# Learning qualitative models of dynamic systems

Ivan Bratko (1, 3),
Stephen Muggleton (2) and
Alen Varšek (1)

(1) Faculty of Electr. Eng. and Computer Sc., Ljubljana
(2) The Turing Institute, Glasgow
(3) J. Stefan Institute, Ljubljana

**Abstract**

A technique is described for learning qualitative models of dynamic systems. The QSIM formalism is used as a representation for learned qualitative models. The problem of learning QSIM-type models is formulated in logic, and the GOLEM learning program is used for induction. An experiment in learning a qualitative model of the connected containers system, also called U-tube, is described in detail.

## 1 Introduction

It has been shown that qualitative models are better suited for several tasks than the traditional quantitative, or numerical models. These tasks include diagnosis (e.g. Bratko, Mozetič and Lavrač 1989), generating explanation of the system's behaviour (e.g. Forbus and Falkenheiner 1990) and designing novel devices from first principles (e.g. Williams 1990). We believe that system identification, a fundamental problem in the theory of dynamic systems, is also a task that is done easier at the qualitative level. This paper presents a case study in how this can be done using a logic-based approach to machine learning.

The system identification problem is defined as follows: given examples of the behaviour of a dynamic system, find a model that explains these examples. In this paper we are interested in finding a *qualitative* model. Our working hypothesis is that such models are much easier to learn than classical differential equations models, and that qualitative models can be constructed by means of logic-based approaches to machine learning. Learning of qualitative models is further motivated by another conjecture, investigated in (Bratko 1989), that such models are often sufficient for the synthesis of control rules for dynamic systems.

For a start we have to choose a formalism for defining qualitative models of dynamic systems. Several such formalisms can be considered. Among them are qualitative differential equations, called confluences (de Kleer and Brown 1986), Qualitative Physics Theory (Forbus 1986) and QSIM (Kuipers 1986). For our experiments we chose QSIM for it seems to be mathematically best founded and understood.

In this paper we describe an experiment in which a learning system, called GOLEM, was used. GOLEM (Muggleton and Feng 1990) can be viewed as a simplified and more efficient version of its more known predecessor CIGOL (Muggleton and Buntine 1988). The learning task, suitably formulated in logic for GOLEM or CIGOL is: given background knowledge B and a set of examples E, find a hypothesis H such that:

$$B \wedge H \vdash E$$

This general framework applies to the learning of QSIM-type qualitative models from examples of system's behaviour as follows:

$$QSIMTheory \wedge QualitativeModel \vdash ExamplesOfBehaviour$$

In section 2 we formulate the QSIM approach to qualitative simulation in logic. In section 3 we convert the QSIM qualitative constraints into a form acceptable as background knowledge by the GOLEM program. In section 4 we describe in detail the learning of a model for the U-tube system. Finally in section 5 we compare our approach with some other approaches.

## 2 Formulating QSIM in logic

We will first illustrate the QSIM approach to qualitative modelling by an example. Consider the two connected containers system, often called U-tube, in Figure 1, that will also be used later in the learning experiment. The two containers, $A$ and $B$, are connected with a pipe and filled with water to the corresponding levels $La$ and $Lb$. Let the flow from $A$ to $B$ be $Fab$, and from $B$ to $A$ be $Fba$. A qualitative model of a dynamic system in QSIM is defined

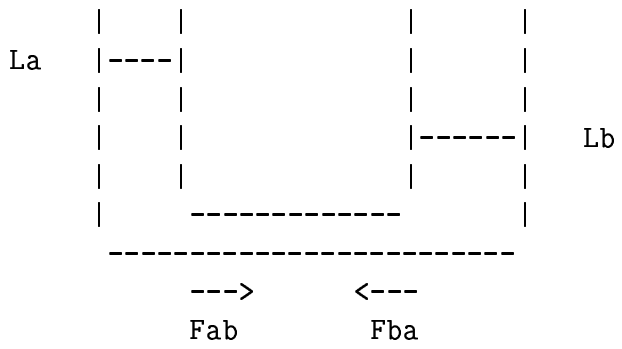


Figure 1: Two connected containers: U-tube system.

as a set of constraints on the time variables of the system. For our system we have two time derivative constraints:

$$\frac{d}{dt}La = Fba$$

$$\frac{d}{dt}Lb = Fab$$

We also have:

$$Fab = -Fba$$

In addition, $Fab$ depends on the water pressure along the pipe: the higher the pressure, the greater the flow. The pressure, in turn, depends on the difference between both levels: the greater the difference, the higher the pressure. This can be formulated in QSIM as:

$$Diff = La - Lb$$

$$Press = M_0^+(Diff)$$

$$Fab = M_0^+(Press)$$

A constraint of the form $y = M_0^+(x)$ means that $y$ is a *monotonically increasing* function of $x$, where the subscript "0" indicates the "corresponding values" $x = 0$, $y = 0$. That is, whenever $x = 0$, also $y = 0$. Notice that if we are not explicitly interested in the pressure, the two $M_0^+$ constraints can be simplified into one:

$$Fab = M_0^+(Diff)$$

The values of variables are in QSIM represented qualitatively by using *landmark* values. For our example, appropriate landmarks for the four variables are, ordered from left to right:

$$La : minf, 0, la0, inf$$

$$Lb : minf, 0, lb0, inf$$

$$Fab : minf, 0, fab0, inf$$

$$Fba : minf, fba0, 0, inf$$

These values are symbolic names corresponding to minus infinity, zero, infinity, and to the initial values of the four variables. The current value of a variable is stated in terms of its landmarks and direction of change. The direction of change can be *inc* (increasing), *std* (steady), or *dec* (decreasing). In the initial state, for example, the value of level $La$ is equal to $la0$ and is decreasing. This will be written as:

```
La  =  la0 / dec
```

In the time interval that follows the initial time point, $La$ is betwen 0 and $la0$, and decreasing:

```
La  =  0..la0 / dec
```

QSIM simulation of our system results in the trace shown in Figure 2. This consists of altogether four qualitative states that correspond to two time points and two time intervals.

| Time | La | Lb | Fab | Fba |
|---|---|---|---|---|
| t0 | la0/dec | lb0/inc | fab0/dec | fba0/inc |
| (t0,t1) | 0..la0/dec | lb0..inf/inc | 0..fab0/dec | fba0..0/inc |
| t1 | 0..la0/std | lb0..inf/std | 0/std | 0/std |
| (t1,inf) | 0..la0/std | lb0..inf/std | 0/std | 0/std |

Figure 2: Trace of behaviour of the system in Figure 1

We will now translate the QSIM approach to qualitative simulation into first order logic. QSIM qualitative simulation algorithm can be sketched in Prolog as something like this:

```
simulate( State) :-               % Start with State
  write( State),
  transition( State, NextState),  % Move to next state
  simulate( NextState).
```

```
% State = state( Variable1, Variable2, ...)

transition( state( V1, V2, ...), state( NewV1, NewV2, ...)) :-
  trans( V1, NewV1),       % Model-independent
  trans( V2, NewV2),
  ...,
  legalstate( NewV1, NewV2, ...).  % Model-dependent
```

Relation *trans* that non-deterministically generates possible transitions is defined as part of the QSIM theory. Thus a model of a particular dynamic system is defined by the relation *legalstate*. This imposes constraints on the values of variables in the model. The repertoire of available constraints that can be used in the definition of *legalstate* is, again, part of the QSIM theory.

For our purpose of learning in logic, we have to formulate these available qualitative constraints as predicates that can be used as background knowledge. The resulting set of predicates may be used in the construction of a model. In the following, $F, F1, F2, \ldots$ stand for terms of the form:

```
Variable: QualValue / DirectionOfChange
```

where $QualValue$ is either a landmark belonging to $Variable$, or an interval between two landmarks written as $Land1..Land2$. QSIM repertoire of constraints corresponds to the following background predicates:

```
add( F1, F2, F3, Corr)      % F1 + F2 = F3
mult( F1, F2, F3, Corr)     % F1 * F2 = F3
minus( F1, F2, Corr)        % F1 = - F2
m_plus( F1, F2, Corr)       % Monotonicly increasing
m_minus( F1, F2, Corr)      % Monotonicly decreasing
deriv( F1, F2)              % F2 is time derivative of F1
```

*Corr* is a list of *corresponding values* that specify particular points of the relation between variables.

A qualitative model consists of a definition of the predicate

```
legalstate( F1, F2, ...)
```

where $Fi$ correspond to time variables of the dynamic system. The learning task consists of defining the predicate *legalstate* in the form:

```
legalstate( ...)  :-
  constraint1( ...),
  constraint2( ...),
  ...
```

where all the constraints are calls of constraint predicates.

# 3 Conversion of constraint predicates into a form accepted by GOLEM

GOLEM accepts definitions of background predicates in terms of ground facts. Therefore logical definitions of constraint predicates were compiled into tables of ground facts. In the experiment reported here, the following simplifications were done to keep the complexity within practical limits of GOLEM:

1. All lists of corresponding values were assumed empty.
2. Consistency of infinite values in constraints was ignored.
3. "mult" constraint was not compiled at all.

The *add* constraint requires very large number of ground facts. This number depends on the number of landmarks and corresponding values, and is in the order of several thousands of facts for each triple of variables. Therefore the *add* constraint was not tabulated explicitly. It was effectively replaced by three more economical relations: *norm_mag* (normalise given qualitative value with respect to a landmark), *lookup_consist_table* (lookup table for adding signs) and *verify_add_deriv* (lookup table for adding derivatives). The correspondence is:

```
add(F1:M1/D1, F2:M2/D2, F3:M3/D3, C) :-
  verify_add_inf_consistence(M1, M2, M3),    % Ignored in experiment
  verify_add_mag(F1, F2, F3, M1, M2, M3, C), % Add magnitudes
  verify_add_der(D1, D2, D3).                % Add derivatives

verify_add_mag(F1, F2, F3, M1, M2, M3, []):-
  norm_mag(F1, M1, 0, A1),     % Normalise magnitude M1 w.r.t. 0
  norm_mag(F2, M2, 0, A2),
  norm_mag(F3, M3, 0, A3),
  lookup_consist_table(A1, A2, A3).
```

The following constraint predicates were thus tabulated:

```
range( F, Range)
deriv( F1, F2)
m_plus( F1, F2, [])
m_minus( F1, F2, [])
minus( F1, F2, [])
norm_mag( FunName, QValue, 0, NormalisedQValue)  % Normalise
lookup_consist_table( NormV1, NormV2, NormV3) % Add norm. val.
verify_add_deriv( Dir1, Dir2, Dir3)     % Dir1 + Dir2 = Dir3
```

# 4 Learning U-tube

We will now describe an experiment with learning of the U-tube system (Figure 1). A usual QSIM-type model of this system can be in our Horn clause notation written as:

```
legalstate( La, Lb, Fab, Fba) :-
  add(Lb, Diff, La, [ c(lb0,d0,la0)]),
   % Correspond. lb0 + d0 = la0
  m_plus( Diff, Fab, [ c(0,0), c(d0,fab0)]),
  minus(Fab, Fba, [ c(mf0,fab0)]),
  deriv(La, Fba),
  deriv(Lb, Fab).
```

The *legalstate* relation can be adequately expressed by the tabulated predicates by substituting the *add* constraint with its definition in terms of other tabulated predicates.

The qualitative states shown in the example behaviour of Figure 2 were used as positive examples for learning. There are four states in Figure 2. As two of them are equal, we have in fact only three positive examples. We added six negative examples. A model induced by GOLEM from these 9 examples was:

```
legalstate( la: A/B, lb: C/D, fab: E/B, fba: F/D)  :-
  deriv( la: A/B, fba: F/D),
  deriv( lb: C/D, fab: E/B),
  minus( la: A/B, lb: G/D, []),
  minus( la: G/B, lb: C/D, []).
```

GOLEM constructed this model using 9 examples and 5408 background atoms representing the necessary atoms from QSIM theory. The model was constructed in 17 seconds on a Sun SPARCStation/330. As an alternative, GOLEM also found a model similar to this with the only difference that the last two constraints were replaced as follows:

```
...
minus( fab: G/B, lb: C/D, []),
minus( fab: G/D, la: A/B, []).
```

The interesting question now is whether the induced model is correct and whether it offers some useful interpretation from the physics point of view. The following analysis shows that both induced models are in fact correct and equivalent to the usual model of U-tube. First, it is easy to see that both induced models are equivalent, and therefore it suffices to only study the first one. In the sequel, this model will be called GolemTube, and the usual tube model will be called StandardTube.

It should be observed first that in the induced GolemTube clause there are some unexpected terms. For example, in addition to

```
la: A/B
```

we also have

```
la: G/B
```

The first of these two terms appears in the head of the clause and corresponds to the level in container A. On the other hand, the second term, although containing atom *la*, does not really correspond to the level in container A: it in effect introduces a new variable whose sign of derivative is equal to that of level A. GOLEM just "borrowed" the name *la* for this new variable. There was no other way for GOLEM to introduce a new variable because only these names appear in the tabulated background relations.

One way of demonstrating that GolemTube is equivalent to StandardTube, is to show that:

1. GolemTube is not overconstrained, and
2. GolemTube is not underconstrained.

We will justify statement (1) by justifying each of the constraints in GolemTube from the point of view of the physics of the modelled system. First, the two *deriv* constraints are obviously correct. Next, the two *minus* constraints both together say the following: $La$ and $Lb$ always have the same sign, but opposite direction of change (directions of change of $La$ an $Lb$ are in the induced model denoted by $B$ and $D$). This is also correct because both $La$ and $Lb$ are non-negative, and whenever $La$ increases $Lb$ must decrease and vice versa. GOLEM thus in a way found that the total amount of water in the system is constant. Note, however, that GOLEM only found a 'weak' way of stating this. Instead of saying $La + Lb = const.$, GolemTube only says that

$$[\frac{d}{dt}La] + [\frac{d}{dt}Lb] = 0$$

where $[x]$ denotes the sign of $x$. In addition to these constraints, the repeated occurrences of $B$ and $D$ in the head of GolemTube clause indicate two other constraints: (a) $La$ and $Fab$ have the same direction of change, and (b) $Lb$ and $Fba$ have the same direction of change. This is again easy to justify in terms of the physics: whenever $La$ increases, $Lb$ must decrease and the difference between the two levels must also increase, and therefore $Fab$ must increase.

Thus we can conclude that all the constraints in GolemTube are justified by the physics of the system, and therefore the GolemTube model is not overconstrained.

It remains to show that GolemTube is not underconstrained. This can be done by demonstrating that each constraint in NormalTube logically follows from the constraints in GolemTube. First, the two *deriv* constraints appear explicitly in both StandardTube and GolemTube. Of the remaining StandardTube constraints it is rather straightforward to show that GolemTube implies the following two:

```
add( Lb, Diff, La)     % Diff = La - Lb
m_plus( Diff, Fab)
```

That is: $Fab$ monotonically increases with the difference $La - Lb$, written shortly as $m\_plus(La - Lb, Fab)$. For simplicity we have omitted the corre-

sponding values. In GolemTube we have that the $La$ and $Fab$ have the same direction of change, and so have $Lb$ and $Fba$. This is equivalent to:

```
m_plus( La, Fab)
m_plus( Lb, Fba)
```

We also have that $La$ and $Lb$ have opposite direction of change, that is:

```
m_minus( La, Lb)
```

From this it follows:

```
m_plus( La, -Lb)
```

and

```
m_plus( La, La - Lb)
```

From this and *m_plus( La, Fab)* we have:

```
m_plus( La - Lb, Fab)
```

The only remaining StandardTube constraint now is *minus( Fab, Fba)*. In simulation, the two *minus* constraints in GolemTube seem to produce the same effect, although it remains unclear how to formally show that the StandardTube *minus* constraint in fact logically follows from the GolemTube constraints. GolemTube was exhaustively tested on all possible behaviours. It was found to be complete and correct with respect to StandardTube.

It should be noted that the analysis above only takes into account constraints with empty corresponding values lists. In relation to this, the corresponding qualitative simulation is not expected to generate new landmark values and new corresponding values.

# 5 Discussion and comparison

In this paper, a technique for inducing models of dynamic systems was presented. The approach relies on the QSIM formalism for qualitative simulation, and learning in the logic framework. So the described technique is an application of "inductive logic programming" (Muggleton 1990) where the GOLEM learning program (Muggleton and Feng 1990) was used.

As an example, we presented the induction of a model of the U-tube system. The model induced from 3 positive and 6 negative examples of the states of the system is correct and to a large extent offers natural interpretation from the point of view of the physics of the system. It may be surprising that it was possible to induce a correct model from such a small set of examples. On the other hand the success can be attributed to the background knowledge available to GOLEM in this exercise. The background knowledge consisted of 5408 ground facts. The relative efficiency of GOLEM compared to other logic induction programs was essential in coping with this amount of background knowledge. Also, it seems that GOLEM's style of generalisation is particularly suitable in this application. As reported by Sašo Džeroski, straightforward application of two other logic learning programs LINUS (Lavrač and Džeroski 1990) and FOIL (Quinlan 1989) to the same problem was not successful. FOIL was impeded by the particular heuristic it uses, while LINUS was too limited by its inability to introduce new variables into the induced formulas.

Another approach to learning qualitative models, also using logic, is (Mozetič 1987a; 1987b), also described in (Bratko, Mozetič and Lavrač 1989). Exploiting abstraction hierarchy, Mozetič obtained impressive experimental results. His approach, however, cannot be directly compared to ours because in his approach the observed system was viewed as static and thus completely different predicates were used as background knowledge.

Another interesting technique for the induction of qualitative models of dynamic systems was developed by Coiera (1989). His program, called GENMODEL, also finds a model in the form of a Horn clause. The program also performs a special kind of least general generalisation where the search is from specific to general descriptions. Coiera (1989) reports that GENMODEL, applied to the U-tube problem, generated a model consisting of 16 constraints when only 6 positive examples were used. In Coiera's approach, the search for qualitative constraints is more restricted so that the magni-

tude and derivative of a dynamic variable always both occur in the same term and cannot be mixed with other landmarks. This restriction has the advantage that it constrains search, and that the physics interpretation of induced models is easier. On the other hand, the learning program, being more constrained, cannot discover a fundamentally new formulation, and interesting new laws may be impossible to represent. Also, a major limitation of GENMODEL is that it does not introduce new variables and the user has to specify explicitly all the variables, or intermediate terms, that may occur in the induced model. As a consequence of this limitation, the U-tube model found by GOLEM cannot be generated by GENMODEL.

**Acknowledgements.** This project was sponsored by the Research Council of Slovenia and the Esprit 2 project Ecoles. Stephen Muggleton is supported by an SERC postdoctoral fellowship.

# 6 References

Bratko, I. (1989) Pole balancing: a study in qualitative reasoning about control. ISSEK Workshop 89, Udine, Italy, September 1989.

Bratko, I., Mozetič, I., Lavrač, N. (1989) *KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems.* MIT Press.

Coiera, E. (1989) Generating qualitative models from example behaviours. DCS Report No. 8901, School of Electr. Eng. and Computer Sc., Univ. of New South Wales, Sydney, Australia.

Džeroski, S. (1990) Personal communication.

de Kleer, J., Brown, J.S. (1986) A qualitative physics based on confluences. *Artificial Intelligence 28*, pp. 127-162.

Falkenheiner, B., Forbus, K. (1990) Self-explanatory simulations: an integration of qualitative and quantitative knowledge. 4th Int. Workshop on Qualitative Physics, Lugano, July 1990.

Kuipers, B.J. (1986) Qualitative simulation. *Artificial Intelligence 29*, pp. 289-338.

Lavrač, N., Džeroski, S. (1990) The use of background and meta-level knowledge in similarity-based learning. J. Stefan Institute: IJS-DP-5810, Ljubljana.

Mozetič, I. (1987a) Learning of qualitative models. In Bratko, I., Lavrač, N. (eds.) *Progress in Machine Learning*, Wilmslow, England: Sigma Press.

Mozetič, I. (1987b) The role of abstractions in learning qualitative models. *Proc. Fourth Int. Workshop on Machine Learning*, Irvine, CA (Morgan Kaufmann).

Muggleton, S.H. (1990) Inductive logic programming. *Proc. First Conf. on Algorithmic Learning Theory*, Tokyo, Sept. 1990 (Tokyo: Omsha).

Muggleton, S.H., Buntine, W. (1988) Machine invention of first-order predicates by inverting resolution. *Proc. Fifth Int. Conf. on Machine Learning*, pp. 339-352 (Morgan Kaufmann).

Muggleton, S.H., Feng, C. (1990) Efficient induction of logic programs. *Proc. First Conf. on Algorithmic Learning Theory*, Tokyo, Sept. 1990 (Tokyo: Omsha).

Quinlan, J.R. (1989) Learning logical definitions from relations. ISSEK Workshop 89, Udine, Italy, Sept. 1989.

Williams, B. (1990) Interaction-based invention: designing devices from first principles. 4th Int. Workshop on Qualitative Physics, Lugano, July 1990.

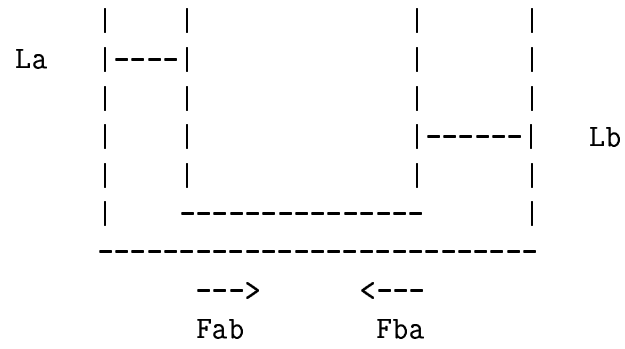


Figure 1 Two connected containers: the U-tube system.

| Time | La | Lb | Fab | Fba |
|---|---|---|---|---|
| t0 | la0/dec | lb0/inc | fab0/dec | fba0/inc |
| (t0,t1) | 0..la0/dec | 0..lb0/inc | 0..fab0/dec | fba0..0/inc |
| t1 | 0..la0/std | 0..lb0/std | 0/std | 0/std |
| (t1,inf) | 0..la0/std | 0..lb0/std | 0/std | 0/std |

Figure 2 Trace of behaviour of the system in Figure 1.