From: ISMB-93 Proceedings. Copyright © 1993, AAAI (www.aaai.org). All rights reserved.

Grammatical Formalization of Metabolic Processes

Ralf Hofestädt
Computer Science Department, University of Koblenz-Landau
Rheinau 3-4, 5400 Koblenz
Germany - hofestae@infko.uni-koblenz.de

Abstract

In the field of biotechnology and medicine it is of interest to model and simulate metabolic processes. The usual methods to model metabolic pathways are chemical descriptions and differential equations. Moreover, the graph theoretical aspect is discussed and the development of expert systems is in process. In this paper we present the formalization of metabolic processes. Our formalization is based on the theory of formal languages. This formalization is called genetic grammar and represents an expansion of the Semi-Thue-System.

1. Introduction

The modelling of cellular metabolism became possible because of the progress made in molecular genetics 40 years ago. Deoxyribonucleid acid (DNA) was identified as the memory for genetic information and as the controlling unit of the metabolism (Gardner,Simmons & Snustad 1991). Within the DNA different functional units were identified and the analysis of genetic processes has shown that the metabolism can carry out complex tasks (e.g. generation of antibodies) by using nondeterministic mechanisms.

Since 1944 it has been known that DNA controls the metabolism. The model of DNA was presented by Watson and Crick in 1953 and in 1961 Jacob and Monod offered their fundamental model of gene regulation (Watson & Crick 1953, Jacob & Monod 1961). Since 1953 gene regulation has been at the centre of molecular biology. Various discoveries like transposons, homeotic genes and mutator genes have made analysis of gene regulation much more complex. Modelling and simulation are important aspects in the study of complex systems. By these methods, different models and simulations can be found for metabolic processes (Kohn & Letzkus 1982, Franco & Canel 1991, Brutlag & Galper 1991, Collado-Vides 1991).

The aim of our work is to model metabolic pathways. The modelling and simulation of metabolic pathways is important for the progress in the field of biotechnology which will influence the domain of medicine and (human) genetics. Therefore we developed a simulation environment based on the grammatical formalization of metabolic processes.

Our grammatical approach (the so called genetic grammar) is an extension of the cellular grammar (Hofestädt 1991) and opens the way for the application of the theory of formal languages and automata in the scope of molecular genetics. The genetic grammar can be interpreted as a stochastic parallel Semi-Thue-System, which allows the definition of different formal languages.

Various formal languages can be classified using genetic grammar and our simulation tool can generate both the set of languages, which belong to the genetic grammar and also the simulations of biosynthesis, which bring the effects of mutation into consideration.

2. Metabolic processing

Since 1953 many metabolic processes have been analyzed. The model of gene regulation from Jacob and Monod was a fundamental contribution to this analysis (Jacob & Monod 1961). Today, with the methods of molecular biology the isolating, sequencing, synthesizing and transforming of DNA structures is possible.

Indeed, there are many gene banks and electronic gene banks available (e.g. EMBL).

Nowadays it is well known that the analyzed DNA structures control metabolism indirectly. DNA controls the metabolism using special proteins (enzymes) which are synthetic products of structure genes and catalyze biosynthetic processes. DNA structures can be interpreted as genetic instructions. Furthermore, molecular experiments and theoretical discussions have reinforced the view that DNA structures can be interpreted as a 'programming language'.

Moreover, the analysis of the DNA structures has highlighted complex language constructs:

- parallel computation
  simultaneous activation of genetic instructions (transcription and translation processes)
- probabilistic computation
  probabilistic activation of genetic instructions (promotor affinity)
- variable granulation
  a genetic instruction can come into action (concentration of RNA-polymerase)
- dynamic genome
  the genome is a dynamic structure, because mutation, virus-DNA(-RNA) and transposons are able to modify the genome
- modular organization
  the genome is organized by modules (homeotic gene)

The features of metabolic processing, which are the basic elements of biological complexity, comply with the methods used in computer science. However, we have to notice that our knowledge of gene regulation and the semantics of some analyzed DNA structures are still rudimentary. The cooperative activity of these features is the main characteristic of metabolic processing. Moreover, there exists no model in computer science to represent all metabolic features. The integration of these elements into a model will lead to a system which will represent biological complexity.

3. Genetic grammar

Section 2 shows the characteristics of metabolic processing. A method which embraces the metabolic features will expand the framework of methods which are discussed in computer science. In this paper we choose a grammatical formalization to define the genetic language. The basis of this formalization is the Semi-Thue-System which will be extended step by step using metabolic features.

Def. 1: Let Σ be a finite alphabet and $n \in \mathbb{N}^+$. $m \in \Sigma^n$ is called a **message**.

Def. 2: Let Σ be a finite alphabet, $n \in \mathbb{N}^+$ and $\Gamma = \Sigma \cup \{\#\}$. A tuple $c = (\alpha, \beta)$ with $\alpha \in \Gamma^n$ (precondition) and $\beta \in \Sigma^n$ (postcondition) is called an **n-rule**. The set $C_n = \{ c \mid c = (\alpha, \beta)$ is an n-rule $\}$ denotes the set of all n-rules.

Def. 3: Let $\alpha \in \Sigma^n$ and $\beta \in \Gamma^n$ with $n \in \mathbb{N}^+$. α is **similar** to β, in symbols $\alpha \leftrightarrows \beta$, iff

$$\forall i \in \{1,..,n\} \qquad \begin{cases} \alpha_i = \beta_i & \text{for } \beta_i \neq \# \\ \alpha_i \in \Sigma & \text{for } \beta_i = \# \end{cases}$$

Def. 4: The 4-tuple $(n, \Sigma, \Phi, \Theta)$ with $n \in \mathbb{N}^+$, Σ a finite alphabet, $\Phi \subseteq C_n$ a set of n-rules and $\Theta \subseteq \Sigma^n$ the start message set is called **basic system**.

The working method of this system will be defined.

Def. 5: Let $G = (n, \Sigma, \Phi, \Theta)$ be a basic system and $D \subseteq \Sigma^n$. Any rule $c = (\alpha, \beta) \in \Phi$ is activated by the message set D, in symbols c(D), iff
$\exists\, m \in D \;\; m \leftrightarrows \alpha$.
$\Phi(D) = \{ c \in \Phi \mid c$ is activated $\}$ denotes the set of all **activated n-rules**.

Any activated n-rule can go into action.

Def. 6: Let $G = (n, \Sigma, \Phi, \Theta)$ be any basic system and $D \subseteq \Sigma^n$, $c \in \Phi$, $m \in D$ and $\beta \in \Sigma^n$. (m, β) is called **action** of n-rule c, in symbols

$m \ _c\!\!-\!\!> \beta$, iff $c = (\alpha, \beta)$ and $m \leftrightsquigarrow \alpha$.

The simultaneous action of all activated n-rules will be called one-step derivation.

Def. 7: Let $G = (n, \Sigma, \Phi, \Theta)$ be any basic system and $D \subseteq \Sigma^n$. D is called **one-step derivation** into D', in symbols $D \ \Phi\!\!=\!\!> D'$, iff
$D' \subseteq \{ \beta \in \Sigma^n \mid \exists\, m \in D \ \exists\, c = (\alpha,\beta) \in \Phi \wedge m \ _c\!\!-\!\!> \beta \}$.

Def. 8: Let $G = (n, \Sigma, \Phi, \Theta)$ be any basic system and $D_i \subseteq \Sigma^n$ for $i = 0..k$ with $k \in \mathbb{N}^+$. $(D_0,..,D_k)$ is called **derivation**, iff
$\forall\, i \in \{ 0,..,k\text{-}1 \} \ D_i \ \Phi\!\!=\!\!> D_{i+1}$.
For a derivation D into D' we write in symbols $D \ \Phi\!\!=\!\!>^* D'$.

The probability feature is the first extension of the basic system. The biological motivation of this extension based on the analyzed promotor affinity, which is the reason for the probabilistic activation of the protein synthetic process.

Def. 9: Any 5-tuple $(n, \Sigma, \Phi, \Theta, \delta)$ with
$G = (n, \Sigma, \Phi, \Theta)$ a basic system and
$\delta: \Phi \to [0,1]_Q$ a total function
is called a **probability basic system** and $\delta(c)$ is the action probability of c.

The action probability can be interpreted as follows: If message m activates n-rule c, the probability of the event "c will occur in action by m" is $\delta(c)$. If there are various messages $m_1,\ldots,m_k$ which can activate the same n-rule $c = (\alpha,\beta)$ ($m_1 \leftrightsquigarrow \alpha$, $m_2 \leftrightsquigarrow \alpha,\ldots,m_k \leftrightsquigarrow \alpha$), all events "c will occur in action by m_i" will be independent.

For any probability basic system $G = (n, \Sigma, \Phi, \Theta, \delta)$ A is called derivation, iff A is a derivation in the basic system. For each derivation the probability can be evaluated. We can evaluate the probability $P(N'|N)$ to transform the message N into the message N' in the next generation. Therefore, we consider any message set $N \subseteq \Sigma^n$ and pairs (m,c) with $m \in N$, $c \in \Phi$ and c is activated by m. Let $(m_1,c_1),\ldots,(m_k,c_k)$ be such pairs in any order (lexicographical order) and k the quantity. Every word $w \in \{ L,R \}^k$ denotes a set of events which describes a transformation into a new message set (one-step derivation).

Let be $w = a_1a_2..a_k$. w corresponds to the event: for $i = 1..k$,
c_i will occur in action by m_i, if $a_i = L$ and
c_i will not occur in action by m_i, if $a_i = R$.
These are indepentend events and the probability of the one-step derivation is:

$$P(w) := \prod_{i=1..k} q_i, \text{ where}$$

$$q_i := \begin{cases} \delta(c_i), & \text{if } a_i = L \\ 1 - \delta(c_i), & \text{if } a_i = R \end{cases}$$

Each event w will produce an output message h(w). This is the set of postconditions of the n-rules which will be in action:

$$h(a_1..a_k) = \{ \beta \mid \exists\, i \in \{ 1,\ldots,k \} \ a_i = L \text{ and } c_i = (\alpha,\beta) \}.$$

The sum of all probabilities of events w which produce output message h(w) is equal to N' and denotes the probability of the message transformation N into N':

$$P(N'|N) = \sum_{h(w)=N'} P(w).$$

Therefore, for any derivation $A = (N_1, \ldots, N_h)$ the probability is defined by

$$P(A) := P(N_2|N_1) * \ldots * P(N_h|N_{h-1}).$$

In this way we can recursively determine the probability P(M,i):

$$\text{I. } P(M,0) := \begin{cases} 1 & \text{iff } M = \varnothing \\ 0 & \text{otherwise} \end{cases}$$

$$\text{II. } P(M,i+1) := \sum_{N \subseteq \Sigma^n} P(M|N)\, P(N,i)$$

In the next step we define a new class of rules which will allow control of probability values. The biological motivation of this extension is given by mutation genes. Moreover, all rules will be extended by visibility flags, so that every

rule is visible or invisible. In order to control these flags it is necessary to define one more class of rules. The biological motivation of the visibility feature is given by the model of gene regulation of Jacob and Monod.

Def. 10: Let $n \in \mathbb{N}$, Σ be a finite alphabet with $\# \notin \Sigma$ and $\Gamma \in \Sigma \cup \{\#\}$.
A 2-tuple (α,β) is called n-message rule with precondition $\alpha \in \Gamma^n$ and postcondition $\beta \in \Sigma^n$. A 3-tuple (α,β,a) is called n-regulation rule with precondition $\alpha \in \Gamma^n$, target domain $\beta \in \Sigma^n$ and regulator $a \in \{+,-\}$.
A 3-tuple (α,β,p) is called n-probability rule with precondition $\alpha \in \Gamma^n$, target domain $\beta \in \Sigma^n$ and the change $a \in [0,1]_Q$.
c is called **n-rule**, iff c is n-message rule or n-regulation rule or n-probability rule.

Now we are able to define the genetic grammar.

Def. 11: Let $n \in \mathbb{N}$, Σ a finite alphabet with $\# \notin \Sigma$, Φ a set of n-rules, $\Theta_0 \subseteq \Sigma^n$ a start message set, B_0: Φ -> { +,- } a total function and δ_0: Φ -> $[0,1]_Q$ a total function.
A 6-tuple $G = (n, \Sigma, \Phi, \Theta_0, B_0, \delta_0)$ is called **genetic grammar** with message length n, message alphabet Σ and rule set Φ. Φ_N, Φ_R and Φ_P denotes the set of message rules, regulation rules and probability rules of Φ.

Furthermore the configuration of a genetic grammar is important.

Def. 12: Let $G = (n, \Sigma, \Phi, \Theta_0, B_0, \delta_0)$ be any genetic grammar. A triple (N,B,δ) with $N \subseteq \Sigma^n$, B: Φ -> { +,- } a total function and δ: Φ -> $[0,1]_Q$ a total function is called **configuration** of the genetic grammar G with message set N, visibility B and rule probability δ. (Θ_0,B_0,δ_0) is called **start configuration**.

Notation:
$S = \{ B: \mid \Phi \text{ -> } \{ +,- \} \text{ a total function } \}$
$R = \{ \delta: \mid \Phi \text{ -> } [0,1]_Q \text{ a total function } \}$

Any n-rule $c \in \Phi$ is visible (invisible), iff $B(c)='+'$ ($B(c)='-'$). For any n-rule c B(c) is called **visibility** and $\delta(c)$ **action probability** of c. An n-rule is activated in any configuration (N,B,δ), iff it is visible and there is a message in the set N which is similar to the precondition of this rule. Any activated rule will occur in action by its rule probability (corresponding to the rule probabilities of δ). The origin of a message is the effect by an action of a special message rule (the same effect as in the probability basic system).

The action of a regulation rule can change the visibility of other rules: If the message in the target domain of a regulation rule r is similar to a precondition of a rule $c' \in \Phi$ and the visibility of rule c' is not equal to the regulator of rule r, the regulator will be the new visibility of c'. This means, regulation '+' will change from visible to invisible and regulation '-' will change from invisible to visible. It is possible that various regulation rules will influence the visibility of a rule. In this case the visibility will change as it is described above.

The action of a propability rule can change the probability of other rules: If the message in the target domain of a probability rule is similar to the precondition of a rule $c' \in \Phi$, then the change of rule r will be the new probability of c'. It is possible that various probability rules will influence the probability of one rule. In this case the change will be the maximum of all changes which are possible in this state.

The configuration (N,B,δ) will be transformed into configuration (N',B',δ'), iff the action of a subset of the activated rules will be produce N', B' and δ' (visibilities and probabilities which will not be modified will be unchanged).

Moreover, there are well known metabolic processes (mutation and genetic operations) which can not be described by any rule system. These metabolic phenomena only occur rarely so it is not possible to bring these phenomena into the grammatical formalization.

4. Application

The genetic grammar allows the discussion of metabolic processes. However, the metabolism can be interpreted as a special derivation.

Proposition: Every metabolic pathway can be formalized by a derivation of a genetic grammar.

Proof idea: If the semantics of DNA structures and the semantics of the substances which take part in the biosynthesis are treated in a suitable way for the formalism (by using messages and rules) then the formalisation of metabolic pathways is possible. With induction over its members every biosynthesis can be simulated by a sequence of one-step derivations.

It is possible to define various languages which represent different point of views.

Firstly, our model is able to compute all message sets (configurations) which can be derived by a sequence of $i \in \mathbb{N}$ one-step derivations:

1. $L(G,i) = \{ N \subseteq \Sigma n \mid \forall B \in S, \delta \in R$ with $(\Theta_0, B_0, \delta_0)_i => (N, B, \delta) \}$.

The generalization of this language represents all message sets (configurations) which can be derivated by the start-configuration:

2. $L(G) = \{ N \subseteq \Sigma n \mid \exists B \in S, \delta \in R$ with $(\Theta_0, B_0, \delta_0) =>^* (N, B, \delta) \}$.

Both languages offer the opportunity to discuss all the possible pathways given by a specific genetic grammar. In the case of language L(G,i) we will restrict our discussion to i one-step derivations. The complexity for both languages is i^x, in time and space, where x denotes the number of defined rules.

Moreover, it could be necessary to discuss specific metabolic concentrations (configurations in our model) which could appear in the metabolism. In the case of medicine and biotechnology the following languages

3. $L_s(G,i) = \{ M \mid P_K(M,i) \geq s \}$ and

4. $L_s(G) = \{ M \mid \exists i \in N\ P(M,i) \geq s \}$

are able to discuss important questions as 'what is the probability of the occurence of a specific concentration ?'. The complexity of both languages is $c*i^x$ where x denotes the amount of given rules and $c \in]0,1]$.

5. Simulation shell

Our simulation shell is implemented in C on a sun workstation and contains an easy user interface. If the semantics of DNA structures and the semantics of the substances which take part in the metabolic pathway are treated in a suitable way for the formalism (by using messages and rules) then our software shell allows simulation and analysis of metabolic pathways. The simulation program derives the different languages from the basis of the genetic grammar. These are pure simulations (derivations) and statements on the probability of the existence of substances or the concentration of substances (stochastic derivations).

Therefore, all languages could be interpreted as specific analysis features. In this case L(G) and L(G,i) will produce all possible configurations and $L_s(G,i)$ and $L_s(G)$ will discuss specific metabolic questions (e.g. detection of specific substances in the metabolism).

Furthermore, the defined model and the simulation shell allow the detailed discussion of the behaviour of metabolic pathways. Normally the life of any cell will be represented by a specific metabolic pathway which is characterized by mutation events. To allow realistic simulations we expand our simulation program by means of a modification procedure. This procedure simulates mutation events and could be activated by chance after each one-step derivation.

However, a specific metabolic pathway is presented in our simulation program by a specific derivation which is based on expanded one-step derivations.

Example: representation of the lac operon of Escherichia coli (see appendix).

The operon model was proposed by Jacob and

Monod (Jacob & Monod 1961) to explain the induction of the biosynthesis of the enzymes involved in lactose utilization when sugar is added to the medium in which E. coli cells are growing.

To simulate this metabolic pathway we have to code the proteins into messages and the chemical reactions into specific n-rules. In our example we chose the alphabet {0,1}, n = 5 and all commentaries are denoted by the symbol %.

```
 % messages
00000     % lactose
11111     % glucose
11100     % complex of
          % galactosidase, permease
          % and transacetylase
10101     % enzymes
00001     % lactose-inductor

 % actual message set
00000, 11111

 % rule set
+ 1.0 00000 00001
          % induction
+ 1.0 00001 10101
          % enzymes for protein synthesis
- 1.0 10101 11100
          % synthesis of galactosidase,
          % permease and transacetylase
+ 1.0 11111 10101 0.09
          % probability modification
+ 1.0 00001 10101 +
          % repression of the repressor
```

The first one-step derivation:
Glucose and lactose is presented. In this case the CAP-cAMP complex is not activated and the promotor affinity of the lac operon will be low.

```
{ (00001) }, P=1.0
+ 1.00 00000 00001
+ 1.00 00001 10101
- 0.09 10101 11100
+ 1.00 11111 10101 0.09
+ 1.00 00001 10101 +
```

After the first derivation the lactose-inductor is present and the CAP-cAMP complex is destroyed.

The second one-step derivation:
The lactose-inductor comes in contact with the lactose operator gene and the RNA-polymerase enzyme.

```
{ (10101) }, P=1.0
+ 1.00 00000 00001
+ 1.00 00001 10101
- 0.09 10101 11100
+ 1.00 11111 10101 0.09
+ 1.00 00001 10101 +
```

The third one-step derivation:
Normally the lactose operon will not go into the process of protein synthesis.

```
{}, P=0.91
+ 1.00 00000 00001
+ 1.00 00001 10101
- 0.09 10101 11100
+ 1.00 11111 10101 0.09
+ 1.00 00001 10101 +
```

6. Related works

In the research area of 'metabolic pathways' existent models can be classified into two main groups: analytical models and discret models. In this section we will describe the main characteristics of both classes in the case of representative papers.

The goal of analytical models is the exact quantitative simulation, where the analysis of kinetic characteristics of enzymes is important. The paper Waser et al. presents a computer simulation of phosphorfruktocinase. This enzyme is a part of the glycolyse metabolism and catalyses a chemical reaction. Waser et al. model all kinetic features of the metabolic reaction by computer simulation. This computer program is based on chemical reaction rules which are described by differential equations (Waser et al. 1983). Franco and Canelas simulate the purin metabolism by differential equations where each reaction is described by the relevant substance and the catalytic enzyme using the Michaelis-

constant of each enzyme (Franco & Canela 1991).

The analytic models produce continuous values of specific concentrations, which can not be produced by the genetic grammar. Tuning the probability of each rule allows to model the Michaelis-Menten-Theory.

Discret models are based on state transition diagrams. Simple models of this class are based on simple production units which can be combined. Overbeek presented an amino acid production system, where a black-box with an input-set and an output-set describes a specific production unit (Overbeek 1992). The graphical model of Kohen and Letzkus, which allows to discuss metabolic regulation processes, is representative for the class of graph theoretical approaches. They expand the graph theory by specific functions which allow to model dynamic processes (Kohn & Letzkus 1982). The highest abstraction level of this model class is represented by expert systems (Brutlag & Galper 1991). Such systems are developed by higher programming languages (Prolog) and allow the modelling of metabolic processes by facts (proteins and enzymes) and rules (chemical reactions).

The genetic grammar is suited to simulate this kind of models, because it is also a discret state transition model with the oppertunity of modelling concurrent, dynamic and probabilistic processes.

7. Discussion

Interdisciplinary research is becoming more and more important in the area of computer science and biology (Hofestädt, Krückeberg & Lengauer 1993). The main application is to provide methods and concepts of computer science in the area of biology.

The key idea of this paper is to model metabolic pathways in a natural way by rule based systems. Therefore, we have expanded the cellular grammar (Hofestädt 1992) and developed a specific formal language which allows the discussion of biosynthesis and gene regulation processes (metabolic pathways).

Our grammar is called genetic grammar which is the basis for the definition of different formal languages. Each language is able to discuss specific metabolic questions. Unfortunately, the evaluation of these languages is very complex (see section 4). Therefore, specific metabolic discussion is only useful in the case of small systems (pathways). Moreover, the simulation of specific metabolic pathways (derivation of a sequence of one-step derivations) is possible.

We developed a simulation shell based on the formalization of genetic grammars. This shell is implemented in the computer language C and runs on the sun workstation.

Both allow a discussion of metabolic pathways. A complete simulation of existing biological systems is still beyond our present scope. On the one hand DNA structures for most organisms have not been examined completely. And on the other hand even the complexity of small systems like e.g. Escherichia coli is too great to be examined by our simulation shell within a reasonable amount of time. Smaller metabolic pathways (e.g. lactose operon or the isoleucin synthesis) can be analyzed.

The simulation of genetic processes is not only of interest for molecular genetics. It is also important in the area of medicine and biotechnology in order to answer questions, like whether a substance or a concentration of a substance can appear in a metabolic pathway and if so, what is the probability of its appearance (e.g. toxic substances).

For computer science the genetic grammar is of importance, because the developed formalism can be interpreted as a computational model. This model represents the qualities of the metabolism which have developed over millions of years of evolution. This is of special interest for computer science, because in the field of theoretical computer science it is necessary to develop new algorithmic methods which will solve complex problems. A transformation of the analyzed mechanisms in biology has been successful, for example in the case of neural

networks or genetic algorithms. Neural networks and genetic algorithms (Hopfield & Tank 1985, Holland 1975) show that there is a new research area in computer science which develops new computational models based on features of biological computation (Conrad 1990).
The presented formalization shows that the power of biological systems is based on controlled correlation of: data flow, associative, probabilistic and dynamic data processing.

At the moment we are extending our concept with respect to parallel computing, because the derivation process itself is a parallel process. The derivation process could be executed by an associative memory. The further development of our simulation system is therefore made in co-operation with "Gesellschaft für Mathematik und Datenverarbeitung" (GMD). The CAPRA-system, which was developed by GMD, appears as a suitable hardware (Großpietsch 1990) environment for the implementation of genetic grammars.

8. References

Brutlag, D.; Galper, A.; and Millis, D. 1991. Knowledge-based simulation of DNA metabolism: prediction of enzyme action. *CABIOS* 7: 9-19

Collado-Vides, J. 1991. A Syntactic Representation of Units of Genetic Information - A Syntax of Units of Genetic Information. *J. Th. Biology* 148: 401-429

Conrad, M. 1990. Molecular Computing, *Advances in COMPUTERS* 31: 235-324

Franco, R.; and Canela, E. 1991. Computer simulation of purine metabolism. *European J. Biochemistry* 144: 305-315

Gardner, E., Simmons, M., and Snustad, D. 1991. *Principles of genetics*. New-York: John Wiley & Sons

Großpietsch, K. 1990. An architecture for an intelligent multi-mode ULSI memory. In IFIP Workshop Wafer Scale Integration. Como: IFIP 1990

Hofestädt, R. 1991. A cellular grammar to model metabolic processes. In Proceedings Medical Informatics Europe 1991. Wien: LNCS in medical informatics, Springer Verlag

Hofestädt, R. 1992. The simulation of genetic processes. In Proceedings Computational Systems Analysis 1992, 523-529. Berlin: International Conference on Complex Systems

Hofestädt, R.; Krückeberg, F.; and Lengauer, T. eds 1993: *Informatik in den Biowissenschaften*. Heidelberg: Springer-Verlag

Holland, J. 1975. Adaptation in Natural and Artificial Systems. Michigan: MIT Press

Hopfield, J.; and Tank, D. 1985. "Neural" Computation of Decisions in Optimization Problems. *Biological Cybernetics* 52: 141-152

Jacob, F.; and Monod, J. 1961. Genetic regulatory mechanisms in the synthesis of proteins. *J. Mol. Biol.* 3: 318-356

Kohn, M.; and Letzkus, W. 1982. A Graph-theoretical Analysis of Metabolic Regulation. *J. Th. Biology* 100: 293-304

Overbeek, R. 1992. Logic Programming and Genetic Sequence Analysis: a Tutorial. In Proceedings of Logic Programming. MIT Press

Waser, M. et al. 1983. Computer Modeling of Muscle Phosphofructokinase. *J. Th. Biology* 103: 295-312

Watson, J.; and Crick, F. 1953. A Structure for Deoxyribose Nucleic Acid. *Nature* 171: 137

Appendix. The lac operon consists of three structural genes which are called lacZ, lacY and lacA. lacZ codes for the enzyme galactosidase, lacY codes for the enzyme permease and lacA codes for transacetylase. The cluster of this genes, lacZYA, is transcribed into a single mRNA from a promotor just upstream from lacZ. Their induction is controlled at the level of transcription. In the absence of an inducer (lactose), the gene cluster is not transcribed. If an inducer is added, transcription starts at the single promotor, lacP, and proceeds through the genes to a terminator located somewhere beyond

lacA. The expression at the three genes via a common mRNA explains why the relative amounts of the three enzymes always remain the same under varying conditions of induction. Induction essentially represents a switch that causes the genes to be expressed. Inducers may vary in their effectiveness, and other factors may influence the absolute level of transcription or translation, but the relationship between the three genes is predetermined by their organization.

Regulator genes are responsible for controlling the expression of the structural gene cluster, usually via the synthesis of proteins that act to control transcription. The regulator proteins exercise this function by binding to particular sites on DNA.

The lac genes are controlled by negative regulation. This means that they are transcribed unless they are turned off by the regulator protein. Transcription starts at the promotor lacP, just to the left of the first gene, lacZ. The regulator gene, lacI, lies a little further to the left and forms an independent transcription unit. The repressor binds to a sequence of DNA called the operator (lacO). In the case that the repressor binds at the operator, its presence prevents RNA polymerase from initiating transcription at the promotor. The repressor protein has a very high affinity for the operator; in the absence of inducer, it binds there so that the adjacent structural genes can not be transcribed.

But the inducer can bind to the repressor to form a repressor inducer complex that no longer associates with the operator. The presence of glucose has long been known to prevent the induction of the lac operon, as well as other operons controlling enzymes involved in carbohydrate catabolism. This phenomenon, called catabolite repression (or the glucose effect), has apparently evolved to assure that glucose is metabolized when present, in preference to other, less efficient, energy sources.

Catabolite repression of the lac operon is now known to be mediated via positive control of transcription by a regulatory protein called CAP (catabolic activator protein) and a small effector molecule called cyclic AMP. The lac promotor contains two separate binding sites: (1) one for RNA polymerase and (2) one for the CAP-cAMP complex. This complex must be bound to its binding site in the lac promotor in order for the operon to be induced. The CAP-cAMP complex thus exerts positive control over the transcription of the lac operon. Although the precise mechanism by which CAP-cAMP stimulates RNA polymerase binding to the promotor is still uncertain.

Only the CAP-cAMP complex binds to the lac promotor; in the absence of cAMP, CAP does not bind. Thus cAMP acts as the effector molecule, determining the effect of CAP on lac operon transcription. The intracellular cAMP concentration is sensitive to the presence or absence of glucose. High concentrations of glucose cause sharp decreases in the intracellular concentration of cAMP. How glucose controls the cAMP concentration is not clear. Perhaps glucose, or some metabolite that forms in the presence of sufficient concentrations of glucose, inhibits the activity of adenylcylase, the enzyme that catalyzes the formation of cAMP from ATP.

The overall result of the positive control of transcription of the lac operon by the CAP-cAMP complex is that in the presence of glucose, lac operon transcription never exeeds 2 percent of the induced rates observed in the absence of glucose.