SIAM J. COMPUT.
Vol. 7, No. 3, August 1978

ISOMORPHISM TESTING FOR GRAPHS, SEMIGROUPS, AND FINITE AUTOMATA ARE POLYNOMIALLY EQUIVALENT PROBLEMS[*]

KELLOGG S. BOOTH[†]

Abstract. Two problems are polynomially equivalent if each is polynomially reducible to the other. The problems of testing either two graphs, two semigroups, or two finite automata for isomorphism are shown to be polynomially equivalent. For graphs the isomorphism problem may be restricted to regular graphs since we show that this is equivalent to the general case. Using the techniques of Hartmanis and Berman we then show that this equivalence is actually a polynomial isomorphism. It is conjectured that the isomorphism problem for groups is not in this equivalence class, but that it is an easier problem. If the conjecture is true then $P \neq NP$; if it is false then there exists a "subexponential" $O(n^{c_1 \log n + c_2})$ algorithm for graph isomorphism.

Key words. graph, regular graph, group, semigroup, finite automaton, isomorphism, polynomial reduction, NP-complete

1. Introduction. Determining the exact computational complexity of graph isomorphism is currently an open problem [11]. No algorithm has been proven to run in less than exponential time yet no nontrivial lower bound has been proven. Many problems of this type have been shown NP-complete but at present graph isomorphism is not among them [1], [6]. We investigate the class of problems which are "complete" over a graph isomorphism in the sense that any such problem will be polynomially equivalent to the problem of graph isomorphism. The notion of equivalence we use is mutual reducibility in the sense of either Karp or Cook [1], although we also show that the equivalence holds under the stronger notion of polynomial isomorphism discussed by Hartmanis and Berman [4].

After defining these basic concepts we introduce three isomorphism problems: isomorphism of graphs, isomorphism of semigroups, and isomorphism of finite automata. We show that testing graph isomorphism is no harder than testing regular graph isomorphism. We then extend the techniques to prove that the three isomorphism problems are in fact equivalent.

We conclude the discussion with a conjecture relating the problem of group isomorphism to the previous problems and to recent results of Tarjan [12] and Miller [10]. The implications of this conjecture to the $P = NP$ question are examined.

2. Basic terminology. The concept of polynomial reducibility has become familiar within the literature [1], [6]. A problem (language) L_1 is *polynomially reducible* to a problem (language) L_2 if there exists an encoding $\mathscr{E}$, computable in polynomial time, from strings over the alphabet of L_1 to strings over the alphabet of L_2 such that $w \in L_1$ iff $\mathscr{E}(w) \in L_2$. This is written as $L_1 \propto_p L_2$. If $L_1 \propto_p L_2$ and $L_2 \propto_p L_1$ we say that the problems are *polynomially equivalent* and write $L_1 \equiv_p L_2$; if the mapping $\mathscr{E}$ is a bijection and $\mathscr{E}^{-1}$ is a polynomial reduction of L_2 to L_1 we say that the problems are *polynomially isomorphic* and write $L_1 \cong_p L_2$ [4].

A graph $G = (V, E)$ is a set of *vertices* V and *edges* E such that each edge is a pair of vertices. A graph is *regular of degree* r if every vertex belongs to exactly r edges. K_n is the *complete graph* on n vertices, in which every vertex belongs to an edge with every other vertex. $K_{n,n}$ is the *complete bipartite graph* on $2n$ vertices. Two graphs

[*] Received by the editors March 14, 1977, and in revised form October 25, 1977.

[†] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. This research was supported by the National Research Council of Canada under Grant A4307.

$G_1=(V_1, E_1)$ and $G_2=(V_2, E_2)$ are *isomorphic* if there exists a bijection f: $V_1 \to V_2$ such that $(u, v)\in E_1$ iff $(f(u), f(v))\in E_2$. This is written $G_1 \cong G_2$. The *graph isomorphism problem* is to decide, given two graphs G_1 and G_2, if such a bijection exists. Although many algorithms have been proposed [11] there is no known polynomial or even subexponential algorithm for deciding this question.

A semigroup $S=(X, \circ)$ is a set X having a single binary operation such that $x\circ(y\circ z)=(x\circ y)\circ z$ for all $x, y, z \in X$ ($\circ$ is an associative operation). The semigroup is commutative if $x\circ y=y\circ x$ for all $x, y\in X$. Two semigroups $S_1=(X_1, \circ_1)$ and $S_2=(X_2, \circ_2)$ are isomorphic if there exists a bijection f: $X_1\to X_2$ such that $x\circ_1 y=z$ iff $f(x)\circ_2 f(y)=f(z)$: This is written $S_1\cong S_2$. The *semigroup isomorphism problem* is to determine if such a bijection exists. More information on semigroups is given in [3].

A *finite automaton* $A=(Q, \Sigma, \delta, s, F)$ is a set of *states* Q, an *alphabet* Σ, a *transition function* δ: $Q\times\Sigma\to Q$, *initial state* $s\in Q$, and a set of *final states* $F\subseteq Q$. Two finite automata $A_1=(Q_1, \Sigma_1, \delta_1, s_1, F_1)$ and $A_2=(Q_2, \Sigma_2, \delta_2, s_2, F_2)$ are *isomorphic* if there exist bijections f: $Q_1\to Q_2$ and g: $\Sigma_1\to\Sigma_2$ such that

1) $f(\delta_1(q, a))=\delta_2(f(q), g(a))$ for all $q\in Q_1$ and $a\in\Sigma_1$,

2) $f(s_1)=s_2$,

and

3) $q\in F_1$ iff $f(q)\in F_2$ for all $q\in Q_1$.

The *finite automaton isomorphism problem* is to determine if two such bijections exist [8]. More information on finite automata is given in [3].

3. Graphs.

Although some classes of graphs such as trees [1] and planar graphs [5] are easy to test for isomorphism there are many well-known classes of graphs for which the isomorphism problem is no easier than for the most general case [2]. One such class having this property is the class of regular graphs.

THEOREM 1. *Regular graph isomorphism* $\equiv_p$ *graph isomorphism.*

Proof. Any isomorphism test for arbitrary graphs will also work for regular graphs. We need only show that graph isomorphism $\propto_p$ regular graph isomorphism.

Let $G=(V, E)$ be any graph having $V=\{v_i|1\leqq i\leqq n\}$ and $E=\{e_j|1\leqq j\leqq m\}$ where every vertex belongs to at least one edge and $m-n>2$. Define the following sets.

$$V_1=\{f_j|1\leqq j\leqq m\},$$
$$V_2=\{g_k|1\leqq k\leqq m-2\},$$
$$V_3=\{h_l|1\leqq l\leqq m-n+2\}$$

and

$$E_1=\{\{v_i, e_j\}|v_i\in e_j,\ 1\leqq i\leqq n \text{ and } 1\leqq j\leqq m\},$$
$$E_2=\{\{v_i, f_j\}|v_i\notin e_j,\ 1\leqq i\leqq n \text{ and } 1\leqq j\leqq m\},$$
$$E_3=\{\{e_j, g_k\}|1\leqq j\leqq m \text{ and } 1\leqq k\leqq m-2\},$$
$$E_4=\{\{f_j, h_l\}|1\leqq j\leqq m \text{ and } 1\leqq l\leqq m-n+2\}.$$

Let REGULAR(G) be the graph ($V\cup E\cup V_1\cup V_2\cup V_3$, $E_1\cup E_2\cup E_3\cup E_4$). We can establish two facts about REGULAR(G): it is a regular graph of degree m and given REGULAR(G) we can recover G uniquely.

The first fact is easily verified. Each $v_i\in V$ has degree m in REGULAR(G) because it is adjacent to either e_j or f_j for all $1\leqq j\leqq m$; each $e_j\in E$ has degree m because it is adjacent to exactly 2 of the $v_i\in V$ and to all $m-2$ of the $g_k\in V_2$; each $f_j\in V_1$ is adjacent to exactly $n-2$ of the $v_i\in V$ and also to all $m-n+2$ of the $h_l\in V_3$;

each $g_k \in V_2$ is adjacent to all m of the $e_j \in E$; finally each $h_l \in V_3$ is adjacent to all m of the $f_j \in V_1$.

The second fact follows from the observation that in REGULAR(G) every $g_k \in V_2$ has exactly the same set of neighbors and every $h_l \in V_3$ has exactly the same set of neighbors. We can tell these two sets apart because $|V_2| > |V_3|$ since $n > 4$ if $m - n > 2$ in a graph. Having thus located V_2, we know that

$$E = \{\text{vertices at distance 1 from } V_2\},$$

$$V = \{\text{vertices at distance 2 from } V_2\}$$

and also that $\{u, v\} \in E$ iff there is an edge in REGULAR(G) from both u and v to some $e_j \in E$. The encoding $(G_1, G_2) \rightarrow$ (REGULAR(G_1), REGULAR(G_2)) thus has the property that $G_1 \cong G_2$ iff REGULAR(G_1) $\cong$ REGULAR(G_2). Moreover, it is clearly computable in polynomial time and hence is a polynomial reduction of graph isomorphism to regular graph isomorphism if we realize that isolated vertices can be handled with a simple pre-test and that adding an equal number of copies of K_4 to both G_1 and G_2 will not affect their isomorphism but will ensure that $m - n > 2$, without increasing the size of the input by more than a polynomial. Q.E.D.

Similar constructions exist which show that graph isomorphism and directed graph isomorphism are polynomially equivalent [2], [10]. We will thus make no distinction between the two problems (indeed, our definition of graph conveniently glossed over any distinction).

Having seen that for at least one subclass of graphs the isomorphism problem is still equivalent to the general case, we should not be surprised to find many others. A survey of classes having this property is contained within [2]. What is of more interest is to find substantially "different" problems which are still equivalent. The next two sections provide examples of such problems, although both are admittedly still isomorphism problems and thus maybe not too "different."

4. Semigroups. Miller and Monk [9] have shown that the isomorphism problem for any algebraic structure is polynomially reducible to graph isomorphism because the operation tables can be encoded as directed graphs. Conversely, if we have an algorithm for handling isomorphism of arbitrary algebraic structures it will also work for graphs if we consider the adjacency relation to be a binary operation [2]. Thus the general algebraic structure isomorphism problem is polynomially equivalent to the graph isomorphism problem. When axioms are added to the algebras, it is no longer obvious that an arbitrary graph can be encoded as an algebra. We can ask for which axioms the algebraic isomorphism problem is polynomially equivalent to the graph isomorphism problem.

Tarjan has given an $O(n^{c_1 \log n + c_2})$ algorithm for testing isomorphism of groups [12]. Miller then extended the algorithm to work for quasigroups and Latin squares while only affecting the constants c_1 and c_2 [10]. Algorithms of this time complexity are "subexponential" in the sense that c^n is not $O(n^{c_1 \log n + c_2})$ for any constants c_1 and c_2 when $c > 1$. Thus a bound of this type would be an improvement over any known bound for existing graph isomorphism algorithms.

Although we believe that graph isomorphism is not an NP-complete problem, we also believe that it is not as easy as group isomorphism. As a motivation for this remark we prove the following.

THEOREM 2. *Semigroup isomorphism $\equiv_p$ graph isomorphism.*

Proof. In light of Miller and Monk's result it is sufficient to show that graph isomorphism $\propto_p$ to semigroup isomorphism.

Let $G=(V,E)$ be any graph. Define SEMIGROUP(G) to be $(V\cup E\cup\{0\},\circ)$ where $\circ$ is the binary operation defined by

$$x\circ y=y\circ x=\begin{cases} x & \text{if } x=y,\\ y & \text{if } x\in y\in E,\\ \{x,y\} & \text{if } \{x,y\}\in E,\\ 0 & \text{otherwise.}\end{cases}$$

We can observe the following two facts about SEMIGROUP(G): it is a semigroup and given SEMIGROUP(G) we can recover G uniquely.

Associativity is easily verified. The second fact follows from the observation that 0 is the unique element with the property that $x\circ 0=0\circ x=0$ for all x, and that

$$E=\{x\,|\,x\circ y=x \text{ for exactly two } y\neq x\}.$$

All other elements must be in V. The rest of the proof is similar to that of Theorem 1. We conclude that the encoding $(G_1, G_2)\to$ (SEMIGROUP(G_1), SEMIGROUP(G_2)) is a polynomial reduction of graph isomorphism to semigroup isomorphism. Q.E.D.

The construction used in Theorem 2 actually proves a stronger result. The binary operation defined there is obviously commutative.

THEOREM 3. *Commutative semigroup isomorphism $\equiv_p$ graph isomorphism.*

The primary difference between semigroups and either groups or quasigroups is the ability to solve equations. This property plays a central role in the algorithms of both Tarjan and Miller. The construction in Theorem 2 relies heavily upon the fact that we could allow $x\circ y=0$ for *any* values of x and y. The constraint of associativity (or commutativity) was not critical, but the ability to solve equations seems to be a luxury we can't have if we want to encode arbitrary graphs. The conjecture that group isomorphism is intrinsically easier than semigroup isomorphism (and thus, by Theorem 2, graph isomorphism) is based upon the belief that the axioms for groups (or even quasigroups) drastically reduce the number of possible isomorphisms and thus the inherent complexity of the isomorphism problem.

5. Finite automata.

The relationship between semigroups and automata is well known [3], so a polynomial reduction appears trivial, but the familiar construction from a semigroup yields a *semiautomaton* not an automaton, because no start or final states are defined [3]. If we are interested in automata with start states we must do a little more work to prove equivalence.

THEOREM 4. *Finite automaton isomorphism $\equiv_p$ graph isomorphism.*

Proof. The Miller–Monk result extends also to finite automata so we need only show that finite automaton isomorphism $\propto_p$ graph isomorphism.

Let $G=(V,E)$ be any graph. Define AUTOMATON(G) to be ($V\cup$ {"*start*", "*stop*"}, V, δ, "*start*", V) where "*start*" and "*stop*" are special states. The transition function δ is given by

$$\delta(\text{"}start\text{"}, v)=v,$$

$$\delta(\text{"}stop\text{"}, v)=\text{"}stop\text{"},$$

$$\delta(u,v)=\begin{cases} u & \text{if } u=v,\\ v & \text{if } \{u,v\}\in E,\\ \text{"}stop\text{"} & \text{otherwise.}\end{cases}$$

(Strictly speaking the alphabet should be a copy of V, but we ignore this technicality.)

Again we notice two things about the construction: AUTOMATON(G) is a finite automaton and given AUTOMATON(G) we can recover G uniquely. Thus the encoding $(G_1, G_2) \rightarrow$ (AUTOMATON(G_1), AUTOMATON(G_2)) is a polynomial reduction of graph isomorphism to finite automaton isomorphism. Q.E.D.

The construction encodes the graph structure using the state set and the alphabet. We might ask if this is necessary. Leiss has pointed out that if we fix the size of the alphabet, automaton isomorphism becomes polynomial if we assume all states are reachable, as they are here [7]. A similar result holds if we fix the size of the state set. We can say more, however, in the case of semiautomata.

Ignoring the initial and final state designations of an automaton we have a semiautomaton. The usual construction of a semiautomaton from a semigroup uses the semigroup for both the state set and the alphabet. This is not necessary for our purposes, and we can restrict ourselves to a binary alphabet.

THEOREM 5. *Semiautomaton isomorphism $\equiv_p$ graph isomorphism, even if we restrict the alphabet to be of size two.*

Proof. As before we prove only that graph isomorphism $\propto_p$ semiautomaton isomorphism. This time, however, we choose the directed version of graph isomorphism.

Given a graph $G = (V, E)$ define SEMIAUTOMATON(G) to be $(V \cup E \cup \{\text{"}dead\text{"}\}, \{\text{"}in\text{"}, \text{"}out\text{"}\}, \delta)$, where "*dead*", "*in*", and "*out*" are new objects and the transition function is given by

$$\delta(q, \text{"}in\text{"}) = \begin{cases} q & \text{if } q \in V, \\ u & \text{if } q \in E \text{ and } q = (u, v), \\ \text{"}dead\text{"} & \text{otherwise,} \end{cases}$$

$$\delta(q, \text{"}out\text{"}) = \begin{cases} v & \text{if } q \in E \text{ and } q = (u, v), \\ \text{"}dead\text{"} & \text{otherwise.} \end{cases}$$

The usual arguments show that the mapping $(G_1, G_2) \rightarrow$ (SEMIAUTOMATON(G_1), SEMIAUTOMATON(G_2)) is a polynomial reduction of graph isomorphism to semiautomaton isomorphism. Q.E.D.

6. Polynomial isomorphism.

Hartmanis and Berman have examined the equivalence relation defined by polynomial isomorphism [4]. It is a refinement of the polynomial equivalence we have been discussing. They have observed that all known NP-complete problems are actually polynomially isomorphic [4]. We prove a similar result here by showing that all of the problems discussed above are polynomially isomorphic. We use a technical result from [4] to establish our claim.

LEMMA 6. *If L_1 and L_2 are languages over the alphabets Σ_1 and Σ_2, respectively, and there exist polynomial time computable functions $\mathscr{E}_1: \Sigma_1^* \times \Sigma_1^* \rightarrow \Sigma_1^*$, $\mathscr{D}_1: \Sigma_1^* \rightarrow \Sigma_1^*$, $\mathscr{E}_2: \Sigma_2^* \times \Sigma_2^* \rightarrow \Sigma_2^*$, and $\mathscr{D}_2: \Sigma_2^* \rightarrow \Sigma_2^*$ such that for $i = 1, 2$*

1) $\forall x, y \in \Sigma_i^*$, $\mathscr{E}_i(x, y) \in L_i$ *iff* $x \in L_i$,
2) $\forall x, y \in \Sigma_i^*$, $\mathscr{D}_i(\mathscr{E}_i(x, y)) = y$,
3) $\forall x, y \in \Sigma_i^*$, $|\mathscr{E}_i(x, y)| > |x| + |y|$,

then $L_1 \cong_p L_2$ iff $L_1 \equiv_p L_2$.

The function $\mathscr{E}_i$ is used to encode the language L_i into itself by "padding" the strings to an appropriate length. $\mathscr{D}_i$ is a decoding function which retrieves the second argument of $\mathscr{E}_i$ from the padded string. In practice, as observed in [4], it is easy to find the appropriate encoding and decoding functions.

THEOREM 7. *All of the following problems are polynomially isomorphic: graph isomorphism, regular graph isomorphism, semigroup isomorphism, commutative semigroup isomorphism, finite automaton isomorphism, and semiautomaton isomorphism.*

Proof. By use of Lemma 6 it is sufficient to prove the existence of the encoding and decoding functions; the previous theorems have already shown that all of the problems are polynomially equivalent. For simplicity we assume, without loss of generality, that all problems are represented over a $\{0, 1\}$ alphabet.

Graph isomorphism. The encoding function $\mathscr{E}_G$ for graph isomorphism is computed in polynomial time as follows. Given strings x and y determine if x is the description of two graphs, (G_1, G_2). If not, let $n = 0$. If yes, then let n be the maximum number of a vertex in either graph. Construct a description of two graphs (G_1', G_2') where

$$G_i' = G_i \cup H_0 \cup \bigcup_{j=1}^{|y|} H_j.$$

The graph H_0 is K_3 with its vertices numbered $n+1$, $n+2$, $n+3$, and each H_j is either K_1 (if $y_j = 0$) or K_2 (if $y_j = 1$) with distinct, successively higher numbered vertices.

The decoding function is easily computed by finding some copy of K_3 having the highest numbered vertices and then "reading off" y in its binary representation. The three properties of Lemma 6 are easily verified.

Regular graph isomorphism. The same functions work except that we first find r, the degree of regularity ($r \geqq 2$ can be assumed without loss of generality) and then let H_0 be the complement of an $(r+3)$-cycle and let H_j be K_{r+1} (if $y_j = 0$) or $K_{r,r}$ (if $y_j = 1$).

Semigroup isomorphism. The encoding function $\mathscr{E}_S$ is computed by finding n, the highest numbered element in the pair of semigroups (S_1, S_2). Elements a_{n+1}, $a_{n+2}, \cdots, a_{n+|y|+1}$ are then added to both semigroups such that

$$a_i \circ a_{n+1} = a_{n+1} \circ a_i = a_{n+1} \qquad \text{for } 1 \leqq i \leqq n + |y| + 1,$$

$$a_i \circ a_{n+j+1} = a_{n+j+1} \circ a_i = \begin{cases} a_{n+j} & \text{if } y_j = 0 \\ a_{n+j+1} & \text{if } y_j = 1 \end{cases} \qquad \text{for } 1 \leqq i \leqq n + j + 1 \text{ and } i \neq n+1.$$

The decoding function locates a_{n+1} (the unique annihilator or reset element) and then "reads off" the binary representation of y.

Commutative semigroup isomorphism. The semigroup encoding and decoding functions work here also since $\mathscr{E}_S(G_1, G_2)$ is a pair of commutative semigroups iff G_1 and G_2 are commutative.

Finite automaton isomorphism. Assume the initial state is q_1. The encoding function computes the number of states, m, and the number of letters, n, and then adds one new letter a_{n+1} and $|y| + 1$ new states. The transition function is augmented by the extra transitions for $1 \leqq i \leqq n$, $1 \leqq j \leqq |y|$, $2 \leqq k \leqq m$

$$\delta(q_1, a_{n+1}) = q_{m+1},$$

$$\delta(q_{m+j}, a_i) = q_{m+j+1},$$

$$\delta(q_{m+j}, a_{n+1}) = q_{m+j+y_j},$$

$$\delta(q_{m+|y|+1}, a_i) = q_{m+|y|+1},$$

$$\delta(q_k, a_{n+1}) = q_k,$$

Again the decoding is easily polynomial and the conditions of the lemma are satisfied.

Semiautomaton isomorphism. This is similar to the previous construction for automata, although the alphabet size will be three instead of two as was the case in Theorem 5. Q.E.D.

7. Concluding remarks.

We have shown that three isomorphism problems are of the same complexity: graph isomorphism, semigroup isomorphism, and finite automaton isomorphism. We have conjectured that group isomorphism is not of the same complexity but that it is easier. If the conjecture is true $P \neq NP$. If it is false there is an $O(n^{c_1 \log n + c_2})$ algorithm for graph isomorphism. A proof of either result would be most interesting.

An area for further research is to find a problem which is not obviously an isomorphism problem but which is polynomially equivalent to those examined here.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] K. S. BOOTH, *Problems polynomially equivalent to graph isomorphism*, Tech. Rep. CS-77-04, Dept. of Computer Science, Univ. of Waterloo, Waterloo, Ontario.

[3] A. GINZBURG, *Algebraic Theory of Automata*, Academic Press, New York, 1968.

[4] J. HARTMANIS AND L. BERMAN, *On isomorphisms and density of NP and other complete sets*, Proceedings of the Eighth Annual ACM Symposium on Theory of Computing (May 1976), Association for Computing Machinery, New York, 1976, pp. 30–40.

[5] J. E. HOPCROFT AND J. K. WONG, *A linear time algorithm for isomorphism of planar graphs*, Proceedings of the Sixth Annual ACM Symposium on Theory of Computing (May 1974), Association for Computing Machinery, New York, 1974, pp. 172–184.

[6] R. M. KARP, *On the computational complexity of combinatorial problems*, Networks, 5 (1975), pp. 45–68.

[7] E. LEISS, private communication.

[8] A. N. MELIKOV, L. S. BERSHTEYN AND V. P. KARELIN, *Isomorphism of graphs and finite automata*, Engrg. Cybernetics, 1 (1968), pp. 124–129.

[9] G. L. MILLER AND L. MONK, private communication.

[10] G. L. MILLER, *Graph isomorphism, general remarks*, Proceedings of the Ninth Annual ACM Symposium on Theory of Computing (May 1977), pp. 143–150; J. Comput. System Sci., (Dec. 1978), to appear.

[11] R. C. READ AND D. G. CORNEIL, *The graph isomorphism disease*, J. Graph Theory, 1 (1977), pp. 239–363.

[12] R. E. TARJAN, Private communication.