Artificial Intelligence 53 (1992) 291–307
Elsevier

Research Note

Belief updating from integrity constraints and queries

Luc De Raedt and Maurice Bruynooghe

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

Received October 1990
Revised June 1991

Abstract

De Raedt, L. and M. Bruynooghe, Belief updating from integrity constraints and queries (Research Note), Artificial Intelligence 53 (1992) 291–307.

It is argued that the problems of intensional knowledge base updating and incremental concept-learning—when formulated in a logical framework—can be understood as instances of the more general problem of belief updating. This insight allows interesting cross-fertilization between both areas. To support this claim, we sketch a simple extension of Shapiro's Model Inference System that solves the belief updating problem within a restricted subset of first order logic. This extension uses integrity constraints and allows for the assertion of non-unit clauses. The former generalizes the use of examples in concept-learning whereas the latter generalizes the set of revisions considered in knowledge base updating.

Keywords. Concept-learning, knowledge base updating, integrity constraints, oracle, machine learning, inductive inference, data bases, knowledge bases, belief revision, belief updating.

1. Introduction

Recently, the logic programming community has shown much interest in the field of knowledge base updating [5, 7, 11, 12, 14, 23, 30]. Also, in machine learning there is a trend towards using logic or logic programming as a knowledge representation formalism (e.g. [4, 8, 21, 22, 29]). The fields of intensional knowledge base updating and incremental concept-learning have been developed independently from each other and from a different perspec-

tive. Reformulating them in a logical framework reveals that they are closely related. Because of this intimate relationship, we argue that they can benefit from applying each other's techniques. To support these claims, we first introduce a general notion of belief updating in a logical framework, and then argue that intensional knowledge base updating and incremental concept-learning can be viewed as special cases of the belief updating problem. This allows us to point out the similarities and differences between both areas. Finally, we show how the belief update problem can be solved—for a restricted subset of Prolog [16]—using a simple adaptation of Shapiro's Model Inference System [29]. The soundness and other properties of the algorithm are proven. It remains an open question whether the belief update problem can be solved for full pure Prolog. Although we are well aware that there are more advanced methods than Shapiro's one, we choose to adapt his method because it is well known, theoretically appealing and sufficient for supporting our claims.

This paper is organized as follows: in Section 2, a simple logical framework for knowledge bases is defined; in Section 3, the relation between incremental concept-learning and intensional knowledge base updating and the more general problem of belief updating is explored; in Section 4, the belief update problem is solved by adapting Shapiro's Model Inference System [29]; in Section 5, we study some theoretical properties of the proposed algorithm and finally, in Sections 6, 7 and 8, we conclude and discuss related work.

2. A logical framework for knowledge bases

We recall the most important notions used throughout this paper. More details can be found in [16].

A *clause* is a formula of the form $A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ where the A_i and B_j are positive literals (atomic formulas). The above clause can be read as A_1 or ... or A_m if B_1 and ... and B_n. Extending the usual convention for *definite clauses* (where $m = 1$), we call $A_1, \ldots, A_m$ the *head* and $B_1, \ldots, B_n$ the *body* of the clause. A *fact* is a definite clause with empty body ($m = 1$, $n = 0$). A *goal* is a clause with empty head ($m = 0$). A *general goal* is a formula of the form $\leftarrow C_1, \ldots, C_k$ where all C_i are either positive or negative literals.

Definition 2.1. A clause is *range-restricted* iff all variables occurring in the head also occur in the body.

Range-restriction is often imposed in the data-base literature (e.g. [10]); it allows us to avoid the derivation of non-ground true facts.

Definition 2.2. A variable is *linked in a clause* if it occurs in the head or if it occurs in a literal L of the body and L contains a linked variable.

Definition 2.3. A clause is *linked* iff all of its variables are linked.

Requiring clauses to be linked allows to avoid considering usually uninteresting clauses such as $p(X) \leftarrow q(X, Y)$, $r(Z)$ when learning concepts. This restriction is often imposed in learning systems, see e.g. [13].

We follow the database literature [6, 10, 31] in distinguishing between intensional and extensional predicates and disallowing functors.

Definition 2.4. An *extensional predicate* is a predicate defined by a set of functor-free ground facts.

Predicates that are not extensional are intensional. However, for the purpose of this paper, we impose some restrictions on their syntax.

Definition 2.5. An *intensional predicate* is a predicate defined by a set of linked, range-restricted, functor-free and constant-free definite clauses.

Notice that the definition of an intensional predicate may not contain facts.

Disallowing constants is not an additional restriction as, by introducing extra extensional predicates, one can easily rewrite clauses. E.g. $p(a, X) \leftarrow r(Y, c)$ can be rewritten as $p(A, X) \leftarrow r(Y, C), \bar{a}(A), \bar{c}(c)$ and the ground facts $\bar{a}(a) \leftarrow$ and $\bar{c}(C) \leftarrow$ [25]. Also, the range-restriction does not limit expressiveness in the absence of functors. For example, $p(X, Y) \leftarrow q(Y)$ can be rewritten as $p(X, Y) \leftarrow q(Y), d(X)$ with $d(X)$ an extensional predicate enumerating all constants.

Definition 2.6. A *knowledge base* contains definitions of a set of extensional and a set of intensional predicates.

In intensional knowledge base updating and incremental concept-learning, the aim is to derive a target-knowledge base fulfilling certain conditions (see Section 3). We postulate the existence of an *intended interpretation* (see also Section 6), which assigns truth and falsity to atoms in the Herbrand-base of the target knowledge base. We will assume that there is an oracle that is willing to answer certain questions about the intended interpretation (cf. Definitions 3.1 and 3.2).

In the literature, e.g. [10], it is usual to impose restrictions on the information contained in the knowledge base in the form of integrity constraints.

Definition 2.7. An *integrity constraint* is a range-restricted functor-free clause.

Integrity constraints need not be linked, definite or constant-free as they are considered to be correct in the intended interpretation; they are not eligible for revision during the learning phase.

Definition 2.8. An *integrity constraint theory* is a set of integrity constraints.

As a notational convention to distinguish clauses in the knowledge base from constraints in the constraint theory, we will write the former with an implication sign to the left ($\leftarrow$) and the latter with an implication sign to the right ($\rightarrow$).

Because the knowledge base KB contains only definite clauses, it has a single minimal Herbrand model: the least fixpoint of the well-known T_p operator (see [16] for more details). In the sequel of this paper, we consider an integrity constraint to be satisfied if it is true in this least Herbrand model, and to be violated otherwise. This is because we feel that the more classical views, which state that an integrity constraint should either be a theorem of $Comp(KB)$ (cf. [17]) or consistent with $Comp(KB)$ (cf. [27]) are not entirely satisfactory for our purposes. Consider the following knowledge base KB and the integrity constraints $IC1$ and $IC2$:

$$KB: p(a) \leftarrow p(a) \qquad IC1: \rightarrow p(a) \qquad IC2: p(a) \rightarrow \quad .$$

We have the $Comp(KB) \not\models p(a)$ and $Comp(KB) \not\models \neg p(a)$ and $IC1$ and $IC2$ are both consistent with $Comp(KB)$, but neither is a theorem of $Comp(KB)$. According to us, $IC1$ should be considered violated and $IC2$ satisfied. So, we take truth in the least Herbrand model (here: the empty set) as the criterion. Several evaluation strategies to compute this least Herbrand model have been proposed (see e.g. [2]). Under the given restrictions (definite clauses without functors), certain strategies mentioned by [2] assure termination. So, in order to verify whether a knowledge base KB satisfies an integrity constraint $IC = B_1, \ldots, B_n \rightarrow A_1, \ldots, A_m$, we execute the query $Q = \leftarrow B_1, \ldots, B_n, \neg A_1, \ldots, \neg A_m$. Because KB is range-restricted, the subquery $\leftarrow B_1, \ldots, B_n$ returns a finite number of ground substitutions θ. Since IC itself is range-restricted θ grounds all literal A_i. Therefore $\neg A_i\theta$ is true in the minimal Herbrand model iff $A_i\theta$ is false in it. The latter can be checked with the above-mentioned terminating evaluation strategies. So, there are procedures such that the evaluation of Q always terminates. We assume the use of such a procedure.

An answer θ to Q indicates that $(B_1, \ldots, B_n, \neg A_1, \ldots, \neg A_m)\theta$ holds in the minimal Herbrand model, in other words that the integrity constraint IC is violated under the substitution θ. Failure of Q occurs when IC is satisfied.

Definition 2.9. *KB satisfies an integrity constraint $IC = B_1, \ldots, B_n \rightarrow A_1, \ldots, A_m$* iff the query $Q = \leftarrow B_1, \ldots, B_n, \neg A_1, \ldots, \neg A_m$ fails. *IC violates KB with substitution θ* iff Q succeeds with answer-substitution θ.

Definition 2.10. A knowledge base *KB satisfies an integrity constraint theory IT* iff KB satisfies all constraints $IC \in IT$.

3. Problem-specification

In the sequel, we will present a method to automatically modify a knowledge base when violating a newly supplied integrity constraint. The method uses inductive learning techniques and relies on an oracle. The oracle must be able to answer existential and membership questions (see also [1, 29]).

Definition 3.1. A *membership question* asks for the truth-value of a ground fact. If the fact is true in the intended interpretation the oracle answers yes. Otherwise, it answers no.

Definition 3.2. An *existential question* asks for the truth-value of a non-ground fact p. The oracle must answer the question with no if there is no substitution θ for which $p\theta$ is true in the intended interpretation. Otherwise, it must return all ground substitutions θ for which $p\theta$ is true in the intended interpretation.

Answering membership and existential questions corresponds to supplying examples in inductive learning. These answers can be reformulated as integrity constraints. Indeed, an atom q, that is false in the intended interpretation, corresponds to a constraint of the form $(q\rightarrow)$; while an atom q, true in the intended interpretation corresponds to the constraint $(\rightarrow q)$. During the presentation of the algorithms, it will be assumed that all answers to queries are added to the integrity constraint theory.

We also define the notions of coverage and incorrectness [29], which will be used in our presentation of the algorithms.

Definition 3.3. A fact $p(t_1, \ldots, t_n)$ is *covered by the predicate p w.r.t. the intended interpretation* if there is a clause c for p in the knowledge base and a substitution θ such that $head(c)\theta = p(t_1, \ldots, t_n)$ and $body(c)\theta$ is true in the intended interpretation.

Definition 3.4. A clause c is *incorrect* if there is a substitution θ for which $body(c)\theta = true$ and $head(c)\theta = false$ in the intended interpretation.

The belief update problem can now be formalized as follows:

- *Given*:
  - a knowledge base KB,
  - an integrity constraint theory IT, satisfied by KB,
  - a new integrity constraint IC,
  - an oracle, willing to answer existential and membership questions;
- *Find*: a knowledge base KB', such that KB' satisfies $\{IC\} \cup IT$ and KB' is obtained by asserting/retracting any clause/fact from/to KB; KB' forms a *solution* to the belief update problem.

The belief update problem

We will now discuss the relation between incremental concept-learning and intensional knowledge base updating and argue that—from a certain perspective—the belief update problem generalizes both of these areas.

Firstly, let us note that incremental concept-learning (e.g. [22, 29]) and intensional knowledge base updating (e.g. [5, 11, 23]) have been studied in a much more expressive representation formalism which includes negation and functors. Certainly in knowledge base updating, as studied in the logic programming community, the expressiveness of the formalism has been a major focus of research. We are well aware that in this respect our definition of and our approach to the belief update problem is quite restricted. Nevertheless, it is easy to extend the *definition* of the belief update problem towards these more expressive formalisms. At the moment, it is—however—an open question whether there is an elegant solution to this extended belief update problem (see also Section 6).

Secondly, in concept-learning, one updates the knowledge base from examples. Examples in concept-learning are a very restricted form of integrity constraints: either they are true or false ground facts as in [8, 29] or they correspond to true or false ground definite clauses as in [22, 28]. In both cases, examples can be represented as constraints. A true (resp. false) definite clause $P \leftarrow Q_1, \ldots, Q_m$ corresponds to a constraint $Q_1, \ldots, Q_m \rightarrow P$ (resp. a denial $Q_1, \ldots, Q_m, P\rightarrow$).[1]

Thirdly, some concept-learning techniques may also generate other types of oracle-questions [1], use only membership-questions [8, 28] or use no questions at all [24]. Because concept-learning has been studied under these different assumptions, the used oracle-type seems not intrinsic to the nature of the concept-learning problem.

In contrast to our definition of belief updating, the above mentioned approaches to knowledge base updating do not rely on an oracle. Furthermore, in knowledge base updating one is usually interested in all possible updates, while in belief updating only one possible solution is wanted. To explain these differences between belief updating and intensional knowledge base updating, observe that in practice people are only interested in *one* (correct) update of their knowledge base. The mentioned approaches to knowledge base updating have divided this practical problem into two subproblems: (1) determining all correct modifications of the knowledge base and (2) choosing the desired modification among the solutions to problem (1). The first subproblem has received much attention in the literature, whereas the second has not. In general, there are a very large number of possible solutions to the first subproblem (Tomasic [30] proves that the number of solutions can be exponen-

[1] This is the interpretation that is intended by the mentioned work.

tial in the size of the knowledge base). Apparently, the second problem is the most difficult one. Furthermore, it is generally agreed that approaches to the second subproblem may have to consult the user. This consulting implicitly assumes an oracle, although it may be an oracle of a different kind. In our definition of belief updating, the original practical problem is addressed globally, rather than dividing it into subproblems. Also, by relying on an oracle, it is possible to filter out many uninteresting solutions: the use of an oracle increases the probability that the found solution is a good one. It also simplifies the search problem and reduces the number of possibilities. Without an oracle one has to explore two different paths: one corresponding to a positive answer on a question and the other to the negative answer. Upon reflection, it seems feasible to adapt the approaches to knowledge base updating towards the use of an oracle in our sense and to extend our technique with a backtracking mechanism to generate alternative solutions. As a consequence, we believe that the differences between belief updating and intensional knowledge base updating with regard to the use of an oracle are rather superficial.

Fourthly, none of the known approaches to the knowledge base update problem allows to assert non-unit clauses for intensional predicates in the knowledge base.[2] Nevertheless, this is often needed in practice [5]. The reason for the restriction seems to be that the proposed methods did not lead to any obvious way to know which clause(s) to add. In machine learning, however, several methods have been studied to induce new clauses (see e.g. [8, 22, 29]). In the sequel, we will show that these machine learning techniques can also be successfully employed in the context of knowledge base updating by also allowing for assertions of non-unit clauses.

One very important issue in our presentation is the import of a logical framework for concept-learning. It has to be noted that most classical approaches to concept-learning (see e.g. [15, 18, 19]) employ a much simpler formalism such as attribute-value pairs. It is only in the past few years that researchers have tried to extend the classical approaches to a first order logic framework (see e.g. [8, 21, 22, 28]).

To summarize this section: within a logical framework, we have that

(1) the intensional knowledge base update problem can be seen as a special case of (a possibly extended) belief updating problem because in belief updating it is also allowed to assert non-unit clauses in the knowledge base and
(2) the problem of incremental concept-learning is a special case of the problem of belief updating because of the generalization of examples into constraints.

[2] This is slightly exaggerated: Guessoum and Lloyd [11] consider modifying clauses by adding exceptions to it. However, this is a very special case of deriving new clauses.

These claims are to be understood at the level of the problem-specifications and not at the level of solutions to these problems! The reason is that it is not (yet) clear whether there exists an elegant solution to the generalized problem of belief updating, which also addresses the use of functors and negation. Nevertheless, in the next section, we will present a solution to the belief updating for the subset of first order logic presented in the previous section.

4. The method

Our method is based on the following two observations:

First, if a constraint $P_1, \ldots, P_n \rightarrow Q_1, \ldots, Q_k$ is violated for a substitution θ, then we have that $\exists\theta : P_1\theta, \ldots, P_n\theta, \neg Q_1\theta, \ldots, \neg Q_k\theta$ is true in the minimal Herbrand model of the knowledge base. Since the constraint is true in the intended interpretation, we must have that at least one of the $P_i\theta$ is false or at least one of the $Q_j\theta$ is true in the intended interpretation. This implies that there must be an error in the knowledge base KB. There are two possibilities for such errors. In one of the $P_i\theta$ is false in the intended interpretation, then our definition of the corresponding predicate P_i is too general, i.e. KB implies $P_i\theta$, but should not. If on the other hand, one of the Q_j is true in the intended interpretation, our definition of the corresponding predicate Q_j is too specific, i.e. KB does not imply $Q_j\theta$ but should.

Secondly, Shapiro [29] describes the Model Inference System, which is able to recover from three kinds of errors in logic programs: a true fact which is not implied by the knowledge base; a false fact, which is implied by the knowledge base, and a query which does not terminate. The first two types of error are precisely the same kind of error we can derive from a violated integrity constraint. The third type of error, non-termination, cannot arise in our framework (cf. the above).

Given the above observations, it is easy to design Algorithm 1 to solve the belief update problem. It starts from a given knowledge base KB and an integrity constraint theory IT. The while loop of the procedure learn is intended to handle multiple knowledge base updates. In order to find a violated integrity constraint and the associated substitution, we use a query evaluation strategy as referred to in Section 2. Given the violated constraint IC and the substitution θ, the error can be located by asking the oracle to classify the literals in $IC\theta$ until an error is found. Note that these questions are membership questions (this follows by induction from the range-restrictedness condition and the form of the clauses in the knowledge base). Also, the answers are added to the integrity constraint theory as they are true in the intended interpretation. Recovering from the error is then possible by invoking the generalization or specialization procedures of Shapiro [29]. The procedure specialize will make an implied false fact unimplied and the procedure general-

```
procedure learn
    initialize the set of marked clauses with {T}
    while not all integrity constraints have been input by the user do
        read the next integrity constraint IC
        add IC to the integrity constraint theory IT
        call kbu(KB,IT)
    endwhile
endproc

procedure kbu(KB : knowledge base, IT : constraint theory)
    while KB does not satisfies IT do
        select a violated integrity constraint IC
        find a substitution θ for which IC is violated
        repeat
            select a literal L of ICθ
            ask the membership question L to the oracle
            add L as a constraint to IT
        until intended interpretation(L) ≠ truth-value of L according to KB
        if L occurs as a positive literal in ICθ
        then call specialize(L)
        else call generalize(L)
        endif
    endwhile
endproc
```

Algorithm 1. The belief updating algorithm.

ize will make an unimplied true fact implied by the updated knowledge base. Below, we will describe these procedures in more detail. After each modification of the knowledge base, the integrity constraint theory is verified; if an integrity constraint becomes violated, it is handled as before. A good heuristic to reduce the involvement of the oracle is to first process the integrity constraints that correspond to answers of questions. To find the cause for a possible violation of such constraints, one does not have to rely on the oracle, as there is only one literal in such integrity constraints. Clearly, if such a constraint is violated, its single literal is responsible.

4.1. Adapting Shapiro's algorithms

Let us now briefly outline how Shapiro's generalization and specialization algorithms [29] can be applied for our purposes. We will only sketch the necessary modifications to his procedures because they are well known and well described in the literature. For a detailed description of his procedures, we refer to [29]. In the remainder of this paper, we assume that Shapiro's *eager* method is used. This method was selected because it has the nicest theoretical properties.

The specialization procedure locates an incorrect clause by analyzing a proof tree for an implied false fact. The incorrect clause is then retracted from the knowledge base. Any of Shapiro's specialization procedures (false_solution, see [29]) can be used here.

Shapiro's generalization procedure infers missing clauses and/or facts when an unimplied true fact is handled by the system. These clauses are then asserted in the knowledge base. First, the system computes in an abductive manner (the procedure missing_solution, see [29]) the predicate(s) that is responsible for the true fact being unimplied. Secondly, it searches the refinement graph (the procedure search_for_cover, see [29]) of the corresponding predicate(s) to find clauses/facts that allow to recover from the error and that are not known to be incorrect. To apply Shapiro's generalization procedure in our context, we only need to redefine the refinement operator. The refinement operator determines the refinement graph and hence the search-space. Since we only use the subset of Prolog defined in Section 2, the following refinement operator ρ is sufficient for our purposes.

Definition 4.1 (*The refinement graph* ρ[3]). For an uncovered positive example $p(a_1, \ldots, a_n)$, the *set of sons of* c, $\rho(p(a_1, \ldots, a_n), c)$, in the refinement graph is defined as:

- If p is known intensional predicate then
  - $\rho(p(a_1, \ldots, a_n), \top) = \{p(X_1, \ldots, X_n) \leftarrow\}$ where the X_i are n different variables; $\top$ denotes the most general clause for p.
  - $\rho(p(a_1, \ldots, a_n), c) = \{c' \mid c'$ is obtained by unifying two different variables in c; or c' is a clause obtained from c by adding a literal $q(Y_1, \ldots, Y_k)$ to the body of the clause c, such that q is a predicate of arity k, the Y_j are k different variables and exactly one of these variables occurs in $c\}$.
- If p is a known extensional predicate then $\rho(p(a_1, \ldots, a_n), \top) = \{p(a_1, \ldots, a_n) \leftarrow\}$.

Notice that the second part of this definition implies that the system will assert an implied true fact that corresponds to an extensional predicate. Notice also that all clauses in the refinement graph are linked. However, they are not all range-restricted. As it is not allowed to assert clauses, that are not range-restricted in the knowledge base, all non range-restricted clauses are to be considered as incorrect (in Shapiro's terminology: marked).

It is easily verified that the following property holds:

Lemma 4.2. *The set of clauses that are range-restricted and that are a refinement*

[3] Observe that our refinement graph takes also the positive example as an argument; this is useful in the extensional case.

of $\top$ *w.r.t.* $\rho(p(a_1, \ldots, a_n), \top)$ *for an intensional predicate* p*, is precisely the set of clauses allowed in the definition of the predicate* p.

As the graph determines the search space, it also determines the efficiency of effecting modifications to the knowledge base. Several knowledge sources may be used to decrease the size of the graph as much as possible:

- knowledge about the types of the involved predicates (see e.g. [16]);
- knowledge about the predicates that may be used in the definition of a predicate to be learned;
- knowledge about the structure of clauses that may be used in the definition of a predicate to be learned, e.g. about the number of existential quantifiers that are allowed in the body of a clause, see e.g. [9], or whether recursion is allowed or not.

In the literature on learning, these knowledge sources are known as *bias*, see e.g. [9, 20, 26, 32]. To cope with such biases, one only needs to consider clauses violating the bias as incorrect (marked).

4.2. An example session

As an illustration of this algorithm, we present a session with our prototype implementation. The initial knowledge base is shown below:

```
female(alice) ←          male(luc) ←
female(rose) ←           male(etienne) ←
female(ann) ←            male(gerard) ←
female(laura) ←          male(leon) ←

parent(leon,rose) ←      parent(etienne,ann) ←
parent(alice,rose) ←     parent(etienne,luc) ←
parent(rose,luc) ←       parent(laura,etienne) ←
parent(rose,ann) ←       parent(gerard,etienne) ←

Mother(X,Y) ← female(X), parent(X,Y)
```

Initially the constraint theory was empty and the bias for the predicate `father` was declared as follows: only the predicates `male` and `parent` may be used in clauses for `father` and no existentially quantified variables may occur in clauses for `father`.

```
(1)  parent(X,Y) → father(X,Y), mother(X,Y)
(2)  father(X,Y) ←
(3)  father(X,X) ←
(4)  father(X,Y) ← male(X)
(5)  father(X,Y) ← male(Y)
(6)  father(X,Y) ← parent(X,Y)
(7)  father(X,Y) ← parent(Y,X)
(8)  father(X,Y) ← male(X), parent(X,Y)
(9)  mother(X,Y), father(X,Y) →
(10) → father(etienne,ann)
(11) → parent(gunther,leon)
```

The user started by supplying the constraint (1) to the system. This constraint is violated for $\theta = \{$X=etienne, Y=ann$\}$; therefore the system asks the user for the intended interpretation of parent(etienne,ann) (true) and father(etienne,ann) (true). The answers imply that there is a missing solution for father(etienne,ann) as this example is true but not implied. Therefore the system has to compute a clause covering father(etienne,ann). To this purpose, it considers the clauses (2–8) in the refinement graph: clause (2) covers the example parent(etienne,ann) but is not range-restricted, and therefore rejected. Clause (3) does not cover the example. Therefore the system verifies whether (4) and (5) cover the example. This is realized by asking the questions male(etienne) (true) for clause (4) and male(ann) (false) for clause (5) to the user. This allows to conclude that (4) covers the example and (5) does not. Because of this and the fact that (4) is not range-restricted, the system proceeds with (6). (6) is known to cover the example because parent(etienne,ann) is already known to be true. It is also range-restricted, and therefore clause (6) is asserted in the knowledge base. This results in a consistent knowledge base. At that point the user is queried for the next constraint. Suppose he then supplies (9) to the system. This constraint is violated for $\theta = \{$X = rose, Y = luc$\}$. The reason for the violation is examined by asking mother(rose,luc) (true) and father-(rose,luc) (false). The answer to the last question means that there is an incorrect clause used to prove father(rose,luc). The clause (6) is identified as the guilty one and retracted from the knowledge base after asking parent(rose,luc) (true). As the clause is retracted, some constraints such as (10) become violated. The system discovers that a new clause covering father(etienne,ann) should be asserted. At that point it considers the next clause in the refinement graph (7) and discovers that this clause does not cover the example father(etienne,ann) by asking for the truth-value of parent(ann,etienne) (false). Therefore it goes to the next clause (8). This clause is known to cover the example and is therefore asserted in the knowledge base. This clause yields a consistent knowledge base, so no furhter questions are generated.

Suppose the user then wants to insert the fact parent(gunther,leon) by supplying the constraint (11). The constraint is violated: there is a missing solution for parent(gunther,leon). This is solved by asserting the fact parent(gunther,leon) ← in the knowledge base (parent is an extensional predicate). However, this gives rise to a violation of constraint (1) for $\tau = \{$X = gunther, Y = leon$\}$. Diagnosing the violation by asking father-(gunther,leon) (true) results in the search for a missing solution for father(gunther,leon). At this point the system's abductive procedure is invoked, and the question male(gunther) (true) is formulated. The answer means that a clause covering male(gunther) has to be asserted in the knowledge base. The found clause is male(gunther) ← as male is an extensional predicate. Hereafter there are no more violations, so the system asks for the next constraint. In a similar way the system can then learn definitions of related predicates such as ancestor, grandparent, aunt, niece, ...

5. Theoretical properties of the algorithm

In this section, we prove some properties of the proposed algorithms. More specifically, we show that (1) the procedure learn identifies knowledge bases in the limit and (2) that the procedure kbu is sound and complete. Before proving these properties, we define the involved notions in more detail.

Defintion 5.1. A procedure for belief updating is *sound* if all solutions generated by the procedure are correct.

Definition 5.2. A procedure for knowledge base updating is *complete* if it finds a solution to the belief updating problem in finite time whenever there is one.

Definition 5.3. A *presentation w.r.t. a knowledge base* is a possibly infinite sequence of integrity constraints satisfied by the knowledge base. A presentation is *complete w.r.t. a knowledge base* if all non-redundant constraints satisfied by the knowledge base occur in the sequence. A constraint IC is *redundant* iff one of the literals l in IC can be deleted, i.e. iff $IC \models (IC - \{l\})$. Otherwise it is *non-redundant*.

The requirement that all non-redundant constraints have to occur in the sequence is sufficient for our purposes. Indeed, if a non-redundant constraint $P \rightarrow Q$ occurs in the sequence it is not necessary that constraints like P, P, $P \rightarrow Q$, Q occur.

Definition 5.4. An algorithm *identifies knowledge bases in the limit* if for each knowledge base KB and all complete presentations w.r.t. KB, there is a time after which the learned knowledge base KB' is logically equivalent to KB and will not be modified afterwards.

Theorem 5.5. *Algorithm* 1 *identifies knowledge bases in the limit.*

Proof. (1) If the user would only supply examples as constraints, the system identifies concepts in the limit. This follows directly from Shapiro's theorem 4.11 [29, p. 108].

(2) Observe that the integrity constraints are only used for verifying the results of the learning. This also means that only the examples are actively used by the generalization and specialization of Shapiro in our algorithm. Therefore, when a normal *consistent* integrity constraint theory is supplied to the system, we have that only a finite number of violations can arise in the while-loop. The reason for this is that each violation corresponds to an incorrectly handled example and that Shapiro's procedures identifies concepts in the limit after processing a finite number of examples (cf. (1)). Therefore the alogithm always terminates and generates a solution to the update problem. □

Theorem 5.6 (Soundness and Completeness). *Procedure* kbu *is sound and complete.*

Proof. Completeness follows directly from the proof of Theorem 5.5. Soundness follows from the halting condition of the while-loop in procedure kbu. □

6. Discussion

In our problem-specification, given in Section 3, we concentrated on one update problem, i.e. the modification of the knowledge base w.r.t. one constraint. This situation is quite common for intensional knowledge base updating. However, in concept-learning it is usual to also consider a sequence of possible update problems. As a result, the constraints which are input at a certain time, are assumed to be valid at all times afterwards. This means that the intended interpretation is fixed and remains the same throughout the whole learning process. For intensional knowledge base updating, this assumption does not hold. Indeed, the reason for the update request is precisely that the intended interpretation has changed, e.g. when a person moves from one place to another, the address will change. In other words, a former update request may be modified later. These considerations imply that for learning, all answers to questions and all input constraints are assumed to be always valid (the intended interpretation is fixed); hence, they may be added to the integrity constraints theory. On the other hand, for intensional knowledge base updating, a distinction should be made between constraints that are always valid, and those that may be overruled by newly incoming information (the ones whose intended interpretation may change). Therefore, the temporary constraints and answers to questions should be removed from the integrity constraint theory after each knowledge base update; the others can remain in the theory.

One very important property of our knowledge base update algorithm is that it will find a solution whenever there is one (see Theorem 5.6: soundness and completeness). Although researchers in the area of knowledge base updating have introduced notions of completeness [9] and correctness [11, 12], there is to the best of our knowledge only one approach that is complete in our sense: the one of Bry [5], but he only allows for insertions and retractions of facts, not of clauses. Completeness is a very important property of our algorithms.

It seems possible to extend our technique to cope with negation as failure and functors. The problem with functors is that termination is no longer guaranteed. There are two possible approaches: (1) allow for non-termination, as is done by e.g. Guessoum and Lloyd [11] and (2) consider some depth-bound on the computation and define a notion of h-easiness as Shapiro does. To cope with functors, the definition of the refinement graph must also be extended.

Although—conceptually—it is straightforward to extend the algorithms (cf. Shapiro [29, Section 3.6.1, pp. 74–75]) to handle negation,[4] there is a problem with deciding whether a certain fact is implied by a knowledge base or not. This leads to a technical discussion on the semantics of knowledge bases, which we do not wish to address here. Another implication of the use of negation would of course be a significant increase of the search-space.

7. Related work

Our work is related to the work on knowledge base updating in the logic programming community, a.o. [5, 9, 11, 14, 23, 30]. The main novelty is that we also add clauses to the knowledge base. Although the possibility of adding clauses to the knowledge base is considered by Pereira et al. [23] and the relation with debugging is shown, they do not specify or indicate any technique to find new clauses for insertion in the knowledge base.

Regarding concept-learning, our approach is closest to [8, 22, 28, 29]. We have made extensive use of the algorithms of Shapiro, although we have adapted them to cope with our representation formalism: instead of Prolog, a query evaluation strategy that terminates for our knowledge representation has been used and a new refinement graph was defined. Most importantly, we have extended Shapiro's mechanism to handle integrity constraints. This may also have some implications for the field of debugging. We are convinced that the use of integrity constraints is also useful in other learning approaches as e.g. [3, 8, 22]. The main difference with the mentioned approaches to learning is the use of integrity constraints rather than examples and also, that these other approaches address different learning techniques as shifting the bias and constructive induction.

8. Conclusion

We have shown that the domains of intensional knowledge base updating and incremental concept-learning have much in common. It was argued that intensional knowledge base updating and incremental concept-learning can be regarded as instances of the more general problem of belief updating. The belief update problem was then solved for a restricted subset of Prolog by adapting Shapiro's Model Inference System. The main conclusions which follow from this work are that (1) integrity constraints form a very useful

[4] Problems because of special cases as e.g. updating $p \leftarrow \neg q$ and $p \leftarrow q$ w.r.t. the constraint $p \rightarrow$, can easily be solved because we rely on an oracle. The answers of the oracle allow to retract the incorrect clause.

generalizaton of examples in concept-learning; they provide the user with a powerful tool to augment the induction process with background knowledge; and (2) it is plausible to assert clauses in the knowledge base, also in the field of intensional knowledge base updating; interesting clauses can be found using inductive inference techniques.

Acknowledgement

We are indebted to Bern Martens for invaluable suggestions and comments on earlier versions of this paper. This research was sponsored by the Belgian National Fund for Scientific Research and by the Esprit BRA Compulog. The use of integrity constraints for machine learning was inspired during Luc's visit to Katharina Morik's group in the German National Center for Computer Science in 1989 and by the work in the Compulog project. We would also like to thank Francois Bry, Gunther Sablon, Hilde Ade and the referees for their comments on this work.

References

[1] D. Angluin, Queries and concept-learning, *Mach. Learn.* **2** (1987) 319–342.
[2] F. Bancilhon and R. Ramakrishnan, An amateur's introduction to recursive query processing strategies, *SIGMOND Record* **15** (1986) 16–51. (Proceedings of the ACM-SIGMOND Conference, C. Zaniolo, ed.).
[3] F. Bergadano and A. Giordana, Guiding induction with domain theories, in: Y. Kodratoff and R.S. Michalski, ed., *Machine Learning: An Artificial Intelligence Approach* **3** (Morgan Kaufmann, Los Altos, CA, 1990) 474–492.
[4] P. Brazdil and K. Konolige, eds., *Machine Learning, Meta Reasoning and Logics* (Kluwer Academic Publishers, Dordrecht, 1990).
[5] Francois Bry, Intensional updates: abduction via deduction, in: D. Warren and P. Szeredi, eds., *Proceedings 7th International Conference on Logic Programming* (MIT Press, Cambridge, MA, 1990) 561–578.
[6] S. Ceri, G. Gottlob and L. Tanca, *Logic programming and databases* (Springer, Berlin, 1990).
[7] H. Decker, Drawing updates from derivations, in: S. Abiteboul and P.C. Kanellakis, eds., *Proceedings 2nd International Conference on Database Theory, Lecture Notes in Computer Science* **470** (Springer, Berlin, 1990).
[8] L. De Raedt and M. Bruynooghe, Towards friendly concept-learners, in: *Proceedings IJCAI-89* (Morgan Kaufmann, Los Altos, CA, 1989) 849–856.
[9] L. De Raedt and M. Bruynooghe, Indirect relevance and bias in inductive concept-learning, *Knowledge Acquisition* **2** (1990) 365–390.
[10] H. Gallaire, J. Minker, and J.M. Nicolas, Logic and databases: a deductive approach, *ACM Comput. Surv.* **16** (1984) 153–185.
[11] A. Guessoum and J.W. Lloyd, Updating knowledge bases, *New Generation Comput.* **8** (1990) 71–88.
[12] A. Guessoum and J.W. Lloyd, Updating knowledge bases ii, *New Generation Comput.*, to appear.
[13] N. Helft, Induction as nonmonotonic inference, in: *Proceedings 1st International Conference*

on Principles of Knowledge Representation and Reasoning (Morgan Kaufmann, Los Altos, CA, 1989) 149–156.

[14] T. Kakas and P. Mancarella, Database updates through abduction, in: D. McLeod, R. Sacks-Davis and H. Scheck, eds., *Proceedings 16th International Conference on Very Large Databases* (Morgan Kaufmann, Los Altos, CA, 1990) 650–661.

[15] Y. Kodratoff and R.S. Michalski, eds., *Machine Learning: An Artificial Intelligence Approach* **3** (Morgan Kaufmann, Los Altos, CA, 1990).

[16] J.W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 2nd ed., 1987).

[17] J.W. Lloyd, E.A. Sonenberg, and R.W. Topor, Integrity constraint checking in stratified databases, *J. Logic Progam.* **4** (1987) 331–343.

[18] R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, *Machine Learning: An Artificial Intelligence Approach* **1** (Morgan Kaufmann, Los Altos, CA, 1983).

[19] R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, *Machine Learning: An Artificial Intelligence Approach* **2** (Morgan Kaufmann, Los Altos, CA, 1986).

[20] T.M. Mitchell, The need for biases in learning generalizations, Tech. Report CBM-TR-117, Department of Computer Science, Rutgers University, 1980.

[21] S. Muggleton, ed., *Proceedings 1st Workshop on Inductive Logic Programming*, Viana De Castelo, Portugal (1991).

[22] S. Muggleton and W. Buntine, Machine invention of first order predicates by inverting resolution, in: *Proceedings 5th International Conference on Machine Learning* (Morgan Kaufmann, Los Altos, CA, 1988) 339–351.

[23] L.M. Pereira, M. Calejo, and J.N. Aparicio, Refining knowledge base updates, Tech. Report, AI Center/UNINOVA, Portugal, 1989.

[24] J.R. Quinlan, Learning logical definition from relations, *Mach. Learn.* **5** (1990) 239–266.

[25] C. Rouveirol and J.-F. Puget, A simple solution for inverting resolution, in: K. Morik, ed., *Proceedings 4th European Working Session on Learning* (Pitman, London, 1989) 201–210.

[26] S. Russell and B. Grosof, A sketch of autonomous learning using declarative bias, in: P.B. Brazdil and K. Konolige, eds., *Machine Learning, Meta-Reasoning and Logics* (Kluwer Academic Publishers, Dordrecht, 1990) 19–54.

[27] F. Sadri and R. Kowalski, A theorem proving approach to database integrity, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, CA, 1987) 313–362.

[28] C. Sammut and R. Banerji, Learning concepts by asking questions, in: R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* **2** (Morgan Kaufmann, Los Altos, CA, 1986) 167–192.

[29] E.Y. Shapiro, *Algorithmic Program Debugging* (MIT Press, Cambridge, MA, 1983).

[30] A. Tomasic, View update translation via deduction and annotation, in: *Proceedings 2nd International Conference on Database Theory, Lecture Notes in Computer Science* **326** (Springer, Berlin, 1988) 338–351.

[31] J.D. Ullmann, *Principles of Data Bases* (Computer Science Press, Rockville, MD, 1989).

[32] P.E. Utgoff and T.M. Mitchell, Acquisition of appropriate bias for concept learning, in: *Proceedings 2nd National Conference On Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1982) 414–418.