One-Way Functions, Robustness, and the Non-Isomorphism of NP-Complete Sets

Juris Hartmanis*
Lane A. Hemachandra†

TR 86-796
December 1986

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501

*Work supported by NSF Research Grant DCR-8520597

†Work supported by a Fannie and John Hertz Fellowship and NSF Research Grant DCR-8520597

One-Way Functions, Robustness, and the Non-Isomorphism of NP-Complete Sets

*Juris Hartmanis**
Lane A. Hemachandra†

Department of Computer Science
Cornell University
Ithaca, NY 14853

Abstract

This paper

1. gives a relativized counterexample to the conjectured connection between the existence of one-way functions and the existence of non-isomorphic NP-complete sets,
2. establishes that in relativized worlds there are NP-complete sets that are non-isomorphic in a strong sense.
3. proves that *robust* machines squander their powerful nondeterministic oracle access in all relativizations.

(1) resolves an open question. (2) extends our knowledge about non-isomorphic NP-complete sets. (3), sharing the proof techniques of (1) and (2), enriches the nascent theory of robustness and presents a consequence of the limited combinatorial control of machines.

*Work supported by NSF Research Grant DCR-8520597.
†Work supported by a Fannie and John Hertz Fellowship and NSF Research Grant DCR-8520597.

1 Introduction

1.1 Overview of Results: p-Isomorphisms

The Berman-Hartmanis Conjecture states that all NP-complete sets are p-isomorphic[1] [BH77]. Thus it asserts that there is only *one* NP-complete set, which appears in many guises. As evidence, Berman and Hartmanis prove that all NP-complete sets known at the time of their paper, and indeed all paddable NP-complete sets, are p-isomorphic.

A rival view is championed by Joseph and Young [JY85]. Buoyed by possibly unpaddable NP-complete sets, they conjecture that all NP-complete sets are *not* p-isomorphic. They suggest the following coupling between the Berman-Hartmanis Conjecture and the nonexistence of one-way functions, which is upgraded to a conjecture in the recent FOCS paper of Kurtz, Mahaney, and Royer [KMR86].

(One-way Conjecture)

One-way functions exist if and only if there are non-p-isomorphic NP-complete sets.

Here, a one-way function is a polynomial time computable, honest (i.e., it shrinks inputs at most polynomially), one-to-one function whose inverse can *not* be computed in polynomial time. Selman [Sel85] notes that one-way functions exist if and only if P $\neq$ UP, where UP (Valiant [Val76]) is the class of languages accepted by an NP machine that never has more than one accepting path per input (i.e., a **categorical** machine). Intuitive support for the One-way Conjecture comes from these observations:

- If one-way functions do not exist, the NP-complete sets constructed by Joseph and Young are isomorphic to SAT [JY85].

[1]Sets A and B are p-isomorphic if and only if there exists a polynomial time computable, onto, one-to-one reduction of A to B whose inverse is also polynomial time computable.

- If a one-way function f exists, the image $f(\mathrm{SAT})$ will be NP-complete. However, an isomorphism between $f(\mathrm{SAT})$ and SAT conceivably would let us invert f.

We call an oracle A **reasonable** if $\mathrm{P}^A \neq \mathrm{NP}^A$. Since both the Berman-Hartmanis Conjecture and the One-way Conjecture imply that $\mathrm{P} \neq \mathrm{NP}$, it follows that in all unreasonable relativized worlds the two conjectures are trivially refuted.

Until now, no known reasonable oracle has refuted the One-way Conjecture. We construct such an oracle. Our key result, which states that there is a relativized world where the One-way Conjecture fails, is that there is an oracle so one-way functions do not exist yet non-p-isomorphic NP-complete sets exist.

Theorem 2.1 There is a reasonable (i.e., $\mathrm{P}^A \neq \mathrm{NP}^A$) oracle A for which $\mathrm{P}^A = \mathrm{UP}^A$ (that is, there are no one-way functions) yet there are sets that are $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete for NP^A and are non-p^A-isomorphic.

A key open question about isomorphisms has been: Are there reasonable relativized worlds in which the Berman-Hartmanis Conjecture fails? A recent result along this line is work by Goldsmith and Joseph [GJ86] that shows there is an oracle A for which some $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete sets for NP^A are not p-isomorphic. A subtle point is that though their sets are complete with respect to reductions accessing the oracle, their sets are non-isomorphic with respect to reductions forbidden access to the oracle (i.e., p-isomorphic as opposed to p^A-isomorphic). Since their non-isomorphic sets are complete with respect to reductions that access the oracle, a fair relativized model must allow the isomorphisms to also access the oracle. This strong result follows from our theorem.

An unpublished paper by Stuart Kurtz also proves this strong relativized refutation of the Berman-Hartmanis Conjecture [Kur83].

Corollary 2.2 There is an oracle A for which some $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete sets for NP^A are not p^A-isomorphic.

Indeed, our techniques not only prevent p^A-isomorphism but also defeat more powerful isomorphisms.

Theorem 2.3 There is an oracle A for which some $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete sets for NP^A are not isomorphic via any primitive recursive isomorphism.

Other important work related to the isomorphism conjecture has been done by Allender [All86], Hartmanis [Har83], Ko, Long, and Du [KLD86], Kurtz, Mahaney, and Royer [KMR86], Mahaney and Young [MY85], and Watanabe [Wat85].

In particular, the work of Kurtz, Mahaney, and Royer [KMR86] shows that some bounded-truth-table complete sets for exponential time have m-degrees (equivalence classes under many-one polynomial time reductions) that collapse (contain a single p-isomorphic type), and some have m-degrees that fail to collapse.

1.2 Overview of Results: Robustness

The same combinatorics that underlie our isomorphism theorem yield strong results about robustness. Informally, a robustness property of a machine is a property that a machine has with every oracle. For example, if two machines accept complementary languages for every oracle $((\forall A)[L(N_1^A) = \overline{L(N_2^A)}])$ we say that the machines are *robustly complementary.*

Schoening [Sch84] considers deterministic machines M that accept some language robustly (i.e., there is a language L so that for all oracles A, $L(M^A) = L$). He shows that the machines of this sort that for some oracle A' run in polynomial time accept exactly the NP $\bigcap$ coNP languages.

In this paper, we ask what price a machine pays to have properties robustly. Throughout the paper, all robust machines (with names like N, N_1, N_2, $\ldots$, N_i) are assumed to be *nondeterministic polynomial time Turing machines.* We discuss *robustly categorical* machines (machines that for no oracle and no input have more than one accepting path), *robustly Σ^*-accepting* machines (machines that for every oracle accept all inputs), *robustly complementary* machines (pairs of machines that accept complementary languages for every oracle), and *robustly Σ^*-spanning* machines (sets of machines whose languages union to Σ^* for every oracle). In each case we show, if P equals NP, that a machine having a property robustly is emasculated.

That is, if P equals NP we conclude that, in some cases for all oracles A and in some cases for all sparse oracles A,

- robustly categorical machines accept trivial (P^A) languages.
- robustly complementary machines accept trivial (P^A) languages.
- robustly Σ^*-accepting machines have feasibly computable functions that determine *why* they accept.
- robustly Σ^*-spanning machines have feasibly computable selector functions. (Selman [Sel79,Sel82] defines a P-selector function over a set of NP machines to be a polynomial time function that, if at least one of the machines accepts an input x, chooses one of the machines that does accept the input.)

Another way of looking at our robustness results is as a study of the complexity of pulling NP from P. If P = NP, we are interested in knowing how oracles can pull NP from P. In general, even if P = NP, there will be oracles A that separate NP^A from P^A. Indeed, Hartmanis and Hemachandra completely characterize, in terms of Kolmogorov complexity, the sparse oracles that pull NP^S from P^S when P = NP [HH86b].

However, the machines that pull NP^A from P^A are wild machines that make extensive use of their oracles. Our robustness theorems (e.g., Corollary 4.4) say that if P equals NP, no robust machine (say, no machine that has a machine that is robustly complementary to it) will be able to pull NP from P.

Corollary 4.4 If P = NP and N_i and N_j are robustly complementary (i.e., $(\forall A)\,[L(N_i^A) = \overline{L(N_j^A)}]$), then for every oracle A

$$L(N_i^A) \in \mathrm{P}^A.$$

Yet another way of looking at these results is as "machine-based" lowness results. A set A is *extended low* [Sch83] if

$$\mathrm{NP}^A \subseteq \mathrm{P}^{\mathrm{NP} \oplus A}.$$

This says we can get by with a surprisingly weak form of access to A; though NP^A might access A exponentially often, $\mathrm{P}^{\mathrm{NP} \oplus A}$ touches A at most polynomially often and has a far weaker acceptance mechanism. Our robustness results (e.g., Theorem 4.1) say that any *language* accepted by a robust (e.g., robustly categorical) machine can be accepted with exactly such weak access to A.

For example, the following theorem shows that if nondeterministic machine N_i is robustly categorical, then for every oracle A the powerful nondeterministic access to the oracle N^A can make is squandered; a weak (polynomial time) machine with $\mathrm{NP} \oplus A$ as its oracle can accept the same language.

Theorem 4.1 $(\forall A)\,[N_i^A$ is categorical$] \Rightarrow (\forall A)\,[L(N_i^A) \in \mathrm{P}^{\mathrm{NP} \oplus A}]$.

We have mentioned general interpretations of our robustness theorems in terms of the difficulty of separating NP from P when P = NP, and in terms of lowness. Some of the results have special individual meaning, as they extend our knowledge about some intriguing facets of structural complexity theory.

One example of this is the study of *why* Σ^*-accepting NP machines accept. Borodin and Demers [BD76] show that $\mathrm{P} \neq \mathrm{NP} \bigcap \mathrm{coNP}$ implies there is an NP machine N so $L(N) = \Sigma^*$, yet there is no polynomial time machine that computes accepting paths of N. Simply put, they prove that if $\mathrm{P} \neq \mathrm{NP} \bigcap \mathrm{coNP}$ there is a machine that always accepts, but we can't easily determine *why* it accepts.

Furthermore, even if P = NP, there will be many sparse oracles S and nondeterministic polynomial time machines N for which $L(N^S) = \Sigma^*$ yet we cannot easily find *why* $N^S(\cdot)$ accepts. Nonetheless, we show that if P equals NP and nondeterministic polynomial time Turing machine N accepts Σ^* for all sparse oracles S, then for all sparse oracles it will be *obvious* why N^S accepts. (This is true even for sparse oracles S for which $\mathrm{P}^S \neq \mathrm{NP}^S$.)

Corollary 4.8 If $\mathrm{P} = \mathrm{NP}$ and $L(N^S) = \Sigma^*$ for every sparse oracle S, then for every sparse set S there is a machine in P^S that on input x computes an accepting path of $N^S(x)$.

Simply put, we show that machines that please all the oracles all the time must be trivial in all worlds.

1.3 Methods Overview

The crucial method used in our "there exist one-way functions $\not\Leftarrow$ there exist non-p-isomorphic NP-complete sets" result is that of *density*. We form two NP-complete sets that are of vastly differing densities (on certain prefixes), and show that their disparate densities preclude p-isomorphism.

Our disproof of the One-way Conjecture starts with a relativization that achieves $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$ with an oracle A of a very special format. Rackoff [Rac82] essentially does this via a priority construction, and we sketch an alternate proof that *directly* diagonalizes. Both our proof and Rackoff's are patterned after the ground-breaking work of Baker, Gill, and Solovay [BGS75].

Given this underlying relativization, we display two NP-complete sets with vastly differing density properties. They can not be p-isomorphic or even p^A-isomorphic; a purported p-isomorphism would have to map some elements in the more dense set on to tremendously large elements of the less dense set—larger than the p-isomorphism can get to in polynomial time. Indeed, by making the densities radically different, we can defeat even powerful isomorphisms. For example, Theorem 2.3 shows that we can prevent all primitive recursive isomorphisms.

In summary, the inspiration for our disproof of the One-way Conjecture is the simple statement "$\mathrm{P} = \mathrm{NP}$." When $\mathrm{P} = \mathrm{NP}$, there are NP-complete sets (e.g., $\{\epsilon\}$ and $\Sigma^* - \{\epsilon\}$) of vastly different densities that are thus not p-isomorphic. In our proof, P^A is *almost* equal to NP^A (just a sparse slice of the oracle keeps P^A and NP^A apart), thus we can create NP^A-complete sets of immensely different densities, which are necessarily non-p^A-isomorphic.

The same techniques used in our underlying $\mathrm{P} = \mathrm{UP} \neq \mathrm{NP}$ relativization are crucial in our robustness proofs. Simply, if we know that ($\mathrm{P} = \mathrm{NP}$ and) for *every* oracle ("robustly") two machines are, e.g., complementary, we can iteratively either find a true accepting path of one machine, or greatly reduce the number of plausible accepting paths. It follows that the languages accepted by robust machines are trivial.

2 Results

2.1 One-Way Functions and the Berman-Hartmanis Conjecture

In this section we show that there is a reasonable (i.e., $\mathrm{P}^A \neq \mathrm{NP}^A$) oracle A for which the One-way Conjecture is contradicted.

As corollaries, we show that there are relativized worlds, A, in which there are NP^A-complete sets that are not p^A-isomorphic and, indeed, are not isomorphic even under tremendously powerful isomorphisms.

Theorem 2.1 (Main Theorem) There is an oracle A that is

1. reasonable (i.e., $\mathrm{P}^A \neq \mathrm{NP}^A$),
2. allows no one-way functions (i.e., $\mathrm{P}^A = \mathrm{UP}^A$), and
3. contradicts the Berman-Hartmanis Conjecture (i.e., there are $\leq_{\mathrm{m}}^{\mathrm{p};A}$-complete sets for NP^A that are not p^A-isomorphic).

We use the following theorem discussed in Section 3.

Theorem 3.1 There is a set $A = \mathrm{PSPACE} \oplus B$ so

1. $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$, and

2. B has only strings of lengths from the widely spaced set E. $E = \bigcup_{i \geq 0} e_i$, $e_0 = 10^{10}$, $e_i = 2^{2^{2^{2^{2^{2^{e_{i-1}}}}}}}$ for $i > 0$.

Proof of Theorem 2.1 The canonical complete set for A, $Univ_A$, is quite dense. We construct a second complete set, U_A, that has huge stretches over which it contains no strings. From this, it follows that the two sets can not be p^A-isomorphic.

Let A be the set constructed in Theorem 3.1. Thus $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$, and A has the special form described in Theorem 3.1. Let

$$Univ_A = \{1\#x\#N_i\#padding \mid N_i^A(x) \text{ accepts in at most } |padding| \text{ steps } \} \cup \{0\#y \mid y \in \Sigma^*\}.$$

Universal set $Univ_A$ is a canonical complete set for A.

Our second complete set U_A will, at lengths l "near" an element of E, code all strings of $Univ_A$ of length less than l. At lengths l "far" from an element of E, U_A will contain *no* elements. We say a length l is "near" an element of E if for some $z \in E$ we have

$$l \in [\log z, 2^{2^z}].$$

We must now show (Claim 1) that U_A and $Univ_A$ are not p^A-isomorphic and (Claim 2) that U_A is NP^A-complete with respect to $\leq_m^{\mathrm{p}, A}$ reductions.

Claim 1: U_A and $Univ_A$ are not p^A-isomorphic.

Figure 1 shows a picture of $Univ_A$ and U_A. It should be clear that U_A and $Univ_A$ are not p^A-isomorphic. Why? Look at a length, such as y in the figure, that lives deep inside the black holes of U_A. There will be over $2^y/2$ strings in $Univ_A$ of length $\leq y$. Now what strings in U_A can a p^A-isomorphism map these strings to? With y properly chosen, y is far less than the log of the e_j immediately greater than y, so the isomorphism can't reach the strings in U_A clustered around e_j. But with y properly chosen, y will be so much bigger than e_{j-1} that the total number of strings in the blocks up to e_{j-1} is much less than $2^y/2$. Thus our p^A-isomorphism is faced with the impossible task of making a one-to-one mapping from more than $2^y/2$ strings to less than $2^y/2$ strings, which is impossible.

More precisely, let f be a purported p^A-isomorphism of $Univ_A$ and U_A, w.l.o.g. f is computable in $\mathrm{DTIME}^A[n^k]$. Choose j so large that

$$2^{2^{2^{e_{j-1}}}} < (\log\log e_j)^k < \frac{\log e_j}{2}.$$

Set $y = \log\log e_j$. Now $Univ_A$ has $\geq 2^y/2 = (\log e_j)/2$ strings of length $\leq y$, since $Univ_A$ is more then 50% dense at each length. Since $y^k < \log e_j$ the strings of length $\leq y$ can not map to strings in U_A clustered around e_j (as these are all of length $\geq \log e_j$). Thus our p^A-isomorphism must map the $(\log e_j)/2$ strings onto the patches of U_A clustered around $e_1, \ldots, e_{j-1}$. Since the largest such patch extends to length $2^{2^{e_{j-1}}}$, there are less than $2^{2^{2^{e_{j-1}}}}$ strings in U_A that can be reached by the isomorphism. So our purported p^A-isomorphism must map over $(\log e_j)/2$ strings in a one-to-one fashion on to fewer than $2^{2^{2^{e_{j-1}}}}$ strings. But since we've assumed that $2^{2^{2^{e_{j-1}}}} < (\log e_j)/2$, this is impossible. So our assumption that there is a p^A-isomorphism between U_A and $Univ_A$ is contradicted and Claim 1 is established.

Claim 2: U_A is $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete for NP^A.

Clearly $U_A \in \mathrm{NP}^A$, since $Univ_A \in \mathrm{NP}^A$ and U_A just codes in $Univ_A$ strings at certain lengths. Let $L \in \mathrm{NP}^A$. We show $L \leq_{\mathrm{m}}^{\mathrm{p},A} U_A$. Given x we must reduce "$x \in L$?" to a question of membership in U_A.

Simply put, if x is close to an e_i (Case 2 below) we can reduce to a nice universal string coded into U_A, and if x is not close to any e_i (Case 1 below), we can discover by brute force all relevant strings of B and then use $\mathrm{P}^{\mathrm{PSPACE}} = \mathrm{NP}^{\mathrm{PSPACE}}$ to determine if $x \in L$.

More precisely, assume L is accepted by machine N^A running w.l.o.g. in $\mathrm{NTIME}^A[n^l]$. Here is our reduction. For all small x's (i.e., those for which neither Case 1 nor Case 2 holds) use table lookup to see if $x \in L$ and map to an appropriate element of U_A. For all sufficiently large x either Case 1 or 2 will hold.

Case 1: $(\exists i)\,[e_{i-1} \leq \log|x| \text{ and } |x|^l < e_i]$ (Figure 2a).

Since $e_{i-1} \leq \log |x|$, we can by brute force find all the strings in B of length up to e_{i-1} (by querying the oracle; recall that $A = \text{PSPACE} \oplus B$ and B has strings only at lengths $e_1, e_2, \ldots$). Now we're in great shape. Let N_j be the machine defined by: $N_j^{\text{PSPACE}}(x\#list)$ simulates $N^{\text{PSPACE}\oplus list}(x)$, that is, whenever N queries the second component of its oracle in the simulation we answer yes to the query exactly when the query is a member of *list*. $L(N_j^{\text{PSPACE}}) \in \text{NP}^{\text{PSPACE}} = \text{P}^{\text{PSPACE}}$, and when the *list* correctly includes all elements of B that could be accessed during the run of $N^A(x)$, as it does here, $N^A(x)$ accepts if and only if $N_j^{\text{PSPACE}}(x\#list)$ accepts. Thus our p^A-reduction actually determines if $x \in L(N^A)$. If $x \in L(N^A)$ we map to a fixed string known to be in U_A, otherwise we map to a fixed string known to be out of U_A.

Case 2: $(\exists e_i)\,[|x| \leq 2^{e_i} \,\wedge\, |x|^l \geq e_i \,\wedge\, \log e_i < |x\#N\#|x|^l| < 2^{2^{e_i}}]$ (Figure 2b). The string in $Univ_A$ coding the action of $N^A(x)$ is $x\#N\#|x|^l$. By the assumption of Case 2, $|x\#N\#|x|^l| \in [\log e_i,\, 2^{2^{e_i}})$. The strings of this length of $Univ_A$ are coded into U_A at length $1 + |x\#N\#|x|^l|$. So we simply reduce x to the location of U_A that $x\#N\#|x|^l$ has been coded to. Thus we've reduced "$x \in L$?" to a membership question about U_A, and proved Claim 2.

We've shown claims 1 and 2 and thus proved our Main Theorem. **QED**

We can extract from our Main Theorem strong results about non-isomorphism of NP-complete sets.

Stuart Kurtz [Kur83] in an unpublished paper has also obtained the following corollary.

Corollary 2.2 There is an oracle A for which there are $\leq_m^{p,A}$-complete sets for NP^A that are not p^A-isomorphic.

Indeed, if we choose the set E of Theorem 3.1 to be even more widely spaced, we can get NP-complete sets of vastly different densities and defeat powerful types of isomorphism.

Theorem 2.3

1. There is an oracle A for which there are $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete sets for NP^A that are not isomorphic via any primitive recursive isomorphism.

2. There is an oracle A for which there are $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete sets for NP^A that are not exponential-time^A-isomorphic.

3 An Oracle for P = UP $\neq$ NP

In this section we show that there is an oracle A making $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$ that is the disjoint union of PSPACE and a set with strings at widely spaced lengths.

Theorem 3.1 (An Oracle for P=UP$\neq$NP)
There is a set $A = \mathrm{PSPACE} \oplus B$ so

1. $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$, and
2. B has only strings of lengths from the widely spaced set E. $E = \bigcup_{i \geq 0} e_i$, $e_0 = 10^{10}$, $e_i = 2^{2^{2^{2^{2^{2^{e_{i-1}}}}}}}$ for $i > 0$.

Baker, Gill, and Solovay [BGS75] show that there is an oracle A for which $\mathrm{P}^A = \mathrm{NP}^A \bigcap \mathrm{coNP}^A \neq \mathrm{NP}^A$. Rackoff [Rac82] modifies their construction to show that there is an oracle A for which $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$; Rackoff's proof can be easily modified to prove Theorem 3.1. However, both [BGS75] and [Rac82] use explicit priority constructions. We now give a *direct* diagonalization proof of Theorem 3.1. Also, we suggest an approach to the combinatorial part of the proof that gives new insight into the structure of the family of potential accepting paths of robustly categorical machines. Our proof exploits the techniques of [BGS75], of [Rac82], of Cai and Hemachandra's relativizations of the counting hierarchy [CH86], and of Hartmanis and Hemachandra's proof that there is an oracle for which P $\neq$ UP $\neq$ NP yet UP has complete languages [HH86a].

Proof Sketch for Theorem 3.1 At stage i we take the i'th NP machine and try to make it noncategorical. Failing this we insure that it does not accept a certain

NP^A language L_A. At the end, we know $\mathrm{P}^A \neq \mathrm{NP}^A$, because no categorical Turing machine (thus no polynomial time Turing machine) accepts $L_A \in \mathrm{NP}^A$. We also give two methods of proving that $\mathrm{P}^A = \mathrm{UP}^A$; both methods exploit combinatorial limitations of the family of accepting paths of categorical machines.

The next few pages sketch the details. First time readers are encouraged to skip to Section 4.

More precisely, at stage i we work on the i'th NP machine N_i, which w.l.o.g. runs in $\mathrm{NTIME}[n^i + i]$. At the start of stage i, B has all strings up to a certain length already determined; call this oracle B_{i-1}. Our final oracle B will be $\bigcup_{j\geq 0} B_j$.

Case 1: there is an extension B' of B_{i-1} satisfying condition 2 of the theorem and a string x (of *any* length) so $N_i^{\mathrm{PSPACE}\oplus B'}(x)$ is noncategorical. In this case, set $B_i = B'$ and consider all strings of length $\leq |x|^i + i$ in B_i to be fixed. Thus we have assured that $N_i^{\mathrm{PSPACE}\oplus B}$ is noncategorical.

Case 2 Since Case 1 fails to hold we know that for every extension B' of B_{i-1} satisfying condition 2 of the theorem, $(\forall x)\,[N_i^{\mathrm{PSPACE}\oplus B'}(x)$ is categorical$]$. We use this later to insure that $\mathrm{P}^A = \mathrm{UP}^A$. For now, choose an x (1) bigger than the biggest string fixed in B_{i-1}, (2) equal to e_m for some m, and (3) big enough so $|x|^i + i \ll 2^{|x|-2}$. We wish to insure that

$$L(N_i^{\mathrm{PSPACE}\oplus B}) \neq L_A = \{x \mid (\exists y)\,[y \in B \wedge |y| = |x|]\} \in \mathrm{NP}^A.$$

Case 2a: $N_i^{\mathrm{PSPACE}\oplus B_{i-1}}(x)$ accepts. Set $B_i = B_{i-1}$ and freeze B_i up to length $|x|^i + i$. Now, $x \in L(N_i^{\mathrm{PSPACE}\oplus B}) - L_A$, so $L(N_i^{\mathrm{PSPACE}\oplus B}) \neq L_A$.

Case 2b: $N_i^{\mathrm{PSPACE}\oplus B_{i-1}}(x)$ rejects. Consider running $N_i^{\mathrm{PSPACE}\oplus(B_{i-1} \bigcup y')}(x)$, for each $y' \ni |y'| = |x|$.

Case 2bi: Some y', say y'', causes rejection. We are are done; set $B_i = B_{i-1} \bigcup y''$ and freeze B_i up to length $|x|^i + i$. $x \in L_A - L(N_i^{\mathrm{PSPACE}\oplus B})$, so $L(N_i^{\mathrm{PSPACE}\oplus B}) \neq L_A$.

Case 2bii: All y' cause acceptance. We argue that we never fall through to this case. For this case to occur, every y' must have a unique accepting path (we are robustly categorical as Case 1 failed) that gets a 'yes' answer from querying y' (as

Case 2a didn't hold). But this situation is impossible! If for some pair of paths, say the paths associated with y'_1 and y'_2, we have

$$y'_1 \notin path_{y'_2} \text{ and } y'_2 \notin path_{y'_1},$$

then $N_i^{\mathrm{PSPACE} \oplus (B_{i-1} \bigcup y'_1 \bigcup y'_2)}(x)$ has two accepting paths and is noncategorical, a contradiction. Each query along one of the $2^{|x|}$ paths can destroy one potential pair (e.g., if y'_7 is queried on the path associated with y'_{13}, the pair (y'_7, y'_{13}) is destroyed), so we can eliminate at most $(|x|^i + i)\, 2^{|x|}$ pairs. However, there are far more pairs than this; there are $\binom{2^{|x|}}{2}$ pairs, and $\binom{2^{|x|}}{2} \gg (|x|^i + i)\, 2^{|x|}$ by our choice of x. So there will be some pair (y'', y''') making $N_i^{\mathrm{PSPACE} \oplus (B_{i-1} \bigcup y'' \bigcup y''')}(x)$ noncategorical, which contradicts the fact that Case 1 didn't hold.

End of Construction

Clearly $\mathrm{P}^A \neq \mathrm{NP}^A$, as no categorical machine (and thus no P^A machine) accepts the NP^A language L_A. Why is $\mathrm{UP}^A = \mathrm{P}^A$? Let N_i^A categorically accept a UP^A language. Here is how to accept $L(N_i^A)$ in P^A. For all small strings (say up to the biggest string fixed in B at stage $i+1$ of the above construction) use table lookup to determine if $x \in L(N_i^A)$. We call an oracle C **valid** if it meets condition 2 of Theorem 3.1. For big strings x, consider the family of all possible accepting paths of $N_i^{\mathrm{PSPACE} \oplus B'}(x)$ over all valid B'. This family may be huge. Our task is to prune it down in a polynomial number of steps. We sketch two methods. Method 2, which follows [Rac82], is given in greater detail.

Method 1- Sketch A combinatorial argument shows that either the family $\mathcal{F}$ of accepting paths of $N_i^{\mathrm{PSPACE} \oplus B'}(x)$, taken over all valid B', is either (1) small (polynomial) in size or (2) contains a string c so that a respectable $(\frac{100}{|x|^{k+1}}\%)$ portion of $\mathcal{F}$'s paths explicitly ask "$c \in B$" and demand a yes answer and a respectable $(\frac{100}{|x|^{k+1}}\%)$ portion of $\mathcal{F}$'s paths explicitly ask "$c \in B$" and demand a no answer. In case (1) we use PSPACE to find all of $\mathcal{F}$ and ask all the queries of all the paths of $\mathcal{F}$ to see if any path agrees with B. In case (2), we use PSPACE to compute the nifty string c, and then ask our oracle A if c is in B. Regardless of the answer, we've eliminated a reasonable portion of $\mathcal{F}$'s size. Repeat this, finding a c' that further reduces $\mathcal{F}$'s size by a respectable portion. This is a slow divide and conquer that reduces $\mathcal{F}$, but we reduce, instead of by $1/2$ each time (in which case we'd be done in $\log|\mathcal{F}|$ steps), by a fraction dependent on $|x|$. Fortunately, we still will shrink $\mathcal{F}$ down to polynomial size in a polynomial number of steps. Noting that $|\mathcal{F}| \leq 2^{\mathcal{O}(n^k)}$, the process takes about

$$\frac{\log|\mathcal{F}|}{\log((|x|^{k+1})/(|x|^{k+1}-1))} = \mathcal{O}(|x|^{2k+2})$$

steps, i.e., polynomial in $|x|$. Thus in a polynomial number of rounds we have reduced $\mathcal{F}$ to a small size and Case 1 finishes us off.

Method 2 (After Rackoff)- Sketch Use PSPACE to find if for some valid value of B' there is an accepting computation of $N_i^{\mathrm{PSPACE}\oplus B'}(x)$; if not, reject x. Use PSPACE to get the path, say $path_0$. Query all elements in the path, let S_0 be all elements queried on the path, and let W_0 be the elements on which the path was wrong (disagreed with B). If the path was never wrong, we have a true accepting path, so accept x.

Similarly, use PSPACE to find if, for some valid value of B' consistent with our knowledge about the elements of S_0, there is an accepting computation of $N_i^{\mathrm{PSPACE}\oplus B'}(x)$; if not, reject x. Use PSPACE to get the path, say $path_1$. Query all elements in the path, let S_1 be all elements queried on the path, and let W_1 be the elements on which the path was wrong (disagreed with B). If the path was never wrong, we have a true accepting path, so accept x.

Keep repeating this. The process finishes quickly. Why? Each $path_k$ must *conflict* with each of the paths $path_0$, $path_1$, $\ldots$, $path_{k-1}$, since we were robustly categorical over all valid extensions. Note that $(\forall j, l \ni j \neq l)\,[W_j \cap W_l = \emptyset]$. Thus $path_k$ must conflict with $path_0$ on some element that is both in W_0 of $path_0$ and $S_k - W_k$ of $path_k$. Similarly, it conflicts with $path_1$ on some element that is both in W_1 of $path_1$ and in $S_k - W_k$ of $path_k$, and so on. But since the W_j's are disjoint, we take up $k-1$ spaces of $S_k - W_k$ just to disagree with the previous paths. Thus the process goes on at most until we examine $|x|^i + i$ paths. At that point we either have eliminated all paths (so $N_i^A(x)$ rejects) or we have found a path consistent with our oracle (so $N_i^A(x)$ accepts). Thus we have accepted an arbitrary UP^A language in P^A, so $\mathrm{P}^A = \mathrm{UP}^A$.

We have shown $\mathrm{P}^A = \mathrm{UP}^A \neq \mathrm{NP}^A$. **QED**

4 Robustness

4.1 Robustness Theorems

This section proves a number of new robustness results. Simply put, machines pay a heavy price for maintaining robustness properties.

Below are a number of Theorem/Corollary pairs. The theorems emphasize a "lowness" approach: an NP^A machine satisfying a robustness property is repeatedly shown to be understandable via the far weaker access method of $\mathrm{P}^{\mathrm{NP}\oplus A}$. The corollaries emphasize that if $\mathrm{P} = \mathrm{NP}$, machines satisfying a robustness property can not separate P^A from NP^A or be too complex.

Theorem 4.1 (Robustly categorical machines accept simple languages) $(\forall A)\,[N_i^A \text{ is categorical}] \Rightarrow (\forall A)\,[L(N_i^A) \in \mathrm{P}^{\mathrm{NP}\oplus A}]$.

Corollary 4.2 If $\mathrm{P} = \mathrm{NP}$ and N_i is robustly categorical (i.e., $(\forall A)\,[N_i^A$ is categorical$]$), then for every oracle A,

$$L(N_i^A) \in \mathrm{P}^A.$$

Theorem 4.3 (Robustly complementary machines accept simple languages)

$$(\forall A)\,[L(N_i^A) = \overline{L(N_j^A)}] \Rightarrow (\forall A)\,[L(N_i^A) \in \mathrm{P}^{\mathrm{NP}\oplus A}].$$

Corollary 4.4 If $\mathrm{P} = \mathrm{NP}$ and N_i and N_j are robustly complementary (i.e., $(\forall A)\,[L(N_i^A) = \overline{L(N_j^A)}]$), then for every oracle A

$$L(N_i^A) \in \mathrm{P}^A.$$

We can restrict our attention to sparse[2] sets and get similar results (with easier proofs).

Theorem 4.5 (Machines robustly Σ^*-spanning on sparse oracles have simple selector functions)
$(\forall \text{ sparse } S)[L(N_{i_1}^S) \cup \cdots \cup L(N_{i_z}^S) = \Sigma^*] \Rightarrow (\forall \text{ sparse } S)(\exists f \text{ computable in } \mathrm{P}^{\mathrm{NP}\oplus S})(\forall x)[x \in L(N_{i_{f(x)}})]$.

Corollary 4.6 If P = NP and for every sparse oracle S, $L(N_{i_1}^S) \cup \cdots \cup L(N_{i_z}^S) = \Sigma^*$, then for every sparse oracle S there is a selector function f computable in P^S that for every input x selects one of the machines that indeed accepts. That is, $(\forall \text{ sparse } S)(\exists f \text{ computable in } \mathrm{P}^S)(\forall x)$

$$x \in L(N_{i_{f(x)}}^S).$$

Theorem 4.7 (Machines robustly Σ^*-accepting on sparse oracles accept for transparent reasons)
$(\forall \text{ sparse } S)[L(N_i^S) = \Sigma^*] \Rightarrow (\forall \text{ sparse } S)(\exists f \text{ computable in } \mathrm{P}^{\mathrm{NP}\oplus S})(\forall x)[f(x)$ prints an accepting path of $N_i^S(x)]$.

Corollary 4.8 If P = NP and N_i robustly accepts Σ^* on sparse oracles (i.e., $(\forall \text{ sparse } S)$ $[L(N_i^S) = \Sigma^*]$), then for every sparse oracle S, there is a function f computable in $\mathrm{P}^{\mathrm{NP}\oplus S}$ so that on any input x

$$f(x) \text{ prints an accepting path of } N_i^S(x).$$

Theorem 4.9 (Machines robustly complementary on sparse oracles accept simple languages)
$(\forall \text{ sparse } S)[L(N_i^S) = \overline{L(N_j^S)}] \Rightarrow (\forall \text{ sparse } S)[L(N_i^S) \in \mathrm{P}^{\mathrm{NP}\oplus S}]$.

Corollary 4.10 If P = NP and N_i and N_j are robustly complementary on sparse oracles (i.e., $(\forall \text{ sparse } S)[L(N_i^S) = \overline{L(N_j^S)}]$), then for every sparse oracle S

$$L(N_i^S) \in \mathrm{P}^S.$$

[2] A set S is *sparse* if for some k, there are at every length n at most $n^k + k$ strings in S.

Those who find the sight of "P = NP" unsettling will be pleased to know that we can trade off strength of structural assumptions for strength of robustness properties as shown below.

Theorem 4.11 (Machines robustly complementary and categorical on sparse oracles accept simple languages)
$(\forall \text{ sparse } S)\,[N_i^S \text{ and } N_j^S \text{ are categorical and complementary}] \Rightarrow (\forall \text{ sparse } S)$ $[L(N_i^S) \in \mathrm{P}^{(\mathrm{UP} \bigcap \mathrm{coUP}) \oplus S}]$.

Corollary 4.12 If P = UP $\bigcap$ coUP and N_i^S and N_j^S are categorical and complementary for all sparse oracles S, then for all sparse oracles S,

$$L(N_i^S) \in \mathrm{P}^S.$$

A final point is that *all* of the above results hold **uniformly**. Taking Corollary 4.2 as an example, not only is each $L(N_i^A)$ in P^A, but there is a *single* polynomial time Turing machine that works for all A! That is, there is a polynomial time machine M so that for every A, $L(M^A) = L(N_i^A)$.

4.2 A Note on Weakening the Hypotheses

In Theorems 4.1 and 4.3 and Corollaries 4.2 and 4.4, we can restrict our hypotheses to sparse oracles. This follows from the easy observation that a machine is, e.g., categorical for all oracles exactly when it is categorical for all sparse oracles.

Lemma 4.13

1. $(\forall A)\,[N^A \text{ is categorical}] \iff (\forall \text{ sparse } S)\,[N^S \text{ is categorical}]$.
2. $(\forall A)\,[L(N_i^A) = \overline{L(N_j^A)}] \iff (\forall \text{ sparse } S)\,[L(N_i^S) = \overline{L(N_j^S)}]$.

Proof Sketch for Lemma 4.13 The $\Rightarrow$ directions are direct. The other directions hold because if a machine is, e.g., noncategorical for (dense) oracle A', it fails to be categorical on some specific string x. Thus for any sparse oracle S' that agrees with A' on a prefix large enough to include all strings queried during the run of $N^{A'}(x)$, we know that $N^{S'}(x)$ will be noncategorical. (Note that the definition of sparseness allows oracles that are quite dense on a finite prefix.)

As an example, we can restate Theorem 4.1 as follows.

Theorem 4.14 (Robustly categorical machines accept simple languages) $(\forall \text{ sparse } S)\,[N_i^S \text{ is categorical}] \Rightarrow (\forall A)\,[L(N_i^A) \in \mathrm{P}^{\mathrm{NP}\oplus A}]$.

4.3 Proof Sketches for Robustness Theorems

Theorem 4.1 This exactly follows from the proof of Theorem 3.1. Theorem 3.1 just shows how to accept in $\mathrm{P}^{\mathrm{PSPACE}\oplus A}$. However, the only way Method 2 of the proof uses PSPACE is to guess paths of a certain form, and NP also can do that just as well. **QED**

Theorem 4.3 This is like Theorem 4.1, but is a bit more involved. Recall we have machines N_i and N_j that are robustly complementary. W.l.o.g. they are respectively in NTIME$[n^i + i]$ and NTIME$[n^j + j]$. We consider the family $\mathcal{F}_i$ of paths, over all oracles A', on which $N_i^{A'}(x)$ accepts, and also consider the family $\mathcal{F}_j$ of paths, over all oracles A', on which $N_j^{A'}(x)$ accepts.

Now we use NP to get an accepting path from $\mathcal{F}_i$ and query in A all elements along the path. Then we use NP to get a path from $\mathcal{F}_j$ that is consistent with our knowledge of A on all elements in the first path. Continue this, crucially alternating between $\mathcal{F}_i$ and $\mathcal{F}_j$.

If we ever fail to find a path we know that family has no accepting paths and we are done (the machine of that family rejects). If we ever find a path that agrees with A we have a true accepting path and again are done (the machine the path belongs to accepts). Note that every pair of one path from $\mathcal{F}_i$ and one path from $\mathcal{F}_j$ must explicitly conflict over the membership of some element in A (or our machines would not be robustly complementary). But now the argument of Method 2 of the proof of Theorem 3.1 applies. Each path we take from $\mathcal{F}_j$ conflicts with each previous path from $\mathcal{F}_i$ on a different element, so our whole process terminates after at most 2 $\max(|x|^i + i,\ |x|^j + j)$ paths have been studied. **QED**

Theorems 4.9, 4.7, and 4.5 These are much easier to prove than the general theorems discussed above. We use an iterative adaptation of the method of Baker, Gill, and Solvay [BGS75] to prove, as an example, Theorem 4.7.

Let N_i be the machine of the theorem. It robustly accepts Σ^* for sparse oracles. Our goal is to, in P^S, find an accepting path of $N_i^S(x)$.

Use NP to find an accepting path of $N_i^{\emptyset}$. Query S about all elements along the path. If none are in S, then we have a true accepting path of $N_i^S(x)$ and are done. If some are in S, then we've discovered some elements of S, call them S_0.

Now use NP to find an accepting path of $N_i^{S_0}$ (there must be one, as N_i robustly accepts Σ^*). Again, if no elements on the path are in $S - S_0$, we have a true accepting path. Otherwise, we have discovered some new elements of S. Let S_1 be the union of these elements and S_0. Keep repeating this.

How long can this go on? Well, we find new elements of S at each step (or we have an accepting path and are done), but since S is sparse, it only has a polynomial number of elements that can be touched by the run of $N_i^S(x)$. So in a polynomial number of steps, we have found *all* the strings of S that $N_i^S(x)$ can touch, and the next use of NP will give us a true accepting path. **QED**

5 Conclusions and Open Problems

By repeatedly reducing the sets of plausible accepting paths, we've shown that if P equals NP robust machines are weak. With no oracle can they separate NP from P, have mysterious accepting paths, or have hard selector functions.

Thus the tragedy of a machine N that has a robustness property is that for every oracle A, N^A squanders its powerful access to its oracle. A mere polynomial time machine with oracle $\mathrm{NP} \oplus A$ can do as much as N^A.

By repeatedly reducing sets of plausible accepting paths and using the fact that sets with different density properties can't be isomorphic, we've given relativized refutations of two important conjectures.

The One-way Conjecture states that one-way functions exist if and only if there exist NP-complete sets that are not isomorphic. We've refuted this by displaying a reasonable (i.e., $\mathrm{P}^A \neq \mathrm{NP}^A$) relativized world in which there are no one-way functions but all NP-complete sets are not isomorphic.

The Berman-Hartmanis Conjecture says that all NP-complete sets are isomorphic. We've given an extremely strong relativized disproof of this conjecture. Our construction yielded a relativized world A in which there are $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete sets for NP^A that not only are non-p-isomorphic, but also are non-p^A-isomorphic.

An important open problem is: Are there relativized worlds in which (1) one-way functions exist yet (2) all NP-complete sets are isomorphic? This would give an interesting alternate refutation of the One-way Conjecture. Even (2) of our question is an open problem. It is not yet known if there is an oracle A for which all $\leq_{\mathrm{m}}^{\mathrm{p},A}$-complete sets for NP^A are p^A-isomorphic. The nearest result to this is a recent theorem of Goldsmith and Joseph [GJ86] that constructs an oracle A for which all $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete sets for NP^A are p^A-isomorphic.

References

[All86] E. Allender. Isomorphisms and 1-L reductions. In *Structure in Complexity Theory*, pages 12–22, Springer-Verlag *Lecture Notes in Computer Science #223*, 1986.

[BD76] A. Borodin and A. Demers. *Some Comments on Functional Self-Reducibility and the* NP *Hierarchy.* Technical Report TR 76-284, Cornell Department of Computer Science, July 1976.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[CH86] J. Cai and L. Hemachandra. The Boolean hierarchy: hardware over NP. In *Structure in Complexity Theory*, pages 105–124, Springer-Verlag *Lecture Notes in Computer Science #223*, 1986.

[GJ86] J. Goldsmith and D. Joseph. Three results on the polynomial isomorphism of complete sets. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 390–397, 1986.

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 439–445, 1983.

[HH86a] J. Hartmanis and L. Hemachandra. Complexity classes without machines: on complete languages for UP. In *Automata, Languages, and Programming (ICALP 1986)*, Springer-Verlag *Lecture Notes in Computer Science*, 1986.

[HH86b] J. Hartmanis and L. Hemachandra. On sparse oracles separating feasible complexity classes. In *STACS 1986: 3rd Annual Symnposium on Theoretical Aspects of Computer Science*, pages 321–333, Springer-Verlag *Lecture Notes in Computer Science #210*, 1986.

[JY85] D. Joseph and P. Young. Some remarks on witness functions for non-polynomial and non-complete sets in NP. *Theoretical Computer Science*, 39:225–237, 1985.

[KLD86] K. Ko, T. Long, and D. Du. A note on one-way functions and polynomial-time isomorphisms. In *ACM Symposium on Theory of Computing*, pages 295–303, 1986.

[KMR86] S. Kurtz, S. Mahaney, and J. Royer. Collapsing degrees. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 380–389, 1986.

[Kur83] S. Kurtz. A relativized failure of the Berman-Hartmanis conjecture. 1983. Unpublished manuscript.

[MY85] S. Mahaney and P. Young. Reductions amoung polynomial isomorphism types. *Theoretical Computer Science*, 39:207–224, 1985.

[Rac82] C. Rackoff. Relativized questions involving probabilistic algorithms. *Journal of ACM*, 29(1):261–268, 1982.

[Sch83] U. Schoening. A low and a high hierarchy in NP. *Journal of Comp. and Sys. Sci.*, 27:14–28, 1983.

[Sch84] U. Schoening. Robust algorithms: a different approach to oracles. In *Automata, Languages, and Programming (ICALP 1984)*, pages 448–453, Springer-Verlag *Lecture Notes in Computer Science*, 1984.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.

[Sel82] A. Selman. Reductions on NP and P-selective sets. *Theoretical Computer Science*, 19:287–304, 1982.

[Sel85] A. Selman. Complexity measures for public-key cryptosystems. November 1985. Unpublished manuscript.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[Wat85] O. Watanabe. On one-one P-equivalence relations. *Theoretical Computer Science*, 38:157–165, 1985.

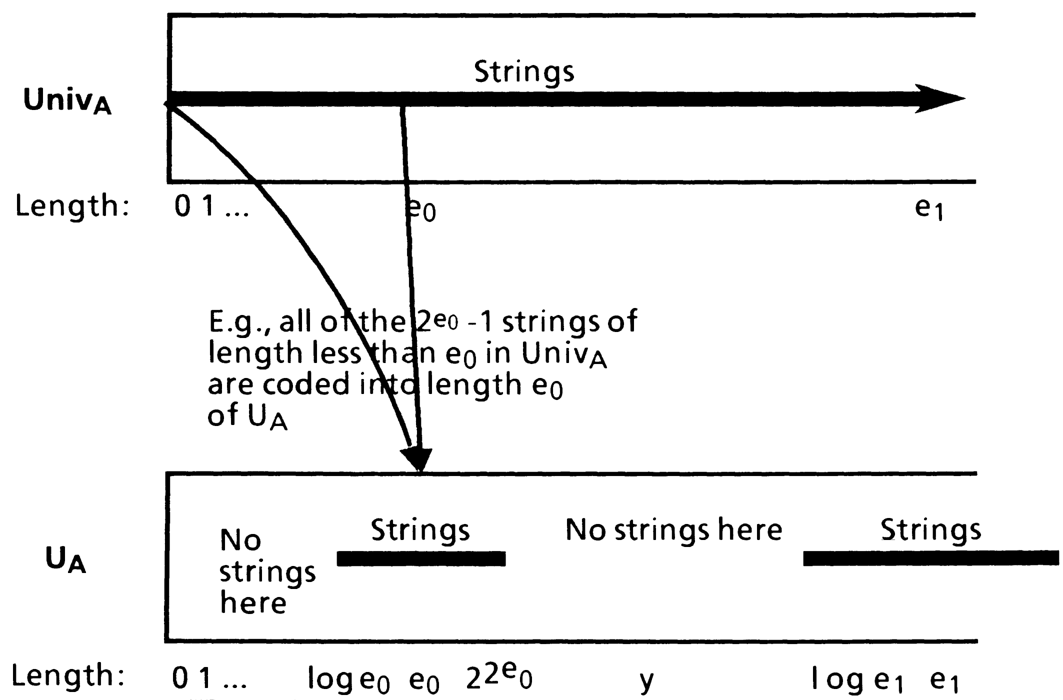

Figure 1: Location of strings in Univ$_A$ and U_A

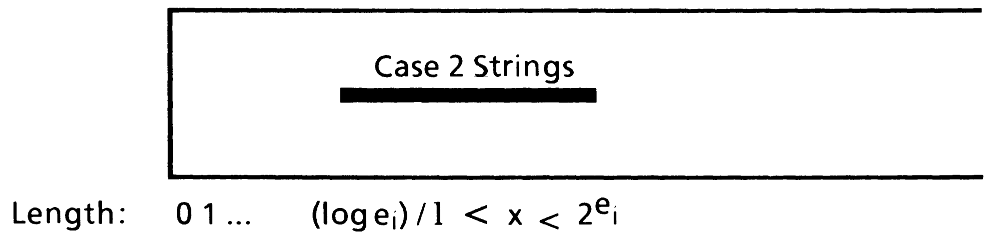

Figure 2b: Location of Case 2 Strings

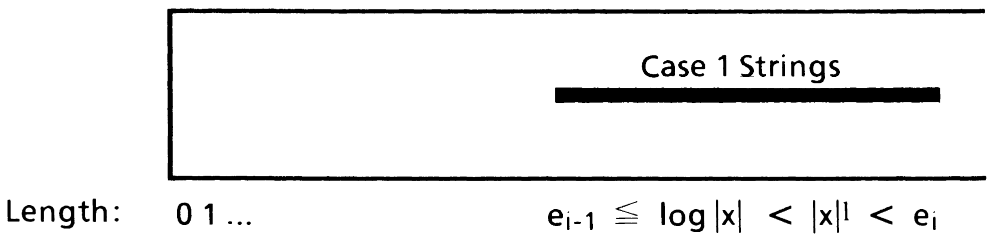

Figure 2a: Location of Case 1 Strings

Figure 2: String Locations