Journal of Machine Learning Research 9 (2008) 1475-1533 Submitted 5/06; Revised 10/07; Published 7/08

Incremental Identification of Qualitative Models of Biological Systems using Inductive Logic Programming

Ashwin Srinivasan[*] ASHWIN.SRINIVASAN@IN.IBM.COM
IBM India Research Laboratory
4-C, Institutional Area, Vasant Kunj Phase II
New Delhi 110 070, India

Ross D. King RDK@ABER.AC.UK
Department of Computer Science
University of Wales, Aberystwyth
Ceredigion, Wales, UK

Editor: Stefan Wrobel

Abstract

The use of computational models is increasingly expected to play an important role in predicting the behaviour of biological systems. Models are being sought at different scales of biological organisation namely: sub-cellular, cellular, tissue, organ, organism and ecosystem; with a view of identifying how different components are connected together, how they are controlled and how they behave when functioning as a system. Except for very simple biological processes, system identification from first principles can be extremely difficult. This has brought into focus automated techniques for constructing models using data of system behaviour. Such techniques face three principal issues: (1) The model representation language must be rich enough to capture system behaviour; (2) The system identification technique must be powerful enough to identify substantially complex models; and (3) There may not be sufficient data to obtain both the model's structure and precise estimates of all of its parameters. In this paper, we address these issues in the following ways: (1) Models are represented in an expressive subset of first-order logic. Specifically, they are expressed as logic programs; (2) System identification is done using techniques developed in Inductive Logic Programming (ILP). This allows the identification of first-order logic models from data. Specifically, we employ an incremental approach in which increasingly complex models are constructed from simpler ones using snapshots of system behaviour; and (3) We restrict ourselves to "qualitative" models. These are non-parametric: thus, usually less data are required than for identifying parametric quantitative models. A further advantage is that the data need not be precise numerical observations (instead, they are abstractions like positive, negative, zero, increasing, decreasing and so on). We describe incremental construction of qualitative models using a simple physical system and demonstrate its application to identification of models at four scales of biological organisation, namely: (a) a predator-prey model at the ecosystem level; (b) a model for the human lung at the organ level; (c) a model for regulation of glucose by insulin in the human body at the extra-cellular level; and (d) a model for the glycolysis metabolic pathway at the cellular level.

Keywords: ILP, qualitative system identification, biology

∗. Also at: Department of CSE & Centre for Health Informatics, University of New South Wales, Sydney.

1. Introduction

There is a general move in biology from seeking an understanding at the level of individual units (genes, proteins and so on) to an understanding at the system-level. Identifying single genes, proteins or metabolite levels cannot be expected to yield an answer to systemic behaviour any more than a list of radio parts could explain its behaviour (a point made in Lazebnik's humourous and perceptive article: Lazebnik, 2002). What is needed is an understanding of the function of each part and, crucially, how these components are connected, how they are controlled and the dynamic behaviour of the system as a whole. Biology, which for the last decade or so has been pre-occupied with establishing the "parts-list" is now moving to address these other issues. Besides the obvious scientific value of understanding whole systems, substantial benefits are expected to follow in clinical medicine. This is concerned with the application of computation and applied mathematics to improve existing pharmaceutical and medical practices. It is expected that results in systems-level biology will allow a better understanding of the nature of diseases, leading to a targeted design of new drugs and drug treatments. The importance of adopting a systemic approach to biology is not new: there are statements in Darwin's *Origin* that clearly anticipate this need. Its relevance in the modern biological context is summarised in a recent article in *Science* (W.Bialek and D.Botstein, 2004):

> The basic nature and goals of biological research is changing fundamentally. In the past, biological processes and the underlying genes, proteins, other molecules, and environmental factors were of necessity studied one by one in relative isolation. In contrast, today we are no longer satisfied with studies or answers that place each of these in a larger context. We now know that there are tens of thousands of genes encoded in the genome and that simple perturbations such as . . . heat shock, alter the expression of thousands of them . . . New goals are in sight, namely robust mathematical models and computer simulations that faithfully predict the behaviour of entire biological systems.

Some substantial research effort is being expended in trying to achieve these goals. The Physiome Project for example (see `http://www.physiome.org/`) lists its principal aim as being "to understand and describe the human organism, its physiology and pathophysiology quantitatively." This it hopes to achieve by using models at different levels (molecular to organ) that "include everything from diagrammatic schema suggesting relationships to fully quantitative computational models." Similarly, the United Kingdom's main research funding body in biology (the BBSRC) has invested over 15 million pounds in centres for integative systems biology: "the aim is to support research in such a way that all the components of the system under study can be researched at all relevant levels of biological organisation. It necessitates being able to to handle large experimental data sets and having the expertise and capacity to manipulate these and combine them with the theoretical base to develop new predictive and holistic models of how living systems function." (see *Bioscience for Society: A Ten Year Vision, January 2003* at `http://www.bbsrc.ac.uk/about/plans_reports/vision.html`).

In the physical sciences, the principal means of understanding complex systems has been through the use of mathematical models. This same approach is adopted in the field of mathematical biology. Following the pioneering work of Alan Turing (Turing, 1952) and Hodgkin and Huxley (1952) differential equations are now used to model a wide range of transport, reaction and conservation phenomena (Murray, 1993). However, while identification of models of physical systems can often

proceed from first principles (for example, balance equations, energy conservation and so on), the complexity of biological systems often force a much more experimental approach. The modeller selects those physical processes believed to be important, constructs a model and checks if solutions match the observed data. If not, the procedure is repeated until an adequate model is found. For example, a first attempt at modelling oxygen transport to red blood cells may consider a model that accounted for convection, diffusion and chemical reaction (these are the principal physical processes involved). In fact, convection makes a negligible contribution and reaction is only important for sick lungs. Once it is known that only the diffusion term is important, a parametric equation can be found relatively easily.

Broadly speaking, system identification can be viewed as "the field of modelling dynamic systems from experimental data" (Soderstrom and Stoica, 1989). We can distinguish here between: (a) classical system identification techniques, developed by control engineers and econometricians; and (b) machine learning techniques for system identification, developed by computer scientists. While the kinds of models identified by the two kinds of techniques are different, neither provides a foolproof method that can be employed without user interaction.

Classical system identification has concentrated on models largely constrained to be either ordinary differential equations (ODEs) or linear difference equations of some order. With this constraint on model structure, the input-output behaviour of the system is observed over a time interval and some statistical method is used to estimate parameters in the model. In its most general formulation, system identification proceeds by repeated estimation of both structure and parameters until an acceptable model is found. In practice, a small set of structures are given *a priori* and the procedure reduces to one of parameter estimation. Classical techniques have been used to identify linear time-invariant models for purposes of extracting control strategies (in engineering) or time-series predictions (in economics).

In this paper we are concerned instead with using machine learning techniques for system identification. Specifically, our interest is in methods that: (a) are not restricted to specific model structures; and (b) allow the incorporation of domain knowledge both to specify constraints on acceptable model structures and to direct the search through the space of acceptable structures. We believe both these features to be necessary in any empirical approach for identifying biological systems from data. Of the machine learning methods available that are capable of satisfying these requirements, those developed under the framework of Inductive Logic Programming (ILP) are amongst the most powerful. There are two reasons for this. First, the rich logic-based formalism used by ILP methods allows them to represent and identify a wide variety of relational descriptions. Second, ILP methods are unusual in that they make explicit provisions for the incorporation of domain knowledge to guide the model identification process. This includes mechanisms for the requirement in (b) above.

One question that is often raised in the context of ILP is that of efficiency. In the context here, this translates to asking if ILP methods are efficient enough to identify significantly complex biological models? As long as it is reasonable to identify such models in an incremental manner, we believe the answer to this question is "yes" and demonstrate this with the identification of four fairly complex systems at different scales of biological organisation (a predator-prey model at the ecosystem level; a model for the human lung at the organ level; a model for glucose regulation at the extra-cellular level; and the glycolysis metabolic pathway at the cellular level).

A second issue, unrelated to the use of ILP, but relevant to the empirical system identification task is the quantity and quality of data available. The identification of both the structure and parameter of quantitative models (like ODEs) requires a substantial amount of good quality numerical

data. While substantial amounts of quantitative data are being generated at some lower levels of biological organisation (a prominent example is provided by the use of DNA microarray data to estimate mRNA levels in a cell), quality is still variable: it is possible, for example to get very different expression profiles for the same tissue using different microarray technologies (Kuo et al., 2002). At higher levels (for example, at the organ or ecosystem level), data are sparse, although of perhaps better quality. In all cases, we believe it to be substantially easier and more reliable to obtain data that are of a qualitative nature. For example, it may be relatively easier to decide whether certain metabolites are present or absent in a cell, whether their levels have been increasing or decreasing and so on, rather than obtain precise measurements of the metabolites. In this paper, we will be concerned exclusively with system identification from such qualitative data. The resulting models are non-parametric: that is, parameter estimation is not required and data are only needed to identify the model structure. Clearly, these qualitative models cannot be treated as being equivalent to their quantitative counterparts. Nevertheless, they can be used to simulate possible system behaviours and may be much more understandable to a non-mathematical biologist than a quantitative model like a differential equation.

The rest of the paper is organised as follows. Section 2 describes an established approach to qualitative reasoning about dynamic systems. This involves the use of qualitative constraints which form the building blocks of qualitative models (these models include abstractions of ordinary differential equations). Section 3 describes informally the the basics of an ILP system used to identify qualitative models. This includes a variant that performs an incremental identification of increasingly complex models. Section 5.1 demonstrates this form of identification using a model physical system. The application to biological systems is in Section 6. Section 7 examines the automatic identification of stages for the incremental learner. Section 8 concludes the paper. Appendix A provides details of the ILP system used for incremental system identification. Appendix B provides details of the procedure for multi-stage decomposition.

2. Constraint-Based Qualitative Reasoning

Figure 1 (slightly modified from Bratko, 2001) shows four different qualitative abstractions of some numerical statements: (a) numbers are represented by intervals (marked by some distinguished values like *zero*,*end*, *inf* and so on); (b) derivatives are represented by directions of change (like *inc*); (c) functions are represented by monotonic relations (like `MPLUS` denoting "monotonically increasing"); and (d) entire sequences of behaviour are represented by qualitative statements that specify a qualitative values and directions of change.

Reasoning with qualitative abstractions requires a calculus: we propose to use the constraint-based formulation used in the qualitative simulation program QSIM (Kuipers, 1994) (here we provide an informal description along the lines described by Bratko 2001). In this, variables take qualitative values from *domains*. Domains are defined by a name and some ordered set of distinguished values called landmarks. For example, the variable *Amount* in Fig. 1 could be from the domain *level* with landmarks $minf, 0, inf$. A qualitative state of a variable is usually denoted by $Domain : QVal$ where $QVal$ is represented as a $\langle Qmag, Qdir \rangle$ pair, sometimes written $Qmag/Qdir$. $Qdir$ is the qualitative rate of change of the variable, which has a fixed, three-valued resolution (the three quantities being *inc*, for increasing; *dec*, for decreasing; and *std*, for steady). For example, the qualitative state of the variable *Amount* could be $level : 0...inf/inc$ (compare with (d) in Fig. 1).

	Quantitative Statement	Qualitative Statement
(a)	$Level(3.2\ s) = 2.6\ cm$	$Level(t1) = zero...inf$
(b)	$\frac{d}{dt}Level(3.2\ s) = 0.12\ m/s$	$\texttt{DERIV}(Level(t1)) = std$
(c)	$Amount = Level \times Level$	$\texttt{MPLUS}(Level, Amount)$
(d)	Time / Amount: 0.0 / 0.00; 0.1 / 0.02; ... / ...; ... / ...; 159.3 / 62.53	$Amount(zero...end) = zero...inf/inc$

Figure 1: Qualitative abstractions of numerical data (from Bratko, 2001).

The qualitative state of a system is simply a list of the qualitative states of the system's variables and a qualitative behaviour is a list of consecutive qualitative states.

Reasoning is accomplished using constraints. In this approach, pioneered by Kuipers (1994), there are four principal constraints: `ADD(A,B,C)`, for addition of qualitative variables `A` and `B` to give `C`;[1] `MULT(A,B,C)`, to denote $\texttt{A} \times \texttt{B} = \texttt{C}$; `MINUS(A,B)`, for sign inversion $\texttt{A} = -\texttt{B}$; and `DERIV(A,B)`, to denote that `B` is the a derivative of `A`. In this paper, we will also use `SUB(A,B,C)` to denote $\texttt{A} - \texttt{B} = \texttt{C}$. In addition to these, two functional constraints are also used: `MPLUS(A,B)`, to denote that when `A` increases then `B` increases as well; and `MMINUS(A,B)`, to denote that when `A` increases then `B` decreases. We will henceforth refer to these constraints as "the QSIM constraints".

Figure 2 shows a qualitative model for a simple physical system, expressed in terms of the QSIM constraints. A qualitative behaviour of this system—that is, a sequence of qualitative states of the system variables `La`, `Lb` and `Fab` that satisfy the model's constraints—is shown in Fig. 3

A number of advantages have been proposed for using qualitative models. First, in some cases they may be more appropriate than quantitative models. This is particularly so if quantitative measurements are either difficult to obtain or are noisy and what is of interest are the essential properties of the system. Second, the models are quite comprehensible. Both these features are particularly relevant to the modelling of biological systems. There is an additional advantage for automatic system identification of the kind we propose here. Since qualitative models are non-parametric all computational effort is focussed on identifying the model structure. This typically requires data of both less precision and quantity than that required for identification of quantitative models.

1. Constraints apply to the qualitative states of `A`, `B` and `C`. Recall that these are of the form $Domain : Qmag/Qdir$. Thus, the `ADD` constraint ensures that both magnitudes and directions of change are consistent. Thus $\texttt{ADD}(level : 0/inc, level : 0...inf/std, level : inf/inc)$ is true, but $\texttt{ADD}(level : 0...inf/inc, level : inf/std, level : 0...inf/inc)$ is not $(0...inf + inf \neq 0...inf)$. Similarly, $\texttt{ADD}(level : 0...inf/inc, level : 0...inf/std, level : 0...inf/inc)$ is true, but $\texttt{ADD}(level : 0...inf/inc, level : 0...inf/std, level : 0...inf/std)$ is not $(inc + std \neq std)$. It should also be apparent, that while quantitative addition is functional (the sum of a pair of numbers is a unique number) , qualitative addition ones is relational (that is, for a pair of qualitative states for `A` and `B`, `ADD` may be true for more than one qualitative state for `C`). Similar remarks apply to the other constraints.

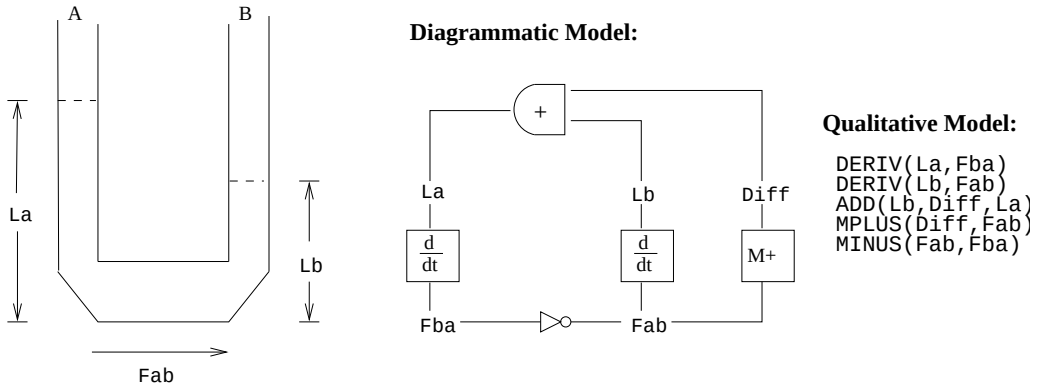


Figure 2: The U-tube and its qualitative model. There are three (measurable) system variables: the water-level in arm A (La); the water-level in arm B (Lb); and the flow of water from A to B (Fab). The diagrammatic model shows the system components involved (two differentiators, an adder, an inverter, and a monotonic function generator) and their interconnections. The qualitative model expresses the same information as a conjunction of constraints (here, we have used the QSIM constraints described in the paper).

La	Lb	Fab
$level : 0/std$	$level : 0/std$	$flow : 0/std$
$level : 0/inc$	$level : 0...inf/dec$	$flow : minf...0/inc$
$level : 0...inf/dec$	$level : 0/inc$	$flow : 0...inf/dec$
$level : 0...inf/dec$	$level : 0...inf/inc$	$flow : 0...inf/dec$
$level : 0...inf/std$	$level : 0...inf/std$	$flow : 0/std$
$level : 0...inf/inc$	$level : 0...inf/dec$	$flow : minf...0/inc$

Figure 3: A qualitative behaviour of the U-tube that is consistent with the qualitative model in Fig. 2. The rows are example states of the qualitative variables and have no implied ordering.

3. Model Identification using Inductive Logic Programming

Given correct definitions for the QSIM constraints, it is our aim in this paper to identify qualitative models such as that shown in Fig. 2, given qualitative states such as those shown in Fig. 3. Since the QSIM constraints are relational, automatic identification of such models clearly requires that the system identification method be able postulate and test relational models. Perhaps the most powerful framework for learning such models is that provided by Inductive Logic Programming (ILP, see Muggleton and Raedt, 1994). ILP is concerned with extracting models in an extremely expressive subset of first-order logic and reasonably efficient implementations have been developed.

To a good first approximation, the basic task addressed by an ILP system can be viewed as a discrete optimisation problem of finding the lowest cost elements amongst a finite set of alternatives. Many ILP systems solve this problem by employing a procedure that searches through a

directed acyclic graph representation of possible models. In this representation, a pair of models are connected in the graph if one can be transformed into another by an operation called "refinement". Figure 4 shows some parts of a graph for the U-tube in which a model is refined to another by the addition of a qualitative constraint. An optimal search procedure (branch-and-bound, for example) traverses this graph in some order, at all times keeping the cost of the best nodes so far. Whenever a node is reached where it is certain that it and all its descendents have a cost higher than that of the best nodes, then the node and its descendents are removed from the search. A portion of the search tree commencing at $\emptyset$ for one such search is shown in Fig. 5.

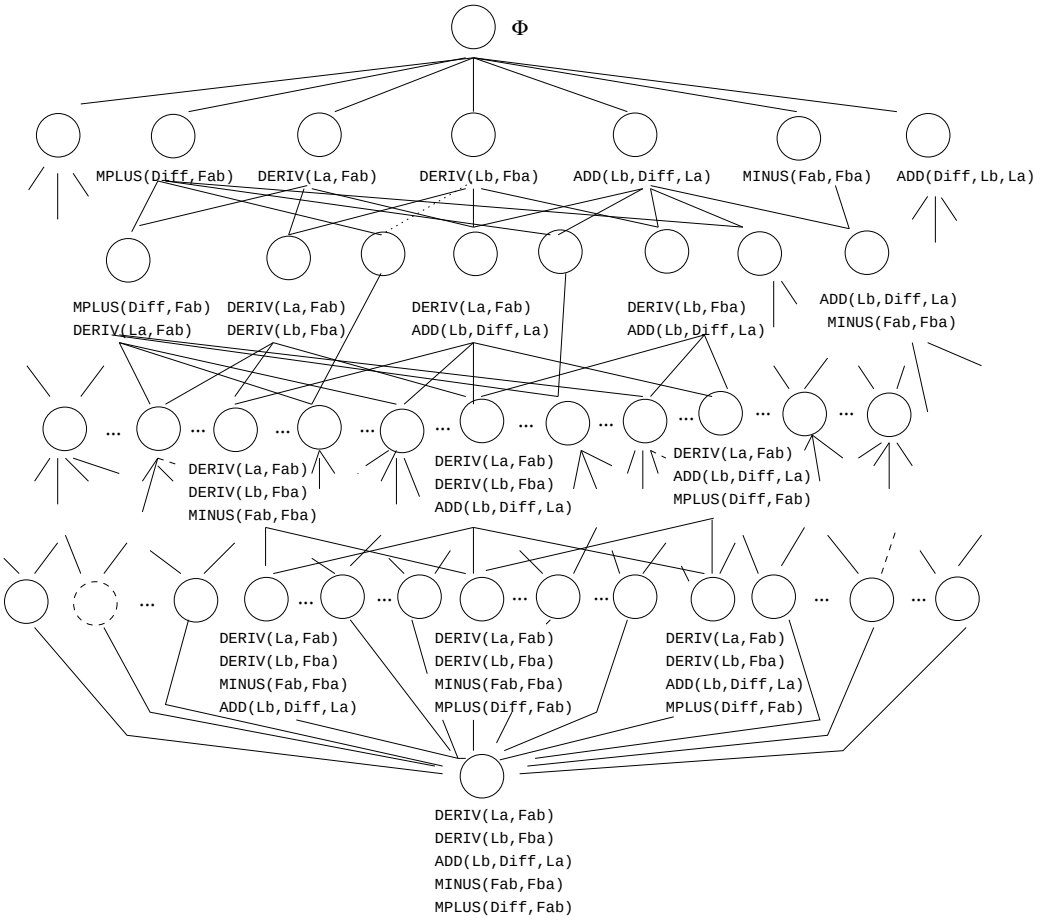


Figure 4: Portions of a refinement graph of models for the U-tube.

Enumerative procedures like branch-and-bound works best if the cost function is monotonic. That is, the score of each node in the search tree is at least as bad as all its descendents (this allows the nodes and its descendents to be removed from the search). The procedure is optimal in the sense that it is guaranteed to find the best solution(s). However in the worst case, it may require examining the entire search space.

Actually, there is more to an ILP system than search. The principal components of such systems are:

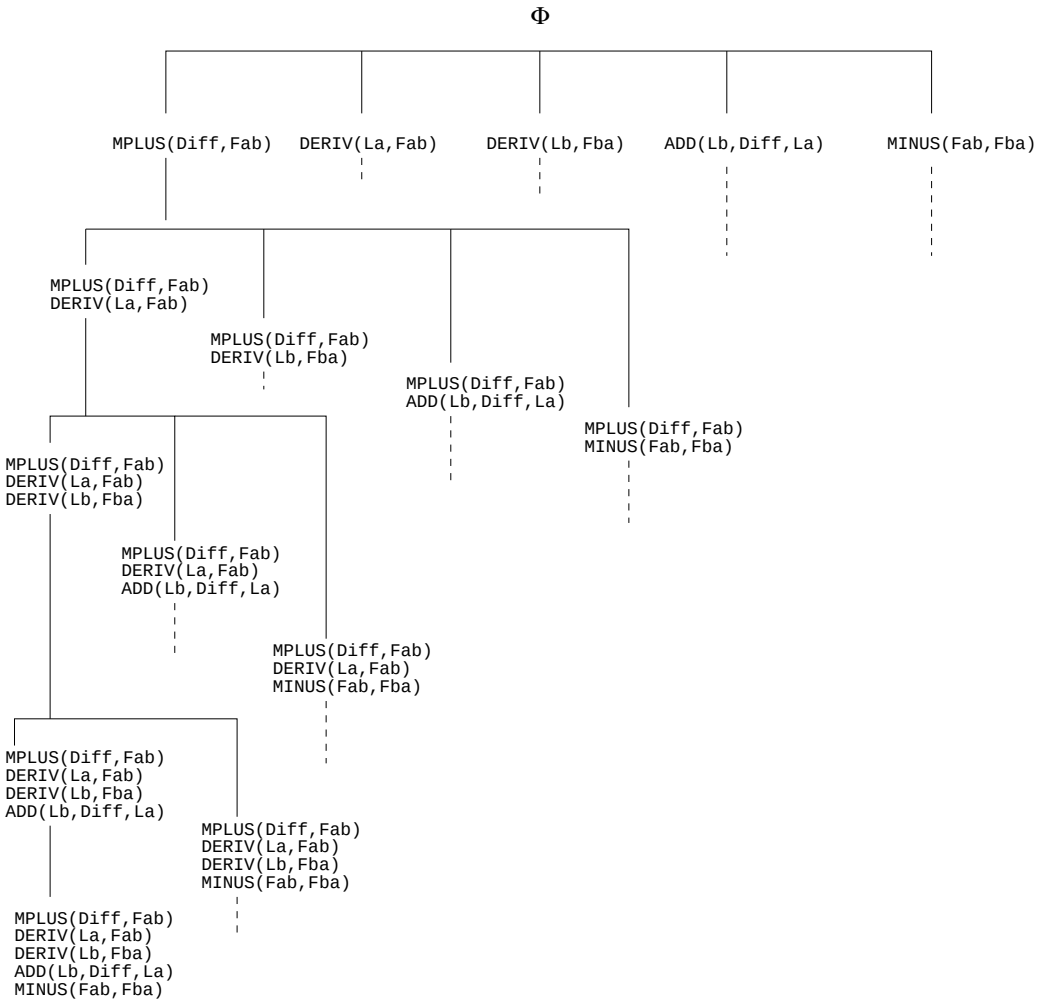


Figure 5: Portions of the search tree explored when searching for models for the U-tube. The search starts from $\emptyset$.

1. *Background knowledge B.* These are statements, usually written in some formal language that specify domain-specific information. We will include in this domain-specific constraints on the kinds of models that are acceptable (or unacceptable, if easier); and directions to the search procedure that allow the system to avoid useless search paths. Examples of these for qualitative model identification are:

   (a) Definitions for qualitative constraints like `DERIV`, `MPLUS`, `ADD` and so on, along with appropriate dimensionality checks *etc.* to ensure their correct usage.

   (b) A constraint specifying that models must not contain relations that are redundant. For example, the relation `ADD(Diff,Lb,La)` is redundant if the model already has `ADD(Lb,Diff,La)` (that is, `ADD` is commutative). The model must respect dimensional

constraints. This prevents relations like `ADD(Lb,Fab,La)` from appearing in the model (`Fab` being a flow has different units of measurement to the level `Lb`).

(c) A directive that the search need not examine models that explain below some proportion of the observed behaviours (more on "explain" in a moment).

2. *Examples E*. These are the observed data. For qualitative model identification, these would be qualitative observations of system behaviour of the form shown in Fig. 3. ILP systems also accept counter-examples of system behaviour. Since this is difficult to obtain for the problems we are concerned with, we do not pursue this further here. Given a set of examples, H is said to explain an observation e if H is consistent with B and e logically follows from B and H (see Appendix A for a precise mathematical formulation). For example, given correct definitions for the qualitative constraints `DERIV`, `MPLUS`, `ADD` and `MINUS` as background knowledge, the qualitative model described by the conjunction `DERIV(La,Fab)` $\wedge$ `DERIV(Lb,Fba)` $\wedge$ `ADD(Lb,Diff,La)` $\wedge$ `MPLUS(Diff,Fab)` $\wedge$ `MINUS(Fab,Fba)` is an explanation of the examples in Fig. 3.

3. *Refinement operator* ρ. This function defines the set of descendents for each node in the refinement graph. With most ILP systems, the set of descendents of a node are (minimal) generalisations or specialisations of the node. Roughly speaking, for qualitative models, generalisations correspond either: to removing one or more qualitative components from the diagrammatic model; or to "disconnecting" qualitative components from each other. Conversely, specialisations correspond to adding new components or connecting existing components.

4. *Cost function* f. This is a real-valued function for each node in the refinement graph. As mentioned earlier, monotonic cost functions are of some importance. A simple cost function satisfying this property in Fig. 4 is $f(H) = -P$, where P is the number of examples explained by model H. If every element H' of $\rho(H)$ contains at least one additional constraint, it can be shown that number of examples explained by H' (and recursively, all its descendents) would be at most P. It follows therefore that the cost of H is no worse than any of its descendents. In practice such a cost function is too simple to be of use (the search would trivially return the most general model), and modifications are made either to: (a) incorporate a trade-off between the explanatory power of the model and its complexity (Muggleton, 1996); or (b) include additional constraints in the background knowledge that prevent the selection of trivial models.

A description of an ILP implementation that uses these components can be found in Section A.2.

4. Identification of Qualitative Models

We refer the reader to Coghill et al. (2005) for an extended review of the literature on learning qualitative models. Briefly, Bratko and colleagues (Bratko et al., 1989; Mozetic, 1987) appear to have the been the first to use qualitative reasoning to build a static model for the electric activity of the heart. Coiera's GENMODEL (Coiera, 1989a,b) was the first machine learning system that constructed qualitative models for dynamic systems. A special-purpose ILP system, GENMODEL (and an updated version in Hau and Coiera, 1997) is restricted to finding qualitative relationships amongst the observed variables only (that is, no intermediate, or hidden, variables are hypothesised).

Model-identification systems that allowed intermediate variables were developed independently by Richards et al. (1992) and Bratko et al. (1992). Both use general-purpose ILP learners (although in different ways) and the principal advantages and shortcomings of these approaches and a later program called QSI (Say and Kuru, 1996) have been documented elsewhere (Coghill et al., 2005).

More recently, the QOPH system for identifying qualitative models exploited the possibility of providing the ILP system ALEPH with a special-purpose refinement operator (Coghill et al., 2002). This operator, with certain "built-in" constraints on acceptable qualitative models, is used by ALEPH to search the space of possible models. Extensive experiments are reported by Coghill et al. (2002) on the reconstruction of some model physical systems. While the results are promising, the scalability of the approach is unclear, since: (a) Model identification is assumed to be possible in a single step. Some simple complexity arguments (see Section A.2) suggest that the complexity of this task grows exponentially with the number of constraints in the model (this is the primary motivation for the incremental approach described in the next section); and (b) The special-purpose refinement operator is difficult to modify and its properties are difficult to analyse. Although not using a general-purpose ILP system, Suc and colleagues have proposed a hybrid approach of combining a logic-based qualitative learner followed by numeric modelling to construct quantitative models of systems (Suc et al., 2003). The approach, called Q^2-learning, first constructs "qualitative model trees". These are like decision trees, with monotonic QSIM constraints in the leaves. These constraints are then used to direct the construction of quantitative models (usually linear models). The success of this approach depends on the availability and quality of numeric data and the system being modelled by a composition of quantitative models (for example, like piecewise linear models).

The advantages of using of a purely qualitative representation for modelling metabolic pathways has been recently advocated (King et al., 2005). In that paper, a special-purpose system is used to generate possible models for the glycolysis pathway. The approach we propose here differs from that in two principal ways. First, it is a general approach as opposed to a specialised one for metabolic pathways. Second, the complexity of the implementation by King and his co-workers is of the same order as QOPH implementation. The incremental approach described in the next section will usually be significantly more efficient.

5. Incremental Model Identification with ILP

Modern ILP systems are largely "one-shot" model constructors. That is, given B, E, ρ and f, they attempt to identify models with the lowest cost in a single search. While this approach has been reasonably successful in the identification of small to medium-sized models (for example, qualitative models containing no more than 4 to 5 constraints), it is unclear whether the approach can scale up to the identification of substantially complex models. For example, the worst-case bound in Remark 2 in Section A.2 grows exponentially in the number of qualitative constraints in the model.

An obvious approach to control this increase in complexity is to decompose system identification into a series of stages, with the final model being some composition of models obtained at each stage. In this paper we use a simple incremental composition in which models identified at any stage are modified at the next stage. That is, one or more models are identified for the first stage (these are all the models consistent with the background knowledge and have the lowest cost). Each of these are then used to give models for the next stage and so on. This is easily achieved by starting the search at each stage at nodes in the refinement graph corresponding to the models found at the

previous stage. Formally the incremental learner is principally provided with: (a) an initial set of models H_0; (b) a sequence of pairs (B_i, E_i) corresponding to the background knowledge and examples for each stage i (the B_i and E_i do not all have to be different); (c) a general-purpose refinement operator ρ for all stages; and (d) a cost function f for all stages. The task of the incremental learner is to construct a set of models from this data (see Fig. 6 and Appendix A).

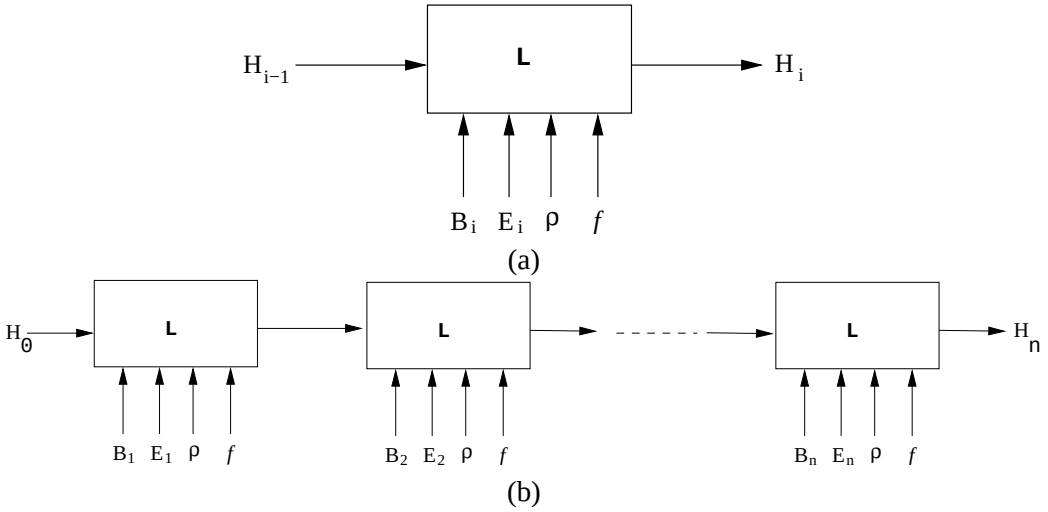


Figure 6: Incremental model identification with ILP. The basic element shown in (a) consists of an ILP learner L that takes as input a set of models, background knowledge, examples, a refinement operator ρ and a cost function f. In (b), this basic unit is repeatedly used to construct a model in n stages. "One-shot" model identification by normal ILP systems is a special case of this process, with $n = 1$ and $H_0 = \{\emptyset\}$ (here $\emptyset$ denotes the empty model).

The actual implementation in Section A.2 contains some additional aspects which are not shown in Fig. 6 (and similar figures in Section 6) for simplicity:

1. A refinement operator that performs both generalisations and specialisations can completely revise models found at a previous stage. However, this is computationally expensive. Instead, we use a refinement operator ρ_A that is restricted to performing specialisations only (for qualitative models, this amounts to adding qualitative constraints and connecting existing qualitative components). To correct partially for this shortcoming, models are subject to a limited generalisation before submission to any L. For qualitative models, this translates to retaining the qualitative components found at the previous stage but disconnecting some or all components, respecting any constraints provided on the usage of the components (in ILP terminology, this amounts to removing variable co-references, respecting any language constraints provided). This allows the incremental procedure to perform a particular kind of revision of models found at the previous stage;

2. Logically redundant models produced by any L are removed and a subset of the result is selected randomly;

3. A cost function f_{Bayes} described in Muggleton (1996) is used. For qualitative models, this performs a trade-off between the likelihood of a qualitative model and its complexity (a quantity related to the number of constraints in the model); and

4. An upper-bound is provided on the amount of search to be conducted by any L.

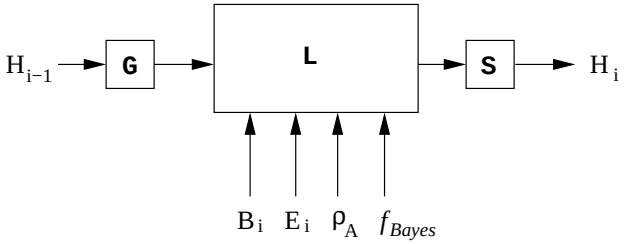


Figure 7: A more accurate representation of the implementation of the basic element used in this paper. Here G performs a limited generalisation of the input models, and can be eliminated for refinement operators that perform both generalisations and specialisations. S performs a random selection of the output models and can be eliminated if all models produced by L are sent to the next stage. ρ_A is a refinement operator that performs specialisations only; and f_{Bayes} is a Bayesian cost function. For simplicity, we will not show G and S in subsequent figures.

A more accurate representation of the basic element of the incremental learner, as implemented by the procedures in Section A.2, is shown in Fig. 7. With these implementation choices it can be shown that, for identification of qualitative models, the size of the search space of such an incremental procedure is dominated by maximum number of additional qualitative constraints that need to be identified at any one of the stages (see Section A.2 for the details). The savings over a non-incremental approach can be substantial, but two points are worthy of re-emphasis:

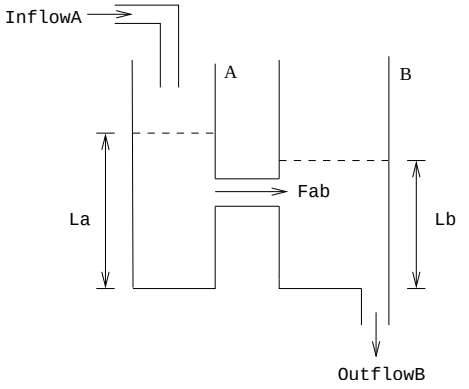
1. The incremental approach requires that a domain-specific decomposition into stages should be possible (by providing background knowledge and observational data for each stage); and

2. We can only guarantee correctness of the incremental approach to the extent that any model identified for a stage will logically entail the observations for that stage (given the background knowledge). The approach cannot, however, provably identify the lowest cost model in the search space. This follows naturally from the fact only lowest cost models are retained at the end of each stage: unless the cost function exhibits a form of monotonocity with stages (or we are simply constructing models in a single-stage), this is tantamount to using a greedy search, which is known to be sub-optimal.

These caveats aside, the incremental approach can be used either: (a) as a "single-shot" model constructor; or (b) to refine approximate models; or (c) to build increasingly larger models using sub-components of smaller ones. In the next section, we illustrate (c) using a model physical system.

5.1 Incremental Qualitative Model Identification: An Example

We consider identifying the qualitative model of the coupled-tanks system shown in Fig. 8. The measurable system variables are these: the input, `InflowA`, that pours into the top of tank A; the output, `OutflowB`, that pours out of the base of tank B; the flow of water from A to B, `Fab`; and the water-levels `La` and `Lb`.



Qualitative Model:

```
DERIV(La,NetflowA)
DERIV(Lb,NetflowB)
ADD(Lb,Diff,La)
MPLUS(Diff,Fab)
MPLUS(Lb,OutflowB)
ADD(NetflowB,OutflowB,Fab
ADD(Fab,NetflowA,InflowA)
```

Figure 8: A system comprised of two coupled tanks and its qualitative model.

The coupled tanks system consists of two tanks connected together. This allows us to decompose the identification of this model into two stages. In the first stage, we focus on identifying a model for tank B, using the single tank system in Fig. 10 (often called the "bathtub" system in qualitative modelling literature). Any consistent models identified for the single tank system are then extended to return final models for the coupled tanks system (see Fig. 9).

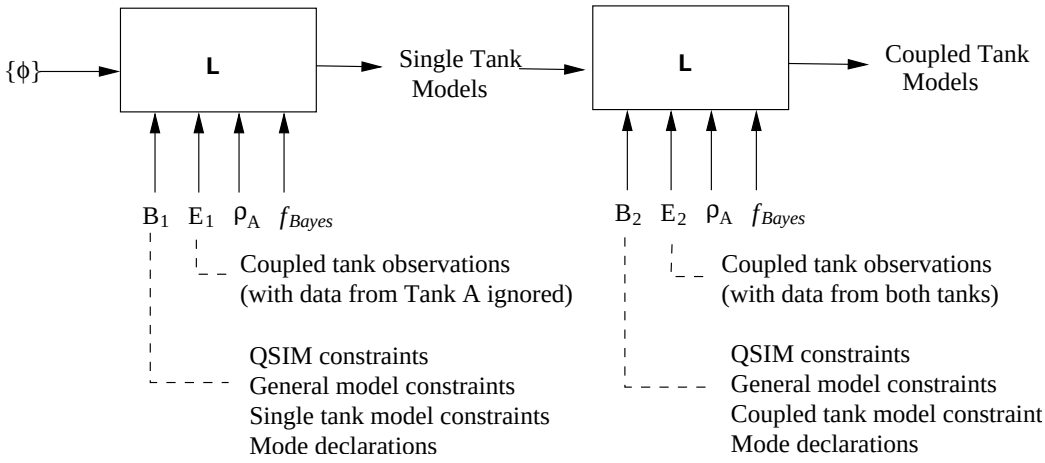
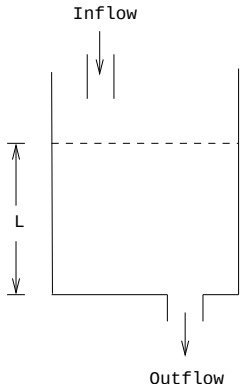


Figure 9: Incremental model identification for the coupled tanks system. Models are first identified by `L` for a single tank system. These are then refined by `L` to models for the coupled tank system. The term "mode declarations" is used in the sense described in Muggleton (1995) and refer to statements that provide domain and connectivity information for the qualitative variables.

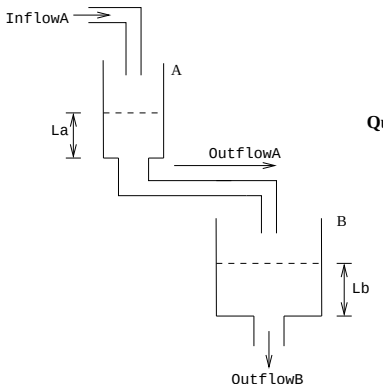
We note in passing that the final model for the coupled tanks is not simply a conjunction of two single tank models. This conjunction would not capture the fact that the flow from tank A to B is related to the difference in levels of fluid in A and B. The conjunction of the two models is, in fact, an appropriate model for the system of cascaded tanks shown in Fig. 11.



Qualitative Model:

```
DERIV(L,Netflow)
MPLUS(L,Outflow)
ADD(Outflow,Netflow,Inflo
```

Figure 10: A single tank system with an input and output. The system variables are `Inflow`, `Outlflow` and , `L`.



Qualitative Model:

```
DERIV(La,NetflowA)
MPLUS(La,Fab)
ADD(OutflowA,NetflowA,InflowA)
DERIV(Lb,NetflowB)
MPLUS(Lb,OutflowB)
ADD(NetflowB,OutflowB,OutflowA
```

Figure 11: A system comprised of two cascaded tanks. The qualitative model is simply a conjunction of two single tank models.

We elaborate further on the elements in Fig. 9:

1. *Background knowledge.* This is comprised of the following components:

   (a) Correct definitions of the QSIM constraints. Our definitions are based on those in Bratko (2001);

   (b) A set of general constraints on "well-posed" qualitative models. We describe these in more detail below;

(c) Stage-specific constraints on the models constructed. This consists of specifying the number of qualitative constraints in the final model for each stage. This is 3 for Stage 1 (the single tank model) and 7 for Stage 2 (the coupled tanks model); and

(d) Stage-specific "mode" declarations similar to the description in Muggleton (1995) that provide domain and connectivity information for the qualitative variables (see Fig. 12);

2. *Examples.* These are in the form of qualitative states for the system variables. Recall that for the coupled tanks system these are: `La`, `Lb`, `InflowA`, `Fab` and `OutflowB` (see Fig. 8). Clearly, flows and levels cannot be negative: we are further only interested in a system with a steady, non-negative input flow. That is, the only valid qualitative state for `InflowA` is $flow:0...inf/std$. `OutflowB`, on the other hand, can be any one of $flow:0/std$, $flow:0/inc$, $flow:0...inf/std$, $flow:0...inf/inc$, $flow:0...inf/dec$. The level of water `La` or `Lb` for the system can similarly assume any of the following qualitative states: $level:0/std$, $level:0/inc$, $level:0...inf/std$, $level:0...inf/inc$, $level:0...inf/dec$. Examples for Stage 1 ignore the values observed for levels and flows for Tank A (that is, `La` and `InflowA` are ignored: this can be easily specified using the mode declarations). Some observations for Stage 1 are shown in Fig. 13. Examples for Stage 2 contain the qualitative states of all the system variables.

3. *Refinement operator and cost function.* These are the operator ρ_A and f_{Bayes} described earlier.

5.2 General Constraints on "Well-posed" Models

In Coghill et al. (2005), the term "well-posed" qualitative models is used to denote those models that satisfy a number of domain-independent constraints. We use the the following constraints from that report:[2]

1. *Size.* A well-posed model must be of a particular size (measured by the number of qualitative constraints).

2. *Completeness.* The model must contain all the measured variables.

3. *Language.* The number of instances of any qualitative constraint in a well-posed model should be below some prescribed number.

4. *Sufficiency.* The model must adequately explain the observed data. By "adequate", we intend to acknowledge here that due to noise in the measurements, not all observations may be logical consequences of the model.[3] The percentage of observations that must be explainable in this sense is a user-defined value.

5. *Redundant.* The model must not contain relations that are redundant. For example, the relation `ADD(Inflow,Outflow,X)` is redundant if the model already has `ADD(Outflow,Inflow,X)`.

2. This list excludes two constraints from the report: the "Determinate" constraint can be effectively enforced by the "Size" constraint. The "Connected" constraint that requires all intermediate variables should appear in at least two qualitative constraints is enforced by the more general "Irrelevant variables" constraint here. All the constraints are assumed to be encoded in the background knowledge for any given stage.
3. Strictly speaking, the model in conjunction with the background knowledge.

Modes:

```
DERIV(+level,-flow)
ADD(+level,+level,-level)      ADD(+level,-level,+level)
ADD(+flow,+flow,-flow)         ADD(+flow,-flow,+flow)
MPLUS(+level,-level)           MPLUS(+level,-flow)
MPLUS(+flow,-flow)             MPLUS(+flow,-level)
MMINUS(+level,-level)          MMINUS(+level,-flow)
MMINUS(+flow,-flow)            MMINUS(+flow,-level)
MINUS(+level,+level)           MINUS(+flow,+flow)
```

A legal model:

```
MPLUS(L,Outflow)
DERIV(L,Netflow)
ADD(Outflow,Netflow,Inflow)
```

Two illegal models:

```
MPLUS(L,Outflow)
DERIV(L,Netflow)
ADD(Outflow,L,Inflow)
```

(ADD cannot add flows to levels)

```
MPLUS(L,Outflow)
ADD(Netflow,Outflow,Inflow)
DERIV(L,Netflow)
```

(ADD needs Netflow to be known)

Figure 12: Example "mode" declarations for the qualitative constraints. For example, the mode declaration `ADD(+level,+level,-level)` states that given values from domain "level" for the the first two arguments, `ADD` computes a value for the third argument (also from domain "level"). This is thus a simple form of dimensionality check. This prevents the ILP system from constructing model M2 (in which a variable from a "flow" domain is added to one from a "level" domain).

Fab	OutflowB	Lb
$flow : 0...inf/std$	$flow : 0/std$	$level : 0/inc$
$flow : 0...inf/std$	$flow : 0...inf/inc$	$level : 0...inf/inc$
$flow : 0...inf/std$	$flow : 0...inf/dec$	$level : 0...inf/dec$
$flow : 0...inf/std$	$flow : 0...inf/std$	$level : 0...inf/std$

Figure 13: Some example observations of the relevant system variables for identification of a single tank model. No ordering is implied amongst these observations.

6. *Contradictory.* The model must not contain relations that are contradictory given other relations present in the model.

7. *Dimensional.* The model must contain relations that respect the dimensionality of the variables involved (this prevents, for example, constraints like `ADD(Inflow,L,...)` from appearing in models for the single-tank system).

8. *Single.* Well-posed models should not contain two or more disjoint sub-models.

9. *Causal.* The model must be causally ordered (Iwasaki and Simon, 1986). In a simple sense, this requires a variable that appears on the right-hand side of a (qualitative) arithmetic constraint should have appeared on the left-hand side of a constraint earlier in the sequence.

The following constraints on the qualitative variables were also used. These are ad-hoc, but were nevertheless found to be extremely effective in constraining the space of possible models:

10. *New variables.* A well-posed model can contain no more than some prescribed number of new, or "hidden", variables. Increasing this number usually increases the value of b in Remark 3 (this is equal to 1 for the single tank model: the hidden variable is `Netflow`).

11. *Irrelevant variables.* Variables in one constraint that are never used by another constraint are taken to be irrelevant. A well-posed model can contain no more than some prescribed number of irrelevant variables (this is equal to 0 for the single tank model).

12. *Distinct variables.* All variables in any constraint are distinct.

13. *Dynamic variables.* Well-posed models must include `DERIV` constraints for any pre-specified "dynamic" variables (these are variables that are known to change with time).

With these inputs, we summarise the results of using incremental model construction to identify a model for the coupled tanks system. Model construction proceeds in two stages. In the first stage, we attempt to identify a single tank model, by ignoring observations for levels and flows in tank A. Figure 14 shows the well-posed models identified by the system. The model with the lowest cost is extended in an attempt to identify a model for the coupled tanks system. Recall that the models selected from Stage 1 are subject to a limited form of generalisation before attempting to identify a model in Stage 2. The result of this generalisation step is shown in Figure 15. Each of these models are extended in Stage 2 to construct final models for the coupled tanks system: the results are in Fig. 16 (the fourth one is the correct model for the system).

Model No.	Model	Cost

1	`MPLUS(Lb,OutflowB)`
	`SUB(Fab,OutflowB,NetflowB)`
	`DERIV(Lb,NetflowB)`
		−9.13

2	`SUB(OutflowB,Fab,NetflowB)`
	`MMINUS(Lb,E)`
	`DERIV(E,NetflowB)`
		−5.37

3	`SUB(Fab,OutflowB,NetflowB)`
	`MPLUS(Lb,E)`
	`DERIV(E,NetflowB)`
		−5.37

4	`SUB(Fab,OutflowB,NetflowB)`
	`DERIV(Lb,E)`
	`MPLUS(NetflowB,E)`
		−5.37

5	`SUB(Fab,OutflowB,NetflowB)`
	`DERIV(Lb,E)`
	`MMINUS(NetflowB,E)`
		−5.37

Figure 14: Well-posed models for the single tank system identified by the first stage of incremental learning. Only the lowest cost model is returned: the rest are shown here for illustrative reasons.

Model No.	Model
1	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB)
2	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,G)
3	MPLUS(Lb,F) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB)
4	MPLUS(Lb,F) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,H)
5	MPLUS(Lb,F) SUB(Fab,F,NetFlowB) DERIV(Lb,NetFlowB)	

Figure 15: Generalisations of the lowest cost model for the single tank system. These models are extended to identify models for the coupled tank system.

Model No.	Model
1	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) ADD(OutflowB,NetflowA,H) ADD(NetflowB,H,InflowA)
2	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) ADD(NetflowB,NetflowA,H) ADD(OutflowB,H,InflowA)
3	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) MPLUS(InflowA,H) MMINUS(InflowA,H)
4	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) MPLUS(Fab,Diff) ADD(Lb,Diff,La)
5	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) MPLUS(NetflowB,Diff) ADD(Lb,Diff,La)	

Figure 16: Well-posed models identified for the coupled tank system. These were obtained by extending the lowest-cost model obtained for the single tank system in Fig. 14 (these are the first three constraints in all the models here). All models shown here have equal (lowest) cost. Model 4 is the target model

The coupled tank system identification task can clearly be decomposed into two stages: the identification of the single tank system consisting of 3 qualitative constraints, followed by its extension by 4 further constraints. It is instructive to illustrate the gain in efficiency from using the incremental procedure. Figure 17 shows the comparative effects of: (1) *No decomposition*. We

attempt to identify all seven constraints in a single stage (this is the "one-shot" approach); and (2) *Correct decomposition*. The single tank model is identified in Stage 1, and then extended to the coupled tank model in Stage 2. This resulted in the models in Figs. 14 and 16.

Figures 16 and 17 illustrate two points we wish to draw attention to about the procedure we have employed, namely:

1. The result may not be a unique model. For the identification of biological systems about which little is known, we do not see this as being a hindrance; and in many cases may even be preferable. New experiments could be proposed to discriminate between the models.

2. Decomposition can significantly increase the efficiency of system identification. No great significance should also be attached to the fact that the correct model is identified even with inappropriate decompositions—recall that the greedy search procedure employed is sub-optimal—although the robustness demonstrated is heartening, since in practice we may not be in a position to know the correct decomposition.

Decomposition	Correct Model Identified?	Time Taken
None	Yes	$>$ 5 days
Correct	Yes	2037 seconds

Figure 17: The effect of decomposition on system identification. The ILP system without decomposition was halted after 5 days of execution.

Finally, while most of the constraints 1–9 on well-posed models are motivated by some well-understood principles underlying qualitative reasoning, the constraints 10–13 on qualitative variables are not. Figure 18 provides some empirical justification for the use of these constraints, by illustrating the proportion of a (uniform) random sample of 10,000 models, all of which satisfy constraints 1–9, but fail these constraints. Based on this proportion constraint 13 has the single strongest effect, followed by 12, 11, and 10.

6. Applications to Biological Systems

In this section we demonstrate the application of the incremental technique described to biological system identification. The demonstrations here serve a dual purpose. First, they are intended to illustrate the ability of a general-purpose ILP system to identify qualitative models for biological systems at significantly different scales of organisation. For this, we have elected to examine modelling problems at the ecosystem, organ, extra-cellular and cellular levels. Second, we intend to demonstrate the ability of the incremental approach proposed to construct models in three different ways: in a single stage without providing any initial model (thus acting as a "one-shot" system identifier); in a single stage by refining an approximate model provided; and in multiple stages. In all cases, the ILP system will use the refinement operator and cost function described in Appendix A. The background knowledge will also largely be the same, consisting of definitions for qualitative constraints. All tasks are of a re-constructive nature and examples are observations generated using

Constraint	Description	Estimate of Proportion of Models Eliminated
10. New variables	New variables = 1	33.07%
	New variables = 2	8.84%
11. Irrelevant variables	Irrelevant variables = 0	73.81%
	Irrelevant variables = 1	33.15%
12. Distinct variables	Distinct variables = true	98.52%
	Distinct variables = false	0.00 %
13. Dynamic variables	Dynamic variables = true	100.00%
	Dynamic variables = false	0.00 %

Figure 18: Estimates of the reduction in the search space by the constraints introduced on qualitative variables. The last column represents the proportion of 10,000 models that satisfy the constraints 1–9 described in Coghill et al. (2005) but fail the corresponding constraint in the second column.

the target qualitative model. The goal in each case is to examine if this target model is amongst the models identified by the ILP system. More details can be found in Section A.3.

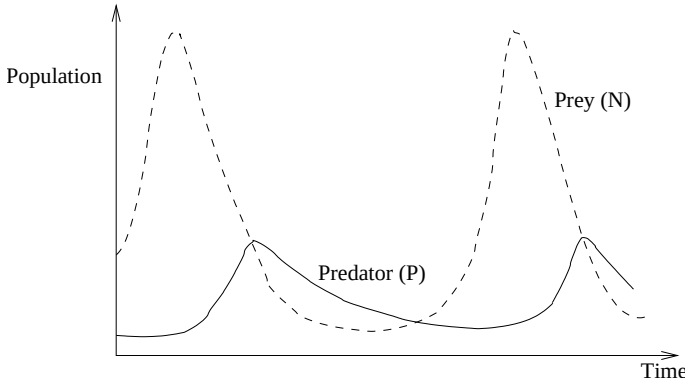
6.1 Ecosystem-Level System Identification

In this section we consider a problem in modelling the dynamics of populations. Specifically we are concerned with identification of a predator-prey model, following the description in Todorovski and Džeroski (2001), which in turn is based on mathematical models developed for the same problem in Murray (1993).

The ecosystem considered is a simple one consisting of populations of predator and prey species—foxes and rabbits, say—that interact in the following manner. Assume that foxes only eat rabbits and that rabbits only eat grass, of which there is an unlimited supply. If the rabbit population is large, the fox population grows. In turn, many rabbits are eaten, resulting in a fall in their numbers. A smaller number of rabbits causes more foxes to die of starvation. Fewer foxes then causes an increase in the rabbit population, which leads to the entire cycle being repeated. This kind of oscillatory behaviour of the two populations is shown in Fig. 19(a). The dynamics of the populations can be modelled using the Lotka-Volterra model, a variant of which is shown in Fig. 19(b). Under certain simplifying assumptions described, the qualitative model is in Fig. 19(c).

We examine reconstructing the model in Fig. 19(c) by using the incremental ILP system as a single-shot model constructor. For this, the ILP system is provided with: (a) the same background knowledge as in Section 5.1; (b) example observations of system behaviour generated using the target model in Fig. 19(c); and (c) the refinement operator and the Bayesian cost function described in Appendix A. The incremental search procedure commences with an empty model (by convention, denoted by $\emptyset$) as the initial hypothesis (see Fig. 20).

The results are shown in Fig.21. Model 1 is the target model. All models were constructed in a single-stage containing 6 qualitative constraints, in approximately 65 seconds of processor time.



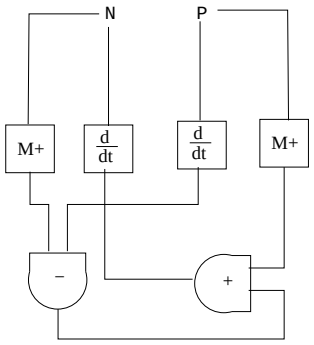
(a)

$$\frac{dN}{dt} = g(N) - c(P)$$

$$\frac{dP}{dt} = c(P) - d(P)$$

(b)

Diagrammatic Model:



Qualitative Model:

```
DERIV(N,Ndot)
DERIV(P,Pdot)
MPLUS(N,G)    ,
MPLUS(P,D)
SUB(G,Pdot,P1)
ADD(P1,D,Ndot)
```

(c)

Figure 19: Modelling predator-prey populations. The changes in populations are shown graphically in (a). There are two system variables: the predator population (P) and the prey population (N). At any given point in time, these variables satisfy the differential equation model in (b). This is a general form of the Lotka-Volterra model for population dynamics. The terms in the model are as follows: $g(N)$ represents the growth-rate of the of the prey in the absence of predators; $c(P)$ is the consumption rate of the predators; and $d(P)$ is the decay-rate of the predators. Under the simple assumptions of $g(N) \propto N$ and $d(P) \propto P$, the corresponding qualitative model is in (c).

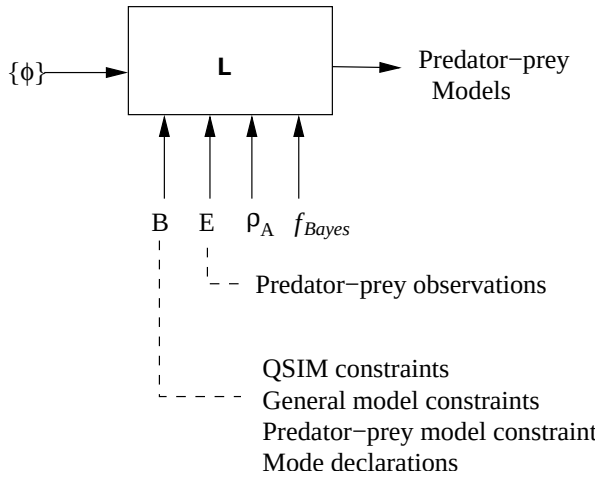


Figure 20: Incremental model identification of predator prey models.

Model No.	Model	Model No.	Model
1	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(P,F) SUB(E,F,G) MPLUS(N,G)	2	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(P,F) SUB(E,F,G) MMINUS(N,G)
3	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(N,F) SUB(E,F,G) MMINUS(P,G)	4	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MMINUS(P,F) SUB(E,F,G) MMINUS(N,G)
5	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(N,F) SUB(F,E,G) MINUS(P,G)		

Figure 21: Predator-prey models identified. The target model in Fig. 19(c) is Model 1.

6.2 Organ-Level System Identification

In this section we consider identification of a qualitative model for the human lung. The primary function of the lung is to act as a gas-exchanger. Exchange of gases across a barrier occurs simply because of a difference in pressures. The pulmonary artery carrying blood from the heart contains low concentrations of oxygen and high concentrations of carbon dioxide (at a constant temperature, the concentrations of the gases are proportional to their partial pressures). Oxygen diffuses across the barrier into the blood (and carbon dioxide diffuses into the lung), where is carried by haemoglobin molecules in the pulmonary vein to the heart. This oxygenated blood is then pumped by the heart to the rest of the body using the arterial network. A model of the lung acting in this man-

ner is shown in Fig. 22. The model is constructed using partial pressures of a measurable "marker" gas. A simplification of the model in Fig. 22(c) results from ignoring the blood vessels and treating the lung as a simple gas chamber as shown in Fig. 23(a). The resulting differential equation model is in Fig. 23(b) and the qualitative model is in Fig. 23(c).

We examine reconstructing a model for the lung by providing the ILP system with the approximate model Fig. 23(c): we are interested in investigating whether the ILP system can refine this to the model in Fig. 22(d). The ILP system is provided with: (a) the same background knowledge as in Section 5.1, with additional mode declarations needed for the `MULT` constraint; (b) example observations of system behaviour generated using the target model in Fig. 22(d); (c) the usual refinement operator and cost function. The incremental search is provided with an intial hypothesis consisting of an approximate model for the lung: `MULT(Va,Pa,F)`, `MULT(Vid,Pi,G)`, `DERIV(F,G)` (see Fig. 24).[4]

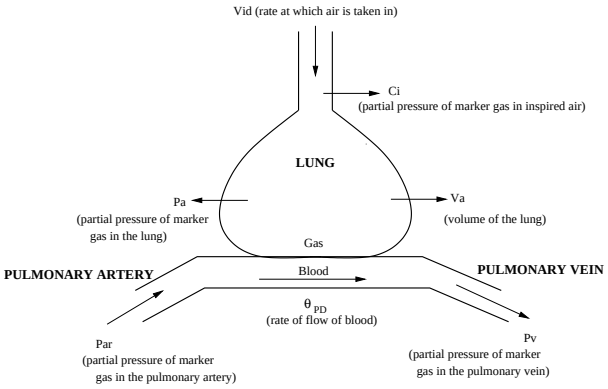
The results, shown in Fig.25, were obtained in 528 seconds of processor time. We note here that model identification required a generalisation of the approximate model provided (`DERIV(F,G)` is changed to `DERIV(F,H)`). Model 2 is the correct model.

6.3 Extra-Cellular System Identification

We use glucose-insulin balance in the human body as a third test case for incremental system identification by ILP. Hormones are chemical messengers, usually small proteins, that play a regulatory role in an organism. Of these, the best known is probably insulin, the first protein whose structure was determined (the amino acid sequence, or primary structure, was determined in 1953 by Sanger and Tuppy). The role of insulin is primarily in maintaining the balance of glucose in the blood. Glucose is used as a source of energy by the central nervous system and by the muscles, and as a source of fat by adipose tissue and the liver, that stores it in the form of a starch called glycogen (see Fig. 26a). If the concentration of glucose in the blood rises too high (usually after digestion of food in the small intestine) then specialised cells in the pancreas are stimulated to produce insulin, by a process involving glycolysis (which we consider in the next section). The presence of insulin signals muscles, fat tissue and the liver to consume glucose, thus lowering it content in the blood. This lower amount of glucose in turn inhibits the production of insulin, and sugar levels rise again until a balance is achieved. This feedback process is not dissimilar to the functioning of a thermostat to maintain a constant temperature in a house. A model of this regulatory mechanism is shown in Fig. 26(b). The model is from Clancy and Kuipers (1994), and is based on a compartmental differential equation model developed by Ironi and Stefanneli (Ironi and Stefanelli, 1994).

Our goal is to reconstruct the qualitative model in Fig. 26(b) using by starting from the empty model. We examine identification of the full model in two stages: the first stage being concerned with identifying the insulin component (the first three constraints in the qualitative model) and the second, the glucose component (the remaining six constraints in the model). As before, the ILP system is equipped with: (a) QSIM relations and their definitions along with additional model-specific constraints; (b) example observatons of system behaviour using the target model in Fig. 26(b); and

4. This is provided *a priori*. In this paper, we do not address how such an hypothesis could have been reached: one possible means could be using the kind of simplified reasoning shown in Fig. 23. There is, of course, nothing preventing the incremental learner described here to start from the empty model $\emptyset$ and construct increasingly better approximations. This would, however, require observational data: something we cannot obtain for the lung model in Fig. 23 (it is impossible to ignore the blood vessels in real-life).



(a)

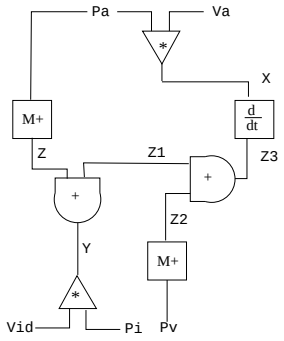
$$\frac{d}{dt}(Va\ Pa) = (Vid\ Pi) - (\lambda\ \theta_{PD}\ (Par - Pv))$$

(b)

$$\frac{d}{dt}(Va\ Pa) = (Vid\ Pi) - (\lambda\ \theta_{PD}\ (Pa - Pv))$$

(c)

Diagrammatic Model:

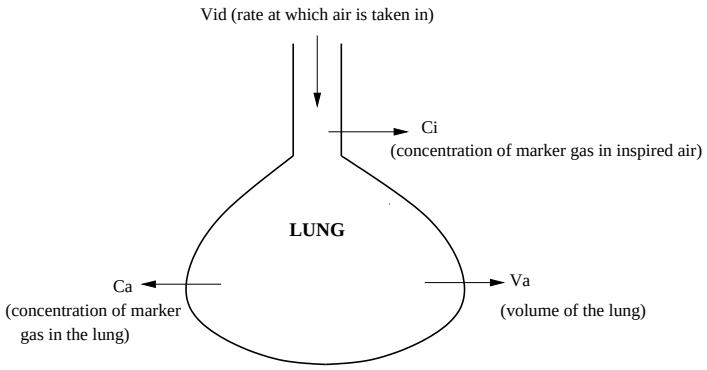


Qualitative Model:

```
MULT(Pa,Va,X)
MULT(Vid,Pi,Y)
MPLUS(Pa,Z)  ,
ADD(Z,Z1,Y)
MPLUS(Pv,Z2)
ADD(Z1,Z2,Z3)
DERIV(X,Z3)
```

(d)

Figure 22: A model for the human lung. In this model, the marker gas is nitrous oxide. There are seven system variables: the rate of inspiration (*Vid*); the concentrations of the marker gas in the inspired air (*Ci*, which is taken to be proportional to the partial pressure *Pi*) and in the lung (*Ca*, proportional to the pressure *Pa*); the volume of the lung cavity (*Va*); the rate of flow of blood θ_{PD}; and the partial pressures in the artery *Par* and the vein *Pv*. On each inspiration, the variables satisfy the differential equation (b). The equation (c) represents the same quantitative model with the assumption that $Pa = Par$, which is reasonable when air is breathed in. The corresponding qualitative model is in (d).

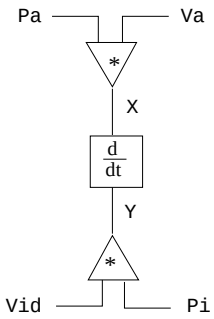


(a)

$$\frac{d}{dt}(Va\ Pa) = (Vid\ Pi)$$

(b)

Diagrammatic Model:



Qualitative Model:

```
MULT(Va,Pa,X)
MULT(Vid,Pi,Y)
DERIV(X,Y)
```

,

(c)

Figure 23: A simplified model of the lung. Ignoring the blood vessels altogether effectively views the lung as the simple gas chamber shown in (a). The resulting quantiative model is in (b) and the corresponding qualitative model is in (c).

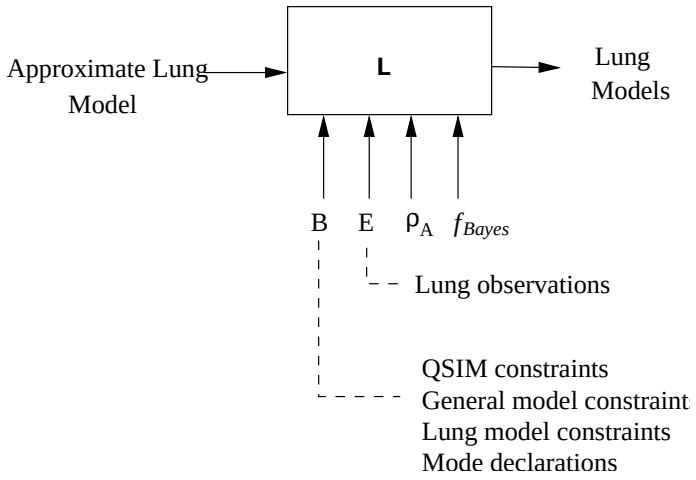


Figure 24: Incremental model identification of lung models.

Model No.	Model	Model No.	Model

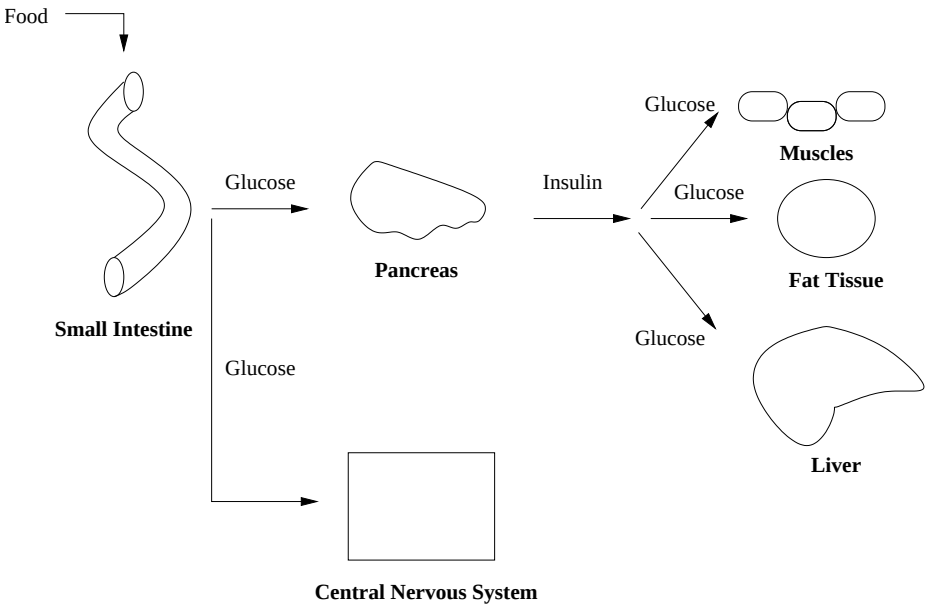
1	`MULT(Va,Pa,F)`
	`MULT(Vid,Pi,G)`
	`DERIV(F,H)`
	`SUB(Pv,Pa,J)`
	`SUB(H,G,I)`
	`MPLUS(I,J)`
		2	`MULT(Va,Pa,F)`
			`MULT(Vid,Pi,G)`
			`DERIV(F,H)`
			`SUB(Pv,Pa,J)`
			`SUB(H,G,I)`
			`MMINUS(I,J)`

3	`MULT(Va,Pa,F)`
	`MULT(Vid,Pi,G)`
	`DERIV(F,H)`
	`SUB(Pv,Pa,I)`
	`SUB(H,I,J)`
	`MPLUS(J,G)`
		4	`MULT(Va,Pa,F)`
			`MULT(Vid,Pi,G)`
			`DERIV(F,H)`
			`SUB(Pv,Pa,I)`
			`SUB(H,I,J)`
			`MMINUS(J,G)`

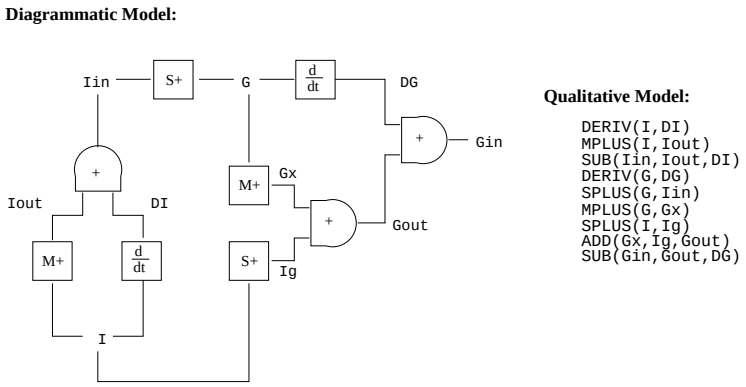
Figure 25: Lung models identified, given the approximate model `MULT(Va,Pa,F)`, `MULT(Vid,Pi,G)`, `DERIV(F,G)`. The target model in Fig. 22(d) is Model 2.

(c) the refinement operator ρ_A and cost function f_{Bayes}. The full system identification process is shown in Fig. 27.

Little difficulty was encountered in identifying the correct constraints for the insulin stage in no more than 1 second of processor time. However, we found it substantially harder to identify the correct constraints for the glucose stage. The principal problems were: (a) a large number of models—over 40, including the one sought—were consistent with the constraints provided; and (b) model evaluation in some cases was extremely slow. We have found two additional constraints to be very useful in reducing the number of models. First, we prevent additions of exogeneous and plasma levels of the same substances (for example, additions of `Iin` and `I`, or `Gin` and `G`). Some plausible justification of this is possible, on the grounds that the two levels are closely related to each other. Second, we prevent monotonic functions of exogeneous inputs `Gin` and `In`, requiring these to be approximated by functions of their counterparts in the blood (that is, `G` and `I`). In addition,



(a)



(b)

Figure 26: Glucose regulation in the blood, shown pictorially in (a), and modelled qualitatively in (b). In the model, `Gin` refers to the glucose intake (in the form of food) and `Iin`, the insulin produced by the pancreas. `G` and `I` are the glucose and insulin levels in the blood. `Gx` is the insulin-independent consumption of glucose by the central nervous system and `Ig` the insulin-dependent consumption of glucose by the muscles, fat tissue and the liver. The qualitative model in Clancy and Kuipers (1994) utilises a sigmoid function `SPLUS`. For the model here, we use the standard `MPLUS` function, which is consistent with the original formulation in Ironi and Stefanelli (1994)

.

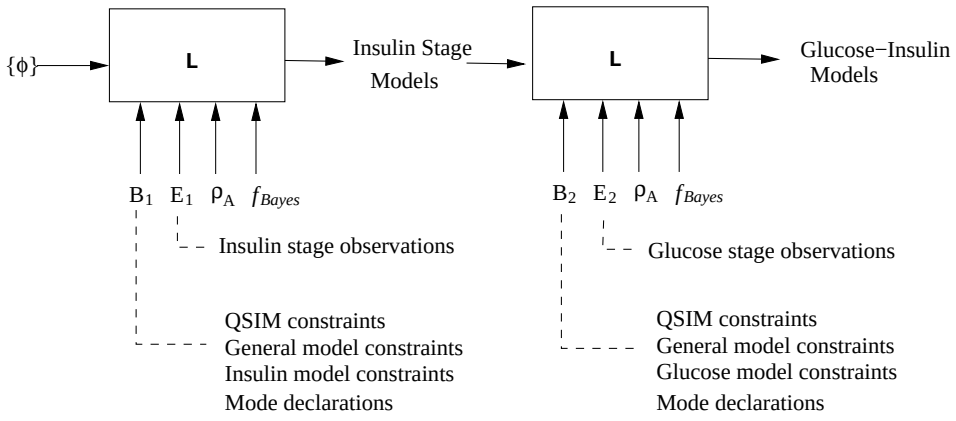


Figure 27: Incremental model identification of models for glucose regulation.

substantially more restrictive mode declarations than in other cases were needed to restrict the search space. Further, we restrict the search to occupy no more than 10,000 seconds of processor time. With these *ad hoc* constraints in place, we are able to repeat model identification using the two stages. The correct insulin model is obtained as before and the results after the glucose stage are shown in Fig.28. Model 3 is equivalent to the target model, given the equivalence of `SPLUS` and `MPLUS` in experiments here.

6.4 Cell-Level System Identification

We use the glycolysis pathway as the final test case for incremental system identification by ILP. Glycolysis is the archetypal pathway. It was historically one of the first to be unravelled, with Otto Meyerhof winning the Nobel prize for discovering key steps in it. Specifically, Meyerhof and colleagues "...were unusually accomplished in breaking down glycolysis into its many separate components, analysing each step separately, then reassembling the constituent parts within an overall system."[5] Glycolysis still presents a challenge to model accurately. The special interest here is that it is significantly different in nature to the models considered so far in the paper, which have all been abstractions of ordinary differential equations. We examine now how the qualitative representation language could be used to develop other kinds of models.

Our qualitative model for glycolysis uses 15 metabolites, namely: pyruvate (pv), glucose (glc), phosphoenolpyruvate (pep), fructose 6-phosphate (f6p), glucose 6-phosphate (g6p), dihydroxyacetone phosphate (dhap), 3-phosphoglycerate (3pg), 1,3-bisphos phoglycerate (1,3bpg), fructose 1,6-biphosphate (f16bp), 2-phosphoglycerate (2pg), glyceraldehyde 3-phosphate (g3p), ADP (adp), ATP (atp), NAD (nad), and NADH (nadh). We have not included H+, H_2O, or Orthophosphate as they are assumed to be ubiquitous. The set of reactions in the pathway are shown in Fig. 29.

We will use the following simple qualitative model for enzymes and metabolites. Metabolites are qualitative variables, whose domains are defined by the name of the metabolite and the landmarks 0 and inf. Qualitative states of the metabolites are restricted to $0/std, 0...inf/std, 0...inf/inc$, $0...inf/dec$. A "qualitative cell-state" is given by the qualitative states of the metabolites of interest in the cell. Enzymes are associated with "qualitative reactions", which result in a qualitative

5. See URL `http://nobelprize.org/physics/articles/states/otto-meyerhof.html`.

Model No.	Model
1	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) ADD(I,Gx,I1) SPLUS(I1,J) SUB(Gin,J,DG)
2	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) ADD(Iout,Gx,I1) SPLUS(I1,J) SUB(Gin,J,DG)
3	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) SPLUS(I,Ig) ADD(Gx,Ig,Gout) SUB(Gin,Gout,DG)
4	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) MMINUS(I,Ig) ADD(Gin,Ig,J) SUB(J,Gx,DG)
5	DERIV(I,DI) DERIV(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(I,Ig) ADD(G,Ig,G1) SPLUS(G1,Gout) SUB(Gin,Gout,DG)	

Figure 28: Models for glucose-insulin regulation. The target model is Model 3.

decrease in the amounts of the reactants and a qualitative increase in the amounts of the products. Examples of each of these are in Fig. 30.

We are interested here in finding a sequence of qualitative reactions that are consistent with the qualitative cell-states before and after glycolysis. For this, we introduce a PATHWAY relation which, for a given sequence of qualitative reactions, holds for pairs of qualitative cell-states $\langle Before, After \rangle$ such that the qualitative state of each metabolite in *Before* can be transformed into its state in $After$ by the qualitative reactions. With this relation, the 3 stage glycolysis process can be modelled as shown in Fig. 31. The reader will note that in this model, reactions proceed sequentially. Of course, biologically speaking, this is not how things happen: reactions that can proceed, do so concurrently. While this can be modelled using a slightly different definition for the PATHWAY relation, the model used here is simpler. There are also good historical reasons to adopt this simpler approach. Glycolsis, as the quote above makes clear, and indeed most other pathways have been uncovered by first experimentally separating them into constituent parts (the qualitative modelling of pathways in (King et al., 2005) did not make this assumption, making the resulting models both difficult to identify—all reactions had to be identified in one-shot—and inefficient to execute).

We examine reconstructing a model for the glycolysis pathway in 3 stages (priming, splitting and phosphorylation). At each stage, the ILP system is provided with: (a) the same background knowledge as in Section 5.1, with additional definitions for the PATHWAY and associated relations. For efficiency, we include three restrictions in the definition of the PATHWAY relation, namely: no

1. (*Hexokinase*): glucose + ATP ⇔ glucose 6-phosphate + ADP.
2. (*Phosphoglucose isomerase*): glucose 6-phosphate ⇔ fructose 6-phosphate.
3. (*Phosphofructokinase*): fructose 6-phosphate + ATP ⇔ fructose 1,6-biphosphate + ADP.

4. (*Aldolase*): fructose 1,6-biphosphate ⇔ dihydroxyacetone phosphate + glyceraldehyde 3-phosphate.
5. (*Triose phosphate isomerase*): dihydroxyacetone phosphate ⇔ glyceraldehyde 3-phosphate.

6. (*Glyceraldehyde 3-phosphate dehydrogenase*): glyceraldehyde 3-phosphate + NAD ⇔ 1,3-bisphosphoglycerate + NADH.
7. (*Phosphoglycerate kinase*): 1,3-bisphosphoglycerate + ADP ⇔ 3-phosphoglycerate + ATP.
8. (*Phosphoglycerate mutase*): 3-phosphoglycerate ⇔ 2-phosglycerate.
9. (*Enolase*): 2-phosphoglycerate ⇔ phospoenolpyruvate.
10. (*Pyruvate kinase*): phospoenolpyruvate + ADP ⇔ pyruvate + ATP.

Figure 29: The reactions comprising the glycolysis pathway. The reactions that consume ATP and NADH are not explicitly included. Glycolysis proceeds in three stages: primary (reactions 1–3), splitting (reactions 4 and 5) and phosphorylation (reactions 6–10). The enzymes involved are in parentheses.

Qualitative states of some metabolites
$atp : 0...inf/std$, $dhap : 0/std$, $nad : 0...inf/dec$

A qualitative cell-state
$\{adp : 0/std, atp : 0...inf/std, f16bp : 0/std, f6p : 0/std, g6p : 0/std, glc : 0...inf/std\}$

A qualitative reaction
$glc + atp \rightsquigarrow g6p + adp$

Some cell-states consistent with $glc + atp \rightsquigarrow g6p + adp$
Before: $\{adp : 0/std, atp : 0...inf/std, f16bp : 0/std, f6p : 0/std, g6p : 0/std, glc : 0...inf/std\}$
After: $\{adp : 0...inf/inc, atp : 0...inf/dec, f16bp : 0/std, f6p : 0/std, g6p : 0...inf/inc, glc : 0...inf/dec\}$
After: $\{adp : 0...inf/inc, atp : 0/std, f16bp : 0/std, f6p : 0/std, g6p : 0...inf/inc, glc : 0...inf/dec\}$

Figure 30: Examples of the qualitative representation used for metabolites, cell-states and chemical reactions. In this, a qualitative reaction causes a qualitative decrease in the reactants and a qualitative increase in the products. The non-determinate nature of qualitative arithmetic means that a cell can be in one of several different states after a reaction.

more than 5 reactions are allowed in a pathway; reactions must use all the metabolites; and reactions have to satisfy some basic constraints of chemical feasibility.[6] In addition, the background knowledge contains an additional constraint that ensures that the model proposed is of the sequential form shown; (b) examples of system behaviour generated using the target model; and (c) the usual refinement operator and cost function. The incremental search procedure commences with the empty model $\emptyset$ as the initial hypothesis (see Fig. 32).

6. The obvious constraint is that products cannot contain elements not available in the reactants. A more sophisticated test estimates the number of chemical bonds broken, and restricts this to at most three: reactions that break more bonds are taken to require an infeasibly large amount of energy, and to be too complex even for an enzyme to manage.

GLYCOLYSIS($Before, After$) if
  PATHWAY($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p, atp + f6p \rightsquigarrow adp + f16bp \rangle$)
  PATHWAY($S_1, S_2, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p \rangle$)
  PATHWAY($S_2, After, \langle g3p + nad \rightsquigarrow 1,3bpg + nadh, 1,3bpg + adp \rightsquigarrow 3pg + atp,$
    $3pg \rightsquigarrow 2pg, 2pg \rightsquigarrow pep, adp + pep \rightsquigarrow atp + pv \rangle$)

where:
PATHWAY($S, F, \langle R_1, R_2, \ldots R_n \rangle$)
  $i = 0$, $S_0 = S$
  for $i = 1 \ldots n$
    QREACTION(S_{i-1}, R_i, S_i)
  $F = S_n$

QREACTION($State, R, NewState$)
  QDECREASE($State, Reactants(R), S$)
  QINCREASE($S, Products(R), NewState$)

Figure 31: A qualitative model for glycolysis. Pathways consist of qualitative reactions, each of which result in a qualitative decrease in the reactants and a qualitative increase in the products. The non-determinacy of qualitative arithmetic means that a qualitative reaction acting on a cell-state could result in one of several new cell-states (since there would be several ways to decrease or increase the qualitative values of metabolites). The system identification task is to find the definition for GLYCOLYSIS given definitions for PATHWAY, QREACTION, QDECREASE and QINCREASE.

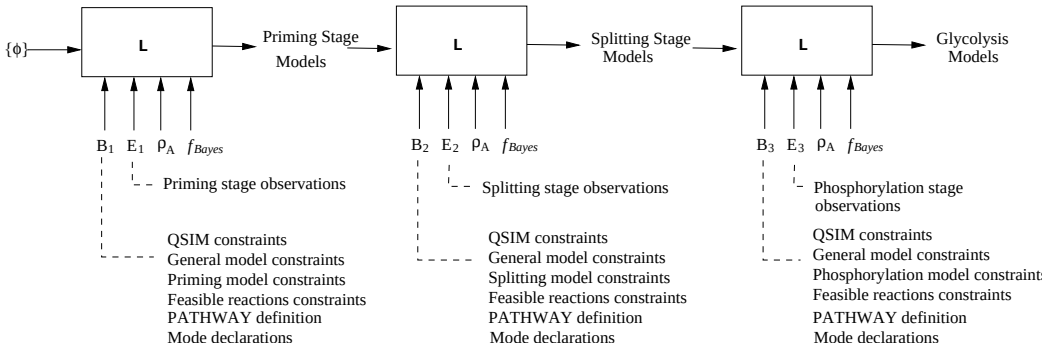


Figure 32: Incremental model identification of models for glycolysis.

The results are shown in Fig. 33. We note here that model identification at Stages 2 and 3 requires a generalisation of the model identified earlier (this removes the co-references to the $After$ variable). The different stages were obtained in 6 seconds (Stage1), 135 seconds (Stage 2) and 5296 seconds (Stage 3).

Stage	Model No.	Model
1	1	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, After, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$)
2	1	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$) PATHWAY($S_1, After, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p\rangle$)
3	1	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$) PATHWAY($S_1, S_2, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p\rangle$) PATHWAY($S_2, After, \langle g3p + nad \rightsquigarrow 1, 3bpg + nadh, 1, 3bpg \rightsquigarrow 3pg,$ $3pg \rightsquigarrow 2pg, 2pg \rightsquigarrow pep, adp + pep \rightsquigarrow atp + pv\rangle$)
	2	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$) PATHWAY($S_1, S_2, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p\rangle$) PATHWAY($S_2, After, \langle g3p + nad \rightsquigarrow 1, 3bpg + nadh, 1, 3bpg + adp \rightsquigarrow 3pg + atp,$ $3pg \rightsquigarrow 2pg, 2pg \rightsquigarrow pep, adp + pep \rightsquigarrow atp + pv\rangle$)

Figure 33: Glycolysis models identified. The target model is Model 2 in Stage 3. The difference in the two models identified in Stage 3 arise in the seventh equation (the second in the last PATHWAY constraint). Model 1 proposes $1, 3bpg \rightsquigarrow 3pg$ and Model 2 proposes $1, 3bpg + adp \rightsquigarrow 3pg + atp$.

7. Decomposition as Search

So far, we have taken the position that the incremental learner will be provided with a decomposition of the system to be identified. While this may be entirely reasonable when we have access to appropriate expertise—a biologist specialising in the kind of systems we are modelling for example—it is of some interest to examine whether a suitable decomposition can be identified automatically. That is, given observations for some system variables, can we automatically decompose the learning task into one that uses a n-stage incremental learner of the form shown in Fig. 6.

Decomposition of complex systems has been studied extensively in econometrics, ever since the pioneering work of Simon and Ando (1961). In this, decomposability of a system is a property of the system by which some subsets of variables (usually non-intersecting) have a greater interaction with each other than other subsets. These subsets define sub-systems into which the larger system can be decomposed. Simon and Ando study the formal properties of linear dynamical systems of the form $x(t+1) = Ax(t)$, where A is some linear operator and the applicability of their results to evolutionary systems has been studied by Shpak et al. (2004a,b). Concerned as we are with a logical representation of a system, our problem is related more to the decomposition of Boolean functions. Most modern work on this stems from that of Ashenhurst (1957) and Curtis (1962). In this, a function f of n variables, denoted here by the set S, is decomposed into Boolean functions h and g, such that $f(X) = h(A, g(B))$, where $A, B \subset S$ and $A \cup B = S$. The techniques are devised for propositional logic, and it is not evident how they could be used to address the decomposition task here. Nevertheless, at least one important principle is directly applicable: both the Ashenhurst and Curtis formulations are essentially procedures that look for suitable decompositions by examining all possible partitions of S. In general though, finding the optimal decomposition of Boolean

functions is computationally hard (see, for example, Boros et al., 1994), and some form of heuristic search is inevitable. This is the basis of the work of Paulson and Wand (1992), who examine the decomposition of a system specified by state variables. Decomposition here means discovering both a partitioning into non-disjoint subsets of system variables, and an assignment of variables in each subset as being "input" or "output". Each subset constitutes a subsystem, and a pair of subsystems are related if an output variable of one is an input variable of another. The final decomposition is the result of a heuristic search process guided by: a set of constraints characterising "good decompositions", a set of rules for enumerating candidate decompositions, and a method of scoring each candidate based on the complexity of the resulting subsystems.

Unlike Paulson and Wand's procedure, we require that related subsystems share models (rather than system variables). Nevertheless, we are able to draw on their basic premise of decomposition being the result of a heuristic search process. Specifically, we use a randomised local search procedure that identifies each stage of the decomposition using a randomised local search procedure that executes the following steps: (a) A subset of system variables is selected randomly from candidate subsets for this decomposition; (b) A model is constructed using this set and its cost determined; (c) All possible "local moves" are constructed. These result in new subsets obtained by adding a system variable not in the original set and by removing a system variable included in the original set; (d) The best local move (the new subset having a model with least cost) is selected and Steps (c)–(d) repeated (the number of repetitions denoted by M). The procedure halts after some fixed number of such iterations, and the best scoring subset is returned. Actually, Steps (a)–(d) are repeated several times (denoted by R) with each repetition starting with a different random subset in Step (a). The best scoring subset across all repetitions is returned and the the entire procedure repeated for the next stage. The reader will recognise the procedure as the GSAT algorithm (Selman et al., 1992) adapted to the problem of automatic decomposition. As with all such procedures, the goal is to obtain an efficient (but sub-optimal) solution to an inherently intractable problem (see Appendix B for details). Needless to say that with this, as with the Paulson and Wand work, automatic decomposition is only worthwhile provided the additional computational burden imposed by searching for the decomposition is less than that of attempting to find the complete model using a one-shot (single-stage) learner.

We apply the procedure to the task of decomposing the coupled tanks system. The reader will recall that this system (shown in Fig 8) is specified by 5 system variables: `InflowA`, `OutflowB`, `Fab`, `La` and `Lb`. The automatic decomposition task is as follows: given values for the 5 system variables, identify the single tank "subsystem" specified by `OutflowB`, `Fab` and `Lb` and then identify the the final model using the single tank submodel and the remaining system variables. Figure 34 summarises the result of employing the randomised local search procedure just described to identify the correct decomposition.

It is evident that for such a small problem, we will quickly explore all of the search space as R and M are increased. Nevertheless, the tabulation shows that very small values of R and M yield variable results. The extent to which the success with moderate values of R and M can be replicated on larger, real systems remains a topic for future research. In Appendix B, we are able to offer some insight by considering artificial problems created by random decompositions of larger sets of variables. These experiments suggest that a 2-stage decomposition of a system like the coupled tanks would require R and M values of approximately 5.

We turn now to the automatic decomposition of the first multi-stage biological system considered in this paper. The glucose-insulin system is comprised of 4 independent system variables

R	M		
	1	2	3
1	0.27	0.70	0.40
2	0.50	0.60	0.90
3	0.40	0.90	1.00

Figure 34: Probability estimates of identifying the correct decomposition of the coupled tanks system, using the randomised local search procedure described in Appendix B. Here R denotes the number of restarts of the randomised procedure and M the number of iterations of local moves. Each entry is is the probability of identifying correctly the single-tank model, followed by the correct coupled tanks model, with the corresponding values of R, M. Probability estimates required for each stage are obtained from 10 repeats of the randomised procedure.

(`Gin`, `G`, `Iin`, `I`), and requires a 2-stage decomposition. Once again, using the experiments on synthetic problems as a guide, we are able to obtain a correct decomposition for this system using low values of R and M (5 in this case). Unfortunately, the decomposition procedure we have just described cannot be used to obtain a decomposition for the glycolysis problem. Here, system variables (metabolites) are re-used across the different stages, which violates a key assumption of the approach (simply speaking, variables used in a stage cannot be re-used at a later stage). This violation makes the search vastly harder, putting in perspective the achievement of Meyerhof and his colleagues.

8. Concluding Remarks

The focus in biology has, until recently, been mainly on individual units. Molecular biology, for example, has mainly focussed on individual molecules and on their properties as isolated entities or as complexes in very simple model systems. However, biological molecules in living systems participate in very complex networks, including regulatory networks for gene expression, intracellular metabolic networks and both intra- and inter-cellular communication networks. Such networks are involved in the maintenance (homeostasis) as well as the differentiation of cellular systems of which we have a very incomplete understanding. Nevertheless, the progress of molecular biology has made possible the detailed description of the components that constitute living systems, notably genes and proteins. Large scale genome sequencing means that we can (at least in principle) delimneate all macromolecular components of a given cellular system. Microarray experiments as well as large scale proteomics will soon give us large amounts of experimental data on gene regulation, molecular interactions and cellular networks. The challenge now becomes to understand how these individual components integrate to complex systems and the function and evolution of these systems, thus scaling up from molecular biology to "systems" biology that provides an understanding at different levels of biological organisation.

Our experience in the physical sciences suggests that the only tractable way of understanding complex systems is through the use of mathematical models. However, biological systems are usually far more complex than physical or human-engineered ones and progress in determining functionality will be crucially dependent on the development of mathematical, and computational tech-

niques specially devised for biological data analysis, modelling and simulation. In this paper, we argue that a qualitative representation of values, along with a powerful machine-learning approach like ILP, provides a useful tool for system-identification at different levels of biological organisation. We have sought to back this claim by demonstrating the use of a general-purpose ILP system to identify models for systems at three disparate levels of biological organisation, namely, ecosystem, organ and cellular. The results are promising, with the target model being amongst a small set of answers returned in each case. While the applications presented have been re-construction of known models, this is clearly not the use we envisage for the approach. Specifically, we expect its principal utility will be in situations where there is some quantitative data of variable quality and quantity, and not much is known about a suitable mathematical model. In these circumstances, the data can be converted to a qualitative representation (in the manner described by Hau and Coiera, 1997) and one or more qualitative models identified. These can then form the basis of understanding the system better and could even be used to direct the construction of quantitative model. For example, they could form the basis of the grammars required for an automated technique such as the one described in Todorovski et al. (2000) (in some sense, this is like extending the Q^2-learning framework in Suc et al. (2003) to the discovery of mathematical models). An additional feature of our work here is that the approach is an incremental one, that seeks to construct the final model in stages. The value of decomposition as an aid to understanding complex systems has long been recognised: Courtois (1985) describes some general principles that motivate the need for such an approach. We believe that when attempting to construct large models with ILP, some form of structured induction (in the sense intended in Shapiro, 1987) would be required. The decomposition into a sequence of stages is an example of such a structuring.

The work presented here has a number of limitations. There are limitations to the power of the qualitative representation used: (1) they can only provide clues to the precise mathematical structure. This may be sufficient for common-sense reasoning about a system, but is clearly insufficient for a complete understanding; (2) simulations with qualitative models can contain spurious behavious; and (3) abstractions appear to be largely restricted to ODE models. It has been suggested that the use of "multivariate constraints" (Wellman, 1991) may allow abstractions of PDE models, but little has been done on that front.

An important limitation of the incremental approach is that the user needs to provide an adequate decomposition of the system-identification task into stages along with the number of constraints in the model for each stage. The latter restriction can be relaxed by providing an upper-bound on the number of constraints. We have described a randomised procedure that attempts to construct a suitable decomposition automatically. The results are encouraging, but the procedure still involves constructing many models and its performance on real problems requires further investigation. The randomised procedure itself is an adaptation of the GSAT algorithm of Selman and colleagues. Minor modifications of this yield procedures akin to WalkSat (Selman et al., 1994) and simulated annealing. Both may yield better algorithms for automatic decomposition than the one here; as would modifications that would allow estimation of model performance without actually requiring their explicit construction.

In the implementation of the incremental learner, the primary limitation of the greedy strategy adopted means that we cannot prove that the models returned are the best possible. A further limitation is the use of a refinement operator that can only perform a restricted kind of refinement of models found at a previous stage: this was done solely to keep the space of possible models within manageable limits (in effect, a limited form of theory-revision is performed). A refinement

operator that performs both generalisations and specialisations could be used, but the computational cost would be substantial.

Finally, applications of the approach have been restricted to re-construction of known target models using simulated data. Clearly, it remains to be shown that similar success can be achieved with real experimental data.

These limitations notwithstanding, we believe the combination of a qualitative representation and an incremental ILP approach to be particularly well-suited to the identifying systems at different levels of biological organisation, for the following reasons: (1) The qualitative representation overcomes some inherent limitations in the data—specifically, noise and sparsity—which make quantitative modelling difficult; (2) Qualitative models provide the correct level of comprehensibility for the non mathematically-minded biologist; and (3) Models of interest usually involve the relationship between a number of different components. Currently, ILP provides the most powerful—and in many cases, the only—framework for identifying such relations, but its use is often hampered by concerns of efficiency. The incremental approach we have described provides one way of overcoming these concerns.

Acknowledgments

The authors would like to thank George Coghill for helping us understand the constraints defining well-posed qualitative models and pointing out an important practical flaw in an early attempt at constructing the lung model. David Gavaghan, of the Computing Laboratory, Oxford helped us with quantitative models for the human lung. Binesh Mangar attempted to construct some of the qualitative models for the coupled tanks and the lung during the course of his M.Sc. at Oxford. Simon Garrett performed extensive experiments with a one-shot ILP model constructor for model physical systems and with a different representation for glycolysis. The results from those experiments were very helpful in motivating the approach described here. Thanks are also due to Ravi Kothari, of the IRL, who drew our attention to the work of Ashenhurst on the decomposition of switching functions. The authors would like to dedicate this paper to the memory of Donald Michie, who died as this paper was being written. He taught us much of what we know in science and machine learning and his guiding hand is greatly missed.

Appendix A. ILP Details

We now describe the specification and implementation details relevant to the ILP system used in the paper.

A.1 Specification

In this paper, we closely follow the specification provided by Muggleton (1994) for an ILP system designed to construct models (usually called hypotheses in the ILP literature) given background knowledge B and observations (usually called examples in the ILP literature) E In this specification an ILP algorithm is one that satisfies the following requirements (reproduced with minor changes from Srinivasan and Kothari, 2005):

Given:

R1. $B \in \mathcal{B}$: background knowledge encoded as statements in logic. This includes I: a set of constraints that should not be violated by an acceptable hypothesis.

R2. $E \in \mathcal{E}$: a finite set of examples $= E^+ \cup E^-$ where:

$E^+ = \{e_1, e_2, \ldots\}$ is a set of definite clauses (these are the positive examples);
$E^- = \{\overline{f_1}, \overline{f_2} \ldots\}$ is an optional set of Horn clauses (these are the negative examples); and
$B \not\models E^+$

Find:

R3. $H \in \mathcal{H}$: a hypothesis such that the following conditions are met:

Sufficiency. This consists of:

$S1$. $B \cup H \models E^+$

Consistency. This consists of:

$C1$. $B \cup H \not\models \square$; and

$C2$. $B \cup H \cup E^- \not\models \square$

The requirement $C1$ ensures that H does not violate any of the constraints I in B. The requirement $C2$ is intended to ensure that H does not contain any over-general clauses. Often, implementations do not require clauses to meet this requirement, as some members of E^- are taken to be noisy. This specification is then refined to allow theories to be inconsistent with some negative examples. We will use the phrase "H explains E, given B" to denote that at least $S1$ and $C1$ are met. An "acceptable H" is any H that explains E, given B.

The specification does not state how acceptable H's are to be constructed, or, if several H's explain the E, then which of them are to be selected. For this, we introduce the following functions:

- A "downward" refinement operator $\rho : \mathcal{H} \rightarrow 2^{\mathcal{H}}$ s.t. $\rho(h) \subseteq \{h' | h \models h'\}$. Given a $h \in \mathcal{H}$, this function returns a subset of the elements of $\mathcal{H}$ that are implied by h.
- A cost function $f : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \Re$. Given a $h \in \mathcal{H}$, $B \in \mathcal{B}$ and $E \in \mathcal{E}$, this function returns an evaluation of h;

Let $\rho^1(h) = \rho(h)$; $\rho^n(h) = \{h'' | \exists h' \in \rho^{n-1}(h)$ s.t. $h'' \in \rho^1_A(h')\}, (n \geq 2)$; and $\rho^*(h) = \rho^1(h) \cup \rho^2(h) \cup \ldots$. With some abuse of notation, let $\rho^1(\{h_1, h_2, \ldots\}) = \rho^1(h_1) \cup \rho^1(h_2) \cup \ldots$; $\rho^n(\{h_1, h_2, \ldots\}) = \rho^n(h_1) \cup \rho^n(h_2) \cup \ldots$; and $\rho^*(\{h_1, h_2, \ldots\}) = \rho^1(\{h_1, h_2, \ldots\}) \cup \rho^2(\{h_1, h_2, \ldots\}) \cup \ldots$

Then, given an initial set of hypotheses $H_0 \subseteq \mathcal{H}$ we specify a particular kind of ILP algorithm $\mathtt{L}(B, E, H_0, \rho, f)$ by modifying the requirement R3 above to:

R3′. $H = \mathtt{L}(B, E, H_0, \rho, f) \subseteq \rho^*(H_0)$: a set of hypotheses such that for each $h \in H$ the following are met:

Sufficiency. This consists of:

$S1$. $B \cup h \models E^+$

Consistency. This consists of:

$C1$. $B \cup h \not\models \square$; and

$C2$. $B \cup h \cup E^- \not\models \square$

Minimal Cost. This consists of:

$F1$. For all $h' \in \rho^*(H_0)$ $f(h', B, E) \geq f(h, B, E)$

We are now in a position to specify an incremental ILP system that uses the algorithm L. Given a finite sequence $\langle S_1, S_2, \ldots, S_k \rangle$ ($k \geq 1$), where each S_i consists of the tuple (B_i, E_i), ($B_i \in \mathcal{B}$ and $E_i \in \mathcal{E}$); $H_0 \subseteq \mathcal{H}$; a downward refinement operator ρ; and a cost function f, find H_k, where $H_i = \mathtt{L}(B_i, E_i, H_{i-1}, \rho, f)$ ($1 \leq i \leq k$).

A.2 Implementation

The basic task of addressed by L described in the previous section can be viewed as a discrete optimisation problem. In general terms, this is posed as follows: given a finite discrete set S and a cost-function $f : S \rightarrow \Re$, find a subset $H \subseteq S$ such that $H = \{s | s \in S \text{ and } f(s) = \min_{s_i \in S} f(s_i)\}$. An optimal algorithm for solving such problems is the "branch-and-bound" algorithm, shown in Fig. 35 (the correctness, complexity and optimality properties of this algorithm can be found in Papadimitriou and Steiglitz, 1982). A specific variant of this algorithm is available within the software environment comprising ALEPH (Srinivasan, 1999). The modified procedure is in Fig. 36. The principal differences from Fig. 35 are:

1. The procedure is given a set of starting points H_0, instead of a single one (i in Fig. 35);
2. A limitation on the number of nodes explored (n in Fig. 36);
3. The use of a boolean function $acceptable : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \{FALSE, TRUE\}$. $acceptable(k, B, E)$ is $TRUE$ if and only if k satisfies requirements $S1$ and $C1$ in Section A.1 (given B and E);
4. Inclusion of background knowledge and examples (B and E in Fig. 36). These are arguments to both the refinement operator ρ (the reason for this will become apparent shortly) and the cost function f.

We now describe an implementation for an incremental procedure for model identification that assumes that the task has been decomposed into a finite sequence of stages $\langle S_1, S_2, \ldots, S_k \rangle$ ($k \geq 1$). Each S_i consists of the tuple (B_i, E_i), where B_i and E_i refer to the background knowledge and examples relevant to stage i. With this decomposition in place, Fig. 37 shows a simple greedy implementation used to identify the final models.
Finally, we turn to some points concerning the implementation used in this paper:

- Qualitative models are represented as definite clauses. Given a definite clause C, the qualitative constraints in the model (the size of the model) are obtained by counting the number of qualitative constraints in C. This will also be called the "size of C".
- Constraints, such as the restrictions to well-posed models, are assumed to be encoded in the background knowledge;
- $acceptable(C, B, E)$ is $TRUE$ for any qualitative model C that is consistent with the constraints in B, given E.

$bb(i, \rho, f)$: Given an initial element i from a discrete set S; a successor function $\rho : S \rightarrow 2^S$; and a cost function $f : S \rightarrow \Re$, return $H \subseteq S$ such that H contains the set of cost-minimal models. That is for all $h_{i,j} \in H$, $f(h_i) = f(h_j) = f_{min}$ and for all $s' \in S \backslash H$ $f(s') > f_{min}$.

1. *Active* := $\langle (i, -\infty) \rangle$.
2. *worst* := ∞
3. *selected* := $\emptyset$
4. while *Active* $\neq \langle\rangle$
5. begin
   (a) remove element $(k, cost_k)$ from *Active*
   (b) if $cost_k < worst$
   (c) begin
      i. $worst := cost_k$
      ii. $selected := \{k\}$
      iii. let $Prune_1 \subseteq Active$ s.t. for each $j \in Prune_1$, $\underline{f}(j) > worst$ where $\underline{f}(j)$ is the lowest cost possible from j or its successors
      iv. remove elements of $Prune_1$ from *Active*
   (d) end
   (e) elseif $cost_k = worst$
      i. $selected := selected \cup \{k\}$
   (f) $Branch := \rho(k)$
   (g) let $Prune_2 \subseteq Branch$ s.t. for each $j \in Prune_2$, $f_{min}(j) > best$ where $f_{min}(j)$ is the lowest cost possible from j or its successors
   (h) $Bound := Branch \backslash Prune_2$
   (i) for $x \in Bound$
      i. add $(x, f(x))$ to *Active*
6. end
7. return *selected*

Figure 35: A basic branch-and-bound algorithm. The type of *Active* determines specialised variants: if *Active* is a stack (elements are added and removed from the front) then depth-first branch-and-bound results; if *Active* is a queue (elements added to the end and removed from the front) then breadth-first branch-and-bound results; if *Active* is a prioritised queue then best-first branch-and-bound results.

- *Active* is a prioritised queue sorted by f;
- The successor function used is ρ_A. This is defined as follows. Let S be the size of an acceptable model and C be a qualitative model of size S' with $n = S - S'$. We assume B constains a set of mode declarations in the form described in Muggleton (1995). Then, given a definite clause C, obtain a definite $C' \in \rho_A(C, B, E)$ where $\rho_A = \rho_A^n = \langle D' | \exists D \in \rho_A^{n-1}(C, B, E)$ s.t. $D' \in \rho_A^1(D, B, E) \rangle, (n \geq 2)$. $C' \in \rho_A^1(C, B, E)$ is obtained by adding a literal L to C, such that:
  - Each argument with mode $+t$ in L is substituted with any input variable of type t that appears in the positive literal in C or with any variable of type t that occurs in a negative literal in C;
  - Each argument with mode $-t$ in L is substituted with of any variable of C of type t that appears before that argument or by a new variable of type t;

$bb_A(B,E,H_0,\rho,f,n)$: Given background knowledge $B \in \mathcal{B}$; examples $E \in \mathcal{E}$; a set of initial elements H_0 from a discrete set of possible hypotheses $\mathcal{H}$; a successor function $\rho : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow 2^{\mathcal{H}}$; a cost function $f : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \Re$; and a maximum number of nodes $n \in \mathcal{N}$ ($n \geq 0$) to be explored, return $H \subseteq \mathcal{H}$ such that H contains the set of cost-minimal models of the models explored.

1. $Active = \langle\rangle$
2. for $i \in H_0$
   (a) add $(i, -\infty)$ to $Active$
3. $worst := \infty$
4. $selected := \emptyset$
5. $explored := 0$
6. while ($explored < n$ and $Active \neq \langle\rangle$)
7. begin
   (a) remove element $(k, cost_k)$ from $Active$
   (b) increment $explored$
   (c) if $acceptable(k,B,E)$
   (d) begin
      i. if $cost_k < worst$
      ii. begin
         A. $worst := cost$
         B. $selected := \{k\}$
         C. let $Prune_1 \subseteq Active$ s.t. for each $j \in Prune_1$, $\underline{f}(j,B,E) > worst$ where $\underline{f}(j,B,E)$ is the lowest cost possible from j or its successors
         D. remove elements of $Prune_1$ from $Active$
      iii. end
      iv. elseif $cost_k = worst$
         A. $selected := selected \cup \{k\}$
   (e) end
   (f) $Branch := \rho(k,B,E)$
   (g) let $Prune_2 \subseteq Branch$ s.t. for each $j \in Prune_2$, $\underline{f}(j,B,E) > worst$ where $\underline{f}(j,B,E)$ is the lowest cost possible from j or its successors
   (h) $Bound := Branch \backslash Prune_2$
   (i) for $x \in Bound$
      i. add $(x, f(x,B,E))$ to $Active$
8. end
9. return $selected$

Figure 36: A variant of the basic branch-and-bound algorithm, implemented within the ALEPH system. Here $\mathcal{B}$ and $\mathcal{E}$ are sets of logic programs; and $\mathcal{N}$ the set of natural numbers.

– Each argument with mode $\#t$ in L is substituted with a ground term of type t. This assumes the availability of a generator of elements of the Herbrand universe of terms; and
– $acceptable(C',B,E)$ is $TRUE$.

The following properties of ρ_A^1 (and, in turn to ρ_A) can be shown to hold (Riguzzi, 2005):

– It is locally finite. That is, $\rho_A^1(C,B,E)$ is finite and computable (assuming the constraints in B are computable);

incsearch(S, H_I, ρ, f, n, m) : Given a sequence of stages $S = \langle (B_1, E_1), (B_2, E_2), \ldots, (B_k, E_k) \rangle$, $(1 \leq k < \infty)$ where $B_i \in \mathcal{B}$; $E_i \in \mathcal{E}$; a set of initial elements H_I from a discrete set of possible hypotheses $\mathcal{H}$; a successor function $\rho : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow 2^{\mathcal{H}}$; and a cost function $f : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \Re$; a maximum number of nodes $n \in \mathcal{N}$ $(n \geq 0)$ to be explored at each stage; and a maximum number of models $m \in \mathcal{N}$ $(m \geq 0)$ to be returned at each stage; and return $H \subseteq \mathcal{H}$

1. $H_0 := randomselect(m, I)$
2. $i := 1$
3. while $(i \leq k)$
4. begin
   (a) $H'_{i-1} := \{h' | h \in H_{i-1}$ and $h' = generalise(h)\}$
   (b) $H'_i := \{h' | h' = bb_A(B_i, E_i, H'_{i-1}, \rho, f, n)\}$
   (c) $H''_i := nonredundant(B_i, H'_i)$
   (d) $H_i := randomselect(m, H''_i)$
   (e) increment i
5. end
6. return H_k

Figure 37: A simple incremental procedure for system identification. Given a decomposition into k stages, the best models found at each stage are refined further.

- It is weakly complete. That is, any clause containing n literals can be obtained in n refinement steps from the empty clause;
- It is not proper. That is, C' can be equivalent to C;
- It is not optimal. That is, C' can be obtained multiply by refining different clauses.

In addition, it is clear by definition that given a qualitative model C, $accep\ table(C', B, E)$ is $TRUE$ for any model $C' \in \rho^1_A(C, B, E)$. In turn, it follows that $acceptable(C', B, E)$ is $TRUE$ for any $C' \in \rho_A(C, B, E)$.

- The cost function used is $f_{Bayes}(C, B, E) = -\mathrm{P}(C|B, E)$ where $\mathrm{P}(C|B, E)$ is the Bayesian posterior probability estimate of clause C, given background knowledge B and positive examples E. Finding the model with the maximal posterior probability (that is, lowest cost) involves maximising the function (McCreath, 1999):

$$Q(C) = \log D_{\mathcal{H}}(C) + p \log \frac{1}{g(C)}$$

where $D_{\mathcal{H}}$ is a prior probability measure over the space of possible models; $p = |E|$, the number of positive examples; and g is the generality of a model. We use the approach used in the the ILP system C-Progol to obtain values for these two functions. That is, the prior probability is related to the complexity of models (more complex models are taken to be less probable, *a priori*); and the generality of a model is estimated using the number of random examples entailed by the model (the details of this are in Muggleton, 1996);

- The function $randomselect(m, H)$ in Fig. 37 randomly selects (without replacement) m elements of the set H (or all the elements of H if its cardinality is less than m);

- For all stages i in Fig. 37, the bb_A constructs no more than n models for each stage. Here we restrict n to 1000;

- For all stages i in Fig. 37, no more than m of the lowest-cost models are returned; Here we restrict m to 1000;

- The function *generalise* in Fig. 37 is restricted to "splitting" variable co-references apart (see Definition 27 and Lemma 31 in Muggleton (1995) and Remark 4 below for more on this); and

- The function *nonredundant* in Fig. 37 returns a set of non-redundant models. Given background knowledge B and a set of models S encoded as definite clauses, a model $C_1 \in S$ is redundant, iff for $S_1 = S - \{C_1\}$, $B \cup S \equiv B \cup S_1$. It can be shown that this entails checking that $B \cup S_1 \models C_1$. $nonredundant(B,H)$ returns all elements $C \in H$ which do not satisfy this redundancy check.

We now report on some properties of the various procedures described. It is evident that *incsearch* in Fig. 37 performs the same function as a non-incremental (single-shot) ILP system if $k = 1$, $H_I = \{\emptyset\}$ (that is, H_I consists of the empty model) and $m \geq n$.

Remark 1 Termination, correctness and sub-optimality *Termination of bb_A follows trivially if the number of nodes searched (n) is finite; and calls to acceptable and f terminate. It is also easy to see that the conditional statement on Step 7c ensures that, for all models $k \in selected$, $acceptable(k,B,E)$ is TRUE. All models returned by bb_A are correct in this sense. Since models returned by incsearch on any iteration i are a subset of the models returned by bb_A, it follows that all models returned by incsearch are also correct. The branch-and-bound procedure is known to be optimal, in that can identify the lowest cost models in the search space $\mathcal{H}$. However, bb_A with $\rho = \rho_A$ is optimal if and only if $n \geq |\mathcal{H}|$ and $H_I = \{\emptyset\}$. It follows that incsearch with $\rho = \rho_A$ is only optimal if and only if $k = 1$, $H_I = \{\emptyset\}$, $n \geq |\mathcal{H}|$, and $m \geq n$.*

Although a general statements about search complexity can be made, the following remarks refer specifically to the search for qualitative models.

Remark 2 Search space for qualitative models. *Let the number of qualitative constraints in acceptable models be restricted to some size d. Given element a single starting element i size d_i, the task of bb_A (we will assume n to be large) is to return all models of size d. This is done by examining all models returned by ρ_A that adds $d - d_i$ constraints to i. In the worst case, each $i = \emptyset$ and ρ_A has to return all models of size d. If the maximum recall number of any mode declaration be bound by some constant b, then there are at most b extensions of size 1, b^2 extensions of size 2 and so on, up to b^d models of size d. That is, given a model i, the number of acceptable models of size d constructed by ρ_A is at most b^d.*

We now consider an incremental procedure that simply selects some of the best qualitative models found at a stage for refinement at the next stage. It follows that the size of the search space depends principally on the maximum number of qualitative constraints added at any stage.

Remark 3 Incremental search space for qualitative models. (simple case). *Assume as before that the target model is restricted to d constraints and the maximum recall of any mode declaration is b. Assume further that model identification can be decomposed into $k \geq 1$ stages, with each stage resulting in models with $d_1, d_2, \ldots, d_k$ constraints (we will assume that all models in the initial set H_I have $d_0 \geq 0$ constraints and that $d_{i+1} \geq d_i$). At each stage i, a model is constructed by addition of $d_i^+ = d_i - d_{i-1}$ constraints to a model selected at stage $i-1$. For each model selected at stage $i-1$, we know from Remark 2 that bb_A constructs at most $b^{d_{i-1}^+}$ acceptable models. Since no more than m are selected at stage $i-1$, the total number of models constructed at stage i is $mb^{d_{i-1}^+}$. The total number of models constructed by the entire procedure is no more than $\sum_{i=1}^{k} mb^{d_i^+}$. That is, the total number of models constructed is $O(b^{d_{max}^+})$ where $d_{max}^+ = \max(d_1^+, d_2^+, \ldots, d_k^+)$.*

Models at a stage may not consist of a simple addition of constraints to those found earlier and we consider generalising models by splitting variables, before adding constraints (as shown in Fig. 37). For qualitative models, this translates to retaining the qualitative components found at the previous stage but disconnecting connections between some or all pairs whose outputs are connected together. While a general analysis will require a detailed description of the variable splitting procedure, a less detailed calculation is possible for the kinds of qualitative models sought here. The simplification results primarily from the "Distinct variables" restriction on well-posed models.

Remark 4 Incremental search space for qualitative models (limited generalisation). *Assume as before that the target model is restricted to d constraints and the maximum recall of any mode declaration is b. Assume further that model identification can be decomposed into $k \geq 1$ stages, with each stage resulting in models with $d_1, d_2, \ldots, d_k = d$ constraints (we will assume that all models in the initial set H_I have $d_0 \geq 0$ constraints and that $d_{i+1} \geq d_i$). We will now examine the effect of allowing generalisation by variable splitting only. For a model M selected at stage i, it is evident that if all variables in a constraint are distinct, then there can be at most $n_i = \max(d_i - 1, 0)$ co-references to any one variable v in M. Let the set of positions with co-references to a variable v be E_v. Variable splitting essentially renames variables at some or all of these positions into new ones. This is tantamount to partitioning the set E_v into equivalence classes, with positions in each equivalence class having the same variable; and each such partitioning giving rise to a model M' that is more general than m (in the sense that M' θ-subsumes Plotkin, 1970). The* nth *Bell number $B(n)$ gives the number of ways in which a set of size n can be partitioned into equivalence classes.*[7] *Thus, the number of models resulting from splitting variable co-references to a variable v in a model M from stage i is at most $B(n_i)$. If the maximum number of variables in any qualitative constraint is bounded by A, then there can be at most $s_i = Ad_i$ splittable variables in any model M from stage i. Therefore the number of models after generalisation of any M from stage i is at most $G(i) = B^{s_i}(n_i)$ Since there are no more than m models at any stage i, the total number of models after generalisation is no more than $mG(i)$. Recall each of these is then specialised by bb_A to construct a model for stage $i+1$. The total number of models constructed by the entire procedure is thus no more than $\sum_{i=1}^{k} mG(i-1)\, b^{d_i^+}$.*

7. The *n*th Bell number is equal to $\sum_{k=0}^{n} S_n^{(k)}$. $S_n^{(k)}$, or Stirling numbers of the second kind, describe the way a set of *n* elements can be partitioned into *k* disjoint, non-empty subsets. These can be computed using the formula $S_n^{(k)} = S_{n-1}^{(k-1)} + kS_{n-1}^{(k)}$ (with $S_n^{(1)} = 1$).

In general, it is evident that variable splitting is not the only form of generalisation that may be needed: components found at a previous stage may have to be discarded entirely before constructing a model for the current stage. It is evident that allowing this form of generalisation will significantly increase the worst-case search complexity.

Remark 5 Incremental search space for qualitative models (general case). *Assume as before that the target model is restricted to d constraints and the maximum recall of any mode declaration is b. Assume further that model identification can be decomposed into $k \geq 1$ stages, with each stage resulting in models with $d_1, d_2, \ldots, d_k$ constraints (we will assume that all models in the initial set H_I have $d_0 \geq 0$ constraints and that $d_{i+1} \geq d_i$). From Remark 4 above, we know that the number of models after generalisation by splitting variables at stage i is $mG(i)$, each with d_i constraints. Each of these models can be generalised further by dropping one or more constraints. This results in a total of $mG(i)2^{d_i}$ models, each of which is then specialised by bb_A to construct a model for stage $i+1$. In the worst case, all the constraints found in each of the models at stage i are removed by the generalisation step and the specialisation step at stage $i+1$ has to construct models with $i+1$ constraints in each case. The total number of models constructed by the entire procedure is thus no more than $\sum_{i=1}^{k} mG(i-1)2^{d_{i-1}}\, b^{d_i}$.*

A.3 Application

In all cases, the application tasks are of a re-constructive nature. That is, a known target model for each stage of the incremental process is used to generate examples for that stage. These, along with the background knowledge and a set of random examples for the stage are given to the learner. (the random examples are needed for the Bayesian calculation described in the previous section) We then check to see if the target model is amongst the results returned by the learner. All experiments were conducted on a laptop equipped with a 1.5 GHz Intel Pentium M Processor and 768 MB of main memory. Examples are restricted to a random sample of no more than 500 observations of system behaviour and no more than 500 random observations (these are needed for the Bayesian cost calculations). Incremental construction of models was accomplished using ALEPH version 5, with the YAP compiler (version 5.0.1).

We describe here the background knowledge and examples relevant to each of the application tasks presented in the paper. Common to all tasks are definitions of the QSIM constraints. The definitions we use are based on those in Bratko (2001) and are available on request from the first author. In the following sections we describe the encoding of the examples, the mode declarations and the values of the main parameters used for each application task. The principal parameters for system identification are these: (1) The number of constraints in the model (the "size" constraint described in Section 5.1 on well-posed models); (2) Upper bound on the number of occurrences of any kind of constraint (the "language" constraint described in Section 5.1); (3) Upper bound on the the number of nodes to be seached (*n* in the *incsearch* procedure); (4) Upper bound on the the number of models to be selected from a stage (*m* in the *incsearch* procedure); (5) Upper bound on the number of new variables in any model (constraint 10 described in Section 5.1); (6) Upper bound on the number of irrelevant variables in any model (constraint 11 in Section 5.1).

A.3.1 THE TANK MODELS

System variables for the coupled tanks system are `La`, `Lb`, `InflowA`, `Fab` and `OutflowB`. Examples for both the coupled tanks and single tank system are encoded using a $state/5$ predicate (the argu-

```
state(l:0/inc,l:0/std,f:0...inf/std,f:0/inc,f:0/std).
state(l:0/inc,l:0...inf/dec,f:0...inf/std,f:minf...0/inc,f:0...inf/dec).
state(l:0...inf/dec,l:0/inc,f:0...inf/std,f:0...inf/dec,f:0/inc).
state(l:0...inf/dec,l:0...inf/dec,f:0...inf/std,f:0...inf/dec,f:0...inf/dec).
state(l:0...inf/dec,l:0...inf/dec,f:0...inf/std,f:0...inf/inc,f:0...inf/dec).
```

Figure 38: Example observations from the coupled tank system.

```
ADD(+level,+level,-level)   SUB(+level,+level,-level)
ADD(+flow,+flow,-flow)      SUB(+flow,+flow,-flow)

MPLUS(+level,-level)        MPLUS(+level,-flow)
MPLUS(+flow,-flow)          MPLUS(+flow,-level)

MMINUS(+level,-level)       MMINUS(+level,-flow)
MMINUS(+flow,-flow)         MMINUS(+flow,-level)

MINUS(+level,+level)        MINUS(+flow,+flow)

DERIV(+level,-flow)
```

Figure 39: Mode declarations used for identifying the tank system.

ments refer to the system variables `La`–`OutflowB`, in the order just listed). Some of the observations are shown in Fig. 38 (the syntax used is in the Prolog language):
22 observations are generated in all using the correct model for the coupled tank system. For the first stage of learning—the single tank system—observations made for Tank A (that is, `La` and `InflowA`) are ignored. This is achieved using the following mode declaration for $state/5$ (here, the "_" denotes that the corresponding argument is to be ignored):

```
STATE(_,+level,_,+flow,+flow)
```

In contrast, the mode declaration for $state/5$ for the coupled tank system is as follows:

```
STATE(+level,+level,+flow,+flow,+flow)
```

Mode declarations for the QSIM constraints for both single and coupled tanks are shown in Fig. 39.

The values of the principal parameters for the two stages are shown in the tabulation below.

Parameter	Stage 1 (single tank)	Stage 2 (coupled tanks)
Size	3	7
Language	2	2
n	1000	1000
m	1000	1000
Newvars	3	3
Irrelev	0	0

```
state(p:0...inf/dec,n:0...inf/inc).
state(p:0...inf/std,n:0...inf/inc).
state(p:0...inf/inc,n:0...inf/dec).
```

Figure 40: Example observations from the predator-prey system.

```
ADD(+qval,+qval,-qval)      SUB(+qval,+qval,-qval)

MPLUS(+predator,-qval)      MPLUS(+prey,-qval)
MPLUS(+qval,-qval)          MPLUS(+predator,+prey)

MMINUS(+predator,-qval)     MMINUS(+prey,-qval)
MMINUS(+qval,-qval)         MMINUS(+predator,+prey)

MINUS(+predator,+qval)      MINUS(+prey,+qval)
MINUS(+predator,+prey)      MINUS(+qval,+qval)

DERIV(+predator,-qval)
DERIV(+prey,-qval)
```

Figure 41: Mode declarations used for identifying the predator-prey system.

A.3.2 THE PREDATOR-PREY MODELS

System variables for the predator-prey system are the predator population `P` and the prey population `N`. Examples for the system are encoded using a *state*/2 predicate (the arguments of which are `P` and `N`). Some of the observations are shown in Fig. 40.
5 observations of system behaviour are obtained using the target model. The mode declaration for the *state*/2 predicate is

```
STATE(+predator,+prey)
```

Mode declarations for the QSIM constraints are shown in Fig. 41.

The values of principal parameters are shown in the tabulation below.

Parameter	Value
Size	6
Language	2
n	1000
m	1000
Newvars	5
Irrelev	0

A.3.3 THE LUNG MODELS

System variables for identifying the human lung model are `Pa`, `Va`, `Pi`, `Vid` and `Pv`. Examples for the system are encoded using a *state*/5 predicate (the arguments of which are `Pa`–`Pv` in the order just listed). Some of the observations are shown in Fig. 42.
500 observations of system behaviour are obtained using the target model. The mode declaration for the *state*/5 predicate is:

```
state(p:0/std,v:0/inc,p:0...inf/inc,f:0/inc,p:0/inc).
state(p:0/std,v:0...inf/dec,p:0/std,f:0...inf/dec,p:0/std).
state(p:0/std,v:0...inf/dec,p:0/inc,f:0/inc,p:0/inc).
state(p:0/std,v:0...inf/dec,p:0/inc,f:0...inf/std,p:0/inc).
state(p:0/std,v:0...inf/dec,p:0/inc,f:0...inf/inc,p:0/std).
```

Figure 42: Example observations from the lung system.

```
ADD(+press,+press,-press)          SUB(+press,+press,-press)
ADD(+vol,+vol,-vol)                SUB(+vol,+vol,-vol)
ADD(+volrate,+volrate,-volrate)    SUB(+volrate,+volrate,-volrate)
ADD(+qval,+qval,-qval)             SUB(+qval,+qval,-qval)

MPLUS(+press,-press)               MPLUS(+press,-vol)
MPLUS(+press,-volrate)             MPLUS(+press,-qval)
MPLUS(+vol,-vol)                   MPLUS(+vol,-volrate)
MPLUS(+vol,-qval)                  MPLUS(+qval,-volrate)
MPLUS(+qval,-qval)

MMINUS(+press,-press)              MMINUS(+press,-vol)
MMINUS(+press,-volrate)            MMINUS(+press,-qval)
MMINUS(+vol,-vol)                  MMINUS(+vol,-volrate)
MMINUS(+vol,-qval)                 MMINUS(+qval,-volrate)
MMINUS(+qval,-qval)

MINUS(+press,+press)               MINUS(+vol,+vol)
MINUS(+volrate,+volrate)           MINUS(+qval,+qval)

MULT(+press,+press,-qval)          MULT(+press,+vol,-qval)
MULT(+press,+volrate,-qval)        MULT(+press,+qval,-qval)
MULT(+vol,+vol,-qval)              MULT(+vol,+volrate,-qval)
MULT(+vol,+qval,-qval)             MULT(+qval,+volrate,-qval)
MULT(+qval,+qval,-qval)

DERIV(+qval,-qval)
```

Figure 43: Mode declarations used to identify the lung system.

```
STATE(+press,+vol,+press,+volrate,+press)
```

Mode declarations for the QSIM constraints are shown in Fig. 43.

The values of principal parameters are shown in the tabulation below.

Parameter	Value
Size	7
Language	2
n	1000
m	1000
Newvars	6
Irrelev	0

```
state(l:0...inf/dec,l:0...inf/inc,f:0...inf/dec,l:0...inf/inc,l:0...inf/std,f:0/inc).
state(l:0...inf/dec,l:0...inf/inc,f:0...inf/dec,l:0...inf/inc,l:0...inf/inc,f:0...inf/dec).
state(l:0...inf/dec,l:0...inf/inc,f:0...inf/dec,l:0...inf/inc,l:0...inf/inc,f:0...inf/inc).
state(l:0...inf/std,l:0...inf/std,f:0/std,l:0...inf/std,l:0...inf/std,f:0/std).
state(l:0...inf/dec,l:0...inf/dec,f:0...inf/std,f:0...inf/inc,f:0...inf/dec).
```

Figure 44: Example observations from the glucose-insulin regulatory system.

```
ADD(+glevel,+glevel,-glevel)   SUB(+ilevel,+ilevel,+iflow)
ADD(+ilevel,+ilevel,-ilevel)   SUB(+glevel,+glevel,+gflow)

MPLUS(+glevel,-glevel)         MPLUS(+glevel,-ilevel)
MPLUS(+ilevel,-glevel)         MPLUS(+ilevel,-ilevel)

SPLUS(+glevel,-glevel)         SPLUS(+glevel,-ilevel)
SPLUS(+ilevel,-glevel)         SPLUS(+ilevel,-ilevel)

MMINUS(+glevel,-glevel)        MMINUS(+glevel,-ilevel)
MMINUS(+ilevel,-glevel)        MMINUS(+ilevel,-ilevel)

SMINUS(+glevel,-glevel)        SMINUS(+glevel,-ilevel)
SMINUS(+ilevel,-glevel)        SMINUS(+ilevel,-ilevel)

MINUS(+glevel,+glevel)         MINUS(+ilevel,+ilevel)

DERIV(+glevel,+gflow)          DERIV(+ilevel,+iflow)
```

Figure 45: Mode declarations used for identifying the glucose-insulin models.

A.3.4 THE GLUCOSE-INSULIN MODELS

System variables for the glucose-insulin models are `Gin`, `G`, `DG`, `Iin`, `I`, and `DI`. Of these `DG` and `DI` are dependent on the glucose and insulin variables and could have been inferred from them. Examples for both the insulin and glucose stages are encoded using a *state*/6 predicate (the arguments refer to the system variables `Gin`–`DI`, in the order just listed). Some of the observations are shown in Fig. 44 (the syntax used is in the Prolog language):
24 observations are generated in all using the correct model for glucose-insulin regulation. For the first stage of learning—the insulin stage—observations relevant to glucose (that is, `Gin`, `G` and `DG`) are ignored. This is achieved using the following mode declaration for *state*/6 (here, the "_" denotes that the corresponding argument is to be ignored):

```
STATE(_,_,_,+ilevel,+ilevel,+iflow)
```

In contrast, the mode declaration for *state*/6 for the glucose stage is as follows:

```
STATE(+glevel,+glevel,+gflow,+ilevel,+ilevel,+iflow)
```

Mode declarations for the QSIM constraints for both insulin and glucose stages are shown in Fig. 45. The values of the principal parameters for the two stages are shown in the tabulation below.

```
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0/std,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0...inf/dec,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0...inf/std,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0...inf/inc,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0/std,g6p:0...inf/dec,glc:0...inf/dec]).
```

Figure 46: Example observations from the priming stage of glycolysis.

Parameter	Stage 1 (insulin)	Stage 2 (glucose)
Size	3	9
Language	2	2
n	1000	1000
m	1000	1000
Newvars	5	5
Irrelev	0	0

A.3.5 THE GLYCOLYSIS MODELS

System variables for identifying models at any stage of glycolysis are cell-states before and after the stage. Examples for a stage are encoded using a *glycolysis*/2 predicate (the arguments of which are the cell-state before and after the reactions involved in that stage). Some of the observations for the first stage (priming) are shown in Fig. 46.
500 observations of system behaviour are obtained using the target models for each stage. The mode declaration for the *glycolysis*/2 predicate is:

```
GLYCOLYSIS(+cellstate,-cellstate)
```

Although QSIM constraints form the basis of qualitative reactions, the models constructed use a *pathway*/3 predicate. The mode declaration for this predicate is simply:

```
PATHWAY(+cellstate,#qreactions,-cellstate)
```

(Here the "#" indicates that a corresponding argument is a ground term: in this case a sequence of qualitative reactions).

The values of principal parameters for the three stages are shown in the tabulation below.

Parameter	Stage 1 (priming)	Stage 2 (splitting)	Stage 3 (phosphorylation)
Size	1	2	3
Language	3	3	3
n	1000	1000	1000
m	1000	1000	1000
Newvars	3	3	3
Irrelev	0	0	0

System Variables	Search Space
4	15
5	181
6	2163
7	27133
8	364395
9	5272861
10	82289163

Figure 47: Number of partition-sequences to be searched for a given number of system variables.

Appendix B. Automatic Decomposition

We present here a specification for the problem of automatic decomposition addressed in the latter half of the paper, along with implementation details of a search procedure that identifies an acceptable decomposition.

B.1 Specification

We will assume that we are looking to decompose a system specified by a set of qualitative system variables. The problem can be specified as follows. Given a non-empty set S of system variables, consider first the notion of a "partition-sequence" $(S_1, S_2, \ldots, S_n)$, in which $S_i \subset S$, and $S_1, S_2 \ldots S_n$ form a partition of S. Given a set E of observed values for the system variables S, background knowledge B, a refinement operator ρ, a cost function f, and a partition-sequence $P = (S_1, S_2, \ldots, S_n)$, we are able to construct a n-stage incremental learner of the form shown in Fig. 6(b) that returns a set of models $H_n = \mathsf{L}(B, E_n, \rho, f, H_{n-1})$ with minimal cost, where $H_0 = \{\emptyset\}$ and at each stage i, E_i are the values observed for variables $S_1 \cup S_2 \cdots S_i$. Let us assume that we are also able to obtain the cost of H_n, which, for reasons that will become obvious immediately, we call C_P. With automatic system decomposition, we are concerned with a procedure that returns an optimal partition-sequence P^* such that, of all possible partition sequences P, $C_{P^*} \leq C_P$ (that is, P^* yields models with the least cost).

Remark 6 Search space for decompositions. *We are able to provide some details on the combinatorics of the search for decompositions. Recall that the number of partitions of a set of n elements into exactly k non-empty blocks is given by $S_n^{(k)}$ (Stirling's number of the second kind). For each such partition, a valid answer is given by an ordering of the blocks into some sequence. The number of k-length partition-sequences is thus $D_n^{(k)} = k!S_n^{(k)}$. We have an additional constraint that requires the first block in any partition-sequence to contain at least 2 elements. This means that we are only interested in partition-sequences of length $1, 2, \ldots, n-2$. The total number of partition-sequences to be considered is therefore $D_n^{(1)} + D_n^{(2)} + \cdots + D_n^{(n-2)}$. This is at most $(n-2)D_n^{(n-2)}$. Of course, models have be constructed with each element of any partition-sequence. The complexity of this has be estimated in the previous section.*

Fig. 47 tabulates the size of the search space for some values of n (the number of system variables), showing how an addition of a system variable increases the size of the search space by an order of magnitude (the number appears always to be greater than 10^{n-3}).

B.2 Implementation

Figure 48 shows a GSAT-like randomised local search procedure that identifies a partition-sequence based on a greedy selection of elements. In experiments for the paper we have made one modification to this procedure, which we have not shown in the figure for simplicity. In Fig. 48 variable subsets are compared simply on costs. If a pair of variable subsets V_1 and V_2 have the same cost, we further examine the following. For each of $V_{1,2}$ we obtain a best-case estimate of the length of the final partition-sequence. The subset with the shorter length is preferred. If these lengths are also the same, then the subset with fewer variables is preferred (on the assumption that the resulting ILP model would be simpler). In addition, of course, we will use $\rho = \rho_A$ and $f = f_{Bayes}$.

Remark 7 Space searched by *rls***.** *Let $|S| = n$. The procedure contains 3 loops in Steps 4, 4f, and 4(f)viii The loop in Step 4(f)viii iterates at most M times. On each iteration, there are at most n local moves from any subset. Therefore, at most Mn subsets are examined by the loop in Step 4(f)viii. This is called no more than R times by the loop in Step 4f, resulting in at most MRn subsets. The outermost loop in Step 4 can iterate no more than n times, which means the number of subsets examined are at most MRn^2.*

B.3 Application

We consider first 3 artificial problems obtained by a random decomposition of a set of 10 variables into 2, 3 and 4 stages. For each problem and stage, the "correct" variable subset is assigned the least cost possible. All other subsets are assigned costs randomly. The task is to find the correct variable subset at each stage for each of the 3 problems.[8] The search space has approximately 10^8 elements (in contrast to the coupled tanks problem, which has about 10^2). Figures 49 summarises the results of attempting to identify the correct decompositions for each of the 3 problems.

More generally, we examine the values of R and M needed for identifying the correct decomposition.[9] The values required using artificial problem sets of the kind just described, with $n = 4, 5, 6, 8,$ and 10 variables with $k = 2, 3,$ and 4 stages are shown in Fig. 50. The probability of obtaining the correct decomposition are shown in Fig. 51.

From these results, it is evident that both an increase in the number of variables or the number of stages usually requires an increase in the values of R and M (Fig. 50a), and that as the values of R and M are increased, the probability of obtaining the correct decomposition increases (Fig. 50b). Both these observations may be evident to the reader since an increase in either the number of variables or stages makes the search space larger. Increasing R and M then allows a more extensive search. A further characteristic of the automatic decomposition problem that may not be as obvious is this: since the number of variables left at each stage less than at the previous stage, we should, in principle, be able to achieve the same performance by starting with high values of R and M and progressively reducing their values after each stage. We do not explore this further, as such a progressive reduction is not a feature of the procedure here.

8. The procedure in the previous section has to be modified slightly, since there is no need to construct models using an ILP learner for these problems.
9. We note that the experiments are concerned with exact identification of the correct decomposition. Approximate identification, not addressed here, would yield higher probabilities.

$rls(S, B, E, \rho, f, R, M)$: Given a non-empty set of system variables S; background knowledge B; a set of values for the system variables E; a refinement operator ρ; a cost function f; an upper bound on the number of restarts R; and an upper bound on the depth of local moves M, returns a partition-sequence $(S_1, S_2, \ldots, S_k)$, in which each S_i results in the lowest cost model at stage i, given models constructed for S_{i-1}.

1. $i = 0$
2. $H_i = \{\emptyset\}$, $S_i = \emptyset$
3. $VarsLeft = S$
4. while $VarsLeft \neq \emptyset$ do
   (a) Increment i
   (b) $VarsUsed = S_0 \cup S_1 \cdots S_{i-1}$
   (c) $bestcost = \infty$
   (d) $VarsSelected = VarsLeft$
   (e) $r = 0$
   (f) while $r < R$ do
      i. $VarsAvail = VarsLeft \setminus VarsUsed$
      ii. Randomly select $V \subset VarsAvail$
      iii. Let c be the cost of the models returned by an incremental learner constructed using H_{i-1}, B, E, $VarsUsed \cup V$, ρ, f
      iv. if $c < bestcost$ then
         A. $bestcost = c$
         B. $VarsSelected = V$
      v. endif
      vi. $bestlocal = V$
      vii. $m = 0$
      viii. while $m < M$ do
         A. Let L be the set of all variable subsets constituting local moves from $bestlocal$
         B. Let V' be the element of L resulting in the least cost c' using an incremental learner constructed using H_{i-1}, B, E, $VarsUsed \cup V'$, ρ, f
         C. $bestlocal = V'$
      ix. if $c' < bestcost$ then
         A. $bestcost = c'$
         B. $VarsSelected = V'$
      x. endif
      xi. Increment m
      xii. endwhile
   (g) Increment r
   (h) endwhile
5. $S_i = VarsSelected$
6. $VarsLeft = VarsLeft \setminus VarsSelected$
7. endwhile

return $(S_1, S_2, \ldots, S_i)$

Figure 48: A randomised local search procedure for decomposing a set of system variables into a partition-sequence from which an incremental ILP learner can be constructed.

R	M			
	3	10	25	50
3	0.00	0.15	0.12	0.10
10	0.30	0.48	0.64	0.64
25	0.30	0.80	0.90	0.90
50	0.50	0.80	1.00	1.00

(a) 2-stage decomposition $(\{4,6,8,9,10\},\{1,2,3,5,7\})$

R	M			
	3	10	25	50
3	0.03	0.18	0.18	0.10
10	0.20	0.30	0.60	0.60
25	0.60	0.60	0.80	0.90
50	0.80	1.00	0.90	1.00

(a) 3-stage decomposition $(\{1,5,6,8,10\},\{4,9\},\{2,3,7\})$

R	M			
	3	10	25	50
3	0.03	0.00	0.00	0.00
10	0.00	0.18	0.04	0.32
25	0.25	0.0.36	0.40	0.40
50	0.18	0.63	0.72	0.64

(a) 4-stage decomposition $(\{2,3,4\},\{10\},\{6,7,8,9\},\{1,5\})$

Figure 49: Probability estimates of identifying the correct decomposition using the randomised local search procedure on artificial problems with a set of 10 variables $S = \{1,2,\ldots,10\}$. The decompositions to be identified are shown as sequences $(S_1,S_2,\ldots,S_k)$ where $S_i \subset S$, $S_i \cap S_j = \emptyset$ and k represents the number of stages. The probability estimates required for each stage are obtained from 30 trials of using the randomised procedure.

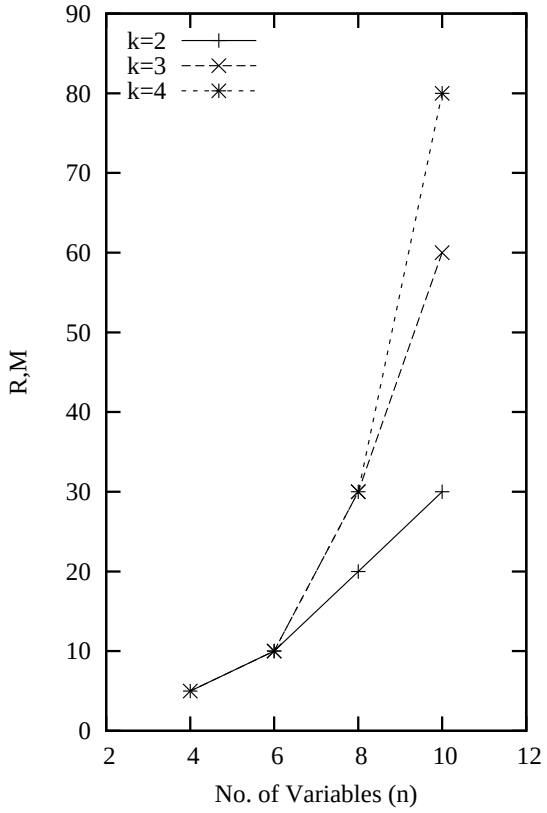


Figure 50: (a) R, M estimates for identification of the correct decomposition of a set of n variables into k stages.

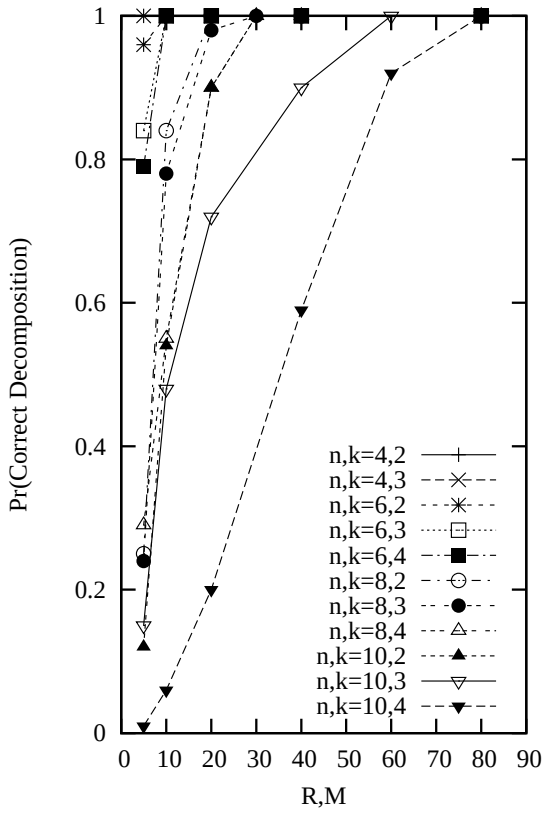


Figure 51: Probability of identifying the correct decomposition for specific values of n, k as a function of R, M.

References

R.L. Ashenhurst. The decomposition of switching functions. In *Proceedings of an International Symposium on the Theory of Switching*, pages 74–116. Harvard University, 1957. (The earliest report with this title by the author is in a Bell Telephone Labs Report No. BL-1(11), in 1952.).

E. Boros, V. Gurvich, P.F. Hammer, T. Ibaraki, and A. Kogan. Decomposition of partially defined Boolean functions. Technical Report 94-9, DIMACS, March 1994.

I. Bratko. *Prolog Programming for Artificial Intelligence*. Addison-Wesley (3rd edition), London, 2001.

I. Bratko, I. Mozetic, and N. Lavrac. *Kardio: A Study in Deep and Qualitative Knowledge for Expert Systems*. MIT Press, Cambridge, 1989.

I. Bratko, S. Muggleton, and A. Varsek. Learning qualitative models of dynamic systems. In S. Muggleton, editor, *Inductive Logic Programming*, pages 437–452. Academic Press, London, 1992.

D.J. Clancy and B. Kuipers. Model decomposition and simulation. In *Proceedings of the Eighth International Workshop on Qualitative Physics about Physical Systems (QR-94)*, Nara, Japan, 1994.

G.M. Coghill, S.M. Garrett, and R.D. King. Learning qualitative models in the presence of noise. In *Proceedings of the QR'02 Workshop on Qualitative Reasoning, Sitges, Spain*, 2002.

G.M. Coghill, S.M. Garrett, A. Srinivasan, and R.D. King. Qualitative system identification from imperfect data. Technical Report AUCS/TR0501, University of Aberdeen, Aberdeen, 2005.

E. W. Coiera. Generating qualitative models from example behaviours. Technical Report 8901, University of New South Wales, Deptartment of Computer Science, May 1989a.

E. W. Coiera. Learning qualitative models from example behaviours. In *Proc. Third Workshop on Qualitative Physics*, pages 45–51, Stanford, August 1989b.

P.-J. Courtois. On time and space decomposition of complex structures. *Communications of the ACM*, 28(6):590–603, June 1985.

H.A. Curtis. *A New Approach to the Design of Switching Circuits*. Van Nostrand, Princeton, NJ., 1962.

D. Hau and E. Coiera. Learning qualitative models of dynamic systems. *Machine Learning Journal*, 26:177–211, 1997. Special Issue on ILP.

A. L. Hodgkin and A. F. Huxley. A quatitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

L. Ironi and M. Stefanelli. In *Proceedings of the Eighth International Workshop on Qualitative Physics about Physical Systems (QR-94)*, Nara, Japan, 1994.

Y. Iwasaki and H. A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986. See also De Kleer and Brown's rebuttal and Iwasaki and Simon's reply to their rebuttal in the same volume of this journal.

R.D. King, S.M. Garrett, and G.M. Coghill. On the use of qualitative reasoning to simulate and identify metabolic pathways. *Bioinformatics*, 21:2017–2026, 2005.

B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.

W.P. Kuo, T-K. Jenssen, A.J. Butte, L. Ohne-Machado, and I.S. Kohane. Analysis of matched mRNA measurements from two different microarray technologies. *Bioinformatics*, 18(2):405–412, 2002.

Y. Lazebnik. Can a biologist fix a radio? Or, what I learned while studying apoptosis. *Cancer Cell*, 2:179–182, 2002.

E. McCreath. *Induction in First Order Logic from Noisy Training Examples and Fixed Example Sizes*. University of New South Wales (PhD. Thesis), Sydney, 1999.

I. Mozetic. Learning of qualitative models. In I. Bratko and Nada Lavrac, editors, *Progress in Machine Learning: Proceedings of EWSL '87: 2nd European Working Session on Learning*, pages 201–217. Sigma Press, 1987.

S. Muggleton. Inductive logic programming: derivations, successes and shortcomings. *SIGART Bulletin*, 5(1):5–11, 1994.

S. Muggleton. Inverse entailment and Progol. *New Gen. Comput.*, 13:245–286, 1995.

S. Muggleton. Learning from positive data. In *Proceedings of the Sixth Inductive Logic Programming Workshop*, LNAI, pages 358–376, Berlin, 1996. Springer-Verlag.

S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

J. D. Murray. *Mathematical Biology*. Springer, Berlin, 1993. Vol. 19, Biomathematics Texts Series, 2nd edition.

C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimisation*. Prentice-Hall, Edgewood-Cliffs, NJ, 1982.

D. Paulson and Y. Wand. An automated approach to information systems decomposition. *IEEE Transactions on Software Engineering*, 18:174–189, 1992.

G.D. Plotkin. A note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163. Elsevier North Holland, New York, 1970.

B. L. Richards, I. Kraan, and B. J. Kuipers. Automatic abduction of qualitative models. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 723–728, San Jose, CA, July 1992.

F. Riguzzi. Two results regarding refinement operators. Technical Report TUM-I0510, Technische Universität Müenchen, Munich, 2005.

A. C. C. Say and S. Kuru. Qualitative system identification: deriving structure from behavior. *Artificial Intelligence*, 83:75–141, 1996.

B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. AAAI Press, 1994.

B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446. AAAI Press, 1992.

A.D. Shapiro. *Structured Induction in Expert Systems*. Addison-Wesley, Wokingham, 1987.

M. Shpak, P.F. Stadler, G.P. Wagner, and L. Altenberg. Simon-Ando decomposability and fitness landscapes. *Theory in Biosciences*, 2004a.

M. Shpak, P.F. Stadler, G.P. Wagner, and J. Hermisson. Aggregation of variables and system decomposition: applications to fitness landscapes. *Theory in Biosciences*, 2004b.

H. A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

T. Soderstrom and P. Stoica. *System Identification*. Prentice Hall, 1989.

A. Srinivasan. The Aleph manual. Available at http://www.comlab.ox.ac.uk/oucl/ research/areas/machlearn/Aleph/, 1999.

A. Srinivasan and R. Kothari. A study of applying dimensionality reduction to restrict the size of a hypothesis space. In *Proceedings of the Fifteenth International Conference on Inductive Logic Programming (ILP2005)*, LNAI 3625, pages 348–365, Berlin, 2005. Springer.

D. Suc, D. Vladusic, and I. Bratko. Qualitatively faithful quantitative prediction. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1052–1057. Morgan Kaufmann, 2003.

L. Todorovski and S. Džeroski. Using domain knowledge on population dynamics modelling for equation discovery. In *Proceedings of the Twelfth European Conference on Machine Learning*, pages 478–490. Springer (LNCS 2167), 2001.

L. Todorovski, S. Dzeroski, A. Srinivasan, J. Whiteley, and D. Gavaghan. Discovering the structure of partial differential equations from example behavior. In *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, 2000. Morgan Kaufmann. URL `ftp://ftp.comlab.ox.ac.uk/pub/Packages/ILP/Papers/AS/pde.ps.gz`.

A.M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B (London)*, 237:37–72, 1952.

W.Bialek and D.Botstein. *Science*, 2004.

M. P. Wellman. Qualitative simulation with multivariate constraints. In *Proc. of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 547–557, 1991.