“Things are not Always what they Seem”

(motto from previous night’s fortune cookie)

Some comments on the 3rd day presentations, and looking back across the Workshop/PG Summer School as a whole

Richard M Young
University of Hertfordshire
r.m.young@herts.ac.uk

At the time of the meeting, I used hand-written slides. These typed pages are a reasonably faithful transcription of those slides. In particular, I have mostly retained the original page breaks, so these pages tend to be rather short.

[I have added a few subsequent notes in square brackets, towards the end of the presentation.]

Overview

- Today's papers — briefly
- Some general remarks about the meeting
  — including (yet more) ACT/Soar comparison points
- A foray into:
  ACT-R/5 and expertise modelling / knowledge engineering

Today's talks:

Learning from Instruction

Little to add, just a couple of points

1) Significance of the work — again, has already been spelled out
   - Newell "Reduce the theoretical degrees of freedom"
   - Anderson "Solves mystery of where productions come from"
   - "Complete the loop", etc.

There are also benefits internal to the ACT-R project

e.g. if we had qualms about some aspect of ACT-R practice, this work should help to resolve them.

2) We need all 3 components — again, stating the obvious
   - Natural Language — to remove analyst bias from the loop
   - representation and interpretation of instructions
   - rule learning, to "compile out" the interpretation

Lessons from Soar

N.B. These comparisons offered not in a spirit of "been there, done that" with the implication that all the problems have been solved, but rather that the earlier Soar efforts covered much the same ground as current ACT-R ones, and there is perhaps something to learn from them.

a) First version of TAQL

- declarative representation held within Soar, and compiled by Soar's learning mechanism into productions

b) First version of instruction-taking

- instructions as a "situated resource"

  [e.g. Vera, Lewis et al, ATM instructions;

  see also "Rosemary's baby" in Young & Lewis account of WM in Soar]

c) High point: Scott Huffman (~1994: Michigan PhD thesis & JAIR article)

- integrate NL, instruction learning, problem solving, …
- no episodic memory as such, but use of "debris" from NL processing as a mnemonic resource …

Miscellaneous Remarks

- The ACT-R enterprise — flying along
- Some care needed
  - — are all the old models *re-implemented* in version 5?
    [Just setting a switch for backwards compatibility misses the point.]
  - — Frank Ritter's warning about chasing high-demand funding sources, and its possible effect on the research community
  - — the learnability constraint
    - * bear in mind at all times?
    - * can ACT learn *all the different kinds* of rules?
- **Prize for most cited thesis** goes to … Raluca!

Production Learning as Second-Class Citizen?

(This is the only slide where I'm "critical" of aspects of the ACT-R enterprise)

- For example, on the new diagram for ACT-R 5,
  - — can't talk about production learning, because productions don't appear on the diagram
  - — breaks the symmetry between declarative and procedural
- Niels Taatgen's new mechanism looks promising
  - — it may be the answer, but …
- In the ACT enterprise as a whole
  - — for rule learning, one mechanism tried after another
  - — we don't really know how to do it right
- This contrasts with the learning of declarative knowledge
  - — where there's a clear theoretical basis
  - — and a deep analysis of the architectural options

(Yet More) ACT vs Soar Comparisons

- The cry of “turn on all learning!” will sound familiar to the ears of Soarers
- Marsha Lovett’s suggestion of “kernal rules”, e.g. for basic capability at instruction interpretation, sounds like the *default rules* in Soar.
- A significant difference between the communities is that Soarers typically build few models (? ~2), while ACTors build many (? ~10)
  - — one factor is the relative size of models (large in Soar, small in ACT)
  - — fits with the observation that an ACT model is of a subject in a psychological experiment: you’d clearly want to have several of those.
- The ACT community adopts an exploratory style to aspects of the architecture: “We’re unsure, so we’ll leave it open for now”

  That’s fine: gather experience at model-building before committing to a decision.

  BUT, occasionally there’s a need to stand back and ask about the in-principle capability of a proposed mechanism,

  e.g. for expressing highly skilled behaviour [see following slides]

  e.g. ability to learn all forms of rules

Some remarks about …

ACT-R 5 and the

Modelling of Expertise / Knowledge Engineering

(with much help from Rick Lewis)

- I have some minor concerns about the engineering of large ACT models …
  - — e.g. it's not obvious how to partition a large system among multiple programmers: is doing it by goals the right tactic?
  - — some aspects of ACT's activity are not explicitly represented, and this may make it hard to write introspective and reflective models
- … but the UHDO RLQV concerns the **modelling of expertise**
- I'm going to present the argument step-by-step

Step 1: A view of expertise

- Consider the view of expertise from old-fashioned Expert Systems:
  - — expert knowledge consists of complex rules, each typically with lots of conditions and few actions
- In other words, in the course of solving a problem, what an expert does next comes to depend on more and more subtle features of the situation.
- For example, a fine discrimination:

  A & B & C & D & … => X

  A & B & C & D' & … => Y
- Thus the focus of expertise is on ***control***
  - — modelling what to do next

Step 2: A view of ACT-R 5.0

- We hear the story that a single ACT 4.0 production, in ACT 5.0 gets split into two separate productions
  - — one to request the retrieval, the other to "harvest" it
- But, Rick points out, it would be a mistake to the regard the 5.0 productions as coming "in pairs"
  - — because once an item is in the retrieval buffer, it can fire any production that matches, not just the "partner" of the one that requested it.
- So, we're re-constituting a view of the architecture, as:
  - — a small set of slots (the "buffers")
  - — each containing one chunk whose components are directly accessible for production matching (i.e. without further retrieval)
  - — production conditions are sensitive to patterns of the components of those chunks
  - — when productions fire, they modify the contents of the buffers.
- It's true that some of the buffers are somewhat "magical"
  - — e.g. the contents of the perceptual buffers change spontaneously
  - — e.g. when something is written into the retrieval buffer, it disappears, and something different appears there a short while later

  but this doesn't affect the argument, concerning the view of the architecture expressed in the previous bullet point.

Step 3: So, we have re-constituted a classical production system!

We have re-constituted (a version of) a classical production system

- The (classical) WM for the production system is the ~4 buffers of ACT 5.0, productions fire depending on the buffers' contents, thereby changing those contents, and so on …
- **But** … ACT's **declarative memory** (DM) has got lost from the story!!
  - — we haven't lost the *knowledge* in DM, because it's still there to be pulled in by retrieval
  - — but so far as control is concerned, DM has gone

    which production fires depends on the buffers, but not on DM

° Notice that the effective state, comprised of a handful of buffers, is very small
  - — during the meeting, someone used the phrase "limiting the flow of information to conflict resolution", and that's certainly what's happening

[Note added later: I realise now that this situation is not all that different to ACT 4.0, where also which production fires — though not which instantiation fires — depends only on the “buffers” (i.e. mainly, the current goal). But the argument seems a lot cleaner in version 5.0.]

Step 4: So how do we reconcile …?

So how do we reconcile these two positions:

(1) Expertise consists in large part of making *control* — i.e. the determination of which production fires next — be sensitive to increasingly subtle aspects of the situation, reflecting a growing base of knowledge;

with

(2) The observation that in ACT-R 5.0, *control*, i.e. which production fires next, depends on only a small state consisting of the buffers only, *without* declarative memory — which is presumably where most of the knowledge base for expertise must reside?

[I am tempted to add: a Soarer might conclude that expertise must reside in productions rather than in declarative memory, but that would not be consistent with the spirit of ACT-R.]

Step 5: Rich symbols

The answer would seem to be, that we must use "rich symbols" in the buffers.

- A ***rich symbol*** is a specialised chunk, which encodes a highly particular pattern of (recursively) more basic chunks.

- The word *chunk* is multiply appropriate here, connecting with
  - — Miller's (1956) notion of hierarchical chunks, in the famous "7±2" paper
  - — ACT-R's sense of chunks as units of declarative knowledge
  - — Soar's notion of "data chunks" which can be used to associate a specific declarative structure with an arbitrary symbol.

- These are definitely *learned* chunks, which would show up in ACT as gensyms, like 'CH2947' or whatever.

- The essential idea behind this solution is that
  - — the problem situation would be encoded (hierarchically) in a small number of rich symbols
  - — those rich symbols trigger a specialised learned rule, based on the pattern of symbols in the buffers.

[**Note 1**. The *foundational* aspects of these rich symbols look a bit shaky in ACT-R. In ACT, each chunk has a number of components which are themselves chunks, so that — in principle — *every* chunk is already a "rich symbol", in the sense of having components that are each particular chunks, each of which itself has components which … etc. In practice, certain chunks are treated as primitive because their *name* appears explicitly in a production, but that needn't prevent them from being spelled out into their components in declarative memory, and having other rules respond to those lower-level components. It's all rather unclear (at least to me) where this process ends. Does ACT-R have a special class of atomic, non-decomposable primitive chunks, perhaps linked in some way to perceptual categories?

Despite this uncertainty over their foundations, the practical notion of rich symbols still seems pretty clear, as learned chunks which encode highly specific combinations of their components, which can themselves be rich symbols.]

[**Note 2**. Stefanie Nellen later pointed out to me a potential problem with these complex chunks ("rich symbols"). Most complex chunks will, necessarily, be very rarely encountered. They will therefore have low activation in declarative memory and will be difficult if not impossible to retrieve.

Without further analysis, I don't know the solution to this problem. Two ideas that occur to me are:

(a) that the associative aspects of retrieval will, in some way that I am still not familiar with, come to the rescue and make even these highly specialised chunks retrievable;

(b) or perhaps we really do have to consider seriously the possibility that productions, rather than chunks, are the natural medium for encoding high-quality but very-low-frequency knowledge?

It also occurs to me that perhaps this difficulty reflects the psychological situation, and that it sets bounds on the speed with which expertise can be acquired?]

Step 6: Minimising retrieval

Notice that according to the view we have suggested, in skilled behaviour ACT may run for many cycles without a declarative memory retrieval.

Compare this with Niels Taatgen's view that "the purpose of production learning is to reduce retrievals".

The convergence suggests that there may be something very right about the theoretical basis of Niels' approach.

Predictions

I make the following predictions, and indicate their implications, for ACT-R 5.0 models of expertise:

a) the current situation will be encoded mainly in rich symbols

   => we must be able to work with ACT-R symbols without meaningful names.

b) models will need to learn productions to encode and decode rich symbols

   => requirements on production learning

c) specialised expert productions will mainly take the form of rules for mapping buffer states to buffer states

   => requirements on production learning

d) => the production learning mechanism had better be adequate!

e) we'll be moving away from the familiar style of ACT-R models, e.g. using rules with hand-written meaningful symbol names, to dealing with learned rules and learned rich symbols

   => we had better be prepared to cope with it.

R.M.Y.

(These notes re-written 27 Aug 2001)