Machine Learning, 10, 311-339 (1993)

Integrating Feature Extraction and Memory Search

CHRISTOPHER OWENS OWENS@CS.UCHICAGO.EDU
The University of Chicago, Department of Computer Science, 1100 E. 58th Street, Chicago, IL 60637

Abstract. Reasoning from prior cases or abstractions requires that a system identify relevant similarities between the current situation and objects represented in memory. Often, relevance depends upon abstract, thematic, costly-to-infer properties of the situation. Because of the cost of inference, a case-retrieval system needs to learn which descriptions are worth inferring, and how costly tht inference will be. This article outlines the properties that make an abstract thematic feature valuable to a case-based reasoner, and recasts the problem of case retrieval into a framework under which a system can explicitly and dynamically reason about the cost of acquiring features relative to their information value.

Keywords. Memory, retrieval, case-based reasoning, similarity judgement

1. Retrieval, description, and learning

For a case-based reasoner to make effective use of recalled prior experiences, it must be able to judge which of its cases are applicable to the current situation. This problem is not new, nor is it unique to case-based reasoning: any system that reasons on the basis of stored knowledge must determine when an element of that stored knowledge is relevant to its current processing. Frame, script, and schema-based reasoning systems such as described by Minsky (1975), Schank and Abelson (1977), and Charniak (1981) all must choose among their knowledge structures on the basis of relevance to the current task.

1.1. The task of retrieval

Although the goal is to retrieve *relevant* knowledge structures or prior cases, the concept of relevance is poorly understood and not easily described in computational terms. A major goal in case-based reasoning is to redefine relevance in terms of better-understood and more computational algorithms and representational constructs, or at least to find some computable property that can be made to stand in for relevance.

Many case-based reasoning systems determine relevance on the basis of features shared between the curent situation and prior cases in memory. A prior case sharing many features with the current situation is considered more likely to be relevant than a prior case sharing few features. Computationally, this means that a case-based reasoner needs:

- A **vocabulary** of features used to describe situations,
- A **case library** of data structures describing prior cases, each represented using features from the system's vocabulary,

- An **inference mechanism** for determining which features apply to newly encountered situations, and
- A **matcher** or a **lookup mechanism** for finding prior cases that share features with the current situation.

Case retrieval, under this view, is generally implemented as a process involving two discrete steps: First the inference mechanism develops a sketchy description of the current situation, and next the matcher or lookup mechanism uses that description to search memory. Although this approach seems to reduce case retrieval to the computationally better-understood task of pattern matching or indexed database retrieval, it suffers from an important shortcoming, one that the work in this article attempts to address.

1.2. Underconstrained description

In this article I argue against the disjoint, or describe-and-search view of case retrieval, and in favor of a model in which developing a description of the current situation is more closely integrated with searching for a relevant prior case. The main argument is that the disjoint model underspecifies the content of the vocabulary and the behavior of the inference mechanism: that the disjoint model ignores a valuable set of constraints, provided by the contents of memory and the process of memory search, that can be brought to bear on the choice of vocabulary and on the task of description.

Feature-based case retrieval is dependent upon the system's (or the system designer's) choice of descriptive vocabulary being a good one, and upon the inference mechanism's ability to generate good descriptions of the situations it enounters. But there is no external criterion by which the goodness or accuracy of a sketchy description can be judged. The goodness of a description is determined solely by the degree to which a pattern-matcher can use it to select cases from memory that are in fact relevant or useful, and therefore the appropriateness of a description depends not only upon the current situation, but also upon the contents of memory. A description that usefully describes a given situation, relative to a particular case library (i.e., one that allows the system to discriminate among its cases and retrieve a relevant one) might not usefully describe the same situation given a different case library.

Because the disjoint model separates the task of describing the current situation from the task of searching for relevant prior situations, it leaves open a loop that, if closed, would provide a mechanism for learning.

1.3. Retrieval and learning

A case retriever is faced with two learning tasks: Acquiring new descriptive features over the course of many experiences, and refining a description over the course of a single experience.

Over the course of reasoning about many situations, a case-based reasoner can learn not only by adding new cases to its memory, but also by adding new descriptive features

to its representational vocabulary. The features that should be learned and added are those likely to lead to the appropriate retrieval of useful cases. This article presents a number of criteria that a system designer or learning system can use to identify such features. Many of the features depend upon the existing contents of memory, and, therefore, depend upon an integration of the feature acquisition mechanism with the case library itself.

Over the course of reasoning about a single situation, a case-based reasoner is attempting to build a description of the current situation using the features currently in its representational vocabulary. Since many of the features in a descriptive vocabulary are abstract, it is often costly to determine whether or not a feature applies to a given situation. Consequently, a case-based reasoner's inference mechanism must be selective in its use of descriptive features. It is important that the reasoner direct its inference toward determining the applicability of those descriptive features most likely to narrow its search to a small set of useful cases. This utility criterion is dependent upon the current contents of the case library, and this article presents an algorithmic mechanism whereby the system can use the contents of memory to gain some control over its descriptive inference mechanism.

1.4. The rest of this article

The main goal of this work is to focus attention on the tradeoff between the cost of identifying a feature and the benefit associated with that feature: both on the static role that tradeoff plays in the design of representation vocabularies, and on the dynamic role it plays in a system's ability to balance the computational costs of feature extraction against memory search. Although the argument applies to the general problem of case retrieval, it is stated here in the context of a planning task.

In the next section, I introduce the planning task against which the argument is framed. In section 3, I discuss the types of features from which a representation vocabulary can be built. I introduce the idea of **labels**—selected features whose presence or absence plays a primary role in determining the applicability of a prior case to a current situation. The idea of labels is further developed in section 4, which defines criteria for selecting labels. Section 5 presents the main argument of the article: that the notion of labeling and the criteria developed in section 4 can be used to integrate the tasks of situation description and memory search, and that the integrated system can reason effectively about the inference cost versus the information content of features. Section 6 describes a computer program that implements the idea of explicitly reasoning about features acquisition cost, and the remainder of the article discusses the implications of this approach.

2. Critical planning situations

Although the main argument of this article applies to case-based reasoning in general, the specific examples herein are drawn from the task of recognizing and reasoning about the critical planning situations an agent may encounter:

- The agent may have **failed**. An action may not have had its intended effect, or an unexpected condition may block the satisfaction of one or more of the agent's goals.

- The agent may be entering a situation in which a failure is likely or imminent. The agent should be able to **predict** the failure.
- The agent may be entering a situation entailing an **opportunity** that it knows how to exploit.

While some failures are the result of circumstances that are genuinely beyond the control and predictive abilities of the agent, others result from planning errors that can be corrected if the agent understands the connection between the planning error and the failure. A widely accepted paradigm for dealing with failed plans (see, for example, Sussman, 1975; Hayes-Roth, 1983; Hammond, 1986; Birnbaum & Collins, 1988), is that an agent must:

- **Explain** the failure. Assign blame for the failure to some condition over which the agent could have had control. (Or, if no such condition can be found, identify the failure as an unforseeable, unavoidable one.)
- **Recover** from the failure. Plan some activity that will lead toward the original goals, taking into account the changed world resulting from the failure. Or, if achieving the original goal now looks too expensive, work on some other goal.
- **Repair** the plan that resulted in the failure. If the explanation assigns blame to some condition internal to the planner—for example, failure to look for a certain contraindicating condition before commencing a particular activity—modify the plan so that future uses of the same plan will not result in the same failure.

The case-based approach has been applied to the task of reasoning about plan failure, but it has not in particular been applied to the subtask of *building explanations* of failures. While case-based programs like CHEF (Hammond, 1989) CLAVIER (Barletta & Mark, 1988), and JULIA (Hinrichs, 1988; Kolodner, 1987) have dealt with the task of recovering from and repairing failures, they have not in particular used a case-based approach to generate the necessary explanations for those repairs and recoveries. SWALE (Kass et al., 1986), EXPLAINER (Leake, 1990), and ABE (Kass, 1990) deal with the task of generating explanations from prior cases, but the explanations are not oriented towards plan failure and repair.

2.1. Knowledge structures for plan criticalities

To use prior cases to explain plan failures requires a content theory of failure, recovery, and repair, and a representational theory of how similarities between instances of recurring failures can be detected. Such a theory is embodied in the PFXP knowledge structures described in Owens (1990). PFXPs package causal descriptions of plan failures with recognition criteria and repair and recovery strategies.

Critical planning situations can be defined in concrete, domain-specific terms. A system operating a chemical plant, for example, might know that the temperature and pressure in a particular reaction tank must be kept within certain limits, and have specific procedures for dealing with exceptions. On the other hand, characterizations of critical planning situations can also be quite general: one can mis-prioritize one's goals, choose inappropriate tasks, or allocate resources inappropriately. PFXPs characterize plan failures abstractly. This

is significant to the task of case retrieval, because abstract features are, in general, more costly to infer from initial situation descriptions than concrete ones.

The planners of Sussman (1975) and Sacerdoti (1975) and their descendants included operational definitions of generalized critical planning situations as part of the programs' control structure. Specific code, for example, detects and resolves situations in which the steps taken to establish a precondition for one goal violate a protected precondition for another goal or task. Wilensky (1983) presented a taxonomy of critical planning situations related to goal interactions. Hammond's (1989) planner moves the knowledge about critical planning situations out of the program's control structure and into a memory, using knowledge structures called *Planning TOPs*. In Owens (1990), I present a typology of critical planning situations derived from a study of a large library of common advice-giving proverbs like *Too many cooks spoil the broth*.[1]

The PFXP is thematically related to a planning TOP in that it captures a recurring planning situation; it is structurally related to the **explanation pattern** or XP discussed in Schank (1986) in that it packages a reusable explanation, and it extends both ideas by packaging recognition, recovery, and repair strategies with the description of the failure.

A PFXP's abstract characterization of a class of critical planning situations can be useful to a planner because an abstract characterization can define a functional equivalence class, significant to repair or recovery. All *too many cooks* situations, for example, share certain causal properties, such as involving a plan that failed in some way because several agents were working on a common task. It is likely that a common set of recovery strategies will be successful in all *too many cooks* situations, e.g., starting the plan over with fewer agents, or allocating more resources to supervision and coordination.

Just as the chemical plant controller can have a knowledge structure packaging recognition criteria and appropriate actions for an *overpressure in tank 17* situation, so can a planner have a knowledge structure that packages recognition criteria and appropriate actions for a *too many cooks* situation. The difference has to do with the abstractness of the identifying features and the specificity of the repair and recovery procedures. But the difference for case retrieval is significant. While a controller for a chemical plant is likely to have a pressure transducer that constantly reports the pressure in tank 17, the analogous feature detector for the *too many cooks* situation is likely to be a set of inference rules that are costly to execute, and can therefore not be checked constantly. Accordingly, a case retriever must be selective in its choice and use of abstract knowledge structures.

3. Retrieval and labels

A major problem for retrieval is that, if a case retriever has a large vocabulary of abstract features it can use to describe situations, its inference mechanism cannot be simply "examine all the features against the current situation and report which ones apply." As observed in Schank et al., (1986), "Real instances have indefinite numbers of features, some explicit, some inferred and many ignored." There are virtually an infinite number of facts about a situation, many of which would be irrelevant, or expensive to compute, or both.

The problem with recognizing an abstract characterization like *too many cooks*, as opposed to a concrete characterizaton like *overpressure in tank 17*, is that the features that

characterize a *too many cooks* situation are themselves abstract, difficult to notice, and costly to infer. The knowledge structure characterizes situations in which MULTIPLE AGENTS are working on a COMPLEX, MULTI-STEP PLAN. Determining whether or not MULTIPLE AGENTS and COMPLEX, MULTI-STEP PLAN characterizes a new situation is inferentially costly.

In spite of the abstractness of these features and the concomitant costs of determining their applicability to a new situation, they are powerful cues to discriminate between *too many cooks* and other potentially applicable knowledge structures in memory. The system cannot do much about the inferential cost of determining whether or not these features apply to newly encountered situations, but it can do something about the cost of determining whether or not these features apply to cases. It can attach these features as **labels** to the cases, in effect caching the results of inference. From that point on, determining whether or not MULTIPLE AGENTS applies to a case in memory is simply a matter of checking the feature against the case's list of labels.

3.1. The role of labels

Labels can be concrete or abstract—they can be any features that describes situations and cases. The fundamental choice in using labels is how complex and abstract the labels will be, and how much inference a system will perform before it begins to search memory. Two widely used approaches described below are **Classical indexing**, which performs a great deal of inference to extract complex, thematic labels that are then used as indices into a database of cases, and **Retrieval as matching**, which uses larger numbers of shallower, less costly features as labels. Both of these approaches, however, suffer from the fundamental flaw of the disjoint model of case retrieval: they do not take advantage of knowledge about the population of cases in memory to determine how feature extraction and inference should proceed, and they ignore feedback that can be used for learning.

3.2. Labels and classical indexing

Classical indexing views a case library as a database, indexed according to certain key features of the cases. The system retrieves a relevant knowledge structure by first characterizing the current situation in a functionally relevant way, and then using that characterization as a probe into a memory of knowledge structures.

For example, a planner might care about situations in which a goal is blocked because of an inadequate supply of a resource.

In the classical indexing approach:

- A symbolic construct (for example BLOCKED GOAL—INSUFFICIENT RESOURCE)[2] is assigned as a label for situations of this type.
- The system has some mechanism for detecting a BLOCKED GOAL—INSUFFICIENT RESOURCE situation, and produces the appropriate symbolic construct accordingly.
- The retrieval mechanism allows the system to cluster together all knowledge structures indexed by this symbolic construct so that they are made available when memory is searched using this index.

In general, this approach performs relatively costly inference to extract highly thematic features from new situations, and it uses those features as the indices around which memory is organized. Examples of the use of indices to retrieve compound knowledge structures is schema-based or case-based reasoning are widespread; see, for example, Schank and Abelson (1977), Charniak (1981), Kolodner (1984), Hammond (1989), or Ashley (1988).

3.3. Labels and matching

The second broad category of approaches to retrieval, **Retrieval as matching**, attempts to use simpler features as labels; features that can be extracted from situations with less costly inference. The approach uses larger numbers of these simpler features to determine the applicability of an old case to a new situation. Labeling features, under this approach, tend to be either surface features, or features cheaply inferred from surface features via a preprocessing pass.

Systems embodying this approach do not extract deep thematic descriptions of the current situation prior to searching for relevant knowledge structures. One instance of this approach is the Memory-Based Reasoning approach of Stanfill and Waltz (1986). This approach dynamically weights the importance of each feature based upon how unusual it is vis-a-vis the contents of memory. Thagard and Holyoak (1989) argue for a connectionist mechanism for adjusting the weights of features in determining a match via constraint relaxation.

3.4. Intermediate approaches

There is an intermediate approach taken by, among others, the PROTOS system of Bareiss (1989) that calls for features more abstract than surface features, but not as overarching or as thematic as the highly abstract features used in classical indexing. Other approaches, like CABARET (Rissland & Skalak, 1989), CASCADE (Simoudis, 1990), CASEY (Koton, 1988), and CLAVIER (Mark, 1989) deal more explicitly with the combined use of cheaply detected surface features and more expensive thematic features in case retrieval. Generally, these systems embody a fixed distinction between cheap and expensive features, with an explicit use of two passes at retrieval, one based on cheap features as indices, and the other based on more expensive thematic comparison of the candidate cases to the current situation. One goal of the work described herein is to integrate these two types of retrieval and comparison more closely.

3.5. Labels and detectors

Both the classical indexing and the retrieval as matching approaches represent a commitment to at least some inference process to extract features from the world—shallow inference in the case of retrieval as matching, and deep inference in the case of indexed retrieval. Even systems that seem to match new situations against knowledge structures in memory on the basis of "all" of the features of the current situation must, in fact, do some kind

of feature extraction, since "all" of the features of any real situation is an unlimited set. Kolodner and Thau (1988), for example, contrast their system PARADYME (also Kolodner, 1989) with some other case memories, saying that indices are "not used to organize memory's heirarchies, nor to direct traversal at retrieval time." In other words, PARADYME's indices do not form the basis of static memory organization as they do in, for example, CYRUS (Kolodner, 1984). Even though such systems do not use indices in the classical sense, they still must face the question of what abstract properties of situations to represent, which is the same question one must answer in choosing labels.

Computationally, labels require **detectors**—if cases in memory are labeled with some feature, then there must exist an inference mechanism to detect the presence of that feature in new situations. A detector is a piece of code that corresponds to a given label, and that has access both to the planner's inference capacity and to the system's sensors. The system gains its knowledge about the world by selectively running its detectors. Running a detector is equivalent to asking a question of the inference mechanism; the inference mechanism, in turn, has access to the system's sensors and working memory, which it uses to return an answer.[3]

Unfortunately for the two models of retrieval discussed above, the inferential cost of detectors can range from virtually free to extremely costly. And the value of the answer returned by a detector can be high or low, depending upon the makeup of cases in memory and the degree to which the feature detected discriminates among those cases. Neither of the retrieval models can take advantage of this cost and benefit information; the model I present in section 5 below addresses this concern.

4. Criteria for labels

Some labels are more useful than others. Labels can be **syntactically** useful, which means that they discriminate among the cases in a particular system's memory, and they can be **semantically** useful, which means that similarly labeled cases are in fact similar from the system's functional point of view. Section 5 below, and the implementation work in section 6 below, discuss how syntactic utility can be measured and taken into account at case-retrieval time. This section discusses the criteria for semantic utility of labels in a planning system.

The problem of choosing labels is comparable with the task of organizing cases or abstractions into a set of meaningful categories, which is one of the general problems of machine learning. AI has used statistical approaches to category formation (e.g., Stepp & Michalski, 1986; Quinlan, 1986; Fisher, 1987; Cheesman et al., 1988) as well as the more knowledge-intensive methods of Explanation-Based learning (e.g., DeJong & Mooney, 1986); Mitchell et al., 1986). A strong argument for a functional approach to category formation is presented in Schank et al., (1986).

The work described herein is not about **how** categories ought to be learned, but about **what kinds** of categories a specific type of system needs. A theory of how categories are formed or how labels are learned is outside the scope of this work. Instead, this is a theory of what constitutes a useful label and how those labels can be used for indexing and retrieval of abstract knowledge structures.

4.1. Functional criteria

It has long been observed (cf. Minsky, 1961) that functional terms define the categories a system should care about. For an example of functional relevance in the task of plan repair, consider the label SPREAD-TOO-THIN, that applies to situations corresponding to the following proverbs:

- *If you run after two hares, you will catch neither.*
- *Drive not too many ploughs at once; some will make foul work.*
- *He that hath many irons in the fire, some of them will cool.*

All situations labeled with SPREAD-TOO-THIN have something in common—an aspect of the agent's capacity is spread too thinly across a number of tasks. The aspect might be a physical resource, it might be a physical capacity of the agent like the amount of weight it can carry, or it might be the agent's capacity to monitor tasks. Although the applicable recovery strategies might differ depending upon what resource is spread too thinly, the strategies are likely to involve behavior such as **offloading subtasks, eliminating redundant steps,** or **increasing the effectiveness of the critical resource/capacity.** Because of this common functionality, SPREAD-TOO-THIN is a good feature around which to organize a portion of memory and, consequently, a potentially good label for cases.

4.2. Types of functional relevance

A case-based or abstraction-based reasoning system seeks to retrieve cases or abstractions relevant to the current situation in which the system finds itself. But the specifics of functional relevance depend upon the task in which the system is engaged:

- For case-based planners like CHEF (Hammond, 1989), CLAVIER (Barletta & Mark, 1988), PLEXUS (Alterman, 1986), or JULIA (Hinrichs, 1988; Kolodner, 1987), a case is relevant to the degree to which the plan constructed from that case satisfies the system's current goals.
- For a system whose goal is to build explanations, like SWALE (Kass et al., 1986) or CASEY (Koton, 1988), a case is relevant to the degree to which the explanation derived from it subsumes the facts of the current situation and is plausible. (See Pearl (1988) for a discussion of explanation plausibility.)
- For systems that explicitly critique, evaluate, and interpret cases (e.g., a legal reasoner like HYPO (Ashley, 1988)), the goal of retrieval is to find a case that is demonstrably similar to the current case along certain dimensions derived from a deep, domain-intensive analysis of the current situation and whose outcome supports the system's desired outcome for the current situation.
- For a categorization system like PROTOS (Bareiss, 1989), the goal is to find cases sharing certain predictive features with the currently examined case, those features being either designed into the system or learned by causal or explanatory reference to a body of domain knowledge.

For labeling knowledge structures relevant to critical planning situations, four types of functional relevance are particularly significant:

Failure-related: Failure-related labels capture some aspect of the central causality represented in the knowledge structure. A failure-related label clusters together knowledge structures in which a common element plays a role in the causal description of the failure. SPREAD-TOO-THIN is one example of a failure-related index. For another example, the *Too many cooks* knowledge structure can be labeled with MULTIPLE-AGENTS, which also labels a number of other knowledge structures that deal with both desirable and undesirable aspects of multiple-agent situations. It can also be labeled with COMPLEX TASK, which differentiates between situations in which multiple agents are desirable and those in which multiple agents create a problem. It can be labeled with domain-specific labels like MOVING, which identifies knowledge structures dealing with problems that occur when carrying objects.

Symptomatic: Symptomatic labels relate an observable condition to the situation characterized by the knowledge structure. For example, RESOURCE CONTENTION between agents is symptomatic of a *too many cooks* situation, even though at the level of representation of the knowledge structure, no explicit causal connection is made between the resource contention and the failure. (Imagine that you are working on a task with some helpers, and every time you reach for a tool someone else has it.)

If the causal description contained within the knowledge structure were more detailed, then RESOURCE CONTENTION would be a failure-related label, and could have been acquired by the same explanatory mechanism. But because the connection between resource connection and the *Too many cooks* failure is beyond the representational power of the knowledge structure, the label can be learned only by statistical correlation. Symptomatic labels provide a place for the system to list observable features suspected to be causally implicated in the failure, but for which no causal connection has yet been inferred. A system that learns would transform symptomatic labels into failure-related lables by deepening the causal structure represented within a knowledge structure.

Recovery-related: Recovery-related labels characterize an aspect of the recovery strategy contained within the knowledge structure. For example, there is a functionally useful difference between recoveries that require additional resources and those that do not. If REPAIR-REQUIRES-RESOURCES labels the former, then the system can, in a resource constrained situation, search for cheap recovery strategies that are not so labeled. Note that recovery-related labels can be used to retrieve knowledge structures that relate to the current status of the planning system, rather than the current situation.

Direct: Labels that exhibit direct functional relevance characterize the central causality in the failure, without being a subset of the symbolic constructs used in the representation of that causality. These labels define categories in which a certain overarching pattern of causality occurs, in cases where that pattern is not captured by any single symbolic construct within the representation. For example, if there are a number of knowledge structures that deal with situations in which a plan failed due to a low level of a resource, they can be clustered using a label—for example INSUFFICIENT RESOURCE. The label does not

appear within the description of any of the PFXPS to which it applies; its semantics derive from the common set of repair and recovery strategies that it labels.

While these types of relevance guide the knowledge-intensive task of acquiring new features to add to the labeling vocabulary, a simpler, shorter-term learning task is to obtain and exploit knowledge about the inferential costs and syntactic utility of features in order to control inference. The following section describes this task and a computational approach to it.

5. Balancing inference and search

The above arguments claim that the choice of labels should be based upon functional criteria, but there is an equally important and often competing requirement: that labels be based upon features that are **observable**. A feature that clusters together failure situations for which there is a common mode of repair is useful, but only to the extent that the system can encounter a new situation and determine whether or not that feature applies to it. If there is a detector for each label, some detectors will be more costly to run than others.

The more useful indices are characterized by relatively low complexity in extraction and therefore lower cost in processing resources. Features present in the input (i.e., low-level sensor readings), for example, are trivially easy to extract. So are error messages from the system's actuators, like ARM WEIGHT LIMIT EXCEEDED. On the other hand, more abstract labels associated with sensing, ranging from the simple "What's in front of me?" that merely requires the process requiring this information to wait in line to use the sensing apparatus, to the more costly "Go look around the corner and see what's there," which requires planning and carrying out a sequence of actions. Information may be computationally expensive if it requires a great deal of inference; for example, "Does this plan describe a task with independently executable subtasks?"

Unfortunately, the problem of feature extraction will not go away. In any given situation, there is always more information that can be gathered—more features that can be learned about the situation—by spending additional time sensing or making inferences. No description is ever complete, so any control structure that suggests first developing a description of the situation and then using that description either to extract abstract features or to select the existing features of the description, faces a difficult question: When is the description sufficiently complete that the system should stop enhancing the description and start using that description to extract indices?

5.1. Active and passive memory

The answer to that question is that the entire model of retrieval on which it is predicated needs to be revised. There are two basic models of lookup applicable to the task of selecting a relevant knowledge structure. Under the classical indexing model, the view that has been implicitly described thus far, memory is passive. It is the responsibility of the problem-solving portion of the system to come up with a description of the knowledge structure

that it wants—presumably to come up with a set of labels. On the basis of this description, memory can select a knowledge structure that best matches and can return it.

Accordingly, the features output by the describer must be functionally grounded in the conditions the system needs to recognize and the actions the system needs to perform. The applicability of a case in memory to the current situation is often predicated upon some kind of abstract, thematic similarity (e.g., for a planner, *''Both the current situation and the prior case involve a finite resource that is spread across too many tasks.''*).

The task of description, as it stands apart from search, is underconstrained because there are many possible abstract descriptions that apply to any one situation, only some of which are likely to be useful in retrieving relevant prior cases. The utility of a description depends not only on its functional grounding in a system's recognition and action capabilities, but also upon its relationship to the cases already in memory and the features used to describe those cases. A criterion for descriptions, ignored in the disjoint model, is that the description enable the memory to discriminate reliably and in a functionally relevant way, among the competing candidates for "best match" to the current situation. The process of description needs to know what is contained in memory and how it is described.

The problem with passive memory is that the process that extracts labels must know a great deal about what is in memory and how it is organized. It must know, for example, how the experiences or abstract knowledge structures in memory are described so that the description of the current experience will indicate the similarities and differences between the current experience and the experiences in memory. In effect, passive memory requires that the system fully understand a situation before it extracts labels from it. But since the entire purpose of extracting labels is to retrieve knowledge structures that will help the system understand the current situation, it makes little sense to require understanding as a prerequisite to label extraction. Accordingly, passive memory is not an acceptable model.

Under the **active memory** model, on the other hand, the contents of memory have a more direct control over the process that extracts labels from the current situation. In one version of the active memory model, the abstractions are viewed as active agents, each competing with the others to characterize the current situation. Each abstraction demands the inferences it needs to make the mapping between itself and the particular facts associated with the current situation. Control of processing resources, under this model, constitutes resolving the competition between abstractions to have their questions answered, and deciding how many of those questions to answer. The questions, in this case, refer to requests to run a particular detector.

5.2. Incremental retrieval

In active memory, the system performs incremental retrieval. When it begins processing a new situation, some information is available essentially for free—the kind of low-cost features described above. Using these features, analogous to presenting symptoms, as labels, the system may retrieve a large number of knowledge structures that match on the basis of those labels.

Given this pool of candidate knowledge structures, the system can examine them for labels with high discriminating power—that separate the pool into subpools. The more evenly balanced the sizes of the subpools, the higher the discriminating power of that label.

Labels with high discriminating power yield the highest benefit. Consider a pool of candidate knowledge structures, and imagine some label that happens to be present on all of those knowledge structures, or another that appears on none of them. Neither of these labels have great information content, because running the detector for these labels and determining whether or not either characterizes the current situation will not help to refine the choice of knowledge structures. If, on the other hand, a label is present on half of the candidate knowledge structures, then it is to the system's advantage to decide whether or not that label characterizes the current situation, because the label has a high information content. It is this benefit that must be balanced against the cost of feature acquisition discussed above.

As the system runs more and more of its detectors—as it infers more and more properties of the current situation—the number of knowledge structures in memory that match on the basis of those properties decreases. It is not necessary, however, to drive the number of matches all the way to 1. If, for example, the system's goal is to recover from a failed plan, and if the known characteristics of the current situation specify 50 of the system's knowledge structures equally well, and if those 50 structures all suggest the same plan repair, then there is no reason to look for more information to help discriminate between those cases.[4] **Recovery-related** labels play an important role in retrieval.

On the other hand, often the same set of features label cases wth different suggested courses of action. The MULTIPLE AGENTS label, for example, applies to the *many hands make light work* class of situations in which a goal was accomplished more quickly due to the increased number of agents, as well as to the *Too many cooks spoil the broth* class of situations, in which the multiple agents resulted in a worse outcome. The system needs a label that is present on one PFXP and absent on the other, that allows it to discriminate between the cases. The feature COMPLEX HIERARCHICAL PLAN is such a discriminator; it applies to the situations in which too many agents get in each other's way, but not to the situations in which multiple agents improve the outcome.

6. ANON

The ANON program provides a mechanism for integrating feature extraction and memory search, and for explicitly reasoning about the costs and benefits of individual features. ANON's task is to maintain a library of abstract knowledge structures and match those knowledge structures against descriptions of planning situations. My goals in writing ANON were twofold. First, the program was to be a vehicle for experimenting with representational constructs, labeling vocabularies and retrieval mechanisms. Secondly, as argued in Owens (1989), it was to demonstrate a computationally feasible implementation of an active memory. In the ANON program, I focus on the concept of incremental retrieval and the tight integration of retrieval with other processing. ANON represents a framework in which to implement decision theoretic and other strategies relating to the choice of where to direct inference.

ANON exists to demonstrate the viability of the design in the context of a larger planning system. It does not stand alone, but rather makes assumptions about the structure and input/output behavior of the rest of the planning system.

In particular, the program assumes that there exist detectors of the kind described above. The program further assumes that it knows something about the relative cost of running

the various detectors. In its current implementation, ANON also assumes that detectors are independent and that they do not share partial results. Running one detector does not change the cost of running another.

Under ANON's model, memory has some control over the inference and sensing processes. The presence or absence of abstract features in the environment is determined in response to requests from memory, and those requests are based upon the need for information to discriminate among the available knowledge structures.

6.1. Top-down and bottom-up search

Allowing memory to gather information by requesting features suggests a very top-down view of search: information from the environment would only be supplied upon request, and information would only be requested when needed to discriminate between competing knowledge structures.

Such an extreme top-down view of search and retrieval is typically implemented by placing all the system's knowledge structures in a discrimination net partitioned according to the system's labeling terms, for example, as in Feigenbaum (1961). The system descends the discrimination net to find a knowledge structure that matches the current situation on the basis of all the features extracted in the course of descending the net.

Unfortunately, the discrimination net method exhibits certain problems:

- A system using a discriminaiton net must ask its questions in a predetermined order and cannot ask the situationally cheaper questions first.
- A system using a discrimination net cannot exploit free information—information generated perhaps as a side effect of other processing that the system must do regardless of whether or not it needs the information. Some of a planning system's detectors might be run by other than the memory search process—the answers they return should be usable immediately by the memory search process even if it has not yet descended to that particular branching point in the discrimination net.
- A system using a discrimination net would require a complex control structure to deal with having one of its questions answered "I don't know."

It is possible to take information utility into account when building a decision net. Quinlan's (1986) ID3 program provides one example of how this can be done; Fisher's (1987) COBWEB system and Cheeseman's (1988) AutoClass systems deal with the issues of learning the right features from which to construct clusters of similar knowledge structures. Static clusters and discrimination nets, however, do not support the kind of dynamic balancing of information cost against information utility at retrieval time as described above. PARADYME (Kolodner, 1989), with its use of preference heuristics, takes information utility into account more dynamically, during the retrieval process, but it does not appear to provide a mechanism for merging information about feature acquisition cost with information about the discriminating power of features. Although PARADYME is also a fine-grained parallel retrieval scheme, it differs from ANON in that while PARADYME takes as input a description of the current situation and searches memory for a matching case, ANON extracts a description incrementally

as memory search proceeds. When a relevant knowledge structure is found, ANON's description of the current situation is only just exactly detailed enough to discriminate among the known cases in ANON's memory.

Another retrieval mechanism from which ANON differs is DMAP (Martin, 1990). DMAP searches memory by passing markers through uses a complex, richly indexed hierarchical knowledge representation. ANON, on the other hand, attempts to gain maximum utility from flat indexing relationships.

The other extreme is the completely bottom-up search, in which the system gathers as much information as it can about the current situation, and then uses that information as a template to match against knowledge structures in memory in a single pass. Using a bottom-up approach effectively says that the system should run *all* of its detectors first to yield a vector of features, and then use this vector of features as a retrieval probe. If the system has a rich and extensive labeling vocabulary, the cost of running a large number of detectors is likely to be prohibitive. Furthermore, only a small number of detectors are likely to correspond to labeling terms with good discriminating power in the current situation. This approach precludes the efficiencies of incremental retrieval.

6.2. Incremental retireval in ANON

The ANON program uses an incremental approach combining some of the features of top-down and bottom-up search. This hybrid system tries to be intelligent about which of the features in its labeling vocabulary it will detect next, at each point in the retrieval process.

The code runs on the Connection Machine fine-grained SIMD parallel computer (Hillis, 1985). Although parallelism is not a particularly important theoretical aspect of the model, fine-grained parallelism provides an effective, natural mechanism for the dynamic calculation of feature utility, which is at the heart of the program. The dynamic calculation of feature utility is built on top of a basic retrieval mechanism similar to that used in the document retrieval system described by Stanfill and Kahle (1986).

6.3. Memory organization in ANON

ANON's memory contains approximately 1000 knowledge structures, each corresponding to a very sketchy representation of a PFXP or critical planning situation. Each is labeled via an unstructured list of labels chosen from the system's labeling vocabulary of about 80 labeling terms. The indexing relationships are represented as doubly linked associations between label terms and knowledge structures, with one processor used for each two-way link.

ANON's memory does not explicitly maintain a hierarchy of categories—categories and hierarchy are implicit in the assignment of labels to knowledge structures.

Given any label term, the system can, essentially in constant time,[5] determine which of the knowledge structures in memory are labeled with that term. Similarly, given any knowledge structure, the system can determine which label terms are used to label it. As will be shown below, these operations form the heart of a flexible, incremental retrieval system.

What has been described thus far is a flat database with boolean, atomic labels (or numerically weighted labels, depending upon which type of detector the system is built around). Beyond this, ANON offers a mechanism for letting the content of memory inform the feature extraction process.

The goal of ANON's retrieval mechanism is to focus the system's feature extraction capabilities on the features that are likely to be most useful as retrieval labels, relative to the cost of acquiring information about those features. Accordingly, measuring the utility of a piece of labeling information is at the heart of ANON's incremental retrieval approach.

6.4. Information utility

6.4.1. Syntactic measurement of benefit

Once the system determines that a certain set of labels characterize the current situation, then it can retrieve all the knowledge structures in its memory associated with those labels. Presumably, some of the knowledge structures in this "candidate" pool also have other labels, each of which may or may not apply to the current situation. The question facing the system in which of these other labels it should test to determine applicability to the current situation.

Figure 1 demonstrates the simplest, most syntactic way to determine which labels the system should care about. This figure shows the state of the system after determining that three label terms, L1, L2, and L3, characterize the current situation (the last by virtue of its absence). A first pass at retrieval using these labels identifies eight knowledge structures as potentially relevant on the basis of matching on these three terms. The question is which of the remaining label terms the system should try to detect next.

L4 is uninteresting because it characterizes all the candidates. Spending inferential and sensing effort on determining the applicability of L4 to the current situation is useless, because this knowledge will not enable the system to discriminate between the candidates. The same is true of L5, because it characterizes none of the candidates. L6 has limited

	L1	L2	L3	L4	L5	L6	L7	L8
Input	•	•	o	-	-	-	-	-
PFXP217	•	•	o	•	o	•	•	o
PFXP362	•	•	o	•	o	•	o	o
PFXP513	•	•	o	•	o	o	•	•
PFXP684	•	•	o	•	o	•	o	•
PFXP801	•	•	o	•	o	•	•	o
PFXP877	•	•	o	•	o	•	o	o
PFXP935	•	•	o	•	o	•	•	•
PFXP982	•	•	o	•	o	•	o	•

Figure 1. Syntactic measurement of feature utility.

discriminating power in that it separates PFXP513 from the others. L7 and L8 have the greatest discriminating power: each splits the pool of candidates into two equal-sized groups. Accordingly, the system should next turn its inferential effort to detecting the cheaper of the features corresponding to these two lables.

6.4.2. *Deeper measures of utility*

The same basic implementation also allows the system to determine feature salience on a deeper, more functionally meaningful basis. Instead of looking for labels that split the set of candidate knowledge structures in half, the ANON system can look for labels that correlate with a property the system cares about for some functional reason. For example, if the system is operating in a resource constrained situation, it may have narrowed its search to a small number of knowledge structures. Some of these knowledge structures are likely to include recovery strategies that consume additional resources, while others do not and are less expensive. Because this distinction is functionally relevant, it can be reflected using a label like RECOVERY-REQUIRES-RESOURCES.

Unfortunately, the label RECOVERY-REQUIRES-RESOURCES is expensive to detect. Accordingly, the system is looking for labels that are cheaper to detect, but that are highly correlated with this feature *in the context of the set of candidate knowledge structures retrieved thus far.*

6.4.3. *Cost of features*

A utility measurement implies balancing the benefit of any particular piece of information against the cost of acquiring that piece of information. Because ANON, as implemented, does not have detectors for each of its labels, it cannot estimate the cost of running those detectors. But even in a fully developed system that includes detectors, it is not clear how well the system can predict the cost of running its detectors or of performing any particular piece of inference.

Two approaches to estimating cost that are possible in the content of ANON's retrieval mechanism are **estimation by feature class** and **caching**. Features can be marked as belonging to a particular class—for example, *low-level sensory features* or *error messages from the effectors* at the cheap end and *unconstrained inference* or *features requiring active sensing* at the costly end. A numerical estimate of the predicted cost of acquiring features of that class can be associated with each class. Alternately, the system can store, for each label, historical information about the cost of determining whether or not that label applies.

The former approach requires categorization of features by the system designer when the system's initial vocabulary is defined and does not specify how the system can assign new features to the appropriate cost class as they are learned. The latter approach allows the system to learn the average complexity of acquiring features as it proceeds. Neither approach guarantees accurately predicting the cost of running a given detector in a given situation, rather these approaches are of heuristic value. The point of ANON, however, is not to suggest any particular mechanism for learning the cost of feature extraction, but rather to provide a computational framework in which computational and other costs of feature acquisition can be seamlessly integrated into the search process.

6.5. ANON's basic mechanisms

The feature utility determination and retrieval are accomplished using three basic operations:

6.5.1. Filter

Given a labeling term or a set of labeling terms, the system identifies the knowledge structures associated with that term. ANON uses boolean features and ranks selections matching based upon the number of features that match. The same approach is trivally extensible to employ numerical weights or feature salience metrics.

Filtering uses a scan/reduce model of computation. In addition to the processors representing the links between labeling terms and knowledge structures, there is a processor for each labeling term and one for each knowledge structure. When all the labeling terms to be selected during a given iteration of the retrieval process are selected, each labeling term broadcasts information to all processors representing links between itself and a knowledge structure. Each of those link processors then forwards information to the knowledge structure to which it points, so that each processor corresponding to a knowledge structure now has information about which of its labels were selected. The system then uses a parallel sort to rank the knowledge structures according to how closely their labeling terms match the labeling tems selected for the current iteration of the retrieval process.

To use numeric weights rather than boolean values, the weights are applied to the link processors.

6.5.2. Typify

Once the system identifies a set of candidate knowledge structures, it can identify labeling terms that are highly representative of this set. Since these might not be the same labeling terms that were used for retrieval, these terms might be useful for learning new clusters and new labeling relationships. This process is easily accomplished by running the retrieval algorithm in reverse—forwarding information from the selected knowledge structures to the links, and from the links to the labeling terms to which they point.

This operation is essentially inductive syntactic generalization. If ANON has used a complicated conjunction of features to retrieve a given set of knowledge structures, it is possible that there is a single, previously unexamined feature that is highly representative of this set—that discriminates this set from the non-selected knowledge structures. This process will find such a feature.

6.5.3. Discriminate

Using the same calculation used to calculate typicality, the system can look for labeling terms that subdivide a specified set of knowledge structures. The closer the partitions are to equal sizes, the better the discrimination score for a particular label term. Just as terms

that are highly typical of the retrieved set of knowledge structures are important, features that are present as labels for half of the current candidate knowledge strucutres and absent for the other half are also interesting—they have a high discriminating power and are likely to be the subject of a request from memory to the system's detectors.

This approach can also be used with weights to separate the knowledge structures into equally weighted subdivisions rather than subdivisions of equal numbers of candidates, so that labels will be found that separate, for example, a large number of low-weighted knowledge structures from a small number of highly weighted ones.

The same computational technique can be used to identify features that correlate with each other across a candidate pool. For example, if the system is interested in a functionally significant but costly feature, it can use the discriminate operation to identify other features that correlate with it, with the expectation that some combination of these other features may be cheaper to acquire.

6.6. Basic retrieval operation

Figure 2 outlines the operation of ANON's incremental retrieval mechanism.

The system starts with an initial rough characterization of the current situation, consisting of a small number of labeling terms believed to apply. This initial characterization consists of information that was provided for free—such as information that becomes available as a side effect of other processing that the system was doing. For example, the fact that MULTI-AGENT PLAN or RESOURCE-CONSTRAINED-SITUATION applies to the current situation might be known as a side effect of planning decisions that the system needed to make.

In the case when few features are already known, the system can have a stock set of features it cares about, analogous to the stock questions that a detective asks upon encountering a crime scene. Which questions are asked next depends upon the answers to the first few questions.

Based upon this initial set of labeling terms, the system retrieves a pool of knowledge structures labeled using these terms, using the **filter** operation described above. Figure 3 shows ANON's output of candidate knowledge structures, given an initial set of two labels.

1. Acquire initial set of labels believed to characterize current situation.
2. Retrieve knowledge structures labeled with these labels, using **filter**.
3. Using **discriminate**, identify labeling terms with high discriminating power.
4. Select cheapest-to-acquire labels from previous step; run detectors to determine whether or not any apply to current situation.
5. Add those known to apply, to initial set of labels from step 1.
6. Repeat from step 2 until no further discrimination possible.

Figure 2. ANON's basic retrieval mechanism.

```
  Selecting: #<label SCAL: "Scaling of a resource.">
  Selecting: #<label BO: "Bad outcome">

I found 10 proverbs that each included all 2 of those labels
  #<proverb P3: "A pig that has two owners is sure to
   die of hunger.">
  #<proverb P331: "He who has one clock always knows
     what time it is; he who has two is never sure">
  #<proverb P150: "Double charge will rive a cannon.">
  #<proverb P151: "Milk the cow but don't pull off the
     udder.">
  #<proverb P156: "Grasp all, lose all.">
  #<proverb P157: "If you run after two hares, you
     will catch neither.">
  #<proverb P158: "If you buy meat cheap, when it
     boils you will smell what you have saved.">
  #<proverb P161: "Too much spoileth.">
  #<proverb P164: "Two captains sink the ship.">
  #<proverb P329: "Too many cooks spoil the broth.">
```

Figure 3. ANON's initial search.

Next, using the **discriminate** operation described above, the system identifies the labeling terms with the highest discriminating power vis-a-vis the current pool of candidate knowledge structures. Using this measurement of discriminating power and any of the estimates of feature acquisition cost discussed above, identify the labels with the best cost/benefit ratio. These labeling terms go onto a list of terms, the presence or absence of which the system will try to detect in the current situation. Figure 4 shows ANON's identification of the best discriminators among the ten knowledge structures.

As the system gains more information (i.e., if the program were connected to an actual planner, as it runs its detectors and determines which of the suggested labeling terms apply to the current situation), it returns to the filtering step to further narrow the set of knowledge structures under consideration.

The system can continue to narrow the set of knowledge structures until either there is one knowledge structure remaining in the pool, or until there is no further discriminating information available. Figure 5 shows two further iterations of the filtering and discriminating steps, employing the user as a detector. If multiple knowledge structures remain in the pool and no further discriminating information available, some of the knowledge structures must be re-labeled.

```
I found 9 proverbs that each included all 2 of those labels.
Are any of the following 3 features present in the
  current problem?

     #<label ACT: "Action">
     #<label GOAL: "Managing goal priorities">
     #<label INSUF: "Insufficient resource">

Enter a feature or nil > insuf

  Selecting: #<label INSUF: "Insufficient resource">
```

Figure 4. Looking for discriminators.

```
I found 4 proverbs that each included all 3 of those labels

 #<proverb P3: "A pig that has two owners is sure to
   die of hunger.">
 #<proverb P156: "Grasp all, lose all.">
 #<proverb P157: "If you run after two hares, you
    will catch neither.">
 #<proverb P158: "If you buy meat cheap, when it
    boils you will smell what you have saved.">

Is the following feature present in the
current problem?

     #<label STT: "Capabilities spread too thin">

Enter a feature or nil > stt

  Selecting: #<label STT: "Capabilities spread too thin">

I found 2 proverbs that each included all 4 of those labels.

 #<proverb P156: "Grasp all, lose all.">
 #<proverb P157: "If you run after two hares, you
    will catch neither.">
```

Figure 5. Incremental refinement.

6.7. A unified control structure

Note that this control structure allows the system to operate in either a top-down or bottom-up search mode, depending upon the availability of information. If a great deal of information is available about the current situation (i.e., if the situation is already represented using the same labeling terms used to organize memory), then memory does not need to assist in formulating a description of the current situation. The system can use those labeling terms as retrieval cues for a single iteration through memory search, looking for the knowledge structure that shares the largest number of labeling terms with the description of the current situation. If ANON starts with a large number of labeling terms in the initial set, then one pass through the process will suffice to uniquely specify one knowledge structure, which will be returned.

If, on the other hand, very little is known about the current situation, then memory should play a large role in developing a description. In this context, discrimination net traversal is a useful mechanism, and in this context ANON can exhibit the same behavior as a discrimination net traverser. If the system starts with an empty list of initial label terms and with no information about relative costs of feature acquisition (i.e., if all costs are set to unity), then the system works very much like a self-balancing discrimination tree. The first request for information would be for some feature, the presence of absence of which would split the system's knowledge structures into two equal-sized pools. The second request would be for some feature that would split the selected pool again into two equal-sized pools.

Given some initial description of the current situation and some information about the relative costs of acquiring knowledge about the current situation, the system operates in a way that is neither strictly top-down nor bottom-up. ANON maximizes the use of the a priori sketchy description of the current situation and, beyond that, uses its knowledge about the contents and organization of memory to direct its inference mechanism efficiently.

7. Memory and idiosyncratic questions

It is important to recognize that ANON's control structure causes the system to seek a great deal of information about the current situation if it is matching against an area of memory densely populated with knowledge structures, and to seek much less information if it is matching against a more sparsely populated area of memory. As a result, changing the knowledge structures in the system's memory changes the questions that the system asks in the process of developing a description of the current situation. There is no need to posit a separate "situation describing" module that must be updated to reflect changes in the system's memory. Questions are asked as needed to discriminate among the knowledge structures in memory; changing memory changes the questions.

Consider again an explanation system faced with a task such as SWALE's (Kass et al., 1986) goal of explaining the mysterious death of a star race horse. Different individuals, each with idiosyncratic goals for an explanation, will build different explanations. An insurance examiner, for example, might be reminded of a valuable painting that mysteriously disappeared a year earlier in what turned out to be a fake burglary staged by the owner to collect the insurance money. A veterinarian might be reminded of the cow that died

mysteriously the previous week and begin investigating whether the medical causes were the same. A racing examiner might be reminded of other cases of one competitor trying to disable another and might suspect the owners of Swale's competitors. A gambler might be reminded of other odds-on favorites suddenly being disabled or otherwise removed from competition. Each of these individuals retrieves different remindings from memory because each has described Swale's death in different terms. Each description is equally correct, but each leads to a different path of explanatory reasoning.

It is unreasonable to assume that veterinarians, insurance adjusters, and gamblers have totally different processes for extracting descriptions from situations. Moreover, it is difficult to account for these differences in retrieval with a system that separates feature extraction from the rest of memory. The differences might be accounted for by a process that maps retrieval goals to predictive features, as discussed by Stepp and Michalski (1986) or Seifert (1988). This process could weight the importance of features depending upon how relevant they were to the current set of retrieval goals. But this approach leaves unanswered the question of how retrieval goals and predictive features are linked together.

A model like ANON explains the individuals' different explanations of Swale's death as due to the fact that abstract feature extraction is driven by the case libraries of each individual. The veterinarian has a large case library of animal diseases and consequently describes the event in terms of features that can discriminate among these cases. Likewise an insurance examiner discriminates among a second, different library of cases, and a gambler among a third. Each of these individuals extracts, from the story, the features necessary to discriminate among the cases in his own memory. Each individual can use the same kind of mechanism to extract abstract features from concrete descriptions of situations, but that mechanism is driven by a different case library in each case, and so results in a different set of features being extracted, a different case retrieved, and a different explanation generated.

8. Discussion

8.1. Retrieval and feature utility

An essential question in learning is *what to learn*: what category distinctions or descriptive features are worthwhile. This question arises more or less **statically** as "What terms should be included in the system's representation vocabulary?" and **dynamically** as "What features should the system try to extract now from this situation?" The main tradeoff in answering this question is to balance the expressive power of a descriptive feature against the computational and physical costs of determining whether or not that feature characterizes the current situation. This is the question that system designers address when they choose a reasonable set of primitives for a representation language, and it is the question that machine learning systems address when they make feature acquisition or categorization decisions.

The work described in this article represents an attempt to build a framework in which this tradeoff can be managed explicitly by a retrieval system. As such, it looks at two distinct means of describing the expressive power of a feature. **External** functional utility describes the utility of the feature to the system's external tasks: selecting actions, recovering from

plan failures, and detecting opportunities or threats. **Internal** functional utility describes the utility of the feature to the system's internal task: choosing the prior case or abstract knowledge structure most relevant to the system's current situation. This decomposition makes evident that the internal task, involving judgments of similarity or relevance, only makes sense as it is grounded in the external task—choosing a relevant prior experience is only useful to the degree that the prior experience tells the system what to do, or what further information to gather.

The ANON program itself does not address what the system does with a knowledge structure once it is determined to be applicable. It is not a complete planning system, and therefore it does not make any statements about the external task. It does, however, illustrate a retrieval mechanism that can efficiently and dynamically calculate the internal utility of a feature, and thereby does bring that utility measurement to bear on the question of what features to try to detect or extract next. It does so in a way that potentially allows external and internal utility to be expressed in comparable terms, and thereby to enter into the same calculation of which piece of detection or inference has the highest expected utility.

It achieves some of the same effect as algorithmic approaches to internal utility calculations such as those found in systems like ID3, AutoClass, and COBWEB, in that it creates a hierarchy of feature salience, such that the features with the highest discriminating power are pursued first. It avoids, however, the problem of committing in advance to a static ordering of questions or a static hierarchy of feature importance. ANON's algorithm allows the information contained in the current candidates at any point in the retrieval process to direct the subsequent retrieval. The features of the candidates known in the candidates but untested in the input can be examined to decide which ones discriminate among the candidates. The system's memory of knowledge structures can be examined *as a whole* to decide which features carry a high information content with regard to the input.

Exclusively top-down search ignores such information as the system might have at its disposal, for example, features acquired as a side effect of other processing that the system needs to perform. Similarly, completely bottom-up search causes the system to waste its time detecting the presence or absence of features that do not discriminate among the knowledge structures that are currently under consideration. The balancing approach taken by ANON is sensitive to and effectively uses available information, but it also allows the contents and organization of memory to drive the feature extraction process.

8.2. Criticalities and future directions

Since the point of this research is to enable systems to reason about the utility of feature acquisition, the major criticalities revolve around the system's measurements of utility. ANON shows how a system can measure discriminating power and use that measurement. The program itself, however, says nothing about external utility, beyond providing a mechanism whereby external and internal utility measurements can be combined to control inference. As a result, an important future direction is to say something about external utility and detections costs, beyond the obvious "They should be taken into account."

In particular, this suggests that content theories of repair and recovery should be extended to include detectability: the features to be included in a representation vocabulary are those that are functionally grounded not only in repair and recovery, but in detectability as well.

The issue here is to build a theory of feature detectors and their cost. Can anything be said about detectability at other than an ad-hoc level? Are there significant different classes of features, relative to detectability? Can detection cost be predicted or estimated reliably? This is an issue not just for the system designer in choosing an initial representation vocabulary, but for the system as well as it acquires new descriptive features.

A key future direction is to apply the representation methodology suggested by this work, to a specific, concrete, real-world planning domain. While the advice-giving proverbs that form the basis of ANON's knowledge structures provide a good cross section of stereotypical plan failures, they are not themselves grounded in a specific planning task. The goal is to identify stereotypical recurring failures and corresponding recovery and repair strategies, to build feature detectors for the descriptive features that discriminate among those failures, and to use an ANON-like memory as a repository of the resulting knowledge structures. When such a system is built it will enable us to determine whether or not the external and internal measurements of feature utility, as described above, can in fact be combined into a single mechanism for inference control.

8.3. Indexing: a compromise

A theory of indexing and retrieval is inherently one of compromise. Given infinite computational resources, a system could choose the best knowledge structure from its memory by trying each in turn, simulating the results obtained from applying the advice contained in that knowledge structure to the current situation, and, after trying all the knowledge structures in its library, choosing the best. Or, a system with infinite computational and sensory resources could perform such a detailed analysis of the current situation that the resulting description of the problem would effectively embody the solution, making complicated matching of the current situation against prior experiences superfluous.

But computational resources are not infinite. A system cannot develop a rich, fleshed-out description of the current problem merely as a precursor to searching memory, nor can it devote a lot of computational resources to estimating the goodness of each of a large number of knowledge structures as a potential solution to the current problem. Furthermore, computing power is not the only limit on the search for the ideal knowledge structure to match against the current situation; information about the world comes at a price, too. An eye or camera, for example, can only look in one direction at a time. Often, gathering even low-level sensory information about the world involves costs in resources and time. A theory of indexing and retrieval is a theory of heuristic management of bounded resources.

The compromises inherent in a theory of indexing and retrieval manifest themselves in several places, most particularly the modularity of the system and the choice of an indexing or labeling vocabulary.

8.3.1. Modularity

The premise underlying most models of retrieval is that a system can solve problems by

1. developing a sketchy description of the current situation,

2. comparing that description against the knowledge structures in its memory and selecting one or more knowledge structures that match it, and
3. adapting the retrieved knowledge structures to the problem at hand.

Within this framework, the tension lies in the balance between the sophistication of the "sketchy describer," the matcher, and the adapter. The more powerful the describer and the more extensive the memory, the less powerful the matcher and the adapter need to be. On the other hand, a strong, flexible, general adapter requires only a small memory and a weak matcher. In each case, the sophisticated module needs more access to the system's general inference mechanism. The modular approach remains a compromise because it cannot guarantee that the sketchy description captures anything essential about the current situation, nor can it guarantee that a knowledge structure matched on the basis of this sketchy description in fact contains useful advice.

Working around this modularity is an important part of the retrieval mechanisms described above. I presented a theory of indexing that tightly integrates the task of describing the current situation and the task of searching memory for relevant knowledge structures, both with each other and with the larger planning task. I developed ANON to demonstrate the ability of a memory, organized along functional lines, to drive a describer—an inference mechanism—and thus tailor descriptions to the needs of the memory search process. Such integration of feature extraction and retrieval will result in retrieved knowledge structures that improve the capability of the system to implement repairs and recovery strategies.

8.4. Learning and retrieval

ANON is not a learning algorithm. It is instead a framework for dealing with knowledge about the costs of feature acquisition that a system learns elsewhere.

The ANON program presents a novel approach to the mechanics of retrieval. As it is equally informed both by the contents of memory and the features of the current environment, it is neither top-down nor bottom-up in its approach to search. While it exploits parallelism, it does so in a way that allows the system to search memory incrementally—interleaving the operation of the searcher with that of the inference mechanism that gathers information about the current situation. This approach allows the system to continually balance the cost of acquiring information against the value of that information. Although other systems, most notably ID3 (Quinlan, 1986), have attended to this balance, ANON manages it dynamically during the retrieval process.

With this algorithm I argue for a re-modularization of memory-based systems. Memory should not be viewed as a passive information-retrieval system to be queried by an intelligent process elsewhere within a system. Memory implicitly contains much of the information necessary to direct inference and sensing; systems should make this information explicit and use it to direct incremental feature extraction and retrieval.

Acknowledgments

The work described herein was done while the author was at Yale University, and was funded in part by the Defense Advanced Research Projects Agency, monitored by the Office

of Naval Research under contract N00014-85-K-0108, and in part by the Air Force Office of Scientific Research under contracts F49620-88-C-0058, AFOSR-85-0343, and AFOSR-89-0100. The comments of the reviewers were particularly helpful in preparing the final version.

Notes

1. See Lehnert (1981), Dyer, (1982), and Schank (1986) for other examinations of the significance of proverbs to memory organization.
2. Of course any unique symbol, such as FOO or the number 23, could be used. Throughout this discussion, bear in mind that the lexical form of any labeling construct has no semantics whatsoever. All semantics come from the particular set of knowledge structures indexed by that construct, and from the behavior of its **detector** (see section 3.5 below) for that construct.
3. In a simple variant on the essentially boolean detector approach described here, detectors could return some numerical value that indicated *how strongly* the labeling term corresponding to the detector characterized the current situation, or *how confident* the detector is in its assessment.
4. This condition does not, however, mean that the system should lump all of these cases together into one equivalence class. There may be other circumstances other than this particular retrieval instance under which the differences between the cases would be significant.
5. Given, of course, that "constant time" on parallel hardware loses its theoretical significance. Once the number of indexing relationships exceeds the number of physical processors on the system, the complexity of the operations described below is essentially n log n in the number of indexing relationships.

References

Alterman, R. (1986). An adaptive planner. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 65–69), Philadelphia, PA. San Mateo, CA: AAAI.

Ashley, K.D. (1988). *Modelling legal argument: Reasoning with cases and hypotheticals*. Ph.D. thesis, Department of Computer and Information Science, Amherst, MA: University of Massachusetts.

Bareiss, R. (1989). *Exemplar-based knowledge acquisition*. Vol. 2 of *Perspectives in artificial intelligence*. San Diego, CA: Academic Press.

Barletta, R., & Mark, W. (1988). Explanation-based indexing of cases. In J. Kolodner (Ed.), *Proceedings of a Workshop on Case-Based Reasoning* (pp. 50–60), Palo Alto, CA. Defense Advanced Research Projects Agency, Morgan Kaufmann.

Birnbaum, L., & Collins, G. (1988). The transfer of experience across planning domains through the acquisition of abstract strategies. In J. Kolodner (Ed.), *Proceedings of a Workshop on Case-based Reasoning* (pp. 61–79), Palo Alto, CA. Defense Advanced Research Projects Agency, Morgan Kaufmann.

Charniak, E. (1981). A common representation for problem solving and language comprehension information. *Artificial Intelligence, 16*, 225–255.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Workshop on Machine Learning*. Morgan Kaufmann.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1(1)*, 145–176.

Dyer, M. (1982). In-depth understanding: A computer model of integrated processing for narrative comprehension (Technical Report 219). Department of Computer Science, Yale University, New Haven, CT.

Feigenbaum, E.A. (1961). The simulation of verbal learning behavior. In *Proceedings of the Western Joint Computer Conference*.

Fisher, D.H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Hammond, K. (1986). *Case-based planning: An integrated theory of planning, learning and memory* (Technical Report 488). Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT.

Hammond, K. (1989). *Case-based planning: Viewing planning as a memory task.* Vol. 1 of *Perspectives in artificial intelligence.* San Diego, CA: Academic Press.

Hayes-Roth, F. (1983). Using proofs and refutations to learn from experience. In R. Michaleski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga, pp. 221–240.

Hillis, W.D. (1985). *The connection machine.* Cambridge, MA: MIT Press.

Hinrichs, T. (1988). Towards an architecture for open world problem solving. In J. Kolodner (Ed.), *Proceedings of a Workshop on Case-Based Reasoning*, Palo Alto, CA. Defense Advanced Research Projects Agency, Morgan Kaufmann.

Kass, A.M., Leake, D.B., & Owens, C.C. (1986). SWALE: A program that explains. In *Explanation patterns: Understanding mechanically and creatively.* Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 232–254.

Kass, A. (1990). Developing creative hypothesis by adapting explanations (Technical Report 6). Institute for the Learning Sciences, Northwestern University.

Kolodner, J.L., & Thau, R. (1988). Design and implementation of a case memory (Technical Report RL88-1). Georgia Institute of Technology, School of Information and Computer Science, Atlanta, GA.

Kolodner, J. (1984). *Retrieval and organizational strategies in conceptual memory.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Kolodner, J. (1987). Extending problem solver capabilities through case-based inference. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 167–178), Los Altos, CA. University of California, Irvine, Morgan Kaufmann.

Kolodner, J. (1989). Selecting the best case for a case-based reasoner. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 155–162). Hillsdale, NJ: Lawrence Erlbaum Associates.

Koton, P. (1988). Reasoning about evidence in causal explanations. In J. Kolodner (Ed.), *Proceedings of a Workshop on Case-Based Reasoning* (pp. 260–270), Palo Alto, CA. Defense Advanced Research Projects Agency, Morgan Kaufmann, Inc.

Leake, D. (1990). *Evaluating Explanations* (Technical Report 769). Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT.

Lehnert, W. (1981). Plot units and narnative summarization. *Cognitive Science, 5,* 293–331.

Mark, W. (1989). Case-based reasoning for autoclave management. In *Proceedings of the Second Workshop on Case-Based Reasoning.* Morgan Kaufmann.

Martin, C.E. (1990). *Direct memory access parsing.* Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT.

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers, 49,* 8–30. Reprinted in E. Feigenbaum and J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill, 1963.

Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.), *The psychology of computer vision.* New York: McGraw-Hill, pp. 211–277.

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1(1),* 47–80.

Owens, C. (1989). Integrating feature extraction and memory search. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* Ann Arbor, MI. CSS.

Owens, C. (1990). *Indexing and retrieving abstract planning knowledge.* Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT. In preparation.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* San Mateo, CA: Morgan Kaufmann.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning, 1(1).*

Rissland, E., & Skalak, D. (1989). Combining case-based and rule-based reasoning: A heuristic approach. In *Proceedings of IJCAI-89,* Palo Alto, CA. Morgan Kaufmann.

Sacerdoti, E. (1975). A structure for plans and behavior (Technical Report 109). SRI Artificial Intelligence Center.

Schank, R.C., & Abelson, R. (1977). *Scripts, plans, goals and understanding.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R., Collins, G., & Hunter, L. (1986). Transcending inductive category formation in learning. *Behavioral and Brain Sciences, 9(4).*

Schank, R. (1986). *Explanation patterns: Understanding mechanically and creatively.* Hillsdale, NJ: Lawrence Earlbaum Associates.

Seifert, C. (1988). A retrieval model for case-based memory. In E. Rissland & J. King (Eds.), *Proceedings of a Case-Based Reasoning Workshop* (pp. 120–125). AAAI.

Simoudis, E. & Miller, J.S. (1990). Validated retrieval in case-based reasoning. In *Proceedings of AAAI-90.* San Mateo, CA: Morgan Kauffmann.

Stanfill, C., & Kahle, B. (1986). Parallel free-text search on the connection machine system. *Communications of the ACM, 29(12),* 1213–1228.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM, 29(12),* 1213–1228.

Stepp, R.E. III, & Michalski, R.S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning*, Vol. 2. Los Altos, CA: Morgan Kaufmann, pp. 471–498.

Sussman, G. (1975). *A computer model of skill acquisition*, Vol. 1 of *Artificial Intelligence Series*. New York: American Elsevier.

Thagard, P., & Holyoak, K. (1989). Why indexing is the wrong way to think about analog retrieval. In *Proceedings of a Workshop on Case-Based Reasoning*. Palo Alto, CA: Morgan Kaufmann.

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Reading, MA: Addison-Wesley.

Received October 2, 1990
Accepted February 18, 1992
Final Manuscript May 7, 1992