EasyChair Preprint

№ 11028

The Emergence of Compositionality in a Brain-Inspired Cognitive Architecture

Howard Schneider

October 6, 2023

The Emergence of Compositionality in a Brain-Inspired Cognitive Architecture

Anonymous[1]

[1]Anonymous
anonymous@anonymous

Abstract

Compositionality can be considered as obtaining (or synthesizing) the correct meaning of the constituents of a non-simple language expression or visual image. The Causal Cognitive Architecture is a brain-inspired cognitive architecture (BICA). It is not a traditional artificial neural network architecture, nor a traditional symbolic AI system but instead uses spatial navigation maps as its fundamental circuits. In older versions of the architecture, sensory inputs are compared in each existing sensory system against previous stored navigation maps for that sensory system, and the best navigation map is chosen and then updated with the new sensory inputs and a best multisensory navigation map is similarly created and used as the working navigation map. Instinctive and learned small procedures are triggered by input sensory inputs as well as matched navigation maps, and in the Navigation Module operate on the working navigation map and produce an output signal. By feeding back intermediate results in the Navigation Module it has been shown previously how causal and analogical behaviors emerge from the architecture. In new work, the Navigation Module is duplicated in a biologically plausible manner. It becomes possible to compositionally process information in the duplicated Navigation Module against the original Navigation Module, and as a result compositional language comprehension and behavior readily emerge. A formalization and simulation of the architecture is presented. A demonstration example, and then its negation case, are explored of solving a compositional problem requiring the placement of an object in a specific location with regard to other objects. Future work is discussed using large language models to create navigation maps. Given the mammalian brain inspiration of the architecture, it suggests that it is indeed feasible for modest genetic changes to have allowed the emergence of compositional language in humans.

Keywords: Compositionality; Brain-Inspired Cognitive Architecture (BICA); Artificial Intelligence (AI); Language Evolution; Large Language Model (LLM)

1. Introduction

1.1 Compositionality and Neural Networks

Frege's principle of compositionality (Frege, 1884) can essentially be summarized as the meaning of a non-simple expression being a function of the meanings of its constituent parts and the way they are syntactically combined (Partee, 1984). Ruis and Lake describe systematic compositionality or generalization as the ability to produce novel combinations from known parts (Ruis and Lake, 2022). Many somewhat varying definitions of compositionality along these lines exist (Pleyer et al., 2022).

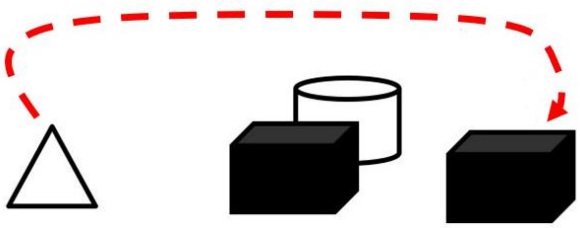

Figure 1. The instruction to "place the white triangle on top of the black block which is not near a white cylinder" may be difficult to successfully follow by connectionist systems. The dashed arrow represents the correct solution to this instruction. (Note: The dashed arrow is *not* shown to the computer system being asked to follow this instruction.)

Consider the compositional instruction "place the white triangle on top of the black block which is not near a white cylinder" as depicted in Figure 1. This requires an ability to extract the meaning from a non-simple expression in terms of its constituent parts and their relations. Symbolic artificial intelligence (AI) systems in the 1970s (e.g., Winograd, 1971) were able to follow such instructions, i.e., possessed compositional abilities. However, as Lake and colleagues (2017) note, while artificial neural networks have had success in many tasks, including those requiring properties of generalization, they do require extremely large numbers of training examples, due in part to a lack of compositional abilities. Liška and colleagues (2018) describe how some recurrent neural networks (RNNs) can show compositional properties although they tend to be weak.

Marcus, Davis, and Aaronson (2022) show even advanced neural networks are largely incompatible with compositionality and have difficulties with problems such as shown in Figure 1. For example, DALL-E2 (a deep learning system that synthesizes often spectacular and complex digital images from English language prompts) is unable to successfully handle relationships between objects. Marcus and coworkers (2022) give an example of prompting DALL-E2 to put a red ball on top of blue pyramid behind a car that is above a toaster. DALL-E2 generates ten different images depicting this request, but none of the images are correct in terms of the specified relationships.

Compositionality is an important property. Without compositionality an artificial system would essentially need to see every permutation of a myriad of situations in order to learn them, i.e., effectively requiring infinite memorization. Compositionality allows humans, for example, to understand and generate an unlimited number of different sentences, images, gestures, and so on, without having to train on billions or trillions of parameters of language, such as occurs with contemporary large language systems (Brushe et al., 2020; Levine et al., 2020).

Given the compositional weaknesses of traditional connectionist systems (Marcus et al., 2022), and given the difficulty of building practical symbolic AI systems (e.g., learning large amounts of data often is impractical and the operation of such systems often proves to be brittle (Hitzler et al., 2022)), there is interest in finding ways to allow neural networks to successfully include compositional properties (e.g., Kim, 2021). One approach is further refinement to the induction of compositionality in RNNs (e.g., Liška et al., 2018) or in transformers (e.g., Jiang and Bansal, 2021). Another approach is neurosymbolic AI which combines the properties of typical artificial neural networks and symbolic AI systems, i.e., artificial neural network learning properties with the ability for symbolic reasoning. The symbolic abilities of such systems allow compositionality to be more readily achieved. A review of neurosymbolic architectures is provided by Kautz (2022) who places them in a number of categories. However, each category of neurosymbolic architecture is essentially an overt hybrid of an artificial neural network with a symbolic AI system.

This paper describes another approach to achieving compositionality in a system which is not a typical symbolic AI system. The Causal Cognitive Architecture is a brain-inspired cognitive architecture (BICA) (Schneider, 2022b, 2023). It is not a traditional artificial neural network architecture, nor a traditional symbolic AI system. Instead, it uses modified neural networks as its fundamental circuits involving manipulations of what are essentially navigation maps (described below). This paper does not focus on the actual neural networks, which can take on a variety of forms as long as a functional navigation map results (allowing straightforward changes and comparisons to cells in these maps) and can be repeated to allow a large number of such maps. In this paper, evolutionarily feasible improvements to the architecture are described which readily allow compositionality including compositional language to emerge in the architecture.

1.2 The Causal Cognitive Architecture: Previous Work

The Causal Cognitive Architecture 5 (CCA5) was previously described in the literature (Schneider, 2023). An overview of this architecture is shown in Figure 2. The new work described in this paper involved experimentation with a CCA5-like architecture. This new work involved brain-inspired evolutionarily plausible changes that resulted in an improved architecture yielding compositional behavior including the emergence of the beginnings of compositional language. This improved architecture is simply termed the Causal Cognitive Architecture 6 (CCA6). An overview is shown in Figure 3, and it is described in more detail in the sections below.

However, in this sub-section the previous work, the Causal Cognitive Architecture including the CCA5 (Schneider, 2023), is briefly reviewed. The Causal Cognitive Architecture does not attempt to replicate the brain at the neuronal spiking level. Similarly, the architecture does not attempt to behaviorally replicate the top-level psychological

manifestations of the mammalian brain. Rather, the architecture attempts to consider what middle level mechanisms could be occurring in the brain, i.e., at approximately the level of the cortical minicolumn and their postulated interactions with each other and other brain circuits and attempts to see what properties and behaviour could emerge. Given the assumptions and approximations being made, it is a loose brain-inspired modeling. It is a functionalist system to a large extent (Lieto 2021, 2021b) although as Lieto (2021b) notes, there is a continuum between functionalist and structuralist models. There are indeed a number of anatomical and biological constraints to the architecture.

The main brain-inspired basis of the architecture is the postulation that just as the hippocampus contains maps of spatial items (e.g., O'Keefe, Nadel, 1978; Samsonovich, Ascoli, 2005; Alme et al., 2014) then so does the mammalian neocortex (Schneider, 2022b). Numerous genetic and developmental mechanisms have been proposed for mechanisms of how brain circuits can duplicate and diverge to new functions, and for the expansion of the neocortex (e.g., Rakic, 2009, e.g., Chakraborty and Jarvis, 2015). Schafer and Schiller (2018) have also hypothesized that the mammalian neocortex contains maps of spatial objects as well as maps of social interactions while Hawkins and colleagues (2019) discuss evidence for the equivalent of grid cells in the mammalian neocortex. Thus, millions of cortical minicolumns in the mammalian neocortex are modeled in the Causal Cognitive Architecture as millions of spatial navigation maps.

Thus, an important aspect of this architecture is that the "navigation map" is used as the main data structure throughout the Causal Cognitive Architecture. The navigation maps used in the Python simulations of the architecture presented previously (e.g., Schneider, 2023) and again in this paper for the improved CCA6 architecture, are greatly simplified. An example is shown in Figure 4. This navigation map is representing the cubes, cylinder, and triangle of Figure 1.

Using this navigation map-based architecture, i.e., the Causal Cognitive Architecture, Schneider (2022a) showed that causal behavior can emerge from a navigation map-based cognitive architecture system previously showing only associative behavior or limited pre-causal behavior. (The limited pre-causal behavior can arise alone from the navigation maps being used in the system). Also, the architecture provided a mechanism which showed that with very small changes to the architecture (i.e., increasing some feedback pathways), evolution from pre-causal primates to fully causal humans was plausible with minimal genetic changes. It also predicted (Schneider, 2020, 2021) that psychosis would more readily emerge in the fully causal humans, something that is indeed well known. For example, van Os and colleagues (2001) show that more than 10% of humans will have psychosis-like symptoms at some point while psychosis is rarely found in other mammals (unlike versions of most of the other human psychiatric disorders) (Xu et al., 2015). Continuing exploration of this navigation map-based architecture, Schneider (2023) showed that with a number of algorithmic changes and additional operations, inductive analogical operations can readily emerge from the architecture. The inductive analogical feedback mechanism allowed significantly enhanced problem-solving skills for the architecture.

Operation of the Causal Cognitive Architecture will be described below in the context of the new work presented in this paper, i.e., the Causal Cognitive Architecture 6 (CCA6).

1.3 New Work

Experimentation with a number of toy problems showed that if the Navigation Module of a CCA5-like architecture (Figure 2) was duplicated into two Navigation Modules, then compositional behavior and the beginnings of compositional language emerged quite readily. This is discussed in the sections below.

As can be seen in the improved CCA6 architecture in Figure 3, the Navigation Modules are now duplicated, along with some of their pathways. Navigation Module B is the partially duplicated module. Numerous genetic and developmental mechanisms have been proposed for mechanisms of how biological brain circuits can duplicate and diverge to new functions (e.g., Rakic, 2009, e.g., Chakraborty and Jarvis, 2015).) As is discussed in more detail below in the paper, this duplication of the Navigation Modules is very evolutionarily plausible, i.e., relatively few genetic changes can result in large changes in a brain-inspired cognitive architecture's behavior. In this case, the architecture can produce compositional solutions and the start of compositional language.

Below in the paper the theory behind the duplication of the Navigation Modules and the operation of the resulting architecture, i.e., the CCA6, are discussed in more detail. The operation of the new architecture is formalized in a series of equations (Appendix A), and a small computer simulation is created which can show compositional results occurring based on these equations.

In an earlier published version of the Causal Cognitive Architecture, it was noted that the basic data structure of the architecture—the spatial navigation maps—resulted in the emergence of explainability and the corresponding potential for language (Schneider, 2022b). Essentially, the series of navigation maps the Causal Cognitive Architecture produced and stored, provided an explanation for any decisions made. In Schneider (2022b) a simple procedure (termed "primitive" in the nomenclature of the architecture) could be used to read back and to communicate these navigation maps to other agents. This primitive, named navigation_map_to_proto_language(),was discussed but no other attempt to comprehend or produce compositional language was made. In this paper, the language abilities of the architecture are extended by showing how compositionality allows the emergence of compositional comprehension.

A number of other changes to the previous architecture are made in the new CCA6 architecture of this paper, in order to better reflect the mammalian origins and inspiration of the architecture. An example is the more distributed storage of the primitives (i.e., procedures) throughout the architecture rather than in one module (e.g., Figure 3 versus Figure 2). While these other changes have no large immediate impact on the properties of the architecture (given the functionalist nature of the architecture) they do allow better correspondence with the mammalian as well as the human brain. The relatively easy emergence of compositional language in this model is then discussed as a possible mechanism in how language could have emerged in humans, despite relatively small genetic differences from a chimpanzee-human last common ancestor.

1.4 Overview of the Paper

In the Introduction section above, the importance of compositionality in intelligent systems is discussed. The compositional weaknesses of traditional connectionist systems are noted. The Causal Cognitive Architecture 6 (CCA6) is presented as a system which is not a typical connectionist system nor a typical symbolic AI system, but which is introduced here as another approach to achieving compositionality.

The second section is titled "Causal Cognitive Architecture 6 (CCA6): Origins and Operation." The brain-inspired origins of the improvements in this version of the architecture are discussed. This is followed by a more detailed and formalized description of the overall operation of the architecture.

The third section is titled "Compositional Language Understanding." The improvements in the architecture discussed in the previous section with some additional instinctive primitives (i.e., built-in procedures that act on the navigation maps containing the data of the system) are shown to allow the start of compositional language understanding.

The fourth section is titled "Demonstration of Compositional Understanding." A simple computer simulation of the new CCA6 architecture is shown to demonstrate language understanding in toy problems.

The fifth section is the Discussion where the results of this work are considered, i.e., the emergence of compositionality in the CCA6 brain-inspired cognitive architecture. The strengths and weaknesses of the results are considered. There is also the consideration of the need for better experimental simulation of the architecture on larger non-toy datasets, which requires a more robust implementation of the architecture including a much more significant collection of instinctive primitives (i.e., built-in procedures). The use of large language models in order to expedite the engineering of more robust computer simulations of the CCA6 is discussed. The CCA6 is also compared with other cognitive architectures. As well, there is a brief discussion of the architecture suggesting a plausible path for the emergence of language in humans.

The next section is the Conclusion where there is review of the findings of this paper, i.e., demonstrating the emergence of compositionality from a navigation map-based architecture such as the CCA6.

In Appendix A there is a formalization of the main portions of the architecture, including the new work on the CCA6 presented in this paper. The equations in Appendix A form the basis for computer simulations of the architecture.

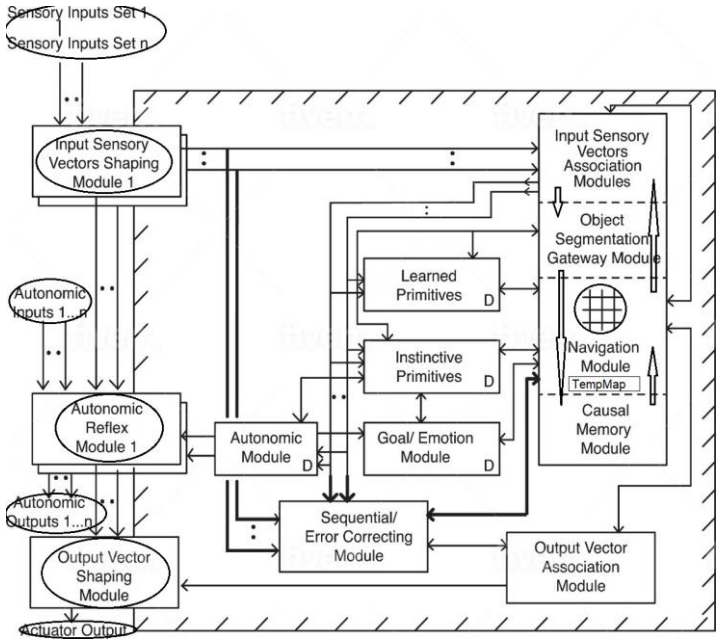

Figure 2. Causal Cognitive Architecture 5 (CCA5). ("D"–changes with development) (Schneider, 2023)

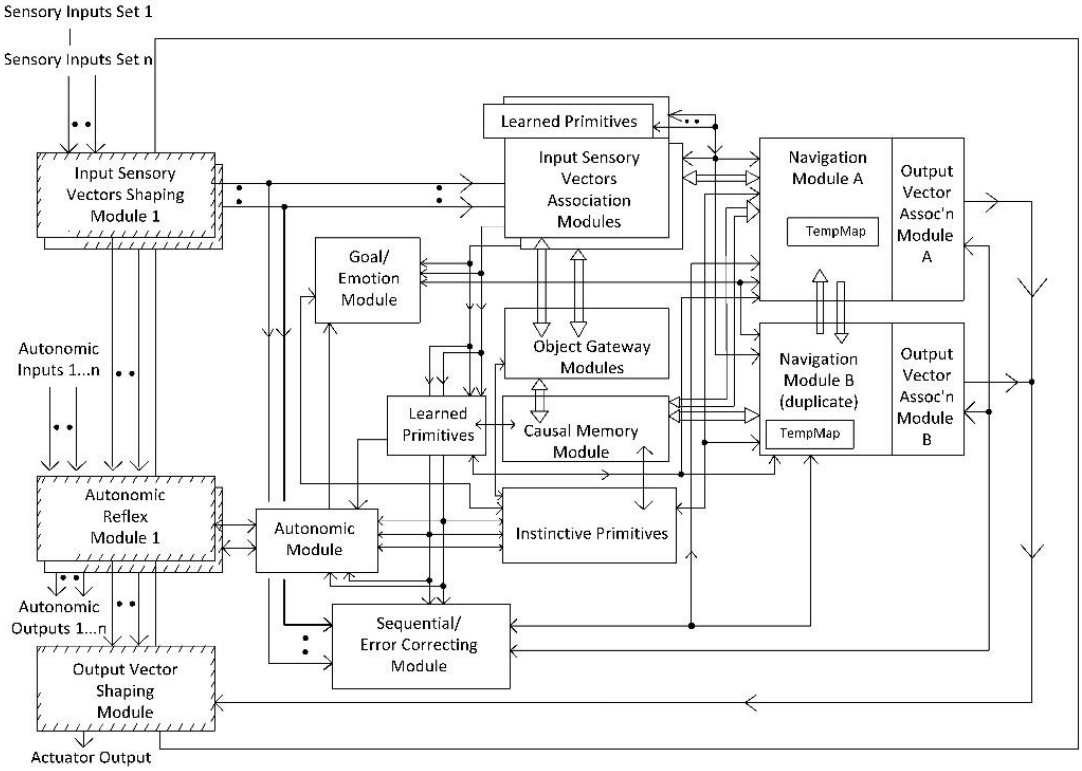

Figure 3. Overview of the Causal Cognitive Architecture 6 (CCA6)

air, link iprimitive {parse_sentence}	air	air	air	air	air
air	air	air	air	air	bleach_ smell
air	air	air	air	air	air
air	air	air	air	air	air
air	air	cylinder, white, link {0023,0,0,0}	air	air	air
triangle, white, {0024,0,0,0}	air	block, black, link {0022,3,3,0}	air	black, block, link {0021,0,0,0}	air

Figure 4. Example of a Navigation Map—the 6x6x0 spatial dimensions are shown containing sensory features, links to other navigation maps, and an instinctive primitive {parse_sentence}. This represents the sensory scene of Figure 1.

2. The Causal Cognitive Architecture 6 (CCA6): Origins and Operation

2.1 Brain-Inspired Changes to the Causal Cognitive Architecture: Duplication of the Navigation Module

The Causal Cognitive Architecture 6 (CCA6) is a brain-inspired cognitive architecture (BICA). In this subsection the inspiration for the changes from the CCA5 architecture to the CCA6 architecture are discussed. As noted above, the Causal Cognitive Architecture does not attempt to replicate the brain at the neuronal spiking level, nor does it attempt to behaviorally replicate the top-level psychological manifestations of the mammalian brain. Rather, it attempts to consider what middle level mechanisms could be occurring in the brain, i.e., at approximately the level of the cortical minicolumns and their postulated interactions with each other and other brain circuits. As noted above, it is largely a functionalist system (Lieto, 2021b), although restrained by various anatomical and biological constraints.

The Navigation Module of the CCA5 architecture (Figure 2) is duplicated in the CCA6 (Figure 3) into Navigation Modules A (the original Navigation Module) and Navigation Module B (the partially duplicated Navigation Module). As noted above, many genetic and developmental mechanisms have been proposed for mechanisms of how brain circuits can duplicate and diverge to new functions (e.g., Rakic, 2009, e.g., Chakraborty and Jarvis, 2015). Navigation Module B (Figure 3) represents the partially duplicated original Navigation Module, lacking some of the connections of the original Navigation Module A, but also developing additional pathways related to language (discussed in more detail below concerning the operation of the architecture). In the new CCA6 architecture presented in this paper, Navigation Module B is used for both reasoning and communication, i.e., language. Although language is not fully implemented in the CCA6, Navigation Module B can perform compositional understanding of language, as will be demonstrated further below in the paper.

This second Navigation Module B is inspired by the ability of human and non-human primate brains to handle communications in a more expressive fashion. The simplistic notion of Wernicke's area as being the specialized area where language is understood may be overly simplistic, and in fact experimental evidence supports an alternative view (Binder, 2015). As well the arcuate fasciculus may more generally connect posterior areas with more general premotor areas, albeit involved in language (Bernal and Ardila, 2009) but that speech comprehension and processing involves many areas of the brain, as is better reflected by a duplicated Navigation Module. From an evolutionary perspective, similarities in the auditory-vocal cortical connectivity in non-human primates and humans (Aboitiz, 2018) would support the duplication of the Navigation Module in non-human primates, i.e., before humans emerged. However, non-human primates would not have been capable of the full causal and analogical reasoning which emerged later and was modeled in the CCA5 (Schneider, 2023).

Broca's area is a critical region in the human brain, more often in the left brain, for producing speech. (Note that none of the Causal Cognitive Architectures attempt to model two hemispheres but are inspired by a more simplistic view of one brain hemisphere.) However, homologues exist in the great apes. For example, Taglialatela and colleagues

(2008) show that in chimpanzees, hand gestures are associated also with activity in a similar region in the chimpanzee left brain.

As noted above, once navigation maps are being used as the main data structure of the Causal Cognitive Architecture, it is a small step to evolve to read through navigation maps and produce a simple proto-language. Such a proto-language can involve gestures as well as speech (Taglialatela et al., 2008; Amici et al., 2022). Assuming this Causal Cognitive Architecture model of brain evolution, proto-language appears to have occurred prior to the existence of humans. Thus, it is assumed in the CCA6 architecture that a partial duplication of the Navigation Module circuitry occurred in primates if not earlier, with further adaptation such as to be advantageous to work in conjunction with the existing Broca's area and other proto-language brain areas.

There are left and right cerebral hemispheres in the mammalian brain. As noted above, the CCA6 currently does not model the function of the two hemispheres. For example, in mammals there is interhemispheric transfer of working memories as objects switch visual hemifields (Brincat et al., 2021). Rather, the CCA6 considers only the left cerebral hemisphere which typically holds the more important regions for language understanding and production.

As discussed above, in order to allow processing of the instructions associated with Figure 1, compositional language processing is required. The main purpose of this paper is to show the ready emergence of compositionality from a navigation map-based architecture such as the CCA6. This readily lends itself to the beginnings of compositional language, i.e., communication abilities only possessed by humans. However, it should be appreciated that while complex language abilities appear to have dramatically arisen evolutionarily quickly in humans, many of the structures required for complex language were already present in the common ancestors. For example, human forkhead box protein P2 (FOXP2)—a protein encoded by the gene *FOXP2* strongly associated with human language production, exists not only in mammals but in other vertebrates. For example, upregulation of versions of this gene occurs in birds during song mimicry and learning (Wu et al., 2021).

2.2 Brain-Inspired Changes to the Causal Cognitive Architecture: Distribution of Learned Primitives

As noted above, the term "primitive" refers to procedures or algorithms that give a sequence of potential actions that can be taken. "Instinctive primitives" are pre-existing, i.e., the architecture is not a tabula rosa but starts with a collection of instinctive primitives. The instinctive primitives are brain inspired by the studies of human infants and nonhuman primates by Spelke and other workers (Spelke, 1994; Kinzler and Spelke, 2007). Although not shown in Figure 3, but discussed in previous versions of the architecture, the instinctive primitives are developmentally sensitive—different portions of the primitives become activated or inhibited depending on the experience level (i.e., age) of the architecture.

"Learned primitives" are not pre-existing, but are learned, as their name suggests. Learned primitives essentially become actions representing algorithms encoded on navigation maps. As such, they can be encoded with recorded features of the environment, and have a spatial origin, as reflected by the navigation map encoding.

Schneider (2023) did not adequately account for the distribution of learned primitives (i.e., learned actions to take) throughout the mammalian brain, as well as the functional differences between instinctive and learned primitives. The complex fashion in which instinctive behaviors are genetically specified (as an example, work by Weber and colleagues (2013) in the instinctive burrowing habits of *Peromyscus* mice) is obviously different than the way millions or billions of byte-equivalents of learned primitives can be accumulated. This is now better reflected in the present paper (i.e., in the CCA6 shown in Figure 3), in keeping with its BICA origins.

In addition, previous versions of the architecture did not adequately account for the evolution of instinctive primitives. In this version of the architecture, new instinctive primitives are not artificially created de novo but are related to previous instinctive primitives. For example, the new instinctive primitives for language processing added to the architecture below, all derive from navigation actions on a navigation map, i.e., processes that already existed within other instinctive primitives.

In the new architecture presented in this paper (Figure 3), learned primitives are physically located within numerous sensory modules, as well as being physically devoted for use by the Navigation Modules and other modules of the architecture. This will allow better future visual, auditory, and other specialized sensory learning, as well as easier operation of the Navigation Modules.

2.3 Causal Cognitive Architecture 6: Compositionality and Compositional Language Emerge

As discussed above, compositionality is vital to an intelligent system, whether artificial or natural. Compositionality obviates the need for infinite experimentation and memorization. Although this paper will be more focused on compositionality in terms of correctly handling objects in the world, as per Figure 1, in fact much of compositionality research in general is related to linguistics (Szabó, 2020).

In an earlier version of the navigation-based BICA Causal Cognitive Architecture it was noted that the basic data structure of the architecture—the spatial navigation maps—resulted in the emergence of explainability and the corresponding potential for language (Schneider, 2022b). The series of navigation maps the Causal Cognitive Architecture produced and stored, effectively provided an explanation for decisions. A very simple procedure (termed "primitive" in the nomenclature of the architecture) could be used to read back and to communicate these navigation maps to other agents. A simple primitive (i.e., procedure) termed `navigation_map_to_proto_language()` was discussed previously but no other attempt to comprehend or produce compositional language was made. In the CCA6, it will be shown below how with the duplicated Navigation Module and a few additional instinctive primitives involved in language processing, that compositional language comprehension readily emerges.

Given the ease in which compositional language emerges in the model and given how the changes in the architecture (i.e., duplication of the Navigation Modules) could have occurred in a biological model with relatively few genetic changes, this suggests a plausible mechanism in how language could have emerged in humans from a chimpanzee-human last common ancestor. This is briefly considered in the Discussion towards the end of the paper.

2.4 Overview of the Operations of the Causal Cognitive Architecture 6: The Navigation Map

In Appendix A there is a more formal representation of the main portions of the architecture, including the new work on the CCA6 presented in this paper. The equations in Appendix A form the basis for computer simulations of the architecture.

The main brain-inspired basis of the architecture is that millions of cortical minicolumns in the mammalian neocortex are modeled in the CCA6 as millions of spatial navigation maps. The navigation map is used as the main data structure throughout the architecture. Sensory data are encoded on navigation maps. Internal computations are largely encoded on navigation maps.

Primitives refer to procedures or algorithms that give a sequence of potential actions that can be taken. Instinctive primitives are pre-existing. Learned primitives are not pre-existing but are learned. Both instinctive primitives and learned primitives are encoded on navigation maps. As noted above, the biological equivalent of instinctive primitives may be stored very differently in the brain than the way the biological equivalent of learned primitives are stored. However, for simplification of the architecture, both are encoded similarly on navigation maps, although as Figure 3 shows, the learned primitives are more broadly distributed in the architecture.

An example of a navigation map is shown in Figure 4. This "navigation map" in Figure 4 actually represents the cubes, cylinder, and triangle of Figure 1. Navigation maps can hold links to other cells within the same navigation map or in a different navigation map. For example, in cell (0,0,0) of the navigation map in Figure 4, there is a link to cell (0,0,0) in navigation map #0024. Various core algorithms of the architecture as well as additional instinctive primitives can follow these links and retrieve and select navigation maps they link to.

2.5 Overview of the Operations of the Causal Cognitive Architecture 6: Cognitive Cycles

"Cognitive cycles" are cycles of sensory inputs being processed and producing an output or a null output. Consider the CCA6 architecture in Figure 5. Each cognitive cycle, sensory detectors stream in sensory inputs into the Input Sensory Vectors Shaping Modules (arrow A). As indicated in Figure 5, there is one such shaping module for each sensory system, e.g., vision, auditory, and so on, which normalizes sensory inputs for further processing within the architecture. In the Input Sensory Vectors Association Modules (arrow B) sensory features are spatially mapped onto navigation maps dedicated to one sensory system. Note again from Figure 5 that there is one association module for

each sensory system, e.g., vision, auditory, and so on. The Object Gateway Modules (arrow C) work with each Input Sensory Vectors Association Module—they attempt to recognize what are considered separate objects and figuratively pull them out of the sensory scene.

Inputs of a given sensory system (e.g., vision) are matched against the best matching stored navigation map in that (e.g., visual) Input Sensory Vectors Association Module (Figure 5). That best matching stored navigation map is retrieved and then updated with the actual sensory input of that sensory system (or a new navigation map is created if there are too many changes). As such, there is a type of predictive coding occurring—the architecture anticipates what it is sensing and then considers the differences with the actual input signal (Schneider, 2023). This works well for the perception of noisy, imperfect sensory inputs.

The result at this step is that visual, auditory, and other sensory features of each segmented object, as well as the overall scenes, are spatially mapped onto navigation maps dedicated to one sensory system. These newly created or updated single-sensory navigation maps are then mapped onto a best matching multi-sensory navigation map taken from the Causal Memory Module (arrow D). The best matched navigation map from the Causal Memory Module is then considered moved to the Navigation Module A (arrow E). This best matching navigation map is updated with the actual sensory information sensed from the environment. The updated navigation map is termed the Working Navigation Map (represented by circle G in Figure 5). These operations are similar to the previous architecture CCA5 (Figure 2) where they have been formalized (Schneider, 2023). A small (but important for compositionality functioning) difference is that the architecture now distinguishes between Navigation Modules A and B. The operations occurring in the CCA5 largely now occur in Navigation Module A of the newer architecture presented here, the CCA6 (Figure 5). As will be shown below, in the newer architecture, Navigation Module B tends to be used with cognitive cycles compositionally processing language-related sensory inputs or intermediate results.

Instinctive and learned primitives, which as noted above are essentially small procedures that can perform operations on the current or linked navigation maps, are then selected by a similar matching process in terms of what are the sensory inputs as well as in terms of signals from the Goal/Emotion Module (Figure 5) and from the previous values of the Navigation Modules (Figure 5). Arrow F shows the actions of the best matching instinctive primitive on the Navigation Module A in Figure 5. Circle H represents the best matching instinctive primitive in Figure 5.

In Navigation Module A in Figure 5, there is the Working Navigation Map represented by circle G, and the best matching instinctive primitive represented by circle H. The best matching instinctive primitive (in other cases it could instead be a best matching learned primitive) then operates on the Working Navigation Map. The result is an output signal to the Output Vector Association Module A (arrow I) and then to the Output Vector Shaping Module (arrow J). The latter then causes activation of an actuator or an electronic signal transmission (arrow K). Again, this is largely similar to what was presented and formalized by Schneider (2023).

The navigation map in Figure 4 can be used as an example of a Working Navigation Map represented by circle G in Figure 5. A best matching instinctive primitive (in other cases it could instead be a best matching learned primitive) is represented by circle H in Figure 5. This best matching instinctive primitive does not have to be the instinctive primitive linked to by cell (0,5,0) in the navigation map of Figure 4 (origin at the lower left corner) but can be any instinctive primitive in the Instinctive Primitives module of Figure 5 (or in other cases, can be any learned primitive in the Learned Primitives module of Figure 5). As an example, the action of the best matching instinctive primitive (circle H) on the Working Navigation Map (circle G, representing the navigation map in Figure 4) could be an action to move the triangle of cell (0,0,0) of that map to another cell in the navigation map. Generally, in keeping with previous versions of the architecture, primitives act on the Working Navigation Map in Module A. However, in the CCA6 as seen in the sections below, newer instinctive primitives are introduced which allow compositional understanding of language and compositional actions.

Compositional operations tend to occur in Navigation Module B, i.e., instinctive primitives associated with language and compositionality tend to operate on Navigation Module B (shown by the dashed arrow L in Figure 5). As well learned primitives associated with language and compositional operations tend to operate on Navigation Module B (shown by the dashed arrow M in Figure 5). Although not fully explored in this paper, Navigation Module A and Navigation Module B can work together in more abstract fashions, each holding different working navigation maps relating to different aspects of the environment or of a problem being solved.

Note that these operations in the Navigation Modules in Figure 5 (i.e., the operations of the best matching primitive on the navigation map in a Navigation Module) are operations which can be performed on or between arrays, which is what essentially navigation maps are. Examples of such operations include, for example, comparing arrays, adding or subtracting a vector to an array, and other simple operations that brain-inspired circuitry could reasonably carry out. This causes the Navigation Module(s) to produce and send a signal to their Output Vector Association Module(s)

and then to the external embodiment via the Output Vector Shaping Module. Then the cognitive cycle repeats—sensory inputs stream in, are processed, and there may be some output action, and then the cycle repeats again, and so on.

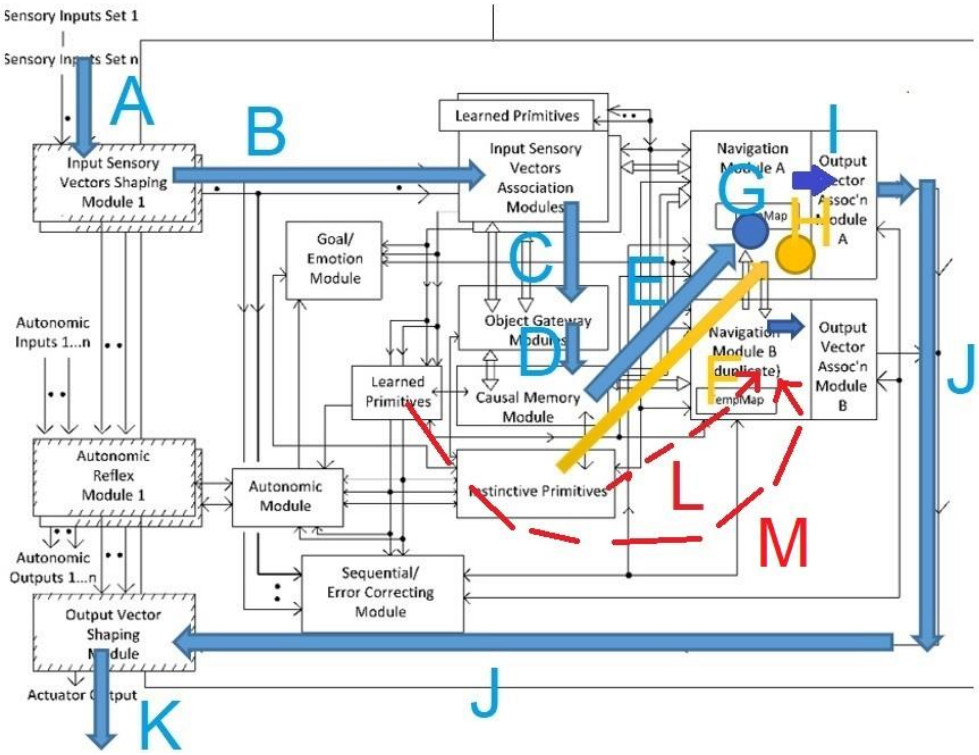

Figure 5. A Cognitive Cycle in the CCA6 Architecture. Sensory inputs are processed by the various modules (described in text). The operation of the best matched instinctive primitive (circle H) on the Working Navigation Map (circle G) produces an output signal to the Output Vector Association Module A (arrow I) and then to the Output Vector Shaping Module (arrow J) and then an output to the external world which can activate an actuator or transmit an electronic signal. Then another cognitive cycle starts.

Arrow A – sensory inputs into the Input Sensory Vectors Shaping Modules
Arrow B – to the Input Sensory Vectors Association Modules
Arrow C – to the Object Gateway Modules
Arrow D – to the Causal Memory Module
Arrow E – best matched navigation map ("Working Navigation Map") to the Navigation Module A
Arrow F – best matched instinctive primitive to the Navigation Module A
Circle G – Working Navigation Map
Circle H – best matched instinctive primitive
Arrow I – to the Output Vector Association Module A
Arrow J – to the Output Vector Shaping Module
Arrow K – outputs to the external world (actuator, electronic communication signal))
Arrow L – instinctive primitives relating to compositionality and language tend to be applied to Navigation Module B
Arrow M – learned primitives relating to compositionality and language tend to be applied to Navigation Module B

2.6 Overview of the Operations of the Causal Cognitive Architecture 6: Feedback Operations

In some, or many, cognitive cycles, there actually is no output signal produced by the Navigation Module(s). For example, the environment may be quiet and inactive. In this case the best matching and processed Working Navigation Map and best matching activated instinctive/learned primitive operating on the Working Navigation Map, simply do not produce an actionable output from the Navigation Modules A or B. Thus, there is no signal from Navigation Modules' Output Vector Association Modules A or B to the Output Vector Shaping Module. However, this can often occur in other non-quiet, i.e., active, situations. It may happen that the best-matched primitive operating on the Working Navigation Map does not produce an actionable output. This may occur for another cognitive cycle or two (or more), and then a sensory input may change, and some sort of actionable output finally occurs.

Feedback pathways are advantageous and commonplace throughout the Causal Cognitive Architecture—the current state of a downstream module can affect the recognition or processing of sensory inputs or data in more upstream modules. Schneider (2021) enhances the feedback pathways from the architecture's sole Navigation Module to the Input Sensory Vectors Association Modules (similar to Figure 2). In cognitive cycles where no actionable output occurs from the Navigation Module, its output can be treated as intermediate results, and fed back more fully and stored in the Input Sensory Vectors Association Modules. In the next cognitive cycle these essentially intermediate results being stored in the Input Sensory Vectors Association Modules will automatically be treated as the processed sensory inputs and processed again in the Navigation Module. Sometimes these intermediate results will be operated on by a different primitive, or sometimes the changes in the intermediate results allow an actionable result to occur this time. By allowing the architecture to feed back and then re-operate on intermediate results the architecture can explore possible cause and effects of actions.

Schneider (2022a) gives an example where a robot controlled by a Causal Cognitive Architecture (similar to Figure 2) working as a patient aide encounters a patient who lets go of a walker and starts to lose balance. The robot has not been preprogrammed what to do in each situation, but the intermediate results of the patient falling are re-operated on with the triggering of a goal/emotion signal which triggers a learned primitive that the patient should not fall. This in turn triggers an instinctive primitive that there should be a push back against something falling or moving in order to stop that movement. Thus, the robot pushes back on the patient's body in the direction of the fall.

Even with re-operating on intermediate results of the Navigation Module, the processed intermediate results nor a different primitive being triggered by the processed intermediate results, often still did not produce a causal or any actionable output. However, experimentation showed that a further slight modification to the feedback circuits, allowed analogical reasoning to emerge (Schneider, 2023). Equations (i) to (v) below represent a pseudocode description of this inductive analogical mechanism, re-written for the CCA6 architecture. (These equations are taken from the more complete formalization of the architecture in Appendix A where they are listed as equations (95) to (99).)

If no actionable result occurs in the Navigation Module, then:
⇒as before, feed back and store the Working Navigation Map values in the Sensory Association Modules **(i)**
⇒propagate the Working Navigation Map to the Causal Memory Module, match it with the best map, make this the new Working Navigation Map **(ii)**
⇒see which navigation map(s) this map recently linked to, store the linked navigation map(s) in the TempMap temporary memory area of the Navigation Module(s) (**iii)**
⇒subtract the value of TempMap from the Working Navigation Map and store the result in the Navigation Module being used (**iv)**
⇒in the next cognitive cycle, add the intermediate results being stored in the sensory circuits to the Navigation Module being used, with the result forming the new Working Navigation Map **(v)**

A definition of induction by analogy can essentially be given by equations (vi) to (ix). Variable **a** has features or properties F_1, F_2, F_3, F_4, … F_n (vi). Variable **b** also has the same features F_1, F_2, F_3, F_4, … F_n (vii). It is now seen that variable **b** has another property G (viii). Therefore in (ix) it can be concluded by induction by analogy that variable **a** also has this other property G.

$$F_1\mathbf{a} \;\&\; F_2\mathbf{a} \;\&\; \ldots\; F_n\mathbf{a} \quad \textbf{(vi)}$$
$$F_1\mathbf{b} \;\&\; F_2\mathbf{b} \;\&\; \ldots\; F_n\mathbf{b} \quad \textbf{(vii)}$$
$$G\mathbf{b} \quad \textbf{(viii)}$$
$$\therefore G\mathbf{a} \quad \textbf{(ix)}$$

□

In equation (i) above the Working Navigation Map can be referred to as variable **a**, or perhaps as navigation map **a**. There is a need in the operation of the architecture to want to know what this navigation map **a** will do next, i.e., which navigation map will it call. Consider variable **b**, or perhaps named as navigation map **b**, as referring to this related but different Working Navigation Map in (ii). It is the best-matching navigation map to navigation map **a** and thus it is reasonable to expect that it will share many features with navigation map **a**. There is now a need to determine what navigation map **b** does next (i.e., what navigation map does it link to). It can be seen that navigation map **b** links to the navigation map which is then stored in the TempMap temporary memory area of the Navigation Module being used in (iii).

The difference between navigation map **b** and TempMap is then stored in the Navigation Module being used (i.e., essentially forming a new Working Navigation Map for that moment) as shown in (iv). This value in (iv) is the new feature G corresponding to equation (viii). By induction by analogy, since navigation map **b** has feature G, therefore navigation map **a** also has feature G (ix). Thus, in (v) feature G is added, which is actually the difference which is currently stored in the Navigation Module being used, to the navigation map **a**, which is actually the original Working Navigation Map stored in the sensory circuits. The resultant value in the Navigation Module being used, i.e., the new Working Navigation Map produced, represents the original Working Navigation Map plus new feature G (v, ix).

Despite these improvements made to the architecture, resulting in the CCA5 architecture (Schneider, 2023), it still was not able to easily handle compositional tasks such compositional language processing required to follow instructions such as in Figure 1. In the sections below it is shown how the new work, which is termed the CCA6 version of the architecture, and the new instinctive primitives (like the architecture, also brain inspired), now readily handle compositional tasks.

3. Compositional Language Understanding

3.1 Compositionality Demonstration Problem

Figure 1 above illustrates the instruction "place the white triangle on top of the black block which is not near a white cylinder." To successfully follow this instruction requires an ability to extract the meaning from a non-simple expression in terms of its constituent parts and their relations.

To solve the compositionality problem of Figure 1 the architecture must be capable of a number of tasks. The first is the ability to understand the language of the instruction which itself is compositional in nature: "place the white triangle on top of the black block which is not near a white cylinder." This requires decomposing the instruction into its relevant parts. The architecture (Figure 3) which includes additional newer instinctive primitives (described below) is now able to do this, as demonstrated below. Then the meaning of each decomposed part of the instruction must be understood. The newer CCA6 architecture is able to do this, as shown below. Note that these compositional abilities are now essentially core mechanisms of the architecture, particularly associated with Navigation Module B—it performs these operations in a simple, mechanical fashion.

3.2 New Instinctive Primitives included as Part of the CCA6

As shown in Table 1, the CCA6 now includes a collection of other instinctive primitives which allow it to understand and react (with actions) to compositional language and allow it to solve compositional tasks such as Figure 1. The function of these instinctive primitives will be demonstrated in the examples below. (Full names and fuller description of the properties of these instinctive primitives are listed in Appendix A.)

As noted above, new instinctive primitives are not artificially created de novo but are related to previous instinctive primitives that exist within the architecture. The new language-related instinctive primitives shown in Table 1 all derive from navigation actions on a navigation map, i.e., processes that already existed within other instinctive primitives.

With regard to the generation of language, the instinctive primitive `apply_primitive_nav_to_protolang()` which originated in a previous version of the architecture remains. This instinctive primitive simply reads back a navigation map, largely verbatim. It is not compositional, but simply reads back navigation maps

and can combine navigation maps, somewhat akin to the combinatoriality language abilities in non-human primates (e.g., Zuberbühler, 2020). Future versions of the architecture are expected to include primitives for the generation of compositional language. However, at present, the instinctive primitives in Table 1 are sufficient for allowing a solution to the compositionality problem of Figure 1, as demonstrated further below.

Instinctive Primitive	**Description**
`apply_primitive_nav_to_protolang()`	combinatorial, non-compositional readback of a navigation map (present also in the CCA5 version of the architecture)
`parse_sentence()`	parses sentence onto an existing navigation map (new part of the CCA6)
`physics_near_object()`	marks objects with a "near" or a "not" that are near or not near a specified object (new part of the CCA6)
`place_object()`	places the specified object somewhere in the navigation map (new part of the CCA6)

Table 1. Language-related Instinctive Primitives in the Causal Cognitive Architecture 6 (CCA6) (Full names and properties of these primitives are listed in Appendix A.)

3.3 Approaching the Compositionality Problem of Figure 1

The navigation map shown in Figure 4, represents a far view (as opposed to a close-up view) of the sensory scene of Figure 1, i.e., where and what the visual objects are, where and what odors may be, where and what sounds are interpreted to be, and so on. This far view sensory scene is constructed from a number of close-up views, reflected by the links to individual objects. For example, in cell (0,0,0) there is a link to navigation map #0024 cell (0,0,0) which contains the close-up view of the triangle object. The binding of objects in sensory scenes has been discussed previously in earlier versions of the architecture. Also, although not relevant in this simple static example, many navigation maps may contain motion prediction vectors showing the current and expected motion of an object, as discussed in earlier versions of the architecture.

Please note that the origin (0,0,0) of the navigation map's coordinate (x, y, z) system is considered to be at the lower left corner of the navigation map. Also, please note that in this paper we are considering everything in two dimensions, i.e., the z dimension is left at zero.

The labels in Figure 4 were generated by the identification of the objects. They are taken from a pre-programmed catalog and represent little language significance to the architecture at this point. Consider, for example, the cell of the navigation map of Figure 4 which has the feature "cylinder, white". In this navigation map's cell with coordinates (2,1,0) there is also a link to {0023,0,0,0} which represents cell (0,0,0) in navigation map #0023. The cells starting at (0,0,0) in that map contain more detailed visual processed sensory data about a close-up view of the object "cylinder, white." This label "cylinder, white" would have been generated in the previous navigation map as the CCA6 attempted to match that navigation map of the close up visual sensory inputs of the white cylinder of Figure 1. The term "cylinder" is simply matched to a catalog of shapes which the visual image was matched to. Again, note that there is little understanding of language here. However, there could be in the future as language-related navigation maps build up, and additional language-related instinctive primitives are added to the architecture.

Thus, the Navigation Module is able to follow links in the navigation map of Figure 4 if so instructed by an instinctive or learned primitive, i.e., it can effectively zoom into a closer description of the white cylinder, and so on. For example, if an instinctive or learned primitive instructed going to the link in the cell of {0023, 0, 0, 0}, then a navigation map representing a close-up view of the white cylinder would become the new Working Navigation Map.

In the introduction to this paper, it was noted that the principle of compositionality is that a non-simple expression is function of the meanings of its constituent parts. Indeed, on a feature level, it is seen that navigation maps can easily be composed and decomposed (i.e., linked) to constituent navigation maps.

As noted above, the CCA6 at this point does not understand the English (or other) language at any level of sophistication. The label "cylinder, white" is just that to it—a feature label it can match from a catalog of shapes and then from a catalog of colors. However, the architecture does perform such operations with compositionality. Different features such as shape and color can easily be combined. Navigation maps can effectively zoom in and zoom out to compose and decompose objects.

Similarly, the instruction "place the white triangle on top of the black block which is not near a white cylinder" is handled quite mechanically, but again in a compositional manner, by the Causal Cognitive Architecture 6 (CCA6). In the next section the operations of the CCA6 in understanding and following the instruction associated with the example of Figure 1 will be demonstrated.

4. A Demonstration of Compositional Language Understanding

4.1 Demonstration of the Emergence of Compositional Language Comprehension and Behavior in the Architecture

In this section the demonstration compositional example of Figure 1 will be processed by a simulation of the CCA6 architecture. The section describes the setup of the problem and then walks through the internal actions taken by the architecture. This explanation of the steps taken by the architecture is intended as much to demonstrate to the reader the operations of the architecture as it is to show that the abilities for compositional language comprehension and behavior, albeit at the level of a toy problem, readily emerge from the CCA6 architecture.

At this point in its development the CCA6 architecture is not at the stage to consider operations on a dataset of thousands of non-toy compositional examples. This is discussed in more detail in a section below discussing further work on the architecture. In particular, larger collections of instinctive primitives are required, and better usage and implementation of the learned primitive system is required. As well, in order for the architecture to simply solve a compositional problem on its own agency, it first needs to have acquired the necessary navigation maps (a sort of education, so to speak) to even understand what is meant by solving a compositional problem. Thus, at this stage, the purpose of this paper is to show that with small biologically evolutionarily plausible changes from the previous architecture, compositional language comprehension and behavior can emerge. This is done in this section with the demonstration example of Figure 1. Then after doing so, further generalization of the example is considered. The example of Figure 1 is negated (i.e., it becomes "place the white triangle on top of the black block which is ~~**not**~~ near a white cylinder"). A walk through the steps of the simulation is given again for this example. At the end of the section, the behavior of the CCA6 in understanding and solving the problem in Figure 1 is compared with the performance of a state of the art (at the time of writing, of course) large language model in understanding and solving the exact same problem.

In Appendix A there is a more formal representation of the main portions of the architecture, including the new compositional work on the CCA6 presented in this paper. The equations in Appendix A form the basis for simple computer simulations of the architecture.

4.2 Python Computer Simulation of the CCA6 Architecture

A Python-language (van Rossum and Drake, 2014) computer simulation of the CCA6 Architecture is performed. The purpose of the simulation at this point is largely conceptual, i.e., to show that the operation of the architecture is feasible and is consistent. The simulation is largely based on the more formal description and equations of the CCA6 architecture listed in Appendix A.

The navigation maps in the simulation are 6x6x0 spatial dimensions. A larger number of dimensions is actually used to store additional information such as the segmentation of a feature (i.e., to what object in a sensory scene does it belong to), and the association of a feature with motion of an object. Given that navigation maps are essentially array-like, they are treated as Numpy-library (Harris et al., 2020) array objects (x). The data structure of simulated navigation maps is given by the lines of Python code shown in (xi) and (xii). Each cell in an array is specified by navigation map number 'n', and spatial coordinates (x,y,z). Each cell has an associated 's' dimension, i.e., its segmentation, to what object in a sensory scene does it belong to. Similarly, each cell has an associated 'a' value, i.e.,

its association value, generally to what motion in a sensory scene does it belong to. (The arrays are actually formatted for 6x6x6 spatial dimensions, as (xi) shows, but were only used as 6x6x0 arrays in the simulation.)

```
import numpy as np    (x)
self.gb = np.empty((self.maps, 6, 6, 6, self.segs, self.assocns), dtype=object)    (xi)
# gb[n,x,y,z,s,a]    (xii)
```

For the compositionality examples such as Figure 1, examples were encoded and entered into the visual perception system as simulated-like inputs, with largely null inputs for the auditory and olfactory systems. At present the CCA6 is not attached to real-world sensors, and all inputs are simulated. Similarly, all outputs are simulated as well.

Although the CCA6 has the start of a compositional proto-language, it does not yet contain the necessary learned primitives to understand the concept of being asked to solve a problem. Rather, the instinctive primitives, including those now in Table 1, are mechanically applied to the simulated sensory inputs. Future versions of the CCA6, or at least future simulations of this version of the architecture, are expected to have additional instinctive primitives to allow simulations more agency in day-to-day actions and recognizing problems to solve.

The CCA6 can output the navigation maps it creates, which then are typeset for purposes of this paper. For example, the actual visual sensory system Local Navigation Map related to Figure 1 is shown in Figure 6. Note that it does not contain auditory or olfactory information as it is the navigation map of the visual sensory system. (The blank (i.e., null) cells of the computer simulation are replaced for aesthetic reasons in the typeset navigation maps by "air." For the moment this is fine, but in future more complex environments this will be changed.)

The Local Navigation Maps are created from their respective input sensory streams. Local Navigation Maps are created for the visual sensory system, for the auditory sensory system, for the olfactory sensory system, and for any other sensory systems the implementation may be using. These are then combined into a multi-sensory navigation map which is matched against the other stored multi-sensory navigation maps from the Causal Memory module (Figure 3). The best matching navigation map is then updated with actual sensory inputs from this combined map, and then forms the Working Navigation Map. This is the navigation map which is sent to Navigation Module A (normal processing) or Navigation Module B (compositional-related associated inputs and primitives).

The simulation follows the description of the architecture given above. It is better formalized in Appendix A. Due to the small size and the small number of navigation maps in the simulation, parallel operations are able to be simulated by iterative processing (i.e., one sensory system after another, and one navigation map after another, rather than in parallel as would occur in a hardware-based system). Often navigation maps need to be compared with other navigation maps. For example, a local navigation map (e.g., visual system or auditory system, and so on) needs to be compared with the best matching stored visual sensory system navigation map (or auditory system, and so on, depending on the local sensory system). For example, the input sensory data needs to trigger a best matching instinctive primitives navigation map and a best matching learned primitives navigation map. While in the future, more typical deep learning components can be used, at present a fuzzy string-matching library is used (FuzzyWuzzy from pypi.org, fuzzy string matching via Levenshtein distances between sequences).

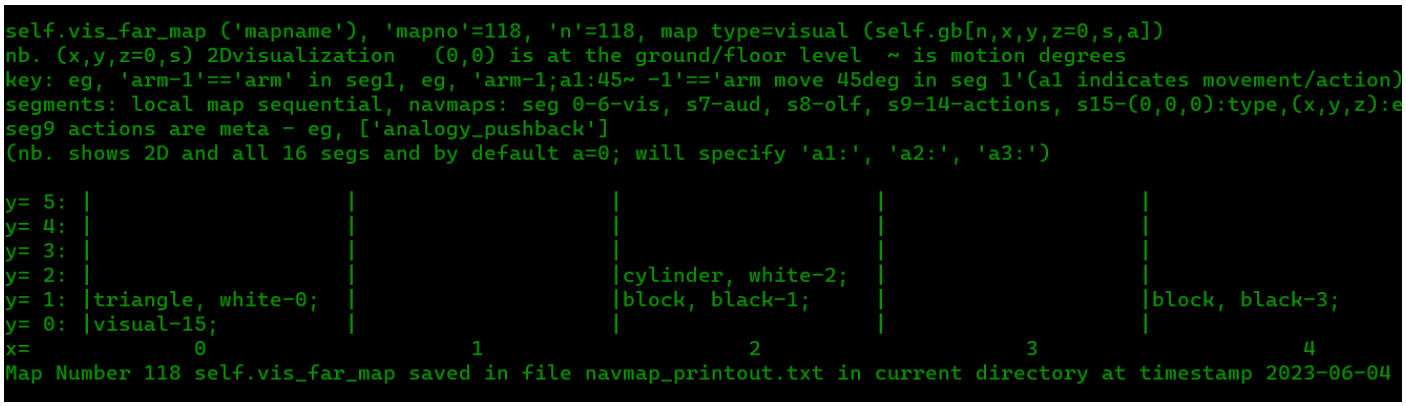

Figure 6. Actual Local (Visual) Navigation Map representing the visual sensory inputs of the sensory scene of Figure 1.

4.3 Solving the Compositionality Problem of Figure 1: Initial Cognitive Cycles

Consider a robot controlled by a CCA6 architecture. Since no such robot has been built, a Python simulation of system is considered. Rather than call the system "computer simulation of robot controlled by a CCA6 architecture," for simplicity the term "CCA6" is used here. Thus, when discussing the operation of the architecture, the term "CCA6" refers to a simulation of the CCA6 architecture plus its robotic embodiment. (The effect of the embodiment is beyond the scope of this paper. Rather the embodiment should simply be considered as capable of moving itself, moving objects much as humans can, and sensing the environment depending on what sensory systems are simulated.)

The CCA6 encounters Figure 1 and the instruction associated with it: "place the white triangle on top of the black block which is not near a white cylinder."

The first thing the CCA6 does is to bind the sensory inputs of the sensory scene. (A "sensory scene" is a scene but refers to all the senses available, e.g., visual inputs, auditory inputs, olfactory inputs, and so on). As noted above, the CCA6 will inspect each object in the sensory scene and make a close-up navigation map of the object. A far visual view is then constructed (i.e., the individual objects are added to another navigation map). A far visual navigation map of the visual sensory scene is shown, for example, in Figure 6. The Working Navigation Map is the fused navigation map of all the local navigation maps (i.e., the different navigation maps constructed by the visual inputs, the auditory inputs, the olfactory inputs, and so on). In the case of this example (which involves only visual sensory stimuli) it is shown in Figure 7 in Navigation Module A.

The CCA6 then processes the language instruction associated with Figure 1. At this point in the development of the architecture and its computer simulation, it is assumed that either an auditory or visual instruction has simply been matched to a catalog of sounds or letters/words (ignoring the reality that perception of instructions actually occurs at a much more sophisticated level in humans). The instinctive primitive `parse_sentence()` (Table 1) is applied to the words associated with Figure 1 (i.e., the instruction to place the white triangle) and operates on the sentence in a very mechanical fashion.

The primitive `parse_sentence()` will copy the instruction (or parts of it) as a navigation map to Navigation Module B (Figure 7). Thus, the instruction, "place the white triangle on top of the black block which is not near a white cylinder" is now the Working Navigation Map in Navigation Module B. Figure 7 shows the contents of Navigation Module A and Module B at this point in time in the simulation.

The navigation map in Navigation Module A differs somewhat from the navigation map shown in Figure 4. In cell (0,5,0) there is no link to the instinctive primitive `{parse_sentence}` as this primitive was located elsewhere in the actual simulation. As well, in cell (5,4,0) there is no bleach_smell sensory feature as no olfactory inputs were used in the actual simulation.

The navigation map in Navigation Module B contains links `{+}`. These just mean to link to the next cell in the navigation map. The x coordinates are incremented until the end of the row and then the y coordinate is incremented, and then another row is linked together, and so on.

The initial cognitive cycles allowed the sensory information from Figure 1 to be processed and bound to the navigation maps shown in Figure 7. In the next set of cognitive cycles described in the next section, the sensory inputs are ignored (i.e., activity in arrow B of Figure 7 is ignored), and instead there is processing and feedback/temporary storage of intermediate results of the navigation maps each cognitive cycle. This occurs each cognitive cycle until the sequence of instinctive primitives which were called terminate, and an output action occurs, and then sensory inputs are not ignored but processed again.

4.4 Solving the Compositionality Problem of Figure 1: Operations on Navigation Modules A and B

At this point, Figure 7 represents the navigation maps in Navigation Modules A and B. The `parse_sentence()` primitive has already been activated, as noted above (and mapped the instruction associated with Figure 1 into the navigation map which was moved to Navigation Module B). This primitive then progresses through a sentence much like other primitives progress through a path until the primitive times out or is overridden by another instinctive or learned primitive.

Whatever word the `parse_sentence()` primitive encounters, it matches against the navigation maps stored in the Causal Memory Module associated with Navigation Module B. A word or section of a word can be matched with one or two cognitive cycles of operations.

The first word encountered “place” (cell (0,0,0) of the navigation map in Navigation Module B of Figure 7) is matched against words in the Causal Memory Module, and recognized as an action. The action will be applied to the next words after it, until the primitive comes to another action word. The next words not considered action words are “the”, “white” and “triangle”. These words are found in cell (0,0,0) in the navigation map in Navigation Module A of Figure 7. Thus, as shown in Figure 8, the Navigation Module has written <”place”> in this cell (0,0,0) containing the feature white, triangle. (Note that these words in the Causal Memory Module have been pre-programmed from a

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	cylinder,white, link{0023,0,0,0}	air	air	air
triangle, white, link{0024,0,0,0}	air	block, black, link {0022,3,3,0}	air	black, block, link{0021,0,0,0}	air

Navigation Module A

“not”, link{+}	“near”, link{+}	“a”, link{+}	“white”, link{+}	“cylinder”, link{+}	
“of”, link{+}	“the”, link{+}	“black”, link{+}	“block” link {+}	“which”, link {+}	“is”, link {+}
“place”, link{+}	“the”, link{+}	“white”, link{+}	“triangle”, link{+}	“on”, link{+}	“top”, link{+}

Navigation Module B

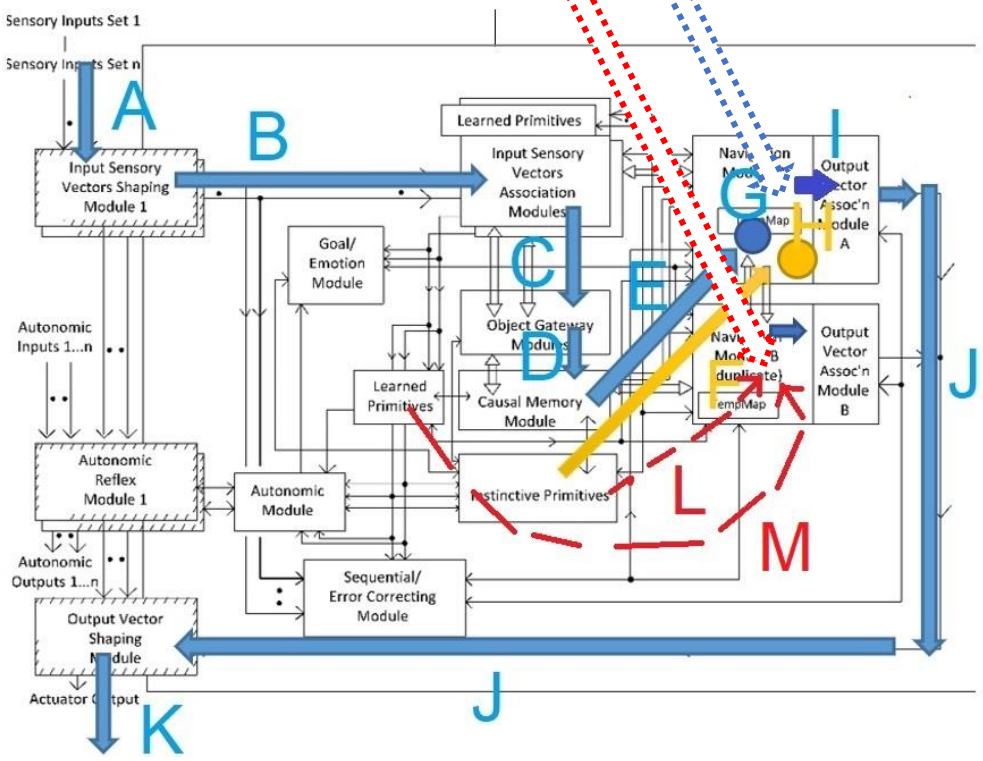

Figure 7. A navigation map representing a far view of Figure 1 is in Navigation Module A. Another navigation map representing the instruction associated with Figure 1 ("place the white triangle on top of the black block which is not near a white cylinder") is in Navigation Module B. Cognitive cycles occur and 'mechanically' process the contents of the Navigation Modules via feedback operations which ignore new real-world sensory inputs for the moment (i.e., arrow B has no/ignored activity during these internal operations). (`link{+}`– link to the next cell. Arrow/circle letters from Figure 5.) See text for full description.

catalog in this simple simulation but can be learned via instinctive and learned primitive learning in more advanced implementations and simulations.)

The next word that the primitive encounters in the sentence is "on" which at present does not match in the Causal Memory Module, i.e., it will be ignored. However, the next the word "top" (Figure 7, the navigation map in Module B) matches as an action word and will be applied to the next words "of", "the", "black" and "block". In the navigation map in Module A (e.g., Figure 7) cells (2,0,0) and (4,0,0) both contain "black" and "block". Thus, as shown in Figure 9, the Navigation Module in response to the instinctive primitive has written <"top"> in cells (2,0,0) and (4,0,0), which contain the words "black" and "block". The next words "which" and "is" are ignored as the primitive cannot match them. The Navigation Modules in conjunction with the instinctive primitives (or learned primitives in other operations) are very mechanically operating on the words or labels in the Navigation Modules, and continue to parse the instruction sentence.

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	cylinder,white, link {0023,0,0,0}	air	air	air
<"place"> triangle, white, link {0024,0,0,0}	air	block, black, link {0022,3,3,0}	air	black, block, link {0021,0,0,0}	air

Figure 8. First word of the sentence of Figure 1 being parsed by the instinctive primitive `parse_sentence()` with application to the navigation map of Navigation Module A (Figure 7) representing the sensory scene of Figure 1. The primitive has written the action word "place" in the cell containing the features "white, triangle." (Navigation Module A)

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	cylinder,white, link {0023,0,0,0}	air	air	air
<"place">, triangle, white, link	air	<"top">,block, black, link {0022,3,3,0}	air	<"top">,black , block, link {0021,0,0,0}	air

{0024,0,0,0}					

Figure 9. The instinctive primitive `parse_sentence()` has now written the word "top" in the cells containing the features "black, block." (Navigation Module A)

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	<"not">, cylinder,white, link {0023,0,0,0}	air	air	air
<"place">, triangle, white, link {0024,0,0,0}	air	<"not">, <"top">,block, black, link {0022,3,3,0}	air	<"top">, black,block, link {0021,0,0,0}	air

Figure. 10. The instinctive primitive `physics_near_object()` has now written "not" in the cells containing or adjoining cells containing the features "cylinder, white." (Navigation Module A)

It should be noted that the Navigation Module will manipulate similarly whatever words or labels such as "black", "block", and so on, which are in the cells, but that these cells are in fact grounded with links to other navigation maps providing more details and/or context about these labels.

The next word in the instruction of Figure 1 is "not" which is considered an action word when matched against the navigation maps in the Causal Memory Module. It is applied to the next word "near" which actually triggers another instinctive primitive – `physics_near_object()`.

The `physics_near_object()` instinctive primitive will mark cells in a navigation map which are "near" or if so specified "not" near a specified object. It continues reading through sentence and encounters the words "a", "white" and "cylinder" before reaching the end of the sentence (Figure 7, Navigation Module B). It matches these words to cell (2,1,0) in the navigation map in Navigation Module A. The `physics_near_object()` instinctive primitive will mark "not" (or in other cases where not is not specified, "near") in the matched cell and all the adjoining non-null cells. Thus, as shown in Figure 10, there is now written <"not"> in cell (2,1,0) and adjoining cell (2,0,0).

The instinctive primitive `parse_sentence()` now continues again parsing the sentence. There is nothing left to parse (as can be seen from Navigation Module B in Figure 7 the instruction sentence has only blank cells after the white cylinder words). Thus, the instinctive primitive `parse_sentence()` goes Navigation Module A and looks for tags. As can be seen in Figure 10, in the cell (0,0,0) there is the tag <"place">. This tag matches against the instinctive primitive `place_object()` which is then activated and starts operating on Navigation Module A.

At this point, the contents of Navigation Modules A and B are shown in Figure 11.

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	<"not"> cylinder,white, link{0023,0,0,0}	air	air	air
<"place"> triangle, white, link{0024,0,0,0}	air	<"not"><"top"> block, black, link {0022,0,3,0}	air	<"top"> black, block, link{0021,0,0,0}	air

Navigation Module A

"not", link{+}	"near", link{+}	"a", link{+}	"white", link {+}	"cylinder", link{+}	
"of", link{+}	"the", link{+}	"black", link{+}	"block", link {+}	"which", link {+}	"is", link {+}
"place", link{+}	"the", link{+}	"white", link{+}	"triangle", link {+}	"on", link{+}	"top", link{+}

Navigation Module B

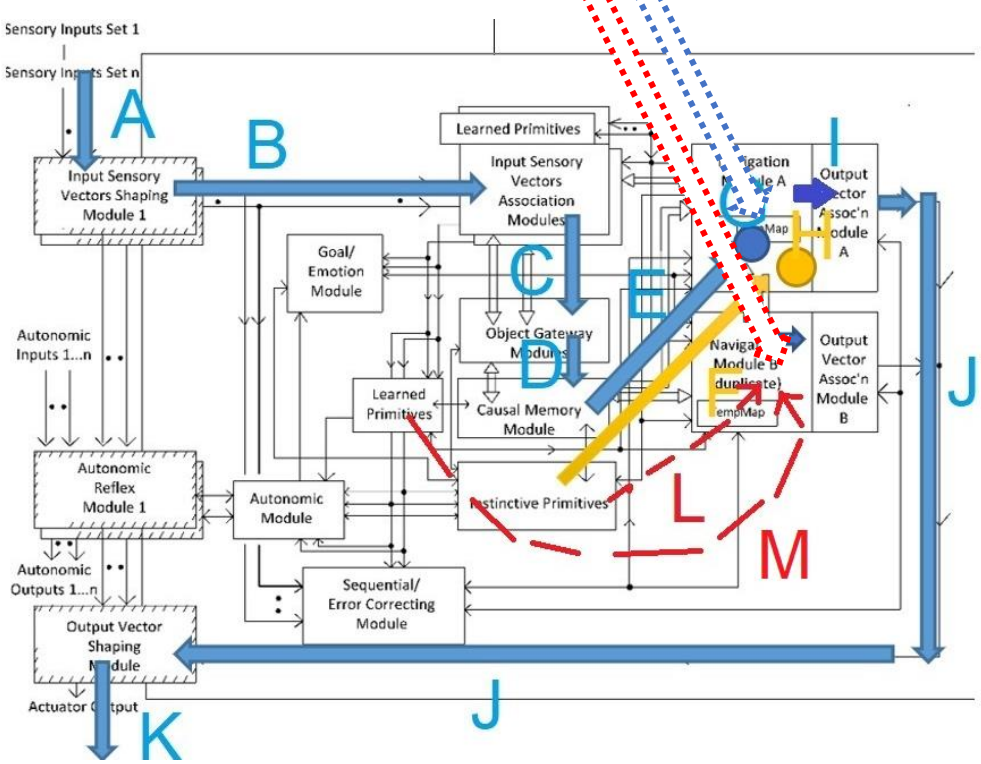

Figure 11. A navigation map representing a far view of Figure 1 is in Navigation Module A. Another navigation map representing the instruction associated with Figure 1 ("place the white triangle on top of the black block which is not near a white cylinder") is in Navigation Module B. Navigation Module A cells have been modified by the instinctive primitives `parse_sentence()` and `physics_near_object()`. See text for full description.

The `place_object()` instinctive primitive will attempt to move and place an object somewhere. The tag <"place"> is most closely associated with "triangle, white"—that is the object that will be placed somewhere. The primitive now looks for other tagged notations such as <"top"> in the navigation map in Navigation Module A (Figure 11). In this example, it will see a <"not"> in the cell with the features of a black block at (2,0,0) and not consider the <"top"> tag in that cell . However, the cell with the features of black block at (4,0,0) has a valid tag such as <"top">.

The instinctive primitive `place_object()` now triggers the instinctive primitive `move()` which produces actions from Navigation Module A to move the object from cell (0,0,0) (i.e., the white triangle) to the cell (4,0,0) (i.e., on top of whatever is in this cell). (The current simulation is running in two dimensions and the result is the simulated output is actually for the white triangle to move into the same cell as the black block on the right.)

The instinctive primitive `move()` produces actions from Navigation Module A that cause an action signal to go to the Output Vector Association Module A. The Output Vector Association Module A performs motion planning and motion corrections via feedback from the Sequential/Error Correcting Module. The motion action signal from the Output Vector Association Module A is shown as Arrow J in Figure 11. This signal goes to the Output Vector Shaping Module which produces the actual signals (arrow K, Figure 11) for actuators to move the white triangle to the cell with the black block on the right. Thus, the white triangle is then moved to (and on top of it in three dimensions) the black block on the right side.

Then a new cognitive cycle repeats again, finally processing again the actual sensory inputs streaming into the architecture.

4.5 Evidence Supporting the Generalization of the CCA6's Ability to Process Compositional Instructions

Formally proving that a logical system, which the CCA6 is in addition to its other properties, is compositional is difficult. In fact, for example, Kracht (2013) argues that logical languages are actually not compositional. While natural languages, e.g., English, would intuitively seem to be compositional, in fact the literature contains many examples of why they are not fully compositional (e.g., Krempel, 2019; Szabó, 2020). However, our goal at this point is not necessarily a fully rigorous mathematical proof of the CCA6's ability to process compositional instructions, but rather, its pragmatic ability to offer such a solution.

Symbolic AI systems are considered to have reasonably good potential for compositionality (Fodor and Pylyshyn, 1988). For example, Winograd's system from the early 1970s was discussed above (Winograd, 1971). In the presented CCA6 architecture, the instinctive primitives can essentially allow any basic symbolic operation that a symbolic computer can perform. Thus, from this theoretical consideration, a similar level of compositionality would be expected from the CCA6 architecture in theory, as would be expected from a symbolic computer system, the issue of resources aside.

Consider the compositional example from Figure 1 of "place the white triangle on top of the black block which is not near a white cylinder." In the previous sub-section the processing of this example by the CCA6 architecture was demonstrated. If the example is now negated, i.e., it becomes "place the white triangle on top of the black block which is ~~**not**~~ near a white cylinder," then correct compositional comprehension and behavior should still be exhibited by the architecture.

Figure 12 shows the new negated instructions corresponding to Figure 1.

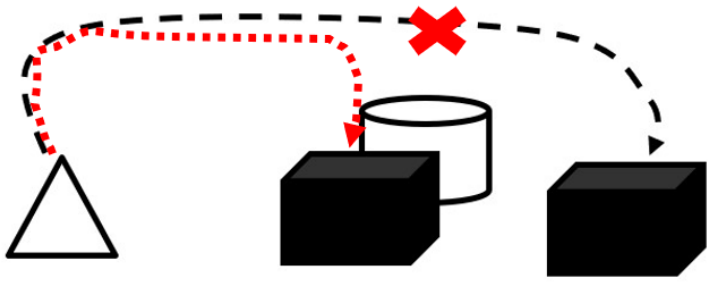

Figure 12. New instructions for the blocks problem shown above in Figure 1: "Place the white triangle on top of the black block which is near a white cylinder." (Note: The dashed arrows, representing the solution, are not shown to the architecture, but are for the reader only.)

Using similar mechanisms as described in the previous sub-section, the CCA6 architecture maps the objects of Figure 12 into Navigation Module A. Using similar mechanisms as described in the previous sub-section, the CCA6 architecture maps the instructions associated with Figure 12 into Navigation Module B. Similar instinctive primitives are triggered, and similar mechanistic steps occur, with operations by the Navigation Module on the contents of Navigation Module A.

After a number of mechanistic steps (very similar to those in the previous sub-section) the contents of Navigation Module A are shown in Figure 13. Note that the tag <"near"> is in cells (2,0,0) and (2,1,0). In the next step the `place_object()` instinctive primitive will find the best match for <"top"> and <"near"> in cell (2,0,0), resulting in the placement of the white triangle on top of the black block which is near a white cylinder.

As before, the instinctive primitive `move()` operation on the Navigation Module A causes an action signal to go to the Output Vector Association Module A. The Output Vector Association Module A performs motion planning and motion corrections via feedback from the Sequential/Error Correcting Module. This signal goes to the Output Vector Shaping Module which produces the actual signals (similar to arrow K in Figure 11) for actuators to move the white triangle on top (in three dimensions) or simply within the cell (in the two-dimensional simulation) of the black block which is indeed near a white cylinder. Then a new cognitive cycle repeats again, finally processing again the actual sensory inputs streaming into the architecture.

This again, further supports the thesis of this paper that with small biologically evolutionarily plausible changes from the previous architecture CCA5, compositional language comprehension and behavior emerge in this new CCA6 cognitive architecture.

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	<"near">, cylinder,white, link {0023,0,0,0}	air	air	air
<"place">, triangle, white, link{0024,0,0,0}	air	<"near">,<"top">, block, black, link {0022,3,3,0}	air	<"top">, black, block, link {0021,0,0,0}	air

Figure. 13. Intermediate results for the instruction "Place the white triangle on top of the black block which is near a white cylinder." (Navigation map from the Navigation Module A, similar to what is seen in Figure 11, albeit with different contents.)

4.6 Evidence Supporting the CCA6's Compositional Abilities: Comparative Evidence

At the time of this writing, neural network large language models (LLM) have started demonstrating human-like responses to many problems. Brown and colleagues (2020) describe the history of pre-trained language representations in natural language processing AI systems, with the recent use of transformer language models (Vaswani et al., 2017). Brown and colleagues note that as the capacity of transformer language models have increased, so have their abilities in text synthesis. For example, they describe a 175 billion parameter autoregressive language model named GPT-3 created by their company OpenAI with strong performance on various natural language processing (NLP) tasks. Versions of this large language model were enhanced with further transfer learning and increased conversational abilities via both supervised learning and reinforcement learning. The version named ChatGPT was released by the company OpenAI in November 2022 (OpenAI, 2023).

Soon after the release of ChatGPT, Kung and coworkers (2022) showed that this large language model, without any extra specialized training, could actually perform at or near the threshold for passing of the full United States Medical Licensing Exam. This multi-part examination is required to become licensed as a physician in many states in the USA. The third part of the examination is taken by candidates who have already graduated from medical school.

It includes complex clinical data and somewhat ambiguous clinical scenarios that can be challenging for graduated physicians.

The compositionality problem given above in Figure 1 and used in the demonstration example of the CCA6 operation above, was as best as possible identically presented to the ChatGPT large language model. The online, readily available version of ChatGPT was used on June 4, 2023 (OpenAI, 2023). The instruction given to ChatGPT is shown in Figure 14 as is the response of ChatGPT. Although ChatGPT was able to perform successfully on the complex United States medical licensing exams, note that it was not able to compositionally solve the instruction of Figure 1 correctly. The question was re-submitted to ChatGPT multiple times since there is a randomness involved in how the answer is arrived at. Only on the ninth try did ChatGPT give a response with the triangle correctly on the right-hand black block.

It should be noted that these results are not expected to remain static. The exact same problem was tried again on June 6, 2023, with similar poor results by ChatGPT. However, there may eventually be training by ChatGPT on the submitted problem, and in the future ChatGPT may be able to solve it properly. As well, LLM's are continually improving. For example, chain of thought and tree of thought automatic re-prompting of LLM's can give dramatic performance improvements (Yao et al., 2023). However, at the time of writing (June 4 and June 6, 2023), the basic compositional abilities which are now part of the Causal Cognitive Architecture, do not, in this simple trial, appear to be present in ChatGPT, an LLM which was able to pass the United States medical licensing examinations.

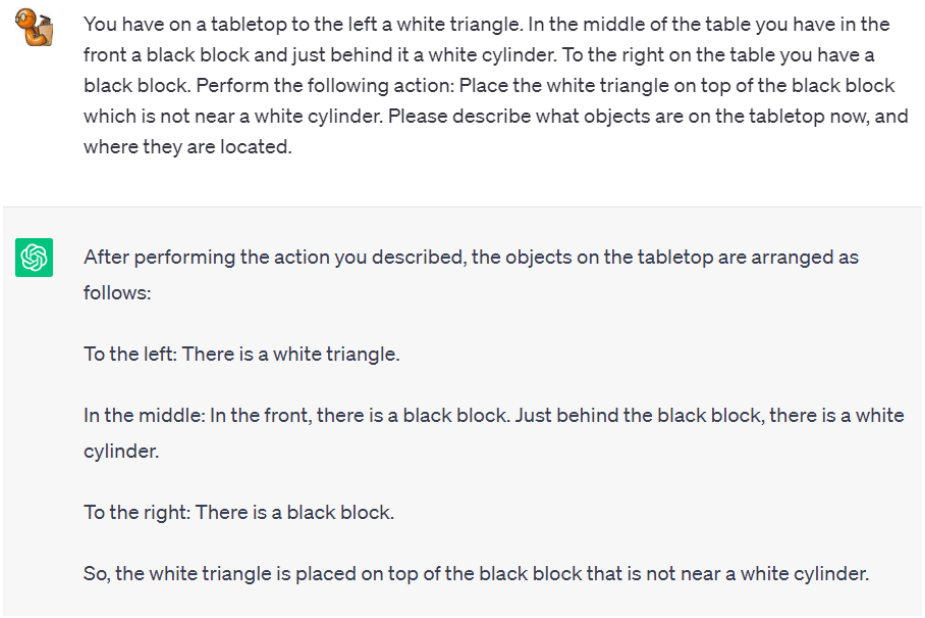

Figure. 14. The compositionality problem given by Figure 1 is presented to ChatGPT (OpenAI, 2023). Results of first trial. (June 4, 2023; the symbol of person represents the instructions presented to ChatGPT; the swirling symbol is a logo of the company OpenAI (reproduced in compliance with usage terms), and represents the response from ChatGPT)

5. Discussion

5.1 Comparison of the CCA6 with Other Cognitive Architectures

Cognitive architectures can be thought of as computational models of cognitive phenomena. As Varma (2014) describes, each field of science attempts to create a unified theory of its phenomena, with cognitive architectures attempting to provide such an account of phenomena in cognitive science.

A large number of cognitive architectures have been presented over the years (Samsonovich, 2010; Kotseruba and Tsotsos, 2020). It is beyond the scope of this paper and section to compare the presented CCA6 architecture individually against the vast array of cognitive architectures proposed and developed in the past. However, it is useful

to consider the work of Laird, Lebiere and Rosenbloom (2017) who have proposed a standard model of the mind (Figure 15). In keeping with Varma (2014) this model is essentially a unification of the many cognitive architectures which attempt to model the mind, just as the well-known standard model in physics attempts to unify phenomena from that field. Figure 15 is a simple depiction of this standard model of the mind, and indeed appears general enough to accommodate most cognitive architectures listed in Samsonovich (2010) or Kotseruba and Tsotsos (2020). However, despite this very generic model, it does not reasonably capture the core elements of the CCA6 presented above.

Unlike the standard model of the mind, in the CCA6, procedural long-term memory can actually be kept in the same navigation maps as the declarative long-term memory. Often the Working Memory, i.e., the Navigation Module is involved directly, as per the standard model of the mind, but it does not have to be—operations can actually be performed on cells in many navigation maps by primitives in those and other navigation maps simultaneously without the direct involvement of the Navigation Module.

Other core features of the Causal Cognitive Architecture 6 (CCA6) are somewhat different than many other cognitive architectures, again as reflected by the standard model of the mind. For example, the CCA6 is specifically defined by the binding of data onto navigation maps which are then bound along with temporal features onto further navigation maps. For example, the ability of the CCA6 to feed back intermediate results of navigation maps and obtain cause-and-effect results as well as analogous inductive inference from this process, is somewhat novel compared to other described cognitive architectures.

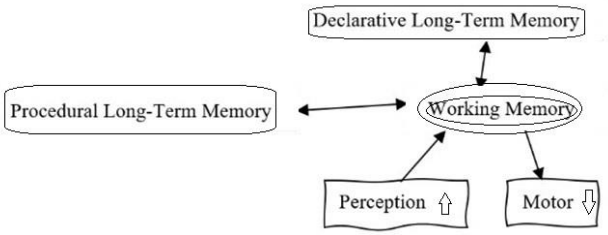

Figure. 15. Overview of the Structure of the Standard Model of the Mind

5.2 A Plausible Evolutionary Path for the Emergence of Language in Humans

Despite the relatively small genetic differences between chimpanzees and humans (e.g., Analysis Consortium, 2005), humans are capable of full compositional language usage, while chimpanzees are not (e.g., Oña, Sandler, Liebal, 2019). Of interest, the Causal Cognitive Architecture 6 suggests a plausible evolutionary path requiring a relatively modest number of genetic changes for the relatively quick emergence of compositional language in humans from chimpanzee-human last common ancestors. ("Chimpanzee" is used here as shorthand for *Pan*, i.e., chimpanzees and bonobos.)

As noted above, the Causal Cognitive Architecture is brain inspired. It treats the cortical minicolumns in the mammalian brain as navigation maps and considers the behavior of a system made up of millions of these navigation maps, and the plausible evolutionary paths in such a model. In earlier versions of this type of architecture (Schneider, 2022b) it was noted that a very simple primitive (i.e., procedure) could be used to read back and to communicate navigation maps to other agents. In this paper, given a number of modest architectural changes and the expansion of the instinctive primitives, there is the ready emergence of compositional language comprehension.

The hardware engineer would find the architecture of the Causal Cognitive Architecture strange—for example, it seems inefficient to feed back the intermediate results of the Navigation Module in some contorted fashion and store these results in the input sensory circuits (e.g., Schneider, 2022a). Why not instead create some temporary storage registers? It is because although the Causal Cognitive Architecture is functionalist in many aspects, it is reasonably constrained by biology where this is considered relevant. From an evolutionary point of view, it requires fewer genetic modifications to simply enhance the existing ubiquitous feedback circuits and store the intermediate results in the upstream module. In the next cognitive cycle these temporarily stored results are in fact swept downstream and automatically processed again in the Navigation Module, as discussed above and in the literature.

Once the advantageous nature of being able to re-operate on the intermediate results became established, it is reasonable to hypothesize that it became evolutionarily more likely to genetically further allow other gene duplications and modifications to create some neural circuits acting as temporary memory registers, as in fact, were required for the analogical reasoning proposed in the CCA5 architecture (Schneider, 2023). These temporary registers are reflected by the "TempMap" register in the Navigation Module (Figures 2, 3).

Architectural changes in the proposed CCA6 architecture of this paper are all biologically constrained as well. Once the usefulness of reading back navigation maps (e.g., the primitive (i.e., procedure) mentioned above `navigation_map_to_proto_language()`) to allow simple communication between agents using a Causal Cognitive Architecture model occurs (most likely occurring already in the chimpanzee-human last common ancestor capable of hand gesture communication, e.g., Taglialatela and colleagues, 2008), it is hypothesized that it became evolutionarily more likely to further allow gene duplication and modification of the Navigation Module to a Navigation Module B (Figure 3) allowing more efficient reading back and comprehension of navigation maps, i.e., the development of compositional language.

As noted above, the emergence of the second Navigation Module B is inspired by the ability of human and non-human primate brains to handle communications in a more expressive fashion. As mentioned, it seems reasonable to assume that in a chimpanzee-human last common ancestor the left Broca's area of the brain was already being used for hand gesture communications, as Taglialatela et al. (2008) note now occurs in chimpanzees. Also, numerous genetic and developmental mechanisms have been proposed for mechanisms of how brain circuits can duplicate and diverge to new functions (e.g., Rakic, 2009, e.g., Chakraborty and Jarvis, 2015). As well, similarities in the auditory-vocal cortical connectivity in non-human primates and humans (Aboitiz, 2018) support the duplication of the Navigation Module to include a Navigation Module B in non-human primates, i.e., before humans emerged.

Leading up to the point of the of the chimpanzee-human last common ancestor, given hypothesized navigation maps-like structures (i.e., the cortical minicolumns), it is a small step to evolve to read back these navigation map-like structures, and produce a simple proto-language. Observations of extant chimpanzees' left Broca's area of the brain being used for hand gestures and other simple muscle-controlled communication, supports a simple proto-language as this time. As well, as noted above, similarities in the auditory-vocal cortical connectivity supports a language area for comprehension starting to have emerged, akin to the duplication of the Navigation Modules in the CCA6 model.

Changes in the ubiquitous feedback circuits in the brain are postulated in Schneider (2023) to have allowed causal abilities and inductive analogical reasoning to emerge from the chimpanzee-human last common ancestor as it did in the Causal Cognitive Architecture (as well as to have allowed the development of more frequent psychoses in humans (Schneider, 2020)). These same changes, which would not have required extensive modification in the genetic code, are required for the mechanisms of the newer compositionality/language instinctive primitives presented in the current paper above. These instinctive primitives could have easily emerged from duplications or modifications of other primitives that already existed with respect to parsing navigation over landscapes and the physics of objects in the environment.

The result, it is hypothesized, is that much as language and compositionality emerged in the development of the Causal Cognitive Architecture, the mammalian and primate brain which the architecture is essentially inspired by, similarly could have easily emerged a proto-language and then over no more than a few million years of evolution, a full compositional language ability.

It is important to emphasize that work on the Causal Cognitive Architecture was not initiated from the viewpoint of linguistic goals. Full consideration of the evolution of language is beyond the scope of this paper. However, as noted above, compositional language abilities seem to readily emerge from the architecture. Given the biological constraints on the development and operation of the architecture, the model gives a possible plausible evolutionary route for the development of human language in a relatively short evolutionary timescale.

5.3 Another Pathway to Real World Artificial Intelligence

In keeping with the BICA Challenge (Samsonovich, 2012), research of biologically inspired architectures can allow cognitive modeling as well as help to create better future AI systems.

Since the architecture presented here is loosely modeled on the mammalian and the human brain at the mesoscopic scale, it provides a possible brain-inspired pathway towards real world AI. The grounding problem (e.g., Harnad, 1990; Barsalou, 2020) is more than a philosophical problem (e.g., a person with no knowledge of language A trying to learn language A from an A–to–A dictionary) but becomes an engineering problem in creating capable AI systems.

Fuller grounding allows an AI system to better ascribe meaning to internal representations, allowing better solutions to some problems. The Causal Cognitive Architecture, including the CCA6, implements full grounding (Schneider, 2023), in contrast to many current popular deep learning AI systems, including large language models, which do not.

Analogical reasoning can be extremely powerful in solving a host of problems, including many day-to-day ones. Analogical reasoning may in fact be at the core of human cognition (Hofstadter, 2001). While various arrangements of deep learning systems, including large language models, do have various levels of analogical reasoning, the emergent core analogical reasoning of the Causal Cognitive Architecture (Schneider, 2023) in combination with a future rich set of instinctive and learned primitives, may prove to yield a robust, human-like reasoning about the world.

As shown in this paper, compositionality emerges with the architectural changes in the CCA6. This compositional ability allows the very simple simulation of the architecture to solve problems such as moving the white triangle onto the black block (Figure 1), while vastly more capable AI systems such as the large language model ChatGPT did not succeed in the examples above. While such large language models (LLMs) may eventually replicate most aspects of human intelligence, doing so in the near future may require somewhat roundabout techniques. For example, the Winograd Schema Challenge (pairs of sentences with referential ambiguities) was difficult for AI systems, particularly statistical methods, while being fairly easy for humans (Levesque, Davis, Morgenstern, 2012; Davis, Marcus, 2015). Nonetheless, the LLM GPT-3 (related to ChatGPT) was able to approach the level of human performance on this challenge (Brown et al., 2020). However, the success of the LLMs on this test may be via biases in the datasets and "knowledge leakage" from LLM training data, rather than through human-like commonsense reasoning (Sakaguchi et al., 2021; Kocijan et al., 2023). As Davis and Marcus essentially wrote in 2015 and Kocijan and colleagues write again in 2023, "the problem of commonsense reasoning still stands as one of the major challenges facing AI."

The purpose of the simulation of the architecture presented in this paper, at this point, is largely conceptual, i.e., to show that the operation of the architecture is feasible. In contrast, current large language models are massive engineering achievements yielding aspects of human level intelligence (e.g., Kung et al., 2022; Bubeck et al., 2023). It is indeed somewhat premature to compare the solution of toy problems by CCA6 architecture to them. Nonetheless, the CCA6 architecture presented in this paper, suggests a different approach to possible AI in the future, with its own advantages as noted above.

5.4 Further Work: Better Experimental Simulations

As noted earlier, at this point in its development the CCA6 architecture is not at the stage to consider operations on a dataset of thousands of non-toy compositional examples. Rather, the purpose of this paper is to show that with small biologically evolutionarily plausible changes from the previous architecture, compositional language comprehension and behavior emerge. This was done above with a computer simulation and a walk through of the mechanisms in the solution of the demonstration example of Figure 1. Then after doing so, further generalization of the example was considered in terms of a solution of the negated case. While the demonstration example of Figure 1 is indeed a toy problem, the performance of the CCA6 architecture on this toy problem in fact compared favorably with the performance of a state of the art (at the time of writing, of course) large language model in understanding and solving the exact same problem shown in Figure 1.

A priority in future work on the architecture is in fact the ability to operate and test the architecture on larger datasets of compositional problems. The immediate need in this area is a larger collection of instinctive primitives, and better usage and implementation of the learned primitive system. As well, in order for the architecture to simply solve a compositional problem on its own agency, it first needs to have acquired the necessary navigation maps (a sort of education, so to speak) to even understand what is meant by solving a compositional problem.

It has proved tedious in constructing a default starting collection of instinctive primitives and other navigation maps representing experience or education. In the future work, the use of large language models (LLMs) (e.g., Bubeck et al., 2023) could be advantageous to rapidly simulate a large collection of navigation maps about the external world to be stored in the Causal Memory Module with respect to both Navigation Module A (general navigation maps) and Navigation Module B (navigation maps related to language and compositional phenomena). For example, the Python library LangChain (Topsakal and Akinci, 2023) has already been included (but not utilized) in the current CCA6 simulation and allows direct Python access to present and future LLMs. Figure 16 shows part of the code from the simulation that uses an LLM to generate the raw data which can be converted into ten versions of a navigation map (printed out for now, two versions shown in Figure 17) for any object. As can be seen by the results, conversion of the LLM outputs to a useful navigation map will obviously take additional work to create more useful navigation maps,

but can be automated. However, using the LLM as such could rapidly populate the Causal Memory Module with background navigation maps.

Using a similar LLM approach to generate instinctive primitives (i.e., the basic procedures which the architecture defaults with) is more complex, as such navigation maps simply do not exist with any frequency in the vast corpus of knowledge the LLM was trained on. However, the LLM could be used generate analogous instinctive primitive navigation maps based on prompted instinctive primitives, and future work may reveal methods to filter a generated set of instinctive primitives for instinctive primitives which are useful.

```
1075    def create_background_navmaps_with_LLM()→ bool:
1076        """
1077        cca6; preproduction; to do: move import, pylint re-enable, exception handling, parse to navmap
1078        using LLM, e.g., ChatGPT OpenAI, to generate background navigation maps for storage as resource
1079        """
1080        from langchain.llms import OpenAI #future production note: move to main import section
1081        #key = #secure temp global
1082        llm = OpenAI(openai_api_key=key)
1083        text_block = get_object_to_generate_background_navmp()  #e.g., "frypan"
1084
1085        prompt = "Use a 6 x 6 table to represent a scene in the world. Please populate this table with features
1086        prompt = prompt + text_block
1087
1088        print("\nThis is an experimental feature for future use. Self-test diagnostic mode.")
1089        print("LLM will generate a navigation map to store in the Causal Memory Module to populate the architectu
1090        print(f"Ten versions of the object {text_block} will be generated to create ten navigation maps for stor
1091        for __ in range(10):
1092            print(llm(prompt))  #future production note: parse and create navigation map and store in Causal Mem
1093        return True
```

Figure. 16. Python code tested within the simulation of the CCA6 which uses a commercially available LLM to generate ten potential versions of navigation maps for various objects. Full text of variable **prompt** which is partially cut off in the diagram: "Use a 6 x 6 table to represent a scene in the world. Please populate this table with features from a "

```
This is an experimental feature for future use. Self-test diagnostic mode.
LLM will generate a navigation map to store in the Causal Memory Module to populate the architecture.
Ten versions of the object picnic basket will be generated to create ten navigation maps for storage:



| Picnic Basket | Food | Drinks | Plates | Utensils | Blanket |
|---------------|------|-------|--------|----------|---------|
| Basket        | Bread| Soda  | Plates | Forks    | Blanket |
|               | Cheese| Juice | Bowls  | Knives   |         |
|               | Fruit | Water | Cups   | Spoons   |         |
|               | Chips |       |        |          |         |
|               |       |       |        |          |         |
|               |       |       |        |          |         |


|Wicker|Food|Drink|Tablecloth|Fork|Spoon|
|------|----|----|----------|----|-----|
|✓     |✓   |✓   |✓         |✓   |✓    |
```

Figure. 17. For the object "picnic basket" from the code above, two of the ten versions of raw data that can be parsed and incorporated into navigation maps to be stored in the Causal Memory Module to give the architecture a store of background information about the external world.

6. Conclusion

Compositionality is an essential property for AI systems. Without compositionality, an intelligent system may need to see a myriad of permutations of the many situations it could encounter in order to learn them. In the area of language, for example, with compositionality an intelligent artificial system (or human) can understand essentially an unlimited number of different sentences.

While symbolic systems experimentally for the last half century have been able to implement compositionality, artificial neural network AI systems have challenges in implementing this ability. Thus, there has been much interest in neurosymbolic-like systems which possess both connectionist and symbolic properties. Many such architectures and tools exist (e.g., Kautz, 2022; Bride et al., 2021). However, such systems are usually agglomerations of different systems with the corresponding challenges of making them work well together.

The Causal Cognitive Architecture is a brain-inspired cognitive architecture (BICA) which is not a traditional artificial neural network architecture, nor a traditional symbolic AI system. Instead, based on the postulation that each mammalian cortical minicolumn represents a navigation map, the architecture uses modified neural networks as its fundamental circuits representing navigation maps. In this paper, the existing Causal Cognitive Architecture 5 (CCA5) was changed into a new architecture the Causal Cognitive Architecture 6 (CCA6). A significant change was the duplication of the Navigation Module.

The duplicated Navigation Module B was inspired by the ability of human and non-human primate brains to handle communications in a more sophisticated fashion than seen in other mammals. Indeed, from an evolutionary perspective, similarities in the auditory-vocal cortical connectivity in non-human primates and humans (Aboitiz, 2018) would support this hypothesis. Given the hypothesis of the previous versions of the Causal Cognitive Architecture that enhanced feedback pathways allowed the emergence full causal and analogical abilities in humans (e.g., Schneider, 2023), it seemed that this improved ability to operate on intermediate results, should allow much improved language abilities as well.

The CCA6 architecture was formalized (Appendix A) and a simple, experimental Python computer simulation was created. The purpose of the simulation of the architecture presented in this paper, at this point, is largely conceptual, i.e., to show that the operation of the architecture is feasible and that with small biologically evolutionarily-like plausible changes from the previous architecture, compositional language comprehension and behavior could emerge.

Figure 7 provides a view into the mechanisms at work in the architecture in processing the sensory scene of Figure 1 and its instruction to place the white triangle on a black block not near a white cylinder. Cognitive cycles are cycles of sensory inputs being processed and producing an output (or no action or a feedback action allowing continued processing of temporary results in the navigation modules, as described above). Inputs of a given sensory system are matched against the best matching stored navigation map in that sensory system's Input Sensory Vectors Association Module (Figure 7). That best matching stored navigation map is retrieved and then updated with the actual sensory input of that sensory system (or a new navigation map is created if there are too many changes). Predictive-like coding occurs—the architecture anticipates what it is sensing and then considers the differences with the actual input signal. Matching continues as the single-sensory navigation maps are then mapped onto a best matching multi-sensory navigation map taken from the Causal Memory Module (arrow D in Figure 7). The best matched navigation map from the Causal Memory Module is then updated with the actual sensory inputs and is copied into Navigation Module A. For example, in Figure 7 the sensory scene of the white triangle and other objects of Figure 1 is represented by the navigation map in Navigation Module A.

The availability of a second Navigation Module B allows the instruction associated with Figure 1 (i.e., "place the white triangle on top of the black block which is not near a white cylinder") to be represented here, as shown in Figure 7. The instruction "place" in Navigation Module B triggers the `parse_sentence()` instinctive primitive. Whatever word the `parse_sentence()` primitive encounters, it matches against Navigation Module A. To continue the example, the first word encountered "place" (cell (0,0,0) of the navigation map in Navigation Module B of Figure 7) is matched against words in the Causal Memory Module, and recognized as an action. The action will be applied to the next words after it, until the primitive comes to another action word. The next words prior to the next action words are "the", "white" and "triangle". These words are found in cell (0,0,0) in the navigation map in Navigation Module A of Figure 7. Thus, as shown in Figure 8, the Navigation Module has written `<"place">` in this cell (0,0,0) containing the feature white, triangle.

The architecture continues processing the contents of Navigation Module A and Module B. By virtue of a series of mechanical-like operations, in this example the architecture goes through the instruction "place the white triangle on top of the black block which is not near a white cylinder." In some cognitive cycles a word in the instruction may trigger the `physics_near_object()` instinctive primitive which will mark cells in a navigation map which are "near" a specified object. The result is that Navigation Module A becomes modified as shown in Figure 11. The `place_object()` instinctive primitive is triggered by the tag <"place"> (Figure 11) and attempts to move and place an object somewhere. The tag <"place"> is most closely associated with "triangle, white"—that is the object that will be placed somewhere. The primitive now looks for other tagged notations such as <"top"> in the navigation

map (Figure 11). It will see a <“not”> in the cell with the features of a black block at (2,0,0) and not consider the <“top”> tag in that cell . However, the cell with the features of black block at (4,0,0) has a valid tag such as <“top”>. The instinctive primitive `place_object()` now triggers the instinctive primitive `move()` operating on the Navigation Module A to move the object from cell (0,0,0) (i.e., the white triangle) to the cell (4,0,0). Thus, the white triangle is successfully moved to the black block on the right side—correct compositional action.

At a conceptual level the architecture appears to work and readily allows compositional language comprehension and behavior. To have solved the compositionality problem of Figure 1 the architecture had to have been capable of a number of tasks. The first was the ability to “understand” the language of the instruction which itself was compositional in nature: “place the white triangle on top of the black block which is not near a white cylinder.” Then the meaning of each decomposed part of the instruction had to be “understood” and correctly processed. The new work, i.e., the CCA6 architecture, appeared able, albeit on a limited toy problem, to do so. These compositional abilities are now core mechanisms of the architecture, particularly associated with Navigation Module B. Note that these operations essentially occur in a simple, mechanical fashion. Each step of the operation has little “understanding” of language. The architecture was readily able to solve the negation of this problem. Of interest, as shown above, a state of the art (at the time of writing) large language model was not able to correctly solve this simple demonstration problem.

It is not reasonable at this point to test the architecture on large datasets of real-world problems—a larger collection of instinctive primitives and a better implementation of the learned primitive system are required. As well, as mentioned above, in order for the architecture to simply solve a compositional problem on its own agency, it first needs to have a considerable education to acquire the necessary navigation maps to even understand what is meant by solving any sort of problem. Further work on the CCA6 was discussed in terms of using automated means such as LLMs to enhance the collection of instinctive primitives and enhancing the architecture’s background of experiential navigation maps (i.e., an education so to speak), to allow testing on larger datasets of real-world compositional problems.

Compositional language abilities seem to readily emerge from the CCA6 architecture. Given the mammalian brain inspiration of the architecture, it suggests that it is indeed feasible for modest genetic changes from the chimpanzee-human last common ancestor to have allowed the emergence of compositional language in humans.

It is, of course, somewhat incongruous to compare sophisticated engineering achievements such as LLMs or other AI models capable of yielding aspects of human level intelligence, with the largely conceptual CCA6 architecture. Nonetheless, the presented CA6 architecture does suggest another pathway towards a real-world artificial intelligence, albeit one inspired by the mammalian brain.

References

Aboitiz, F. A. (2018). Brain for Speech. Evolutionary Continuity in Primate and Human Auditory-Vocal Processing. *Front Neurosci.* **12**:174 doi: 10.3389/fnins.2018.00174.

Alme, C. B., Miao, C., Jezek, K., Treves, A., Moser, E. I., & Moser, M. B. (2014). Place cells in the hippocampus: eleven maps for eleven rooms. *Proceedings of the National Academy of Sciences of the United States of America*, 111(52), 18428–18435. https://doi.org/10.1073/pnas.1421056111

Amici, F., Oña, L., Liebal, K. (2022). Compositionality in primate gestural communication and multicomponent signal displays. *International Journal of Primatology* **28**:1-9.

Analysis Consortium. (2005). Initial sequence of the chimpanzee genome and comparison with the human genome. *Nature*, *437*(7055).

Barsalou L. W. (2020). Challenges and Opportunities for Grounding Cognition. *Journal of Cognition*, **3**(1),31. doi.org/10.5334/joc.116

Bernal, B., Ardila, A. (2009). The role of the arcuate fasciculus in conduction aphasia. *Brain.* Sep 1;**132**(9):2309-16

Binder, J.R. (2015). The Wernicke area: Modern evidence and a reinterpretation. *Neurology* **85**(24):2170-5 doi: 10.1212/WNL.0000000000002219.

Bride, H., Cai, C. H., Dong, J., Dong, J. S., Hóu, Z., Mirjalili, S., Sun, J. (2021). Silas: A high-performance machine learning foundation for logical reasoning and verification. *Expert Systems with Applications*, **176**, 114806.

Brincat, S. L., Donoghue, J. A., Mahnke, M. K., Kornblith, S., Lundqvist, M., & Miller, E. K. (2021). Interhemispheric transfer of working memories. *Neuron*, **109**(6), 1055–1066.e4 doi: 10.1016/j.neuron.2021.01.016

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, **33**, 1877-1901.

Brushe, M.E., Lynch, J.W., Reilly, S., Melhuish, E., Brinkman, S.A. (2020) How many words are Australian children hearing in the first year of life? BMC Pediatr. Feb 3;20(1):52. doi: 10.1186/s12887-020-1946-0

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., Nori, H. (2023). Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv*:2303.12712

Chakraborty, M., Jarvis, E.D. (2015). Brain evolution by brain pathway duplication. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, **370**(1684), 20150056 https://doi.org/10.1098/rstb.2015.0056

Davis, E., & Marcus, G. (2015). Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, *58*(9), 92-103. https://doi.org/10.1145/2701413

Fodor, J. A., Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, **28**(1-2), 3-71.

Frege, G., [1884] (1950) *The Foundations of Arithmetic*, Austin, J.L. (trans.), Oxford: Blackwell. Reprinted Northwestern University Press, Evanston, IL (1980).

Harnad, S. (1990). The symbol grounding problem. *Physica D*, **42** (1-3), 335-346. doi:10.1016/0167-2789(90)90087-6

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, **585**(7825), 357-362.

Hawkins, J., Lewis, M., Klukas, M., Purdy, S., Ahmad, S. (2019). A framework for intelligence and cortical function based on grid cells in the neocortex. *Frontiers in neural circuits*, **12**, 121.

Hitzler, P., Eberhart, A., Ebrahimi, M., et al. (2022). Neuro-symbolic approaches in artificial intelligence, *National Science Review* **9**(6) https://doi.org/10.1093/nsr/nwac035

Hofstadter, D.R. (2001). Analogy as the core of cognition. In Gentner, D., Holyoak, K.J., and Kokinov, B.N. editors, *The Analogical Mind: Perspectives from Cognitive Science*, pp 499–538. MIT Press.

Jiang, Y., & Bansal, M. (2021). Inducing Transformer's Compositional Generalization Ability via Auxiliary Sequence Prediction Tasks. *arXiv preprint arXiv:2109.15256*.

Kautz, H. A. (2022). The third AI summer: AAAI Robert S. Engelmore Memorial Lecture. *AI Magazine*, **43**(1), 93-104 doi: 10.1002/aaai.12036

Kim, N. (2021). *Compositional Linguistic Generalization in Artificial Neural Networks*. Doctoral dissertation, Johns Hopkins University. Retrieved Aug 28, 2023: https://jhir.library.jhu.edu/handle/1774.2/66745

Kinzler, K.D., Spelke, E.S., (2007). Core systems in human cognition. In von Hofsten, C., Rosander, K. *Progress in Brain Research*, vol **164**, chap 14.

Kocijan, V., Davis, E., Lukasiewicz, T., Marcus, G., & Morgenstern, L. (2023). The defeat of the Winograd Schema Challenge. *Artificial Intelligence*, 103971. https://doi.org/10.1016/j.artint.2023.103971

Kracht, M. (2013). Are logical languages compositional? *Studia Logica*, **101**, 1319-1340.

Krempel, R. (2019). Can Compositionality Solve the Thought-or-Language Problem? *Philosophical Papers*, **48**(2), 265-291.

Kotseruba, I., Tsotsos, J.K. (2020). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artif Intell Rev* **53**, 17-94. https://doi.org/10.1007/s10462-018-9646-y

Kung, Tiffany H., et al. (2022). Performance of ChatGPT on USMLE. *medRxiv*: https://doi.org/10.1101/2022.12.19.22283643

Kwon, K. (2014). Expressing Algorithms as Concise as Possible via Computability Logic. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **97**(6), 1385-1387. Open source: *ArXiv:* 1305.2004

Laird, J. E., Lebiere, C., Rosenbloom, P. S. (2017) A Standard Model of the Mind: Toward a Common Computational Framework across Artificial Intelligence, Cognitive Science, Neuroscience, and Robotics. *AI Magazine*, **38**(4), 13-26. https://doi.org/10.1609/aimag.v38i4.2744

Lake, B.M., Ullman, T.D., Tenenbaum, J.B., & Gershman, S.J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences, 40*, e253.

Levesque, H., Davis, E., Morgenstern, L. (2012). The Winograd schema challenge. In *Thirteenth Int. Conf. on the Principles of Knowledge Representation and Reasoning*. Retrieved 09/15/23: https://cdn.aaai.org/ocs/4492/4492-21843-1-PB.pdf

Levine, Y., Wies, N., Sharir, O., Bata, H., Shashua, A. (2020). The depth-to-width interplay in self-attention. *arXiv:*2006.12467.

Lieto, A. (2021). *Cognitive design for artificial minds*. Routledge.

Lieto, A. (2021b). Functional and Structural Models of Commonsense Reasoning in Cognitive Architectures. In Laird, J. (Ed.) *Virtual International Symposium on Cognitive Architecture VISCA 2021*. Retrieved Aug 28, 2023: https://visca.engin.umich.edu/wp-content/uploads/sites/27/2021/06/Lieto.pdf

Liška, A., Kruszewski, G., & Baroni, M. (2018). Memorize or generalize? searching for a compositional RNN in a haystack. *arXiv preprint arXiv:1802.06467*.

Marcus, G., Davis, E., Aaronson, S. (2022). A very preliminary analysis of Dall-e 2. *arXiv*:2204.13807

O'Keefe, J., Nadel, L. (1978). *The Hippocampus as a Cognitive Map*. Oxford Univ Press, Oxford.

Olsen, A.L. (2005). Using pseudocode to teach problem solving. *Journal of Computing Sciences in Colleges, 21*, 231-236.

Oña, L.S., Sandler, W., Liebal, K. (2019). A stepping stone to compositionality in chimpanzee communication. *PeerJ* 7:e7623 https://doi.org/10.7717/peerj.7623

OpenAI (2023). *Introducing ChatGPT.* https://openai.com/

Partee, B. (1984). Compositionality. In Landman, F., Veltman, F. (eds), *Varieties of Formal Semantics* **3:**281-311, Foris, Dordrecht, Netherlands

Pleyer, M., Lepic, R., Hartmann, S. (2022). Compositionality in Different Modalities: A View from Usage-Based Linguistics. *Int J Primatol* doi:10.1007/s10764-022-00330-x

Rakic, P. (2009). Evolution of the neocortex: a perspective from developmental biology. *Nat Rev Neurosci*. Oct;**10**(10):724-35. doi: 10.1038/nrn2719

Ruis, L., Lake, B. (2022). Improving Systematic Generalization Through Modularity and Augmentation. *arXiv*:2202.10745

Sakaguchi, K., Bras, R. L., Bhagavatula, C., & Choi, Y. (2021). WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Communications of the ACM*, **64**(9), 99-106. https://doi.org/10.1145/3474381

Samsonovich, A.V., Ascoli, G.A. (2005). A simple neural network model of the hippocampus suggesting its pathfinding role in episodic memory retrieval. Learn Mem. **12**(2):193-208 doi: 10.1101/lm.85205.

Samsonovich, A.V. (2010). Toward a Unified Catalog of Implemented Cognitive Architectures. In *Proceedings of the 2010 Conference on Biologically Inspired Cognitive Architectures 2010: Proceedings of the First Annual Meeting of the BICA Society*. IOS Press, NLD, 195-244.

Samsonovich, A.V. (2012). On a roadmap for the BICA Challenge. *Biologically Inspired Cognitive Architectures*. **1**;1:100-7

Schafer, M., Schiller, D. (2018). Navigating Social Space. *Neuron*, **100**(2):476-489

Schneider, H. (2020). The Meaningful-Based Cognitive Architecture Model of Schizophrenia. *Cognitive Systems Research* **59** 73-90 doi: 10.1016/j.cogsys.2019.09.01

Schneider, H. (2021). Causal Cognitive Architecture 1: Integration of connectionist elements into a navigation-based framework. *Cognitive Systems Research 66*:67-81 doi: 10.1016/j.cogsys.2020.10.021

Schneider, H. (2022a). Causal cognitive architecture 3: A Solution to the binding problem. *Cognitive Systems Research* **72**:88-115 doi: 10.1016/j.cogsys.2021.10.004

Schneider, H. (2022b). Navigation Map-Based Artificial Intelligence. *AI*, **3**(2) 434-464 doi:10.3390/ai3020026

Schneider, H. (2023). An Inductive Analogical Solution to the Grounding Problem. *Cognitive Systems Research*, **77** 74-216 doi: 10.1016/j.cogsys.2022.10.005

Spelke, E.S. (1994). Initial knowledge. *Cognition*, **50**, 431-45. doi:10.1016/0010-0277(4)90039-6

Szabó, Z.G. (2020). Compositionality. In Zalta, E. N. *(ed.). Stanford Encyclopedia of Philosophy* (rev.2020). Retrieved Mar 8,2023: https://plato.stanford.edu/entries/compositionality/

Taglialatela, J. P., Russell, J.L., Schaeffer, J.A., Hopkins, W.D. (2008). Communicative signaling activates 'Broca's' homolog in chimpanzees. *Current Biology* **18**.5:343-348

Topsakal, O., Akinci, T.C. (2023). Creating Large Language Model Applications Utilizing LangChain. In *International Conference on Applied Engineering and Natural Sciences*, Vol. 1, No. 1, pp. 1050-1056.

van Rossum, G., Drake Jr, F. L. (2014). *The Python language reference*. Python Software Foundation: Wilmington, DE, USA.

van Os, J., Hanssen, M., Bijl, R.V. , et al. (2001). Prevalence of psychotic disorder and community level psychotic symptoms: an urban-rural comparison. *Arch. Gen. Psychiatry* Jul;58(7):663-8.

Varma, S. (2014). The Subjective Meaning of Cognitive Architecture: A Marrian Analysis. *Frontiers in psychology* **5**: 440. 10.3389/fpsyg.2014.00440

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, 30, p.1.

Weber, J., Peterson, B., Hoekstra, H. (2013). Discrete genetic modules are responsible for complex burrow evolution in Peromyscus mice. *Nature* **493**,402–405 doi: 10.1038/nature 11816

Winograd, T. (1971). Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. hdl:1721.1/7095

Wu L, Jiao X, Zhang D, Cheng Y, Song G, Qu Y, Lei F. (2021). Comparative Genomics and Evolution of Avian Specialized Traits. *Curr Genomics*. Dec 31;**22**(7):496-511 doi: 10.2174/1389202923666211227143952

Xu, K., Schadt, E.E., Pollard, K.S. et al. (2015). Genomic and Network Patterns of Schizophrenia Genetic Variation in Human Evolutionary Accelerated Regions. *Molecular Biology and Evolution*. 32(5):1148-1160.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *arXiv:*2305.10601 https://doi.org/10.48550/arXiv.2305.10601

Zuberbühler, K. (2020). Syntax and compositionality in animal communication. *Philosophical Transactions of the Royal Society* B 375.1789 p20190062

Appendix A

To provide a compact yet understandable and relevant description of the Causal Cognitive Architecture 6 (CCA6), the following formalization is provided. To allow the best continuity with the existing literature, the same nomenclature adopted by the previous architecture (CCA5 – Schneider, 2023) is used here again. The architecture is described by a set of equations, albeit with many containing pseudocode. A pseudocode is an English (or other) language description of the steps a software routine follows (Kwon, 2014; Olsen, 2005). The pseudocode used in the equations below represents relatively small, straightforward operations. As such, the use of pseudocode allows a more concise description of the architecture without sacrificing much detail.

While the lower-level properties of the CCA6 architecture, and the corresponding equations, are similar to the prior CCA5 architecture, the compositional behavior and compositional language properties are new, and equations have been modified, as well as new equations added, where relevant. The similarities and differences between the architectural versions are described in the text associated with each group of equations.

An overview of the operations of the equations in terms of the architecture is provided with the equations. As well, an attempt is made to provide explanations of all symbols and all pseudocode used. Bold capitalized letters represent arrays. As an example, in equation (18) **LNM** represents an array which actually represents a navigation map. Vectors are represented in bolded italics (e.g., vector ***s*** in equation (9)). Many of the values change with time and thus are represented with the subscript *t*. For example, $\mathbf{S'}_{\sigma,t}$ (18) represents a particular array $\mathbf{S'}_{\sigma}$ whose value changes with time *t*. Note that the equations below largely follow the operation of the architecture over one cognitive cycle. Thus, the subscript *t-1* represents the previous cognitive cycle and similarly the subscript *t+1* represents the following cognitive cycle.

Although the equations below initially follow the forward flow of the input sensory information through the architecture, some of the relevant backward flow of the feedback information is also considered. In any case, an overview is always provided describing the information an equation or set of equations is representing.

The formalization does not cover the complete operation of the architecture. Many details of the architecture which are not in the direct path of the described information flow are left out of these equations. Thus, the equations do not represent, for example, the full formalization of the autonomic reflex modules, the goal/emotion module, and so on. Again, the reader should keep in mind that these equations represent a single cognitive cycle through the architecture (except for portions of the next cognitive cycle in the re-processing of intermediate results). After a cognitive cycle is over, then another cycle begins, and so on. These cognitive cycles occur many times each second and collectively are responsible for the overall behavior of the architecture.

A.1 Input Sensory Vectors Shaping Modules

In Figure 5, arrow A shows sensory inputs streaming into the Sensory Vectors Shaping Modules of the architecture. Sensory inputs for any sense are propagated into the architecture as a two-dimensional or three-dimensional spatial array of inputs. Spatial information means some sort of sensory information about a small volume of space which can be addressed and modeled as cell x,y,z. For example, for the visual sensory system, this spatial information would be visual sensory inputs (e.g., lines or no lines as a very simple example) observed at particular locations in the environment.

An array $\mathbf{S}_{\sigma}$ receives the sensory inputs of a sensory system σ every cognitive cycle, i.e., essentially changing with respect to time *t* (1 – 9). Vector ***s***(*t*) is processed by the Input Sensory Vectors Shaping Modules (via pseudocode `Input_Sens_Shaping_Mods.normalize()`, described below in Table AA1) into vector ***s'***(*t*) (10). This transformation ensures that each element of ***s'*** will be compatible with the dimensions used by the navigation maps in many of the modules of the CCA6 (11).

As Table A2 shows, processed and normalized sensory system arrays $\mathbf{S'}_{\sigma,t}$ leave this module compatible with the other data structures, i.e., the navigation maps, of the architecture. $\mathbf{S'}_{\sigma,t}$ are in arrays of dimensions m x n x o x p corresponding to the three spatial dimensions x,y,z and a fourth non-spatial dimension p (used in implementations of the architecture for the storage of segmentation data, i.e., defining objects in a scene, and metadata).

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture.

$$\mathbf{S}_1 \in R^{m1 \text{ x } n1 \text{ x } o1} \quad (1)$$

$$\mathbf{S}_{1,t} = \text{visual inputs}(t) \quad (2)$$

$$\mathbf{S}_2 \in R^{m2 \text{ x } n2 \text{ x } o2} \quad (3)$$

$$\mathbf{S}_{2,t} = \text{auditory inputs}(t) \quad (4)$$

$$\mathbf{S}_3 \in R^{m3 \text{ x } n3 \text{ x } o3} \quad (5)$$

$$\mathbf{S}_{3,t} = \text{olfactory inputs}(t) \quad (6)$$

$$\sigma = \text{sensory system identification code} \in N \quad (7)$$

$$\Theta\_\sigma = \text{total number of sensory systems} \in N \quad (8)$$

$$\boldsymbol{s}(t) = [\mathbf{S}_{1,t}, \mathbf{S}_{2,t}, \mathbf{S}_{3,t}, \ldots, \mathbf{S}_{\Theta\_\sigma,t}] \quad (9)$$

$$\boldsymbol{s'}(t) = \texttt{Input\_Sens\_Shaping\_Mods.normalize}(\boldsymbol{s}(t)) = [\mathbf{S'}_{1,t}, \mathbf{S'}_{2,t}, \mathbf{S'}_{3,t}, \ldots, \mathbf{S'}_{\Theta\_\sigma,t}] \quad (10)$$

$$\mathbf{S'}_{\sigma,t} \in R^{m \text{ x } n \text{ x } o \text{ x } p} \quad (11)$$

$\mathbf{S}_1 \in R^{m1 \text{ x } n1 \text{ x } o1}$	-defining $\mathbf{S}_1$ to be an array of dimensions m1 x n1 x o1 -the form of our data structures is important; as will be seen below, most of the architecture utilizes a 'navigation map' data structure, i.e., arrays, of size m x n x o x p -as noted in the text above, an array $\mathbf{S}_\sigma$ receives the sensory inputs of a sensory system σ every cognitive cycle; in this case array $\mathbf{S}_1$ receives the sensory inputs of sensory system 1, which are the visual sensory inputs
$\mathbf{S}_{1,t}$	an array holding visual sensory inputs, which change with time
$\mathbf{S}_{2,t}$	an array holding auditory sensory inputs, which change with time
$\mathbf{S}_{3,t}$	-an array holding olfactory sensory inputs, which change with time -for simplification implementing the olfactory sensory inputs similar to other sensory system, although this does not reflect actual mammalian neurophysiology
σ	=1 for visual, =2 for auditory, =3 for olfactory, other senses not used at present
Θ_σ	total sensory systems which above would be 3 for the current simulation
$\boldsymbol{s}(t)$	a vector holding all the sensory input arrays
`Input_Sens_Shaping_Mods.normalize()`	normalizes the sensory input arrays to the same dimensions used by the other navigation maps in the architecture
$\boldsymbol{s'}(t)$	a vector holding all the normalized sensory input arrays, each now with dimensions m x n x o x p (which in the simulation in the paper are implemented as 6 x 6 x 6, corresponding to x,y,z axes of 6 x 6 x 6, and another dimension used for object segmentation data x 16)
$\mathbf{S'}_{\sigma,t}$	normalized array (compatible with navigation maps in the other modules of the architecture) of sensory inputs of sensory system σ (which change with time)

Table A1. Explanation of Symbols and Pseudocode in Equations (1) – (11)

Input:	sensory inputs from sensory systems 1…Θ_σ
Output:	$\boldsymbol{s'}(t)$ -- a vector holding all the normalized sensory input arrays $\mathbf{S'}_{1,t \ldots} \mathbf{S'}_{\Theta\ \sigma,t}$
Description:	-Raw sensory inputs from the environment are normalized into a format compatible with navigation map data structure used by the other modules of the architecture. -The current simulation of the architecture includes an experimental environment simulation module (i.e., simulates the sensory inputs).

Table A2. Summary of the Operations of the Input Sensory Vectors Shaping Modules per Equations (1) – (11)

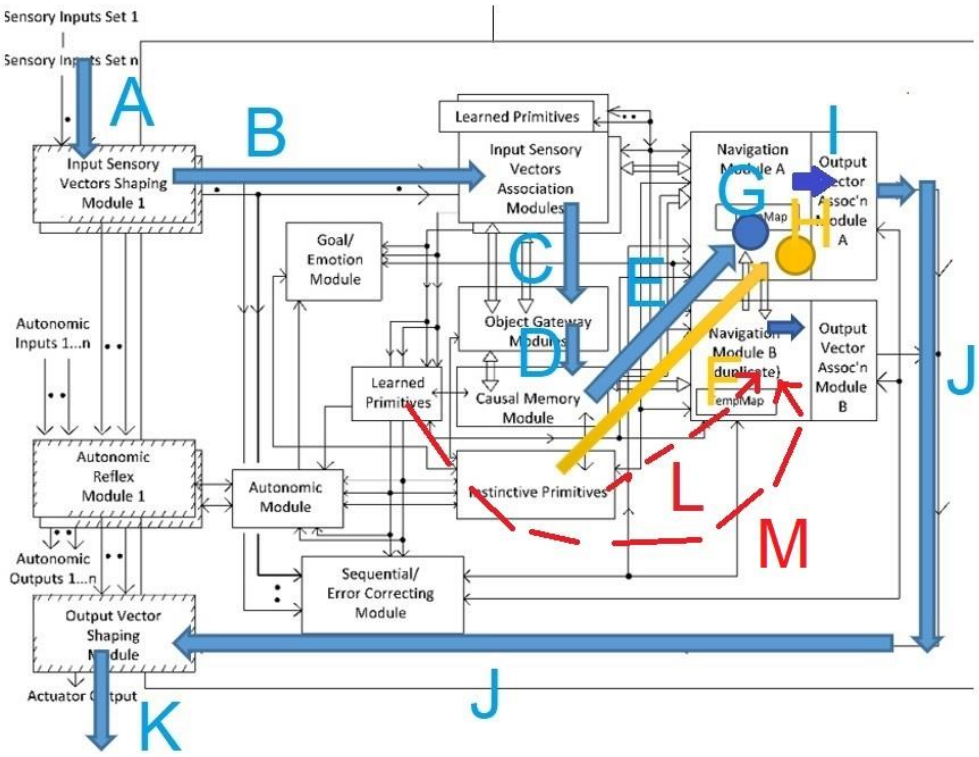

Figure 5 REPRODUCED FOR APPENDIX. A Cognitive Cycle in the CCA6 Architecture. Sensory inputs are processed by the various modules (described in text). The operation of the best matched instinctive primitive (circle H) on the Working Navigation Map (circle G) produces an output signal to the Output Vector Association Module A (arrow I) and then to the Output Vector Shaping Module (arrow J) and then an output to the external world which can activate an actuator or transmit an electronic signal. Then another cognitive cycle starts.
Arrow A – sensory inputs into the Input Sensory Vectors Shaping Modules
Arrow B – to the Input Sensory Vectors Association Modules
Arrow C – to the Object Gateway Modules
Arrow D – to the Causal Memory Module
Arrow E – best matched navigation map ("Working Navigation Map") to the Navigation Module A
Arrow F – best matched instinctive primitive to the Navigation Module A
Circle G – Working Navigation Map
Circle H – best matched instinctive primitive
Arrow I – to the Output Vector Association Module A
Arrow J – to the Output Vector Shaping Module
Arrow K – outputs to the external world (actuator, electronic communication signal))
Arrow L – instinctive primitives relating to compositionality and language tend to be applied to Navigation Module B
Arrow M – learned primitives relating to compositionality and language tend to be applied to Navigation Module B

A.2 The Input Sensory Vectors Association Modules

In Figure 5, arrow B shows normalized sensory inputs for each sensory system from the Input Sensory Vectors Shaping Modules to the Input Sensory Vectors Association Modules. The normalized sensory input arrays $\mathbf{S'}_{1,t}$ ... $\mathbf{S'}_{\Theta\_\sigma,t}$ are fed into corresponding modules (one module for each sensory system) of the Input Sensory Vectors

Association Modules. Each module will map the sensory inputs it is receiving into a navigation map. This navigation map is called a "local navigation map" $\mathbf{LNM}_{(\sigma,mapno)}$ (i.e., local navigation map **LNM** with address *mapno* in sensory system σ) (14). Each cell in three spatial dimensions in $\mathbf{LNM}_{(\sigma,\ mapno)}$ can represent the full contents of each cell, i.e., all the features, procedures, and link addresses associated with a cell. As (14) shows there is also a non-spatial dimension p which is used in implementations to store various non-spatial information.

$\boldsymbol{all\_maps}_{\sigma,t}$ is a vector holding all the local navigation maps in the σ sensory system Input Sensory Vectors Association Module (15). For example, $\boldsymbol{all\_maps}_{1,t}$ represents all the stored local navigation maps in the visual Input Sensory Vectors Association Module (which as noted above is $\sigma = 1$).

As noted above $\mathbf{S'}_{1,t}$ is an array of the visual processed inputs (i.e., $\sigma = 1$). $\mathbf{S'}_{2,t}$ are the auditory processed inputs since $\sigma = 2$, and so on. The visual processed inputs $\mathbf{S'}_{1,t}$ should be matched against $\boldsymbol{all\_maps}_{1,t}$, i.e., against all the other visual local navigation maps stored in the visual Input Sensory Vectors Association Module. The auditory, olfactory and any other sensory inputs should be matched against the respective local area maps stored in a particular Input Sensory Vectors Association Module.

In (18) `Input_Assocn_Mod`$_{\sigma}$`.match_best_local_navmap` is pseudocode that matches an incoming sensory array σ (e.g., if σ=1 then $\mathbf{S'}_{1,t}$ is an array of the visual processed inputs) against the respective stored local navigation maps $\boldsymbol{all\_maps}_{\sigma,t}$ (e.g., if σ=1 then $\boldsymbol{all\_maps}_{1,t}$ holds all the stored local navigation maps in the visual Input Sensory Vectors Association Module). $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$ represents the local navigation map **LNM** in sensory system σ with a *mapno* of Υ which is the best match of $\mathbf{S'}_{\sigma,t}$. For example, if sensory inputs array $\mathbf{S'}_{1,t}$ best matches to *mapno* 3456 (as an example) in the local navigation maps stored in the visual Input Sensory Vectors Association Module, then $\mathbf{LNM}_{(1,\ \Upsilon,t)}$ would be local navigation map 3456 in the visual Input Sensory Vectors Association Module.

Above it was noted that the Causal Cognitive Architecture 6 (CCA6) heavily makes use of feedback pathways—states of a downstream module can influence the recognition and processing of more upstream sensory inputs. Thus, the previous cognitive cycle's Working Navigation Map $\mathbf{WNM'}_{t-1}$ (the navigation map in the Navigation Module A in Figure 5) and on which the Navigation Module A can perform operations on) is used in the pseudocode method (18) in deciding which is a best match.

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture, although there is the allowance of multiple navigation modules in performing operations.

$$mapno = \text{map identification code} \in \mathrm{N} \quad (12)$$

$$\Theta = \text{total number of used local navigation maps in a sensory system } \boldsymbol{\sigma} \in \mathrm{N} \quad (13)$$

$$\mathbf{LNM}_{(\sigma,mapno)} \in \mathrm{R}^{mxnxoxp} \quad (14)$$

$$\boldsymbol{all\_maps}_{\sigma,t} = [\mathbf{LNM}_{(\sigma,1,t)}, \mathbf{LNM}_{(\sigma,2,t)}, \mathbf{LNM}_{(\sigma,3,t)}, \ldots, \mathbf{LNM}_{(\sigma,\ \Theta,\ t)}] \quad (15)$$

$$\Upsilon = \text{map number of best matching map in a given set of navigation maps} \in mapno \quad (16)$$

$$\mathbf{WNM'} = \in \mathrm{R}^{mxnxoxp} \quad \text{(17 and defined again below)}$$

$$\mathbf{LNM}_{(\sigma,\ \Upsilon,t)} = \texttt{Input\_Assocn\_Mod}_{\boldsymbol{\sigma}}\texttt{.match\_best\_local\_navmap}(\ \mathbf{S'}_{\sigma,t},\ \boldsymbol{all\_maps}_{\sigma,t},\ \mathbf{WNM'}_{t-1}) \quad (18)$$

$\mathbf{LNM}_{(\sigma,mapno)}$	a LNM or "local navigation map" which is a navigation map (i.e., an array of x,y,z dimensions m x n x o, which in the simulation in the paper are 6 x 6 x 6, corresponding to x,y,z axes of 6 x 6 x 6) that is held in the Input Sensory Vectors Association Module σ (e.g., if $\sigma = 1$, then that would be the module receiving the visual sensory inputs) and is map number *mapno* (there may be millions of other local navigation maps stored in that Input Sensory Vectors Association Module σ)
$\boldsymbol{all\_maps}_{\sigma,t}$	a vector holding all the LNMs in module σ (i.e., starting with the first LNM and going to the last utilized LNMΘ) e.g., $\boldsymbol{all\_maps}_{1,t}$ would be all the LNMs in the Input Sensory Vectors Association Module 1 which are all the local navigation maps created and stored from the visual sensory inputs

	-there are new sensory inputs each cognitive cycle, and thus new local navigation maps will be produced, and thus there is a t subscript representing a change with time
WNM'	a WNM is a "Working Navigation Map" it is a navigation map which for the moment is in the Navigation Module (Figure 5) and on which the Navigation Module can perform operations on As will defined below, operations can be performed by Navigation Module A or Navigation Module B, where are referred to as 'Navigation Module' in this section
`Input_Assocn_Mod`$_\sigma$`.match_best_local_navmap`(**S'**$_{\sigma,t}$, ***all_maps***$_{\sigma,t}$, **WNM'**$_{t-1}$)	pseudocode for an algorithm which will match the σ sensory inputs (e.g., if σ = 1 then it would be visual sensory inputs) against all the local navigation maps stored in the σ module of the Input Sensory Vectors Association Modules (Figure 5) and returns the best matched local navigation map **LNM** for that sensory module -note that the results of the Navigation Module in the previous cycle as represented by previous cognitive cycle's Working Navigation Map **WNM'**$_{t-1}$ are considered in the determining the best match
Υ	represents a map number of a navigation map in an Input Sensory Vectors Association Module
LNM$_{(\sigma, \Upsilon, t)}$	the local navigation map **LNM** stored in the Input Sensory Vectors Association Module σ with map number Υ which best matches in the incoming σ sensory inputs (e.g., if σ = 1 then it would be visual sensory inputs) e.g. if the visual sensory inputs best match the local navigation map #3456 in the visual module of the Input Sensory Vectors Association Modules, then at that moment **LNM**$_{(1, \Upsilon, t)}$ would be navigation map #3456 in the visual module

Table A3. Explanation of Symbols and Pseudocode in Equations (12) – (18)

At this point, in every sensory module in the Input Sensory Vectors Association Modules (Figure 5), there is a best matching **LNM**$_{(\sigma, \Upsilon, t)}$ ("local navigation map" since these navigation maps are stored locally in the sensory module rather than being stored in the Causal Memory Module attached to the Navigation Module, as seen in Figure 5). The next operation is to update the best matching local navigation map **LNM**$_{(\sigma, \Upsilon, t)}$ with the actual sensory inputs **S'**$_{\sigma,t}$ (21), creating an updated best matching navigation map **LNM'**$_{(\sigma, \Upsilon, t)}$ which is renamed (for simulation compatibility issues) to **LNM**$_{(\sigma, \Upsilon, t)}$ again.

If too many differences exist between the actual sensory inputs **S'**$_{\sigma,t}$ and the best matching navigation map **LNM**$_{(\sigma, \Upsilon, t)}$, then instead of updating the matched navigation map **LNM**$_{(\sigma, \Upsilon, t)}$, a new local navigation map **LNM**$_{(\sigma, new\_map, t)}$ is created and updated with the actual sensory inputs **S'**$_{\sigma,t}$ forming an updated best matching navigation map **LNM'**$_{(\sigma, \Upsilon, t)}$ (22), which is renamed (for simulation compatibility issues) to **LNM**$_{(\sigma, \Upsilon, t)}$ again.

In each sensory system Input Sensory Vectors Association Module the updated local navigation map (or newly created and updated one) is stored in the Input Sensory Vectors Association Module σ, and in future cognitive cycles, sensory inputs will be matched against it and the other local navigation maps stored there.

Vector ***lnm***$_t$ represents the best-matching and updated local navigation maps **LNM**$_{(\sigma, \Upsilon, t)}$ of all the different sensory modules of the Input Sensory Vectors Association Modules (23).

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture, although there is the allowance of multiple navigation modules in performing operations.

$$h = \text{number of differences allowed to be copied onto existing map} \in R \quad (19)$$

$$new\_map = \text{map number of new local navigation map added to current sensory system } \sigma \in mapno \quad (20)$$

$$| \texttt{Input\_Assocn\_Mod}_\sigma\texttt{.differences} (\mathbf{S'}_{\sigma,t}, \mathbf{LNM}_{(\sigma, \Upsilon, t)}) | \leq h, \Rightarrow \mathbf{LNM}_{(\sigma, \Upsilon, t)} = \mathbf{LNM'}_{(\sigma, \Upsilon, t)} = \mathbf{LNM}_{(\sigma, \Upsilon, t)} \cup \mathbf{S'}_{\sigma,t} \quad (21)$$

$$| \texttt{Input\_Assocn\_Mod}_\sigma\texttt{.differences} (\mathbf{S'}_{\sigma,t}, \mathbf{LNM}_{(\sigma, \Upsilon, t)}) | > h, \Rightarrow \mathbf{LNM}_{(\sigma, \Upsilon, t)} = \mathbf{LNM'}_{(\sigma, \Upsilon, t)} = \mathbf{LNM}_{(\sigma, new\_map, t)} \cup \mathbf{S'}_{\sigma,t} \quad (22)$$

$$\boldsymbol{lnm}_t = [\mathbf{LNM}_{(1,\,\Upsilon,t)}, \mathbf{LNM}_{(2,\,\Upsilon,t)}, \mathbf{LNM}_{(3,\,\Upsilon,t)}, \ldots, \mathbf{LNM}_{(\Theta\_\sigma,\,\Upsilon,t)}] \quad (23)$$

h	the number of differences allowed to be copied onto existing map – if there are too many differences between the best matching local navigation map retrieved from the sensory system's Input Sensory Vectors Association Module and the actual input sensory signal (i.e., $\mathbf{S'}_{\sigma,t}$) then rather than copying the information from input sensory array onto the best matching local navigation map, and simply makes $\mathbf{S'}_{\sigma,t}$ into a new local navigation map
new_map	the *mapno* of an empty local navigation map in a sensory system's Input Sensory Vectors Association Module used when making $\mathbf{S'}_{\sigma,t}$ into a new local navigation map
`Input_Assocn_Mod`$_\sigma$`.differences`	-pseudocode for an algorithm that calculates the differences between two navigation maps -used in (21) and (22) to calculate the differences between the input sensory signal $\mathbf{S'}_{\sigma,t}$ and the retrieved best matching local navigation map $\mathbf{LNM}_{(\sigma,\,\Upsilon,t)}$
$\mathbf{LNM}_{(\sigma,\,\Upsilon,t)} \cup \mathbf{S'}_{\sigma,t}$ → **LNM'** → **LNM**	update $\mathbf{LNM}_{(\sigma,\,\Upsilon,t)}$ with any new information in input sensory signal $\mathbf{S'}_{\sigma,t}$ (i.e., copy $\mathbf{S'}_{\sigma,t}$ onto $\mathbf{LNM}_{(\sigma,\,\Upsilon,t)}$) thereby creating an updated $\mathbf{LNM'}_{(\sigma,\,\Upsilon,t)}$ (for simulation compatibility issues renaming $\mathbf{LNM'}_{(\sigma,\,\Upsilon,t)}$ to $\mathbf{LNM}_{(\sigma,\,\Upsilon,t)}$ again)
$\mathbf{LNM}_{(\sigma,new\_map,t)} \cup \mathbf{S'}_{\sigma,t}$ → **LNM'** → **LNM**	copy $\mathbf{S'}_{\sigma,t}$ onto an empty $\mathbf{LNM}_{(\sigma,\,new\_map\,,t)}$ thereby creating an updated $\mathbf{LNM'}_{(\sigma,\,\Upsilon,t)}$ (for simulation compatibility issues renaming $\mathbf{LNM'}_{(\sigma,\,\Upsilon,t)}$ to $\mathbf{LNM}_{(\sigma,\,\Upsilon,t)}$ again)
$\boldsymbol{lnm}_t$	i.e., a vector holding $[\mathbf{LNM}_{(1,\,\Upsilon,t)}, \mathbf{LNM}_{(2,\,\Upsilon,t)}, \mathbf{LNM}_{(3,\,\Upsilon,t)}, \ldots, \mathbf{LNM}_{(\Theta\_\sigma,\,\Upsilon,t)}]$ -- the local navigation maps **LNM** of each sensory system which best matches the corresponding sensory input array $\mathbf{S'}_{\sigma,t}$ and then are updated to **LNM'** with the actual sensory information conveyed by $\mathbf{S'}_{\sigma,t}$ (for simulation compatibility issues renaming $\mathbf{LNM'}_{(\sigma,\,\Upsilon,t)}$ to $\mathbf{LNM}_{(\sigma,\,\Upsilon,t)}$ again)

Table A4. Explanation of Symbols and Pseudocode in Equations (19) – (23)

Input:	***s'(t)*** -- a vector holding all the normalized sensory input arrays $\mathbf{S'}_{1,t\,\ldots}\,\mathbf{S'}_{\Theta\,\sigma,t}$
Output:	$\boldsymbol{lnm}_t = [\mathbf{LNM}_{(1,\,\Upsilon,t)}, \mathbf{LNM}_{(2,\,\Upsilon,t)}, \mathbf{LNM}_{(3,\,\Upsilon,t)}, \ldots, \mathbf{LNM}_{(\Theta\_\sigma,\,\Upsilon,t)}]$ The local navigation maps **LNM** of each sensory system which best match the corresponding sensory input array $\mathbf{S'}_{\sigma,t}$, and then are updated to **LNM'** with the actual sensory information conveyed by $\mathbf{S'}_{\sigma,t}$ (for simulation compatibility issues renaming $\mathbf{LNM'}_{(\sigma,\,\Upsilon,t)}$ to $\mathbf{LNM}_{(\sigma,\,\Upsilon,t)}$ again)
Description:	-The normalized sensory input arrays are best matched with stored navigation maps in each sensory system (called a local navigation map **LNM**). -The best matched **LNM**s in each sensory system are then updated with the actual sensory information from the sensory input for that sensory system. -The updated best matched **LNM** is stored locally in that Input Sensory Vectors Shaping Module for matching against future sensory inputs. -The updated best matched **LNM** is output from this module; the best matched **LNM**s from each Input Sensory Vectors Shaping Module together make up vector $\boldsymbol{lnm}_t$ –the effective output.

Table A5. Summary of the Operation of the Input Sensory Vectors Associations Modules per Equations (12) – (23)

A.3 Data Structures in the CCA6

This section reviews a number of data structures used in the CCA6 architecture, most of them being compatible with the navigation map data structure. Equation (14) defines the local navigation map **LNM** as an array of four (or more) dimensions. There are three spatial dimensions (m,n,o in the definition representing x,y,z) as well as an extra dimension p for non-spatial information such storing which features belong to which objects, for storing meta-data, and so on.

Local navigation maps (i.e., stored locally in each sensory Input Sensory Vectors Association Module) were defined above in (14). Multisensory navigation maps **NM**, instinctive primitive navigation maps **IPM**, and learned primitive navigation maps **LPM** are similarly defined in (24). The multisensory navigation maps **NM** are stored in the Causal Memory Module (Figure 5) and contain visual, auditory, olfactory, etc. sensory features unlike the local navigation maps which have features from only one sensory system. The instinctive primitive navigation maps **IPM** and the

learned primitive navigation maps **LPM** have the same dimensional structure as other navigation maps, but they contain largely only procedures to perform on other navigation maps.

Equation (26) defines $\boldsymbol{all\_LNMs_t}$ as holding all the local navigation maps **LNM** in all the different Input Sensory Vectors Association Modules. Recall from above that $\boldsymbol{all\_maps_{\sigma,t}}$ holds all the local navigation maps **LNM** within a *given* Input Sensory Vectors Association Module. In (27) $\boldsymbol{all\_NMs_t}$ are defined as holding all the multisensory navigation maps **NM** in the Causal Memory Module (Figure 5). In (28) $\boldsymbol{all\_IPMs_t}$ are defined as holding all the instinctive primitive navigation maps **IPM** (or "instinctive primitives"—the procedures that are included with the architecture) in the Instinctive Primitives Module (Figure 5). In (29) $\boldsymbol{all\_LPMs_t}$ are defined as holding all the learned primitive navigation maps **LPM** (or "learned primitives"—the procedures that are learned by the architecture) in the Learned Primitives Module (Figure 5). And in (30) $\boldsymbol{all\_navmaps_t}$ are defined as holding all of these preceding navigation maps, i.e., $[\boldsymbol{all\_LNMs_t}, \boldsymbol{all\_NMs_t}, \boldsymbol{all\_IPMs_t}, \boldsymbol{all\_LPMs_t}]$. (The vector $\boldsymbol{all\_navmaps_t}$ does not hold the entirety of navigation maps in the CCA6 architecture as there are a number of other specialized navigation maps, particularly in the Sequential/Error Correcting Module.)

In (31–33) an addressing protocol is defined to address any particular cell within any particular navigation map within $\boldsymbol{all\_navmaps_t}$. For example, $\chi_{\text{modcode=*Causal\_Memory\_Mod*, } mapno=3456,x=2,y=3,z=4}$ is cell x=2,y=3,z=4 in map number 3456 in the Causal Memory Module.

In (34) a "*feature*" is defined as some arbitrary real number representing a feature modality and value. For example, $feature_{3,(\chi \text{modcode=*Causal\_Memory\_Mod*, } mapno=3456, \text{ x=2,y=3,z=4})}$ would be *feature* number 3 in the cell x=2,y=3,z=4 in map number 3456 in the Causal Memory Module. Its value, for example, could represent a visual line. In (35) similarly a "*procedure*" is defined as some arbitrary real number representing a procedure modality and value. For example, $procedure_{3,(\chi \text{modcode=*Causal\_Memory\_Mod*, } mapno=3456, \text{ x=2,y=3,z=4})}$ would be *procedure* number 3 in the cell x=2,y=3,z=4 in map number 3456 in the Causal Memory Module. Its value, for example, could represent a *procedure* (and required sub-procedures too; the value can have an unlimited number of digits) to move forward. In (36) a ***linkaddress*** is defined as $\boldsymbol{\chi'}$, i.e., pointing to $\boldsymbol{\chi'}$ which is some cell location in some navigation map in the architecture. The linkaddress provides a link between one cell in one map to another cell, possibly in the same map but often in a different navigation map. (Note: For stylistic reasons, in some places in the text "*features"* may be written which should be taken as the same variable as "*feature*".)

In (38) the equation defines $\boldsymbol{cellfeatures_{\chi,t}}$ as all the features within a given cell $\boldsymbol{\chi}$. In (39) the equation defines $\boldsymbol{cellprocedures_{\chi,t}}$ as all the procedures within a given cell $\boldsymbol{\chi}$. In (40) the equation defines $\boldsymbol{linkaddresses_{\chi,t}}$ as all the linkaddresses within a given cell $\boldsymbol{\chi}$. In (41) the equation defines $\boldsymbol{cellvalues_{\chi}}$ as all the values—the features, the procedures, the link addresses—held by a cell of a navigation map at location $\boldsymbol{\chi}$. The numbers in each cell can represent any collection of low-level sensory features, higher-level sensory features, procedures, and links.

As shown above, ***all_navmaps*** represents all the navigation maps in the architecture. In (41) it is seen that the value of a cell in one of the navigation maps in ***all_navmaps*** are its $\boldsymbol{cellvalues_{\chi,t}}$, i.e., the features, the procedures and the linkaddresses that the cell contains. Equation (42) shows that the pseudocode `link`($\boldsymbol{\chi}$,*t*) will return all the links that a cell at address $\boldsymbol{\chi}$ contains.

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture.

$$\mathbf{NM}_{mapno} \in R^{mxnxoxp}, \mathbf{IPM}_{mapno} \in R^{mxnxoxp}, \mathbf{LPM}_{mapno} \in R^{mxnxoxp} \quad (24)$$

$$\Theta\_NM = \text{total used NM's} \in N, \ \Theta\_IPM = \text{total used IPM's} \in N, \ \Theta\_LPM = \text{total used LPM's} \in N \quad (25)$$

$$\boldsymbol{all\_LNMs}_t = [\boldsymbol{all\_maps}_{1,t}, \boldsymbol{all\_maps}_{2,t}, \boldsymbol{all\_maps}_{3,t}, \ldots, \boldsymbol{all\_maps}_{\Theta\_\sigma,t}] \quad (26)$$

$$\boldsymbol{all\_NMs}_t = [\mathbf{NM}_{1,t}, \mathbf{NM}_{2,t}, \mathbf{NM}_{3,t}, \ldots, \mathbf{NM}_{\Theta\_NM,t}] \quad (27)$$

$$\boldsymbol{all\_IPMs}_t = [\mathbf{IPM}_{1,t}, \mathbf{IPM}_{2,t}, \mathbf{IPM}_{3,t}, \ldots, \mathbf{IPM}_{\Theta\_IPM,t}] \quad (28)$$

$$\boldsymbol{all\_LPMs}_t = [\mathbf{LPM}_{1,t}, \mathbf{LPM}_{2,t}, \mathbf{LPM}_{3,t}, \ldots, \mathbf{LPM}_{\Theta\_LPM,t}] \quad (29)$$

$$\boldsymbol{all\_navmaps}_t = [\boldsymbol{all\_LNMs}_t, \boldsymbol{all\_NMs}_t, \boldsymbol{all\_IPMs}_t, \boldsymbol{all\_LPMs}_t] \quad (30)$$

$$modcode = \text{module identification code} \in \mathrm{N} \quad (31)$$

$$\boldsymbol{mapcode} = [modcode, mapno] \quad (32)$$

$$\chi = [\boldsymbol{mapcode}, x, y, z] \quad (33)$$

$$feature \in \mathrm{R} \quad (34)$$

$$procedure \in \mathrm{R} \quad (35)$$

$$\boldsymbol{linkaddress}\ \chi' \in \chi \quad (36)$$

$\Phi\_feature$ = last *feature* contained by a cell, $\Phi\_procedure$ = last *procedure* contained by a cell, Φ_χ = last χ (i.e., address) contained by a cell (37)

$$\boldsymbol{cellfeatures}_{\chi,t} = [feature_{1,t}, feature_{2,t}, feature_{3,t}, \ldots, feature_{\Phi\_feature,t}] \quad (38)$$

$$\boldsymbol{cellprocedures}_{\chi,t} = [procedure_{1,t}, procedure_{2,t}, procedure_{3,t}, \ldots, procedure_{\Phi\_procedure,t}] \quad (39)$$

$$\boldsymbol{linkaddresses}_{\chi,t} = [\chi_{1,t}, \chi_{2,t}, \chi_{3,t}, \ldots, \chi_{\Phi\_\chi,t}] \quad (40)$$

$$\boldsymbol{cellvalues}_{\chi,t} = [\boldsymbol{cellfeatures}_{\chi,t}, \boldsymbol{cellprocedures}_{\chi,t}, \boldsymbol{linkaddresses}_{\chi,t}] \quad (41)$$

$$\boldsymbol{cellvalues}_{\chi,t} = \boldsymbol{all\_navmaps}_{\chi,t} \quad (42)$$

$$\boldsymbol{linkaddresses}_{\chi,t} = \texttt{link}(\chi,t) \quad (43)$$

LNM	local navigation map – one sensory system maps to any local navigation map different set of LNMs for each sensory system each set of LNMs stored in that particular sensory Input Sensory Vectors Association Module (Figure 5) array structure of dimensions m,n,o,p (x,y,z and non-spatial p dimension)
NM	multisensory navigation map – different sensory system features can be written to this map stored in the Causal Memory Module (Figure 5) array structure of dimensions m,n,o,p (x,y,z and non-spatial p dimension)
IPM	-instinctive primitive navigation map – "instinctive primitive" -primitives are procedures to perform on cells of other navigation maps -primitives are navigation maps mainly filled with procedures in their cells, but they can store features and linkaddresses in their cells as well -these primitives come with the architecture -stored in the Instinctive Primitives Module (Figure 5)

	-array structure of dimensions m,n,o,p (x,y,z and non-spatial p dimension)
LPM	-learned primitive navigation map – "learned primitive" -primitives are procedures to perform on cells of other navigation maps -primitives are navigation maps mainly filled with procedures in their cells, but they can store features and linkaddresses in their cells as well -these primitives are learned by the architecture -stored in the Learned Primitives Module (Figure 5) -array structure of dimensions m,n,o,p (x,y,z and non-spatial p dimension)
all_maps$_1$	vector of all of the LNMs in σ=1, i.e., visual Input Sensory Vectors Association Module (*note: 't' which indicates changes with time has been removed here and following entries for simplification purposes*) (***all_maps***$_2$ i.e., σ=2, is auditory module, and so on)
all_LNMs	vector of all of the LNMs in all of the Input Sensory Vectors Association Modules
all_NMs	vector of all of the NM's in the Causal Memory Module
all_IPMs	vector of all of the IPM's in the Instinctive Primitives Module
all_LPMs	vector of all of the LPM's in the Learned Primitives Module
all_navmaps	vector of all of the LNMs, NM's, IPM's, LPM's in the architecture
mapcode	[module identification code, map number] points to a particular navigation map among all the LNMs, NM's, IPM's, and LPM's in the architecture
χ	[mapcode, x, y, z] points to particular cell (x,y,z) in a particular navigation map ([module identification code, map number])
feature *features*	arbitrary real number representing a feature modality and value within a cell χ (Note: For stylistic reasons, in some places the text may write "*features"* which should be taken as the same variable as "*feature*".)
procedure	arbitrary real number representing a procedure modality and value within a cell χ
linkaddress	address within a cell χ pointing to another cell (possibly in another navigation map) χ**'**
cellfeatures$_\chi$	vector of all the features within a cell χ
cellprocedures$_\chi$	vector of all the procedures within a cell χ
linkaddresses$_\chi$	vector of all the linkaddress within a cell χ
cellvalues$_\chi$	vector of all of the features, procedures and linkaddresses within a cell χ
all_navmaps$_\chi$	vector of all of the features, procedures and linkaddresses within a cell χ note that the value of some cell χ in some cell in some navigation map (i.e., within the collection of ***all_navmaps***) = ***cellvalues***$_\chi$
`link`(χ,*t*)	pseudocode that returns all the linkaddresses within a cell χ , i.e., same value as ***linkaddresses***$_\chi$

Table A6. Explanation of Symbols and Pseudocode in Equations (24) – (43)

Input:	*not applicable – definitions of data structures in this section*
Output:	*not applicable – definitions of data structures in this section*
Description:	-In this section some of the key data structures utilized by the architecture are discussed. -These data structures are largely based on the "navigation map" which maps spatial features (e.g., are there pixels representing ground at this x,y,z coordinate?), potential procedures (e.g., do something with the data in this or other cells of the navigation map), and link addresses (e.g., possibly go to the cell in possibly another navigation map specified by the link address) at a particular x,y,z coordinate ("cell" or "cube"). -The architecture in conjunction with the navigation map data structure allows a solution to the classical binding problem.

Table A7. Summary of the Operation of Equations (24) – (43)

A.4 The Sequential/Error Correcting Module

Consider the sensory inputs through the CCA6 architecture as they become temporally bound. The Sequential/Error Correcting Module plays a key role in temporally binding the sensory inputs.

Normalized sensory input arrays **S'**$_{\sigma,t}$ represented by ***s'(t)*** (from the output of the Input Sensory Vectors Shaping Modules (10, 11)) feed into the Sequential/Error Correcting Module. After processing these signals the Sequential/Error Correcting Module sends a "motion prediction vector" to the Navigation Module. The motion

prediction vector allows changes in an object(s) in a navigation map to be represented much as other spatial features on a navigation map.

In (44) $\boldsymbol{s'\_series}_t$ is a time series of the input sensory vector $\boldsymbol{s'(t)}$. In (45) and (46) the visual and auditory sensory times series are extracted from $\boldsymbol{s'\_series}_t$ and stored respectively as $\boldsymbol{visual\_series}_t$ and $\boldsymbol{auditory\_series}_t$. Although it is possible to create motion prediction vectors for all the senses, in the current simulation this is only done so for the visual and auditory sensory systems.

In (47) the pseudocode `Sequential_Mod.visual_match()` matches $\boldsymbol{visual\_series}_t$ with visual time series stored in the Sequential/Error Correcting Module. If there is a reasonable match with few differences, then the matched time series will be updated with the new information. If there is no close enough match, then a new times series will be stored in the Sequential/Error Correcting Module. A motion prediction vector $\boldsymbol{visual\_motion}_t$ is then computed from visual time series data (47). A similar process occurs in computing the motion prediction vector $\boldsymbol{auditory\_motion}_t$ (48). These motion prediction vectors are then stored much like any spatial feature in a navigation map called the Vector Navigation Map $\mathbf{VNM''}_t$ (50a, 50b). $\mathbf{VNM''}_t$ is then propagated to the Navigation Module complex (Figure 5).

In (51) the pseudocode `Sequential_Mod.auditory_match_process()` extracts sound patterns from $\boldsymbol{auditory\_series}_t$, and stores these patterns spatially in a navigation map $\mathbf{AVNM}_t$. Then $\mathbf{AVNM}_t$ is propagated to the Navigation Module A (Figure 5).

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture, other than distinguishing the two Navigation Modules as mentioned in the text.

$$\boldsymbol{s'\_series}_t = [\boldsymbol{s'(t-3)}, \boldsymbol{s'(t-2)}, \boldsymbol{s'(t-1)}, \boldsymbol{s'(t)}] \quad (44)$$

$$\boldsymbol{visual\_series}_t = \texttt{Sequential\_Mod.visual\_inputs}(\boldsymbol{s'\_series}_t) \quad (45)$$

$$\boldsymbol{auditory\_series}_t = \texttt{Sequential\_Mod.auditory\_inputs}(\boldsymbol{s'\_series}_t) \quad (46)$$

$$\boldsymbol{visual\_motion}_t = \texttt{Sequential\_Mod.visual\_match}(\boldsymbol{visual\_series}_t) \quad (47)$$

$$\boldsymbol{auditory\_motion}_t = \texttt{Sequential\_Mod.auditory\_match}(\boldsymbol{auditory\_series}_t) \quad (48)$$

$$\mathbf{VNM} \in R^{mxnxoxp}, \ \mathbf{AVNM} \in R^{mxnxoxp} \quad (49)$$

$$\mathbf{VNM'}_t = \mathbf{VNM}_t \cup \boldsymbol{visual\_motion}_t \quad (50a)$$

$$\mathbf{VNM''}_t = \mathbf{VNM'}_t \cup \boldsymbol{auditory\_motion}_t \quad (50b)$$

$$\mathbf{AVNM}_t = \texttt{Sequential\_Mod.auditory\_match\_process}(\boldsymbol{auditory\_series}_t) \quad ((51)$$

In the next section it will be seen that the Object Segmentation Gateway Module (Figure 5) will segment a sensory scene into objects of interest. The individual objects segmented in the sensory scene, as well as the entire scene itself treated as one composite object, will then trigger similar navigation maps in the Causal Memory Module to be retrieved and moved to the Navigation Module A. For example, a visual scene of a river, a rock and a leaf floating in the river, might be segmented in the river object, the rock object and the leaf object. In order to obtain the motion information about a segmented object, each segmented object navigation map must be sent to the Sequential/Error Correcting Module.

In (52) a navigation map called the Visual Segmented Navigation Map **VSNM** is defined. $\mathbf{VSNM}_{i,t-3}$, $\mathbf{VSNM}_{i,t-2}$, $\mathbf{VSNM}_{i,t-1}$, and $\mathbf{VSNM}_{i,t}$ containing a visual segmented object on a navigation map at different time intervals, are sent from the Object Segmentation Gateway Module to the Sequential/Error Correcting Module where they are stored in vector $\boldsymbol{visual\_segmented\_series}_{i,t}$ (53). There may be several **VNSM**'s produced for one sensory scene in the Object Segmentation Gateway Module, e.g., a sensory scene of a river with a rock and leaf floating in, it will produce a **VSNM** for the river, for the rock and for the leaf. Hence, the use of subscript *i* to refer to a particular $\mathbf{VSNM}_i$ for the sensory scene in a cognitive cycle.

The same pseudocode used in (47) is used again in (54) but this time on ***visual_segmented_series***$_{i,t}$ and produces ***visseg_motion***$_{i,t}$ which is a motion prediction vector for the motion information, if any, in ***visual_segmented_series***$_{i,t}$. This motion prediction vector is copied, like any spatial feature, onto the original navigation map **VSNM**$_{i,t}$ and the updated Visual Segmented Navigation Map **VSNM**'$_{i,t}$ is then sent back to the Object Segmentation Gateway Module/Navigation Module (55).

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture, other than distinguishing the two Navigation Modules as mentioned in the text.

$$\mathbf{VSNM} \in R^{mxnxoxp} \quad (52)$$

$$\textbf{\textit{visual\_segmented\_series}}_{i,t} = [\mathbf{VSNM}_{i,t\text{-}3}, \mathbf{VSNM}_{i,t\text{-}2}, \mathbf{VSNM}_{i,t\text{-}1}, \text{ and } \mathbf{VSNM}_{i,t}] \quad (53)$$

$$\textbf{\textit{visseg\_motion}}_{i,t} = \texttt{Sequential\_Mod.visual\_match}(\textbf{\textit{visual\_segmented\_series}}_{i,t}) \quad (54)$$

$$\mathbf{VSNM'}_{i,t} = \mathbf{VSNM}_{i,t} \cup \textbf{\textit{visseg\_motion}}_{i,t} \quad (55)$$

s'(t)	-vector representing the normalized sensory input sensory arrays of the different sensory systems $\mathbf{S'}_{\sigma,t}$ produced by the Input Sensory Vectors Shaping Module (Figure 5) -sent to both Input Sensory Vectors Association Modules and to the Sequential/Error Correcting Module (Figure 5)
s'(t-1)	***s'(t)*** value in the previous cognitive cycle, i.e., t-1
s'_series$_t$	a time series of ***s'*** at time t, time t-1, time t-2 (two cognitive cycles ago) and time t-3 = [***s'(t-3), s'(t-2), s'(t-1), s'(t)***]
`Sequential_Mod.visual_inputs`(***s'_series***$_t$) → ***visual_series***$_t$	-pseudocode for an algorithm that extracts the visual sensory normalized inputs, i.e., $\mathbf{S'}_{1,t}$ (σ =1 for visual system) from ***s'(t)*** -produces ***visual_series***$_t$ as its output, which essentially is [$\mathbf{S'}_{1,t}$, $\mathbf{S'}_{1,t\text{-}1}$, $\mathbf{S'}_{1,t\text{-}2}$, $\mathbf{S'}_{1,t\text{-}3}$]
`Sequential_Mod.auditory_inputs`(***s'_series***$_t$) → ***auditory_series***$_t$	-pseudocode for an algorithm that extracts the auditory sensory normalized inputs, i.e., $\mathbf{S'}_{2,t}$ (σ =2 for auditory system) from ***s'(t)*** -produces ***auditory_series***$_t$ as its output, which is [$\mathbf{S'}_{2,t}$, $\mathbf{S'}_{2,t\text{-}1}$, $\mathbf{S'}_{2,t\text{-}2}$, $\mathbf{S'}_{2,t\text{-}3}$]
`Sequential_Mod.visual_match`(***visual_series***$_t$) → ***visual_motion***$_t$	-pseudocode for an algorithm that matches ***visual_series***$_t$ with visual time series stored in the Sequential/Error Correcting Module (Figure 5) -a best match is chosen (or if no matches close enough then ***visual_series***$_t$ used itself as the best match -the best match is then updated from ***visual_series***$_t$ (much in the same manner as for example the matching and updating of navigation maps illustrated above in Figures 4 and 5) -the updated best match is stored in the Sequential/Error Correcting Module for future matching -the updated best match is then transformed into a motion prediction vector (i.e., showing and predicting the motion of the object) is output as ***visual_motion***$_t$
`Sequential_Mod.auditory_match`(***auditory_series***$_t$) → ***auditory_motion***$_t$	-pseudocode for an algorithm that matches ***auditory_series***$_t$ with auditory time series stored in the Sequential/Error Correcting Module (Figure 5) -a best match is chosen (or if no matches close enough then ***auditory_series***$_t$ used itself as the best match -the best match is then updated from ***auditory_series***$_t$ (much in the same manner as for example the matching and updating of navigation maps illustrated above in Figures 4 and 5) -the updated best match is stored in the Sequential/Error Correcting Module for future matching -the updated best match is then transformed into a motion prediction vector (i.e., showing and predicting the motion of the object) and is output as ***auditory_motion***$_t$
VNM	a "Vector Navigation Map" is just another ordinary navigation map used to store the motion prediction vectors

AVNM	an "Audio Vector Navigation Map" is just another ordinary navigation map used to store more detailed motion prediction vectors about the sound patterns, useful for advanced analysis of sound patterns in perceiving the environment and for language
VNM$_t$ ∪ ***visual_motion***$_t$ → **VNM'**$_t$	the ***visual_motion***$_t$ motion prediction vector is copied to, i.e., stored, in the Vector Navigation Map **VNM** just like any other spatial feature – binding of temporal features as a spatial feature occurs here **VNM'**$_t$ is the updated **VNM** (essentially a new navigation map has ***visual_motion***$_t$ copied to it)
VNM'$_t$ ∪ ***auditory_motion***$_t$ → **VNM''**$_t$	-the ***auditory_motion***$_t$ motion prediction vector is copied to, i.e., stored, in the Vector Navigation Map **VNM'** just like any other spatial feature – binding of temporal features as a spatial feature occurs here again **-VNM''**$_t$ is the updated **VNM'** (essentially **VNM''**$_t$ is a navigation map which has ***visual_motion***$_t$ and ***auditory_motion***$_t$ copied to it – vectors respectively showing the previous and predicted motion of an object based on visual sensory inputs and based on auditory sensory inputs) -(note: in the simulation at present if there are different ***visual_motion***$_t$ and ***auditory_motion***$_t$ vectors, i.e., pointing in different directions, then the better quality result is used as a sole result in upstream algorithms in the Navigation Module; in the future more sophisticated algorithms can be used to take advantage of these motion prediction vectors)
`Sequential_Mod.auditory_match_process` (***auditory_series***$_t$) → → **AVNM**$_t$	-pseudocode for an algorithm that matches ***auditory_series***$_t$ with auditory time series stored in the Sequential/Error Correcting Module (Figure 5) -a best match is chosen (or if no matches close enough then ***auditory_series***$_t$ used itself as the best match -the best match is then updated from ***auditory_series***$_t$ (much in the same manner as for example the matching and updating of navigation maps illustrated above in Figures 4 and 5) -the updated best match is stored in the Sequential/Error Correcting Module for future matching -the updated best match is then transformed into a complex advanced motion prediction vector (i.e., showing in more detail the pattern of the auditory sensations) and is output as **AVNM**$_t$ (which as described above is an "Audio Vector Navigation Map")
VSNM$_{i,t}$	-a "Visual Segmentation Navigation Map" **VSNM** which is an ordinary navigation map with visually segmented information stored on it -the Object Segmentation Gateway Module (Figure 5) will "segment" a sensory scene into objects of interest (described in more detail in the following section) -each segmented object is treated as a separate navigation map (there is also a navigation map of all the objects together on the navigation map) -the subscript *i* refers to multiple **VSNM**'s that can be created for a given sensory scene -the subscript *t* refers to time since the values change each cognitive cycle -it is useful to calculate motion prediction vectors, if they exist (often they will not) for each segmented object (e.g., if there is a river object, a rock object and leaf floating in the river object, then it is very useful to segment the sensory scene into these objects (i.e., river, rock and leaf) and a motion prediction vector to show the motion of the leaf is very useful – thus there will be **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock and **VSNM**$_{3,t}$ for leaf, in this example) -only visual motion is calculated for segmented objects (auditory motion and motion of other senses is not calculated in the current simulation, but could be in the future, for example, the motion of a radar signal in a future embodiment)
visual_segmented_series$_{i,t}$	-a time series of the current **VSNM**$_{i,t}$ and **VSNM**$_i$ from one cognitive cycle ago (i.e., t-1), two cycles ago (i.e., t-2) and three cycles ago (i.e., t-3) = [**VSNM**$_{i,t-3}$, **VSNM**$_{i,t-2}$,**VSNM**$_{i,t-1}$, and **VSNM**$_{i,t}$] -if there were three **VSNM**'s produced from scene (such as the example of the river with the rock and the leaf floating in it) then there would be three different ***visual_segmented_series***$_{i,t}$ produced, with i=1, i=2, and i=3
`Sequential_Mod.visual_match`(***visual_segmented_series***$_{i,t}$) → ***visseg_motion***$_{i,t}$	-pseudocode for an algorithm that matches ***visual_segmented_series***$_{i,t}$ with visual time series stored in the Sequential/Error Correcting Module (Figure 5) -a best match is chosen (or if no matches close enough then ***visual_segmented_series***$_{i,t}$ is used itself as the best match

	-the best match is then updated from ***visual_segmented_series***$_{i,t}$ (much in the same manner as for example the matching and updating of navigation maps illustrated above in Figures 4 and 5) -the updated best match is stored in the Sequential/Error Correcting Module for future matching -the updated best match is then transformed into a motion prediction vector (i.e., showing and predicting the motion of the object) and is output as ***visseg_motion***$_{i,t}$
VSNM$_{i,t}$ ∪ ***visseg_motion***$_{i,t}$ → **VSNM'**$_{i,t}$	-**VSNM**$_{i,t}$ was previously received from the Object Segmentation Gateway Module (Figure 5) containing the spatial information about an object (e.g., it could be the leaf mentioned above floating in the river, and shown in Figure 7) (also received and stored were [**VSNM**$_{i,t-3}$, **VSNM**$_{i,t-2}$,**VSNM**$_{i,t-1}$, and **VSNM**$_{i,t}$] so that a motion prediction vector could be calculated) -in this step the motion prediction vector ***visseg_motion***$_{i,t}$ about the object (e.g., perhaps the leaf in the example above) in this **VSNM**$_t$ navigation map is copied to the **VSNM**$_{i,t}$ navigation map–binding of temporal features as a spatial feature occurs here for this segmented object -**VSNM'**$_{i,t}$ is produced in this operation (i.e., the original **VSNM**$_{i,t}$ plus the motion prediction vector (***visseg_motion***$_{i,t}$) and is sent back to Object Segmentation Gateway Module/Navigation Module (Figure 5) -if there were three **VSNM**'s produced from scene (such as the example of the river with the rock and the leaf floating in it) then there would be three different **VSNM'**$_{i,t}$ produced and returned to the Object Segmentation Gateway Module/Navigation Module (Figure 5), with i=1, i=2, and i=3

Table A8. Explanation of Symbols and Pseudocode in Equations (44) – (55)

Input:	- ***s'(t)*** is a vector representing all the normalized sensory input arrays **S'**$_{1,t}$ … **S'**$_{\Theta\_\sigma,t}$ (**s'**(t-1), **s'**(t-2), **s'**(t-3) also from t-1, t-2 and t-3 previous cognitive cycles are stored so that a time series can be processed) -**VSNM**$_{i,t}$ is one of several **VSNM** navigation maps propagated from the Object Segmentation Gateway Module containing visual segments (hence the name **VSNM**), i.e., segments of the sensory scene recognized as distinct objects -e.g., if the sensory scene was a river with a rock and a leaf floating down a river, a **VSNM**$_{1,t}$ could contain the river, a**VSNM**$_{2,t}$ could contain the rock, and a **VSNM**$_{3,t}$ could contain the leaf – they are being sent to the Sequential/Error Correcting Module for temporal binding of motion, i.e., for insertion of a motion prediction vector if the object is moving
Output:	-**VNM''**$_t$ is a navigation map binding visual motion and auditory motion of the sensory scene) propagated to the Navigation Module A complex (Figure 5) -**AVNM**$_t$ is a navigation map binding advanced auditory patterns propagated to the Navigation Module A complex (Figure 5) (useful for auditory analysis as in better recognition of the environment and language) -**VSNM'**$_{i,t}$ is the original **VSNM**$_{i,t}$ from the Object Segmentation Gateway Module with a motion prediction vector added if the object was moving (or else unchanged if there was no movement), and it is then returned to the Object Segmentation Gateway Module/Navigation Module A (Figure 5)
Description:	-If there is movement in a sensory scene rather than requiring thirty versions of the sensory scene per second, only a single version is required but a motion prediction vector is added to show motion that has occurred and is still predicted to occur. -**VNM''**$_t$ is a navigation map produced with motion prediction vectors showing any motion with regard to overall visual and sound features in the sensory scene. -**AVNM**$_t$ is a navigation map produced with multiple motion prediction vectors showing more detailed and advanced sound patterns useful for environment recognition and spoken language. -**VSNM'**$_{i,t}$ is a navigation map for a given object (or "segment") of the sensory scene with a motion prediction vector added if there is motion of the object. There can be several (i.e., i=1, i=2, and so on) **VSNM'**$_{i,t}$ navigation maps for any given scene. -**VNM''**$_t$, **AVNM**$_t$, and **VSNM'**$_{i,t}$ are sent to the Object Segmentation Gateway Module/Navigation Module (Figure 5). -This module allows a solution to the temporal binding.

	-Although the examples such as river, rock and leaf floating in it, are very concrete ones, there can be binding of motion of more abstract concepts by the same mechanisms described above.

Table A9. Summary of the Operation of the Sequential/Error Correcting Module per Equations (44) – (55)

A.5 The Object Segmentation Gateway Module

At this point the sensory inputs have been transformed into best matching and updated visual, auditory, and olfactory local navigation maps (represented by the vector $\boldsymbol{lnm}_t$). The local navigation maps **LNM**s are propagated to the Object Segmentation Gateway Module. The complex of three tightly connected modules—the Object Segmentation Gateway Module, the Navigation Module and the Causal Memory Module will transform the input sensory data into a Working Navigation Map **WNM** upon which instinctive and learned primitives (i.e., essentially small algorithms) can act and possibly produce an action output.

The Object Segmentation Gateway Module will attempt to segment each sensory scene into different objects. In the current version of the CCA6 it is only performed visually—the module attempts to recognize coherent visual shapes in the sensory scene. (In theory, this could be done with multiple senses, and may be done in future versions of the architecture.)

Continuing the example given above of a sensory scene with only visual stimuli– a river with a rock and leaf floating in the river, the leaf, the rocks and the river are recognized by the Object Segmentation Gateway Module and segmented as separate objects each on their own **VSNM** navigation map (while at the same time keeping the entire scene and all objects on another navigation map). There will be an attempt to calculate motion prediction vectors for each different object's navigation map, but only the leaf will have a motion prediction vector since its position is changing in this example. Note that the use of labels such as 'river', 'rocks', 'leaf' is for the benefit of the reader. The CCA6 does not have a full English language implemented at this time, and uses its own internal labels. As well, current recognition is from a catalog of objects and simulated—the architecture has little deep understanding of the objects.

The "Visual Segmentation Navigation Map" $\mathbf{VSNM}_{i,t}$ (52) is an ordinary navigation map with visually segmented information stored on it, i.e., each object such as the leaf floating in the river, the rocks, and the river of the example get put onto a separate $\mathbf{VSNM}_i$ navigation map. For example, $\mathbf{VSNM}_{1,t}$ for river, $\mathbf{VSNM}_{2,t}$ for one of the group of rocks, and perhaps $\mathbf{VSNM}_{3,t}$ for the leaf. (There is a subscript *t* since the values of these navigation maps change each cognitive cycle.)

The pseudocode/algorithm `Object_Seg_Mod.visualsegment` in equation (61) takes the best-matching visual input sensory local navigation map $\mathbf{LNM}_{(1,\ \Upsilon,t)}$ and segments it into whatever objects it can find in its local memory stores or that meet certain algorithmic criteria (e.g., pixels separate from other pixels, and so on). The navigation maps $\mathbf{VSNM}_{t,i=1..\Theta\_i}$ (e.g., in the example above of the river, group or rocks and leaf producing $\mathbf{VSNM}_{1,t}$ for river, $\mathbf{VSNM}_{2,t}$ for one of the group of rocks, and perhaps $\mathbf{VSNM}_{3,t}$ for the leaf) are produced as a result. These navigation maps $\mathbf{VSNM}_{t,i=1..\Theta\_i}$ are sent to Sequential/Error Correcting Module where according to equations (52–55) attempts are made to see if there is motion occurring (via time series of **VSNM**'s from different cognitive cycles) and if so a motion prediction vector is bound onto the particular **VSNM** navigation map, which is returned back to the Object Segmentation Gateway.

One of the arguments to pseudocode/algorithm `Object_Seg_Mod.visualsegment` (61) is as noted above the best-matching visual input sensory local navigation map $\mathbf{LNM}_{(1,\ \Upsilon,t)}$. This **LNM** is actually transmitted in parallel to the Object Segmentation Gateway Module along with the other **LNM**s from other sensory system. Having the different sensory systems' best-matching **LNM**s via vector $\boldsymbol{lnm}$, and in (56) it is trivial to extract $\mathbf{LNM}_{(1,\ \Upsilon,t)}$. Another argument is the contextual value **CONTEXT**, which actually is a navigation map, and will help influence which objects are detected. In (57–59) it is seen to be set to the value of the previous cognitive cycle's Working Navigation Map **WNM'**, which effectively is the previous cognitive cycle's action taken or intermediate results. The contextual value will help influence which objects are detected. The third argument to (61) is $\mathbf{VNM''}_t$. As shown above in the Sequential/Error Correcting Module, $\mathbf{VNM''}_t$ shows the main visual motion and sound motion of the overall sensory scene. The vector navigation map $\mathbf{VNM''}_t$ helps influence segmentation in (61) by different motions and sound production. In (62) $\mathbf{LNM}_{(1,\ \Upsilon,t)}$ is updated to $\mathbf{LNM'}_{(1,\ \Upsilon,t)}$ with the information (i.e., motion prediction vectors, which does include visual and sensory motion of the main object in the scene) from $\mathbf{VNM''}_t$.

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture.

$$\mathbf{LNM}_{(1,\ \Upsilon,t)} = \boldsymbol{lnm}_t[0] \quad (56)$$

$$\mathbf{CONTEXT} = \in R^{mxnxoxp} \quad (57)$$

$$\mathbf{WNM'} = \in R^{mxnxoxp} \quad (58)$$

$$\mathbf{CONTEXT}_t = \mathbf{WNM'}_{t-1} \quad (59)$$

$$\Theta\_i = \text{total objects segmented in this sensory scene} \in N \quad (60)$$

$$\mathbf{VSNM}_{t,i=1..\Theta\_i} = \texttt{Object\_Seg\_Mod.visualsegment}(\ \mathbf{LNM}_{(1,\Upsilon,t)}, \mathbf{CONTEXT}_t, \mathbf{VNM''}_t\ ) \quad (61)$$

$$\mathbf{LNM'}_{(1,\Upsilon,t)} = \mathbf{LNM}_{(1,\Upsilon,t)} \cup \mathbf{VNM''}_t \quad (62)$$

$\mathbf{LNM}_{(\sigma,\Upsilon,t)}$ Equation (18) described in an earlier section	the local navigation map **LNM** stored in the Input Sensory Vectors Association Module σ with map number Υ which best matches in the incoming σ sensory inputs (e.g., if $\sigma = 1$ then it would be visual sensory inputs) e.g., if the visual sensory inputs best match the local navigation map #3456 in the visual module of the Input Sensory Vectors Association Modules, then at that moment $\mathbf{LNM}_{(1,\Upsilon,t)}$ would be navigation map #3456 in the visual module
$\mathbf{LNM}_{(\sigma,\Upsilon,t)} \cup \mathbf{S'}_{\sigma,t}$ → **LNM** Equation (21) described previously	as shown previously: update $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$ with any new information in input sensory signal $\mathbf{S'}_{\sigma,t}$ (i.e., copy $\mathbf{S'}_{\sigma,t}$ onto $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$) thereby creating an updated $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$
$\mathbf{LNM}_{(\sigma,new\_map,t)} \cup \mathbf{S'}_{\sigma,t}$ → **LNM** Equation (22) described previously	OR copy $\mathbf{S'}_{\sigma,t}$ onto an empty $\mathbf{LNM}_{(\sigma,new\_map,t)}$ thereby creating an updated $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$
$\boldsymbol{lnm}_t$ Equation (23) described previously	i.e., a vector holding [$\mathbf{LNM}_{(1,\Upsilon,t)}$, $\mathbf{LNM}_{(2,\Upsilon,t)}$, $\mathbf{LNM}_{(3,\Upsilon,t)}$, …, $\mathbf{LNM}_{(\Theta\_\sigma,\Upsilon,t)}$] -- the local navigation maps **LNM** of each sensory system which best match the corresponding sensory input array $\mathbf{S'}_{\sigma,t}$ and then are updated to **LNM** with the actual sensory information conveyed by $\mathbf{S'}_{\sigma,t}$
$\mathbf{LNM}_{(1,\Upsilon,t)}$	from the definition of $\boldsymbol{lnm}_t$ this is just the first member of $\boldsymbol{lnm}_t$ which is $\boldsymbol{lnm}_t[0]$ (indexing starts from 0 in this array) which is the **LNM** for the visual sensory inputs
WNM'	-Working Navigation Map -the current Working Navigation Map $\mathbf{WNM'}_t$ is the navigation map which the Navigation Module A focuses its attention on, i.e., applies operations on to make a decision to take some sort or no action -as will be seen in the following sections, a navigation map can be assigned as being the $\mathbf{WNM'}_t$ for that cognitive cycle, and then a different one in the next cognitive cycle, and so on, depending on the sensory information being processed and the results obtained
CONTEXT	-a normal navigation map (i.e., same dimensions as the other navigation maps in the architecture) used to hold contextual information which help influence which objects are detected -at present it is set to the value of the previous (i.e., in the previous cognitive cycle) Working Navigation Map $\mathbf{WNM'}_{t-1}$
VNM Equation (49) described previously	a "Vector Navigation Map" is just another ordinary navigation map used to store the motion prediction vectors
$\mathbf{VNM''}_t$ Equations (50a, 50b) described previously	as described previously in the Sequential/Error Correcting Module: -essentially $\mathbf{VNM''}_t$ is a navigation map which has $\boldsymbol{visual\_motion}_t$ and $\boldsymbol{auditory\_motion}_t$ copied to it – vectors respectively showing the previous and predicted motion of an object based on visual sensory inputs and based on auditory sensory inputs

	-tends to refer to motion of the overall scene or major object rather than the segmented objects in the sensory scene
VSNM$_{i,t}$ → **VSNM'**$_{i,t}$ (Note: **VSNM**$_{i,t}$ is first created in the Object Segmentation Gateway Module, and then updated to **VSNM'**$_{i,t}$ in the Sequential/Error Correcting Module)	-a "Visual Segmentation Navigation Map" **VSNM** which is an ordinary navigation map with visually segmented information stored on it -the Object Segmentation Gateway Module (xxx~~Figures 9, 10)~~ will "segment" a sensory scene into objects of interest -each segmented object is treated as a separate navigation map (there is also a navigation map of all the objects together on the navigation map) -the subscript *i* refers to multiple **VSNM**'s that can be created for a given sensory scene -the subscript *t* refers to time since the values change each cognitive cycle -it is useful to calculate motion prediction vectors, if they exist (often they will not) for each segmented object (e.g., if there is a river object, a rock object and leaf floating in the river object, then it is very useful to segment the sensory scene into these objects (i.e., river, rock and leaf) and a motion prediction vector to show the motion of the leaf is very useful – thus there will be **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock and **VSNM**$_{3,t}$ for leaf, in this example) -continuing this example: **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock and **VSNM**$_{3,t}$ for leaf will be propagated to the Sequential/Error Correcting Module for computation of motion prediction vectors for each of these **VSNM**'s -continuing this example: a motion prediction vector is computed and stored on **VSNM**$_{3,t}$ for leaf, but **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock did not have enough motion for such vectors and are unchanged -continuing this example: **VSNM'**$_{1,t}$ for river (unchanged), **VSNM'**$_{2,t}$ for rock (unchanged) and **VSNM'**$_{3,t}$ for leaf (motion prediction vector added) are then returned back to the Object Segmentation Gateway Module
VSNM$_{t,i=1..\Theta\_i}$	-all the Visual Segmentation Navigation Maps **VSNM**'s created for a sensory scene -for example, in the example just given there would be **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock and **VSNM**$_{3,t}$ for leaf (xxx~~for the actual example in Figure 10 there would be additional **VSNM**'s for the additional rocks: **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rocks, **VSNM**$_{3,t}$ for leaf, **VSNM**$_{4,t}$ for rocks, **VSNM**$_{5,t}$ for rocks~~)
`Object_Seg_Mod.visualsegment`(**LNM**$_{(1,\Upsilon,t)}$, **CONTEXT**$_t$, **VNM''**$_t$) → **VSNM**$_{t,i=1..\Theta\_i}$	-pseudocode for an algorithm that takes the best-matching visual input sensory local navigation map **LNM**$_{(1,\Upsilon,t)}$ and segments it into whatever objects it can find in its local memory stores or that meet certain algorithmic criteria (e.g., pixels separate from other pixels, and so on) -currently only segments via visual features (although one of the arguments **VNM''**$_t$ can contain information about auditory motion) -arguments were described above: **LNM**$_{(1,\Upsilon,t)}$: the best-matching visual input sensory local navigation map **CONTEXT**$_t$: to hold contextual information which help influence which objects are detected **VNM''**$_t$: auditory and visual motion of the overall scene or major object -produces **VSNM**$_{t,i=1..\Theta\_i}$ which are all the Visual Segmentation Navigation Maps **VSNM**'s created for a sensory scene
LNM'$_{(1,\Upsilon,t)}$	**LNM'**$_{(1,\Upsilon,t)}$ is the updated best-matching visual sensory Local Navigation Map **LNM**$_{(1,\Upsilon,t)}$ further updated with any visual or auditory motion prediction vectors from **VNM''**$_t$

Table A10. Explanation of Symbols and Pseudocode in Equations (56) – (62)

Initial Input:	-**AVNM**$_t$: a navigation map binding advanced auditory patterns; used in the next section -**LNM**$_{(1,\ \Upsilon,t)}$: the best-matching visual input sensory local navigation map; from the Input Sensory Vectors Association Module (Figures 1,9); will be segmented for objects in the sensory scene -other sensory system **LNM**s (represented by ***lnm***$_t$); used in the next section -**WNM'**$_t$: Working Navigation Map from the Navigation Module (Figure 1); stored and used as **WNM'**$_{t-1}$ to produce a value for **CONTEXT**$_t$ which holds contextual information which helps influence which objects are detected -**VNM''**$_t$: auditory and visual motion of the overall scene or major object; from the Sequential/Error Correcting Module (Figure 1)
Initial Output:	-produces **VSNM**$_{t,i=1..\Theta\_i}$ which are all the Visual Segmentation Navigation Maps **VSNM**'s created for a sensory scene; sent to the Sequential/Error Correcting Module (Figure 1)
Final Input:	-**VSNM'**$_{t,i=1..\Theta\_i}$: the Sequential/Error Correcting Module (Figure 1) will add where indicated a motion prediction vector to the **VSNM**'s it receives if there is motion of the object, and then send the **VSNM'**$_{t,i\ i=1..\Theta\_i}$ (i.e., all the **VSNM**'s) back to the Object Segmentation Gateway Module -e.g., in the example above: : **VSNM'**$_{1,t}$ for river (unchanged), **VSNM'**$_{2,t}$ for rock (unchanged) and **VSNM'**$_{3,t}$ for leaf (motion prediction vector added since apparent motion of leaf detected)
Final Output:	- **VSNM'**$_{t,i=1..\Theta\_i}$: visual segmented (i.e., objects separated) navigation maps with motion prediction vectors if motion detected - **AVNM**$_t$: a navigation map binding advanced auditory patterns; used in the next section - **LNM'**$_{(1,\ \Upsilon,t)}$: updated best-matching visual sensory Local Navigation Map **LNM**$_{(1,\ \Upsilon,t)}$ including any motion prediction vectors from **VNM''**$_t$ - other sensory system **LNM**s (represented by ***lnm***$_t$); passed through the module; used in the next section
Description:	-This module attempts to segment the input sensory by objects in it. -Segmentation is largely visual at present (although some limited auditory motion at present). -After segmenting a sensory scene into objects, the individual objects are sent to the Sequential/Error Correcting Module to detect motion of the objects, and if motion exists then a motion prediction vector is added to the **VSNM** navigation maps representing each object producing **VSNM'**. -Figure 10 illustrates the segmentation and additional of motion prediction vectors

Table A11. Summary of the Operation of the Object Segmentation Gateway Module per Equations (56) – (62)

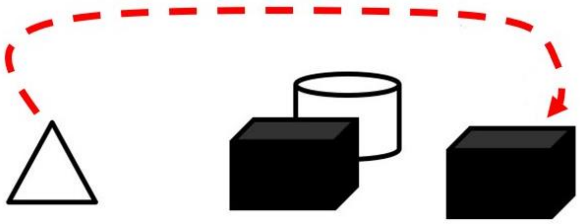

Figure 1 REPRODUCED FOR APPENDIX. The instruction to "place the white triangle on top of the black block which is not near a white cylinder" may be difficult to successfully follow by connectionist systems. The dashed arrow represents the correct solution to this instruction. (Note: The dashed arrow is *not* shown to the computer system being asked to follow this instruction.)

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	cylinder,white, link{0023,0,0,0}	air	air	air
triangle, white, link{0024,0,0,0}	air	block, black, link {0022,3,3,0}	air	black, block, link{0021,0,0,0}	air

Navigation Module A

"not", link{+}	"near", link{+}	"a", link{+}	"whi", link{+}	"cylinder", link{+}	
"of", link{+}	"the", link{+}	"black", link{+}	"block", link {+}	"which", link {+}	"is", link {+}
"place", link{+}	"the", link{+}	"white", link{+}	"triangl", link{+}	"on", link{+}	"top", link{+}

Navigation Module B

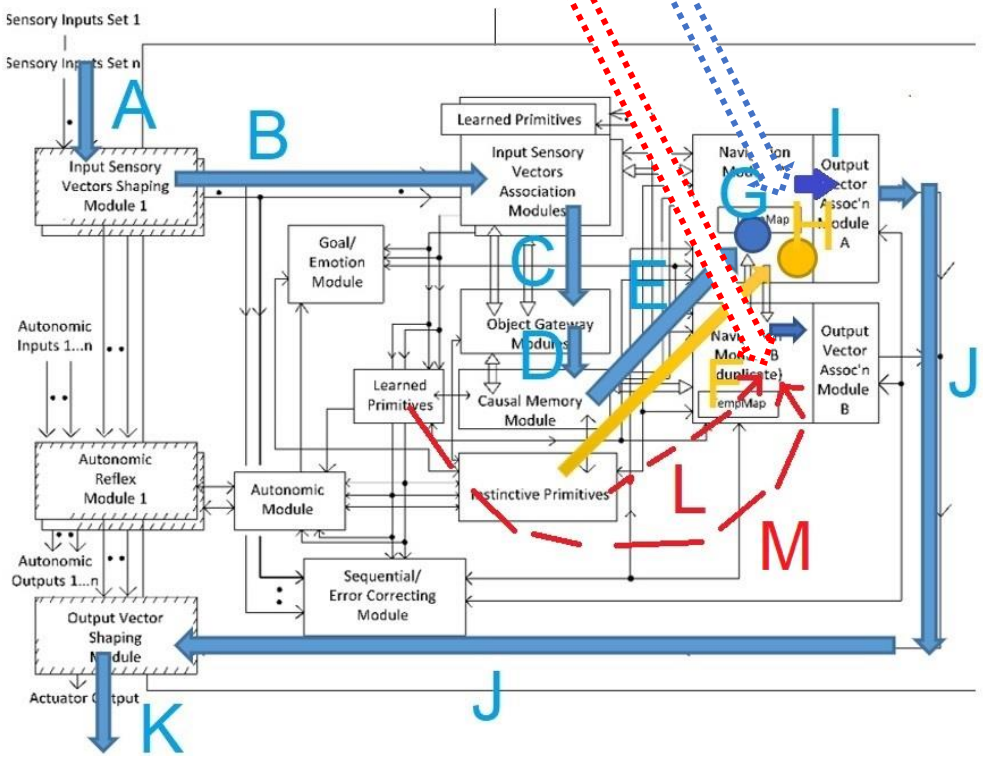

Figure 7 REPRODUCED FOR APPENDIX. A navigation map representing a far view of Figure 1 is in Navigation Module A. Another navigation map representing the instruction associated with Figure 1 ("place the white triangle on top of the black block which is not near a white cylinder") is in Navigation Module B. Cognitive cycles occur and 'mechanically' process the contents of the Navigation Modules via feedback operations which ignore new real-world sensory inputs for the moment (i.e., arrow B has no/ignored activity during these internal operations). (`link{+}`– link to the next cell. Arrow/circle letters from Figure 5.) See text for full description.

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	cylinder,white, link {0023,0,0,0}	air	air	air
<”place”> triangle, white, link {0024,0,0,0}	air	block, black, link {0022,3,3,0}	air	black, block, link {0021,0,0,0}	air

Figure 8 REPRODUCED FOR APPENDIX. First word of the sentence of Figure 1 being parsed by the instinctive primitive `parse_sentence()` with application to the navigation map of Navigation Module A (Figure 7) representing the sensory scene of Figure 1. The primitive has written the action word “place” in the cell containing the features “white, triangle.” (Navigation Module A)

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	cylinder,white, link {0023,0,0,0}	air	air	air
<”place”>, triangle, white, link {0024,0,0,0}	air	<”top”>,block, black, link {0022,3,3,0}	air	<”top”>,black , block, link {0021,0,0,0}	air

Figure 9 REPRODUCED FOR APPENDIX. The instinctive primitive `parse_sentence()` has now written the word “top” in the cells containing the features “black, block.” (Navigation Module A)

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	<”not”>, cylinder,white, link {0023,0,0,0}	air	air	air
<”place”>, triangle, white, link {0024,0,0,0}	air	<”not”>, <”top”>,block, black, link {0022,3,3,0}	air	<”top”>, black,block, link {0021,0,0,0}	air

Figure. 10 REPRODUCED FOR APPENDIX. The instinctive primitive `physics_near_object()` has now written “not” in the cells containing or adjoining cells containing the features “cylinder, white.” (Navigation Module A)

air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	air	air	air	air
air	air	<"not"> cylinder,white, link{0023,0,0,0}	air	air	air
<"place"> triangle, white, link{0024,0,0,0}	air	<"not"><"top"> block, black, link {0022,0,3,0}	air	<"top"> black, block, link{0021,0,0,0}	air

Navigation Module A

"not", link{+}	"near", link{+}	"a", link{+}	"white", link {+}	"cylinder", link{+}	
"of", link{+}	"the", link{+}	"black", link{+}	"block", link {+}	"which", link {+}	"is", link {+}
"place", link{+}	"the", link{+}	"white", link{+}	"triangle", link {+}	"on", link{+}	"top", link{+}

Navigation Module B

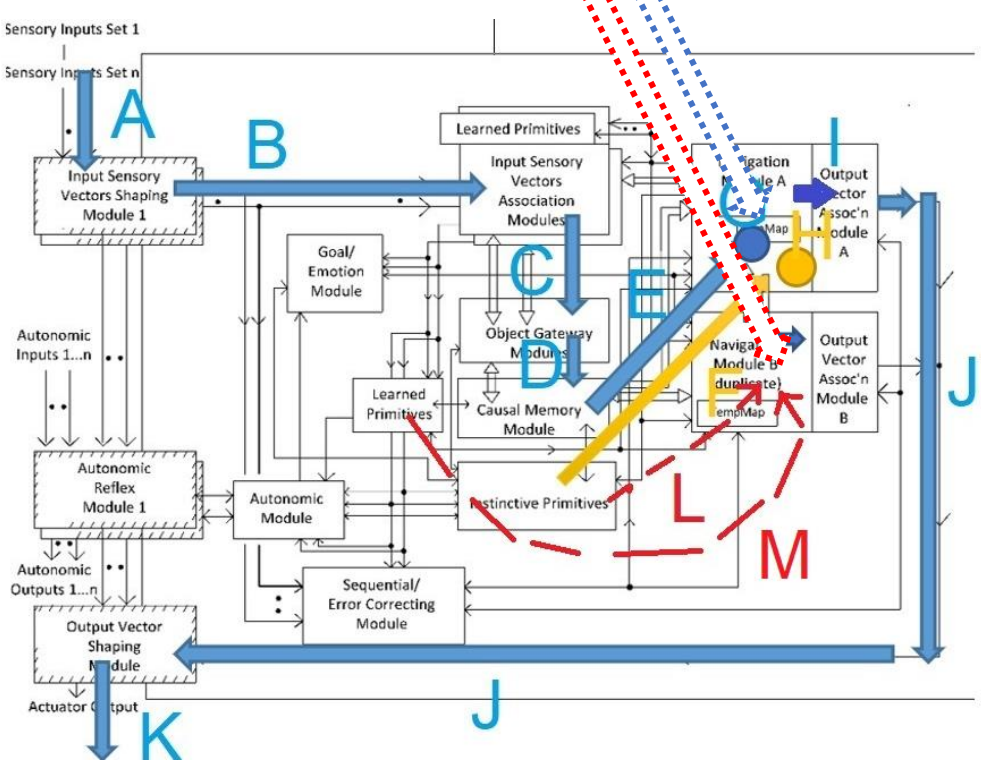

Figure 11 REPRODUCED FOR APPENDIX. A navigation map representing a far view of Figure 1 is in Navigation Module A. Another navigation map representing the instruction associated with Figure 1 ("place the white triangle on top of the black block which is not near a white cylinder") is in Navigation Module B. Navigation Module A cells have been modified by the instinctive primitives `parse_sentence()` and `physics_near_object()`. See text for full description.

A.6 The Causal Memory Module

The input sensory inputs at this point have been transformed as follows:

- visual sensory inputs → $\mathbf{VSNM'}_{(t,i=1...\Theta\_i)}$: visual segmented (i.e., objects separated) navigation maps with motion prediction vectors if motion detected
- visual sensory inputs → $\mathbf{LNM'}_{(1,\Upsilon,t)}$: updated best-matching visual sensory Local Navigation Map $\mathbf{LNM}_{(1,\Upsilon,t)}$ including any motion prediction vectors from $\mathbf{VNM''}_t$
- auditory sensory inputs → $\mathbf{AVNM}_t$: a navigation map binding auditory patterns
- olfactory sensory inputs → $\mathbf{LNM}_{(3,\Upsilon,t)}$: updated best-matching olfactory sensory Local Navigation Map $\mathbf{LNM}_{(3,\Upsilon,t)}$
- other sensory systems inputs → $\mathbf{LNM}_{(4...n\_\sigma,\Upsilon,t)}$ **:** updated best-matching other sensory (e.g., tactile, radar, etc.) Local Navigation Maps $\mathbf{LNM}_{(4...n\_\sigma,\Upsilon,t)}$

These single sensory system navigation maps will now be matched against the previously stored multi-sensory navigation maps stored in the Causal Memory Module. The matching algorithm `Object_Seg_Mod.match_best_map` (63) chooses previously stored Navigation Map A as the best match. Thus, Navigation Map A (i.e., χ = [[modcode=Causal_Mem_Mod, *mapno*=3579], x,y,z]) is set to become the Working Navigation Map **WNM** (63).

WNM must be updated with the actual sensory inputs, since after all it is at this point simply a retrieved best matching navigation map. If there are too many changes between the actual sensory inputs $\boldsymbol{actual}_t$ (64) and an arbitrary threshold h' (65), then rather than update **WNM** a new Working Navigation Map **WNM'** will be created from the actual sensory inputs $\boldsymbol{actual}_t$ via copying the actual sensory inputs $\boldsymbol{actual}_t$ to an empty navigation map **TempMap** (69). Note from (66) that **TempMap** is actually defined as the temporary memory area "TempMap" within the Navigation Module A(Figure 5). Often values from the Causal Memory Module (e.g., a linked navigation map) or Navigation Module may have automatically been sent (i.e., copied) to **TempMap**. Thus, the pseudocode `Nav_ModA.TempMap.erase()` (68) is called to ensure **TempMap** has no contents and is equivalent to an empty navigation map before using it in (69).

However, if **WNM** (i.e., which is a retrieved navigation map from the Causal Memory Module) (63) is close enough to the actual input sensory inputs $\boldsymbol{actual}_t$, then the information from $\boldsymbol{actual}_t$ is copied to **WNM**, and creates the updated Working Navigation Map $\mathbf{WNM'}_t$ (67).

In (70) **WNM'** is stored in the Causal Memory Module—in future operations of the architecture it will also be matched against various sensory inputs. This pseudocode also specifies that there is updating of the ***linkaddresses*** to and from other navigation maps in the Causal Memory Module and the updated Working Navigation Map $\mathbf{WNM'}_t$.

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture, other than distinguishing the two Navigation Modules.

$$\mathbf{WNM}_t = \texttt{Object\_Seg\_Mod.match\_best\_map}$$
$$(\mathbf{VSNM'}_{(t,i=1...\Theta\_i)}, \mathbf{LNM'}_{(1,\Upsilon,t)}, \mathbf{AVNM}_t, \mathbf{LNM}_{(3,\Upsilon,t)}, \mathbf{LNM}_{(4...n\_\sigma,\Upsilon,t)}) \quad (63)$$

$$\boldsymbol{actual}_t = [\mathbf{VSNM'}_{(t,i=1...\Theta\_i)}, \mathbf{LNM'}_{(1,\Upsilon,t)}, \mathbf{AVNM}_t, \mathbf{LNM}_{(3,\Upsilon,t)}, \mathbf{LNM}_{(4...n\_\sigma,\Upsilon,t)}] \quad (64)$$

$$h' = \text{number of differences allowed to be copied onto existing navigation map} \in R \quad (65)$$

$$\mathbf{TempMap} = \texttt{Nav\_ModA.TempMap} \in R^{mxnxoxp} \quad (66)$$

$$|\ \texttt{Object\_Seg\_Mod.differences}(\boldsymbol{actual}_t, \mathbf{WNM}_t)\ | \leq h',$$
$$\Rightarrow \mathbf{WNM'}_t = \mathbf{WNM}_t \cup \boldsymbol{actual}_t \quad (67)$$

`Nav_ModA.TempMap.erase()` (68)

`|Object_Seg_Mod.differences(`$\boldsymbol{actual}_t$, $\mathbf{WNM}_t$`)|` >h', $\Rightarrow \mathbf{WNM'}_t = \mathbf{TempMap} \cup \boldsymbol{actual}_t$ (69)

`Causal_Mem_Mod.store_WNM_update_links(`$\mathbf{WNM'}_t$`)` (70)

`Object_Seg_Mod.match_best_map(` **VSNM'**$_{(t,i=1...\Theta\_i)}$, **LNM'**$_{(1,\Upsilon,t)}$, **AVNM**$_t$, **LNM**$_{(3,\Upsilon,t)}$, **LNM**$_{(4...n\_\sigma,\Upsilon,t)}$ **)** → **WNM**	-pseudocode for an algorithm that takes processed single sensory navigation maps and matches them to the best matching multisensory navigation map in the Causal Memory Module (Figure 1) -note that this pseudocode is run from the Object Segmentation Gateway Module since all of the arguments of this pseudocode are in the latter module -a small example of this algorithm is illustrated in Figure 11 where the best-matching visual local navigation map **LNM'**$_{(1,\Upsilon,t)}$, the visual segmentation navigation maps **VSNM'**$_{(t,i=1...\Theta\_i)}$, the navigation map binding auditory patterns **AVNM**$_t$, and the olfactory local navigation map **LNM**$_{(3,\Upsilon,t)}$, are matched against the multi-sensory navigation maps of the Causal Memory Module -the arguments to the algorithm (**VSNM'**$_{(t,i=1...\Theta\_i)}$, **LNM'**$_{(1,\Upsilon,t)}$, **AVNM**$_t$, **LNM**$_{(3,\Upsilon,t)}$, **LNM**$_{(4...n\_\sigma,\Upsilon,t)}$) are described individually below -the best matching multisensory map is designated as the Working Navigation Map **WNM**
argument used in `match_best_map`**:** **VSNM'**$_{t,i=1...\Theta\_i}$ **VSNM**$_{t,i=1...\Theta\_i}$ → **VSNM'**$_{t,i=1...\Theta\_i}$ (Note: **VSNM** is first created in the Object Segmentation Gateway Module, and then updated to **VSNM'** in the Sequential/Error Correcting Module)	-a "Visual Segmentation Navigation Map" **VSNM** is an ordinary navigation map with visually segmented information stored on it -the Object Segmentation Gateway Module (xxx~~Figures 9, 10~~) will "segment" a sensory scene into objects of interest -each segmented object is treated as a separate navigation map (there is also a navigation map of all the objects together on the navigation map) -the subscript *i* refers to multiple **VSNM**'s that can be created for a given sensory scene -i=1…Θ_i refers to the full set for **VSNM**'s created for a given sensory scene, i.e., **VSNM**$_1$ to the last **VSNM**$_{\Theta\_i}$ created -the subscript *t* refers to time since the values change each cognitive cycle -it is useful to calculate motion prediction vectors, if they exist (often they will not) for each segmented object (e.g., if there is a river object, a rock object and leaf floating in the river object, then it is very useful to segment the sensory scene into these objects (i.e., river, rock and leaf) and a motion prediction vector to show the motion of the leaf is very useful – thus there will be **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock and **VSNM**$_{3,t}$ for leaf, in this example) -continuing this example: **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock and **VSNM**$_{3,t}$ for leaf will be propagated to the Sequential/Error Correcting Module for computation of motion prediction vectors for each of these **VSNM**'s -continuing this example: a motion prediction vector is computed and stored on **VSNM**$_{3,t}$ for leaf, but **VSNM**$_{1,t}$ for river, **VSNM**$_{2,t}$ for rock did not have enough motion for such vectors and are unchanged -continuing this example: **VSNM'**$_{1,t}$ for river (unchanged), **VSNM'**$_{2,t}$ for rock (unchanged) and **VSNM'**$_{3,t}$ for leaf (motion prediction vector added) are then returned back to the Object Segmentation Gateway Module
argument used in `match_best_map`**:** **LNM'**$_{(1,\Upsilon,t)}$	**-LNM'**$_{(1,\Upsilon,t)}$ is the updated best-matching visual sensory Local Navigation Map -- **LNM**$_{(1,\Upsilon,t)}$ is further updated with any visual or auditory motion prediction vectors from **VNM''**$_t$ to create **LNM'**$_{(1,\Upsilon,t)}$

argument used in `match_best_map`: **AVNM**$_t$ `Sequential_Mod.auditory_match_process` (***auditory_series***$_t$) → → **AVNM**$_t$	-pseudocode for an algorithm that matches ***auditory_series***$_t$ with auditory time series stored in the Sequential/Error Correcting Module (Figure 1) -a best match is chosen (or if no matches close enough then ***auditory_series***$_t$ used itself as the best match) -the best match is then updated from ***auditory_series***$_t$ (much in the same manner as for example the matching and updating of navigation maps illustrated above in Figures 4 and 5) -the updated best match is stored in the Sequential/Error Correcting Module for future matching -the updated best match is then transformed into a complex advanced motion prediction vector (i.e., showing in more detail the pattern of the auditory sensations) and is output as **AVNM**$_t$ (which as described above is an "Audio Vector Navigation Map")
argument used in `match_best_map`: **LNM**$_{(3, \Upsilon, t)}$	the local navigation map **LNM** stored in the Input Sensory Vectors Association Module σ with map number Υ which best matches in the incoming σ sensory inputs (e.g., if $\sigma = 3$ then it would be olfactory sensory inputs)
argument used in `match_best_map`: **LNM**$_{(4 \ldots n\_\sigma, \Upsilon, t)}$	-for illustrative purposes only three sensory systems: visual, auditory and olfactory are often used here, but there can be additional ones such as tactile, or synthetic ones such as radar, etc. -4…n_σ means from **LNM**$_4$ to the last sensory system being used and its local navigation map **LNM**$_{n\_\sigma}$
actual$_t$	-this is a vector that holds all the arguments for the pseudocode `match_best_map` = [**VSNM'**$_{(t,i=1...\Theta\_i)}$, **LNM'**$_{(1, \Upsilon, t)}$, **AVNM**$_t$, **LNM**$_{(3, \Upsilon, t)}$, **LNM**$_{(4 \ldots n\_\sigma, \Upsilon, t)}$]
TempMap (NavMod A module)	-a navigation map of the same dimensions as the other navigation maps in the architecture but it is a short-term memory region in the Navigation Module A `Nav_ModA.TempMap` rather than being an ordinary navigation map
`Object_Seg_Mod.differences()`	-pseudocode for an algorithm that calculates the differences between two navigation maps -similar to `Input_Assocn_Mod`$_\sigma$`.differences` described above (21, 22) -used in (67) and (68) to calculate the differences between the processed input sensory signal ***actual***$_t$ and the retrieved best matching multi-sensory navigation map **WNM**$_t$
WNM$_t$ ∪ ***actual***$_t$ → **WNM'**$_t$	-**WNM** must be updated with the actual sensory inputs -the processed sensory inputs ***actual***$_t$ are copied to **WNM**, thereby forming **WNM'** the updated Working Navigation Map which will be used by the Navigation Module in the section below
h' **TempMap**$_t$ ∪ ***actual***$_t$ → **WNM'**$_t$	-**WNM** must be updated with the actual sensory inputs -if there are too many changes between the actual sensory inputs ***actual***$_t$ (64) and this arbitrary threshold h', then rather than update **WNM** , the actual sensory inputs ***actual***$_t$ will be used to create a new navigation map which is considered the Working Navigation Map **WNM'** which will be used by the Navigation Module in the section below -as noted above **TempMap** is a temporary memory region in the Navigation Module A that simulates a navigation map; ***actual***$_t$ is being copied to an empty navigation map and results in **WNM'**$_t$
WNM'$_t$	-updated Working Navigation Map -the current Working Navigation Map **WNM'**$_t$ is the navigation map which the Navigation Module A focuses its attention on, i.e., applies operations on to make a decision to take some sort or no action -as will be seen in the following sections, a navigation map can be assigned as being the **WNM'**$_t$ for that cognitive cycle, and then a different one in the next cognitive cycle, and so on, depending on the sensory information being processed and the results obtained

`Causal_Mem_Mod.store_WNM_update_links` (**WNM'**$_t$)	-pseudocode specifying that the updated Working Navigation Map **WNM'**$_t$ is stored in the Causal Memory Module - pseudocode specifying updating of the ***linkaddresses*** to and from the updated Working Navigation Map **WNM'**$_t$ is stored in the Causal Memory Module -in future cognitive cycles the processed sensory inputs ***actual***$_t$ will be matched against the navigation maps in the Causal Memory Module which includes this newly stored navigation map
`Nav_ModA.TempMap.erase()`	-pseudocode specifying that **TempMap** is erased -thus, it will act as an empty navigation map the next time it is required

Table A12. Explanation of Symbols and Pseudocode in Equations (63) – (70)

Input:	**-VSNM'**$_{(t,i=1...\Theta\_i)}$: visual segmented (i.e., objects separated) navigation maps with motion prediction vectors if motion detected **-LNM'**$_{(1,\Upsilon,t)}$: best-matching visual sensory Local Navigation Map **LNM**$_{(1,\Upsilon,t)}$ including any motion prediction vectors from **VNM''**$_t$ **-AVNM**$_t$: a navigation map binding auditory patterns **-LNM**$_{(3,\Upsilon,t)}$: the olfactory local navigation map **-LNM**$_{(4...n\_\sigma,\Upsilon,t)}$: other sensory local navigation maps (these processed sensory inputs are represented by the vector ***actual***$_t$)
Output:	**WNM'**$_t$: updated Working Navigation Map which will be used in the Navigation Module A (as well as other modules of the architecture)
Description:	-In this module the processed single sensory system navigation maps (***actual***$_t$) will be matched against the previously stored multi-sensory navigation maps stored in the Causal Memory Module. -The best matching navigation map is then updated with the actual sensory inputs, i.e., ***actual***$_t$, and acts as the Working Navigation Map **WNM'** for this cognitive cycle. -The current Working Navigation Map **WNM'**$_t$ is the navigation map which the Navigation Module A focuses its attention on, i.e., applies operations on to make a decision to take some sort or no action.

Table A13. Summary of the Operation of the Causal Memory Module per Equations (63) – (70)

A.7 Navigation Module A

As seen in Figure 3 there are now in the CCA6 two Navigation Modules—Navigation Module A and Navigation Module B (as opposed to the single Navigation Module in the prior CCA5 architecture, as seen in Figure 2). In the Navigation Module A an instinctive primitive or a learned primitive (which themselves are navigation maps) in conjunction with the Working Navigation Map **WNM'** may result in an action signal. This signal goes to the Output Vector Association Module A and to the external embodiment, completing the cognitive cycle.

In some cognitive cycles there will be no output but intermediate results from the Navigation Module A are fed back and re-operated on in the next cognitive cycle. This will be discussed in the following section. In this section, the more straightforward case is considered where a learned or instinctive primitive is applied against a Working Navigation Map **WNM'** and an action is produced.

From Figure 5 the Autonomic Module and the Goal/Emotion Module can be seen. These modules, which operate in the areas much as their names indicate, are essential for an agent such as the Causal Cognitive Architecture 6. However, these modules will not be formalized so as to focus on the Navigation Module A in this section. The *autonomic* variable (71) which reflects the energy levels and maintenance issues with embodiments of the architecture, will affect the value of the *emotion* variable (72) and the **GOAL** navigation map (73) as shown in equation (74). The **GOAL** variable reflects both intuitive and learned goals of the architecture acting as an agent and is represented as a full navigation map. The *emotion* variable will affect which features the architecture pays more attention to in their processing as well as in selecting the appropriate instinctive or learned primitive to act on the current Working Navigation Map **WNM'**.

As mentioned earlier, instinctive and learned primitives, which act as small rules or productions, are stored in modified navigation maps, called respectively instinctive primitive navigation maps **IPM**, and learned primitive navigation maps **LPM**. These primitive navigation maps have cells that contain procedures. However, the cells can also contain features and linkaddresses.

In the Navigation Module A, a learned or instinctive primitive navigation map is compared against the Working Navigation Map **WNM'**. By simple array operations and logical operations, the Navigation Module A may produce an *action* signal, i.e., a signal to move some actuator or send an electronic signal. The *action* signal goes to the Output Vector Association Module A and then to the external embodiment.

Instinctive primitives **IPM'**s are stored in the Instinctive Primitives Module and come preprogrammed with the architecture. Learned primitives **LPM**'s in the CCA6 are now distributed among the many Input Sensory Vectors Association Modules and the Causal Memory Module (Figure 5). Both instinctive and learned primitive navigation maps have cells that contain procedures. However, the cells can also contain features and linkaddresses.

The pseudocode indicating the algorithm `Instinctive_Primitives_Mod.match_best_primitive` (***actual***$_t$, *emotion*$_t$, **GOAL**$_t$) (76) will choose the most appropriate instinctive primitive **WIP**$_t$ (75) to apply against the Working Navigation Map **WNM'** in the Navigation Module A. Similarly, the pseudocode indicating the algorithm `Learned_Primitives_Mod.match_best_primitive` (***actual***$_t$, *emotion*$_t$, **GOAL**$_t$) (78) will choose the most appropriate learned primitive **WLP**$_t$ (77) to apply against the Working Navigation Map **WNM'** in the Navigation Module A.

Then a decision is made to use either the best instinctive primitive **WIP**$_t$ or the best learned primitive **WLP**$_t$ as the Working Primitive **WPR**$_t$ (79) which will actually be the primitive applied against the Working Navigation Map **WNM'**. At present a simple scheme is used to make this decision. If there is no learned primitive **WLP**$_t$ then the best instinctive primitive **WIP**$_t$ is assigned to be the Working Primitive **WPR**$_t$ (80). Otherwise, if a best learned primitive exists, then it is assigned to be the Working Primitive **WPR**$_t$ (81).

At this point in the Navigation Module A there is a Working Primitive **WPR**$_t$ and a Working Navigation Map **WNM'**. By simple array operations and logical operations, the Navigation Module A compares and operates on the Working Primitive **WPR**$_t$ and the Working Navigation Map **WNM'**. Equation (82) shows this is indicated by pseudocode `Nav_ModA.apply_primitive`(**WPR**$_t$, **WNM'**$_t$). As a result, an *action*$_t$ value may be produced, which is a signal to move some actuator or send an electronic signal. The *action* signal goes to the Output Vector Association Module A and then to the external embodiment (Figure 5).

The *action* signal from the Navigation Module A may be, for example, **<move right>**. This signal is processed by the Output Vector Association Module A (Figure 5). The pseudocode `Output_Vector_Mod.action_to_output`(*action*$_t$, **WNM'**$_t$) (84) results in ***output_vector***$_t$ which contains more detailed instructions for the actuators to cause the embodiment to move right. However, ***output_vector***$_t$ is first sent to the Sequential/Error Correcting Module where it will be corrected for motion and timing issues, resulting in a ***motion_correction***$_t$ vector (86). The pseudocode `Output_Vector_Mod.apply_motion_correction`(***output_vector***$_t$, ***motion_correction***$_t$) (87) produces the final signal ***output_vector'***$_t$ which is sent to the Output Vector Shaping Module (Figure 5). In the Output Vector Shaping Module the final signal shaping and low-level transformations of the signal are performed (not indicated in the equations) to directly signal movement of the embodiment's actuators or signal transmission of an electronic signal.

These equations in the CCA6 architecture are somewhat changed from the prior CCA5 architecture to reflect the duplication of the Navigation Modules.

$$autonomic \in R \quad (71)$$

$$emotion \in R \quad (72)$$

$$\mathbf{GOAL} \in R^{mxnxoxp} \quad (73)$$

$$[emotion_t, \mathbf{GOAL}_t] = \texttt{Goal/Emotion\_Mod.set\_emotion\_goal}(\ autonomic_t, \mathbf{WNM'}_{t-1}\ ) \quad (74)$$

$$\mathbf{WIP} \in R^{mxnxoxp} \quad (75)$$

$$\mathbf{WIP}_t = \texttt{Instinctive\_Primitives\_Mod.match\_best\_primitive}(\boldsymbol{actual}_t, \boldsymbol{emotion}_t, \mathbf{GOAL}_t) \quad (76)$$

$$\mathbf{WLP} \in R^{mxnxoxp} \quad (77)$$

$$\mathbf{WLP}_t = \texttt{Learned\_Primitives\_Mod.match\_best\_primitive}(\boldsymbol{actual}_t, \boldsymbol{emotion}_t, \mathbf{GOAL}_t) \quad (78)$$

$$\mathbf{WPR} \in R^{mxnxoxp} \quad (79)$$

$$\mathbf{WLP}_t = [\ ], \Rightarrow \mathbf{WPR}_t = \mathbf{WIP}_t \quad (80)$$

$$\mathbf{WLP}_t \neq [\ ], \Rightarrow \mathbf{WPR}_t = \mathbf{WLP}_t \quad (81)$$

$$\boldsymbol{action}_t = \texttt{Nav\_ModA.apply\_primitive}(\mathbf{WPR}_t, \mathbf{WNM'}_t) \quad (82)$$

$$\boldsymbol{output\_vector} \in R^{n'} \quad (83)$$

$$\boldsymbol{action}_t = [\text{“move*”}], \Rightarrow \boldsymbol{output\_vector}_t = \texttt{Output\_Vector\_Mod.action\_to\_output}(\boldsymbol{action}_t, \mathbf{WNM'}_t) \quad (84)$$

$$\boldsymbol{motion\_correction} \in R^2 \quad (85)$$

$$\boldsymbol{action}_t = [\text{“move*”}], \Rightarrow \boldsymbol{motion\_correction}_t = \texttt{Sequential\_Mod.motion\_correction}(\boldsymbol{action}_t, \mathbf{WNM'}_t, \boldsymbol{visual\_series}_t) \quad (86)$$

$$\boldsymbol{output\_vector'}_t = \texttt{Output\_Vector\_Mod.apply\_motion\_correction}(\boldsymbol{output\_vector}_t, \boldsymbol{motion\_correction}_t) \quad (87)$$

autonomic	variable which reflects the energy levels and maintenance issues with embodiments of the architecture
emotion	variable which affects which features the architecture pays more attention to
GOAL	reflects both intuitive and learned goals of the architecture acting as an agent, and is represented as a full navigation map
`Goal/Emotion_Mod.set_emotion_goal`(*autonomic*$_t$, **WNM'**$_{t-1}$) → [*emotion*$_t$, **GOAL**$_t$]	-pseudocode that calculates the values of the *emotion* and **GOAL** variables -`Goal/Emotion_Mod` refers to the Goal/Emotion Module (Figure 1) -the argument **WNM'**$_{t-1}$ is the Working Navigation Map from the previous (i.e., t-1) cognitive cycle
`Instinctive_Primitives_Mod.match_best_primitive` (***actual***$_t$, *emotion*$_t$, **GOAL**$_t$) → **WIP**$_t$	-pseudocode that returns the most appropriate instinctive primitive navigation map **IPM** from the Instinctive Primitives Module (~~xxxxFigure 14~~) given the arguments of the processed sensory inputs ***actual***$_t$, the current *emotion*$_t$ value, and the current **GOAL**$_t$ navigation map -`Instinctive_Primitives_Mod` refers to the Instinctive Primitives Module (xxx ~~Figure14)~~ -the most appropriate instinctive primitive **IPM** returned is considered to be the Working Instinctive Primitive **WIP**$_t$
`Learned_Primitives_Mod.match_best_primitive` (***actual***$_t$, *emotion*$_t$, **GOAL**$_t$) → **WLP**$_t$	-pseudocode that returns the most appropriate learned primitive navigation map **LPM** from the Instinctive Primitives Module (xxxx ~~Figure 14~~) given the arguments of the processed sensory inputs ***actual***$_t$, the current *emotion*$_t$ value, and the current **GOAL**$_t$ navigation map -`Learned_Primitives_Mod` refers to the Learned Primitives Module (xxxx ~~Figure 14)~~ -the most appropriate instinctive primitive **LPM** returned is considered to be the Working Learned Primitive **WLP**$_t$

$\mathbf{WLP}_t$ = [ ], ⇒ $\mathbf{WPR}_t$ = $\mathbf{WIP}_t$ $\mathbf{WLP}_t \neq$ [ ], ⇒ $\mathbf{WPR}_t$ = $\mathbf{WLP}_t$ → **WPR**	-the Working Primitive **WPR** is the primitive that will be applied against the Working Navigation Map **WNM'** in the Navigation Module A -it is composed of inputs from the Working Instinctive Primitive **WIP** and the Working Learned Primitive **WLP** -currently a simple scheme to calculate it is used since there is only a very small number of learned primitives **LPM**'s—if a Working Learned Primitive **WLP** exists then $\mathbf{WLP}_t$ will be used as $\mathbf{WPR}_t$, otherwise the Working Instinctive Primitive $\mathbf{WIP}_t$ will be used as $\mathbf{WPR}_t$
procedure	-a cell in a navigation map can hold values which are *procedure* variables (35), *feature* variables (34), and linkaddress χ' variables (36) -a *procedure* value could be, for example, **<move right>** which would indicate that the embodiment of the architecture should move towards the right
`Nav_ModA.apply_primitive`(**WPR**$_t$, **WNM'**$_t$) → *action*$_t$	-`Nav_ModA` refers to the Navigation Module A (Figure 5) -`apply_primitive`(**WPR**$_t$, **WNM'**$_t$) is pseudocode for an algorithm which applies the Working Primitive $\mathbf{WPR}_t$ to the Working Navigation Map **WNM'**, often producing an *action*$_t$ value -note that in some cognitive cycles no *action*$_t$ value may result from the current $\mathbf{WPR}_t$ and $\mathbf{WNM'}_t$ -in other cognitive cycles instead of an actionable *action*$_t$ value there is the feeding back of intermediate results for re-processing in the next cognitive cycle (discussed in the next section) -the mechanism of producing an *action*$_t$ value is via simple array operations (both $\mathbf{WPR}_t$ and $\mathbf{WNM'}_t$ are arrays) and very simple logical operations that could be biologically feasible as possible
`Output_Vector_Mod.` `action_to_output`(*action*$_t$, **WNM'**$_t$) → ***output_vector***$_t$	-`Output_Vector_Mod` refers to the Output Vector Association Module A (xxxx ~~Figure 14~~) -pseudocode describing taking the *action*$_t$ value and the current Working Navigation Map $\mathbf{WNM'}_t$ and creating ***output_vector***$_t$ which would provide more detailed instructions for the actuators of the embodiment in order to carry out the *action*$_t$ value -for example, the ***action*** value could be, for example, **<move right>** which this pseudocode would transform into ***output_vector***$_t$ which provides more detailed instructions for the embodiment's actuators with regard to moving to the right
`Sequential_Mod.` `motion_correction`(*action*$_t$, **WNM'**$_t$, ***visual_series***$_t$) → ***motion_correction***$_t$	-`Sequential_Mod` refers to the Sequential/Error Correcting Module (Figure 1) -pseudocode which produces a motion correction value based on the relative movement of the embodiment itself and the motion which is desired to achieve -the pseudocode considers what *action*$_t$ is desired, the current Working Navigation Map $\mathbf{WNM'}_t$ and the current movement of the embodiment via the visual movement detected via ***visual_series***$_t$ (44, 45) -in the future additional sensory systems (e.g., a positioning system, an inertial system, and so on) could be used in addition to the information in the ***visual_series***$_t$ -the vector ***motion_correction***$_t$ is computed and returned to the Output Vectors Association Module -for example, the current ***action*** value is **<move right>** so the pseudocode would compute if any motion correction was required based on the current actual movement of the embodiment of the architecture
`Output_Vector_Mod.` `apply_motion_correction`(***output_vector***$_t$, ***motion_correction***$_t$) → ***output_vector'***$_t$	-`Output_Vector_Mod` refers to the Output Vector Association Module A (xxx ~~Figure 14)~~ -pseudocode that applies the computed vector ***motion_correction***$_t$ against the previously computed ***output_vector***$_t$ -an updated ***output_vector'***$_t$ is produced

	→ ***output_vector'***$_t$ provides more detailed and motion corrected instructions for the actuators in this case, e.g, **<move right>**, to move the embodiment to the right

Table A14. Explanation of Symbols and Pseudocode in Equations (71) – (87)

Input:	**WPR**$_t$: the Working Primitive **WPR** is the primitive that will be applied against the Working Navigation Map **WNM'** in the Navigation Module A **WNM'**$_t$: the current Working Navigation Map which is derived from the processed sensory inputs
Output:	*action*$_t$: a signal to move some actuator or send an electronic signal e.g., **<move right>** it is sent to the Output Vector Association Module A (xxx ~~Figure 14~~) where more detailed instructions are created to effect the required actuator outputs
Description:	-The Navigation Module A applies the Working Primitive **WPR**$_t$ against the current Working Navigation Map **WNM'** and may produce an *action*$_t$ signal (sometimes no signal is produced or sometimes there is a signal to feedback intermediate results, discussed in the next section).

Table A15. Summary of the Operation of the Navigation Module A per Equations (71) – (87)

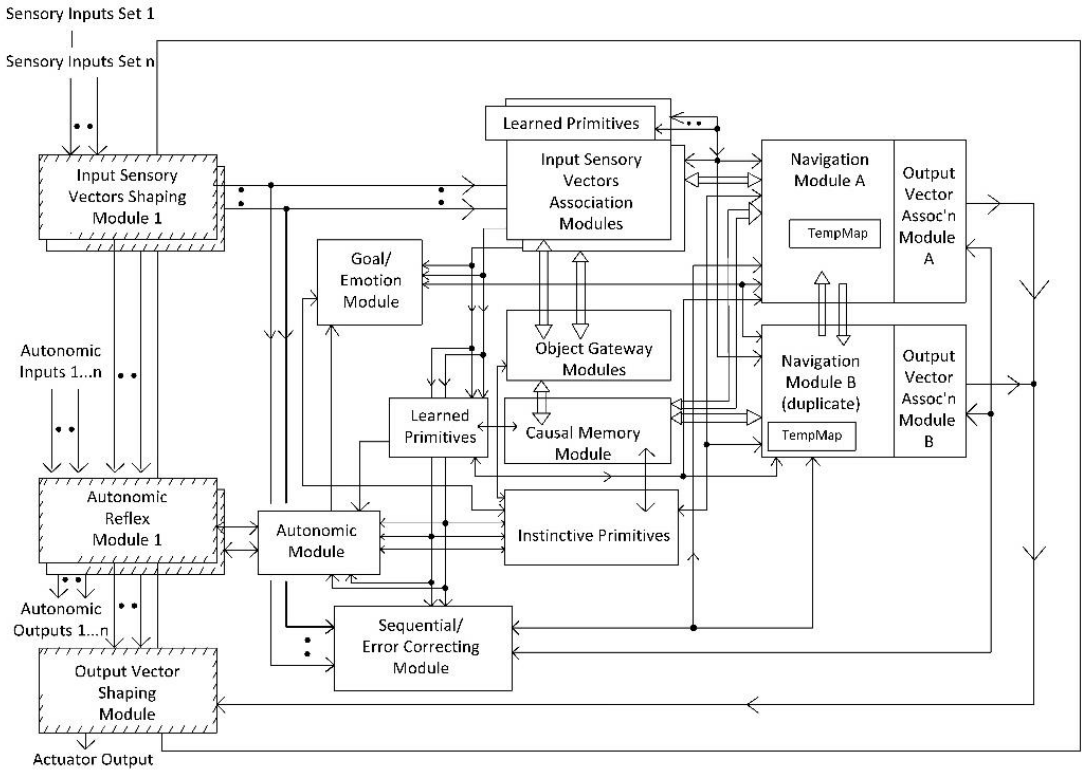

Figure 3 REPRODUCED FOR APPENDIX. Overview of the Causal Cognitive Architecture 6 (CCA6)

A.8 Feedback Signals and Intermediate Results

The Causal Cognitive Architecture, including the CCA6, makes use of feedback pathways—states of a downstream module can influence the recognition and processing of more upstream sensory inputs. This is advantageous in order to better recognize noisy or incomplete input sensory information. In previous versions of the architecture and again in the CCA6, the feedback pathways between the Input Sensory Vectors Association Modules and the Navigation Module A remain enhanced so that they can allow not just a feedback signal, but the full intermediate results from the Navigation Module A to be stored in the Input Sensory Vectors Association Modules (Figure 3).

In some cognitive cycles in the CCA6 there is no output signal. In certain states (for example, no actionable output from the Navigation Module A) the information in the Navigation Module A can be fed back and stored in the Input Sensory Vectors Association Modules (Figure 3). When this happens then in the next cognitive cycle these intermediate results will automatically be considered as the input sensory information and propagated to the Navigation Module A (Figure 3) and operated on again. As noted above in the body of the paper, by feeding back and re-operating on the intermediate results, the Causal Cognitive Architecture can formulate and explore possible cause and effect of actions, i.e., generate causal behavior.

The Working Primitive $\mathbf{WPR}_t$ has been applied against the Working Navigation Map $\mathbf{WNM'}_t$) producing an $action_t$ signal: $action_t$ = `Nav_ModA.apply_primitive`($\mathbf{WPR}_t$, $\mathbf{WNM'}_t$) (82). In the previous section the $action_t$ signal was actionable, i.e., it contained the string "move*" and so via equations (84–87) it was propagated to the Output Vector Association Module A and to the Output Vector Shaping Module A where it then moved the actuators of the embodiment or sent an electronic signal in accordance with the *action* specified. However, what if the $action_t$ signal produced is not actionable, i.e., it does not contain the string "move*"? There is no indication that any actuator should be moved, or any electronic signal should be sent. In this case $action_t$ would be discarded. Another $action_t$ signal would be produced in the next cognitive cycle, and perhaps it would be actionable.

However, given the enhancement in feedback pathways between the Navigation Module A and the Input Sensory Vectors Association Modules, the Causal Cognitive Architecture can in these cases feed back the intermediate results of the Navigation Module A and store them in the Input Sensory Vectors Association Modules (88). Note that equation (88) will feedback the intermediate results (i.e., the Working Navigation Map $\mathbf{WNM'}_t$ if $action_t \neq$ "move*", i.e., the $action_t$ signal is not actionable. However, if the Working Primitive $\mathbf{WPR}_t$ explicitly specifies to discard the intermediate results they will not be fed back. Similarly, if the Working Primitive $\mathbf{WPR}_t$ explicitly specifies to feed back the intermediate results they will be fed back even if they are actionable. In equation (88) the pseudocode `Nav_ModA.feedback_to_assocn_mod`($\mathbf{WNM'}_t$) specifies to feed back and store $\mathbf{WNM'}_t$ in the Input Sensory Vectors Association Modules.

In the next cognitive cycle, in equation (89) the pseudocode `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`$\mathbf{WNM'}_{t-1}$`)` specifies that the Working Navigation Map just fed back (i.e., $\mathbf{WNM'}_{t-1}$ which is $\mathbf{WNM'}_t$ of the previous cognitive cycle) is to have its components extracted (i.e., effectively stored) into the best matching local navigation maps of the various Input Sensory Vectors Associations Modules. Thus, as the cognitive cycle progresses, the actual sensory inputs will be ignored, and instead these components of the previous Working Navigation Map $\mathbf{WNM'}_{t-1}$ (i.e., the intermediate results from the Navigation Module A in the last cognitive cycle) are propagated towards the Navigation Module A and operated on again, perhaps by new instinctive or learned primitives in this new cognitive cycle. Feeding back the Navigation Module A's intermediate results back to the sensory stages and then processing the intermediate results in the next cycle by the Navigation Module A, over and over again as needed, will allow a robust application of rules to be applied onto the modeled world, and allow exploration and examination of what the effect of a cause will be. As described in the text of the paper above, previous Causal Cognitive Architectures have shown that by feeding back and re-operating on the intermediate results, the architecture can generate causal behavior.

These equations in the CCA6 architecture are somewhat changed from the prior CCA5 architecture to reflect the duplication of the Navigation Modules.

(*action*$_t$ ≠ "move*" and **WPR**$_t$ ≠ ["discard*"]) or **WPR**$_t$ = ["feedback*"],

⇒ `Nav_ModA.feedback_to_assocn_mod(`**WNM'**$_t$`)` (88)

⇒ $\forall_\sigma$: **LNM**$_{(\sigma, \Upsilon, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`**WNM'**$_{t-1}$`)` (89)

(*action*$_t$ ≠ "move*" and **WPR**$_t$ ≠ ["discard*"]) or **WPR**$_t$ = ["feedback*"], ⇒	-if *action*$_t$ which the Navigation Module A produced (82) is not actionable (i.e., does not specify an activation of an actuator or the transmission of an electronic signal) and also there is no specification by the Working Primitive **WPR**$_t$ to discard *action*$_t$ (i.e., the Navigation Module A should not do anything this cognitive cycle) then operations after the ⇒ symbol will occur -if Working Primitive **WPR**$_t$ specifies that the intermediate results of the Navigation Module A should be fed back, then regardless of whether *action*$_t$ is actionable or not, the operations after the ⇒ symbol will occur
⇒`Nav_ModA.feedback_to_assocn_mod(`**WNM'**$_t$`)`	-pseudocode to feed back and store the intermediate results of the Navigation Module A, i.e., the Working Navigation Map **WNM'**$_t$ to the Input Sensory Vectors Association Modules
⇒$\forall_\sigma$: **LNM**$_{(\sigma, \Upsilon, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`**WNM'**$_{t-1}$`)`	-pseudocode that specifies that the Working Navigation Map just fed back (i.e., **WNM'**$_{t-1}$ which is **WNM'**$_t$ of the previous cognitive cycle) is to have its components extracted (i.e., effectively stored) into the best matching local navigation maps of the various Input Sensory Vectors Associations Modules -thus, as the cognitive cycle progresses, the actual sensory inputs will be ignored, and instead these components of the previous Working Navigation Map **WNM'**$_{t-1}$ (i.e., the intermediate results from the Navigation Module A in the last cognitive cycle) will be propagated towards the Navigation Module A and operated on again

Table A16. Explanation of Symbols and Pseudocode in Equations (88) – (89)

Initial Input:	*not applicable as this is a continuation of Navigation Module A operations*
Initial Output:	**WNM'**$_t$: the current Working Navigation Map is fed back to the Input Sensory Vectors Association Modules
Second Input:	in the next cognitive cycle **LNM**$_{(\sigma, \Upsilon, t)}$ for some or all of the sensory systems, derived from the fed back **WNM'**$_{t-1}$, are treated as inputs and processed again by the Navigation Module A complex (i.e., the Object Segmentation Gateway Module, the Causal Memory Module and the Navigation Module A)
Second Output:	-in the next cognitive cycle, the Navigation Module A will produce again an *action*$_t$ signal as per equation (82) -if *action*$_t$ is actionable (i.e., an activation of an actuator or the transmission of an electronic signal) then the *action*$_t$ signal will be sent to the Output Vector Association Module A, as in the previous section -if *action*$_t$ is not actionable then intermediate results of the Navigation Module A (i.e., **WNM'**$_t$) may be fed back again to the Input Sensory Vectors Association Modules (88, 89)
Description:	-In the previous section it was seen that the Navigation Module A applies the Working Primitive **WPR**$_t$ against the current Working Navigation Map **WNM'** and produced an *action*$_t$ signal which was sent to the Output Vector Association Module A and then to the external embodiment to activate an actuator or send an electronic signal.

	-In this section it is seen that if $action_t$ is not actionable then intermediate results of the Navigation Module A (i.e., $\mathbf{WNM'}_t$) may be fed back again to the Input Sensory Vectors Association Modules and then in the next cognitive cycle return back to the Navigation Module A where it can be operated on again. -By feeding back and re-operating on the intermediate results, the architecture can generate causal behavior.

Table A17. Summary of the Feedback Operations from the Navigation Module A per Equations (88) – (89)

A.9 Analogical Problem Solving in the CCA6

In the last section above, it was shown how with enhanced feedback pathways from the Navigation Module A to the Input Sensory Vectors Association Modules (Figure 3) and some small changes, if there is no actionable output from the Navigation Module A, then the Navigation Module A can feed back the Working Navigation Map **WNM'**. Without many changes to the architecture, when this happens then in the next cognitive cycle these intermediate results will automatically be considered as the input sensory information and propagated back to the Navigation Module A and operated on again. As discussed in the text of the paper above, it has previously been shown that in this type of Causal Cognitive Architecture that by feeding back and re-operating on the intermediate results, the architecture is sometimes able to formulate and explore possible cause and effect of actions, i.e., generate causal behavior.

However, often the combinations of sensory inputs leading to **WNM'** and the chosen instinctive or learned primitives leading to **WPR'** which operates on the Working Navigation Map **WNM'**, does not give a causally related or even a useful output.

This section describes a small, evolutionarily plausible modification of the feedback algorithm from the Navigation Module A to the Input Sensory Vectors Association Modules which readily emerges. The result is the ready generation of analogical results that may be more useful than simply feeding back and returning the intermediate results unchanged in the next cognitive cycle.

In the last section above it was seen in (88) that if the $action_t$ which is produced by the Navigation Module A (i.e., $action_t$ = `Nav_ModA.apply_primitive`($\mathbf{WPR}_t$, $\mathbf{WNM'}_t$) (82)) is not actionable (i.e., there is no actuator to move or no electronic signal to send) then the Navigation Module A feeds back what will now be considered intermediate results of the Navigation Module A (i.e., $\mathbf{WNM'}_t$) for storage in the Input Sensory Vectors Association Modules. (A property which can easily emerge by simply enhancing feedback pathways in the architecture.) In the next cognitive cycle, $\mathbf{WNM'}_t$ is treated as the input signal (89) and thus automatically propagated back to the Navigation Module A where it can be operated on. Perhaps different instinctive or learned primitives operating on $\mathbf{WNM'}_{t-1}$ ("t - 1" just means it is from the previous cognitive cycle) or the transformation of the data, will produce a useful, actionable output during this cognitive cycle. Equations (88) and (89) are reproduced again below. Note that the Working Primitive $\mathbf{WPR}_t$ can specify to discard results (i.e., $\mathbf{WPR}_t$ = ["discard*"]) after a cognitive cycle (i.e., nothing will happen except wait for the next cognitive cycle) or it can specify to feed back the results of the Navigation Module A even if the action is actionable (i.e., $\mathbf{WPR}_t$ = ["feedback*"]).

($action_t \neq$ "move*" and $\mathbf{WPR}_t \neq$ ["discard*"]) or $\mathbf{WPR}_t$ = ["feedback*"],
⇒ `Nav_ModA.feedback_to_assocn_mod`($\mathbf{WNM'}_t$) (88)
⇒ $\forall_\sigma$: $\mathbf{LNM}_{(\sigma, \gamma, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`$\mathbf{WNM'}_{t-1}$`)` (89)

These small changes in the architecture (which occurred in the CCA5 version of the architecture) are sufficient to easily allow the emergence of a different type of feedback by the Navigation Module A. The feedback shown in (88) and (89) will no longer occur by default. However, if the Working Primitive $\mathbf{WPR}_t$ specifically signals "feedback" then the same feedback algorithm as before (i.e., described above (88, 89)) will occur. Equations (88, 89) are replaced by (88a, 89) to specify this condition where the previously described feedback algorithm will still occur:

$\mathbf{WPR}_t$ = ["feedback*"],
⇒ `Nav_ModA.feedback_to_assocn_mod`($\mathbf{WNM'}_t$) (88a)
⇒ $\forall_\sigma$: $\mathbf{LNM}_{(\sigma, \gamma, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`$\mathbf{WNM'}_{t-1}$`)` (89)

The new default feedback algorithm which will occur is now specified as starting in (90). If the $action_t$ signal produced by the Navigation Module A (i.e., $action_t$ = `Nav_ModA.apply_primitive`(**WPR**$_t$, **WNM'**$_t$) (82)) is not actionable ($action_t \neq$ " move*"—there is no actuator to move or no electronic signal to send) then fed back are what will now be considered intermediate results of the Navigation Module A (i.e., **WNM'**$_t$) for storage in the Input Sensory Vectors Association Modules (90). This is the same effect as occurred before and still occurs in (88a). (Note also from (90) that if the Working Primitive **WPR**$_t$ specifies "analogical" feedback then this will occur even if the $action_t$ signal is actionable. Similarly, if **WPR**$_t$ specifies to "discard" the results, then nothing will happen except wait for the next cognitive cycle.) Thus, **WNM'**$_t$ is sent (i.e., fed back) to the Input Sensory Vectors Association Modules as before (90).

WNM'$_t$ is also sent to the Causal Memory Module (91). Sending **WNM'**$_t$ to the Causal Memory Module (91) triggers `Causal_Mem_Mod.match_best_map` (**WNM'**$_t$). This pseudocode matches **WNM'**$_t$ against the multisensory navigation maps stored in the Causal Memory Module, and assigns this best matching navigation map as the new **WNM'**$_t$ value. This matching algorithm is similar to other matching algorithms for navigation maps used by the architecture. **WNM'**$_t$ is also sent to the TempMap region of the Navigation Module A. This happens automatically. This was a change in the previous CCA5 version of the architecture. Results of almost all operations from the Navigation Module A and the Causal Memory Module are routinely copied to the TempMap region. Often the information in the TempMap region (i.e., "TempMapA", the TempMap region in Navigation Module A) is not used and it is overwritten in the next cognitive cycle. This is the case now—the contents of the TempMap region will not be used. However, in later steps in equations (92, 93) the TempMap region is indeed used. Note that in the equations the TempMap region is defined as an array **TempMap** which acts like an ordinary navigation map (66).

In (92) the pseudocode `Nav_ModA.use_linkaddress1_map`(**WNM'**$_t$) activates in the Causal Memory Module the navigation map which the most recently used linkaddress used in the past by the current Working Navigation Map **WNM'**$_t$ points to. As shown in (92) this result of the Causal Memory Module is automatically copied to **TempMap**$_t$. (As mentioned above results of the Navigation Module A and the Causal Memory Module are routinely and automatically copied to **TempMap**.)

(Note that other algorithms besides `use_linkaddress1_map()` are possible. For example, rather than the most recently used linkaddress, a number of the linkaddresses used by this navigation map in the past can be explored and one particular linkaddress chosen, and so on. Similarly, most recently produced *action*'s can also be explored.)

In (93) the pseudocode `Nav_ModA.subtract` (**TempMap**, **WNM'**$_t$) the navigation map **WNM'**$_t$ is subtracted from **TempMap**. (The navigation maps are arrays and can be treated as such via simple array operations.) The result becomes the new Working Navigation Map **WNM'**$_t$ (93).

The next cognitive cycle then occurs. At this point, the contents of the **WNM'**$_t$ Working Navigation Map in the Navigation Module A contains the difference between the previous **WNM'**$_t$ and retrieved ***linkaddress χ'*** Navigation Map stored in **TempMap**. At this point, the Input Sensory Vectors Association Modules contain the original Working Navigation Map **WNM'**$_{t-1}$ (*t*-1 just means something from the previous cognitive cycle) which was fed back and stored here. Thus, during this cognitive cycle, the actual sensory inputs will be ignored, and the previous **WNM'**$_{t-1}$ will be propagated towards the Navigation Module A.

To indicate continuation of the operations on associated equations (90–93), in (94) it is specified "($action_{t-1} \neq$ "move*" or **WPR**$_{t-1}$ = ["analogical*"]) and **WPR**$_{t-1} \neq$ ["discard*"] and **WPR**$_{t-1} \neq$ ["feedback*"])**,**" and then specified to run this specific pseudocode: **WNM'**$_t$ = `Nav_ModA.retrieve_and_add_vector_assocn()` (94). Rather than propagate **WNM'**$_{t-1}$ towards the Navigation Module A and then it replace the existing **WNM'**$_t$ and is operated on by the current Working Primitive **WPR'** , as occurred automatically with the previous feedback algorithm above, the pseudocode `retrieve_and_add_vector_assocn()` specifies that **WNM'**$_{t-1}$ should not replace but simply be added to the existing **WNM'**$_t$.

Now the original Working Navigation Map **WNM'**$_t$ in the Navigation Module A also contains the action that occurred in the past of a similar Working Navigation Map in a possible analogical situation.

These equations in the CCA6 architecture are somewhat changed from the prior CCA5 architecture to reflect the duplication of the Navigation Modules.

((*action*$_t$ ≠" move*" or **WPR**$_t$ = ["analogical*"]) and **WPR**$_t$ ≠ ["discard*"] and **WPR**$_t$ ≠ ["feedback*"]),
⇒`Nav_ModA.feedback_to_assocn_mod(`**WNM'**$_t$`)` (90)
⇒ **WNM'**$_t$ = `Causal_Mem_Mod.match_best_map(`**WNM'**$_t$`)` (91)
⇒ **TempMap**$_t$ = `Nav_ModA.use_linkaddress1_map(`**WNM'**$_t$`)` (92)
⇒ **WNM'**$_t$ = `Nav_ModA.subtract(`**WNM'**$_t$, **TempMap**`)` (93)

((*action*$_{t-1}$ ≠ "move*" or **WPR**$_{t-1}$ = ["analogical*"]) and **WPR**$_{t-1}$ ≠ ["discard*"] and **WPR**$_{t-1}$ ≠ ["feedback*"]),
⇒ **WNM'**$_t$ = `Nav_ModA.retrieve_and_add_vector_assocn()` (94)

WPR$_t$ = ["feedback*"], ⇒	-replaces equation (88) (see Table A17) -previously described feedback algorithm now only occurs if the Working Primitive **WPR**$_t$ specifies it (i.e., **WPR**$_t$ = ["feedback*"],).
⇒`Nav_ModA.feedback_to_assocn_mod(`**WNM'**$_t$`)`	-previously described in Table A17 -applied to equations (88)/(88a) or (90) -pseudocode to feed back and store the intermediate results of the Navigation Module A, i.e., the Working Navigation Map **WNM'**$_t$ in the Input Sensory Vectors Association Modules
⇒$\forall_\sigma$: **LNM**$_{(\sigma, \gamma, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`**WNM'**$_{t-1}$`)`	-previously described in Table A17 -applies to equation (89) -pseudocode that specifies that the Working Navigation Map just fed back (i.e., **WNM'**$_{t-1}$ which is **WNM'**$_t$ of the previous cognitive cycle) is to have its components extracted (i.e., effectively stored) into the best matching local navigation maps of the various Input Sensory Vectors Associations Modules -thus, as the cognitive cycle progresses, the actual sensory inputs will be ignored, and instead these components of the previous Working Navigation Map **WNM'**$_{t-1}$ (i.e., the intermediate results from the Navigation Module A in the last cognitive cycle) will be propagated towards the Navigation Module A and operated on again
((*action*$_t$ ≠" move*" or **WPR**$_t$ = ["analogical*"]) and **WPR**$_t$ ≠ ["discard*"] and **WPR**$_t$ ≠ ["feedback*"]), ⇒	-**if** the *action*$_t$ signal produced by the Navigation Module A (i.e., *action*$_t$ = `Nav_ModA.apply_primitive(`**WPR**$_t$, **WNM'**$_t$`)` (82)) is not actionable (*action*$_t$ ≠" move*"—there is no actuator to move or no electronic signal to send) -**or** if the Working Primitive **WPR**$_t$ specifies "analogical" feedback (regardless if the *action*$_t$ signal is actionable) -**and** the Working Primitive **WPR**$_t$ does not specify to "discard" the results (regardless if the *action*$_t$ signal is actionable, if "discard" is specified then nothing happens except to wait for the next cognitive cycle) -**and** the Working Primitive **WPR**$_t$ does not specify to use the previous "feedback" pseudocode (i.e., equations (88, 89)) (regardless if the *action*$_t$ signal is actionable) -**then** equations/pseudocode which follows—which are equations (90,91,92, 93)—will be run
⇒`Nav_ModA.feedback_to_assocn_mod(`**WNM'**$_t$`)`	-equation (90) is the same pseudocode as equations (88)/ (88a) -pseudocode to feed back and store the intermediate results of the Navigation Module A, i.e., the Working Navigation Map **WNM'**$_t$ in the Input Sensory Vectors Association Modules
⇒`Causal_Mem_Mod.match_best_map(`**WNM'**$_t$`)` → **WNM'**$_t$	-**WNM'**$_t$ is also sent to the Causal Memory Module (91) -sending **WNM'**$_t$ to the Causal Memory Module (91) triggers `Causal_Mem_Mod. match_best_map (`**WNM'**$_t$`)`

	-this pseudocode matches **WNM'**$_t$ against the multisensory navigation maps stored in the Causal Memory Module, and assigns this best matching navigation map as the new **WNM'**$_t$ value
⇒`Nav_ModA.use_linkaddress1_map(`**WNM'**$_t$`)` → **TempMap**$_t$	-the pseudocode `Nav_ModA.use_linkaddress1_map(`**WNM'**$_t$`)` activates in the Causal Memory Module the navigation map which the most recently used linkaddress used in the past by the current Working Navigation Map **WNM'**$_t$ points to (92) -the result of the Causal Memory Module is automatically copied to **TempMap**$_t$. -as noted earlier, in the CCA6 version of the architecture, results of the Navigation Module A and the Causal Memory Module are routinely and automatically copied to **TempMap**
⇒`Nav_ModA.subtract(`**WNM'**$_t$, **TempMap**`)` (93) → **WNM'**$_t$	-the pseudocode `Nav_ModA.subtract(`**TempMap**, **WNM'**$_t$`)` subtracts the navigation map **WNM'**$_t$ from the navigation map in **TempMap** (93) -the result becomes the new Working Navigation Map **WNM'**$_t$
((*action*$_{t-1}$ ≠ "move*" or **WPR**$_{t-1}$ = ["analogical*"]) and **WPR**$_{t-1}$ ≠ ["discard*"] and **WPR**$_{t-1}$≠ ["feedback*"])**,** ⇒	-"*t-1*" simply refers to the cognitive cycle before this one, i.e., to the values from the previous cognitive cycle -if true then the pseudocode of equation (94) is run -if equation (90)'s conditions are true and its pseudocode is run, then this will also be true (exact same conditions except referring to the previous cognitive cycle), thus equations (90) to (94) are run one after another
⇒`Nav_ModA.retrieve_and_add_vector_assocn()` (94) → **WNM'**$_t$	-rather than propagate **WNM'**$_{t-1}$ (which is in the Sensory Input Vectors Associations Modules, and will utilized in the new cognitive cycle, rather than the actual sensory inputs) towards the Navigation Module A and then automatically replace the existing **WNM'**$_t$, the pseudocode specifies that **WNM'**$_{t-1}$ should not replace but simply be added to the existing **WNM'**$_t$ -thus, now the original Working Navigation Map **WNM'**$_t$ in the Navigation Module A will contain the action or navigation map that occurred in the past of a similar Working Navigation Map in a possible analogical situation

Table A18. Explanation of Symbols and Pseudocode in Equations (88a) – (94)

Input:	*not applicable as this is a continuation of Navigation Module A operations*
Temporary Inputs/Outputs occurring:	***note:*** *to distinguish between the several different navigation maps which are considered the current Working Navigation Map **WNM'** at some point, descriptive suffixes are added to **WNM'*** *(however, note that there is only one navigation map in the Navigation Module A at one time designated as **WNM'**; the descriptive suffixes are for the reader, they do not exist in the architecture)* **-WNM'**$_t$*-original*: the original current Working Navigation Map is fed back to the Input Sensory Vectors Association Modules (90) - **WNM'**$_t$*-original*: the original current Working Navigation Map is propagated to the Causal Memory Module and the resultant best matching navigation map is sent back to the Navigation Module A and replaces **WNM'**$_t$*-original* with **WNM'**$_t$*-best_match* (91) - **WNM'**$_t$*-best_match-linkaddress1*: its most recently used ***linkaddress*** (i.e., pointing and accessing another navigation map) is sent to the Causal Memory Module and puts the retrieved (actually just activated) navigation map into **TempMap** (92) -**TempMap**: holding the retrieved navigation map **WNM'**$_t$*-best_match-linkaddress1* (92), which will then subtract the current Working Navigation Map **WNM'**$_t$*-best_match* and the result will be the new Working Navigation Map **WNM'**$_t$*-difference* (93) -in the next cognitive cycle **LNM**$_{(\sigma,\Upsilon,t)}$ for some or all of the sensory systems, derived from the fed back **WNM'**$_{t-1}$*-original*, are treated as inputs and added to the existing Working Navigation Map **WNM'**$_t$*-difference* resulting in **WNM'**$_t$*-analogical* (94)
Output:	*not applicable as this is a continuation of Navigation Module A operations*
Description:	-Earlier it was seen that the Navigation Module A applies the Working Primitive **WPR**$_t$ against the current Working Navigation Map **WNM'** and produced an *action*$_t$ signal which was sent to the

	Output Vector Association Module A and then to the external embodiment to activate an actuator or send an electronic signal. -In the previous section it was seen that if $action_t$ is not actionable then intermediate results of the Navigation Module A (i.e., **WNM'**$_t$) may be fed back again to the Input Sensory Vectors Association Modules and then in the next cognitive cycle return back to the Navigation Module A where it can be operated on again. By feeding back and re-operating on the intermediate results, the architecture can generate causal behavior. -In this section shown is a different "analogical" feedback algorithm to use if $action_t$ is not actionable. -The analogical feedback algorithm in this section starts off similarly to the previous feedback algorithm, but rather than simply feeding back the Working Navigation Map **WNM'**$_t$ unchanged to be processed further in the next cognitive cycle, it feeds back and constructs a navigation map that occurred in the past of a similar Working Navigation Map **WNM'**$_t$ (which can call "**WNM'**$_t$-*analogical*" to distinguish it from the other navigation maps that are set as **WNM'**$_t$ at different points). -**WNM'**$_t$-*analogical* may be more likely in a possible analogical situation. This navigation map can then be processed further in the next cognitive cycle. -By using a possibly analogical navigation map it makes it more likely the navigation map which becomes the Working Navigation Map **WNM**$_t$ in the next cognitive cycle, and upon which possibly another (or possibly the same) **WPR**$_t$ will be applied, that an actionable $action_t$ will result since it is using a navigation map that was tried in the past. -At the present, the algorithm underlying the pseudocode `Nav_ModA.use_linkaddress1_map(`**WNM'**$_t$) (92) is simple and straightforward. However, in future versions more sophisticated algorithms for this pseudocode could choose better among matching navigation maps that were more successful in the past in producing an actionable *action*. -The next section will show that induction by analogy is actually occurring in this process.

Table A19. Summary of the Analogical Feedback Operations from the Navigation Module A per Equations (88a) – (94)

A.10 Induction by Analogy

Below it is shown that the movement and comparison of navigation maps in equations (90 – 94) allows induction by analogy to occur. These equations are re-written using the more descriptive terms for the Working Navigation Maps used in Table A19—to distinguish between the several different navigation maps which are considered the current Working Navigation Map **WNM'** at some point, descriptive suffixes are added to **WNM'**. These descriptive suffixes are for the reader, they do not exist in the architecture.

(($action_t \neq$ " move*" or **WPR**$_t$ = ["analogical*"]) and **WPR**$_t \neq$ ["discard*"] and **WPR**$_t \neq$ ["feedback*"]),
⇒`Nav_ModA.feedback_to_assocn_mod(`**WNM'**$_t$-*original*`)` (95) *(from 90)*
⇒ **WNM'**$_t$-*best_match* = `Causal_Mem_Mod.match_best_map(`**WNM'**$_t$-*original*`)` (96) *(from 91)*
⇒ **TempMap**$_t$ = `Nav_ModA.use_linkaddress1_map(`**WNM'**$_t$-*best_match*`)` (97) *(from 92)*
⇒ **WNM'**$_t$-*difference* = `Nav_ModA.subtract(`**WNM'**$_t$-*best_match* , **TempMap**$_t$`)` (98) *(from 93)*

(($action_{t-1} \neq$ "move*" or **WPR**$_{t-1}$ = ["analogical*"]) and **WPR**$_{t-1} \neq$ ["discard*"] and **WPR**$_{t-1} \neq$ ["feedback*"]),
⇒ **WNM'**$_t$-*analogical* = `Nav_ModA.retrieve_and_add_vector_assocn()` (99) *(from 94)*

Consider a definition of induction by analogy. There are two variables **x** and **y**. Variable **x** has properties P_1, P_2, P_3, P_4, … P_n (100). Variable **y** also has properties P_1, P_2, P_3, P_4, … P_n (101). It happens that variable **y** has another property N (102). Therefore in (103) it can be concluded by induction by analogy that variable **x** also has property N.

In (95) consider **WNM'**$_t$-*original* as variable **x**, or perhaps as navigation map **x**. It is desired to know what this navigation map **x** will do next, i.e., which navigation map will it call. Consider variable **y**, or perhaps named as navigation map **y**, as referring to (96) **WNM'**$_t$-*best_match*. It is the best matching navigation map to navigation map **x** and thus assumed it will share many properties. Next there is exploration of what navigation map **y** does next (i.e., what navigation map does the ***linkaddress*** chosen link to). Navigation map **y** calls the navigation map in **TempMap**

(97) and that the difference between navigation map **y** and **TempMap** is **WNM'**$_t$*-difference* (98). Thus, consider this difference, i.e., **WNM'**$_t$*-difference* to be property N. Since navigation map **y** has property N, therefore by induction by analogy, it can be said that navigation map **x** also has property N (103). Thus, add property N, which is actually **WNM'**$_t$*-difference*, to navigation map **x**, which is actually **WNM'**$_t$*-original*, producing the result of navigation map **x** with property N as being **WNM'**$_t$*-analogical* (99).

$P_1\mathbf{x}$ **&** $P_2\mathbf{x}$ **& …** $P_n\mathbf{x}$ (100)
$P_1\mathbf{y}$ **&** $P_2\mathbf{y}$ **& …** $P_n\mathbf{y}$ (101)
N**y** (102)
∴ N**x** □ (103)

It is important to note that analogical problem solving is not a separate module in the architecture (for example, to be used when solving intelligence tests or difficult problems) but rather part of the core mechanism of day-to-day functioning of the architecture. These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture although there are small modifications to account for the duplicate Navigation Module. Note that equations (i) to (ix) in the text of paper above are easy-to-read versions of equations (95) to (103).

((*action*$_t$ ≠" move*" or **WPR**$_t$ = ["analogical*"]) and **WPR**$_t$ ≠ ["discard*"] and **WPR**$_t$ ≠ ["feedback*"]), ⇒	-see Table A18
⇒`Nav_ModA.feedback_to_assocn_mod`(**WNM'**$_t$*-original*) (95)	-see Table A18 - **WNM'**$_t$*-original* – the current Working Navigation Map **WNM'**$_t$ which is given the label 'original' for better understanding of what is happening
⇒`Causal_Mem_Mod.match_best_map`(**WNM'**$_t$*-original*) (96) → **WNM'**$_t$*-best_match*	-see Table A18 **WNM'**$_t$*-original* – the current Working Navigation Map **WNM'**$_t$ which is given the label 'original' for better understanding of what is happening
⇒`Nav_ModA.use_linkaddress1_map`(**WNM'**$_t$*-best_match*) (97) → **TempMap**$_t$	-see Table A18 **-WNM'**$_t$*-best_match* – the current Working Navigation Map **WNM'**$_t$ which is given the label 'best-match' for better understanding of what is happening
⇒`Nav_ModA.subtract`(**WNM'**$_t$*-best_match* , **TempMap**$_t$) (98) → **WNM'**$_t$*-difference*	-see Table A18 - **WNM'**$_t$*-difference* – the current Working Navigation Map **WNM'**$_t$ which is given the label 'difference' for better understanding of what is happening
((*action*$_{t-1}$ ≠ "move*" or **WPR**$_{t-1}$ = ["analogical*"]) and **WPR**$_{t-1}$ ≠ ["discard*"] and **WPR**$_{t-1}$≠ ["feedback*"]), ⇒	-see Table A18
⇒ **WNM'**$_t$*-analogical* = `Nav_ModA.retrieve_and_add_vector_assocn()` (99) → **WNM'**$_t$*-analogical*	-see Table A18 - **WNM'**$_t$*-analogical* – the current Working Navigation Map **WNM'**$_t$ which is given the label 'analogical' for better understanding of what is happening
$P_1\mathbf{x}$ **&** $P_2\mathbf{x}$ **& …** $P_n\mathbf{x}$ (100)	-explanation of definition of induction by analogy - variable (or navigation map) **x** has properties P_1, P_2, P_3, P_4, … P_n
$P_1\mathbf{y}$ **&** $P_2\mathbf{y}$ **& …** $P_n\mathbf{y}$ (101)	-explanation of definition of induction by analogy -variable (or navigation map) **y** also has properties P_1, P_2, P_3, P_4, … P_n
N**y** (102)	-explanation of definition of induction by analogy - variable (or navigation map) **y** also has property N

∴ N**x** □ (103)	-explanation of definition of induction by analogy - thus, can conclude by induction by analogy that variable (navigation map) **x** also has property N

Table A20. Explanation of Symbols and Pseudocode in Equations (95) – (103)

Input:	*not applicable as this is a continuation of Navigation Module A operations*
Output:	- **WNM'**$_t$*-analogical* (99)
Description:	-See Table A18. Equations (95 – 99) are re-write of equations (90 – 94) using the more descriptive terms for the Working Navigation Maps. These descriptive suffixes are for the reader, they do not exist in the architecture. -To distinguish between the several different navigation maps which are considered the current Working Navigation Map **WNM'** at some point, descriptive suffixes are added to **WNM'** listed above. - Equations (100 – 103) consider a definition of induction by analogy applied to the operations performed on the Working Navigation Maps.

Table A21. Summary of the Analogical Feedback Operations from the Navigation Module A per Equations (95) – (103)

A.11 Grounding

Like the prior CCA5 architecture, the CCA6 architecture provides a solution to the grounding problem—information enters the system through experiential sensory systems. Information will automatically be mapped to real-world sensations and actions. No information is kept as an isolated symbol—*everything* in the CCA6 is within a navigation map linked to other navigation maps.

A *feature* was defined initially in (34). Cells in navigation maps can contain *feature*s, *procedure*s, or ***linkaddresses***. Below equation (104) defines a *grounded_feature* as being any *feature* such that the *feature* is in some sensory system Local Navigation Map ***all_LNMs*$_\chi$** and that the *feature* also was present in some sensory system array $\mathbf{S}_{\sigma,t}$ (1–7) represented by ***s***(*t*) (9). This implies that the CCA6 experientially acquired this *feature*, and it is in its navigation maps associated with other sensory inputs and possibly associated with previous or upcoming *feature*s. The symbol grounding problem is largely resolved by grounding symbols with their real-world sensations, interactions, and linkages.

However, the second part of equation (104) deals with the case where information has been acquired by electronic transfer or by the equivalent of human memorization, or possibly an experiential acquisition of the *feature* was poor, and grounding is poor. A *feature* is considered as a *grounded_feature* if the *feature* is present in a Working Navigation Map **WNM'**$_t$ and in the previous cognitive cycle the analogical feedback mechanism was used. Note that if in the previous cognitive cycle (*t-1*) the action was not actionable, i.e., $action_{t-1} \neq$ "move*", then the analogical feedback mechanism will occur, or if the previous Working Primitive $\mathbf{WPR}_{t-1}$ specified an analogical feedback loop, then the analogical feedback mechanism will occur.

Equation (105) essentially states that any cell in any of the addressable navigation maps (e.g., it will not include, for example, the specialized navigation maps in the Sequential/Error Correcting Module) will either contain a *grounded_feature* or contain a ***linkaddress*** pointing to another cell which most likely will be in another navigation map, or else the cell has no features.

With regard to accepting as grounded a ***linkaddress*** pointing to another cell/navigation map, it is assumed that there are so many links pointing to so many other navigation maps in a chain of ***linkaddresses*** that if a cell points to another cell most likely in another navigation map, it will be grounded there by association with some valid grounding condition (e.g., sensations, actions, etc.) or else point to another navigation map where it will be ground there, and so on. There may be differences in the quality of the grounding as such, but it will always be grounded to some extent.

These equations remain largely unchanged in the CCA6 architecture from the prior CCA5 architecture.

$$grounded\_feature = \forall_{feature}: (feature \in \mathbf{all\_LNMs}_\chi \text{ AND } feature \in \mathbf{s}(t))$$

OR

$\forall_{feature}$: ((*feature* ∈ **WNM'**$_t$ AND $action_{t-1} \neq$ "move*" AND $\mathbf{WPR}_{t-1} \neq$ ["feedback*"]) OR $\mathbf{WPR}_{t-1}$ = ["analogical*"]) (104)

$\forall_{\chi,t}$: $\boldsymbol{all\_navmaps_{\chi,t}}$ = *grounded_feature* OR `link`$\boldsymbol{(all\_navmaps_{\chi,t})} \neq$ [ ] OR $\boldsymbol{cellfeatures_{\chi,t}}$ = [ ] (105)

feature	- *feature* ∈ R (34) - some arbitrary real number representing a *feature* modality and value - for example, $feature_{13,(\chi \text{modcode=*Causal\_Memory\_Mod*, } mapno=3456,2,3,4)}$ would be *feature* number 13 in the cell x=2,y=3,z=4 in map number 3456 in the Causal Memory Module; its value, for example, could represent a visual line
grounded_feature	- any *feature* such that the *feature* is in some sensory system Local Navigation Map $\boldsymbol{all\_LNMs_{\chi}}$ and that the *feature* also was present in some sensory system array $\mathbf{S}_{\sigma,t}$ (1–7) represented by $\boldsymbol{s}(t)$ (9) - or if the *feature* has been acquired by electronic transfer or by the equivalent of human memorization then consider a *feature* as a *grounded_feature* if the *feature* is present in a Working Navigation Map **WNM'**$_t$ and in the previous cognitive cycle the analogical feedback mechanism was used
$\forall_{feature}$: (*feature* ∈ $\boldsymbol{all\_LNMs_{\chi}}$ AND *feature* ∈ $\boldsymbol{s}(t)$) → *grounded_feature*	- any *feature* such that the *feature* is in some sensory system Local Navigation Map $\boldsymbol{all\_LNMs_{\chi}}$ and that the *feature* also was present in some sensory system array $\mathbf{S}_{\sigma,t}$ (1–7) represented by $\boldsymbol{s}(t)$ (9)
$\forall_{feature}$: ((*feature* ∈ **WNM'**$_t$ AND $action_{t-1} \neq$ "move*" AND $\mathbf{WPR}_{t-1} \neq$ ["feedback*"]) OR $\mathbf{WPR}_{t-1}$ = ["analogical*"]) → *grounded_feature*	- for example, a *feature* that has been acquired by electronic transfer or by the equivalent of human memorization, then the analogical feedback mechanism will occur (when the *feature* is considered), and consider that *feature* as a *grounded_feature* - although the older feedback mechanism *may* result in grounding the feature adequately, often it won't, thus requiring utilization of the analogical feedback mechanism to ensure grounding
$\boldsymbol{cellfeatures_{\chi,t}}$ = [ ]	-$\boldsymbol{cellfeatures_{\chi,t}}$ are all the features within a given cell χ (38) -thus, $\boldsymbol{cellfeatures_{\chi,t}}$ = [ ] means there are no features in that cell at address χ
$\forall_{\chi,t}$: $\boldsymbol{all\_navmaps_{\chi,t}}$ = *grounded_feature* OR `link`$\boldsymbol{(all\_navmaps_{\chi,t})} \neq$ [ ] OR $\boldsymbol{cellfeatures_{\chi,t}}$ = [ ]	- any cell in any of the addressable navigation maps will either contain a *grounded_feature*, or contain a ***linkaddress*** pointing to another cell which most likely will be in another navigation map, or else the cell has no features

Table A22. Explanation of Symbols and Pseudocode in Equations (104) – (105)

Input:	*not applicable as this is a continuation of Navigation Module A operations*
Output:	*not applicable as this is a continuation of Navigation Module A operations*
Description:	-These equations specify that for every addressable navigation map in the Causal Cognitive Architecture 6 (CCA6), *feature*s can be considered to be grounded or to be used in a grounded fashion.

Table A23. Summary of the Grounding Requirements with the Architecture per Equations (104) – (105)

A.12 Simple Language Generation

The generation of simple language (i.e., as opposed to full compositional language) was discussed in the previous Causal Cognitive Architectures, including the previous CCA5 one. To a large extent, some way of communicating what is in a navigation map(s) from one CCA6 embodiment to another CCA6 embodiment is required. In the CCA5 architecture this was accomplished with a straightforward instinctive primitive named `apply_primitive_nav_to_protolang()`.

In the CCA6 while compositional language comprehension and behavior makes use of Navigation Module B, as was illustrated in the text and figures of the main body of the paper, language expression remains simple (i.e., essentially a simple read-back of what is on the navigation maps is being communicated) and still remains generated by Navigation Module A. In future versions of the architecture, full compositional generation of language is expected.

In (106) `Nav_ModA.apply_primitive_nav_to_protolang` (**WPR**$_t$, **WNM'**$_t$) produces an output action *action*$_t$. As before, per equations (83 – 87), *action*$_t$ is propagated from the Navigation Module A to the Output Vector Association Module A (xxx ~~Figure 14~~). As before, there is motion correction of the output signal with interactions with the Sequential/Error Correcting Module, and then the output signal is sent to the Output Vector Association Module A and then on to the actual output actuators (Figure 3).

The equation below remains largely unchanged in the CCA6 architecture from the prior CCA5 architecture, although it now specifies the Navigation Module being used.

$$action_t = \texttt{Nav\_ModA.apply\_primitive\_nav\_to\_protolang}\ (\mathbf{WPR}_t, \mathbf{WNM'}_t) \quad (106)$$

`Nav_ModA.apply_primitive_nav_to_protolang`(**WPR**$_t$, **WNM'**$_t$) → *action*$_t$	-`Nav_Mod` refers to the Navigation Module A (xxx ~~Figure 14~~) -`apply_primitive_nav_to_protolang` (**WPR**$_t$, **WNM'**$_t$) is pseudocode for an algorithm which applies the Working Primitive **WPR**$_t$ to the Working Navigation Map **WNM'**, such that the current **WNM'** and sequence of **WNM'** Working Navigation Maps preceding it (i.e., what happened? why did it happen?) are converted into a simple proto-language which is represented in the *action*$_t$ value -this is a simple read-back (limited verbs used) of the navigation maps - the *action*$_t$ value will be transformed into an output signal that activates the appropriate actuators to communicate this simple proto-language communication to another CCA6 embodiment or a human - see Table A14 for more details of converting **WPR**$_t$ **and WNM'**$_t$ into an actionable output, which in this case represents a communication

Table A24. Explanation of Symbols and Pseudocode in Equation (106)

Input:	**WPR**$_t$: the Working Primitive **WPR** is the primitive that will be applied against the Working Navigation Map **WNM'** in the Navigation Module A, and with regard to (106) will involve transforming the recently stored (in the Causal Memory Module) Working Navigation Maps **WNM'**s into an explanation and communication of what happened and why it happened, as well as possibly communication of actions, requests, or concepts **WNM'**$_t$: the current Working Navigation Map which is derived from the processed sensory inputs or analogic feedback results
Output:	*action*$_t$: a signal to move some actuator or send an electronic signal e.g., **<move right>** e.g., <**sound 2000Hz**> it is sent to the Output Vector Association Module A (xxxx ~~Figure 14~~) where more detailed instructions are created to effect the required actuator outputs
Description:	-The Navigation Module A applies the Working Primitive **WPR**$_t$ against the current Working Navigation Map **WNM'** in the Navigation Module A, and with regard to (106) will involve transforming the recently stored (in the Causal Memory Module) Working Navigation Maps

	WNM's into an explanation and communication of what happened and why it happened, as well as possibly communication of actions, requests, or concepts.

Table A25. Summary of the Operation of Simple (Non-Compositional) Language Generation per Equation (106)

A.13 Navigation Module B

As seen in Figure 3 there are now in the CCA6 two Navigation Modules—Navigation Module A and Navigation Module B (as opposed to the single Navigation Module in the prior CCA5 architecture, as seen in Figure 2). Similar to Navigation Module A, in Navigation Module B an instinctive primitive or a learned primitive (which themselves are navigation maps) in conjunction with the Working Navigation MapB **WNMB'** (similar to the Working Navigation Map **WNM'** that Navigation Module A operates on) may result in an action signal. This signal goes to the Output Vector Association Module B and to the external embodiment (Figure 3).

Working Navigation Map B **WNMB'** (107) is defined similarly to Working Navigation Map A **WNM'**–both are navigation maps, i.e., treated as arrays. Similarly, as equation (108) shows, any operation that in theory Navigation Module A can perform on a Working Navigation Map, Navigation Module B can also perform a similar operation on a similar Working Navigation Map. This implies that equations (1) to (106) above all also apply to the operations Navigation Module B on Working Navigation Map B **WNMB'**. Indeed, Navigation Module B originates from the duplication of Navigation Module A. However, the duplication is assumed to be somewhat of a partial duplication (not all pathways would have duplicated similarly necessarily) and in the architecture described in the main body of the text above, Navigation Module B is used mainly for compositional processing. Hence, in (108) there are what are called *navA_only_operation*'s—operations which are only performed by Navigation Module A.

While in the architecture described in the main body of the text above, Navigation Module B is used mainly for compositional language comprehension and resultant compositional behavior, in fact, Navigation Module B can be used for much more abstract compositional comparisons of two sets of data in the two Navigation Modules. This is beyond the scope of the present paper. The set of *navA_only_operation*'s can readily be adjusted in different versions of the architecture.

Consider the demonstration example shown in Figure 1. There are a number of shape objects in the sensory scene plus the instruction "place the white triangle on top of the black block which is not near a white cylinder." As described by equations (1) to (106) above, the objects in the sensory scene of Figure 1 will become mapped/matched/updated onto the **WNM'** navigation map in Navigation Module A. This is depicted in Figure 7. The instruction, however, will trigger the instinctive primitive `parse_sentence()` (which is broken down in sub-parts in equations (109) to (111)).

Equation (109) shows that if a sensory input is recognized as an instruction sentence *instruction_sentence* then the instinctive primitive `Nav_ModB.parse_sentence.copy()` maps the instruction sentence onto Working Navigation Map B **WNMB'** in Navigation Module B. This can be seen in Figure 7.

The instinctive primitive `Nav_ModB.parse_sentence.parse()` then parses through the instruction sentence, i.e., Navigation Map B **WNMB'** in Navigation Module B (110). For each word in the instruction sentence `Nav_ModB.parse_sentence.parse.match()` matches the word against the Causal Memory Module (Figure 7) (111). If an action word is found (e.g., "place" in the example shown in Figure 7) then it is mapped to cells in Navigation Map A **WNM'** in Navigation Module A which contain matching features to the cells in the instruction sentence (i.e., Navigation Map B **WNMB'** in Navigation Module B) associated with the action word. In the example in the text of the paper in Figure 7, the subsequent Figures 8 and 9 illustrate this matching.

Equation (112) shows that if a *near_trigger* occurs in parsing, then the instinctive primitive `Nav_ModB.physics_near_object()` is triggered. A *near_trigger* is a word (or other sensory input that does the same thing) related to the geometric concepts of near and not near. Continuing the compositional example of Figure 1, the word "near" acts as a *near_trigger* (Navigation Module B in Figure 7). Since there was a "not" beside it, and then "white, cylinder" this instinctive primitive flags the cell in Navigation Module A (i.e., Navigation Map A **WNM'**) with "cylinder, white" with the flag "not". This is shown in Figure 10. Also, the `physics_near_object()` instinctive primitive will mark "not" in the matched cell and all the adjoining non-null cells. This is shown in Figure 10.

Equation (113) shows that once the instinctive primitive `Nav_ModB.parse_sentence()` has reached the end of the instruction or communication sentence, i.e., there is nothing left to parse, then it looks for cells which have been tagged or marked with tags. Possible associated tags depend on the collection of instinctive primitives, which at present

only has the single option of `place_object()` (but thousands of instinctive and learned primitives are possible here).

Given a <*place*> tag in cell (0,0,0) of Navigation Module A in Figure 11 in the example in the text, the instinctive primitive `Nav_ModA.place_object()` will be triggered (113). Note that this instinctive primitive will operate on Navigation Module A (i.e., Navigation Map A **WNM'** which is the representation of the sensory scene.

The `Nav_ModA.place_object()` instinctive primitive will attempt to move and place an object somewhere. Continuing the example above, the tag <"place"> is most closely associated with "triangle, white"—that is the object that will be placed somewhere. The primitive now looks for other tagged notations such as <"top"> in the navigation map (Figure 11). It will see a <"not"> in the cell with the features of a black block at (2,0,0) and not consider the <"top"> tag in that cell . However, the cell with the features of black block at (4,0,0) has a valid tag such as <"top">. Thus, this primitive will in turn trigger the instinctive primitive `Nav_ModA.move()` to perform the movement of the white triangle to cell (4,0,0) (on top of it if three dimensions were being used) (114).

`Nav_ModA.move()` can be used for many actions of the CCA6. In this example, actions from Navigation Module A cause an action signal to go to the Output Vector Association Module A (Figure 11). The Output Vector Association Module A performs motion planning and motion corrections via feedback from the Sequential/Error Correcting Module. The corrected motion action signal from the Output Vector Association Module A is shown as Arrow J in Figure 11. This signal goes to the Output Vector Shaping Module which produces the actual signals (arrow K, Figure 11) for actuators to move the white triangle to the cell with the black block on the right. Thus, the white triangle is then moved to (and on top of it in 3 dimensions) the black block on the right side.

Then a new cognitive cycle repeats again, finally processing again the actual sensory inputs streaming into the architecture.

$$\mathbf{WNMB'} = \in R^{m \times n \times o \times p} \quad (107)$$

$$\forall\, operation_{NavModA}(),$$
$$(\sim navA\_only\_operation \implies operation_{NavModB}(\mathbf{WNM}_m) = operation_{NavModA}(\mathbf{WNM}_m)) \quad (108)$$

(*instruction_sentence*),
⇒ $\mathbf{WNMB'}_t$ = `Nav_ModB.parse_sentence.copy()` (109)
⇒ `Nav_ModB.parse_sentence.parse(`$\mathbf{WNMB'}_t$`)`, (110)
⇒ `Nav_ModB.parse_sentence.parse.match()` (111)
⇒ *near_trigger*,
⇒`Nav_ModB.physics_near_object()` (112)
⇒ *end_of_communication*,
<*place*>,
⇒ `Nav_ModA.place_object()` (113)
⇒ `Nav_ModA.move()` (114)

WNMB'	-Working Navigation Map B -the current Working Navigation Map B $\mathbf{WNMB'}_t$ is the navigation map which the Navigation Module B focuses its attention on, i.e., applies operations on to make a decision to take some sort or no action
$\forall\, operation_{NavModA}(\mathbf{WNM}_m)$, ($\sim navA\_only\_operation \implies operation_{NavModB}(\mathbf{WNM}_m) = operation_{NavModA}(\mathbf{WNM}_m)$)	-any operation that in theory Navigation Module A can perform on some Working Navigation Map *m* in Navigation Module A $\mathbf{WNM}_m$ ($operation_{NavModA}(\mathbf{WNM}_m)$), Navigation Module B can also perform a similar operation on a similar Working Navigation Map *m* $\mathbf{WNM}_m$ in Navigation Module B -implies that equations (1) to (106) above all also apply to the operations of Navigation Module B on Working Navigation Map B **WNMB'**

	-*navA_only_operation*'s—operations which are only performed by Navigation Module A; set of these nav_modA-only operations can be adjusted for different versions of the architecture -in CCA6 Navigation Module B largely used for compositional language comprehension and behavior
instruction_sentence	-a sensory input which triggers recognition as an input instruction sentence by the Input Sensory Vectors Association Module and/or Instinctive Primitives and/or Learned Primitives
WNMB'$_t$ = `Nav_ModB.parse_sentence.copy()`	-instinctive primitive `parse_sentence.copy()` maps the *instruction_sentence* instruction sentence onto Working Navigation Map B **WNMB'** in Navigation Module B
`Nav_ModB.parse_sentence.parse(`**WNMB'**$_t$`)`	-instinctive primitive `parse_sentence.parses()` parses through the instruction sentence, i.e., Navigation Map B **WNMB'** in Navigation Module B
`Nav_ModB.parse_sentence.parse.match()`	-for each word in the instruction sentence `Nav_ModB.parse_sentence.parse.match()` matches the word against the Causal Memory Module - if an action word is found (e.g., "place" in the example shown in Figure 7) then it is mapped to the Navigation Map A **WNM'** in Navigation Module A.
near_trigger	-word related to positional relationships of near and not near -triggers instinctive primitive `physics_near_object()`
near_trigger, ⇒`Nav_ModB.physics_near_object()`	-if a *near_trigger* occurs in parsing, then the instinctive primitive `Nav_ModB.physics_near_object()` is triggered -the `Nav_ModB.physics_near_object()` will mark "near" (or "not" if "not" is specified in the instruction) the cell, in Navigation Map A **WNM'** in Navigation Module A, which matches to this instinctive primitive, as well as all the adjoining non-null cells.
end_of_communication	-end of a language sentence, i.e., nothing left to parse
<*place*>	-existence of a <"place"> tag
end_of_communication, <*place*>, ⇒`Nav_ModA.place_object()`	-once the instinctive primitive `Nav_ModB.parse_sentence()` has reached the end of the instruction or communication sentence, i.e., there is nothing left to parse, then it will trigger associated tags -possible associated tags depend on the collection of instinctive primitives, which at present only has the single option of `place_object()` (but thousands of instinctive and learned primitives are possible here) -given a <*place*> tag in cell (0,0,0) of Navigation Module A in Figure 11 in the example in the text, the instinctive primitive `Nav_ModA.place_object()` is triggered - Note that this instinctive primitive will operate on Navigation Module A (i.e., Navigation Map A **WNM'**) - The `Nav_ModA.place_object()` instinctive primitive will attempt to move and place an object somewhere depending on what tags the other instinctive primitives `Nav_ModB.parse_sentence.parse.match()` and `Nav_ModB.physics_near_object()` have written to Navigation Map A **WNM'**
`Nav_ModA.move()`	-once `Nav_ModA.place_object()` has a location to move the object to, `Nav_ModA.move()` is triggered and actually sends an action signal to the Output Vector Association Module A which then outputs a more processed action signal to other modules resulting in the architecture's output actuators being instructed to move the object

Table A26. Explanation of Symbols and Pseudocode in Equations (107) – (114)

Input:	**WNM'**$_t$: the current Working Navigation Map A in Navigation Module A which is derived from the processed sensory inputs representing the sensory scene (e.g., Figure 7) **WNMB'**$_t$: the current Working Navigation Map B in Navigation Module B which is derived from a communication sentence (e.g., Figure 7) **WPR**$_t$: various Working Primitives **WPR**'s are the primitives that will be applied against the Working Navigation Map A **WNM'** in the Navigation Module A and/or Working Navigation Map B **WNMB'** in the Navigation Module B; these include: - `Nav_ModB.parse_sentence()` - `Nav_ModB.physics_near_object()` - `Nav_ModA.place_object()` - `Nav_ModA.move()`
Output:	*action*$_t$: a signal to move some actuator or send an electronic signal -it is sent from Navigation Module A to the Output Vector Association Module A (Figure 11) where more detailed instructions are created to effect the required actuator outputs to the Output Vector Shaping Module (Figure 11) to the actual real-world actuators
Description:	-A communication or instruction sentence is parsed in Navigation Module B and applied against a relevant represented sensory scene in Navigation Module A. As such, the beginnings of compositional language comprehension and behavior emerge.

Table A27. Summary of the Operation of Compositional Language Comprehension per Equations (107) – (114)