

## Materials:

## 1. Projectable form of Roman numeral production system

## I. Introduction

- -----

A. At the start of the course, we mentioned that two key issues in AI are knowledge representation and search.

1. Earlier, we looked at one knowledge representation system - Predicate Calculus - and at a programming language based on it - Prolog.

2. Then, we spent time talking about search.

B. Now, we are going to look at two schemes which, in some ways, combine these two areas. That is, they are a way of representing knowledge in a way that facilitates efficient search, while still maintaining a clear separation between knowledge and control.

C. In both cases, predicate calculus could be used as the underlying "language" for representing the knowledge, but does not have to be. (In some sense predicate calculus can be to knowledge representation as English is to Literature).

## II. Production Systems

-- -----

A. One widely-used methodology which we want to develop in some detail is called a LINEAR PRODUCTION SYSTEM.

1. The domain-knowledge portion of an AI program organized as a production system will contain a collection of PRODUCTIONS or condition-action rules of the conceptual form:

IF condition(s) THEN action(s)

or

condition -> action.

(though they may actually be expressed somewhat differently in the actual program code.)

2. The conditions will all be expressed in terms of the current state of problem.

3. The system repeatedly executes the following cycle:

identify the rule(s) whose conditions match the problem state  
select one.

carry out the action(s) specified in the rule

a. Sometimes, the chosen rule is said to "fire".

b. The cycle terminates either when no more productions match the current problem state, or when the action part of a chosen rule specifies to stop.

4. One important consideration in the design of a production system is how to handle the case where two or more rules are eligible to fire (i.e. their if parts match the current state of the problem.)
  - a. The set of eligible rules is sometimes called the “conflict set”.
  - b. There are many alternative ways of handling the choice of one rule to fire from among those in the conflict set.
    - i. One way to handle this, of course, is to design the productions in such a way as to ensure that no more than one is applicable at any given time. However, there are many problems for which this is not doable.
    - ii. A frequent way to handle this issue is to order the productions in some way, and to choose the FIRST rule that is eligible to fire. We call such a system an ORDERED PRODUCTION SYSTEM.

B. The following is an example of a very simple ordered production system. It prints out integers (from 1-39) in roman numeral form, repeatedly reading in a number, translating it, then reading in another, etc.

#### PROJECT RULES

```

if      X is null
then    read a number and call it X

if      X is greater than 39
then    print an error message
         set X to null

if      X is greater than 9
then    print 'X'
         subtract 10 from X

if      X is equal to 9
then    print 'IX'
         set X to 0

if      X is greater than 4
then    print 'V'
         subtract 5 from X

if      X is equal to 4
then    print 'IV'
         set X to 0

if      X is greater than 0
then    print 'I'
         subtract 1 from X

if      X is equal to 0
then    print a newline
         set X to null

```

1. Trace through execution of the above for:
  - a. 28
  - b. 40
  - c. -1
2. Note the flexibility of a system like this. The program could easily be extended to handle numbers larger than 39 by simply writing appropriate new rules.

Class exercise: extend to numbers up to 89.

C. We might organize a program for simplifying arithmetic expressions as a production system.

1. We would have productions like

if there is a subexpression of the form some number + some number  
then replace the subexpression by the result of evaluating it

...

if there is a subexpression of the form 0 + E  
then replace the subexpression by E

...

2. Note that this is superficially similar to the approach you are taking for Project 1, but is actually quite different.
  - a. These production rules are couched in terms of REPEATEDLY scanning the expression, looking for a subexpression that matches the condition. (Note that this can be time-consuming if the scanning is complex, as it might be in this case.)
  - b. In project 1, simplify is called ONCE on any given subexpression, after first simplifying its subexpressions.

D. Weizenbaum's ELIZA program was actually a form of production-system.

1. The heart of the program was a script which specified how to transform an input sentence into a response.
2. Here are examples of actuals rule from Weizenbaum's program, and their equivalent in a more readable form:

(I = YOU)

...

((0 YOU REMEMBER 0) (DO YOU OFTEN THINK OF 4) ... )

if the input contains "I", then replace it with "YOU"

...

if the (transformed) input is of the form (0 more words) I REMEMBER  
(0 or more words)

then respond with DO YOU OFTEN THINK OF whatever followed REMEMBER

- E. Production systems play a very important role in expert systems, as we shall see later.
- F. The Luger text noted a number of key advantages of production systems. Several of these are related to the notion of modularity - each rule represents a distinct unit of knowledge.

### III. Blackboard Systems

--- -----

- A. A blackboard system takes the idea of modularity we have just noted one step further by organizing a system as a collection of modules called KNOWLEDGE SOURCES (KS).
  1. Each knowledge source has its own control (either a separate CPU or a separate thread within a process running on a single CPU). The control of each knowledge source runs independently of the others.
  2. Communication between the knowledge sources takes place through a shared data structure called the BLACKBOARD. When any knowledge source discovers something that it wants to share with other knowledge sources, it writes it on the blackboard. This knowledge then becomes part of the input that other knowledge sources can consider.
- B. An early example of a blackboard system was HEARSAY-II - a speech recognition system (1976). Its raw input was the waveform of the acoustic signal. Its other knowledge sources included:
  1. A recognizer of phonemes (possible sound segments of the acoustic signal). When this KS recognized the (possible) presence of a certain phoneme, it wrote it on the blackboard.
  2. A recognizer of syllables. It looked at the phonemes and wrote down syllables that they might correspond to.
  3. A recognizer of words. It looked at the syllables and wrote down words they might correspond to.
  4. A recognizer of combinations of words from different parts of the input (words in context).
  5. A recognizer of word sequences.
  6. A recongizer of phrases.

Of course, any given KS might "recognize" something in the input that wasn't really there. For example, the word recongizer might recognize that a certain word might be possible; but if the next KS did not recognize a word combination based on the context of other words, then that recongition would go no further.