The Isomorphism Conjecture Holds Relative to an Oracle

Stephen Fenner[*] Lance Fortnow[†] Stuart A. Kurtz[‡]

August 16, 1996

Abstract

We introduce symmetric perfect generic sets. These sets vary from the usual generic sets by allowing limited infinite encoding into the oracle. We then show that the Berman-Hartmanis isomorphism conjecture [BH77] holds relative to any sp-generic oracle, i.e., for any symmetric perfect generic set A, all $\mathbf{NP}^A$-complete sets are polynomial-time isomorphic relative to A. Prior to this work there were no known oracles relative to which the isomorphism conjecture held.

As part of our proof that the isomorphism conjecture holds relative to symmetric perfect generic sets we also show that $\mathbf{P}^A = \mathbf{FewP}^A$ for any symmetric perfect generic A.

1 Introduction

> *Is it possible to define a notion of genericity such that all* **NP***-complete sets are p-isomorphic?* Judy Goldsmith and Deborah Joseph [GJ86]

We construct an oracle relative to which the Berman-Hartmanis Isomorphism Conjecture [BH76, BH77] is true. This conjecture holds that any two **NP**-complete sets are isomorphic to one another by a polynomial time computable and invertible one-one reduction. The Isomorphism Conjecture has been the subject of considerable research. We recommend the surveys by Joseph and Young [JY90] and Kurtz, Mahaney and Royer [KMR90].

The attempt to construct oracles relative to which the isomorphism conjecture either succeeded or failed began soon after the conjecture was made in 1976.

Success was first obtained in finding oracles relative to which the conjecture fails. In 1983, Kurtz (in an unpublished manuscript) constructed an oracle relative to which the conjecture failed. Hartmanis and Hemachandra [HH91] later combined Kurtz's construction with Rackoff's construction [Rac82] of an oracle relative to which $\mathbf{P} = \mathbf{UP}$ and thus no one-way functions exists [GS88]. In 1989, Kurtz, Mahaney and Royer [KMR89] showed that the conjecture fails relative to a random oracle; and Kurtz [Kur88] gave an improved version of his original construction that showed that the conjecture fails relative to a Cohen generic oracle.

The attempt to construct an oracle relative to which the conjecture succeeds has proven much more difficult. Even partial successes have been viewed as important advances. In 1986, Goldsmith

[*] University of Southern Maine, 96 Falmouth St., Portland, ME 04103.
Email: fenner@cs.usmsas.maine.edu. Partially supported by NSF grant CCR 92-09833.

[†] University of Chicago, 1100 E. 58th St., Chicago, IL 60637. Email: fortnow@cs.uchicago.edu. Partially supported by NSF grants CCR 90-09936 and CCR 92-53582.

[‡] University of Chicago, 1100 E. 58th St., Chicago, IL 60637. Email: stuart@cs.uchicago.edu.

and Joseph [GJ86] constructed an oracle relative to which a partially relativized version of the isomorphism conjecture holds. Namely, they constructed an oracle A such that all of the p-complete sets for $\mathbf{NP}^A$ are p^A-isomorphic.

A *m-degree* is an equivalence class of sets all many-one reducible to each other. In 1987, Kurtz, Mahaney and Royer [KMR87] gave a relativized version of their collapsing degree construction [KMR88], and showed that there is an oracle A relative to which *some* m-degree in $\mathbf{NP}^A$ collapses. Finally, in 1989, Homer and Selman [HS89, HS92] gave an oracle relative to which the complete degree for Σ_2^{P} collapsed.

We introduce a new notion of genericity, and define the *symmetric, perfect generic sets* (a.k.a. the *sp-generic sets*), and show

Theorem 1.1 *Relative to any symmetric perfect generic set A, all* $\mathbf{NP}$*-complete sets are polynomial-time isomorphic.*

We improve upon the work of Goldsmith and Joseph [GJ86] by allowing $\mathbf{NP}$-complete sets via relativized reductions.

After describing the mathematical background needed for this paper, in Section 3 we will describe sp-generic sets and give some of their properties. In Section 4 we show that $\mathbf{P}^A = \mathbf{FewP}^A$ for any sp-generic A which will form a necessary part of our proof that the isomorphism conjecture holds relative to any sp-generic oracle. In Section 5 we will give some intuition for the proof of the isomorphism conjecture followed by the detailed proof.

2 Mathematical Preliminaries

The natural numbers are denoted by $\mathbf{N}$. The cardinality of a set X is denoted by $\|X\|$. Let $\Sigma = \{0,1\}$.

We will use lower case Greek letters for partial functions from $\Sigma^* \to \{0,1\}$. We say τ *extends* σ to mean that τ is equal to σ everywhere that σ is defined. We often identify a language $A \subseteq \Sigma^*$ as its characteristic function, for instance in saying A extends σ. The everywhere undefined function is denoted by $\emptyset$. Two functions are *compatible* if they agree everywhere both are defined. For compatible σ and τ, the smallest partial function extending both is denoted $\sigma \cup \tau$. We use dom(τ) and range(τ) to represent the domain and range of τ respectively.

We say a computation path of an oracle Turing Machine using τ is *defined* if $\tau(x)$ is defined for all queries x along that path. If M is an oracle nondeterministic Turing machine, we say that $M^\tau(x)$ accepts on a path p if all queries to the oracle made along p are in the domain of τ and are answered according to τ, and p ends in an accepting state.

We will sometimes need a machine to know the domain of τ as well as the values of τ on its domain. For these machines we will define the total function $\overline{\tau} : \Sigma^* \times \{0,1\} \to \{0,1\}$ as follows:

$$\overline{\tau}(x,i) = \begin{cases} 1 & \text{if } x \in \text{dom}(\tau) \text{ and } \tau(x) = i \\ 0 & \text{otherwise} \end{cases}$$

By abuse of terminology we will on occasion use the expression "$f^A(x)$" to refer to one of: (i) the value $f^A(x)$, (ii) the function f^A, or (iii) the computation of a particular machine
g f on input x using oracle A. We will try to make clear which interpretation of "$f^A(x)$"
when it cannot easily be inferred from context.

A 1-1 polynomial-time function f^A is *(left-)invertible relative to* A if there exists a polynomial-time function g^A such that for all $x \in \Sigma^*$, $g^A(f^A(x)) = x$.

For a function f and an oracle A, let $f^{A(-1)}(z)$ be the set of strings x such that $f^A(x) = z$.

Let CNF^A be a relativized version of CNF formulae (see [GJ86]). We will also consider the formulae in a closed form, e.g., instead of a formula looking like $(x \vee y)$ it will look like $\exists x \exists y(x \vee y)$. This will allow us to talk about "true" and "false" formulae and make it easier to combine formulae with other expressions. Because we only talk about $\mathbf{NP}^A$-completeness, we only allow "$\exists$" as a quantifier. SAT^A consists of the true formulae relative to the oracle A.

Using standard encoding tricks and simple modifications of Cook's theorem [Coo71] and Berman and Hartmanis [BH77], we get that the following properties of CNF^A and SAT^A hold for all oracles A:

1. For every nondeterministic oracle Turing machine M that runs in time $O(n^i)$ there exists a polynomial-time unrelativized function f such that
   (a) f reduces $L(M^A)$ to SAT^A, and
   (b) for all x, $|f(x)| = O(|x|^{2i})$.
2. Every formula $\varphi \in$ CNF^A has a representation as a binary string. Every binary string represents a formula in CNF^A.
3. Every formula represented by a binary string of n bits can only depend on A on strings of length shorter than n.
4. There is an unrelativized polynomial-time padding function p such that for all formulae φ and strings z,
   (a) $P(\varphi, z)$ is true if and only if φ is true,
   (b) $|P(\varphi, z)| > \max(|\varphi|, |z|)$, and
   (c) from $P(\varphi, z)$ we can in unrelativized polynomial time recover φ and z.

Berman and Hartmanis [BH77] observed that for any languages B and L such that B is $\mathbf{NP}^A$-complete and has such a padding function and L in $\mathbf{NP}^A$, there is a one-to-one length-increasing invertible reduction from L to B.

Let $i = \langle i_0, i_1 \rangle$ using the standard pairing function. Let $f_0, \ldots$ be an enumeration of functions where f_i simulates the deterministic oracle Turing machine with code i_0 running in time n^i. Let $M_0, \ldots$ be an enumeration of nondeterministic oracle machines where M_i simulates the Turing machine with code i_1 running in time n^i.

We use **FP** to represent the class of polynomial-time computable functions.

3 Symmetric Perfect Generic Sets

Definition 3.1 *A sequence $\langle a_i \rangle_{i \in \mathbf{N}}$ of integers form an* iterated-polynomial *sequence if there exists a polynomial p such that $p(n) \geq n^2$ for all n, $a_0 \geq 2$, and $a_{i+1} = p(a_i)$ for all i.*

Definition 3.2 *A partial characteristic function $\tau : \Sigma^* \to \{0,1\}$ is a* symmetric perfect forcing condition *if there is a iterated-polynomial sequence $\langle a_i \rangle_{i \in \mathbf{N}}$ such that*

$$(\bigcup_{i \in \mathbf{N}} \Sigma^{a_i}) \cap \mathrm{dom}(\tau) = \emptyset$$

In other words $\tau(x)$ is undefined for all x such that $|x| = a_i$ for some $i \in \mathbf{N}$. Note that $\tau(x)$ may be undefined on other x as well.

We generally refer to symmetric perfect forcing conditions as *sp-conditions*. As opposed to most types of forcing conditions, sp-conditions cannot necessarily be coded into finite objects.

The name *symmetric perfect* is intended to describe the topological structure of the conditions, and to honor our intellectual debts.

Topologically, we can view a symmetric perfect condition τ as a complete binary tree, the branchings of which correspond to points x at which $\tau(x)$ is undefined. The paths of a complete binary tree form a closed set without isolated points in their natural topology, i.e., they are *perfect*.

From a scholarly point of view, our *symmetric perfect* conditions are special cases of Gerald Sacks' *pointed perfect* conditions [Sac71]. The unique contribution of Sacks was to recognize that forcing conditions need not be recursive (as they are in the standard finite extension arguments or in the recursion theoretic minimal degree construction). Rather, it is sufficient that τ be recursive in each of its members. This is his notion of *pointedness*. Our conditions are pointed, because they can be conceived of as complete binary tree which has been pruned at a coinfinite recursive set of points. This pruning is *symmetric*, in that we either remove all of the left branchings at x (by setting $\tau(x) = 1$) or we remove all of the right branchings at x (by setting $\tau(x) = 0$).

Definition 3.3 *A set S of symmetric perfect forcing conditions is* dense *if every sp-condition τ, there exists an sp-condition σ in S such that σ extends τ.*

Definition 3.4 *A language A is* symmetric perfect generic *(sp-generic) if for every definable dense set S of sp-conditions, there is a $\sigma \in S$ extended by A.*

By *definable* we mean the set $\{\overline{\sigma} \mid \sigma \in S\}$ is a Π^1_1 class (see [Rog87]).

The following theorem is a simple adaptation of the Baire Category Theorem.

Theorem 3.1 *Every sp-condition τ is extended by an sp-generic language A.*

Proof: Let $D_1, \ldots$ be an enumeration of the definable dense sets. Let $\sigma_0 = \tau$. For every $i > 0$, let $\sigma_i = \sigma$ for some $\sigma \in D_i$ such that σ extends σ_{i-1}. For all $x \in \Sigma^*$, let $A(x) = \lim_{i \to \infty} \overline{\sigma}_i(x, 1)$. □

Definition 3.5 *A proposition $P(A)$ is said to be* forced *by an sp-condition τ if $P(A)$ is true for all oracles A extending τ.*

Note that this definition is simpler but different from the usual definition of forcing on generic sets.

If $P(A)$ is a first order proposition in A then the set S of conditions that force $P(A)$ is definable since $\sigma \in S$ if and only if for all A extending σ, $P(A)$ holds.

We can already see the power of sp-generic sets by the following lemma:

Lemma 3.2 *Given any sp-condition τ and any language X there is an sp-condition σ extending τ such that σ forces $X \in \mathrm{P}^A$.*

Proof: Let $\langle a_i \rangle_{i \in \mathbf{N}}$ be the iterated-polynomial sequence such that τ is undefined on strings of length $\langle a_i \rangle_{i \in \mathbf{N}}$. For each i, let $b_i = a_{2i}$ and $d_i = a_{2i+1}$. Let $f(x) = x01^j$ where j is the smallest value such that $|x01^j| = d_i$ for some i. Clearly j is bounded by a polynomial in $|x|$, f is 1-1 and $\mathrm{range}(f) \cap \mathrm{dom}(\tau) = \emptyset$. Define $\sigma(y)$ as

$$\sigma(y) = \begin{cases} \tau(y) & \text{if } y \in \mathrm{dom}(\tau) \\ 1 & \text{if } y = f(x) \text{ and } x \in X \\ 0 & \text{if } y = f(x) \text{ and } x \notin X \\ \text{undefined} & \text{otherwise} \end{cases}$$

Thus for any A extending σ, $x \in X$ if and only if $f(x) \in A$. The partial function σ is undefined on strings of length $\langle b_i \rangle_{i \in \mathbf{N}}$ so σ is an sp-condition. □

Of course, Lemma 3.2 does not imply that there is an sp-generic set G such that for every set X, X is polynomial-time Turing reducible to G. For example an sp-generic set G cannot be reducible to the halting problem relative to G. Lemma 3.2 only implies that all X such that the predicate "$X \in \mathrm{P}^A$" is first-order definable are encoded into all sp-generics.

4 P = FewP Relative to sp-Generics

In this section we will show that relative to sp-generics, acceptance of nondeterministic machines with a small number of accepting paths can be decided in polynomial time.

Theorem 4.1 *If A is an sp-generic oracle then $\mathbf{P}^A = \mathbf{FewP}^A$.*

This proof will build on ideas from Blum and Impagliazzo [BI87], Hartmanis and Hemachandra [HH91] and Rackoff [Rac82].

An immediate corollary is:

Corollary 4.2 *For any sp-generic oracle A, $\mathbf{P}^A = \mathbf{UP}^A$.*

Let R_i be the requirement: "Either there is some input x such that $M_i^A(x)$ has more than n^i accepting paths, or $L(M_i^A) \in \mathbf{P}^A$."

By our enumeration of Turing machines at the end of Section 2, if A satisfies R_i for all i, then $\mathbf{P}^A = \mathbf{FewP}^A$.

Fix i. The set of sp-conditions that force R_i is definable since R_i is a first-order proposition in A. We will show that the set of sp-conditions that force R_i is dense. Then any sp-generic A will extend a τ such that τ forces R_i. We will show these sets are dense by showing how to extend any sp-condition τ to another condition σ such that σ forces R_i.

Let $M = M_i$ and let τ be an sp-condition. Suppose τ does not force "For all x, $M_i^A(x)$ has at most $|x|^i$ accepting paths." For some A extending τ and some x we will have that $M^A(x)$ has more than $|x|^i$ accepting paths. Let $\sigma = \tau \cup (A$ restricted to strings of length at most $|x|^i)$. Clearly σ extends τ and forces "for some x, $M^A(x)$ has more than $|x|^i$ accepting paths." To see that σ is an sp-condition pick a c such that $a_c > |x|^i$ and let $b_j = a_{c+j}$ for all $j \in \mathbf{N}$.

For the rest of this section we will assume τ forces "For all x, $M^A(x)$ has at most $|x|^i$ accepting paths."

By Lemma 3.2 there is an sp-condition σ extending τ such that σ forces $\mathrm{SAT}^{\overline{\tau}} \in \mathbf{P}^A$.

Suppose A extends σ. We will show that $L(M^A) \in \mathbf{P}^A$.

Consider the following algorithm for computing $M^A(x)$ using A as an oracle. The idea is the same as that used in [BI87]. We repeatedly look for *some* extension α of the partial oracle (not necessarily compatible with A) which makes M have the maximum possible number of accepting paths. To ensure consistency with A, we then answer all queries in the domain of α according to A.

In the algorithm below, we maintain the following invariants for all j:

- A extends γ_j,
- γ_{j+1} extends γ_j,
- $|\gamma_{j+1}| \leq |\gamma_j| + n^{i+1}$, and
- $\mathrm{dom}(\gamma_j) \cap \mathrm{dom}(\tau) = \emptyset$ (This fact is not crucial for the proof).

```
BEGIN  ALGORITHM
  $\gamma_0 \leftarrow \emptyset$.
  FOR $j \leftarrow 0$  TO $|x|^{2i} - 1$  DO
    Let $n$ be the largest number for which there is an $\alpha$ extending $\gamma_j$ such that
      • $\alpha$ is compatible with $\tau$ and
      • $M^{\alpha \cup \tau}(x)$ has at least $n$ distinct accepting paths.
    Choose some $\alpha$ that satisfies these two conditions with minimal domain, meaning that dom($\alpha$)
      contains only those queries made along $n$ distinct accepting paths which are not in dom($\tau$).
      If $n = 0$, then $\alpha = \emptyset$.
    $\gamma_{j+1} \leftarrow$ ($A$ restricted to dom($\alpha$)).
    /* This trick is borrowed from [BI87]. It will be explained later. */
  ENDFOR
  $\gamma \leftarrow \gamma_{|x|^{2i}}$.      /* Note that $A$ extends $\gamma$. */
  IF $M^{\gamma \cup \tau}(x)$ has an accepting path
    THEN accept
    ELSE reject.
END  ALGORITHM.
```

Theorem 4.1 now follows from the following two lemmas.

Lemma 4.3 *The above algorithm runs in polynomial time relative to* $\mathrm{SAT}^{\overline{\tau}}$ *and thus relative to* A.

Proof: We show that there is a fixed polynomial bound on both the size of γ_j and the running time of the jth iteration of the FOR loop for all $j < |x|^{2i}$. Assume, inside the jth iteration of the FOR loop, that γ_j has polynomial size. By our assumptions about the behavior of machine M on oracles extending τ, we have $0 \leq n \leq |x|^i$. For any such n, the question—given γ_j—of whether there exists an α extending γ_j compatible with τ such that $M^{\alpha \cup \tau}(x)$ has at least n accepting paths is an $\mathbf{NP}^{\overline{\tau}}$ question and hence can be answered by a single query to $\mathrm{SAT}^{\overline{\tau}}$ (such an α can always be chosen to have polynomial size: only include oracle queries not already in $\mathrm{dom}(\tau)$ made along n distinct accepting paths). Thus n can be determined using polynomially many queries to $\mathrm{SAT}^{\overline{\tau}}$. Once n is found, a polynomial-size α causing $M^{\alpha \cup \tau}(x)$ to accept on n distinct paths can be constructed bit by bit in a straightforward way by making $\mathbf{NP}^{\overline{\tau}}$ queries of the form, "given a sequence $\vec{v}$ of k bits, is there such an α whose first $k+1$ bits are $\vec{v}0$?" Similarly, we can construct the n paths. Once such an α is found, $\mathrm{dom}(\alpha)$ can be made to be minimal simply by eliminating any queries in $\mathrm{dom}(\alpha) - \mathrm{dom}(\gamma_i)$ not made along any accepting path of $M^{\alpha \cup \tau}(x)$, thus we can find a minimal α with at most polynomially many additional $\mathbf{NP}^{\overline{\tau}}$ queries. The size of $\mathrm{dom}(\alpha) - \mathrm{dom}(\gamma_j)$ is at most a polynomial in $|x|$ independent of j, so we can compute γ_{j+1} by asking polynomially many queries to A, and its size is the same as that of α. We thus have that for *all* $j \leq |x|^{2i}$, the size of γ_j and the running time of the jth iteration of the FOR loop are both bounded by a fixed polynomial in $|x|$, and thus the entire FOR loop runs in polynomial time, and $\gamma_{|x|^{2i}}$ has size polynomial in $|x|$.

Since after the FOR loop, γ has polynomial size, we can determine whether $M^{\gamma \cup \tau}(x)$ has an accepting path by asking one additional $\mathbf{NP}^{\overline{\tau}}$ question. Thus, the entire algorithm runs in polynomial time relative to A, which proves Lemma 4.3. □

Lemma 4.4 *The above algorithm correctly decides $M^A(x)$.*

Proof: Suppose $M^A(x)$ has exactly k accepting paths. Let β be the partial function of minimal domain such that

- A extends β, and
- $M^{\beta \cup \tau}(x)$ has k distinct accepting paths.

Since $k \leq |x|^i$ and each path of $M^A(x)$ can make only $|x|^i$ queries, the size of $\mathrm{dom}(\beta)$ is at most $|x|^{2i}$ (if $k = 0$, then $\beta = \emptyset$).

Claim 4.5 *After the FOR loop, $\gamma_{|x|^{2i}} = \gamma$ extends β.*

Lemma 4.4 immediately follows from Claim 4.5 and the fact that A extends γ. Indeed, since $M^{\beta \cup \tau}(x)$ and $M^A(x)$ have the same number of accepting paths, we know that $M^{\gamma \cup \tau}(x)$ and $M^A(x)$ have the same number of accepting paths, because extending a partial oracle can never decrease the number of accepting paths. Thus we accept if and only if $M^A(x)$ has at least one accepting path.

It remains only to prove Claim 4.5. This is similar to the incompatibility argument in [BI87]. It suffices to show that $\mathrm{dom}(\beta) \subseteq \mathrm{dom}(\gamma)$, since both β and γ are compatible with A. Suppose that for some $j < |x|^{2i}$ we have $||\mathrm{dom}(\beta) - \mathrm{dom}(\gamma_j)|| = \ell > 0$. If the α chosen in the jth iteration of the FOR loop does not extend β, then it must be incompatible with β, otherwise the union $\beta \cup \alpha$ would cause $M^{\beta \cup \alpha \cup \tau}(x)$ to have at least one more accepting path than $M^{\alpha \cup \tau}(x)$ (the extra path is "contributed" by β). This contradicts the fact that α was chosen to allow the maximum possible number of accepting paths of $M^{\alpha \cup \tau}(x)$. Hence β and γ_{j+1} share at least one additional point in their domains, so $||\mathrm{dom}(\beta) - \mathrm{dom}(\gamma_{j+1})|| \leq \ell - 1$. Since $||\mathrm{dom}(\beta) - \mathrm{dom}(\gamma_0)|| \leq |x|^{2i}$, we must have $||\mathrm{dom}(\beta) - \mathrm{dom}(\gamma)|| = 0$, which proves the claim. □

5 The Isomorphism Conjecture

5.1 Intuition

In this section we give some of the ideas of the proof that the isomorphism conjecture holds relative to sp-generic oracles. A full and complete proof is presented beginning in Section 5.2.

We first consider how researchers created oracles for which the isomorphism conjecture fails. Typically, they would create a hard function f^A and an oracle A such that $f^A(\mathrm{SAT}^A)$ is **NP**-complete, but not isomorphic to SAT^A. One approach is to have f^A scramble SAT in a way that no reduction to $f^A(\mathrm{SAT}^A)$ could be invertible. Kurtz, Mahaney and Royer used this approach to show that the isomorphism conjecture fails to a random oracle [KMR89]. However since we know that $\mathbf{P}^A = \mathbf{UP}^A$ for sp-generic oracles A (Corollary 4.2), any such scrambling function can be unscrambled.

When Kurtz showed that the isomorphism conjecture fails for regular generic oracles [Kur88] and Hartmanis and Hemachandra created an oracle A relative to which the isomorphism conjecture fails while $\mathbf{P}^A = \mathbf{UP}^A \neq \mathbf{NP}^A$ [HH91] they had to use a different approach. They created functions f^A that work as follows: For φ a boolean formula represented as a string, define $\xi^A(\varphi)$ by

$$\xi^A(\varphi) = A(\varphi 01)A(\varphi 011)\ldots A(\varphi 01^n)$$

where n is the number of variables in φ. Let θ be a small true instance of SAT^A and define f^A by

$$f^A(\varphi) = \begin{cases} \theta & \text{if } \xi^A(\varphi) \text{ is a satisfying assignment of } \varphi \\ \varphi & \text{otherwise} \end{cases}$$

Note for any oracle A and this kind of f^A, $f^A(\mathrm{SAT}^A)$ is $\mathbf{NP}^A$-complete. Using an A that encodes solutions to SAT^A, Kurtz and Hartmanis and Hemachandra show that $f^A(\mathrm{SAT}^A)$ contains large gaps and for reasons of density alone cannot be isomorphic to SAT^A.

In order to give some intuition to how we prove our main result, we will describe how for sp-generic sets A, $f^A(\mathrm{SAT}^A)$ must be isomorphic to SAT^A.

The oracles designed by Kurtz and Hartmanis and Hemachandra that prevent isomorphisms to SAT^A work by having $\xi^A(\varphi)$ be a satisfying assignment to φ. Since we are trying to create an oracle A such that the isomorphism conjecture holds, we will call the computation $f^A(\varphi)$ *bad* if $\xi^A(\varphi)$ is a satisfying assignment to φ and all other computations $f^A(\varphi)$ we will call *good.* Note that if $f^A(\varphi)$ is good then $f^A(\varphi) = \varphi$. Whether $f^A(\varphi)$ is good will, of course, depend on A.

Berman and Hartmanis [BH77] show that in order to have SAT^A isomorphic to $f^A(\mathrm{SAT}^A)$ we need only find a polynomial-time 1-1 length-increasing invertible function g^A that reduces SAT^A to $f^A(\mathrm{SAT}^A)$. Our $g^A(\varphi)$ will work as follows: Find a formula ψ such that

1. $|\psi| > |\varphi|$
2. ψ is true relative to A iff φ is true relative to A
3. $f^A(\psi)$ is good.

Then $g^A(\varphi) = f^A(\psi) = \psi$ is our reduction. The trick is for $g^A(\varphi)$ to find such a ψ.

We use a straight-forward combinatorial argument to show that there exists an invertible polynomial-time function $h(\varphi, w)$ such that

1. For all w, $|h(\varphi, w)| > |\varphi|$
2. For all w and sp-generic A, $h(\varphi, w)$ is true relative to A iff φ is true relative to A
3. For all sp-generic A there exists a w such that $f^A(h(\varphi, w))$ is good.

Now all g^A has to do is find a w such that $f^A(h(\varphi, w))$ is good. We will use f^A to help g^A in this task.

Let $s(\varphi)$ be the formula that encodes the **NP** statement: "φ is true and there exists a w such that $f^A(h(\varphi, w))$ is good." Clearly for all A, $|s(\varphi)| > |\varphi|$ and $s(\varphi)$ is true iff φ is true because there always is a w such that $f^A(h(\varphi, w))$ is good.

We now create $g^A(\varphi)$ as follows: Look at the computation of $f^A(s(\varphi))$. If $f^A(s(\varphi))$ is good then output $f^A(s(\varphi)) = s(\varphi)$. Otherwise $\xi^A(s(\varphi))$ is a satisfying assignment to $s(\varphi)$ and thus from $\xi^A(s(\varphi))$ we can obtain a w such that $f^A(h(\varphi, w))$ is good. The function g^A then outputs $f^A(h(\varphi, w)) = h(\varphi, w)$ for that w. Notice that g^A is not only length-increasing but also 1-1 and invertible.

Of course there is no *a priori* reason that a general reduction has to act like f^A. We will however force a general f^A to look similar to the f^A described above or not be a reduction.

Suppose f^A reduces SAT^A to $L(M^A)$ where f^A is an arbitrary deterministic function running in time n^i and M^A is a nondeterministic Turing machine also running in time n^i. Let us define $h(\varphi, w)$ to be a formula that encodes the following:

$$(\varphi \wedge \exists y \langle \varphi, w, y, 1\rangle \in A) \vee \exists y \langle \varphi, y, 0\rangle \in A$$

where the y's are quantified over strings of length exactly $|\varphi|^i$. If we put in at least one string of the form $\langle \varphi, w, y, 1\rangle$ into A and no strings of the form $\langle \varphi, y, 0\rangle$ then φ is true if and only if $h(\varphi, w)$ is true.

We now need a notion of goodness for $f^A(h(\varphi, w))$ for arbitrary f^A. We would like to call $f^A(h(\varphi, w))$ good if $f^A(h(\varphi, w))$ fails to find a satisfying assignment to $h(\varphi, w)$. However such a thing could be hard to verify. We could, however, determine which queries to A are made by $f^A(h(\varphi, w))$. Thus we call $f^A(h(\varphi, w))$ good if $f^A(h(\varphi, w))$ does not query any string $\langle \varphi, w, y, 1\rangle$ such that $\langle \varphi, w, y, 1\rangle \in A$, i.e., $f^A(h(\varphi, w))$ does not find this part of a satisfying assignment to $h(\varphi, w)$. If $f^A(h(\varphi, w))$ is good then we can alter the truth value of $h(\varphi, w)$ without affecting the value $f^A(h(\varphi, w))$.

Suppose $f^A(h(\varphi, w))$ is good and $q = |f^A(h(\varphi, w))| \leq |\varphi|$. Then $M^A(q)$ cannot ask questions of the form $\langle \varphi, w, y, 1\rangle$ or $\langle \varphi, y, 0\rangle$ because they are too long. We can prevent f^A from being a reduction by setting $h(\varphi, w)$ to true if $M^A(q)$ rejects or setting $h(\varphi, w)$ to false if $M^A(q)$ accepts.

Suppose $f^A(h(\psi, w_1)) = f^A(h(\theta, w_2))$ and neither of these computations ask questions about whether $\langle \psi, w_1, y, 1\rangle$, $\langle \psi, y, 0\rangle$, $\langle \theta, w_2, y, 1\rangle$ or $\langle \theta, y, 0\rangle$ are in A. Then we can prevent f^A from being a reduction by setting $h(\psi, w_1)$ to true and $h(\theta, w_2)$ to false.

We can combine the above techniques under the auspices of sp-generics to produce a reduction g that is length-increasing and almost 1-1. Using the fact that $\mathbf{P}^A = \mathbf{FewP}^A$ for sp-generic A and applying this construction twice we can produce a 1-1 length-increasing reduction g. Grollmann and Selman [GS88] show that we get g invertible for free since $\mathbf{P}^A = \mathbf{UP}^A$ for sp-generic A.

5.2 Proof of the Relativized Isomorphism Conjecture

In order to formally prove Theorem 1.1, we need the following technical lemma, whose proof we defer to Section 5.3.

Lemma 5.1 *Let A be an sp-generic set, M^A a relativized nondeterministic polynomial-time Turing machine and f^A a relativized polynomial time reduction from* SAT^A *to $L(M^A)$. There is a polynomial-time function g^A and a polynomial $p(n)$ such that*

1. *g^A reduces SAT^A to $L(M^A)$*
2. *g^A is length increasing*
3. *for all $q \in \Sigma^*$, if $||g^{A(-1)}(q)|| > 1$ then*
   - *(a) $||f^{A(-1)}(q)|| > 1$*
   - *(b) q is in $L(M^A)$*
   - *(c) $||g^{A(-1)}(q)|| \leq p(|q|)$.*

Proof of Theorem 1.1 (assuming Lemma 5.1): Let L be $\mathbf{NP}^A$-complete. There must exist an nondeterministic polynomial-time machine M and a polynomial-time function f such that $L = L(M^A)$ and f^A reduces SAT^A to L. Apply Lemma 5.1 and let g^A be the function that fulfills the properties of this lemma.

Let $T = \{q \mid ||g^{A(-1)}(q)|| > 1\}$. Note that T is in $\mathbf{FewP}^A$ because of 2 and 3(c) and thus T is in $\mathbf{P}^A$ since $\mathbf{P}^A = \mathbf{FewP}^A$ relative to sp-generic oracles (Theorem 4.1).

Let θ be a fixed member of SAT^A such as $(\exists x)x \vee \overline{x}$. Define $\hat{f}^A(\varphi)$ as follows:

$$\hat{f}^A(\varphi) = \begin{cases} g^A(\theta) & \text{if } g^A(\varphi) \in T \\ g^A(\varphi) & \text{otherwise} \end{cases}$$

Note that $\hat{f}^A$ is a reduction from SAT^A to L because of 3(b).

Apply Lemma 5.1 this time to $\hat{f}^A$ and let $\hat{g}^A$ be the resulting function. Note that the only possible q such that $||\hat{g}^{A(-1)}(q)|| > 1$ is $q = g^A(\theta)$ because of 3(a).

Let $G^A(\varphi) = \hat{g}^A(P(\varphi, q))$ where p is the padding function for SAT^A. Berman and Hartmanis [BH77] show that the claim below immediately implies that SAT^A is p^A-isomorphic to L. □

Claim 5.2 *The function G^A is a 1-1 length increasing reduction from SAT^A to L whose inverse is computable in* $\mathbf{FP}^A$.

Clearly G^A is a reduction. Since $\hat{g}^A$ and p are length increasing then G^A is length increasing. Also G^A is 1-1: Suppose $G^A(\varphi) = G^A(\psi)$ then $\hat{g}^A(P(\varphi, q)) = \hat{g}^A(P(\psi, q)) = q$ but this contradicts the fact that $\hat{g}^A$ is length increasing.

By Corollary 4.2, we know that $\mathbf{P}^A = \mathbf{UP}^A$. Grollmann and Selman [GS88] show that $\mathbf{P}^A = \mathbf{UP}^A$ implies that all 1-1 length-increasing polynomial-time functions relative to A are invertible relative to A. This proves the claim. □

5.3 Proof of Lemma 5.1

We define requirement R_i as follows: "Lemma 5.1 holds for $f^A = f_i^A$ and $M^A = M_i^A$." Note that by the definitions of f_i and M_i in Section 2, all pairs of reductions and machines will be covered by some R_i.

Fix i. Let S_i be the set of sp-conditions that force R_i. Since Lemma 5.1 is first-order definable in A, we have that S_i is a definable set of conditions. We need now show that S_i is dense. Then any sp-generic A will extend a τ such that τ forces R_i. We will show S_i is dense by showing how to extend any sp-condition τ to another condition σ such that σ forces R_i.

Fix i and let τ be an sp-condition and $\langle a_j \rangle_{j \in \mathbf{N}}$ be the corresponding iterated-polynomial sequence. We will create an sp-condition σ with corresponding sequence $\langle b_j \rangle_{j \in \mathbf{N}}$ that forces R_i. Let $f = f_i$ and $M = M_i$.

Suppose τ does not force "f^A reduces SAT^A to $L(M^A)$." For some A extending τ and some φ, we will have that either φ is true and $f^A(\varphi) \notin L(M^A)$ or φ is false and $f^A(\varphi) \in L(M^A)$. Let $m = \max(|\varphi|^i, |f^A(\varphi)|^i)$. Let $\sigma = \tau \cup (A$ restricted to strings of length at most $m)$. Clearly σ extends τ and forces "f^A does not reduce SAT^A to $L(M^A)$," and thus forces R_i. To see that σ is an sp-condition pick a c such that $a_c > m$ and let $b_j = a_{c+j}$ for all $j \in \mathbf{N}$.

For the remainder of this proof we will assume that τ forces "f^A reduces SAT^A to $L(M^A)$."

Pick an e such that $a_{j+e} > a_j^{3i}$ for all j. Since $p(n) \geq n^2$ for all n by Definition 3.1, any $e > \log_2(3i)$ will suffice. Pick a c such that a_c is sufficiently large to avoid all the degenerate cases in this proof. For all j, let $b_j = a_{c+2ej}$ and $d_j = a_{c+2ej+e}$. This proof will never do any encoding on strings of length b_j, guaranteeing that σ is an sp-condition. In fact all of the interesting coding for σ will occur for strings of length d_j. Initially, set $\sigma = \tau$ and also define $\sigma(x) = 0$ for every $x \notin \mathrm{dom}(\tau)$ such that x does not have length b_j or d_j for some j.

Let φ be an arbitrary CNF^A formula. Pick the smallest j such that $d_j > 4|\varphi|^i$. We will define special tupling functions $\langle \varphi, y, 0 \rangle$, $\langle \varphi, w, y, 1 \rangle$ and $\langle \varphi, w, y, 2 \rangle$ where we are only interested in w and y as they range over strings of length $\lfloor d_j/4 \rfloor$. We design these tupling functions so that they have disjoint ranges over strings of length exactly d_j. Since $d_j/4$ is at least $|\varphi|$, $|y|$ and $|w|$, such an encoding is not hard to achieve.

Let $h(\varphi, w)$ be the formula that encodes:

$$(\varphi \wedge \exists y \langle \varphi, w, y, 1 \rangle \in A) \vee \exists y \langle \varphi, y, 0 \rangle \in A$$

In other words, create a nondeterministic oracle Turing Machine M such that M^A accepts if this expression is true and apply the relativized version of Cook's Theorem mentioned in Section 2. We will have $|h(\varphi, w)| = O(d_j^2)$.

Define $\mathbf{f}^A(h(\varphi, w))$ as follows: Simulate $f^A(h(\varphi, w))$. Whenever $f^A(h(\varphi, w))$ queries a string of the form $\langle \varphi', w', z, 1 \rangle$, $\mathbf{f}^A$ will query $\langle \varphi', w', z, 2 \rangle$.

We say the computation $\mathbf{f}^A(h(\varphi, w))$ is *good* if, for all z, if $\mathbf{f}^A(h(\varphi, w))$ queries $\langle \varphi, w, z, 2 \rangle$ then $\langle \varphi, w, z, 2 \rangle \notin A$. Note that whether $\mathbf{f}^A(h(\varphi, w))$ is good does not depend on whether any string of the form $\langle \varphi', w', z, 1 \rangle$ is in A.

Let $r(\varphi)$ be the formula that encodes:

$$\exists w[(\exists y \langle \varphi, w, y, 1 \rangle \in A) \text{ and } \mathbf{f}^A(h(\varphi, w)) \text{ is good}].$$

Let $s(\varphi)$ be the formula that encodes:

$$(\varphi \wedge r(\varphi)) \vee \exists y \langle \varphi, y, 0\rangle \in A.$$

By suitable padding in Cook's theorem [Coo71], we can construct s and h such that each is 1-1 and $\mathrm{range}(h) \cap \mathrm{range}(s) = \emptyset$. Note that h, r and s can be computed in unrelativized polynomial time.

Lemma 5.3 *There is a way to set σ on the strings of length $\langle d_j\rangle_{j\in\mathbf{N}}$ such that*

1. *σ forces "For every formula φ, $r(\varphi)$ is true."*
2. *For every φ and w there is exactly one y such that $\sigma(\langle \varphi, w, y, 1\rangle) = 1$.*
3. *For all φ and y, $\sigma(\langle \varphi, y, 0\rangle) = 0$.*
4. *For all φ, y and w, $\sigma(\langle \varphi, w, y, 1\rangle) = \sigma(\langle \varphi, w, y, 2\rangle)$.*

This is a combinatorial lemma that follows mainly because there are many more ways to set σ than there are extensions to σ that $\mathbf{f}^A(h(\varphi, w))$ could query. We will give a complete proof of Lemma 5.3 in Section 5.4.

Note that $\mathbf{f}^A(h(\varphi, w)) = f^A(h(\varphi, w))$ for all w and A where A extends σ. Also note that σ forces "For every φ and w, φ is true if and only if $h(\varphi, w)$ is true if and only if $s(\varphi)$ is true".

We now describe the algorithm for $g^A(\varphi)$:

```
BEGIN ALGORITHM
(1) Simulate $f^A(s(\varphi))$ and let $S$ be the set of $w$ such that $f^A(s(\varphi))$ queried a string of the form
      $\langle \varphi, w, y, 1\rangle$.
(2) If for some $w \in S$, $\mathbf{f}^A(h(\varphi, w))$ is good then output $\mathbf{f}^A(h(\varphi, w))$ for the first such $w$.
(3) Otherwise output $f^A(s(\varphi))$.
END ALGORITHM.
```

Claim 5.4 *For any A extending σ, the function g^A is a reduction from SAT^A to $L(M^A)$.*

Proof: The fact that g^A is a reduction now follows from the construction of g^A and the fact that τ forces f^A to be a reduction. □

We now show that σ forces g^A to fulfill conclusions (2) and (3a-c) of Lemma 5.1. We will show that if there exists an oracle A extending σ such that g^A fails to fulfill these conditions, then there exists an oracle B extending τ such that f^B does not reduce SAT^B to $L(M^B)$. This contradicts the assumption that τ forces f^B to be such a reduction.

We will create a B that disagrees with A only on strings longer than formulas involved in the assumed failure of some part of (2) or (3a-c) for g^A. This will guarantee that the truth values of these formulas will remain unchanged.

First we show that the sp-condition σ forces that g^A is length-increasing thus fulfilling condition (2) of Lemma 5.1.

Suppose by way of contradiction that $|g^A(\varphi)| \leq |\varphi|$. Let $q = g^A(\varphi)$. Note that $M^A(q)$ cannot look at any string of the form $\langle \varphi, y, 0\rangle$, $\langle \varphi, w, y, 1\rangle$ or $\langle \varphi, w, y, 2\rangle$ because they are too long.

We have two cases each with two subcases:

1. $q = \mathbf{f}^A(h(\varphi, w))$ output by the algorithm for $g^A(\varphi)$ in step (2):

   (a) $M^A(q)$ rejects: There must be some y such that $f^A(h(\varphi, w))$ did not query $\langle \varphi, y, 0\rangle$. Then with $B = A \cup \{\langle \varphi, y, 0\rangle\}$, f^B would not be a reduction.

   (b) $M^A(q)$ accepts: By the definition of g^A, $\mathbf{f}^A$ and parts 3 and 4 of Lemma 5.3 we have that $f^A(h(\varphi, w))$ queries $\langle \varphi, w, z, 1\rangle$ only if $\langle \varphi, w, z, 1\rangle \notin A$. Then with B equal to A minus all strings of the form $\langle \varphi, w, z, 1\rangle$ then f^B would not be a reduction.

2. $q = f^A(s(\varphi))$ output by the algorithm for $g^A(\varphi)$ in step (3):

   (a) $M^A(q)$ rejects: There must be some y such that $f^A(s(\varphi))$ did not query $\langle \varphi, y, 0\rangle$. Then with $B = A \cup \{\langle \varphi, y, 0\rangle\}$, f^B would not be a reduction.

   (b) $M^A(q)$ accepts: Let S be the set from the definition of g^A. Let B equal A minus all strings of the form $\langle \varphi, w, z, 1\rangle$ for $w \notin S$. If $r(\varphi)$ is false relative to B then f^B is no longer a reduction since $f^B(s(\varphi)) = f^A(s(\varphi)) = q$, $s(\varphi)$ is false relative to B, and $q \in L(M^B)$. Suppose $r(\varphi)$ is true relative to B. By the definition of $r(\varphi)$ we have that for some $w \in S$, $\mathbf{f}^B(h(\varphi, w))$ is good. Note that the computation of $\mathbf{f}^B(h(\varphi, w))$ is identical to the computation of $\mathbf{f}^A(h(\varphi, w))$. Since $\mathbf{f}^B(h(\varphi, w))$ is good then $\mathbf{f}^A(h(\varphi, w))$ also is good, and so the algorithm for g^A would have output $\mathbf{f}^A(h(\varphi, w))$ in step (2).

Thus g^A is length-increasing.

Suppose for some A extending σ and some q, $||g^{A(-1)}(q)|| > 1$. Clearly by the definition of g^A and the fact that h and s are both 1-1 with range$(h) \cap$ range$(s) = \emptyset$, $||f^{A(-1)}(q)|| > 1$. Thus we have fulfilled condition (3a) of Lemma 5.1. We need to show how to fulfill conditions (3b) and (3c).

Suppose $q \notin L(M^A)$. Let ψ and η be such that $g^A(\psi) = g^A(\eta) = q$. We can assume without loss of generality that $|\psi| \leq |\eta| < |q|$. There must be some y such that neither $g^A(\psi)$ nor $g^A(\eta)$ queries $\langle \eta, y, 0\rangle$. Let $B = A \cup \{\langle \eta, y, 0\rangle\}$. Suppose $g^A(\psi) = f^A(\nu)$ and $g^A(\eta) = f^A(\mu)$ for some formulas μ and ν. Then $f^B(\nu) = f^B(\mu) = q$ but ν is false and μ is true relative to B and thus f^B is not a reduction. Thus we have fulfilled condition (3b) of Lemma 5.1.

We will now show how to fulfill condition (3c) with $p = t + 2$ where t is the running time of g^A. Suppose $q \in L(M^A)$ and $||g^{A(-1)}(q)|| \geq p(|q|) = t(|q|) + 2$. Let ψ be a minimum length formula such that $g^A(\psi) = q$. By the pigeonhole principle there is some formula $\eta \neq \psi$ such that $g^A(\eta) = q$ and $g^A(\psi)$ does not ask any queries of the form $\langle \eta, w, z, 1\rangle$ or $\langle \eta, w, z, 2\rangle$. Suppose $g^A(\psi) = f^A(\nu)$ for some formula ν.

Note that the value $f^A(\nu)$ and the truth value of ν cannot depend on whether strings of the form $\langle \eta, w, z, 1\rangle$ are in A: Since $g^A(\psi)$ simulates $f^A(\nu)$ and $g^A(\psi)$ does not ask any queries of the form $\langle \eta, w, z, 1\rangle$ then $f^A(\nu)$ does not ask any queries of the form $\langle \eta, w, z, 1\rangle$. The truth value of ν can only depend on whether strings of the form $\langle \psi, y, 0\rangle$, $\langle \psi, w, y, 1\rangle$ and $\langle \psi, w, y, 2\rangle$ are in A and the truth value of ψ and whether $\mathbf{f}^A(h(\psi, w))$ is good for some w. None of these depend on a whether strings of the form $\langle \eta, w, z, 1\rangle$ are in A.

We have two cases:

1. $q = g^A(\eta) = \mathbf{f}^A(h(\eta, w))$ output by the algorithm for $g^A(\eta)$ in step (2): By the definition of g^A, $\mathbf{f}^A$, and Lemma 5.3, we have that $f^A(h(\eta, w))$ queries $\langle \eta, w, z, 1\rangle$ only if $\langle \eta, w, z, 1\rangle \notin A$. Thus if we let B equal A minus all strings of the form $\langle \eta, w, z, 1\rangle$ the following four properties hold:

   (a) $h(\eta, w)$ is false relative to B,
   (b) ν is true,
   (c) $f^B(h(\eta, w)) = f^A(h(\eta, w)) = \mathbf{f}^A(h(\eta, w)) = g^A(\eta) = q$, and
   (d) $q = g^A(\psi) = f^A(\nu) = f^B(\nu)$.

   Thus f^B will not be a reduction.

2. $q = f^A(s(\eta))$ output in step (3) of the algorithm: Let S be the set from the definition of g^A. Let B equal A minus all strings of the form $\langle \eta, w, z, 1\rangle$ for $w \notin S$. Note that $f^B(s(\eta)) = f^A(s(\eta))$. Also $r(\eta)$ is false relative to B for the same reasons as in case 2b of the proof of condition (2) above and thus $s(\eta)$ is also false relative to B. Thus

$$f^B(s(\eta)) = f^A(s(\eta)) = g^A(\eta) = q = g^A(\psi) = f^A(\nu) = f^B(\nu)$$

   and ν is true relative to B and thus f^B is not a reduction.

Thus we have fulfilled condition (3c). We have now fulfilled all of the conditions of Lemma 5.1.
□

5.4 Proof of Lemma 5.3

We will use relativized Kolmogorov complexity for this proof. For an excellent background in Kolmogorov complexity see the book by Li and Vitányi [LV93].

We need to find a σ extending τ that fulfills the conditions of Lemma 5.3. Note that by our construction of the d_j sequence, we have that $d_j > d_{j-1}^{9i^2}$. Fix j and let ℓ be the least integer such that $d_{j-1} \leq 4\ell^i$ and u be the greatest integer such that $d_j > 4u^i$. For $j = 0$ let $\ell = 0$. The values ℓ and u bound the lengths of formulae φ such that $\langle \varphi, y, 0\rangle$, $\langle \varphi, w, y, 1\rangle$ and $\langle \varphi, w, y, 2\rangle$ all have length d_j.

Let $z = |w| = |y| = \lfloor d_j/4 \rfloor$. Let $m = z2^z \sum_{\ell \leq e \leq u} 2^e$. Let x be a string of length m such that $K^{\overline{\tau}}(x) \geq m$, i.e., x is Kolmogorov random with respect to $\overline{\tau}$.

View x as a concatenation of strings $x_{\varphi,w}$ of length z where φ ranges over all formulae of length between ℓ and u, and w ranges over all strings of length z. Set $\sigma(\langle \varphi, w, x_{\varphi,w}, 1\rangle) = \sigma(\langle \varphi, w, x_{\varphi,w}, 2\rangle) = 1$ for all φ and w, and set σ to zero for all other strings of length d_j.

Clearly this σ fulfills conditions 2, 3 and 4 of Lemma 5.3. We still need to show that σ forces "For all formula φ, $r(\varphi)$ is true".

Suppose there is some oracle A extending σ such that for some formula φ, $r(\varphi)$ is false relative to A. We will show how to describe x with a string of length much shorter than x contradicting the fact that x is Kolmogorov random.

Recall that $h(\varphi, w)$ has length $O(d_j^2)$. Thus $\mathbf{f}^A(h(\varphi, w))$ has running time at most $O(d_j^{2i})$. Thus $\mathbf{f}^A$ can only depend on the strings in $A' = A^{<b_{j+1}}$.

Initialize B to be A' with all the strings of the form $\langle \varphi, w, y, 2\rangle$ removed.

Create a string v as follows:

1. Initially set $v = \epsilon$.

2. For w ranging over strings of length z do

(a) For $j = 1$ to $|h(\varphi, w)|^i$

If the jth query of $\mathbf{f}^A(h(\varphi, w))$ exists and is a string $\langle \varphi, w', y, 2\rangle \in A' - B$ then mark j and add $\langle \varphi, w', y, 2\rangle$ to B.

(b) Concatenate to v the number of marked j followed by a list of marked j. Write these numbers with leading zeros if necessary to keep the lengths consistent in order to make the encoding simpler.

The orders in which w and j are chosen in this procedure play an important role in allowing us to keep the description of A' small.

At the end of this procedure we will have $B = A'$: Since $r(\varphi)$ is false relative to A then for every w, $\mathbf{f}^A(h(\varphi, w))$ queries the unique string of the form $\langle \varphi, w, y, 2\rangle$ in A (and thus in A'). Thus every such string will be added to B in step w of the above procedure if not before.

The length of v is bounded by $O(2^z \log d_j)$ because there are 2^z strings in $A' - B$ initially. Note that each $\langle \varphi, w, y, 2\rangle \in A'$ can only contribute to one marking.

Now we claim we can construct A' and thus x using an oracle for $\overline{\tau}$ with the following description: $A^{\leq b_j}$, v, φ and x' where x' is the concatenation of $x_{\psi,w}$ for all formulae $\psi \neq \varphi$ of length between ℓ and u, and w ranging over all strings of length z. We can reconstruct A' by repeating the procedure above using v to tell us which queries of the form $\langle \varphi, w, y, 2\rangle$ are in A'.

We can encode the tuple $\langle A^{\leq b_j}, v, \varphi, x'\rangle$ as a string of length $|A^{\leq b_j}| + |v| + |\varphi| + |x'|$ plus an additional $O(d_j)$ bits to encode the length of each piece.

The total length is bounded by $O(d_j) + 2^{b_j+1} + O(2^z \log d_j) + d_j + m - z2^z < m - O(1)$. There might exist a finite number of j such that this inequality fails. We can eliminate this possibility by an appropriate choice of a_c in the beginning of Section 5.3.

We have created a fixed Turing machine that outputs x with oracle $\overline{\tau}$ and input $\langle A^{\leq b_j}, v, \varphi, x'\rangle$, a tuple whose length is strictly less than $|x|$. This contradicts the fact that x was Kolmogorov random relative to $\overline{\tau}$. □

6 Conclusions and Open Questions

Later work by Fenner, Fortnow, Kurtz and Li [FFKL93] show that relative to sp-generics, $\mathbf{P} = \mathbf{BPP} = \mathbf{NP} \cap \text{co-}\mathbf{NP} = \mathbf{SPP}$ but the polynomial-time hierarchy is proper. They also look at notions of genericity in a broader sense and show several interesting oracle results based on these ideas.

We have shown that relative to sp-generic oracles the isomorphism conjecture holds. Several obvious open questions remain:

- Does the isomorphism conjecture hold in the unrelativized world? Despite Theorem 1.1, the authors believe the evidence supports the position that the conjecture does not hold. Theorem 1.1 show that a proof of this result will require nonrelatizing techniques.

- How complicated must an oracle A be such that the isomorphism conjecture holds relative to A? By Lemma 3.2 we can easily see that there are no sp-generic oracles in the arithmetic hierarchy. However we can fulfill just the requirements necessary for the isomorphism conjecture with a set recursive in the halting problem.

Is there a recursive oracle A? One could wonder whether we could recursively fulfill all the necessary requirements. We could use time-bounded Kolmogorov complexity in Section 5.4 but determining whether τ forces f to be a reduction is not decidable. However, we believe that a careful finite injury argument could lead to a recursive oracle.

- Is there an oracle relative to which the isomorphism conjecture is true and $\mathbf{P} \neq \mathbf{UP}$, i.e., there exist one-way functions?
- Is there an oracle relative to which the isomorphism conjecture is true and the polynomial-time hierarchy collapses? A related question is whether there exists an oracle A such that $\mathbf{P}^A = \mathbf{UP}^A$ and $\mathbf{NP}^A = \mathbf{EXP}^A$. This oracle A also would imply that the isomorphism conjecture holds (see [HS92]).

Acknowledgments

This paper grew from discussions at the Dagstuhl workshop on structural complexity theory with many people especially Ted Slaman, Bill Gasarch, Steve Homer and Alan Selman.

We would like to thank other people for helpful discussions on the isomorphism conjecture over time: Steve Mahaney, Jim Royer and Mike Sipser.

We also thank Richard Beigel, Bill Gasarch, Steve Homer and Jim Royer for their comments on earlier drafts of this paper. We would also like to thank the anonymous referee for his or her extensive comments.

And, of course, we would like to acknowledge Juris Hartmanis for bringing this challenging problem to the world.

References

[BH76] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. In *Proceedings of the 8th ACM Symposium on the Theory of Computing*, pages 30–40. ACM, New York, 1976.

[BH77] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 1:305–322, 1977.

[BI87] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 118–126. IEEE, New York, 1987.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158. ACM, New York, 1971.

[FFKL93] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 120–131. IEEE, New York, 1993.

[GJ86] J. Goldsmith and D. Joseph. Three results on the polynomial isomorphism of complete sets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 390–397. IEEE, New York, 1986.

[GS88] J. Grollmann and A Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17:309–355, 1988.

[HH91] J. Hartmanis and L. Hemachandra. One-way functions and the nonisomorphism of NP-complete sets. *Theoretical Computer Science*, 81(1):155–163, 1991.

[HS89] S. Homer and A. Selman. Oracles for structural properties: The isomorphism problem and public-key cryptography. In *Proceedings of the 4th IEEE Structure in Complexity Theory Conference*, pages 3–14. IEEE, New York, 1989.

[HS92] S. Homer and A. Selman. Oracles for structural properties: The isomorphism problem and public-key cryptography. *Journal of Computer and System Sciences*, 44(2):287–301, 1992.

[JY90] D. Joseph and P. Young. Self-reducibility: Effects of internal structure on computational complexity. In A. Selman, editor, *Complexity Theory Retrospective*, pages 82–107. Springer, 1990.

[KMR87] S. Kurtz, S. Mahaney, and J. Royer. Progress on collapsing degrees. In *Proceedings of the 2nd IEEE Structure in Complexity Theory Conference*, pages 126–131. IEEE, New York, 1987.

[KMR88] S. Kurtz, S. Mahaney, and J. Royer. Collapsing degrees. *Journal of Computer and System Sciences*, 37(2):247–268, 1988.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails to a random oracle. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 157–166. ACM, New York, 1989.

[KMR90] S. Kurtz, S. Mahaney, and J. Royer. The structure of complete degrees. In A. Selman, editor, *Complexity Theory Retrospective*, pages 82–107. Springer, 1990.

[Kur88] S. Kurtz. The isomorphism conjecture fails relative to a generic oracle. Technical Report 88-018, Department of Computer Science, University of Chicago, 1988.

[LV93] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Texts and Monographs in Computer Science. Springer, New York, 1993.

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

[Rog87] H. Rogers. *Theory of Recursive Functions and Effective Computability.* MIT Press, Cambridge, Massachusetts, 1987.

[Sac71] G. Sacks. Forcing with perfect closed sets. In *Proceedings of Symposia in Pure Mathematics*, volume XIII, pages 331–356. American Mathematical Society, 1971.