

Knowledge Acquisition as Knowledge Assimilation

Lawrence S. Lefkowitz
Victor R. Lesser

COINS Technical Report 87-13
February 1987

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

Knowledge bases are constructed and refined using information obtained from domain experts. The assimilation of this information into an existing knowledge base is an important facet of the knowledge acquisition process. This knowledge assimilation requires an understanding of how the new information corresponds to that already known by the system and how this existing information must be modified so as to reflect the expert's view of the domain.

This paper describes a system, K^{NAc} , that modifies an existing knowledge base based on a discourse with a domain expert. Using heuristic knowledge about the knowledge acquisition process, K^{NAc} anticipates modifications to the existing entity descriptions. These anticipated modifications, or *expectations*, are used to provide a context in which to assimilate the new domain information.

This research was supported, in part, by the External Research Program of Digital Equipment Corporation, by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and by the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F 30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC).

1 Introduction

An often overlooked aspect of the knowledge acquisition process is the assimilation of information presented by the domain expert into an existing knowledge base. Consider the typical means by which knowledge bases are currently constructed. This usually involves a series of dialogs between an expert, or experts, in the application domain and a knowledge engineer familiar with the target expert system. The knowledge engineer's task is the modification of the expert system's knowledge base so as to reflect the domain expert's knowledge. To a large extent, this knowledge acquisition task may be viewed as a recognition problem. All of the problems facing other recognition systems are present here as well, including: noisy data (i.e., incomplete or inaccurate information), ambiguous interpretations, and the need to produce intermediate results before all the data is available. Thus, a significant portion of this interactive knowledge acquisition task is a matching problem: How does the expert's description of the domain correlate with the description contained in the knowledge base? How should the knowledge base be modified based on new information from the expert? What should be done when the expert's description differs from the existing one?

K_{AC}^n is a system that we have developed that implements this knowledge assimilation approach to knowledge acquisition. It was developed to assist in the construction of knowledge bases for the POISE [CLLH82] intelligent interface system. These knowledge bases used a frame-like representation, described more fully in [CL84,Lef87] to describe *tasks*, *objects* and *relationships* in the application domain. POISE's initial knowledge bases, for the office automation and software engineering domains, were created by hand from interviews between a knowledge engineer and the appropriate domain experts. Transcriptions of these interviews were examined and the results served as the basis of the K_{AC}^n system.

It is important to note that the goal of the domain expert was *not* to modify POISE's knowledge base; this was the knowledge engineer's role. The expert simply presented the domain information, e.g., descriptions of tasks, objects, etc., and responded to questions and comments from the knowledge engineer. The burden of assimilating the information, that is,

<p>EVENT TAKE-A-TRIP-AND-GET-PAID</p> <p><i>STEPS: (TAKE-A-TRIP GET-REIMBURSED)</i></p> <p><i>TEMPORAL-RELATIONSHIPS:</i></p> <p><i>((TAKE-A-TRIP before GET-REIMBURSED))</i></p> <p><i>CONSTRAINTS: (...)</i></p> <p><i>ATTRIBUTES: ((TRAVELER ...) (COST ...) (DESTINATION ...))</i></p>
--

Figure 1: Knowledge Base Event Description

recognizing where it fit into the existing knowledge base and what additions or modifications were needed, was not placed upon the domain expert. (Contrast this to approaches such as [EEMT86,GWP85,KNM85].) By modeling the knowledge engineer's role in this task, KⁿAc attempts to provide this same support.

Consider the opening portion of a discourse in which the expert, the principal clerk of an academic department, is describing the procedure for being reimbursed for business-related travel expenses.

"O.K. — on travel. The proper way of doing it, if it's out of state, is that a travel authorization should be issued before the trip."

From this information one can conclude that some unnamed task consists of two temporally ordered steps. However, it is not clear what modifications need be made to the knowledge base to reflect this information.

If the knowledge base is examined (prior to this interview), a description of this reimbursement process will be found (see Figure 1). In this simplified view of the task, which knows nothing about a "travel authorization", the traveler simply goes on a trip and gets reimbursed. Though the knowledge engineer may realize that the clerk and this description are describing the same task, it is not readily apparent from the two descriptions. Matching such descriptions, and recognizing the implied modifications, are central to the assimilation process.

To accomplish this, K^{nAc} was required to perform two basic tasks: 1) recognizing where the expert's information fits into the existing knowledge base, and 2) appropriately modifying the existing knowledge so that it reflects the expert's view of the domain. Determining where the expert's information fits into the existing knowledge requires that the new information be matched against the existing information. To avoid matching the new information against the entire (existing) knowledge base, the most likely candidate matches must be selected. Furthermore, since the goal of a knowledge acquisition discourse is the modification of the knowledge base, exact matches between the new and the existing information are not always expected.

Thus, the procedure for matching the expert's entity descriptions with those already in the knowledge base must be specialized for knowledge acquisition. K^{nAc} 's matching and match evaluation procedures are described in Section 3. Discrepancies between these descriptions may imply needed modifications and need not degrade a match, especially if the discrepancies (or the implied modifications) can be predicted. Anticipated modifications, or *expectations*, arise from an understanding of the knowledge acquisition process. They can be derived from the state of the existing knowledge base, from cues in the discourse, from previous modifications to entity descriptions, or from the state of the knowledge acquisition task. The generation and management of these expectations is described in Section 4. Finally, the status of this work and its contributions to the knowledge acquisition task are summarized in Section 5.

2 The K^{nAc} System

In this section, the basic architecture and functionality of the K^{nAc} system¹ is presented. During each cycle of the K^{nAc} system, descriptions of domain entities are accepted from the user (1) and compared with entities in the existing knowledge base (2). (Figure 4 contains a portion of this knowledge base.) These candidate entities (3) are selected based on K^{nAc} 's

¹Figure 2 contains the architecture of the knac system. The parenthesized numbers in this paragraph (e.g., (1)) refer to this figure.

expectations of changes to the knowledge base. The comparisons (4) are evaluated both in terms of how well they match and the extent to which the differences between them were expected (5) within the context of the match. Once the best matches are selected, the implied modifications (6) are made to the existing entity knowledge base (7), after being verified with the user (8), if necessary. Expectations of further modifications are generated from a variety of sources, including the information obtained from the discourse (9), the state of entities in the knowledge base (10), previously made modifications (11) and the state of the acquisition process.

The descriptions obtained from the expert must be presented to the matcher in the knowledge base's representation language. The purpose of the *discourse manager / user interface* is twofold: to permit a more "user-friendly" specification (e.g., natural language, graphics, menus, etc.) of these descriptions, and to provide Kⁿ_{Ac} with any available cues as to the state of the discourse. Currently, the natural language protocols are translated, by hand, into the system's representation language. Discourse cues are assumed to be minimal, such as "topic" information.

Thus, the discourse fragment presented in Section 1 translates, approximately, into the structures shown in Figure 3. These structures may then be compared with selected entities from the existing knowledge base.

To avoid having to examine the entire knowledge base in order to assimilate the new information, entities that are most likely to be modified are selected as candidate matches. Thus, if there exists an expectation of some modification to a given entity description, that entity is compared to the new information from the expert. The way in which these modifications are anticipated will become clearer in Section 4. Initially, the only expectations available are based on the discourse cue recognizing that TRAVEL is a topic of interest. Hence, the entities semantically close to TRAVEL in the knowledge base are selected as candidate matches. These entities include TAKE-A-TRIP-AND-GET-PAID, shown in Figure 1.

The system then compares the expert's descriptions with the selected match candidates. The matching process, described more fully in Section 3, determines the similarities and dif-

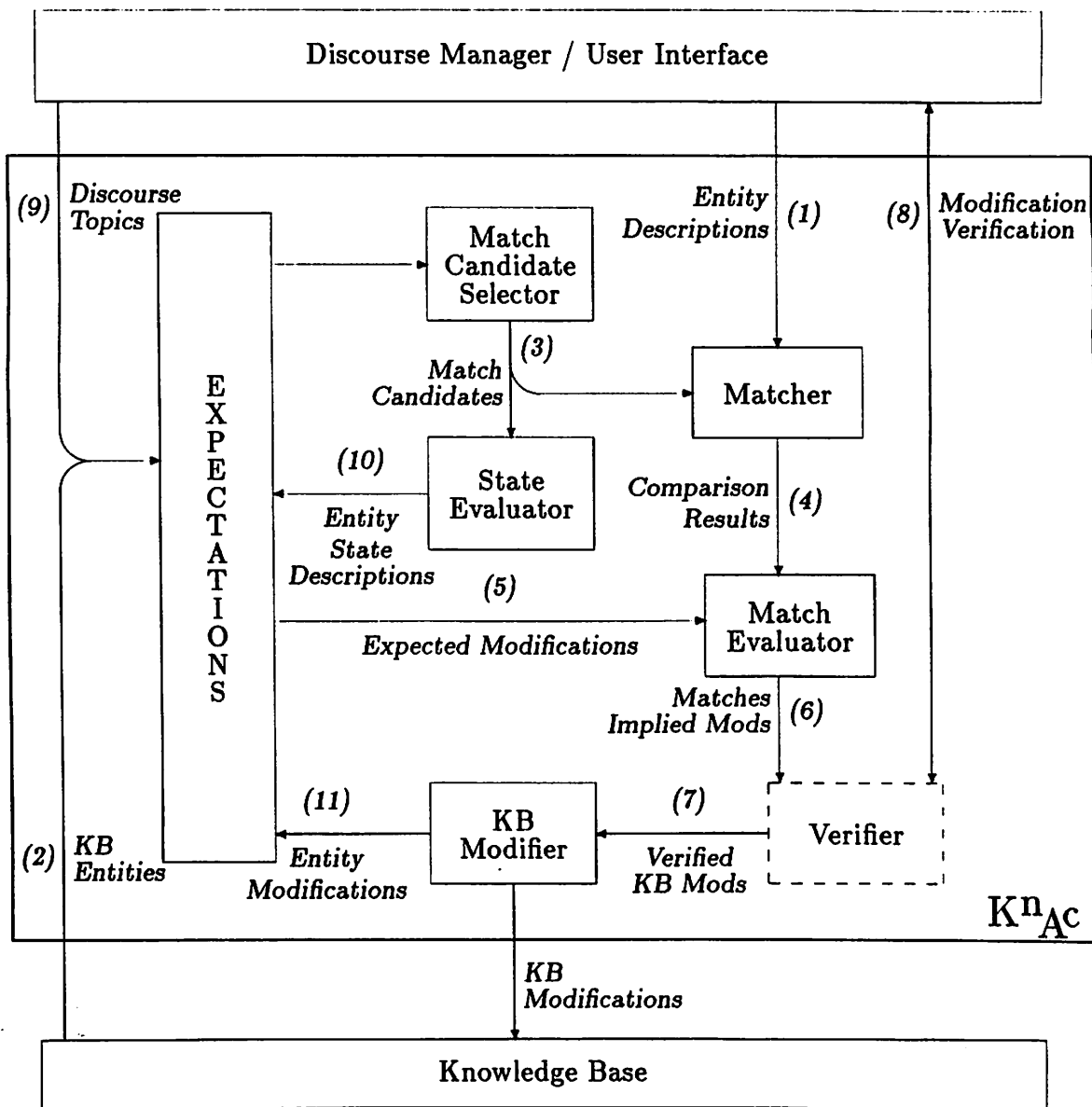


Figure 2: The KⁿAc System Architecture

<p>EVENT EVENT-1</p> <p><i>STEPS:</i> (ISSUE-TRAVEL-AUTHORIZATION TAKE-A-TRIP)</p> <p><i>TEMPORAL-RELATIONSHIPS:</i></p> <p>((ISSUE-TRAVEL-AUTHORIZATION <i>before</i> TAKE-A-TRIP))</p> <p><i>CONSTRAINTS:</i> ((DESTINATION <i>outside-of</i> STATE))</p> <p><i>ATTRIBUTES:</i> ((TRAVELER ...) (DESTINATION ...))</p>

EVENT ISSUE-TRAVEL-AUTHORIZATION

EVENT TAKE-A-TRIP

OBJECT TRAVEL-AUTHORIZATION

Figure 3: Discourse Manager Output

ferences between a pair of entity descriptions. The results of these comparisons are then evaluated in order to select the best match for each of the expert's entity descriptions. Section 3.2 describes the evaluation process, which rates the match results on a field-by-field basis and combines these ratings to produce an overall rating for each match.

For example, when the task described by the expert, i.e., EVENT-1, is compared with the existing task description TAKE-A-TRIP-AND-GET-PAID, the contents of each field of these structures (e.g., *parts*, *generalizations*, *temporal-relationships*, *pre-* and *post-conditions* etc.) are compared. In the *steps* field, they have one entity in common (TAKE-A-TRIP) and each has one entity not found in the other (ISSUE-TRAVEL-AUTHORIZATION in EVENT-1 and GET-REIMBURSED in TAKE-A-TRIP-AND-GET-PAID). They have several *attributes* in common (TRAVELER and DESTINATION). Their *temporal-relationships* are mutually consistent, though different. Both entities are *specializations* of EVENT.

The ratings for these field matches is shown in Figure 5. Remember, these ratings reflect not only the degree to which the fields match, but more importantly (from the point of view of knowledge acquisition) the likelihood of the modifications, in future parts of the dialog, required to make them match.

From these ratings, the best matches are selected. If there is significant ambiguity, the

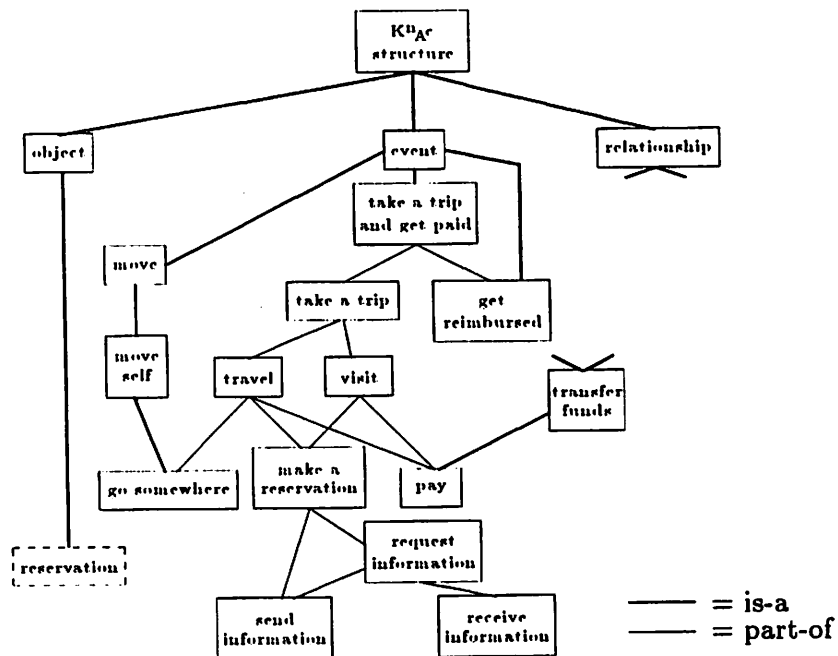


Figure 4: A portion of the knowledge base

matches are verified with the expert. If there are no acceptably good matches for one of the expert's entity descriptions, a new entity is added to the knowledge base.

When the best matches have been selected, the differences between the expert's description and the existing one are used to modify the knowledge base. For instance, the "extra" step in EVENT-1, i.e., ISSUE-TRAVEL-AUTHORIZATION, is added as a step of TAKE-A-TRIP-AND-GET-PAID. The extent to which this modification requires confirmation from the expert depends on the level of autonomy granted to the system. At various levels, all such modifica-

EVENT-1 vs. TAKE-A-TRIP-AND-GET-PAID

Field	Rating
generalizations	1.000
parts	0.133
attribute-names	0.055
constraints	0.007
Match Rating	0.299

Figure 5: Ratings for field matches

tions could be verified with the expert or only unexpected ones or only deletions, etc.

Once the knowledge base has been modified, new expectations are generated to be used in interpreting the next “discourse frame”. The generation and management of these expectations is described in Section 4.

3 Matching for Knowledge Acquisition

In order to match entity descriptions provided by the domain expert to those already known to the system, K^{nAc} must be able to compare these structures and evaluate the results. This matching process, while in some ways similar to that found in most recognition/interpretation systems, displays certain characteristics unique to knowledge acquisition. In particular, since the goal of a knowledge acquisition dialog is the modification of the knowledge base, the information provided by the domain expert should not completely match the existing entity descriptions. The matching process must be modified so as to be able to recognize and, where possible, anticipate these discrepancies. The matching techniques presented in this section make these discrepancies explicit; the evaluation of these match results, described in Section 3.2, incorporates the extent to which these discrepancies were anticipated into its rating procedure.

3.1 Matching Entity Descriptions

POISE’s knowledge base is represented in a frame-based language, similar to that used by systems like Knowledge Craft®[WF83,KC86]. Within these frames, each field may be considered to be one of two types of structures: *sets of elements* such as *steps* and *generalizations* of an event, or *collections of constraints* such as the *temporal relationships* and the *pre- and post-conditions*. K^{nAc} contains matching techniques for both of these types of structures.

3.1.1 Set Matching

Determining how well two sets of elements match is not difficult; neither is determining how “different” they are (e.g., see [Tve77]). For knowledge acquisition purposes, however, the relevant question is “How likely is it that they can be modified so as to match?” To determine this, K^{NAc} examines not only the elements in each (set) field of the knowledge structure, but also makes use of information *about* that field (i.e., *meta-information* or *facets*). In particular, information about the size of each field and the range of the elements in that field permit K^{NAc} to calculate the probability that the *extra* elements in one of the structures will be added to the other.

Consider the comparison of the *steps* of EVENT-1 and TAKE-A-TRIP-AND-GET-PAID. A typical measure of similarity is:

$$\text{match-rating} = \begin{cases} 1 & \text{if } \text{Set}_1 = \text{Set}_2 = \emptyset \\ \frac{|\text{Set}_1 \cap \text{Set}_2|}{|\text{Set}_1 \cup \text{Set}_2|} & \text{otherwise.} \end{cases}$$

With one step out of the three unique steps between them in common, the similarity of these fields would be 1/3. How likely is it, however, that the “extra” step in the expert’s description (i.e., ISSUE-TRAVEL-AUTHORIZATION) will be added to the existing description? Without requiring a deep understanding of domain-specific semantics, some additional information can still be used. If events usually have few steps (as specified by the meta-information about the *step* field of EVENT), the addition of this particular step is less likely than if there are many more steps still to be added. Similarly, if a step is going to be added, the range of possible steps will affect the probability of the desired one being added. This range is determined by combining the “type restriction” meta-information about the slot (i.e., a *step* of an EVENT must be an EVENT) with the existing knowledge base (i.e., the number of EVENT descriptions known to the system). The derivation of these ratings is fully described in [Lef87].

3.1.2 Constraint Matching

Although comparing (or combining) arbitrarily constrained sets of entities is a difficult problem often requiring substantial domain knowledge, K_{AC}^n compares sets of constraints, pairwise related by a common relation, in a purely mechanical fashion.² This section describes a mechanism for comparing such constrained sets; the mechanism is independent of the particular relation, requiring only a description of its algebraic properties, i.e., whether or not the relation is *reflexive*, *symmetric* and/or *transitive*.

First, any implicit constraints are made explicit by propagating the specified constraints using the relation's algebraic properties. The temporal relation *before*, for example, is only transitive; if the constraints (*A before B*) and (*B before C*) were specified, then (*A before C*) could be deduced. When the constraints are thus propagated, inconsistencies may be detected by check the results for any of the prohibited properties of the relation (i.e., non-reflexive, non-symmetric and non-transitive). If each set of constraints is internally consistent, the two sets may be merged and re-propagated, and this combined set of constraints may be checked for consistency.

If two sets of constraints are mutually consistent, a measure of their similarity may be obtained by determining the changes required to make them equivalent. Simply adding each set of constraints to the other would accomplish this, but this may add more information than is necessary. For instance, if the first set of constraints contained (*A before B*) and (*A before C*) and the second sets contained (*B before C*), then only (*A before B*) need be added to the second set. Requiring the addition of both constraints from the first set would imply a larger discrepancy between the sets than actually exists.

Obtaining the minimal set of of constraints that need to be added is not a trivial problem. A new approach to generating these sets, called "opensures"³, based on their algebraic

²The current system checks each relation separately; it does not handle interaction between different relations, though it is able to combine relations with their inverses. For instance, *before* and *after* constraints are handled together.

³i.e., the inverse of "closures"

properties, is presented in [Lef87].

3.2 Match Evaluation

After comparing each of the entity descriptions provided by the domain expert against those candidate entities selected from the existing knowledge base, K_{AC}^n evaluates these match results in two passes. First, the likelihood of two entities matching, based on the extent to which they differ and the probability of these differences being corrected, is determined as described above. This “degree of fit” is a relatively simple and inexpensive means of pruning the set of possible matches.

The second pass takes into account the context in which the comparison is being made. In addition to *how* the descriptions differ, it considers whether these differences are expected in a particular situation. The extent to which the modifications implied by the differences between the structures are expected serves as a “context-dependent” measure of the match.

Consider the addition of the step ISSUE-TRAVEL-AUTHORIZATION to the description of TAKE-A-TRIP-AND-GET-PAID. The addition of such a step could have been foreseen for several reasons. First, since there were fewer steps in the existing description than are typically found in events, adding another step was reasonable. More importantly, upon examining the existing description to see if it was consistent and complete, it was discovered that a *precondition* of the step GET-REIMBURSED, describing the traveler as the recipient of funds, could not be satisfied by the only earlier step in the task (i.e., TAKE-A-TRIP). This further supported the addition of another step (occurring before GET-REIMBURSED) to the task. The way in which K_{AC}^n anticipates these modifications is described in the following section.

4 Anticipating Modifications

K_{AC}^n provides a context in which to interpret information provided by a domain expert by anticipating modifications to an existing knowledge base. These anticipated modifications, or

expectations, are derived from K_{AC}^n 's heuristic information about the knowledge acquisition process. This section describes how these expectations are generated, how they are used to provide a context in which matches may be evaluated, and how they ranked and managed.

4.1 Generating Expectations

K_{AC}^n contains a body of heuristics about the knowledge acquisition process. These heuristics, obtained through the analysis of several knowledge acquisition dialogs, allow K_{AC}^n to anticipate modifications to an existing knowledge base. These heuristics may be divided into four categories. The first group is based on the state of the knowledge, both that already contained in the knowledge base and new information provided by the expert. The second category depends on modifications previously made to the existing knowledge base. The third set makes use of a model of the discourse process, while the final set incorporates knowledge about teaching and learning strategies. Since it is expected that collection of heuristics will be modified, both as a result of improved understanding of the knowledge acquisition process and through the addition of domain specific heuristics, K_{AC}^n allows heuristics to be added (or removed) in a straight-forward way.

Consider a simple heuristic based on the state of an entity description:

Heuristic S2: *Fields with too few components will be augmented.*

This heuristic states that if information is detected to be missing, the addition of that information may be expected. Missing information may be detected in various ways. One simple approach compares the number of entries in a field of a knowledge structure with the field's expected cardinality. If the field is determined to contain too few values, additional values (of the appropriate type) will be expected. The expected field size may come, in order of specificity, from meta-information about a given field of a given entity, via inheritance from a generalization of the entity, from the default information for the type of the entity, or from an overall field default size. This size information may be static or determined dynamically by the system. An expectation generated by this heuristic is:

Exp146: Expecting (certainty 0.360):

MOD: ADD ?New-part<is-a-knac-structure-p> to the

Parts field of Take_a_trip_and_get_paid

Derived from Take_a_trip_and_get_paid and H_S2.

Other heuristics based on the state of the knowledge exploit references to unknown entities, unsatisfied preconditions, and range/value conflicts to anticipate changes.

Changes to the knowledge base may imply additional modifications. For instance, two heuristics that are triggered when a new entity description is added are:

Heuristic M1: *Detailed information usually follows the introduction of a new entity.*

Heuristic M2: *Context information usually follows the introduction of a new entity.*

Additions of to specific fields of entity descriptions (e.g., attributes, steps, constraints, etc.) form the basis of other modification heuristics.

Cues from the discourse manager, such as the topic of discourse or recently referred to entities, are the key to K^{NAC} 's discourse heuristics. Because the current system lacks a sophisticated discourse manager, the only discourse heuristics being used are:

Heuristic D1: *Entities close to specified topics are likely to be referenced or modified.*

Heuristic D2: *Referenced entities are likely to be modified or referenced again.*

4.2 Managing Expectations

As the number of expected modifications to the knowledge base grows, K^{NAC} 's ability to use the expectations to focus its attention diminishes. Thus, a means of selecting the most likely expectations (for a given time) is required. This is accomplished by assigning a rating to each expectation and pruning the set of heuristics based on this rating.

Each heuristic is responsible for determining a rating for each expectation it generates. This rating depends on the quality of the data and the specificity of the heuristic. In addition, each heuristic assigns a function to its expectations that specify how these ratings will change with time. For instance, certain expectations are important when they are created but become

less valid with the passage of time; others become more critical as time passes. Some become more (or less) significant based on some state of the knowledge base; others are always valid. The functions currently available in K^{nAc} are: *fade*, *increase*, *until*, *while*, *after*, *for*, *always* and *never*.

5 Status and Conclusions

In this paper we have examined an often overlooked aspect of the knowledge acquisition process: the assimilation of information presented by a domain expert into an existing knowledge base. Though a fundamental part of the current conventional knowledge base development process, the issue of automatically locating and appropriately modifying existing knowledge to conform to the domain experts descriptions has received little, if any, emphasis. Most current knowledge acquisition tools place this burden on the domain expert, forcing him to take over part of the knowledge engineer's task. By automating this assimilation process, the K^{nAc} system better insulates its user from the knowledge base.

K^{nAc} accomplishes this assimilation by: 1) comparing entity descriptions provided by the domain expert with existing knowledge base descriptions, 2) evaluating these matches in the context of the knowledge acquisition discourse, 3) making the modifications to the existing descriptions implied by the expert's information, and 4) generating (and managing) expectations of further changes to the knowledge base.

This work has developed several generic matching (and match evaluation) techniques especially adapted for knowledge acquisition. They shift the focus of matching from examining how closely two entities match to exploring the likelihood of their being modified so as to match. This is accomplished by matching procedures (for sets of entities and collections of constraints) which determine differences as well as similarities, an evaluation technique which explores the probability of the modifications required to make entities match and the degree to which these modifications are expected, and a means of anticipating modifications to the knowledge based on heuristic information about the knowledge acquisition process.

The KⁿAc system is implemented in Common Lisp running on a TI Explorer®. Its current use is still experimental, though it has been able to assimilate a 20 step dialog on the travel reimbursement process, correctly constructing an internal representation of the plan in the POISE knowledge base. A complete description of the implementation and the sample dialog can be found in [Lef87].

References

- [CL84] W. Bruce Croft and Lawrence S. Lefkowitz. Task support in an office system. *ACM Transactions on Office Information Systems*, July 1984.
- [CLLH82] W. Bruce Croft, Lawrence S. Lefkowitz, Victor R. Lesser, and Karen Huff. POISE: An intelligent assistant for profession based systems. In *Proceedings of the Conference on Artificial Intelligence*, Oakland University, Michigan, April 1982.
- [EEMT86] Larry Eshelman, Damien Ehret, John McDermott, and Ming Tan. MOLE: A tenacious knowledge acquisition tool. In *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, Canada, 1986.
- [GWP85] Allen Ginsberg, Sholom Weiss, and Peter Politakis. SEEK2: A generalized approach to automatic knowledge base refinement. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 367–374, 1985.
- [KC86] *Knowledge Craft Users Manual*. Carnegie Group Inc., 1986.
- [KNM85] Gary Kahn, Steve Nowlan, and John McDermott. MORE: An intelligent knowledge acquisition tool. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 581–584, 1985.
- [Lef87] Lawrence S. Lefkowitz. *Knowledge Acquisition through Anticipation of Modifications*. PhD thesis, University of Massachusetts, Amherst, MA, 1987.
- [Tve77] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.
- [WF83] M. Wright and Mark S. Fox. *SRL 1.5 User Manual*. Intelligent Systems Laboratory, Carnegie-Mellon University Robotics Institute, 1983.