NEW ZEALAND JOURNAL OF MATHEMATICS
Volume 21 (1992), 33–89

AN INVITATION TO STRUCTURAL COMPLEXITY*

ROD DOWNEY

(Received May 1991)

1. Introduction

In this article I plan to give the reader a (hopefully) appetizing taste of what I feel is one of the most interesting, important and fascinating areas of mathematics (and computer science). Because of space limitations, the presentation will be necessarily sketchy. The audience is assumed to be mathematically mature, yet probably unfamiliar with the basic concepts and techniques in the area. Because of this I will include quite a lot of background material. Finally I should stress that this is not a survey of the area, merely an idiosyncratic view of a number of topics that I feel fit the description of "appetizing taste". For a more comprehensive account the reader should refer to Balcazar, Diaz and Gabarro [**1988, 1990**], as well as the classic text of Garey and Johnson [**1979**]. What is structural complexity? *Complexity theory* is concerned with understanding the algorithmic nature of mathematical structures and processes. This area grew out of *recursion theory* which is also concerned with the algorithmic nature of mathematics. The crucial difference is that in complexity theory, we are concerned with the *amount of resource* needed to perform an algorithm (or process) whereas in recursion theory we are concerned as to whether a process can be performed by an algorithm *at all*. While there is no general agreement as to what defines *structural* complexity ("I know it when I see it") the area does seem to be delineated by certain aspects. The key ones seem to be that we are concerned with algorithms "in the abstract" and try to reduce the central issues (such as (e.g.) nondeterminism) to their basics. The situation is analogous to Post's realization that a simple diagonalization argument lay at the heart of Gödel's incompleteness theorem (Post [**1944**]). Anyhow, aside from these vague comments, I hope the present article will give the reader some idea of the area.

The principle issues of the area are trying to understand the relationship between 'complexity classes' of different resources. The most well known of these is the infamous P = NP? question, or its cousin P = PSPACE? (see §3 for all relevant definitions). These famous questions are illustrated as follows: Given an (undirected) graph G, a *Hamilton path* is a path through each of the vertices of G, passing through each vertex *exactly once*. This is an extremely important problem related to many practical situations (ask your students doing operations research!). Suppose you are asked to dream up an algorithm A to do the following : given a graph G, $A(G)$ computes a Hamilton path in G, or tells us one does not exist.

One algorithm is the following : use trial and error. Alas, this is not a very nice algorithm in terms of time since in the worst case it may take $> O(2^n)$ many

*This is an expanded version of an invited address presented to the New Zealand Mathematics Colloquium, University of Otago, Dunedin, May 1991. The author wishes to thank Ewan Tempero for various corrections.

steps where n is the number of edges of G. So for any reasonable size (say > 1000 vertices) its performance is woeful. What we would desire is an algorithm that ran in "polynomial time" say $O(n^3)$ or $O(n^4)$. So would we all! Despite the combined efforts of a huge number of mathematicians over a long period, no polynomial time algorithm for the problem above has been found. Furthermore, the problem above lies in a large class of diverse practical problems which we have shown to be *equally difficult* in the sense that if we can solve any of them *fast* then we can solve *all* of them fast.

One of the things I hope to do in the present article is to make precise what I mean in the discussion above. Staying at the informal level, note that as far as the recourse of *space* is concerned, the problem above is *tractable.* Space is reusable. So we can, one at a time, in the same space try each possible path. Thus we see that the Hamilton path problem lies in PSPACE (polynomial space). We believe that the Hamilton path problem does *not* lie in P (polynomial time) *nor* is it the 'most difficult problem' in PSPACE. But note the general structural complexity question here: DOES P = PSPACE? That is, is there *any* problem we can do in polynomial space that needs more than polynomial time?

In fact, the situation is even more intriguing. We assume our data is input in base n some $n \neq 1$. (It turns out that $n = 2$ is representative in a robust sense). So we can define LOGSPACE to be the collection of problems that can be solved in space log of the length of the input. Now we *can* prove that LOGSPACE $\neq$ PSPACE. And we know that LOGSPACE $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE. But we don't know which inclusions are proper.

In the initial sections we will look at some of the basic notions and definitions of the area. We will see that sometimes one can prove separation results and sometimes we can prove coincidence. Often we don't know and, indeed can show that 'known techniques will fail' in a strong sense. We do this by examining the effect of varying the model of computation and "oracle" results, as we see in later sections.

For reasons of simplicity we concentrate on the complexity classes above. Anyhow, they seem the most important classes. We do try to mention the relationship between a number of models such as Turing machines, RAMs and circuit models. We will also look more deeply into the question of whether questions such as "does P = NP" matter in "real life". While there are many issues here, in essence this questions has to do with 'approximate solutions' in some sense or another. We look at whether it is possible always to give good approximate solutions (and what we mean by this). We also look at the interactive proofs systems: going back to the Hamilton path example, is it possible to give someone overwhelming evidence that G has a Hamilton path in *such a way that they cannot find a path.* (These are the subset of the so-called *zero-knowledge proofs.*)

We will briefly look at delicate separation results using Ramsey type arguments and Kolmogorov complexity as well as the probabilistic separation arguments via circuit complexity. These are all model dependent.

We finish with some fascinating recent work related to the Robertson-Seymour theorem which challenges the notion of P-time as a good measure of complexity.

Finally, I will try to give a small sample of the easier proof techniques although even for these space will not allow more than a general outline in many cases. Of course most results are stated without proof.

There are so many other interesting questions we will not look at. Here is one to try (Hartmanis and Stearns [**1965**] : is there any irrational number q for which there is an algorithm such that for some constant n the algorithm will compute the next member q's decimal expansion once every n steps?

Finally, I would like to dedicate this paper to Juris Hartmanis who fired my interest in the area when I heard him give a truly exciting talk at Monash (in 1984, I think). Whilst I don't have the same inspirational talents of Hartmanis, I do hope this article will convince some of you to look more deeply into this fascinating and important area.

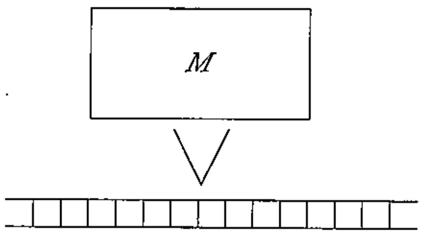
2. Preliminaries

Many mathematicians and computer scientists do not even have a basic training in recursion/complexity theory. There are several reasons for this. Some have to do with fashions and the evolution of the subjects, but mainly it is because the importance of the area has come to the fore in only the last 20 or so years. I hope this section will give the reader some rudiments of the area. The reader should see, say, Salomaa [**1985**], Rogers [**1967**], Soare [**1987**] or Davis and Weyuker [**1983**] for more details.

Our concern will be functions $f : A \to B$ with $B, A \subseteq \mathbb{N}$. These are called *partial* functions and if $A = \mathbb{N}$ then we call them *total*. It may seem restrictive to look only at $\mathbb{N}$, but from an algorithmic point of view, we can consider algorithmic structures as *coded* into $\mathbb{N}$ in some way. For instance if we consider $\{0,1\}^*$, the set of all (finite) binary strings, then we can assign codes called (Gödel numbers) as follows: let $n(0) = 1$, $n(1) = 2$. Then if $\sigma = a_1 \ldots a_n$ we can assign $\#(s) = 2^{n(a_1)}3^{n(a_2)} \ldots P_n^{n(a_n)}$ where P_i denotes the ith prime. Incidentally, this gives a generic way of showing countability of many well known structures. I find it useful when teaching such matters. The reader should try the trick with the collection of finite subset of $\mathbb{N}$, or with $\mathbb{Q}$ (e.g. write if $r = (-1)^\delta \frac{a}{b}$ with $a, b \in N$ in lowest terms and $\delta = 0$ or 1, then $\#(r) = 2^\delta 3^a 5^b$). I should point out that there are many other ways of proving (e.g.) $\{0,1\}^*$ is countable. For instance let $f(\sigma) = n(a_1) \ldots n(a_n)$. Later, when we study *resource* bounded algorithms, the representation of σ will be very important. The reader should also note that the coding procedures described above are *algorithmic* since we give an 'effective procedure' for the relevant embedding.

While we are on the subject of algorithms, what do we mean? Here we come to the famous thesis of Church: paraphrased in modern terminology this thesis states *the collection of algorithmic partial functions coincide precisely with those that can be computed by (Fortran)(Basic) (Pascal) programs*. Clearly Church didn't express his thesis this way. There have been many formulations claiming to capture the class of "intuitively computable" functions. These all turn out to be equivalent, thus providing strong evidence for Church's thesis. We will call this class the class of *partial recursive functions*. If f is a partial recursive total function, we simply say that f is recursive.

For our purposes, we will here describe only one model and only later will we describe some others (when we look at complexity issues). A *Turing machine* (T.M.) M can be visualized as a device as follows



M can scan a square on an infinite two way tape. This machine has a finite number of internal states $\{q_0, \ldots, q_n\}$ say and can print/read symbols from an alphabet $\{b, 1, 2\}$ say where b denotes a blank square. According to a given $\langle$state, symbol currently scanned$\rangle$, M will go into a potentially new state and perform one of the following operations: move left, move right, or (over) print a symbol on the current square. Thus a T.M. is simply a finite set of quadruples of the form $\langle q_i, S_j, P_{ij}, q_{ij}\rangle$ where $P_{ij} \in \{S_{ij}, R, L\}$. For instance $\langle q_i, S_j, S_k, q_t\rangle$ is read as if M is in state q_i reading S_j then it prints S_k in place of S_j and goes to state q_t. We say M is deterministic if for each $\langle q_i, S_j\rangle$ configuration there corresponds without loss of generality exactly one quadruple. M is *nondeterministic* otherwise. We set aside q_0 as the initial state and q_s as the stop one. Finally we can code x as a block of x ones and say that a deterministic M simulates a partial function f if for all $x \in \text{dom}f$, M when started on the leftmost one of a block of x ones stops on the leftmost one of a block of $f(x)$ ones (i.e. on an otherwise blank tape), and if $x \notin \text{dom}f$, then M does not halt when started on the leftmost one of a block of x ones. The following example may be instructive:

Example. $f(x) = 2x$ is recursive. A T.M. to compute f is

$$\{\langle q_0, b, b, q_s\rangle, \langle q_0, 1, 2, q_1\rangle, \langle q_1, 2, L, q_1\rangle, \langle q_1, b, 2, q_2\rangle, \langle q_2, 2, R, q_2\rangle,$$
$$\langle q_2, 1, 2, q_1\rangle, \langle q_2, b, L, q_3\rangle, \langle q_3, 2, 1, q_3\rangle, \langle q_3, 1, L, q_3\rangle, \langle q_3, b, R, q_s\rangle\}.$$

The reader might like to ponder how one would go about proving that the class of (say) Pascal-computable functions coincide with those computed by Turing Machines. For the purposes of this section, the model is not all that important, but two *properties* of this class are:

Theorem 2.1. (Enumeration Theorem — Universal Turing Machine). *There is an algorithmic way of enumerating all partial recursive functions. That is, there is a partial recursive function $f(x, y)$ of two variables with*

$$f(x, y) = \phi_x(y)$$

where $\phi_x(y)$ denotes the result, if any, of the x-th Turing machine on input y, and this makes sense.

Proof Sketch. Use Gödel numbering. A T.M. is a finite set of quadruples. Each quadruple Q can be given a number $\#(Q)$. If $Q_i = \langle q_i, S_j, P_{ij}, q_{ij}\rangle$, then $\#(Q_i) = 2^{n(q_i)}3^{n(S_j)}5^{n(P_{ij})}7^{n(q_{ij})}$. Let $M = \{Q_1, \dots, Q_m\}$ with $\#(Q_i) < \#(Q_{i+1})$. Then assign $\#(M) = 2^{\#(Q_j)}3^{\#(Q_2)}\dots(P_m)^{\#(Q_m)}$. Now enumerate the M's in increasing order. ∎

The point of (2.1) is that we can pretend we have the machines $\phi_0, \phi_1, \dots$ in front of us and to compute 10 steps of the computation of ϕ_9, on input 30 (written $\phi_9^{10}(30)$) we can pretend to walk to ϕ_9, put 30 on the tape and run it for 10 steps. This property is the most important one of this class since it allows us to diagonalize *over* the class of partial recursive functions *without leaving the class.* As an example we have the following.

Theorem 2.2. (The Halting problem). *There is no algorithm which, given x and y decides if $\phi_x(y)\downarrow$ (i.e. $\phi_x(y)$ halts). Indeed there is no algorithm to decide if $\phi_x(x)\downarrow$.*

Proof. This is similar to Cantor's proof that the reals are uncountable. Suppose such an algorithm exists. Then g below is (total) recursive

$$g(x) = \begin{cases} 1 & \text{if } \phi_x(x)\uparrow \\ \phi_x(x)+1 & \text{if } \phi_x(x)\downarrow \end{cases}$$

Now by (2.1) there is a y with $g = \phi_y$. As g is total, $\phi_y(y)\downarrow$. So $\phi_y(y) = g(y) = \phi_y(y)+1$, a contradiction. ∎

Remark 2.3. Notice that the same proof shows that there is no algorithm to enumerate all the *total* recursive functions. Note that we can define a *partial* recursive g via $g(x) = \phi_x(x)+1$. All this means is that for any *index* y of g, (i.e. $\phi_y = g$) we have $g(y)\uparrow$. The reader is invited to prove that g so defined has no (total) recursive extension.

This brings us to an important idea: reducibility. We often prove that problem A is unsolvable by showing that 'A is as hard as B' and B is unsolvable. This describes what we call a *reduction* $B \leq_T A$. We will define this more formally, but first an example.

Theorem 2.4. *There is no algorithm to decide if* dom $\phi_x = \varnothing$

Proof. We show that if we could decide *if* dom$\phi_x = \varnothing$ *then* we solve the halting problem. This is written as $\{x : \phi_x(x)\downarrow\} \leq_T \{x : \text{dom}\phi_x = \varnothing\}$. Define a partial recursive function g via

$$g(x,y) = \begin{cases} 1 & \text{if } \phi_x(x)\downarrow \\ \uparrow & \text{if } \phi_x(x)\uparrow \end{cases}$$

To see that g is partial recursive, we describe an algorithm for it. We compute g in stages. At stage s, we compute $g^s(x,y)$ as 1 if $\phi_x^s(x)\downarrow$ and to not *as yet* halt if $\phi_x^s(x)\uparrow$. We can consider $g(x,y)$ as a recursive collection of partial recursive functions. That is, there is a recursive $s(x)$ with $\phi_{s(x)}(\ \ ) = g(x,\ \ )$.

Then
$$\text{dom}(\phi_{s(x)}\ \ ) = \begin{cases} \mathbb{N} & \text{if } \phi_x(x)\downarrow \\ \varnothing & \text{if } \phi_x(x)\uparrow \end{cases}$$

So if we can decide if dom $\phi_{s(x)} = \varnothing$, we could decide the halting problem. ∎

A couple more definitions. First we code problems as subsets of $\mathbb{N}$. (e.g. $K^* = \{2^x 3^y : \phi_x(y)\downarrow\}$)

Definition 2.5. We call $A \subseteq \mathbb{N}$

(i) *recursive enumerable* (r.e.) iff $A = \text{dom}\phi_x$ for some x.

(ii) *recursive* if both A and $\overline{A} = \mathbb{N} - A$ are r.e.

Let $W_e = \text{dom}\phi_e$ denote the e-th r.e. set with $W_{e,s} = \text{dom}\phi_e^s$. The following result (i) explains the name recursively (or "computably") enumerable and (ii) provides a very useful tool.

Theorem 2.6.

(i) *An infinite set A is r.e. iff there is a recursive injective function f with $f(\mathbb{N}) = A$.*

(ii) *(Kleene). B is r.e. iff there is a recursive relation R such that $x \in B$ iff $(\exists y)R(x, y)$.*

Thus an r.e. set is a "listable" one. Note that recursive sets correspond to decidable questions: If A is recursive, then to decide if $n \in A$, let $f(\mathbb{N}) = A$, $g(\mathbb{N}) = \overline{A}$. List $f(0), g(0), f(1), g(1), \ldots$ and wait till n occurs. If n occurs on the g-list, $n \notin A$. Otherwise n occurs on the f-list and so $n \in A$. Note that $K = \{x : \phi_x(x)\}$ is an r.e. but not recursive set. We can push the reasoning of (2.4) a lot further: we call A an *index set* if $x \in A$ and $\phi_x = \phi_y$ implies $y \in A$.

Theorem 2.7. (Rice). *An index set A is recursive (and so the problem it codes is decidable) iff $A = \mathbb{N}$ or $A = \varnothing$.*

Of course many problems are decidable so are not coded by index sets. Now to formalize the notion of reducibility. We can say $A \leq_T B$ (A is Turing reducible to B) if we can decide $x \in A$ given that we can ask a finite number of questions of B (for free). This can be made precise by the notion of an *oracle Turing Machine*. We augment the normal model by putting a further infinite (oracle) tape upon which *we* write the members of B in order, and allow the question *is x on the oracle tape*? to be asked in the process of a computation. In modern terminology, the oracle tape is a read only memory (containing B). Again there is an enumeration of all oracle T.M.'s, and we let $\Phi_{e,s}(C; x)$ denote the effect of computing s steps in the computation of the e-th oracle T.M. with C on the oracle tape with input x. The analogue of (2.2) is that for any set A, the set $A' = \{x : \Phi_x(A; x)\downarrow\}$ is not *recursive* in A, that is $A' \not\leq_T A$. We refer to A' as "K relativized to A" or the *jump of A* and the process above as *relativisation*. Finally the relation $\leq_T$ is a partial ordering and generates equivalence classes that measure "equality of complexity". These are called *degrees (of unsolvability)*. We let $\mathbf{0}$ denote the degree of the recursive sets and $\mathbf{0}'$ the degree of the halting problem, $\mathbf{0}''$ the degree of K' etc. We call a degree r.e. if it contains an r.e. set. It is not difficult to show that if $\boldsymbol{a}$ is an r.e. degree then $\boldsymbol{a} \leq \mathbf{0}'$, but it can be shown that the converse does not hold. Also since the collection of oracle T.M.'s is countable, there are uncountably many degrees. The proof of (2.7)-which is similar to (2.4) - shows that if A is an r.e. index set then $A \in$

$\mathbf{0}$ or $A \in \mathbf{0}'$. Post [**1944**] asked if there exists an r.e. degree $\boldsymbol{a}$ with $\mathbf{0} < \boldsymbol{a} < \mathbf{0}'$. Post's problem was finally solved via an ingenious method now called the priority method, independently by Friedberg [**1957**] and Muchnik [**1956**]. They showed:

Theorem 2.8. (Friedberg, Muchnik). *There exist r.e. degrees* $\boldsymbol{a} \mid \boldsymbol{b}$. *That is* $\boldsymbol{a} \not\leq \boldsymbol{b}$ *and* $\boldsymbol{b} \not\leq \boldsymbol{a}$.

There have been many extensions to (2.8). For instance, Sacks [**1964**] showed that the r.e. degrees are dense and Lachlan [**1966**], Yates [**1966**] showed they do not form a lattice, yet these are $\boldsymbol{a} \mid \boldsymbol{b}$ with greatest lower bound $\mathbf{0}$. Soare [**1987**] has a lot of information on r.e. degree, r.e. sets and related matters. These results lead us a little far afield, but there is one important ingredient we need from the proofs.

Lemma 2.9. (Use principle). *Suppose* $\Phi(A;x)\downarrow$. *Let* $u = u\big(\Phi(A;x)\big)$ *denote the maximal* n *such that "*$n \in A$*" is asked in the computation. Suppose that* $C[u] = A[u]$ *(where* $B[x] = \{p : p \leq x \text{ and } p \in B\}$*). Then* $\Phi(A;x) = \Phi(C;x)$.

Proof. A and C give the same answers so the computations are identical. ∎

We remark that r.e. sets and degrees occur everywhere once one starts looking at mathematics and asking if algorithms to perform our theorems exist. For instance, every vector space has a basis. This fails recursively: there are recursive vector spaces with no recursive basis. (Metakides – Nerode [**1977**]). The word problem for groups: if $\boldsymbol{a}$ is an r.e. degree there is a finitely presented group G such that $\{x = 1$ and $x \in G)$ has degree $\boldsymbol{a}$. (Boone [**1966**]), Hilbert's tenth problem: if A is an r.e. set, there is a polynomial $P(x_1, \ldots, x_n, y)$ with integer coefficients such that $y \in A$ iff $(\exists x_1, \ldots, x_n > 0)\big(P(x_1, \ldots, x_n, y) = 0\big)$ (Matajeseivic [**1970**]). This last result has the amazing consequence that (e.g.) the primes are the exact positive solutions of some polynomial equation.

We finish the section with an amusing illustration due to Conway [**1972**].

Conway considered the following question: Those familiar with number theory and/or famous unsolved problems in intuitive mathematics will recognise the Collatz function:

$$c(n) = \begin{cases} \frac{n}{2} & \text{if } n \text{ even} \\ 3n+1 & \text{if } n \text{ odd} \end{cases}$$

The infamous conjecture is that for all n there is a k with $c^k(n) = 1$. Now $c(n)$ behaves in an apparently unpredictable way. For instance $c^{77}(27) = 9232$.

Conway observed that $c(n)$ is but an example of a family of more general questions. Consider $g(n) = a_i n + b_i$ if $n \equiv i \pmod p$ where $a_0, \ldots, a_{p-1}, b_0, \ldots, b_{p-1}$ are chosen to make $g(n) \in \mathbb{N}$. Conway's question was:

2.10. *Given a system* $\{a_0, \ldots, b_{p-1}\}$ *as above. What can be predicted about* $g^k(n)$*?*

Perhaps surprisingly, the answer is "nothing, in general". Indeed, this is true even if $b_i \equiv 0$ for all i so that $g^k(n)/n$ is periodic.

Theorem 2.11. (Conway [**1972**]). *Given a partial recursive function ϕ, there is a system $\{a_0, \dots , a_{p-1}\}$, as above, such that*

(i) *if $n \in \operatorname{dom}\phi$ then $2^{\phi(n)}$ is the first power of 2 in the sequence $g^k(2^n)$, and*

(ii) *if $n \notin \operatorname{dom}\phi$, $g^k(2^n)$, $k = 0,1,2,\dots$ contains no power of 2.*

An immediate consequence is that there is no algorithm to decide if $(\exists k)\big(g^k(2^n) = 1\big)$. The ingenious proof of this result uses another model of computation well suited to coding recursive functions into algebraic structures.

A *register (or Minsky)* machine M is a machine with n registers that we write as an n-tuple. We say M computes f if given $(x, 0, 0, \dots , 0)$ it halts with output $\big(f(x), 0, 0, \dots , 0\big)$ for $x \in \operatorname{dom} f$, and does not halt otherwise. M can perform only two labelled instructions as follows

Instruction m

$$\begin{tikzcd} m \arrow[r] & a^+ \arrow[r] & n \end{tikzcd}$$

add one to register a and move to instruction n.

$$\begin{tikzcd} m \arrow[r] & a^- \arrow[r] \arrow[dr, two heads] & n \\ & & k \end{tikzcd}$$

If the content of register a is $>$ 0 subtract one from register a and go to the instruction labelled n; otherwise go to instruction k.

For instance, the machine below simulates $f(x) = 2x$:

$$\begin{tikzcd} 0 \arrow[r] & 1^- \arrow[r] \arrow[dr, two heads] & 2^+ \arrow[r] & 3^+ \arrow[r] & 1^- \arrow[lll, bend right] \arrow[r, two heads] & 2^- \arrow[r] \arrow[d, two heads] & 1^+ \arrow[l, bend right] \\ & & s & & & 3^- \arrow[r] \arrow[d, two heads] & 1^+ \\ & & & & & s & \end{tikzcd}$$

In fact, it is not difficult to show that the class of functions that can be simulated by Minsky machines consists of exactly the partial recursive functions. The beauty of Minsky machines is that they can often be easily coded into (e.g.) algebraic structures.

Conway codes Minsky machines into Collatz-like functions as follows:

A *vector game* is played on a finite (ordered) list of vectors $L = \{v_1, \dots, v_p\}$ with $v_i \in \mathbb{R}^d$. For example

$$\begin{aligned} v_1 &= (0,0,0;1,-1) \\ v_2 &= (-1,0,0;-3,1) \\ v_3 &= (0,0,0;-3,0) \\ v_4 &= (0,0,1;-2,3) \\ v_5 &= (0,1,0;-1,2) \\ v_6 &= (-1,0,0;0,1) \\ v_7 &= (0,0,0;0,0) \end{aligned}$$

Upon L we play the following game. Starting with a vector v with non-negative integer coordinates, add v to the first vector on the list that preserves the property of having non-negative integer coordinates. That is v becomes replaced by the least i with $v + v_i \in \mathbb{N}^d$. Repeat. Call this $m(v)$ define $g_L(v) = q$ if $(k, \vec{0})$ is first of the form $(\phi, \vec{0})$ in the sequence $m\big((k,\vec{0})\big)$, $m^2\big((k,\vec{0})\big) \dots$. For instance, if $v = (3,0,0;0,0)$ and L is as given above $m(v) = (2,0,0;0,1)$, $m^2(v) = (2,0,0;1,0)$.

Theorem 2.12. (Conway [**1972**]). *For any partial recursive ϕ, there is a list L such that $g_L = \phi$.*

Proof Sketch. Simulate Minsky machines:

For an instruction $$\begin{tikzcd} m \arrow[r] & a^+ \arrow[r] & n \end{tikzcd}$$ have a vector of length $d(0, \dots, 0, \underset{\substack{\uparrow \\ m\text{th position}}}{1}, \dots, 0;; -m, n)$ and $$\begin{tikzcd} m \arrow[r] & a^- \arrow[r] \arrow[dr] & n \\ & & k \end{tikzcd}$$ $(0, \dots, 0, \underset{\substack{\uparrow \\ m\text{th position}}}{-1}, 0, \dots, 0;; -m, n)$ and $(0, \dots, 0; -m, k)$ where $d-2$ is the number of registers. These instructions are then ordered in decreasing order of m and preceded by $(0, \dots, 0; 1, -1)$. The machine

$$\begin{tikzcd} & & 2^+ \arrow[d, "2"] \\ 0 \arrow[r] & 1^- \arrow[ur, "1"] \arrow[d, Rightarrow] & 3^+ \arrow[l, "3"] \\ & s & \end{tikzcd}$$

corresponds to the game L above.

It is easy to prove by induction that for L so construed, $g_L = \phi$. ∎

Finally we replace vector game by *rational games*; these using primes to code vector locations. For a rational game $L = \{r_1, \dots, r_p\}$ is a finite list of rationals. We let $m_L(n) = nr_i$ if i is least with nr_i an integer. We define $h_L(n) = k$ if 2^k is the first power of 2 in the list $m(2^n), m^2(2^n), \dots$.

Now if we replace $(a, b, c, d, \dots)$ by $2^a 3^b 5^c 7^d \dots$, we replace a vector game by the corresponding rational game, and these compute the same partial recursive function (an easy induction). It is not too difficult to see that a rational game can be represented as a generalized Collatz question by picking the r_i appropriately and using congruences. This concludes our sketch of the proof. ∎

3. Complexity Theory – Basics

When we begin to worry about the amount of resources we use in a computation, it seems that the device we use for computation might be important. Some computers are faster than others. This is true in some ways and false in others. It depends on the robustness of the measure we use. First we should clarify what we mean by a complexity measure.

Definition 3.1. (M. Blum [**1967**]). Let $\{\phi_e : e \in \mathbb{N}\}$ list all the partial recursive functions. We say a set $\{\Gamma_e : e \in \mathbb{N}\}$ is a *complexity measure* if for all x,

(i) $\Gamma_e(x)\downarrow$ iff $\phi_e(x)\downarrow$, and

(ii) the relation "$\Gamma_e(x) = y$" is recursive.

The classical examples are (i) time: $\Gamma_e(x) = y$ means $\phi_e(x)\downarrow$ in exactly y steps, and (ii) space: $\Gamma_e(x) = y$ means that $\phi_e(x)\downarrow$ and uses exactly y squares of tape (or memory) in the computation.

Remark. Note that in (ii) the reason $\Gamma_e(x) = y$ is recursive since we can simulate $\phi_e(x)$ until either the space y is exceeded, or $\phi_e(x)\downarrow$ or we get a repeated configuration (in which case $\phi_e(x)\uparrow$).

Up to a recursive cost function, all measures are the same: Blum [**1967**] observed that if $\{\Gamma_e : e \in \mathbb{N}\}$ and $\{\Delta_e : e \in \mathbb{N}\}$ are two measures, there is a recursive function g such that for all e and almost all x, $\Gamma_e(x) \le g(x, \Delta_e)$ and $\Delta_e(x) \le g(x, \Gamma_e(x))$.

For any partial recursive f, there are infinitely many ways to compute it. The natural question then arises : how should we measure the "best" complexity of a function. Unfortunately, there is no simple answer to this. The obvious way is to take the ϕ_e which equals f and runs in the least number of steps. Alas, no such ϕ_e may exist.

Theorem 3.2. (Speedup theorem, Blum [**1967**]). *For all increasing recursive $g(\ ,\ )$ and complexity measure $\{\Gamma_e : e \in \omega\}$, there is a recursive $\{0,1\}$-valued function f such that if ϕ_e equals f, then there is an i with $\phi_e = f$ and such that, for almost all $x, g(x, \Gamma_i(x)) < \Gamma_e(x)$. That is, f has no g-fastest algorithm.*

The proof of (3.2) is a quite involved priority argument which we omit. For more on the abstract axiomatically defined complexity the reader should see Blum and Marques [**1973**] and Soare [**1977**]. One can also use "confinal sequence" to measure complexity see Maass-Slaman [**ta**].

Rather than study the general implications of complexity measures, the real depth of the subject comes from studying well defined classes of functions of the same complexity. So suppose we have a total recursive function f and a fixed model and measure. We define the complexity class with same f to be $C_f = \{\phi :$

$(\exists e)(\phi_e = \phi$ and $(aax)(\Gamma_e(x) \leq f(|x|))\}$ where $|x|$ denotes the length of x. If we fix the model as that of a deterministic Turing machine we write C_f^T as DTIME $(f(|x|))$. Here we must be careful how to present x. Two standard representations are as x ones (tally notation, written $x \in \{1\}^*$) and $x \in \{0,1\}^*$, (i.e. x in binary notation). We have the following standard classes

$$C^T_{|x|} = \text{real time} = \text{DTIME}(|x|)$$
$$\bigcup_n C^T_{2^{n|x|}} = \text{exponential time, EXT} = \bigcup_n \text{DTIME}(2^{n|x|})$$
$$\text{EXPTIME} = \bigcup_{c>0} \text{DTIME } (2^{n^c}).$$
$$\bigcup_n C^T_{|x|^n} = \text{polynomial time, } P \text{ or PTIME.}$$

We remark that it is not altogether clear if P or EXPTIME are complexity classes in the sense of the definition above. Why should there be a *single* recursive f such that $\phi_e \in P \Rightarrow \Gamma_e(x) \leq f(|x|)$ for almost all x. That such an f exists follows from the following.

Theorem 3.3. (Union theorem – McCreight – Meyer [**1969**]). *If $\{f_i : i \in N\}$ are uniformally recursive increasing functions $\big($i.e. $(\forall n, x)(f_n(x) \leq f_{n+1}(x))\big)$ then there is a recursive f such that $C_f = \cup_n C_{f_n}$*

Corollary 3.4. *P, EXPTIME and EXT are complexity classes in the sense above.*

Proof (of 3.3). We build f to meet the requirements (for $e \in \mathbb{N}$)

$$P_e : (aax)\big(f_e(x) \leq f(x)\big).$$

For simplicity we work with $x \in \{1\}^*$. These have priority order $P_0, P_1, P_2, \ldots$ and note that if we satisfy all the P_e then $\cup_e C_{f_e} \subseteq C_f$.

Let $d(x) = f_x(x)$. Note that $C_{d(x)} \supseteq C_{2^{|x|}}$ and so $C_{d(x)} \neq P$. To keep $C_f \subseteq \cup_e C_{f_e}$, we need also satisfy the requirements.

$$N_e : (aax)\big(\Gamma_e(x) \leq f(x) \rightarrow \phi_e \in \cup_n C_{f_n}\big).$$

We always keep $f(x) \leq d(x)$ and ensure that if $\phi_e \notin \cup_n C_{f_n}$ then $(\forall n)(\exists^\infty x)$ $\big(f_n(x) < \Gamma_e(x)\big)$.

We set aside $\{f_{\langle e,z\rangle} : z \in \mathbb{N}\}$ for the sake of N_e. Here $\langle \quad , \quad \rangle$ is a bijection from $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. We build f in stages. At stage x, search for the least $\langle e, z\rangle \leq x$ *not yet cancelled* with $f_{\langle e,z\rangle}(x) < \Gamma_e(x) \leq d(x)$. If $\langle e, x\rangle$ exists, set $f(x) = f_{\langle e,z\rangle}(x)$ and cancel $\langle e, z\rangle$. If no $\langle e, z\rangle$ exists set $f(x) = d(x)$. Note that $f(x) \leq d(x)$ in either case.

First all the P_e are met $\big($that is $f_e(x) \leq f(x)\big)$ for aax. To see this we need only show $f(x) < f_e(x)$ only finitely often. After stage $x, f(x) < f_e(x)$ only if $\langle i, z\rangle$ is cancelled for $\langle i, z\rangle < e$. Thus $f(x) < f_e(x)$ at most e further times.

Finally all the N_e are met. For suppose not. Then there is $\phi_e \in C_f$ with $\phi_e \notin \cup_n C_{f_n}$. As $\phi_e \in C_f$, and for all x, $f(x) \leq d(x)$ so (aax) $\big(\Gamma e(x) \leq d(x)\big)$. Hence $(\exists^\infty x)\big(f_{\langle e,z\rangle}(x) < \big(\Gamma e(x) \leq d(x)\big)\big)$.

By construction, it follows that for each z, there is an x with $f(x) = f_{\langle e,z\rangle}(x) < \Gamma_e(x)$, and hence $\phi_e \notin C_f$ after all. ∎

For the rest of our studies we will fix some model of computation (Turing machine unless otherwise specified), although it is not necessary in all cases. It is appropriate at this point to mention why we study classes like PTIME and EXPTIME. It has been observed that for all reasonable models of computation these classes are invariant in the sense that if D_1 and D_2 are two devices, then for some polynomial p, D_1 computes $g(x)$ in $f(|x|)$ steps then D_2 computes $g(x)$ in $p\big(f(|x|)\big)$. *This is essentially a refinement of the Church-Turing thesis, namely that feasible computations are those that can be done in* PTIME *and this class is invariant.*

We should remark that this idea has a pragmatic component. The general principle that practising algorithm builders tend to use is that "if a process Q is natural and occurs in practice (i.e. not dreamed up by some theoretician) and it is in PTIME, then there is a reasonable polynomial to do Q. Namely one where the polynomial is of low degree and the constants of proportionality small". As we will see in §6 a number of recent results challenge this belief (although they do not yet refute it). We should mention that for practical purposes (i.e. modelling actual computation of a "real computer"), the Turing machine model is not really accurate since the trade off is $O(|x|^2)$. There are models more suited to modelling actual computation. These include the RAM (random access model) that we look at later §6. For that model, we wish to look at lower bounds for problems such as (e.g.) matrix manipulation. The techniques tend to be rather intricate combinatorial ones.

Returning to our story, we can prove a number of general basic results about complexity classes. This work really begins with the ground breaking paper of Hartmanis and Stearns [**1965**] from which the area derives its name. First we will need a slight change of notation. Rather than looking at performing a computation of a function (i.e. $x \to f(x)$) we will speak of *accepting a string* y (i.e. machine M on input y outputs 1 rather than 0). Obviously accepting $\langle x, f(x)\rangle$ is identifiable with computing $f(x)$ from x so we will not lose any generality by this procedure. Yet it does make the following definitions easier. First we will augment the Turing machine model by adding a work tape. Such a machine M has an input and a *work tape*. The input tape is a read only tape, and all the computation must be done on the work tape. If the input tape is one way we call M *online and two way offline*. With this model we can define the following.

Definition 3.5. DSPACE $\big(f(|x|)\big)$ is the class of sets for which there is a machine M such that, for almost all $x, x \in A$ iff M accepts x and uses at most $f.(|x|)$ spaces of work tape.

Some typical classes are

$$\text{PSPACE} = \bigcup_n \text{DSPACE}(|x|^n)$$

$$\text{EXPSPACE} = \bigcup_n \text{DSPACE}(2^{n|x|})$$

$$\text{LOGSPACE} = \bigcup_n \text{DSPACE}\big(n\text{LOG}(|x|)\big).$$

We remark that the last example (which turns up in practice) shows why the separate work tape is desirable.

Definition 3.6. We say a set A is in the *nondeterministic time class with name* f, NTIME $\big(f(|x|)\big)$, if there is a nondeterministic machine M such that for all $y, y \in A$ iff some computation of M accepts y.

Note that since a given $\langle q_i, S_j \rangle$ may correspond to many quadruples, it follows that M has potentially exponentially many computation paths of length $f(|x|)$. In a similar way we can define NSPACE $\big(f(|x|)\big)$, etc.

Theorem 3.7. (Linear Speedup, Hartmanis and Stearns [**1965**]). *Let $d > 0$ be a real constant. Then*

(i) DSPACE $\big(df(|x|)\big) = \text{DSPACE}\big(f(|x|)\big)$

(ii) $\lim\limits_{|x| \to \infty} \dfrac{f(|x|)}{|x|^2} \to \infty$ *implies* DTIME $\big(f(|x|)\big) = \text{DTIME}\big(df(|x|)\big)$

The proof of (3.7) is not difficult and entails essentially a change of basis. (One also needs to be careful with other models eg RAM's).

Actually, there are a lot of interesting results about varying the model of computation slightly: for instance suppose one allows 2 work tapes rather than one. Using the fact that space is reusable and by direct simulation, Hartmanis and Stearns [**1965**] showed.

Theorem 3.8. (i) *Any 2-tape T.M. computable in space $f(|x|)$ and in time $g(|x|)$ can be simulated by a 1-tape T.M. in the same space $f(|x|)$ and in time $\big(g(|x|)\big)^2$.*

The problem of whether the $\big(g(|x|)\big)^2$ was sharp for (e.g.) online Turing machines was open for quite a while. Paul [**1979**] was the first to show that two tapes were better than one. The complete answer was given for DTIME$(|x|^2)$ by Ming Li [**1985**] and Maass [**1984, 1985**]. We look at this in §6. For nondeterministic machines, the $\big(g(|x|)\big)^2$ bound is not known to be sharp, but using a very clever combinatorial argument based on "Kolmogorov Complexity" Maass [**1985**] showed that there were sets acceptable in time $|x|$ by a 2-tape online nondeterministic T.M. that were not acceptable in time $\big(n^2/(\log n)^2 \log\log n\big)$ by a 1-tape nondetermistic T.M. Using Maass's language, this was improved to $\Omega\big(n^2/\log^{(k)} n\big)$ (any k, where $\log^{(k)} n$ denotes k iterations of the log function) by Galil, Kannan and Szemeredi [**1986**].

The reader should note that since, by the union theorem there is a recursive function f such that PTIME $=$ DTIME$\big(f(|x|)\big)$, there must be functions such that no Turing machine (or even computation device) has running time exactly $f(|x|)$. $\big($Since otherwise $f(|x|)$ would need to be a polynomial$\big)$. Following Hartmanis and Stearns, we call a function f space (time) constructable if there is a machine M that runs in space (time) exactly $f(|x|)$ on input x.

Theorem 3.9. (Hierarchy theorems).

(i) (Hartmanis – Lewis – Stearns [**1965**]). *If t_2 is space constructable and* $\lim_{|x|\to\infty} \inf \frac{t_1(|x|)}{t_2(|x|)} = 0$ *then*

$$\text{DSPACE}(t_1(|x|)) \neq \text{DSPACE}(t_2(|x|)).$$

(ii) (Hartmanis) (Probably suboptimal). *If f_2 is time constructable and* $\lim_{|x|\to\infty} \inf \frac{t_1(|x|)}{t_2(|x|)} = 0$ *then* DTIME $(t_1(|x|)) \neq \text{DTIME}(t_2(|x|)^2)$

(iii) (Hopcroft-Ullman [**1969**]) (again probably suboptimal). *If t_2 is time constructable then* $\lim_{|x|\to\infty} \inf \frac{t_1(|x|)\log t_1(|x|)}{t_2(|x|)} = 0$ *then implies*

$$\text{DTIME}(t_1(|x|)) \neq \text{DTIME}(t_2(|x|)).$$

Proof Sketch. (i) The idea is to diagonalize against all Turing machines running in space $\leq t_1(|x|)$. First use $t_2(|x|)$ to map out the allowable work space. Then run each currently unsatisfied (ie. not yet diagonalized) $\phi_e(x)$ in space $t_2(|x|)$ until either run out of space or $\phi_e(x)\downarrow$, or a configuration is repeated. In the case that $\phi_e(x)\downarrow$, we cancel e the least such e define $g(x) = \phi_e(x)+1$. If no e is cancelled set $g(x)=0$. Note that $g(x) \in \text{DSPACE}(t_2(|x|))$ by construction, and if $\phi_e \in \text{DSPACE}(t_1(|x|))$ then eventually we get to diagonalize it, by the hypothesis that the $\liminf \to 0$.

We remark that (ii) and (iii) are proven by analysing time simulatations of the methods above. ∎

Probably the most important questions in complexity theory are the costs of time-space, determinism-nondeterminism trade offs. There are some obvious things:

Observations 3.10. (Space tradeoff).

(a) $\text{DTIME}(f(|x|)) \subseteq \text{DSPACE}(f(|x|))$

(b) If $t(|x|) > \log|x|$ then $\text{DSPACE}(t(x)) \subseteq \bigcup_{d\in\mathbb{N}} \text{DTIME}(d^{f(|x|)})$.

Proof Sketch. To see that (b) holds count the total number of possible configurations a Turing machine can generate in space $t(|x|)$. ∎

We remark that the precise relationship between DSPACE $(t(|x|))$ and DTIME $(t(|x|))$ seems to vary as t varies, and is of fundamental importance. For some t they coincide: Call f *elementarily time (computable)* if $f \in \cup_n \text{DTIME}(2^{2^{\cdot^{\cdot^{2^{|x|}}}}})$ (a tower of n twos). Then the result above shows that ELEMENTARY TIME = ELEMENTARY SPACE. With respect to the other basic tradeoff we have the following beautiful result of Savitch [**1970**].

Theorem 3.11. (Deterministic-Nondeterministic trade-off).

(i) *If t is space constructable, then $NSPACE(t(|x|)) \subseteq DSPACE\Big(\big(t(|x|)\big)^2\Big)$, and*

(ii) $NTIME(t(|x|)) \subseteq \bigcup\limits_{d \in \mathbb{N}} DTIME(d^{t(|x|)})$.

Proof Sketch. Note that (i) $\Rightarrow$ (ii) by the last theorem. The idea of the following is that of "divide and conquer", where we keep guessing and testing half way points of a computation. So suppose M accepts A nondeterministically in space $f(|x|)$. So there is a $d \in \mathbb{N}$ such that the number of possible configurations is $\leq d^{t(|x|)}$. If C_1 and C_2 are configurations we write if $C_1 \underset{i}{\vdash} C_2$ if C_2 can be obtained from C_1 in $\leq 2^i$ steps. If $i \geq 1$ then $C_1 \underset{i}{\vdash} C_2$ iff $(\exists C)(C_1 \underset{i-1}{\vdash} C$ and $C \underset{i-1}{\vdash} C_2)$. (Here C is a half way point.)

To see if $x \in A$ we,

1. Take C_0 and initial input x, and
2. for each possible final configuration C_e of M of length $\leq t(|x|)$ we see if $C_0 \underset{n}{\vdash} C_e$ for $n = (\log d)(t|x|)$ $\big($note that $2^n = 2^{(\log d)(t(|x|))} = d^{t(|x|)}\big)$ as follows:

If C_e is really correct $C_0 \underset{n}{\vdash} C_e$. How to test this is in a small space? We do this inductively:

(i) For $i = 0$, $C_0 \vdash C_e$ iff $C_0 = C_e$ or can be gotten in 1 step.

(ii) For $i \geq 1$, $C_0 \underset{i}{\vdash} C_e$ iff $C_0 \underset{i-1}{\vdash} C$ and $C \underset{i-1}{\vdash} C_e$ for some C of length $t|x|$.

Call an *i-module* as above $\boxed{C_1 \mid C_2 \mid i \mid C}$. Now C_1, C_2 have length $< t(|x|)$ and $i \leq n$ we then do inductive guessing so the tape at any stage appears as

$\dots$	C_1	C_2	i	C	C_1	C	$i-1$	C'	C_1	C_1'	$i-2$	C''	$\dots$
	i-module				$i-1$ module				$i-2$ module				

Then the idea is to start with C_1, C_2, i, C and in the next module cycle through all the possible C' to see if $C_1, C, C-1, C'$ is a legal $i-1$ module. To do this cycle for each C' we look at all possible $C-2$ modules etc. If we get to show $C_1, C, i-1$, via C' then we need to show $C', C_2, i-1, D$ for some D. This can be done in the same way. There are n modules with $n = (\log d)t(|x|)$ and each module of length $\leq 0(\log d)\big(t(|x|)\big)$ the total length is $0\big(t(|x|)\big)^2$. Hence by linear speed-up, DSPACE $\big(t(|x|)\big)^2 \supseteq$ NSPACE$\big(t(|x|)\big)$. ∎

Note that it is an immediate corollary to (3.11) (i) that PSPACE is NPSPACE. Another corollary to the hierarchy theorem is that LOGSPACE $\subsetneq$ PSPACE. Putting all of the facts together we have the following fundamental picture.

3.12. LOGSPACE $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE.

We know only that LOGSPACE $\neq$ PSPACE, no other containment is known to be proper. In the next section we will look at this in detail.

To finish this section we will look at another space result. If Q is any class, then co-Q is the class of sets whose complement is in Q. So co-NP is the class of sets A such that $x \notin A$ is NP. It is easy to see that co-P = P. It is not known, as we shall see, if NP = co-NP. Space is rather more tractable as the following recent results of Immerman [**1988**] and Szelepscenyi [**1988**] shows:

Theorem 3.13. (Immerman – Szelepscenyi). *For any space constructable f with $f(|x|) \geq \log|x|$,*

$$\text{co-NSPACE}(f(|x|)) = \text{NSPACE}(f(|x|)).$$

Remarks 3.14. The result above solves a 25 year old question and has a delightful short and subtle proof using the idea of a *census* function. This allows us to non-deterministically compute a *number* rather than accept a string. The proof is quite elementary and can be taught to a beginning graduate or advanced undegraduate class. For those who know some formal language theory, $A \in$ NSPACE($|x|$) iff A is *context free*. Hence *context free languages are closed under complementation.*

Proof of (3.13). (Sketch). Let $f(|x|) \equiv |x|$. Let M be a machine (acceptor) running in NSPACE ($|x|$) and accepting A. We desire to build a machine $\overline{M}$ in NSPACE ($|x|$) that accepts $\overline{A}$. The thing the reader must keep in mind in the following is that it does not hurt us for *some computation path* of $\overline{M}$ to reject x although $x \notin A$ provided that

(i) some computation path of $\overline{M}$ accepts such an x, and

(ii) no computation path accepts $y \in A$.

Suppose there was a machine $\overline{M}_1$, in NSPACE ($|x|$) such that, for each x, $\overline{M}_1$ could compute the *number* $n(x)$ of configurations M could reach. Then we could build $\overline{M}$ to be accept $\overline{A}$ as follows. For a given x, first use $\overline{M}_1$ to compute $n(x)$. Then cycle successively through each configuration σ and see if M has some computation path to get to σ. That is, to see if x is accepted by M start running through all possible (trial) configurations and keep a counter. To see if a given σ_1 is a final one, guess a computation path and see if this is a valid one leading to σ_1. If it does increment the counter by one. If the path accepts x declare $x \notin \overline{A}$. If we get through all the configurations and the counter is not yet at $n(x)$ then we made some wrong guess. Declare $x \notin \overline{A}$. However for some correct sequence of nondeterministic guesses we will have the counter at $n(x)$ and we will then know we have seen *all the final configurations from* x. Therefore we can say $x \in \overline{A}$, as we have not seen $x \in A$. Note that (i) and (ii) above are preserved by this procedure.

To complete the proof we need only say how to compute the 'census function' $n(x)$. This is done inductively. Let $d(x,t)$ be the number of configurations computable by $\overline{M}_1$ in t steps. We can compute $d(x,1)$ deterministically in $|x|$. Suppose M_1 can compute $d(x,t)$. We can get to σ in $t+1$ steps only if we can get to some σ_1 in t steps and then σ in one more. To compute $d(x,t+1)$ as we did before we cycle through all the sequences $\sigma_1, \sigma_2, \ldots$ of length $|x|$ and guess computation paths of length t to see if they can be reached in length t. If we find that σ_1 is so obtainable, we cycle through each of the strings of length $|x|$ and see which can be reached from σ_1 in one (more) step. For each with a yes, increment $d(x, t+1)$ by one. Using $d(x,t)$ and a counter we will know (at the end) if we $d(x,t)$ of them and hence all the sequences reachable in t steps. If we don't get them all reject x. If the counter agrees with $d(x,t)$ we know $d(x, t+1)$ is correct too.

Finally $n(x)$ is the value of $d(x,t)$ for the first t with $d(x,t) = d(x,t+1)$. ∎

4. P, NP, LOGSPACE and PSPACE

No doubt many of you have heard of the "famous" P vs NP problem, perhaps (before reading the present article!) without knowing what exactly was meant by it. One of the reasons it is so famous is because it seems very hard and fundamental. The other reason is that it seems so important in 'practical' applied mathematics.

From a pure mathematics point of view P vs NP strikes at the heart of our profession : consider a proof as a series of lines each an axiom or a consequence of proceeding lines. It makes sense, under appropriate coding, to speak of a polynomial size proof. P vs NP then boils down to the difference (if any) between *finding* a (polynomial size) proof and *verifying* a proof which is exhibited. If P = NP they are the same. At a more down to earth level, we also know of hundreds of practical problems that are in NP yet not in P if P $\neq$ NP. Indeed, we can, in a precise way, argue for most of these that if any of them are in P then all of them are. To make this precise we return to our recursion theory model, and look at *resource bounded reducibilities* between (codes of) combinatorial problems.

Definition 4.1. We say $A \leq_T^P B$, A is *p-time Turing reducible to* B if there exists an oracle machine M and a polynomial q such that for all x,

(i) $x \in A$ iff M with oracle B accepts x, and

(ii) the computation for x runs in time bounded by $q(|x|)$ where if the question $y \in B$? is asked, we take it to take $|y|$ many steps to answer. (Alternatively, one can use time 1 here. The current one is more natural if we look at space reducibility).

Actually, as with normal Turing reducibility, we often can use a far simpler, and stronger reducibility:

Definition 4.2.

(i) We say $A \leq_m B$ (A is many-one reducible to B) iff there is a recursive function f with $x \in A$ iff $f(x) \in B$

(ii) Analogously, for p-time, we say $A \leq_m^P B$ iff there is a polynomial time computable function f such that $x \in A$ iff $f(x) \in B$.

There are several other reducibilities we will later look at, but for the moment we will concentrate on $\leq_T^P$ and $\leq_m^P$. Clearly $A \leq_m B$ implies $A \leq_T B$ and it is not difficult to devise examples to show each converse fails. For the corresponding resource bounded reducibilities, to show that $\leq_m^P$ and $\leq_T^P$ differ on NP sets would imply P $\neq$ NP. On the other hand, if there is 'enough time' to diagonalize, we can show that these reducibilities really do differ.

Theorem 4.3. (Ladner, Lynch, and Selman [1975]). *$\leq_T^P$ and $\leq_m^P$ differ on* EXPTIME.

Proof Sketch. One needs to build $C \leq_T^P B$ with $C \not\leq_m^P B$ and C, B in EXPTIME. We diagonalize meeting the requirements.

$$R_e : \neg(C \leq_m^P B)\text{via}\gamma_e$$

with γ_e the e-th p-time unary function. To keep $C \leq_T^P B$ we declare that $0^n \in C$ iff $0^{2n} \in B$ or $0^{2n+1} \in B$. Then to meet one R_e alone we see what $\gamma_e(0^n)$ is. If $\gamma_e(0^n)$ is already in B, keep 0^n out of C. Otherwise put 0^n into C, and put 0^k into C with $0^k \in \{0^{2n}, 0^{2n+1})$ and $0^k \neq \gamma_e(0^n)$. It is easy to convert this into an argument that meets R_e for all e. ∎

Now for the r.e. sets we can define

$$K_0 = \{\langle x, y\rangle : x \in W_y\}.$$

It is clear that if A is r.e. then by the enumeration Theorem (2.1), $A = W_y$ for some y. Hence $x \in A$ iff $\langle x, y\rangle \in K_0$. It follows that K_0 is *m-complete* (for r.e. sets) in the sense that if A is r.e., $A \leq_m K_0$. (Actually it is not difficult to show that $K_0 \equiv_m K$ where $K = \{x : \phi_x(x)\downarrow\}$).

Now let us begin looking at analogues for NP, modelling NP on 'r.e.' and P on 'recursive'. First we know by (2.6) that an infinite A is r.e. iff A is the range of a $(1-1)$ recursive function iff there is a recursive R with $x \in A$ iff $(\exists y)\big(R(x, y)\big)$. The analogy for NP is as follows. Call a partial p-time function f *honest p-time* if there is a polynomial q such that for all $x \in \operatorname{dom} f$, $q\big(|f(x)|\big) > |x|$. The intuition here is that f cannot *shrink* the input more than a polynomial amount.

Remark. We already know f cannot stretch the input more than a polynomial amount as f is p-time. The reason for taking this definition is that one can show that for any infinite r.e. set W there is a polynomial time computable function with $ra\, g = W$. This uses the idea of *padding* : we know there is a recursive function f with $ra\, f = W$. Now g is a 'slow' version of f compute $f(0)$. Define $g(x) = f(0)$ for all x with $|x| < s$ until a stage s occurs with $f^s(1)\downarrow$. Note as f is $1-1$, $f(0) \neq t(1)$. Now define $f(y) = t(1)$ for all y with $s \leq |y| < t$ until $t > s$ is found with $f^t(2)\downarrow$. Continue in the obvious way. Note that this p-time function g can be very *dishonest*. We have:

Theorem 4.4. (Karp, Cook). *The following are equivalent:*

(i) $A \in \mathrm{NP}$

(ii) *There is a partial honest p-time function g such that $x \in A$ iff $x \in ra\, g$.*

(iii) *There is a polynomial time relation R and a polynomial q such that $x \in A$ iff $(\exists y)\big(|y| < q(|x|)$ and $R(x, y)\big)$. (This last condition is written $\exists^P y R(x, y)$.)*

Proof Sketch. (i) $\Rightarrow$ (ii). $A \in \mathrm{NP}$ if we have a p-time machine M such that some computation path of M accepts x in $< P(|x|)$ steps. So with a p-time bijection $\langle\ ,\ \rangle : \mathbb{N} \times \mathbb{N} \to N$, define $f(\langle x, y\rangle) = x$ if y codes an accepting computation path of M of length $< p(|x|)$, and $\langle x, y\rangle \notin \operatorname{dom} f$ otherwise. Then f is polynomial honest and $ra\, f = A$.

(ii) $\Rightarrow$ (iii). Let $A = ra\, f$ with f p-honest. So $x \in A$ iff $(\exists y)\big(f(y) = x\big)$. This is of the form $(\exists^P y)\big(R(x,y)\big)$ as f is p-honest.

(iii) $\Rightarrow$ (i) $x \in A$ iff $(\exists^P y)\big(R(x,y)\big)$. Then build the nondeterministic T.M. M to accept A as follows:

(a) guess y with $|y| < p(|x|)$.

(b) see if $R(x,y)$ holds and accept iff $R(x,y)$ holds.

Definition 4.5. (Cook, Karp).

(i) We say a set A is NP-*T-complete* if $A \in$ NP and for all $B \in$ NP, $B \leq_T^P A$.

(ii) We say A is NP-m-complete if $A \in$ NP and for all $B \in$ NP, $B \leq_m^P A$.

Again following the r.e. vs recursive analogy we have.

Theorem 4.6. (Cook [**1971**]). *NP-m-complete sets exists. That is, there is $K_p \in$ NP such that $B \in NP \Rightarrow B \leq_m^P K_p$.*

Proof. Define, as with the r.e. sets.

$$K_p = \{\langle x, e, 0^n\rangle;\ \text{some computation of } \Phi_e \text{ on } x \text{ accepts in } \leq n \text{ steps}\}.$$

Here $\{\Phi_e\}_{e\in\mathbb{N}}$ lists all nondeterministic machines. Now $K_p \in$ NP: to see if $y \in K_p$ first check if y is of the form $\langle x, e, 0^n\rangle$. This is p-time. If so, guess a computation z of length $\leq n$ and see if it accepts x. To see that K_p is m-complete, $A \in$ NP iff for some Φ_e and polynomial $q, x \in A$ iff some computation of Φ_e halts in $\leq q(|x|)$ steps. Thus $x \in A$ iff $\langle x, e, 0^{q(|x|)}\rangle \in K_p$, so $A \leq_m^P K_p$. ∎

Usually the phrase NP-complete means NP-m-complete and we adopt this convention.

We will now look at practical instances of the structural results above. In many ways it is the fact that NP-complete problems are so common in real life that makes the P vs NP question so compelling. If any of these NP-complete problems is in P so too are all of them. It should be remarked that the milestone papers in this area were the fundamental one of Cook [**1971**] who proved the first real NP complete result, and of Karp [**1973**] who discovered a "compelling mass" of NP problems. We remark that the class NP also implicit in earlier work of Edmonds [**1965**] and Cook's Theorem 4.7 below is also implicit in the work of Levin [**1973**] where it was called *perebor*, and a number of Russians' work tried to show that there were some NP problems that could only be solved via trial and error (exhaustive search) (see Trachtenbrot [**1984**]). We also know that in 1957, Gödel raised this issue in a letter to John von Neumann (see Hartmanis [**198+** (in particular EATCS, **1989**)]).

The first practical problem proven to be NP-complete was that of SAT or satisfiability of a propositional formula. We recall that a formula ϕ of propositional calculus is built up from connectives, say, $\vee$ (or), $\wedge$ (and), $\neg$ (not) and variables $x_1, x_2, \ldots$. Then ϕ is satisfiable if there is an assignment of its variables that makes it true. The obvious way to check if ϕ is satisfiable is to draw up its truth table. This takes exponential time in the number of variables. Note that being satisfiable is in NP, since we can *guess* a verifying line of the truth table. So to summarize, in the notation of Garey and Johnson [**1979**]:

Problem	SAT
Instance:	A formula $\phi(x_1, \dots, x_n)$
Question:	Is ϕ satisfiable?

Cook's Theorem 4.7. (Cook [1971]). SAT *is* NP-*complete.*

Proof. Let M be a N.T.M. computing K_p with bound $p(|x|)$. We come up with a formula ϕ that "simulates M". Clearly we need only specify the tape contents from $-p(|x|)$ to $p(|x|)+1$ where we take as the origin the initial head position. Let M have states $\{q_0, \dots, q_r\}$ and symbols $\{s_0, \dots, s_t\}$. We use these to generate propositional variables defined on M : Let $n = |x|$

Variable	**Range**		**Meaning**
$Q(i,k)$	$0 \le i \le P(n)$, $0 \le k \le r$	$0(P(n))$	At time i, M is in state q_k.
$H(i,k)$	$0 \le i \le P(n)$, $-p(n) \le k \le p(n)+1$	$0(P(n))^2$	At time i, the head is scanning square k.
$S(i,j,k)$	$0 \le i \le P(n)$, $-p(n) \le g \le p(n)+1$, $0 \le k \le t$	$0(P(n))^2$	At time i, the contents square j is symbol s_k.

The idea is to derive a formula based on the above such that M has an accepting computation iff ϕ is satisfiable. To do this we need to write down a formula saying what it means to be a valid computation of M. These we divided into 6 groups which say.

(i) At each step M is in one and only one state eg: for $0 \le i \le P(n)$, $Q(i,0) \vee \dots\dots Q(i,t)$ and $\bigwedge_{k \neq i} \neg(Q(i,h) \wedge Q(i,k))$.

(ii) At step i the head is scanning one and only one square.

(iii) At step i the content of square j is exactly one symbol.

(iv) The initial set up is q_0 reading the leftmost symbol k_1 and $x = k_1 \dots k_n$ is written on the tape.

(v) What it means to be an accepting computation.

(vi) How can we get the next move? That is

(a) If the head at time i is not reading square j, do nothing to it. (e.g. $S(i,j,k) \wedge \neg(H(i,j)) \to S(i+1,j,k)$) for each i,j,k.

(b) If the head at i is reading j and the state at i is k and the symbol on j is ℓ, then do one of what the relevant quadruples tells you to do.

When the above is all written out, we see the $\phi(x)$ so derived is $O(|x|^4)$, and $\phi(x)$ is satisfiable iff M accepts x. ∎

In many ways SAT is the basis for most practical NP-completeness results. The above techniques are called *generic reduction*. Usually if we want to show B is NP-complete we show some known NP-complete A, $A \leq_m^P B$. We illustrate this by a couple of examples. First we need a refinement of (4.7). We say ϕ is in conjunction normal form (CNF), if $j = \gamma_1 \wedge \ldots \wedge \gamma_n$ with each γ_i a disjunction of literals (i.e. variables or their negations). For example, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3)$ is in CNF. By de Morgan's laws all γ can be written in CNF. Finally we say that ϕ is in n-CNF if the largest disjunction in ϕ has size n. The ϕ given above is in 3-CNF. While it can be shown that to decide if a ϕ in 2-CNF is satisfiable is in P, Cook found a reduction from SAT to show

Corollary 4.8. (Cook [**1971**]). *The following problem,* 3-CNFSAT *or* 3-SAT, *is* NP-*m-complete.*

Instance: *A given ϕ in* 3-SNF
Question: *is ϕ satisfiable?*

Let G be an (undirected) graph. We say G is *complete* if for all vertices $x, y \in G$, (x, y) is an edge of G. The following is a well known problem called CLIQUE.

Instance: A graph G and an integer k
Question: Does G have a clique of size $\geq k$? (That is, a complete subgraph on $\geq k$ vertices).

Theorem 4.9. (Cook [**1971**]). CLIQUE *is* NP-*m-complete.*

Proof Sketch. We show 3-SAT $\leq_m^P$ CLIQUE. Let $\phi = \bigwedge_{i=1}^{k} \bigvee_{v=1}^{3} \sigma_{ij}$ be an instance of 3-SAT, with $\sigma_{ij} \in \{x_1, \neg x_1, \ldots, x_m, \neg x_m\}$, we generate an instance of CLIQUE as follows:

Graph: $G(\phi)$
Vertices: $\{(\sigma_{i,j}; i) : 1 \leq i \leq k, 1 \leq j \leq 3\}$
Edges: $\{((\sigma, i), (\gamma, \phi)) : \sigma \neq \neg\gamma$ and $i \neq j\}$
Clique size k

It is not too hard to show that ϕ is satisfiable iff $G(\phi)$ has a clique of size $\geq k$. ∎

Some other natural NP-m-complete problems are given below.

HAMILTON CIRCUIT. (Karp [**1972**]).
Instance: A graph G
Question: Does G have a Hamilton cycle? That is, is there a closed path passing through all the vertices of G exactly once?

TRAVELLING SALESPERSON. (Karp [**1972**]).
Instance: Graph G with weighted edges
Question: Find a minimum cost tour
Remark: This is not a decision problem, but is NP-complete by iterated use of the NP-complete decision problem: Does G have a tour of weight m?

KNAPSACK. (Karp [**1972**]).

Instance: Given $\{x_1, \dots, x_n\} \subseteq \mathbb{N}$ and $m \in \mathbb{N}$

Question: $\exists A \subseteq \{1, \dots, n\}$ with $\sum_{i \in A} x_i = m$?

PLANAR GRAPH 3-*COLORABILITY*. (Stockmeyer [**1973**]).

Instance: A planar graph G

Question: Can G be 3-coloured?

Note that recent work of Appel and Haken shows that any planar G can be 4 coloured in polynomial time.

QUADRATIC DIOPHANTINE EQUATION. (Manders and Adleman [**1978**]).

Instance: Positive integers a, b, c.

Question: Do there exist positive integers x, y with $ax^2 + by = c$.

COSINE PRODUCT INTEGRATION. (Plaisted [**1976**]).

Instance: Sequence $(a_1, \dots, a_n)$ of integers.

Question: Does $\int_0^{2\pi} \prod_{i=1}^n \cos(a_i\theta) d\theta = 0$?

It should be noted that all proofs of NP-completeness above show that in fact the sets are NP-m-complete. *Indeed, the phrase NP-complete is now taken to mean* NP-m-*complete* as we mentioned earlier. Furthermore all of the known NP-complete sets exhibit deep similarities. This leads to a conjecture of Berman and Hartmanis [**1977**]: If A and B are NP-complete then $A \equiv_1^P B$. *That is there is a polynomial time bijection of* $\{0,1\}^*$ *taking A to B (we say A and B are p-isomorphic).*

Of course since P = NP implies that there are finite NP-complete sets, the Berman-Hartmanis conjecture implies P $\neq$ NP. Again despite a lot of work, very little progress has been made on this question. It is not known if the conjecture and P = NP are equivalent. We do know that the failure of a weaker incarnation of this conjecture does imply P = NP. We call a set A *sparse* if there is a polynomial p such that for all $n \geq 0$, $|\{x \in A : |x| \leq n\}| < p(n)$. For various technical reasons, sparse sets occupy a central place in structural complexity. One such reason for this is the following.

Theorem 4.10.

(i) (Berman and Hartmanis [**1977**]). *No sparse set is p-isomorphic to* SAT.

(ii) (Berman [**1977**]). *If* NP *has a sparse complete set, then* P = NP.

The intuition here is that a sparse set cannot contain enough information to decode all NP sets. The proofs are a little technical so we omit them. We can pursue the notion of m- and T- completeness quite a bit further, and see if analogues of the Berman – Hartmanis conjecture hold for other classes. Here the best results so far are

Theorem 4.11.

(i) (Berman [**1977**], Watanabe [**85**]). *Let D be any 'reasonable' complexity class that is specified by deterministic Turing machines and has the property that for some $k > 0$, $\cup_c\{\mathrm{DTIME}(2^{c(n^{\frac{1}{k}})} : n \in \mathbb{N}\} \subseteq D$. Then all $\leq_m^P$ complete sets for D are $\equiv_{1,\ell i}^P$ equivalent where $A \equiv_{1,\ell i}^P B$ means that there exist $\leq_m^P$ reductions from A to B and B to A that are both one to one and length increasing).*

(ii) (Ganesan and Homer [**1989**]). *If D is any reasonable class specified by nondeterministic machines and has* DEXT $\subseteq D$*, then all* $\leq_m^P$*-complete sets for D are* $\equiv_{1,\ell i}$ *equivalent.*

We should remark that quite aside from problems such as Travelling Salesperson which need to be constantly solved for many real life applications of operations research, there is a whole host of other applications which have NP-complete problems at their hearts. We will keep ourselves to one further example. The readers may be aware of various "public key" and other encryption techniques. The basic idea is that we wish to send a message from A to B so that it cannot be decoded unless one possesses "a key". One of the main ideas in many such codes is to use an NP complete problem (or, rather one similar to an NP-complete problem) as the device so that the key is the solution to some NP problem which, if P $\neq$ NP cannot be *found* quickly but can be used (checked) quickly. Space does not permit discussion of how this is achieved and the reader is referred to the excellent text Salomaa [**1985**, Chapter 7] for a leisurely account of the basics of the area. In particular there the author discusses systems based on the KNAPSACK problem. Recently it has been shown that – in some strong sense – safe encryption systems exist iff P $\neq$ NP.

It should be remarked that this application of the theory basically relies on the assumption P $\neq$ NP, or at least on the assumption that if P = NP then there is no [really] computer-feasible algorithm to do the relevant NP-complete problem. (For instance if the algorithm is $O(n^{10^{100}})$ although polynomial it wouldn't be much good!). Applications include, for instance, encryption systems used for sending cable T.V. channels from satellites. Here the code is changed very quickly and often, and the "pseudo intractability" of the techniques used means that, in theory at least, no pirate decoding device can be used.

Turning to PSPACE, again we find several natural PSPACE complete problems. In the place of SATISFIABILITY,

QUANTIFIED BOOLEAN FORMULA. (QBFSAT) (Stockmeyer & Meyer [**1973**]).
Instance: A quantified boolean formula of the form
$\phi = (Q_1x_1)\dots(Q_nx_n)E$ where E is a Boolean expression involving variables $x_1,\dots,x_n$ and Q_i is either $\forall^P$ or $\exists^P$ (n arbitrary here).
Question: Is ϕ valid?

Actually the above is related to a refinement of PSPACE called the POLYNOMIAL HIERARCHY. We know that A is NP if there is a p-time relation R such that $x \in A$ iff $(\exists^P y)\big(R(x,y)\big)$ (Theorem (4.4)).

We say that an NP set is $\sum_1^P$. This indicates one alternation of quantifiers beginning with an existential one. Similarly a co-NP set is one of the form $(\forall^P y)R(x,y)$. This is called Π_1^P. In general we can continue the definition so that a $\sum_3^P$ expression would be of the form

$$\underbrace{\exists^P\vec{x}_1\forall^P\vec{x}_2\exists^P\vec{x}_3}_{\text{3 alternations}} R(x,\vec{x}_1,\vec{x}_2,\vec{x}_3).$$

or by coding

$$\exists^P x_1\forall^P x_2\exists^P x_3\widehat{R}(x,x_1,x_2,x_3).$$

If an expression is both $\sum_n^P$ and Π_n^P we call it Δ_n^P. This allows us to define the polynomial hierarchy as

$$\begin{tikzcd}
 & \Sigma_1^P \arrow[dr, dash] & & \Sigma_2^P \arrow[dr, dash] & \\
P=\Delta_0^P=\Sigma_0^P=\Pi_0^P-\Delta_1^P \arrow[ur, dash] \arrow[dr, dash] & & \Delta_2^P \arrow[ur, dash] \arrow[dr, dash] & & \cdots \\
 & \Pi_1^P \arrow[ur, dash] & & \Pi_2^P \arrow[ur, dash] &
\end{tikzcd}$$

Note that PSPACE $\supseteq$ PH $= \cup_n \sum_n^P$.

Now we don't know if *any* of the inclusions above are proper, but we do know that if $\sum_{n+1}^P = \Delta_n^P = \Pi_{n+1}^P$ then for all $m \geq n$, $\sum_m^P = \sum_n^P$ (collapse propagates upwards). Nevertheless, most workers believe that the polynomial hierarchy is infinite with essentially the same conviction they believe P $\neq$ NP.

Quite a number of PSPACE complete problems have been discovered. These occur in for instance, game theory (e.g. the so-called "pebble games" Pippenger [**1980**]), automata theory and logical theories. As an example, for the readers familiar with automata theory the following is PSPACE-complete.

REGULAR EXPRESSION NON-UNIVERSALITY. (Aho, Hopcroft, Ullman [**1974**]).

Instance:	A regular expression α over alphabet $\sum$
Question:	Is $L(\alpha) \neq \sum^*$.

For the other classes we have looked at so far (P and LOGSPACE), the situation is slightly different. Here the reduction $\leq_T^P$ or even $\leq_m^P$ is rather useless since it is too *coarse* a measure of complexity. We refine our reduction to $\leq_{\text{LOG}}$, LOGSPACE – reducibility. This turns out to be a reducibility (transitivity is not obvious) and seems the appropriate measure for P. The first log space complete problem (for P) found was

PATH SYSTEM ACCESSIBILITY. (Cook [**1974**]).

Instance:	A relation $R \subseteq X \times X \times X$ and two sets $S, T \subseteq X$.
Question:	Is there an 'accessible' member of T?

Here $x \in X$ is accessible iff $x \in S$ or there exist accessible y, z with $\langle x, y, z\rangle \in R$.

We will not look at the structure of P under LOGSPACE reducibility in detail, but mention that $\leq_{\text{LOG}}$ is strongly related to circuit complexity. We look briefly at circuit complexity in §6. Ladner [**1975b**], for instance, showed the natural circuit evaluation problem is LOGSPACE complete for P. (See also Minyano, Shiraishi, and Shoudai [**ta**]). Also by Goldschlager, Shaw and Staples [**1982**] the maximum flow problem is LOGPACE complete for P. As a final comment on LOGSPACE and P, we remark that as an analogy with the polynomial time hierarchy, we can define a LOGSPACE hierarchy, and NL (nondeterministic LOGSPACE). Again, it is open if NL $\subseteq$ LOGSPACE, but as Cook observed, we have the following.

Theorem 4.12. (Cook [1974]). NL $\subseteq$ P.

Proof. Suppose f is accepted by a N.T.M. in space $t(|x|) \leq \log(|x|)$. There as at most $c^{t(|x|)}$ many configurations for some constant c, and in polynomial time we can enumerate all of them since $t(|x|) \leq \log(|x|)$. ∎

We remark that it is open if DSPACE $\left((\mathrm{LOG}|x|)^2\right) \subseteq P$. Again, we note that if NL is $\sum_1^L$ and we thus get the logspace hierarchy. However, by the Immerman-Szelepcsenyi result, we know that NL = co − NL, so the heirarchy behaves quite differently from the space one. (Assuming NP $\neq$ co-NP).

At this stage we'd also like to mention that there are many other approaches to studying the structure of P. Most of these depend on specifying a model of computation. For instance, Maass and Slaman [**ta**] fix the model as a RAM (random access machine). This is similar to a Minsky machine. The memory of a RAM consists of a sequence $r_0, \ldots, r_n, \ldots$ of *registers* each capable of holding an arbitrary integer. Let $\langle i \rangle$ denote the current contents of r_i. All arithmetical operations take place in r_0. Then n input numbers are located in $r_1, \ldots, r_n$. Then a RAM consists of a finite set of instructions of the form: put k into r_0, put $\langle k \rangle$ into r_0, put $|\langle k \rangle|$ into r_0, put $\langle r_0 \rangle$ into r_k, put $\langle r_0 \rangle$ into r_i where $i = |\langle r_0 \rangle|$, $\langle 0 \rangle := \langle 0 \rangle + \langle k \rangle$, $\langle 0 \rangle := \langle 0 \rangle - \langle k \rangle$ and, if $\langle 0 \rangle > 0$, go to j.

As we said earlier, a RAM is a very good model for practical purposes as it has running time quite similar to that of a real machine. Maass [**1988**] has proven nonlinear bounds for some NP-complete-problems on a RAM.

Maass and Slaman [**ta**] define $A =_C B$ ("A and B have the same time complexity") iff for all time constructable $t, A \in \mathrm{DTIME}_{\mathrm{RAM}}(t) \Leftrightarrow B \in \mathrm{DTIME}_{\mathrm{RAM}}(t)$. The equivalence classes are called *complexity types*. Maass and Slaman observe that $=_C$ provides an incisive tool with which to analyse, for instance, the structure of P. They show, among other things that

Theorem 4.13. (Maass and Slaman [**ta**]). *A complexity type Q contains sets A, B incomparable with respect to $=_C$ iff $Q \not\subseteq P$.*

They also use $=_C$ to analyse the structure of P and indeed show that each complexity type $Q \not\subseteq \mathrm{DTIME}_{\mathrm{RAM}}(n)$ has a very rich structure. Furthermore Maass and Slaman establish that not all complexity types are isomorphic. The proofs are very sophisticated arguments along the lines of the Blum speedup theorem. Much work remains to be done here.

We finish this section with a brief discussion of 'upward translation' theorems: namely those that show that collapse of nondeterminism propagates upwards. Define for $n \geq 1, \mathrm{DEXT}_n = \cup_{c>0}\mathrm{DTIME}(2^{2^{\cdot^{\cdot^{2^{c|x|}}}}})$($n$ times) and similarly $\mathrm{EXP}_n\mathrm{SPACE}$ and NEXT_n, etc. We have the following relationships.

$$\begin{tikzcd}
\mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \arrow[r] \arrow[d] & \mathrm{PSPACE} \arrow[r] \arrow[d] & \mathrm{EXT}_2 \arrow[r] \arrow[d] & \mathrm{NEXT}_2 \arrow[r] \arrow[d] & \mathrm{EXP}_2\mathrm{SPACE} \arrow[r] \arrow[d] & \ldots \arrow[d] \\
\mathrm{EXP} \arrow[r] & \mathrm{NEXT} \arrow[r] & \mathrm{EXSPACE} \arrow[r] & \mathrm{EXT}_2 \arrow[r] & \mathrm{NEXT}_2 \arrow[r] & \mathrm{EXP}_2\mathrm{SPACE} \arrow[r] & \ldots
\end{tikzcd}$$

$\mathrm{PSPACE} \subsetneqq \mathrm{EXPSPACE} \subsetneqq \mathrm{EXT}_2\mathrm{SPACE} \subseteqq \ldots$

$\mathrm{P} \subsetneqq \mathrm{EXT} \subsetneqq \mathrm{EXT}_2 \subseteqq \ldots$

$\mathrm{NP} \subsetneqq \mathrm{NEXT} \subsetneqq \mathrm{NEXT}_2 \subsetneqq \ldots$

we have

Theorem 4.14. (Book [**1974**]). *$P = NP$ implies $EXT_n = NEXT_n$ all $n > 1$ and $NP \subseteq EXT$.*

Proof. Suffices to show that P = NP $\Rightarrow$ EXT = NEXT. So suppose EXT $\neq$ NEXT, and let $A \in$ NEXT $-$ EXT. Let $2^x \in B$ iff $x \in A$. It is not hard to see that $B \in$ NP since $A \in$ NEXT. Now if $B \in P$ then $A \in$ EXT, since then $x \in A$ can be settled in time polynomial in $|2^{|x|}| = 2^{|x|}$ and hence exponential in $|x|$. ∎

Hartmanis, Immerman and Sewelson [**1985**] have shown that *there are sparse sets in* NP $-$ P *iff* EXT $\neq$ NEXT and Selman [**1979**] observed that *if* EXT $\neq$ NEXT *then $\leq_m^P$ and $\leq_T^P$ differ nontrivially on* NP. Finally, Book observed that via the same technique as (4.14).

Theorem 4.15. (Book [**1974a**]).

$$\begin{aligned}
&\text{(a) P} = \text{PSPACE} \Rightarrow \text{EXP}_n\text{SPACE } \textit{for all } n > 0\\
&\qquad\qquad\qquad\quad \Rightarrow \text{PSPACE} \subseteq \text{EXT}\\
&\text{(b) NP} = \text{NPSPACE} \Rightarrow \text{NEXT}_n = \text{NEXTP}_n\text{SPACE}\\
&\qquad\qquad\qquad\qquad \Rightarrow \text{PSPACE} \subseteq \text{NEXT}.
\end{aligned}$$

As we see later, there is no known 'downward translation' result and we have evidence that it is possible for (e.g.) EXT = NEXT yet P $\neq$ NP. We return to this when we discuss relativisations.

5. The Structure of NP $(\text{REC}, \leq_T^P)$ and Recursion-Theoretic Techniques

If we pursue our earlier analogy, we see that r.e. corresponds to NP and recursive corresponds to P. The fundamental problem P = NP? shows us how we don't as yet have an analogy for the halting problem. Nevertheless we should pursue the formal analogy rather further. First equivalence classes under $\leq_m^P$ (resp $\leq_T^P$) would be call *polynomial time m-(resp T) degrees*. We have seen that if P $\neq$ NP as we believe, there are at least two degrees of NP sets. Namely the degree of the NP-complete sets and that of the p-time sets. Ladner [**1975a**] introduced a fundamental technique (delayed diagonalization) that yields a plethora of other degrees (in NP, if P $\neq$ NP).

Theorem 5.1. (Ladner [**1975a**]). *The p-T-degrees and the p-m-degrees of recursive sets are dense. That is if A and B are recursive with $A <_T^P B (A <_m^P B)$ then there is a C with $A <_T^P C <_T^P B$. (resp. $A <_m^P C <_m^P B$).*

Proof Sketch. We prove this for p-T-degrees. Let A and B be recursive with p-time functions $f(\mathbb{N}) = A$ and $g(\mathbb{N}) = B$ ($\mathbb{N}$ in unary notation). We can write $A_s = \{f(0), \ldots, f(s)\}$ and $B_s = \{g(0), \ldots, g(s)\}$. Let $\{\langle \Phi_e, p_e \rangle : e \in \mathbb{N}\}$ enumerate all pairs consisting of an oracle T.M. and a polynomial. By convention we will assume that $\Phi_e(\ ; x) \downarrow$ in $\leq p_e(|x|)$ many steps. We need to meet for all $e \in \mathbb{N}$

$$R_{2e} : \Phi_e(C) \neq B$$
$$R_{2e+1} : \Phi_e(A) \neq C.$$

Now we meet these in order. We begin with R_0. We therefore desire to build C so that $\Phi_e(C) \neq B$. We know how to do this : if $C = A$ then it must be that $\Phi_e(C) \neq A$ as $A <_T^P B$. The basic action is then as follows at stage 0, we declare that $C(x) = A(x)$ for all x with $|x| = 0$. At stage 1 we declare $C(x) = A(x)$ for all x with $|x| = 1$. We continue in the obvious way. Note that at stage 10, say we may have declared that $C(x) = A(x)$ for all x with $|x| \leq 10$ *we may not know what $A(1^{10})$, for instance, is until a much later stage. This is because A may be only enumerated "slowly". Nevertheless the reduction is p-time.* Now the idea is to wait till we see a stage t_0 and a string such that

5.2. $\Phi_{0,t_0}(C_{t_0} : y) \downarrow\neq B_{t_0}(y)$, and furthermore $p_e(|y|) < t_0$ and for all z if $|z| \leq \max\{|y|, p_O(|y|)\}$ then $z \in C$ iff $z \in C_{t_0}$ and $z \in B$ iff $z \in B_{t_0}$.

Such a stage must occur and we can recognise it. At this stage, we realize, by the use principle, that if we don't change $C[p_O(|y|)]$ then we must have diagonalized against B forever. So now we can move on to R_1. For the sake of R_1 we now make C "look like" B (instead of A). So for stages $t \geq t_0 + 1$, we now declare $C_t(z) = B_t(z)$ for all z with $|z| = t$, till "looking back" we see a stage t_1 there is a y with

5.3. $\Phi_{o,t}(A_t; y) \neq B_{t_1}(y) = C_{t_1}(y)$ and A_t is now correct on $p_e(|y|)$ and $B_t(y) = B(y)$.

Again a stage t_1 occurs eventually lest $B \leq_T^P A$. After t_1 is found we are free to move on to R_2 and so go back to setting $C(z) = A(z)$.

The final set C we get is this broken into pieces $P_0, P_1, P_2, P_3, \ldots$ which alternatively look "like A" or "like B". It is easy to see that this set will have a strictly intermediate degree between A and B by construction. ∎

A large number of variations of the construction above have been analysed. Lader [**1975a**] showed how to embed various lattices into the p-T-degrees of recursive sets and Ambos-Spies [**1984**] observed that the method extends to all finite distribute lattices. However Ambos-Spies [**1984**] observed that there are limitations to the technique since it must give a distributive embedding due to technical considerations. The true complexity of the structure of the p-T-degrees of recursive sets was found by Shinoda and Slaman [**1990**] who extended an earlier technique of Ambos-Spies [**1984**] to show that the structure of the p-T-degrees is "as complicated as possible". (Namely its theory is as complex as first order arithmetic). The technique is to combine the Π_2 priority method with a 'speedup' technique along the lines of Blum's theorem. It was also used by Downey [**ta**]. A limitation of this technique is that it involves the construction of *nonelementary sets* and so has nothing to say about the structure of NP assuming P $\neq$ NP It seems hard to reduce this to smaller complexity classes such as EXT, say. Ambos-Spies and Nies [**ta**] have recently obtained similar information about the structure of the p-m-degrees, using earlier work of Herrmann on the lattice of r.e. sets.

The result above leads to the question of the existence of 'natural' intermediate problems. That is is there any natural problem which, assuming NP $\neq$ P, is neither p-time nor NP-complete? The answer is, "we think so". To make our notions precise, we need the analogue of another concept from recursion theory. Let NP^A

denote the sets B with B accepted by some nondeterministic p-time machine with oracle A. Clearly $\mathrm{NP} \subseteq \mathrm{NP}^A$. We can similarly define $\sum_n^{P,A}$ for any level of the p-time hierarchy.

Definition 5.4. (Schöning [**1983**]).

We call a set A *low*$_i$ if $\sum_i^{P,A} = \sum_i^P$. In particular if $i = 1$ we call A *low*. Similarly we call a set A *high*$_i$ if $\sum_i^{P,A} = \sum_{i+1}$. The structure $\mathrm{low}_1 \subseteq \mathrm{low}_2 \subseteq \dots$ (in NP) called the *low hierarchy* for NP.

Now while it is easy to show that if A is r.e. and co-r.e. then A is recursive, we do not believe that the analogous statement is true for NP. Namely we believe that $\text{co-NP} \cap \mathrm{NP} \neq \mathrm{P}$. What we do know is

Theorem 5.5. (Schöning [**1983**], Ko and Schöning [**1985**]).

(i) *If A is $\leq_T^P$ complete for* NP, *then A is* High_0.

(ii) *If A is sparse and in* NP *then A is* low_2.

(iii) *If $B \leq_T^P A$ and A is sparse in* NP, *then B is low_3.*

Long [**1985**] extended (ii) to show that if A is sparse in $\sum_i^P$ then $\sum_i^{P,A} \subseteq \Delta_{i+1}^P$. Pushing these ideas a lot further, Balcazar, Book and Schöning [**1986**] proved that the polynomial time hierarchy is infinite iff there is a sparse set relative to which it is infinite iff it is infinite relative to all sparse sets.

We remark that if we look at other operators we do have absolute results. For instance, call A *exponentially low* if $\mathrm{EXT(A)} = \mathrm{EXT}$. It has been shown that there exists a sparse set in EXT-P that is exponentially low (Book, Orponen, Russo and Watanabe [**1988**]).

What does (5.5) have to do with practical problems? One of the classical problems listed as an open problem in Garey and Johnson [**1979**] is the following.

GRAPH ISOMORPHISM.

Instance:	Two graphs G_1 and G_2
Question:	Are G_1 and G_2 isomorphic?

It is easy to see that this problem is in NP: just guess an isomorphism and verify it in p-time. The reason we do not believe it is NP-complete is:

Theorem 5.6. (Schöning [**1987a**]). *GRAPH ISOMORPHISM is* Low_2. *Hence if GRAPH ISOMORPHISM is* NP*-complete, then the polynomial time hierarchy collapses to the second level.*

We remark that there are a number of other well known open problems here. Two of them are

PRIME NUMBER.

Instance:	$n \in \mathbb{N}$.
Question:	Is n prime?

FACTORIZATION.

Instance:	$n \in \mathbb{N}$
Question:	Find nontrivial factors if n is composite.

It has been shown that PRIME NUMBER is in NP $\cap$ co-NP and hence in low$_1$ (Pratt [**1975**]). In Miller [**1976**] it is shown that this problem is in P if one assumes the extended Riemann hypothesis.. The other problem is the basis of many public key encryption systems (e.g. "Trapdoor ones") and is widely believed to be intractable. It is really interesting that so much security is based on a problem not even known to be NP-complete.

We next look at some results that perhaps point at why P = NP is so hard. One of the basic principles of recursion theory is that of relativisation. Recall from §2 this meant roughly, for most statements true of oracle-free machines, they remain true of machines with oracles. An archetype here is the result that $\varnothing <_T \varnothing'$ and the relativisation is that $A <_T A'$ for any A. Roughly this says that "normal diagonalization arguments" relativize, as do "normal simulation arguments". We will show that this principle *fails* for resource bounded complexity and hence, in particular, P =?NP *cannot be settled by arguments that relativise.*

Theorem 5.7. (Baker, Gill and Solovay [**1975**]). *There exist recursive oracles B, A such that*

(i) $P^A \neq NP^A$

(ii) $P^B = NP^B$

Proof.

(i) We build a recursive set A and define a set $C = \{x : (\exists y)(|y| = |x| \text{ and } y \in A)\}$. Thus $C \in \mathrm{NP}^A$. We meet the requirements, for $e \in \mathbb{N}$,

$$R_e : \Phi_e(A) \neq C.$$

Here Φ_e is the e-th p-time reduction with polynomial $|x|^e + e = p_e$.

Assume we have already met R_j for $j < e$. To meet R_e, we find a number m much bigger than any seen so far in the construction, such that $2^m > p_e(m)$. Now we will have specified A_e (where $A = \cup_s A_s$). Now compute $\Phi_e(A_e; 0^m)$.

Case 1. $\Phi_e(A_e : 0^m) = 1$. Then we win by asking that for all z if $|z| < p_e(m)$ then $z \in A$ iff $z \in A_e$, and thus keeping 0^m out of C.

Case 2. $\Phi_e(A_e : 0^m) = 0$. During the computation of $\Phi_e(A_e)$ on imput 0^m we can address at most $p_e(|x|)$ may strings. Now as there are 2^m strings of length m, and $2^m > p_e(|x|)$. It follows that there is at least one string σ of length m not addressed in this computation. We add σ to A_e to make A_{e+1}, but otherwise add nothing. This puts 0^m into C, yet note that it will not affect the computations since σ was not addressed in the computation (by the use principle).

Thus in either case we can ensure that $\Phi_e(A; 0^m) \neq C(0^m)$, and hence meet all R_e by diagonalization, simply by making sure $m = m(e)$ is sufficiently large so as not to affect the R_j for $j < e$.

(ii) The set $K_p^B = \{\langle x, e, 0^m\rangle :$ some computation of $\Phi_e(B; x)$ of length $< n$ accepts $x\}$ is NP^B-m-complete. Thus it suffices to construct B with $K_p^B \in P^B$. Thus, *we define* B via $\langle x, e, 0^n\rangle \in B$ iff some computation of $\Phi_e(B; x)$ accepts in $< n$ steps.

It suffices to show that B is (inductively) well defined. Given $z = \langle x, e, 0^n \rangle$ to see if $z \in B$, we run $\Phi_e(B; x)$ only for fewer than n steps, and hence no string of length $> n$ is queried of B. Now since $|z| > n$ it follows that to decide if $z \in B$ we only use strings of length $< |z|$ and hence B is well defined. Thus $K_p^B = B$ hence $\mathrm{NP}^B = \mathrm{P}^B$. ∎

Hartmanis has always noted that in some sense (that we don't yet understand) the above constitute *weak independence results*; asserting that $\mathrm{P} = \mathrm{NP}$ is not provable in some (weak) proof system (*For Logicians*. It would be very interesting if, for instance, it would be shown that $\mathrm{P} = \mathrm{NP}$ is independent of say IZF. Hartmanis and Hopcroft [**1976**] noted that $\{B : B$ is recursive and $\mathrm{P}^B \neq \mathrm{NP}^B\}$ is Π_2^0 complete, but that $\{B : \mathrm{P}^B = \mathrm{NP}^B$ is provable in ZFC$\}$ and $\{A : \mathrm{P}^A \neq \mathrm{NP}^A$ is provable in ZFC$\}$ are both $\sum_1^0$. From this we can conclude that there are recursive sets A and B with $\mathrm{P}^A \neq \mathrm{NP}^A$ and $\mathrm{P}^B \neq \mathrm{NP}^B$, *yet neither result is provable in* ZFC).

Bennet and Gill extended (5.7) as follows:

Theorem 5.8. (Bennet and Gill [**1981**]). *If A is a random oracle, then $P^A \neq NP^A$. That is $P^A \neq NP^A$ with probability one.*

Theorem (5.8) led to the *random oracle conjecture*: This asserted that if Q was a property which held for oracles with probability one, then Q in fact held without oracles. This conjecture if true, clearly implied $\mathrm{P} \neq \mathrm{NP}$. However, using "double oracles", Kurtz [**1983**] showed that the random oracle conjecture fails. We now know of many 'natural' results that refute this conjecture. Nevertheless, it is felt that there is something to be salvaged from the conjecture.

The Baker-Gill-Solovay result leads to a rather general approach. Given some separation result we are attempting, if we find we cannot solve it, then often we try to find contradictory relativizations. If nothing else, this is a fair indication that the question is very hard indeed. For instance, we know that for P, PSPACE, NP, co-NP, LOGSPACE, EXPTIME any consistent combination is possible (we must keep $\mathrm{LOGSPACE}^A \neq \mathrm{PSPACE}^A$ as the argument relativises). Thus for instance, we can have $\mathrm{P}^A \neq \text{co-NP}^A \cap \mathrm{NP}^A$, or $\mathrm{P}^B \neq \mathrm{NP}^B$ yet $\text{co-NP}^B \cap \mathrm{NP}^B = \mathrm{P}^B$, or $\mathrm{P}^C \neq \mathrm{NP}^C$ yet $\mathrm{EXPTIME}^C = \mathrm{NEXTPTIME}^C$. (We cannot have $\mathrm{P}^D = \mathrm{NP}^D$ unless $\mathrm{EXPTIME}^D = \mathrm{NEXTPTIME}^D$ by upward translation).

One of the most difficult oracle separations is due to Yao [**1985**] who showed that there was an oracle that separated the whole polynomial time hierarchy. This impressive result used the very clever idea first proposed by Furst, Saxė and Sipser [**1984**], of replacing combinatorial counting arguments by probabilistic ones, and used results from "small depth circuits". Recently, I am told, it has been announced that there is a probability one separation.

We mention that *not all results relativise*. For instance, Cook's theorem that $\mathrm{NL} \subseteq \mathrm{P}$ and Shamir's result that $\mathrm{IP} = \mathrm{PSPACE}$ (which we look at later) *do not always hold true in relativised worlds*. It is important for us to understand why this is the case, if we are to understand what the Baker-Gill-Solovay results mean in terms of approaches to $\mathrm{P} = \mathrm{NP}$.

We now turn to some related material on probabilistic methods. Whilst we will not be able to do justice to this material, it is hoped that we can give the reader some flavour of it.

One of the key issues related to NP is whether it "really" matters. For instance suppose NP $\neq$ P yet hard instances occurred only exponentially often. Then for all practical purposes, P = NP. One is reminded here of the simplex method for linear programming. As we all know, in worst case this is $\Omega(2^{|x|})$ yet in real life it almost always takes nearly linear time. Smale and others eventually gave an explanation that the "average time" behavior of this algorithm really is polynomial.

There are many issues here. Do we want algorithms that are *always fast* and give good approximate solutions? We can formalize this interpretation in several ways, but for many sharp interpretations, we can show *that the existence of such a good approximation algorithm itself implies* P = NP. For a good discussion of the issues here, see Garey and Johnson [**1979**, §6].

If, on the other hand, one asked for the good behavior "on average" then the answer will depend on the probability distribution. For instance, if all graphs with n vertices are equally likely, then a backtracking algorithm solves 3-COLOURABILITY in 197 steps on average (i.e. in *constant* time) (Wilf [**1985**]). Similarly if the event '(x, y) is an edge' is random and of constant probability, then the expected computation time for Hamilton circuit is *linear* on average (Gurevich-Shelah [**1987**]). Even for CNFSAT, it has been shown that for certain probability distributions, the Davis-Putnam procedure (see Davis, Lagemann, and Loveland [**1962**]) is polynomial time on average (Goldberg, Purdam, and Brown [**1982**])

Another well-known form of approximation algorithm is the so-called Monte-Carlo methods whose origins go back to Ulam and others. A classic instance of this is for PRIME NUMBER. Here the algorithm of Solovay-Strassen [**1977**] does the following. Given a number n using a random number as a "seed", it will use a test to see if n is prime. Now if the test says 'n is composite', *n really is* composite, if the test says 'n is prime' it will be composite with probability at most $\frac{1}{2}$. Now if we use really random seeds, then each trial is independent. Also the trial runs (always) in linear time. So if we trial n say 2000 times and we are always told 'n is prime' then the probability that n is composite is $< \left(\frac{1}{2}\right)^{2000}$. Incidentally, this all revolves around being able to generate random numbers, or as we do in practice pseudo-random numbers. Again this is related to NP as follows. We call a function *one way* if f is polynomial time and honest, yet f^{-1} is not polynomial time. Note that if P = NP there are no one way functions. Homer and Long [**1981**] have proven that P = NP *iff there is a function no restriction of which is one way*. What is really interesting is that Impagliazzo, Levin and Luby [**1989**] have proven that if one way functions exist (as we believe) then there exist perfect psedo-random number generations (again this is related to cryptography). It should be remarked that Levin [**1984**] has developed an analogue of NP-completeness for random problems.

For a leisurely account of these and related matters, the reader should look at Johnson [**1984**] or Gurevich [**1989**].

We finish this section with a brief discussion of three related classes #P, IP and PP. PP is the class of probabilistically polynomial time accepters. To make this precise, we can say a NTM M *probabilistically accepts* x if more than half of the computation paths accept x. With this we define the class PP as the collection of sets B for which there is a p-time NTM M which probabilistically accepts B. The class PP is closely related to Valiant's counting class #P which is the class of functions which compute the number of accepting paths of a NTM M. It is easy to see that, for instance, PRIME NUMBER is in #P. Let #SAT be the set $\{\langle k, F\rangle : F \text{ is a boolean formula with} \geq k \text{ satisfying assignments}\}$. Then Gill [**1977**] showed that #SAT is $\leq_m^P$-complete for #P. It has been shown that computation of the Jones polynomial for certain classes of knots is #P-hard (that is, it is at least as hard as a #P-complete set). We have the following results:

Theorem 5.9. (Gill [**1972**]). NP $\subseteq$ PP $\subseteq$ PSPACE.

Proof. The inclusion PP $\subseteq$ PSPACE is by the direct simulation. Let $A \in$ NP. Let M be a NTM accepting A. We need a machine M$'$ that accepts on input x on more than $\frac{1}{2}$ of the paths iff M accepts x on at least one path. Let M$'$ be the machine that runs the following algorithm. At each step it chooses nondeterministically to either accept x or to run M on x.

Now we extend the "do nothing but accept" computation paths to have length that of those of M. Now if $x \notin A$, then exactly half of the paths accept x, and if $x \in A$ then as some computation path of M accepts x, more than half will accept x, so $A \in$ PP. ∎

The class PP has interesting closure properties. For instance, by changing accept and reject we see that PP is closed under complementation, i.e. PP = co-PP. It is also true (Gill [**1977**]) that PP is closed under $\leq_m^P$. Using intricate arguments about rational functions and simulation, the closure properties (5.10) and (5.11)were proven.

Theorem 5.10. (Biegel, Reingold and Spielman [**1990**]). *PP is closed under intersection.*

The argument of (5.10) was subsequently extended. We say $A \leq_{tt}^P B$ if there is a number k such that at most k questions are asked during the computation of A from B. This is called a polynomial truth table reduction.

Theorem 5.11. (Fortnow and Reingold [ta FOCS, Structures [**1991**]]). PP *is closed under truth table reductions.*

Ultimately one would like to prove that PP is closed under $\leq_T^P$, that is $\mathrm{P}^{\mathrm{PP}} =$ PP. Again there are oracles that give conflicting answers in relativised worlds here, so the problem seems very hard. It should be remarked that our intuition here is not that good since recent results shown that P^{PP} seems quite large. Let PH denote the polynomial time hierarchy.

Theorem 5.12. (Toda [**1989**]). $\mathrm{PP}^{\mathrm{PH}} \subseteq \mathrm{P}^{\mathrm{PP}}$.

There are a number of refinements of PP depending on the amount of errors one allows. The class PP is usually identified with the Monte Carlo methods, since errors are allowed. Another class of algorithms where no error is allowed are known as "Las Vegas" algorithms. This is often identified with the zero error probabilistic p-time machines (ZPP) defined below.

Now we allow 3 states, accept, reject and "I don't know". For such machines, the error probability is defined as the probability of halting on reject when we should accept plus that of halting on accept when we should reject. Let ZPP be the class accepted by such machines where the probability of an error is zero. Then ZPP is closed under complementation, union and intersection, and Gill [**1977**] showed.

Theorem 5.13. P $\subseteq$ ZPP $\subseteq$ P $\cap$ CO-NP.

Adleman and Huang have shown that PRIME NUMBER is in ZPP. It is believed that ZPP = P is quite possible.

Finally, we will turn to the class IP of those sets with "Interactive proofs". The first thing to realize is that the name is a slight misnomer. They really ought to be called "Interactive arguments that convince with high probability". The idea originated in Goldwasser, Micali and Rackoff [**1985**]. They conceived an interactive system as two communicating Turing machines called a *prover* and a *verifier*. The prover has no resource bounds except that there is a polynomial bound on the length of a string it sends to the verifier. The verifier must act in a fixed polynomial time bound. This is a way of viewing NP. The prover can convince the verifier by communicating an accepting configuration to the verifier who then verifies it. The idea is generalised as follows. Now we only require that for a set A, if $x \in A$ the prover convince the verifier with probability at least $\frac{2}{3}$ to accept x, and with probability only $\frac{1}{3}$ can the prover persuade the verifier to accept x if $x \in A$. (Of course the $\frac{1}{3}$, $\frac{2}{3}$ are only arbitrary here, and by amplification techniques can be chosen arbitrarily). We then let IP be the collection of sets accepted by an interactive protocol described as above where there is a polynomial number of rounds (communications between the prover and the verifier).

Another related class is IP$[k]$. This is the subclass of IP where at most k communications occur between the prover and the verifier. By the work of Goldwasser and Sipser [**1986**] another characterisation of IP$[k]$ is *a* set A is in IP$[k]$ if there is a set B in NP, and a polynomial p such that for all x, Prob$[\langle x, y\rangle \in B$ iff $x \in A] \geq \frac{3}{4}$ where y is chosen at random from the strings of length $p(|x|)$, (and in particular IP$[2]$ = IP$[k]$ for all k).

It is not difficult to see that ZPP $\subseteq$ IP[2] and it was shown that a slightly larger class containing NP (called BPP) was contained in IP[2]. Goldreich, Micali and Wigderson [**1986**] showed that IP[2] contained language believed not to be in NP, namely NON-ISOMORPHIC GRAPHS: The protocol is as follows: Let G_1 and G_2 be given graphs. The verifier randomly picks a graph Q isomorphic to one of G_1 or G_2 say G_i obtained by a random permutation of the G_i's vertices, and sends this to the prover. The prover then checks to see if one of G_1 or G_2 is isomorphic to H, and if so, communicates this G_j to the verifier. Repeat the procedure twice, and the verifier accepts if $j = i$ on both rounds. Note that if $G_1 \not\cong G_2$ then $j = i$ on both rounds hence the verifier accepts with probability 1. Otherwise the verifier accepts with probability at most $\frac{1}{4}$. We remark that in Goldreich, Mansour, and Sipser [**1987**] it is proven that as above it is always possible to replace the $\frac{2}{3}$ acceptance probability, by 'perfect acceptance' probability : that is, if $x \in A$ then the verifier accepts with probability one.

The above was the state of play until the breakthrough paper of Lund, Fortnow, Karloff and Ganesi [**1990**] who built on earlier work of Toda, Valiant and others to show surprisingly that PH $\subseteq$ IP. It was then not long that the 'algebraic' methods introduced by these authors were extended and simplified by Shamir who proved the following remarkable result:

Theorem 5.14. (Shamir [**1990**]). IP = PSPACE.

Proof. (following Shamir [**1990**]). We build on interactive protocol for a variant of QBFSAT, which is PSPACE complete as we saw in §4. A closed QBF formula is called *simple* in the given representation if every occurrence of each variable is separated from its point of qualification by at most one *universal* quantifier.

Examples.

(i) $\forall x_1 \forall x_2 \exists x_1 \big((x_1 \wedge x_2) \wedge \forall x_4(x_2 \wedge x_3 \wedge x_4)\big)$ is simple,

(ii) $\forall x_1 \forall x_2 \big((x_1 \vee x_2) \wedge \forall x_3(\overline{x}_1 \wedge x_3)\big)$ is not simple.

By an easy indication, Shamir observed that all QBF formulae can be transformed into equivalent simple QBF formulae with only a polynomial increase in size. Now we come to the decisive idea of Lund et al [**1990**].

Definition 5.15. Given a QBF formulae ϕ, the arithmetization $A(\phi)$ of ϕ is defined via

(i) Replace each variable x_i by one z_i that ranges over Z.

(ii) Replace $\overline{x}_i$ by $1 - z_i$.

(iii) Replace $\wedge$ by multiplication; replace $\vee$ by $+$, replace $\forall x_i$ by the integer product $\prod_{z_i \in \{0,1\}}$ and $\exists x_i$ by $\sum_{z_i \in \{0,1\}}$.

For instance, $\phi = \forall x_1 \exists x_2 \big(x_1 \wedge x_2 \vee \exists x_3(\overline{x}_2 \wedge x_3)\big)$ becomes

$$A(\phi) = \prod_{z_1 \in \{0,1\}} \sum_{z_2 \in \{0,1\}} \big(z_1 z_2 + \sum_{z_3 \in \{0,1\}} (1 - z_2) z_3\big).$$

It is easy to show that ϕ *is true iff the value of* $A(\phi) = 0$, *and that if size of* ϕ *is* n *then the value of* $A(\phi)$ *cannot exceed* $0(2^{2^n})$. This last result would seem a restriction on p-time proofs, but Shamir found a device to get around.this.

Lemma 5.16. *Let* ϕ *be a* QBF *of size* n. *Then there exists a prime* p *of length polynomial in* n *such that* $A(\phi) \not\equiv 0 (\text{mod } p)$ *iff* ϕ *is true.*

Proof. Suppose $A(\phi) = \varnothing$. If $A(\phi) \equiv 0(\text{mod } p)$ for all polynomial size primes, then by the Chinese remainder theorem, it is $\equiv 0$ modulo their product. By the prime number theorem, this product is $0(2^{2^{n^d}})$, for any desired constant d; this contradicts the assumption $A(\phi) \neq 0$ and $A(\phi)$ is $0(2^{2^n})$. If ϕ is false then $A(\phi) \equiv 0$ for any p. ∎

Now given an arithmetical form $A(\phi)$ we can define the *functional form* $F(\phi)$, as the result of deleting the outermost $\sum$ or $\prod$ and considering the result as a polynomial in one variable. Also, define the randomized form $A(\phi)(z_i = r)$ to be the result of setting z_i to be the random number r (mod p) supplied by the verifier. It is not hard to see that the polynomial $q(z_i)$ representing $F(\phi)$ can have exponentially high degree, yet Shamir observed that if ϕ is simple, $F(\phi)$ *grows at most linearly in the size of* ϕ (this is the key reason for simple forms). Now we can finish with the interactive protocol to prove $A \not\equiv 0$ (mod p):

The prover sends the claimed value a of $A = A(\phi)$ (mod p) to the verifier, and justifies this claim by considering successively smaller subexpressions of A. Thus at any stage, the current expression for A is given as $A_1 + A_2$ or A_1A_2, where A_1 is a polynomial with fully instantiated variables, (*whose value a_1 can be computed by the verifier; this is crucial*) and A_2 starts with the leftmost $\sum$ or $\prod$ of $A(\phi)$. The prover and verifier then repeat the following:

1. If $A_2 = \varnothing$, the verifier stops and accepts the claim iff $a = a_1$.

2. If $A_1 \neq \varnothing$, the verifier replaces A by A_2 and replaces a by $a - a_1$ (mod p) or $\frac{a}{a_1}$ (mod p) (depending on the operator between A_1 and A_2). If the verifier tries to divide by $a_1 \equiv 0$ (mod p) he stops and accepts if $a \equiv 0$ (mod p).

3. Otherwise, the prover sends the polynomial $q(z_i)$ describing $F(A)$ to the verifier. The verifier checks that $a \equiv q(0) + q(1)$ (mod p) (resp. $q \equiv q(0) \cdot q(1)$) (mod p) and sends random $r \in \mathbb{Z}_p$ to the prover. He then replaces A by $A(z_i = r)$ (mod p) and a by $q(r)$ (mod p).

Example.

We write $\prod_{z_1}$ for $\prod_{z_1 \in \{0,1\}}$ etc. Let $\phi = (\forall x)(\overline{x}_1 \vee \exists x_2 \forall x_3 (x_1 \wedge x_2 \vee x_3))$ $A(\phi) = \Pi_{z_1}(1 - z_1 + \sum_{z_2} \Pi_{z_3}(z_1 z_2 + z_3))$. Now $F(\phi) = q(z_1) = z_1^2 + 1$ and $A(\phi)(z_1 = 3) = ((1-3) + \sum_{z_2} \Pi_{z_3}(3z_2 + z_3)) = 10$.

Now in the protocol above, suppose P(rover) claims the value is 2. Then P sends 2 and its polynomial $q(z_1) = z_1^2 + 1$ to V(erifier). V checks that $q(0) \cdot q(1) = 2$ (here, of course $A_1 \equiv 1$) sends $z_1 = 3$ to P, replaces A by $F(\phi)(z_1 = 3)$ and replaces $a(=2)$ by $q(3) = 10$.

The new A starts with nonempty A_1 whose value $a_1 = (1 - 3) = -2$ can be computed by V (the new A is of course of the $+$ form). We now adjust A and a.

$$A = \sum_{z_2} \prod_{z_3} (3z_2 + z_3)$$
$$a = 10 - (-2) = 12$$

$F(A) = \prod(3z_2 + z_3)$ so $q(z_2) = 9z_2^2 + 3z_2$. This is sent to V who checks that $q(0) + q(1) = 0 + 12 = 12 = a$, and picks a random $z_2 = 2$. A and a are adjusted accordingly:

$$A = \prod(6 + z_3)$$
$$a = q(2) = 42.$$

P now sends $q(z_3) = z_3 + 6$ to V and V checks that $q(0) + q(1) = 42 = a$. Finally, V chooses $z_3 = 5$ and V verifies that

$$A(z_3 = 5) = 11 = a = q(5).$$

Shamir's result follows from

Lemma 5.17.

(i) *If ϕ is true and P is honest, V always accepts the proof.*

(ii) *If ϕ is false, V accepts the proof with negligible probability.*

Proof Sketch. To see that (ii) holds, we note that a cheating P who supplies a bad a must provide a false $q(z_1)$ to support his claim. By the interpolation theorem, such an incorrect polynomial of degree t can agree with a correct one on at most t points of $\mathbb{Z}_p$. In fact this means that it is likely that we will pick up a dishonest P *on the first round.*Thus when the value of p is exponential in the size of ϕ and the value of t is polynomial, there is only a negligible probability that an *incorrect* q yields a *correct* value when computed at a *random* r. As a result with overwhelming probability a false P is forced to provide false values for smaller and smaller A until V discovers this fact, and this concludes the proof of Shamir's theorem. ∎

6. Separation Results and Other Topics

In this last section, we will look at some separation results we *can* prove, as well as some other interesting results on circuit complexity and nonuniform algorithms. First we look at separation results. Here, results are weak in the sense that they depend on a specific model. Of course, to prove P $\neq$ NP on a specific model would suffice, but *other* classes such as linear time are not robust under change of models. With the Turing machine model, using an intricate combinatorial argument, which built on work of Paul and Reichuk [**1981**], Paul et al showed

Theorem 6.1. (Paul, Pippenger, Szemeredi, and Trotter [**1983**]). *DLIN $\neq$ NLIN. Here DLIN $= \cup_{c\geq 0}$DTIME (c.n) and NLIN $= \cup_{c\geq 0}$NTIME (c.n).*

It is perhaps noteworthy that at least two of the authors (Trotter and Szemeredi) are classical combinatorialists, and Szemeredi is surely one of the best combinatorialists in the world. To me, this is a comment on the 'Ramsey theoretical' style of the arguments. As another perhaps slightly easier example for Turing machines is the old question of Hartmanis and Stearns [**1965**] of whether the bound $O(n^2)$ for simulating a two head Turing machine was optimal. As we mentioned earlier this was solved for online deterministic machines by Li and by Maass, the latter also proving that 2 heads are better than one for nondeterministic machines. (Here, as we mentioned earlier, it is unknown if $O(n^2)$ is optimal). These results are quite instructive to study in detail as they employ intricate combinatorial techniques in combination with two ubiquitous ideas: *crossing sequences* and *Kolmogorov complexity*, so we sketch the ideas used in the proofs. Thus remember we have the setting of a one way input tape with a work tape. We follow the presentation Ming Li [**1985**].

Definition 6.2.

(i) Let A be a block of input tape and R a region of the work tape. We say M maps A *into* R if, whilst reading A, the work head h_2 does not leave R. If h_2 traverses all of R we say M maps A *onto* R.

(ii) A crossing sequence (c.s.) is a collection of instantaneous descriptions (ID) of the form $\langle$state, h_1's position$\rangle$. Let |c.s.| be the amount of space needed to represent a c.s.

As our machines only go right on h_1, we can give the ith ID, ID_i of a c.s. for M can be represented as $\mathrm{ID}_i = \langle$state, h_1's current position − h_1's position of $\mathrm{ID}_{i-1}\rangle$, with $\mathrm{ID}_0 = \langle \dots, 0\rangle$

Lemma 6.2. (Li [**1985**], implicit in Maass [**1984, 1985**]).

(a) *If a c.s. has d ID's and the length of the input is n, then using the above technique, $|c.s| \le d(|M|) + \sum_{i=1}^{d} \log k_i d$ with $\sum_{i=1}^{d} k_i = n$, where k_i is the number of squares between h_1's position in $ID_i - ID_{i-1}$. Consequently, as log is convex, $|c.s| \le d(|M|) + d\log\left(\frac{n}{d}\right)$.*

(b) *Let $A_1, \dots, A_k$ be blocks of input tape of equal length c and delete d of them, then to represent the remainder (and still remember the relevant positions), we can use $m\overline{A}_1 \dots \overline{A}_k(p_1, d_1)(p_2, d_2)\dots(p_b, d_b)$ where m denotes the number of nonempty A_i's, and*

$$\overline{A} = A_i \text{ if } A_i \text{ is not deleted and } = \varnothing,$$

otherwise, and (p_i, d_i) means the next p_i A_j's are in one group followed by d_i deleted A_i's (some technical conditions are needed here to sort out the (p_i, d_i). For instance locally double and put 01 *at beginning and end* (01011 $\rightarrow$ 01001100111101)) *(so $A_1A_2\cancel{A}_3A_4\cancel{A}_5\cancel{A}_6 \rightarrow 3A_1A_2A_4(2,1)(1,2)$). Then the space needed is*

$$\begin{aligned} &\le \sum_{i=1}^{k} |A_i| + d\log\left(\frac{n}{d}\right) \\ &= mc + d\log\left(\frac{n}{d}\right). \end{aligned}$$

The proof of the above is a straightforward analysis of the requirements. This lemma allows a critical observation that says roughly, if the machine maps too much of the input tape into a small region, then there is a small machine that accepts the input tape. This is very important since we see that it implies the input is not *random*. The instructive idea is summarized as

Lemma 6.3. (Jamming Lemma, Li [**1985**], implicit also in Maass [**1985**]).

Suppose on input $A_1 \dots A_k\#$ with the A_i's of equal length c, M maps $A_{i_1}, \dots, A_{i_t}$ into region R by the time h_1 reaches $\#$. Then the contents of the work tape can be reconstructed from $\{A_1, \dots, A_k\} - \{A_{i_1}, \dots, A_{i_t}\}$, R and the two crossing sequences on the left(l) and right(r) boundaries of R.

Proof. Put $\{A_1, \ldots, A_k\} - \{A_{i_1}, \ldots, A_{i_t}\}$ in their correct positions, and write R on the work tape. As indicated by the c.s.'s we can run M with h_2 staying left of R. If h_2 reaches l interrupt the programme and move to the next ID of the c.s. Do this on the left and right independently. ∎

We remark that, for obvious reasons, the above is called a *cut and paste* argument; and furthermore as we shall see, it implies that if

$$\sum_{i=1}^{t} |A_{i_j}| \geq 2(|R|) + 2|\text{c.s.}| + |M|),$$

then $A_1 \ldots A_k$ is not random. The critical notion that makes this precise is the following.

Definition 6.4. (Chiatin [**1966,1969**], Kolmogorov [**1965, 1968**]).

(i) The *Kolmogorov complexity* $K(y)$ of a string y is the size of the smallest T.M. which accepts *only* y.

(ii) y is called *random* if $K(y) \geq |y| - 1$.

There are many formulations of the notion above, but ultimately any notion of randomness depends on computability theory. Without going into details, this notion has a lot of applications and philosophical applications. The reader should see Chiatin [**1974**] and Li and Vitanyi [**1988**], the latter for a good survey of applications of Kolmogorov complexity to complexity theory. For our purposes, the use will be that it will force h_2 to behave 'honestly' on a random input. Namely, we force h_2 to have *long traverses* (and hence take a long time) on a random input, by arguing that if it does not then the input is not random.

Define a language

$$L = \{A_1\$A_2\$\ldots\$A_k\$Y_1\$\ldots\$Y_t\#(1^{i_1}, 1^{j_s}), \ldots (1^{i_s}1^{j_s}) :$$
$$A_p = Y_q \text{ for } p = i_1 + \ldots + i_u, q = j_1 + \ldots + j_u \text{ and } u = 1, \ldots, s\}$$

Example 6.5.

$$M\$M\$N\$M\$N\$N\#M\$M\$N\#(1^3, 1^3)(1^2, 1^1)(1^3, 1^2) \in L.$$

It is easy to see that a 2-tape online machine can accept L in time $O(n)$.

Theorem 6.6. (Li [**1985**], Maass [**1984, 1985**]). *For an online one tape machine M to accept L requires $\Omega(n^2)$ time. Hence the Hartmanis and Stearns [*__1965__*] 2-tape 1-tape trade-off is optimal.*

Proof Sketch. (After Li [**1985**]). Suppose each square of M's work tape is coded by c_0 bits. Fix an n and c so that the inequalities below hold and

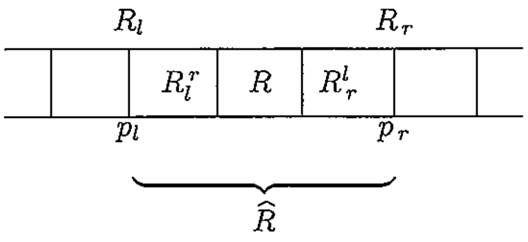
$$c > 100c_0|M|; \frac{2n\log c}{c} < \frac{n}{48}$$
$$\frac{(\log c)}{c^2} < \frac{n}{8c}.$$

Now take A with $|A| = n$ and $K(A) > n$. Break A into $k = \frac{n}{c}$ pieces each c long, and call them $A_1, \ldots, A_k$. Now consider M's response to $A_1\$A_2\$\ldots\$A_k\#$.

If more then $\frac{k}{2}$ of the A_i's are mapped onto regions size $\geq \frac{n}{c^3}$ then M requires $\Omega(n^2)$ time, since the time is then $\geq \frac{k}{2}\left(\frac{n}{c^3}\right) = \frac{n^2}{2c^4}$.

We can conclude that at least half of the A_i's are mapped into regions of size $< \frac{n}{c^3}$ on the work tape. Now order these A_i's by left boundary of the regions into which they map. Let A_m be the median. There are two cases.

Case 1. (Jammed). There are $> \frac{k}{c}$ A_i's and a fixed R of length $\frac{n}{c^2}$ on the work tape with M mapping these A_i's into R. We claim that this cannot happen as then there is a short programme accepting A contradicting the fact that $K(A) \geq |x|$. Consider the 2 regions R_l and R_r of length $|R|$ on the left and right of R. Let p_z be the position of R_z with the shortest c.s. Both of these c.s. must have length $< \frac{n}{c^3}$ lest M uses $\geq \left(\frac{n}{c^2}\right)\left(\frac{n}{c^3}\right)$ time. Now cut and paste. Let $R_r^l(R_l^r)$ be the positions of $R_r(R_l)$ to the left (right) of $p_r(p_l)$ and $\widehat{R}$ the regions from p_l to p_r.



We use the information

(i) There are $< k - \frac{k}{c}$ A_i's not mapped into $\widehat{R}$

(ii) the 2 c.s. at p_l and p_r

(iii) $\widehat{R}$

To see if $X = A$ first see if $|X| = |A|$ via the jamming lemma we get the contents of M's work tape by the time we reach $\#$. Divide X into $X_1\$\ldots\$X_k$ and run M with input $\#X_1\$\ldots\$X_k\#(1,1)\ldots(1,1)$ attached. Then M accepts iff $X = A$.

The crucial point is that the programme is short; namely by the lemmas (6.2) we need

(i) $\leq n - \left(\frac{n}{c}\right) + \left(\frac{n}{48c}\right) \leq n - \frac{n}{c}$ for the $k - \frac{k}{c}$

(ii) $\leq \frac{3n}{c^2}$ for $\widehat{R}$

(iii) $\left(\frac{n}{c^3}\right)(|M| + 3\log c)$ for the 2 c.s.

(iv) $O(\log n)$ for counters

which is $< n$ as n dominates the above.

Case 2. (Unjammed). For each region R with $|R| = \frac{n}{c^2}$ there are $\leq \frac{k}{c}$ A_i's mapped into R.

Fix a region R_m of size $\frac{n}{c^2}$ that A_m maps into, with R_m the median region. Then there exist $\geq \frac{k}{6}$ A_i's say, $S_r = \{A_{i_1}, \ldots, A_{i_{\frac{k}{6}}}\}$ all mapped right of R_m and similarly $S_l = \{A_{j_1}, \ldots, A_{j_{\frac{k}{6}}}\}$ all mapped left of R_m. Order as $i_1 < \ldots < i_{\frac{k}{6}}$ and $j_1 < \ldots < j_{\frac{k}{6}}$. Now let Y be generated as $Y_1 = A_{i_1}$, $Y_2 = A_{j_1}$, $Y_3 = A_{i_2} \ldots$ so that the partial input is read as

$$A_1\$\ldots\$A_k\#Y_1\$\ldots\$Y_{\frac{k}{3}}\#$$

The idea is that we argue that we must cross R_m *a lot* and hence take *a lot of time*.

If there are not at least $\frac{k}{12}$ pairs $Y_{2i-1}\$Y_{2i}\$$ such that h_2 traverses $> \frac{n}{4c^2}$, then we use $> n^2$ time. If there are $\frac{k}{12}$ pairs such that h_2 traverses $< \frac{n}{4c^2}$ for each then there is a region $R \subseteq R_m$ and $\frac{k}{24}A_i$'s from either S_r or S_l mapped onto one side of R and *the corresponding Y_i's mapped onto the other side of R.* Call these sets S_A and S_Y with indices $a_1 < \ldots < a_{\frac{k}{24}}$, $b_1 < \ldots < b_{\frac{k}{24}}$. Now append $\#(1^{a_1}, 1^{b_1})(1^{a_1+a_2}, 1^{b_1+b_2})\ldots$.

Now one finishes as follows: if the shortest c.s. of R is big then the run time is $\Omega(n^2)$. If the shortest c.s. of R is *small* we can, as before, use the jamming lemma to accept A. ∎

The situation for off line T.M's is not so clear. The only concrete results I am aware of is that of Dietzfelbinger, Maass and Schnifger [**ta**] who proved an optimal lower bound of $\Omega(\text{n.q.} \quad \lceil |\log(q)|p\rceil^{\frac{1}{2}})$ for the problem of transposing an $q \times q$-matrix with elements of bit length p and input size n. This again uses Kolmogorov complexity as well as clever combinatorial techniques.

There have been a number of other separation type result for other models such as RAMS and decision trees/circuits. Many of these use delicate Ramsey-theoretical arguments, or similar ideas. Here the reader should recall that Ramsey's theorem is a powerful generalization of the pigeonhole principle. Let $[n]^k$ denote the set of k-subsets of $\{1, \ldots, n\}$. For a colouring χ of $[n]^k$ a *homogeneous* set for χ if H is a subset of $\{1, \ldots, n\}$ such that $[H]^k$ is monochromatic. Ramsey's theorem, which we call H, asserts.

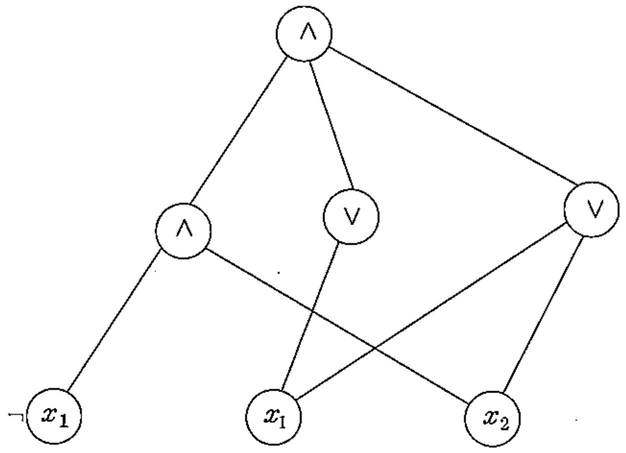
Theorem 6.7. (Ramsey [**1930**]). *For all k, r, p there is an n so large that for any r-colouring of $[n]^k$ there is a homogeneous set of size at least p. This is written as*

$$n \rightarrow (p)^k_r.$$

As a point of explanation, let $k = r = 2$, so are 2-colour pairs. Colour a pair red if they are friends, and blue if nonfriends. Then Ramsey's Theorem asserts that for all p there is an n so large that in any party of size n, there is a group of p mutual friends, or p mutual nonfriends. Ramsey's Theorem is but a single example of a large class of similar theorems asserting good behaved bits in seemingly chaotic large numbers (see Graham, Rothschild and Spencer [**1980**], or (Downey [**1989**] for a friendly survey aimed at college students)).

It is easy to see why Ramsey type theorems can be used in complexity theoretical arguments. We have some process P we feel ought to take a long time. We have another process Q *we know* takes a (fairly) long time. We use a Ramsey type argument to show that to perform P we need to perform Q *often*. This type of argument has been especially useful in RAMS, branching programmes and decision tree complexity. The reader is referred to, for instance, Maass [**1988**], Pudlak [**1984**], Alon and Maass [**1986**], and Ajtal, Babai, Hajnal, Komlos, Pudlak, Rodl, Szemeredi and Turan [**1986**].

One last area where some fine results have been obtained is that of *circuit complexity*. A boolean circuit is a directed cyclic graph with internal nodes labelled $\wedge$; (and), $\vee$ (and), $\neg$ (not). It has one sink called the *output node* and a number of input nodes labelled with variables x_i or $\neg x_i$.



It is not hard to see that by de-Morgan's law's we need only consider $\wedge$ or $\vee$ as the only internal nodes, and by stretching we need only alternating rows of $\wedge$ and then $\vee$ gates. For a given input assignment of the variables as 0 or 1 we can evaluate the circuit in the obvious way. The *depth* of the circuit is the length of the longest path and the size of the circuit is the number of gates. We say a collection of circuits $\{C_i : i \in A\}$ accepts a set A if for each input x of length $n, x \in A$ iff C_n accepts x (i.e. C_n on input x output 1). It is easy to show that if A is accepted by a T.M. M in time $0(f(n))$ then we can uniformly construct a polynomial time collection of circuits that accepts A in time $0(f(n)^2)$. (Indeed this can be improved to $0(f(n)logf(n))$).

Clearly if we could prove lower bounds on the size of circuits, we could prove lower bounds on the running times of T.M's. Alas, the best known lower bound on the size of circuits for a NP-problem is $3n$ (N. Blum [1984]) so the situation is not all that good, to say the least. (I should point out that if we do not demand uniformity of the circuits, one gets a new class. This has been widely studied and certainly includes sets not in P. We do not mention these results here due to lack of space).

Good results have been obtained by restricting the circuit models. The best progress has been obtained for *monotone* and *small depth* circuits. A monotone circuit has no negated inputs nor $\neg$ gates. A key idea was supplied by Furst, Saxe, and Sipser [**1984**]:

Definition 6.8. A *restriction* q is a mapping of the variables to $\{0,1,*\}$ with $q(x_i) = j$ (0 or 1) means we replace x_i by the value j. If $q(x_i) = *$ then do nothing to x_i.

Now suppose we wish to argue that there is no small small depth circuit that can compute some set A. We would like to argue by induction on the depth d. For $d = 2$ we usually need an absolute argument, but this is not too hard. At the induction level suppose we go from d to $d+1$. There is an obvious strategy. Suppose the output circuit is an $\wedge$ gate. So the level below is a set of $\vee$ gates. If we could replace this portion of the circuit with an equivalent one where the second layer is all $\wedge$ gates, we could collapse the top and second top into one layer to get a depth d circuit. We can do this via de-Morgan's laws. Unfortunately, the trade-off in size by this approach is exponential. That is a *small* circuit of depth $d+1$ becomes a *large* circuit of size d. The key idea of Furst et al [**1984**] was to observe that if, for instance, one input of an or gate has value 1 the gate has value 1, or if one input of an and gate has value 0 the gate has value 0. Then the idea is that using restrictions we should be able to perform the process and still get a small circuit. The decisive idea is to use random restrictions, and then, with high probability, nothing is lost. A classic example of this approach was an improvement of Hastad who showed that

Theorem 6.9. (Hastad [**1986**]). *The pairing function, $f(x) = \sum x_i \pmod 2$, of n inputs and depth k requires size $\geq 2^{c_k n^{\frac{1}{k-1}}}$ for $n > n_k$ where $c_k = 10^{-\frac{k}{k-1}}$.*

Also Yao [**1985**] and Hastad [**1986**] built on earlier work of Sipser [**1983**] to examine the power of depth k over depth $k-1$, and obtain suitable separation results. This allowed Yao to construct an oracle separating PH (since $\sum_k^P$ is closely related to depth $k+1$ circuits).

Finally another approach to P =?NP was suggested. We should study monotone circuits. The idea was that if f could be computed by a small normal circuit then it could be also by a small(ish) monotone one, for some suitable class of functions $f \in$ NP. One then proves exponential lower bounds for the monotone circuit complexity of some such f. In a major paper, Rasborov [**1985**] developed another technique of *approximations* where he replaces large $\vee$'s ($\wedge$'s) by small $\vee$'s ($\wedge$'s) and gets a related circuit that computes the same value on most inputs. This technique allowed Rasborov to settle the above conjectured approach to P =?NP by showing exponential monotone complexity for several sets known to be in P, as well as some NP complete problems such as CLIQUE. His work has been extended by, for instance, Alon and Boppana [**1987**]. It is not clear if it is possible to use his ideas for more general circuits. It should be remarked that his technique improved to exponential previous bounds known only to be linear.

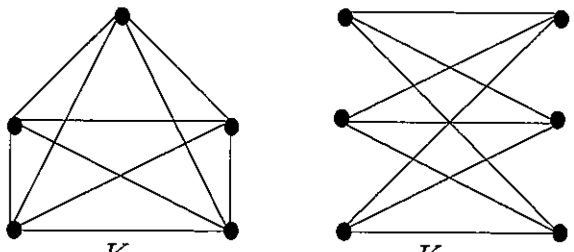
To finish this article, we will briefly look at some lovely work applying a major theorem from graph theory to complexity, and its philosophical implications.

The work we shall now study challenges the notion that tractable should be identified as P and comes as a result of the magnificent theorem of Robertson and Seymour ['Graph Minors I–XVI']. It has long been recognized that it is conceivable that P = NP yet it may not be possible to *exhibit* p-time algorithms for all members of P due to lack of uniformity. The study of P is particularly driven by *practical* computation where algorithm builders believe that if process Q has an algorithm in P and 'Q was a reasonably natural problem' then Q would have an algorithm that was *feasible* in the sense that the polynomial would have low degree and the constants would be small; and furthermore if we could prove Q was in P, then we could *find* the algorithm.

For the first time we have tools to challenge (but not yet refute) this long held view. We have tools that allow us to prove that various processes are in P *without ever exhibiting an algorithm for them and furthermore the technique can yield algorithms whose constants of proportionality exceed the number of atoms in the universe.* As in Fellows [**1989**], this can be likened to the situation with Hilbert's solution to Gordon's problem of invariants, where as we know Hilbert used a non-constructive proof to show any set of forms in a finite number of variables over $\mathbb{Q}(\alpha)$ has a finite basis, and Gordon's famous reaction that "This is not mathematics this is theology!"

Turning to some details, the Robertson-Seymour theorems concern finite (undirected) graphs, under the *immersion* and *minor* orderings. Let G be a graph and $\langle x,y\rangle, \langle y,z\rangle$ two edges of G, with y of degree 2. We say G' is a *contraction* of G (via y) if we delete these edges and replace them with a single edge $\langle x,z\rangle$. We say G' is a *minor* of $G(G' \leq_m G)$ if G' results from G by repeated application of taking subgraphs and then contracting edges. The notion of immersion ($\leq_i$) is similar but for our purposes it will suffice to concentrate on the minor ordering.

A famous theorem of Kuratowski says that G is planar (i.e. can be drawn on the plane with no edges crossing) iff G has neither K_5 nor $K_{3;3}$ (below) as a minor (actually it says something stronger, but this suffices)



It was an old conjecture that there was a Kuratowski theorem for surfaces. Namely if H is any surface of genus k there is a finite set of graphs $F_1, \ldots, F_m$ such that G is embeddable on H iff G does not have F_i as a minor for any i. A strengthening of this is another conjecture of Wagner: Define a set of graphs S to be *closed* under the minor (immersion) order if $G \in S$ and $G' \leq_m G$ implies $G' \in S$. Note that if G embeds on a surface H then so does any minor so that the collection of graphs embeddable on H is a closed set under $\leq_m$. If S is a closed set, under $\leq_m$, define an *obstruction set* O for S to be a set of graphs $\{F_1, \ldots, F_n\}$ such that $G \in S$ iff $F_j \not\leq_m G$. Wagner conjectured the following:

Theorem 6.10. (Robertson – Seymour). *Any set of graphs closed under the minor ordering has a finite obstruction set.*

Similarly, they solved a conjecture of Nash-Williams:

Theorem 6.11. (Robertson – Seymour). *Any set of graphs closed under the immersion ordering has a finite obstruction set.*

These theorems are the result of a very long series of papers (see the references) where a deep and important structure theory is developed. I should point out that we know that the finite sets are *intrinsically non-constructive* in various technical senses (see Friedman-Robertson-Seymour [**1985**]). In particular, the finite sets can be very large and grow faster than (e.g.) Ackermann's function.

What has this to do with complexity? The crucial point is this: the problem of determining if $H \leq_m G$ or $H \leq_i G$ is $O(n^3)$ so that we get the corollaries

Corollary 6.12. (Robertson-Seymour).

(i) *If C is any class closed under $\leq_m$ or $\leq_i$ then membership of C is $O(n^3)$.*

(ii) *Indeed, if C excludes a planar graph, then membership of C is $O(n^2)$.*

Nevertheless, the algorithms depend on (i) knowing the obstruction set and (ii) the constants inherited by the proof of the Robertson – Seymour Theorems. In particular the size of the obstruction set in (i), and the notion of *tree width* in (ii) give astronomically large constants. These algorithms, and (ii) also makes the technique intrinsically nonconstructive in general.

To apply the above results in 'real life' one takes one's favourite problem and shows that it can be modelled by a class of graphs closed under $\leq_m$ or $\leq_i$. Then immediately we get $O(n^3)$ acceptance. Here are some examples of where this methodology has been applied.

KNOTLESSNESS

Instance: A graph G

Question: Does G have a knotless embedding in 3-space.

Comment: It is not at all obvious that this is even *decidable* and there is no known technique for proving decidability except the 'RS-poset' method.

GATE MATRIX LAYOUT (k)

Instance: A boolean Matrix B and

Question: Is there a permutation of the columns of B so that,

if we replace each 0 lying between arrows leftmost and rightmost ones, no column contains more than k ones and *'s.

Comment: If k varies, the problem is NP-complete. The problem comes from problems in VLSI layout. This is $O(n^2)$ as it can be formulated so that we exclude a planar graph.

CROSSING NUMBER (k)

Instance: A graph G

Question: Can G be drawn on the plane with at most k edges

crossing?

***Comment*:** If k varies, the problem is NP-complete (Garey and Johnson [**1978**]).

We will not go into further details, but refer the reader to the slightly dated survey paper of Fellows [**1989**]. Currently there are many issues in this area. One is to try to make the algorithms 'practical'. For instance, one idea is to develop a *finer* theory for certain classes of minor closed sets which yield small constants. This seems conceivable for small tree width problems. A very promising approach is to look at approximate algorithms. For instance in Kuratowski's theorem $K_{3,3}$ is a very good approximation to the obstruction set. Here we run into the usual problems of using the appropriate distribution (that we saw earlier). Perhaps Renyi's theory of relative probabilities is appropriate here.

Finally, the approach above yields the following observation. Many classes of problems can be *parameterized* by a parameter k such that for each k the problem is fixed time tractable. There are, of course, lots of other examples of this: For example –

VERTEX COVER (k)

Instance: A graph G

Question: Does G have a vertex cover of size $\leq k$.

***Comment*:** This is solvable in time $O(n)$ and is NP-complete if k varies (Garey and Johnson [**1978**]).

LINEAR INEQUALITIES (k)

Instance: A system of k linear inequalities

Question: Can this be made consistent over $\mathbb{Q}$ by deleting k of them.

***Comment*:** For each k there is an $0(n^{\alpha(k)})$ algorithm, but no known fixed α works.

The observation is that there seems no relation between the complexity of the fixed parameter problems and the complexity as k varies. This leads one to study reducibilities between classes of fixed parameter families. For instance, we consider sets $Q \subseteq \mathbb{N} \times \sum^*$ so that $Q_k = \{\langle k, \sigma\rangle : \langle k, \sigma\rangle \in Q\}$ represents the k-parameter problem. For simplicity, assume the set Q is given in some uniform way. We can define a number of reducibilities between such families. For instance $Q \leq \widehat{Q}$ can mean:

There is a procedure Φ and a recursive f and a fixed n such that for each $k, \langle k, x\rangle \in Q$ iff $\Phi(\widehat{Q}^{(f(k))}, \langle k, x\rangle) = 1$ where $Q(f(k))$ denotes $\cup_{i\leq f(k)}\widehat{Q}_i$ and the computation must run in time $\leq f(k)|x|^n$.

The 0-degree for such classes is called the *fixed parameter tractable sets* (and are related to the PGT classes of Abrahamson, Ellis, Fellows, Mata [**1989**]) and are those, such as those arising from RS posets, for which there is a *fixed* α with an $O(n^\alpha)$ membership test for each slice Q_k (i.e. α is independent of k). Abrahamson, et al first proposed there should be a completeness theory for such objects. For instance, the set representing {Linear inequalities (k)} is complete in this setting (for the appropriate reduction). We generalized and developed a good completeness theory, and have shown (Downey, Fellows and Slaman [**ta**]) that, for instance, the degrees are dense, for certain reducibilities. Downey and Fellows had also refined the notions of completeness, and developed several completeness theories related to circuit complexity. Other work on this area can be found in Downey-Fellows [**ta**

1, 2, 3], Abrahamson, Downey Fellows [**ta**] and Fellows-Koblitz [**ta**].

Limitations of time and space (sic) do not allow me to present other very interesting topics such as inductive learning theory (e.g. Blum and Blum [**1975**], Gold [**1967**], Gasarch, Pleszkoch and Solovay [ta]). There one considers a T.M. as the model of a "learner" who must eventually recognise the index for a recursive function after a finite number of mistakes (consider language acquisition).

Another topic, I wish I could have included is looking at feasible analogues of classical objects. For instance, Cenzer, Buss, Nerode and Remmel have investigated 'polynomial time' logic and algebra. For instance, over GF(2) the problem of determining independence is a natural NP problem. One can then study polynomial time *presented* structures and look at the feasibility of other processes on such structures. Closely related is the theory of Ker-i-Ko and Friedman on the P-time content of analysis whereas, say, Pour-El and Richards [**1989**] look at effectiveness conditions, these authors ask what happens in P-time. A typical theorem is that while there exists a p-time C^2 computable differential equation with no computable solution, analytic ones do possess computable solutions.

Finally another very interesting topic I have not covered is that of bounded query reducibilities. Here one examines the effect of circumscribing the allowable type of questions (and their number) of an oracle in a reduction. This is a very lively area with a lot of nice work. Key figures include Amir, Biegel, Gasach and others. A particularly pretty illustration of this area is a recent result of Kummer [**ta**]: Suppose that A and B are any sets, $n \in \mathbb{N}$ and suppose there is a reduction Γ, that on input $\langle x_1, \ldots, x_{2^n}\rangle$ and oracle B computes $|\{x_1, \ldots, x_{2^n}\} \cap A|$ *by only* $\leq n$ *questions to* B. *Then* A *is recursive.* (2^n is sharp here). Gasach has a survey article in the 1991 6th annual structures conference and we refer the reader to the IEEE proceedings.

In any case we will stop with the work on fixed parameter sets. This seems appropriate as we may be finishing with what we hope will be a very interesting re-examination of NP-completeness that should yield much deeper insights, and is still in its infancy.

References

1989. K. Abrahamson, J. Ellis, M. Fellows and M. Mata, *Completeness for families of fixed parameter problems*, in *Proc. 29th IEEE FOCS*.

ta. K. Abrahamson, M. Fellows and R. Downey, *Fixed Parameter Tractability and Completeness IV: W[P] and PSPACE*, (to appear).

1978. L. Adleman, *Two theorems on random polynomial time*, in *Proc 19th IEEE, FOCS*, pp. 75–83.

1977. L. Adleman and K. Manders, *Reducability, randomness and intractibility*, in *Proc 9th ACM STOC*, pp. 151–163.

1974. A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Compute Algorithms*, Addison-Wesley.

1986. M. Ajtai, L. Babai, P. Hajnal, J. Komlos, P. Pudlak, V. Rodl, E. Szemeredi and G. Turan, *Two lower bounds for branching programmes*, in Proc. 18th ACM STOC, pp. 30–38.

1987. N. Alon, and R. Boppana, *The monotone circuit complexity of boolean functions*, Combinatorica **7**, 1–22.

1986. N. Alon, and W. Maass, *Meanders, Ramsey theory and lower bounds for branching programmes*, in *Proc 27th IEEE FOCS*.

1988. N. Alon and W Maass, *Meanders and their applications in lower bound arguments*, J.C.S.S. **37**, 118-129.

1984. K. Ambos-Spies, *On the Structure of the Polynomial Degrees of Recursive Sets*, Habilitationschrift, Universitat Dortmund.

1985. K. Ambos-Spies, *Three theorems on polynomial degrees of NP sets*, in *Proc. 26th IEEE FOCS*.

1986. K. Ambos-Spies, *Inhomogeneities in the polynomial time degrees*, Inf. Proc. Letters **22**, 113–117.

1986. K. Ambos-Spies and A. Nies, *The theory of polynomial many one degrees of recursive sets is undecidable*, (to appear).

1988. L. Babai and S. Moran, *Arthur-Merlin games: A ramdomized proof system and a hierarchy of complexity classes*, JCSS **36**, 254–276.

1975. T. Baker, J. Gill and R. Solovay, *Relativizations of the $P =?NP$ question*, SIAM J. of Comput. **4**, 431–442.

1986a. J. Balcazar, R. Book and U. Schöning, *The polynomial time hierarchy and sparse oracles*, J.A.C.M. **33**, 603–617.

1986b. J. Balcazar, R. Book and U. Schöning, *Sparse sets, lowness and highness*, SIAM J. of Comput. **15**, 739–747.

1988. J. Balcazar, J. Diaz and J. Gabarro, *Structural Complexity I*, Springer Verlag.

1990. J. Balcazar, J. Diaz and J. Gabarro, *Structural Complexity II*, Springer-Verlag.

1986. D. Barrington, *Bounded width polynomial size branching programmes recognise exactly those languages in NC^1*, in *Proc. 18th ACM STOC*, pp. 1–5.

ta. R. Beigel, N. Reingold and D. Spielman, *PP is closed under intersection*, Technical Report, Yale University, pp. 1990.

1989. R. Beigel, L. Hemachandra and G. Wechsung, *On the power of probabilistic polynomial time; $P^{\mathrm{NP[log]}} \subseteq PP$*, in *Proc. 4th Annual Structures in Complexity theory*, pp. 225-227.

1981. C. Bennett and J. Gill, *Relative to a random oracle A, $P^A \neq NP^A \neq co\text{-}NP^A$ with probablity 1*, SIAM J. of Comput. **10**, 96–113.

1976. L. Berman, *On the structure of complete sets*, in *17 IEEE FOCS*, pp. 76–80.

1977. L. Berman, *Polynomial Time Reducibilities and Complete Sets*, Ph.D. Dis., Cornell University.

1977. L. Berman and J. Harmanis, *On isomorphism and density of NP and other complete sets*, SIAM J. of Comput. **6**, 305–322.

1975. L. Blum and M. Blum, *Toward a mathematical theory of inductive inference*, Inf. Control **28**, 125–155.

1967. M. Blum, *A machine independent theory of complexity of recursive functions*, J.A.C.M. **18**, 322–336.

1986. M. Blum, *How to prove a theorem so that no one else can claim it*, in *Proc. Int. Congress of Mathematicians*, Berkeley.

1973. M. Blum and I. Marques, *On complexity properties of recursively enumerable sets*, J.S.L. **38**, 579–593.

1984. N. Blum, *A boolean function requiring* $3n$ *network size*, Theor. Comput. Sci. **28**, 337–345.

1972. R. Book, *On languages accepted in polynomial time*, SIAM J. of Comput. **1**, 281–287.

1974a. R. Book, *Comparing complexity classes*, J.C.S.S. **9**, 213–229.

1974b. R. Book, *Tally languages and complexity classes*, Inf. and control **26**, 186–193.

1976. R. Book, *Translation lemmas, polynomial time and* $(\log n)^j$*-space*, Theor. Comput. Sci. 1, 215–226.

1984. R. Book, J. Balcazar, T. Long, U. Schöning and A Selman, *Sparse oracles and uniform complexity classes*, in *Proc. 25th IEEE. FOCS*, pp. 308–311.

1988. R. Book, P. Orponen, D. Russo and D. Watanake, *On exponential lowness*, SIAM J of Comput. **17**, 50–56.

1966. W.W. Boone, *Word problems and recursively enumerable degrees of unsolvability*, Ann. of Math. **84**, 49–84.

1984. R. Boppana, *Threshold functions and bounded depth monotone circuits*, in *Proc. 16th STOC*, pp. 475–479.

1987. R. Boppana, J. Hastad and S. Zachas, *Does co-NP have short interactive proofs?*, Inf. Proc. Letters **25**, 127–132.

1987. R. Boppana and J. Lagavias,, *One way functions and circuit complexity,*, in *Springer-Verlag Lecture Notes Computer Science*, (ed A. Selman), pp. 51–65.

1973. R. Borodin, *Computational complexity: theory and practice*, in *Currents in the Theory of Computing*, (ed A. Aho), Prentice-Hall, pp. 35-89.

1971. R. Borodin, *On relating time and space to size and depth*, SIAM J of Comput. **6**, 733–744.

1982. A. Borodin and S. Cook, *A space time trade-off for sorting on a general sequented model of computation*, SIAM J of Comput. **11**, 287–297.

1989. A. Borodin, A, S. Cook, P. Dymond, W. Ruzzo and M. Tompa, *Two applications of complementation via inductive counting*, SIAM J. of Comput. **18**, 559–578.

1986. J. Cai, *With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy*, in *Proc. 18th ACM STOC*, pp. 21–30.

1966. G. Chaitin, *On the length of programmes for computing finite binary sequences,*, J.A.C.M. **13**, 547–569.

1969. G. Chaitin, *On the simplicity and speed of programmes for computing definite sets of natural numbers*, J.A.C.M. **16**, 407–412.

1974. G. Chaitin, *Information-theoretic limitations of formal systems*, J.A.C.M **21**, 403-424.

1984. A. Chandra, D. Kozen and L. Stockmeyer, *Alternation*, J.A.C.M **28**, 114–133.

1981. P. Chew and M. Machtey, *A note on structure and looking back applied to the relative complexity of computable functions*, J.C.S.S **22**, 53–59.

ta. P. Cholak and R. Downey, *Undecidadility results for parameterized polynomial time tractability*, (to appear).

1936. A. Church, *An unsolvable problem of elementary number theory*, Amer. J. Math **58**, 345–363.

1964. A. Cobram, *The intrinsic computational difficulty of functions*, in *Proc. Inf. Congress for Logic*, Methodology and Phil. of Science, Borth-Hollan, pp. 24-30.

1972. J. Conway., *Unpredictable iterations*, in *Number Theory Conference*, Univ. of Colorado, Proceedings, Springer-Verlag, pp. 49–52.

1971. S.A. Cook, *The computational complexity of theorem proving procedures*, in *Proc. 3rd ACM STOC*, pp. 151–158.

1973. S.A. Cook, *An observation of time-storage trade-off*, in *Proc. 5th ACM STOC*, pp. 29–33.

1985. S.A. Cook, *A taxonomy of problems with fast parallel algorithms*, Inf. and Control **64**, 2–22.

1987. S.A. Cook and P. McKenzie, *Problems complete for deterministic logarithmic space*, J. Algorithms **8**, 385-394.

1976. S. Cook and R. Sethi, *Storage requirements for deterministic time recognisable languages*, J.C.S.S. **13**, 25–37.

1962. M. Davis, G. Logemann and D. Loveland, *A machine program for theorem proving*, Comm. ACM **5**, 394–397.

1962. M. Davis and E. Weyeuker, *Computability Complexity and Languages*, Academic Press.

1988. M. Dietzfelbinger and W. Maass, *Lower bounds via 'inaccessible numbers'*, J.C.S.S. **36**, 313–335.

ta. M. Dietzfelbinger, W. Maass and G. Schnifger, *The complexity of matrix transposition for one-tape off-line Turing machines*, Theor. Comput. Sci. (to appear).

1989. R.G. Downey, *On Ramsey-type Theorems and their applications*, Singapore Math Medley **17**, 58–78.

1991. R.G. Downey, *On computational complexity and honest polynomial degrees*, Theor. Comput. Sci. **78**, 305–317.

ta. R.G. Downey, *Nondiamond theorems for polynomial time reducibilities*, J.C.S.S. (to appear).

1989. R.G. Downey, W. Gasarch, S. Homer and M. Moses, *On honest polynomial reductions, relativisations, and $P = NP$*, in *Proc. IEEE 4th Structures in Complexity*, pp. 196–207.

ta. R.G. Downey, W. Gasarch and M. Moses, *On the structure of honest polynomial m-degrees*, (to appear).

ta. R.G. Downey and M. Fellows, *Fixed Parameter Tractability and Completeness I: Basic Results*, (to appear).

ta. R.G. Downey and M. Fellows, *Fixed Parameter Tractability and Completeness II: On Completeness for W[1]*, (to appear).

ta. R.G. Downey and M. Fellows, *Fixed Parameter Tractability and Completeness III: Structural Aspects of the W-Hierarchy and Density*, (to appear).

ta. R.G. Downey and M. Fellows, *Fixed Parameter Tractability and Completeness*, monograph in preparation.

ta. R.G. Downey and M. Fellows, *Fixed Parameter Tractability*, monograph in preparation, Congressus Numerantium (to appear).

ta. R.G. Downey, M. Fellows, and T. Slaman, *Fixed Parameter Tractability and Completeness V: a general density theorem*, (to appear).

1965. J. Edmonds, *Minimum partition of a matroid into independent sets*, Res. Nat. Bur. standards set B **69**, 67–72.

1989. M. Fellows, *The Robertson-Seymour theorems; a survey of applications*, in Contempory Mathematics, Amer. Math. Soc. **89**, 1–18.

ta. M. Fellows and N. Koblitz, *On security of small secret keys and parameterized complexity*, (to appear).

1987. M. Fellows and M. Langston, *Nonconstructive advances in polynomial time complexity*, Inf. Proc. Letters **26**, 157–162.

1989. M. Fellows and M. Langston, *Nonconstructive tools for proving polynomial time decidability*, J.A.C.M.

1989. M. Fellows and M. Langston, *Layout permutation problems and well-partially-ordered sets*, in *Proc. 5th MIT. Conf. on Advanced Research in VLSI*.

1979. J. Ferrante and C. Rackoff, *Computational Complexity of Logical Theories*, Springer Lecture notes in Math, 718.

1987. L. Fortnow, *The complexity of perfect zero-knowledge*, in *Proc 18th ACM STOC*, pp. 204–209.

1987. L. Fortnow and M. Sipser, *Interactive proof systems with a log space verifier*, Preprint MIT.

1991. L. Fortnow and N. Reingold, *PP is closed under truth-table reductions*, FOCS (to appear).

1957. R.M. Friedberg, *Two recursively enumerable sets of incomparible degrees of unsolvability Proc.*, Natl. Acad. Sci. USA **43**, 236–238.

1985. H. Friedman, N. Robertson and P. Segmour, *The metamathematics of the graph minor theorem*, Amer. math. Soc. **65**, 229–262, in *Contempory Mathematics*, (ed S. Simpson), pp..

1984. M. Furst, J. Saxe and M. Sipser, *Party, circuits and the polynomial time hierarchy*, Math. Sys. Theory **17**, 13–27.

1986. Z. Galil, R. Kannan and E. Szemeredi, *On nontrivial separators for k-page graphs and simulations by deterministic one tape Turing machines*, in *Proc. 18th ACM STOC*, pp. 39–49.

1989. K. Ganesan and S. Homer, *Complete problems and strong polynomial time reducibilities*, in *Proc 6th STOC*.

1978. M. Garey and D. Johnson, *Computers and Intractability : a Guide to the theory of NP Completeness*, Freeman.

1983. W. Gasarch and S. Homer, *Relativisations comparing NP and exponential time*, Inf. Control **58**, 88–100.

ta. W. Gasarch, M. Pleszkoch and R. Solovary, *Learning via queries in* $[+,<]$, (to appear).

1972. J. Gill, *Probabilistic Turing Machines and Complexity of Computations*, Ph.D. Dis U.C. Berkeley.

1977. J. Gill, *Computational complexity of probabilistic Turing machines*, SIAM J. of Comput. **6**, 675–695.

1931. K. Godel, *On formally undecidable propositions of Principia Mathematica and related systems*, Monat. fur Math. und Phys. **38**, 173–198.

1967. E. Gold, *Language identification in the limit*, Inf. Control **10**, 447–474.

1982. A. Goldberg, P. Purdam and C. Brown, *Average time analysis of the simplified Davis-Putram procedure*, Inf. Proc. Letters **15**, 72–75.

1982. L. Goldschlager, R. Shaw and J. Staples, *The maximum flow problem is logspace complete for P*, Theor. Comput. Sci. **21**, 105–111.

1988. S. Goldreich, *Randomness, interactive proofs and zero-knowledge; a survey*, in *The Universal Turing Machine : A Half Century Survey*, (ed, R. Herken) Kammerer and Unverzagt, Berlin, pp. 377–405.

1987. S. Goldreich, Mansour and Sipser, *Interactive proof systems: Provers that never fail and random selection*, in *Proc 26th ACM FOCS*, pp. 174–187.

1986. S. Goldreich, S. Micali and A. Widgerson, *Proofs that yield nothing but their validity and a methodology for protocol design*, in it Proc 27th IEEE FOCS, pp. 174–187.

1983. S. Goldwasser, S. Micali and C. Rackoff, *The knowledge complexity of interactive proofs*, in *Proc 17th ACM STOC*, pp. 291–305.

1986. S. Goldwasser and M. Sipser, *Private coins versus public coins in interactive proof systems*, 18th ACM STOC., 59–68.

1980. R. Graham, B. Rothschild and J. Spencer, *Ramsey Theory*, Wiley.

1989. V. Gurevich, *The challanger-solver grame : variations on* $P = NP$, in *EATCS Bulletin*, pp. 122–121.

1987. V. Gurevich and S. Shelah, *Expected computation time for the hamilton path problem*, SIAM J. Comput. **16**, 486–502.

1983. J. Hartmanis, *On sparse sets in $NP - P$*, Inf. Proc. Letters **16**, 55–60.

198+. J. Hartmanis, *The Structual Complexity Column*, in *in the EATCS Bulletin.*

1978. J. Hartmanis and L. Berman, *On polynomial time isomorphisms of some new complete sets*, J.C.S.S. **16**, 418–422.

1976. J. Hartmanis and J. Hopcroft, *Independence results in computer science*, SIGACT news **8**, 13–24.

1985. J. Hartmanis, N. Immerman and V. Sewelson, *Sparse sets in $NP - P$*, Inf. Control **65**, 158–181.

1965. J. Hartmanis, P. Lewis and R. Stearns, *Classification of computations by time and memory requirements*, *Proc. IFIP conf Spartan*, New York, pp. 31–35.

1987. J. Hartmanis and L. Hemachandra, *One way functions, robustness, and the nonisomorphism of NP-complete sets*, in *Proc. 2nd Structures in Complexity*, pp. 160–174.

1965. J. Hartmanis and R. Stearns, *On the computational complexity of algorithms*, Trans A.M.S. **117**, 285–306.

1987. J. Hastad, *Computational Limitations of Small Depth Circuits*, MIT Press.

1984. H. Heller, *Relativized polynomial hierarchies extending two levels*, Math. Sys. theory **17**, 71–84.

1989. L. Hemachandra and G. Wechsung, *Using randomness to characterise the complexity of computation*, in *Information Processing*, North-Holland, pp. 281–285.

1981. S Homer and T. Long, *Honest polynomial degrees and $P = NP$*, Theor. Comput. Sci. **51**, 265–280.

1977. J. Hopcroft, W. Paul and L. Valiant, *On time versus space*, JACM **24**, 322–337.

1969. J. Hopcroft and J. Ullman, *Formal Languages and Their Relation to Automata*, Addison-Wesley.

1988. N. Immerman, *Nondeterministic space is closed under complementation*, SIAM J. of Comput. **17**, 935–938.

198+. P. Johnson, *The NP-completeness Column*, in *J. Algorithms.*

1989. L.Impagliazzo, L. Levin and M. Luby, *Pseudo-random generators from one way functions*, in *Proc. 21st ACM STOC*, pp. 12–24.

1972. R. Karp, *Reducibility among combintorial problems*, in *Complexity of Computer Computations*, (ed R. Miller and W. Thatcher), Plenum Press, pp. 85–103.

1975. R.M. Karp, *The fact approximate solution of hard combinatorial problems*, in *Proc. 6th Southeastern Conf. on Combinatorics*, Utilitas, Winnipey, pp. 15-31.

1976. R. Karp, *The probabilistic analysis of some combinatorial search algorithms*, in *Algorithms and Complexity*, (F. Traub, ed) Academic Press, pp. 1–19.

1985. Ko Ker-I, *Continuous optimization problems and a polynomial hierarchy of real functions*, J. Complexity **1**, 210–231.

1985. Ko Ker-I, *Relativized polynomial time hierarchy having exactly k levels*, Proc. 20th ACM STOC, 245–253.

1985. Ko, Ker-I and U.Schöning, *On circuit size complexity and the low hierarchy in NP*, SIAM J. Comput. **14**, 41–51.

1988. M. Krentel, *The complexity of optimization problems*, J.C.S.S. **36**, 490–509.

1965. A. Kolmogorov, *Three approaches for dfining the concept of information quantity*, Probl. Inf. Transmission **1**, 1–7.

1968. A. Kolmogorov, *Logical basis for information theory and probability theory*, IEEE Trans. on Inf. theory **14**, 662–664.

1986. E. Krankis, *Primality and Cryptography,*, Wiley-Teubner Series in Computer Science, Stuttgart.

ta. M. Kummer, *A proof of Beigel's cardinality conjecture*, (to appear).

1983. S.A. Kurtz, *On the random oracle hypothesis*, Inf. Control **57**, 40–47.

1975a. A. Ladner, *On the structure of polynomial time reducibility*, J.A.C.M. **22**, 155–171.

1975b. R. Ladner, *The circuit value problem is log space complete for P. SIGACT News.*

1975. R. Ladner, N. Lynch and A. Selman, *A comparison of polynomial time reducibilities*, Theor. Comput. Sci. **1**, 103–123.

1973. L.A. Levin, *Universal sorting problems*, Problems of Information Transmission **9**, 265–266.

1984. L.A. Levin, *Problems complete in "average" instance*, in *proc. 16th ACM STOC.*.

1985. L.A. Levin, *One way functions and pseudo-random generators*, Proc 17th ACM STOC, 363–365.

1985. Ming Li, *Lower bounds by Kolomogorov complexity*, in *Proc. 12th Int. Coll. on Antomata Languages and Programming*, Springer Lecture notes in Computer Science, pp. 383–393.

1988. Ming Li and P. Vitanyi, *Two decades of applied Kolmogorov complexity*, in *Proc 3rd structures in complexity*, pp. 80–101.

1985. T. Long, *On restricting the size of Oracles compared with restricting access to oracles,*, SIAM J. Comput. **14**, 585–597.

1986. T. Long and A. Selman, *Relativising complexity classes with sparse sets*, J.A.C.M. **33**, 618–628.

1990. C. Lund, L Fortnow, H. Karloff and N. Nisan, *Algebraic methods for interactive proof systems*, in *Proc 31st IEEE, FOCS*, pp. 2–10.

1984. W. Maass, *Quadratic lowr bounds for deterministic and non deterministic one-tape Turing mahcines*, in *Proc. 16th ACM STOC*, pp. 401–408.

1985. W. Maass, *Combinatorial lower bounds for deterministic and nondeterministic Turing machines*, Trans. Amer. Math. Soc. **293**, 675–693.

1987. W. Maass, G. Schnitger and E. Szemeredi, *Two tapes are better than one for off line Turing machines*, in *Proc. 19th ACM STOC*, pp. 94-100.

1988. W. Maass, *On the use of inaccessible numbers and order indiscernables in lower bound arguments for random access machines*, JSL **53**, 1098–1109.

ta. W. Maass and T. Slaman, *The complexity types of computable sets*, (to appear).

1978. M. Machtey and P. Young, *An Introduction to the General Theory of Algorithms*, North-Holland.

1982. S. Mahaney, *sparse complete sets for NP : a solution to a conjecture by Berman and Hartmanis*, JCSS **25**, 130–143.

1978. K. Manders and L. Adleman, *NP-complete problems for binary quadratics*, J.C.S.S. **16**, 168–184.

1970. Y. Matajeseivic, *Enumerable sets are diophantine*, Dok. Akad. Nauk, SSSR **191**, 279–282.

1969. M. McCreight and A. Meyer, *Classes of computable functions defined by bounds on computation*, ACM STOC, 79–81.

1977. G. Metakides and A. Nerode, *Recursively enumerable vector spaces*, Ann. Math. Logic **11**, 141–171.

1976. G.L. Miller, *Riemann's hypothesis and tests for primality*, J.C.S.S. **13**, 300–317.

1967. M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-hall.

ta. S. Minyano, S. Shiraishi and T. Shoudai, *A list of P-complete problems*, Kyushu Univ. Tech. Report (to appear).

1985. S. Moran, M. Snir and V. Manber, *Applications of Ramsey's theorem to decision tree complexity*, J.A.C.M. **32**, 938–949.

1988. C. Papadimitrou and Y. Yannakakis, *Optimization, approximation and complexity classes*, Proc. 20th ACM STOC, 229–234.

1979. W. Paul, *Kolmogorov complexity and lower bounds*, in *Proc. 2nd Inf. Conference on fundamentals of computer theory*, (ed. L. Budach), Akademie-Verlay, pp. 325-335.

1983. W. Paul, N. Pippenger, E. Szemeredi and W. Trotter, *On determinism versus nondeterminism and related problems*, in *Proc. 24th IEEE FOCS*, pp. 429–438.

1981. W. Paul and R. Reichuk, *On time versus space II*, J.C.S.S. **22**, 312–327.

1979. N. Pippinger, *On simultaneous resource bounds*, in *Proc. 20th IEEE, FOCS*, pp. 307–311.

1980. N. Pippinger, *Pebbling*, in *Proc. of the 5th IBM Symp. on the Math. Foundations of Computer Science*, IBM, Japan, pp. 1–19.

1976. D. Plaisted, *Some polynomial and integer divisibility problems are NP-hard*, in *Proc. 17th IEEE FOCS*, pp. 264–267.

1944. E. Post, *Recursively enumerable sets of integers and their decision problems*, Bull. Amer. Math. Soc. **50**, 284–316.

1989. M. Pour-El and I. Richards, *Computability in Anaysis and Physics*, Springer Verlag (Omega Series).

1975. V. Pratt, *Every prime has a succinct certificate*, SIAM. J. of Comput. **4**, 214–220.

1976. M. Rabin, *Probabilistic algorithms*, in *Algorithms and Complexity*, (J. Traub, ed), Academic Press, pp. 21–39.

1930. F. Ramsey, *A problem in formal logic*, Proc. London Math. soc **30**, 264–286.

1985. A. Razborov, *Lower bounds for Monotone complexity of some boolean functions*, Dokl. Akad. Nauk SSSR **281**, 798–801.

ta. A. Razaborov, *Lower bounds for the size of circuits of bounded depth with basis* $\wedge$, $\oplus$, Mat. Zam (to appear).

1983. N. Robertson and P.D. Seymour, *Graph Minors I. Excluding a forest*, J. Combinatorial Theory Ser. b **35**, 39–61.

1983. N. Robertson and P. D. Seymour, *Graph Minors II. Algorithmic aspects of tree-width*, J. Algorithms **7**, 309–322.

1983. N. Robertson and P. D. Seymour, *Graph Minors V. Excluding a planar graph*, J. Combinatorial Theory Ser. B **41**, 92–114.

1983. N. Robertson and P. D. Seymour, *Graph Minors VIII. A Kuratowski theorem for general surfaces*, manuscript.

1983. N. Robertson and P.D. Seymour, *Graph Minors X. Obstructions to tree-decompositions*, manuscript.

1983. N. Robertson and P. D. Seymour, *Graph Minors XII. Excluding a non-planar graph*, manuscript (June 1986).

1983. N. Robertson and P.D. Seymour, *Graph Minors XIII. The disjoint paths problem*, manuscript (September 1986).

1983. N. Robertson and P.D. Seymour, *Graph Minors XIV. Taming a vortex*, manuscript (1987).

1983. N. Robertson and P. D. Seymour, *Graph minors XV. Surface hypergraphs*, manuscript (1987).

1983. N. Robertson and P.D. Seymour, *Graph minors XVI. Wagner's conjecture*, manuscript (1987).

1967. H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill.

1985. A. Salomaa, *Computation and Automata*, Cambridge University Press.

1970. W. Savitch, *Relationships between nondeterministic and deterministic tape complexities*, JCSS **4**, 177–192.

1970. U. Schöning, *A low and a high hierarchy within NP*, JCSS **27**, 14–28.

1987a. U. Schöning, *Graph isomorphism is in the low hierarchy*, in *4th Symp. on Theoretical Aspects of Comput. Sci.*, Springer Lecture notes in Computer Science, pp. 114–124.

1987b. U. Schöning, *Probabilistic complexity classes and lowness*, Proc. 2nd Structures in Complexity, 2–8.

1979. A. Selman, *P-selective sets, tally languages, and the behaviour of polynomial time reducibility on NP*, Math. Sys. Theory **52**, 36–51.

1982. A. Shamir, *A polynomial algorithm for breaking the basic Merkle-Hellman Cryptosystem*, Proc. 23rd IEE FOCS, 145–152.

1990. A. Shamir, *IP = PSPACE*, in *31st ACM FOCS*, pp. 11–15.

1990. J. Shinoda and J.A. Slaman, *On the theory of PTIME degrees of recursive sets*, J.C.S.S. **41**, 321–366.

ta. R. Shore and T. Slaman, *The p-T-degrees of the recursive sets; lattice embeddings, extensions of embeddings and two quantifier theory*, (to appear).

1983. M. Sipser, *Boolean sets and circuit complexity*, in *Proc 15th ACM STOC*, pp. 61–69.

1983. S. Smale, *On the average number of steps of the simplex method of linear programming*, Math. Programming **27**, 241–262.

1977. R.I. Soare, *Computation complexity, speedability and levelable sets*, J.S.L. **42**, 545–563.

1987. R.I. Soare, *Recursively Enumerable Sets and Degrees*, Springer-Verlag (Omega Series).

1977. R. Solovay and V. Strassen, *A fast Monte-Carlo test for primality*, SIAM J. of Comput. **6**, 84–85; Errata, ibid **7** (1978), 118.

1973. L. Stockmeyer, *Planar 3-colourability is NP-complete*, SIGACT News **5**, 19–25.

1977. L. Stockmeyer, *The polynomial time hierarchy*, Theor. Comp. Sci. **3**, 1–22.

1988. R. Szelepcsenyi, *The method of forced enumeration for nondeterministic automata*, Acta Info. **26**, 279–284.

1987. S. Toda, $\sum_2$*-SPACE is closed under complementation*, J.C.S.S. **35**, 145–152.

1989. S. Toda, *PP is* $\leq^{pLLT}$*-hard for the polynomial time hierarchy,*, in *Proc. 30th IEEE STOC*, pp. 514–519.

1984. B. Trakhtenbrot, *A survey of Russian approaches to perebor (brute force search) algorithms*, Ann. History of Comput. **6**, 384–401.

1936. A. Turing, *On computable numbers with an application to the Entscheidungs problem*, Proc. London Math. Soc. **2**, 230–265.

1976. L. Valiant, *Relative complexity of checking and evaluating*, Inf. Proc. letters **5**, 20–23.

1985. L. Valiant and V. Vazirani, *NP is as easy as detecting unique solutions*, in *Proc 17th ACM STOC*.

1988. R. Venkatesan and l. Levin, *Random instances of a graph colouring problem are hard*, in *Proc 20th ACM, STOC*.

1986. K. Wagner and G. Wechsung, *Complexity Theory*, Reidel.

1985. O. Watanabe, *On one-one polynomial time equivalence relations*, Theor. Comput. Sci. **38**, 157–165.

1988. O. Watanabe, *On hardness of one-way functions*, Inf. Proc. Letter **27**, 151–157.

1985. H. Wilf, *Some examples of combinatorial averaging*, Amer. Math. Monthly **92**, 250–261.

1983. P. Young, *Some structural properties of polynomial reducibilities and sets in NP*, in *Proc. 15th ACM STOC*, pp. 292–401.

1982. A. Yao, *Theory and applications of trap door functions*, in *Proc. 23rd IEEE FOCS*, pp. 80–91.

1985. A. Yao, *Separating the polynomial time hierarchy by oracles*, in *Proc 26th IEEE FOCS*, pp. 1–10.

Rod Downey
Department of Mathematics
Victoria University of Wellington
PO Box 600
NEW ZEALAND