Bias: Preference, Restriction or Core of Induction

Birgit Tausend
Fakultät Informatik, Universität Stuttgart, Breitwiesenstr. 20-22, D-70565 Stuttgart
tausend@informatik.uni-stuttgart.de

Abstract

In this paper, we aim to give a more precise definition of bias in order to clarify different views. Since this definition involves the problem setting of inductive learning, we have to take into account this definition as well. A more precise definition of the bias can be used to declare, to adapt and to shift the bias of an inductive system. We demonstrate these advantages by declaring and shifting the bias for an Inductive Logic Programming system in MILES-CTL.

1 Introduction

Although the term bias is often used in the context of inductive reasoning, its definition is rather vague since it is used in at least three different meanings. First, the term bias is used to refer to preferences in an inductive system, i.e. why learning results in one particular hypothesis and not in alternatives given a set of examples. In this view, bias is the set of all factors influencing the hypothesis selection [Mitchell 1980, Utgoff 1986]. In the second meaning, bias refers to restrictions imposed on parts of the learning system, e.g. restrictions of the hypothesis language as in [De Raedt 1991]. Thirdly, bias is viewed as the core of induction because it triggers the inductive leap [Mitchell 1977]. Without bias, a hypothesis need not to be a generalization of the input examples. This contradicts the common definition of induction. These different meanings cause a lot of confusion and discussion of what bias is and how it should be defined.

In this paper, we give a more precise definition of bias that clarifies the relation of the three different views. Since the definition of bias is closely related to the definition of the problem setting of inductive learning, we have to consider it as well.

A more precise definition is not only useful to avoid confusion, but can also improve the declaration of bias and the comparison of inductive systems. If the parameters that guide the induction can be identified, inductive systems can be compared with respect to various settings. The results can be used as a basis for selecting indcutive systems for a particular application. In addition, a declarative definition of bias renders the description of inductive learning systems at the knowledge level against the conjecture in [Dietterich 1986].

We focus in this paper on a declarative definition of bias and the control of inductive systems. Controlling a system means to inspect and change the current bias. This is necessary to adapt the system for different learning applications and to recover from overly strong biases.

To illustrate the advantage of controlling a system by declarative bias, we choose an example from the class of Inductive Logic Programming (ILP) systems [Muggleton 1990], a subclass of inductive systems. The problem setting of ILP is to induce logic programs given classified examples and background knowledge. These systems face the problem that they have to strictly control their hypothesis language because Horn logic is a very powerful representation language. Thus, it is an important and demanding task to develop an approach that allows a declarative definition of the hypothesis language.

Our approach, MILES-CTL [Tausend 1994b] covers most biases in ILP. In this paper, we show how it can be used to declaratively define the hypothesis language and to shift the bias without interaction with the user.

This paper is organized as follows. In the next section, we discuss the problem setting and the bias of inductive systems with respect to the different meanings of bias. Then, we give a unifying definition of bias with respect to the problem setting of inductive learning. Section 4 shows an example for the declarative definition of bias. We explain how the bias can be declared by MILES-CTL and show in section 5 how it can be shifted. Then, we discuss the definitions and conclude.

2 Bias and Problem Setting of Inductive Learning

A common understanding of inductive reasoning [Ferber 1994] is that it involves generalization. Given some observations, inductive reasoning leads to a generalization that holds for the observations. This generalization is expected to explain further, unseen observables. Additional knowledge, e.g. knowledge of the domain and common-sense knowledge, may be exploited for inductive inferences as well as a rough idea of the generalization.

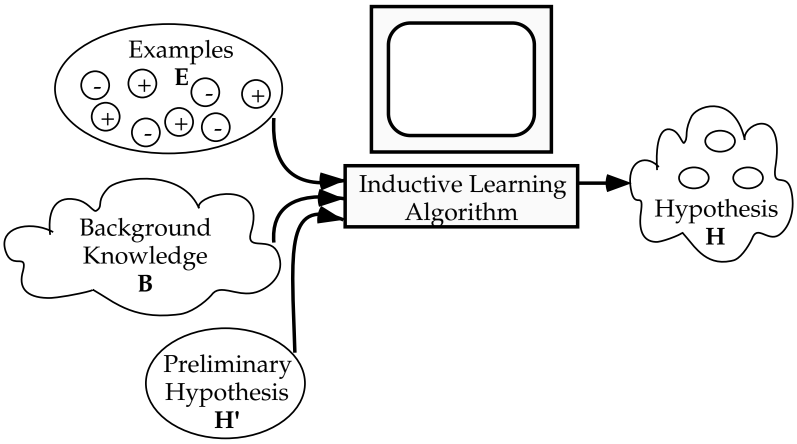

Figure 1: A common definition of inductive learning from examples

In machine learning, the following terminology is used. Observations are called examples and the additional knowledge is referred to as background knowledge. The generalization is called hypothesis in order to emphasize the fact that induction does not result in a logically sound conclusion as, e.g. the deductive inference. This means that it may happen that the generalization does not explain unseen examples. A rough idea of the hypothesis is called a preliminary hypothesis, and it is empty except for systems that learn incremetally or integrated systems the parts of which cooperate to improve a knowledge base.

Definitions of the problem setting of inductive learning attempt to capture this understanding of induction. Among these definitions, the following given in [Michalski 1983] is often used.

Definition 1: The problem setting of inductive learning is as follows.
Given

- the examples E (observational statements),
- a preliminary hypothesis H' (tentative inductive observation),
- the background knowledge B,

find

- a hypothesis H that together with the background knowledge B explains the examples E.

$\diamond$

This definition illustrated in figure 1 subsumes a wide range of machine learning approaches. Comparing different systems shows that they differ in their results, i.e. in which hypothesis they build given a particular set of examples and particular background knowledge. However, even if we assume as in [Michalski 1983] that the background knowledge includes a criterion for selecting the most preferred from a set of alternative hypotheses, this definition fails to explain adequately the input/output behaviour of the system.

In [Utgoff 1986], the differences in the inductive results are ascribed to further factors subsumed by the term bias:

> ...all factors that influence hypothesis selection constitute bias.

Among the influence factors, there are the representation languages of the hypotheses, the examples and the background knowledge. If we describe inductive learning as the search in space of hypotheses orderd by their generality, there are further factors. According to [Utgoff 1986], they include

1. The language in which the hypotheses are described.
2. The space of hypotheses that the program can consider.
3. The procedures that define in what order hypotheses are to be considered.
4. The acceptance criteria that define whether a search procedure may stop with a given hypothesis or should continue searching for a better choice.

In the view of this definition, bias is a set of preferences guiding the induction. A preference for particular hypotheses is expressed by the setting of the factors. For example, if an inductive learning system should result in a logic program, the hypothesis language must contain a set of Horn clauses.

Another weakness of definition 1 is that it conflicts with the common understanding of induction sketched above. According to definition 1, a hypothesis is not necessarily a generalization of the examples. The collection of all positive examples is sufficient to be accepted as hypothesis. Obviously, such a hypothesis lacks the predictive power as it will fail on explaining further positive examples. Given this definition, there is even no reason for an inductive system to make an inductive leap at all.

[Mitchell 1977] was aware of this problem, and argued that generalization is due to the bias of the system.

> In retrospect, it is not surprising that an unbiased generalization system cannot make classifications of instances other than the training instances. An unbiased system is one whose inferences logically follow from the training instances, whereas classifications of new training instances do not logically follow from the classification of the training instances. If totally unbiased generalization systems are incapable of making the inductive leap to characterize new instances, the power of a generalization system follows directly from its biases.

In contrast to [Utgoff 1986], bias is not viewed as something supplementary to the problem setting of inductive learning but merely as its core. This means that there is no induction without bias.

Because of these problems, it is obviuos that defintion 1 is not sufficient and has to be extended to take into account the influence factors and to avoid trivial hypotheses.

A common approach to overcome the latter problem is to require that the hypothesis language is different from the example language. This restriction avoids simple non-generalizing hypotheses, e.g. the set of positive examples.

With the first problem in mind, several problem definitions have been given for inductive learning from classified examples, e.g. [Mitchell 1982], as well as for particular subclasses. For example, the problem setting of Inductive Logic Programming (ILP) [Muggleton 1990] has been defined as follows [Muggleton 1993].

Definition 2: The problem setting of ILP is as follows.
Given

- the examples $E = E^{\oplus} \cup E^{\ominus}$,
- the background knowledge B,

find a logic program P such that

- $B \not\vdash E^{+}$ (Necessity),
- $B \wedge P \vdash E^{+}$ (Sufficiency),

- $B \wedge P \wedge E^{+} \not\vdash \Box$ (Consistency).

$\diamond$

This definition specifies the acceptance criteria by the necessity, the sufficiency and the consisency of the hypothesis P and the hypothesis language, that is the set of logic programs. But it does not take into account other influence parameters, e.g. the the order in which the hypothesis space is searched.

Given a more specific problem setting, bias is often viewed as restriction of the learning task. [De Raedt 1991]

> A second reason why bias is necessary, is that the concept-description language may be very large (...). Without bias or preferences the complete space of descriptions has to searched; this may be impossible or require too much computation. Bias can then be used to prune the search.

Though the later definitions are more adequate, many of them faces the problem that they are still incomplete in explaining the input/output behaviour of an inductive system. They do not reveal the reasons for why a system results in a particular set of hypotheses because important influence factors are disregarded. This often happens when the influence parameters are unclear or hiddden in the system and not identified as influence factors. For example, many operators in ILP include bias but this bias is often disregarded in comparing ILP systems because it is not obvious. Taking this into account, many different biases used in ILP can be identified [Tausend 1994a].

This problem gives the motivation to carefully investigate and compare the influence factors of inductive systems. A new definition should take into account the complete set of factors. But this is not sufficient. Moreover, it is necessary to declare the bias declaratively because this avoids 'hidden' parameter settings.

3 Parameterizing Inductive Learning Systems

The discussion in section 2 shows the definition of the inductive problem setting has to be extended by the factors that influence the hypothesis construction of an inductive system. We integrated them by adding so-called induction parameters. Informally, we can define the problem setting as follows.

> Given a set of examples and the background knowledge, the task of an inductive system is to build the hypothesis with respect to the setting of the induction parameters.

As discussed in section 2, it is necessary to describe the induction parameters in greater detail. A simple approach is to choose the set of influence factors [Utgoff 1986] described in section 2. But these parameters are not very suitable because of several reasons. First, they are incomplete because they do not include important factors, e.g. the representation language of the examples and the background knowledge. Second, some of the factors are derived factors instead of primary factors, i.e. their value depends on the setting of the primary factors. For example, the hypothesis space the program considers varies with respect to other factors, e.g. the hypothesis language and the criterion defined to order the hypotheses. Similarly, the order in which the hypotheses are to be considered depends on the operators used to proceed in the hypothesis space and the parameters of the basic search procedure. For example, if the operator specializes and the basic search procedure is beam search, the hypothesis space is traversed from the general to the specific hypotheses. Given the beam with w and the heuristic selection criterion c, w specialized hypotheses ordered by some criterion c are considered after each application of the operator.

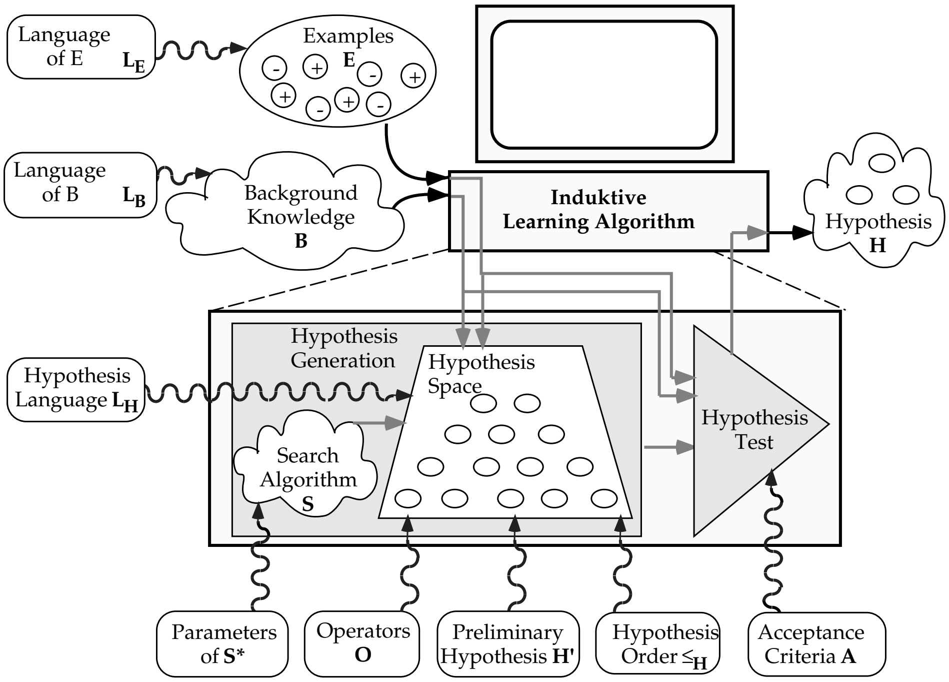

Figure 2: Definition of the problem setting of inductive learning from examples

Taking the induction parameters into account, we define the problem setting of inductive learning from examples as follows.

Definition 3: The problem setting of inductive learning from examples is:
Given

- the examples E in L_E,
- the background knowledge B in L_B,
- and the setting of the induction parameters including
  - the representation language of the examples L_E
  - the representation language of the background knowledge L_B,
  - the representation language of the hypothesis L_H,
  - the parameters S^* of the search procedure S,
  - the operators O to proceed in the hypothesis space,
  - the initial hypotheses H',
  - the criterion $\leq_H$ to order the hypotheses,
  - the acceptance criterion A.

Find

- a hypothesis H in L_H that fulfills the acceptance criterion A.

$\diamond$

By this definition sketched in figure 2, the relation between the three meanings of bias discussed in section 2 can be explained as follows:

- *Core*: The induction parameters are central to the inductive learning task. Whether the system generalizes the input examples depends on their setting. Thus, we view them as the core of induction.
- *Preference*: A particular setting of the induction parameters defines the preference for hypotheses selection.
- *Restriction*: One of the motivations to define a particular setting of the parameters is to restrict the learning system. Other motivations are to use knowledge about useful hypotheses in the application domain, to reduce the complexity of the hypotheses or to deal with noise, for example.

The bias for a system or a class of systems can be declared by setting the induction parameters in definition 3.

Definition 4: The bias of an inductive system (of a class of inductive systems) with induction parameters L_E, L_H, L_B, S^*, O, H', $\leq_H$ and A is defined by the (partial) settings of these parameters. $\diamond$

For example, settings for the class of ILP approaches are that the hypothesis language L_H can be specified as a subset of Horn logic, $\leq_H$ includes the alternatives θ-subsumption, generalized subsumption, relative subsumption and logical implication, L_B is either Horn logic or allows for extensions of predicates and the acceptance criteria include the ones in definition 2.

With respect to the induction parameters, the input/output behaviour of an inductive system can be explained adequately. They also can be used to compare inductive systems. The results of the comparison could serve as criteria for the selection of inductive systems and for supporting users in this selection.

The induction parameters are not only useful for explanation and comparison of inductive systems but also for the control of these systems. This task requires that the bias can be defined declaratively because the parameters should be inspectable. In this case, the induction parameters of a system can be adapted to different applications by varying the parameter setting. While learning, they can be inspected and changed by a user or by the system. This ability is very important as it makes it possible to recover from overly strong biases.

4 Declarative Representation of Bias

To demonstrate the usefulness of the definitions in section 3 and the declarative definition of bias, we focus on the class of ILP systems described in section 2 and 3.

In the recent years, there has been a growing interest in this area. One motivation is that new results from the areas of theorem proving and logic programming can be exploited to further investigate approaches as [Plotkin 1970, Plotkin 1971]. Another motivation is to provide a more powerful representation language in order to overcome the limitations of the attribute-based representation, e.g. as in ID3 [Quinlan 1986]. Choosing Horn logic and its subsets instead of other powerful representation approaches, e.g. KL-ONE [Brachmann & Schmolze 1985, Nebel 1990], is suitable because logic programms can be integrated in many AI applications [Feng 1992, Dolsak & Muggleton 1992] and other programs, e.g. deductive databases [Ceri, Gottlob & Tanca 1990], without changing the representation. Results from logic programming can be integrated as well, e.g. that particular syntactic forms of logic problems are sufficient for many applications.

For this class, several of the induction parameters and the effects of different settings have been investigated, e.g. the criteria to order the hypotheses in the search space [Jung 1993], initial hypotheses [Rouveirol, Adé & De Raedt 1993, Nédellec & Rouveirol 1993], or different acceptance criteria to deal with noisy data.

In contrast, the bias that involves the hypothesis language is not very well investigated. One reason is that there is a great varity of language biases in ILP and they are neither uniquely represented nor their basic constituents are easy to identify. Moreover, many language biases in ILP are not explicitly declared but 'hidden' in the operators, for example.

Our motivation to develop MILES-CTL [Tausend 1994b] was to overcome these problems. In MILES-CTL, the declarative representation of all biases is possible, the declaration of a hypotheses languauge is concise but understandable and it is suitable for large hypothesis languages as well as for hypothesis languages consisting of several subset with different biases.

The basic idea of the representation is an extension of the approach in MOBAL [Morik, Wrobel, Kietz & Emde 1994], i.e., sets of hypotheses are described by schemes.

A scheme for a hypothesis clause called clause template includes schemes for each literal in the clause. Similar asin in a record data type, a literal template consists of several identifiers followed by a scheme variable or a constant. In addition, the domain of a scheme variable in a literal template can be further restricted by conditions. Since the items in a literal template describe the set of covered literal, they have to include at least an item for the predicate name and the arguments. Other items, for example, describe the arity, the number of new variables, the argument or the predicate types or the depth of the covered literals. A declaration of a hypothesis language in MILES-CTL consists of a set of schemes T, the vocabulary Σ including predicates, functors, and types, and an instantiation function I. Given a set T and Σ, I constructs hypotheses by instantiating the scheme variables in a clause template.

For example, suppose that we are to define a hypothesis language L_{H_1} in MILES-CTL. The vocabulary Σ_1 of L_{H_1} includes predicates of type *comp* and *listp* and arity less than 4 and the instantiation funtion is I_1. The following sets of hypothesis clauses has to be in L_{H_1}:

- clauses with exactly one body literal the arity of which is less than 3
- clauses with two body literals where the arity of the first one is less than 2 and the predicate type of the second one is *comp*,
- clauses with two body literals where the arity of the first one is less than 2 and of the second is 3.

These clauses have to be covered by the clause templates in T_1 of L_{H_1}. In MILES-CTL, T_1 can be defined by $T_1 = \{T1, T2, T3\}$ with

$$T1: \begin{bmatrix} predicate : P11, \\ arguments : A11 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P12, \\ arguments : A12, \\ arity : Ar12 || (Ar12 \leq 3) \end{bmatrix}.$$

$$T2: \begin{bmatrix} predicate : P21, \\ arguments : A21 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P22, \\ arguments : A22, \\ arity : Ar22 || (Ar22 \leq 2) \end{bmatrix}, \begin{bmatrix} predicate : P23, \\ arguments : A23, \\ predicate\_type : comp \end{bmatrix}.$$

$$T3: \begin{bmatrix} predicate : P31, \\ arguments : A31 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P32, \\ arguments : A32, \\ arity : Ar32 || (Ar32 \leq 2) \end{bmatrix}, \begin{bmatrix} predicate : P33, \\ arguments : A33, \\ arity : 3 \end{bmatrix}.$$

In these clause templates, $P11, P23, A11, \ldots$ are scheme variables to be instantiated by I.

Obviously, the first set of clauses in the specification of L_{H_1} is covered by $T1$, the second by $T2$ and the third by $T3$. This example shows how a hypothesis language can be defined declaratively in MILES-CTL.

5 Shift of Bias in a Declarative Hypothesis Language

The declarative representation of the hypothesis language by MILES-CTL is also suitable to solve another problem of inductive learning. If the hypothesis language does not include a suitable hypothesis, the inductive learning system fails. In this case, the bias is too strong and has to be shifted towards a more powerful hypothesis language. There are two methods for extending the hypothesis language, predicate invention [Stahl 1993] and relaxing the restrictions imposed by the bias.

In this paper, we focus on the second approach. Since this task is not easy to solve for a user, it should be done by the system.

This task can be divided into three subtasks. The system has to decide

- whether a hypothesis language is too weak for successfully learning,
- whether another hypothesis language is an extension of the current,
- which hypothesis language should be selected if there are several alternatives.

In ILP, CLINT [De Raedt 1991] is one of the few systems that is able to recover from overly strong language biases without help of the user. It uses series of predefined hypothesis languages for shifing the bias. In a series, the bias imposed on the hypothesis language is gradually relaxed from the first to the last language. The first task is easy to solve for CLINT because all languages in the series are finite. Thus, the system shifts to the next language in the series whenever it cannot find a hypothesis in the current hypothesis language. Since the languages in a series are ordered, i.e. the next language includes the previous, the second task is also easy to solve by the system. Solving the third task is not necessary because the next language in a series is the only alternative to shift towards.

This approach is rather simple to use for a system but it requires that the language series are carefully designed and ordered by a user before learning.

Since preordering hypothesis languages is a demanding task for users, we developed another approach for MILES [Stahl & Tausend 1993]. We assume that there are several hypothesis languages $L_{H_1},\ldots,L_{H_n}$ as in CLINT. In contrast, these languages need not to be preordered since MILES-CTL allows to define a criterion $\subseteq_{L_H}$ that solves the second task.

In MILES-CTL, each clause in a hypothesis language has to be covered by a clause template. If we assume that the vocabulary Σ and the instantiation function I do not differ for $L_{H_1},\ldots,L_{H_n}$, a hypothesis language L_{H_j} is an extension of the current language L_{H_i} if L_{H_j} covers at least the clauses covered by L_{H_i}. There are two cases in which L_{H_j} is an extension of L_{H_i}. Fist, the clause templates T_i in L_{H_i} are a subset of the clause templates T_j in L_{H_j}. Second, for each clause template t in L_{H_i} there exists a clause template t' that covers at least the clauses covered by t.

This can be easily tested by the system in the following way. Obviously, t and t' have to include the same number of literal templates. For each literal template l in t, t' has to include a unique literal template that coveres at least the literals covered by the according literal template in t. That means, that all items of l' have to be in l and in l', these itemes have to be equally or less restictively defined as in l, e.g. by using variables instead of constants or by less restrictive conditions.

We illustrate the application of the criterion $\subseteq_{L_H}$ by the following example. Assume that there are two hypothesis languages, L_{H_1} as in section 4 and L_{H_2} defined as follows. In L_{H_2}, the vocabulary and the instantiation function are the same as in L_{H_1}, i.e. Σ_1 and I_1, but the clause templates are defined by $T_2 = \{T4, T5, T6\}$ with

$$T4: \begin{bmatrix} predicate : P41, \\ arguments : A41 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P42, \\ arguments : A42, \\ arity : Ar42 || (Ar42 \leq 3) \end{bmatrix}.$$

$$T5: \begin{bmatrix} predicate : P51, \\ arguments : A51 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P52, \\ arguments : A52, \\ arity : Ar52 || (Ar52 \leq 4) \end{bmatrix}, \begin{bmatrix} predicate : P53, \\ arguments : A53 \end{bmatrix}.$$

$$T6: \begin{bmatrix} predicate : P61, \\ arguments : A61 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P62, \\ arguments : A62, \\ arity : Ar62 || (Ar62 \leq 4) \end{bmatrix}, \begin{bmatrix} predicate : P63, \\ arguments : A63, \\ predicate\_type : comp \end{bmatrix},$$

$$\begin{bmatrix} predicate : P64, \\ arguments : A64, \\ predicate\_type : listp \end{bmatrix}.$$

Obviously, $L_{H_1} \subseteq_{L_H} L_{H_2}$ holds because $T1$ and $T4$ are the same and the clauses covered by $T2$ and $T3$ are covered by $T5$. The clause templates $T5$ and $T6$ may cover further clauses not covered by the clase templates of L_{H_1}.

The first task is solved in MILES-CTL similarly to the approach in MOBAL [Morik, Wrobel, Kietz & Emde 1994] by searching with respect to a pruning criterion. To solve the third task, i.e. selecting a hypothesis language among several alternatives, we defined a model that allows to combine several heuristics in a vector

$$[heuristics_1, heuristics_2, \ldots, heuristics_n].$$

These heuristics are applied subsequently until a single alternative is found. However, this solution is subject to further investigations, e.g. experiments with further heuristics and combinations.

6 Conclusion

Bias is an old but very important problem in inductive learning that many people discuss and write about. Despite this importance and interest, bias is rather vaguely defined. In this paper, we gave a more precise definition since there is a lot of confusion concerning the term bias. The definitions are not revolutionary compared to former definitions but they suitable to clarify the relation of the different views. Another advantage is that they can be used to declaratively define the bias of a system and to recover from overly strong biases. We demonstrated these advantages by examples in our approach MILES-CTL, a declarative representation for the language bias in ILP.

The variety of induction parameters in our definition of the problem setting of inductive learning might be viewed as a problem in declaring and adapting the bias for a system. However, in practice, most parameters remain fixed if a system is adapted and there are a lot of well investigated settings for most of the parameters, e.g. acceptance criteria or criteria to order the hypotheses in the search space.

Acknowledgements

This work has been partially supported by ESPRIT BRA 6020 ILP. I would like to thank Irene Stahl and Katharina Morik for many discussions concerning this work.

References

Brachmann, R. J. & Schmolze, J. G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, *9*, 171–216.

Ceri, S., Gottlob, G., & Tanca, L. (1990). *Logic Programming and Databases.* Springer.

De Raedt, L. (1991). *Interactive Concept-Learning.* PhD thesis, Katholieke Universiteit Leuven.

Dietterich, T. (1986). Learning at the knowledge level. *Machine Learning*, *1*(3), 287–316.

Dolsak, B. & Muggleton, S. (1992). The application of inductive logic programming to finite element mesh design. In S. Muggleton (Ed.), *Inductive Logic Programming.* Academic Press.

Feng, C. (1992). Inducing temporal fault diagnosis rules from a qulitative model. In S. Muggleton (Ed.), *Inductive Logic Programming.* Academic Press.

Ferber, R. (1994). *Philosophische Grundbegriffe: eine Einführung.* Beck'sche Reihe, Nr. 1054. Beck.

Jung, B. (1993). On inverting generality relations. In *Proc. of 3rd International Workshop on Inductive Logic Programming ILP-93*, Technical Report, IJS-DP-6707. J. Stefan Institute.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach.* Tioga.

Mitchell, T. (1977). Version spaces: A candidate elimination approach to rule learning. In *Proc. of 5th International Joint Conference on AI.* Morgan Kaufmann.

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence*, *18*(2), 203–226.

Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical Report TR-117, Department of Computer Science, Rutgers University.

Morik, K., Wrobel, S., Kietz, J., & Emde, W. (1994). *Knowledge Aquisition and Machine Learning: Theory Methods and Applications.* Academic Press.

Muggleton, S. (1990). Inductive logic programming. In *Proc. of 1st Conference on Algorithmic Learning Theory*, Tokio. Ohmsha.

Muggleton, S. (1993). Inductive logic programming: Derivations, successes and shortcoming. In *Machine Learning: ECML-93, European Conference on Machine Learning, Wien, Austria.* Springer.

Nebel, B. (1990). *Reasoning and Revision in Hybrid Representaion Systems.* Springer.

Nédellec, C. & Rouveirol, C. (1993). Biases for incremental hypothesis-driven systems. In A., C. (Ed.), *The AAAI Spring Symp. on Training Issues in Incremental Learning.* AAAI Press.

Plotkin, G. (1970). A note on inductive generalisation. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence*, volume 5 (pp. 153–163). Edinburgh: Edinburgh University Press.

Plotkin, G. (1971). A further note on inductive generalisation. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence*, volume 6 (pp. 101–124). Edinburgh: Edinburgh Univeristy Press.

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

Rouveirol, C., Adé, H., & De Raedt, L. (1993). Bottom-up generalisation in ilp. In *Proc. of IJCAI-93 Workshop on Inductive Logic Programming.* Morgan Kaufmann.

Stahl, I. (1993). Predicate invention in ILP – an overview. In *Machine Learning: ECML-93, European Conference on Machine Learning, Wien, Austria.* Springer.

Stahl, I. & Tausend, B. (1993). MILES – a Modular Inductive Logic Programming Experimentation System. Deliverable STU1.2 of ESPRIT BRA 6020: Inductive Logic Programming (ILP).

Tausend, B. (1994a). Biases and their effects in inductive logic programming. In *Machine Learning: ECML-94, European Conference on Machine Learning, Catania, Italy.* Springer.

Tausend, B. (1994b). Representing biases for inductive logic programming. In *Machine Learning: ECML-94, European Conference on Machine Learning, Catania, Italy.* Springer.

Utgoff, P. (1986). Shift of bias for inductive concept learning. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Morgan Kaufmann.