Conventional AI: Production systems and expert systems

Jocelyn Paine

June 3, 1997

1 Introduction — the birth of expert systems and production systems

When we did the General Problem Solver, GPS, you saw that it is often seen as marking the end of the let's-find-a-generally-intelligent-algorithm approach to AI. Proponents of this approach believed that we would eventually be able to find a powerful and intelligent algorithm capable of solving any problem, whether in chess-playing, algebra, or translation. All it would need to know about the problem would be a statement of the goal to be achieved and the "rules of the game". (In GPS, this information is conveyed in "operator tables" which specify the allowable "moves" or operations.)

Although GPS was called a General Problem Solver, it turned out not to be one. Its failure, some writers say, helped cause a shift in AI research: a shift from looking for generally intelligent algorithms (where all the intelligence is in the algorithm, and only a small amount of knowledge is added about each new problem), to devising problem-specific methods (where the "intelligence" is provided by large quantities of problem-specific knowledge).

The histories of AI generally agree that this was the motivation behind Dendral, one of the first *expert systems*. Instead of trying to construct grand methods which would solve any problem, Dendral's creators took a highly specific problem, that of interpreting mass spectra. They tried to answer the question: what knowledge do we need to do this, how should it be represented, and how can we acquire it? This is of interest to many chemists, and the Dendral project was set up as a collaboration between a group of chemists who needed to interpret mass spectra, and a group of Artificial Intelligentsia who saw this as a good test of the new approach. For an excellent account of this change of view, see Feigenbaum's article *On generality and problem solving — a case study using the Dendral project*, in *Machine intelligence 6* (1971) (RSL, probably the stack).

The approach chosen was to represent much of the knowledge as IF-THEN rules:

```
IF you know this
THEN you can deduce this
```

Systems which use rules in this way are often called *rule-based systems.*

The same kind of shift took place in the thinking of Newell and Simon, the originators of GPS. They had devised GPS as a cognitive model — a program which embodies a theory about how we perform certain types of cognitive task. GPS, however, is only a partial theory — it does not, for example, say anything about the structure of memory, and it makes very specific assumptions about how control passes from one sub-task to the next.

Their successor to GPS for cognitive modelling was production systems — a special class of rule-based system whose architecture is restricted to fit assumptions about low-level mental structure. In production systems, attention focuses much more than with GPS on what information is represented in memory, and what strategies are used to process it.

So this week's tutorial is on rule-based systems, on how they work, and on production systems and their use as cognitive models. I said above that production systems are deliberately restricted so they model various assumptions about mental structure — such as the capacity of working memory. For engineering purposes, when you just want a rule-based system to do a certain job, these constraints are counter-productive. It's therefore important to understand how the internal mechanism of production systems differs from that of rule-based systems in general.

One difference is that whereas rule-based systems can be either forward-chaining or backward-chaining, production systems are always forward-chaining. These names, forward-chaining and backward-chaining, name two methods of *inference.* They are not differences in the rules themselves, but in the way one reasons with the rules: it's possible to reason both forward and back with the same set of rules.

The difference between forward-chaining and backward-chaining is, like the difference between depth-first and breadth-first strategies, one that appears in many places; you should know it for the exam, and you should be able to work examples of either direction of chaining, given a set of rules.

In the readings, you will often come across the name *expert system.* Most, but not all expert systems are rule-based systems. There is no exact definition of an expert system, but it's generally agreed that expert systems:

- Represent their knowledge symbolically, possibly — but not invariably — as rules. This would rule out programs such as (most?) neural networks, and databases of statistical correlations.
- After giving an answer, can justify or explain it by showing which knowledge they used to come to that conclusion.
- In a similar way, can explain why they have asked their user a particular question.

- Can imitate the performance of a human expert in a narrow domain of expertise, such as mortgage tax advice, interpreting mass spectra, or planning computer systems.

2 An introduction to inference: mainly expert systems

Please start this week's work by reading Winston, *Artificial Intelligence*, Chapter 6.

- Read *Rule based systems for synthesis*, pp 166–174.

  Learn the algorithm on p 168, because you'll need it for production systems. Work through the Bagger example. Although it looks frivolous, it illustrates how control passes from one rule to another in production systems, and why there is more than one way to do conflict resolution. As you'll see later, the choice of control and conflict resolution methods defines part of the functional architecture of mind when production systems are used to model cognition.

  When you dry-run Bagger, you check the conditions of the rules against what Winston calls the database. This database corresponds to short-term memory in the cognitive models. Notice how the current step is stored in the database (e.g., on the bottom of p 169, `Step: Check-order`), and how it is tested in rule conditions. This amounts to representing one's goals in STM, and using them to drive future processing.

- Skim the XCON section, pp 174–175.

  This shows how a Bagger-like expert system can be employed for an industrial planning task. XCON frequently appears in the literature because it was one of the most successful commercial expert systems: it performed a task that couldn't be done manually or with a conventional computer program. (An alternative approach would have been for DEC to simplify their product range: but perhaps that's not as much fun as building expert systems).

- Read pages 176–182, up to the start of the section on AND/OR trees.

  Work through both the forward- and backward-chaining animal identifiers.

  Do you think forward- or backward-chaining is better for synthesis tasks like Bagger? Why?

- Read the section on explanation, pp 184–185.

  Winston does not question whether the explanations thus generated are suitable for the user of an expert system (or which type of user, if any, they'd suit). He was writing when it was taken for granted that these rule-by-rule traces would be.

- Read the section on Mycin, pp 192–195.

  Like XCON, MYCIN is often cited. It was the first backward-chaining expert system, and still serves as the model for many commercial products. It's probably true to say that until 1986 or so, most commercial systems were essentially MYCIN with its medical knowledge replaced by rules on tax-accountancy, telephone fault-diagnosis, or whatever.

- At last turning to cognitive modelling, read the latter half of p 200 to p 202.

  Note the difference between STM and LTM.

  Don't bother to learn what an "elementary production system" is from here — it's better to take an account from one of the cognitive modellers. The point is that the computer simulation is being restricted to what's believed to be true of the mind's functional architecture. Different workers have different hypotheses about this architecture, and do not come up with exactly the same kind of model.

3 More on expert systems and inference

If you're still puzzled about inference and the difference between forward and backward chaining, I've worked some examples. You can find these in my lecture notes 6 and 7, in the folder by the Psychology library catalogue.

There are also examples in *The Guide to Expert Systems*, by Alex Goodall (Learned Information, 1985), RSL, Comp BD 36, chapter 3. This contains a more concise than Winston worked example of the difference between forward- and backward-chaining. Incidentally, Chapter 5 is a discussion, for non-computer-scientists, of several types of knowledge representation.

For general information on expert systems, see the article in *The Encyclopaedia of AI*. There's also a nice book called *Expert systems*, edited by Richard Forsyth, in the same bookshelf in the RSL.

Now you should know the two directions of inference. Forward-chaining is usually (but not always) *data-driven*: rules act on data, producing new data. Backward-chaining is usually (but not always) *goal-driven*: the system's goal is to prove some conclusion, and rules are called to prove ever simpler sub-conclusions until you get down to known facts.

The distinction between data-driven and goal-driven can be applied in many places. For example, a vision system might continuously monitor its perceptions, all the time updating a primal sketch, from it a 2-1/2 D sketch, and from it a 3-D sketch, and finally a list of the objects it thinks it sees. Or it might form hypotheses about what objects are present ("I hear a roar") and then call the lower-level modules to do only as much processing as is necessary to prove or disprove the existence of a tiger.

Incidentally, the names "forward-chaining" and "backward-chaining" denote general strategies, not specific tactics. For example, in the cycle on p 168 of Winston, a forward-chaining system has to know when to stop trying to fire rules. It might stop when firing a rule produces no change in the database; or it might stop once some fact about a particular individual appears ("the animal I just heard is a ..."). There are many variations on both types of inference.

4 Production systems and cognitive modelling

Now to get onto some psychology at last: some more detailed reading on production systems and modelling.

- *Learning and Problem Solving 3*, Open University Course D303, Block 4, Units 26–28, pp 83–118. This shows production systems at work modelling two different tasks, and discusses some of the issues in designing a good model.

- Boden, *Computer models of Mind*, 154–168 and 210–213. Pages 210–213 deal with the same topic as pp 112–117 of the OU book: modelling childrens' performance on the task of seriation (picking bricks and putting them in order of size) and how their cognitive development improves performance. The task was explained by Piaget in terms of sudden progressions from one stage of development to another. Boden's description of the production system model is less detailed than the OU book's, but she's stronger on discussing its fit with Piaget's analysis.

  Note when reading her productions that you work from the bottom up, not the top down!

  Pages 154–168 continue from the topic of GPS as a model to production systems. Page 164 lists what Newell and Simon take to be the architectural features of the mind.

You may find the OU book heavy going if you work straight through from page 83. Unfortunately, it starts by discussing models of cryptarithmetic-solving, and they are not particularly simple.

Possibly this order will help:

- Pages 112–117 OU: seriation.

  See what Young was trying to do with his model. Try dry-running the simple rules in figure 25 page 115.

  The notation for rules is different from before. The condition and action are separated by arrows. In the condition, there are tests of the form `goal=seriate` or `goal=add first block`. These test the top goal on the *goal stack*, which is a part of STM. In terms of the next paragraph, these look at the top message on the spike, never anything lower down. There are also tests of the form "task just started" or "holding block in hand". These test perceptions of the outside world.

In the action, there are things like `push(goal = add first block)`. This put the symbols `add first block` onto the goal stack. Think of the goal stack as a spike, initially empty, onto which you can push bits of paper. `push(goal = add first block)` pushes a piece of paper with `goal = add first block` onto the spike. As mentioned above, the tests `goal = X` always look at the message at the top: anything lower down is invisible until it's uncovered.

There are also actions of the form `pop goal stack`. These take the top (and only the top) piece of paper off the spike, uncovering what's underneath.

This notion of a stack is described on p 105 of the OU book. Using goals stored on a stack to control rules is a bit like what Bagger did. The difference is that Bagger only ever noted one goal (the *current goal*). Here, we can store a whole stack of them.

Question: is this psychologically realistic? How deep can the stack be?

- Pages 210–213 of Boden.

  Read the comparison between this model and Piaget's explanation.

- OU, pages 83–94.

  Skim the computational details, except in section 2.4. Pick out the issues relevant to psychological modelling. Note the comment about "cheats" on p 92. Ignore comments about SOLO (it's a bit like Prolog, if you know Prolog). For STRIPS, read GPS: in the points he's making, there's little difference between them.

- OU, pages 95–102.

  Read in detail. Note the concepts of production memory, working memory (i.e. STM), how rules are activated, conflict resolution.

- OU, pages 103–111.

  Note the three different strategies. In general, you can make many different sets of production rules to model any one behaviour. How do you know which best fits the mind....?

- OU, seriation again.

  Now go back and dry-run the different sets of rules. See how well they model a child as it develops.

- Boden, pages 154–168.

  The issue of modelling.

5 Developmental psychology: other references

Now find a copy of *Computers and Thought — a practical introduction to AI* by Sharples, Hogg, Hutchinson, Torrance and Young (MIT press, 1989). Read the production system part of Chapter 8 (on childrens' subtraction), and follow up the references to the work of Young and O'Shea, in particular their article in *Cognitive Science* for 1981.

Note their (generally favourable) comments on Brown and Van Lehn's work on repair theory. There is a paper on this in *Cognitive Science* for 1980. More recently, Van Lehn has written a book called *Mindbugs*, now available in the library. If you have time, read at least the introduction.

6 Summary on production systems: what do they explain?

Why do Newell and Simon consider production systems good theories? There's an early article by Newell, detailing some of the flaws (as he sees them) in experimental psychology, and what production systems have to offer, in *You can't play 20 questions with nature and win*, in *Visual Information Processing*, edited by Chase (PSY BC:C 39).

For a more recent approach, sometime during the course you should certainly look at *Unified Theories of Human Cognition* by Newell, also in our library. Try to read the first four chapters.

Finally, please read Richard Young's chapter on *Production Systems for Modelling Human Cognition* in *Expert systems in the micro-electronic age* edited by Michie (E.U.P. 1980) (in the Psychology library and RSL). Pay particular attention to the three levels of modelling he distinguishes, on page 43.

7 ACT*

The type of production system model you met above is not the only one possible. For an alternative, read the preface and Chapter 1 of *The Architecture of Cognition* by John R. Anderson (PSY BH:A 547). By its differences from Newell's approaches, it throws them into sharper relief, and also provides another example of what is meant by "functional architecture".

8 Essay

Unfortunately, the features that make production systems suitable as models also make them tedious to read about. Lots of very simple rules, each performing a minute change on a big database: any model of reasonable interestingness takes many many cycles to run. To get a feel for production system models, it's essential to try lots of examples. So you may not have much time left to write an essay.

I'll set one anyway. Please choose *one* of the following questions, and write an essay on it. (If you want, you can write on more; it would be useful to have the others prepared for Finals.)

a How do expert systems differ from production systems? Discuss the differing rôles of AI as engineering and AI as cognitive simulation.

b Recent advances in computer modeling lead to the conclusion that the mind is a rather low-grade expert system, with a grossly restricted working memory, and impoverished rule format. Thus knowledge engineering, the task of transferring knowledge from mind to expert system, becomes merely the task of transferring data from one rule-based system to another, and is hence essentially trivial. Discuss.

c What, if anything, have production systems contributed to developmental psychology?

d Production systems can simulate cognitive tasks. Does it therefore follow that the mind *is* a production system?