BioSim - A New Qualitative Simulation Environment for Molecular Biology

Karsten R. Heidtke & Steffen Schulze-Kremer
Max-Planck-Institute for Molecular Genetics
Department Lehrach, Ihnestraße 73
D-14195 Berlin, Germany
steffen@chemie.fu-berlin.de
http://igd.rz-berlin.mpg.de/~steffen

Abstract

Traditionally, biochemical systems are modelled using kinetics and differential equations in a quantitative simulator. However, for many biological processes detailed quantitative information is not available, only qualitative or fuzzy statements about the nature of interactions. In a previous paper we have shown the applicability of qualitative reasoning methods for molecular biological regulatory processes. Now, we present a newly developed simulation environment, BioSim, that is written in Prolog using constraint logic programming techniques. The simulator combines the basic ideas of two main approaches to qualitative reasoning and integrates the contents of a molecular biology knowledge base, EcoCyc. We show that qualitative reasoning can be combined with automatic transformation of contents of genomic databases into simulation models to give an interactive modelling system that reasons about the relations and interactions of biological entities. This is demonstrated on the glycolytic pathway.

Introduction

Nearly all of the many existing simulation packages require users to specify their models in precise, numerical terms. For biologists this is a handicap since there is a large amount of mainly qualitative information emerging from genome mapping and functional genomic experiments that can not be included into numerical kinetic simulations. Even in those cases where e.g. concentrations or enzymatic activities are known those numbers come with considerable variance.

Qualitative reasoning has been invented (Forbus 1984; Kuipers 1986) to handle qualitative information for simulation tasks and since then been shown to be a powerful method in the domains of medicine (Kuipers & Kassirer 1987) and qualitative physics (Weld & de Kleer 1990). Coarsely, the work done in qualitative reasoning about biological systems can be grouped into modelling and reasoning on the one hand (Kazic 1993a; 1993b; Karp & Mavrovouniotis 1994) and modelling and simulation on the other (McAdams & Shapiro 1995; Meyers & Friedland 1984; Schulze-Kremer 1990; 1995). Related work on qualitative reasoning in medicine has been done by (Artoli *et al.* 1996). In our previous work (Heidtke & Schulze-Kremer 1997) we have shown how to apply qualitative reasoning methods to molecular biological regulatory processes. Here, we present a novel approach that integrates qualitative reasoning and a molecular biology knowledge base into a qualitative simulation environment.

Our new method facilitates automatic translation of rules and facts of a knowledge base into a simulation model and its subsequent execution. Qualitative reasoning is a class of constraint satisfaction problems. We use *Constraint Logic Programming*, CLP, to combine the efficiency of constraint solving with the declarativity of logic programming (Frühwirth *et al.* 1992). The simulator provides an object and process oriented view on the simulation model and a knowledge base, which in the current version consists of EcoCyc (Karp *et al.* 1997) translated into Prolog. As an application of our simulation environment, we qualitatively simulate the glycolytic pathway where we explicitly represent all substrates and enzymes involved. The basic simulation model was automatically built from the knowledge base and manually refined to focus on a selected set of behaviours.

System and Methods

Several tools have been developed for qualitative reasoning purposes (Dague 1995): The *Qualitative Simulator*, QSim 4.0 Alpha, based on *qualitative differential equations*, QDEs (Kuipers 1994) and the *Qualitative Process Engine*, QPE (Forbus 1990), based on *qualitative process theory*, QPT (Forbus 1984). QPE is a tool for qualitative reasoning on objects using domain and scenario descriptions. The interaction between objects is defined by processes. Another tool is QPC (Crawford,

Abbreviations: CLP - Constraint Logic Programming; EcoCyc - *E. coli* genes and metabolism knowledge base; BioSim - a qualitative process simulator connected to EcoCyc; KEE - Knowledge Engineering Environment; QDE - qualitative differential equation; QPC - a compiler from physical models to qualitative differential equations; QPE - qualitative process engine; QSim - qualitative simulator.

add	+	0	-
+	+	+	?
0	+	0	-
-	?	-	-

mult	+	0	-
+	+	0	-
0	0	0	0
-	-	0	+

Table 1: Qualitative arithmetic function constraints. Addition and multiplication are represented by relations. The qualitative value of a variable can be positive (+), negative (−) or 0, respectively. The question-mark indicates that a variable can take its value out of $\{+, 0, -\}$. Similar tables are used to resolve qualitative derivatives.

Farquhar, & Kuipers 1990), which compiles QPE's domain and scenario models into QSim's qualitative differential equations. QSim and QPE are implemented in Common LISP and so are the models built for QSim and QPE.

Our new simulation environment presented here is implemented in Eclipse 3.5.2 Prolog (Aggoun *et al.* 1995), for several reasons. First, Prolog is a declarative language with built-in symbolic and non-numeric inference capability, which simplifies the implementation of a qualitative model and a simulation engine. Second, Eclipse Prolog provides a library for constraint handling rules and a library to generate suspensions and a relational knowledge base (Brisset *et al.* 1995).

Qualitative Process Simulator

The approach presented in this paper combines the basic ideas of QPT and QSim. Similar to QPT, a model is described in terms of objects and processes.

Definition 1 *Given an object name o as unique identifier, a print name n for graphical output, a set of parameters OP characterising the object and a set of constraints OC applying to the set of parameters, an object is defined as*

$$object = \{o, n, OP, OC\}.$$

Similar to QSim, object parameters are functions over time. The value of a parameter is taken out of its quantity space which is a totally ordered set of landmarks. A value includes a magnitude and a direction of change, written as a pair `(qmag,qdir)`. The magnitude (`qmag`) is a landmark or an interval in the range of two neighboured landmarks. The direction (`qdir`) indicates whether the magnitude is increasing (`inc`, ↑), decreasing (`dec`, ↓) or steady (`std`, ○). The basic set of landmarks is $\{-\infty, 0, \infty\}$. Additional landmarks can be added when the quantity space of a parameter is defined.

Definition 2 *Given a parameter name pn, a quantity space QS defining the respective domain and a set of initial values IV, a parameter is defined as*

$$parameter = \{pn, QS, IV\}.$$

References in constraints of an object are restricted to its own parameters.

Definition 3 *Given a set of object parameters $\{x, y, z\}$ to be functions over time and M a class of monotonically increasing functions, the qualitative constraints are then defined as follows:*

`minus(x,y)`	$\equiv$	$x(t) = -y(t)$
`d/dt(x,y)`	$\equiv$	$\frac{d}{dt}x(t) = y(t)$
`constant(x)`	$\equiv$	$\frac{d}{dt}x(t) = 0$
`increasing(x)`	$\equiv$	$\frac{d}{dt}x(t) > 0$
`decreasing(x)`	$\equiv$	$\frac{d}{dt}x(t) < 0$
`m+(x,y)`	$\equiv$	$y(t) = f(x(t)), f \in M$
`m-(x,y)`	$\equiv$	$y(t) = -f(x(t)), f \in M$
`add(x,y,z)`	$\equiv$	$x(t) + y(t) = z(t)$
`mult(x,y,z)`	$\equiv$	$x(t) \times y(t) = z(t)$

We implemented a basic set of arithmetic, differential and analytic function constraints. These constraints are qualitative abstractions of their quantitative counterparts: addition, multiplication, derivative, monotonically increasing and monotonically decreasing. Constraints may include corresponding pairs (or triplets) of argument-result values to specify the relationship between variables.

Processes describe functional dependencies between object parameters.

Definition 4 *Given a process name p as unique identifier, a print name n for graphical output, a set of preconditions PP defining when a process is active and a set of constraints PC applying to the parameters of objects, a process is defined as*

$$process = \{p, n, PP, PC\}.$$

The set of preconditions is a set that states whether or not a process holds, i.e. it is active or inactive. A precondition is a relation between a parameter and a qualitative magnitude.

Definition 5 *Given two expressions E1, E2, a precondition is defined as:*

$$E1\ rel\ E2, rel \in \{<, \leq, >, \geq, =, \neq\},$$

where E1 is a parameter of an object and E2 is a qualitative magnitude.

If a process is active its constraints are applied to the parameters of the constrained objects. To refer to a parameter of an object in a process definition the parameter is written as a argument of its object, e.g. object `GLC-6-P` has `concentration` as parameter then `GLC-6-P(concentration)` can be used in a set of preconditions or constraints. An excerpt of our simulation model is shown in Figure 1.

QPT and QSim

In QPT and QSim magnitudes of quantities are described in terms of quantity spaces and landmark values. In contrast to QPT, QSim allows automatic extension of the quantity space when the algorithm detects

Model: glucose-6-phosphate ⇌ fructose-6-phosphate

```
object( 'GLC-6-P', 'glucose-6-phosphate',
   [ [concentration, [0,inf], [((0,inf), std)]],
     [degraded,      [-inf,0], [((-inf,0),std)]],
     [produced,       [0,inf], [((0,inf), std)]] ],
   [ add(produced, degraded, concentration) ] ).

object( 'FRUCTOSE-6P', 'fructose-6-phosphate',
   [ [concentration, [0,inf], [(0,std)]],
     [degraded,      [-inf,0], [(0,std)]],
     [produced,       [0,inf], [(0,std)]] ],
   [ add(produced, degraded, concentration) ] ).

object( 'PGLUCISOM', 'phosphoglucose isomerase',
   [ [activity,       [0,inf], [(0,std)]],
     [concentration, [0,inf], [((0,inf), std)]] ],
   [] ).


process('PGLUCISOM-RXN','hexosephosphate isomerase',
   [ ['GLC-6-P'(concentration) gt 0] ],
   [ decreasing 'GLC-6-P'(degraded),
     increasing 'FRUCTOSE-6P'(produced),
     'm-'('GLC-6-P'(degraded),
          'FRUCTOSE-6P'(produced),
          [(0,0),(-inf,inf)]) ] ).

process('PGLUCISOM-ENZRXN','phosphogl. isomerase',
   [ ['PGLUCISOM'(concentration) gt 0,
      'GLC-6-P'(concentration)   gt 0],
     ['PGLUCISOM'(concentration) gt 0,
      'PGLUCISOM'(activity)      gt 0,
      'FRUCTOSE-6P'(produced)    gt 0] ],
     [ 'd/dt'('FRUCTOSE-6P'(produced),
              'PGLUCISOM'(activity)) ] ).
```

Figure 1: This is an excerpt of the glycolysis model that has been generated automatically. It can be used to run a simulation without any changes. Substrates GLC-6-P, FRUCTOSE-6P and enzyme PGLUCISOM are represented as objects. Reactions PGLUCISOM-RXN and PGLUCISOM-ENZRXN are represented as processes. The object description contains a set of parameters produced, concentration and degraded with a basic quantity space [0,0] and [-inf,inf], respectively. The initial concentration of GLC-6-P and PGLUCISOM is ((0,inf),std), and of FRUCTOSE-6P is (0,std). If GLC-6-P(concentration) gt 0 holds, PGLUCISOM-RXN is active. The behaviour is such that GLC-6-P(concentration) is increasing while FRUCTOSE-6P(concentration) is decreasing. As long as enzyme PGLUCISOM is active FRUCTOSE-6P(produced) is increasing.

critical values of a variable. BioSim currently provides the possibility to add new landmark values only when the quantity space is defined, but for a future version we work on including automatic landmark generation.

The principle difference between QPT and QSIM is that for QPT the user has to positively describe objects and processes in his model whereas in QSIM the key item is a constraint that restricts the parameter space of otherwise unspecified objects and processes. Similar to objects in QPT, a component-connection model has been introduced in (Kuipers 1994). In contrast to our method a component-connection model does not allow for explicit specifcation of processes and it still has to be compiled into a QSim QDE before running a simulation.

In QPT functional relationships are described by direct or indirect influences. The use of influences is critical (Kuipers 1994) because we can not conclude from the fact that a variable x is increasing that an influenced variable y is also increasing since there may be other influences which dominate y. In QSim and BioSim, monotonic function constraints are used to imply that variable x must be increasing while y is increasing.

In QSim available quantitative information can be used to perform a semi-quantitative or numeric simulation. This helps restricting the possible behaviour of a simulation model. Nevertheless, in QSim a simulation is restricted to the manually implemented model, whereas in BioSim the integration with EcoCyc (Karp *et al.* 1997) shows how to automatically create qualitative equations for simulation from a knowledge base of genetics and metabolism. The integrated knowledge base serves as a library to simplify the building of simulation models.

Constraint Logic Programming

CLP is an attempt to overcome the performance problems of the *generate and test* procedure with large search applications in logic programming languages. The work of (Teleki 1996) presents a basis for qualitative reasoning using constraint logic programming techniques. CLP in Prolog is done by applying constraint handling rules to constraint domains. A variable's quantity space can be seen as a finite domain. Therefore, we developed a domain library that allows to handle a quantity space as a totally ordered set of landmarks. Qualitative constraints are implemented using the Eclipse suspension library. Constraints that are not satisfied for all values in their domains are suspended and woken if one of the domains is updated by another constraint.

Knowledge Base and Simulator

We connected our simulator to EcoCyc (Karp *et al.* 1997), which is an Encyclopedia of *E. coli* genes and metabolism. EcoCyc consists of a collection of frames, where each frame encodes information about single objects, such as an enzyme, a gene or a biochemical pathway. We translated EcoCyc from Lisp into Prolog to obtain a relational knowledge base that can be queried us-

ing the expressive clauses of Prolog. The major classes of EcoCyc are *compounds and elements*, *enzymatic reactions*, *genes*, *pathways*, *proteins* and *reactions*.

Each class contains a set of slots describing properties of an object. From those slots we derive qualitative constraints for automatic generation of simulation models. Each object in EcoCyc has a unique identifying name in slot *unique-id*. We use this name for naming objects and processes. Each class provides slots such as *common-name* or *synonyms* that can be used to determine the print name of an object or process. The class *reactions* holds specific slots such as *left* and *right* which contain the compounds of the left and right side of the slot *reaction-equation*. This information suffices to determine the qualitative functional dependencies between compounds. In Section Implementation we explain in more detail, how to automatically generate a basic simulation model from here.

Algorithm

BioSim predicts a set of possible qualitative behaviours represented by one or more behaviour trees. Each node represents a state of the entire system described by all values of object parameters at one point in time. Root nodes are initial states and leaves are final states. The behaviour is then represented by a path from an initial state to a final state. A path which contains an inconsistent state does not represent a valid behaviour. The behaviour tree can be generated in breadth or depth-first order.

Algorithm 6 *Given a set of objects O, BioSim generates all initial states I and initialises agenda A with I. Given a set AP containing the parameters of all objects and a set of processes P the algorithm proceeds as follows:*

S1) If $A = \emptyset$ or a resource limit has been exceeded the algorithm stops.
Otherwise, get a state s from agenda A in the user defined order (breadth-first or depth-first).

S2) Initialise set V of active object parameters to $V = \emptyset$. C is the set of constraints applicable to object parameters in V and set to $C = \emptyset$.

S3) For each process $p \in P$:
- *Let PP be the set of preconditions and PC the set of constraints of process p.*
- *Test each precondition $pp \in PP$ with the actual object parameter values of state s.*
  - *If successful let U be the set of parameters constrained by PC.*
    Let $V = V \cup U$ and $C = C \cup PC$.

S4) If $C = \emptyset$ then state s is said to be final.
Continue from step S1.

S5) For each object $o \in O$:
- *Let OP be the set of parameters and let OC be the set of constraints inherent to object o.*
- *If $op \in OP \wedge op \in V$ then let $V = V \cup OP$ and $C = C \cup OC$.*

S6) Initialise the set of possible successor values S for all object parameters and set to $S = \emptyset$.

S7) For each parameter $v \in AP$, compute SV to be the set of possible successor values of parameter v by applying the transition rules of Table 2. Let $S = S \cup SV$.

S8) Let $NV = AP \setminus V$ the set of parameters that are not constrained by any $c \in C$.
For each $nv \in NV$ assume nv to be constant and let $C = C \cup \{constant(nv)\}$.

S9) With each constraint $c \in C$, filter the set of possible transitions S for consistency. Let R be the set of the filtered successor values.

S10) From R generate a set N of new states as successors to the current state s.

S11) For each $n \in N$ unless:
- *$pred(n, pr)$: state n matches with a predecessor state pr, or*
- *$divergence(n)$: the qualitative magnitude of at least one object parameter is $+inf$ or $-inf$, i.e. an endpoint has been reached*

add state n to agenda A.

S12) Continue from step S1.

The behaviour of a system is described by changes of the qualitative value of parameters. The manner of changing the qualitative value of a parameter while changing from one state to another is restricted by a set of state transitions. Table 2 shows the transition table used to determine successor values for a given qualitative value.

Implementation

The basic simulation model for glycolysis is generated automatically. Therefore, the user needs to identify the items of interest from the knowledge base, for example by selecting a pathway. The class *pathways* contains a slot *net-reaction-equation* which describes the chemical transformation accomplished. The net reaction equation of glycolysis is: $glucose + 2\ P_i + 2\ ADP + 2\ NAD^+ \rightarrow 2\ pyruvate + 2\ ATP + 2\ NADH + 2\ H^+ + 2\ H_2O$. The slot *reaction-list* contains all reactions of the current pathway. From the reaction slots *left* and *right* we recursively retrieve all substrates involved. Table 3 shows all substrates and reactions that are generated as objects and processes in our example.

An object's name is retrieved from the slots *left* and *right* of a reaction and the print name from slot *common-name* or *synonyms* of the substrate. The following qualitative parameters and quantity spaces are added to each object:

- `produced`, `[0,inf]`,
- `degraded`, `[-inf,0]` and
- `concentration`, `[0,inf]`.

The parameter `produced` is used to reflect newly produced molecules during simulation, while the parameter

	$(\text{qmag}, \text{qdir})_{s_i}$	$\Rightarrow$	$(\text{qmag}, \text{qdir})_{s_{i+1}}$
1)	(l_j, std)		(l_j, std)
2)	(l_j, std)		(l_{j+1}, inc)
3)	(l_j, std)		$((l_j, l_{j+1}), \text{inc})$
4)	(l_j, std)		(l_{j-1}, dec)
5)	(l_j, std)		$((l_j, l_{j-1}), \text{dec})$
6)	$((l_j, l_{j+1}), \text{std})$		$((l_j, l_{j+1}), \text{std})$
7)	$((l_j, l_{j+1}), \text{std})$		(l_{j+1}, inc)
8)	$((l_j, l_{j+1}), \text{std})$		$((l_j, l_{j+1}), \text{inc})$
9)	$((l_j, l_{j+1}), \text{std})$		(l_j, dec)
10)	$((l_j, l_{j+1}), \text{std})$		$((l_j, l_{j+1}), \text{dec})$
11)	(l_j, inc)		(l_{j+1}, inc)
12)	(l_j, inc)		(l_{j+1}, std)
13)	(l_j, inc)		$((l_j, l_{j+1}), \text{inc})$
14)	(l_j, inc)		$((l_j, l_{j+1}), \text{std})$
15)	$((l_j, l_{j+1}), \text{inc})$		(l_{j+1}, inc)
16)	$((l_j, l_{j+1}), \text{inc})$		(l_{j+1}, std)
17)	$((l_j, l_{j+1}), \text{inc})$		$((l_j, l_{j+1}), \text{inc})$
18)	$((l_j, l_{j+1}), \text{inc})$		$((l_j, l_{j+1}), \text{std})$
19)	(l_j, dec)		(l_{j-1}, dec)
20)	(l_j, dec)		(l_{j-1}, std)
21)	(l_j, dec)		$((l_{j-1}, l_j), \text{dec})$
22)	(l_j, dec)		$((l_{j-1}, l_j), \text{std})$
23)	$((l_j, l_{j+1}), \text{dec})$		(l_j, dec)
24)	$((l_j, l_{j+1}), \text{dec})$		(l_j, std)
25)	$((l_j, l_{j+1}), \text{dec})$		$((l_j, l_{j+1}), \text{dec})$
26)	$((l_j, l_{j+1}), \text{dec})$		$((l_j, l_{j+1}), \text{std})$

Table 2: Transition table of BioSim. $(\text{qmag}, \text{qdir})_s$ represents a qualitative magnitude, qmag and direction, qdir at state s. l represents a landmark in a quantity space. The notation (l_j, l_{j+1}) represents an interval in the range of two neighboured landmarks. Assuming a quantity space $[0, a, \text{inf}]$ and a qualitative value (a, inc) at state s_2 the possible successor values at state s_3 are determined from 11), 12), 13) and 14) to be (inf, inc), (inf, std), $((a, \text{inf}), \text{inc})$ and $((a, \text{inf}), \text{std})$. An exception to the transition table are the qualitative magnitudes +inf and −inf which have no successor values.

degraded reflects consumed molecules. The quantity space of parameter degraded is defined as `[-inf,0]`. Although no negative biological quantities exist, this has been done for easy combination with the addition constraint. The parameter `concentration` results from the qualitative addition of generated and degraded molecules, represented by the qualitative constraint `add(produced, degraded, concentration)`. For the current example, we assume `concentration` to be constant if parameter produced is increasing and degraded is decreasing. Although not true in the general case this is in accordance with the stoichiometry of the current application. If parameter degraded is constant and produced is increasing then `concentration` is monotonically increasing, too. Appropriate constraints are generated automatically. Initially, the software asks the user to identify the absence or presence of compounds in the simulation scenario. For those that are

glucose + ATP → glucose-6-phosphate + ADP + H^+

glucose-6-phosphate ⇌ fructose-6-phosphate

fructose-6-phosphate + ATP → fructose-1,6-diphosphate + ADP + H^+

fructose-1,6-diphosphate ⇌ dihydroxy-acetone-phosphate + glyceraldehyde-3-phosphate

dihydroxy-acetone-phosphate ⇌ glyceraldehyde-3-phosphate

glyceraldehyde-3-phosphate + P_i + NAD^+ ⇌ 3-phospho-D-glyceroyl-phosphate + NADH + H^+

3-phospho-D-glyceroyl-phosphate + ADP ⇌ 3-phosphoglycerate + ATP

3-phosphoglycerate ⇌ 2-phosphoglycerate

2-phosphoglycerate ⇌ phosphoenolpyruvate + H_2O

phosphoenolpyruvate + ADP + H^+ → pyruvate + ATP

Table 3: This table shows the reactions of the glycolysis pathway. In our simulation model we represent all substrates, enzymes, reactions and enzymatic reactions.

said to be present the initial value of produced and `concentration` is `((0,inf),std)` and for degraded it is `((-inf,0),std)`. All other parameters of the remaining objects are assigned to be `(0,std)`.

For each reaction we generate a process. The process name is retrieved from the *reaction-list* and the print name from the *common-name* slot of the reaction.

The set of preconditions of a process is generated from all substrates appearing in the slot *left* of the reaction. The precondition to activate a process requires all reactants to have a concentration greater than 0, e.g for the `PGLUCISOM-RXN` in Figure 1 the precondition is `GLC-6-P(concentration) gt 0`.

We can state that for all substrates in slot *left* the absolute value of parameter degraded is increasing. For all substrates in slot *right* the parameter produced is increasing. Further, any increase of products (produced) implies a decrease of reactants (degraded). From these statements we generate the appropriate constraints and add them to the set of constraints of a process. Figure 1 gives an example of a process where `m-(GLC-6-P(degraded), FRUCTOSE-6P(produced))` represents a monotonic function constraint (m-), saying that while glucose-6-phosphate is degraded fructose-6-phosphate is produced.

Enzymatic activity can be represented, too. From the slot *enzymatic-reaction* we can retrieve information about the reaction and enzymes. We automatically generate the appropriate objects and processes. A unit of enzymatic activity is qualitatively defined as the amount required to transform substrate per time. The parameter produced of the product represents the transformed substrate. Therefore, we can add an appropriate qualitative derivative constraint. Figure 1 gives an example, `d/dt(FRUCTOSE-6P(produced), PGLUCISOM(activity))`, saying that the enzymatic activity of phosphoglucose isomerase is linked to the in-

crease of newly produced fructose-6-phosphate.

Although glucose is present in slot *net-reaction-equation*, we found the slot *reaction-list* of object glycolysis in EcoCyc missing the reaction of glucokinase. To eliminate this inconsistency we extended our model by adding the missing reaction.

Results

The simulation run of the above described model results in a tree with possible behaviours. Figure 2 shows one behaviour that is discussed below. The figure is the graphical output of a simulation run using BioSim. The qualitative change of parameter values is one of increasing (↑), steady (o) or decreasing (↓). The dotted line between those symbols and its slope have no significance other than visual guidance. The time labels on the x-axis do not represent a constant time interval. They represent critical states of the system where the qualitative values of parameters change.

The initial time point is marked 0 in the simulation charts. This indicates the initial state of the pathway. Only a limited amount of glucose, phosphate, ATP, ADP and NAD^+ (not shown) are defined to be initially present.

The simulation engine dynamically computed the events described in the following paragraph by checking available processes, evaluating constraints for their activation and monitoring the presence or absence of compounds as explained in the algorithm section above. Each state represents a moment in time that is characterised not only by the key elements cited in the text, but by all object parameters that are connected to glycolysis.

In the first stage of glycolysis, from t_1 to t_2, phosphate is transferred from ATP to glucose yielding glucose-6-phosphate. From t_4 to t_5 we can observe the isomerisation of glucose-6-phosphate to fructose-6-phosphate. This reaction is catalysed by phosphogluco-isomerase. A phosphorylation reaction occurs at t_7 when 6-phosphofructokinase-1 becomes active. Fructose-6-phosphate is phosphorylated by consumption of ATP to fructose-1,6-diphosphate, from t_7 to t_8. With the activity of fructose-bisphosphate-aldolase at t_{10} we enter the second stage of glycolysis. Fructose-1,6-diphosphate is split to yield dihydroxy-acetone-phosphate and glyceraldehyde-3-phosphate. From t_{13} glyceraldehyde-3-phosphate is converted into 3-phospho-D-glyceroyl-phosphate. The active enzyme is glyceraldehyde-phosphate-dehydrogenase. At the same time the enzyme triose-phosphate-isomerase catalyses the isomerisation of dihydroxy-acetone-phosphate to glyceraldehyde-3-phosphate. Note, the actual time passing between two successive states can vary. The only implication is that qualitative values of parameters change. At t_{19} we can observe the first ATP-generating reaction. From 3-phospho-D-glyceroyl-phosphate and ATP we get ADP and 3-phosphoglycerate. Phosphoglycerate-kinase is active from t_{20} to t_{21}. From t_{23} to t_{24} phosphoglyceromutase is active and 3-phosphoglycerate is converted into 2-phosphoglycerate. Phosphoenolpyruvate is formed by dehydration of 2-phosphoglycerate, from t_{26} to t_{27}. Pyruvate-kinase is active from t_{29} to t_{30} and the second ATP-generating reaction of glycolysis takes place. Pyruvate and ATP are formed by transfer of a phosphoryl group from phosphoenolpyruvate to ADP.

Related Work

A knowledge-based simulation of gene regulation is presented in (Meyers & Friedland 1984) were they showed the reproductive behaviour of λ phage without detailed kinetic data. A simulator database contains the data about λ phage and the rules for the simulator. However, their article does not make clear how far the simulation depends on numeric information integrated in the database and what are the connectivities between the different components of their model. Our simulator provides a basic set of qualitative constraints that represent the connectivities between object parameters.

Order-of-magnitude reasoning can be used to combine qualitative and quantitative knowledge for the analysis of biochemical pathways, as shown in (Mavrovouniotis & Stephanopoulos 1990). This reasoning method provides approximate relations among quantities, such as "A is much smaller than B" or "C is slightly greater than D". How this method can be combined with envisionment of QDEs is shown in (Davis 1990).

The work of (Thomas 1991) is also concerned with the dynamic behaviour of molecular biological systems. His main issue, however, is the logical structure of biological systems and how that can be conveniently formalised without actually running simulations of biological systems.

A system for knowledge-based simulation of DNA repair based on the *Knowledge Engineering Environment*, KEE, is presented by (Brutlag, Galper, & Millis 1991). In their simulation available quantitative information is mapped into symbolic ranges for qualitative reasoning. They argue that this will reduce the number of rules produced. In contrast to our work, any activity in their model has to be specified in IF-THEN-ELSE rules and object definitions with the disadvantage of having to circumscribe direct qualitative arithmetic relations as constraining pseudo-rules. The authors also employ non-monotonic reasoning with the drawback that their actions cannot be logically explained in retrospection since they modify the knowledge base in a potentially destructive way.

Gensim of (Karp 1993) was also used to simulate molecular biological systems. Gensim is based on production rules that can be used to pre-program the behaviour of a simulated system. However, the user gains more flexibility in modelling with the object and process oriented methods presented in this paper than with fixed rules.

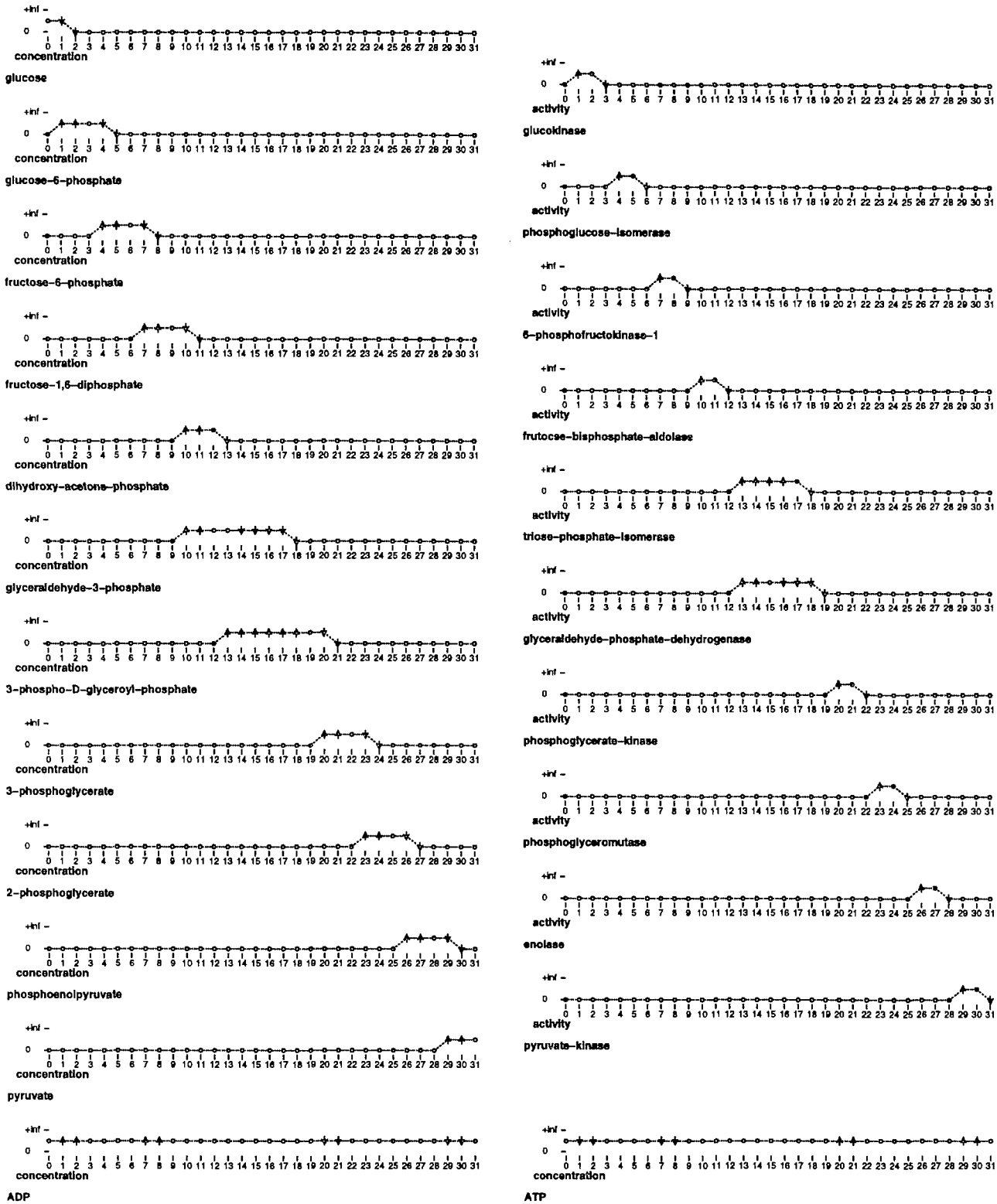

Figure 2: A qualitative simulation of the glycolysis pathway. Each object was produced by BioSim. The symbols denote whether the magnitude of a parameter is increasing (↑), steady (○) or decreasing (↓). The slope of the connecting dotted lines does *not* carry any information on the magnitude or speed of change of a parameter. The labels of the x- and y-axes are sequences of symbols representing distinguished time points on the x-axis and distinguished landmark values in a quantity space of each parameter on the y-axis.

The work of (Shimada *et al.* 1995) is a knowledge-based simulation on two levels of abstraction demonstrated on the protein regulated genetic action of the decision between the lytic and lysogenic pathway of λ phage after infecting *E. coli*. The first level describes the model in qualitative terms as object oriented concepts. On the second level quantitative information is added. In addition to the rules describing the interaction of proteins, rules for synchronisation are needed some of which depend on manually prescheduled events. This contradicts the essential significance of simulation and has no biological foundation, as the authors themselves acknowledge. In BioSim, parameters are functions over time and take their values out of an ordered set of landmarks. Successor values of a parameter are determined by a set of possible transitions so that there is neither a need nor a possibility to manually pre-program simulation events.

Another approach to handle qualitative behaviour is the work of (McAdams & Shapiro 1995) who used a circuit simulation to simulate the lytic and lysogenic pathway of λ phage after infecting *E. coli*. At some stage their circuit simulation requires kinetic models and artificial quantities. They postulate the existance of a hypothetical "post-reaction gene product" to account for enzymatic activity of CII and CIII proteins.

In BioSim we can avoid this by introducing landmarks in a parameter's quantity space. Landmarks describe qualitative distinct regions and can be used as thresholds to activate other processes which then may invoke additional qualitative constraints.

An expert system based on the KEE development shell with simulation capability for post-cardiosurgical patients in intensive care units is described by (Ursino, Avanzolini, & Barbini 1992). Qualitative and quantitative information is used together. Like BioSim, the qualitative status of quantities is represented in their value and direction of change. The value of a quantity can be "low", "normal" or "high". For diagnostic systems in medicine this set of landmarks is suitable (Ursino *et al.* 1994) but they state that a limitation to use only three landmarks is a problem when numerical values are mapped onto landmarks (Artoli *et al.* 1996).

In (Heidtke & Schulze-Kremer 1998) we presented a qualitative simulation model of an *E. coli* bacterium infected by λ phage. We used the qualitative simulator QSim to simulate the lysogenic and lytic behaviour of λ phage in an infected *E. coli*. Our model of *E. coli* consists of several QDEs where each represents a stage from the very early to the late lysogenic or late lytic stage. Different stages, encoded in separate QDEs, put different constraints on the simulation variables. The major problem with QSim is that it does not provide an easy way to constrain variables across different QDEs. In our new approach a process can be triggered at any time during simulation if only its preconditions are met. In that way we overcome the limitation of having to split the simulation model into a temporarily ordered sequence of events.

A simulation of the glucose-insulin regulatory system is presented in (Clancy & Kuipers 1997). The authors present a new algorithm DecSim based on QSim. The given model is manually partitioned into smaller components with each component simulated separately, thereby reducing the complexity of the model. The interactions between components are reasoned about separately. Like QSim, DecSim does not provide objects and processes which are the means of achieving modularity in BioSim.

Discussion

Several genome sequencing projects have been completed or are on the way which generate large quantities of sequence and other biological information. Biology has become "functional genomics" to a significant extent and an appropriate representation of the myriad of experimental facts has become critical to get the most out of the data. Computer simulation of biological processes cannot be a replacement for real experiments but in general can be useful for a number of purposes:

- tutoring systems for teaching the complex interactions of biological systems;
- information systems that combine data from different sources on one subject and make it interactively accessible;
- a testbed for detecting inconsistencies and missing links among a potentially large number of empirical facts;
- if-then analysis to explore the effects of changing one or few components in a larger system;
- and, once a simulation has been validated, an animated scenario for generating time series to predict behaviour.

Qualitative computer simulation models can operate on a wide range of biological information (Kazic 1993b) and are therefore suitable to utilise a greater portion of present genomic and functional data than pure numerical simulation alone can do. This kind of information is available from sequencing and homology modelling and from hybridisation studies before a detailed biochemical analysis of the individual system has been done. Symbolic, qualitative simulation allows to assemble a consistent model even if no quantitative data about enzyme catalysis rates or substrate concentrations are available.

In this report, we present a new methodology to perform qualitative simulation that combines the key features of two well established simulation engines. Our approach enables structured model building using objects and processes and at the same time allows application of functional constraints to characterise a model's behaviour for any time interval. The performance of our approach has been evaluated in three ways:

- qualitative equations in objects and processes can be directly interpreted in biological terms and thus examined on their biological validity;

- qualitative plots show that our simulation captures in fine detail the biological processes involved; and
- the simulation produces biologically meaningful and correct results.

Using our qualitative simulation environment biologists can:

- store and utilise non-numerical, qualitative knowledge,
- inspect functional dependencies between molecular chemical compounds in a knowledge base,
- test a set of empirical facts for inconsistencies or missing information,
- and experiment with alternative hypotheses *in silico* to see whether other models would be equally well suited to explain a given set of experimental evidence.

The presented simulation environment incorporates the basic ideas of qualitative reasoning, i.e. an object and process oriented view (Forbus 1984), and the manner of qualitatively constrained variables (Kuipers 1986).

We will continue our work on this qualitative simulator to extend constraint propagation, resolution facilities and tree pruning. For example, the behaviour of a model can be further constrained by using common quantity spaces for different parameters and by detecting states that have been already expanded in another branch of the simulation tree. Furthermore, we plan to implement a numeric constraint solver to handle available quantitative information, as well as to include built-in primitives for basic molecular biological processes as e.g. transcription and translation. We are also working on the development of further conversion tools to automatically import information from other molecular biology databases into the qualitative simulation environment.

Acknowledgments

We are indebted to Peter Karp for his consent to employ the data of EcoCyc in our system.

References

Aggoun, A.; Chan, D.; Dufresne, P.; Falvey, E.; Grant, H.; Herold, A.; MacCartney, G.; Meier, M.; Miller, D.; Mudambi, S.; Perez, B.; van Rossum, E.; Schimpf, J.; Tsahageas, P. A.; and de Villeneuve, D. H. 1995. Eclipse 3.5 user manual. http://www.ecrc.de/eclipse/eclipse.html.

Artoli, E.; Avanzolini, G.; Martelli, L.; and Ursino, M. 1996. An expert system based on causal knowledge: validation on post-cardiosurgical patients. *International Journal of Bio-Medical Computing* 41:19 – 37.

Brisset, P.; Früwirth, T.; Gervet, C.; and Lim, P. 1995. Eclipse 3.5 extension user manual. http://www.ecrc.de/eclipse/eclipse.html.

Brutlag, D. L.; Galper, A. R.; and Millis, D. H. 1991. Knowledge-Based Simulation of DNA Metabolism: Prediction of Enzyme Action. *Computer Applications in Biosciences* 7(1):9 – 19.

Clancy, D. J., and Kuipers, B. 1997. Model decomposition and simulation: A component based qualitative simulation algorithm. In *The 14th National Conference on Artificial Intelligence (AAAI-97).* Rhode Island: AAAI Press.

Crawford, J.; Farquhar, A.; and Kuipers, B. 1990. QPC: A compiler from physical models into qualitative differential equations. In *Proc. 8th National Conf. on Artificial Intelligence (AAAI-90)*, 365–372. San Mateo, CA: Morgan Kaufmann.

Dague, P. 1995. Qualitative reasoning: A survey of techniques and applications. *AI Communications* 8(3/4):119 – 192.

Davis, E. 1990. Order of magnitude reasoning in qualitative differential equations. In Weld, D. S., and de Kleer, J., eds., *Readings in Qualitative Reasoning about Physical Systems.* San Mateo, CA: Morgan Kaufmann. 422–434.

Forbus, K. D. 1984. Qualitative process theory. *Artificial Intelligence* 24:85–168.

Forbus, K. D. 1990. The qualitative process engine. In Weld, D. S., and de Kleer, J., eds., *Readings in Qualitative Reasoning about Physical Systems.* San Mateo, CA: Morgan Kaufmann. 220–235. http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/reasonng/qualittv/qpe/0.html.

Frühwirth, T.; Herold, A.; Küchenhoff, V.; Provost, T. L.; Lim, P.; Monfroy, E.; and Wallace, M. 1992. Constraint logic programming - an informal introduction. In Comyn, G., ed., *Logic Pragrammming in Action: Second International Logic Programming Summer School*, volume 636. Zürich: Springer-Verlag. 3–35. Springer Lecture Notes.

Heidtke, K. R., and Schulze-Kremer, S. 1997. Qualitative reasoning on gene control and reproduction demonstrated on λ phage growth. In Frishman, D., and Mewes, H. W., eds., *Proceedings of the German Conference on Bioinformatics GCB'97*, 37–40. MIPS-GSF Martinsried.

Heidtke, K. R., and Schulze-Kremer, S. 1998. Design and implementation of a qualitative simulation model of λ phage infection. *Bioinformatics.* in press.

Karp, P. D., and Mavrovouniotis, M. M. 1994. Representing, analayzing, and synthesizing biochemical pathways. *IEEE Expert* 9(2):11 – 22.

Karp, P. D.; Riley, M.; Paley, S.; Pellegrini-Toole, A.; and Krummenacker, M. 1997. EcoCyc: Encyclopedia of *E. coli* genes and metabolism. *Nucleic Acids Research* 25(1):43 – 50.

Karp, P. D. 1993. A qualitative biochemistry and its application to the tryptophan operon. In Hunter, L.,

ed.; *Artificial Intelligence and Molecular Biology*, 289 – 324. AAAI Press.

Kazic, T. 1993a. Reasoning about biochemical compounds and processes. In H.A. Lim, J.W. Fickett, C. C., and Robbins, R., eds., *Second International Conference on Bioinformatics, Supercomputing and the Human Genome Project*, 35 – 49. Singapore: World Scientific.

Kazic, T. 1993b. Representation, reasoning and the intermediary metabolism of *E. coli*. In T.N. Mudge, V. M., and Hunter, L., eds., *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, volume 1, 853 – 862. Los Alamitos, CA: IEEE Computer Society Press.

Kuipers, B., and Kassirer, J. 1987. Knowledge acquisition by analysis of verbatim protocols. In Kidd, A., ed., *Knowledge Acquisition for Expert Systems*, volume 29, 289 – 338.

Kuipers, B. 1986. Qualitative simulation. *Artificial Intelligence* 29:289 – 338.

Kuipers, B. 1994. *Qualitative Reasoning. Modeling and Simulation with Incomplete Knowledge.* Cambridge, MA: MIT Press. http://www.cs.utexas.edu/users/qr/QR-software.html.

Mavrovouniotis, M. L., and Stephanopoulos, G. 1990. Formal order-of-magnitude reasoning in process engineering. In Weld, D. S., and de Kleer, J., eds., *Readings in Qualitative Reasoning about Physical Systems.* San Mateo, CA: Morgan Kaufmann. 327–336.

McAdams, H. H., and Shapiro, L. 1995. Circuit simulation of genetic networks. *Science* 269:650–656.

Meyers, S., and Friedland, P. 1984. Knowledge based simulation of genetic regulation in bacteriophage lambda. *Nucleic Acids Research* 12(1):1–9.

Schulze-Kremer, S. 1990. CS-Prolog Parallel Programming in Logic with Transputers. Application: MolSIM, a Program for Simulation of Concurrent Molecularbiological Processes. In Turner, S. J., ed., *12. OUG Conference, Exeter, Tools and Techniques for Transputer Applications*, 97 –110. Amsterdam: IOS Press.

Schulze-Kremer, S. 1995. *Molecular Bioinformatics: Algorithms and Applications.* Berlin, New York: de Gruyter.

Shimada, T.; Hagiya, M.; Arita, M.; Nishizaki, S.; and Tan, C. 1995. Knowledge-based Simulation of Regulatory Action in Lambda Phage. *International Journal of Artificial Intelligence Tools* 4(4):511–524.

Teleki, L. 1996. Constraint logic programming: A framework for qualitative reasoning. In *Proceedings of the Tenth International Workshop on Qualitative Reasoning.*

Thomas, R. 1991. Regulatory Networks Seen as Asynchronous Automata: A Logical Description. *Journal of Theoretical Biology* 153:1 – 23.

Ursino, M.; Avanzolini, G.; and Barbini, P. 1992. Qualitative simulation of dynamic physiological models using the KEE environment. *Artificial Intelligence in Medicine* 4:53 – 73.

Ursino, M.; Artioli, E.; Avanzolini, G.; and Potuto, V. 1994. Integration of quantitative and qualitative reasoning: an expert system for cardiosurgical patients. *Artificial Intelligence in Medicine* 6:229 – 247.

Weld, D. S., and de Kleer, J. 1990. *Readings in Qualitative Reasoning About Physical Systems.* San Mateo, CA: Morgan Kaufmann.