P = NP
The Collapse of Hierarchies

Journal:	*Journal of the ACM*
Manuscript ID:	Draft
Manuscript Type:	Paper
Date Submitted by the Author:	n/a
Complete List of Authors:	Riveros, Oscar
Computing Classification Systems:	P vs NP, Complexity, Proof, P=NP

$P = NP$

The Collapse of Hierarchies

Oscar Ariel Riveros Rodiguez

July 1, 2014

The purpose of this publication is to give a formal proof, and construction of the necessary structures to collapse hierarchies $\sum_2$ and $\prod_2$ and therefore, by "The Karp-Lipton Theorem" proof that $P = NP$.

1 Introduction

The P versus NP problem is a major unsolved problem in computer science. Informally, it asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer. It was introduced in 1971 by Stephen Cook in his seminal paper "The complexity of theorem proving procedures" and is considered by many to be the most important open problem in the field. And this is my modest attempt to resolve this question. I dedicate all this work to my wife *Natalia Jaimes*, for their infinite patience and inexhaustible love.

2 Tools and the class $P/Poly$

In this section, will often require the notion of an *oracle* Turing machine. Such a machine has additional (besides input and regular work tapes) *oracle tape* and three states $?, \texttt{true}, \texttt{false}$. During its computation M is allowed to write a word w on the *oracle tape* and then enter the so-call *query state* $?$. Then in one time step M is transferred into `true` if the word w is in the oracle language, and into `false` if w is not in the oracle language; moreover, the contents of the oracle tape are deleted.

Let $B \subseteq \{0,1\}^*$. Then P^B is the class of all languages A for which there exists a polynomially time-bounded oracle Turing machine M such that M with oracle language B accepts A. We also say that A is polynomial time Turing-reducible to B and write $A \leq_T^P B$.

Analogously, NP^B is the class of all languages A for which there exists a nondeterministic polynomially time-bounded oracle Turing machine M such that M with oracle

language B accepts A. (Here, a word is accepted by M if there is at least one computation path of M with input x, that accepts using oracle B.) If $\mathcal{K}$ is a class of sets then $P^K =_{def} \bigcup_{B\in\mathcal{K}} P^B$ and $NP^K =_{def} \bigcup_{B\in\mathcal{K}} NP^B$.

Definition 1. A set $A \subseteq \{0,1\}^*$ is *sparse* if there is a polynomial p such that, for every n, the number of words in A of length n is bounded above by $p(n)$.

Theorem 2. *A set A is in $P/Poly$ if and only if there is a sparse set S such that $A \in P^S$.*

Proof. ($\Rightarrow$): Let $A \in P/Poly$. Let $B \in P$ and p be a polynomial such that $f \in F(p)$ and for all x we have: $x \in A \Leftrightarrow \langle x, f(1^{|x|})\rangle \in B$. Let $f'(1^n) =_{def} 1f(1^n)$, and define S as follow: For every n and i, the i-th string of length n is in S if and only if the i-th symbol of $f'(1^n)$ is a 1. Since f is polynomially length-bounded, S is sparse. A polynomial-time oracle Turing machine on input x, $|x| = n$, can compute $p(n)$, use the oracle language S to reconstruct $f(1^n)$, and then simulate a machine for B to decide if $x \in A$. Hence $A \in P^S$.

($\Leftarrow$): Let $A \in P^S$ via oracle machine M. Let p be a polynomial such that $|S \bigcap \{0,1\}^n| \leq p(n)$. Let q be a polynomial bounding the runtime of M. If for $n \in \mathbb{N}$, $s1, s1, \ldots s_k$ are the words of length at most $q(n)$ in S, then we define

$$f(1^n) =_{def} \langle s_1, s_1, \ldots, s_k\rangle.$$

Since $k \leq p(q(n))$, we have $f \in Poly$. Observe that due to the runtime restriction, M on an input of length n can never ask the oracle for words longer than $q(n)$. Thus all words in the oracle that are relevant for a computation of M on such an input can be found in $f(1^n)$. Thus, a set $B \in P$ can be defined such that for all x, $\langle x, f(1^{|x|})\rangle \in B$ if and only if M with oracle S accepts x. Hence $A \in P/Poly$. □

Theorem 3. *(Karp-Lipton) If* **NP** $\subseteq$ **SIZE**$(n^{O(1)})$ *then* $\Sigma_2 = \Pi_2$ *and therefore the polynomial hierarchy would collapse to its second level.*

The above result is a well-known theorem, see the appropriate literature for their demonstration.

3 $\mathbb{N}_b$ is SPARSE

Definition 4. $\mathbb{N}_b = \{k_b \in \{0,1\}^* : k \in \mathbb{N}\}$, i.e. $\mathbb{N}$ consist of all binary numbers.

Let be $H(n) : \mathbb{N} \to \mathbb{N}_b$ the function (for the moment not necessarily polynomial) such that,

Remark 5. Its clear that $\mathbb{N}_b$ is closed under concatenation.

1. $|H(n)| = 2^n$
2. $H(n) = (2^n - 1)_b \ldots 2_b 1_b 0_b$
3. $n > 1$

The condition 1 and 2 determines the number of zeros, necessary for i_b to complete an well balanced distribution of binary encoding of $\mathbb{N}$, thus:

$H(2) = 3_b2_b1_b0_b$

$H(2) = 11100100$

$H(3) = 7_b6_b5_b4_b3_b2_b1_b0_b$

$H(3)$=111110101100011010001000

$H(4) = 15_b14_b13_b12_b11_b10_b9_b8_b7_b6_b5_b4_b3_b2_b1_b0_b$

$H(4) = 1111111011011100101110101001100001110110010101000011001000010000$

$\vdots$

$H(n) = (n^2 - 1)_b \ldots 2_b1_b0_b$

Lemma 6. *$H(n) = (2^n)^{(2^n)} + \frac{2^n-(2^n)^{(2^n)}}{(2^n-1)^2}$, and $\mathbb{N}_b$ is* SPARSE.

Proof. Observe that:

$H(2)_{10} = 228 = 0 + 4 + 32 + 192$

$H(3)_{10} = 16434824 = 0 + 9 + 162 + 2187 + 26244 + 295245 + 3188646 + 33480783 + 344373768$

$H(4)_{10} = 18364758544493064720 = 0 + 16 + 512 + 12288 + 262144 + 5242880 + 100663296+1879048192+34359738368+618475290624+10995116277760+193514046488576+3377699720527872+58546795155816448+1008806316530991104+17293822569102704640$

$\vdots$

$H(n) = \sum_{i=0}^{2^n-1} i(2^n)^i = (2^n)^{(2^n)} + \frac{2^n-(2^n)^{(2^n)}}{(2^n-1)^2}$.

This formula is easily verifiable by hand or even with any modern CAS, do, if you want more references, http://oeis.org/A062813, the formulation is $H(n) = A062813(2^n)$.

Then by construction $H(n)$ contains all words of length n (i.e, all $i < n \in \mathbb{N}$, with binary encoding), partitioning $H(n)$ by consecutive chunks of length n and since that $|H(n)| = 2^n$ and is polynomial then bound all words of the length n by n, i.e, $|n_b| \leq n$ and therefore $\mathbb{N}_b$ is SPARSE. □

Fact 7. *Then all words n_b of $\mathbb{N}$ of length $n \in H(n)_n$ i.e, $H(n)$ its a "one-step" function such that the partition of $H(n)$ on n consecutive chunks, contains all $n_b \in \mathbb{N}$ such than $n < 2^n$.*

Example 8. $H(n)$ is the full solution of "n-Disks The Tower of Hanoi" under usual binary solution, but, this algorithm $H(n)$ is only composed by elementary operations, with lower bound complexity $\Omega(\log n)$. but ToH$\in$ EXPTIME.

for the more information about the binary solution of ToH,

please, visit: http://en.wikipedia.org/wiki/Tower_of_Hanoi.

Theorem 9. *$P = NP$*

Proof. Note if $H(n)$ outputs the truth table for strings of length n, we can obviously decide membership of $\forall s \in SAT$. However, in this case $H(n) = (2^n)^{(2^n)} + \frac{2^n-(2^n)^{(2^n)}}{(2^n-1)^2}{}^n$, so it would not b e an admissible advice function for P/poly. Then define $h_n(x) =$

$\sum_{k=0}^{n} \frac{H^{(k)}(n)}{k!} (x-n)^k$, the Taylor expansion of $H(n)$ over the point n and degree n, there is a polinomial of degree n. (Note in the definition of h, $H(n)$ is considered as a function and not evaluated at point n, to note this fact has been written H instead of $H(n)$)

But that $H(n) = h_n(n)$. it is obvious that all terms except $H(n)$ (already evaluated) vanish. then $h_n(x)$ is a polinomial of degree n and the number of strings at length n in SAT is at most polynomial in n. Then by the Karp-Lipton theorem and this result, $P = NP$. □

4 References

As references, are already listed by definition 1 theorems 2 and 3, and are widely known, it is not necessary to list a reference.

if you need more information, it is because it needs from the basics, and this is a good place to start.

http://en.wikipedia.org/wiki/P_versus_NP_problem