Towards a Formal Foundation of Cognitive Architectures

Marco Ragni (ragni@cs.uni-freiburg.de)[1], Kai Sauerwald (kai.sauerwald@fernuni-hagen.de)[2]
Tanja Bock (Tanja.Bock@tu-dortmund.de)[3], Gabriele Kern-Isberner (gabriele.kern-isberner@cs.uni-dortmund.de)[3]
Paulina Friemann (friemanp@cs.uni-freiburg.de)[1], Christoph Beierle (christoph.beierle@fernuni-hagen.de)[2]
[1]Cognitive Computation Lab, University of Freiburg, Freiburg, Germany
[2]Department of Mathematics and Computer Science, University of Hagen, Hagen, Germany
[3]Department of Computer Science, TU Dortmund University, Dortmund, Germany

Abstract

Cognitive architectures are an advantageous tool for creating cognitive models. They provide a framework integrating general cognitive structures and assumptions about the mind as for example the working memory, structural modularity or their interconnections. A vast number of cognitive architectures have been developed in the last decades. While the architectures realize the cognitive perspective, the formal foundation and similarities of cognitive architectures remain open. To identify the cognitive substrate of the architectures, we propose a generalized cognitive framework allowing to embed different cognitive architectures to analyze their properties and to have a common and formal ground for comparisons. We demonstrate our approach – as proof-of-concept – by embedding the two most popular architectures, ACT-R and SOAR, and evaluate cognitive models for recognition memory in our approach. Potentials and limitations are discussed.

Keywords: Formalization; Knowledge Representation; Cognitive Architecture; Cognitive Modeling; ACT-R; SOAR

1 Introduction

Describing cognitive processes by concepts and terms from computer science allowed for major progress in cognitive psychology (Newell & Simon, 1972). This includes the distinction between the underlying cognitive data structure (e.g., working memory) and operations performed on it (e.g., retrieval of an item). A core assumption for cognitive modeling is that the data structure is stable across problems. It is crystallized in a *cognitive architecture* that build on associations with brain regions (Anderson, 2007). At a general and abstract level a cognitive architecture consists of a data structure, which can contain an arbitrary number of substructures, and a set of operations to manipulate the data structure. Still, most architectures are built on the idea of production rule systems operating on different modality specific modules and buffers (Anderson, 2007; Laird, 2012; Sun, 2001). Architectures have been developed for each of the levels proposed by Marr (1982). Recently, hybrid approaches have been introduced (e.g., ACT-R (Anderson, 2007)) incorporating symbolic and subsymbolic processes building on concepts from neural nets. There are more than 26 cognitive architectures that have been descriptively compared in 2012[1]. In the last 3 years a desire to present a formal description by employing means from software technique for ACT-R have led to some abstractions and formalizations, e.g., using predicate logic (Albrecht & Westphal, 2014) or constraint handling rules (Gall & Frühwirth, 2015). A reconceptualization of ACT-R has been undertaken by Stewart and West (2007) and an analysis for CLARION was made (Sun, 2007). However, formal approaches are still rare and while each cognitive architecture realizes assumptions and findings from psychology and neuroscience – the common grounds, the intersection and differences of the diverse architectures are not yet identified. In this sense each architecture is a hypothesis by itself, but the different hypotheses are not compared. To approach this issue, a general framework independent of a concrete modeling approach is necessary. We will analyze the preconditions of such a framework. At this point, a precise distinction between cognitive models and architecture is required. *Cognitive models* operate on cognitive architectures and use the given "infrastructure" provided by the architecture. A mental representation is localized within the architecture of the human mind and so a cognitive model can include domain-dependent operations on the cognitive architecture.

While the response adequacy of the model can be determined from the given responses of a reasoner for a given task, it is impossible to directly observe the operations and mental representations a human agent applies. Cognitive models predict typical answers, response time, and process steps. However, cognitive models are never just *simulation models* that reproduce existing experimental data. A good cognitive model is largely independent of experimental data and has general predictive qualities. At the same time each cognitive model implemented for a specific architecture cannot be necessarily transferred to another architecture that is built on a different data structure. So it is possible that, if models in different architectures model the same data, they can explain it by different strategies depending on specifics of the respective architecture. Consequently, a general framework is necessary that is able to represent both models and to clarify their different assumptions.

In this paper we propose a formal model general enough to represent different cognitive architectures, identifying and generalizing essential features of cognitive architectures. We provide a notion of equivalence in this framework and present a method about how ACT-R and SOAR (and models in these cognitive architectures) can be embedded in our formal framework. Furthermore, we illustrate these embeddings by two concrete models of the Two-High-Threshold model (2-HTM for short), one originally realized in ACT-R and one in SOAR. While the implementations of the 2-HTM model look very different, it is possible to compare them in our framework and we obtain the result that they are equivalent by employing our notion of response equivalence, i.e., these models generate the same response for the same input.

The remainder of the paper is structured as follows. In the next section we briefly introduce two popular cognitive architectures. In Section 3, we will present a general cognitive architecture (GCA), transitions and cognitive models. In Section 4, we demon-

[1]bicasociety.org/cogarch/architectures.php

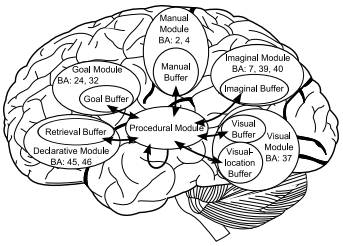


Figure 1: ACT-Rs modules and buffers and their localization

strate some features and embeddings of the cognitive architectures ACT-R and SOAR for the domain of memory. In Section 5 we demonstrate the embeddings on a theory from recognition memory and compare the models. A conclusion discussing the possibilities of a generalized cognitive architecture concludes the article.

2 State-of-the-Art: Cognitive Architectures

Most cognitive architectures are inspired by the General Problem Solver (GPS), a model that uses means-end analysis as a search heuristic (Newell & Simon, 1972). It has been reimplemented as a production rule system, a system that realizes the physical symbol system hypothesis. These systems are composed of (i) *production rules*, consisting of a *condition* part and an *action* part, and an (ii) *interpreter* that checks if conditions of existing production rules are satisfied in a given model's state (they can *fire*). We focus on the two architectures with the most published models: the hybrid cognitive architecture ACT-R and the AI oriented approach SOAR.

The cognitive architecture ACT-R 7.0 (Anderson, 2007) aims at a unified cognition approach. It is a hybrid theory, meaning it consists of symbolic and subsymbolic parts. Its data structure is oriented on modality specific knowledge modules for perception, goal and sub-goal representations (*goal, imaginal*) and interfaces (so-called *buffers*) which can be accessed by production rules. ACT-R uses chunks as the atomic knowledge representation format with procedural knowledge encoded in production rules and declarative knowledge that uses the concept of activation. Cognitive models have been developed for learning and memory, problem solving, deductive reasoning, perception, attention control, and human-computer interaction (HCI), many of which can be found on the website[2]. Since recently, ACT-R allows even for the prediction of task-specific brain activations, modeling findings from fMRI research (see, Fig. 1).

SOAR[3] (States, Operators, And Reasoning) is a production rule system with reinforcement learning built upon the GPS (Newell, 1990). SOAR separates conceptually between short term (called *working memory*) and long term memory (consisting of the three components *production*, *semantic* and the *episodic memory*). Working memory contains objects which are similar to chunks in ACT-R: they consist of a list of key-value-priorities pairs called *Working Memory Elements (WMEs)*, with constants or links to other WMEs as values and a list of comparisons attributes for the priorities. In contrast to ACT-R, an object in SOAR may contain more than one WME with the same key, where the priorities are used to break a tie between these WMEs. A central aspect of the SOAR architecture is a special object in the working memory representing the current state. All other relevant objects in the working memory are indirectly connected (via values in WMEs) to the current state object. Production memory contains production rules which are similar to the productions in ACT-R. In contrast to ACT-R, all matching productions in SOAR fire at the same time. Declarative memory is represented as a net of objects in the semantic memory. The episodic memory saves periodically (partial) snapshots of the actions in working memory. The semantic and episodic memory can be accessed by a retrieval mechanism which adds the result to the working memory. A unique mechanism of SOAR is the recursive creation of sub states to solve an unambiguously operator tie. The current version SOAR 9 integrates non-symbolic representations and other learning mechanisms (Laird, 2012). It performs a variety of problems from planning, robotic systems, interactions with virtual humans, and an air combat simulation for pilot training at the USAF (Tambe et al., 1995).

[2] http://act-r.psy.cmu.edu

[3] http://soar.eecs.umich.edu

3 A General Cognitive Architecture (GCA)

To define the frame of a generalized cognitive architecture (GCA), we need to capture the essential structural properties of a cognitive architecture with respect to memory, constraints on the processes, and interaction with the environment. As introduced above most cognitive architectures, such as ACT-R or SOAR, have an atomic information unit, often called a chunk, which we will use in the following as well. Chunks are lists of key-value pairs, where the keys and values are based on the set of symbols Σ, e.g., constants or references to other chunks. Cognitive architectures may allow the informations that are stored in a chunk to change over time. Hence, a chunk is not directly identified with its knowledge. Most architectures have modality specific modules that can contain these chunks. Some modules can interact with others in a directed way, shifting information from one into the other, while others cannot. Hence, we introduce a relation F over the modules. The implementation of subsymbolic processes in hybrid architectures leads to two kinds of extensions: (i) the possibility of having a probabilistic behaviour selecting the next action of the system, modeled here as nondeterminism; and (ii) the addition of a probabilistic knowledge access mechanism. The access to a chunk depends on the point in time, events in the record of the chunk, or on the current state. For the representation of time we differentiate between $\mathbb{T}$, the set of absolute time points, and $\mathbb{D}$, the set of durations. This definition of time points is relevant for the internal knowledge management in the module. Finally, we introduce compartments, i.e., data structures that can represent the functional analogon of modules and buffers in architectures. A compartment $m \in \mathcal{M}$ may contain chunks and furthermore, some architectures associate for some specific compartments a subsymbolic process controlling the retrieval of chunks. An example is the activation function in ACT-R, or modules that are implemented by a neural net. To allow such subsymbolic processes we assign to each compartment m a function g_m depending on a time parameter and

operations O_m that can be performed on that compartment, e.g., store information etc. To model the effect of subsymbolic mechanisms, each compartment m provides a set of *operations* O_m, which are used to mark that a specific event occurred. We define a *record* as set of elements from $\mathbb{T} \times \mathcal{P}(O_m)$, where $\mathcal{P}(X)$ denotes the power set of a set X. The function g_m guards if a retrieval of an information is successful for the given record, and is called the *subsymbolic guard* function. Hence, we represent a compartment m by a tuple (g_m, O_m). In case of declarative memory in ACT-R, g_m can be for example the base-level activation function, and the operations are *chunk retrieval* and *chunk storage*. The definition of compartments is general enough to capture the notion of buffers, too. Altogether we obtain the following definition of a frame:

Definition 1 (Frame of a General Cognitive Architecture) *The frame of a* ***general cognitive architecture*** $\mathcal{A}$ *is a tuple* $\langle \Sigma, \Gamma, \mathcal{C}, \mathcal{M}, E, \mathcal{F} \rangle$ *with*

- Σ*, a set of* ***internal*** *and* $\Gamma \subseteq \Sigma$*, a set of* ***input symbols***
- $\mathcal{C}$*, a set of* ***chunks****,*
- $\mathcal{M}$*, a finite set of* ***modules****, each a tuple* (g_m, O_m)*, where* $g_m : \mathcal{C} \times O_m \times \mathcal{P}(\mathbb{T} \times O_m) \to \{0,1\}$ *is called a* ***guard function****, and* O_m *is a set of* ***operations****.*
- $E \subseteq \mathcal{M}$*, set of modules interacting with the* ***environment****,*
- $\mathcal{F} \subseteq \mathcal{M} \times \mathcal{P}(O) \times (\mathcal{M} \setminus \{m_E\})$*, an* ***information flow*** *relation, where* $O = \bigcup_{m \in \mathcal{M}} O_m$ *is the set of of all operations.*

States in a cognitive architecture consists of chunks, an assignment of the chunks to their content and a location of the chunks.

Definition 2 (State of a Cognitive Architecture) *A state S of a given cognitive architecture* $\mathcal{A}$ *is a triple* (C, I, ℓ)*, with*

- $C \subseteq \mathcal{C}$*, a finite subset of the chunks of* $\mathcal{A}$*,*
- $I : C \to \mathcal{P}^{\text{bag}}(\Sigma \times (C \uplus \Sigma))$*, an* ***interpretation****-function, that assigns to every chunk in S a multiset of key-value pairs, where the keys are symbols and each value is either a symbol or chunk,*
- $\ell : \mathcal{M} \to \mathcal{P}(C)$*, a* ***localization****-function that assigns to every module a finite set of chunks.*

Each operation performed in a general cognitive architecture is represented by a transition between two states, where a transition consists of essential components: the source and target of the transition, the duration of the operation, and which inputs are present. Additionally a transition specifies which operations are performed on chunks, represented by a tuple (m_1, c, o, m_2). The set of all such operation descriptions is defined as $\mathbb{O} = \mathcal{M} \times \mathcal{C} \times O \times \mathcal{M}$.

Definition 3 (Transition) *A* ***transition*** *of a cogntive architecture* $\mathcal{A}$ *is a tuple* $(s_1, u, \Omega, d, s_2) \in (\mathcal{S} \times \mathcal{P}(\Gamma) \times \mathcal{P}(\mathbb{O}) \times \mathbb{D} \times \mathcal{S})$*, graphically represented by* $s_1 \xrightarrow{u, \Omega : d} s_2$*.*

A set of transitions will be denoted with Δ. Not every transition is consistent with the information flow restrictions or the guard functions of the modules formulated in Definition 1. We call transitions that are not in conflict with the information flow and guard functions *admissible*. A sequence over a set of admissible transitions Δ is called a *trace over* Δ. Furthermore, we call a trace admissible, if all specified operations are successful, i.e. the guard function g_m evaluates to 1 in every transition of a trace.

We write $\mathbb{R}_{\mathcal{A}}^{\Delta}$ for the set of all admissible traces of $\mathcal{A}$ consisting only of traces from Δ and $\mathbb{R}_{\mathcal{A}}$ for the set of all traces of $\mathcal{A}$ over the set of all admissible transitions of $\mathcal{A}$. A *cognitive computational model* in a given cognitive architecture specifies algorithms based on (a subset of) operations allowed in the cognitive architecture, to compute an input-output mapping for the task to be modeled. While the architecture specifies the data structure, the general information flow and processes, a model specifies concrete operations and the given background knowledge.

Definition 4 (Cognitive Model) *A* ***cognitive model*** L *of a cognitive architecture* $\mathcal{A}$ *is a tuple* $(S, s_0, \Delta, F, \alpha)$ *where*

- $\mathcal{S}$ *is a set of* ***states****,* $s_0 \in \mathcal{S}$ *is the* ***initial state****,*
- $\Delta \subseteq (\mathcal{S} \times \mathcal{P}(U) \times \mathcal{P}(\mathbb{O}) \times \mathbb{D} \times \mathcal{S})$ *is a set of transitions.*
- $F \subseteq \mathcal{S}$ *is a set of* ***final states*** *,i.e., where the computation stops*
- $\alpha : F \times \mathbb{R}_{\mathcal{A}}^{\Delta} \to \mathcal{C}$ *is an* ***output*** *function.*

We introduce a notion of equivalence between models, based on the ability of producing the same input-output behaviour.

Definition 5 (Response Equivalence of Models) *Two cognitive models* L_1, L_2*, with* $L_i = (S_i, s_i, \Delta_i, F_i, \alpha_i)$*, are called* ***response equivalent****, if for each input string there exist a trace* ρ_1 *for* L_1 *that leads to a final state* $f_1 \in F_1$ *iff there exists a trace* ρ_2 *for* L_2 *that leads to a final state* $f_2 \in F_2$ *such that* $I_1(\alpha_1(f_1, \rho_1)) = I_2(\alpha_2(f_2, \rho_2))$*, where* I_i *is the interpretation function from* f_i*.*

Other forms of equivalence are, e.g., a *process equivalence*, i.e., that arbitrary partial processes of a model are isomorphic to the processes of a corresponding model, therefore are interdefinable.

4 Embedding Architectures into GCA

In this section we demonstrate an embedding of the two (main) cognitive architectures ACT-R and SOAR into the GCA. In the following we restrict our embedding to the most relevant aspects of the architectures due to space limitations.

4.1 Embedding ACT-R in GCA

Following Definition 1, it suffices to show that we can write ACT-R as $\mathcal{A}_{\text{ACT-R}} = \langle \Sigma, \Gamma, \mathcal{C}, \mathcal{M}_{\text{ACT-R}}, E_{\text{ACT-R}}, \mathcal{F}_{\text{ACT-R}} \rangle$. The concrete Σ, Γ, and chunks $\mathcal{C}$ are model dependant. Hence, we specify the set of modules $\mathcal{M}_{\text{ACT-R}}$ and the information flow relation $\mathcal{F}_{\text{ACT-R}}$. ACT-R consists of the declarative module (m_d), the procedural module (m_p), the goal module (m_g), the imaginary module (m_i), and the input environment modules $E_{\text{ACT-R}} = \{m_v, m_a\}$, where m_v is the perceptual modules for the visual input and m_a is the module for the auditory input. Each module m consists of specific operations O_m and a guard function g_m. So taking the declarative module m_d as an example, we would have $O_{m_d} = \{\text{retrieve}, \text{modify}\}$. Hence, we can set $\mathcal{M}_{\text{ACT-R}}$ to $\{m_d, m_g, m_i, m_v, m_a\}$. Note that the production module in ACT-R contains only the production rules which are represented by the transitions. Buffers are not explicitly represented in our definition with the exception of the declarative module, where ACT-R allows to have chunks as well within the module and as possibly one chunk in the buffer. In our definition

of a cognitive architecture this could be modeled if necessary by an additional module m_{db} simulating the declarative buffer and the restriction that only chunks of the declarative module can move from and to m_{db}, via the information flow function $\mathcal{F}_{\text{ACT-R}}$.

The production rule system of ACT-R consists of three steps: *match*, *select*, and *execute*. The *matching* of production rules is represented by the transitions between states, i.e., there is only a transition if the precondition of the corresponding production rule is satisfied. Depending on whether probabilistic processes, like noise in the utility calculation, are activated in an ACT-R model, the transition system is either deterministic, i.e., there is only one outgoing transition corresponding to the production rule with the highest utility, or non-deterministic. One transition models the *selection* of one matching production rule. The *execution* of production rules is visible throughout the states.

The subsymbolic activation of chunks can be modeled with the retrievability function α of a cognitive model. Depending on the past retrievals of a chunk c, which are accessible through a trace ρ, it can be decided if a chunk is retrievable, i.e. if the base-level activation $B(c,\rho)$ (Anderson, 2007) is above a certain threshold τ. ACT-R can use in fact up to three components for the activation of a chunk but due to space limitation we do not outline partial matching and spreading activity.

4.2 Embedding SOAR in GCA

We will now outline the embedding of SOAR into our GCA. We have $\mathcal{A}_{\text{SOAR}} = \langle \Sigma, \Gamma, \mathcal{C}, \mathcal{M}_{\text{SOAR}}, E_{\text{SOAR}}, \mathcal{F}_{\text{SOAR}} \rangle$. Here $\mathcal{M}_{\text{SOAR}}$ is the fixed set of modules $\mathcal{M}_{\text{SOAR}} = \{m_w, m_s, m_e\}$, where m_w represents the working memory, m_s the semantic memory and m_e the episodic memory. Since SOAR creates new objects in the working memory for sensory inputs, it is $E_{\text{SOAR}} = \{m_w\}$. As most information processing in SOAR is performed in the working memory, we restrict the information flow in that way that either the source $x = (g_x, O_x)$ or the target module $y = (g_y, O_y)$ is the working memory: formally $\mathcal{F}_{\text{SOAR}} = \{(x,o,y) \in \mathcal{M}^2_{\text{SOAR}} \mid o \in O_x \text{ and } (x = m_w \text{ or } y = m_w)\}$. A concrete SOAR model defines then elements of Σ,Γ and $\mathcal{C}$.

We use a state (C,I,ℓ) of the GCA to represent the memory content of the SOAR system before the input phase. C is the set of all objects that appeared in the execution of the SOAR model. The function I assigns to every object the values of the WMEs and ℓ assigns to the modules (different memory's) contained objects.

The production rules in the production memory of a SOAR model control the specific behaviour of the model. This is captured by our GCA through a concrete cognitive model $L = \langle \mathcal{S}, s_0, \Delta, \mathcal{F}, \alpha \rangle$. Since states capture the situation of SOAR before the input phase an embedding of a SOAR model has a transition $(s_1, d, \Omega_1, u, s_2)$ between states s_1 and s_2 if in SOAR, after receiving the inputs u, a production employs an operation Ω_1 that is (possibly) selected and following the changes in the application after time d has passed and output phases leads to the destination of the transition.

5 Empirical Example

A core motivation for proposing a general framework is to make *implicit* (that is partially hidden) modelling approaches *explicit*. In the following we consider the current best cognitive model for recognition memory (Kellen, Klauer, & Bröder, 2013).

5.1 Test case: Modeling recognition memory

A recognition memory experiment consists of two phases: The first is a *learning phase* where participants learn several new items, e.g. from a word list. The second is a *recognition phase* in which the participants are presented with items they have already learned ("old") and new items. The cognitive processes for this specific task are modeled by the 2-HTM shown in Fig. 2.

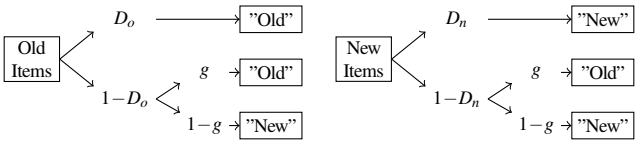


Figure 2: The Two-High-Threshold model (2-HTM).

The cognitive model is presented as a multinomial process tree (MPT) (Singmann & Kellen, 2012). It consists of two trees with the word type, old or new, specified in the tree's root. The tree represents transition probabilities to reach specific states; in the case that an old item is presented to the participant, she discovers with probability D_0 (state of *target detection*) that the item is an already presented item, and gives the response "old". If a new word is presented, she discovers with probability D_n that it is a "new" item. However, the participant can in either case be uncertain and has to guess the answer. This is represented by a subtree structure, where if the item is not remembered, the word is guessed as "old" with probability g and as "new" with probability $1-g$. We model the 2-HTM both in ACT-R and SOAR and compare them in our general framework.

5.2 Cognitive Model in ACT-R

For the following ACT-R model, we assume that the learned words ("old items") are already stored in declarative memory. The recognition phase is implemented as follows: First the system waits for a word to appear on the screen (handled by a production rule). When a word is shown it is stored in a chunk in the imaginal buffer. The guessing part is handled through two production rules with the same preconditions but different answers, `guess-old` and `guess-new`. The production rule with the higher utility is selected. After the response is determined, it will be returned on the screen and compared with the real answer.

An intuitive way to model the 2-HTM is to use the activation threshold of chunks to distinguish between old and new words. In the experiment phase the model tests if a matching chunk can be retrieved. If not, the activation is below the threshold or it is a new word. In this case the model guesses the answer. The problem of this approach is that the guessing path is considered every time the retrieval of the chunk failed. The probability of detecting something as new is not given.

If we do not consider the subsymbolic aspects of ACT-R and model the 2-HTM directly as in Figure 2, these limitations can be overcome. We modify the previous model and use the chunk retrieval process solely as a way to distinguish between old and new

words. We choose a low threshold for chunk retrieval and disable the base-level learning parameter, meaning that the base-level is a constant value determined by a parameter. The consequence is that a matching chunk is always found in the presence of an old word. If the retrieval process is successful, a specific `recall` production rule is triggered, otherwise `cannot-recall`. Both production rules solely switch the states to identify if there was a new or an old word. For each of these production rules we need more rules to handle either the direct answer or the guessing phase. Hence, it is possible to model the probabilities of the 2-HTM with ACT-R. We can set an initial utility value for each production rule and enable a noise parameter with a probability distribution, which is added to the utility. We tested the ACT-R model with 10,000 runs[4] and the model fits are $D_n = 31.74$ and $D_o = 47.02$, resembling the results by (Bröder & Schütz, 2009).

5.3 Cognitive Model in SOAR

The subsymbolic mechanism of SOAR allows the specification of probabilities for the breaking of ties for proposed operators. We can use this to model a decision of the 2-HTM with the three fixed parameters D_n,D_o and g straightforward: if the system gets an item i as input, the system queries the semantic memory if i is already known. If the query returns with an answer, the system proposes two operators o_1,o_2 and equip o_1 with the probability D_n and o_2 with the probability $1 - D_n$ (respectively D_o and $1 - D_o$). If o_1 is selected the system outputs the right result, otherwise the system proposes in the next step two operators equipped with the probabilities g and $1 - g$, and proceeds in dependence of the selection.

5.4 Comparing the Models in GCA

Let $W_o = \{w_1,...,w_m\}$ and $W_n = \{w_{m+1},...,w_{m+n}\}$ two disjunct list of words, where W_o is the set of words that the participant received in the learning phase and W_n the set of new items for the recognition phase of the 2-HTM.

Embedding the ACT-R Model. We can translate the ACT-R model into our GCA as $\mathcal{A} = (\Sigma,\Gamma,C,\mathcal{M}_{\text{ACT-R}},E_{\text{ACT-R}},\mathcal{F}_{\text{ACT-R}})$ with $\mathcal{M}_{\text{ACT-R}}$,$E_{\text{ACT-R}}$,$\mathcal{F}_{\text{ACT-R}}$ described as above (cf. Section 4.1). We have the set of chunks $C = \{c_{o1},...,c_{o10},g,c_{n1},...,c_{n10},c_v\}$ with c_{oi} for the old words, stored in the declarative memory, g as goal chunk, c_{ni} for the new words and c_v as representative for the word on the screen. The new words are the only inputs from the environment, i.e. $\Gamma = \{c_{n1},...,c_{n10}\}$.

For the cognitive model $L_{\text{A-2HT}}$ we need the set of states $\mathcal{S}$, a set of admissibile transitions Δ, final states F and the output function α. The model does not use the base-level activation in the declarative memory which leads to a special guard function of the declarative module: $g_{m_d}(c,\text{retrieve},\rho) = 1$ if $c \in l(m_{db})$ and 0 otherwise. Let $\mathcal{S} = \{s_0,s_1,s_2^o...,s_9^o,s_2^n...,s_9^n\}$ be the states with the final states $F = \{s_9^o,s_9^n\}$. The states and transitions can be seen in Figure 3a.

In the initial state $s_0 = (C_0,I_0,l_0)$ we describe the situation that the participant has learned the old words, i.e. $C_0 = \{c_{o1},...,c_{o10},g\}$ with $I_0(c_i) = (\text{word} : w_i)$ for all $1 \le i \le n$ and $I_0(g) = (\text{state} : \text{start})$. All the chunks are located in the declarative memory, so we have $l(m_d) = C_1$ and $l(m) = \emptyset\ \forall m \in \mathcal{M}_{\text{ACT-R}}, m \neq m_d$. s_1 describes the state that a word appeared on the screen and the participant has to read it. Notice that in this state the word has not been read yet and therefore is not as a chunk in the state yet. Then we change the state, $I_1(g) = (\text{state} : \text{attending-probe})$. After the participant reads a word on the screen, a chunk is created in the visual buffer and the word is stored in the imaginal buffer. At this point the system creates a state for each word that may appear but for the sake of clarity we combine the state of each old word in the state s_2^0 and the new words in s_2^n with

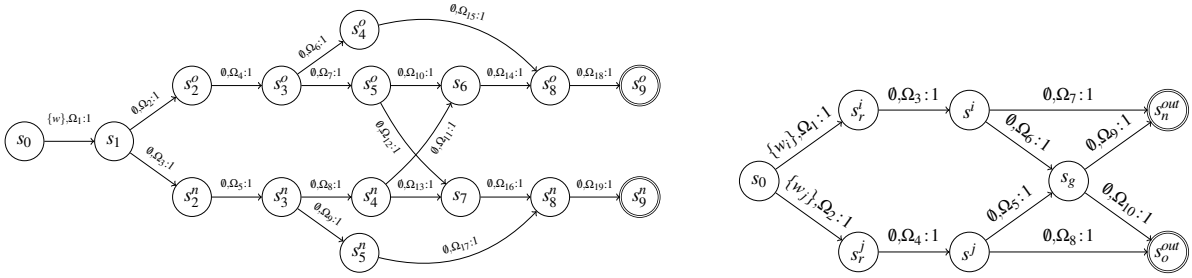
$$C_2 = C_0 \cup \{c_v\}, l_2(m_v) = c_v, l_2(m_i) = c_v \text{ and } I_2(g) = (\text{state} : \text{testing}).$$

In the following we focus on the path for an old word. s_3^o then describes the successful retrieval of the word with $l_3(m_{db}) = c_i$, where c_i is the corresponding chunk from declarative memory, and $I_3(g) = (\text{state} : \text{handle-old-item})$. In the presence of an old word the participant either gives the answer directly, s_4^o, or does not know the answer, s_5^o, and has to guess. The guessing is handled by the states s_6 and s_7 for both paths. State s_8^o then describes that the answer "old" is given leading to the state s_9^o where the actual word appears on the screen. The path of new words is analogous.

Embedding the SOAR Model. Applying the embedding from Section 4.2 to the SOAR model 2-HTM (see Section 5.3) leads to the following result: Let $\mathcal{A}_{\text{SHTM}} = \langle \Sigma,\Gamma,C,\mathcal{M}_{\text{SOAR}},E_{\text{SOAR}},F_{\text{SOAR}} \rangle$ with $\Sigma \setminus \Gamma = W_o \cup \{\texttt{s},\texttt{r},\texttt{o},\texttt{n},\texttt{g},\texttt{on},\texttt{oo}\}$, $\Gamma = W_n \cup W_o$ and $C = \{c_1, \ldots, c_m, \text{STATE}, \text{OLD}, \text{NEW}\}$. Note that $\{\texttt{s},\texttt{r},\texttt{o},\texttt{n},\texttt{g},\texttt{on},\texttt{oo}\}$ are symbols from the SOAR model, that are necessary for the model to be internal self aware of the states. Due to space reasons we only outline the definition of the cognitive model $L_{\text{SHTM}} = (S,s_0,\Delta,F,\alpha)$: Expanding the space of reachable states leads to the structure in Figure 3b, where for every $1 \le i \le m$ a copy of the state s_r^i exists (the same applys to s_r^j for all $m+1 \le j \le m+n$). For example we obtain $s_r^i = (C,I_r^i,\ell_r^i)$ for all $1 \le i \le m$ with $I_r^i(c_i) = \{\texttt{word} : w_k\}$ for all $1 \le k \le m$, $\ell_r^i(m_w) = \{\text{STATE}, c_i\}$, $I_r^i(\text{STATE}) = \{\texttt{s} : \texttt{r}, \texttt{word} : c_i\}$ and $\ell_r^i(m_s) = \{c_i \mid 1 \le i \le m\}$. The final states are $F = \{s_n^{out},s_o^{out}\}$, and we define the output function as $\alpha(s_x^{out},\rho) = \begin{cases} \text{OLD} & x = o \\ \text{NEW} & x = n \end{cases}$ with $I_o^{out}(\text{OLD}) = \{result : \texttt{o}\}$ and $I_n^{out}(\text{NEW}) = \{result : \texttt{n}\}$.

A comparison. Our core motivation demonstrated on this example is to have a common ground to compare different architectures and models. As we saw for the 2-HTM in ACT-R, there are some limitations on the implementation that do not allow for the intuitive model to represent guessing. We were forced to use the utilities functionality, which seems to be implausible from a cognitive science perspective. However, if we apply the Definition 5 of equivalency we see that both the SOAR model and the ACT-R model are response equivalent, i.e., they generate the same response for the same input. Still, the processes slightly differ as can be seen in Fig. 3. This indicates too that if modules in two different architectures have a different specification in the GCA then these cannot be process equivalent. Future work will investigate restrictions.

[4] using parameter settings: (:rt -2, :bll nil, :esc t, :egs 1)



(a) Embedding the ACT-R model in GCA

(b) Embedding the SOAR model in GCA

Figure 3: Embeddings of the 2-HTM into the two architectures ACT-R and SOAR

6 Conclusion

We propose a formal framework general enough to embed different cognitive architectures into it, inspired by the idea that all architectures represent to some degree insights into the cognitive processes on the "data structure" of the human mind. The motivation of our approach is twofold. First, to lay the ground for supporting/rejecting, classifying, and categorizing cognitive mechanisms provided by a cognitive architecture – something which requires a general framework. Second, to resolve the desire to "fully explore the space of possible computational cognitive architectures [...] to further advance the state of the art in cognitive modeling" (Sun, 2007). We outlined conceptually how ACT-R and SOAR can be translated into the GCA. Furthermore, a notion of equivalence for cognitive models was formalized based on the GCA, demonstrated its power on two exemplary 2-HTM models. This is a first step towards a common ground for comparing more cognitive architectures in a formal way. Future work will focus on embedding more architectures, compare them on a broader variety of cognitive models, and investigating the comparison of general concepts of modeling and the impact of restrictions within this new framework. Having more embeddings available, the general cognitive architecture will allow us to perform a comparison of the different approaches and to classify the necessary conditions that are required to model a task. From a theoretical perspective, an analysis of the different architectures and their implicit and explicit assumption is possible. Furthermore, it allows a formal analysis of the limitations of recent approaches, e.g., the declarative knowledge module in ACT-R which makes the incorporation of guessing difficult.

7 Acknowledgments

This work has bee supported by DFG-Grants RA1934/3-1, 4-1, and 5-1 to MR, KE1413/10-1 to GKI and BE 1700/9-1 to CB.

References

Albrecht, R., & Westphal, B. (2014). F-ACT-R: defining the ACT-R architectural space. In *Cognitive Processing* (Vol. 15, pp. S79–S81).

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Bröder, A., & Schütz, J. (2009). Recognition ROCs are curvilinear—or are they? On premature arguments against the two-high-threshold model of recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *35*(3), 587.

Gall, D., & Frühwirth, T. (2015). A Refined Operational Semantics for ACT-R: Investigating the Relations Between Different ACT-R Formalizations. In *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming* (pp. 114–124). New York, NY, USA: ACM. doi: 10.1145/2790449.2790517

Kellen, D., Klauer, K. C., & Bröder, A. (2013). Recognition memory models and binary-response ROCs: A comparison by minimum description length. *Psychonomic Bulletin & Review*, *20*(4), 693–719.

Laird, J. E. (2012). *The soar cognitive architecture*. The MIT Press.

Marr, D. (1982). *Vision. a computational investigation into the human representation and processing of visual information*. San Francisco: Freeman.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Singmann, H., & Kellen, D. (2012). MPTinR: Analysis of multinomial processing tree models in R. *Behavioral Research Methods*, *45*(2), 560–575.

Stewart, T. C., & West, R. L. (2007). Deconstructing and reconstructing ACT-R: Exploring the architectural space. *Cognitive Systems Research*, *8*(3), 227 - 236.

Sun, R. (2001). *Duality of the mind - a bottom-up approach toward cognition*. Lawrence Erlbaum.

Sun, R. (2007). The importance of cognitive architectures: An analysis based on CLARION. *Journal of Experimental & Theoretical Artificial Intelligence*, *19*(2), 159–193.

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F. V., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. (1995). Intelligent agents for interactive simulation environments. *AI Magazine*, *16*(1), 15–39.