Circuit Minimization Techniques Applied to Knowledge Engineering

Alexander Pasik

Department of Computer Science, Columbia University
New York City, New York 10027

Abstract

Often knowledge engineers encounter situations during the interviewing process in which experts have difficulty expressing the knowledge to be captured. In these situations, the experts cannot readily present their knowledge so that the knowledge engineers can encode it in the chosen formalism (for example, in production rules). During the development of an expert system for underwriting homeowner insurance policies, this situation was occasionally encountered. When the experts could not express their knowledge in chunks suitable for encoding directly in production rules, circuit minimization techniques were used to construct the set of production rules from exhaustive tables of acquired knowledge. The techniques also served to find errors in the acquired knowledge. Circuit minimization techniques, therefore, have been found to provide valuable assistance in the knowledge engineering process, both in the acquisition and verification of knowledge.

Circuit Minimization Techniques Applied to Knowledge Engineering

1. Introduction

Knowledge engineering is the process by which expertise is educed from humans proficient in a given domain and captured in an expert system. Often this expertise is represented in a production system. The production system formalism is a convenient representation for expert systems because of the modularity of production rules and the independence that exists between rules in a given system. Rules represent guidelines which the experts use in solving problems, and working memory contains the details of the particular problem being solved [Rychener 1976]. The working memory of many production systems can also be used to represent certain knowledge in tabular form [Pasik and Schor 1984].

A knowledge engineer must interview experts extensively, not only filtering relevant knowledge, but also deciding on the appropriate representation for different types of knowledge that the expert system will use. The more interesting aspects of the expertise are generally represented in rules. Although many of the rules reflect a straightforward encoding of the expert's knowledge, occasionally the expert will have difficulty expressing decisions in those terms. It is in this case that circuit minimization techniques can be helpful. Also, using these techniques can point out possible errors in the acquired expertise which could be the result of carelessness, misinterpretation, or an actual flaw in the expert knowledge being encoded.

The example used to demonstrate the technique is a small portion of an expert system for underwriting applications for homeowners insurance. Applications are submitted to the expert system which analyzes the information and responds with any reasons for rejection, suspension, or elaboration required.

2. Deriving Rule Sets by Minimization

Many rules capture a single criterion for making a decision. For example, one rule in the example system may state the following:

If the district in which the dwelling is located is unprotected
(far from a fire station),
Then this is a reason for rejecting the application.

However, rules such as this one are only applicable if a single item is sufficient for the decision. Rules are often more complex, involving the interaction of two or more attributes. Nevertheless, if the expert naturally associates these multiple attributes during the interviews, the rules remain easy to construct. An example of this is the rule:

If 1. the insured is a musician,
2. there is a musical instrument as a scheduled item,
Then this is a reason for elaborating on the application to determine if the item is used professionally.

Problems arise when the combination of several attributes affects the decision but there is no obvious relationship among them. Thus the expert does not have preconceived rules about the interaction of these attributes, but makes decisions based on the overall picture that the set of attributes creates.

The technique used for creating the set of rules to implement these decision-making processes is an application of circuit minimization [Mano 1972] to a table of the possible situations. The technique is applicable for determining the rules to solve a small subproblem within domain. The knowledge engineer first educes the expertise which can be easily described by rules. When subproblems are identified in which the expert can no longer express the knowledge in this fashion, the rules can be derived using the method described below.

1. The attributes and their possible values are arranged in a table of all possible configurations.

2. Each situation is presented to the expert for analysis; each situation is thus assigned one of the possible decisions. If the expert can apply some restrictions, each scenario need not be explicitly presented. This amounts to a partial minimization.

3. Each decision is then interpreted as a binary function on the attributes.

4. Each function is minimized. This can be accomplished either by hand (using Karnaugh maps, for instance) or by using existing algorithms which perform the minimization automatically.

5. The resulting functions are translated into production rule representation.

Although the table of possible situations can grow large even when only few attributes are involved, the expert can often supply partial rules which limit the number of scenarios. The minimization often leads to a dramatic reduction in the number of rules required to capture the expertise. Also, the resulting rules can reveal an underlying relationship between seemingly unrelated attributes.

An illustrative example of applying these techniques follows. Five pieces of information on the homeowner insurance application were identified as contributing factors in the underwriter's overall decision, without necessarily having direct consequences when isolated.

- Dwelling is in a questionably protected district. (P)
- Dwelling is between 1 and 2 miles from the shore. (S)
- Occupation of the insured is questionable. (O)
- Coverage requested is regarded as underinsurance. (U)
- Dwelling is of frame construction. (F)

Since each of these attributes was phrased in such a way as to assign a binary value to each, a table of 32 scenarios was created, each one being presented to the expert for a decision. As mentioned earlier, one of four possible results was assigned to each situation: reason to reject, suspend, elaborate, or no reason for any action (that is, approve the application without reservation). The table of scenarios along with the expert's decisions is shown in Figure 1.

P S O U F	Decision
0 0 0 0 0	Approve
0 0 0 0 1	Approve
0 0 0 1 0	Suspend
0 0 0 1 1	Suspend
0 0 1 0 0	Elaborate
0 0 1 0 1	Elaborate
0 0 1 1 0	Suspend
0 0 1 1 1	Suspend
0 1 0 0 0	Approve
0 1 0 0 1	Approve
0 1 0 1 0	Suspend
0 1 0 1 1	Suspend
0 1 1 0 0	Elaborate
0 1 1 0 1	Elaborate
0 1 1 1 0	Suspend
0 1 1 1 1	Suspend
1 0 0 0 0	Approve
1 0 0 0 1	Approve
1 0 0 1 0	Suspend
1 0 0 1 1	Suspend
1 0 1 0 0	Elaborate
1 0 1 0 1	Elaborate
1 0 1 1 0	Suspend
1 0 1 1 1	Suspend
1 1 0 0 0	Elaborate
1 1 0 0 1	Suspend
1 1 0 1 0	Suspend
1 1 0 1 1	Reject
1 1 1 0 0	Elaborate
1 1 1 0 1	Suspend
1 1 1 1 0	Suspend
1 1 1 1 1	Reject

Figure 1. Decisions for each scenario.

Karnaugh maps were drawn for each of the decisions interpreted as functions (see Figure 2). The *reject* function was done first. After that, the *suspend* function was minimized with the *reject* values as *don't cares*. This was legitimate because any reason to reject an application is also enough to suspend it. In general, the functions were minimized in the order such that each one subsumed all previous ones. Hence, the *elaborate* function was minimized with the

reject and *suspend* values as *don't cares*. An application is approved when none of the other three functions are true. The functions which are derived are simple and can be encoded in only four rules corresponding to:

1. *reject* = P∧S∧U∧F,
2. *suspend* = U ∨ (P∧S∧F) creating two rules,
3. *elaborate* = O.

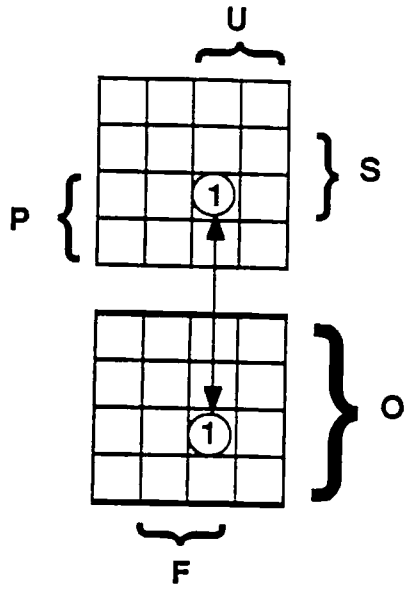
Two of the resulting rules are very simple, depending only on one attribute. That is, if the occupation is questionable, elaboration is required, and if there is suspected underinsurance, the application should be suspended. The rule corresponding to the *reject* function is the most complex one, involving four condition elements.

If
1. the protection is questionable,
2. the distance from the shore is between 1 and 2 miles,
3. the coverage is considered underinsurance,
4. the construction is frame,

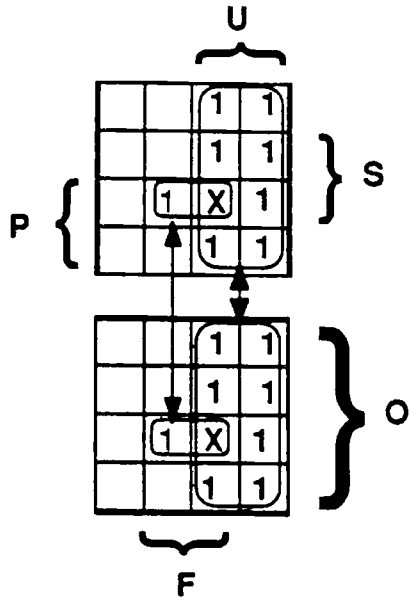
Then there is a reason for rejecting the application.

In this example, of course, the technique was particularly applicable because the attributes were easily expressed in terms of binary values.

As a result of this analysis, only a small set of rules were necessary to encode all the possible outcomes based on the five features. In addition to this advantage, these four rules represent relationships between outcomes and features which were implicit and hidden in the expert's reasoning. The rules were shown to the expert who then reacted favorably to the optimized knowledge.



(a) Reject Function.



(b) Suspend Function.

Figure 2. Karnaugh Maps.

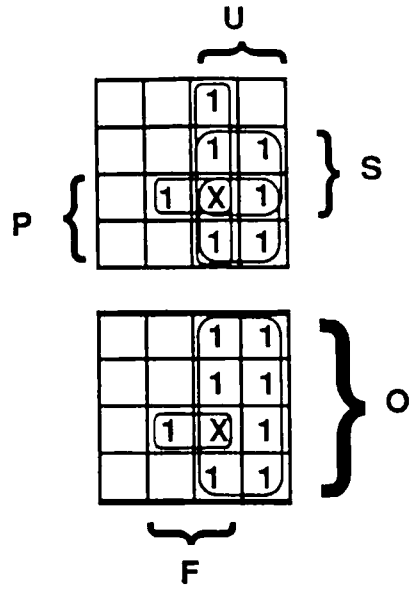


Figure 3. Five Terms Result from the Faulty Map.

3. Finding Errors in Acquired Knowledge

Attempting to perform minimization as described can illuminate possible flaws in the acquired knowledge. On minimizing the functions, it may become clear that a subset of the scenarios are suspicious. These cases can then be brought to the attention of the expert for further consideration. The criterion for determining if a scenario is suspicious is based on the hypothesis that expert knowledge is often regular. If some irregularity is noticed, even if it is correct it should be verified.

As demonstrated in the following example, a single scenario with an incorrect outcome specified can lead to a greater number of rules being required to implement the function. In Figure 3, a single scenario's outcome was incorrectly given as *elaborate* rather than *suspend*. This generated the *suspend* function:

$$\textit{suspend} = (U \wedge F) \vee (U \wedge S) \vee (U \wedge P) \vee (U \wedge O) \vee (P \wedge S \wedge F)$$

The result was that five rules would be necessary (one for each logical disjunction) for the *suspend* function instead of the two rules required in the previous case. This was brought to the expert's attention, who identified the error in the acquired knowledge. Even if the scenario

was correct, it would have indicated a greater importance on one of the factors, which could lead to an investigation of the relative weights of the attributes.

Techniques such as this have been researched in an effort to provide aids for the knowledge acquisition process. A prototype expert system environment was developed in which individual inferences only referenced single attributes [Pasik *et al.* 1985]. In this situation, the circuit minimization technique could be performed automatically for detection of possible errors in the knowledge base. The prospect of applying this mechanism to a more general production system environment such as OPS5 [Forgy 1981] is more difficult. The flexible patterns and actions must be translated into combinations of binary attributes and outcomes before the minimization and possible error detection can occur.

4. Conclusion

Circuit minimization can be applied to knowledge engineering tasks. Although the technique may have limited applicability, it can be used in conjunction with other knowledge engineering techniques as an additional mechanism for understanding acquired knowledge and optimizing rule sets.

The technique requires that the attributes being considered be presented in terms of binary situations. Also, the set of outcomes should be clearly defined. In these situations, the knowledge engineer can use circuit minimization to derive concise rule sets from exhaustive enumerations of scenarios, as well as find possible flaws in the acquired knowledge which can then be referred back to the experts.

Experts can often explain their decision-making processes in ways that knowledge engineers can readily encode in production rules. Occasionally, however, the expert cannot identify how a decision is made even though the actual decision can be determined. In these cases, the expert can provide partial rules to limit the scenarios that need be considered, and then the remaining situations can be exhaustively enumerated. This list can then be minimized, yielding a set of rules which not only solve the problem but also represent new relationships among the attributes and between attributes and outcomes.

This technique should not be considered as a general purpose mechanism for knowledge engineering; the number of scenarios describing and entire domain is prohibitive (and likely infinite). Nevertheless, once the majority of rules have been derived by straightforward interviews, the remaining situations which defy easy elucidation can be analyzed using the circuit minimization technique.

References

Forgy C.L. (1981) OPS5 User's Manual. Technical Report. Department of Computer Science, Carnegie-Mellon University.

Mano M. (1972) *Computer Logic Design.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Pasik A., Christensen J., Gordin D., Stancato-Pasik A., and Stolfo S.J. (1985) Explanation and Acquisition in Expert Systems Using Support Knowledge. Technical Report, Department of Computer Science, Columbia University.

Pasik A. and Schor M.I. (1984) Table-driven Rules in Expert Systems. *SIGART Newsletter 87*: 31-33.

Rychener M. (1976) *Production Systems as a Programming Language for Artificial Intelligence.* Ph.D. Thesis, Carnegie-Mellon University.