THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Computer and Communication Sciences Department

Technical Report

COMPUTER SIMULATION OF A LIVING CELL: INTERDISCIPLINARY SYNERGISM

Roger Weinberg

with assistance from:

Department of Health, Education, and Welfare
National Institutes of Health
Grant No. GM-12236
Bethesda, Maryland

and

National Science Foundation
Grant No. GJ-519
Washington, D.C.

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

June 1970

ACKNOWLEDGEMENTS

A synergestic profit was gained by close collaboration with Mr. Erik D. Goodman, Dr. Bernard P. Zeigler, and Mr. Richard A. Laing.

Dr. J. A. Jacquez and the Biomedical Data Processing group provided me with a milieu in which I could begin studies at The University of Michigan. Dr. John H. Holland encouraged me to enter the strange new field of computer and communication sciences from genetics, and taught at a level of excellence which permitted me to remain a student for the requisite initiation period. E. Stewart Bainbridge, and Daniel J. Cavicchio, Jr. tutored me with patience and skill through many perilous passages. Ronald Brender, Thomas Schunior and John Foy gave willingly of their computing knowledge. Dr. Robert B. Helling and Dr. Prasanta Datta kept me abreast of key happenings in biochemistry and genetics. Dr. Arthur W. Burks understood my academic problems at all times, and inspired me by his leadership of the Logic of Computers Group and the Department of Computer and Communication Sciences at The University of Michigan. Dr. H. H. Swain catalyzed my work with his editorial comments. Mr. Thomas Dawson administered the affairs of the Logic of Computers Group with morale-boosting zeal. Mrs. Marilyn Abramson and Miss Jan McDougall typed with skill and dedication, and made many helpful editorial comments. I have erred at points in spite of all of these accomplished and generous helpers because I am human.

This research was supported in part by National Science Foundation Grant No. GJ-519, Department of Health, Education, and Welfare, National Institutes of Health Grant No. GM-12236.

TABLE OF CONTENTS

Chapter		Page
I	COMPUTER SIMULATION OF A LIVING CELL EXTENDS MOLECULAR BIOLOGY TO SYSTEMS ANALYSIS	1
	Interdisciplinary Synergism: Biochemistry, Molecular Biology, Automata Theory, Computer Systems	1
	Literature Survey	4
	Molecular Biology	4
	System Theory	8
II	COMPUTER SIMULATION OF A LIVING CELL	10
	Model of a Living Cell	10
	Metabolic Topology	13
	Functional Relationships	13
	Experimental Data	13
	Computer Program of a Living Cell	
	Summary	16
	Simulated Environments: Mineral Glucose, Casamino Acids, Broth	23
	DNA Replication	24
	Repression	24
	Allosteric Modification of Enzymes	26
	Overall Program	27
III	ALLOSTERIC MODIFICATION: HOMOMORPHISM: SIMULATION	28
	Homomorphisms of Systems; Basic Definitions	31
	Allosteric Modification; a Homomorphic Model	34
	Calculating Catalytic Activity of Model Enzymes	38
	Validity of Homomorphic Model	43
	Computer Implementation of Allosteric Modification	44

TABLE OF CONTENTS CONT'D

Chapter		Page
IV	STRUCTURE OF THE SIMULATION, AN EXAMPLE OF COMPLEXITY REDUCTION BY AGGREGATION (Zeigler & Weinberg, 1970)	54
	Computer Simulation Complexity Measures	54
	Complexity Reduction by Aggregation	57
	Aggregated Models and Simulation	63
V	RESULTS OF THE SIMULATION	67
	The Simulated Cell Grows Correctly in Three Environments	78
	The Simulated Cell Grows Correctly in a New Medium	78
	Allosteric Modification is Necessary for Shifts to Richer Media	79
	Oscillating Concentrations Reflect Metabolic Stability	85
	Discussion of the Results:	89
	Conclusions from the Results	92
VI	COMPUTER SIMULATION OF EVOLVING DNA	93
	Genetic Operators	93
	Populations	96
	Holland's Formal Adaptive System	98
	Selection	99
	Genetic Loci	111
	Simulated Phenotype from Genotype	113
	Simulated Crossing Over	115
	Simulated Inversion	120
	Simulated Mutation	121
	Other Genetic Operators; Duplication and Lumping Loci	122
	Generality of Reproductive Scheme	124

TABLE OF CONTENTS CONT'D

Chapter		Page
VII	A CELL SPACE EMBEDDING OF SIMULATED LIVING CELLS (Goodman, Weinberg and Laing, 1970)	125
	Summary	125
	Introduction	125
	The Model of DNA Replication in ONECELL	127
	Implementing the Replication Model in ONECELL	128
	Description of CELLSPACE	130
	Background	130
	Modelling a Colony	131
	Interactions Among Cells	134
	Computer Implementation of ONECELL and CELLSPACE	135
	Potential Applications	140
	Preliminary Results	142
VIII	CONCLUSIONS	144
	The Simulation is Realistic and Useful	144
	Allosteric Modification is Necessary to Shift to Richer Media	144 144
	Equivalence Classes with Substitution Property Provide a Validation Measure for the Simulation	144
	Complexity Can Be Reduced in a Valid Aggregated Model	144
	Evolution of DNA Has Been Outlined for Simulation	144
	DNA Reproduction is Effective for Intracellular and Intercellular Information Transfer	145
APPENDIX TO CHAPTER II		146
	Variables in a Computer Simulation of a Living Cell	146
BIBLIOGRAPHY		151

LIST OF FIGURES

Figure		Page
1.	Computer Simulation of a Model Formulated from Experimental Measurements.	2
2.1	Some of the Primary Molecular Constituents and Metabolic Pathways in the E. coli Cell.	11
2.2	The Primary Chemical Pools Used in the Model.	12
2.3	Model of a Living Cell Used for the Computer Simulation.	14
2.4	Summary of Program.	17
2.5	Differential Equations.	18
2.6	New Concentrations.	19
2.7	Cell in Environment 1 (Minimal Medium).	20
2.8	Cell in Environment 2 (Minimal Medium + Amino Acids).	21
2.9	Cell in Environment 3 (Broth).	22
2.10	Repression and Allosteric Inhibition.	25
3.1	Repression and Allosteric Inhibition(FORTRAN).	47
3.2	Preliminary Rate Constants K(1), ..., K(14) for Flow Rates Between Pools.	48
3.3	Storing Different Variables for the Cell in Different Environments.	49
3.4	Calculations for Allosteric Inhibition Necessary to Fit Data for Real Cells.	49
3.5	Simulated Allosteric Modification Within the Overall Program.	50
3.6	The Cell Successfully Adjusted its Enzyme Rate Constants by Simulated Allosteric Modification.	51
3.7	Steady-State Concentrations.	53
4.1	Possible State-Spaces and Transition-Functions for Models of the Living Cell.	56
4.2	Illustrating the Aggregation Process.	59

LIST OF FIGURES CONT'D

Figure		Page
5.1	Logarithm of Various Quantities in a Growing Culture.	68
5.2	ATP During Simulated Shift-Up.	69
5.3	ADP During Simulated Shift-Up.	70
5.4	DNA During Shift-Up.	71
5.5	Protein During Shift-Up.	72
5.6	Ribosomal RNA During Shift-Up.	73
5.7	ATP During Simulated Shift-Down.	74
5.8	DNA During Shift-Down.	75
5.9	Ribosomal RNA During Shift-Down from Broth to Minimal Glucose.	76
5.10	Protein/Cell During Shift-Down.	77
5.11	Simulated Growth in Low Glucose Concentration.	80
6.1	The Chrom Array.	94
6.2	Evolution of Strings.	102
6.3	Evolution of Adaptive Genetic Programs.	103
6.4	Description of Strings Referenced When Third Dimension of CHROM Array Ranges from 1 to 40.	105
6.5	Adaptive Genetic Programs.	114
6.6	Non-adaptive Genetic Program Directing the Evolution of Adaptive Genetic Programs as they Operate on Populations of Strings.	116
7.1	DNA Replication Model for 60 and 20 Minute Cells.	129
7.2	Preliminary Experiments Using ONECELL + CELLSPACE.	133

ABSTRACT

COMPUTER SIMULATION OF A LIVING CELL

by
Roger Weinberg

Chairman: John H. Holland

The computer simulation of a living cell describes and predicts the biochemical behavior of a living system adapting to different environments. To describe a process as complicated as life I have chosen a very simple organism, *Escherichia coli*. This one celled organism embodies the basic principles of life. *E. coli* has a very simple developmental cycle, and it has fewer specialized functional systems than higher organisms. It is therefore an ideal system in which to study the basic principles of life, because many complicating phenomena are absent from its makeup. Furthermore, microbiologists have studied it in detail in the laboratory, and have analyzed the theoretical basis of many of the separate subsystems in its metabolism. My simulated living cell incorporates these separate systems into one program, and studies their interaction in the entire organism. We constantly compare simulated results with laboratory results from the literature. We use these comparisons to test the simulation and the hypotheses on which it is based. When the simulated cell behaves incorrectly, we modify the assumptions used in its formulation. In this way we can use the simulation to test and refine current hypotheses concerning cellular control of biochemical reaction rates, cell division, and development.

I wrote a computer simulation of a living cell in 1968 in order to extend molecular biology to system studies (Weinberg, 1968a). The simulated cell continued to prove useful in studies of biochemistry, automata theory, and development. In carving out the theoretical universe in which to model on paper, and in simulating the model on a digital computer, the concept of homomorphism helped us. It helped us to simplify the original system, and it enabled us to judge the validity of both the simplified model and its computer simulation (Weinberg, 1968a, b; Weinberg and Zeigler, 1970; Zeigler and Weinberg, 1970). I took advantage of the notion of homomorphism to form my state space. The concomitant idea of equivalence classes with substitution property was useful in testing the accuracy of our simulation (Weinberg and Berkus, 1969a, b). Automata theoretic notions were abstracted to describe the structure of the simulation, and to develop a general theory of computer simulation (Zeigler and Weinberg, 1970; Zeigler, forthcoming.)

Results of the simulation indicated that the role of allosteric modification of enzyme action was to permit shifts from poorer to richer media (Weinberg and Zeigler, 1970). Following these initial results, the biochemical simulation of a single living cell was embedded in a cell space, where it was used to study both 1) interactions between the replication mechanism and metabolism in a single cell, and 2) communication of information between cells in an interacting population (Goodman, Weinberg and Laing, 1970).

CHAPTER I

COMPUTER SIMULATION OF A LIVING CELL EXTENDS MOLECULAR BIOLOGY TO SYSTEMS ANALYSIS

Interdisciplinary Synergism: Biochemistry, Molecular Biology, Automata Theory, Computer Systems: The computer simulation of a living cell describes and predicts the biochemical behavior of a living system adapting to different environments. To describe a process as complicated as life I have chosen a very simple organism, *Escherichia coli*. This one celled organism embodies the basic principles of life, but it is relatively uncomplicated. *E. coli* has a very simple developmental cycle, and it has fewer specialized functional systems than higher organisms. It is therefore an ideal system in which to study the basic principles of life because many complicating phenomena are absent from its makeup. Furthermore, microbiologists have studied it in detail in the laboratory, and have analyzed the theoretical basis of many of the separate subsystems in its metabolism. My simulated living cell incorporates these separate systems into one program, and studies their interaction in the entire organism. We constantly compare simulated results with laboratory results from the literature. We use these comparisons to test the simulation and the hypotheses on which it is based. When the simulated cell behaves incorrectly we modify the assumptions used in its formulation. In this way we can use the simulation to test and refine current hypotheses concerning cellular control of biochemical reaction rates, cell division, and development (Figure 1).

To simulate an entire organism and compare the simulation to real

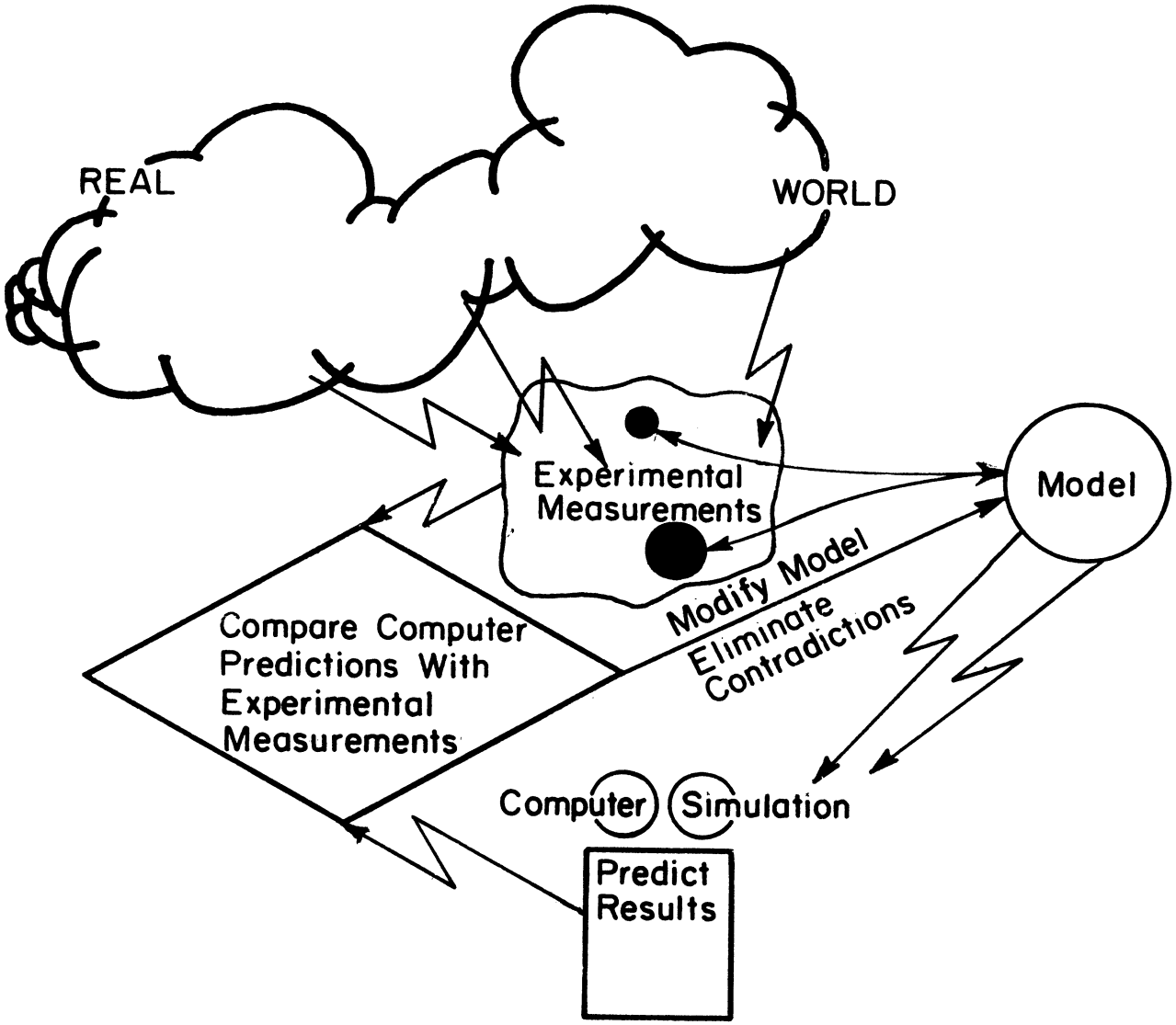


Figure 1. Computer Simulation of a Model Formulated From Experimental Measurements.

world data required tools from biochemistry, automata theory, molecular biology, and computer systems.

In order to reap the synergistic reward of bringing together these various areas, I have collaborated closely with colleagues in the diverse fields which make up computer science at The University of Michigan.

The synergism of these collaborations is apparent in the following short history. I wrote a computer simulation of a living cell in 1968 in order to extend molecular biology to system studies (Weinberg, 1968a). The simulated cell continued to prove useful in studies of biochemistry, automata theory, and development. In carving out the theoretical universe in which to model on paper, and in simulating the model on a digital computer, the concept of homomorphism helped us. It helped us simplify the original system, and it enabled us to judge the validity of both the simplified model and its computer simulation (Weinberg, 1968a, b; Weinberg and Zeigler, 1970; Zeigler and Weinberg, 1970). I took advantage of the notion of homomorphism to form my state space. The concomitant idea of equivalence classes with substitution property was useful in testing the accuracy of our simulation (Weinberg and Berkus, 1969a, b). Automata theoretic notions were abstracted to describe the structure of the simulation, and to develop a general theory of computer simulation (Zeigler and Weinberg, 1970; Zeigler, forthcoming).

Results of the simulation indicated that the role of allosteric modification of enzyme action was to permit shifts from poorer to richer media (Weinberg and Zeigler, 1970). Following these initial results, the biochemical simulation of a single living cell was embedded in a cell space, where it was used to study both 1) interaction between the

replication mechanism and metabolism in a single cell, and 2) communication of information between cells in an interacting population (Goodman, Weinberg and Laing, 1970).

I will present the model and simulation, automata theory, biochemistry, outline for evolving DNA, cell space studies, and then summarize the conclusions drawn from these works. The care with which the original equations for the simulated living cell were formulated and compared with actual experimental data makes possible further applications of the simulation in suggesting hypotheses, and predicting experimental results from extant theories.

Literature Survey: Molecular Biology: A computer simulation of a living cell capable of analyzing interacting subsystems in the cell has been made possible by the enormous advances in molecular biology in the past fifteen years. These studies in molecular biology have greatly increased our understanding of the basic subsystems in the living cell. My own genetic studies have passed from classical analyses of higher organisms such as fruit flies (e.g., Weinberg, 1954) through studies of molecular biology using lower organisms (e.g., Weinberg, 1962; Weinberg and Boyer, 1965; Boyer, Englesberg and Weinberg, 1962; Englesberg, Anderson, Weinberg, Lee, Hoffee, Huttenhauer and Boyer, 1962) back to studies of interacting systems in whole organisms (e.g., Weinberg, 1968a, b). The actions of chemicals on specific DNA bases has been calculated for bacteria (e.g., Weinberg and Boyer, 1965; Boyer, Englesberg and Weinberg, 1962), following pioneer work by Freese and co-workers in bacterial viruses (in Davis et al, 1968).

In a brilliant series of experiments by a host of experimenters, DNA was found to be the molecule on which the cell records information

to be passed on to the next generation. M. H. F. Wilkins, James D. Watson and Francis H. C. Crick established the precise structure of the DNA molecule. The molecular code used by DNA to record hereditary information has been successfully analyzed. Viable hypotheses have been formulated for translation of genetic information into cellular activity and structure. Experiments have yielded support for specific theories concerning the control of cellular activity at the level of DNA function (e.g., Helling and Weinberg, 1963; Englesberg, Anderson, Weinberg, Lee, Hoffee, Huttenhauer and Boyer, 1962). Control mechanisms have also been discovered which control a given cellular process by modification of molecules far removed from DNA (e.g., enzymes which catalyze amino acid production may be temporarily inactivated when enough amino acids are already present in a cell).

In particular, much is known about the biochemical behavior of bacterial cells (e.g., Hoffee, Weinberg and Englesberg, 1961; Weinberg, 1960a; Weinberg, 1960b). Biological laboratory studies have been done elucidating control of DNA replication in bacteria (Eberle and Lark, 1969; Clark, 1968; Helmstetter, Cooper, Pierucci and Revelas in Frisch, 1969; Smith and Pardee, 1970). Indeed, the whole 1968 Cold Spring Harbor Symposium in Quantitative Biology was devoted to papers concerning replication of DNA in micro-organisms (edited by Frisch, 1969). Lark has reviewed initiation and control of DNA synthesis (1969), and the subject is covered in books by Mandelstam and McQuillen (1968), Hayes (1968), and Davis et al (1968).

Control of enzyme production by repression of messenger RNA production by the DNA is still being studied (Umbarger, 1969) and the original hypotheses are being modified. Modification of enzymes already present in the

cell, as a control mechanism, is reviewed by Umbarger (1969), and Datta (1969). The relationship of this type of control to energy relationships in the cell has been reviewed by Atkinson (1966), discussed in *Control of Energy Metabolism* (Chance et al 1965), and analyzed by Murray and Atkinson (1968). These works show that the ATP/ADP ratio is involved in energy control relationships. Convenient presentation of data on metabolic pathways appear in books by Bernhard (1968), Westley (1969), Reiner (1968, 1969) and Mahler and Cordes (1966).

Mathematical analyses of enzyme modification utilizing computer techniques have been published by Walter (1969a, 1969b), Cennamo (1969), Griffith (1968), Heinmets (1964), and Yeisley and Pollard (1964). Formal aspects of self-reproducing systems are described in Waddington (1969), Burks (1969), Codd (1968), Mesarovic (1968), and von Neumann and Burks (1966). Techniques for automata-theoretic studies, numerical analysis, and also computer simulation are well described in Gordon (1969), Mize and Cox (1968), IBM Corporation Scientific Subroutine Package (1969), Knuth (1969), Wendroff (1969), Ginzburg (1968), Kalman, et al (1969) and Ulam (1966).

The interactions of various subsystems analyzed by molecular biologists are becoming objects of study (Roosen Runge, 1967; Rosen, 1968). These subsystems interact among themselves and with the environment external to the cell.

In a study of interacting systems, the large, high-speed digital computer is an enormous aid. One can write down hypotheses in the form of logical and analytic equations. These equations form the program. Environmental conditions can be given to the computer as input data. If the program represents a cell, the program generates cell behavior as output.

Availability of experimental and theoretical analyses of the systems operative in living and reproducing organisms, as well as excellent presentations of powerful and convenient computer techniques make a computer simulation of a living cell a logical endeavor at this time. Weinberg and Berkus (1969), Weinberg (1969a, 1969b), and Stahl (1967) have modelled living cells as computer programs.

To construct our model of a whole cell (Weinberg, 1968a), we created the metabolic network consisting of a small number of chemical pools as shown in Figures 2.1, 2.2. The pools were to represent the network behavior in an aggregate manner. As shown in Figure 2.3 the metabolic and informational transactions established among the pools was to reflect in a coarser way the underlying relations among their constituents. A detailed description of the basic model is given in Chapter II. Some results of the simulation are reported in Chapter IV (Weinberg and Zeigler, 1969).

This represents a novel approach to models of the metabolism of the cell. Some simulation studies (Garfinkel, 1966; Yeisley and Pollard, 1964; Heinmetz, 1964; Savageau, 1969, 1970) are concerned with simulating particular pathways in detail rather than attempting to simulate the global cellular operation.

Schulze and Gerhardt (1969), on the other hand, used simulation to study populations of cells rather individual cells. Still another approach to simulation is used by Stahl (1967). Stahl represents each molecule as a separate variable in his simulated cell. He is severely limited with respect to the quantity of results he can obtain. He is unable to make any quantitative comparisons between his simulated results and real laboratory data. Our present computer simulation of a living cell is the

first successful effort to compare the predictions of hypotheses concerning a complete, functional cell with detailed laboratory data.

The connection between molecular controls and evolutionary mechanisms has been outlined for our computer simulation (Weinberg and Berkus, 1969). We expressed basic genetic mechanisms as a computer program (Strickberger, 1969; Kimura, 1964), and also made use of theoretical analyses connecting econometric studies (Gale, 1967) to a general theory of adaptive systems (Holland, 1969a, 1969b). Relationships have been drawn between regulatory mechanisms in microbial cells, and in higher cells by Mitchison (1969), Tsanev and Sendov (1969), Comings and Kakefuda (1968), Britten and Davidson (1969), Gause (1966), and Heinmets (1966), making plausible the extension of computer simulation studies of a bacterial cell to studies of cancer in higher organisms (Tsanev and Sendov, 1969; Goodman, Weinberg and Laing, 1970).

Literature Survey: System Theory: The problem immediately confronting any attempt to model a biological system such as a living cell for computer simulation is the enormous complexity inherent in such systems. We present an approach to this problem based upon the concepts of homomorphism and aggregation (Weinberg and Zeigler, 1970; Zeigler and Weinberg, 1970). A mathematical formulation will be developed in Chapter III to apply these concepts to our simulation of a living cell.

Ulam (1966), and Klir and Valach (1967) have pointed out the important connection between simulation and model construction, and the mathematical notion of homomorphism. One may attempt to achieve such homomorphic mappings by lumping entities together; this constitutes the basic character of our model. The importance of this process of aggregation has been known to economists for some time (e.g., Green, 1964; Ijiri, 1968).

However, the concepts of aggregation and homomorphism have not been much appreciated by simulators and modellers, especially those of biological systems. They are not mentioned, for example, in the extensive discussion of Fein (1966), nor by Rosen (1968). In Chapter IV we show that at the conceptual level, these concepts help explicate the notions of model complexity and adequacy (Zeigler and Weinberg, 1970). At the more practical level, they suggest and justify techniques for the reduction of complexity and for the testing of model adequacy in the context of biological system simulations.

Simon and Ando (1961) presented a mathematical formulation of aggregation of variables for dynamic systems and studied a particular class of systems (nearly decomposable systems) whose long term behavior satisfied these conditions. An informal discussion of the relationship of these systems to the complexity of natural systems was given by Simon (1962).

Since 1961, there has arisen an extensive study of the structural and behavioral relations which hold between automata (Hartmanis and Stearns, 1968; Krohn and Rhodes, 1965; Arbib, 1969; Zeigler, 1968) and the relation of automata to general systems (Kalman, Falb, and Arbib, 1968). With these developments it is possible to give a unified mathematical formulation of aggregation and its relation to the more basic notion of the homomorphism of systems. Based on this formulation and work in system complexity (Zeigler, 1968, 1969) one can demonstrate precisely how the complexity (as determined by measures relevant to computer implementation) can be reduced in going from a system to a homomorphic image (Zeigler, 1970).

The basic structure underlying our simulated living cell illustrates a valid reduction of complexity from the original measurement space available from experimental data.

CHAPTER II

COMPUTER SIMULATION OF A LIVING CELL

Model of a Living Cell: The model of a living cell used for the simulation will be described, and then the routines in the computer simulation which realize the model will be presented. Often the programming routines will merely repeat in different notation the verbal description of the model. They will both be listed for clarity.

In writing a simulation of the *Escherichia coli* cell (Weinberg, 1968a, 1969b) I recognized that in even the simplest of cells there are more than 3000 different kinds of molecules in an intricate spatial and functional relationship. These molecules co-exist in a complex metabolic network with superposed genetic and enzymatic controls. This complexity had to be drastically reduced to enable a model to be simulated on a digital computer. It is important to note that while electronic computers continue to attain increasingly greater memory stores and rates of operation, this capacity is very much limited when one considers simulating *in detail* the operation of a natural system.

To construct our model we created the metabolic network consisting of a small number of chemical pools as shown in Figure 2.1. The pools were to represent the network behavior in an aggregate (i.e., lumped) manner. As shown in Figure 2.2, the metabolic and informational transactions established among the pools were to reflect in a coarser way the underlying relations among their constituents.

This represents a novel approach to models of the metabolism of the cell since other simulation studies (Garfinkel, 1966; Yeisley and Pollard, 1964; and Savageau, 1969) are concerned with simulating parti-

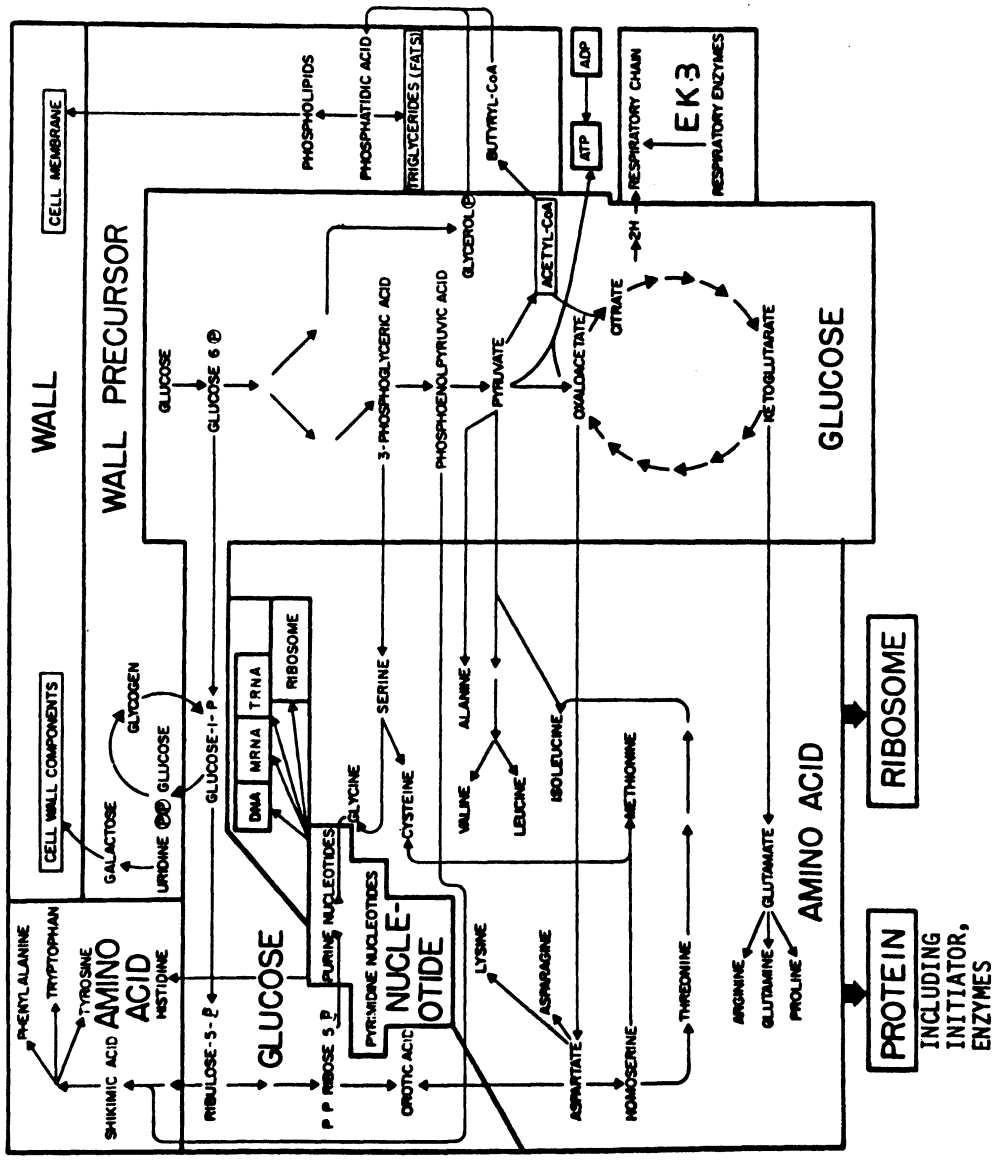


Figure 2.1 Some of the primary molecular constituents and metabolic pathways in the E. coli cell are shown. These constituents are shown grouped together into "pools", e.g., GLUCOSE, AMINO ACIDS, etc., which serve as the entities for a model of the cell. Steady-state growth is assumed. Therefore, there are few degradative pathways shown.

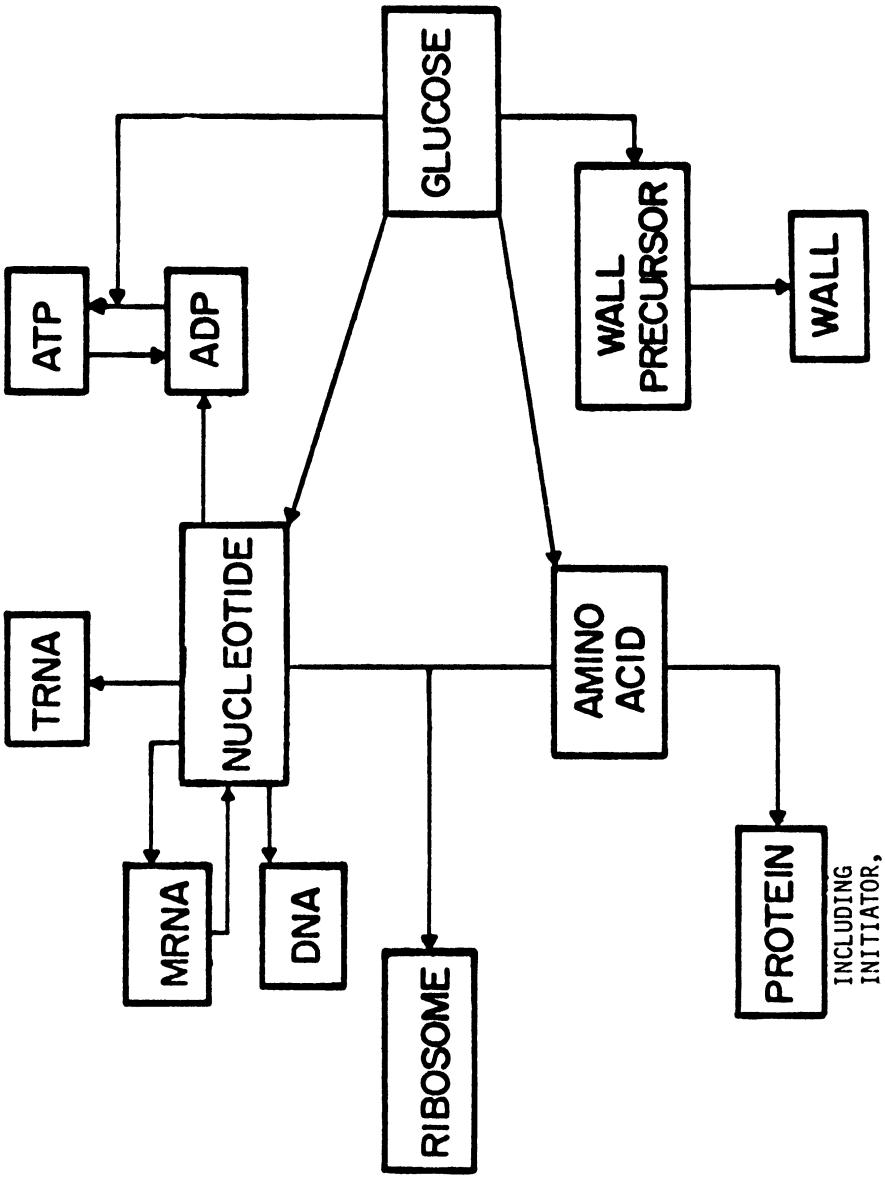


Figure 2.2 The primary chemical pools used in the model are shown. The arrows indicate the direction of the flow of material between pools. The topology of this flow preserves the metabolic topology of Figure 1, in a manner explained in the text.

cular pathways in detail rather than attempting to simulate the global cellular operation. Although whole cell simulations have been written (Heinmetz, 1964 and Stahl, 1969) they were not designed to produce results which could be compared in detail with results from laboratory experiments. On the other hand, the results of our simulation were compared to laboratory data for *Escherichia coli*.

Certain considerations guided our aggregation of molecules into pools:

Metabolic topology led us to combine into a single entity only those molecules which could be drawn adjacent to one another on the metabolic map (Figures 2.1, 2.2). For example, in a pathway A→B→C, chemicals A, B, C might be partitioned into A, and BC, but would not be lumped into AC, and B.

Functional relationships between groups of molecules were extremely important, and the molecules lumped together in any one model entity were, in some way, functionally a unit (Figure 2.3). Thus, all the molecules produced in the breakdown of carbohydrate to CO_2 and water to produce energy were lumped in this model since they could be considered functionally as molecules forming a chemical pathway used to produce energy. Later models will employ more refined partitions, such as partitions which capture the subtle and important relationships between molecules at different points in the glycolytic pathway, citric-acid cycle, and cytochrome system.

Experimental data were often available for large chemical pools, e.g., products of glycolysis and the citric-acid cycle. A pool for which data was available became a logical candidate for inclusion as a single entity in our model of the cell. The different kinds of molecules making up

Figure 2.3 MODEL OF A LIVING CELL USED FOR THE COMPUTER SIMULATION

STATE

COORDINATE = ENTITY	RANGE OF VALUE OF COORDINATES = ATTRIBUTE OF ENTITY
Pools of Chemicals, PRDC(1),..., PRDC(10)	Concentration of pool
Enzymes, EK(1),...,EK(10)	Concentration of enzyme
Messenger RNA, RNK(1),..., RNK(10)	Concentration of RNA
Genetic Apparatus	Amount of DNA, site of replication, number of genes in cell for producing sites for replication
Cell Volume	Total volume of the cell
Cell Number	Number of cells represented in the culture

TRANSITION FUNCTION
(for calculating the state of the system in the next time step from the present state)

A. *The differential and boolean equations* relating concentrations of variables at a given time to the concentrations of those variables DT seconds later. e.g. for AA, the amino acid pool, one needs enzyme EK(2) to catalyze the production of AA from glucose, and one uses ATP as an energy source. At the same time, AA is lost as it is used for the production of RIB and PRTN.

1. DAA = K(2)*GLUC*DT*EK(2)*ATP - 1.E6/102.*DRIB - (4.E4/102.)*DPRTN
   production of AA from GLUC; loss of AA to RIB; loss of AA to PRTN

2. RNK(2) produced EK(2) from AA under the direction of DNA, using ATP for energy.

3. DEK(2) = K(7)*AA(RNK(2)/MRNAO)*DT*EK(7)*ATP

4. RNK(2) itself was produced from NUC under the direction of DNA, catalyzed by EK8, using ATP for energy. RNK(2) decayed spontaneously at the same time, producing some loss of RNK(2) already present.

5. DRNK(2) = (K8K(2)*NUC*DNA*EK(8)*ATP - KDRNK*RNK(2))*DT
   production of RNK(2); decay of RNK(2)

B. *Allosteric modification* of enzymes simulated by modifying the rate constant which characterizes all different forms of any enzyme associated with a particular reaction.

C. *Repression* of messenger RNA directing the production of a particular enzyme

D. *Genetic behavior* of DNA in response to the state of the cell

E. *Permeability*

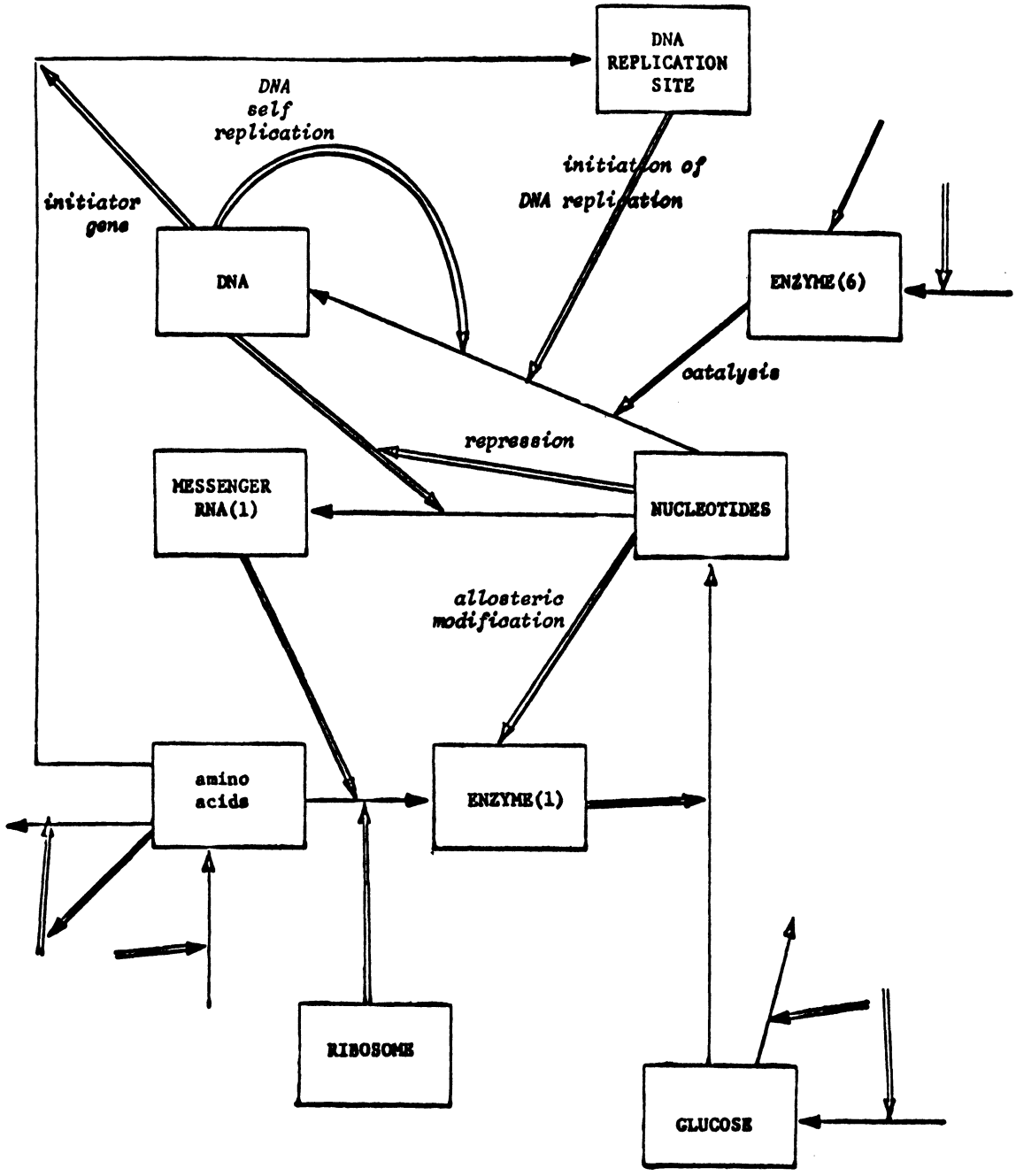


Figure 2.3 (continued)

Some of the relationships which are embodied in the equations of the transition function of the cell model are shown. These relationships, repression, allosteric modification, etc., are shown by a double arrow (⟹) to distinguish them from the flow of materials indicated by a single arrow (⟶).

such a pool would be lumped into a single entity, the pool itself.

Computer Program Summary: The simulation can be described by its state together with its transition function (Figures 2.1 - 2.6). The state of the cell is described 1) by the concentrations of thirty internal chemicals pools, 2) by the genetic apparatus, and 3) by the cell volume. The transition function used to obtain the state of the cell at the next time step of the simulation from a given time step consists of difference equations and Boolean expressions describing 1) enzyme catalyzed chemical reaction, 2) allosteric modification of enzymes, 3) repression of messenger RNA-production, 4) self-replication of DNA under genetic controls, and 5) permeability of the cell to the chemical pools represented in the simulation (Figure 3). Input to the simulation consists of the concentrations of chemicals in the liquid environment in which the cell is growing (Figures 2.7 - 2.9). Output from the simulation consists of state descriptions during successive time increments (Figure 2.5). Comparison between simulation output and experimental data from the real world (Chapter V) enables us to use the simulation in three ways: 1) we can judge the validity of the hypotheses used to write the simulation; 2) we can modify the hypotheses used to write the simulation in order to make the simulation more realistic; and 3) we can suggest critical real world experiments.

The equations in the transition function are general, thus allowing us to simulate cell behavior in many different environments, and in changes from one environment to another. This is especially significant since we wish to test the stability of the model under a variety of conditions.

Although the chemical constituents of the cell were lumped into the

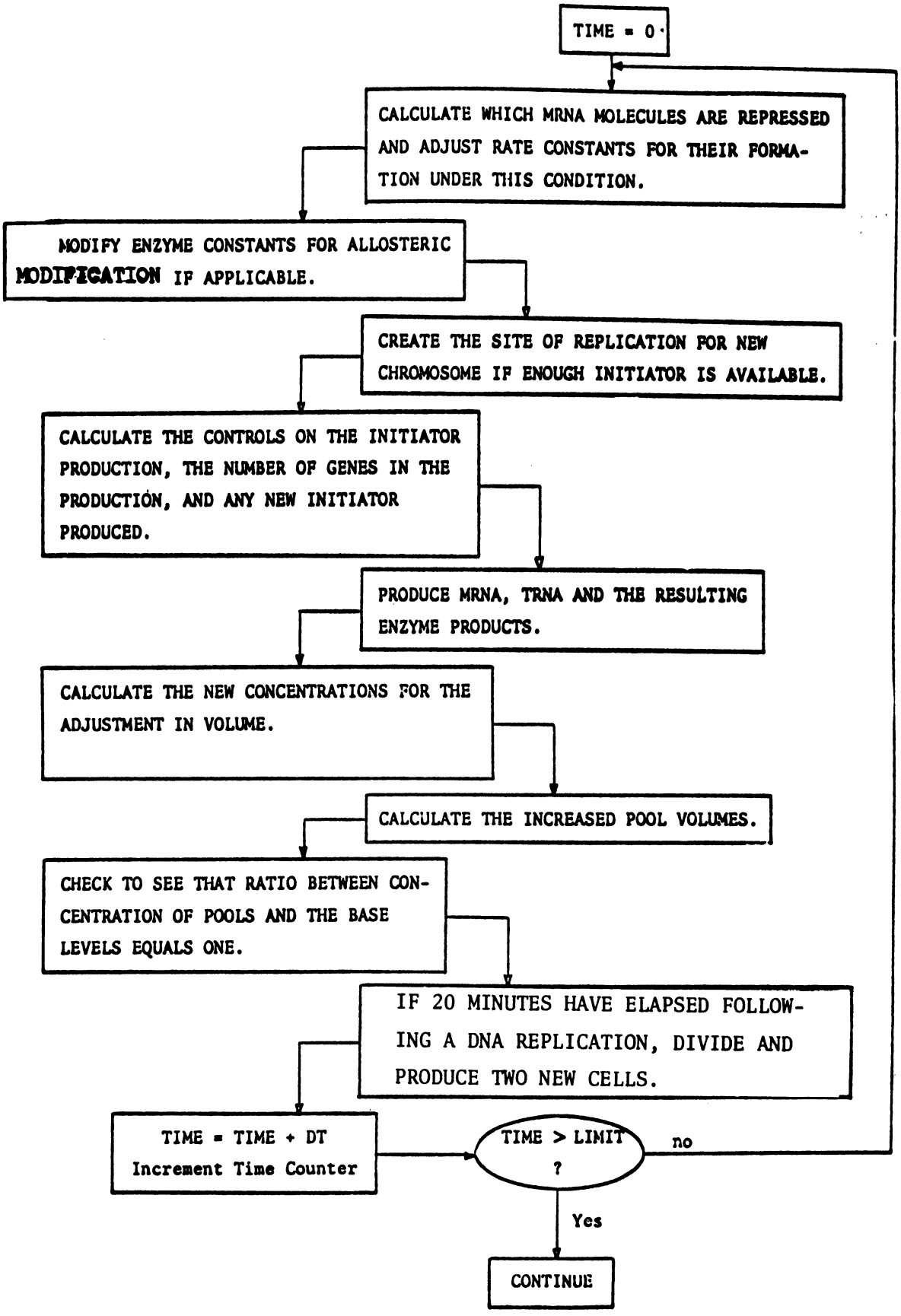


Figure 2.4 Summary of Program

```
   91  DDNA = K(6)*NUC*DT*EK(6)*ATP
       DDNA1 = K(6)*(T/DBLE)*.5*IN1*NUC*DT*EK(6)*ATP
       DDNA2 = K(6)*(T/DBLE)*.5*IN2*NUC*DT*EK(6)*ATP
       DDNA3 = K(6)*(T/DBLE)*.5*IN3*NUC*DT*EK(6)*ATP
C****      40 MINUTES TO REPLICATE .5 * DNA
       DO 100 I = 1,10
  100  DRNK(I) = (K8K(I)*NUC*DNA*EK(8)*ATP - KDRNK*RNK(I))*DT
C****
       DMRNA = DRNK(1) + DRNK(2) + DRNK(3) + DRNK(4) + DRNK(5) + DRNK(6)
     1 + DRNK(8) + DRNK(9) + DRNK(10) + DRNK(7)
C****
       DTRNA = K(10)*NUC*DT*EK(10)*ATP
       DRIB = K(9)*NUC*AA*DT*EK(9)*ATP
       DRNA = DMRNA + .25*DTRNA + .75*DRIB
       DWALL = K(4)*GLUC*DT*EK(4)*ATP
C****
       DO 101 I = 1,10
 101   DEK(I)= K(7)*AA*(RNK(I)/MRNA0)*DT*EK(7)*ATP
C****
       DPRTN = DEK(1) + DEK(2) + DEK(3) + DEK(4) + DEK(5) + DEK(6)
     1 + DEK(8) + DEK(9) + DEK(10) + DEK(7)
C****
       DNUC=           -(2.5E9/660.)*DDNA - (1.E6/660.)*DMRNA
     1 +K(1)*GLUC*DT*EK(1)*ATP - (2.5E4/660.)*DTRNA
     2  -(2.E6/660.)*DRIB - K(5)*NUC*DT*EK(5)*ATP
C****
       DAA=K(2)*GLUC*DT*EK(2)*ATP - 1.E6/102.*DRIB - (4.E4/102.)*DPRTN
       DATP = K(3)*GLUC*DT *ATP*EK(3) - DNAP*DDNA - MRNAP*DMRNA
     1  - TRNAP*DTRNA - RIBP*DRIB - PRTNP*DPRTN-WALLP*DWALL
     2 - (AAP*K(2)*GLUC*EK(2)*ATP + NUCP*K(1)*GLUC*EK(1)*ATP +2*K(5)*NUC
     3 *EK(5)*ATP )*DT
C****
       DADP = -DATP + K(5)*NUC*DT*EK(5)*ATP
       DVOL = K(14)*WALL*DT
C****      INCREASE IN VOLUME PER UNIT INCREASE IN CELL WALL
```

Figure 2.5

Differential Equations: quantity to the left of = is the change in amount of the substance; e.g., DDNA represents the change in the amount of DNA in one time increment DT. The differential equation underlying the first equation is

DDNA = K(6)*NUC*EK(6)*ATP*DT for a discrete time interval DT. As DT approaches 0, we get the underlying continuous differential equation

$$\lim_{DT \to 0} D(DNA)/DT = d(DNA)/dt = K(6)*NUC*EK(6)*ATP$$

```
      ADJST = VOL0/(VOL0 + DVOL)
C****
      VOLN= VOL*EXP (DT* K(14)*WALL/VOL)
C****
      DNA = (DNA + DDNA)*ADJST
      DNA1 = (DNA1 + DDNA1)*ADJST
      DNA2 = (DNA2 + DDNA2)*ADJST
      DNA3 = (DNA3 + DDNA3)*ADJST
      DNA1T = DNA1*VOL
      DNA2T = DNA2*VOL
      DNA3T = DNA3*VOL
      MULT = 0.
         IF (DNA3T - 1.000010) 151,151,150
  150  MULT = MULT + 2.
       GO TO 153
  151  CONTINUE
         IF (DNA3T - .1) 153,153,152
  152  MULT = MULT + 1.
  153  CONTINUE
      IF (DNA2T-1.000010) 155,155,154
  154  MULT = MULT + 2.
       GO TO 158
  155  CONTINUE
      IF (DNA2T-0.1) 157,157,156
  156  MULT = MULT + 1.
  157  CONTINUE
      IF (DNA1T-1.000010) 159,159,158
  158  MULT = MULT + 2.
         GO TO 161
  159  CONTINUE
      IF (DNA1T-0.1) 161,161,160
  160  MULT = MULT + 1.
  161  CONTINUE
      DIN = MULT*KIN*(AA/AA0)*DT
      IN = IN + DIN
      DO 102 I = 1,10
  102 RNK(I) = (RNK(I) + DRNK(I)) * ADJST
C****
      MRNA = (MRNA + DMRNA)*ADJST
      TRNA = (TRNA + DTRNA)*ADJST
      RIB = (RIB + DRIB)*ADJST
      RNA = (RNA + DRNA)*ADJST
      WALL = (WALL + DWALL)*ADJST
C****
      DO 103 I = 1,10
  103 EK(I) = (EK(I) + DEK(I)) * ADJST
C****
      PRTN = (PRTN + DPRTN)*ADJST
      NUC = (NUC + DNUC)*ADJST
      AA = (AA + DAA)*ADJST
      ATP = (ATP + DATP)*ADJST
      ADP = (ADP + DADP)*ADJST
```

Figure 2.6 New Concentrations. The amount of new material made during one time increment DT is added to the amount of old material present at the beginning of the time increment, and the new concentration is obtained by adjusting for the increase in cell volume during the time increment.

```
 1      T = 3000.
C****       GENERATION TIME 50 MINUTES
        NO = 1.
       DT = 1.
       CHRM0 = 2.
       FACTR = 2.
C****       2 CHROMOSOMES AT TIME 0, BOTH REPLICATE IMMEDIATELY
       DBLE = 50*60
C****       50 MINUTES FOR 1 CHROMOSOME TO REPLICATE
       DNA1Z = 1.
       DNA2Z = 1.
       DNA3Z = 0.
       DNA0 = DNA1Z + DNA2Z + DNA3Z
       IN1Z = 1.
       IN2Z = 1.
       IN3Z = 0.
       IN11Z = 0.
       IN21Z = 0.
       IN31Z = 0.
       INZ = 0.
       MRNA0 = 1.E3
       TRNA0 = 4.E5
       PRTN0 = 1.E6
       NUC0 = 1.2E7
       RIB0 = 1.5E4
       RNA0 = MRNA0 + .25*TRNA0 + .75*RIB0
C****       CONSIDER WEIGHT OF RNA 1E6
       AA0 = 3.E7
       GLUC0 = 40.E-03*1.E-15*2.25*(1/180.)*6.02E23
       WALL0 = 2.25E8
       ATP0 = 173.E-14*6.02E23*1.E-06
       ADP0 = 23.E-14*6.02E23*1.E-06
       LN2 = ALOG(2.)
           DO 2  II = 1,14,1
  2    PRDC0(II) = PRDC0(II)*LN2
       RNA0= RNA0*LN2
       VOL0 = 1.
       CAA = 0
       BROTH = 0
       COUNT = 0.
       IF(CNTRL.EQ.1)CRAZY=0
  3    DNA0= DNA1Z + DNA2Z + DNA3Z
       RNA = RNA0
           DO 4  II = 1,14
  4    PRDC K(II) = PRDC0(II)
       DNA1 = DNA1Z
       DNA2 = DNA2Z
       DNA3 = DNA3Z
       DNA1T = DNA1*VOL
       DNA2T = DNA2*VOL
       DNA3T = DNA3*VOL
       IN1 = IN1Z
       IN2 = IN2Z
       IN3 = IN3Z
       IN11 = IN11Z
       IN21 = IN21Z
       IN = INZ
       IN31 = IN31Z
```

Figure 2.7 Cell in Environment 1 (Minimal Medium).

```
      MRNA0 = 1.1*MRNA0
      TRNA0 = .9*TRNA0
      ATP0 = 1.1*ATP0
      PRTN0 = 1.1*PRTN0
      CAA = 1
C****
      VOL0 = 92.2/50.7
      DNA1Z = 1./VOL0
      DNA2Z = 1./VOL0
      DNA3Z = 1./VOL0
      T = (60./2.14)*60.
      CHRM0 = 3.
      FACTR = 3.
      DBLE = T
      IN3Z = 1./VOL0
      IN31Z = 1.
      RIB0 =((2.14 - 1.20)/(2.4 - 1.20)*( (250./135.)*1.5E4 -
     1 1.5E4) )*LN2
      RNA0 = MRNA0 + .25*TRNA0 + .75*RIB0
      AA0 = 1.5E8*LN2
C****  ADDITION FOR SOLVE
      NUC0 = 1.E7
      ADP0 = (38./62.)*ADP0
      GLUC0 = GLUC0*VOL0*3.
      WALL0 = WALL0*VOL0
```

Figure 2.8 Cell in Environment 2 (Minimal Medium + Amino Acids).

These are the quantities which change upon addition of amino acids to minimal medium.

```
      TRNA0 = 2.*TRNA0
      MRNA0 = MRNA0*1.1
      PRTN0 = PRTN0*1.1
      BROTH = 1
      CAA = 1
C****
      VOL0 = 117./50.7
      DNA1Z = 1./VOL0
      DNA2Z = 1./VOL0
      DNA3Z = 1./VOL0
      T = 25.*60.
      DBLE = T
      NUC0 = 6.E8*LN2
C****  ADDITION FOR SOLVE
      AA0 = 3.E8
      RIB0 = (250./135.)*1.5E4*LN2
      RNA0 = MRNA0 + .25*TRNA0 + .75*RIB0
      ATP0 = 1.1*ATP0
      ADP0 = (59./62.)*23.E-14*6.02E23*1.E-6*LN2 *1.1
      GLUC0 = 40.E-3*1.E-15*2.25*(1./180.)*6.02E23*LN2 *2.
      WALL0 = 2.25E8*VOL0*LN2
```

Figure 2.9 Cell in Environment 3 (Broth).

These are the quantities which are different in broth than in minimal medium.

aforementioned pools (Figure 2.1, 2.2), the pool of proteins was further divided into different enzyme groups: each enzyme group was associated with one group of chemical reactions responsible for converting one pool into another pool in the simulated cell. For example, the enzyme group EK2 catalyzed the production of amino acids from carbyhydrates. The messenger RNA pool was subdivided into a separate messenger RNA for each enzyme group. Variables used in the FORTRAN program for the simulation of a living cell are defined in the appendix to Chapter II.

Simulated environments: The environment simulated was a chemically defined liquid growth medium, at a temperature of 37 degrees Centigrade, with an abundance of oxygen. The changes in the environment simulated were changes in the chemical consituents of the media. Three different media were "fed" to the simulated cell: (1) a medium containing glucose, ammonium-salt, and minerals = mineral - glucose = environment (1) (Figure 2.7), (2) a medium containing glucose, minerals, and amino-acids = casa-mino-acids = environment (2) (Figure 2.8), (3) a medium containing glucose, minerals, amino-acids, and nucleosides = broth = environment (3) (Figure 2.9). With successive additions of amino-acids, and amino-acids and nucleosides the cell grew faster since it had fewer molecules to make on its own. This agreed with experimental data from literature on real cells (Maaløe and Kjeldgaard, 1966). The equations in the simulation were tested by comparing the growth of the simulated cell with that of a real cell (Figures 2.4 and 2.6). The simulated cell and the real cell both took 50 minutes to reproduce in mineral medium containing ammonium and glucose, 28 minutes in medium to which amino-acids had been added, and 25 minutes in medium containing both amino-acids and nucleosides.

DNA replication: The replication mechanism in our simulated cell may be summarized as follows (after Helmstetter, Cooper et al, 1968):

1) One initiator is produced per generation time per DNA end. DNA end refers to the origin of replication of DNA. As a consequence of the production of the initiator, the DNA starts to replicate.
2) DNA replication takes 40 minutes. During this 40 minutes, the DNA molecule is completely duplicated.
3) Twenty minutes after a DNA molecule finishes replicating, the cell physically divides to produce two new cells, each with half of the DNA present in the original cell.

Internal information from the simulated cell's cytoplasm is transmitted to the replication apparatus by the dependency of the rate of initiator-production on concentrations of internal cell-pools of amino-acid, ATP and the enzyme catalyzing the production of initiator from these substrates. The equation representing this interrelationship is:

$$d(IN)dt = k \cdot AA \cdot ENZYME$$

where $\frac{d(IN)}{dt}$ = initiator molecules produced per time step

k = the rate constant for the enzyme involved

AA = cellular amino acid information in the simulated cell.

The equation is based on the observable dependency of initiation of DNA replication on protein synthesis from amino acids in living cells.

The DNA replication mechanism efficiently receives information, both intracellular and intercellular, and is discussed more fully in the description of the cell space embedding of simulated living cells (Chapter VI).

Repression: The simulated cell employed *repression* to control the production of its enzymes (Figure 2.10). Repression operated at the DNA

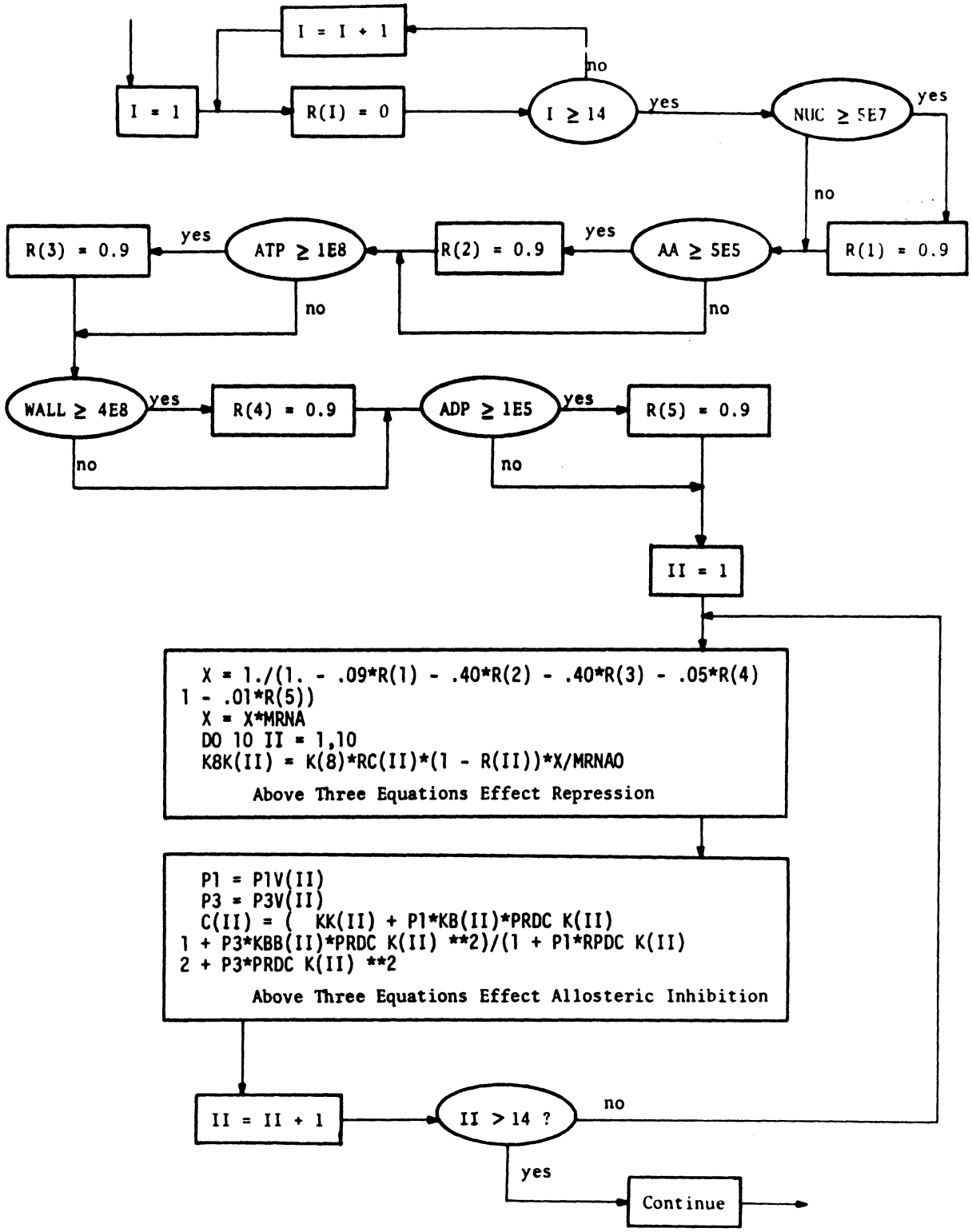


Figure 2.10 Repression and Allosteric Inhibition.

Repression is obtained by adjustment of KK8K(INTGR). Allosteric inhibition is obtained through adjustment of C(INTGR).

level. For example, EK2 was the enzyme pool needed for producing amino acids from carbohydrates. EK2 was produced under control of DNA by way of the RNA pools as long as the amino acid pool concentration was below a certain critical level. DNA directed the production of messenger RNA specific for the production of EK2. EK2 was produced by combining amino acids into a polypeptide chain. This peptide chain formation was done by messenger RNA attached to ribosomes. If the amino acid level rose above the critical level, production by DNA of messenger RNA responsible for EK2 production was soon curtailed; the messenger RNA already present rapidly decayed, and so almost no messenger RNA for EK2 was available after a short time. Since messenger RNA for EK2 is a prerequisite of EK2 production, the absence of this messenger RNA prevented the production of new EK2.

Of course, if the amino acid concentration fell too low, insufficient amino acids were available for synthesis into the EK2 enzyme. Production of all enzymes were blocked in the event of extreme scarcity of amino acids.

Allosteric Modification of Enzymes: The simulated cell employed feedback inhibition to control the activity of the enzymes already present (Figure 2.9), (Chapter III). For example, EK2, the enzyme for production of amino acids from carbohydrates, appeared in three different forms: pure enzyme, enzyme with one molecule of amino acid attached to it, and enzyme with two molecules of amino acid attached to it. These three forms of EK2 had different catalytic abilities. The relative amount of EK2 in each form determined the activity of the EK2 present in the cell in terms of its efficiency in converting carbohydrate into amino acids. The percentage of EK2 in each of the three forms was determined

by the number of amino acid molecules per cell volume unit (one cell volume unit was taken as the volume of a cell growing rapidly in mineral glucose medium with ammonium salt). The higher the amino acid concentration in the simulated cell rose, the lower the percentage of highly active EK2 fell. This slowed down the rate of production of amino acids from carbohydrates when amino acid concentration was too high. The cell doesn't make a lot of amino acids when a surplus already exists.

Overall Program: For each time increment (DT) in the running program, new concentrations of pools were calculated from the transition function, taking into account DNA replication,repression of enzyme production, and allosteric modification of enzyme catalysis. Uses of the simulated cell will be illustrated in later chapters. E.g., allosteric modification will be shown to be necessary for rapid shifts up from poor to rich media (Chapter V).

CHAPTER III

ALLOSTERIC MODIFICATION: HOMOMORPHISM; SIMULATION

We will describe our model of allosteric modification of enzymes in two sections: 1) We will develop the model as a valid example of a homomorphic mapping which preserves interesting functional relationships of the system modelled. 2) We will describe the simulation of the model, repeating much of the first section on the formulation of the model. The simulation of allosteric modification will be described in FORTRAN notation for clarity, and also to emphasize that the model and the simulation are separate entities.

As a preface, we will give some examples of system simplification. Then we will list some elementary definitions from automata theory. We will use the concepts defined in order to formulate and validate our model of allosteric modification. The use of homomorphism to formulate and validate models will be extended to the more general case in CHAPTER IV (Zeigler and Weinberg, 1970). Zeigler (1970) is developing a general theory of model-building. His theory is related to the use of homomorphism in valid computer simulations. For our computer simulation, allosteric modification illustrates that a homomorphic simplification of a complex system can capture exactly those functional characteristics which we wish to study in our system.

We will now give two particular examples of ways in which we can simplify models. First, we may simplify the way in which we describe the state of the model. Second, we may simplify the transition function

of the model. (We may use either kind of simplification. We may even choose to use both kinds of simplification simultaneously. Indeed, we may find that using both kinds of simplification is the most natural tactic to employ). 1) The *state* of a model cell may be described by the concentrations of all primary molecular constituents in the cell. This description would give the concentrations of all of the compounds presented in Figure 2.1. In this state description, there are many different variables. For example, SERINE and ALANINE are each separate variables. To simplify the state description, we may group variables. In grouping, we decrease the number of variables in our state description. Figure 2.2 shows such a grouping. It is a vastly simplified model of a cell. In it, SERINE and ALANINE are no longer separate variables. Instead, they are both included in the AMINO pool. We have lost some information in lumping SERINE and ALANINE into one pool. We can no longer follow the individual values of SERINE and ALANINE. Let us assume for our example that AMINO is equal to the sum of SERINE and ALANINE. In our lumped model we only can follow the behavior of the sum of the SERINE and ALANINE concentrations. Thus, we have lost some information in our state simplification. 2) The *transition function* of a model can also be simplified. We will illustrate the simplification of a transition function by continuing with our example. Assume that the unsimplified state space contains three variables; 1) SERINE, 2) ALANINE and 3) AMINO. AMINO is the sum of SERINE and ALANINE. The first, unsimplified transition-function will update the values of SERINE, ALANINE, and AMINO. It will operate on the unsimplified state space. We will write the first, unsimplified transition

function as the following three equations.

1) SERINE(t+1) = SERINE(t) + K(1)

2) ALANINE(t+1) = ALANINE(t) + K(2)

3) AMINO(t+1) = AMINO(t) + K(1) + K(2)

SERINE(t+1) indicates the value of the concentration of SERINE at time = t+1.

The second, simplified transition-function will update the value of AMINO. The simplified transition function may operate on either the lumped-model state-space or on the original,unsimplified state-space. The second transition-function in equation form is AMINO(t+1) = AMINO(t) + K(3). Let us consider the information we have lost when we use our simplified transition-function. Our second, simplified transition-function only gives us the value of AMINO over time. We have lost information as to the individual values of SERINE and ALANINE when we use the simplified transition-function.

We will now introduce the notion of homomorphism into our little example. To see whether our lumped state state-space and simplified transition-function together constitute a homomorphic image of our original state-space and transition-function, we compare the results of two procedures. 1) We lump our state-space, apply the simplified transition-function, and obtain a new state value. 2) We apply the unsimplified transition-function to the unsimplified state-space, and then lump the state-space. If both of these procedures always give the same final state-value, then our lumped state-space and simplified transition-function represent a homomorphic image or our original state-space and unsimplified transition-function. Illustrating with

our example, if we know the values of AMINO, SERINE, and ALANINE at time =0, we have two distinct ways of obtaining the value of AMINO 10 times steps later, at t = 10. 1) We may apply the unsimplified transition-function 10 times to obtain the values of SERINE (10) from SERINE (0), and to obtain the value of ALANINE (10) from ALANINE (0). We may then lump our state space by adding SERINE (10) and ALANINE (10) to obtain AMINO (10). We used the unsimplified transition-function first, and then lumped the state-space, and finally obtained the value of AMINO(10) by adding SERINE (10) and ALANINE (10). Alternately, we could have added SERINE (0) and ALANINE (0) to obtain AMINO (0). We could then have applied the simplified transition-function 10 times to obtain AMINO (10). In our second procedure, we lumped our state-space first, and then applied our simplified transition-function. If the value of AMINO (10) is the same by both procedures, we may have a homomorphism. If the two procedures give different values for AMINO (10), we know that we do not have a homomorphism.

Homomorphism of Systems: Basic Definitions:

In its most basic form a *system* is defined as a set of states S, together with a *transition function* $\tau:S\rightarrow S$. τ describes the behavior of the system over time by indicating which next state is to follow the present state. Thus if the state at time t is s(t) then the state at time t+1, $s(t+1) = \tau(s(t))$ where $t = 0,1,2,\ldots$.

The *state space* S is usually described as a cartesian product of component state sets, i.e., $S = \underset{\alpha\in D}{x} S_\alpha$ where D is a finite set of *co-ordinates* (or entities) and S_α is the state set (or attribute set) of co-ordinate α. In Figure 4.1 we list a number of possible state spaces and indi-

cate the form a transition function might take in each. In Figures 2.2 and 2.3 we specify in more detail the state space and transition function of the present model. Thus, the co-ordinates of the model include the chemical pools of Figure 2.2 (Amino Acids, Protein, Glucose, etc.), the enzymes (Ek_1, Ek_2,..., Ek_{10}), and the different messenger RNAs (RNk_1, ...RNk_{10}). We must also keep track of volume of the cell, the number of cells, and information to be used by the replication subroutine. Accordingly, these are also represented in our state vector.

The transition function for our model consists of difference equations and Boolean expressions describing 1) enzyme catalyzed chemical reactions, 2) allosteric modification of enzymes, 3) repression of messenger RNA-production, 4) self-replication of DNA under genetic controls, and 5) permeability of the cell to the chemical pools represented in the simulation (Figure 2.2). The transition function captures relationships not apparent in the state coordinate representation (Figure 2.3).

Noting that $S = \underset{\alpha\varepsilon D}{x} S_\alpha$, τ is equivalent to a set of maps $\{\tau_\beta \mid \beta\varepsilon D\}$ where each τ_β maps S to S_β. The co-ordinate function τ_β tells how the next state of co-ordinate β depends on the present state of the system. (Clearly, if we tell how each co-ordinate is to change state we will be specifying how the system changes state, and conversely.) Indeed, τ_β may not depend on all the co-ordinates in D: --let I_β be the subset of D on which τ_β depends, then τ_β maps $\underset{\alpha\varepsilon I_\beta}{x} S_\alpha$ into S_β.

Based on the sets I_β we can draw a *digraph* (directed graph) which represents the coordinate interdependence in the system. The digraph, which we also denote by D, has as points the set D, and we draw a line from α to β just in case $\alpha\varepsilon I_\beta$.

As an example, consider the schematic view of some of the main meta-

bolic pathways in *E. Coli* of Figure 1. Under conditions where the enzyme concentration remains fixed we can obtain the digraph, D^E, of the system with concentration state space, S^E, directly, as follows: $--D^E$ is the set of chemicals shown in Figure 2.1. For each of the lines shown we must add a line in the opposite direction; further, when any two points are adjacent to a third they must be joined by lines in both directions. (These rules are inferred from the chemical kinetic equations.) For example, the input set I_{GLU6}, of Glucose-1-P, consists of Glucose, Glucose-1-P and Glycogen. This means that τ_{GLU6}, which specifies the next concentration of this chemical, depends only on the concentrations of the chemicals in I_{GLU6}.

We mention that the interrelation between properties of the digraph (structure) and properties of the transition function (behavior) have been fruitfully applied to biochemical systems by Kauffman (1969) and others.

Let S, S' be state spaces and τ,τ' their associated transition functions. We seek conditions under which the system (S', τ') can be reasonably said to be a model or simplification of (S, τ).

To do this, we say that a *trajectory* through S beginning at s in S is any sequence

$<s, \tau(s), \tau^2(s),\ldots,\tau^n(s)>$, where $\tau^i(s) = \tau^{i-1}(\tau(s))$, $\tau^1(s) = \tau(s)$.

The set of such trajectories is the *behavior* of the system (S,τ). Let $h:S\to S'$ be a map from S onto S'; we want to preserve the behavior of (S,τ) in the sense that for any trajectory $<s, \tau(s), \tau^2(s),\ldots, \tau^n(s)>$ in the behavior of (S, τ), $<h(s), h(\tau(s)), h(\tau^2(s)),\ldots,h(\tau^n(s))>$ is a trajectory in the behavior of (S', τ').

This is equivalent to the condition that for every $s \in S$, $h(\tau(s)) =$

$\tau'(h(s))$. In practice this condition may be required to hold only approximately, and then only for that part of the state space which may be explored experimentally. If h satisfies this condition it is said to be a *homomorphism* and (S', τ') is a homorphic image of (S, τ). When h is a one-one homomorphism, it is an isomorphism and (S', τ') is isomorphic to (S, τ). We shall take as the formal counterpart of the statement that "(S', τ') is a model of (S, τ)" the statement "(S', τ') is a homomorphic image of (S, τ)" our justification being that the notion of "model" entails the concept of behavior preservation we have given.

Allosteric Modification; a Homomorphic Model; Consider a subsystem consisting of substrate A, product B, and allosteric enzyme E. Enzyme E can exist in three forms; E (pure enzyme), EB (enzyme with one molecule of B attached), and BEB (enzyme with two molecules of B attached). The total catalytic activity per unit of enzyme E will be determined by the percentage of enzyme in each of the three forms E, EB, and BEB.

The chemical kinetics of the rates of conversion of the three enzyme forms E, EB, and BEB are given by:

$$E \underset{p_2}{\overset{p_1}{\rightleftarrows}} EB \underset{p_4}{\overset{p_3}{\rightleftarrows}} BEB$$

$$A + E \xrightarrow{k} B + E$$

$$A + EB \xrightarrow{k_b} B + EB$$

$$A + BEB \xrightarrow{k_{BB}} B + BEB$$

where the p's and k's are suitable rate constants. Call this system S. Then S has as co-ordinates D = {A, B, E, EB, BEB} with state space $S = (A) \times (B) \times (E) \times (EB) \times (BEB)$ where, for example, $\overline{A}$ denotes the concentration of A, and (A) is the range over which $\overline{A}$ varies. The transition

function τ is specified by the co-ordinate function $\{\tau_A, \tau_B, \tau_E, \tau_{BEB}\}$ namely,

$$\tau_A(\overline{A},\ \overline{E},\ \overline{EB},\ \overline{BEB}) = \overline{A} - k\,\overline{E}\,\overline{A} - k_B\overline{EB}\cdot\overline{A} + k_{BB}\overline{BEB}\cdot\overline{A}$$

$$\tau_B(\overline{A},\ \overline{B},\ \overline{E},\ \overline{EB},\ \overline{BEB}) = \overline{B} + k\,\overline{E}\,\overline{A} + k_B\overline{EB}\cdot\overline{A} + k_{BB}\overline{BEB}\cdot\overline{A}$$

$$\tau_E(\overline{B},\ \overline{E},\ \overline{EB}) = \overline{E} - p_1\overline{E\cdot B} + p_2\overline{EB}$$

$$\tau_{EB}(\overline{B},\ \overline{E},\ \overline{EB},\ \overline{BEB}) = \overline{EB} + p_1\overline{E}\cdot\overline{B} - p_2\overline{EB} - p_3\overline{EB}\cdot\overline{B} + p_4\overline{BEB}$$

$$\tau_{EBB}(\overline{B},\ \overline{EB},\ \overline{BEB}) = \overline{BEB} + p_3\overline{EB}\cdot\overline{B} - p_4\overline{BEB}. \qquad \ldots 1)$$

(These are discrete versions of the usual kinetic differential equations.)

We wish to simplify S, i.e., to replace it with a homomorphic image system S'. Let S' have as co-ordinates $\{S, B, E_t\}$ where E_t is to represent the aggregate of the enzyme in its three forms; i.e., $\overline{E}_t = \overline{E} + \overline{EB} + \overline{BEB}$. $\overline{E}_t$ is the total amount of enzyme, and is obtained by adding together the amount of enzyme in each of the three different forms 1) $\overline{E}$, 2) $\overline{EB}$, and 3) $\overline{BEB}$. $\overline{E}_t$ will be used in the following development. Then S' has as state space $S' = (A) \times (B) \times (E_t)$ and we define a map $h:S\to S'$ given by $h(\overline{A},\ \overline{B},\ \overline{E},\ \overline{EB},\ \overline{BEB}) = (\overline{A},\ \overline{B},\ \overline{E} + \overline{EB} + \overline{BEB}) = (\overline{A},\ \overline{B},\ \overline{E}_t)$.

From the above we know that the state of system S is defined by the values of 5 variables: 1) $\overline{A}$, 2) $\overline{B}$, 3) $\overline{E}$, 4) $\overline{EB}$, and 5) $\overline{BEB}$. The state of system S' is defined by the values of only 3 variables: 1) $\overline{A}$, 2) $\overline{B}$, and 3) $\overline{E}_t$.

The problem now is to define a transition function τ' for S' which will make h a homomorphism. To do this we shall consider the kinetics of S' to be given by:

$$A + E_t \xrightarrow{K} B + E_t$$

where K(B) will be a rate "constant" depending on B thus reflecting the feedback effect of allosteric inhibition.

The transition function of system S' must give the values of A, B, and E_t at time t + 1 from the values of A, B, and E_t at time t. τ'_A is that part of the transition function used to obtain the value of A at time t + 1 from the values of A and E_t at time t. The amount of A lost during one time step is $K(\overline{B})\cdot\overline{E}_t\cdot\overline{A}$. Thus the value of A at time t + 1 will be the value of A at time t minus the amount of A lost during the time step. That is $A(t + 1) = A(t)-K(B)\cdot\overline{E}_t\cdot A(t)$. Letting a capitol letter with a bar over it designate the value of a variable at time = t, we may express the function which updates the value of A as follows:

$$\tau'(\overline{A}, \overline{E}_t) = \overline{A} - K(\overline{B})\cdot\overline{E}_t\cdot\overline{A}$$

The transition equations of system S' then are:

$$\tau'_A(\overline{A}, \overline{E}_t) = \overline{A} - K(\overline{B})\cdot\overline{E}_t\cdot\overline{A}$$

$$\tau'_B(\overline{A}, \overline{B}, \overline{E}_t) = \overline{B} + K(\overline{B})\cdot\overline{E}_t\cdot\overline{A}$$

$$\tau'_{E_t}(\overline{E}_t) = \overline{E}_t \qquad \ldots 2)$$

The condition under which h is a homomorphism is then that

$$h(\tau(\overline{A}, \overline{B}, \overline{E}, \overline{EB}, \overline{BEB})) = \tau'(h(\overline{A}, \overline{B}, \overline{E}, \overline{EB}, \overline{BEB})).$$

In words, using the original transition-function and then lumping is equivalent to lumping and then using the simplified transition-function. By working out the A co-ordinate we obtain the equation:

$$\overline{A}(1 - k\overline{E} - k_B\overline{EB} - k_{BB}\overline{BEB}) = \overline{A}(1 - K(\overline{E} + \overline{EB} + \overline{BEB})).$$

A similar equation results in the B position. An equation involving $\overline{E}$, $\overline{EB}$, $\overline{BEB}$ appears in the E_t position which is trivially satisfied (it is a result of the fact that in system 1 the total enzyme concentration $\overline{E} + \overline{EB} + \overline{BEB}$ is constant). Thus h will be a homomorphism just in case there is a function, which we denote as K(B), such that

$$k\overline{E} + k_B\overline{EB} + k_{BB}\overline{BEB} = K(\overline{B})\ (\overline{E} + \overline{EB} + \overline{BEB}) \qquad \ldots 3)$$

for all values <u>of interest</u> of $\overline{E}$, $\overline{EB}$, $\overline{BEB}$.

If equation 3 holds, we may use the total concentration of enzyme ($\overline{E} + \overline{EB} + \overline{BEB}$) to calculate the total catalytic activity of the three forms of the enzyme 1) $\overline{E}$, 2) $\overline{EB}$, and 3) $\overline{BEB}$. We no longer have to calculate the catalytic activity of each form of the enzyme separately. We have made a homomorphic mapping from three co-ordinates 1) E, 2) EB and 3) BEB onto one c-ordinate (E_t). In order to use the one co-ordinate in a new transition-function, we have also developed a function of $\overline{B}$, denoted as $K(\overline{B})$. The function $K(\overline{B})$ is used in the new transition function in the same manner as the three rate constant 1) k, 2) k_B, and 3) k_{BB} were used in the old transition function. This is what was implied by our equation 3:

$$k\overline{E} + k_b \cdot \overline{EB} + k_{BB} \cdot \overline{BEB} = K(\overline{B}) \cdot (\overline{E} + \overline{EB} + \overline{BEB}).$$

We shall shortly examine the factors determining the extent to which equation 3 can be satisfied.

Now under steady state conditions we assume that the concentrations of the enzyme forms E, EB, BEB of system S remain constant. This results in the following equilibrium equations:

$$\begin{aligned} - p_1 \hat{E} \cdot \hat{B} + p_2 \cdot \hat{EB} &= 0 \\ p_1 \hat{E} \cdot \hat{B} - p_2 \cdot \hat{EB} - p_3 \cdot \hat{EB} \cdot \hat{B} + p_4 \cdot \hat{BEB} &= 0 \\ p_3 \cdot \hat{EB} \cdot \hat{B} - p_4 \cdot \hat{BEB} &= 0 \end{aligned} \qquad \ldots 4)$$

where, for example, $\hat{E}$ is the constant value of concentration $\overline{E}$ in a steady state condition.

Manipulating the equations for equilibrium conditions, one can obtain the following expressions for $\hat{E}$, $\hat{EB}$, and $\hat{BEB}$:

$$\tilde{E} = \tilde{E}_{total} \cdot \frac{1}{(1 + P_1 \cdot B + P_3 \cdot \tilde{B}^2)}$$

$$\tilde{E}B = P_1 \cdot \tilde{B} \cdot \tilde{E}$$

$$B\tilde{E}B = P_3 \cdot \tilde{B}^2 \cdot E$$

where

$$P_1 = p1/p2$$

$$P_3 = (p3/p4) \cdot P1 \qquad \ldots 5)$$

(To obtain these expressions, note that:

$$\tilde{E}B = (p1/p2) \cdot \tilde{B} \cdot \tilde{E} = P_1 \cdot \tilde{B} \cdot \tilde{E}$$

$$B\tilde{E}B = P1 \cdot (p3/p4) \cdot \tilde{B}^2 \cdot \tilde{E} = P_3 \cdot \tilde{B}^2 \cdot \tilde{E}$$

$$\tilde{E} = \tilde{E}_{total} - \tilde{E}B - B\tilde{E}B = E_{total} - P1 \cdot \tilde{B} \cdot \tilde{E} - P3 \cdot \tilde{B}^2 \cdot \tilde{E} .$$

Dividing out E from both sides, one obtains

$$1 = P1 \cdot \tilde{B} - P3 \cdot \tilde{B}^2 + \tilde{E}_{total} / \tilde{E}$$

Manipulating, and substituting $P_1 = p1/p2$ and $P_3 = p3/p4$ one obtains

$$\tilde{E} = \tilde{E}_{total} \cdot \frac{1}{1 + P_1 \cdot \tilde{B} + P_3 \cdot \tilde{B}^2}$$

Substituting these expressions in the homomorphism equation 3, cancelling and collecting terms yields:

$$k + (P_1 \cdot k_B)\tilde{B} + (P_3 \cdot k_{BB})\tilde{B}^2 = K(\tilde{B}) \ (1 + P_1 \cdot \tilde{B} + P_3 \cdot \tilde{B}^2). \qquad \ldots 6)$$

Calculating Catalytic Activity of Model Enzymes: The catalytic activity of the different allosteric forms of an enzyme are summarized as one number, K(B). K(B) thus summarizes the catalysis of a reaction by the system of enzyme forms E, EB, and BEB. We shall now show how a value $K(\tilde{B})$, can be estimated from experimental data for a given steady-state concentration-value $\tilde{B}$. Steady-state concentrations of the pools are assumed to be known for given nutritional environments.

Rate constants for flow of materials between pools are calculated for each environment from the steady state concentrations of the pools in that environment, the initial volume of the cell at the beginning of a generation, and the time necessary for all quantities in the cell to double in that environment, i.e., the time for one cell generation. Exponential increase in cell mass and in all pool amounts, except for a linear increase in DNA and genetic apparatus, is assumed. The calculations will be illustrated for protein production from amino acids given by:

$$\text{PROTEIN} \xleftarrow{K} \text{AMINO ACID}$$

The chemical kinetic equation is:

$$\frac{d(\text{PROTEIN}(t))}{dt} = K \cdot [\text{AMINO}(t)] \cdot \text{VOLUME}\ (t)$$

where

PROTEIN(t) = number of protein molecules in the cell at time t.

[AMINO(t)] = the number of amino acid molecules per cell.

VOLUME(t) = volume of cell at time t, (set at 1 for cell at beginning of generation).

Integrating over one cell cycle, during which volume doubles

$$\int_{\text{PROTEIN}(0)}^{\text{PROTEIN}(1)} d(\text{PROTEIN}(t)) = \int_{t=0}^{t=1} K \cdot [\text{AMINO}(T)] \cdot \text{VOLUME}(T)\,dt$$

where

PROTEIN(0) = total number of protein molecules in the cell at time = 0, the beginning of a cell cycle.

PROTEIN(1) = total number of protein molecules in the cell at time = 1, the end of a cell cycle.

In our model the increase in volume in a steady-state growth-condition in one environment is just sufficient to keep pool concentrations constant. Increase in volume occurs exponentially, at a rate permitting volume to double over one cell-generation. The amino-acid concentration in the cell, therefore, remains constant in a given environment. K is also constant in this environment, so that K and [AMINO(t)] can be placed in front of the integration sign to give

$$\int_{\text{PROTEIN}(0)}^{\text{PROTEIN}(1)} d(\text{PROTEIN}(t)) = K \cdot [\text{AMINO}(t)] \cdot \int_{t=0}^{t=1} \text{VOLUME}(t)\,dt$$

Since volume is assumed to increase exponentially during steady state growth condition, and VOLUME(0) = 1, one can set up the equation

$$\text{VOLUME}(t) = \text{VOLUME}(0) \cdot e^{at} = 1 \cdot e^{at} = e^{at}$$

Since the volume of a cell doubles in one generation, VOLUME(1) = 2. Thus a = ln (VOLUME(1) = ln(2). Going back to the integral relating increase in number of protein molecules to volume and amino-acid concentration, and substituting $e^{\ln(2) \cdot t}$ for VOLUME(t), one obtains

$$\int_{\text{PROTEIN}(0)}^{\text{PROTEIN}(1)} d(\text{PROTEIN}(t)) = K \cdot [\text{AMINO}(t)] \cdot \int_{t=0}^{t=1} e^{\ln(2) \cdot t} dt$$

Integrating, one obtains PROTEIN(1) - PROTEIN(0) = $\frac{K \cdot [\text{AMINO}(t)] \cdot}{\ln(2)}$

All amounts double in one cell cycle, therefore PROTEIN(1) = 2·PROTEIN(0). The concentration of amino acid is equal to total amount of the amino acid pool per cell volume, i.e., [AMINO(t)] = AMINO(t)/VOLUME(t). At t=0, [AMINO(0)] = AMINO(0)/VOLUME(0). Since [AMINO(t)] remains constant in an environment for steady state growth, [AMINO(t)] = AMINO(0)/VOLUME(0). With VOLUME(0) = 1, [AMINO(t)] = AMINO(0) and one obtains PROTEIN(0) = $\frac{K \cdot \text{AMINO}(0)}{\text{in}(2)}$. Solving for K, one obtains K = (PROTEIN(0)/AMINO(0))·ln(2).

The amounts of the protein and amino acid pools at the beginning of a cell cycle, PROTEIN(0) and AMINO(0), can be calculated from the data available in the literature on the average amounts of these pools. We will show that since the pool size increases exponentially during growth, the pool size at the beginning of a generation equals the average pool size times the natural logarithm of 2. The equation for calculating the average volume over one cell cycle is

$$\int_{t=0}^{t=1} \text{VOLUME}(t)\cdot\text{PROBABILITY}(t)dt = \text{average volume over a cell cycle}$$

where

t = time in life cycle,

t = 0 at beginning of cell cycle,

t = 1 at end of cell cycle,

VOLUME(t) = volume of cell at time t,

PROBABILITY(t) = probability density for the time in the life cycle of a cell being considered.

We shall set PROBABILITY(t) = 1 since bacterial growth is ergodic, and unsynchronized cultures were used for experimental data from the literature, giving the uniform density. In other words, we are assuming that our data comes from cells at all different parts of the bacterial life cycle. Thus

$$\int_{t=0}^{t=1} \text{VOLUME}(0)\cdot e^{\ln(2)\cdot t}\cdot 1dt = \text{average volume over a cell cycle.}$$

Carrying out the integration, average volume = VOLUME(0)/ln(2), and VOLUME(0) = average volume ·ln(2).

Similar equations indicate that the average pool size for a cell must be multiplied by ln(2) to give the pool size at the beginning of a cell cycle. The equations for the amino acid pool are developed as an illustration.

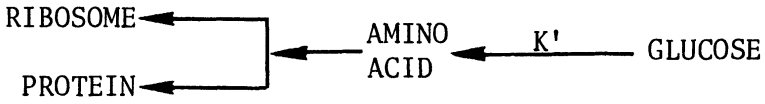
$$\int_{t=0}^{t=1} \text{AMINO}(t)\cdot\text{PROBABILITY}(t)dt = \text{average amino acid pool size}$$

In a steady state environment AMINO(t) = [AMINO(0)]·VOLUME(t), VOLUME(t) = $e^{\ln(2)t}$, and PROBABILITY(t) = 1. Substituting for AMINO(t) and PROBABILITY(t), one obtains

$$\int_{t=0}^{t=1} \text{AMINO}(0)\cdot e^{\ln(2)\cdot t}dt = \text{average amino-acid pool size.}$$

Integrating, one obtains average amino-acid pool-size = AMINO(0)·ln(2).

From the assumed metabilic-pathway:



one can set up the equation

$$\frac{d(\text{AMINO}(t))}{dt} = K'\cdot[\text{GLUCOSE}] - k_{AR}\frac{d(\text{RIBOSOME}(t))}{dt} - k_{AP}\frac{d(\text{PROTEIN}(t)}{dt},$$

where k_{AR} is the number of amino acid moleucles used to form one ribosome molecule (k_{AP} is similarly defined).

One can solve for K' using the same procedure as before thus obtaining

$$\text{AMINO}(0) = K'\cdot\text{GLUCOSE}(0)/\ln 2 - k_{AR}\cdot\text{RIBOSOME}(0) - k_{AP}\cdot\text{PROTEIN}(0).$$

In this way rate constants can be calculated for different chemical pools in different environments.

Assume that for system (S,τ) values of $K(\tilde{B})$ and $\tilde{B}$ have been computed for two different environments and that in similar fashion the constants

k, k_B, k_{BB} have also been estimated from the literature. This results in two simultaneous linear equations which can be solved for P_1 and P_3. Let the values so obtained be $\bar{P}_1$ and $\bar{P}_3$ and consider the form of $K(\bar{B})$ derived from equation 6:

$$K(\bar{B}) = \frac{k + (\bar{P}_1 \cdot k_B)\bar{B} + (\bar{P}_3 \cdot k_{BB})\bar{B}^2}{1 + \bar{P}_1 \cdot \bar{B} + \bar{P}_3 \cdot \bar{B}^2} \qquad \ldots 7)$$

Validity of Model; Using this function for $K(\bar{B})$ we see that equation 3 will hold and h will be a homomorphism to the extent that:

1) The steady state conditions assumed in equation 4 are satisfied, and

2) The values obtained for P_1 and P_3 in two environments hold for other environments of interest.

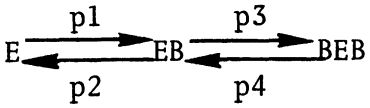
The state conditions will hold to the extent that, as is usually assumed, the adjustment in enzyme form concentrations is rapid with respect to the formation of product from substrate. Also, if the values obtained for P_1 and P_3 do not hold for other environments one can complicate system S by assuming more forms for the enzyme-product complex, thus introducing higher order terms in equations 6 and 7.

We see that in order to fully discuss the notion of homomorphism in practical applications we need to determine how close an arbitrary map comes to being a homomorphism. Ulam (1966) has suggested the introduction of a numerical error functional which in this context takes the form $\varepsilon(\tau'(h(s)), h(\tau(s))$. Then the extent which h is a homomorphism is measured by the least upper bound on ε over the region of the state space of interest.

Let us put this example in a more general perspective. We have a

system S in which allosteric inhibition is modelled by the presence and effects of a number of different enzyme-product complexes. This system is not used in our living cell model but is replaced by a simpler system S'. In S', allosteric inhibition is effected by a feedback path from the product to the rate "constant" governing its rate of formation. The reader may wish to check that as far as the substrate and product are concerned, the behavior of the two systems is the same (given that equation 3 is valid). *In fact, this kind of behavior preservation is just that entailed by requiring that S' be a homomorphic image of S.* We see here how the homomorphism concept elucidates a technique actually used to implement and simplify the allosteric inhibition component of our model.

Computer Implementation of Allosteric Modification: Discrete difference equations simulate differential equations in the simulated cell's transition function. Fortran arrays will index the products and substrates in the simulated cell. Therefore, allosteric modification will be described for the simulation in terms of the differential equations being simulated, and the indexed arrays used to represent substrates and products. The system of enzymes E, EB, and BEB assumed for simulating allosteric modification is exactly the same as those underlying the formal model:



Three different forms of enzyme:

E = concentration of pure enzyme,

EB = concentration of enzyme with one molecule of product B attached,

BEB = concentration of enzyme with two molecules of product B attached.

p1, p2, p3, and p4 are rate constants for the rate of formation of various

forms of the enzyme as a function of the concentrations of other forms of the enzyme, and concentrations of product B.

$$\frac{d(E)}{dt} = p1 \cdot E \cdot B + p2 \cdot EB$$

$$\frac{d(EB)}{dt} = p1 \cdot E \cdot B - p2 \cdot EB - p3 \cdot EB \cdot B + p4 \cdot BEB$$

$$\frac{d(BEB)}{dt} = p3 \cdot EB \cdot B - p4 \cdot BEB$$

At equilibrium, $\frac{d(E)}{dt} = \frac{d(EB)}{dt} = \frac{d(BEB)}{dt} = 0$. The total concentration of enzyme is equal to the sum of the concentrations of its three alternate forms:

$$E_{total} = E + EB + BEB$$

From the model, there are three rate constants for different forms of an enzyme catalyzing a reaction.

KK = rate constant of pure enzyme E.

KB = rate constant of enzyme EB which has one molecule of B attached to it.

KBB = rate constant of enzyme EBB which has two molecules of B attached to it.

These rate constant lead to the following form for catalysis by different forms of the enzyme, and the differential equation for the chemical reaction catalyzed by the enzyme.

$$A + E \xrightarrow{K} B + E$$

$$A + EB \xrightarrow{KB} B + EB$$

$$A + BEB \xrightarrow{KBB} B + BEB$$

$$\frac{d(B)}{dt} = KK \cdot E \cdot A + KB \cdot EB \cdot A + KBB \cdot BEB \cdot A =$$

$= (KK \cdot E + KB \cdot EB + KBB \cdot BEB) \cdot A$

= change in amount of B with respect to time.

(See Figure 3.1.)

In terms of the rate constants calculated for each environment (Figure 3.2) K(k,j) = rate constant for production of product (k) in environment (j) where

$$\frac{d(B)}{dt} = K(k,j) \cdot E_{total} \cdot A$$

B = amount of PRDC(k,j) = amount of product (k) in environment (j).

$$E_{total} = E + EB + BEB$$

Given the three equations in three unknowns KK(k), P1 and P3 (Figure 3.3) one solves for the unknowns in the SOLVE routine of the program (Figure 3.4). Trial values of KB(k) and KBB(k) are used in the equations. All other quantities are available after data is collected for the simulated cell growing in each of its three environments.

Figure 3.5 illustrates the calculation of allosteric modification of simulated enzymes in the overall program of the simulated living cell; 1) Experimental data from three environments is stored. The three environments are mineral glucose, casamino acid enriched, and broth liquid media. 2) Equilibrium constants for enzymes are calculated such that overall enzyme activity agrees with experimental data. 3) Our model of allosteric modification is given a preliminary test by ascertaining whether it performs the function assumed in the model, and written into the simulation.

The cell calculated the correct rate constants for each enzyme system by simulating allosteric modification (Figure 3.6). The rate of production of each pool in the simulation was the same for simulated allosteric modi-

```
43      DO 44 II = 1,14
44      R(II) = 0.
          IF (NUC - 5.E7) 46,45,45
45      R(1) = .9
46      IF (AA - 5.E7) 48,47,47
47      R(2) = .9
48      IF (ATP - 1.E8) 50,49,49
49      R(3) = .9
50      IF (WALL - 4.E8) 52,52,51
51      R(4) = .9
 52     IF (ADP - 1.E5) 54,53,53
53      R(5) = .9
54      CONTINUE
        X = 1./(1. - .09*R(1) - .40*R(2) - .40*R(3) - .05*R(4)
     1 -.01*R(5))
        X = X*MRNA
        DO 10 II = 1,10
        K8K(II) = K(8)*RC(II)*(1 - R(II))*X/MRNA0
        P1 = P1V(II)
        P3 = P3V(II)
        C(II) = (  KK(II) + P1*KB(II)*PRDC K(II)
     1 + P3*KBB(II)*PRDC K(II) **2)/(1 + P1*PRDC K(II)
     2 + P3*PRDC K(II) **2  )
10      WRITE(6,114)II,C(II),II,K8K(II)
114     FORMAT(4H0 C(I2,2H)= 1PE13.6,5H K8K(I2,2H)= 1PE13.6)
```

Figure 3.1 Repression and Allosteric Inhibition (FORTRAN).

Repression is obtained by an adjustment of K8K(II). Allosteric inhibition is obtained through an adjustment of C(II).

```
      K(5) = (  (ADP0/T + ATP0/T)/NUC  )                *LN2
      K(6) = (DNA0/NUC0)/T                              *LN2
      KDRNK = LN2/60.
      K(8) = (LN2*MRNA0/T + KDRNK*MRNA0)/(NUC0*DNA0)
      K(10) = (TRNA0/NUC0)/T                           *LN2
      K(9) = RIB0 /(NUC0*AA0*T    )                   *LN2
      K(4) = WALL0/(GLUC0*T    )                       *LN2
      K(7) = PRTN0/(AA0*T    )                        *LN2
      KIN = (FACTR*CHRM0 - INZ)/(CHRM0*DBLE*(IN1Z+IN2Z) )
C**** 1ST DBLE MINUTES, CHRM0 GENES MAKE IN - INZ
C**** KIN IN IN PER GENE PER SEC
      DDNA = K(6)*NUC
      DMRNA = K(8)*NUC*DNA - KDRNK*MRNA
      DTRNA = K(10)*NUC
      DRIB = K(9)*NUC*AA
      DWALL = K(4)*GLUC
      DPRTN = K(7)*AA
C****
      K(1)=(LN2*NUC0/T + (2.5E9/660.)*DDNA+(1.E6/660.)*DMRNA
     1                 + (2.5E4/660.)*DTRNA + K(5)*NUC
     2 +(2.E6/660.)*DRIB)/GLUC0
C****
      IF(BROTH.EQ.1) K(1) = 0.1*K(1)
C****
      DNUC = K(1)*GLUC - (2.5E9/660.)*DDNA - K(5)*NUC
     1 -(1.E6/660.)*DMRNA
     2 - (2.5E4/660.)*DTRNA- (2.E6/660.)*DRIB
      K(2)=(LN2*AA0/T    +(1.E6/102.)*DRIB
     1 + (4.E4/102.)*DPRTN)/GLUC0
C****
      IF((CAA.EQ.1).OR.(BROTH.EQ.1)) K(2) = 0.1*K(2)
C****
      IF(ABS(COUNT).GT.1.AND.COUNT.LT.9.9) GO TO 110
      DAPO2 = (LN2*15.*1.E-8*6.02E23*38./6.)
     1 / (  3600.*22.4*1.E3*1.E3  )
      K(3) = DAPO2/GLUC0
      DNAP = (6.E4/.0033)*(2.5/2.)
      MRNAP = 7.5E4/12.5
      TRNAP = (7.5E4/12.5)*(2.5E4/1.E6)
      PRTNP = (2.12E6/1.4E3)*(4.E4/6.E4)
      RIBP = (7.5E4/12.5)*(2.E6/1.E6) + (2.12E6/1400.)*(1.E6/6.E4)
      WALLP = (2.E8/2.25E8)*(6.5E4/32.5)*(150./2.E6)
     1 +(.25E8/2.25E8)*(8.75E4/1.25E4)*(750./1.E3)
      NUCP = (K(3)*GLUC - DNAP*DDNA - MRNAP*DMRNA - TRNAP*DTRNA
     1 - RIBP*DRIB - PRTNP*DPRTN - WALLP*DWALL -ATP0/T*LN2
     2 - 2*K(5)*NUC)
     3 /(K(2)*GLUC + K(1)*GLUC + K(4)*GLUC)
      AAP = NUCP
      WALLP = WALLP + NUCP
  110 CONTINUE
      IF(ABS(COUNT-4.).LE.0.1.OR.ABS(COUNT-7.).LE.0.1)
     2 K(3) = (ATP/T*LN2+DNAP*DDNA + MRNAP*DMRNA + TRNAP*DTRNA +
     3 RIBP*DRIB + PRTNP*DPRTN + WALLP*DWALL + AAP*K(2)*GLUC +
     4 NUCP*K(1)*GLUC + 2*K(5)*NUC)/GLUC
      DATP = K(3)*GLUC - DNAP*DDNA - MRNAP*DMRNA
     1  - TRNAP*DTRNA - RIBP*DRIB - PRTNP*DPRTN
     2  - WALLP*DWALL - AAP*K(2)*GLUC - NUCP*K(1)*GLUC - 2*K(5)*NUC
      K(5) = (ADP*LN2/T + DATP)/NUC
      DADP = -DATP + K(5)*NUC
      K(14) = VOL0/(T    *WALL0)                        *LN2
      DVOL = K(14)*WALL
```

Figure 3.2 Preliminary Rate Constants K(1), ..., K(14) for Flow Rates Between Pools.

These rate constants are later used to calculate enzyme rate constants.

```
   16  DO 13 ID = 1,10
         XK(ID,J) = C(ID)
         XEK(ID,J) = EK(ID)
          PRDC (ID,J) = PRDC K(ID)
13    XK8K(ID,J) = K8K(ID)
```

Figure 3.3 Storing different variables (which variable is indicated by the ID subscript) for the cell in different environments (which environment is indicated by the J subscript.

```
      DO 12 L = 1,10
      IX = L
      P1 = 10./PRDC0(L)
      P3 = 100./PRDC0(L)**2
      P1V(L) = P1
      P3V(L) = P3
****  REPLACE INTERNAL FUNCTION SOLVE(IX)
 17    DO 14 II = 1,3,1
      A2(II) = P1*PRDC (IX,II)
      A3(II) = P3*PRDC (IX,II)**2
14    SUM(II) =XK(IX, II) * (1. + A2(II)+ A3(II))
      DUM1 = A2(3) - A2(1)
      DUM2 = A2(2) - A2(1)
      DUM3 = A3(2) - A3(1)
      KBB(IX) = (SUM(3)-SUM(1))-(DUM1/DUM2)*(SUM(2) - SUM(1))
     1 /(A3(3) - A3(1)  - (DUM1/DUM2)*DUM3)
      KB(IX) = ( SUM(2) - SUM(1)  -  DUM3*KBB(IX))/DUM2
      KK(IX) = SUM(1) - A3(1)*KBB(IX) - A2(1)*KB(IX)
```

Figure 3.4 Calculations for allosteric inhibition necessary to fit data for real cells.

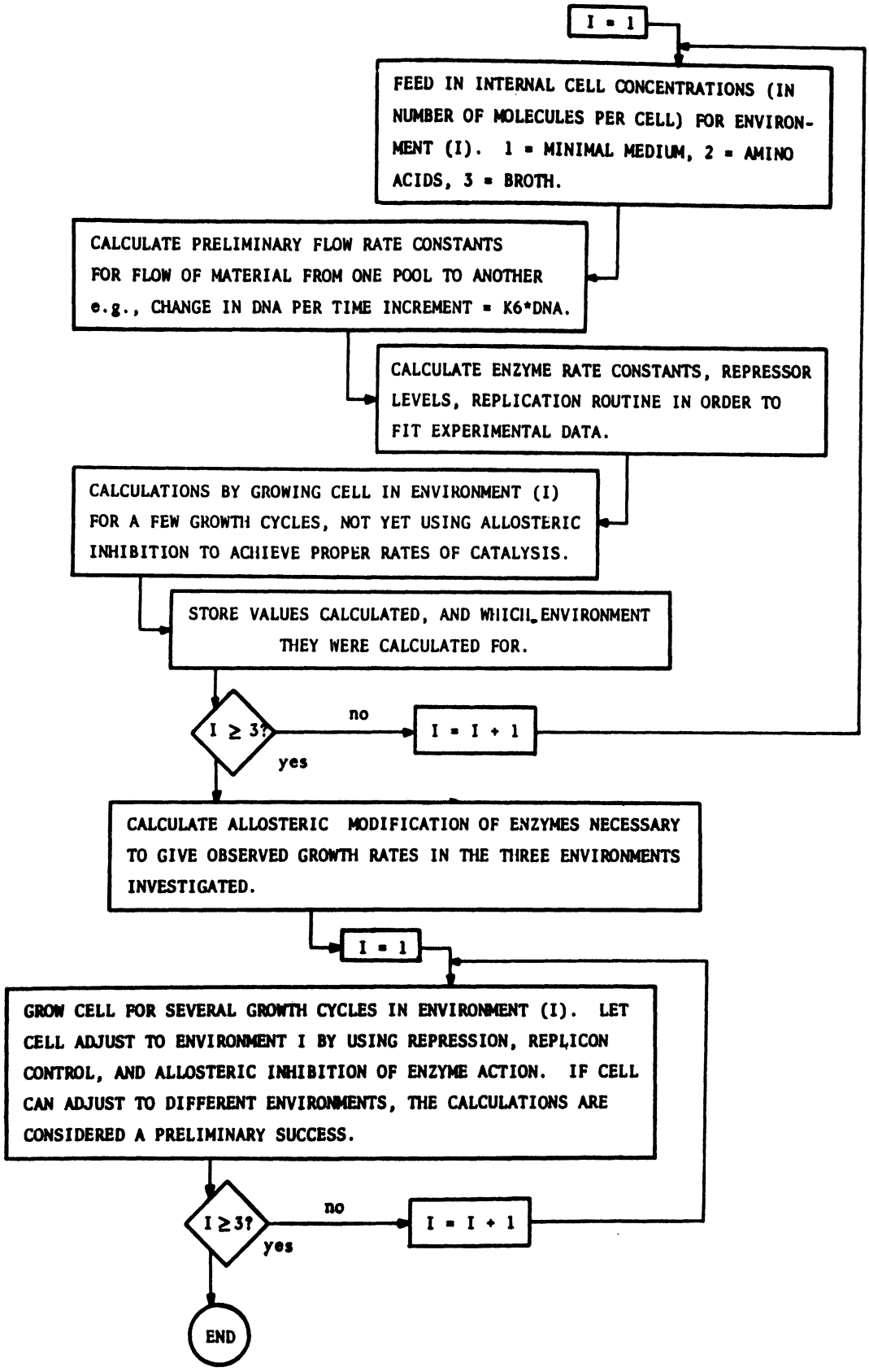


Figure 3.5 Simulated Allosteric Modification within the Overall Program

K(1)=0.146960E-11	XK(1 1)=0.146982E-11
K(2)=0.218303E-11	XK(2 1)=0.218304E-11
K(3)=0.925982E-04	XK(3 1)=0.849804E-04
K(4)=0.690331E-11	XK(4 1)=0.690330E-11
K(5)=0.456416E-14	XK(5 1)=0.456499E-14
K(6)=0.111028E-19	XK(6 1)=0.111028E-19
K(7)=0.153918E-14	XK(7 1)=0.153918E-14
K(8)=0.130453E-16	XK(8 1)=0.130452E-16
K(9)=0.277559E-23	XK(9 1)=0.277571E-23
K(10)=0.153920E-14	XK(10 1)=0.153918E-14
Rate constants calculated at start of program	Rate constants by simulating "Allosteric Modification"

Figure 3.6 The Cell Successfully Adjusted its Enzyme Rate Constants by Simulated Allosteric Modification. Above figures are for a cell in Minimal Medium. E indicates multiplication by a power of 10. e.g., 1.469818E-12 = $1.469818 \cdot 10^{12}$

fication as for experimental results on logarithmic phase growing cells (Figure 3.7). Therefore, simulated allosteric modification maintained correct equilibrium ratios of major cell constituents, as compared to real living cells in laboratory experiments (Figure 3.7).

These preliminary results assure us that the Fortran program is running correctly. This kind of test is the first step in running most simulations, and will be followed by demonstrations of the usefulness of the simulation. The simulated cell can predict laboratory results not assumed in its formulation, and clarify the role of allosteric modification in global metabolism (Chapter V). The program is useful for studies of the theory of modelling (Chapters IV, V), as well as for examining interacting populations of living cells in a cell space (Chapter VII).

RATIO(1)=0.100000E 01	RATIO(1)=0.100000E 01
RATIO(2)=0.100000E 01	RATIO(2)=0.999999E 00
RATIO(3)=0.100000E 01	RATIO(3)=0.999996E 00
RATIO(4)=0.100000E 01	RATIO(4)=0.100000E 01
RATIO(5)=0.100000E 01	RATIO(5)=0.100004E 01
RATIO(6)=0.100000E 01	RATIO(6)=0.100000E 01
RATIO(7)=0.100000E 01	RATIO(7)=0.100000E 01
RATIO(8)=0.100000E 01	RATIO(8)=0.100000E 01
RATIO(9)=0.100000E 01	RATIO(9)=0.100000E 01
RATIO(10)=0.100000E 01	RATIO(10)=0.100000E 01
Calculated Algebraically from Experimental Data	Simulated Using Allosteric Modification

Figure 3.7 Steady State Concentrations. Ratio(i) indicates the ratio of pool(i) to its proper steady state concentration. E indicates multiplication by a power of 10, e.g., .99999E 00 = $.999999 \cdot 10^{0}$ = .999999. The indices for the pools are: 1=NUC 2=AA 3=ATP 4=WALL 5=ADP 6=DNA 7=PROTEIN 8=MRNA 9=RIBSOME 10=TRANSFER RNA.

CHAPTER IV

STRUCTURE OF THE SIMULATION, AN EXAMPLE OF COMPLEXITY REDUCTION BY AGGREGATION (Zeigler & Weinberg, 1970)

The abstract structure of the computer simulation of a living cell illustrates the valid simplification possible in a homomorphic mapping. Our particular homomorphic mapping was from the measurement-space of laboratory-data obtained from experiments on real living-cells to the simplified model of a cell used as a basis for our simulation. Simplification serves as a general modelling technique, and may be viewed as a simplification through aggregation (lumping) of coordinates. In order to approach simplification from this viewpoint, we must use a coordinate description to represent our state-space. If aggregation produces a homomorphism, we will say we have a valid aggregation. The formal development of this aspect of the simulated living cell is being investigated by Zeigler (1970), who is extending the example of the simulated living cell to a general theory of model building. The technique of simplification of real systems for modelling and simulation is outlined in the following chapter, using the technique of aggregation of coordinates (Zeigler and Weinberg, 1970).

Computer Simulation Complexity Measures: We can easily discuss the kinds of complexity measures available to us, since we have abstracted the formal structure of our simulated living cell. The analysis of computational complexity of a working simulation is relevant and valuable. Although investigation of computational complexity is actively being pursued by computer scientists (e.g., A.C.M. Symposium

on Theory of Computing, 1969), little attention has been devoted to measures of complexity relevant to simulation of a model system on a computer. We now introduce two very basic formal measures which embody the ideas of state and transition-function, as related to complexity. These measures are directly related to the representing digraph (directed graph) of a system. One is related to the memory capacity required to store the current *state*[1] of the system; we require at least $\log_2|S|$ bits of storage if there are $|S|$ states. Thus our first measure is $memory(S) = \log_2|S|$. Since $S = \underset{\alpha\in D}{X} S_\alpha$ we have $memory(S) = |D| \cdot \left(\frac{1}{|D|}\sum_{\alpha\in D}\log|S_\alpha|\right)$ which indicates the proportionality of the memory capacity to the number of coordinates $|D|$ if all state sets are the same size.

We can also measure the complexity associated with the *transition-function* of the system. The transition-function is coded as a computer program. Assume (not unrealistically) that the number of instructions necessary to encode τ_α is proportional to the number of coordinates in its domain, i.e., to $|I_\alpha|$. Then the length of the program for τ is proportional to the sum $\sum_{\alpha\in D}|I_\alpha|$. Thus our second measure is $length(D) = \sum_{\alpha\in D}|I_\alpha|$, which is, in fact, just the number of lines in the digraph. Also, the time to run the program (for one state transition) is "somewhat" proportional to its length. The existence of iterations complicates this relationship.

Both $memory(S)$ and $length(D)$ cannot exceed the bounds set by the computational capacity of real computers. It is this fact, among others, which would thwart attempts to use the state spaces at the atomic, molecular, and concentration levels of Figure 4.1. In other words, were we to try describing the state of a biological cell by listing the states

STATE SPACE	COORDINATES	COORDINATE SETS	TRANSITION FUNCTION
Atomic	Electrons, nuclei	Position, momentum	Schrodinger's equation (quantum mechanics)
Molecular	Molecules	Shape, active site, energy level	e.g. Koch probabilistic model
Concentration	Molecule Type e.g.ADP,DNA	Number of molecules of ADP/cell, etc.	Mathematical logical equations based on chemical kinetics e.g. Chance et al
Higher level groupings	Bio-chemical pools e.g. amino acids	Number of molecules of pool/cell	Mathematics and logical equations based on chemical kinetics and Molecular control mechanisms e.g. repression, allosteric inhibition (Weinberg, Goodwin)

Figure 4.1 Possible state-spaces and transition-functions for models of the living cell are shown. The coordinates at one level are aggregations of the coordinates at a lower level of organization.

of each of the elementary particles of which the cell is composed, (the atomic model), we would hardly have enough data storage capacity to keep track of such a long list. Moreover, in our program we would not have enough storage capacity to specify how each atomic state changes as a function of the prior states of the atoms which influence it. (The time required to run such a program would exceed a scientist's patience if not his lifetime.) In our new terminology, then, we seek model systems whose memory and length complexities are small enough to enable an actual simulation on a real computer.

4.2 Complexity Reduction by Aggregation

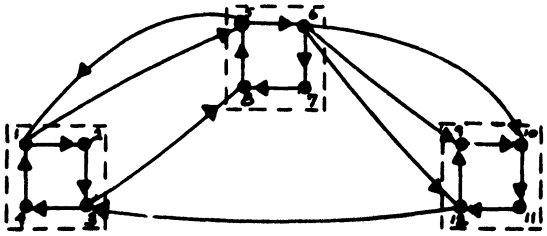
Let us illustrate how different coordinate descriptions may define different state-spaces, or may define the same state-space. We will use an example related to our simulated cell. We will first assume two coordinates, SERINE and ALANINE, to describe the concentrations of two pools, the serine pool and the alanine pool. The state-space describable by this two-coordinate system consists of all combinations of values of SERINE and ALANINE. But we may describe the same state-space with only one coordinate, (SERINE,ALANINE). In this one coordinate description, the first entry is the concentration of the serine pool, and the second entry is the concentration of the alanine pool. A serine concentration of 6, and an alanine concentration of 4 is indicated as (6, 4). We will now show how a one-coordinate system may have a reduced state-space. Let the one coordinate AMINO indicate the sum of the serine and alanine concentrations. This one coordinate system has a reduced state-space. A serine concentration of 6 and an alanine concentration of 4 gives an AMINO value of 10. So does a serine concentration of 4

and an alanine concentration of 6. We can no longer distinguish these two different states, because in both states the sum of SERINE and ALANINE is 10. Summarizing:

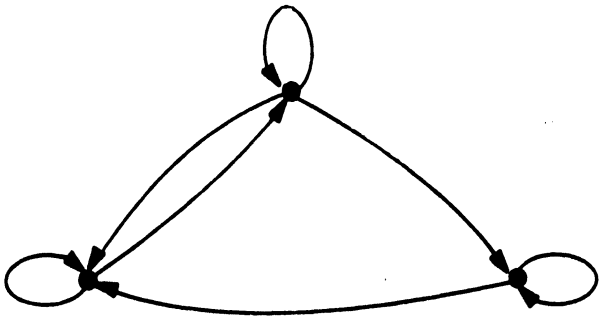
COORDINATES	NUMBER OF COORDINATES	STATE SPACE
SERINE ALANINE	2	all concentrations of serine and alanine.
(SERINE, ALANINE)	1	same as above.
AMINO	1	reduced to all distinguishable sums of serine and alanine concentrations.

We shall now give a fairly general mathematical formulation for the process of aggregation or lumping by which relatively simple systems are derived from more complex ones. This kind of process was illustrated in the preceding example and is basic to the model of the *E. coli* cell from which it was taken. (Recall the manner in which the chemicals in the cell were combined into pools (Figures 2.1, 2.2).) We shall relate the aggregation (lumping) process to the more basic notion of homomorphism, deriving thereby necessary and sufficient local conditions which guarantee that such a process will yield a valid homomorphic image system. (Recall that homomorphism means 1) lumping coordinates and then using a transition-function gives the same result (state) as, 2) using a transition-function and then lumping.) At the same time we shall see how the complexity measures introduced before may be simultaneously reduced by aggregation (Figure 4.2).

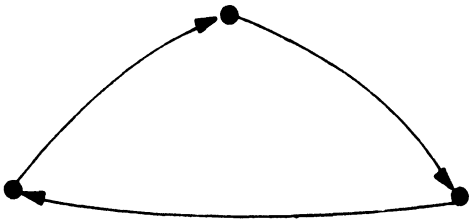
Suppose then we are given a system (S,τ) with coordinates D, state space $S = \underset{\alpha\in D}{\times} S_\alpha$ and transition-function given by $\{\tau_\alpha \mid \alpha\in D\}$ where $\tau_\alpha: \underset{\beta\in I_\alpha}{\times} S_\beta \to S_\alpha$. We wish to construct a homomorphic image of S by aggre-



a) Digraph of System S



b) Digraph of System S'



c) Digraph of System S"

System	# of co-ordinates	size of each co-ordinate state set	size of state space	storage capacity ($\|D\| \log_2 \|S_\alpha\|$)	program length (# lines i digraph)
S	$\|D\| = 12$	$\|S_\alpha\| = 8$	$\|S\| = 8^{12}$	*memory*(S) = 36	*length*(D) = 19
S'	$\|D'\| = 3$	$\|S'_\alpha\| = 8^4$	$\|S'\| = 8^{12}$	*memory*(S') = 36	*length*(D') = 7
S"	$\|D''\| = 3$	$\|S''_\alpha\| = 8$	$\|S''\| = 8^3$	*memory*(S") = 9	*length*(D") = 3

Figure 4.2 Illustrating the Aggregation Process. A system S with 12 co-ordinates each having a state set of size 8 is simplified to a system S" having 3 co-ordinates each having the original state set size (8). The intermediate system S' is obtained partitioning the 12 co-ordinates of S into 3 blocks of 4 co-ordinates each (1' = {1, 2, 3, 4}, 2' = {5, 6, 7, 8}, 3' = {9, 10, 11, 12}) with state set size 8^4. Finally S" is obtained from S by reducing the state set size of each co-ordinate to 8. S" is less complex than S with respect to the memory and length complexity measure.

gating coordinates. Recall from our example that this process can be decomposed into a composition of two sub-processes. In Process 1 the coordinates are lumped together but the state-space is not reduced. In Process 2 the coordinates of the system are retained but the state-space is reduced. Figure 4.2 illustrates the application of these processes.

For Process 1, partition the coordinate set D into blocks B_1, B_2,... (where the blocks are mutually disjoint subsets of D which together exhaust it). The new system (S', τ') will have as coordinates $D' = \{B_i\}$, the state set of coordinate B_i will be $S_{B_i} = \underset{\alpha \in B_i}{x} S_\alpha$ and the state-space is then $S' = \underset{B_i \in D'}{x} S_{B_i}$. Note that S' = S so that the state-space has not really been altered in this process. You see S' really does equal S if we just describe S by different symbols. Define the transition-function for any coordinate B_i as

$$\tau'_{B_i}(s) = (\tau_{\alpha_1}(s), \tau_{\alpha_2}(s), \ldots, \tau_{\alpha_n}(s))$$

for each $s \in S'$ (=S) where $B_i = \{\alpha_1, \alpha_2, \ldots \alpha_n\}$. One can verify that the subset of D' on which τ'_{B_i} actually depends is $I'_{B_i} = \{B_j \mid B_j \cap J_{B_i} \neq \emptyset\}$ where $J_{B_i} = \bigcup_{\alpha \in B_i} I_\alpha$. This is because J_{B_i} is the set of coordinates on which the functions $\{\tau_\alpha \mid \alpha \in B_i\}$ depend and I'_{B_i} is the set of blocks which contain coordinates in J_{B_i}.

It is readily verified that the system (S', τ') so defined is isomorphic to the system (S, τ). (Under the identity map $i: S \to S'$, $i(s)=s$).

Constructing the digraph (directed graph) D' from the sets I'_{B_i} we find that there is a line from B_i to B_j in D' just in case there are

points α in B_i and β in B_j and there is a line from α to β in the digraph D. (This is called the *condensation* of D with respect to the partition D' by Harary, Normand and Cartwright (1965)). This means that while the *number of points in the digraph may be made to decrease in going from* D *to* D', *the total number of lines in the digraph cannot increase.*

In Process 2 starting with a system (S, τ) with coordinates D and state-space $S = \underset{\alpha\varepsilon D}{x} S_\alpha$ we construct a system (S', τ') *also having coordinates* $\underline{D}$ but with (possibly smaller) state-space $S' = \underset{\alpha\varepsilon D}{x} S'_\alpha$. We assume that for each α there exists an onto map $h_\alpha: S_\alpha \to S'_\alpha$ and define the map $h: S \to S'$ given by $h(s_\alpha, s_\beta, s_\gamma, \ldots) = (h_\alpha(s_\alpha), h_\beta(s_\beta), h_\gamma(s_\gamma), \ldots)$ for each $(s_\alpha, s_\beta, s_\gamma, \ldots) \varepsilon S$. We seek a condition which makes h a homomorphism. The condition, which is both necessary and sufficient, is that for each $\alpha \varepsilon D$,

$$h_\alpha(\tau_\alpha(s_{\beta_1}, \ldots, s_{\beta_n})) = \tau'_\alpha(h_{\beta_1}(s_{\beta_1}), \ldots h_{\beta_n}(s_{\beta_n}))$$

for all $(s_{\beta_1}, \ldots s_{\beta_n}) \varepsilon \underset{\beta\varepsilon I_\alpha}{x} S_\beta$. We call this the condition of *preservation of coordinate functionality* since it translates the *global* requirements for S' to be a homomorphic image of S into *local* requirements involving the transition-functions τ_α and τ'_α at each coordinate α. In particular if for any α a function τ'_α can be made to satisfy the condition, then it *depends at most on the coordinates* I_α. In fact if any simplification is achieved I'_α will be a proper subset of I_α. This means that D' is a *subdigraph* of D, i.e., if a line from α to β appears in D' it must also appear in D. Thus in this case, in going from S to S' the *total number of lines may be made to decrease* (it cannot increase).

To establish the condition, we use the maps, $\text{proj}_\alpha: S \to S_\alpha$ where

$proj_\alpha(s)$ is the projection of s on the α coordinate. Thus $proj_\alpha(h(s)) = h_\alpha(s)$ and $proj_\alpha(\tau(s)) = \tau_\alpha(s)$ for each $\alpha \varepsilon D$. Then $\tau'(h(s)) = h(\tau(s))$ if, and only if, $proj_\alpha(\tau'(h(s)) = proj_\alpha(h(\tau(s)))$ for each $\alpha \varepsilon D$. Now $proj_\alpha(\tau'(h(s)) = \tau'_\alpha(h(s))$ and $proj_\alpha(h(\tau(s)) = h_\alpha(\tau_\alpha(s))$ so that h is a homomorphism if, and only if, $\tau'_\alpha(h(s)) = h_\alpha(\tau_\alpha(s))$. We shall use the notation $(\cdot)$ to indicate a place for an argument of a function. Now $h_\alpha(\tau_\alpha(\cdot))$ is a function of at most the coordinates in I_α and for the equality to hold for *all* s ε S it can be readily seen that $\tau'_\alpha(h(\cdot))$ is necessarily a function of these same coordinates. This establishes the necessity of the condition as given before. Its sufficiency is easily verified.

Suppose that starting with a system (S,τ); 1) we apply Process 1 to arrive at a system (S',τ') with fewer coordinates than (S,τ) but with the same state-space, and then, 2) we apply Process 2 to (S',τ') to arrive at a system (S'',τ'') with a smaller state-space than S'. D'', the digraph of (S'',τ''), will have fewer lines than D, the digraph of (S,τ). Because of its smaller state-space *memory*(S'') < *memory*(S) and because it has fewer lines *length*(D'') < *length*(D). The net result is that (S'', τ'') is a homomorphic image of (S, τ). The complexity of (S'', τ''), as determined by the memory and length complexity measures, is less than the complexity of (S, τ).

Note that the digraph D'' is a subdigraph of a condensation of the digraph D. This means that both state behavior and the *local structure* have been preserved in going from (S, τ) to $S'', \tau'')$. The *local structure* refers to the manner in which the coordinates are interdependent. This preservation of local structure is *not* a necessary consequence of the preservation of state behavior. In general, a system (S', τ') may be a homomorphic image of a system $\S, \tau)$ while D' is not a subdigraph of a conden-

sation of D. The preservation of local structure *is*, however, a consequence of the aggregation process and the requirements for homomorphism. Preservation of local structure implies we may be able to study behavior of parts of our simplified model, and apply the results of such studies to the behavior of analogous blocks back in the original, unsimplified system. We often wish to use computer simulations in this manner.

Aggregated Models and Simulation: In this chapter we have utilized system-theory concepts to describe and analyze the basic character of our simulation of a living cell. In contrast to other simulations of cell metabolism, we simplified our model by lumping (aggregating) the various molecular species into pools. We then attempted to simulate the behavior of these pools over time. This aggregation technique greatly reduced the complexity of our simulation but raised the issue of the validity of our model. In this regard, we gave necessary and sufficient conditions which must obtain if an aggregated system is to model (in the sense of being an exact homomorphic image) the original system. These conditions were used to show that the memory and length measures of complexity can be simultaneously reduced in a valid aggregated model.

In contrast to models in general, aggregated models preserve the local structure of the coordinate-dependency digraph. Recall that lines with arrows connect functionally related coordinates in the coordinate-dependency digraph. This preservation of local structure by aggregated models is important.

The greater is this preservation of structure, the more readily can components of the model be identified with corresponding components of the real world biological system, and the more completely can their

behavior be compared. (The identification of components, i.e., systems formed by subsets of the coordinate set, can be accomplished easily when the coordinates of the system and a model are related by a (univalent) mapping as for aggregated models, but is much more difficult in the more general case where the relation between coordinates may be many to many.) As we have indicated, the possibility of checking the behavior of components of our model of the cell against that of corresponding real world components played a significant role in determining the assignment of molecules to pools.

We may conclude that system-theory concepts help us to understand and formulate problems encountered by biologists and other scientists in modelling and simulation. At the conceptual level, these concepts help explicate the notions of model complexity and adequacy. At the more practical level they may suggest and justify techniques for the reduction of complexity, and for the testing of model adequacy.

In our particular simulation, we have used formal concepts from system theory. The formal concepts 1)helped us to simplify a complex system, and 2)provided criteria for validating our simplified model.

In particular, the aggregation of entities as a modelling technique has been given a system-theoretic formulation. This technique is fundamental to our model of the living cell and may provide a powerful tool for other simulation studies.

FOOTNOTES TO CHAPTER IV

1. Our definition of "system" is essentially that routinely employed in automata theoretic formulations. (For example, Hartmanis and Stearns (1968)). A more general definition (incorporating continuous time) is given by Kalman, Falb and Arbib (1969) (pages 5-12). Arbib (1969)(pages 51-57) traces the steps involved in restriction to discrete time systems and finite state systems.

   We will give two examples: 1) a continuous system and 2) a discrete system. 1) Our continuous system uses a differential equation to update the value, x, of its state:

   S = all real numbers

   $\frac{dx}{dt} = 2 \cdot x$

   The differential equation gives us a way of determining the state of the system at some future time, given the value of x at the present. 2) Our discrete system is analogous to our continuous system. It has a time step of 1, and uses the transition function $\tau(x)$ to obtain the value of x at time t+1 from the value of x at time t.

   S = all integers

   $\tau(x) = x + 2 \cdot x$

2. Given any subset $T \subseteq X_{\alpha \in D} S_\alpha$, a coordinate α *depends on a set* I_α^T if $(\forall s, s' \in T)(\forall \beta \in I_\alpha^T)(\text{proj}_\beta(s) = \text{proj}_\beta(s') \Longrightarrow \tau_\alpha(s) = \tau_\alpha(s'))$

   and no proper subset of I_α^T has this property. In other words, a coordinate depends on just those coordinates which influence the change in its value over time. This formal treatment of dependency

allows us to describe in exact fashion the functional relationships existing among variables in a model. This definition is that given in Zeigler (1968, 1969) and the well-known partition pair calculus of Hartmanis and Stearns (1966). This criterion selects a minimal set I_α^T such that a function defined only on $\underset{\beta\varepsilon I_\alpha^T}{X} S_\beta$ can be found which equals τ_α on T.

Assuming a finite number of coordinates (though not necessarily a finite state set) will guarantee the existence of such sets; more generally a minimum condition is needed.

We are not guaranteed, however, that for each α there is only one set I_α^T on which it depends. Nevertheless, when $T = \underset{\alpha\varepsilon D}{X} S_\alpha$ these sets are unique. This situation is discussed by Zeigler (1968) and Roosen Runge (1967).

For each choice of sets $\{I_\alpha^T | \alpha\varepsilon D\}$ we can construct an associated digraph D^T. Thus to each subset of the state-space there cooresponds one (or more) well-defined digraphs. In particular, there is a unique well-defined digraph associated with the whole space $\underset{\alpha\varepsilon D}{X} S_\alpha$.

3. Of course the situation is much more complicated than this and our complete model will represent this fact. For example, coordinate interdependences arising from the allosteric modification, and also by repression components of the transition-function are reflected in the digraph.

4. Kauffman's state-space is the discrete set of all genome off-on configurations. This is an abstraction of the state-space of our model. Our discussion of homomorphism can be used to relate the two models.

CHAPTER V

RESULTS OF THE SIMULATION

We obtained two major kinds of results from our simulation: 1) results which showed that for the variables tested, the simulated cell behaved like a real cell; 2) results which could be used to investigate the role of allosteric modification in the metabolism of a real cell. To obtain the first sort of result (simulated behavior comparable to real-cell behavior) two tests were performed on the simulated cell. 1) The equations built into the simulation were tested to see whether they gave back the results used to formulate the model of the simulated cell. This test was successfully passed when the simulated cell grew correctly in three environments. 2) The simulated cell was tested to see if it could grow under conditions which had not been specifically used to determine the parameter settings which characterize the simulation. The ability of the simulated cell to grow like a real cell both in new medium, and during shifts in environments, indicated that the cell had passed this second test. Laboratory and simulated shift-up data were compared for this test (Figures 5.2- 5.10). (Most of the laboratory data was taken from Maaloe and Kjeldgaard, 1966). The other major type of result (the result used to investigate allosteric modification) was obtained by varying parameters in simulated experiments.

Let us now list the specific results we obtained from the simulation, and discuss these results in a bit more detail.

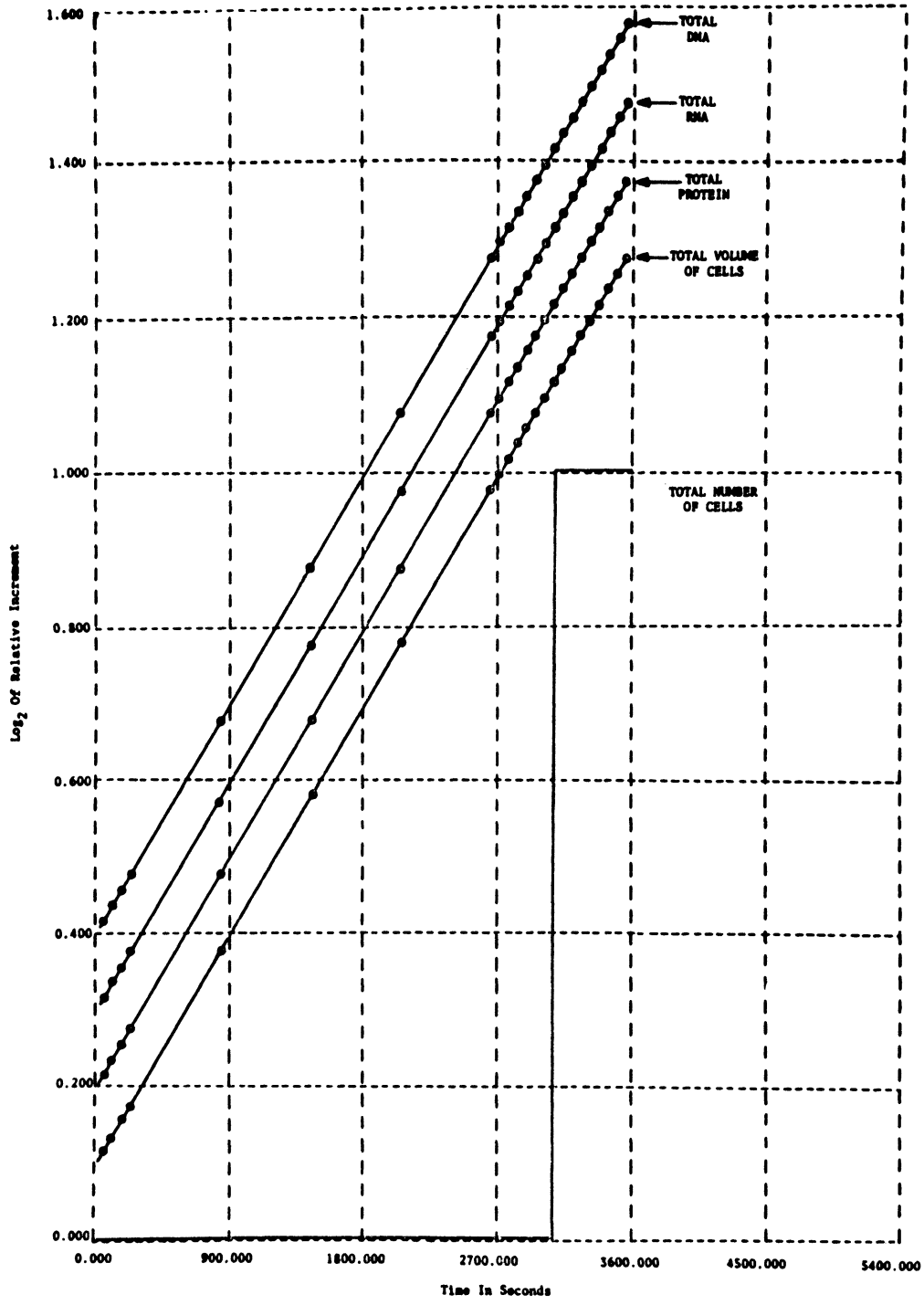


Figure 5.1 Logarithm of Various Quantities in a Growing Culture. Logarithmic Increase in Cell Mass over Time, Stepwise Increase in Number of Cells. Comparative magnitude of various quantities are a function of the scaling factors used in order to plot all quantities on one graph. A cell doubles after a DNA replication cycle. The doubling takes a relatively short time, as indicated by the sudden, stepwise increase in "TOTAL NUMBER OF CELLS", whereas "TOTAL DNA" increases throughout the replication cycle.

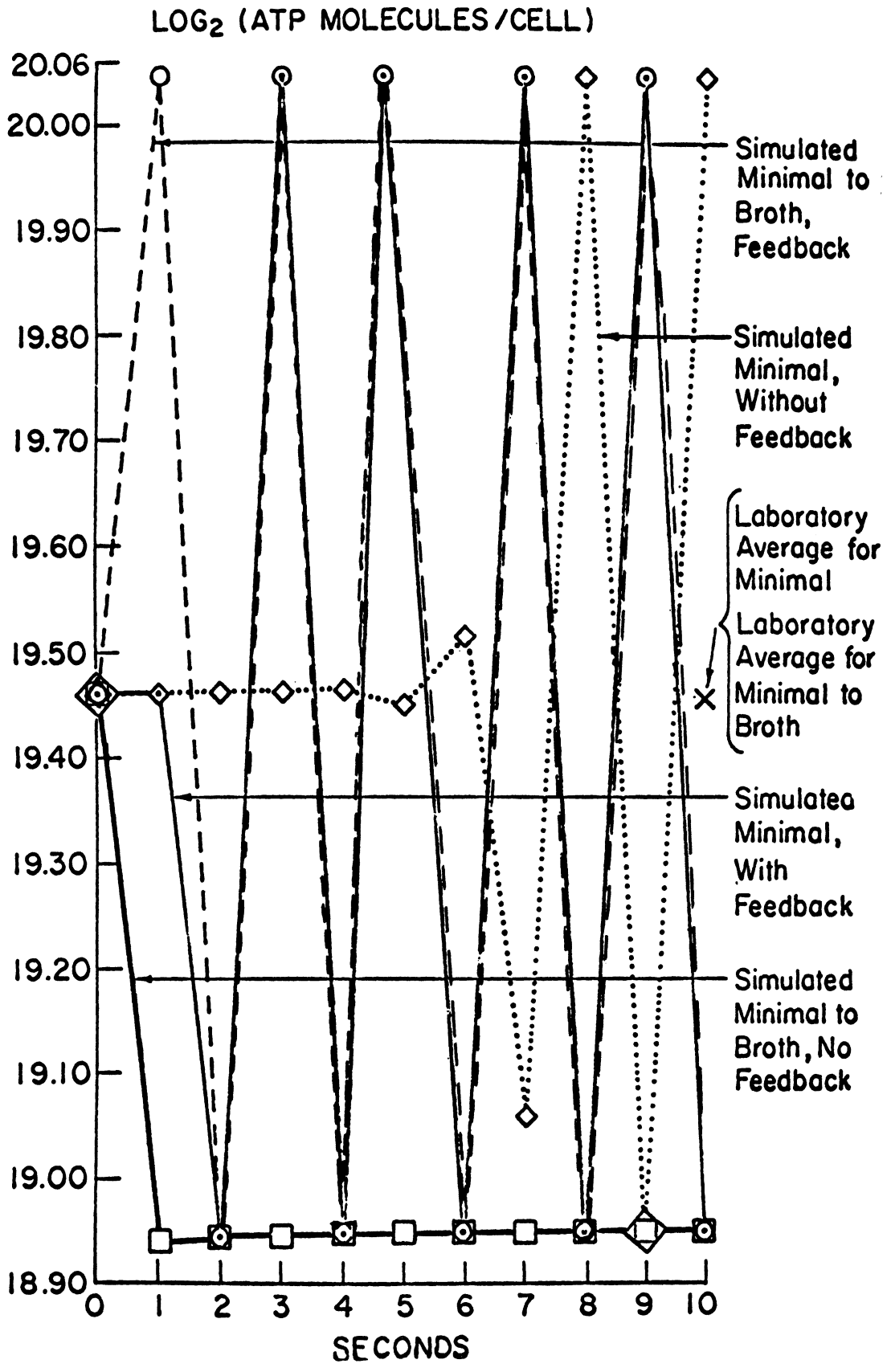


Figure 5.2 ATP during Simulated Shift Up.

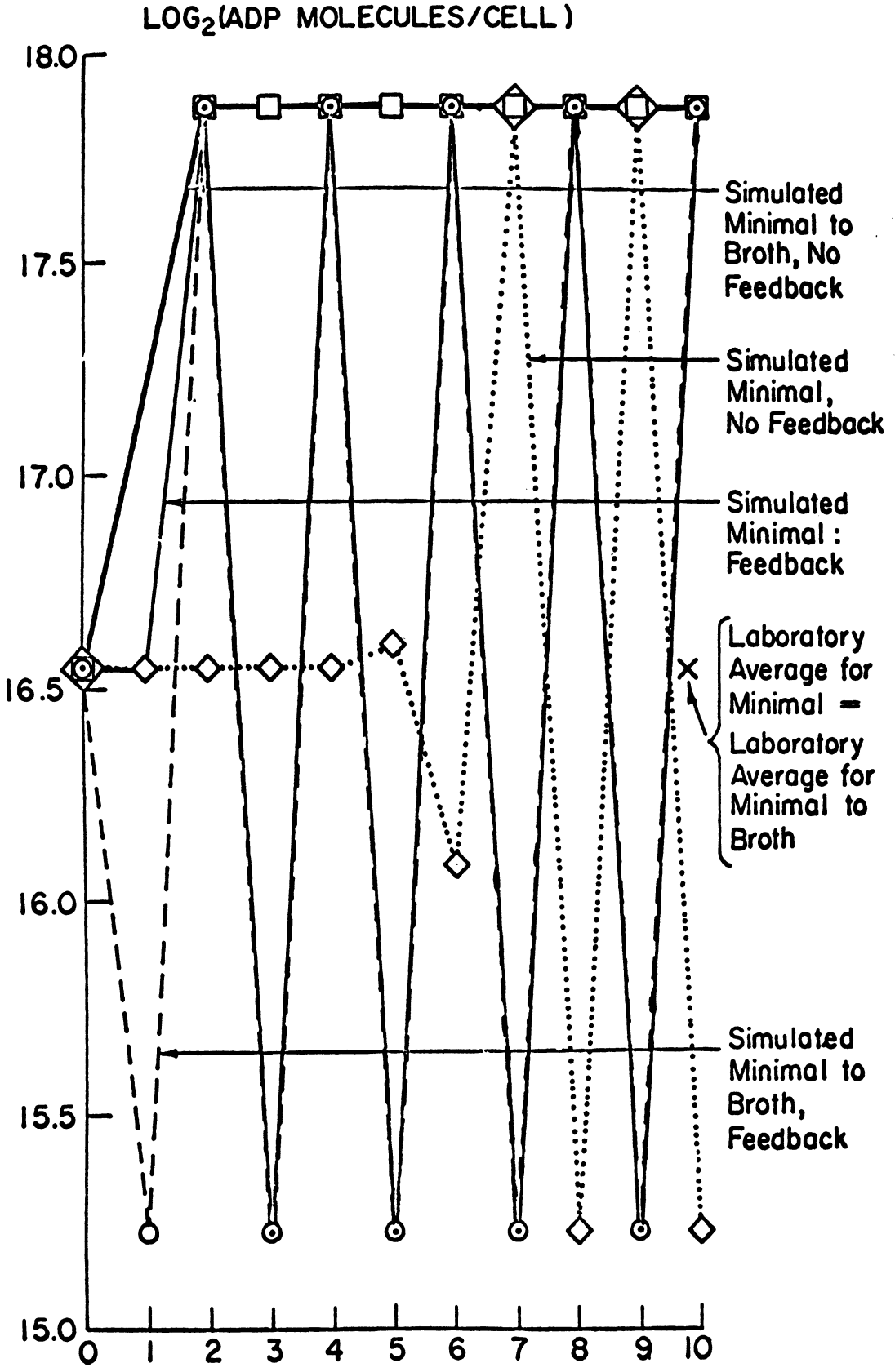


Figure 5.3 ADP during Simulated Shift Up.

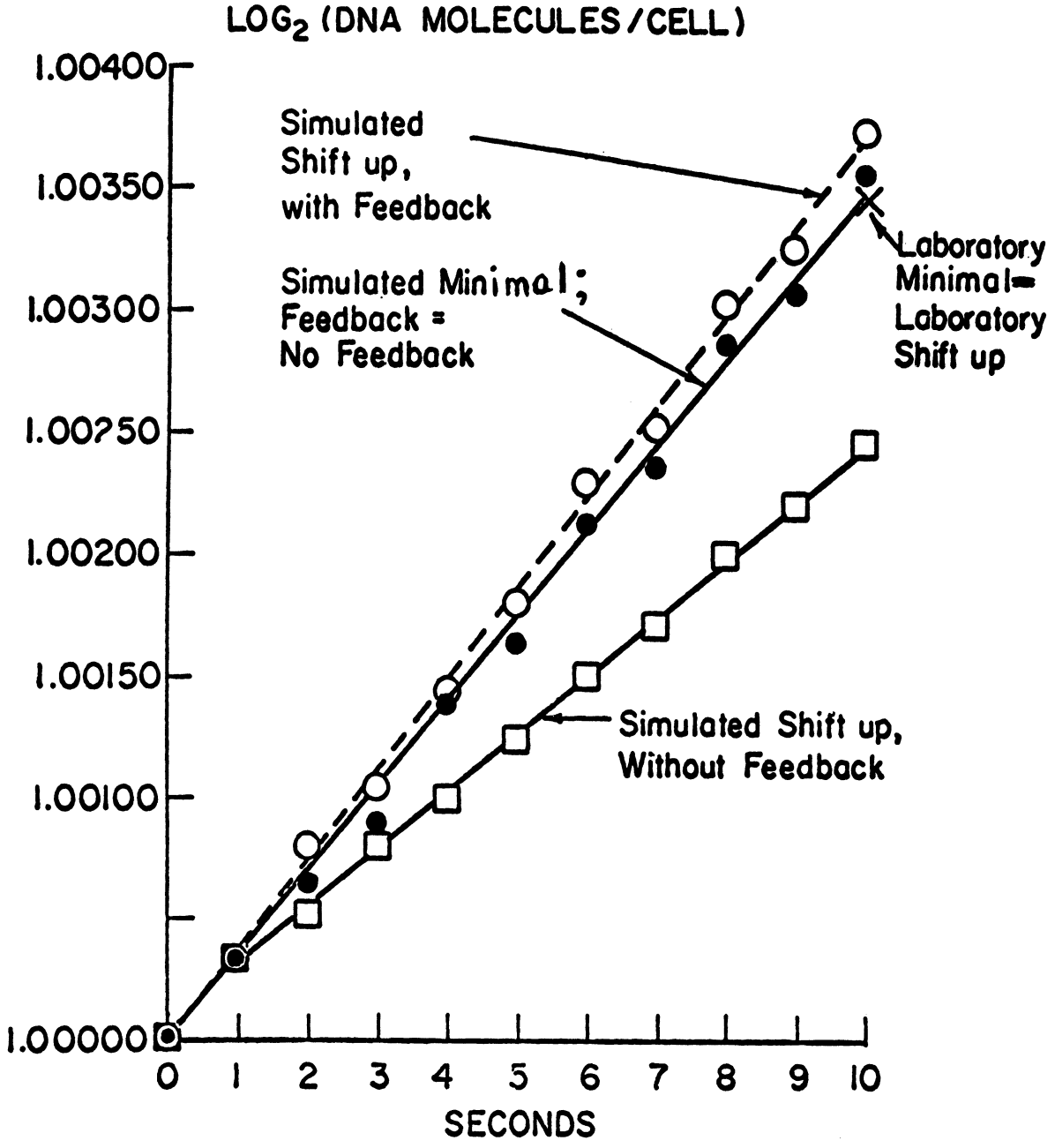


Figure 5.4 DNA During Shift Up. Well aerated liquid cultures at 37°C were used for simulated laboratory data. Symbols: ○ simulated shift from minimal glucose to broth at time zero; □ simulated shift without feedback from minimal glucose to broth at time zero; ● simulated minimal; X laboratory shift up = laboratory minimal glucose.

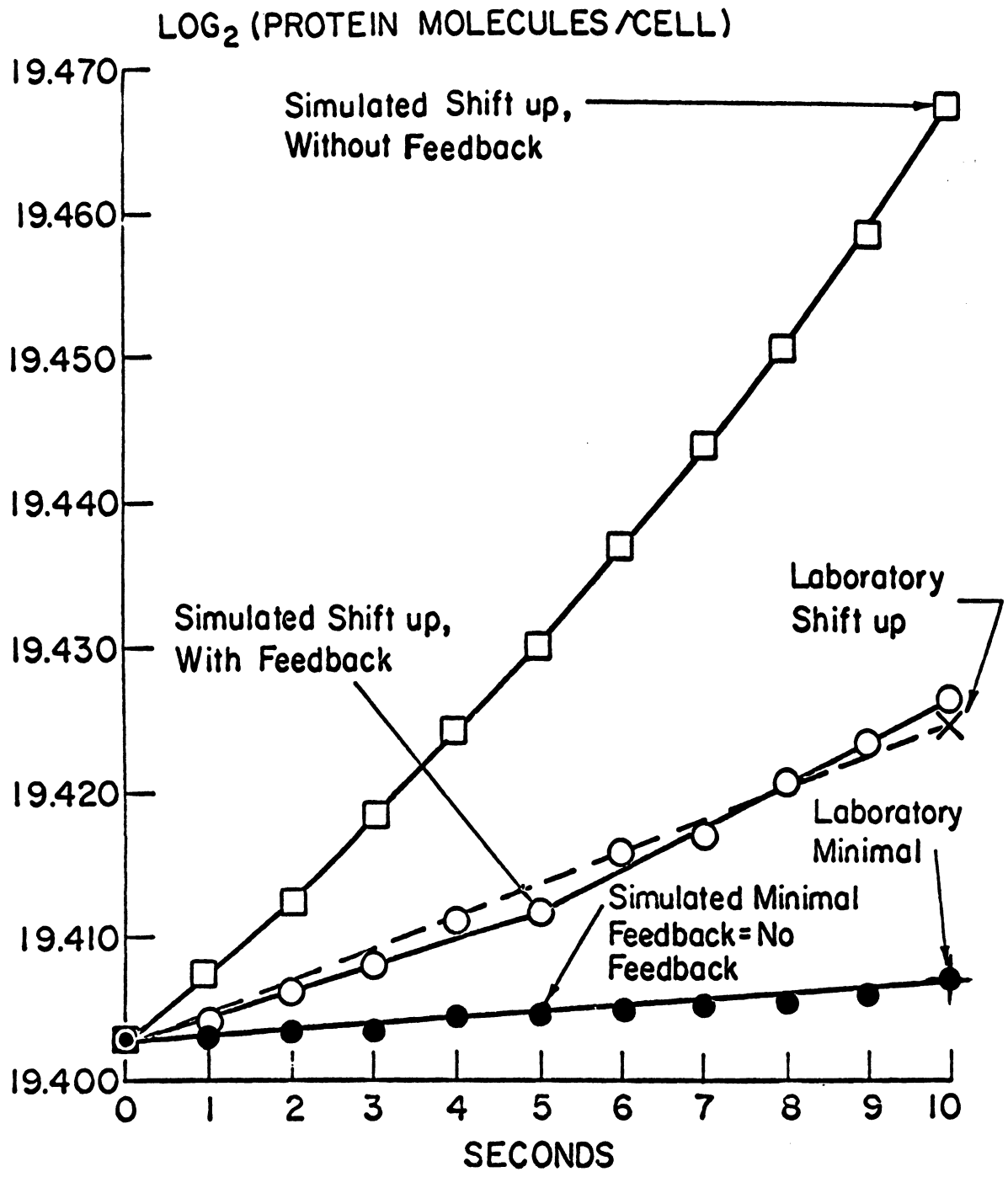


Figure 5.5 Protein During Shift Up. Well aerated liquid cultures were used for laboratory and experimental data. Symbols: □○ simulated shift from minimal glucose to broth at time zero; ● simulated minimal glucose control; X laboratory shift up; + laboratory minimal.

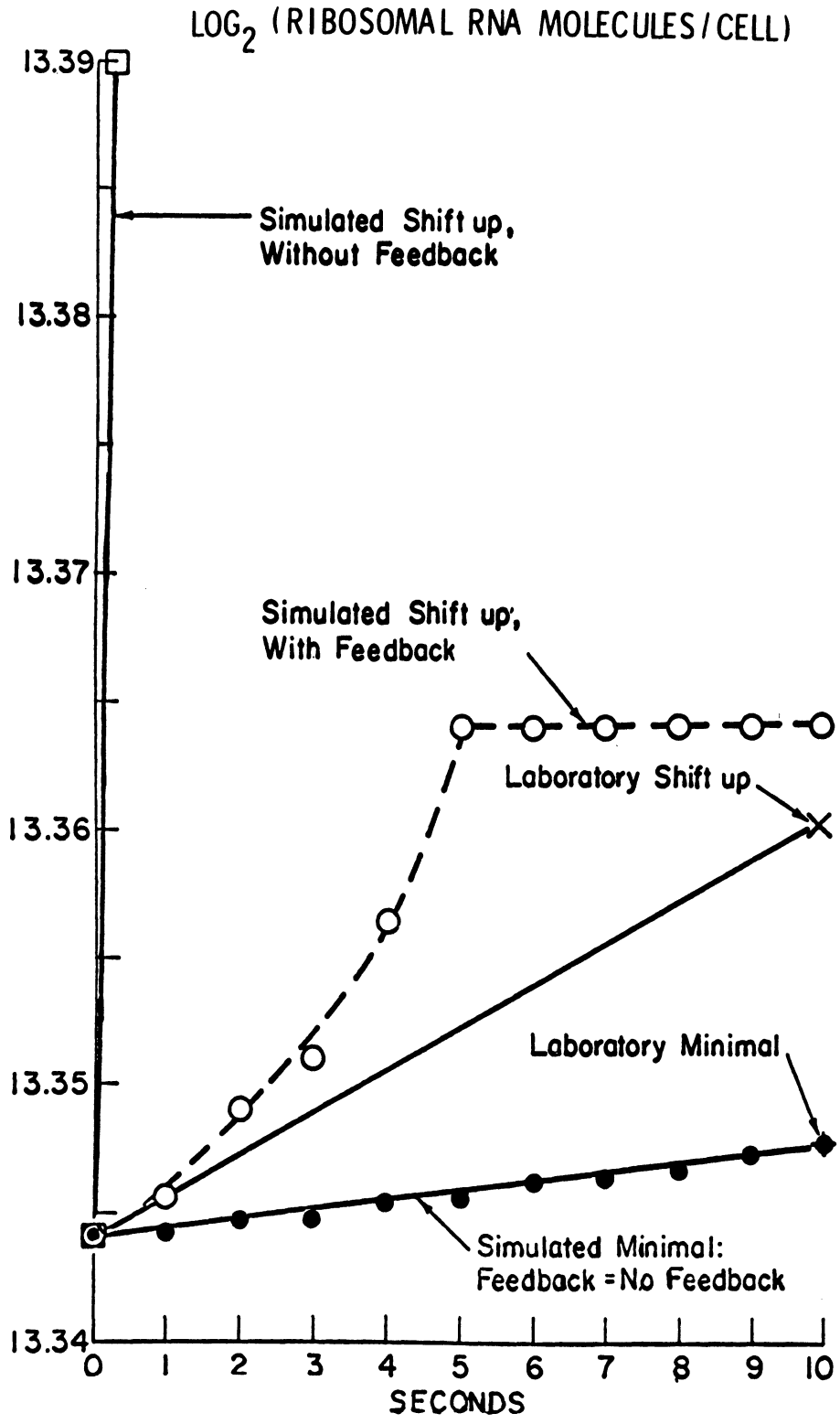


Figure 5.6 Ribosomal RNA During Shift Up. Well aerated liquid cultures at 37°C were used for laboratory and simulated data. Symbols: ● simulated minimal glucose liquid medium; □○ simulated shift from broth to minimal glucose at time zero; X laboratory shift up; + laboratory minimal glucose.

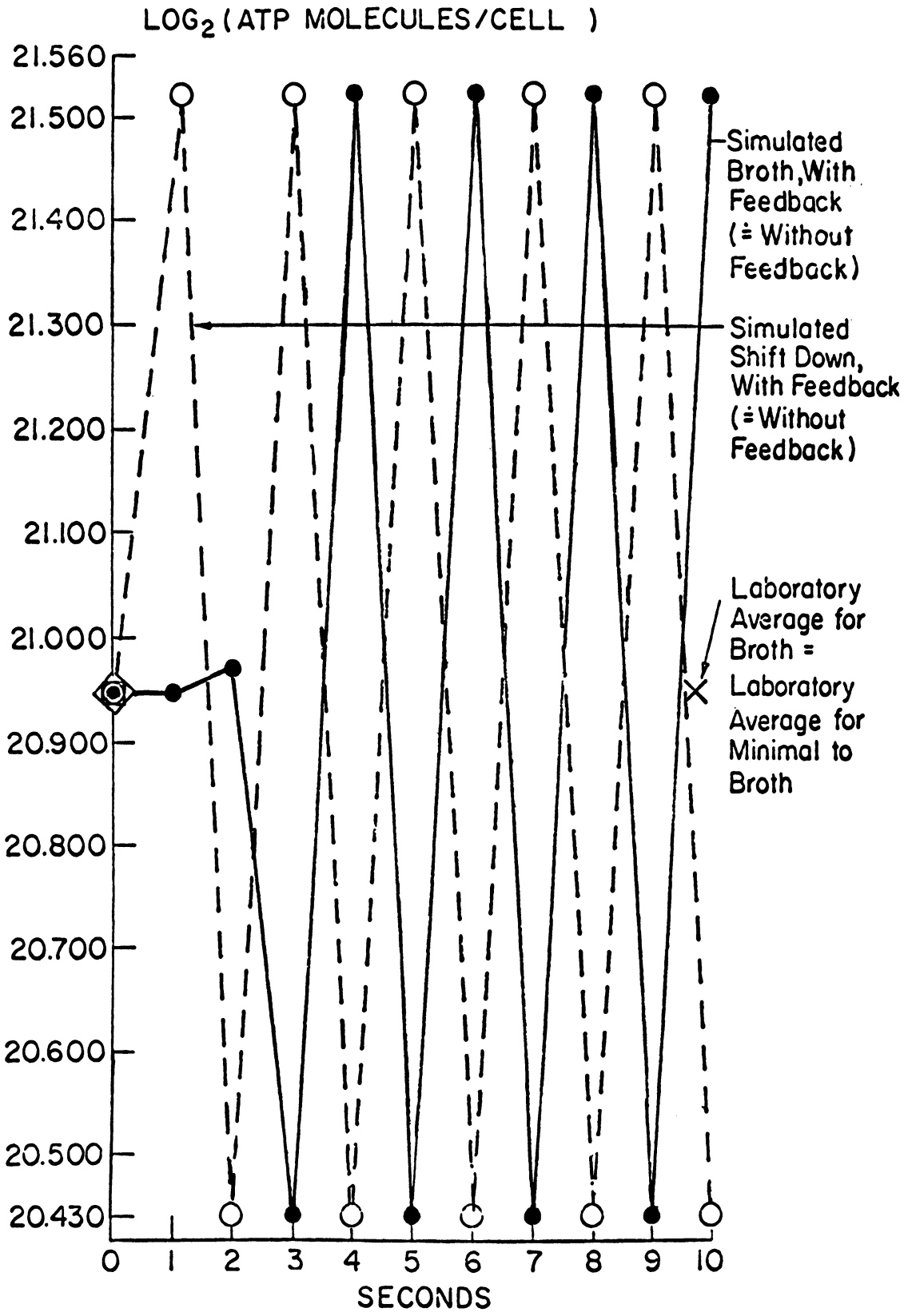


Figure 5.7 ATP During Simulated Shift Down.

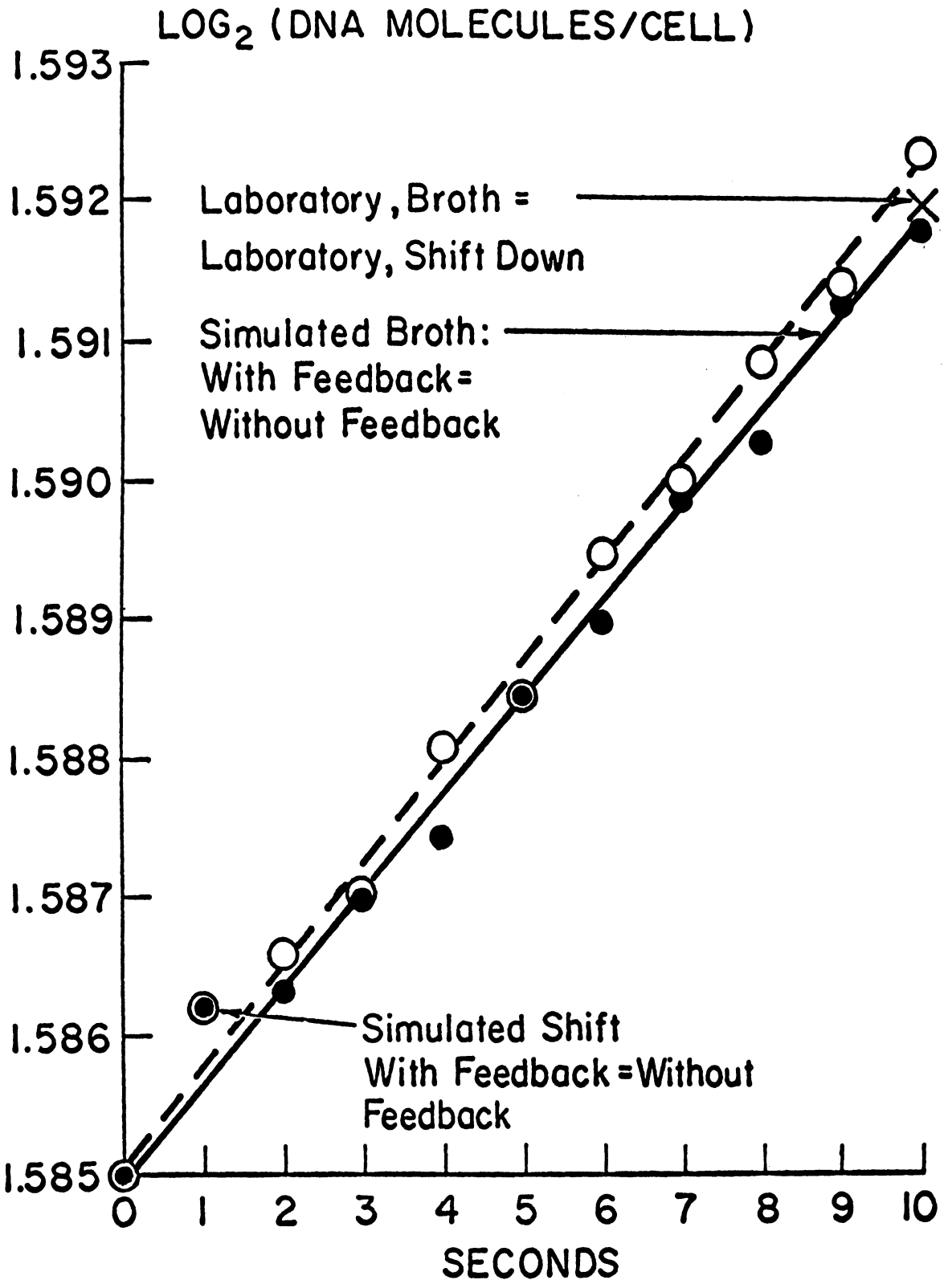


Figure 5.8 DNA During Shift Down. Well aerated liquid cultures at 37°C were used for laboratory and experimental data. Symbols: ○ simulated shift down from broth to minimal glucose at time zero; ● Broth control; X laboratory shift down = laboratory control.

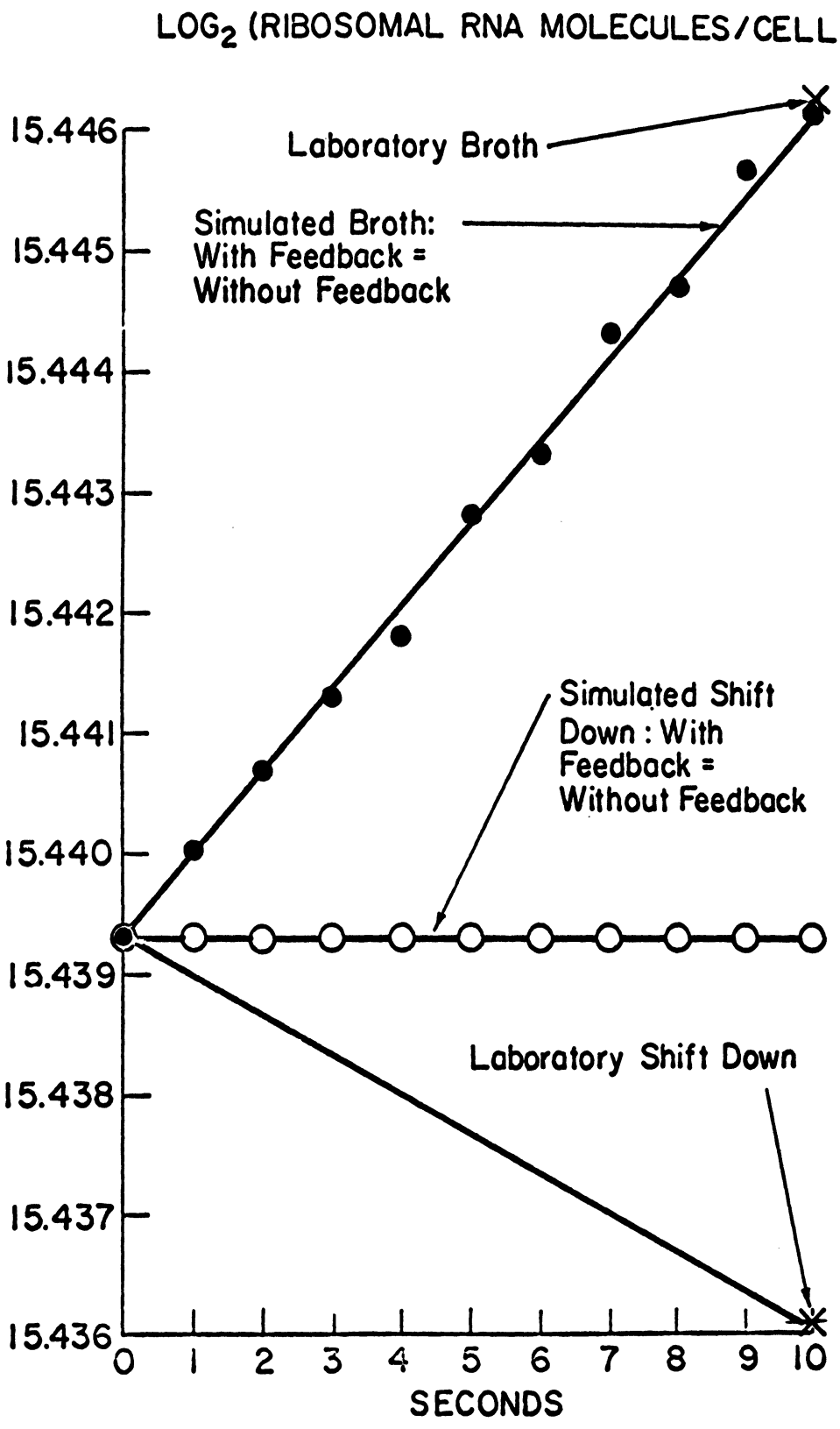


Figure 5.9 Ribosomal RNA during Shift Down from Broth to Minimal Glucose. Well aerated liquid cultures at 37°C were used for simulated and laboratory data. Symbols: ● nutrient broth; ○ shifted to minimal glucose from nutrient broth at time zero; ✱ laboratory shift down; X laboratory broth.

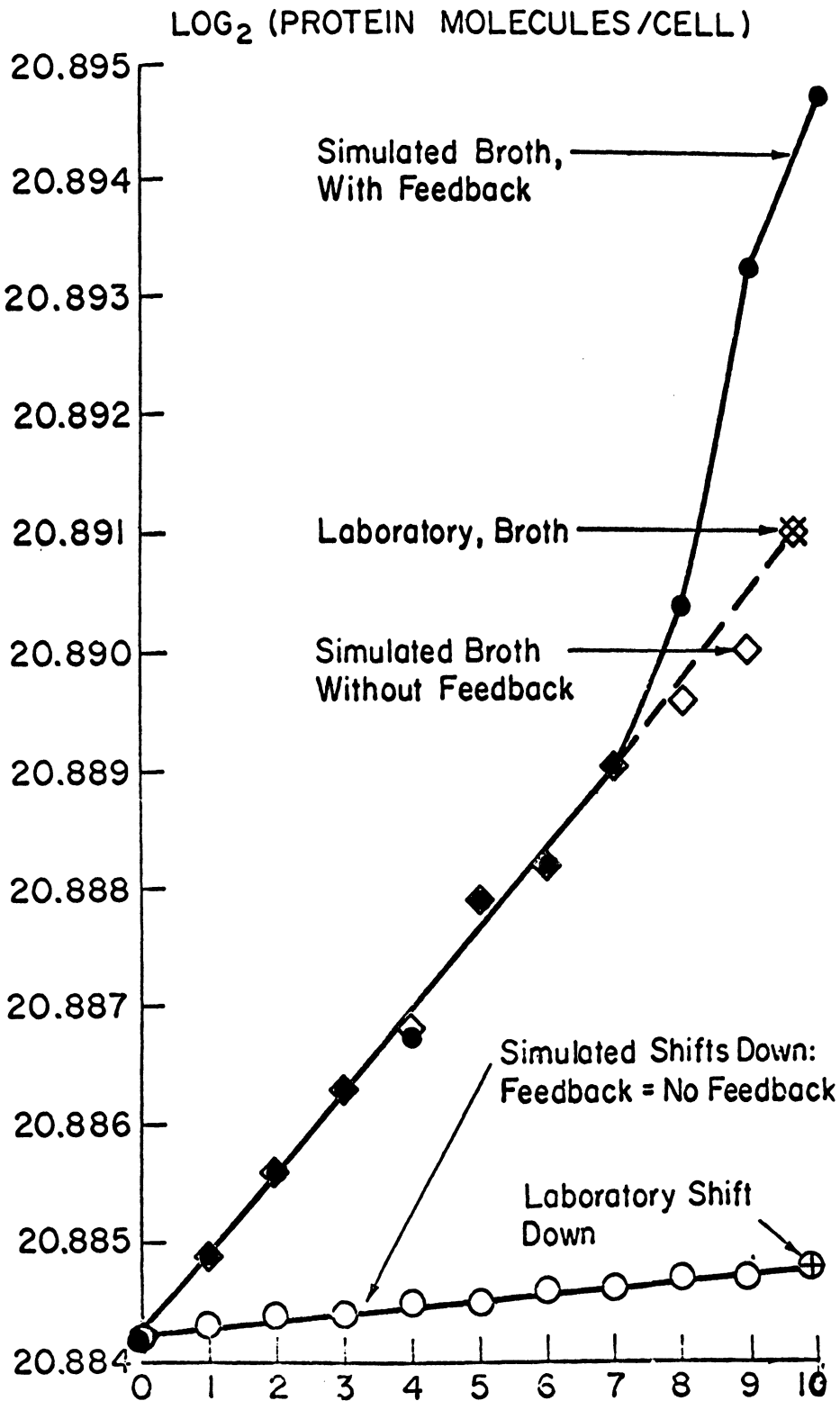


Figure 5.10 Protein/Cell During Shift Down. Well aerated liquid cultures at 37°C were used for laboratory and simulated data. Symbols: ○ simulated shift from broth to minimal glucose at time zero; ○◇ simulated broth; ⊕ laboratory shift down; X laboratory broth.

The Simulated Cell Grows Correctly in Three Environments: The environment in which the simulated cell grew was always a liquid growth medium at a temperature of 37 degrees Centigrade. Oxygen was non-limiting. We performed simulated experiments to see whether our simulation would produce the same results as laboratory experiments. The simulated cell was grown in three media: 1) minimal medium (medium containing glucose, ammonium salts, and minerals); 2) casamino-acid-glucose medium (medium containing additional amino acids) and 3) broth-glucose medium (a rich medium containing additional amino acids, nucleosides and vitamins). The simulated cell grew faster with each additional enrichment of its medium, in agreement with laboratory data (Weinberg and Berkus, 1969b). This is a reasonable result, since these additional enrichments provide biochemical intermediates and cofactors. As more biochemical intermediates and cofactors are provided, they need not be synthesized by the cell itself. With less materials to be synthesized, the cell can devote more of its materials and energies to faster growth.

The simulated cell produced chemicals and cell mass at a logarithmic rate, but duplicated in a stepwise fashion (Figure 5.1). This is just what the real cell does. Since the simulated cell produced these smooth growth curves from a complex interaction of many equations, the growth curves are a good preliminary confirmation of the model used to write the simulation.

The Simulated Cell Grows Correctly in a New Medium: We tested the ability of the simulated cell to grow in a medium it had never "seen" before by simulating a shift-down to low glucose concentration.

A minimal medium with low-glucose concentration was the new medium. The simulated cell decreased its growth rate in this shift-down, just as the real cell does (Figure 5.11 and Moser, 1958). This result shows that our simulation is general. It is not limited to the environments used to tune its parameters.

Allosteric Modification is Necessary for Shifts to Richer Media: The cell could adjust to shifts up from minimal medium to broth, and it could also adjust to shifts back down from broth to minimal medium. This orderly shift-up and shift-down behavior occurred when feedback controls were present. Two kinds of feedback controls were present in the simulated cell: allosteric modification of enzyme activity and repression of MRNA production. We shall only discuss the effects of feedback inhibition by allosteric modification of enzyme activity. Activation of enzyme activity by small molecules is also well known, and can be handled in our simulation. We did the following simulation experiment. We observed simulated-cell behavior for a ten second period following a shift. The rapid change in reaction rate effected by allosteric modification is important during this period. The slower effect of MRNA repression is not significant for such a short time period.

It is significant that without allosteric modification, the orderly shift-up from minimal medium to broth was not possible (Figures 5.2-5.6). The need for allosteric modification for a realistic shift-up suggested a role for allosteric modification in cellular metabolism as a whole. This role is to maintain metabolic stability in the face of rapidly shifting environments. The advantages of allosteric modifi-

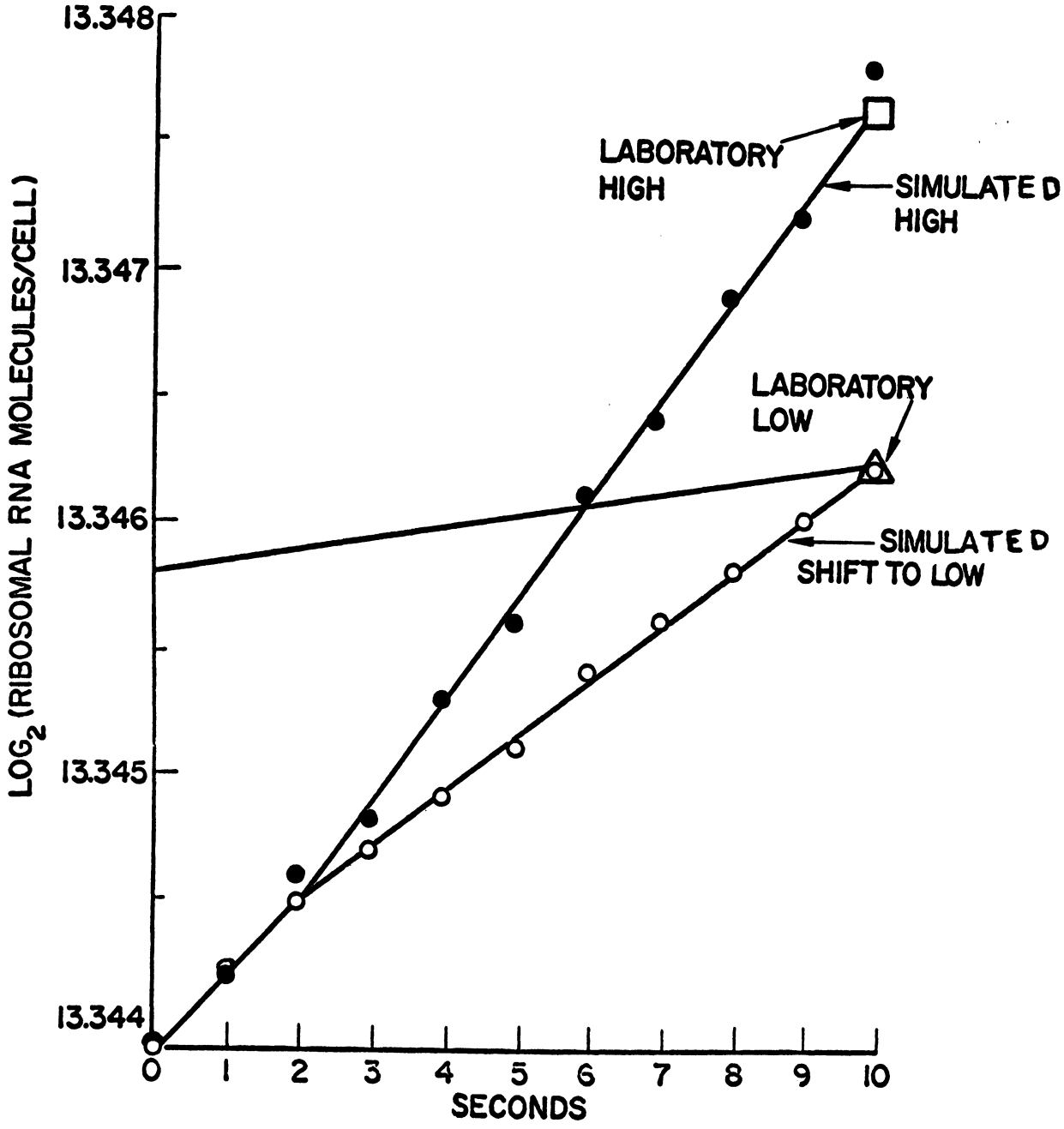
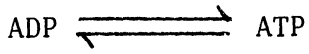


Figure 5.11 Simulated Growth in Low Glucose Concentration. Well aerated liquid cultures at 37°C were used for simulated and laboratory data. Symbols: all cultures were grown in minimal medium; □ laboratory, 4 mg. glucose per liter; △ laboratory, 4.10^{-4} mg. glucose per liter; ○ simulated shift from 4 mg. glucose per liter to 4.10^{-4} mg. glucose per liter at time zero; ● simulated, 4 mg. glucose per liter.

cation for control of individual pathways, and sets of related pathways, have been discussed by many other workers (e.g., in Frisch, 1961; Datta, 1969). Allosteric modification allows the cell to economize on the production of specific metabolic products. Allosteric modification turns off un-needed pathways. This saves both material and energy. Consider again the global effect of allosteric modification, stability during stress or, essentially, "saving the cell's life" during rapid environmental change (our result). Two cellular mechanisms allow the cell to maintain metabolic stability in the face of rapidly shifting environments: allosteric modification and metabolic topology. Let us digress for a moment to explain what we mean by metabolic topology. Metabolic topology is the metabolic map. It is a representation of the network of pathways through which materials flow in metabolism. The metabolic map imposes certain kinds of stability on the cell. An example of a small portion of the metabolic map is the representation of interconversion of ADP and ATP. This is drawn as:



We see, from the diagram, that formation of ATP uses up ADP. Using this information alone, we can predict that the more ATP that is formed, the slower will be its rate of production. This prediction is based on the metabolic topology, using only laws of mass action. It predicts a feedback effect. High concentrations of ATP limit formation of more ATP. This is a mechanism for cellular stability that exists in addition to more complex mechanisms such as allosteric modification. The metabolic map is extremely complex, and contains many relationships of the sort just described. The simulation experiments already discussed were

designed to investigate the role of allosteric modification, as separate from the role of metabolic topology, in maintaining metabolic stability. These simulation experiments were done using cells with the same metabolic topology (operative biochemical-pathways) in all cases. The cells differed only in the presence or absence of allosteric modification. The simulation experiments were performed as follows: The simulated cell was shifted up from minimal-glucose medium to broth medium (a shift-up). Growth was observed for ten seconds following the shift. The results of the shift were plotted. The same experiment was performed with allosteric modification, and then repeated without allosteric modification. Along with these shift-up experiments, analogous experiments were performed during a shift-down from broth to mineral-glucose medium, during growth in broth and during growth in mineral-glucose medium. The results of these experiments are shown in Figures 5.2 to 5.10. Each separate figure contains data from a shift, and from growth in original medium without a shift. On each figure, the data are shown for cells with feedback controls, and also for cells without feedback controls. Each figure contains data from one pool in the simulated cell. For example, Figure 5.2 shows the concentrations of ATP. At time 0, the cell was shifted from minimal glucose to broth (shift-up), and adjustment to the new growth condition is seen to be essentially complete within a few seconds. Time steps of one second were used, but time steps of 0.5 seconds and 0.1 second gave simular results. This gave us confidence that our results were a function of our model, rather than an artifact introduced at the level of the time step used in the

computer program.

Before describing our data in earnest, let us illustrate how simulation data was compared to laboratory data. Figure 5.4 on DNA concentrations is fairly simple, and will serve as our example figure. The X represents laboratory data. A shift-up does not alter DNA production for the first 10 second in laboratory cells. Therefore, there is only one X in the figure. It represents both the concentration of DNA in the laboratory cell 10 seconds after a shift-up and the concentration of DNA in a laboratory cell grown in minimal medium for 10 seconds. All concentrations are set at 1.0000 at the beginning of the simulation experiment, i.e., at time 0. The line with large, open circles shows the concentration of DNA in a simulated shift-up experiment in which feedback controls were operating. We can see that the 10 second simulated reading matches the 10 second laboratory reading for the shift-up experiment. Other comparisons are made in an analogous way. The squares represent a simulated shift-up without feedback. Obviously, this curve does not fit the corresponding laboratory data.

Let us return to our data analysis. For simulated cells with feedback controls, the data from the simulated cell agreed well with laboratory data (Figures 5.2-5.10). For the ten second time period studied, allosteric modification was the only feedback control of importance. Without feedback controls, the simulated cell was unable to realistically handle shifts up from minimal medium to broth (Figures 5.2-5.6). The major synthetic pathways of a real cell are represented in our simulation. We therefore extrapolate from our re-

sults with the simulated cell to conclude that allosteric modification serves to handle shifts up in the real cell. (This conclusion must be modified to account for situations such as degradative pathways, which are not represented in the simulation. We must also allow as exceptions portions of the synthetic pathways which are activated, rather than inhibited, by allosteric modification, but which do not show up in our simulation because of our lumping process.)

Simulated results often gave apparently counter-intuitive data. Such data became reasonable when analyzed, and often made apparent relationships which would have been otherwise easy to overlook. Data on DNA production by the simulated cell is a case in point. Part of the data is immediately reasonable, and part of the data seems startling at first. In a simulated shift-up with allosteric modification, there was a lag during which DNA synthesis continued at its preshift rate (Figure 5.4). This seems reasonable, and is in accord with experimental results (Maaloe and Kjeldgaard, 1966). Feedback controls simply maintained the original rate of DNA production during the shift-up. Surprisingly, though, DNA production *decreased* during a shift *up* without allosteric modification. This result says that in richer medium, DNA production falls off. At first, it seems that DNA production should increase if the food supplied to the cell is greater. But we saw that removing allosteric modification led to a decrease in DNA production during a shift-up. Why? The answer consists of a sequence of relationships, and events, which occur during a shift-up without allosteric modification: 1) The nucleotide pool is the precursor of DNA. 2) Depletion of the nucleotide pool will lower the

amount of nucleotides available for DNA synthesis. 3) The nucleotide pool will be depleted if too much RNA is produced, since RNA also uses nucleotides as precursors. 4) RNA is over-produced if allosteric modification is not operating during a shift-up from poor to rich medium (Figure 5.6). 5) RNA over-production (during a shift-up without allosteric modification) depletes the nucleotide pool, and leads to under-production of DNA. In other words, when allosteric modification is not present during a shift-up, the nucleotide pool is used up for the production of various RNA pools; messenger RNA, transfer RNA, and ribosomal RNA. There is less nucleotide left for DNA production, so DNA production decreases.

After a little reflection, the counter-intuitive result makes perfectly good sense. The counter-intuitive result even serves to confirm our model. The model predicted an unexpected, yet valid result. This result, by forcing us to think along unaccustomed paths, increased our understanding of complex relationships which exist in the metabolism of the entire cell.

The DNA results reflect the effect of kinetic equations in our model, rather than the effect of the replicon controls of DNA synthesis (replicon controls will be discussed in Chapter VII). For the present experiments, DNA synthesis is assumed to be continuous during the 10 second period under consideration. We are only seeing the results of the kinetic equations, because we only follow the simulated cell for 10 seconds after a shift-up. This time interval is long enough to observe the effect of allosteric modification superimposed on kinetic equations, but it is too short to study the longer term effects of messenger RNA repression, or of the replicon equations.

Oscillating Concentrations Reflect Metabolic Stability: The

simulated cell, with allosteric modification, maintained stability through feedback controls, namely allosteric modification itself. These conditions resulted in rapid oscillation of concentrations about equilibrium points, a phenomenon well known in the literature on feedback control (Hess, 1968). Oscillation of concentration during maintenance of equilibrium was strikingly illustrated by ATP and ADP concentrations during a shift-up from minimal medium to broth (Figure 5.2, 5.3). The "restoring force" effected by the feedback equations enabled the cell to maintain equilibrium concentrations of ATP and ADP. In contrast, the concentration of ATP was too high, and the concentration of ADP was too low, after a simulated shift-up without feedback controls.

Similarly, an overshoot in ribosomal RNA concentration, produced during a shift-up, was quickly corrected by a simulated cell with feedback controls (Figure 5.6). A similar shift-up experiment without feedback controls gave quite a different result (Figure 5.6). A simulated cell which was shifted up without feedback controls produced a great excess of ribosomal RNA. If a real cell had contained this much RNA, it may well have lysed (barring secretion or other complicating phenomena). In any case, the excess is not produced by real cells during a shift-up. We have reaffirmed our original hypothesis: Allosteric modification is necessary to maintain realistic metabolic stability during a shift-up from a poor medium to a rich medium.

Our major result was that allosteric modification is necessary for shifts from poor to rich media. We also observed oscillations in ATP and ADP concentrations. For our time step of 1 second, we were not able to determine the period or the magnitude of these

oscillations. We were only able to say that the oscillations did not lead to a prolonged increase or decrease in concentration of any pool, so long as feedback controls were used. Since an artificial bound on concentration was used during ADP and ATP oscillation, we feel obligated to explain how the bound was used. We will use ATP, and the upper bound, in our example. The same discussion applies to ADP oscillation, and the lower bound for both ATP and ADP. Proceeding with our example, if ATP went above a given maximum bound, it was set back to that maximum bound. The question we ask is: does the *upper bound* (arbitrarily set) maintain ATP within a certain range, or are the *equations in our model* maintaining ATP within a certain range? The answer is that as long as ATP concentration oscillates at each time step, the equations are maintaining the concentration between bounds. By oscillate we mean specifically that when ATP concentration reaches the upper bound in one time step, it goes back down in the next time step (and when ATP concentration reaches the lower bound in one time step, it goes back up in the next time step). Why can we conclude that oscillation implies the equations of the model are maintaining the ATP concentration between bounds?

During oscillation in the simulated cell, ATP concentration fluctuates between its upper and lower bounds. If ATP is at its upper bound, it will go to its lower bound in the next time step. (Figure 5.2). We know that the model equations are preventing ATP from passing the upper bound. Whenever we set the ATP concentration to its upper bound, in the simulation, the ATP concentration decreased. The differential equations in our model represent a continuous system. If these model

equations predict a fall in ATP concentration at the upper bound, than the ATP concentration in the model can not pass beyond the upper bound. What about the simulation? With a discrete time step, the simulated ATP concentration does pass through the upper bound. But this is because we assume that the rate of ATP change throughout the time step is the same as the rate of ATP change at the beginning of the time step. Using our bounds, we make the course adjustment of not allowing the ATP concentration to rise above the upper bound. We see that in the simulation, during ATP oscillation, if ATP reaches the upper bound, the ATP concentration will always fall in the next time step. This is what we describe when we say oscillating ATP concentration. Our point is that the equations in the model cell underlying the simulation are self correcting if allosteric modification is operating. We know this, because under these conditions the ATP concentration always decreased when it was set to the upper bound in the simulation. Similar reasoning applies to the lower bound. Since in the case that the simulated cell was using allosteric modification, ATP was always going down at its upper bound, (and up at its lower bounds), we could conclude that the equations in the model, rather than the upper and lower bounds, were functioning to keep ATP concentrations within bounds. Moreover, in the simulation cases where ATP hit the lower bound and remained there, the simulated cell was shifting up without allosteric modification. Our conclusion was that ATP was out of bounds. The lower bound restriction of our simulation was keeping ATP higher than the model equations, but ATP

was still too low. Thus, our conclusion is now even stronger than before. We can state that ATP concentration was much too low during a shift-up without allosteric modification, and that it would have been even lower if we had not set a lower bound through which the concentration could not pass. In any case, our major conclusion is still that allosteric modification is necessary for maintenance of metabolic stability during a shift-up. This conclusion is not invalidated in any way by our use of upper and lower bounds to restrict our ATP and ADP concentrations.

Discussion of Results: It is reasonable that allosteric modification of enzymes would be useful in shifts from poor to rich media, but not vice versa, although this insight was obtained from the striking results of the simulation, rather than being immediately apparent from experiments on real cells. In poor medium, the cell has a plentitude of enzymes. It is synthesizing all of its own biochemical necessities, and therefore no repression of synthetic enzymes is in effect. Thus, rapid over-production occurs when a cell with fully operating metabolic machinery is shifted to a rich medium where substrates, and endproducts, are supplied. Turning off many of the enzymes in a cell during such a shift saves the cell's life. On the other hand, a cell in rich medium has repressed the production of most of its synthetic enzymes at the DNA level. Since few enzymes are present, turning off enzyme catalysis by allosteric modification has little effect. Therefore, the shift down from rich to poor media can be accomplished without allosteric modification.

We will rephrase, in evolutionary terms, our conclusion that allosteric modification is important for shift-up conditions. We conclude that allosteric modification in the real cell was evolved to handle shift-up situations. The major metabolic-pathways, without allosteric modification, do not produce a cell which is stable during a shift-up. By not stable, we mean that concentrations of pools of chemicals in the cell become very high, or very low. It seems unlikely that a cell could survive with such extreme concentrations. A cell without allosteric modification would be at a selective disadvantage when competing with a cell with allosteric modification. Therefore, we would expect allosteric modification to be strongly favored during periods when the environment was changing. We will discuss the evolutionary aspect of our simulation further in Chapter VI. We may reasonably postulate that allosteric modification evolved in cells which had to shift suddenly from poor to rich environments. It is known that violent environmental-fluctuations occur frequently during evolution of life on earth (Wallace, 1968). This lends further credence to our conclusion that allosteric modification arose during evolution in response to fluctuating environments. With respect to the effect of violent environmental fluctuations on the organism, we may use the organism itself as evidence for environmental fluctuation. Many genetic mechanisms support the idea that environmental fluctuation is a strong force during evolution. Genetic mechanisms, such as directed mutation and diploidy, exist so that the organism is genetically flexible. This flexibility allows the race to adjust to an environmental fluctuation (Chapter VI). Allosteric modification is a physio-

logical mechanism paralleling these genetic mechanisms.

Our conclusions are probably correct for major synthetic pathways. The synthetic pathways are the pathways which we represent in our model. We chose these pathways since they predominate in a bacterial cell growing in the steady state, and our simulation is run mainly in steady-state growth-conditions. Our shift experiments consist of shifts from one steady-state to another steady-state. There certainly exist pathways which operate in a direction contrary to the one we have discussed. Despite these other pathways, our conclusion as to the importance of allosteric modification during environmental shifts still holds. Although these other pathways speed *up* their metabolic rate during a shift-*up*, they still use allosteric modification. It is doubtful that these exceptional pathways, without allosteric modification, could maintain metabolic stability in shifting environments. Since even these exceptional pathways use allosteric modification, the role of allosteric modification is more, not less, important because of their existence. In addition to speeding up their metabolic rate during a shift-up, these exceptional pathways may slow *down* their metabolic rate during a shift *down*. Examples of such pathways are degradative pathways, and small parts of synthetic pathways. Many pools have to be broken down at a more rapid rate under more rapid growth conditions, and these are the pathways which operate in the aforementioned exceptional way. Our model represents major synthetic pathways, and does not contain every molecule in these pathways. We can still make the same general conclusion: For the major synthetic-pathways of metabolism, allosteric modification is essential during shifts-up

from poor environments to rich environments.

Conclusions from Results: In summary, our simulated cell has been validated insofar as it was able to realistically shift environments while maintaining metabolic stability, and to grow realistically in a new environment. Studies performed with variation of simulation parameters enable us to conclude that:

1) Shifts from poor to rich medium are more of a challenge to the cell than shifts from rich to poor medium. The shift up to rich medium requires elaborate feedback control mechanisms, whereas the shift down to poor medium does not require feedback control as strongly. The shift-down can be handled by metabolic topology.

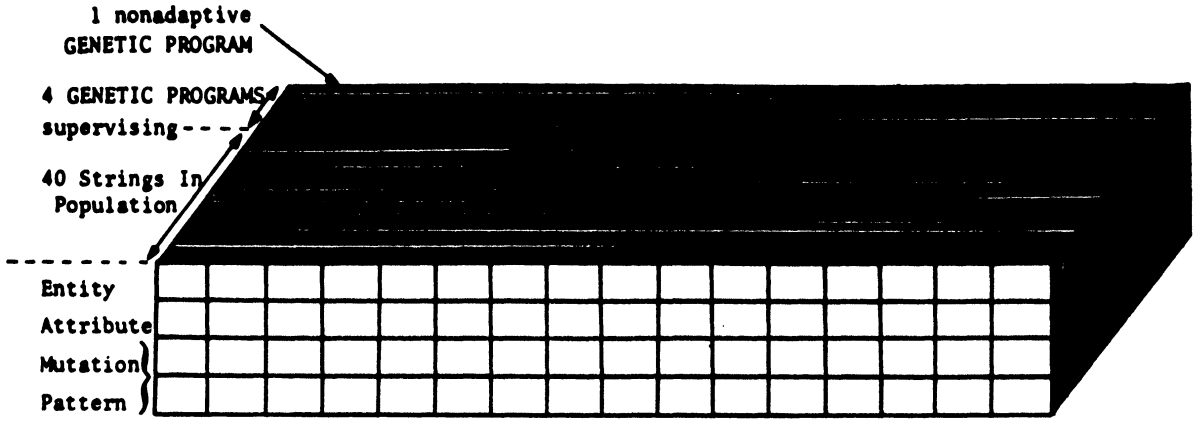
2) Oscillations occur about equilibrium concentrations; fixed equilibrium concentrations are not maintained in the simulation. However, this agrees with experimental observations which indicate that real cells are constantly oscillating biochemical systems. There oscillations suggest, indeed, that the oscillating concentrations produced by feedback-control systems are necessary for the flexibility characteristic of living systems.

CHAPTER VI

COMPUTER SIMULATION OF EVOLVING DNA

The computer simulation of a living cell adapts phenotypically to three different chemical environments (Chapter V). We will extend the simulation so that the cell can adapt genetically, as well as phenotypically. We will represent DNA as an array in the computer. In this array are stored indices and values of various rate-constants. The rate-constants appear in the equations representing the simulated cell. Each rate-constant is stored as two quantities: 1) an index number tells which rate-constant we are referring to; 2) the value of the rate-constant which is indexed is stored also (Figure 6.1). The genetic map for storing indices and values is separate from the operating program which realizes the expression of this information. It is the operating program which simulates the phenotype of a cell. The genetic map represents the genotype of a cell. It turns out that separation of genotype from phenotype is vital in order to utilize non-linear interactions among different genes. This non-linear interaction is used by genetic operators to attain evolutionary gains. Genetic operators capable of taking advantage of such non-linear interaction will be basic to the computer simulation of evolving DNA.

Genetic Operators: Four powerful genetic operators for evolution of populations are 1) crossing-over, 2) inversion, 3) mutation, and 4) dominance. 1) *Crossing-over* is the interchange of segments between two chromosomes. It permits preferential multiplication of groups of sub-routines which interact well. (Let us digress for a moment to con-



15 Entities Referenced by 1 String = 1 Individual

Column Number = Position of Entity Reference on String
= 2nd Dimension of Array
$1 \leq$ Column Number ≤ 15

Row Number = 1st Dimension of Array. $1 \leq$ Row Number ≤ 4

Row Number :1 = Index of Entity Referenced = Locus Referenced

2 = Attribute of Entity Referenced = Value of Locus

3 and 4 Describe Mutation Pattern for Entity Referenced in Row 1

3 = Number Controlling Interval over Which Random Number is Generated for Mutation

4 = Index of Probability Distribution over Increments of Mutation for Monte Carlo Method.

INDIVIDUAL IN POPULATION = STRING IN POPULATION IS INDICATED BY THE THIRD DIMENSION OF THE ARRAY. THE CONTENTS OF THE ROWS AND COLUMNS CONTAIN INFORMATION ABOUT ATTRIBUTES AND GENETICS OF THE INDIVIDUAL INDEXED BY THE THIRD SUBSCRIPT.
e.g., CHROM (1,4,2) = 3 MEANS THAT THE 3^{rd} ATTRIBUTE OF THE 2^{nd} INDIVIDUAL IS INDEXED BY THE 4^{th} COLUMN OF CHROM ARRAY INDICATED. $1 \leq$ INDIVIDUAL INDEXED ≤ 40.

THE LAST FOUR STRINGS INDEXED CONTAIN INFORMATION ABOUT THE GENETIC PROGRAM SUPERVISING EVOLUTION. $41 \leq$ GENETIC PROGRAM ≤ 44.

Figure 6.1 The Chrom Array.

sider these interacting groups of sub-routines. Evolution of groups of genes which function as a system is quite common in real populations. The process is called co-adaptation. In order to implement co-adaptation, the interacting system of genes often becomes closely linked on a linkage map. Genes which are closely linked tend to remain together in an evolving population. If such closely linked genes work well together, then the complex confers a strong selective advantage upon the lucky organism possessing the felicitous combination.) Returning to our explanation of crossing-over, crossing-over essentially sets up the concept of genetic distance. Without genetic distance, there could not be closely-linked genes versus weakly-linked genes. Crossing-over is basic for the system an evolving population uses to take advantage of non-linear interactions among systems of genes. 2) *Inversion* is a reversal in sequence of part of the chromosome. It is necessary to rearrange the genetic locations of different subroutines. Functionally related genes which work well together need a chance to get close to each other on the genetic map. A closely linked system of genes will tend to remain together (as was discussed under crossing-over) favoring co-adaptation. 3) *Mutation* is a change in an existing gene. It is necessary in order to explore a large genetic space. Mutation also regenerates attributes of functions when those attributes are lost through selection. 4) *Dominance* is masking of a recessive gene by its dominant mate. It is often used to preserve genes which are at a temporary disadvantage. These temporarily disadvantaged genes may become useful at some later time in evolution. We will not use dominance in our simulation.

Populations: Duplication of individuals, as well as the genetic operators discussed, will be simulated. We will accomplish all of these ends by operations on the contents of our arrays. Recall that we use arrays to store information which real organisms store on DNA molecules. We will modify the arrays in various ways in order to simulate the effects of genetic operators on populations of cells. We will be simulating haploid bacterial populations. Haploid organisms do not make extensive use of dominance to store variability. These haploid populations use other means to store variabilities. They are still quite successful in the struggle for survival, e.g., many haploid populations of bacteria exist in nature. Furthermore, haploid organisms often displace diploid competitors from an environmental niche. Haploid bacteria crowd out their diploid protozoan competitors in beef stew left out in a warm climate. Indeed, haploid bacteria and algae replace diploid fish in a stream, if food for bacterial and algal growth becomes available in the stream. We will "remember" favorable genes in our simulation by strongly structuring our mutation pattern. A particular mutation pattern will represent alleles which are easy to obtain. Since the mutation patterns are under selection, we may obtain mutation patterns which "save" temporarily disadvantaged genes which are likely to be useful at a later time. This is what we meant by "remember" favorable genes. We still lose the ability to save valuable, temporarily disadvantaged substrings when we discard diploidy. These linked, co-adapted sets of successful alleles can be saved in our simulation only when they are at a selective advantage.

Directed mutation is particularly easy to simulate with Monte Carlo techniques. In these techniques, one can easily manipulate the proba-

bility of obtaining any particular number. One makes appropriate choice of a random-number interval and cumulative probability-density (this will be discussed later in this chapter). We implicitly define the recessives stored by a string as its high-probability mutants. The whole population of potential mutants changes when an attribute changes. This change in potential mutants partially simulates a change in dominance. This correspondence is so indirect, however, that we would consider it an experimental part of the program. Since the random number interval and frequency distribution referenced will also be subject to mutation and selection along with the rest of the string, the population has a chance to evolve an efficient replacement for natural dominance.

In addition to functioning in the same way as dominance, directed mutation also saves time and storage in a computer simulation: 1) Much unnecessary mutation is ruled out. The probability structure of the mutation space serves to rule out many mutations as highly unlikely; 2) Recessive alleles do not have to be stored; and 3) Complex calculations for dominant alleles in the canonical realization of the string are eliminated. The directed mutation procedure will be outlined later for the computer program.

Efficiency of programming also motivated our representation of a population of strings. There is only one representation of any particular string in the program. The utility of a string is increased if it is supposed to represent a large number of individuals. This is used, instead of storing the same information in a large number of strings, to represent multiple copies of a string. By allowing each array to represent a different string, we are preserving maximum var-

iability with minimum computation and storage. Since the rate-in-change of fitness-of-the-population is approximated by the variability (by Fisher's Fundamental Theorem), we would like to preserve maximum variability in the population.

There might also be a population of the best unused string from each population; individuals in this population would be saved, but would be used only for recombination and directed mutation. This permits preservation of temporarily disadvantaged strings of genes, and non-random mutation for those alleles which are likely to be useful. This realizes the function of dominance without lengthy computation. Lack-of-fixation of fit individuals may result from this high variability, but the maintenance of variability will also enable the evolving strings to try out more different combinations of genes. We remedy the lack-of-fixation of fit individuals in two ways: 1) old strings are preserved each generation as members of the next generation (40% of the old population will be saved intact in an example we will use) and 2) new population members are obtained through directed mutation (directed mutation is rigged to produce useful alleles). The population is therefore unlikely to "forget" a good set of parameters once they are obtained.

Holland's Formal Adaptive System: John Holland has formulated a general theory of adaptive systems (Holland, 1969). Our genetic simulation is a specific example of Holland's general adaptive system. An excellent feature of a general scheme like Holland's is its extreme flexibility. One can consider part of an organism as the string which forms an individual in the population. The rest of the organism may be treated

as part of the environment. Since the theoretical development is much easier for a stationary environment, I will consider all loci which are fixed during the whole evolution of the programs as the environment. Only unfixed loci will be treated as evolving strings. Since we are free to set linkage parameters as we wish, we can very the amount of linkage between genes. We will increase linkage to account for spaces on the chromosome which are occupied by fixed loci which do not explicitly appear in the simulation. Most of the genetic characteristics of the individuals, such as mutation rate and crossover, will be represented as separate strings of adaptive or non-adaptive genetic programs. Each individual genetic program will supervise the evolution of a population of strings.

Selection: Survival of the fittest strings will govern the evolution of our populations of strings. It is also important to have adaptive genetic-programs which can evolve. Bad genetic programs may be written, and they should be modified as the programs run. An example of a bad genetic technique in the real world will illustrate this point. Chicken farmers wanted long-legged chickens. The chicken-farm selective procedure led to an unexpected consequence. Long-legged chickens died. (Wallace, p. 455).

We can easily select the best strings in a population. There are two properties of the simulated cells which indicate phenotypic limitations for the cell, and are particularly easy to observe: 1) a wide disparity between simulated chemical concentrations of cell metabolites and the concentrations necessary for balanced growth, 2) the inability to modify simulated enzymes to produce proper growth-rates in the three

simulated environments used. Each of these properties is correlated to a program variable, and each program variable is used to calculate the performance of a string. 1) In regard to the first property we mentioned (a wide disparity between chemical concentrations) we can see the inability to maintain biochemical equilibia necessary for life in a disparity between the concentrations of biochemicals in the running simulation, and optimal concentrations.

The ratios of simulated biochemical pools to optimal biochemical pools should be 1 if metabolic equilibrium is perfectly maintained. Departures of the ratios from 1 indicate metabolic instability. The further the ratio departs from 1, the lower should be the value of the utility function. The utility function decreases as the ratio departs from 1 because we have included the term (ratio + 1/ratio) in the denominator of the utility function. Recall that the lower is the utility function, the less is the rate of reproduction of the individual under consideration. Therefore, by the previous calculation, we select against metabolic instability. 2) We will now discuss the second property of the simulated cells which indicates phenotypic limitations, the inability to manipulate enzymes to produce proper growth rates. This second phenotypic limitation shows up in the simulation as an inability to produce a solution in the solve routine of the simulation (Chapter 3). We penalize a program which is unable to produce a solution to the solve routine in the following way. The penalty consists of a decrease in the utility of the offending program. Inability to solve the equations for allosteric modification shows up as an inability to produce a solution to the solve routine. Such a failure adds a 10

to the denominator of the utility function. This lowers the utility of the program, and since utility relates to reproductive capacity, it has the effect of selecting against the program. Failure to produce a solution to solve indicates an inability to manipulate enzyme rates for proper growth in our simulated environments. Therefore the net effect of this process to select against an inability to manipulate enzyme rates.

We will now return to the subject of modifying genetic procedures during evolution of our populations of programs. In the following discussion, one should keep in mind that there are two kinds of genetic programs directing evolution: 1) *Non-adaptive* genetic-programs supervise the evolution of adaptive genetic-programs. Non-adaptive genetic-programs do not evolve. 2) *Adaptive* genetic-programs supervise the evolution of strings of simulated-cells. Adaptive genetic-programs do evolve. In order to allow evolution of adaptive genetic-programs, we need a measure of how well the adaptive genetic-program did its job. In other words, we need a measure of the utility of each adaptive genetic-program. The non-adaptive genetic-program will apportion offspring to each adaptive genetic-program in proportion to the adaptive genetic-program's utility (Figures 6.2, 6.3).

We calculate the utility of an adaptive genetic-program indirectly, by using a god-like judge, the non-adaptive genetic program. The non-adaptive genetic program calculates the utility of an adaptive genetic-program by judging the population which has evolved under the direction of the adaptive genetic-program. It calculates a utility for the best string in the adaptive genetic-program's population. This is

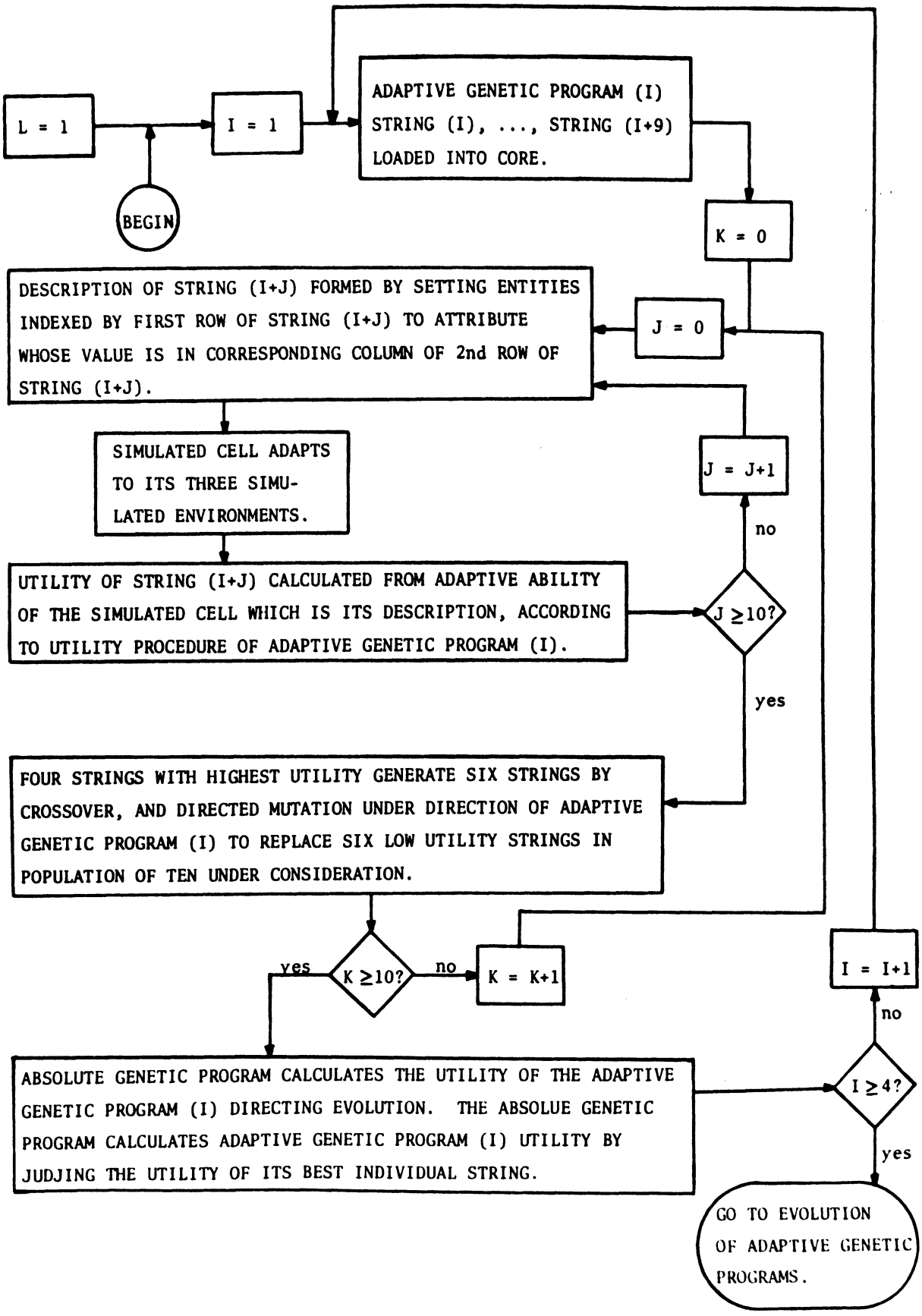


Figure 6.2 Evolution of Strings.

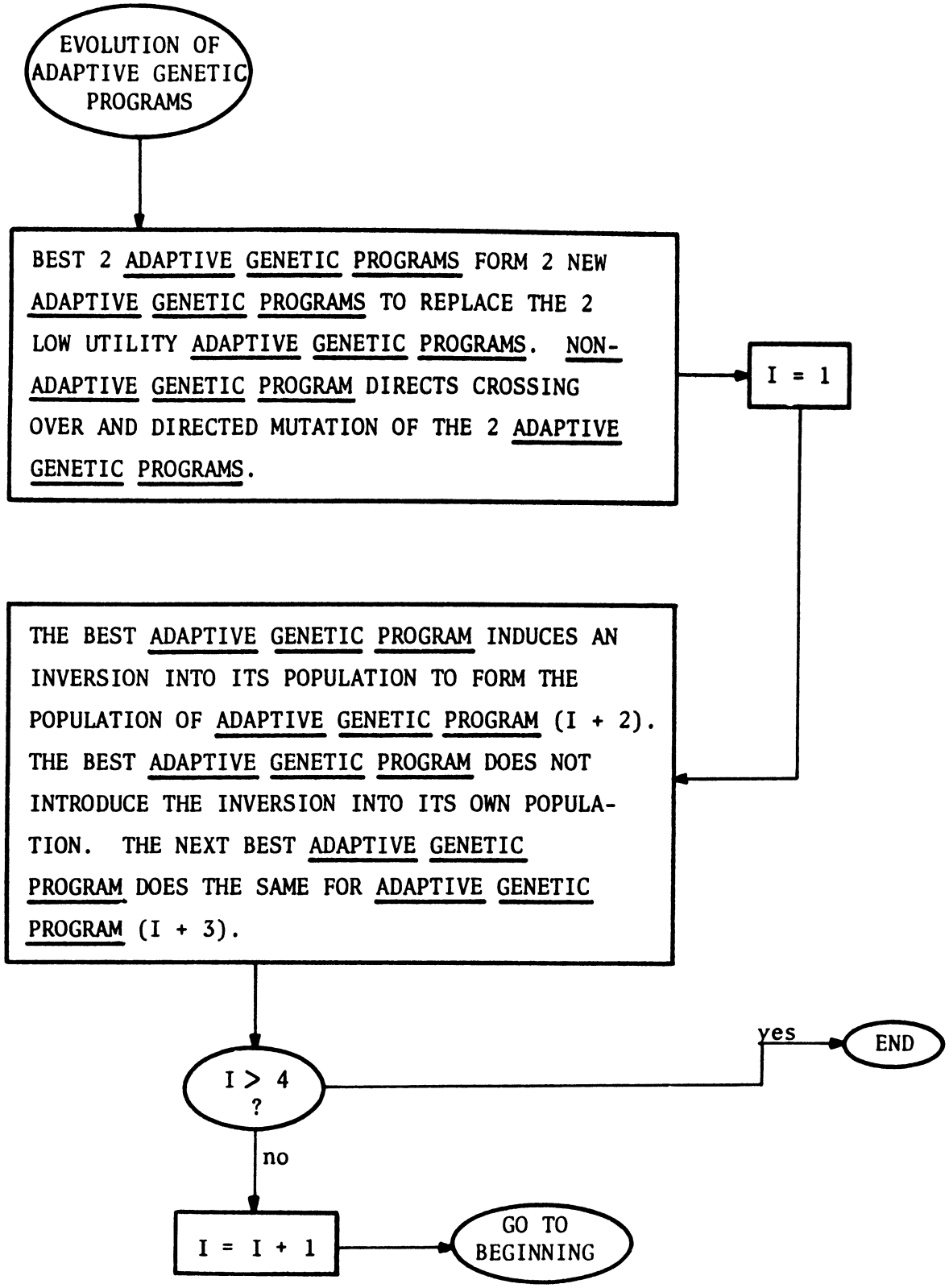


Figure 6.3 Evolution of Adaptive Genetic Programs.

the utility awarded to that adaptive genetic-program. After each adaptive genetic-program has been awarded a utility, the non-adaptive genetic-program directs the evolution of the population of adaptive genetic-programs. In modifying adaptive genetic-programs, elimination of unnecessary genetic operations used by the adaptive genetic-programs will probably occur. The faster a population evolves, the higher the utility awarded to the adaptive genetic-program directing that population's evolutionary progress. This is accomplished by 1) recording the time it took the adaptive genetic-program to produce its final population and then 2) including this time in the denominator of the utility awarded to that adaptive genetic-program. The net effect of rewarding speedy evolution will be to eliminate unnecessary genetic operations used by the adaptive genetic programs. Eliminating unnecessary genetic operations accomplishes two ends: 1) it is an advantage to the programmer because it saves machine time. 2) it is close to phenomena really present in natural populations of organisms. There are various kinds of genetic information which seem likely candidates for elimination. Much of the information contained in rows 3 and 4 of a string's CHROM column allows use of sophisticated probability distributions for mutation patterns (Figures 6.1, 6.4). We could ignore the cumulative frequency distribution indexed in the 4th row, and simply use the uniform distribution. (An adaptive genetic-program which does this will gain in utility by gaining in speed.) For example, FREQ(3) may be set to a uniform frequency distribution, and effectively ignored for the jth entity during mutation. This can be done, since the random number generator itself sets up a uniform distribution over a specified interval. If the loss in evolutionary power through eliminating complex probability-distributions does not over-balance

INDEX OF ENTITY	ENTITY EQUALS LOCUS	IOTA = RANGE FOR ATTRIBUTE OF ENTITY INDEXED	MUTATION PATTERN EQUALS INCREMENT IN ATTRIBUTE
1	P1V(1)	$(-10^{-6}, +10^{6})$	+Normal: mean P1V(1)/10 = variance
2	P1V(2)	" "	2
3	P1V(3)	" "	3
4	P1V(4)	" "	4
5	P1V(5)	" "	5
6	P3V(1)	" "	+Normal: mean P3V(1)/10 = variance
7	P3V(2)	" "	2
8	P3V(3)	" "	3
9	P3V(4)	" "	4
10	P3V(5)	" "	5
11	R(1)	(0, 1)	+Uniform: -1, 1. $R \geq 0$. If $R < 0$, set to 0.
12	R(2)		
13	R(3)		
14	R(4)		
15	Utility of individual referenced by 3rd dimension of array. Utility is calculated by the genetic program supervising evolution, so there are some empty spaces here.		

Figure 6.4 Description of Strings Referenced when Third Dimension of CHROM Array Ranges from 1 to 40.

the gain in utility by economizing on computer time, some of the information in the 3rd and 4th rows of the CHROM array may be dropped from the genetic programs eventually. Rows 3 and 4 were included in our original genetic programs to indicate the ease with which a general and powerful evolutionary program may be written. After translation of a string into an operating simulated cell, the whole growth and phenotypic adaptation procedure only takes 3 seconds of IBM 360 computer time, emphasizing again the ease with which complex cells may be set up for fast evolutionary procedures. Furthermore, the cost of programming may be lowered by storage of large blocks of the program on disks or in files until they are needed.

Thus far our formulas have been fairly simple. We may wonder whether simple genetic operations on finite populations of strings can capture any of the power and scope of natural evolution. Yes, they can. Inducing sophisticated phenomena by simple genetic operations is explained in Holland's considerations of schemata which exist in finite populations. A schema (plural schemata) is a partial description of a genotype. A schema defines the sub-population fitting the partial description specified by that schema. An example of a schema in human populations is blue eyes and blond hair. All people who have blue eyes and blond hair are in the schema. Schemata are useful in describing inter-related genes. Co-adapted genes can be described as schemata. When one relates simple genetic operations on strings to analagous operations on schemata, one gets some idea of the complex processes which occur in finite populations undergoing evolution. Holland presents a convincing argument that such populations have at their command extremely powerful tools for evolution.

Average excess is an example of a complex quantity which is hard to calculate, but is easy to induce by simple, finite genetic operations. Elimination of sixty percent of a population at each reproductive cycle is easy to accomplish in the simulation. The average excess induced by such an elimination may be extremely difficult to describe in quantitative terms. If one wanted to analyze in a quantitative way the phenomenon of average excess, one would find it much easier to do if his populations were greatly limited. For such a study, limiting one's population to schemata evolving in a "successful" population might prove extremely advantageous. Let us continue with our comparison between simple genetic operations on strings, and the complex selective phenomena induced on populations. We must distinguish between 1) simple criteria used on the computer programs themselves and 2) selection induced on schemata by such simple criteria. To illustrate this perhaps subtle, and certainly profound distinction, we will use as the simple criterion the elimination of sixty percent of the strings (6 out of 10 in the actual calculation) at each reproductive cycle. Average excess will illustrate the complex quantity which is induced by the simple criterion. After eliminating 60% of the old strings, we will generate the missing sixty percent from the surviving old strings. To produce the new strings, the genetic operators will be used. These will be crossing-over, inversion, and mutation. A string's utility will be proportional to how well and how quickly the simulated cell (which is the description of the string) adjusts to changes in simulated environments. The environment for the evolving strings is all fixed loci represented by the equations simulating cell growth and adaptation, as well as the changing simulated environment for the cell. Let us return to our consideration of a string's utility.

Utility is a number which judges how well a string performed in various simulated environments. Performance of a string is directly related to the string's ability to produce a metabolically stable cell. Utility is not as sophisticated as many other measures of performance, such as average excess for a string. The whole utility function will be simply expressed as the following formula:

utility	= 1/denominator
denominator	= a sum of ratios of current chemical concentrations compared to the desired chemical concentrations + the computer time it took the genetic operators to modify the string during evolution + 10 if the program was unable to correctly accomplish allosteric modification.

From the quantities in the denominator of the formula for calculating utility, we can perceive two relationships: 1) The larger the deviation of the chemical ratios is from 1, the smaller is the utility; 2) The greater the failure to solve for allosteric modification is, the smaller is the utility. In regard to these two points, we should notice that our criteria for selection (by way of the utility function) are clumsy and unrealistic. First, "clumsy and unrealistic" should be enlarged a bit. We used computer time in judging our program's performance. Computer time doesn't even exist in real cells. We destroyed sixty percent of the programs each generation. Death rates are not this regular in real populations. However, even these primitive genetic procedures do induce a complex quantity, average excess, on each string in the population. To obtain this average excess, we would have to calculate it over the run of the program. It is important to perceive that although this average excess exists, it does not appear as

a number in the running genetic-program which effects the evolution of programs in the computer. If one were simulating the theory of evolution rather than attempting the evolution of an effective (here effective means metabolically stable) program, one would probably want to calculate the average excess of each program, rather than to define it implicitly. Our implicit definition of average excess consists of the genetic procedure we use when we produce new programs from old ones.

A brief consideration of the probability of replacement of a program in the population shows that the amount of utility judged to be associated with the program influences the probability that the program will be erased by its genetic supervisor. In our example, 60% of the programs disappear after a run. In order to be a member of the survival population, a newly generated program has to be in the best 40% (4 top out of a population of 10). The higher the value of one of the old surviving programs is, the less likely it is that the old-timer will be supplanted by a newcomer in the next reproductive cycle. I do not want to discuss these calculations in detail, since my main point is simply to use Holland's formal theory of adaptive systems to support the validity of writing extremely simple, albeit evolutionarily powerful, heuristic programs.

In exploring a genetic space with evolutionary procedures, we can never expect to explore the entirety of the genetic space. Indeed, part of the value of the evolutionary search is the exploration of "interesting" regions of the genetic space. The type of simulated cell we start out with in our simulated evolution makes our search space even more interesting. We have used sophisticated molecular interactions in our

beginning simulated cell, and we use this as a take-off point for further evolution. We have included negative feedback of metabolic processes at both the DNA and cytoplasmic levels. We have also used positive controls of DNA and cell division. All of these mechanisms enable real cells to survive, and imply that further evolution will take place in a particularly productive subset of possible physiological states. Furthermore, the molecular mechanisms underlying many of these sophisticated relationships are often simple and direct, and easy to program. For example, DNA replication involves only a few basic concepts. We have an initiation site at which DNA replication begins. Once DNA starts to replicate, it continues to replicate until the whole DNA molecule has been duplicated. A few simple rules of this type insure a stable transfer of genetic information from one generation to the next. The ease with which one can program realistic life-phenomena makes it easy to test fairly large populations of cells as to fitness in a particular environment.

We have considered methods for altering the genetic makeup of populations of simulated cells. These alterations will allow us to explore a space consisting of alternate genotypes. The point at which we begin our exploration, as well as the region we will investigate, are very small as compared to the complete space of all possible genotypes theoretically available to the simulated cell. The complexity of our simulated cell at the beginning of an evolutionary run implies that we begin by storing information in a complex genotype. In effect, we begin our evolution from an interesting point in our genetic space. We are limited as to the modifications of this genetic point we may effect.

Genetic inertia exists in that genetic changes from a given point in evolution are small compared to the total changes which can occur in a very long period of evolutionary time. A mutation only effects a relatively small change in total genotype. The same may be said for other genetic modifications which occur. Which genetic changes survive is very closely dependent on the structural and functional relationships existing in the cell. Sophisticated evolutionary schemes may well embody this information. We hope that by allowing our genetic programs to evolve, we can obtain a good evolutionary scheme for our simulation. In simulating our evolutionary scheme, we must specify the genetic loci we wish to use, as well as procedures for modifying the values at each locus.

Genetic Loci: We will pick as unfixed variables from which to generate our population of strings (and schemata) fifteen control parameters which the cell uses for phenotypic adaptation to changing environments (Figure 6.4). These fifteen parameter fall into two natural groups: 1) five control constants (R(1), ..., R(5)) which function to repress enzyme production by DNA (see appendix to Chapter II). 2) ten control constants (P1V(1), ..., P1V(5), P3V(1), ..., P3V(5)) which function to calculate allosteric modification of enzyme activity after the enzyme has already been formed.

The computer details of the total evolutionary system are straightforward. We will keep the simulated-cell program, and all variables in the program, in program common. (Program common is a fixed storage area in the computer which is shared by many sub-programs.) We will instruct the computer to bring into its core storage simulated-environments, an adaptive genetic-program, and the ten strings supervised by the adaptive genetic-program. These will be brought

in from disk storage. The program for the simulated cell receives the values of variables according to the information in a string. Each string represents one individual cell. The simulated-cell program will be run, and a utility will be calculated for it according to how well it did at maintaining metabolic stability in fluctuating simulated-environments. The adaptive genetic-program will calculate this utility. The supervising adaptive genetic-program will continue to allow cells in its populations to grow for a short time, until it has awarded a utility to each of the ten strings in its population. The adaptive genetic-program will then operate on its ten-string population to form a new population. The adaptive genetic-program will direct evolution of its ten-string population according to two kinds of information: 1) Utilities are awarded to the members of its population. The adaptive genetic-program will use these utilities to select the best members in its population. 2) Genetic information is present in the genetic-program itself. The adaptive genetic-program will utilize its own genetic information to operate on the strings in its population. After the adaptive genetic-program has generated a new, and hopefully better population, both the adaptive genetic-program and its new ten-string population will be returned to disk storage. The next adaptive genetic-program, together with its own ten-string population, will be brought into core storage. The process of evolution will be repeated for this population. They will be returned to disk, and the next adaptive genetic-program and its population will be brought into core for a similar procedure. This will be repeated until all populations have been run, and have gone through one reproductive cycle (Figures 6.2 and 6.3). We have said that each population of strings has been modified, but what about the adaptive genetic-programs them-

selves? In order to allow the adaptive genetic-programs to evolve, a non-adaptive genetic program will direct matters. The non-adaptive genetic-program will judge the excellence of each of the adaptive genetic-programs. It will do this by calculating the utility of the best string in the population of each adaptive genetic-program. The non-adaptive genetic-program uses its own, supposedly inviolate formula to calculate these utilities. Each adaptive genetic-program will then receive the utility of the best string in its final population. This utility is the one calculated by the non-adaptive genetic-program. After the non-adaptive genetic-program has awarded each adaptive genetic-program a utility, it will supervise the evolution of the population of adaptive genetic-programs. This evolution will be analogous to the previous evolutionary procedure we have described. To effect evolution of the four adaptive genetic-programs they will be loaded into core, together with the non-adaptive genetic-program. The two worst adaptive genetic-programs will be replaced by programs generated from the two best genetic-programs. The non-adaptive genetic-program will use its genetic operators to form these two new adaptive genetic-programs (Figures 6.2, 6.3).

Simulated Phenotype from Genotype: Information concerning genetic manipulation of the evolving strings will be stored in the adaptive genetic-programs (Figure 6.5). This information determines crossover, inversion, mutation and selection. The adaptive genetic-program will determine which locus will mutate in a string. Once the locus has been chosen, the string itself will contain information as to the mutation pattern. The pattern of mutation will be referenced by the third and fourth rows of the array containing the string which will mutate. The non-adaptive genetic-program directs evolution

INDEX OF ENTITY	ENTITY	ATTRIBUTE RANGE	RANDOM NUMBER INTERVAL AND CUMULATIVE FREQUENCY DISTRIBUTION USED TO EFFECT GENETIC OPERATION OF INDEXED ENTITY
1	crossover	(1,14)	Random (1,N(1)), Freq. (1)
2	inversion	(1,14)	Random (1,N(2)), Freq. (2) length = 2*Random (1,5)
3	mutation	(1,15)	Random (1,N(3)), Freq. (3)
4 5 6 7 8 9 10 11 12 13 14	coefficients for utility polynomial for string being operated on.	(0,1)	Random (1,N(4)), Freq. (4) etc.
15	Utility of this adaptive genetic program, calculated by non-adaptive genetic program.		

Figure 6.5 Adaptive Genetic Programs.

Adaptive genetic programs: N(I) and FREQ(I) are determined by a non-adaptive GENETIC PROGRAM, which also operates on the adaptive genetic programs as evolving strings. The nonadaptive GENETIC PROGRAM determines utility by a polynomial which evaluates the best individual the adaptive genetic program offers it from its population. The attribute of the adaptive genetic program references the column in the string(s) upon which the adaptive genetic program is currently operating.

of the adaptive genetic-programs in much the same way that the adaptive genetic-programs direct the evolution of the strings of individuals (Figures 6.3, 6.5 and 6.6).

The actual mechanics of programming are straightforward. The description of the $(I + J)^{th}$ string is easily obtained by a DO loop which loads the attribute of the entity into the entity in the performing program. This is done for ENTITY(1) through ENTITY(14). The program for the simulated cell then has the proper values for its variables. It can therefore make a simulated run and receive a utility rating. The indexes for the respective entities, together with the attribute associated with the entity, are stored in the CHROM array. The value of the third dimension of the array is equal to I + J to indicate that the $(I + J)^{th}$ string is under consideration. The program statement is

```
      DO 1, K = 1, 14, 1
    1 ENTITY(CHROM(1,K,I + J) = CHROM(2,K,I + J)
```

Simulated Crossing Over: Crossing over is very easy to program since all individuals which form crossover pairs have the same linkage map. Inversions do occur, but when an inversion is produced, it is used to generate a population which evolves as a group. The unequal probabilities of crossing over for different regions of the linkage map is realized by associating a cumulative frequency-distribution with the crossover operator indexed in the genetic program (adaptive or non-adaptive). This probability is an attempt to simulate the inequalities in probability of crossing over for different regions of real chromosomes. Such inequalities are induced by the presence of inversion heterozygotes during crossing over in real organisms. Such inversion heterozygotes

INDEX OF ENTITY	ENTITY	ATTRIBUTE RANGE	RANDOM NUMBER INTERVAL AND CUMULATIVE FREQUENCY DISTRIBUTION USED TO EFFECT GENETIC OPERATION ON CURRENT ADAPTIVE GENETIC PROGRAM
1	crossover	(1,14)	Random (1,14)
2	inversion	(1,14)	Random (1,14) length 2* Random (1,5) or less.
3	mutation	(1,15	Random (1,15) for entity attribute/2 for magnitude.
4 5 6 7 8 9 10 11 12 13 14	coefficients for utility polynomial for best individual produced by adaptive genetic program being operated on	(0,1)	all coefficients = 1.
15			

Figure 6.6 Non-adaptive Genetic Program Directing the Evolution of Adaptive Genetic Programs as they Operate on Populations of Strings.

are not simulated because of the complications introduced into the programming procedure for crossing over. Inversions are simulated, however, since they are a powerful permutation operator allowing the evolving populations to experiment with various linkage maps.

An example of crossing over will be programmed. The two individuals with the highest and next highest utilities will be used in our example. We will consider the population which is supervised by ADAPTIVE GENETIC PROGRAM (1). These strings will be ordered so that they occupy the first two positions in the population of ADAPTIVE GENETIC PROGRAM (1) i.e., CHROM (I,J,1) refers to the Ith row, and Jth column of the individual with highest utility in the population supervised by ADAPTIVE GENETIC PROGRAM (1). CHROM (I,J,2) refers to the string with second highest utility in an analogous manner. To obtain the crossing over parameter, a random number is generated in the range 1 to N (N is stored in the location CHROM(41,2), i.e., N is the attribute of the 41st entry.) The crossover will then occur at the right of the column in the CHROM vector designated by the random number if the random number is less than the number of columns in the CHROM vector; otherwise no crossover will take place. The larger the value of N is, the less is the probability of a crossover. We might want to use a probability distribution other than the uniform distribution for crossing over. To generate any particular distribution, the cumulative distribution of that particular probability distribution could be used. First, a random number (between 0 and 1) would be generated. One would then pick the point on the chromosome whose cumulative distribution function is less than, but closest to, the random number (This type of simulation technique is presented in Mize and Cox (1968)). This Monte Carlo

technique can be used to obtain probability distributions for mutational increments for any desired probability distribution.

To program a crossover between individuals CHROM (I,J,1) and CHROM (I,J,2), the following sequence of instructions can be used. I and J are variable, and denote individual strings. The crossover takes place at position X. The crossover products are loaded into strings CHROM (I,J,5) and CHROM (I,J,6).

```
      K = 1
      DO 1 I = 1,4,1
      DO 1 J = 1,X,1
      CHROM (I,J,K + 4) = CHROM (I,J,K)
1     CHROM (I,J,K + 5) = CHROM (I,J,K + 1)
      DO 2 I = 1,4,1
      DO 2 J = X,15,1
      CHROM (I,J,K + 5) = CHROM (I,J,K)
2     CHROM (I,J,K + 4) = CHROM (I.J.K + 1)
```

KEY

K is the base for denoting the individual.

I denotes the row of the CHROM array.

J denotes the column of the CHROM array.

the following 6 lines effect crossing over.

Since the best 4 strings are saved after a round of phenotypic adaptation and evaluation by the ADAPTIVE GENETIC PROGRAM (1), the recombinant will be loaded into CHROM (I,J,5). When the recombinant is loaded into CHROM (I,J,5), the recombinant replaces the 5th best individual in the population. If both products of the crossover are saved, the second recombinant will be loaded into CHROM (I,J,6). We will now give the formula for picking out the column for crossover. As we mentioned, the crossover point will be obtained by Monte Carlo techniques. We will use a random-number interval and cumulative-frequency distribution.

They will be specified by information located in the column of the adaptive genetic-program which indexes the crossover entity. (1 happens to index crossover, so the 3rd and 4th rows of the column containing a 1 in its first row will contain, respectively, the random number interval and the cumulative distribution used to generate the point of crossing-over). For a simplified example of Monte Carlo techniques, let the random number be generated over the interval (1,14). Let x be the column to the left of the crossover point. Let F(x) be the cumulative distribution for the probability of a crossover occurring to the right of x. The formula for crossover point μ is: $\mu = F^{-1}(x)$ (Mize and Cox, p. 74). The crossover does not occur if a random number is generated outside of the range of $F^{-1}(x)$. This allows mutation to increase or decrease the probability of crossing-over for the whole chromosome. The change in probability of crossing-over is accomplished by changing the length of the interval over which the random number is generated. For this procedure, the interval must be wider than the range of $F^{-1}(x)$. If this sophisticated procedure for picking a crossover point is not useful, it can be discarded by the genetic-program. The probability distribution can be replaced, for this simplification, by a random-number interval with a range extending from 1 to 14.

Biological crossover-frequency is influenced by many factors. We have already mentioned inversion heterozygotes. Chemical differences on different parts of real chromosomes also alter crossover-frequency of each chromosome region. There are also complex interactions between

different parts along the length of chromosomes undergoing crossing-over. However, the assumption of constant crossover-frequency per unit string-length for simulated strings is a close approximation to the relation between percent recombination per unit physical-length for real chromosomes. The linkage map of the real chromosome approximates a linear-function of percent-recombination for map-distances of less than forty percent. Constant probabilities of crossover per unit-length is therefore a reasonable first approximation for Holland's theoretical development. However, the probability distributions we use to simulate the action of inversion heterozygotes will also take care of much of the genetic control exercized by real cells on different rates of crossover for different regions of their chromosomes. Since the probability distribution used is under genetic selection, and the probability distributions stored in the computer may be mixed for Monte Carlo simulations (Mize and Cox, Chapter 6), this simple expedient realizes many complex genetic-functions. Therefore, generating probability distributions for crossover actuation seems a practical utilization of computer facilities in effecting simulated evolution. Experimental observations upon relationships between linkage, percent recombination, and cytological observations are available in the literature (Strickberger, 1968).

Simulated Inversion: Inversions are somewhat artificially simulated. For programming simplicity, there is never any population in which different members have different inversions. This is accomplished as follows. The best adaptive genetic-program, as judged by the non-adaptive genetic-program, selects a site of inversion. This inversion site will be eventually introduced into a new population of strings. To select

the actual inversion site, the adaptive genetic-program uses a random number interval. It chooses the frequency distribution indicated by the column which indexes the inversion entity (the column in the adaptive genetic-program with a 2 in row 1, since 2 is the index for inversion, and row 1 contains all indices). The adaptive genetic-program inverts the proper segment of the strings in its evolved population. The population with the newly introduced inversion is then loaded into the space for the population supervised by the third-best adaptive genetic-program. The best adaptive genetic-program keeps its original, un-inverted population for itself. Another inversion is introduced into the population of the second-best adaptive genetic-program. As in the previous process, the inverted population is loaded into the space for the evolving population supervised by the worst adaptive genetic-program. The second best adaptive genetic-program keeps its own un-inverted population. At this point each of the two worst adaptive genetic-programs will themselves be replaced with modified versions of the two best adaptive genetic-programs. The point of this complex interaction is the improvement of linkage maps for the purpose of putting genes in favorable positions on the string (recall our discussion of co-adaptation).

Simulated Mutation: The locus to undergo mutation is chosen by a genetic program. The mutational increment is then obtained by using the random number interval stored under the locus (entity) undergoing mutation. Mutation in a string is effected by its adaptive genetic-program. Mutation in an adaptive genetic-program is analogously effected by the non-adaptive genetic program. The non-adaptive genetic-

program only mutates under direct manipulation by the programmer. An example of the kinds of values used in directed mutation follows. The entity to mutate is k(5), which has a current value of 5. The mutational increment is set at n · 5 by the information stored (along with the index of the entity) in the CHROM array. The value for n is obtained by calling on the random number generator, and generating a number between -20 and +20 as directed by the density distribution specified along with the index of the entity. The mutational increment thus ranges from -20 · 5 to +20 · 5, in intervals of 5. The next value of k(5) will be one of the numbers in the range -20 · 5 + 5 to +20 · 5 + 5. This value of k(5) will be obtained by adding the value of the mutational increment (-20 · 5 to +20 · 5) to the current value for k(5) which is 5. This procedure differs from natural mutation where most mutants are random alterations. The advantages of directed mutation have been discussed.

Other Genetic Operators: Duplication and Lumping Loci: In duplicating a gene, the action of the operon in the simulated cell is particularly interesting. The control system used by the operon can be modified to signal us that a particular gene needs duplication. The idea will be that one copy of a gene is not doing its job correctly. Another copy of the gene will be added to a string so that the original gene may be modified during further evolution. An alternate procedure would be to directly introduce a modified gene which is more complex than the gene which is not doing its job. The operon control system will be adapted to signal the need for gene duplication as follows. The signal for gene duplication will be that a quantity in the cell reaches a danger level (either too high or too low). The response will be to duplicate the gene asso-

ciated with the dangerous concentration. The duplication can be accomplished in three ways: 1) The gene may be duplicated, so that the duplicate copy may be modified in further evolution. 2) A modified duplicate of the gene may be added to the string from a prepared list of genes. 3) The program may be interrupted, so that the operator adds a complicated version of the gene to the string. This does not exactly duplicate the gene, but approximately duplicates it. In a program as complex as the simulated cell, the duplication operator would be a signal for man-machine interaction. The man would modify the subroutine not predicting correctly, or he would write a new subroutine to extend one already present. The quantities which signal danger are easy to define for the simulated cell. Each parameter in our simulated evolution is closely associated with a simulated activity in the simulated cell. P1(k) and P3(k) would need modification when the catalytic activity of enzyme EK(k) increased in spite of the fact that too much PRDC(k) was already present. R(k) should be examined if MRNA(k) increased when PRDC(k) was already too high, or if MRNA(k) decreased when PRDC(k) was too low. Either of these latter two actions would indicate that R(k) is not doing the job it is predicted that it will do.

In a program with simpler subroutines, the signal for duplication of a locus might well enable the genetic-program directing evolution to modify an old locus to produce the needed function. Cavicchio (1968) has written a program with such simple subroutines. Cavicchio's program consists of a population which evolves the ability to recognize patterns. Cavicchio's subroutines are simple enough to modify by machine rather than by man.

Another genetic operator we might want to use would be a lumping operator. The lumping operator would modify strings. A gene would be merged with another gene by the lumping operator. The lumping operator would convert the references to two parameters to a reference to one parameter. Two separate genes would become a single gene after lumping. The lumped genes are closely related to a successful schema, since their survival as a unit indicates that they are successful in a combination with each other. By allowing directed mutation within the lumped group of genes, one may reap the reward of hidden recessives becoming dominant. One could also maintain the advantage of keeping a co-adapted set of genes linked during evolution.

Generality of Reproductive Scheme: Since the parameters indexed by the chromosome arrays are not limited to biochemical rate-constants, the realization of Holland's reproductive scheme as a computer simulation may be used to explore many different genetic spaces. This generality may be used to realize various heuristic programs. Examples of diverse evolutionary programs are 1) selection of sets of subroutines useful in pattern recognition (Cavicchio, 1968), or 2) the evolution of programs which use a generative grammar to produce English sentences (Bono, 1968). Both of these tasks have been written as populations of computer programs which evolve over time. Their evolution is a function of how well they do the specific task assigned to them. Heuristic programming may have interesting applications. We can obtain programs by heuristic programming which carry out ill-defined algorithms. Tasks with a well-defined goal-and-reward scheme are particularly suitable for evolutionary programs (Holland, 1969a, b, and c).

CHAPTER VII

A CELL SPACE EMBEDDING OF SIMULATED LIVING CELLS

(Goodman, Weinberg & Laing, 1970)

Summary

The computer simulation of the metabolism of a living cell, ONECELL, has been embedded in CELLSPACE, a tessellation model for expressing and manipulating intercell relationships. ONECELL and CELLSPACE parameters can be set so as to achieve normal, slow, monolayer growth of cells, or abnormal, rapid, multilayer growth. The biochemical basis of such phenomena is explicit, and can be varied at will.

Introduction

Our simulated living cell is named ONECELL for the following development. Just as different states exist for our simulated living cell, different states exist for different copies of ONECELL. The universe in which populations of copies of ONECELL are embedded is called CELLSPACE. That is, in order to investigate interactions among developing *populations* of cells, we have embedded ONECELL in a tessellation space (an automata-theoretic cell space with a fixed-neighborhood relationship). We simulate the tessellation space using a flexible set of computer assembly-language routines; this set of routines is called the Cellular Space Simulator (CELLSPACE). The combined system (ONECELL + CELLSPACE) is being used to study the biochemical events responsible for normal and abnormal development in groups of living cells.

"Cell" will be used in three ways in our development: 1) A cell in CELLSPACE refers to a coordinate in the space. It locates a point in the universe. 2) Cell, as applied to ONECELL, refers to a computer-simulation of a living cell. Any particular copy of ONECELL grows according to information as to its neighborhood, and its internal state. Therefore, different copies of ONECELL may become different as to their physiological states. 3) A living cell is a real, laboratory entity. It refers to a real cell grown in a laboratory, and determines a data base with which we compare simulation data. We hope that context will aid in clarifying the different uses of "cell".

Information in the ONECELL + CELLSPACE system is handled as follows:

1) In order to determine its metabolic status each ONECELL must sense and react to the environment (nutrients, etc.) at its CELLSPACE location. (The individual cell's metabolic behavior has been adjusted to agree with laboratory data.)
2) Each ONECELL must communicate information of its internal metabolic state to its self-replication mechanism. (The experimental evidence of the Helmstetter-Cooper model of replication has been incorporated. We will describe the Hellmstetter-Cooper model later.)
3) CELLSPACE must calculate the spatial effects of cell growth (both in terms of cell size and numbers of cells) and communicate the new spatial (as well as nutritional) properties to each ONECELL location. (The tessellation-automaton concept of von Neumann, as implemented in a computer by R. Brender (1970) and T. Schunior, has been employed here.)

By suitably choosing the CELLSPACE condition for inhibition among neighboring cells it was possible to achieve slow, monolayer growth by "normal" cells, while "abnormal" cells grew more rapidly and in multiple layers.

Relatively minor changes in the metabolic pathways can produce markedly different growth patterns in colonies of living cells. The biochemical basis of such global effects in cell growth can be examined in detail, since the underlying biochemical equations responsible for these effects are completely explicit and can be varied at will.

The Model of DNA Replication in ONECELL: Self-reproduction in ONECELL is controlled by an initiator substance in association with the cell's genetic complex; the Helmstetter-Cooper model for control of DNA replication is incorporated as part of our genetic apparatus.

The Helmstetter-Cooper theory (1968) provides the most sophisticated model of DNA replication presently available.

Basically, the Helmstetter-Cooper model asserts that the life sequence of a cell begins with the accumulation of a protein substance which is the "initiator" of chromosome replication. When a threshold amount of initiator substance has accumulated, replication of DNA commences at a fixed end-point on the chromosome and the replicating process proceeds to traverse the entire length of the chromosome. At the completion of DNA replication, a sequence of further cellular events follows, terminating in physical cell-division. Even before completion of cell-division, however, a new DNA replication cycle may have begun -- this will depend on the amount of initiator substance accumulated by that time. It is thus possible for the DNA-replication process to be

taking place simultaneously at several points on the chromosome, as represented by the multiple forks shown in Figure 7.1.

The ultimate rate of cell division is thus a function of the periodic attainment of a sufficient fixed amount of the initiator substance which is synthesized continuously and which can be affected by the richness of the cell's nutritional environment.

Under optimal pH, temperature, and cell density conditions, "log phase" *E. coli* cells double one to three times per hour. The rate is a function of the medium in which the cells are grown. Although cell life outside this "log phase" range should not be ignored, much of what we wish to know of cell life takes place under these conditions. In particular, many anomalies of control of cell behavior (including some "cancerous" behavior) takes place while cells have a "normal range" nutritional environment.

Implementing the Replication Model in ONECELL: ONECELL's rate of initiator production is a function of the metabolic rates along the various simulated, internal biochemical-pathways; these rates in turn are dependent on the nutritional levels supplied to ONECELL.

In ONECELL we can simulate the effect of growth in three different media: 1) mineral glucose medium; 2) a medium to which amino acids (casamino acids) have been added (thus obviating the necessity for the cell to synthesize these itself); and 3) a still richer medium which contains both amino acids and nucleosides (broth). In ONECELL these three nutritional environments result in steady-state cell-division rates of once every fifty minutes, once every twenty-eight minutes, and once every twenty-five minutes respectively. In effect, through the

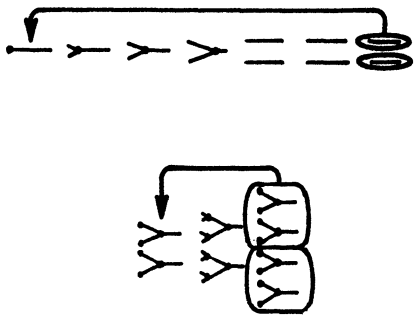


Figure 7.1 DNA Replication Model for 60 and 20 Minute Cells (after Helmstetter, Cooper, et. al.)

mechanisms of the replicon, we simulate the production of varying amounts of DNA, depending on the richness of the nutritional environment, and the DNA amount in turn has both internal biochemical consequences and consequences for the overall cell-division rate.

The replication mechanism in ONECELL may be summarized as follows: the rate of initiator production depends on concentration of internal pools of amino acid, ATP, and the enzyme catalyzing the production of initiator from these substrates. This internal information from ONECELL's cytoplasm simulation is transmitted to the replication apparatus. When sufficient initiator substance is produced, replication begins at a DNA initiation-site. Replication of the DNA molecules takes 40 minutes to complete. After the replication of a DNA molecule is completed, the cell will proceed to divide; the physical division of a cell into two cells occurs after twenty minutes, and each of the two new cells receives half of the DNA present in the original simulated cell.

Description of CELLSPACE; Background: In order to present a mathematically rigorous proof of machine self-reproduction, J. von Neumann (1966) introduced the idea of cellular automaton systems. His cellular model can be described in an informal way as an infinite chess board, each square of which contains a copy of the same (in his case twenty-nine state) finite automaton. One decides on some initial setting of state for each of the automata occupying the squares. This determines the state of the whole system at the initial moment of time. At subsequent discrete moments of time, the state of the automaton in each square at time t + 1 is uniquely determined by its state at time t and the states, also at time t, of its four "North, East, South, West" neighbors. Usually all but a finite number of the automata in the

squares are initially set at a quiescent state. This region of quiescent states is usually interpreted as the unorganized external environment into which the developing cellular organism (expressed in the active automaton squares) may penetrate.

Using this system, von Neumann was able to demonstrate constructional universality (including self-reproduction) for non-biological systems. Cellular-automaton systems can, however, be used to express the behaviors of many other systems (including the biological) in which "modest" initial conditions and local neighborhood relations may grow to have global effects.

One need not be limited to "checkerboard", two-dimensional, cellular spaces. The finite automata in each cell space need not be limited to the twenty-nine state devices von Neumann found sufficient. The neighborhood relationship may involve other than "cardinal direction" cells. Although our application does use the "cardinal direction" neighborhood, the number of states in each of our finite automata is very large, each state representing a possible biochemical state of a living cell.

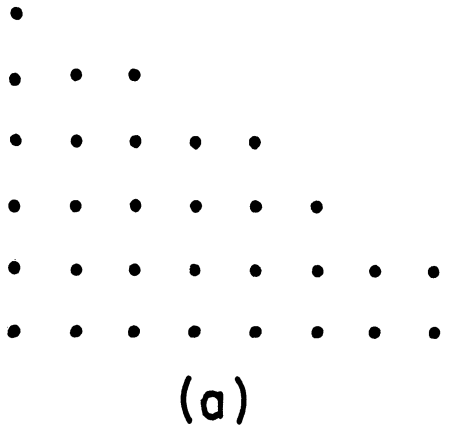
At the Logic of Computers Group at The University of Michigan we have engaged in a number of theoretical analyses of cellular spaces and other parallel computation spaces (Brender, 1970; von Neumann, 1966; Codd, 1968; Zeigler, 1970). We have also designed and implemented a cell space computer simulation system (CELLSPACE) in which cell array size, dimensionality, transition function, and neighborhood relations can be easily assigned and easily modified (Brender, 1970).

Modelling a Colony: The concept of a cell space captures in the abstract many fundamental properties of populations of living cells.

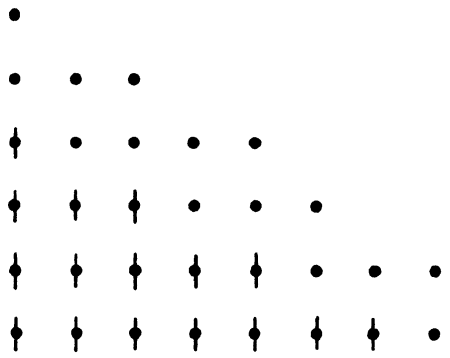
Almost all of the cells in an individual organism have the same genetic complement; all of the coordinates of the cell space have the same transition function.

The complex biochemical interactions leading to the complete adult organism are functions of the chromosome complement of the original zygots, together with the neighborhood relationships established as the cell reproduces. Self-reproduction and specification of neighborhoods are natural concepts in cell spaces, and make cell spaces good vehicles for studies of the development of a few living cells into large numbers of interacting cells.

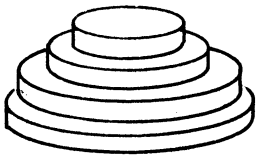
In order to maintain the potential for comparing the output of the simulation with laboratory data, the simulation was designed to model the growth of cells on a solid medium (in a Petri dish, for example). Because of the obvious need to model large colonies using as few simulated cells as possible in representing them, it was decided that a simplifying assumption of cylindrical symmetry about the center of the colony should be made. Under this assumption, it is possible to represent the three-dimensional colony in a two-dimensional cell space. Figure 7.2(a) shows the pattern of 31 cells which has been used for the initial experiments. Each point in the figure represents a possible position of a cell in a vertical section through a radius of a colony; i.e., if the figure were rotated around the left hand vertical row of points, the shape of the colony represented would be traced out. Of course, the arrangement of these 31 points is such that it can accomodate a variety of colony shapes; however, the right hand "diagonal" edge is a boundary since colonies would not grow with a vertical edge.



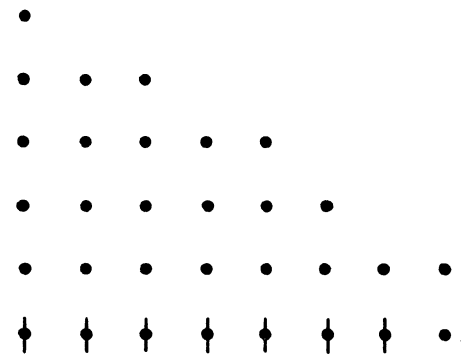
(a)



(b)



(c)



(d)

(e)

Figure 7.2 Preliminary Experiments Using ONECELL + CELLSPACE - (a) the space; (b) fast, multi-layer growth; (c) colony shape represented in (b); (d) slow, monolayer growth; (e) colony shape represented in (d).

In practice, the simulation must be stopped (or the space enlarged) when cells reach this boundary. The bottom row represents the layer of cells resting directly on the growth medium. When the simulation is started, only the lower-left hand point actually represents cells; the other points will come to represent cells when adjacent cells divide and daughter cells assume positions at the new points. It should be made clear that the few points in the space represent large numbers of cells in the colony; in fact, each represents all of the cells in the colony at a particular distance from the center and height from the dish. Thus these 31 cells (which can readily be expanded up to 50 or more) are used to simulate colonies of several hundred cells, and can simulate more cells if less detailed representation is acceptable. The position of a cell in the cellular space (e.g., bottom row, third from left) is not necessarily indicative of absolute position of the modelled cells in the colony. It indicates relative position; however, a better idea of the geometry or shape of the colony is obtained by computing, for each cell in the space, the distance of the cells it represents from the center of the colony. This is done using the information stored at each point concerning the number and volume of the cells it represents.

Interactions Among Cells: Interactions among cells in the colony are modelled by allowing each cell in the space to sense portions of the states of the cells which surround it, or are in its "neighborhood". The neighborhood is here defined to be the 4 nearest points, i.e., above, below, left (toward the center of the colony), and right (toward the outer edge). Of course, each cell also has access to the state of the other cells in its own radius, since they are represented by that cell itself.

It is important to notice some properties of information transfer in a cell space. With the neighborhood relationship defined as above, information can "flow" at a rate of only one cell per time step. That is, a change of state in cell A two points away from cell B cannot influence cell B immediately, but must spend one time step in influencing an intervening cell. In a sense, one might expect that this consideration would prove to be very restrictive; its effect seems to be minimal. It does not prevent one from modelling such "global" types of information transfer as nutrient diffusion; all it requires is that the time step be sufficiently small that the effect of the diffusion is not significantly altered by the time required for the effect of a change to propagate throughout the space.

Computer Implementation of ONECELL and CELLSPACE: The simulation computation has been carried out on an IBM 1800 equipped with 16 K of core storage and associated disk memory. CRT pictures of the states of the simulation are made available to the investigator on a DEC display driven by a PDP-7. The PDP-7 is, in turn, connected to the IBM 1800 by means of a specially designed high-speed interface. Investigator interaction with the simulation during run-time is carried out through the teletype keyboard of the PDP-7 and the display's light pen.

The cell-space simulator, CELLSPACE, allows the user to specify a finite cellular space by supplying its transition-function, neighborhood relationships, initial state, and other such data. For our application, CELLSPACE is supplied a call to ONECELL's FORTRAN routines which furnish a part of CELLSPACE's transition-function. CELLSPACE also requires specification of certain parameters and procedures concerned with the implementation itself -- for example, translation

tables for the visual display routines, an external-cell procedure, etc.

ONECELL is actually a general name for a set of programs with various capabilities. It consists of 1) a set of routines which are used to establish the parameters for a single cell by "growing" the cell in each of three environments; 2) a set of routines for examining the behavior of a single cell in a single environment, of which a subset is used in the ONECELL + CELLSPACE runs; and 3) a set of routines for taking a "snapshot" of the 1,800 words which comprise the state of the cell and all of the associated parameters at any time in any environment, and storing this information on the disk for use in the ONECELL + CELLSPACE configuration. All of these routines are written in IBM Basic Fortran IV.

Typical operation of the combined ONECELL + CELLSPACE system begins with the establishment of the parameters of the cell which is to be the first cell in the colony. The routines to accomplish this (Link 1) are much too large for the 16 K core of our IBM 1800, so the program is written to allow five sections of Link 1 to be loaded on call, overlaying one another. These routines allow the user to obtain a variety of printed outputs or CRT display (controllable at run-time) as the cell is shifted through a variety of environments. When the desired environment for simulation in the cell space is reached, a data switch can be used to cause the entire COMMON area (containing both system parameters and the current state of the cell) to be written to the disk. One can then choose Link 2, to examine in detail the behavior of the cell in isolation (to check out new intracellular mechanisms, etc.), or Link 3,

the first link of the ONECELL + CELLSPACE programs, to start simulating the growth of a clone of cells from the data for the single cell stored on the disk. Link 3 does certain data manipulation and debugging output, and calls Link 4 to perform the growth simulation in CELLSPACE.

Link 4 contains an IBM 1800 assembly language program which specifies certain parameters and procedures for CELLSPACE and provides calls to the portion of ONECELL which performs the "biochemical" transitions for each cell at each time step. Link 4 also includes FORTRAN subprograms and functions, for allowing interactions of various chemical pools among neighboring cells and interactions of the cells with the nutritional medium and the surrounding air. All output from Link 4 is via CELLSPACE, and is in the form of decimal digits displayed on the DEC 338 Display Scope associated with the PDP-7 computer. The entire simulation process carried out in Link 4 is under the user's control at run time. Transitions, output, parameter changes, and all other manipulations are controlled either from the 338 using a light pen or from the PDP-7's teletype. The commands presently implemented include commands to:

1. perform one transition step for each cell,
2. back-up one step,
3. perform transitions until halted,
4. display the 31 cells on the CRT using the current display map,
5. change the display map,
6. change the scale or position of the display,
7. save the external-state vectors (for access by neighboring cells) on the disk,

8. restore the external-state vectors of all cells from the disk,
9. save the internal-state vector (biochemistry) for a cell on the disk,
10. manipulate the internal-state vectors stored on the disk,
11. change selected parameters in the transition function.

In a sense, each cell in the ONECELL + CELLSPACE simulation is a copy of ONECELL, together with certain modifications to allow interactions with neighbors. It is not necessary, of course, to duplicate the transition-function of ONECELL for each individual cell in the simulation, but rather to store the *state* of each cell in the space. Then transitions can be carried out for each cell in sequence by applying the transition-function to each state vector in sequence. Of course, the next state of any given cell depends not only on its present state, but also on the present state of its neighboring cells in the space. In this implementation, however, only a portion of the state vector of the neighboring cells is available to the transition-function. This is necessitated by core storage limitations, but it should be clear from the following that it represents no real restriction on the capabilities of the simulation. The entire state vector for each cell (i.e., ONECELL's state vector) contains approximately 60 real variables (120 IBM 1800 words). Because it is desirable to be able to simulate as many cells as possible, this 60-variable state vector is partitioned into two parts: 1) the *external*-state vector and 2) the *internal*-state vector. The external-state vector contains ONECELL's external state. The external state of ONECELL is that part of ONECELL's state which directly influences neighboring copies of ONECELL. Nutritional environment is a part of the external state of ONECELL, and nutritional environment is an entry in the external-state vector asso-

ciated with ONECELL. A particular copy of ONECELL uses up nutrient, and produces waste. This is reflected in the external-state of that copy of ONECELL. The value of the nutritional concentrations are stored in the external-state vector which is associated with that particular copy of ONECELL. Neighboring copies of ONECELL can observe this, and other, external-state vectors. These neighboring copies of ONECELL use the external-state vectors to calculate the nutrition available to them. A growing copy of ONECELL senses its nutritional environment by observing the external-state vectors of surrounding coordinates of CELLSPACE. The internal-state vector contains ONECELL's internal state. The internal-state of ONECELL is that part of ONECELL's state which influences the metabolism of ONECELL itself. Nucleotide-pool concentration (NUC) is a part of the internal-state vector which is associated with a particular copy of ONECELL. This individual copy of ONECELL can remember what its NUC value is by observing its internal-state vector. Only the external-state vector for each cell remains in core storage at all times for access by neighboring cells. The internal-state vector for a cell is copied from disk into core only to perform the transition for that cell. The external-state vector consists (at present) of 16 words. These 16 words are of two sorts, integer and real. Each integer-variable requires 1 word of storage. Each real variable requires 2 words of storage. The 16-word external-state vector contains 6 integer-variables (requiring 6 words of storage) and 5 real-variables (requiring 10 words of storage). In determining the external-state of ONECELL, the external-state variables are used. The formation of the external-state variables expresses the external-state which a particular copy of ONECELL shows to its neighbors. The way the external-

state of ONECELL is formed may be changed prior to each simulation run. Specifically, the use of 3 integer-variables and 3 real-variables may be decided upon prior to a simulation run. Since we are permitted to vary the use of our external-state variables from one simulation run to the next, we can study a wide range of interactions among populations of ONECELL during a simulation run.

Any variable in the external-state vector is available as output on the display scope. Thus if it is desired to monitor some particular concentration in each cell, a single FORTRAN statement can be used to put that concentration in the external-state vector for the course of the experiment. In running the simulation, one can display at any time any of the 11 external-state variables. While other forms of output are certainly possible, they have not been found necessary as yet, and have not been implemented.

Setting up a particular experiment involves two operations: 1) placing the appropriate variables in the external-state vector (for interaction among cells and for output); and 2) altering those equations in the transition-function which allow the interaction variables to influence the state-transition. All of this can be done in FORTRAN and is very simple to accomplish.

Potential Applications: The range of problems open for study using ONECELL + CELLSPACE is broad. It is important that the simulation provide data which can be compared with laboratory data in order to make testable the hypotheses being modelled. Thus we wish to simulate as many of the observable properties of a colony of cells as possible. Among the properties within the scope of the ONECELL + CELLSPACE simulation are:

1. average rates of cell-division in various environments,
2. differences in cell-division rates in various parts of the clone,
3. cell volumes as a function of nutrients and position,
4. radius and height (shape) of the colony,
5. concentrations of various diffusible and non-diffusible substances in different parts of the colony.

If we desire to study the effects of changes in internal biochemical-pathways within ONECELL, this can be largely accomplished by employing ONECELL alone. The great potential importance of the combined ONECELL + CELLSPACE systems instead lies in the study of the consequences of different conditions for interaction between cells. These interactions may be of several sorts, including:

1. reduction of access to metabolites because of intervening cells,
2. waste accumulation due to surrounding cells,
3. intracellular reactions specifically induced or altered by cell wall contact with, or crowding by, neighboring cells, and
4. intercellular diffusion or transport of chemical messengers, inhibiting or otherwise influencing cellular metabolism.

Interactions of types 3. and 4. above are of great interest to biologists at the present time, particularly in the study of the regulatory mechanisms which are altered in cancerous growth. For example, there may be a hypothesis that a particular chemical messenger, produced under certain circumstances, alters a particular biochemical pathway in cells into which it diffuses. One can simulate the action of the messenger, and observe its effects on the growth of a colony, either in terms of the con-

centrations of various chemical pools in each cell or in terms of gross characteristics of the colony. An important feature of this simulation system is the opportunity for the experimenter to make changes in the amount or even the type of effect caused by the variables under study *while the simulation is in progress*. Heuristic exploration may be carried out by altering parameters as the simulation progresses. By repeatedly making alterations and observing their effects, one may be able to find parameters which yield the desired behavior.

Preliminary Results: We shall present one experiment to illustrate the sorts of experiments which can be done with the system. The starting cell was initialized (in Link 1) for environment 1, called "minimal medium". An inhibition-function was programmed to act as follows: the cells in the colony were exposed to varying concentrations of a substance, call it INHIB, which affects the rate of production of IN, a protein which causes initiation of DNA replication upon accumulation of a certain amount of IN. The concentration of INHIB in each cell was made to depend upon the degree to which the cell was surrounded by other cells. Thus, cells on the outer periphery of the colony had relatively low concentrations in INHIB, while those in the center had relatively high concentrations. The effect of this inhibition was to change the entire character of the colony: without the inhibition it was a rapidly growing multi-layer colony; with the inhibition it became a slowly-growing, monolayer colony (see Figure 7.2 (b) - (e)). It became clear that this sort of change (from multi-layer to monolayer) is a function of both the amount of inhibition induced by neighboring cells, and also the underlying rate of growth as determined by the nutrients available in the simulated growth-medium.

It may seem at first surprising that the multi-layer "cancerlike" growth is the easier of the two growth patterns to implement in the cell-space; that is, our chief difficulty is to simulate the mechanisms which result in the more restricted "normal" growth. However, this situation has a parallel in biological research. Often, the more difficult problem in studying regulatory aberrations is to learn the exact nature of the normal controls; once this is known, it is much easier to study the control aberrations which give rise to pathological conditions.

It is expected that further experiments of this sort will render explicit many implications of the ONECELL model, the Helmstetter-Cooper replicon, and the sorts of inhibition which may be programmed into the CELLSPACE + ONECELL system.

Our CELLSPACE + ONECELL system should be able to simulate the development of a complete organism from simpler components. The basic principles of development are embodied in a twelve-cell slime mold. This primitive organism has a complete life cycle. The organism is the fruity mutant of *Dictyostelium discoideum*, and experimental data on its life cycle make it ideally suited for analysis by simulation.

Our present system of interacting living cells in different environments represents an ecosystem in miniature. Since we can define our system formally and compare the results of our simulation with experimental data from real cells growing in a petri dish, we can utilize our simulation for testing and formulating hypotheses about simple ecosystems. Since we are studying an ecosystem, some of our techniques are applicable to ecosystems in general, and may be used to analyze local models of interacting populations, such as groups of men and women struggling to survive in and enjoy their complex world.

CHAPTER VIII

CONCLUSIONS

Results of the Computer Simulation of a Living Cell:

1. The computer simulation of living cell is realistic and useful. The simulated cell reproduced the behavior of a real cell in fixed environments, and could realistically predict the behavior of a real cell in changing environments. The behavior of the simulated cell in changing environments was a good test of the simulation, since the ability to switch from one environment to another was not written into the simulation *ab initio*. The simulated cell permitted the study of interacting biochemical systems in one cell, and could be used to study interactions among populations of cells. The simulation techniques used provided an example of general modelling techniques.

2. Allosteric modification in real living cells is necessary for shifts up to richer media, while metabolic topology suffices in shifts down to poorer media.

Structure of the Simulation and Model:

1. Equivalence classes with substitution property is a useful concept, both for blocking out a simulation, and validating a working simulation.

2. System theory concepts suggest and justify techniques for valid reduction of complexity in models and computer simulations.

Evolution:

1. Evolutionary programs for artificial evolution of a population of simulated living cells has been outlined, comprising in essence a computer simulation of evolving DNA.

2. These evolutionary programs may be written for existing computers in FORTRAN. They select for desired characteristics in a population of simulated cells, even though such characteristics may involve nonlinear interactions of many primitive entities in each individual cell in the population of programs undergoing evolution. A formal proof of some of the advantages of the particular instance of the reproductive plan outlined for simulation appears in Holland (1969c).

Cell Space Embedding of Simulated Living Cells:

1. DNA replication in real living cells may be kept in phase, and turned on and off in a cell by less than five chemicals. These chemicals may communicate information from metabilic pathways in the cell to the DNA reproduction apparatus, and may also diffuse among neighboring cells to effect growth patterns in developing populations of cells.

2. The DNA replication apparatus used in the simulation predicted the production of cancer from non-cancer cells from a simple and analyzable biochemical event.

APPENDIX TO CHAPTER II

VARIABLES IN PROGRAM: A = array
0 = floating point
1 = integer

A2	A	0	arrays used in solve function to obtain rate constants used
A3	A	0	in allosteric inhibition
AAO		0	amino acid concentration at time zero
AAP		0	ATP molecules used to make 1 amino acid molecule
AAO		0	amino acid concentration
ADJST		0	adjustment factor for concentrations from volume increase
ADPO		0	ADP concentration at time 0
ADP		0	ADP concentration
ATPO		0	ATP concentration at time zero
ATP		0	ATP concentration
ATPSB	A	0	array to store ATP concentrations in different environments
BROTH		1	equals 1 if cell growing in broth
CAA		1	equals 1 if cell growing in casamino acid
CHRMO		0	number of chromosomes at time 0
CNTRL		1	equals 1 if cell using metabolic controls to adjust growth rate
COUNT		0	number of growth cycles made
CRAZY		1	used as a logical variable
C(I)	A	0	enzyme rate constants
DAA		0	change in amino acid concentration
DADP		0	change in ADP concentration
DAPO2		0	change in ATP concentration from literature
DATP		0	change in ATP concentration calculated from rate constants in one time step
DDNA1		0	change in amount chromosome 1 in one time increment
DDNA2		0	change in amount of chromosome 2 in one time increment
DDNA3		0	change in amount of chromosome 3 in one time increment
DDNA		0	change in total DNA in one time increment
DEK(1)		0	change in enzymes for nucleotide production in one time increment
DEK(2)		0	change in enzymes for amino acid production in one time increment
DEK(3)		0	change in enzymes for glycolysis production in one time increment
DEK(4)		0	change in enzymes for wall production in one time increment
DEK(5)		0	change in enzymes for ADP, ATP synthesis in one time increment
DEK(6)		0	change in enzymes for DNA synthesis in one time increment
DEK(7)		0	change in enzymes for protein production in one time increment
DEK(8)		0	change in enzymes for MRNA synthesis in one time increment
DEK(9)		0	change in enzymes for ribosome synthesis in one time increment

Appendix to Chapter II (Cont.)

DEK(10)		O	change in enzymes for TRNA production in one time increment
DIN		O	change in initiator concentration in one time increment
DNAO		O	DNA at time zero
DNA1		O	chromosome 1 "concentration", i.e., amount/volume of cell
DNA1Z		O	chromosome 1 at zero time
DNA1T		O	total chromosome 1
DNA2		O	chromosome 2 "concentration"
DNA2T		O	total chromosome 2
DNA2Z		O	chromosome 2 at zero time
DNA3		O	chromosome 3 "concentration"
DNA3T		O	total chromosome 3
DNA3Z		O	chromosome 3 at zero time
DNAP		O	ATP used per DNA molecule synthesized
DNA		O	DNA
DNASB	A	O	array to save concentrations of DNA in different environments
DNUC		O	change in nucleotide concentration
DBLE		O	time for cell to go through one reproductive cycle
DPRTN		O	change in protein in one time increment
DRIB		O	change in ribosome in one time increment
DMRNA		O	change in MRNA in one time increment
DRNA		O	change in total RNA in one time increment
DRNK(i)	A	O	change in MRNA for enzyme EK(i) in one time increment. i ranges from 1 to 10.
DT		O	length of one time increment, = differential
DUM1		O	dummy variable in solve function
DUM2		O	dummy variable in solve function
DUM3		O	dummy variable in solve function
DVOL		O	change in cell volume in one time increment
DWALL		O	change in cell membrane and cell wall in one time increment
DPRDK	A	O	array of change in product concentration in one time increment
PRD	A	O	the stored array of the previous four product values, for predictor corrector
DPRD	A	O	array of the four previous D(product) values for the predictor corrector
PPRD	A	O	current array of the predictor values of products
CPRD	A	O	current array of corrector values of products
EK(1)		O	concentration of enzymes for nucleotide production
EKZ(1)		O	concentration of enzymes for nucleotide production at zero time
EK(2)		O	concentration of enzymes for amino acid production
EKZ(2)		O	concentration of enzymes for amino acid production at zero time

.

.

.

Appendix to Chapter II (Cont.)

where 3 indicates glycolysis
4 indicates cell wall production
5 indicates ADP, ATP production
6 indicates DNA production
7 indicates protein production
8 indicates MRNA production
9 indicates ribosome production
10 indicates TRNA production

FACTR		0	factor by which chromosomes multiply in one reproductive cycle
GLUCO		0	glucose concentration at zero time
GLUC		0	glucose concentration
ID		1	integer variable in RPLACE routine
IN11		0	site for replication of chromosome 11, = 1 if it is present
IN11Z		0	site for replication of chromosome 11 at zero time
IN1		0	site for replication of chromosome 1, = 1 if it is present
IN1Z		0	site for replication of chromosome 1 at zero time
IN21		0	site for replication of chromosome 21
IN21Z		0	site for replication of chromosome 21 at zero time
IN2		0	site for replication of chromosome 2
IN2Z		0	site for replication of chromosome 2 at zero time
IN31		0	site for replication of chromosome 31
IN31Z		0	site for replication of chromosome 31 at zero time
IN3		0	site for replication of chromosome 3
IN3Z		0	site for replication of chromosome 3 at zero time
IN		0	concentration of initiator in cytoplasm
II		1	an integer variable
INZ		0	initiator concentration at zero time
K(1)		0	preliminary rate constant for nucleotide production
K(2)		0	preliminary rate constant for amino acid production
K(3)		0	preliminary rate constant for glycolysis
K(4)		0	preliminary rate constant for cell wall production
K(5)		0	preliminary rate constant for ADP production
K(6)		0	preliminary rate constant for DNA production
K(7)		0	preliminary rate constant for protein production
K(8)		0	preliminary rate constant for MRNA production
K(9)		0	preliminary rate constant for ribosome production
K(10)		0	preliminary rate constant for TRNA production
K(14)		0	preliminary rate constant for volume increase as a function of wall
KDRNK		0	rate constant for MRNA decay
K8K(i)	A	0	rate constant for MRNA EK(i)
K8KZ(i)	A	0	rate constant for MRNA for EKZ(i)
KBB(i)	A	0	rate constant for allosterically inhibited enzyme EK(i) with two molecules of product attached to the enzyme
KB	A	0	array of rate constants of allosterically inhibited enzymes with one molecule of product attached to the enzyme

Appendix to Chapter II (Cont.)

KIN		0	preliminary rate constant for initiator production
K8K(i)	A	0	rate constant for production of EK(i)
KK(i)	A	0	rate constant for uninhibited enzyme EK(i)
K(i)	A	0	array to store preliminary rate constants, used for each environment
LN2		0	natural logarithm of 2
L		1	integer variable for calling on solve function
M		1	integer variable for printing loop
MRNAO		0	MRNA concentration at time zero
MRNAP		0	ATP per MRNA molecule produced
MRNA		0	MRNA concentration
MULT		0	number of genes producing initiator
NO		0	number of cell in population (doubles when cell divides)
NUCO		0	molecules of nucleotide at zero time
NUCP		0	molecules of ATP to make one nucleotide
NUC		0	concentration of nucleotide
P1		0	rate constant
P1V	A	0	array of equilibrium rate constants for enzymes
P3		0	equilibrium rate constant for two molecule allosteric inhibition
P3V	A	0	array of equilibrium rate constants for two molecule allosteric inhibition
PRDCO	A	0	array equivalenced to products at zero time
PRDCK	A	0	array equivalenced to products
PRDC	A	0	array for storing concentrations of products in different environments
PRDC(1)		0	NUC
PRDC(2)		0	AA
PRDC(3)		0	ATP
PRDC(4)		0	WALL
PRDC(5)		0	ADP
PRDC(6)		0	DNA
PRDC(7)		0	PRTN
PRDC(8)		0	MRNA
PRDC(9)		0	RIB
PRDC(10)		0	TRNA
PRDC(11)		0	GLUC
PRDC(14)		0	VOL
PRTNO		0	protein concentration at zero time
PRTNP		0	ATP molecules used per protein molecule formed
PRTN		0	protein concentration
RAA		0	ratio of amino acid concentration to a base level
RADP		0	ratio of ADP concentration to a base level
RATP		0	ratio of ATP concentration to a base level

Appendix to Chapter II (Cont.)

RC	A	0	array of repression constants for MRNA repression
RDNA1		0	ratio of chromosome 1 concentration to a base level
RDNA2		0	ratio of chromosome 2 concentration to a base level
RDNA		0	ratio of DNA concentration to a base level
REK(i)	A	0	ratio of EK(i) concentration to a base level, i = 1,...,10
RIBO		0	ribosome concentration at time zero
RIBP		0	ATP used per ribosome made
RIB		0	ribosome concentration
RNAO		0	RNA concentration at time zero
RNA		0	RNA concentration
TRNAO		0	transfer RNA concentration at time zero
TRNAP		0	ATP per transfer RNA molecule made
TRNA		0	transfer RNA concentration
RNK(i)	A	0	concentration of MRNA for enzyme EK(i), i = 1,...,10
RNKZ(i)	A	0	concentration at zero time of MRNA for EKZ(i), i = 1,...,10
RNUC		0	ratio of nucleotide concentration to a base level
RON		1	used as a logical variable turning repression on
RPRTN		0	ratio of protein concentration to a base level
RRIB		0	ratio of ribosome concentration to a base level
RMRNA		0	ratio of MRNA concentration to a base level
RRNA		0	ratio of RNA concentration to a base level
RTRNA		0	ratio of TRNA concentration to a base level
RRNK(i)	A	0	ratio of RNK(i) concentration to a base level, i = 1,...,10
R	A	0	array for repression constants
RVOL		0	ratio of new volume to old volume at end of one time increment
RWALL		0	ratio of pool for wall to a base level in terms of concentration
SUM	A	0	array used in solve function
T		0	generation time in seconds
VOLO		0	volume of cell at time zero
VOLN		0	volume at end of one time increment
VOL		0	volume
WALLO		0	concentration of pool for wall production at time zero
WALLP		0	ATP molecules used per molecule of cell wall produced
WALL		0	concentration of pool for wall production
X		0	variable used in repression routine
XK(i,j)	A	0	value of K(i) in environment (j)
XEK(i,j)	A	0	value of EK(k) in environment (j)
XK8(i,j)	A	0	value of K8K(i) in environment (j)

BIBLIOGRAPHY

ACM Symposium on Theory of Computing, Marina del Rey, California, 1969.

Arbib, M. A. *Theories of Abstract Automata*, Englewood Cliffs, New Jersey: Prentice Hall, Inc. 1969.

Atkinson, D. E. "Regulation of Enzyme Activity." *Annual Rev. Biochem.* 35, 85-123, 1966.

Bernhard, Sidney. *The Structure and Function of Enzymes*, New York: Benjamin, 1968.

Bono, P. R. "A Heuristic Program Which Produces Generative Grammars." Ann Arbor, Michigan: Project for Course in Simulation of Biological Systems, CCS 680, The University of Michigan, 1968.

Boyer, H.; Englesberg, E.; and Weinberg, R. "Direct Selection of L-arabinose Negative Mutants of *Escherichia coli* Strain B/r." *Genetics*, 47, 412-425, 1962.

Brender, R. F. "A Programming System for the Simulation of Cellular Spaces." Ph.D. Dissertation, University of Michigan, 1970.

Britten, F. J., and Davidson, E. H. "Gene Regulation for Higher Cells: A Theory." *Science*, 165, 349-357, 1969.

Burks, A. W., (ed.) *Papers on Cellular Automata*, Urbana, Illinois: University of Illinois Press, in press.

Cavicchio, D. J., Jr. "A Heuristic Program Which Recognizes Patterns." Ann Arbor, Michigan: Project for Course in Simulation of Biological Systems, CCS 680, The University of Michigan, 1968.

Cennamo, C. "Stead-state Kinetics of One-substrate Enzymic Mechanisms Involving Two Enzyme Conformation II. Kinetic Treatment of a Reversible Mechanism: Effects of Modifiers and Product Inhibition." *Journal of Theoretical Biology*, 23, 53-71, 1969.

Chance, B.; Estabrook, R. W.; and Williamson, J. R. (eds.) *Control of Energy Metabolism*, New York: Academic Press, 1965.

Clark, J. D. "Regulation of Deoxyribonucleic Acid Replication and Cell Division in *Escherichia coli* B/r." *Journal of Bacteriology*, 96, 1214-1224, 1968.

Codd, E. F. *Cellular Automata*, New York: Academic Press, 1968.

Comings, D. E. and Kakefuda, T. "Initiation of Deoxyribonucleic Acid Replication at the Nuclear Membrane in Human Cells." *Journal of Molecular Biology*, 33, 225-562, 1969.

Datta, P. "Regulation of Branched Biosynthetic Pathways in Bacteria." *Science*, 165, 556-562, 1969.

Davis, B. D.; Dulbecco, R.; Eisen, H. N.; Ginsberg, H. S.; and Wood, W. B. *Microbiology*, New York: Harper & Row, 1968.

Eberle, H. and Lark, K. G. "Relation of the Segregative Origin of Chromosome Replication to the Origin of Replication After Amino Acid Starvation." *Journal of Bacteriology*, 98, 536-542, 1969.

Englesberg, E.; Anderson, R. L.; Weinberg, R.; Lee, N.; Hoffee, P.; Huttenhauer, G.; and Boyer, H. "L-arabinose Sensitive, L-ribulose 5-phosphate 4-epimerase Deficient Mutants of *Escherichia coli*." *Journal of Bacteriology*, 84, 137-164, 1962.

Fein, L. "Biological Investigations by Information Processing Simulations." *Natural Automata and Useful Simulations*, New York: Spartan Books, 181-201, 1966.

Fisher, R. A. *The Genetical Theory of Natural Selection*, New York: Dover, 1958.

Frisch, Leonora, (ed.) *Cellular Regulatory Mechanisms*, Cold Spring Harbor Symposia on Quantitative Biology, Vol. 33, 1961.

___, *Replication of DNA in Micro-organisms*, Cold Spring Harbor Symposia on Quantitative Biology, Vol. 33, 1969.

Gale, D. "A Geometric Duality Theorem with Economic Application." *Review of Economic Studies*, 32, 19-24, 1967.

Garfinkel, D. "A Simulation Study of Mammalian Phosphofructokinase." *Journal of Biol. Chem.*, 241, 286-294, 1966.

Gause, G. F. *Microbial Models of Cancer Cells*, Philadelphia: Saunders, 1966.

Ginzburg, A. *Algebraic Theory of Automata*, New York: Academic Press, 1968.

Goodman, Erik D.; Weinberg, R.; and Laing, R. A. "A Cell Space Embedding of Simulated Living Cells." *1970 Summer Computer Simulation Conference, Sponsored by ACM, IEEE, SHARE, and SCI*, Denver, Colorado, 1970.

Goodwin, B. C. "A Statistical Mechanics of Temporal Organization in Cells." *Society for Experimental Biology Symposium*, 18, 301-326, 1965.

Gordon, Geoffrey. *System Simulation*, Englewood Cliffs, New Jersey: Prentice-Hall, 1969.

Green, H. A. J. *Aggregation in Economic Analysis*, Princeton, New Jersey: Princeton University Press, 1964.

Griffith, J. S. "Mathematics of Cellular Control Processes." *Journal of Theoretical Biology*, 20, 202-216, 1968.

Harary, F.; Norman, D.; and Cartwright, R. *Structural Models: An Introduction to the Theory of Directed Graphs*, New York: John Wiley & Sons, 1965.

Hartmanis, J. and Stearns, R. E. *Algebraic Structure Theory of Sequential Machines*, Englewood Cliffs, New Jersey: Prentice-Hall, 1966.

Hayes, W. *The Genetics of Bacteria and Their Viruses*, 2nd edition. New York: John Wiley & Sons, 1968.

Hess, B. "Biochemical Regulations." in *Systems Theory and Biology*. Edited by M. Mesarovich. Pages 88-114, New York: Springer-Verlag, 1968.

Heinmets, F. "Analog Computer Analysis of a Model-System for the Induced Enzyme Synthesis." *Journal of Theoretical Biology*, 6, 69-75, 1964.

___, *Analysis of Normal and Abnormal Cell Growth*, New York: Plenum, 1966.

Helmstetter, C. E.; Cooper, S.; Pierucci, O.; Revelas, E. "On The Bacterial Life Sequence." *Cold Spring Harbor Symposia on Quantitative Biology*, 23, 809-822, 1968.

Hildebrand, F. B. *Introduction to Numerical Analysis*, New York: McGraw-Hill, 1956.

Helling, R. B. and Weinberg, R. "Complementation Studies of Arabinose Genes in *Escherichia coli*." *Genetics*, 48, 1397-1410, 1963.

Hoffee, P.; Weinberg, R. and Englesberg, E. "Absence of the Glucose Effect on the Frequency of Lysogenization in *Salmonella typhimurium*." *Nature*, 189, 160-161, 1961.

Holland, J. H. "Hierarchical Descriptions, Universal Spaces and Adaptive Systems." *Essays on Cellular Automata*. Edited by A. W. Burks. Urbana, Illinois: University of Illinois Press (in press). 1969a.

___, "Adaptive Plans Optimal for Payoff only Environments." in *Proceedings of the Second Hawaii Conference on System Sciences*, 1969b.

___, "A New Kind of Turnpike Theorem." *Bulletin of the American Mathematical Society*, 75, 1311-1317, 1969c.

IBM Corporation. *System/360 Scientific Subroutine Package (360-CM-03X) Version III Programmer's Manual*, New York: IBM, 1969.

Ijiri, Y. "The Linear Aggregation Coefficient as the Dual of the Linear Correlation Coefficient." *Econometrica*, Vol. 36, No. 2, 1968.

Kalman, R. E.; Falb, P. L.; Arbib, M. A. *Topics in Mathematical Systems Theory*, New York: McGraw-Hill, 1969.

Kauffman, S. A. "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets." *Journal of Theoretical Biology*, 22, 437-467, 1969.

Kimura, M. "Changes of Mean Fitness in Random Mating Populations when Epistasis and Linkage are Present." *Genetics*, 51, 349-363, 1965.

Klir, J. and Valach, M. *Cybernetic Modelling*, Princeton, New Jersey: Van Nostrand, 1967.

Knuth, Donald E. *The Art of Computer Programming, Volume 2, Semi-numerical Algorithms*, Massachusetts: Addison Wesley, 1969.

Koch, A. L. "Metabolic Control Through Reflexive Enzyme Action." *Journal of Theoretical Biology*, 15, 75-102, 1967.

Krohn, K. and Rhodes, J. "Algebraic Theory of Machines." *Transactions of the American Mathematical Society*, Vol. 116, 450-464, 1965.

Lark, K. G. "Initiation and Control of DNA Synthesis." *Annual Rev. Biochem.*, 38, 549-604, 1969.

Maaløe, O. and Kjeldgaard, N. O. *Control of Macromolecular Synthesis*, New York: Benjamin, 1966.

Mahler, H. R. and Cordes, E. H. *Biological Chemistry*, New York: Harper & Row, 1966.

Mandelstam, J. and McQuillen, K. (ed.) *Biochemistry of Bacterial Growth*, New York: John Wiley & Sons, 1968.

Mesarovic, M. "Auxiliary Functions and Constructive Specification of General Systems." *Math. Systems Theory*, 2, 203-222, 1968.

___, (ed.) *Systems Theory and Biology*, New York: Springer-Verlag, 1968.

Mitchison, J. M. "Enzyme Synthesis in Synchronous Cultures." *Science*, 165, 657-663, 1969.

Mize, J. H. and Cox, J. G. *Essentials of Simulation*, Englewood Cliffs, New Jersey: Prentice-Hall, 1968.

Moser, H. *The Dynamics of Bacterial Populations Maintained in the Chemostat*, Washington, D. C.: Carnegie Institution of Washington Publication 614, 1958.

Murray, A. W. and Atkinson, M. R. "Adosine 5' Phosphorothioate. A Nucleotide Analog that is a Substrate, Competitive Inhibitor, or Regulator of Some Enzymes that Interact with Adenosine 5'-Phosphate." *Biochemistry*, 7, 4023-4029, 1968.

Reiner, J. M. *The organism as an Adaptive Control System*, Englewood Cliffs, New Jersey: Prentice-Hall, 1968.

___, *Behavior of Enzyme Systems*, New York: Reinhold, 1969.

Roosen Runge, P. "Algebraic Description of Access and Control in Information Processing Systems." Ph.D. Dissertation, The University of Michigan, 1967.

Rosen, R. "Some Comments on the Physico-Chemical Description of Biological Activity." *Journal of Theoretical Biology*, 18, 380-386, 1968.

Savageau, M. A. "Biochemical Systems Analysis." *Journal of Theoretical Biology*, 25, 365-369, 1969.

___, "Biochemical Systems Analysis III. Dynamic Solutions Using a Power-law Approximation." *Journal of Theoretical Biology*, 26, 215-226, 1970.

Schultz, J. S. and Gerhardt, P. "Dialysis Culture of Microorganisms: Design, Theory, and Results." *Bacteriological Reviews*, 33, 1-47, 1969.

Simon, H. A. "The Architecture of Complexity." *Proc. American Philosophic Society*, Vol. 106, 6, 1962.

Simon, H. A. and Ando, H. "Aggregation of Variables in Dynamic Systems." *Econometrica*, 29, 111-138, 1961.

Smith, H. S. and Pardee, A. B. "Accumulation of a Protein Required for Division During the Cell Cycle of *Escherichia coli*." *Journal of Bacteriology*, 101, 901-909, 1970.

Stahl, W. R. "A Computer Model of Cellular Self-Reproduction." *Journal of Theoretical Biology*, 14, 187-205, 1967.

Strickberger, M. W. *Genetics*, New York: Macmillan, 1968.

Sugita, M. and Fukuda, N. "Functional Analysis of Chemical Systems in vivo Using a Logical Circuit Equivalent." *Journal of Theoretical Biology*, 5, 412-425, 1968.

Sussman, Maurice. *Growth and Development*, Englewood Cliffs, New Jersey: Prentice-Hall, 1964.

Tsanev, R. and Sendov, B. "A Model of Cancer Studied by a Computer." *Journal of Theoretical Biology*, 23, 124-134, 1969.

Ulam, S. M. "On General Formulations of Simulation and Model Construction." in *Prospects for Simulation and Simulators of Dynamic Systems*. Edited by G. Shapiro and M. Rogers. New York: Spartan, 1966.

Umbarger, H. E. "Regulation of Amino Acid Metabolism." *Annual Rev. Biochem.* 38, 323-370, 1969.

von Neumann, J. *Theory of Self-Reproducing Automata*. Edited by A. W. Burks. Urbana, Illinois: University of Illinois, 1966.

Waddington, C. H. *Towards a Theoretical Biology*, Birmingham, Great Britain: Kynoch, 1969.

Wallace, B. *Topics in Population Genetics*, New York: W. W. Norton, 1968

Walter, Charles "Stability of Controlled Biological Systems." *Journal of Theoretical Biology*, 23, 23-38, 1969a.

___, "The Absolute Stability of Certain Types of Controlled Biological Systems." *Journal of Theoretical Biology*, 23, 39-52, 1969b.

Weinberg, Roger "The Chromosomes of *Drosophila macrospina* with Comparisons of The Chromosome Elements with Other Species." *University of Texas Publication 5422*, 153-162, 1954.

___, "Apparent Mutagenic Effect of Thymine Deficiency for a Thymine-requiring Strain of *Escherichia coli*." *Journal of Bacteriology*, 72, 570-572, 1956.

___, "Phage T1-resistance Mutants of *Escherichia coli*." *Journal of Bacteriology*, 79, 612-614, 1960.

___, "Gene Transfer and Elimination in Bacterial Crosses with Strain K-12 of *Escherichia coli.*" *Journal of Bacteriology*, 79, 558-563, 1960.

___, "Analytic and Logical Equations in a Computer Simulation of Cell Metabolism and Replication." *Sixth Annual Symposium on Biomathematics and Computer Science in the Life Sciences*, The University of Texas, 102-103, 1968a.

___, "Computer Simulation of a Living Cell." *Bacteriological Proceedings*, G114, 1968b.

___, "Computer Simulation of Self-Reproduction by a Living Cell." *Genetics*, 60, 235, 1968c.

Weinberg, R. and Boyer, H. "Base Analogue Induced Arabinose Negative Mutants of *Escherichia coli.*" *Genetics*, 51, 545-553, 1965.

Weinberg, R. and Berkus, M. "Computer Simulation of Evolving DNA." *Biometrics*, 25, 447, 1969a.

___, "Computer Simulation of a Living Cell." Ann Arbor, Michigan: The University of Michigan, Technical Report 01252-2-T, 1969b.

Weinberg, R. and Zeigler, B. P. "Computer Simulation of a Living Cell: Multilevel Control Systems." *Communications of the American Society for Cybernetics*, (to be published) 1970.

Wendroff, B. *First Principles of Numerical Analysis*, Massachusetts: Addison-Wesley, 1969.

Westley, John *Enzymic Catalysis*, New York: Harper & Row, 1969.

Wright, Barbara E. "Differentiation in the Cellular Slime Mold." in *Proceedings of the Third Systems Symposium* at Case Institute of Technology. Edited by M. D. Mesarovich. New York: Springer-Verlag, 1968.

Yeisley, W. G. and Pollard, E. C. "An Analog Computer Study of Differential Equations Concerned with Bacterial Cell Synthesis." *Journal of Theoretical Biology*, 7, 485-501, 1964.

Zeigler, B. P. "On the Feedback Complexity of Automata." Ph.D. Dissertation, University of Michigan, 1968.

___, "On the Feedback Complexity of Automata." *Proc. Third Annual Princeton Conference on Systems and Information Sciences*, Princeton, New Jersey, 1969.

___, "On the Formulation of Problems in Simulation and Modelling in the Framework of Mathematical System Theory." *VIth International Congress on Cybernetics*, Namur, Belgium, 1970.

Zeigler, B. P. and Weinberg, R. "System Theoretic Analysis of Models: Computer Simulation of a Living Cell." *Journal of Theoretical Biology*, (to be published) 1970.