Predicate Invention: A Comprehensive View [1]

Stefan Kramer

Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
E-mail: stefan@ai.univie.ac.at

Keywords: Predicate Invention, Constructive Induction, Inductive Learning, Inductive Logic Programming

Abstract

This paper discusses predicate invention (PI) from various, previously neglected viewpoints. First of all, we argue that predicate invention should build on existing work on constructive induction in propositional learning. We recall the major reasons for constructive induction in propositional languages, and give a brief overview of the frameworks for constructive induction by Matheus, Wnek and Michalski. We then apply these frameworks to predicate invention in order to categorize systems and to identify relevant aspects of PI. The discussion demonstrates that some relevant aspects are treated only implicitly, and some are largely neglected in many systems. Secondly, we review current criticism against constructive induction that also concerns predicate invention. In particular, we agree with Sutton's demand that constructive induction should be based on continuing learning, i.e. it should reuse representational "tricks" in a series of learning tasks. Thirdly, we discuss the advantages and disadvantages of fully-automatic vs. interactive predicate invention. The question is how to create meaningful new predicates. Since comprehensibility and syntactical complexity are not necessarily the same, human intervention may be required if humans shall make sense of the resulting theory.

We try to attract more attention to important aspects that have not yet been recognized clearly, and still are present in the work of many authors. These aspects are illustrated by existing PI systems.

[1] This research is sponsored by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant number P10489-MAT. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science, Research, and Arts. I would like to thank Gerhard Widmer for valuable discussions.

Contents

1 Introduction

[Michalski 93] distinguishes between selective induction and constructive induction as follows:

> "Empirical learning uses little domain knowledge, while constructive induction uses more domain knowledge. A more precise way to characterize this distinction is that in empirical induction the description space for examples and for the hypotheses is the same, while in constructive induction these spaces are different."

Constructive induction was first defined for learners using a propositional language. Early work focussed on the use of constructive operators during induction [Michalski 83]. These so-called "constructive generalization rules...generate inductive assertions that use descriptors not present in the original observational statements". Later, researchers investigated *feature construction*, i.e. the construction of new features from existing features.

In the early nineties, a new area called *Inductive Logic Programming (ILP)* [Muggleton 92] developed, dealing with learning theories in first-order logic. From the very beginning, ILP also was concerned with constructive induction in first-order logic, better known as *predicate invention* [Muggleton & Buntine 88]. Obviously, there is a close correspondence between features in propositional languages and predicates in first-order languages. Consequently, there is a close correspondence between feature construction and predicate invention. However, there exists no systematic comparison between work done in both areas. Apart from one exception (DUCE [Muggleton 87] and CIGOL [Muggleton & Buntine 88]), work on predicate invention neither tries to extend ideas from propositional constructive induction to first-order logic, nor builds on existing work on feature construction. In this report we try to apply the frameworks for constructive induction by Matheus [Matheus & Rendell 89, Matheus 91] and Wnek and Michalski [Wnek & Michalski 94] to predicate invention.

In the next section, we recall the major reasons for constructive induction in propositional languages. In the third section we review the frameworks for constructive induction by Matheus, Wnek and Michalski. We then give a brief overview of PI methods and the types of predicates that the systems are able to invent. Subsequently, we apply the frameworks for constructive induction to PI in order to understand which relevant aspects distinguish existing PI systems. In the sixth section, we review current criticism against constructive induction that also concerns predicate invention. In the subsequent section, we discuss the advantages and disadvantages of fully automatic vs. interactive predicate invention. In eighth section, we discuss a selection of PI systems according to the aspects we identified in the previous sections. Finally, we draw our conclusions, and sketch possible directions of further research.

2 Motivations for Constructive Induction

In this section we want to recall the motivation for constructive induction in general, since it also applies to predicate invention.

Generally, the goal of constructive induction is an increase in accuracy and a decrease in complexity of a hypothesis [Wnek & Michalski 94]. It is important to include the complexity of the newly defined features or predicates in the complexity of the hypothesis — we have to pay a price for defining a large number of new features or predicates. Otherwise we would overfit the potential noise in the data by means of the language, a problem termed "language fitting" in [Pfahringer 94]. [Pfahringer 94] also suggests the usefulness of a function measuring both accuracy and complexity of a hypothesis *and* the definitions of the new features or predicates. This can be done by a measure based on the *Minimum Description Length* (MDL) principle [Rissanen 78] that measures both accuracy and complexity in a common currency, namely bits needed for encoding an inductive theory and the training examples, given the theory.

Two additional reasons for constructive induction are related to the *bias*[2] of learning systems, more precisely to a mismatch between the learning task and the learner. The first reason is concerned with a property of certain real-world datasets which makes learning hard for most conventional learning algorithms. Secondly, the algorithmic bias itself is a major reason why constructive induction is necessary for some datasets. As the discussion will show, both reasons are closely related.

2.1 Hard Concepts

Standard learning algorithms perform quite well in terms of accuracy and complexity, if domain experts can readily specify a favorable abstract representation language. Most of the work has already been done, and hence the selective induction algorithms can produce accurate and concise hypotheses without appearing to take advantage of domain knowledge. It is a well-known fact that the results of machine learning algorithms strongly depend on how well the domain is understood.

Whereas easy concepts can be learned quite well even by simple learners [Holte 93], Rendell claims that *hard concepts* [Rendell & Seshu 90] mostly cannot even be learned by sophisticated learners. Basically, a concept is *hard* if the

[2]The term "bias" refers to any basis for excluding hypotheses from the search space other than strict completeness and consistency with the examples [Mitchell 80]. An important part of the bias is the hypothesis language: it restricts the range of expressible concepts. The *language bias* is given by the *vocabulary* (the available predicate, function and constant symbols) and the syntactic form of the potential target theories. Another crucial part of the bias is the *algorithmic bias*, i.e. the way how the learner excludes hypothesis from the hypothesis space by its search strategy (other than by pruning).

disjuncts of the concept to be learned are "spread out" in the instance space, but it yet can be formulated with the given features. Since an accurate description would be extremely complex in terms of the original features, selective induction algorithms usually fail to learn hard concepts. Examples for hard concepts are the prediction of protein structure and the classification of chess positions as won or lost in terms of low-level features. It is important to note that in both example domains the information with respect to the target concept is complete: Besides the used variables, no other (possibly unknown) variables influence the target concept.

In domains with incomplete information, we cannot hope to get beyond the accuracy of very simple benchmark algorithms such as Holte's [Holte 93]: The independent variables may influence the value of the dependent variable, but do not "determine" it — there is simply not enough information available. However, this is not a matter of all-or-nothing: The "causal relevance" of variables rather is gradually different. Only in a few extreme domains, the causal connection between the independent and the dependent variables seems very weak. Sidestepping philosophical problems concerning causality, [Rendell & Seshu 90] discuss this issue under the heading of *intrinsic accuracy*.

In order to turn a hard problem into an easy one, useful abstractions have to be formed from the initial low-level features/predicates. One of the goals of constructive induction is to solve this problem. Especially interactive forms of constructive induction appear to be promising to gain a better understanding of the domain and to form useful abstractions.

According to Matheus, another reason for constructive induction related to real-world datasets is *feature interaction*. Feature interaction occurs if the features are not independent. More precisely, there are relations and dependencies between original features hidden in the data that may not be visible to the learner.

2.2 Algorithmic Bias

The other major reason for constructive induction is the algorithmic bias of learning algorithms. For instance, the *greedy selection* [Matheus 91] of features is a kind of bias that is problematic when applied to datasets with high feature interaction. Experiments showed that very complex hypotheses are generated [Pagallo & Haussler 90], since a large number of conditions are required to express interrelationships among features in the dataset.

From a theoretical point of view, the algorithmic bias is the only justification for feature construction in propositional learning[3]. For standard propositional learning algorithms using DNF-hypotheses or decision trees of arbitrary size,

[3]This is only true for hypotheses that may be arbitrarily large. For hypotheses of limited size, such as Holte's one-level decision trees, the construction of new features in fact enhances the set of expressible concepts.

the construction of new features does not change the basic ability to express a concept. In other words, the occurrence of the real concept in the hypothesis space is not affected by the construction of new features. So in the propositional setting emphasizing the *language bias* does not make sense except for fixed-size languages. Nevertheless, the way how a learner makes use of the features can be changed by an extended representation. For some learning problems, new features enable greedy learners to find generalizations that otherwise could not be found. Constructing binary features that represent relations among features makes feature interaction visible even to algorithms which greedily select features.

3 Constructive Induction Frameworks

According to Matheus, there are four major questions that a system must answer when performing constructive induction [Matheus & Rendell 89, Matheus 91]:

1. When should new features/predicates be constructed ? (*Detection*)

   Detection is necessary, because the construction of irrelevant features/predicates usually affects learning like noise in the data [Wnek & Michalski 94]. Moreover, the complexity of the definitions has to be added to the complexity of the hypothesis. Additionally, the complexity of the hypothesis space is increased by the definition of useless features/predicates, making search in this space error-prone and more costly in potential subsequent steps. Last, but not least, the inductive theory usually is interpreted by humans, and it will hardly be comprehensible if many definitions of useless features/predicates have to be considered.

2. Which potential new features/predicates shall be included in the transformed representation ? (*Selection*)

   Since the set of features/predicates that can be constructed is potentially very large and its detailed evaluation is intractable in general, only a small subset can be included in the representation.

   In the propositional case, this problem is subsumed by what now is discussed as *feature subset selection* (e.g. [Pfahringer 95]).

3. Shall the definitions be generalized ? (*Generalization*)

   Generalizing the definition might effect an improvement, if the newly defined term is too specific to be useful for the concept to be learned. Only few systems generalize the definitions of new features/predicates.

4. Which (if any) features/predicates should be discarded ? (*Evaluation*)

   Again, for propositional languages this is the same as feature subset selection.

Matheus further distinguishes detection, selection and evaluation with respect to the source of information they are based on. The potential sources of information are the instances of the training set, an existing hypothesis, knowledge provided by an expert and any possible combination. For instance, detection may be based on the initial set of training instances or on the analysis of a concept description. Another approach to detection is taken by CN2-MCI [Kramer 94]: The system basically performs a kind of look-ahead: it constructs new features and evaluates the results. If the quality of the hypothesis induced in the transformed representation is worse than the original hypothesis, CN2-MCI concludes there is no need for a representation change. As we will see below, a similar kind of look-ahead is done by systems for predicate invention.

[Wnek & Michalski 94] introduce a taxonomy of constructive induction systems according to the sources of information that are used by constructive induction operators. So the main distinction is different from the one in Matheus's framework, which is based on aspects like detection and selection.

- *Data-driven constructive induction (DCI):* DCI analyzes the training examples in order to perform constructive induction. Specifically, new descriptors are found by the search for interrelationships among examples, attributes and concepts.

- *Hypothesis-driven constructive induction (HCI):* HCI refers to methods of transforming representation spaces by analyzing generated inductive hypotheses.

  Methods for hypothesis-driven constructive induction typically construct new features/predicates in iterations, where each iteration involves a learning step and a step which constructs new features/predicates based on the hypotheses of the learning step.

- *Knowledge-driven constructive induction (KCI):* These systems apply expert-provided domain knowledge to construct new features/predicates. Furthermore, representation changes can be validated by a domain expert.

- *Multi-strategy constructive induction (MCI):* MCI systems combine different approaches to the transformation of description spaces.

4 Predicate Invention

4.1 What is Predicate Invention ?

Stahl informally explains Predicate Invention in the following way: In ILP the search space is restricted by both the syntactic form of target theories and the available vocabulary, i.e. the predicate, function and constant symbols. The vocabulary strongly affects if it is more or less easy to find the correct hypothesis, or

if it cannot be found at all. For instance, the predicate "daughter-in-law" can easily be learned if the predicate "parent-in-law" is known to the learner. However, learning "daughter-in-law" is (usually) harder if only the predicates "parent", "married", "female" and "male" are known. Even worse, the target predicate can obviously not be learned if nothing is known about existing marriages.

If the vocabulary is limited for a learning task at hand, predicate invention can be a means to successfully learn a concept that otherwise could not be learned. Predicates that make it possible to learn a given concept are called *necessary* [Ling 91]. Predicates that are not crucial but help to compress a theory are called *useful*.

4.2 Reformulation vs. Demand-Driven Approaches

In her overview article on predicate invention [Stahl 93], Stahl distinguishes between *reformulation approaches* and *demand-driven approaches* to predicate invention. Reformulation approaches introduce new intermediate predicates as a reformulation of an existing theory in order to express it more compactly. This is done in any case, not only if learning fails in the given representation. The knowledge base can be compressed either by *inverse resolution* or by a *schema-driven approach*. The input of compression algorithms are usually clauses, either instances of the training set or disjuncts of a previously learned or partial hypothesis.

In contrast to reformulation approaches, demand-driven systems try to detect situations where the given vocabulary is insufficient for the learning task at hand. Unfortunately, this problem is undecidable in first-order Horn logic [Stahl 95]. However, Stahl proved that the problem is decidable in fixed-size languages, and she also demonstrated that PI is *useful* in this context. Nevertheless, most systems have to take a heuristic decision (e.g. [Srinivasan et al. 92]) for pragmatic reasons anyway.

Stahl's classification is based on *two* different aspects of predicate invention: First, demand-driven and reformulation methods essentially differ in their approach to *detection*. Demand-driven systems need a reason for predicate invention, reformulation-based systems do not. The other distinction is based on different *methods* for PI, namely inverse resolution and the schema-based approach.

4.3 Methods for Predicate Invention

There are basically three kinds of methods for predicate invention:

- **Inverse Resolution** is a method for "factoring out" the generalization of two or more clauses, and assigning the "residues" to a new predicate. This results in a new predicate describing the variation of the input clauses

relatively to their common generalization. Note that in this way the new predicate is defined by several clauses, whereas it is only used once.

More precisely, several inverse resolution operators are theoretically possible. Two of them could be used for predicate invention, and mostly only one of them is used (as described above): *intraconstruction*.

Only RINCON [Wogulis & Langley 89] employs the other operator usable for PI, namely *interconstruction*. In contrast to intraconstruction, interconstruction puts the commonalities between the input clauses into the definition of the new predicate. The differences are kept in the clauses which then make use of the newly defined predicate. Note that the new predicate is used several times, whereas its definition is only conjunctive.

Inverting a resolution step in first-order logic involves computing substitutions and inverse substitutions. In particular, the computation of inverse substitutions involves the risk of combinatorial explosion, because we have to choose which subterms shall be replaced by one common variable. For this reason, inverse resolution methods are known to be inefficient in certain cases.

- **Schema-Driven Methods** utilize second-order clauses for predicate invention. Newly defined predicates are simply instantiations of predefined or induced schemata. A schema can be instantiated by turning its predicate variables into predicate symbols.

- **Clause-Refinement Methods** make an over-general clause consistent by adding a literal which contains a new predicate. Before the actual refinement step, we have to determine the clause of the theory which is to blame for the incorrectly covered instances. This blame assignment can be simple as in [Srinivasan et al. 92] or more sophisticated as in [Wrobel 93].

The third type of methods cannot be found in [Stahl 93] — it is a type we identified besides inverse resolution and schemata.

The kinds of methods differ in their use of negative examples. Clause-refinement methods depend on the existence of negative examples, because they shall discriminate positive and negative instances. Methods for inverse resolution only take positive instances or clauses of a theory as input. Methods for learning recursive new predicates e.g. by schemata shall characterize positive examples in a special way: They shall describe everything that the given examples have in common, namely their base-case and the repeated application of an operator.

4.4 Utility and Decidability of Predicate Invention

In the discussion of the utility of predicate invention the *language bias* plays a central role [Stahl 95]. The assessment of the utility and appropriateness of

predicate invention is based on the question if a predicate is necessary to express a concept. For instance, a newly invented recursive predicate is necessary to express a recursive concept, if the original language does not contain recursive concepts at all. According to this view, predicate invention is useful only if it makes possible to express the given concept.

Unfortunately, the question if predicate invention is necessary to successfully learn a theory is undecidable in first-order Horn logic [Stahl 95]. So in unconstrained or weakly constrained first-order Horn logic, predicate invention may be useful, but its necessity is undecidable. However, induction in weakly-constrained first-order languages is known not to be feasible anyway.

In function-free Horn logic, the problem if PI is needed is decidable, but PI is a useless bias shift operation. New predicates can be eliminated without affecting the success set of the theory.

The languages for which predicate invention is useful and still decidable are restricted to a fixed size. The new predicates are useful because they cannot be eliminated without violating the size constraints of the language. The decidability stems from the finite search space.

As experience from propositional languages suggests, exhaustive search for deciding if PI is necessary is not realistic. Hence, a heuristic decision has to be taken for pragmatic reasons. We think the usefulness of predicate invention should be related to the algorithmic bias of learning algorithms, as it is done in the motivation of feature construction. For standard propositional learning algorithms using DNF-hypotheses or decision trees of unconstrained size, the construction of new features does not change the basic ability to express a concept. Still, feature construction is useful in practice to avoid known problems of certain learning algorithms.

Generally, literature on predicate invention has put more emphasis on the language bias than on the algorithmic bias of ILP systems. We believe that both of them are equally important parts of the overall bias of a learning system.

4.5 Types of Predicates

In this subsection, we try to sketch a taxonomy of the predicates invented by various systems. In Table 1 we relate this taxonomy to the systems we want to examine in this report.

- *Extensional*:
  The tuples of the predicates are given extensionally. Predicate invention often is extensionally driven: We assume some individuals belong together, and need a name to refer to them in a simple fashion. In some cases, we do this by drawing a border between two groups of individuals, assuming the separation is useful or necessary for some reason. The assumption of such a border can be seen as a hypothesis by itself.

- *Intensional without Recursion*:
  The predicates are intensionally defined, but without recursion. The definition describes the set of tuples in the relation. Intensional descriptions are what [Thagard & Nowak 90] call *concept combinations*. That is, a new concept is defined as a combination of existing concepts. Concepts can be combined in numerous ways. In many cases constraints on meaningful combinations are beyond the "knowledge" that is available to the machine. Sometimes selecting the right and interesting combinations requires a thorough understanding of the particular domain.

  1. *Conjunction of 2 Literals*:
     The predicates in [Silverstein & Pazzani 93] are restricted to conjunctions of two literals. This restriction is due to efficiency reasons.
  2. *Conjunction of n Literals*:
     For instance, RINCON's [Wogulis & Langley 89] intermediate concepts consist of a deliberate number of literals.
  3. *Disjunctive Definition*:
     Systems using the intraconstruction operator [Muggleton & Buntine 88, Rouveirol 92, Wirth 89] construct predicates with disjunctive definitions.

- *Recursive Predicate*:
  Several systems [Ling 91, Lapointe et al. 93, Ling 95] are designed to invent recursive new predicates. Ling states that recursive predicates are the only ones that are really *necessary*, because they cannot be eliminated without failing to learn certain concepts. Furthermore, they are *useful* because they often allow for dramatic reductions in size in cases of unbounded growth of a theory.

  Other kinds of predicates are not necessary in the strict sense, but nevertheless useful. For example, clichés are useful to overcome the algorithmic bias of FOCL (its selection of literals) [Pazzani & Kibler 92, Silverstein & Pazzani 91].

Furthermore, the semantics of the new predicate possibly reveals a dependency between variables:

- *Functional Dependency*: New predicates can be used to express functional dependencies.

- *Multi-Valued Dependency*: New predicates can express multi-valued dependencies, i.e. dependencies that assign a *set of values* instead of only one value (as in the case of functional dependencies) to the values of the input variables.

	Reference	Ext.	Conj.	2 Lits.	Disj.	Rec.	Deps.
CIGOL	[Muggleton & Buntine 88]				×		
LFP2	[Wirth 89]				×		
ITOU	[Rouveirol 92]				×		
Banerji	[Banerji 92]				×		
RINCON	[Wogulis & Langley 89]		×				
INPP	[Ling 95]					×	
CILP	[Lapointe et al. 93]					×	
CLINT-CIA	[De Raedt & Bruynooghe 92]		×				
FOCL	[Silverstein & Pazzani 93]			×			
GOLEM	[Muggleton 94]		×				
CWS	[Srinivasan et al. 92]	×	×	×	×		
MOBAL	[Wrobel 94]	×	×	×	×	×	
CHAMP	[Kijsirikul et al. 92]	×	×	×	×	×	
CHILLIN	[Zelle et al. 94]	×	×	×	×	×	
SIERES	[Wirth & O'Rorke 92]		×	×	×		
INDEX	[Flach 93]						×

Figure 1: Types of predicates that can be invented by PI systems. (Ext.-extensional, Conj.-conjunctive, 2 Lits.-a conjunction of two literals, Disj.-disjunctive, Rec.-recursive, Deps.-dependencies among arguments.)

In INDEX [Flach 93], new predicates are either functional respectively multi-valued dependencies or represent partitionings of tuples in a relation.

- *Partial Determination*: Partial determinations are like functional dependencies, but allow for exceptions. If exceptions are likely to arise in the presence of noise, partial determinations are a useful dependency model [Pfahringer & Kramer 95].

It should be noted that an extensional definition of predicates can be turned into an intensional definition by a selective induction algorithm. This step is performed by the systems MOBAL [Morik 93, Wrobel 94], CHAMP [Kijsirikul et al. 92] and CWS [Srinivasan et al. 92]. Learning an intensional definition is in fact generalization in the sense of Matheus. The definitions could further be generalized by dropping a condition of the definition.

Figure 1[4] shows that many kinds of predicates are used, and also that only few systems are capable of inventing predicates of several types. CHAMP and

[4]Not considered in this report are the predecessors of several systems: IRES [Rouveirol & Puget 90] (the predecessor of ITOU [Rouveirol 92]), LFP [Wirth 88] (the predecessor of LFP2 [Wirth 89]), MENDEL [Ling 91] (the predecessor of INPP [Ling 95]), BLIP [Morik 89, Wrobel 89] (the predecessor of MOBAL [Morik 93, Wrobel 94]). Furthermore, the so-called transformation approaches (creating predicates from formal specifications) to PI (e.g. [Franova & Kodratoff 92, Le Blanc 94]) are not discussed, as they are only partially relevant in the inductive learning setting. For the same reason we omitted the discussion of FENDER [Sommer 95], a component of MOBAL that restructures a knowledge base by certain inverse resolution operators.

CHILLIN are the most flexible systems [5] with respect to the variety of types of predicates that can be invented.

4.6 Some Considerations on Predicate Invention

PI is often triggered by the need to name a group of individuals. However, defining a new concept only pays off if we make use of the name many times. Normally, we cannot know beforehand if a new concept turns out to be useful in practice. This is the reason why Zytkow [Zytkow 93] metaphorically calls new terms "investments". These considerations obviously favor a process that is iterative rather than a process that stops after a single iteration of constructive induction.

One of the main problems of PI is to find patterns and subpatterns in the data or in a theory that are not due to chance. If we find such patterns, they will help to compress the theory, because they will help to subsume a lot of observations in a certain respect. Significant patterns help to describe parts of observations. Ling emphasized the benefits of new recursive predicates: They are useful in their capability of enumerating and compressing data. Stahl also emphasized the role of compression by proving that PI theoretically only makes sense to overcome size-bounds in fixed-size languages. Summing up, we believe that compression is a major motivation for PI.

Nevertheless, the goal of the overall system strongly affects the use of PI. For instance, the goal of *interactive theory revision* affects the way of PI in MOBAL [Wrobel 94]. Likewise, the goal of *inductive data engineering* [Flach 93] needs a different strategy for the invention of new predicates than employed by other systems.

5 Applying Constructive Induction Frameworks to Predicate Invention

In this section we will first show how predicate invention can be discussed within Matheus' framework for constructive induction: In the first subsections we review how aspects like detection and selection are present in existing PI systems. Subsequently, we apply the framework of [Wnek & Michalski 94] to PI. Finally, we summarize the benefits of the application of those frameworks to predicate invention.

[5] MOBAL is able to construct recursive predicates given the required rule models. However, recursion does not seem to make much sense for MOBAL, as the representation is function-free.

5.1 Detecting the Need for Predicate Invention

Stahl's distinction between demand-driven and reformulation approaches is essentially based on detection. The task of recognizing the demand for predicate invention is nothing else than detection. For pragmatic reasons, systems have to decide heuristically if the existing vocabulary is insufficient. Reformulation-based approaches perform a kind of look-ahead, searching for new predicates that make the theory more compact.

Demand-driven systems like CWS [Srinivasan et al. 92] and CHAMP-DBC [Kijsirikul et al. 92] detect the need for predicate invention, if the top-down algorithm fails to discriminate the tuples covered by the clause. The over-general clauses are handed over to the algorithm for predicate invention. Note that the component for predicate invention can be used without other parts of the system. Among others, this scenario also highlights the relationship between predicate invention and *theory revision* [Le Blanc 94]: A theory with exceptions can be corrected by adding literals with new predicates to the over-general clauses in order to separate those instances that are incorrectly covered from those that are correctly covered. More sophisticated techniques from theory revision can be used to determine which clause is really to be blamed for an incorrect classification.

A major task in this process is to decide if the incorrectly covered instances are due to noise, or if they are real exceptions [Srinivasan et al. 92]. If they were due to chance, inventing a new predicate would overfit the noise. As in feature construction, we have to control the complexity of the theory *and* the newly defined predicates.

Correcting a clause in this way is often used together with top-down algorithms [Srinivasan et al. 92, Kijsirikul et al. 92], but does not presuppose their use. For instance, the predicate invention component of GOLEM [Muggleton 94] works in a similar way, correcting bottom-up generated clauses. CHILLIN [Zelle et al. 94] combines bottom-up and top-down techniques: First, it computes the *least general generalization (lgg)* of two clauses, then it tries to make the generalization consistent by specializing it. If this fails, predicate invention is performed to make the clause consistent.

SIERES [Wirth & O'Rorke 92] triggers predicate invention if no extension of the clause exists that is correct with respect to *argument dependency graphs* and *critical terms*, i.e. unused input or unbound output terms.

Generally, this kind of detection puts the blame on the language bias, if the algorithmic bias causes the algorithm to fail. In many cases the over-general clause was built by greedy hill-climbing, so we believe there are alternatives to predicate invention, e.g. backtracking to a previous decision.

5.2 Selection of Predicates

Systems for PI implicitly select a new predicate from a large number of possible predicates. Selection can be distinguished with respect to the method for predicate invention.

Inverse resolution methods have many degrees of freedom in their choice of appropriate substitutions and inverse substitutions. Most systems a priori restrict the possibilities by simplifying assumptions. For instance, one operator of CIGOL [Muggleton & Buntine 88] is simplified by assuming that one input clause and the output clause have no common literal (*separability assumption*). This and other restrictions determine which of the potential new predicates are included in the vocabulary.

Schema-driven methods have to include selection, if several second-order schemata apply. In CLINT [De Raedt & Bruynooghe 92] the user is responsible for selection. In FOCL [Silverstein & Pazzani 93], selection takes place in two steps: First, the selection of candidates for clichés is based on the occurrence in an inductive hypothesis and on the information gain measure [Quinlan 90]. The next step is a complicated procedure to integrate the candidate cliché in a class hierarchy of existing clichés. During this second step further candidate clichés are dropped.

For the case of clause-refinement methods, we have to face a theory that is too general and therefore has to be specialized with a new predicate. The first task is to select the set of arguments for the new predicate. [Srinivasan et al. 92] select the same arguments as in the head of the over-general clause. CHAMP [Kijsirikul et al. 92] starts with all variables and drops variables as long as there are enough to discriminate the positive and the negative instances. As Stahl [Stahl 94] points out, searching for minimum sets of discriminating arguments has a subjective component, since there might be several such sets. The intersection of all minimal discriminating sets of arguments roughly corresponds to the *core* as defined in rough set theory [Ziarko 92]. The core of arguments is in any case necessary to discriminate the examples. Unfortunately, the core is empty if the noise level is sufficiently high. We therefore believe that a robust measure like the MDL measure in [Pfahringer & Kramer 95] could be more successfully applied to the problem of finding good sets of arguments in the presence of noise than rough set theory.

5.3 Generalizing the Definition of a New Predicate

Only a few systems perform a generalization step after the initial definition of a new predicate: CHAMP [Kijsirikul et al. 92], CWS [Srinivasan et al. 92] and MOBAL [Wrobel 94]. An extensional definition can easily be generalized by applying a relational learning algorithm to the tuples of the definition.

CIGOL only generalizes its definitions if this step helps to compress the knowl-

edge base. Systems like CILP define recursive predicates without considering extensional definitions as an alternative. Furthermore, it is not obvious if it makes sense at all to generalize recursive new predicates.

If we do not include a generalization step, we implicitly accept the generality of the initial definition. However, the desired generality strongly depends on the particular context of the learning task. Thus, considering the generalization of definitions makes the learner more flexible in learning situations.

5.4 Evaluating the Set of Existing Predicates

MOBAL [Morik 93] is the only system that also evaluates its newly defined theoretical vocabulary. Therefore, it is the only system that considers all four aspects of Matheus' framework.

The reason for this step is the need to check our theoretical vocabulary from time to time, since otherwise the complexity of the theory would grow more than is necessary. Hence, neglecting the evaluation of the vocabulary is a serious shortcoming of existing systems.

5.5 Applying the Framework of Wnek and Michalski to Predicate Invention

Inverse resolution methods do not distinguish between example clauses and clauses that are generalizations. Therefore, inverse resolution methods are based on both examples and a partial hypothesis. Consequently, inventing predicates by inverse resolution is *multi-strategy constructive induction (MCI)*.

Clause refinement methods mostly process a single clause of a theory and the set of examples that are covered by the clause. Thus, constructive induction by clause refinement is also multi-strategy constructive induction (MCI).

Schema-driven methods have to be discussed separately. CILP is the only system that performs *data-driven constructive induction (DCI)*. In CLINT-CIA [De Raedt 91] the instantiation of a schema is data-driven, and the subsequent suggestion of the new predicate is accepted or rejected by a human expert. Since it is based on two sources of information, CLINT-CIA is a method for multi-strategy constructive induction (MCI), too.

With respect to this framework, FOCL is the most interesting system. In the first step it explicitly analyses a hypothesis. So it would be tempting to categorize it as the only PI system performing *hypothesis-driven constructive induction (HCI)*. However, FOCL subsequently evaluates pairs of literals by their information gain, and this computation requires looking through the data. So FOCL is not a system for HCI, but also for MCI.

The framework of [Wnek & Michalski 94] does not reveal fundamental differences and commonalities among the systems under consideration. The methods

primarily differ in how strongly parts of the hypothesis and subsets of examples influence the construction of a new predicate. Interestingly, there is no method that constructs a new predicate merely based on the analysis of a hypothesis, and only one method based on data alone (CILP). In existing systems, the process is almost always related to partial or over-general hypotheses *and* example sets.

5.6 Benefits from Applying Constructive Induction Frameworks to Predicate Invention

The benefits of the application of frameworks for propositional constructive induction can be stated as follows:

- The distinction demand-driven/reformulation is in fact based on *detection*.
- Many systems *implicitly select* from a large number of possible new predicates.
- Important aspects like generalization and evaluation are largely neglected in existing systems, and deserve more attention.
- Almost all systems utilize information from the data as well as information from a partial or over-general hypothesis. Experience from the propositional setting, however, suggests that pure HCI-methods [Pagallo & Haussler 90, Wnek & Michalski 94] are worth consideration, too. Successful work in this field should be extended to first-order logic.

Summing up, the application of the frameworks suggests directions of further research, and shows that the mentioned aspects of CI should be taken seriously in PI systems.

6 Criticism against Constructive Induction and Continuing Learning

While everyone agrees that a good representation is crucial for obtaining good learning results, work on constructive induction has to face serious criticism that also concerns predicate invention.

Schaffer [Schaffer 94] proved that the generalization performance of any inductive learning algorithm over all learning situations is null. Positive performance in some learning situations must be offset by an equal degree of negative performance in others. Roughly speaking, Schaffer's result reminded us that the whole task of machine learning research is to find biases that work well when applied to real-world datasets.

In the light of his result, Schaffer discusses constructive induction as an attempt to enhance a learner. Since every attempt to enhance a learner without simultaneously impairing the generalization performance in other learning situations must fail, constructive induction also has to fail in this respect. However, the theorem makes clear that constructive induction just makes learning algorithms more flexible, a flexibility that does not guarantee any improvement.

Sutton [Sutton 94] argues that constructive induction is doomed to effect only minor improvements with a single learning task at hand. As a solution he proposed a new methodology for constructive induction that is based on continuing learning (as outlined in [Sutton 92]). Constructive induction will effect major improvements when the learner is confronted with a sequence of learning tasks. These tasks may have different solutions, but supposedly often share the same useful representations.

Note that Sutton's idea is especially interesting with respect to Schaffer's theoretical results: The success of continuous learning critically depends on the similarity between learning tasks from the real world. We need to employ biases that work for real-world applications, so we should learn from these applications to improve the performance on subsequent learning tasks by analogy.

These ideas have yet to prove their usefulness. Strictly, only two existing systems implement this idea (learning across domain borders): FOCL extended by the capability to learn clichés [Silverstein & Pazzani 93], and MOBAL-MAT [Thieme 89], a component of MOBAL that learns rule models from rules. Furthermore, CLINT-CIA could be used in this way, but it seems that it only has been tested in single domains.

Several systems are able to utilize schemata (possibly acquired in another domain) for learning. In fact, using such schemata or rule models is the basic principle of MOBAL. Moreover, work on second-order learning (e.g. [Hamfelt & Nilsson 94]) and on schema-driven approaches to predicate invention deals with the problem of "retrieving" schemata.

ILP methods are apparently well suited for this task, since a lot of structure is available for learning useful patterns in the representation. For instance, type information, levels of existential quantification, modes and schemata could be utilized to find similarities between useful representations.

7 Autonomy vs. Human Intervention in Predicate Invention

The original goal of constructive induction was to *automatically* transform the initial representation into a representation usable by the learner. A learner using constructive induction shall detect if a shift to a better bias is required and, if so, automatically transform the representation. In this scenario, the learner shall

find a suitable representation on his own. This amounts to a higher degree of autonomy, and turns a learner into a discoverer [Zytkow 93]. In this section we want to discuss the advantages and disadvantages of autonomy in contrast to human intervention in predicate invention. There are two kinds of systems for predicate invention:

1. Systems like GOLEM [Muggleton 94] perform constructive induction without human intervention. Constructive induction takes over in order to improve the best "human" representation.

2. Systems like CLINT [De Raedt & Bruynooghe 92] and CIGOL [Muggleton & Buntine 88] perform *interactive constructive induction*, a kind of constructive induction that is not performed fully automatically. Instead, a constructive induction component proposes the application of constructive induction operators to an oracle. The oracle rejects or accepts, and in the latter case has the opportunity to name the new feature or predicate. This procedure ensures that the new features/predicates are meaningful to the user [Muggleton 87].

The scenario for interactive constructive induction has a great potential for helping to learn hard concepts. It is exploratory in nature, and should help to gain a better understanding of poorly understood domains. The exploration of the data shall help to find the abstractions that are necessary for the learning of hard concepts.

Furthermore, this scenario is best suited for reusing patterns of useful representations, much in the spirit of Sutton's idea of performing constructive induction in a sequence of learning tasks.

The scenario was first realized by DUCE [Muggleton 87], a system that transforms a propositional knowledge base via a number of operators. The system searches for the operation achieving the best compression. If the operation is not truth-preserving, it is proposed to an oracle, otherwise it is performed without asking. If the application of a *constructive induction operator* is proposed, the oracle is asked to reject or accept the new feature, and, in the latter case, is asked to name it.

In the same manner, CIGOL [Muggleton & Buntine 88] (the successor of DUCE) works in first-order logic. CIGOL was the first system based on inverse resolution, and employed three operators to compress a given knowledge base.

In general, the fundamental question is if and how we can create meaningful new predicates. Comprehensibility and syntactical complexity might be correlated, but surely are not the same. Basically, we think that this decision should be based on who the "user" of the theories is. Human intervention may be necessary if humans shall make sense of the resulting theory.

	CIGOL	LFP2	ITOU	Banerji
Selective Induction Algorithm: Top-down (TD), Bottom-up (BU)	BU	BU	BU	BU
When ? Detection: Reformulation (R), Demand-driven (DD)	R	R	R	R
How ? Method: Inverse resolution (IR), Schema-driven (SD), Clause-Refinement (CR)	IR	IR	IR	IR
How much Input ? n Clauses	2	≥ 2	2	2
Generalization ?	no	yes	no	no
Compression-based ?	yes	yes	no	no
Meta-Learning ?	no	no	no	no
Human Intervention	yes	no	yes	yes
	RINCON	INPP	CILP	CLINT-CIA
Selective Induction Algorithm: Top-down (TD), Bottom-up (BU)	BU	BU	TD	TD/BU
When ? Detection: Reformulation (R), Demand-driven (DD)	R	DD	DD	R
How ? Method: Inverse resolution (IR), Schema-driven (SD), Clause-Refinement (CR)	IR	IR,SD	SD	SD
How much Input ? n Clauses	1	≥ 2	-	1
Generalization ?	no	yes	no	no
Compression-based ?	yes	yes	no	no
Meta-Learning ?	no	no	no	yes
Human Intervention	no	no	no	yes
	FOCL	GOLEM	CWS	MOBAL
Selective Induction Algorithm: Top-down (TD), Bottom-up (BU)	TD	BU	TD	TD
When ? Detection: Reformulation (R), Demand-driven (DD)	R	DD	DD	DD
How ? Method: Inverse resolution (IR), Schema-driven (SD), Clause-Refinement (CR)	SD	CR	CR	CR
How much Input ? n Clauses	1	1	1	1
Generalization ?	no	yes	yes	yes
Compression-based ?	no	no	yes	no
Meta-Learning ?	yes	no	no	yes
Human Intervention	no	no	no	yes
	CHAMP	CHILLIN	SIERES	INDEX
Selective Induction Algorithm: Top-down (TD), Bottom-up (BU)	TD	BU/TD	BU/TD	-
When ? Detection: Reformulation (R), Demand-driven (DD)	DD	DD	DD	-
How ? Method: Inverse resolution (IR), Schema-driven (SD), Clause-Refinement (CR)	CR	CR	CR	-
How much Input ? n Clauses	1	1	1	-
Generalization ?	yes	yes	yes	-
Compression-based ?	yes	yes	no	no
Meta-Learning ?	no	no	no	no
Human Intervention	no	no	no	yes

Figure 2: An overview of systems for predicate invention

8 Discussion

In this section, we systematically compare existing systems for PI according to the aspects we identified in the previous sections. The comparison will be supported by the summary of all systems under consideration in figure 2. (The references of the systems can be found in figure 1.) In figure 2, systems are distinguished by the following properties:

- **Selective Induction Algorithm**
  The selective induction algorithms of the systems mostly employ top-down (specializing the most general theory) or bottom-up (generalizing the most specific theory) search, but several systems combine top-down and bottom-up strategies. The strategy is usually independent of the constructive induction operator, but mostly the selective induction algorithm and the constructive induction operator are similar or related.

- **When ? (Detection)**
  Detection was discussed in the section on the application of Matheus' framework.

- **How ? (Method)**
  We distinguished three types of methods for predicate invention in Section 4: Inverse resolution methods, schema-driven methods and clause refinement methods.

- **How much input ?**
  Methods for PI differ in the number of clauses that are processed by the constructive induction operator.

- **Generalization**
  We discussed the generalization of definitions in Section 5. Generalization is an important issue in predicate invention, but only few systems include a generalization step.

- **Compression-based**
  Compression-based measures are a good means of avoiding overfitting in the presence of noise. Systems like CHAMP, CWS, and INPP have a compression-based criterion for detection. In contrast to these systems, the *overall goal* of CIGOL and CHILLIN is compression. RINCON in a way tries to compress the knowledge base by intermediate concepts, but does not use a compression-based measure.

- **Meta-Learning**
  As we said in Section 6, meta-learning and continuing learning could be useful to overcome inherent limitations of current constructive induction systems.

- **Human Intervention**
  In section 7 we argued that human intervention may be necessary, depending on the usage of the theories. However, it should be clear that human intervention has its limitations as well, such as the cognitive overhead involved.

Figure 2 shows that inverse resolution methods mostly employ a reformulation approach. An interesting exception to this rule is INPP, an inverse resolution system that is demand-driven: If the theory size grows beyond a certain threshold, INPP decides to invent a recursive new predicate by inverse resolution and a subsequent schema-driven step. Only those schema-driven approaches which create new recursive predicates are demand-driven (INPP, CILP). As expected, all clause refinement methods are demand-driven.

Figure 2 also demonstrates that most systems apply their operator for predicate invention to maximal two input clauses. (Since CILP is data-driven, its input does not contain clauses of a partial hypothesis at all.) Mostly the resulting predicates are invented just for one or two clauses. These predicates are quite *ad hoc*, and they are not likely to capture significant and meaningful patterns in the data.

INDEX [Flach 93] is not really comparable to other systems, since its goal is restructuring a relational database, and not concept learning. So there is no selective induction algorithm, and no detection if predicate invention is needed. However, other PI systems can learn from INDEX its way to invent new predicates with a special semantics, namely dependencies among arguments.

Compression-based learning, meta-learning and human intervention are known to be ingredients of successful methods. Nevertheless figure 2 indicates that no system actually integrates all three of them. Future systems should integrate several such elements that are currently used in isolation.

9 Conclusion and Further Research

In this report we argued that predicate invention can benefit from existing work in propositional constructive induction in several ways:

- First, the general motivation for constructive induction also applies to PI. In particular, the notion of "hard concepts" should be kept in mind when considering PI. However, the notion of hard concepts has to be re-thought for relational domains, since it is not clear how structural examples are "spread out" in the instance space.

- We emphasized the role of the algorithmic bias instead of the language bias.

- We applied frameworks for propositional constructive induction to predicate invention. Having identified important aspects and building blocks of

predicate invention in this way, we are ready to select and combine features of these systems in order to put predicate invention to work.

- The idea that CI should be based on continuing learning should be adapted by work on PI. (Interestingly, FOCL [Silverstein & Pazzani 93] anticipated this idea.)

- Another lesson from propositional constructive induction is that we have to take care of the complexity. We have to avoid what is called "language fitting" in [Pfahringer 94] . Robust measures are needed to avoid fitting insignificant patterns in the theory or in the data. Except for CWS [Srinivasan et al. 92], no system appears to account for noisy data.

- Systems for PI tend to construct insignificant predicates. New predicates are invented to complete a single clause, or they are used only once. Two exceptions are RINCON [Wogulis & Langley 89] and MOBAL [Wrobel 94]. RINCON chooses the one conjunctive concept that allows to rewrite as many other concept definitions as possible. MOBAL evaluates new predicates by a number of structural properties, including the usage of the new predicate in other clauses.

Basically, there are two possible directions of further research: First, the *interactive* approach could be further developed, aiming at the support of *data engineering* for inductive learning (which is explorative in nature) and for inductive data engineering [Flach 93]. For this purpose it would be promising to integrate compression-based and second-order learning approaches.

The second challenge is to build autonomous machine discoverers, which are capable of creating and transforming intermediate concepts and theoretical terms according to their needs. These discoverers would act in a continuing learning situation.

Although constructive induction was first tackled in the late seventies, it remains a vital and interesting problem. This is due to the fact that finding a suitable representation is crucial for every application of machine learning algorithms. Predicate invention is the next step in this development, but has not yet been applied successfully in practice. We think that predicate invention has still a long way to go before it will successfully be applied to hard real-world problems.

References

[Banerji 92] Banerji R.B.: Learning Theoretical Terms, in Muggleton S.(ed.), *Inductive Logic Programming*, Academic Press, London, U.K., pp.93-112, 1992.

[De Raedt & Bruynooghe 89] De Raedt L., Bruynooghe M.: Constructive Induction by Analogy, in Segre A.M.(ed.), *Proceedings of the Sixth International Workshop on Machine Learning*, Morgan Kaufmann, Los Altos, CA, pp.476-477, 1989.

[De Raedt 91] De Raedt L.: *Interactive Concept-Learning*, Ph.D. Thesis, Katholieke Universiteit Leuven, 1991.

[De Raedt & Bruynooghe 92] De Raedt L., Bruynooghe M.: Interactive Concept-Learning and Constructive Induction by Analogy, *Machine Learning*, 8(2), 1992.

[Flach 93] Flach P.A.: Predicate Invention in Inductive Data Engineering, in Brazdil P.B.(ed.), *Machine Learning: ECML-93*, Springer, Berlin, pp.83-94, 1993.

[Franova & Kodratoff 92] Franova M., Kodratoff Y.: *Predicate Synthesis from Formal Specifications: Using Mathematical Induction for Finding the Preconditions of Theorems*, Rapport de Recherche No.781, L.R.I., Univ. de Paris-Sud, 1992.

[Hamfelt & Nilsson 94] Hamfelt A., Nilsson J.F.: Inductive Metalogic Programming, in *Proceedings of the Fourth International Workshop on Inductive Logic Programming (ILP-94)*, GMD-Studien Nr. 237, pp.85-96, 1994.

[Holte 93] Holte R.C.: Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, *Machine Learning*, 11(1), 1993.

[Kietz & Wrobel 92] Kietz J.-U., Wrobel S.: Controlling the Complexity of Learning in Logic through Syntactic and Task-Oriented Models, in Muggleton S.(ed.), *Inductive Logic Programming*, Academic Press, London, U.K., pp.335-359, 1992.

[Kijsirikul et al. 92] Kijsirikul B., Numao M., Shimura M.: Discrimination-Based Constructive Induction of Logic Programs, in *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park, pp.44-49, 1992.

[Kramer 94] Kramer S.: CN2-MCI: A Two-Step Method for Constructive Induction, *Proceedings of the Workshop on Constructive Induction and Change of Representation*, 11th International Conference on Machine Learning (ML-94/COLT-94), New Brunswick, New Jersey, 1994.

[Lapointe et al. 93] Lapointe S., Ling C., Matwin S.: Constructive Inductive Logic Programming, in Bajcsy R.(ed.), *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, pp.1030-1036, 1993.

[Le Blanc 94] Le Blanc G.: BMWk Revisited - Generalization and Formalization of an Algorithm for Detecting Recursive Relations in Term Sequences, in Bergadano F. & Raedt L.de(eds.), *Machine Learning: ECML-94*, Springer, Berlin, pp.183-197, 1994.

[Ling 91] Ling C.: *Inventing Necessary Theoretical Terms in Scientific Discovery and Inductive Logic Programming*, Report No. 302, Dept. of Computer Science, University of Western Ontario, London, Ontario, 1991.

[Ling 95] Ling C.: Introducing New Predicates to Model Scientific Revolution, to appear in: *International Studies in the Philosophy of Science*, 9(2), 1995.

[Ling & Narayan 91] Ling C., Narayan M.A.: A Critical Comparison of Various Methods Based on Inverse Resolution, in Birnbaum L.A. & Collins G.C.(eds.), *Machine Learning: Proceedings of the Eighth International Workshop (ML91)*, Morgan Kaufmann, San Mateo, CA, pp.168-172, 1991.

[Matheus & Rendell 89] Matheus C.J., Rendell L.A.: Constructive Induction On Decision Trees, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, Los Altos, CA, 645-650, 1989.

[Matheus 91] Matheus C.J.: The Need for Constructive Induction, in Birnbaum L.A. & Collins G.C.(eds.), *Machine Learning: Proceedings of the Eighth International Workshop (ML91)*, Morgan Kaufmann, San Mateo, CA, pp.173-177, 1991.

[Michalski 83] Michalski R.S.: A Theory and Methodology of Inductive Learning, in Michalski R.S., et al.(eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, CA, pp.83-134, 1983.

[Michalski 93] Michalski R.S.: Inferential Theory of Learning as a Conceptual Basis for Multistrategy Learning, in Special Issue on Multistrategy Learning, *Machine Learning*, 11(2/3), 1993.

[Mitchell 80] Mitchell T.M.: The Need for Biases in Learning Generalizations, in Shavlik J.W., Dietterich T.G.(eds.), *Readings in Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1990.

[Morik 89] Morik K.: Sloppy Modeling, in Morik K.(ed.), *Knowledge Representation and Organization in Machine Learning*, Vol. 347 of Lecture Notes in Artificial Intelligence, Springer, Berlin, pp.107-134, 1989.

[Morik 93] Morik K.: Balanced Cooperative Modeling, in Special Issue on Multistrategy Learning, *Machine Learning*, 11(2/3), 1993.

[Muggleton & Feng 90] Muggleton S., Feng C.: Efficient Induction of Logic Programs, in Muggleton S.(ed.), *Inductive Logic Programming*, Academic Press, London, U.K., pp.281-298, 1992.

[Muggleton 87] Muggleton S.: Duce, An Oracle-Based Approach to Constructive Induction, in *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, Morgan Kaufmann, Los Altos, CA, p.287-292, 1987.

[Muggleton 94] Muggleton S.: Predicate Invention and Utilization, in Special Issue: Algorithmic Learning Theory, *JETAI Journal of Experimental and Theoretical Artificial Intelligence*, 6(1), 1994.

[Muggleton 92] Muggleton S.: *Inductive Logic Programming*, Academic Press, London, U.K., 1992.

[Muggleton 92] Muggleton S.: Inductive Logic Programming, in Muggleton S.(ed.), *Inductive Logic Programming*, Academic Press, London, U.K., 1992.

[Muggleton & Buntine 88] Muggleton S., Buntine W.: Machine Invention of First-Order Predicates by Inverting Resolution, in Laird J.(ed.), *Proceedings of the Fifth International Conference on Machine Learning*, Univ.of Michigan, Ann Arbor, June 12-14, Morgan Kaufmann, San Mateo, CA, pp.339-352, 1988.

[Pagallo & Haussler 90] Pagallo G., Haussler D.: Boolean Feature Discovery in Empirical Learning, *Machine Learning*, 5(1), 71-100, 1990.

[Pazzani & Kibler 92] Pazzani M., Kibler D.: The Utility of Knowledge in Inductive Learning, in *Machine Learning*, 9, 57-94, 1992.

[Pfahringer 94] Pfahringer B.: Controlling Constructive Induction in CiPF: An MDL Approach, in Bergadano F. & Raedt L.de(eds.), *Machine Learning: ECML-94*, Springer, Berlin, pp.242-256, 1994.

[Pfahringer 95] Pfahringer B.: Compression-Based Feature Subset Selection, *Proceedings of the IJCAI-95 Workshop on Data Engineering for Inductive Learning*, Montreal, Canada, 1995.

[Pfahringer & Kramer 95] Pfahringer B., Kramer S.: Compression-Based Evaluation of Partial Determinations, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, AAAI Press, 1995.

[Quinlan 90] Quinlan J.R.: Learning Logical Definitions from Relations, *Machine Learning*, 5, 239-266, 1990.

[Rendell & Seshu 90] Rendell L.A., Seshu R.: Learning Hard Concepts through Constructive Induction: Framework and Rationale, *Computational Intelligence*, 6, 247-270, 1990.

[Rissanen 78] Rissanen J.: Modeling by Shortest Data Description, *Automatica*, 14, 465-471, 1978.

[Rouveirol & Puget 90] Rouveirol C., Puget J.F.: Beyond Inversion of Resolution, in Porter B.W. & Mooney R.(eds.), *Machine Learning*, Morgan Kaufmann, Los Altos, CA, pp.122-131, 1990.

[Rouveirol 92] Rouveirol C.: ITOU: Induction of First-Order Theories, in Muggleton S.(ed.), *Inductive Logic Programming*, Academic Press, London, U.K., 1992.

[Schaffer 94] Schaffer C.: A Conservation Law for Generalization Performance, in *Proceedings of the Eleventh International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA, pp.259-265, 1994.

[Silverstein & Pazzani 91] Silverstein G., Pazzani M.J.: Relational Clichés: Constraining Constructive Induction During Relational Learning, in Birnbaum L.A. & Collins G.C.(eds.), *Machine Learning: Proceedings of the Eighth International Workshop (ML91)*, Morgan Kaufmann, San Mateo, CA, pp.203-207, 1991.

[Silverstein & Pazzani 93] Silverstein G., Pazzani M.J.: Learning Relational Clichés, in Bergadano F., et al., *Proceedings of the IJCAI-93 Workshop on Inductive Logic Programming*, Chambery, France, 1993.

[Sommer 95] Sommer E.: FENDER: An Approach to Theory Restructuring, in Lavrač N. & Wrobel S.(eds.), *Machine Learning: ECML-95*, Springer, Berlin, pp.356-359, 1995.

[Srinivasan et al. 92] Srinivasan A., Muggleton S., Bain M.: Distinguishing Exceptions from Noise in Non-Monotonic Learning, *Proceedings of the 2nd International Workshop on Inductive Logic Programming*, 1992.

[Stahl 94] Stahl I.: On the Utility of Predicate Invention in Inductive Logic Programming, in Bergadano F. & Raedt L.de(eds.), *Machine Learning: ECML-94*, Springer, Berlin, pp.272-286, 1994.

[Stahl 93] Stahl I.: Predicate Invention in ILP - an Overview, in Brazdil P.B.(ed.), *Machine Learning: ECML-93*, Springer, Berlin, pp.313-322, 1993.

[Stahl 94] Stahl I.: The Arguments of Newly Invented Predicates in ILP, in *Proceedings of the Fourth International Workshop on Inductive Logic Programming (ILP-94)*, GMD-Studien Nr. 237, pp.233-245, 1994.

[Stahl 95] Stahl I.: The Appropriateness of Predicate Invention as Bias Shift Operation in ILP, to appear in *Machine Learning*, 1995.

[Sutton 92] Sutton R.S.: Adapting Bias by Gradient Descent: An Incremental Version of Delta-Bar-Delta, in *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park, pp.171-176, 1992.

[Sutton 94] Sutton R.S.: Constructive Induction Needs a Methodology based on Continuing Learning, *Panel of the Workshop on Constructive Induction and Change of Representation*, 11th International Conference on Machine Learning (ML-94/COLT-94), New Brunswick, New Jersey, 1994.

[Thagard & Nowak 90] Thagard P., Nowak G.: The Conceptual Structure of the Geological Revolution, in Shrager J., Langley P.(eds.): *Computational Models of Discovery and Theory Formation*, Morgan Kaufmann, San Mateo, CA, 1990.

[Thieme 89] Thieme S.: The Acquisition of Model-Knowledge for a Model-Driven Machine Learning Approach, in Morik K.(ed.), *Knowledge Representation and Organization in Machine Learning*, Vol. 347 of Lecture Notes in Artificial Intelligence, Springer, Berlin, pp.177-191, 1989.

[Weber & Tausend 94] Weber I., Tausend B.: A Three-Tiered Confidence Model for Revising Logical Theories, in *Proceedings of the Fourth International Workshop on Inductive Logic Programming (ILP-94)*, GMD-Studien Nr. 237, pp.391-402, 1994.

[Wirth 88] Wirth R.: Learning by Failure to Prove, in *Proceedings of the Third European Working Session on Learning (EWSL-88)*, pp.237-251, 1988.

[Wirth 89] Wirth R.: Completing Logic Programs by Inverse Resolution, in *Proceedings of the Fourth European Working Session on Learning (EWSL-89)*, Pitman, London, pp.239-250, 1989.

[Wirth & O'Rorke 92] Wirth R., O'Rorke P.: Constraints on Predicate Invention, in Muggleton S.(ed.), *Inductive Logic Programming*, Academic Press, London, U.K., pp.299-318, 1992.

[Wnek & Michalski 94] Wnek J., Michalski R.S.: Hypothesis-Driven Constructive Induction in AQ17-HCI: A Method and Experiments, in Special Issue on Evaluating and Changing Representation, *Machine Learning*, 14(2), 1994.

[Wogulis & Langley 89] Wogulis J., Langley P.: Improving Efficiency by Learning Intermediate Concepts, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, Los Altos, CA, 657-662, 1989.

[Wrobel 89] Wrobel S.: Demand-Driven Concept Formation, in Morik K.(ed.), *Knowledge Representation and Organization in Machine Learning*, Vol. 347 of Lecture Notes in Artificial Intelligence, Springer, Berlin, pp.289-319, 1989.

[Wrobel 93] Wrobel S.: On the Proper Definition of Minimality in Specialization and Theory Revision, in Brazdil P.B.(ed.), *Machine Learning: ECML-93*, Springer, Berlin, pp.65-82, 1993.

[Wrobel 94] Wrobel S.: Concept Formation during Interactive Theory Revision, in Special Issue on Evaluating and Changing Representation, *Machine Learning*, 14(2), 1994.

[Zelle et al. 94] Zelle J.M., Mooney R.J., Konvisser J.B.: Combining Top-down and Bottom-up Techniques in Inductive Logic Programming, in *Proceedings of the Eleventh International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA, pp.343-351, 1994.

[Ziarko 92] Ziarko W.: The Discovery, Analysis, and Representation of Data Dependencies in Databases, in Piatetsky-Shapiro G.,Frawley W.J.(eds.), *Knowledge Discovery in Databases*, AAAI Press, Palo Alto, CA, 1992.

[Zytkow 93] Zytkow J.M.: Introduction: Cognitive Autonomy in Machine Discovery, in Special Issue on Machine Discovery, *Machine Learning*, 12(1-3), 1993.