Are Substitutions the Better Examples? Learning Complete Sets of Clauses with Frog

Dieter Fensel (+), Monika Zickwolff (*), and Markus Wiese (#)

(+) Department SWI, University of Amsterdam, 1018WB Amsterdam, The Netherlands
e-mail: dieter@swi.psy.uva.nl

(*) Forschungsgruppe Begriffsanalyse, University of Darmstadt, 64289 Darmstadt, Germany
e-mail: zickwolff@mathematik.uni-darmstadt.de

(#) Institut AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
e-mail: wiese@aifb.uni-karlsruhe.de

Abstract. The paper presents an approach for machine learning in a restricted first-order language with finite minimal Herbrand models by means of a search through a propositional representation space. The learning target is to find a set of goal clauses which can be used to define a target predicate. That is, we deal with single-predicate learning. For the search process we use the learning algorithm JoJo/ Frog which provides a flexible search strategy. The transition from the first-order representation to the representation in propositional logic is achieved by ground substitutions which transform clauses into ground clauses. Taking a closer look at this transition makes clear that the *sufficiency* condition which is used by algorithms like FOIL and LINUS as a criterion for judging the achieved learning results does *not* correspond to the *completeness* condition in the propositional case. Therefore, we use an extended completeness condition which captures all information given by the example knowledge. As a consequence we get a new definition of positive and negative examples. Instead of ground facts we regard *ground substitutions* as examples.

Introduction

The paper presents a machine learning approach in a restricted first-order language. Example knowledge (also called evidence in the following) and background knowledge are generalized to form a finite set of *program clauses* [Llo87] which describe a target predicate. The language used to describe the input knowledge and the learned results is restricted to sets of first-order clauses having a *finite* minimal (i.e., perfect) Herbrand model [Ull88]. In this case, the given knowledge can be translated into an equivalent finite propositional representation. This representation can be used as a search space by the learning algorithm. In this paper we achieve finiteness by excluding function symbols (cf. [Rou94] for further possibilities).

The use of propositional logic as a search space for learning in first-order logic is not a new idea. The learning algorithms FOIL[1] (cf. [Qui90a], [QuC93]), LINUS (cf. [LDG91], [DzL93], [LDG91]), and MILP [Kov94] have already shown how a propositional representation can be used to learn clauses in first-order logic. Compared to these approaches, two main characteristics of our approach can be identified:

- We use the stronger *completeness* criterion of algorithms like CLAUDIEN [RaB93] instead of the weaker *sufficiency* criterion of FOIL and LINUS. We demonstrate that learning strategies like those of FOIL do not use the full information given by the input knowledge. This becomes obvious when we take a closer look at the transition from first-order to propositional logic which is achieved via the set of all substitutions. *From our point of view each ground substitution and not only each ground fact of the goal predicate corresponds to an example*. [MuR94] distinguishes two settings for machine learning in first-order logic which they call *normal semantics* and *nonmonotonic semantics*. The *(posteriory) sufficiency criterion* of FOIL and LINUS is oriented toward the first setting, whereas the *completeness criterion* which is used by CLAUDIEN and our approach is related to the second one. On the other hand, we distinguish between background knowledge and (incomplete) evidence as is generally done in the normal semantics setting. As a result, our approach bridges both settings.
- We use a more elaborated search strategy for the learning algorithm. In fact, we apply the learning algorithms *JoJo* and *Frog* for the search process. JoJo [FeW93] integrates generalization and specialization into a bi-directional search process. JoJo can start at an arbitrary point in the search space and iterates generalization and specialization steps until a local optimum has been found. Frog [Fen93b] can additionally make jumps in the search space.

Like FOIL and LINUS we deal with the case of single-predicate learning. That is, we assume a correct background knowledge B and incomplete positive and negative information (called evidence E) over a target predicate p. The learned hypothesis H describes p in a generalized but still correct manner.

The content of the paper is organized as follows. In the first chapter, we describe *what* our approach does. That is, we give a *declarative* description of the result of the learning process. In particular, we show how completeness in the propositional case corresponds to completeness in the first-order case. Chapter two briefly sketches out the heuristic learning procedures which are used to find such a hypothesis. Thus chapter two discusses *procedurally how* we reach our learning goals. Chapter three gives an evaluation of our implementation and, finally, we outline some directions for future work.

---

1. FOIL searches through the space of all ground instantiations of a target clause. Therefore, FOIL uses a propositional search space by applying ground substitutions to first-order clauses. During the learning process, the search space is extended by introducing new variables.

1 Representing First-Order Theories with Propositional Logic

The theoretical background for translating a first-order language into a propositional representation is given by the theory of *Herbrand interpretations*, which enable the description of clauses in a first-order language through the use of propositional means—which are Herbrand universes, bases, interpretations, and models (cf. [Llo87]). In general, if the set of ground literals is infinite, the propositional representation requires infinitely many literals. An easy way to achieve finite Herbrand models (i.e., finite propositional representations) is to forbid function symbols. In fact, this is the way we achieve finiteness in the representation of the evidence, the background knowledge, and the learned goal clauses.

There are, however, some major differences depending on whether a machine learning system is applied for learning propositional rules or whether it is applied to clauses in a first-order language. This is due to the fact that variables are introduced in first-order logic, a circumstance which has several consequences. This becomes important when investigating the definitions of *covering an example* and *completeness*—especially if we compare these concepts with the ideas used in learning systems like FOIL and LINUS.

In order to compare our approach to the standard definitions given in [MuR94], we will—for reasons of clarity—roughly sketch each of them:

- (1) *Normal semantics*: This setting distinguishes between background knowledge B and example knowledge E (evidence). The learning goal is a hypothesis H which is *consistent* with the negative evidence E^-
  $\mathrm{B} \cup H \cup E^- \not\models \Box$
  and *sufficient* in regard to the positive evidence E^+
  $\mathrm{B} \cup H \models E^+$.
- (1a) *Definite semantics*: This setting replaces the logical consequence $\models$ with a non-monotonic entailment $|\approx$ using the closed-world assumption (i.e., the minimal Herbrand model as semantics)[2].
- (2) *Nonmonotonic semantics*: In this setting, background knowledge B is given, but no evidence E. Therefore a different learning goal is defined. A hypothesis H is *correct* if H is true in the minimal Herbrand model $\mathrm{M}^+(B)$. A hypothesis H is *complete* if every clause g which is true in $\mathrm{M}^+(B)$ is true in $\mathrm{M}^+(H)$. Again the closed-world assumption is applied.

The main difference between (1) and (2) appears to be that in the nonmonotonic setting:

$B \Leftrightarrow H$ holds true according to the minimal model semantics.

Hence in (2), H is "only" a reformulation of B. In (1), however, H is normally not equivalent to $\mathrm{B} \cup E$ as the scope of the theory should be extended to include unknown cases.

1.1 The Representation Formalism and Learning Setting

Most of the terms and definitions we use refer to [Llo87]. The first definition introduces the alphabet and the language for representing input and learned knowledge. As the

2. See [Cla78], [Llo87], [Ull88].

number of variables influences the content of information which is processed by the learning algorithm and, conversely, the search effort in the enlarged hypothesis space it is necessary to restrict the number of variables. In our case we define an upper limit for the number of variables which can be used in a clause by restricting the number of variables offered by the alphabet. Our definition uses finiteness as a language bias for the learning task. The language provides only a finite number of denotations and, as function symbols are excluded, the minimal (i.e., perfect) Herbrand models are also finite.

Def. 1 (cf. Alphabet and language)

An *alphabet A* of a first-order language consists of a finite set of predicate symbols $\{p, b_1,..., b_m\}$, a finite set of constants $\{c_1,...,c_l\}$, and a finite set of variables $\{x_1,..., x_k\}$. In addition, it provides the usual logical connectives ($\vee$, $\wedge$, $\neg$, $\leftarrow$) and punctuation symbols. As we regard every variable as universally quantified, we do not need quantifiers in the alphabet. A *language L(A)* to an alphabet *A* is the set of well-defined and universally quantified clauses over the alphabet *A*.

The following definition introduces the difference between a target predicate symbol—which is used as the learning target—and further predicates symbols for the background knowledge. These different predicate symbols are used to define evidence, background knowledge, and the learning goal.

Def. 2 (cf. Target and background predicates and literals)

One predicate symbol is called the *target predicate* and denoted by p. The others are called *background predicates* and denoted by $b_1,..., b_m$.
A literal using the target predicate is called a *target literal*. A literal using a background predicate is called a *background literal*. These literals will be denoted by l_i.

The following definition ensures syntactical finiteness of the regarded clauses. Each atom of the language can appear at most once in an allowed premise. As we regard only finite Herbrand models as semantics, this restriction does not change the expressive power of the language.

Def. 3 (cf. Allowed premises)

A ground (non-ground) *allowed premise P* is a conjunction of ground (non-ground) background literals $l_1 \wedge ... \wedge l_n$ with

- $l \in \{l_1,...,l_n\} \Rightarrow \neg l \notin \{l_1,...,l_n\}$ with $\neg\neg l = l$ and
- $l_i \neq l_j$ for all $i,j = 1,...,n$.

Only allowed premises are regarded further on.

The case of recursion where the target predicate is also used as a premise of a goal clause is discussed in section 1.5.

In the normal semantics setting, background knowledge *B* is generally viewed as a set of true facts and clauses. Actually, *B* corresponds to the deductive closure of a set of formulae. In the nonmonotonic setting, which uses the closed-world assumption, *B* is viewed as the least Herbrand model $M^+(B)$ of a set of clauses and facts. As we apply the closed-world assumption for background knowledge, we view background knowledge as its least Herbrand model $M^+(B)$. In fact, because we allow stratified negation [Ull88], we use the *perfect* Herbrand model. Thus we allow a restricted version of negation in the bodies of our clauses. In the case of negation in the bodies of clauses, there is no longer a unique minimal Herbrand model. The stratification can be used to select one of them, which is called the *perfect Herbrand model*. Therefore our conception of background knowledge corresponds to the definite semantics setting slightly generalized to the use of perfect Herbrand models.

Def. 4 (cf. Background knowledge)

The *background knowledge* B is a set of stratified program clauses using only background predicates. $M^+(B)$ denotes the perfect Herbrand model of B and defines its semantics.

The evidence E should provide the learner with partial knowledge about the target predicate. Again, this can be expressed by a Herbrand interpretation. In general, however, we cannot assume the completeness of the interpretation since we cannot expect complete knowledge about the target predicate. In fact, we view the induction of new knowledge about the target predicate as the most important goal of our learning algorithm. We intend to learn clauses which make predictions for unknown ground instances of the target predicate. As a consequence, the Herbrand interpretation is a *partial interpretation* for the target predicate. Again, the definition of evidence corresponds to the definitions of FOIL and LINUS.

Def. 5 (cf. Evidence)

The *evidence* E is a set of ground target literals. E^+ (*positive* evidence) contains positive and E^- (*negative* evidence) contains negative literals.

To ensure *consistency* or *prior satisfiability* of the knowledge $M^+(B) \cup E$ it must hold:

$$l \in E^+ \Rightarrow \neg l \notin E^-$$

The following definition introduces the syntactical structure of the goal of the learning process.

Def. 6 (cf. Goal clause and hypothesis)

A *goal clause* is a non-ground program clause of the form:

$$p(x_1,..., x_p) \leftarrow P$$

where P is an allowed premise.

A *hypothesis* H is a set of goal clauses.

1.2 Transition From First-order to Propositional Logic: What is an Example?

In the case of representation formalisms based on attribute values (and therefore in propositional logic), an example can be viewed as an rule which has a conjunction of the attribute values of the example as its premise and its class membership as its conclusion. Table 1 shows a well-known illustration from [Qui84]. The first and the fourth rows, that

Table 1. An illustration for an attribute-value-based representation.

class	eye-color	hair-color	size
+	blue	blond	small
+	blue	red	tall
+	blue	blond	tall
−	brown	blond	tall
...	...	...	...

is the first positive and negative examples, correspond to the following rules:

$$class \leftarrow eye\text{-}color = blue \wedge hair\text{-}color = blond \wedge size = small.$$

$$\neg class \leftarrow eye\text{-}color = brown \wedge hair\text{-}color = blond \wedge size = tall.$$

The transition from our restricted first-order logic to propositional logic is achieved by means of ground substitutions. Each ground substitution produces a row in a table. Each substitution θ with $p(x_1,..., x_p)\theta \in E^+$ produces a positive example and each substitution θ' with $\neg p(x_1,..., x_p)\theta' \in E^-$ produces a negative example in the propositional representation. Therefore it becomes clear that each substitution and not only each ground literal of the goal predicate corresponds to what is called an example in propositional logic. Let us regard the following small problem given in first-order logic:

Predicates of the background knowledge:
$\{P(\_,\_), Q(\_)\}$
Constants:
$\{a, b, c\}$
Set of positive literals holding in the background knowledge B:
$M^+(B) = \{P(a,a), P(a,b), Q(a), Q(b)\}$
We can infer with the closed-world assumption that $\neg P(b,a)$ = true, etc.
Evidence E for the target predicate:
$\{p(a), \neg p(b)\}$. That is, $E^+ = \{p(a)\}$ and $E^- = \{\neg p(b)\}$ Notice that E defines only a partial interpretation.
Number of variables: 2.

This knowledge can be propositionally represented by Table 2. In addition to the given predicates we add some built-in predicates (called utility predicates in LINUS, cf. [LDG91]). These predicates can be binary and used to express equality between variables like $=(x,y)$ or they can be unary and used to express the equality between a variable and a constant like $=_a(x)$, see [RoP89]. The pre-defined semantics of these predicates can be used to avoid redundancy. For example, we know that $=(x,y)$ implies $=(y,x)$ or $=_a(x)$ implies $\neg(=_b(x))$. Clearly, the user can decide whether he want to use these predicates or not. The set of all non-ground atoms of the background knowledge (including the goal predicate in the case of recursion) determines the number of columns of the table. Each non-ground atom creates one column. The rows of the table are determined by all possible ground substitutions which depends on the number of variables and constants in the alphabet.[3] Each background predicate together with the variables it uses, defines a binary attribute. The truth values according to a given substitution define its value (i.e., 0 or 1).

3. For a more detailed discussion of this procedure and on complexity results see [LDG91].

For example, the second row of Table 2 corresponds to the propositional example:

Table 2. Propositional representation of function-free first-order logic

(x,y)	$p\ (x)$	$P(x,y)$	$P\ (y,x)$	$Q(x)$	$Q(y)$	$=(x,y)$	$=_a(x)$	...	$=_c(y)$
(a,a)	+	1	1	1	1	1	1		0
(a,b)	+	1	0	1	1	0	1		0
(a,c)	+	0	0	1	0	0	1		1
(b,a)	-	0	1	1	1	0	0		0
(b,b)	-	0	0	1	1	1	0		0
(b,c)	-	0	0	1	0	0	0		1
(c,a)	?	0	0	0	1	0	0		0
(c,b)	?	0	0	0	1	0	0		0
(c,c)	?	0	0	0	0	1	0		1

$$p(a) \leftarrow P(a,b) = 1 \wedge P(b,a) = 0 \wedge Q(a) = 1 \wedge Q(b) = 1 \wedge =(a,b) = 0 \wedge {}_a(a) = 1 \wedge ... \wedge {}_c(b) = 0$$

This expression is equivalent to:

$$p(a) \leftarrow P(a,b) \wedge \neg P(b,a) \wedge Q(a) \wedge Q(b) \wedge \neg =(a,b) \wedge {}_a(a) \wedge ... \wedge \neg {}_c(b)$$

Each *ground variable assignment* or each *ground substitution* of the target predicate generates an example (or an unknown case) in our propositional representation of the learning problem. Therefore, we expanded the definition of an *example* used by FOIL or LINUS. FOIL and LINUS regard each element of E^+ as a positive example and each element of E^- as a negative examples. Therefore, an example corresponds to a ground fact of the target predicate. According to our point of view, *each ground substitution* θ which assigns a constant to each variable of the language $L(A)$ is either a positive example, a negative example, or an unknown case.

Def. 7 (cf. Examples)

A *positive example* is a ground substitution θ with $p(x_1,..., x_p)\theta \in E^+$. A *negative example* is a ground substitution θ' with $\neg p(x_1,..., x_p)\theta' \in E^-$.

Therefore, we obtain the three positive examples $\theta_1 = \{(x/a),(y/a)\}$, $\theta_2 = \{(x/a),(y/b)\}$, $\theta_3 = \{(x/a),(y/c)\}$ in our illustration and not just one positive example $p(a)$. Further on, we have three negative examples $\theta_4 = \{(x/b),(y/a)\}$, $\theta_5 = \{(x/b),(y/b)\}$, $\theta_6 = \{(x/b),(y/c)\}$ and three unknown cases. In the next section, we will show that this definition of an example provides a strong result concerning the completeness of the learned hypothesis. A hypothesis H which is *complete in the propositional representation space* when it covers all positive ground substitutions (i.e., substitutions for which the target literal is in E^+) is also *complete in the first-order representation space*. Thus all *goal clauses* which are true in $M^+(B) \cup E^+$ are also true in $M^+(B \cup H)$. This may not hold true for a hypothesis learned by FOIL or LINUS. *As ground substitutions are the means for transforming a first-order theory into its propositional counterpart, it is, in fact, this point of view on an example which translates the completeness of the propositional case into the first-order case.*

To learn first-order clauses with our propositional model we regard ground substitutions, i.e., ground goal clauses with most-specific premises as examples.[4] We have to make the following transfer: in the propositional case each example can be directly used as a (most specific) classification rule which is correct if the example knowledge is separable[5]; the transfer from examples to goal clauses in the first-order case is more complicated: each positive example, that is, ground substitution θ with $p(x_1,..., x_p)\theta \in E^+$, provides a goal clause:

$$p(x_1,..., x_p) \leftarrow l_1,..., l_n$$

where each non-ground literal l_i of the language $L(A)$ occurs in the hypothesis if $M^+(B) \mid\approx l_i\theta$. As in the propositional case, this clause is correct if the examples are *separable*, which means that the language considered (predicates and number of variables) is sufficient to separate positive from negative examples. However, the transfer from a (propositional) example to a universally quantified clause with variables which hold true *for all* substitutions requires a more complex definition of separability.

Def. 8 (cf. Most-specific premises)

A ground (non-ground) *most-specific premise* P is a ground (non-ground) allowed premise where

$P \subseteq P'$ implies that P' is not an allowed premise.

Def. 9 (cf. Reverse substitution)

A *reverse substitution* $\overline{\theta}$ is a set $\{(t_1/x_1),..., (t_k/x_k)\}$, where each t_i is a constant of $L(A)$ and $x_i \neq x_j$, $t_i \neq t_j$ for $i \neq j$. Such a reverse substitution $\overline{\theta}$ is applied to a ground clause by simultaneously replacing each ground term t_i with the variable x_i.

Def. 10 (cf. Separable)

Knowledge which is described by a partial Herbrand interpretation $M^+(B) \cup E$ is called *separable* if no two ground implications

$p(t_{11},..., t_{1p}) \leftarrow P$

$\neg p(t_{21},..., t_{2p}) \leftarrow P'$

exist with P, P' are most-specific premises, $M^+(B) \mid\approx \{P, P'\}$, $p(t_{11},..., t_{1p}) \in E^+$, $\neg p(t_{21},..., t_{2p}) \in E^-$ and two reverse substitution $\overline{\theta}_1$ and $\overline{\theta}_2$ exist with

$P\overline{\theta}_1 = P'\overline{\theta}_2$ and $p(t_{11},..., t_{1p})\overline{\theta}_1 = p(t_{21},..., t_{2p})\overline{\theta}_2$.

Separability enables the generalization of most-specific ground clauses through the replacement of constants with variables without this leading to inconsistency. If separability is not given, then the language provided does not allow a separate description of the positive and negative examples by means of non-ground clauses. In propositional logic, separability is implied by consistency, but this no longer holds true in the first-order case. For example, the following theory is not separable:

$E^+ = \{p(a)\}$; $E^- = \{\neg p(b)\}$; $B = \{q(a), q(b)\}$

The problem of non-separable theories can always be solved by introducing additional

4. Clauses with most-specific premises are also constructed by ILP systems using a bottom-up search strategy. For this purpose, *saturation* is defined (see [Rou94]).

5. Thus no positive and negative examples exist which have the same values for each attribute.

unary predicates which mimic the constants. Thus these predicates are true if the according constant is substituted for the variable. Actually, the propositional learning procedure Frog (which is introduced in section 2) can also deal with noise. Therefore, it can be applied to non-separable theories. It uses the ratio of all ground substitutions θ cover ed by a goal clause which are positive and negative examples as a measurement for the accuracy of the clause.

1.3 Sufficiency and Completeness

In the propositional case, the *completeness* of a set of goal clauses H is achieved if every positive example is *covered* by H—that is, if the positive atom *class* can be derived from every $H \cup p$ where p is the premise of a positive example. A set of goal clauses H for a given class is *correct* if it covers no negative example—that is, if the positive atom *class* can*not* be derived from any $H \cup p'$ where p' is the premise of a negative example.

One could transfer these definition to the first-order case: a hypothesis H is called *complete* if it covers all examples. This implies that

$$M^+(B \cup H) \mid\approx E^+.$$[6]

This is, in fact, the definition of *posterior sufficiency* [MuR94] and the *sufficiency*-condition of FOIL and LINUS.

Considering each *ground variable assignment* or each *ground substitution* of the target predicate as an example provides an expanded definition. We have chosen this point of view for two reasons: first, each ground substitution generates an example in our propositional representation of the learning problem; second, a hypothesis H which is *complete in the propositional representation space* when it covers all positive ground substitutions (i.e., substitutions for which the target literal is in E^+) is also *complete in the first-order representation space*. We will prove that a hypothesis H which covers all ground substitutions is *complete*. Thus all *goal clauses* which are true in $M^+(B) \cup E^+$ also follow from the background knowledge $M^+(B)$ and the set of learned goal clauses H. Before we illustrate the difference between hypotheses H which cover only true goal atoms and hypotheses H which cover each ground substitution, we will give the definitions for correctness (i.e., posterior satisfiability), sufficiency, and completeness.

The definition of *posterior satisfiability* (i.e., correctness) is modified in accordance to our different view on an example.

Def. 11 (cf. Correctness)

A goal clause $(p(x_1,..., x_p) \leftarrow P)$ is called correct if no ground substitution θ exists such that $P\theta$ is true in $M^+(B)$ for a negative example θ (i.e., $\neg p(x_1,..., x_p)\theta \in E^-$). A hypothesis H is called *correct* if H contains only correct goal clauses.

Def. 12 (cf. Sufficiency)

A hypothesis H is *sufficient* if $M^+(B \cup H) \mid\approx E^+$.

6. A minimal Herbrand model H entails a positive ground atom l if $l \in H$, a negative ground atom $\neg l$ if $l \notin H$, and an universally-quantified formulae γ if all ground instantiations of γ can be entailed by H.

Def. 13 **(cf. Completeness)**

A hypothesis H is *complete (in the first-order sense)* if every goal clause ϕ which is true in $M^+(B) \cup E^+$ is also true in $M^+(B \cup H)$ (i.e., $M^+(B \cup H) \models\approx \phi$).

A hypothesis H is called *complete (in the propositional sense)* if H covers all positive examples of E. H *covers* the positive example θ (i.e., $p(x_1,..., x_p)\theta \in E^+$) if and only if a clause $h \in H$ exists, with $h = (p(x_1,..., x_p) \leftarrow P)$, such that $P\theta$ is true in $M^+(B)$.[7]

We will show that completeness in the first-order sense and in the propositional sense are equivalent. Note that not only $p(x_1\theta,..., x_p\theta)$ is an example (as it is considered in FOIL), but also the whole substitution θ. The following example illustrates that only considering all substitutions as examples can allow us to describe (logical) completeness in a first-order sense by propositional means via *covering an example*. Consider the following background and example knowledge

$$M^+(B) = \{b_1(a,a), b_2(a,b), b_3(a,a)\}$$
$$E^+ = \{p(a)\},\ E^- = \{\neg p(b)\}.$$

The propositional representations for two variables is given in Table 3.

Table 3. A brief example of completeness and sufficiency.

(x,y)	$p\ (x)$	$b_1(x,y)$	$b_1(y,x)$	$b_2(x,y)$	$b_2(y,x)$	$b_3(x,y)$	$b_3(y,x)$
(a,a)	+	1	1	0	0	1	1
(a,b)	+	0	0	1	0	0	0
(b,a)	–	0	0	0	1	0	0
(b,b)	–	0	0	0	0	0	0

For instance, in regard to two variables x,y, the following clause (1) covers each true ground atom of the target predicate (= $\{p(a)\}$) and thus fulfils the sufficiency condition:

$$p(x) \leftarrow b_1(x,y) \tag{1}$$

On the other hand, the background knowledge and evidence allow two different ground substitutions:

$$\theta_1 = \{(x/a), (y/a)\} \text{ and } \theta_2 = \{(x/a), (y/b)\}$$

with $p(x)\theta_1 \in E^+$ and $p(x)\theta_2 \in E^+$. Clause (1) covers θ_1, but not θ_2. The following set of clauses $H = \{(1), (2)\}$ cover all substitutions θ with $p(x)\theta \in E^+$ and thus is complete in the propositional and—as we prove—in the first-order sense:

$$p(x) \leftarrow b_1(x,y) \tag{1}$$
$$p(x) \leftarrow b_2(x,y) \tag{2}$$

Hence a system based on the sufficiency condition cannot learn both clauses. In this case, *not all information expressed by the number of variables and each possible substitution is used for the learning process*. The information provided by the second substitution is neglected if the system only learns clauses which cover $p(a)$, since clause (1) already covers this ground fact.

To illustrate first-order completeness we mention the third clause

7. Except for the different treatment of examples, our definition of *covering* is the same as in [RLD93].

$$p(x) \leftarrow b_3(x,y) \tag{3}$$

which is true in $M^+(B') \cup E^+$ but cannot be derived from $H = \{(1), (2)\}$ only. Otherwise, with $M^+(B') \mid\approx b_1(x,y) \leftarrow b_3(x,y)$ holds $M^+(B' \cup H) \mid\approx (3)$. Therefore our completeness result hold only for $M^+(B' \cup H)$ as we do not generalize the background knowledge B.

Theorem Let $M^+(B)$ be the background knowledge, E be the evidence, with $M^+(B) \cup E$ is separable, and H is a set of correct goal clauses.

H is complete in a propositional sense with respect to B and E.	$\Leftrightarrow$	H is complete in a first-order sense with respect to B and E.

Proof

First we prove the "$\Rightarrow$" direction. For each substitution θ with $p(x_1,...,x_p)\,\theta \in E^+$ there exists a clause $h :=(p(x_1,...,x_p) \leftarrow P_h) \in H$ with $P_h\theta$ being true in $M^+(B)$. Let $g := (p(x_1,...,x_p) \leftarrow P_g)$ be a goal clause which is true in $M^+(B) \cup E^+$. We then construct an induction proof with respect to the number of premises in g. Starting from the maximal number of literals of a most specific premise (cf. Def. 8) n we conclude to n-1.

Induction start:[8] Let P_g be a non-ground most-specific premise (the number of literals in P_g is maximal, each non-ground literal—except for the target—occurs exactly once, negated or not). If g is non-trivially true in $M^+(B) \cup E^+$, a ground substitution θ exists such that $P_g\theta$ is true in $M^+(B)$, hence $p(x_1,...,x_p)\theta \in E^+$.[9] As a consequence of the assumption a goal clause $h :=(p(x_1,...,x_p) \leftarrow P_h) \in H$ exists which covers θ (i.e., $P_h\theta$ is true in $M^+(B)$). As P_g is most-specific, it must hold true that: $P_h \subseteq P_g$. Therefore $h \mid\approx g$.

Induction step: We have established that for each non-ground goal clause $g := (p(x_1,...,x_p) \leftarrow P_g)$ which is true in $M^+(B) \cup E^+$ and $|P_g| = n$ it follows that $H \mid\approx g$. It remains to show that for each non-ground goal clause $\bar{g} := (p(x_1,...,x_p) \leftarrow P_{\bar{g}})$ which is non-trivially true in $M^+(B) \cup E^+$ with $|\,p_{\bar{g}}| = n - 1$ it also holds true that $M^+(B \cup H) \mid\approx \bar{g}$.

Let θ be a ground substitution with $P_{\bar{g}}\theta$ being true in $M^+(B)$. Furthermore let l be a literal which is not in $p_{\bar{g}}$. W.l.o.g. let $l\theta$ be true in $M^+(B)$, which implies that $(p(x_1,...,x_p) \leftarrow P_{\bar{g}} \wedge l)$ is non-trivially true in $M^+(B) \cup E^+$. Hence with the induction start it follows that

$$M^+(B \cup H) \mid\approx (p(x_1,...,x_p) \leftarrow P_{\bar{g}} \wedge l). \tag{1}$$

Case 1: Assume that for all substitutions $\bar{\theta}$ with $P_{\bar{g}}\bar{\theta}$ being true in $M^+(B)$ $l\bar{\theta}$ is also true in $M^+(B)$. Then $(l \leftarrow P_{\bar{g}})$ is true in $M^+(B)$. Together with (1) it follows that $M^+(B \cup \mathrm{H}) \mid\approx \bar{g}$.

8. One can see from the induction start that each most-specific goal clause can already derived from H. That is, the background knowledge $M^+(B)$ is not required. Actually this follows immediately from the propositional completeness.

9. If such a substitution does not exist then it follows immediately that $M^+(B) \mid\approx g$.

Case 2: Assume that a ground substitution $\overline{\theta}$ exists, with $P_{\overline{g}}\overline{\theta}$ being true and $l\overline{\theta}$ being false in $M^+(B)$. Then $(p(x_1,...,x_p) \leftarrow P_{\overline{g}} \wedge \neg l)$ is true in $M^+(B)$ and we have with the induction start:

$$M^+(B \cup H) \mid\approx \left\{ \begin{array}{c} p(x_1,...,x_p) \leftarrow P_{\overline{g}} \wedge \neg l \\ \\ p(x_1,...,x_p) \leftarrow P_{\overline{g}} \wedge l \end{array} \right\} \mid\approx (p(x_1,...,x_p) \leftarrow P_{\overline{g}}) = \overline{g}.$$

Now we show "$\Leftarrow$": Let θ be a positive example and P_θ the set of all literals with $l\theta$ being true in $M^+(B)$. Then the conjunction of P_θ is a most-specific premise. Because of the separability of E it holds true that: $(p(x_1,...,x_p) \leftarrow P_\theta)$ is true in $M^+(B) \cup E^+$, and by our assumption $M^+(B \cup H) \mid\approx (p(x_1,...,x_p) \leftarrow P_\theta)$. This implies that H covers the positive example θ.

It should be clear that the proof holds only for finite case. That is, we require finite perfect Herbrand models for our background and example knowledge.

Corollary 1 First-order completeness implies sufficiency.

As completeness in the first-order is equivalent to completeness in the propositional sense it becomes clear that $M^+(B \cup H) \mid\approx \phi$ for all true goal clauses in $M^+(B) \cup E^+$ implies that $M^+(B \cup H) \mid\approx E^+$.

In the definition of completeness in Def. 13, which will be called *weak completeness* in the following, is enforced that each correct goal clause which is ***true*** in $M^+(B) \cup E^+$ must also be true in $M^+(B \cup H)$. A stronger version of completeness can be achieved by including all correct goal clauses. That is, all goal clauses which are ***not false*** according to $M^+(B)$ and E^-. This *strong completeness* is formalized by the following definition:

Def. 14 **(cf. Strong completeness)**

A hypothesis H is *strongly complete (in the first-order sense)* if every correct goal clause ϕ is true in $M^+(B \cup H)$. That is, $M^+(B \cup H) \mid\approx \phi$.

A hypothesis H is called *strong complete (in the propositional sense*) if H covers all positive examples and unknown cases of E (i.e., all θ with $\neg p(x_1,..., x_p)\theta \notin E^-$).
H covers the positive example or the unknown case θ if and only if a clause $h \in H$ exists, with $h = (p(x_1,..., x_p) \leftarrow P)$, such that $P\theta$ is true in $M^+(B)$.

The equivalence proof of the propositional and first-order version of strong completeness follows the above given proof. To achieve strong completeness all variable substitutions θ which do not lead to a negative example must be covered by H. Actually, strong completeness corresponds to the definition given in [Hel89]. Strong and weak completeness are equivalent if the closed-world assumption applies to E.

1.4 Learning Task

Finally, we want to characterize the learning task. We do not regard each complete and correct hypothesis as a solution as we prefer nonredundant hypotheses. We require that each clause of H be *most-general* (i.e., that their premises contain a minimal set of literals) and that H be *minimal* (i.e., that it does not contain unnecessary goal clauses). The following definition remains decidable as only finitely many goal clauses can be expressed in our language (cf. Def. 1, Def. 3, and Def. 6).

Def. 15 (cf. Solution)

A goal clause ϕ is *most-general* iff for each goal clause ϕ' with $\phi' \models \phi$ and $\phi \not\models \phi'$, $H = \{\phi'\}$ is not a correct hypothesis. A correct and complete hypothesis H is most-general if each $h \in H$ is most-general.
A correct and complete hypothesis H is *minimal* if no $h \in H$ exists with $H \setminus \{h\}$ being complete.
A correct and complete hypothesis H which is most-general and minimal is called a *solution*.

1.5 Recursive Goal Clauses

A possibility which has not yet been discussed is to allow *recursive goal clauses* (i.e., goal clauses that also contain the target predicate symbol in the premise). This can be achieved by weakening Def. 3. In the recursive case, also positive target literals could be used in a premise of a goal clause. Negated target literals can not be allowed in the premise of a goal clause as this would lead to non-stratified clause sets. That is, we could no longer use our minimal model semantics which uses stratification to select one unique model (called the perfect Herbrand model) from all minimal Herbrand models.

As we will assume the closed-world assumption for the predicates used in premises this implies that we have to apply the closed-world assumption for E in the recursive case. That is,

$$E^- = \{\neg p(t_1,...,t_p) \mid p(t_1,...,t_p) \in \text{Herbrand base} \setminus E^+\}.$$

The transition to the propositional representation is achieved by introducing additional columns in the table using non-ground atoms of the target predicate. But one has to choose carefully the selected atoms (i.e., the language bias for goal clauses, compare [QuC93]) to prevent the learner from producing goal clauses like:

$$p(x_1,...,x_p) \leftarrow p(x_1,...,x_p) \text{ or } p(x_1,...,x_p) \leftarrow p(y_1,...,y_p) \wedge =(x_1,y_1) \ ... \wedge =(x_p,y_p).$$

[Ber93] reports some problems of—what he calls—*extensional* learning methods like FOIL or LINUS for learning recursive clauses. Their results need neither be complete nor correct in a logical sense (i.e., in an *intensional* sense). Thus, either a correct ground target atom cannot or an incorrect ground target atom can be entailed by H. Similar problems arise for our approach.

The following definitions introduces the notion of propositional completeness for recursive goal clauses. There is no difference between what was called weak or strong completeness as we apply the closed-world assumption to E.

Def. 16 (cf. Propositional completeness for recursive hypotheses)

A recursive hypothesis H is called *complete (in the propositional sense)* if H covers all positive examples of E. *H covers* the positive example θ (i.e., $p(x_1,..., x_p)\theta \in E^+$) if and only if a clause $h \in H$ exists, with $h = (p(x_1,..., x_p) \leftarrow P)$, such that $P\theta$ is true in $M^+(B) \cup \boldsymbol{E}^+$.

But this kind of completeness does no longer ensure first-order completeness as defined in the earlier sections. We will illustrate this by a small example.

$$\mathrm{M}^{+}(B) = \{q(a), r(b)\}$$
$$E^{+} = \{p(a)\}.$$

The propositional representations for one variable is given in Table 4.

Table 4. A small example for completeness an sufficiency.

(x,y)	$p\ (x)$	$p(x)$	$q(x)$	$r(x)$
(a)	+	1	1	0
(b)	–	0	0	1

For instance, the following clause is correct and complete in the propositional sense

$$p(x) \leftarrow p(x)$$

but $\mathrm{M}^{+}(B \cup H) \mid\neq p(x) \leftarrow q(x)$. Actually, H is not even sufficient as $p(a)$ can no longer be derived from it. As a result, in the recursive case Frog learns correct goal clauses but can neither guarantee sufficiency nor completeness. Still, it can find interesting result as shown in section 3. The incompleteness is not a result of using a propositional search space (compare [Zic91] who does not have this problem). It is caused by the fact that the distinction of evidence and background knowledge is abused in the recursive case. On the one hand, the evidence E is used as background knowledge (i.e., $\mathrm{M}^{+}(B) \cup \boldsymbol{E}^{+}$ is used) during learning clauses. On the other hand, it is no longer available in $\mathrm{M}^{+}(B)$ which causes the incompleteness of $\mathrm{M}^{+}(B) \cup H$.

2 Frog: A Procedural Description of Our Approach

After describing *what* we want to have we will now discuss *how* this can be realized by a heuristic search procedure. In fact, we will briefly sketch the propositional learning algorithms *JoJo* ([Fen93a], [FeW93]) and *Frog* [Fen93b].

2.1 Frog in a Nutshell

Finding a hypothesis H which describes a set of examples in a correct, complete, and minimal manner can be viewed as a search problem. The *version space algorithm* [Mit81] is a classical algorithm which learns a set of goal clauses H from a set of examples. It uses a *dual* search strategy including generalization and specialization. As it uses an *exhaustive search* it can only be applied to small data sets. A lot of effort has been spent in the last few years to develop *heuristic* search procedures for this task. Two classes of these algorithms can be recognized: Algorithms which use *specialization* as search direction (for example, AQ [MMH+86], ASSISTANT [Ces87], FOIL [Qui90a], and LINUS [LDG91]) and algorithms which use *generalization* as search direction (for example, CIGOL [MuB88], CLINT [Rae92], GOLEM [MuF90]).

Because one generally cannot determine which search direction is better, we developed the procedure JoJo (cf. [Fen93a]), which integrates generalization and specialization into one bidirectional search process. The algorithm can start at an arbitrary point in the lattice of goal clauses and can generalize and specialize as long as the quality or correctness of the goal clauses regarded can be improved, i.e., until a local optimum can be found, or the search resources (e.g., computation time) are consumed. Because JoJo integrates both search directions, two main advantages are achieved:

- Depending on the preference determined by domain- and task-specific circumstances, the procedure can utilize the advantages of both search strategies in a dynamic manner.
- The search procedure can use knowledge about meaningful starting points for the search. As JoJo can use an arbitrary starting point (i.e., an arbitrary goal clause) for its search process it can naturally work in *an incremental mode* by using the goal clauses which should be refined as starting points for its new search process.

An evaluation of JoJo and its extension into a four-step procedure for refining, completing, reducing, and minimizing a set of goal clauses according to new examples is given in [FeW93] and [Wie93].

JoJo generalizes the heuristic search by integrating specialization and generalization into one bidirectional search process. But JoJo inherits a strong restriction on its search strategy from its predecessors. It can add or relax just one premise per step. An extended strategy is to add or delete several premises in one step. This search strategy as used by Frog allows jumping through the search space. Frog evaluates several goal clauses in the environment of the current goal clause and selects one (or several in the case of a beam search) of them as the successor goal clause. The size and location of the environment where Frog is searching for successor goal clauses depends on the quality of a current goal clause and the allowed search effort per step. In Figure 1 we sketch the Frog algorithm using pseudo code. Two possibilities exist for deriving a successor goal clause from a current one:

- A current goal clause covers too many negative examples and must be *specialized*. The user has to specify a threshold for the allowed ratio of the negative examples and all examples covered by a goal clause. A successor goal clause is derived by adding some premises to the current goal clause.
- A current goal clause is correct but should be *generalized*. A successor goal clause is derived by deleting some literals of the body of the current goal clause.

2.2 Implementation of the Learning Environment

The implementation of JoJo and its environment RJ for preprocessing data and testing learning results is described in [FKN93]. The learning environment is implemented in C and available for UNIX and MS-DOS. The learning environment consists primarily of three different components: the *preprocessing component*, the *learning component*, and the *postprocessing component*. Examples given as ASCII data are converted into binary data by the *preprocessing element*. This is done to improve the efficiency of the learning algorithms. In addition, continuous and multi-valued attributes are preprocessed and the data can be divided into a test and a training data set. The *learning component* implements the RELAX, B-RELAX, JoJo and Frog algorithms. Each of them generates a set of goal clauses. The binary coded goal clauses are transferred into text data by the *postprocessing routine*. The postprocessing component also tests the goal clauses against the given examples. Applications and evaluations of RJ/JoJo can be found in [FKN93], [FeW93], [FGS93], and [Wie93].

In order to apply Frog to problems specified in first-order logic, a transition from this representation into propositional logic has to be implemented. A simple realization of this transition was to implement an extension of the preprocessing component which

```
current goal clause := init();[1]
while current total search amount < total search amount threshold
do
        successor goal clause := current goal clause
        initialize current search amount
        while current search amount < search amount threshold
        do
              compute a new successor goal clause of the current goal clause
              if quality(successor goal clause) < quality(new successor goal clause)
              then successor goal clause := new successor goal clause endif
              increase current search amount
        enddo
        current goal clause := successor goal clause
        increase total search amount
enddo
```

1. In this paper we do not discuss how an initial starting point is derived (see [Fen93b] for more details).

Figure 1. The Frog algorithm

translated the first-order representation into a table format as shown in Table 2. This table can be further processed by the preprocessing component of RJ which has already been implemented.

On the other hand our current implementation is very inefficient concerning storage amount and computational effort. A re-implementation derives explicitly only all ground substitutions which are positive or negative example. Similar to FOIL partial representations of the search space (i.e., the premises of the goal clauses) can be constructed during the heuristic search process without losing the completeness of the learning result. In addition it turn out that several steps of the current implementation which check whether a possible goal clause covers a substitution can be drastically optimized. Therefore, we expect a significant improvement of performance.

Still, the effort of the learning task increases expotentially with the number of variables. Therefore, the scope of our approach is limited to learning problems with small number of variables per clause. The user of Frog can use the number of variables as a language bias to ensure tractability as the number of variables are fixed before the learning process. Actually, he could start the learning process with small numbers of variables. If the learned results are not sufficient, he could stepwise increase the number of variables.

3 An Evaluation of Our Approach

We use examples from [Qui90a] to illustrate and evaluate our approach. In the following we present the results for a *small network*, for *learning an arch definition*, for learning *recursive relations of lists*, and, finally, for learning *family relationships*.

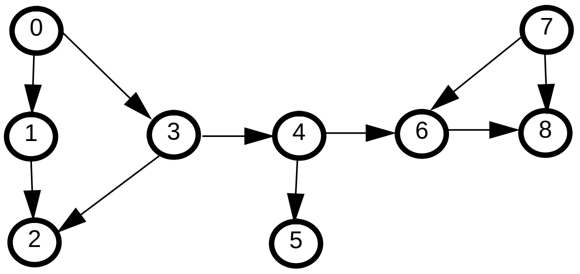

Figure 2. A small network.

3.1 A Small Network

Given is a small network as shown in Figure 2. The language for describing the learning problem consists of nine constants C = {0,1,2,3,4,5,6,7,8} (the nodes) and two predicate symbols *linked-to* and *can-reach*. *linked-to*(a,b) is true if a and b are linked and *can-reach*(a,b) is true if a and b are connected. The learning task is to compute clauses which describe the target predicate *can-reach*.

We start the search with two variables x,y using *can-reach*(x,y) as the goal and *can-reach*(y,x), *linked-to*(x,y), *linked-to*(y,x) as background literals. Table 5 shows a part of the representation of the learning problem. Frog produces clause (1) describing the target

Table 5. Search for *can-reach* with two variables.

(x,y)	*can-reach*(x,y)	*linked-to*(x,y)	*linked-to*(y,x)	*can-reach*(y,x)
(0,0)	-	0	0	0
(0,1)	+	1	0	0
...				
(8,8)	-	0	0	0

literal *can-reach*(x,y):

$$\textit{can-reach}(x,y) \leftarrow \textit{linked-to}(x,y) \tag{1}$$

This clause identifies the directly linked nodes as connected nodes. Thus the result covers 10 of 19 cases. We repeated the search with three variables x,y,z again using *can-reach*(x,y) as the target literal and *can-reach*(x,z), *can-reach*(y,x), *can-reach*(y,z), *can-reach*(z,x), *can-reach*(z,y), *linked-to*(x,y), *linked-to*(x,z), *linked-to*(y,x), *linked-to*(y,z), *linked-to*(z,x), *linked-to*(z,y) as the background literals. Frog produces the following clauses describing the target literal *can-reach*(x,y):

$$\textit{can-reach}(x,y) \leftarrow \textit{linked-to}(x,y) \tag{2}$$

$$\textit{can-reach}(x,y) \leftarrow \textit{can-reach}(x,z) \wedge \textit{can-reach}(z,y) \tag{3}$$

The search process took around 2 seconds on a SPARC-II. Clause (2) covers 90 ground substitutions θ with *can-reach*(x,y)$\theta \in E^+$ and clause (3) covers 16. Both clauses together cover 105. Thus one substitution is shared by both clauses. Both clauses are correct. The solution is very similar to FOIL´s solution (see [Qui90a]) which also consists of two clauses. The first clause is identical but the second one differs in the first premise. FOIL learned the following second clause:

$$\textit{can-reach}(x,y) \leftarrow \textit{linked-to}(x,z) \wedge \textit{can-reach}(z,y) \tag{3´}$$

Clause (3) as found by Frog is a generalization of the clause (3´) as *linked-to*(x,z) |= *can-reach*(x,z). Frog finds clause (3) instead of (3´) because five substitutions are cover by (3) but not by (3´). An example for such a substitution is $\theta = \{(x/0),(y/8),(z/4)\}$. The following statements hold true:

> *can-reach*(x,y)θ = *can-reach*(0,8) $\in E^+$, *can-reach*(x,z)θ = *can-reach*(0,4) $\in \mathrm{M}^+(B)$,
> *can-reach*(z,y)θ = *can-reach*(4,8) $\in \mathrm{M}^+(B)$

On the other hand: *linked-to*(x,z)θ = *linked-to*(0,4) $\notin \mathrm{M}^+(B)$.

Taking a closer look at our solution it becomes visible that 66 positive examples remain uncovered by $H = \{(2),(3)\}$. Frog cannot find correct goal clauses for them because the description of our problem is not separable. For example, $\theta' = \{(x/0),(y/2),(z/4)\}$ and $\theta'' = \{(x/0),(y/7),(z/2)\}$ are two ground substitutions which cannot be separated with the given background predicates and

$$\textit{can-reach}(x,y)\theta' \in E^+ \text{ versus } \neg\textit{can-reach}(x,y)\theta'' \in E^-.$$

If we allow goal clauses with some relative error—that is, if we apply the possibility of our propositional learner to deal with *noise*—we get an additional goal clause:

$$\textit{can-reach}(x,y) \leftarrow \textit{can-reach}(x,z) \wedge \textit{can-reach}(y,z) \wedge \neg\textit{linked-to}(x,z). \quad (4)$$

This clause covers 38 positive examples (13 of them are already covered by {(2),(3)}) but also 9 negative examples. Its accuracy is therefore 80%.

3.2 Learning Definitions For Arches

Winston introduced the task of learning *definitions for arches* from positive and negative examples for them. We applied Frog to the target predicate *Arch* with three variables and it took 6.2 seconds on a SPARC-II to come up with the following clause:

$$\textit{Arch}(x,y,z) \leftarrow \textit{Left-of}(y,z) \wedge \textit{Supports}(z,x) \wedge \neg\, \textit{Touches}(y,z)$$

This is also the clause learned by FOIL and it covers the two positive examples and none of the negative examples.

3.3 Learning Recursive Relations On Lists

The example for learning *recursive relations on lists* provides the following input facts:

> *list*(()), *list*((*a*)), *list*((*b*(*a*)*d*)), *list*(((*a*)*d*)), *list*((*d*)),
> *null*(())
> *components*((*a*),*a*,()), *components*((*b*(*a*)*d*),*b*,((*a*)*d*)), *components*(((*a*)*d*),(*a*),(*d*)), *components*((*d*),*d*,()), *components*((*e*.*f*),*e*,*f*)

As we do not provide function symbols, each different list is treated as a different constant. We applied Frog to the target predicate *list* with three variables and it took 10.1 seconds on a SPARC-II to come up with the following three clauses:

$$\textit{list}(x) \leftarrow \textit{null}(x) \quad (1)$$

$$\textit{list}(x) \leftarrow \textit{components}(z,y,x) \wedge \textit{list}(z) \quad (2)$$

$$\textit{list}(x) \leftarrow \textit{components}(x,z,y) \wedge \textit{list}(y) \quad (3)$$

Compared to FOIL, an additional clause (3) is derived since {(1),(2)} do not cover 4 of the 129 positive examples.

3.4 Learning Family Relationships

The example for *family relationships* provides the twelve predicates:

> *Aunt, Brother, Daughter, Father, Husband, Mother, Nephew, Niece, Sister, Son, Uncle, Wife*

and uses 24 constants to define 104 facts about family relationships. In the case of three variables, the translation into a propositional representations produces 13,824 examples described by 71 attributes. We applied Frog to the target predicate *Aunt* using the ground atoms of the other predicate symbols as background knowledge. It took around one minutes on a SPARC-II to get the following two clauses:

$$Aunt(x,y) \leftarrow Nephew(y,x) \wedge \neg Uncle(x,y) \tag{1}$$

$$Aunt(x,y) \leftarrow Niece(y,x) \wedge \neg Uncle(x,y) \tag{2}$$

Both cover 96 positive examples and no negative examples.

4 Conclusions and Related Work

In the following we compare Frog with related approaches and identify tasks for future research.

4.1 Comparison to Related Work

FOIL (cf. [Qui90a], [QuC93]), LINUS (cf. [LDG91], [DzL93], [LDG91]), and MILP [Kov94] are approaches closely related to Frog, as they use a propositional learning space for learning goal clauses of a restricted first-order language. Two main distinctions exists between them and Frog. First, we extend their concept of *sufficiency* to include (logical) *completeness* of the learned solution. A learned hypothesis H contains all knowledge about the target predicate which can be expressed in the restricted first-order language. This is not the case when using the sufficiency condition as used by FOIL and LINUS. Normally, one would assume that the result of a learning step should enrich the knowledge of the learner. But using the sufficiency criterion of FOIL, $M^+(B \cup H)$ is weaker than $M^+(B) \cup E^+$. Thus possible goal clauses

$$p(x_1,..., x_p) \leftarrow l_1 \wedge ... \wedge l_m$$

exist which are a logical consequence of $M^+(B) \cup E^+$ but which are no longer a consequence of $M^+(B \cup H)$. Therefore, learning by using LINUS and FOIL is—although not explicitly mentioned—learning by forgetting.

Second, the search strategy of Frog is much more flexible than the top-down search strategy of FOIL and LINUS. The search strategy can leave a local optimum and Frog can start at arbitrary points in the search space, which easily allows for an incremental and reversible learning strategy. Because MILP uses simulated annealing, it is close in spirit to Frog—as it can also leave a local optimum—but the search is still viewed only as a specialization process. Frog can use knowledge about meaningful starting points for the search process, domain- and task-specific preferences for specialization and generalization as search directions and extend search strategies which only regard direct successors in the search space.

CLAUDIEN [RaB93] does not use a propositional search space, but the completeness notion of our approach is similar to its nonmonotonic semantics setting. Despite this similarity, there are some differences between CLAUDIEN and our approach.

CLAUDIEN cannot deal with incomplete evidence (i.e., partial interpretations) like Frog can because it applies the closed-world assumption to the entire input knowledge. Frog distinguishes between complete background knowledge and incomplete evidence about the target predicate.

CLAUDIEN deals with multiple-predicate learning whereas Frog only learns goal clauses for a single target predicate. The restriction to single-predicate learning is not a result of our theoretical framework, which maps first-order representations onto propositional representations via ground substitutions (see [Zic91] which applies it to multiple-predicate learning), but it is inherited by our learning procedure, Frog, which is only designed for learning goal clauses.

[Zic91] and [GaZ94] describe a learning system for definite clauses with finite minimal Herbrand models, called *rule exploration*, which is close in spirit to CLAUDIEN. As in the nonmonotonic setting, they do not distinguish between background knowledge and evidence and learn a set of clauses H for which:

$$M^+(B) \Leftrightarrow M^+(H)$$

but H is a compact description (i.e., a minimal description by most-general definite clauses) of B.[10] As is the case with CLAUDIEN they apply their algorithm to the multiple-predicate learning task. Until now, no comparison of these two approaches can be found in the literature.

4.2 Directions of Future Work

First, we already mentioned that we implement a partial and dynamic construction of the search space. Currently the entire search space is statically represented by a complete set of maximally-specific clauses. In addition to the dynamic expansion of the search space, the search itself has to be improved by using the symmetry within the search space in respect to variable permutation (see [Zic91]). The propositional representations of syntactically different clauses like $P(x) \leftarrow Q(x,y)$ and $P(x) \leftarrow Q(x,z)$ differ, although the clauses have the same semantics.

Second, we want to enrich the expressive power of our representation language. The omission of function symbols is a strong limitation of our language. Enriching the language could be done by allowing the use of function symbols. The absence of function symbols is only a sufficient but not a necessary condition for finiteness. In fact, even with the use of function symbols it is possible to ensure finite minimal models by defining rules for the use of function symbols and recursion and for the use of variables in the body and head of a clause (range restriction) [Ull89].

Acknowledgement

We thank Jörg-Uwe Kietz and several anonymous reviewers from ILP´94 and ECML´95 for very helpful comments and Jeffrey Butler for correcting our English.

10. This approach is based on the *Formal Concept Analysis* (cf. [Wil89]).

References

[Ber93] F. Bergadano: Inductive Database Relations. In *IEEE Transactions on Knowledge and Data Engineering*, vol 5, no 6, December 1993.

[Ces87] B. Cestnik: *ASSISTANT PROFESSIONAL, A Software Tool for Inductive Learning of Decision Rules*, system user manual, Edvard Kardelj University, Ljubljana, Slovenia, 1987.

[Cla78] K. L. Clark: Negation as Failure. In H. Gallaire et al. (eds.), Logic and Data Bases, Plenum Press, New York, 1978.

[DzL93] S. Dzeroski and N. Lavrac: Inductive Learning in Deductive Databases. In *IEEE Transactions on Knowledge and Data Engineering*, vol 5, no 6, December 1993.

[Fen93a] D. Fensel: JoJo: Integration of Generalization and Specialization. In *Proceedings of the Workshop Knowledge and Data Engineering*, Atelier d´Ingenierie des Connaissances et des Donees, A.I.C.D., Strasbourg, France, January 25-27, 1993.

[Fen93b] D. Fensel: RELAX, JoJo, and Frog: Step by Step Generalization of Search Strategies in Applied Machine Learning. In research report, no 279, Institut AIFB, University of Karlsruhe, 1993.

[FeW93] D. Fensel und M. Wiese: Incremental Refinement of Rule Sets with JoJo. In *Proceedings of the European Conference on Machine Learing ECML-93*, Vienna, Austria, April 5-8, 1993, Lecture Notes in Artificial Intelligence, no 667, Springer-Verlag, Berlin, 1993.

[FGS93] D. Fensel, U. Gappa, and S. Schewe: Applying a Machine Learning Algorithm within a Knowledge Acquisition Scenario. In *Proceedings of the Workshop Knowledge Acquisition and Machine Learning at the IJCAI´93*, Chambery, France, August 30th, 1993.

[FKN93] D. Fensel, J. Klein, and U. Neubronner: RJ: An Environment for Learning from Example. In *Proceedings of the 13th International Conference Expert Systems and Their Applications*, 24-28 Mai, Avignon, 1993.

[GaZ94] B. Ganter and M. Zickwolff: A Tool to Acquire Conceptual Knowledge. In D. Fensel et al. (eds.), *Maschinelles Lernen: Theoretische Ansätze und Anwendungen* (Machine Learning: Theoretical Approaches and Applications), Proceedings of the Workshop Machine Learning at the 17. Fachtagung für Künstliche Intelligenz (KI-93), Berlin, September 13-16, 1993, research report, no 298, Institut AIFB, University of Karlsruhe, 1994.

[Hel89] N. Helft: Induction as Nonmonotonic Inference. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[Kov94] M. Kovacic: MILP-a Stochastic Approach to Inductive Logic Programming. In *Proceedings of the 4th International Workshop on Inductive Logic Programming (ILP-94)*, Bonn, Germany, September 12-14, 1994.

[LaG94] N. Lavrac and S. Dzeroski: Weakening the Language Bias in Linus. In *Journal of Experimental & Theoretical Aritificial Intelligence*, vol 6, no 1, 1994.

[LDG91] N. Lavrac, S. Dzeroski, and M. Grobelnik: Learning of Nonrecursive Definitions of Relations with Linus. In Y. Kodratoff (ed.), *Proceedings of the European Working Session on Learning (EWSL)*´91, Porto, Portugal, March 6-8, 1991, Lecture Notes in Artificial Intelligence, no 482, Springer-Verlag, Berlin, 1991, pp. 265-281.

[Llo87] J. W. Lloyd: *Foundations of Logic Programming,* Springer-Verlag, 2nd edition, Berlin, 1987.

[Mit81] T.M. Mitchell: Generalization as Search, B. Webber et al. (eds). In *Readings in Artificial Intelligence,* Tioga Publishinh Co., Palo Alto, 1981.

[MMH+86] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac: The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. IN *Proceedings of the 5th National Conference on AI (AAAI-86),* Philadelphia, August 11-15, 1986, pp. 1041-1045.

[MuB88] S. Muggleton and W. Buntine: Machine Invention of First-Order Predicate by Inverting Resolution. In *Proceedings of the 5th International Conference on Machine Learning (ICML´88)*, 1988.

[MuF90] S. Muggleton and C. Feng: Efficient Induction of Logic Programs. In *Proceedings of the Workshop on Algorithmic Learning Theory (ALT´90),* Tokyo, October 8-10, 1990.

[MuR94] S. Muggleton and L. De Raedt: Inductive Logic Programming: Theory and Methods. In *Journal of Logic Programming*, vol 19/20, May/July 1994.

[QuC93] J. R. Quinlan and R. M. Cameron-Jones: FOIL: A Midterm Report. In *Proceedings of the European Conference on Machine Learning, Machine Learning: ECML-93*, Vienna, Austria, April 5-7, 1993, P. B. Brazdiol (ed.), Springer-Verlag, Lecture Notes in Artificial Intelligence, no 667, 1993.

[Qui84] J.R. Quinlan: Learning Efficient Classification Procedures and Their Application to Chess End Games. In R.S. Michalski et al. (eds.), *Machine Learning. An Artificial Intelligence Approach, vol.1,* Springer-Verlag, Berlin, 1984, pp. 463-482.

[Qui90a] J. R. Quinlan: Learning Logical Definitions from Relations. In *Machine Learning*, vol 5, no 3, 1990, pp. 239-266.

[RaB93] L. De Raedt and M. Bruynooghe: A Theory of Clausal Discovery. In *Proceedings of the 13th International Joint Conference on Aritificial Intelligence (IJCAI´93)*, Chambery, France, 28 August - 3 September, 1993.

[Rae92] L. De Raedt: *Interactive Theory Revision: An Inductive Logic Programming Approach*, Academic Press, 1992.

[RLD93] L. De Raedt, N. Lavrac, and S. Dzeroski: Multiple Predicate Learning. In *Proceedings of the 13th International Joint Conference on Aritificial Intelligence (IJCAI´93)*, Chambery, France, 28 August - 3 September, 1993.

[RoP89] C. Rouveirol an J.-F. Puget: A Simple Solution For Inverting Resolution. In K. Morik (ed.), *Proceedings of the 4th European Working Session on Learning (EWSL-89),* pp. 201-210, Pitman, 1989.

[Rou94] C. Rouveirol: Flattening and Saturation: Two Representation Changes For Generalization. In *Machine Learning*, vol 14, 1994, pp. 219-232.

[Ull88] J.D. Ullman: *Database and Knowledge-Base Systems, vol. 1,* Computer Science Press, New York, 1988.

[Wie93] M. Wiese: JoJo: *Integration von Generalisierung und Spezialisierung in ein heuristisches Verfahren zum maschinellen Lernen von Regeln aus Beispielen*, master thesis, Institut AIFB, University of Karlsruhe, 1993.

[Wil89] R. Wille: Knowledge Acquisition by Methods of Formal Concept Analysis. In E. Diday (ed.), Data Analysis, Learning Symbolic and Numerical Knowledge, Nova Science Publ., New-York, 1989.

[Zic91] M. Zickwolff: *Rule Exploration: First Order Logic in Formal Concept Analysis*, PhD thesis, University of Darmstadt, 1991.

Appendix - Open-world Assumption Applied to the Background Knowledge

For a general discussion of applying the open-world assumption to inductive logic programming compare [BeW93]. In our context, the open-world assumption applied to background knowledge implies an extended definition of background knowledge.

Def. 17 (cf. Incomplete background knowledge)

Incomplete background knowledge B is defined by two disjunct subsets $M^+(B)$ and $M^-(B)$ of the Herbrand base of $L(A)$ containing only background predicate symbols.

The transformation from a first-order representation to a propositional representation translates each ground literal to an attribute having the values true (1), false (0), and unknown (?). Unknown facts correspond to *unknown* as attribute value. As propositional learner are able to deal with data sets containing unknown values it is no surprise that a propositional learner can also deal with incomplete background knowledge. Different possibilities exist to interpret the attribute value unknown (i.e., as a missing value, as an irrelevant value etc., cf. [Qui89]). Similar, we get different possibilities to define correctness of goal clauses and to define completeness. In the following, we sketch only one of these possibilities.

Def. 18 (cf. Satisfaction and model)

Let A be a atom, l a literal, and $M = (M^+,M^-)$ a partial Herbrand interpretation.

• $(M^+,M^-) \models A$	if $A\,\theta \in M^+$ for all ground substitutions θ
• $(M^+,M^-) \models \neg A$	if $A\theta \in M^-$
• $(M^+,M^-) \models l_1 \vee ... \vee l_n$	if $(M^+,M^-) \models l_i$ for all i=1,...,n
• $(M^+,M^-) \models A \leftarrow l_1 \wedge ... \wedge l_n$	if $A\,\theta \in M^+$ or $(M^+,M^-) \models (\neg l_1 \vee ... \vee \neg l_n)\theta$ for all ground substitutions θ

If $(M^+,M^-) \models \phi$ holds, we say that ϕ is true in (i.e., M is a model of ϕ).

If ϕ is true in M then it is also true in all conservative extensions of M. A conservative extension M' of M changes only the truth value of unknown facts. That is, $M^+ \subseteq M'^+$ and $M^- \subseteq M'^-$.

We get the following completeness definition which hold for the produced result of Frog.

Def. 19 (cf. Correctness and Completeness)

A hypothesis H is called *correct* if H does not cover a negative examples of E. H *covers* a negative example θ (i.e., $p(x_1,..., x_p)\theta \in E^-$) if and only if a clause $h \in H$ exists, with $h = (p(x_1,..., x_p) \leftarrow P)$, such that $B \models P\theta$.

A hypothesis H is called *complete* if H covers all positive examples of E. H *covers* the positive example θ (i.e., $p(x_1,..., x_p)\theta \in E^+$) if and only if a clause $h \in H$ exists, with $h = (p(x_1,..., x_p) \leftarrow P)$, such that $B \models P\theta$.

[BeW93] S. Bell an S. Weber: *A Three-valued Logic For Inductive Logic Programming,* LS-8 Report 4, Lehrstuhl VIII, Artificial Intelligence, Universität Dortmund, Junly 1993.

[Qui89] J. R. Quinlan: Unknown Atribute Values in Induction. In *Proceedings of the 6th International Workshop on Machine Learning,* New York, June 26-27, 1989.