volume 114

lecture notes in pure and applied mathematics

computers in geometry and topology

from a conference 1986 at Illinois Chicago

Edited by
Martin C. Tangora

Computers in Geometry and Topology

PREFACE

the Mandelbrot set could
chine.
ology and homological al-
hat David Eisenbud some-
ricate and beautiful
ronomically large compu-

m the reader will also
matical software and one
ology.
n Computers in Geometry
the University of Illinois
the UIC Department of
, with Martin C. Tangora
orship and financial sup-
llege of Liberal Arts and
and from the National Sci-

akers will be found here,
ributed papers from other
eed.
ter, which, through its
ulty and students, has fos-
ys to bring the new tech-

titude to the referees; to
upport in the whole process
mail networks, BITNET
waiting for correspondence;
—especially to Ioan James—
the book was brought to

Martin C. Tangora

Contributors

David J. Anick Associate Professor, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts

Thomas F. Banchoff Professor, Department of Mathematics, Brown University, Providence, Rhode Island

Max Benson Associate Professor, Department of Computer Science, University of Minnesota, Duluth, Minnesota

Robert R. Bruner Assistant Professor, Department of Mathematics, Wayne State University, Detroit, Michigan

Edward Curtis Professor, Department of Mathematics, University of Washington, Seattle, Washington

Donald M. Davis Professor, Department of Mathematics, Lehigh University, Bethlehem, Pennsylvania

Peter J. Giblin Professor, Department of Mathematics, University of Massachusetts at Amherst, Amherst, Massachusetts, and Liverpool University, Liverpool, England

Ivan Handler Independent Software Developer, Chicago, Illinois

John C. Harris* Assistant Professor, Department of Mathematical Sciences, Purdue University Calumet, Hammond, Indiana

Louis H. Kauffman Professor, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, Illinois

*Current affiliation: Acting Assistant Professor, Department of Mathematics, University of Washington, Seattle, Washington

Larry A. Lambe* Professor, Department of Mathematics, North Carolina State University, Raleigh, North Carolina

Mark Mahowald Professor, Department of Mathematics, Northwestern University, Evanston, Illinois

John Milnor Professor, School of Mathematics, Institute for Advanced Study, Princeton, New Jersey

Douglas C. Ravenel Professor, Department of Mathematics, University of Rochester, Rochester, New York

David L. Rector Professor, Department of Mathematics, University of California at Irvine, Irvine, California

Robert F. Riley Assistant Professor, Department of Mathematical Sciences, State University of New York at Binghamton, Binghamton, New York

Dan Sandin Professor, Department of Art and Design, University of Illinois at Chicago, Chicago, Illinois

*Current affiliation: Professor of Mathematics and Computer Science, Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, Illinois

Contents

1

The Computation of Rational Homotopy Groups Is #𝒫-Hard

DAVID J. ANICK

Massachusetts Institute of Technology
Cambridge, Massachusetts

We give a measure of the computational complexity of homotopy groups. Given a finite simply connected CW complex X, a common problem in algebraic topology is to evaluate $\dim(\pi_n(X) \otimes Q)$. This problem is shown to belong to the class of #𝒫-hard problems, which are believed to require more than polynomial time to compute deterministically. Computing the Hilbert series of a graded algebra or the Poincaré series of a local Artinian ring is also #𝒫-hard.

Intended for topologists, the exposition is self-contained, assuming no prior familiarity with theoretical computer science concepts.

1. HISTORICAL CONTEXT

Beginning with the earliest definitions of homotopy groups $\pi_*(\cdot)$ by Čech [10] and Hurewicz [16], topologists have been interested in computing or describing them. In [25] Serre recognized that the homotopy groups of a finite 1-connected simplicial complex X would be finitely generated abelian groups. By the known classification of such groups, any $\pi_n(X)$ would have to have a simple description as

$$\pi_n(X) = \underbrace{Z \oplus \cdots \oplus Z}_{r} \oplus Z_{a_1} \oplus \cdots \oplus Z_{a_m} \qquad [1]$$

where $a_1 \mid a_2 \mid \cdots \mid a_m$. This discovery meant that $\pi_*(\cdot)$ could be be viewed as a function, with a finite description of X and an integer n as inputs and with the finite list $(r; a_1, a_2, \ldots, a_m)$ as output.

In 1957 Brown [9] proved that there exists an algorithmic procedure, which always terminates after a finite number of steps, for evaluating this function. At the time this seemed to settle the matter: homotopy groups can be computed, and topologists' job is to do it. However, despite much creative work since then, our explicit knowledge of homotopy groups remains dismally poor. As Paul Selick has observed, there is not a single noncontractible finite 1-connected CW complex X for which $\pi_n(X)$ is known for all n. Experience teaches us that to call the computation of general homotopy groups "very hard" would be a laughable understatement.

A crucial ray of light in this darkness was offered by Quillen in 1969 and later expanded by Sullivan [27] and others. Quillen [23] showed that the integers r of (1), corresponding to the ranks of the rational homotopy groups $\pi_*(X) \otimes Q$, participate in an algebraic model for X_Q (X_Q is X localized at Q) which renders them far more accessible than the whole $\pi_*(X)$. By using this model, the complete rational homotopy of X is now known for a great many finite 1-connected X. Rational homotopy seems to be significantly easier to compute than the total homotopy group, and developments in rational homotopy continue to shed light on questions about $\pi_*(X)$.

During the past 15 years the field of theoretical computer science has blossomed tremendously, from a near obscurity practiced by a few specialists to a major mathematical effort involving many deep concepts and theorems. Questions about the computability of algebraic or topological objects can now be posed in far more refined ways, and the answers will have both theoretical and practical importance.

This, then, is the context in which the results of this chapter appear. Computation of the free abelian component of $\pi_n(X)$ will be shown to be $\#\mathcal{P}$-hard, a well-known concept in computer science. Since $\#\mathcal{P}$-hard problems sit rather high on the complexity scale, this confirms our experience that homotopy groups, even rational homotopy groups, are difficult to evaluate.

Computer scientists conjecture, but have not proved, that $\#\mathcal{P}$-hard problems cannot be solved in polynomial time on an ordinary deterministic machine. If true, this conjecture would imply that a worst-case rational homotopy calculation requires more than polynomial time, which would be a very powerful result. Thus topologists are left in the position of waiting until further advances are made in the field of theoretical computer science!

Even if this conjecture were solved, it would not close the chapter on computability in homotopy. For one thing, there has been increased interest recently in extending Brown's results to non-simply connected spaces. Weld [29] succeeded at this for nilpotent CW complexes. Her results suggest that the best we can hope for with a typical nilpotent space X will be a recursively enumerable presentation for $\pi_n(X)$. Whether one can obtain $\pi_n(X_Q)$ precisely for nilpotent X and whether Brown's results can be extended to the determination of [Y; X] when Y is not a sphere and $\overline{H}_*(X)$ is not a torsion group remain for future research to reveal.

2. SUMMARY OF RESULTS

In computer science, a "problem" is a function f from a subset of N, the nonnegative integers, to N. Sometimes the argument of this function, also called the "input," may be viewed naturally as a single integer, while at other times it may encode, through some fixed injection $\eta : \coprod_{j=1}^{\infty} Z^j \to N$, the finite description of some other object. (The map η is said to N-encode the finite description.) Regardless of the proper interpretation of the input, a computer scientist may imagine that a machine or algorithm exists which can accept an arbitrary $N \in \mathrm{Dom}(f)$ as input and deliver f(N) as output, utilizing in the process some number T(N) of steps. Given f, he or she may seek theoretical lower bounds for the function T(N) or seek algorithms (machines) on which T(N) exhibits a certain level of efficiency.

Computer scientists have developed a scale or continuum along which various problems may be placed according to their complexity. Certain classes of problems, including the classes known as 𝒫, #𝒫-complete, and "computable in exponential time," serve as landmarks. In particular, there is a transitive, reflexive relation on the set of problems, called "Turing reducible in polynomial time" and denoted $\underset{T}{\leq}$, by which f_2 is at least as hard as f_1 if $f_1 \underset{T}{\leq} f_2$. The problems f_1 and f_2 are "Turing equivalent," denoted $f_1 \underset{T}{\approx} f_2$, if if $f_1 \underset{T}{\leq} f_2$ and $f_2 \underset{T}{\leq} f_1$.

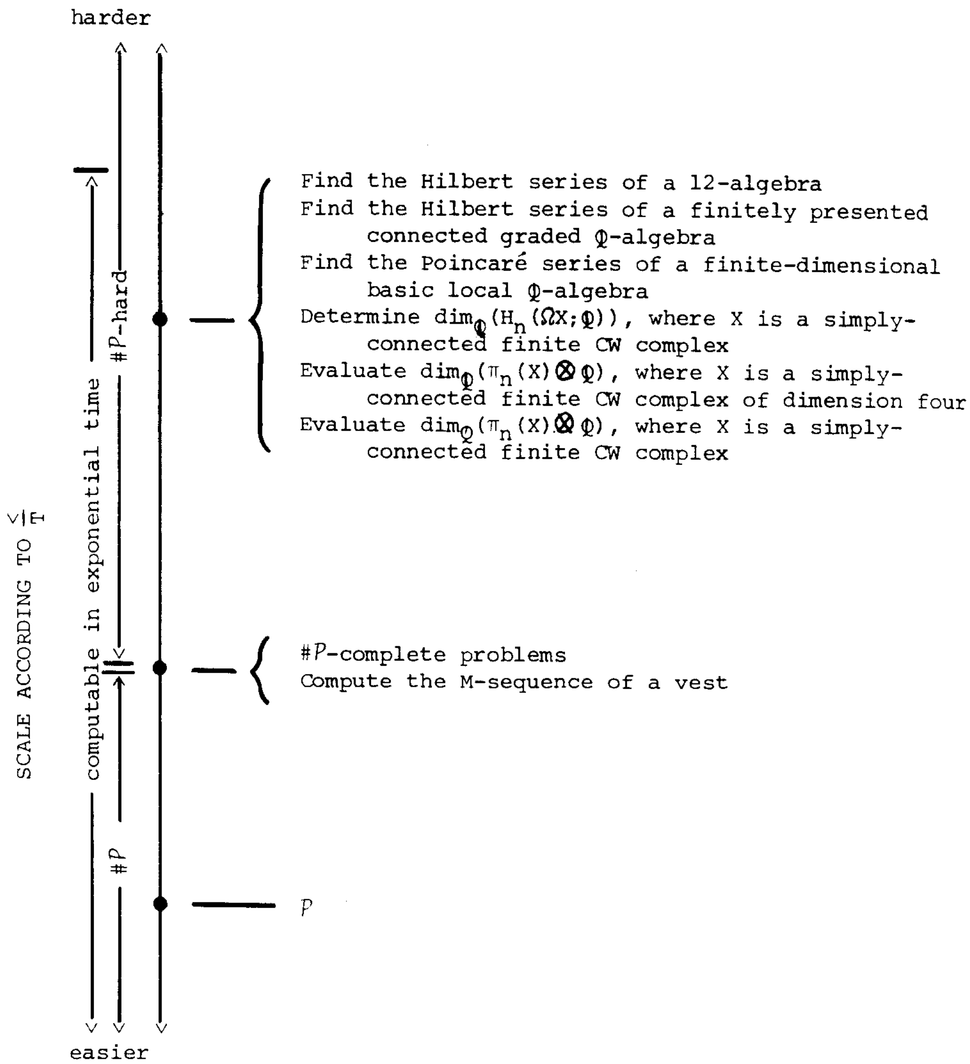

Figure 1 Diagram showing a portion of the computational complexity scale and the relative positions of three Turing equivalence classes.

The goal of this chapter is to locate the problem "compute rational homotopy" and some related problems on this continuum.

Our results are summarized in Figure 1. More difficult problems are placed higher on the scale, and Turing equivalent problems are bracketed. Problems at a specific difficulty level are marked by filled circles on the vertical axis. Classes of problems which encompass a range along the continuum are exhibited as intervals.

The purpose of Figure 1 is to give a visual overview of our results, and some technical points were sacrificed for crispness. The figure may be misleading in that the partial order $\underset{T}{\leq}$ is depicted as a linear order. For instance, one might conclude from the figure that any problem unsolvable in exponential time is $\#\mathcal{P}$-hard, but this has not been proved (and is probably false). Nor have we proved that the three Turing equivalence classes marked by filled circles must be actually distinct.

As to interpretation, bear in mind that the class $\#\mathcal{P}$-hard starts near the bottom of Figure 1, even though this class is viewed by computer scientists as being "very difficult." In other words, except for "$\mathcal{P}$," the section of the scale shown in Figure 1 actually starts far <u>above</u> such familiar computer mainstays as "solve a linear system of equations," "find the roots of a polynomial," or "factor the integer N."

There are no known algorithms which can evaluate a $\#\mathcal{P}$-hard problem in less than exponential time. Algorithms which require exponential time are generally thought of as being beyond the scope of today's machinery to implement efficiently. Thus computing rational homotopy groups and the other problems listed in Figure 1 may truly be described as very complex problems.

3. THE CONNECTION WITH HILBERT SERIES

The computational complexity of rational homotopy groups can best be discerned by studying certain closely related calculations which involve graded algebras. In this section we shall explore this connection. Finding the rational homotopy groups of a space is computationally equivalent to evaluating something we will call "Tor-sequences." These in turn contain as a subset the collection of "M-sequences." We close with some interesting examples of M-sequences in order to illustrate how complicated rational homotopy can be.

For the remainder of this chapter, a <u>space</u> will refer to a finite simply connected CW complex whose 1-skeleton is trivial.

In studying the computational complexity of homotopy groups, a certain technical sticking point arises immediately. As we have noted, $\pi_*(\cdot)$ may be viewed as a function whose inputs or arguments are an integer n and a description of a space X. How, exactly, does one describe a space?

One possible answer was adopted by Brown. He assumed that the description of X would consist of a simplicial decomposition, i.e., a list of the simplices for a simplicial complex X' having the same homotopy type as X. Such a list, however, is extremely long for any space of even moderate complexity. For example, several hundred simplices are involved in the smallest such description of $S^2 \times S^2$. Ideally, one would like to describe a CW complex in such a way that the length of the description is roughly comparable to the number of cells and/or the complexity of the various attaching maps.

Fortunately, in the case of a rational homotopy, Quillen's Lie algebra model ($\mathcal{L}X$, d_X) for a space X does provide such a description. Any space $X = * \cup (\cup_{i=1}^{h} e^{m_i})$ (assume $m_i \geq 2$) can be specified up to rational homotopy type by giving a nondecreasing list $(m_1, \ldots, m_h)$ of the degrees in which the cells occur followed by a description of the differential d_X. Since $\mathcal{L}X$ can be taken to be the free graded Lie Q-algebra on generators $\{x_1, \ldots, x_h\}$, where $\deg(x_i) = m_i - 1$, describing d_X amounts to giving m homogeneous elements of $\mathcal{L}X$. The ith entry in this list should have degree $m_i - 2$. Thus for rational homotopy calculations the <u>description</u> of a space will consist of a list $(m_1, \ldots, m_h)$ of cell dimensions, followed by a list $(d_X(x_1), \ldots, d_X(x_h))$ of boundaries. For example, when $S^2 \times S^2$ has its usual CW decomposition its description could be

$$(2, 2, 4); (0, 0, [x_1, x_2]) \qquad [2]$$

In Section 5 we specify a precise form for this description.

Three interrelated points need to be made. First, a valid description of a space is always obtainable from a simplicial decomposition, so we are justified in assuming that the description is available as input. Second, we shall want the input to be comparable in length to the "size" or "complexity" of X, and for this the above notion of description works well. Third, one can easily N-encode a string of symbols such as (2) into a natural number whose length (when written, say, in decimal) is bounded by a constant multiple of the total number of symbols in the original description.

We will henceforth view the problem of computing rational homotopy as the problem of using a valid Quillen model (equivalently, a valid description or N-encoded description) and an integer n to generate the integer.

$$r_n(X) = \dim(\pi_{n+1}(X) \otimes Q) \qquad [3]$$

The theorem which makes this feasible and which motivates the dimension shift in (3) is

Theorem 3.1 (Quillen)

If $(\mathcal{L}X, d_X)$ is the Quillen model for a space X, then $d_X^2 = 0$ and

$$H_q(\mathcal{L}X, d_X) \approx \pi_q(\Omega X) \otimes Q$$

Equivalently,

$$\pi_{q+1}(X) \otimes Q = \ker(d_X)_{(q)}/\mathrm{im}(d_X)_{(q+1)}$$

enclosed subscripts denoting graded components, or

$$r_n(X) = \dim(\ker(d_X)_{(n)}) - \dim(\mathrm{im}(d_X)_{(n+1)}) \qquad [4]$$

We shall see that it is easier to work with the Betti sequence of ΩX than with $\{r_n(X)\}$. The Betti sequence of ΩX is the sequence $\{b_n(\Omega X)\}$, where $b_n(\Omega X) = \dim(H_n(\Omega X; Q))$. By [22] these sequences are related by the formula

$$\sum_{i=0}^{\infty} b_i z^i = \prod_{j=1}^{\infty} \frac{(1 + z^{2j-1})^{r_{2j-1}}}{(1 - z^{2j})^{r_{2j}}} \qquad [5]$$

where $b_i = b_i(\Omega X)$ and $r_j = r_j(X)$. The left-hand side of (5) is called the Poincaré series of ΩX and will be denoted by $P_{\Omega X}(z)$.

By formula (5), knowing either one of $\{b_i\}_{i \leq n}$ or $\{r_j\}_{j \leq n}$ enables us to compute the other. Furthermore, the computation involved is a one which computer scientists would think of as being executable "quickly." "Fast" calculations are those which require only a polynomial number of steps. Let us see why this computation qualifies.

To obtain $\{b_i\}_{i\leq n}$ when one has the list $\{r_i\}_{i\leq n}$, one can view (5) as a congruence modulo z^{n+1}. To evaluate the right-hand side of (5) requires that we multiply together the polynomials

$$(1+z)^{r_1},\ (1+z^2+z^4+\cdots+z^{\bar{n}})^{r_2},\ (1+z^3)^{r_3},\ (1+z^4+z^8+\cdots+z^{\bar{n}})^{r_4},\ldots \qquad [6]$$

where "$z^{\bar{n}}$" is used loosely to denote "stop after z^n." To multiply together two polynomials of degree n (modulo z^{n+1}) takes at most $1 + 2 + \cdots + (n + 1)$ multiplications and $0 + 1 + \cdots + (n)$ additions, which we summarize as $(n + 1)^2$ operations. To raise anything to the power r requires at most $2\cdot\log_2(r)$ multiplications; to see this, write r in binary and use the trick that $x \to x^2 \to x^4 \to \cdots \to x^{2^m}$ takes only m multiplications. Thus at most

$$(n+1)^2\,(2)\,[\log_2(r_1) + \log_2(r_2) + \cdots + \log_2(r_n)]$$

operations are needed in order to evaluate the entries of the list (6), and at most $(n + 1)^2(n - 1)$ further operations are involved in forming their product (modulo z^{n+1}).

Finally, one sees easily for each space X that $\{r_j(X)\}$ grows at most exponentially with j, that is, $\log_2(r_j) \leq Kj$ for some fixed K. [An upper bound for K is $\log_2(m)$ if X has m cells.] Evaluating the entire right-hand side of (5) takes at most

$$(n+1)^2\,(n-1) + (n+1)^2\,(2)\,K\left(\frac{n^2+n}{2}\right)$$

operations. Since this is a polynomial in n, it justifies the description of this calculation as "fast."

Likewise, one can "quickly" obtain $\{r_j\}_{j\leq n}$ from knowledge of $\{b_i\}_{i\leq n}$. The problem of computing $\{r_j\}_{j\leq n}$ and the problem of computing $\{b_i\}_{i\leq n}$ are viewed as being "equivalent" to one another. We postpone the precise definition of this equivalence until Section 5. For the time being, we hope the reader is convinced that it suffices to study the complexity of $\{b_n(\Omega X)\}$ in order to understand the problem of computing $\{r_n(X)\}$.

x. For the purposes of implementation, there are more effi-
s to multiply two polynomials than the naive way analyzed
e [24, chapters 4 and 36] for an excellent discussion of these
rithms.

termine $\{b_i(\Omega X)\}_{i\le n}$ is the same as to determine the Poincaré
ΩX modulo z^{n+1}. This in turn is equivalent to evaluating the
fficients of the Hilbert series of a certain graded algebra.
ore detour to introduce some terminology related to graded al-

s chapter, a graded algebra is a connected N-graded finitely
d Q-algebra A. Such an object always has a presentation as

$$Q\langle x_1,\dots,x_g\rangle/\langle\alpha_1,\dots,\alpha_r\rangle \qquad [7]$$

$\langle x_1,\dots,x_g\rangle$ is the free associative Q-algebra on $\{x_1,\dots,x_g\}$,
N-graded by assigning a positive integral degree $|x_i|$ to each
notation $\langle\alpha_1,\dots,\alpha_r\rangle$ designates the two-sided ideal of $Q\langle x_1,$
generated by the finite set of homogeneous relations $\{\alpha_1,\dots,$
nce $\langle\alpha_1,\dots,\alpha_r\rangle$ is homogeneously generated, the quotient alge-
herits a gradation from $Q\langle x_1,\dots,x_g\rangle$ and we may write A =
The Hilbert sequence of A is the sequence $\{h_j(A)\}$, where
$\dim_Q(A_j)$, and the Hilbert series is $H_A(z)=\sum_{j=0}^{\infty}h_j(A)z^j$.
ne-two algebra (or 12-algebra) is a graded algebra A which has
ntation (7) such that each $|x_i| = 1$ and each $|\alpha_k| = 2$. A 123-
a is a 12-algebra which has global dimension three. A 12H-alge-
a 12-algebra which is also a Hopf algebra. This is equivalent to
ndition that each α_k be a Q-linear combination of $(x_ix_j + x_jx_i)$'s
)'s. These were called Roos algebras in [3]. Lastly, a 123H-
a is a 12H-algebra of global dimension three.

e connection between graded algebras and arbitrary spaces is
in [6]. By Theorem 1 of [6] there exists for every space X a
algebra A such that $H_A(z)$ and $P_{\Omega X}(z)$ are rationally related.
eans that there exist polynomials $p_i(z)\in Z[z]$, $1\le i\le 4$, such
$p_1p_4 \ne p_2p_3$ and

$$P_{\Omega X}(z) = \frac{p_1(z)H_A(z)+p_2(z)}{p_3(z)H_A(z)+p_4(z)} \qquad [8]$$

Conversely, for any 12-algebra A, there exists a space X of dimension four for which (8) holds.

Formula (8) implies that the problem of computing $\{b_i(\Omega X)\}_{i\leq n}$ and the problem of determining $\{h_j(A)\}_{j\leq n}$ are equivalent, in the sense that one could obtain either list from the other after only q(n) additional operations for some fixed polynomial q. To see this, note that in passing from H_A to $P_{\Omega X}$ the only possible trouble spot is the inversion of $P_3(z)H_A(z) + p_4(z)$. By the proof of Theorem 1 of [6], however, we can assume that $p_3(z)H_A(z) + p_4(z) \equiv 1 (\mathrm{mod}\ z)$ and we may write

$$p_3(z)H_A(z) + p_4(z) = 1 - z\cdot u(z), \qquad u(z) \in Z[z]$$

Then $(1 - zu(z))^{-1} = 1 + zu(z) + z^2u(z)^2 + \cdots$ and

$$(1 - zu(z))^{-1} \equiv \theta^n(1) \ (\mathrm{mod}\ z^{n+1})$$

where θ is the endomorphism of Z[z] defined by $\theta(x) = 1 + zu(z)x$. Since evaluating $\theta(x)\ (\mathrm{mod}\ z^{n+1})$ takes up to $(n + 1)^2$ operations, we have inverted $p_3(z)H_A(z) + p_4(z)$ in only $n(n + 1)^2$ operations.

Thus the general problem of computing $\{r_j(X)\}_{j\leq n}$ is equivalent to the problem of finding $\{h_j(A)\}_{j\leq n}$ for a certain 123H-algebra A associated to X.

Remark. In practice, there are more efficient ways to take the quotient of two power series. See, for example, Sections 4.6.1 and 4.7 of [18].

We perform just one more reduction. Because gl.dim(A) = 3, its Hilbert series can be expressed neatly in terms of another series, which we will call its "Tor-series." Specifically, note that $\mathrm{Tor}_3^A(Q, Q)$ is a graded Q-module because A is graded, and let $c_q = c_q(A) = \dim(\mathrm{Tor}^A_{3,q+3}(Q, Q))$. The Tor-sequence of A is $\{c_q\}_{q\geq 0}$ and the Tor-series of A is $T_A(z) = \sum_{q=0}^{\infty} c_q z^q$. Because A has global dimension three,

$$H_A(z)^{-1} = 1 - gz + rz^2 - z^3T_A(z) \tag{9}$$

Here g and r count the numbers of generators and relations in a minimal presentation (7) for A.

On the basis of the previous remarks about multiplying and inverting power series, we assert without further proof that computing $\{r_i(X)\}_{i\leq n}$ is equivalent to computing the Tor-sequence $\{c_q\}_{q\leq n-3}$ of a certain 123H-algebra A associated to X.

We have gotten rather far afield from rational homotopy groups, but Tor-sequences are a good place to stop. Tor-sequences capture what is intrinsically complicated about rational homotopy but express that intricacy in more accessible algebraic or combinatorial terms. A further advantage of Tor-sequences is that we may more readily construct bizarre or amusing examples to illustrate their diversity. Lastly, the class contains a special subset, to be called "M-sequences," which is natural in the following sense. The general problem of "compute an M-sequence" is #℘-complete, an important computer science concept.

To summarize what we have shown so far, we have

Proposition 3.2

For every space X there exists an associated 123H-algebra A with the following property. Given the first n terms of any one of the following sequences, one can with $\tau(n)$ additional operations compute the first n terms of any other sequence, where $\tau(n)$ is a polynomial in n:

1. The rational homotopy ranks $r_j(X) = \dim(\pi_{j+1}(X) \otimes Q)$
2. The Betti sequence $b_i(\Omega X) = \dim(H_i(\Omega X; Q))$
3. The Hilbert sequence $h_j(A) = \dim(A_j)$
4. The Tor-sequence $c_i(A) = \dim(\mathrm{Tor}^A_{3,3+i}(Q, Q))$

Conversely, given any 123-algebra A, there exists a space X with dim(X) = 4 for which the same conclusion holds.

We will now jump right in with the definition of an M-sequence. For motivation, the reader is welcome to glance ahead to Theorem 3.4.

Definition 3.3

A vector evaluated after a sequence of transformations, henceforth vest, is a four-tuple (d, v_0, T, S) as follows. The first entry d is any positive integer, and v_0 is any vector, called the initial vector, in Q^d. The third entry T denotes a list $T_1, \ldots, T_m$ of $d \times d$ matrices over Q viewed as linear transformations on Q^d, and S is any h×d matrix over Q. Given a vest (d, v_0, T, S) and a length n sequence of indices $\sigma = (i_1,\ldots,i_n)$ having $1 \le i_j \le m = \#(T)$ define $v_0 * \sigma$ to be the vector

$$v_0 * \sigma = T_{i_n} \cdots T_{i_2} T_{i_1}(v_0) \in Q^d$$

Let $M_n = \{\sigma = (i_1,\ldots,i_n) \mid S(v_0 * \sigma) = 0\}$ and let $e_n = \#(M_n)$. The <u>M-sequence</u> for (d, v_0, T, S) is $\{e_n\}_{n \ge 0}$ and its <u>M-series</u> is the formal power series $M(z) = \sum_{n=0}^{\infty} e_n z^n$. An arbitrary formal power series M(z) is an <u>M-series</u> if and only if it equals the M-series of some vest.

Note that a vest can be specified by listing an integer d, d rational numbers, an integer m, md^2 more rational numbers, and the integer h followed by hd rationals. Note that the M-sequence is not affected if v_0 or S or any T_i is multiplied by a nonzero scalar, so no harm is done by clearing denominators and assuming that all entries are integers. A vest can therefore be thought of as being specified by a list of integers, the length of the list being $1 + d + 1 + md^2 + 1 + hd$.

The motivation for considering Definition 3.3 lies in the following theorem, which translates Theorem 1.3 of [3] into the language of that definition.

Theorem 3.4

Let (d, v_0, T, S) be a vest and let M(z) be its M-series. Then there exists a 123H-algebra A, with $g = 2m + d + h + 3$ generators and $r = (m+1)(m+d+h+2)+1$ relations, whose Tor-sequence equals M(z). In other words, every M-sequence is a Tor-sequence.

Since every M-sequence is a Tor-sequence, Tor-sequences must be at least as difficult to compute, in general, as M-sequences. Since Tor-sequences are comparable in computational complexity to rational homotopy groups, rational homotopy is at least as hard to calculate as M-sequences. We shall see in the next section that general M-sequences are as hard as or harder to compute than any problem belonging to a large class called #𝒫. The remainder of this section is devoted to some examples of M-sequences which we hope will illustrate how wide a class of functions they encompass. It can be skipped by the reader without loss of continuity.

Theorem 3.5

Fix positive integers m and h. Let $\mathfrak{a}$ denote the m-tuple $(a_1,\ldots,a_m) \in N^m$. For $1 \le i \le h$ let $E_i(n,\mathfrak{a})$ be a Q-linear combination of expressions of the form

$$c\, n^{d_0} a_1^{d_1} \cdots a_m^{d_m} \qquad [10]$$

where $c \in Q - \{0\}$ and $d_j \in N$. Let $B_1,\ldots,B_m \in N - \{0,1\}$ be arbitrary and put

$$I_n = \bigcap_{i=1}^{h} \{\mathfrak{a} = (a_1,\ldots,a_m) \in N^m \mid E_i(n,\mathfrak{a}) = 0 \text{ and } 0 \le a_j < B_j^n \text{ for } 1 \le j \le m\}$$

Then the list of cardinalities $\{\#(I_n)\}_{n \ge 0}$ is an M-sequence.

The proof is postponed until after two motivating examples are given.

<u>Example 3.6</u>. The sequence $e_n = [(3/2)^n]$, brackets denoting the greatest integer function, is an M-sequence.

<u>Proof</u>. Consider the single equation

$$E(a_1, a_2) = 2^n(a_1 + 1) + a_2 - 3^n = 0 \qquad [11]$$

and put $B_1 = 2$, $B_2 = 3$. Nonnegative integral solutions to (11) which satisfy the bounds $0 \le a_1 < 2^n$ and $0 \le a_2 < 3^n$ are in one-to-one correspondence with

$$\{a_1 \in N \mid 2^n(a_1 + 1) \le 3^n\}$$

This set has cardinality e_n.

<u>Example 3.7</u>. The sequence $\{e_n\}$, where e_n equals the nth digit in the decimal representation of $\sqrt{2}$, is an M-sequence.

<u>Proof</u>. Consider the system of h = 4 equations:

$$E_1(n,\mathfrak{a}) = (10a_1 + a_2)^2 + a_3 - 2(100)^n = 0$$

$$E_2(n,\mathfrak{a}) = (10a_1 + a_2 + 1)^2 - a_4 - 2(100)^n = 0$$

$$E_3(n,\mathfrak{a}) = a_2 + a_5 - 10 + 1 = 0$$

$$E_4(n,\mathfrak{a}) = a_6 + a_7 - a_2 + 1 = 0$$

and seek simultaneous solutions subject to the bounds $B_3 = B_4 = 100$, $B_1 = B_2 = B_5 = B_6 = B_7 = 10$.

Putting $x = 10a_1 + a_2$, the first two equations say that

$$x^2 \leq 2(10)^{2n} \leq (x+1)^2$$

so any solution has $x = [10^n \sqrt{2}]$. The third equation assures us that $0 \leq a_2 < 10$, which forces a_2 to equal the last decimal digit of x; clearly this digit is e_n. So far, there is a unique solution for (a_1, a_2, a_3, a_4, a_5). The fourth equation permits the total number of simultaneous solutions for $\mathfrak{a}$ to equal $a_2 = e_n$. Thus $\#(I_n) = e_n$, as desired.

Proof of Theorem 3.5. Given a system of expressions $\{E_j(n,\mathfrak{a})\}$ in which each $E_i(n,\mathfrak{a})$ is assumed to be a linear combination of terms (10), we want to construct a vest (δ, v_0, T, S) whose M-sequence measures the number of suitably bounded solutions to "$E(n,\mathfrak{a}) = 0$." We will do this by letting T consist of $B_1B_2\cdots B_m$ linear transformations denoted T_λ, where λ runs through the set of m-tuples

$$\Lambda = \{\lambda = (\lambda_1,\ldots,\lambda_m) \in N^m \mid 0 \leq \lambda_j < B_j\}$$

We set up a bijection $\mathcal{G}$ between the set Λ^n of length n sequences $\sigma = (\lambda(1),\ldots,\lambda(n))$ and the set of m-tuples

$$\mathscr{A}(n) = \{\mathfrak{a} = (a_1,\ldots,a_m) \in N^m \mid 0 \leq a_j < B_j^n\}$$

in the following manner. The $(n-i)$th digit, in base B_j, of a_j will be the jth entry of the m-tuple $\lambda(i)$. The m-tuple $\mathfrak{a}$ obtained in this way from a sequence $\sigma \in \Lambda^n$ will be denoted $\mathcal{G}(\sigma)$.

Let $L_1(n,\mathfrak{a}),\ldots,L_t(n,\mathfrak{a})$ be a complete list of all the terms (10) which appear in any of the $E_i(n,\mathfrak{a})$'s. For $1 \leq j \leq t$ we claim the existence of a vector space V_j and a $v_j \in V_j$, together with an action of each T_λ on V_j, such that for any $\sigma = (\lambda(1),\ldots,\lambda(n)) \in \Lambda^n$ the first component of the vector $v_j * \sigma$ equals $L_j(n,\mathcal{G}(\sigma))$.

Granting this claim, put $V = \oplus_{j=1}^{t} V_j$ and $v_0 = (v_1,\ldots,v_t) \in V$. Then each $E_i(n,\mathcal{G}(\sigma))$ equals a linear combination of certain components of $v_0 * \sigma$. The matrix S may be chosen so that the inner product of the ith row of S with $v_0 * \sigma$ equals $E_i(n,\mathcal{G}(\sigma))$. Then

$$S(v_0 * \sigma) = E(n,\mathcal{G}(\sigma))$$

in Q^h. As σ runs through Λ^n, $\mathcal{G}(\sigma)$ runs through $\mathscr{A}(n)$ exactly once. Thus $\#(I_n) = \#\{\sigma \mid S(v_0 * \sigma) = 0\}$, proving $\{\#(I_n)\}$ to be the M-sequence for the vest (dim(V), v_0, T, S).

We will prove the claim by induction on $d = d_0 + d_1 + \cdots + d_n$, where in keeping with (10) we write $L_j(n,\mathfrak{a}) = c^n n^{d_0} a_1^{d_1} \cdots a_m^{d_m}$. If $d = 0$ then $d_0 = d_1 = \cdots = d_m = 0$ and $L_j(n,\mathfrak{a}) = c^n$, so we may take $V_j = Q$ with $v_j = 1$ and take every T_λ to be multiplication by c.

Assuming the claim has been proved for exponent sums less than d, suppose that $d_0 + \cdots + d_m = d > 0$. For $\lambda = (\lambda_1,\ldots,\lambda_m) \in \Lambda$ and $\mathfrak{a} = (a_1,\ldots,a_m) \in \mathscr{A}(n)$, let $\mathfrak{a} * \lambda$ denote

$$\mathfrak{a} * \lambda = (B_1a_1 + \lambda_1, B_2a_2 + \lambda_2, \ldots, B_ma_m + \lambda_m)$$

The binomial formula applied to $L_j(n+1, \mathfrak{a} * \lambda)$ shows

$$L_j(n+1, \mathfrak{a} * \lambda) = (cB_1^{d_1} \cdots B_m^{d_m}) L_j(n,\mathfrak{a}) + E_j'(n,\mathfrak{a})$$

where $E_j'(n,\mathfrak{a})$ is a linear combination of expressions of the form (10) whose exponent sums are smaller than d. By our inductive hypothesis there exists a vector space V_j' on which transformations T_λ' act and an initial vector v_j' for which

$$S_j'(v_j' * \sigma) = E_j'(n,\mathcal{G}(\sigma))$$

for some row vector S_j'. Let $V_j = Q \oplus V_j'$ with

$$v_j = \begin{bmatrix} 0 \\ v_j' \end{bmatrix}$$

and let T_λ be the matrix

$$\begin{bmatrix} c_j & S'_j \\ 0 & T'_\lambda \end{bmatrix}$$

where $c_j = cB_1^{d_1} \cdots B_m^{d_m}$. An induction on n shows that the first component of $v_j * \sigma$ is indeed $L_j(n, \mathcal{G}(\sigma))$ for $\sigma \in \Lambda^n$. This completes the proof.

Remark. For further interesting examples of M-sequences the reader is referred to Corollary 2.5 of [3].

4. COMPUTING M-SEQUENCES IS # $\mathcal{P}$-COMPLETE

Section 4 is the technical core of the chapter, and it contains two major results. We first review the definitions of the terms $\mathcal{P}$, #$\mathcal{P}$, and #$\mathcal{P}$-complete. These are classes of problems which are solvable by various sorts of Turing machines. The first major result, Theorem 4.5, asserts that computing the M-sequence of a vest (see Definition 3.3) is #$\mathcal{P}$-complete. We indicate precisely how a vest should be N-encoded for this result to be true. The second major theorem, Theorem 4.8, proves a wide class of functions to be M-sequences. Membership in this class is contingent on computability by a $K\cdot(n)^{\varepsilon}$-time algorithm, a criterion satisfied by many familiar functions.

We begin by reviewing the concept of a Turing machine, for which there are several equivalent formulations. We adopt a variation on the description found in [26, p. vii], which is recalled next.

> Let there be given a tape of infinite length [in both directions] which is divided into squares and a finite list of symbols which may be written on these squares. There is an additional mechanism, the head, which may read the symbol on a square, replace it by another or the same symbol and move to the adjoining square to the left or right. This is accomplished as follows: At any given time the head is in one of a finite number of internal states. When it reads a square it prints a new symbol, goes into a new internal state and moves to the right or left depending on the original internal state and the symbol read. Thus a Turing machine is described by a finite list of quintuplets such as 3,4,3,6,R which means: If the machine is in the third internal state and reads the fourth symbol it prints the third symbol, goes into the sixth internal state and moves to the right on the tape.

Let U denote the (finite) set of internal states. We specify an initial state $u^* \in U$ and two possible "terminal states" u_0 and u_1. Let $\hat{U} = U - \{u_0, u_1\}$ consist of the nonterminal states.

The initial configuration of the tape is assumed to consist of a single contiguous finite segment of nonblank squares with the head initially positioned immediately to the right of the rightmost nonblank symbol. This contiguous segment is the input I, whose length, denoted $\ell(I)$, is the number of nonblank squares. Without loss of generality we identify the set of tape symbols with $[B] = \{0,1,2,\ldots,B-1\}$ for some integer $B \geq 4$, with "zero" being viewed as the "blank." The next B_0 symbols $(2 \leq B_0 \leq B-2)$ are viewed as the digits 0 through $B_0 - 1$ in some base B_0, and among the remaining symbols we reserve one, called "semicolon," to delimit various segments of the input.

As noted above, the Turing machine itself consists of a collection Γ of quintuplets,

$$\Gamma \subseteq \hat{U} \times [B] \times [B] \times U \times \{+1,-1\}$$

where we are assuming that computation ceases if ever the machine attains a terminal state. Thus Γ may be viewed as a relation between the sets $\hat{U} \times [B]$ and $[B] \times U \times \{\pm 1\}$. In an ordinary deterministic Turing machine, the relation Γ is a function, and each configuration in $\hat{U} \times [B]$ leads to exactly one successive configuration (as indicated by an element of $[B] \times U \times \{\pm 1\}$). Deterministic machines embody the usual concept of predictable, reproducible, serial calculation. For a given input there is just one possible flow path through the various configurations.

On the other hand, a nondeterministic Turing machine is one for which the set Γ need not be a function. We do assume, for each $(u,b) \in \hat{U} \times [B]$, that there is at least one $(b',u',z') \in [B] \times U \times \{\pm 1\}$ such that $(u,b,b',u',z') \in \Gamma$. For each input I there could be many possible flow paths which a nondeterministic machine could follow, and some paths may end in each of the two terminal states. Nondeterministic machines are different from but closely related to probabilistic machines, which are computers that can incorporate the output of a random number generator into their branching decisions.

Definition 4.1

A function $f : \mathrm{Dom}(f) \to N$, $\mathrm{Dom}(f) \subseteq N$, is in the class $\underline{\mathcal{P}}$ if and only if there exists a deterministic Turing machine, together with a polynomial $\tau(x)$, with the following property. Whenever the input is $N \in \mathrm{Dom}(f)$, the Turing machine will attain the terminal state u_0 after at most $\tau(\ell(N))$ steps, and before attaining the state u_0 the output $f(N)$ will be printed on the tape. Without loss of generality we assume that the machine terminates only after repositioning the head to the right of the rightmost nonblank square.

Example. The problem of multiplying two numbers is in $\mathcal{P}$. Given N_1 and N_2, we may N-encode the pair (N_1,N_2) as

$$I = (N_1)_{(10)};(N_2)_{(10)}$$

where $(N)_{(10)}$ denotes the base 10 representation of the number N. (For convenience we have taken $B_0 = 10$.) By way of illustration, notice that to multiply $N_1 = 11374$ by $N_2 = 1286$ takes $5 \cdot 4 = 20$ individual multiplication operations followed by a comparable number of additions. The total number of steps needed is on the order of $\ell(N_1) \cdot \ell(N_2) < \ell(I)^2$.

Remark. This example illustrates that, when an n-tuple of inputs $(N_1,\ldots,N_n)$ is N-encoded as $I = (N_1)_{(10)};(N_2)_{(10)};\ldots;(N_n)_{(10)}$ then a machine runs in $\tau(\ell(I))$ time for a polynomial $\tau(x)$ if and only if it runs in $\mu(\ell(N_1),\ldots,\ell(N_n))$ time for some polynomial of n variables μ. What if an algorithm requires, say, $N_1^2 \cdot \ell(N_2)^3$ steps? We wish to express the polynomial dependence on N_1 (not $\ell(N_1)$) and $\ell(N_2)$. To do this, we can redefine the problem so that the input consists of N_1 written in unary, followed by a semicolon and $(N_2)_{(10)}$. By "unary" we mean a string of N_1 1's to represent the number N_1, which we denote by $(N_1)_{(1)}$. With this redefinition we would have

$$I = (N_1)_{(1)};(N_2)_{(10)}$$

and consequently $\ell(I) = \ell((N_1)_{(1)}) + 1 + \ell((N_2)_{(10)}) = N_1 + 1 + \ell(N_2)$, hence $\ell(I)^5$ bounds the run time of $N_1^2\,\ell(N_2)^3$. Thus the problem "compute $f(N_1,N_2)$ from the input $I = (N_1)_{(1)};(N_2)_{(10)}$" belongs to $\mathcal{P}$, even though the problem "compute $f(N_1,N_2)$ from the input $I = (N_1)_{(10)};(N_2)_{(10)}$" does not. The only thing that has changed is the N-encoding used for the input. Because of this we will be very careful to spell out the N-encodings used when setting up a problem.

Definition 4.2

A function $f : \mathrm{Dom}(f) \to N$, $\mathrm{Dom}(f) \subseteq N$, belongs to the class $\underline{\#\mathcal{P}}$ if and only if there exists a nondeterministic Turing machine, together with a polynomial $\tau(x)$, with the following property. Whenever the input is $N \in \mathrm{Dom}(f)$, each of the possible paths taken by the machine reaches a terminal state within $\tau(\ell(N))$ steps. The number of possible paths which lead to the terminal state u_1 (as opposed to u_0) equals $f(N)$. Paths leading to the state u_1 are called accepting paths.

Remark. It is important that $\tau(x)$ be universal with respect to the set of paths. That is, there is a single polynomial bound $\tau(\ell(N))$ such that any possible path uses fewer than this many steps.

Example. The function d, where d(N) denotes the number of positive integral divisors of N, belongs to #$\mathcal{P}$. To see this, consider the following three-stage "nondeterministic program." First, write down any two numbers N_1 and N_2 of length $\le \ell(N)$. To do this, the machine writes the first decimal digit, then the second decimal digit, and so on. Nondeterministic branching is involved at this point because, after writing a digit and comparing the current length with $\ell(N)$ and repositioning the head, it may enter any of $B_0 = 10$ states to get ready for the next digit. At the second stage multiply N_1 and N_2 to obtain a result N_3. Lastly, compare N and N_3; if equal enter state u_1, if unequal enter state u_0.

The ith stage of this program clearly takes at most $\tau_i(\ell(N))$ steps for some polynomials τ_i, so all paths terminate within $\tau(\ell(N))$ time if $\tau = \tau_1 + \tau_2 + \tau_3$. Furthermore, paths are classified by the pair (N_1, N_2) written during stage one, since the remainder of the program flows deterministically. The number of paths leading to state u_1 equals the number of pairs (N_1,N_2) such that $N_1N_2 = N$. Thus the number of accepting paths equals $d(N)$, as needed in Defintion 4.2.

Note. In the above argument it is implicitly assumed that "pairs of numbers written down during stage one" are in one-to-one correspondence with pairs of numbers (N_1,N_2) whose product N_3 is a candidate to equal N. This is correct except that leading zeros can create a problem; for instance, when N = 30 the two pairs (6,5) and (06,05) should not both be allowed. Stage one of the program must either forbid leading zeros or else require that N_1 and N_2 have length equal to exactly $\ell(N)$. This issue recurs when we are modeling a Turing machine via a vest in the proof of Theorem 4.5. In that proof we adopt the approach of insisting that the lead digit be nonzero.

Another example of a problem in $\#\mathcal{P}$ is the computation of the permanent of a square matrix whose entries lie in N [28]. The $\#\mathcal{P}$ problem with greatest relevance to the computational complexity of rational homotopy is described next.

Theorem 4.3

Computing the M-sequence of a vest is in $\#\mathcal{P}$. Specifically, let $f \subseteq N \times N$ be the function which takes as input the list of $1 + d + 1 + md^2 + 1 + hd$ integers (each written in decimal) describing a vest (d,v_0,T,S) followed by an integer n written in unary. The value of the function is the nth entry in the M-sequence of (d,v_0,T,S). Then f belongs to $\#\mathcal{P}$.

Remark. By the comments preceding Theorem 3.4, we know that a vest may be specified by a list of integers. Since these integers may be positive or negative or zero, however, it is useful to assume that a minus sign is included in the symbol set [B] available to our Turing machine.

Proof of Theorem 4.3. Consider the following program for a nondeterministic Turing machine. By successively writing down its decimal digits choose any index i satisfying $1 \le i \le m$. Then, (deterministically) compute T_iv_0 and replace the vector v_0 by the new value T_iv_0. Repeat these two steps exactly n times. Then (deterministically) compute Sv_0 and compare the result with the zero vector in Z^h. If it is zero, enter state u_1; otherwise, enter state u_0.

The only nondeterministic part of the program concerns choosing the various indices, which we call i_1 through i_n. Permissible flow paths correspond to n-tuples $\sigma = (i_1,\ldots,i_n)$. A path ends in state u_1 if and only if it corresponds to a σ satisfying $S(v_0 * \sigma) = 0$. Thus the number of accepting paths equals e_n, the nth entry of the M-sequence for (d, v_0,T,S).

It remains to find a polynomial τ such that $\tau(\ell(I))$ bounds the run time regardless of the path taken. Let q denote the maximum absolute value of the inputs which are encoded in decimal. By induction on j, the entries of the vector $(T_{i_j} \cdots T_{i_1})(v_0)$ are bounded in absolute value by $d^{j-1}q^j$. During the jth pass through the loop we choose and perhaps copy one of m matrices, introducing bookkeeping on the order of $K_1md^2\ell(I)$ steps. We then multiply a $d \times d$ matrix, each of whose entries is bounded by q, by a $d \times 1$ vector, each of whose entries is bounded by d^nq^n. This requires d^2 multiplications and about d^2 additions, each of which needs at most $K_2 \log(d^nq^n) = K_2n\log(dq)$ steps. Each pass through the loop uses on the order of $K_1md^2\ell(I) + K_3nd^2 \log(dq)$ steps, so the loop takes $K_1md^2n\ell(I) + K_3n^2d^2\log(dq)$ steps. The final stage of the program, namely multiplication by S and comparison with zero, requires another $K_4hdn\log(dq)$ steps, so everything is certainly finished in $K_1md^2n\ell(I) + K_5nd(nd + h)\log(dq)$ steps. But notice that the numbers m, d, and h are all smaller than $\ell(I)$ because the input contains, if nothing else, $2 + d + md^2 + dh$ semicolons or separators. The number n is smaller than $\ell(I)$ because n is written in unary, and $\log(q) < \ell(I)$ because q is written in decimal. Thus all paths terminate in $K \cdot \ell(I)^5$ steps, as desired.

The function f described in Theorem 4.3 is actually a very special kind of $\#\mathcal{P}$ problem. It is a "universal example" for the class $\#\mathcal{P}$ in the sense that any $g \in \#\mathcal{P}$ can be factored through f, the other factor belonging to the class $\mathcal{P}$. This property, called "$\#\mathcal{P}$-completeness," provides a kind of lower bound on the computational complexity of f. A precise definition follows.

Definition 4.4

Let $f : \mathrm{Dom}(f) \to N$, $\mathrm{Dom}(f) \subseteq N$, be any function. The function f is $\#\mathcal{P}$-complete if $f \in \#\mathcal{P}$ and, for any $g \in \#\mathcal{P}$, there exists a function $\phi_g \in$

$\mathcal{P}$ such that, whenever $N \in \mathrm{Dom}(g)$, then $\phi_g(N) \in \mathrm{Dom}(f)$ and $f(\phi_g(N)) = g(N)$.

The function ϕ_g occurring in Definition 4.4 is called a "polynomial time many-one reduction," a phrase which happily we will use only once again. One may think of it as a preprocessor which quickly transforms or translates (in the sense of translating a language) the original input N into an $I = \phi_g(N)$ which is suitable for use by f. The point is that <u>any</u> $g \in \#\mathcal{P}$ can, modulo a "fast" translation, be viewed as a special case of f. In other words, a $\#\mathcal{P}$-complete function f is so powerful that it already encompasses, in thinly disguised form, <u>every</u> $\#\mathcal{P}$ problem. On top of that, it is itself in $\#\mathcal{P}$!

Remarkably, the class $\#\mathcal{P}$ does contain universal examples of this type. The first example to be discovered was the problem of computing the permanent of a matrix [28]. We claim next that the problem of computing an M-sequence also satisfies Definition 4.4.

Theorem 4.5

Computing the M-sequence of a vest is $\#\mathcal{P}$-complete. Specifically, the function f described in the statement of Theorem 4.3 is $\#\mathcal{P}$-complete.

<u>Proof</u>. Theorem 4.3 asserts that this f is in $\#\mathcal{P}$, so it remains only to verify the universal property. Let $g : \mathrm{Dom}(g) \to N$ belong to $\#\mathcal{P}$. We must construct $\phi_g \in \mathcal{P}$ such that $g = (f \circ \phi_g)|_{\mathrm{Dom}(g)}$.

We really have very little to work with. The function g belongs to $\#\mathcal{P}$. Therefore there exists a nondeterministic Turing machine which, when given N Dom(g) as input, has all paths terminate in $\tau(\ell(N))$ steps, while the number of paths terminating in state u_1 equals $g(N)$. Let U, [B], Γ denote respectively the set of states, the symbol set, and the collection of transition quintuples for this Turing machine.

Here's the key idea. We will model the Turing machine $(U, [B], \Gamma)$ by a vest. The list of successive states experienced by the Turing machine will correspond to a sequence σ of linear transformations for the vest. A sequence σ which either (i) corresponds to a flow path which terminates in state u_0 or (ii) corresponds to an invalid flow path (i.e., some transitions not in Γ) will result in $S(v_0 * \sigma)$ being nonzero. On the other hand, a sequence σ which ends in state u_1 and which utilizes only valid transitions to get there will lead to $S(v_0 * \sigma)$ being zero. Thus the M-sequence entry e_n will be counting the number of valid paths ending in u_1, a number which equals $g(N)$.

The input I to the function f consists of a vest description along with a unary integer n. This N-encoded I is to be the output of the preprocessor ϕ_g. As it turns out, we may always take d = 16, and v_0 has fixed components except for two entries which equal N and N^2. So far, computing and writing down this much of the input takes $K_1 + K_2 \cdot \ell(N)^2$ time. The remainder of the vest description depends solely on g, so the algorithm for ϕ_g may "memorize" it and write it down in a constant number K_3 of steps. Lastly, the argument n may be taken to be any integer greater than $\tau(\ell(N)) \cdot (\tau(\ell(N)) + \ell(N) + 1)$. To compute this number and to write it down in unary also takes a polynomial in $\ell(N)$ steps. Thus ϕ_g runs in polynomial time.

We will now describe explicitly how a vest can model the nondeterministic Turing machine $(U, [B], \Gamma)$ with input N, so that the M-sequence entry e_n equals $g(N)$ when $n \gg 0$. As noted above, the dimension d of the vector space involved in the vest will always be d = 16. The number m of transformations will be $\#(T) = \#(\Gamma) + 1 + 4B$. And h will equal 1.

To describe the m transformations it is useful to borrow from computer science the concept of "registers." A register is a dedicated, easily accessed, named computer memory location with enough space to store a single integer. Each of the 16 coordinates of the vector v_0, before and after applying the various transformations, will be thought of as a register. A list of these registers and their purposes is given in Figure 2; we elaborate on this outline in the text as well. For the time being the reader should ignore the rows and columns of Figure 2 which are marked with a superscript "a."

The first two registers are called "1" and PD (for path detector). The register denoted "1" always contains the numerical value 1, and each linear transformation is to leave this component of v_0 unchanged. The initial value of PD is zero. At any given moment, the path detector measures whether or not the sequence of transformations applied so far represents a valid flow path according to the Turing machine's transition quintuples Γ.

REGISTER NAME	INITIAL VALUES: COMPONENTS OF v_0	(a) $\tilde{v}_0$	DESCRIPTION OR PURPOSE
1	1	1	Always contains the numerical value 1
PD	0	0	Path Detector: remains zero as long as a sequence of transformations models a valid flow through the Turing machine, otherwise becomes positive
R	0	0	Describes tape contents to the right of the head, viewed as a base B integer written in reverse
R^2	0	0	$(R^2) = (R)^2$
L	N	0	Describes tape contents to the left of the head, viewed as a base B integer
L^2	N^2	0	$(L^2) = (L)^2$
H	0	0	Describes the tape symbol directly under the head
H^2	0	0	$(H^2) = (H)^2$
F	2	2	Contains the code for the current state of the Turing machine (note: $\nu(u^*) = 2$)
F^2	4	4	$(F^2) = (F)^2$
A	1	-1	Used to help control the flow of the program. Codes are as follows: -1: first digit of SI 0: subsequent digits of SI, or like 1 1: ready for T_0 or a T_γ -2 (+2): first digit of M, head moving left (right) -3 (+3): subsequent digits of M, head moving left (right)
A^2	1	1	$(A^2) = (A)^2$
M	0	0	A work space in which the integer part of (R)/B or of (L)/B is constructed non-deterministically
M^2	0	0	$(M^2) = (M)^2$
MR	0	0	$(MR) = (M)\cdot(R)$
ML	0	0	$(ML) = (M)\cdot(L)$
(a) SI		0	Simulated Input: stores an arbitrary number constructed non-deterministically before the modeling of the Turing machine begins
(a) NS		0	Number of Steps: every matrix increments (NS) by one each time it acts

Figure 2 List of the 16 (resp. 18) registers, along with their initial values in v_0 (resp. $\tilde{v}_0$) and their interpretations.

[a] This information is relevant to Theorem 4.8 only.

In a vest any sequence $T_{i_1} \cdots T_{i_n}$ of matrices must be allowed, but only certain transitions are permitted in the Turing machine. To get around this problem we permit the use of any matrix at any stage, but record in the register PD whether or not the selected transformation represents a valid continuation of the path. If so, PD is unchanged; if not, it is increased. At the end, a positive value in PD signifies an invalid path. The $h \times d$ matrix S of the vest measures PD. A necessary condition for $S(v_0 * \sigma)$ to be zero will be that the PD-component of $v_0 * \sigma$ vanish.

The remaining 14 registers come in pairs, denoted $F,F^2;R,R^2;H,H^2;L,L^2;M,M^2;A,A^2;MR,ML$. The names are indeed suggestive: within each pair except the last, one register's contents will always be the square of the other's contents, and each transformation will act so as to preserve this relationship. The last two registers always contain the products of the contents of M and of R, and of M and of L, respectively. To express this in symbols, let (X) denote the contents of the register X. We are saying that the relations $(F^2) = (F)^2$, $(MR) = (M)\cdot(R)$, etc. always hold.

Let $\nu : U \to N$ be any injection which assigns distinct integers to the internal states U. For convenience assume $\nu(u_0) = 0$, $\nu(u_1) = 1$, $\nu(u^*) = 2$. The contents of (F,F^2) will always be $(\nu(u),\nu(u)^2)$ when the Turing machine being modeled is in the state $u \in U$. In particular, the initial values (in terms of components of v_0) for (F,F^2) are $(\nu(u^*), \nu(u^*)^2) = (2,4)$.

The next three register pairs tell us what is on the tape, thus completing the description of the machine's configuration. Since the tape is infinite in both directions, with only finitely many squares nonblank, we can summarize its contents as three finite integers written in base B. The single symbol directly under the head gives the contents of H. The base B number found by reading that portion of the tape to the left of the head (its units digit is in the square immediately to the left of the head) gives the contents of L. Likewise, that portion of the tape to the right of the head can be interpreted as a base B number written in reverse (units digit immediately to the right of the head), and this number gives the contents of R. Because of our convention that the head starts out on the first blank square to the right of the input, our initial values for (L,L^2) are (N,N^2). (This is why ϕ_g had to compute and write down N and N^2.) The registers H,H^2,R,R^2 all have initial values of zero, indicating blank regions of the tape.

We describe next the $m = \#(\Gamma) + 1 + 4B$ matrices in the set T, and in the process we illuminate the roles of the remaining registers. The action of each of these linear transformations is summarized in Figure 3. Each $\gamma \in \Gamma$ is of course a quintuple, $\gamma = (\gamma_1,\gamma_2,\gamma_3,\gamma_4,\gamma_5)$, and to each $\gamma \in \Gamma$ we associate one matrix T_γ whose precise effect on Q^d is described in Figure 3. Basically, when a vector v has components which describe the configuration of our Turing machine, then $T_\gamma v$ will describe the configuration after the transition γ. (If γ represents a transition from a configuration other than that described by v, then the path detector will be incremented.)

We also stipulate an extra transitionlike matrix T_0, which has the effect of leaving all registers intact except PD, which is incremented unless the machine's state is u_1. Including T_0 simulates the idea of making u_1 an absorbing rather than a terminal state: if the Turing machine enters u_1, it keeps returning to u_1 forever. With this change, the number of accepting paths becomes recast as the number of paths which survive for more than $\tau(\ell(N))$ steps. Thus e_n, which for the M-sequence counts the number of valid matrix products of length n, equals precisely the number of accepting paths once n is large enough.

It seems that we have everything we need already, but a problem arises which motivates the inclusion of the remaining 4B matrices. If a transition γ did not move the head, there would be no difficulty representing the effect on the register vector v by a linear transformation. Now suppose the head is moved by γ, say one square to the right, a move signaled by $\gamma_5 = +1$. The new contents of L become $B\cdot(L) + (H)$, which is a linear change. But the new contents of H become the unit's digit (base B) of the old (R), and the new contents of R are the old (R) with its units digit truncated. Both of these are highly nonlinear! Thus no single linear transformation will achieve all of the desired effects on the register vector v. The transition γ cannot be simulated by a single matrix T_γ.

A specific instance may help to clarify this. Suppose the tape reads 12763514, with the head positioned over the six. Our registers will con-

T_γ, $\gamma=(\gamma_1,\gamma_2,\gamma_3,\gamma_4,\gamma_5)\in\Gamma$

$PD \leftarrow (PD)+(A^2)-(A)+(F^2)$
$\quad -2\nu(\gamma_1)(F)+(H^2)-2\gamma_2(H)$
$\quad +(\nu(\gamma_1)^2+\gamma_2^2)(1)$
(a) $\leftarrow (PD)+(A)[(A)-1]+[(H)-\gamma_2]^2$
$\quad +[(F)-\nu(\gamma_1)]^2$
$F \leftarrow \nu(\gamma_4)(1)$
$F^2 \leftarrow \nu(\gamma_4)^2(1)$
$M \leftarrow 0$
$M^2 \leftarrow 0$
$MR \leftarrow 0$
$ML \leftarrow 0$
$A \leftarrow 2\gamma_5(1)$
$A^2 \leftarrow 4(1)$
(b) $NS \leftarrow (NS)+(1)$
If $\gamma_5 = +1$ we also have
$L \leftarrow B(L)+\gamma_3(1)$
$L^2 \leftarrow B^2(L^2)+2B\gamma_3(L)+\gamma_3^2(1)$
(a) $\leftarrow [B(L)+\gamma_3]^2$
If $\gamma_5 = -1$ we also have
$R \leftarrow B(R)+\gamma_3(1)$
$R^2 \leftarrow B^2(R^2)+2B\gamma_3(R)+\gamma_3^2(1)$
(a) $\leftarrow [B(R)+\gamma_3]^2$

T_0

$PD \leftarrow (PD) + (F^2) - 2(F) + (A^2)$
$\quad -2(A) + 2(1)$
(a) $\leftarrow (PD) + [(F) - 1]^2 + [(A) - 1]^2$
(b) $\leftarrow (NS) + (1)$

(b) **E_j, $0\le j\le B_0-1$**

$L \leftarrow B(L) + (j + 1)(1)$
$L^2 \leftarrow B^2(L^2) + 2B(j + 1)(L) + (j + 1)^2(1)$
(a) $\leftarrow [B(L) + (j + 1)]^2$
$A \leftarrow 0$
$A^2 \leftarrow 0$
$SI \leftarrow B_0(SI) + j(1)$
$NS \leftarrow (NS) + (1)$
If $j = 0$ we also have
$PD \leftarrow (PD) + (A^2)$
If $j > 0$ we also have
$PD \leftarrow (PD) + (A^2) + (A)$
(a) $\leftarrow (PD) + [(A) + 1](A)$

D_j^ε, $\varepsilon=\pm1$, $0\le j\le B-1$

$M \leftarrow B(M)+j(1)$
$M^2 \leftarrow B^2(M^2)+2Bj(M)+j^2(1)$
(a) $\leftarrow [B(M)+j]^2$
$MR \leftarrow B(MR)+j(R)$
(a) $\leftarrow [B(M) + j](R)$
$ML \leftarrow B(ML) + j(L)$
(a) $\leftarrow [B(M) + j](L)$
$A \leftarrow 3\varepsilon(1)$
$A^2 \leftarrow 9(1)$
(b) $NS \leftarrow (NS) + (1)$
If $j = 0$ we also have
$PD \leftarrow (PD) + (A^2) - 6\varepsilon(A) + 9(1)$
(a) $\leftarrow (PD) + [(A) - 3\varepsilon]^2$
If $j > 0$ we also have
$PD \leftarrow (PD) + (A^2) - 5\varepsilon(A) + 6(1)$
(a) $\leftarrow (PD) + [(A) - 2\varepsilon][(A) - 3\varepsilon]$

C_j^ε, $\varepsilon=\pm1$, $0\le j\le B-1$

$H \leftarrow j(1)$
$H^2 \leftarrow j^2(1)$
$M \leftarrow 0$
$M^2 \leftarrow 0$
$MR \leftarrow 0$
$ML \leftarrow 0$
$A \leftarrow (1)$
$A^2 \leftarrow (1)$
(b) $NS \leftarrow (NS) + (1)$
If $\varepsilon = +1$ we also have
$R \leftarrow (M)$
$R^2 \leftarrow (M^2)$
$PD \leftarrow (PD) + (A^2) - 5(A) + B^2(M^2)$
$\quad + 2Bj(M) - 2B(MR) + (R^2)$
$\quad -2j(R) + (j^2 + 6)(1)$
(a) $\leftarrow (PD) + [(A) - 2][(A) - 3]$
$\quad +[B(M) + j - (R)]^2$
If $\varepsilon = -1$ we also have
$L \leftarrow (M)$
$L^2 \leftarrow (M^2)$
$PD \leftarrow (PD) + (A^2) + 5(A) + B^2(M^2)$
$\quad + 2Bj(M) - 2B(ML) + (L^2)$
$\quad -2j(L) + (j^2 + 6)(1)$
(a) $\leftarrow (PD) + [(A) + 2][(A) + 3]$
$\quad + [B(M) + j - (L)]^2$

Figure 3 Explicit description of the action of each transformation on the registers.

Rules of the form $X \leftarrow (X)$ are omitted. If a register is not altered by a particular transformation, this information is omitted.
[a]Denotes a nonlinear equivalent expression (see the proof of Lemma 4.6, claim 2).
[b]Relevant to Theorem 4.8 only.

tain (L,H,R) = (127,6,4153). If the head moves to the right without altering any symbols the new values of (L,H,R) should be (1276,3,415). This describes a nonlinear function of the three variables.

In order to get around this problem we introduce 4B additional matrices, calling them $D_j^+, D_j^-, C_j^+, C_j^-$, $0 \le j \le B - 1$. Mnemonically, D stands for "add another digit" and C stands for "compare with the desired result," with the + or − signaling head motion right or left. Here's the plan. Rather than try to obtain (R)'s last digit directly, we use a nondeterministic approach. We detour to construct a new arbitrary number (M), one base B digit at a time, and then simply compare the result with (R). We can keep track of (M)'s units digit and the integer part of (M)/B as we construct it. In the likely event that (M) ≠ (R) we simply augment the path detector to jettison this sequence. But if (M) = (R), we now have (R)'s units digit and the characteristic of (R)/B in our hands!

The use of (M) to denote the new arbitrary integer was no accident: it will indeed be built in the register M. The register pair (A, A^2) will serve to control the flow of the program among the T_γ's, $D_j^\pm$'s, and $C_j^\pm$'s.

We can now describe the $h \times d$ matrix S of the vest: it is the 1×16 row vector which computes $(PD) + (F^2) - 2(F) + (A^2) - 2(A) + 2(1)$. Because of the relations $(F^2) = (F)^2$ and $(A^2) = (A)^2$, this formula coincides with the nonlinear formula $(PD) + [(F) - 1]^2 + [(A) - 1]^2$. Since we always have $(PD) \ge 0$, the formula vanishes only under the simultaneous conditions (PD) = 0 and (F) = 1 and (A) = 1. The condition (PD) = 0 reflects a valid flow path; $(F) = 1 = \nu(u_1)$ indicates an accepting path; and (A) = 1 is the code for having successfully completed any last movements of the Turing machine head.

This completes an overview of the vest which models the Turing machine $(U, [B], \Gamma)$ with input N. We now explore in detail the sequences σ of matrices which lead to $S(v_0 * \sigma)$ being zero.

Suppose we have a path of length $q \le \tau(\ell(N))$ through the Turing machine. By this we mean a list $\delta = \{\delta_i\}_{1 \le i \le q} \subseteq \Gamma$ of transition quintuples, together with a list of quadruples $(u_{(i)}, r_i, h_i, l_i)_{0 \le i \le q} \subseteq U \times N \times [B] \times N$, with the following coherencies. We have $\delta_i = (u_{(i-1)}, h_{i-1}, (\delta_i)_3, u_{(i)}, (\delta_i)_5)$ for each i, $1 \le i \le q$. If $(\delta_i)_5 = +1$ we require $l_i = Bl_{i-1} + (\delta_i)_3$ and $r_{i-1} = Br_i + h_i$. If $(\delta_i)_5 = -1$ we expect $r_i = Br_{i-1} + (\delta_i)_3$ and $l_{i-1} = Bl_i + h_i$. Briefly, $u_{(i)}$ denotes the ith state of the machine for the path δ (with $u_{(0)}$ being u^*) and (r_i, h_i, l_i) describes the tape immediately after the ith transition. It should be clear that δ and (r_0, h_0, l_0) uniquely determine $(u_{(i)}, r_i, h_i, l_i)$ for $1 \le i \le q$. Let Δ_N (resp. Δ_N^a) consist of all paths δ for the input (0,0,N) (resp. all such paths having $u_{(q)} = u_1$).

Given $\delta = \{\delta_i\}$, let $J_{(i)}$ denote the matrix sequence

$$J_{(i)} = D_{j_t}^{\ \varepsilon} D_{j_{t-1}}^{\ \varepsilon} \cdots D_{j_1}^{\ \varepsilon} C_{j_0}^{\ \varepsilon} \qquad [12]$$

where ε is the symbol + (resp. −) if $(\delta_i)_5 = +1$ (resp. −1) and $j_t \cdots j_0$ is the base B representation of r_i (resp. l_i). It is assumed that $j_t > 0$ if $t > 0$. Let Σ denote the set of all sequences of matrices in T. Define $\Phi' : \Delta_N \to \Sigma$ by

$$\Phi'(\delta) = T_{\delta_1} J_{(1)} T_{\delta_2} J_{(2)} \cdots T_{\delta_q} J_{(q)} \qquad [13]$$

if the path δ has length q. The function Φ' is obviously injective. We have been building up to

Lemma 4.6

A matrix sequence σ satisfies $S(v_0 * \sigma) = 0$ if an only if S has the form of $\Phi'(\delta)$ followed by a string of T_0's, for some $\delta \in \Delta_N^a$.

<u>Proof</u>. We first verify four easy claims. For any $\sigma \in \Sigma$, we assert

1. The value of the A-component of $v_0 * \sigma$ is always 0, ±1, ±2, or ±3.
2. The relations $(F^2) = (F)^2$, $(MR) = (M) \cdot (R)$, etc. hold for the components of $v_0 * \sigma$.
3. Acting via any of the transformations on $v_0 * \sigma$ cannot decrease (PD).
4. The value of (PD) in $v_0 * \sigma$ is nonnegative.

Claim 1 is trivial, 2 is by induction on the length of σ, 3 relies on 1 and 2, and 4 follows by induction using 3.

The reader is now referred to Figure 4, which gives a "flowchart" for the vest. We are only interested in sequences σ of matrices which lead to $S(v_0 * \sigma) = 0$. Figure 4 indicates that certain matrices may succeed others only when certain requirements are met. These conditions are "enforced" by the fact that PD is rendered positive if they are disobeyed. The reader is welcome to check this using the explicit actions given in Figure 3.

Now suppose a matrix sequence $\sigma \in \Sigma$ does have the form specified in the statement of Lemma 4.6. By induction on j, one verifies that the components (registers) of

$$v_0 * T_{\delta_1} J_{(1)} T_{\delta_2} J_{(2)} \cdots T_{\delta_j} J_{(j)}$$

are determined by $(F,R,H,L,A,M,PD) = (\nu(u_{(j)}), r_j, h_j, l_j, 1, 0, 0)$. Once j reaches $q = \text{length}(\delta)$, we have $(F) = (A) = 1$ and $(PD) = 0$; hence repeated applications of T_0 will not alter these values. Thus $S(v_0 * \sigma) = 0$.

Conversely, suppose $S(v_0 * \sigma) = 0$. We know that $(PD) = 0$ and $(F) = (A) = 1$ at the end of this sequence of transformations. Since (A) starts out at one and (PD) ends up at zero we must begin σ with a T_γ having $(\gamma)_1 = u^*$. Since T_0 can be applied only when $(F) = (A) = 1$ and no quintuple γ has $(\gamma)_1 = u_1$, any occurrence of T_0 in σ can be followed only by another T_0. Because $(A) = (F) = 1$ at the end, σ must end with T_0 or a $C_j^{\pm}$. Thus σ has the form

$$\sigma = T_{\delta_1} \tilde{J}_{(1)} T_{\delta_2} \tilde{J}_{(2)} \cdots T_{\delta_q} \tilde{J}_{(q)} T_0^{\,s}$$

where $s \geq 0$ and $\tilde{J}_{(i)}$ is a product of $D_j^{\pm}$'s and $C_j^{\pm}$'s. Furthermore, Figure 4 quickly shows that each $\tilde{J}_{(i)}$ must have the form

$$\tilde{J}_{(i)} = D_{j_t}^{\,\varepsilon} \cdots D_{j_1}^{\,\varepsilon} C_{j_0}^{\,\varepsilon} \qquad \text{where } \varepsilon = (\delta_i)_5$$

The remainder of the argument involves induction on i. Suppose that, for some $i \geq 1$, we know that $\{\delta_1, \ldots, \delta_{i-1}\} \in \Delta_N$ and also that $\tilde{J}_{(k)} = J_{(k)}$ for $k < i$. One can determine that the components (registers) of $v_0 * T_{\delta_1} \cdots J_{(i-1)}$ accurately reflect the configuration of the Turing machine after the transitions $\delta_1, \ldots, \delta_{i-1}$. Then δ_i must be a

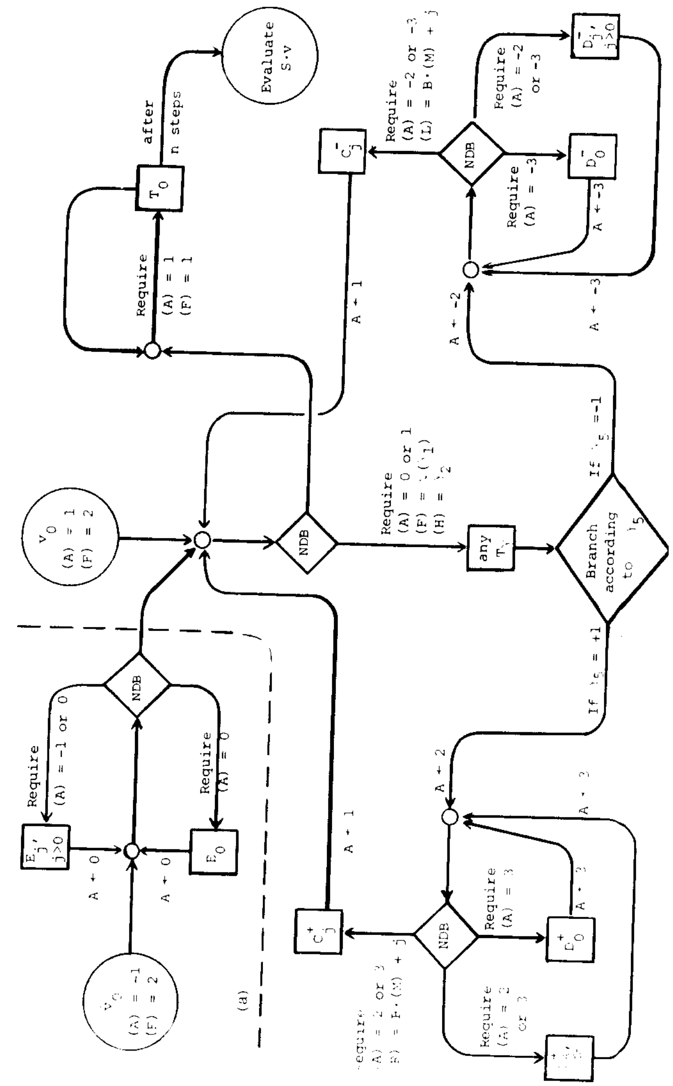

Figure 4 Flowchart for the vest which simulates a nondeterministic Turing machine.

"NDB" stands for "nondeterministic branching."
[a] The upper left region applies only to the vest of Theorem 4.8

valid continuation of the path, and $\tilde{J}_{(i)}$ must equal $J_{(i)}$, or else (PD) would become positive. Eventually we see that $\delta = \{\delta_1, \ldots, \delta_q\} \in \Delta_N$. But we also know that $\nu(u_{(q)}) = 1$, so in fact $\delta \in \Delta_N^{\,a}$.

Lemma 4.7

Let $n \geq \tau(\ell(N)) \cdot (\tau(\ell(N)) + \ell(N) + 1)$. Sequences $\sigma \in \Sigma$ of matrices in T which have length n and which satisfy $S(v_0 * \sigma) = 0$ are in one-to-one correspondence with accepting paths on $(U,[B],\Gamma)$ for the input N.

Proof of Lemma 4.7. We observe first that any $\sigma \in \mathrm{im}(\Phi')$ has length $\leq n$. Since $\sigma = \Phi'(\delta)$ we have by (12) and (13) that s = length $(\sigma) \leq q(1 + p)$, where q is the path length of δ and p is the maximum length of any $J_{(j)}$. Recall that the length of $J_{(j)}$ equals the number of digits in the base B representation of either r_j or l_j. Note that the initial l_0 has $\ell(N)$ digits and that $r_0 = 0$. Since the Turing machine head can move just one square per step and it finishes in q steps, neither r_j nor l_j can ever exceed $\ell(N) + q$ digits. Thus $p \leq \ell(N) + q$ and

$$s \leq q(1 + q + \ell(N)) \leq \tau(\ell(N)) \cdot (\tau(\ell(N)) + \ell(N) + 1) \leq n$$

Now let $\Sigma_n^{\,+}$ consist of length n sequences σ for which $S(v_0 * \sigma) = 0$. Consider the function $\Phi : \Delta_N^{\,a} \to \Sigma$ given by, if $\Phi'(\delta)$ has length s, then $\Phi(\delta)$ consists of $\Phi'(\delta)$ followed by $(n - s)$ T_0's. By the previous paragraph this is well defined. Using Lemma 4.6, $\mathrm{im}(\Phi) = \Sigma_n^{\,+}$. Since Φ is injective, it offers the desired one-to-one correspondence.

It should now be clear that $e_n = \#\{$accepting paths for $(U,[B],\Gamma)$ with input $N\} = g(N)$. This completes the proof of Theorem 4.5.

We cannot resist proving one more theorem which uses the above construction. Theorem 4.8 opens up a vast array of sequences which can be shown to be M-sequences.

Theorem 4.8

Let $g : \mathbf{N} \to \mathbf{N}$ be any function whose domain is all of $\mathbf{N}$. Suppose that $g \in \#\mathcal{P}$. More generally, suppose that there exists a nondeterministic Turing machine, together with constants $K > 0$ and $\epsilon < \frac{1}{2}$, with the following property. When the input is N (given in base B_0), then the number of accepting paths equals g(N) and all paths terminate within $K \cdot N^{\epsilon}$ steps. Then there is an M-sequence $\{\tilde{e}_n\}$ and an integer n_0 such that $\tilde{e}_n = g(n)$ for $n \geq n_0$.

Remark. The restriction that the domain be all of **N** is not a serious constraint in practice. For reasonable **N**-encodings one can generally decide deterministically in polynomial time (resp. in $K \cdot N^{\epsilon}$ time) whether or not a given N belongs to Dom(g). If one extends such a $g : \mathrm{Dom}(g) \to \mathbf{N}$ over all of **N** by putting $\hat{g}|_{\mathrm{Dom}(g)} = g$ and $\hat{g}|_{\mathbf{N}-\mathrm{Dom}(g)} \equiv 0$, then $\hat{g}$ still belongs to $\#\mathcal{P}$ (resp. satisfies the more general hypothesis of Theorem 4.8). The Turing machine for $\hat{g}$ is simply the machine for g together with a deterministic preprocessor which inspects the input and enters state u_0 if it lies in $\mathbf{N} - \mathrm{Dom}(g)$.

Proof of Theorem 4.8. Our task is to construct a vest $(\tilde{d}, \tilde{v}_0, \tilde{T}, \tilde{S})$. We want valid sequences σ of length N ("valid" meaning $\tilde{S}(\tilde{v}_0 * \sigma) = 0$) to be in one-to-one correspondence with accepting paths on a Turing machine $(U,[B],\Gamma)$ whose input is N.

We need to modify the vest developed for Theorem 4.5. For one thing, we need here a specific vest depending only on g. No dependence on an input is allowed, not even in the initial vector $\tilde{v}_0$. For another, we can no longer control the length. Sequences σ of arbitrary length must be allowed to occur.

Essentially, our new vest models a Turing machine which is $(U,[B],\Gamma)$ preceded by a nondeterministic preprocessor. This preprocessor writes down a random base B_0 integer N, called the "simulated input," whose lead digit is nonzero. It does this by writing any digit and then moving to the right over the tape, repeating this step as often as it feels like it. After each such step it is also free to flip into state u^*. When it does eventually enter state u^*, it proceeds to view N as its input and to compute g(N) ("compute" in the nondeterministic sense of path counting).

In order to model this new machine, we keep the 16 registers from before and we introduce two new ones, bringing the total vector space dimension up to $\tilde{d} = d + 2 = 18$. The new registers are called SI for "simulated input" and NS for "number of steps." We also extend the set of matrices T to a larger set $\tilde{T}$ by introducing B_0 new matrices called E_j, $0 \le j \le B_0 - 1$. Lastly, we extend S from a $1 \times d$ to a $2 \times \tilde{d}$ matrix $\tilde{S}$, the second component evaluating (NS) − (SI). Unless both components of $\tilde{S}(v_0 * \sigma)$ are zero, the matrix sequence σ is rejected as invalid.

The NS register counts the number of matrices which have acted on $\tilde{v}_0$. Its initial value is zero, and every transfromation in $\tilde{T}$, including the E_j's, will increase (NS) by 1. Thus the NS component of $\tilde{v}_0 * \sigma$ will equal s if σ has length s.

The SI register is not affected by any of the T_γ's, $D_j^{\pm}$'s, $C_j^{\pm}$'s, or T_0. The initial value of (SI) in $\tilde{v}_0$ is zero. The effect of the E_j's is to build an arbitrary number N in the register L but simultaneously to store that number in the register SI. Once state u^* is entered, (L) will start to change but (SI) is not touched. The SI register maintains a permanent record of the simulated input.

[Recall that a number in base B_0 uses tape symbols 1 through B_0, not 0 through $B_0 - 1$. This explains why (j + 1) and not j gets involved when building the simulated input N in register L; see the action of E_j in Figure 3.]

The initial values for $\tilde{v}_0$ are listed in the third column of Figure 2. The fact that (R) = (L) = (H) = 0 simulates an initially blank tape. Having (A) = −1 forces a valid σ to begin with an E_j, $j > 0$.

Valid sequences σ are those for which (PD) = 0 and (A) = (F) = 1 and (SI) = (NS) = length(σ) at the end. The full Figure 4 illustrates the possibilities. As noted just above, any valid σ must start out with an E_j where $j > 0$, otherwise PD is immediately ruined. Using Figure 4 and Lemma 4.6, we see that a valid σ must have the form

$$\sigma = (E_{j_t} \cdots E_{j_0})\Phi'(\delta)T_0 \cdots T_0, \qquad j_t > 0 \qquad [14]$$

for some accepting path $\delta \in \Delta_n{}^a$, where $(n)_{(B_0)} = j_t \cdots j_0$.

Conversely, any sequence of the form (14) will result in the first component of $\tilde{S}(\tilde{v}_0 * \sigma)$ vanishing. We therefore ask, which σ of the form (14) cause the second component of $\tilde{S}(\tilde{v}_0 * \sigma)$ to vanish as well?

Let n_0 be large enough so that

$$N > K \cdot (N)^{\varepsilon}[K \cdot (N)^{\varepsilon} + \ell(N) + 1] + \ell(N)$$

whenever $N \ge n_0$ (n_0 exists because $\varepsilon < \frac{1}{2}$). Suppose that the simulated input n, which equals the base B_0 number $j_t \cdots j_0$ of (14), is greater than n_0. By the reasoning of Lemma 4.7, the sequence

$$E_{j_t} \cdots E_{j_0}\Phi'(\delta)$$

has length bounded by $\ell(n) + K \cdot (n)^{\varepsilon}[K \cdot (n)^{\varepsilon} + \ell(n) + 1]$, which in turn is smaller than n when $n \ge n_0$. Thus when $n \ge n_0$ there are exactly g(n) sequences of the form (14) which have length n and which have $j_t \cdots j_0 = (n)_{B_0}$. These are precisely the valid sequences σ, for which (SI) must equal (NS) in $\tilde{v}_0 * \sigma$. We have shown that $\tilde{e}_n = \#\{\sigma \mid \tilde{S}(\tilde{v}_0 * \sigma) = 0\} = g(n)$ when $n \ge n_0$, as desired.

Remark. It is trivial that $\mathcal{P} \subseteq \#\mathcal{P}$. More generally, if an algorithm for g(n) runs deterministically in $\tau(n)$ steps, then at most $\tau(n) + \ell(g(n)) + 1 \le 2\tau(n)$ steps are needed to "compute" g(n) in the nondeterministic sense. Once g(n) has been written down on the tape, with the head positioned over its rightmost digit, just one nondeterministic pass through the number g(n) is needed in order to get the number of accepting paths to equal g(n). Thus Theorem 4.8 also applies to any g which can be computed deterministically in $K \cdot (n)^{\varepsilon}$ time.

Example. Let g denote the characteristic function for the set of primes. That is, g(n) = 1 if n is prime but g(n) = 0 for composite n. In view of [2] or [21], g(n) can be computed deterministically in $K \cdot (n)^{\varepsilon}$ steps, where $\varepsilon < 0.15$. Assuming the extended Riemann hypothesis, one even has $g \in \mathcal{P}$! By Theorem 4.8 we deduce the existence of a vest whose M-sequence has $e_n = g(n)$ for all sufficiently large n.

Observation. Although the fast primality-testing algorithms have a certain appeal, in truth the naive algorithm of "look for factors by dividing n by every number smaller that $\sqrt{n}$" will also work in the previous example. This algorithm takes $K_1 \cdot (\sqrt{n}) \log^{c_1}(n)$ time, so it appears to violate the hypotheses of Theorem 4.8. However, in the proof it suffices that every accepting path δ correspond under Φ' to a matrix sequence σ of length $< n$. Referring to the proof of Lemma 4.7, the length of $\Phi'(\delta)$ is actually bounded by the product of q, the maximum time needed, and p, the maximum space needed. Since the naive algorithm needs only polynomial space, this product is $K_2 \cdot (\sqrt{n}) \log^{c_2}(n)$.

which grows more slowly than n. Theorem 4.8 actually applies to any nondeterministically computable g(n) for which the product of the maximum run time $\tau(n)$ and the maximum space needed $\rho(n)$ grows more slowly than n.

Corollary 4.9

Let g satisfy the hypotheses of Theorem 4.8 or the more general hypotheses suggested by the previous sentence. There exists a 123H-algebra A (resp. a finite simply connected CW complex X) whose Hilbert series $H_A(z)$ (resp. $P_{\Omega X}(z)$) is rationally related to the series $\sum_{n=0}^{\infty} g(n)z^n$.

Proof. By Theorem 4.8 we have

$$\sum_{n=0}^{\infty} g(n)z^n = M(z) + p(z)$$

for some M-series M(z) and some polynomial p(z) of degree $< n_0$. Consequently, M(z) is rationally related to $\sum_{n=0}^{\infty} g(n)z^n$. As noted in Section 3, we can arrange for $H_A(z)$ and $P_{\Omega X}(z)$ to be rationally related to M(z).

Example. Let g again denote the characteristic function of the set of primes. There is a 123H-algebra A whose Hilbert series $H_A(z)$ is rationally related to

$$\sum_{p \text{ prime}}^{\infty} z^p = z^2 + z^3 + z^5 + z^7 + z^{11} + z^{13} + \cdots$$

Remark. Because some computations can be done more quickly on a multitape Turing machine than on a one-tape machine, we remark that multitape machines can also be modeled by vests. Now we think of the head as fixed, with any one of the tapes moving one square left or right during each transition. For each new tape we create three new register pairs to record its (R), (H), and (L). We also permit the control register A to assume two more pairs of values per tape, and one introduces more $C^{\pm}$'s and $D^{\pm}$'s accordingly. Thus the machines referred to in Theorem 4.8 and Corollary 4.9 can be taken to be multitape machines.

5. SOME #𝒫-HARD PROBLEMS

In Section 5 we offer rigorous definitions for Turing reducibility (in polynomial time), Turing equivalence, and #𝒫-hardness. We demonstrate that 12 natural problems concerning Hilbert and Poincaré sequences are all equivalent to one another and that all are #𝒫-hard. Certain conventions are adopted in the course of N-encoding these problems, and we engage in considerable discussion of the rationale behind these conventions.

The concept of Turing reducibility was originally formulated in order to describe a hierarchy of functions (or of sets) in recursion theory. Consider two functions, $f: \mathrm{Dom}(f) \to N$ and $g: \mathrm{Dom}(g) \to N$. The output g(N) may be difficult or impossible to compute directly, but g and f might be related in such a way that g(N) can be computed quickly provided that $f(N_1), \ldots, f(N_s)$ are available. Here it is assumed that the arguments N_i can be computed quickly from N and from $f(N_1), \ldots, f(N_{i-1})$. Thus the problem g has been "reduced" to the problem of determining certain f-values.

In practice, a computer programmer might write a subroutine to evaluate f( ) and call it during the routine which computes g( ). The theoretical computer science analog is called an oracle. An oracle is a special state of a hypothetical Turing machine which is entered after an argument N_i is N-encoded onto a section of the tape. Once in the oracle state, the machine replaces N_i by $f(N_i)$ on the tape, a process which is presumed to require only $\ell(N_i)$ steps. The machine then enters an ordinary state and resumes its calculations.

In practice, this requires a user-transparent computation. In the world of theory, the Turing machine is permitted to look up $f(N_i)$ in a hypothetical table of f-values and to copy the answer in just $\ell(N_i)$ steps. Since f might even be a nonrecursive function, presuming the availability of such a table is akin to expecting magic or divine intervention, hence the term "oracle."

Definition 5.1

Let $f: \mathrm{Dom}(f) \to N$ and $g: \mathrm{Dom}(g) \to N$ be two functions. We say that g is Turing reducible to f [in polynomial time], written $g \leq_T f$, if and only if there exists a deterministic Turing machine with the following property. It has an oracle which looks up f-values, and it computes g(I) in polynomial time. [As before, "polynomial time" means that the number of steps is bounded by a fixed polynomial in $\ell(I)$.]

Remark. The relation $\leq_T$ is readily seen to be reflexive and transitive. We may interpret "$g \leq_T f$" as "g is no harder than f" for the following reason. If g is Turing reducible to f, then f being computable in polynomial time would make g computable in polynomial time. Ease of computing f immediately translates into ease of computing g. On the other hand, g might be strictly easier to obtain than f, since there could be other algorithms for g which do not even use the oracle. Contrapositively, if g cannot be computed in polynomial time, then neither can f; in this sense f is "at least as hard as g."

Definition 5.2

Let $f: \mathrm{Dom}(f) \to N$ and $g: \mathrm{Dom}(g) \to N$ be two functions. We say that f and g are Turing equivalent [in polynomial time], denoted $f \approx_T g$, if and only if both $f \leq_T g$ and $g \leq_T f$.

Terminology. In this chapter we will always say "Turing reducible (resp. equivalent)" when we mean "Turing reducible (resp. equivalent) in polynomial time."

It should be clear that Turing equivalence is an equivalence relation on functions from subsets of N to N. As a nontrivial example, note that any #$\mathcal{P}$-problem is Turing reducible to any #$\mathcal{P}$-complete problem. Consequently, all #$\mathcal{P}$-complete problems lie in the same Turing equivalence class.

Definition 5.3

A function $f: \mathrm{Dom}(f) \to N$ is #$\mathcal{P}$-hard if and only if $g \leq_T f$ for every $g \in \#\mathcal{P}$.

This definition says roughly that a #$\mathcal{P}$-hard problem is as hard as or harder than anything in #$\mathcal{P}$. Note that all #$\mathcal{P}$-complete problems are #$\mathcal{P}$-hard, but not conversely.

Lemma 5.4

(i) If $f_0 \leq_T f_1$ and f_0 is #$\mathcal{P}$-complete or #$\mathcal{P}$-hard, then f_1 is #$\mathcal{P}$-hard.
(ii) If $f_1 \approx_T f_2$ and f_1 is #$\mathcal{P}$-hard, then f_2 is #$\mathcal{P}$-hard.

Proof. This is a trivial exercise in using the definitions.

We next present a list of nine problems which will all be shown to be Turing equivalent to one another and #$\mathcal{P}$-hard.

(A) Find the nth term of the Tor-sequence of a 123H-algebra A.
(B) Find the nth term of the Hilbert sequence of a 123H-algebra A.
(C) Find the nth term of the Hilbert sequence of a 12H-algebra A.
(D) Find the nth term of the Hilbert sequence of a 12-algebra A.
(E) Find the nth term of the Hilbert sequence of a degree-one-generated finitely presented connected graded Q-algebra A.
(F) Find the nth entry of the Poincaré sequence of a commutative local Q-algebra $(R, \mathfrak{M})$ for which $\mathfrak{M}^3 = 0$.
(G) Find the nth entry in the Poincaré-Betti sequence of a finite-dimensional basic local Q-algebra R.
(H) Evaluate $\dim_Q(H_n(\Omega X; Q))$, where X is a simply connected finite CW complex having cells (other than the base point) in dimensions two and four only.
(I) Determine $\dim_Q(\pi_{n+1}(X) \otimes Q)$, where X is as described in (H) above.

Let us specify how the input is to be N-encoded for each of these problems. Problems (A) through (D) call for an integer n and a description of a 12-algebra. In the presentation

$$A = Q\langle x_1, \ldots, x_g\rangle / \langle \alpha_1, \ldots, \alpha_r\rangle$$

we have

$$\alpha_k = \sum_{i=1}^{g} \sum_{j=1}^{g} c_{ijk} x_i x_j \qquad [15]$$

for certain constants $c_{ijk} \in Q$. We can assume, by clearing denominators if necessary, that $c_{ijk} \in Z$. To specify a general 12-algebra A we will therefore give g and r in decimal, followed by the $g^2 r$ decimal integers $\{c_{ijk}\}$. The input n then follows in unary.

Objections. This N-encoding is appropriate for very complicated or "generic" presentations, but it seems very wasteful for relatively simple algebras, which tend to occur in practice. Consider the presentation

$$A = Q\langle x_1,\ldots,x_{10}\rangle / \langle x_1x_2 + 6x_3x_4, x_5x_6, x_7x_8, x_9x_{10}\rangle \qquad [16]$$

Our N-encoding will require over 400 entries, nearly all of them being zero, whereas (16) presents the same information in just one line. However, if concise inputs like (16) were sometimes allowed, this could skew the computational complexity of the overall problem and throw off the upcoming theorem on Turing equivalence.

The situation is analogous to that of scientific or other space-saving notations. Borrowing from FORTRAN the notation xEy for $10^y x$, note that the answer to the addition problem "2E1000 + 3E25" would have to be written as

20000···[974 zeros]···0003E25

The size of this output is not a polynomial in the input length, which suggests that addition does not belong to $\mathcal{P}$. To avoid this false or at best misleading conclusion we require that the input always be written out in full decimal notation, no matter how inefficient this seems for a given problem. Likewise, we always insist that the presentation for a 12-algebra be N-encoded as described above.

A second possible objection to our proposed N-encoding is that it is redundant for 12H-algebras. When A is a 12H-algebra the constants $\{c_{ijk}\}$ satisfy $c_{ijk} = c_{jik}$, so nearly half of them are superfluous. The response to this objection is that it doesn't matter, since cutting the input size by a factor of two has no effect on the computational complexity. To make this precise, consider a problem (C'), which seeks the same output as problem (C) but whose input omits the double entries. In polynomial time an N-encoded input for (C') can be converted to the corresponding input for (C), and also vice versa, so (C') and (C) are Turing equivalent. In general, when one has several choices of N-encoding for the input to a problem, its Turing equivalence class is independent of the choice as long as the various inputs can be obtained from one another in polynomial time.

Returning to our list of problems, the input for problem (E) is handled similarly. Now the algebra A is allowed to have a presentation of the form

$$A = Q\langle x_1,\ldots,x_g\rangle / \langle \beta_1,\ldots,\beta_r\rangle \qquad [17]$$

where each $|x_i| = 1$ but the relations' degrees $t_k = |\beta_k|$ may be arbitrary positive integers. To describe the β_k's let Λ_s denote the set of length s sequences whose entries come from $\{1,2,\ldots,g\}$, and for $\lambda = (i_1,\ldots,i_s) \in \Lambda_s$ let x_λ denote the monomial $x_{i_1}\cdots x_{i_s}$. A typical relation β_k has the form

$$\beta_k = \sum_{\lambda \in \Lambda_{t_k}} c_{\lambda k} x_\lambda, \qquad c_{\lambda k} \in Q$$

where again we may clear denominators and assume that $c_{\lambda k} \in Z$. In order to specify a degree-one-generated algebra A we write

$$(g)_{(10)}; (r)_{(10)}; (t_1)_{(10)}; \ldots; (t_r)_{(10)}$$

followed by the $\{c_{\lambda k}\}$ in decimal. The input n is in unary.

Now consider problems (F) and (G). A finite-dimensional basic local Q-algebra is an associative ring R with unity such that $R/\mathcal{R}(R) \approx Q$ and $\dim_Q(R) < \infty$, where $\mathcal{R}(R)$ denotes the Jacobson radical of R. Viewing Q as an R-module, we may form the Poincaré-Betti sequence $\{\rho_n\}$, where

$$\rho_n = \dim_Q(\mathrm{Tor}_n^R(Q,Q))$$

The Poincaré sequence of a commutative Artinian Q-algebra $(R,\mathfrak{M})$ having $\mathfrak{M}^3 = 0$ is a very special case. The rings discussed in problem (F) are always isomorphic to polynomial ring quotients of the form

$$Q[y_1,\ldots,y_g]/J$$

Here J denotes an ideal satisfying $\mathfrak{a}^2 \supseteq J \supseteq \mathfrak{a}^3$ when $\mathfrak{a}$ denotes the ideal $(y_1,\ldots,y_g)$.

A general finite-dimensional basic local Q-algebra R may be written as $Q \oplus \mathcal{R}(R)$. If $\{b_1,\ldots,b_s\}$ is a Q-basis for $\mathcal{R}(R)$, the multiplication on R determines s^3 constants d_{ijk} via

$$b_i b_j = \sum_{k=1}^{s} (d_{ijk}) b_k \qquad [18]$$

Conversely, the constants $\{d_{ijk}\} \subseteq \mathbb{Q}$ determine R.

We may convert all the $\{d_{ijk}\}$ to whole numbers by choosing a least common denominator δ and by using $\{b_i'\}$ instead of $\{b_i\}$ for our basis, where $b_i' = \delta b_i$. This converts (18) to the expression

$$b_i' b_j' = \sum_{k=1}^{s} (d_{ijk}') b_k'$$

where $d_{ijk}' = \delta d_{ijk} \in \mathbb{Z}$. Our N-encoding for problems (F) and (G) will consists of s and the s^3 integers $\{d_{ijk}'\}$, all written in decimal, together with n written in unary.

As to problems (H) and (I), the space X is determined up to homotopy type by the collection of r homotopy classes $[f_k] \in \pi_3(W)$, where $W = \vee_{i=1}^{g} S^2$. Since it is a free abelian group on $\frac{1}{2}g(g+1)$ generators, $\pi_3(W)$ is easy to understand. When $\iota_1, \ldots, \iota_g$ denote the natural generators of $\pi_2(W) \approx Z^g$, then $\pi_3(W)$ is generated by the g compositions $\iota_i \circ \eta$ (η denotes the Hopf map) and by the $\frac{1}{2}g(g-1)$ Whitehead products $[\iota_i, \iota_j]$ for $i < j$. The homotopy class $[f_k]$ may be written

$$[f_k] = \sum_{i=1}^{g} c_{iik}[\iota_i \circ \eta] + \sum_{i<j} c_{ijk}[\iota_i, \iota_j]$$

for some integers $\{c_{ijk}\}$. In order to specify X it suffices to list the $\frac{1}{2}r(g^2 + g)$ coefficients $\{c_{ijk} \mid 1 \leq i \leq j \leq g,\ 1 \leq k \leq r\}$. Our N-encoding will give r and g and the $\{c_{ijk}\}$ in decimal, followed by n in unary.

Proposition 5.5

Problem (A) is #$\mathcal{P}$-hard.

Proof. By Lemma 5.4(i) and Theorem 4.5, we need only show that the problem "compute the nth entry in the M-sequence of a vest" is Turing reducible to problem (A). This reduction is implicit in Theorem 3.4, as long as the 123H-algebra corresponding to a given vest can be N-encoded in polynomial time.

The proof in [4] of Theorem 3.4 indicates how this can be done. From the vest $(d, v_0, \{T_i\}, S)$ we get a 123H-algebra A having $g = 2m + d + h + 3$ generators and $r = (m+1)(m+d+h+2) + 1$ relations. Some of the constants $\{c_{ijk}\}$ in A's presentations are 0 or ±1 according to a straightforward pattern, and the remainder may simply be copied from the entries of the $\{T_i\}$ or S.

Remark. The reduction described in the above proof is actually of the special "polynomial time many-one" type mentioned in Section 4. If it were also true that problem (A) belonged to #$\mathcal{P}$, then it would be #$\mathcal{P}$-complete. However, the author doubts that (A) $\in$ #$\mathcal{P}$.

Theorem 5.6

Problems (A) through (I) are all Turing equivalent to one another, and all are #$\mathcal{P}$-hard.

Proof. In view of Lemma 5.4(ii) and Proposition 5.5, it suffices to demonstrate the Turing equivalence of the nine problems.

The calculations in Section 3 indicate the equivalences of (A) with (B) and of (H) with (I). For instance, if the answer to (I) were available through an oracle, one could make a list on the tape of the rational homotopy ranks $\{r_j(X)\}$ for $1 \leq j \leq n$ and then apply the indicated procedures to determine $\dim(H_n(\Omega X; \mathbb{Q}))$. Notice how critical it is here that n be given in unary. Both the listing of $\{r_j(X)\}$ and the subsequent computation require a length of time whose dependence on n is a polynomial in n. Because n is in unary, a polynomial in n is bounded by a polynomial in the input length.

Likewise, one easily obtains the equivalences (C) $\underset{T}{\approx}$ (H) and (see [6]) (C) $\underset{T}{\approx}$ (F). In the Turing reduction, the coefficients $\{c_{ijk}\}$ need only be copied and rearranged in preparation for the oracle. For instance, to show that (C) $\underset{T}{\leq}$ (H), we first convert the input I describing a 12H-algebra A into an input I' describing the associated four-dimensional CW complex X. We then invoke the oracle n times in order to list on our tape the first n terms of the Betti sequence of ΩX. Lastly, we calculate, as described in Section 3, the nth term of the Hilbert sequence for A. Each of these steps requires only a polynomial in $\ell(I)$ time.

It is trivial that (F) $\underset{T}{\leq}$ (G) and that (B) $\underset{T}{\leq}$ (C) $\underset{T}{\leq}$ (D). That (D) $\underset{T}{\leq}$ (B) is a consequence of [17, theorem 1.1]. Jacobsson's construction of a 123H-algebra whose Hilbert series is rationally related to that of an arbitrary 12-algebra takes only polynomial time.

We demonstrate next that (D) $\underset{T}{\approx}$ (E). The reduction (D) $\underset{T}{\leq}$ (E) is trivial. To show that (E) $\underset{T}{\leq}$ (D), use "link (c)-(d)" of [6]. In that article a construction is given which obtains from an arbitrary degree-one-generated A a 12-algebra A' such that $H_A(z)$ and $H_{A'}(z)$ are rationally related. When A has the presentation (17), then A' will have

$$g' = 1 + 3g + 9g^2 + \cdots + (3g)^{\bar{t}-1}$$

generators, where $\bar{t} = \max\{|\beta_1|, \ldots, |\beta_r|\}$. The coefficients c_{ijk} involved in the presentation of A' are 0 or ±1 or are obtained by copying the $c_{\lambda k}$'s. Using the facts that $g \geq 2$ (since the case $g = 1$ is trivial) and

$$g^{\bar{t}} = \#(\Lambda_{\bar{t}}) = \#\{c_{\lambda k'}\} < \ell(I)$$

when k' denotes an index such that $|\beta_{k'}| = \bar{t}$, we obtain

$$g' < (3g)^{\bar{t}} < g^{3\bar{t}} = (g^{\bar{t}})^3 < \ell(I)^3$$

Therefore the input I' describing the 12-algebra A' can be written down from the I which describes A, in an amount of time which is a polynomial in $\ell(I)$. It follows that (E) $\underset{T}{\leq}$ (D).

The above arguments show that eight of the nine problems, all except (G), are Turing equivalent. It remains only to show that (G) is reducible to one of the others; we will see that (G) $\underset{T}{\leq}$ (D). Applying the cobar construction to a finite-dimensional basic local Q-algebra R yields a free finitely generated differential graded algebra (B,d) for which

$$H^*(B,d) \approx \mathrm{Tor}^R_*(Q,Q)$$

as graded vector spaces. By Gulliksen's construction (see corollary 2 of [6]) there is a 12-algebra A whose Hilbert series is rationally related to the "homology series" of (B,d), which coincides with the Poincaré-Betti series for R. Obtaining A's presentation from R's takes only polynomial time; again, it is mostly a reshuffling of the coefficients. This completes the proof of Theorem 5.6.

Remark. Notably absent from the list (A) through (I) is the problem of finding the Poincaré series of an arbitrary commutative Noetherian (not necessarily Artinian) local Q-algebra. There are two known reductions from such rings to problems in the list, namely Levin's reduction to Artinian rings [19] and "link (e)-(f)" of [6] to 12-algebras. However, neither reduction is certifiably accomplished in polynomial time. For instance, Levin's reduction requires us to choose a "sufficiently large" integer q so that $P_R(z)$ and $P_{R/\mathfrak{M}^q}(z)$ will be rationally related. The proof is nonconstructive and relies on the Artin-Rees lemma. If R is described via a presentation, e.g., as the quotient of a polynomial ring, the integer q need not be bounded by a polynomial in the size of the presentation. That even the length of q might grow faster than a polynomial in the presentation size is suggested by the results of [8], where some similar problems are studied.

We consider next three problems which are natural generalizations of problems (E), (H), and (I). We show that they too are Turing equivalent to the problems of Theorem 5.6, but we must make an unexpected assumption about their N-encodings.

(J) Find the nth term of the Hilbert sequence of a finitely presented connected graded Q-algebra A.

(K) Evaluate $\dim_Q(H_n(\Omega X;Q))$ for a finite simply connected CW complex X.

(L) Determine $\dim_Q(\pi_{n+1}(X) \otimes Q)$, where X is a finite simply connected CW complex.

Again we must be careful to specify how the input is to be N-encoded. As to problem (J), a presentation for the algebra A looks like

$$A = Q\langle y_1, \ldots, y_g\rangle / \langle \beta_1, \ldots, \beta_r\rangle \qquad (19)$$

where y_i has degree $m_i \geq 1$ and β_k has some homogeneous degree t_k. For a sequence $\lambda = (i_1, \ldots, i_q)$ of indices, $1 \leq i_j \leq g$, let $|\lambda|$ denote $m_{i_1} + \cdots + m_{i_g}$ and let y_λ denote the monomial $y_{i_1} \cdots y_{i_q}$ in the free associative algebra $F = Q\langle y_1, \ldots, y_g\rangle$. If Λ_s denotes the set of sequences λ for which $|\lambda| = s$, then $\{y_\lambda \mid \lambda \in \tilde{\Lambda}_s\}$ is a Q-basis for the sth homogeneous piece F_s of F. Since each relation β_k is homogeneous of degree t_k, we can write (after clearing denominators)

$$\beta_k = \sum_{\lambda \in \tilde{\Lambda}_{t_k}} e_{\lambda k} y_\lambda , \qquad e_{\lambda k} \in Z$$

Thus the algebra A is specified by the "presentation degree vector"

$$(g;m_1,\ldots,m_g;r;t_1,\ldots,t_r)$$

and by the collection $\{e_{\lambda k}\}$ of coefficients. Let $\bar{t}$ denote $\max(t_1,\ldots,t_r)$.

To be consistent with problem (E) our input should provide

$$(g)_{(10)};(m_1)_{(10)};\ldots;(m_g)_{(10)};(r)_{(10)};(t_1)_{(10)};\ldots;(t_r)_{(10)} \qquad [20]$$

followed by the $\{e_{\lambda k}\}$ in decimal and n in unary. However, with this N-encoding we run into difficulties. Let us denote by (J') the problem which seeks $\dim(A_n)$ from the input just described. It is trivial that (E) $\underset{T}{\le}$ (J'); what happens if we try to prove that (J') $\underset{T}{\le}$ (E)?

We must relate an arbitrary Hilbert series to that of a degree-one-generated algebra. One efficient way to do this is as follows. Starting with A given as in (19), let

$$\psi : Q\langle y_1,\ldots,y_g\rangle \to Q\langle x_0,x_1,\ldots,x_g\rangle$$

be the graded algebra homomorphism in which $|x_j| = 1$ and $\psi(y_i) = x_0^{m_i-1}x_i$. Then ψ induces a monomorphism $\hat{\psi}: A \to \hat{A}$, where

$$\hat{A} = Q\langle x_0,x_1,\ldots,x_g\rangle/\langle\psi(\beta_1),\ldots,\psi(\beta_r)\rangle$$

is degree-one generated. By [5, theorems 3.1 and 2.4] one has the rational relationship

$$H_{\hat{A}}(z)^{-1} = H_A(z)^{-1} - (g+1)z + (z^{m_1} + \cdots + z^{m_g})$$

The presentation for $\hat{A}$, which has as relations $\hat{\beta}_1 = \psi(\beta_1),\ldots,\hat{\beta}_r = \psi(\beta_r)$, can be obtained easily from knowledge of $\beta_1,\ldots,\beta_r$. If

$$\beta_k = \sum_{\lambda \in \tilde{\Lambda}_{t_k}} e_{\lambda k} y_\lambda$$

then

$$\hat{\beta}_k = \sum_{\lambda \in \tilde{\Lambda}_{t_k}} e_{\lambda k} x_{\tilde{\psi}(\lambda)} \qquad [21]$$

where $\tilde{\psi}: \tilde{\Lambda}_s \to \Lambda_s$ is the injection such that $\psi(y_\lambda) = x_{\tilde{\psi}(\lambda)}$. It appears that $\hat{\beta}_k$ can quickly be obtained from β_k. But in keeping with our previous N-encoding conventions, (21) should really be written out in full as

$$\hat{\beta}_k = \sum_{\sigma \in \Lambda_{t_k}} c_{\sigma k} x_\sigma \qquad [22]$$

where

$$c_{\sigma k} = \begin{cases} e_{\lambda k} & \text{if } \sigma = \tilde{\psi}(\lambda) \text{ for some } \lambda \in \tilde{\Lambda}_{t_k} \\ 0 & \text{if } \sigma \in \Lambda_{t_k} - \operatorname{im}(\tilde{\psi}) \end{cases}$$

Now we see the problem which arises in passing from an input I' describing A to an input $\hat{I}$ for $\hat{A}$. Because $\#(\Lambda_{t_k})$ may grow much faster than $\#(\tilde{\Lambda}_{t_k})$, the number of coefficients which are in zero in (22) cannot be bounded by any fixed polynomial in $\ell(I')$. The reduction from (J') to (E) fails to take polynomial time, simply because there may be more than polynomially many zeros to write down! We must do one of three things: revise our N-encoding scheme for problem (E) so that it omits copious zeros; alter our N-encoding for problem (J'); or find another reduction, which does take polynomial time, from (J') to (E). The first course of action would again leave us open to the various objections discussed earlier, and I have not succeeded at the third, so we will adopt the second option.

In problem (J), the input I will consist of expression (20), the $\{e_{\lambda k}\}$ in decimal, $(g^{\bar{t}})_{(1)}$, and $(n)_{(1)}$. The insertion of $(g^{\bar{t}})_{(1)}$ guarantees that $\ell(I)$ will always exceed $g^{\bar{t}}$. The number of new zeros to be listed while describing the $\hat{\beta}_k$'s is smaller than

$$\#(\Lambda_{t_1}) + \#(\Lambda_{t_2}) + \cdots + \#(\Lambda_{t_r}) \le r\, g^{\bar{t}} < \ell(I)^2$$

so now the reduction from (J) to (E) does take polynomial time. Thus $(J) \leq_T (E)$. Conversely, the input to (E) describing a degree-one-generated algebra already lists at least $g^{\bar{t}}$ coefficients, so its length exceeds $g^{\bar{t}}$ and only polynomial time is needed in order to insert $(g^{\bar{t}})_{(1)}$ into the input. Hence $(E) \leq_T (J)$ as well.

With the above N-encoding convention, we have shown

Lemma 5.7

Problems (E) and (J) are Turing equivalent.

Finally, let us turn our attention to problems (K) and (L). We will at last justify our title assertion that problem (L) is #$\mathcal{P}$-hard.

The input for (K) and for (L) is an N-encoded finite description of a finite simply connected CW complex X. We mentioned in Section 3 that we would rely upon the Quillen model of X for this description. The Quillen model is a free differential graded Lie algebra $(\mathcal{L}X, d_X)$ whose generators correspond with, but lie in one degree lower than, a basis for $\bar{H}_*(X;Q)$. Calling this basis $\{a_1,\ldots,a_h\}$ and letting $m_1,\ldots,m_h$ denote their degrees in $H_*(X;Q)$, we must specify their boundaries

$$d_X(a_k) \in (\mathcal{L}X)_{m_k-2} = (L_Q\langle a_1,\ldots,a_h\rangle)_{m_k-2}$$

Here $L_Q\langle S\rangle$ denotes the free Lie Q-algebra on a graded set S and $(L)_s$ denotes the degree s component of a graded group L.

For $x_1,\ldots,x_q \in \mathcal{L}X$ let $[x_1,\ldots,x_q]$ denote the repeated bracket

$$[\cdots[[x_1,x_2],x_3],\ldots,x_q]$$

Let Λ'_s consist of all sequences $\lambda = (i_1,\ldots,i_q)$ for which $(m_{i_1} - 1) + \cdots + (m_{i_q} - 1) = s$, and for each such λ let $[a_\lambda]$ denote $[a_{i_1},\ldots,a_{i_q}]$. Any element y of $(\mathcal{L}X)_s$ may be written (not uniquely) as

$$y = \sum_{\lambda\in\Lambda'_s} e_\lambda[a_\lambda], \quad e_\lambda \in Q$$

In particular, $(\mathcal{L}X, d_X)$ is specified by h and the degree list $(m_1,\ldots,m_h)$, together with a list of coefficients $\{e_{\lambda k} \mid 1 \leq k \leq h, \lambda \in \Lambda'_{m_k-2}\}$ for which

$$d_X(a_k) = \sum_{\lambda\in\Lambda'_{m-2}} e_{\lambda k}[a_\lambda]$$

We may always assume the $\{e_{\lambda k}\}$ to be whole numbers after replacing the $\{a_k\}$ by suitable multiples of themselves.

Let $\bar{m}$ denote $\max\{m_1,\ldots,m_h\}$. The input I for problems (K) and (L) will consist of the expression

$$(h)_{(10)};(m_1)_{(10)};\ldots;(m_h)_{(10)}$$

followed by the $\{e_{\lambda k}\}$ in decimal, followed by $(h^{\bar{m}})_{(1)}$ and $(n)_{(1)}$. By including $(h^{\bar{m}})_{(1)}$ in the input we guarantee that $\ell(I) > h^{\bar{m}}$, which is useful to us in much the same way that including $(g^{\bar{t}})_{(1)}$ in the input for problem (J) was useful.

Theorem 5.8

The 12 problems (A) through (L) are all Turing equivalent to one another, and all are #$\mathcal{P}$-hard.

<u>Proof</u>. In view of Theorem 5.6 and Lemma 5.7, it suffices to show that $(H) \leq_T (K)$, that $(K) \leq_T (J)$, and that $(K) \approx_T (L)$. The last of these claims follows from the remarks of Section 3 and the fact that n is given in unary.

That $(H) \leq_T (K)$ is virtually trivial. We need to know that, in polynomial time, a description $\hat{I}$ of a space X of dimension four can be converted into a description I of the same X as a general space. The coefficients c_{ijk} for $i < j$ (resp. $i = j$) in the input to (H) can simply be copied (resp. halved) to obtain suitable $e_{\lambda k}$'s. The only major difference between I and $\hat{I}$ is therefore the insertion of $(h^{\bar{m}})_{(1)}$ into $\hat{I}$. But here we have $\bar{m} \leq 4$, hence

$$\ell(h^{\bar{m}})_{(1)} = h^{\bar{m}} \leq h^4 = (g + r)^4 < \ell(\hat{I})^4$$

so only polynomial time is needed in order to build I from $\hat{I}$.

For the Turing reduction $(K) \leq_T (J)$, use the Adams-Hilton model together with "link (a)-(b)" of [6]. By [7] the Adams-Hilton model for X may be taken to be the universal enveloping algebra of $(\mathcal{L}X, d_X)$.

which may be written out in full in polynomial time. In the subsequent reduction to a finitely presented graded algebra, a space X whose reduced rational homology had a basis in dimensions $(m_1,\ldots,m_h)$ is associated to an algebra A having $g = h + 3$ generators in degrees

$$(1,1,2,m_1 - 1,m_2 - 1,\ldots,m_h - 1)$$

The relations for A occur in degrees no greater than $\overline{m} + 1$ and they may be written down in a straightforward manner from knowledge of the Adams-Hilton model. The N-encoded description for A has length which is a polynomial in $(h + 3)^{\overline{m}+1}$; this is bounded by a polynomial in $h^{\overline{m}}$, as needed. The proof is now complete.

Remark. The reader who is still concerned about the inclusion of $(g^{\overline{t}})_{(1)}$ (resp. $(h^{\overline{m}})_{(1)}$) in our input is reminded that this can only decrease the computational complexity. In other words, if (J') (resp. (K') or (L')) is the problem which seeks the same output as (J) (resp. (K) or (L)) but without the redundant unary in the input, then (J) $\underset{T}{\leq}$ (J') (resp. (K) $\underset{T}{\leq}$ (K'), (L) $\underset{T}{\leq}$ (L')). In view of Lemma 5.4(i) and Theorem 5.8, we know that (J') (and (K') and (L')) are $\#\mathcal{P}$-hard. Thus we may assert without equivocation that "the computation of rational homotopy groups is $\#\mathcal{P}$-hard."

6. COMPUTABILITY IN EXPONENTIAL TIME

We will define "exponential time" and prove that problems (A) through (L) of the previous section can be computed in exponential time. This will complete our proof of the information summarized in Figure 1. We close with some philosophical remarks about computational complexity.

Definition 6.1

A function $f : \mathrm{Dom}(f) \to N$, $\mathrm{Dom}(f) \subseteq N$, is computable in exponential time if and only if there exists a deterministic Turing machine, together with a polynomial $\mu(x)$, having the following property. When the input is $N \in \mathrm{Dom}(f)$, the output is $f(N)$, and the total number of steps needed is bounded above by $2^{\mu(\ell(N))}$.

Lemma 6.2

If $f \underset{T}{\leq} g$ and g is computable in exponential time, then f is computable in exponential time.

Proof. If $f \underset{T}{\leq} g$, then f can be computed deterministically in $\tau(\ell(N))$ steps, $\tau(x)$ being a polynomial, provided we have access to an oracle for g. In particular, the oracle can be invoked at most $\tau(\ell(N))$ times, and the arguments N_i for which we request $g(N_i)$ all have length bounded by $\tau(\ell(N))$.

Modify the machine for f by replacing the special oracle state with a "subroutine call" to the machine which computes g in $2^{\mu(\ell(N_i))}$ time. The new machine still computes f and it runs in fewer than

$$\tau(\ell(N)) + (\tau(\ell(N)))\cdot(2^{\mu(\tau(\ell(N)))}) \qquad [23]$$

steps. Putting $\mu'(x) = \mu(\tau(x)) + \tau(x)$, we have expression (23) being majorized by $2^{\mu'(\ell(N))}$, as needed.

Lemma 6.3

If $g \in \#\mathcal{P}$, then g can be computed in exponential time.

Proof. In view of Lemma 6.2, it suffices to prove this for a single $\#\mathcal{P}$-complete problem. Let g denote the problem of Theorems 4.3 and 4.5. The function g can be computed deterministically by serially considering each length n sequence $\sigma = (\sigma_1,\ldots,\sigma_n)$ and evaluating $v_0 * \sigma$. Each evaluation takes $\tau(\ell(I))$ steps, $\tau(x)$ being a polynomial, and there are $m^n < (\ell(I))^{\ell(I)}$ sequences to consider. The entire deterministic process is completed in

$$\tau(\ell(I))\cdot \ell(I)^{\ell(I)} < 2^{\tau(\ell(I))+\ell(I)^2}$$

steps.

Theorem 6.4

Problem (D) of Section 5 can be computed in exponential time.

<u>Proof</u>. Let

$$A = Q\langle x_1,\ldots,x_g\rangle/\langle\alpha_1,\ldots,\alpha_r\rangle$$

be a 12-algebra. Let F denote the free algebra $F = Q\langle x_1,\ldots,x_g\rangle$ and let J denote the two-sided ideal in F generated by the relations $\alpha_1,\ldots,\alpha_r$, so that $A = F/J$. We view J as a graded vector subspace of $F = \oplus_{m=0}^{\infty} F_m$ and write $J = \oplus_{m=2}^{\infty} J_m$. The nth entry in the Hilbert sequence for A is

$$\dim(A_n) = \dim(F_n) - \dim(J_n) = g^n - \dim(J_n)$$

so we need only show that $\dim(J_n)$ can be computed in exponential time. Let I denote the input to problem (D) so that I N-encodes the algebra A and the integer n.

As a vector space, J_n is spanned by the finite set $\{u\alpha_k v\}$ as k runs from 1 to r and as (u,v) runs through all pairs of monomials in the $\{x_i\}$ for which $|u| + |v| = n - 2$. This finite set has cardinality

$$s = (n-1)rg^{n-2} < (\ell(I)^2)(\ell(I))^{\ell(I)} < 2^{\ell(I)^2} \qquad [24]$$

From the expressions (15) we obtain s expressions of the form

$$u\alpha_k v = \sum_{i=1}^{g}\sum_{j=1}^{g} c_{ijk}ux_ix_jv \qquad [25]$$

Using (25), we may construct an $s \times g^n$ matrix C. Its rows are indexed by triples (u,k,v), its columns are indexed by the set of length n monomials w in F, and its entries are either c_{ijk} (if $w = ux_ix_jv$) or zero (if not). The row space of C is easily identified with J_n. We need only compute the rank of C.

Since C is an $s \times g^n$ matrix and both s and g^n are bounded by $2^{\ell(I)^2}$, we can write down C in exponential time. To find the rank of C, use Gaussian elimination. This requires fewer than $(2^{\ell(I)^2})^3 = 2^{3\ell(I)^2}$ operations. Our computation therefore requires only exponential time.

Combining this result with Lemma 6.2 and Theorem 5.8, we have at once

Corollary 6.5

Each of the 12 problems (A) through (L) described in Section 5 is computable in exponential time.

Combining this corollary with Lemma 6.3, we have these problems nicely bracketed between "#$\mathcal{P}$-hard" and "computable in exponential time."

<u>Philosophical Remarks</u>. What does it all mean? Rational homotopy groups can be computed in exponential time. But exponential time is generally considered too slow for practical implementation.

It is virtually certain that rational homotopy cannot be computed in polynomial time. If it could, then $\mathcal{P}$ would equal #$\mathcal{P}$, contradicting a widely believed conjecture. I believe further that problems (A) through (L) are strictly harder than anything in #$\mathcal{P}$. That is to say, problem (L) is (I believe) not Turing reducible to anything in #$\mathcal{P}$. Even if $\mathcal{P}$ were to equal #$\mathcal{P}$, rational homotopy might still be uncomputable in polynomial time.

Experience tells us that integral homotopy groups are considerably less accessible than rational homotopy groups, so integral homotopy could require exponential time or more. One runs into difficulty just trying to set up the problem of computing integral homotopy. We are thrown back upon the issue of how one N-encodes, with even moderate efficiency, an arbitrary simply connected finite CW complex. One typically needs more information than is available in the Quillen model. [A notable exception occurs for the subclass of spaces having cells in dimensions two and four only. These spaces are determined up to homotopy type by the input to problem (H).] Nevertheless, it is safe for practical purposes to declare that the problem of computing the whole $\pi_n(X)$ is #$\mathcal{P}$-hard.

Interestingly, if one considers a fixed simply connected Y and asks for $\pi_n(Y)$ as a function of n only, Curtis [11,12] showed that $\pi_n(Y)$ can be obtained using a semisimplicial group model for Y whose lower central series has been truncated at depth 2^n. This suggests that for each Y, the function $\theta(n) = \pi_n(Y)$ might be computable in $O(c^n)$ time. Expression (24) for s and the subsequent discussion show that the rational homotopy of a fixed Y can definitely be obtained in $O(c^n)$ time.

We have developed other positive results as well. In the "no news is good news" category, our methods give no lower bound at all on the complexity of the (integral) homotopy groups of spheres, so hope remains for that problem. More affirmatively, our careful consideration of N-encodings suggests that we may have discovered the relative importance of the various inputs. For instance, doubling n has approximately the same effect on the computational difficulty of $\pi_n(X) \otimes Q$ as does squaring the coefficients used in constructing X, since either operation potentially doubles $\ell(I)$. Lastly, Theorems 3.5 and 4.8 greatly expand our repertoire of series known to be rationally related to Hilbert or to Poincaré series.

In summary, computing homotopy groups of arbitrary simply connected spaces is a genuinely computationally difficult problem. We cannot expect that rapid algorithms for it will ever be discovered. Somewhat anticlimactically, we close with Ed Brown's observation to this effect, prophetically written more than 30 years ago [9, p. 1]:

> While the procedures developed for [computing homotopy groups] are finite, they are much too long to be considered practical.

ACKNOWLEDGMENT

I could not have written this chapter if Michael Sipser had not patiently explained to me what #$\mathcal{P}$ means, and the chapter would not now be in print if Lisa Court and Judy Romvos had not worked so diligently to type it. This work was partially supported by National Science Foundation grant 8509191-DMS.

REFERENCES

1. J. F. Adams and P. J. Hilton, On the chain algebra of a loop space, Comm. Math. Helv. 30, (1955), 305–330.
2. L. Adelman and F. T. Leighton, An $O(n^{1/10.89})$ primality testing algorithm, Math. Comput. 36(153), (1981), 261–266.
3. D. J. Anick, Diophantine equations, Hilbert series, and undecidable spaces, Ann. Math. 122, (1985), 87–112.

4. D. J. Anick, Generic algebras and CW complexes, Algebraic Topology and Algebraic K-Theory (W. Browder, ed.), Annals of Math. Study, 113, Princeton University Press, (1987), 247–321.
5. D. J. Anick, Non-commutative graded algebras and their Hilbert series, J. Algebra 78(1), (1982), 120–140.
6. D. J. Anick and T. H. Gulliksen, Rational dependence among Hilbert and Poincaré series, J. Pure Appl. Algebra 38, (1985), 135–157.
7. H. J. Baues and J. -M. Lemaire, Minimal models in homotopy theory, Math. Ann. 225, (1977), 219–242.
8. D. Bayer and M. Stillman, On the complexity of computing syzygies, preprint.
9. E. H. Brown, Finite computability of Postnikov complexes, Ann. Math. 65, (1957), 1–20.
10. E. Čech, Höherdimensionale Homotopiegruppen, Verhandlungen des Internationalen Mathematikerkongress, Zürich, 1932, Orel Füssli, Zürich (1932), 203.
11. E. B. Curtis, Lower central series of semi-simplicial complexes, Topology 2, (1963), 159–171.
12. E. B. Curtis, Some relations between homotopy and homology, Ann. Math. 82, (1965), 386–413.
13. R. Froberg, T. Gulliksen, and C. Löfwall, Flat families of local Artinian algebras with an infinite number of Poincaré series, Algebra, Algebraic Topology and their Interactions, Lecture Notes in Math., 1183, Springer-Verlag, New York (1986), 170–191.
14. T. H. Gulliksen, Reducing the Poincaré series of local rings to the case of quadratic relations, Algebra, Algebraic Topology and their Interactions, Lecture Notes in Math., 1183, Springer-Verlag, New York (1986), 195–198.
15. J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, Mass. (1979).
16. W. Hurewicz, Beiträge zur Topologie der Deformationen, Nederl. Akad. Wetensch. Proc. Ser. A 38, (1935), 112–119, 521–528; 39, (1936), 117–126, 215–224.

17. C. Jacobsson, Finitely presented graded Lie algebras and homomorphisms of local rings, J. Pure Appl. Algebra 38, (1985), 243–253.

18. D. E. Knuth, Semi-numerical algorithms, The Art of Computer Programming, vol. 2, 2nd ed., Addison-Wesley, Reading, Mass. (1981).

19. G. Levin, Local rings and Golod homomorphisms, J. Algebra 37, (1975), 266–289.

20. S. MacLane, Homology, Die Grundlehren der Mathematische Wissenschaften in Einzeldarstellungen, 114, Springer-Verlag, Berlin (1963).

21. G. L. Miller, Riemann's hypothesis and tests for primality, J. Comput. Syst. Sci. 13(3), (1976), 300–317.

22. J. Milnor and J. C. Moore, On the structure of Hopf algebras, Ann. Math. 81, (1965), 211–264.

23. D. G. Quillen, Rational homotopy theory, Ann. Math. 90, (1969), 205–295.

24. R. Sedgewick, Algorithms, Addison-Wesley, Reading, Mass. (1983).

25. J. -P. Serre, Groupes d'homotopie et classes de groupes abéliens, Ann. Math. 58, (1953), 258–294.

26. C. E. Shannon and J. McCarthy, eds., Automata Studies, Study No. 34, Princeton University Press (1956).

27. D. Sullivan, Infinitesimal computations in topology, Publ. Math. I.H.E.S. 47, (1977), 269–331.

28. L. G. Valient, The complexity of computing the permanent, Theoret. Comput. Sci. 8, (1979), 189–201.

29. K. Weld, Computability of Homotopy Groups of Nilpotent Complexes, Thesis, Graduate School and University Center of City University of New York (1984).

30. G. W. Whitehead, Elements of Homotopy Theory, Springer-Verlag, New York (1978).

2

Geometry of the Hopf Mapping and Pinkall's Tori of Given Conformal Type

THOMAS F. BANCHOFF

Brown University
Providence, Rhode Island

One of the most fertile geometrical examples in mathematics is the Hopf mapping from the 3-sphere S^3 to the 2-sphere S^2. On the occasion of the Conference on Computers in Geometry and Topology at the University of Illinois at Chicago, we considered three aspects of this mapping which are particularly well suited to investigation by computer graphics.

The study of Hamiltonian dynamical systems can be motivated and illustrated by a linear system which leads to the Hopf fibers as circular orbits lying on a constant energy 3-sphere. This aspect has been reported at length elsewhere in the article of Hüseyin Koçak, Fred Bisshopp, David Laidlaw, and the author [1].

Regular polytopes in 4-space can be decomposed into rings of polyhedra which correspond to solid tori which are preimages of cells on the 2-sphere under the Hopf mapping. This aspect has appeared in the author's article in the proceedings of the conference Shaping Space [2].

The third topic considered was an elementary presentation of a remarkable construction by Ulrich Pinkall which determines the conformal structures of tori obtained by lifting a closed curve on S^2 under the Hopf mapping. This short note is an exposition of Pinkall's result which is well-suited to interactive computer graphics investigation.

In [3], Pinkall showed that the inverse image under the Hopf mapping of a simple closed curve on S^2 is a flat torus on S^3 which is conformally equivalent to a parallelogram in the plane with basis vectors