

## Introduction

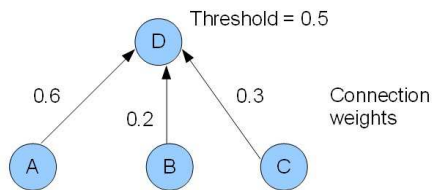
Two broadly different approaches to building models of cognitive processes are common:

1. High level models, which look and behave as much like the system they model but without necessarily reflecting the actual components that underlie the real system - rule based systems.

2. Low level models, which take the components of the real system and try to understand their behaviour, to see how they might work together to create the whole - parallel distributed processing (PDP) / connectionist systems.

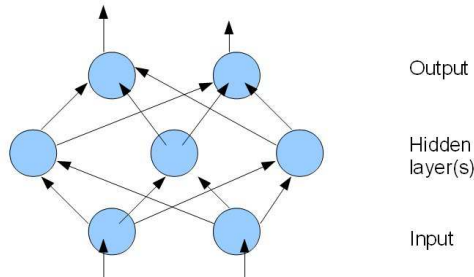
### Parallel distributed processing (PDP)

**Rummelhart and McClelland** - artificial neuron models. Neurons are activated if their activation threshold value is reached. Links between neurons have weights.



If activation = 1, then D is activated if A is, B and C are, or A plus any other node(s), but not if only B or C are.

Most common form of model is a three layer network:



## Chapter 16 - Cognitive modelling and cognitive architectures

PDP networks learn through training. Many 100s/1000s examples typically used to adjust the weights of the connections between neurons to get particular outputs from particular inputs. Known as backward propagation of error.

Features:

(i) PDPs do not need to contain explicit rules, but behave as if they are following them. e.g. **Price and Smolensky** - model of how words are arranged to produce grammatical sentences conforms to rules but is not governed by them explicitly.

(ii) Damage to a network results in graceful degradation - similar to what is observed in humans.

(iii) They can exhibit emergent properties - i.e. a behaviour which develops over time and is not specifically programmed for - e.g. generating a prototype 'dog' from specific dog examples during training. Similar to the way humans learn.

### Rule based systems

Main difference with PDP systems is that cognition is modelled as a set of explicit rules.

Production rules consist of a condition+action - i.e. IF the condition is matched; THEN the rule fires.

**Marr's** levels of explanation are useful in understanding the relationship between PDP and rule based systems.

PDP approach similar to the hardware level. The model reflects basic properties of the human brain (i.e. simplified neurons, links between them, activation) and uses it to see if interesting psychological behaviour will emerge. The rules based approach places little emphasis on the actual brain,

as this is in the biological realm, rather than the psychological one.

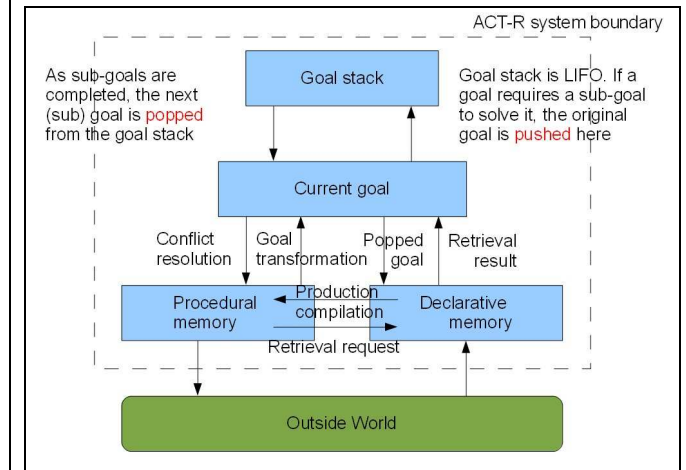
### Cognitive architectures

**Newell** - discrete models of cognition are not sufficient. Instead, there needs to be some integration and consistency between models for different task - for example, the way in which LTM is implemented should be consistent between models. Cognitive architecture is used as an over-arching framework to account for a range of phenomena using a fixed set of mechanisms.

PDP architectures certainly comply with this, as do rule based systems such as Soar (**Newell**) and ACT-R (**Anderson & Lebiere**).

### ACT-R overview

Primarily rule based, but with some PDP elements (even more so in ACT-RN, a later development). This reflects the trend towards the development of hybrid models.



### Three main components:

1. Declarative memory - split into chunks, representing 'facts'. Each fact has a level of activation - ones that have been used more recently have higher activation levels.

2. Procedural memory - consisting of production rules. These also have activation, may require retrieval of facts from declarative memory to complete, they fire if the condition of the rule is met. Procedural memory is not fixed - new rules can be learned (production compilation), mimicking human skill acquisition.

3. Goal stack - this has the effect of making high level cognition serial, which is a human characteristic so a positive feature of the model. However, the goal stack is perfect - all goals and sub-goals are remembered and acted on in the right order - a characteristic of human cognition is that this does not happen!

**Rabinowitz & Goldberg** - experiment: differences between declarative & procedural encodings of the same instructions. Aim: if a declaratively encoded instruction can be reversed more easily than a procedurally encoded one, it would imply distinct memory systems for declarative and procedural memory.

Found P's with less practice on particular tasks performed better when the examples were the same as the training ones. If a P has more practice (implying the formation of a rule), this was not the case. => two distinct memory systems at work.

### ACT-R accounts of memory problems

List memory - how people recall information from STM

**Baddeley** (and many others) - empirical evidence is consistent when looking at how individuals perform using forward, backward and free recall.

**Anderson et al** gathered data on how human P's performed in both accuracy and latency (of response) on a 9 element numerical list. P's aware of the number of list elements at the start of the test. One item displayed at a time; list had to be recalled forwards or backwards.

Results: Recall is best for 1<sup>st</sup> elements in the list (primacy effect) => greater rehearsal. Accuracy then higher again for the last two elements of the list - the recency effect.

ACT-R set up to model forward recall, using:

### Declarative representation of lists

The lists were represented in declarative memory, chunks being used to represent the list as 3 groups of 3 items. [c.f **Hitch et al**, **Chase & Simon** - novice/expert chess players form the same number of chunks in STM, experts can put more items in each group however.]

### Production rules for rehearsal & list retrieval

A number of procedures required, as for example:

- (i) List contents need rehearsal
- (ii) List contents have to be retrieved
- (iii) Responses have to be 'said'

Such rules are generalised - e.g. 'get next item from group'; 'output the item'.

### List activation

Higher activation for a chunk => easier retrieval. Activation is part of the PDP-like nature of ACT-R => chunk activation is determined using a set of activation equations.

High rehearsal => high activation; time is a factor in activation too - more recently used items have higher activation.

Association strength - strength of the bond between an item and the required chunk depends on the total number of associations an item has.

ACT-R assumes activation is a limited resource. A single item associated with a single chunk gets full activation; otherwise, activation has to be shared between the chunks the item is associated with.

Limited capacity of association strength in ACT-R allows the fan effect to be explained by the model (**Anderson**) - i.e. the greater the number of facts related to a single concept the slower a P is to recall any one of them - a feature that applies to list memory.

For any chunk to be retrieved by ACT-R, its activation has to be greater than an activation threshold. Additionally, the weaker the activation, the slower recall will be.

Partial matching can also occur in the model - e.g. if a partially matching chunk has high activation and a fully matching one is below the activation threshold. Further equations in ACT-R are used to determine if positional confusion occurs - i.e. digits get mis-ordered on recall.

### Running the model

**Anderson & Lebiere** - a simulation in ACT-R for list memory closely matches human performance in both recall accuracy and latency.

### Evaluation

ACT-R was able to be used to develop versions of a model which can explain a range of empirical data related to list memory. However, the model required significant customisation to fit the experimental data.

**Anderson et al** - acknowledge this as a weakness and => if this is necessary for a limited area it will be difficult to provide a unified model with ACT-R for all

<p>cognition.</p> <p>However, ACT-R does impose architectural constraints which limit how the model can work - e.g. the procedural/declarative memory distinction is enforced; chunks work in a particular way etc.</p> <p>So called <u>architectural assumptions</u> like the procedural-declarative memory distinction is common to all ACT-R models. <u>Auxiliary assumptions</u> are then made on a case-by-case basis to deal with peculiarities of specific experiments. It is argued that only if an architectural assumption needs changing that it would imply ACT-R does not reflect a general feature of human cognition.</p> <p><b>Learning and using arithmetic skills</b></p> <p>In ACT-R simulations of this domain, the focus is on production rules (rather than on declarative memory, as in the previous example).</p> <p><u>Production compilation</u></p> <p>Chunks representing a step in a process are known as <u>dependency chunks</u>. One is created every time a goal from the stack has been completed. e.g. if the goal is 3+4, then a dependency chunk is created if it is solved (for example) by matching it with the addition fact 3+4=7.</p> <p>Each dependency goal chunk therefore represents a specific lesson from experience. These can eventually become generalised through production compilation - e.g. 3+4=7 becoming X+Y=Z produces a generalised addition rule.</p> <p>Not all generalisations succeed (just like human learning) - those that are not successful lose activation until they are 'forgotten'.</p>	<p><u>Addition by counting - an example of human behaviour</u></p> <p>If there is no addition fact for 3+4=7, then one strategy may be to solve it by counting on from 3 - as <b>Siegler</b> notes that children sometimes do.</p> <p>Something like:</p> <p>Add-numbers  Subgoal-counting  Startcount  Add1count  Stopcount  Found-sum  Say</p> <p>would be the way ACT-R solves the problem the first time around, followed by something like:</p> <p>Retrieve-sum  Say</p> <p>the second time around as the addition fact 3+4=7 would be in declarative memory and have high activation (<b>Anderson &amp; Lebiere</b>).</p> <p>However, this may not be the whole story - <b>Siegler</b> also notes that some children don't do this - they learn by rote. This demonstrates that cognitive models tend to correspond more to individuals rather than generalised populations.</p> <p><b>Models of learning / problem solving in practice</b></p> <p>Learning is modelled by ACT-R as an incremental process; knowledge acquired declaratively; performance becomes more accurate and faster through practice. It implies an ACT-R model might be able to explain the skill of learning to drive, for example.</p> <p>However, what is less easy to explain is the 'U' shaped pattern of learning seen in:</p>	<p>(a) Children's acquisition of English past tense verbs (<b>Bowerman</b>)</p> <p>(b) Children's ability to solve mathematical puzzles (<b>Richards and Siegler</b>)</p> <p>(c) Accuracy of radiological diagnosis by medical students (<b>Lesgold et al</b>)</p> <p><u>Comparing ACT-R &amp; PDP models</u></p> <p>Children's learning of past tense verbs can be modelled in both ACT-R and PDP models. The central question is 'what causes the U shaped pattern of development?'</p> <p>PDP - <b>Rumelhart &amp; McClelland</b> - changes in accuracy are caused as new verbs are acquired. At first, each verb is dealt with by a PDP network uniquely. This strategy has to be revised as more words are learnt than available units. The network initially over-generalises, before it being corrected through further training - equivalent to <u>feedback</u> in human learning.</p> <p>ACT-R - <b>Taatgen &amp; Anderson</b> - initial learning is in declarative memory as chunks. A process of production compilation leads to a somewhat unreliable production rule for past tenses. A blocking mechanism for the rule then develops; declarative memory is used to deal with irregular verbs.</p> <p>Evaluation: both models approximate to human behaviour. The PDP model is particularly problematic in its requirement for feedback - something that does not always occur in real life. ACT-R does not require feedback to correct its initial production rule.</p> <p>PDP models closer to <b>Marr's</b> hardware level; are sub-symbolic; ACT-R is a symbolic model.</p> <p>Hybrid models have emerged - e.g. ACT-RN (<b>Lebiere and Anderson</b>) - implements the goal stack and chunks as networks. Features of ACT-R that are not neurologically plausible have also been removed - e.g.</p>
--	--	---

slots are no longer allowed to contain a long list of items.

**When is a model a good model?**

Three evaluation criteria:

- (i) Extent to which the model fits observed human behaviour
- (ii) Psychological validity of the model
- (iii) Parsimony of the model

However, some negatives/positives:

(a) Too close a fit to human behaviour as derived from experiments does not necessarily mean it is the best model - **Pitt & Myung**. Over-fitting the data may mean that too many auxiliary assumptions make the model not general enough.

(b) Each part of ACT-R (declarative & procedural memory distinction, goal stack etc.) can be compared against psychological research and debated.

(c) Occam's razor needs to be applied.

Cognitive modelling still lacks clear methods of evaluation for models, but these three criteria can be applied to judge both PDP and rules-based models.

**Anderson & Lebiere**, after **Newell** - propose the "Newell Test" as a possible way forward:

Newell test criteria applied to ACT-R:

1. Be universal - act as a fn of the environment +
2. Operate in real time +
3. Exhibit rational adaptive behaviour +
4. Use vast amounts of knowledge
5. Robust against error & the unknown +
6. Integrate diverse knowledge
7. Use natural language -
8. Exhibit self-awareness -
9. Learn from its environment +
10. Acquire capabilities via development
11. Arise through evolution
12. Be realisable within the h/w of the brain -