Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Working Paper 255 October, 1983

Switching Between Discrete and Continuous Models To Predict Genetic Activity

Daniel S. Weld

ABSTRACT

Molecular biologists use a variety of models when they predict the behavior of genetic systems. A discrete model of the behavior of individual macromolecular elements forms the foundation for their theory of each system. Yet a continuous model of the aggregate properties of the system is necessary for many predictive tasks.

I propose to build a computer program, called PEPTIDE, which can predict the behavior of moderately complex genetics systems by performing qualitative simulation on the discrete model, generating a continuous model from the discrete model through aggregation, and applying limit analysis to the continuous model. PEPTIDE's initial knowledge of a specific system will be represented with a discrete model which distinguishes between macromolecule structure and function and which uses five atomic processes as its functional primitives. Qualitative Process (QP) theory [Forbus 83] provides the representation for the continuous model.

Whenever a system has multiple models of a domain, the decision of which model to use in a given time becomes a critically important issue. Knowledge of the relative significance of differing element concentrations and the behavior of process structure cycles will allow PEPTIDE to determine when to switch reasoning modes.

Table of Contents

1. Introduction

My overall goal is to understand what kind of model of molecular genetics biologists use to reason about certain simple problems of gene expression. Although a set of differential equations might model many such systems, it is clear that people use a different, more qualitative method composed of several independent models: both discrete and continuous. The discrete model is at the level of the molecular mechanisms of biochemical action; it resembles the model of a complex factory. Enzyme machines step down polymeric chains, building parts, clamping subassemblies together and moving them about.[1] There is also a continuous model, grounded in statistics, which is necessary to predict the gross changes in the concentrations of certain molecules.

This paper describes PEPTIDE,[2] an "understander" which uses multiple qualitative models, one mechanistic or discrete and one continuous, to predict the behavior of genetic systems of moderate size, such as individual operons or small viruses.

1.1 Types of Reasoning

The problem domain that I am addressing is the prediction of gene expression in prokaryotic biological systems. A typical problem would take as input a "cell" containing different amounts of various molecules (e.g. DNA, RNA, ribosomes and enzymes). The output would be the predicted history of the concentrations and activities of these molecules.

In solving this problem, I wish to support the kinds of reasoning that a student in a first year course in biochemistry might use. In particular, this thesis will explore the importance and limitations of the following issues:

1. The choice of a good process vocabulary for the domain of molecular genetics.

2. Qualitative simulation that is based on a detailed mechanistic model of the behavior of individual objects in the system, and the utility of such a discrete model.

3. A technique called aggregation which generates a high level continuous representation (like that of [Forbus 83]) from the discrete model. In certain cases, simulation at this higher level is much more powerful than mechanistic simulation.

4. The composite reasoning that results from switching back and forth between the two levels of

---

[1] In fact I am also interested in what seems to be a higher layer of discrete modeling which biologists use to reason about larger systems. This extra layer deals with stereotyped patterns of gene regulation analogous to a factory's middle management. I have some hope that these control "cliches" could be manipulated with techniques similar to those that the Programmers Apprentice uses to represent commonly occuring fragments of code [Rich 80].

[2] PEPTIDE is a system for the "Prediction of Enzymatic Process inTeractions from Individual Descriptions of Enzymes."

simulation described above. How knowledge about the numerical significance (encoded with the "much greater than" relation, >>) facilitates this aggregation process.

To actually predict the activity of genetics systems, PEPTIDE will need to reason about several other types of knowledge. However, I will not attempt to push the state of the art in these areas; rather I will try and utilize simplistic approaches to the following topics:

1. A naive model of statistical processes. How different copies of an enzyme can be doing different things at once.

2. Simple spatial reasoning. PEPTIDE will need to reason about the relative location of interacting regions on nucleic acid chains and also about complementarity of the conformations of active sites and the objects that might fit in them.

3. The difference between the structure and function of certain objects. I will attempt to transport the concepts of [Davis 82] from the domain of hardware troubleshooting to that of molecular genetics.

Although I am currently restricting my attention to the problem of prediction, the structure of the PEPTIDE system should be able to support other types of reasoning such as measurement interpretation and planning.

1.2 Goals in the Development of PEPTIDE's Language

A language for building qualitative models of different biochemical processes is a prerequisite for an AI system that can reason about the molecular mechanisms of genetic expression and enzymatic action. Actually several languages may be necessary to support reasoning which requires several layers of abstraction; however, for now I am concentrating on only the lowest level. Motivated by the success of [Davis 82] and by the "No Structure in Function" principle of [de Kleer 82a], I am assuming that this language will distinguish between two types of knowledge:

1. Structural information which describes how the macromolecular objects are physically characterized and related. An example would be the number of active binding sites in an enzyme or the concentration of a substrate.

2. Functional information which describes the processes which cause change in a system. An example would be the catalytic action of an enzyme or a ribosome. Since the functional information will describe causality (or influences), a method for expressing precise temporal relationships is necessary.

Given these assumptions, here are my criteria for judging PEPTIDE's language. It is instructive to compare my criteria to Forbus' desiderata for qualitative dynamics theories [Forbus 83].

1. The language should be clean. If the difference between A and B is important for reasoning about

that system, then the language should be able to represent the distinction concisely. Another way of looking at this requirement is as a modified version of Forbus' first property for Qualitative Process (QP) theory: "a dynamics theory must explicitly specify direct effects and specify the means by which effects are propagated".

2. The language should allow process descriptions to be composed to form macro descriptions. These compositions should support the same types of reasoning as the original primitives, yet they should be able to conceal the details of their construction (e.g. provide abstraction).

3. The language should be complete. All common mechanisms should be expressible in the language. In his quest for generality, Forbus does not consider completeness. However I wish PEPTIDE to be capable of reasoning about simple spatial relations and other types of information which are outside QP theory.

1.3 Paper Overview

Section 2 discusses previous work both on theories of qualitative reasoning about processes and change and on applications to molecular genetics. Section 3 presents a brief discussion of molecular biology sufficient to allow a computer scientist to understand the issues in the paper. Section 4 describes PEPTIDE's structural representation. Section 5 describes PEPTIDE's functional representations, both discrete and continuous, and aggregation, a technique which generates a continuous representation from a discrete one. Two examples are presented in section 6, the enzyme RNA polymerase, and a repressor operon; a simplified version of the operon is simulated. Finally, in Section 7 several conclusions and problems are listed.

2. Previous Work

PEPTIDE has been influenced by previous applications to genetics and existing theories of qualitative representation of physical processes. But with regard to both influences, PEPTIDE has evolved considerably.

2.1 Other Representations of Physical Change

PEPTIDE fits into the Naive Physics [Hayes 79a] concept as an instance of a cluster: a domain-specific package of reasoning techniques.

2.1.1 Quantities

My work is strongly influenced by the ideas in Qualitative Process theory [Forbus 83]. QP theory may be thought of a naive physics cluster, since it is an internally-cohesive reasoning system. QP theory has a very general notation: it may be capable of representing a model of any continuous process. But although it is partially applicable to many domains, it must be augmented by other forms of reasoning in almost every case.[3]

From QP theory I take the notion of a quantity space and that of individual views. The value of any parameter (or quantity) is a number whose meaning depends on the quantity's quantity space. A quantity space is a partial order on a set of possible values (which I call distinguished points or d-points, but Forbus calls limit points) for that quantity. Considerable simplification is achieved when reasoning about a specific quantity by considering only the relations between the current value and the d-points in the quantity space, instead of dealing with an actual real valued number.

2.1.2 Multiple Views

To model changing aspects of individual objects, I use Forbus' notion of individual views. Multiple views of an object are useful because they express multiple ways of considering that object.[4] I expect to implement each individual view as a set of conditional relations. If the conditions are true at a certain time then the relations are true. An object will often have a view whose conditional part is always true; the relations specified by these views are always applicable.

---

[3] For example, Forbus has used QP theory to describe fluid flow, but he required extra-QP theory logic to model pipe connections in the description.

[4] For example, multiple views are frequently used to represent the fact that a single molecule can act both as a bindee and as a binder.

2.1.3 Process Models

QP theory has also influenced my notion of process. In PEPTIDE, as in QP theory, the following assumption is made: all changes are caused to the world solely by the activity of one or more ***processes***. The further assumption, that all possible processes are known, allows reasoning by exclusion. This approach differs from those of [Rieger 77, Rieger 75] and [Doyle 83] where several types of causal links are described. I believe that a profusion of causal relations is an unnecessary complication which obscures the true nature of physical action.

Forbus' representation of physical processes seems very general (perhaps it can represent any continuous process) and quite powerful (since it supports a continuous type of analysis in which interactions between processes can be handled). Unfortunately, it is not always convenient to consider processes as continuous; some processes (like collisions and many biochemical mechanisms) are discrete or discontinuous. Forbus reasons about these discrete situations with ***encapsulated histories***, which are histories formed by the concatenation of sequential process episodes. For example, a collision can be represented by an encapsulated history composed of the episode where the ball is moving towards an obstacle followed by an episode where it is moving away. These encapsulated histories are quite useful for certain types of reasoning[5] but because they are histories, not processes, they are not generally composable.

Originally I hoped to extend QP theory to handle discrete processes through the addition of a multigranular temporal representation where the "discrete" processes simply acted much more quickly than the "continuous" ones. But this did not solve the problem of summarizing the apparently discrete actions of the faster processes to the slow ones (to produce a summary, some sort of vocabulary is required) and it added the problem of handling intuitively impossible interactions between the fast processes.

My solution is to use two process models: a ***discrete model*** of molecular mechanisms and a ***continuous model***, derived from QP theory, of aggregate behavior. The domain-specific discrete model is composed of five atomic processes at the molecular level[6] and contains all information about the system. Although these primitives might be viewed as encapsulated histories, they are not! Each one is an instantaneously acting process that may not be decomposed into historical episodes.

This representation of process resembles the representation of state space operators as production rules. A similar model was used as the qualitative representation of geologic processes in [Simmons 83] for the task of

---

[5] Like energy analysis [Forbus 83].

[6] These primitive operations are "atomic" in the sense of a computer's test-and-set instruction: they either happen completely or not at all. Since the operations act on objects of type molecule (which are formed of atoms), there is potential for terminology confusion .

geologic map interpretation.[7]

2.1.4 Multiple Models

In our view, a model has some specific information (the data) which is expressed in a representation that provides routines for manipulating that data (the representational framework [Simmons 83]). Therefore, programs which use multiple models may be contrasted in two dimensions: the overlap of data in the models and the overlap of representational frameworks for the models.

Using multiple domain models is a relatively old concept. MACSYMA [MACSYMA 83, Moses 71] increased algorithmic efficiency by using three different representational frameworks for the polynomial data type. The new database for the Programmer's Apprentice project uses two representational frameworks for program plans [Rich 82].

[Simmons 83] describes a program for geologic map interpretation which has five different representational frameworks for different types of data. [Davis 82] also uses this approach in programs for hardware troubleshooting.

ABEL, a program for diagnosing acid base disorders [Patil 81a, Patil 81b], is interesting because it creates a patient specific model (PSM) which is formed of multiple views of the same data at different levels of abstraction using the same representational framework. At each level of the PSM, causal relations among normal or abnormal states are represented in a semantic network. The PSM's different levels provide a remarkable breadth of coverage: depending on the level, a causal link could signify an empirical disease association or the interaction between ion transport systems which cause the association. Initial formulation, aggregation, elaboration, and projection are four operators that ABEL uses to build and extend the multilevel PSMs. Yet the power gained by the multiple use of a single representational framework serves to emphasize how important consistency is for a representational framework: ABEL is flawed because its notion of cause is ambiguous.

PEPTIDE represents processes in a genetic system at different abstraction levels with different representational frameworks: a discrete framework for the mechanistic level and a continuous one for the statistical level. A technique called ***aggregation*** builds a continuous model from a discrete model, much as ABEL's aggregation moves causal knowledge up an abstraction level in the PSM. PEPTIDE will use

[7] Simmons uses the term "kinematic" to refer to a model that I would label discrete, and the term "dynamic" where I say "continuous". Forbus uses the terms with meanings derived from Classical Mechanics. Thus QP theory [Forbus 83] is a "dynamics" theory, not because it is continuous, but because it deals with processes which cause all change. Forbus calls a theory "kinematic" if it deals with the connections and locations of objects, not whether the theory is discrete or continuous. I will follow this latter usage.

information about historically recent cycles in the process structure and knowledge about the relative significance of quantity values to determine when to shift reasoning activities between levels. Because its problem definition is smaller than ABEL's, PEPTIDE doesn't need to move knowledge between the different models as frequently as ABEL does. Thus, PEPTIDE's analog to ABEL's elaboration operator is quite simple and there is no analog to projection.

PEPTIDE's process models are quite similar to the (independently developed) theory of evaporation which is described in [Collins 83, Collins 82]. After analyzing protocols, Collins and Gentner postulated that human reasoning about evaporation is performed using mental models at three abstraction levels: macroscopic models, aggregate models, and molecular models. Collins and Gentner claim that there are multiple models at each level, and that each high level model is supported by multiple models at the next lower abstraction level (i.e. a macroscopic model is supported by several aggregate models, etc.) As yet, they have not advanced a mechanism for information flow between models at different levels or a trigger for reasoning shifts from one level to another.

2.1.5 Reasoning about Processes

Several people have worked on predicting the behavior of qualitatively-specified systems which have processes as the sole cause of physical change. As a result, the terminology has become quite confusing. Envisioning was introduced in [de Kleer 75] with the loose meaning: to take the qualitative description of a physical system and predict its future behavior. Since a qualitative description may be under-constrained, the device behavior could be ambiguous. Thus the envisioner returned the list of all possible behaviors [de Kleer 79]. In recent papers, however, de Kleer and Brown have changed the definitions. In [de Kleer 82a] they describe it as the first step of qualitative simulation: envisioning takes a structural device description and produces a functional description called a causal model (also called an envisionment). Then a technique called running takes the causal model and generates the specific behavior of the device, completing the qualitative simulation. But in [de Kleer 82b], they say that qualitative simulation is a degenerate form of envisioning; apparently, this is because a form of qualitative simulation is a useful tool for (as well as the goal of) envisioning. To further complicate matters, envisioning is equated with qualitative simulation in [Kuipers 82]. Because of this confusing terminology, I will attempt to avoid the word.

Qualitative simulation I take to mean a general procedure which produces one possible behavior of a system given a qualitative description of the system. Limit analysis [Forbus 83] is the backbone technique that is useful in the qualitative simulation of processes that are described in QP theory's continuous style.

Imagining [Simmons 83] is another technique for predicting system behavior. It differs from Forbus' qualitative simulation in several ways. Simmons' discrete process model neglects process interactions--an

unnecessary complication for the task of geologic interpretation. Another difference is imagining's use of quantitative and diagrammatic information.

In PEPTIDE there are two levels of qualitative simulation. Simulation at the mechanistic level is similar to imagining without the quantitative and diagramatic aspects. However, sometimes it is appropriate to use higher level reasoning. Aggregation is a technique which takes the discrete model of enzyme mechanism and produces a continuous model. This continuous model may be subjected to limit analysis as part of the predictive process.

2.1.6 Representations of Change

By using ***histories*** [Hayes 79a] to represent the predicted system behavior, PEPTIDE escapes from some of the complexities of the frame problem. Unlike the situational calculus which represents change as a string of spatially unbounded, temporally specific situations, PEPTIDE will store information about change as a collection of histories each of which is temporally unbounded but spatially limited to a single object. As noted in [Forbus 83] this substitutes the (presumably easier) history interaction problem for the frame problem. I also use the concept of a history ***slice***: the spatially limited situation that is generated by considering the value of a history at a given instant [Hayes 79b].

2.2 Other Applications to Biochemistry

Although they have not been implemented, the designs of some of my earlier projects in biochemistry have influenced the architecture of PEPTIDE. In [Weld 81a] I described the design of a system for *Computer Aided Organic Synthesis*. Critical to that design was the RSL language [Weld 81b] for specifying the kinetics of organic chemical reactions. In [Weld 82] I described the architecture of a CAD system for genetic engineers which would aid them in the design of new organisms at three levels: construction of enzymes, generation and kinetic analysis of pathways, and the developmental change of pathways over time. Because of its representation of gene expression, PEPTIDE would be an important component of the CAD subsystem which generates developmental control.

MOLGEN and GENEX are two other recent AI systems with molecular genetics as a problem domain. MOLGEN [Stefik 81a, Stefik 81b] is a system for planning experiments in molecular genetics. Through the use of hierarchical planning and least commitment operator selection with constraint propagation, it worked for a small number of examples. Yet its domain knowledge is not as detailed as PEPTIDE's will be, and it lacks a deep understanding of the ways genetic expression is regulated.[8]

[8] I.e. it manipulated "promoters", but did not have a model of transcription.

GENEX [Koton 83] hypothesizes operon level mechanisms of regulation given data on observed expressional behavior. The most important difference between GENEX and PEPTIDE is the kind of knowledge that each use to understand genetic systems. GENEX's knowledge is a set of "facts or 'rules' grouped into procedures according to the region of the operon to which they refer" [Koton 83]. The information in the rules is empirical rather than mechanistic in nature. For example, a typical GENEX rule might say "If no gene transcript is made then maybe there is a defective promoter region." GENEX recognizes possible operons by following a fixed investigative flowchart and "matching the right facts to the observed phenomena" [Koton 83]. This leads to several problems including inadequate explanations, inability to handle multiple mutations, and the inability to reason about nonstandard operon types. Still GENEX performs quite well in a number of cases.

PEPTIDE, on the other hand, will use a detailed model of the molecular mechanisms involved with individual biochemical process activity. Also PEPTIDE has a more general control flow: reasoning switches between discrete and continuous models depending on the specific case at hand. This approach should alleviate some of the problems of GENEX.

3. Biology Background

The following concepts from molecular biology will clarify the aims of this research. Be forewarned that this chapter is a gross simplification of currently accepted theory. For a more detailed explanation, see [Wood 81] or [Watson 77].

Living organisms are divided into two categories: prokaryotes and eukaryotes. Eukaryotic cells are distinguished by the presence of a membrane enclosed sack, called a nucleus, where all genetic activity occurs. Eukaryotes are very complicated and not well understood. Prokaryotes, on the other hand, are smaller, simpler and considerably better understood. PEPTIDE will only consider (single celled) prokaryotic organisms.

3.1 Biochemistry

PEPTIDE needs a detailed model for two classes of chemicals, proteins and nucleic acids, which form the foundation of the molecular mechanism of life. ***Proteins*** are long sequences of amino acids linked together end to end. Once formed, each protein folds into a specific three dimensional shape called a ***conformation***. Although proteins have other functions, e.g. structural, we will be most interested in the class of proteins, called ***enzymes***, which catalyze chemical reactions. Enzymes often form conformations with neatly shaped holes or clefts, called ***active sites***, into which other molecules may fit. When a substrate molecule (which could be another protein, DNA or any other molecule) fits into the active site (a process called ***binding***[9]) the enzyme may change its conformation by folding differently or catalyze some reaction. In fact, enzymes speed up chemical reactions so much that by comparison uncatalyzed reactions just don't happen.

The nucleic acids, DNA and RNA, are very long strings of monomeric units called nucleotides. In both cases only four nucleotides are used to form the chains. DNA is a very stable molecule which is uniquely suited to store vital information for long periods of time by encoding it with its alphabet of four nucleotides. Whenever a cell divides, it must copy its DNA (through the action of enzymes) with a process called replication. See Figure 1.

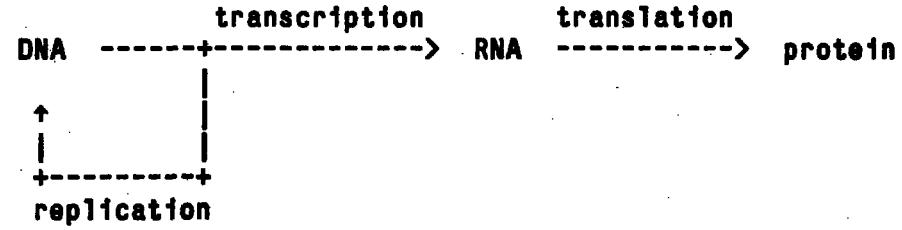

Figure 1. Gene Replication and Expression

[9] The term "bind" is somewhat ambiguous in this document. In addition to the meaning given above, there is variable binding.

To influence the actions of a cell, the information stored in the DNA must be converted into protein. This action is called ***gene expression*** and is accomplished with a two step process. The first step, called ***transcription***, copies an individual message, or gene, from DNA into RNA. This process is carried out by the enzyme RNA Polymerase (RNAP) by a four-step process. First RNAP binds to a specific signal region (called a ***promoter***) that is adjacent to the gene. Then the enzyme slides towards the gene until it recognizes the gene's start signal. Methodically the RNAP enzyme slides along the gene, at each step adding to its growing RNA chain a nucleotide complementary to its current position on the DNA template. Finally, when the enzyme recognizes a stop signal, it drops off the DNA and frees its messenger RNA.[10] The second step, called ***translation***, takes the RNA transcript and from it builds a protein. Ribosomes execute this function by dividing the transcript into small subsequences, called codons, formed of three nucleotides each and associating each codon with a specific amino acid. Although ribosomes are not really proteins (since they are partially composed of RNA), their function is to catalyze the formation of chemical bonds linking amino acids into proteins. The actual mechanism of translation is even more complicated than that of transcription.

3.2 Regulation of Gene Expression

Since variations in the promoter regions affect RNA polymerase binding activity, promoter efficiency determines the efficiency of transcription and, ultimately, that of expression. Thus by putting a efficient promoter before an important gene and poorer promoters before genes whose products are required less, an organism can produce proteins in different quantities.

Genes such as those described above are called constitutive because they are always expressed. Yet some genes are needed rarely, but in quantity. For example, consider the case below.

Gene1 codes for protein1 which functions as the enzyme which catalyzes the critical reaction in a biosynthetic pathway which makes molecule1 from much smaller chemicals which are always available. Making molecule1 via this pathway is quite energy intensive, but the cell tolerates the expense because molecule1 is necessary for life and is almost never available from the environment.

However, sometimes molecule2 is available from the environment and a very simple reaction (catalyzed by some different enzyme) cuts molecule2 into molecule1 and another useful molecule. If molecule2 is available, we would like to avoid making molecule1 the expensive way. One kind of regulation works by turning off (repressing) the expression of gene1 into protein1 whenever molecule2 is present. A third area of DNA, called an ***operator***, is added between the promoter and the gene. See Figure 2.

[10] An example later in the paper shows how to describe the mechanism for RNAP in PEPTIDE's discrete model.

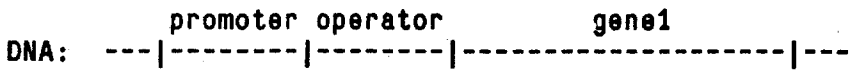

Figure 2. A Repressible System.

A different gene (say gene2) which is constitutive produces a repressor protein for gene1. The repressor protein spends most of its time doing nothing. But whenever it sees molecule2 it grabs it in one of two active sites. This binding causes it to change its three dimensional conformation, altering the shape of its other active site. As a result of this change the remaining active site is now shaped to recognize and bind to a specific pattern of DNA: the operator region near gene1. Once the repressor binds to the operator, however, RNA polymerase is blocked from moving from its binding site (the promoter region) and transcribing gene1. So the repressor protein acts as a switch: as soon as it sees molecule2, it turns off transcription (and thus expression) of gene1. Since all proteins are broken down regularly, this will effectively stop the action of the costly production line for molecule1 which will be made exclusively from molecule2. And if the supply of molecule2 ever ends, then the repressor protein will return to its original conformation and release its operator, causing renewed production of protein1.

This is just one simple example of a series of regulatory mechanisms which have been discovered. When a specific promoter, operator, gene and regulatory proteins are considered together, they are called an ***operon.*** The second example in section 6 shows how PEPTIDE can predict the behavior a repressor operon like the one described above.

4. Structural Representation

This chapter presents facilities for describing the physical relationships and structural organization of the molecular objects in a given genetic system. The structural aspects may be divided into three classes: object definitions (and modifiers); predicates about the conditions of an object, history or slice; and functions which return information about objects, histories, and slices.[11]

4.1 Object Definitions

Objects may be considered in a variety of views: bindee, binder and chain.

A molecule may be viewed as a bindee if it can be bound to some other molecule's active site. Therefore, since almost every molecule can be bound in SOME active site, almost every object will have at least one bindee view.

An object may be viewed as a binder if it has active sites that can take on conformations and bind bindees.

An object may be viewed as a chain if it is composed of an ordered list of monomers (i.e. members of some set of molecules e.g. amino acids). Chains may be considered as a tree in the following manner: each internal node is an interval (or sub-chain) and each leaf node is an individual monomer. Thus the order on descending edges at each internal node specifies a global ordering on the monomers. Any position on a chain (i.e. any leaf node) may be a bindee.

* (DEF-SET name elt1 elt2 ...)

* (DEF-OBJ name view1 view2 ...)
  Each object can have many different views, including both different kinds and multiple views of the same type.

  - (BINDEE optional-name)
    The bindee view is the simplest. The name is only necessary if multiple bindee views are attached to this object. If so each name points to a different conformation of part of the object. If no name is given then the object name is the pointer to the conformation.

  - (BINDER site1 site2 ...)
    Each binder has one or more sites (defined separately with the DEF-SITE command) which may be used to bind bindees.

  - (CHAIN (interval1 interval2 ...) monomer)
    As explained above chains are ordered lists of monomers that are conceptually regarded as trees where the internal nodes are intervals. This view defines one level of a given sub-tree.

[11] From this section forward through the rest of the paper, I specify the syntax of a descriptional language. The primary design goal in the syntax was ease of writing this paper--narrow examples between wide margins... In the actual implementation, many things will be changed--the scoping of various commands most notably.

* (DEF-SITE name conformation1 conformation2...)
  Shapes describe the three dimensional conformation of a molecule or an active site. PEPTIDE knowledge of these shapes is quite limited: it knows when two conformations are complementary (i.e. they fit together), but that is all.

* (DEF-QREL relation d-point1 d-point2)
  Add the relation between the two distinguished points to the database. Possible relations include: greater-than (>), much-greater-than (>>), equal (=), less-than (<), and much-less-than (<<). The distinguished points do not need to refer to the same quantity.[12] Thus if one wished to say that the saturation level of glucose was much greater than the starvation level, one might say:

```
(DEF-QREL >> ((NUMBER-OF glucose) saturation)
             ((NUMBER-OF glucose) starvation))
```

Objects with binder or chain views are represented as hierarchies; the "/" character separates descriptors at different levels. Thus we can refer to the "control-site" site of the "frobase" binder by saying "frobase/control-site". Prolog style variables (denoted with a prefix question mark) may be used at any point; thus we can refer to any binding site of "frobase" by saying "frobase/?foobar".

4.2 Predicates

These predicates return T or NIL and are used mainly to test the conditions of an object, history or slice inside the preconditions of a process specification.

* (AT-END? pos dir)
  Is the position on the chain the endpoint for that chain in the direction specified?

* (BOUND? place-on-object site)
  Is the site binding the object at the specified bindee place? If the object has only one bindee place, then the object name is enough of a specifier, otherwise use: object/bindee-place.

* (BETWEEN? p q r ...)
  Returns true if all arguments are positions on one chain and the order of the arguments is a consistent spatial ordering. E.g. p <= q and q <= r etc.

* (HAS-VIEW? view-type object1 object2 ...)
  Do the objects listed all have a view of the type specified (i.e. either BINDEE, BINDER, or CHAIN)?

* (IS-INTERVAL? a b ...)
  Are these things intervals, either on a chain or between distinguished points in a quantity space?

* (IS-POINT? a b ...)
  Are these things points, either in a quantity space or between intervals in a chain?

* (IS-SITE? x1 x2 ...)

[12] Thus a qrel unifies the QP theory notions of quantity conditions and correspondences [Forbus 83].

Are the arguments active sites on some binder?

4.3 Functions

The functions return information about the structure of an object, history, or slice. Functions are typically used for two purposes: as part of predicates in process preconditions, and as a means of addressing objects or their parts.

* (C site)
  Refers to the contents of a site.

* (CODES-FOR pos-on-rna-chain)
  Returns the amino acid that is coded for by the three nucleotides, read LEFT to RIGHT, from the position on a RNA chain.

* (COMPLEMENTARY conformation)
  Returns the shape of objects which will bind to objects with the parameter's conformation. Conformations are simply atoms that are related by the equality and complementary relations.

* (DIST pos1 pos2)
  Returns the distance between two positions on a single chain in units of chain members. Returns infinity if the two are not on the same chain.

* (END chain dir)
  Returns the position which is the dir-most end to the chain. If the chain is circular and has no end, then NIL is returned.

* (JOIN-CHAINS c1 c2 ...)
  Returns the chain that would be created if a reaction were to link the argument chains together in order. All of the arguments must be the same kind of chain. If one of the arguments is a short chain with no interval abstraction (e.g. a single nucleotide), then it gets appended to an existing interval from one of the other arguments.

* (NUMBER-OF object)
  Returns the number of objects present.

* (P pos dir dist)
  Refers to the point on a chain which is dist chain-member-units along the chain in the appropriate direction (RIGHT or LEFT) from the starting position.

* (PARENT x)
  Returns the parent of a sub-object in the structure hierarchy of a particular view. E.g. site -> enzyme, position -> interval, interval -> chain, etc.

* (S conformation)
  Refers to the set of objects which have the specified conformation. This is useful in conjunction with the COMPLEMENTARY relation.

5. Process Representation

PEPTIDE is based on the assumption that any change in the world is caused by one or more processes, and that all possibly active processes are known. There are two types of process representations: discrete and continuous. The discrete representation is more detailed--with it the mechanistic action of individual enzymes may be described. Continuous processes are represented using QP theory [Forbus 83]; they are more abstract than discrete processes, and are thus useful for describing the statistical behavior of a genetic system. With the aggregation technique PEPTIDE will create a continuous process model from a discrete model. Thus behavior-predicting reasoning can switch back and forth between models as necessary.

5.1 Discrete Processes

By discrete process I mean a process that occurs in an instant; i.e. is atomic, it either happens completely or not at all. A model of some specific genetic system is described by writing a set of discrete process instances. These instances are formed by refining combinations among the five generic discrete processes: bind, drop, react, fold and slide.[13] Thus, the generics act as building blocks in the construction of actual instances.

5.1.1 Generic Discrete Processes

Many molecular genetics mechanisms may be described using combinations of the following five discrete primitives. Although they are described as actual processes, they are generic (underspecified); from them instances may be defined as is described in the next section.

For each of the generics I will give an English description, a list of pre and postconditions, and a first order predicate calculus definition (except for REACT).

5.1.1.1 (BIND site object)

This attaches the object to the binding site specified.

Preconditions The site must be empty. There must be at least one bindee view of the object which isn't already bound and has a conformation complementary to the conformation of the site. There must be at least one copy of object and at least one copy of the object with the site. The bindee must not be the same object as the binder.

Postconditions The object is bound to the site.

[13] Note: although this set of five generics can be used to represent most of the common biochemical processes, some spatially complex processes can not be described.

```
(forall
  ((t time) (s site) (x bindee))
  (implies
    (bind x s)@t
    (and (forall
           ((a bindee))
           (not (bound? a s)@t))
         (forall
           ((other-s site))
           (not (bound? x other-s)@t))
         (equal (conformation x)@t
                (complementary (conformation s)@t))
         (> (number-of x)@t zero)
         (> (number-of (parent s))@t zero)
         (not (equal x (parent s)))
         (exists ((t2 time))
                 (and (< t t2)
                      (forall
                        ((t1 time))
                        (implies (< t t1 t2)
                                 (bound? x s)@t1)))))))
```

Thus (bind x s) at time t implies that all the preconditions for bind are satisfied at time t, and there exists a time t2 such that for all times between t and t2, the object is bound to the site.[14]

5.1.1.2 (DROP object site)

This unbinds object from site.

Preconditions The object must be bound to the site.

Postconditions The object is not bound to the site.

```
(forall
  ((t time) (s site) (x bindee))
  (implies
    (drop x s)@t
    (and (bound? x s)@t
         (exists ((t2 time))
                 (and (< t t2)
                      (forall
                        ((t1 time))
                        (implies
                          (< t t1 t2)
                          (not (bound? x s)@t1))))))))
```

[14] In this logic formalism, the variable T2 replaces the persistence values described by [McDermott 82].

5.1.1.3 (REACT del-list add-list site-list)

REACT designates a covalent reaction that might be catalyzed by an enzyme. Thus when active this process acts as an influence on the concentrations of molecules listed. The sole precondition for this process is that all the objects in del-list must be bound to sites on the same enzyme.

Preconditions All the objects in del-list must be bound to sites on the same binder object, the enzyme.

Postconditions There is a negative influence on the concentration of all the objects in del-list and a positive influence on the molecules in add-list. Site-list contains a list of site-object pairs that tell which sites (if any) the members of add-list are bound to.

5.1.1.4 (FOLD site conformation)

This process changes the conformation of the site to the new conformation specified. This is important because the binding activity of a site depends on its shape.

Preconditions None.

Postconditions The site has the conformation specified.

```
(forall
  ((t time) (st site) (new-conf conformation))
  (implies (fold s new-conf)@t
           (exists
             ((t2 time))
             (and (< t t2)
                  (forall
                    ((t1 time))
                    (implies (< t t1 t2)
                             (equal (s st)@t1 new-conf)))))))
```

5.1.1.5 (SLIDE site chain dir)

This process causes the enzyme with the named site (assumed to be bound to the chain) to slide along that chain in the direction specified for a distance equal to the length of one of the chains unit members.

Preconditions The chain is bound to site at some position p. P is not the dir-most end of the chain. There is no other object bound to the chain in the way.

Postconditions The chain is bound to site at a position one unit to the dir of p.

```
(forall
  ((t time) (s site) (c chain) (d dir))
  (implies
    (slide s c d)@t
    (exists
      ((p-on-c (position-on c)))
      (and (bound? p-on-c s)@t
           (not (at-end p-on-c dir))
           (not (exists
                  ((blocker binder))
                  (bound? (p p-on-c dir) blocker)))
           (exists
             ((t2 time))
             (and (< t t2)
                  (forall
                   ((t1 time))
                   (implies
                     (< t t1 t2)
                     (bound? (p p-on-chain) s)))))))))
```

5.1.2 Process Instances

The DEF-PI construct may be used to specify a process instance that describes an actual process that exists in a specific genetic system of interest.

* (DEF-PI owner name instance-spec)
  The process owner[15] is the object or object view which has the behavior described by the process instance. This attachment of processes to owners allows PEPTIDE to reason about the differences in process activity that would occur if their were a million copies of an owner instead of one. The name is local to the owner; thus the process below would be refered to as "enzyme1-twiddle". The instance spec must contain at least one generic process with all arguments specified, either as a fixed identifier (e.g. repressor/control-site) or as a variable (e.g. ?site) which could be free or already bound to an identifier. In addition, the IF construct allows a user to add further preconditions to those of the generic process thus restricting the situations where the process instance will be active.

For example a user might describe the following process instance:

```
(DEF-PI enzyme1 twiddle
    (IF (> NUMBER-OF(glucose) threshold1)
        (BIND enzyme1/?any-site glucose)))
```

In other words, if the number of glucose molecules is above a specific threshold value (i.e. a d-point in the quantity space of the glucose concentration), then enzyme1 will bind the glucose object at any free site (on enzyme1) whose conformation is complementary to that of glucose. (Mentioning the active site of any enzyme other than that of the process owner would be very bad style, since it would violate the no-structure-in-function prinicple [de Kleer 82a].)

---

[15] The process owner is usually an enzyme.

5.1.3 Macros

Because it isn't always convenient to instantiate a process directly, a macro facility is provided. Generic process macros, defined with the DEF-GPM command, simply expand into one or more generic processes whenever used.

* (DEF-GPM name param-list proc-spec1 proc-spec2 ...). This defines a macro which will expand into a new process at read time. Whenever (name actual-params) gets read, the actual parameters will be substituted for the formal parameters in param-list in the proc-spec bodies.

For example, consider the following macro for moving an object from one active site to another on the same parent object. It expands into two processes: the first causes the object to be bound to the destination site (while it is still bound to the source site) and the second drops the object from the source site once it is bound to the source.

```
(DEF-GPM MOVE (from-site to-site)
    (IF (AND (IS-SITE? from-site to-site)
             (EQUAL (PARENT from-site) (PARENT to-site))
             (BOUND? ?obj/?old-place from-site)
             (NOT (BOUND? ?obj/?new-place ?other-site))
             (EQUAL (S ?obj/?new-place)
                    (COMPLEMENTARY (S to-site)))
             (NOT (BOUND? ?y to-site)))
        (BIND ?obj/?new-place to-site))
    (IF (AND (IS-SITE? from-site to-site)
             (EQUAL (PARENT from-site) (PARENT to-site))
             (BOUND? ?obj/?old-place from-site)
             (BOUND? ?obj/?new-place to-site))
        (DROP ?obj/?old-place from-site)))
```

Because the macro binds both sites in the first process, the macro will often act as though it were atomic. However, it is possible to construct cases where it won't act indivisibly; socare must be exercised.

5.1.4 Simulation of Discrete Processes

It is straight-forward to simulate the effects of discrete processes. First, PEPTIDE collects all the process instances that are ***active***, i.e. instances whose preconditions are satisfied. The objects, which are affected by the postconditions of the active instances, are collected to see if any of the processes interact. All noninteracting processes are simply simulated and the histories of the modified objects are updated.

If two processes interact then there must be either one or more shared objects. If there is only one, then we assume that only one of the processes can actually operate (i.e. the object is acting like a semaphore). The simulator chooses one and continues despite the ambiguity thus generated. If there are multiple copies of the affected object, then the simulator assumes that some of the objects support one process and some support the

other; both of the interacting processes go on independently and in parallel.[16]

5.2 Aggregation

Discrete simulation is essential to the understanding of systems with processes that act on single objects (like repressors act on DNA operator sequences). Discrete simulation is helpless to predict the effects of statistically large numbers of similar processes or of processes that are repeated cyclicly. When PEPTIDE discovers situations like these, it uses a technique called aggregation to generate a continuous model of the active processes. Limit analysis [Forbus 83] may then be used to predict the system's behavior using the continuous model. Finally, these predictions are translated back to the discrete model.

The actual aggregation procedure is quite simple. The set of discrete processes to be aggregated are marked. An analyzer notes the net effect of the processes from all pertinent REACT and SLIDE actions and computes what influences they would have on continuous quantities if repeated. REACT processes affect NUMBER-OF quantities and SLIDE processes affect the POSITION of binders to chains. The set of influences generated by a single group of aggregated discrete processes forms a continuous process.

5.2.1 Triggering Aggregation

Aggregation can be triggered by two different types of situations: process cycles and the situations involving the >> relation. In both cases a set of discrete processes which would actively run many times are converted into a single continuous process.

Cycles in the recent process history will trigger aggregation because the whole loop could be repeated multiple times. All process instances which are members of the cycle are aggregated.

The much greater than (>>) relation may also trigger aggregation. Whenever there is an active process where the NUMBER-OF its owner is >> the NUMBER-OF all the objects affected by the process (besides the owner), that process could occur many, many times. Consequently, aggregation composes a continuous model of the effects of the process and reasoning shifts to limit analysis of the continuous model.

5.3 Continuous Processes

PEPTIDE uses a version of Qualitative Process (QP) theory [Forbus 83] as a representational framework for its continuous process models. In this section I will describe as few aspects of QP theory as necessary to understand the examples in this paper; for more detail, see [Forbus 83]. Qualitative simulation in QP theory has three phases: generating a complete process structure, determining the net influence on each quantity (i.e.

[16] I must think more about how statistical quantities of objects are divided up. Consider FOLD.

the sign of the derivative, or Ds value), and performing limit analysis. Qualitative simulation of the continuous models in PEPTIDE is somewhat simpler, because aggregation does much of the work of the first two steps. Instead of producing a set of process definitions, aggregation generates a set of influences and a set of quantity spaces with all known relations among distinguished points. All that remains is limit analysis.

5.3.1 Limit Analysis

Limit analysis determines the next distinguished point for a quantity that is being influenced: i.e. the first quantity value which would imply a change to the structure of active processes. For example if the quantity water-temperature is being influenced with a positive Ds (perhaps the water is on a stove burner) and the current value of water-temp is 70 degrees celcius, then the next d-point is 100 degrees when the boiling process would start.

To make its prediction, limit analysis requires the quantity's Ds value which tells the direction of the change and the q-space of the influenced quantity. A quantity space is simply a set of related d-points for a given quantity; in the example above, the q-space might consist of the water's freezing point, its boiling point, and the less than relation between them.

When limit analysis is used during a simulation, a new value for the influenced quantity must be assigned before resuming simulation at the dicrete level. This value is not the d-point, but a value infinitesimally before the d-point. This ensures that PEPTIDE will always use the more exact dicrete model when reasoning about the tricky ends of process intervals.

When multiple quantities are being influenced, more knowledge is necessary to unambiguously predict which quantity will first reach a d-point. One idea is to assume that each discrete process acts in unit time. This allows the continuous processes formed by loops to be ordered by the length of the process cycles from which they were formed. This may help some, but will not resolve all ambiguities. For now, I am ignoring this admittedly important issue.

6. Examples

In this section, I present two moderately complex examples: RNA polymerase (an enzyme which copies DNA to RNA) and a repressor operon (which selectively controls the expression of a gene). The predictive actions of PEPTIDE will be hand simulated for the second example.

6.1 RNA Polymerase

RNA Polymerase (RNAP) is a complex enzyme which acts to copy a section of one strand of DNA into a complementary strand of RNA. RNAP acts in four steps: binding, initiation, elongation and termination [Wood 81]. During binding, RNAP attaches to a specific promoter interval. During initiation, the first diphosphate bond is formed in a reaction that is sensitive to rifampicin and is driven by the production of a volatile biproduct. During elongation, RNAP moves down the DNA chain right to left (from 3' to 5') building a complementary RNA chain left to right. When RNAP detects a termination signal in the newly created RNA (which I assume to be the end of the gene region), it stops by releasing the chains.

6.1.1 Structural Description

For simplicity, I shall ignore the action of the sigma subunit and the effects of various antibiotics.

```
(DEF-SET nucs
    (DEF-OBJ a (BINDEE))
    (DEF-OBJ t (BINDEE))
    (DEF-OBJ g (BINDEE))
    (DEF-OBJ c (BINDEE)))
(DEF-SET rnucs
    (DEF-OBJ a (BINDEE))
    (DEF-OBJ u (BINDEE))
    (DEF-OBJ g (BINDEE))
    (DEF-OBJ c (BINDEE)))

(DEF-OBJ dna (CHAIN (after stop gene promoter before)
                    (MEMBER nucs)))

(DEF-OBJ rna (CHAIN () (MEMBER rnucs)))

(DEF-OBJ rnap (BINDER rna-site dna-site dna-site)
(DEF-SITE rna-site
    (start (COMPLEMENTARY (MEMBER rnucs)))
    (chain (COMPLEMENTARY (S rna/?position))))
(DEF-SITE rnuc-site
    (nil (COMPLEMENTARY (MEMBER rnucs))))
(DEF-SITE dna-site
    (nil (COMPLEMENTARY (S dna/?region/?pos))))
```

6.1.2 Discrete Process Description

The actions of RNAP may be defined by its six discrete process instances.

First RNAP binds to the DNA promoter region.

```
(DEF-PI rnap bind
    (IF (NOT (BOUND? dna/promoter/?x ?y))
        (BIND dna/promoter/?w rnap/dna-site)))
```

From the promoter it slides to the left until it reaches the gene.

```
(DEF-PI rnap position
    (IF (AND (BOUND? dna/?region/?x rnap/dna-site)
             (NOT (EQUAL ?region gene)))
        (SLIDE rnap/dna-site dna LEFT)))
```

Once on the gene, it grabs a ribo-nucleotide that is complementary to the first (deoxyribo) nucleotide on the gene.

```
(DEF-PI rnap grab
    (IF (AND (BOUND? dna/gene/?x rnap/dna-site)
             (EQUAL (S ?rnuc) (COMPLEMENTARY (S ?x))))
        (BIND ?rnuc rnap/rnuc-site)))
```

This is the initial reaction in the task of building the actual RNA chain.

```
(DEF-PI rnap begin
    (IF (AND (BOUND? ?rnuc rnap/rnuc-site)
             (NOT (BOUND? ?x rnap/rna-site)))
        (PARALLEL (MOVE rnap/rnuc-site rnap/rna-site)
                  (SLIDE rnap/dna-site dna LEFT))))
```

This is the inner loop reaction: it adds one complementary nucleotide to the growing RNA chain.

```
(DEF-PI rnap add1
    (IF (AND (BOUND? ?rnuc rnap/rnuc-site)
             (BOUND? ?rna rnap/rna-site))
        (PARALLEL (REACT (?rnuc ?rna)
                         (JOIN-CHAIN ?rna ?rnuc)
                         ((rna-site (END (C rna-site) RIGHT))))
                  (SLIDE rnap/dna-site dna LEFT))))
```

When the whole gene has been copied, RNAP stops by releasing the template DNA and the new RNA chain.

```
(DEF-PI rnap stop
    (IF (BOUND? dna/stop/?x)
        (PARALLEL (DROP dna/stop/?x rnap/dna-site)
                  (DROP ?rna rnap/rna-site))))
```

A critical question is how does PEPTIDE know that the chain it just dropped from the rna-site has a mRNA view that is complementary to the gene it was transcribed from. It will need this knowledge to correctly

predict the effects of processes which are owned by a newly expressed gene. Yet this deduction would seem to require some knowledge of loops and induction.

6.2 A Repressor Operon

The operon defined in this section and simulated in the next was described in section 3.2; a review of that section will clairify this presentation.

6.2.1 Structural Description

The structural description of this more complex example is divided into three sections: the operon definition, definition of other auxiliary enzymes, and statement of qualitative relations.

6.2.1.1 Operon Definition

The repressor operon has two enzymes which act on a single DNA sequence and on several effector molecules.

The promoter, operator, and gene regions of the DNA strand are especially important.

```
(DEF-OBJ dna (CHAIN (after stop gene operator promoter before)
                    (MEMBER nuc)))
```

The gene sequence of dna codes for the protein frobase, a simple enzyme that "frobs" the molecule foo to produce the molecule bar. Note that frobase has two views. The binder view is necessary for frobase's catalytic effect; yet another enzyme, protease, considers frobase a chain of amino acids (i.e. a protein) for its actions.

```
(DEF-OBJ frobase (BINDER cat-site)
                 (CHAIN () (MEMBER amino-acids)))
(DEF-SITE cat-site (nil (COMPLEMENTARY (S foo)))

(DEF-OBJ foo (BINDEE))
(DEF-OBJ bar (BINDEE))
```

The repressor protein has two sites: a control site, which can bind bar if there is enough around, and site that (when active) can bind to the operator sequence of dna but otherwise is closed.

```
(DEF-OBJ repressor (BINDER control-site op-site))
(DEF-SITE control-site
    (nil (COMPLEMENTARY (S bar))))
(DEF-SITE op-site
    (active (COMPLEMENTARY (S dna/operator)))
                (inactive CLOSED))
```

6.2.1.2 Auxiliary Enzymes

The definitions above suffice to specify the operon, but four more enzymes must be present for repression to work correctly: RNAP (which was defined in the last example), ribosomes, protease, and ribonuclease. Ribosomes translate messenger RNA into the proteins they code for; although their mechanism is very complex, a simple model will suffice for this example. Protease, an enzyme present in very low quantities, slowly destroyes proteins at random--effectively allowing new enzymes to replace old. Ribonuclease does the same thing but with the messanger RNA transcripts that are created by RNAP in the process of gene expression.

```
(DEF-OBJ ribosome (BINDER mrna-site))
(DEF-SITE mrna-site (nil
    (COMPLEMENTARY (S (CHAIN (MEMBER rnucs))/?p0))))

(DEF-OBJ protease (BINDER prot-site))
(DEF-SITE prot-site (nil
    (COMPLEMENTARY (S (CHAIN (MEMBER amino-acids))/?p1))))

(DEF-OBJ ribonuclease (BINDER rna-site))
(DEF-SITE rna-site (nil
    (COMPLEMENTARY (S (CHAIN (MEMBER rnucs))/?p2))))
```

6.2.1.3 Qualitative Relations

The only relations necessary are those that specify the relative concentrations of objects in the system. All calls to NUMBER-OF refer to initial conditions.

```
(DEF-QREL >> (NUMBER-OF repressor) (NUMBER-OF dna))
(DEF-QREL >> (NUMBER-OF rnap) (NUMBER-OF dna))
(DEF-QREL >> (NUMBER-OF foo) (NUMBER-OF frobase))
(DEF-QREL =  (NUMBER-OF dna) 1)
(DEF-QREL =  (NUMBER-OF protease) 0)
(DEF-QREL =  (NUMBER-OF ribonuclease) 0)
(DEF-QREL =  (NUMBER-OF rna) 0)
(DEF-QREL <  (NUMBER-OF bar) rep-threshold)
```

6.2.2 Discrete Process Description

The processes described below are organized according to the object that owns the process. RNAP's processes are descibed above.

6.2.2.1 Frobase

Whenever frobase sees the object foo, it binds it to the cat-site.[17]

```
(DEF-PI frobase bind
    (BIND frobase/cat-site foo))
```

[17] This is the first example of a process instance whose preconditions are not augmented by an IF.

Then it converts the foo to a bar object and releases it.

```
(DEF-PI frobase react
    (REACT (foo) (bar) ()))
```

6.2.2.2 Repressor

The repressor is a bit more complicated: it acts as part of a feedback mechanism. Whenever the concentration of bar is above a threshold, it binds bar to its control-site.

```
(DEF-PI repressor bind
    (IF (> (NUMBER-OF ?bar) rep-threshold)
        (BIND repressor/control-site ?bar)))
```

If bar is bound, then the op-site folds to the active conformation.

```
(DEF-PI repressor fold
    (IF (AND (BOUND? repressor/control-site ?bar)
             (EQUALS (S repressor/op-site) inactive))
        (FOLD repressor/op-site active)))
```

If the op-site has an active conformation, then the repressor tries to bind the operator segement of dna. This will effectively block rnap from transcribing the frobase gene.

```
(DEF-PI repressor block
    (BIND repressor/op-site ?op))
```

But whenever the concentration of bar goes below threshold, repressor drops it.

```
(DEF-PI repressor unbind
    (IF (AND (BOUND? repressor/control-site ?bar)
             (< (NUMBER-OF ?bar) rep-threshold))
        (DROP repressor/control-site)))
```

And if the control-site ever gets empty, the op-site folds back to the inactive conformation and drops whatever it currently holds.

```
(DEF-PI repressor unblock
    (IF (NOT (BOUND? repressor/control-site ?bar))
        (PARALLEL (FOLD repressor/op-site inactive)
                  (DROP repressor/op-site))))
```

6.2.2.3 Ribosome

The mechanism of ribosomes is very, very complex. However, for this example a gross simplicfication will suffice. Whenever the ribosome sees a piece of RNA it binds it.

```
(DEF-PI ribosome bind
    (BIND ribosome/mrna-site ?mrna))
```

Then it copies the mRNA into its protein and releases both the RNA and the protein.

```
(DEF-PI ribosome react
    (IF (BOUND? ribosome/mrna-site ?mrna)
        (REACT () (TRANSLATION-OF ?mrna) ())))
```

6.2.2.4 Protease and Ribonuclease

Because these two enzymes are similar in their recycling activities, they are described together. First, they bind any hapless object that matches their active site.

```
(DEF-PI protease bind
    (BIND protease/prot-site ?prot))

(DEF-PI ribonuclease bind
    (BIND ribonuclease/rna-site ?rna))
```

Then they destroy it.

```
(DEF-PI protease kill
    (IF (BOUND? protease/prot-site ?prot)
        (REACT (?prot) () ())))

(DEF-PI ribonuclease kill
    (IF (BOUND? ribonuclease/rna-site ?rna)
        (REACT (?rna) () ())))
```

6.3 Qualitative Simulation of Repressor Operon

The operation of the repressor operon is highly dependent on the activity of protease and ribonuclease. In the unrepressed situation, rnap makes a RNA transcript at about the same rate that ribonuclease destroyes it. At the same time, ribsomes are producing new copies of frobase from the RNA transcripts at the same slow rate that protease is destroying them. The unrepressed situation, therefore, is a steady state.

When frobase converts most of the foo objects to bars, this activates the repressor which binds to the operator sequence on the DNA. With the repressor bound to the operator, rnap can no longer produce RNA transcripts, so ribonuclease eventually destroyes all the transcripts. Once all the transcripts gone, the ribosomes can no longer produce new copies of frobase. Thus protease slowly removes all the frobase objects, effectively repressing frobase activity.

6.3.1 Simplified Simulation

Because it would be quite complex to simulate the situation described above, I am going to assume that the NUMBER-OF protease and ribonuclease is zero., which means that protease-bind, protease-kill, ribonuclease-bind, and ribonuclease-kill will never activate. Because of this assumption, the operon will not repressible. But it will still have interesting behavior. Rnap will initially make RNA transcripts that will be turned into frobase. Frobase will still covert foo to bar, thus activating the repressor. The activated repressor will still halt transcription by rnap. But because there is no protease, the NUMBER-OF frobase will never decrease.

The situation has fifteen possibly active processes: rnap-bind, rnap-position, rnap-grab, rnap-begin, rnap-

addl, rnap-stop, frobase-bind, frobase-react, repressor-bind, repressor-fold, repressor-block, repressor-unbind, repressor-unblock, ribosome-bind, and ribosome-react.

Since I assume that all bindees are initially unbound, only two processes will be active at the start: rnap-bind and frobase-bind. Because there are no process conflicts, both are assumed to run in parallel. Rnap-bind causes rnap to be divided into two classes: one has size one and has rnap bound to dna, the other class is larger and is unaffected. Frobase-bind causes all frobase objects to be bound to foos; there are now two classes of foos: bound and unbound.

Now two different processes are active: rnap-position, and frobase-react. These two instances may also be run in parallel. Rnap-position causes the single rnap object in the bound-to-dna class to slide to the beginning of the gene. Frobase-react causes a negative influence to the NUMBER-OF foo and a positive influence to the NUMBER-OF bar; all the frobase objects are now unbound.

Now three processes are active: rnap-bind, rnap-grab, and frobase-bind. The first two are simulated as was done above. But the activation of frobase-bind implies the completion of a process cycle with two members. The aggregator notes this and builds a continuous representation of the process cycle's encapsulated history, showing its two influences.

```
(I-MINUS (NUMBER-OF foo))
(I-PLUS  (NUMBER-OF bar))
```

But reasoning doesn't shift to the continuous level, because the model is not yet complete: there are still active discrete processes which have not been aggregated.

Discrete simulation continues with the a series of rnaps moving down the dna chain. The rnap-begin, rnap-grab, rnap-addl, rnap-grab, and rnap-addl processes are executed revealing another cycle.

Once again, the aggregator is called to build a continuous description. This cycle has three influences.

```
(I-MINUS (NUMBER-OF nucs))
(I-PLUS  (LENGTH-OF bound-rna-chain))
(I-PLUS  (POSITION-OF rnap/dna-site ON dna))
```

Since remaining active discete processes instances are part of the two discovered cycles, reasoning shifts to limit analysis at the continuous level. Limit analysis predicts that two things could happen: the frobase cycle could continue until the NUMBER-OF bar hits the value of rep-threshold which would cause the repressor-bind process to become active. Or the rnap cycle could continue until the POSITION-OF rnap/dna-site reaches the stop sequence which would cause rnap-stop to become active. Extra-theoretic heuristics will be necessary to decide which of these events would happen first. Since it doesn't much matter in this case, I assume that both events happen at the same time. Limit analysis complete, reasoning switches back to the

discrete level.

The NUMBER-OF bar is now just a hair less than rep-threshold, and the POSITION-OF rnap/dna-site is one nucleotide before the stop sequence. The two loops which were aggregated before, now continue for one more cycle each, then inactivate.

Rnap-stop becomes active ending the existing cycle and creating changing NUMBER-OF mrna from zero to one. The presence of mrna causes ribosome-bind to become active. This, in turn, activates ribosome-react which creates a new instance of frobase.

Meanwhile repressor-bind becomes active which causes repressor-fold and then repressor-block. The resulting binding of repressor/control to dna/operator prevents the activation of rnap-position and thus stops new rnap cycles from starting. Thus only one piece of rna is made.

However, no process is acting to decrease the NUMBER-OF mrna, so the NUMBER-OF frobase will continue to increase due to the effects of the ribosome cycle.

Thus one can see that repression of frobase does not work in this simplified model. There are now two cycles that are active: the original frobase cycle and a ribosome cycle which creates more frobase. Thus the eventual result (which would be determined by one more call to limit analysis) would be the depletion of foo by frobase.

6.3.2 Effects of a More Complete Model

A more accurate model would include copies of the protease and ribonuclease objects. These objects cause a steady negative influence to each enzyme. Simulation would be more complicated for this case for two reasons.

1. Many more object classes would be necessary to represent the increased object interactions. E.g. some frobase objects would be doing nothing, some binding foo, some being bound by protease, some binding foo while being bound by protease...

2. More heuristics will be required for correct prediction of inherently ambiguous, multi-influence, steady state situations.

7. Conclusion

7.1 Summary of Main Points

The following aspects of PEPTIDE are especially interesting:

* The specification of overlapping domain models that support different reasoning styles: one discrete, one continuous.

* The use of aggregation to generate a continuous model from a discrete representation.

* Using information about the relative significance of quantities (encoded with the much greater than (>>) relation) and the existence of cycles in the process structure to determine when to aggregate.

7.2 Problems and Future Work

There are also several problems.

* I don't know how the system will determine that the mRNA produced by several hundred process actions is actually complementary to its gene. This would seem to require some kind of loop analysis induction hair. Presumably I will kludge it somehow.

* The simplicity of interval-based spatial reasoning causes incompleteness. There are several mechanisms which can't be represented, e.g. bacteriophage lambda's DNA splicing activities or RNA secondary structure. I expect to leave this as a limitation for the foreseeable future.

* I must consider the nature of the heuristics that biochemists use to resolve the ambiguities generated by their qualitative models. Does the use of the > and >> relations on cycle length help significantly? Is suspect that it doesn't. These heuristics are probably empirical; I plan to ignore them at first.

* Biochemists have more accurate models of statistical behavior which support my discrete model. For example, there are cases where the approximation "(IF (> NUMBER-OF substrate) threshold) (BIND ...))" is not reasonable. How important are these lower level models?

* Another level with a more abstract model (operon level) may well be necessary to predict the behavior of really complex systems. This might be a good topic for a PhD thesis.

* Most importantly, PEPTIDE has not yet been implemented. This is just strarting and is certain to disclose numerous other shortcomings.

Acknowledgments

Dan Brotsky, Dave Chapman, Randy Davis, Dedre Gentner and Ken Forbus were exceedingly helpful in the design of PEPTIDE.

Phil Agre, Alan Collins, Dave Gifford, and Pete Szolovits also provided useful comments.

Jill Hoskins was a meticulous proofreader, but takes no responsibility for residual grammatical errors.

Kent Pitman helped me with EMACS, aided me with acronyms, and was a generally cool guy.

I am most appreciative of the people and environment at Bolt Beranek and Newman Inc.

References

[Collins 82]
Collins, A., Gentner, D.
Constructing Runnable Mental Models.
In *Proceedings of the Fourth Annual Conference of the Cognitive Science Society.* August, 1982.

[Collins 83]
Collins, A., Gentner, D.
Multiple Models of Evaporation Processes.
In *In Publication.* July, 1983.

[Davis 82]
Davis, R., Shrobe, H., Hamscher, W., Wieckert, K., Shirley, M., Poit, S.
Diagnosis Based on Descriptions of Structure and Function.
In *Proceedings of the AAAI.* August, 1982.

[de Kleer 75]
Johan de Kleer.
Qualitative and Quantitative Knowledge in Classical Mechanics.
AI-TR-352, MIT AI Lab, December, 1975.

[de Kleer 79]
de Kleer, J.
The Origin and Resolution of Ambiguities in Causal Arguments.
In *Proceedings of the Sixth IJCAI.* 1979.

[de Kleer 82a]
de Kleer, J., Brown, J.
Assumptions and Ambiguities in Mechanistic Mental Models.
Cognitive and Instructional Sciences Series CIS-9, Xerox PARC, March, 1982.

[de Kleer 82b]
de Kleer, J., Brown, J.
Foundations of Envisioning.
In *Procceedings of the AAAI.* August, 1982.

[Doyle 83]
Doyle, R.
Representing Change for Common-Sense Physical Reasoning.
Working Paper 243, MIT AI Lab, January, 1983.

[Forbus 83]
Forbus, K.
Qualitative Process Theory.
AI Memo 664A (revised), MIT AI Lab, May, 1983.

[Hayes 79a]
Hayes, P.
The Naive Physics Manifesto.
Edinburgh University Press, 1979,.

[Hayes 79b]
Hayes, P.
Naive Physics 1 - Ontology for Liquids.
Memo, Centre pour les etudes Semantiques et Cognitives, Geneva, 1979.

[Koton 83]
Koton, P.
A System to Aid in the Solution of Problems in Molecular Genetics.
MS Thesis, MIT Laboratory for Computer Science, May, 1983.

[Kuipers 82]
Kuipers, B.
Getting the Envisionment Right.
In *Proceedings of the AAAI.* August, 1982.

[MACSYMA 83]
The Mathlab Group.
MACSYMA Reference Manual.
MIT Laboratory for Computer Science, 1983.

[McDermott 82]
McDermott, D.
A Temporal Logic for Reasoning About Processes and Plans.
Cognitive Science (6):101-155, April, 1982.

[Moses 71]
Moses, J.
Algebraic Simplification: A Guide for the Perplexed.
Communications of the ACM 14(8), August, 1971.

[Patil 81a]
Patil, R.
Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis.
TR-267, MIT Laboratory for Computer Science, October, 1981.

[Patil 81b]
Patil, R., Szolovits, P., Schwartz, W.
Causal Understanding of Patient Illness in Medical Diagnosis.
In *Proceedings of the Seventh IJCAI.* 1981.

[Rich 80]
Rich, C.
Inspection Methods in Programming.
AI-TR-604, MIT AI Lab, June, 1980.

[Rich 82]
Rich, C.
Knowledge Representation Languages and Predicate Calculus: How to Have Your Cake and Eat it Too.
In *Proceedings of the AAAI*, pages 193-196. 1982.

[Rieger 75]
Rieger, C.
One System for Two Tasks: A Commonsense Algorithm Memory that Solves Problems and Comprehends Language.
Working Paper 114, MIT AI Lab, November, 1975.

[Rieger 77]
Rieger, C., Grinberg, H.
The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms.
In *Proceedings of the Fifth IJCAI.* 1977.

[Simmons 83]
Simmons, R., Davis, R.
Representing and Reasoning about Change.
AI Memo 702, MIT AI Lab, 1983.

[Stefik 81a]
Stefik, M.
Planning with Constraints (MOLGEN: Part 1).
A. I. Journal, 1981.

[Stefik 81b]
Stefik, M.
Planning and Meta-Planning (MOLGEN: Part 2).
A. I. Journal, 1981.

[Watson 77]
Watson, J.
Molecular Biology of the Gene.
The Benjamin/Cummings Publishing Company, Menlo Park, CA, 1977.

[Weld 81a]
Weld, D.
Computer Aided Organic Synthesis.
ARA-CS- 81-2, Andover Research Associates, 1981.

[Weld 81b]
Weld, D.
The Design of a Reaction Specification Language.
ARA-CS- 81-5, Andover Research Associates, 1981.

[Weld 82]
Weld, D.
Computer Tools for Organism Design.
ARA-CS- 82-10, Andover Research Associates, 1982.

[Wood 81]
Wood, W., Wilson, J., Benbow, R., Hood, L.
Biochemistry, A Problems Approach.
The Benjamin/Cummings Publishing Company, Menlo Park, CA, 1981.