A CONSTRAINED OBJECT APPROACH TO SYSTEMS BIOLOGY

by

Manu Pushpendran

September 2006

A thesis submitted to the
Faculty of the Graduate School of
the State University of New York at Buffalo
in partial fulfillment of the requirements for the
degree of
Master of Science

Department of Computer Science and Engineering

Acknowledgements

I would like to express my gratitude towards the following people for their support and guidance in completing my thesis. First and foremost, sincere thanks to my advisor Dr. Bharat Jayaraman for his time, support and thoughtful guidance without which this endeavor would not have been possible. I would also like to thank my committee member Dr. Aidong Zhang for taking time out from her schedule to review the thesis draft and serve my committee. I would also like to extend my gratitude towards Dr. Daniel Fischer for allowing me to audit his course on Computational Biology which helped me gain significant insights on many biological terms. Special thanks to Dr. Gokul Das from Roswell Park Cancer Institute for some thoughtful discussions on the biological aspects of the thesis. I would also like to acknowledge the valuable discussions I had with the members of the Language Research Group here at University at Buffalo, such as Jeffrey Czyz, Hani Girgis, Weihan Huang and Li Xu. I would also like to thank Dr. Pallavi Tambay for helping out with the implementation aspects of the Cob language. Special thanks to the Graduate Secretary Jodi Reiner for taking care of all the paperwork and to the administrative and support staff of the Computer Science and Engineering department for all their help and co-operation. Last but by no measure the least, I would like to thank my family and friends for their support and encouragement.

Contents

Abstract

Recent advances in bioinformatics combined with genome sequencing and annotation has lead to the generation of large volumes of molecular biological data. This influx of information has necessitated better computational models to interpret the data available and thereby assist in relating the molecular behavior to system characteristics and functions. Such holistic approach to biological modeling is called *systems biology*.

Research has shown that all the information needed to build a detailed model of a cell, including the properties of all constituent components and their interconnectivity, is still not available. However, cells are subject to various constraints such as mass-balance of reactions, thermodynamics, regulation, etc., that help to define its behavioral solution space. Thus, a constraint-based approach to systems biology can overcome the lack of detailed information by successive imposition of constraints on the cell behavior. A purely constraint-based approach however, tends to overlook the structural properties that define the composition of biological systems. Therefore, in this thesis we propose and explore the *constrained object* approach to systems biology. The constrained object paradigm offers a unified approach towards modeling the structural properties of biological systems in terms of an object hierarchy and the behavioral aspects using declarative constraints.

We illustrate our hypothesis by providing a fairly detailed model of a typical metabolic network using the Cob programming language. The time-varying behavior of such networks is modeled using the conceptual extension of Cob called dynamic constrained objects. Additionally, the paradigm allows objective exploration of the phenotypic solution space using preference predicates. Our conclusion from this research shows that constrained objects offer a promising approach to modeling more complex metabolic networks.

Chapter 1

Introduction

1.1 Motivation and Significance

Recent advances in the field of molecular biology combined with rapid technological progress have led to an overwhelming flow of biological data[1,10,16]. This approach has successfully generated information about individual cellular components and their functions. It is estimated that, at the current rate, very soon we would have catalog of individual cellular components and their functions for a large number of organisms[10]. Such knowledge, while necessary in understanding what constitutes the system, is not sufficient in understanding or predicting the system's behavior. Biological systems tend not to abide by the principle of behavioral compositionality, i.e. the behavior of the system is not deducible from the behavioral knowledge of its individual components. Therefore, in order to understand the emergent properties exhibited by biological systems, we need computational models that can simulate the component behavior and their interactions when functioning within a system's environment. Experimental studies to observe systemic behavior tend to be particularly expensive for biological systems[20]. The challenge posed now is to understand how all the cellular components collaborate within living systems.

Systems biology[16,21,35] aims to develop a system-level understanding of biological systems. Such holistic knowledge will enable scientists to link molecular behavior to system characteristics and functions. Computational models for biological systems, thus, will be helpful in analyzing, interpreting and even predicting the genotype - phenotype relationship. The approaches to systems biology can be broadly classified as graph-theoretic, mathematical, object-based, or constraint-based approaches. Graph-theoretic approaches[31] represent the structure of reaction pathways in terms of how 'substances' are connected to each other by reactions. However, they exhibit severe shortcoming in representing reactions other than monomolecular. Purely mathematical models[42] are extremely limited at representing the structural characteristics inherent to all biological systems. Object-based approaches[36] are impeded by the lack of detailed structural information. Constraint-based approaches[22] to cell modeling have the distinct advantage that they can overcome the lack of detailed information by imposing constraints such as mass-balance of reactions, thermodynamics, regulation, etc., that limit the possible cellular behavior. By imposing these constraints on a cell it is possible to predict what the cell can and cannot do. This approach leads to the formulation of solution spaces in which the behavior of a cell is likely to be. The solution space defines the likely phenotypic behavior a cell. Thus, we feel that the constraint-based strategy offers a promising approach to modeling biological systems and processes.

1.2 Constrained Object approach to Systems Biology

A purely constraint-based approach fails to account for the inherent structural characteristics exhibited by biological entities and the variation in their interactions influenced by a dynamic environment. For this reason, we explore a more comprehensive paradigm which caters to both structural and behavioral modeling. The *constrained object paradigm*[37] (Cob) is aimed at modeling systems that are compositional in nature, and whose emergent behavior is governed by certain laws or rules governing the constituent components and their interactions. The

Cob language[38] and its modeling environment, which was developed at the University at Buffalo, has been successfully applied at modeling engineering entities such as electrical networks of interconnected components.

The Cob language supports some of the traditional object oriented features such as inheritance, encapsulation, aggregation etc. as well as declarative specification of system behavior through constraints. The Cob environment also facilitates visual development and manipulation of models. In this thesis we apply the constrained object paradigm towards modeling complex biological entities such as metabolic networks. We show the application of Cob principles in modeling a representative metabolic network. The traditional Cob model, however, is aimed at modeling static system behavior. Therefore, in order to model the dynamic behavior exhibited by biological networks, such as metabolic pathways, we employ the concept of *dynamic constrained object*[23]. The metabolic pathways are often defined by a system of underdetermined biochemical reactions. Therefore, in order to understand specific system behavior we need to employ optimization criteria on the network. Cob facilitates the observation and analysis of specific system behavior by application of *preference predicates*.

The rest of this thesis is organized as follows: Chapter 2 presents the motivation behind building a computational model for biological systems. We review the notion of systems biology aimed at an integrative analysis the data obtained from molecular biology and identify some of the requirements for a computational biological model based on our literature survey. We then elaborate on the motivation for a Constrained Object approach and the advantages it offers over other models. Finally, we present a brief overview of some of the constraints that we will incorporate in our model of a metabolic pathway in Chapter 4.

Chapter 3 details the constrained object paradigm. We briefly review the syntax of the Cob language and illustrate the paradigm of constrained objects through the example of a DC circuit. The basic paradigm of constrained object however cannot easily model dynamic behavior. Therefore, we present the concept of dynamic constrained object and illustrate it in modeling AC Circuits and the nerve cell behavior. In chapter 4 we illustrate the Cob approach to systems biology by building a dynamic Cob model for a hypothetical metabolic network. We also illustrate the ability of Cob in exploring the under-determined behavioral solution space, defined by such reaction pathways, through specification of preference predicates. Finally, we list the advantages in employing the Cob paradigm over traditional modeling methodologies. Chapter 5 presents the conclusions from our study and some open areas for future research in this area.

Chapter 2

Systems Biology

Genome sequencing and annotation combined with high-throughput technologies continues to generate large amounts of molecular information for a wide range of organisms. Such reductionist approaches have focused on analyzing individual cellular components, their composition and functions. However the cellular components and their functions in isolation do not enable us to understand the overall cellular behavior. In this chapter we review the systems biology approach towards understanding biological behavior, and state the requirements for a quantitative computational model to simulate and/or predict systemic behavior exhibited by biological entities.

Molecular biology has traditionally focused on identifying and analyzing individual cellular components, their composition and functions. Such approaches may soon result in a catalog of cellular components for a large number of organisms[10]. Although an understanding of the individual components of the system is important towards understanding the system as a whole, it is not sufficient[15,16]. In order to make sense of the all the molecular data being generated we need to understand the component behavior from a systems perspective. The behavioral properties of biological systems can be better understood by studying the interactions of these components with one another in the context of their operating environment.

To put things in perspective, let's consider the example of a complex engineering entity such as a spaceship. A thorough understanding of all the components it is built of and their functions, although important and essential, would be insufficient in determining the overall functional properties of the spaceship. Similarly, biological systems exhibit emergent properties that cannot be predicted based on the properties of their components alone. Systems biology promises to provide a comprehensive quantitative analysis of the manner in which all the components of a biological system interact functionally over time[1].

2.1 Requirements for a Computational Biological Model

Experimental results indicate that there is no one-to-one relationship between individual cellular components and overall cellular functions [34]. This relationship is extremely complex and highly nonlinear, and thus cannot be predicted from knowledge of the components and their functions alone. Therefore in order to understand and predict cellular behavior, the function of each cellular component must be placed in the context of the interrelatedness and connectivity of cellular components working towards attaining the overall goals of a cellular function [27]. This interrelatedness constitutes a network. Due to the complexity of such networks operating within cells and large number of components involved, it is extremely difficult to understand the behavior of such networks by purely analytical techniques. We therefore need to build representative computational models that can simulate the behavior of such networks in order to understand their complex patterns and relationships and thereby be in a better position to predict their behavior [18].

Our study of literature in this area indicates that quantitative models built to simulate biological behavior should address the following issues:

- *System structure*

  One of the key requirements to understand a biological system is to first identify the structure of the system. The model should be able to depict the component based structure exhibited by cells and cellular components. Towards this end one must identify all the components of the system, their functions and associated parameters. The model should facilitate understanding of physical structure of whole organisms at the cellular level. Such models can then be used to simulate a quantitative analysis of the system's response and its behavioral profile [16].

- *Component attributes*

  Once the system structure has been identified, we need to focus on identifying attributes that define the behavior of each of those components. The model also needs to account for attributes that influence the interaction between these components.

- *Emergent behavior*

  As already stated biological systems don't tend to abide by the principles of behavioral compositionality. In other words the overall behavioral properties exhibited by biological systems tend to be vastly different from the properties exhibited by their individual components. The model should be able to identify each component's behavior in context of the whole organism and its environment. It is therefore imperative that any computational model that simulates biological systems or processes be able to emulate their emergent behavior. It needs to account for how the organism adapts to changes in the environment, and various stimuli, how it maintains robustness against potential damages to the system, how it exhibits the functions observed [16].

- *Dynamic behavior*

  The emergent behavior exhibited by biological systems also tends to change with time and operating conditions. In other words the system behavior at a given instant depends upon the constraints under which it is acting and the cell objective at that instant. For example in glucose rich medium the cell may decide to maximize growth by optimizing biomass generation.

- *Visual modeling environment*

  Since such models are intended to be used primarily by biologists, usability is another important issue that needs to be factored in the design goals. Biologists would prefer to access and modify the underlying model using the visual representation. Therefore such models need to be able to map the visual representation to the underlying code. We need to provide a visual modeling environment that facilitates interactive design and verification. Thereby, it should be possible to fine tune the model using the visual interface as opposed to manipulating the underlying code written to build the model.

- *Incremental design*

  Flexibility in design is another key consideration for any such computational model. This need arises from incomplete knowledge of constraints and erroneous annotation when building such models [15,20,35]. Much of the modeling would thus be hypothesis driven; wherein the model would enable us to make behavioral predictions which will then be tested using *in vivo* /*in vitro* methods. The results would either confirm the hypothesis or lead to reevaluation of the model. The model may also need to be updated as and when new information becomes available from the molecular databases. Thus, the model should be flexible enough to allow iterative development.

- *Biological fidelity*

  Any such computational model needs to be consistent with underlying biochemistry and genetics. Although complete gene portfolios for a large number of organisms are available, functional assignment to these genes is presently incomplete [reference]. Thus, in spite of impressive bioinformatics databases, not all information needed to build a detailed computational model of a whole cell is still available[20]. So, it is likely that such a model may be built on certain assumptions/extrapolation about attributes and/or functions. However such hypothesis needs to be then verified experimentally and any deviation between the observed and expected behavior should be fed back in the model so that it is consistent with observed biological behavior.

- *Scalability*

  According to the Human Genome Project, the human genome contains around 3164.7 million chemical nucleotide bases and is estimated to contain around 20000-25000 genes. The mere size of these numbers is indicative of the kind of scale such a model will need to confront with. Thus, scalability is a critical design parameter for any computational model of a biological system.

2.2 Towards a Constrained Object Approach

As stated in section 2.1, the ability of a computational model of a biological system to predict phenotypic system behavior lies in its ability to model, through a visual modeling environment, the systems structure and its components i.e. their attributes and interactions.

In independent research earlier [Regev A., Shapiro E.] had proposed the idea of modeling cells as computational entities. They proposed the idea of using *abstraction* to model bio-molecular systems so that these complex systems could be described hierarchically. Thus, a system of interacting molecular entities could be described and modeled by a system of interacting computational entities.
Elsewhere [Hartwell L.H., Hopfield J.J. Leibler S., Murray A.W.] have proposed the notion of *modular biology* towards identifying 'functional modules' for representing the biological organization. They define functional modules as discrete entities whose function is separable from those of other modules. The higher-level properties of the cell are then

described by the pattern of connections among their functional modules. [Kitano H. et.al.] proposed modeling the systemic structure and behavior exhibited by biological entities, which would enable us to control the state of the system. They have emphasized on understanding the physical structure of whole organisms at the cellular level as the first step towards understanding biological systems.
Meanwhile, in independent parallel research, investigators at UCSD led by Dr. Palsson have proposed a constraint based approach to cell modeling. Their motivation for constraint based approach stems from the fact that, in spite of huge bioinformatics databases, all the information needed to build a computer model of a whole cell, at a level of detail that contains information on both, the properties of each component in the system as well as their interconnectivity, is still not available. However, this lack of information can be countered by modeling cellular behavior based on the constraints acting on the cell [5,22,24,]. The constraints help to define a solution space in which the phenotypic behavior of the cell is likely to lie, as opposed to a unique solution. This solution space can then be refined further by optimizing on some criteria which would represent a particular phenotypic trait. A purely constraint based approach however fails to account for the structural characteristics, modularity and the contextual behavior exhibited by biological entities.

Based on the literature analysis presented above and the issues faced therein, we propose a unified approach to modeling biological systems that overcomes the limitations of a purely structural modeling approach by incorporating constraints, in the definition of those structural entities and their interactions, in the model. The Constrained Object paradigm developed by researchers lead by Dr. B. Jayaraman at the University at Buffalo, has been shown to model engineering entities like trusses, circuits, etc. with considerable amount of success [37]. We'll review some of the constraints a cell is subject to in the following section; a detailed explanation of the Constrained Object paradigm with relevant examples is presented in the following chapter.

Biological constraints

The constraints acting on a cell can be classified broadly as invariant (hard) or adjustable (soft) constraints. The hard constraints define the boundaries of the solution space and thus represent the range of possible phenotypic behaviors for a cell. Several classifications schemes have been proposed for the type of constraints a cell is subject to [14,22]. Following the work of Dr. Palsson and collaborators we use the following constraint classification scheme:

Physicochemical constraints

They represent hard constraints on the cell. Examples of such constraints include balance of mass and energy. Mass and energy can be never created or destroyed in the cell. Therefore elements entering the cell need to be either incorporated for cell growth and replication or utilized to generate energy needed to carry out cellular functions or secreted into the extra cellular environment. Excess biochemical products tend to accumulate over time and result in cellular toxicity and death [cite reference]. Energy imbalance has similar detrimental consequences. Balance of mass and energy thus imposes critical constraints on the cellular behavior. The total number of components that can be contained in the cell is constrained by the cell volume; another physicochemical constraint. Reaction thermodynamics and enzyme capacity pose additional physicochemical constraints on the cell. The thermodynamics of internal reactions determine the direction in which the reactions proceed. The presence of enzymes facilitates conversion of substrates into products. The maximum enzyme capacity thus influences the possible cellular behavior.

Environmental constraints

The constraints imposed on the cell by its internal and external environment has a significant influence on the cellular behavior. The presence or absence of necessary compounds, physical characteristics of the external environment such as temperature, pressure, pH, exposure to light or water etc. are some of the constraints the external environment can impose on the cell. Inadequate knowledge of these constraints may lead to incorrect or misleading predictions of cell behavior and hence they need to be factored in the quantitative analysis of cellular behavior [5]. The dense internal environment of a cell creates osmotic pressure in relation to the external environment that must be balanced while maintaining an electro neutral environment on both sides of the cell membrane. The osmotic and electroneutrality constraints affect the cell volume which in turn restricts the total number of components that can be contained in the cell [5].

Regulatory constraints

Unlike the above, regulatory constraints are imposed by the cell on themselves in order to cope with the constraints imposed by the internal and external environments. They, thus tend to be time dependent. Through these constraints, cells are able regulate to a certain extent which genes are expressed, which proteins are present and even the activity of proteins in cells. These constraints help to further limit the space of possible cell functions.

For a more detailed analysis of above constraint classification scheme the reader is referred to [20].

Summary

Thus, in this chapter we have looked at the motivation in building computational models to represent biological phenomenon. We detailed the systems biology approach towards understanding the emergent behavior exhibited by biological systems. We also reviewed the requirements for such a computational biological model. We then elaborated on the motivation for a constrained object approach, which offers the distinct advantage of being able to model structural characteristics, and at the same time can overcome the lack of detailed information therein, through declarative specification of constraints on the model's behavioral solution space. Towards this end, we also saw some of the typical constraints acting on a cell. Some of these constraints will be employed when we build the Cob model for a metabolic pathway in Chapter 4.

Chapter 3

Constrained Objects

The object oriented modeling paradigm bas been successfully applied to model many real world complex entities. In this paradigm, objects are essentially containers for data and the behavior is abstracted through an interface of procedures. In the *constrained object* modeling paradigm[37] that we present in this chapter, an object is also a container for data. However, in contrast with traditional objects, a constrained object is one whose attributes are governed by certain laws or invariants. When such objects are aggregated to form complex entities, their internal attributes are often subject to additional interface constraints. Thus, the resultant state of the complex entity can be deduced only by satisfying both the internal and the interface constraints of the constituent objects.

To illustrate the notion of constrained objects, let us consider the example of a resistor in an electrical circuit. Its state maybe represented by three variables V (voltage), I (current) and R (resistance). However, these state variables cannot change independently, instead they are governed by Ohm's law: $V = I * R$. Thus, a resistor is a constrained object. Similarly other electrical components such as capacitors, inductors, voltage sources, etc. can also be viewed as constrained objects as we explain further in the example in the next section. Now, when several such objects meet at a node, the node is subject to Kirchhoff's current law, namely, the sum of currents at the node must be zero. Thus, the node is also a constrained object. Constrained objects, thus, provide a principled approach to compositional modeling of complex systems wherein the behavior of a component by itself and in relation to other components is governed by laws or rules.

In general, modeling such entities involves the specification of both the structure and behavior of their constituent components. While structure can be modeled using *objects* and aggregation/inheritance hierarchies, modeling behavior using traditional imperative procedures places the responsibility of enforcing them on the programmer. Constraints facilitate a declarative specification of the behavior of a complex system. The *Constrained Object* paradigm thus, can be viewed as a declarative approach to object-oriented programming.

3.1 Overview of Cob Language

Cob (for Constrained object) is a programming language[37] that supports some of the traditional object oriented features such as inheritance, encapsulation and aggregation as well as declarative language features such as arithmetic equations and inequalities, quantified and conditional constraints etc. Cob provides a modeling environment that facilitates compositional specification of the structure of a system, declarative specification of its behavior and visual development and manipulation of the underlying model. The following description of Cob syntax has been adapted from [37].

A Cob program is a sequence of class definitions and each constrained object is an instance of some class.

$$program ::= class\_definition^{+}$$

A class definition consists of attributes, constraints, predicates and constructors.

```
class_definition ::= [abstract] class class_id
                     [extends class_id] {body}

            body ::= [attributes attributes]
                     [constraints constraints]
                     [predicates predicates]
                     [constructors constructor_clause]
```

An attribute is a typed identifier, where the type is either a primitive type or user-defined type or an array of primitive or user-defined type.

```
attributes        ::= decl; [decl;]+
decl              ::= type id_list
type              ::= primitive_type_id | class_id |
                        type[]
primitive_type_id ::= real | int | bool | char | string
id_list           ::= attribute_id [, attribute_id]+
```

Constraints define the relation over the attributes of one or more classes.

```
constraints             ::= constraint; [constraint;]+
constraint              ::= simple_constraint |
                          quantified constraint |
                          creational_constraint
creational_constraint   ::= complex_id = new class_id(terms)
quantified_constraint   ::= forall var in enum:(constraints)|
                            exists var in enum:(constraints)
simple_constraint       ::= conditional_constraint |
                            constraint_atom
conditional_constraint::= constraint_atom :- literals
constraint_atom         ::= term relop term |
                            constraint_predicate_id(terms)
relop                   ::=  =|!=|>|<|>=|<=
```

Terms can appear in constraints or as arguments to functions, predicates or constructors.

```
term ::= constant | var | complex_id | (term) |
         arithmetic_expr | func_id (terms) | [terms]
         | sum var in enum : term
         | prod var in enum : term
```

```
            | min var in enum : term
            | max var in enum : term
terms ::= term [,term]+
```

`complex_id`: A complex identifier refers to an element of an array or to an attribute of an object.

```
complex_id ::= attribute_id[.attribute_id]+ |
               complex_id [term]
```

A Literal can be an atom or the negation of an atom:

```
literals ::= literal[,literal]+
literal  ::= [not] atom
atom     ::= predicate_id(terms) | constraint_atom
```

Constructor – A class can have more than one constructors and a class without a constructor must be declared as abstract:

```
constructor_clauses ::= constructor_clause+
constructor_clause ::= constructor_id(formal_pars) {
                           constructor_body }
constructor_body ::= constraints
```

Example (DC Circuit)

Consider the example of an electrical circuit consisting of a series-parallel combination of resistors. The components and connections of such a circuit can be modeled as constrained objects. The `component` class models any electrical entity (e.g resistor, battery) that has two ends. The attributes of this class represent the currents and voltages at the two ends of the entity. The constraint in class `resistor` represents Ohm's law. The class `end` represents the terminal ends of a component. A collection of ends meet at a `node`, where they are subject to Kirchhoff's current law constraint i.e. the sum of currents entering and leaving that node must be zero.

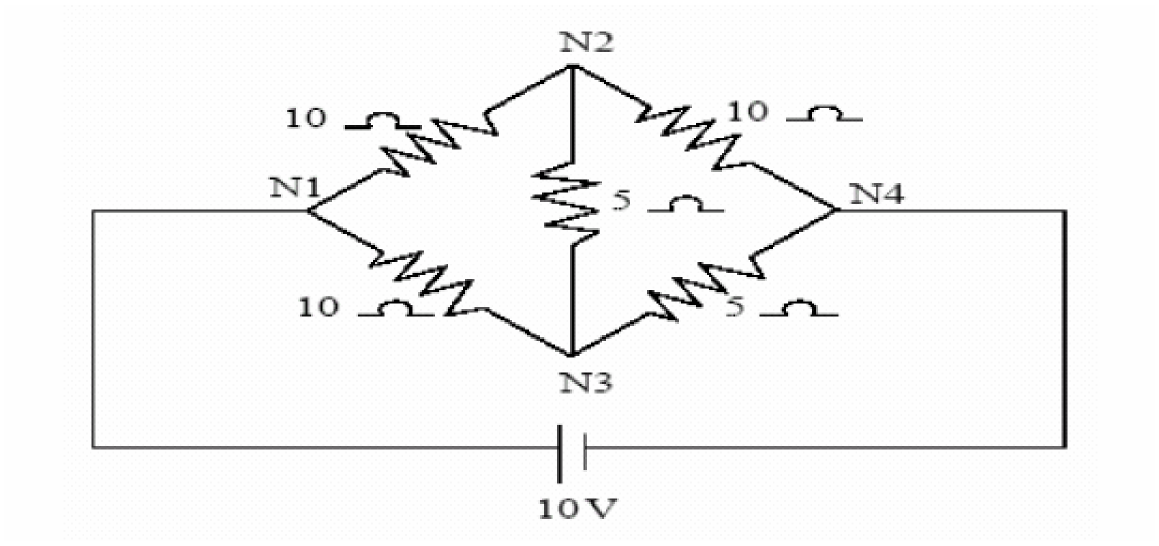


Figure 3.1 An example of DC circuit (adapted from [37])

```
abstract class component {
 attributes
   real V1, V2, I1, I2;
  constraints
     I1 + I2 = 0;
}

class resistor extends component {
 attributes
   real R;
 constraints
   V1 - V2 = I1 * R;
 constructor resistor(D) { R = D; }
}

class battery extends component {
 attributes
  real V;
 constraints
  V2 = 0;
 constructor battery(X) { V1 = X; }
}
```

```
class end {
 attributes
  component C;
  real E,V,I;
 constraints
  V = C.V1 :- E = 1;
  V = C.V2 :- E = 2;
  V = C.I1 :- E = 1;
  V = C.I2 :- E = 2;
constructor end(C1,E1)
{ C = C1; E = E1; }
}

class node {
 attributes
  end [] Ce;
  real V;
 constraints
  sum C in Ce: C.I = 0;
  forall C in Ce: C.V = V;
 constructor node(L) {
  Ce = L; }
}
```

Using the above class definitions we can give a constrained object definition of the circuit class as:

```
class samplecircuit {
 attributes
  resistor R12, R13, R23, R24, R34;
  battery B;
  end Re121, Re122, Re131, Re132, Re231, Re232, Re241,
      Re242, Re341, Re342, Be1, Be2;
  node N1, N2, N3, N4;
 constructor samplecircuit(X) {
  R12 = new resistor(10);
  R13 = new resistor(10);
  R23 = new resistor(5);
  R24 = new resistor(10);
  R34 = new resistor(5);
  Re121 = new end(R12, 1); Re122 = new end(R12,2);
  Re131 = new end(R13,1); Re132 = new end(R13,2);
  Re231 = new end(R23,1); Re232 = new end(R23,2);
  Re241 = new end(R24,1); Re242 = new end(R24,2);
  Re341 = new end(R34,1); Re342 = new end(R34,2);
  B = new battery(10);
  Be1 = new end(B,1); Be2 = new end(B,2);
  N1 = new node([Re121, Be1, Re131]);
  N2 = new node([Re122, Re241, Re231]);
  N3 = new node([Re132, Re232, Re341]);
  N4 = new node([Re242, Re342, Be2]);
 }
}
```

3.2 Dynamic Constrained Objects

Biological entities tend to exhibit dynamic behavior, that is, their behavior tends to change with time and the environmental conditions under which they are functioning. The notion of constrained objects presented in the section above is suitable for modeling systems whose behavior remains essentially static over time. However, for modeling continuously evolving biological entities, we illustrate in this section the notion of dynamic constrained objects[23]. This is followed by the syntax and usage characteristics and some examples in the next section.

The constrained object paradigm and its applications discussed in the section above relied on the steady state behavior of systems. However, certain systems tend to exhibit dynamic behavior, i.e. their state changes with time. For some systems, such state changes can be represented mathematically, whereas on other occasions certain aspects of the time-varying behavior can be characterized by behavioral constraints, while other aspects need to be provided as time-series data[23]. Therefore, in order to represent the dynamic behavior exhibited by such systems we need to maintain information regarding previous states and also be able to enforce constraints that relate a state to those of its previous or succeeding states. To incorporate such functionality in the Cob paradigm the notion of *series variable* was conceived. Series variables can hold a range of values representing different system states. These values can be updated according to certain constraints. The dynamic system behavior can now be modeled by specifying constraints over these series variables.

Syntax and Usage [adapted from [23]]

The D-Cob extends the Cob syntax by introducing two new features: the series variable and dynamic class.

```
program             ::=  class_definition+
class_definition    ::= [abstract] [dynamic] class
                        class_id [extends class_id] { body }
body                ::= [attributes attributes]
                        [constraints constraints]
                        [predicates pred_clauses]
                        [constructors constructor_clause]
attributes          ::= decl ; [decl ;]+
decl                ::= type id_list |
                        series_decl
series_decl         ::= series attribute_id = series_type
series_type         ::= term | [ terms ]
type                ::= [ series ] primitive_type_id |
                        class_id | type[]
primitive_type_id ::= real | int | bool | char | string
id_list           ::= attribute_id [, attributes_id]+
```

The keyword **dynamic** indicates that the class has constraints which specify dynamic behavior. The keyword **series** indicates the variable can store values over different instants of time.

Usage: The ` operator applied on a series variable addresses values in the previous state(s), whereas the ′ operator addresses values in the future state(s).

Example (AC Circuit)

In section 3.1.3 we illustrated the use of Cob in modeling DC circuit with resistors connected in parallel. Now, consider the case for an AC circuit. The primary difference between the two is that in an AC circuit the voltage across the circuit will vary over time and secondly the circuit may have inductors and/or capacitors.

The electrical law that governs the behavior of a capacitor is given as

$$I = C \times dV/dt$$

where C is the capacitance, and dV/dt represents the change in voltage over time. In order to model such behavior in D-Cob the differentiation is approximated by a difference equation which can be represented using the series variable. Thus, the above equation remodeled as a difference equation can be written as,

$$I = C \times \Delta V / \Delta t$$

Now, assuming unit time difference for Δt, current I and voltage V can be represented using series variables. So the D-Cob code for above behavioral constraint becomes,

```
I = C × (V – V′)
```

Similarly, an inductor's electrical constraint given as,

$$V = L \times dI/dt$$

can be transformed into the equivalent D-Cob code,

```
V = L × ( I – I′)
```

Consider the simple AC circuit shown below,

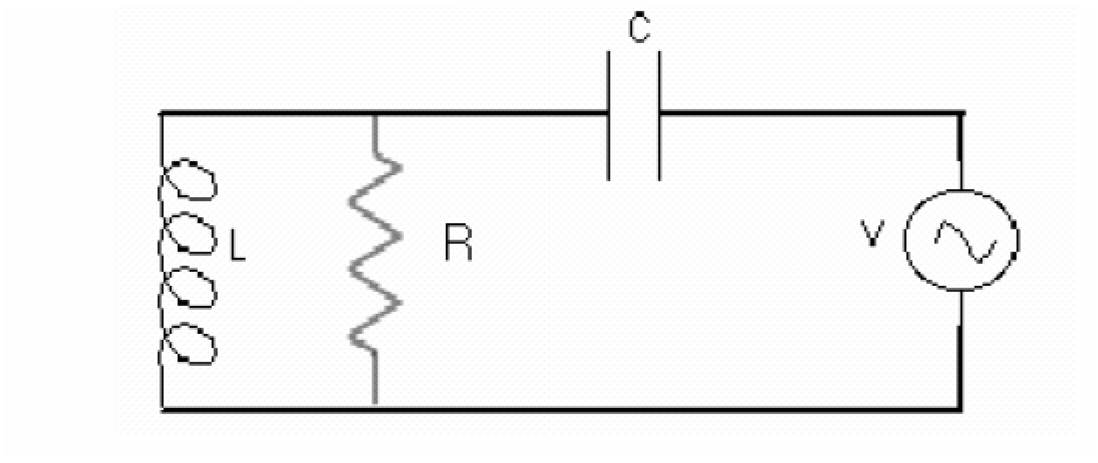


Figure 3.2: Sample AC Circuit

Let's assume the following specifications for the components in the above circuit: Inductor 0.1 henry, Resistor 10 ohm, Capacitor 0.1 farad, AC voltage source 10 * sin(10 * t).

Cob code for the parent class component is given below.

```
abstract dynamic class component {
  attributes
    series real I1, I2, V1, V2;
  constraints
    I1 + I2 = 0;
}
```

Note that the constraint `I1 + I2 = 0` in the above code holds over all progressive values of the series variables I1 and I2.

```
class resistor extends
component {
  attributes
    real R;
  constraints
    V1 - V2 = I1 * R;
  constructor resistor(D) {
    R = D;
  }
}
```

```
class capacitor extends component
{
  attributes
   real C;
  constraints
    I1 = C * (( V1 - V2) -
        ( V1' - V2'));
  constructor capacitor(C1) {
    C = C1;
    V1<1> = 0;
    V2<1> = 0;
  }
}
```

```
class inductor extends
component {
  attributes
    real L;
  constraints
    V1 - V2 = L*(I1 - I1')
  constructor inductor(L1) {
    L = L1;
    I1<1> = 0;
  }
}
```

```
class voltagesource extends
component {
  constraints
    V2 = 0;
  constructor voltagesource(X) {
  V1 = X;
  }
}
```

```
dynamic class componentEnd {
  attributes
   component C;
   series real V, I;
   int End;
  constraints
    V = C.V1 :- End = 1;
    V = C.V2 :- End = 2;
    I = C.I1 :- End = 1;
    I = C.I2 :- End = 2;
  constructor componentEnd(C1,E)
  {
   C = C1;
   End = E;
  }
}
```

```
dynamic class node {
  attributes
    componentEnd [] Ce;
    series real[] V;
  constraints
    sum X in Ce: X.I = 0;
    forall X in Ce: X.V = V;
   constructor node(L) {
     Ce = L;
   }
}
```

```
dynamic class samplecircuit {
  attributes
    resistor R;
    real [] voltages;
    voltagesource B;
    capacitor Cl
    inductor Il
    componentEnd R1,R2,B1,B2,C1,C2,I1,I2;
    node N1,N2,N3;
  constructors samplecircuit() {
    R = new resistor(10);
    C = new capacitor(0.2);
    I = new inductor(0.1);
    Time[1] = 0;
    Voltages = 10 * sin(0.1 * Time);
    B = new voltagesource(Voltages);
    B1 = new componentEnd(B,1);
    B2 = new componentEnd(B,2);
    R1 = new componentEnd(R,1);
    R2 = new componentEnd(R,2);
    C1 = new componentEnd(C,1);
    C2 = new componentEnd(C,2);
    I1 = new componentEnd(I,1);
    I2 = new componentEnd(I,2);
    N1 = new node([C1,B1]);
    N2 = new node([B2,R1,I1]);
    N3 = new node([C2,R2,I2]);
  }
}
```

Example (Nerve Cell Behavior Model)

Under the Hodgkin and Huxley's mathematical model[44] for nerve cell behavior, the total current flow through a cell membrane is the sum total of capacitive and resistive current flows. The capacitive current is defined by the equation:

`I = C * dv/dt` where C and V denote the membrane capacitance and trans-membrane potential. The resistive current is defined as:

$I_{ion} = g_{ion} - (V - E_{ion})$ where V represents the transmembrane potential, E_{ion} the equilibrium potentials of the individual ions and g_{ion} the conductance of ion channels.

There are three different types of ion flow viz. sodium, potassium and a leak current. However, experiments demonstrated that only currents induced by sodium and potassium are time variant. The total resistive current is given as:

$$I_{res} = g_{Na} \times m^3 \times h \times (V - E_{Na}) + g_k \times n^4 \times (V - E_K) + g_L \times (V - E_L)$$

where m and h represent the gates that control sodium flow, while the n gate controls potassium flow.

Each of these gates satisfies the following equation:

$$dX/dt = \alpha_x(v) \times (1 - x) - \beta_x(v) \times x$$

in which x stands for m, h or n and α_x and β_x are coefficients that depend on V and associate with m, h or n respectively.

We now list details of the dynamic cob representation for above model. This representation has been adapted from [43].

The series variable V is used to represent the voltage between the inner and outer side of the cell and I for current. M, H and N represent coefficients for the resistive current.

```
dynamic class HodgkinHuxley {
  attributes
    series real V,M,H,N;
    real I;
  constraints
    V - V’ = I - (120*pow(M,3) * H(V+155) + 36 * pow(N,4) *
             (V -12) + 0.3 * (V+10.6));
    M - M’ = (1 - M)*((V’ + 25)/10) / (exp((V’+25)/10) - 1)
               - M*4*exp(V’/18);
    H - H’ = (1 - H)*0.07*exp(V’/20) -
               H/(1+exp((V’+30)/10));
    N - N’ = (1 - N)*0.1*((V’+10)/10)/(exp(((V’+10)/10)-1)-
              N*0.125*exp(V’/80);
  constructors HodgkinHuxley(A) {
     I = A;
     V<1> = 0; M<1> = 0; H<1> = 0; N<1> = 0;
  }
}
```

3.3 Preference Predicates in Cob

Sometimes the imposition of constraints on a system may lead to solution spaces as opposed to any unique solution. In order to achieve desired objective within this solution space Cob provides the notion of preference predicates. Thus, by specifying an optimization criterion for under-determined systems we can specify the desired behavior from such systems.

For example consider the problem of minimizing the use of raw materials in the combination of mixers and separators in chemical engineering domain. This problem was originally formulated by [Tambay et.al] in her doctoral dissertation[37] and is adapted for presentation in this thesis. The problem can be modeled using the notion of preference predicates in Cob. Consider the scheme of separators and mixers as shown in figure 3.3:

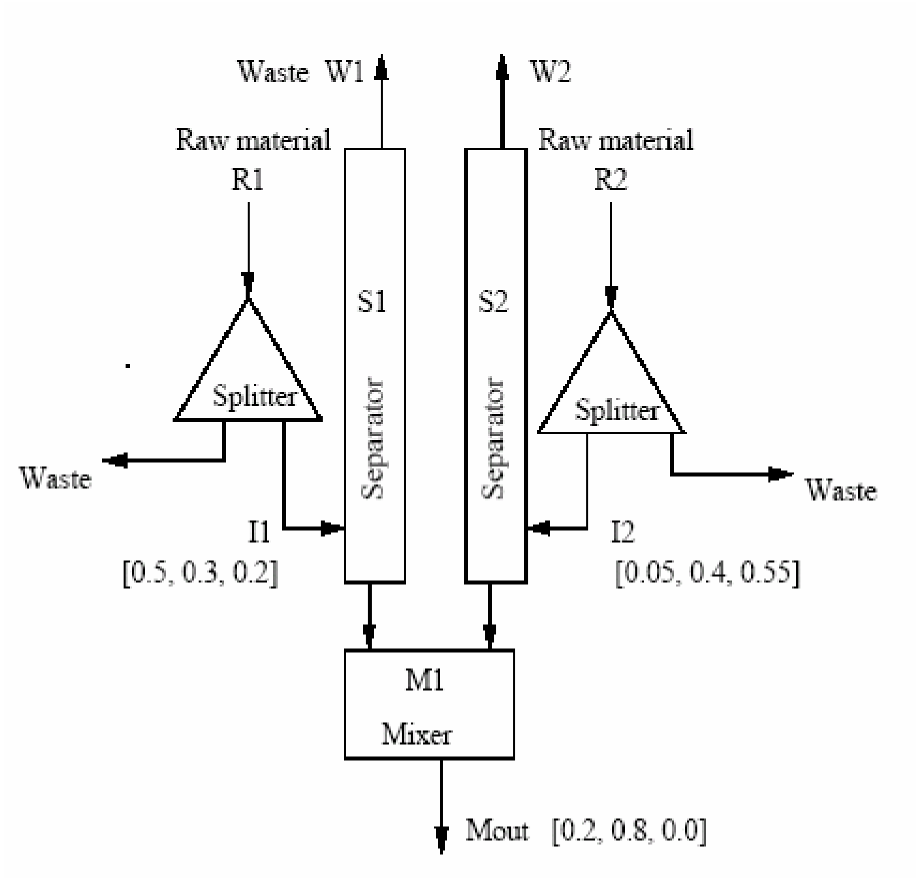


Figure 3.3 Mixers and Separators [redrawn from 37]

The raw materials R1 and R2 are split and a part of each (I1 and I2 respectively) is sent to a separator which separates its ingredients. The mixer combines these ingredients in some proportion to produce the desired chemical Mout. W1 and W2 are waste streams from the separators. The problem then, is to produce Mout while minimizing I1 and I2 thereby minimizing the cost of processing material in the separators.

The key classes identified are `stream` to represent the input raw material stream, `equipment` class models any equipment with some input and output streams and the `Flowsheet` class wherein we specify the preference `min (I1.FlowRate + I2.FlowRate)`. The Cob model can then be used to determine the optimal consumption of input raw material.

```
class stream {
 attributes
  real FlowRate;
  real [] Concentrations;
 constraints
  sum C in Concentrations:C = 1;
 constructors stream(Q, C) {
  FlowRate = Q; Concentrations = C;
 }
}
```

```
class equipment {
 attributes
   stream [] InStream, OutStream;
   int NIngredients;

 constraints % law of mass balance
  forall I in 1..NIngredients :
  (sum J in InStream :(J.FlowRate * J.Concentrations[I])) =
  (sum K in OutStream: (K.FlowRate * K.Concentrations[I]));

 constructors equipment(In,Out,NumIng) {
  InStream = In;
  OutStream = Out;
  NIngredients = NumIng;
 }
}
```

```
class sampleFlowsheet {
 attributes
   stream I1, I2, S1out, S2out, Mout, W1, W2;
   equipment S1, S2, M1;
   real Q1, Q2;
 constraints
   Mout.FlowRate = 150;
   Mout.Concentrations = [0.2,0.8,0.0];
   I1.FlowRate = 500;
   I2.FlowRate = 600;
 preferences
   min (I1.FlowRate + I2.FlowRate).
 constructors sampleFlowsheet() {
   I1 = new stream(Q1, [0.5, 0.3, 0.2]);
   I2 = new stream(Q2, [0.05, 0.4, 0.55]);
   S1 = new equipment([I1], [S1out, W1], 3);
   S2 = new equipment([I2], [S2out, W2], 3);
   M1 = new equipment([S1out, S2out], [Mout], 3);
 }
}
```

Summary

In this chapter we have seen the Constrained Object approach applied at modeling some real world entities from the engineering domain such as circuits. We also illustrated the notion of dynamic constrained objects, their significance and application in modeling dynamic behavior such as in an AC circuit. Finally, we looked at the application of Cob in modeling a biological phenomenon namely, the nerve cell behavior, based on the mathematical model proposed by Hodgkin and Huxley. However, being a purely mathematical model it failed to account for the structural characteristics that are inherent to every biological system. In the next chapter we will explore a specific biological process, its structural and behavioral characteristics, and how it could be modeled using the Cob paradigm. We also discuss the distinct advantages it offers in doing so, over traditional modeling methodologies.

Chapter 4

Modeling Metabolic Networks

The idea of modeling biological systems as constrained objects has been the main theme of this thesis. Earlier, in section 2.3, we cited the distinct advantages offered by the constrained object paradigm over purely constraint based models or purely object models that simulated behavior by enforcing constraints as procedural code. To reiterate, constrained object paradigm allows compositional specification of the structure of any biological system and declarative specification of its behavior through constraints on the objects and their interactions. Besides, the paradigm also facilitates visual development and manipulation of the underlying model where appropriate. In this chapter we illustrate this idea by modeling a representative metabolic network using the Cob paradigm and present the distinct advantages it offers in doing so, over traditional modeling methodologies.

4.1 Metabolic Networks

Metabolism can be considered as a highly integrated network of chemical reactions that converts a particular molecule into some other molecule or molecules in a carefully defined fashion[2]. This process may be accompanied by the consumption or liberation of energy. Metabolism helps us understand how a cell meets it survival objectives. The choice of modeling metabolic network was motivated by the fact that most of the functional annotation completed thus far has been for genes that encode for metabolic functions. Although the number of reactions that constitute a metabolic network even in the simplest of organisms is fairly large, the types of reactions are small and principles for reconstruction of such reactions are well established[2]. Such networks tend to be structurally similar to circuits encountered in electrical engineering domain, although the interactions within these metabolic networks tend to be many orders of magnitudes more complicated. Metabolism facilitates distribution and processing of metabolites throughout its extensive map of pathways. Thus, computational models simulating such networks would help us to understand and thereby be in a position to predict their behavior. It would thus enable us to enhance performance of certain pathways or introduce entirely novel routes for the production of various biochemicals of interest[29]. In section 2.3 we presented one of the classifications schemes for constraints acting on a biological system. We'll consider their relevance as applicable to metabolic networks as we build our Cob model for the same.

Stoichiometric constraints

The reaction equations define the interconnectivity and interactions between the metabolites in the network. They represent the stoichiometric conversion of substrates into products. Some of these reactions are regulated by concentration of enzymes in their environment. These enzymatic reactions as well as the transport of metabolites across system boundaries constitute fluxes which help to dissipate and generate metabolites [29]. A flux balance equation can be written around each metabolite where the difference between the rate of production and consumption of that metabolite is equivalent to the change in concentration of that metabolite over time, in accordance with the law of conservation of mass. Thus, considering the quasi-steady state behavior inside the cell, we can write the following mass balance equation around each metabolite for a system of *m* metabolites involved in *n* reactions:

$$S.v = 0 \qquad (1)$$

This equation represents the stoichiometric constraints on the metabolic network. S is an *m* X *n* matrix wherein the S_{ij}th element represents the number of moles of metabolite *i* participating in reaction *j*. *v* is a vector of unknown metabolic fluxes through the *j* reactions. Eq(1) thus imposes the constraint that total rate of production for any metabolite must equal the total rate of consumption for that metabolite [23]. Excess biochemical products tend to accumulate over time with detrimental consequences for the cell. This mass balance equation is formally analogous to Kirchhoff's current law used in electrical circuit analysis, where the currents entering and leaving a node must sum to zero. Once the genomic sequence for an organism has been annotated the entire metabolic map representing stoichiometry of all metabolic reactions taking place in the cell can be constructed [26]. However, this matrix formulation representing stoichiometry of metabolic reactions provides a purely mathematical perspective of the metabolic pathway. It fails to account for the structural properties inherent to constituent metabolites, the pathway itself and its operating environment. A matrix based representation of the reactions is thus very limited in expressing structural characteristics of the involved entities.

Thermodynamic and enzyme capacity constraints

In addition to stoichiometric constraints, thermodynamics and enzyme capacity constraints are also employed to further limit the possible range of flux values. Thermodynamics associated with reaction equilibrium causes some reactions in the metabolic network to be irreversible. Furthermore, reversible reactions can be decomposed into a forward and reverse component thereby constraining the flux values through these reactions to be positive values [5]. Enzyme capacity constraints place an upper limit on the values a given flux can take. These values can be determined experimentally using procedures detailed in [33].

Regulatory constraints

In order to cope with the hard physicochemical constraints listed above the cell imposes upon itself certain regulatory constraints. As opposed to the rigid physicochemical constraints these regulatory constraints tend to be transitory, that is they are influenced by the state of external and internal environment at any given time. Regulatory constraints are often expressed through enzymes that control the transcriptional activity of genes and thereby are able to control to a certain extent which genes are expressed, which proteins are present and even the activity of proteins in cells. Inclusion of these regulatory constraints has been shown to significantly influence the prediction capabilities of metabolic networks[6,8].

Thus, in the section below we will consider the application of these constraints i.e. stoichiometric, thermodynamic, enzyme capacity and regulatory constraints to a hypothetical metabolic network using the Cob paradigm.

4.2 Example

We will be often using the terms *substrate*, *metabolic product*, *biomass constituents* and *intracellular metabolite* in this and some of the subsequent sections. Given below is a brief definition of these terms as pertinent our example.

Substrates are compounds found in the external medium that can be further metabolized by, or directly incorporated into, the cell. Some examples of substrates are carbon, nitrogen, energy sources etc. essential for cell function.

A *metabolic product* is a compound produced by the cell that is secreted into the extracellular environment. These could be compounds produced in primary metabolism such as carbon dioxide, ethanol, acetate etc.
Biomass constituents are pools of macromolecules that make up biomass. This group includes cellular constituents like macromolecular pools of proteins, lipids, carbohydrates, etc. as well as macromolecular products accumulating inside the cell.
Intracellular metabolite includes all other compounds within the cell. This includes intermediates in different cellular pathways and building blocks used for macromolecular synthesis [33].

We'll illustrate some of the constraints explained above with the help of a simple reaction system as shown in Figure 4.1 below.

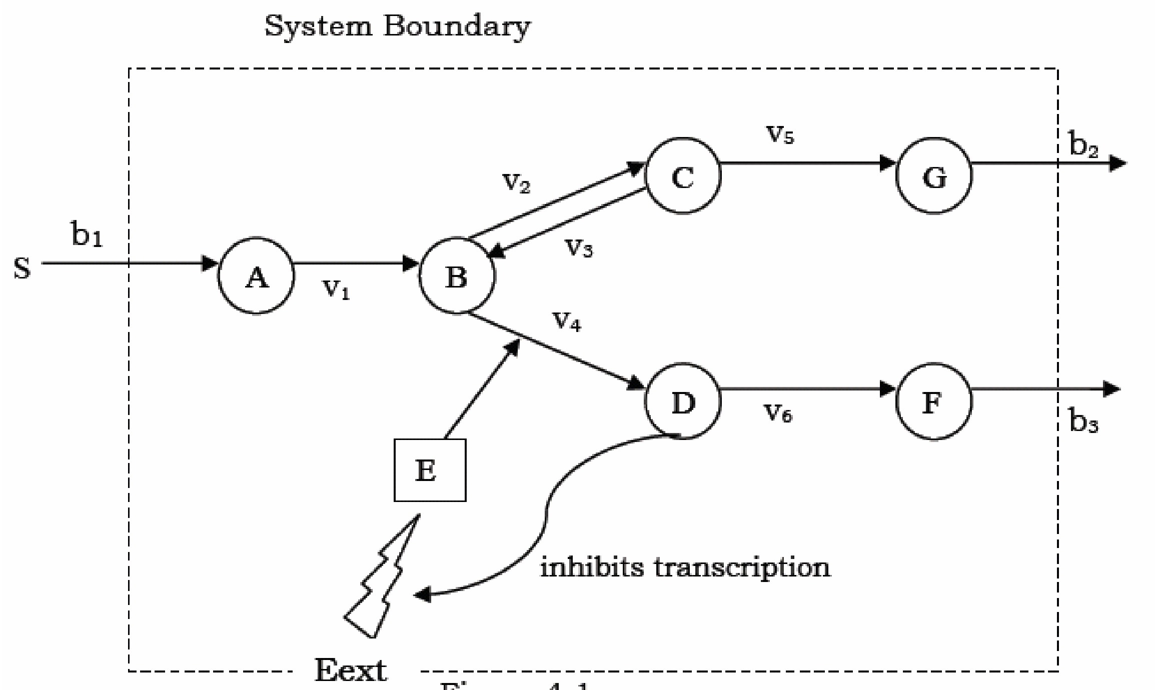


Figure 4.1

The system consists of 6 metabolites namely A, B, C, D, G and F. These metabolites are linked through 5 reactions. Each of these reactions constitutes a flux namely v_1, v_2, v_3, v_4, v_5 and v_6. The reversible reaction between metabolites B and C has been decomposed into two equivalent reactions with positive fluxes. System boundary indicated by dotted lines demarcates the internal environment from the external environment. Metabolite A enters the system with transport flux b_1, metabolites G and F (that can be considered as biomass precursors) exit the system with fluxes b_2 and b_3 respectively. One of the reactions, that is conversion of metabolite B to D is regulated by enzyme E. Thus, this reaction can only occur only if enzyme E is present in the internal environment. The presence of enzyme E in the internal environment is constrained on the presence of substrate Eext in the external environment. However, the product of this reaction D has a negative influence on the transcription of the gene producing E thereby leading to the depletion of E.

Now, let us consider the constraints acting on this system. Mass balance constraints require that the formation fluxes of a metabolite must be balanced by the consumption fluxes for that metabolite. For example, in the network above metabolite A is involved in 2 reactions: the transport flux b_1 that brings A into the internal environment and the reaction flux v_1 that converts it to metabolite B. Thus, imposing flux balance constraint around metabolite A in the network above, we can write $v_1 - b_1 = 0$. Similarly, flux balance equations can be written around every other metabolite in the system. Maximum uptake and secretion rates for transport proteins can be determined experimentally [33]. These can be used to constrain the maximum possible values for the transport fluxes i.e. $b1 \leq b1\_Max$ (similar constraints can be written for b2 and b3). Enforcing thermodynamic constraints we get $v1, v2, v3, v4, v5, v6, b1, b2, b3 \geq 0$. Presence of A in the internal environment is constrained upon the presence of substrate S in the external environment i.e. $b1 > 0$ iff external environment contains (S). The regulated reaction between B and D and the resultant feedback mechanism can be expressed as:

$$\left.\begin{array}{r} -1\,B + 1\,D\ :\ \text{if (E)} \\ E\ :\ \text{if (NOT D)} \end{array}\right\}\ \text{regulation}$$

Abstract Metabolic Network (adapted from [8])

Now, consider an abstract metabolic network as shown in the figure below:

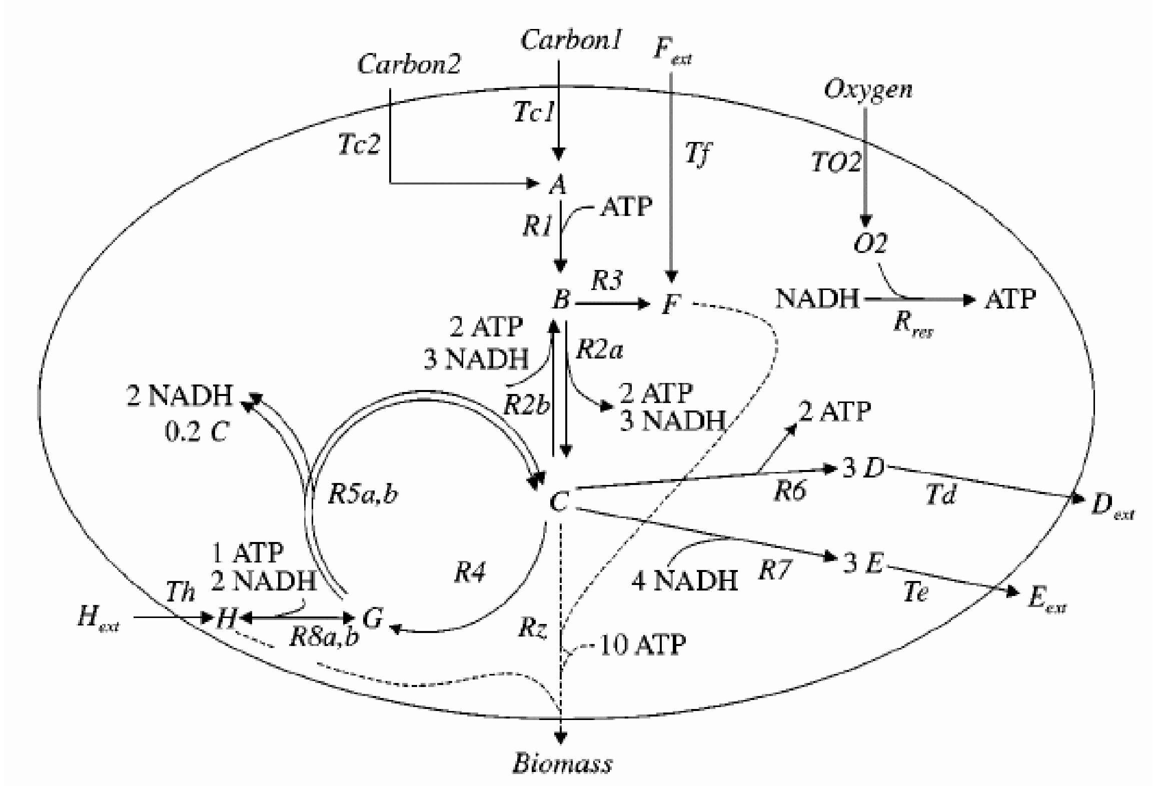


Figure 4.2: Hypothetical metabolic network [redrawn from 8]

The network is an abstract representation of a typical metabolic network. The network consists of 20 reactions, 7 of which are regulated by four regulatory proteins. For modeling convenience we use mnemonic letters such as A, B, C etc as abstractions of actual metabolites. We consider only a few hypothetical reactions and metabolites in this scheme otherwise it would lead to an overwhelmingly large system, without significantly adding to the usefulness of the model to justify the inclusion of additional detail.

The external environment provides carbon sources in the form of *Carbon1* and *Carbon2* through transport processes T_{c1} and T_{c2}. Oxygen, and metabolites F and H enter the network from the external environment through transport processes T_{O2}, T_f and T_h respectively.
ATP is used as the energy currency and NADH serves as the charge carrier. The network is composed of metabolites A, B, C, D, E, F, G, H and O2 linked through 12 reactions. Some of these reactions are regulated by regulatory proteins RPO2, RPc1, RPh and RPb. The reactions and regulatory rules are listed in Table 1.

Reaction	**Name**	**Regulation**
Metabolic reactions		
-1 A – 1 ATP + 1 B	R1	
-1 B + 2 ATP + 2 NADH + 1C	R2a	IF NOT (RPb)
-1 C – 2 ATP -2 NADH + 1 B	R2b	
-1 B + 1 F	R3	
-1 C + 1G	R4	
-1 G + 0.8 C + 2 NADH	R5a	IF NOT (RPO2)
- 1 G + 0.8 C + 2 NADH	R5b	IF RPO2
-1 C + 2 ATP + 3 D	R6	
-1 C – 4 NADH + 3 E	R7	IF NOT (RPb)
-1 G – 1 ATP – 2 NADH + 1 H	R8a	IF NOT (RPh)
1 G + 1 ATP + 2 NADH – 1 H	R8b	
- 1 NADH – 1 O2 + 1 ATP	Rres	IF NOT (RPO2)

Transport processes		
- 1 Carbon1 + 1 A	Tc1	
- 1 Carbon2 + 1 A	Tc2	IF NOT (RPc1)
- 1 Fext + 1 F	Tf	
- 1 D + 1 Dext	Td	
- 1 E + 1 Eext	Te	
- 1 Hext + 1 H	Th	
-1 Oxygen + 1 O2	To2	
Regulatory proteins		
	RPO2	IF NOT (Oxygen)
	RPc1	IF Carbon1
	RPh	IF ($v_{Th} > 0$)
	RPb	IF ($v_{R2b} > 0$)

Table 1 Reactions and regulatory constraints for the simplified metabolic network [adapted from 8]

The metabolic reactions represent stoichiometric constraints on the network. The concentration of carbon sources, oxygen and metabolites in the external environment represents environmental constraints. Column 3 in Table 1 above represents the regulatory constraints imposed by the enzymes RPO2, RPc1, RPh and RPb.

Maintenance and growth processes are approximated by the relation Biomass – 1 C – 1 F – 1 H – 10 ATP. Since the reactions typically form an underdetermined system we will maximize growth using this relation as the objective function when determining the unknown flux values. Maximizing the growth function is in accordance with the normally behavior observed in microbial organisms [40].

4.3 Metabolic Networks as Dynamic Constrained Objects

We will identify important and representative components from the above network and explain their corresponding Cob representation. The idea is illustrated with the help of following class diagram. This will be followed by explanation of Cob code for significant portions of identified classes.

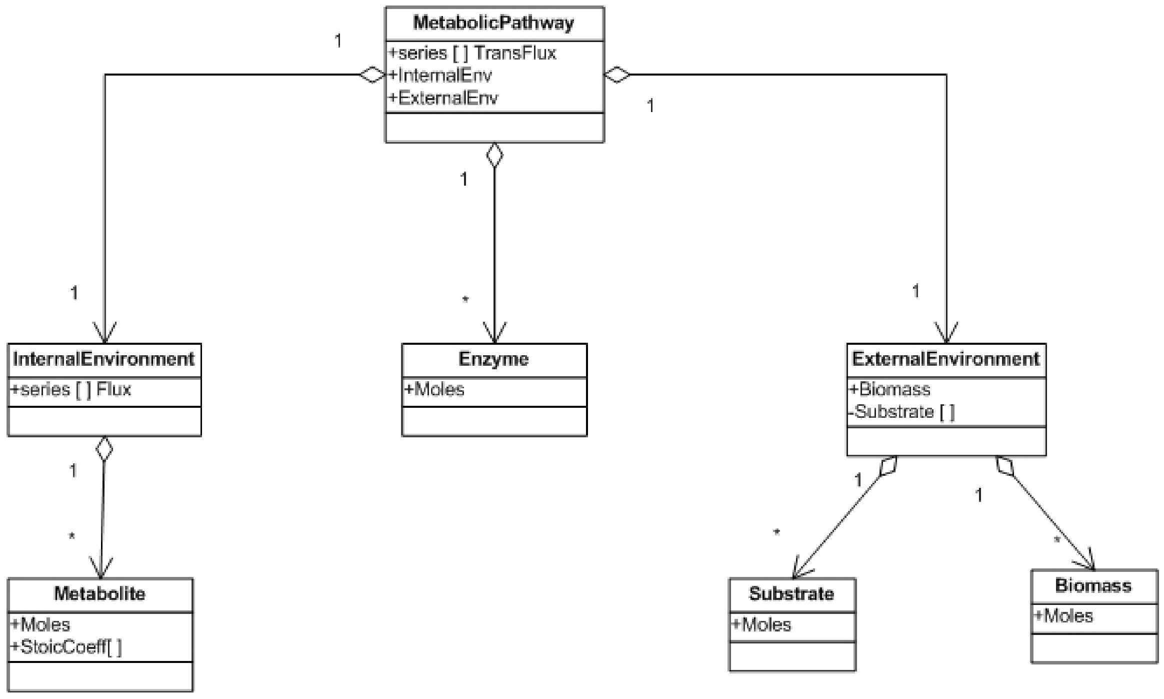


Figure 4.3 Class diagram for Cob representation of the metabolic network in fig 4.2

Let's first consider an unregulated reaction from the network above and identify the important Cob concepts applicable. For the sake of illustration we consider reaction R1.

R1: -1 A – 1 ATP + 1 B

This reaction consumes one molecule of metabolite A and uses up one energy molecule in the form of ATP to form one molecule of metabolite B.

(We'll use the term mole and molecule interchangeably during the course of our explanation). These concentration values may be subject to further change depending upon the other reactions they are involved in. Since this represents a *series* formation for the concentration values of all entities involved, we can represent the molecular concentration of metabolites using a series variable. Thus, metabolites can be modeled aptly as a dynamic Cob class as shown below:

```
dynamic class metabolite {
  attributes
    series real Moles[20];
  constraints
   forall I in 1..20 : Moles[I] >= 0;
  constructor metabolite(Conc,M) {
    ConcPool<1> = Conc;
    forall I in 1..20: Moles[I]<1> = M[I];
  }
}
```

The vector of 20 values represents the number of moles of a metabolite formed or consumed per unit flux of the 12 internal reactions and 8 transport processes. These coefficients form an invariant property of the network and can be obtained from the metabolic genotype of an organism. The initial stoichiometric vector will be initialized when the metabolite is constructed inside the internal environment. For example the metabolite A can be initialized with the Cob syntax:

```
A = new metabolite([0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,
                    0,0]);
```

where the vector element represents the stoichiometric coefficients associated with reactions R1, R2a, R2b, R3, R4, R5a, R5b, R6, R7, R8a, R8b, Rres and transport processes T_{c1}, T_{c2}, T_f, T_d, T_e, T_h and T_{o2} respectively.

The obvious constraints in the reaction R1 above are that the number of moles of metabolite A and energy carrier ATP has to be greater than or equal to 1. Thus, the result of this reaction and its direct execution constraints can be represented using *conditional constraints* in the equivalent Cob syntax:

```
A.Moles[1]` = A.Moles[1] - 1,
ATP.Moles[1]` = ATP.Moles[1] - 1,
B.Moles[1]` = B.Moles[1] + 1
             :- A.Moles[1] >=1, ATP.Moles[1] >=1;
```

The index 1 represents the fact the concentration changes are associated with reaction 1.

However, as we can see from the network the only other reactions producing A are transport processes Tc1 and Tc2. Furthermore, the transport process Tc2 itself is regulated by the presence of enzyme RPc1 in the metabolic pathway. Thus, Tc1 and Tc2 impose indirect constraints on the execution of this reaction. In order to consider the influence of such interactions, we model the internal environment as a dynamic Cob class, with the flux values generated by individual reactions represented as series variables. Given below is the Cob representation of the `internalEnv` class. It includes the metabolites A, B, C, D, E, F, G, H, O2, ATP and NADH represented via the metabolite array `Meta[]`. The reactions involving these metabolites are represented as constraints in the class. This includes only the unregulated reactions in the internal environment, as representing regulatory constraints requires knowledge of the enzymes that are considered as part of the pathway. Furthermore, some these enzymes depend on the presence of compounds in the external environment; hence specification of these constraints is deferred to the `metabolicPathway` class that aggregates both the `internalEnv` and `externalEnv` classes.

Given below is an extract from the `internalEnv` class highlighting the significant components. For the complete code listing please see Appendix A.

```
dynamic class internalEnv {
  attributes
    metabolite [11] Meta;
    series real [12] Flux;
  constraints
    Meta[1].Moles[1]` = Meta[1].Moles[1] - 1,
    Meta[10].Moles[1]` = Meta[10].Moles[1] - 1,
    Meta[2].Moles[1]` = Meta[2].Moles[1] + 1,
          :- Meta[1].Moles[1] >=1, Meta[10].Moles[1] >=1;
    .
    .
    .

   constructors internalEnv() {
    Meta[1] = new metabolite([-1,0,0,0,0,0,0,0,0,0,0,0,1,
                               1,0,0,0,0,0]);
    Meta[2] = new metabolite([1,-1,1,-1,0,0,0,0,0,0,0,0,0,
                              0,0,0,0,0,0]);
    .
    .
    .
  }
}
```

The external environment provides Carbon sources in the form of Carbon1 and Carbon2 through transport processes Tc1 and Tc2 respectively. Oxygen is made available through transport process To2, thus, anaerobic growth can be simulated by restricting the external Oxygen concentration to zero. Some of the intracellular metabolites like H and F can be made by the cell internally or transported from substrates Hext and Fext in the external environment through transport process such as T_h and T_f. Growth is represented by the biomass equation -1C – 1F – 1H – 10ATP + 1Biomass. The objective of the metabolic pathway is to maximize growth by optimizing this reaction.

However, as we can see from the biomass reaction above, it depends on the concentration levels of metabolites C, F, H and ATP that are present in the internal environment. Therefore, it is appropriate to impose the growth objective in the metabolic pathway that has knowledge of both the internal and external environments.

```
class externalEnv {
  attributes
    substrate Carbon1, Carbon2, Oxygen,Fext,Ext,Dext,Hext;
    biomass Bio;
  constraints
    Carbon1.Moles >= 0;
    Carbon2.Moles >= 0;
    Oxygen.Moles >= 0;
    .
    .
    .
  constructors externalEnv(C1,C2,Oxy,Fe,He) {
    Carbon1 = new substrate(C1);
    Carbon2 = new substrate(C2);
    Oxygen = new substrate(Oxy);
    Fext = new metabolite(Fe);
    Eext = new metabolite(0);
    .
    .
  }
}
```

The last significant class we detail here is the metabolic pathway itself. The regulatory and stoichiometric constraints will be enforced by this class. The stoichiometric constraints impose conservation of mass and thereby require that the consumption fluxes for a metabolite be balanced by the corresponding production fluxes for that metabolite. Thus, for a metabolite A with stoichiometric coefficients {-1,0,0,0,0,0,0,0,0,0,0,0, 1,1,0, 0,0,0,0} and the corresponding fluxes through the 12 internal reactions represented by the flux vector `Flux[]` and those through the 8 transport processes represented by the flux vector `TransportFlux[]`, we can write the following formulation in Cob to represent the stoichiometric constraint.

```
(sum J in 1..12:(IntEnv.A.Moles[J] - IntEnv.A.Moles[J]') *
                                          IntEnv.Flux[J]) +
(sum K in 1..8: (IntEnv.A.Moles[K] - IntEnv.A.Moles[K]') *
                                            TransFlux[K])
= 0;
```

Thus, generalizing over all the metabolites in the internal environment, we can write:

```
forall I in 1..11:
(sum J in 1..12:(IntEnv.Meta[I].Moles[J] -
                 IntEnv.Meta[I].Moles[J]') *
                           IntEnv.Flux[J]) +
(sum K in 1..8: (IntEnv.Meta[I].Moles[K] -
                 IntEnv.Meta[I].Moles[K]') *
                             TransFlux[K])
= 0;
```

Regulatory constraints

In order to explain the Cob representation of regulatory constraints lets consider one of the regulated reactions for example reaction R2a:

R2a: -1 B + 2 ATP + 2 NADH + 1 C IF NOT (RPb)

In biological context this indicates transcriptional regulation to maintain concentration of metabolite B. From a computational perspective it indicates that the reaction R2a can take place only if the concentration of enzyme RPb in the pathway is zero. The other execution constraint for this reaction is that the molecular concentration of metabolite B has to be ≥ 1. The side effects are production of 2 molecules of ATP and NADH and 1 molecule of metabolite C. Thus, the Cob notation for the above reaction using conditional constraints is given below:

```
IntEnv.B.Moles[2]` = IntEnv.B.Moles[2] – 1,
IntEnv.C.Moles[2]` = IntEnv.C.Moles[2] + 1,
IntEnv.ATP.Moles[2]` = IntEnv.ATP.Moles[2] + 2,
IntEnv.NADH.Moles[2]` = IntEnv.NADH.Moles[2] + 2
          :- IntEnv.B.Moles[2] >= 1, RPb.Moles == 0;
```

The concentration of enzyme `RPb` is represented by the attribute `Moles`.

Optimization predicates

The system of reaction equations cited in Table1 and their corresponding fluxes often form an underdetermined system [21,27]. That is the number of fluxes often exceeds the number of metabolites. A particular solution can be sought by optimizing on a linear objective function. Here, growth is considered as the *objective* and we determine the corresponding flux distribution that would maximize this objective. Growth is represented by the relation:

$$- 1\ C – 1\ F – 1\ H – 10\ ATP + 1\ Biomass$$

Thus, a biomass molecule is produced by consuming 1 molecule each of metabolite C, F and H and 10 molecules of energy ATP. Cob provides preference clauses to specify the optimization criteria. In presence of preferences the resultant optimal state of the constrained object is obtained by employing constraint satisfaction and optimization techniques. Given below is an extract from the metabolicPathway class where this optimization criterion is specified.

```
dynamic class metabolicPathway extends pathway {
 attributes
   internalEnv IntEnv;
   externalEnv ExtEnv;
   series real [8] TransFlux;
   enzyme RPO2,RPc1,RPh,RPb;
 constraints
   ExtEnv.Bio.Moles` = ExtEnv.Bio.Moles + 1,
  IntEnv.Meta[3].Moles[20]` = IntEnv.Meta[3].Moles[20] - 1,
  IntEnv.Meta[6].Moles[20]` = IntEnv.Meta[6].Moles[20] - 1,
  IntEnv.Meta[8].Moles[20]` = IntEnv.Meta[8].Moles[20] - 1,
  IntEnv.Meta[10].Moles[20]` = IntEnv.Meta[10].Moles[20]-10
                  :- IntEnv.Meta[3].Moles[20] >= 1,
                     IntEnv.Meta[6].Moles[20] >= 1,
                     IntEnv.Meta[8].Moles[20] >= 1,
                     IntEnv.Meta[10].Moles[20] >= 10;
   .
   .
   .

 preferences
   max (ExtEnv.Bio.Moles).
 constructors metabolicPathway(MC1,MC2,MOxy,MFext,MHext,
                              MRPO2,MRPc1,MRPh,MRPb) {
   .
   .
   .
 }
}
```

The constraint highlighted above indicates the restriction that for biomass to be generated the number of moles of metabolite C, F and H in the internal environment have to be greater that 1 and that of ATP has to be more than 10. The results of the reaction are also specified as part of the constraint. The preference clause indicates that the concentration of biomass molecules has to be maximized when more than one optimal solution exists.

Code snippet listing the metabolicPathway class with the regulated reactions, stoichiometric constraints and growth objective is shown below. For the complete code listing please see Appendix A.

```
dynamic class metabolicPathway extends pathway {
 attributes
   internalEnv IntEnv;
   externalEnv ExtEnv;
   series real [7] TransFlux;
   enzyme RPO2,RPc1,RPh,RPb;
```

```
constraints
  RPO2.Moles = 0 :- ExtEnv.Oxygen.Moles > 0;
  RPc1.Moles = 1 :- ExtEnv.Carbon1.Moles > 0;
  RPc1.Moles = 0 :- ExtEnv.Carbon1.Moles = 0;
  RPh.Moles = 1 :- TransFlux[6] > 0;
  .
  .
  .

  forall I in 1..11:
   (sum J in 1..12:(IntEnv.Meta[I].Moles[J] -
                IntEnv.Meta[I].Moles[J]’) *
                          IntEnv.Flux[J]) +
   (sum K in 1..8: (IntEnv.Meta[I].Moles[K] -
                IntEnv.Meta[I].Moles[K]’) *
                            TransFlux[K])
   = 0;


  IntEnv.Meta[2].Moles[2]` = IntEnv.Meta[2].Moles[2] - 1,
  IntEnv.Meta[3].Moles[2]` = IntEnv.Meta[3].Moles[2] + 1,
  IntEnv.Meta[10].Moles[2]` = IntEnv.Meta[10].Moles[2]+ 2,
  IntEnv.Meta[11].Moles[2]` = IntEnv.Meta[11].Moles[2] + 2
  :- IntEnv.Meta[2].Moles[2] >= 1, RPb.Moles == 0;

  .
  . <other regulated reactions>
  .
  .

  ExtEnv.Carbon1.Moles` = ExtEnv.Carbon1.Moles - 1,
  IntEnv.Meta[1].Moles[13]` = IntEnv.Meta[1].Moles[13] + 1
  :- ExtEnv.Carbon1.Moles >= 1;

  .
  . <other transport reactions>
  .
  .

  ExtEnv.Bio.Moles` = ExtEnv.Bio.Moles + 1,
  IntEnv.Meta[3].Moles[20]` = IntEnv.Meta[3].Moles[20]- 1,
  IntEnv.Meta[6].Moles[20]` = IntEnv.Meta[6].Moles[20]- 1,
  IntEnv.Meta[8].Moles[20]` = IntEnv.Meta[8].Moles[20]- 1,
  IntEnv.Meta[10].Moles[20]` =IntEnv.Meta[10].Moles[20]-10
                         :- IntEnv.Meta[3].Moles[20] >= 1,
                            IntEnv.Meta[6].Moles[20] >= 1,
                            IntEnv.Meta[8].Moles[20] >= 1,
                                    IntEnv.Meta[10].Moles[20] >= 10;

 preferences
   max (ExtEnv.Bio.Moles).
 constructors metabolicPathway(MC1,MC2,MOxy,MFext,MHext,
                                   MRPO2,MRPc1,MRPh,MRPb)
{
   RPO2.Moles = MRPO2;
   RPc1.Moles = MRPc1;
   RPh.Moles = MRPh;
   RPb.Moles = MRPb;

   forall I in 1..8: TransFlux[I] = 0;
   IntEnv = new InternalEnv();
   ExtEnv = new externalEnv(MC1,MC2,MOxy,MFext,MHext);
 }
}
```

Applications

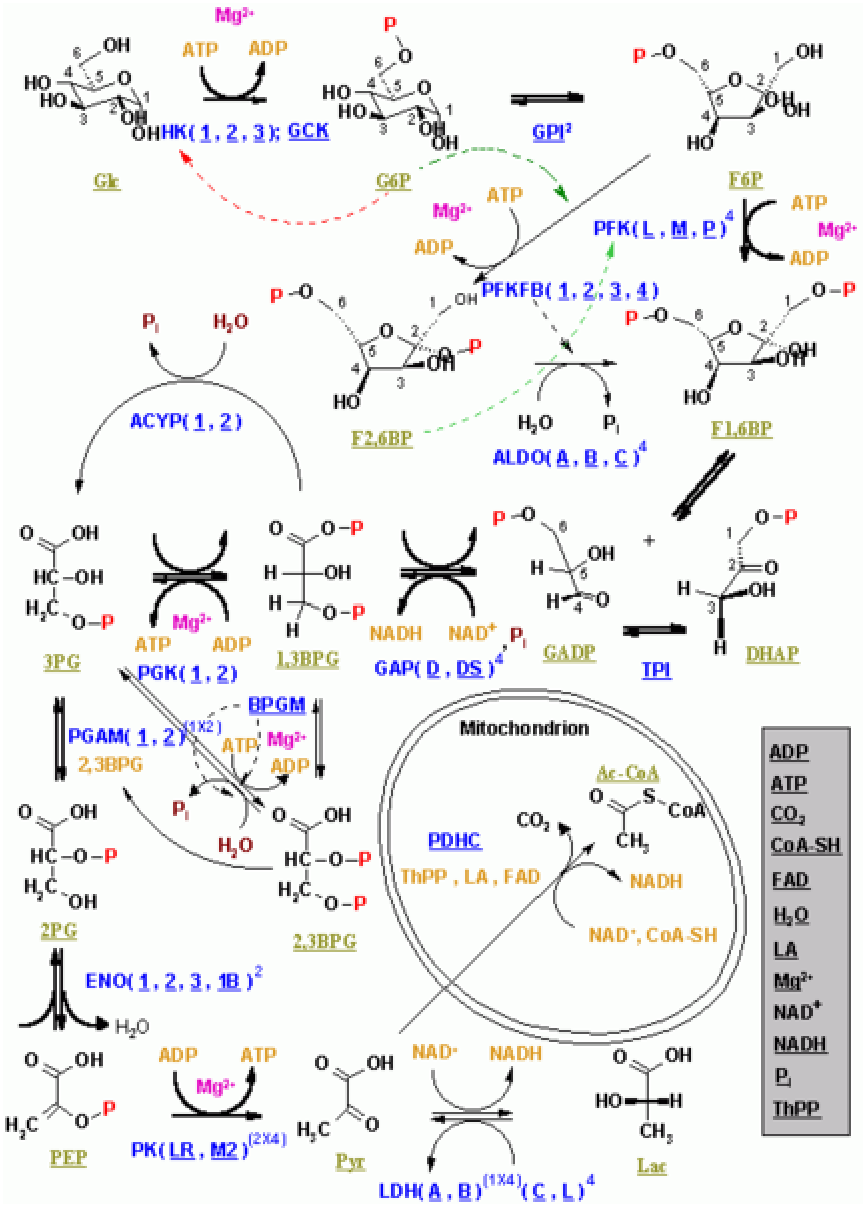
Using the model above, we can determine the unknown fluxes through reactions R1, R2a, R2b, R3, R4,R5a, R5b, R6, R7, R8a, R8b, Rres and the transport processes T_{c1}, T_{c2}, T_f, T_h, T_e, T_d, T_{02}. During each run the reaction rules are executed thereby causing a change in concentrations of the substrates and metabolites. The resultant fluxes through these reactions can be determined by optimizing on some criteria such as minimize ATP production or maximize metabolite production or minimize nutrient uptake or maximize biomass production etc. Depending on the optimization criteria we choose when determining these fluxes, it would enable us to find specific routes through the pathway that can be optimized to achieve desired cell objective. By constraining the concentration of metabolites we can understand how the cell would respond to changes in the environment for example addition or deletion of a substance, the effects of gene deletion and thus the cell behavior in adverse environmental conditions. The flux values will also help us understand the contribution of different components in attaining cell objective for a given criteria. This would enable directed manipulation of gene content of an organism to obtain desired results. Some of these applications bear significant importance in the field of drug discovery.

Advantages

A purely constraint based model provides a reaction based mathematical perspective, and thus fails to capture the structural essence of biological systems. In such models system parameters are invariably considered as isolated variables related through some set of reactions that impose constraints on their interaction. However, in reality these parameters could be attributed to system (sub) components whose behavior can be defined in terms of the constraints acting on these attributes. When larger systems are assembled from these smaller objects, their attributes are further constrained by the interactions they share with other components in these systems. As can be seen from the Cob representation of the metabolic network above, each entity is defined as an independent object with a distinct set of attributes and constraints that captures and defines the essential structural and behavioral features of that entity. More complex structures can be built as an aggregation of these smaller entities with well defined interface and structural signatures. Besides, Cob offers some of the traditional advantages of an object-oriented language such as aggregation/ inheritance hierarchies, encapsulation etc.

Glycolysis Pathway

Consider the glycolysis pathway as shown in the figure below. Glycolysis is the sequence of reactions that metabolizes one molecule of glucose to two molecules of pyruvate accompanied with the net production of two molecules of ATP [2]. Similarities can de drawn with metabolic network in figure 4.2 in terms of the network structure, reaction interconnections, regulatory constraints, etc. The level of detail is comparable to the network we modeled using Cob environment. Thus, we believe the approach we have presented here can be applied towards modeling more complex biological networks.



Glycolysis Pathway (Source Wikipedia http://en.wikipedia.org/wiki/Glycolysis)

Summary

In this chapter we analyzed a hypothetical metabolic network, explored the biological constraints acting on it and detailed a dynamic Cob representation for the same. We then looked at some of the advantages in employing the Cob paradigm over traditional modeling methodologies and discussed some of the applications of the model. In the next chapter we present our conclusions from this study and some open issues for future work in this area.

Chapter 5

Conclusions and Future Work

In this thesis we have proposed and explored the constrained object approach to systems biology. The motivation for this work stems from the large amounts of molecular data being generated and the need to integrate and understand this data from a systems perspective. Biological systems exhibit emergent behavior that cannot be predicted solely from an understanding of the behavior of the individual components of these systems. The network of connectivity and interrelatedness between these components is hard to comprehend using purely analytical techniques. Different approaches have been explored towards modeling biological systems. The need to model structural characteristics was identified by some researchers as the first step in understanding biological entities. However, the lack of information required to build a detailed model of the cell using structural information alone has been an impediment to this approach. Using constraint-based approaches helps to overcome this lack of information by successive identification and imposition of constraints on the behavioral solution space. But a purely constraint-based approach tends to treat system components as independent entities related mathematically through some reactions. This fails to capture the structural characteristics inherent of all biological systems.

The constrained object approach we proposed, in this thesis, offers a unified approach to modeling biological systems, by facilitating a compositional specification of the structure of the system through objects, declarative specification of its behavior through constraints, and visual development and manipulation of the underlying model. Since the traditional Cob model is limited at modeling dynamic behavior exhibited biological networks, we explored the application of dynamic constrained objects in modeling such networks. We also saw that Cob facilitates objective exploration of different behaviors exhibited by these under-determined networks through the application of preference predicates. We illustrated these ideas by modeling an abstract metabolic network using the Cob environment. Towards the end we presented the glycolysis pathway as a detailed extension of the abstract network we modeled using Cob.

Implementation issues

We found the current implementation of the constrained object paradigm limited at integrating the several concepts we have proposed in this thesis. However, we were able to simulate some of the proposed modeling behaviors by employing isolated Cob constructs. For example, we were able to simulate the behavior of underdetermined networks. The results, as expected, were returned as internal SICStus variables. However, the implementation was unable to support subsequent exploration of a specific behavioral trait, using simultaneous application of optimization through preference predicates. On the other hand, the preference predicates, by themselves, were employed and computed by the implementation when functioning within a system of isolated equations. This problem of enforcing optimization in under-determined systems has been a standard topic in linear algebra and has been studied extensively elsewhere [45,46]. We also proposed the concept of enforcing optimization in a dynamic environment. Under such conditions it remains debatable whether future implementations of the system should employ optimization using a local or global perspective. However, we believe the appropriateness of the constrained object framework would encourage a more robust implementation capable of supporting the kind of exhaustive modeling scenarios like the one we have explored in this thesis.

Future Work

The model needs to be tested with metabolic networks of micro-organisms. That would help us understand how well the model scales to incorporate larger systems.

We would also like to link the model to online knowledgebase such as EcoCyc, KEGG, etc., that provide information on the genes, enzymes and pathways employed in the model. These resources provide species specific information on metabolic pathway structures, references to regulatory information etc.

We also need to incorporate feedback mechanism into the model to facilitate incremental development. The model will be used as basis to form hypothesis which should then be tested using *in-vivo* or *in-vitro* methods. Any deviation in the experimental observation and the model prediction should be used to refine the model.

We would also like to build state transition graphs using the model. This would help us understand the action taken by the network at each stage in satisfying the system objective.

We would also like to provide visual interfaces for building complex systems using the Cob environment. This would facilitate observation and interaction with the model through the interface. We also foresee the development of *pluggable* components once sufficient information is available from molecular databases about system components and their behavior.

Finally, the Cob environment sometimes exhibits performance degradation when handling large number of constraints and objects. This is an area of concern that needs to be addressed as we scale up and model more realistic biological systems.

References

1) Aderem A. *Systems Biology: Its practice and challenges* Cell Vol.121 p. 511-513.

2) Berg J.M., Tymoczko J.L., Stryer L. *Biochemistry* Fifth Edition Freeman Press 2002.

3) Bowden A.C., Cardenas M.L. *Systems Biology may work when we learn to understand the parts in terms of the whole* Biochemical Society Transactions Vol.33 p.516-519.

4) Bowden A., Hofmeyr J.S. *The role of stoichiometric analysis in studies of metabolism: An Example* J.theor. Biol. Vol. 216 p 179-191.

5) Covert M.W., Familli I., Palsson B.O. *Identifying Constraints that govern cell behavior: A key to converting conceptual to computational models in biology?* Biotech. and Bioeng. vol 84 p 763-772.

6) Covert M.W., Palsson B.O. *Constraint-based models: Regulation of Gene Expression Reduces the Steady-state solution space* J.theor. Biol. vol. 221 p 309-325.

7) Covert. M.W., Schilling C.H., Familli I., Edwards J.S., Goryanin I.I., Selkov E., Palsson B.O. *Metabolic modeling of microbial strains in silico* Trends in Biotechnology Vol. 26. p179-186.

8) Covert M.W., Schilling C.H., Palsson B.O., *Regulation of Gene Expression in Flux Balance Models of Metabolism* J.theor.Biol. 2001 vol. 213 73-88.

9) Csete M.E., Doyle J.C. *Reverse Engineering of Biological Complexity* Science Vol.295 p. 1664-1669.

10) Edwards J.S., Palsson B.O. *How will Bioinformatics influence metabolic engineering?* Biotech. & Bioeng. vol. 58 p 162-169.

11) Gombert A.K., Nielsen J. *Mathematical modeling of metabolism* Curr. Op. in Biotech. Vol. 11 p 180-186.

12) Hartwell L. H., Hopfield J. J., Leibler S., Murray A. W. *From molecular to modular cell biology* Nature Vol. 402 Supp. p. C47-C52.

13) Heinrich R., Schuster S. *Modeling of metabolic systems: Structure, control and optimality*. BioSystems vol. 47 p 61-77.

14) Jacob F. *Evolution and Tinkering* Science v 196 p.1161-1166.

15) Kitano H. *Computational Systems Biology* Nature Vol. 420 p. 206-210.

16) Kitano H. *Foundations of Systems Biology* MIT Press 2001.

17) Kohl P., Noble D., Winslow R.L., Hunter P.J. *Computational modeling of biological systems: tools and visions* Phil. Trans. R. Soc. Lond. Vol 358 p 579-610.

18) Loew L.M., Schaff J.C. *The Virtual Cell: a software environment for computational cell biology* Trends in Biotechnology Vol.19 p. 401-406.

19) Mahadevan R., Schilling C. H., *The effects of alternate optimal solutions in constraint based genome scale metabolic models* Metabolic Eng. 5 p. 264-276.

20) Palsson B.O., *The challenges of in silico biology* Nature Vol. 18 Nov 2000.

21) Palsson B.O. *Systems Biology: Properties of Reconstructed Networks* Cambridge University Press 2006.

22) Price N.D., Papin J. A., Schilling C.H., Palsson B.O. *Genome scale microbial in silico models: the constraints-based approach* Trends in Biotechnology Vol. 21 p.162-169.

23) Raux R., Jayaraman B. *Modeling Dynamic Systems with Constrained Objects* Technical Report 2004-05 University at Buffalo.

24) Reed. J.L., Palsson B.O. *Thirteen years of building constraint-based in silico models of E. Coli* Journal of Bacteriology p.2692-2699

25) Regev A., Shapiro E. *Cells as computation* Nature Vol.419 p.343.

26) Schaff J., Fink C.C., Slepchenko B., Carson J.H., Loew L.M. *A general computational framework for modeling cellular structure and function* Biophysics Journal Vol.73 p1135-1146.

27) Schilling C.H., Edwards J.S., Palsson B.O. *Toward Metabolic Phenomics: Analysis of Genomic Data Using Flux Balances* Biotechnology 1999 p. 288-295.

28) Schilling C.H., Edwards J.S., Letscher D., Palsson B.O. *Combining Pathway Analysis with Flux Balance Analysis for the Comprehensive Study of Metabolic Systems* Biotechnol. and Bioeng. v 71: p 286-306.

29) Schilling C.H., Letscher D., Palsson B.O. *Theory for the Systemic Definition of Metabolic Pathways and their use in Interpreting Function from a Pathway-Oriented Prespective* J.theor. Biol. Vol. 203 p 229-248.

30) Schilling C.H., Schuster S., Palsson B.O., Heinrich R. *Metabolic Pathway Analysis: Basic Concepts and Scientific Applications in the post-genomic era* Biotechnology prog. 1999 p. 296-303.

31) Seo H., Lee D.Y., Park S., Fan L.T., Shafie S., Bertok B., Friedler F. *Graph-theoretical identification of pathways for biochemical reactions* Biotechnology Letters 23 p. 1551-1557.

32) Stelling J., Klamt S., Bettenbrock K., Schuster S., Gilles E.D. *Metabolic network structure determines key aspects of functionality and regulation* Nature Vol. 420 p190-193.

33) Stephanopoulos G.N., Aristidou A.A., Nielsen J. *Metabolic Engineering: Principles and Methodologies* Academic Press 1998.

34) Strothman R.C. *The Coming Kuhnian Revolution in Biology* Nat. Biotechnology 1997 Vol.15, 194-199.

35) Tomita M. *Whole cell simulation: a grand challenge of the 21st century* Trends in Biotechnology Vol.19 p.205-210.

36) Tomita M., Hashimoto K., Takahashi K., Shimizu T. S., Matsuzaki Y., Miyoshi F., Saito K., Tanida S., Yugi K., Venter J. C., Hutchison C. A. *E-Cell: Software environment for whole cell simulation* Bioinformatics Vol. 15 p. 72-84.

37) Tambay P., *Constrained Objects for modeling Complex Systems* PhD Thesis October 2003, University at Buffalo.

38) Tambay P., Jayaraman B. *The Cob Programmer's Manual* Technical Report 2003-01 University at Buffalo.

39) Varma A., Boesch B.W., Palsson B.O., *Stoichiometric interpretation of E. Coli glucose catabolism under various oxygenation rates*. Appl. And Environ. Microbiol 59:2465-2473

40) Varma A., Palsson B. O. *Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use* Biotechnology Vol. 12 p. 994-998.

41) Varma A., Palsson B.O., *Stiochiometric Flux Balance models quantitatively predict growth and metabolic by-product secretion in wild-type Escherichia coli W3110* Appl. And Environ. Microbiology 1994 60:p3724-3731

42) Varner J., Ramkrishna D. *Mathematical models of metabolic pathways* Curr. Op. in Biotech. v 10 p.146-150.

43) Zhang Y., *Case Studies in Constrained Objects* M.S. Thesis June 2004, University at Buffalo.

44) Hodgkin, A. L. and Huxley, A. F., *A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes* J. Physiol, 117:500-544.

45) Drori I., Donoho D. L., *Solution of Minimization Problems by Ars/Homotopy Methods* Stanford University, Department of Statistics.

46) Garcia C. B. and Zangwill W. I., *Pathways to Solutions, Fixed Points and Equilibria* Englewood Cliffs, NJ: Prentice Hall, 1981.

Appendix A

Cob Model for Metabolic Network shown in Fig. 4.2.

```
dynamic class metabolite {
  attributes
    series real Moles[20];
  constraints
   forall I in 1..20 : Moles[I] >= 0;
  constructor metabolite(Conc,M) {
    ConcPool<1> = Conc;
    forall I in 1..20: Moles[I]<1> = M[I];
  }
}


dynamic class substrate {
 attributes
   series real Moles;
 constraints
   Moles >= 0;
 constructor substrate(M) {
   Moles<1> = M;
 }
}

class biomass {
 attributes
   series real Moles;
 constraints
   Moles >= 0;
 constructor biomass(M) {
   Moles<1> = M;
 }
}

class externalEnv {
  attributes
    substrate Carbon1, Carbon2, Oxygen,Fext,Ext,Dext,Hext;
    biomass Bio;
  constraints
    Carbon1.Moles >= 0;
    Carbon2.Moles >= 0;
    Oxygen.Moles >= 0;
    Fext.Moles >= 0;
    Eext.Moles >= 0;
    Dext.Moles >= 0;
    Hext.Moles >= 0;
    Bio.Moles >= 0;
  constructor externalEnv(C1,C2,Oxy,Fe,He) {
    Carbon1 = new substrate(C1);
    Carbon2 = new substrate(C2);
    Oxygen = new substrate(Oxy);
    Fext = new substrate(Fe);
    Hext = new substrate(He);
    Eext = new substrate(0);
    Dext = new substrate(0);
    Bio = new biomass(0);
  }
}

dynamic class internalEnv {
  attributes
    metabolite [11] Meta;
    series real [12] Flux;
  constraints
    Meta[1].Moles[1]` = Meta[1].Moles[1] - 1,
    Meta[10].Moles[1]` = Meta[10].Moles[1] - 1,
    Meta[2].Moles[1]` = Meta[2].Moles[1] + 1,
               :- Meta[1].Moles[1] >=1,
                  Meta[10].Moles[1] >=1;

    Meta[3].Moles[3]` = Meta[3].Moles[3] - 1,
    Meta[10].Moles[3]` = Meta[10].Moles[3] - 2,
    Meta[11].Moles[3]` = Meta[11].Moles[3] - 2,
    Meta[2].Moles[3]` = Meta[2].Moles[3] + 1
               :- Meta[3].Moles[3] >= 1,
                  Meta[10].Moles[3] >= 2,
                  Meta[11].Moles[3] >= 2;

    Meta[2].Moles[4]` = Meta[2].Moles[4] - 1,
    Meta[6].Moles[4]` = Meta[6].Moles[4] + 1
              :- Meta[2].Moles[4] >= 1;

    Meta[3].Moles[5]` = Meta[3].Moles[5] - 1,
    Meta[7].Moles[5]` = Meta[7].Moles[5] + 1
              :- Meta[3].Moles[5] >= 1;

    Meta[4].Moles[8]` = Meta[4].Moles[8] + 3,
    Meta[10].Moles[8]` = Meta[10].Moles[8] + 2,
    Meta[3].Moles[8]` = Meta[3].Moles[8] - 1
              :- Meta[3].Moles[8] >= 1;
    Meta[7].Moles[11]` = Meta[7].Moles[11] + 1,
    Meta[8].Moles[11]` = Meta[8].Moles[11] - 1,
    Meta[10].Moles[11]` = Meta[10].Moles[11] + 1,
    Meta[11].Moles[11]` = Meta[11].Moles[11] + 2
              :- Meta[8].Moles[11] >= 1;

   constructor internalEnv(X) {
    Meta[1] = new metabolite([0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,
                               0,0,0,0,0]);
    Meta[2] = new metabolite([1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,
                              0,0,0,0,0]);
    Meta[3] = new metabolite([0,1,0,0,0,0.8,0.8,0,1,0,0,0,0,
                              0,0,0,0,0,0,0]);
    Meta[4] = new metabolite([0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,
                              0,0,0,0]);
    Meta[5] = new metabolite([0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,
                              0,0,0,0]);
    Meta[6] = new metabolite([0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,
                              0,0,0,0]);
    Meta[7] = new metabolite([0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,
                              0,0,0,0,0]);
    Meta[8] = new metabolite([0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
                              0,1,0,0]);
    Meta[9] = new metabolite([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                              0,0,0,1,0]);
    Meta[10] = new metabolite([0,2,0,0,0,0,0,2,0,1,1,1,0,0,
                                0,0,0,0,0,0]);
    Meta[11] = new metabolite([0,2,0,0,0,2,2,0,0,0,2,0,0,
                               0,0,0,0,0,0,0]);
   }
 }

 dynamic class metabolicPathway {
  attributes
    internalEnv IntEnv;
    externalEnv ExtEnv;
    series real [8] TransFlux;
    enzyme RPO2,RPc1,RPh,RPb;
  constraints
    RPO2.Moles = 0 :- ExtEnv.Oxygen.Moles > 0;
    RPc1.Moles = 1 :- ExtEnv.Carbon1.Moles > 0;
    RPc1.Moles = 0 :- ExtEnv.Carbon1.Moles = 0;
    RPh.Moles = 1 :- TransFlux[6] > 0;
    RPh.Moles = 0 :- TransFlux[6] <= 0;
    RPb.Moles = 1 :- IntEnv.Flux[3] > 0;
    RPb.Moles = 0 :- IntEnv.Flux[3] <= 0;
    for all I in 1..11
    (sum J in 1..12:
     (IntEnv.Meta[I].Moles[J] - IntEnv.Meta[I].Moles[J]’) *
                               IntEnv.Flux[J]) +
    (sum K in 1..8:
      (IntEnv.Meta[I].Moles[K] - IntEnv.Meta[I].Moles[K]’) *
                                  TransFlux[K])
     = 0;


    IntEnv.Meta[2].Moles[2]` = IntEnv.Meta[2].Moles[2] - 1,
    IntEnv.Meta[3].Moles[2]` = IntEnv.Meta[3].Moles[2] + 1,
    IntEnv.Meta[10].Moles[2]` = IntEnv.Meta[10].Moles[2]+ 2,
    IntEnv.Meta[11].Moles[2]` = IntEnv.Meta[11].Moles[2] + 2
                     :- IntEnv.Meta[2].Moles[2] >= 1,
                        RPb.Moles = 0;

    IntEnv.Meta[7].Moles[6]` = IntEnv.Meta[7].Moles[6] - 1,
    IntEnv.Meta[3].Moles[6]` = IntEnv.Meta[3].Moles[6]+ 0.8,
    IntEnv.Meta[11].Moles[6]` = IntEnv.Meta[11].Moles[6] + 2
                     :- IntEnv.Meta[7].Moles[6] >= 1,
                        RPO2.Moles = 0;

    IntEnv.Meta[7].Moles[7]` = IntEnv.Meta[7].Moles[7] - 1,
    IntEnv.Meta[3].Moles[7]` = IntEnv.Meta[3].Moles[7]+ 0.8,
    IntEnv.Meta[11].Moles[7]` = IntEnv.Meta[11].Moles[7] + 2
                   :- IntEnv.Meta[7].Moles[7] >= 1,
                      RPO2.Moles > 0;


   IntEnv.Meta[3].Moles[9]` = IntEnv.Meta[3].Moles[9] - 1,
   IntEnv.Meta[11].Moles[9]` = IntEnv.Meta[11].Moles[9] - 4,
   IntEnv.Meta[5].Moles[9]` = IntEnv.Meta[5].Moles[9] + 3
                     :- IntEnv.Meta[3].Moles[9] >= 1,
                        IntEnv.Meta[11].Moles[9] >= 4,
                        RPb.Moles = 0;

   IntEnv.Meta[7].Moles[10]`= IntEnv.Meta[7].Moles[10] - 1,
   IntEnv.Meta[10].Moles[10]`= IntEnv.Meta[10].Moles[10]- 1,
   IntEnv.Meta[11].Moles[10]`= IntEnv.Meta[11].Moles[10]- 2,
   IntEnv.Meta[8].Moles[10]`= IntEnv.Meta[8].Moles[10] + 1
                  :- IntEnv.Meta[7].Moles[10] >= 1,
                     IntEnv.Meta[10].Moles[10] >= 1,
                     IntEnv.Meta[11].Moles[10] >= 2,
                     RPh.Moles = 0;
    IntEnv.Meta[11].Moles[12]`= IntEnv.Meta[11].Moles[12]- 1,
    IntEnv.Meta[9].Moles[12]` = IntEnv.Meta[9].Moles[12] - 1,
    IntEnv.Meta[10].Moles[12]` = IntEnv.Meta[10].Moles[12]+ 1
                  :- IntEnv.Meta[9].Moles[12] >= 1,
                     IntEnv.Meta[11].Moles[12] >= 1,
                     RPO2.Moles = 0;

     ExtEnv.Carbon1.Moles` = ExtEnv.Carbon1.Moles - 1,
     IntEnv.Meta[1].Moles[13]` = IntEnv.Meta[1].Moles[13] + 1
                  :- ExtEnv.Carbon1.Moles >= 1;

     ExtEnv.Carbon2.Moles` = ExtEnv.Carbon2.Moles - 1,
     IntEnv.Meta[1].Moles[14]` = IntEnv.Meta[1].Moles[14] + 1
                     :- ExtEnv.Carbon2.Moles >= 1,
                        RPc1.Moles = 0;

     ExtEnv.Fext.Moles` = ExtEnv.Fext.Moles - 1,
     IntEnv.Meta[6].Moles[15]` = IntEnv.Meta[6].Moles[15] + 1
                     :- ExtEnv.Fext.Moles >= 1;

     ExtEnv.Dext.Moles` = ExtEnv.Dext.Moles + 1,
     IntEnv.Meta[4].Moles[16]` = IntEnv.Meta[4].Moles[16] - 1
                     :- IntEnv.Meta[4].Moles[16] >= 1;

     ExtEnv.Eext.Moles` = ExtEnv.Eext.Moles + 1,
     IntEnv.Meta[5].Moles[17]` = IntEnv.Meta[5].Moles[17] - 1
                     :- IntEnv.Meta[5].Moles[17] >= 1;

     ExtEnv.Hext.Moles` = ExtEnv.Hext.Moles - 1,
     IntEnv.Meta[8].Moles[18]` = IntEnv.Meta[8].Moles[18] + 1
                     :- IntEnv.Meta[8].Moles[18] >= 1;

     ExtEnv.Oxygen.Moles` = ExtEnv.Oxygen.Moles - 1,
     IntEnv.Meta[9].Moles[19]` = IntEnv.Meta[9].Moles[19] + 1
                     :- ExtEnv.Oxygen.Moles >= 1;

     ExtEnv.Bio.Moles` = ExtEnv.Bio.Moles + 1,
     IntEnv.Meta[3].Moles[20]` = IntEnv.Meta[3].Moles[20]- 1,
     IntEnv.Meta[6].Moles[20]` = IntEnv.Meta[6].Moles[20]- 1,
     IntEnv.Meta[8].Moles[20]` = IntEnv.Meta[8].Moles[20]- 1,
     IntEnv.Meta[10].Moles[20]`=IntEnv.Meta[10].Moles[20]- 10
                     :- IntEnv.Meta[3].Moles[20] >= 1,
                        IntEnv.Meta[6].Moles[20] >= 1,
                        IntEnv.Meta[8].Moles[20] >= 1,
                        IntEnv.Meta[10].Moles[20] >= 10;

   preferences
```

```
   maximize (ExtEnv.Bio.Moles);

constructor metabolicPathway(MC1,MC2,MOxy,MFext, MHext,
                             MRPO2,MRPc1,MRPh,MRPb)
 {
   RPO2 = new enzyme(MRPO2);
   RPc1 = new enzyme(MRPc1);
   RPh  = new enzyme(MRPh);
   RPb = new enzyme(MRPb);

   forall I in 1..8: TransFlux[I] = 0;
   IntEnv = new InternalEnv();
   ExtEnv = new externalEnv(MC1,MC2,MOxy,MFext,MHext);
 }
}
```