Declarative Bias for Specific-to-General ILP Systems*

HILDE ADÉ

hilde@cs.kuleuven.ac.be

LUC DE RAEDT

lucdr@cs.kuleuven.ac.be

MAURICE BRUYNOOGHE

maurice@cs.kuleuven.ac.be

*Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium*

Editor: M. des Jardins and D. Gordon

Abstract. A comparative study is presented of language biases employed in specific-to-general learning systems within the Inductive Logic Programming (ILP) paradigm. More specifically, we focus on the biases employed in three well known systems: CLINT, GOLEM and ITOU, and evaluate both conceptually and empirically their strengths and weaknesses. The evaluation is carried out within the generic framework of the NINA system, in which bias is a parameter. Two different types of biases are considered: syntactic bias, which defines the set of well-formed clauses, and semantic bias, which imposes restrictions on the behaviour of hypotheses or clauses. NINA is also able to shift its bias (within a predefined series of biases), whenever its current bias is insufficient for finding complete and consistent concept definitions. Furthermore, a new formalism for specifying the syntactic bias of inductive logic programming systems is introduced.

Keywords: inductive logic programming, declarative bias, specific-to-general learning systems, shift of bias, generic framework

1. Introduction

Inductive Logic Programming (ILP) (see (Muggleton, 1992), (Muggleton & De Raedt, 1994)) is a research area that recently emerged at the intersection of Logic Programming and Machine Learning. It focuses on logical theories for induction, on efficient implementations of inductive algorithms, and on practical applications, see e.g. King et al., 1992, Muggleton et al., 1992, and Lavrač et al., 1993. Benefiting from the results achieved in both inductive learning and logic programming, it aims at overcoming the limitations of early inductive systems and at extending the deductive logic programming paradigm towards the use of induction.

Most ILP systems address the problem of concept learning from examples (see Muggleton & De Raedt, 1994)). The main difference from classical concept learning techniques is that an expressive first order formalism is used instead of a limited attribute value representation. Additional advantages are that it is easy to use background knowledge, and that the results are usually transparent to humans.

In this paper, we focus on the language bias issue in inductive logic programming. Language bias determines the way hypotheses are restricted, both in their syntax and in their behaviour. Language bias is of crucial importance for inductive logic programming,

* This paper extends the papers (Adé & Bruynooghe, 1992) and (Rouveirol et al., 1993).

because the hypothesis spaces of logic programs are usually much larger (often infinite) than those of attribute value representations.

In the first part of this paper, we develop a conceptual framework for declarative language bias specification. Two types of biases are distinguished: syntactic bias, which imposes restrictions on the form of clauses in hypotheses, and semantic bias, which imposes restrictions on the behaviour of induced hypotheses. Furthermore, a new general framework for declarative (syntactic) bias specification is presented. It integrates the formalisms by Bergadano (Bergadano, 1993) and the MOBAL team (Emde et al., 1983), (Kietz & Wrobel, 1992).

In the second part of the paper, we study important biases employed in specific-to-general inductive logic programming systems such as ITOU (Rouveirol & Puget, 1989), (Rouveirol, 1991), GOLEM (Muggleton & Feng, 1990), and CLINT (De Raedt, 1992), (De Raedt & Bruynooghe, 1992). As for other concept learning systems (Mitchell, 1982), (De Raedt & Bruynooghe, 1992), specific-to-general search is one of the two most common strategies. In this second part, we first present the generic specific-to-general ILP system NINA. In NINA, the semantic and syntactic bias are parameters. Crucial in this respect will be a generic procedure that turns positive examples into so-called *starting clauses*, which are most specific clauses (for a given bias and background theory) covering a given positive example. Starting clauses corresponding to several positive examples are later generalized using the *lgg* operator of (Plotkin, 1970), (Plotkin, 1971), which is also employed in Muggleton and Feng's GOLEM. NINA is also able to shift its bias within a predefined series of language biases (syntactic or semantic) using a bias shift operator as in CLINT. This allows the NINA system to modify its current language bias whenever it proves to be insufficient. Secondly, we use NINA to empirically and conceptually evaluate the biases employed in three well-known specific-to-general systems: ITOU, GOLEM and CLINT. In particular, we evaluate the determinacy restriction of GOLEM, the shift of bias of CLINT and the use of non-factual evidence in ITOU. This comparison results in a better understanding of the effects of various types of biases, which may then help to determine which biases are suited for which types of learning tasks.

This paper is organised as follows: in Section 2, we introduce some inductive logic programming concepts; in Section 3, we present a conceptual framework for bias and its specification; in Section 4, an overview of the NINA system is provided; in Section 5, we evaluate different biases for inductive logic programming; finally, in Section 6, we conclude.

2. Terminology and problem specification

We first introduce some standard (inductive) logic programming concepts.

Definition 1. A definite clause is an implicitly universally quantified logical formula $h \leftarrow b_1, \dots, b_n$, where h and the b_i are atoms.

Notation 1 *The background theory T and the hypotheses H are sets of definite clauses. $T \models c$ denotes that c is logically entailed by the theory T . $\square$ denotes the empty clause, which is always inconsistent.*

The large majority of ILP approaches derive logical theories from positive and negative examples formulated as true and false *ground facts*, and from background knowledge. Following (De Raedt, 1992), (De Raedt & Bruynooghe, 1992), we use a more expressive form of evidence. Positive evidence is represented as definite clauses while negative evidence is represented as denials.

Definition 2. A denial is an implicitly universally quantified logical formula $\leftarrow b_1, \dots, b_n$, where the b_i are atoms.

Roughly speaking a denial means that not all of the b_i can be true at the same time.

We will require (as (Muggleton & De Raedt, 1994)) that the positive evidence P is logically entailed by the union of the background theory T and the hypothesis H , i.e., $T \cup H \models P$; and that the union of the background theory and the hypothesis is consistent with the denials N , i.e., $T \cup H \cup N \not\models \square$.

Checking whether a theory T is consistent with a denial $\leftarrow b_1, \dots, b_n$, i.e., to determine whether $T \cup \{\leftarrow b_1, \dots, b_n\} \not\models \square$, can be done with a theorem prover. Using PROLOG, one can use T as a program, and execute the query $\leftarrow b_1, \dots, b_n$. Consistency is indicated by finite failure. Success with a substitution θ allows one to locate the inconsistency.

Each false ground fact n can be represented as a denial $\leftarrow n$, and each true ground fact p can be represented as a definite clause $p \leftarrow$. So our representation of evidence generalizes that of most other ILP approaches. An advantage is that one denial can replace many false ground facts. For instance, the denial $\leftarrow \text{father}(X, Y), \text{younger}(X, Y)$ replaces all negative examples for $\text{father}(X, Y)$ where X is younger than Y . Similarly, definite clauses as positive evidence may replace several true ground facts. We therefore argue that the representation of evidence is also a form of language bias, and we evaluate the use of definite clauses and denials (which do not correspond to ground facts) as examples separately. This is relevant for Rouveirol's ITOU system, cf. Section 5, where we show that definite clauses may help to overcome some problems with the syntactic and semantic language bias.

We find it convenient to distinguish between syntactic bias $\mathcal{L}$ and semantic bias $\mathcal{B}$. The former determines the set of syntactically well-formed clauses that constitutes the hypothesis language, whereas the latter imposes restrictions on the behaviour of the induced hypotheses, such as determinacy restrictions (Muggleton & Feng, 1990), coverage requirements or efficiency constraints.

Formally speaking, we have:

Notation 2 *A syntactic bias $\mathcal{L}$ denotes a set of definite clauses. A semantic bias $\mathcal{B}(H, T, P, N)$ denotes a function that inputs a hypothesis H , a theory T , positive evidence P , and negative evidence N , and that returns true or false.*

The aim of ILP as studied in this paper is to start from a theory T , positive and negative evidence P and N , together with bias restrictions $\mathcal{L}$ and $\mathcal{B}$, and to induce a hypothesis

H that is complete (it covers all positive evidence), and consistent (it does not violate any of the denials).

This can be formalised as follows:

Given

- the set of syntactically well-formed clauses $\mathcal{L}$ (the *syntactic bias*),
- a *semantic bias* $\mathcal{B}(H, T, P, N)$,
- a background theory T (represented as a set of definite clauses),
- positive evidence P (represented as a set of definite clauses),
- negative evidence N (represented as a set of denials),

Find: a hypothesis H such that

1. $H \subseteq \mathcal{L}$ (syntactic bias)
2. $\mathcal{B}(H, T, P, N)$ is true (semantic bias)
3. $T \cup H \models P$ (completeness)
4. $T \cup H \cup N \not\models \square$ (consistency)

Our problem specification is illustrated below. Note that predicates are mentioned together with their arity. E.g., `male/1` indicates the predicate `male` that is of arity 1.

Example 1 Assume that T defines the relations `male/1`, `female/1`, `tall/1`, `small/1`, `younger/2`, `parent/2`, `father/2` and `mother/2` of the family displayed in Figure 1 of Section 4.4. Let $\mathcal{L}$, $\mathcal{B}$, P and N be defined as follows:

$$\mathcal{L} = \{\text{grandparent}(X, Y) \leftarrow \text{Body} \mid \text{the only variables in Body are } X, Y, Z\}$$

$\mathcal{B}(H, T, P, N) = \text{true}$ if and only if all clauses $c \in H$ cover at least 5 ground facts for the `grandparent/2` predicate.

$$P = \{\text{grandparent}(X, Y) \leftarrow \text{male}(X), \text{father}(X, Z), \text{male}(Z), \text{father}(Z, Y), \text{male}(Y), \text{tall}(Y), \text{small}(X)\}$$

$$N = \{\leftarrow \text{grandparent}(X, X); \leftarrow \text{grandparent}(X, Y), \text{younger}(X, Y)\}$$

Given this setting, each of the hypotheses H_1 and H_2 are in $\mathcal{L}$ and are complete and consistent.

$$\begin{aligned} H_1 = \{ & \text{grandparent}(X, Y) \leftarrow \text{father}(X, Z), \text{mother}(Z, Y); \\ & \text{grandparent}(X, Y) \leftarrow \text{father}(X, Z), \text{father}(Z, Y); \\ & \text{grandparent}(X, Y) \leftarrow \text{mother}(X, Z), \text{mother}(Z, Y); \\ & \text{grandparent}(X, Y) \leftarrow \text{mother}(X, Z), \text{father}(Z, Y) \} \end{aligned}$$

$$H_2 = \{\text{grandparent}(X, Y) \leftarrow \text{parent}(X, Z), \text{parent}(Z, Y)\}$$

Hypothesis H_2 satisfies the semantic bias but H_1 does not since the first clause does not cover 5 facts. $\square$

Throughout the paper we employ Plotkin's well-known θ -subsumption framework among clauses as the notion of generality. Considering clauses as sets of literals, a clause c_1 is more general than a clause c_2 (or c_1 θ -subsumes c_2) if and only if there exists a substitution θ for which $c_1\theta \subseteq c_2$. For example, the clause $\text{father}(X,Y) \leftarrow \text{parent}(X,Y)$ is more general than $\text{father}(\text{jef},\text{paul}) \leftarrow \text{parent}(\text{jef},\text{paul}), \text{male}(\text{jef})$ because $\{\text{father}(X,Y), \neg\text{parent}(X,Y)\}\theta \subseteq \{\text{father}(\text{jef},\text{paul}), \neg\text{parent}(\text{jef},\text{paul}), \neg\text{male}(\text{jef})\}$ where $\theta = \{X = \text{jef}, Y = \text{paul}\}$. For more information on θ -subsumption, we refer to (Plotkin, 1970), (Plotkin, 1971), (Muggleton & De Raedt, 1994).

3. Bias

In this section, we elaborate on three different aspects of bias: syntactic bias, semantic bias, and shifting the bias. In particular, we define a new formalism to declaratively define the syntactic bias of ILP systems, and present some important forms of semantic bias. We argue that declaratively specifying the bias is advantageous for two reasons. First, declarative bias specifications allow us to easily port the bias from one system to another, hence facilitating comparisons and evaluations of different biases. Second, by making declarative bias explicit, it becomes possible to reason about the bias at the meta-level. This is particularly important when the given bias is insufficient for the learning task and the bias has to be shifted.

3.1. Syntactic bias

3.1.1. A new framework for specifying syntactic bias

Within inductive logic programming, four fundamental formalisms exist to specify syntactic bias (see (Muggleton & De Raedt, 1994) and (Tausend, 1994) for an overview). These are: the attribute description grammars of Cohen (Cohen, 1994), the schemata of Emde et al. (Emde et al., 1983), (Kietz & Wrobel, 1992) and their variants (see e.g. (Tausend, 1994)), the predicate sets of Bergadano et al. (Bergadano & Gunetti, 1993), (Bergadano, 1993), and the parameterized languages of De Raedt and others (De Raedt, 1992), Muggleton & Feng, 1990). It is generally agreed that the attribute description grammars of Cohen are the most flexible and expressive. However, predicate sets and schemata are more declarative because there is a direct correspondence between them and the clauses they represent. In contrast, attribute description grammars are more procedural (as they specify a procedure to generate clauses) and are often harder to understand. Parameterized languages are specified by a number of parameters, which determine the complexity of the syntactic language bias, implying that they are rather procedural. Some important parameters are presented in the next subsection.

Schemata and predicate sets are complementary in the sense that syntactic biases that are easy to represent in one formalism are hard to represent in the other formalism (as we

will show below). This motivated us to design a straightforward integration of schemata and predicate sets. The resulting formalism approaches the expressive power of Cohen's formalism while retaining the declarative spirit of the other two representations. We introduce our framework here, and then show how it relates to the other formalisms.

A syntactic bias $\mathcal{L}$ is specified as a set of clause models. A clause model is an expression of the form $Head \leftarrow Body, BodySet$; where

- $Head$ is either an atom or a variabilized atom A ;
- $Body$ is of the form $A_1, \dots, A_n$ ($n \geq 0$) where the A_i are either atoms or variabilized atoms;
- $BodySet$ is a set $\{A_1, \dots, A_n\}$ ($n \geq 1$), where the A_i are atoms;
- an atom is of the form $p(t_1, \dots, t_n)$ ($n \geq 0$) where p is a predicate and the t_i are terms or term-sets;
- a variabilized atom is of the form $P(t_1, \dots, t_n)$ where P is a predicate variable and the t_i are terms or term-sets;
- a term is a constant, a variable or the application of a functor symbol f to the terms $t_1, \dots, t_n$ yielding the term $f(t_1, \dots, t_n)$;
- a term-set $\{t_1, \dots, t_n\}$ is a set of terms;

Following Bergadano, further syntactic sugar could be added to this language. A full discussion of these further extensions is outside the scope of this paper.

The language specified by a clause model $Head \leftarrow Body, BodySet$ is defined as follows:

1. The language specified by a clause model $Head \leftarrow Body, BodySet$ where $Head$, $Body$ and $BodySet$ do not contain term-sets is:

$\{Head\Theta \leftarrow Body\Theta \cup B \mid \Theta \text{ is a second order substitution that substitutes all predicate variables in } Head \leftarrow Body \text{ with predicate names; and } B \subseteq BodySet\}$

2. The language specified by a clause model $Head \leftarrow Body, BodySet$, where $BodySet = \{b_1, \dots, b_n\}$ contains an atom $b_i = p(T_1, \dots, T_k)$ with T_j a term-set $\{t_1, \dots, t_l\}$ is the language specified by the clause model

$Head \leftarrow Body, BodySet'$ where

$BodySet' = (BodySet - \{p(T_1, \dots, T_k)\}) \cup \{p(T_1, \dots, T_{j-1}, t, T_{j+1}, \dots, T_k) \mid t \in \{t_1, \dots, t_l\}\}$

3. The language specified by a clause model $Head \leftarrow Body, BodySet$ where $Body = b_1, \dots, b_n$ contains an atom (or variabilized atom) $b_i = p(T_1, \dots, T_k)$ with T_j a term-set $\{t_1, \dots, t_l\}$ is the language specified by the set of clause models:

$$\{Head \leftarrow b_1, \dots, b_{i-1}, p(T_1, \dots, T_{j-1}, t, T_{j+1}, \dots, T_k), b_{i+1}, \dots, b_n, BodySet \mid t \in \{t_1, \dots, t_l\}\}$$

4. The language specified by a set of clause models consists of the union of the languages specified by the clause models in the set.

Example 2 Assume that the predicates in the background theory are *male/1*, *female/1*, *parent/2*. Consider the following clause model (*P* and *Q* are predicate variables):

$$\mathcal{L} = \{\text{grandfather}(X,Y) \leftarrow P(\{X,Y\}), Q(X,Z), \{\text{parent}(\{X,Z\},Y)\}\}$$

Using (3) we obtain:

$$\mathcal{L} = \{\text{grandfather}(X,Y) \leftarrow P(Y), Q(X,Z), \{\text{parent}(\{X,Z\},Y)\}; \\ \text{grandfather}(X,Y) \leftarrow P(X), Q(X,Z), \{\text{parent}(\{X,Z\},Y)\}\}$$

Using (2), we obtain:

$$\mathcal{L} = \{\text{grandfather}(X,Y) \leftarrow P(Y), Q(X,Z), \{\text{parent}(Z,Y), \text{parent}(X,Y)\}\}; \\ \text{grandfather}(X,Y) \leftarrow P(X), Q(X,Z), \{\text{parent}(Z,Y), \text{parent}(X,Y)\}\}$$

Using (1), we obtain:

$$\begin{aligned} \mathcal{L} = & \{\text{grandfather}(X,Y) \leftarrow \text{male}(Y), \text{parent}(X,Z); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(Y), \text{parent}(X,Z); \\ & \text{grandfather}(X,Y) \leftarrow \text{male}(X), \text{parent}(X,Z); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(X), \text{parent}(X,Z); \\ & \text{grandfather}(X,Y) \leftarrow \text{male}(Y), \text{parent}(X,Z), \text{parent}(Z,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(Y), \text{parent}(X,Z), \text{parent}(Z,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{male}(X), \text{parent}(X,Z), \text{parent}(Z,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(X), \text{parent}(X,Z), \text{parent}(Z,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{male}(Y), \text{parent}(X,Z), \text{parent}(X,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(Y), \text{parent}(X,Z), \text{parent}(X,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{male}(X), \text{parent}(X,Z), \text{parent}(X,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(X), \text{parent}(X,Z), \text{parent}(X,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{male}(Y), \text{parent}(X,Z), \text{parent}(Z,Y), \text{parent}(X,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(Y), \text{parent}(X,Z), \text{parent}(Z,Y), \text{parent}(X,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{male}(X), \text{parent}(X,Z), \text{parent}(Z,Y), \text{parent}(X,Y); \\ & \text{grandfather}(X,Y) \leftarrow \text{female}(X), \text{parent}(X,Z), \text{parent}(Z,Y), \text{parent}(X,Y)\} \end{aligned}$$

which is the set of definite clauses specified by the initial syntactic bias. $\square$

Schemata (as used in MOBAL) and predicate sets (as used in the FILP system of Bergadano) are special cases of our formalism. Schemata are clause models in which neither predicate sets nor term sets occur, and Bergadano's inductive logic programming

language contains – in essence – clause models without predicate variables. Schemata and predicate sets are complementary in that schemata can easily represent fixed length clauses, whereas predicate sets can easily represent variable length clauses. Furthermore, representing variable length clauses using MOBAL’s formalism requires many schemata, while representing fixed length clauses, which differ only in predicate names, using Bergadano’s formalism requires many clause models. Therefore it is advantageous to integrate both frameworks.

Example 3 *The syntactic bias $\mathcal{L}$ of the previous example can be formulated in Bergadano’s inductive logic programming language as follows:*

$$\mathcal{L} = \{\text{grandfather}(X,Y) \leftarrow \text{male}(\{X,Y\}), \text{parent}(X,Z), \{\text{parent}(\{X,Z\},Y)\}; \\ \text{grandfather}(X,Y) \leftarrow \text{female}(\{X,Y\}), \text{parent}(X,Z), \{\text{parent}(\{X,Z\},Y)\}\}$$

With MOBAL’s schemata, $\mathcal{L}$ can be represented as:

$$\mathcal{L} = \{\text{grandfather}(X,Y) \leftarrow P(Y), Q(X,Z); \\ \text{grandfather}(X,Y) \leftarrow P(Y), Q(X,Z), \text{parent}(X,Y) ; \\ \text{grandfather}(X,Y) \leftarrow P(Y), Q(X,Z), \text{parent}(Z,Y) ; \\ \text{grandfather}(X,Y) \leftarrow P(Y), Q(X,Z), \text{parent}(X,Y), \text{parent}(Z,Y); \\ \text{grandfather}(X,Y) \leftarrow P(X), Q(X,Z) ; \\ \text{grandfather}(X,Y) \leftarrow P(X), Q(X,Z), \text{parent}(X,Y) ; \\ \text{grandfather}(X,Y) \leftarrow P(X), Q(X,Z), \text{parent}(Z,Y) ; \\ \text{grandfather}(X,Y) \leftarrow P(X), Q(X,Z), \text{parent}(X,Y), \text{parent}(Z,Y)\}$$

This shows that our approach allows a concise and elegant declaration of the syntactic bias. The expressivity could be further enhanced by limiting the range of predicate variables, by generalizing term sets, by introducing compound atoms, etc. $\square$

3.1.2. Important parameters in syntactic bias

Whereas the above formalism can be used to declaratively specify the syntactic bias, the computational complexity of the learning task is determined by a number of parameters, employed also in the parametric approaches to bias specification, see (De Raedt, 1992), (Muggleton & Feng, 1990), (Buntine, 1987). These parameters will be systematically varied in the experiments of Section 5.

Definition 3. The depth $d(V)$ of a variable V is 0. The depth $d(c)$ of a constant c is 1. The depth $d(f(t_1, \dots, t_n))$ of a term $f(t_1, \dots, t_n)$ is $1 + \max \{d(t_1), \dots, d(t_n)\}$. The depth of a clause is the maximum of the depth of its terms.

Limiting the depth of terms in hypotheses to 1 corresponds to working with functor-free clauses.

Definition 4. The level $l(t)$ of a term t in a clause c is 0 if t occurs as an argument in the head of c ; and $1 + \min\{l(s) \mid s \text{ is an argument of a literal in } c \text{ which also has } t \text{ as an argument}\}$. The level of a clause is the maximum of the level of its terms.

Definition 5. A clause is linked if the level is defined for all arguments in all of its literals.

Note that we consider only linked clauses in this paper.

Example 4 *The variable F in father(F) $\leftarrow$ male(F), parent(F,C) has level 0, the variable C has level 1, the variable G in grandfather(F) $\leftarrow$ male(F), parent(F,C), parent(C,G) has level 2, etc.* $\square$

The level of a term corresponds to Muggleton and Feng's i parameter (Muggleton & Feng, 1990) and De Raedt's level of existential quantification (De Raedt, 1992). The level and the depth are especially important in the context of specific-to-general ILP systems such as ITOU (Rouveirol & Puget, 1989), (Rouveirol, 1991), GOLEM (Muggleton & Feng, 1990), CLINT (De Raedt, 1992) and PGA (Buntine, 1987), because this class of learners starts learning from so-called starting clauses, i.e., maximally specific clauses covering the example with regard to the background theory, cf. Section 5.1.

3.2. Semantic bias

Whereas syntactic bias imposes restrictions on the *form* or *syntax* of hypotheses, semantic bias imposes restrictions on the *behaviour* of the hypotheses. Various forms of semantic bias have been considered in inductive logic programming, including mode and type declarations, number of covered examples (Muggleton & Feng, 1990), De Raedt et al., 1993), determinacy restrictions (Muggleton & Feng, 1990), (Quinlan, 1991), significance tests, and encoding length heuristics (Quinlan, 1991). In this paper, we focus on the determinacy restriction because it is widely employed in and characteristic of inductive logic programming, see for instance Muggleton & Feng, 1990, Quinlan, 1991 and Lavrač & Džeroski, 1994. Furthermore, type and mode restrictions are well understood and can also be enforced syntactically, cf. Lavrač & Džeroski, 1994 and Pazzani & Kibler, 1992. The other restrictions are not characteristic of inductive logic programming.

Below we define the determinacy restriction, adopting the simplified definition of Džeroski et al, 1992 instead of the original one by Muggleton & Feng, 1990.

Definition 6. (adapted from Džeroski et al, 1992) A definite clause $h \leftarrow l_1, \dots, l_n$ is determinate (with respect to background theory T and factual examples E) if and only if for every substitution θ for h that unifies h to a ground instance $e \in E$, and for all $i = 1, \dots, n$ there is a unique substitution θ_i such that $(l_1 \wedge \dots \wedge l_i)\theta\theta_i$ is both ground and $T \models (l_1 \wedge \dots \wedge l_i)\theta\theta_i$.

Example 5 *Consider the following background theory T :*

$T = \{\text{parent(jef,paul)} \leftarrow; \text{parent(jef,ann)} \leftarrow; \text{male(paul)} \leftarrow; \text{female(ann)} \leftarrow\}$

The clause $\text{has-father}(Y) \leftarrow \text{parent}(F, Y)$ is determinate since given a Y there is a unique instantiation of F that is true. On the other hand, the clause $\text{is-father}(F) \leftarrow \text{parent}(F, Y)$ is not determinate since there exist two instantiations of Y given F . $\square$

In the GOLEM system this determinacy restriction is enhanced with a parameter j , called the degree of determinacy.

Definition 7. A clause is j -determinate if it is determinate, and if the maximal degree of determinacy of its body literals is j .

A literal in a clause has degree j of determinacy if the number of different variables occurring in the literal, that have occurrences to the left of the literal, is j .

By fixing the value of the parameter j one can impose an upper bound on the degree of determinacy of clauses.

Example 6 The clause $\text{mult}(A, B, C) \leftarrow \text{successor}(B', B), \text{mult}(A, B', C'), \text{plus}(A, C', C)$ is determinate. The literal $\text{successor}(B', B)$ is determinate with degree 1, and the literals $\text{plus}(A, C', C)$, $\text{mult}(A, B', C)$ are determinate with degree 2. So the clause is 2-determinate. $\square$

Determinate clauses are one way to get around some of the problems in syntactic languages that can have large starting clauses (cf. below), or more than one starting clause for one example. Indeed, some of the results in computational learning theory show that certain classes of determinate clauses can be learned efficiently (cf. Džeroski et al, 1992). This, however, comes at the cost of losing completeness (cf. also Section 5.3).

3.3. Shift of bias

One advantage of specifying the bias explicitly is that it becomes a tunable and portable parameter of inductive systems. Moreover, systems can reason about their biases at the meta-level, cf. (Russell & Grosz, 1990), (Russell, 1989), (Utgoff & Mitchell, 1982), (Utgoff, 1986), (De Raedt, 1992). This is especially useful for shifting the learning bias. Within our framework for bias specification, shifting the bias is the process whereby the system is given a series of semantic and syntactic biases $(\mathcal{L}_0, \mathcal{B}_0), \dots, (\mathcal{L}_n, \mathcal{B}_n)$ and can detect when the current bias $(\mathcal{L}_i, \mathcal{B}_i)$ is insufficient for the learning task. Shifting the bias then involves trying the next bias $(\mathcal{L}_{i+1}, \mathcal{B}_{i+1})$ in the learner. Although advanced approaches might reason about the cause of the learning failure and dynamically select the next bias for learning, our NINA system assumes a predefined sequence of biases as De Raedt's CLINT system, cf. Section 5.1.

4. A generic specific-to-general ILP learner : NINA

In this section we present the NINA system. As already mentioned in the introduction, NINA is a generic framework, containing a number of parameters that must be instanti-

ated in order to obtain a real algorithm. The most important parameters are the syntactic and the semantic criteria discussed in Section 3.1 and 3.2. The main advantages of separating those criteria and specifying them declaratively is that this enables us to shift the bias of the system (see Section 3.3) and to experimentally compare the frequently employed biases of GOLEM, ITOU and CLINT (see Section 5).

4.1. The top level algorithm

The algorithm in Table 1 describes the overall NINA algorithm.

At the top-level NINA abstracts the way in which specific-to-general ILP-learners proceed in general. Given are a background theory T , positive evidence P and negative evidence N , and an initially empty hypothesis H . Note that in NINA we use the above described generalized notions of positive and negative evidence.

The system handles the elements of P incrementally, splitting up the treatment of one element into two phases.

Table 1. The Generic Algorithm NINA

```

procedure NINA( $\mathcal{L}, \mathcal{B}, P, N, T$ )
   $H := \emptyset$ 
  for all  $p \in P$ 
    if  $T \cup H \not\models p$ 
      then  $c := \text{starting\_clause}(p, T \cup H, \mathcal{L}, \mathcal{B})$ 
        if  $T \cup H \cup \{c\} \cup N \not\models \square$ 
          then  $H := \text{generalize}(c, H, T, P, N)$ 
        endif
      endif
    endif
  endfor
  post\_process  $H$ 
  output  $H$ 
endproc

```

The first phase (see the starting clause generation in Section 4.2) is a constructive induction step. Taking into account the syntactic and semantic restrictions, the system uses the available background knowledge in order to compute the starting clause,¹ i.e., to replace the example by a most specific description, that implies the given example, and that is consistent with the negative evidence. Of course, if the positive example is already covered by the current hypothesis, the system simply proceeds with the next example.

In the second step (see Section 4.3), the generalization operator takes the computed starting clause and the current hypothesis, and generalizes them into a new hypothesis that covers all previously handled examples, and the new example. For our bias evaluation

purposes, this generalization step is less important than the starting clause generation. It is basically a simplification of the GOLEM strategy (Muggleton & Feng, 1990).

Notice also that – in contrast to systems such as FOIL (Quinlan, 1990) and GOLEM (Muggleton & Feng, 1990) – NINA performs correct (intensional) coverage tests. The problems with extensional systems, i.e., systems using a ground model as background theory, are well-known (see for instance (Bergadano & Gunetti, 1993), (De Raedt et al., 1993)). Problems arise because the test for coverage may yield incorrect results. To test whether a clause $p(X) \leftarrow q(X)$ covers an example $p(e)$, extensional systems would check whether $q(e)$ is in the model. This may lead to problems when learning recursive predicates or multiple predicates, as discussed in (De Raedt et al., 1993), (Bergadano & Gunetti, 1993) and (Cameron-Jones & Quinlan, 1993).

Finally the hypothesis is post-processed by removing redundant literals in the clauses² and removing redundant clauses in the hypothesis³.

4.2. Computing starting clauses

In order to implement bias as a parameter, we introduce here the notion of a *starting clause*. As the expression itself indicates, this is a clause that forms a starting point for the learning system. For specific-to-general learning systems, such a clause has to be a lower bound in the space of possible hypothesis clauses, with respect to the *is more general than* relation.

More formally, within the settings presented in Section 2 and Section 3, a starting clause can be defined as follows:

Definition 8. A starting clause $sc \in \text{starting_clause}(e, T, \mathcal{L}, \mathcal{B})$ with respect to a (positive) example e , a theory T , a language $\mathcal{L}$ and a semantic bias $\mathcal{B}$ is a maximally specific clause sc in $\mathcal{L}$ such that $T \cup \{sc\} \models e$ and $\mathcal{B}(sc, T, P, N)$ is true.

According to this definition, a starting clause for a positive example e is thus a most specific clause within the syntactic bias, that satisfies the semantic bias, and that covers the example e given the background theory T .

Starting clauses need not be unique, as shown below:

Example 7 Consider the following background theory T , and language $\mathcal{L}$:

$$T = \{\text{parent}(\text{jef}, \text{paul}) \leftarrow; \text{parent}(\text{jef}, \text{ann}) \leftarrow; \text{male}(\text{paul}) \leftarrow; \text{female}(\text{ann}) \leftarrow\}$$

$$\mathcal{L} = \{\text{is-father-of-son}(X) \leftarrow \text{parent}(X, Y), \{\text{male}(Y), \text{female}(Y)\}\}$$

$\mathcal{L}$ is thus a language of depth 0 (only variables are allowed as arguments) and level 1.

Now the starting clauses for $\text{is-father-of-son}(\text{jef})$ are:

is-father-of-son(X) $\leftarrow$ parent(X, Y), male(Y);
 is-father-of-son(X) $\leftarrow$ parent(X, Y), female(Y). $\square$

Before presenting the algorithm in Table 2, we introduce a notion of a refinement operator (cf. (Shapiro, 1983))⁴:

Definition 9. A refinement operator ρ for a language $\mathcal{L}$ is a function from $\mathcal{L}$ to $2^{\mathcal{L}}$ such that for all $c \in \mathcal{L}$: $\rho(c) = \{c' \mid c' \text{ is a proper maximally general specialization of } c \text{ in } \mathcal{L}\}$.

Notice that according to this definition, the refinement operator is uniquely determined by the syntactic bias and the notion of generality employed (in our case this is θ -subsumption).

The algorithm in Table 2 shows how refinement operators can be used to compute $\text{starting_clause}(e, T, \mathcal{L}, \mathcal{B})$. The idea is to initialize the starting clause with a variabilized version of e , and then repeatedly apply the refinement operator ρ until no further refinement can be found that satisfies the bias $\mathcal{B}(c, T, P, N)$ and that covers e . In this way a most specific clause is found that satisfies the syntactic and the semantic bias, and that covers the given evidence e .

Table 2. The function $\text{starting_clause}(e, T, \mathcal{L}, \mathcal{B})$ computes a most specific clause that satisfies the syntactic and the semantic bias, and that covers the given evidence e .

```

function starting_clause( $e, T, \mathcal{L}, \mathcal{B}$ )
   $p := e$  where all constants are replaced by variables5
   $sc := p$ 
  if  $\mathcal{B}(sc, T, P, N)$ 
  then
    while  $\exists c \in \rho(sc) : \mathcal{B}(c, T, P, N)$  and  $T \cup \{c\} \models e$ 
    do  $sc := c$ 
    endwhile
    return  $sc$ 
  else return fail
  endif
endfunction

```

In the remainder of this section, we present some properties of this algorithm. This part of the paper may be skipped without problems by the casual reader.

The properties rely on a notion of *monotonic* semantic bias.

Definition 10. A semantic bias $\mathcal{B}$ is monotonic with regard to $\mathcal{L}$ if and only if $\forall c \in \mathcal{L}, n, sc \in \rho^n(c), P, N : \mathcal{B}(sc, T, P, N) \rightarrow \exists c' \in \rho^{n-1}(c)$ such that $\mathcal{B}(c', T, P, N)$.

Roughly speaking, monotonicity requires that if one starts from positive evidence c and computes a starting clause sc from c , there must exist a path (in the refinement graph)

from c to sc for which all intermediate clauses satisfy the semantic bias. Consider for instance a semantic bias enforcing a minimum number of covered positive examples. Such a bias is clearly monotonic because if a clause satisfies this restriction, all generalizations will also satisfy this restriction. Similarly, one can prove that determinacy is monotonic (although not all generalizations need to be determinate).

Claim 1 *The algorithm for the function $\text{starting_clause}(e, T, \mathcal{L}, \mathcal{B})$ is sound and complete if the semantic bias is monotonic and if the initial starting clause p satisfies the semantic bias.*

Soundness means that clauses generated by the algorithm will be true starting clauses, i.e., they will satisfy Definition 8; completeness means that all starting clauses will be generated.

Claim 2 *If $e \in \mathcal{L}$ then all clauses $sc \in \text{starting_clause}(e, T, \mathcal{L}, \mathcal{B})$ will be logically equivalent to e with regard to T , i.e., $T \models (sc \leftrightarrow e)$.*

Under the conditions of this claim, generated starting clauses are always consistent with the negative evidence (unless the evidence itself is inconsistent) and therefore the (syntactic) bias will never be shifted. This is the case in the original GOLEM and ITOU. In NINA, starting clauses are always of depth 0 (i.e., all arguments are variables). So $e \notin \mathcal{L}$, and shifting the bias is a useful operation.

4.2.1. Starting clauses in a series of languages

Suppose that the system is given a series of biases (each composed of a syntactic and a semantic bias). When building a starting clause for a given example, NINA starts using the first bias in the series. When unable to construct a consistent starting clause, NINA shifts its bias to the next bias in the series. This process of computing starting clauses and shifting the bias is repeated until a consistent starting clause is found, or all biases are tried.

Enhancing the computation of starting clauses with this feature results in the algorithm of Table 3.

In the algorithm in Table 1 the call to the function *starting_clause* should then be replaced by a call to the function *starting_clause_with_shift_of_bias*.

The final algorithm is able to handle languages where more than one starting clause can be generated. For this purpose backtracking is introduced.

4.3. The generalization procedure

The generalization function is shown in Table 4.

Since most specific-to-general ILP-learners use an *lgg*-like operator, we based the generalization operator in NINA on Plotkin's framework (Plotkin, 1970), (Plotkin, 1971). Our system loops over the clauses in the current hypothesis, each time computing the

Table 3. The function `starting_clause_with_shift_of_bias` ($p, T, (\mathcal{L}_0, \mathcal{B}_0), \dots, (\mathcal{L}_m, \mathcal{B}_m)$) computes a starting clause for the given evidence e , within a predefined series of language biases.

```

function starting_clause_with_shift_of_bias ( $p, T, (\mathcal{L}_0, \mathcal{B}_0), \dots, (\mathcal{L}_m, \mathcal{B}_m)$ )
   $i := 0$   $\{i \text{ denotes the bias index}\}$ 
  repeat
     $sc := \text{starting\_clause}(p, T, \mathcal{L}_i, \mathcal{B}_i)$ 
    if  $T \cup H \cup \{sc\} \cup N \models \square$ 
      then if alternative starting clauses exist
        then backtrack on  $sc$ 
      else  $i := i + 1$ 
      endif
    endif
  until  $T \cup H \cup \{sc\} \cup N \not\models \square$  or  $i = m + 1$ 
  if  $i \leq m$ 
    then return  $sc$ 
  else return fail
  endif
endfunction

```

Table 4. The generalization algorithm.

```

function generalize( $c, H, T, P, N$ )
  if  $\exists c' \in H: T \cup H \cup lgg(c, c') \cup N \not\models \square$ 
    and  $\mathcal{B}(T, H \cup lgg(c, c'), P, N)$ 
  then output  $H - \{c'\} \cup lgg(c, c')$ 
  else output  $H \cup \{c\}$ 
  endif
endfunction

```

least general generalization of the starting clause and a clause of the hypothesis, until a consistent *lgg* is found, or until all clauses in the hypothesis are tried. If NINA finds a consistent *lgg*, the corresponding clause in the hypothesis is replaced by the *lgg*. If not, the starting clause is added to the hypothesis.

4.4. An example

In this section, we demonstrate NINA on a simple example.

Given are:

Figure 1. The family tree : jef, jan, piet, bart, joost and hans are male; mia, an, els, miet, lieve, inge and tine are female; and & denotes the relation married/2. Partners of children are in italic.

- T: background knowledge containing the relations *female/1*, *male/1*, *minor/1* and *parent/2* defined on the family presented in Figure 1. inge, lieve, hans and tine are minor; jef, jan, bart, piet, joost and hans are male; and the others are female.
- $\{(\mathcal{L}_0, \mathcal{B}), (\mathcal{L}_1, \mathcal{B})\}$: a series of biases, where
 - $\mathcal{L}_0 = \{\text{has-a-son}(X) \leftarrow \{\text{male}(X), \text{female}(X), \text{minor}(X), \text{parent}(X,X)\}\}$
 i.e., $\mathcal{L}_0$ is the language of completely bound DATALOG Horn clauses⁶;
 - $\mathcal{L}_1 = \{\text{has-a-son}(X) \leftarrow \{\text{male}(X), \text{female}(X), \text{minor}(X), \text{parent}(X,X), \text{parent}(X,Y), \text{parent}(Z,X), \text{parent}(Y,Z), \text{parent}(Z,Y), \text{parent}(Y,Y), \text{parent}(Z,Z), \text{female}(Y), \text{male}(Y), \text{female}(Z), \text{male}(Z), \text{minor}(Y), \text{minor}(Z)\}\}$
 i.e., $\mathcal{L}_1$ is nearly the language of DATALOG Horn clauses with maximum level 1⁷;
 - $\mathcal{B} = \text{true}$, i.e., there are no semantic restrictions.
- Positive evidence: $P = \{\text{has-a-son}(\text{bart}) \leftarrow; \text{has-a-son}(X) \leftarrow \text{female}(X), \text{parent}(X,Y) \text{ male}(Y); \text{has-a-son}(\text{jef}) \leftarrow\}$
- Negative evidence: $N = \{\leftarrow \text{has-a-son}(\text{piet}); \leftarrow \text{has-a-son}(X), \text{minor}(X) \}$

Below we give a trace of NINA. Comments are given in italics.

```
?- nina.
Handling the example has-a-son(bart) ...
Found the following starting clause in bias  $(\mathcal{L}_0, \mathcal{B})$  :
has-a-son(X) :- male(X).
```


Since there are no alternative starting clauses in $(\mathcal{L}_0, \mathcal{B})$, the system shifts its bias.

$has-a-son(X) :- male(X), parent(X,Y), parent(Z,X), minor(Y), male(Y), female(Z)$
 $has-a-son(X) :- male(X), parent(X,Y), parent(Z,X), minor(Y), female(Y), male(Z)$
 $has-a-son(X) :- male(X), parent(X,Y), parent(Z,X), minor(Y), male(Y), male(Z)$

Because the hypothesis H is empty, no (consistent) lgg can be computed. Therefore the starting clause is added to H .

The system checks this definite clause, and finds out that it belongs to the language $\mathcal{L}_1$. Next it tries to add literals to the clause, that are in $\mathcal{L}_1$, and that cover all the instances covered by the given clause. Given the original knowledge base, no literal can be added. However, if the clause $\text{mother}(X,Y) \leftarrow \text{female}(X), \text{parent}(X,Y)$ were present in the original knowledge base, the literal $\text{mother}(X,Y)$ could be added to complete the example.

Found the following starting clause in $(\mathcal{L}_1, \mathcal{B})$:

```
has-a-son(X) ← female(X), parent(X,Y), male(Y)
```

... checking the consistency ...

It is consistent !

... generalizing the hypothesis and the starting clause ...

Found a consistent lgg:

```
has-a-son(X) :- parent(X,Y), male(Y)
```

Handling the example `has-a-son(jef)` ...

It is already covered by the current hypothesis.

No more positive evidence.

... removing redundancies ...

The following definition was learned :

```
has-a-son(X) :- parent(X,Y) , male(Y)
```

□

5. Bias evaluation

In this section, we aim at establishing a three-way relation between the background knowledge, the bias and the performance of the learner, which is measured in terms of accuracy as well as efficiency. In order to achieve this aim, we performed experiments with NINA with a number of biases that are typically employed in specific-to-general systems: the shift of bias as used in CLINT (De Raedt, 1992), the determinacy restriction of GOLEM (Muggleton & Feng, 1990) and the use of definite clauses as positive examples as in ITOU (Rouveirol, 1991). Tuning NINA's bias parameters allowed us to *emulate* the syntactic and semantic biases of these three systems. It is important to realize that it was never our intention to reproduce the same learning behaviour as the original systems. Rather we focussed in on the biases they employ. Furthermore, the emulated versions, incorporating NINA's general framework, have a number of abilities that go beyond the original systems. One important difference between ITOU and GOLEM and their emulations is that we restrict the languages to depth 0, i.e., only variables are allowed as arguments in the clauses. In the original GOLEM and ITOU systems, constant and functor symbols are allowed in the hypothesis language. The motivations for this assumption are that it makes starting clauses in NINA not necessarily logically equivalent to the positive examples for which they are derived (cf. also Claim 2), and that it makes a bias shift relevant. Finally, at the procedural level, the NINA emulations also differ from the original systems. E.g. CLINT works interactively and generates its own examples. GOLEM uses extensional coverage tests, whereas all tests for coverage in NINA are done intensionally. However, these differences are not important for us as we focus on bias here.

The emulations will also clearly demonstrate the generality and power of the NINA approach as a general framework for specific-to-general learning and for comparing language bias. In a sense, NINA is to specific-to-general learning what Cohen's GRENDL (Cohen, 1994) is for general-to-specific learning.

Table 5. Biases in NINA.

Syntactic Bias	Originally in :	Difference
Series of languages with increasing level of variables	GOLEM	In GOLEM the level (the i -parameter) is fixed
Series of languages with increasing level of variables and extra restrictions on the new variables	CLINT	
Language dependent on the positive evidence	ITOU	
Semantic Bias	Originally in :	Difference
No extra restriction	CLINT, ITOU	
Series of j -determinate languages with variable j	GOLEM	In GOLEM the j -parameter is fixed

5.1. Typical biases in specific-to-general ILP learners

We analyzed the biases of three typical ILP learning systems: CLINT, GOLEM and ITOU. Table 5 summarizes the biases we incorporated in our NINA-system. Each of these features is explained in more detail in the discussion of the biases of the systems they originated from.

5.1.1. The bias of CLINT

In the CLINT system of De Raedt and Bruynooghe, the syntactic criteria are explicitly formulated by defining a series of parameterized concept description languages, ordered according to growing expressiveness. Typically the languages have a depth of 0. By varying the level, one obtains a series. Several series of languages are defined in (De Raedt, 1992), each varying different parameters. As a full presentation of these alternatives would lead us too far, we illustrate here the series recommended to users of CLINT. We do not go into further detail on the actual syntactic properties of these languages, since they are less important than the fact that there is a *series* of languages that is ordered according to growing expressiveness.

Example 8 Suppose the arity of the target predicate is 2, and the predicates in the background theory are *male*/1, *female*/1, *parent*/2. Then CLINT's $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2$ are:

$$\mathcal{L}_0 = \{ P(X,Y) \leftarrow \{ \text{male}(\{X,Y\}), \text{female}(\{X,Y\}), \text{parent}(\{X,Y\},\{X,Y\}) \} \}$$

$\mathcal{L}_0$ is the language of completely bound functor free Horn clauses.

$$\mathcal{L}_1 = \{ P(X,Y) \leftarrow \{ \text{male}(\{X,Y\}), \text{female}(\{X,Y\}), \text{parent}(\{X,Y\},\{X,Y\}), \text{parent}(X,Z_1), \}$$

$\text{parent}(Y, Z_2), \text{parent}(Z_3, X), \text{parent}(Z_4, Y)\}$

$\mathcal{L}_1$ allows for the introduction of variables at level 1, with the extra restriction that each literal can contain at most one such variable, and that each such variable can only appear once in the clause. Note that it is due to this last restriction that no male- or female-literals concerning the Z_i are allowed.

$\mathcal{L}_2 = \{P(X, Y) \leftarrow \{\text{male}(\{X, Y\}), \text{female}(\{X, Y\}), \text{parent}(\{X, Y\}, \{X, Y\}), \text{parent}(X, Z_1), \text{parent}(Y, Z_2), \text{parent}(Z_3, X), \text{parent}(Z_4, Y), \{X, Y, Z_1, Z_2, Z_3, Z_4\} = \{X, Y, Z_1, Z_2, Z_3, Z_4\})\}$

$\mathcal{L}_2$ imposes the same restrictions as $\mathcal{L}_1$, however there is an $=/2$ predicate, which can unify existing variables, but cannot introduce new variables. This explicit use of a unification predicate is due to the actual implementation of the CLINT system. $\square$

No restrictions other than syntactic ones are employed by CLINT, i.e., $\mathcal{B}(H, T, P, N)$ is always true.

5.1.2. The bias of GOLEM

The GOLEM system of Muggleton and Feng uses the notion of ij -determination, which includes both syntactic and semantic criteria. The syntactic restriction is basically the i -part of GOLEM. It corresponds to the level parameter introduced in Definition 3.1.2. Originally GOLEM allowed clauses up to any depth. In the emulation of GOLEM's bias with NINA however, we restrict the syntactic bias to depth 0. The reason is two-fold: with infinite depth, a shift of bias is useless (cf. the properties of Algorithm 3), and for ease of comparison with CLINT's bias. The semantic restriction corresponds to j -determinacy (Definition 3.2 and 3.2), i.e., $\mathcal{B}(c, T, P, N)$ is true if c is determinate with degree at most j in T . The syntactic restrictions are illustrated in Example 9.

Example 9 Suppose the predicate to be learned is of arity 2, and the predicates in the background theory are male/1, female/1, parent/2. Then the emulated GOLEM's $\mathcal{L}_0, \mathcal{L}_1$ are :

$\mathcal{L}_0 = P(X, Y) \leftarrow \{\text{male}(\{X, Y\}), \text{female}(\{X, Y\}), \text{parent}(\{X, Y\}, \{X, Y\})\}$

$\mathcal{L}_0$ is the language of completely bound functor free Horn clauses (as in CLINT).

$\mathcal{L}_1 = P(X, Y) \leftarrow \{\text{male}(\{X, Y\}), \text{female}(\{X, Y\}), \text{parent}(\{X, Y\}, \{X, Y\}), \text{parent}(X, Z_1), \text{parent}(Y, Z_2), \text{parent}(Z_3, X), \text{parent}(Z_4, Y), \text{male}(\{Z_1, Z_2, Z_3, Z_4\}), \text{female}(\{Z_1, Z_2, Z_3, Z_4\})\}$

$\mathcal{L}_1$ allows for new variables in the body of clauses up to level 1.

In general $\mathcal{L}_i$ allows for clauses with depth 0 and maximum level i . $\square$

5.1.3. The bias of ITOU

For Rouveirol's ITOU system, it is harder to clearly separate the syntactic and semantic criteria. One of the main differences between ITOU and the previous systems is that

ITOU accepts (as NINA) definite clauses as positive evidence. ITOU completes the body with literals logically entailed by the flattened example and the flattened background theory, i.e., $\mathcal{B}(c, T, P, N)$ is true if the body of c is logically entailed by the body of the flattened initial definite clause. Because we do not want to go into details about flattening, we reformulate this bias into NINA, while assuming range-restricted clauses⁸. In ITOU the language $\mathcal{L}$ depends on the positive example. In particular, ITOU is not able to introduce arguments in the starting clause which are not present in the positive example.

As for GOLEM's and CLINT's emulations, we assume the depth to be 0. Therefore, in NINA's emulation of ITOU, positive examples that are ground facts always have starting clauses in the language of completely bound clauses. However, as indicated in Example 1 and discussed in Section 5.3, this restriction can be overcome by providing definite clauses with a non-empty body as positive evidence, rather than with an empty body. The terms introduced in the body of the definite clause help ITOU to introduce relations on terms that are not in the head.

The semantic bias for ITOU can be considered to be always true. ITOU is illustrated in the next example.

Example 10 *Suppose the predicate to be learned is of arity 2, and the predicates in the background theory are male/1, female/1, parent/2. Suppose the given example is $\text{brother}(\text{jeff}, \text{paul}) \leftarrow \text{parent}(\text{ann}, \text{jeff})$. Assuming depth = 0, we then have as syntactic bias:*

$$\mathcal{L} = P(X, Y) \leftarrow \{ \text{male}(\{X, Y, Z\}), \text{female}(\{X, Y, Z\}), \text{parent}(\{X, Y, Z\}, \{X, Y, Z\}) \}$$

$\mathcal{L}$ is obtained by first flattening the original clause, which yields $\text{brother}(X, Y) \leftarrow \text{parent}(Z, Y), \text{jeff}(X), \text{paul}(Y), \text{ann}(Z)$. Applying saturation on the flattened clause (corresponding to the example), ITOU can never introduce new variables. As the only predicates are male/1, female/1, and parent/2, the only literals that can be introduced by ITOU for this example are those occurring in the language $\mathcal{L}$ above. Note that in the original ITOU system after saturation the example is unflattened by folding on the (temporary) predicates jeff/1, paul/1 and ann/1. In our emulated version, we just drop the literals jeff(X), paul(Y) and ann(Z) since we require clauses to be of depth 0.

The language used thus depends on the example. Since there are three terms in the example, we have three variables in the clauses of $\mathcal{L}$. $\square$

The emulation of ITOU appears to be somewhat unconventional in the sense that the semantic restrictions of ITOU (imposed by the flattening and completion operators) can be described in our framework as syntactic restrictions.

5.2. Important issues in bias

In this section we identify a number of important questions concerning different syntactic and semantic biases for inductive logic programming. As all of these questions are

complex, we do not expect to formulate a full and definite answer to these questions. Nevertheless, our experiments indicate some plausible answers to them.

Question 1 *What is the influence of the semantic restriction of determinacy?*

Given the success of the inductive logic programming systems GOLEM and FOIL (Quinlan, 1990), which both employ the determinacy restriction and which are both very efficient, it would be interesting to know whether it is always justified to impose the determinacy restriction, and if not, when it should be avoided. From another perspective it would be interesting to find out what the influence of determinacy is on the computational resources needed for learning.

Question 2 *What is the use of the shift of bias?*

When shifting the bias with regard to a fixed sequence of biases which are ordered according to growing expressiveness, one may wonder whether it is easier (or more efficient) to learn immediately in the largest language without a need for shifting the bias.

Question 3 *How do non-factual examples, i.e., denials and definite clauses, compare with factual examples?*

In other words, is the more expressive form of evidence (as in NINA and ITOU) useful? Does it indeed reduce the number of needed examples, and what is the tradeoff for this?

Question 4 *What is the influence of the background theory on the learning result?*

Within inductive logic programming and other forms of concept learning, researchers have always argued that background theory is very important. Usually the effect of the background theory is evaluated in terms of accuracy. From a bias perspective it is also useful to measure its effect on the efficiency of the learner. In particular, in inductive logic programming, what is the influence of the number of predicates on the resources consumed by the learner?

5.3. *Experimental results*

In this section we describe a number of experiments to provide insight into the above questions. We applied our system NINA on two different domains: the first one is a relatively small domain, involving family relations. The second domain is a larger one, namely that of "Finite Element Mesh Design" as described in Dolšák, 1991.

5.3.1. *Learning Family Relations*

Although family relations is a toy domain, it is sufficiently complex to contribute to an improved understanding of bias. First, the background knowledge can contain many

relations, not all of which are relevant for a certain learning problem. Second, some relations that are very well understood by people turn out to be surprisingly complex. Consider for example the relation `uncle/2`. The exact definition consists of 4 clauses, containing variables up to depth 2, as shown in Table 6.

Table 6. Definition of `uncle/2`.

<code>uncle(x,y)</code>	$\leftarrow$	<code>brother(x,z), father(z,y)</code>
<code>uncle(x,y)</code>	$\leftarrow$	<code>brother(x,z), mother(z,y)</code>
<code>uncle(x,y)</code>	$\leftarrow$	<code>married(x,z), sister(z,t), father(t,y)</code>
<code>uncle(x,y)</code>	$\leftarrow$	<code>married(x,z), sister(z,t), mother(t,y)</code>

Finally, the way the knowledge is expressed will influence the learning result. For example the result of learning `grandparent/2` will be different when the parent-relation is expressed by `parent/2` or by `father/2` and `mother/2`.

Figure 1 (see Section 4.4) contains the family tree that is used in our experiments.

In the experiments we performed in this domain, we concentrated on learning the following four relations : `brother/2`, `grandparent/2`, `sibling/2` and `parent_in_law/2`. Table 7 gives the number of positive and negative instances for each of these relations in the given family.

In tables 9 and 12 that present results of our experiments with NINA on this domain, the following parameter instantiations are listed for each run:

BT (Relations in the background theory): the predicates available in the background theory, that can be used by the system in order to build starting clauses.

P (Positive evidence): the selected positive instances and/or the definite clauses that are given to the system. P:all means that all positive instances are used.

N (Negative evidence) : CWA means closed world assumption, i.e., the use of all possible negative instances as negative evidence; in case denials are used, they are explicitly mentioned.

Bias : we use three different combinations of biases:

1. a series $\{(\mathcal{L}_0, \mathcal{B}), (\mathcal{L}_1, \mathcal{B}), (\mathcal{L}_2, \mathcal{B})\}$, with the syntactic language bias as in Example 8, and the semantic bias $\mathcal{B}(c, T, P, N)$ always true. In the table this bias is referred to as CLINT.

Table 7. Size of the example sets in the family relations domain

	<code>brother/2</code>	<code>grandparent/2</code>	<code>sibling/2</code>	<code>parent_in_law/2</code>
number of positive instances	5	14	10	6
number of negative instances	164	155	159	163

2. a series $\{(\mathcal{L}_0, \mathcal{B}), (\mathcal{L}_1, \mathcal{B}), (\mathcal{L}_2, \mathcal{B})\}$, where the subscript of the syntactic languages denotes the maximal level of its clauses (as in Example 8), and the semantic bias $\mathcal{B}(c, T, P, N)$ is true if the clause c is determinate with degree at most 1. This bias is referred to as GOLEM in the table.
3. the bias of ITOU, i.e., requiring logical entailment from the flattened positive evidence and the flattened background theory. So the syntax of the hypothesis clauses is determined by the syntax of the positive evidence and the background theory (cf. Example 10).

Tables 9 and 12 list the following results :

sc-literals: the number of literals in the starting clauses. For each bias we list all the numbers that occurred in our experiments. E.g., $\mathcal{L}_2:12-13-14$, means that when searching in the language $\mathcal{L}_2$, NINA found starting clauses with 12, 13 or 14 literals (where different numbers of course concern different examples).

H-clauses: the number of clauses in the hypothesis (after post-processing).

Time : expressed in cpu-seconds on a Sparc Sun4. We split the result in two parts: the time needed for inducing a hypothesis (l.), and the time needed for reducing (post-processing) this hypothesis (r.).

Accuracy: the number of correctly and incorrectly classified positive and negative instances. **cp** is the number of covered positives, **up** the number of uncovered positives, **cn** the number of covered negatives and **uc** the number of uncovered negatives.

Finally, *fails* in the table means that the system was unable to construct consistent starting clauses within the given bias settings.

5.3.2. Learning Finite Element Mesh Design

As a larger domain we chose the “Finite Element Mesh Design”-domain, as described in Dolšák, 1991 and Dolšák and Muggleton, 1992. We will not go into detail about finite element (FE) methods. We just mention here that the problem involves the partitioning of physical structures into a finite number of elements for the purpose of analyzing stresses. For deciding how to partition a structure a number of parameters including the shape of the structure, the loadings and the boundary conditions should be taken into account.

Table 8 describes the FE domain. Positive examples have the form `mesh(Edge, Number)`, meaning that a particular `Edge` is partitioned in `Number` parts. Example : `mesh(a23,2)`. The set of possible numbers of finite elements is $\{1,2,3,4,5,7,8,9,10,11,12\}$, and for the negative examples the closed world assumption is made, i.e., $N = \{ \text{mesh(Edge, Number)} \mid \text{Edge} \in \{a1, \dots, e96\}, \text{Number} \in \{1, \dots, 12\} \} - P$, where P is the set of positive examples.

We first describe the general setup of the experiments and list the results. We divided the task of learning rules for the predicate `mesh/2`, into the subtasks of learning rules

Table 8. The Finite Element Mesh Domain.

Structures	a (a cylinder) b (a hydro) c (a paper mill) d (a roller) e (a bearing box)
Labeling of the edges	a1,a2,...,a55 b1,b2,...,b42 c1,c2,...,c28 d1,d2,...,d57 e1,e2,...,e96
Background predicates	
structure	short/1, circuit/1, half_circuit/1, quarter_circuit/1, short_for_hole/1 long_for_hole/1, circuit_hole/1, half_circuit_hole/1, not_important/1
loadings	not_loaded/1, one_side_loaded/1, two_side_loaded/1, cont_loaded/1
boundary conditions	free/1, one_side_fixed/1, two_side_fixed/1, fixed/1 neighbour/2, opposite/2, equal/2

for mesh(.,1), mesh(.,2), mesh(.,3), mesh(.,4), mesh(.,5), mesh(.,6), mesh(.,7), mesh(.,8), mesh(.,9), mesh(.,10), mesh(.,11) and mesh(.,12).

In order to prevent the system from just stopping when it does not succeed in constructing a consistent starting clause for a certain example, we make the system add the example to the hypothesis. By doing so, we obtain a hypothesis containing a number of clauses, and a number of ground facts. This allows us to determine the number of examples covered by the hypothesis and the number of examples for which the system is not able to find a starting clause, and thus also no covering hypothesis. (This can be due to an overly-restrictive bias, or to noise present in the data.)

The application of GOLEM to this domain is described in Dolšák & Muggleton, 1992 and Dolšák, 1991. However, to obtain more determinate clauses, the background knowledge was adapted. Dolšák et al. added suffixes *_xy*, *_yz* and *_zx* to the predicate *neighbour/2*, and suffixes *_r* and *_l* to the predicates *equal/2*, *opposite/2* and to the extended *neighbour* predicates. However, using the learned rules for deciding on the FE-mesh design of new structures, these suffixes are not needed. FE methods apparently do not need the division into three planes in the xyz-space, nor do they need the symmetry information contained in the *_r*- and *_l*-suffixes. Therefore we removed these suffixes, and removed also all the information doubly contained in the background knowledge.

In the experiments, the structures b, c, d and e are used as the training set, and the structure a is used as the test set. The reason for this is that for our purposes it is more interesting to discuss in detail one experiment, than to summarize the 5 experiments which would be obtained applying cross-validation. This is particularly true since starting clauses and cpu-times are compared.

In the Tables 10 and 11 that present the results of our experiments in this domain, the headline indicates the bias instantiation. Each table presents the following results:

Accuracy on the training set: the total number of edges of the structures **b**, **c**, **d** and **e** that were not handled by NINA. There are no covered negative examples, since NINA is designed such that both starting clauses and hypotheses are required to be consistent with the negative evidence.

Accuracy on the test set:

- a *correct positive* is an edge of the structure **a** that is partitioned into the correct number of parts by the rules induced on the training set. The correct number is the number as indicated in the positive example for this edge.
 - a *covered negative* is an edge of the structure **a** that is partitioned into the wrong number of parts by one or more of the rules induced on the training set. A wrong number is a number that occurs in one of the negative examples for this edge.
 - *no value* means that the edge is not classified by any of the rules induced on the training set.
 - *correct + wrong* means that the edge is covered by more than one rule induced on the training set. One of these rules gives the correct partitioning, the others divide the edge into a wrong number of parts.
- **Time:** the times in cpu seconds on a Sparc Sun4. We give separate times for inducing a hypothesis (learn) and post-processing the hypothesis (reduce).
 - **Literals in starting clause:** for each language bias all numbers of literals that occurred in our tests are listed. E.g., $\mathcal{L}_2:12-13-14$, means that in $\mathcal{L}_2$ NINA found starting clauses with 12, 13 and 14 literals. In case there is only one bias in the test, the language is not explicitly mentioned. Note also that we detailed these results for the different subrelations of the relation `mesh(.,.)`. For the relation `mesh(.,11)` there are no results, since there are no edges in the training set that should be divided into 11 parts.

Note that both the accuracy and the time results concern the entire `mesh/2` relation, i.e., all subrelations together.

5.3.3. Partial answers to the questions

In this section we discuss a number of specific experiments which indicate partial answers to some of the questions posed in Section 5.2.

Determinacy

To answer the first question, i.e., to determine the influence of the semantic restriction of determinacy on the learning results, we ran two experiments, one in `mesh-design` and one in `family relations`.

The results of the tests in family relations are shown in Table 9. We start the discussion with the first two tests on the *brother/2* relation. The second of these shows that with the determinacy restriction starting clauses are shorter, and learning takes less time. The first test, however, shows that the determinacy requirement can be too restrictive. With the GOLEM bias NINA cannot build consistent starting clauses since the *parent* relation is not determinate (a person has 2 parents), although the definition for *brother/2* belongs syntactically to $\mathcal{L}_1$ of the GOLEM bias. With the CLINT bias on the other hand, NINA identifies a correct definition, since it belongs to $\mathcal{L}_2$ of the CLINT bias, and there are no semantic restrictions. These two experiments suggest the following trend:

Trend 1 *The determinacy restriction efficiently prunes the search space, but at the risk of losing solutions.*

At first sight, the second test on the *grandfather/2* relation seems to contradict this trend, as the GOLEM bias seems to be much less efficient than the CLINT bias. This is due to the fact that with the CLINT bias, the system only needs to introduce variables at level 1, and to perform unification between those variables. With the *ij*-determinate languages, however, it was in some cases necessary to shift to a language introducing variables at level 2, thus making the starting clauses unnecessarily complex.

Notice also that in these four tests, when the determinacy restriction is employed, the system produces shorter starting clauses than when using no semantic restrictions.

Table 9. Influence of determinacy in the family relations domain.

brother/2	P: all, N: CWA, BT: male/1, female/1, parent/2				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
	CLINT	$\mathcal{L}_2$: 5-6-7	1	3.00 + 0.53	cp:5 cn:0 up:0 un:164
	GOLEM	<i>fails</i>			
	P: all, N: CWA, BT: male/1, female/1, father/2, mother/2				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
grandparent/2	CLINT	$\mathcal{L}_2$:8-10	1	6.27 + 0.67	cp:5 cn:0 up:0 un:164
	GOLEM	$\mathcal{L}_2$:8	1	5.41 + 0.24	cp:5 cn:0 up:0 un:164
	P: all, N: CWA, BT: male/1, female/1, parent/1				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
	CLINT	$\mathcal{L}_2$:5-6	1	2.57 + 0.27	cp:14 cn:0 up:0 un:1
	GOLEM	<i>fails</i>			
	P: all, N: CWA, BT: male/1, female/1, father/2, mother/2				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
	CLINT	$\mathcal{L}_2$:6-7-8	4	12.22 + 2.17	cp:14 cn:0 up:0 un:155
	GOLEM	$\mathcal{L}_1$:7 $\mathcal{L}_2$:10-16	4	84.32 + 1.38	cp:14 cn:0 up:0 un:155

We also tested our hypothesis with regard to determinacy on the larger domain of finite element mesh design. The results of these experiments are shown in Table 10. In these

specific experiments, we ran NINA with a language allowing for level 1 ($i = 1$), and this both with and without the determinacy restriction. The results of these experiments confirm our earlier hypothesis, i.e., with the determinacy restriction, starting clauses tend to be smaller and the cpu-time required shorter. However, the hypothesis derived under the determinacy restriction has a slightly lower accuracy, which confirms the fact that with determinacy certain solutions may be lost. The latter effect seems to be less important in the finite element mesh domain.

Table 10. Influence of determinacy in the FE domain

	$i = 1$	$i = 1$ and determinacy (degree 1)
Accuracy on training-set	33 uncovered positives	95 uncovered positives
Accuracy on test-set	16 correct	15 correct
	14 wrong	15 wrong
	20 no value	23 no value
	5 correct + wrong	2 correct + wrong
Time (learn+reduce)	2248.52 + 8045.02 secs	1788.58 + 877.50 secs
Length of starting clauses		
mesh(.,1)	12-13-14-16-17-20	4-8-9-12-13
mesh(.,2)	8-12-13-16-17-21-22-29	4-8-9-12-13-16
mesh(.,3)	12-13-14-16-22	8-12-16
mesh(.,4)	12-13-16	-
mesh(.,5)	16-22	12
mesh(.,6)	16-22-27	8-12-13-18-23
mesh(.,7)	17-21	12-13-16
mesh(.,8)	21-22-26-27	12-13-16-17-19-22-27
mesh(.,9)	19-21-22-23	8-12
mesh(.,10)	12	4
mesh(.,11)	-	-
mesh(.,12)	12-18-23	4-18-23

Shift of bias

To test whether a shift of bias is useful or not, we ran NINA on the finite element meshes, once without a shift of bias (employing CLINT's $\mathcal{L}_3$), and once with a shift of bias (employing CLINT's $\mathcal{L}_0, \dots, \mathcal{L}_3$). In both experiments the global expressivity of hypotheses was the same, as the last language in both sequences was $\mathcal{L}_3$. The results of this experiment are shown in Table 11. In both runs, the hypothesis derived was the same; however, using a shift of bias proved to be much more economic. For this specific run, the cpu-time was reduced by nearly 50 per cent. A possible explanation is that with a shift of bias the average length of the starting clauses is much shorter. Of course, if NINA had to shift its bias for all positive examples to the last language in the series (in order to find a consistent starting clause), the resources consumed by the test with the shift of bias would be larger than without the shift of bias. This situation is however rather unlikely, unless the sequence is chosen badly with regard to the application.

We observed the following trend:

Trend 2 *Using a sequence of languages and shifting the bias can be more economic than learning in the union of the languages in the sequence.*

Table 11. Shift of bias in the FE domain

	$\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$	$\mathcal{L}_3$
Accuracy on training-set	33 uncovered positives	33 uncovered positives
Accuracy on test-set	16 correct 14 wrong 20 no value 5 correct + wrong	16 correct 14 wrong 20 no value 5 correct + wrong
Time (learn+reduce)	1292.16 + 7535.55 secs	2248.52 + 8045.02 secs
Length of starting clauses		
mesh(.,1)	$\mathcal{L}_0:4 \mathcal{L}_1:7 \mathcal{L}_3:12-13-16$	12-13-14-16-17-20
mesh(.,2)	$\mathcal{L}_0:4 \mathcal{L}_1:7-8 \mathcal{L}_2:7 \mathcal{L}_3:12-16-17-21$	8-12-13-16-17-21-22-29
mesh(.,3)	$\mathcal{L}_1:7 \mathcal{L}_3:16-17$	12-13-14-16-22
mesh(.,4)	$\mathcal{L}_3:12-13-16$	12-13-16
mesh(.,5)	$\mathcal{L}_3:16-22$	16-22
mesh(.,6)	$\mathcal{L}_0:4 \mathcal{L}_1:10 \mathcal{L}_3:16-25-27$	16-22-27
mesh(.,7)	$\mathcal{L}_0:4 \mathcal{L}_3:17-21$	17-21
mesh(.,8)	$\mathcal{L}_0:4 \mathcal{L}_1:8-9 \mathcal{L}_3:26$	21-22-26-27
mesh(.,9)	$\mathcal{L}_0:4$	19-21-22-23
mesh(.,10)	$\mathcal{L}_0:4$	12
mesh(.,11)	-	-
mesh(.,12)	$\mathcal{L}_0:4 \mathcal{L}_3:18-23$	12-18-23

Non-factual evidence

The influence of non-factual evidence, i.e., of definite clauses as positive examples and denials as negatives, was tested in the family relations domain. The results on learning brother are shown in Table 12. The results for the other predicates and for larger databases (containing 10 predicates or more) are very similar. No results for the GOLEM bias are included here, because of the determinacy restriction.

In this table, the sets of examples are as follows:

$$\begin{aligned}
 P_1 &= \{\text{brother}(\text{bart}, \text{joost}) \leftarrow; \text{brother}(\text{bart}, \text{miet}) \leftarrow\} \\
 P_2 &= \{\text{brother}(\text{bart}, \text{joost}) \leftarrow \text{parent}(\text{jan}, \text{joost}); \text{brother}(\text{bart}, \text{miet}) \leftarrow\} \\
 P_3 &= \{\text{brother}(\text{hans}, \text{tine}) \leftarrow; \text{brother}(\text{bart}, \text{joost}) \leftarrow; \text{brother}(\text{bart}, \text{miet}) \leftarrow\} \\
 P_4 &= \{\text{brother}(\text{hans}, \text{tine}) \leftarrow \text{parent}(\text{bart}, \text{hans}); \text{brother}(\text{bart}, \text{joost}) \leftarrow; \\
 &\quad \text{brother}(\text{bart}, \text{miet}) \leftarrow\} \\
 N_1 &= \{\leftarrow \text{brother}(X, Y), \text{female}(X); \leftarrow \text{brother}(X, Y), \text{parent}(X, Y)\}
 \end{aligned}$$

Observe that with only facts as positive examples (P_1 and P_3), the bias of ITOU (which cannot introduce new constants or variables in clauses) makes NINA fail. With definite clauses, this bias produced the same results obtained with CLINT's bias, but

Table 12. Results on the use of non-factual evidence in the family relations domain.

brother/2	P: P_1 , N: CWA, BT: male/1, female/1, parent/2				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
	CLINT	$\mathcal{L}_2:6-7$	1	2.16 + 1.02	cp:5 cn:0 up:0 un:164
	ITOU	<i>fails</i>			
	P: P_2 , N: CWA, BT: male/1, female/1, parent/2				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
	CLINT	$\mathcal{L}_2:7$	1	2.23 + 0.96	cp:5 cn:0 up:0 un:164
	ITOU	5	1	0.41 + 0.53	cp:5 cn:0 up:0 un:164
	P: P_3 , N: N_1 , BT: male/1, female/1, parent/2				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
	CLINT	$\mathcal{L}_2:7$	1	1.76 + 0.13	cp:5 cn:0 up:0 un:164
	ITOU	<i>fails</i>			
	P: P_4 , N: N_1 , BT: male/1, female/1, parent/2				
	Bias	sc-literals	H-clauses	Time (l.+r.)	Accuracy
	CLINT	$\mathcal{L}_2:6$	1	1.85 + 0.29	cp:5 cn:0 up:0 un:164
	ITOU	5	1	0.5 + 0.23	cp:5 cn:0 up:0 un:164

much faster. The gain in speed can be explained by the fact that ITOU's language is still less expressive than the corresponding one in CLINT, which results in shorter starting clauses for ITOU. These tests clearly show that definite clauses as examples can alleviate problems with the bias and problems with introducing new terms in starting clauses. If all terms are present in the evidence the learning task becomes easier. With regard to the negative evidence, it is easy to see that replacing CWA by a few denials has a positive influence on the computation time. We believe the trend in these experiments is very clear: non-factual evidence should be used whenever possible as it may simplify the bias and reduces both the learning time and the number of examples needed.

We can identify the following trend:

Trend 3 *Non-factual evidence can reduce the number of examples needed and the learning time, and can help to alleviate problems with the bias.*

Influence of the Background Theory

To test the influence of the background theory, we performed a number of subsequent tests on the family relations domain, in which we increased each time the number of predicates in the background knowledge. In these tests the predicate `grandparent/2` was learned, given 6 ground facts as positive evidence, CWA as negative evidence, and the series $\{\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3\}$ as syntactic bias. No extra semantic restrictions were imposed.

Table 13 shows the times needed for learning and postprocessing a hypothesis for the `grandparent/2` relation. In Figure 2 the length of the starting clauses is plotted against the number of predicates in the background knowledge. Next to each plot, we indicate

Table 13. Results of learning the relation `grandparent/2` with increasing number of predicates in the background theory.

	Predicates in the background theory	Time (learn+reduce)
Test 1	male/1, female/1, father/2, mother/2	3.6 + 0.75 secs
Test 2	male/1, female/1, father/2, mother/2 sister/2, brother/2, married/2	23.77 + 0.83
Test 3	male/1, female/1, father/2, mother/2 sister/2, brother/2, married/2, uncle/2, aunt/2	272.1 + 1.97
Test 4	male/1, female/1, father/2, mother/2 sister/2, brother/2, married/2, uncle/2 aunt/2, niece/2, nephew/2, cousin/2	37167.29 + 1.95 secs

also the average time needed for computing a starting clause. All starting clauses were found in the language $\mathcal{L}_2$.

It is clear that the size of the starting clauses increases as the background knowledge grows larger. Indeed, the more predicates there are in the background theory, the more - possibly irrelevant - literals are included in the starting clause. Moreover, also the time needed for computing a starting clause increases significantly. As a consequence, the time needed for inducing a hypothesis, and post-processing it, will grow with the number of predicates in the background theory.

We can identify the following trend:

Trend 4 *Both the size of the starting clauses and the computation time grow (exponentially) with the number of relations in the background theory.*

This trend may seem counterintuitive in the sense that one would expect that extra background knowledge would speed up the learning process. However, the trend formulated here is specific to the ILP approach and to the method of using the background knowledge as predicates in the search for a starting clause. The trend also points out the need for biases in order to make the learning process more efficient.

6. Conclusions and related work

We have presented several contributions concerning the declarative bias issue in inductive logic programming. First, we have formulated a conceptual framework for bias, that cleanly separates syntactic and semantic bias and that discusses the role of a shift of bias. Second, we have introduced a new and attractive formalism for the declarative specification of syntactic bias in inductive logic programming systems. Third, we have presented the generic algorithm and system NINA that learns predicates from examples, bias and background knowledge. NINA combines interesting abilities from several different systems, mostly from GOLEM, ITOU and CLINT, i.e., semantic bias from GOLEM, non-factual positive evidence from ITOU, and shift of bias from CLINT. Fourth, we

Figure 2. Computation of starting clauses with increasing background knowledge. E.g., with four predicates in the background theory, the number of literals in the starting clauses ranged from 6 to 8, and the average time needed for computing such a starting clause is 0.54 cpu-secs.

have shown that NINA is appropriate for bias evaluation purposes. In particular, using NINA to experiment in the family relations domain and the finite element mesh domain, we have identified some important trends concerning bias. More specifically, these are: 1) the determinacy restriction efficiently prunes the search space, but at the risk of losing completeness; 2) using a sequence of languages and shifting the bias can be more economic than learning in the union of the languages in the sequence; 3) non-factual evidence can reduce the learning time and the number of examples needed, and can help to alleviate problems with the bias; and 4) having too many predicates in the background theory can substantially reduce the performance of the system. Finally, NINA has allowed us to show that the three seemingly disparate systems GOLEM, ITOU and CLINT

have much in common concerning the generation of starting clauses and the use of a syntactic bias.

We believe this is the first experimental and general study of the influence of bias in specific-to-general ILP. Although the experimental results were obtained on relatively few and simple domains, several interesting trends were identified. They seem to confirm the need for using strong biases in ILP (cf. the experiments with the family relations domain), and the possible impact of using a shift of bias mechanism. This puts some of the earlier work on the CLINT system (De Raedt, 1992) into a new perspective. Whereas in CLINT, a shift of bias was used to find a solution at all, it can now also be used to find solutions in a more efficient manner (cf. Trend 2). In this context, it would also be interesting to investigate the influence of predicate invention, when viewed as a bias shift operation (cf. (Stahl, 1994)). These and other experiments would be needed to provide stronger evidence for the trends we identified. However, as long as ILP lacks a real experimental methodology, it remains hard to systematically justify conjectures in this field.

Concerning the bias representation formalism, we showed that our approach integrates earlier frameworks by Bergadano and the MOBAL team. Schemata as employed in MOBAL correspond to the subset of our framework without the set notations, and Bergadano's language is essentially ours without predicate variables. Other declarative bias specification formalisms have been introduced by Tausend (this is related to the MOBAL schemata, see (Tausend, 1994)) and Cohen (see (Cohen, 1994)). A comparative survey of these biases has been published by Tausend (see (Tausend, 1994)). Whereas our formalism focuses on specifying the syntax of clauses in hypotheses, some of the others have also addressed semantic issues such as types, modes, etc. We feel that such semantic aspects should be seen as complementary to the syntax. Syntax and semantics are therefore best handled separately, allowing for a clear understanding of both issues.

As far as related work is concerned, we would like to mention William Cohen's GRENDDEL system (Cohen, 1994), which can be considered a generic framework for evaluating the bias of general-to-specific inductive logic programming systems. In spirit, NINA is for specific-to-general ILP what GRENDDEL is for general-to-specific ILP.

Acknowledgments

The authors would especially like to thank Céline Rouveirol at the LRI, University of Paris-Sud for the discussions, the comments, and certainly for her invaluable share in the work that preceded this paper.

This work is supported by the Esprit Basic Research Action ILP (project 6020) and by the 'Vlaamse Gemeenschap' through contract nr.93/014. Hilde Adé did part of the research during her stay at LRI, University of Paris Sud, made possible by a grant of the Belgian National Fund for Scientific Research. Luc De Raedt and Maurice Bruynooghe are supported by the Belgian National Fund for Scientific Research.

The authors would like to thank Diana Gordon for encouraging us to write this paper. We also thank the anonymous reviewers for their very valuable comments and suggestions. We are grateful to Boyan Dolšak for making the data on the Finite Element Mesh Design Domain available, and to Peter Flach for making his implementation of the lgg-operator available, which can be found in his book (Flach, 1994).

Finally, many thanks to Gunther Sablon for his comments on the earlier versions of the paper, and for his help in the implementation of the NINA system. Also thanks to Luc Dehaspe for correcting this text.

Notes

1. For the time being, we assume here that *starting_clause* returns a single starting clause. The extension to the more general case consists of making *starting_clause* nondeterministic, and enumerating all solutions through backtracking. This extension is introduced in Section 4.2.
2. A literal is redundant in a clause if the removal of this literal will not cause the clause to be inconsistent with the negative evidence.
3. A clause is redundant if removing this clause will not harm the completeness of the concept definition.
4. Various other kinds of refinement operators exist, but a full discussion of these would be very technical and contribute little to our bias evaluation method. We refer to (Muggleton & De Raedt, 1994), (van der Laag & Nienhuys-Cheng, 1993) for more information.
5. Variabilizing the constants is done here because NINA assumes the depth of clauses in hypotheses is 0, i.e., NINA allows neither constants nor functors in its hypotheses.
6. Completely bound DATALOG Horn clauses are Horn clauses in DATALOG - i.e., without constants or functors - in which all variables in the head also appear in the body, and vice versa.
7. It is not exactly the language of clauses with maximum level 1, because in that language one could have several atoms of the same form, such as for instance $\text{parent}(X, Y_1), \dots, \text{parent}(X, Y_n)$ where n could be any positive integer, resulting in an infinite language. As this is impractical and in all specific-to-general systems excluded in one way or another (in GOLEM because of determinacy), we will work only with approximations of languages defined in terms of level.
8. A clause is range restricted if all the variables appearing in its head also appear in the body.

References

- Adé, H. & Bruynooghe, M. (1992). A comparative study of declarative and dynamically adjustable language bias in concept learning. In *Proceedings of the ML-92 Workshop on Biases in Inductive Learning*.
- Bergadano, F. (1993). Towards an inductive logic programming language. Technical Report ESPRIT project no. 6020 ILP Deliverable TO1, Computer Science Department, University of Torino.
- Bergadano, F. & Gunetti, D. (1993). An interactive system to learn functional logic programs. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1044–1049. Morgan Kaufmann.
- Buntine, W. (1987). Induction of Horn-Clauses: methods and the plausible generalization algorithm. *International Journal of Man-Machine Studies*, 26:499–520.
- Cameron-Jones, R.M. & Quinlan, J.R. (1993). Avoiding pitfalls when learning recursive theories. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1050–1055. Morgan Kaufmann.
- Cohen, W.W. (1994). Grammatically biased learning: learning logic programs using an explicit antecedent description language. *Artificial Intelligence*, 68:303–366.
- De Raedt, L. (1992). *Interactive Theory Revision: an Inductive Logic Programming Approach*. Academic Press.

- De Raedt, L. & Bruynooghe, M. (1992). Belief updating from integrity constraints and queries. *Artificial Intelligence*, 53:291–307.
- De Raedt, L. & Bruynooghe, M. (1992). Interactive concept-learning and constructive induction by analogy. *Machine Learning*, 8(2):107–150.
- De Raedt, L. & Bruynooghe, M. (1992). A unifying framework for concept-learning algorithms. *The Knowledge Engineering Review*, 7(3):251–269.
- De Raedt, L., Lavrač, N., & Džeroski, S. (1993). Multiple predicate learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1037–1042. Morgan Kaufmann.
- Dolšak, B. (1991). Constructing finite element meshes using artificial intelligence methods. Master's thesis, Faculty of Technical Sciences, University of Maribor, Slovenia. in Slovene.
- Dolšak, B. & Muggleton, S. (1992). The application of inductive logic programming to finite element mesh design. In S. Muggleton, editor, *Inductive logic programming*, pages 453–472. Academic Press.
- Džeroski, S., Muggleton, S., & Russell, S. (1992). PAC-learnability of determinate logic programs. In *Proceedings of the 5th ACM workshop on Computational Learning Theory*, pages 128–135.
- Emde, W., Habel, C.U., & Rollinger, C.R. (1983). The discovery of the equator or concept driven learning. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 455–458. Morgan Kaufmann.
- Flach, P. (1994). *Simply logical - Intelligent Reasoning by Example*. John Wiley & Sons.
- Kietz, J.-U. & Wrobel, S. (1992). Controlling the complexity of learning in logic through syntactic and task-oriented models. In S. Muggleton, editor, *Inductive logic programming*, pages 335–359. Academic Press.
- King, R.D., Muggleton, S., Lewis, R.A., & Sternberg, M.J.E.. (1992). Drug design by machine learning: the use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23).
- Lavrač, N. & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.
- Lavrač, N., Džeroski, S., Pirnat, V., & Križman, V. (1993). The use of background knowledge in learning medical diagnostic rules. *Applied Artificial Intelligence*, 7:273 – 293.
- Mitchell, T.M. (1982). Generalization as search. *Artificial Intelligence*, 18:203–226.
- Muggleton, S. editor. (1992). *Inductive Logic Programming*. Academic Press.
- Muggleton, S. & De Raedt, L. (1994). Inductive logic programming : Theory and methods. *Journal of Logic Programming*, 19,20:629–679.
- Muggleton, S. & Feng, C. (1990). Efficient induction of logic programs. In *Proceedings of the 1st conference on algorithmic learning theory*, pages 368–381. Ohmsma, Tokyo, Japan.
- Muggleton, S., King, R.D., & Sternberg, M.J.E. (1992). Protein secondary structure prediction using logic. *Protein Engineering*, 7:647–657.
- Pazzani, M. & Kibler, D. (1992). The utility of knowledge in inductive learning. *Machine Learning*, 8:57–94.
- Plotkin, G. (1970). A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press.
- Plotkin, G. (1971). A further note on inductive generalization. In *Machine Intelligence*, volume 6, pages 101–124. Edinburgh University Press.
- Quinlan, J.R. (1990). Learning logical definitions from relations. *Machine Learning*, 5:239–266.
- Quinlan, J.R. (1991). Knowledge acquisition from structured data - using determinate literals to assist search. *IEEE Expert*, 6(6):32–37.
- Rouveirol, C. (1991). Semantic model for induction of first order theories. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 685–690. Morgan Kaufmann.
- Rouveirol, C., Adé, H. & De Raedt, L. (1993). Bottom-up generalisation in ILP. In F. Bergadano, De Raedt, L., Matwin S., and Muggleton, S., editors, *Proceedings of the IJCAI93 workshop on ILP*, pages 59–70.
- Rouveirol, C. & Puget, J.-F. . (1989). A simple solution for inverting resolution. In Katharina Morik, editor, *Proceedings of the 4th European Working Session on Learning*, pages 201–210. Pitman.
- Russell, S. & Grosof, B. (1990). A sketch of autonomous learning using declarative bias. In P.B. Brazdil and K. Konolige, editors, *Machine Learning, Meta-Reasoning and Logics*, pages 19–54. Kluwer Academic Publishers.
- Russell, S.J. (1989). *The use of knowledge in analogy and induction*. Pitman.
- Shapiro, E.Y. (1983). *Algorithmic Program Debugging*. The MIT Press.

- Stahl, I. (1994). On the Utility of Predicate Invention in Inductive Logic Programming. In F. Bergadano and De Raedt, L., editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 272–286. Springer-Verlag.
- Tausend, B. (1994). Biases and Their Effects in Inductive Logic Programming. In F. Bergadano and De Raedt, L., editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 431–434. Springer-Verlag.
- Tausend, B. (1994). Representing Biases for Inductive Logic Programming. In F. Bergadano and De Raedt, L., editors, *Proceedings of the 7th European Conference on Machine Learning*, volume 784 of *Lecture Notes in Artificial Intelligence*, pages 427–430. Springer-Verlag.
- Utgoff, P.E. (1986). 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*, pages 107–148. Morgan Kaufmann.
- Utgoff, P.E. & Mitchell, T.M. (1982). Acquisition of appropriate bias for concept learning. In *Proceedings of the 1st National Conference on Artificial Intelligence (AAAI-82)*, pages 414–418. Morgan Kaufmann.
- van der Laag, P.R.J. & Nienhuys-Cheng, S.-H. (1993). Subsumption and refinement in model inference. In P. Brazdil, editor, *Proceedings of the 6th European Conference on Machine Learning*, volume 667 of *Lecture Notes in Artificial Intelligence*, pages 95–114. Springer-Verlag.