Machine Learning, 4, 259–283 (1989)

Supporting Start-to-Finish Development of Knowledge Bases

RAY BAREISS BAREISS@VUSE.VANDERBILT.EDU
Computer Science Department, Vanderbilt University, Nashville, TN 37235

BRUCE W. PORTER PORTER@CS.UTEXAS.EDU
KENNETH S. MURRAY MURRAY@CS.UTEXAS.EDU
Computer Sciences Department, University of Texas, Austin, TX 78712

Abstract. Developing knowledge bases using knowledge-acquisition tools is difficult because each stage of development requires performing a distinct knowledge-acquisition task. This paper describes these different tasks and surveys current tools that perform them. It also addresses two issues confronting tools for start-to-finish development of knowledge bases. The first issue is how to support multiple stages of development. This paper describes Protos, a knowledge-acquisition tool that adjusts the training it expects and assistance it provides as its knowledge grows. The second issue is how to integrate new information into a large knowledge base. This issue is addressed in the description of a second tool, KI, that evaluates new information to determine its consequences for existing knowledge.

Key words: knowledge-acquisition tools, knowledge-base refinement, knowledge-base development

1. Introduction

The purpose of a knowledge-acquisition tool is to help with *knowledge-base development*—the progression of a knowledge base from a level of complete ignorance to a desired level of knowledge. Supporting start-to-finish development is hard because different stages of development require different forms of assistance. In this paper we describe the support that current knowledge-acquisition methods provide, issues in supporting start-to-finish development, and specific tools we have built to study these issues.

Developing a knowledge base typically involves three stages: elicitation, refinement, and reformulation. During elicitation, the basic terminology and conceptual structure of the knowledge base is acquired. During refinement, knowledge is added to the skeletal structure and debugged. Finally, during reformulation, the knowledge base is optimized for problem solving.

Knowledge-acquisition tasks have been defined for each stage of development. For example, a task during knowledge refinement is to explain how a faulty conclusion was reached so that the knowledge base can be debugged. Section 2 describes the tasks required for each stage of development and surveys knowledge-acquisition tools that perform these tasks.

To support start-to-finish development, knowledge-acquisition tools must do more than sequentially perform these tasks. In addition to performing the tasks required for each stage of development, the knowledge needed to perform successive knowledge-acquisition tasks must be acquired. For example, during systematic elicitation, the justification for inferences should be acquired so that conclusions can be explained during knowledge refinement.

Requiring such stage-setting complicates developing knowledge bases with a single tool or a collection of tools.

Our research addresses two issues confronting tools for start-to-finish development of large-scale knowledge bases. The first issue is spanning multiple stages of development, which requires versatility to meet the needs of each stage. Ideally, transitions between stages are seamless, and there are no gaps in support during development. Section 3 describes Protos, which adjusts the problem-solving assistance it provides and the training it expects as the knowledge base develops. Its ability to support start-to-finish development is empirically demonstrated; through direct interaction with a domain expert, Protos has achieved proficiency at diagnosing hearing disorders and continues to learn as it is used.

The second issue is integrating new information into existing knowledge. *Knowledge integration* involves evaluating new information to determine its consequences for existing knowledge. For example, new information might conflict with existing knowledge or reveal gaps in the knowledge base. Although knowledge integration is performed throughout development, our research focuses on automating the task during the advanced stages of development. This focus identifies the knowledge required to perform the task, which is critical to its application. Section 4 discusses our current research on KI, a tool for knowledge integration that efficiently determines nonsuperficial consequences of new information.

2. The Tasks and Tools of Knowledge-Base Development

In general, there are three stages in developing a knowledge base:

- During systematic elicitation, the basic terminology and conceptual structure of the knowledge base is acquired.
- During knowledge refinement, the knowledge base is debugged and extended.
- During knowledge reformulation, the knowledge base is compiled to solve problems more efficiently.

Sections 2.1 through 2.3 describe these stages and survey knowledge-acquisition tools supporting each stage.

An ideal tool supports all stages of development. When there is little problem-solving knowledge available, the tool interviews the domain expert to acquire basic information. As the knowledge base grows, the tool helps identify gaps and inconsistencies responsible for problem-solving failures. Finally, the tool solves problems and improves performance without explicit training.

A "workbench of tools" approximates the ideal tool. The workbench organizes a collection of tools, each of which helps with a particular development phase. As discussed in Section 2.4, this method differs from the ideal tool in that the support it offers is not continuous. Moreover, many current tools are not good candidates for the workbench because they do not set the stage for the tools used after them. In order to compare research results, much of this discussion focuses on knowledge acquisition tools for heuristic classification. Heuristic classification is the predominant problem-solving method in current expert systems

[Clancey, 1985]. The method relates the features describing a problem to a predetermined set of solutions and is useful for a broad range of tasks, such as diagnosis, that require classifying an unknown object or situation.

2.1. Systematic Elicitation

The primary task during systematic elicitation is acquiring the conceptual structure of a knowledge base through a structured interview with the domain expert. The conceptual structure is a "description of the kinds of domain-specific inferences that the consultant will perform and the facts that will support these inferences" [Bennett, 1985]. For heuristic classification, this includes the predetermined set of solutions and the features that describe problems. Researchers believe elicitation of the conceptual structure must be systematic in order to prod the domain expert's memory and to avoid overlooking elements of the conceptual structure.

The knowledge required to perform systematic elicitation is a *model* to guide interaction with the user. The model describes the important components of a knowledge base for a problem-solving method or task. Some tools use a weak model of a generic problem-solving method, such as heuristic classification. Others use a strong model of a problem-solving method specific to a particular domain, such as diagnostic reasoning for disk failures.

ETS [Boose, 1984] is representative of systematic-elicitation tools that use a weak model of classification. The ETS model contains information about the classification process, such as the use of discriminating features to select among competing classifications. The first step in using ETS is enumerating the conclusions that the knowledge-based system should be able to reach. Given these conclusions, ETS systematically elicits the conceptual structure by directing the expert through two tasks. The first task is to identify features that discriminate among conclusions. The second task is to rate each feature's importance to each conclusion. ETS uses these features and associations to construct a prototype knowledge base. Ongoing research on Aquinus [Boose and Bradshaw, 1987] focuses, in part, on refining the knowledge base by adjusting the features' importance ratings and by expanding the sets of conclusions and features.

A systematic-elicitation tool using a strong model can be more focused. Such a tool is Roget [Bennett, 1985], which acquires the conceptual structure of a knowledge base by selecting and instantiating one of several available models. For example, starting with a model of medical diagnosis manually abstracted from a previously built knowledge base, Roget interviews the domain expert for specific diagnostic categories, symptoms, test results, predisposing factors, and rules for diagnosing blood infections. Going further in strengthening the problem-solving model, Opal [Musen, Fagan, Combs, and Shortliffe, 1987] uses a model of oncology to elicit chemotherapy treatment plans. The expert communicates by completing treatment forms using domain-specific terms.

Current tools for systematic elicitation are effective during the initial stage of knowledge base development. During this stage there is insufficient domain knowledge to solve problems. The tools interview a domain expert, but ask questions unlike those typically answered by the expert. In contrast, the tools discussed next acquire knowledge during problem solving.

2.2. *Knowledge Refinement*

The primary task during knowledge refinement is to incrementally debug a prototype knowledge base. Unlike conventional software systems, top-down development is impractical because the specification and design of a knowledge base cannot be formalized. Researchers have taken two approaches to knowledge refinement. Static analysis scans the knowledge base for patterns that suggest weak inference paths or missing knowledge. Dynamic analysis uses the knowledge base to process a set of test cases to reveal problem-solving errors. These analysis methods focus the developer on repairs and extensions of the knowledge base.

Teiresias [Davis, 1977], which employs dynamic analysis, exemplifies tools for knowledge refinement. The domain expert presents a test case to the performance system. If the expert deems the result incorrect, Teiresias traces the erroneous reasoning path. Teiresias highlights portions of the knowledge base that may be responsible for the mistake, and the expert repairs the gap or inconsistency. As the expert introduces inference rules, Tieresias compares them with rule models abstracted from the knowledge base. Each rule model records correlations between antecedent terms and consequent terms. If a new rule violates a pattern, Teiresias reports to the domain expert and suggests a modification of the rule that conforms to the pattern. The refinement process continues until the domain expert is satisfied with the system's performance. Other knowledge-refinement systems include MORE [Kahn, Nowlan, and McDermott, 1985], which uses static analysis, and MOLE [Eshelman, Ehret, McDermott, and Tan, 1987], which combines static and dynamic analysis.

Knowledge-refinement tools use the knowledge base in two ways. First, they use the problem-solving ability of the knowledge base to identify failures. When problem solving fails, the tool elicits knowledge from the expert to advance the knowledge base's development. Second, some tools use explicit knowledge of justifications for inference rules to determine the cause of problem-solving failures [Smith, Winston, Mitchell, and Buchanan, 1985] and to explain each failure [Neches, Swartout, and Moore, 1985]. Using the knowledge base in these ways requires an initial conceptual structure capable of solving problems, revealing bugs, and explaining failures. Most tools for knowledge refinement (and systematic elicitation) do not acquire this initial knowledge.

2.3. *Knowledge Reformulation*

The primary task during knowledge reformulation is compiling the knowledge base for more efficient problem solving. This requires an initial knowledge base, called the domain theory, which is assumed to be complete but nonoperational. For example, the domain theory for chess encodes all the rules for play but is inefficient for selecting good chess moves.

Leap/Vexed [Mitchell, Mahadevan, and Steinberg, 1985] exemplifies tools for knowledge reformulation. This tool is an apprentice to an expert VLSI circuit designer. Vexed is given a design specification and attempts to design a circuit. If Vexed fails, then the expert provides a solution, and Leap attempts to learn from this training. First, Leap uses its pre-existing domain theory, consisting of rules of logic and primitive building blocks for logic circuits, to construct a proof that the expert's design correctly implements the specification. Then,

the specification and the circuit design are generalized, subject to the constraints in the proof of correctness, to form a new design rule for Vexed's future problem solving.

Knowledge reformulation performed by Leap/Vexed is restricted by two requirements. First, the domain theory must be complete to explain every solution presented by the expert. Second, the domain theory must be strong and consistent to prove the solution is correct. Unfortunately, most domains of interest (e.g., medicine) cannot be formalized; consequently, their domain theories are incomplete, weak, and inconsistent. Additionally, knowledge-reformulation tools ignore the acquisition of the required knowledge base and address only improvements in efficiency, not in competence or explanation ability.

Integrating knowledge refinement with knowledge reformulation can address these limitations. For example, the Odysseus system [Wilkins, 1988] extends the domain theory used in knowledge reformulation. This learning apprentice observes an expert solving problems and attempts to explain the reasons underlying the expert's actions. Learning occurs when the explanation process fails. When one of the expert's actions cannot be explained, Odysseus conjectures new knowledge that would allow it to complete an explanation. The conjectured knowledge is validated by comparing it with a database of cases. If the hypothesized knowledge is consistent with the cases, it is added to the knowledge base.

2.4. Supporting Start-to-Finish Development with a Workbench of Tools

Tools for the knowledge-acquisition tasks discussed thus far support narrow phases of knowledge-base development. Conceivably, a collection of these tools could be combined into a workbench, which could support the start-to-finish development of knowledge bases. The approach is appealing, but several problems must be addressed.

The first problem with the workbench approach is that the knowledge base is rarely at a uniform level of development. For example, a knowledge base supporting heuristic classification contains inference rules relating observable data to final answers. During the knowledge base's development, parts of the knowledge base reliably classify and explain. Other parts are incomplete and erroneous. No narrow-spectrum tool from the workbench can refine the entire knowledge base. The second problem with the workbench approach is that the support it provides is not continuous. The developmental path is decomposed into discrete steps, and the consequences of supporting each step with a separate, narrow-spectrum tool can be severe:

- Mismatched knowledge representations—Each tool constructs and uses different representations for domain knowledge. For example, a tool for knowledge elicitation might represent correlations among domain terms in a rating grid, while a tool for knowledge refinement might use a causal model to solve a problem and explain its solution.
- Inconsistent user roles—Each tool has different requirements. For example, a tool for systematic elicitation requires a user familiar with basic terminology and high-level rules. However, a tool for knowledge refinement requires a user capable of debugging complex problems with the knowledge base.
- Inconsistent user interface—Each tool presents the user with a different interface, which is an unnecessary source of confusion.

A final problem with the workbench approach concerns the specific tools that have been developed thus far. Many tools for acquiring the initial domain knowledge do not set the stage for the tools that follow them. Tools for knowledge refinement and knowledge reformulation require a support structure for domain knowledge. For example, Teiresias presents the inference path leading to an incorrect conclusion so that the domain expert can identify faulty inferences. However, many tools for systematic elicitation do not acquire the intermediate inferences that support conclusions. Because of such mismatches, many tools are inappropriate for the workbench.

Because of these problems with the workbench approach, our research pursues an alternative—a single tool that supports start-to-finish development of knowledge bases. Sections 3 and 4 describe our approach to solving two issues critical to the success of such a tool.

3. Protos: Continuous Support During Knowledge Acquisition

Protos is a knowledge-acquisition tool that supports knowledge elicitation and refinement. It learns to perform heuristic classification by solving problems under the guidance of an expert teacher. When the expert describes a case, Protos assigns the case to a known category and explains the classification. If the classification is wrong or the explanation is inadequate, Protos interacts with the expert to acquire knowledge. Its domain-independent method of classification suggests possible causes of problem-solving failures and guides interaction to extend and refine the knowledge base.

Protos' problem-solving ability and interaction with the expert improve as it acquires knowledge. Initially, Protos lacks adequate knowledge for problem solving. It fails to classify cases and depends on the expert to provide classifications and explanations. As Protos acquires more knowledge, it becomes increasingly able to perform classification. However, misclassifications are frequent, and its ability to discriminate is refined. Through this evolving interaction, Protos becomes an autonomous problem solver whose learning concentrates on acquiring specific knowledge to improve its ability to explain its classifications.

The following dialogue illustrates Protos' early interaction with the expert as it acquires knowledge to classify hearing disorders.

Expert: NewCase has the features: notch_at_4k, sensorineural:mild, hystory:noise, air:normal, speech:normal, static:normal, tympanogram:a, *et cetera*
Protos: These features suggest the classification normal_ear. To confirm this, a known case of normal_ear is being compared to NewCase. All features match except notch_at_4k, sensorineural:mild, and history:noise. Is normal_ear the collect classification?
Expert: No.
Protos: Are any of the unmatched features of NewCase incompatible with normal_ear?
Expert: All are.
Protos: Protos cannot classify NewCase. What is its classification?
Expert: Cochlear_noise.
Protos: Please explain the relevance of case features to this classification.
Expert: notch_at_4k is usually caused by cochlear_noise.
et cetera
Protos: NewCase is being retained as the first exemplar of cochlear_noise.

Through this interaction, Protos refined its ability to discriminate instances of a known category of hearing disorder and acquired initial knowledge of a new one.

3.1. How Protos Performs Classification

Protos' classification method guides knowledge acquisition by relating steps in the classification process to the domain knowledge required to perform them successfully. Each type of failure suggests types of knowledge that are absent or faulty. This section describes the classification method and the knowledge it requires. Section 3.2 discusses how failures of the method guide knowledge acquisition.

Protos' method of classifying a case involves two steps. The first step is to hypothesize classifications based on the case's features (Figure 1). This step uses remindings, which are heuristic associations between features and categories learned from previous expert-provided explanations. The remindings associated with the features of a new case are combined to produce an ordered list of possible classifications.

The second step in classification attempts to confirm a hypothesis by matching the new case with prototypical exemplars (Figure 2). A process of *knowledge-based pattern matching* determines the similarity of the case and each exemplar. It uses previously acquired domain knowledge to explain how features of the case provide the same evidence as features of the exemplar. Overall similarity of the two cases is assessed by evaluating the quality of the resulting explanation and the importance of unmatched features.

If a match is imperfect, Protos searches for a more similar exemplar by traversing difference links associated with the current exemplar. Difference links connect exemplars and record their criterial differences.

Confirmation of the hypothesis is evaluated to determine Protos' next action. If the match is strong (i.e., adequately explained), it is presented to the user for approval and discussion. If it is weak, Protos considers other hypotheses and exemplars. Protos reports failure if its hypotheses are exhausted without finding an adequate match.[1]

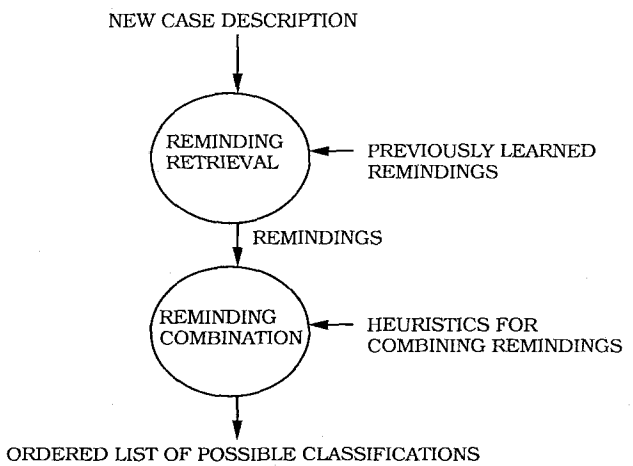


Figure 1. Step 1—Hypothesize classifications.

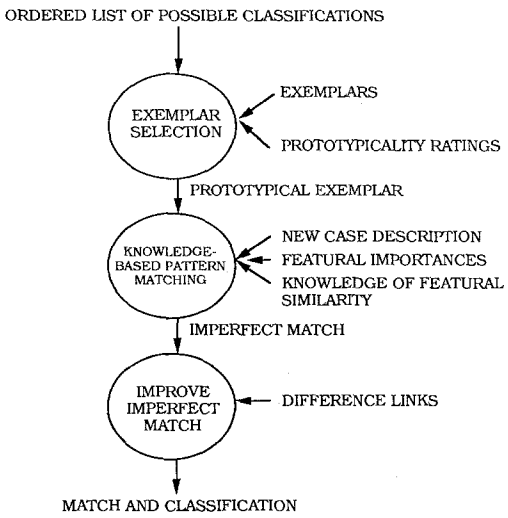


Figure 2. Step 2—Confirm a hypothesized classification.

3.2. How Failures Guide Knowledge Acquisition

Protos learns by analyzing and discussing failures of the classification method. The following general types of failures are possible:

1) Failure to classify—no classification can be determined
2) Failure to discriminate—an incorrect classification is reported
3) Failure to explain—the correct classification is inadequately explained

Protos associates each failure with a type of domain knowledge and interacts with the expert to acquire or refine the knowledge. Figure 3 presents Protos' algorithm for learning from failures.

Failure to classify a case indicates that Protos lacks knowledge of how a case's features determine its classification. The expert is asked to classify the case. Protos tries to relate each feature of the case to the provided classification by explaining its relevance. If Protos cannot relate a feature to the classification, the expert provides an explanation, which is added to the system's domain knowledge. After all of the features have been explained, the case is retained as a new exemplar of the classification.

Failure to discriminate occurs when Protos reports an incorrect classification. This indicates that Protos lacks knowledge to discriminate between instances and noninstances of the classification. Protos should not be able to match a new case to an exemplar of an incorrect classification. When such a match occurs, three possible causes are discussed with the expert. First, the expert is asked to evaluate the explanation relating the case and exemplar. Second, he is asked about unmatched features of the new case to determine whether any are incompatible with the classification. Third, he is asked for additional discriminating features.

```
GIVEN: a new case
FIND:  a classification of the case and an explanation of the
       classification

Search for an exemplar that matches the new case
IF not found
  THEN {classification failure}
       Ask teacher for classification
       Acquire explanations relating features to classification
       Compile remindings
       Retain case as an exemplar
  ELSE IF the teacher disapproves
         THEN {discrimination failure}
              Reassess remindings
              Discuss featural matches with the teacher
              Ask for discriminating features
              Remember unmatched features to add difference link
         ELSE {classification is correct}
              Increase exemplar's prototypicality rating
              IF match is incompletely explained
                THEN {explanation failure}
                     Ask teacher for explanations of featural
                     equivalence
                     IF not given
                       THEN Retain case as exemplar
                       ELSE {processing was successful}
```

Figure 3. The Protos algorithm for learning from failures.

Failure to explain a correct classification indicates that Protos lacks knowledge to support its classification. Protos and the expert discuss improvements to the explanation of the match between the new case and the recalled exemplar. Unmatched features of the exemplar are of particular concern. For each, the expert is asked to identify a corresponding feature in the new case and to explain their relationship. If the expert cannot provide these explanations, the case is retained as a new exemplar.

Protos is also concerned with learning efficient problem solving. Just as it elicits and refines domain knowledge by discussing problem-solving failures, Protos acquires and refines an indexing structure of remindings, difference links, and prototypicality ratings. As discussed in Section 3.1, these indices limit the search for matching exemplars during classification.

When Protos fails to classify, it acquires remindings. To correct the failure, the expert provides explanations relating each case feature to the classification. Protos compiles the explanations into remindings that directly associate features and classifications. The strength of each reminding is determined by evaluating the explanation's quality, using heuristics similar to Cohen's path endorsements [Cohen and Kjeldsen, 1987; Bareiss, 1989].

When Protos fails to discriminate, it refines remindings. The remindings that suggested the incorrect classification are reassessed to determine whether they are consistent with the system's current knowledge. Because Protos is incrementally acquiring domain knowledge, it attempts to regenerate the explanation from which each reminding was compiled to determine whether it is still valid. If the explanation is no longer valid, the reminding is weakened or removed.

Protos also acquires a difference link when a failure to discriminate occurs. A difference link records important featural differences that distinguish two exemplars. Upon adding the new case as an exemplar, Protos creates a difference link between the case and the improperly matched exemplar. Protos suggests the features to annotate the difference link, and the expert approves them.

When a correct match occurs, Protos increases the exemplar's prototypicality rating. Prototypicality is determined by family resemblance, that is, the degree to which an exemplar matches other category members. An increased rating makes selection of the exemplar more likely during subsequent classification attempts.

Explanations play two roles in knowledge acquisition. First, explanations describe the relevance of exemplar features to categories. Such explanations enable remindings to be compiled and importance of features to classifications to be estimated. Second, explanations describe how different features provide equivalent evidence for a classification. Such explanations provide knowledge to match cases that are not uniformly described.

An explanation is a plausible chain of relations linking domain terms in the knowledge base (e.g., "fur is consistent with mammal which has specialization dog"). Explanations are expressed in a predefined language of relations (e.g., "causes," "co-occurs with," "has part"), qualifiers (e.g., "usually," "sometimes," "occasionally") and expert-supplied domain terms [Bareiss, 1989]. Heuristics associated with specific relations allow Protos to evaluate their plausibility in the context of a particular explanation [cf. Cohen and Kjeldsen, 1987].

In summary, Protos elicits and refines domain knowledge by interacting with the expert in the context of problem-solving failures. Through classification and discrimination failures, it acquires exemplars, an indexing structure, and general domain knowledge. Through explanation failures, it acquires the ability to explain its (otherwise correct) classifications.

3.3. An Example of Protos' Evolving Interaction

This section presents two examples illustrating how Protos' interaction with an expert audiologist evolved to support start-to-finish development of a knowledge base for classifying hearing disorders. The first example is from early in training; Protos systematically elicits knowledge of a new classification from the expert. The second example is from late in training; Protos refines its ability to explain an otherwise correct classification. To enable direct comparison of the two stages of training, *NewCase,* the case discussed in the introduction to Section 3, was presented to Protos twice. Independent copies of the knowledge base were used so that knowledge acquired by processing *NewCase* the first time did not affect its processing the second time.

3.3.1. Processing NewCase Early in Training This example elaborates on the dialogue in the introduction to Section 3. *NewCase* was processed when Protos had seen few cases and lacked domain knowledge to classify correctly. At this stage of training, interaction with the expert primarily involved acquiring exemplars and systematically eliciting knowledge relating their features and classifications.

Based on past training, the features of *NewCase* (Figure 4) remind Protos of two possible diagnoses (Figure 5). When the individual remindings are combined, *normal_ear* is the strongest hypothesis. Protos retrieves the most prototypical exemplar of *normal_ear* and

Case: NewCase
Classification: Unknown

sensorineural: mild
notch_at_4k
history: noise
speech: normal
oc_acoustic_reflex: normal
oi_acoustic_reflex: elevated
i_acoustic_reflex: normal
c_acoustic_reflex: normal
static: normal
tympanogram: a
air: normal

Figure. 4. The features of the example case.

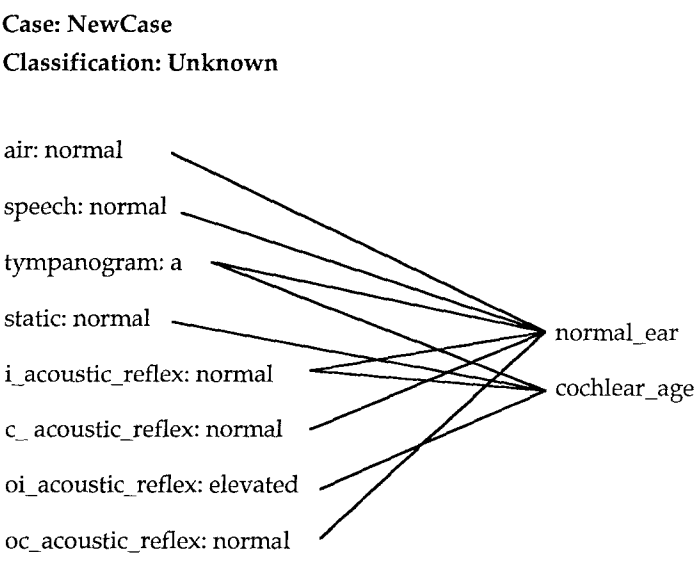


Figure. 5. Hypotheses associated with the features of *NewCase* when presented early in training.

attempts to match it to *NewCase* to confirm the hypothesis (Figure 6). Protos believes the match to be strong since all of the exemplar's features are matched. However, when the match is presented for discussion, the teacher rejects it as incorrect.

This failure to discriminate results in learning to classify cases of *normal_ear* more accurately. Since the exemplar's features are perfectly matched by *NewCase,* there are only a few possible reasons for the problem-solving failure. Protos pursues them systematically. First, Protos independently reassesses the remindings, which led it to *normal_ear,* with respect to its domain knowledge. It is able to regenerate the explanation from which each was compiled, suggesting that all are still valid. Second, Protos asks about the features of *NewCase* that were not matched by the exemplar and is told that all are incompatible

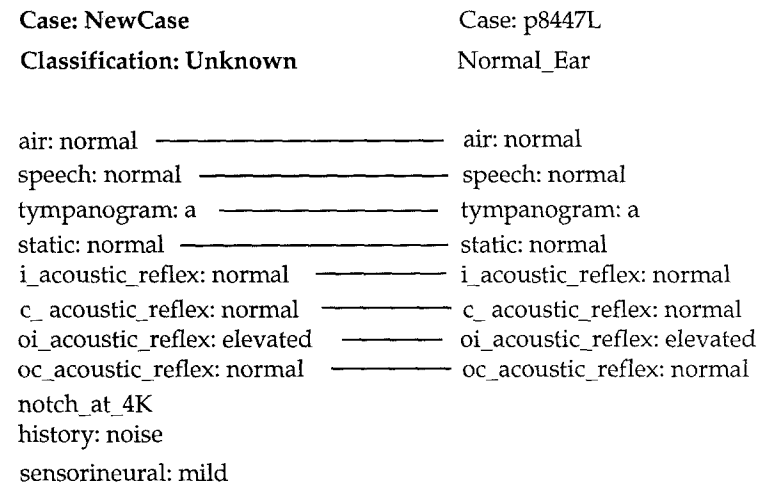


Figure. 6. Matching *NewCase* to an exemplar of normal ear.

with the classification. Third, when it asks whether the exemplar has additional features that discriminate it from *NewCase,* the teacher does not identify any.

Protos then tries to confirm its second diagnostic hypothesis, *cochlear__age.* It is unable to find an adequately similar exemplar of this category, so discussion with the teacher is not possible. Protos fails to confirm its hypothesis and, as before, reassesses the remindings that suggested *cochlear__age.*

Having exhausted the hypotheses, Protos reports a failure to classify. It asks the teacher to provide a classification, and he classifies *NewCase* as *cochlear__noise.* Since Protos has no exemplars of this category, *NewCase* is retained as an exemplar. Protos then interacts with the teacher to acquire general knowledge of *cochlear__noise.* It asks the teacher to explain the relevance of each case feature to the classification (using the predefined explanation language). The teacher provides explanations such as "*history:noise* is required by *cochlear__noise.*" From these explanations, Protos compiles remindings linking the features and the classification. For example, *history: noise* is inferred to be highly predictive of *cochlear__noise.*

Since the failure to classify was preceded by a failure to discriminate, Protos installs a difference link between the new exemplar and the exemplar of *normal__ear* that is erroneously matched. The difference link is annotated with the features of *NewCase* that were not matched by features of the exemplar of *normal__ear.*

3.3.2. Processing NewCase Late in Training After Protos had seen 175 cases, *NewCase* was presented to illustrate the shift from knowledge elicitation to refinement.[2] As it acquires knowledge, Protos becomes increasingly competent at problem solving and expects qualitatively different training from the expert. At this stage, Protos' classifications are generally correct, and interacton focuses on refining explanations.

As before, Protos hypothesizes classifications using remindings compiled from explanations of previous cases. Based on combining the remindings shown in Figure 7, Protos' best

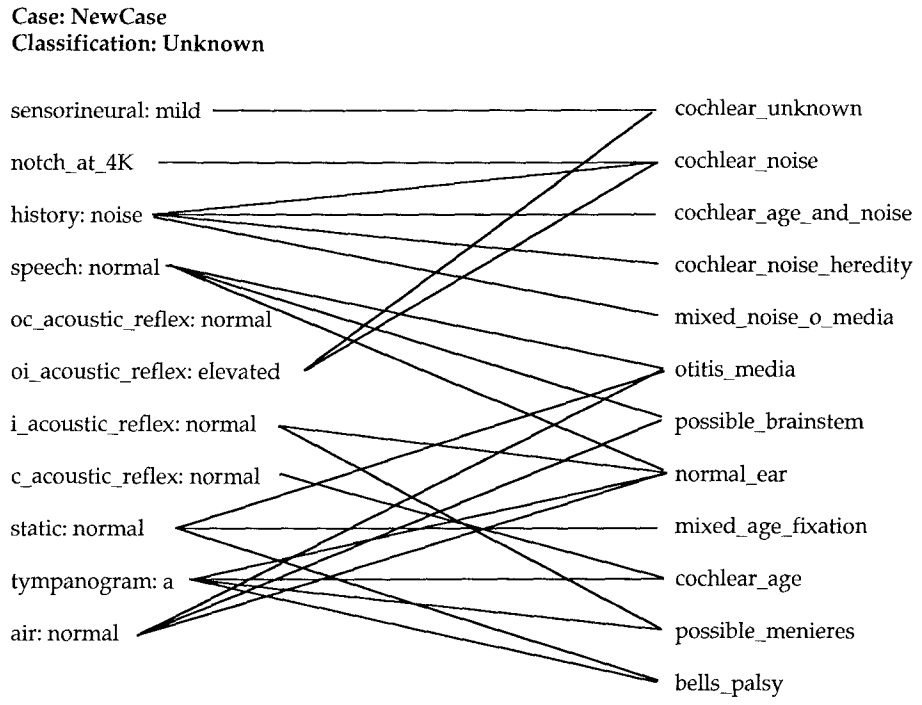


Figure. 7. Hypotheses associated with features of *NewCase* when presented late in training.

hypotheses are *cochlear__age&noise* and *cochlear__noise.* Protos rejects the first hypothesis when it cannot find a matching exemplar and reassesses the remindings to *cochlear__age&noise.* It then tries *cochlear__noise* and finds a good match, which is illustrated in Figure 8.

Most of the features of the two cases match directly. The match between *sensorineural:mild* and *sensorineural:moderate* is an exception. Protos can match these features because of a past, expert-provided explanation that the two values of *sensorineural* are sometimes interchangeable in the context of this diagnosis.

Discussion with the expert focuses on Protos' failure to explain the match completely. Protos asks the expert whether the unmatched features of *NewCase* are equivalent to the unmatched exemplar features. He tells Protos:

> *notch__at__4k* is definitionally equivalent to *notch__4k*

and

> if the category is *cochlear__noise* then *c__acoustic__reflex: normal*
> is sometimes interchangeable with *c__acoustic__reflex: elevated*

Protos does not retain *NewCase* because any future case that would match *NewCase* would match the existing exemplar equally well. The prototypicality of the exemplar is increased to credit its participation in a close, successful match.

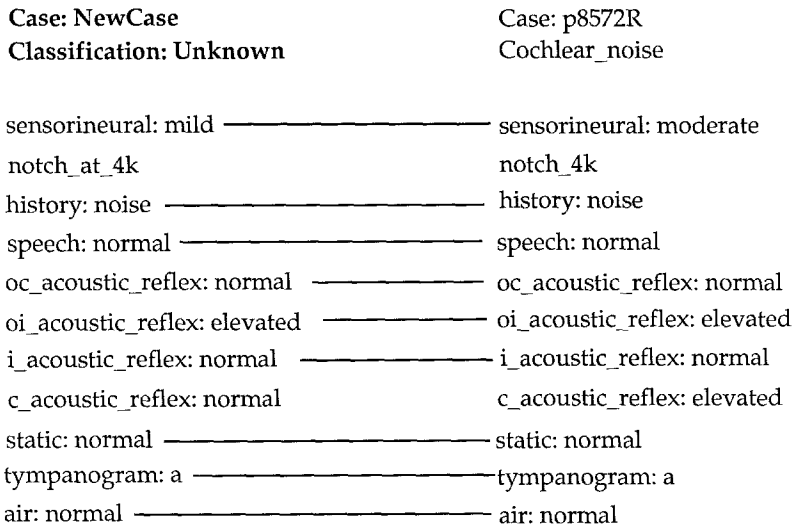


Figure 8. Matching *NewCase* to an exemplar of cochlear_noise.

Processing *NewCase* at two points in the evolution of the audiology knowledge base illustrates how Protos supports different stages of knowledge acquisition. The first time *NewCase* was processed, the system had little domain knowledge and was unable to classify it correctly. Through discussing the failure to classify, Protos acquired knowledge of a new classification, an exemplar, and the relevance of the exemplar's features to the classification. The expert was asked to provide a considerable amount of explanation relating *NewCase* to the system's existing knowledge. Discussion of the failure to discriminate *NewCase* from a case of *normal_ear* refined Protos' indexing knowledge.

The second time *NewCase* was processed, Protos had more extensive knowledge and could determine the correct classification independently. The expert played the more limited role of explaining relationships between features, which improved Protos' ability to explain its classification.

3.4. Evaluating Protos in Clinical Audiology

A distinct advantage of studying knowledge acquisition for expert systems is the evaluation criteria that it affords. Problem-solving proficiency can be measured as knowledge accumulates and, ultimately, can be compared with human experts. It is somewhat surprising that, with few exceptions [*e.g.*, Quinlin, 1986; Michalski, 1987], knowledge acquisition tools have not been evaluated. This section describes some of the data collected to assess Protos' viability.

Protos was trained using 200 hearing-disorder cases from the files of a large clinic. The training set was random. Its size was restricted to 200 cases because this is approximately the number of cases that a human audiologist sees during graduate school. For Protos to

be considered successful, it was deemed necessary for the system to classify accurately and efficiently given a similar amount of training. After training, Protos' performance was evaluated using a random set of 26 new cases. The characteristics of the training and test cases are presented in Table 1.

The fundamental assessment of Protos' performance is the correctness of its classifications. Protos correctly classified 82% of the training set while learning. Afterwards, Protos correctly classified 100% of the test cases.[3]

Table 1. Characteristics of Cases Presented to Protos

Characteristic	Training Set	Test Set
Number of Cases	200	26
Number of Categories	24	6
Exemplars Retained	120	—
Mean Features/Case (Total number of features=73)	10.6	11.5

Protos' problem-solving efficiency can be measured by the amount of effort it expended during classification. The average number of diagnostic hypotheses pursued and the number of matches attempted gradually increased (Table 2). However, as a percentage of possible hypotheses, the number of hypotheses pursued decreased. As a percentage of possible exemplars, the number of matches attempted remained fairly constant. The number of matches presented to the expert remained fairly constant as well. The corresponding percentage decreased, indicating increasing autonomy. Most of the classification process was independent of the expert.

Table 2. Classification Effort Expended

Cases	Hypotheses Pursued	Matches Attempted	Matches Discussed
1–50	2.7 (25.5%)	not available	1.7 (3.7%)
51–100	2.8 (17.5%)	not available	1.6 (1.9%)
101–150	2.5 (11.9%)	4.6 (4.4%)	1.5 (1.4%)
151–200	4.0 (16.7%)	7.4 (6.2%)	1.9 (1.6%)
average	3.0	6.0	1.6
test	3.7 (15.4%)	5.3 (4.4%)	1.1 (0.9%)

The evolution of Protos' interaction with the expert can be seen in a gradual shift in the type of explanations Protos elicited (Figure 9). As the knowledge base evolved, Protos' focus shifted from attaining competence at classification to attaining competence at explanation. Early training was dominated by classification failures. Protos primarily elicited explanations relating features to classifications as new exemplars were retained. Gradually, classification failures were superceded by explanation failures, and interaction with the teacher shifted to explaining the similarity of features in the context of particular diagnostic categories. This knowledge refined the system's ability to explain its correct classifications.

The design of Protos and its experimental evaluation in the domain of clinical audiology are more completely described in Bareiss [1989]. A Common Lisp implementation of Protos is available [Dvorak, 1988] and has been widely distributed.

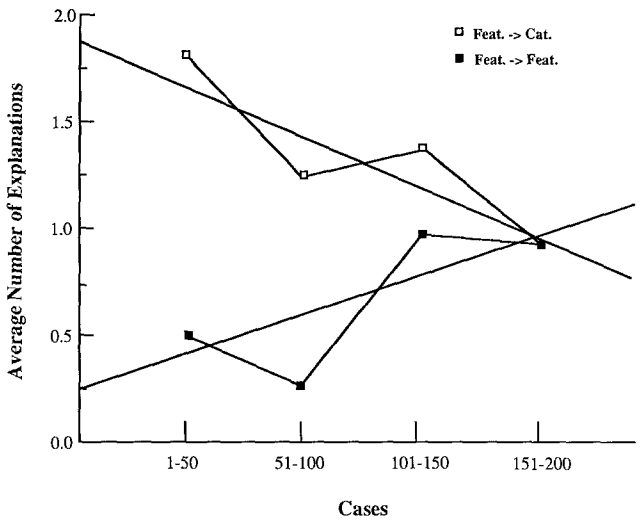


Figure 9. Teacher-provided explanations per case.

3.5. Strengths and Limitations

Protos' primary strength as a knowledge-acquisition tool is its knowledge of the classification method. This initial knowledge relates steps in the classification process to the domain knowledge required to perform them. In particular, Protos relates failures during classification to particular forms of knowledge that are absent or faulty. This rich decomposition of the problem-solving task is perhaps the most useful form of a priori knowledge for knowledge-acquisition tools. [Bylander and Chandrasekaran, 1987].

Protos' classification method is effective during both systematic elicitation and knowledge refinement. Initially, Protos is unable to classify cases, and it acquires knowledge from the expert in the form of explained examples. As Protos acquires more knowledge, it becomes increasingly able to classify cases, although misclassifications are common. Using explicit knowledge of possible failures, Protos interacts with the expert to refine the knowledge base. Protos becomes an autonomous problem solver and acquires knowledge to improve its ability to explain its classifications. However, Protos' classification method is ineffective for determining the consequences of knowledge-base modifications and extensions. As with most knowledge-acquisition tools, Protos is a "lazy evaluator" of new information. Some of the consequences of a knowledge-base change are revealed during problem solving, which is interleaved with knowledge acquisition; other consequences are undetected. "Eager evaluation" of new information is preferred for two reasons. First, it detects inconsistencies and knowledge-base gaps before they cause failures. Second, it enables a tool to respond to new information with follow-up inferences and questions. The next section describes a tool that performs this task and identifies the required knowledge.

4. KI: Integrating New Information During Knowledge Refinement

KI is a knowledge acquisition tool being developed to support integrating new information during knowledge refinement. Although this research is preliminary, a prototype of KI has been implemented that demonstrates several benefits from aggressively evaluating knowledge-base modifications and extensions. When new information is provided, KI uses the existing knowledge to critique the new information and determine its consequences. Determining these consequences reveals inconsistencies and gaps in the knowledge base. KI elicits information from the knowledge engineer to fill the gaps and resolve the inconsistencies. KI's computational model of knowledge integration includes three prominent activities:

1) Recognition—identifying the knowledge relevant to new information
2) Elaboration—applying the expectations provided by relevant knowledge to determine the consequences of the new information
3) Adaptation—modifying the knowledge base to accommodate the elaborated information

Current knowledge refinement tools avoid in-depth evaluation of knowledge acquired during knowledge refinement. Some tools simply add new information and ignore its consequences, assuming that inconsistencies will be exposed as problem-solving failures and corrected as they occur. Other approaches have been limited to detecting surface inconsistencies [e.g., Davis, 1977; Wilkins, 1988]; however, these approaches cannot detect subtle inconsistencies introduced by knowledge-base revisions, because they ignore implicit consequences of new information for existing domain knowledge. FIE [Cohen, 1984] improves on these approaches by using resolution to determine the shallow interaction between new information and existing beliefs. However, this approach lacks sufficient control to integrate extensions into a large knowledge base or to identify the deep consequences of new information.

KI's approach to controllng the search for the consequences of new information uses a form of domain knowledge called *views*. Each view defines a segment of the knowledge base comprised of concepts that interact in some significant way. Views are used to heuristically guide the search during knowledge integration by identifying the inference paths worth pursuing when the representation of a concept is extended with new information.

KI is being developed to assist knowledge engineers to extend the Botany Knowledge Base [Porter, Lester, Murray, Pittman, Souther, Acker, and Jones, 1988], which contains approximately 4,000 frames representing task-independent knowledge about plant anatomy, physiology, and development. The following sections describe KI's preliminary development. Section 4.1 describes an example of knowledge integration that is representative of the complex knowledge-base extensions we expect KI to perform. A prototype implementation of KI has been successfully tested with this example. Sections 4.2 through 4.4 describe how KI performs the tasks of recognition, elaboration, and adaptation. In Section 4.5, the strengths and limitations of this approach are reviewed.

4.1. An Example of Knowledge Integration

This example involves extending the knowledge base with new information about plant seeds. The knowledge base already has the information that plant seeds contain nutritive tissue

called endosperm (which is analogous to an egg's yoke). The plant embryo consumes the endosperm during its development inside the seed. A knowledge engineer wishes to extend the knowledge base with a representation of *nonendospermic seed,* a type of seed that contains no endosperm. The task of KI is to interact with the knowledge engineer to integrate this new information. The knowledge engineer presents new information to KI:[4]

Knowledge Engineer: There is a class of seeds that have no endosperm.

With assistance from the knowledge engineer, KI identifies and retrieves knowledge structures relevant to this new information. Using inference rules defined for the retrieved knowledge structures, KI forms the expectation that the embryo in a nonendospermic seed will die of starvation. It then attempts to confirm this expectation with the knowledge engineer:

KI: Endosperm is a source of plant nutrients; is this relevant?
Knowledge Engineer: Yes.
KI: Seeds without endosperm do not give rise to healthy seedlings.
Knowledge Engineer: Explain.
KI: The embryo starves since the endosperm provides nutrients, and nutrients are essential for health and survival.
Knowledge Engineer: Yes, nutrients are essential; however, the embryo survives.

KI attempts to resolve the inconsistency between its expectations and the assertion that embryos in nonendospermic seeds survive. One possible explanation is that these embryos acquire nutrients from some other source. KI searches for alternate ways an embryo-stage plant might acquire nutrients. A focused discussion with the knowledge engineer ensues that resolves the anomaly and extends the knowledge base.

KI: Does the embryo acquire nutrients from photosynthesis?
Knowledge Engineer: Yes, the shoots of some plant embryos emerge from the seed during germination and start producing photosynthate.

As this example illustrates, KI integrates new information by determining its consequences. When conflicts are encountered, KI searches for alternative explanations to resolve them. The computational issues that arise during knowledge integration include identifying knowledge relevant to new information, relating relevant knowledge to the new information, and adapting the knowledge base to accommodate the new information. The following three sections describe in greater detail how KI performs these activities.

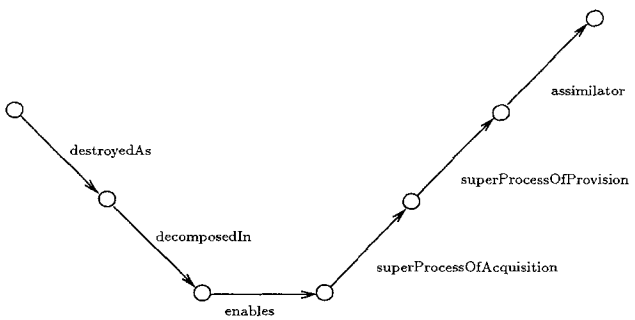
4.2. Recognition

KI begins knowledge integration by identifying relevant knowledge structures. In the previous example about seeds with no endosperm, KI must determine which among the thousands of frames in the Botany Knowledge Base may be affected in some way.

The representation of each object in the Botany Knowledge Base is structured with views to focus the search for knowledge relevant to new information. Each view is a segment of the knowledge base that identifies concepts that interact in some significant way. Perspectives

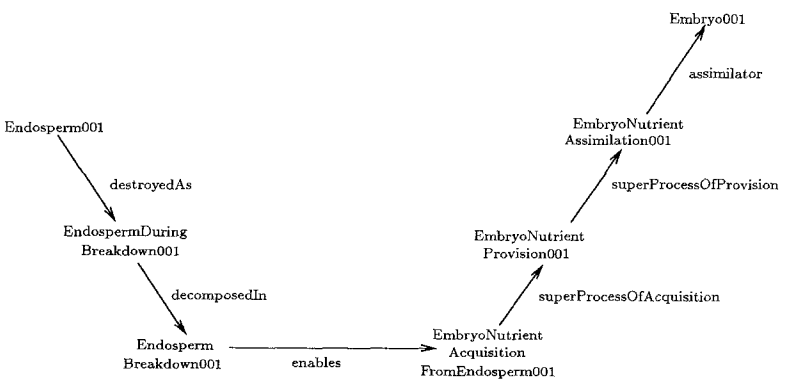
are a common type of view that represent concepts in particular roles. For example, one perspective of endosperm is *Plant Food Source,* as shown in Figure 10. Other perspectives include: endosperm as a *Product Of Reproduction*, endosperm as a *Contained Object*, and endosperm as a *Taxon Defining Part*. KI collects the views for objects referenced by new information and prompts the knowledge engineer to select which are appropriate.

A view is represented as a semantic-net template that can be instantiated for hypothetical objects. KI instantiates the views selected by the knowledge engineer. The instantiation of *Plant Food Source* for an endosperm is presented in Figure 11. Collectively, these instantiated frames comprise a *context* representing an endosperm in its role as a plant food source; this context is used to simulate the effects of the new information about endosperm.



This perspective, represented as a semantic-net template, defines the concepts relevant to an object in its role as a plant food source: a plant food source must have a stage when it is destroyed and decomposed into nutrients; this decomposition enables the nutrients to be assimilated by the plant; nutrient assimilation involves the provision and acquisition of nutrients.

Figure 10. The perspective *Plant Food Source.*



The perspective of Figure 10 is instantiated for a hypothetical endosperm: the endosperm is decomposed into nutrients which are assimilated by a hypothetical plant embryo. This context is used to simulate the effects of seeds not having endosperms.

Figure 11. The context created by instantiating *Plant Food Source.*

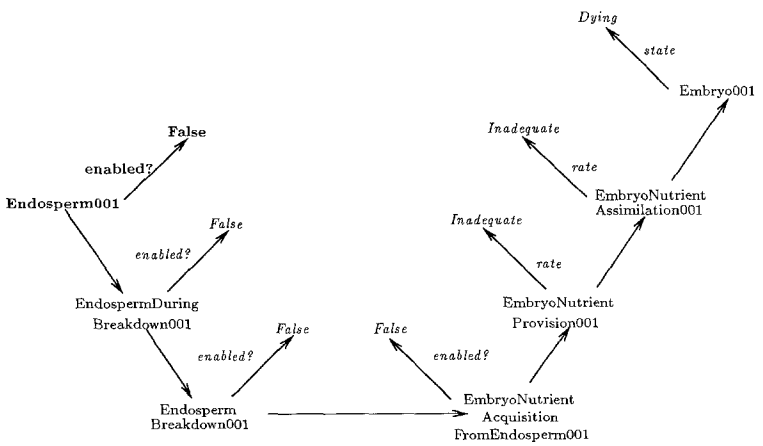
4.3. Elaboration

During recognition, KI creates a context by instantiating concepts in the knowledge base most relevant to the new information. Next, during elaboration, KI determines how the new information interacts with existing knowledge within this context. Elaboration involves applying inference rules to propagate the effects of the new information throughout the context.

In the endosperm example, elaboration begins when KI asserts that the endosperm is absent from the context by assigning value *False* to the slot *enabled?* of *Endosperm 001*. This assignment triggers inference rules that determine the consequences of seeds lacking endosperm. For example, without the endosperm, the embryo cannot get enough nutrients to survive. Only rules that apply to frames in the context are considered; therefore, by selectively instantiating frames during recognition, KI controls the inferences that are attempted during elaboration. The inference rules applicable to this example are listed in Figure 12, and the elaborated context is presented in Figure 13.

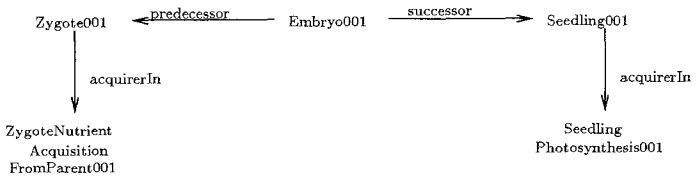
1. When an entity is disabled, all of its developmental stages are disabled.
2. When an entity is disabled, all the processes involving the entity are disabled.
3. When a process is disabled, all the processes that its completion enables are disabled.
4. When the known methods of acquiring some essential resource are disabled, the rate of providing the resource is inadequate for survival.
5. When the assimilation rate for some resource is unknown, it is the same as the provision rate.
6. When nutrient assimilation is inadequate for survival, the assimilator is dying.

Figure 12. Heuristic rules relevant to endosperm as a plant food source.



The hypothetical endosperm is disabled, triggering the inference rules of Figure 12, which propagate the effects of this assertion throughout the context. The predicted consequences of seeds lacking endosperms are presented in italics.

Figure 13. The elaborated context.



The context of Figure 13 is extended to include the developmental predecessor and successor of *Embryo 001* and their methods of nutrient acquisition.

Figure 14. The context extension.

Through elaboration, KI concludes that the plant embryo is dying from lack of nutrients. This triggers the instantiatin of a second view defined for plants that are starving and in danger of dying. The original context is expanded to include the plant's developmental stages immediately before and after its embryo stage and how nutrients are acquired during each of these developmental stages. This additional knowledge is presented in Figure 14. Through continued elaboration, KI concludes that the plant's seedling stage is not reached because the plant dies during its embryo stage.

An important function of elaboration is identifying confounded expectations. These occur when expectations of the knowledge base are violated by new information or when two rules reach conflicting conclusions. Resolving inconsistencies involves correcting the new information to comply with current expectations or adapting the existing knowledge structures to accommodate the new information.

4.4. Adaptation

Elaboration reveals anomalies in the knowledge base; adaptation resolves them. An anomaly can result from inconsistencies introduced either by inference rules used during elaboration or by facts the knowledge engineer asserts. In the endosperm example, an anomaly is detected when the knowledge engineer asserts that the embryos of nonendospermic seeds survive, correcting the prediction that these embryos starve.

Resolving anomalies requires correcting explanations that support failed expectations and constructing alternative explanations to account for new information. When the knowledge engineer refutes the prediction that embryos of nonendospermic seeds starve, KI inspects the explanation for this prediction to determine its weakest premise. This suspect explanation is presented in Figure 15. Rule 4 (from Figure 12) relies on a closed-world assumption and is considered a relatively weak inference. Therefore, KI retracts its conclusion and assumes *Embryo Nutrient Provision 001* is adequate for the embryo's survival. This change propagates through the explanation, retracting the belief that the embryo starves.

The original anomaly has been resolved by assuming that the embryos of nonendospermic seeds receive adequate nutrients. However, no alternative method is known for plant embryos to acquire nutrients. KI seeks to construct an explanation for the assumed nutrient acquisition using the following inference:

If a resource provision is adequate for survival, but no acquisition method is known, then assume the acquisition method of the developmental successor is employed.

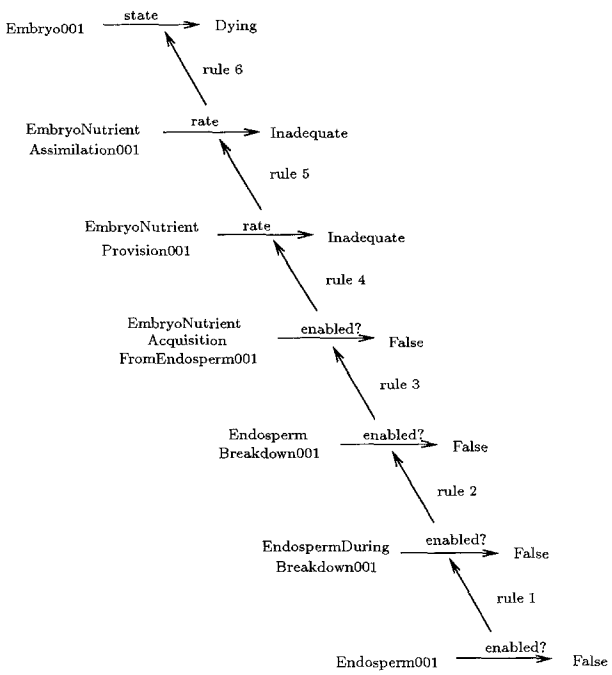
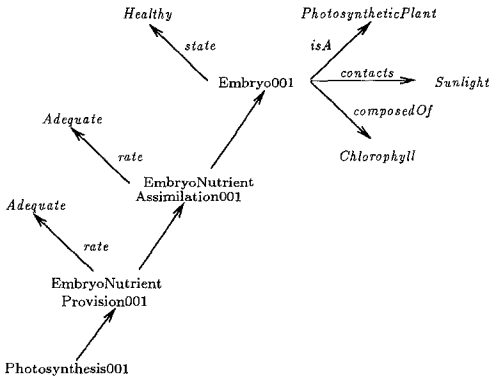


Figure 15. The suspect explanation.

This rule suggests the embryos of nonendospermic seeds acquire nutrients by photosynthesis, as is done by seedlings. However, this hypothesis introduces new constraints on the embryos of nonendospermic seeds. For example, to acquire nutrients by photosynthesis, the embryo must be a photosynthetic plant. Therefore, to apply this inference, KI asserts that *Embryo 001* is an instance of *Photosynthetic Plant.* As a photosynthetic plant, the embryo inherits the following features: it contacts sunlight, and its composition includes chlorophyll. This is illustrated in Figure 16. In short, the plausibility of explaining the survival of nonendospermic embryos by assuming they engage in photosynthesis is contingent on their contacting sunlight and possessing chlorophyll. Confirming these assumptions leads to the acquisition of further knowledge from the knowledge engineer.

4.5. KI's Strengths and Limitations

KI can partially determine the consequences of new information because it has access to substantial domain expertise and a method for heuristically determining what existing knowledge is relevant. Using existing knowledge to elaborate new information enables KI to acquire more than what is literally expressed by the new information. KI uncovers implicit conflicts between new information and existing knowledge and assists the knowledge engineer with resolving them. However, because our approach assumes substantial domain knowledge, it is inappropriate during the initial stages of knowledge-base development when the encoded domain expertise is sparse.



The context is adapted to account for adequate nutrient provision when no nutrients are acquired from the endosperm. Assuming the plant embryo acquires nutrients through photosynthesis requires that it contacts sunlight and possesses chlorophyll.

Figure 16. The adapted context.

Identifying all interactions between new information and existing knowledge is intractable; performing knowledge integration requires restricting this search. KI uses views in two ways to avoid an intractable search. First, views are used as a control mechanism to provide a coarse granularity in searching for the consequences of new information. This permits KI to efficiently identify deep consequences of new information within selected contexts. Second, views define local, computational environments. We are developing KI to enforce consistency of the knowledge-base within views. This policy operationalizes the adage of maintaining *local consistency* and obviates the intractable task of computing the deductive closure of the knowledge base.

Our prototype implementation of KI has two shortcomings, which our current research partially addresses. First, the knowledge engineer is required to choose from views that KI considers relevant. We are designing an agenda-based search mechanism that automates view selection [Murray and Porter, 1989]. Second, views are rigid because each is represented by an explicit path of slots (as in Figure 10). The number of views needed to structure a knowledge base is large, and each view is manually created. We are studying alternative representations of views [e.g., Porter, Souther, Lester, and Acker, 1989] and ways to acquire views during knowledge integration.

5. Summary

Knowledge-acquisition tools help with knowledge-base development—the progression of a knowledge base from a level of complete ignorance to a desired level of knowledge. Development typically involves three stages: elicitation of initial knowledge, refinement of a prototype knowledge base, and reformulation of knowledge to improve performance. Each stage requires a particular form of assistance, and most knowledge-acquisition tools support only a single stage.

Conceivably, start-to-finish development of knowledge bases could be supported by a workbench of tools. The workbench organizes a collection of tools, each of which helps with a particular development stage. However, the support is not continuous, and many current tools are inappropriate for the workbench because they do not acquire the knowledge that subsequent tools require.

Our research pursues an alternative—supporting start-to-finish development of knowledge bases with a single tool. We have studied two issues crucial to building such a tool. The first issue is spanning multiple stages of development. We have built Protos, a tool that acquires knowledge while assisting a domain expert to solve classification problems. Initially, Protos' knowledge is inadequate for solving problems; it interviews the expert to acquire the domain's conceptual structure. As the knowledge base develops, problem solving improves and errors are discussed with the expert to refine the knowledge base. Protos becomes an autonomous problem solver and continues to improve its ability to explain its classifications. Its effectiveness has been demonstrated with the construction of a knowledge-based system for diagnosing hearing disorders.

The second issue addressed by our research is knowledge integration. As a large-scale knowledge base is developed, the importance and the difficulty of performing knowledge integration increases. We are building KI, a tool that evaluates new information to determine its consequences for existing knowledge. KI controls the search for consequences with a form of domain knowledge called views. Each view identifies the inference paths to pursue when the representation of a concept is extended with new information. We are applying KI to make complex extensions to a large-scale knowledge base.

Acknowledgments

Support for this research was provided by the Army Research Office under grant ARO-DAAG29-84-K-0060, the National Science Foundation under grant IRI-8620052, and contributions by Apple, Texas Instruments, and the Cray Foundation. We are indebted to Professor Craig Wier for serving as the domain expert for the application of Protos to clinical audiology and to Professor James Jerger of the Baylor College of Medicine for providing the training cases. We are grateful to Robert Holte for providing useful commentary on early drafts of this report. We appreciate the assistance of Joe Ross, Claudia Porter, and Ken Murray during the development of Protos. Ken Murray and Bruce Porter are the primary researchers on the KI project, with substantial contributions from Art Souther, Liane Acker, James Lester, and Karen Pittman.

Notes

1. Protos does not consider all of the categories of which it is reminded. Only the strongest remindings are considered. Furthermore a category is considered only if no matching subordinate category can be found.
2. Again, an independent copy of the knowledge base was used that did not contain *NewCase* or the associated indices and domain knowledge.

3. Note that the ability to classify cases into *known* categories is being reported; the 24 training instances that introduced new diagnostic categories are excluded from the training percentage.
4. This example has been simplified for presentation. For example, KI does not generate and parse natural language; this discourse has been converted from a language of frames, slots, and values. A complete description of the prototype implementation and this example are provided in Murray [1988].

References

Bareiss, R. 1989. *Exemplar-based knowledge acquisition: A unified approach to concept representation, classification, and learning.* (Based on PhD dissertation, University of Texas at Austin, Austin, TX: Department of Computer Sciences), Academic Press.

Bennett, J.S. 1985. ROGET: A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system. *Automated Reasoning, 1,* 49–74.

Boose, J. 1984. Personal construct theory and the transfer of expertise. *Proceedings of the National Conference on Artificial Intelligence,* (pp. 27–33).

Boose, J., and Bradshaw, J. 1987. Expertise transfer and complex problems: Using Aquinus as a knowledge acquisition workbench for knowledge-based systems. *International Journal of Man-Machine Studies 26,* 1, 3–28.

Bylander, T., and Chandrasekaran, B. 1987. Generic tasks for knowledge-based reasoning: The right level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies 26,* 231–243.

Clancey, W.J. 1985. Heuristic classification. *Artificial Intelligence 27,* 289–350.

Cohen, P., and Kjeldsen, R. 1987. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management 23,* 255–268.

Davis, R. 1977. Interactive transfer of expertise: Acquisition of new inference rules. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 321–328).

Dvorak, D. 1988. *Guide to CL-Protos: An exemplar-based learning apprentice.* (Technical Report AI88–87). Austin, TX: University of Texas, Department of Computer Sciences.

Eshelman, L., Ehret, D., McDermott, J., and Tan, M. 1987. MOLE: A tenacious knowledge acquisition tool. *International Journal of Man-Machine Studies 26,* 41–54.

Kahn, G., Nowlan, S., and McDermott, J. 1985. MORE: An intelligence knowledge acquisition tool. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 581–584).

Michalski, R.S. 1987. How to learn imprecise concepts: a method for employing a two-tiered knowledge representation in learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 50–58).

Mitchell, T.M., Mahadevan, S., and Steinberg, L.I. 1985. LEAP: A learning apprentice for VLSI design. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 573–580).

Murray, K. 1988. *KI: An Experiment in Automating Knowledge Integration.* (Technical Report AI88–90). Austin, TX: University of Texas, Department of Computer Sciences.

Murray, K., and Porter, B. 1989. Controlling search for the consequences of new information during knowledge integration. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 290–295).

Musen, M.A., Fagan, L.M., Combs, D.M., and Shortliffe, E.H. 1987. Use of a domain model to drive an interactive knowledge-editing tool. *International Journal of Man-Machine Studies 26,* 105–121.

Neches, R., Swartout, W.R., and Moore, J.D. 1985. Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Transactions on Software Engineering 11,* 1337–1351.

Porter, B., Souther, A., Lester, J., and Acker, L. 1989. Generating explanations in an intelligent tutor designed to teach fundamental knowledge. *Proceedings of the 2nd Intelligent Tutoring Systems Research Forum,* (pp. 55–69).

Quinlan, J.R. 1986 Induction of Decision Trees. *Machine Learning 1,* 81–106.

Smith, R.G., Winston, H.A., Mitchell, T.M., and Buchanan, B.G. 1985. Representation and use of explicit justifications for knowledge base refinements. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 673–680).

Wilkins, D.C. 1988. Knowledge base refinement using apprenticeship learning techniques. *Proceedings of the National Conference on Artificial Intelligence* (pp. 646–651).