Explanation-Based Interpretation of Open-Textured Concepts in Logical Models of Legislation

Stefania Costantini
Gaetano Aurelio Lanzarone
Dipartimento di Scienze dell'Informazione
Universita' degli Studi di Milano
via Comelico 39/41, I-20135 Milano (Italy)
costanti@imiucca.csi.unimi.it
lanzarone@hermes.mc.dsi.unimi.it

Abstract

In this paper we discuss a view of the machine– learning technique called Explanation– Based Learning (EBL) or Explanation– Based Generalization (EBG) as a process for the interpretation of vague concepts in logic-based models of law.

The open-textured nature of legal terms is a well-known open problem in the building of knowledge-based legal systems. EBG is a technique which creates generalizations of given examples on the basis of background domain knowledge. We relate these two topics by considering EBG's domain knowledge as corresponding to statute law rules, and EBG's training example as corresponding to a precedent case.

By making the interpretation of vague predicates as guided by precedent cases, we use EBG as an effective process capable of creating a link between predicates appearing as open-textured concepts in law rules, and predicates appearing as ordinary language wording for stating the facts of a case.

Standard EBG algorithms do not change the deductive closure of the domain theory. In the legal context, this is only adequate when concepts vaguely defined in some law rules can be reformulated in terms of other concepts more precisely defined in other rules. We call 'theory reformulation' the process adopted in this situation of 'complete knowledge'.

In many cases, however, statutory law leaves some concepts completely undefined. We then propose extensions to EBG standard that deal with this situation of 'incomplete knowledge', and call 'theory revision' the extended process. In order to fill in 'knowledge gaps' we consider precedent cases supplemented by additional heuristic information. The extensions proposed treat heuristics represented by abstraction hierarchies with constraints and exceptions.

In the paper we also precisely characterize the distinction between theory reformulation and theory revision by stating formal definitions and results, in the

context of the logic programming theory.

We offer this proposal as a possible contribution to cross fertilization between machine learning and legal reasoning methods.

1 Introduction

This paper proposes a novel view and treatment of legal vague concepts in logic-based models of law, on the basis of a suitable interpretation, combination and elaboration of Artificial Intelligence and Logic Programming techniques.

The open-textured, vague nature of some legal terms is well known, and how to deal with them in knowledge-based legal systems is an important open problem.

For example, Kowalski and Sergot [KS89] express the problem as follows. In statute law, vague concepts either are given a precise definition in terms of other concepts, or are left entirely undefined. In the latter case, either the user decides whether or not an undefined predicate holds, or the derived conclusions will depend on conditions still to be decided. To do more than this would require augmenting the logical model with additional information not present in the statute. One possible source of such additional information is given by decisions made in previous cases. This is the most common way that statute law interacts with case law.

Complementarity of rules and precedents has been recognized by several authors. Branting and Porter [BP91], for example, point out that rule-based reasoning alone incurs the problem of open texture: a 'generality gap' separating abstract terms from specific facts. Without knowledge of precedents, rule-based systems cannot determine whether the open-textured term in rule antecedents match case description, but must leave this decision to their users. On the other hand, case-based reasoning alone is often inadequate: because of specificity and detail of case description, few pairs of distinct cases have identical facts, thus matching a new case to a precedent typically requires rules to establish their equivalence. Therefore, both rule-based and case-based reasoning are useful for performing *classification*: the task of assigning a given case to a legal category and explaining the assignment.

In the subfield of Artificial Intelligence called Machine Learning, two main streams of research have been pursued:

- Empirical learning: inductive recognition of regularities from data, to extrapolate general knowledge. Relies on large quantities of data.
- Analitical learning: uses background domain knowledge to guide empirical learning from examples. Relies on good quality of domain knowledge.

Though complementary in nature, they have mainly followed parallel paths of development. It is only very recently that possible integrations have emerged as a major research direction in the field [MT94].

EBL (Explanation-Based Learning) is a technique, developed in the field of Machine Learning, by which an intelligent agent can learn through the observation of examples (see [El89] for an overview). Differently from other methods of concept learning, EBL creates justified generalizations from training instances by relying on background knowledge of the domain.

EBL explains why a training instance is an example of the goal concept by generating the proof that the example satisfies the concept definition. The proof structure is generalized by replacing each instantiated rule by the associated general rule, respecting constraints on variables that ensure the validity of the generalized proof. To emphasize that learning may involve generalizing the example(s), the term Explanation-Based Generalization (EBG) is also used. We will use the term EBL in the introduction, then we will adopt the EBG term and attitude in the rest of the paper since, especially in the context of logic programming, generalization is a natural outcome of the inference process.

In this paper we propose a view of the previously mentioned interaction between rules and cases in terms of an EBL process, where they play the roles of background knowledge and training instances respectively, so that interpretation of open-textured predicates, appearing as conditions in legal rules to be applied to judge a case at hand, is guided by precedent cases.

In considering the interpretation of the theory given by a logic model of rules of law, we introduce a distinction between two different possibilities, termed as theory reformulation and theory revision.

In the theory reformulation situation, the statutory law is considered complete (at least w.r.t. the concepts/predicates involved in the case to be judged) and concepts vaguely defined in some rules are reformulated in terms of other concepts more precisely defined in other rules of the same model of the concerned statute, based on given examples. This is the standard application of EBL, where a concept defined by 'non-operational' predicates is reformulated in terms of 'operational' predicates appearing in other rules of the background knowledge, according to the guidance provided by the example. In this context we interpret 'non-operationality' of a predicate as open-texture or vagueness and 'operationality' as precision (at least at a degree sufficient for the case at hand).

In the theory revision situation, the statutory law is considered as incomplete domain knowledge, and precedent cases (together with some abstraction facility) as fragmentary knowledge tentatively used for knowledge completion (again, inasmuch as it is sufficient for the current case). In the EBL literature, this situation spreads over a variety of open problems. One of these is how to provide abstraction principles capable of filling in knowledge gaps; for example, by considering more general classes encompassing the concepts of both rules and cases, or by noting similarities. In the last case EBL approaches analogical and case-based reasoning.

Adopting Dietterich's definition (reported in [El89]), a system is said to perform knowledge-level learning only when there is a change in the 'deductive closure' of its domain theory. Intuitively, standard EBL algorithms (thus most proposed EBL systems) do not change the deductive closure of the domain theory and therefore do not perform knowledge-level learning. By only changing the way concepts are expressed, theory reformulation is mainly oriented to make easier the process of using previous cases for later reference. On the contrary, enhanced EBL algorithms aim at building bridges over knowledge holes, and at making plausible leaps in the reasoning; in other terms, at performing knowledge-level learning.

Beyond the intuition, however, a formal characterization has to be given of the notion of knowledge-level learning. We deal with the above mentioned topics in the context of logic programming. Our contribution is threefold. First, we discuss how EBL can be related to the interpretation of predicates in order to connect the vocabulary of statutory rules with the way facts of precedent cases are expressed. Second, we give an original treatment to two open problems of the EBL theory: abstraction and exceptions. Third, we characterise the distinction between theory reformulation and theory revision by stating formal definitions and results.

The paper is organized as follows. In Section 2 we introduce EBL as it has been considered in the literature, especially in the context of logic programming, and present our view of for which aspects its ideas might contribute to the field of legal representation and reasoning. In Section 3 we face the problem of semantically characterising what is generated by the standard EBL process (theory reformulation). We show, under suitable definitions, that the program obtained is not strongly equivalent to the original one (equivalence under subsumption) but is weakly equivalent to it (equality of their least Herbrand models). In Section 4 we consider extensions of the standard EBL process (theory revision), aimed to fill in knowledge gaps. The extensions are capable to treat heuristics represented by abstraction hierarchies with constraints and exceptions. We also consider that, though the extended theory is clearly no longer equivalent to the original one, a characterisation of the consequences of the extended theory can still be given. We conclude the paper in Section 5 with a discussion of previous works and some final remarks.

2 Defining and viewing EBL in the legal context

EBL/EBG is usually phrased in the literature as follows [MKK86],[DM86].

Given in input:

1. a Target Concept (TC): a predicate representing the concept to be learned/generalized;

2. a Training Example (TE): a set of facts constituting an example of the target concept;

3. a Domain Theory (DT): a set of facts and rules representing background knowledge about the domain;

4. an Operationality Criterion (OC): a set of predicates in terms of which TC has to be defined

EBL/EBG gives in output, using DT, a definition of TC which generalizes TE and satisfies OC.

The EBG algorithm consists of two stages:

1. construct an explanation in terms of the domain theory that shows how the training example satisfies the target concept definition. Each branch of the explanation structure must terminate with an expression that satisfies the operationality criterion.

2. determine a set of sufficient conditions under which the explanation holds, stated in terms that satisfy the operationality criterion.

In Logic Programming, EBG has been introduced by the works reported in [PM87], [Hi87], [KM87], and since then this kind of formalization has also been used to describe most of the previous domain–independent characterizations of EBG.

In the following, we will take [KM87] as base reference w.r.t. EBG. In order to relate this technique to the legal context, we will borrow some legal concepts, terminology and examples from [Ga87]. Throughout the paper, we will assume that the reader is familiar with logic programming and the programming language Prolog. We will use Prolog for showing the main features through examples. This is a simplified presentation of a more elaborate treatment actually implemented in an extended logic programming framework (we will mention it in the conclusions).

To explain how the process goes, let us consider the so-called suicide example, due to [DM86].

The Domain Theory is as follows:
$kill(A, B) :- hate(A, B), possess(A, C), weapon(C).$
$hate(W, W) :- depressed(W).$
$possess(U, V) :- buy(U, V).$
$weapon(Z) :- gun(Z).$

The Training Example is a set of facts:
$depressed(john).$
$buy(john, obj1).$
$gun(obj1).$

The Operational predicates are defined by means of the following facts:

$operational(depressed).$

$operational(buy).$

$operational(gun).$

The Target Concept is the binary predicate $kill$. Applying the EBG algorithm to the goal: $kill(john, john)$, EBG generates the clause:

$$kill(X, X) : -depressed(X), buy(X, Y), gun(Y).$$

which represents a generalization of the Training Example, usable for further cases.

We relate the problem of interpreting a legal open-textured term to the EBG technique by stating a correspondence between these two topics, as discussed in the following.

EBG's domain knowledge corresponds to statute law, and EBG's training example corresponds to a precedent case, in a quite natural way. The only remark here is that, while the choice of examples is critical in other domains, in legal cases this is not a problem. Then, we view the EBG's goal concept as corresponding to the legal open-textured predicate that has to be interpreted (some more comments on this correspondence will be given later).

A bit more intriguing is the role of the operationality criterion. Intuitively, operational predicates are directly observable on the data, while non-operational predicates, defined by other predicates, need to be deduced. According to [MKK86], the notion of operationality is crucial for EBG. It requires that the final concept definition be described in terms of the predicates used to describe the training example, or in terms of a selected set of easily evaluated predicates from the domain theory. This means that learned concept definitions need to be not only correct, but also in a form usable by a particular agent for a particular task. Which predicates to consider operational is debated in the EBG literature. EBG systems usually rely on various operationality heuristics. The simplest one is to consider some predicates as operational a priori; a more sophisticated approach is to state conditional operationality in rules which can therefore be reasoned about themselves.

In the legal context, we interpret non-operational predicates as open-textured concepts in rules, and operational predicates as concrete wording in cases. We consider the level of operationality as the level of concreteness at which witnesses must ordinarily testify [Ga87]. For instance, again borrowing from [Ga87], while manifestation of willingness is not an operational concept, the fact John said he wanted to buy Bill's car is expressed at the operational level.

Thus we take the concept of operationality as a possible link between legal predicates, i.e. terms in which statutory or other stipulating rules and regulations are expressed, and ordinary language predicates providing the vocabulary for stating the facts of a case. This link is made effective by the EBG process, which restates the abstract definition in more concrete terms. EBG creates a synthetic rule that summarizes the logical argument supporting the goal concept, making explicit the relationship between the legal rule premises and the case facts.

With this interpretation, the application of standard EBG fits well with the need of replacing an open-textured term with a combination of terms, reported by several researchers (as discussed e.g. in [BP91]).

Overall, the EBG process corresponds to generalizing precedent cases using law rules, or (the other way round) interpreting open-textured antecedents of law rules in the light of previous cases.

This process has, in fact, two facets. One can view EBG as the acquisition of a general concept from a specific example (a *generation* facet), or as the interpretation of a general concept in a specific context (a *recognition* facet). In the EBG literature, the generation facet has been most emphasized; here we stress the recognition facet, but they are the two faces of the same coin.

We can recast the two aspects as follows. One aspect is to produce a new sufficient definition of the goal concept. This is an answer to the problem of interpreting the vague concept in the given case. Another aspect is to *learn* the concept, i.e. to assimilate the new definition within the previously available knowledge. Following Michalski's equation [Mi94]:

$$Learning = Inferencing + Memorizing$$

this aspect amounts to decide when and how to memorize the generalized concept. In the next section we will consider this aspect in more detail.

3 Theory reformulation

Once the output of EBG has been obtained, the issue becomes whether to consider it together with or separated from the other clauses. Objective of most EBG systems is efficiency, that is, to improve the overall performance of the system on future cases with the same structure. However, it has been observed that the simple addition of the new definition not only does not necessarily improves performance, but may even slow down a subsequent proof.

In the Logic Programming terminology, this phenomenon is known as the *lemma generation problem* [Sou], i.e. the addition to a set of clauses of an inferred new clause is logically redundant and, procedurally, may cause redundant computation and unwanted repeated answers to new queries.

To solve this problem, we will take the approach, not present in the literature, of building a new domain theory DT' including the learned definitions without redundancy, with respect to a suitable set of queries. In so doing, two aspects will be relevant. On the one hand, DT' and DT will not be equivalent under clause subsumption, since the objective of learning is precisely to get a DT' that is a shorter, ready-made way of defining the concept predicate thus encompassing the learning example. On the other hand, all and only the consequences of DT have to be consequences of DT'. With the usual declarative semantics of Horn clauses, this last aspect amounts to require the equivalence of DT and DT' as equality of their least Herbrand models.

We will construct the theory DT' by means of a set of definitions. Being DT' a set resulting from the application of EBG, we will call it Resulting Set.

Definition 3.1 *The Resulting Set (RS) is the set of clauses obtained from the application of EBG with the given Domain Theory, a Training Example and a fixed Operationality Criterion.*

Definition 3.2 *A clause $A :- B_1, \ldots, B_n$ is said to be below the operationality level if one of the following conditions holds:*

- *operational(A)$\in$ OC.*
- *there exists a sequence of goals $G_1, \ldots, G_h$ such that:*
    1. *$G_1 = X\theta$ where operational(X)$\in$ OC and θ is a substitution.*
    2. *G_h is of the form $:- K_1, \ldots, K_r$ and $\exists K_j, j \leq r$, which unifies with A.*

Definition 3.3 *We call Completion Theory (CT) the set of all the clauses of DT that are below the operationality level.*

Definition 3.4 *A query Q is said to be above the operationality level if there exists a sequence $G_1, \ldots, G_h$ of goals, where $Q = G_1$ and G_h is of the form $:- K_1, \ldots, K_r$ where each $K_j, 1 \leq j \leq r$ unifies with the head of a clause that is below the operationality level and there exists $i \leq r$ such that $K_i = X\theta$ and operational(X) $\in$ OC, where θ is a substitution.*

If the query Q is above the operationality level, then the application of EBG with Q as a parameter terminates.

Definition 3.5 *The Multiple Resulting Set (MRS) is the set of clauses obtained from the application of EBG with the given Domain Theory, a Training Example, a fixed Operationality Criterion and, as input, the set of queries which are above the operationality level.*

Definition 3.6 *We define Completed Resulting Set (CRS) the set resulting from the union of the Multiple Resulting Set and the Completion Theory. That is, CRS=MRS$\cup$ CT.*

For instance, considering the previous example we have the following sets:
MRS :

$kill(X, X) : -depressed(X), buy(X, C), gun(C).$
$hate(X, X) : -depressed(X).$
$possess(X, C) : -buy(X, C).$
$weapon(C) : -gun(C).$

CT is empty because all the operational predicates are facts. The (Completed) Resulting Set is thus MRS $\cup$ CT=MRS.

Given the above definitions, the following proposition is easily proved:

Proposition 3.1 *The least Herbrand models of the two sets $DT \cup TE$ and $CRS \cup TE$ are equal.*

It is not possible to get a stronger equivalence [Ma86] of the two theories. Maps T_P [Ll87] are in fact not equal for the two programs CRS $\cup$ TE and DT $\cup$ TE. Thus the subsumption equivalence does not hold, since the equality of maps T_P is a prerequisite for that kind of equivalence.

4 Theory revision

The standard application of EBG requires a complete, correct and tractable domain theory. In many real-world situations, however, only an incomplete domain theory is available. This is precisely the case when a predicate appearing in conditions of legal rules is not defined by further rules. In a sense, the knowledge represented by rules is incomplete, leaving the interpretation of some predicates completely open. In such cases, the EBG process in the previous 'standard' form (theory reformulation) is of no assistance.

Cases may again be taken as guiding the interpretation, but this time provided that they are used together with some additional knowledge. One simple yet useful heuristics is to abstract single instances provided by cases to higher classes of an abstraction hierarchy.

In the above example, *gun* is a subclass of the class *weapon*, abstracting to which also e.g. *machine_pistol* is accommodated. *sleeping_pill* does not belong to the class *weapon* but may be used for suicide; a superclass of both may be expressed by the predicate *dangerous_object*, and so on.

The classical examples of Hart and Fuller, reported in [Ga87], can be considered in this respect.

The question of whether or not the rule prohibiting use of vehicles in a park is applicable to trucks is solved by using a simple generalization hierarchy.

Abstraction hierarchies, however, are not absolute, but purpose- dependent. The English action of cattle trespass includes in the class cattle many kinds of domestic animals, but not dogs and cats, and in a case has been held to include geese, for the purpose of that action.

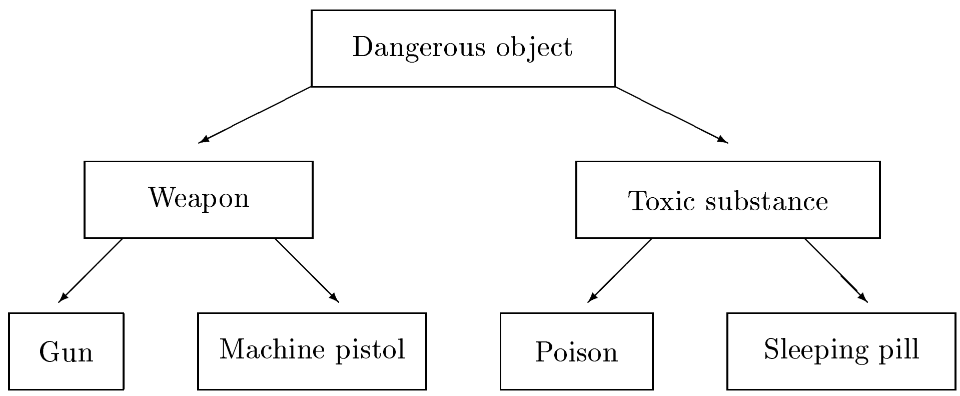
Constraints and exceptions have to be taken into account, too. Fire trucks are not prohibited in the park. Sleeping in railway stations is prohibited, except for a person who is a passenger.

In the following, two extensions of the standard EBG algorithm are shown, the first one treats abstraction hierarchies with constraints, the second one deals with exceptions and negative cases. We will call them EBG-1 and EBG-2 respectively; they are shown separately, but can easily be combined.

The extension dealing with taxonomies or hierarchy trees allows climbing the hierarchy up to the most abstract node compatible with the constraints, given by means of assertions that use the binary predicate *compatible*.

The predicate *compatible* expresses the fact that a class is compatible with its siblings and therefore with its superclass as a whole. Compatibility is not a characteristic peculiar of the hierarchy, i.e. it is not static, but depends on the particular example being dealt with.

If we consider again the previous suicide example we can see how to obtain a more abstract clause using the following taxonomy:



which is expressed by means of facts with unary predicates:

$weapon(gun).$

$weapon(machine\_pistol).$

$toxic\_substance(poison).$

$toxic\_substance(sleeping\_pill).$

$dangerous\_object(weapon).$

$dangerous\_object(toxic\_substance).$

We add to the Training Example the following fact:

$belongs\_to(gun, weapon).$

as an assertion which gives the starting point of the abstraction, and compatibility constraints between a class and its superclass indicating whether or not it is possible to continue the abstraction. The fact:

$(*) compatible(weapon, dangerous\_object).$

states that the class *weapon* is compatible with its superclass *dangerous_object*, thus the abstraction is allowed in this particular example. In such a situation, EBG-1 generates the final clause:

$kill(X, X) : -depressed(X), buy(X, Y), dangerous\_object(Y).$

which can be used to prove suicide with all the dangerous substances included in the hierarchy and not only with the gun of the previous Training Example. On the contrary, the lack of an assertion *compatible* means that the abstraction is not possible (for the purpose represented by the given Training Example). For instance, if fact (*) is not given, the generated clause is simply:

$kill(X, X) : -depressed(X), buy(X, Y), weapon(Y).$

and does not generalize up to *dangerous_object*, but only to *weapon*.

Assertions with predicates *belongs_to* and *compatible* are domain- dependent directives for the abstraction process. They must be suitably specified by the user with respect to the particular TE, in order to ensure that the abstraction process is adequate with respect to the represented situation.

When dealing with abstractions there is the problem of considering possible exceptions. Sacrifying generality to deal with all possible cases is not the proper solution. If a class is divided in two equivalence subclasses, each containing items sharing the same property, it is possible to deal with exceptions using EBG-2, i.e. a suitably modified version of the supporting EBG algorithm.

If we want to express that two sets of items of a class share a certain property or, on the contrary, that they don't share it we can use assertions the form:

$equival(Property, Class, ListA, ListB).$

$exclude(Prop, Class, ListA, ListB).$

As an example, let us consider the case of the use of vehicles in a park. A class vehicle can include, among others, the items car, truck and fire truck as represented by the following facts:

$vehicle(car).$

$vehicle(truck).$

$vehicle(fire\_truck).$

The Domain Theory states that if a vehicle is in a park it will get a fine:

$get\_fine(X) : -prohibited(X, Y).$

$prohibited(X, Y) : -vehicle(X), park(Y).$

$vehicle(X) : -car(X).$

The Training Example gives an istance of park and car, and states that car is included in the class vehicles, which is divided in two subclasses with respect to the property of being or not prohibited in the park:

$park(central\_park).$

$car(car1).$

$belongs\_to(car, vehicle).$

$exclude(prohibited, vehicle, [car, truck], [fire\_truck]).$

The clause thus generated by EBG-2 is:

$get\_fine(X) : -vehicle(X), check(X), park(Y).$

where $check(X)$ tests membership to one or the other subset and behaves accordingly.

The clause deals with the exceptions to the general rule, i.e. states that a fire truck would not get a fine since it does not share with the other items of the class vehicle the property of being prohibited in the park.

With EBG-1 it is not possible to give this distinction and therefore the clause generated would be:

$get\_fine(X) : -vehicle(X), park(Y).$

stating, incorrectly, that every vehicle in a park is subjected to fines (fire trucks included).

Two remarks are in order at this point. First, the additional knowledge, represented by both taxonomies and partitions of items into lists of positive and negative instances of a class with respect to a given property, is not intended to complement the Domain Theory, i.e. is not considered to have the same strength of the set of stipulating rules the Domain Theory consists of. Rather, it is viewed as pragmatic knowledge provided by cases and thus containing relevant but fragmentary information more than classifications of wide scope and general value. Second, the examples were presented as if the taxonomies and the exceptions were all known in advance. The extended algorithms, however, are such that they can be dynamically arranged as long as new cases (Training Examples) are added to previous ones, thus building hierarchies and lists of positive and negative items incrementally.

In the following we give the definitions to construct the (Completed) Resulting Set in both cases (abstraction and exceptions). The definition is reformulated to deal with the abstraction, with which the generated clauses contain predicates representing classes of the taxonomy.

Definition 4.1 *A query Q is said to be above the operationality level if there exists a sequence $Q = G_1, \ldots, G_h$ of goals such that G_h has the form: $? - K_1, \ldots, K_r$ where each $K_i, i \leq r$, unifies with the head of a clause that is below the operationality level and there exists $j \leq r$ such that $K_j = X\theta$ where $T : -C1, \ldots, X, \ldots, C_n$ is the clause generated by EBG, and θ is a substitution.*

Definition 4.2 *We call Class Set (CS) the set of classes used for the abstration. An item $X \in CS$ has the form: Class(Item).*

Definition 4.3 *We call Positive Set (PS) of a class C the set PS=$\{$ Y: pos(X) with Y item of the list X$\}$.*

Similarly, we define the Negative Set (NS). We call Gender Set the union of NS and PS. I.e., $GS = PS \cup NS$.

The CRS is the union of MRS, CT and CS for the case of simple abstraction. When dealing with exceptions, the CRS is MRS $\cup$ CT $\cup$ CS $\cup$ GS.

The cases presented in this section deal with abstraction supported by a taxonomy, which represents knowledge external to the Domain Theory. This knowledge is integrated by the algorithm in the clauses that it generates. Thus the new set CRS is wider in terms of the number of queries that can be proved using it. Formally speaking we can state that:

Proposition 4.1 *The least Herbrand model of CRS$\cup$ TE contains the least Herbrand model of DT$\cup$ TE.*

To put it in another way, the set of consequences of CRS $\cup$TE is equal to the set of consequences of DT $\cup$ TE, union the set of the facts expressing the taxonomy and the classification of positive and negative items of a class.

5 Previous works and concluding remarks

The notion of open texture, rooted in philosophical investigations of language, was introduced in jurisprudence by Hart [Ha61]. Its importance for statutory interpretation (i.e. the process of trying to determine the meaning of a legal rule by analyzing its terms and then applying it to a particular set of facts [RS89]) has been addressed by many researchers concerned with building logical models of law and knowledge-based legal systems.

After the recognition that rule-based reasoning (RBR) alone (in models like [BS85],[SS86],[WP80],[WP86]) or case-based reasoning (CBR) alone (as remarked by the authors themselves of systems like HYPO and TAX-HYPO [AR86],[RS89]) is insufficient, combining CBR and RBR to cope with the interpretation problem has been considered recently.

One such approach was taken in the mixed paradigm system CABARET by Rissland and Skalak [RS89] [SR92]. The system interleaves the two modes of reasoning by means of an agenda-based controller adopting a set of heuristic control rules. Similarly, the GREBE system [BP91] is based on a specific control strategy, which tries to be impartial, i.e. without arbitrary ordering restrictions on the use of rules and precedents.

Our approach is different in that we do not specify a control strategy whatsoever, but try instead to combine rules and cases in a general, declarative fashion. Our rationale is that an implemented control strategy is difficult to understand, reproduce and communicate (briefly, to evaluate), not to say to formalize. On the contrary, the way knowledge is manipulated by EBG is more explicit, since result of the EBG process is knowledge expressed in the same form as the input knowledge. We have shown that, provided that EBG is embedded in a suitable framework like logic and metalogic programming, the relationship between original and derived knowledge is more amenable to a formal characterization.

Our objective is to give a symbolic, declarative account of possibile incorporation into rule-based reasoning of other knowledge like precedent cases, secondary sources (e.g. authoritative text-books) and even common-sense knowledge. A discussion about the adequacy of a purely declarative approach to representation of legal knowledge, as opposed to a purely procedural view as strongly advocated for instance in [HL94], or even to a connectionist technique (like [OW89], where a neural network is in charge of dealing with open textured concepts) is clearly beyond the purpose of this paper.

According to [SR92], possible modes of CBR to extend RBR are:

- passive: cases may be used as addenda to rules (e.g. PLEXS);
- participating: cases may participate in problem solving and explanation (e.g. CABARET);
- active: cases may be used actively to revise the rule set explicitly. Machine Learning (inductive) techniques are mentioned.

As far as we know, no such 'active' case-based reasoners have been proposed in the literature up to now. The novel approach proposed in this paper may be a good candidate in this direction. In [SR92], the use of cases to actually revise a rule set is mentioned, and machine learning generalization techniques are suggested to be explored for this aim. Purely inductive techniques are seeked there, however (empirical learning, in the terminology stated in the introduction), while in this paper we consider that generalization from examples (analytical learning) may be more conducive in presence of background knowledge, such as the one given by legal rules together with supplementary knowledge.

The topic of theory reformulation/revision is related to the easy/hard cases distinction introduced by Hart and later revisited by many authors. Gardner's work [Ga87], for example, discussed how such legal open-textured concepts give rise to what are known as 'hard' cases, that is, cases over whose resolution experts disagree. A discussion of the concept of easy and hard cases is also found in [Su86]. In our framework we just say that, while theory reformulation is appropriate to easy cases, theory revision might be needed in hard cases.

Some authors have related the EBG process to Logic Programming techniques such as partial evaluation and program transformation [vH88], [BD89]. They have been concerned mainly with using given examples to control the partial evaluation [vH88] or the fold/unfold transformation process [BD89]. In the latter paper, the problem of replacing clauses of the original program by equivalent more efficient clauses generated by the transformation process is coped with in some detailed examples, and a possible automation of this process is discussed. General definitions, however, are not given for this theory reformulation task, and possibilities of extensions to non identical but similar subcomputations (theory revision) is only mentioned in passing.

Revisitation of EBG in presence of incomplete theories has been considered in [DK90]. They also use the method of climbing generalization trees for theory refinement (our extension of the suicide example is modeled after theirs). Our present paper goes furhter by dealing with constraints and exceptions, by giving detailed extended algoritms and by formalizing the relationship between original and refined theories.

A complete implementation of the algorithms underlying the extensions to EBG shown in this paper has been developed by Stefano Bertarello, and is reported in full detail in [Be93], and partly in [BC94]. The implementation has been worked out in the metalogic language Reflective Prolog, which is an extension of Prolog amalgamating language and metalanguage to treat metalevel knowledge and metalevel reasoning. Reflective Prolog is fully defined in [CL94] and applied to analogical reasoning in a legal context in [CL92]. The utility of using Reflective Prolog for extended EBG comes from the fact that EBG is intrinsically a metatheoretical process (acting on an object-level program), and that metastatements are involved since the beginning, like the declaration of operationality, which is a property of predicates, not of terms. This metalevel treatment allows operationality to be conditionally defined and reasoned about, instead of being given beforehand and simply used. The metalevel features of Reflective Prolog are exploited expecially for theory revision, where, in fact, the possibility of making statements about predicates is needed.

The content of the present paper is tentative in nature. While the technical aspects in the logic programming framework are fully worked out, the suggested application of the extended EBG approach to the interpretation of open-textured predicates has clearly to be empirically verified. Experimental data are needed to prove the effectiveness of the suggested method.

We do not claim that at the present stage a perfect fit has been identified between what is needed to link statutory rules to precedent cases and what the EBG technique can offer, even when extended as discussed above. We have shown that domain independent abstraction algorithms can be defined, that are easily understandable by the user, which can guide the process by means of simple directives. In the metalogic framework, where different levels of knowledge can be clearly distinguished, and where tools are available for defining in a uniform language abstraction algorithms, directives and constraints, the user could even interact with the system, and define his/her own specific abstraction methodologies (instead of simple directives). Thus, the paper may give a new methodology to utilize old cases. Though we believe that this idea can be very fruitful, furhter work is needed to exploit its potential.

References

[AR86] Ashley K.D., Rissland E.L. *Toward Modelling Legal Arguments.* in: Martino A., Socci Natali F., Automated Analysis of Legal Texts, North Holland 1986, 19-30.

[BS85] Bench-Capon T.J.M., Sergot M.J. *Towards a Rule-based Representation of Open Texture in Law.* Imperial College, Dept. of Computing, London 1985.

[Be93] Bertarello S. *Estensione delle tecniche di Explanation-Based Learning mediante l'uso dell'abduzione.* Degree Thesis in Computer Science (G.A. Lanzarone, S. Costantini supervisors), University of Milano, 1993.

[BC94] Bertarello S., Costantini S. and Lanzarone G.A.. *Explanation-Based Learning of Open-Textured Predicates in Logic Programming Models of Law.* in: Biagioli C., Sartor G. and Tiscornia D. (eds.), Proc. of the Workshop on Legal Applications of Logic Programming, held in conjunction with ICLP94, Santa Margherita Ligure, Italy, June 13-18, 1994, 14-32.

[BP91] Branting L.K., Porter B.W. *Rules and Precedents as Complementary Warrants.* in: Procs. Ninth Nat. Conf. on Artificial Intelligence, MIT Press 1991, vol. I, 3-9.

[BD89] Bruynooghe M., De Raedt L., and De Schreye D. *Explanation Based Program Transformation.* in: Procs. IJCAI '89, Morgan Kaufmann 1989, vol I, 407-412.

[CL92] Costantini S., Lanzarone G.A. *Analogical Reasoning in Reflective Prolog.* in: A.A. Martino (ed.), Expert Systems in Law, Elsevier 1992, 133-145.

[CL94] Costantini S., Lanzarone G.A. *A Metalogic Programming Approach: Language, Semantics and Applications.* Journal of Experimental and Theoretical Artificial Intelligence, 6, 1994, 239-287.

[DM86] DeJong G., Mooney R. *Explanation-Based Learning: An Alternative View.* Machine Learning, 1, 1986, 145-176.

[DK90] Duval B., Kodratoff Y. *A Tool for the Management of Incomplete Theories: Reasoning about Explanation.* in: Brazdil P.B., Konolige K. (eds.), Machine Learning, Meta-reasoning and Logics, Kluwer Academic Publishers, 1990, 135-158.

[El89] Ellman T. *Explanation-Based Learning: a Survey of Programs and Perspectives.* ACM Computing Surveys, vol. 21, n.2, June 1989, 163-221.

[Ga87] Gardner A. *An Artificial Intelligence Approach to Legal Reasoning.* MIT Press, 1987.

[HL94] Hage J.C., Leenes R., and Lodder A.R. *Hard Cases: A Procedural Approach.* Artificial Intelligence and Law, 2, 1994, 113-167.

[Ha61] Hart H.L.A. *The Concept of Law.* Clarendon Press, Oxford, 1961.

[Hi87] Hirsh H. *Explanation-Based Generalization in a Logic Programming Environment.* in: Procs. Tenth Int. Joint Conf. on Artificial Intelligence, Milano, August 23-28, 1987, 221-227.

[KM87] Kedar-Cabelli S.T., McCarty L.T. *Explanation-Based Generalization as Resolution Theorem Proving.* in: Procs. 4th Int. Workshop on Machine Learning, Ivine (CA), June 22-25, Morgan Kaufmann 1987, 383-389.

[KS89] Kowalski R., Sergot M. *The Use of Logical Models in Legal Problem Solving.* Expert Systems in Law, Int. Conf. on Law and Artificial Intelligence, Bologna, May 1989.

[Ll87] Lloyd J.W. *Foundations of Logic Programming.* 2nd edition, Springer-Verlag, 1987.

[Ma86] Maher M.J. *Equivalence of Logic Programs.* Third Int. Conf. on Logic Programming, 1986.

[Mi94] Michalski R. *Inferential Theory of Learning: Developing Foundations for Multistrategy Learning.* in [MT94], 3-61.

[MT94] Michalski R., Tecuci G. *Machine Learning - A Multistrategy Approach - Vol. IV.* Morgan Kaufmann, 1994.

[MKK86] Mitchell T.M., Keller R.M., and Kedar-Cabelli S.T. *Explanation-Based Generalization: A Unifying View.* Machine Learning 1:1, 1986, 47-80.

[OW89] Oskamp A., Walker R.F., Schricks J.A., and vanden Berg P.H. *Prolex, Devide and Rule: A Legal Application.* in: Procs. 2nd Int. Conf. on Artificial Intelligence and Law, ACM Press, 1989, 54-62.

[PM87] Prieditis A.E., Mostow J. *PROLEARN: Towards a Prolog Interpreter that Learns.* AAAI 87, 494-498.

[RS89] Rissland E.L., Skalak D.B. *Combining Case-Based and Rule-Based Reasoning: A Heuristic Approach.* Procs. IJCAI 1989, 524-530.

[SS86] Sergot M.J., Sadri F., Kowalski R.A., Kziwaczek F., Hammond P., and Cory H.T. *The British Nationality Act as a Logic Program.* CACM, vol. 29, n.5, May 1986, 370-386.

[SR92] Skalak D.B., Rissland E.L. *Using Case-Based Reasoning to Extend the Expertise of Expert Systems.* in: A.A. Martino (ed.), Expert Systems in Law, Elsevier 1992, 321-338.

[Sou] Southwick R.W. *The Lemma Generation Problem.* Imperial College Technical Report, London.

[Su86] Susskind R. *Expert Systems in Law.* Oxford University Press, 1986.

[vH88] van Harmelen F., Bundy A. *Explanation Based Generalization = Partial Evaluation.* Artificial Intelligence, 36, 1988, 401-412.

[WP80] Waterman D.A., Peterson M.A. *Rule-Based Models of Legal Expertise.* in: Procs. First Annual Conf. on Artificial Intelligence, Stanford, August 1980, 272-275.

[WP86] Waterman D.A., Peterson M.A. *Models of Legal Decisionmaking.* in: Klahr P., Waterman D.A. (eds.), Expert Systems, Techniques, Tools and Applications, Addison-Wesley 1986, 135-185