# Learning without limits

**Niels Taatgen**
Cognitive Science and Engineering
University of Groningen
Grote Kruisstraat 2/1
9712 TS Groningen, Netherlands
+31 50 3140486
niels@tcw3.ppsw.rug.nl

One of the most pervading criticisms of rule-based models of reasoning is that they are "pre-programmed". Cognitive models based on production systems, including most ACT-R models, already have the necessary rules in memory to perform the specific task they model. This type of modeling has a number of problems. A first problem is that rules for a new task have to be learned, and that it is not always reasonable to suppose that all of the rules have been acquired during instructions. A second problem is that in complex problem-solving situations, a model based on a fixed set of production rules cannot reason "outside the system", a problem often described by the term "brittleness." Many discussions have been devoted to this issue in a criticism of artificial intelligence in general, often invoking Gödel's theorem as definite proof that a rule-based approach cannot work.

A potential solution to this problem is to add learning to the system. The simplest solution is to add a learning module to the reasoning system that adds new knowledge to memory based on experience. Many learning algorithms have been proposed, often involving some sort of induction. This solution, however, is insufficient. By adding a fixed learning algorithm, the problem of "brittleness" and "being unable to reason outside the system" is now shifted to the learning algorithm. So the learning algorithm should be flexible itself, requiring a learning learning algorithm.

Instead of heading towards an infinite regress of learning-learning-learning algorithms, there is another solution, a solution with a clear link to human information processing, and a solution that is entirely consistent with ACT-R's learning and the idea that Thought is Adaptive. This solution assumes learning consists of two parts, a fixed part (implicit learning) and an adaptive part (explicit learning). The fixed part is part of the cognitive architecture, and acts in a more or less syntactic, automatic fashion. The adaptive part of learning is not a module by itself, but rather consists of knowledge in memory. Although knowledge in memory cannot change other knowledge directly, it can accomplish changes by putting the implicit learning system to work. It can do this by setting learning goals, and using learning strategies to act on these goals and gather or derive new knowledge. So in addition to learning mechanisms, we have learning skills.

In terms of ACT-R, the fixed, implicit part of learning consists of the learning mechanisms of the architecture. Explicit strategies can be represented by both production rules and declarative knowledge. An example of an explicit strategy is to use analogy. This strategy can be supported by declarative knowledge in the form of plans or schema's that can serve as the basis for an analogy.

Alan Newell once proposed a diagram with a time scale of human action. I propose a similar time scale of human learning:

| | | | |
|---|---|---|---|
| Developmental band | | | |
| $10^8$ | years | learning strategies | procedural |
| $10^6$ | weeks | reusable plans and schemas | declarative |
| Skill band | | | |
| $10^4$ | hours | task-specific rules | procedural |

| | | | |
|---|---|---|---|
| $10^2$ | minutes | task-specific plans | declarative |
| Instance band | | | |
| $10^0$ | seconds | task-specific examples | declarative |

In order to support these ideas, I have developed a number of cognitive models.

- A model of an implicit/explicit memory task by Tulving et al., demonstrates that there is no need for separate implicit and explicit memory systems
- A model of a balance-scale task and discrimination-shift learning. Demonstrates re-usable learning strategies and differences in strategy due to development.
- A model of alternating search and reflection strategies in a rational fashion. Demonstrates that meta-learning is unnecessary
- A model of the Fincham task. Demonstrates several types of representation are needed to fully explain learning a complex task.
- A model of scheduling. Demonstrates a model that learns its own task-specific rules by re-using plans for other tasks in declarative memory.

All these models are described in my dissertation. The dissertation and all of the models can be obtained from:

http://tcw2.ppsw.rug.nl/~niels/thesis

One of the remaining problems in mapping this approach onto ACT-R is procedural learning. In my view, procedural learning is an implicit learning mechanism, while in the current ACT-R it is goal-driven (by dependency chunks).

## Learning without Limits

Niels Taatgen

- Theory
- Evidence

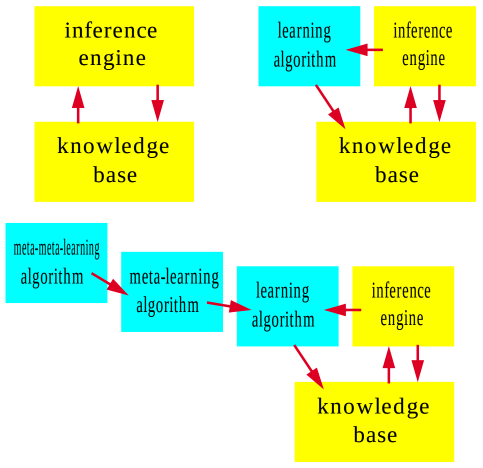
## The problem of computational models of complex reasoning

- Models are “pre-programmed”
- Models cannot step “outside their boundaries”
- “Brittleness”

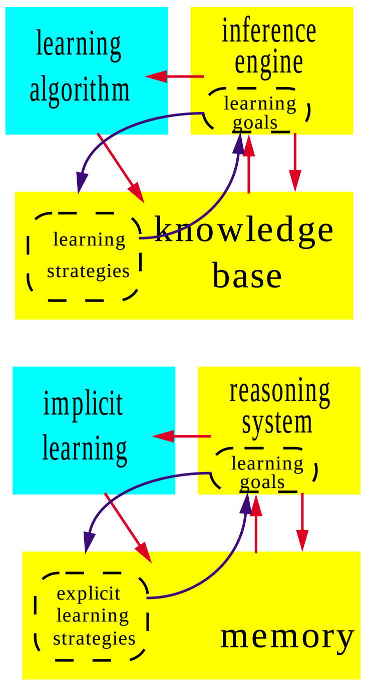
More specifically:

- A model already starts out with the task-specific knowledge it needs to accomplish the task.

## Reasoning systems (1)



## Reasoning systems (2)



## In terms of ACT-R

**Procedural learning strategies**

- Analogy
- Hypothesis-and-test
- Rehearsal

**Declarative strategies (plans)**

- Plan to build lists
- --> Reusable knowledge

## Learning time scale

Time scale of human learning

| Scale (sec) | Time Units | Type of representation | Memory system used | |
|---|---|---|---|---|
| $10^8$ | years | Learning strategies | Procedural | Develop–mental band |
| $10^6$ | weeks | Generally useful declarative rules | Declarative | |
| $10^4$ | hours | Task-specific production rules | Procedural | Skill band |
| $10^2$ | minutes | Task-specific declarative rules | Declarative | |
| $10^0$ | seconds | Task-specific facts | Declarative | Instance band |

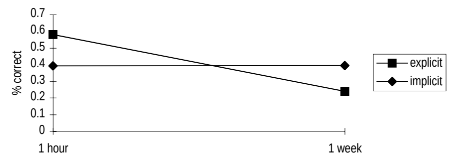
## Implicit vs. Explicit memory

Isn't there a separate implicit memory system and a separate explicit memory system, as evidenced by dissociation experiments?

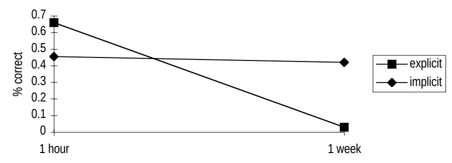
Tulving, Schacter and Stark experiment

- Explicit knowledge degrades in time
- Implicit knowledge is stable

**Data**



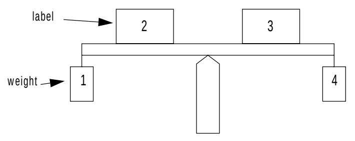
**Model**

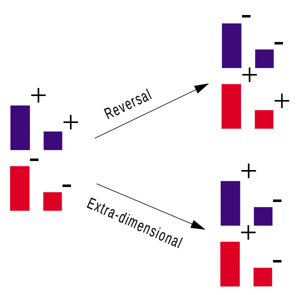


## Re-usable strategies (1)

It is possible to define some learning strategies that can learn several different task?

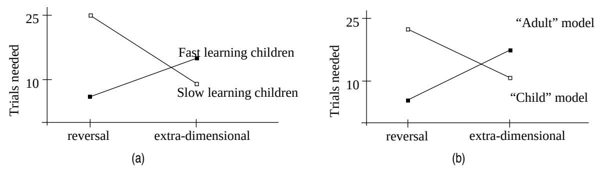
Model of balance problem / discrimination-shift learning demonstrates this aspect





## Re-usable strategies (2)

- The same set of rules can learn both the balance problem and the discrimination-shift problem
- When some rules are removed, the model behaves like a small child



## The scheduling problem

In the scheduling problem, a number of tasks have to be assigned to workers in a time schedule, satisfying a number of constraints. Example:

There are 2 workers with 6 hours each

Task A takes 1 hour
Task B takes 1 hour
Task C takes 2 hours
Task D takes 2 hours
Task E takes 3 hours
Task F takes 3 hours

Task C must be before task B
Task D must be before task C
Task E must be before task A
Task F must be before task B

Solution:

Worker 1: DFA
Worker 2: ECB

**This problem is NP-complete**

## Example verbal protocol

So you must say... We will try to combine G and B. So we have GB and need one of three. So F...FGB... that is possible, cause G is before B... FGB... than F should go before D.. but if you do... FGD I said, let's put my fingers there.. FGD oh FGB yes... than E must E is two hours so E should... not necessarily at the beginning but I must have something before A.. but A must be at the end.. and D must be before C. So you would say.. eh.. E...D is impossible because F is not before D. E... I have to add one, which can't be A cause it should go at the end. E... No they can't go together cause EDC is impossible. Ehm.... EDC that was.. that was only five hours anyway. So you have to say... FGB.... FGB... and than E.... EI....C..EIC.. ehm... what is left.. Yes seven hours together...

## Modelling scheduling behaviour

The model starts out with

- The general strategy of analogy
- The general strategy of rehearsal
- General declarative plans to build lists, fill containers, add and subtract
- Production rules to interpret declarative plans
- Production rules to proceduralize declarative plans
- Some additional declarative plans for scheduling (the only task-specific ingredient)

## Results of the model

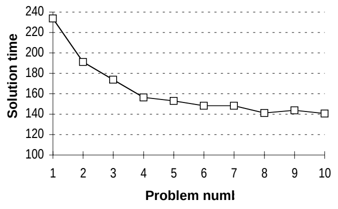
- It can schedule! (only simple schedules, however)
- The analogy strategy adapts declarative plans to the current situation
- Declarative plans are gradually proceduralized, resulting in shorter solution times and less errors.
- It produces pseudo-verbal protocols that get shorter when the model has had more practice
- Most importantly, it demonstrates how a complex task can be learned without initial task-specific knowledge

## Example protocol

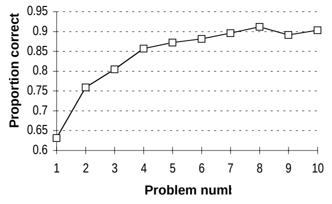
First I will find something to begin with. Begin with F. F.. Now I have to find the next thing. F before A. A.. Now I have to find the next thing. No more items for the list, let's check whether we're done. F.. A.. Is this a schedule for one worker or for more? Now I am going to count how many hours we already have F.. Add this to what we have. Nothing plus three equals three. A.. Add this to what we have. Three plus one equals four. Do we have enough for one worker? No, the schedule is not full, yet. F.. A.. Now find the task that fits in. Task C takes two hours. C.. We can move to the next worker.. NEXT-WORKER Let's do the rest F.. A.. C.. NEXT-WORKER.. I now try to find any unused order constraints. B before C. B before C. This one hasn't been used, so the constraint has been found. B before C. B before C. B.. Now we are going to look at all the tasks, and see which ones are not yet in

## Solution times, errors and learned productions

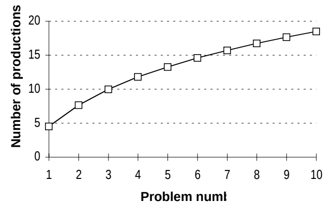
Solution time



Proportion solved



Rules learned



## Conclusions

Human learning is too complicated to be explainable by a handful of learning principles.

Understanding what strategies people use for learning, and how these strategies themselves develop is crucial to understanding human learning.

Lots of future work!