An Introduction to the Cognitive Calculus: A Calculus of the Human Mind

Christian C. Wagner and James G. Schiiller
wagner@oakland.edu jgschiil@oakland.edu
Oakland University, Rochester, MI 48309

Abstract

The Cognitive Calculus is the result of decades of work in artificial intelligence, psychology, linguistics, and systems engineering. It is a notation we use to model human cognition and to guide our development of a Cognitive Database (CDB), our evolving computer model of human cognition. The Cognitive Calculus acknowledges human memory (labeled as Total Memory, TM) as the central component of human intelligence. TM is composed of all memory subsystems including: short term memory (STM – our working memory), episodic memory (EM – our life history), and abstracted memory (AM – our mental models). The Cognitive Calculus defines the basic element of TM as a memory node (MemNo) along with a set of operations intrinsic to the creation, retrieval, update, and deletion of MemNos. All content in all of TM is a sensory input, an effector activity, a cognitive activity, an abstracted set (T) or an abstracted sequence (Q). This paper describes how a human cognitive system takes the experiences of life in as inputs to STM, passes those inputs through STM into EM, finds patterns in EM for creating abstractions in AM, and then uses those abstractions of the past to comprehend its current experience. The paper ends with guidance concerning the things that must be in a Cognitive Database so that a computer can better model human cognition.

Introduction

Cognitive calculus is an attempt to model human cognition. The word *cognitive* comes from psychology and references the mental processes of the mind. The word *calculus* comes from formal systems and means a formalism that defines the interrelationships between abstract objects. The reason cognitive calculus is a calculus is because it takes the functions observed in human cognitive behavior and describes how they are done operationally. Cognitive calculus uses theories and approximations with no proof of their correctness, as no one knows how human cognition functions.

The fundamental element of cognition is not a unitary object but, instead, a buffer of *N* cognitive chunks held in STM at the current moment of time. A basic flaw in most prior cognitive modeling techniques has been to ignore this automatic aggregative mechanism. Because it seems like there is a unitary concept like “dog” or “cat” or “computer,” past research has assumed that there was a “node” or a “nexus” that represented the concept. This is not the case. Instead, there is a set of STM buffers, each of *N* elements, filled by recursive references to other STM buffers, which recursion finally terminates in the sensory, motor, and cognitive realities of the human body. Some of the ideas have been suggested in the work by (Holland et al. 1986), but no one has examined the singular importance of the fundamental element of intelligence. The fundamental element of intelligence is the human STM buffer. The motivation for this work is based on the need for a new kind of database modeled on human cognition, called a Cognitive Database (CDB), and a new notation to talk about a CDB, called The Cognitive Calculus.

Basic Element of Cognition

The Cognitive Calculus begins with a memory node (MemNo) as the basic element organizing human intelligence. It is called a MemNo instead of a chunk, so there is no confusion between the psychological definition of the term chunk and other ideas about the definition of the term chunk.

Memory Node

Generally speaking, a MemNo is the basic aggregative mechanism in the intelligent mind – it is a buffer of up to *N* elements that captures the co-occurrence of the mental state at one moment in time. The parts of a MemNo are themselves other MemNos. With this recursion, the information represented by a MemNo may be anything from the most basic physiological state (e.g., “I am hungry”) to a complex and deeply recursive concept (e.g., “Humanity has caused great pollution in the world”).

MemNos are of many types, as all conscious aspects of human existence are represented therein. There are five types of MemNos – Sensory, Motor, Cognitive, Object and Relationship. The first three are primitives – meaning that they are grounded in the reality of the human body, we SENSE, we MOVE, and we THINK. The last two are aggregative in nature, summarizing experience where order is important, or when order is irrelevant. The MemNo types are:

- Sensory MemNos are the summation of a collection of sensory experiences into a concept (sight, hearing, touch, taste, smell, balance, thirst, hunger, and so on; for example, input from the eye recognized as the concept “blue”).
- Motor MemNos are the summation of a collection of commands that drive the physical effectors of the human experience into a concept (muscle movement in all its forms; for example, output of moving the hand back and forth recognized as “wave”).
- Cognitive MemNos are the summation of a collection of cognitive tasks into a concept (decide, choose, think, remember, and so on; for example, the cognitive task of looking through a set of alternatives and selecting one recognized as “choose”).
- Object MemNos are the summation of an unordered collection of MemNos into a concept (generalization and abstraction of objects: for example, the unordered concepts of “furry-1”, “tail-3”, and so on are recognized as defining the concept “dog-2”).
- Relationship MemNos are the summation of an ordered collection of MemNos into a concept (relationships that are verbal, prepositional, and so forth; for example, the concepts of MemNos “eat-4”, “man-2”, “sandwich-2” are recognized as the concept “man-eats-sandwich”).

Operations on Memory Nodes

A set of operations are intrinsic to the operation of MemNos. As one might expect from ordinary database operations, the four CRUD operations are critical to MemNos:

- The create operation can create any one of the five different MemNo types. Actions of the body have automatic predefined means for doing this. Object and Relationship represent abstracted and learned knowledge.
- The retrieve operation can retrieve MemNos associated with specific senses, effectors, cognitions, sets, and sequences.
- The update operation can add an instance to an object MemNo, add an instance to a relationship MemNo, change the goodness of a MemNo, change the certainy of a MemNo, and the like.
- The delete operation can drop useless MemNos, drop excess examples from an object MemNo, and drop excess examples from a relationship MemNo.

As seen below in the functional notation, when any of the operations are applied to a set of MemNos (M_i), the result will be a new MemNo.

$$M_{new} = \text{operation}\ (M_a, M_b, \ldots)$$

The arguments are MemNos and the operations result in new MemNos. This property of MemNo operations is important because the operations that compare words can compare higher level elements, such as concepts, through recursion. For example, the type of process that recognizes the power button on the radio can, through recursion, recognize the radio in the dashboard, and the dashboard in the car. Only the content of the process changes, not the process itself. Operations will be described in another paper.

Total Memory

In the computer model of human cognition, TM is equivalent to all human memory. TM is composed of all memory systems including:

- STM – short term memory, the current moment – the current working memory that holds the last few things that happened to the cognitive system
- EM – episodic memory, life experience – the sequence of STM values over time
- AM – abstraction memory, mental models – the generalization of EM into predictions for the future

$$\text{TM} = \text{STM} \times \text{EM} \times \text{AM}$$

STM is the portion of human memory that models the current moment of experience. And, as that current moment passes, some partial copy is transferred into EM. As processing time permits, the experiences of EM are generalized into AM. Note that all parts of TM hold nothing more than MemNos – the recursive structural element of human cognition.

Short Term Memory

In the human brain, there is a network of concepts built from experiences and connected through some kind of STM medium. The cognitive system builds this network of concepts from those experiences in memory. The key part missing from past computer models (Shortliffe 1975), (Schank & Abelson 1977), (Fahlman 1979), (Holland 1986), and (Minsky, 1988) is the emphasis on the computer equivalent of the human STM, which The Cognitive Calculus suggests bringing into focus.

At every point of consciousness, human STM is filled with MemNos. STM is an array of *N* such MemNos. For the purpose of this paper, *N* is generally fixed; here we use the number five. STM is visually laid out in figure 1.

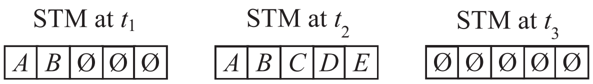

Figure 1: Visual layout of STM.

The three STMs in figure 1 are taken at consecutive times – as time passes, working memory changes. Time (t) marks when the data entered the STM buffer. As shown, there are times when STM is receiving inputs, when the experience *AB* entered STM at t_1. At the next time, STM is full because experience *ABCDE* entered STM at t_2. Finally, there are null or idle times (Ø) when the cognitive system is doing nothing, such as at t_3. Time is a critical dimension, as the concept of causality begins with co-occurrence of elements in STM at a particular moment. Time is measured internally by a cognitive timing function that is separate from our external measures of time.

Short Term Memory Size

The size of STM is an important parameter of intelligence. It is well-known that people can only remember five to nine bits of information after the original information is no longer present (Miller 1956). But why that size? It is our conjecture that, if the size of STM were smaller, humans could not comprehend the complexity of life because STM capacity would be exceeded. We need to be able to connect together at least 5 things. Similarly, if STM were larger, we would have greater discriminative power than our world requires – an inefficient design choice. Previous work in natural language understanding seems to indicate that about 99% of language fits into a branching factor of five (Kijek 2007). Five MemNos are all that is needed to process language. The human STM size specification is the evolutionarily perfect size. Of course, the physiological constraint of human cognition does not constrain the computer model in the CDB. Increasing its size is a fascinating research direction.

Each time there is a change made to STM, it is recorded in EM. The current inputs pass through STM into EM, as shown in figure 2 below.

Figure 2: Input passes through STM into EM.

Episodic Memory

Life happens in sequence. EM is the memory for the sequential events of experience. A cognitive system takes the experiences of life in and creates a memory representation from them. EM is the sequence of everything that ever happened to this intelligence (its life history). EM is never deleted. One cannot go back and say something did not happen to it. See figure 3 for the visual layout of EM.

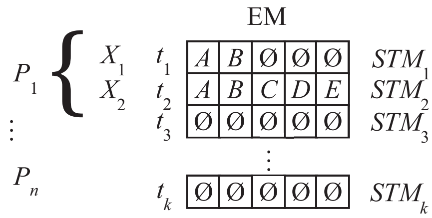

Figure 3: Visual Layout of EM.

EM is a unitary sequence of life. Yet, within it, moments of quietude (e.g., t_3 in the diagram) indicate that our episodic memory includes within it separate episodes *(P*) that are composed of a set of experiences (*X*). For example, this cognitive system experienced episode P_1 from the sequence X_1, X_2 before it entered a state of quietude at time t_3. EM has the same basic structure as STM except there are *k* rows instead of one.

In our CDB, the entire life experience of the CDB is stored in a file. This is the most important file created in the system. If the system somehow dies, there is a way to reconstruct it. The system can be recovered by rerunning its entire life.

Abstracted Memory

A cognitive system's AM is the generalizations of experience stored in the EM that are used to predict the future. This prediction is done by presenting suggestions of MemNos that should be added into the current STM. These suggestions, of course, compete with the sensory, motor and cognitive systems that also suggest candidates to be entered into STM at the current moment.

AM is the result of abstracting from experience. Intelligent life takes the episodes of life (P_i) and converts them into useful abstractions for predicting the future. For example, witnessing a fellow villager being eaten by a lion, an intelligent system might abstract a problem when the roar of a tiger is heard. The purpose of abstraction is to build a mental model of life from the content of EM. The methods for creating, validating, using, and forgetting these mental models are a key aspect of AM. Note that this question how to abstract is also a key to the work of Numenta and Hawkins (Hawkins & Ahmad, 2011). See figure 4 for the visual layout of AM.

	AM					
LTM_1	*d*	*o*	*g*	Ø	Ø	*Sensory (see "dog")*
LTM_2	*t*	*a*	*i*	*l*	Ø	*Sensory (see "tail")*
LTM_3	1	2	Ø	Ø	Ø	*Object (a dog)*
LTM_4	*w*	*a*	*g*	Ø	Ø	*Sensory (see "wag")*
LTM_5	3	4	Ø	Ø	Ø	*Relation (a dog wags)*

Figure 4: Visual Layout of AM.

AM has the same basic structure as EM, except the MemNos are more tightly compressed and each object (T, a set) and each relation (Q, a sequence) fits into long term memory (LTM). Notice in AM that time is unimportant. AM is built from l long term memories, which in turn are each built recursively from N MemNos.

Abstracted Sets and Sequences

All content in TM is a MemNo. Three of the MemNo types are primitive – sensory, motor, and cognitive. Two types are recursive, the abstracted sets (Ts) and sequences (Qs). People have millions of these Ts and Qs. The names could be T_1, T_2, Q_1, Q_{15}, Q_{94}, or Q_n because the name does not matter to the computer. For all it cares, the set of all the nouns could be named T_{13}.

There may be only one mechanism in the brain for both T and Q abstraction; both abstractions have the same prototype, an abstracted list of MemNos. The only difference is Q values order and T does not. For example, the object man has a set of MemNos connected to it, e.g., {parts, face, beard, head, arms, legs}, and {instances, George, Mike, Bob, Larry}, and {actions, eats, drinks, runs}. Another example of a T is the equivalence set {cat, dog, rat}. All the elements in this set belong to the same category and order does not matter.

In contrast, each Q is a relationship where the sequence does matter. For example, the recipe to bake a cake: ⟨break two eggs into a bowl, mix the eggs into the batter, and so on⟩ or to cook a steak: ⟨heat up the pan first then put in the steak⟩ has to be done in that order. The sequence: ⟨the big red pocket knife⟩ has a very different meaning than ⟨the big red knife pocket⟩.

All the familiar operations of set theory can be applied in The Cognitive Calculus because everything in AM is MemNos. For example, {boy, man, men} ∪ {women, girl, girls} = {boy, man, men, girl, girls, women}.

Preliminaries of a Cognitive Database

Theories are all well and good, but the practicalities must be proven. This last portion of the paper speaks to initial design aspects of a cognitive database (CDB).

In the human brain, verbal and visual processing occur in different parts of the brain. We do not posit that visual recognition of an apple is done via the verbal processing of a sentence. The Cognitive Calculus is fundamentally based on verbal processing, a linguistic model of cognition, so that is where the CDB begins. A major design constraint is simply not to start at ground zero! The computer is not an intelligent system. We give it as much information as possible to get things up and running quickly, and therefore start with a simple morphological analyzer.

Morphological Analyzer

For morphological analysis, we start with the model of Residential Grammar (Binkert, 1984). A word can belong to one or more syntactic types, such as adjective_1, adjective_2, …, adverb_1, adverb_2, …, determiner_1, determiner_2, noun_1, noun_2, …, verb_1, verb_2, … (Wagner & Binkert n.d.). For example, the word *ran* is a verb_3, so the morphological analyzer gives the MemNo representing the sensory experience of the word *ran* the special marker for verb_3. Words also have a root or base form. The base form of *ran* is *run*, so that is added to the MemNo. Further, the form of the verb *ran* is past tense… another marker to add. Without special markers, the computer would not know that words, such as *run* and *ran*, are related. Contained in a sentence, a word, such as *take* would be marked as "*the man* (*took* = take*verb_2*past*English*) *the dog on the boat*." There are many more of these types of properties on words. Marked words have a general form.

word = *base form*syntactic types*class*language*

Many words have multiple syntactic types, and even base forms, vying for the same slot. When this happens, only context can help disambiguate and choose the proper classification. Though at some future time it might be interesting to see if such relationships can be learned from scratch, the CDB approach says to keep it simple. Add in the known morphological analysis and move on.

Grammatical Abstraction

Sensory input comes into the brain faster than can be processed, so at night when we sleep (offline), we dream and try to organize inputs by integrating or connecting things. Down time is when T and Q are most easily created and changed. In The Cognitive Calculus, the four transformational rules from transformational grammar (insertion, deletion, substitution, and movement) (Chomsky 1965) are applied in a different way we call grammatical abstraction.

There are a set of three grammatical abstraction rules: {ins/del, sub, mov}. Ins and del can be seen as the same rule. The initial abstraction algorithm (AA_1), given in pseudo code, processes through the episodes of life in EM and creates useful abstractions in AM:

```
foreach Episode P in EM do
   for i = 1 to (# Experiences in P) -1 do
      for j = (i+1) to (# Experiences in P) do
         Experience X_1 = P[i];
         Experience X_2 = P[j];
         Abstract_1 (X_1, X_2);
```

The actual grammatical abstraction rules are applied inside the abstract function.

```
Abstract_1 (Experience X_1, Experience X_2)
   if is_ins (X_1, X_2)
      ins (X_1, X_2)
   if is_sub (X_1 X_2)
      sub (X_1, X_2)
   if is_mov (X_1, X_2)
      mov (X_1, X_2)
```

Of course, pseudo code cannot capture the true complexity of the process. For example, the if_sub predicate checks to see if two experiences are identical in form except for one position where they each have a different MemNo. If X_1 and X_2 do differ only in this one way, a potential object MemNo (T) is created along with a tentative relation MemNo (Q). If X_1 = "*the dog*" and X_2 = "*the cat*", then is_sub is true, and the sub action proceeds. The word *the* is the same on the left sides of both sentences, so a new sequence can be created, $Q_1 = \langle \text{the} \rangle$. Then, the words at the substitution position, *dog* and *cat*, are added to a new set, $T_1 = \{\text{dog, cat}\}$, and then added to $Q_1 = \langle \text{the } \{\text{dog, cat}\} \rangle$, resulting in $Q1 = \langle \text{the } T_1 \rangle$. This means *the* can be followed by either *dog* or *cat*. The other rules have similar logic to the sub rule.

Phrase Structure Grammar

Chunking is the main operation in STM. Any effort to understand a sentence must demonstrate chunking if it is to accurately simulate intelligence. A good explanation of the idea of chunking is taken directly from (Wagner & Binkert n.d.). For example, the string of numbers 1–4–9–2–2–0–0–1–1–7–7–6–2–0–1–3 can easily be recalled when chunked as in 1492–2001–1776–2013. When information is grouped into chunks, more can be remembered. The same idea of chunking applies to sentences, for example, $y_1 \Leftarrow \langle x_1\, x_2\, x_3 \ldots x_p \rangle$.

English sentences contain three basic units called phrases (Wagner & Binkert n.d.). Phrases can be labeled as noun phrases (N_3), verb phrases (V_3), or characterizer phrases (C_3). A phrase is a group of words acting as a unit. The basic phrase structure for one English sentence can be represented in a parse tree as seen in figure 5.

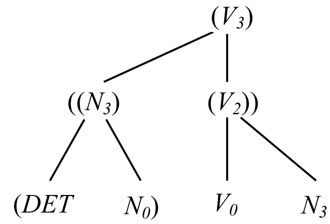

Figure 5: Tree Representation of one Sentence.

Note that in Residential Grammar, V_3 is used to designate a sentence. The phrase structure in this diagram can, of course, also be represented with a bracketed list.

Without chunks and limited by the size of our own STM, sentence lengths would have to be constrained to the size of STM to comprehend them. Chunking is the result of biological constraints on human memory. It is not by accident that language is broken up into phrases, it is rather a reflection of the human brain. We are dealing with the constraints of our human anatomy by making sentences ordered as they are in the parse tree (Wagner & Binkert n.d.).

Supervisory Learning

The human mind can make sense out of anything. The human mind guesses and sees how well the world supports the guess. It throws the guess away if the guess is not supported and useful in the future. For example, a child may guess the past tense of the word *go* by adding an *ed* onto the end as *goed*, similar to the word *jump* with past tense *jumped*. The child can play with a guess and say "*mommy goed to the store*," but the child's dad would need to correct the child by saying "*no, it is went, not goed*." The point is that there needs to be a way to correct the child. The same idea applies to the CDB. The computer is not smart at the beginning and cannot learn on its own without some supervisory learning.

Special Symbols

The CDB uses language to communicate with a teacher. The special symbols used for teaching are: ⟨, ⟩, %, !, -c, -g. When the CDB senses these special symbols, it detects a teacher and switches to learning mode. The teacher has to be able to tell the CDB "yes" and "no," so CDB can learn good abstractions and unlearn bad abstractions. For example, if the teacher wanted to teach the CDB both a correct and incorrect grouping of "*he ate the food*," the teacher would use a combination of special symbols, such as ⟨he ⟨ate ⟨the food⟩⟩⟩ !⟨he ate ⟨the food⟩⟩ to show that the first sequence is correct and the second sequence is not (!) correct. Likewise, properties can be learned by using special symbols, such as ⟨ate he ⟨the food⟩⟩, which is a two-place predicate (ate he, food). This is the way properties are defined, through the predicates. Predicates are the verbs and prepositions of a sentence.

Certainty and Goodness Measure

CDBs reason with certainty like humans. The teacher can attach a certainty factor (CF or -c) to an abstraction, where CF values range from -100 to +100%. The negative values are evidence against the truth of an abstraction, from (Shortliffe, 1975). The teacher can also attach a goodness measure (-g) to an abstraction to indicate how useful the abstraction is to the system. For example, attaching -g +1 to an abstraction would be understood by the CDB as adding one point of usefulness to the said abstraction.

After the initial training, a CDB has something to help guide learning. For example, if the teacher taught the CDB to always group ⟨the lion⟩, the CDB would always group it that way, assuming the group continued to be useful to the system. From the grammatical abstraction, CDB would learn to substitute the word *lion* with *tiger* and group as the teacher showed it with the substitution, ⟨the tiger⟩. Despite the teacher being a good source of abstraction, the CDB confirms and discords abstractions with -g and -c, as it reads on its own.

Guessing

Guesses are not always right. In fact, many times they are pure speculation, and it is okay; the guess was just a possibility or an assumption to be tested. For instance, if the CDB guessed at grouping "*the man took the dog*" as ⟨the man ⟨took the⟩ dog⟩ and resulted in separate groups with goodness attached as ⟨the man⟩ -g 4000, ⟨took the⟩ -g 2, ⟨dog⟩ -g 8. The group ⟨dog⟩ was liked by the CDB eight times. However, grouping the same sentence as ⟨the man ⟨took ⟨the dog ⟩⟩⟩ and the groups with goodness attached as ⟨the man⟩ -g 4000, ⟨took⟩ -g 99487, ⟨the dog⟩ -g 8000; results in the group ⟨the dog⟩ being liked by the CDB 8,000 times. The CDB took a guess at making a group ⟨dog⟩ and found that the guess was wrong.

Learning Meaning

Meaning is sensory elements being integrated and correlated among the senses. For instance, you have pictures of a car, sounds of a car, smells of a car, vibrations of a car, and the word *car*. The association of a word to its senses becomes its meaning.

Thought Experiment

An attempt to learn the meaning of a sentence can now be illustrated. Imagine the first experience that a CDB has is X_1 = "*the dog chased the cat.*" The CDB processes X_1 and creates four new sensory MemNos: S1 – "the", S2 – "dog", S3 – "chased" and S4 – cat," as well as one new relational MemNo: R1 – ⟨S1, S2, S3, S1, S4⟩. The utility of the new MemNos are all initialized to 1.

When the CDB next processes experience X_2 as "*the dog chased the rat*", S5 – "rat" is created as well as R2 – ⟨S1, S2, S3, S1, S5⟩. Applying the AA_1 to these two experiences in the episode creates one new object MemNo, T_1 = {cat, rat}, and one new relation MemNo, Q_1 = ⟨the dog chased the T_1⟩.

As one might expect, the complex building of concepts is quite epiphenomenal, depending upon the exact sequence of experiences of the system.

Conclusion

The Cognitive Calculus and the CDB are evolving works. In the tradition of AI, the model is defined, and then its implications are observed in implementation. As Hawkins has observed (Hawkins & Ahmad, 2011), intelligence is largely held in the human memory system. This paper lays out the beginnings of The Cognitive Calculus to define human memory and the design of the Cognitive Data base (CDB) to model human intelligence.

References

Binkert, P. J. 1984. *Generative Grammar without Transformations*. Berlin, Germ.: Mouton.

Chomsky, N. 1965. *Aspects of the Theory of Syntax.* Cambridge, Mass.: MIT Press.

Fahlman, S. 1979. *NETL: A System for Representing and Using Real-World Knowledge.* Cambridge, Mass.: MIT Press.

Hawkins, J., and Ahmad, S. 2011. Hierarchical Temporal Memory including HTM Cortical Learning Algorithms. Groksolutions.com.

Holland, J., Holyoak, K., Nisbett, R., and Thagard, P. 1986. *Induction: Processes of Inference, Learning, and Discovery.* Cambridge, Mass.: MIT Press.

Kijek, D. W. 2007. Cognitive Inspired Unsupervised Methods to Acquire Linguistic Knowledge from Unlabeled Text. Ph.D. diss., Department of Industrial and Systems Engineering, Oakland University, Rochester, MI.

Miller, G. A. 1956. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review* 63:81-97.

Minsky, M. 1988. *The Society of Mind.* New York, N.Y.: Simon & Schuster Inc.

Schank, R., and Abelson, R. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures.* Cambridge, Mass.: MIT Press.

Shortliffe, E. H. 1975. *Computer-Based Medical Consultations: MYCIN.* New York, N.Y.: Elsevier.

Wagner, C., and Binkert P. n.d. Chapter One: Fundamentals of Linguistics. Lecture, Oakland University, Rochester, MI.