ARCHITECTURE-DIRECTED PROCESSING

Richard M. Young

MRC Applied Psychology Unit, Cambridge, England

Abstract. Certain general characteristics of human cognition may be due to properties of the functional architecture of the cognitive processor. While proposed cognitive architectures are almost always "universal" and can be forced to execute arbitrarily chosen computations, nonetheless it is possible to delineate a class of "compliant" processes that allow the architecture of the processor to influence the course of processing. A speculative case is made that such compliant processing is responsible for invariants of human cognition, such as that problem solving occurs as heuristic search in a problem space, that long-term memory search takes place in cycles of retrieval and re-description, and that uncertain information is dealt with by prominence heuristics.

Compliant processes

A central theme in Cognitive Science is the explanation of features of human cognition in terms of properties of the programs that generate and regulate behaviour. The paradigm is to account for the empirical phenomena observed in some domain, e.g. the time taken to decide the truth or falsity of simple propositions, by showing that they derive from properties of the processes responsible for the behaviour. For all its undoubted merits, there is a gap at the heart of this approach. Although such computational explanations have genuine scientific value, for example by offering a single coherent account for a range of apparently diverse phenomena, there is a need also to try to understand why those particular programs are found but not conceivable others.

The idea explored in this paper is that the functional architecture of the processor itself influences and constrains the kind of programs it can execute, and hence leads to invariants in the resulting behaviour. There is little novelty in this idea: all I hope to do here is to draw together a number of threads from various places. The idea derives mainly from the work of Pylyshyn (1980) and especially Newell (1973, 1980). Pylyshyn (1980) discusses the notion of the functional cognitive architecture, i.e. the fixed structural properties of the human cognitive system. Building on that notion, we extend it to the properties of the processes that the architecture supports. The argument is inspired by, and is closely similar to, that of Moore & Newell (1974). In describing a system called Merlin built round a single processing mechanism, that of assimilation by analogy, they suggest that certain general problem solving methods (Generate and Test, Heuristic Search, etc.) arise within Merlin as "natural methods". In other words, Merlin exhibits these methods not because it runs a program directing it to do so, but because they arise as consequences of its single processing technique. In a similar way, this paper is proposing that certain general characteristics of human cognition arise as "natural methods" from the functional architecture of the cognitive processor.

It may be helpful to consider an analogy, both to understand the idea better and also to highlight its idiosyncracies. Most of us are familiar with the idea that different programming languages lend themselves selectively to different sorts of programs for different sorts of tasks. It is possible in principle to use LISP for commercial programming and COBOL for list processing, but in practice certain kinds of program fit "naturally" into certain languages and only with difficulty into others. Note, however, that this "naturalness" is extremely hard to pin down in a formal definition.

The notion of architecture-directed processing is somewhat similar. It centres on the idea that for a given architecture certain programs will run "naturally", while others can only be coaxed on with a sledgehammer. However, architecture-directed processing goes beyond the idea of naturalness, since it allows the architecture to influence the actual selection and sequencing of the steps to be taken. To some extent this is also true of programming languages. With ordinary sequential flow languages, such as FORTRAN and PASCAL, there is a kind of "default" control structure (i.e. execute the next statement) which the programmer can override when she wants to (by iterations, jumps, subroutines, and so on). However, in normal practice programmers use this sequential control in order deliberately to specify the order of execution, so to regard it as a default is a little misleading. With architecture-directed processing the influence is more pervasive, since at least for certain production system architectures (PSAs) (Newell, 1973, 1980; Anderson, 1976; Waterman & Hayes-Roth, 1978) the program does not have to specify an order of execution at all. Once the repertoire of possible steps has been supplied, the selection and sequencing can be left to the architecture, which appropriate PSAs can perform in a highly flexible manner responsive to the particulars of the task (Young, 1977, 1979). The program can still, of course, specify the control structure where it needs to. This freedom leads to the idea that responsibility for the flow of control has been split between the program and the architecture. It follows that programs will differ in the extent to which they insist upon a particular control regime. Programs that allow the architecture to have largely its own way we will call compliant. Not to be taken too seriously, but as a starting point, we can offer a tentative

> Definition. A program is "compliant" to the extent that it allows the selection and sequencing of steps to be determined by the architecture it runs on.

The examples given below will try to demonstrate that, given an architecture, compliancy leads to the appearance of certain invariants in the generated behaviour.

It's hard to be precise...

In one important respect this notion of "compliancy" is very similar to the idea of "naturalness" in programming languages, and that is in the difficulty of making it more precise. Despite our recognition of the selective suitabilities of different languages for different kinds of programs, it remains the case that the languages are almost always computationally "universal", and therefore formally equivalent in power. It follows that any program can, in principle, be written in any of the languages, and that it is

hard or impossible to capture the idea of "naturalness" in a formally precise way. So far as I know, even the recent progress in computational complexity has nothing to say about this important practical problem. The story is similar for compliancy. Proposed cognitive architectures are almost always universal, and thus it is possible in principle to run any program on any architecture. In most cases, of course, this will require a non-compliant program which imposes an alien control structure. Our interest is in the cases where this kind of brute force is not needed.

The following examples will make clear that further progress depends upon being able to specify what is meant by compliancy in more precise terms. I am not totally optimistic that we will succeed in this, but I can see two avenues worth exploring. The first is to take advantage of the fact that compliancy has to do specifically with flow of control. There is a sense in which the steps of compliant programs execute at "base level", whereas non-compliant programs require an extra level of interpretation. If this difference can be captured reasonably precisely, there is some hope of deriving the consequences of compliancy in a more rigorous way. The second possibility depends on achieving some understanding of the mechanisms by which new programs are acquired. This might provide a much stronger basis for placing constraints on the kinds of program the cognitive processor will run: not that non-compliant programs are "unnatural", but by showing that only compliant programs could ever be learned.

Examples

There follow three examples to illustrate how compliant programs lead to the appearance in behaviour of certain invariants dictated by the underlining architecture. Two warnings need to be given. One is that the difficulty of making the notion of compliancy even moderately rigorous makes it impossible in any strict sense to derive the invariants from the architecture. The arguments given, though intended to be plausible, have to be regarded as hand-waving. The other is that these examples are speculative. I would not wish to give the impression that the arguments are summaries of a more complete story already worked out. Rather they should be regarded as the goals for a programme of work still to be undertaken.

Ex.1: Problem solving is carried out by heuristic search in a problem space. That assertion can reasonably be taken as the one-sentence conclusion of Newell & Simon's (1972) study of human problem solving. It arises as a consequence of compliant programs running on PSAs of certain types. This is the clearest of the three examples, and the argument is essentially due to Newell (personal communication).

Consider a PSA which is like OPS (Forgy & McDermott, 1977) in the following respect. Whenever more than one production rule is applicable, the one to fire is determined by the following principles ("conflict resolution"). (1) Recency: rules whose conditions are sensitive to more recent information take priority over those matching only older information. (2) Special case: rules which are special cases of other rules take priority over them. (For further details see McDermott & Forgy, 1978; Forgy & McDermott, 1977). Suppose that knowledge of the problem domain is coded by specifying the possible moves that can be taken in circumstances C as rules like:

Rule 1: C & <? side conditions> = <action1>
Rule 2: C & <? side conditions> = <action2>,
etc.

Note that such rules provide a highly compliant representation. They impose only local constraints on how they are used, and thus have individually, as it were, no opinion about the more global flow of control. Suppose that C is known. Then one of the rules shown will fire, Rule1 say. If the action taken leads to some new information and there exist rules responsive to that information, then by the recency principle it will be one of those rules that fires next. And so it continues, as long as there is new information and rules to respond to it. Once that is no longer so, processing falls back, say to the rules shown, and one of the alternative rules at that level will fire; in this case, Rule2. In other words, a depth-first search is performed. On the other hand, if at any time a rule which is sensitive to a particular configuration of information becomes satisfied, then by special case it will be the one to fire. In other words, specific knowledge is brought to bear when appropriate. The upshot of all this is that the principle of recency generates depth-first search, while special case adds heuristic guidance.

It is worth emphasising the contrast between this explanation and virtually all earlier accounts in the cognitive modelling literature (including, for example, Newell & Simon, 1972). We have just argued that people solve problems by heuristic search, not because they run a "heuristic search program", but because, in the absence of guidance to the contrary -- i.e. with a compliant program -- heuristic search is the natural thing for the PSA to do.

Ex.2: Indirect recall from long term memory. When the cues presented are insufficient to elicit some target information from long term memory directly, both theory (Norman & Bobrow, 1979) and the experiment (Williams & Hollan, 1981) suggest that recall occurs in a series of cycles of alternating retrieval and re-description.

Again, this behaviour is a consequence of the conflict resolution principles of a PSA. Suppose that the target information is on the action side of a rule. Then by supposition, not all the information on its condition side is yet present (the point is to gather it so that the rule does fire). Whatever information is present, constituting a partial description of the item being sought, will trigger some rule or other. This in turn will add to the description. Special case ensures that each item retrieved is relevant to the current description; if there is no relevant information, then general procedural heuristics will fire. As in problem solving, the recency principle ensures that newly retrieved information is followed up first.

Ex.3: Uncertain information is dealt with by "prominence" heuristics (Fox, 1980a), such as representativeness and availability (Tversky & Kahneman, 1974). The implied contrast is with rational, non-heuristic techniques such as the use of Bayes' theorem and the maximisation of expected value. For this example we have to move beyond the OPS architecture, to a PSA which assigns different strengths to different items, and thereby recognises a degree of match between the data and a rule. Examples are HPSA (Newell, 1980) and the PSYCO architecture used for simulating medical

diagnosis (Fox, 1979, 1980b). The argument essentially follows those two authors.

The key issue is the representation of the degree of certainty. If it is coded explictly as simply another component of the data,
e.g. (DISEASE-IS GASTRIC-ULCER CF = 0.7),
then it will be treated as part of the information content by whatever rules happen to process it, and no consequences follow from the architecture. If, on the other hand, certainty is coded as the strength of the item,
(DISEASE-IS GASTRIC-ULCER) [0.7],
then the certainty has effects at the level of the architecture (i.e. it appears as an aspect of the form rather than the content of the item), and influences processing at this level. What happens of course is that certainty enters as a factor in conflict resolution, with stronger items, other things being equal, being processed before weaker ones. The outcome is that processing of uncertain information is dominated by the data that for whatever reason are more "prominent" in memory (Fox, 1980a). Items which are highly familiar, already in working memory, or more closely linked to other relevant items will be the first to come to mind and will carry more than their fair share of responsibility for guiding behaviour.

References

Anderson, J. R. (1976) Language, Memory and Thought. Erlbaum.

Forgy, C. L. & McDermott, J. (1977a) OPS, a domain-independent production system language. Proceedings of the 5th International Joint Conference on Artificial Intelligence, 933-939.

Forgy, C. L. & McDermott, J. (1977b) The OPS2 reference manual. Technical Report, Department of Computer Science, Carnegie-Mellon University.

Fox, J. (1979) Medical diagnosis: Inference, recall and a theory of skill. Unpublished ms.

Fox, J. (1980a) Making decisions under the influence of memory. Psychological Review, 87, 190-211.

Fox, J. (1980b) The PSYCO manual. MRC Social and Applied Psychology Unit, University of Sheffield,, England.

McDermott, J. & Forgy, L. (1978) Production system conflict resolution strategies. In Waterman and Hayes-Roth (1978), 177-179.

Moore, J. & Newell, A. (1974) How can Merlin understand? In L. W. Gregg (Ed.), Knowledge and Cognition, 201-252. Erlbaum.

Newell, A. (1973) You can't play 20 questions with Nature and win: Projective comments on the papers of this symposium. In W. G. Chase (Ed.), Visual Information Processing, 283-308. Academic Press.

Newell, A. (1980) HARPY, production systems and human cognition. In R. Cole (Ed.), Perception and Production of Fluent Speech. Erlbaum.

Newell, A. & Simon, H. A. (1972) Human Problem Solving. Prentice-Hall.

Norman D. A. & Bobrow, D. G. (1979) Descriptions: An intermediate stage in memory retrieval. Cognitive Psychology, 11, 107-123.

Pylyshyn, Z. (1980) Computation and cognition: Issues in the foundation of cognitive science. Behavioural and Brain Sciences, 3, 111-169.

Tversky, A. & Kahneman D. (1974) Judgement under uncertainty: heuristics and biases. Science, 185, 1124-1131.

Waterman, D. A. & Hayes-Roth, F. (1978) Pattern-Directed Inference Systems. Academic Press.

Williams, M. D. & Hollan, J. D. (1981) The process of retrieval from very long-term memory. Cognitive Science, 5, 87-119.

Young, R. M. (1977) Mixtures of strategies in structurally adaptive production systems: Examples from seriation and subtraction. Proceedings of workshop on pattern-directed inference systems. SIGART Newsletter No. 63, June, 65-71.

Young, R. M. (1979) Production systems for modelling human cognition. In D. Michie (Ed.), Expert Systems in the Microelectronic Age, 35-45. Edinburgh University Press.