Chapter 3:

Position Papers

Predicate Invention in ILP – an Overview

Irene Stahl*

Fakultät Informatik, Universität Stuttgart, Breitwiesenstr. 20-22,
D-7000 Stuttgart 80

Abstract. Inductive Logic Programming (ILP) is a subfield of machine learning dealing with inductive inference in a first order Horn clause framework. A problem in ILP is how to extend the hypotheses language in the case that the vocabulary given initially is insufficient. One way to adapt the vocabulary is to introduce *new predicates.*
In this paper, we give an overview of different approaches to *predicate invention* in ILP. We discuss theoretical results concerning the introduction of new predicates, and ILP-systems capable of inventing predicates.

1 Introduction

Inductive inference aims to construct a theory covering given facts. More formally, given a set of positive examples $E^{\oplus}$, a set of negative examples $E^{\ominus}$ and a theory T, the system is to find a theory T' such that $T' \vdash E^{\oplus}$ and $T' \nvdash E^{\ominus}$. That is, T' has to be consistent with the examples.

Inductive Logic Programming (ILP) [Mug92] is inductive inference in a restricted first order logic framework: both the given theory T and the target theory T' are restricted to Horn clause theories. Using that framework as opposed to propositional calculi, ILP belongs to the most powerful inductive inference paradigms.

In order to restrict the generally infinite search space for T', ILP systems impose a bias on the hypotheses. This bias includes the vocabulary for the hypotheses, i.e. the available predicate, function and constant symbols. If the search space does not include a hypothesis that is consistent with the examples, an ILP system should shift its bias. This can be done by introducing *new predicates* in the hypotheses language such that the extended vocabulary is sufficient for the induction task.

This operation is referred to as *Predicate Invention.* The paper aims to discuss theoretical and practical aspects of predicate invention in ILP.

2 Two Theoretical Results

Inductive inference is based on examples true or false in an intended model. The system is expected to produce a finite set of formulas in a language $\mathcal{L}$ explaining the positive and excluding the negative examples.

* This work has been supported by the European Community ESPRIT project ILP (Inductive Logic Programming).

More formally, let $\mathcal{M}$ be the intended model and $\mathcal{L}_0$ the language of ground facts over the predicate, function and constant symbols in $\mathcal{M}$. A *complete presentation* of $\mathcal{M}$ consists of

$$E^{\oplus} = \{\phi \in \mathcal{L}_o \mid \mathcal{M} \models \phi\} \quad \text{and} \quad E^{\ominus} = \{\phi \in \mathcal{L}_o \mid \phi \notin E^{\oplus}\}.$$

For the following proofs, we assume $E^{\oplus}$ to be recursively enumerable.

The positive and negative example sets inductive inference systems are supplied with are - in practice finite - subsets of $E^{\oplus}$ and $E^{\ominus}$. However, for finite example sets there is no need to invent new predicates, as the positive example set can always be used as correct result of inductive inference. Therefore, we consider the limit case that a complete, possibly infinite presentation of $\mathcal{M}$ is given.

The goal of inductive inference in this setting is to find a *finite set of formulas* T in $\mathcal{L} \supseteq \mathcal{L}_0$ such that

$$\begin{aligned} & T \vdash E^{\oplus} \wedge T \nvdash E^{\ominus} \\ \equiv\ & \forall \phi \in \mathcal{L}_0 \left((\phi \in E^{\oplus} \rightarrow T \vdash \phi) \wedge (\phi \notin E^{\oplus} \rightarrow T \nvdash \phi)\right) \\ \equiv\ & \forall \phi \in \mathcal{L} \left(T \vdash \phi \leftrightarrow \phi \in E^{\oplus}\right) \end{aligned}$$

The last formula with finite T is exactly the definition of T being a *finite axiomatization* of $E^{\oplus}$. Therefore, the problem of inductive inference is in the limit the problem of finding a finite axiomatization for a given model. If the intended model is *not finitely axiomatizable* within a language $\mathcal{L}$, inductive inference can not succeed. Furthermore, it is not even *decidable* whether the set $E^{\oplus}$ of ground facts valid in $\mathcal{M}$ is finitely axiomatizable within the language $\mathcal{L}$.

Theorem 1. *Given a recursively enumerable set of ground facts $E^{\oplus}$ in a language $\mathcal{L}_0$ it is* undecidable *whether $E^{\oplus}$ is finitely axiomatizable in $\mathcal{L} \supseteq \mathcal{L}_0$.*

Proof. We omitt the application of Rice's Theorem on the undecidability of nontrivial index sets [Ric53].

Thus an inductive inference system can not decide whether its vocabulary is sufficient for axiomatizing the intended model. Only finite, decidable hypotheses languages $\mathcal{L}$ as e.g. CLINT's language series [Rae92] or RDT's rule schemata [KW91] allow a decision by enumerating all possible hypotheses. If none of them is consistent with the examples, the intended model is not finitely axiomatizable within $\mathcal{L}$. For infinite or undecidable languages $\mathcal{L}$ as e.g. first order Horn logic, the insufficiency for axiomatizing the intended model has to be assumed heuristically.

If the hypotheses language $\mathcal{L}$ is not sufficient, the inductive inference system may try to extend $\mathcal{L}$ by a new predicate. The introduction of a new predicate is useful for finding a finite axiomatization, as Kleene proved [Kle52]:

Theorem 2. *Any recursively enumerable set C of formulas in a first order language $\mathcal{L}_0$ is finitely axiomatizable in a first order language $\mathcal{L}$* using additional predicate symbols *except from those in $\mathcal{L}_0$.*

For our purpose, the set C is the presentation $E^{\oplus}$ of the intended model. Kleene's theorem assures that inductive inference will *always succeed* provided the system invents the appropriate new predicates.

The practical application of Kleene's result faces some problems. Though Kleene's proof of the theorem is constructive, it gives no practicable method for constructing a finite axiomatization. In the course of his proof Kleene introduces new predicates regardless of whether they are really necessary for finite

axiomatizability. In contrast, inductive inference systems should introduce new predicates only if they are needed.
This capability involves dealing with the following problems:

1. *when* to introduce a new predicate. It is undecidable whether a new predicate is necessary for finitely axiomatizing the intended model.
2. *how* to induce a definition for the new predicate and *what* to use as base for that induction process. The intended model provides no direct information about the new predicate.
3. *how* to determine the arity and the argument terms of the new predicate.

In the following, we are going to discuss the two main approaches to predicate invention, *reformulation* and *demand driven approaches.*

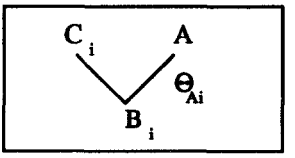
3 Reformulation Approaches and Inverse Resolution

Reformulation approaches introduce new intermediate predicates as a reformulation of an already existing theory in order to express it more compact and concise. This is done without direct reference to the goal of learning a specific concept.

The introduction of intermediate predicates is a kind of reversal of explanation based learning which deductively replaces intermediate predicates by their definition. *Inverse resolution systems* directly explore the idea of inverting deductive resolution steps. *Scheme-driven* approaches define new intermediate predicates as combinations of known predicate literals that match one of the schemes for useful literal combinations given initially.

3.1 Inverse Resolution Systems

The inverse resolution operator that introduces a new predicate is often called *W-operator* or *intraconstruction.* Given a set of clauses $\{B_1, ..., B_n\}$ it constructs a set of clauses $\{C_1, ..., C_n\}$ and a clause A such that B_i results from resolving A with C_i on a fixed literal $L \in A$. Since L is resolved away in B_i and nothing is known about its predicate symbol, a new predicate is invented.



The intraconstruction operator we are going to use throughout the following discussion is the G_2-operator [Wir88]. It sets A to $lgg(\{B_1, ..., B_n\}) \cup \{L\}$, where *lgg* is the least general generalisation [Plo70] of the input clauses and L is a new predicate. L is negative only if $lgg(\{B_1, ..., B_n\})$ already contains a positive head literal. The clauses C_i are set to $\{\overline{L}\}\Theta_{A_i} \cup (B_i - lgg(\{B_1, ..., B_n\})\Theta_{A_i})$.

The crucial problems are the arguments of the new predicate literal and the application criteria for the operator.

CIGOL [MB88] uses a restricted form of G_2. The new predicate literal in A must be negative, i.e. all clauses B_i must have the same head predicate. The clauses C_i are restricted to unit clauses such that the equation simplifies to $C_i = \{\overline{L}\}\Theta_{A_i}$.

For determining the arguments of the new predicate, the substitutions Θ_{A_i} are considered. If a substitution for a variable in Θ_{A_i} has no variables in common with substitutions for other variables in Θ_{A_i} it is irrelevant and must not appear as argument of the new predicate.

The operator is applied whenever it results in a reduction of the size of the knowledge base.

LFP2 [Wir88] employs the unrestricted G_2-operator described above. In contrast to CIGOL, argument terms are considered to be relevant for the new predicate if they occur in the remaining as well as in the omitted parts of the B_i. The arguments of the new predicate L are set to

$$\{\, t \in args(lgg(\{B_1, .., B_n\})) \mid \forall i \in \{1, .., n\} \quad t\Theta_{A_i} \in args(B_i - lgg(\{B_1, .., B_n\})\Theta_{A_i}) \,\}$$

where $args(C)$ are the argument terms of the literals in clause C. As in CIGOL the operator is triggered by the minimum size of the knowledge base.

ITOU The intraconstruction operator of ITOU [Rou91] applies only to flattened, i.e. function free, Horn clauses. It is a slight modification of G_2.

First, the new predicate literal L in A must be negative. Secondly, the operator is applied only to two clauses B_1 and B_2 such that the new predicate describes the variation of B_1 and B_2 relatively to their common generalisation. The last modification concerns the definition of C_i:

$$C_i = \{\overline{L}\} \cup (B_i\Theta_{A_i}^{-1} - lgg(\{B_1, ..., B_n\})).$$

Because of the lack of structured terms, the inverse substitution $\theta_{A_i}^{-1}$ is a variable renaming and can be simply determined from B_i and A. Similar as in LFP2, the variables of the new predicate L are defined to

$$\{\, x \in var(lgg(\{B_1, B_2\})) \mid x\Theta_{A_1} \in var(B_1 - lgg(\{B_1, B_2\})\Theta_{A_i}) \vee x\Theta_{A_1} \in var(B_2 - lgg(\{B_1, B_2\})\Theta_{A_i}) \,\}$$

Intraconstruction in ITOU is triggered by a user request.

RINCON [WL89] tries to improve the efficiency of its theory by introducing intermediate predicates. It restricts G_2 to two input clauses B_1 and B_2. As the definition of the new predicate is to be conjunctive, it must be the positive head literal in A. As arguments of the new predicate all variables within $lgg(\{B_1, B_2\})$ are used.

The process is triggered by an example clause the theory does not explain yet. RINCON tries intraconstruction with each clause in the background theory and choses the one that can be used for rewriting the most rules if several possible reformulations exist.

Banerji's System The peculiarity of the intraconstruction operator DREAM is that the two input clauses B_1 and B_2 must have an identical head P and a nonempty set C of common identical body literals. Then, without use of lgg A is set to $(P \leftarrow C, L)$ where L is a new predicate literal. As arguments of L, the most complex terms common to the remaining parts of B_1 and B_2 are taken. If L is $D(t_1, ..., t_n)$, DREAM produces as definition of D the sentences

$$D(x_1, ..., x_n) \leftarrow P_i' \text{ where } P_i'[x_1/t_1, ..., x_n/t_n] = B_i - \{P \leftarrow C\}$$

Banerji's system revises incrementally a theory based on examples. After each revision, it tries to apply DREAM on every pair of clauses within the revised theory.

3.2 Scheme-driven Systems

Scheme-driven systems define new predicates as combinations of known predicate literals. In order to restrict the space of possible combinations, they use *schemes* describing useful literal combinations at an abstract level. If an instantiation of a scheme with known predicate literals proved useful during learning, an oracle is asked to approve it as definition of a new predicate.

CIA [Rae92] is used together with the example driven learner CLINT [Rae92]. It uses *higher order schemes* for predicate invention.

A higher order scheme S is a Horn clause with existentially quantified predicate variables. A Horn clause c *matches* S if there exists a substitution Θ for the predicate variables in S and a substitution ρ for the variables in S such that $head(S)\Theta\rho = head(c)$ and $body(S)\Theta\rho \subseteq body(c)$. It matches S *partially* if only the second condition holds, i.e. if the head predicate in S remains variable.

Given a positive example, CLINT generates a starting clause. CIA matches this clause against all schemes available in the data base. If it matches a scheme, the resulting clause is proposed to the oracle. If it matches only partially a scheme, the oracle is requested to approve or reject the partial clause as a new predicate definition. In the former case, the oracle names the variable head predicate.

The same method is applied on the generalised starting clause produced by CLINT. In a last step, all new rules are transformed into schemes to be used in subsequent learning steps. Thus, the space of new predicate definitions CIA considers depends on the theory induced so far.

FOCL [SP91] is an extenstion of Quinlan's FOIL [Qui90]. Both systems induce Horn clauses from general to specific by successively adding body literals to the target clauses according to the maximum information gain.

As FOIL adds literals one at a time, it overlooks the possibility that a combination of literals may have a large information gain whereas the single literals have zero or less gain. To overcome this limitation, FOCL considers combinations of literals restricted by *relational clichés*. Relational clichés are schemes that restrict the number of literals in a combination, the predicates used to fill the single literal positions and the variables the literals should share.

If no literals with positive gain exist to be added to the clause, the instantiation of the relational clichés with maximum gain is added and cached for later reuse. After the system completed its learning, the cached instantiations of relational clichés are proposed to the user to be named and adapted in the background theory.

As in CIA, the new predicates FOCL induces depend strongly on the available clichés. Directly recursive defined new predicates can not be found.

4 Demand-driven Approaches

As opposed to reformulation approaches, demand-driven systems try to discover situations where the given vocabulary is not sufficient for finitely axiomatizing the intended model. Then, a new predicate is invented. The decision about introducing a new predicate is done heuristically, as an exact decision can not be done in general.

MENDEL [Lin91] is an inverse resolution system that differs from the reformulating systems mainly in the control of intraconstruction. Only if the theory grows beyond a fixed bound, the intraconstruction operator EXTRACT is applied. In that case MENDEL assumes that the current theory is not finitely axiomatizable within the given vocabulary, and introduces a new predicate.
EXTRACT is a nondeterministic G_2-operator as it sets A instead of the unique

$$lgg(\{B_1, ..., B_n\}) \cup \{L\} \text{ to } g(\{B_1, ..., B_n\}) \cup \{L\}$$

where g may be any common generalisation of $\{B_1, ..., B_n\}$. Thus, MENDEL has to cope with an additional nondeterministic choice apart from which clauses $\{B_1, ..., B_n\}$ to use. The arguments of the new predicate literal L are the variables in Θ_{A_i}.

The clauses generated for the new predicate L are subject to further generalisation steps. Only if they result in a directly recursive clause for L, the new predicate is accepted.

SIERES [WO91] starts with the least general generalisation of the examples as clause head and specialises this clause by successively adding literals to its body until the output of each example is computed correctly by the clause.

The space of Horn clauses SIERES searches is constrained by *argument dependency graphs* specifying the number of literals within a clause and the argument dependencies between them. The arguments of possible body literals are restricted further by the preference for *critical terms*, i.e. unused input- or unbound output terms. Only if not enough of them exist, uncritical terms or new variables are used as arguments.

If none of the existing predicates yields a correct extension of the clause, a new predicate is invented. A minimal selection of critical terms is taken as arguments such that the resulting clause contains no more critical terms. A definition of the new predicate is determined by a recursive call of SIERES on the example set defined by the bindings of the chosen argument terms for all examples. The criterion for inventing a new predicate is applicable only because SIERES searches a finite subset of Horn logic as hypotheses space.

There are some restrictions and difficulties with this approach. First, a new predicate can only be invented at the end of a clause. A second problem is that sparse examples for the new predicate may cause the induction process to fail. A more complex problem is that the recursive call of the system on the examples of the new predicate may not terminate, but lead to an infinite chain of new predicates.

DBC (Discrimination-Based Constructive Induction) [KNS92b] proceeds similar to SIERES. An overgeneral clause is completed with a new predicate such that all negative examples are excluded from being covered. Instead of using critical terms for determining the arguments of the new predicate, DBC searches for a minimal relevant variable set that discriminates between positive and negative examples.

First, all variables of the clause are used as arguments of the new predicate. Its instantiations according to the examples for the clause constitute its preliminary definition. This definition corrects the overgeneral clause but may contain irrelevant terms. DBC tests greedily for each variable whether the variable and

its instantiations can be omitted from the new predicate and its definition without sacrificing correctness. The resulting reduced new predicate is added to the overgeneral clause, and its instantiations for the positive and negative examples of the clause are used as positive and negative examples for the new predicate in the subsequent induction process.

DBC is used with the top-down heuristically guided learner CHAM [KNS92a]. A new predicate is constructed whenever there are no correct clauses fullfilling the encoding length restriction [Qui90]. The clauses to be specialized by new predicates are chosen heuristically among the overgeneral clauses.

The system stops predicate invention when the whole theory violates the encoding length restriction, or when the instances of the new predicate are the same as the examples of the clause except for the name of the predicate. Thus, it avoids heuristically the looping problem of SIERES.

MOBAL/CLT [Mor91, Wro92] is a knowledge acquisition system that helps creating a model for an application domain. The domain model of MOBAL consists of function free Horn clauses, where each rule has an attached *support set* listing exceptions and examples for the applicability of that rule. An exception is an instantiation of the rule it must not be applied with, whereas an example is a successful instantiation.

If a new fact contradicts the latest theory, the knowledge revision module KRT selects in a first step the rule to be specialised among those responsible for the inconsistency. In the second step the selected rule is minimally specialised by adding the wrong instantiation to its exception set. Then reformulation operators are tried that aim to replace the extensional exception list with an intensional description.

If a known predicate discriminates the positive examples of the rule from its exceptions, it is added to the rule definition. Otherwise the concept learning tool CLT [Wro92] invents a new predicate c for that purpose. As arguments of the new predicate c a subset of the rule variables is taken. The values of those variables in the support set yield the positive and negative examples of c. Based on them, the learning module RDT [KW91] tries to induce an intensional definition of the new concept c. If RDT succeeds, c is added to the body of the incorrect rule. Else the search for a definition is postponed until further exceptions occur. As the grouping of exceptions resembles concept formation, this approach to predicate invention is called a *concept formation approach.*

RDT is capable of inducing rules both with c as head and c within the body. This is important for the evaluation criterion of CLT. A new concept is accepted if at least two sufficient conditions were found about c, i.e. rules with c as head, *and* at least two necessary conditions, i.e. rules with c as only body literal, or at least one rule that uses c in its body among other literals. The acceptance criterion demands that the new predicate can be used for rewriting a minimum number of rules in the knowledge base.

Non-Monotonic Learning A special case of predicate invention is closed world specialisation [BM92]. Given a clause covering negative examples, it is specialised by the negation of a new exception predicate, i.e. negative examples are treated as exceptions of the rule applicability. As negation by failure is used, all previously covered examples are still covered.

5 Comparison and Evaluation

5.1 Decision Criteria for Introducing a New Predicate

The decision criteria for introducing a new predicate can be classified in *demand-driven* and *goal-free* methods.

Demand-driven approaches decide heuristically on their vocabulary being insufficient for finitely axiomatizing the intended model. That is, new predicate invention is triggered whenever the system fails in expressing the target theory within the given vocabulary. Using a finite hypotheses language as in SIERES, this may be decided by checking all hypotheses on consistency. For infinite hypotheses languages, the decision about the fail of the system has to be done heuristically. MENDEL assumes the vocabulary being insufficient whenever the hypotheses grows too large. Similarly, DBC uses the encoding length restriction. MOBAL introduces a new predicate if none of the existing predicates characterizes a rule's exception set.

Goal-free approaches introduce new predicates without a direct reference to the goal of learning a specific concept. Nevertheless, due to the constraints systems impose on new predicates, they are very likely to be useful for finitely axiomatizing the intended model either. One constraint used by CIGOL, LFP2 and RINCON is *knowledge base compression*. Among the possible reformulations of the knowledge base the one that reduces the size of the theory most is chosen. This is closely connected to MENDEL's decision criterion and to MOBAL's acceptance criterion.

Another goal-free constraint on new predicates is imposed by *schemes*. If scheme-driven systems observe a literal combination they assume to be useful and interesting according to their schemes, they propose it as new predicate definition to the oracle.

5.2 Evaluation of the Different Approaches

The *evaluation* of the different approaches to predicate invention is difficult. There are only sparse theoretical and empirical results concerning the quality of the new predicates introduced by the systems. Wrobel [Wro92] and Ling [Lin91] propose a set of *quality dimensions* along which the new predicates could be measured.

Though we were not able to do any experimental tests on the systems we discussed, in table 1 we try to give an assessment of the systems according to Wrobels and Lings quality dimensions:

1. Is the *learnability* of the desired target concepts improved by means of predicate invention? For most systems this is - at least possibly - the case. Only in RINCON it is not improved as it uses new predicates as a mere reformulation of the theory.

2. Is the system capable of inducing *directly recursive defined new predicates*? Only those *could* be necessary for finite axiomatizability, but need not necessarily be. Non-recursively defined new predicates are eliminable by unfolding and therefore not necessary. Most systems except RINCON and FOCL allow for inducing recursive new predicates. However, this may require further learning steps.

3. Is the *classification accuracy* of the resulting axiomatization improved by the use of new predicates? For inverse resolution and scheme-driven systems it is

	learn-ability	recursive definitions	accuracy	efficiency	structure
CIGOL	possibly	yes	?	decreases	more compact theory
LFP2	possibly	yes	?	decreases	
ITOU	possibly	yes	?	decreases	
RINCON	no	no	same	better	
Banerji	yes	yes	?	decreases	
CIA	possibly	yes	?	decreases	theory size increases
FOCL	yes	no	same	decreases	
MENDEL	yes	yes	?	decreases	more compact theory
SIERES	yes	yes	at least not worse	decreases	
DBC	yes	yes	at least not worse	decreases	
MOBAL	possibly	yes	at least not worse	decreases	

Table 1. Evaluation

unclear how new predicates affect classification accuracy. The operators themselves as pure reformulations have of course no effect. But if the new predicate definitions are subject to further learning steps, there is the danger of overgeneralisation or -specialisation and therefore less accuracy.

4. Does the *efficiency* of the resulting axiomatization improve or decrease? Usually, efficiency should decrease as additional resolution steps are necessary to apply the new predicates. Only RINCON produces a more efficient theory. However, this is mostly due to the fact that RINCON's rules are organized in a kind of Rete-net.

5. Does *structure* and *understandability* of the resulting knowledge base improve? The problem is how to measure structure or understandability. Most systems aim to produce smaller, more compact theories. Only for CIA and FOCL new predicates increase the theory size as they are not used to rewrite the theory.

Table 1 shows only weak results about the properties of predicate invention operators. This may be due to the fact that some properties, e.g. learnability, are undecidable, whereas others, e.g. knowledge base understandability, can not be properly quantified. Additionally, the experimental evaluation of systems performing predicate invention in ILP is almost lacking.

6 Conclusions

In this paper, we have discussed different approaches to predicate invention in ILP. There are only few systems able to invent new predicates and only weak or no results about the properties of their operators. The crucial problems concerning the introduction of new predicates, have not yet been solved satisfactorily. Nevertheless, the *need* for predicate invention is undoubted.

References

[BM92] Bain, M., Muggleton, S. (1992): *Non-Monotonic Learing* in S. Muggleton (ed): Inductive Logic Programming, Academic Press

[Ban92] Banerji, R. B. (1992): *Learning Theoretical Terms* in S. Muggleton (ed): Inductive Logic Programming, Academic Press

[KW91] Kietz, J., Wrobel, S. (1991): *Controlling the Complexity of Learning in Logic through Syntactic and Task-Oriented Models* in S. Muggleton (ed): Inductive Logic Programming, Academic Press

[KNS92a] Kijsirikul, B., Numao, M., Shimura, M. (1992): *Efficient Learning of Logic Programs with Non-determinate, Non-dicriminating Literals* in S. Muggleton (ed): Inductive Logic Programming, Academic Press

[KNS92b] Kijsirikul, B., Numao, M., Shimura, M. (1992): *Discrimination-Based Constructive Induction of Logic Programs*, Proc. of the 10th Nat. Conf. on AI, San Jose, CA

[Kle52] Kleene, S. C. (1952): *Finite Axiomatizability of Theories in the Predicate Calculus Using Additional Predicate Symbols* in S. C. Kleene: Two Papers on the Predicate Calculus, Memoirs of the American Mathematical Society No. 10, Providence, RI

[Lin91] Ling, C. X. (1991): *Inventing Necessary Theoretical Terms in Scientific Discovery and Inductive Logic Programming*, Report No. 302, Dept. of Computer Science, University of Western Ontario, London, Ontario

[Mor91] Morik, K. (1991): *Balanced Cooperative Modeling*, in R. S. Michalsky, G. Tecuci (eds): Proc. First Int. Workshop on Multistrategy Learning, 65 – 80

[MB88] Muggleton, S., Buntine, W. (1988): *Machine Invention of First-Order Predicates by Inverting Resolution*, Proc. of the 5th Int. ML Workshop, Morgan Kaufman

[Mug92] Muggleton, S. (1992): *Inductive Logic Programming*, in S. Muggleton (ed): Inductive Logic Programming, Academic Press

[Plo70] Plotkin, G. D. (1970): *A Note on Inductive Generalisation* in: B. Meltzer, D. Mitchie (eds): Machine Intelligence 5, Edinburgh University Press

[Qui90] Quinlan, J. R. (1990): *Learning Logical Definitions from Relations*, Machine Learning 5, 239 – 266

[Rae92] De Raedt, L., Bruynooghe, M. (1992): *Interactive Concept-Learning and Constructive Induction by Analogy*, Machine Learning 8(2), 107-150

[Ric53] Rice, H. G. (1953): *Classes of Recursively Enumerable Sets and their Decision problems*, Trans. AMS 89

[Rou91] Rouveirol, C. (1991): *Extensions of Inversion of Resolution Applied to Theory Completion* in S. Muggleton (ed): Inductive Logic Programming, Academic Press

[SP91] Silverstein, G., Pazzani, M. J. (1991): *Relational Cliches: Constraining Constructive Induction During Relational Learning*, Proc. MLW 91

[Wir88] Wirth, R. (1988): *Learning by Failure to Prove*, Proceeding of EWSL 88, Pitman, 237 – 251

[WO91] Wirth, R., O'Rorke, P. (1991): *Constraints on Predicate Invention* in Proc. of the 8th Int. Workshop on ML, Morgan Kaufmann

[WL89] Wogulis, J., Langley, P. (1989): *Improving Efficiency by Learning Intermediate Concepts*, Proc. of the 11th IJCAI, Detroit

[Wro92] Wrobel, S. (1992): *Exploiting a Problem-Solving Context to Focus Concept Formation*, to appear in Machine Learning journal