Biases and their Effects in Inductive Logic Programming

Birgit Tausend
Fakultät Informatik
Universität Stuttgart
Breitwiesenstr. 20-22
D-70565 Stuttgart
Germany

Abstract

In inductive learning, the shift of the representation language of the hypotheses from attribute-value languages to Horn clause logic used by Inductive Logic Programming systems accounts for a very complex hypothesis space. In order to reduce this complexity, most of these systems use biases. In this paper, we study the influence of these biases on the size of the hypothesis space. For this comparison, we first identify the basic constituents the biases are combined of. Then, we discuss the restrictions set on the distribution of terms in the clause by the constituents of the bias. The effects of several constituents and of some combinations are shown by seven experiments.

1 Introduction

Inductive logic programming (ILP) [Mug90] aims to overcome some limitations of attribute-value based learning algorithms by using Horn clause logic as the representation language for the hypotheses. However, a powerful representation language like Horn clause logic accounts for a a very complex hypothesis space. This complexity may prevent the systems from finding an appropriate hypothesis or from finding it in a reasonable time. As a consequence, there is a growing interest in appropriate biases, in particular in language biases, i.e. biases concerning the representation language of the hypotheses [Utg86].

In the last years, a plenty of biases for controlling the hypothesis space of ILP systems has been developed. Although they are often combinations of several similar constituents, their influence on the size of the hypothesis space varies. For example, comparing the language restrictions of CLINT [DR91] and of GOLEM [MF90] as described in [AB92] shows that adding the determinism as a further restriction strongly reduces the hypothesis space.

In this paper, we aim to study the biases in ILP systems with respect to the restrictions they set of the hypothesis space.

The first task in this comparison is to identify the underlying basic constituents each bias is combined of. These constituents involve very different features of a

Horn clause. Syntactic features, as e.g. the number of variables or literals, can be restricted as well as semantic features, e.g. determinate terms or the types of arguments in a clause.

The second issue is to investigate how the constituents restrict the set of hypothesis clauses. The measure introduced by [PK92] is the number of variabilizations. As the variabilization does not take structured terms into account, we extend this term to the term distribution, i.e. the distribution of the terms to argument positions in a clause.

This paper is organized as follows. The next section gives a description of the learning problem of ILP with respect to the bias. Then, basic constituents of ILP biases are identified and described in section 2. In section 4, the effects of the constituents of bias imposed on the hypothesis space are discussed. These effects are evaluated in section 5, an the last section concludes with some ideas for further work.

2 Inductive Logic Programming and Bias

The inductive logic programming problem can be formalized as described in [Mug93]. Given the positive examples E^+, the negative examples E^- and the background knowledge B, the task is to find a logic program H, and H is a set of definite clauses $C : l_0 \leftarrow l_{1, \ldots,} l_n$. This hypothesis H has to be necessary, sufficient, and consistent. The necessity means that none of the positive examples can be explained by the background knowledge, and sufficiency means that the program P should explain all positive examples together with the background knowledge [Mug93]. The consistency criterion requires the hypothesis H to be consistent with the background knowledge and the negative examples. This restriction can be weakened in the presence of noise, i.e. the weak consistency criterion means that the hypothesis H has to be consistent with the background knowledge.

In addition to the formalization in [Mug93], the representation languages of the examples, the background knowledge, and the hypotheses can be taken into account. Consequently, there is an additional constraint for the hypothesis H that requires H to be in the language of the hypotheses L_H.

The problem setting of ILP can be described by:

Given • the examples $E = E^{\oplus} \cup E^{\oplus}$ in L_E,
• the background knowledge B in L_B,
• the hypothesis language L_H,

find a logic program P in L_H such that
• $B \not\vdash E^+$ (Necessity),
• $B \wedge P \vdash E^+$ (Sufficiency),
• $P \subseteq L_H$.

The learning success strongly depends on the hypothesis language L_H. If L_H is very restricted, the learning is very efficient as there are less hypothesis. However, if L_H is overly restricted, the hypothesis cannot be expressed in L_H, and the learning fails. In contrast, a weakly restricted hypothesis language is often very inefficient as the the hypothesis H cannot be found in a reasonable time.

The goal of using Horn clause logic instead of an attribute-value based representation language was to overcome the limitation of an overly restricted representation language. However, using unrestricted Horn clause logic turned out to result in very inefficient learning systems. To increase the efficiency, the ILP systems use different biases. Each of these biases aims to restrict the representation language L_H to a subset of the Horn clause logic that is efficient but large enough to include the hypothesis.

3 Basic Constituents of Biases

In order to make the language biases of ILP systems comparable and to make clear which syntactic or semantic feature of the clause they involve, we identify the basic constituents these biases are combined of. For this purpose, we recall or give a unifying definition for each constituent.

In the following definitions, $Vars(l_i)$ ($Terms(l_i)$) denotes the variables (terms) in a literal l_i of a clause $C : l_0 \leftarrow l_1, \ldots, l_n$. Similarly, $Vars(C)$ ($Terms(C)$) denotes all variables (terms) in C. A variable X is said to be shared by the literals l_i and l_j if $X \in (Vars(l_i) \cap Vars(l_j))$.

3.1 Syntactic Constituents of Biases

The syntactic constituents of bias involve syntactic features of the hypothesis clauses.

An important syntactic restriction is the restriction to function-free Horn clause logic. A clause of this subset does not include functors. Similarly, the limitation to constant-free Horn clause logic means that the hypothesis language does not include constants. Using a function-free language for the representation of the hypothesis is a widely used restriction, e.g. in FOIL [Qui90], CLINT [DR91] or MOBAL [MWKE93]. Other systems transform the clauses to a equivalent function-free clauses by flattening, as e.g.in ITOU [Rou92].

Both, the range restriction and the restriction to constrained clauses, are syntactic restrictions addressing the relation of the variables of the head literal to those of the body literals.

The first one requires that all variables of the head literal must occur in the body of the clause. This relation is called the range restriction in CLINT [DR91] and the restriction to generative clauses in GOLEM [MF90].

Definition 3.1: A clause $C : l_0 \leftarrow l_1, \ldots, l_n$ is generative or range restricted if $Vars(l_0) \subseteq (Vars(l_1) \cup \ldots \cup Vars(l_n))$.

The second restriction concerning this relation, the restriction to constrained clauses, requires that the body literals only contain variables of the head literal.

Definition 3.2: A clause $C : l_0 \leftarrow l_1, \ldots, l_n$ is constrained if $Vars(l_0) \supseteq (Vars(l_1) \cup \ldots \cup Vars(l_n))$.

The restriction to constrained clauses is a very strong bias that is is useful when the hypothesis can be expressed in the DATALOG language.

In contrast, the restriction to linked Horn clauses is part of almost all biases in ILP systems, e.g. ITOU [Rou92], CLINT [DR91], GOLEM [MF90] or SIERES [WO91]. This restriction aims to disable reducible literals in a clause. A linked Horn clause contains only linked literals, i.e. literals that share at least one variable with another linked literal or the head literal. The following definition subsumes the definitions of linked clauses in CLINT [DR91] and of connected clauses in ITOU [Rou92].

Definition 3.3: A clause $C : l_0 \leftarrow l_{1,} \ldots, l_n$ is *linked* if all of its literals are linked.

$$\text{A literal } l_i \text{ is linked } \begin{cases} \text{if it is the head literal of } C\text{,i.e. } i = 0 \\ \text{if it contains a variable } X \\ \quad \text{where } X \in Vars(l_j) \text{ and } l_j \text{ is a linked literal.} \end{cases}$$

Example 1: Let $C_1 : predecessor(A, C) \leftarrow parent(A, B), predecessor(B, C)$ and $C_2 : father(A, C) \leftarrow parent(A, C), male(D)$ be clauses. C_1 is a linked clause because all of the literals are linked whereas C_2 is not linked because of the literal $male(D)$. $\diamond$

Another kind of syntactic restrictions concerns the sharing of variables by the literals in the clause.

With respect to the sharing of the variables in the clause, a connection path of a variable from the head literal to a literal containing this variable can be described. This path includes all literals involved by shared variables. As a variables may occur in different literals, a variable may have several connection paths.

In particular, approaches representing the bias by graphs, e.g. SIERES [WO91] or the approach described in [BJ93], use the edges of the graph to specify the literals allowed to share variables and disable the sharing of literals by omitting the edges between these literals.

The length of shortest connection path of a variable results in the depth of a variable. The depth of a literal is the maximum of the depth of its variables. This syntactic feature of the literals in the clause affected, e.g. by parameter i of the ij-determinism in GOLEM [MF90]. In CLINT [DR91], this restriction is also used.

For the definition of the connection paths and the depth, we adapt the definition of the connection relation chains in [KW91].

Definition 3.4: Let $C : l_0 \leftarrow l_{1,} \ldots, l_n$ be a clause. A connection path of a variable X in $Vars(l_i)$ is

$$cp(X) = \begin{cases} \emptyset & \text{if } i = 0 \\ cp(Y) \circ l_i & \text{if } \exists Y \in l_i, Y \neq X, \text{ and } cp(Y) \end{cases}$$

The depth $depth(X)$ of a variable X is defined by

$depth(X) = min\{length(cp(X)) | cp(X) \text{ is a connection path of } X\}$.

The depth of a clause C (a literal l_i) is the maximal depth of its variables, i.e.

$depth(C) = max\{depth(X) | X \in Vars(C)\}$.

Example 2: In a clause $C_3 : p1(A,C) \leftarrow p2(A,B), p3(A,C,D), p4(B,D)$, for example, $cp(A) = \emptyset$,$cp(C) = \emptyset$ and $depth(C) = 0$, $depth(A) = 0$. The variable D has several connection paths, e.g.

$cp(D) = cp(A) \circ p3(A,C,D) = p3(A,D)$, or

$cp(D) = cp(C) \circ p3(A,C,D) \circ cp(C) = p3(A,C,D)$, or

$cp(D) = cp(B) \circ p4(B,D) = p2(A,B) \circ p4(B,D)$,

and its depth is $depth(D) = 1$.

$\diamond$

The connection by shared variables can be used to impose a order on the literals in a Horn clause. A Horn clause can be ordered by ordering its literals according to their depth. A literal l_i of which the maximal depth is smaller than the depth of another literal l_j should preceed this literal l_j in the ordered clause.

Definition 3.5: A clause $C : l_0 \leftarrow l_1, \ldots, l_n$ is ordered by literal depth if $i \leq j$ for each two literals l_i and l_j iff $depth(l_i) \leq depth(l_j)$.

Example 3: The clause $C_{4'} : p1(A,C) \leftarrow p4(B,D), p2(A,B), p3(A,C,D)$ is not ordered by literal depth because $depth(p4(B,D)) = 1 < 0 = depth(p2(A,B)) = depth(p3(A,C,D))$. However, the equivalent clauses

$C_4 : p1(A,C) \leftarrow p2(A,B), p3(A,C,D), p4(B,D)$

in example 2 and

$C_{4''} : p1(A,C) \leftarrow p3(A,C,D), p2(A,B), p4(B,D)$

are ordered by literal depth because $depth(A) = depth(C) = 0$ and $depth(B) = depth(D) = 1$ $\diamond$

As a variable may have several connection paths of the same length and a clause may include literals of the same depth, a clause can be ordered by literal depth in several ways. All these clauses are equivalent because the ordering the literals by their depth does not change the success set of the clause.

In clause ordered by literal depth, a literal contains a new variable if this variables does not occur in the preceeding literals, e.g. the variable B in $p2(A,B)$ of the clause C_4 in example 3 is new. This restriction is often combined with the restriction of the depth, e.g. CLINT [DR91].

Definition 3.6: A variable X is new in a literal of a Horn clause $C : l_0 \leftarrow l_1, \ldots, l_n$ that is ordered by literal depth if it does not occur in the preceeding literals $l_0, \ldots, l_{i-1}$.

The restriction to unique variables means that the clause is not allowed to contain a literal with the same variable at different argument positions e.g. a literal $p1(X,X,Y)$ is excluded by this restriction. This restriction is applied for binary predicates in FOCL [PK92].

Definition 3.7: A literal l_i in a clause $C : l_0 \leftarrow l_1, \ldots, l_n$ has unique variables if $X_i \neq X_j$ for every two variables $X_i, X_j \in \{X_1, \ldots, X_n\}$ of a literal $l_k = p(X_1, \ldots, X_n)$. A clause is restricted to unique variables if it includes only literals with unique variables.

3.2 Semantic Constituents of Biases

Besides the syntactic constituents of bias described in section 3.1, there are several bias constituents involving semantic properties of hypothesis clauses, e.g. the mode and the type of a literal.

A mode declaration describes the variables of a predicate in terms of input and output variables. In addition, a mode declaration may contain information about the number of solutions existing for the predicate with this mode declaration as proposed in [Mug93]. As mode declarations are often available in the background knowledge, this restriction is used by many systems, e.g. in INDICO [STW93], GOLEM [MF90], or as an optional bias in ITOU [Rou92] or in the recent version of FOIL [QCJ93].

Definition 3.8: A mode declaration of predicate $p(X_1, \ldots, X_n)$ is denoted by
$mode(p(X_1, \ldots, X_n), [md(X_1), \ldots, md(X_n)], sol)$,
where $md(X_i) \in \{+, -\}$ and

$$sol = \begin{cases} * & \text{if } p(X_1, \ldots, X_{vn}) \text{ has} \quad \text{finitely many solutions} \\ 1 & \text{"} \quad \text{exactly 1 solution} \\ 2 & \text{"} \quad \text{exactly 2 solutions} \\ \ldots & \end{cases}$$

A variable $X_i \in Vars(p)$ is an input variable if $md(X_i) = +$, a variable with $md(X_i) = -$ is an output variable.

Example 4: With respect to the mode declaration
$mode(append(X, Y, Z), [+, +, -], 1)$,
the variables X and Y are input variables and Z is the output variable of the predicate $append(X, Y, Z)$. There is one solution of the predicate $append(X, Y, Z)$ with respect to this declaration because the third argument of the declaration is one. $\diamond$

Type declarations can be given for both predicates and arguments of predicates. Argument types may either define the syntactic structure of an argument, as in INDICO [STW93], or they can be semantic types. Restrictions by the type of predicates are often a bias constituent in schemata. For example, FOCL [PK92] uses relational clichés, a particular kind of schemata with typed predicate variables. Another approach is to define sets of predicates that can be used as body literals with respect to the head literals. This approach is realized by the topology of predicates in MOBAL [MWKE93].

Definition 3.9: A type declaration of a predicate $p(X_1, \ldots, X_n)$ is denoted by
$predicate\_type(p(X_1, \ldots, X_n), PType)$
where $PType$ is the type of $p(X_1, \ldots, X_n)$.
A declaration of the types of arguments of $p(X_1, \ldots, X_n)$ is denoted by
$argument\_types(p(X_1, \ldots, X_n), [type_{i_1}(X_1), \ldots, type_{i_n}(X_n)])$,
where $type_{i_1}, \ldots, type_{i_1}$ are argument types.

Example 5: According to $predicate\_type(append(X, Y, Z), list\_pred)$ the predicate $append(X, Y, Z)$ has the predicate type $list\_pred$.

The types of the arguments of $append(X, Y, Z)$ can be defined by $argument\_types(append(X, Y, Z), [type1(X), type1(Y), type1(Z)])$ with respect to a type description of lists defined recursively by the clauses

$type1([]).$

$type1([A|B]) \leftarrow atom(A), type1(B).$

The semantic types *list* and *sorted_list* may be used in a type declaration $argument\_types(merge(X, Y), [list(X), sorted\_list(Y)])$. $\diamond$

Two other related constituents of bias are the determinism and the functionality. A clause is determinate if there is only one solution for each of the body literals if the head literal is instantiated by an example. This restriction is part e.g. of the ij-determinism in GOLEM [MF90].

Definition 3.10: [MDR93] A definite clause $C : l_0 \leftarrow l_1, \ldots, l_n$ is determinate (with respect to the background knowledge B and the examples E) if and only if for every substitution θ for l_0 that unifies to a ground instance $e \in E$, and for all $i = 1, \ldots, n$ there is a unique substitution θ_i such that $(l_1 \wedge \ldots \wedge l_n)\theta\theta_i$ is both ground and true in $\mathcal{M}^+(B)$.

Example 6:

Let $C_4 : list\_pred([H|T], L1, [H|L]) \leftarrow test1(H, L1), list\_pred(T, L1)$. C_4 is determinate with respect to the examples E and the background knowledge B where

$$E = \{list\_pred([1,2],[3,5],[1,2,3,5]),\ list\_pred([2],[3,5],[2,3,5]),\ list\_pred([],[3,5],[3,5])\}$$

$$B = \{test1(1,[3,5]), test1(2,[3,5]),\ test2([3,5],3), test2([3,5],5)\}$$

In contrast, the clause

$C_5 : list\_pred([H|T], L1, [H|L]) \leftarrow test2(L1, X), list\_pred(T, L1).$

is not determinate with respect to E and B because of the literal $test2(L1, X)$.$\diamond$

In contrast to the determinism, the functionality used e.g. in ITOU [Rou92] as an optional bias constituent, does not require that that the body literals are also determinate.

Definition 3.11: A definite clause $C : l_0 \leftarrow l_1, \ldots, l_n$ is functional if for every ground substitution θ of a particular subset of the arguments of l_0 there is a unique substitution σ of the remaining arguments.

For example, both clauses of the example 6 are functional with respect to the examples as the body literals are not required to be functional. However, the predicate $test2/2$ is not functional.

Another semantic constituent of bias introduced in the FOCL system [PK92] aims to eliminate commutative literals. For example, a clause including two literals $equal(X, Y)$ and $equal(Y, X)$ is disabled by this restriction if the predicate *equal* is commutative.

Definition 3.12: A predicate $p(X_1, \ldots, X_n)$ is commutative according to a permutation $perm(X_1, \ldots, X_n) = (X_{p_1}, \ldots, X_{p_n}$ iff $p(X_1, \ldots, X_n)$ is equivalent to $p(X_{p_1}, \ldots, X_{p_n})$.

The commutativity of a predicate is denoted by

$commutative(p(X_1, \ldots, X_n), [X_{1_p}, \ldots, X_{n_p}])$

where $[(X_{1_p}, \ldots, X_{n_p}]$ is a permutation of the arguments $X_1, \ldots, X_n$.

4 Effects of Biases Constituents

In [PK92] the variabilization was introduced to measure the effect of several biases. A variabilization of a clause describes the distribution of the variables or terms to the argument positions of the literals in a clause.

Example 7: The following clauses are eight of the variabilizations of a constrained clause with the head literal $p1/2$ and the body literals $p2/2$, $p3/1$:

$$
\begin{array}{ll}
p1(X,Y) \leftarrow p2(X,Y), p3(X). & p1(X,Y) \leftarrow p2(X,Y), p3(Y). \\
p1(X,Y) \leftarrow p2(Y,X), p3(X). & p1(X,Y) \leftarrow p2(Y,X), p3(Y). \\
p1(X,Y) \leftarrow p2(X,X), p3(X). & p1(X,Y) \leftarrow p2(X,X), p3(Y). \\
p1(X,Y) \leftarrow p2(Y,Y), p3(X). & p1(X,Y) \leftarrow p2(Y,Y), p3(Y).
\end{array}
$$

There are eight further clauses where the body is the same and the head literal is $p1(Y,X)$. $\diamond$

As the definition of the variabilization is only useful for function-free languages, we have to extend by introducing the term distribution in order to take structured terms and their subterms into account.

Definition 4.1: Let $C = L_0 \leftarrow L_2 \vee, \ldots, \vee L_n, n \geq 0$ be a clause.

An argument position in C is a pair $\langle p_1, p_2 \rangle$ where p_1 denotes the position of L_i in C and $0 \leq i \leq n$, and p_2 with $0 < p_2 \leq Arity(L_i)$ is the position of the argument in the literal L_i.

A term distribution $\mathcal{T}(\mathcal{P}(C), T)$ is a function from the set of argument positions $\mathcal{P}(C)$ in a clause C to a set of terms T where $\mathcal{T}(\mathcal{P}(C), T)$ assigns exactly one term to each position in $\mathcal{P}(C)$.

Example 8: Let $C : p1(X,Y) \leftarrow p2(Y,X), p3(X)$ be a clause.
Then, $\mathcal{P}(C) = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle, \langle 3,1\rangle\}$ and $\mathcal{T}(\mathcal{P}(C), \{X,Y\}) = \{\langle 1,1\rangle/X, \langle 1,2\rangle/Y, \langle 2,1\rangle/Y, \ldots\}$. $\diamond$

The number of term distributions $|\mathcal{T}(\mathcal{P}(C), T)|$ depends on three factors, the number of terms to be distributed, the number of argument positions in the clause and restrictions on the mapping of terms to argument positions. Applying a particular bias constituent means changing these parameters, i.e. reducing the number of terms or argument position or restricting the mapping.

4.1 Reducing the Number of Terms

Reducing the number of terms, i.e. limiting T in $\mathcal{T}(\mathcal{P}(C), T)$ strongly decreases the size of L_H. In [PK92], it is shown that the number of hypotheses represented in a function-free language increases exponentially by the number of variables.

Example 9: Allowing one new variable, e.g. Z, in one of the body literal in the clauses in example 7 yields 15 additional hypotheses:

$$
\begin{array}{ll}
p1(X,Y) \leftarrow p2(Z,X), p3(X), & p1(X,Y) \leftarrow p2(Z,X), p3(Z), \\
p1(X,Y) \leftarrow p2(Z,Y), p3(X). & p1(X,Y) \leftarrow p2(Z,Y), p3(Z). \\
p1(X,Y) \leftarrow p2(X,Z), p3(X). & p1(X,Y) \leftarrow p2(X,Z), p3(Z). \\
p1(X,Y) \leftarrow p2(Y,Z), p3(X). & p1(X,Y) \leftarrow p2(Y,Z), p3(Z). \\
p1(X,Y) \leftarrow p2(X,Y), p3(Z). & p1(X,Y) \leftarrow p2(Y,X), p3(Z). \\
p1(X,Y) \leftarrow p2(X,X), p3(Z). & p1(X,Y) \leftarrow p2(Y,Y), p3(Z). \\
p1(X,Y) \leftarrow p2(Z,Z), p3(X). & p1(X,Y) \leftarrow p2(Z,Z), p3(Z). \\
p1(X,Y) \leftarrow p2(Z,Z), p3(Y). &
\end{array}
$$

$\diamond$

Obviously, simply limiting the number of terms is not appropriate as shown by example 9. Enumerating all term distributions in a clause yields a lot of useless hypotheses, i.e. clauses containing unlinked and reducable literals, e.g. the literal $p3(Z)$ in the clause $p1(X,Y) \leftarrow p2(X,Y), p3(Z)$ of example 9 .

As a consequence, the introduction of new variables described in definition 3.6 has to be controlled more carefully as they cause unliked literals.

An extreme restriction of the new variables is the restriction to constrained clauses as in definition 3.2. As this bias does not allow new variables in the body literals it is often ttoo strong and only suitable for learning a small subset of Horn clauses.

Combining the restriciton of new variables with other bias constituents leads to more useful biases. For example, the languages of the series in CLINT [DR91] limit the number of new variables depending on the depth of the literal introducing the new variable. Similarly, the ij-determinism used e.g. in GOLEM [MF90] limits the number of new terms with new variables at depth i to j. In addition, the ij-determinism requires the terms including new variables to be determinate.

Another approach to reduce the number of terms to be distributed in a clause is the restriction to function-free clauses. In contrast to hypothesis languages with functors where all terms together with their subterms can be used in the distribution, only a new variable or the variables at preceeding argument positions can be put to an argument position of a new literal in a function-free clause. The restriction to a function-free hypothesis language is mostly combined with other bias constituents, e.g. with the range restriction and limitations of the new variables in CLINT [DR91] or with restrictions to the connection paths and predicate types in MOBAL [MWKE93].

4.2 Reducing the Number of Argument Positions

The second parameter in $\mathcal{T}(\mathcal{P}(C), T)$ affected by the biases in order to restrict the hypothesis language is the number of argument positions available in a clause $|\mathcal{P}(C)|$. This parameter depends on both the number of literals in C and the arity of these literals.

Restricting the number of literals can be done either explicitly by defining a maximum length of the clauses or by providing a set of schemata, e.g. rule models or graphs. As scheme-based systems only use a finite set of schemata, the number of literals in a hypothesis is limited to the maximim number of literals in a scheme.

The number of argument positions depends not only on the number of literals but also on which literals may occur in the clause. This restriction can be achieved by e.g disabling a commutative to be used twice in a clause [PK92]. The type of a predicate, a further constituent of bias in definition 3.9, can be used to limit the literals in a schema to those matching only predicates of a particular type [Tau93].

Similarly, the number of argument positions in a clause can be restricted by limiting the aritiy of literals in a schema [Tau93].

4.3 Restricting the Mappings of Terms to Positions

Apart from decreasing the number of argument positions and terms available for the term distribution in a clause, setting restrictions on the mapping of terms to argument positions is a further way of reducing the number of hypotheses.

As shown above, restricting the new terms is one way to prevent from unlinked literals as in definition 3.3. However, the restriction of new variables has to be combined with the restriction to linked clauses in order to avoid unliked literals, as e.g. $p2(Z,Z)$ in the clause $p1(X,Y) \leftarrow p2(Z,Z), p3(X)$ in example 9. Applying this restriction to the clauses in example 9 excludes the last seven clauses.

Another way of restricting the mapping of terms to argument positions is to use argument types. As this restriction means that the type of a term has to agree with the type specified for the argument position by an argument type declaration, only term distributions that do not cause type conflicts are enabled by this restriction.

A similar approach is to provide mode declarations as in definition 3.8. Mode declarations restrict the mappings in $\mathcal{T}(\mathcal{P}(C), T)$ because usually a variable or term occurring at an argument position defined as output must not occur at an argument position declared as output of any other literal but has to be used for the input positions of other literals.

The restriction to unique variables in a clause in definition 3.7 also restricts the mapping because it allows only variables at a particular argument position that do not appear on other positions in the same literal.

The range restriction in definition 3.1 also restricts the mapping of terms to argument because it requires all head variables to occur in the body literals.

A boundery case of restricting the mapping is realized by the rule models [MWKE93] as the term distribution of these schemata is fixed. This approach is weakened by the graph-based representations, as described in [WO91] or [Tau93], that specify specify particular connection paths as in definition 3.4 by their edges. This restriction limits the mapping of terms to argument positions to mappings that agree with the connection paths specified.

The effects of the constituents of biases in ILP are summarized in the table 4.3.

Restriction	$\mid T \mid$	$\mid \mathcal{P}(C) \mid$	$\mid \mathcal{T}(\mathcal{P}(C), T))\mid$
to function-free clauses	x		
to constant-free clauses	x		
to a limited number of variables	x		
to limited number of literals		x	
of new variables	x		
to unique variables			x
to linked clauses			x
to particular connection path			x
of the literal depth		x	
to generative clauses			x
to determinate terms	x		
to functionality	x		
by predicate types		x	
by arguments types			x
of commutative literals			x

Table 1: Effects of Bias Constituents

5 Empirical Evaluation of the Bias Constituents

The task for the follwoing experiments is to learn the definition of a predicate $pred1/3$. The background knowledge contains the predicates $pred2/2$, $pred3/2$, and $pred4/2$. The type declarations for the predicates are as follows:

$predicate\_type(pred1(A, B, C), list)$,
$predicate\_type(pred2(A, B), comp)$,
$predicate\_type(pred3(A, B), comp)$,
$predicate\_type(pred4(A, B), list)\}$

The declarations of the argument types of the predicates are given by the set $Argument\_Types =$

$\{argument\_types(p1(A, B, C), [type1(A), type1(B), type1(C)])$,
$argument\_types(p2(A, B), [number(A), number(B)])$,
$argument\_types(p3(A, B), [number(A), number(B)])$,
$argument\_types(p4(A, B), [number(A), type1(B)])\}$

In all experiments, the hypothesis clauses have to be linked and to contain three literals.

We use two different restrictions on the connection paths. The first, $CP1$ requires that both literals include of least one term of the terms in the head literals. The second, $CP2$, additionally requires that the second body literal contains at least on term of the first body literal.

In experiment $Exp1$, the hypothesis language allows functors and the hypotheses are restricted to respect the connection path defined in $CP1$. In contrast, the hypothesis language in experiment $Exp2$ has to be function-free.

In experiment $Exp3$, no new variables may occur in the body literals, i.e. the clause is constrained. The experiment shows that $CP2$ is not appropriate in combination with constrained hypotheses as according to this restriction, the second body literal has to include at least one new variable. However, because of the restriction to constrained hypotheses, there is no new variable.

The setting in experiment $Exp4$ is the same except that the connection paths are restricted by $CP1$ instead of $CP2$.

The restriction $AT1$ in experiment $Exp5$ causes the types of the arguments to be checked, i.e. the type of a term assigned to an argument position has to agree with the type of this position specified in $Argument\_Types$.

In experiment $Exp6$, the body literals are restricted by $PT1$ with respect to their predicate types. $PT1$ requires the first literal in the body to be of predicate type *comp* whereas the second has to be of type $list$.

The last experiment $Exp7$ combines the constituents such that a restriction similar to a rule model $R1 : P(X,Y,Z) : -Q1(X,Y), Q2(X,X2,Z)$ is produced. For this purpose, the arguments are fixed by the restriction $Args1$ where the arguments correspond to that of the rule model $R1$ and the connection paths to $CT2$.

	Experiments						
Bias Constituents	$Exp1$	$Exp2$	$Exp3$	$Exp4$	$Exp5$	$Exp6$	$Exp7$
linked clauses	yes	yes	yes	yes	yes	yes	yes
function-free	no	yes	yes	yes	yes	yes	yes
number of literals	3	3	3	3	3	3	3
number of new variable	∞	∞	0	0	∞	∞	∞
argument types	none	none	none	none	$AT1$	none	none
predicate types	none	none	none	none	none	$PT1$	none
connection paths	$CT2$	$CT2$	$CT2$	$CT1$	$CT2$	$CT2$	$CT2$
fixed arguments	none	none	none	none	none	none	$Args1$
Number of Hypothesis	1075	279	0	100	46	117	3

The bias constituents used in the experiments have been represented in $MILES - BRL$, a graph-based representation formalism for biases. This language is an extension of the unifying representation language for biases described in [Tau93].

6 Conclusion

In this paper, we identified the basic constituents of the biases used in ILP systems, and we discussed the restrictions they impose on the hypothesis clauses with respect to the term distribution.

The effects of several bias elements have been demonstrated by seven experiments. These experiments show that not all combinations of bias constituents are useful, e.g. the combination in $Exp2$. Furthermore, they shows that of using a hypothesis language that allows structured terms increases strongly the number of hypotheses.

However, these experiments should be extended to study the effect of further bias constituents. In addition, new combinations not realized by the biases used in ILP systems can be tested.

References

[AB92] H. Adé and M. Bruynooghe. A comparative study of declarative and dynamically adjustable language bias in concept learning. In *Proc. of the MLC Workshop Bias on Inductive Learning*, 1992.

[BJ93] P. Brazdil and A. Jorge. Exploiting algorithm sketches in ILP. In *Proc. of the Third International Workshop on Inductive Logic Programming ILP-93*, Technical Report, IJS-DP-6707. J. Stefan Institute, 1993.

[DR91] L. De Raedt. *Interactive concept-learning*. PhD thesis, Katholieke Universiteit Leuven, 1991.

[KW91] J.U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. Arbeitspapiere der GMD Nr. 503, GMD, 1991.

[MDR93] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. Technical Report 178, K. U. Leuven, 1993.

[MF90] S. Muggleton and C. Feng. Efficient induction of logic programs. In *First Conference on Algorithmic Learning Theory*, Tokyo, 1990. Ohmsha.

[Mug90] S. Muggleton. Inductive logic programming. In *First Conference on Algorithmic Learning Theory*, Tokio, October 1990. Ohmsha.

[Mug93] S. Muggleton. Inductive logic programming: Derivations, successes and shortcoming. In *Machine Learning: ECML-93, European Conference on Machine Learning, Wien, Austria*. Springer, 1993.

[MWKE93] K. Morik, S. Wrobel, J. Kietz, and W. Emde. *Knowledge Aquisition and Machine Learning: Theory Methods and Appllications*. Academic Press, 1993.

[PK92] M. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9(1), 1992.

[QCJ93] J.R. Quinlan and R.M. Cameron-Jones. Foil: A midterm report. In *Machine Learning: ECML-93, European Conference on Machine Learning*. Springer, 1993.

[Qui90] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Rou92] C. Rouveirol. Extensions of inversion of resolution applied to theory completion. In S. Muggleton, editor, *Inductive logic programming*. Academic Press, 1992.

[STW93] I. Stahl, B. Tausend, and R. Wirth. Two methods for improving inductive logic programming systems. In *Machine Learning: ECML-93, European Conference on Machine Learning, Wien, Austria*. Springer, 1993.

[Tau93] B. Tausend. A unifying representation for language restrictions. In *Proc. of the Third International Workshop on Inductive Logic Programming ILP-93*, Technical Report, IJS-DP-6707. J. Stefan Institute, 1993.

[Utg86] P. Utgoff. Shift of bias for inductive concept learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine learning: An artificial intelligence approach*, volume 2, chapter 5. Morgan Kaufmann, 1986.

[WO91] R. Wirth and P. O'Rorke. Constraints on predicate invention. In *Eighth International Conference on Machine Learning*. Morgan Kaufmann, 1991.