Production Compilation: A versatile cognitive mechanism

The goal of cognitive modelling with cognitive architectures is to explain as much of cognition as possible using only a small set of explanatory mechanisms. In this article I will focus on a particular mechanism, production compilation,[1] within a particular architecture, ACT-R.[2] Production compilation is a mechanism that learns new production rules on the basis of existing rules and declarative knowledge. The basic mechanism is very simple, but the combination with other ACT-R mechanisms and some general cognitive strategies allows for a broad spectrum of models that all entail learning a particular skill on the basis of discovery, experience and instruction.

ACT-R

A fundamental assumption of the ACT-R theory is the distinction between a declarative (fact) and a procedural (rule) memory. Declarative memory contains pieces of knowledge that are open to conscious inspection, consisting of episodic knowledge, self-derived facts and knowledge gained through perception. Each fact in declarative memory has an activation value that reflects its past relevance and connection to the current context. Procedural memory contains rules that act upon the current goal and state of the perceptual and motor systems. With each rule a utility value is maintained that represents an estimate of its success rate and costs. Production rules are not open to conscious inspection.

Production compilation

The production compilation mechanism generates a new rule whenever two rules fire in sequence. The two rules are combined into one new rule that performs the same function as the two original rules. If the two rules interact with declarative memory, i.e., when the first rule request a certain fact from declarative memory and the second rule uses the knowledge in some fashion, this fact from memory is substituted directly into the new rule. Although production compilation always specializes rules, it can nevertheless achieve some sort of generalisation when the rules comprise some general cognitive strategy that is applied to a specific example. I will illustrate this by giving two examples.

Example 1: Learning the English past tense

When children learn the past tense, they at some point discover the regularity in regular past tenses, which is that the past tense is the stem plus *–ed*. This rule can be learned by a model that tries to use an analogy strategy to generate past tenses.[1] Given a verb that has to be inflected for past tense, the model attempts to retrieve an example from declarative memory (see Figure for an example). If this example is not the word we try to find (e.g., we try to retrieve the past tense of *walk* but instead find the past tense of *work*) then the model will try to find a pattern in the retrieved example an apply this to the current verb. In the case of *work – work-ed,* the pattern is to add *–ed* after the stem. Production compilation will specialise the analogy strategy on the basis of the —in this case— regular example, producing the regular rule. The model produces the U-shaped learning effect associated with learning the past tense, and does so without external feedback on its own behaviour, contrary so several existing models of the past tense.

Example 2: Learning Air Traffic Control

When people have to learn a new complicated task, they are initially very slow and make many errors. As experience progresses behaviour becomes faster, and the number of errors decreases. This behaviour can be explained by a model that starts with a declarative representation of the task instructions and production rules that interpret these instructions. After some initial practice on a new task, experience will build up, and those experiences can be retrieved later when a similar situation occurs. We[3] modelled this process in the case of the Kanfer-Ackerman Air Traffic Controller task,[4] a task in which several types of planes have to be landed on several different types of runways in different weather conditions. The Figure shows two examples of how production compilation can produce efficient rules that produce a significant speed-up in performance. In the first example a declarative instruction is compiled into a production rule. After the rule is learned the model no longer has to retrieve the instruction, but can act at once at the appropriate moment. The second example shows how previous experience is compiled into a rule: in this particular case that a DC10 can be landed on a wet runway. The model is able to fit human data across learning sessions at a general level (number of planes landed), at the unit task level (time to complete certain sub-tasks), and at the level of individual keystrokes.

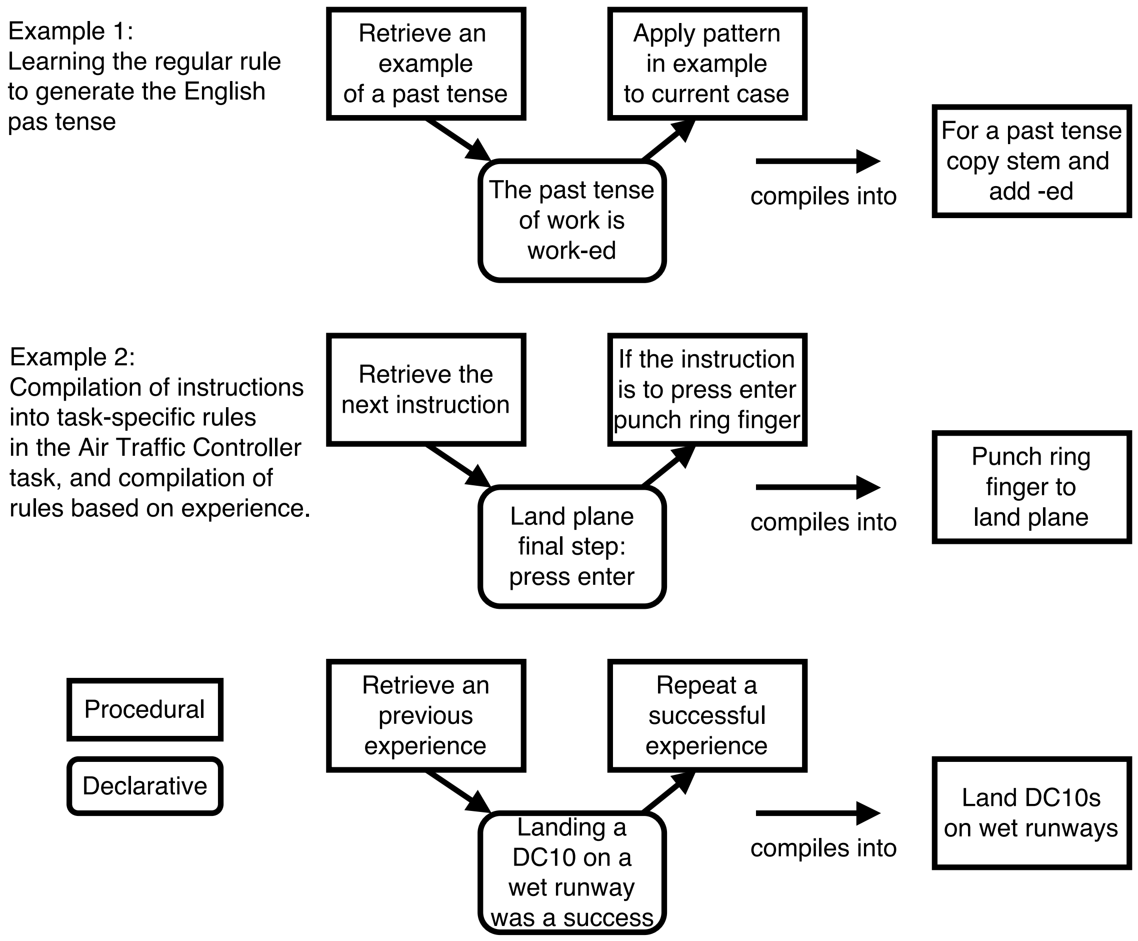
References

1. N.A. Taatgen and J.R. Anderson, *Why do children learn to say "broke", a model of learning the past tense without feedback,* **Cognition 86** (2), pp 123-155, 2002.
2. J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere and Y. Qin, *An integrated theory of the mind,* **Psychological Review**, in press.
3. N.A. Taatgen and F.J. Lee, *Production Compilation: A simple mechanism to model Complex Skill Acquisition,* **Human Factors 45** (1), pp 61-76, 2003.
4. P.L. Ackerman, *Determinants of individual differences during skill acquisition: Cognitive abilities and information processing*. **Journal of Experimental Psychology: General 117**, pp 288-318, 1988.

Figure captions

Figure. Examples of the production compilation process

Niels Taatgen
Carnegie Mellon University, Pittsburgh, USA and
University of Groningen, Netherlands
E-mail: taatgen@cmu.edu
http://www.ai.rug.nl/~niels


Example 1:
Learning the regular rule
to generate the English
pas tense
Retrieve an example of a past tense
Apply pattern in example to current case
The past tense of work is work-ed
compiles into
For a past tense copy stem and add -ed
Example 2:
Compilation of instructions
into task-specific rules
in the Air Traffic Controller
task, and compilation of
rules based on experience.
Retrieve the next instruction
If the instruction is to press enter punch ring finger
Land plane final step: press enter
compiles into
Punch ring finger to land plane
Procedural
Declarative
Retrieve an previous experience
Repeat a successful experience
Landing a DC10 on a wet runway was a success
compiles into
Land DC10s on wet runways