# A note on the Isomorphism Conjecture and one-way functions
*Preliminary*

John D. Rogers*
School of CTI
DePaul University

February 24, 1999

## 1 The Isomorphism Conjecture

What if all **NP**-complete problems are simply variations on the well-known Satisfiability problem (hereafter denoted **SAT**)? As usual, we encode instances of problems as binary strings and call a collection of strings a *language*. For example, the set of strings that represent propositional formulae that are satisfiable is the **NP**-complete language **SAT**. The **NP**-complete language representing the Traveling Salesperson Problem (denoted **TSP**) contains strings encoding a collection of $n$ cities, the costs of traveling between each pair of cities, and a cost $c$ such that there is a tour of the $n$ cities whose cost is no more than $c$.[1]

It is well-known that we can map strings in **SAT** to strings in **TSP**. That is, there is a polynomial-time computable function $f$ from strings to strings that maps strings in **SAT** to strings in **TSP** and strings not in **SAT** to strings not in **TSP**, that is, $x \in \mathbf{SAT}$ iff $f(x) \in \mathbf{TSP}$. Such a mapping is called a reduction. When $f$ is a many-to-one function, it is a *many-one reduction (m-reduction)* and we denote this by $\mathbf{SAT} \leq_m^p \mathbf{TSP}$, where the superscript $p$ stands for polynomial-time computable and the subscript $m$ stands for many-one. There is also an m-reduction from **TSP** to **SAT** so $\mathbf{TSP} \leq_m^p \mathbf{SAT}$. So, in some sense, these two problems are very similar: If we can solve one efficiently, we can solve the other efficiently as well.

But we are interested in a more stringent kind of similarity: What if the **NP**-complete languages are *isomorphic*? An isomorphism between languages is a polynomial-time computable reduction that is one-to-one and *polynomial-time* invertible. A reduction $f$ is polynomial-time invertible if there is a polynomial-time computable function $g$ such that $g(f(x)) = x$.

Berman and Hartmanis [BH77] were the first to consider whether the **NP**-complete languages are isomorphic to one another. They proved the following two theorems:

**Theorem 1.1 ([BH77])** *If two languages $A$ and $B$ are equivalent by polynomial-time reductions that are one-to-one, length-increasing, and polynomial-time invertible then $A$ and $B$ are polynomial-time isomorphic. (The proof is a polynomial-time version of the Cantor-Bernstein proof in set theory.)*

A function $f$ is length-increasing if, for all $x$, the length of $f(x)$ is greater than the length of $x$.

**Theorem 1.2 ([BH77])** *If two languages $A$ and $B$ are equivalent by polynomial-time reductions that are many-to-one and if they are both* paddable *then $A$ and $B$ are equivalent by polynomial-time reductions that are one-to-one, length-increasing, and polynomial-time invertible.*

*Email: `rogers@cs.depaul.edu`

[1] For more on **NP**-complete problems and **NP**-completeness, please see Garey and Johnson's definitive treatment [GJ79].

A language $L$ is paddable if there is a one-to-one, polynomial-time invertible function $p(x, y)$, called the *padding function*, such that, for all $y$, $p(x, y) \in L$ iff $x \in L$. In other words, a padding function is a reduction from a language to itself. Every **NP**-complete language known to date is paddable.

Noting this, Berman and Hartmanis posed:

**The Isomorphism Conjecture (IC)[BH76, BH77]** *Every* **NP**-*complete language is isomorphic to* **SAT**.

If their conjecture is true, $\mathbf{P} \neq \mathbf{NP}$ because finite languages in **P** cannot be isomorphic to infinite languages like **SAT**. Not much is known about what is implied if the **IC** is false. It is possible that it is false and $\mathbf{P} = \mathbf{NP}$ or that it is false and $\mathbf{P} \neq \mathbf{NP}$ or even that it is false and $\mathbf{NP} = \mathbf{EXP}$. The conjecture has been neither proven nor refuted, although much work has gone into both efforts.

In complexity theory, when a proposition refuses to yield its truth value, one approach is to construct oracles relative to which the proposition is true or false. Although this approach is unlikely to provide a direct proof of or counterexample to the proposition, it often provides insight into the structure of a problem and can be used to show that the most commonly used proof techniques will be of little use in settling the matter. For more about oracles and oracle computations, see any introductory text on computational complexity (for example, Hopcroft and Ullman [HU79]). For a more complete explanation of the usefulness of oracle results, please see the excellent paper by Fortnow [For94].

The first oracles to appear in connection with the **IC** were those relative to which it is false, including ones constructed by Kurtz [Kur83]; Kurtz, Mahaney, and Royer [KMR89]; and Hartmanis and Hemachandra [HH91].

Finding an oracle relative to which the **IC** is true was a difficult task finally accomplished by Fenner, Fortnow, and Kurtz [FFK92] in 1992. Their oracle was *sp-generic*, a complicated sort of generic oracle defined using infinite conditions that allowed infinite encoding. The proof that the IC holds relative to this oracle was intricate and technical, relying on the notions of genericity and relativized Kolmogorov complexity. One of the open questions they posed was whether there exists a simpler oracle that would make the **IC** true.

This was settled in 1997 when Beigel, Buhrman, and Fortnow [BBF98] used a straightforward and entirely computable construction to create an oracle relative to which the IC holds. To understand their result, we must first refer to three earlier theorems:

1. Grollmann and Selman [GS88] showed that one-way functions exist iff $\mathbf{P} \neq \mathbf{UP}$.

2. Berman [Ber77] showed that the complete m-degree of **EXP** is a 1-li-degree.

3. Hartmanis and Berman [BH77] showed that every 1-li-degree is an isomorphism degree if one-way functions do not exist.

A *one-way function* is a polynomial-time computable, one-to-one, length increasing function whose inverse is not polynomial-time computable. The class **UP** contains all languages that can be recognized by a nondeterministic, polynomial-time Turing machine that, on all inputs, has at most one accepting path. The class **EXP** contains all languages that can be recognized by a deterministic Turing machine running in time $O(2^{n^k})$. A *degree* is collection of languages that are all reducible to one another. For example, the m-degree containing **SAT** also contains all languages m-reducible to **SAT** and that to which **SAT** is m-reducible. This degree is complete for **NP** because **SAT** is **NP**-complete. The complete 1-li-degree of **EXP** contains all of the languages that are complete for **EXP** under 1-li-reductions. An isomorphism degree is one where the languages are each isomorphic to the other.

Combining the three facts above, Homer and Selman [HS92] were the first to note that if one could create an oracle relative to which $\mathbf{NP} = \mathbf{EXP}$ and $\mathbf{P} = \mathbf{UP}$, the **IC** would be true relative to that oracle. They were able to construct an oracle that makes $\mathbf{NP}^{\mathbf{NP}} = \mathbf{EXP}$ and $\mathbf{P} = \mathbf{UP}$ but could not get **EXP** to collapse all the way to **NP**. But this is exactly what Beigel, Buhrman, and Fortnow did: They created an oracle $A$ that makes $\mathbf{P} = \mathbf{UP}$ (in fact, $\mathbf{P} = \oplus\mathbf{P}$) and $\mathbf{NP} = \mathbf{EXP}$. Please see their paper for details about the construction.

Relative to $A$, $\mathbf{P} = \mathbf{UP}$, so there are no one-way functions. A question raised by Fenner, Fortnow, and Kurtz [FFK92] was whether there was an oracle relative to which the IC holds and one-way functions exist. This was settled in 1995 in [Rog97]. The construction there begins with the oracle in [FFK92], relative to which $\mathbf{P} = \mathbf{UP}$, and re-relativize to a $\mathbf{UP}$-generic oracle. The result is based on the following theorem:

**Theorem 1.3 ([Rog97])** *Assume that* $\mathbf{P} = \mathbf{UP}$. *Then relative to a* $\mathbf{UP}$*-generic oracle* $G$, *there are one-way functions but every paddable 1-li-degree collapses to an isomorphism type.*

A $\mathbf{UP}$-generic oracle is one that is empty except at lengths that are exponentially far apart. At these lengths, the oracle contains at most one string. We will use a slightly different oracle, called a $\mathbf{UP} \cap \mathbf{coUP}$-generic. At exponentially far apart lengths, these have exactly one string and are empty elsewhere. Note that theorem 1.3 still works when $G$ is $\mathbf{UP} \cap \mathbf{coUP}$-generic.

Generic oracles turn out to be enormously useful in a variety of situations. For more about them, see the papers by Blum and Impagliazzo [BI87]; Fenner, Fortnow, and Kurtz [FFK92]; and Fenner, Fortnow, Kurtz, and Li [FFKL93]. For more about $\mathbf{UP}$-generic and $\mathbf{UP} \cap \mathbf{coUP}$-generic oracles, see the paper by Fortnow and Rogers [FR94].

# 2 Results

**Theorem 2.1** *There is an oracle* $C$ *such that:*

1. $\mathbf{NP}^C = \mathbf{EXP}^C$, *so the complete m-degree of* $\mathbf{NP}^C$ *is a 1-li-degree;*
2. $\mathbf{P}^C \neq \mathbf{UP}^C$, *so one-way functions exist;*
3. *the* $\mathbf{IC}^C$ *is true.*

Proof. Let $C$ be the oracle $A \oplus G = \{0x : x \in A\} \cup \{1x : x \in G\}$, where $A$ is the oracle constructed in [BBF98] and $G$ is a $\mathbf{UP}^A \cap \mathbf{coUP}^A$-generic oracle. Because $\mathbf{NP}^A = \mathbf{EXP}^A$, $\mathbf{NP}^C = \mathbf{EXP}^C$. To see this, let $L$ be a language accepted by an $\mathbf{NP}^A$ machine $M$ and let $x$ be a string in $L$. Let $N$ be a machine that first guesses the unique string in $G$ that can affect the computation. If a path's guess does not find the string, the path rejects. If it does, it proceeds to do the computation it would have done relative to $A$ with the additional knowledge of the string in $G$.

A simple diagonalization argument shows that $\mathbf{P}^G \neq \mathbf{UP}^G \cap \mathbf{coUP}^G$, so $\mathbf{P}^C \neq \mathbf{UP}^C \cap \mathbf{coUP}^C$.

Applying theorem 1.3 tells us that the complete 1-li-degree of $\mathbf{NP}^C$ collapses to an isomorphism type. Thus, the $\mathbf{IC}$ is true. ■

This result is a nice contrast to another result in [BBF98]:

**Theorem 2.2 ([BBF98])** *There is an oracle* $B$ *and a language* $L$ *that is 1-tt-complete for* $\mathbf{NP}^B$ *but not m-complete.*

This separates $\mathbf{NP}$ from $\mathbf{EXP}$ relative to $B$ because Homer, Kurtz, and Royer [HKR93] showed that the complete 1-tt-degree of $\mathbf{EXP}$ is an m-degree (and so a 1-li-degree). The proof of theorem 2.2 actually shows that if $\mathbf{P} = \mathbf{PSPACE}$ then, relative to a $\mathbf{UP} \cap \mathbf{coUP}$-generic, $L$ is 1-tt-complete for $\mathbf{NP}$ but not m-complete. Our theorem show that if $\mathbf{P} = \mathbf{UP}$ and $\mathbf{NP} = \mathbf{EXP}$ then, relative to a $\mathbf{UP} \cap \mathbf{coUP}$-generic, all languages 1-tt-complete for $\mathbf{NP}$ are also m-complete.

# References


[BBF98] R. Beigel, H. Buhrman, and L. Fortnow. NP might not be as easy as detecting unique solutions. In *Proceedings of the 30th ACM Symposium on the Theory of Computing*, pages 203–208. ACM, New York, 1998.

[Ber77] L. Berman. *Polynomial reducibilities and complete sets.* PhD thesis, Cornell University, 1977.

[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.

[FFK92] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 30–39. IEEE, New York, 1992.

[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.

[For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the European Association for Theoretical Computer Science*, 52:229–244, February 1994.

[FR94] Lance Fortnow and John D. Rogers. Separability and one-way functions. In *Proceedings of the 5th International Symposium on Algorithms and Computation*, volume 834 of *Lecture Notes in Computer Science*, pages 396–404. Springer, 1994.

[GJ79] Michael Garey and David Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Company, New York, 1979.

[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.

[HKR93] S. Homer, S. Kurtz, and J. Royer. A note on many-one and 1-truth table complete sets. *Theoretical Computer Science*, 115(2):383–389, 1993.

[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.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Mass., 1979.

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

[Kur83] Stuart A. Kurtz. A relativized failure of the Berman-Hartmanis conjecture. Technical Report 83–001, University of Chicago, 1983.

[Rog97] John D. Rogers. The Isomorphism Conjecture holds and one-way functions exist relative to an oracle. *Journal of Computer and System Sciences*, 54(3):412–423, 1997.