

Representation of Incomplete Knowledge by Induction of Default Theories

Pascal Nicolas and Béatrice Duval

LERIA - University of Angers - France
2, boulevard Lavoisier, F-49045 Angers Cedex 01
{pascal.nicolas|beatrice.duval}@univ-angers.fr

Abstract. We present a method to learn simultaneously definitions for a concept and its negation. This problem is relevant when we have to deal with a complex domain where it is difficult to acquire a complete theory and where we have to reason from incomplete knowledge. We use default logic to represent such incomplete theories. This paper specifies the problem of learning a default theory from a set of examples and a background knowledge. We propose an operational method to inductively construct such a theory. Our learning process relies on a generalization mechanism defined in the field of Inductive Logic Programming. We first consider the case where the initial knowledge is sure because it contains only ground facts. Then, we extend the framework to the case where the initial knowledge is a default theory.

1 Introduction

We present here a method that enables to construct a default theory from a set of positive and negative examples and an initial background knowledge. The learning process that we propose is strongly related to research realized in the field of Inductive Logic Programming (ILP). ILP investigates theory and methods to induce first-order clausal theories from examples and background knowledge [18]. More precisely, in the normal framework of ILP, if B is the background knowledge and E^+ and E^- are the sets of positive and negative examples respectively, the aim is to induce hypotheses H such that $B \wedge H \models E^+$ and $B \wedge H \wedge E^- \not\models \perp$. In most cases, E^+ and E^- are examples of a single target predicate and B and H are definite Horn clauses (but some systems [6] induce full first-order theories). The ILP community has also considered the problem of using more expressive formalisms, specially in systems that construct clauses containing the negation as failure operator [1, 2, 15].

The problem that we consider in this paper extends [9] and concerns the simultaneous learning of definitions for a predicate p and its negation $\neg p$. So in our framework, negative examples for a predicate p will play the full role of leading to explicit definitions of $\neg p$. The relevance of this approach has been first pointed out by De Raedt [5] who argued that the closed world assumption is not suited to the learning paradigm because we cannot assume that everything is known. Our proposition follows the same idea and is concerned with the construction

of theories where it seems difficult to apply the closed world assumption. Let us imagine a secretary-agent that must learn from observations when it must pass on a phone call to the manager. This concept seems difficult to define completely. So it is a good representation for the agent to define explicitly situations where the call can be passed on, and situations where the manager must not be disturbed. This formalism enables the agent to recognize cases where the concept remains undefined according to the current learned theory. A situation may also be undetermined because it satisfies at the same time a positive and a negative definition.

In order to give explicitly definitions of p and $\neg p$ and to deal with possible inconsistencies between them, we propose to represent the learned knowledge by a default theory. *Default logic* [22] is a powerful language to represent incomplete knowledge, which enables our method to obtain compact theories where the relationships between the definitions for p and $\neg p$ appear clearly. In default logic, knowledge is represented by a default theory (W, D) , where W is a set of classical formulas (the sure knowledge) and D is a set of *default rules* (or *defaults*) that represent non completely specified inference rules, often considered as *rules with exceptions*. Formally, a default $\frac{\alpha:\beta}{\gamma}$ has a *consequent* γ and two types of antecedents: a *prerequisite* α and a *justification* β ¹. Then, the intuitive meaning of a default rule is : “if α is proved, and if $\neg\beta$ is not deducible (in other words if β is coherent) then conclude γ ”. In whole generality, α , β and γ can be any first order logic formula. But in our work, they will be formulas with free variables, like $p(X, Y)$, so our defaults are said to be *open*. As usual in default logic, each formula $p(X, Y)$ represents the set of all ground formulas $p(a, b)$ that can be obtained by instantiation with the constants of the domain. In this work, we only consider finite domains (without symbol function) and then our set of open defaults is in fact a compact representation of a finite set of *closed defaults* (without free variables) obtained by instantiation over the constant set.

We recall below the definition of an *extension* that is a set of plausible conclusions inferred from a given *closed default theory* (see [22] for more details on default logic). A default theory is said to be closed if all its defaults are closed.

Definition 1. [22] *Let (W, D) be a closed default theory. For any set of closed formulas S , let $\Gamma(S)$ be the smallest set satisfying :*

- $W \subseteq \Gamma(S)$
- $Th(\Gamma(S)) = \Gamma(S)$
- For any $\frac{\alpha:\beta}{\gamma} \in D$, if $\alpha \in \Gamma(S)$ and $\neg\beta \notin S$, then $\gamma \in \Gamma(S)$.

A set of closed formulas E is an extension of (W, D) iff $E = \Gamma(E)$.

A fundamental feature of default logic is its ability to represent incomplete knowledge, so it is not surprising that a default theory may have multiple extensions : one for each point of view that we can adopt in front of the missing information. For instance $(W, D) = (\{a\}, \{\frac{a:b}{c}, \frac{a:\neg c}{\neg b}\})$ has two extensions

¹ If δ is a default rule, $pre(\delta)$, $jus(\delta)$ and $cons(\delta)$ respectively denote the prerequisite, the justification and the consequent of δ .

$E_1 = Th(W \cup \{c\})$ and $E_2 = Th(W \cup \{\neg b\})$. That is why it is necessary to distinguish between *skeptical* (or *cautious*) *theorems* and *credulous theorems*. The former are formulas that occur in every extension ($c \vee \neg b$ in our previous example) and can be considered as sure deductions. The later are formulas that occur in at least one extension (c in our previous example) and are only hypothetical conclusions. As it will be described later, this distinction is central in the paradigm that we present in our work.

The rest of the paper is organised as follows : section 2 considers default learning in the case where the initial theory does not contain defaults. Our methodology is illustrated on examples in section 3. In section 4, we develop the more general framework of learning with an initial theory that contains defaults. Then, we compare our work with other approaches in section 5.

2 Learning Default Theories

2.1 Definition and algorithm

The following definition formally precises the framework of learning a default theory; it is inspired by a well known semantic specification of ILP. In this section, we consider the special case where the initial background knowledge is expressed by ground facts.

Definition 2. Let $E^+ = \{p(a_1), \dots, p(a_n)\}$ be a set of positive examples and $E^- = \{\neg p(a'_1), \dots, \neg p(a'_m)\}$ a set of negative examples of a target predicate p . Let W be an initial set of ground facts containing no occurrence of p or $\neg p$,

Learning a default theory for the concept described by p and $\neg p$ consists in finding a default theory (W', D') such that:

- D' is a set of defaults, the consequents of which are p or $\neg p$
- $W' = W \cup E_p$, where E_p is a set of examples that cannot be generalized
- $(\bigwedge_{e \in E^+} e) \wedge (\bigwedge_{e \in E^-} \neg e)$ is a skeptical theorem of (W', D') .

Definition 3. An example e ($p(a)$ or $\neg p(a)$) is covered by a default theory (W, D) if e is a credulous theorem of (W, D) .

An example e ($p(a)$ or $\neg p(a)$) is an **exception** to a default theory (W, D) if $\neg e$ is a credulous theorem of (W, D) .

Our approach considers that the training examples constitute a sure knowledge from which we induce default rules. As a default theory may have multiple extensions mutually inconsistent, our definition requires that the training examples become skeptical theorems of the induced default theory. For instance, let $E^+ = \{flies(1)\}$, $E^- = \{\neg flies(2)\}$ and $W = \{bird(1), bird(2), penguin(2)\}$, and let D' be the default set $D' = \left\{ \frac{bird(X): flies(X)}{flies(X)}, \frac{penguin(X): \neg flies(X)}{\neg flies(X)} \right\}$. The theory (W, D') does not satisfy definition 2. In fact, (W, D') has two extensions $E_1 = Th(W \cup \{flies(1), flies(2)\})$ and $E_2 = Th(W \cup \{flies(1), \neg flies(2)\})$ and consequently, $\neg flies(2)$ is not a skeptical theorem. The reader can easily check

that if we take $D' = \left\{ \frac{bird(X) : flies(X) \wedge \neg penguin(X)}{flies(X)}, \frac{penguin(X) : \neg flies(X)}{\neg flies(X)} \right\}$, then (W, D') is a solution to this simple learning problem.

The main idea is to give symmetric roles to positive and negative examples; the positive examples are used to build defaults defining p and the negative examples are used to build defaults defining $\neg p$. Generalization of positive examples leads to a general rule defining p but this definition may admit exceptions, that are found by examining the negative examples. Generalization from such a set of exceptions enables to specialize the rule defining p and moreover gives a general definition of $\neg p$. A symmetric treatment is also applied to generalization of negative examples. To resume, our method to construct a set of defaults alternates generalization and specialization steps.

The generalization process is based on a generic ILP algorithm named here **Gen** $(q, \mathcal{E}^+, T, \varphi)$ that, from a set of positive examples \mathcal{E}^+ of the predicate q and a background theory T , induces one definition $\varphi(X)$ that characterizes a part of \mathcal{E}^+ . More precisely, that means that the theory T and the clause $(q(X) : \neg\varphi(X))$ enables to prove all the examples $q(a)$ generalized by φ (see section 3 for details).

Formally, the algorithms that we propose are the followings.

Algorithm DefaultLearning

In : $p(X), W, E^+, E^-$; Out : W', D'

Begin

$W' \leftarrow W \quad D' \leftarrow \emptyset$
 $\overline{E^+} \leftarrow E^+$
While $\overline{E^+} \neq \emptyset$

Gen $(p, \overline{E^+}, W, \varphi)$ searches φ that generalizes a part of $\overline{E^+}$
If a formula φ is found, then

Add to D' the default $\delta = \frac{\varphi(X) : p(X)}{p(X)}$

Remove from $\overline{E^+}$ the examples generalized by φ
 $Exc \leftarrow \{e \in E^- \mid e \text{ is an exception to } (W', D')\}$
If $Exc \neq \emptyset$ then **Specialise** $(Exc, W, \delta, W', D', \{pre(\delta)\})$

else

$W' \leftarrow W' \cup \overline{E^+} \quad \overline{E^+} \leftarrow \emptyset$

Endwhile

$JUS(X) \leftarrow \bigwedge \neg pre(\delta)$, for all $\delta \in D'$ s. t. $cons(\delta) = p(X)$
 $\overline{E^-} \leftarrow \{e \in E^- \mid e \text{ is not covered by } (W', D')\}$
While $\overline{E^-} \neq \emptyset$

Gen $(\neg p, \overline{E^-}, W, \varphi)$ searches φ that generalizes a part of $\overline{E^-}$
If a formula φ is found, then

Add to D' the default $\frac{\varphi(X) : \neg p(X) \wedge JUS(X)}{\neg p(X)}$

Simplify $JUS(X) = \bigwedge_{i=1}^k \neg J_i(X)$ by removing each $J_i(X)$ s.t. there is no constant tuple \overline{X} satisfying $p(\overline{X}) \in E^+$ and $W \vdash \varphi(\overline{X}) \wedge J_i(\overline{X})$

Remove from $\overline{E^-}$ the examples generalized by φ

else

$W' \leftarrow W' \cup \overline{E^-} \quad \overline{E^-} \leftarrow \emptyset$

Endwhile

End

Algorithm Specialise

In : Exc, W ; InOut : δ, W', D' ; In: *ForbForm*

Begin

$\overline{Exc} \leftarrow Exc$
While $\overline{Exc} \neq \emptyset$

Gen $(\neg cons(\delta), \overline{Exc}, W, \psi_{Exc})$ searches ψ_{Exc} that generalizes a part of \overline{Exc}
If ψ_{Exc} is found and $\psi_{Exc} \notin ForbForm$, then

$jus(\delta) \leftarrow jus(\delta) \wedge \neg \psi_{Exc}$
Add to D' the default $\delta_{Exc} = \frac{\psi_{Exc}(X) : \neg cons(\delta)}{\neg cons(\delta)}$

Remove from \overline{Exc} the examples generalized by ψ_{Exc}

Endwhile

```

Remove from  $\overline{Exc}$  the examples generalized by  $\psi_{Exc}$ 
/*  $E$  stands for  $E^+$  (resp.  $E^-$ ) if  $cons(\delta) = p(X)$  (resp.  $cons(\delta) = \neg p(X)$ )*
 $Exc_{Exc} \leftarrow \{e \in E \mid e \text{ is an exception to } (W', D')\}$ 
If  $Exc_{Exc} \neq \emptyset$  then Specialise( $Exc_{Exc}, W, \delta_{Exc}, W', D', ForbForm \cup \{pre(\delta_{Exc})\}$ )
else
   $W' \leftarrow W' \cup \overline{Exc}$      $\overline{Exc} \leftarrow \emptyset$ 
Endwhile
End

```

In our main algorithm **DefaultLearning**, p stands for the predicate to learn, E^+ and E^- are the set of positive and negative examples of the concept; W is the initial theory; W' is the theory W which may be augmented by some examples that cannot be generalized; D' is a set of defaults the consequents of which are p and $\neg p$. For clarity, the algorithm is written by assuming that we begin by learning p . But as our method deals with positive and negative examples in a symmetric manner, it could as well begin by learning $\neg p$ by exchanging the roles of E^+ and E^- .

The process starts by a generalization step, that means that the learning algorithm **Gen** is applied to E^+ in order to compute one formula φ that represents a subset of E^+ . If it is possible to find such a formula, the default $\delta = \frac{\varphi(X):p(X)}{p(X)}$ is build into D' .

This default δ may admit exceptions (see definition 3). The set of exceptions is obtained by checking for each $\neg p(e)$ in E^- whether $p(e)$ is a theorem of (W', D') . If the set of exceptions is not empty, we must specialize δ . **Gen** is used to induce a formula ψ that generalizes these exceptions and we modify the default δ into $\frac{\varphi(X):p(X) \wedge \neg \psi(X)}{p(X)}$. By this way, this default is no longer applicable to the negative examples that verify $\psi(X)$. At the same time, these negative examples, generalized by $\psi(X)$, lead to a general definition of $\neg p$, represented by the default $\frac{\psi(X):\neg p(X)}{\neg p(X)}$. This default is specialized on its turn if it is necessary. This recursive process always ends because we use the set of forbidden formula, *ForbForm*, that avoids possible loops in the situations where exceptions and examples are generalized by the same formula. For instance, with $E^+ = \{flies(1), flies(2)\}$, $E^- = \{\neg flies(3), \neg flies(4)\}$ and $W = \{bird(1), bird(2), bird(3), bird(4)\}$, we obtain the first default $\delta_1 = \frac{bird(X):flies(X)}{flies(X)}$ which has the exceptions E^- . If we specialize δ_1 without taking into account *ForbForm*, we obtain the new default $\delta_2 = \frac{bird(X):\neg flies(X)}{\neg flies(X)}$ and δ_1 is specialized in $\delta'_1 = \frac{bird(X):flies(X) \wedge \neg bird(X)}{flies(X)}$. This is not acceptable because the positive examples $flies(1), flies(2)$ are no longer covered by this theory, and become exceptions to δ_2 , which leads to a loop in this recursive specialization. The use of *ForbForm* enables to find the final theory $(W \cup \{\neg flies(3), \neg flies(4)\}, \left\{ \frac{bird(X):flies(X)}{flies(X)} \right\})$, because an example e that cannot be generalized is simply added to W' as a ground fact. This ensures that e is a skeptical theorem of (W', D') .

When all the positive examples are generalized (first Endwhile), we check whether there are still any negative examples not covered by the current theory. If it is the case, we begin to complete the definition of $\neg p$ by a similar process. At this time, all the positive examples, that are the potential exceptions for

defaults defining $\neg p$, have already been treated. So the formulas to characterize these possible exceptions have already been computed: they are the prerequisites of some defaults defining p . That is why all the new defaults that are introduced for $\neg p$ are constrained by a justification JUS that is the conjunction of all the prerequisites of all defaults concluding p . By this way, we avoid the computation of exceptions, which is an expensive process. The counterpart of this strategy is that the justifications of these last defaults are certainly too complex and they are simplified by a mechanism that checks for each new default whether these formulas really correspond to some exceptions.

A last point to notice is that in our algorithm the sets of examples that are not yet covered (\overline{E}^+ and \overline{E}^-) decrease each time that **Gen** generalizes a subset of examples: this is the principle of iterative covering common to many learning algorithms. But when we have to determine the exceptions to a current theory, we must take into account the initial sets of examples E^+ and E^- . This is necessary to be sure that we have found all the possible exceptions.

2.2 Correctness

The work presented here extends a previous method [9] that concerned only Lukasiewicz' default theories where the existence of an extension is guaranteed. In Reiter's default logic, this point must be more carefully studied.

Theorem 1. *The algorithm **DefaultLearning** induces a default theory that has always an extension.*

Proof: In [14] it is shown that a Reiter's default theory has at least one extension if its *block-graph* contains only even cycles. For a default theory (W, D) , the block-graph is a pair (\overline{D}, A) . The vertex set \overline{D} contains all closed defaults obtained from D except those that are incompatible with W , ie: defaults δ s.t. $W \vdash \neg jus(\delta)$. In our particular case, the arc set A contains the pair (δ, δ') if δ "blocks" δ' , i.e.: $W \vdash pre(\delta)$ and $W \cup cons(\delta) \vdash \neg jus(\delta')$. By construction, each induced default is normal or semi-normal² and its consequent is $p(\overline{X})$ or $\neg p(\overline{X})$. So, it is obvious that only even cycle may exist and then our learned default theories have always an extension. \square

When it ends, our algorithm guarantees that all examples are covered (each given example e is a credulous theorem) and that there are no remaining exceptions (for each given example e , $\neg e$ is not a credulous theorem). So we have to prove that it is sufficient to make all the examples skeptical theorems, as it is required by definition 2.

Theorem 2. *Let (W, D) be a default theory induced by the algorithm **DefaultLearning** and e a given example.*

If e is a credulous theorem of (W, D) and $\neg e$ is not a credulous theorem of (W, D) , then e is a skeptical theorem of (W, D) .

² A default is semi-normal if it is like $\frac{\alpha : \beta \wedge \gamma}{\gamma}$.

Proof: Without loss of generality we fix that e is a positive example $p(a)$ (the proof for a negative example is similar) such that $p(a)$ is a credulous theorem and $\neg p(a)$ is not a credulous theorem. Since $p(a)$ is a credulous theorem it means that there exists a closed default $\delta = \frac{\alpha(a) : p(a) \wedge \beta(a)}{p(a)}$ (particularly we may have $\beta(a) = \text{true}$) with $W \vdash \alpha(a)$.

Let us suppose that $p(a)$ is not a skeptical theorem. In other words, there exists an extension E not containing $p(a)$, then δ is blocked in E that is $E \vdash \neg p(a) \vee \neg \beta(a)$. Since $\neg p(a)$ is not a credulous theorem, it is never possible to obtain $\neg p(a)$, so the only way to block δ is to derive $\neg \beta(a)$. $\neg \beta(a)$ cannot be obtained by a default δ' because its consequent can only be $p(X)$. So, we must have $\neg \beta(a) \in W$. But in this case, δ is always blocked and $p(a)$ is not a credulous theorem. This contradiction gives our result. \square

The next section gives examples illustrating our methodology.

3 Commented Examples

In order to test the relevance of our method, we have simulated its main steps on some artificial examples. It is fundamental in our work to compute generalization formulas that may have exceptions. This can be realized in ILP systems (like FOIL [21] for instance) by allowing a certain level of noise. But it is difficult to adjust this parameter: if we accept a high level of noise, we find too general formulas, if the level of noise is too weak, the generalization is too specific or impossible because of the exceptions. To avoid this difficulty that must be studied carefully for each application domain, we use a generalization mechanism that rely only on positive examples. The ILP system Progol [16] has the ability to learn from positive data only [17] and we use it as the generalization tool described by the function **Gen** in our algorithm. Progol is an ILP system based on inverse entailment. The input file for Progol specifies the set of positive examples and the initial background theory that may contain definite Horn clauses but also integrity constraints expressed by headless Horn clauses. Moreover, the user specifies type and mode declarations for the predicates. These biases are very important to determine the space of possible generalizations that Progol searches with an A*-like algorithm in order to return a clause that realizes the best data compression.

In the following examples, the different stages of our method have been simulated by switching learning steps of p and $\neg p$. When learning p , the theory with only E^+ was considered and in order to learn $\neg p$, the negative examples are considered with $\neg p$ renamed in an ad-hoc predicate *not_p*. The covering tests, that are necessary to determine which examples are not yet generalized and also to determine exceptions to a default, require either extension calculus or query answering in Reiter's default logic. For both tasks, operational systems exist (for instance DeRes [4], GADEL [19], XRay [20]), and they could be integrated in a whole system for default theory learning.

Example 1. The initial theory W concerns a set of people and a set of dishes³.

$$W = \left\{ \begin{array}{l} hb(1), \dots, hb(45), hb(46), \dots, hb(50), hb(51), \dots, hb(55) \\ v(46), \dots, v(50), diab(51), \dots, diab(55) \\ a(mutton), a(bee\ f), a(fish), \\ di(mutton), di(bee\ f), di(fish), \\ oa(egg), oa(milk), di(egg), di(milk), \\ sug(ice_cream), sug(cake), di(ice_cream), di(cake) \end{array} \right\}$$

The aim is to induce what people eat and what they do not eat from the following sets of examples.

$$E^+ = \left\{ \begin{array}{l} eats(2, egg), \dots, eats(50, egg), \\ eats(1, milk), \dots, eats(50, milk), eats(1, mutton), \dots, eats(45, mutton), \\ eats(1, bee\ f), \dots, eats(45, bee\ f), eats(1, fish), \dots, eats(45, fish) \end{array} \right\}$$

$$E^- = \left\{ \begin{array}{l} \neg eats(1, egg), \\ \neg eats(46, mutton), \dots, \neg eats(50, mutton), \\ \neg eats(46, bee\ f), \dots, \neg eats(50, bee\ f), \\ \neg eats(46, fish), \dots, \neg eats(50, fish), \\ \neg eats(51, ice_cream), \dots, \neg eats(55, ice_cream), \\ \neg eats(51, cake), \dots, \neg eats(55, cake) \end{array} \right\}$$

Let us suppose that we begin to learn the definition of $eats(X, Y)$. So we run Progol in order to generalize from the examples E^+ and W . The best clause according to Progol evaluation is $(eats(X, Y) :- hb(X), oa(Y))$, which means that all the persons eat dishes that have an animal origin (eggs and milk). From this formula we build into D' a first default $\delta_1 = \frac{hb(X) \wedge oa(Y) : eats(X, Y)}{eats(X, Y)}$. By examining the set of negative examples, we find that this default admits only one exception $\neg eats(1, egg)$, that cannot lead to a relevant generalization. So this exception $\neg eats(1, egg)$ is added to W' . There are still some positive examples that are not covered by (W', D') and a second call to the generalization of Progol returns the clause $(eats(X, Y) :- hb(X), a(Y))$. So we build the default $\delta_2 = \frac{hb(X) \wedge a(Y) : eats(X, Y)}{eats(X, Y)}$. But this default admits a set of exceptions $Exc_{\delta_2} = \{\neg eats(46, mutton), \dots, \neg eats(50, fish)\}$. In order to characterize these exceptions by a general formula, we submit this subset of examples to Progol (after a replacement of $\neg eats$ by not_eats). Progol returns the clause $(not_eats(X, Y) :- v(X) \wedge a(Y))$. So δ_2 is specialized into $\delta'_2 = \frac{hb(X) \wedge a(Y) : eats(X, Y) \wedge \neg(v(X) \wedge a(Y))}{eats(X, Y)}$ and at the same time, we build $\delta_3 = \frac{v(X) \wedge a(Y) : \neg eats(X, Y)}{\neg eats(X, Y)}$. This default δ_3 admits no exception.

At this moment, we have finished the covering of all the positive examples and we consider the negative examples that are not yet covered by $(W', \{\delta_1, \delta'_2, \delta_3\})$, namely $\{\neg eats(51, ice_cream), \dots, \neg eats(55, cake)\}$. To generalize these instances, Progol finds the formula $(diab(X) \wedge sug(Y))$ and we build a default δ_4 with this formula as prerequisite. To take into account the whole job that has been realised during the learning of the positive part $eats(X, Y)$, this default has a justi-

³ The following notations are used: hb stands for human_being, v for vegetarian, di for dish and $diab$ for diabetic; a for animal qualifies dishes that are animal flesh, and oa for animal_origin qualifies dishes that have an animal origin, sug qualifies sugary food.

fication which is the conjunct of the prerequisites of defaults defining $eats(X, Y)$: $\delta_4 = \frac{diab(X) \wedge sug(Y) : \neg eats(X, Y) \wedge \neg (hb(X) \wedge oa(Y)) \wedge \neg (hb(X) \wedge a(Y))}{\neg eats(X, Y)}$. The justification in δ_4 is simplified by checking that there does not exist a couple (X, Y) such that $eats(X, Y) \in E^+$ and $(diab(X) \wedge sug(Y))$ and $(hb(X) \wedge oa(Y))$ are true simultaneously. So $(hb(X) \wedge oa(Y))$ is removed from the justification of δ_4 . The same is true for $(hb(X) \wedge a(Y))$ and finally, we obtain $\delta'_4 = \frac{diab(X) \wedge sug(Y) : \neg eats(X, Y)}{\neg eats(X, Y)}$. The simplification process relies on theorem proving in Horn logic and is much less expensive than the computation of exceptions that requires theorem proving in default logic. One can easily check that all the positive and all the negative examples are skeptical theorems of $(W', \{\delta_1, \delta'_2, \delta_3, \delta'_4\})$.

The following example illustrates that a learned default theory may have multiple extensions.

Example 2. Let us consider that we want to learn the predicate p^4 with $W = \{q(b1), q(b2), q(b3), q(nixon), r(t1), r(t2), r(nixon)\}$, $E^+ = \{p(b1), p(b2), p(b3)\}$ and $E^- = \{\neg p(t1), \neg p(t2)\}$.

Let us note that *nixon* is not given as a positive example nor as a negative one. So the simplification step applies to the second default, inducing $D' = \{\frac{q(X) : p(X)}{p(X)}, \frac{r(X) : \neg p(X)}{\neg p(X)}\}$. As it is required by our definition, the conjunct of all the examples is a skeptical theorem of (W, D') even if this theory has two distinct extensions $E_1 = Th(W \cup \{p(b1), p(b2), p(b3), \neg p(t1), \neg p(t2), p(nixon)\})$ and $E_2 = Th(W \cup \{p(b1), p(b2), p(b3), \neg p(t1), \neg p(t2), \neg p(nixon)\})$. Knowledge about *nixon* remains undefined since it is not a training example.

4 Learning With Initial Defaults

In both previous sections we consider special default theories where W only contains ground facts. This requirement was necessary to make a bridge between default logic where the sure knowledge can be expressed by any first order formula and ILP where the initial background knowledge is expressed by Prolog clauses, that are not equivalent to implications. In the case of the example 1, the whole initial theory is expressed by ground facts, whereas some general Prolog clause like $(hb(X) :- v(X))$ could have been used. Let us notice that such an oriented rule could be written in default logic by $\frac{v(X) : true}{hb(X)}$.

We consider now that we want to learn a new concept from an initial default theory and a set of examples. The initial default theory may have multiple extensions, but this difficulty can be resolved if the learning process relies only on the sure initial knowledge. That is why we propose a method where generalization uses a background knowledge including only all the ground facts that are skeptical theorems of our initial theory. This new learning problem can be stated as followed.

Definition 4. *Let E^+ and E^- be positive and negative examples of a target predicate p . Let (W_0, D_0) be an initial default theory containing no occurrences of p or $\neg p$.*

⁴ r stands for republican, q for quaker and p for pacifist.

Learning a default theory for the concept described by p and $\neg p$ consists to build a default theory (W', D') such that:

- $D' = D_0 \cup D_p$, where D_p is a set of defaults, the consequents of which are p or $\neg p$
- $W' = W_0 \cup E_p$, where E_p is a set of examples that cannot be generalized
- $(\bigwedge_{e \in E^+} e) \wedge (\bigwedge_{e \in E^-} \neg e)$ is a skeptical theorem of (W', D') .

We propose the following method to induce W' and D' in such a case. First, we compute W the set of ground facts that are skeptical theorems from the initial default theory (W_0, D_0) . Then the algorithm to learn the default theory (W', D') is the same as the one given in subsection 2.1, except the two following modifications:

- **DefaultLearning** works on the inputs: $p(X), W, W_0, D_0, E^+, E^-$
- the two first initializations $W' \leftarrow W$ and $D' \leftarrow \emptyset$ are replaced by

$$W' \leftarrow W_0 \quad D' \leftarrow D_0$$

So the background knowledge used for generalization by **Gen** is always W , the set of skeptical theorems of (W_0, D_0) . But each time we have to compute a set of exceptions, we consider the exceptions of the current theory (W', D') . This current theory contains the initial default theory (W_0, D_0) augmented by some new defaults defining p or $\neg p$ and eventually by some examples that cannot be generalized. So, generalization relies on sure knowledge but the search of exceptions takes into account the credulous theorems of (W_0, D_0) . This is necessary to insure that each example will be a skeptical theorem of the resulting default theory (W', D') .

Example 3. Let us consider the initial theory (W_0, D_0) with $W_0 = \{q(b1), q(b2), q(nixon), r(t1), r(t2), r(nixon), usp(nixon), p(john)\}$ and $D_0 = \{\frac{q(X):p(X)}{p(X)}, \frac{r(X):\neg p(X)}{\neg p(X)}\}$. Let $E^+ = \{no(b1), no(b2), no(john)\}$ ⁵ and $E^- = \{\neg no(t1), \neg no(t2), \neg no(nixon)\}$

The initial theory (W_0, D_0) has two extensions and we consider only the set W of ground facts that are skeptical theorems in order to learn a definition of no . As $W = W_0 \cup \{p(b1), p(b2), \neg p(t1), \neg p(t2)\}$, our method constructs the first default $\delta_1 = \frac{p(X):no(X)}{no(X)}$. The negative example $\neg no(nixon)$ is an exception for δ_1 since there exists an extension where δ_1 can be applied to $nixon$. A generalization of this exception leads to the formula $usp(X)$. Then δ_1 is specialized into $\delta'_1 = \frac{p(X):no(X) \wedge \neg usp(X)}{no(X)}$ and at the same time we build the default $\delta_2 = \frac{usp(X):\neg no(X)}{\neg no(X)}$ that has no exceptions. The learning process completes the definition of $\neg no$ by the default $\delta_3 = \frac{\neg p(X):\neg no(X)}{\neg no(X)}$. We can check that each example is a skeptical theorem of (W', D') with $W' = W_0$ and $D' = D_0 \cup \{\delta'_1, \delta_2, \delta_3\}$. This final resulting default theory (W', D') has two extensions because of the remained incomplete specification about $p(nixon)$ and $\neg p(nixon)$. But, each of these extensions contains the conclusion $\neg no(nixon)$ as it is required by our objective.

⁵ no stands for nuclear_opponent and usp stands for US President.

5 Related Works

The problem of learning non-monotonic theories by learning both a concept and its negation has been pointed as very interesting for many years. In [5] a concept and its negation are effectively learned, but in the framework of definite clauses : the negative concept is represented by a new predicate *not_p* and the learning algorithm checks that no contradiction occurs between the definitions of *p* and *not_p*. The framework proposed in [8] learns a concept and its exceptions by means of general rules, and conflicts between rules are solved by additional priority relations. This framework captures the notion of *specificity* of a rule as it is done in [3] in *prioritized default logic*. But, it is known [23] that specificity can be handled by means of *semi-normal* defaults and that is exactly what our method does. In [7] the problem of contradiction between definition of *p* and $\neg p$ is solved by using integrity constraints in order to restrict the conclusions derivable from too general rules.

More recently, some works deal with this problem in the context of extended logic programs [11, 13, 12]. Extended Logic Programs (ELP) have been introduced by Gelfond and Lifschitz [10] to extend the class of normal logic programs by allowing explicit negation. A rule in an ELP has the form $L_0 \leftarrow L_1, \dots, L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$, where each L_i is a literal (positive or negative). [11, 13] propose methods to learn an ELP that contains a definition of *p* and a definition of $\neg p$. Each definition may have exceptions that are described by abnormality predicates, and these abnormality predicates are defined by normal clauses. So, the aim of these works is the same as ours. The main difference is that we do not rely on abnormality predicates to specialize overgeneral rules. For instance, the algorithm presented in [13] learns rules for *p* and specializes them if they have exceptions, then it computes on the same manner a set of rules for $\neg p$. For our example 3, the following rules

$$\begin{array}{lll} p(X) : -q(X), \text{not } \neg p(X). & \text{abl}(X) :- \text{usp}(X). & \neg p(X) : -r(X), \text{not } p(X). \\ no(X) : -p(X), \text{not } \text{abl}(X). & & \neg no(X) : -usp(X). \end{array}$$

are learned. We can observe that the algorithm has dealt twice with the set of US presidents, once when US presidents are considered as a characterization of *abl* and another time when US presidents are considered as examples of the concept $\neg no$. This illustrates that using abnormality predicates to specialize rules hides the deep relationships that exist between definitions of *no* and $\neg no$ and this leads to redundancy in the learning process and in the resulting rules. Furthermore, the complete theory induced by [13] would in fact transform the rule defining *no* into the two rules : $(no(X) : -p(X), \text{not } \text{abl}(X), \text{not } \neg no(X).)$ and $(no(X) : -p(X), \text{undefined}(\neg no(X)).)$ and similarly for the other rules concluding $\neg no(X)$. The well founded semantics requires these modifications in order to deal correctly with the examples where the definitions of *no* and $\neg no$ overlap.

The method we have presented, like those described in [11, 13, 12], is a method to build a consistent theory in a non-monotonic framework. The common feature of our work and those presented in [11, 13, 12] is to rely on a standard ILP procedure to compute definitions for the positive and the negative parts of the concept; then we use a theorem prover for default logic (a theorem prover for the

answer set semantics in [11] and for WFSX semantics in [13, 12]) to compute the potential exceptions to the definitions that have been induced. So, the central point is to study for each used semantics how to aggregate these definitions into non monotonic rules able to deal with potential contradictions.

A more recent work [25, 24] presents another approach where the induction of hypotheses is realized directly from the answer sets of the initial program. So this work redefines the learning process accordingly to the framework used. In the case of a background program having multiple answer sets, the author proposes to learn different rules for each answer set, which is very different from our proposition of part 4. Of course, further study and also experimentation of those formalisms on real problems are necessary to decide whether induction must rely on credulous or skeptical knowledge.

6 Conclusion

We have presented a framework to induce default theories from training examples. Default logic is probably the most general framework that we can imagine to represent at the same time a concept and its negation and the recent tools realized for extension calculus or query answering enable to consider its application to some real domains. This paper has shown how to control the inductive construction of a default theory to insure that it correctly represents the knowledge contained in the training examples. The availability of ILP systems allowed us to check the relevance of this approach on some artificial examples. We have now to further study the generalization process, specially on real examples. We think that our method can be the basis of a system that helps a user to formalize its knowledge in default logic.

References

1. M. Bain and S. Muggleton. Non-monotonic learning. In S. Muggleton, editor, *Inductive Logic Programming*, pages 145–161. Academic Press, 1992.
2. F. Bergadano, D. Gunetti, M. Nicosia, and G. Ruffo. Learning logic programs with negation as failure. In L. De Raedt, editor, *Proceedings of Inductive Logic Programming conference*, pages 33–51, K.U. Leuven, 1995.
3. G. Brewka. Adding priorities and specificity to default logic. In L. Pereira and D. Pearce, editors, *European Workshop on Logics in Artificial Intelligence (JELIA'94)*, Lecture Notes in Artificial Intelligence, pages 247–260. Springer Verlag, 1994.
4. P. Cholewiński, V. Marek, A. Mikitiuk, and M. Truszczyński. Computing with default logic. *Artificial Intelligence*, 112:105–146, 1999.
5. L. De Raedt and M. Bruynooghe. On negation and three-valued logic in interactive concept learning. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 207–212. Pitman, 1990.
6. L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In R. Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1058–1063. Morgan Kaufmann Publishers, 1993.

7. Y. Dimopoulos, S. Džeroski, and A. Kakas. Integrating explanatory and descriptive learning in ILP. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2, pages 900–906. Morgan Kaufmann Publishers, 1997.
8. Y. Dimopoulos and A. Kakas. Learning non-monotonic logic programs: Learning exceptions. In N. Lavrač and S. Wrobel, editors, *European Conference on Machine Learning '95*, volume 912 of *Lecture Notes in Artificial Intelligence*, pages 122–137. Springer Verlag, 1995.
9. B. Duval and P. Nicolas. Learning default theories. In S. Parsons A. Hunter, editor, *Qualitative and Quantitative Approaches to Reasoning with Uncertainty*, volume 1638 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1999.
10. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):363–385, 1991.
11. K. Inoue and Y. Kudoh. Learning Extended Logic Programs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 1, pages 176–181. Morgan Kaufmann Publishers, 1997.
12. E. Lamma, F. Riguzzi, and L. M. Pereira. Strategies in combined learning via logic programs. *Machine Learning*, 38(1/2):63–87, January 2000.
13. E. Lamma, F. Riguzzi, and L.M. Pereira. Learning with extended logic programs. In *Workshop Learning Programming and non monotonic reasoning. Principles of Knowledge Representation and Reasoning*, Trento, 1998.
14. T. Linke and T. Schaub. Alternative foundations for Reiter's default logic. *Artificial Intelligence*, 124:31–86, 2000.
15. L. Martin and C. Vrain. A three-valued framework for the induction of general logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 219–235. IOS Press, 1996.
16. S. Muggleton. Inverse entailment and Progol. *New generation Computing*, 13:245–286, 1995.
17. S. Muggleton. Learning from positive data. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, volume 1314 of *Lecture Notes in Artificial Intelligence*, pages 358–376. Springer-Verlag, 1996.
18. S. Muggleton and L. De Raedt. Inductive Logic programming: Theory and Methods. *Journal of Logic Programming*, 19/20:629–679, 1994.
19. P. Nicolas, F. Saubion, and I. Stéphan. Gadel : a genetic algorithm to compute default logic extensions. In *Proceedings of the European Conference on Artificial Intelligence*, pages 484–488, 2000.
20. P. Nicolas and T. Schaub. The XRay system : An implementation platform for local query-answering in default logics. In Simon Parsons Antony Hunter, editor, *Applications of Uncertainty Formalisms*, volume 1455 of *Lecture Notes in Computer Science*, pages 254–378. Springer Verlag, 1998.
21. J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
22. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.
23. R. Reiter and G. Criscuolo. On interacting defaults. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 270–276, 1981.
24. C Sakama. Learning by answer sets. In *AAAI Spring Symposium: Answer set programming*.
25. C Sakama. Inverse entailment in nonmonotonic logic programs. volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 209–224. Springer-Verlag, 2000.