A three-valued logic for Inductive Logic Programming[1]

LS–8 Report 4

Siegfried Bell and Steffo Weber
Informatik VIII
University of Dortmund
D-44221 Dortmund
email: bell@ls8.informatik.uni-dortmund.de

Dortmund, July 12, 1993

[1]This report is an extended version of the presentation *On the close logical relationship between FOIL and the frameworks of Helft and Plotkin* given at the 3rd International Workshop of Inductive Logic Programming in Bled, April 1993

Abstract

Inductive Logic Programming (ILP) is closely related to Logic Programming (LP) by the name. We extract the basic differences of ILP and LP by comparing both and give definitions of the basic assumptions of their paradigms, e.g. closed world assumption, the open domain assumption and the open world assumption used in ILP.

We then define a three–valued semantic of ILP and point out relations between our semantic and the framework of Plotkin, [Plotkin, 1971], and of Helft, [Helft, 1989]. Finally, we show how FOIL, [Quinlan, 1990] fits in our work and we compare our semantic with other three-valued logics.

Contents

1 Introduction

The relationship of logic, control and algorithms can be expressed symbolically by the equation: *Algorithm = Logic + Control*, [Kowalski, 1979]. Kowalski used this equation to define that logic programs express the logical component of algorithms and logic programming (LP) is the corresponding research area, [Apt and van Emden, 1982]. Further Kowalski pointed out that LP is closely related to the closed world assumption (CWA) in contrast to the open world assumption (OWA) in classical logic, [Kowalski, 1989].

Inductive Logic Programming is a new area of research, introduced and described by S. Muggleton, [Muggleton, 1990], as *the intersection of logic programming and machine learning*. We give some understandings of this description, because there is no unique view. First ILP can be understood as programsynthesis (P. Brazdil), second as an inductive method for programmers (K. Morik[1]) to develop and verify programs or to represent knowledge and third our understanding: We emphasize the words *logic programming* and understand ILP as the intersection of the research areas of inductive logic and logic programming. In the following we will describe ILP from this point of view.

In Inductive Logic Programming there are two main frameworks, and they are proposed by [Plotkin, 1971] and [Helft, 1989]. These frameworks differ in the underlying assumptions; what they consider false, and what they consider to be still unknown. We characterize the first one with the OWA. Examples of this framework are CLINT [DeRaedt and Bruynooghe, 1989], GOLEM [Muggleton and Feng, 1992] and RDT [Kietz and Wrobel, 1991]. FOIL [Quinlan, 1990] can be used in two modes. First it can be used regarding the OWA. Second, FOIL can be used regarding the CWA. With respect to this mode, it is an example of Helft's framework. But neither Helft nor Quinlan give formal definitions of their CWA or OWA.

In the following we argue, that the CWA in ILP cannot the same as the CWA in logic programming. Therefore, we need a new definition of the CWA in ILP (CWA_{ILP}). Also we define, what we mean by a open domain assumption (ODA) and the OWA. We then formalize these settings with a modelbased three–valued interpretation of formulas in ILP taking into consideration the well known field of logic programming. Finally we show that this framework incorporates Plotkin's settings of inductive inference, that it is a logical description of FOIL and we compare our logic with other three–valued logics.

2 Logic Programming and Inductive Logic Programming

In this section we compare the semantics of logic programming (LP) and Inductive Logic Programming (ILP) to work out the differences between these two paradigms.

Table 1 contains the logic program 1 which should represent 4 *elements* and *stuff*. But what do we mean, if we write such a logic program. K. Apt, [Apt, 1990], argued that we mean in logic programming that *water* is an element, ...and *mud* is not an *element*, *water* is no *stuff*... This meaning is also listed in Table 1.

Shepherdson, [Shepherdson, 1987], claimed, that we want to deduce from the fact *son(bill,joe,jane)* and the *parent*–relation given by the logic program 2 in table 1 the negative information $\neg son(bill, harry, maude)$ and further $\neg parent(harry, maude, bill)$.

[1] Personal communication

Logic Program 1:
element(fire). element(air). element(water). element(earth). stuff(mud).

Meaning:
$\neg$ element(mud). $\neg$ stuff(water). $\neg$ stuff(earth). $\neg$ stuff(air).stuff(mud).
$\neg$ stuff(fire). element(fire). element(air). element(water). element(earth).

Logic Program 2:
parents(Father, Mother, Child) :- son(Child, Father, Mother).
parents(Father, Mother, Child) :- daughter(Child, Father, Mother).
son(bill,joe,jane). daughter(sue,harold,maude).

Meaning A:
parents(Father, Mother, Child) $\leftarrow$ son(Child, Father, Mother).
parents(Father, Mother, Child) $\leftarrow$ daughter(Child, Father, Mother).

Meaning B:
parents(Father, Mother, Child) $\leftrightarrow$
son(Child, Father, Mother) $\vee$ daughter(Child, Father, Mother).
$\neg$ son(bill,harry,maude).

Table 1: Logic Programming

This means we should interpret the logic program 2 as given by *Meaning B*. Another reading is the interpretation of rules as premises, material implication and conclusion, *Meaning A*. This reading would not permit to deduce the negative information, because we can conclude from a false premise everything. So, we mean with logic program 2, that the *parent*–relation is defined by the *son*– and the *daughter*–predicate or the *parent*–relation is equivalent to son– or daughter–relation.

Kowalski, [Kowalski, 1989], related this interpretations of logic programs to the closed world assumption in contrast to the open world assumption. Rules of logic programs should be interpreted as *if-then-and-only-if*–definitions regarding the CWA. The *if-then-and-only-if*-reading reflects our Meaning B in table 1. He defined a static and a dynamic management of a data base depending upon the form of definitions: whether data (facts or rules) is defined by means of complete *if-then-and-only-if*–definitions or by only means of the *if*–halves, whether the *only–if*–half of an *if-then-and-only-if*–definitions is stated explicitly or is assumed implicitly, and whether the *only-if* assumption is understood as a statement of the object language or as a statement of the meta-language.

The meaning of logic programs was formally captured by the closed world assumption [Reiter, 1978]. Because this solution has some computational problems, Clark proposed the Negation As Failure - rule, [Clark, 1978], whereas Apt and van Emden described semantics based on least fixpoints, [Apt and van Emden, 1982]. This corresponds to a static–meaning which is described by the following form of the CWA. It states that all

We say:		
Background Knowledge	p(c), p(a), p(b)	incomplete set
and Positive Examples:	p(a) $\rightarrow$ q(a)	of formulas
	p(b) $\rightarrow$ q(b)	
Negative Examples:	q(d)	
We mean		
Hypothesis:	p(X)$\rightarrow$ q(X)	
Prediction:	q(c)	

Table 2: Inductive Logic Programming

is true what is given and all is false, what is not given. Shepherdson summarizes in the helpful motto: *We mean, what we say and nothing more.*

Definition 1 (Closed World Assumption, CWA_{LP}**)** *Let P be a logic program. Then define*

$$CWA_{LP}(X) := P \cup \{\neg A : P \not\Vdash A\}$$

A less restrictive interpretation is used in ILP in which we *mean more, as what we say* regarding Shepherdson, or we have a dynamic representation regarding Kowalski.

Following the framework of [Plotkin, 1971], the inductive learning task in ILP can be described as using a background theory **B**, a set of positive examples E^+ and a set of negative examples E^-. The desired output is a hypothesis which describes the positive facts w.r.t. the background knowledge and no negative facts. Plotkin has proven, that there is no hypothesis which covers exactly the positive examples and no other fact. In general, we can deduce from a hypothesis, facts which are not known. Let us call this set of facts *prediction*.

Or in another sense, in learning we complete an incomplete set of data with the prediction. Table 2 shows such an incomplete set of formulas from which we want to infer a complete set of formulas. Again we distinguish in what we say and what we mean. In a semantical view, we want to infer an interpretation in which the prediction is satisfied. We do this by adding the hypothesis $p(X) \rightarrow q(X)$ which describes our intended interpretation and satisfies $q(c)$. So, we see the learning task as to infer a correct interpretation with respect to an incomplete set of formulas.

Many learning systems make special assumptions in order to infer the *correct* interpretation. For example FOIL [Quinlan, 1990] uses two assumptions, CWA and ODA. Let us recall how FOIL can be used:

1. A set of positive examples E^+ is presented to the system.
2. The set E^- will be generated according to the closed world assumption (CWA) with respect to the minimal signature of E^+.
3. The completed set E will be generalized with the background knowledge B.

Positive examples:	Relation member
	$e, [eo[x]]$
	$o, [eo[x]]$
	$[x], [eo[x]]$
	$o, [o[x]]$
	$[x], [o[x]]$
	$[x], [[x]]$
	$x, [x]$
Background knowledge	Relation $*null$ []
	Relation *components
	$[eo[x]], e, [o[x]]$
	$[o[x]], o, [[x]]$
	$[[x]], [x], []$
	$[x], x, []$
Hypotheses:	$member(A, B) : - * components(B, C, D), = (A, C)$
	$member(A, B) : - * components(B, C, D), member(A, D)$
Prediction:	$member(1, [1])$

Table 3: FOIL learns member

4. The prediction is not affected by the CWA according to the open domain assumption (ODA).

Table 3 shows positive examples of the predicate *member* and background knowledge in the form of the predicates *null* and *components*. FOIL learns two hypotheses which describes the positive examples and does not describe the negative examples. The negative examples are like $null[x]$ and are generated by the CWA. The prediction is for example $member(1, [1])$, which is not affected by the CWA, although they are not mentioned as positive examples.

First, we cannot use the CWA as defined by [Lloyd, 1987] in logic programming[2]. Lloyd and others relate the CWA to the *domain closure assumption* DCA, whereas we need a CWA which is related to the ODA.

Hence CWA_{LP} is not equal to CWA_{ILP}, but then we have to ask what is the formal definition of the CWA_{ILP} or why is the prediction not influenced by the CWA_{ILP}. Obviously, the definition of the CWA_{ILP} must allow for prediction.

One solution was presented by Helft, [Helft, 1989]. He relates the CWA to the Open Domain Assumption (ODA). Helft proposes that only a subset of the formulas according a given signature should be influenced by the CWA_{ILP}. Table 4 shows an example of Helft's framework with an given logic program. He constructs regarding the logic program a minimal model and finds then hypotheses which have to be valid in this minimal model.

[2]The CWA determines a unique model, because equality axioms, freeness axioms and the domain closure axioms restricts the domain to ground terms, and the set of formulas and the completion determines the truth values of all ground literals.

Logic Program:
deputy(tom). deputy(x) $\rightarrow$ corrupt(x).
rich(tom). rich(bill).
Minimal Model
deputy(tom),corrupt(tom),rich(tom),rich(bill)
Hypotheses:
ϕ_1: rich(x) $\leftarrow$ deputy(x)
ϕ_2: rich(x) $\leftarrow$ corrupt(x)
ϕ_3: corrupt(x) $\leftarrow$ rich(x)
$I_{CWA} \models \phi_1, \phi_2$

Table 4: Example of Helft

In the example are the rules ϕ_1 and ϕ_2 valid. Finally he applies the hypotheses regarding to the ODA.

But Helft does not give any formal definition of the CWA_{ILP} and the ODA. We give in the next section definitions of theses assumptions based on a three-valued semantic. Three–valued semantics are widely used in logic programming, see [Fitting, 1985] or [Shepherdson, 1987], but we cannot transfer their techniques identically. In logic programming three-valued semantics are motivated by the observation, that each logic program *succeeds*, *fails* or *go on forever*. So it is natural to use three values in the semantics of logic programs.

In ILP we have a similar case: each clause is *true*, *false* or *unknown*, because the prediction is a set of proposition which we not know. They are maybe true or false.

In the words of Kowalski, [Kowalski, 1989], logic programs or data bases have a static representation whereas in ILP, sets of formulas should be have a dynamic representation. This seems very natural, because in learning it should be possible to have anything to learn which is currently unknown and later known.

3 A logical framework

In this section we formalize our logical framework. We define the terms minimal signature and the three assumptions CWA_{ILP}, OWA and ODA. We then give a semantic of logic programs regarding ILP based on partial Herbrand interpretations and describe the validity of hypotheses. For logical definitions of first order language we refer to [Apt, 1990]. We restrict ourselves to the languages of Horn clause logic which are mainly used in ILP. Any undefined expressions can be found in [Lloyd, 1987] or [Apt, 1990].

3.1 Assumptions in ILP

In the following, using $\Sigma = \{PS, FS, V\}$ we denote a signature where PS is a set of predicate symbols, FS is a set of function symbols, and V is a denumerable set of variables.

FORM(Σ) denotes the set of well-formed-formulas according to Σ which are defined as usual. TERM(Σ) is defined as the set of terms. A signature is the specific part of a language.

Let us define what is meant by the term *minimal generating signature.*

Definition 2 (Generating signature) *Let* $\Sigma = \{PS, FS, V\}$ *be a signature,* X *a set of formulas. We say that* Σ generates X, *iff for every* $A \in X$, *it holds that* $A \in \mathrm{FORM}(\Sigma)$.

Definition 3 (Minimal generating signature) *Let* $\Sigma = \{PS, FS, V\}$, $\Sigma' = \{PS', FS', V'\}$ *be two signatures and* X *a set of formulas. We write* $\Sigma \subset \Sigma'$, *iff* $PS \subset PS'$, $FS \subset FS'$. *We say that* Σ *is a minimal signature w.r.t.* X, *iff*

1. Σ *generates* X
2. *for every generating signature of* X, Σ *is minimal under* $\subset$.

The point is, that the prediction should not be affected by the CWA. Thus, the completion of the given set of formulas is only done w.r.t. the minimal signature Σ. In table 3 we generalize the positive and negative examples to the hypotheses ($member(A, B) : - * components(B, C, D), \ldots$) which is also valid for numbers, e.g. $member(1, [1])$.

The idea is, that if we have a model M for the given incomplete set of formulas, then there is a model M' for the generalization. Clearly, M is not necessarily a model of theses generalized formulas because M may not interpret all instantiations of these formulas. Hence, M is only a *partial* interpretation. But partial interpretations are closely connected with a third truth-value, because one can regard the uninterpreted formulas as formulas which have the truth-value *undefined.* What happens within the generalization step is that formulas which have the truth-value *unknown* are assigned the value *true.* And the ODA permits that valid formulas remains valid.

Let us put these ideas into a more precise form. According to the way how FOIL can be used, we relate the ODA to a minimal signature and the CWA to a minimal model based on the minimal signature of a given set of formulas or a logic program.

Definition 4 (Open Domain Assumption, ODA) *Let* Σ *be a signature,* $A \in FORM(\Sigma)$ *a formula,* V *a variable,* $\sigma : V \rightarrow TERM(\Sigma)$ *a state (or variable assignment),* σ_G *a ground variable assignment. (This means in* $TERM(\Sigma)$ *are no variables.) The ODA can be stated as: if*

$$M_\Sigma \models \sigma_G(A), \textit{for every state } \sigma_G$$

then for every $\Sigma' \supset \Sigma$, *every state* $\sigma'_G : V' \rightarrow TERM(\Sigma')$ *and* M_Σ $(M_{\Sigma'})$ *is a modelframe w.r.t.* Σ (Σ')

$$M_{\Sigma'} \models \sigma'_G(A)$$

Definition 5 (Closed World Assumption, CWA_{ILP}**)** *Let* Σ *be a minimal generating signature according to a set of formulas* X, A *is a ground atom.* M_{min} *is the least model of* X. $\Vdash$ *is the semantical entailment relation. Then define*

$$\mathrm{CWA(X)} := \{A : M_{min} \Vdash A \textit{ and } A \in FORM(\Sigma)\}$$

$$\cup \{\neg A : M_{min} \nVdash A \textit{ and } A \in FORM(\Sigma)\}$$

In logic programming OWA and CWA are used mutually exclusive, [Kowalski, 1989]. Kowalski related the CWA to a static representation and the OWA to a dynamic representation. Thus, in ILP are CWA_{ILP} and OWA mutually exclusive used. But to express the OWA we need more formal definitions and a closer look to three–valued semantics.

3.2 A partial Herbrand interpretation

Here, we follow the framework of [Wagner, 1991]. For the signature Σ, the Herbrand Universe U_Σ consists of all ground terms. We regard non–ground clauses as a dynamic representation of the corresponding set of ground clauses formed by means of the current domain of individuals. Therefore, a state or variable assignment σ is simply a function which assigns each variable a ground term. The Herbrand base, B_Σ, is the set of all ground atoms. We use the same symbol for syntactical and semantical objects.

$M = \langle M^+, M^- \rangle$ should characterize a partial Herbrand interpretation in which M^+ and M^- are disjoint subsets of B_Σ. M^+ contains the objects which are true and M^- contains those which are false. Partial Herbrand interpretations give rise to a model as to a countermodel relation, $=\!\mid$.

Definition 6 (Satisfaction) *Let σ be a state, F a formula. We write $M \models \sigma(F)$ to say M satisfies F in a state σ.*

$M \models F$	*iff*	$M \models \sigma(F)$ *for all states* σ
$M \models A$	*iff*	$A \in M^+$
$M \models F_1 \vee \ldots \vee F_n$	*iff*	$M \models F_i$ *for some* $1 \leq i$ *leqn*
$M \models F_1 \wedge \ldots \wedge F_n$	*iff*	$M \models F_i$ *for all* $i = 1, \ldots, n$
$M \models A \leftarrow F_1 \wedge \ldots \wedge F_n$	*iff*	$A \in M^+$ *or* $F_i \in M^-$ *for all* $i = 1, \ldots, n$
$M \models \neg F$	*iff*	$M =\!\mid F$
$M =\!\mid A$	*iff*	$A \in M^-$
$M =\!\mid \neg F$	*iff*	$M \models F$
$M =\!\mid F_1 \wedge \ldots \wedge F_n$	*iff*	$M =\!\mid F_i$ *for all* $i = 1, \ldots, n$
$M =\!\mid F_1 \vee \ldots \vee F_n$	*iff*	$M =\!\mid F_i$ *for some* $1 \leq i$ *leqn*

We write $M \models X$, where X is a set of formulas, if $M \models A$, for each $A \in X$. The symbol $\Vdash$ denotes the semantical entailment relation. If, for example, X is a set of formulas and A is a formula, then we write $X \Vdash A$, iff every model of X is model of A, or if $M \models X$ then $M \models A$.

Definition 7 (Least Model) *Let $M = \langle M^+, M^- \rangle, M' = \langle M'^+, M'^- \rangle$ be two models for a set X of formulas. We say that M is a* submodel *of M' (denoted by $M \subseteq M'$) iff $M^+ \subseteq M'^+$. We say that M is a least model for X, if for every $M' \models X$, $M \subset M'$ holds.*

Note that due to the fact that nothing is said about M^- and M'^-, there can be incomparable models with respect to $\subseteq$. However we have now an ordering on models and interpretations, respectively.

Let us briefly recall some results of logic programming.

Observation 1 *Let M, M' be two interpretations such that $M \subseteq M'$, X a set of formulas and P a logical program. Then the following holds:*

1. *if* $M \models X$ *then* $M' \models X$
2. *P has a least model*

A proper partial Herbrand interpretation determines a three–valued assignment v_m on ground atoms. We use v_m to describe OWA and give an equivalent definition of CWA_{ILP} regarding the truth assignment v_m.

Definition 8 (Truth Assignment with OWA) *Let $M = \langle M^+, M^- \rangle$ be the partial Herbrand interpretation and A a ground atom. The OWA is used iff the following three–valued truth assignment, v_m, yields:*

$$v_m(A) = \begin{cases} true & if \quad M \models A \\ false & if \quad M \dashv A \\ unknown & if \quad otherwise \end{cases}$$

Definition 9 (Truth Assignment with CWA_{ILP}) *Let $M = \langle M^+, M^- \rangle$ be the partial Herbrand interpretation and A a ground atom. Σ be a minimal generating signature according to a set of formulas X. The CWA_{ILP} is used iff the three–valued truth assignment, v_m, yields:*

$$v_m(A) = \begin{cases} true & if \quad M \models A \ and \ A \in FORM(\Sigma) \\ false & if \quad not M \models A \ and \ A \in FORM(\Sigma) \\ unknown & if \quad otherwise \end{cases}$$

3.3 A semantic of a program P

We now give two interpretations: I_{CWA} and I_{Neg}. The first interpretation behaves exactly like the CWA_{ILP} combined with the ODA, in that those formulas which are not explicitly interpreted as *true* are (implicitly) false. The second interpretation, I_{Neg} which correspond to the OWA, interprets formulas as false only if they are stated *explicitly* as false. Hence, the interpretations are able to handle formulas which are *unknown*, namely those which are neither *true* nor *false*.

Let $B_{\Sigma'}$ be the Herbrand base of a logical program $P \subseteq \mathrm{FORM}(\Sigma')$ and B_Σ the Herbrand base of a minimal generating signature $\Sigma \subseteq \Sigma'$ of P. To determine the least model, we introduce according to [Apt, 1990] an immediate consequence operator $T_P(I)$. Then, we iteratively apply $T_P(I)$ to construct M^+. The result is a least fixpoint of M^+ with respect to a minimal generating signature Σ of P. The set M^- is determined by the Herbrand base of the minimal signature without M^+.

Definition 10 (T operator) *The $T_P(I)$ operator maps one Herbrand interpretation to another Herbrand interpretation.*

$$\begin{array}{ll} A \in T_P(I) \text{ iff} & \text{for some substitution } \sigma \\ & \text{and a clause } B \leftarrow B_1, \ldots, B_n \text{ of } P \\ & \text{we have } A \equiv B\sigma \text{ and } I \models (B_1, \ldots, B_n)\sigma \end{array}$$

If $T_P(I_f) = I_f$ holds, I_f is called a pre-fixpoint of T. We know from the results of logic programming that $T_P(I_f)$ is a model of P if I_f is a pre–fixpoint of T_P. Another result is that each monotonic operator T has a least fixpoint which is also its least pre-fixpoint. If Σ is a minimal generating signature for P, we can construct M^- by $B_{\mathrm{MS}} without M^+$.

P:	deputy(tom),deputy(x) $\rightarrow$ corrupt(x),rich(tom),rich(bill)
B_{MS}	deputy(tom),corrupt(tom),rich(tom),rich(bill) deputy(bill) corrupt(bill)
M^+:	deputy(tom),corrupt(tom),rich(tom),rich(bill)
M^-:	deputy(bill) corrupt(bill)

Table 5: Example of I_{CWA}

Definition 11 (Interpretation I_{CWA}) *Let P be a program generated by a minimal signature Σ. The interpretation I_{CWA} is characterized by $\langle M^+, M^- \rangle$, whereas M^+ is a pre-fixpoint of $T_P(I)$. M^- is determined by B_Σ or formally:*

$$M^- = \{a : \ a \in B_\Sigma \ and \ a \notin M^+\}$$

Table 5 shows a set of formulas P and the corresponding Herbrand base B_{MS}. M^+ is constructed by the least model of P and M^- by the difference of B_{MS} and M^+.

Observation 2 *Let $P \subseteq FORM(\Sigma')$ be a program and $\Sigma \subseteq \Sigma'$ the minimal generating signature of P. If $FORM(\Sigma) = FORM(\Sigma')$, then M^+ corresponds exactly to the least model of P.*

It is easy to see that the interpretation I_{CWA} corresponds to the CWA_{ILP} and the ODA. Now, let us formulate an interpretation which handles explicit negation regarding the OWA. The interpretation I_{Neg} of a set of formulas with negative examples is constructed from the partial Herbrand models. M^+ was constructed from the least fixpoint of $T_P(I)$. The set M^- is a least model of the negative examples.

Definition 12 (Interpretation I_{Neg}) *The interpretation I_{Neg} is characterized by $\langle M^+, M^- \rangle$, whereas M^+ is a pre-fixpoint of $T_P(I)$. M^- is a Herbrand Interpretation of the negative examples, or formally:*

$$M^- = \{a : \ a \in E^- \ and \ a \ is \ ground\}$$

Clearly, formulas which are neither in M^+ nor in M^- are unknown. We do not have a normative truth-value assignment like the CWA (those things which are unknown are false).

Table 6 shows an example of I_{Neg}, with a set of clauses, P, and the corresponding Herbrand base B_{MS}. M^+ is constructed from the least model of P and M^- from the given negative examples.

3.4 Valid hypotheses

We define a set of hypotheses H which characterizes the intended interpretation of a given set of formulas in ILP. We add the set H to the given formulas and determine again with

P:	deputy(tom),deputy(x) $\rightarrow$ corrupt(x),rich(tom),rich(bill) $\neg$ deputy(bill)
B_{MS}	deputy(tom),corrupt(tom),rich(tom),rich(bill) deputy(bill) corrupt(bill)
M^+:	deputy(tom),corrupt(tom),rich(tom),rich(bill)
M^-:	deputy(bill)

Table 6: Example of I_{Neg}

the T_P operator the truth value *true* or *false* of the up to now formulas with the truth value *unknown*. Formally, the T_P operator map the truth value *unknown* of any formulas to *true* or *false*. But first we define the set H and in favor of its size two more restricted sets. The interpretation $M = \langle M^+, M^- \rangle$ is related to I_{CWA} or I_{Neg}.

Definition 13 (Set of Hypotheses H) *Let $Pre \in FORM(\Sigma)$, where $FORM(\Sigma)$ is the minimal generating signature of the examples, $Q \in FORM(\Sigma)$. Q may be a conjunction of atoms.*

$$H = \{Q \leftarrow Pre: \ \textit{for all } \sigma \ (\sigma(Q) \notin M^- \ \ \textit{or } \sigma(Pre) \notin M^+)\}$$

Helft's idea, [Helft, 1989], is to restrict this set H by there is a σ, such that $\sigma(Pre) \in M^+$. This ensures that of each hypothesis both, the premise and conclusion, have to be once satisfied. This gives a better confirmation, because for example, if we add the negative example *woman(tom)*, we don't have the hypothesis *deputy(tom) $\leftarrow$ woman(tom)* in contradiction to H.

Definition 14 (Helft's Set of Hypotheses, H_{Helft})

$$H_{Helft} = \{Q \leftarrow Pre: \ \forall\sigma \ (\sigma(Q) \notin M^- \ \ or \ \sigma(Pre) \notin M^+) and \exists\sigma \ \sigma(Pre) \in M^+\}$$

A further restriction is to regard only hypotheses with the same conclusion. This is called single predicate learning in contrast to multiple predicate learning of the set H, [Muggleton, 1993].

Definition 15 (Single Predicate Learning, H_{SPL})

$$H_{SPL} = \{Q \leftarrow Pre: \ \forall\sigma \ (\sigma(Q) \notin M^- \ \ or \ \sigma(Pre) \notin M^+)$$

$$and \ \exists\sigma \ \sigma(Pre) \in M^+ \ and \ \forall i,j \ Q_i = Q_j\}$$

Table 7 shows a procedural view of our framework. We determine from a given set of formulas the partial Herbrand interpretation. We construct then a set H and try to map the truth value *unknown* to *true* or *false* with the T_P operator. Hence, we have determined the truth value of up to now unknown formulas which is the prediction.

Given set of WFFs		Partial Herbrand Interpretation		Set H		Prediction

Given set of WFFs $\Rightarrow$ Partial Herbrand Interpretation $\Rightarrow$ Set H $\Rightarrow$ Prediction

Table 7: A procedural view of our framework

Logic Programming	Inductive Logic Programming	
	I_{CWA}	I_{neg}
CWA + DCA	CWA_{ILP} + ODA	OWA

Table 8: Summary of the assumptions

3.5 Summary

We have related the OWA to the interpretation I_{neg} and the combination of CWA_{ILP} and ODA to the interpretation I_{CWA}, table 8. The set of hypotheses is determined regarding the interpretation and is a set of all possible hypotheses. Several learning algorithm restricts this set, for example FOIL with *single predicate learning* uses only a subset with always the same conclusion in contrast to multiple predicate learning. However, in the next section we point out several relations which can be discarded using this framework.

4 Relations

4.1 Helft: induction as nonmontonic inference

In the following we restrict the framework of [Helft, 1989] to horn clause logic. It is easy to see that the minimal models coincide with the least fixpoint under a Herbrand–Interpretation.

Definition 16 (Helft's language) *Helft uses a language which is restricted by the following items:*

1. **Groundable**: *Function symbols are not used and every variable in the head of the clause has to appear in the body. This ensures that a finite least model always exists.*
2. **Injective:** *There is a substitution or state τ, mapping the literals of P onto elements of A, such that for every pair of variables x, y of P, $\tau(x) \neq \tau(y)$. This ensures that the set of generalizations is finite and no unnecessary variables are introduced.*

Helft constructs a minimal model with respect to a set of formulas. In our terminology, M^+ corresponds to the least model and M^- is the Herbrand base without the least model with respect to the minimal signature Σ. We use only one set of generalizations because

P:	deputy(tom),deputy(x) $\rightarrow$ corrupt(x),rich(tom),rich(bill)
ϕ_1:	rich(x) $\leftarrow$ deputy(x)
ϕ_2:	rich(x) $\leftarrow$ corrupt(x)
ϕ_3:	corrupt(x) $\leftarrow$ rich(x)
B_{MS}	deputy(tom),corrupt(tom),rich(tom),rich(bill) deputy(bill) corrupt(bill)
M^+:	deputy(tom),corrupt(tom),rich(tom),rich(bill)
M^-:	deputy(bill),corrupt(bill)
	$I_{CWA} \models \phi_1, \phi_2$

Table 9: Example of Helft

we always have an unique minimal model. Helft has to distinguish between strong and weak generalizations.

Definition 17 (Γ) *Let M be an interpretation and $\phi = P \leftarrow Q$ a clause. Δ is a finite set of groundable clauses and Γ is the set of generalizations which are defined as:*

- *$M \models \phi$ and $M \models_\tau Q$, whereas $\tau_\Sigma(Q)$ is one ground instance and Q is injective over M.*
- $\Delta \not\Vdash \phi$
- *For every clause ϕ' of $\Gamma(\Delta)$, if $\phi' \Vdash \phi$ then $\phi \Vdash \phi'$*

Theorem 1 $I_{CWA}(\Delta) \models \Gamma$

Proof
Assume that $I_{CWA}(\Delta) \not\models P \leftarrow Q$ and $P \leftarrow Q \in \Gamma$. Without loss of generality assume that, the rule is ground. Note that Q can be a conjunction of atoms.
We know from the definition of satisfiability that $P \notin M^+$ and that $Q \notin M^-$. We also know, that P and Q are in the minimal signature Σ. Therefore $P \in M^-$ and $Q \in M^+$ because all members of B_P are by definition in M^+ or in M^-.
In words of [Helft, 1989], $M \not\models P \leftarrow Q$. This contradicts the first condition of Helft's definition Therefore $P \leftarrow Q$ is not in Γ and this concludes our proof. $\Box$

Table 9 shows an example of Helft, in which each proposed generalization is satisfied by $I_{CWA}(\Delta)$.

4.2 FOIL in a logical framework

We show, in this section, that FOIL[3] is an example for a learning system based on Plotkin's conditions. In another view, FOIL is an example of Helft's framework. Both fit into our

[3] To confirm our results we have used in fact FOIL2.

Input of FOIL	p: bill,tom,hank,joe,pete,carl,go.	
	rich(p)	bill,+ tom,+ .
	*corrupt(p)	tom,+.
	*deputy(p)	tom,+ ; bill,- .
Output of FOIL:	*** Warning: the following definition *** does not cover 1 tuple in the relation rich(A) :- *corrupt(A)	
M^+:	deputy(tom),corrupt(tom),rich(tom),rich(bill)	
M^-:	deputy(bill)	

Table 10: Example of FOIL

logical framework. We do not want to investigate the information–based selection of a certain generalization out of the hypothesis set. We do not regard anything like predicate invention in this framework. We also do not allow rules with negative literals in the body.

FOIL is a learning system which infers from a trainingset of examples a formula which describes the selected concept. The examples are given by ground atoms and are divided into positive examples and background knowledge. FOIL selects the best hypothesis with an information based measure that covers as much as possible of the positive examples and no negative example. FOIL works in two modes. In one mode, only positive examples are given, and the CWA is used to construct negative examples. This corresponds to our interpretation I_{CWA}. The other mode in which positive and negative examples are given corresponds to our interpretation I_{Neg}. Our observation that each hypothesis is a member of our hypotheses set H is independent of the chosen model.

Table 10 shows FOIL applied to the example of Helft. The rule is valid because the conditions of Plotkin are fulfilled. The only exception is that $rich(bill)$ cannot be deduced from the hypothesis and the background knowledge.

Theorem 2 *Each FOIL learning result is a member of H_{SPL}.*

Proof:
This is an easy proof because H_{SPL} is exactly constructed to fulfill the conditions of Plotkin and Helft. First, we show this for the case where we give explicitly positive and negative examples, and second, where FOIL uses the CWA. $P \leftarrow Q$ is a FOIL learning result.

1. Here the conditions of Plotkin are fulfilled. By definition of the set H_{SPL}, we know that $P \notin M^-$ or $Q \notin M^+$. Then we see that P also has to be in M^+, because $Q \in M^+$, and we want to deduce P from the hypothesis. The consistency is guaranteed by the disjointness of M^+ and M^-.
2. In this case, we know that all positive examples are in M^+ and if we want to deduce the positive examples, P also has to be in M^+. To deduce the positive examples, we also need to satisfy the background knowledge. This means that for instance $Q \in M^+$. This fulfills the conditions of H. □

4.3 Plotkin's inductive task

Following the framework of [Plotkin, 1971], the inductive learning task can be described using a background theory **B**, a set of positive examples E^+, a set of negative examples E^-, a hypothesis H, and a partial ordering relation $\leq$. The conditions are:

1. The background theory should not entail the positive examples, $B \not\Vdash E^+$ (Prior necessity),
2. and it should be consistent with the examples, $B, E^-, E^+ \not\Vdash \perp$ (Prior satisfiability).
3. The background theory and the hypothesis have to entail the positive examples, $B, H \Vdash E^+$ (Posterior sufficiency).
4. The background theory and the hypothesis should not entail the negative examples, $B, H, E^- \not\Vdash \perp$ (Posterior Satisfiability).
5. We should use the most specific hypothesis that fulfills the above requirements. Therefore we have an ordering $\leq$ on the hypotheses, whereas h1 $\leq$ h2 means h1 is more general than h2

Plotkin has proven in general, there is no hypothesis which covers exactly the positive examples and nothing else. In general, we can deduce from a hypothesis facts which are not known. We call this set of facts *prediction*. To see that the set H fulfills the conditions of Plotkin, we test each condition:

1. The background theory should not entail the positive examples. This can be guaranteed by the restriction that background theory and the examples are disjointed.
2. The backgroundknowledge and the examples should be consistent is satisfied by the restriction that M^+ and M^- are disjointed.
3. The condition $B, H \models E^+$ guarantees that we can conclude the positive examples from the hypothesis and the background knowledge. This means, if the body of each hypothesis is in M^+, we can deduce the head. Therefore $Q \in M^+$.
4. The consistency is satisfied by the restriction that M^+ and M^- are disjoint.

4.4 Comparison with other three–valued logics

Using our framework the underlying logic of ILP can be compared with other three–valued logics fairly easy. We will compare our three–valued logic with the logics of Kleene and Lukasiewicz, [Turner, 1984], whereas we only regard the implication.

First let us look on Kleene's strong three–valued implication. Kleene's concern was to model mathematical statements which that means his undecidable third truth value represents neither true nor false, but rather a state of partial ignorance [Turner, 1984]. We guess, that this is common to our approach, where the third truth value also indicates that we do not know anything. Nevertheless, we have a very *optimistic* view in learning, because there is always the danger of assuming that **all** our hypotheses are true. Kleene

Logic of ILP:

Pre → Q	t	f	u
t	t	f	t
f	t	t	t
u	t	t	t

Kleene's logic:

Pre → Q	t	f	u
t	t	f	u
f	t	t	t
u	t	u	u

Łukasiewicz's logic:

Pre → Q	t	f	u
t	t	f	i
f	t	t	t
i	t	i	t

Table 11: Comparison of logics

did not want to conclude from an unknown sentence something known. Specially in the case where $u \rightarrow u$ is undecided.

For a second comparison let us look of Łukasiewicz three–valued logic. In this logic the case $u \rightarrow u$ is true. Turner points out that the difference origin from the different interpretation of th third truth value. Whereas in context of Kleene this third truth value is a truth gap, Łukasiewicz assigned the third truth value an statement, if no assignment of true or false is possible. However he deals with statements about the future.

Again have we the most optimistic view of the implication. This means if it is possible that a fact implies another fact, this statement is true. The other semantics are more cautious, because obviously this can be false.

5 Conclusions and further works

In this work, we have compared the basic assumptions of logic programming and ILP. Then, we have given a three–valued model–based semantic of ILP and shown that FOIL fits in this framework. The semantic reflects the two main assumptions which we need in machine learning. Finally we have related our framework with other works of ILP and three–valued logics.

Currently we are implementing our framework with ISABELLE, [Paulson, 1989], and doing experiments. We are also evaluating more restrictions on the set H.

In further works we want to show the need to handle more than three truth–values in machine learning. For example, we will show that it is useful to handle inconsistencies in an incremental learning system. We often come up with inconsistencies, and we want to wait for a less busy time to be able to make revisions because making revisions is an expensive task.

Acknowledgments

We would like to thank Katharina Morik and the members of Computer Science VIII of the University Dortmund. This work was partly supported by the European Community (ESPRIT Basic Research Action 6020, project Inductive Logic Programming).

References

[Apt, 1990] Apt, K. (1990). Logic programming. In van Leeuwen, J., editor, *Handbook of theoretical Computer Science, Volume B*. Elsevier.

[Apt and van Emden, 1982] Apt, K. and van Emden, M. (1982). Contributions to the theory of logic programming. *ACM*, 29(3).

[Clark, 1978] Clark, K. (1978). Negation as failure. In Gallaire, H. Minker, J., editor, *Logic and Data Bases*, pages 293 – 322. Plenum Press, New York.

[DeRaedt and Bruynooghe, 1989] DeRaedt, L. and Bruynooghe, M. (1989). Towards friendly concept-learners. In *Proc. of the 11th Int. Joint Conf. on Artif. Intelligence*, pages 849 – 854, Los Altos, CA. Morgan Kaufman.

[Fitting, 1985] Fitting, M. (1985). A Kripke-kleene semantics for general logic programs. *Logic Programming*, 2:295–312.

[Helft, 1989] Helft, N. (1989). Induction as nonmonotonic inference. In *Proceedings of the 1st International Conference on Knowledge Representation and Reasoning.*

[Kietz and Wrobel, 1991] Kietz, J.-U. and Wrobel, S. (1991). Controlling the complexity of learning in logic through syntactic and task-oriented models. In Muggleton, S., editor, *Inductive Logic Programming*, chapter 16, pages 335 – 360. Academic Press, London. Also available as Arbeitspapiere der GMD No. 503, 1991.

[Kowalski, 1979] Kowalski, R. (1979). *Logic for Problem Solving.* Elsevier Science Publishing Co. Inc.

[Kowalski, 1989] Kowalski, R. (1989). Logic for data description. In Mylopoulos, J. and Brodie, M., editors, *Readings in Artificial Intelligence and Databases.* Morgan Kaufmann.

[Lloyd, 1987] Lloyd, J. (1987). *Foundations of Logic Programming.* Springer Verlag, Berlin, New York, 2nd edition.

[Muggleton, 1990] Muggleton, S. (1990). Inductive logic programming. In *Proceedings of the 1th conference on Algorithmic Learning Theory.*

[Muggleton, 1993] Muggleton, S. (1993). Inductive logic programming: Derivations, successes and shortcommings. In Brazdil, P., editor, *ECML-93, European Conference on Machine Learning.*

[Muggleton and Feng, 1992] Muggleton, S. and Feng, C. (1992). Efficient induction of logic programs. In Muggleton, S., editor, *Inductive Logic Programming*, chapter 13, pages 281–298. Academic Press, London.

[Paulson, 1989] Paulson, J. (1989). The foundation of a generic theorem prover. *Automated Reasoning*, 5.

[Plotkin, 1971] Plotkin, G. D. (1971). A further note on inductive generalization. In Meltzer, B. and Michie, D., editors, *Machine Intelligence*, chapter 8, pages 101–124. American Elsevier.

[Quinlan, 1990] Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, 5(3):239 – 266.

[Reiter, 1978] Reiter, R. (1978). On closed world data bases. In Gallaire, H. Minker, J., editor, *Logic and Data Bases.* Plenum Press, New York.

[Shepherdson, 1987] Shepherdson, J. (1987). Negation in logic programming. In Minker, J., editor, *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufmann Publisher Inc.

[Turner, 1984] Turner, R. (1984). *Logics for Artificial Intelligence*. Ellis Horwood limited.

[Wagner, 1991] Wagner, G. (1991). Logic programming with strong negation and inexact predicates. *Logic and Computation*, 1(6):835–859.