Production Compilation

Niels Taatgen
Artificial Intelligence
University of Groningen, Netherlands
Contents

• Goals of the new mechanism
• How does it work in ACT-R 4.0?
• Some examples
  – Learning the past tense
  – Learning the balanced beam
  – Learning from instructions/skill acquisition
• Implementation in 5.0
• Introduction of production compilation raises some issues
Goals of production compilation

- Gradual speedup as declarative knowledge is compiled into procedural knowledge
- Discontinuous learning: “sudden jumps”
Production compilation: specialization and combination

- General idea: two rules that fire in sequence are combined into one
- Production rules are specialized by substituting the retrieval into the rule (or, in 5.0, factoring out the retrieval request and retrieval match)
- Production rules are combined by merging two rules into one
Schematic overview (4.0)

(p rule1

Goal Match

Retrieval

==>

Goal Modification

)

(p rule2

Goal Match

Retrieval

==>

Goal Modification

)

Matches

Chunk from Declarative Memory

Two rules fire in sequence
Specialization

(p rule1

Goal Match

Chunk from Declarative Memory

==> Matches

Goal Modification


)

(p rule2

Goal Match

 Retrieval

==>

Goal Modification

)

Chunk from Declarative Memory
Specialization

(p rule1

Goal Match

Chunk from Declarative Memory

==>

Goal Modification

)==>

Specialization instantiates variables

(p rule2

Goal Match

Retrieval

==>

Goal Modification

)
Specialization

(p rule1

Goal
Match

==> 

Goal
Modification

)

(p rule2

Goal
Match

Retrieval

==> 

Goal
Modification

)
Combination

(p rule1

Goal Match

==>

Goal Modification

)}

(p rule2

Goal Match

Retrieval

==>

Goal Modification

)
Combination

(p rule1 => Goal Modification)

(p new-rule => Goal Match Retrieval => Goal Modification)

(p rule2 => Goal Match)

(new-rule => Goal Modification)
The new rule

(p new-rule
  Retrieval
  Goal Match
  ==> Goal Modification
)
How are rules introduced?

- Rule learning only occurs if parent rules have sufficient experience (!)
- New rule inherits parameters from parents, except for the number of initial experiences
- New rule receives an initial penalty on expected gain to ensure gradual introduction
Properties of the new compilation

• It is implicit
• It supports a transition from declarative to procedural knowledge
• Doesn’t produce illegal or short-circuiting rules because all new rules are specializations of existing rules
Support for gradual speedup

- By factoring out retrievals
  - Speedup because retrievals are no longer needed
  - It reduces errors (retrieval errors)
  - Frees up “working memory” by reducing interference (example in KA-ATC model)
Discontinuous learning

- In declarative memory, *activation* (how often is a chunk used) determines its success.
- In procedural memory, *expected gain* (how efficient is the use of knowledge) determines its success.
- A shift from declarative to procedural may produce transition effects: for example, U-shaped learning.
Example: Learning the past tense

- At some point during language development, children discover and use the regular rule of past tense inflection:

  To get the past tense of a verb, add –ed to it.

- The regular rule can be learned by applying production compilation to two productions that implement analogy.
Declarative Memory

Current goal

Environment (i.e., parents)

stem

work
work ed

break
broke

break ed

break

? ?

break ed

break

past tense

break
Regular rule is learned by specializing Analogy

\[
\text{(p copy-suffix)} \quad \Rightarrow \quad \text{Matches} \quad \Rightarrow \quad \text{(p copy-equal-slots)}
\]

**Example:**
- Work -> work
- Work -> work ed
Regular rule is learned by specializing Analogy

\[(p \text{ copy-suffix} = \text{stem nil}) \Rightarrow (p \text{ copy-equal-slots} = \text{stem nil})\]

\[\text{work work ed} \Rightarrow \text{work work ed}\]

Since this is the same chunk, it also instantiates the second retrieval

\[\Rightarrow \text{stem ed} \Rightarrow \text{stem}\]
Regular rule is learned by specializing Analogy

(p learned-regular-rule

=stem
nil nil

==> 

=stem
=stem ed

)
U-shaped learning
Important aspect of past tense

• In the “base-level activation” domain, using -ed examples is rare, because they have low activations (regular verbs are low-frequent)

• In the “expected gain” domain, using -ed is frequent, because it is an efficient strategy (you can add -ed to any verb you want)
Example: Balanced Beam
(work by Hedderik van Rijn)

• Stage 1: Only look at weights
• Stage 2: Look at weights, and if they are equal, look at distance
Pre-stage 1 strategy

Select a feature

Retrieve feature values

equal?

Yes

Start again

No

Base answer on difference

The most promising feature for a beam is weight, so weight is a prime candidate for selection.
Stage 1 strategy

Select a feature

Retrieve feature values

equal?

Yes

Start again

No

Base answer on difference

Repeated use of “weight” as feature will proceduralize the first two steps

Select weight and retrieve the weight values

equal?

Yes

Start again

No

Base answer on weight
Stage 1 --> Stage 2

Select weight and retrieve the weight values

Select a feature

Retrieve feature values

Equal?

Yes

We further assume some instructional or attentional process marks “distance” as an applicable feature (by boosting activation)

No

Base answer on weight

We replace the “Start again” by the original strategy.

Start again

Equal?

Yes

No

Base answer on difference
Stage 2

Select weight and retrieve the weight values

- equal? (No: Base answer on weight; Yes: Select distance and retrieve the distance values)

Select distance and retrieve the distance values

- equal? (No: Base answer on distance; Yes: Start again)
Learning from instructions

- If the chunk that is specialized is an instruction, we can have general task-independent productions that interpret task-specific chunks.
- We therefore need retrieve and interpret instructions
- More on this in the instruction session
Skill acquisition

• Learning from instructions is the first step in the acquisition of new skills

• Preview of ICCM talk: model of the KA-ATC task
The present model

Instructions

Representation of what to do (incomplete!)

Rules to interpret instructions

Rules to retrieve and carry out representations of what to do

Perceptual Motor Interface

Declarative

Procedural
Skill acquisition involves gaining experience.

- Instructions
- Representation of what to do (incomplete!)
- Rules to retrieve and carry out representations of what to do
- Experience
- Rules to interpret instructions
- Perceptual Motor Interface
- Declarative
- Procedural
Skill acquisition also involves learning new rules.

- Representation of what to do (incomplete!)
- Rules to retrieve and carry out representations of what to do
- Experience
- Rules that carry out the task
- Perceptual Motor Interface
- Declarative
- Procedural
Production compilation

- Merge two rules into one
- While substituting the retrieved chunk into the new rule
Production compilation based on experience

Retrieve an successful experience to land the current plane type given the current runway condition on a certain runway

Landing a DC10 when the runways were DRY was successful on the short runway

If the plane is a DC10 and the runway is DRY, take the short runway

Decide to land the plane on the runway from the retrieved experience

Procedural

Declarative
Schematic overview (5.0)

(p rule1

Goal Match

==> Retrieval request

Goal Modification

Matches

Chunk from Declarative Memory

(p rule2

Goal Match

==> Retrieval

Goal Modification

Retrieval request
Specialization

(p rule1

Goal Match

==> Chunk from Declarative Memory

Goal Modification

Matches

Chunk from Declarative Memory

(p rule2

Goal Match

==> Chunk from Declarative Memory

Goal Modification

Retrieval request

Chunk from Declarative Memory)
Specialization

(p rule1

Goal Match

==> Chunk from Declarative Memory

Goal Modification)

(p rule2

Goal Match

Chunk from Declarative Memory

==> Goal Modification

Retrieval request

Specialization instantiates variables
Specialization

(p rule1

Goal Match

==>)

Goal Modification

)}

(p rule2

Goal Match

==>)

Goal Modification

Retrieval request

}
Combination

(p rule1

Goal Match

==> Goal Modification

)
Combination

(p new-rule

Goal Match

==> Retrieval request

Goal Modification

)
Issues in production compilation

- You cannot properly use production compilation without switching on all the other learning mechanisms and expected gain noise. You need at least base-level learning and production parameter learning for it to work at all.
Issues in production compilation

- Production compilation emphasizes the need to work towards models that contain as little task-specific knowledge as possible, except in the form of instructions
  - Declarative instructions
  - General problem solving productions (e.g., retrieval, analogy)
Issues in production compilation

- Interaction with declarative memory
  - Initially, selected of knowledge is based on activation: how often does it occur in the environment, and how often is it retrieved
  - Eventually, selection of knowledge is based on expected gain: how useful is the knowledge.

- Potential evidence: U-shaped learning
Issues in production compilation

• We want the new rule to gradually replace the parent rules, but only if the new rule is better
The case of fixed noise

New rule has a eventual PG-C of 10.5

Parent rule (red) has a PG-C of 10.

new rule receives a cost-penalty of 1

Probability of using new rule goes up
Learned rule is worse with fixed noise

- New rule has an eventual PG-C of 9.5
- Parent rule (red) has a PG-C of 10.

- Probability of using new rule goes up but should go down
- New rule receives a cost-penalty of 1
One idea: decreasing noise

- New rule has an eventual PG-C of 10.5
- Parent rule has a PG-C of 10.
- Probability of using new rule goes up.
- New rule receives a cost-penalty of 1.

- One idea: decreasing noise.
If the new rule is worse than the parents...

If the new rule is worse than the parents...

- Parent rule has a PG-C of 10.
- New rule has an eventual PG-C of 9.5.

Probability of using new rule goes down.
Gradual introduction has the same effect as the discontinued strength learning.

![Graph showing latency over experiences]
Issues in production compilation

- Addition of production compilation closes the learning cycle, with the possibility of more autonomous behavior
- And it raises the bootstrapping question
Some final issues

• May solve the baselevel learning decay problem: is it 0.5 (short-term) or 0.3 (long-term)?

• Retrieval failures proceduralize are hard to proceduralize

• How to learn productions without goals? How does production learning handle other buffers?