A framework for learning constraints : Extended Abstract

Srinivas Padmanabhuni,Jia-Huai You
Department of Computing Science,
Univ Of Alberta, Canada T6G 2H1.
e-mail: srinivas,you@cs.ualberta.ca

Aditya Ghose,
School of Computing and Information Technology
Griffith University
Nathan Queensland 4111 Australia
aditya@cit.gu.edu.au

June 20, 1996

1 Motivation and Introduction

Machine Learning [Shavlik and Dietterich 90] is the subfield of artificial intelligence that studies the automated acquisition of knowledge as a result of experience. Learning is studied for a wide variety of reasons: to discover general principles of intelligence , to get a better understanding of human learning and to acquire domain-specific knowledge for performing a task better.

Machine learning research can be classified on various dimensions depending upon the context. The primary dimensions of learning we need to consider are representation of knowledge, the technique employed in learning and the domain of application.

We shall study the machine learning systems keeping in view one particular application domain, i.e. constraints. Before we look at the role of learning in constraints, we shall explore the area of constraints.

A *constraint* [Mackworth 92] refers to *relation that must be satisfied.* For example, the relation that all bodies falling under gravity must have the same acceleration is a *constraint.*

Since there exist a variety of real world problems, the variables being used in a problem can be from diverse domains. Similarly the constraints being used in the problem can also be of diverse types, e.g. integer variables with interval domains, real variables with continuous domain etc. Again diverse types of constraints involving the above variables are used in practice like arithmetic and order constraints, set constraints etc.

Constraint programming refers to process of solving the problems specified as sets of constraints. The most common method of specifying constraints in constraint programming applications is as a constraint satisfaction problem(CSP). A CSP [Mackworth 92]is defined by a set of variables, each of which have a set of possible values (their domain), and a set of constraints (relations) between these variables. The solution to a CSP is a set of variable-value assignments which satisfy the constraints.

As described earlier the domains can be of diverse types as can be constraints. But the pre-

dominant type of specification of CSP involves variables with discrete and finite domains. The CSP is then solved by use of certain techniques called *consistency* techniques in conjunction with backtracking.

Another common way of specification of constraints is the form of a *constraint logic program*. A *constraint logic program* is a generalization of a logic program, in the sense that the clauses can contain constraints in the body. A constraint logic program is a set of clauses with predicates in the head, but may contain constraints along with predicate symbols in the body of the clause. e.g. The CLP P = {p(Y) :- Y < 2, q(Y). ; a(Z) :- Z > 1 } is a two clause CLP. The semantics of a constraint logic program is similar to a logic program except that the process of unification as in logic programs is replaced by the process of constraint satisfaction to solve the constraints involved in the clauses.

There are situations in real life warranting the need for development of machine learning systems for constraints. Typically in any problem solving domain, the constraints are fully specified before a solution to the problem is attempted. But there are certain application domains where it might be difficult to obtain an explicit set of appropriate constraints to define a problem. In such a case a mechanism for automatically acquiring constraints becomes necessary.

Automatic constraint acquisition calls for development of learning mechanisms for constraints. As the constraint representations vary in the type of domain, a variety of machine learning systems can be developed for constraints.

Thus in this paper we develop a model for automatic constraint acquisition.

2 Constraints and learning

Empirical learning has been the principal source of systems capable of learning symbolic knowledge, the common form of knowledge in constraints. In these systems, inductive reasoning is performed on externally supplied examples to produce general rules. In any such system, the learner is fed a series of characterizing examples classified into two sets of instances: positive and negative examples. The objective is to produce a generalized description which is able to explain all the positive instances and yet not explain any of the negative instances. Based on the methods involved in the obtaining of the generalized description, the inductive learning systems[Wu 95] are broadly classified into the following four categories:

Attribute based induction The purpose of such systems is to produce a generalized description as decision trees or control rules from a set of positive and negative relational tuples as input so that all the input positive examples are covered and is consistent with the negative examples. The correctness of the learned description is tested by a sample *test set* from among the same relational base.

Incremental Generalization and Specialization Here background knowledge is specified as a hierarchy to choose between different descriptions. The descriptions are generalized and specialized according to this hierarchy to decide the final form of the learned description.

Unsupervised Concept Formation The learner finds a generalized description from the input examples by use of clustering similar data into a concept, and giving in the end a concept a description satisfying certain description like Minimum description Length or some similar measure. This technique is most popular in data mining systems.

Inductive Logic Programming [Muggleton 92] The most recent class of machine learning algorithms based on use of first-order logic as underlying knowledge representation to generate descriptions. Inductive Logic programming systems use a combination of all the above three techniques in conjunction with background knowledge in the learning process. Inductive Logic programming systems use a combination of all t he three systems above in the learning process.

As seen from above the most recent work in the area of learning is in learning first-order clauses or logic programs to give the stream of "Inductive Logic Programming". Logic programs are more expressive than any propositional knowledge representation scheme. Hence the field of Inductive Logic Programming has been applied in a variety of daily life applications. Also as seen from the definition in the earlier section, we find that constraint logic programs are even more expressive than logic programs because of incorporation of the concept of constraints in them. Thus it would be worthwhile to explore if ideas from inductive logic programming can be exported or extended to Constraint Logic Programs.

But in a constraint logic program, the process of deductive inference of logic programs is replaced by a more generalized notion of constraint solving. Thus if any learning scheme were to be developed for constraint logic programs, the first step in that direction would be to develop models for learning simple constraints which can then be combined with the Inductive Logic programming techniques to get the notion of *Inductive Constraint Logic Programming.*

This is a strong motivation for us to develop frameworks for automatic constraint acquisition or constraint learning.

Before proceeding with the illustration of our model of learning constraints, we shall review existing work related to constraint learning . Because the field of constraints is relatively nascent, there are relatively fewer systems which apply machine learning to constraint based applications.

2.1 Earlier work in Constraint Learning

Mizoguchi and Ohwada[Mizoguchi and Ohwada 92] developed the notion of *constraint-directed generalization* in constraint logic programs to acquire spatial layout information. They propose a mechanism to learn generalized spatial constraint given a set of input spatial constraints so that each of the input spatial constraint is satisfied by the generalized spatial constraint. In physical terms the generalized constraint represents a region which satisfies all input spatial constraints.

Zweben et al.[Zweben et al. 1992] describe an explanation-based learning variant to capture the notion of analytic learning in the domain of constraint based schedules. The idea is to learn search control knowledge for a constraint based scheduling system. The learned knowledge is used for improving the performance in future occasions especially to depth-first backtracking through the space of partial schedules.

Furukawa et al.[Kawamura and Furukawa 93] describe a generic learning model for constraints based on exporting ideas from Inductive Logic Programming to Constraint Logic Programming. They describe the model for linear algebra, and how more general constraints can be learned from initial linear constraints.

Most of the work reported in automatic constraint acquisition has been in the area of constraints on continuous domains. This motivates us to develop a generic model for automatic constraint acquisition in discrete domains.

3 A framework for learning constraints

As discussed earlier, the best paradigm to implement constraint learning mechanisms is the empirical learning method. Now there is a spectrum of empirical learning paradigms which have been applied in machine learning systems. The fundamental process of learning in empirical learning is "Induction" or "Generalization". To date the vast majority of the inductive empirical learning systems have been based on propositional knowledge representation. The only learning systems based on non-propositional system are based on the Inductive Logic Programming Systems based on clausal representation.

Before we proceed with a generalized framework for learning constraints, we need to identify the type of constraints we shall be concentrating for development of learning schemes. Clearly for constraint systems on variables with continuous domains, there cannot be a uniform representation of all the constraints because of the infinite number of choices for each variable. Thus such type of constraint domains call for learning schemes specific to the application being considered[Kawamura and Furukawa 93][Mizoguchi and Ohwada 92].

On the other hand when we consider constraints on variables with discrete domains, we have a uniform representation as relations for the underlying constraints. This may of course be infeasible if the underlying domain is infinite. But a majority of the work in constraint programming is based on finite domain constraints. Thus we shall be concentrating on the constraints with finite and discrete domains. A major portion of the work in constraint programming is based on such type of constraints. In fact the CSP problem formulation is based on finite domain discrete constraints. *Thus our motivation in this thesis to develop a framework for learning constraints on discrete and finite domains.*

As a starting point we shall develop an inductive generalization mechanism for constraints on finite and discrete domains which are explicitly specified as relations (i.e. sets of allowable tuples). We shall call such a representation of constraints as *ECSP form*, short for Explicit CSP form. To begin with we should see whether existing learning frameworks of empirical learning can be exported to take care of constraints in *ECSP form*. Theoretically any constraint on finite and discrete domain can be converted to *ECSP form* by explicit enumeration of allowed combination of values. Hence any learning scheme on *ECSP form* can be theoretically applied to any finite domain discrete constraint application. In contrast to the highly application specific nature of constraint learning mechanisms to be developed for constraints on continuous domains, this uniformity for finite discrete domains prompted us to develop a comprehensively applicable learning system for *ECSP form* of constraints.

But the existing inductive empirical learning methods are unsuitable for the ECSP form of representation, for the following two reasons:

1. Learning methods based on clausal representation (Inductive Logic Programming) systems [Muggleton 92], when applied to ECSP form, generalize too much, because they replace any set of ground instances of a variable with a variable, e.g. Given the inputs R(1,2), and R(3,4), the algorithm produces R(X,Y), which is too general.

2. Learning methods based on propositional knowledge representation forms cannot be used because of two reasons:

   (a) Intractability of the three stage process: Conversion of ECSP form to the particular knowledge representation , application of the generalizing algorithm in that form and

the conversion back to ECSP form.

(b) Intractable size of the conversion of ECSP form into the native knowledge representation in most of the cases.

This prompts us to look for an alternative form of knowledge representation which is closer to the representation of constraints in order to reduce the complexity of the learning process. We achieve this by using the same relational representation for both the input constraints and the output learned form of constraint. In the next few sections we shall develop the model for learning ECSP form of knowledge. In the end we shall discuss the extensions to the basic models in detail, which will enable us to generalize the model to handle more generic form of finite discrete constraints and will allow us to use existing empirical learning frameworks to that end. We discuss a few applications of this model of learning ECSP form in detail.

4 A framework for learning ECSP form of knowledge

In the previous sections we were motivated as to the need for a generalized learning model for constraints. Existing constraint learning frameworks concentrate on use of domain specific representations, especially continuous domains, for design of learning schemes for constraints. In this section first we shall develop a model for learning constraints on discrete domain. Our model shall be developed for the simplest case where we have constraints on variables with finite and discrete domains. Such constraints shall be called as the ECSP form (Explicit Constraint Satisfaction problem). We shall show how existing learning frameworks cannot be extended to the domain of discrete constraints due to several context-dependent reasons.

First we shall develop the most basic model for empirically learning ECSP form of constraints with just positive examples in the learning context. Then we discuss how we shall generalize and enhance the model in different directions.

5 The model for learning ECSP constraints

In this section, we propose a learning (generalization) scheme for data represented in the ECSP form. We have shown in the preceding section how existing learning algorithms are unsuitable for the ECSP type of knowledge representation. The model for learning constraints in the ECSP form is presented in the subsections below.

5.1 Subsumption Ordering in the *ECSP form*

In this section the mechanism of generalizing predicates in *ECSP form* will be presented.

In *ECSP form*,the predicates are represented by relations on finite domains. The generalization operations in the representation are based on subsumption ordering between the predicates.

Definition 5.1 (Predicate Subsumption) *A relation R subsumes another relation S iff $S \subseteq \prod_{attr(S)}(R)$, where $\prod_{attr(S)}$ denotes the projection operator onto attribute columns of S.*

Consider the following example:

```
R =
       A   B
      ------
       0   1
       1   0
      ------
S  =
       A   B   C
      -----------
       0   1  0
       1   0  0
       1   1  0
       ----------
Here,   S subsumes   R.
```

The abovementioned subsumption relation induces a partial order on the predicates. As with any partial order , the subsumption ordering induces a lattice on the predicates. The lattice has the universal relation $\mathcal{R}$ as the Top of the lattice and the empty relation ϕ as the Bottom of the lattice.

Thus the above subsumption relation defines the notions of "generality" and "specificity" in relation to knowledge represented in the ECSP form. Any constraint higher in the lattice(i.e. more general) than a given constraint C captures all the knowledge contained in the constraint C. Consider the example above, Here R states that A can take the values 0 and 1 only. S also conveys the same information. But S conveys an additional knowledge to the effect that C can take the value 0 only. If we consider the solution space of a CSP represented by the relation R, S is more conservative than R because of the additional constraint on the variable C. But at the same time all the knowledge exhibited by relation R with regard to variables A and B is also contained in S. Hence we have a form of representation of knowledge which generalizes the relevant knowledge in a given constraint and yet is able to reduce the solution space of the problem. The method conveniently combines knowledge in two or more constraints and gives a closer approximation to the solution because of the reduction of solution space. Hence this provides us a mechanism to speedup the process of finding solution in a CSP problem.

6 Learning model in the positive only case

In the most basic case, we shall consider the case of learning in the presence of only positive constraints. In such a model, it is natural to consider as a generalization a point in the subsumption lattice described in the previous section, which is above every input constraint. We introduce the notion of "most specific" generalizations to the most useful among the multiple generalizations possible in this case. In this section, we shall use the term "constraint" to denote a positive constraint.

We start by investigating the generalization of two constraints.

6.1 Most Specific Generalization and its properties

Given any two constraints(relations) we can then define the Most Specific Generalization of the two constraints (relations)as follows:

Definition 6.1 (Most Specific Generalization) *A relation R is called a Most Specific Generalization of relations R_1 and R_2 iff R subsumes both R_1 and R_2, and attrib(R)= attrib(R_1) $\cup$ attrib(R_2), and for any relation $R^{'}$ such that R subsumes $R^{'}$, it is not the case that $R^{'}$ subsumes both R_1 and R_2.*

Projection of R on R1 or R2 should be a minimal superset of R1 or R2 respectively.

Consider the following example:

```
R   =                    S=

A  B                    B  C
-----                  ------
0  0                    1  1
0  1                    1  0
----                   ------
```

The above two relations can have many MSG's some of which are given below:

```
 RS= MSG of R and S =

A B   C    A  B C      A B C    A  B C
-------    -------    -------   ------
0 0   1    0  0 0      0 0 1    0  0 0
0 1   1    0  1 1      0 1 1    0  1 1
1 1   0    0  1 0      0 1 0    1  1 0
-------   -------    --------  --------
```

So the general form of the MSG of R and S above is:

```
          RS =    A  B  C
                  -------
                  0  0  X
                  0  1  1
                  Y  1  0
                 --------
```

Here any relation RS of the above form is a MSG of the given constraints R and S. But to have a conservative generalization it is necessary for us to impose further restrictions on the definition of MSG. We can achieve this by a "Maximal Faithfulness" criterion which restricts the unconstrained slots (like X and Y above), to a value which the particular variable is allowed to take by at least one of the subsumed constraints(here R or S). In such a case there is no relaxation or tightening of the constraint on a variable which is not common to both the relations. Thus we can state the following lemma:

Lemma 6.1 *For any MSG S of two relations R_1 and R_2 , S is maximally faithful iff it is true that $\forall X$ such that X is not a variable common to R_1 and R_2, $\prod_X(S)$ is equal to $\prod_X(R_1)$ or $\prod_X(R_2)$ whichever applicable.*

We now look at the solution space of the MSG S as compared to the solution space of R_1 or R_2. If we impose the condition of "Maximal Faithfulness" on S, it is possible for a solution tuple(vector) to satisfy R_1 and yet not satisfy S, and similarly for R_2. This is not necessarily true with all MSG's But if the MSG is maximally faithful, this situation is avoided because of the restrictions on the values the slots can take. Now we can show that any solution tuple satisfying a *maximally faithful* S, will satisfy at least one of R_1 or R_2 but not necessarily both because in such a case it would be join of both the constraints. We state this result below. The proof is very straight forward.

Theorem 6.1 *Let R1 and R2 be two constraints and S be a maximally faithful MSG of R1 and R2. Any solution vector satisfying S will satisfy at least one of R1 or R2.*

In the normal circumstances it often is the case that the join of the two relations R and S forms a subset of the MSG as described in the previous paragraphs. The join is trivially a subset of the MSG of two relations except the following g case. The case is described below:

Let R and S represent the two relations. Let colrs=cols(R)$\cap$ cols(S). Let RC=$\prod_{colrs}$(R) and SC=$\prod_{colrs}$(S). The only case when the join is not a subset of the MSG is given by the following rule: Let card(t,R) represent the cardinality of a tuple t in a relation R. The join of R and S, is not a subset iff $\exists t$ such that $t \in RC$ and $t \in SC$, and both card(t,SC) and card(t,RC) are greater than one. In such a case multiple MSG's can be formed by adjoining the appropriate rows. In all other cases, the join is a subset of the MSG. When R and S are not joinable the join, the empty relation, is a subset of the MSG trivially.

We now summarize the properties of the subsumption ordering and MSG as studied above .

- The subsumption ordering induces a partial order on the constraints(relations).
- Any two constraints can have more than one MSG.
- The empty constraint is subsumed by any constraint.
- The Universal constraint (All the possible variables with all possible values) subsumes any constraint.
- The projection of a *maximally faithful* MSG of two constraints on an attribute that is not common to both the constraints, is the same as the projection of the constraint containing the attribute onto the attribute column.
- The solution space allowed by a maximally faithful MSG of any two constraints is tighter than the solution space of at least one of the constraints.
- Any solution of a maximally faithful MSG of two constraint will satisfy at least one of the constraints.
- The join of any two relations is a subset of the MSG except the case described in the previous section.

6.2 Learning model based on MSG

The MSG algorithm shall be the basis of the learning model we shall be considering. In the most basic model, the inputs are a set of positive constraints, each of which is a finite relation on a set of variables with finite domains. In the section we describe the most basic learning model for

constraints. In this section we shall consider only positive constraints to be present in the model. In this simplistic model we assume that we have series of positive constraints input to the learner, without any background knowledge. In such a case the idea is to obtain a generalized representation which encompasses all the knowledge contained in each of the constraints in the input model.

Thus we can obtain the generalization of n constraints by applying the MSG algorithm n times. We show a schematic illustration of the algorithm in the following section. We shall study the properties of such an algorithm in a later section.

6.3 Algorithm for the Positive Only model

Input: A set of constraints C = $C_1, C_2, C_3, ..., C_n$.

Required : to output a constraint C' such that for every i such that C_i belongs to C, C' subsumes C_i.

Algorithm:

R = C_1; for (i:= 2 to n) do begin t=MSG(R,C_i); R=t; end;

7 Applications of the basic model

The *ECSP form* of knowledge representation is quite useful in a variety of application domains. Some of the applications that we considered are:

- As a mechanism to learn relations and hence use it in any scenario where a learning model for relations is desired.
- As a tool to improve search efficiency in CSP solving techniques.
- As a model for developing methods to improve scheduling techniques.
- To develop specialized methods for spatio-geometrical reasoning.

We shall explore one such application in slight detail below.

7.1 Improving search efficiency in constraint satisfaction problems

Any finite domain CSP, essentially involves a set of relations with a finite set of variables from finite domains. The solution of a CSP corresponds to a relation satisfying all the constituent input relations. Any CSP solving method essentially traverses the solution space, by continually changing the valuations of the variables.

The *ECSP form* gives a handy tool to aid in the search process for solving a CSP. The idea is analogous to the concept of explanation based generalization in Machine learning. Here the knowledge obtained by traversing through a proof tree is encoded and put to use later. Similarly the search control knowledge can be gathered in the process of searching the solution space of the CSP by keeping search paths in the form of a generalized relation by the MSG algorithm. So when searching for the next valuation in the process of solving, the knowledge encoded in the MSG can be put to use as a heuristic. Conversely the knowledge of failed paths can be used to guide as to which paths not to choose. Since the representation scheme of the *ECSP form* coincides with the scheme used by CSP's, the subsumption relation can be used directly to determine if a certain valuation of a variable is desirable or not.

Thus this is a direct application of learning where the *ECSP form* of knowledge representation scheme is used.

8 Extensions of the framework presented

In this section we shall briefly describe the different directions in which we are working in order to extend the basic framework presented in this article. The extensions being considered in logical order are presented in the following subsections.

8.1 Extension of the basic model to handle negative information

One of the immediate extensions being looked at by us is the incorporation of the notion of a *negative constraint* into the model. Before we proceed further we shall define the notion of "Negative Constraint". A negative constraint is a constraint or a relation on a set of variables which must not be explainable by any learned concept obtained by generalization of the input constraints. In the presence of a negative constraint, also commonly known as a *nogood*, we need to define the notions of *consistency* with respect to a nogood, analogous to a negative example in empirical learning.

This extended model with capability of handling negative examples is very useful in a large number of situations. All the applications mentioned in the basic model case, can be dealt with more efficiently and elegantly. Consider, for example, the CSP speedup application. In this application negative constraints can help in reducing unwanted paths while positive constraints can act as heuristics. This dramatically increases the efficiency of the CSP solving procedure. When we extend the learning model presented in the previous section to handle negative constraints alongside the positive constraints, we have a situation where we can have a spectrum of possible constraints depending upon the idea of how general our model needs to be represented.

8.2 Extension of the model to handle clustering

In the basic model of learning constraints presented, we do not have a difference in representations of data and information, i.e. learned information from the constraints. The representation scheme chosen for representing the learned constraints is the same as the one of the constraints, i.e. the relational form.

It is a good starting point as a model for incorporating learning into constraints. But from a general case of constraint applications, such a type of learning model needs to be diversified depending upon the application domain.

The simplest way of dissociating information learnt from the data input, is by using clustering techniques to cluster similar data into groups and work with these groups in the learning process. Hence this type of scheme will be generalization of the model presented in the previous section, in the sense that the representation schemes for the information and the data differ. Also a whole set of conceptual clustering techniques used in machine learning literature can be applied to constraints represented in ECSP form.

Ellman [Ellman, 1993] describes a system to incorporate clustering techniques into Constraint Satisfaction. He describes a method of clustering approximately equivalent objects into classes, and then using these classes to develop hierarchical constraint solvers.

We can improve upon our basic model by incorporating conceptual clustering techniques in the constraint generalization algorithm. We need to define a measure by which constraints are

compared and found similar or dissimilar. Based on such a measure of classification, we can develop a application specific clustering technique over the constraints and develop learning algorithms using these approximated clusters. Thus this is one of the directions in which the basic model presented in our learning model is being extended.

8.3 Background Knowledge incorporation

The existing framework proposed in the earlier section, has no notion on background knowledge which is a fundamental part of present day learning systems based on Inductive Logic Programming.

Thus one direction in which the work presented in the earlier section needs to be extended is to incorporate inference in the presence of background knowledge.

In Inductive Logic programming, the notion of θ-subsumption which is defined as an absolute generality relation is extended in the presence of background first-order knowledge, to give the notion of "Relative Subsumption". Analogously, we should extend our notion of the constraint subsumption to handle background constraints. In such a case we must have a choice of a representation of background knowledge which gives a reasonable motion of relative subsumption which is not too costly to compute.

In this context research needs to be done as to what form of background knowledge of constraints is the best suitable for a generalized constraint learning paradigm with background knowledge. A relational type representation for the constraints in the background knowledge will be intractable , while on the other hand a first-order clausal form will be a disadvantage in terms of granularity of the data considered. A decision tree or a control rule type of background knowledge might be preferable in terms of computational limitations.

Thus incorporation of background knowledge into the learning system developed in the previous sections, is one of the important extensions which needs to be studied.

8.4 A framework for Inductive Constraint Logic Programming

An immediate extension of the basic model presented in this proposal seeks to incorporate the constraint generalization methods into clausal generalization methods based on Inductive Logic Programming to develop a solid foundation for Inductive Constraint Logic Programming. As a first step we shall consider generalizing the framework to handle discrete domain constraint logic programs like CLP(FD) , CC(FD) etc. Freuder and Wallace [Freuder and Wallace 95] extend the notion of the basic "nogoods" to use high-level abstraction for generalizing the nogoods to higher level concepts, in the solving a CSP.

In such a framework we need to develop a neat way to integrate the constraint learning mechanisms with θ-subsumption of ILP, to give notion of generality between two constraint clauses,i.e. clauses with constraints in the body of the clause. Such a framework will be helpful in developing tools for automatic constraint acquisition in domains where CLP's have been used extensively, e.g. intelligent scheduling.

8.5 Development of Inductive Hierarchical Constraint Logic Programming Framework

The other direction in which the model of previous section should be extended is to investigate if constraint learning techniques can be extended to handle constraint hierarchies. Ideally, we would

like to have a learning system which will generate the constraint hierarchies once we provide the input constraints as an input.

This type of system would be very useful in automating applications involving hierarchical constraint solving. More precisely this notion of automatically generating hierarchies can be combined with the hierarchical constraint logic programming techniques to give the notion of inductive hierarchical constraint logic programming.

As a starting point we can generalize our work on constraint generalization to generate constraint hierarchies by importing ideas from data mining. In particular work on Mining Knowledge at multiple knowledge levels by Han et al. [Han 95], is a good starting point for such a system.

8.6 Correctness Results

The most important work in the file of learning has been in the computational learning theory. In particular the theory of PAC learning is a neat way to characterize theoretically a learning system which clearly identifies what is the "new" knowledge learnt in a learning system. In our learning system, we can identify the correctness results for data represented as relations based on the PAC learning system.

9 Conclusions and Scope for Future Work

In this paper, we have outlined a model of learning where the underlying knowledge representation scheme is close to the one used in a majority of constraint programming applications. We have shown how existing learning algorithms based on propositional representations or those based on inductive logic programming paradigm are inadequate for learning in the model. We then discuss the subsumption hierarchy in the model and present the learning algorithm for generalizing in our model. We have studied the properties of the results obtained by learning in this model.

Thus we have outlined a generalized learning model for constraints on finite discrete domains. We are working on extending this model in the directions mentioned in the previous section.

References

[Ellman, 1993] Ellman T., Abstraction via approximate symmetry. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Chamberry, France, August 1993.

[Freuder and Wallace 95] Freuder E.C. and Wallace R.J., "Generalizing inconsistency learning for constraint satisfaction", Proceedings of IJCAI-95 the Fourteenth International Joint Conference on Artificial Intelligence, C. Mellish, ed., Morgan Kaufmann.

[Han 95] Han J., " Mining Knowledge at Multiple Concept Levels", Proc. 4th Int'l Conf. on Information and Knowledge Management (CIKM'95), Baltimore, Maryland, Nov. 1995, pp. 19-24.

[Kawamura and Furukawa 93] Kawamura T. and Furukawa K., "Towards Inductive Generalization in Constraint Logic Programs", Proceedings of the IJCAI-93 Workshop on Inductive Logic Pro-

gramming, Chambery ,France(August 93), pp. 93-104.

[Mackworth 92] Mackworth A., Constraint Satisfaction, in S.C.Shapiro,ed.,The Encyclopedia of AI,pp 285-293,Wiley, New York,1992.

[Mizoguchi and Ohwada 92] Mizoguchi F. and Ohwada H., "Constraint-directed generalization for learning spatial relations", Proceedings of the International Workshop on Inductive Logic Programming, ICOT TM-1182, Tokyo ,1992.

[Muggleton 92] Muggleton S., Inductive Logic Programming. Academic Press, 1992.

[Page and Frisch 91] Page C.D., and Frisch A.M., "Generalizing atoms in constraint logic ", Proceedings of the Second International Conference on Knowledge Representation and Reasoning, pp 429-440, 1991.

[Scheix et al 93] Scheix T. et al., "Nogood recording for static and dynamic CSP's", Proc. of International Conference on Tools for AI, 1993.

[Shavlik and Dietterich 90] Shavlik J.W., and Dietterich T.G. (eds.), Readings in machine learning. Morgan Kaufmann Publishers , 1990.

[Wu 95] Wu Xindong, Knowledge Acquisition from Databases. Ablex Publishing Corporation, 1995.

[Zweben et al. 1992] Zweben M., Davis E. , Daun B.,Drascher E.,Deale M. and Eskey M., Learning to improve constraint-based scheduling, Artificial Intelligence 58(1992) 271-296.