

AN INFERENCE NET COMPILER
FOR THE PROSPECTOR RULE-BASED CONSULTATION SYSTEM

Kurt Konolige

Artificial Intelligence Center
SRI International
Menlo Park, California 94303

Abstract*

A compiler for the production system used in a rule-based consultation system is described. This compiler produces machine code which performs antecedent or data-driven inference in an efficient manner. Timing results and some limitations of the compiler are discussed.

1. Introduction

Typically a rule-based consultation system (which we shall call an expert system)** is used in an interactive mode as a consultant: a user volunteers information from which conclusions are drawn, or a goal is pursued by the system and appropriate questions asked, or there is a combination of both kinds of system interaction. The end result is that the system presents a conclusion about some hypothesis to the user, based on information provided by him.

By contrast, a noninteractive mode for an expert system can also be useful. Noninteractive situations arise when information is presented to the system in large doses, rather than parceled out piecemeal by an interactive user. Typical applications include data-base screening, in which the system must process a large number of records; handling map data, in which the system must be run on each of a large number of cells covering the map; and sensitivity analysis of the system, in which answers provided by the user are changed incrementally. These noninteractive applications are characterized by the fact that the type of input is known before the system is run, but the exact values are not. This paper describes a method for compiling a certain type of expert system into an efficient linear form under these conditions.

2. Expert Systems

Many expert systems use a production system to encode the knowledge gleaned from experts [1, 4]. Such a production system consists of a

data base of assertions; a set of production rules of the form $\langle \text{antecedent} \rangle \Rightarrow \langle \text{consequent} \rangle$; a control program that decides on an inference strategy; and an inference mechanism to propagate the effects of input data. The last two functions are often integrated into a single program called the production system interpreter.

The compilation technique described here was developed for a restricted class of probabilistic production systems* with the following properties:

1. No variables are bound in the production rules.
2. There are no inference loops (e.g., $A \Rightarrow B, B \Rightarrow C, C \Rightarrow A$).

Given these restrictions, a set of productions can be converted into an inference net with no loops: a portion of this net is illustrated in Figure 1.

Productions are indicated by arrowed lines. At the leaf nodes of the net are input assertions to which the user (or some other input source) can assign probabilities. These probabilities are propagated from the leaf nodes through intermediate hypotheses to the root hypotheses.

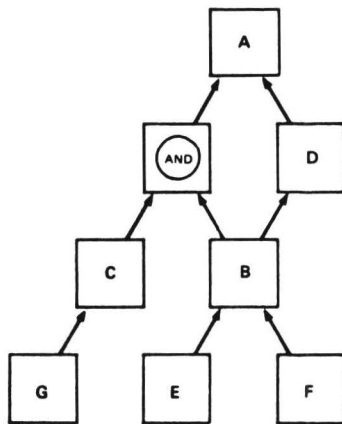
3. The Compilation Technique

A compilation technique for inference nets was developed to take advantage of the noninteractive nature of the input. The

* This research was funded in part by USGS Contract No. 14-08-001-15985 and NSF Grant No. AER77-04499.

** Examples of such systems are [2, 6].

* Probabilistic production systems allow probabilistic inference by attaching probabilities rather than a binary truth value to assertions, and allowing production rules to have varying strengths in updating an assertion. See, e.g., [2], [6].



Productions:

B & C => A
 D => A
 B => D
 E => B
 F => B
 G => C

FIGURE 1 A SIMPLE INFERENCE NET

assumption of a noninteractive mode has two parts:

1. "Batch" assumption: all input to the leaf nodes of the inference net is available at the start of the run.
2. Explicit goal assumption: the hypotheses for which results are to be computed are known at the start of the run.

The consequences of these assumptions are exploited in the compilation technique as follows:

Elimination of Control Strategy Overhead -- Starting from a set of root goal nodes, the control program normally searches down a path in the inference net to an input leaf node; it then asks the user for a probability for that node. The control strategy is typically goal-directed; it tries to establish a query order for the leaf nodes that is both efficient for hypothesis discrimination and meaningful to the user.

Under the noninteractive assumption of an explicit goal set, it is possible to eliminate the goal-directed strategy entirely. Since the leaf nodes for which the probabilities are known are given in advance, there is no need to search for them in the noninteractive mode. Inferencing becomes data-directed from the input leaf nodes, rather than goal-directed from the hypothesis nodes.

Propagation -- In the interactive mode, the consequences of new information on a leaf node are propagated immediately throughout the net. Obviously, the probability for a hypothesis node may be calculated many times through this procedure; for example, node A may be updated a total of four times -- once each time the

effects of changing the probabilities of nodes E, F, C, and D are propagated.

If all probabilities on leaf nodes are available at the beginning of propagation, it is possible to devise a propagation scheme in which the probability of each hypothesis node is computed only once, in a single bottom-up pass. Because it has no loops, the inference net defines a partial ordering of nodes from the leaf nodes to the root node. In this ordering, for example, the probability of node B must be computed before node C, and both of them before A. The compilation technique converts the partial ordering defined by the net into a linear ordering. In the compiled net, probabilities on hypothesis nodes are computed in this linear order, and the results saved in temporary cells for later calculations.

The input to a compiler embodying this compilation technique is an inference net of the type described above. Its output is a linear (i.e., loop-free) code sequence that calculates the probability of the root hypothesis, given a data set for the leaf nodes. This sequence simulates the action of the net interpreter on the net in a noninteractive mode. The amount of time taken to execute the code can be estimated as:

$$C_L \cdot N_L + C_P \cdot N_P \quad (1)$$

where the C's are constant for a given machine executing the code, and the N's are defined as:

N_L = number of logical productions^{*}

N_P = the number of probabilistic productions

4. Timing Results for a Compiler Implementation

A compiler for the inference nets of the PROSPECTOR [2] expert system was implemented to test the practicality of the compilation technique. It compiles PROSPECTOR nets into a linear machine code sequence. A short initializing sequence is also included to enable linking of the compiled inference net to any set of inputs; thus the inference net need only be recompiled if the net structure has changed.

The implementation of the compiler follows in a straightforward way from the compilation techniques of the last section. Total compilation time is proportional to the number of productions in the net; a nominal amount of time is spent processing each production.

^{*} Logical productions are logical combinations of assertions, e.g., the AND node in Figure 1.

The compiler was tested using a medium-scale PROSPECTOR inference net comprising some 94 nodes and 105 productions (59 of them were logical productions). This inference net encoded expert knowledge about geological characteristics of a favorable exploration drilling site for a type of copper mineral deposit [3].

In a typical application, data for 13 of the input nodes was extracted from geological maps of an area. The area was partitioned by a 128 x 128 grid into 16,384 map cells, producing input data for 16,384 runs of the compiled model. The resulting probabilities on the root hypothesis were recombined to yield Figure 2.

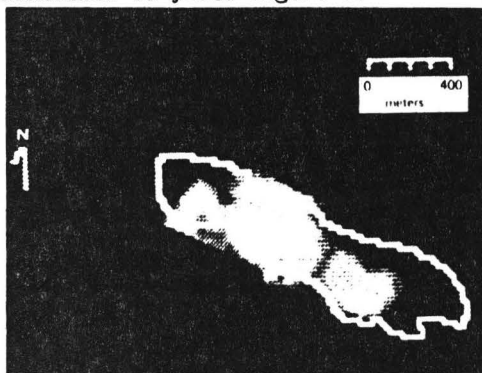


FIGURE 2 RESULTS OF 16,384 RUNS OF A COMPILED INFERENCE NET USING MAP DATA

Note: The brightness of each individual pixel indicates the favorability of the root hypothesis for that pixel. The bright line is the actual ore body outline. The area is Island Copper in British Columbia, Canada [3].

Timings for this run are given in Table 1. The estimated running time given by equation 1 is also listed, and agrees well with the actual time. These timings indicate typical speeds that we have achieved under the noninteractive assumptions.

For comparison, the inference net interpreter was run using the data set from a single pixel. Timings for a typical interpreted run are also given in Table 1. For this specialized application the compiled version of the net is about four orders of magnitude faster than the net interpreter in the test case.

* C_L and C_P were estimated to be 6.0 us and 44 us, respectively, by looking up machine instruction times in [5]. Experience with many runs indicates that equation 1 is a reliable estimator of running time.

Table 1

CPU Timings	Control	Propagation	Total
Interpreted	17 s	13 s	30 s
Compiled, actual	0	3.1 ms	3.1 ms
Compiled, estimated	0	2.4 ms	2.4 ms

5. Conclusions

A compiler for PROSPECTOR-type inference nets has been described. It generates code that is approximately four orders of magnitude faster than the net interpreter. Its efficiency and practicality for noninteractive use has been demonstrated in an application using digitized map data as input.

As currently conceived and implemented, the compiler suffers from the drawback of not being able to handle production rules containing variables. A compiler that produces linear code is probably inadequate for dealing with arbitrary variables in production rules. However, the use of variables in the productions used by such expert systems as PROSPECTOR and MYCIN is very limited. We are exploring restrictions on the use of variables in productions that will make compilation techniques for variables feasible.

REFERENCES

- [1] Randall Davis and Jonathan King. "An Overview of Production Systems," E. W. Elcock and D. Michie (ed.), Machine Intelligence, Vol. 8, Wiley, N.Y., 1976.
- [2] P. E. Hart et al. PROSPECTOR - A Computer-Based Consultation System for Mineral Exploration. Mathematical Geology 10(5):589-610, 1978.
- [3] Richard O. Duda et al. Development of the PROSPECTOR Consultation System for Mineral Exploration. Technical Report, Artificial Intelligence Center, SRI International, 1978.
- [4] Frederick Hayes-Roth et. al. "Principles of Pattern-Directed Inference Systems," D. A. Waterman and Frederick Hayes-Roth (ed.), Pattern-Directed Inference Systems, Academic Press, 1978.
- [5] Digital Equipment Corp., Maynard, Mass. DecSystem10 System Reference Manual, 1974
- [6] E. H. Shortliffe. Computer-Based Medical Consultations: MYCIN. American Elsevier, N.Y., 1976.