A Guided Tour Through Hypothesis Spaces in ILP

Birgit Tausend

Fakultät Informatik, Universität Stuttgart, Breitwiesenstr. 20-22, D-70565 Stuttgart

Abstract. In spite of the desirable properties of using Horn logic as hypothesis language, the expressivness leads to huge hypothesis spaces containing up to millions of hypotheses for even simple learning problems. Controlling hypothesis spaces by biases requires knowledge on the effects and applicability of biases in different domains. This knowledge can be gained experimentally by comparing the size of hypothesis spaces with respect to the language bias and the application domain. This approach contrasts theoretical comparisons of the complexity where the results are very general and small bias variations mostly cannot be considered. In order to yield more detailed information on small bias variations and to compare the results independently of systems, their implementations and additional more or less hidden biases, we use MILES-CTL for the experiments. As application domains, we selected a function-free domain including family relations and a non-function-free domain including list-processing programs.

1 Introduction

Obviously, expressive hypothesis languages as Horn logic in Inductive Logic Programming (ILP) offer a wide range of desirable properties, e.g., the representation of relational knowledge, or a better understandability of hypotheses. However, a closer look to the size of hypothesis spaces raises the question how a learning system is able to find an acceptable hypothesis among several thousands of alternative hypotheses. Exploiting the background knowledge, another goal of to ILP systems, seems to worsen the problem of very large hypothesis spaces, and one may ask if a step back to attribute-value representations as in LINUS [LD92] is the only reasonable solution.

The feeling of helplessness facing the mere size of hypothesis spaces of even very simple learning problems contrasts the success of ILP systems learning in different more or less real application domains, e.g. GOLEM [MKS92], Progol [SMKS94] or MOBAL [MWKE94, SMAU93]. In spite of the temptation to believe so, this success cannot be ascribed to some kind of magic but is the result of choosing appropriate language, search, or validation biases. Using strong language biases, e.g. predicate types, leads to considerably smaller hypothesis spaces. Another solution of the size problem is to apply heuristic search instead of complete search, e.g., beam or best search, guided by powerful heuristics.

To avoid exploding hypothesis spaces the designer of an ILP system needs to know a lot about biases, e.g.,

- which parameters of inductive learning systems can be affected by biases,
- which effects particular biases have,
- which combinations of biases are useful,
- which biases are appropriate to solve a particular learning problem.

Concerning a particular parameter of the learning system to design, he has to decide either to use more general and widely applicable biases, e.g. the information gain heuristic, or more specific and domain-dependent biases, e.g. predicate types. Another decision concerns the adaption of the learning system that can be supported by parameterizable biases, e.g., the parameters i and j in the ij-determinacy.

The knowledge about biases can be gained either by theoretical considerations or by experiments. Theoretically, the complexity of a learning algorithms with particular biases can be computed [Coh93, KL94, MP94]. Another theoretical approach is to approximate the size of the hypothesis space with respect to the language bias [PK92, Tau94b]. In contrast, experimental comparisons study for example the effects of some language biases on the size of the starting clauses in bottom-up ILP approaches [ADRB94] or on the size of the hypothesis space [PK92].

In this paper, we focus on the experimental setting comparing the effects of a particular class of biases, the language biases in ILP systems measuring the size of the hypothesis space by the number of clauses to be constructed with respect to the bias and the application domain. In the experiments, we use two domains, the domain of family relations and the domain of list-processing programs. The biases are represented in MILES-CTL [Tau94d, Tau94a], a declarative, scheme-based representation for language biases in ILP since using MILES-CTL offers several advantages. First, in contrast to the experiments in [ADRB94], language biases can be compared independently of particular systems and their implementations, i.e., independently of other more or less obvious biases in the systems compared. Second, a wider range of language biases as well as new combinations can be investigated because most of them can be represented. Third, in comparison to theoretical results, experiments using MILES-CTL give more detailed information on the effects of language biases since the represention enables small bias variations.

This paper is organized as follows. The following section describes the problem setting of ILP and classifies the language biases used in ILP systems. Section 3 gives a short introduction to MILES-CTL. In section 4, different language biases are experimentally evaluated with respect to the size of the hypothesis space using the function-free domain of family relations. Section 5 describes similar experiments in the domain of list-processing programs that is not function-free. The last section discusses the results and concludes.

2 ILP and Language Biases

Using the general model in [Tau94c] shown in figure 1, the problem setting of Inductive Logic Programming can be defined as follows. Given positive and negative examples E and a logic program B as background knowledge, the task is

to learn a logic program H that fulfills A. Thus, L_H is restricted to subclasses of Horn logic. All other parameters may vary with respect to particular algorithms and systems, e.g. accptance criteria A mostly include the necessity, the completeness and the consistency.

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, i.e., classified fact or clauses with their classification,
  - the representation language of the background knowledge L_B, i.e., facts and/or clauses,
  - 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 A.

Table 1. A general model for inductive learning systems

The hypothesis language is defined by the so-called language bias. Since a particular set of Horn clauses can be defined by a signature and rules to construct terms and clauses, the languages biases can be classified with respect to which of them they affect. In addition, several model-dependent biases are used in ILP, e.g. functionality, that depend on the Herbrand universe derived from the background knowledge B and the examples E. This leads to the following classification of biases described in more detail in [Tau94a]

- **Signature**: function-free clauses, constant-free clauses, predicate types, argument types, sort hierarchies, mode declarations, ...;
- **Rules of Term Construction**: no term construction, term construction limited to head terms/subterms, limited depth of term construction, ...;
- **Rules of Clause Construction**: generative/ range restricted clauses, constrained clauses, limited number of new variables, restrictions of variable sharing, limited clause length, limited arity of literals, ...;
- **Model-dependend Restrictions**: functionality, determinacy, ...

Although these biases are very different, MILES-CTL [Tau94b] offers a unifying approach to represent most of them declaratively.

3 Representing the Language Bias in CTL

The basic idea of the representation MILES-CTL [Tau94d] is to describe sets of hypotheses by schemes. A scheme for a hypothesis clause called clause template includes schemes for each literal in the clause. Similar to a record data type, a literal template consists of several identifiers followed by a scheme variable or a constant. 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 literals, 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, argument or predicate types, or the depth of the covered literals.

Additionally, the vocabulary Σ including predicates, functors, and types, and an instantiation function I have to be specified in order to give a complete declaration of a hypothesis language $L_H = (T, \Sigma, I)$ in MILES-CTL. Given T and Σ, I constructs hypotheses by instantiating the scheme variables in a clause template.

For example, the clause template $T1$ with

$$T1: \begin{bmatrix} predicate : P21, \\ arguments : A21 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P22, \\ arguments : A22, \\ predicate\_type : comp \end{bmatrix}, \begin{bmatrix} predicate : P23, \\ arguments : A23, \\ arity : Ar23 || (Ar23 \geq 3) \end{bmatrix}.$$

covers clauses with two body literals where the predicate type of the first is *comp* and the arity of the second is greater or equal than 3. Let the language L_H be defined by $T = \{T1\}$, Σ containing the predicates $member/2$,$leq/2$ and $geq/2$ of type *comp* and the predicates $append/3, reverse/2$ and $intersection/3$ of type *listp*, and an instantiation function I. Given a clause head $intersection([A|B], C, [A|E])$ induced from the examples, the following clauses can be constructed, for example:

$intersection([A|B], C, [A|E]) \leftarrow member(A, C), intersection(B, C, E).$
$intersection([A|B], C, [A|E]) \leftarrow member(A, C), intersection([A|B], C, E).$
$intersection([A|B], C, [A|E]) \leftarrow member(A, C), intersection(B, C, [A|E]).$
$intersection([A|B], C, [A|E]) \leftarrow member(B, C), intersection(B, C, D).$
$intersection([A|B], C, [A|E]) \leftarrow member([A|B], C), append(B, C, D).$
$intersection([A|B], C, [A|E]) \leftarrow geq(A, C), intersection(B, C, E).$
$intersection([A|B], C, [A|E]) \leftarrow geq(B, C), intersection(B, C, E).$
$intersection([A|B], C, [A|E]) \leftarrow leq(A, C), intersection([A|B], C, E).$
...

However, the clauses

$intersection([A|B], C, [D|E]) \leftarrow reverse(B, E), intersection(B, C, E).$
$intersection([A|B], C, [A|E]) \leftarrow member(A, C), leq(A, A).$
$intersection([A|B], C, [A|E]) \leftarrow member(A, C), reverse(B, E).$

cannot be constructed because of the first, respectively the second, body literal description in $T1$.

4 Effects of Language Biases in a Function-Free Domain

In the first five experiments, we restrict the hypothesis language to function-free Horn clauses because this restriction applies for a wide range of ILP systems,

e.g., CLINT[DR91], ITOU[Rou92], FOIL [Qui90]. A function-free domain often used for comparisons is the domain of family relations. In the following experiments, the concept *father* is learned from a set of positive and negative examples given by

positive	**negative**
$father(theo, marc)$.	$father(penelope, arthur)$.
$father(theo, anna)$.	$father(penelope, victoria)$.
$father(peter, lisa)$.	$father(anne, lisa)$.
$father(peter, jennifer)$.	$father(anne, jennifer)$.
$father(peter, ina)$.	$father(anne, ina)$.
$father(klaus, jan)$.	$father(maria, jan)$.
$father(rainer, robin)$.	$father(helga, robin)$.
$father(andreas, susanne)$.	$father(doris, susanne)$.

The hypothesis languages in the experiments are represented by clause templates in $\mathcal{T}_1$ and the signature Σ_1 given by

$$\begin{aligned}\Sigma_1 = [\ & s\_data : \{person, numeric\},\\ & s\_pred : \{relative, relation, sex, date, comp\},\\ & s\_relation : \{\ \},\\ & func : \{\ \},\\ & const : \{\ \}\\ & pred : Predicates,\\ & modes : \{\ \}\ ]\end{aligned}$$

The sets *s_data* and *s_pred* contain the argument and predicate types, *func* and *const* the functors and constants to be used in hypotheses, *pred* contains the predicates and *modes* the mode declarations in this domain. *Predicates* is varied to study the effect of increasing background knowledge B, and it includes subsets of

$$\begin{aligned}\{\ & father(X : person, Y : person)/relative\\ & male(X : person)/sex,\\ & female(X : person)/sex,\\ & parent(X : person, Y : person)/relative,\\ & greater(X : person, Y : numeric)/comp\\ & married(X : person, Y : person)/relation\\ & age(X : person, Y : numeric)/date\ \}.\end{aligned}$$

Given Σ_1, the most general hypothesis language covering clauses with arbitrary literals is defined by clause templates $T_{unrestricted}$ with

$$T_{unrestricted} : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2 \end{bmatrix}, \ldots, \begin{bmatrix} predicate : Pn, \\ arguments : An \end{bmatrix}.$$

The first experiment FD_1 compares the size of the hypothesis space with respect to biases affecting the signature of the clauses. Among these biases, argument types can be exploited either by defining partially fixed argument types for the arguments of each literal or by checking the argument type of a term

distributed to a particular argument position with the definition of the predicate in the signature. Predicate types can be fixed or specified as one of several alternatives. Thus, we vary Σ_1 and $\mathcal{T}_1$ including one of the clause templates for

– checking argument types

$$T_1 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \\ argument\_types : T1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2, \\ argument\_types : T2 \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3, \\ argument\_types : T3 \end{bmatrix}.$$

– partially fixed argument types

$$T_2 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \\ argument\_types : T1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2, \\ argument\_types : T2 : (\{person : \langle 1 \rangle\} \subseteq T2) \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3, \\ argument\_types : T3 : (\{person : \langle 1 \rangle, \\ \qquad\qquad person : \langle 2 \rangle\} \subseteq T3) \end{bmatrix}.$$

– fixed predicate types

$$T_3 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2, \\ predicate\_type : sex \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3, \\ predicate\_type : relative \end{bmatrix}.$$

– alternative predicate types

$$T_4 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2, \\ predicate\_type : Y2 : (Y2 \in \{sex, date\}) \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3, \\ predicate\_type : Y3 : (Y3 \in \{relative, relation\}) \end{bmatrix}.$$

The size of the hypothesis space is measured by the number of hypotheses constructed from $\mathcal{T}_1$ and Σ_1 in FD_1 with respect to an increasing set of predicates in *Predicates*. The results are shown in the following table.

	Predicates in Σ_1 in experiment FD_1					
Clause Template	*father female*	*father female male*	*father female male parent*	*father female male parent greater*	*father female male parent greater married*	*father female male parent greater married age*
$T_{unrestricted}$	302	418	1238	2494	4168	6232
T_1 : *check arg. types*	302	418	1238	1374	2678	3035
T_2 : *part. arg. types*	51	122	210	210	298	298
T_3 : *fix pred. types*	44	88	176	176	176	176
T_4 : *altern. pred. types*	44	88	176	176	274	918

This experiment shows that predicate types strongly restrict the hypothesis space. The reason is that all variabilizations of a predicate are excluded if its type does not agree with the type specified for the literal. Fixed argument types as well result in strong reductions of the number of hypotheses because they implicitely fix the minimum arity of the predicate to be used. Thus, all variabilizations of literals with smaller arity are excluded. Compared to the other biases, checking argument types in this domain imposes minor restrictions on the search space. But satisfying reductions of more than 50% compared with the number of hypotheses constructed from an unrestricted clause template are gained if the types information can be exploited as in the last columns of the table.

In the next experiment FD_2, we study the effect of biases restricting the rules of clause construction. In particular, we compare several widely used biases, the restrictions to constrained and generative clauses, to clauses without recursive literals and to clauses with unique variables. Additionally, a new bias is compared, namely the restriction to clauses without single variable, that is similar to the search bias in SIERES [WO91] and INDICO [STW93]. This bias is defined by

$$A \textit{ clause } C : l_1 \leftarrow l_1 \ldots l_n \textit{ does not include single variables if}$$
$$\forall l_i \in C : vars(l_i) \subseteq vars(\{l_0, l_1, \ldots l_n\} - \{l_i\})$$

For example, the clause $father(A, B) : -male(A), parent(A, C)$ includes the single variable C.

These biases in experiment FD_2 are represented by the clause templates for

- constrained clauses

$$T_1 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2, \\ new\_variables : \emptyset \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3, \\ new\_variables : \emptyset \end{bmatrix}.$$

- range restricted/generative clauses

$$T_2 : \begin{bmatrix} predicate : P1, \\ arguments : A1, \\ generative : yes \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2 \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3 \end{bmatrix}.$$

- clauses without recursive literals

$$T_3 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2 \| (P2 \neq P1), \\ arguments : A2 \end{bmatrix}, \begin{bmatrix} predicate : P3 \| (P3 \neq P1), \\ arguments : A3 \end{bmatrix}.$$

- clauses with unique variables

$$T_4 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2, \\ unique\_variables : yes \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3, \\ unique\_variables : yes \end{bmatrix}.$$

- clauses without single variables

$$T_5 : \begin{bmatrix} predicate : P1, \\ arguments : A1, \\ no\_singles : yes \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2 \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3 \end{bmatrix}.$$

A growing number of predicates in *Predicates* yields the numbers of hypotheses in the hypothesis space shown in the following table.

Clause Templates	Predicates in Σ_1 in experiment FD_2				
	father *female*	*father* *female* *male*	*father* *female* *parent*	*father* *female* *male* *parent*	*father* *female* *male* *parent* *married*
$T_{unresrricted}$	302	418	1033	1238	2498
T_1 : *constrained*	30	56	90	132	240
T_2 : *range resticted*	116	150	406	468	962
T_3 : *no recursive literals*	7	34	302	418	1238
T_4 : *unique variables*	206	302	683	848	1672
T_5 : *no single variables*	71	111	246	314	623

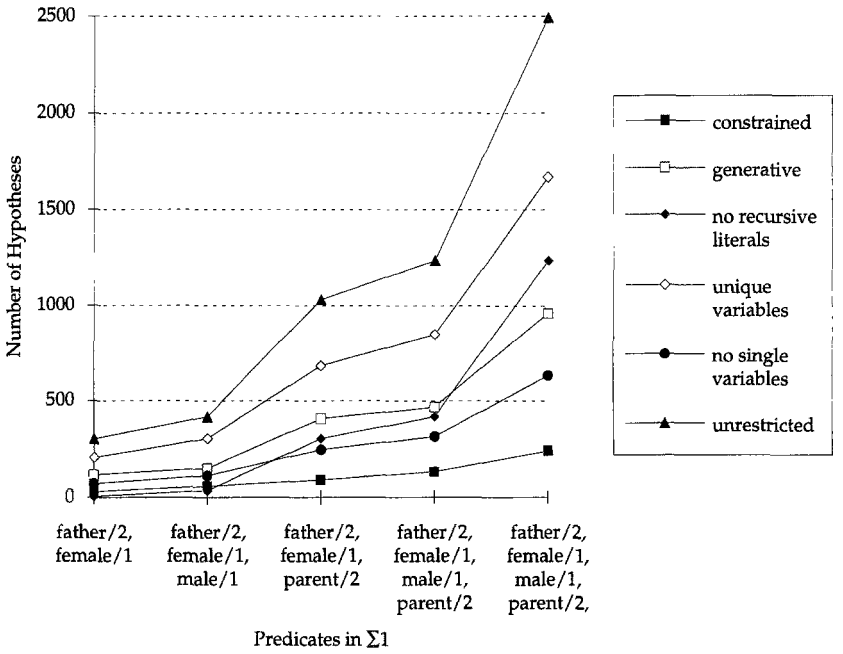

Fig. 1. The size of the hypothesis space in experiment FD_2

In this experiment, the restriction to constrained clauses is the strongest among the biases as shown in the table and in figure 1. Although *father* can be learned, many other concepts are excluded because the body literals of hypothesis clauses must not include new variables. The new restriction to clauses without single variable also imposes considerable restrictions to the hypothesis space. In contrast to the first restriction, this bias is useful from a computational point of view since it excludes unbound variables in the body of hypothesis clauses. Excluding recursive literals helps only when the number of predicates in the

background knowledge is small. If it increases, the reductions decrease if compared to the number of hypotheses covered by the unrestricted clause template. The weakest bias in this experiment is the restriction to clauses with unique variables because it does not exclude many variabilizations.

The next experiment FD_3 investigates the effect of controlling the number of new variables since the bias of several ILP systems mainly relies on this restriction, e.g., CLINT [DR91]. We vary the maximum number of new variables by the scheme conditions i_1 and i_2 in the following clause template T in $\mathcal{T}_1$.

$$T: \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2, \\ new\_variables : i_1 \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3, \\ new\_variables : i_2 \end{bmatrix}.$$

Experiment FD_3 results in the following sizes of hypothesis spaces.

		Predicates in Σ_1 in experiment FD_3				
		father *female*	*father* *female* *male*	*father* *female* *parent*	*father* *female* *male* *parent*	*father* *female* *male* *parent* *married*
	i_1, i_2					
1	$i_1 : \emptyset$ $i_2 : \emptyset$	30	56	90	132	240
2	$i_1 : \emptyset$ $i_2 : N3 \mid\mid (\lvert N3\rvert \leq 1)$	78	128	240	324	608
3	$i_1 : N2 \mid\mid (\lvert N2\rvert \leq 1)$ $i_2 : \emptyset$	118	182	390	500	976
4	$i_1 : N2 \mid\mid (\lvert N2\rvert \leq 1)$ $i_2 : N3 \mid\mid (\lvert N3\rvert \leq 1)$	254	362	855	1044	2080
5	$i_1 : \emptyset$ $i_2 : N3 \mid\mid (\lvert N3\rvert \leq 2)$	84	136	260	348	656
6	$i_1 : N2 \mid\mid (\lvert N2\rvert \leq 2)$ $i_2 : \emptyset$	187	205	460	578	1141
7	$i_1 : N2 \mid\mid (\lvert N2\rvert \leq 1)$ $i_2 : N3 \mid\mid (\lvert N3\rvert \leq 2)$	268	379	905	1100	2197
8	$i_1 : N2 \mid\mid (\lvert N2\rvert \leq 2)$ $i_2 : N3 \mid\mid (\lvert N3\rvert \leq 1)$	287	400	979	1178	2368
9	$i_1 : N2 \mid\mid (\lvert N2\rvert \leq 2)$ $i_2 : N3 \mid\mid (\lvert N3\rvert \leq 2)$	302	418	1033	1238	2498
10	$i_1 : N2 \mid\mid (\lvert N2\rvert \leq 3)$ $i_2 : N3 \mid\mid (\lvert N3\rvert \leq 3)$	302	418	1033	1238	2498

In this table, row 10 equals row 9 as the maximum arity of the predicates in this domain is 2. Thus, limiting the number of new variables to a bound greater than the maximum arity in the domain does not restrict the hypothesis space. Increasing the background knowledge shows that the growth of the hypothesis space depends on the arity of the predicate. For example, adding unary predicates like *female*/1 does not increase the number of hypothesis as much as *parent*/2.

This table shows that restricting the number of new variables is not very effective in this domain. For example, there is only a small difference between the rows 8 and 7, and row 9. Only excluding all new variables in one or more body literals as in the rows 1 to 3 or 5 considerably restricts the hypothesis space. The reason is that the maximum arity of the predicates in this domain is 2. Thus, at most two new variables can be distributed to the argument positions and restricting their number to one does not reduce very much the number of variables to be distributed.

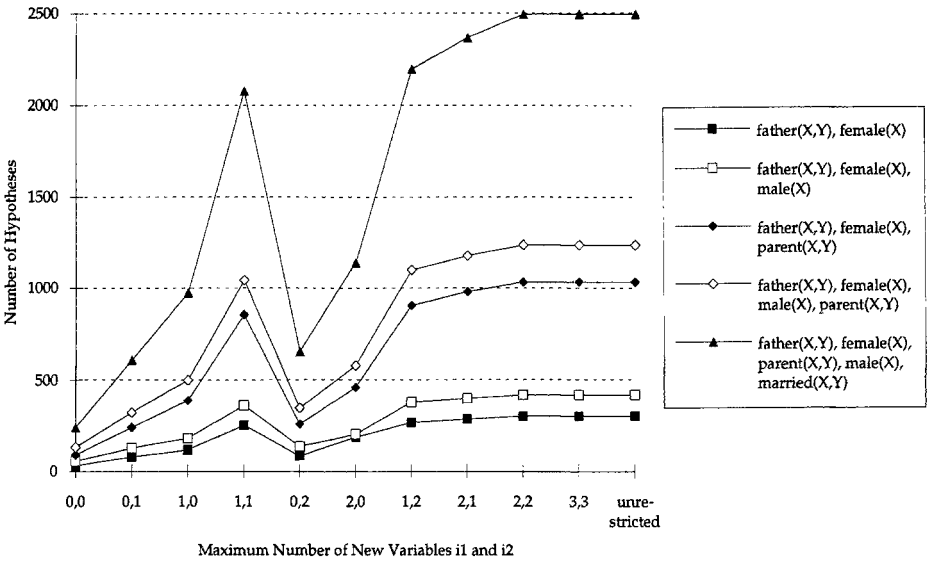

Fig. 2. The size of the hypothesis space in experiment FD_3

Another effect to be seen from the results is that restricting the number of new variables in the first literals of a clause imposes stronger restrictions than restricting the latter literals. For example, the clause templates in row 5 and 6 only differ in the order of the restrictions of new variables. But the order in row 5 results in stronger restrictions since there are less variables available to be distributed to the argument positions of the second literal as in case of row 6.

In experiment FD_4, the growth of the hypothesis space with respect to the limit of the clause length is investigated. $\mathcal{T}_1$ includes one of the three clause templates $T1$, $T2$, and $T3$ that differ in the number of literal descriptions but are not restricted by further biases.

$$T_1 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2 \end{bmatrix} .$$

$$T_2 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2 \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3 \end{bmatrix} .$$

$$T_3 : \begin{bmatrix} predicate : P1, \\ arguments : A1 \end{bmatrix} \leftarrow \begin{bmatrix} predicate : P2, \\ arguments : A2 \end{bmatrix}, \begin{bmatrix} predicate : P3, \\ arguments : A3 \end{bmatrix}, \begin{bmatrix} predicate : P4, \\ arguments : A4 \end{bmatrix}.$$

Using these clause templates and varying the predicates in Σ_1 results in the numbers of hypotheses shown in the following table.

	Predicates in Σ_1 in experiment FD_4				
Clause Template	*father married age*	*father married age female*	*father married age female male*	*father married age female male parent*	*father married age female male parent greater*
T_1 : 1 *body literals*	18	21	24	30	36
T_2 : 2 *body literals*	534	679	844	1358	1994
T_3 : 3 *body literals*	23304	31432	41456	86908	160620

These results show that the clause length is an important factor for the size of the hypothesis space, in particular when the background knowledge includes many predicates.

In experiment FD_5, both restrictions affecting the signature and the rules for clause construction are combined, namely the restrictions to constrained and generative clauses, to clauses without recursive literals, to clauses unique variables, and to using predicate types. *Predicates* includes the predicates *father*, *female*, *male*, *parent*, and *married* as given in experiment FD_1.

	Bias 1 in experiment FD_5				
Bias 2	*constrained*	*predicate types*	*generative*	*no recur. literals*	*unique variables*
unique variables	90	136	704	848	1672
no recursive literals	132	88	468	1238	
generative	200	56	962		
predicate types	32	176			
constrained	240				

This table shows that weak biases like the restriction to clauses with unqiue variables can be considerably improved by further biases since combinations lead to reductions between 5.3% and 50% of the original size of the hypothesis space. A similar effect occurs in case of clauses without recursive literals, another weak bias, where the hypothesis space can be reduced up to 7.1% of its original size.

In general, combining biases of different classes strongly decreases the size of the hypothesis space, e.g. combining predicate types with the restrictions affecting clause construction. In particular, combinations with the restrictions to generative constrained clauses yield strong reductions. Obviously, a second reason is that in both cases strong restrictions have been combined whereas combinations with weaker restrictions like clauses without recursive literals result in smaller reductions.

Another useful combination is given by the restriction to clauses with unique variables and constrained clauses since the clause must not include new variables and the distribution of the old variables is restricted.

5 Effects of Language Biases in a Non-Function-Free Domain

In contrast to the family domain, the second domain LPD containing list-processing clauses allows for using functors in the literals. In the following experiments the concept *merge* is to be learned given the positive and negative examples

positive	negative
$merge([1],[2],[1,2])$.	$merge([1],[2],[2,1])$.
$merge([2,3,4,5],[4,7],[2,3,4,4,5,7])$.	$merge([55,66],[22,33,55],[55,66,22])$.
$merge([55,66],[22,33,55],[22,33,55,55,66])$.	$merge([2],[54,66,77,88,97],[])$.
$merge([5],[54,66,77],[54,66,5,77])$.	$merge([22,23,24,25],[],[23,24,25])$.
$merge([1],[5,10],[1,5,10])$.	$merge([4],[3],[4])$.
$merge([22,23,24],[25],[22,23,24,25])$.	$merge([22,23,24],[25],[25])$.
$merge([22,23,24,27],[12],[12,22,23,24,27])$.	$merge([22,23,24],[12],[22,12,23,24])$.
$merge([24,33],[25],[24,25,33])$.	$merge([22,23,24,25],[],[24,25])$.

and the signature Σ_2 given by

$$\begin{aligned}\Sigma_2 = [\ & s\_data : \{list, number, atomic\},\\ & s\_pred : \{comp, listbasic, listcomb, listset\},\\ & s\_relation : \{number : atomic\}\\ & func : \{\ .:(X : atomic, Y : list)/list\},\\ & const : \{\ \}\\ & pred : Predicates,\\ & modes : \{\ \}\ ],\end{aligned}$$

and the head literal $merge([A|B],[C|D],[A|E])$. The set $Predicates$ in Σ_2 including clauses from the set

$$\begin{aligned}\{& merge(X : list, Y : list, Z : list)/listcomb,\\ & leq(X : number, Y : number)/comp,\\ & geq(X : number, Y : number)/comp,\\ & reverse(X : list, Y : list)/listcomb,\\ & append(X : list, Y : list, Z : list)/listbasic,\\ & intersect(X : list, Y : list, Z : list)/listset,\\ & union(X : list, Y : list, Z : list)/listset,\\ & subset(X : list, Y : list, Z : list)/listset\ \}\end{aligned}$$

is varied as in the experiments in the familiy domain in order to study the biases with resepct to increasing background knowledge.

In experiment LPD_1, we study the effect of increasing the number of literals in $Predicates$ for unrestricted clauses of length 3, i.e., clause template $T2$ from experiment FD_4 is used. This experiment yields the results shown in figure 3.

Compared to the results of experiment FD_4 in the family domain, the hypothesis space grows much faster. The reason is that not only variables can be

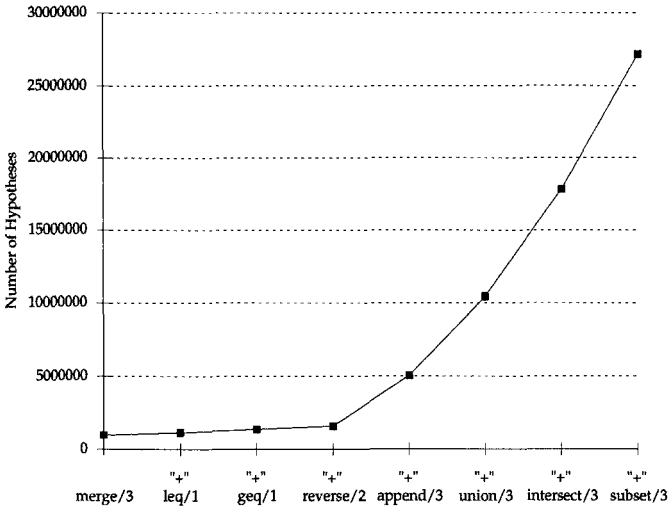

Fig. 3. The size of the hypothesis space in experiment LPD_1

distributed to argument positions of the body literals, but all subterms of the head literal. Thus, further language biases or powerful search biases have to be used in order to learn effectively in this domain.

In the next experiment LPD_2, several of the language biases studied in experiment FD_1 and FD_2 are compared in the non-function-free domain of list-processing programs. To learn the concept *merge*, the respective clauses from the family domain defined in the previous section are used. So, experiment LPD_2 results in the numbers of hypotheses shown in the next table.

	Predicates in Σ_2 in experiment LPD_2			
Clause Template	*merge*	*merge leq*	*merge leq geq*	*merge leq geq append*
T_2 : *constrained*	46010	47742	49506	192282
T_2 : *predicate types*	0	3087	6174	12348
T_3 : *no recursive literals*	0	116	482	151778
T_4 : *unique variables*	3478	4124	4892	16502

In contrast to the family domain, the restriction to constrained clauses results in minor reductions whereas the restriction to unique variables gives better results because it excludes all terms not resulting in unique variables in the literals. As in the family domain, predicate types turn out to be a powerful restriction. Again, the effect of the restriction to clauses without recursive literals is strong as long as the total number of predicates is small.

As in experiment FD_3 in the family domain, we investigate the effects of limiting the maximum number of new variables. For this experiment LPD_3, clause template T from experiment FD_3 is used and the scheme conditions i_1 and $i2$ are varied as shown in the following table. This experiments yields the results shown in the following table.

i_1, i_2	Predicates in Σ_2 in exp. LPD_3			
	merge	*merge leq*	*merge leq geq*	*merge leq geq append*
$i_1 : \emptyset, i_2 : \emptyset$	46010	47742	49506	192282
$i_1 : \emptyset, i_2 : N3 \parallel (\lvert N3\rvert \leq 1)$	98496	102054	98496	410026
$i_1 : \emptyset, i_2 : N3 \parallel (\lvert N3\rvert \leq 2)$	107715	112347	117075	451292
$i_1 : \emptyset, i_2 : N3 \parallel (\lvert N3\rvert \leq 3)$	107930	112566	117298	452170
$i_1 : N2 \parallel (\lvert N2\rvert \leq 1), i_2 : \emptyset$	103370	107034	110762	407114
$i_1 : N2 \parallel (\lvert N2\rvert \leq 1), i_2 : N3 \parallel (\lvert N3\rvert \leq 1)$		127848		
$i_1 : N2 \parallel (\lvert N2\rvert \leq 1), i_2 : N3 \parallel (\lvert N3\rvert \leq 2)$		134480		
$i_1 : N2 \parallel (\lvert N2\rvert \leq 2), i_2 : \emptyset$		95944		

In general, experiment LPD_3 shows that restricting the maximum number of new variables does not apply as well as the family domain. Again, the reason is that reducing the set of terms to be distributed to the argument positions by the new variables does not have strong effects because the total number is large.

6 Conclusions

For comparing language biases, the representation MILES-CTL turned out to be very useful because it enables fine-grained variations of the language bias and investigating language biases in isolation of other biases in the learning system. In addition, new biases, e.g. the restriction to clauses without single variables, can be tested easily.

The results of the experiments in this paper give detailed knowledge of the effects of language biases in the two domains that have been investigated. As the experiments show, some of the biases, e.g. predicate types, perform well in both domains. Other biases, e.g. the restriction to clauses with unqiue variables, apply well in one domain but not in the other. Another result is that biases often used in ILP, e.g. restrictions of the new variables, result in minor restrictions compared to other language biases. A general observation is that biases from different classes in section 2 can be combined successfully, e.g., using predicate types together with the restriction to generative clauses.

Because of the size of the hypothesis spaces, especially in the domain of list-processing programs, efficient learning procedures require carefully selected language biases and additional powerful search biases. Most of the biases studied in this paper are syntactic biases applying to many domains. However, for better restrictions, the designer of new ILP systems should try to find and use more specific and domain-dependent restrictions like predicate types because this knowledge is not well exploited by the biases currently used in ILP.

References

[ADRB94] H. Adé, L. De Raedt, and M. Bruynooghe. Declarative bias for specific-to-general ILP systems. Deliverable D.KUL.4, ESPRIT Proj. 6020 ILP, 1994.

[Coh93] W.W. Cohen. Rapid prototyping of ILP systems using explicit bias. In *Proc. of IJCAI-93 Workshop on Inductive Logic Programming.* Morgan Kaufmann, 1993.

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

[KL94] J.-U. Kietz and M. Lübbe. An efficient subsumption algorithm for inductive logic programming. In S. Wrobel, editor, *Proc. of 4th Workshop on Inductive Logic Programming ILP-94*, GMD-Studien Nr. 237, 1994.

[LD92] N. Lavrac and S. Dzeroski. Inductive learning of relational descriptions from noisy examples. In S. Muggleton, editor, *Inductive Logic Programming.* Academic P., 1992.

[MKS92] S.H. Muggleton, R.D. King, and M.J.E. Sternberg. Protein secondary structure prediction using logic-based machine learning. In *Protein Engineering*, 5(7), 1992.

[MP94] S.H. Muggleton and C.D. Page. A learning model for universal representations. In S. Wrobel, editor, *Proc. of 4th Workshop on Inductive Logic Programming ILP-94*, GMD-Studien Nr. 237, 1994.

[MWKE94] K. Morik, S. Wrobel, J. Kietz, and W. Emde. *Knowledge Acquisition and Machine Learning: Theory Methods and Applications.* Academic P., 1994.

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

[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.

[SMAU93] E. Sommer, K. Morik, J.M. Andre, and M. Uszynski. What on-line machine learning can do for knowledge acquisition - a case study. GMD-Studien Nr. 757, GMD, St. Augustin, 1993.

[SMKS94] A. Srinivasan, S.H. Muggleton, R.D. King, and M.J.E. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proc. of 4th Workshop on Inductive Logic Programming ILP-94*, GMD-Studien Nr. 237, 1994.

[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.

[Tau94a] B. Tausend. *Beschränkungen der Hypothesensprache und ihre Repräsentation in der Induktiven Logischen Programmierung.* Dissertation, Fakultät Informatik, Universität Stuttgart, 1994. (in German).

[Tau94b] B. Tausend. Biases and their effects in inductive logic programming. In *Machine Learning: ECML-94, European Conference on Machine Learning, Catania, Italy.* Springer, 1994.

[Tau94c] B. Tausend. Modelling inductive learning for knowledge acquisition tasks. In *ECAI 94 Workshop Integration of Knowledge Acquisition and Machine Learning. Amsterdam, NL*, 1994.

[Tau94d] B. Tausend. Representing biases for inductive logic programming. In *Machine Learning: ECML-94, European Conference on Machine Learning, Catania, Italy.* Springer, 1994.

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