A Standard Model of Mind:
AAAI Technical Report FS-17-05

Cognitive Architectures: Innate or Learned?

Niels A. Taatgen
Institute of Artificial Intelligence, University of Groningen
Nijenborgh 9, 9747 AG Groningen, Netherlands
n.a.taatgen@rug.nl

Abstract

Cognitive architectures are generally considered to be theories of the innate capabilities of the (human) cognitive system. Any knowledge that is not innate is encoded in the architectures memory systems, either by the modeler or learned by the architecture itself. However, in human intelligent behavior few things are innate. An alternative is to acknowledge that learning occurs at different levels of abstraction. A standard model of the mind should therefore span multiple levels of abstraction, encouraging research efforts to establish learning mechanism that connect them.

Human babies are born almost completely helpless. Although developmental psychology has a lot to say about what newborns are already capable of, they acquire almost everything they need to know at some stage in life. This is why the human species is so successful: it can adapt to many different circumstances, and is therefore, with the same genome, successful in current day's society just as well as, say, the time that we were still hunter-gatherers.

Piaget (1952) argued that child development goes through a number of stages, starting with a sensorimotor stage, and progressing to more cognitive and abstract stages with age. Even though developmental psychologists still debate the existence of concrete stages, it is clear that learning in an infant is quite different from learning in an adult.

Current cognitive architectures, such as ACT-R (Anderson 2007), Soar (Laird, Newell, and Rosenbloom 1987) and PRIMs (Taatgen 2013), have a somewhat more simple view of development. The assumption is that the cognitive architecture represents the innate cognitive capabilities of intelligence, and that everything that is learned is represented as knowledge and skills in the memory systems of the architecture (Laird, Lebiere, and Rosenbloom in press). This may seem a reasonable assumption, but it carries along a big problem. Whenever we want to create a model of a specific task, we have to encode the knowledge to perform that task in terms of knowledge in the memory of the architecture. However, the architecture's memory is empty, which means that there is a gap between what the architecture can do by itself, and the requirements of the task. The modeler has to fill this gap with ad hoc solutions that are probably not completely justified, but that help to achieve the particular goal of modeling the particular task. However, this approach means that many models, even within the same architecture, are completely incompatible with one another. Therefore, it has the danger that it runs into the same problem that Newell (1973) signaled in psychology in general: that it produces a collection of micro-theories for specific phenomena, but that it fails to achieve a unified theory of cognition. Moreover, certain domains that are relatively late in development, such as language and social cognition, are very hard to model in an architecture where knowledge starts from scratch, because they require large amounts of prerequisite knowledge. If cognitive architectures want to break new grounds, they have to acknowledge this problem, and look for new solutions.

Horizontal vs. Multilevel Architectures

Newell (1973) identified multiple levels of abstraction to study human cognition, ranging from the level of organelles that operate in the 100 μs time range to social cognition that operates at a time scale of months. What Newell considered the levels that are most interesting for cognition are in between these extremes: the level of deliberate acts (100 ms), operations (1 sec), and unit tasks (10 sec). It is tempting, and consistent with ideas that go back to Turing (1950), to pick a level of abstraction as the base level to model human cognition, consider all levels below that level as implementation, and build up the theory from that level. I call this a *horizontal architecture*, because it focusses on a single level of abstraction as a starting point both up and down (Figure 1). The developmental literature suggests another approach, in which we take into account that learning takes place at each level of abstraction. I call this a *multilevel architecture*. The central idea in the multilevel architecture is that there is not a particular level of abstraction that is "special". Depending on the phenomenon we want to model, we have to select one or more levels of abstraction that are most appropriate. Therefore, the neural level is not the most appropriate to model learning the past tense (Taatgen and Anderson 2002), and object recognition is hard to model at a symbolic level.

Multiple Levels of Learning

Human learning can takes many different forms. If students are cramming for an exam, trying to force facts into their

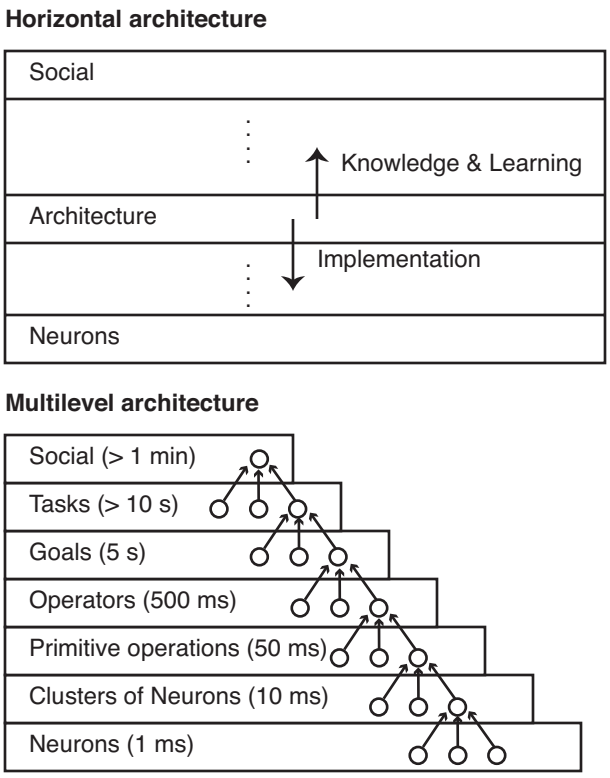

Figure 1: Horizontal vs. Multilevel architectures. In the multilevel architecture, composition takes units from a lower level to compose new units at the higher level, while evaluation processes prioritize units within a level.

memories, they are doing something different then when they practice solving equations. Children learning their native language seem to be able to acquire it by mere exposure, whereas learning a second language requires a more explicit learning effort. Cells in the visual cortex learn to detect particular patterns of light, eventually enabling use to recognize complex objects. In order to explain different types of learning, we often need different mechanisms, ranging from associative learning, reward-driven learning, example-based learning to explicit reasoning. Interestingly enough, these learning mechanisms also play out at different time scales. In particular learning mechanisms that are related to neural networks require huge numbers of training cycles, whereas explicit reasoning may produce ”one-shot” learning, where only a single exposure is enough to learn something new.

How many learning mechanisms do we need in a cognitive architecture? If we take the horizontal approach, we have to try to capture all learning with as few mechanisms as possible. The more mechanisms for learning we have, the more difficult it becomes to determine what mechanisms of learning should be used in what situation. For example, in ACT-R there are several experimental results that can be explained by both procedural (utility) learning as by declarative (instance-based) learning.

The different time scales that learning mechanisms operate at are a clue that we are dealing with different levels of abstraction, but now in a reverse order compared to time time scale that they operate on: learning at the lowest levels takes most time, whereas learning at the highest levels can be ”one-shot”, so very fast. Also, learning follows a developmental trajectory in which learning starts at the lowest levels of abstraction, and gradually builds up to the highest (hence the staircase in Figure 1).

Learning in a multilevel architecture does not involve reinventing the wheel, but putting different learning mechanisms in the right perspective. As Laird et al. (in press) indicate, there are two categories of learning: *composition* and *tuning*. When new knowledge is composed, we take multiple elements that we combine into one new element. Although this may happen within a level of abstraction, for example when multiple pieces of declarative knowledge are connected into a new declarative item, it may also mean that we take multiple elements from a lower level and combine them into a single unit at a higher level. In other words, the lower level supplies the primitives for the the higher level. But primitives are never “truly” primitive, because they themselves are rooted in even lower levels of abstraction. In the case of tuning, knowledge items are evaluated, leading to decisions on what to retain. The input for this tuning process may be based on rewards or other local information, but also on information that feeds back from higher levels.

Possible Levels and their Learning Mechanisms

What are possible meaningful levels of abstraction? Although I consider this an open question, there are a number of likely candidates.

Neural and Clusters of Neurons Level

The neural level is an obvious level of abstraction, even though there is debate about the right level of detail. Neural models can be very powerful, because they acquire all their knowledge through learning, even though this learning is typically very slow. There are a number of attempts to build a complete cognitive architecture based on neural networks, and the programs in that direction aim at general-purpose systems that have strong correspondences with traditional cognitive architectures (Stocco, Lebiere, and Anderson 2010; Eliasmith et al. 2012). Neural architectures use clusters of neurons to represent knowledge, where particular patterns of activation represent certain concepts. Models build within these architectures take a lot of time and ingenuity, and large amounts of training, for relatively easy tasks. For example, the model by Stocco requires an elaborate neural network for aural-vocal choice reaction time task that requires only a few production rules in ACT-R.

Although performance at the neural level is in terms of milliseconds, learning is slow. Learning in neural networks can be done unsupervised, in which the network has to discover meaningful patterns itself, or supervised, in which the network is supplied with the correct answer. In this latter case the question is what the source is of the correct answer.

Primitive Operations

If clusters of neurons represent concepts, we need methods to do something with those representations. For example, a standard neural network may take a visual image and turn this in a motor command. This means that a pattern of activation is transformed and forwarded through several layers of neurons in order to produce an output. If we want to generalize this to a more task-general approach, activation patterns in the network have to routed through the network depending on the particular task it carries out. We can specify such routing patterns in terms of primitive operations. A single primitive operation can compare patterns in two different neural clusters, or forward information from one cluster to another.

We can abstract from the neural implementation by assuming a symbolic representation of activation patterns. Some details are lost in this abstraction, but the advantage is that the symbolic implementation is much more efficient. Primitive operations are the smallest building blocks of the PRIMs architecture (Taatgen 2013). If assume primitive operations are the basic building blocks, we are hiding a necessary learning process under the hood: activation patterns for the same symbol are typically not identical between neuronal clusters (e.g., the representation of a perceived table may be different from the representation of a remembered table), and therefore the network has to learn the mapping between the two representations through associative learning.

Operators or Productions

Our cognition system is capable of doing many things in parallel, but constraints on the task, the structure of the brain or the logic of the chosen strategy also force serial behavior. At the operator (Soar or PRIMs terminology) or production (ACT-R terminology) level, the smallest unit of knowledge represents what can be carried out in parallel in one step. It is also the level of abstraction that almost all (non-neural) cognitive architectures operate at. A challenge for the horizontal architecture that operates at this level is the question how the operators and/or productions are learned. Both ACT-R and Soar have solutions for this, but the problem is that they are based on knowledge of the same level of abstraction: in ACT-R, two productions rules are compiled into a single one rule (Taatgen and Lee 2003), whereas in Soar new productions and operators are learned through impasses that have to be resolved by other productions and operators. As a consequence, learning becomes an infinite regression with unknown origins.

However, ACT-R and Soar productions can be broken down into primitive operations from the lower level of abstraction (Taatgen 2013; Stearns, Laird, and Assanie 2017)[1]. This means that we can also see learning at this level as a composition process of lower-level primitive operations.

[1]Note that the Soar implementation uses production rules and operators at the same level of abstraction that are composed into larger units. This raises an interesting issue whether each level of abstraction should correspond to a level of implementation (as in computer architecture), or just as a level description.

Goals

Goals represent the minimal groups of operators that together form a meaningful unit. Whereas operators represent what can be carried out in parallel, goals support several sequential steps. I call them goals, because this is the name that is typically used within the cognitive architecture field, even though our everyday use of goals is much more encompassing. However, we can use goals in a slightly different way than is traditionally done in cognitive architectures. Instead of identifying goals with tasks, with the consequence that knowledge is compartmentalized into separate task units, we can also view them as flexible building blocks for tasks. As a consequence, we should consider goals as task-general pieces of knowledge that can be instantiated in particular situations (tasks). Learning goals is again a matter of composition: a goal is composed of a set of operators that carry out that goal.

Tasks

Eventually, we need all our knowledge to be able to apply it in particular contexts, which we typically refer to as *tasks*. To carry out a task, we have to select the right set of goals and instantiate them with the particulars of the task. This is the level of abstraction where we can perform one-shot-learning: to carry out a task, we just need to combine a few goals. It is, of course, only one-shot-learning if we have all the knowledge to carry out these goals. By separating tasks from goals, they become a much more flexible unit of knowledge than the typical goal in horizontal architectures. A task corresponds to a small knowledge structure that can be build on the fly, and that activates all the detailed knowledge necessary to carry out the task. If a particular task recurs frequently, it is worthwhile to retain the task knowledge structure, but otherwise it can be forgotten to be reconstructed when it is needed again.

Higher Levels...

The ability to use language provides a substantial boost in the ability of children to carry out tasks, because they can now receive instructions that provide top-down guidance. The construction of a task representation is probably closely linked to language and the semantics of language. It is no coincidence that formal education starts at around six years, when presumably enough of the groundwork has been laid by play, exploration and exposition to be able to benefit from such instruction. Social cognition is typically also attributed (at least by Newell) to higher levels. However, for me it is not entirely clear whether language and social cognition correspond to higher levels of abstraction (which would mean they are composed of multiple tasks), or that they just persist over longer periods of time (which is true for many other types of knowledge).

Example: Rapid Instructed Task Learning

An experimental paradigm that encourages thinking in terms of a higher level of modeling is *Rapid Instructed Task Learning* (RITL) (Cole et al. 2010; Cole, Laurent, and Stocco 2013). The key idea in RITL is that subjects receive a new

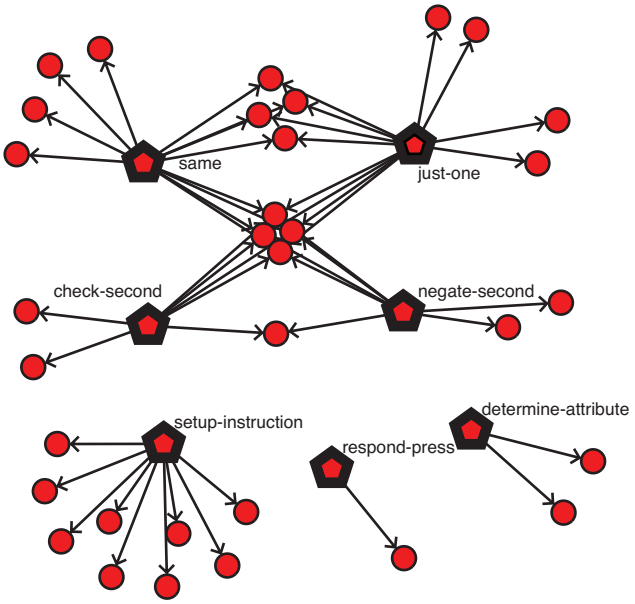

Figure 2: Prior knowledge necessary for the RITL task. Pentagons represent goals, while circles represent operators. Some of the goals share the same operators.

instruction on every trial in the experiment. For example, in Cole et al.'s experiments, subjects are presented with three words that describe the task, for example SAME – SWEET – LEFT INDEX. After a delay of a few seconds, this instructions is followed by two words to which the instruction has to be applied, for example Grape – Apple. In this case the instruction translates into: If the answer to 'is it SWEET' is the SAME for both words, press your LEFT INDEX finger. For each of the three words in the instruction there are four alternatives: four logical operators, four object attributes and four fingers, creating a space of 64 possible tasks. Cole at al. (2010) found that different brain regions are active when a subject receives a new task than a repeated task. In our own experiments, we found that repeated tasks are performed (slightly) faster than new tasks.

A model of the RITL task has to be able to construct a task representation on the fly. For this, it needs a number of goal primitives, which are displayed in Figure 2. For each of the four logical operators (*same, just-one, second, negate-second*) separate goals are present, even though these goals share operators. To determine the particular attribute value, a generic *determine-attribute* goal is used that is instantiated with the specific attribute that is in the instruction. The same is true for the *respond-press* goal that presses the instructed finger. Finally, the model includes a *setup-instruction* goal that is particular for this experiment, and that is a proxy for a more general task-construction process. Setup-instruction takes the three instruction words, and creates a task representation that ties together three goals with the appropriate instantiations. Figure 3 shows the structure it creates for the SAME – SWEET – LEFT INDEX task.

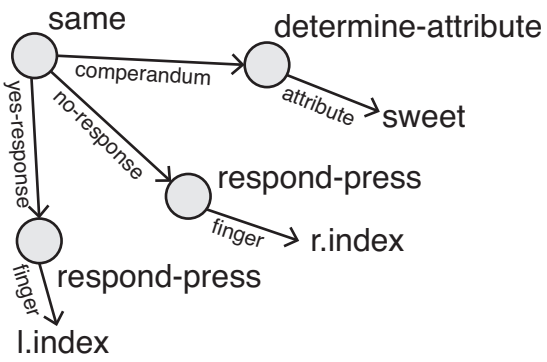

Figure 3: Task representation that the model constructs for the SAME – SWEET – LEFT INDEX task

Figure 4: The same (RITL) model, but now each of the white and grey circles represents a primitive operation

If we assume all the primitives at the goal level are in place, the simple graph in Figure 3 *is* the model of the task. This task-level model is much more simple (and therefore appropriate for the rapid changes in the RITL task) than the operator-level model in Figure 2, or even more so if we compare it to the same model, but now at the level of primitive operations in Figure 4.

Discussion

The standard model in physics describes the elementary units of the universe. Everything can, in theory, and including cognition, be described by building upwards from that model. A standard model of the mind is tougher to specify, because it is not clear what the best level of description is. Any proposal for a standard model, including Laird et al. (in press), will have a hard time to justify the basic unit of knowledge, because breaking down those units into smaller units may be necessary for a full understanding.

The reason is that a particular choice of level of abstraction can always be criticized because assumptions have to be made about the lower levels of representation (leading

to parameter fitting). The alternative, a multi-layer architecture, can allow the modeler to choose an appropriate level of abstraction, from which assumptions can be made about the primitives at the lower level. But by explicating these primitives (which are not primitives at the lower level), they can be investigated themselves. For this to work, levels of abstraction have to be connected together through composition. Composition mechanisms are different between levels, ranging from slow associative learning at the lowest levels to one-shot task model construction at the highest level.

The benefit of this approach is that primitives at a certain level can more easily be reused, potentially even between different modelers. This requires a somewhat better specification than modelers are used to, but has great potential rewards.

Many details still need to be investigated, and different solutions are possible and should be pitted against each other, but if the overall research program of cognitive architecture wants to survive, it needs to leave its comfort zone and face new tough challenges.

References

Anderson, J. R. 2007. *How can the human mind occur in the physical universe?* New York: Oxford university press.

Cole, M. W.; Bagic, A.; Kass, R.; and Schneider, W. 2010. Prefrontal Dynamics Underlying Rapid Instructed Task Learning Reverse with Practice. *Journal of Neuroscience* 30(42):14245–14254.

Cole, M. W.; Laurent, P.; and Stocco, A. 2013. Rapid instructed task learning: A new window into the human brains unique capacity for flexible cognitive control. *Cognitive and Affective Behavioral Neuroscience* 13:1–22.

Eliasmith, C.; Stewart, T. C.; Choo, X.; Bekolay, T.; DeWolf, T.; Tang, Y.; Tang, C.; and Rasmussen, D. 2012. A large-scale model of the functioning brain. *Science* 338(6111):1202–5.

Laird, J. E.; Lebiere, C.; and Rosenbloom, P. S. in press. A standard model of the mind: toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *AI Magazine*.

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence* 33:1–64.

Newell, A. 1973. You can't play 20 questions with nature and win. In *Visual information processing*. New York: Academic Press.

Piaget, J. 1952. *The origins of intelligence in children*. New York: International University Press.

Stearns, B.; Laird, J. E.; and Assanie, M. 2017. Applying primitive elements theory for procedural transfer in soar. In *Proceedings of the 15th international conference on cognitive modeling*.

Stocco, A.; Lebiere, C.; and Anderson, J. R. 2010. Conditional routing of information to the cortex: A model of the basal ganglia's role in cognitive coordination. *Psychological Review* 117(2):541–574.

Taatgen, N. A., and Anderson, J. R. 2002. Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition* 86(2):123–155.

Taatgen, N. A., and Lee, F. J. 2003. Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors* 45:61–76.

Taatgen, N. A. 2013. The nature and transfer of cognitive skills. *Psychological Review* 120(3):439–471.

Turing, A. M. 1950. Computing machinery and intelligence. *Mind* 59:433–460.