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

Technical Report

COMPUTER SIMULATION OF A LIVING CELL

Roger Weinberg

and

Michael Berkus

supported by:

Department of Health, Education, and Welfare
National Institutes of Health
Grant No. GM-12236-03
Bethesda, Maryland

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

May 1969

ABSTRACT

COMPUTER AND SIMULATION OF A LIVING CELL

by

Roger Weinberg and Michael Berkus

Area

Computer simulation of living adaptive systems.

Problem

Write a realistic and useful computer simulation of a living cell.

Method of Inquiry

Simulate the simple unicellular bacterium Escherichia coli as a program in FORTRAN IV for an IBM/360:67 digital computer.

Criteria for Success

I. Reality of the Simulation

   a. The simulated cell should realistically simulate growth curves of a living cell in the real environments used to adjust the parameters in the simulation.

   b. The simulated cell should realistically simulate growth during changes from one chemical environment to another.

   c. The simulated cell should predict real cell behavior in environments not used to adjust the parameters in the simulation.

II. Future Usefulness of the Simulation

   a. The simulation should supply unique information useful in answering current questions in biology.

      1. The simulation should help to analyze interactions between metabolic pathways and cellular control mechanisms. It should

accomplish this by its ability to predict data from the hypotheses incorporated in the program, thereby permitting tests of these hypotheses against experimental data in the literature.

2. The simulation should add information to the literature about the concentrations of major cell constituents during complex changes in environmental conditions.
3. The simulation should be useful for theoretical studies not amenable to direct experimental analysis in the biological laboratory. An example would be to study different effects on metabolism by simulating genetic and metabolic conditions necessary for the existence of new biochemical pathways in cell metabolism.
4. It should be possible to outline a simulation of an evolving population of living cells.

ACKNOWLEDGEMENTS

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. Dr. Bernard P. Zeigler, E. Stewart Bainbridge, and Daniel J. Cavicchio 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 and Richard Laing catalyzed my work with their editorial comments and creative writing. Mr. Thomas Dawson administered the affairs of the Logic of Computers Group, and the Department of Computer and Communication Sciences with morale boosting zeal. Miss Linda Beattie typed and drew with skill and dedication. I have erred at points in spite of all of these accomplished and generous helpers because I am human.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Cont.)

CHAPTER I

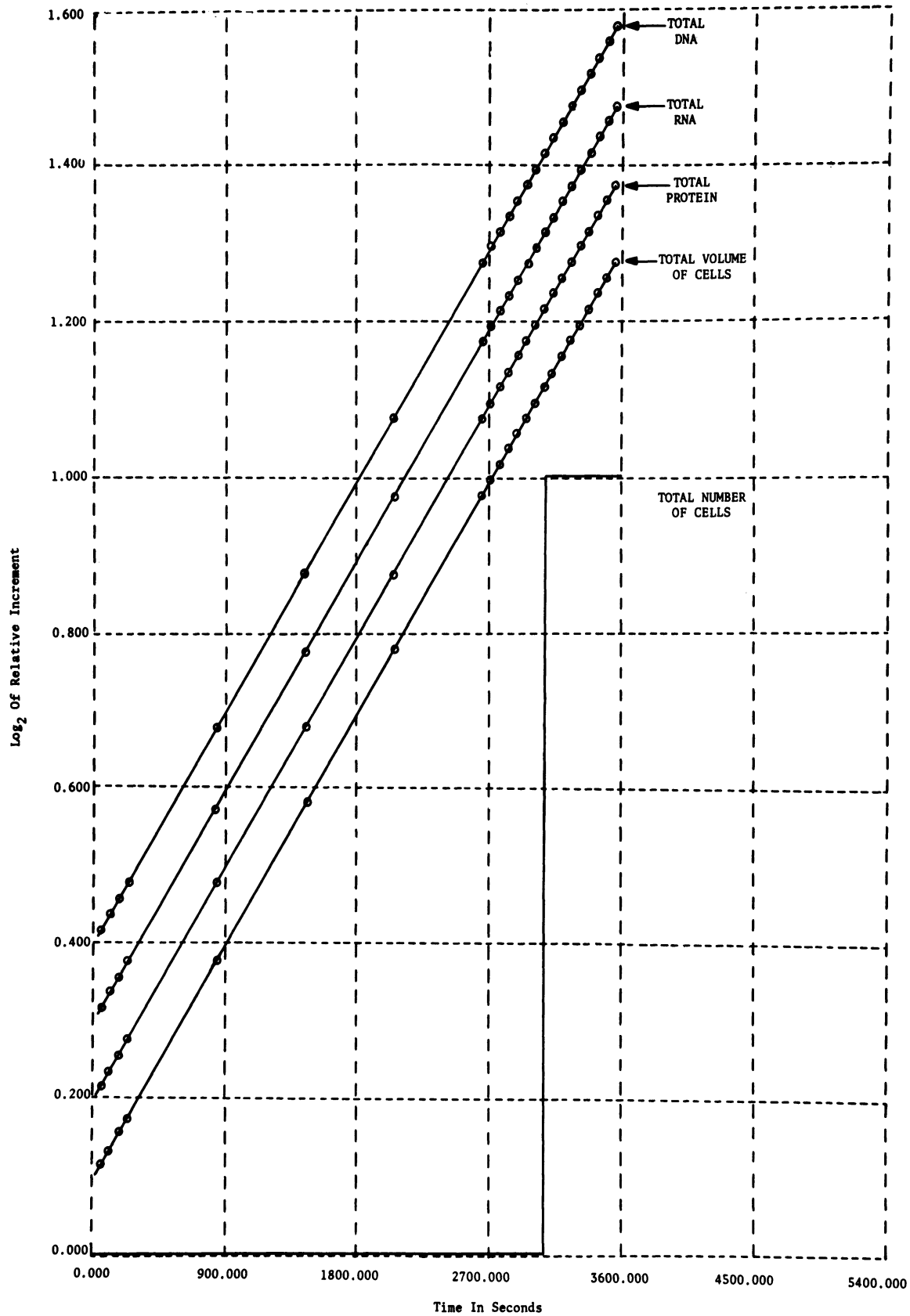
CRITERIA FOR SUCCESS

1.1. Reality of the Simulation

a) The simulated cell has realistically simulated growth curves of a living cell in the real environments used to adjust the parameters in the simulation. Since a vast amount of experimental data on E. coli is available in the literature, it was easy to check the behavior of the simulated cell against data from the real world. The simulated cell produced chemicals and cell mass at logarithmic rates, and duplicated in stepwise fashion, just as the real cell does (Figure 1.1). Since the simulated cell produced these realistic growth curves from a complex interaction of many equations, the growth curves are a good preliminary confirmation of the simulation.

b) The simulated cell should realistically simulate growth during changes from one chemical environment to another. Since the control equations were fitted for growth in different fixed environments, but not for behavior during a shift in environmental conditions, the behavior of the simulated cell during a change in its environment comprises a test of new conditions not used to adjust the parameters in the simulation. Although the simulated cell could adapt to changes in its environment, further work is necessary to resolve artificial fluctuations in ATP and ADP concentrations following the shift.

c) The simulated cell should predict real cell behavior in environments not used to adjust the parameters in the simulation. The most stringent test would be a comparison with an experimental situation in which the pools represented in the simulation are available as experimental measurements with which the simulation must agree. Growth in a new, completely defined



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", where as "TOTAL DNA" increases throughout the replication cycle.

Figure 1.1. Logarithm of Various Quantities in a Growing Culture.

chemical medium by the real cell will be used as the data with which the simulated cell must agree. The response of the program changes both in different simulated environments, and as a function of modifications of the hypotheses used to formulate the equations in the program. This output response serves as data for analyzing the interactions occurring among the hypothetical subsystems on which the program is based. Interesting and valuable works have been programmed for solving theoretical mathematical equations in the study of feedback controls of individual pathways in metabolism (Koch, 1966; Garfinkel, 1964) and to study repression or induction of one gene (Griffith, 1968). Garfinkel has simulated the glycolytic pathway in elegant detail for higher protists. Heinmetz (1964) has solved differential equations on an analog computer in order to study repression of one enzyme. I am synthesizing in one program for a digital computer interacting systems important and necessary for replication and growth of a real, specific living cell, Escherichia coli (Weinberg, 1968a, b, c; Weinberg and Berkus, 1969). My simulation includes homomorphic representations of many biochemical pathways, gene controls, and mechanisms for cell replication, permitting studies of interactions of these vital organizational entities in ways not intended or possible with simulations limited to at most a few metabolic pathways (e.g., Garfinkel, 1964).

Simulations have been written for abstract systems which resemble living cells in their general organization. These simulations were not models of any particular cell, and were not designed so that the output of the simulation could be checked against experimental data (Heinmetz, 1966; Yeisley and Pollard, 1964; Tsanev and Sendov, 1967; Stahl, 1967). Yeisley and Pollard have simulated production of metabolic intermediates, emphasizing organization of sequences rather than including feedback, repression, and interaction of

metabolic pathways to control enzyme activity and production. Heinmetz has simulated a general model for an abstract cell on an analog computer with no attempt at comparison of the results of the simulation with data from the real world (1966). Tsanev has simulated an abstract system for regulation of multiplication of a diploid cell, again with emphasis on making a workable model rather than on comparison of predictions of the simulation with real experimental data. Although simulating a real cell requires careful literature searches, and painstaking accuracy as to metabolic topology and concentrations, my realistic simulation of E. coli can be used in ways neither envisioned nor accessible for the abstract models mentioned. I can predict experimental results for a real organisms, and can improve my simulation as I use it by correcting discrepancies between the simulation's predictions and real experimental results.

1.2. Future Usefulness of the Computer Simulation of a Living Cell

a) I will try to show that the simulated living cell is useful by demonstrating that it can by applied to current questions in biology. I will choose questions concerning interrelationships among metabolic pathways and control mechanisms which are important in the establishment of a realistic and stable simulation. The answers to the questions will therefore serve the two-fold purpose of illustrating the simulation's usefulness, and making it even more useful.

1) Typical questions relating to the interactions among metabolic pathways and cellular control mechanisms are: Can the simulated ATP/ADP controls, along with the simulated operon, replicon, and allosteric inhibition account for the stability of a real cell in changing environments? Is the simulated cell's stability lost when the ADP/ATP controls are relaxed? Does charged and uncharged TRNA as a control on protein

synthesis lead to simulated results corresponding to the difference between normal cells and relaxed mutants?

2) The simulation should add information about the concentrations of major cell constituents during complex changes in environmental conditions. Since all of the twenty-two pools used to represent metabolism are represented at all times during the simulation, the concentrations of these pools will be calculated in every program run. Many of these concentrations are not measurable in some of the relevant biological experiments, although the concentrations may influence important factors being studied. Availability of realistic approximations of all twenty-two concentrations during many different conditions will be useful for theoretical analyses of metabolism.

3) The simulation should be useful for theoretical studies not amenable to direct experimental analysis in the biological laboratory. Adding new metabolic pathways, or changing the efficiency of old metabolic pathways is easily implemented in the simulation, with automatic prediction of the concentrations of the twenty-two pools of biochemicals represented. Viability of the simulated cell may be defined as the ability to duplicate at a rate similar to the duplication rate of a real cell in a similar environment. Many possible alterations can be analyzed as to effect, and some chosen for their feasibility for experimental analysis using criteria such as the viability of the altered cell, or the simplicity of the alteration in terms of genetic changes necessary to produce the alteration.

4) The computer simulation should be a working simulation of a living cell with evolutionary potentialities for the simulation as well as for the model being simulated.

Living organisms adapt to their environments in two ways; during evolution and at the level of the individual organism. At the individual level, an organism adapts over its own life span by changing its phenotype in response to information it receives from its external and internal environments. The simulated cell accomplished this type of phenotypic adaptation when it maintained stability in changing simulated chemical environments (Weinberg, 1968a, b, c).

In an evolutionary sense, populations of organisms adapt over many cycles of reproduction, with more fit organisms selected for greater ability to survive and to produce offspring. A scheme for evolutionary adaptation of the simulated cell will be outlined (Weinberg and Berkus, 1969), and its validity supported by theoretical as well as practical considerations (Holland, 1968a, b; 1969a, b). DNA of a cell can be represented by an array in which is stored information concerning the various rate constants in the equations representing a simulated cell. Evolving DNA can be simulated by modifying arrays representing the DNA of different cells in a population. The evolutionary operators crossing over, inversion, mutation and selection can be simulated by a genetic program written to modify the contents of the DNA arrays. Fundamental theories of population genetics by Fisher, Kimura and Holland (1968a, b; 1969a,b) assure one of the effectiveness of this type of modification in achieving desired goals. An example of a startling and interesting prediction arising from Holland's theory is the global optimization available to the reproductive plan through the use of local optimization techniques. The theorem makes heavy use of a series of developments in economics (Gale, 1967). Not only does a reproductive plan exist for such an accomplishment, but many different goals may be reached by using the same plan over much of the evolutionary path. The turnpike theorems from economics which

Holland relates to reproductive plans indicate the existence of such robust plans. The extraordinarily efficient sampling and search technique of natural populations becomes apparent within the framework of Holland's formal adaptive system. This efficient sampling technique has profound implications as to the kinds of task possible to an evolving population.

CHAPTER II

COMPUTER SIMULATION OF A LIVING CELL

We are trying to write a realistic and useful computer simulation of a living cell. The enormous advances in molecular biology in the past fifteen years have greatly increased our understanding of the basic subsystems in the living cell. 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 the information to be passed on 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. Control mechanisms have also been discovered which control a given cellular process by modifications of molecules far removed from DNA. Biology is entering a new phase. The interactions of the various subsystems analyzed by the molecular biologists are becoming objects of study. These subsystems interact among themselves, and with the environment external to the cell. Current research on regulation of DNA replication (Clark, 1968) and regulation of enzyme activity (Atkinson, 1966; Murray and Atkinson, 1968), as well as numerous experimental and theoretical studies of molecular interactions within the cell (Davis et. al., 1968) have made it possible to model a living cell in which both metabolism and cell replication are represented.

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 living cell, the program generates cell behavior as output.

In order to analyze interactions among DNA, RNA, protein and the activities of self-reproduction and metabolism in a living cell, a simulation of the simple unicellular bacterium Escherichia coli was written as a program in FORTRAN IV for an IBM/360:67 digital computer (Weinberg and Berkus, 1969). The cell was represented in terms of (1) the concentrations of twenty-two pools of chemicals (Figure 2.1) and (2) equations representing functional relationships among pools (Figures 2.2, and 2.3). These equations were used to predict the change in concentration and total amount of chemicals in each pool over time (Figures 2.3, 2.4, and 2.5).

Homomorphic mappings are often used from the real world onto the models used for computer simulations. One must be aware of the approximate nature of these homomorphisms, and consequent limitations on the predictive ability of the simulation. An example of a domain for such a mapping would be the twenty-two pools of chemicals in the simulation of a living cell used to represent corresponding pools in a real living cell. I hoped that these pools would create equivalence classes of chemicals with substitution property over time. The substitution property is usually approximate, and limits the domain in which the predictions of the simulation are accurate. This approximation does not necessarily invalidate the model, but should be kept in mind while using it. One can often use imperfect predictions from imperfect models in a Bayesian sense, and work with the new probabilities one obtains to select possible courses of action.

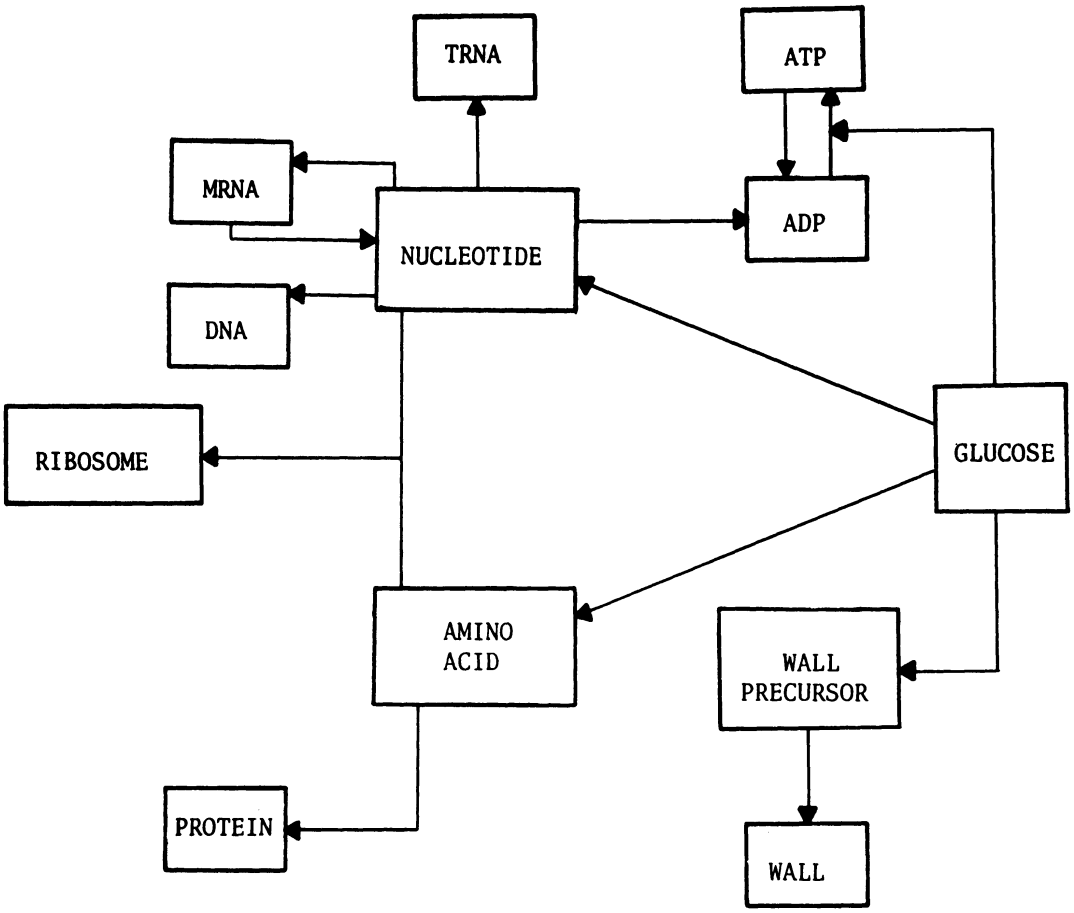


Figure. 2.1. Flow of Materials.

```
   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 2.2

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 2.3. 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.

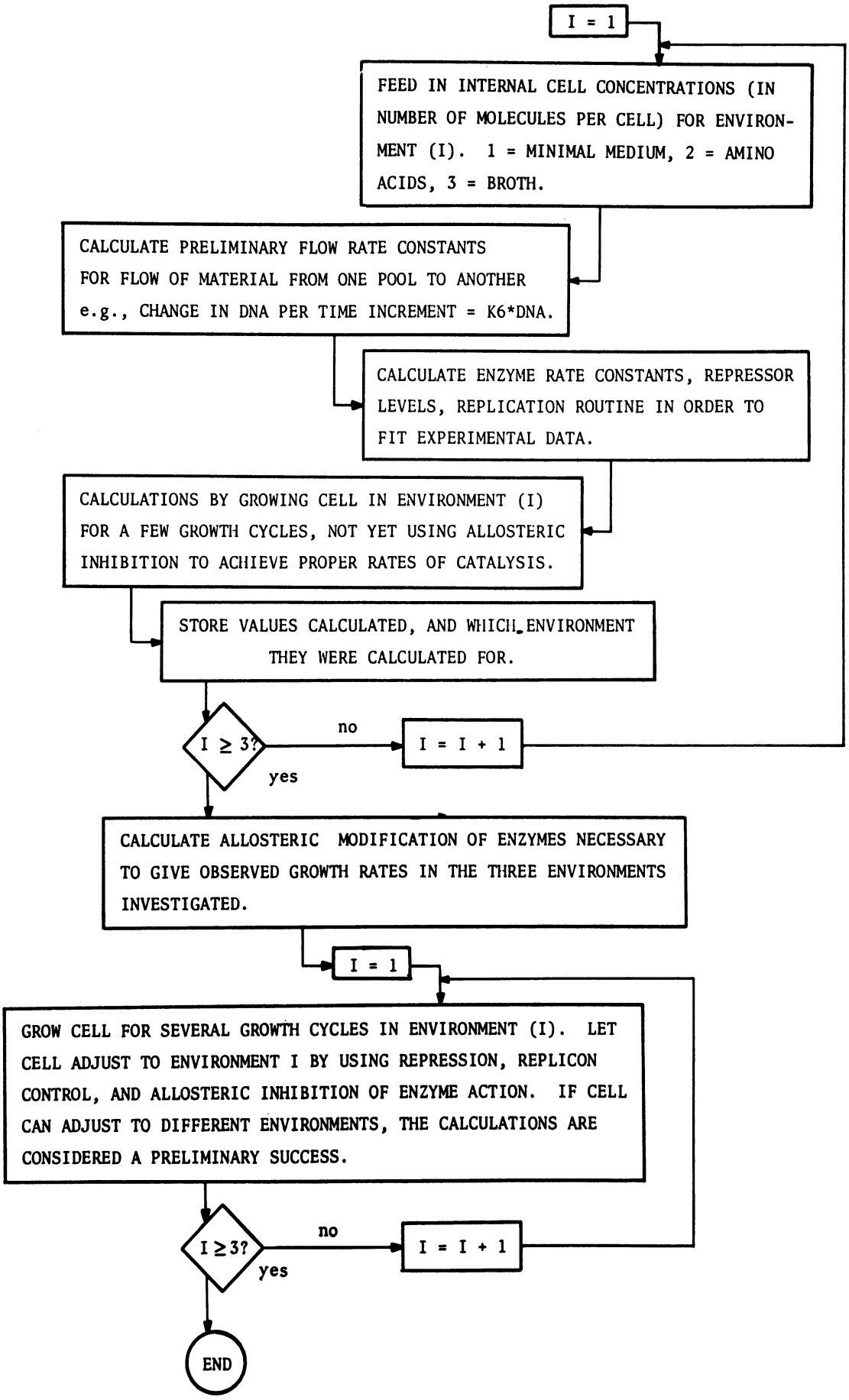


Figure 2.4. Summary of Program.

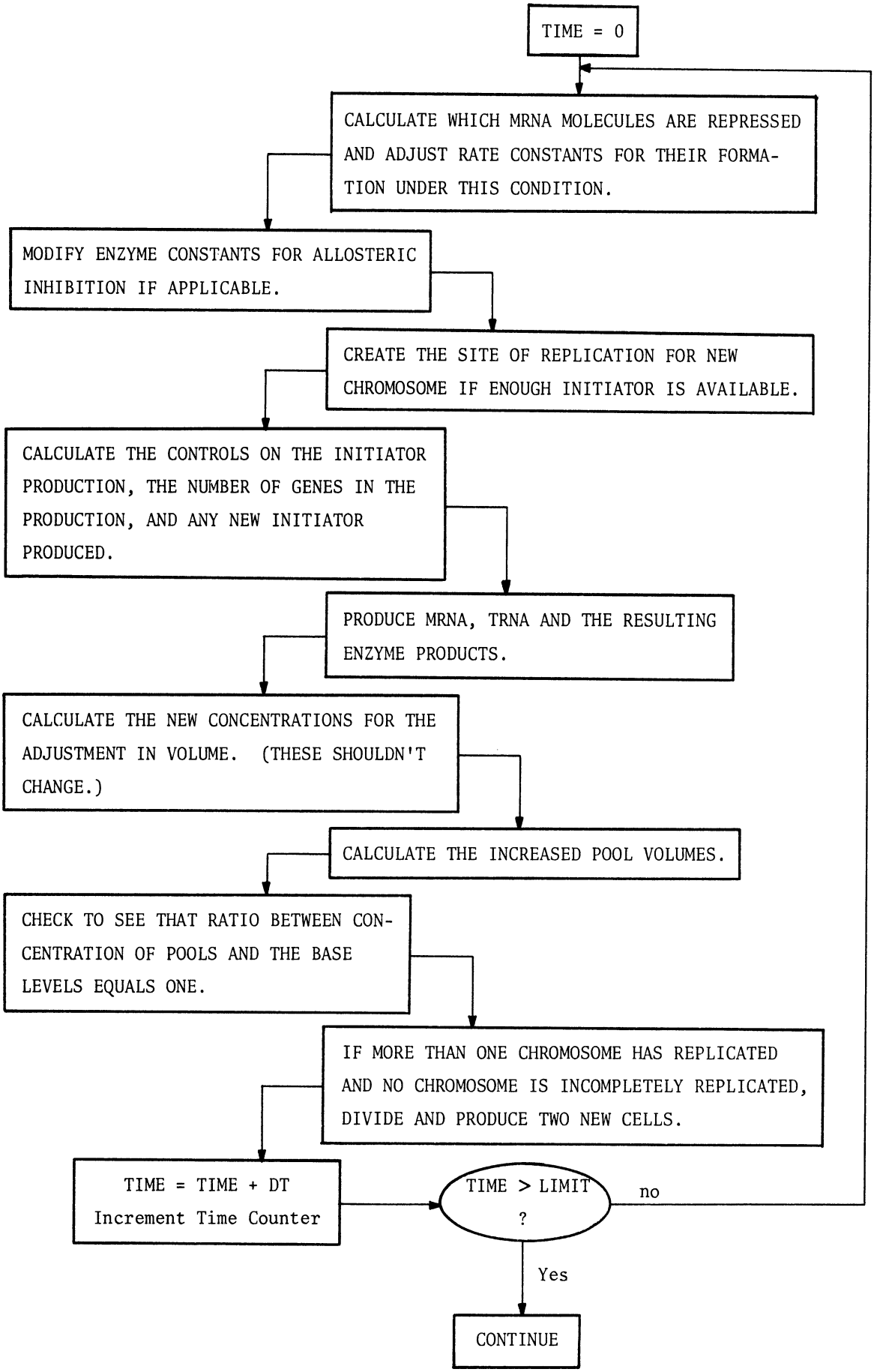


Figure 2.5. Growth Cycle

The model of the cell was simplified in order to make the computer simulation manageable. The chemical constituents of the cell were lumped into the aformentioned pools (Figure 2.1). 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 carbohydrates. 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.

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 constituents of the media. Three different media were "fed" to the simulated cell: (1) a medium containing glucose, ammonium salt, and minerals (Figure 2.6), (2) a medium containing glucose, minerals, and amino acids (Figure 2.7), (3) a medium containing glucose, minerals, amino acids, and nucleosides (Figure 2.8). 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 (Maaloe 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 2.5). 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.

The number of DNA molecules in the simulated cell at the beginning of a cycle of cell reproduction was 2 for mineral medium, and 3 for both amino acid

```
 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 2.6. 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 2.7. 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 2.8. Cell in Environment 3 (Broth).

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

medium and amino acid nucleoside medium. This amount of DNA agrees with data for real cells (Lark, 1966).

One can see from Figure 2.9 that the cell could use allosteric modification of its enzyme pools to calculate rates constants which agreed rather well with the rate constants necessary for realistic growth. All of the rate constants agreed in the first figure with the proper rate constants when the calculation was done by the equations representing allosteric modification for the simulation, and by observation of experimental data for proper rate constants. However, the rate of production of ATP and ADP changed when the allosterically modifed enzymes were used in the simulation. This is shown in Figures 2.10 and 2.11. RADP represents the ratio of ADP at the end of one time increment to the ratio the simulated cell should maintain for proper growth in minimal medium. RADP should be 1.000000 ± 0.00001 if the cell is growing correctly. RADP = 3.760564 instead of 1.000000 (Figure 2.11), and RATP = 0.6329882 instead of 1.000000 (Figure 2.11). The ratios of other chemical pools are approximately 1 when allosteric modification of enzyme pools is used to simulate real control of metabolism (Figure 2.11).

The artificial fluctuation of ATP concentration and ADP concentration did not agree with data for real cells. The errors may have been introduced by approximating changes in concentrations of the pools during a one-second time interval by the rate of change of the pools at the beginning of the one-second interval. The fault in the program is being corrected by introducing predictor corrector methods (Schied, 1968), by decreasing the size of the time interval used for calculating successive concentrations for the pools in the simulation, and by using evolutionary techniques (Chapter IV).

This error illustrates one of the many constraints imposed upon one in realizing a computer simulation. The hardware and software one uses to obtain

C(1) = 1.469818E-12	C(1) = 1.835376E-12
C(2) = 2.183038E-12	C(2) = 2.294225E-12
C(3) = 1.177201E-10	C(3) = 1.177268E-10
C(4) = 6.903309E-12	C(4) = 6.907760E-12
C(5) = 4.533463E-15	C(5) = 4.531281E-15
C(6) = 1.110285E-20	C(6) = 1.110301E-20
C(7) = 1.539183E-15	C(7) = 1.539176E-15
C(8) = 9.812282E-17	C(8) = 9.812185E-17
C(9) = 2.775713E-24	C(9) = 2.775785E-24
C(10) = 1.539183E-15	C(10) = 1.539060E-15
Rate constants calculated at start of program	Rate constants by simulating "Allosteric Modification"

Figure 2.9. The Cell Successfully Adjusted its Enzyme Rate Constants by Allosteric Modification of its Enzymes.

E indicates multiplication by a power of 10.
e.g., 1.469818E-12 = $1.469818 \cdot 10^{-12}$.

```
  RDNA  = 9.999995E-01 RMRNA = 9.999996E-01
  RNUC  = 1.000000E 00 RAA = 9.999992E-01
  RDNA1 = 9.999999E-01 RDNA2 = 9.999999E-01

  RTRNA = 9.99998E-01 RRIB = 9.999996E-01 RWALL = 1.000000E 00 RPRTN = 1.000000E 00
RATP = 9.999976E-01 RADP = 1.000014E 00
  RRNA = 9.999992E-01
```

Figure 2.10. Ratio of concentrations of chemicals at the end of one time step to base concentrations cell should maintain, for rate constants calculated at start of program.

E indicates multiplication by a power of 10. E.g., 9.999995E-01 = $9.999995 \cdot 10^{-1}$.

```
  RDNA  = 9.999990E-01 RMRNA = 9.999993E-01
  RNUC  = 1.000413E 00 RAA   = 1.000221E 00
  RDNA1 = 9.999999E-01 RDNA2 = 9.999999E-01

  RTRNA = 9.999995E-01 RRIB = 9.999992E-01 RWALL = 1.000000E 00 RPRTN = 1.000000E 00
RATP = 6.329882E-01 RADP = 3.760564E 00
  RRNA  = 9.999984E-01
```

Figure 2.11. Ratio of concentrations of chemicals at the end of one time step to base concentrations cell should maintain, for rate constants obtained by simulating allosteric modification.

E indicates multiplication by a power of 10. E.g., 9.999990E-01 = $9.999990 \cdot 10^{-1}$. A ratio of 1 indicates success. Note that all ratios are 1 (approximately) with the important exceptions of ADP and ATP using allosteric modification.

the advantages of rapid computational ability and large memory have limitations. The computer uses digital arithmetic to accomplish calculations involving real numbers. In mapping the field of real numbers onto a finite field one may lose track of some of the implications in terms of numerical errors. One may lose the associative and distributive properties of the real field one is attempting to simulate. Several common types of errors must be acknowledged, and their baleful influence avoided where possible. Roundoff error results from approximating a real number of a finite string of digits. Truncation error results from approximating an infinite series by a finite series in a numerical technique. Experimental error may be implicit in the data one analyzes from the real world (Hildebrand, 1956).

The simulated cell produced chemicals and cell mass at a logarithmic rate, but duplicated in a stepwise fashion (Figure 1.1) just as 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 models used to write the simulation.

The simulated cell employed repression to control the production of its enzymes (Figure 2.12). Repression operated at the DNA 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 hooking together amino acids attached to transfer RNA. This hooking 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 shorply curtailed; the messenger RNA already present rapidly decayed, and almost no new messenger RNA for EK2

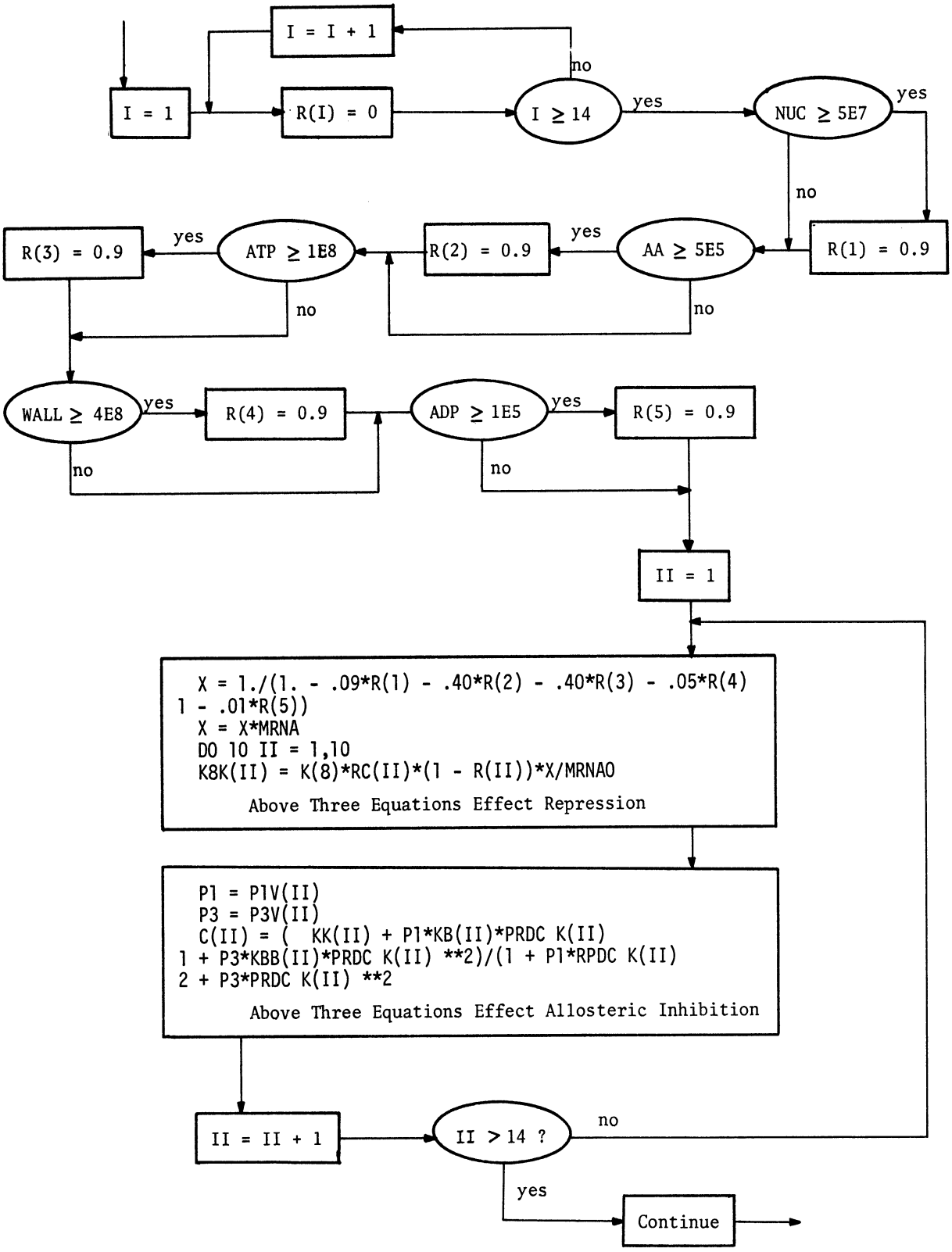


Figure 2.12. Repression and Allosteric Inhibition.

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

production was produced.

In DNA replication a new DNA molecule was produced using the old DNA molecule as a model. In cell division, a cell split into two new cells by forming a cell wall partition through its center. DNA replication (Figures 2.13, 2.14, 2.15 and 2.16) and cell division (Figures 2.17 and 2.18) were represented as two separate though interdependent processes.

DNA was simulated as a circular molecule. Before initiating DNA replication, an empty site for attachment of the newly produced DNA to the cell membrane had to be available. If an empty DNA site was present on the cell membrane, a DNA molecule already present in the cell would begin replication of a new DNA molecule which would attach to the empty DNA site. Once a DNA molecule started to replicate its rate of replication was dependent on energy available as ATP (adenosine triphosphate), and on the availability of enough nucleotides to form the new DNA.

New DNA sites of attachment on the membrane were produced under control of the initiator gene (Figures 2.13 and 2.14). High concentration of amino acid stimulated the rapid production of new sites, and therefore allowed more DNA molecules to replicate at the same time. This led to faster growth. This faster growth expressed the relationship between the number of replicating DNA molecules and metabolic conditions.

Whenever any DNA molecule had completely replicated, a decision was made by the simulated cell as to whether it should divide (Figure 12). If the cell could divide without breaking any DNA strands in the process of replication, it did divide. Otherwise it waited until interfering strands completed their replication, and then looked again to see if the road was clear for division. This model of DNA replication and cell division produced simulated division rates similar to division rates of real cells growing in

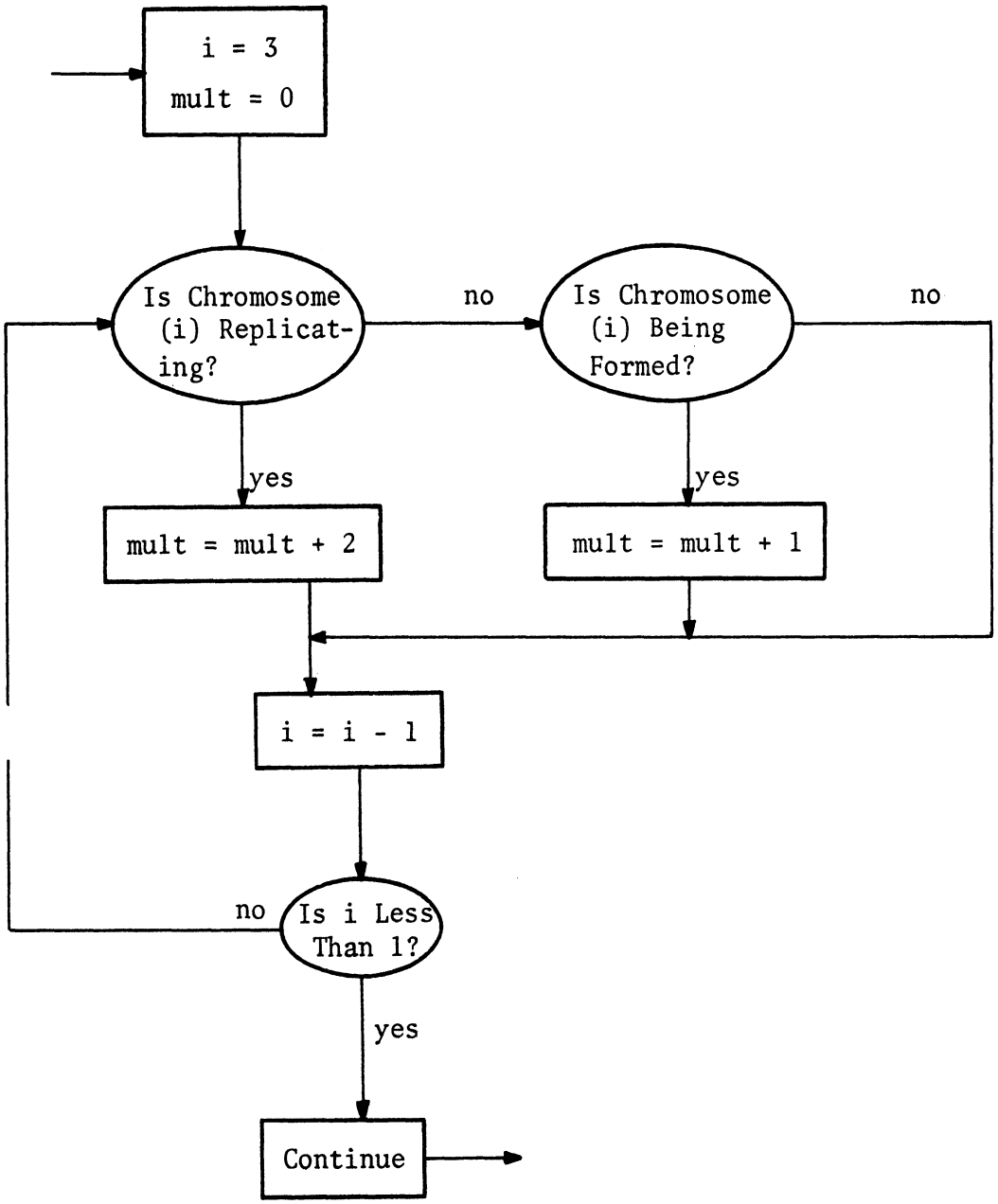


Figure 2.13. Number of genes producing initiator substance into cytoplasm.

Number of genes = MULT. The number of genes increases by one for every chromosome which has multiplied greater than ten percent of its length.

```
      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
```

Figure 2.14. Number of genes producing initiator substance into the cytoplasm.

Number of genes = MULT. The number of genes increases by one for every chromosome which has multiplied for greater than ten percent of its length.

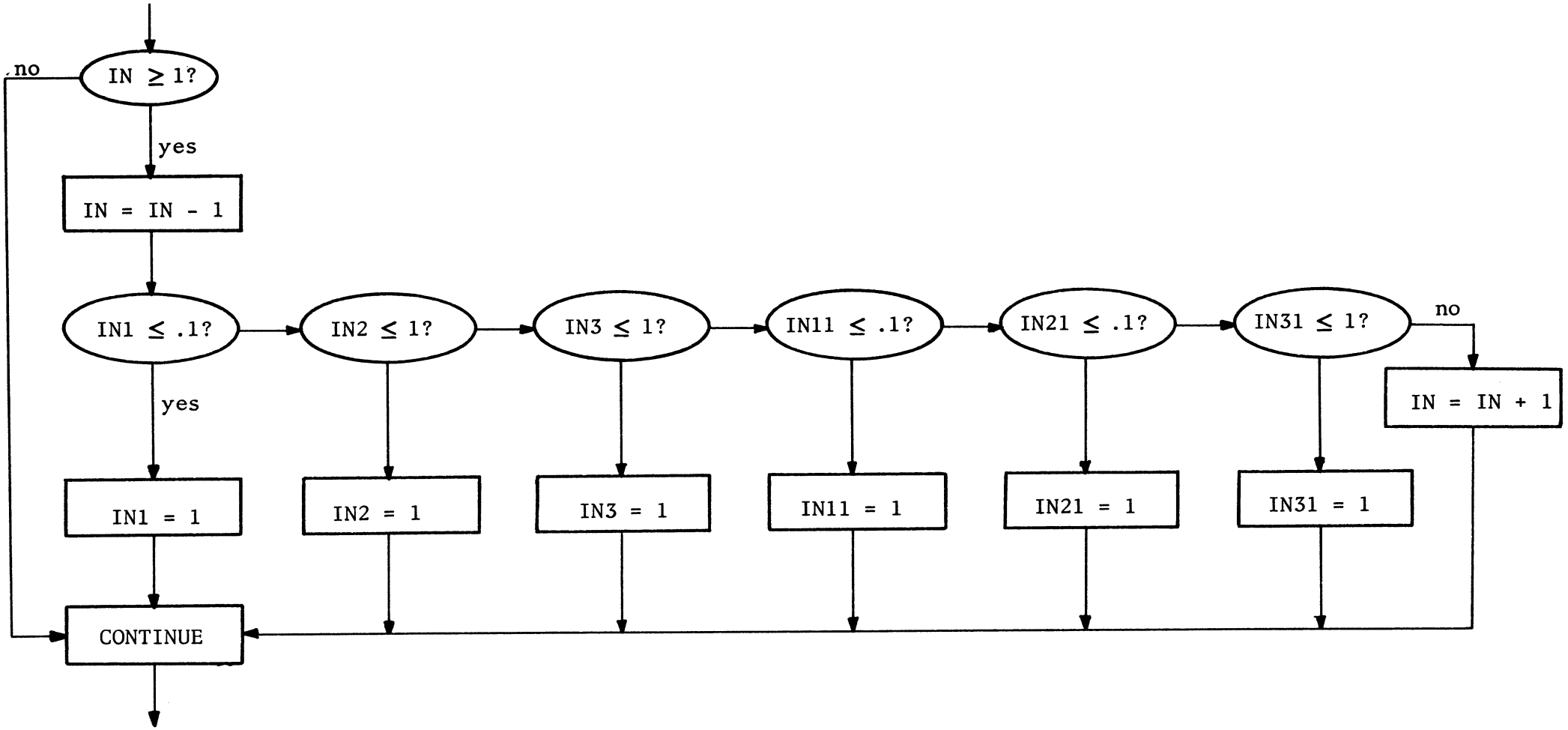


Figure 2.15. Creation of new replication sites when cytoplasmic initiator substance is high enough (greater than one).

E.g., IN1 will be the replication site created if IN is greater than or equal to one. If IN1 is already created, other sites get a chance.

```
          IF (IN - 1.) 69,55,55
55        IN = IN-1.
56        IF(IN1 - 1.) 57,57,58
57        IN1 = 1.
            GO TO 69
58        IF (IN2 - .1) 59,59,60
59        IN2 = 1.
            GO TO 69
60        IF (IN3 - 1.) 61,61,62
61        IN3 = 1.
            GO TO 69
62        IF (IN11 - .1) 63,63,64
63        IN11 = 1.
            GO TO 69
64        IF (IN21 - .1) 65,65,66
65        IN21 = 1.
            GO TO 69
66        IF (IN31 - .1) 67,67,68
67        IN31 = 1.
68        IN = IN + 1
69        CONTINUE
```

Figure 2.16. Creation of new replication sites when cytoplasmic initiator substance is high enough (greater than one).

E.g., IN1 will be the replication site created if IN is greater than or equal to one. If IN1 has already been created, other sites get a chance to appear.

```
          IF (DNA1T - 2.) 81,76,76
76        IF (1.000010 - DNA2T) 77,77,78
77        IF (DNA2T - 2.) 81,78,78
78        IF (1.000010 - DNA3T) 79,79,80
79        IF (DNA3T - 2.) 81,80,80
80        NO = 2*NO
            IN = IN/2.
            IN1 = IN11
            IN2 = IN21
            IN3 = IN31
            IN11 = 0.
            IN21 = 0.
            IN31 = 0.
            COUNT = 0.
            DT = 1.
            VOL = VOL/2.
```

Figure 2.17. Cell division occurs if conditions are appropriate.

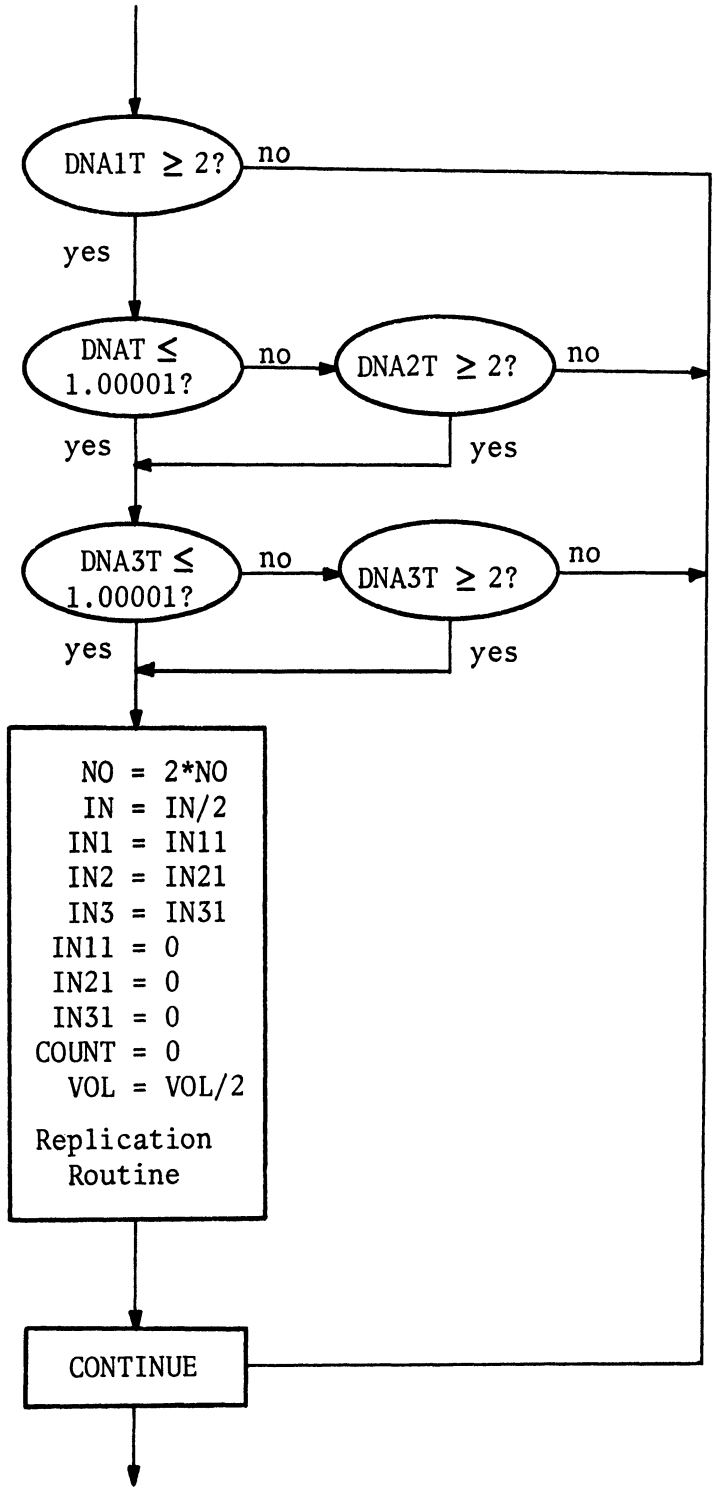


Figure 2.18. Cell division occurs if conditions are appropriate.

real media (Davis et. al., 1968).

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

The simulated cell employed feedback inhibition to control the activity of the enzymes already present (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 ability. The relative amount of EK2 in each form determined the activity of the EK2 present in the cell in terms of its efficienty 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, the greater was the percentage of EK2 in its low activity form, and the less effective was the EK2 in production of amino acids from carbohydrates.

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)		0	change in enzymes for TRNA production in one time increment
DIN		0	change in initiator concentration in one time increment
DNAO		0	DNA at time zero
DNA1		0	chromosome 1 "concentration", i.e., amount/volume of cell
DNA1Z		0	chromosome 1 at zero time
DNA1T		0	total chromosome 1
DNA2		0	chromosome 2 "concentration"
DNA2T		0	total chromosome 2
DNA2Z		0	chromosome 2 at zero time
DNA3		0	chromosome 3 "concentration"
DNA3T		0	total chromosome 3
DNA3Z		0	chromosome 3 at zero time
DNAP		0	ATP used per DNA molecule synthesized
DNA		0	DNA
DNASB	A	0	array to save concentrations of DNA in different environments
DNUC		0	change in nucleotide concentration
DBLE		0	time for cell to go through one reproductive cycle
DPRTN		0	change in protein in one time increment
DRIB		0	change in ribosome in one time increment
DMRNA		0	change in MRNA in one time increment
DRNA		0	change in total RNA in one time increment
DRNK(i)	A	0	change in MRNA for enzyme EK(i) in one time increment. i ranges from 1 to 10.
DT		0	length of one time increment, = differential
DUM1		0	dummy variable in solve function
DUM2		0	dummy variable in solve function
DUM3		0	dummy variable in solve function
DVOL		0	change in cell volume in one time increment
DWALL		0	change in cell membrane and cell wall in one time increment
DPRDK	A	0	array of change in product concentration in one time increment
PRD	A	0	the stored array of the previous four product values, for predictor corrector
DPRD	A	0	array of the four previous D(product) values for the predictor corrector
PPRD	A	0	current array of the predictor values of products
CPRD	A	0	current array of corrector values of products
EK(1)		0	concentration of enzymes for nucleotide production
EKZ(1)		0	concentration of enzymes for nucleotide production at zero time
EK(2)		0	concentration of enzymes for amino acid production
EKZ(2)		0	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

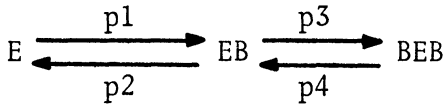
Appendix to Chapter II (Cont.)

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)

CHAPTER III

COMPUTER SIMULATION OF ALLOSTERIC INHIBITION

Feedback Inhibition Used in Simulation of a Living Cell



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:

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

Manipulating the equations for equilibrium conditions one can obtain the following expressions for E, EB, and BEB, where new rate constants are defined as follows:

$$P1 = p1/p2$$

$$P3 = p3/p4$$

Expressions for E, EB, and BEB:

$$E = E_{total} \cdot \frac{1}{(1 + P1 \cdot B + P3 \cdot B^2)}$$

$$EB = P1 \cdot B \cdot E$$

$$BEB = P3 \cdot B^2 \cdot E$$

Manipulations leading to expressions for E, EB, and BEB:

at equilibrium,

$$p1 \cdot B \cdot E = p2 \cdot EB$$

$$p3 \cdot B^2 \cdot E = p4 \cdot BEB$$

$$EB = (p1/p2) \cdot B \cdot E = P1 \cdot B \cdot E$$

$$BEB = (p3/p4) \cdot B^2 \cdot E = P3 \cdot B^2 \cdot E$$

$$E = E_{total} - EB - BEB = E_{total} - \frac{p1}{p2} \cdot B \cdot E - \frac{p3}{p4} \cdot B^2 \cdot E$$

Dividing out E from both sides, one obtains

$$1 = \frac{-p1}{p2} \cdot B - \frac{p3}{p4} \cdot B^2 + E_{total}/E$$

Manipulating, and substituting P1 = p1/p2 and P3 - p3/p4 one obtains

$$E = E_{total} \cdot \frac{1}{1 + P1 \cdot B + P3 \cdot B^2}$$

which gives the three expressions we started out to obtain. Setting equal the two equations giving $\frac{d(B)}{dt}$ one obtains

$$K(k,j) \cdot E_{total} \cdot A = [KK \cdot E + KB \cdot EB + KBB \cdot BEB] \cdot A$$

Substituting in expressions for E, EB, and BEB in terms of E_{total}, one obtains

$$K(k,j) \cdot E_{total} \cdot A = A \cdot \frac{[\, E_{total} \,]}{1 + P1 \cdot B + P3 \cdot B^2} \cdot [KK + KB \cdot P1 \cdot B + KBB \cdot P3 \cdot B^2]$$

Cancelling out A and E_{total} from both sides of the equation, and re-arranging, one gets

$$KK + P1 \cdot B \cdot KB + P3 \cdot B^2 \cdot KBB = K(k,j) \cdot (1 + P1 \cdot B + P3 \cdot B^2).$$

Since B is simply a product of the reaction A → B the concentration of product (k) in environment (j) is PRDC(k,j). One similarly indexes the KK, KB, and KBB series so that

KK(k) = the rate constant KK associated with product PRDC(k)

KB(k) = the rate constant KB associated with product PRDC(k)

KBB(k) = the rate constant KBB associated with product PRDC(k)

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

Substituting in the indexes for purposes of iterative calculations, one obtains the equation

$$KK(k) + P1 \cdot PRDC(k,j) \cdot KB(K) + P3 \cdot PRDC(k,j)^2 \cdot KBB(k) = K(k,j) \cdot (1 + P1 \cdot PRDC(k,j) + P3 \cdot PRDC(k,j)^2).$$

For each product k, one obtains three linear equations for the three unknowns KK(k), KB(k), and KBB(k) for the three values of the environment j = 1, j = 2, and j = 3. j = 1 indicates mineral glucose environment, j = 2 indicates amino acids have been added (indicated by the program variable

CAA = 1), while j = 3 indicates a broth environment (indicated by the program variable BROTH = 1). 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.

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

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

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

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

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

(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).

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

```
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.

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.

$\frac{d(B)}{dt}$ = change in amount of B as a derivative with respect to time.

Given the three equations in three unknowns KK(k), KB(k) and KBB(k) (Figure 3.3) one solves for the unknowns in the SOLVE routine of the program (Figure 3.4). Trial values of P1 and P3 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.

```
      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
C****   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.

CHAPTER IV

COMPUTER SIMULATION OF EVOLVING DNA

The computer simulation of a living cell adapts phenotypically to three different chemical environments (Chapter II). I will extend the simulation so that the cell can adapt genetically as well as at the phenotypic level. I will represent DNA as an array in the computer in which are stored the indexes and values of various rate constants in the equations representing the simulated cell (Figure 3.1).

Four powerful genetic operators for evolution of populations are 1) crossover, 2) inversion, 3) mutation and 4) dominance.

Crossing over permits preferential multiplication of groups of subroutines which interact well, giving coadaptation. Without crossing over all subroutines are equally linked on the string referencing an individual as a collection of subroutines, so that there is no such concept as "close together on the linkage map".

Inversion is necessary to rearrange the genetic location of different subroutines, so that those that should be close together on the genetic map get a chance to approach each other during evolution.

Mutation is necessary to explore a large genetic space, and also to regenerate attributes of functions lost through selection.

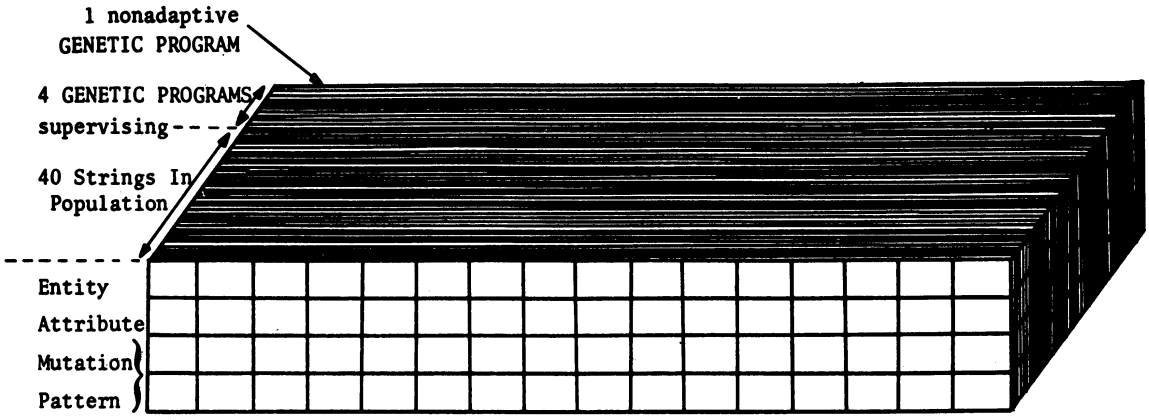
Dominance is necessary to preserve attributes of functions which are at a temporary disadvantage, but may be useful at some later time in evolution.

Duplication of individuals, as well as the genetic operators crossing over, inversion, and mutation will be simulated by operations on the contents of the arrays. The modified arrays will be used to calculate the modified rate constants by which the new populations of simulated cells grow.

I will partially realize the function of dominance in my simulation by strongly directed mutation to restricted sets of mutant alleles, a mechanism not possible in real DNA, but one which realizes one function of dominance, i.e., the conservation of genetic variability in a population.

Since the simulation is of a haploid bacterial population, I will attempt to simulate in a reasonable amount of computer storage populations of haploid bacteria, which store variability without extensive use of dominance and diploidy. This is to make easier the realization of genetic mechansims used by real bacterial populations, rather than because diploidy is unreasonable. Indeed diploidy offers a natural and straightforward way to realize the power of genetic operators, and to store genetic variability for evolutionary demands put on the population by changing environments. Furthermore, diploidy permits storage in the form of valuable substrings of successful alleles, allowing many sampling advantages which will be diminished in my representation of a population by applying genetic operators to individuals who represent the means of probability density functions, defined by the formula for the density together with the mean and variance of the density. However, many haploid populations of bacteria exist in nature in environmental niches which are accessible to less successful diploid competitors, e.g., protozoa, indicating that the haploid mechanisms are more successful than the diploid ones in certain circumstances.

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, and the rest of the organism 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, and will consider only unfixed



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 4.1. The Chrom Array.

loci as strings. Since I am free to set linkage parameters as I wish, I can increase linkage to account for those fixed loci which do not explicitly appear. Most of the genetic characteristics of the indivudals such as mutation rate and crossover will be represented as separate strings of adaptive or non-adaptive genetic programs, each of which will supervise the evolution of a population.

It is important to have adaptive genetic programs which can evolve, since selective procedures may lead to unexpected consequences, such as death for long legged chickens, and should be amenable to modification. (The biological example, death for long legged chickens, refers to an experiment in which longer shanks were selected in populations of chickens (Wallace, p. 455).) The populations so selected always became extremely unfit.

There are two detectors of phenotypic limitations on genetic evolution of the simulated cells which are particularly easy to observe. One is a wide disparity between simulated chemical concentrations of cell metabolites and the concentrations necessary for balanced growth. The second straightforward detector of phenotypic imbalance is the inability to modify the simulated enzymes to account for the growth rates required in the three simulated environments. The impossibility of manipulating the enzymes to produce required growth rates immediately shows up as the inability to produce a solution in the solve routine of the computer program (Chapter 3). The inability to maintain biochemical equilibria necessary for life shows up in a departure of the ratios of the concentrations of biochemicals to the necessary concentrations. These ratios should be 1 if metabolic equilibrium is maintained, and departures of the ratios from 1 indicate instability. For this reason, the utility function which directs the rate

of reproduction of each individual in the population contains the sum of (ratio + 1/ratio) in its denominator, so that the further the ratio departs from 1, the lower the value of the utility function, and the less the rate of reproduction of the individual under consideration. Inability to solve the equations for allosteric modification of the enzyme pools adds a 10 to the denominator of the utility function, so that individuals which can not use allosteric modification correctly can still be ranked as a function of how far off their ratios are.

Inducing sophisticated quantities by simple genetic operations on finite strings relates directly to Holland's description of the complex populations of schemata and operators on schemata, both conservative and nonconservative which are present in simple finite populations of evolving strings, and which confer upon these simple finite populations of strings powerful evolutionary capabilities. The complexity of calculation of average excess induced by elimination of forty percent of the population at each reproductive cycle illustrates the ease with which one can realize something which takes a good deal of effort to describe in quantitative terms, and points up the advantages of studying evolving schemata in the space of a "successfully" evolving population of strings.

It is important to distinguish between selection induced on schemeta by genetic operations on three dimensional arrays referencing heuristic programs, and the criteria used directly on the programs themselves. To illustrate this perhaps subtle, and certainly profound distinction, I am going to illustrate the genetic operation of selection by eliminating sixty percent of the strings each reproductive cycle, and fill in the missing sixty percent with new strings produced from the old strings not eliminated by the genetic operators crossing over, inversion, and mutation. A strings 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. To return to the number judging performance by the string description as opposed to a sophisticated measure such as average excess for that string, how well and how quickly the simulated cell adjusts to changes in its chemical environment will be simply expressed as utility = 1/(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 inhibition). Since the last three quantities are in the denominator of the formula for utility, the larger the deviation of the chemical ratios, the longer the time to evolve, and the greater the failure to solve for allosteric modification, the less the utility. Obviously computer time doesn't even exist in the real cell, and to destroy sixty percent of the programs is a clumsy and unsubtle procedure. However, given this environment, each string does have some average excess induced on it, which would have to be calculated over the run of the program. It is important to perceive that this average excess exists, but 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 the evolution of an effective program to accomplish a task, one would certainly want to calculate the average excess of each program rather than defining it implicitly by the genetic procedure for producing new programs from old ones.

It will be recalled that the utility of the best individual produced

under direction of an adaptive genetic program, as judged by that adaptive genetic program is recalculated by the nonadaptive genetic program. The nonadaptive genetic program gives the utility it calculates for the description of the best string in its population to the adaptive genetic program directing evolution of that population of strings. This utility is then used by the nonadaptive genetic program to direct the evolution of the adaptive genetic programs. Elimination of unnecessary genetic operations is both a practical advantage to the programmer, and an experimental fact in competitive natural populations. Therefore the time it took the adaptive genetic program to manipulate its population to produce the best string is added to the denominator of the SUM which is the utility given to that adaptive genetic program. Much of the information contained in rows 3 and 4 of the strings CHROM column can be approximated by ignoring the cumulative frequency distribution indexed in the 4th row, and simply using the uniform distribution. An adaptive genetic program which does this will gain in utility. For example FREQ(3) may be set to a uniform frequency distribution, and effectively ignored for the jth entity during mutation since the random number generator itself sets up a uniform distribution by choice of the correct interval in which the random numbers are generated. If the loss in evolutionary power does not overbalance this 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 program eventually. It was included to indicate the ease with which a general and powerful evolutionary program may be written. Since the whole growth and phenotypic adaptation procedure takes less than 3 seconds of IBM 360 computer time, and may be shortened, the program is not as time consuming as it might appear to be at first glance. Storage of large blocks of the program on disks or in files until needed

would also economize on computer costs.

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 order to be a member of the survival population, a newly generated program has to be in the best four in the population of running programs. The higher the value of one of the old surviving programs, the less likely it is to 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 to use Holland's formal theory of adaptive systems to support the validity of writing extremely simple, albeit evolutionarily powerful heuristic programs.

Sophisticated molecular interactions effecting negative feedback of metabolic processes at both the DNA and cytoplasmic level, as well as positi controls of DNA and cell division enable a real cell to survive well, and to explore only a particularly productive subset of possible physiological states. The molecular mechanisms underlying many of these sophisticated relationships are often simple and direct from a molecular point of view. For example, looking to see whether DNA is in the process of replicating, and not replicating the cell unless all DNA which has begun replication has completed replication simply involves a replication site occurring at a potential site of cell division. Only when replication of the circular DNA molecule is completed can the new cell wall be laid down. This type of sophisticated limitation of possible actions also occurs in genetic modification through inertia, in that a chromosome only undergoes small changes compared to all possible changes which may occur. Which changes survive is very closely dependent on the structural and functional relationships existing

in the cell, and sophisticated evolutionary schemes may well embody this information.

I will pick as unfixed variables from which to generate my population of strings (and schemata) fifteen control parameters which the cell uses for phenotypic adaptation to changing environments (Figure 3.4). The five control constants corresponding to repression of enzyme production by DNA are R(1), ..., R(5) (Appendix to Chapter II). The ten control constants corresponding to allosteric inhibition of enzyme activity after the enzyme has already been formed are P1V(1), ..., P1V(5), P3V(1), ..., P3V(5).

I will keep the simulated cell program and all variables in program common, and call in the simulated environments, adaptive genetic program, and the ten string population from disk storage. The program for the simulated cell will receive the values of its variables according to the information of a string representing one individual, the simulated cell program will be run, a utility will be calculated according to the adaptive genetic program, values of the variables will be filled in according to the next string in the population, until all of the strings in the ten string population have obtained a utility for that run. The adpative genetic program will then operate on the ten string population to form a new population according to its genetic information and the utilities awarded to each string when its values were loaded into the simulated cell program and run. The adaptive genetic program and its ten string population will then be returned to disk storage, and the next adaptive genetic program and population will be brought into core (Figures 3.2 and 3.3). Each adaptive genetic program will be awarded a utility dependent on information in a non-adaptive genetic program. The four adaptive genetic programs and the non-adaptive genetic program will then be loaded into core, and the two worst adaptive genetic

programs will be replaced by programs generated from the two best adaptive genetic programs. I will introduce inversions in the populations of strings supervised by the two best adaptive genetic programs, and use these modified strings to form the new populations of the new adaptive genetic programs just formed. The inversions will only appear in the populations of the new adaptive genetic programs, and will be homozygous in these populations for ease of genetic operations on the populations (Figures 3.2 and 3.3).

Information concerning genetic manipulation of the evolving strings will be stored in the references to the adaptive genetic programs (Figure 3.5). This information determines crossover, inversion, mutation and selection. The pattern of mutation, once the locus to mutate has been determined by the adaptive genetic program, 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 of the adaptive genetic programs in much the same way that the adaptive genetic programs direct the evolution of the strings of individuals (Figures 3.3, 3.5 and 3.6).

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 for ENTITY(1) through ENTITY(14). The program for the simulated cell then has the proper values for its variables so that it can 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 with a value of the third dimension of the array equal to I + J. The program statement is

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

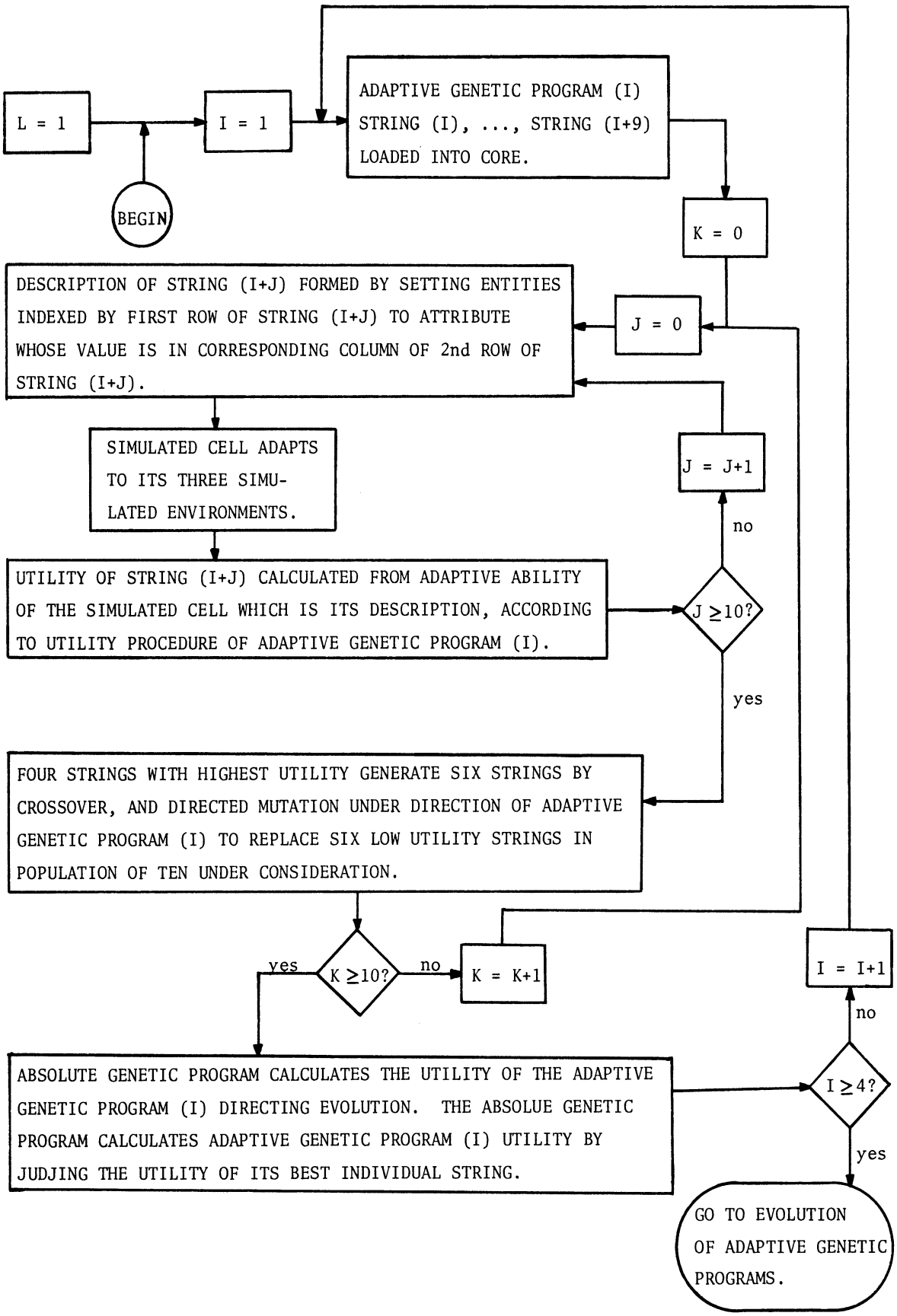


Figure 4.2. Evolution of Strings.

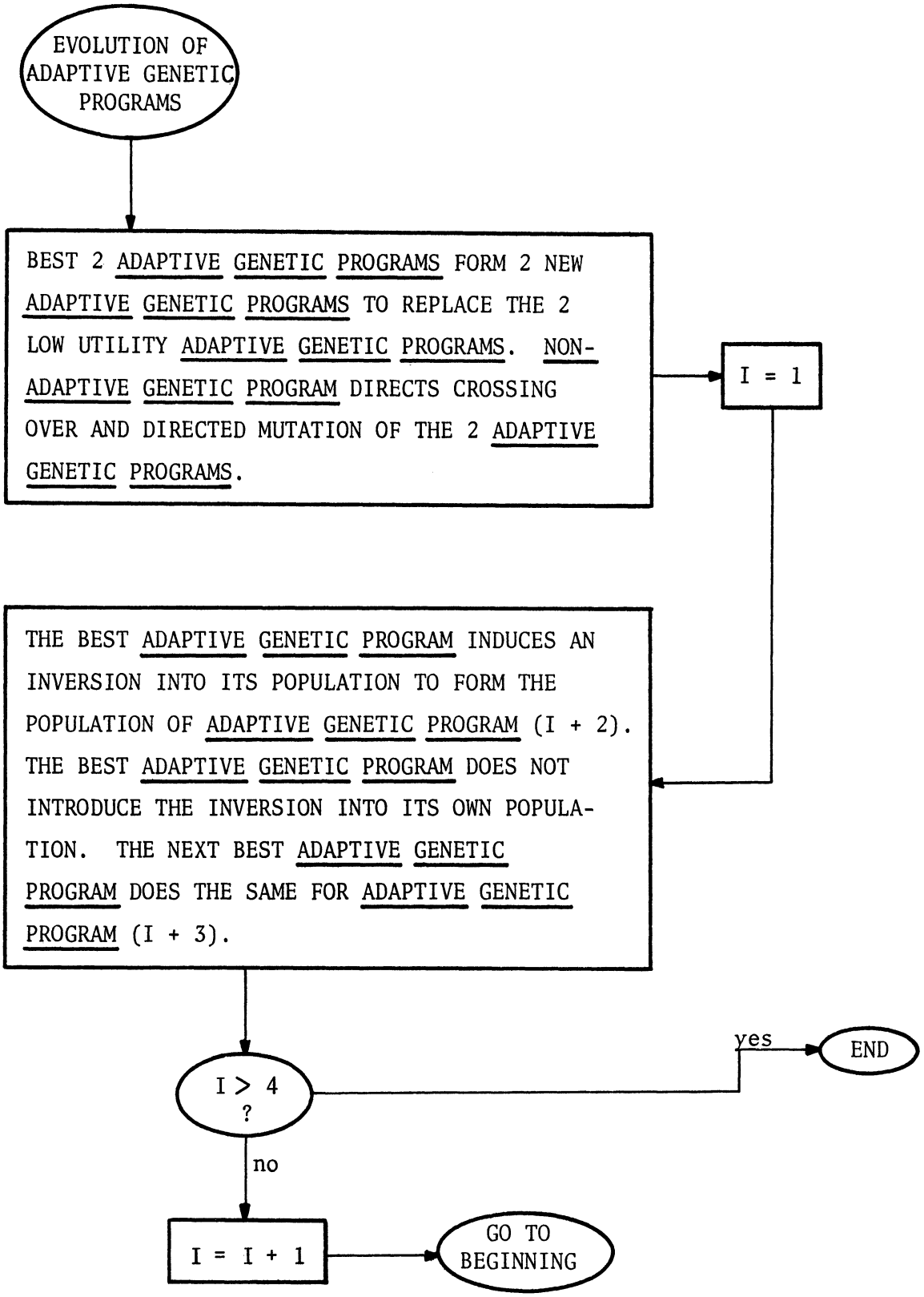


Figure 4.3. Evolution of Adaptive Genetic Programs.

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 4.4. Description of Strings References when Third Dimension of CHROM Array Ranges from 1 to 40.

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 4.5. 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.

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

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 induced by the presence of inversion heterozygotes during real crossing over, since such heterozygotes 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 for the individuals with the highest and next highest utilities in the population of 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, where 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, the less the probability of a crossover. To generate a probability curve other than uniform for crossing over, a cumulative probability distribution could be used, and one could pick the point on the chromosome whose cumulative distribution function is less than but closest to a random number which was generated between 0 and 1 (Mize and Cox, 1968). The same 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, where I and J are variable, and denote individual strings, the crossover takes place at position X, and 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), thus destroying the 5th individual in ranking with respect to utility. If both products of the crossover are saved,

the second recombinant will be loaded into CHROM (I,J,6).

The column for crossover will be obtained by Monte Carlo techniques, using the random number interval and cumulative frequency distribution located in the column of the adaptive genetic program which indexes the crossover entity. (1 happens to index crossing over, so the 3rd and 4th rows of the column containing a 1 in its first row will contain, respectively the random number interval and cumulative distribution used to generate the point of crossover point). For a simplified example of Monte Carlo techniques let the random number be generated over (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 occuring to the right of x. The crossover point $x = F^{-1}(u)$ (Mize and Cox, p74). The crossover does not occur if a random number is generated outside of the range of $F^{-1}(x)$, allowing mutation to increase or decrease the probability of crossing over for the whole chromosome by changing the length of the interval over which the random number is generated if it is wider than the range of $F^{-1}(x)$. If this is not useful, it can be discarded by the genetic program, and a random number interval (1,14) used.

Not only inversions, but also chemical differences on different parts of real chromosomes alter biological crossover frequency. There are complex interactions between different parts along the length of chromosomes undergoing crossing over. The assumption of constant crossover frequency per unit string length is a close approximation to the relation between linkage and percent recombination for real chromosomes. The linkage map of the real chromosome approximates a linear function of percent recombination for map distances less than forty percent. Constant probability of crossover per unit length is therefore a reasonable first

approximation for Holland's theoretical development. However, the probability distributions I 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, and 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).

Inversions are somewhat artificially simulated for programming simplicity. The best adaptive genetic program as judged by the non-adaptive genetic program, selects the sites of inversion using the random number interval and frequency distribution in its column which indexes the inversion entity (column with a 2 in row 1). It then inverts this segment of the strings of its population for loading into the population of the third best adaptive genetic program. Similarly the second best adaptive genetic program introduces an inversion into its population for loading into the population of the worst adaptive genetic program. The inversions are not introduced into the populations of the two best genetic programs. However, the two worst adaptive genetic programs are replaced by genetic combinations of the two best genetic programs, so there is some possibility for good genetic procedures to evolve and interact with improved linkage maps effected by inversion. By appropriate choice of random number interval and cumulative density, one can easily manipulate the probability of obtaining any particular

number. This is particularly useful in directed mutation, where the increment in an attribute becomes easy to control, implicitely defining the recessives stored by a string as its high probability mutants. The whole population of potential mutants changes when an attribute changes, partially simulating a change in dominance. This correspondence is so indirect, however, that I would consider it an experimental part of the program. Since the random number interval and frequency distribution referenced are also subject to mutation and selection along with the rest of the string, the population may evolve an efficient simulation of natural dominance since dominance is useful.

The locus to undergo mutation is obtained by the genetic program, and the mutational increment is then obtained by using the random number interval stored under the locus (locus = entity) undergoing mutation. Mutation in the adaptive genetic programs is analogously effected by the nonadaptive 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, 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 natural mutation where most mutants are random alterations, and

therefore useless, so that saving recessive alleles through protection by dominance in diploids becomes necessary. Directed mutation saves time and storage in a computer simulation since much unnecessary mutation is ruled out, recessive alleles do not have to be stored, and complex calculations for dominant alleles in the canonical realization of the string are eliminated. The directed mutation procedure can be quite simple as outlined above for the entity k(5).

My motivation for only one representation of any particular string in the program is that I would like to preserve maximum variability with minimal computation and storage, since variability is equal to the rate in change of fitness of the population by Fisher's Fundamental Theorem. The value of the utility of an individual, rather than a number of copies of that individual, determines the contribution of that individual to the next generation. There might also be a population of the best unused string from each population to be saved but not used except for recombination, as well as the directed mutation scheme, which permits nonrandom mutation to alleles likely to be useful, in order to permit realization of dominance without lengthy computation. This may lead to lack of fixation of fit individuals, but will enable the evolving strings to try out more combinations. Since old strings are preserved each generation both as members of the next generation (40% of the old population is saved intact) and as potential population members through directed mutation which is rigged to produce useful alleles, the population is unlikely to "forget" a good set of parameters once they are obtained.

The action of the operon in the simulated cell is particularly interesting, since the same repression technique can be applied to replication of portions of DNA by examining concentration of quantities in the program

produced under the direction of one of the loci in the string. If the evolving string references simple subroutines, the genetic program may generate duplicates of the locus which needs modification when the threshold for the quantity reaches a danger level (either too high or too low), or pick alternate subroutines from a list to add to the string, or call the operator for man machine interaction in production of a new subroutine to add to the string, the new subroutine being designed to supplement the offending subroutine already present which was not doing its job.

In a program as complex as the simulated cell, the duplication operator would be a signal for man machine interaction, with the man modifying the subroutine not predicting correctly, or adding a new subroutine to extend one already present. The predictors, however, are easy to define for the simulated cell, since each parameter is closely associated with a simulated activity. 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, for either of these actions would indicate that R(k) is not doing the job it is predicted that it will do.

In a program with simpler subroutines, like Cavicchio's pattern recognition program, 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.

A lumping operator on strings would be related to schemata (Holland). A gene could be merged with another gene by the lumping operator which would convert the references to two parameters to a reference to one parameter. The lumped genes are closely related to successful schemata, since their

survival indicates that the genes which constitute them are successful in combination with each other. By allowing directed mutation within the lumped group of genes, one may reap the reward of hidden recessives becoming dominant without sacrificing the advantage of a coadapted set of genes remaining linked through evolution.

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 do a genetic search of many different spaces, thereby realizing heuristic programs. Examples are the kinds of subroutines useful in pattern recognition (Cavicchio, 1968) or production of English sentences using a generative grammar (Bono, 1968). Both of these tasks have been written in preliminary form as populations of computer programs which evolve over time as a function of how well they do the specific task assigned to them. Heuristic programming may have interesting applications in obtaining programs to accomplish many ill defined algorithms for tasks with a well-defined goal and reward scheme.

EPILOGUE

CANCER IN RELATION TO THE COMPUTER SIMULATION OF A LIVING CELL

Curing cancer is an example of the type of extension of the simulation of a living cell which I would like to eventually make (Heinmetz, 1966). Modifications of the computer simulation might help to screen for cancer curing environments. Cancer is caused by uncontrolled growth of cells in the body. Normally many human cells stop growing in adults, or grow slowly. A human liver cell, for example, is inhibited from growing and dividing by contact inhibition, that is by contact with other liver cells. In a cancerous cell, this inhibition is ineffective. The cancer cells grow without proper controls, crowding out other cells. Another type of control of normal cells is determined by the tissue in which they may grow. Normal liver cells will not grow at all in the lungs, while cancer cells derived from liver may grow in the lungs and crowd out lungs cells. Invasiveness and uncontrolled growth of cancer cells makes them difficult to remove by surgery, and may kill the man afflicted. Understanding the types of changes in cellular control mechanisms giving rise to uncontrolled growth may help in curing cancer by suggesting rational approaches to the prevention of changes leading to cancer, and to plans of attack against cancer cells.

Two types of changes in cellular control mechanisms may lead to uncontrolled, cancer-like growth: 1) a mutation in the cell's genes may alter the cell's control genes; 2) virus genes may enter the cell's chromosome, and subsequently alter the cellular control systems (Davis et. al., 1968). Simulation of cancer caused by mutation and by hidden viruses may enable one to simulate the effect of various environments on normal and cancer

cells, and thereby help to find chemical environments which are likely candidates to test as cancer cures. The rapidity with which one can simulate the effect of different environments may enable one to "test" in far greater amount and detail than one could investigate in actual experiments where limitations of time, space, and experimental organisms are strong constraints. The control of DNA replication in bacteria and in humans is intimately related to the cell membrane, suggesting that extension of the microbial model of DNA replication to human cells may prove feasible (Clark, 1968. Comings and Kakefuda, 1968).

REFERENCES

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

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

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

Comings, D. E. & Kakefuda, T. (1968). "Initiation of Deoxyribonucleic Acid Replication at the Nuclear Membrane in Human Cells", J. Mol. Biol., 33, 225-230.

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

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

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

Gale, D. (1967) "A Geometric Duality Theorem with Economic Applications", Review of Economic Studies, 32, 19-24.

Garfinkel, D. (1966) "A Simulation Study of Mammalian Phosphofructokinase", J. Biol. Chem., 241, 286-294.

Griffith, J. S. (1968) "Mathematics of Cellular Control Processes", J. Theoret. Biol., 20, 202-216.

Heinmets, F. (1964) "Analog Computer Analysis of a Model-System for the Induced Enzyme Synthesis", J. Theoret. Biol., 6, 60-75.

Heinmets, F. (1966) Analysis of Normal and Abnormal Cell Growth, New York: Plenum Press.

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

Holland, J. H. (1968a) "Hierarchical Descriptions, Universal Spaces and Adaptive Systems", University of Michigan Technical Report 08226-4-T, Ann Arbor, Michigan.

Holland, J. H. (1968b) "Theory of Adaptive Systems", Course in Department of Computer and Communication Sciences, The University of Michigan.

Holland, J. H. (1969a) "Hierarchical Descriptions, Universal Spaces and Adaptive Systems", in a Collection of Papers on Cellular Automata (ed. A. W. Burks), Urbana: University of Illinois Press.

Holland, J. H. (1969b) "Adaptive Plans Optimal for Payoff by Environments", in Proceedings of the Second Hawaii Conference on System Sciences.

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

Koch, A. L. (1967) "Metabolic Control Through Reflexive Enzyme Action", J. Theoret. Biol., 15, 75-102.

Lark, K. G. (1966) "Regulation of Chromosome Replication and Segregation in Bacteria", Bacteriol. Rev., 30, 3-32.

Mize, J. H., and Cox, J. G. (1968) "Essentials of Simulation", Englewood Cliffs, N. J.: Prentice-Hall

Murray, A. W. and Atkinson, M. R. (1968) "Adenosine 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.

Stahl, W. R. (1967) "A Computer Model of Cellular Self-Reproduction", J. Theoret. Biol., 14, 187-205.

Strickberger, M. W. (1968) Genetics, New York: Macmillan.

Sugita, M., and Fukuda, N. (1963) "Functional Analysis of Chemical Systems in vivo Using a Logical Circuit Equivalent", J. Theoret. Biol., 5, 412-425.

Tsanev, R. and Sendov, B. (1966) "A Model of the Regulatory Mechanism of Cellular Multiplication", J. Theoret. Biol., 12, 327-341.

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

Weinberg, R. (1968a) "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, pp102-103.

Weinberg, R. (1968b) "Computer Simulation of a Living Cell", Bacteriological Proceedings, G114.

Weinberg, R. (1968c) "Computer Simulation of Self-reproduction by a Living Cell", Genetics, 60, 235.

Weinberg, R. and Berkus, M. (1969) "Computer Simulation of Evolving DNA", Abstract to be published in Biometrics.

Yeisley, W. G. and Pollard, E. C. (1964) "An Analog Computer Study of Differential Equations Concerned with Bacterial Cell Synthesis", J. Theoret. Biol., 7, 485-501.