Running Head: CONSTRAINTS IN COGNITIVE ARCHITECTURES

Constraints in Cognitive architectures

Niels Taatgen and John Anderson

Constraints in Cognitive Architectures

1. Introduction

When Turing wrote his famous paper in which he asked the question whether machines can think, and how this can be tested (Turing, 1950), he set out the goal of creating an intelligent machine whose intelligence is indistinguishable from human intelligence. Turing's earlier work (Turing, 1936) proved that the basic digital computer's potential is as great as any conceivable computational device, suggesting that was only a matter of time before a computer could be developed that is as intelligent as a human. The exponential growth in power did not, however, turn out to make computers more intelligent, leading to a divergence into modern artificial intelligence and the smaller field of cognitive modeling. In modern artificial intelligence the main goal is to create intelligent programs, with the human intelligence aspect only as a source of inspiration, while cognitive modeling has taken the opposite route of focusing on faithfully modeling human intelligence, but not being really interested in creating intelligent programs.

Cognitive architectures are on the one hand echoes of the original goal of creating an intelligent machine faithful to human intelligence, and on the other hand attempts at theoretical unification in the field of cognitive psychology[1]. These two aspects imply a duality between functionality and theory. Cognitive architectures should offer functionality, i.e., representations and cognitive mechanisms to produce intelligent behavior. More choices in representation and mechanisms offer a larger toolbox to create a model for a certain phenomenon. But cognitive architectures should also be theories. A theory offers only a single and not multiple explanations

[1] Note that we will restrict our discussion to cognitive architectures that have the goal to model psychological phenomena.

for a phenomenon. From the theory perspective having many representations and mechanisms is not a good idea, because it increases the probability that many models can fit the same data. Functionality and theory are therefore generally conflicting goals, and different architectures strike a different balance between them. There are even cognitive architectures that primarily focus on the functionality aspect and have no or few theoretical claims (e.g., COGENT, Cooper & Fox, 1998).

The term cognitive architecture is an analogy of the term computer architecture (Newell, 1990; see also the discussion in the chapter by Sun in this volume (chapter 1)). A computer architecture serves as a flexible basis for a programmer to create any program. Similarly, a cognitive architecture allows modelers to create simulation models of human cognition. A model specifies the initial set of knowledge for the architecture to work with. For example, a model of multi-column addition might consist of as set of simple addition facts and a set of production rules that specify that you have to start in the right column, how to handle carries, etc. The classical method of finding this set of knowledge is through task analysis: a careful study of the necessary knowledge and the control structure associated with it. The knowledge specified in the task analysis is then fed into the architecture which can subsequently make predictions about various aspects of human performance, including reaction times, errors, choices made, eye movements and fMRI.

A problem of cognitive models is that it is not easy to assess their validity. One might assume that a model that produces the same behavior as people do is a valid model. However, several different models might produce the same behavior, in which case a different criterion is needed to determine which model is best. Unfortunately, there is no quantitative measure for model validity, but most cognitive modelers agree that the following qualitative factors contribute to the validity of a model:

- *A good model should have as few free parameters as possible*. Many cognitive architectures have free parameters that can be given arbitrary values by the modeler. Because free parameters enable the modeler to manipulate the outcome of the model, increasing the number of free parameters diminishes the model's predictive power (Roberts & Pashler, 2000)
- *A model should not only describe behavior, but should also predict it*. Cognitive models are often made after the experimental data have been gathered and analyzed. A model with high validity should be able to predict performance.
- *A model should learn its own task-specific knowledge*. Building knowledge into a model increases its specificity, and may decrease its validity.

As discussed above, many current models use task analysis to specify the knowledge that an expert would need to do the task. This violates the validity criterion that a model should acquire task specific knowledge on its own. Moreover, basing a model on a task analysis of expert performance means that the model is of an expert user whereas the typical user may not have mastered the task being modeled. Useful predictions and a complete understanding of the task requires that models are built starting at the level of a novice and gradually proceeding to become experts in the same way people do. In other words, many applications require building models that not only perform as humans do, but that also learn as human do.

2. Overview of Cognitive Architectures

In order to discuss the current state of cognitive architectures, we will briefly characterize four prime examples in this section, and then go through areas of cognitive modeling and discuss what constraints the various architectures offer in that area.

Soar

The Soar (States, Operators, And Reasoning) architecture, developed by Laird, Rosenbloom and Newell (1987; Newell, 1990), is a descendant of the General Problem Solver (GPS), developed by Newell and Simon (1963). Human intelligence, according to the Soar theory, is an approximation of a knowledge system. Newell defines a knowledge system as follows (Newell, 1990, page 50):

> A knowledge system is embedded in an external environment, with which it interacts by a set of possible actions. The behavior of the system is the sequence of actions taken in the environment over time. The system has goals about how the environment should be. Internally, the system processes a medium, called knowledge. Its body of knowledge is about its environment, its goals, its actions, and the relations between them. It has a single law of behavior: the system takes actions to attain its goals, using all the knowledge that it has. (...)

According to this definition, the single important aspect of intelligence is the fact that a system uses all available knowledge. Errors due to lack of knowledge are not failures of intelligence, but errors due to a failure in using available knowledge are. Both human cognition and the Soar architecture are approximations of an ideal intelligent knowledge system. As a consequence, properties of human cognition that are not directly related to the knowledge system are not central to Soar.

The Soar theory views all intelligent behavior as a form of problem solving. The basis for a knowledge system is the problem-space computational model, a framework for problem solving in which a search process tries to accomplish a goal state through a series of operators. In Soar, all tasks are represented by problem spaces. Performing a certain task corresponds to reaching the goal in a certain problem space. To be able to find the goal in a problem space, knowledge is needed about possible operators, about consequences of operators and about how to choose between operators if there is more than one available. If a problem (an *impasse* in Soar terms) arises due to the fact that certain knowledge is lacking, resolving this impasse automatically becomes the new goal. This new goal becomes a subgoal of the original goal, which means that once the subgoal is achieved, control is returned to the main goal. The subgoal has its own problem space, state and possible set of operators. Whenever the subgoal has been achieved it passes its results to the main goal, thereby resolving the impasse. Learning is keyed to the subgoaling process: whenever a subgoal has been achieved, new knowledge is added to the knowledge base to prevent the impasse that produced the subgoal from occurring again. If an impasse occurs because the consequences of an operator are unknown, and in the subgoal these consequences are subsequently found, knowledge is added to Soar's memory about the consequences of that operator. Because Soar can also use external input as part of its impasse resolution process, new knowledge can be incorporated into the learned rules.

Characteristic for Soar is that it is a purely symbolic architecture in which all knowledge is made explicit. Instead of attaching utility or activation to knowledge it has explicit knowledge about its knowledge. This makes Soar a very constrained architecture, in the sense that the only means to model a phenomenon are a single long-term memory, a single learning mechanism and only symbolic representations. Despite the theoretical advantages of such a constrained theory, current developments in Soar seek to extend the architecture to achieve new functional goals, with more long-term memory systems, sub-symbolic mechanisms, and a module to model the effects of emotion on the cognitive system (Nason & Laird, 2004; Marinier & Laird, 2004).

ACT-R

The ACT-R (Adaptive Control of Thought, Rational) theory (Anderson et al., 2004) rests upon three important components: *rational analysis* (Anderson, 1990), the distinction between *procedural* and *declarative* memory (Anderson, 1976), and a *modular structure* in which components communicate through *buffers*. According to rational analysis, each component of the cognitive architecture is optimized with respect to demands from the environment, given its computational limitations. If we want to know how a particular aspect of the architecture should function, we first have to look at how this aspect can function as optimally as possible in the environment. Anderson (1990) relates this optimality claim to evolution. An example of this principle is the way choice is implemented in ACT-R. Whenever there is a choice between what strategy to use or what memory element to retrieve, ACT-R will take the one that has the highest utility, which is the choice that has the lowest expected cost while having the highest expected probability of succeeding. This is different from Soar's approach, which would involve finding knowledge to decide between strategies.

The principle of rational analysis can also be applied to task knowledge. While evolution shapes the architecture, learning shapes knowledge and possibly part of the knowledge acquisition process. Instead of only being focused on acquiring knowledge per se, learning processes should also aim at finding the right representation. This may imply that learning processes have to attempt several different ways to represent knowledge, so that the optimal one can be selected. For example, in a model of the past tense (Taatgen & Anderson, 2002), the model had to choose between an irregular and a regular solution to inflect a word. It chose the more efficient irregular solution for the high-frequency words, because storing the exception is worth the efficiency gain. For low-frequency words having an efficient exception does not pay off, so the model selected the more economic regular solution.

The second ACT-R foundation is the distinction between *declarative* and *procedural* knowledge. ACT-R has a separate procedural and declarative memory, each of which has their own representation and learning mechanisms. Procedural memory stores productions that can directly act upon the current situation. Each of these productions maintains a *utility* value to keep track of its past success. Declarative memory is more passive: knowledge in it has to be requested explicitly in order to be accessed. Elements in declarative memory have an *activation* value to track their past use that can model, among other things, forgetting. Declarative memory also incorporates the function of working memory, making it unnecessary to have a separate working memory. Because ACT-R uses activation and utility values in addition to purely symbolic representations, it is called a hybrid architecture.

The third, and also most recent foundation of ACT-R is its modular structure. The production system, which forms the core of the architecture, cannot arbitrarily access any information it wants, but has to communicate with other systems through a buffer interface. For example, if the visual module attends new information, it places the encoded information in the visual buffer, after which this information can be accessed by production rules. Although this restricts the power a single production rule, it does allow each module to do its own processing in parallel with other modules.

Both Soar and ACT-R claim to be based on the principles of rationality, although they define rationality differently. In Soar rationality means making optimal use of the available knowledge to attain the goal, while in ACT-R rationality means optimal adaptation to the environment. Not using all the knowledge available is irrational in Soar, although it may be rational in ACT-R if the costs of using all knowledge are too high. On the other hand ACT-R takes into account the fact that its knowledge may be inaccurate, so additional exploration is rational. Soar will explore only when there is a lack of knowledge, but has, contrary to ACT-R, some built-in strategies to do so.

EPIC

Although most cognitive architectures start from the perspective of central cognition, the EPIC (Executive-Process Interactive Control) architecture (Meyer & Kieras, 1997) stresses the importance of peripheral cognition as a factor that determines task performance. In addition to a cognitive processor with its associated memory systems EPIC provides a set of detailed perceptual and motor processors. The perceptual modules are capable of processing stimuli from simulated sensory organs, sending their outputs to working memory. They operate asynchronously, and the time required to process an input depends on the modality, intensity and discriminability of the stimulus. The time requirements of the perceptual modules, as well as other modules, are based on fixed equations like Fitts' law, and serve as a main source of constraints.

EPIC's cognitive processor is a parallel matcher: in each cycle, which takes 50 ms, production rules are matched to the contents of working memory. Each rule that matches is allowed to fire, so there is no conflict resolution. It is up to the modeler to prevent this parallel firing scheme from doing the wrong thing. Whereas both Soar and ACT-R have a production firing system that involves both parallel and serial aspects, EPIC has a pure parallel system of central cognition. As a consequence, EPIC predicts that serial aspects of behavior are mainly due to communication between central and peripheral processors and structural limitations of sense organs and muscles. An important aspect of EPIC's modular structure is the fact that all processors can work in parallel. Once the cognitive processor has issued a command to the ocular motor processor to direct attention to a spot, it does not have to wait until the visual processor has processed a new image. Instead, it can do something else. In a dual-task setting the cognitive processor may use this extra time to do processing on the secondary task. EPIC can represent multiple goals in a non-hierarchical fashion, and these goals can be worked on in parallel, provided they do not need the same peripheral resources. If they do, as is the case in experiments where participants have to perform multiple tasks simultaneously, executive processes are needed to coordinate which of the goals belonging to the tasks may access which peripheral processors. Because EPIC's executive processes are implemented by production rules, they do not form a separate part of the system. This makes EPIC very flexible, but it also means that EPIC's theory of central cognition is rather weak, in the sense of having few constraints as a theory, as opposed to a very strong theory of peripheral cognition. EPIC is mainly focused on expert behavior and presently has no theory of how knowledge is learned. As we will see all the other architectures have picked up EPIC's peripheral modules, but try to have a constrained central cognitive system as well.

Clarion

The Clarion architecture (Connectionist Learning with Adaptive Rule Induction Online; Sun, 2003; Sun, Slusarz, & Terry, 2005; Sun, Merrill & Peterson, 2001) has as its main architectural assumption that there is a structural division between explicit cognition and implicit cognition. As a consequence the architecture has two subsystems, the top and the bottom layer, that each have their own representations and processes. Furthermore each of the two layers is subdivided into two systems: and action-centered system and a non-action-centered system. This latter distinction roughly corresponds to procedural and declarative, respectively: the action-centered system can directly influence action, while the non-action-centered system can only do so indirectly. Learning can be bottom-up, in which case knowledge is first acquired implicitly, and serves as a basis for later explicit learning, or top-down, in which case knowledge is acquired explicitly, and implicit learning follows later. A final central assumption of Clarion is that when there is no explicit knowledge available a priori, learning will be bottom-up. Many, but not all, of Clarion's representations use neural networks. In that sense it is more a true hybrid architecture than ACT-R in having truly connectionist and symbolist characteristics.

The central theory of Clarion is that behavior is a product of interacting implicit (bottom-up) and explicit (top-down) processes, further modulated by a motivational subsystem (which holds, among others, the system's goals) and a meta-cognitive subsystem. The explicit action-oriented system has a rule system in which rules map the perceived state onto actions. The implicit action-oriented system assigns quality measures to state/action pairs. The final choice of an action is a combination of the values assign to each action by the top-down and the bottom-up system. Each of the two systems has its own learning mechanisms: the implicit system uses a combination of reinforcement learning and backpropagation to improve its assessment of state/action pairs based on rewards, while the explicit system uses a rule-extraction mechanism that uses extraction, generalization and specialization to generate new rules. Apart from these two subsystems, each of the other subsystems of Clarion uses their own mechanisms and representations.

3. Constraints on Modeling

As pointed out earlier, in each architecture there is a tension between functional and theory goals. From the functional perspective there is a pressure to add features, mechanisms and systems to the architecture in order to capture more phenomena. From the theory perspective there is a pressure to simplify representations and mechanisms, and to remove features that are not strictly necessary from the architecture. The goal of this pressure on simplicity is to keep the possible space of models for a particular phenomenon as small as possible. If an architecture allows many different models of the same phenomenon, there is no a priori method to select the right one. In section we will review how architectures can help constrain the space of possible models. We will examine a number of topics that can serve as constraints on modeling, and discuss how four architectures offer solutions to help modeling in that topic area. A summary can be found in Table 1. Note that not all architectures address all topic areas, so for example EPIC does not constrain learning because it presently has no theory of learning.

Working Memory Capacity

One of the findings that established cognitive psychology as a field was Miller's experiment in which he found that people can only retain a limited number of unrelated new items in memory (Miller, 1956). This phenomenon quickly became associated with short-term memory and later working memory. More generally, the function of working memory is to maintain a representation of the current task environment. What Miller's and subsequent experiments showed was that the capacity to maintain this representation is limited.

A naive model of working memory is to have a system with a limited number of slots (for example the seven suggested by Miller) that can be used to temporarily store items. Once you run out of slots items have to be tossed out. Although such a model is an almost direct implementation of the phenomenon on which it is based, it does not work very well as a component in an architecture. Miller's task is about completely unrelated items, but as soon as knowledge *is* related, which is the case in almost any natural situation, the slot-model no longer holds.

A good theory of working-memory capacity can be a powerful source of constraint in a cognitive architecture, because it rules out models that can interrelate unrealistically large sets of active knowledge. Although working memory is traditionally viewed from the perspective of *capacity*, a resource that can run out, another perspective is to consider working memory as a *cognitive function*. In the functional approach, limited working memory is not a hindrance, but the ability to separate relevant from irrelevant information. It allows the rest of the cognitive system to act upon information that is relevant instead of irrelevant.

Capacity limitations in Soar.

An example of a functional approach of working memory is Soar (Young & Lewis, 1999). Young and Lewis explain working memory limitations in terms of what the current set of skills can do in limited time. For example, consider the following three sentences:

1. The defendant examined the courtroom.
2. The defendant examined by the jury was upset.
3. The evidence examined by the jury was suspicious.

Assuming people read these sentences one word at a time from left to right, the word *examined* is ambiguous in sentences 1 and 2, because they can either be the main verb or the starting verb of a relative clause, but not in sentence 3 because the word *evidence* is inanimate. Just and Carpenter (1992) found that people differ in how they handle sentence 3, and attribute this to working memory capacity: high capacity individuals are able to keep the information that evidence is inanimate in working memory, disambiguating the sentences, while low-capacity individuals do not hold that information in memory, forcing them to disambiguate the sentence later like in sentence 2. Lewis (1996), however, presented a different account of the individual differences based on a Soar model of natural language comprehension. In sentence 3 after reading *examined*, Soar will propose two operators to update the current comprehension of the sentence, one corresponding to each interpretation of the sentence. This will create an impasse, which Soar will try to resolve this impasse in a new problem space. Although the Soar model has the knowledge to solve this problem, this takes time, and given the time pressure the model can revert to selecting the normally preferred disambiguation of interpreting a verb as the main verb, which means it will run into trouble later in the sentence.

In this model the individual differences are not explained by a limit in capacity of working memory as such, because the fact that *evidence* is animate is perfectly available in working memory, but a limitation of the available knowledge to actually do something with that fact in the given problem context.

Capacity limitations in ACT-R.

Similarly to Soar, ACT-R has no system that directly corresponds to the notion of working memory capacity. Indeed, ACT-R does not even have a working memory as such. Instead the function of working memory is tied to several of ACT-R's systems. ACT-R's current task context is maintained in the set of buffers. A buffer is a means for the central production system to correspond to the various modules in the system. For example, there is a visual buffer to hold the representation of the currently attended item in the visual field, there is a retrieval buffer to hold the last item retrieved from declarative memory, and there is a goal item that holds the current goal context. Each of these buffers has a capacity of a single item and is constrained by their function (i.e., vision, manual, retrieval, etc.).

Although the buffers together are the main means of holding the current context, the system that is mainly associated with the notion of working memory capacity is declarative memory. Any new item that enters the system is eventually stored in declarative memory. If the task is to memorize a string of numbers, each of the numbers is stored in memory as separate item that is linked to the other numbers (Anderson & Matessa, 1997). In order to recall the string of numbers each of the items must be retrieved successfully. However, as the string of numbers becomes longer, interference and decay in declarative memory decrease the probability that recall is successful, producing the phenomenon of a limited working memory capacity.

Although ACT-R's explanation seems to be closer to a capacity explanation, in the root of the theory the explanation is functional. The purpose of activation in declarative memory is not to model forgetting, but to rank knowledge in order of potential relevance. Knowledge receives a high activation due to frequent past use or a high correlation with the current context because that makes it more available and distinguishable from irrelevant knowledge. From that perspective working memory capacity is the ability to increase the signal-to-noise ratio in declarative memory, and individuals who are good at increasing this ratio have a high working memory capacity (Lovett, Reder & Lebiere, 1999).

Cognitive Performance

How powerful is the human reasoning system? According to, for example, Penrose (1989), the human reasoning is more powerful than a Turing Machine, making it possible for humans to solve problems that are computationally intractable. The challenge for cognitive architectures is, however, not to make the computing machinery more powerful, but to put constraints on the power that is already there, without constraining it so much that it cannot do certain tasks that humans can.

The serial bottleneck.

A recurrent topic of debate in the psychology of human perception and performance is whether there is a central bottleneck in human cognition (Pashler, 1994; Schumacher et al., 2001). In terms of cognitive architectures the debate is centered between ACT-R and EPIC. In ACT-R the central production system can only fire one rule at a time. Although each rule firing only takes 50 ms, it limits the number of cognitive steps that can be taken. In EPIC the central rule system can fire any number of rules in parallel. EPIC can therefore naturally explain dual-tasking experiments in which participants achieve perfect time-sharing. An example of such an experiment is by Schumacher et al. (2001). In that experiment participants were given a visual stimulus and a tone at the same time. They had to respond to the visual stimulus by pressing a key, and to the tone by saying a word. Given sufficient training, participants were eventually able to do the two tasks perfectly in parallel, meaning that their reaction times on each task were the same in the dual-task and in the single-task situation.

For ACT-R dual-tasking experiments are a challenge. Nevertheless Byrne and Anderson (2001) constructed a model that was able to perfectly share time between the models, and Taatgen, Anderson and Byrne made models that can learn the perfect time-sharing that captured not only the eventual performance but also the learning trajectory towards this final performance (Taatgen, 2005; Anderson, Taatgen & Byrne, 2005). In the ACT-R models the key to perfect dual tasking is the fact that most of time consumed in these tasks is needed for either perception or motor actions, especially when the task is highly trained. The occasional central action is needed to shift attention or to select a response. In the highly trained cases each of these actions only take a single production rule of 50 ms. Unless the response selection for both tasks has to happen at exactly the same moment (which is unlikely given noise in the perceptual processes), the costs of dual-tasking are very low or absent.

An interesting aspect of the central bottleneck is the way the discussion plays out. With a serial bottleneck ACT-R has the more constrained theory, because it is always possible to do things serially in EPIC, but one cannot do them in parallel in ACT-R. ACT-R principally has the ability to predict circumstances in which the serial bottleneck constrains performance, while EPIC poses no constraints at all. In order for EPIC to prove its point, it needs to identify a phenomenon or task where ACT-R's serial rule system just does not have the time to do everything that needs to be done. Even when such a phenomenon would be found, it would only prove that ACT-R is wrong, and not necessarily that EPIC is right. This example shows that a more constrained architecture almost automatically gains the scientific upper ground, despite (or, as Popper, 1962, would say, because of) the fact that it makes itself vulnerable to refutation.

Hidden computational power.

The simplicity of production rules can be deceptive. If production rules can match arbitrary patterns, it is possible to write production rules in which matching a condition is an NP-complete problem (Tambe, Newell & Rosenbloom, 1990). Production rules in Soar have that nature, and this is why Soar needs a powerful rule-matching algorithm (Rete, see Forgy, 1982). Although powerful rules offer a great deal of flexibility, it under-constrains what can be done in a single production-matching cycle. To counter this, Soar modelers try to refrain writing rules that use the full Rete power. In Clarion (Sun, 2003), on the other hand, the rule system (the explicit, action-oriented system) is implemented in a neural network. Given the localist nature of neural networks there is no hidden computational power, producing a more constrained system.

ACT-R also has a constrained production system: it can only match items in its buffers. This implies that it cannot match arbitrary patterns in declarative memory, but can only retrieve items one at a time. A complex match of information might therefore take up multiple retrieval steps, and is in no way a fail-safe process. ACT-R's performance system may eventually prove to be too restrictive, preventing it from fulfilling all functional goals.

Perceptual and Motor Systems

Perceptual and motor systems are potentially a strong source of constraint, because the perceptual and motor actions can be registered more precisely in experiments than cognitive actions, and because the psychophysical literature offers precise predictions about the timing of these actions. The EPIC architecture (Meyer & Kieras, 1997) is based on this premise.

The perceptual-motor modules in EPIC can handle only a single action at a time, and each of these actions take a certain amount of time. Although a module can do only one thing at a time, expert behavior on a task is exemplified by skillful interleaving of perceptual, cognitive, and motor actions. EPIC's modules incorporate mathematical models of the time it takes to complete operations that are based on empirical data. The knowledge of the model is represented using production rules.

An example of how perceptual and motor constraints can inform a model is menu search (Hornof and Kieras, 1997). The task was to find a label in a pull-down menu as quickly as possible. Perhaps the simplest model of such a task is the serial-search model in which the user first attends to the top item on the list and compares it to the label being searched for. If the item does not match the target, the next item on the list is checked; otherwise, the search is terminated. EPIC's predictions for search time using this method can be obtained by implementing the strategy in EPIC production rules and performing a simulation in a test environment in which menus have to be searched can obtain. It turns out that a naive serial-search model grossly overestimates actual search time (obtained with human subjects), except when the target is in the first position to be searched. For example, if the menu item is in position 10, the serial search model predicts that finding the item takes 4 seconds while participants only need in the order of 1.6 seconds.

Hornof and Kieras propose an alternative model, the overlapping search model that exploits the parallelism of the cognitive system. Instead of waiting for the cognitive system to finish deciding whether or not the requested label is found, the eye moves on to the next item in the list while the first item is still being evaluated. Such a strategy results in the situation that the eye has to move back to a previous item in the list once it has been decided that the item has been found, but this is a small price to pay for the speed-up this parallelism produces. Parallelism is allowed in EPIC as long as perceptual-motor modules do one thing at a time. In practice, the most influential constraint is posed by the duration of actions. For example, in the serial-search model, the parameter that influences the search time could, in theory, be changed to make this (incorrect) model match the data. EPIC precludes this from occurring because an eye-movement takes a certain amount of time, as does a decision as to whether the label is correct or not, such that the data can only be explained if these actions occur in parallel.

The menu-search example shows that while the perceptual and motor systems in EPIC provide strong constraints, central cognition is under-constrained in the sense that it allows both correct and incorrect models of menu search. EPIC's perceptual and motor modules, however, have proved to be so powerful as constraints that all the other architectures (ACT-R, Soar and Clarion) have copied them.

Learning

As we mentioned in the introduction, a desirable feature of a model is that it learns its own knowledge. In the classical modeling paradigm the only constraints placed on the knowledge come from the architecture and task analysis and this usually leaves the knowledge specification partly up to the whim of the modeler. A more constrained approach is to have a model that learns its own knowledge. A weak variant of this type of modeling is to supply the model with some initial method that becomes faster through learning. For example, in an ACT-R model of alphabet-arithmetic (Anderson & Lebiere, 1998) the task is to verify additions using letters and numbers, like G+3=J. The model's initial strategy is to find the answer through a counting process. But because it stores its answers, the model gradually accumulates addition facts in declarative memory, allowing it to give a direct answer instead of having to count. Although the model learns the new addition facts, it already starts out with the production rules that can count and with the productions rules that attempt retrieval of initially non-existing addition facts.

Learning from direct instruction.

A more ambitious approach is to have a model that learns all of its own task-specific knowledge through either instruction or feedback. An example in Soar of such a system is Instructo-Soar (Huffman & Laird, 1995). Instructo-Soar can learn to carry out commands in natural language. If it doesn't know how to do a certain command, it will ask for an instruction. A sample dialog from Huffman and Laird is as follows (Soar's questions are in italics):

Push the green button.

That's a new one. How do I do that?

Move to the grey table.

Ok. What next?

Move above the green button.

> *How do I do that?*
>
> Move the arm up.
>
> *Oh, I see! What next?*

Move down.

Ok, What next?

The operator is finished.

In this example Soar receives instructions how to push a green button. The indentation represents the structure of the problem solving, with each level of indentation an impasse that has to be resolved. Soar's learning mechanism will learn new rules to resolve similar cases in the future. For example, after this exchange Soar will know how to move above things, and how to push buttons. One of the challenges is to make the right generalization: instead of learning how to push buttons, another generalization might have been a procedure to push green things. To make the right generalization Soar used background knowledge to reason out that green is not a relevant attribute for pushing things. An alternative to knowledge-based generalization is Clarion's bottom-up generalization, in which associations between state, action and success are first gathered by the implicit learning process. These bottom-up associations then gradually inform the rule-extraction mechanism to make the right generalization. So instead of making inferences about colors and buttons Clarion would rather induce out of experiences that colors don't matter but buttons do.

Interpreting instructions stored in memory.

Instead of direct instruction, a model can also be taught what to do by memorizing an initial set of instructions. Several ACT-R models are based on this paradigm (Taatgen & Lee, 2003; Anderson et al., 2004; Taatgen, 2005). The idea is that the system first reads instructions that it then stores in declarative memory. When the task is performed these instructions are retrieved from memory and carried out by production rules. These production rules are not specific for the task, but rather represent general skills like pushing buttons, finding things on the screen, comparing items, etc. The declarative instructions string the general skills together to produce task-specific behavior. The cycle of retrieving and interpreting instructions from memory can explain many aspects of novice behavior. Performance is slow because the process of retrieving an instruction from memory is a time-consuming process during which the system cannot do much else. It is serial, because only one instruction is active at the same time, making it impossible to do two steps in parallel. It is prone to errors, because instructions may have been forgotten, requiring the model to reconstruct them through a time-consuming problem-solving process. It also puts heavy demands on working memory capacity: both instructions and temporary information has to be stored and retrieved from declarative memory, making it the main bottleneck of novice processing. Because declarative memory is the bottleneck it is almost impossible to do other tasks in parallel that also make demands on declarative memory.

Novice behavior is gradually transformed into expert behavior through a knowledge compilation process (*production compilation*, Taatgen & Anderson, 2002). Production compilation combines two existing rules into one new rule, while substituting any memory retrieval in between those rules into the new rule. If the memory retrieval in between the two rules is an instruction, this instruction is effectively encoded into the newly learned rule, creating a production rule that is specific to the task. Production learning in ACT-R therefore gradually transforms task-specific declarative knowledge and general production rules into task-specific production rules. These newly learned rules exhibit many characteristics of expert behavior. They are longer tied to a linear sequence of instructions, so they can be used out of sequence whenever they apply, allowing parallel performance and increased flexibility of carrying out a task (Taatgen, 2005).

Although models that learn from instructions cannot yet directly parse natural language, they do offer more constrained models than models that are given expert knowledge right away. Not all the expert models that can be encoded using production rules are learnable, and those that are not can therefore be ruled out. In addition to that, the fact that the model learns its knowledge offers the opportunity to match predictions about the learning trajectory to human data. This means that some expert models that are learnable in the sense that the knowledge could be produced by the mechanisms in the architecture can still be ruled out because their learning trajectory doesn't match the human data.

From implicit to explicit learning.

One other way for a model to obtain its knowledge is by discovering regularities in the environment. Although many classical models of discovery focus on explicit discovery processes, many modern models start from the assumption that knowledge is often learned implicitly. In, for example, the sugar factory experiment by Berry and Broadbent (1984), participants have to decide how many workers they should send into the factory each day to achieve some target output. The output depends not only on the number of workers, but also on the production of the previous day. Although participants in the experiment generally do not explicitly discover the relationship, they do get better at adjusting the number of workers in the course of the experiment. This and similar experiments suggest that there is some unconscious component to learning, implicit learning, that improves our performance without awareness. Several models have been proposed to capture this effect. An ACT-R model by Wallach (Taatgen & Wallach, 2002) stores examples of input/output relations in declarative memory, and retrieves the example that has the highest activation and similarity to the current situation. This model never gains explicit knowledge of the relationships in the task, but achieves better performance by learning a representative set of examples.

Sun, Slusarz and Terry (2005) have modeled an extension to the original experiment, in which in some conditions participants were explicitly taught particular input-output pairs, or were given simple heuristic rules. In the control, no explicit training, version of the model, the implicit layer of Clarion was solely responsible for pickup up the regularities in the task. In the instructed version of the model, the explicitly given instructions were represented in Clarion's explicit memory, driving the implicit learning processes together with experience. The explicit instructions provided a performance boost in the data, which was successfully captured by the model.

Constraints from Neuroscience

Human cognition is implemented in the brain. This fact can offer additional sources of constraint in a cognitive architecture. The architecture of the brain offers two levels of constraints: at the level of individual neurons and their interconnections, and at the level of global brain structures.

Constraints at the level of individual brain cells.

The actual substrate of cognition is an interconnected network of neurons. Whether or not this is a significant source of constraint is open to debate. One view is that brain cells implement some virtual architecture, and that the characteristics of brain cells are irrelevant for an understanding of cognition (e.g., Newell, 1990). A more moderate version of this point of view is adapted by the ACT-R architecture (Anderson & Lebiere, 2003). In that view the main level of abstraction to study cognition is higher than the level of brain cells. However, Lebiere has implemented a neural network version of ACT-R that is functionally identical to the regular implementation. Nevertheless this model proved to supply some additional restrictions on the architectures, for example the restriction that declarative memory can only retrieve one item at a time.

Clarion (Sun, 2003) takes the point of view that elements and mechanisms that resemble neurons are an important source of constraint on the architecture. Many of Clarion's subsystems are composed from neural networks. This offers additional constraints, because neural networks are less easy to "program" than symbolic models. Clarion is still a hybrid architecture with both symbolic and subsymbolic aspects. One can go even further and design an architecture completely build out of neural network components. Leabra (O'Reilly & Munakata, 2000) is an example of such an architecture. The challenge with an architecture built out of simulated neurons is that it has to deal with a number computational problems that are trivial to partly symbolic architectures. A first problem is the *binding problem,* the problem how the cognitive systems groups features together, like associating a filler with a role, or a value with a variable or attribute (Fodor & Pylyshyn, 1988). Given the large number of possible combinations of attributes and values, it is not feasible that each combination already has a pre-wired connection that can be activated, so some other means must be found to temporarily connect concepts together. A second problem is called *catastrophic interference* (McCloskey & Cohen, 1989), which refers to the fact that many networks unlearn previously learned knowledge when new knowledge is presented. A third problem is *serial behavior*. Although networks perform particularly well with respect to massively parallel performance, it is much harder to let them do steps that have to be performed in a certain serial order. The three problems might seem to be an argument against neural network architectures, but they may turn out to be the opposite, because overcoming these problems will offer new constraints on the architecture. In fact, for each of the three problems solutions have been proposed. One of the solutions of the binding problem is Smolensky's (1990) tensor product system. There are several solutions to the catastrophic interference problem, for example McClelland, McNaughton and O'Reilly's (1995) solution to have a slow (cortical) and fast (hippocampal) learning system, now implemented in the Leabra (O'Reilly & Munakata, 2000) system. Serial behavior can be achieved partially by recurrent networks (e.g., Elman, 1990), although not with the full flexibility that symbolic systems offer. A neural network cognitive architecture that would offer solutions to each of these problems could make a very strong theory.

Constraints at the global brain architecture level.

Recent advances in brain imaging have allowed neuroscientists to build increasingly finer-grained theories of what the functions of various regions in the brain are, and how these regions are interconnected. The result is a map of interconnected, functionally labeled regions. What brain imaging does not provide is the actual processing in these regions. Cognitive architectures can provide processing theories constrained by the processing map of the brain. ACT-R (Anderson et al, 2004; Anderson, 2005) has mapped its buffers and production system onto brain regions, and is capable of making predictions of brain activity on the basis of a cognitive model. For example, in a study in which children had to learn to solve algebra equations, the ACT-R model predicted how activity in several brain areas would differ with problem difficulty and the effects of learning (Anderson, 2005).

4. Conclusions

The viewpoint of cognitive constraint is different from the perspective of how much functionality an architecture can provide, as expressed by, for example, Anderson and Lebiere (2003). Anderson and Lebiere have elaborated Newell's (1990) list of constraints that are mainly (but not all) functional goals (e.g., *use natural language*). Although both functionality and strength as a theory are important for a cognitive architecture, modelers tend to focus on functionality, and the critics tend to focus on theory strength. One symptom of the fact that cognitive architectures are still relatively weak theories is that few predictions are made, as opposed to fitting a model onto data after the experiment has been done (but see Salvucci & Macuga, 2002 and Taatgen, van Rijn & Anderson, submitted, for examples of successful predictive research). A research culture in which modelers would routinely model their experiment *before* they would conduct the experiment would create a much better research environment, in which confirmed predictions would be evidence for theory strength, and in which failed predictions would be great opportunities to strengthen the theory. For this research strategy to work, it is necessary that architectures limit the number of possible models for a particular phenomenon. Alternatively, attempts could be made to rank the possible space of models with the goal of identifying the most plausible one based on non-architectural criteria. Chater and Vitányi (2003) argue, following a long tradition in science in general, that the most simple explanation should be preferred. More specific in the architecture context, Taatgen (in press), argues that if there is a choice between multiple models, the model should be preferred with the most simple control structure.

This is also the great promise for the field: as architectures become stronger theories, they can go beyond modeling small experimental tasks, and provide a synergy that can lead to the more ambitious functional goals to make cognitive architectures truly intelligent systems.

References

Anderson, J. R. (1976). *Language, Memory and Thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R. (1990). *The adaptive character of thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R. (2005). Human symbol manipulation within an integrated cognitive architecture. *Cognitive Science, 29*(3), 313-341.

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review*, *111*(4), 1036-1060.

Anderson, J. R. & Lebiere, C. L. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R. & Lebiere, C. L. (2003). The Newell test for a theory of cognition. *Behavioral & Brain Sciences, 26*, 587-637.

Anderson, J. R. & Matessa, M. P. (1997). A production system theory of serial memory. *Psychological Review*, *104*, 728-748.

Anderson, J.R., Taatgen, N.A. & Byrne, M.D. (2005). Learning to Achieve Perfect Time Sharing: Architectural Implications of Hazeltine, Teague, & Ivry (2002). *Journal of Experimental Psychology: Human Perception and Performance, 31*(4), 749-761.

Berry, D. C., & Broadbent, D.E. (1984). On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology, 36A*, 209-231.

Byrne, M. D., & Anderson, J. R. (2001). Serial modules in parallel: The psychological refractory period and perfect time-sharing. *Psychological Review, 108,* 847-869.

Chater, N. & Vitányi, P. (2003). Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences, 7*(1), 19-22.

Chong, R. S. (1999). Modeling dual-task performance improvement: Casting executive process knowledge acquisition as strategy refinement. Unpublished dissertation. University of Michigan.

Cooper, R. & Fox, J. (1998). COGENT: A visual design environment for cognitive modelling. *Behavior Research Methods, Instruments, & Computers*, *30*, 553-564.

Elman, J.L. (1990). Finding structure in time. *Cognitive Science, 14,* 179-211.

Fodor, J. A. & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: a critical analysis. *Cognition, 28,* 3-71.

Forgy, C. L. (1982). Rete: a fast algorithm for the many object pattern match problem. *Artificial Intelligence, 19,* 17-37.

Hornof, A.J., & Kieras, D.E. (1997). Cognitive modeling reveals menu search is both random and systematic. *Proceedings of CHI-97* (pp. 107–114). New York: Association for Computing Machinery.

Huffman, S.B. & Laird, J.E. (1995). Flexibly instructable agents. *Journal of Artificial Intelligence Research, 3,* 271-324.

Just, M. A. & Carpenter, P. A. (1992). A Capacity Theory of Comprehension: Individual Differences in Working Memory. *Psychological Review, 99,* 122-149.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Lewis, R. L. (1996) Interference in short-term memory: The magical number two (or three) in sentence processing. *Journal of Psycholinguistic Research, 25,* 93-115.

Lovett, M. C., Reder, L. M., & Lebiere, C. (1999). Modeling working memory in a unified architecture: An ACT-R perspective. In A. Miyake & P. Shah (Eds.) *Models of Working Memory* (pp. 135-182). Cambridge, MA: Cambridge.

Marinier, R. P. & Laird, J. E. (2004). Toward a comprehensive computational model of emotions and feelings. *Proceedings of the Sixth International Conference on Cognitive Modeling* (pp. 172-177). Mahwah, NJ: Erlbaum.

McClelland, J.L., McNaughton, B.L., & O'Reilly, R.C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review, 102,* 419-457.

McCloskey, M. & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G.H. Bower (Ed.), *The Psychology of Learning and Motivation, vol. 24* (pp. 109-164). San Diego, CA: Academic Press.

Meyer, D. E. & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance. Part 1. Basic mechanisms *Psychological Review, 104,* 2-65.

Miller, G.A. (1956). The magic number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review, 63*, 81-97.

Nason, S. & Laird, J. E. (2004). Soar-RL: Integrating Reinforcement learning with Soar. *Proceedings of the Sixth International Conference on Cognitive Modeling* (pp. 208-213). Mahwah, NJ: Erlbaum.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard.

Newell, A. & Simon, H. A. (1963). GPS, a program that simulates human thought. In E. A. Feigenbaum & J. Feldman (eds.), *Computers and Thought*. New York: McGraw-Hill.

O'Reilly, R. C. & Munakata, Y. (2000). *Computational explorations in cognitive neuroscience*. Cambridge, MA: MIT Press.

Pashler, H. (1994). Dual-task interference in simple tasks: Data and Theory. *Psychological Bulletin, 116*, 220-244.

Penrose, R. (1989). *The emperor's new mind.* Oxford, UK: Oxford University Press.

Popper, K. R. (1962). *Conjectures and refutations: The growth of scientific knowledge*. New York: Basic Books.

Roberts, S. & Pashler, H. (2000). How persuasive is a good fit? A comment on theory testing. *Psychological Review, 107,* 358-367.

Salvucci, D.D., & Macuga, K. L. (2002). Predicting the effects of cellular-phone dialing on driver performance. *Cognitive Systems Research*, 3, 95-102.

Schumacher, E. H., Seymour, T. L., Glass, J. M., Fencsik, D. E., Lauber, E. J., Kieras, D. E., & Meyer, D. E. (2001). Virtually perfect time sharing in dual-task performance: Uncorking the central cognitive bottleneck. *Psychological Science, 12* (2), 101-108.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence, 46,* 159-216.

Sun, R. (2003). A tutorial on Clarion. Technical report Cognitive Science Department, Rensselaer Polytechnic Institute. http://www.cogsci.rpi.edu/~rsun/sun.tutorial.pdf.

Sun, R., Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science, 25*(2), 203-244.

Sun, R., Slusarz, P., & Terry, C. (2005). The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review, 112*(1), 159-192.

Sun, R. & Zhang, X. (2004). Top-down versus bottom-up learning in cognitive skill acquisition. *Cognitive Systems Research, 5*(1), 63-89.

Taatgen, N.A. (2005). Modeling parallelization and speed improvement in skill acquisition: from dual tasks to complex dynamic skills. *Cognitive Science, 29*, 421-455.

Taatgen, N. A. (in press). The minimal control principle. In W. Gray (Ed.), *Integrated models of cognitive systems*. Oxford: Oxford University Press.

Taatgen, N. A. & 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. & Lee, F.J. (2003). Production Compilation: A simple mechanism to model Complex Skill Acquisition. *Human Factors, 45*(1), 61-76.

Taatgen, N.A., van Rijn, D.H. & Anderson, J.R. (submitted). An integrated theory of prospective time interval estimation: the role of cognition, attention and learning.

Taatgen, N.A. & Wallach, D. (2002). Whether skill acquisition is rule or instance based is determined by the structure of the task. *Cognitive Science Quarterly, 2*(2), 163-204.

Tambe, M., Newell, A. & Rosenbloom, P. S. (1990). The problem of expensive chunks and its solution by restricting expressiveness. *Machine Learning, 5,* 299-348.

Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 2nd series, 42,* 230-265.

Turing A. (1950). Computing machinery and intelligence. *Mind, 59,* 433-460.

Young, R.M. & Lewis, R.L. (1999). The Soar cognitive architecture and human working memory. In A. Miyake & P. Shah (Eds.) *Models of Working Memory* (pp. 224–256). Cambridge, MA: Cambridge.

Table 1. Overview on how architectures constrain aspects of information processing.

Process	Architecture	Constraint	Reference
Working Memory			
	Soar	Limitations of working memory arise on functional grounds, usually due to lack of reasoning procedures to properly process information.	Young & Lewis (1999).
	ACT-R	Limitations of working memory arise from decay and interference in declarative memory. Individual differences are explained by differences in spreading activation.	Lovett, Reder & Lebiere (1999).
	Clarion	Limitations of working memory are enforced by a separate working memory with decay.	Sun & Zhang (2004).
Cognitive performance			
	Soar	A decision cycle in Soar takes 50 ms, although many production rules may fire in parallel leading to the decision.	Newell (1990).
	ACT-R	A production rule takes 50 ms to fire, no	Anderson, et al.

		parallel firing is allowed. A rule is limited to inspecting the current contents of the perceptual and memory-retrieval systems and initiating motor action and memory-retrieval requests.	(2004).
	EPIC	Production rules take 50 ms to fire, but parallel firing of rules is allowed.	Meyer & Kieras (1997).
	Clarion	Performance is produced by an implicit (parallel) and explicit (serial) reasoning system that both have an action-oriented and a non-action-oriented subsystem.	Sun (2003).
Perceptual and Motor systems			
	EPIC	Perceptual and motor modules are based on timing from the Model Human Processor (Card, Moran & Newell, 1983). Modules operate asynchronously alongside central cognition.	Kieras & Meyer (1997).
	ACT-R; Soar; Clarion	Use modules adapted from EPIC.	Byrne & Anderson (2001), Chong (1999), Sun (2003).
Learning			
	Soar	Learning is keyed to so-called impasses,	Newell (1990).

		where a sub-goal is needed to resolve a choice problem in the main goal.	
	ACT-R	Learning is based on rational analysis in which knowledge is added and maintained in memory on the basis of expected use and utility.	Anderson et al. (2004).
	Clarion	Learning is a combination of explicit rule extraction/refinement and implicit reinforcement learning.	Sun, Slusarz & Terry (2005).
Neuroscience			
	ACT-R	Components in ACT-R are mapped onto areas in the brain, producing predictions of fMRI activity.	Anderson (2005).
	Clarion	Uses brain-inspired neural networks as components in the architecture	Sun (2003).