An Integrated Framework for Learning and Reasoning

Christophe Giraud-Carrier and Tony Martinez
Brigham Young University, Department of Computer Science, Provo, UT 84602

Abstract

Learning and reasoning are both aspects of what is considered to be intelligence. Their studies within AI have historically been separated, learning being the topic of machine learning and neural networks, and reasoning falling under classical (or symbolic) AI. However, learning and reasoning are in many ways interdependent. In this paper, we discuss the nature of some of these interdependencies, and propose a general framework called FLARE, that integrates inductive learning with prior knowledge and reasoning. Several examples are presented that serve as a benchmark to test the framework, including classical induction, several commonsense protocols, and the use of reasoning to discover prior knowledge that can be used as a learning bias for inductive learning.

1. Introduction

Induction and deduction both appear to be underlying processes in intelligent agents[1]. Induction "involves intellectual leaps from the particular to the general." [8, p.13]. It is often considered to be a mechanism for knowledge acquisition or learning. Indeed, "All the laws of nature were discovered by inductive reasoning." [8, p.13]. Deduction is a form of reasoning with and about acquired knowledge. It typically does not result in the generation of new facts, rather it establishes cause-effect relationships between existing facts. Deduction may be applied *forward* by seeking the consequences of certain existing hypotheses, or *backward* to discover the necessary conditions for the achievement of certain goals. Despite their differences, induction and deduction are very much interrelated. The ability to reason about a domain of knowledge is often based on rules about that domain, that must be acquired somehow; and the ability to reason can often guide the acquisition of new knowledge, or learning.

Inductive learning has been the subject of much research leading to the design of a variety of algorithms [6, 25, 28, 31]. In general, inductive learning systems generate classification rules from examples. Typically, the system is first presented a set of examples (objects, situations, etc.), also known as a *training set*. Examples are usually expressed in the attribute-value language, and represent recorded instances of attribute-value pairs together with their corresponding classification. The system's goal is then to discover sets of sufficient critical features or rules, that properly classify the examples of the training set (convergence), and adequately extend to previously unseen examples (generalization).

Though machines are still a far cry from matching human qualitative inductive leaps, inductive learning systems have proven useful over a wide range of applications in medicine (breast cancer, hepatitis detection), banking (credit screening), defense (mine-rock discrimination), botany (iris variety detection, mushroom identification), and others [27].

The study of deductive reasoning goes at least as far back as the early Greek philosophers, such as Socrates and Aristotle. Its formalization has given rise to a variety of logics, from propositional to first-order predicate logic to default logic to several non-monotonic extensions.

Many of these logics have been successfully implemented in artificial systems (e.g., PROLOG, expert systems). They typically consist of a pre-encoded knowledge or rule base, a given set of facts (identified as either causes or consequences), and some *inference engine*. The inference engine carries out the deductive process using the rules in the rule base and the facts it is provided. Several of these systems have been successfully used in various domains, such as medical diagnosis [4] and geology [9].

One of the greatest challenges of current deductive systems is knowledge *elicitation*, that is the construction of the rule base. Typically, rules are obtained from human experts. Elicitation is a tedious task that presents many difficulties both practically and theoretically. If a sufficiently rich training set can be obtained, then inductive learning may provide an alternative. It is typically easier and faster to record examples than it is to gather rules. On the other hand, if some *commonsense* knowledge can easily be obtained and instantiated to the specific domain, then deductive reasoning may also be used to improve inductive learning by, for example, generating useful learning biases or *precepts* (see [11] for a definition). Note that rules and examples need not be mutually exclusive. A system's knowledge base can indeed be constructed from both prior rules and inductively generated ones. The strong knowledge principle [36] and early work on bias [26] suggest the need for prior knowledge.

Although proposals have been made to enhance rule-based (i.e., deductive) systems with some learning capabilities [15, 30], we are not aware of any work attempting to combine induction and deduction into a unified framework. Yet, as indicated above these two processes share many interdependencies, and their integration may lead to more powerful models. This paper describes a system called FLARE (Framework for Learning And REasoning), that attempts to integrate inductive learning with prior knowledge and reasoning. In particular, inductive learning is handled incrementally, prior knowledge is used to prune and constrain the search in the input space, and reasoning is non-monotonic. Prior knowledge may be given by a teacher or deductively obtained by instantiating existing commonsense knowledge. Results on several datasets and classical commonsense protocols demonstrate promise.

The paper is organized as follows. Section 2 discusses related work in induction and deduction. Section 3 presents FLARE and further argues the validity of the unified framework. Section 4 shows simulation results on classical datasets and a benchmark of "well-designed" commonsense protocols. Section 5 introduces several preliminary examples of the use of the framework on more real-world applications. In particular, it shows reasoning patterns following learning, and automatic generation and use of prior knowledge. Finally, section 6 concludes the paper by summarizing the results, and discussing further research.

2. Related Work

FLARE follows in the tradition of [13, 14], as it attempts to combine inductive learning with prior knowledge and reasoning. There does not seem to exist, to the authors knowledge, any other such attempt. There are however a variety of inductive learning models, reasoning systems, and models combining induction with prior knowledge, that bear similarity to the corresponding components of FLARE.

Induction in FLARE is carried out much the same way as in NGE [31]. However, because generalization is effected only by setting some attribute(s) to don't-care, the produced generalizations, or generalized exemplars [31], are hyperplanes, rather than hyperrectangles, in the

input space. Hence, FLARE implements a nearest-hyperplane learning algorithm. FLARE also uses static and dynamic priorities to break ties between equidistant generalizations. Moreover, where it was shown that overlapping hyperrectangles may hinder performance [37], FLARE allows overlapping hyperplanes for purposes of dealing with conflicting defaults.

In the case that no generalizations are constructed from the training examples, FLARE degenerates into a restricted form of MBR [33]. The distance metric used is similar to IBL's metric [1], but it also deals with don't-care attributes (which are non-existent in instance-based learners), and treats missing attributes somewhat differently. Where IBL considers them to be complete mismatches, FLARE chooses a more middle-ground approach that may better capture the inherent notion of missing (or "don't-know") attributes.

Learning in FLARE contrasts with algorithms such as CN2 [6], where all training examples must be available *a priori*. Rather, it follows an incremental approach similar to that of [10], except that it is the knowledge itself that is evolved, rather than the system's structure. Moreover, learning in FLARE can be effected continually.

Systems combining inductive learning with prior knowledge include PDLA [11], ScNets [16], and ASOCS [23]. ScNets aim at providing an alternative to knowledge elicitation from experts. Known rules may be pre-encoded and new rules can be learned inductively from examples. The representation lends itself to rule generation, but the constructed networks are complex and generalization does not appear trivial. In both PDLA and ASOCS, prior knowledge must also be explicitly encoded. FLARE's approach is more flexible. Because the system can reason, some prior knowledge can be deduced automatically from commonsense knowledge. Moreover, ASOCS and PDLA are limited to nominal domains. FLARE, like ScNets, handles both nominal and linear (including continuous) domains. Prior knowledge may take a variety of forms, some of which are discussed in [3]. FLARE uses either pre-encoded defaults or deducible domain knowledge. FLARE's prior knowledge is often used as a learning bias to prune and constrain the search in the input space. A formal discussion on the need for such biases is in [26].

FLARE's reasoning is a combination of default (i.e., rule-based) and similarity-based reasonings, as in CONSYDERR [34]. However, CONSYDERR is strictly concerned with a connectionist approach to concept representation and (commonsense) reasoning. The resulting model is elegant. It consists of a two-level architecture that naturally captures the dichotomy between conceptual and subconceptual (i.e., features) levels. However, it does not address the problem of learning (how such a skill could be incorporated is also unclear), and is currently only instantiated at the conceptual level. FLARE's representation is not as elegant, but the model can effectively be instantiated both at the conceptual and subconceptual levels. CONSYDERR deals only with Boolean features and a concept's representation is limited to a single conjunction of features. FLARE's concepts are disjunctive conjunctions and the domain of features is not restricted. Hence, FLARE uses a more general distance metric than CONSYDERR's similarity measure based on feature overlap. However, FLARE has no mechanisms for individual weighting of features.

FLARE can be used as a simple expert system's inference engine. As an expert system, it can perform forward chaining in an attempt to reach a target goal. Unlike expert systems, whose knowledge base is encoded *a priori*, FLARE learns (and evolves) its knowledge base from both examples and rules. Hence, generalization may take place. There are currently no mechanisms in FLARE to deal explicitly with fuzzy rules. However, several mechanisms exist to handle

inconsistencies and conflicts. FLARE always makes a decision based on available evidence. A confidence level can also be produced to characterize the "goodness" of the decision.

FLARE offers solutions to many useful commonsense protocols. Many solutions have been proposed to the inheritance problem for example (see [18, 35, 38]). FLARE's solution is similar to CONSYDERR's [35]. However, FLARE is able to handle its protocols either extensionally or intensionally. Most reasoning systems, because they do not have any inductive mechanisms, are confined to the intensional approach. The extensional approach endows the system with more self-adaptivity.

3. FLARE - A Framework for Learning and Reasoning

In this section, FLARE's learning and reasoning mechanisms are detailed, preceded by a description of how first-order clauses are translated into their equivalent attribute-value language representation and a discussion of the semantical difference between learned "classification" rules and "conceptual" implications.

3.1. First-Order to Attribute-Value Translation

Reasoning tasks are often expressed in English and have simple, direct counterparts in the classical first-order logic language (FOL). Because FLARE's representation language is the attribute-value language (AVL), FOL clauses must first be translated into their AVL equivalent. Note that AVL is not as expressive as FOL, so that FLARE has some inherent limitations. In particular, there is no support for variable binding and unification. Essentially, only FOL clauses of the form $(\forall x)p(x) \Rightarrow q(x)$, i.e., with a single (universally quantified) variable, may be translated. In some cases, this restriction may be partially lifted by contextual information and/or the addition of predicates (see Examples 2 and 4). Despite said limitations, FLARE effectively handles a wide range of applications. Moreover, AVL accounts for simple, efficient matching mechanisms, and lends itself naturally to many inductive learning problems. Indeed, many successful learning systems use AVL [6, 25, 28].

FOL statements of the aforementioned form are translated in a straight forward way into an equivalent symbolic-valued AVL representation, as follows.

1. Each predicate (e.g., on_table(x)) becomes a Boolean attribute.
2. Each "class" of objects (e.g., blocks) becomes a multi-valued (i.e., nominal) attribute, such that each of the attribute's value represents an instance of the class (e.g., A, B, and C if A, B, and C are blocks).
3. Each implication becomes a *vector* (defined over the cross-product of the domains of the attributes defined in steps 1 and 2) in which the attributes corresponding to the premise and the conclusion are asserted and all other attributes are don't-care (i.e., $*$). The attribute representing the conclusion or target is tagged as such (e.g., subscript T).

Note that steps 1 and 2 must be done first (i.e., the set of clauses must be "scanned" once) so as to identify all attributes. Dynamic binding of new attributes (for example in an incremental setting) is a topic of further research.

It is clear that as the number of predicates increases, so does the size of the vectors. Since all vectors are of the same size, and many of them may only have values set for a relatively small number of their attributes, this may result in large memory requirements, as well as in an increase of execution time of operations on vectors. A simple extension to the above transformation can be made to limit the growth of vectors. Predicates that qualify different possible values of the same concept (e.g., red(x), yellow(x), for colors) do not have to give rise to distinct Boolean attributes. They are instead combined into a single multi-valued attribute (e.g., color(x, V), where V is a constant: red, yellow, etc., see also Example 4). This is particularly useful for the conclusion part $q(x)$ as it may correspond to a classification for x.

Following are four simple examples that demonstrate the transformation.

Example 1: Nixon Diamond [29]

1. Republicans are normally not Pacifist — $\text{Republican}(x) \Rightarrow \neg\text{Pacifist}(x)$
2. Quakers are normally Pacifist — $\text{Quaker}(x) \Rightarrow \text{Pacifist}(x)$
3. Nixon is a Republican and a Quaker — $\text{Republican}(\text{Nixon}), \text{Quaker}(\text{Nixon})$

Scan: 3 Boolean attributes for predicates Republican, Quaker and Pacifist

AVL Translation:

1. $1 \quad * \quad 0_T$
2. $* \quad 1 \quad 1_T$
3. $1 \quad 1 \quad ?_T$ where ? stands for don't-know

Example 2: Root Node Definition [19]

1. A root-node *is* some other node's parent but is not itself a child-node
   $\text{Parent}(x,y), \neg\text{Child-node}(x) \Rightarrow \text{Root-node}(x)$
2. A child-node *has* some other node as a parent
   $\text{Parent}(y,x) \Rightarrow \text{Child-node}(x)$

Scan: 4 Boolean attributes for predicates Child-node, Root-node, *Has*-parent and *Is*-parent

AVL Translation:

1. $0 \quad 1_T \quad * \quad 1$
2. $1_T \quad * \quad 1 \quad *$

Note that it is information about the semantics of the predicate Parent that makes things work in AVL. In particular, Parent gives rise to two different concepts depending on the order of its variables.

Example 3: Flying or Not Flying [21]

1. Animals typically do not fly — $\text{Animal}(x) \Rightarrow \neg\text{Fly}(x)$
2. Birds are animals — $\text{Bird}(x) \Rightarrow \text{Animal}(x)$
3. Birds typically fly — $\text{Bird}(x) \Rightarrow \text{Fly}(x)$
4. Penguins are birds — $\text{Penguin}(x) \Rightarrow \text{Bird}(x)$
5. Penguins do not fly — $\text{Penguin}(x) \Rightarrow \neg\text{Fly}(x)$

Scan: 4 Boolean attributes for predicates Animal, Bird, Penguin and Fly

AVL Translation:

1. $1 \quad * \quad * \quad 0_T$
2. $1_T \quad 1 \quad * \quad *$
3. $* \quad 1 \quad * \quad 1_T$
4. $* \quad 1_T \quad 1 \quad *$
5. $* \quad * \quad 1 \quad 0_T$

Example 4: Fitting Lenses

1. Low tear production rate causes dry eyes — Tpr(x, low)⇒Eyes(x, dry)
2. Dry eyes should not have things put in them — Eyes(x, dry),Thing(y)⇒¬Fit(x,y)
3. Lenses are things — Lenses(y)⇒Thing(y)

Scan: 2 Multi-valued attributes for predicates Tpr and Eyes
3 Boolean attributes for Thing, Lenses and Fit

AVL Translation:

1.	low	dry_T	*	*	*
2.	*	dry	1	*	0_T
3.	*	*	1_T	1	*

Note that it is contextual information about the relation between x and y that makes things work in AVL

To reason in the AVL context, special matching mechanisms are developed, and an execution scheme is devised to properly handle conflicting defaults and inheritance. FLARE's execution implements a non-monotonic default logic.

3.2. A Question of Semantics

As discussed above, FLARE only handles statements of the form $(\forall x)p(x)\Rightarrow q(x)$, where p is a conjunction of predicates on x. The semantics attached to such statements may be of at least two forms: *definition* and *implication*. If the predicates in p correspond to features of x, and q corresponds to some classification, or definition for x, then the statement is a definition. Rules obtained by induction from examples (i.e., learned classifications) represent such definitions. In the context of a definition, attributes (or predicates) of p may be viewed as subconceptual while q is conceptual. On the other hand, rules that relate concepts (at the same relative cognitive level) are implications. Such rules typically capture some domain knowledge, or may be generated by reasoning about the current knowledge base.

Definitions may be viewed as "if and only if" statements, while implications cannot. Induction, which produces definitions, lends itself naturally to the completion principle proposed in [5]. That is, given a definition, if the knowledge base is instantiated at the conceptual level (i.e., q is known), then it is appropriate to assert p. However, the same does not hold for implications. For example, given that birds are animal, and that some x is an animal, it does not follow that x is a bird. The usefulness of the completion principle for definitions is in similarity-based reasoning, and the potential generation of new conceptual rules. Suppose, we have learned that $(\forall x)p(x)\Rightarrow q(x)$ and $(\forall x)p'(x)\Rightarrow q'(x)$, where p and p' overlap (i.e., have some predicates in common). If we now assert that $q(x)$, then, using the completion principle and subsequently asserting p(x), we may be able to derive $q'(x)$ based on the amount of overlap between p and p'. A new "conceptual" implication may thus be generated.

Consider an example from [7]. Assume it is known that west Texas is cattle country, and the question is whether the Chaco is also cattle country. If it is known that the Chaco is similar to west Texas (in some relevant way), then it may be plausibly concluded that the Chaco is cattle country (even though this conclusion is less certain than the one about west Texas). Using the above notation, we have $p(x)\Rightarrow$theChaco(x), $p'(x)\Rightarrow$westTexas(x), and westTexas(x)⇒ CattleCountry(x). Asserting theChaco (at the conceptual level) causes its definition p (at the subconceptual level) to be asserted, and then the overlap (which captures the similarity) between p and p' allows westTexas to be derived first, and finally CattleCountry.

In FLARE, the representation is extended and a *definition indicator* is tagged to those statements that may be completed (i.e., definitions, inductively learned classifications). Note that the need for such an (somewhat cumbersome) extension stems from the fact that FLARE does not physically separate conceptual and subconceptual levels. CONSYDERR on the other hand provides natural support for the dichotomy. FLARE's representation makes learning more readily applicable and preserves consistency with previously developed models. At this point, the issue of achieving both the dichotomy and easy learning remains open.

3.3. FLARE's Reasoning

FLARE implements a form of default reasoning combined with similarity-based reasoning, similar to CONSYDERR [34]. Such a combination decreases the system's susceptibility to brittleness. It has also been shown to effectively account for several useful commonsense protocols, such as inheritance (see [34], Appendix A).

Because FLARE uses AVL, its knowledge base is a set of vectors, and similarity is computed as a distance function in the corresponding vector space. To initiate reasoning, a new vector is created in which attributes whose values are known are asserted, while all other attributes are set to ? (i.e., don't-know). The system then uses its knowledge base to derive as many conclusions as possible based on given information.

Let *v* be the vector representing the current state of knowledge of the system. One attribute, designated as T-attribute, is chosen as the actual, final target (sometimes referred to as top goal) for the current context. To derive a conclusion (i.e., set the T-attribute of *v* to an appropriate value), two mechanisms are available: rule application and similarity-based assertion. If a rule exists that can be applied, then it is applied. It is only when no rule can be applied that similarity-based reasoning takes effect. The following sections describe these mechanisms, and detail FLARE's reasoning scheme.

To illustrate the definitions and algorithms of sections 3.3 and 3.4, we construct an example based on the *mediadv* knowledge base [17]. This purposely simple example will serve as a running example throughout the paper. A discussion of the complete *mediadv* knowledge base is in Section 5.3. Here, two conditions (i.e., instructional feedback and presentation modification) have been left out, and many of the disjunctions predicting the stimulus situation and the appropriate response have been omitted. The goal is for the system to predict the media to consider, based on 4 conditions: stimulus situation, appropriate response, stimulus duration, and training budget. Only a few of the original rules are used, and some are more general to illustrate the use of examples and precepts (see Section 3.4.1). A few examples have also been added to exercise important features of the algorithms. Table 1 shows the informal English version of the knowledge (with reference to the rule of *mediadv* it was generated from when applicable) used and its corresponding translation into AVL vectors. We let *KB* be the resulting set of vectors. The attributes are given in the order: situation, stimulus-situation, response, appropriate-response, stimulus-duration, training-budget, and media-to-consider. Note that all the attributes are nominal. The symbolic values used in the English statements are transformed into equivalent nominal values in the vectors. Hence, for example, the first statement gives rise to a vector in which the attribute situation is set to 0 (the corresponding nominal value of schematics for this attribute), and the target attribute stimulus-situation is set to 0 (the corresponding nominal value of symbolic for this attribute).

Table 1 - Simple *KB* Running Example

English Statements	Equivalent AVL Vectors
If situation = schematics then stimulus-situation = symbolic (R3)	$v_1 = 0\ 0_T * * * * *$
If situation = conversation then stimulation-response = verbal (R4)	$v_2 = 1\ 1_T * * * * *$
If situation = photograph then stimulus-situation = pictorial (R2)	$v_3 = 2\ 2_T * * * * *$
If response = observing or response = thinking then appropriate-response = covert (R5)	$v_4 = * * 0\ 0_T * * *$ $v_5 = * * 1\ 0_T * * *$
If response = emoting then appropriate-response = affective (R10)	$v_6 = * * 2\ 1_T * * *$
If stimulus-situation = verbal and appropriate-response = covert and stimulus-duration = brief then media-to-consider = lecture (R13)	$v_7 = * 1 * 0\ 0 * 2_T$
If stimulus-situation = verbal or stimulus-situation = symbolic or stimulus-situation = pictorial and appropriate-response = covert and stimulus duration = brief and training-budget = medium then media-to-consider = lecture-with-slides (R14)	$v_8 = * 1 * 0\ 0\ 1\ 3_T$ $v_9 = * 0 * 0\ 0\ 1\ 3_T$ $v_{10} = * 2 * 0\ 0\ 1\ 3_T$
If stimulus-situation = verbal and stimulus-duration = brief then media-to-consider = role-play-w/verbal-feedback (R16)	$v_{11} = * 1 * * 0 * 0_T$ pty = 1
If stimulus-situation = verbal and appropriate-response = affective then media-to-consider = role-play-w/video-feedback (R17)	$v_{12} = * 1 * 1 * * 1_T$ pty = 3
If situation = conversation and response = observing or response = thinking and stimulus-duration = brief and training budget = medium then media-to-consider = lecture	$v_{13} = 1 * 0 * 0\ 1\ 2_T$ $v_{14} = 1 * 1 * 0\ 1\ 2_T$
If situation = photograph and response = emoting and stimulus-duration = persistent and training-budget = small then media-to-consider = role-play-w/verbal-feedback	$v_{15} = 2 * 2 * 1\ 0\ 0_T$
If situation = photograph and response = emoting and stimulus-duration = persistent and training-budget = small then media-to-consider = lecture	$v_{16} = 2 * 2 * 1\ 0\ 2_T$
If situation = photograph and response = emoting and stimulus-duration = persistent and training-budget = small then media-to-consider = role-play-w/verbal-feedback	$v_{17} = 2 * 2 * 1\ 0\ 0_T$

3.3.1. Rule Application. A rule may be applied if it *covers v*. A vector *x* is said to cover a vector *y* if and only if *x* and *y* have the same T-attribute, and the set of remaining attribute-value pairs of *x* is a subset of the set of remaining attribute-value pairs of *y*. For example, in *KB*, v_{11} covers v_7 and v_8, but v_{11} does not cover v_9 or v_{12}. Intuitively, *x* covers *y* if *y* satisfies all of the premises of *x*.

The notion of subset is extended slightly to accommodate real-valued attributes. Given that the probability of two real values being equal is extremely small, the cover relation would essentially never hold. The notion of equality is extended as follows. Two values x_1 and x_2 are *equal* if and only if $|x_1\text{-}x_2| \leq \delta$, for some $\delta > 0$ (see [14] for details). Hence, the vector $(*, 1.2, 3.52, *, 0_T)$ covers the vector $(2, 1.3, 3.48, *, 0_T)$ if $\delta = .5$. In the current implementation δ is some fraction of the range of possible values of each attribute.

3.3.2. Similarity-Based Assertion. The following non-symmetric distance function is defined over vectors. Vector $\boldsymbol{x}$ is stored in the knowledge base, and vector $\boldsymbol{y}$ is presented to the system to reason about. The distance is always computed from $\boldsymbol{x}$ to $\boldsymbol{y}$.

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\displaystyle\sum_{i=1}^{n} \begin{cases} x_i \neq y_i & \text{if } x_i, y_i \neq *, ? \text{ and attribute } i \text{ is nominal} \\ \dfrac{|x_i - y_i|}{range(i)} & \text{if } x_i, y_i \neq *, ? \text{ and attribute } i \text{ is linear} \\ .5 & \text{if } (x_i = ?) \vee [(x_i \neq *) \wedge ((y_i = *) \vee (y_i = ?))] \\ 0 & \text{otherwise} \end{cases}}{num_asserted(\boldsymbol{x})}$$

where *range*(*i*) is the range of values of attribute *i*, and
num_asserted($\boldsymbol{x}$) is the number of attributes that are not $*$ in $\boldsymbol{x}$

A detailed discussion of and partial justification for the definition of *D* is in [12]. *D*(*x*,*y*) is meaningful only if *x* and *y* have the same T-attribute, and the T-attribute is left out of the computation. For example, $D(v_{11}, v_8) = 0$, $D(v_{13}, v_{14}) = 1/5$, $D(v_7, v_{16}) = 2/3$, $D(v_{16}, v_7) = 5/8$, and $D(v_1, v_4)$ is undefined.

Similarity-based assertion consists of asserting the T-attribute of *v* to the value of that attribute in *v'*s closest match. *D* extends the similarity function defined in [34] to non Boolean attributes. It relies on distance between values rather on the amount of overlap between Boolean attributes. If all attributes are Boolean and there are no missing values, then *D* is actually the reciprocal of the aforementioned similarity function.

Note that (since *D* is not symmetric) *x* covers *y* if and only if *D*(*x*,*y*) = 0. Hence, since 0 is the minimum of the distance function, *D* can be used to apply both reasoning mechanisms with the correct order (i.e., rules first, similarity next), by computing the distance from all the rules in the current knowledge base to *v*, and simply selecting the rule that minimizes *D*. As it is possible that more than one rule minimizes *D*, a priority scheme is devised to choose a winner. This conflict resolution procedure is outlined in Section 3.3.3.

3.3.3. Conflict Resolution. When more than one rule minimizes *D* (i.e., can be selected for application), a winner is chosen according to the following priority scheme, where each subsequent condition is invoked if a tie exists at the previous level.

1. most specific
2. highest static priority
3. highest dynamic priority
4. greatest cover

Specificity is defined to be the number of attributes, other than the T-attribute, whose value is not $*$. A vector x is more specific than a vector y if specificity(x) > specificity(y). For example, in *KB*, v_7 has specificity 3, v_8 has specificity 4, and v_{13} has specificity 5, so that v_{13} is more specific than both v_7 and v_8. The static priority value may be set as rules are learned (e.g., v_{11} and v_{12} in *KB*). It allows the intensional handling of conflicts. The dynamic priority is updated through learning. It allows the extensional handling of conflicts. The cover of a vector x is the number of other vectors in the current knowledge base that x covers and that have the same T-attribute's value. It can intuitively be viewed as a confidence level in the validity of the rule, based on available evidence. In *KB*, all vectors have cover 0, except v_7 and v_{11} that have covers 1 and 2, respectively.

Giving priority to the most specific vector allows FLARE to handle exceptions and cancellation of inheritance. The static priority scheme makes it possible to handle conflicting defaults as defined by a teacher. The dynamic priority scheme accounts for epistemological inconsistencies that may be resolved over time as more information becomes available in support of one belief or the other (see Section 3.4.2). Finally, selecting the vector with greatest cover allows evidence gathered from experience to guide a final selection. Note that the current scheme gives precedence to teacher-provided information. Other ordering schemes can easily be defined. For example, static priorities could be given as a simple form of initial bias, and evidence gathered through learning (e.g., dynamic priority and cover) could be used to confirm or modify these priorities.

3.3.4. Completion. The above description and the use of *D* assumes that the model is instantiated at the subconceptual level. As discussed in Section 3.2., it is sometimes useful to instantiate the model at the conceptual level. Each vector is tagged with a flag that characterizes the rule it represents as either an implication or a definition. Before FLARE performs any of its other reasoning mechanisms, it finds all asserted attributes of v that are T-attributes of definitions in the current knowledge base. If any such attribute is found, and for all of them, completion is applied. That is, all asserted attributes of the corresponding definitions that are missing (i.e., ?) in v, are "copied" into v. Potential conflicts are resolved by giving precedence to the first copy made. The algorithm is as follows.

Completion

1. For each non-asserted attribute a of v other than the T-attribute
   If a is the T-attribute of some definition d and their values are equal
   Copy all asserted attributes of d that are still ? in v, into v

Because FLARE's concepts and rules are disjunctive conjunctions, where each vector is a conjunction and all the vectors having the same T-attribute form the disjunction, a choice must be made for the application of the completion principle. Note that each conjunction need not be mutually exclusive with other conjunctions. Currently, the system chooses one conjunction at

random to perform the completion on. Other mechanisms (e.g., apply to all, select a winner based on some criteria, etc.) are the topic of further research.

3.3.5. Forward Chaining. As mentioned above, v has a single T-attribute, corresponding to the final goal to achieve. However, at any given time, any one of the (yet) non asserted attributes of v may be designated as a temporary T-attribute. A temporary T-attribute shall be called a t-attribute. A t-attribute represents a possible subgoal that may be useful (or necessary) in reaching the final goal. FLARE then performs forward chaining as follows.

Forward_chaining

1. Do
   a. Let $w = v$
      { create a temporary copy of v }
   b. For each non-asserted attribute[2] a of v other than the T-attribute
      i. Designate a as a t-attribute
      ii. If a rule can be applied to $\boldsymbol{v}$ to assert a, apply it by asserting a in $\boldsymbol{w}$
      { based on v, assert all possible attributes in w }
   c. Let $v = w$
      { copy result back into v for the next level of inference }

   Until no new attribute of v has been asserted
2. If a rule can be applied to assert v's T-attribute, then apply it
   Else perform similarity-based assertion

Step 1 is the heart of the algorithm. Each execution of step b corresponds to the achievement of all possible subgoals at a given depth in the inference process. Each iteration uses knowledge acquired in the previous iteration to attempt to derive more new conclusions. Consider *KB*, and assume the vector $v = (1, ?, 1, ?, 0, 0, ?_T)$ is input to the above algorithm. Execution proceeds as follows. There are only 2 steps in the forward chain.

Step 1. a. Let $w = v$
b. i. Designate the second attribute as a t-attribute (first subgoal)
b. ii. Apply rule v_2: result is $w = (1, 1, 1, ?, 0, 0, ?_T)$
b. i. Designate the fourth attribute as a t-attribute (second subgoal)
b. ii. Apply v_5: result is $w = (1, 1, 1, 0, 0, 0, ?_T)$
c. Let $v = w$

Step 2. Compute distance to all vectors in *KB*
Two conflicting winners: $D(v_7, v) = D(v_{11}, v) = 0$
Apply conflict resolution: v_7 is more specific so it wins
Apply v_7: result is $v = (1, 1, 1, 0, 0, 0, 2_T)$

Notice that only rules are applied for the assertion of t-attributes (i.e., subgoals). As a result, the accuracy of the final goal is increased, but the ability to perform approximate reasoning is reduced. It is possible to relax this restriction thus potentially achieving more subgoals, but reducing the confidence in the final result. For example, step ii could be modified to allow not only rules, which are perfect matches (i.e., $D = 0$), but also matches deemed to be "close enough."

The measure of closeness can be implemented via a threshold value T_D, placed on D. That is, step ii is replaced with:

ii$_1$'. Find closest match (according to Section 3.3.3) - distance D'
ii$_2$'. If $D' \leq T_D$ perform similarity-based assertion (rule application if $D' = 0$)

The value of T_D then offers a simple mechanism to increase the level of approximate reasoning. This is particularly useful for cases such as the Chaco example, where, after completion, some of the reasoning is based on the amount of similarity between concepts. Notice that steps ii$_1$' and ii$_2$' are functionally equivalent to step ii when $T_D = 0$.

Step 2 concludes the reasoning phase by asserting the target attribute (or goal). The goal is always asserted, either by rule application or similarity-based assertion. Hence, FLARE always makes a decision. The accuracy of the decision depends upon available information[3].

3.3.6. FLARE's Reasoning Algorithm. Given all of the above mechanisms, FLARE reasons as follows. Let *v* be the vector containing the facts to reason about.

Reasoning
1. Perform Completion
2. If *v*'s T-attribute has been asserted, then done
   Else Perform Forward_chaining

Step 2 causes reasoning to stop if the goal can be achieved by completion based on existing definitions. This is consistent with the semantics of definitions, and allows the system to reach subconceptual goals that are not achievable by existing rules.

In order to test the predictive accuracy of FLARE, the standard training set/test set approach may be used. The value of *v*'s T-attribute is provided, but it is not used during reasoning. Rather, the system reasons based on its current knowledge base and all of the asserted attributes of *v*. When reasoning is completed, the "computed" target value is compared with the "actual" target value. More information may also be displayed to show how the goal is achieved. Currently, FLARE is non-interactive, that is, it cannot query a user for the values of missing attributes that may help improve the accuracy of its result.

3.4. FLARE's Learning

The kind of learning supported by FLARE is supervised. Thus, a target output is provided along with each training item. Learning is incremental. Items in the training set become available one at a time and are never reprocessed. The system continually adapts to the new information it receives. Moreover, FLARE extends inductive learning from examples with prior knowledge in the form of precepts. Section 3.4.1 contains some preliminary definitions. Section 3.4.2 presents FLARE's learning algorithm. Section 3.4.3 discusses how generalization is effected. Section 3.4.4 shows the details of the adaptation phase of the learning algorithm The *KB* example of Section 3.3 is used throughout, and Section 3.4.5 details how FLARE would learn *KB*. Finally, Section 3.4.6 addresses the issue of extensionality and intensionality.

3.4.1. Examples vs. Precepts vs. Rules. Induction consists of going from the specific to the general. The setting typically involves some target concept and a set of attributes whose values are believed to be relevant in predicting the values of the concept. In this context, the target concept is a T-attribute. The relevant attributes shall be called R-attributes. Note that each T-attribute thus has a unique associated set of R-attributes. For example, the vectors of *KB* can be used to predict the value of the attribute media-to-consider. We shall designate that attribute as the T-attribute, and attributes stimulus-situation, appropriate-response, stimulus-duration, and training-budget as the R-attributes in this context.

An *example* of a concept *C* is a vector that has all of *C*'s R-attributes set to either ? or one of their possible values. The special symbol ? accounts for possible missing attribute values in real-world examples. *Precepts* and *rules* are vectors in which some of the R-attributes are set to the special value *don't-care* (∗). In *KB*, v_8, v_9, and v_{10} are examples, while v_7 and v_{12} are precepts.

The don't-care value has the following semantics. If i_j is an R-attribute whose value is don't-care in a vector p, then every vector p' obtained from p by asserting i_j to any one of its possible values and leaving p's other R-attributes unchanged has the same output (or concept) value as p. Informally, this says that in the context of p, the value of i_j has no effect on the value of the output. A vector containing don't-care R-attributes *represents* several examples. Consider the following example.

> Let there be 3 R-attributes ranging over the set {0, 1, 2}
> Let the T-attribute be Boolean
> Then the vector (∗, 1, 0, 0_T) represents (or covers) all of the following:
> (0, 1, 0, 0_T) (1, 1, 0, 0_T) (2, 1, 0, 0_T)

The distinction between rules and precepts is limited to learning. In reasoning, all vectors (including examples that do not generalize) are rules. In learning, the term rule is applied to an induced generalization from examples. In FLARE, rules are formed by applying the dropping condition generalization rule [25], that is, under certain conditions (see Section 3.4.3), one attribute is set to don't-care. The term precept is applied to a generalization given either by a teacher or deduced from an instantiation of commonsense knowledge to the domain under study. Precepts reflect some high-level knowledge (or commonsense) about the real-world. A precept "suggests something advisory and not obligatory communicated typically through teaching" (Webster's Dictionary). Precepts are not necessarily correct, but convey some hint or bias. For instance, when dealing with how weather conditions affect what one wears, a possible precept is: "if it rains, take an umbrella," regardless of the temperature, the day of the week, or any other unnecessary (don't-care) detail.

Seminal work on the need for biases [26], the strong knowledge principle [36], and empirical results on the use of precepts to augment inductive learning [11] suggest the validity of using both raw examples and prior knowledge to improve the performance of learning systems.

3.4.2. FLARE's Learning Algorithm. The set of all examples, rules and precepts that share the same R-attributes and T-attribute can be viewed as a function mapping the R-space into the T-space. In this context, an example maps a single point of the R-space to a value in the T-space, while precepts and rules are hyperplanes of the R-space that map all of their points to the same value in the T-space.

Learning then follows a form of nearest-hyperplane learning. It consists of first applying the reasoning scheme, and then making adjustments to the current knowledge base to reflect the newly acquired information. Note that forward reasoning on partial information is effected prior to the induction. Hence, if there are missing attributes and the knowledge base contains rules that can be applied to assert these attributes, the rules are applied so that as many of the missing attributes as possible are asserted before the final target is predicted. Hence, the accuracy of the prediction is increased. Learning proceeds as follows.

Learning

1. Perform Reasoning[4]
2. Perform Adapting

The reason the algorithm is said to be nearest-hyperplane is that the reasoning phase essentially identifies a closest match for the (possibly filled-in) incoming vector. This closest match is either a rule (i.e., true hyperplane) or a stored example (i.e., point or degenerated hyperplane). The scheme for choosing a closest match is discussed in Section 3.3.

The system starts with an empty knowledge base. It then adapts to each new vector *v*, where *v* is either a precept or an example. If *v* is the first vector, then there is no closest match and *v* is automatically stored in the current knowledge base. In that sense, the first vector represents yet another bias for the learning system. If *v* is not the first vector, then a closest match, say *m*, is found and the knowledge base adapts itself (step 2) based on the relationship between *v* and *m*. Note that *m* is closest given the current available information, and can indeed be "far" from *v*. Hence, the order in which training takes place impacts the outcome. Chronology could therefore be used as another learning bias. An example of the implicit use of chronology to always give priority to new information is in PASOCS [24].

Notice that the adaptation phase takes place regardless of the target value predicted by the reasoning phase. A possible alternative would be to adapt only if the predicted target value differs from the actual target value. It has been found empirically, however, that too much useful information is lost with this approach due to the incrementality of the system and its sensitivity to ordering. A possibly viable alternative would make use of memory. Vectors currently accounted for could be saved in memory and presented later to the system. This may be done a few times over some period of learning time until either the vectors must be stored in the knowledge base (due to changes in the knowledge base) or they are discarded as the system has gained enough confidence in its ability to account for them. The problem being addressed here is the inherent non-monotonicity of learning and reasoning.

3.4.3. Generalization. Two vectors can produce a generalization when the following five conditions hold.

1. they differ in the value of exactly one of their R-attributes
2. the R-attribute on which they differ is nominal
3. they are concordant
4. the number of their R-attributes not equal to $*$ differ by at most 1[5]
5. at least one of them has more than one non $*$ R-attribute

Generalization then consists of setting to ∗ the R-attribute on which the two vectors differ, in the vector that is most general, as long as that vector has more than one non ∗ R-attribute. For example, vectors v_8 and v_9 of *KB* satisfy the above conditions and would generalize to produce a vector $v_{8\text{-}9} = (*, *, *, 0, 0, 1, 3_T)$.

Choosing the most general vector maximizes generalization, and the condition on the number of non ∗ R-attributes guarantees that no rule is generated that would cover every other vector. This rule is only applied to nominal attributes as it makes little sense for linear (especially real-valued) domains. For linear attributes, generalization is achieved through an artifact due to the extension of the notion of equality shown in Section 3.3.1 (see Note 5 in Section 3.4.4).

Though dropping the condition may be argued to also be too strong for large nominal domains, it is balanced by the system's ability to handle exceptions. Notice however that given a set of vectors of the form $v_i = S\ x_i\ T_k$ where S is fixed, T is the T-attribute, and $x_i \neq x_j$ for all $i \neq j$, any pair of concordant vectors satisfy the generalization condition, yet only the first such pair will generalize. All other vectors then become either covered by this generalization or exceptions to it. If most are exceptions, this leads to the storage of more vectors than needed, especially for large domains where various subsets of values give rise to different T-attribute's values. Moreover the outcome depends upon ordering of the vectors.

Also, if there exists conflicts involving one (or more) value of x, then the system will end up giving unfounded precedence to the exceptions (being more specific), and again these depend on the ordering. Support for internal disjunction, or a more complex generalization scheme may help alleviate some of these problems. They are the topic of future research. The current simple generalization scheme gives promising empirical results for induction (see Section 4).

3.4.4. Adapting. Along with each vector, FLARE stores the following information.

- a static priority value (*pty*),
- a dynamic priority value (*dyn_pty*),
- the number of vectors covered (*num_covers*),
- a definition indicator, and
- a set of counters (*counters*[]) for each possible value of the T-attribute.

The value of pty is set to 0 by default, but may be set to any other value *a priori*. The value of num-covers records the number of training items covered by the vector, and is a kind of confidence level for that vector. The definition indicator is a Boolean flag that identifies the vector as being a definition. It is used to perform completion during reasoning. The counters are all initialized to 0, except the one corresponding to the vector's T-attribute value which is initialized to 1. The counters evolve over time, and are used to handle noise. For any vector p in the current knowledge base, exactly one counter value is incremented (by 1) each time a new vector is presented, whose R-attributes' values are all equal to those of p. The value incremented corresponds to the new vector's T-attribute value. The counter value that is highest represents the statistically "most probable" T-attribute value. In effect, the T-attribute value of a vector is always the one with highest count. Note that this value may change over time, as new information becomes available.

The value of dyn_pty is initialized to 0. As it is intended to resolve conflicting defaults extensionally, its evolution requires the identification of such conflicts. Identification takes place

in the reasoning phase as FLARE discovers two rules (or precepts) that apply equally well to the new vector. Two rules *R* and *S* are in conflict for a vector *v* if the following conditions hold.

1. $D(R,v) = D(S,v) = 0$
2. *R* and *S* have the same specificity
3. *R* and *S* have the same static priority
4. *R* and *S* have different T-attribute's value
5. *R* and *S* overlap (i.e., the set of all possible vectors they cover intersect)

When reasoning and coming upon conflicting defaults, FLARE increments by 1 the value of dyn_pty of that default whose T-attribute's value is the same as *v*'s (i.e., the default that covers *v*). If none of the defaults cover *v*, then no change is made. Reasoning then proceeds as outlined in Section 3.3.3. Notice that dyn_pty evolves over time so that the system's response also changes based on accumulated evidence. Note that if none of the defaults cover *v*, then *v* is an exception to these defaults, and thus it will be stored in the knowledge base (see algorithm Adapting below). In turn, the resolution of conflicts would be handled by the counters rather than dyn_pty due to the reasoning's priority scheme. Functionally, the result is identical.

The details of step 2 (i.e., Perform Adapting) of the learning algorithm are as follows. Let *v* be a new vector, other than the first one, and let *m* be *v*'s closest match in the current knowledge base. The vectors *v* and *m* are said to be *concordant* if they have the same T-attribute's value.

Adapting

```
If (v.R-attributes' values = m.R-attributes' values)[6]        { duplicates or noise }
    add 1 to m.counters[v.T-attribute's value]
Else if (m covers v and m and v are concordant)
    add 1 to m.num_covers
Else if (v covers m and m and v are concordant)
    add 1 to v.num_covers
    delete m from the knowledge base
    add v to the knowledge base
Else if (v and m can produce a generalization)
    If (v is more specific than m and m has more than one non * R-attribute)
        drop the condition in m
        m.pty = max(m.pty, v.pty)
    Else if (v has more than one non * R-attribute)
        drop the condition in v
        v.pty = max(m.pty, v.pty)
        v.num_covers = m.num_covers
        delete m from the knowledge base
        add v to the knowledge base
    Else add v to the knowledge base
Else add v to the knowledge base
```

Because a best match is first identified, changes to the knowledge base are localized, and are guided by the kinds of possible relationships between *v* and *m*. These relationships are summarized below.

- v is identical to m (i.e., noise or duplicates), or
- v is covered by m (i.e., v is a special case of m), or
- v covers m (i.e., v is a general case of m), or
- v and m can produce a generalization, or
- all other cases (e.g., v is an exception to m, v and m are too far apart, etc.)

In the first two cases, there is no need to store v as the current knowledge base has sufficient information to correctly predict v's target value. In the third case, v is stored and m removed as v is more general than m, and thus accounts for it. In the fourth case, only one of v or m is generalized and stored. The values of pty and num_covers are set so that the generalization inherits the maximum static priority value and the current value of num_covers. Finally, in the fifth case, v must be added as the current knowledge base either does not produce the correct target value for v (e.g., exceptions), or is not deemed reliable enough to properly account for v.

3.4.5. A Detailed Example: Learning *KB*. This section shows how FLARE would learn *KB*. It highlights several interesting features of both reasoning and adaptation. Let *KB′* be the current knowledge base of FLARE. As discussed above, FLARE starts with $KB' = \varnothing$. Now assume that each vector in *KB* is presented to FLARE in the order in which it appears in *KB*. Also, assume that vectors v_7 through v_{17} are tagged as definitions.

1. *Presentation of* v_1. $KB' = \varnothing$. v_1 is simply added to *KB′*.

2. *Presentation of* v_2. $KB' = \{v_1\}$. v_1 is the closest match. v_1 and v_2 are not concordant, so v_2 is added to *KB′*.

3. *Presentation of* v_3. $KB' = \{v_1, v_2\}$. Same as above with either v_1 or v_2. So v_3 is added to KB'.

4. *Presentation of* v_4. $KB' = \{v_1, v_2, v_3\}$. No winner can be found since none of the vectors in *KB′* have the same T-attribute as v_4. So v_4 is added to *KB′*.

5. *Presentation of* v_5. $KB' = \{v_1, v_2, v_3, v_4\}$. v_4 is the only (and hence closest) match since none of the other vectors in *KB′* have the same T-attribute as v_5. v_4 and v_5 satisfy conditions 1-4 for generalization but violate condition 5, so v_5 is added to *KB′*.

6. *Presentation of* v_6. $KB' = \{v_1, v_2, v_3, v_4, v_5\}$. Same as step 3. So v_6 is added to *KB′*.

7. *Presentation of* v_7. $KB' = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. Note that v_7 is a precept. No winner can be found since none of the vectors in *KB′* have the same T-attribute as v_7. So v_7 is added to *KB′*.

8. *Presentation of* v_8. $KB' = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. v_7 is the only (and hence closest) match since none of the other vectors in *KB′* have the same T-attribute as v_8. v_8 is an exception to v_7 since v_7 covers v_8 but they are not concordant. Hence, v_8 is added to *KB′*.

9. *Presentation of* v_9. $KB' = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$. Only v_7 and v_8 may compete. $D(v_7, v_9) = 1/3$ and $D(v_8, v_9) = 1/4$. Hence, v_8 wins. v_8 and v_9 satisfy all five conditions for

generalization. The second attribute is dropped (i.e., replaced by $*$) in either one, say v_8, to produce $v_8' = (*, *, *, 0, 0, 0, 3_T)$. v_8' is added to *KB′*.

10. *Presentation of* v_{10}. *KB′* = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8'\}$. Only v_7 and v_8' may compete. $D(v_7, v_{10}) = 1/3$ and $D(v_8', v_{10}) = 0$. Hence, v_8' wins. v_8' covers v_{10} and they are concordant, so FLARE adds 1 to num_covers(v_8'). v_{10} need not be added to *KB′*.

11. *Presentation of* v_{11}. *KB′* = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8'\}$. Notice that v_{11} is also a precept, so that precepts may be given at any time during learning. Only v_7 and v_8' may compete. $D(v_7, v_{11}) = 1/6$ and $D(v_8', v_{11}) = 1/3$. Hence, v_7 wins. Neither one covers the other; they are not equal; they cannot produce generalization (violate condition 3). Thus, v_{11} is added to *KB′*. Note that v_{11} has a static priority of 1.

12. *Presentation of* v_{12}. *KB′* = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8', v_{11}\}$. v_7, v_8', and v_{11} compete. $D(v_7, v_{12}) = 1/2$, $D(v_8', v_{12}) = 2/3$, and $D(v_{11}, v_{12}) = 1/4$. Hence, v_{11} wins. Neither one covers the other; they are not equal; they cannot produce generalization (violate condition 3). Thus, v_{12} is added to *KB′*. Note that v_{12} has a static priority of 3.

13. *Presentation of* v_{13}. *KB′* = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8', v_{11}, v_{12}\}$. In this case, some non-asserted attributes of v_{13} may be asserted through reasoning, before the T-attribute. Rules v_2 and v_4 are applied to assert the second and fourth attributes respectively. The result is $v_{13}' = (1, 1, 0, 0, 0, 1, 2_T)$. v_7, v_8', v_{11}, and v_{12} compete to assert the T-attribute. $D(v_7, v_{13}') = 0$, $D(v_8', v_{13}') = 0$, $D(v_{11}, v_{13}') = 0$, and $D(v_{12}, v_{13}') = 1/2$. Both v_7 and v_8' win over v_{11} since they are more specific. However, v_7 and v_8' have the same specificity. In fact, they satisfy all five conditions that identify them as conflicting defaults in the current context. Hence, FLARE adds 1 to dyn_pty(v_7) since v_7 and v_{13}' are concordant. Reasoning then proceeds, giving precedence to v_7. Since v_7 covers v_{13}' and they are concordant, v_{13}' need not be added to *KB′*.

14. *Presentation of* v_{14}. *KB′* = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8', v_{11}, v_{12}\}$. Some non-asserted attributes of v_{14} may be asserted through reasoning, before the T-attribute. Rules v_2 and v_5 are applied to assert the second and fourth attributes respectively. The result is $v_{14}' = (1, 1, 1, 0, 0, 1, 2_T)$. The rest is identical to step 13. Now, dyn_pty(v_7) = 2, and v_{14}' is not added to *KB′*.

15. *Presentation of* v_{15}. *KB′* = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8', v_{11}, v_{12}\}$. Some non-asserted attributes of v_{15} may be asserted through reasoning, before the T-attribute. Rules v_3 and v_6 are applied to assert the second and fourth attributes respectively. The result is $v_{15}' = (2, 2, 2, 1, 1, 0, 0_T)$. v_7, v_8', v_{11}, and v_{12} compete to assert the T-attribute. $D(v_7, v_{15}') = 1$, $D(v_8', v_{15}') = 1$, $D(v_{11}, v_{15}') = 1$, and $D(v_{12}, v_{15}') = 1/2$. Hence, v_{12} wins. Neither one covers the other; they are not equal; they cannot produce generalization (violate condition 3). Thus, v_{15}' is added to *KB′*.

16. *Presentation of* v_{16} *and* v_{17}. *KB′* = $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8', v_{11}, v_{12}, v_{15}'\}$. Both are equal to v_{15}'. Neither v_{16} nor v_{17} need be added to KB', but the appropriate counter values are incremented in v_{15}'. The result is counters[0] = 2, counters[1] = 0, counters[2] = 1, and counters[3] = 0. Thus, the T-attribute's value of v_{15}' is currently 0.

The resulting *KB'*, after learning *KB* is as follows. The variables *p*, *c*, and *dp* stand for static priority, cover number, and dynamic priority, respectively.

$v_1 = 0\ 0_T * * * * *$ (p=c=dp=0)
$v_2 = 1\ 1_T * * * * *$ (p=c=dp=0)
$v_3 = 2\ 2_T * * * * *$ (p=c=dp=0)
$v_4 = * * 0\ 0_T * * *$ (p=c=dp=0)
$v_5 = * * 1\ 0_T * * *$ (p=c=dp=0)
$v_6 = * * 2\ 1_T * * *$ (p=c=dp=0)
$v_7 = * 1 * 0\ 0 * 2_T$ (p=c=0, dp=2)
$v_8' = * * * 0\ 0\ 1\ 3_T$ (p=dp=0, c=1)
$v_{11} = * 1 * * 0 * 0_T$ (p=1, c=dp=0)
$v_{12} = * 1 * 1 * * 1_T$ (p=3, c=dp=0)
$v_{15}' = 2\ 2\ 2\ 1\ 1\ 0\ 0_T$ (p=c=dp=0)

To further demonstrate reasoning, assume that a new vector $v = (1, ?, 0. ?, 0, 1, ?_T)$ is presented to FLARE after *KB'* is acquired. The second and fourth attributes are asserted as previously discussed to give $v' = (1, 1, 0, 0, 0, 1, ?_T)$. Then, v_7, v_8', and v_{11} compete. v_7 and v_8' win due to specificity. v_7 and v_8' also have same static priorities, but v_7 wins due to dynamic priority and the result is $(1, 1, 0, 0, 0, 1, 2_T)$. If a vector $(1, ?, 2, ?, 0, 0, ?_T)$ is presented, a similar situation would arise between v_{11} and v_{12}. The conflict would be resolved with static priorities.

3.4.6. Extensionality and Intensionality. Precepts may have a static priority associated with them. Both rules and precepts keep track of the number of examples they cover. Rules are the result of extensionality, while precepts are intensional. With this combination of the intensional and extensional approaches, FLARE can resolve conflicting defaults, like the Nixon Diamond [29], by either being told which default prevails (e.g., religious conviction is more important than political affiliation), or by computing a dynamic priority (see discussion in sections 3.3.3 and 3.4.4) from examples of Republican-Quakers.

Most learning systems are purely extensional, while most reasoning systems are purely intensional. It is our contention that a combination of the two approaches offers the maximum flexibility. On the one hand, extensionality accounts for the system's ability to adapt to its current environment, i.e., to be more autonomous. On the other hand, intensionality provides a mechanism by which the system can be taught, and thus does not have to unnecessarily suffer from poor or atypical learning environments.

In the context of reasoning, precepts provide a useful medium to encode certain first-order language statements (e.g., the rule base of an expert system), that can in turn be learned by FLARE (in the usual way), and later be used for reasoning purposes.

3.5. FLARE's Automatic Generation of Precepts

Section 3.4. introduced the notion of precepts, and argued their usefulness as learning biases and rule encoding (in the classical sense). Precepts may be obtained from the domain knowledge of a human teacher. In the context of the construction of expert systems, this process is called knowledge elicitation. In this section however, we focus on one mechanism for the automatic generation of precepts implemented in FLARE.

Precepts can be viewed as some instantiation of commonsense knowledge to a particular domain. Hence, if some commonsense knowledge is available, it may potentially be used to generate precepts. Several attempts have been made at formalizing commonsense knowledge [2, 22], and a huge database of commonsense facts, called Cyc, is currently being compiled [20]. With such facts expressed in first-order language, and an appropriate setting for instantiation,

FLARE can generate precepts that can then be used as biases for inductive learning, or for further reasoning about the specific domain to which they apply.

Consider Example 4 from Section 3.1. Assume that the system is to inductively learn rules regarding the suitability of lenses for patients from a set of examples whose attributes include the patient's tear-production rate (tpr). The first two statements in Example 4 capture commonsense knowledge about eyes, the relationship between low tear-production rate and dryness, and the danger of placing objects (or things) in dry eyes. When instantiated with the third statement (i.e., lenses are things), the general knowledge may be used to produce a domain-dependent precept that states that if a patient has low tear-production rate, then he/she should not be fitted lenses. The precept in turn provides a useful bias to the system during induction.

The process of generating precepts (or instantiating commonsense) is essentially one of acquiring commonsense and reasoning from it. In general, it proceeds as follows.

Generate_precepts

1. Perform Learning using
   a. Available commonsense rules.
   b. Rules that establish the connection between the domain and commonsense.
2. Perform Reasoning with
   a. The R-attributes asserted as given in the rules (all others are $*$).
   b. The T-attribute designated as the target.

The result of step 2 is a domain-dependent precept. The above algorithm uses mechanisms that are well defined in FLARE. Prior to any other learning, and if some commonsense knowledge is provided, FLARE always apply the Generate_precepts algorithm. In Example 4 of Section 3.1, the three statements are used in step 1, with the third statement establishing the needed connection. In step 2, the attribute lenses is asserted to 1 (i.e., true), the attribute Tpr is asserted to low, and the attribute Fit is designated as the target.

Step 2 of the Generate_precepts algorithm relies on a teacher setting up the correct attributes. More automatic mechanisms could be used, where the system could try any combination of the learning problem's R-attributes values to instantiate commonsense knowledge. Then, any such instantation that causes the T-attribute to become asserted is a potential precept. Of course, that process is exponential, and most of it would probably not lead to any useful conclusion.

4. Simulation Results

A set of classical commonsense benchmark problems is proposed in [21], and [27] contains many useful training sets for inductive learning. This section reports results of simulations of FLARE on several of these datasets.

One artifact of the implementation is that since variables can not be dynamically added, all attributes must be defined *a priori*. All attributes that do not appear in rules, examples, or precepts are set to don't-care. This is consistent with the semantics of don't-care, and does not interfere with the algorithm since the distance D essentially treats learned don't-cares as neutral values.

4.1. Inductive Learning and Prior Knowledge

Several datasets from [27] were chosen. They represent a wide variety of situations, as shown in Table 2. The column Size indicates the total number of examples in the dataset. The column Attributes records the number and type (L for linear, N for nominal) of all the attributes, other than the target (or output) attribute. The column Output shows the number of output classes.

Table 2 - Selected Applications

Application	Size	Attributes	Output
lenses	24	1L/3N	3
voting-84	435	16N	2
tic-tac-toe	958	9N	2
hepatitis	155	6L/13N	2
zoo	90	16N	7
iris	150	4L	3
soybean-small	47	4L/31N	4
segmentation	420	19L	7
glass	214	9L	7
breast-cancer	699	9L	2
sonar	208	60L	2
LED	1000	7N	10

FLARE's results were gathered for each of the above applications. They are shown in Table 3. Each of the reported results is an average over 10 runs of FLARE. The dataset is split into two halves, one of which, the training set, is used for learning, and the other, the test set, only for reasoning. In each run, the training set and the test set are randomly regenerated. The first number (PA) represents predictive accuracy (in %) on the test set after training and the second number (IR) is the inductive ratio, defined as the ratio of the size (in number of rules) of the final knowledge base to the number of instances used in learning. IR is another measure of the generalization power of FLARE, as well as an indication of FLARE's memory requirements.

Table 3 - FLARE: Induction

Application	PA	IR
lenses	78.3	.58
voting-84	91.6	.68
tic-tac-toe	80.4	1.0
hepatitis	80.8	.97
zoo	95.3	.47
iris	95.1	.20
soybean-small	99.1	1.0
segmentation	92.2	.99
glass	58.9	.27
breast-cancer	95.8	.49
sonar	81.7	.83

LED	70.4	.15
Averages	**85.0**	**.64**

These results compare favorably with results reported for other inductive learning algorithms (see [1, 28, 37, 39]). Note that LED is a noisy application, for which the Bayesian optimal is 74%. FLARE handles the noise well, both by achieving good predictive accuracy, and by constructing a significantly small network.

Simulations of the first five applications with precepts given *a priori* (before the training set is presented) are reported in Table 4. Each column shows both PA and IR. Here, the precepts are obtained from domain knowledge provided with the application (voting-84) or generated from the authors commonsense (zoo, lenses, hepatitis, tic-tac-toe). They serve as learning biases.

Table 4 - FLARE: Induction and Prior Knowledge

Application	no prec.	w/prec.
lenses	78.3 - .58	81.7 - .42
voting-84	91.6 - .68	94.9 - .28
tic-tac-toe	80.4 - 1.0	87.7 - .72
hepatitis	80.8 - .97	82.2 - .66
zoo	95.3 - .47	95.3 - .42
Averages	**85.3 - .74**	**88.4 - .50**

The results with precepts show an average increase of 3.6% in predictive accuracy, and a decrease of 32.4% of IR (i.e., the resulting networks are significantly smaller on average).

4.2. Classical Commonsense Protocols

Several problems from the set of Benchmark Problems for Formal Nonmonotonic Reasoning [21], were presented to FLARE. The problems were first translated into their corresponding AVL representation. FLARE is able to properly incorporate the premises, and correctly derive the expected conclusions for the following classes of problems from [21]:

- A1 - basic default reasoning[7]
- A2 - default reasoning with irrelevant information
- A3 - default reasoning with several defaults
- A5 - default reasoning in an open domain
- A9 - priority between defaults
- B1 - linear inheritance (top-down)
- B2 - tree-structured inheritance
- B3 - one-step multiple inheritance
- B4 - multiple inheritance

Problem A4 involves a disabled default, and problems A6 through A8 deal with unknown exceptions. Such problems can not be represented in FLARE. Problems A10 and A11 deal with instances of defaults and reasoning about priority. Though not directly representable in FLARE,

they are effectively solved via the use of static (A10) or dynamic (A11) priorities. The other classes of problems in [21] (i.e., reasoning about actions, uniqueness of names, and autoepistemic reasoning) are beyond the current scope of FLARE.

In addition to the above protocols, another important commonsense protocol is one based on similarity, where if A implies B and C is similar to A, then instantiating C causes the assertion of B. This is exemplified by the Chaco example discussed earlier, and involves completion. With a value greater than 0 for the threshold T_D (see Section 3.3.5), FLARE correctly performs completion and forward chaining based on similarity to derive the proper conclusion.

The above problems are characteristics of important forms of human commonsense reasoning. However, they are artificial as they have been manufactured explicitly with the intent of isolating one salient feature of nonmonotonic reasoning, independent of all others. To further investigate the properties of FLARE and the combination of learning and reasoning, other more "real-world" applications must be designed and experimented with. The following section presents such preliminary applications.

5. Results on More Real-World Applications

In this section, several useful and more realistic applications are reported, that further exercise FLARE's abilities. Extensionality and the combination of learning with reasoning are of particular interest.

5.1. The Nixon Diamond

The Nixon Diamond [29], reproduced as Example 1 in Section 3.1, is important as a prototype of a class of interesting problems involving conflicting defaults. It is used here to demonstrate FLARE's mechanisms to handle such conflicts both intensionally and extensionally.

FLARE's static priorities offer a simple way of resolving the Nixon Diamond intensionally, based on some externally provided information (e.g., religious convictions supersede political affiliations). In that case, both defaults are given along with an appropriate static priority.

Another alternative consists of providing the defaults without any priority. This corresponds to a possibly more natural situation where the system really is in a don't-know state when it comes to deciding on Nixon's dispositions. Yet, such don't-know states are uncomfortable and it is our contention that any kind of information that may allow a decision to be made should be used. Hence, we choose an epistemological approach, where the conflict arises due to beliefs rather than facts. In this case, it is possible to attempt to resolve the conflict by observing instances of Republican-Quaker. The relative number of Pacifist and non Pacifist could then serve as evidence to lean towards one decision or the other. In other words, it is the system's observation of what seems most common in its environment that creates its belief. This is not unlike the way humans deal with many similar situations.

A final approach, which combines inductive learning and reasoning, consists of not providing the system with any default. Rather, examples of Republicans, Quakers, and Republican-Quakers are shown and the system automatically comes up with both the defaults (through induction) and their relative priorities.

All three of these experiments were run with FLARE, and the results are as expected. In the third case, the actual knowledge base depends upon the ordering. It consists of one vector for

Republicans that are not Quakers or one vector for Quakers that are not Republicans, one default vector for Quakers or Republicans, and one vector for Republican-Quakers. The target value of the vector for Republican-Quakers is not had via dyn_pty, but via the counters. Functionally, the result is identical.

5.2. Do Birds Typically Fly?

Incremental learning is one of FLARE's important features. With incrementality, the system is self-adaptive in the sense that its current knowledge base is representative of its experience with its environment so far. And the knowledge base can be continually updated as new information becomes available. To exercise incrementality a simple example of bottom-up inheritance involving birds was designed.

The application has four R-attributes, two of which correspond to Penguin and Bird. The other two are other (undetermined) attributes of birds (e.g., Feather). The T-attribute is Boolean and characterizes the ability to fly. At first, the system is exposed mostly to ostrich-birds (maybe the experiment is started in Australia). When asked whether birds typically fly (i.e., only the Bird attribute is asserted, and all other inputs are don't-know), FLARE concludes that birds do not fly, which is consistent with its current experience with the "world." However, as more new instances of flying birds (i.e., other than ostriches) are encountered, FLARE adapts its knowledge and when asked again, concludes that birds fly.

Of course, a precept may also be given to the system at any given time, stating that birds typically fly. The idea is that FLARE offers both options naturally. The system may be taught so as not to suffer from poor or atypical learning environments (e.g., Australia for birds' flying ability prediction), or it may be left to adapt to its environment. As research on autonomous agents continues, this later ability becomes very attractive and useful.

5.3. Learning Expert Systems

In order to better assess FLARE's reasoning mechanisms, two expert systems knowledge bases were used. One is called *mediadv* [17] and is intended to help designers or committees choose the most appropriate media to deliver a training program. It consists of 20 rules with chains of inference of length at most 2. The other is called *health* [32] and is intended to predict the longevity of patients based on a variety of factors (e.g., weight, personality, etc.). It is much larger as it contains 77 rules, and more complex as it involves longer chains of inference. Five rules were left out as one is redundant (i.e., rule 17 is identical to rule 14) and four are only needed in the interactive setting in which the original system is described. Hence, only 72 rules are considered.

Both sets of rules were translated into AVL. The 20 rules of *mediadv* produce 99 vectors, and the 72 rules of *health* produce 72 vectors. The number of vectors for *mediadv* is much larger than the original number of rules because many of the rules contain internal disjunctions. In AVL, a new vector must be constructed for each possible combination arising from the disjunctions. For example, the rule: If ((A=1 or A=2 or A=5) and (B=2.9 or B=7.8)) then C, gives rise to 6 vectors corresponding to the equivalent set of rules: If ((A=1) and (B=2.9)) then C, If ((A=1) and (B=7.8)) then C, If ((A=2) and (B=2.9)) then C, etc.

The sets of vectors corresponding to the original knowledge bases are not encoded into FLARE. Rather, they are presented to the system to be learned (i.e., the algorithm Learning is

applied to them). Hence, some generalization may take place. In fact, the final number of vectors (after learning) in *mediadv* is only 71, while in *health* it is 65.

FLARE seems to perform reasonably well when asked to reason about its acquired knowledge. The matrix analysis of *mediadv* in [17] was used to derive the 46 possible situations covered by the 20 rules in the knowledge base. After learning the rules, FLARE was exposed to these 46 situations and derived the proper conclusion for all except three. The three situations missed correspond to pairs of situations having identical premises, but different conclusions, and result from the inherent inconsistency (conflict between rules 13 and 14, and rules 16 and 17) in the knowledge base. In other words, there are only 43 distinct situations. When a precedence scheme is selected, and one vector from each pair is removed, FLARE derives the proper conclusion for all remaining 43 situations. This result is significant here because FLARE generalizes the original knowledge base (from 99 to 71 vectors). Hence, the knowledge base learned by FLARE may not be logically equivalent to the original one. It appears however that, at least in the case of *mediadv*, the system is logically equivalent and also more parsimonious. A similar experiment is difficult to conduct for the *health* knowledge base as the number of possible situations it covers is much larger. A formal investigation of FLARE's generalization scheme is necessary to provide a result on logical equivalence in the general case.

Further representative experiments on new situations (i.e., not explicitly encoded in the knowledge bases) were performed. They are summarized in Table 5 and Table 6. In each table the first column contains the list of attributes used in the knowledge base. Then, each pair (setting, result) of columns represents an experiment in reasoning with the knowledge base. The setting column contains the data FLARE starts with. Unknown conditions (or attributes) are initialized as don't-knows (i.e., ?). The result column shows the state of knowledge after reasoning. Each derived piece of information is italicized and subscripted by the depth of inference at which it was derived.

Table 5 - Mediadv Knowledge Base

	setting 1	**result 1**	**setting 2**	**result 2**	**setting 3**	**result 3**	**setting 4**	**result 4**
situation	view	view	diagram	diagram	object	object	diagram	diagram
stim. sit.	?	*pictorial*$_1$	?	*pictorial*$_1$	?	*environ.*$_1$	?	*selective*$_1$
response	building	building	moving	moving	?	?	matching	matching
app. resp.	?	*motor*$_1$	?	*motor*$_1$	?	?	?	
duration	persistent	persistent	brief	brief	?	?	brief	brief
feedback	yes	yes	yes	yes	?	?	yes	yes
when	response	response	response	response	?	?	response	response
budget	medium	medium	medium	medium	?	?	small	small
media	?	*self-study*$_2$	?	*tutor*$_2$	?	*lab*$_2$	?	*tutor*$_2$

The first setting matches both rules 11 and 20, so one of them is chosen at random. In the second setting, the change in duration from persistent to brief suffices to bring the situation closer to rule 20. The third setting tests the system in cases where very little information is available, but that information is relevant. Only one rule contains a premise for stimulus-situation equal to environmental (rule 7). It is therefore applied. The fourth setting was designed to reflect a pattern recognition scenario that may be used in a first-grade classroom, where the children would

be shown various shapes and their names, and then asked to classify new objects in terms of their shapes. In this case, the system's prediction seems to match intuition. It suggests the use of a human tutor to perform the training. Notice that this last setting and its outcome give support to the (now classical) notion of supervised learning.

Table 6 - Health Knowledge Base

	setting 1	**result 1**	**setting 2**	**result 2**	**setting 3**	**result 3**
rel. weight	?	*normal*$_1$	?	*obese*$_1$	?	*normal*$_1$
val	?	*yes*$_2$	?	?	?	*yes*$_2$
heart-dis-risk	?	?	?	*>average*$_1$	?	*average*$_1$
hddanger	?	?	?	?	?	*low*$_2$
start	yes	yes	yes	yes	yes	yes
age	25-55	25-55	>55	>55	<25	<25
gender	F	F	M	M	F	F
base-longevity	?	*67*$_1$	?	*60*$_1$	?	*72*$_1$
weight	<110	<110	>170	>170	110-170	110-170
frame	small	small	small	small	large	large
cholesterol	?	?	high	high	low	low
fat intake	normal	normal	high	high	normal	normal
salt intake	normal	normal	high	high	normal	normal
blood-pressure	?	*average*$_1$	?	*>average*$_1$	?	*average*$_1$
calcium	?	?	?	?	normal	normal
osteoporo-risk	?	?	?	?	?	*average*$_1$
smoker	no	no	yes	yes	no	no
outlook	?	?	?	*bleak*$_2$	?	*fair*$_3$
race	asian	asian	caucasian	caucasian	caucasian	caucasian
origin	?	?	medit.	medit.	n-amer.	n-amer.
risk	?	*high*$_2$	?	*high*$_2$	unknown	unknown
personality	aggressive	aggressive	aggressive	aggressive	docile	docile
person. type	?	*type_a*$_1$	?	*type_a*$_1$	?	*type_b*$_1$
alcohol cons.	moderate	moderate	excessive	excessive	none	none
add	?	*good*$_1$	?	*poor*$_1$	?	*fair*$_1$
factor	?	*none*$_3$	?	*minus_12*$_3$	?	*plus_12*$_4$
longevity	?	*67*$_4$	?	*48*$_4$	?	*84*$_5$

Notice the greater length in the chains of inference compared to *mediadv*. The first setting corresponds to an average individual, with little vices or excesses. As one would expect, its longevity is average. The second setting corresponds to a very unhealthy individual which one would not expect to live very long. FLARE concludes accordingly. The third setting corresponds to a healthy female individual which one would expect to live a good number of

years, as predicted by the system. Another setting, not shown here, was used in which only the value of start was set to "no." As expected, the system concludes with longevity value "quit."

Note that as in classical expert systems, the identification of a closest match during reasoning can be used to query the user for missing information as well as to justify both the queries and the decisions made. Of interest also is the case of conflicts. As in the previous examples, learning can beneficially be combined with reasoning. For example, in *mediadv*, the following rules are found, where X is some fixed conjunction of conditions.

- rule 13: if (X) and (training_budget = small or training_budget = medium) then
  media_to_consider = lecture
- rule 14: if (X) and (training_budget = medium) then
  media_to_consider = lecture-with-slides

It is clear that in some cases, these rules conflict. It is difficult to avoid such occurrences in large knowledge bases elicited form experts. As FLARE supports learning, it is possible however to look at various (historical) situations where training_budget was medium, and which media was used. This information can in turn be used to give precedence to one rule over the other. Moreover, this precedence need not be fixed after so many examples have been considered. Indeed, it may evolve over time, and even change radically depending on circumstances. Alternatively, static priorities allow

An example was implemented in which several instances of [(X) and (training_budget = medium)] together with a target value for media_to_consider. The instances used cause the value of dyn_pty of rule 13 to be greater than that of rule 14, thus effectively giving (evidential) precedence to rule 13.

5.4. Generating Precepts

Section 3.5 discussed the possibility of using reasoning to instantiate some commonsense knowledge to a particular domain, thus producing domain-dependent precepts that can in turn be used as learning biases during induction.

Example 4 from Section 3.1 was implemented (as described in Section 3.5) and a precept stating that if the Tpr attribute is set to low, then lenses should not be prescribed was generated. That precept was in turn used prior to performing inductive learning on the lenses dataset of Section 4.1.

The above process is two-phase, where both phases perform the same operations on different pieces of information. In the first phase, commonsense knowledge expressed in first-order logic (and translated into AVL) is learned by FLARE. Then FLARE reasons based on some instantiation that links the general commonsense knowledge to the current domain. The result of this reasoning phase is one (or more) vector containing domain-dependent information. In the second phase, FLARE learns from the generated domain-dependent information and any other available examples. The result is a set of inductively generated rules.

5.5. Further Experiments

The above applications serve to demonstrate that FLARE holds promise. Further usage of the proposed framework will be developed. In particular, work must be pursued in finding ways to

reason about inductively generated rules. For example, in the Chaco example, the definitions used were given *a priori*, rather than learned by induction. Several questions remain open.

- Will inductively learned rules always be good for reasoning?
- What forms of reasoning would they support?
- How do we deal with disjunctive representations?
- Where do we find applications of such systems (like the UC Irvine repository)?

Another interesting avenue to pursue is the discovery of mechanisms to use the ability to reason as a guide to further learning. For example, is it possible to generate other types of biases than precepts? Or, could the identification of a conflict (such as the one discussed in Section 5.3) be used to determine what kinds of examples to seek?

6. Conclusion

This paper introduces a model that attempts to combine inductive learning with prior knowledge and reasoning into a unified framework. The system, called FLARE, is detailed. Its reasoning mechanism is a combination of rule-based reasoning and similarity-based reasoning, which reduces the susceptibility to brittleness. Reasoning is inherently nonmonotonic and accounts for many useful commonsense protocols. Learning is incremental and follows a form of nearest-hyperplane learning. Results on several real-world datasets compare favorably with other similar inductive learning systems.

Several applications of the framework are presented, including the automatic generation of precepts to bias inductive learning and the use of examples to resolve conflicting defaults extensionally. Current results demonstrate promise. Future work includes the following:

- Finding mechanisms to use reasoning to guide learning
- Attempting to overcome (or appropriately use) the order-dependency
- Providing support for internal disjunction
- Using inductively learned rules in reasoning (the support is available, but the induction may not produce useful rules)
- Possibly incorporating backward chaining
- Translating system's knowledge base back from AVL to FOL
- Further experimenting with larger applications

Acknowledgments

This work was supported in part by grants from Novell Inc. and WordPerfect Corp.

Notes

[1]The philosopher Pierce actually identified three basic forms of reasoning: inductive, deductive and abductive. See Hartshorn et al. (Eds.), *Collected Papers of Charles Sanders Pierce*, Harvard University Press, Vol. 2, pp. 1931-1958. We do not address abductive mechanisms here.

[2]These are either ∗ or ?. They are ∗ when precepts and rules with differing premises and conclusions are used. In such cases, it is not clear until reasoning whether they are true don't-cares (see Section 3.4.1) or only don't-knows.

[3]The accuracy, or confidence level may be computed in a variety of ways from information about static priorities, dynamic priorities, covers, and counters (see Section 3.4). It may be given relative to other alternatives (e.g., this decision is twice as reliable as this other one based on their relative static or dynamic priorities), or in absolute (e.g., this decision has a confidence of 95% based on counters). A wide range of possibilities is available.

[4]Reasoning based upon available knowledge prior to adapting is plausible. Even when available information is insufficient and/or incomplete, humans often attempt to make a tentative decision and get corrected if necessary. The more information becomes available, the more accurate decisions become. But, even in a don't-know state, humans (if they have to) are able to make a best guess that corresponds to their current representation of the world.

[5]This value is based upon empirical evidence.

[6]Note that the extension of the notion of equality discussed in Section 3.3.1 allows this part of the algorithm, in conjunction with the counters, to produce some generalization for linear attributes. In effect, the vector retained in the knowledge base acts as a "prototype," and its T-attribute's value is the one most probable among its neighbors.

[7]A1's problem statement includes implications as well as facts. It is given as follows: blocks A and B are heavy, heavy blocks are normally located on the table, A is not on the table. Translating to AVL gives: A 1_T ∗, B 1_T ∗, ∗ 1 1_T, A ∗ 0_T. Now, if A ? $?_T$ is shown, A 1 $?_T$ will be derived from the first vector, and then A 1 $?_T$ matches exactly ∗ 1 1_T and A ∗ 0_T. Priority should be given to the later since it involves A explicitly. To solve this problem, vectors involving explicit references to instances of objects have their static priority set to 1 while all other vectors have their static priority set to 0. This is of course an artifact of encoding. An alternative is to write all facts relative to a given instance as a definition whose target is the instance value. Then completion would guarantee the correct outcome. The same applies to problems A2, A3, and A5.

References

[1] Aha, D.W., Kibler, D., and Albert, M.K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, 6, 37-66.

[2] Brewka, G. (1991). *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press.

[3] Buntine, W.L. (1990). A Theory of Learning Classification Rules. Ph.D. Dissertation, University of Technology, School of Computing Science, Sydney.

[4] Clancey, B.C., and Shortliffe, E.H. (1984), Editors. *Readings in Medical Artificial Intelligence: The First Decade*. Reading, MA: Addison-Wesley Publishing Company.

[5] Clark, K.L. (1978). Negation As Failure. In Gallaire, H., and Minker, J., (Eds.). *Logic and Databases*. Plenum Press, 293-322.

[6] Clark, P., and Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning*, 3, 261-283.

[7] Collins, A., and Michalski, R.S. (1989). The Logic of Plausible Reasoning: A Core Theory. *Cognitive Science*, 13, 1-49.

[8] D'Ignazio, F., and Wold, A.L. (1984). *The Science of Artificial Intelligence*. Franklin Watts Library Edition.

[9] Duda, R.O., and Reboh, R. (1984). AI and Decision Making: The PROSPECTOR Experience. In Reitman, W., (Ed.). *Artificial Intelligence Applications for Business*. Norwood, NJ: Ablex Publishing Corp.

[10] Elman, J.L. (1991). Incremental learning, or The importance of starting small. CRL Technical Report 9101, La Jolla, CA: University of California, San Diego, Center for Research in Language.

[11] Giraud-Carrier, C., and Martinez, T.R. (1993). Using Precepts to Augment Training Set Learning. In *Proceedings of the First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems (ANNES'93)*, 46-51.

[12] Giraud-Carrier, C., and Martinez, T.R. (1994). An Efficient Metric for Heterogeneous Inductive Learning Applications in the Attribute-Value Language. To appear in *Proceedings of the Third Golden West International Conference on Intelligent Systems (GWIC'94)*.

[13] Giraud-Carrier, C., and Martinez, T.R. (1994). An Incremental Learning Model for Commonsense Reasoning. To appear in *Proceedings of the Seventh International Symposium on Artificial Intelligence (ISAI'94)*.

[14] Giraud-Carrier, C., and Martinez, T.R. (1994). ILA: Combining Inductive Learning with Prior Knowledge and Reasoning. Submitted.

[15] Haas, N., and Hendrix, G.G. (1983). Learning by Being Told: Acquiring Knowledge for Information Management. In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M., (Eds.). *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann Publishers, Inc., Vol. I, Chapter 13.

[16] Hall, L.O., and Romaniuk, S.G. (1990). A Hybrid Connectionist, Symbolic Learning System. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 783-788.

[17] Harmon, P., and King, D. (1985). *Expert Systems*. John Wiley & Sons, Inc.

[18] Horty, J.F., Thomason, R.H., and Touretzky, D.S. (1990), A Skeptical Theory of Inheritance in Nonmonotonic Networks. *Artificial Intelligence*, 42, 311-348.

[19] Kundu, S. (1994). The Completion Principle: A Technique for Discovering New Concepts in Inductive Learning. In *Proceedings of the Seventh Florida AI Research Symposium (FLAIRS'94)*, 126-130.

[20] Lenat, D.B., et *al*. (1990). Cyc: Toward Programs With Common Sense. *Communications of the ACM*, Vol. 33, No. 8, 30-49.

[21] Lifschitz, V. (1988). Benchmark Problems for Formal Nonmonotonic Reasoning. In *Proceedings of the 2nd International Workshop on Non-Monotonic Reasoning*, LNCS 346, 202-219.

[22] Lukaszewicz, W. (1990). *Non-Monotonic Reasoning: Formalization of Commonsense Reasoning*. Ellis Horwood Limited.

[23] Martinez, T.R. (1986). Adaptive Self-Organizing Networks. Ph.D. Thesis, UCLA Tech. Rep. - CSD 860093.

[24] Martinez, T.R., Hughes, B., and Campbell, D.M. (1994). Priority ASOCS. *Journal of Artificial Neural Networks*. To appear.

[25] Michalski, R.S. (1983). A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, 20, 111-161.

[26] Mitchell, T.M. (1980). The Need for Biases in Learning Generalizations. CBM-TR 5-110, Rutgers University, New Brunswick, NJ.

[27] Murphy, P.M., and Aha, D.W. (1992). *UCI Repository of machine learning databases*. Irvine, CA: University of California, Department of Information and Computer Science.

[28] Quinlan, J.R. (1986). Inductive Learning of Decision Trees. *Machine Learning*, 1, 81-106.
[29] Reiter, R., and Griscuolo, G. (1981). On Interacting Defaults. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 270-276.
[30] Rychener, M.D. (1983). The Instructible Production System: A Retrospective Analysis. In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M., (Eds.). *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann Publishers, Inc., Vol. I, Chapter 14.
[31] Salzberg, S. (1991). A Nearest Hyperrectangle Learning Method. *Machine Learning*, 6, 277-309.
[32] Sawyer, B., and Foster, D.L. (1986). *Programming Expert Systems in Pascal*. John Wiley & Sons, Inc.
[33] Stanfill, C., and Waltz, D. (1986). Toward Memory-Based Reasoning. *Communications of the ACM*, Vol. 29, No. 12, 1213-1228.
[34] Sun, R. (1992). A Connectionist Model for Commonsense Reasoning Incorporating Rules and Similarities. *Knowledge Acquisition*, 4, 293-321.
[35] Sun, R. (1993). An Efficient Feature-Based Connectionist Inheritance Scheme. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 2, 512-522.
[36] Waterman, D.A. (1986). *A Guide to Expert Systems*. Addison Wesley.
[37] Wettschereck, D., and Dietterich, T.G. (1994). An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms. *Machine Learning*. To appear.
[38] Young, A., and Sattar, A. (1993). Nonmonotonic Inheritance Reasoning in Classifiers. In *Proceedings of the First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems (ANNES'93)*, 160-162.
[39] Zarndt, F. (1994). An Empirically Derived Test Suite for Machine Learning and Connectionist Algorithms. In preparation.