist some i such that $d = \varphi_i$. Consider d on its own index i.

$$d(i) = 1 \iff h(i,i) = 0 \iff \varphi_i(i) \text{ loops} \iff d(i) \text{ loops}$$

$$d(i) = \text{loops} \iff h(i,i) = 1 \iff \varphi_i(i) \text{ halts} \iff d(i) \text{ halts.}$$

A contradiction. There cannot exist a total recursive function for h.

Now we prove a weaker form of the time-hierarchy-theorem. A hierarchy is like a ladder, we are able to prove that more asymptotic time gives more power. The strongest form of the theorem says

$$\text{TIME}(o(f(n)/\log f(n)) \subsetneq \text{TIME}(f(n))$$

Rather than worry about the tightness between the runs, we prove a weaker form of the theorem to demonstrate the same result. We show $\forall k$ $TIME(n^k) \subsetneq TIME(n^{k+1})$. The containment is obvious, so we show existence of a language decidable in time $O(n^{k+1})$ which is not decidable in time $O(n^k)$.

Let $M_0, M_1, \ldots$ be an enumeration of the $TIME(n^k)$-machines. Each M_i halts within n^k steps on an input of size n. Consider the following algorithm

D on input w_i:
compute $n = |w_i|$
Simulate M_i on w_i for n^k steps
if it accepts, reject
if it rejects, accept

Notice D on input w_i returns $(1 - M_i(w_i))$. So $\nexists\, i$ such that $L(D) = L(M_i)$ so $L(D) \notin TIME(n^k)$. Since it differs from every n^k-time machine, there is no n^k time algorithm to decide $L(D)$. What is the cost of simulation? Turns out this is complicated but we can safely upper bound. That simulation of M_i for a single step takes at most $O(n)$ steps for the simulator. Since $n^k n = n^{k+1}$, we conclude $L(D) \in TIME(n^{k+1})$.

I want to remark on both of these proofs. Both times we interacted with other machines but only in a black-box way. We simulated them to disagree with their output. We did nothing with M_i except run it, we did not deal with the internal mechanics of computation.

If we can separate the decidable from the undecidable and do infinite hierarchy of deterministic time classes. Could we use such techniques to separate P from NP?

The oracle of Delphi was like a witch or a shaman. You would bring her gifts and she would answer your questions. There would be no provided explanation.

An oracle machine has the same mysticism. It is a Turing machine with an additional tape. It writes down a query on this special tape and issues an instruction. Then the tape clears and all that is left is a 1 or 0. We formalize an oracle as a language A, and the oracle machine M^A. M^A can test membership to language A in unit time. Many natural things we have discussed appear to be representable in an oracle way. Nondeterminism was originally formulated like an oracle.

For a class C and language A, we let C^A be the languages decidable by C-machines with oracle access to A. For example consider the structure of P^{SAT}. Certainly any oracle machine can ignore its oracle, so $P \subseteq P^{SAT}$. Also notice that all of NP is polytime relative to SAT, since SAT is NP-complete under polynomial time reductions, $NP \subseteq P^{SAT}$. Certainly P^{SAT} is very different than P. We won't explain why, but NP^{SAT} is actually bigger than NP.

Relativization: We don't really care about comparing two classes, one with and the other without the oracle. We care about a relativized world. One in which every machine has oracle access. For some fixed A, what does the world look like? What is the relationships between L^A, P^A, NP^A, $PSPACE^A$, and so on. How does this relativized world differ from our own? For each fixed A, there exists an entire separate world with its own laws and rules and relationships. We can relate these worlds to our own.

We say a proof relativizes if you can copy paste it from our world with only minor modification

D on input w_i	D^A on input w_i
compute $n = \|w_i\|$	compute $n = \|w_i\|$
simulate M_i on w_i for n^k steps	simulate* M_i^A on w_i for n^k steps
if accepts, reject	if accepts, reject
if rejects, accept	if rejects, accept.

In our world, D simulates M_i. In the relativized world, D^A simulates M_i^A. If M_i^A makes an oracle call, D^A simulates this instruction of M_i^A by calling its own oracle. Here the "simulation" has this slight difference.

Could there exist a proof of diagonalization for $P \neq NP$? It might look like

enumerate P machines $M_0, M_1, \ldots$	enumerate P^A machines $M_0, M_1, \ldots$
D on input w_i	D^A on input w_i
simulate M_i on w_i	simulate* M_i^A on w_i
return opposite	return opposite.
somehow show $L(D) \in NP$	Somehow show $L(D^A) \in NP^A$.

If a proof holds in our world, we say it relativizes to all worlds. If $P \neq NP$ in our world, if is provable this way, then $\forall A$ $P^A \neq NP^A$. It holds in all relativized worlds. However, if there exist A such that $P^A = NP^A$, then there exists a world where $P^A = NP^A$, so there cannot exist a relativizing proof that $P \neq NP$. Similarly if $\exists B$, a relativized world where $P^B \neq NP^B$, then there cannot exist a relativizing proof that $P = NP$.

We show two oracles A, B such that

$$P^A = NP^A \qquad P^B \neq NP^B$$

Since there exists two worlds, one where $P=NP$, and one where $P \neq NP$. No proof of $P \stackrel{?}{=} NP$ in our world can generalize in this black box way. Our demonstration of contradictory oracles will prove that there is not a relativizing proof of P vs NP!!!

We choose A to elevate P^A, NP^A to the same class where nondeterminism gives no power, certainly $\forall A \; P^A \subseteq NP^A$ so we show for some A that $NP^A \subseteq P^A$. We will use space complexity. Let $A = TQBF$. Then

$$NP^A = NP^{TQBF} = NPSPACE = PSPACE \subseteq P^{TQBF} = P^A$$

(1) (2) (3)

relative to TQBF, every PSPACE ~~complete~~ language is decidable by nondeterministic polynomial time. The second containment holds by Savitch's theorem. The final one holds similarly to why $NP \subseteq P^{SAT}$.

Showing oracle B such that $P^B \neq NP^B$ will be much harder. Ironically, we will construct B by diagonalization. We want to show a language exists which could not be in P^B. How we will show it cannot be done in polynomial number of steps is for correctness, we will require an exponential number of oracle queries. So since each query takes unit time, an exponential number of queries implies that the machine to decide this language must take exponential time, and then could not have been in P^B.

Let L_B = set of strings such that there is some string of the same length in B. We don't say which string to require making 2^n queries.
$L_B = \{ w \mid \exists x \in A \text{ with } |x| = |w| \}$

Let $M_1^B, M_2^B, \ldots$ be an ordering of the machines of P^B. Lets even suppose they are weakly sorted to guarantee M_i^B halts in time n^i on all inputs.

We construct B in a sequence of stages ~~to stage~~ such that M_i^B does not decide L_B is ensured in stage i. At each stage, only a finite amount of strings have been decided to be in B and decided to not be in B.

Suppose we are at stage i. Let w be the largest string in B. Choose n such that $2^n > n^i$ and $n > |w|$. We will increase the knowledge about B such that M_i^B accepts $1^n \Leftrightarrow 1^n \notin L_B$.

Run M_i^B on 1^n. On any oracle query to B, The oracle will respond consistently, past strings are fixed. If B has not seen the string before, it will prophesize no. Since M_i^B runs in time n^i, it does not have time to query B on all 2^n strings of length n.

If M_i^B accepted 1^n, all other strings of length n are declared not to be in B.
If M_i^B rejected 1^n, declare one string of length n to be in B.
Therefore ~~$L(L_B)$~~ $L(M_i^B) \neq L_B \notin P^B$.

Why is $L_B \in NP^B$? Rather than testing all 2^n strings against the oracle, nondeterministically guess the right one to test the oracle against. This takes unit time on an NP^B machine but exponential time on a P^B machine. Since $L_B \in NP^B \setminus P^B$ we see

$$P^B \neq NP^B$$

This result has set the stage for the next fifty years of complexity research. Any proof which could resolve P vs NP would genuinely have to use new techniques, ones which would not relativize. The last fifty years has seen attempts trying to break the rules:

Randomness: What if SAT is decidable in polytime by an algorithm which returns the assignment correctly only 2/3 rds of the time

Approximation: What if there exists an algorithm for SAT to satisfy a majority of the clauses in polytime, but this last stretch to all clauses requires exponential?

Circuits: proofs using circuits do not relativize. Does there exist a super polynomial lower bound on circuit size for SAT?

All of these methods have probably NP barriers similar to our relativization barrier. Truly, there is no harder problem, no problem has produced more corpses than P vs NP.