LEARNING QUALITATIVE MODELS WITH INDUCTIVE LOGIC PROGRAMMING

INFORMATICA 4/92

Keywords: machine learning, qualitative modelling, qualitative simulation, inductive logic programming

Sašo Džeroski
Institut Jožef Stefan

Qualitative models can be used instead of traditional numerical models in a wide range of tasks. These tasks include diagnosis, generating explanations of the system's behaviour and designing novel devices from first principles. Also, qualitative models are in some cases sufficient for the synthesis of control rules for dynamic systems. An important task in the theory of dynamic systems is the problem of identification of a model that explains given examples of system behaviour. This task can be formulated as a machine learning task of inducing a hypothesis that explains given examples. As the induced hypothesis (model) has to capture relations among the parameters of the observed system, we have to use an inductive tool for learning relations, i.e., an inductive logic programming system. In this paper we describe the application of the inductive logic programming system mFOIL to the problem of learning a qualitative model of the connected-containers dynamic system.

Učenje kvalitativnih modelov z induktivnim logičnim programiranjem

Kvalitativne modele lahko uporabimo za reševanje različnih nalog, npr. za diagnostiko, generiranje razlage obnašanja dinamičnega sistema ter načrtovanje naprav iz osnovnih načel delovanja. V nekaterih primerih zadošča kvalitativni model tudi za sintezo pravil vodenja dinamičnega sistema. Pomemben problem v teoriji dinamičnih sistemov je problem identifikacije modela, ki razloži znane primere obnašanja sistema. Omenjeni problem lahko formuliramo kot problem avtomatskega učenja kjer je treba generirati hipotezo, ki razloži podane primere. Glede na to, da sestoji model iz relacij med parametri sistema, uporabimo za reševanje problema sistem za avtomatsko učenje relacij oz. induktivno logično programiranje. V članku je opisana uporaba sistema za induktivno logično programiranje mFOIL pri problemu učenja kvalitativnega modela sistema povezanih posod.

1 Introduction

Qualitative models can be used instead of traditional numerical models in a wide range of tasks [Bratko 1991]. These tasks include diagnosis (e.g., [Bratko et al. 1989]), generating explanations of the system's behaviour (e.g., [Falkenheiner and Forbus 1990]) and designing novel devices from first principles (e.g., [Williams 1990]). Bratko [1991] conjectures that qualitative models are sufficient for the synthesis of control rules for dynamic systems, and supports this conjecture with an example.

Among several established formalisms for defining qualitative models of dynamic systems, the most widely known are qualitative differential equations called confluences [De Kleer and Brown 1984], Qualitative Process Theory [Forbus 1984] and QSIM [Kuipers 1986]. In this paper, we will adopt the QSIM (Qualitative SIMulation) formalism, as it has been already used for learning qualitative models.

A fundamental problem in the theory of dynamic systems is the identification problem, defined as follows [Bratko 1991]: given examples of the behaviour of a dynamic system, find a model that explains these examples. Motivated by the hypothesis that it should be easier to learn qualitative than quantitative models, [Bratko et al. 1992] have recently formulated the identification problem for QSIM models as a machine learning problem. Formulated in this framework, the task of learning QSIM-type qualitative models is as follows: given $QSIMtheory$ and $ExamplesOfBehaviour$, find a $QualitativeModel$, such that $QSIMtheory$ and $QualitativeModel$ explain the $ExamplesOfBehaviour$, or formally,

$$QSIMtheory \wedge QualitativeModel \models ExamplesOfBehaviour.$$

The identification task can be formulated as a machine learning task. Namely, the task of inductive machine learning is to find a hypothesis that explains a set of given examples. In some cases the learner can also make use of existing background knowledge about the given examples and the domain at hand. So, the learning task can be formulated as follows: given background knowledge $\mathcal{B}$ and examples $\mathcal{E}$, find a hypothesis $\mathcal{H}$, such that $\mathcal{B}$ and $\mathcal{H}$ explain $\mathcal{E}$, i.e., $\mathcal{B} \wedge \mathcal{H} \models \mathcal{E}$. We can see that $ExamplesOfBehaviour$ correspond to $\mathcal{E}$, $QSIMtheory$ corresponds to $\mathcal{B}$ and the target $QualitativeModel$ to $\mathcal{H}$.

As a qualitative model consists of relations among the parameters of the modelled system, we have to use an inductive system for learning relations. Systems that learn relations from examples and relational background knowledge, represented as a logic program, have been recently called inductive logic programming (ILP) systems [Muggleton 1992]. Bratko et al. [1992] describe the application of the inductive logic programming system GOLEM [Muggleton and Feng 1990] to the problem of learning a qualitative model of the dynamic system of connected containers, usually referred to as the U-tube system. There have been, however, several problems with the application of GOLEM to this task, stemming from the inability of GOLEM to use non-ground and non-determinate background knowledge.

In the paper, we describe the application of the inductive logic programming system mFOIL [Džeroski 1991], which can use non-ground background knowledge, to the same task [Džeroski and Bratko 1992]. A brief introduction to inductive logic programming is first given, followed by an outline of the main features of mFOIL. We proceed with an overview of the QSIM formalism and illustrate its use on the connected-containers (U-tube) system. The experiments and results of learning a qualitative model of the U-tube system with mFOIL are next presented, followed by a discussion of related work. Finally, we conclude with some directions for further work.

2 Inductive logic programming

In this section we introduce the field of machine learning of relations, or, as it has been recently called, inductive logic programming (ILP). We first mention some systems for learning relations, define the task of empirical inductive logic programming and illustrate it on a simple example. We then briefly outline some features of the ILP system mFOIL, which was used in our experiments in learning qualitative models.

Various logical formalisms have been used in inductive learning systems to represent examples and concept descriptions. These formalisms are similar to the formalisms for representing knowledge in general. Several widely known inductive learning systems, such as ID3 [Quinlan 1986] and AQ [Michalski 1983] use propositional languages to represent examples (objects) and concepts. In both cases objects are represented as tuples of attribute values, i.e, in terms of their global features. To represent concepts, decision trees are used in ID3 and if-then rules in AQ.

Another class of learning systems induce descriptions of relations (definitions of predicates). In these systems, objects are described structurally, i.e., in terms of their components and the relations between them. Training examples are represented by tuples of their components, while the relations between components belong to background knowledge. The languages used to represent examples, background knowledge and concept descriptions are typically subsets of first-order logic (logic programs). In this case, learning is in

fact logic program [1] synthesis and has recently been named *inductive logic programming (ILP)* [Muggleton 1991, Muggleton 1992].

Two different approaches can be distinguished in the ILP paradigm [De Raedt 1992]: the interactive and the empirical ILP approach. *Interactive ILP* systems include MIS [Shapiro 1983], MARVIN [Sammut and Banerji 1986] and CLINT [De Raedt 1992] as well as CIGOL [Muggleton and Buntine 1988] and other approaches based on inverting resolution [Rouveirol 1991, Wirth 1989]. These systems typically learn definitions of multiple predicates from a small set of examples and queries to the user.

Empirical ILP, on the other hand, is typically concerned with learning a definition of a single predicate from a large collection of examples. This class of ILP systems includes FOIL [Quinlan 1990], mFOIL [Džeroski 1991], GOLEM [Muggleton and Feng 1990] and LINUS [Lavrač et al. 1991]. LINUS, FOIL and mFOIL upgrade attribute-value learners from the ID3 and AQ family towards a first-order logic framework. A different approach is used in GOLEM which is based on Plotkin's notion of relative least general generalization (*rlgg*) [Plotkin 1969].

Empirical ILP systems are more likely to be applied in practice for two reasons. First, there is more experience with learning single concepts from large collections of data than with deriving knowledge bases from a small number of examples. Second, empirical ILP systems are much more efficient because of the use of heuristics, because there is no need to take into account dependencies among different concepts, and because no examples are generated [De Raedt and Bruynooghe 1992]. In fact, they are already efficient enough to be applied to real-life domains [Bratko 1992]. Several applications have been reported, including learning qualitative models from example behaviours [Bratko et al. 1992] [Džeroski and Bratko 1992], inducing temporal rules for satellite fault diagnosis [Feng 1991], learning to predict protein secondary structure [Muggleton et al. 1992] and learning rules for finite element mesh design [Dolšak and Muggleton 1992, Džeroski and Dolšak 1991].

[1] For an introduction to logic programming we refer the reader to [Bratko 1990]. A detailed theoretical treatment of the subject is given in [Lloyd 1987].

Empirical ILP

The task of empirical ILP, which is concerned with learning a single predicate, can be formulated as follows.

Given:

- a set of training examples $\mathcal{E}$, consisting of true $\mathcal{E}^+$ and false $\mathcal{E}^-$ facts of an unknown predicate p,
- a description language $\mathcal{L}$, specifying syntactic restrictions on the definition of predicate p,
- background knowledge $\mathcal{B}$, defining predicates q_i (other than p) which may be used in the definition of p and which provide additional information about the arguments of the examples of predicate p,

Find:

- a definition $\mathcal{H}$ for p, expressed in $\mathcal{L}$, such that $\mathcal{H}$ is complete, i.e., $\forall e \in \mathcal{E}^+ : \mathcal{B} \wedge \mathcal{H} \models e$, and consistent with respect to the examples, i.e., $\forall e \in \mathcal{E}^- : \mathcal{B} \wedge \mathcal{H} \not\models e$.

The true facts $\mathcal{E}^+$ are called *positive examples*, the false facts $\mathcal{E}^-$ are called *negative examples* and the hypothesis $\mathcal{H}$, i.e., the definition of p, is usually called the definition of the *target* predicate. When learning from noisy examples, the completeness and consistency criteria need to be relaxed in order to avoid overly specific hypotheses.

The ILP system mFOIL

The ILP system mFOIL [Džeroski 1991] is largely based on the FOIL [Quinlan 1990] approach. We thus briefly describe FOIL first and then outline some of the key features of mFOIL.

FOIL extends some ideas from attribute-value learning algorithms to the ILP paradigm. In particular, it uses a covering approach similar to AQ's [Michalski 1983] and an information based search heuristic similar to ID3's [Quinlan 1986]. The hypothesis language $\mathcal{L}$ in FOIL is the language of function-free program clauses, which means that no constants or terms other than variables may appear in the induced clauses. Function-free ground facts (relational tuples) are used to represent both training examples and background knowledge.

After the pre-processing of the training set, which consists of generating negative examples if none are given, the outermost loop of the FOIL algorithm repeats the following two steps untill all positive facts are covered:

- find a clause that covers some positive and no negative facts,
- remove the facts covered by this clause from the training set.

Finding a clause consists of a number of refinement steps. The search starts with the clause with empty body. At each step, the clause c built so far is refined by adding a literal to its body. These literals are positive or negative atoms of the form $X_i = X_j$ or $q_k(Y_1, Y_2, ..., Y_{n_k})$, where the X's appear in c, the Y's either appear in c or may be new variables and q_k is a relation (predicate) from the background knowledge or the target predicate p itself.

To stop the search for literals to be added to a clause, FOIL employs the *encoding length restriction*, which limits the number of bits used to encode a clause to the number of bits needed to explicitly indicate the positive examples covered by it. The construction of a clause is stopped when it covers only positive examples (is consistent) or when no more bits are available for adding literals to its body. The search for clauses stops when no new clause can be constructed under the encoding length restriction, or alternatively, when all positive examples are covered. One should be aware, however, that there are several problems with the encoding length restriction that actually degrade FOIL's performance on both noisy and non-noisy data as shown in [Džeroski and Lavrač 1991].

The most important differences between mFOIL and FOIL are related to the noise-handling mechanism used. mFOIL uses Bayesian probability estimates, namely the Laplace and the m-estimate [Cestnik 1990], of expected clause accuracy as search heuristics. These estimates have been successfully used in a similar way in propositional learning systems [Clark and Boswell 1991, Džeroski et al. 1992]. mFOIL also uses significance tests, similar to the ones in [Clark and Boswell 1991]. It achieved better results than FOIL on a test domain with artificially added noise and on a real-life domain of learning rules for finite element mesh design [Džeroski 1991, Džeroski and Bratko 1992].

Another key difference is the capability of mFOIL to use background knowledge which may contain rules and not ground facts only. This feature is especially important for learning qualitative models, since the *QSIMtheory* consists of rules and not ground facts only.

Other differences between FOIL and mFOIL are related to the search strategy and the space of possible hypotheses. As opposed to the hill-climbing search in FOIL, where only the best partially built clause is kept, mFOIL uses beam search and keeps several most promising clauses (the beam), which are refined gradually. mFOIL can also use information about the background knowledge, such as symmetry of predicates and types of predicate arguments, to reduce the space of possible hypotheses.

3 Qualitative modelling

In this section, we first introduce the QSIM [Kuipers 1986] formalism and then illustrate it on the U-tube system. We also describe how the QSIM theory can be formulated in logic [Bratko et al. 1992], so that it can be used as background knowledge in the process of learning qualitative models from examples.

The QSIM formalism

In the theory of dynamic systems, a physical system is represented by a set of continuous variables, which may change over time. Sets of differential equations, relating the system variables, are typically used to model dynamic systems numerically. Given a model (a set of differential equations) of the system and its initial state, the behaviour of the system can be predicted by applying a numerical solver to the set of differential equations.

A similar approach is taken in qualitative simulation [Kuipers 1986]. In QSIM, a physical system is described by a set of variables representing the *physical parameters* of the system (continuously differentiable real-valued functions) and a set of *constraint equations* describing how those parameters are related to each other. In this case, a (qualitative) model is a set of constraint equations. Given a qualitative model and a qualitative initial state of the system, the QSIM simula-

tion algorithm [Kuipers 1986] produces a directed graph consisting of possible future states and the immediate successor relation between the states. Paths in this graph starting from the initial state correspond to behaviours of the system.

The value of a physical parameter is specified qualitatively in terms of its relationship with a totally ordered set of *landmark values*. The *qualitative state* of a parameter consists of its value and direction of change. The direction of change can be *inc* (increasing), *std* (steady) and *dec* (decreasing). Time is represented as a totally ordered set of symbolic distinguished time points. The current time is either at or between distinguished time-points. At a distinguished time-point, if several physical parameters linked by a single constraint are equal to landmark values, they are said to have *corresponding values.*

The constraints used in QSIM are designed to permit a large class of differential equations to be mapped straightforwardly into qualitative constraint equations. They include mathematical relationships, such as $deriv(Velocity, Acceleration)$ and $mult(Mass, Acceleration, Force)$. In addition, constraints like $M^+(Price, Power)$ and $M^-(Current, Resistance)$ state that there is a monotonically increasing/decreasing functional relationship between two physical parameters, but do not specify the relationship completely.

The U-tube system

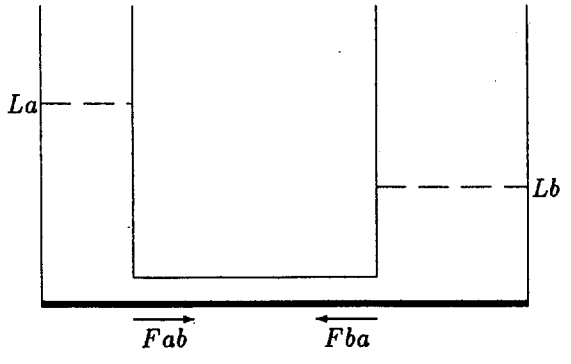

Figure 1: The U-tube system.

Let us illustrate the above notions on the connected-containers (U-tube) example, adapted from [Bratko et al. 1992]. The U-tube system (illustrated in Figure 1) consists of two containers, A and B, connected with a pipe and filled with water to the corresponding levels La and Lb. Let the flow from A to B be denoted by Fab, the flow from B to A by Fba. The variables La,Lb,Fab and Fba are the parameters of the system.

The flows Fab and Fba are the time derivatives of the water levels Lb and La, respectively, and run in opposite directions. Let the difference in the levels of the containers A and B be $Diff = La - Lb$. The pressure $Press$ along the pipe influences the flow Fab: the higher the pressure, the greater the flow. A similar dependence exists between the level difference and the pressure. The above constraints can be formulated in QSIM as follows:

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

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

$$Fab = -Fba$$

$$Diff = La - Lb$$

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

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

If we are not explicitly interested in the pressure, the last two qualitative constraint equations can be simplified into one:

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

For comparison, in a numerical model, the last two equations might have the form

$$Press = c_1 \cdot Diff$$

$$Fab = c_2 \cdot Press$$

or, when simplified

$$Fab = c \cdot Diff$$

where c, c_1 and c_2 are positive constants. In this case, the relationship between the variables Fab, $Press$ and $Diff$ is completely, and not only qualitatively, specified given the values of c, c_1 and c_2.

The landmark values for the variables of this model for the U-tube, ordered left to right, are as follows:

$$La : minf, 0, la0, inf$$
$$Lb : minf, 0, lb0, inf$$
$$Fab : minf, 0, fab0, inf$$
$$Fba : minf, fba0, 0, inf$$

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

Table 1: Qualitative behaviour of the U-tube system.

These values are symbolic names corresponding to minus infinity, zero, infinity and the initial values of the four variables. The left-to-right ordering corresponds to the *less than* relation between the corresponding numerical values.

The QSIM simulation of the U-tube system produces the trace given in Table 1. From the trace we can see, for example, that in the initial state the value of the level La is equal to $la0$ and is decreasing (dec). This is represented as $La = la0/dec$. In the time interval that follows, La is between 0 and $la0$ and decreasing, which is written as $La = 0..la0/dec$.

Formulating QSIM in logic

Bratko et al. [1992] translate the QSIM approach to qualitative simulation into a logic programming formalism (pure Prolog). A sketch in Prolog of the QSIM qualitative simulation algorithm is given below.

$simulate(State) \leftarrow$
    $transition(State, NextState)$,
    $simulate(NextState)$.

$transition(state(V1,\ldots), state(NewV1,\ldots)) \leftarrow$
    $trans(V1, NewV1)$, $\%Model-independent$
    $\ldots$,
    $legalstate(NewV1,\ldots)$. $\%Model-dependent$

The simulation starts from the initial qualitative *State*, consisting of the qualitative values and directions of change of the system parameters. The simulator first finds a possible transition to a $NewState$ and then continues the simulation from the new state. The relation *trans* is a non-deterministic relation that generates possible transitions of the system parameters, i.e., possible new values for them. It is defined as part of the QSIM theory [Kuipers 1986]. The model of a particular system is defined by the predicate *legalstate* which imposes constraints on the values of the system parameters. The definition of this predicate is of the following form:

$legalstate(\ldots) \leftarrow$
    $constraint1(\ldots)$,
    $constraint2(\ldots)$,
    $\ldots$

where the constraints are part of the QSIM theory. Under continuity assumptions, the problem of learning the legality of states is equivalent to the problem of learning the dynamics of the system.

The following Prolog predicates correspond to the QSIM constraints:

$add(F1,F2,F3,Corr)$	$\%F1 + F2 = F3$
$mult(F1,F2,F3,Corr)$	$\%F1 * F2 = F3$
$minus(F1,F2,Corr)$	$\%F1 = -F2$
$m\_plus(F1,F2,Corr)$	$\%F2 = M^+(F1)$
$m\_minus(F1,F2,Corr)$	$\%F2 = M^-(F1)$
$deriv(F1,F2)$	$\%F2 = dF1/dt$

In the above $F1, F2$ and $F3$ stand for system parameters and $Corr$ stands for a list of corresponding values.

The qualitative model for the U-tube system can be written in Prolog notation as follows:

$legalstate(La, Lb, Fab, Fba) \leftarrow$
    $add(Lb, Diff, La, [c(lb0, d0, la0)])$,
    $m\_plus(Diff, Fab, [c(0,0), c(d0, fab0)])$,
    $minus(Fab, Fba, [c(fab0, fba0)])$,
    $deriv(La, Fba)$,
    $deriv(Lb, Fab)$.

where $c(x,y,z)$ means that x, y and z are corresponding values for the constraint. For example, in the *add* constraint, $c(lb0, d0, la0)$ means that $lb0 + d0 = la0$.

If we are not interested in the $Diff$ parameter, the first two constraints can be replaced by the constraint $add(Lb, Fab, La, [c(lb0, fab0, la0)])$ or the symmetrical constraint $add(La, Fba, Lb, [c(la0, fba0, lb0)])$.

4 An experiment in learning qualitative models

In this section, we describe the application of the ILP system mFOIL to the problem of learning a qualitative model of the U-tube system [Džeroski and Bratko 1992]. We first describe in detail the experimental setup and then present the results generated by mFOIL, followed by a comparison with the results obtained by GOLEM and FOIL on the same problem.

Experimental setup

As mentioned earlier, when formulating the task of identification of models as a machine learning task, $ExamplesOfBehaviour$ become training examples and $QSIMtheory$ constitutes the background knowledge. The $QualitativeModel$ to be learned corresponds to the hypothesis to be induced by the machine learning system. In the following we describe in more detail the training examples and background knowledge used in our experiment. We also give the parameter settings for the inductive logic programming system mFOIL used in the experiment.

As the model of a system is defined by the predicate *legalstate*, the learning task is to induce a definition of this predicate. The behaviour trace of the U-tube system (Table 1) provides three positive training examples for the predicate *legalstate* (the last two states of the behaviour trace are equal). In addition, a positive example which corresponds to the case where there is no water in the containers is considered. The set of positive examples considered is given in Table 4.

Negative examples (illegal states) represent 'impossible' states which cannot appear in any behaviour of the system. For instance, the state $(la : la0/inc, lb : lb0/inc, fab : f0/dec, fba : mf0/inc)$ is a negative example, because it cannot happen that the water levels in both containers increase. Negative examples can be either hand-generated by an expert, or can be generated under the closed-world assumption, when all positive examples are known.

Bratko et al. [1992] used six hand-crafted negative examples, which only slightly differ from the positive ones. Such examples are called near misses. In a preliminary experiment, from the four positive examples in Table 4 and the six near misses mFOIL generated an overly general, i.e., underconstrained model. We thus used a larger set of 543 negative examples, randomly chosen by Žitnik [1991] from the complete set of negative examples generated under the closed-world assumption.

The QSIM theory, formulated in logic, serves as background knowledge. To reduce the complexity of the learning problem, the corresponding values argument $Corr$ is omitted from the constraints. The background knowledge thus consists of the predicates $add(F1, F2, F3)$, $mult(F1, F2, F3)$, $minus(F1, F2)$, $m\_plus(F1, F2)$, $m\_minus(F1, F2)$ and $deriv(F1, F2)$, which correspond to the QSIM constraint primitives with empty lists of corresponding values. For comparison with GOLEM [Bratko et al. 1992], the *mult* relation was excluded from the background knowledge. All arguments of the background predicates are of the same type; they are compound terms of the form $FuncName : QualValue/DirOfChange$.

As mFOIL allows for the use of non-ground background knowledge, we used directly the Prolog definitions of the background predicates and did not tabulate them as ground facts. All of the background predicates, except *deriv*, are symmetric. For example, $add(X, Y, Z)$ is equivalent to $add(Y, X, Z)$. The same holds for the predicate *mult*. Similarly, $minus(X, Y)$ is equivalent to $minus(Y, X)$. This reduces the space of models to be considered.

All arguments of the background knowledge predicates were considered input, i.e., only relations between the given system parameters (La, Lb, Fab, Fba) were considered. This is reasonable, as the correct qualitative model can be formulated in terms of these parameters and without introducing new variables. In some cases, however, the introduction of new variables is necessary. For example, if the U-tube system were described only in terms of La and Lb, the new variables Fab and Fba (or at least one of them) would be necessary for the construction of a qualitative model of the system. As mFOIL allows for the introduction of new variables, such a case could be, in principle, handled if necessary.

Finally, let us mention that the default search heuristic and stopping criteria were used in mFOIL. The Laplace estimate was used as a

$legalstate(la : la0/dec, lb : lb0/inc, fab : fab0/dec, fba : fba0/inc).$
$legalstate(la : 0..la0/dec, lb : lb0..inf/inc, fab : 0..fab0/dec, fba : fba0..0/inc).$
$legalstate(la : 0..la0/std, lb : lb0..inf/std, fab : 0/std, fba : 0/std).$
$legalstate(la : 0/std, lb : 0/std, fab : 0/std, fba : 0/std).$

Table 2: Positive examples for learning a qualitative model of the U-tube.

search heuristic and a significance level of 99 % was employed in the significance tests. The default beam width of 5 was increased to 20 in order to avoid getting stuck in local optima. Namely, if the beam width is one, beam search is actually hill-climbing search and is prone to getting stuck in local optima during the search for good models.

Results

Given the 4 positive, the 543 negative examples and the background knowledge as described above, mFOIL generated 20 different models, shown in Table 4. They are all evaluated as equally good by mFOIL as they correctly distinguish between the given positive and negative examples. In addition, they are of the same length, i.e., consist of four constraints each. However, not all of them are equivalent to the correct model.

Model # 6 is equivalent to the correct model, shown in Section 3, provided that corresponding values are ignored in the latter. The same holds for model # 16 [Žitnik, personal communication]. Out of the 194481 possible states, these models cover 130 states (among which 32 are physically possible, i.e., are positive examples). When all 32 positive examples and the same 543 negative examples were given to mFOIL, it was able to generate, among other models, the two models from Table 4 which are equivalent to the correct model, even with a beam of size 10 and the *mult* relation in the background knowledge.

An important issue arises from the above results, namely, the need for a criterion of quality for qualitative models other than the standard criteria used in machine learning. Considering the models from Table 4, all of which cover all positive and no negative examples, and all of which are of same length, this need is obvious. This problem could be reduced by imposing additional constraints on the models considered in the search process. However, additional semantic criteria may still be needed.

Comparison with other ILP systems

Bratko et al. [1992] applied GOLEM to the problem of learning of qualitative models in the QSIM formalism. They used the four positive examples from Table 4 and six hand-crafted negative examples (near misses). The model induced by GOLEM was shown to be dynamically equivalent to the correct model.

However, several modifications had to be done in order to apply GOLEM. As GOLEM accepts only ground facts as background knowledge, the Prolog definitions mentioned above had to be compiled into tables of ground facts. To reduce the complexity of the learning problem, the predicates were tabulated with empty lists of corresponding values (argument $Corr$). The *mult* constraint was not compiled at all. Finally, the *add* constraint had to be decomposed into several subconstraints in order to avoid the explosion of the number of ground facts generated.

Žitnik [1991] conducted further experiments in learning a qualitative model of the U-tube system, using both GOLEM and FOIL. She used several sets of examples, including the set of 4 positive examples and the set of 543 randomly generated negative examples used by mFOIL. Among the conclusions of her work, we would like to mention the following:

- The need to compile the background knowledge into ground facts for FOIL and GOLEM causes an explosion in the complexity of the learning problem, which has to be handled by decomposing predicates into simpler primitives.
- The absence of type information causes problems in interpreting the generalizations produced by FOIL and GOLEM.
- Top-down systems, such as FOIL, need a larger number of negative examples in order to prevent over-generalization.

#	Model: $legalstate(La, Lb, Fab, Fba) \leftarrow$
1	minus(Fab,Fba), add(Lb,Fab,La), m_minus(La,Fba), deriv(Fab,Fba)
2	minus(Fab,Fba), add(Lb,Fab,La), m_minus(La,Fba), deriv(Lb,Fab)
3	minus(Fab,Fba), add(Lb,Fab,La), m_minus(La,Fba), deriv(La,Fba)
4	minus(Fab,Fba), add(La,Fba,Lb), deriv(La,Fba), m_minus(Lb,Fab)
5	minus(Fab,Fba), add(La,Fba,Lb), deriv(La,Fba), m_plus(Lb,Fba)
6	**minus(Fab,Fba), add(La,Fba,Lb), deriv(Lb,Fab), deriv(La,Fba)**
7	minus(Fab,Fba), add(La,Fba,Lb), deriv(Fab,Fba), deriv(La,Fba)
8	minus(Fab,Fba), add(La,Fba,Lb), deriv(Fba,Fab), deriv(La,Fba)
9	minus(Fab,Fba), add(La,Fba,Lb), m_plus(La,Fab), deriv(Fba,Fab)
10	minus(Fab,Fba), add(La,Fba,Lb), m_plus(La,Fab), deriv(Fab,Fba)
11	minus(Fab,Fba), add(La,Fba,Lb), m_plus(La,Fab), deriv(Lb,Fab)
12	minus(Fab,Fba), add(La,Fba,Lb), m_plus(La,Fab), deriv(La,Fba)
13	minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Lb), deriv(Fba,Fab)
14	minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Lb), deriv(Fab,Fba)
15	minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Lb), deriv(Lb,Fab)
16	**minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Lb), deriv(La,Fba)**
17	minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Fba), deriv(Fba,Fab)
18	minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Fba), deriv(Fab,Fba)
19	minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Fba), deriv(Lb,Fab)
20	minus(Fab,Fba), add(La,Fba,Lb), m_minus(La,Fba), deriv(La,Fba)

Table 3: Qualitative models for the U-tube generated by mFOIL.

The following model [Žitnik 1991] was induced by GOLEM from the same examples as used by mFOIL and background knowledge as in [Bratko et al. 1992].

$$legalstate(la : A/B, lb : C/D, fab : E/B, fba : F/D) \leftarrow$$
$$deriv\_simplified(D, E),$$
$$legalstate(la : A/G, lb : C/H, fab : I/G, fba : J/H).$$

The condition $deriv\_simplified(D, E)$ actually means $deriv(Lb, Fab)$. This example illustrates the problem that a model induced by GOLEM can have several interpretations. This is due to the fact that GOLEM may introduce new variables, such as the ones in the term $fba : J/H$ in the model below, the meaning of which may be difficult to grasp.

Similar problems appear when using FOIL [Žitnik 1991]. Some of these problems are absent in LINUS, as it has typed variables, and can use non-ground background knowledge. However, LINUS cannot introduce new variables, which can prevent it from learning an appropriate model if such variables are needed.

None of the above problems appear in mFOIL, as it can use non-ground background knowledge and the typing of variables prevents unclear generalizations, while still having the possibility to introduce new variables. It should be noted, however, that new variables that may be introduced by the background knowledge predicates are likely to be non-discriminating and thus some kind of lookahead would be needed to treat them properly.

5 Related work

An early system for learning qualitative models named QuMAS, using a restricted form of logic, is described in [Mozetič 1987] and also in [Bratko et al. 1989]. The idea of QuMAS to transform the problem of learning in logic to propositional form was further developed in LINUS. QuMAS was used, however, to learn about a *static* system. It used a completely different set of primitives (background knowledge predicates) and is thus incomparable to the work on learning qualitative models of *dynamic* systems [Bratko et al. 1992].

GENMODEL [Coiera 1989] also generates a QSIM model in logic, using a special kind of least general generalization. Similarly to mFOIL, it uses a strong typing of variables, so that the value and direction of change always appear together and cannot be mixed (unlike FOIL and GOLEM where they can get mixed). It shares the limitation of LINUS, namely no new variables may be introduced. However, it takes into account the corresponding values in the constraints.

The approach taken in MISQ [Kraan et al. 1991] is essentially identical to GENMODEL, sharing most of GENMODEL's limitations, including the inability to introduce new variables. The useful additions include a facility of extracting qualitative behaviours out of quantitative data and generation of alternative maximally consistent models when incomplete information is given about the example behaviours. Furthermore, dimensional analysis is used to reduce the set of constraints generated.

Varšek [1991] applied a genetic algorithm QME (Qualitative Model Evolution) to the problem of learning qualitative models from examples. The corresponding values are not taken into account. Models are represented as trees in the genetic algorithm. The genetic operator of *crossover* exchanges subtrees between trees, while the *mutation* genetic operator generates random subtrees on single trees. Using a different training set, QME obtained five models equivalent to the correct one. In addition, QME was apllied to several other small domains, including a RC-circuit (resistor-capacitor-circuit). Similarly to GENMODEL, QME can not introduce new variables.

PAL [Morales 1992], originally designed to learn chess patterns, is also based on the idea of least general generalization. It can also use non-ground background knowledge. Using only the four positive examples and the same background knowledge as mFOIL, PAL induced a model equivalent to the correct one. The model contains several redundant constraints, as PAL uses no negative examples to reduce it. [Morales 1992] also successfully induced models of the U-tube system described with three (La, Lb, Fab) and five $(La, Lb, Fab, Fba, Diff)$ parameters. However, adding new system parameters requires additional effort and has to be handled in a special way in PAL. The *rlgg*-based systems PAL, GENMODEL and MISQ do not need negative examples.

6 Conclusion

We have successfully applied the inductive logic programming system mFOIL to the problem of learning a qualitative model of the connected-containers dynamic system. The ability of mFOIL to use non-ground background knowledge proved useful in this respect and advantageous as compared to GOLEM and FOIL. mFOIL produced many models which correctly distinguish between the given positive and negative examples. Two of these proved to be equivalent to the correct model. At the same time, however, this reveals the necessity for criteria other than completeness, consistency and length (complexity) to distinguish among different qualitative models. Although the number of different models may be reduced by using dimensional analysis, this does not avoid the need for suitable criteria (bias).

Another problem with learning qualitative models is the problem of introducing new variables. Although GOLEM, FOIL and mFOIL can introduce new variables, the literals that introduce these variables are typically both non-determinate (the new variables can have more than one value) and non-discriminating (the literals do not distinguish between positive and negative examples). For example, if variable X has a positive qualitative value and variable Y has a negative value, the literal $add(X, Y, Z)$, where Z is a new variable will be non-determinate, as Z can be either positive, zero or negative. However, if X, Y and Z are described numerically, then Z is uniquely determined given X and Y. This suggests that a LINUS-like approach [Lavrač and Džeroski 1992] operating on real-valued variables, where new variables are introduced before learning, coupled with the approach of learning qualitative models from real-valued data [Kraan et al. 1991], might be effective. It might also be possible to generate qualitative models directly from numerical data, without extracting qualitative behaviours.

Acknowledgements

This research was funded by the Slovenian Ministry of Science and Technology. mFOIL was developed as a part of my MSc thesis under the supervision of professor Ivan Bratko, who also commented on earlier versions of several parts of the paper. Many thanks to Nada Lavrač and Tanja Urbančič for their comments on the paper.

References

[Bratko 1990] Bratko, I. (1990). *Prolog Programming for Artificial Intelligence.* Addison-Wesley, Wokingham, 2 edition.

[Bratko 1991] Bratko, I. (1991). Qualitative modelling: learning and control. In *Proc. Inter-*

national Conference on Artificial Intelligence. Prague.

[Bratko 1992] Bratko, I. (1992). Applications of machine learning: Towards knowledge synthesis. In *Proc. International Conference on Fifth Generation Computer Systems*, pages 1207–1218. Tokyo.

[Bratko et al. 1989] Bratko, I., Mozetič, I., and Lavrač, N. (1989). *KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems.* MIT Press, Cambridge, MA.

[Bratko et al. 1992] Bratko, I., Muggleton, S., and Varšek, A. (1992). Learning qualitative models of dynamic systems. In Muggleton, S., editor, *Inductive Logic Programming*, pages 437–452. Academic Press, London.

[Cestnik 1990] Cestnik, B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proc. Ninth European Conference on Artificial Intelligence*, pages 147–149. Pitman, London.

[Clark and Boswell 1991] Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *Proc. Fifth European Working Session on Learning*, pages 151–163. Springer, Berlin.

[Coiera 1989] Coiera, E. (1989). Learning qualitative models from example behaviours. In *Proc. Third International Workshop on Qualitative Physics.* Stanford, California.

[De Kleer and Brown 1984] De Kleer, J. and Brown, J. (1984). A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83.

[De Raedt 1992] De Raedt, L. (1992). *Interactive Theory Revision: An Inductive Logic Programming Approach.* Academic Press, London.

[De Raedt and Bruynooghe 1992] De Raedt, L. and Bruynooghe, M. (1992). Interactive concept learning and constructive induction by analogy. *Machine Learning*, 8(2):107–150.

[Dolšak and Muggleton 1992] Dolšak, B. and Muggleton, S. (1992). The application of inductive logic programming to finite element mesh design. In Muggleton, S., editor, *Inductive Logic Programming*, pages 453–472. Academic Press, London.

[Džeroski 1991] Džeroski, S. (1991). Handling noise in inductive logic programming. Master's thesis, Faculty of Electrical Engineering and Computer Science, University of Ljubljana, Ljubljana, Slovenia.

[Džeroski and Bratko 1992] Džeroski, S. and Bratko, I. (1992). Handling noise in inductive logic programming. In *Proc. Second International Workshop on Inductive Logic Programming.* Tokyo, Japan. ICOT TM-1182.

[Džeroski et al. 1992] Džeroski, S., Cestnik, B., and Petrovski, I. (1992). The use of Bayesian probability estimates in rule induction. Technical Report TIRM-92-051, The Turing Institute, Glasgow, Scotland.

[Džeroski and Dolšak 1991] Džeroski, S. and Dolšak, B. (1991). A comparison of relation learning algorithms on the problem of finite element mesh design. In *Proc. XXVI Yugoslav Conference of the Society for ETAN.* Ohrid, Yugoslavia. In Slovenian.

[Džeroski and Lavrač 1991] Džeroski, S. and Lavrač, N. (1991). Learning relations from noisy examples: An empirical comparison of LINUS and FOIL. In *Proc. Eighth International Workshop on Machine Learning*, pages 399–402. Morgan Kaufmann, San Mateo, CA.

[Falkenheiner and Forbus 1990] Falkenheiner, B. and Forbus, K. (1990). Self-explanatory simulations: an integration of quantitative and qualitative knowledge. In *Proc. Fourth International Workshop on Qualitative Physics.* Lugano, Switzerland.

[Feng 1991] Feng, C. (1991). Inducing temporal fault diagnostic rules from a qualitative model. In *Proc. Eighth International Workshop on Machine Learning*, pages 403–406. Morgan Kaufmann, San Mateo, CA.

[Forbus 1984] Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence*, 24:85–168.

[Kraan et al. 1991] Kraan, I., Richards, B., and Kuipers, B. (1991). Automatic abduction of qualitative models. In *Proc. Fifth International Workshop on Qualitative Physics.* Austin, Texas.

[Kuipers 1986] Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence*, 29(3):289–338.

[Lavrač and Džeroski 1992] Lavrač, N. and Džeroski, S. (1992). Background knowledge and declarative bias in inductive concept learning. In Jantke, K., editor, *Proc. Third International Workshop on Analogical and Inductive Inference*. Springer, Berlin.

[Lavrač et al. 1991] Lavrač, N., Džeroski, S., and Grobelnik, M. (1991). Learning nonrecursive definitions of relations with LINUS. In *Proc. Fifth European Working Session on Learning*, pages 265–281. Springer, Berlin.

[Lloyd 1987] Lloyd, J. (1987). *Foundations of Logic Programming*. Springer, Berlin, 2 edition.

[Michalski 1983] Michalski, R. (1983). A theory and methodology of inductive learning. In R.S. Michalski, J. C. and Mitchel, T., editors, *Machine Learning: An Artificial Intelligence Approach*, volume I, pages 83–134. Tioga, Palo Alto, CA.

[Morales 1992] Morales, E. (1992). *First-order induction of patterns in chess*. PhD thesis, Department of Computer Science, University of Strathclyde, Glasgow, Scotland.

[Mozetič 1987] Mozetič, I. (1987). Learning of qualitative models. In Bratko, I. and c, N. L., editors, *Progress in Machine Learning*, pages 201–217. Sigma Press, Wilmslow, UK.

[Muggleton 1991] Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, 8(4):295–318.

[Muggleton 1992] Muggleton, S., editor (1992). *Inductive Logic Programming*. Academic Press, London.

[Muggleton and Buntine 1988] Muggleton, S. and Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. In *Proc. Fifth International Conference on Machine Learning*, pages 339–352. Morgan Kaufmann, San Mateo, CA.

[Muggleton and Feng 1990] Muggleton, S. and Feng, C. (1990). Efficient induction of logic programs. In *Proc. First Conference on Algorithmic Learning Theory*, pages 368–381. Ohmsha, Tokyo.

[Muggleton et al. 1992] Muggleton, S., King, R., and Sternberg, M. (1992). Protein secondary structure prediction using logic. In *Proc. Second International Workshop on Inductive Logic Programming*. Tokyo, Japan. ICOT TM-1182.

[Plotkin 1969] Plotkin, G. (1969). A note on inductive generalization. In Meltzer, B. and Michie, D., editors, *Machine Intelligence 5*, pages 153–163. Edinburgh University Press, Edinburgh.

[Quinlan 1986] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[Quinlan 1990] Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, 5(3):239–266.

[Rouveirol 1991] Rouveirol, C. (1991). Completeness for inductive procedures. In *Proc. Eighth International Workshop on Machine Learning*, pages 452–456. Morgan Kaufmann, San Mateo, CA.

[Sammut and Banerji 1986] Sammut, C. and Banerji, R. (1986). Learning concepts by asking questions. In R.S. Michalski, J. C. and Mitchel, T., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2, pages 167–191. Morgan Kaufmann, San Mateo, CA.

[Shapiro 1983] Shapiro, E. (1983). *Algorithmic Program Debugging*. MIT Press, Cambridge, MA.

[Varšek 1991] Varšek, A. (1991). Qualitative model evolution. In *Proc. Twelfth International Joint Conference on Artificial Intelligence*, pages 1311–1316. Morgan Kaufman, San Mateo, CA.

[Williams 1990] Williams, B. (1990). Interaction-based invention: designing devices from first principles. In *Proc. Fourth International Workshop on Qualitative Physics*. Lugano, Switzerland.

[Wirth 1989] Wirth, R. (1989). Completing logic programs by inverse resolution. In *Proc. Fourth European Working Session on Learning*, pages 239–250. Pitman, London.

[Žitnik 1991] Žitnik, K. (1991). Machine learning of qualitative models. Technical Report IJS-DP-6239, Jožef Stefan Institute, Ljubljana, Slovenia. In Slovenian.