THE COMPLEXITY OF INFINITE H-COLOURING

by

Bruce Bauslaugh

B.Sc. (Hons), University of Victoria, 1988

B.A. (Hons), University of Victoria, 1988

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the department

of Mathematics and Statistics

SIMON FRASER UNIVERSITY

November 1990

ISBN 0-315-69493-9

Canada

Approval

Name:	Bruce Bauslaugh
Degree:	Master of Science (Mathematics)
Title of Thesis:	The Complexity of Infinite H-Coloring

Examining Committee:

Chairman: A. H. Lachlan

P. Hell, Senior Supervisor

B. Alspach

K. Heinrich

A. Mekler, External Examiner

Date Approved November 2, 1990

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

The Complexity of Infinite H-Colouring

Author:

(signature)

Bruce Bauslaugh

(name)

Dec 3 1990

(date)

Abstract

For a fixed graph H, the homomorphism problem for H is the problem of determining whether or not there is a homomorphism of a finite input graph G into H. In this thesis we investigate the complexity of this problem when H is allowed to be countably infinite. We show that there exist infinite graphs with solvable homomorphism problems of very high complexity, as well as ones with arbitrary recursively enumerable degrees of unsolvability. Some results for specific families of infinite graphs are also derived.

Acknowledgements

I would like to thank the following for their assistance and support: Ann Antignano, Gary Bauslaugh, Pavol Hell, Brian Alspach, Kathy Heinrich, Alan Mekler, Rick Brewster, Gary MacGillivray, Frank Ruskey, Charles Morgan, Luis Goddyn, Alison Marchant, John 'Blind Dog' Kerkhoven, Gwyneth Evans, Bill Tucker, Mike Swift, Syvlia Holmes, Karen Seyffarth, Helen Verrall, Susan Marshall, Steve White, Wild1, Terminator, Scoop, Schpeik, Flash, Piku, RLK, Roger Waters, Woz, and the Natural Sciences and Engineering Research Council of Canada.

Table of Contents

List of Figures

Chapter 1: Introduction

In this thesis we will examine the problem of H-colouring, also referred to as the homomorphism problem for H, where H is allowed to be a countably infinite graph. We begin with a few basic definitions and a formal definition of the problem.

We will restrict our attention to simple undirected graphs. Refer to [4] for definitions of the standard terms used. By the *odd girth* of a graph G, denoted og(G), we mean the length of the smallest odd cycle in the graph. A *homomorphism* from a graph G into a graph H is a mapping F: $V(G) \rightarrow V(H)$ such that $uv \in E(G)$ implies that $F(u)F(v) \in E(H)$. The subgraph of H with vertex-set $\{F(v): v \in V(G)\}$ and edge-set $\{F(u)F(v): uv \in E(G)\}$ is called the *image* of G under F. We will define a relation $\leq$ on graphs by letting $G \leq H$ if and only if there is a homomorphism F from G into H. In the special case where H is a subgraph of G and $F(v) = v$ for all $v \in V(H)$, we call the homomorphism F a *retraction* and the graph H a *retract* of G. If G has no retraction onto a proper subgraph, then we say that G is *endomorphism rigid,* or *retract-free*, or a *core*. If there exists a graph H which is a retract of G and is a core, then this graph is unique up to isomorphism, and is called the core of G [16]. Such an H always exists when G is finite, but might not if G is infinite, such as, for example, when $G = K_1 \cup K_2 \cup K_3 \cup \cdots$.

The homomorphism problem for a graph H is the problem of determining whether an arbitrary graph G maps homomorphically into H. We may formally state the problem as follows:

Fix a graph H.

Instance: A finite input graph G.

Question: Is there a homomorphism F: $V(G) \rightarrow V(H)$?

Note that in the special case when H is the complete graph K_n, this problem is exactly the problem of n-colouring. The pre-images of the vertices of H under a homomorphism are exactly the colour classes of an n-colouring of G.

As we have defined the problem above, H may be either finite or infinite. The finite case has been completely solved and is discussed in the next chapter. The infinite case has not been studied until now, and is the subject of this thesis.

When neither $G \leq H$ nor $H \leq G$ are true we say that G and H are *mutually rigid*. A collection $\{G_i: i \in I\}$ for some index set I is called a mutually rigid family of graphs if G_i and G_j are mutually rigid whenever $i \neq j$.

For the standard definitions from the theory of complexity, see [7] and [13]. In this thesis the functions we refer to as complexity measures will always be nondecreasing recursive functions acting on the set of positive integers. When we say that one function is larger than another, we mean that it is asymptotically larger, i.e. f is larger than g if and only if there exist positive integers N and c such that $c \cdot f(n) > g(n)$ whenever $n > N$. If f is larger than g we will write $f \gg g$. In the field of complexity theory, this is usually denoted by $g = O(f)$. In our discussions of complexity, the notation ||a|| will always mean the length of the encoding of an input a. When the input is an integer i, we assume a standard binary encoding, so $||i|| = \lceil \lg(i) \rceil$, when it is a graph G, we will apply the convention ||G|| = |V(G)|, and when it is a word we will assume

$||a_1a_2...a_n|| = n$. The symbol s will always denote the successor function $s(n) = n+1$.

The *membership problem* for a set **A** is the problem of determining whether a given input is a member of the set. Formally:

Fix a finite alphabet Σ, and a subset **A** of Σ^*, where Σ^* denotes the set of all words over Σ.

Instance: a word w in Σ^*.

Question: is w in **A**?

If the membership problem for **A** is solvable, then **A** is called a recursive set. If **A** is not recursive, we may further classify it according to the complexity of its membership problem relative to the membership problems of other non-recursive sets. To do this, we modify the standard definition of Turing machines by allowing them to include special states called *oracles*. An oracle for a set **A** is a special state which can solve the membership problem for **A** in one machine step. When discussing algorithms, we may consider this oracle to be a function in our programming language which returns *true* or *false* in constant time according to whether the value passed is in **A** or not. A set **B** is said to have a membership problem that is ***A-solvable*** if by using an oracle for **A** we may solve the membership problem for **B**. If it is also the case that an oracle for **B** allows us to solve the membership problem for **A**, then the membership problem for **B** is said to be *exactly* **A**-solvable, and **A** and **B** are said to have the same degree of unsolvability. The set of all membership problems can be partitioned into classes with the same degree of unsolvability, and so it is possible to refer to the degree of unsolvability of a problem without specific reference to another problem. An arbitrary problem **P** is said to be exactly **A**-solvable if an oracle for **A** would allow us to solve **P**, and a solution

for **P** would allow us to solve the membership problem for **A**. If we let **H** denote the *Halting Problem* [13], then any problem which is **H**-solvable is said to be *recursively enumerable*, or to have a *recursively enumerable degree of unsolvability*.

Finally, we say that a graph G is *recursive* if the vertex-set V and edge-set E of G are both recursive. In this thesis our graphs will always be countable, and we will always assume V = {1, 2, 3, ...}. Thus, V will always be recursive. Therefore G will be recursive if and only if there exists an algorithm which, given two positive integers, determines whether the vertices are adjacent or not. This will be referred to as the *edge decision problem for G*.

In the next section we give a survey of some known results in the area of homomorphism problems for finite graphs H. In chapter 3 some preliminary results are proved. In chapter 4 we give short proofs of the existence of graphs with highly complex homomorphism problems and graphs with unsolvable homomorphism problems. However, the graphs we examine in chapter 4 will turn out to be more complex than we would like them to be, in the sense that when we construct a graph with an unsolvable homomorphism problem, it will also turn out to be non-recursive; and when we construct graphs with solvable but highly complex homomorphism problems, they will have highly complex edge-decision problems. In chapter 5 we will construct graphs with difficult or unsolvable homomorphism problems, but with a uniform bound on the complexity of their edge decision problems. In chapter 6 we will derive several other results by restricting our attention to specific families of graphs.

Chapter 2: Known Results

For finite undirected graphs, the complexity of the homomorphism problem has been completely determined. In [12] Hell and Nešetřil prove that if H is bipartite, then the homomorphism problem for H has polynomial complexity, and otherwise it is **NP**-complete.

The directed case has proven much more difficult. The separation of bipartite and nonbipartite cases does not carry over, as there are bipartite digraphs known to have **NP**-complete homomorphism problems, and non-bipartite digraphs with polynomial homomorphism problems. In fact, in [9] it is proven that there are oriented trees with **NP**-complete homomorphism problems. However, it is also proven that for oriented paths, at least, the problem is polynomial. At the same time, in [17] it is shown that all directed cycles have polynomial homomorphism problems.

One of the most general results to date is proven by Bang-Jensen et al. in [2]. They show that when H is a semicomplete digraph, that is, a superdigraph of a tournament, then H has an **NP**-complete homomorphism problem if H contains at least two cycles, polynomial otherwise. In [1], Bang-Jensen and Hell further explore the properties of digraphs containing at least two cycles. The first major result in this paper is that when D is a semicomplete bipartite digraph, that is, a superdigraph of a complete bipartite digraph, D has an **NP**-complete homomorphism problem if D contains at least two cycles, polynomial otherwise. A second important result is that if a digraph D contains exactly two directed cycles of lengths p and q, then the homomorphism problem for D is **NP**-hard if

(1) q does not divide p;

(2) the cycles are in different strong components of D;

(3) there is a directed path between the two cycles.

These results, along with other work, led them to conjecture that if H is a digraph without sources or sinks and each component of the core of H is a directed cycle then the homomorphism problem is polynomial, and otherwise it is **NP**-hard. This remains open.

Chapter 3: Preliminary Results

In this section we prove several simple results which will be useful in subsequent sections.

Lemma 3.1: If $G_1 \leq G_2$ and $G_2 \leq G_3$, then $G_1 \leq G_3$.

Proof: If F_1: $G_1 \rightarrow G_2$ and F_2: $G_2 \rightarrow G_3$ are homomorphisms, then $F_3 = F_2 \circ F_1$ is a homomorphism, since $uv \in E(G_1)$ implies $F_1(u)F_1(v) \in E(G_2)$ which implies that $F_2 \circ F_1(u)F_2 \circ F_1(v) \in E(G_3)$. ■

Lemma 3.2: If $G \leq H$, then $\chi(G) \leq \chi(H)$.

Proof: If $\chi(H) = n$, then $H \leq K_n$, and so if $G \leq H$, then $G \leq K_n$ by lemma 3.1. Hence $\chi(G) \leq n$. ■

Lemma 3.3: Let G and H be non-bipartite. If $G \leq H$, then $og(G) \geq og(H)$.

Proof: Let C be a smallest odd cycle in G. Since C is not 2-colorable, the image of C under any homomorphism into H must contain an odd cycle by lemma 3.2. However, the image of C cannot contain more vertices than C, and so $og(G) \geq og(H)$. ■

Lemma 3.4: If $G \leq H$ and $H \leq G$, then for any graph X, $X \leq G$ if and only if $X \leq H$.

Proof: Suppose $X \leq G$. Then as $G \leq H$, and by lemma 3.1, $X \leq H$. If $X \leq H$, then as $H \leq G$ and applying lemma 3.1, $X \leq G$. ■

Corollary 3.5: Let H_1 be a retract of H. Then $G \leq H$ if and only if $G \leq H_1$.

Proof: If H_1 is a retract of H, then H_1 is a subgraph of H so $H_1 \leq H$ using the identity map on H_1. Also, $H \leq H_1$ using the retraction mapping. Thus, by lemma 3.4, $G \leq H$ if and only if $G \leq H_1$. ■

Since every non-empty bipartite graph retracts to an edge, corollary 3.5 implies that for all bipartite graphs H, the homomorphism problem for H is polynomial.

The existence of infinite mutually rigid families of graphs is well-known, for example, see [11]. For our purposes it will be useful to deal with a specific infinite rigid family. The family is defined as follows: let $G = G(n,k)$, $n \geq 3$, $k \geq 3$, be a graph with $\chi(G) = n$ and girth(G) = k. The existence of such graphs is shown in [8, 15, 18]. Furthermore, choose G(n,k) to have as few vertices as possible. Let $G_i = G(i+2, 2i+3)$, that is, $G_1 = G(3,5)$, $G_2 = G(4,7)$, etc. We show in the next lemma that $\mathcal{G} = \{G_i: i > 0\}$ is an infinite mutually rigid family, and then present two other properties of the graphs G_i.

Lemma 3.6: If $i \neq j$, then G_i and G_j are mutually rigid.

Proof: Suppose $G_i \leq G_j$. Then by lemma 3.2, $\chi(G_i) \leq \chi(G_j)$, so $i \leq j$. Also, by lemma 3.3, $og(G_i) \geq og(G_j)$, so $2i+3 \geq 2j+3$, implying $i \geq j$. Thus i=j, a contradiction. ■

The following lemma will also be useful in subsequent sections.

Lemma 3.7: All $G_i \in \mathcal{G}$ are retract-free.

Proof: Suppose G is a retract of G_i. Then $G \leq G_i$ and $G_i \leq G$. By applying lemmas 3.2 and 3.3, we see that $\chi(G) = \chi(G_i)$ and $og(G) = og(G_i)$. However, G_i is a minimal graph with these properties, and so $G = G_i$. ■

Lemma 3.8: All G_i are connected.

Proof: If G_i has two components A and B containing vertices u and v, respectively, then we may color each of A and B with $\chi(G_i) = i+2$ colors so that u and v have the same color. Therefore, if we identify u and v (figure 3.1) obtaining H, H has the same chromatic number as G_i. Furthermore, this new vertex is a cut-vertex, so if it lies in a cycle, that cycle must lie entirely in A or B. Thus, no new cycles are created, and clearly none are destroyed. Thus, this new graph has fewer vertices than G_i, but the same odd girth and chromatic number, a contradiction. ■

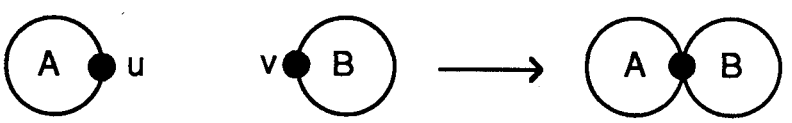

Figure 3.1

This last result will be useful because the homomorphic image of a connected graph must also be connected.

The next two lemmae are related to computational complexity, and will be used extensively in our complexity results. The first one is proved in [7, 13].

Lemma 3.9: For every recursive function f, there is a recursive set **A** whose membership problem has complexity at least f.

Lemma 3.10: Let f, g, and h be nondecreasing recursive functions where $\lim_{n\to\infty} g(n) = \infty$. If $f \circ g \circ s \ll h \circ g$, then $f \ll h$.

Proof: Let N and c be positive integers such that $(f \circ g \circ s)(n) \leq c \cdot (h \circ g)(n)$ for all $n \geq N$, and let M be the smallest integer in the range of g such that $g^{-1}(M) \geq N$. Now let m be any integer greater than or equal to M and let m' be the largest integer less than or equal to m which is in the range of g. Let k be the largest integer such that $k \geq N$ and $g(k) = m'$. Note that $g(k) \leq m \leq g(k+1)$. Now,

$c \cdot h(m) \geq c \cdot h(g(k))$ since h is increasing, and

$c \cdot h(g(k)) \geq f(g(s(k)))$, by the antecedent condition, and

$f(g(s(k))) \geq f(m)$ since f is increasing.

Thus, $f \ll h$. ■

Finally, an important comment on descriptions of infinite graphs. Since an explicit representation of an infinite graph is impossible in a finite algorithm, we describe these graphs by a labelling of the vertices with the positive integers, and a decision procedure which, given a pair of integers u and v, decides whether uv is an edge. Given a graph, the complexity of the edge decision procedure will generally depend upon the numbering of the vertices.

For example, let H consist of an infinite 1-factor and an infinite independent set. If we take some set **A** of natural numbers with an undecidable membership problem, and number the vertices so that (2i)(2i+1) is an edge if and only if $i \in \mathbf{A}$, then the edge decision problem is undecidable. Also, if **A** has a solvable membership problem with high complexity, the edge decision problem under the above ordering will be solvable but will have high complexity. However, if we number the vertices so that (2i)(2i+1) is an edge if and only if i is even, then clearly the edge decision problem has very low complexity.

When we refer to the complexity of the homomorphism problem for some graph H, we do not assume any fixed description of H. In fact, in many cases we will see that a complete description of H is not needed. For example, if H contains arbitrarily large complete subgraphs, then we know that any graph G will map into H, and so our solution to the homomorphism problem for H could be a program which simply ignores its input and answers 'yes'. In such a case H could have no finite representation, but this would not prevent us from solving its homomorphism problem.

Chapter 4: Existence of Graphs with Hard Homomorphism Problems

In this section we prove the existence of graphs H with homomorphism problems of certain complexities. Specifically, we demonstrate the existence of graphs H with unsolvable homomorphism problems, as well as the existence of graphs H with homomorphism problems of arbitrarily high complexity. All of our constructions will involve the family $\mathcal{G}$ defined previously. We note that it is possible to effectively generate the graphs G_i, since we can enumerate all graphs, ordered by number of vertices, until we find one with the desired properties. Thus, let $f(i)$ be the time required to construct the graph G_i.

Our construction is as follows: let $\mathbf{A} = \{a_i : i>0\}$ be a set of positive integers. We define the graph $\mathbf{G_A}$ to be the disjoint union of the graphs $\{G_a : a \in \mathbf{A}\}$. Because $\mathcal{G}$ is a mutually rigid family, G_k maps homomorphically into $\mathbf{G_A}$ if and only if $k \in \mathbf{A}$. We can now prove our first result.

Theorem 4.1: There exist sets **A** such that the graph $\mathbf{G_A}$ has an unsolvable homomorphism problem.

Proof: Let **A** be a set with an unsolvable membership problem. We claim that $\mathbf{G_A}$ must have an unsolvable homomorphism problem. Suppose this were not the case. Then some algorithm **P** would solve the homomorphism problem for $\mathbf{G_A}$, and given an integer n, to test whether $n \in \mathbf{A}$, we need only construct G_n, and use **P** to test whether or not $G_n \leq \mathbf{G_A}$. Since $G_n \leq \mathbf{G_A}$ if and only if $n \in \mathbf{A}$, we now know whether or not $n \in \mathbf{A}$. ■

However, when membership in **A** is solvable it is not immediately clear that the homomorphism problem for $\mathbf{G_A}$ is also solvable. In our next theorem, we see that this is in fact the case.

Theorem 4.2: If **A** has a solvable membership problem, then the homomorphism problem for $\mathbf{G_A}$ is solvable.

Proof: Let G be the input graph. If G is bipartite, then we may immediately conclude that $G \leq \mathbf{G_A}$. If G is not bipartite, let k be og(G). By lemma 3.3, G cannot map into any G_i whose girth is greater than k. Thus, any component G_i where $2i+3 > k$ may be ignored, leaving only finitely many components of $\mathbf{G_A}$ to consider, each of which is itself finite. Let **G'** be defined as the disjoint union of all G_i where $2i+3 \leq k$ and $i \in \mathbf{A}$. Since the membership problem for **A** is solvable, we can effectively construct **G'**. Note that $G \leq \mathbf{G_A}$ if and only if $G \leq \mathbf{G'}$. Also, **G'** is finite so the homomorphism problem for **G'** is solvable. ■

Given the solvability of the homomorphism problem for $\mathbf{G_A}$ when **A** has a solvable membership problem, we may prove the existence of graphs whose homomorphism problems are arbitrarily complex.

Theorem 4.3: For any function f, there exists a graph H whose homomorphism problem has complexity at least f.

Proof: Let f be given. Let g be the function defined by $g(n) = \max\{\, |V(G_i)| : |i| = n, G_i \in \mathcal{G} \}$. Note that $\lim_{n\to\infty} g(n) = \infty$. Let **A** be a set whose membership problem has complexity at least $m = f \circ g \circ s$ (one must exist by lemma 3.9). Let h denote the complexity of the homomorphism

problem for the graph $\mathbf{G_A}$. Since $i \in \mathbf{A}$ if and only if $G_i \leq \mathbf{G_A}$, we may decide the membership in $\mathbf{A}$ of a given input i by using our solution to the homomorphism problem for $\mathbf{G_A}$ on input G_i. Note that $|V(G_i)| \leq g(|i|)$. Thus we know that

$$m = f \circ g \circ s \prec h \circ g$$

and so $f \prec h$ by lemma 3.10.

Thus, $\mathbf{G_A}$ has a homomorphism problem with complexity at least f. ■

In some sense, these results are somewhat unsatisfying because the real difficulty in the homomorphism problem for $\mathbf{G_A}$ is not in determining whether a homomorphism exists, but rather in describing $\mathbf{G_A}$. In theorem 4.1 especially, the reason the homomorphism problem is unsolvable is that it is impossible to even describe $\mathbf{G_A}$ finitely. Similarly in the solvable but highly complex cases, the complexity is largely in the edge decision problem for $\mathbf{G_A}$, although in these cases the complexity of the edge decision problem may be lowered by re-ordering the vertices (as discussed in Ch.3). In subsequent sections we will develop reductions which will yield more satisfying examples of graphs with complex and unsolvable homomorphism problems.

Chapter 5: Homomorphism Problems of Arbitrary Complexities

In this chapter we will show that for any recursively enumerable set **A**, there exist homomorphism problems which are exactly **A**-solvable, that is, solvable with an oracle for **A** but not solvable without an oracle sufficient to decide **A**. In particular, if **A** is recursive, then the homomorphism problem is solvable with no oracle. Furthermore, we will show that for any complexity measure *f*, there is a solvable homomorphism problem with complexity at least *f*. To do this we will show how to reduce the following problem for groups to a homomorphism problem for a graph H.

A *finitely presented* (or f.p.) *group* is a pair (A; R), where A is a finite set of generators $\{a_1, \ldots, a_n\}$, and R is a finite set of equivalences of the form $w = e$, where w is an element of $(A \cup A^{-1})^*$, $A^{-1} = \{a^{-1}: a \in A\}$, and e is the empty word. For our purposes it will be convenient to assume that for all generators a, the equivalence $aa^{-1} = e$ is always explicitly included in R. This pair defines a unique group G whose elements are equivalence classes of $(A \cup A^{-1})^*$, with concatenation modulo R as the group operation, and e as the identity. Two words u and v in $(A \cup A^{-1})^*$ are said to be equivalent if u and v are equivalent modulo R, that is, they represent the same element of G. A *finitely generated* (or f.g.) *group* is a group in which every element is a product of elements of some finite set A of group elements. We will denote such a group by (A; R) where R is now the set of all true equivalences in the group, and either equal to $\{e = e\}$, or is infinite. Given an f.g. group (A; R), it is not always the case that there exists an f.p. group (A; R') that is equal to (A; R) [19]. The word problem for a finitely presented or finitely generated group W is the problem of determining whether a given string w is equivalent to the

identity [19]. Given two groups G_1 and G_2, an embedding of G_1 into G_2 is a one-to-one mapping F: $G_1 \rightarrow G_2$ which preserves the identity and the group operation. The following theorems make claims about the possible complexities of word problems for f.p. groups. The first one is proved in [6].

Theorem 5.1: For any recursively enumerable set **A**, there exists a finitely presented group whose word problem is exactly **A**-solvable.

The proof in [6] shows that if **A** is a recursively enumerable set, then the f.g. group $G = (\{a, b, c, d\}; \{b^{-n}ab^nd^{-n}cd^n = e: n \in \mathbf{A}\})$ has a word problem which is exactly **A**-solvable. The proof of this result uses the fact that in this group the word $b^{-n}ab^nd^{-n}cd^n$ is equal to the identity if and only if $n \in \mathbf{A}$. Furthermore, it is shown that G may be embedded in a f.p. group which is also exactly **A**-solvable.

Lemma 5.2: For any function f, there is a f.g. group whose word problem is solvable and has complexity at least f.

Proof: Let f be given, and let $e(n) = 3^n$. Let **A** be a set whose membership problem is solvable and has complexity at least $g = f \circ e \circ s$. Note that $\lim_{n\rightarrow\infty} e(n) = \infty$. Let G be the group defined in the explanation following theorem 5.1. By theorem 5.1, G has a solvable word problem; let h be the complexity of the word problem for G. We know that $i \in \mathbf{A}$ if and only if $b^{-i}ab^id^{-i}cd^i = e$, so to decide the membership in **A** of a given input i, we need only construct the word $b^{-i}ab^id^{-i}cd^i$, and use our word problem algorithm to see if it is equivalent to the identity. Recall that n has length $|\!|i|\!| = \lceil \lg(i) \rceil$, and note that the word $b^{-i}ab^id^{-i}cd^i$ has length $4i+2 \leq 4 \cdot 2^{|\!|i|\!|}+2 < 3^{|\!|i|\!|} = e(|\!|i|\!|)$.

Since we can solve the membership problem for **A** by applying an algorithm of complexity h to an input of size less than $e(|i|)$, we may conclude that for all n,

$$g = f \circ e \circ s \prec h \circ e$$

so $$f \prec h \text{ by lemma 3.10,}$$

and therefore the word problem for G has complexity at least f. ■

Theorem 5.3: For any f, there exists an f.p. group whose word problem has complexity at least f.

Proof: Let f be given, and let $e(n) = n^2$. Let G be an f.g. group whose word problem is solvable and has complexity at least $g = f \circ e \circ s$. Note that $\lim_{n \to \infty} e(n) = \infty$. Using theorem 5.1 and the comments following it, let G' be an f.p. group that also has solvable word problem and in which we may embed G. Call the embedding function E. Since G is finitely generated, the function E is determined by its action on the finite set of generators of G. Because this set of generators is finite, there is some m such that for all generators a of G, $||E(a)|| \leq m$, and thus for any word w in G, $||E(w)|| \leq m||w||$.

Now, let $h(n)$ be the complexity of the word problem for G'. We may solve the word problem for G given input w by using a solution to the word problem for G' on input E(w). Also,

$$||E(w)|| \leq m||w|| \prec ||w||^2 = e(||w||),$$

therefore

$$g = f \circ e \circ s \prec h \circ e,$$

and so $$f \prec h \text{ by lemma 3.10.}$$

Thus, the word problem for G' has complexity at least f. ■

We note that if some word v is equivalent to another word w in a f.p. group W, then there is some finite sequence of applications of the relations in R which transforms v into w. Each individual application of a relation $(u = e) \in R$ will consist either of replacing a subword u of w with e (a deletion), or replacing an empty subword e of w with u (an insertion). The following lemma is central to our reduction.

Lemma 5.4: If v = w, then there is a transformation of v to w in which all insertions occur before any deletions.

Proof: Suppose we have a transformation which does not have this property. At some point there must be a deletion followed by an insertion. Suppose the string x is deleted and then the string y is inserted, so

$$u_1xv_1 \rightarrow u_1v_1 \rightarrow u_2yv_2 \text{ where } u_1v_1 = u_2v_2.$$

We note that either u_1 is a prefix of u_2 or vice versa. In both cases, the proof is almost identical, so we will just show the case where $u_1 = u_2z$, and so $zv_1 = v_2$. We now have

$$u_2zxv_1 \rightarrow u_2zv_1 \rightarrow u_2yzv_1 = u_2yv_2.$$

This same result may be obtained by first inserting y as follows:

$$u_2zxv_1 \rightarrow u_2yzxv_1 \rightarrow u_2yzv_1 = u_2yv_2.$$

So we may reduce the number of "deletions before insertions" by one, and thus by repeated applications of this procedure we may do all insertions first. ■

We now know that if v = w, then there is a transformation of w to e in which the length of w is increased by insertions and then decreased by deletions. We will call such a transformation *unimodal*. The following lemma is an immediate result.

Lemma 5.5: If $w_1 = w_2 = ... = w_k$ modulo R, then there exists a word u such that all w_i may be transformed into u by insertions only.

Proof: We apply induction on k. For k = 1 let u be w_1. Now suppose the theorem is true for k. Given $w_1 = ... = w_{k+1}$, let v be a word obtainable from all w_i, i = 1, ... , k, by insertions only. Note that v is equivalent to w_{k+1}. Let u be the largest intermediate word in a unimodal transformation of v to w_{k+1}. Clearly u results from a series of insertions in v. Also, w_{k+1} results from performing a series of deletions to u, so u may be obtained by a series of insertions in w_{k+1}. ■

Now, let W = (A; R) be a finitely presented group. Using W, we will construct an infinite graph H whose homomorphism problem is solvable exactly when the word problem for W is solvable. Furthermore, we will show that when the homomorphism problem for H is solvable, we can force its complexity to be arbitrarily high. For our construction we will need the infinite mutually rigid family $\mathcal{G} = \{G_i\}$ constructed in Chapter 3. We first fix some arbitrary vertex in each G_i and call it v_i. Now construct new graphs H_i by taking G_i and

attaching to v_i a path whose length is $\max_{j \le i} |V(G_j)|$. Call the end of this path u_i.

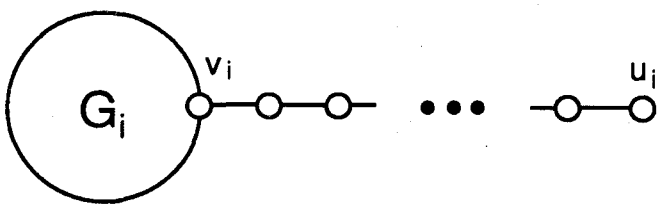

Figure 5.1

We will refer to the subgraph of H_i which is equal to G_i as the *body* of H_i and to the added path as the *tail* of H_i. The following lemmas will prove very useful in our construction.

Lemma 5.6: If H_i is an induced subgraph of a graph H, and u_i is the only vertex in H_i adjacent to anything not in H_i, then any mapping of G_i into H where the image of G_i intersects with the body of H_i is an isomorphism of G_i onto the body of H_i.

Proof: G_i is connected, and so its image must be connected. Since the tail of H_i contains at least $|V(G_i)|$ vertices, and the image of G_i contains no more vertices than G_i, if the image of G_i intersects with the body of H_i, the image must be contained in H_i.

Assume some vertex of G_i maps to a vertex in the tail of H_i, and so there is a vertex in the body of H_i not in the image of G_i. Also, the image of G_i cannot be entirely contained in just the tail of H_i and v_i because $\chi(G_i) \ge 3$, and the subgraph induced by the vertices of the tail and v_i is bipartite. Therefore, the image of G_i must contain v_i and one of its neighbours x in the body of H_i. Let H' be the intersection of the image of G_i with the body of H_i. Since there is a vertex in the body of H_i which is not in the image of G_i, H' is a proper

subgraph of the image of G_i. We may map the image of G_i onto H' by using the identity map on vertices in H', and mapping all vertices in the tail of H_i onto v_i or x, depending on whether their distance from v_i is even or odd. Thus, we now have a homomorphism in which the image of G_i is properly contained in the body of H_i, which is itself isomorphic to G_i. The image of G_i must (by lemmas 3.2 and 3.3) have odd girth no larger than $og(G_i)$ and chromatic number at least as large as $\chi(G_i)$, but since it is a subgraph of G_i it must have the same odd girth and chromatic number as G_i. However, it has fewer vertices, contradicting the minimality of G_i. Thus, G_i must map onto the body of H_i. Therefore, it is an isomorphism. ■

This immediately implies that if we map H_i into a graph H containing a copy of H_i as a subgraph (as above), and if the image of the body of H_i intersects with the body of the copy, then u_i maps into the copy of H_i.

Lemma 5.7: If H_i is an induced subgraph of a graph H, and u_i is the only vertex in H_i adjacent to any vertex not in H_i, then every mapping of G_j, $j<i$, into H has an image which does not intersect the body of H_i.

Proof: G_j is connected, and the tail of H_i contains at least $|V(G_j)|$ vertices. Thus, if the image of G_j intersects the body of H_i, then the image is contained in H_i. However, $H_i \leq G_i$, and so $G_j \leq G_i$, a contradiction. ■

Let G = (A; R) be an f.p. group. We begin the construction by taking a lexicographic enumeration L of all words over $A = \{a_1, \ldots, a_n\}$, that is, $L = \{e, a_1, \ldots, a_n, a_1a_1, \ldots\}$. The i^{th} element of this enumeration will be denoted L(i), and will be said to have index i. We associate with L(i) the

graph H_i. We construct our infinite graph H as follows. We begin with one copy of each H_i. Next, for each pair of the special vertices u_i and u_j, $i \neq j$, we add a new vertex u(i, j), which we connect to u_i and u_j with an edge if and only if the word L(i) may be obtained from the word L(j) by insertions only, or by deletions only. We note at this point that it is possible to determine whether one word is obtainable from another by insertions only because insertions always increase the length of a word, and there are only finitely many possible insertions to try (a symmetric argument applies to deletions). We obtain the graph like the one shown in figure 5.2, where some of the u_i are connected to others by 2-paths (in this illustration we use equivalences $a_1a_1 = e$ and $a_1a_2 = e$).

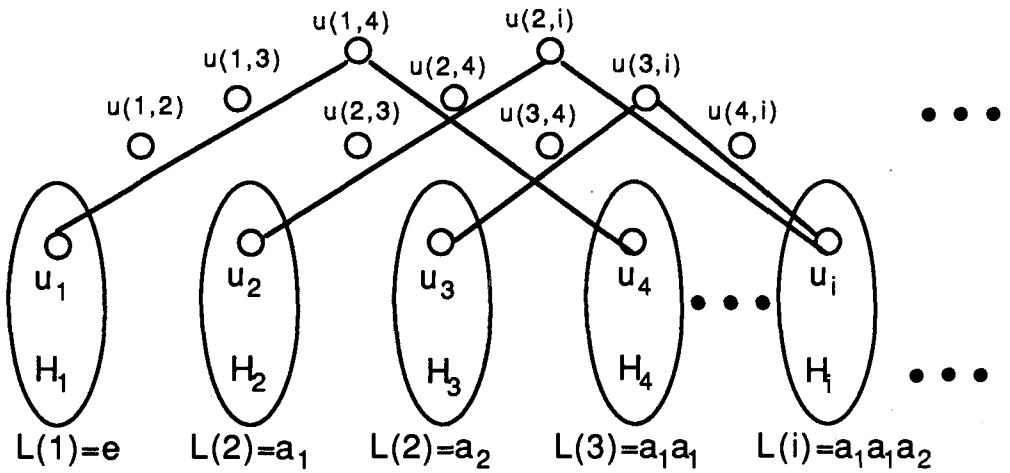

Figure 5.2

Note that two subgraphs H_i and H_j of H will be in the same component of H if and only if the words L(i) and L(j) are equivalent.

We will now show that if we can solve the homomorphism problem for H, then we can solve the word problem for W, and vice versa. Thus, if the

word problem for the group W is exactly A-solvable, then the homomorphism problem for H is exactly A-solvable.

Lemma 5.8: Let W be a f.p. group, and let H be the graph obtained from the above construction. If the homomorphism problem for H is solvable, then the word problem for W is solvable.

Proof: Suppose the homomorphism problem for H is solvable. Let w be the input to the word problem for W, and let i be the index of w in L. The index i is easily determined since L is a lexicographic enumeration of all words over the alphabet A. We construct the graph G (figure 5.3) as input to the homomorphism algorithm.

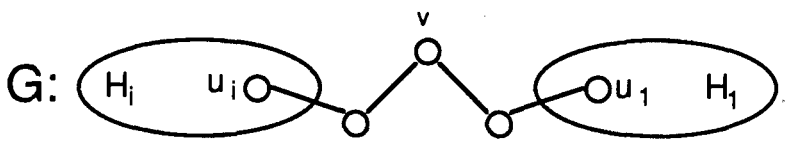

Figure 5.3

We claim that w is equivalent to the identity if and only if $G \leq H$. If w is equivalent to the identity, then by lemma 5.5 and our construction, there is a vertex u_j in H which is connected by 2-paths to both u_i and u_1, that is, L(j) is obtainable from both w and e by insertions only, and the 2-paths are $[u_1, u(1,j), u_j]$ and $[u_j, u(i, j), u_i]$. Thus, G maps isomorphically into H, with v mapping to u_j, and so maps homomorphically into H.

Suppose that w is not equivalent to the identity. We will show that $G \not\leq H$. We claim first that if $G \leq H$, then the bodies of H_1 and H_i in G must map to the corresponding bodies in H. Note that if we delete all of the bodies

of the H_j in H, what remains is 2-colourable (colour all of the u_i the same colour). Thus, the images of the H_i and H_1 in G must intersect with the bodies of some H_j in H. No image of a body G_x, x = 1 or i, can intersect with the body of an H_j when j > x, by lemma 5.7. Recall that each G_x must have an image with chromatic number at least $\chi(G_x)$. If we delete the bodies of all H_j, $j \geq x$, from H, then what remains has chromatic number smaller than $\chi(G_x)$. This is clear since all of the blocks of this graph have chromatic number less than $\chi(G_x)$. Thus, G_x cannot map into this graph, so its image in H must intersect with the body of some H_j, $j \geq x$. However, we know it cannot intersect with an H_j, j > x, and thus the image of G_x must intersect with the body of the copy of H_x in H. Now, by lemma 5.6, we know that the body of the H_x in G must map to the body of the H_x in H, x = 1 or i.

Also, by our construction, if two words L(i) and L(j) are not equivalent, then there is no path in our graph from u_i to u_j. Thus, if w is not equivalent to the identity, the H_i in G must map into a different component than the H_1 in G, but this is imposible as G is connected. ■

Lemma 5.9: Let W be an f.p. group, and let H be the graph obtained from our construction. If the word problem for W is solvable, then the homomorphism problem for H is solvable.

Proof: Suppose the word problem for W is solvable. Let G be an input to the homomorphism problem for H, and assume without loss of generality that G is connected. If G is bipartite then it maps to an edge, and so $G \leq H$ and we are done. If G is not bipartite, then let k = og(G), and let n = |V(G)|. Now, by lemma 3.3, the image of G cannot be contained in any H_i with odd girth larger than k. Note that the odd girth of H_i is equal to the odd girth of its body,

and that $og(H_j) = 2j + 3 > 2j$. The image of G cannot intersect with the body of any H_i which has odd girth greater than k and tail length greater than n, since this would imply that the image of G is contained in this H_i. Furthermore, the tail length of H_j is at least the number of vertices in G_j, which is greater than 2j, because G_j contains a (2j+3)-cycle. It follows that if $m = \max\{k/2, n/2\}$, the image of G cannot intersect with the body of any copy of H_j with $j \geq m$.

We now construct a finite graph H^1 with the property that $G \leq H^1$ if and only if $G \leq H$. First, let H^2 be the subgraph of H induced by all H_i, $i < m$, and all u(i, j), i, j < m. Now, using our solution to the word problem, separate the words L(i), i < m, into equivalence classes $\{I_1, \ldots, I_r\}$. The graph H^1 is obtained by adding, for each equivalence class I_j, a new vertex v_j to H^2, and connecting it by 2-paths to all u_i, $i \in I_j$. The center of the 2-path between v_j and u_i will be labelled w_i (figure 5.4).

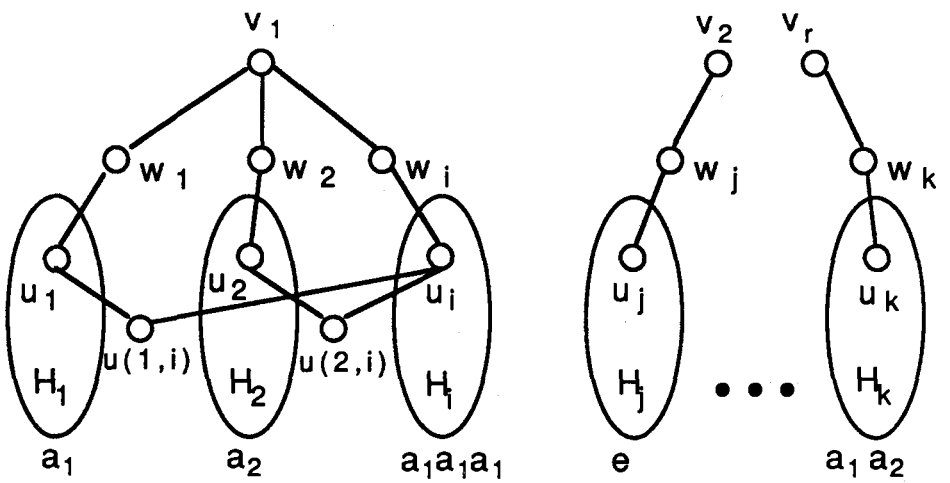

Figure 5.4

We claim that $G \leq H^1$ if and only if $G \leq H$. Suppose that $G \leq H^1$. We will show that $H^1 \leq H$, and so $G \leq H^1 \leq H$. Consider some component of H^1, and

the equivalence class I_j associated with this component. There is a word w which is obtainable from every word in $\{L(i): i \in I_j\}$ by insertions only (use lemma 5.5). Let t be such that $w = L(t)$.

Observe that this component of H^1 is actually an induced subgraph of H, with all H_i and $u(i, j)$ mapping to their conterparts, v_j mapping to u_t, and w_i mapping to $u(i,t)$. Thus $H^1 \leq H$.

Suppose now that G maps into H, and look at the image of G in H under a mapping F. It must be in one component of H, and so the set $\{L(i): H_i$ intersects with the image of $G\}$ is contained in an equivalence class I_k of $(A \cup A^{-1})^*$. Also, G is finite, so this set is finite. This image cannot contain part of any body of an H_i, $i \geq m$, and so it is contained in a subgraph of H like the one in figure 5.5.

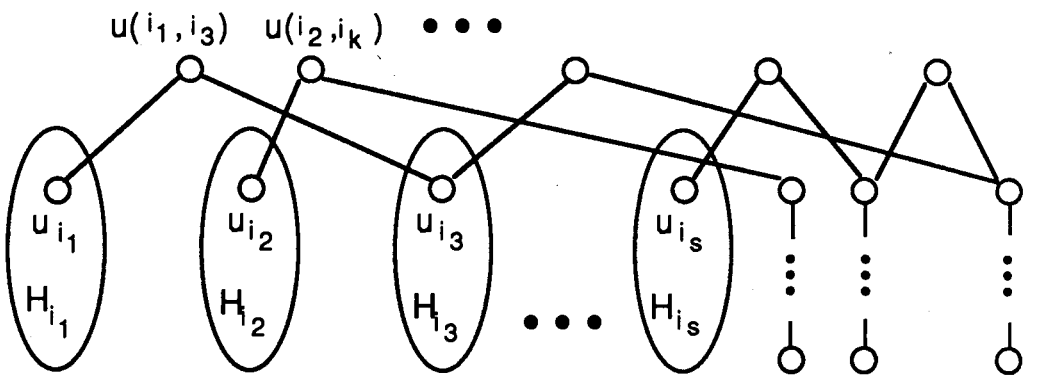

Figure 5.5

Map this graph into the component of H^1 corresponding to I_k by mapping all u_i, $i \geq m$, to v_k (with the tails mapping in the obvious way), mapping all $u(i, j)$, $i, j \geq m$, to any neighbour of v_k, mapping all $u(i, j)$, $i < m$, $j \geq m$, to w_i, and mapping each other vertex to the corresponding vertex in H^1. ■

We are now in a position to state the first major result in this chapter.

Theorem 5.10: For every recursively enumerable set **A**, there exists a recursive graph H whose homomorphism problem is exactly A-solvable.

Proof: By the above construction, for any word problem there is a graph H whose homomorphism problem has the same degree of unsolvability. Furthermore, H is recursive, as demonstrated by the above construction. Together with theorem 5.1, this proves our result. ■

The following result for those graphs which have solvable homomorphism problems can also be proven.

Theorem 5.11: For any complexity function f, there exists a graph H with a solvable homomorphism problem whose complexity is at least f.

Proof: Let f be given. Define a function c by $c(n) = \max\{|V(G(i))|: ||L(i)|| = n\}$, where G(i) is the graph in figure 5.3, using H_1 and H_i. Note that $\lim_{n\to\infty} c(n) = \infty$. Let W be an f.p. group whose word problem is solvable and has complexity at least $g = f \circ c \circ s$, and let H be the graph obtained by applying our construction to W. Since W has a solvable word problem, the homomorphism problem for H is solvable. Now, given an input w to the word problem, we may decide if w is the identity as follows. First, find i such that L(i) = w and next, construct the graph G of figure 5.3. We know from the proof of theorem 5.8 that w is the identity if and only if $G \leq H$. So we may now use an algorithm which solves the homomorphism problem for H to decide whether w is the identity. Note that the complexity of constructing G from w is

constant, and so is insignificant when f is large. Let h be the complexity of the homomorphism problem for H. We obtain

$$g = f \circ c \circ s \prec h \circ c,$$

so $f \prec h$ by lemma 3.10.

Hence, the homomorphism problem for H has complexity at least f. ■

The one remaining property we would like these graphs H to possess is that we would like them to be relatively simple to construct. That is, we would like the complexity of the edge decision problem under a natural ordering of the vertices to be low when compared to the complexity of the homomorphism problem. We say that the ordering should be natural to avoid the changes in edge decision complexity that may arise as a result of reordering the vertices of a graph. We will formalize this notion of naturalness in a moment. Note that for any f.p. group, our graph H always consists of the same set of H_i, $i \geq 1$, and $u(i, j)$, $1 \leq i < j$. Thus, we will assume that our enumeration of the vertices of these objects is constant for all groups. Also, all of the edges and non-edges are constant except those which may or may not occur between vertices u_i or u_j and $u(i, j)$. If uv is a constant edge or non-edge we will call $\{u, v\}$ a constant pair of vertices. For an ordering to be deemed natural, then, we will be fairly generous and require only that the following are possible under the ordering:

(i) Given two vertex labels u and v, it is decidable whether or not $\{u, v\}$ is a constant pair.

(ii) Given a constant pair, it is decidable whether or not it is an edge or a non-edge; and

(iii) given a non-constant pair $\{u, v\}$, we may calculate the indices i and j such that u and v represent one of u_i and u_j, and $u(i, j)$.

Such an ordering is easily obtained and in fact nearly any ordering which one would intuitively call natural satisfies these conditions. For example, let (a_i, b_i), $i \geq 1$, be an computable enumeration of $\{(i, j): i < j\}$. If we enumerate first the vertices of H_1, then the vertex $u(a_1, b_1)$, then the vertices of H_2, then the vertex $u(a_2, b_2)$, etc., we will have such an ordering. If we order the vertices in this way, then if we are given two vertices u and v, we may always construct a subgraph of H which is sufficiently large to contain u and v, and with all vertices labelled as being either $u(i, j)$, u_i, or something else. At this point we can easily obtain the solution to (i), (ii), and (iii) from our partial graph. Note also that the same algorithm may be used for the above problems for all graphs H, since the properties decided are constant over all f.p. groups.

Theorem 5.12: There is a recursive function *h* such that the graph H obtained from any f.p. group by the above contruction has an edge decision problem with complexity bounded by *h*.

Proof: Let *t* be the combined complexity of determining (i), and then (ii) or (iii) as necessary. Let H be the graph obtained by applying our construction to an f.p. group $W = (A; R)$. Let $r = |R|$, and assume $|A| > 1$. Suppose we want to decide whether an edge uv is in H. By first applying an algorithm for the problem just discussed, in time $t(\|\{u, v\}\|)$ we may decide whether $\{u, v\}$ is a constant pair and if it is, decide if it is an edge or a non-edge. If it is not, we may calculate the indices i and j such the u and v represent u_i and $u(i, j)$, or u_j and $u(i, j)$. Thus, the only computation left to perform is to determine whether one of L(i) and L(j) is obtainable from the other by insertions only. Assume without loss of generality that L(i) is longer than L(j). Then there are at most $\|L(i)\|-\|L(j)\|$ insertions required to transform L(j) to L(i). There are

never more than $\|L(i)\|$ possible insertion points, since we can stop as soon as our word has length $\|L(i)\|$, and there are always r choices of strings to insert. Thus, the total number of possible transformations is no more than $(r\cdot\|L(i)\|)^{\|L(i)\|-\|L(j)\|}$. Also,

$$(r\cdot\|L(i)\|)^{\|L(i)\|-\|L(j)\|} \leq (r\cdot\|L(i)\|)^{\|L(i)\|}$$

and $\|L(i)\|$ is $\lceil \log_n(i) \rceil$ when $|A| = n$. Since $\log_n < \log_2 = \lg$ for all $n > 1$, the total number of possible transformations is bounded by $(r\cdot\lg(i))^{\lg(i)}$.

Now, define a function f by $f(n) = \max\{i : \exists\ v \in V(H),\ \|v\| \leq n,$ and $\exists\ j \leq i$ such that $v = u_i$, $v = u(i, j)$, or $v = u(j, i)\}$. Intuitively, given a non-constant pair of vertices $\{u, v\}$ with $n = \max\{\|u\|, \|v\|\}$, $f(n)$ is a bound on i and j, where L(i) can be obtained from L(j) by insertions if and only if $\{u, v\}$ is an edge. Note that f is recursive and nondecreasing, and that f is the same for any graph H obtained from an f.p. group by our construction. Thus, to determine whether a non-constant pair $\{u, v\}$ is an edge requires time $(r\cdot\lg(f(m)))^{\lg(f(m))}$, where $m = \max\{\|u\|, \|v\|\}$. Since the actual complexity of this problem must be stated in terms of $n = \|\{u, v\}\|$, we note that since $m \leq n$, the complexity is bounded by $(r\cdot\log(f(n)))^{\log(f(n))}$.

Thus, given an arbitrary pair $\{u, v\}$, with $n = \|\{u, v\}\|$, the total time required to determine whether or not $\{u, v\}$ is an edge is bounded by $g(n) = t(n) + (r\cdot\log(f(n)))^{\log(f(n))}$.

The only parameter of the function g which varies between graphs is r, and so we may easily find a function h which bounds g for all r. For example, let $h(n) = t(n) + (n\cdot\log(f(n)))^{\log(f(n))}$. Since n is always eventually larger than r, which is of course constant for a given graph, h always asymptotically bounds the complexity of deciding edges in a graph H. ■

Thus, although the complexity of the homomorphism problem for H may be arbitrarily high, the complexity of the edge decision problem is never higher than the function h described above. Therefore, we have succeeded in finding graphs with difficult homomorphism problems but relatively simple edge decision problems under a natural ordering of vertices.

Chapter 6: Results on Restricted Classes of Graphs

In this chapter we will examine particular families of graphs and give some results regarding the complexity of the homomorphism problem when we restrict our attention to graphs in these families. We begin by presenting two constructions which will be used subsequently.

Given a recursive graph G, define the graph **F**(G) as follows. Let G_i, $i \geq 1$, be the subgraph of G induced by the vertices $\{1,..., i\}$. Then **F**(G) is the disjoint union of the G_i, $i \geq 1$.

Lemma 6.1: Let G be a recursive graph. Then **F**(G) is also recursive, and for any finite graph H, $H \leq G$ if and only if $H \leq \mathbf{F}(G)$. Furthermore, **F**(G) has no infinite components.

Proof: Observe that **F**(G) is recursive, since to construct a component G_k of **F**(G) we need only use our edge decision procedure for G on the finite set $\{(i,j): 1 \leq i < j \leq k\}$. It is also immediately obvious that **F**(G) has only finite components.

Since $\mathbf{F}(G) \leq G$, if H maps into **F**(G) then H maps into G. If $f: H \rightarrow G$ is a homomorphism then let $k = \max\{f(x): x \in V(H)\}$. Then H maps into G_k, so H maps into **F**(G). ■

Note also that since **F**(G) has no infinite components, all of its vertices have finite degree. Such a graph is said to be *locally finite.*

Our next construction does not work for all recursive graphs, but only on a restricted class called *highly recursive* graphs. A graph is highly recursive if it is recursive and the degree function $d(v)$ for vertices of G is also recursive

[14]. Note that such a graph must be locally finite. The important property these graphs possess which not all recursive graphs possess is that given a vertex v in a highly recursive graph, the neighbourhood N[v] can be determined. This allows us also to find set of all vertices at distance no more than two from v, that is, N[N[v]]; the set of all vertices of distance no more than three from v, and so on.

Given a highly recursive graph G, define $\mathbf{R}(G)$ as follows. Choose some arbitrary vertex v in G which is called the *center* of $\mathbf{R}(G)$ (we will get different constructions depending upon which center we choose, but this will not affect subsequent results). Let G_i, $i \geq 0$, be the subgraph induced by all vertices of distance no more than i from v. Let $\mathbf{R}(G)$ be the disjoint union of the G_i.

Lemma 6.2: Let G be highly recursive. Then $\mathbf{R}(G)$ is highly recursive, and for all finite graphs H, $H \leq G$ if and only if $H \leq \mathbf{R}(G)$. Also, $\mathbf{R}(G)$ has no infinite components.

Proof: Since G is highly recursive, we can construct each of the components of $\mathbf{R}(G)$, and so $\mathbf{R}(G)$ is recursive. Also, we can calculate the degree of any vertex of $\mathbf{R}(G)$ by constructing the component which contains it, and simply counting its neighbours, so $\mathbf{R}(G)$ is highly recursive.

Now, let v be the center of $\mathbf{R}(G)$. If $f: H \rightarrow G$ is a homomorphism, then let k be the maximum distance from v of any f(x), $x \in H$. Then $H \leq G_k$, so $H \leq \mathbf{R}(G)$. Also, $\mathbf{R}(G) \leq G$, so if $H \leq \mathbf{R}(G)$, then $H \leq G$. ■

We note at this point that given any graph G with the required properties for the above constructions, the complexity of the homomorphism

problem for G is the same as that for **F**(G) and **R**(G), since the complexity is given in terms of the size of the input graph, not any property of the target graph, and an algorithm for one of these graphs will give correct answers for the homomorphism problem for the others. We are now in a position to state our first result.

Theorem 6.3: There are recursive graphs G in which all components are finite, but the homomorphism problem for G is unsolvable. In fact, the homomorphism problem for G may have any degree of complexity or unsolvability.

Proof: From the results of chapter 5, there exist recursive graphs with homomorphism problems of the required complexities. Given such a graph G, **F**(G) has the same properties and has only finite components. ■

This result shows that hard homomorphism problems do not depend upon the presence of vertices of infinite degree in the target graph. In fact, the proof of theorem 6.3 shows that locally finite graphs may have unsolvable homomorphism problems. However, the maximum degree of vertices in a graph **F**(G) or **R**(G) may still be unbounded. We will prove a still stronger result later.

Two especially interesting classes of graphs are vertex-transitive graphs and Cayley graphs. A graph is called vertex-transitive if for every pair of vertices u and v, there is an automorphism of the graph mapping u to v. In other words, the automorphism group of the graph is a transitive group. A Cayley graph for a group G with symbol S, denoted (G;S), is a graph whose vertices are the elements of G, and uv is an edge whenever $v = us$ for some $s \in S$. We assume that $e \notin S$, $S^{-1} = S$, and that S generates G so (G;S) is

connected, loopless, and undirected. For the purposes of complexity analysis, we will always assume without loss of generality that the elements of S are represented by single characters. It is easy to prove that Cayley graphs are vertex-transitive. We will now prove some simple results for these families.

Theorem 6.4: All locally finite recursive vertex-transitive graphs G have solvable homomorphism problems.

Proof: Let G be recursive and vertex-transitive. Vertex-transitive graphs are regular, so a locally finite recursive vertex-transitive graph is trivially highly recursive. Thus, **R**(G) is defined. Let v be the center of **R**(G). Given an input graph H, let u be any vertex of H. If f: $H \rightarrow G$ is a homomorphism, then let w = f(u). Now let g be an automorphism of G which maps w to v. The composition $g \circ f$ is a homomorphism from H into G which maps u to v. Thus, we may arbitrarily choose some vertex u of H and we know that there is a homomorphism of H to G which maps u to v if and only if there is a homomorphism from H to G. Assume without loss of generality that H is connected and let d = diameter(H). Then $H \leq G$ if and only if $H \leq G_d$, and G_d is finite. ■

We may generalize this result slightly as follows.

Proposition 6.5: If the automorphism group of a locally finite recursive graph G has only finitely many orbits, then G has a solvable homomorphism problem.

Proof: Suppose G has k orbits and let $v_1, \ldots, v_k$ be representatives from each orbit. Let R_i be the graph **R**(G) with v_i used as the center of **R**(G). Let H be an input graph, and let u be a vertex of H. Assume again that H is

connected and d = diameter(H). We know that $H \leq G$ if and only if there is a homomorphism $f: H \rightarrow G$ with $f(u) = v_j$ for some j, $1 \leq j \leq k$. Thus, $H \leq G$ if and only if for some j, $1 \leq j \leq k$, $H \leq R_j$ with u mapping to v_j. But $H \leq R_j$ if and only if H maps to the d^{th} component of R_j, and so we need only construct the d^{th} component of each R_j and check whether $H \leq R_j$. ■

Unfortunately, we have no results regarding the complexity of the edge decision problem for $\mathbf{R}(G)$, and so we cannot make any stronger claim than that the problem is solvable. However, if we further restrict ourselves to Cayley graphs, we may make a stronger claim.

Theorem 6.6: Let (G;S) be a Cayley graph and suppose we have an oracle **A** for the word problem for G. Then there is a non-deterministic polynomial time algorithm for the homomorphism problem for (G;S). Furthermore, the length of the words passed to **A** is never larger than the size of the input graph.

Proof: Suppose we are given a graph Z as input. Assume without loss of generality that Z is connected. We claim that there exists a vertex v in Z such that every other vertex in Z has distance no more than $\lfloor 1/2 \cdot |V(Z)| \rfloor$ from v. To prove this, take a spanning tree T of Z and look at a longest path P of T. Let v be the center of this path (or one of the centers if it has odd length). The maximum distance from v to any vertex on this path is $r = \lfloor 1/2 \cdot |V(P)| \rfloor$. Since T is a tree and P a longest path of T, it follows that the maximum distance from v to any vertex in T is $\lfloor 1/2 \cdot |V(P)| \rfloor$. Thus, the maximum distance from v to any vertex in Z is no more than $\lfloor 1/2 \cdot |V(P)| \rfloor \leq \lfloor 1/2 \cdot |V(Z)| \rfloor$. If Z is not a path, then this inequality is strict. We may easily find such a vertex in polynomial time by doing |V(Z)| breadth-first searches of our graph.

Now, we non-deterministically 'guess' what the image of Z in (G;S) is. Since (G;S) is vertex-transitive we may assume that v gets assigned the empty word (that is, gets mapped to the identity). We assign to the edges of Z elements of S. We assign to each vertex u of Z a word over S equal to the concatenation of the symbol elements assigned to a shortest path from v to u. All vertices get words of length no more than r, since any vertex in the image of Z may be reached from v by a path of no more than r edges. So far we have used a number of non-deterministic steps which is only polynomial in the size of Z.

The next step is to verify that this assignment of words determines a homomorphism of Z to (G;S). To do this we simply take all adjacent pairs of vertices r,s from Z. Call their words l(r), l(s), and call the symbol element of the edge d connecting them l(d). We note that the sum of lengths $|||l(r)|| + |||l(s)|| + |||l(d)||$ is no more than n, since $|||l(r)||$ and $|||l(s)||$ are strictly less than $\lfloor 1/2 \cdot |V(Z)| \rfloor$ when Z is not a path, and we cannot have adjacent vertices both of distance $\lfloor 1/2 \cdot |V(Z)| \rfloor$ from v in a path. We now use our word problem oracle to determine whether l(r)l(d) = l(s) or l(s)l(d) = l(r), and so whether or not there is an edge between the images of r and s in (G;S).

Thus, after non-deterministically guessing the image of Z in (G;S), we were able to verify that it was an image in polynomial time using our oracle. Hence, this problem is in **NP**. ■

Corollary 6.7: Let (G;S) be a Cayley graph, where G has a solvable word problem with complexity $w(n)$. There exists a deterministic algorithm which solves the homomorphism problem for (G;S) with complexity no more than $n^2 \cdot w(n) \cdot (|S|^{n^2})$.

Proof: There are $|S|^{|E(Z)|}$ ways to assign elements of S to the edges of Z. For each of these we must check to see if this assignment determines a homomorphic image of Z in (G;S). To do this we can use our word problem algorithm for each pair of adjacent vertices. By the proof of theorem 6.6 the input we pass will have length no more than $|V(Z)|$. Also, there are no more than $|V(Z)|^2$ such pairs of vertices. The number of steps used by the algorithm, then, is no more than $|V(Z)|^2 \cdot w(|V(Z)|) \cdot (|S|^{|E(Z)|})$. Since $|E(Z)| < |V(Z)|^2$, and $||Z|| = |V(Z)|$, the above complexity may be bounded by $n^2 \cdot w(n) \cdot |S|^{n^2}$. ■

The following two corollarys are now trivial.

Corollary 6.8: Let (G;S) be a Cayley graph. If the word problem for G is in **NP** then the homomorphism problem for (G;S) is in **NP**.

Corollary 6.9: Let (G;S) be a Cayley graph. If the word problem for G is **NP**-hard then the complexity of the homomorphism problem for (G;S) is no greater than the complexity of the word problem for G.

Having proven these upper bounds, there remains a question regarding lower bounds for these problems. At this point we present an example of a Cayley graph (G;S) where G has an unsolvable word problem but (G;S) has a very easy homomorphism problem.

Let G = (A; R) be a finitely presented group with generators A and relations R and an unsolvable word problem. Construct G' = (A'; R') by letting

$A' = \{a': a \in A\}$ and $R' = \{ (a_1'a_1'...a_k'a_k' = e) : (a_1...a_k = e) \in R \}$. It is easy to see that the word problem for G' is also unsolvable, since the relations in R' can only apply to strings of pairs of a_i', and must treat each pair exactly as the relations in R treat the a_i. Thus, for any word w on A, the word w' on A' obtained by replacing a_i by $a_i'a_i'$ is the identity if and only if w is. Now if we look at the Cayley graph (G'; A') we see that it is bipartite, since any word on A' equal to the identity must have even length, and so all cycles in (G'; A') must have even length. Thus, even though the word problem for G' is unsolvable, the homomorphism problem for (G'; A') has only polynomial complexity.

This raises an interesting question. Is it, in fact, the case that all Cayley graphs, or even all vertex-transitive graphs, have relatively easy homomorphism problems? For vertex-transitive graphs in general, at least, we may answer negatively, as the following theorem shows.

Theorem 6.10: There exist locally finite vertex-transitive graphs with unsolvable homomorphism problems.

Proof: By construction. We begin with a Directed Cayley Colour-Graph. This is a Cayley graph in which edges are directed and coloured according to which symbol element they represent, that is, if uv is an edge because $us = v$ for some $s \in S$, then the colour of this edge is s and it is an edge from u to v. Let $G = (A; R)$ be a finitely presented group with an unsolvable word problem, and let H be the Directed Cayley Colour-Graph (G; A). Define a modified homomorphism problem for H, where edges in the input graph are directed and coloured and must be mapped to edges of the same direction and colour. This problem is unsolvable, since a word $w = a_1...a_k$ is

equal to the identity if and only if a directed k-cycle with edges coloured $a_1,...,a_k$ maps into H. For the same reason, this homomorphism problem is unsolvable even when we restrict our input graphs to directed cycles. The general method of this proof is to reduce this modified homomorphism problem to a homomorphism problem in an undirected, uncoloured graph which is still vertex-transitive (even though it may no longer be a Cayley graph).

We first need to define a new mutually rigid family of graphs. Let $G_1 = K_5$ and recursively define G_{i+1} to be the graph with the least number of vertices which has both chromatic number and odd girth greater than the chromatic number and odd girth of G_i. If our original graph H has n different edge colours then we will need to construct $G_1,...,G_{3n}$. In each G_i fix some arbitrary pair of adjacent vertices u_i and v_i. Let m be one more than the number of vertices of the largest G_i, and let D be the graph obtained by taking two m-paths and connecting so as to create a string of K_4 (Figure 6.1).

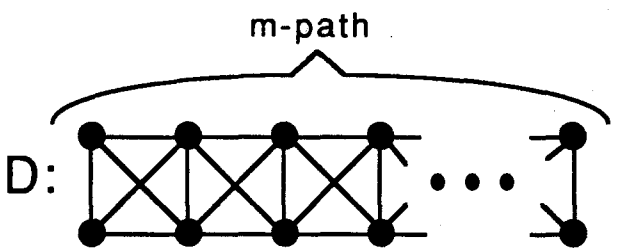

Figure 6.1

We now construct graphs E_i, i = 1, ..., n, by joining G_{3i+1}, G_{3i+2}, and G_{3i+3} by identifying the end vertices of four copies of D to their respective u_i and v_i, as indicated in figure 6.2. Label the loose ends a_1, a_2, b_1, and b_2.

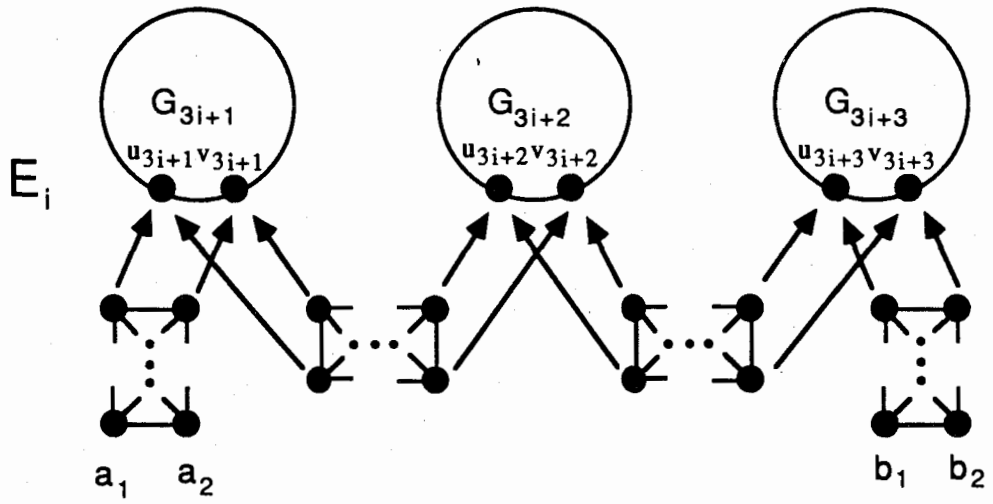

Figure 6.2

Next, we replace each vertex v in the Directed Cayley Colour-Graph H with a copy of K_2 with vertices v_1 and v_2 and replace each directed edge e = xy of colour i with E_i, by identifying a_i with u_i and b_i with v_i. Call this graph B. We claim that if G and H are Directed Cayley Colour-Graphs and we perform this replacement on both G and H to obtain G' and H', then $G \leq H$ if and only if $G' \leq H'$. To see this, consider first the image of some G_j in G' under a homomorphism f. Because of the length of the copies of D connecting different G_k, $f(G_j)$ may only intersect one G_k in H', call it G_a. Note that G_j has chromatic number greater than four, D has chromatic number equal to four, and $f(G_j)$ contains some portion of G_a plus possibly some subgraphs of one or two copies of D which intersect with G_a in a K_2. Thus, the intersection of $f(G_j)$ and G_a must have chromatic number at least as large as $\chi(G_j)$, so $a \geq j$. By definition of the G_i, $og(G_a) \geq og(G_j)$, and so the odd girth of the intersection of $f(G_j)$ and G_a is also at least $og(G_j)$. Therefore, f must be a one-to-one mapping of G_j into G_a, or we would have a graph smaller than G_j with equal or larger odd girth and chromatic number, a contradiction. It now easily follows by

definition of the G_i that $j = a$, and so G_j and G_a are isomorphic, and f is an isomorphism. Also, since a copy of G_{3i+1} is only attached by a copy of D to one G_{3i+2}, and a G_{3i+2} is only attached by a copy of D to one G_{3i+3}, the subgraph E_i must map onto another copy of E_i, and the G_j within E_i must map isomorphically to their copies. This forces the copies of D to also map isomorphically, and so the E_i will act exactly as the coloured edges in the original graph. This proves the claim.

The problem, of course, is that this graph is not vertex transitive. However, it is clear that its automorphism group has only finitely many orbits. We can automorphically map any copy of E_i onto any other copy of E_i, since any edge in a Cayley colour graph may be mapped to any other edge of the same colour by an automorphism. Thus, there are no more orbits than there are vertices in all of the E_i. We will now construct a new graph B^* which is vertex transitive which also has an unsolvable homomorphism problem.

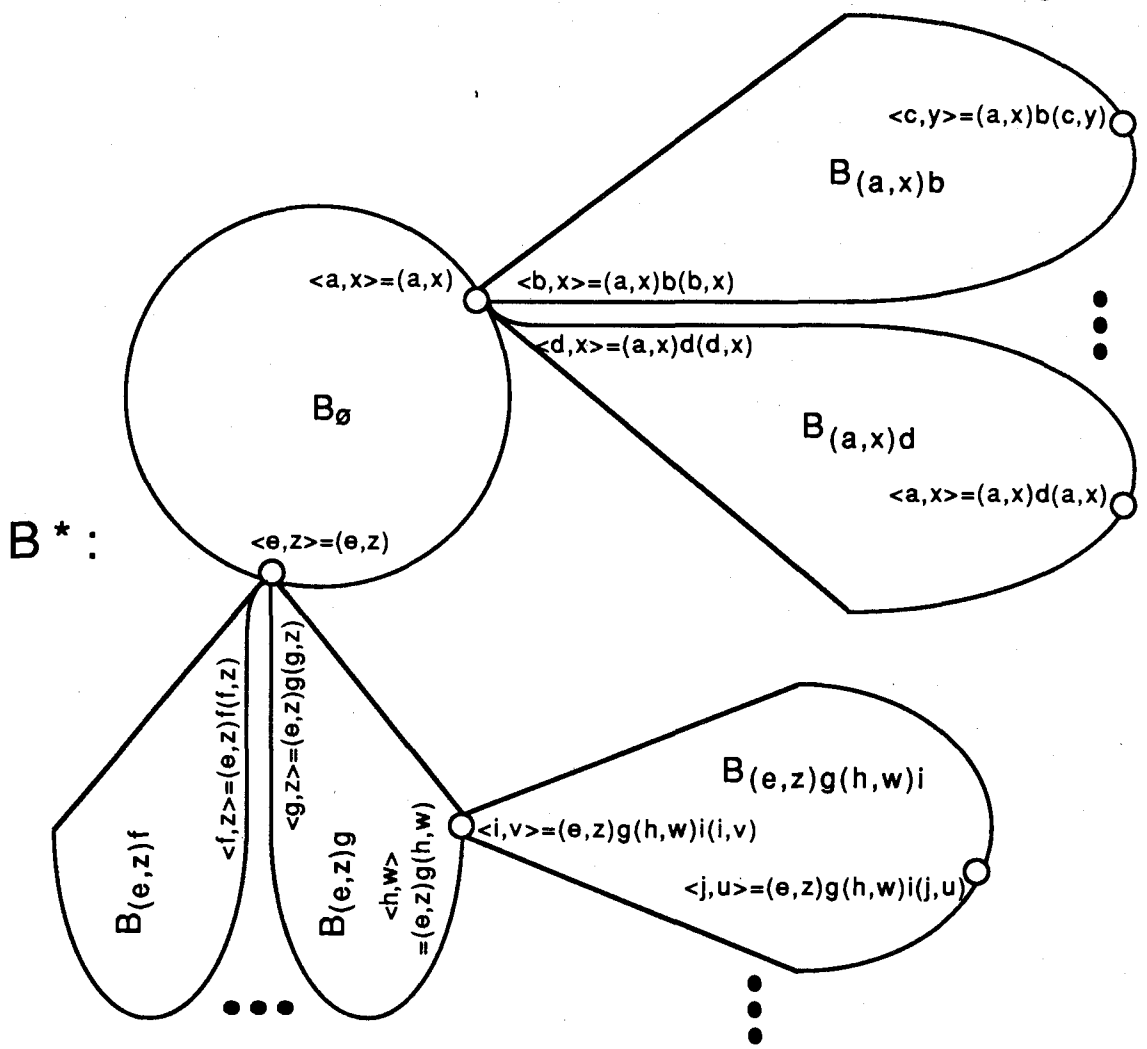

Figure 6.3

Let k be the number of orbits in the automorphism group of B. Note that each orbit contains an infinite number of vertices. Assume that the orbits are numbered 1, ..., k and the vertices within each orbit are numbered 1, 2, 3, ... Label the j^{th} vertex in the i^{th} orbit with <i, j>. To construct B^*, for each string of the form $ß = (a_1,b_1)c_1(a_2,b_2)c_2...c_{n-1}(a_n,b_n)c_n$, where $n \geq 0$, and each a_i, c_i = 1, ..., k, $b_i \geq 1$, take one copy of B called $B_ß$. We will call ß the *label* of $B_ß$. We will denote the empty string (n = 0) by ø. Next, partition the $ß_i$ into equivalence classes where $ß_1$ and $ß_2$ are equivalent if $ß_1$ may be transformed to $ß_2$ by substitutions of the form (a,b)a = ø and (x,y)z(z,y) = (x,y). For,

example, (a,b)a(c,d)e is equivalent to (c,d)e. For each equivalence class, delete all $B_ß$ except the one with the (unique) shortest label in the class.

Now, in a graph $B_ß$, we will label the vertex <i, j> with the string ß(i,j), that is, ß concatenated with (i,j). We complete the construction by identifying two vertices if their labels are equivalent under the substitutions given above ((a,b)a = ø and (x,y)z(z,y) = (x,y)), and labelling the resultant vertex with the shortest label of any of the original vertices. It will sometimes be convenient to refer to vertices by labels other than this one, and so when we say a vertex has label ß(i,j), we mean the shortest string equivalent to ß(i,j). The resulting graph is shown (in part) in figure 6.3. Each vertex is incident to k copies of B, and occurs in a different orbit of the automorphism group of B in each one. For example, the vertex (a,x) occurs in orbit a in $B_ø$, and for each $q \neq a$ it is the x^{th} orbit q in $B_{(a,x)q}$.

The overall structure of B^* is that of an infinite tree-structure with copies of B as nodes, and $B_ø$ as the root. The labels of the vertices represent paths from the root to the vertex in question, that is, the label $(a_1,b_1)c_1(a_2,b_2)...c_{n-1}(a_n,b_n)$ is interpreted as: begin with vertex $v_1 = (a_1,b_1)$ in $B_ø$, move to the copy of B incident to v_1 in which v_1 is in orbit c_1 ($B_{(a1,b1)c1}$), move to vertex v_2=<a_2, b_2>=$(a_1,b_1)c_1(a_2,b_2)$ in this copy of B, move to the copy of B adjacent to v_2 in which v_2 is in orbit c_2 ($B_{(a1,b1)c1(a2,b2)c2}$), and so on. The equivalences we used may now be interpreted as follows.

If a vertex label contains (a,b)a, then at some point in the path we move to the b^{th} vertex of the a^{th} orbit of some copy of B, and then we move to the copy of B in which this vertex is in the a^{th} orbit. This is the same copy of B, so this substring may be deleted without effect. If a vertex label contains (x,y)z(z,y), then at some point in the path we move to the y^{th} vertex of the x^{th}

orbit of some copy B_1 of B, and then we move to the copy B_2 of B in which this vertex is in the z^{th} orbit, and then we move to the y^{th} vertex in the z^{th} orbit of this copy. However, this is the same vertex we came from, except that it is being viewed as a vertex in B_2 instead of B_1. Thus, the substring z(z,y) may be deleted from (x,y)z(z,y) without effect.

Note that B^* is locally finite. To show that B^* is vertex-transitive, we present the following two sets of automorphisms.

(i) $F_{(x,y)z}(ß(a,b)) = (x,y)zß(a,b)$, is an automorphism for all x, y, z. That is, if we concatenate any fixed string (x,y)z to the left side of the label of every vertex in B^* we obtain an automorphism of B^*.

(ii) if f is an automorphism of B, then

$G_f((a_1,b_1)c_1(a_2,b_2)c_2...c_{n-1}(a_n,b_n)) = (f(a_1,b_1))c_1(f(a_2,b_2))c_2...c_{n-1}(f(a_n,b_n))$

is an automorphism of B^*.

That these mappings preserve edges follows easily from the definition of B^*. Furthermore, each mapping of form (i) has an inverse, specifically, $F_{(x,y)z}^{-1} = F_{(z,y)x}$ since $(z,y)x(x,y)zß(a,b) = (z,y)zß(a,b) = ß(a,b)$ for all vertices ß(a,b). Each mapping of form (ii) also has an inverse, as $(G_f)^{-1} = G_{(f^{-1})}$. Thus, all of these mappings are automorphisms.

Now, we show that by means of these automorphisms, the vertex (1, 1) may be mapped to any other vertex in B^*, and so the automorphism group has only one orbit. To map (1,1) to $(a_1,b_1)c_1...c_{n-1}(a_n,b_n)$, first find an automorphism f of B that maps <1, 1> to <1, b_n>. Using G_f, we may map (1, 1) to (1, b_n). Now use $F_{(a_n,b_n)1}$ to map (1,b_n) to $(a_n,b_n)1(1,b_n) = (a_n, b_n)$. The applications of $F_{(a_{n-1},b_{n-1})c_{n-1}}$, ..., $F_{(a_1,b_1)c_1}$ complete the procedure.

We now demonstrate the other crucial property of B^*, which is that it still has an unsolvable homomorphism problem. First of all, we claim that if C^P

is the graph obtained by applying our edge-replacement construction to a coloured directed cycle P, then $C^P \leq B$ if and only if $C^P \leq B^*$. This will suffice because, as was stated earlier, the homomorphism problem for the Directed Cayley Colour-Graph H is unsolvable even when input is restricted to cycles. So for a cycle P, $P \leq H$ if and only if $C^P \leq B$ if and only if $C^P \leq B^*$, and therefore if we could solve the homomorphism problem for B^* we could solve the homomorphism problem for H with input restricted to cycles.

We will now prove the above claim. Let C be a graph obtained from a directed coloured cycle by our construction. Obviously $C \leq B$ implies $C \leq B^*$. Suppose $C \leq B^*$. If the image of C is contained in only one copy of B, then $C \leq B$. On the other hand, if the image intersects with more than one copy of B, then the image contains a cut point, since copies of B intersect only in cut points of B^*. We will show that this cannot happen.

Let us first look at the image of some copy of G_i in C under a homomorphism f. If this $f(G_i)$ contains a cut point, then all blocks in the image must have fewer vertices than G_i and each block must be contained in only one copy of B. Since the chromatic number of $f(G_i)$ is at least as large as that of G_i, some block Z of $f(G_i)$ must have chromatic number at least as large as $\chi(G_i)$, which is at least five. Hence, Z cannot be entirely within a copy of D, which has chromatic number 4. Also, by the fact that the distance between the ebds of D is larger than the size of any G_i, we know that Z cannot intersect with more than one G_j in B. Thus, it intersects with exactly one G_j, say G_a. It may of course also contain part of the copies of D incident with G_a. However, since D has chromatic number four and intersects with G_a in a K_2, we know that the chromatic number of Z is the same as the chromatic number of the intersection of Z and G_a. Thus, $\chi(G_a) \geq \chi(Z) \geq \chi(G_i)$, and so by the definition

$og(G_a) \geq og(G_i)$. Now, the intersection of Z and G_a has chromatic number at least that of G_i, odd girth at least that of G_i, and fewer vertices than G_i, a contradiction. We now know that $f(G_i)$ cannot contain a cut point, and so must be contained in one copy of B, and so in fact G_i must map isomorphically to some copy of G_i, by our previous argument.

Note also that the image of a K_4 under a homomorphism is always a K_4, which contains no cut point. We will now show that if we "glue" graphs G_i and K_4 together in a certain way, the new graphs also contain no cut point in their image in B^*.

If X and Y are two nonempty graphs, define X!Y to be the graph obtained by identifying some K_2 in X with one in Y (figure 6.4).

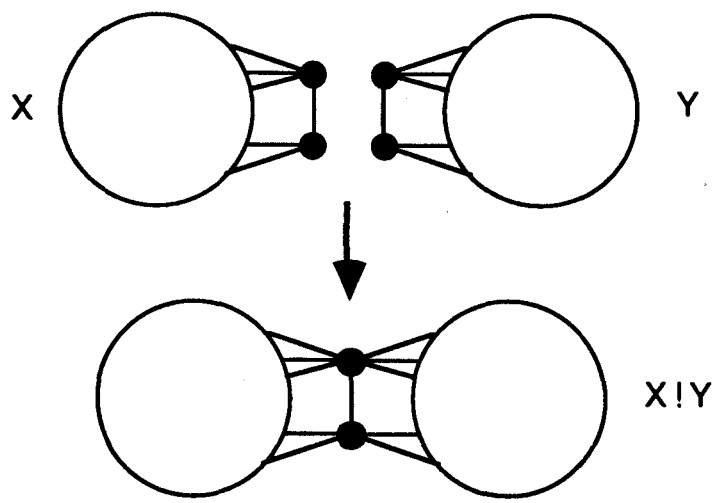

Figure 6.4

Suppose F is a homomorphism from X!Y into some fixed graph W, and suppose also that neither F(X) nor F(Y) contains a cut point. We claim that F(X!Y) does not contain a cut point. This is easily shown, since 2-connectivity is equivalent to the statement that for every three distinct vertices u, v, and w there is a path from u to v which misses w (see [4]). Let u and v be vertices in

F(X!Y). When u and v are in the image of the same graph (X or Y), then such a path must exist as these images are 2-connected. If one of these vertices (say u) is in the image of X and the other is in the image of Y, then we look at the image of the K_2 which joins X and Y. The image of this K_2 must be a K_2, and so contains two vertices, at least one of which is not w. Let z be a vertex different from w in the image of the K_2. We now simply take a path from u to z in the image of X avoiding w, and one from v to z in the image of Y avoiding w. The concatenation of the paths gives us a walk from u to v in the image of X!Y avoiding w and so there is a path from u to v avoiding w.

We can 'almost' construct C just using copies of various G_i and copies of K_4 joined using the ! operation in the obvious way. Specifically, we may construct the graph C' in figure 6.5, which is the same as C except that it is missing the four dashed edges.

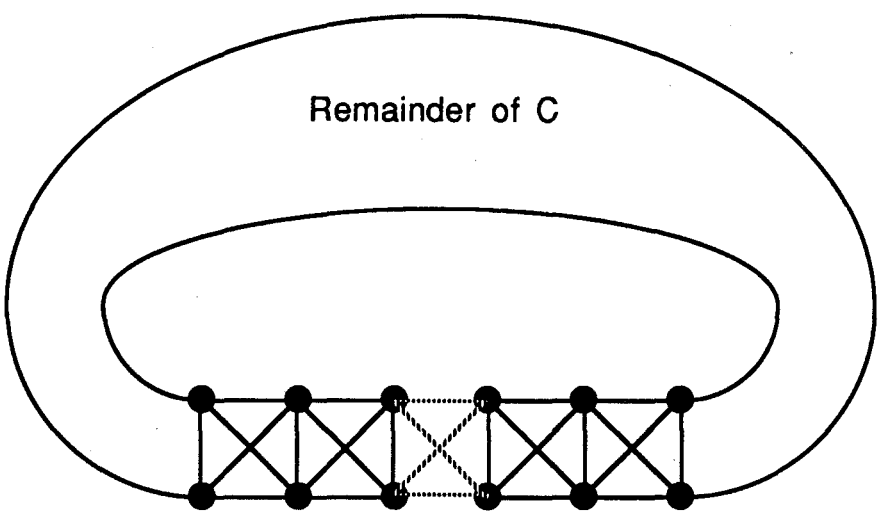

Figure 6.5

However, C' is a spanning subgraph of C, so whenever C has an image in B^* with connectivity k, C' has an image in B^* with connectivity $k' \leq k$. This is

because every image of C contains an image of C' as a spanning subgraph. Thus, since any image in B^* of the graph in figure 6.5 (without the dashed edges) has connectivity at least two, so does any image of C. Thus, if $C \leq B^*$ then the image of C is contained in one copy of B, so $C \leq B$.

Therefore, B^* is a vertex-transitive graph with an unsolvable homomorphism problem. ■

Corollary 6.11: There exist graphs of bounded degree (and therefore with finite chromatic number) with unsolvable homomorphism problems.

Proof: The graph B^* is locally finite and vertex-transitive, and so has bounded degree. B^* also has an unsolvable homomorphism problem. ■

We may well ask at this point whether there are any interesting families of graphs for which the homomorphism problem is easily solved. It appears that a high degree of symmetry is of no help, since any class containing all vertex transitive graphs contains unsolvable instances. We may, however, derive some elementary sufficient conditions for solvability of this problem.

Theorem 6.12: Let H be a graph and suppose there is a retraction of H to a finite graph. Then the homomorphism problem for H is in **NP**.

Proof: Let G be a finite retraction of H. G is equivalent to H with respect to homomorphisms by lemma 3.5 so the complexity of the homomorphism problem for H is the same as for G. However, all finite graphs fall into **NP**. Furthermore, if G is bipartite them the homomorphism problem for H is in **P**, otherwise it is **NP**-complete. ■

It should be clear that a similar result applies for graphs H when there exists a finite set **S** of finite graphs with the property that for any finite G, $G \leq H$ if and only if $G \leq S$ for some $S \in \mathbf{S}$.

In closing, we present a trivial but nice-sounding result.

Corollary 6.13: All perfect graphs have homomorphism problems in **NP**.

Proof: Any k-chromatic perfect graph contains a k-clique, and so it retracts to a k-clique. If we want to consider graphs with infinite cliques, they are obviously in **P**. ■

References

[1] J. Bang-Jensen, P. Hell, *The Effect of Two Cycles on the Complexity of Colorings by Directed Graphs,* Discrete Applied Math., no. 26 (1990), pp 1-23.

[2] J. Bang-Jensen, P. Hell, G. MacGillivray, *The Complexity of Coloring by Semicomplete Digraphs,* Siam J. Disc. Math., vol. 1 no. 3 (Aug. 1988), pp 281-298.

[3] J. Bang-Jensen, P. Hell, G. MacGillivray, *On the Complexity of Coloring by Superdigraphs of Bipartite Graphs*, Simon Fraser University CSS/LCCR TR90-01 (1990).

[4] J.A. Bondy, U.S.R. Murty, *Graph Theory with Applications*, (Elsevier) 1976.

[5] J.I. Brown, D.G. Corneil, *On Generalized Graph Colorings*, J. Graph Theory, vol. 11 no. 1 (1987), pp 87-99.

[6] C.R.J. Clapham, *Finitely Presented Groups with Word Problems of Arbitrary Degrees of Insolubility*, Proc. London Math. Soc., 3rd Series vol. XIV no. 56 (Oct. 1964), offprint.

[7] M.D. Davis, J.D. Weyuker, *Computability, Complexity and Languages*, (Academic Press) 1983.

[8] P. Erdos, *Graph Theory and Probability*, Canad. J. Math., 11 (1959), pp 34-38.

[9] W. Gutjahr, E. Welzl, G. Woeginger, *Polynomial Graph-Colorings*, Freie Universitat Berlin, Tech. Rep. B-88-06.

[10] J. Hartmanis, R.E. Stearns, *On the Computational Complexity of Algorithms*, T.A.M.S., no. 117 (1965), pp 285-306.

[11] P. Hell, *On Some Strongly Rigid Families of Graphs and the Full Embeddings they Induce*, Algebra Universalis, vol. 4 fasc. 1 (1974), pp 108-126.

[12] P. Hell, J. Nešetřil, *On the Complexity of H-Coloring*, J. Combin. Theory ser. B, Vol. 48, no. 1 (Feb 1990), pp 92-110.

[13] J.E. Hopcroft, J.E. Ullman, *Introduction to Automata Theory, Languages and Computation*, (Addison-Wesley) 1979.

[14] H.A. Kierstead, *Recursive Colorings of Highly Recursive Graphs*, Can. J. Math., vol. XXXIII no. 6 (1981), pp 1279-1290.

[15] L. Lovasz, *On Chromatic Number of Finite Set Systems*, Acta Math. Acad. Sci. Hung., vol. 19(1-2) (1968), pp 59-67.

[16] G. MacGillivray, *The Complexity of Generalized Colorings*, Ph.D. Thesis, Simon Fraser University, 1990.

[17] H.A. Maurer, J.H. Sudborough, E. Welzl, *On the Complexity of the General Coloring Problem*, Inform. and Control 51 (1981), pp 123-145.

[18] J. Nešetřil, V. Rodl, *A short Proof of the existence of Highly Chromatic Hypergraphs without short cycles*, J. Comb. Theory., series B no. 27 (1979), pp 225-227.

[19] J. R. Shoenfield, *Mathematical Logic*, (Addison-Wesley) 1967.

[20] G. Turan, *Models of Computation for Graph Theoretic Problems*, Bonn University report no. 83299-OR.