Explicitly using default knowledge in concept learning: an extended Description Logics plus strict and default rules

Véronique Ventos[1], Pierre Brézellec[2], and Henry Soldano[3]

[1] Université Paris-Sud
LRI Bat. 490,
91405 Orsay, France
ventos@lri.fr
[2] Université de Versailles-St-Quentin
LGI Av. des Etats-Unis
78035, Versailles Cedex
brezel@genetique.uvsq.fr
[3] Université Paris-Nord
LIPN, av. J.B. Clément
93430 Villetaneuse, France
soldano@lipn.univ-paris13.fr

Abstract. This work concerns the use of default knowledge in concept learning from positive and negative examples. Two connectives are added to a description logics, C-CLASSIC, previously defined for concept learning. The new connectives (δ and ϵ) allow to express the idea that some properties of a given concept definition are default properties, and that some properties that should belong to the concept definition actually do not (these are excepted properties). When performing concept learning both hypotheses and examples are expressed in this new description logics but prior to learning, a saturation process using default and non default rules has to be applied to the examples in order to add default and excepted properties to their definition. As in the original C-CLASSIC, disjunctive learning is performed using a standard greedy set covering algorithm whose generalization operator is the Least Common Subsumer operator of C-CLASSIC$_{\delta\epsilon}$. We exemplify concept learning using default knowledge in this framework and show that explicitly expressing default knowledge may result in simpler concept definitions.

1 Introduction

The general aim of concept learning consists of inducing hypotheses from a set of examples of an unknown target concept. The choice of the concept (and therefore hypothesis) and example languages is very important in this framework. Inductive Logic Programming (ILP, [15, 17]) studies learning within the framework provided by clausal logic. However, the language is often restricted to Horn clauses for complexity reasons. Description Logics (DLs) are other restrictions of first-order logic[1] in which the subsumption computation and its complexity have been deeply studied [11]. Several ILP approaches presented a learning framework where the learned theories or the entailment relation are non-monotonic (e.g. [1, 9]) in order to put emphasis on the problems that can not be captured by classical definite logic programs. We propose here to learn in a framework combining rules, default rules and a description logics allowing to handle default knowledge. In our approach, concept learning is performed in two steps. First, we use both default rules and strict rules in order to extend the definitions of the examples by adding default properties and excepted properties. Then, concept learning is performed using subsumption and Least Common Subsumer algorithms that handle examples and hypothesis including such default and excepted properties.

The description logics used here, C-CLASSIC$_{\delta\epsilon}$ [23, 22], extends C-CLASSIC with two non classical connectives (δ,ϵ) used to represent default knowledge. C-CLASSIC is equivalent to CLASSIC [2] without the SAME-AS connective and augmented by the connectives MIN and MAX which allow set of reals to be defined. C-CLASSIC is one of the most expressive previously known tractable DL, which preserves its good computational properties both for subsumption and PAC-learnability. The connective δ intuitively represents the common notion of default. For instance, having $\delta Viviparous$ as a conjunct in the definition of the concept $Mammal$ states that mammals are generally viviparous. The connective ϵ is used to represent a property that is not present in the description of the concept or of the instance but that should be. Thus, for instance, being a mammal, an ornithorynchus[2] should be viviparous since mammals are generally viviparous. However, an ornithorynchus is "exceptional" w.r.t. this property (i.e. it has the property $Viviparous^{\epsilon}$) as it is an oviparous mammal.
Default and excepted properties can be deduced by applying default rules closely related to Reiter's normal defaults. For instance, the Reiter's normal default $\frac{Bird(x):Fly(x)}{Fly(x)}$ is interpreted as "if x is a bird and if it is consistent that x can fly then infer that x can fly". Note that this default rule handles in the same way particular birds that cannot fly, as penguins, and non flying animals as cats for instance: nothing is deduced. In our framework, the rule corresponding to the previous normal default is $Bird(x) \rightarrow_d Fly(x)$ which is interpreted as "if x is a bird and if it is consistent that x can fly then infer that x generally flies (δFly(x) is inferred) else infer that x is exceptional w.r.t. Fly (Fly^{ϵ}(x) is inferred). As a consequence, penguins are stated as exceptional w.r.t. Fly and nothing is deduced concerning cats.
Such default rules together with strict rules (i.e. non default rules) constitutes the background knowledge that will be used to extend the description of the examples prior to learning. We use non default rules in order to express incoherences (e.g. $Fly \sqcap$ *Inapt-to-fly* $\rightarrow \perp$) that will be used to add excepted properties.

As in the original C-CLASSIC, disjunctive learning is performed using a standard greedy set covering algorithm whose generalization operator is the Least

[1] Note that comparing the expressive power of DLs and restrictions of First Order Logic used in ILP is still an open issue [7].
[2] Ornithoryncus = duck-billed platypus.

Common Subsumer operator of C-CLASSIC$_{\delta\epsilon}$. The computation of the subsumption relation of C-CLASSIC$_{\delta\epsilon}$, that is used to check whether a hypothesis covers an example, has been proved to be correct, complete and polynomial. Furthermore, C-CLASSIC$_{\delta\epsilon}$ is PAC-learnable [23, 21, 25].

This paper is organized as follows. Section 2 gives some needed background information on the C-CLASSIC$_{\delta\epsilon}$ description logics. Learning and saturation process in C-CLASSIC$_{\delta\epsilon}$ are described in sections 3 and 4 together with a comparison with learning in C-CLASSIC. Finally, in section 5 we briefly discuss related work in DLs and ILP fields and we present future work.

2 C-CLASSIC$_{\delta\epsilon}$

Description Logics are a family of knowledge representation formalisms which stem from KL-ONE [3]. Several systems have been built based on DLs (e.g. CLASSIC [2], FLEX [18]) and they have been used in real-world applications (e.g. CLASSIC in [20]). Besides, DLs facilitate the use of background knowledge and are more expressive than attribute-value representations. The field of DLs has received increased attention over the recent years in the Machine Learning community (e.g. [14, 7, 6]). These previous approaches used terminological languages unable to define concepts with default properties whereas allowing for default properties in concept definitions is frequently required in applications where few concepts can be strictly defined with necessary and sufficient properties [12].
In DLs, a concept is defined as a set of properties satisfied by individuals that are instances of the concept. These properties are expressed by terms that are built from atomic concepts and roles and from a set of connectives. Concepts are partially ordered by a subsumption relation which expresses the inclusion relation between concepts and is usually based on a standard model-based logical semantics. The subsumption relation in C-CLASSIC$_{\delta\epsilon}$ which is central for the learning task is presented in section 2.2. Knowledge is mainly separated into two components: a terminological component (T-box) which contains the definition of concepts and an assertional component (A-box) containing statements about individuals. We assume here that the A-box is empty since we represent the examples using the terminological language presented in section 2.1 (see section 3 for more details about examples). Section 2.3 presents the Least Common Subsumer operation in C-CLASSIC$_{\delta\epsilon}$ which is the generalization operator used during the learning process.

2.1 Terminological language

The connectives of C-CLASSIC$_{\delta\epsilon}$ are the connectives of C-CLASSIC [7] plus the connectives δ and ϵ introduced in $\mathcal{AL}_{\delta\epsilon}$ [8]. The terminological language of C-CLASSIC$_{\delta\epsilon}$ is defined using a set $\mathbf{R}$ of atomic roles, a set $\mathbf{P}$ of atomic concepts, the constants $\top$ and $\bot$, a set $\mathbf{I}$ of individuals (called *classic-individuals*), and the following syntactic rule (C and D are concepts, P is a atomic concept, R is a atomic role, u is a real, n is an integer and I_i are classic-individuals):

$C, D \rightarrow$	$\top$	the most general concept
	$\mid \bot$	the most specific concept
	$\mid P$	atomic concept
	$\mid ONE\text{-}OF\ \{I_1 \ldots I_n\}$	concept in extension
	$\mid MINu$	u is a real number
	$\mid MAXu$	u is a real number
	$\mid C \sqcap D$	concept conjunction
	$\mid \forall R : C$	value restriction
	$\mid R\ FILLS\ \{I_1 \ldots I_n\}$	subset of values for R
	$\mid R\ AT\text{-}LEAST\ n$	cardinality for R (minimum)
	$\mid R\ AT\text{-}MOST\ n$	cardinality for R (maximum)
	$\mid \delta C$	default concept
	$\mid C^{\epsilon}$	exception to the concept C

This syntactic rule is used to define *terms* of C-CLASSIC$_{\delta\epsilon}$.
Example: *Student* $\sqcap\ \delta($*publications AT-LEAST 4*$)\ \sqcap\ \forall$*age:MAX 27* $\sqcap$ *publications FILLS* $\{$*JAIR,AI*$\}$ $\sqcap\ \forall$*publications:* $(\forall$*year: ONE-OF*$\{$*97,98,99*$\})$ describes all the students who generally have at least four publications, are less than 27 years old, have at least one publication in JAIR and AI, and whose publications have been published in the years 97, 98 or 99.
Defining a concept[3] means giving a name A to a *term* T of the C-CLASSIC$_{\delta\epsilon}$ language using the expression $A \equiv C$.
Example: *Mammal* $\equiv$ *Animal* $\sqcap\ \delta Viviparous\ \sqcap$ *Vertebrate*
A T-box of C-CLASSIC$_{\delta\epsilon}$ is composed of concept definitions.

2.2 Subsumption in C-CLASSIC$_{\delta\epsilon}$

In DLs, concepts are organized in a taxonomy via a subsumption relation. Concerning the strict part of C-CLASSIC$_{\delta\epsilon}$, subsumption in C-CLASSIC$_{\delta\epsilon}$ is equivalent to subsumption in C-CLASSIC. More precisely, a concept C is subsumed by a concept D if C has (explicitly or implicitly) all properties of D. In our framework, we must distinguish strict and default properties. Roughly speaking, a concept C is subsumed by a default property if its definition contains either the default property, the strict property or the excepted property. For instance, δFly subsumes concepts having explicitly or implicitly either δFly, Fly or Fly^{ϵ} in their definition, while concepts whose definition does not mention anything (strict, default, exception, exception of exception ...) about Viviparous are not subsumed by $\delta Viviparous$.
Example :
Bird $\equiv$ *Animal* $\sqcap$ *Has-Wings* $\sqcap\ \delta Flies$ (a bird generally flies)
Penguin $\equiv$ *Animal* $\sqcap$ *Has-Wings* $\sqcap\ \delta(Flies^{\epsilon})$ (a penguin is generally exceptional w.r.t. Flies) $\sqcap\ \delta$*Inapt-to-fly* (a penguin is generally inapt to fly)
SuperPenguin $\equiv$ *Animal* $\sqcap$ *Has-Wings* $\sqcap\ (Flies^{\epsilon})^{\epsilon}$ (a Superpenguin is an exception to an exception since it is an exceptional Penguin) $\sqcap$ *Inapt-to-fly*$^{\epsilon}$ (a SuperPenguin is exceptional w.r.t. Inapt-to-Fly since it can fly)
With these definitions, *Bird* subsumes *Penguin* and *SuperPenguin* ($\delta Flies$ both subsumes $\delta(Flies^{\epsilon})$ and $(Flies^{\epsilon})^{\epsilon}$). *SuperPenguin* is subsumed by *Penguin* ($\delta(Flies^{\epsilon})$ subsumes $(Flies^{\epsilon})^{\epsilon}$ and δ *Inapt-to-fly* subsumes *Inapt-to-fly*$^{\epsilon}$). Note

[3] Note that cyclic concept definitions are not allowed.

that if *Bird* and *SuperPenguin* were defined with the strict property *Fly* and *Penguin* with the strict property *Inapt-to-fly*, *Penguin* would no more be subsumed by *Bird* and *SuperPenguin* would no more be subsumed by *Penguin*.
More formally, let C and D be two elements of C-CLASSIC$_{\delta\epsilon}$, $C \sqsubseteq D$, i.e. D subsumes C, iff C satisfies the strict properties of D, and satisfies or is explicitly "exceptional" w.r.t. the default properties of D.
The definition of the subsumption of C-CLASSIC$_{\delta\epsilon}$ is based on an "equational system" fully defined in [22] called EQ. EQ is a set of axioms fixing the main properties of the C-CLASSIC$_{\delta\epsilon}$ connectives (e.g. the axiom $A \sqcap B = B \sqcap A$ expresses the commutativity of concept conjunction, the axioms $A \sqcap \delta A = A$ and $A^{\epsilon} \sqcap \delta A = A^{\epsilon}$ express a subsumption relationship between A and δA (A is subsumed by δA) and between A^{ϵ} and δA (A^{ϵ} is subsumed by δA), the axiom $\delta\delta A = \delta A$ expresses the idempotence of δ).
Let $=_{EQ}$ denote the equality (modulo EQ axioms) between two *terms* of C-CLASSIC$_{\delta\epsilon}$. Subsumption in C-CLASSIC$_{\delta\epsilon}$ is defined as follows.

Definition 1 (Subsumption) *Let C and D be two elements of C-CLASSIC$_{\delta\epsilon}$, $C \sqsubseteq D$, i.e. D subsumes C, iff $C \sqcap D =_{EQ} C$.*

In DLs, the subsumption computation (for instance $C \sqsubseteq D$) is performed in two steps. First C and D are expanded (i.e. their definition is then exclusively made up of atomic concepts and roles). This expansion step allows us to take into account the background knowledge linked to the T-box. Then, a subsumption algorithm is applied on them.
In [23, 22], a polynomial-time subsumption algorithm for C-CLASSIC$_{\delta\epsilon}$ based on EQ has been proposed. The algorithm has been proved complete and correct. This subsumption is not a pure syntactic relation like θsubsumption. It is a semantic relation like logical implication or generalized subsumption [4]. Indeed, the subsumption takes into account the whole T-box which expresses a kind of background knowledge. In other words, the subsumption relation corresponds to logical implication within C-CLASSIC$_{\delta\epsilon}$.

2.3 Least Common Subsumer in C-CLASSIC$_{\delta\epsilon}$

As mentioned above, learning in C-CLASSIC$_{\delta\epsilon}$ relies both on the subsumption relation and on the computation of the Least Common Subsumer (LCS)[4] of two concept definitions. The definition of the LCS in C-CLASSIC$_{\delta\epsilon}$ is as follows:

Definition 2 (LCS in C-CLASSIC$_{\delta\epsilon}$) *LCS(A,B) $\rightarrow C \in$ C-CLASSIC$_{\delta\epsilon}$ if and only if $A \sqsubseteq C$ and $B \sqsubseteq C$ (C subsumes both A and B), $\nexists$ D, $D \in$ C-CLASSIC$_{\delta\epsilon}$ such that $A \sqsubseteq D$, $B \sqsubseteq D$ and D is strictly subsumed by C).*

An LCS algorithm has been designed for C-CLASSIC$_{\delta\epsilon}$ in [23, 24]. It has been proved that this algorithm is correct and polynomial, and that the LCS is unique.
Example:

[4] In the framework of DLs, the notion of Least Common Subsumer has been introduced by Borgida, Cohen and Hirsh in [5].

C $\equiv$ *Animal* $\sqcap$ *Vertebrate* $\sqcap$ *With-beak* $\sqcap$ *Oviparous* $\sqcap$ *Has-teats* $\sqcap$ Viviparous$^{\epsilon}$ $\sqcap$ $\forall$*weight:MIN 20* $\sqcap$ $\forall$*age:MAX 15*
D $\equiv$ *Animal* $\sqcap$ *Vertebrate* $\sqcap$ *Has-teats* $\sqcap$ *Viviparous* $\sqcap$ $\forall$*weight:MIN 10* $\sqcap$ $\forall$*age:MAX 10*
LCS(C,D) = *Animal* $\sqcap$ *Vertebrate* $\sqcap$ *Has-teats* $\sqcap$ δ*Viviparous* $\sqcap$ $\forall$*weight:MIN 10* $\sqcap$ $\forall$*age:MAX 15*

3 Learning in C-CLASSIC

Cohen and Hirsh [7, 6] give theoretical and experimental results on the learnability of description logics. In particular, they prove that C-CLASSIC is PAC-learnable.
From a practical point of view, the authors propose several algorithms allowing to learn concepts of C-CLASSIC from positive and negative examples of these concepts. The language of both concepts and examples is the terminological language of the DL.
The *covers relation* that specifies how hypotheses relate to examples is the subsumption relation: an hypothesis H covers an exemple e if and only if $e \sqsubseteq H$ (i.e. H subsumes e). The aim is then to find a hypothesis H that covers all positive examples (completeness) and none of the negative examples (consistency). As C-CLASSIC only contains a limited kind of disjunction (the ONE-OF connective), many target concepts of practical interest can not be expressed using a single term of C-CLASSIC. One way to overcome this limitation is to consider algorithms which learn a disjunction of terms rather than a single term, i.e. a hypothesis H such that $H \equiv \mathrm{H}_1 \vee \mathrm{H}_2 \ldots \vee \mathrm{H}_n$ (however, note that the connectives δ et ϵ allow to limit the number of disjuncts used to represent concepts (see section 4.3)). The cover of an example is then as follow :
If $H \equiv \mathrm{H}_1 \vee \mathrm{H}_2 \ldots \vee \mathrm{H}_n$ and e is a concept, then H covers e if and only if $\exists$ H_i, $e \sqsubseteq \mathrm{H}_i$ (i.e. H_i subsumes e)
The basic idea behind the LCSLEARNDISJ algorithm described in [7] is to use the LCS to implement a specific-to-general greedy search for hypotheses that cover many positive examples and no negative examples (this approach is similar to GOLEM [16] where multi-clause Prolog predicates are learned).
Example 1
Let $\mathrm{E}^{+}=\{\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3, \mathrm{e}_4\}$ be a set of positive examples of the concept to learn and $\mathrm{E}^{-}=\{\mathrm{ce}_1\}$ a set of negative examples of this concept.
$\mathrm{e}_1 \equiv$ *Animal* $\sqcap$ *Viviparous* $\sqcap$ *Vertebrate* $\sqcap$ *Barks.*
$\mathrm{e}_2 \equiv$ *Animal* $\sqcap$ *Vertebrate* $\sqcap$ *Oviparous* $\sqcap$ *Has-teats.*
$\mathrm{e}_3 \equiv$ *Animal* $\sqcap$ *Vertebrate* $\sqcap$ *Flies* $\sqcap$ *Quacks.*
$\mathrm{e}_4 \equiv$ *Animal* $\sqcap$ *Vertebrate* $\sqcap$ *Lives-in-Antartica* $\sqcap$ *Has-Wings* $\sqcap$ *Inapt-to-fly.*
ce'$_1$ $\equiv$ *Animal* $\sqcap$ *Vertebrate* $\sqcap$ *Lives-in-the-sea* $\sqcap$ *Scales.*
Results :
LCSLEARNDISJ computes the Least Common Subsumer of various subsets of positive examples. Since all the computed LCS (e.g. *Animal* $\sqcap$ *Vertebrate*) cover the negative example, no consistent generalization can be performed. As a conse-

quence, LCSLEARNDISJ returns the disjunction of the description of the four positive examples: (*Animal* ⊓ *Viviparous* ⊓ *Vertebrate* ⊓ *Barks*) ∨ (*Animal* ⊓ *Vertebrate* ⊓ *Oviparous* ⊓ *Has-teats*) ∨ (*Animal* ⊓ *Vertebrate* ⊓ *Flies* ⊓ *Quacks*) ∨ (*Animal* ⊓ *Vertebrate* ⊓ *Lives-in-Antartica* ⊓ *Has-Wings* ⊓ *Inapt-to-fly*).
In this example, we can see that for instance e_1 and e_2 have more in common than *Animal* ⊓ *Vertebrate* since e_1 is viviparous and e_2 has teats. However, the relationship between *Viviparous* and *Has-teats* can not be expressed in C-CLASSIC since it is not a strict knowledge (i.e. it is neither true that all animals that have teats are viviparous nor that all animals being viviparous have teats) and we can not add *Viviparous* to e_2 since it is oviparous (the same problem appears with e_3 that flies and e_4 which has wings but which is inapt to fly). In other words, e_2 and e_4 have exceptional properties but C-CLASSIC does not allow to express these exceptional properties. We show in section 4 how the saturation process allow to learn a more suited concept in C-CLASSIC$_{\delta\epsilon}$ without explictly expressing the exceptional properties of e_2 and e_4.

4 Learning in C-CLASSIC$_{\delta\epsilon}$

Learning in C-CLASSIC$_{\delta\epsilon}$ is similar to learning in C-CLASSIC. C-CLASSIC$_{\delta\epsilon}$ has been proved PAC-learnable [23, 21]. The same LCSLEARNDISJ algorithm can be used as polynomial subsumption and Least Common Subsumer algorithms have been defined on C-CLASSIC$_{\delta\epsilon}$.
However, the example definitions have to be saturated prior to learning using background knowledge. A part of this background knowledge is related to default and excepted properties and it is used to add such properties in the positive and negative examples. The learning problem for our framework is therefore defined as follows:
Given: a set of T-box statements, a finite set of rules[5] β (background knowledge), and sets of C-CLASSIC$_{\delta\epsilon}$ concepts E^+, E^- (positive and negative examples).
Build: sets of saturated examples E^+' and E^-' (E^+' and E^-' are the result of the saturation process linked to β and applied on E^+ and E^-).
Find: a hypothesis H (disjuncts of C-CLASSIC$_{\delta\epsilon}$ terms), such that H is complete w.r.t. E^+' and consistent w.r.t. E^-'.

4.1 Background knowledge

The background knowledge β is composed of two sets of rules (C and D are terms of C-CLASSIC$_{\delta\epsilon}$): a set *Def* of default rules in the form C $\rightarrow_d$ D meaning that if a concept is subsumed by C, it is generally subsumed by D, together with a set *Inc* of strict incoherence rules in the form C $\rightarrow \perp$ meaning that if a concept is subsumed by C, it is incoherent. More precisely, the definition of *Def* and *Inc* are the following:

[5] The syntax of these rules is defined in section 4.1.

Definition 3 (Def) *Def is composed of m rules called $R_1,\ldots,R_m$ such that R_i = precondition$_i \rightarrow_d$ Conclusion$_i$ where precondition$_i$ is a term of C-CLASSIC*[6] *and Conclusion$_i$ a term of C-CLASSIC where the only allowed concept conjunctions are in the value restriction of roles.*

For instance, the rule *Animal* $\sqcap$ *Has-teats* $\rightarrow_d$ *Viviparous* means that generally animals having teats are viviparous.

Definition 4 (Inc) *Inc is composed of n rules called $R_1,\ldots,R_n$ such that R_i = precondition$_i \rightarrow \perp$ where precondition$_i$ is a term of C-CLASSIC.*

Definition 5 *A rule of Def or Inc is applicable if its precondition subsumes the example.*

4.2 Saturation process

One of the main operations of the saturation process is to detect a potential incoherence between the definition of an example and the conclusion of an applicable default rule in order to add a default property (none incoherence has been detected) or an excepted property (an incoherence has been detected) to the example.
We distinguish two kinds of incoherences: incoherences of type 1 and incoherences of type 2.
An incoherence of type 1 corresponds to an incoherence linked to one or more general axioms concerning the connectives of the language. For instance, *child AT-LEAST* 2 $\sqcap$ *child AT-MOST* 1 is incoherent and more generally for all role R, R *AT-LEAST* m $\sqcap$ R *AT-MOST* n is incoherent if m ¿ n.
An incoherence of type 2 corresponds to an incoherence linked to a rule belonging to *Inc* (e.g. *Inapt-to-fly* $\sqcap$ *Flies* is incoherent).
It must be highlighted that in our framework incoherences are only linked to strict knowledge. Indeed, a default property cannot be incoherent. This is the reason why when there is no conflict between the conclusion of the default rule and the current description of e' we add δConclusion rather than Conclusion which could later be in conflict with knowledge issued from other rules.
Thus, for instance the concept C defined by C $\equiv$ δ(child AT-LEAST 2) $\sqcap$ δ(child AT-MOST 1) is not incoherent since the individual whose definition is δ(child AT-LEAST 2) $\sqcap$ δ(child AT-MOST 1) $\sqcap$ (child AT-LEAST 2)$^\epsilon$ is an instance of C. Besides, an excepted property never leads to an incoherence since an exception to a concept is not a negation of this concept. For instance, Fly$^\epsilon$ $\sqcap$ Fly is not incoherent.
The fact that incoherences are linked only to strict knowledge has two implications. First, we can state that a term T of C-CLASSIC$_{\delta\epsilon}$ is incoherent if and only if the term T' equivalent to T without default and excepted properties is

[6] We could extend the process by using terms of C-CLASSIC$_{\delta\epsilon}$ in the theory but our goal is to show that we can obtain default and excepted properties from rules whose precondition and conclusion are described using strict properties.

incoherent. Thus, incoherence of type 1 can be detected by translating a term of C-CLASSIC$_{\delta\epsilon}$ into a term of C-CLASSIC (i.e. by removing default and excepted properties) and by applying the normalization procedure defined for C-CLASSIC in [23][7].
Besides, it allows us to be sure that the addition of default and excepted properties will not lead to a future incoherence. This guarantees the monotonicity of the extension process.
The sketch of the extension algorithm is the following. Let e be an example described by a term C of C-CLASSIC$_{\delta\epsilon}$, for each rule of *Def* we check whether C is subsumed by the premisse of the rule. In order to achieve this task default and excepted properties of C are removed and C' the term of C-CLASSIC obtained is compared with the premisse of the rule by applying the subsumption algorithm designed for C-CLASSIC. If the premisse subsumes C', we check if the conclusion of the default rule is incoherent with C' (i.e. if it leads to incoherences of type 1 or 2). Incoherences of type 1 are detected by computing the normal form of C' $\sqcap$ Conclusion using the normalization algorithm of C-CLASSIC terms. If this normal form is equivalent to the denotation of $\perp$, there is an incoherence. Incoherences of type 2 are detected by verifying whether a premisse of a rule belonging to *Inc* subsumes C' $\sqcap$ Conclusion. If any incoherence is detected the conclusion is excepted and added to C (i.e. Conclusion$^{\epsilon}$ is added to C) otherwise the conclusion of the rule by default (i.e. δConclusion) is added to C.

The extension algorithm is as follows :

Inputs: a term C of C-CLASSIC$_{\delta\epsilon}$, a set Def $=\{R_1,\ldots,R_n\}$ of "default rules", a set Inc $=\{R'_1,\ldots,R'_m\}$ of strict incoherence incoherent rules.

Output: ENF-C the extended normal form of C.

External procedures used:
Remove$\delta\epsilon$(d): transforms a term d of C-CLASSIC$_{\delta\epsilon}$ into a term of C-CLASSIC by removing default and excepted properties of d (since incoherences concern only strict properties).
Subsume(C,D): returns **true** if C subsumes D, C and D being two terms of C-CLASSIC.
NF'(d): computes the normal form of a term d of C-CLASSIC.
BEGIN
C' $\leftarrow$ Remove$\delta\epsilon$(C)
  For all R_i of Def such that Premisse$_i$ $\rightarrow$ Conclusion$_i$ and Subsume(Premisse$_i$,C')
  begin
  Add $\leftarrow$ **false** {* Add is true if an excepted property has been added *}
  *** Search for incoherences of type 1**
  if NF'(C' $\sqcap$ Conclusion$_i$) = $\perp$ {* Conclusion$_i$ is incoherent with the description of C' and C *}
    then begin
    C $\leftarrow$ C $\sqcap$ (Conclusion $_i$)$^{\epsilon}$

[7] Applying this procedure on an incoherent term leads to normalize the term by $\perp$ which denotes incoherences.

Add $\leftarrow$ **true**
end
*** Search for incoherences of type 2**
if not Add then if there exists in Inc a premisse D such that Subsume(D,C' $\sqcap$ Conclusion$_i$)
then C $\leftarrow$ C $\sqcap$ (Conclusion$_i$)$^\epsilon$ else C $\leftarrow$ C $\sqcap$ δConclusion$_i$
end For all
END

Example
We consider example 1 described in section 3.
Let β a background knowledge made of two sets *Inc* and *Def*.
Def = {R1: *Animal* $\sqcap$ *Has-teats* $\rightarrow_d$ *Viviparous*, R2: *Animal* $\sqcap$ *Has-Wings* $\rightarrow_d$ *Flies*, R3: *Animal* $\sqcap$ *Lives-in-the-sea* $\sqcap$ *Scales* $\rightarrow_d$ *Gills* } is a set of default rules meaning that generally animals having teats are viviparous, that generally animals having wings fly and that generally animals with scales and living in the sea have gills.
Inc = {R4: *Viviparous* $\sqcap$ *Oviparous* $\rightarrow \perp$, R5: *Inapt-to-fly* $\sqcap$ *Flies* $\rightarrow \perp$} is a set of strict rules respectively meaning that an example can not be both oviparous and viviparous and that it is impossible to fly and to be inapt to fly. We illustrate now the saturation process on the example 1.
e'$_1$ $\equiv$ e$_1$
e'$_2$ $\equiv$ e$_2$ $\sqcap$ *Viviparous*$^\epsilon$
e'$_3$ $\equiv$ e$_3$
e'$_4$ $\equiv$ e$_4$ $\sqcap$ *Flies*$^\epsilon$.
ce'$_1$ $\equiv$ ce$_1$ $\sqcap$ δ*Gills*.
Some explanations about the saturation of e$_2$:
e$_2$ verifies (i.e. is subsumed by) the precondition of R1. The addition of *Viviparous* to e$_2$ leads to an incoherence (*Viviparous* $\sqcap$ *Oviparous* is subsumed by $\perp$ from R4). The property *Viviparous*$^\epsilon$ is added to e$_2$. Adding this property makes it possible to highlight that a part of e$_2$ (*Oviparous*) is incoherent with *Animal* $\sqcap$ *Has-teats* $\rightarrow_d$ *Viviparous*. This information can be useful during the learning process described in the next section.

4.3 Learning in C-CLASSIC$_{\delta\epsilon}$ vs C-CLASSIC

Using the example 1 and the background knowledge described in the previous section, we show now that, given the same positive and negative examples, C-CLASSIC$_{\delta\epsilon}$ allows to learn disjunctive concepts represented with less disjuncts than concepts learned in C-CLASSIC.
LCSLEARNDISJ is applied on the saturated examples. The first disjunct learned by the algorithm is LCS(e'$_1$,e'$_2$) (i.e. *Animal* $\sqcap$ δ*Viviparous*[8] $\sqcap$ *Vertebrate*).

[8] Note that this property does not belong to the LCS computed from the C-CLASSIC definitions of e$_1$ and e$_2$. Now, this property is crucial since it prevents the negative example to be subsumed (let us remind that ce'$_1$ has the properties *Animal* $\sqcap$ *Vertebrate*).

The examples e'$_1$ and e'$_2$ are then removed from the learning set. The next learned disjunct is LCS(e'$_3$,e'$_4$) that covers these two positives examples and no negative examples. The algorithm returns the following hypothesis: $(Animal \sqcap \delta\, Viviparous \sqcap Vertebrate) \vee (Animal \sqcap \delta Flies \sqcap Vertebrate)$.
The following instance: $e \equiv Animal \sqcap Vertebrate \sqcap Lives\text{-}in\text{-}Australia \sqcap Wings \sqcap Big\text{-}feet \sqcap Inapt\text{-}to\text{-}fly$ is recognized by the definition learned in C-CLASSIC$_{\delta\epsilon}$: the saturation process adds $Flies^{\epsilon}$ to e and $Animal \sqcap \delta Flies \sqcap Vertebrate$ subsumes the saturated instance. Note that e is not recognized by the definition learned in C-CLASSIC (see section 3).

5 Related and further work

Cohen and Hirsh suggest that learning systems based on description logics may prove to be a useful complement to ILP systems. One issue is the investigation of combining our framework and non-monotonic frameworks in ILP. In [9], the authors present a framework for learning non-monotonic logic programs. Hence given a background theory and a set of examples they generate a hypothesis within the language bias of a subclass of non-monotonic logic programs[9] that covers all the positive examples and none of the negative examples. In such theories in order to decide if an atom, A, holds they need to show that A can be derived classically using some rule, r, for A and that ¬A can not be derived classically using some rule r' which is designated higher than the rule r by the priority relation on the program.
For instance, consider the background theory B:
bird(x) ← penguin(x)
penguin(x) ← superpenguin(x)
bird(a), bird(b), penguin(c), penguin(d), superpenguin(e), superpenguin(f)
Consider also the set of examples $E = E^+ \cup E^-$ where $E^+ = \{flies(a), flies(b), flies(e), flies(f)\}$ and $E^- = \{flies(c), flies(d)\}$.
The result of the algorithm is the hypothesis H :
R1 : *flies(x) ← bird(x)*
R2 : ¬ *flies(x) ← penguin(x)*
R3 : *flies(x) ← superpenguin(x)*
where R1 has lower priority than R2 and R2 has lower priority than R3.
In such a non-monotonic framework, the goal is to learn strict predicates (e.g. Fly whose penguin is a negative example) by generating default rules. This approach is not suited to learn strict concepts having default properties in their definition (e.g. $Bird$ whose penguin is a positive example despite the fact it is exceptional w.r.t. the Fly property). A further work could consist in using default rules learned in this non-monotonic framework (e.g. R1, R2, R3[10]) in order to

[9] Theories where their set of contradictory rules can be separated into classes where the rules in each class are totally ordered by the priority relation of the theory.

[10] The presence of ¬ is not a problem since it is straighforward to add the negation on atomic concepts in C-CLASSIC$_{\delta\epsilon}$ (and the axiom $A \sqcap \neg A \equiv \bot$ in the equational system in order to take into account such incoherences.

improve learning in C-CLASSIC$_{\delta\epsilon}$ (for instance, positive and negative examples of the concept *Bird* could be saturated with $\delta(Flies^{\epsilon})$ or $(Flies^{\epsilon})^{\epsilon}$ using R1, R2 and R3). Note that the problem of learning with a non-monotonic background knowledge is one of the possible directions for further research listed in [9]. Abduction [10,13] also is the basis for non-monotonic learning frameworks by providing a uniform technique to handle negation as failure, incomplete predicates and integrity constraints. We need also to compare our work with these approaches.

As a conclusion, in this paper we have defined a problem setting concerning learning concept in a framework combining a description logics allowing to define concepts with default and excepted properties, and a background knowledge represented by rules and default rules. We proposed a prior saturation of the examples using the background knowledge and we showed that learning from extended examples can lead to the construction of a more satisfactory learned concept. The presence of defaults is a way to improve the expressive power of the DL (few concepts can be defined with necessary and sufficient properties using only strict knowledge) and therefore to improve the learning process. Finally, note that the application of default rules is difficult since it can lead to ambiguities. For instance, in [19] the authors integrated defaults in DLs using incident rules of the form c1 $\rightarrow_d$ c2 meaning "whenever an object is an instance of c1 it is also an instance of c2 unless this is in conflict with some other piece of knowledge". This approach requires to handle multi extensions by defining preference criteria (e.g. the preferred models contain the most specific knowledge or the most applied defaults). In our framework, the connectives δ and ϵ allow us to avoid this problem.

References

1. M. Bain and S. Muggleton, 'Non-monotonic learning', in *Knowledge Representation and Organization in Machine Learning*, ed., Stephen Muggleton, 289–319, ACADEMIC PRESS LIMITED, (1992).
2. A. Borgida and P. F. Patel-Schneider, 'Complete algorithm for subsumption in the CLASSIC description logic', *Artificial Intelligence Research*, **1**, 278–308, (1994).
3. R.J. Brachman, 'A structural paradigm for representing knowledge', Technical Report 3605, BBN Report, (1978).
4. W. Buntine, 'Generalized subsumption and its application to induction and redundancy', *Artificial Intelligence*, **36**, (1988).
5. W. W. Cohen, A. Borgida, and H. Hirsh, 'Computing least common subsumers in description logics', in *10th National Conference of the American Association for Artificial Intelligence*, pp. 754–760, San Jose, California, (1992).
6. W. W. Cohen and H. Hirsh, 'The learnability of description logics with equality constraints', *Machine Learning*, **2**(4), 169–199, (1994).
7. W. W. Cohen and H. Hirsh, 'Learning the CLASSIC description logic: theoretical and experimental results', in *International Conference on Knowledge Representation and Reasoning*, 121–133, (1994).
8. P. Coupey and C. Fouqueré, 'Extending conceptual definitions with default knowledge', *Computational Intelligence*, **13**(2), (1997).

9. Y. Dimopoulos and A. Kakas, 'Learning non-monotonic logic programs: learning exceptions', in *Proceedings of the 11th European Conference on Artificial Intelligence, 1995*, ed., Springer-Verlag, pp. 107–121, (1995).
10. Y. Dimopoulos and A. Kakas, 'Abduction and inductive learning', in *Advances in Inductive Logic Programming*, ed., L. De Raedt, pp. 144–171, (1996).
11. F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt, 'The complexity of concept languages', in *Principles of Knowledge Representation and Reasoning: 2nd International Conference*, eds., J.A. Allen R. Fikes and E Sandewall, pp. 151–162, Cambridge, Mass., (1991).
12. J. Doyle and R.S. Patil, 'Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services', *Artificial Intelligence*, **48**(3), 261–297, (1991).
13. K. Inoue and C. Sakama, 'Abducing priorities to derive intended conclusions', in *16th International Joint Conference on Artificial Intelligence*, pp. 44–49, Japan, (1999).
14. J. U. Kietz and K. Morik, 'A polynomial approach to the constructive induction of structural knowledge', *Machine Learning*, **14**(2), 193–217, (1994).
15. S. Muggleton, 'Inductive logic programming', *New Generation Computing*, **8**, 295–318, (1991).
16. S. Muggleton and C. Feng, 'Efficient induction of logic programs', in *Inductive Logic programming*, ACADEMIC PRESS, (1992).
17. S. Muggleton and L. De Raedt, 'Inductive logic programming: Theory and methods', *Journal of Logic Programming*, **19**, (1994).
18. J. Quantz, G. Dunker, F. Bergmann, and I. Kellner, 'The flex system', Technical report, KIT-Report, Technische Universität, Berlin, Germany, (1996).
19. J. Quantz and V. Royer, 'A preference semantics for defaults in terminological logics', in *Principles of Knowledge Representation and Reasoning: 3rd International Conference*, 294–305, Bernhard Nebel, Charles Rich and William Swartout, Cambridge, MA., USA, (1992).
20. J.R. Wright, E.S. Weixelbaum, K. Brown, G.T. Vesonder, S.R. Palmer, J.I. Berman, and H.H. Moore, 'A knowledge-based configurator that supports sales, engineering and manufacturing at att bell network systems', in *Proceedings of the Innovative Applications of Artificial Intelligence Conferences*, pp. 183–193, Menlo Park, California, USA, (1993).
21. V. Ventos, P. Brézellec, P. Coupey, and H. Soldano, 'C-classic$_{\delta\epsilon}$: un langage de descriptions pac-learnable', in *Actes Journées Acquisition Validation Apprentissage, JAVA95*, pp. 192–196, Grenoble, France, (1995).
22. V. Ventos, 'A deductive study of c-classic$_{\delta\epsilon}$', in *Proceedings of the International Workshop in Description Logics 96*, pp. 192–196, Boston, USA, (1996).
23. V. Ventos, *C-CLASSIC$_{\delta\epsilon}$: une logique de descriptions pour la définition et l'apprentissage de concepts avec défauts et exceptions*, PhD thesis, France, 1997.
24. V. Ventos, P. Brézellec, P. Coupey, and H. Soldano, 'Lcs operation in c-classic$_{\delta\epsilon}$: formal properties and applications', in *International KRUSE Symposium Knowledge Retrieval, Use, and Storage for Efficiency*, pp. 124–135, Vancouver, Canada, (1997).
25. V. Ventos, P. Brézellec, H. Soldano, and D. Bouthinon 'Learning concepts in c-classic$_{\delta\epsilon}$', in *Proceedings of the International Workshop on Description Logics 98*, pp. 50–54, Trento, Italy, (1998).