1 Introduction

Historically, the first attempts to study cell regulation were based on establishing systems of non-linear differential equations that "simulate" the behavior of the variables involved, as a function of time. In that context the variables correspond to genes capable of producing compounds (mRNA and proteins) that in turn influence the production capacity of other genes. This problem has been thoroughly studied in the mathematical theory of dynamic systems.

The advent of micro-arrays urgently motivates the solution of the *inverse* problem: given the curves expressing the capacity of each gene of producing known compounds as time progresses, attempt to infer the original system of equations or equivalently, the graph expressing the interactions among genes (see Appendix A for a brief overview of the desirable biological background). Presently, biologists are capable of generating huge sets of data expressing the amount of production of proteins or mRNA as a function of time [26]. One then wishes to establish the corresponding gene interaction graph. The inverse problem may be called *modeling* in contrast with *simulation*, the term used to describe the direct problem of determining the behavior of the variables in a system of differential equations.

There have been various attempts to provide discrete solutions to the system of differential equations describing cell regulation. By considering Boolean variables, instead of continuous variables, one can simplify remarkably the complexity of the original problem. Several papers are now available that describe solutions of both the direct and inverse problem using the Boolean approach.

An important aspect of the inverse problem is that the data obtained by biologists is not reliable, in the sense that many experimental errors occur when dealing with thousands of genes. It then becomes imperative to take into account a probabilistic approach of the inverse problem. In other words, given imperfect data expressing the behavior of variables with time, can one infer the corresponding gene interaction graph ?

More recently this last problem has been simplified by having biologists conjecture the interaction graph and propose the simpler question: How likely is the hypothesized graph capable of generating the given data set ? (See for example [24])

An initial classification of the existing methods proposed in the literature to solve the direct and inverse problems involves three parameters and thus eight subclasses:

Probabilistic versus Nonprobabilistic
Continuous versus Discrete
Direct versus Inverse

In a recent article de Jong and Page [8] propose a *Direct-Discrete-Nonprobabilistic* type of approach; that amounts to a qualitative (*à la* Kuipers [13]) simulation of the systems of differential equations representing a given influence graph. Even though their algorithms are implemented in a standard programming language, the authors make extensive use of constraints in formulating the solution to that problem.

In this paper we first propose the use of constraint logic programming as a paradigm for solving the discrete versions of the cell regulation problem. Basically one considers *regions* in an n-dimensional space, and *transitions* that specify possible paths between the regions. Both regions and transitions are specified by sets of constraints that restrict the values of the variables being studied. A path linking an initially specified region to subsequent regions via the transitions, expresses the behavior of the system. (That description is inherent to the approach taken by de Jong and Page.)

The inverse problem amounts to determining regions and transitions when given the paths that may be derived from the data expressing the behavior of the variables. From those regions and transitions one should be able to determine the desired gene interaction graph.

In the particular case of Boolean constraints – a special case among the discrete approaches – the determination of the regions and transitions amounts to the problem of synthesizing Boolean circuits from input and output data. The regions correspond to subsets of the input data and the transitions correspond to Boolean formulas that allow transforming the input data into output. Therefore the problem using Boolean variables can be expressed in terms of program synthesis or machine learning. Similarly, it is hoped that the synthesis of constraints from data should play a role in the discrete solution of the inverse problem.

An interesting payoff of the use of constraints is that they may well pave the way for probabilistic solutions to both the direct and indirect versions of the cell regulation problem. The approach reinforces the importance of developing probabilistic machine learning algorithms expressed in logic programming and in constraint logic programming.

Even though the emphasis in this paper is on approaches dealing with constraints, we found it worthwhile to provide brief descriptions of other methods, particularly the ones recently proposed in the AI area now called *Soft Computing*, which includes neural nets and belief networks. Some of these soft computing approaches have roots on the differential equations and constraints described in the sequel.

1.1 Objectives

The emphasis of this paper is to provide a unified view of the various approaches that have been proposed in the study of regulation. This is achievable by making extensive use of constraints, in the sense of CLP (Constraint Logic Programming). It will be seen that different versions of the direct problem, namely determining gene behavior as a function of time from given labeled graphs, can be described by the same general principles.

It will be shown in the following sections that most of the existing approaches to solve the direct problem are based on the concepts of *regions* (in n-dimensional space), *transitions* specifying how control moves from one a region to another, and *trajectories*, a sequence of consecutive transitions.

In most cases the inverse problem (modeling) can then be described as in logic programming: if the clauses specifying the program contain no impure constructs (i.e., all predicates are "backtrack able" in the sense of Prolog) then one should be able, at least theoretically, to perform inverse operations that, given the gene behavior with time, determine the possible labeled graphs by generate-and-test techniques. This may be not be practically achievable in the case of large problems using current computers, but the stress on defining clearly the operations involved in the solution of a problem has definite benefits. Incidentally, this is not unlike the *generate-and-test* approaches used in machine learning and inductive logic programming. The proposed framework may also lead to future approaches using synthesis of finite state automata.

As indicated in the title, the paper's aim is to find commonalities among existing approaches. Nevertheless, it is hoped that the insight gained by a unified view will be helpful in attempting to discover new variants that remain unexplored.

The structure of the paper is as follows. First we present the original formulation of the continuous approach using systems of non-linear differential equations. We then provide a simple motivating example that illustrates the concepts of regions, transitions, and trajectories. That is followed by presentations of the available discrete formulations using the proposed general framework of constraints. The respective subsections deal with approaches to solve the inverse problem. A section entitled "Soft Computing Approaches" deals with recent work on the modeling of cell regulation using neural nets, rough sets, Bayesian networks, and support vector machines. In the conclusion we make suggestions for extensions and further work.

2 The Direct Continuous Approach

Graphs expressing the interactions among genes are often used as convenient abstractions enabling the formulation of the appropriate systems of differential equations. The nodes of those graphs correspond to the variables (genes) and the directed edges – labeled by *plus* or *minus* signs – express the positive or negative interactions among the variables. In the case of cell regulation thousands of genes may interact with each other.

The continuous approach is based on the theory of dynamic systems. From a given regulation graph with n nodes one can always establish a set of n differential equations expressing the behavior of the system with time. The solution of these equations for a given initial condition yields the curves depicting how the genes are expressed as a function of time. The difficulty with this approach is that highly non-linear functions (sigmoids) are used to define the variation of each gene's expression with time.

One of the first approaches in simulating cell behavior consisted of setting up a system of non-linear differential equations of the form:

$$dx_i/dt = f_i(\boldsymbol{x}) - \gamma_i x_i \ , \ x_i \geq 0$$

where i denotes the i^{th} gene and $\boldsymbol{x}$ is the vector $(x_1, x_2, \ldots, x_n)$

The term $-\gamma_i x_i$ states that the concentration of the i^{th} product decreases through spontaneous processes like degradation, diffusion, etc.; f_{ij} is the function specifying a combination of sigmoids (highly non-linear) and which describes the interaction between nodes (genes) i and j; m is a parameter specifying the steepness of the function around θ_{ij} (see Figure 1)

$$f_{ij} = h^+(x_j, \theta_{ij}, m) = \frac{x_j^m}{x_j^m + \theta_{ij}^m}$$

We now open a parenthesis to describe by an example how a set of differential equations can be generated from a labeled graph expressing the interaction between genes. Each gene corresponds to a *node* in the graph, or equivalently, to a *variable* in the system of equations. The simple graph in Figure 2(a) contains three nodes and five edges expressing positive and negative interactions among the genes. The resulting equations are also shown in that figure. Basically, one has to consider different functions h^+ or h^- corresponding to sigmoids that express either a sharp increase (h^+) or a sharp decrease (h^-) of the variable being simulated.

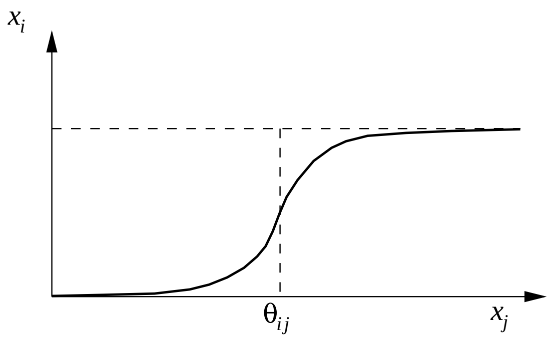
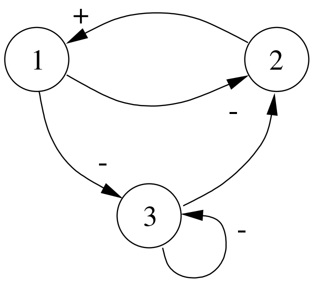


Figure 1: Sigmoid function.



$$\dot{x}_1 = k_{12} h^-(x_2, \theta_{12}, m) - \gamma_1 x_1$$
$$\dot{x}_2 = k_{21} h^-(x_2, \theta_{21}, m).k_{23} h^-(x_3, \theta_{23}, m). - \gamma_2 x_2$$
$$\dot{x}_3 = k_{31} h^-(x_1, \theta_{31}, m) + k_{33} h^-(x_3, \theta_{33}, m). - \gamma_3 x_3$$

(a) (b)

Figure 2: (a) A regulatory network modeled by the 3 equations in (b).

A remark is in order: a simple directed labeled graph does not provide the full information about the nature of the system of equations that can be generated. One can, for example, specify by additional graphical notation that the incoming edges to a node should be considered as being connected by a Boolean *and*, or a Boolean *or*. It will be seen that the Boolean and qualitative discrete type of approaches can incorporate those additional graphical notations. In Figure 2 it is assumed that the two edges entering node 2 are linked by an *and* connector; and the edges entering node 3 are linked by an *or* connector. That explains the product and the sum of the h functions in Figure 2 (b).

It is known from the theory of dynamic systems that the solution of a system with n variables – when given an initial condition specified by the vector $\boldsymbol{x}_0$ – can be described by an n-dimensional curve expressing the behavior of the system as a function of time. We call that curve a trajectory. It is known that trajectories may contain loops or may converge to a stationary point.

2.1 The Inverse Problem

It appears that this continuous type of approach makes it extremely difficult to obtain inverse solutions analitically.

2.2 Brief Outline of the Following Sections

In a computer implementation of the direct problem one has to find discrete solutions of the given system of equations. Some of the approaches that have been utilized in rendering the solution discrete involve either using Booleans or the so-called qualitative methods. Those approaches will be described by examples in later sections of the paper. In the next section we present a motivating example that illustrates the use of constraints in solving a problem that is akin to that of dynamical systems. It will be seen that the constraint formulation in that example can be generalized to cover both the Boolean and qualitative approaches.

3 A Motivating Example

Figure 3 presents an intuitive example of regions, transitions, and trajectories in two-dimensional space. In this simplified version of the direct problem one specifies regions by linear constraints. The arrows in that figure specify the transition slopes applicable to each of the four regions. Initially, we deal with deterministic situations, so that there is a single arrow associated to each region.

The direct problem then becomes: given an internal initial point in one of the regions, determine a trajectory, i.e., a sequence of line segments that start at the given point and have the slopes as specified by the arrows. (To avoid ambiguous situations we assume that the arrow associated to a region is not directed towards the intersection of two lines defining that region. One also assumes that a trajectory does not contain any of the lines defining the regions' boundaries.)

Asarin, Maler, and Pnueli [6] have shown that in two-dimensional real space the following property is decidable: given two points P_1 and P_2 one can determine if the trajectory starting from P_1 passes through P_2. The complexity of the problem is exponential. The authors also proved that that property *is not* decidable in three-dimensional and higher spaces.

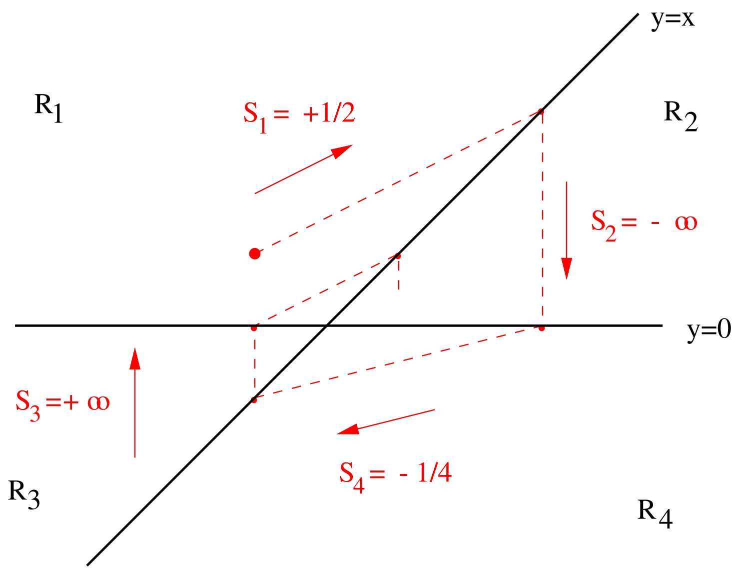


Figure 3: Example of linear constraints and slopes.

Let us assume that regions and their associated slopes are specifiable by constraints; one can then sketch a logic program that determines trajectories that start at a given point. That program is presented in Figure 4.

The predicate `region` specifies a region defined by constraints. The predicate `transition` defines the next region accessible from a given region via a given `slope`. One can then define the trajectory as the sequence of regions that are traversed from a given starting point. One such trajectory is also shown in Figure 3.

In the program of Figure 4, `trajectory` is a predicate that determines a sequence of transitions having n segments, n being specified by a user. Notice that the concept of trajectory is time-related. By considering that each transition takes a unit time to be simulated, one can plot a curve expressing how a property of the regions varies with time, given an initial starting point. From now on we may refer to a trajectory as $T(t)$, a function of the variable t expressing time.

A few remarks are in order about the relationship between the example in Figure 3 and the area of "Hybrid Systems" (see [11]). These systems are characterized by dynamics that are continuous at some levels and discrete at others. For example, a hybrid automaton can be used to define the behavior of a feeder that fills a tank T_1 with a liquid until the it reaches a certain level in which case the feeder changes state and starts filling a second tank T_2 . When the level at T_2 reaches another pre-assigned value the feeder re-starts feeding T1 and so forth.

In the above example time varies continuously while a tank is being filled and discretely when a change of tanks occurs. The example of Figure 3 could be interpreted in a similar vein. Time would change continuously in certain parts of a trajectory and when certain bounds of coordinates are attained then a change of state would occur.

Hybrid systems are often used in studying the complex behavior of industrial plants that are automatically controlled by sensors and valves. The theoretical problems involved in the study of hybrid systems have to do with, among others, the concepts of reachability (e.g., would a critical state ever be reached from a given initial condition?), or verification (e.g., does the system behave as designed?). I believe that many desirable links have yet to be established between the hybrid system's community and that studying cell regulation (see URLs on Hybrid Systems [28]).

3.1 The Inverse Problem

The inverse problem corresponding to the example of Figure 3 can be stated as follows: given a set of sequences of regions traversed from given initial points determine the constraints that constitute the regions and slopes. In other words, given $T(t)$ determine the corresponding regions and slopes. This is a typical problem in machine learning and a challenge is made to practitioners of inductive logic programming to attempt to solve the problem of Figure 3 in the two-dimensional space using linear constraints (The challenged reader may want to consult the work of Washio and Motoda [18] dealing with the generation of simultaneous equations from data).

Notice that even though the direct problem is linear, the inverse one *is not*, since the coefficients of the linear equations and inequations are unknown.

It appears that, in the two-dimensional case, a trajectory can also be expressed by a regular expression in the vocabulary of regions. (An argument to that effect is that the number of regions is finite and that the sequence of regions in a trajectory eventually repeats itself; see Asarin, Maler, and Pnueli [6] for a proof in the case of 2D).

This suggests that the direct problem in 2D consists of finding the regular expressions describing the trajectories when given a set of initial points, the constraints specifying the regions and the respective slopes. The inverse problem would consist of: given the initial states and the regular expressions describing an experiment, determine the constraints and slopes that would yield the given data.

One can also express a sequence of transitions by a directed graph: its nodes represent the regions and the edges any possible transitions. This description corresponds to a finite-state transition graph having unlabeled edges.

A non-deterministic logic program that would try all possible combinations of possible slopes could simulate the non-deterministic case of regions having multiple slopes. This would likely result in a highly combinatorial program with exponential complexity. In that case the finite-state machine describing all possible transitions would be non-deterministic since there may well be two edges emanating from a state representing a region.

Actually one could think of a brute force approach to solving the inverse problem using interval constraints as proposed by Older and Velino [15]. In this case each variable would be initially defined as being in the interval $[-\infty, +\infty]$. The constraints would be used to attempt to narrow down the intervals. Note that interval constraints can be conveniently used to solve non-linear systems of equations, which is the case of the inverse problem.

A more restricted version of the inverse problem would be: given all regions and slopes except that of a single region, determine its value so that the resulting data has a given configuration. In that case one would use interval constraints non-deterministically by splitting the interval being considered for the unknown slope.

4 The Direct Boolean Approach

Let us now consider the case of the Boolean domain with n variables. An n-digit binary number represents each region. A "slope" in the example of Figure 5 can be viewed as a Boolean formula that, when applied to a given binary number, yields another binary number also representing a region. That situation is depicted in Figure 7(a). Notice that essentially the same framework program in Figure 6 is applicable, provided that slopes are defined by Boolean formulas. That example is taken from Glass and Kauffman [10].

Actually the Boolean formulas may contain *and* and *or* operators, and can therefore be generated from the directed graph expressing the interactions among genes with additional notations to specify conjunctions and disjunctions, as suggested in the Section 2.

A trajectory, in the sense of the example of Figure 7(a), then becomes a sequence of binary numbers. As in Section 3 individual transitions are assumed to take unit time. It should be obvious that since we are dealing with a finite number (2^n) of regions the determination of all possible trajectories – starting from each possible region – is an exponential problem.

It should also be clear that trajectories could be represented by a finite graph, which may contain loops (including self loops involving a single node). Figure 7(a) represents all the trajectories for the input graph considered in Figure 5. Finally, Figure 7(b),(c) and (d) shows how the two variables change with time when starting from the initial configurations depicted in the figure.

4.1 A More Complex Example

It is simple to develop programs similar to the one presented in Figure 6 to deal with more complex graphs. For example, the graph in Figure 8 (a), yields the results presented in (c) and (e). In this particular example, the transitions are given explicitly (*in lieu* of a boolean formula) by the contents of the table in Figure 8 (b).

4.2 The Inverse Boolean Approach

The inverse Boolean domain problem then becomes: given a set of trajectories determine the Boolean formula that represents the transitions between regions. This is a typical problem in circuit design. In that case one deals with an optimization problem: the determination of a minimal Boolean formula (circuit) capable of yielding a binary number from another one. It is expected that a suitable ILP program could generate the appropriate Boolean formula when given a sequence of pairs of binary numbers.

4.3 The Inverse Boolean Approach Proposed by Somogyi

The members of the group lead by Roland Somogyi have published several interesting papers on the Boolean approach to modeling. Many of those papers are described and referenced in the recent book edited by Bower and Bolouri [1]. In the inverse approach proposed in REVEAL [14], Somogyi introduces the concept of entropy, which is essentially a way to speed up the determination of genes that could possibly interact with other genes.

Given a table such as that depicted in Figure 8 (b), one wishes to determine the corresponding graph, namely that of Figure 8 (a) . An exhaustive generate-and-test approach consists of searching for all Boolean formulas that would express the input-output behavior of the system being considered. By introducing the notion of entropy, Somogyi is able to make that search efficient in cases where a few genes are influenced by a relatively small number of other genes. Essentially, the entropy approach allows one to first consider a single gene that may interact with a given gene; then all pairs of genes, and so forth.

The reference [5] essentially enumerates the possible searches of genes that can influence other genes within the Boolean approach. One can view these efforts as worthy ones in analyzing and improving the efficiency of the inverse Boolean approach. Though Somogyi and his group have been able to deal with modeling problems involving 50 or more genes, much remains to be done by incorporating probabilistic techniques and making it possible to study the interaction of hundreds, and possibly thousands of genes.

5 The Qualitative Approach

With the above background material it becomes easy to explain the approach taken by de Jong and Page [8]. Their so-called qualitative approach (in the sense of Kuipers [13]) can be viewed as a generalization of the Boolean approach. Recall that in the latter one assumes that the Boolean formulas representing a transition between regions imply that the n-digit binary numbers involved are changed simultaneously. De Jong and Page assume that each variable representing a gene changes according to a step function simulating a sigmoid function. They also assume that the change of value occurs at a specified position θ (see Figure 1).

In the deJong-Page approach the regions amount to n-dimensional mini-cubes whose edge sizes are determined by the various θ's. Figure 10 shows a view of the regions involved in the case of 2 variables. As in the example of Figure 10, a trajectory is determined by applying the known slopes (associated to each mini-cube). A slope in that case is defined by qualitative values $q\dot{x}_i$, which may take the values $q\dot{x}_i > 0$, $q\dot{x}_i < 0$, or $q\dot{x}_i \gtreqless 0$, meaning that the qualitative value qx_i is likely to increase, decrease or to be the same; where qx_i represents a set of constraints specifying a given mini-cube.

My student Tri Nguyen-Huu developed a prototype logic program – using the paradigm proposed in this paper – that, given a set of differential equations, discretizes them according to the given step functions (representing the sigmoids, each with its own θ), and then constructs the set of applicable regions and slopes. In that case each region may be associated with more than one slope. The program then performs a (*don't know*) non-deterministic search capable of determining all trajectories emanating from a given initial point.

The program in Figure 9 can be viewed as having been generated by the Nguyen-Huu's program when given the equations and graph in Figure 10 (b) and (c). The program in Figure 9 has the same general structure as that of the programs in Figures 4 and 6 used to describe the motivating example and the Boolean approach. Assuming the trajectory depicted by the dashed line in Figure 10 (a), one would obtain the curves in Figure 11.

In Appendix B we provide another example of the direct qualitative approach by solving the (continuous) problem depicted in Figure 2.

Nguyen-Huu's program is presently non-invertible but we believe that it could be perfected to have that property. Our current goal is to attempt to use it as a tool for helping biologists check if a given set of data is compatible with a hypothesized labeled graph. As a matter of fact this is the approach presently used by de Jong and Page as well as other authors (Gifford [24]).

5.1 The Inverse Qualitative Approach

Consider the differential equation for a variable x:

$$\dot{x} = k - \gamma x$$

A discrete solution means that

$$\frac{x(t+\delta t)-x(t)}{\delta t} = k - \gamma x$$

$$k = \frac{1}{\delta t}x(t+\delta t) + (\gamma - \frac{1}{\delta t})x(t)$$

We first remark the importance of the following property of the solution of the direct qualitative problem:

If one computes the above values of k for a trajectory, then each minicube contains points exhibiting the same value of k.

This is a key property in outlining a possible solution of the inverse qualitative problem. We start by estimating the maximum number of different values of k we can obtain assuming that the deJong-Page approach was used to solve the direct problem. Obviously, one also knows are many genes are involved, say n.

Let us consider the case of two genes and therefore two dimensions. Assuming that the direct qualitative approach has been used, then there are $2^2 = 4$ possible regions in the solution space and there are therefore 4 different values of k. This means that we can label the points in a trajectory constituting data available in the indirect qualitative approach according to those 4 values.

Consider the first gene, and determine the corresponding θ's so that each of the 4 regions contains the same value of k. That enables the determination of 2 values for θ. Call M_1 the defined regions corresponding to the θ's associated with the first gene. One then performs a similar operation considering the second gene and thus determines M_2. The superposition of M_1 and M_2 yields the desired results. This is so because the direct application of the qualitative approach would result in all the regions defined in the superposition of M_1 and M_2.

In the general case of r edges in the input graph converging to a given node i, one would have 2^r possible values for k. Then one would proceed as above by determining the set of regions M enabling the determination of θ's for each gene. The superposition of all such M's yields the desired results.

This is obviously a highly complex set of computations but they outline what is needed to solve the inverse problem using the qualitative approach. This suggests that there may well be approximate less complex solutions to the problem.

6 Comparison Between the Boolean and Qualitative Approaches

A few remarks are in order when comparing the Boolean approach with the qualitative approach. In the Boolean approach there are no specified θ's defining the time at which there is a change in a Boolean value. In contrast in the qualitative approach time intervals are increased according to the various values of θ that define the faces of the mini-cubes.

A trajectory, in the sense taken in this paper, consists of a traversal in the space that constitutes the minicubes. Trajectories are governed by the current minicube and the possible slopes that it specifies.

Let R_i be the i^{th} region and $Neighbor(R_i)$ be the set of its neighbors. In the qualitative approach the concept of neighboring minicubes is well defined and transitions in a trajectory always take place from neighbor to neighbor.

This may not be the case in the Boolean approach, since one has to clarify the notion of neighborhood. In reference [10] a region is considered as neighboring another one when their binary representations differ by a single digit. This amounts to defining neighborhood in the spatial sense of a torus. Two regions are neighbors if they are adjacent to each other in the torus. Figure 8 (d) depicts the adjacency of the various regions corresponding to the example provided in Figure 8.

7 The General Paradigm Expressed in a Constraint Language

The examples in Sections 3,4 and 5 allow us to generalize the proposed approach using a constraint logic program description. This is depicted in Figure 12.

It seems that it would not be extremely difficult to generalize the direct qualitative approach to consider other shapes of sigmoids that would be approximated by piecewise linear segments. In that case one would have to generate a set of regions, slopes and transitions that would constitute the foundation for the general program computing trajectories as described in Figure 12.

It should be noticed that the programs previously described in this work essentially simulate a finite-state-like automaton. It is convenient to select the set of regions as its terminal vocabulary, as well as its the set of states. One basically has to determine the constraints specifying regions (corresponding to states) and slopes (corresponding to the transitions).

One can then view the inverse problem as the one of synthesizing finite state automata from given data. That synthesis has been used quite often in the past and it represents an alternative to machine learning *à la* ILP. There are Web sites that offer programs capable of performing the synthesis of automata when given thousands of strings [25].

A few words about those programs are in order. First it is important to sketch the functioning of those synthesizers. The given input strings are first represented in the form of a tree that constitutes the automaton accepting exactly the set of input strings, and no other strings. The synthesizer then attempts to merge pairs, then triples, etc., of nodes in the tree therefore introducing loops that are essential for obtaining more general automata. In so doing it is of paramount importance to have a good set of negative examples, otherwise the generated automata is useless since it may consist of a single state. The automata synthesizer approach should be given consideration in future work.

8 Soft Computing Approaches

In recent years the term "soft computing" has been used to denote a number of AI approaches that are tolerant of uncertainty and imprecision. Presently, the main components of soft computing (SC) are fuzzy logic (FL), neural networks (NN) and probabilistic reasoning (PR). The latter subsumes belief networks, genetic algorithms, and parts of learning theory. Support Vector Machines (SVM) and Rough Sets (RS) are also included in SC since their objectives are to probabilistically classify imprecise information. (A textbook on SC has just been published [4].)

In this section we briefly describe three of the SC approaches that have been used in solving the probabilistic inverse problem: given noisy data representing micro-array experiments attempt to determine the regulation graph or some of its parts.

8.1 Recurrent Neural Nets

A highly recommended monograph describing this type of approach is the recent dissertation of Patrik D'haeseleer (see [2]). His work is based on variants the differential equations described in Section 2.

The recurrent neural net proposed by D'haeseleer has three components. The first represents a sum of various functions h^+ and h^- in the differential equations described in Section 2. The second component simulates the sigmoid functions as depicted in Figure 1. Finally, the third component simulates the decaying terms expressed by $-\gamma_i x_i$. The algorithms used by D'haeseleer are based on Pearlmutter's work on recurrent NN [16].

D'haeseleer's approach is said to be additive in the sense that only sums and not products of the h's are allowed. His dissertation provides results attesting the effectiveness of the recurrent NN approach and comparing it with other methods including Boolean and SVM's. According to the classification suggested in Section 1, the NN approach is *Inverse-Probabilistic-Continuous.*

8.2 Rough Sets

Rough sets (RS) and fuzzy sets (FS) are generalizations of classical sets. Fuzzy sets allow partial set memberships to handle vagueness, while rough sets allow multiple set memberships to deal with indiscernibility (see [29]).

A recent paper by the bio-informatics group at Norwegian University of Science and Technology in Trondheim, describes an interesting RS approach related to the inverse problem (see [12]). The aim of the research is to generate a set of rules needed to predict gene function based on available ontologies. These are tree structures defining categories and subcategories of known gene functions. Biologists are now developing databases containing ontologies that are aimed at cataloguing gene functions common to several species.

The problem of predicting a yet unknown gene function is both important and difficult; it is related to establishing metabolic pathways and regulation graphs (see Appendix A) . The proposed approach consists of inspecting both the ontology tree and the micro-array data. One performs a bottom-up tree traversal to determine the (lowest) node reflecting the expected function of a gene as governed by the micro-array experiment.

The micro-array data curves are first discretized by defining various qualitative levels of gene expression (e.g., none, low, high, highest, etc).

The generated rules contain Boolean connectors involving those levels and are used to classify the probable function of each gene according to the ontology.

In a sense the RS approach is not dissimilar to the one used in generating formulas expressing the inverse Boolean problem. Both approaches attempt to generate Boolean formulas expressing gene interactions. The rough set approach can be classified as *Inverse-Probabilistic-Discrete* (see Section 1).

8.3 Bayesian Networks and SVM's

By using Bayesian networks one attempts to determine a DAG that reflects the probabilistic interactions among genes when provided with the micro-array data. Recent work using Bayesian networks is described in [9] A present disadvantage of the Bayesian approach is that only DAG's are considered and it is known that in actual cell regulation the envisaged graphs often contain loops.

The approach suggested by Gifford [24] is also Bayesian. He does not attempt to generate a graph or DAG establishing gene interactions. He assumes that a competent biologist would propose a hypothetic gene interaction graph. Gifford's algorithms then determine the likelihood that the hypothesized graph produces the given micro-array data.

Support Vector Machines can be viewed as classifiers of n-dimensional data. They divide the data space into regions that are bounded by hyperplanes. SVM's are essentially special clustering algorithms. In Section 5.1 we mentioned the importance of clustering in solving the inverse qualitative problem. Reference [7] describes the use of SVM's in interpreting micro-array data.

According to the classification presented in Section 1 the Bayesian and SVM's methods can be described as being *Inverse-Probabilistic-Continuous.*

9 Final Remarks

It has been shown in the previous sections that the concepts of regions, transitions, and trajectories were applicable to all the versions of the cell regulation problem considered in this paper. Starting from the original formulation of the direct continuous problem using systems of differential equations, it was shown that the Boolean and qualitative approaches could be described using the constraint logic programming (CLP) paradigm for expressing the relationships between regions, transitions and trajectories.

Under the proposed paradigm both the direct and inverse problems are describable by the same program. We posit that this conceptual development is important because: (1) it presents a unified view of the regulation problem and its possible solutions, and (2) it enables to develop efficient versions of the inverse problem (modeling).

The second point deserves some clarification. Let us consider as an example the case of the Fibonacci function defined by the CLP program for `Fibb(N, Result)`. When solving the inverse problem, i.e., given `Result`, determine `N`, one essentially solves a large system of linear equations. A careful examination of how those equations are solved by an inverse evaluation could lead to more efficient computations.

It was seen in Section 3 that the inverse problem of determining the regions from trajectories is non-linear even though the direct problem is defined by linear inequalities. This suggests the use of interval constraints for solving the inverse problem.

A study of the available literature reveals that clustering plays a key role in the development of solutions to the inverse regulation problem. This was evidenced in the case of the remarks made in Section 5.1 about the inverse qualitative problem. Support Vector Machines (SVM) are related to clustering and have been used in attempts to solve the modeling problem (see [7] and [20]).

The experience gained from the present approach motivates us to extend it to consider the probabilistic case. That is undoubtedly the type of approach needed to solve problems involving a very large number of genes whose regulation data contains experimental errors.

10 Acknowledgements

I wish to acknowledge Tri Nguyen-Huu for all his help in preparing this article. Tri prototyped in Prolog and in CLP the various programs described in this work. The availability of those languages enabled us to develop the programs in a matter of a few weeks.

References

General References (Books, Monographs, Dissertations)

[1] Bower, J.M., and Bolouri, H., editors , 2001. Computational Modeling of Genetic and Biochemical Networks, MIT Press.

[2] D'haeseleer, P. 2000. Reconstructing Gene Networks from Large Scale Gene Expression Data, Ph.D. dissertation, University of New Mexico, 2000, available at `http://www.cs.unm.edu/~patrik`.

[3] de Jong, H., 2000. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review, Report 4032, INRIA Alpes.

[4] Kecman, V., 2001. Learning and Soft Computing: Support Vector Machine, Neural Networks, and Fuzzy Logic Models, MIT Press.

Articles

[5] Akutsu, T., Miyano, S., and Kuhara, S. 2000. Algorithms for Identifying Boolean Networks and Related Biological Networks Based on Matrix Multiplication and Fingerprint Function, *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, Recomb 2000*, Tokyo, ACM Press.

[6] Asarin, A., Maler, O., and Pnueli, A. 1995. Reachability Analysis of Dynamical Systems having Piecewise Constant Derivatives, *Theoretical Computer Science* vol. 138, pp. 35-65. available at
`http://www-verimag.imag.fr//PEOPLE/Oded.Maler`

[7] Brown M. P. S., *et al*, 2000. Knowledge-based Analysis of Microarray Gene Expression Data Using Support Vector Machines, Proceedings of the National Academy of Sciences. vol. 97(1), pp. 262-267.

[8] de Jong, H., and Page, M. 2000. Qualitative Simulation of Large and Complex Genetic Regulatory Systems, *notes of the 14^{th} Int. Workshop on Qualitative Reasoning*, Morelia, Mexico.

[9] Friedman, N., Linial, M., Nachman, I., and Pe'er, D. 2000. Using Bayesian Networks to Analyze Expression Data, *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, Recomb 2000,* Tokyo, ACM Press.

[10] Glass, L., and Kauffman, S. 1973. The Logical Analysis of Continuous Non-Linear Biochemical Control Networks. *J. Theor. Biol.* vol. 39, pp. 103-129.

[11] Henzinger, T.A., 1995. Hybrid Automata and Finite Bisimulations, 1995, *Proceedings of the* 22^{nd} *International Colloquium on Automata, Languages and Programming (ICALP),* Lecture Notes in Computer Science no 944, Springer Verlag, pp.324-335.

[12] T.R. Hvidsten, T.R., J. Komorowski, J., Sandvik, A.K., Loegreid, A., 2001. Predicting Gene Function from Gene Expressions and Ontologies, *in Proceedings of the Pacific Symposium on Bio-Computing*, Hawaii, pp. 299-310, available from `http://smi-web.stanford.edu/projects/helix/psb01/` .

[13] Kuipers, B. 1994. *Qualitative Reasoning.* MIT Press.

[14] Liang, S., Fuhrman, S.Somogyi, R. 1998. REVEAL, a General Reverse Engineering Algorithm for Inference of Genetic Network Architectures. *Pacific Symposium on Biocomputing'98* vol. 3,pp. 18-29.

[15] Older, W., and Vellino, A., 1993. Constraint Arithmetic on Real Intervals. InF. Benhamou and A. Colmerauer, editors, Constraint Logic Programming: Selected Research, MIT Press.

[16] Pearlmutter,B.A., 1995. Gradient Calculations for Dynamic Recurrent Neural Networks: A Survey, *IEEE Transactions on Neural Networks*, vol. 6, no 5, pp. 1212-1228.

[17] Thieffry D., Thomas R., 1997. Qualitative analysis of gene networks. *Proceedings of the Pacific Symposium on Biocomputing'98*, pp. 77-88.

[18] Washio, T., Motoda, H., Niwa, Y., 2001. Discovering Admissible Simultaneous Equation Models from Observed Data, in L. De Raedt, P. Flach(eds) 12^{th} *European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence 2167 pp. 539-551.

Web sites

[19] URL for the Pacific Symposim on Biocomputing
`http://www-smi.stanford.edu/projects/helix/psb98/`

[20] Support Vector Machines (SVM) URL for basic references
`http://www.eleceng.adelaide.edu.au/Personal/hgchew/svm.html`

[21] Haussler M.,Brown, P., et al, Knowledge-based Analysis of Microarray Gene Expression Data Using Support Vector Machines, available at `http://www.support-vector.net/svm_bbl.html`

[22] Clustering:
`http://genome-www.stanford.edu/~sherlock/tutorial.html` (tutorial)
`http://cne.gmu.edu/modules/dau/stat/clustgalgs/clustgalgs_frm.html` (algorithms)
`http://www-genome.wi.mit.edu/MPR/GeneCluster/GeneCluster2.html` (program).

[23] Tutorial on Bayesian Networks
`http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html`

[24] Gifford, D., URL for publications and courses
`http://tesla.lcs.mit.edu/genechips`
and `http://tesla.lcs.mit.edu/6892`

[25] Synthesis of Finite-State Automata
`http://abbadingo.cs.unm.edu`
and `http://abbadingo.cs.unm.edu/dfa.html`

[26] Bibliography on Micro Arrays in Biology
`http://linkage.rockefeller.edu/wli/microarray/index.html`

[27] URL on Micro-array Data Analysis
`http://www.bioinformatics.duke.edu/microarray.htm`

[28] URL's on Hybrid Systems
`http://www-cad.eecs.berkeley.edu/~tah/`
`http://www.cis.upenn.edu/~alur/`

[29] URL on Rough Sets
`http://www-idss.cs.put.poznan.pl/research/rough_sets/`

Appendix A

A Brief Overview of Cell Regulation

The problem of cell regulation can be succinctly and informally explained as follows. DNA may be represented by a string over a vocabulary of four letters (A, C, G, T) representing a sequence of nucleotides. A gene is a substring of DNA that is recognized by the cell machinery as a set of instructions capable of producing a protein. Proteins are the biochemical components that make up a living cell and govern its functioning and survival.

For the purposes of this paper one can assume that, to each gene, there is a single corresponding generated protein. (In actuality, it has been found that a gene may produce several proteins due to a process called alternate splicing).

Let us consider the simplest case of two genes G_i and G_j producing two proteins P_i and P_j. It is known that a protein P_j can position itself around the DNA of gene G_i and prevent the production of protein P_i. It is said that P_j inhibits the production of P_i , or alternatively, that gene G_j inhibits gene G_i.

In other cases it is known that a protein P_i can influence positively the production of P_j because, for example, P_i is a component or contains subcomponents that are necessary for the cell machinery producing P_j. It is said that G_i reinforces G_j.

One can therefore express interactions among genes by a directed graph whose nodes represent genes and whose edges are labeled by the tokens "+" or "-". The former corresponds to a reinforcement of one gene on another, and the latter to an inhibitory effect between two genes.

It should be remarked that the processes of reinforcement and inhibition are usually temporary, in the sense that the effect of a gene on another degrades with time.

In micro-array experiments one can infer a measure of the capacity of a given gene to produce its corresponding protein(s), as time varies. For example, one may be interested in studying gene interactions in an organism subjected to lack of nutrients. In a single micro-array experiment one can determine at time t the *potential* of each of thousands of genes of producing compounds that influence the production of their corresponding proteins. The curves determined by multiple experiments reflect the variation in the capacity of producing proteins as time progresses. The problem that interests us is the determination of the graph G from the curves obtained through micro-array experiments.

A few remarks are in order to differentiate between the graph G representing interactions among genes and another graph M representing what biologists call "metabolic pathways". M offers much more detailed information than G in the sense that, in M, one expresses every single biochemical reaction taking place in the cell. These reactions use as components the compounds generated by previous reactions and the various proteins produced by the cell. M is considerably harder to obtain experimentally than G.

In an oversimplified way the availability of G is helpful in the elaboration of M. It is also the case that biologists feel that it is unnecessary to analyze the detailed information in M and are satisfied in determining which genes interact with a given gene being studied. In those cases G is more desirable than M.

Appendix B

A More Complex Example of the Qualitative Approach

In this appendix we present the results of the qualitative approach using constraints, for the specific example of the system of differential equations and interaction graph shown in Figure 2. That example is taken from DeJong and Page [8]. The interested reader is referred to their paper for further details about how regions, slopes, and successors are determined.

Notice that there is a single value of θ for the variable x_2 , and two values of θ for the variables x_1 and x_3. That implies that the range for x_1 consists of 2 regions and the ranges for x_2 and x_3 consist each of 3 regions. There are therefore $2 \times 3 \times 3 = 18$ mini-cubes making up the space for the variables x_1, x_2, and x_3.

Each mini-cube corresponds to a region. The programs developed by Tri Nguyen-Huu are capable of reading the three differential equations of the problem in Figure 2 and determining the 18 regions and the corresponding slopes, as explained in Section 5. In the material that follows we have given specific numerical values for the parameters of the differential equations. They are:

$$
\begin{aligned}
dx_1/dt &= 30\ h^{+}(x_2, 2.5) - 5\ x_1 \\
dx_2/dt &= 24\ h^{-}(x_1, 3.5).h^{-}(x_3, 4.5) - 6\ x_2 \\
dx_3/dt &= 16\ h^{-}(x_1, 5.5) + 20\ h^{+}(x_3, 6.5) - 4\ x_3
\end{aligned}
$$

The eighteen regions and slopes (similar to those presented in the program of Figure 9) are shown in Figure 13. The nucleus of the program using the data in Figure 13 is identical to the one provided in Figure 9.

Figure 14 depicts all possible trajectories that can be traced starting from the mini-cube defined by the constraints in the box on the top right corner of that figure. That box specifies the starting point of the simulation. The identifiers QV_i in each box correspond to the Qualitative Values of the variables x_i. The identifiers QVP_i correspond to the derivatives of each variable. A QVP_i value of 1 corresponds to an increase in the value of the variable; when $QVP_i = -1$ there is a decrease in value; finally when $QVP_i = 0$, the value of QV_i remains unchanged. The reader should notice the non-deterministic nature of the graph in Figure 14. It represents all possible trajectories emanating from a given starting point.

```
region((1,X,Y)):- Y>=0,Y>=X.    region((3,X,Y)):- Y=<0,Y>=X.
region((2,X,Y)):- Y>=0,Y=<X.    region((4,X,Y)):- Y=<0,Y=<X.

slope(1,X,Y):- X = 2 * Y.      slope(3,X,Y):- X = 0,  Y > 0.
slope(2,X,Y):- X = 0,  Y < 0.  slope(4,X,Y):- X = -4 * Y.

trajectory(N,Region,[Region]):- N = 0.
trajectory(N,Start_region,[Start_region|Result]):-
              N > 0,
              region(Start_region),
              transition(Start_region, Next_region),
              region(Next_region),
              N2 = N - 1,
              trajectory(N2,Next_region,Result).

% transition determines the new point of coordinates (X2,Y2)
% which is the intersection of the boundary common to the
% regions N1 and N2 with the straight line of slope(N1,_,_)

transition((N1,X1,Y1),(N2,X2,Y2)):-
              ...

% Note for the particular case in question (Figure 3),
% transition could be written as

transition((R1,X1,Y1),(R2,X2,Y2)):-
              R2 \= R1,
              Delta_X = X2 - X1,
              Delta_Y = Y2 - Y1,
              slope(R1,Delta_X,Delta_Y),
              region((R1,X2,Y2)).
```

Figure 4: Nucleus of a constraint logic program simulating the example in Figure 3.

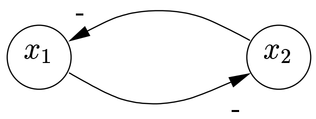


Figure 5: Graph depicting the interaction between two genes.

```
region([X1,X2]):- X1=0,X2=1.   region([X1,X2]):- X1=1,X2=1.
region([X1,X2]):- X1=0,X2=0.   region([X1,X2]):- X1=1,X2=0.

slope([X1,X2],[NX1,NX2]):- NX1 = not(X2), NX2 = not(X1).

trajectory(N,Region,[Region]):- N = 0.
trajectory(N,Start_region,[Start_region|Result]):-
              N > 0,
              region(Start_region),
              transition(Start_region, Next_region),
              region(Next_region),
              N2 = N - 1,
              trajectory(N2,Next_region,Result).

% transition determines the region R2 that can be reached
% from region R1.

transition(R1,R2):- slope(R1,R2).
```

Figure 6: Constraint program for the Boolean approach.

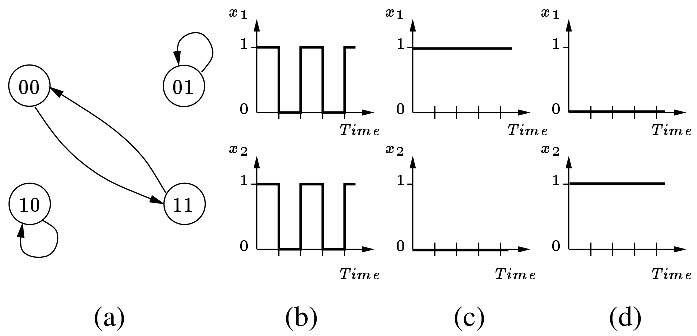
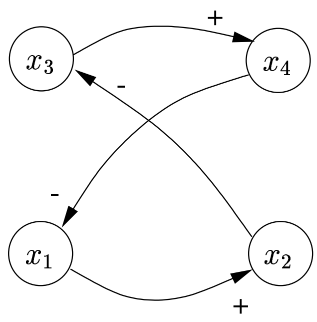


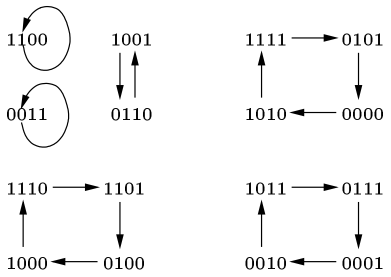
Figure 7: (a) A direct graph depicting all trajectories starting from each region and variations of variables x_1 and x_2 with time, for an initial state $x_1x_2 = 11$ (b), 10 (c), 01 (d) .



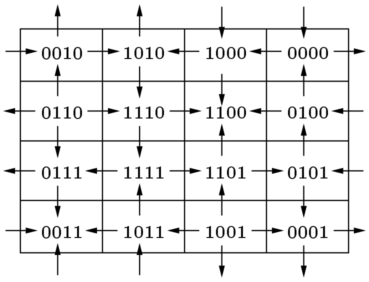
(a)

x_1	x_2	x_3	$x_4\,(t)$	x_1	x_2	x_3	$x_4\,(t+1)$
1	1	1	1	0	1	0	1
1	1	1	0	1	1	0	1
1	1	0	1	0	1	0	0
1	1	0	0	1	1	0	0
1	0	1	1	0	1	1	1
1	0	1	0	1	1	1	1
1	0	0	1	0	1	1	0
1	0	0	0	1	1	1	0
0	1	1	1	0	0	0	1
0	1	1	0	1	0	0	1
0	1	0	1	0	0	0	0
0	1	0	0	1	0	0	0
0	0	1	1	0	0	1	1
0	0	1	0	1	0	1	0
0	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0

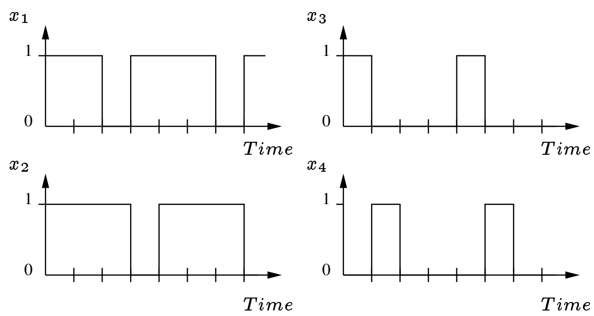
(b)



(c)



(d)

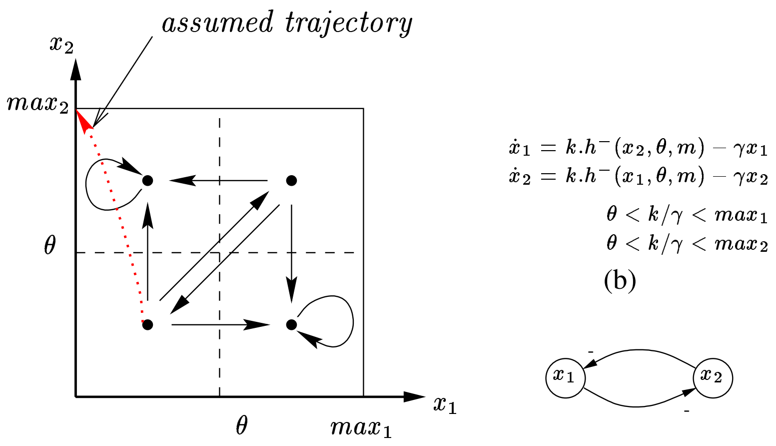


(e)

Figure 8: (a) Graph depicting the interaction between four genes and (b) the associated table of transition. (c) A direct graph depicting all trajectories and (d) the mapping of those trajectories on a torus. (e) Curves showing the behavior of the different genes with the initial state 1110.

```
region((1,X,Y)):- X>=0, X=<Theta1, Y>=0, Y=<Theta2.
region((2,X,Y)):- X>=Theta1, X=<max1, Y>=0, Y=<Theta2.
region((3,X,Y)):- X>=0, X=<Theta1, Y>=Theta2, Y=<max2.
region((4,X,Y)):- X>=Theta1, X=<max1, Y>=Theta2, Y=<max2.

slope(1,SX,SY):- SX = 1,  SY = 1.
slope(2,SX,SY):- SX = 0,  SY = 0.
slope(3,SX,SY):- SX = 0,  SY = 0.
slope(4,SX,SY):- SX = -1,  SY = -1.

trajectory(N,Region,[Region]):- N = 0.
trajectory(N,Start_region,[Start_region|Result]):-
           N > 0,
           region(Start_region),
           transition(Start_region, Next_region),
           region(Next_region),
           N2 = N - 1,
           trajectory(N2,Next_region,Result).

transition((R1,X1,Y1),(R2,X2,Y2)):-
           slope(R1,SX1,SY1),
           slope(R2,SX2,SY2),
           successor((R1,X1,Y1),(R2,X2,Y2),SX1,SY1,SX2,SY2).

% the following predicate tests if two regions R1 and R2 are
% adjacent, and that R2 can be reached from R1 according to
% their corresponding slopes.

successor((R1,X1,Y1),(R2,X2,Y2),SX1,SY1,SX2,SY2):-
              ...
```

Figure 9: Constraint program for the qualitative approach.



(a) (c)

Figure 10: (a) Mini-cubes corresponding to equations in (b) and the interaction graph in (c). The arrows are the slopes associated to each mini-cube.

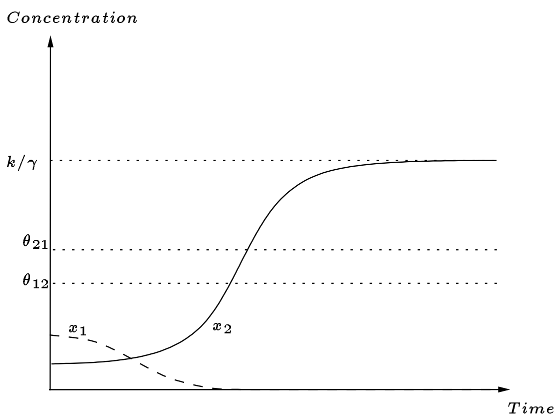


Figure 11: Possible curves obtained for Figure 10.

$$region_i(\boldsymbol{x}) \leftarrow region\_constraint_i(\boldsymbol{x}).$$

$$slope_i(\boldsymbol{x}, \boldsymbol{x}\text{'}) \leftarrow slope\_constraint_i(\boldsymbol{x}, \boldsymbol{x}\text{'}).$$

$$transition(region_i(\boldsymbol{x}), region_j(\boldsymbol{x}\text{'})) \leftarrow slope_i(\boldsymbol{x}, \boldsymbol{x}\text{'}),$$

$$\begin{aligned}
&trajectory(n, region_i(\boldsymbol{x}), list(region_i(\boldsymbol{x}))) \leftarrow n = 0.\\
&trajectory(n, region_i(\boldsymbol{x}), list(region_j(\boldsymbol{x}), result)) \leftarrow\\
&\qquad n > 0,\\
&\qquad transition(region_i(\boldsymbol{x}), region_j(\boldsymbol{x}\text{'})),\\
&\qquad trajectory(n-1, region_i(\boldsymbol{x}\text{'}), list((region_j(\boldsymbol{x}\text{'}), result)).
\end{aligned}$$

Figure 12: General structure of a constraint program traversing regions using transitions ($\boldsymbol{x}$ and $\boldsymbol{x}$' are vectors, i, j and m are integers).

```
region((1,X1,X2,X3)):- X1>0.0, X1<3.5, X2>0.0, X2<2.5, X3>0.0, X3<4.5.
region((2,X1,X2,X3)):- X1>0.0, X1<3.5, X2>0.0, X2<2.5, X3>4.5, X3<6.5.
region((3,X1,X2,X3)):- X1>0.0, X1<3.5, X2>0.0, X2<2.5, X3>6.5, X3<9.0.
region((4,X1,X2,X3)):- X1>0.0, X1<3.5, X2>2.5, X2<4.0, X3>0.0, X3<4.5.
region((5,X1,X2,X3)):- X1>0.0, X1<3.5, X2>2.5, X2<4.0, X3>4.5, X3<6.5.
region((6,X1,X2,X3)):- X1>0.0, X1<3.5, X2>2.5, X2<4.0, X3>6.5, X3<9.0.

region((7,X1,X2,X3)):- X1>3.5 ,X1<5.5 ,X2>0.0 ,X2<2.5 ,X3>0.0 ,X3<4.5.
region((8,X1,X2,X3)):- X1>3.5, X1<5.5, X2>0.0, X2<2.5, X3>4.5, X3<6.5.
region((9,X1,X2,X3)):- X1>3.5, X1<5.5, X2>0.0, X2<2.5, X3>6.5, X3<9.0.
region((10,X1,X2,X3)):- X1>3.5, X1<5.5, X2>2.5, X2<4.0, X3>0.0, X3<4.5.
region((11,X1,X2,X3)):- X1>3.5, X1<5.5, X2>2.5, X2<4.0, X3>4.5, X3<6.5.
region((12,X1,X2,X3)):- X1>3.5, X1<5.5, X2>2.5, X2<4.0, X3>6.5, X3<9.0.

region((13,X1,X2,X3)):- X1>5.5, X1<6.0, X2>0.0, X2<2.5, X3>0.0, X3<4.5.
region((14,X1,X2,X3)):- X1>5.5, X1<6.0, X2>0.0, X2<2.5, X3>4.5, X3<6.5.
region((15,X1,X2,X3)):- X1>5.5, X1<6.0, X2>0.0, X2<2.5, X3>6.5, X3<9.0.
region((16,X1,X2,X3)):- X1>5.5, X1<6.0, X2>2.5, X2<4.0, X3>0.0, X3<4.5.
region((17,X1,X2,X3)):- X1>5.5, X1<6.0, X2>2.5, X2<4.0, X3>4.5, X3<6.5.
region((18,X1,X2,X3)):- X1>5.5, X1<6.0, X2>2.5, X2<4.0, X3>6.5, X3<9.0.

slope((1,SX1,SX2,SX3)):- SX1 = 0, SX2 = 1, SX3 = 0.
slope((2,SX1,SX2,SX3)):- SX1 = 0, SX2 = 0, SX3 = -1.
slope((3,SX1,SX2,SX3)):- SX1 = 0, SX2 = 0, SX3 = 0.
slope((4,SX1,SX2,SX3)):- SX1 = 1, SX2 = 0, SX3 = 0.
slope((5,SX1,SX2,SX3)):- SX1 = 1, SX2 = -1, SX3 = -1.
slope((6,SX1,SX2,SX3)):- SX1 = 1, SX2 = -1, SX3 = 0.
slope((7,SX1,SX2,SX3)):- SX1 = -1, SX2 = 0, SX3 = 0.
slope((8,SX1,SX2,SX3)):- SX1 = -1, SX2 = 0, SX3 = -1.
slope((9,SX1,SX2,SX3)):- SX1 = -1, SX2 = 0, SX3 = 0.
slope((10,SX1,SX2,SX3)):-SX1 = 1, SX2 = -1, SX3 = 0.
slope((11,SX1,SX2,SX3)):- SX1 = 1, SX2 = -1, SX3 = -1.
slope((12,SX1,SX2,SX3)):- SX1 = 1, SX2 = -1, SX3 = 0.
slope((13,SX1,SX2,SX3)):- SX1 = -1, SX2 = 0, SX3 = 0.
slope((14,SX1,SX2,SX3)):- SX1 = -1, SX2 = 0, SX3 = -1.
slope((15,SX1,SX2,SX3)):- SX1 = -1, SX2 = 0, SX3 = -1.
slope((16,SX1,SX2,SX3)):- SX1 = 0, SX2 = -1, SX3 = 0.
slope((17,SX1,SX2,SX3)):- SX1 = 0, SX2 = -1, SX3 = -1.
slope((18,SX1,SX2,SX3)):- SX1 = 0, SX2 = -1, SX3 = -1.
```

Figure 13: Regions and slopes for the example in Figure 2.

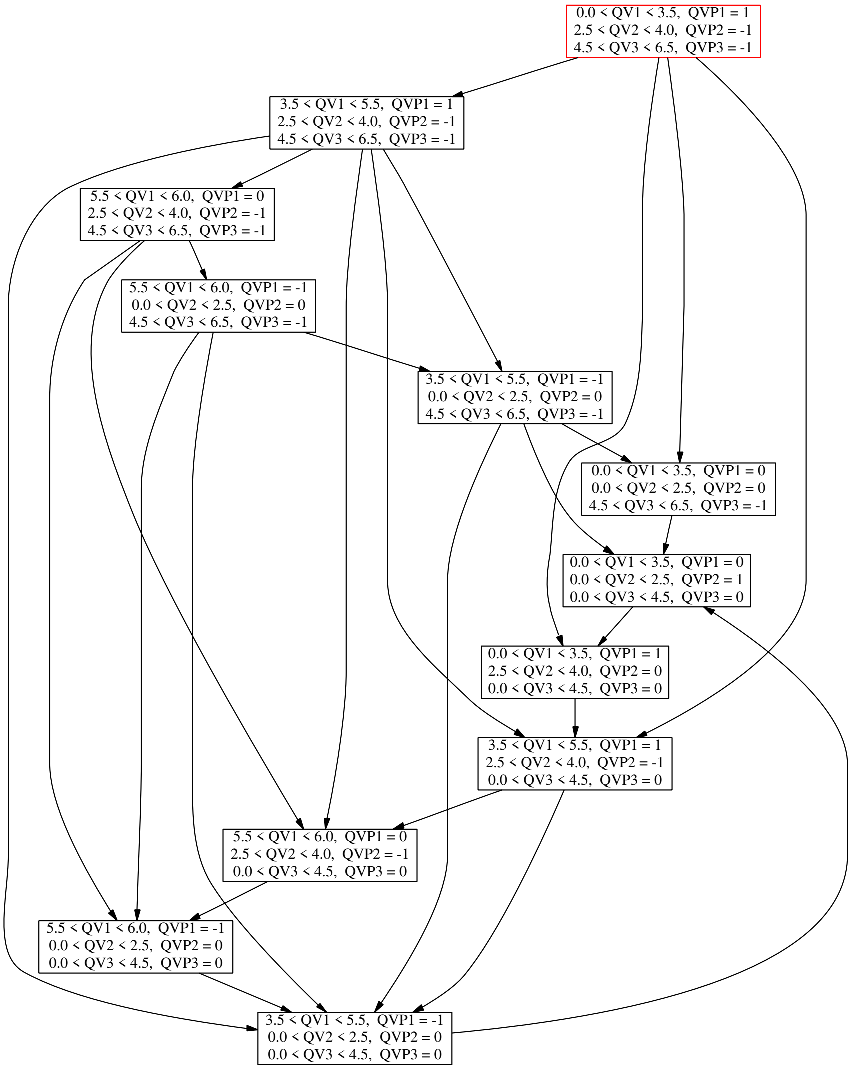


Figure 14: Possible trajectories emanating from the initial condition specified in the top rightmost box. The boxes specify (some) of the regions defined in Figure 13. The edges connecting the boxes represent the possible successors of each region; these successors correspond to those that appear in the program of Figure 9.