ABSTRACT

Most expert systems require access to a knowledge base of domain facts. This paper raises issues concerning the granularity of such knowledge bases and the need to update them. In particular, it discusses the use of an inference engine and associated truth maintenance system in constructing and maintaining such knowledge bases and presents an implementation of a prototype system which uses these techniques.

INTRODUCTION

Most expert systems require access to a knowledge base of domain facts. Construction and maintenance of this knowledge base is a significant part of the knowledge engineering task which is to some extent separate from the elicitation of problem-solving rules from a domain expert. Two issues arise. First, there is the matter of granularity of the knowledge base. For the sake of efficiency at problem-solving time, the knowledge base is best implemented at a detailed level in the form of a database, so that rules may simply do database lookups when information is needed, rather than using inference to derive required facts. On the other hand, such a database is tedious to construct by hand and does not necessarily match the granularity of the models used by human experts. Second, frequent updating of the knowledge base may be necessary to reflect changes in the real world. A truth maintenance system [3] may be used in conjunction with an inference engine to construct and maintain the database while insuring its integrity and allowing the user to operate at a high level of abstraction.

In this paper, we discuss these issues in the context of an expert system for assigning courses to faculty. A truth maintenance system has previously been used in assigning courses to faculty [2], but the emphasis was on using truth maintenance in the expert system inference engine, rather than in conjunction with knowledge base construction and maintenance.

KNOWLEDGE BASE GRANULARITY

An expert in a given domain frequently uses rather general attributes and relationships when describing a domain. For example, "The faculty in my department prefer to teach graduate courses." An expert system used to assign courses to faculty ultimately needs this knowledge at a much lower, more specific level. For example, "Professor Smith prefers to teach CS601." In this latter form, we are directly expressing relationships between the fundamental entities (particular faculty and particular courses) in the domain. A knowledge base in which all simple relationships between objects of interest in the world are directly represented is said to be "vivid" [5]. Knowledge expressed in vivid form lends itself to database-like lookup which results in very efficient access to the knowledge base by the rules of an expert system. Thus Etherington et al [4] propose transforming a knowledge base into vivid form where possible, so that the use of inference techniques to extract information from the knowledge base is minimized. This transformation process is called "vivification." In the following sections, we discuss the construction and maintenance of a vivid knowledge base for an expert system that assigns courses to faculty.

KNOWLEDGE BASE CONSTRUCTION

In order to construct the knowledge base for a course assignment expert system, we use vivification rules and an inference engine to transform the initial knowledge base into vivid form. The vivification rules result in the assertion of additional, lower-level facts. Two of the types of vivification rules employed are:

1. If the knowledge base contains an assertion of the form

   \[ p(..., c, ...) \]

   and \( c \) is a disjoint union

   \[ c = c_1 \cup c_2 \cup ... \cup c_n \]

   \[ \text{and } c_i \text{ is a disjoint union} \]

   \[ c_i = c_i \cup c_i' \]

   then use rule 1 to transform the knowledge base into a vivid form.
then assert \( p(..., e, ...) \) for \( 1 \leq i \leq n \).

For example, if the knowledge base contains the assertion

\[ \text{qualified-to-teach}(\text{smith, undergraduate-courses}) \]

then

\[ \text{qualified-to-teach}(\text{smith, lower-division-courses}) \]

and

\[ \text{qualified-to-teach}(\text{smith, upper-division-courses}) \]

would be asserted since \( \text{undergraduate-courses} \) is the disjoint union of \( \text{lower-division-courses} \) and \( \text{upper-division-courses} \).

2. If the knowledge base contains an assertion of the form

\[ p(..., s, ...) \]

where \( s \) is a set, then for each \( x \) such that the knowledge base contains the assertion

\[ \text{member-of}(x, s), \]

assert

\[ p(..., x, ...). \]

For example, if the knowledge base contains the assertion

\[ \text{qualified-to-teach}(\text{smith, ai-courses}) \]

and the assertions

\[ \text{member-of}(\text{cs400, ai-courses}) \]

and

\[ \text{member-of}(\text{cs420, ai-courses}), \]

then

\[ \text{qualified-to-teach}(\text{smith, cs400}) \]

and

\[ \text{qualified-to-teach}(\text{smith, cs420}) \]

would also be asserted.

After the inference engine has completed its work, the database-like facts that are now present in the vivid knowledge base are extracted and used as the underlying database for the expert system. However, the entire knowledge base (consisting of the original knowledge base plus all assertions added during the vivification process) is kept for use in future modifications, so no information is lost.

When reasoning with the database, we assume domain closure [6] for courses and faculty, i.e., we assume that there are no courses or faculty in the world other than those explicitly represented in the database. We also use the closed world assumption [7] for certain predicates such as qualified-to-teach and prefer-to-teach. For example, if

\[ \text{qualified-to-teach}(\text{smith, cs501}) \]

is not present in the database, we assume that Smith is not qualified to teach cs501.

**KNOWLEDGE BASE EVOLUTION**

Over a period of time, it is desirable for the knowledge base to evolve. One reason for this is that the world modeled by the knowledge base is not static. For example, in the course assignment world, people’s qualifications and course preferences may change. The knowledge base must be revised to reflect these changes. Also, after evaluating the quality of problem solutions produced by the expert system, the user may determine that the knowledge base contains incomplete or erroneous information. Thus, the knowledge base will need to be changed occasionally by including new assumptions and retracting incorrect assumptions. When these changes are made, the vivid form of the knowledge base directly used by the expert system must be revised accordingly using the inference engine and truth maintenance system.

**ROLE OF THE TRUTH MAINTENANCE SYSTEM**

The truth maintenance system provides several important capabilities in the revision of the knowledge base and subsequent revision of the vivid knowledge base by the inference engine.

1. Providing explanations: the justifications for a current belief may be examined. Thus the user can determine which high-level facts and transformation rules were used during the vivification process to add a given low-level fact to the knowledge base.
2. Making new assumptions: if it is discovered that there is information missing from the knowledge base, then additional assumptions may be added (and their consequences computed).

3. Retracting assumptions: assumptions which no longer hold may be retracted. The truth maintenance system will also automatically retract all conclusions which depend upon the retracted fact. Thus if a high-level assumption A is retracted, all low-level facts produced during the vivification process which depend upon belief in A will automatically be retracted.

4. Maintaining a cache of inferences: inferences previously made are preserved and need not be repeated. Thus if a high-level assumption A is retracted (resulting in retraction of additional derived facts) and then later re-asserted, the derived facts are automatically reinstated without the necessity of repeating the inference procedure originally used to derive those facts.

5. Detecting contradictions: the truth maintenance system may be used to detect the presence of contradictory facts in the knowledge base. The inference engine is signaled when this condition occurs and a contradiction handler may be invoked to resolve the conflict. The user may be notified of the assumptions which lead to the contradiction and be allowed to choose which assumptions to retract in order to remove the contradiction. For example, suppose one has declared a contradiction condition

contradiction(course-preference-contradiction)

and the knowledge base contains the rule

if prefers(?f, ?c) and
does-not-want-to-teach(?f, ?c)
then
assert(!course-preference-contradiction).

If the knowledge base contains the assertions

prefers(smith, upper-division-courses)

and

does-not-want-to-teach(smith, cs409)

then since cs409 is an upper-division course, during the vivification process, the assertion

prefers(smith, cs409)

will be added to the knowledge base. This raises the contradiction condition course-preference-contradiction. The truth maintenance system can be set up to detect this contradiction and notify the user that the contradiction arises from the joint assumptions that smith prefers upper-division courses but does not want to teach cs409. The user may then choose to retract one of the two assumptions in order to remove the conflict. The truth maintenance system reports the top-level justification

prefers(smith, upper-division-courses)

for the derived fact

prefers(smith, cs409).

Note that monotonic TMSs will not allow the user to retract the derived fact that Smith prefers cs409 without retracting its justification.

These capabilities make it possible to provide the user a convenient high-level interface to the knowledge base, so that changes can be made at the level of the user world model while maintaining the integrity of the low-level database.

IMPLEMENTATION OF THE EXPERT SYSTEM

The expert system is a separate module that accesses the database extracted from the vivid knowledge base. It starts with a list of all sections of all courses being offered and assigns an instructor to each section if possible. Faculty preferences and qualifications are taken into account. For example, a typical rule used is

if member-of(c, courses) and
offered(c) and
not-assigned(c) and
member-of(f, faculty) and
qualified-to-teach(f, c) and
prefers-to-teach(f, c)
then
assign(f, c)

The hypotheses in the above rule may be tested using simple lookups in the database produced from the vivid knowledge base. For example, if the original knowledge base contains the assertion

prefers-to-teach(smith, ai-courses)
and cs450 is an ai-course, then the vivid knowledge base
(and therefore the database accessed by the expert system)
will contain the fact

\texttt{prefers-to-teach(smith, cs450)}

which was produced by rule 2 mentioned above in the
description of the construction of the vivid knowledge base.
This results in very fast execution time for the expert
system.

The system described above has been implemented in
Common Lisp. The portion of the system used to construct
and maintain the knowledge base incorporates a monotonic
justification-based truth maintenance system and inference
engine written by de Kleer and Forbus [1].

CONCLUSIONS

Experience with the prototype system described in this
paper indicates that use of an inference engine and truth
maintenance system to construct and maintain a vivid
knowledge base which is accessed by an expert system
using simple database lookup is a promising technique
which merits further investigation. Spending execution
time on the initial vivification process results in improved
performance of the expert system since no further inference
is necessary at runtime. Use of a truth maintenance system
is especially important as the knowledge base evolves over
time. Caching inferences avoids repetition of identical
computation, the retraction mechanism allows the user to
remove high-level assumptions with the system automati-
cally taking care of all consequences of the retraction at the
low-level, and the contradiction detection mechanism helps
the user to maintain the integrity of the knowledge base as
changes are made.

Directions for future work include quantitative studies to
determine the trade-off between the improvement in run-
time performance which results from the vivification process
and the increased memory required to maintain the TMS
cache.

REFERENCES

[1] de Kleer, J., and Forbus, K.D. Building Problem
Solvers: Program notes on Truth Maintenance
[2] Dhar, V., and Ranganathan, N. Integer Program-
ming vs. Expert Systems: An Experimental
Comparison. Communications of the ACM 33, 3
(March 1990), pp. 323-336.
Intelligence 12, 3 (1979), pp. 231-272.
[4] Etherington, D.W., Borgida, A., Brachman, R. J.,
Kautz, H. Vivid Knowledge and Tractable Rea-
Eleventh International Joint Conference on Artifi-
cial Intelligence. (Detroit, Michigan, August 20-
Levesque,
H.J. Making
Believers out of Comput-
[6] Reiter, R. Equality and Domain Closure in First-
Order Databases. Journal of the Association for
235-249.
and Data Bases, H. Gallaire and J. Minker, Eds.,