An Analogical Inductive Solution to the Grounding Problem

Howard Schneider[1]

[1]Sheppard Clinic North, Vaughan, ON, Canada
hschneidermd@alum.mit.edu

Abstract
The Causal Cognitive Architecture 5, a brain inspired cognitive architecture, is presented. The navigation map which holds features, procedures, and linkages in its spatially mapped cells, is the basic data structure of the architecture. By enhancing the feedback processing of intermediate results not only can causal abilities emerge but as well inductive analogical processing readily emerges as a core mechanism of the architecture. It is shown that abstract or poor features or concepts which cannot readily be grounded in sensorimotor interactions with the environment, can effectively be grounded via the core inductive analogical processing of the architecture, and provides a performance advantage in solving problems. As well, it is shown that relatively small changes in architecture can allow analogical reasoning and a solution to the grounding problem to emerge in a brain-inspired navigation map-based architecture.

Keywords: Grounding Problem; Brain-Inspired Cognitive Architecture (BICA); Cognitive Architecture; Artificial General Intelligence (AGI); Analogies; Causality; World Models; Hippocampus; Navigation Maps; Embodied Cognition

1. Introduction

1.1 The Grounding Problem

In cognitive architectures, often "cognition" is simply considered as one of many modules in the architecture. Barsalou (2020) points out that such a cognition-like module (or set of modules effectively making up a cognition-like module) is typically treated as an independent module, with other modules simply passing information into and out of the module. However, the grounding problem then raises the question of how the abstract symbols of such a cognition-like module can understand the external world, and thus how such an independent cognition-like module can truly produce cognition. Indeed, a "symbol" on its own simply represents a reference to something else. This is in contrast to a "concept" which involves some understanding and abstraction based on similar attributes that a set of objects, conditions, or events have in common (Guilford and colleagues, 1967).

Harnad (1990) gives a more obvious example of the symbol grounding problem. In the symbolic model of the mind, symbol strings which can be manipulated by rules, are considered to represent mental phenomena such as thoughts and beliefs. If a person with no understanding of the Chinese language tries to learn Chinese from a Chinese-Chinese dictionary, then the person would go from one string of symbols to another string of symbols without any meaning attached to the symbols. The symbols are "grounded" in other Chinese symbols, which provide no meaning to the person. Similarly, if an artificial intelligence system is not grounded in the environment around it, then it faces a grounding problem as to ascribing meaning to its internal symbols.

Davidsson (1993) notes that a possible solution to the binding problem would seem to be to describe the meaning of the symbols in a more powerful language than simply being the symbols the cognitive-like module is operating on. However, he notes that this potential solution would unfortunately lead to another set of symbols which then needs to be interpreted somehow to describe their meaning, and so on.

Harnad notes that the symbol grounding problem essentially is how symbols get their meaning, and thus related to what actually is meaning. When symbols are being operated on in a cognition-like module, the symbols are being manipulated following rules which are based on the symbols' shapes, so to speak, rather than their meaning. With regard to meaning, one way of thinking about this is how is a symbol connected to the things it refers to. Thus, Harnad

distinguishes between symbol grounding and symbol meaning. Harnad notes that if you have a robot which has a sensorimotor system sensing the external world and then processing these sensed signals in a Turing symbol system, the system would be grounded (since the internal symbols are connected to the external objects being sensed) but whether the symbols have meaning is not that clear. However, Harnad (1994) does write, "my guess is that the meanings of our symbols are grounded in the substrate of our robotic capacity to interact with the real world of objects, events and states of affairs that our symbols are systematically interpretable as being about."

Dijkstra and colleagues (2014) consider concepts as concrete or abstract. Concrete concepts refer to objects or actions present in the physical world, e.g., a dog, e.g., the act of smiling. Concrete concepts have physical or spatial constraints. However, abstract concepts refer to entities rather than objects, and these entities do not have the physical or spatial constraints that we see with concrete concepts. Examples given are the concept of democracy or abstract actions such as thinking. Thus, Dijkstra and colleagues note that while concrete concepts can be mapped to a sensory-motor domain in the brain, it is problematic to do so for an abstract concept such as democracy.

Harnad's solution to the grounding problem, i.e., grounding symbols with their real-world sensations, interactions, and linkages, would seem to be satisfactory for concrete concepts. However, how can we ground symbols representing abstract concepts? Reinboth and Farkaš (2022) consider whether abstract concepts can be represented within grounded cognition. They note that while concrete concepts tend to apply to physical objects and interactions, abstract concepts tend to refer to situations and events. They put forth the thesis that all cognition, including abstract concepts, are grounded in experience—via the body's sensorimotor interactions with the environment or via linguistic and social communication. They consider a graded utilization of these different grounding pathways, noting that abstract concepts may very well be grounded via the direct pathway of interactions with the external world.

While the grounding problem would initially seem to be more of philosophical issue, in fact researchers in the areas of building artificial intelligence-based robotics and language systems have noted the need for embodied cognition in order to produce better robotics (Brooks, 1991; Hoffmann, Pfeifer, 2018; Neisser, 1993; Pezzulo et al., 2013; Roy, Posner, Barfoot, Beaudoin, Bengio et al., 2021; Smith, Gasser, 2005) and the need for grounded language systems in order to produce better automated language systems (Bisk, Holtzman, Thomason, Andreas, Bengio et al., 2020; de Vries et al, 2018; Dubova, 2022; Guckelsberger et al., 2021; Kiela et al., 2016; Lake, Murphy, 2021; McClelland et al., 2020). In this paper we show how in the design of a mammalian brain-inspired cognitive architecture (BICA), the grounding problem emerges, and how the navigation map data structure used by the architecture and an analogical inductive approach similarly emerges as its solution.

1.2 Overview of this Paper

This paper follows this order:

1. Introduction to the Grounding Problem.
2. New work—the Causal Cognitive Architecture 5 (CCA5).
3. The emergence of the analogical problem-solving properties in the Causal Cognitive Architecture 5 (CCA5).
4. A solution to the grounding problem via the navigation map data structure and the analogical inductive properties of the Causal Cognitive Architecture 5 (CCA5).
5. Discussion.

As will be discussed in the sections below, the Causal Cognitive Architecture is inspired by the mammalian brain, in particular the mammalian hippocampus, and it uses navigation maps (i.e., maps whose principal purpose is to represent spatial locations) as its main data structure. Neuroscience research in the last few decades has shown the key role of navigation in the mammalian brain, particularly in the hippocampus (O'Keefe, Nadel, 1978; Samsonovich, Ascoli, 2005; Alme, Miao, Jezek, Treves, Moser, Moser, 2014; Moser, Rowland, Moser, 2105; Wernle, Waaga, Mørreaunet, Treves, Moser, Moser, 2018; Sugar, Moser, 2019; O'Keefe, Krupic, 2021). Given the evolutionary duplication of neural pathways which occurs (e.g., Chakraborty and Jarvis, 2015) and the long-evolutionary history of spatial cognition throughout the vertebrates (e.g., Rodríguez, et al., 2021), the postulation of the centrality of navigation map-like structures behind mammalian cognition (i.e., most of cognition, not just navigation) was made in the development of the Causal Cognitive Architecture.

Schneider (2021, 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).

Schneider (2022b) showed that with a number of algorithmic changes and additional operations, that analogical operations could be incorporated into the architecture's core decision making. The new architecture was called the Causal Cognitive Architecture 4 (CCA4). Schneider (2022b) notes the symbol grounding issue, but does not provide a full solution to the issue.

In this paper we present an improved architecture (Figure 1) and improved core mechanisms (the equations (1) to (106) below)) operating the architecture. This new architecture is called the Causal Cognitive Architecture 5 (CCA5). Below we show that inductive analogical operations can more readily emerge from the architecture. We also show that this architecture will now readily allow the emergence of a solution to the full grounding problem, i.e., both for concrete objects and abstract situations and concepts, as well as for any incompletely acquired information that is poorly grounded. We show that this full solution involving the inductive analogical feedback mechanism provides significantly improved problem-solving abilities for the architecture.

The following versions of the Causal Cognitive Architecture have been developed and presented, along with their key findings:

- Early exploratory versions – Provides a model for the evolutionary emergence of psychotic disorders in humans, noting that such disorders are much rarer in non-human mammals (Schneider, 2020).
- Causal Cognitive Architecture 1 (CCA1) – Shows how causal abilities can emerge in a navigation map-based cognitive architecture (Schneider, 2021).
- Causal Cognitive Architecture 2 (CCA2) – Shows the issues associated with the binding problem and provides a solution to the binding problem resulting in significant problem-solving performance improvements (described in Schneider, 2022a).
- Causal Cognitive Architecture 3 (CCA3) – Shows that there is also a temporal version of the binding problem and the issues associated with this problem. Provides a solution to the full binding problem (both spatial and temporal) resulting in significant problem-solving performance improvements (Schneider, 2022a).
- Causal Cognitive Architecture 4 (CCA4) – Shows that with a number of algorithmic changes and additional operations that analogical operations could emerge from the architecture, providing it with an important problem-solving mechanism (Schneider, 2022b).
- Causal Cognitive Architecture 5 (CCA5) – Presented in this paper. Shows that with some small changes (evolutionarily feasible ones involving feedback pathways and addition of circuits) to the architecture as well as improvements to the mechanisms operating the architecture (represented by equations (1) to (106) below) an inductive analogical feedback mechanism more readily emerges, and provides a full solution to the grounding problem, resulting in significant problem-solving performance improvements.

2. New Work: The Causal Cognitive Architecture 5 (CCA5)

2.1 Overview of the Architecture

Schneider (2021, 2022a, 2022b) describes versions of a mammalian brain inspired cognitive architecture called the Causal Cognitive Architecture (Figure 1 shows the current version 5 of the architecture presented in this paper, but is derived from previous versions). The "navigation map" (an example is shown in Figure 2) is the basic data element of the Causal Cognitive Architecture.

A navigation map holds spatial data, much as for example, a paper map motorists used in the pre-GPS era, showed streets and landmarks. However, the navigation maps in the Causal Cognitive Architecture also hold instructions (termed "primitives") for performing operations on other navigation maps as well as being a storage medium upon where the operations can take place.

In simulations of the Causal Cognitive Architecture (2022a), an array with 6x6x6 spatial dimensions (i.e., x,y,z axes) is arbitrarily used to simulate each such navigation map. Thousands to billions of such navigation maps can exist within the architecture. Figure 2 is an example of a navigation map showing the 6x6x0 spatial dimensions (i.e., z dimension not used) with spatial features of a section of a river. In each cell there can be additional information such

as operations (termed "primitives") to perform on the cell or cells in other navigation maps, or links to other cells or other navigation maps.

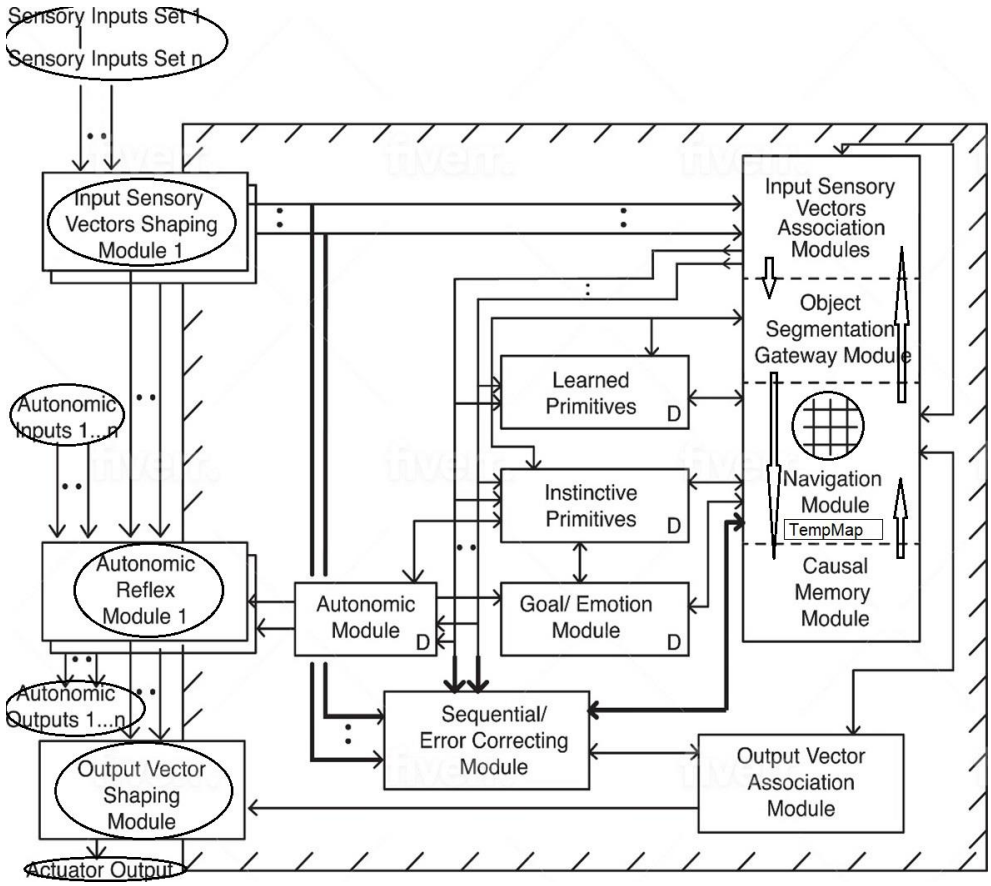

Figure 1. Causal Cognitive Architecture 5 (CCA5). The operation of the modules is explained in more detail in the text and the equations below. The letter "D" in some modules means that its properties change as the architecture develops, i.e., with experience and usage.

solid	solid	solid	water	solid	solid
solid	solid	solid	water	solid	solid
solid	solid	water	water	solid	solid
solid	solid	water	water	solid	solid
solid	solid	water, *link*{4574}	solid	solid	solid
solid, *iprimitive*{8974}	solid	water	solid	solid	solid

Figure 2. Example of a Navigation Map–the 6x6x0 spatial dimensions are shown. Sensory elements (in this case visual) are shown in each cell. As well some cells have links to other navigation maps and to instinctive primitives (i.e., instructions for operations).

The Causal Cognitive Architecture makes use of feedback pathways—states of a downstream module can influence the recognition and processing of more upstream sensory inputs. We can enhance the existing feedback processing of the intermediate results, i.e., the results of operations of the Navigation Module being fully but temporarily fed back and stored in the Input Sensory Vectors Association Modules (Figure 3-- #1 operation). Then in the next cognitive

cycle, these intermediate results of the Navigation Module are treated as the sensory inputs and propagated forward to the Navigation Module (Figure 3 -- #2 operation) where they can be operated on again. As a result, it was shown (Schneider 2021, 2022a) that causal behavior can emerge from a system previously showing only associative behavior or limited pre-causal behavior. (The more limited pre-causal behavior can arise alone from the navigation maps being used in the system).

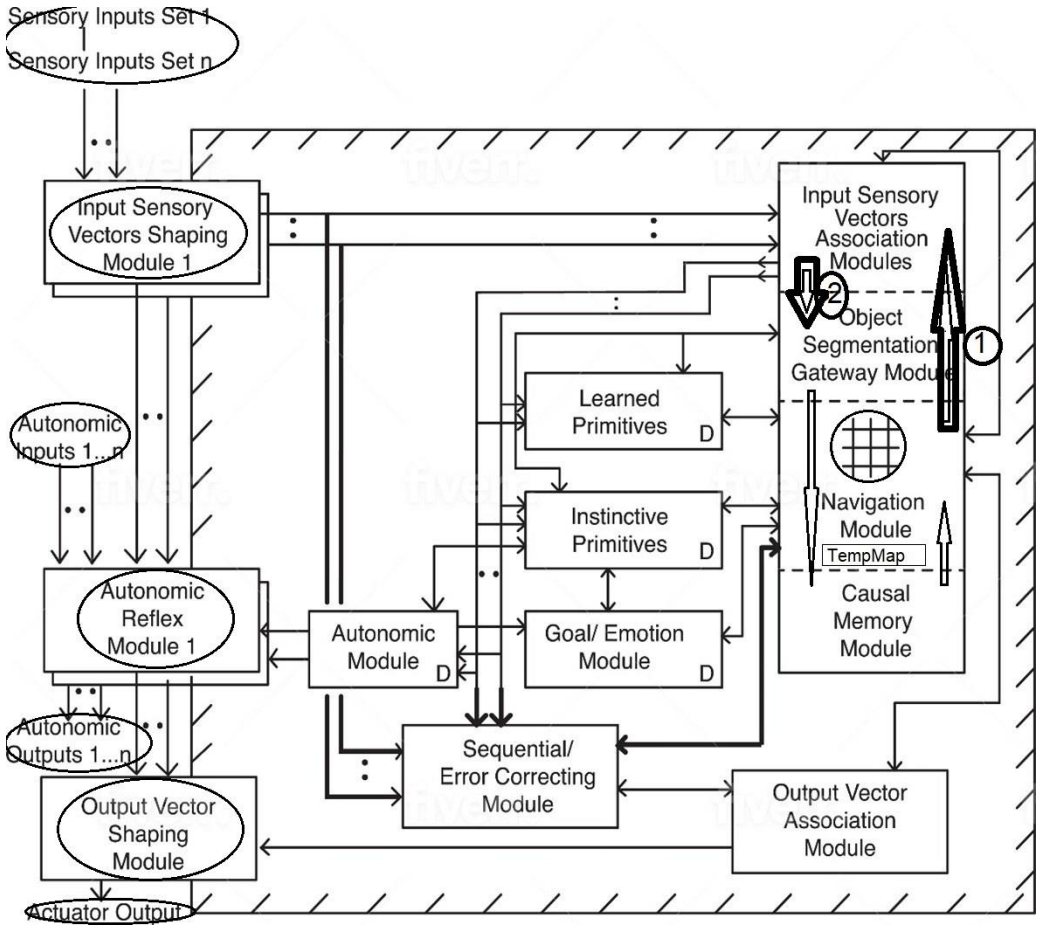

Figure 3. Causal Cognitive Architecture 5 (CCA5) – Results of operations of the Navigation Module can be temporarily fed back and stored in the Input Sensory Vectors Association Modules (#1 operation). In the next cognitive cycle, these intermediate results from the Navigation Module are treated as the sensory inputs for that cognitive cycle, and they are propagated forward towards the Navigation Module where they can be processed again (#2 operation).

Schneider (2022b) showed that with a number of algorithmic changes and additional operations, that analogical operations could be incorporated into the architecture's core decision making. The new architecture was called the Causal Cognitive Architecture 4 (CCA4). Schneider (2022b) briefly touched upon the symbol grounding issue, but did not provide a full solution. In this paper we present an improved architecture (Figure 1) and improved core mechanisms (the equations (1) to (106) below)) operating the architecture. This new architecture is called the Causal Cognitive Architecture 5 (CCA5). Below we show that inductive analogical operations can more readily emerge from the architecture. As a result, it is shown that a solution to the full grounding problem now readily emergences, i.e., both for concrete objects and abstract situations and concepts, as well as for any incompletely acquired information that is poorly grounded. Below in the paper we demonstrate that this full solution to the grounding problem involving the inductive analogical feedback mechanism provides an important improved problem-solving performance for the architecture.

The Causal Cognitive Architecture is a brain-inspired cognitive architecture (BICA). The Causal Cognitive Architecture is inspired by the mammalian brain, in particular the mammalian hippocampus, and it uses navigation maps (i.e., maps whose principal purpose is to represent spatial features) as its main data structure. These navigation maps are coopted for the data storage and representational needs of the architecture, as well as for the various small algorithms, termed "primitives" which can operate on information in a navigation map.

Sensory features stream in from different perceptual sensors. Objects detected in the streams of sensory features are segmented. Visual, auditory, and other sensory features of each segmented object 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. For example, in Figure 4 if there are sensory inputs of a river, then visual features, auditory features and olfactory features will be mapped onto what we term a "local" visual navigation map, a local auditory navigation map and a local olfactory navigation map. (Although the architecture is brain-inspired, for simplification of the equations below, we treat olfaction similarly to the other senses which relay through the thalamus to the neocortex.) These navigation maps are then matched against multisensory navigation maps already stored in the Causal Memory Module (Figure 4). The best matched navigation map from the Causal Memory Module is then considered moved to the Navigation Module (which can be virtually by way of activating the navigation map in place). Then as shown in Figure 5, this best matching navigation map (called in the example "Navigation Map A") will be updated with the actual sensory information sensed from the environment. The resulting map will be called the Working Navigation Map **WNM'**.

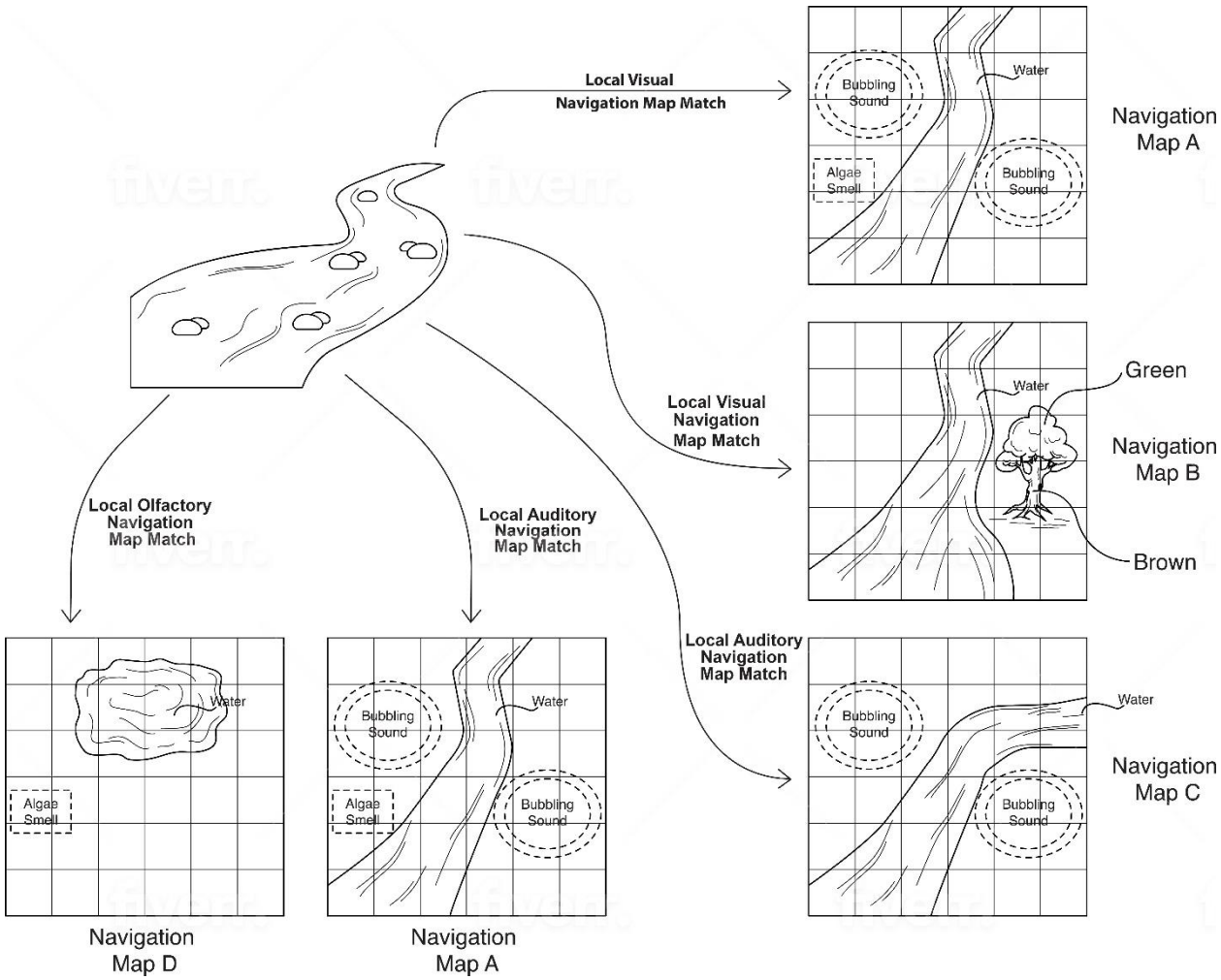

Figure 4. Matching Local Input Sensory Data Navigation Maps to a Previously Stored Multi-Sensory Navigation Map in the Causal Memory Module. The best matching navigation map from the Causal Memory Module will be Navigation Map A. (Please see text for details.)

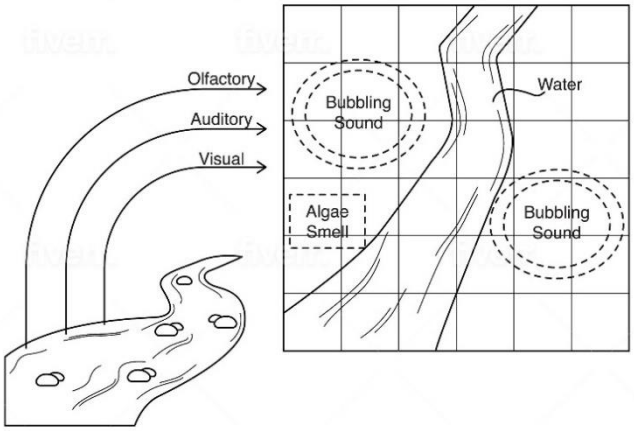

Figure 5. Navigation Map A from Figure 4. Features which are different in the sensory scene will be bound to this navigation map. Note that different sensory systems' features are bound onto a navigation map representing the object, in this case a river.

Instinctive and learned primitives, which act as small rules or productions, and which also are stored in modified navigation maps, then operate on the Working Navigation Map **WNM'**. This causes the Navigation Module to produce and send a signal to the Output Vector Association Module and then to the external embodiment (Figure 6).

A cognitive cycle is generally a cycle in a cognitive architecture during which the environment is perceived, processing of the sensory information occurs, and then there is some output action (Madl, 2011). In the CCA5 a cognitive cycle is similarly a cycle of sensory inputs into and through the architecture usually resulting in an output from the Navigation Module to the output modules (Figure 6). Then this repeats again—sensory inputs come into the Input Sensory Vectors Shaping Modules and propagate through the architecture, resulting in an output vector to the Output Vector Association Module which in turn results in an actual physical output. And then another cognitive cycle occurs, and so on.

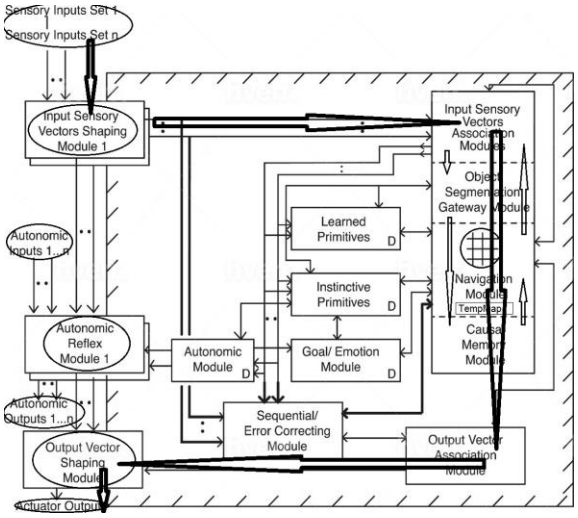

Figure 6. A "cognitive cycle" from sensory inputs to actuator output in the Causal Cognitive Architecture 5 (CCA5). It then repeats. As noted in the text, in some cognitive cycles, there is no output, but instead intermediate results of the Navigation Module are fed back and stored (Figure 3).

The Causal Cognitive Architecture 5 (CCA5) heavily makes use of feedback pathways—states of a downstream module can influence the recognition and processing of more upstream sensory inputs. As was noted above, in the CCA5 the feedback pathways between the Input Sensory Vectors Association Modules and the Navigation Module/ Object Segmentation Gateway Module are enhanced so that they allow intermediate results from the Navigation Module to be stored in the Input Sensory Vectors Association Modules (Figure 3).

In some cognitive cycles in the CCA5 there is no output signal. In certain states (for example, no actionable output from the Navigation Module) the information in the Navigation Module 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 (Figure 3) and operated on again. 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 (Schneider, 2022a).

Further experimentation with the architecture has revealed that the inductive analogical reasoning which emerges from the architecture, in combination with the navigation map structure, readily allows a solution to the grounding problem. As well, a solution to the grounding problem appears to yield a more powerful approach to solving problems the architecture encounters in operation, and which an embodiment of the architecture could encounter in day-to-day life. This is presented in this paper.

Also presented in this paper, is a new version of the architecture, the Causal Cognitive Architecture 5 (CCA5). There are many improvements to the architecture and the core algorithms of the architecture (reflected in the equations below). In particular, there are enhanced feedback pathways among the components of the Navigation Module complex (the solid box around the Navigation Module and other components in Figure 1). As well, there is a temporary memory storage register which should have been present in earlier versions of the architecture. Given the brain-inspired nature of the architecture, we are very conservative about introducing hardware or software components that would be taken as for granted otherwise in the design of a more conventional computer or neural network architecture. As well, many of the core algorithmic operating features of the architecture have been reworked to allow better operation with the newer architecture, including an effective solution to the grounding problem. A more detailed description of the Causal Cognitive Architecture 5 (CCA5) is presented below.

2.2 Formal Description of the Causal Cognitive Architecture 5 with Inductive Analogical Properties

To provide a compact yet understandable and relevant description of the CCA5, we will use a set of equations with pseudocode within the equations. A pseudocode is a conventional (e.g., English) language description of the steps a piece of software follows (Kwon, 2014; Olsen, 2005). For example, in equation (18) below, "`match_best_local_navmap()`" is pseudocode for an algorithm which matches navigation map $\mathbf{S'}_{\sigma,t}$ against all navigation maps ("local navigation maps") stored in the Input Sensory Vectors Association Module σ (Figure 1, with σ specifying the particular module) and returns the best matched navigation map.

In each section below describing the operation of the CCA5 with the equations below, we provide an overview of what main operations are happening. We then also explain every symbol in every equation, as well as all references to pseudocode.

Bold capitalized letters represent arrays. Using equation (18) below again as an example, **LNM** represents an array. (It represents a navigation map which are arrays.) Many of the values change with time which we represent with subscript *t*. For example, $\mathbf{S'}_{\sigma,t}$ (18) represents a particular array $\mathbf{S'}_{\sigma}$ whose value changes with time *t*. Vectors are in bolded italics (e.g., $\boldsymbol{s}$ in (9)).

The equations below first follow the forward flow of the input sensory information through the architecture (Figure 6). Then the backward flow of some of the feedback information is considered (Figure 3). Some of the details of the architecture which are not in the direct path of the described information flow are left out of these equations. As well, 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 would begin, and so on.

2.3 Normalizing Sensory Inputs: The Input Sensory Vectors Shaping Modules

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 we can address and model 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 $\boldsymbol{s}(t)$ is processed by the Input Sensory Vectors Shaping Modules (via pseudocode `Input_Sens_Shaping_Mods.normalize()`, described below in Table 1) into vector $\boldsymbol{s'}(t)$ (10). This transformation ensures that each element of $\boldsymbol{s'}$ will be compatible with the dimensions used by the navigation maps in many of the modules of the CCA5 (11).

As Table 2 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).

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

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

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

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

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

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

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

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

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

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

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

$\mathbf{S}_1 \in R^{m1 \text{ x } n1 \text{ x } o1}$	-we are 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 -as is noted below, for simplification we are 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
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
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 1. Explanation of Symbols and Pseudocode in Equations (1) – (11)

Input:	sensory inputs from sensory systems $1 \ldots \Theta\_\sigma$
Output:	***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 2. Summary of the Operations of the Input Sensory Vectors Shaping Modules per Equations (1) – (11)

2.4 Mapping of the Sensory Inputs: The Input Sensory Vectors Association Modules

From Figure 1 note that the normalized sensory input arrays $\mathbf{S'}_{1,t} \ldots \mathbf{S'}_{\Theta\_\sigma,t}$ 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. We want to match the visual processed inputs $\mathbf{S'}_{1,t}$ 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. We want to match auditory, olfactory and any other sensory inputs against the respective local area maps stored in a particular Input Sensory Vectors Association Module.

In (18) `Input_Assocn_Mod`σ`.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 we noted that the Causal Cognitive Architecture 5 (CCA5) 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 (Figure 1) and on which the Navigation Module can perform operations on) is used in the pseudocode method (18) in deciding which is a best match.

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

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

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

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

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

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

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

$\mathbf{LNM}_{(\sigma,mapno)}$	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 σ = 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 LNM_{Θ}) 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 1) and on which the Navigation Module can perform operations on
$\texttt{Input\_Assocn\_Mod}_{\sigma}\texttt{.match\_best\_local\_navmap}(\mathbf{S'}_{\sigma,t}, \boldsymbol{all\_maps}_{\sigma,t}, \mathbf{WNM'}_{t-1})$	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 1) 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 $\mathbf{WNM'}_{t-1}$ are considered in the determining the best match
Υ	represents a map number of a navigation map in an Input Sensory Vectors Association Module
$\mathbf{LNM}_{(\sigma,\Upsilon,t)}$	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 $\mathbf{LNM}_{(1,\Upsilon,t)}$ would be navigation map #3456 in the visual module

Table 3. Explanation of Symbols and Pseudocode in Equations (12) – (18)

At this point, in every sensory module in the Input Sensory Vectors Association Modules (Figure 1), there is a best matching $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$ ("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 1). The next operation is to update the best matching local navigation map $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$ with the actual sensory inputs $\mathbf{S'}_{\sigma,t}$ (21), creating an updated best matching navigation map $\mathbf{LNM'}_{(\sigma,\Upsilon,t)}$ which we rename (for simulation compatibility issues) to $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$ again.

If too many differences exist between the actual sensory inputs $\mathbf{S'}_{\sigma,t}$ and the best matching navigation map $\mathbf{LNM}_{(\sigma,\Upsilon,t)}$, then instead of updating the matched navigation map $\mathbf{LNM}_{(\sigma,\ \Upsilon\ ,t)}$, a new local navigation map $\mathbf{LNM}_{(\sigma,new\_map,t)}$ is created and updated with the actual sensory inputs $\mathbf{S'}_{\sigma,t}$ forming an updated best matching navigation map $\mathbf{LNM'}_{(\sigma,\ \Upsilon,t)}$ (22), which we rename (for simulation compatibility issues) to $\mathbf{LNM}_{(\sigma,\ \Upsilon,t)}$ 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 $\boldsymbol{lnm}_t$ represents the best-matching and updated local navigation maps $\mathbf{LNM}_{(\sigma,\ \Upsilon,\ t\ )}$ of all the different sensory modules of the Input Sensory Vectors Association Modules (23).

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

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

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

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

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

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, we will simply make $\mathbf{S'}_{\sigma,t}$ 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 that we will use when we make $\mathbf{S'}_{\sigma,t}$ into a new local navigation map
$\texttt{Input\_Assocn\_Mod}_{\sigma}\texttt{.differences}$	-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**	we update $\mathbf{LNM}_{(\sigma,\ \Upsilon,t)}$ with any new information in input sensory signal $\mathbf{S'}_{\sigma,t}$ (i.e., we copy $\mathbf{S'}_{\sigma,t}$ onto $\mathbf{LNM}_{(\sigma,\ \Upsilon,t)}$) thereby creating an updated $\mathbf{LNM'}_{(\sigma,\ \Upsilon,t)}$ (for simulation compatibility issues we rename $\mathbf{LNM'}_{(\sigma,\ \Upsilon,t)}$ to $\mathbf{LNM}_{(\sigma,\ \Upsilon,t)}$ again)
$\mathbf{LNM}_{(\sigma,new\_map,t)} \cup \mathbf{S'}_{\sigma,t}$ → **LNM'** → **LNM**	we 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 we rename $\mathbf{LNM'}_{(\sigma,\ \Upsilon,t)}$ 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 we rename $\mathbf{LNM'}_{(\sigma,\ \Upsilon,t)}$ to $\mathbf{LNM}_{(\sigma,\ \Upsilon,t)}$ again)

Table 4. Explanation of Symbols and Pseudocode in Equations (19) – (23)

Input:	$\boldsymbol{s'(t)}$ -- 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 we rename $\mathbf{LNM'}_{(\sigma,\ \Upsilon,t)}$ 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 5. Summary of the Operation of the Input Sensory Vectors Associations Modules per Equations (12) – (23)

2.5 Binding Space: Data Structures in the CCA5

In neuroscience it has been problematic explaining how the brain can process and bind different sensory pieces of data as well as sensory modalities (Herzog, 2008). For example, visual motion and visual color sensory inputs are processed in very different regions of the brain, and these regions have very modest interconnectivity, yet the brain can recognize and "bind" together these separately processed sensory inputs (Revonsuo, 1999). Feldman (2013) considers the binding problem as the following subproblems: general coordination of objects and activities, the subjective unity of perception, visual feature-binding, and variable binding such as the binding of words in a sentence that allow reasoning.

Schneider (2022a) considers in detail the binding problem—its philosophical aspects, its manifestation as an engineering problem in the Causal Cognitive Architecture 1, and provides a solution to the binding problem that greatly enhances the potential of the Causal Cognitive Architecture. Schneider (2022a) also distinguishes between spatial binding and temporal binding. We consider spatial binding here, which is similar to the classical binding problem. Without going into the details of Schneider (2022a), to a large extent the spatial binding problem is solved in the Causal Cognitive Architecture 2 (CCA2) (Schneider, 2022a) by the reorganization of the architecture such that the navigation map is the basic data structure, and there is a mapping of sensory data into local navigation maps (**LNM**s described above), and mapping of local navigation maps into multisensory navigation maps (Figures 4 and 5 above; described in more detail below).

The CCA5 fully incorporates the changes of the CCA2 and its data structures. We will pause for a moment in following the data through the architecture, and consider in this section the data structures the CCA5 uses.

Most of the CCA5 architecture's modules are compatible with navigation map as the fundamental 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 (which in Schneider, 2022a is actually three more dimensions) for non-spatial information such storing which features belong to which objects, for storing meta-data, and so on. In this section we will review a number of data structures used in the CCA5 architecture, most of them being compatible with the navigation map data structure.

Local navigation maps (i.e., stored locally in each sensory Input Sensory Vectors Association Module) were defined above in (14). We now in (24) similarly define multisensory navigation maps **NM**, instinctive primitive navigation maps **IPM**, and learned primitive navigation maps **LPM**. The multisensory navigation maps **NM** are stored in the Causal Memory Module (Figure 1) 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.

In (26) we define $\boldsymbol{all\_LNMs}_t$ 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) we define $\boldsymbol{all\_NMs}_t$ as holding all the multisensory navigation maps **NM** in the Causal Memory Module (Figure 1). In (28) we define $\boldsymbol{all\_IPMs}_t$ 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 1). In (29) we define $\boldsymbol{all\_LPMs}_t$ 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 1). And in (30) we define $\boldsymbol{all\_navmaps}_t$ 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 CCA5 architecture as there are a number of other specialized navigation maps, particularly in the Sequential/Error Correcting Module. Some of these are discussed below as well as in Schneider (2022a).)

In (31–33) we define an addressing scheme 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) we define a "*feature*" as some arbitrary real number representing a feature modality and value. For example, $feature_{3,(\chi\text{modcode=*Causal\_Memory\_Mod*, } mapno=3456,\ x=2,y=3,z=4)}$ 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) we similarly define a "*procedure*" as some arbitrary real number representing a procedure modality and value. For example, $procedure_{3,(\chi\text{modcode=*Causal\_Memory\_Mod*, } mapno=3456,\ x=2,y=3,z=4)}$ 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) we define a ***linkaddress*** as χ', i.e., pointing to χ' 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 we may write "*features"* which should be taken as the same variable as "*feature*".)

In (38) we define $\boldsymbol{cellfeatures}_{\chi,t}$ as all the features within a given cell χ. In (39) we define $\boldsymbol{cellprocedures}_{\chi,t}$ as all the procedures within a given cell χ. In (40) we define $\boldsymbol{linkaddresses}_{\chi,t}$ as all the linkaddresses within a given cell χ. In (41) we define $\boldsymbol{cellvalues}_{\chi}$ as all the values—the features, the procedures, the link addresses—held by a cell of a navigation map at location χ. 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) we show 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. In (42) we show that the pseudocode `link`(χ,*t*) will return all the links that a cell at address χ contains.

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

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

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

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

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

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

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

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

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

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

$$feature \in R \quad (34)$$

$$procedure \in R \quad (35)$$

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

$\Phi\_feature$ = 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 1) 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 1) 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 1) -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 1) -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 in the text we 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'
$\boldsymbol{cellfeatures_{\chi}}$	vector of all the features within a cell χ
$\boldsymbol{cellprocedures_{\chi}}$	vector of all the procedures within a cell χ
$\boldsymbol{linkaddresses_{\chi}}$	vector of all the linkaddress within a cell χ
$\boldsymbol{cellvalues_{\chi}}$	vector of all of the features, procedures and linkaddresses within a cell χ
$\boldsymbol{all\_navmaps_{\chi}}$	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***) = $\boldsymbol{cellvalues_{\chi}}$
`link`(χ,*t*)	pseudocode that returns all the linkaddresses within a cell χ , i.e., same value as $\boldsymbol{linkaddresses_{\chi}}$

Table 6. 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 7. Summary of the Operation of Equations (24) – (43)

2.6 Binding Time: The Sequential/Error Correcting Module

As discussed above, the improvements from the CCA1 to the CCA2, are reflected in its data structures which are also present in the CCA5, and were discussed in the above section. These improvements largely solved the binding problem. The architecture was now able to integrate different phenomena and objects within one sensory system as well as integrate multiple sensory inputs of the same objects. As Schneider (2022a) discussed, it was now able to handle more complex sensory inputs and to display causal behavior in more complex environments.

However, in the real world, there are changes with time. Objects change position with time. Higher-level concepts which can be represented on a navigation map also change with time. Human sensory systems tend to operate as a function of time. Vision is not static—our eyes make rapid saccadic movements to send signals as a function of time. Touch is not static—tactile changes with respect to time are the signals sent to our brain. Hearing is similarly the detection of changes in sounds with respect to time. If the CCA2 senses and binds sensory inputs at 30 inputs per second, then hundreds and hundreds of different navigation maps are produced which must be processed and stored and then used for comparisons in the future. Operations on hundreds of navigation maps associated with a scene in the environment as opposed to a single navigation map, becomes exponentially more difficult to work with.

Changes in the CCA2 architecture resulting in the CCA3 version were made to allow not only spatial binding but temporal binding of data (Schneider, 2022a). In the CCA3 there is preprocessing the change and rate of change of objects, and binding of these values as a physical feature onto the navigation map. Thus, hundreds or thousands or even tens of thousands of navigation maps get reduced to a single navigation map. This greatly simplifies the processing of the navigation maps to allow precausal as well as causal behavior. These changes remain incorporated in the CCA5.

Figure 5 above is a navigation map of sensory scene of a river, showing how different features are bound onto this navigation map. Now imagine this sensory scene over ten seconds with a leaf flowing down the river. If the architecture is sampling sensory inputs 30 times per second, then 300 navigation maps are created. However, the CCA3 will bind time, i.e., will bind various motions. Figure 7 shows the same sensory scene over ten seconds. Only a single navigation map is required—a motion prediction vector represents the motion of the leaf in the river.

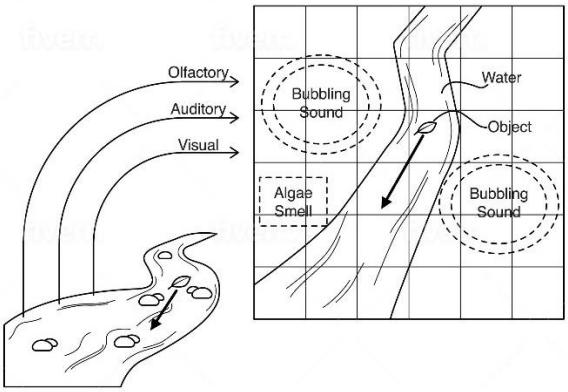

Figure 7. A moving object such as a leaf is moving down the river. Rather than requiring hundreds of different navigation maps as a function of time, we use a single navigation map with a motion prediction vector.

We will now follow sensory inputs through the CCA5 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 $\mathbf{S'}_{\sigma,t}$ represented by ***s'(t)*** (from the output of the Input Sensory Vectors Shaping Modules (10, 11)) feed into the Sequential/Error Correcting Module. The pathways can be seen in Figure 8. After processing these signals the Sequential/Error Correcting Module sends a "motion prediction vector" to the Navigation Module. As described above and as illustrated in Figure 7, 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 ***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 we only do so for the visual and auditory sensory systems.

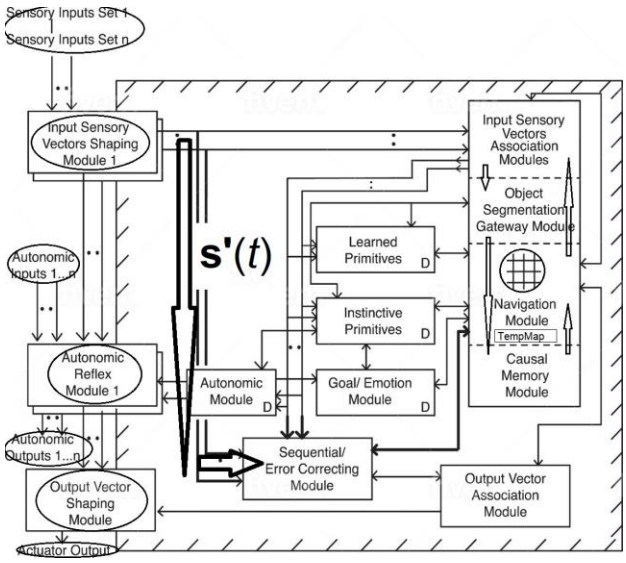

Figure 8. Normalized sensory inputs are also fed into the Sequential/Error Correcting Module.

In (47) the pseudocode `Sequential_Mod.visual_match()` matches ***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 ***visual_motion***$_t$ is then computed from visual time series data (47). A similar process occurs in computing the motion prediction vector ***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 **VNM''**$_t$ (50a, 50b). **VNM''**$_t$ is then propagated to the Navigation Module complex (Figure 1).

In (51) the pseudocode `Sequential_Mod.auditory_match_process()` extracts sound patterns from ***auditory_series***$_t$, and stores these patterns spatially in a navigation map **AVNM**$_t$. Then **AVNM**$_t$ is propagated to the Navigation Module complex (Figure 1).

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

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

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

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

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

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

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

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

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

In the next section we will see that the Object Segmentation Gateway Module (Figure 1) 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. (In the current software implementation of the CCA5 the visual inputs are the only ones currently segmented, i.e., the visual input local navigation map will be segmented into objects.) 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. $\mathbf{VSNM}_{i,t\text{-}3}$, $\mathbf{VSNM}_{i,t\text{-}2}$, $\mathbf{VSNM}_{i,t\text{-}1}$, 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 $\boldsymbol{visual\_segmented\_series}_{i,t}$ and produces $\boldsymbol{visseg\_motion}_{i,t}$ which is a motion prediction vector for the motion information, if any, in $\boldsymbol{visual\_segmented\_series}_{i,t}$. This motion prediction vector is copied, like any spatial feature, onto the original navigation map $\mathbf{VSNM}_{i,t}$ and the updated Visual Segmented Navigation Map $\mathbf{VSNM'}_{i,t}$ is then sent back to the Object Segmentation Gateway Module/Navigation Module (55).

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

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

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

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

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 1) -sent to both Input Sensory Vectors Association Modules and to the Sequential/Error Correcting Module (Figure 1)
s'(t-1)	***s'(t)*** value in the previous cognitive cycle, i.e., t-1
$\boldsymbol{s'\_series}_t$	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***$_t$) → $\boldsymbol{visual\_series}_t$	-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 $\boldsymbol{visual\_series}_t$ 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***$_t$) → $\boldsymbol{auditory\_series}_t$	-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 $\boldsymbol{auditory\_series}_t$ 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`($\boldsymbol{visual\_series}_t$) → $\boldsymbol{visual\_motion}_t$	-pseudocode for an algorithm that matches $\boldsymbol{visual\_series}_t$ with visual time series stored in the Sequential/Error Correcting Module (Figure 1) -a best match is chosen (or if no matches close enough then $\boldsymbol{visual\_series}_t$ used itself as the best match -the best match is then updated from $\boldsymbol{visual\_series}_t$ (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 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 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 algori$_t$hm 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")
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 1) 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 $\mathbf{VSNM}_{1,t}$ for river, $\mathbf{VSNM}_{2,t}$ for rock and $\mathbf{VSNM}_{3,t}$ 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 $\mathbf{VSNM}_{i,t}$ and $\mathbf{VSNM}_{i}$ from one cognitive cycle ago (i.e., t-1), two cycles ago (i.e., t-2) and three cycles ago (i.e., t-3) = [$\mathbf{VSNM}_{i,t-3}$, $\mathbf{VSNM}_{i,t-2}$,$\mathbf{VSNM}_{i,t-1}$, and $\mathbf{VSNM}_{i,t}$] -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 1) -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***$_{i,t}$
$\mathbf{VSNM}_{i,t} \cup$ ***visseg_motion***$_{i,t}$ → $\mathbf{VSNM'}_{i,t}$	-$\mathbf{VSNM}_{i,t}$ was previously received from the Object Segmentation Gateway Module (Figure 1) 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 [$\mathbf{VSNM}_{i,t-3}$, $\mathbf{VSNM}_{i,t-2}$,$\mathbf{VSNM}_{i,t-1}$, and $\mathbf{VSNM}_{i,t}$] 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 $\mathbf{VSNM}_{t}$ navigation map is copied to the $\mathbf{VSNM}_{i,t}$ navigation map–binding of temporal features as a spatial feature occurs here for this segmented object -$\mathbf{VSNM'}_{i,t}$ is produced in this operation (i.e., the original $\mathbf{VSNM}_{i,t}$ plus the motion prediction vector (***visseg_motion***$_{i,t}$) and is sent back to Object Segmentation Gateway Module/Navigation Module (Figure 1) -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 $\mathbf{VSNM'}_{i,t}$ produced and returned to the Object Segmentation Gateway Module/Navigation Module (Figure 1), with *i*=1, *i*=2, and *i*=3

Table 8. Explanation of Symbols and Pseudocode in Equations (44) – (55)

Input:	- ***s'(t)*** is a vector representing all the normalized sensory input arrays $\mathbf{S'}_{1,t}$ … $\mathbf{S'}_{\Theta\_\sigma,t}$ (**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 complex (Figure 1) -**AVNM**$_t$ is a navigation map binding advanced auditory patterns propagated to the Navigation Module complex (Figure 1) (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 (Figure 1)
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 1). -This module allows a solution to the temporal binding problem discussed above. -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 9. Summary of the Operation of the Sequential/Error Correcting Module per Equations (44) – (55)

2.7 Segmenting Objects: The Object Segmentation Gateway Module

As illustrated in Figure 9, 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. We sometimes refer to the complex of three tightly connected modules—the Object Segmentation Gateway Module, the Navigation Module and the Causal Memory Module—as the "Navigation Complex." These interconnected modules 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.

As mentioned in the previous section, the Object Segmentation Gateway Module will attempt to segment each sensory scene into different objects. In the current version of the CCA5 we do so only visually—we attempt to recognize coherent visual shapes in the sensory scene. (We could do so with multiple senses. For example, perhaps in the future the auditory and an additional radar sense, for example, could also be used to recognize coherent objects.)

Figure 10 shows the same sensory scene as Figure 7—a river, some rocks, a leaf floating in the river. We ignore the auditory and olfactory sensations for the moment and segment visually. 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. Note that the use of labels such as 'river', 'rocks', 'leaf' is for the benefit of the reader. The CCA5 does not have a full English language implemented at this time, and uses its own internal labels.

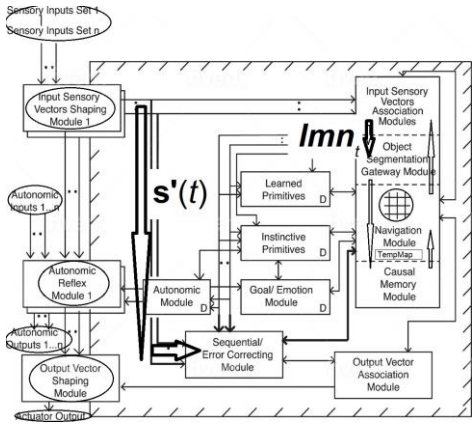

Figure 9. At this point the sensory inputs have been transformed into best matching and updated visual, auditory and olfactory local navigation maps (represented by the vector $\boldsymbol{lnm}_t$). The local navigation maps **LNM**s are propagated to the Object Segmentation Gateway Module.

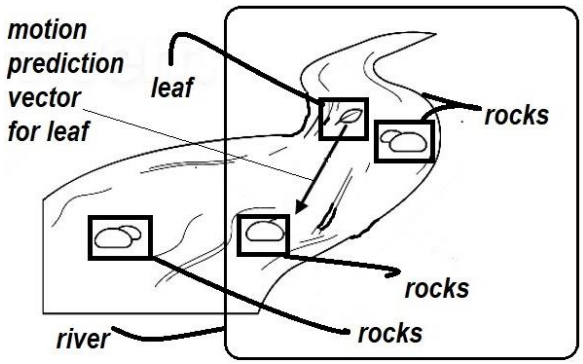

Figure 10. The sensory scene of Figure 7—a river, some rocks, a leaf floating in the river. We ignore the auditory and olfactory sensations for the moment and segment visually. The leaf, the rocks and the river are recognized by the Object Segmentation Gateway Module and segmented as separate objects 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. Note: the use of labels such as 'river', 'rocks', 'leaf' is for the benefit of the reader. The CCA5 does not have full language implemented at this point and uses its own internal labels.

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 in Figure 10 get put onto a separate $\mathbf{VSNM}_i$ navigation map. For example, in Figure 10, $\mathbf{VSNM}_{1,t}$ 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 $\mathbf{LNM}_{(1,\ \gamma\ ,t)}$ 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. We have all the different sensory systems' best-matching **LNM**s via vector ***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) we see it is 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 $\mathbf{VNM''}_t$.

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

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

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

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

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

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

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

$\mathbf{LNM}_{(\sigma, \Upsilon, t)}$ 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: we update $\mathbf{LNM}_{(\sigma, \Upsilon, t)}$ with any new information in input sensory signal $\mathbf{S'}_{\sigma, t}$ (i.e., we 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 we 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 $\boldsymbol{lnm}_t$ we see that 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 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
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 **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
VNM''$_t$ Equations (50a, 50b) described previously	as described previously in the Sequential/Error Correcting Module: -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 -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 (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 (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 10. 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 11. Summary of the Operation of the Object Segmentation Gateway Module per Equations (56) – (62)

2.8 Matching and Storing Multi-Sensory Navigation Maps: The Causal Memory Module

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

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

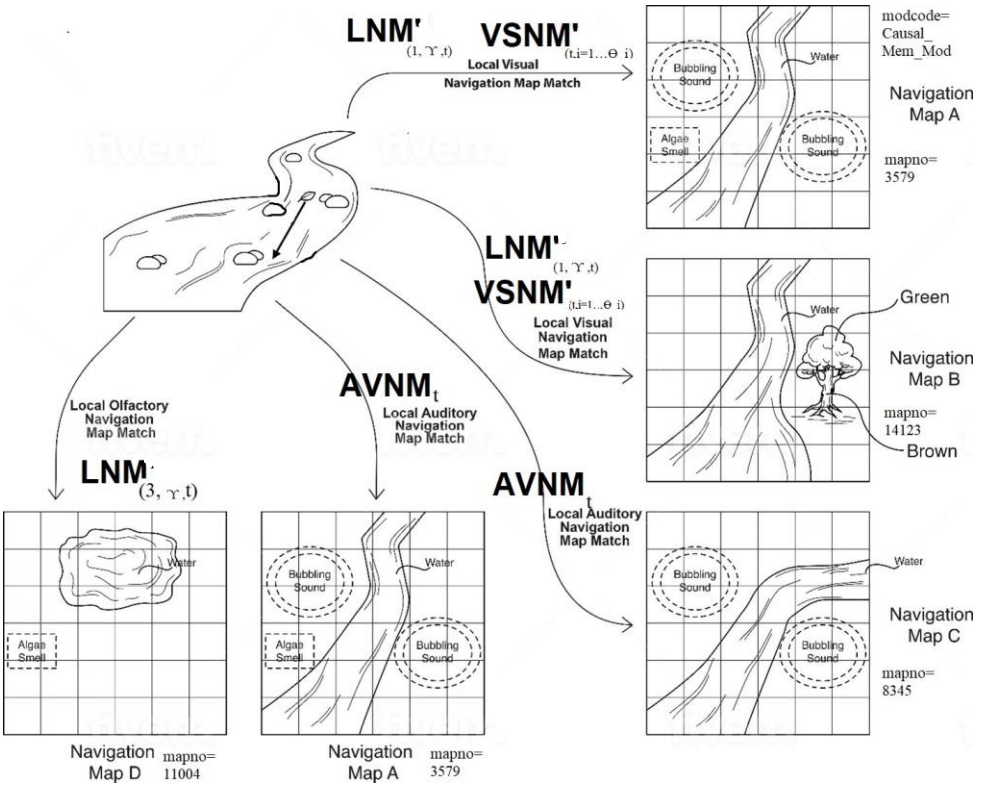

Figure 11. Matching single sensory Local Navigation Maps **LNM**s to a Previously Stored Multi-Sensory Navigation Map in the Causal Memory Module (Figure 4 with same nomenclature used by equations). As seen in equation (63) **VSNM'**$_{(t,i=1...\Theta\_i)}$, **LNM'**$_{(1,\Upsilon,t)}$, **AVNM**$_t$, **LNM**$_{(3,\Upsilon,t)}$ are matched against stored multisensory navigation maps in the Causal Memory Module. The navigation maps in the Causal Memory Module have full addresses. As an example, "Navigation Map A" really would have the address such as modcode=Causal_Mem_Mod, *mapno*=3579. In this example, the best matching navigation map from the Causal Memory Module is Navigation Map A (i.e., Causal_Mem_Mod, 3579). Thus, as per equation (63) the Working Navigation Map **WNM** is set to Navigation Map A (i.e., Causal_Mem_Mod, 3579).

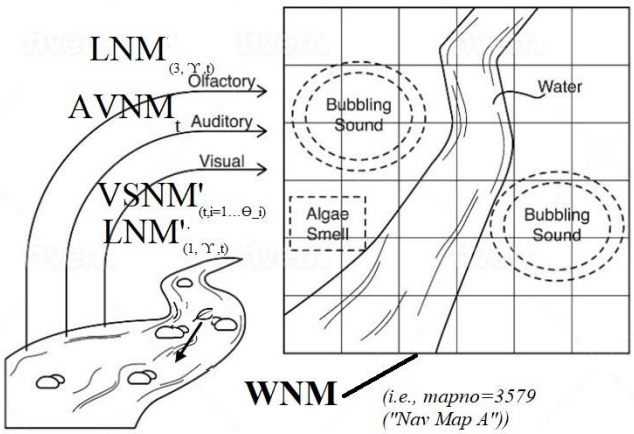

Figure 12. The best matching navigation map from the Causal Memory Module, Navigation Map A (i.e., Causal_Mem_Mod, 3579), is now updated with the actual sensory inputs, i.e., ***actual***$_t$ (**VSNM'**$_{(t,i=1...\Theta\_i)}$, **LNM'**$_{(1,\,\Upsilon,t)}$, **AVNM**$_t$, and **LNM**$_{(3,\,\Upsilon,t)}$), as shown in equation (67). This results in an updated Working Navigation Map **WNM'**.

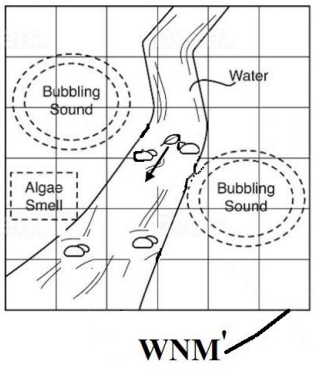

Figure 13. The updated Working Navigation Map **WNM'** produced in Figure 12, according to equation (67).

These single sensory system navigation maps will now be matched against the previously stored multi-sensory navigation maps stored in the Causal Memory Module. An example is shown in Figure 11 where the best-matching visual local navigation map **LNM'**$_{(1,\,\Upsilon,t)}$, the visual segmented navigation maps **VSNM'**, 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 (63).

(Note that the navigation maps in the Causal Memory Module have full addresses but we label them for simplicity as "Navigation Map A, B, C…" in Figure 11. However, the full address χ of Navigation Map A in Figure 11 would be, as an example, [[modcode=Causal_Mem_Mod, *mapno*=3579], x,y,z] where x,y,z are coordinates of any cell in this navigation map (31–33).)

In Figure 11, the visual local navigation map **LNM'**$_{(1,\,\Upsilon,t)}$ and the visual segmented navigation maps **VSNM'** match best to previously stored Navigation Maps A and B. The navigation map binding auditory patterns **AVNM**$_t$ matches best to previously stored Navigation Maps A and C. The olfactory local navigation map **LNM**$_{(3,\,\Upsilon,t)}$ matches best to previously stored Navigation Map D.

In this case 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).

We then must update **WNM** 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 ***actual***$_t$ (64) and an arbitrary threshold h' (65), then rather than update **WNM** which we just assigned to the best matching map (63), we will create a new Working Navigation Map **WNM'** which we consider updated from the actual sensory inputs ***actual***$_t$. We do

this by 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 (Figure 1). 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, we call the pseudocode `Nav_Mod.TempMap.erase()` (68) 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). This is illustrated in Figures 12 and 13.

In (70) we store **WNM'** in the Causal Memory Module—in future operation 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 $\mathbf{WNM'}_t$.

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

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

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

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

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

$$\texttt{Nav\_Mod.TempMap.erase()} \quad (68)$$

$$|\texttt{Object\_Seg\_Mod.differences}(\boldsymbol{actual}_t, \mathbf{WNM}_t)| > h',$$
$$\Rightarrow \mathbf{WNM'}_t = \mathbf{TempMap} \cup \boldsymbol{actual}_t \quad (69)$$

$$\texttt{Causal\_Mem\_Mod.store\_WNM\_update\_links}(\mathbf{WNM'}_t) \quad (70)$$

`Object_Seg_Mod.match_best_map(` **VSNM'**$_{(t,i=1\ldots\Theta\_i)}$, **LNM'**$_{(1,\Upsilon,t)}$, **AVNM**$_t$, **LNM**$_{(3,\Upsilon,t)}$, **LNM**$_{(4\ldots n\_\sigma,\Upsilon,t)}$ **)** → **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'**$_{(t,i=1\ldots\Theta\_i)}$, 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'**$_{(t,i=1\ldots\Theta\_i)}$, **LNM'**$_{(1,\Upsilon,t)}$, **AVNM**$_t$, **LNM**$_{(3,\Upsilon,t)}$, **LNM**$_{(4\ldots n\_\sigma,\Upsilon,t)}$) are described individually below -the best matching multisensory map is designated as the Working Navigation Map **WNM**
argument used in `match_best_map`: **VSNM'**$_{t,i=1\ldots\Theta\_i}$ **VSNM**$_{t,i=1\ldots\Theta\_i}$ → **VSNM'**$_{t,i=1\ldots\Theta\_i}$	-a "Visual Segmentation Navigation Map" **VSNM** is an ordinary navigation map with visually segmented information stored on it -the Object Segmentation Gateway Module (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)

(Note: **VSNM** is first created in the Object Segmentation Gateway Module, and then updated to **VSNM'** in the Sequential/Error Correcting Module)	-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 σ = 3 then it would be olfactory sensory inputs)
argument used in `match_best_map`: **LNM**$_{(4…n\_\sigma, \Upsilon, t)}$	-in the example in the text we only use three sensory systems: visual, auditory and olfactory -however, 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**$_{(4…n\_\sigma, \Upsilon, t)}$]
TempMap	-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 `Nav_Mod.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 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 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_Mod.TempMap.erase()`	-pseudocode specifying that **TempMap** is erased -thus, it will act as an empty navigation map the next time it is required

Table 12. 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 (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 focuses its attention on, i.e., applies operations on to make a decision to take some sort or no action.

Table 13. Summary of the Operation of the Causal Memory Module per Equations (63) – (70)

2.9 Making Decisions and Taking Actions: The Navigation Module

In the Navigation Module 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 and to the external embodiment, completing the cognitive cycle (Figure 6).

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

From Figure 1 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 5. However, we will not spend much time discussing their operation in order to focus on the Navigation Module 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 learned or instinctive primitive navigation map is compared against the Working Navigation Map **WNM'**. By simple array operations and logical operations, the Navigation Module 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 and then to the external embodiment (Figures 1, 6).

Instinctive primitives **IPM'**s are stored in the Instinctive Primitives Module, and come preprogrammed with the architecture. As can be seen in Figure 1, the "D" in the module indicates that the module's properties change as the system develops, i.e., is used more. With time some instinctive primitives will be less likely to become activated, while others more likely. Learned primitives **LPM**'s are stored in the Learned Primitives Module, and are learned by the architecture (Figure 1). We will not spend much time discussing learning operations in this paper in order to focus on the Navigation Module in this section, and to focus on the grounding problem later in the paper. 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. 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.

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).

As shown in Figure 14, at this point we have in the Navigation Module a Working Primitive **WPR**$_t$ and a Working Navigation Map **WNM'**. By simple array operations and logical operations, the Navigation Module compares and operates on the Working Primitive **WPR**$_t$ and the Working Navigation Map **WNM'**. Equation (82) shows this is indicated by pseudocode `Nav_Mod.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 and then to the external embodiment (Figure 14).

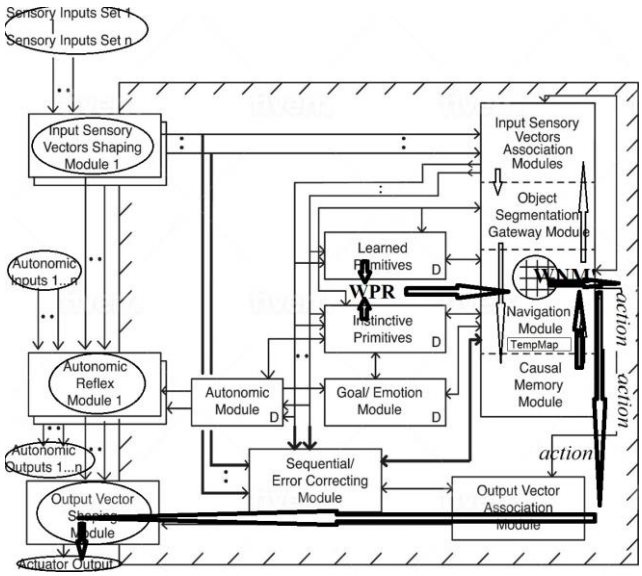

Figure 14. The Working Primitive Map **WPR** is applied on the Working Navigation Map **WNM'** in the Navigation Module to produce an *action* signal (82). The *action* signal is sent to the Output Vector Association Module and causes the embodiment of the architecture to move some actuator or send an electronic signal. As mentioned in the text, in other cases no *action* signal may result but instead other operations such as feeding back a signal may occur, which is described in the following section.

The *action* signal from the Navigation Module may be something like, for example, **<move right>**. This signal is processed by the Output Vector Association Module (Figure 14). The pseudocode `Output_Vector_Mod.action_to_output(` $action_t$, $\mathbf{WNM'}_t$) (84) results in $\boldsymbol{output\_vector}_t$ which contains more detailed instructions for the actuators to cause the embodiment to move right. However, $\boldsymbol{output\_vector}_t$ is first sent to the Sequential/Error Correcting Module where it will be corrected for motion and timing issues, resulting in a $\boldsymbol{motion\_correction}_t$ vector (86). The pseudocode `Output_Vector_Mod.apply_motion_correction(` $\boldsymbol{output\_vector}_t$, $\boldsymbol{motion\_correction}_t$) (87) produces the final signal $\boldsymbol{output\_vector'}_t$ which is sent to the Output Vector Shaping Module (Figure 14). 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.

$$autonomic \in \mathrm{R} \quad (71)$$

$$emotion \in \mathrm{R} \quad (72)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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`($autonomic_t$, $\mathbf{WNM'}_{t-1}$) → [$emotion_t$, $\mathbf{GOAL}_t$]	-pseudocode that calculates the values of the *emotion* and **GOAL** variables -`Goal/Emotion_Mod` refers to the Goal/Emotion Module (Figure 1) -the argument $\mathbf{WNM'}_{t-1}$ is the Working Navigation Map from the previous (i.e., t-1) cognitive cycle
`Instinctive_Primitives_Mod.match_best_primitive` ($\boldsymbol{actual}_t$, $emotion_t$, $\mathbf{GOAL}_t$) → $\mathbf{WIP}_t$	-pseudocode that returns the most appropriate instinctive primitive navigation map **IPM** from the Instinctive Primitives Module (Figure 14) given the arguments of the processed sensory inputs $\boldsymbol{actual}_t$, the current $emotion_t$ value, and the current $\mathbf{GOAL}_t$ navigation map -`Instinctive_Primitives_Mod` refers to the Instinctive Primitives Module (Figure 14) -the most appropriate instinctive primitive **IPM** returned is considered to be the Working Instinctive Primitive $\mathbf{WIP}_t$
`Learned_Primitives_Mod.match_best_primitive` ($\boldsymbol{actual}_t$, $emotion_t$, $\mathbf{GOAL}_t$) → $\mathbf{WLP}_t$	-pseudocode that returns the most appropriate learned primitive navigation map **LPM** from the Instinctive Primitives Module (Figure 14) given the arguments of the processed sensory inputs $\boldsymbol{actual}_t$, the current $emotion_t$ value, and the current $\mathbf{GOAL}_t$ navigation map -`Learned_Primitives_Mod` refers to the Learned Primitives Module (Figure 14) -the most appropriate instinctive primitive **LPM** returned is considered to be the Working Learned Primitive $\mathbf{WLP}_t$
$\mathbf{WLP}_t = [\ ],\ \Rightarrow \mathbf{WPR}_t = \mathbf{WIP}_t$ $\mathbf{WLP}_t \neq [\ ],\ \Rightarrow \mathbf{WPR}_t = \mathbf{WLP}_t$ → **WPR**	-the Working Primitive **WPR** is the primitive that will be applied against the Working Navigation Map **WNM'** in the Navigation Module -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_Mod.apply_primitive`(**WPR**$_t$, **WNM'**$_t$) → *action*$_t$	-`Nav_Mod` refers to the Navigation Module (Figure 14) -`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 (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 (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 14. 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 **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 (Figure 14) where more detailed instructions are created to effect the required actuator outputs
Description:	-The Navigation Module 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 15. Summary of the Operation of the Navigation Module per Equations (71) – (87)

2.10 Postponing Actions: Feedback Signals and Intermediate Results

As was discussed in the introductory sections, the Causal Cognitive Architecture, including the CCA5, 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. As was noted in the introductory sections, in previous versions of the architecture and especially so in the CCA5, the feedback pathways between the Input Sensory Vectors Association Modules and the Navigation Module/ Object Segmentation Gateway Module are enhanced so that they can allow not just a feedback signal, but the full intermediate results from the Navigation Module to be stored in the Input Sensory Vectors Association Modules (Figure 15).

In some cognitive cycles in the CCA5 there is no output signal. In certain states (for example, no actionable output from the Navigation Module) the information in the Navigation Module can be fed back and stored in the Input Sensory Vectors Association Modules (Figure 15). 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 (Figure 16) and operated on again. 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 (Schneider, 2022a).

The observant reader may wonder given the existence of the "TempMap" temporary storage area in the Navigation Module (Figure 1), why for causal behavior it is necessary to feed back the intermediate results of the Navigation Module to the Input Sensory Vectors Association Modules, rather than just store them in **TempMap** (66). The reason is that the Causal Cognitive Architecture is biologically inspired, and from an evolutionary perspective, it seems more reasonable that by enhancing feedback pathways the Navigation Module's intermediate results could be stored in the Input Sensory Vectors Association Modules and reprocessed in the next cognitive cycle, with few other evolutionary changes required. In fact, the TempMap region was not present in the CCA4 or earlier versions of the architecture, but first appears in this paper with the CCA5 architecture. In order to efficiently carry out analogical operations, as described in the following sections, the evolutionary usurpation of some brain region as a temporary storage region would have been very advantageous.

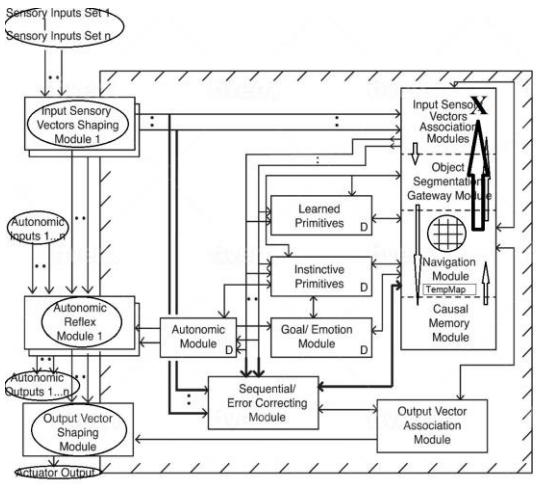

Figure 15. Instead of producing an *action* signal, the Navigation Module feeds back its intermediate results to the Input Sensory Vectors Association Modules using the enhanced feedback pathways between the modules.

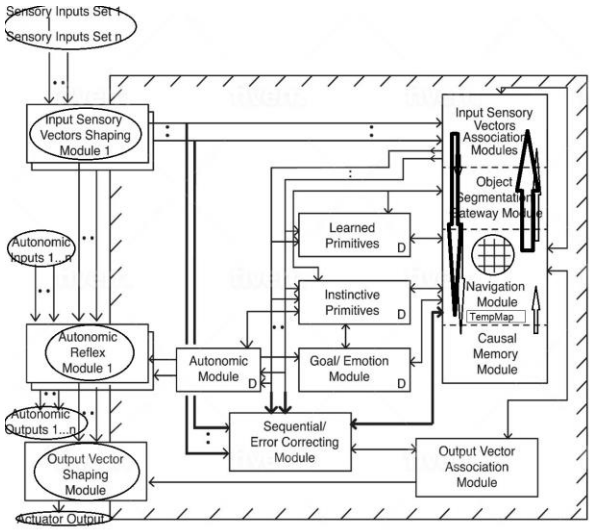

Figure 16. In the next cognitive cycle, the intermediate results which have been stored in the Input Sensory Vectors Association Modules are treated as the sensory input and are treated like input sensory information and automatically propagated to the Object Segmentation Gateway Module and eventually the Navigation Module. (The actual sensory inputs from the environment are not processed.) 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_Mod.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 and to the Output Vector Shaping Module 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 and the Input Sensory Vectors Association Modules, the Causal Cognitive Architecture can in these cases feedback the intermediate results

of the Navigation Module and store them in the Input Sensory Vectors Association Modules (88), as illustrated in Figure 15. 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_Mod.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 in the last cognitive cycle) are propagated towards the Navigation Module and operated on again, perhaps by new instinctive or learned primitives in this new cognitive cycle. Feeding back the Navigation Module's intermediate results back to the sensory stages and then processing the intermediate results in the next cycle by the Navigation Module, 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. Schneider (2022a) shows that by feeding back and re-operating on the intermediate results, the architecture can generate causal behavior.

($action_t \neq$ "move*" and $\mathbf{WPR}_t \neq$ ["discard*"]) or $\mathbf{WPR}_t$ = ["feedback*"],
⇒ `Nav_Mod.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)

($action_t \neq$ "move*" and $\mathbf{WPR}_t \neq$ ["discard*"]) or $\mathbf{WPR}_t$ = ["feedback*"], ⇒	-if $action_t$ which the Navigation Module 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 $\mathbf{WPR}_t$ to discard $action_t$ (i.e., the Navigation Module should not do anything this cognitive cycle) then operations after the ⇒ symbol will occur -if Working Primitive $\mathbf{WPR}_t$ specifies that the intermediate results of the Navigation Module should be fed back, then regardless of whether $action_t$ is actionable or not, the operations after the ⇒ symbol will occur
⇒`Nav_Mod.feedback_to_assocn_mod(`$\mathbf{WNM'}_t$`)`	-pseudocode to feed back and store the intermediate results of the Navigation Module, i.e., the Working Navigation Map $\mathbf{WNM'_t}$ to the Input Sensory Vectors Association Modules
⇒$\forall_\sigma$: $\mathbf{LNM}_{(\sigma, \gamma, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`$\mathbf{WNM'}_{t-1}$`)`	-pseudocode that 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 in the last cognitive cycle) will be propagated towards the Navigation Module and operated on again

Table 16. Explanation of Symbols and Pseudocode in Equations (88) – (89)

Initial Input:	*not applicable as this is a continuation of Navigation Module 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 complex (i.e., the Object Segmentation Gateway Module, the Causal Memory Module and the Navigation Module)
Second Output:	-in the next cognitive cycle, the Navigation Module 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, as in the previous section -if *action*$_t$ is not actionable then intermediate results of the Navigation Module (i.e., **WNM'**$_t$) may be fed back again to the Input Sensory Vectors Association Modules (88, 89)
Description:	-In the previous section we saw that the Navigation Module 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 and then to the external embodiment to activate an actuator or send an electronic signal. -In this section we see that if *action*$_t$ is not actionable then intermediate results of the Navigation Module (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 where it can be operated on again. -By feeding back and re-operating on the intermediate results, the architecture can generate causal behavior.

Table 17. Summary of the Feedback Operations from the Navigation Module per Equations (88) – (89)

3. The Emergence of Analogical Inductive Properties in the CCA5

3.1 Emergence of Analogical Problem Solving in the CCA5

In the last section above, we showed how with enhanced feedback pathways from the Navigation Module to the Input Sensory Vectors Association Modules (Figure 16) and some small changes, if there is no actionable output from the Navigation Module, then we 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 to the Navigation Module and operated on again (Figure 16). Schneider (2022a) shows 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.

With the small but important changes to the architecture of the CCA5, we describe here a modification of the feedback algorithm from the Navigation Module 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 we saw in (88) that if the *action*$_t$ which is produced by the Navigation Module (i.e., *action*$_t$ = `Nav_Mod.apply_primitive`(**WPR**$_t$, **WNM'**$_t$) (82)) is not actionable (i.e., there is no actuator to move or no electronic signal to send) then we feed back what will now be considered intermediate results of the Navigation Module (i.e., **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, **WNM'**$_t$ is treated as the input signal (89) and thus automatically propagated back to the Navigation Module where it can be operated on. Perhaps different instinctive or learned primitives operating on **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 **WPR**$_t$ can specify to discard results (i.e., **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 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_Mod.feedback_to_assocn_mod(`**WNM'**$_t$`)` (88)
⇒ $\forall_\sigma$: $\mathbf{LNM}_{(\sigma, \gamma, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`**WNM'**$_{t-1}$`)` (89)

The small changes in the architecture in the CCA5 presented in this paper are sufficient to easily allow the emergence of a different type of feedback by the Navigation Module. 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. We replace (88, 89) by (88a, 89) to specify this condition where the previously described feedback algorithm will still occur:

$\mathbf{WPR}_t$ = ["feedback*"],
⇒ `Nav_Mod.feedback_to_assocn_mod(`**WNM'**$_t$`)` (88a)
⇒ $\forall_\sigma$: $\mathbf{LNM}_{(\sigma, \gamma, t)}$ = `Input_Sens_Vectors_Assoc_Module`$_\sigma$`.extract_`σ`(`**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 (i.e., *action*$_t$ = `Nav_Mod.apply_primitive(`$\mathbf{WPR}_t$, **WNM'**$_t$`)` (82)) is not actionable (*action*$_t \neq$" move*"—there is no actuator to move or no electronic signal to send) then we feed back what will now be considered intermediate results of the Navigation Module (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 $\mathbf{WPR}_t$ specifies "analogical" feedback then this will occur even if the *action*$_t$ signal is actionable. Similarly, if $\mathbf{WPR}_t$ 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.

There is no equation for this, but from Figure 17 we can see that **WNM'**$_t$ is also sent to the TempMap region of the Navigation Module. This happens automatically. This is a new change in the CCA5 version of the architecture. Results of almost all operations from the Navigation Module and the Causal Memory Module are routinely copied to the TempMap region. Often the information in the TempMap region 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 we describe the TempMap region as an array **TempMap** which acts like an ordinary navigation map (66).

In (92) the pseudocode `Nav_Mod.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, in the CCA5 version of the architecture, results of the Navigation Module 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_Mod.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). This is illustrated in Figure 18.

The next cognitive cycle then occurs. At this point, the contents of the **WNM'**$_t$ Working Navigation Map in the Navigation Module 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.

To indicate we are continuing equations (90–93), in (94) we specify "(*action*$_{t-1}$ ≠ "move*" or **WPR**$_{t-1}$ = ["analogical*"]) and **WPR**$_{t-1}$ ≠ ["discard*"] and **WPR**$_{t-1}$≠ ["feedback*"])**,**" and then we specify to run this specific pseudocode: **WNM'**$_t$ = `Nav_Mod.retrieve_and_add_vector_assocn()` (94). Rather than propagate **WNM'**$_{t-1}$ towards the Navigation Module and then it replaces the existing **WNM'**$_t$ and is operated on by the current Working Primitive **WPR'** , as occurred automatically with the previous feedback algorithm, the pseudocode `retrieve_and_add_vector_assocn()` specifies that **WNM'**$_{t-1}$ should not replace but simply be added to the existing **WNM'**$_t$. This is depicted in Figure 19.

Note that now the original Working Navigation Map **WNM'**$_t$ in the Navigation Module also contains the action that occurred in the past of a similar Working Navigation Map in a possible analogical situation. The example in the section below will further illustrate the analogical inductive solution these equations allow the architecture to generate in response to new situations.

((*action*$_t$ ≠" move*" or **WPR**$_t$ = ["analogical*"]) and **WPR**$_t$ ≠ ["discard*"] and **WPR**$_t$ ≠ ["feedback*"])**,**
⇒`Nav_Mod.feedback_to_assocn_mod(`**WNM'**$_t$`)` (90)
⇒ **WNM'**$_t$ = `Causal_Mem_Mod.match_best_map(`**WNM'**$_t$`)` (91)
⇒ **TempMap**$_t$ = `Nav_Mod.use_linkaddress1_map(`**WNM'**$_t$`)` (92)
⇒ **WNM'**$_t$ = `Nav_Mod.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_Mod.retrieve_and_add_vector_assocn()` (94)

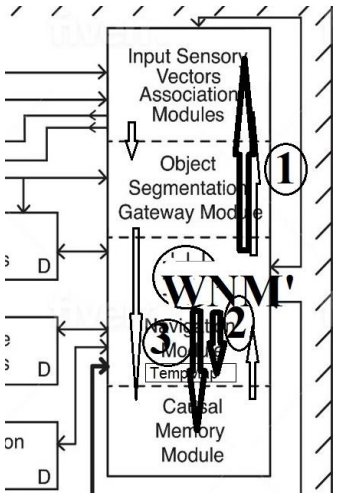

Figure 17. Now if an *action* signal is not actionable, the intermediate results of the Navigation Module as represented by **WNM'**$_t$ are:
#1 – (same as before) fed back as before to the Input Sensory Vectors Association Module (equation (90));
#2—(new, automatic for CCA5) sent to and stored in the TempMap region of the Navigation Module (no equation – in the CCA5 architecture results of operations from the Navigation Module and the Causal Memory Module are routinely and automatically copied to the TempMap region—not used here and will be overwritten in the next step)
#3—(new) sent to the Causal Memory Module triggering the matching with the closest navigation map (i.e., "best match") which now becomes set as the new Working Navigation Map **WNM'** (equation (91)).

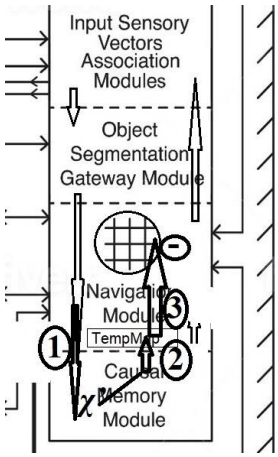

Figure 18. #1: TempMap$_t$ = `Nav_Mod.use_linkaddress1_map(`**WNM'**$_t$`)` (92)—The current Working Navigation Map's **WNM'** most recently used ***linkaddress χ'*** (i.e., an address pointing to another cell in the same or usually another navigation map, or an entire other navigation map) is sent to the Causal Memory Module causing that navigation map to become activated.
#2: In the CCA5 architecture results of the Causal Memory Module are automatically stored in **TempMap**. Thus, the navigation map which ***linkaddress χ'*** pointed to, is now in **TempMap**. (Recall that in the CCA5, **TempMap** is an array that functions like a normal navigation map. It is implemented as a "TempMap" region in the Navigation Map.)
#3: WNM'$_t$ = `Nav_Mod.subtract(`**WNM'**$_t$, **TempMap**`)` (93)— The pseudocode operates with the navigation maps (i.e., which are arrays) **WNM'**$_t$ (the current Working Navigation Map which from equation (91) (Figure 17) represents the best matching navigation map of the original **WNM'**) and **TempMap** (which contains the navigation map which ***linkaddress χ'*** pointed to). Navigation map **WNM'**$_t$ is subtracted from navigation map **TempMap**, with the result becoming the new **WNM'**$_t$ (93).

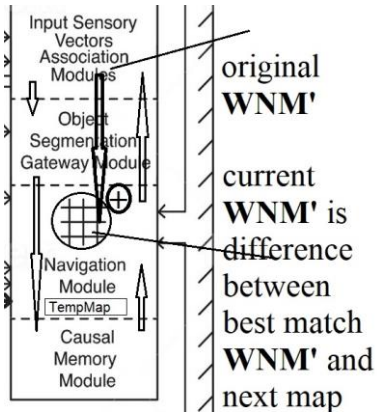

Figure 19. The original Working Navigation Map **WNM'**$_{t-1}$ originally fed back to the Input Sensory Vectors Association Modules (90) is now added to the current Working Navigation Map **WNM'**$_t$ (94). Equation (94) specifies that Working Navigation Map **WNM'**$_{t-1}$ now stored in the Input Sensory Vectors Association Modules should be added to the current Working Navigation Map **WNM'**$_t$ in the Navigation Module: **WNM'**$_t$ = `Nav_Mod.retrieve_and_add_vector_assocn()`. We are adding the difference between the original **WNM'** best matching navigation map's ***linkaddress χ'*** (i.e., the navigation map called most recently in the past by that best matching navigation map) navigation map and the best matching navigation map (i.e., what happened?) to the original **WNM'**. As a result, the new **WNM'** produced contains the event (navigation map, possible *action*) that occurred in the past of a similar Working Navigation Map in a possible analogical situation.

WPR$_t$ = ["feedback*"], ⇒	-replaces equation (88) (see Table 17) -previously described feedback algorithm now only occurs if the Working Primitive **WPR**$_t$ specifies it (i.e., **WPR**$_t$ = ["feedback*"],).
⇒`Nav_Mod.feedback_to_assocn_mod(`**WNM'**$_t$`)`	-previously described in Table 17 -applied to equations (88)/(88a) or (90) -pseudocode to feed back and store the intermediate results of the Navigation Module, 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 17 -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 in the last cognitive cycle) will be propagated towards the Navigation Module 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 (i.e., *action*$_t$ = `Nav_Mod.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_Mod.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, 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_Mod.use_linkaddress1_map(`**WNM'**$_t$`)` → **TempMap**$_t$	-the pseudocode `Nav_Mod.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 CCA5 version of the architecture, results of the Navigation Module and the Causal Memory Module are routinely and automatically copied to **TempMap**
⇒`Nav_Mod.subtract(`**WNM'**$_t$, **TempMap**`)` (93) → **WNM'**$_t$	-the pseudocode `Nav_Mod.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_Mod.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 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 will contain the action or navigation map that occurred in the past of a similar Working Navigation Map in a possible analogical situation

Table 18. Explanation of Symbols and Pseudocode in Equations (88a) – (94)

Input:	*not applicable as this is a continuation of Navigation Module 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, we are adding below descriptive suffixes to **WNM'** (however, note that there is only one navigation map in the Navigation Module 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 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 operations*
Description:	-Earlier we saw that the Navigation Module 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 and then to the external embodiment to activate an actuator or send an electronic signal. -In the previous section we saw that if *action*$_t$ is not actionable then intermediate results of the Navigation Module (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 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 we show 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 we 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 we are using a navigation map that was tried in the past. -At the present, the algorithm underlying the pseudocode `Nav_Mod.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*. -In the next section we will show that induction by analogy is actually occurring in this process.

Table 19. Summary of the Analogical Feedback Operations from the Navigation Module per Equations (88a) – (94)

3.2 Induction by Analogy

In the previous section although it seems like we were moving around navigation maps here and there, induction by analogy actually is occurring.

Below we re-write equations (90 – 94) using the more descriptive terms for the Working Navigation Maps used in Table 19—to distinguish between the several different navigation maps which are considered the current Working Navigation Map **WNM'** at some point, we add below descriptive suffixes 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_Mod.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_Mod.use_linkaddress1_map(`**WNM'**$_t$-*best_match*`)` (97) *(from 92)*
⇒ **WNM'**$_t$-*difference* = `Nav_Mod.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_Mod.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). We now see that variable **y** has another property N (102). Therefore in (103) we can conclude by induction by analogy that variable **x** also has property N.

In (95) we can refer to as **WNM'**$_t$-*original* as variable **x**, or perhaps as navigation map **x**. We want 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 we assume it will share many properties. We explore what navigation map **y** does next (i.e., what navigation map does the ***linkaddress*** we chose link to). We see that 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). We will consider this difference, i.e., **WNM'**$_t$-*difference* to be property N. Since navigation map **y** has property N, therefore by induction by analogy, we can say that navigation map **x** also has property N (103). Thus, we 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)

Note that symbolic variable **x** (or **y**) simply *refers* to a neurosymbolic conceptual structure which is a particular navigation map. The navigation map is neither purely symbolic nor connectionist. A navigation map can be operated

on via connectionist approaches, in particular whether a navigation map contains certain information or not. However, a navigation map also has many symbolic features as well.

With regard to symbols, note that symbols such as, for example, "dog" or "cat" do not need to discretely exist in the architecture to represent dogs and cats, or to perform actions with regard to dogs or cats. (For example, something that matches with navigation maps representing dogs, is sufficient to, for example, to perform an action with regard to a dog in the environment.) If a more sophisticated language collection of instinctive primitives and navigation maps related to words exists, then yes, symbols such as "dog" or "cat" could be represented by navigation maps. Learned primitives allow encoding of almost anything in any higher-level format but the core mechanisms of the architecture discussed above are unchanged. The navigation map contains both symbolic and connectionist information. The navigation map contains representations of objects, of places, of concepts, of procedures, and so on.

With regard to analogical problem solving, work in this area goes back to the start of the field of artificial intelligence. Mitchell (2021) reviews the abilities of various artificial intelligence (AI) systems to form analogies. Symbolic analogy-making AI systems tend to require the inputs to be human curated logic statements. Symbolic machine analogy-making goes back the 1956 Dartmouth Summer AI Project (McCarthy and colleagues, 1955). Gentner's Structure-Mapping Engine (SME) is a symbolic approach based on the requirements that the analogy example and the target problem are in logical form and a representation is given, and it only maps the syntactical properties of the analogy to the target problem, not the context of the specific domain (Falkenhainer et al., 1986). As Mitchell notes the SME required much prior and human knowledge in order for it to work properly. Hofstadter and Mitchell's (1994) Copycat program was also a symbolic approach to analogy-making that had the advantage of building its own representation, but also required human structuring and knowledge. The Copycat program operated on letter analogies. For example, if **abc** changed to **abd** then how should **ijk** change, to which the program would respond **ijl**.

With the rise of deep learning in the last decade, there have been more connectionist solutions to analogy-making. Mikolov and colleagues (2013) describe how vector-space word representations capture syntactic and semantic regularities and so allow vector-oriented reasoning via offsets between words. They give the example of "King – Man + Woman" producing a vector close to the word "Queen." Wu and colleagues (2020) report that their Scattering Compositional Learner can arrange neural networks in a sequence to solve the compositional structure of a problem and use analogical reasoning. This system was able to demonstrate zero-shot solutions for many out-of-distribution Raven Progressive Matrices.

Mitchell (2021) acknowledges that the deep learning methods can bypass much of the human prior knowledge that symbolic systems for analogical reasoning have required in the past, but that massive training sets are still required. As well, a lack of transparency and performance via biases rather than understanding still are problems with these newer approaches. Mitchell notes that analogical problem-solving including abstractions is still an open problem in the field of artificial intelligence two thirds of a century since the 1956 Dartmouth Summer AI Project.

The Causal Cognitive Architecture 5 (CCA5) is a brain-inspired cognitive architecture. We show above how we accomplish analogical problem solving by moving around navigation maps, in a somewhat different method than typical approaches in the field of artificial intelligence. As well, note that elements of both symbolic and connectionist approaches are present in the CCA5 analogical problem solving. No claims are made that this is a superior method of analogical problem solving compared to methods developed in the field and briefly reviewed above. However, this assumed brain-inspired method of analogical solving feedback mechanism, in conjunction with a powerful collection of instinctive and learned primitives and a Causal Memory Module experienced with a vast number of stored navigation maps, may as a system prove extremely advantageous. In fact, analogical problem solving may be central to human cognition. Chen and colleagues (1997) showed that one year old infants are capable of analogical problem solving. Adult humans use analogies to solve day to day problems, to invent new solutions and to plan behavior. While analogical problem solving would seem to be a very conscious process, where we note similarities between a past situation and a new present situation, work by Reber and colleagues (2014) in functional magnetic resonance imaging (fMRI) showed that simple analogous episodes could be detected unconsciously.

There are many ways in which problems can be understood and be solved. However, humans tend to make extensive use of analogical problem solving. While polished mathematical proofs might seem to arise from pure deductive reasoning of the premises, in Pólya's (1954) classic reference on mathematical discovery, in fact the first book is dedicated to human-like induction and analogy. Gick and Holyoak (1980) note that new theories in science often depend on taking an analogy from a different domain and applying it to the new target theory. They present experiments based on a story analogy and a target problem which can be solved by taking advantage of the story analogy. They note that the analogical problem-solving process requires three main steps:

i. Construction of a representation of the story analogy and the target problem.

ii. Mapping of the representation of the story analogy onto the representation of the target problem via detecting similar relations in the story analogy and the target problem.

iii. Using the mapping to create what they term a parallel solution to the target problem.

Analogical reasoning is used not only by human scientists but by all humans in making sense of day-to-day life. Hofstadter (2001) has defined "concepts" as "packages of analogies." In addition, analogical problem solving may be unique to humans. Penn, Holyoak and Povinelli (2008) note that while comparative psychologists tend to stress the biological continuity between humans and nonhumans, in fact with regards to the mind, there appears to be a strong discontinuity in the ability of nonhumans to understand the relational aspects of a physical symbol system. Chimpanzees are able to respond with an action if two objects are the same or are different (a relational match-to-sample task), and while this may be necessary for analogical reasoning, fuller analogies can involve causal logic and finding similarities between relations which are perceptually different. Except for one unreplicated report, no chimpanzee or other nonhuman appeared capable of fuller analogical problem-solving (Penn, Holyoak and Povinelli, 2008). However, there have been more recent reports of nonhumans with various aspects of analogical problem-solving behavior (Flemming et al., 2013; Hagmann et al., 2015). Flemming and colleagues (2011) note that while chimpanzees like humans can complete relational match-to-sample tasks, more distantly related monkeys have difficulty with these problems.

Given the Causal Cognitive Architecture 5's brain-inspired roots, it is unsurprising that analogical problem solving readily emerges from the architecture. 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.

((*action*$_t$ ≠" move*" or **WPR**$_t$ = ["analogical*"]) and **WPR**$_t$ ≠ ["discard*"] and **WPR**$_t$ ≠ ["feedback*"]), ⇒	-see Table 18
⇒`Nav_Mod.feedback_to_assocn_mod` (**WNM'**$_t$-*original*) (95)	-see Table 18 - **WNM'**$_t$-*original* – the current Working Navigation Map **WNM'**$_t$ which we give 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 18 **WNM'**$_t$-*original* – the current Working Navigation Map **WNM'**$_t$ which we give the label 'original' for better understanding of what is happening
⇒`Nav_Mod.use_linkaddress1_map(` **WNM'**$_t$-*best_match*) (97) → **TempMap**$_t$	-see Table 18 -**WNM'**$_t$-*best_match* – the current Working Navigation Map **WNM'**$_t$ which we give the label 'best-match' for better understanding of what is happening
⇒`Nav_Mod.subtract(`**WNM'**$_t$-*best_match* , **TempMap**$_t$) (98) → **WNM'**$_t$-*difference*	-see Table 18 - **WNM'**$_t$-*difference* – the current Working Navigation Map **WNM'**$_t$ which we give 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 18
⇒ **WNM'**$_t$-*analogical* = `Nav_Mod.retrieve_and_add_vector_assocn()` (99) → **WNM'**$_t$-*analogical*	-see Table 18 - **WNM'**$_t$-*analogical* – the current Working Navigation Map **WNM'**$_t$ which we give the label 'analogical' for better understanding of what is happening
P_1**x** & P_2**x** & … P_n**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**y &** P_2**y &** … P_n**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 - we can conclude by induction by analogy that variable (navigation map) **x** also has property N

Table 20. Explanation of Symbols and Pseudocode in Equations (95) – (103)

Input:	*not applicable as this is a continuation of Navigation Module operations*
Output:	- **WNM'**$_t$*-analogical* (99)
Description:	-See Table 18. In equations (95 – 99) we re-write 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, we add the descriptive suffixes to **WNM'** listed above. - Equations (100 – 103) we consider a definition of induction by analogy applied to the operations performed on the Working Navigation Maps.

Table 21. Summary of the Analogical Feedback Operations from the Navigation Module per Equations (95) – (103)

3.3 Demonstration Example of Analogical Problem Solving in the CCA5

Analogical problem solving is typically thought of as the type of question seen on human intelligence tests (Prade, Richard, 2011). For example, analogies may be in the form of, for example, "sock is to foot, as glove is to what?" The answer would be "hand" as a glove goes on the hand, just as a sock goes on a foot.

While we have been talking about analogical problem solving in the CCA5, it will not on its own solve the type of analogies seen on human intelligence tests. The analogical problem solving in the CCA5 is a ubiquitous mechanism in the core functioning of the architecture. If a sensory input cannot be processed to an actionable output, then analogical reasoning, i.e., induction by analogy, is used to help find and create a navigation map which could better lead to an advantageous actionable output. However, there is no "analogical center" in the CCA5 which would be used to solve analogical problems. Rather, analogical problems as well as just about all other problems which the architecture encounters are solved by the core mechanisms which heavily use analogical reasoning, in conjunction with the instinctive and learned primitives the embodiment of the architecture has access to.

In order to solve human analogical problems such as socks going onto feet and thus gloves go onto hands, a developed set of instinctive primitives and learned primitives are required, which the architecture does not have at this point. Chollet's Abstraction and Reasoning Corpus (ARC) is an intelligence benchmark which is intended to be based on "innate human priors" to avoid results being skewed by previous knowledge and experience, as is the case for most intelligence tests (Chollet, 2019). While the CCA5 at this point does not have a full set of instinctive primitives and some necessary learned primitives to approach the level of "innate human priors," we will nonetheless take a problem from Chollet's ARC and simplify it somewhat to be in keeping with the more limited instinctive priors in the experimental computer simulation of the CCA5. To solve more complex problems in Chollet's corpus, additional instinctive primitives (as well as enhancements to the learned primitive system of the CCA5—what Chollet considers "innate human priors" will not be fully instinctive) could be added to the CCA5. Even something that seems straightforward as solving a problem is not—learned primitives are required by the CCA5 to even know that a problem is to be solved to have an answer. As well, so as not to have to provide encoded simulated inputs for the CCA5, enhancements to the visual perceptual system would allow the existing core architecture to solve problems more autonomously.

A Python computer simulation incorporating the equations of this paper (1) to (106) has been created. In this simulation we are using 6x6x0 sized navigation maps. Examples in this section are encoded and entered into the visual perception system as simulated-like inputs, with null inputs for the auditory and olfactory systems. The CCA5 does not understand, i.e., does not have the learned primitives to understand, the concept of solving an analogical problem. Instead, we enter Chollet's simplified problem as a simulated sensory input and examine the automatic analogical problem solving in the feedback mechanism.

Consider an example where the CCA5 embodiment is a mobile robot. It sees a simulated visual input of a simplified Chollet ARC problem, and the Working Navigation Map **WNM'** produced from this visual input is shown in Figure 20. The visual system can recognize lines, solids, water, and a few other features which are considered at a perceptual level. The CCA5 considers that its embodiment is on the SOLID cell in Figure 20, i.e., position (1,2) with the origin in the top left corner. It's $\mathbf{GOAL}_t$ is to move either east, west, north, or south. It can move on solids but not on water. What direction should it move next?

	WATER				
WATER	SOLID				

Figure 20. Working Navigation Map $\mathbf{WNM'}_t$ – what *action* should occur?
(To distinguish among the different navigation maps that will be set as **WNM'**,
we will call this navigation map "$\mathbf{WNM'}_t$-*original*" .)

The Working Navigation Map **WNM'** in Figure 20 is operated on by the Working Primitive **WPR** and produces an *action* signal: $action_t$ = `Nav_Mod.apply_primitive`($\mathbf{WPR}_t$, $\mathbf{WNM'}_t$) (82). Unfortunately, the $\mathbf{WPR}_t$ chosen when operated on in the Navigation Module on $\mathbf{WNM'}_t$ was not able to produce an actionable output, i.e., $action_t \neq$" move*". This is not surprising. Looking at Figure 20, if the CCA5 embodiment is on the cell "SOLID" it is indeed hard to say which direction it should move to so as to be able to land in the physical world to a location corresponding to another SOLID cell, i.e., it does not want to land in the water, i.e., a cell "WATER" corresponding to water in the physical world.

According to equation (95), since $action_t \neq$" move*", $\mathbf{WNM'}_t$-*original* will be fed back and stored until the next cognitive cycle in the Input Sensory Vectors Association Modules: `Nav_Mod.feedback_to_assocn_mod`(**WNM'**$_t$-*original*. Then in equation (96) $\mathbf{WNM'}_t$-*original* (i.e., Figure 20) is also sent to the Causal Memory Module where it is matched against the best matching navigation map there. The best matching navigation map is then set as the Working Navigation Map : $\mathbf{WNM'}_t$-*best_match* = `Causal_Mem_Mod.match_best_map`(**WNM'**$_t$-(*original*)) (96). This is shown in Figure 21.

WATER	WATER				
WATER	SOLID				

Figure 21. Best match of Figure 20 from the Causal Memory Module becomes the
new Working Navigation Map **WNM'** which we label as "$\mathbf{WNM'}_t$-*best_match*".

In equation (97) we then access the navigation map in the Causal Memory Module that $\mathbf{WNM'}_t$-*best_match*'s most recently used ***linkaddress*** points to. This navigation map is automatically sent to **TempMap** by the Causal Memory

Module: **TempMap**$_t$ = `Nav_Mod.use_linkaddress1_map(`**WNM'**$_t$*-best_match*`)` (97). This is shown in Figure 22. To find what is the difference, i.e., the new property N of equation (102), we then subtract **WNM'**$_t$*-best_match* (Figure 21) from **TempMap**$_t$ (Figure 22): **WNM'**$_t$*-difference* = `Nav_Mod.subtract` (**WNM'**$_t$*-best_match* , **TempMap**$_t$) (98). **WNM'**$_t$*-difference* is displayed in Figure 23.

WATER	WATER				
WATER	SOLID				
	SOLID				

Figure 22. This is the navigation map in **TempMap** that is the navigation map which **WNM'**$_t$*-best_match* of Figure 21 most recently linked to.

	SOLID				

Figure 23. WNM'$_t$*-difference*—this is the difference between **TempMap** (i.e., the navigation map which **WNM'**$_t$*-best_match* most recently linked to (Figure 22) and **WNM'**$_t$*-best_match* (Figure 21).

	WATER				
WATER	SOLID				
	SOLID				

Figure 24. WNM'$_t$*-analogical*—this will become the current **WNM'**$_t$ to which **WPR**$_t$ will be applied and generate a new *action*$_t$ value. In the example discussed in the text the *action*$_t$ value is "**<move south>**" . This is an actionable *action*, and it is propagated to the output stages of the architecture. The CCA5 architecture automatically thus produced an analogical result for **WNM'**-*original* which can be tried again and may have a higher possibility of producing an advantageous actionable result, as it does here.

In (103) we were able to conclude by induction by analogy that just as navigation map **y** (i.e., here which would correspond to **WNM'**$_t$*-best_match*) has property N, then navigation map **x** (i.e., which corresponds to **WNM'**$_t$*-original*) should also have property N. Thus, property N, which corresponds to **WNM'**$_t$*-difference* (Figure 23) is added to navigation map **x**, which corresponds to **WNM'**$_t$*-original*. Equation (99) shows **WNM'**$_t$*-original* (Figure 20) returning to the Navigation Module and being added to **WNM'**$_t$*-difference* (Figure 23): **WNM'**$_t$*-analogical* = `Nav_Mod.retrieve_and_add_vector_assocn()`. **WNM'**$_t$*-analogical* is displayed in Figure 24.

The Working Navigation Map **WNM'**$_t$ in the Navigation Module is now the analogical result shown in Figure 24, i.e., **WNM'**$_t$*-analogical*. This Working Navigation Map (i.e., **WNM'**$_t$*-analogical*) will now be operated on by the Working Primitive **WPR**. An *action* signal is produced: *action*$_t$=`Nav_Mod.apply_primitive`(**WPR**$_t$, **WNM'**$_t$)

(82). Now there is a SOLID cell to the south of the current position of the embodiment. The $action_t$ value is "**<move south>**" . This is an actionable *action*, and it is propagated to the Output Vector Association Module and then the Output Vector Shaping Module and then the actual actuators which will move the embodiment in a southward direction.

This example shows that if an actionable resolution of a navigation map is not immediately possible (i.e., an instinctive or learned primitive is applied to a navigation map resulting from various sensory inputs, and there is not an actionable output), the architecture will automatically produce an analogical result which can be tried again and may have a higher possibility of producing an advantageous actionable result.

The mechanism of producing this analogical result is largely via matching navigation maps, copying some navigation maps, adding navigation maps, and subtracting navigation maps—all in keeping with a feasible evolutionary pathway for a brain-inspired cognitive architecture.

It is important to note again that analogical problem solving has become a ubiquitous core mechanism in the Causal Cognitive Architecture 5 (CCA5). There is no "analogical module" for solving human-like analogical intelligence tests. Rather, solving such higher-level analogy problems makes use of the instinctive and learned primitives of the architecture, in conjunction, with the core mechanisms of the architecture which includes, of course, the analogical problem solving described above.

4. A Solution to the Grounding Problem in the CCA5

4.1 Solutions Emerge from the Architecture

The role of brain maps of sensory and other information has been postulated for many years. Damasio (1998, 2003) suggested that much of thought was grounded in neural structures that provided map-like organization of information with respect to physical aspects of the body. Advances in neuroscience research in the last few decades has better revealed the importance of navigation in the mammalian brain, particularly in the hippocampus (O'Keefe, Nadel, 1978; Samsonovich, Ascoli, 2005; Alme, Miao, Jezek, Treves, Moser, Moser, 2014; Moser, Rowland, Moser, 2105; Wernle, Waaga, Mørreaunet, Treves, Moser, Moser, 2018; Sugar, Moser, 2019; O'Keefe, Krupic, 2021).

The basis for the early versions of what eventually would be called the Causal Cognitive Architecture essentially was the postulate that the mammalian cortex arose largely from a large-scale duplication of hippocampal-like circuitry. Although the evolution of brain pathways for producing intelligent behavior in the brain is still not well understood, it is thought that duplication of pathways may play an important role (e.g., Chakraborty and Jarvis, 2015). The implication was that just as navigation properties were a key function of the hippocampus, they should be so for the entire mammalian cortex. Thus, a "navigation map" data structure was postulated as the basic data structure for all cortical brain functions, and it was implemented in a cognitive architecture (e.g., Schneider, 2021, 2022a, 2022b).

Obviously, navigation is an important requirement of robotics, and other researchers have considered its importance in artificial intelligence, robotics, and cognitive architectures. For example, Epstein (2017) considers cognitive and robotic modeling of spatial navigation. Schafer and Schiller (2018) discuss how the hippocampus as well as possibly various cortical regions store maps of both spatial features and non-spatial features such as sound, time, rewards, concepts, and even social information about others. This can result in both better navigation and as well as decision making. Hawkins, Lewis, Klukas, Purdy and Ahmad (2019) discuss how neurons similar to the grid cells in the hippocampal regions representing the location of an animal or human in their environment, may also be present in the neocortex.

Although there is a "Navigation Module" in the Causal Cognitive Architecture (e.g., Figure 1), the "Navigation Module" is not just used for navigation but for every decision (other than reflex and autonomous decisions) the cognitive architecture makes about anything in day-to-day life, even if it has no relation to navigation whatsoever. The Causal Cognitive Architecture is a brain-inspired cognitive architecture (BICA). However, the architecture does not attempt to replicate a biological system at the neuronal spiking level. As well, the models do not implement every known detail of mammalian neurophysiology. For example, evidence indicates that in the reptilian ancestor of mammals, the cortex had three layers, similar to allocortex in mammals such as parts of the olfactory cortex involved in processing olfactory inputs (Luzzati, 2015). However, we have avoided special modelling of olfactory pathways in the Causal Cognitive Architecture and for the time being treat olfaction similar to the other senses which relay through the thalamus to the neocortex. The purpose of the architecture at this time, is to consider what the result would be of building a mammalian brain inspired cognitive architecture around the navigation map data structure.

Schneider (2021) showed that a cognitive architecture built around the navigation map data structure would exhibit pre-causal behavior, but with a small enhancement to one of the feedback pathways, that full causal behavior would emerge (as well as the increased tendency towards psychotic-like behavior if any flaws were present in the system). Schneider (2022a) showed that with modifications to the Schneider (2021) architecture, solutions to both the spatial and temporal binding problems emerged easily from the new architecture (the Causal Cognitive Architecture 3, CCA3). Schneider (2022b) showed that with a number of algorithmic changes and additional operations, that analogical operations could be incorporated into the architecture's core decision making. The new architecture was called the Causal Cognitive Architecture 4 (CCA4). Schneider (2022b) touched upon the grounding symbol issue, but did not provide a full solution, instead taking a "pragmatic approach" to the issue.

Above we showed that with modifications to the architecture of Schneider (2022a) and Schneider (2022b), inductive analogical operations could more readily emerge from the architecture, albeit, again as part of the architecture's core decision making. This new architecture is called the Causal Cognitive Architecture 5 (CCA5). Below we show that this architecture will now readily allow emergence of a solution to the full grounding problem, i.e., both for concrete objects and abstract situations and concepts, as well as for any incompletely acquired information that is poorly grounded. We also demonstrate that this full solution involving the inductive analogical feedback mechanism provides significantly improved problem-solving abilities for the architecture.

4.2 The Symbol Grounding Problem and the Navigation Map Architectural Solution

Above in the Introduction section we discussed Harnad's solution (1990, 1994) of the symbol grounding problem by grounding symbols with their real-world sensations, interactions, and linkages. Within the Causal Cognitive Architecture 5 (CCA5) new information, via the equations (1) to (99) above, enters the system through the sensory systems. It will automatically be mapped to real-world sensations and actions. No information is kept as an isolated symbol—*everything* in the CCA5 is within a navigation map linked to other navigation maps.

In Figure 2 an example of a navigation map is given. Features are either "solid" or "water". Although both these features can be explained in terms of even lower-level physical sensory inputs, they are both considered lowest-level, grounded sensory inputs in terms of the instinctive primitives. Many times, a visual sensory scene will be changed into such a mapping by the time the Working Navigation Map **WNM'** is constructed. The instinctive primitives (and learned primitives as well) considered and selected for the Working Primitive **WPR** are able to successfully operate on such lowest level, grounded sensory inputs.

However, limiting features to "solid" or "water" is not sufficient for processing real-world inputs. Consider Figure 25, which is a photo of a bridge over a river. After processing through the input sensory stages, matching with the best navigation map in the Causal Memory Module, segmentation of objects, and updating with the actual sensory information, Figure 26 is the Working Navigation Map **WNM'** which results. The current version of the CCA5 makes little use of language, but language is not essential for its operation or to consider grounding at a basic level.

Consider an embodiment of the CCA5 that has little experience with bridges and will not necessarily recognize one. As shown in Figure 26, during object segmentation operations there was no perception of a "bridge" (either as a word or as non-language representation of a bridge), but rather, the Working Navigation Map **WNM'** produced shows two types of solids. The solids are labeled by the system as "solid32" (corresponding to the grass in the scene) and "solid43" (corresponding to the stones of the bridge).

The processing of this **WNM'** depends on the Working Primitive **WPR** chosen by the system and indirectly by the **GOAL** navigation map. If the CCA5's embodiment needed to go from a solid area and cross the river to go to the solid area on the other side, then the "solid43"-containing cells would form a pathway. Instinctive primitives exist which can find solid pathways. The architecture would not need to know much more about what "solid43" means than it is a solid.

We have not in detail discussed the creation and utilization of learned primitives in this paper, but crossing water would now also trigger a learned primitive $\mathbf{WLP}_t$ (78). There would not be immediate acceptance of "solid43" as solid. The link to other navigation maps about "solid43", e.g., ***linkaddress*** {1043} (Figure 26) would be followed. If there was an experience about falling into the water via walking on "solid43" it would be there. There is not such experience. However, there is the experience(s) of successfully walking on "solid43" (stones) in another situation. Thus, the original instinctive primitive to follow the path along "solid43" which are treated as "solid" *feature*s by the instinctive primitive, across the river would be followed. Thus, the embodiment of the CCA5 would walk over the bridge (without actually knowing the term "bridge" in this formal language-less implementation of the architecture) and successfully cross the river without falling into the water.

Figure 25. Bridge over a river in the real-world environment.

solid32, *link*{1032}	solid32, *link*{1032}	solid32, *link*{1032}	solid32, *link*{1032}	solid32, *link*{1032}	solid32, *link*{1032}
solid32, *link*{1032}	solid43, *link*{1043}	solid43, *link*{1043}	solid32, *link*{1032}	water	water
solid32, *link*{1032}	solid43, *link*{1043}	water	water	water	water
water	solid43, *link*{1043}	water	water	water	water
solid32, *link*{1032}	solid43, *link*{1043}	solid32, *link*{1032}, *link*{4574}	solid32, *link*{1032}	solid32, *link*{1032}	solid32, *link*{1032}
solid32, *link*{1032} *iprimitive*{8974}	solid43, *link*{1043}	solid32, *link*{1032}	solid32, *link*{1032}	solid32, *link*{1032}	solid32, *link*{1032}

Figure 26. Working Navigation Map **WNM'** derived from visual sensory input of the visual scene of a bridge over a river in Figure 25. The cells in the Working Navigation Map **WNM'** contain the *feature*s "water", "solid32" (grass), and/or "solid43" (stone). *link{ }* represents a ***linkaddress*** to another cell, often in another navigation map.

Note that automatically by virtue of equations (1) to (99) and the architecture of the CCA5 (e.g., Figure 1), grounding emerges and occurs. As per Harnad's solution (1990, 1994) of the symbol grounding problem, symbols (even non-formal language, internal ones such a "solid43" in this example) are automatically grounded with their real-world sensations, interactions, and linkages. These real-world sensations, interactions and linkages are very advantageous to solving problems using a minimum of instinctive and learned primitives.

4.3 Failure of The Symbol Grounding Problem and an Inductive Analogical Solution

Consider the example again of crossing the river shown in Figure 25. The CCA5 does not have experiences or recognition of the concept of "bridges" but instead processes the sensory scene as represented in the Working Navigation Map **WNM'** shown in Figure 26—the scene is mapped into *feature*s of "solid32" (grass), "solid43" (stone), or "water". In this example, the CCA5 does *not* have experience of previously walking on "solid43" (stones) before, but instead had this and other information electronically fed into it bypassing many of the normal sensory processes. Thus, this information is not necessarily grounded with their real-world sensations, interactions, and linkages. Just as it can be very efficient for human students to read and memorize lists about different aspects of their world than need to experience everything firsthand, it can be similarly efficient to feed information into a CCA5 architecture/embodiment in a way such that it does not need to experience everything. However, whether one is a human or a CCA5 architecture, without the grounding experience the information as such may not be as useful. In this paper we focus on the Causal Cognitive Architecture rather than human cognition. Thus, without direct experiential experience with regard to "solid43" (i.e., the CCA5 architecture acquired this information much as a human memorizes

a table, without experiencing the information in the table), we ask if the embodiment of the CCA5 architecture will take a path on the "solid43" elements (i.e., the stones of the bridge) in order to cross the river?

Continuing this example, the CCA5 had not had any negative experiences about using any solid pathways in the past to cross water. The CCA5 has memorized information about "solid43", largely that it is in the solid category as given by its name. As in the example in the previous section, the instinctive primitive which becomes the Working Primitive **WPR** operates on the Working Navigation Map **WNM'** to produce actions to follow the path along the "solid43" (stone) containing cells of the navigation map. These "solid43" containing cells are treated as "solid" by the instinctive primitive. It turns out that since indeed "solid43" (stone) *feature*s are indeed capable of supporting the weight of the embodiment of the CCA5, it successfully crosses the river, as in the example in the section above.

Consider now a new example. We will first consider a CCA5 that is *not* using its core inductive analogical feedback mechanism. The sensory scene is shown in Figure 27. There is a river filled with leaves. The CCA5 wants to cross the body of water and go to the other side.

The CCA5 processes the sensory information of Figure 27 and the Working Navigation Map **WNM'** shown in Figure 28 is constructed. The *feature* "solid08" (leaves) is a feature which had been electronically loaded as a list into the CCA5's memory. It has never had any real experience with "solid08" (leaves) before. As before, it does have some limited factual information about "solid08", particularly that it is in the solid category. Thus, as in the example of stepping on "solid43" (stone) which the CCA5 had no experience with but since it was a solid, the instinctive primitive mapped out a pathway over "solid43" (stone), this time since "solid08" (leaves) is a solid, the instinctive primitive will also map out a pathway over the cells containing the *feature* "solid08" so that it can cross the river. Figure 29 shows the transformation of the Working Navigation Map **WNM'** of Figure 28, into a Working Navigation Map **WNM'** which the Working Primitive **WPR** can operate on. A pathway of cells containing the *feature* "solid" is clearly visible to the reader in Figure 29 and readily usable for the Working Primitive **WPR** to cause actions which will direct the embodiment of the CCA5 on this pathway to cross the river. Unfortunately, the embodiment of the CCA5 steps onto the floating leaves, and falls into the river, possibly damaging its internal elements from the water immersion.

The next time the CCA5 saw floating leaves, it would recognize the danger of stepping onto floating leaves, and avoid this action. As well, the next time the CCA5 saw any solid element it did not recognize and was trying to cross a river, it might call an instinctive (or learned) primitive and test the strength of the solid object before crossing the river. Even if there was piece of concrete ("solid 71") across the river (as in a concrete bridge) and it had no experience with "solid71" perhaps it would not blindly follow a "solid71". Rather, it might again call an instinctive (or learned) primitive and test the strength of the solid object (i.e., the concrete "solid71") before crossing the river.

However, let us consider Figure 27, again, and consider a CCA5 which has never seen this particular river before, has never tried to cross this river before, and has never tried to walk on leaves ("solid08") before. However, as before, this CCA5 has instinctive primitives that find a pathway over solid objects in order to cross bodies of water such as a river. Also, this CCA5 has crossed other bodies of water, including crossing rivers stepping on stones such as "solid43" in Figure 25. This time, we will use the full capabilities of the architecture including the core inductive analogical feedback mechanism. We will see that this mechanism can be very advantageous in grounding information, especially with regard to information which has been fed electronically into the CCA5 or which the CCA5 has read about, and does not have much sensory experience with, or even information which the CCA5 has experienced but perhaps only tangentially and there is poor linkage (i.e., grounding) of the information.

In our new example, the CCA5 encounters the sensory scene shown in Figure 27—a river covered with leaves, i.e., "solid08". There is a similar **GOAL** navigation map activated as in the example in the previous section, i.e., the CCA5 wants to cross the river to go to the other side. In the past this embodiment of the CCA5 has crossed a river stepping on "solid43" (i.e., a stone bridge as shown in Figure 25). As well, this embodiment of the CCA5 has attempted to cross a deep puddle that had some newspapers ("solid22") floating in it, and experienced the newspapers not providing any advantage in crossing the puddle, i.e., its legs got wet although there was no damage and crossing the puddle went fine otherwise.

Similar to the previous example, Figure 28 shows the Working Navigation Map **WNM'** that results from the sensory scene of Figure 27 being processed by the CCA5. The CCA5 has had some information about "solid08" previously electronically fed into its visual Input Sensory Vectors Association Module (Figure 1). It has knowledge that "solid08" (leaves) are thin sheets, causally from "solid33" (trees). It is able to recognize "solid08" on the river, and incorporate these *feature*s into the Working Navigation Map **WNM'** produced (Figure 28). However, it has no actual experience with "solid08". It will be able to recognize it as a solid, and it will be able to access the ***linkaddress*** {1008} where it can obtain additional information that "solid08" is a thin sheet, and that it causally derives from

"solid33" (trees). Again, the CCA5 has no other actual experience with "solid08", just this memorized-like factual knowledge.

For one moment, let us assume this is again the previous example with a Causal Cognitive Architecture without the causal or analogical feedback properties. The instinctive primitive which becomes the Working Primitive $\mathbf{WPR}_t$ (equations (74–81)) operates on the Working Navigation Map $\mathbf{WNM}_t$ attempting to find a solid path over the water. It will first attempt to convert the *feature*s specified, based on the navigation maps specified in the "*link*{ }", to "solid" as that is what the instinctive primitive forming $\mathbf{WPR}_t$ actually operates on. If additional information about the *feature* is in the specified ***linkaddress*** then it will use that information. In transforming "solid08" to "solid", it has no experience (and thus no navigation maps) of walking on "solid08". It will not be able to process the information in the ***linkaddress*** *link*{1008} much further in terms of resolving if "solid08" is a "solid", the *feature* $\mathbf{WPR}_t$ requires in order to plan a path across the river. Thus, it will as a default take that the *feature* is a solid since it already is in a "solid" classification such as "solid08". Figure 29 shows the previous $\mathbf{WNM'}_{t-1}$ Figure 28 transformed into a Working Navigation Map $\mathbf{WNM'}_t$ which the Working Primitive $\mathbf{WPR}_t$ can operate on. Unfortunately, some of the cells with *feature*s of "solid" will actually be leaves floating in the water. And as occurred before in the previous example, the Working Primitive $\mathbf{WPR}_t$ produced actions (equations (82–87)) directing the embodiment of the CCA5 to step on the floating leaves which resulted in failure, i.e., falling the water.

Now let us continue the example where we are using the analogical feedback mechanisms of the CCA5 architecture. The instinctive primitive which becomes the Working Primitive $\mathbf{WPR}_t$ operates on the Working Navigation Map $\mathbf{WNM'}_t$ (Figure 28) attempting to find a solid path over the water. It will attempt to convert the *feature*s specified, based on the navigation maps specified in the "*link*{ }", to "solid" as that is what the instinctive primitive forming $\mathbf{WPR}_t$ actually operates on. If additional information about the *feature* is in the specified ***linkaddress*** then it will use that information. In transforming "solid08" to "solid", again it has no experience (and thus no navigation maps) of walking on "solid08". Previously, it was not able to process the information in the ***linkaddress*** *link*{1008} much further in terms of resolving if "solid08" is a "solid", the *feature* $\mathbf{WPR}_t$ requires in order to plan a path across the river. However, this time, there will occur automatically the inductive analogical feedback mechanism described above in equations (88–94).

The ***linkaddress*** *link*{1008} (leaf) navigation map will be matched (equation (96)) to the navigation map associated with "solid22" (newspaper) since they both have similar properties of being thin and being on the water. However, the navigation map associated with "solid22" has the experience of not supporting the embodiment of the CCA5 in the past over a deep puddle, i.e., it is not solid. (To the reader, of course leaves are solid. However, to the simple instinctive primitives the leaves are not what it considers solid, i.e., it does not stop falls into water.) This information will be become part of the new Working Navigation Map $\mathbf{WNM'}_t$ (equations (95–99).

In the next cognitive cycle, this new Working Navigation Map $\mathbf{WNM'}_t$ will be operated on by another new Working Primitive $\mathbf{WPR}_t$ to see if it is solid. It will turn out not to be.

In the next cognitive cycle, the Working Primitive $\mathbf{WPR}_t$ accesses the previous Working Navigation Map (Figure 28). Cells in that navigation map with the *feature* "solid08" will have the *feature* changed to "water". In another cognitive cycle, other cells in that navigation map will have "solid" placed where the CCA5 can best determine, as in the previous paragraph, there is a solid. This continues for a few, but reasonably limited number of cognitive cycles.

The transformed new Working Navigation Map $\mathbf{WNM'}_t$ is shown in Figure 30. The Working Primitive $\mathbf{WPR}_t$ will trigger the pathfinding primitive to become $\mathbf{WPR}_t$. However, based on the navigation map $\mathbf{WNM'}_t$ in Figure 30 no path across the river is seen. There will be attempts by analogy again, but the mechanism will not find anything actionable in this CCA5 that provides a solution. Thus, the embodiment of the CCA5 will not cross the river at this point.

Not only does the inductive analogical feedback mechanism allow an instantaneous (so to speak, noting that a dozen or so cognitive cycles may be required) grounding of abstract or poorly grounded features or concepts, but there is a record of the linkages made during these grounding operations. The ***linkaddresses*** of the *feature* "solid08" *link{1008}* navigation map, will have links added that point to the newly transformed Working Navigation Map **WNM'** shown in Figure 30. This navigation map will be saved in the Causal Memory Module and links are updated in the other connected navigation maps (as indicated in equation (70) which happens typically by default for operations in the Causal Memory Module). Thus, the *feature* "solid08" is not only instantaneously better grounded (i.e., that it acts like "water" when in pathfinding mode trying to cross a body of water) but permanently better grounded with this linkage now in the Causal Memory Module.

The inductive analogical feedback mechanism thus provides a solution to the symbol grounding problem when information is acquired by the CCA5 in a non-experiential manner such as it being fed in electronically, or perhaps

where the architecture is reading and memorizing a table of information, or even experientially acquired information where the acquisition is poor and incomplete.

Figure 27. Leaves filling a river. There is some noise in the sensory scene which will be ignored when constructing the Working Navigation Map **WNM'** consisting of cells with the *feature*s "water", "solid08" (leaves), "solid32" (grass), "solid33" (trees), and/or "air" (the sky as well as perceived empty places). While the "solid08" (leaves) are considered a "solid", if the CCA5 embodiment walks on them to cross the river, it will fall into the river.

air	air	air	air	air	air
solid33, *link*{1033}	solid08, *link*{1008}	solid08, *link*{1008}	solid08, *link*{1008}	solid33, *link*{1033}	solid33, *link*{1033}
solid33, *link*{1033}	solid08, *link*{1008}	solid08, *link*{1008}	solid08, *link*{1008}	solid33, *link*{1033}	solid32, *link*{1032}
solid33, *link*{1033}	water	solid08, *link*{1008}	solid08, *link*{1008}	solid32, *link*{1032}	solid32, *link*{1032}
solid32, *link*{1032}	water	solid08, *link*{1008}	solid08, *link*{1008}	solid33, *link*{1033}	solid32, *link*{1032}
water	water	water	water	solid08, *link*{1008}	solid32, *link*{1032}

Figure 28. Working Navigation Map **WNM'** derived from visual sensory input of a river filled with leaves ("solid08") in Figure 27. *Feature*s include: "water", "solid08" (leaves), "solid32" (grass), and/or "solid33" (trees), and/or "air" (the sky as well as perceived empty places). (Note that if a 3D Working Navigation Map is constructed then the sky *feature*s (i.e., the "air" *feature*s in the top row of the navigation map) will not be present on the same plane as the "solid" and "water" *feature*s.)

air	air	air	air	air	air
solid	solid	solid	solid	solid	solid
solid	solid	solid	solid	solid	solid
solid	water	solid	solid	solid	solid
solid	water	solid	solid	solid	solid
water	water	water	water	solid	solid

Figure 29. Non-analogical transformation of the Working Navigation Map **WNM'** of Figure 28 into a Working Navigation Map **WNM'** which the Working Primitive $\mathbf{WPR}_t$ can operate on. Unfortunately, some of the cells with *feature*s of "solid" will actually be leaves floating in the water.

air	air	air	air	air	air
solid	water	water	water	solid	solid
solid	water	water	water	solid	solid
solid	water	water	water	solid	solid
solid	water	water	water	solid	solid
water	water	water	water	water	solid

Figure 30. Analogical transformation of the Working Navigation Map **WNM'** of Figure 28 into a Working Navigation Map **WNM'** which the Working Primitive $\mathbf{WPR}_t$ can operate on. By analogical comparison with "solid22" (the newspaper on the water) the *feature* "solid08" (the leaves on the water) is not considered to be a solid, i.e., it will not support the weight of the embodiment of the CCA5.

4.4 Formalizing the Above Solutions to the Grounding Problem

Above in Section 4.2 we described how the CCA5 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 CCA5 is within a navigation map linked to other navigation maps. Above in Section 4.3 we described how through a core inductive analogical feedback mechanism of the architecture, the CCA5 provides a solution to better grounding information acquired in a non-experiential manner such as it being fed in electronically, or in even in the case of experiential acquisition of information where the acquisition is poor and incomplete.

A *feature* was defined initially in (34). As discussed earlier, 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 $\boldsymbol{all\_LNMs}_{\chi}$ and that the *feature* also was present in some sensory system array $\mathbf{S}_{\sigma,t}(1–7)$ represented by $\boldsymbol{s}(t)$ (9). This implies that the CCA5 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. This is in keeping with Harnad's solution (1990, 1994) of the symbol grounding problem by grounding symbols with their real-world sensations, interactions, and linkages.

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. We 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. 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.

In equation (105) we are stating 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, we are assuming 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. The worst case would be a piece of information electronically fed into the CCA5 not associated with any other piece of information. In this worst case, there would still be a link to the event of electronically feeding in data to the system and the date of occurrence. We would still consider this grounding, although a very poor grounding. Future usage and experience, including usage of the *feature* and occurrence of the analogical feedback mechanism, may better ground the piece of information. Similar occurrences happen with humans as well. For example, a human introductory science student may memorize for examination preparation a feature of quantum physics that has little to do with day-to-day life. The student may just memorize it and associate it with something irrelevant such as the examination he/she is studying for. Perhaps in the human student's future studies that feature of quantum physics will be encountered again but in a richer context of experiments done by humans or a richer mathematical context relating that quantum feature to other aspects of tangible events, all which will better ground the feature.

$$\begin{aligned} grounded\_feature \;=\; & \forall_{feature}: \; (\; feature \in \boldsymbol{all\_LNMs_{\chi}} \text{ AND } feature \in \boldsymbol{s}(t) \;) \\ & \text{OR} \\ \forall_{feature}: \; (\; (\; feature \in \mathbf{WNM'}_t \text{ AND } & action_{t-1} \neq \text{"move*" AND } \mathbf{WPR}_{t-1} \neq [\text{"feedback*"}]) \text{ OR} \\ & \mathbf{WPR}_{t-1} = [\text{"analogical*"}] \;) \end{aligned} \quad (104)$$

$$\forall_{\chi,t}: \boldsymbol{all\_navmaps_{\chi,t}} = grounded\_feature \text{ OR } \texttt{link}(\boldsymbol{all\_navmaps}_{\chi,t}) \neq [\;] \text{ OR } \boldsymbol{cellfeatures_{\chi,t}} = [\;] \quad (105)$$

For every addressable navigation map in the Causal Cognitive Architecture 5 (CCA5), *feature*s can be considered to be grounded or to be used in a grounded fashion. Thus, the symbol grounding problem is largely solved within the architecture. However, below we consider the more advanced cases of grounding of abstract concepts and the relation between grounding and language.

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* ∈ $\mathbf{WNM'}_t$ AND $action_{t-1}$ ≠ "move*" AND $\mathbf{WPR}_{t-1}$ ≠ ["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 we consider that *feature* as a *grounded_feature* - although the older feedback mechanism *may* result in grounding the feature adequately, often it won't, thus we are 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`($\boldsymbol{all\_navmaps}_{\chi,t}$) ≠ [ ] 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 22. Explanation of Symbols and Pseudocode in Equations (104) – (105)

Input:	*not applicable as this is a continuation of Navigation Module operations*
Output:	*not applicable as this is a continuation of Navigation Module operations*
Description:	-These equations specify that for every addressable navigation map in the Causal Cognitive Architecture 5 (CCA5), *feature*s can be considered to be grounded or to be used in a grounded fashion.

Table 23. Summary of the Grounding Requirements with the Architecture per Equations (104) – (105)

4.5 The Abstract Symbol Grounding Problem and the CCA5

In the introductory section we noted that Harnad's solution to the grounding problem (1990, 1994), i.e., grounding symbols with their real-world sensations, interactions, and linkages, would seem to be satisfactory for concrete objects. However, how can we achieve grounding of symbols representing more abstract concepts? We noted above that Reinboth and Farkaš (2022) have considered whether abstract concepts can be represented within grounded cognition. "Concrete" concepts tend to apply to physical objects and interactions, while "abstract" concepts tend to refer to situations and events (Dijkstra et al., 2014). Reinboth and Farkaš (2022) consider that all cognition, including abstract concepts, are grounded in experience—via the body's sensorimotor interactions with the environment or via linguistic and social communication. They discuss a graded utilization of these different grounding pathways, noting that abstract concepts indeed may very well be grounded via the direct pathway of interactions with the external world.

We believe it is more realistic to accept that certain abstract concepts are simply not that well grounded. It is not that there is no grounding. Above, with regard to equation (105) we discussed how by virtue of the ***linkaddresses*** there will be grounding of anything in any addressable navigation map in the CCA5 architecture. However, although we do not provide a quantitative score with regard to symbol grounding strength, we discussed above how some groundings are better and some are poorer. The worst case of grounding in the CCA5 would be a piece of information electronically fed into the CCA5 (i.e., not learned experientially) and not associated with any other piece of information. In this worst case, there would still be a link to the event of electronically feeding in data to the system and the date of occurrence. We would still consider this symbol grounding, although a very poor grounding.

As we noted above, human cognition also involves more richly and poorly grounded experiences. We gave that example of a human introductory science student memorizing for an upcoming examination a feature of quantum physics that has little to do with day-to-day life. The student may just memorize it and associate it with something irrelevant such as the examination he/she is studying for. However, grounding can improve. Perhaps in another course

the future the student learns how the mysterious concept he/she memorized is associated with a day-to-day experience of Newtonian physics. For example, a "spin quantum number" term memorized for the previous test, now becomes better grounded in concepts about angular momentum the student already possesses.

Consider the similar (but hypothetical—current experimental computer simulations of the CCA5 have minimal instinctive and learned primitives) case of an advanced/experienced CCA5 that memorized the term "spin quantum number" (or had the term electronically loaded into a Local Navigation Map **LNM**). If the CCA5 then read or experienced that the "spin quantum number" was related to the rotation of an electron around an axis, mostly likely that cognitive cycle there would be no actionable $action_t$ (the system is learning something, no action to do anything is immediately obvious). There would as a result be an analogical feedback cycle, and the other navigation maps holding information about related concepts such as the rotation of bodies and concepts of electrons (and possibly other particles) retrieved, and some linking would occur that cognitive cycle—the grounding of this very abstract concept would become richer.

Equations (104) and (105) would appear to successfully allow grounding of abstract concepts in the CCA5. However, we note from an engineering point of view, that there is a spectrum in the richness of the grounding of different concepts in the particular individual CCA5, and similarly, in individual humans as well.

4.6 The Symbol Grounding Problem in Natural Language and the CCA5

The symbol grounding problem is often expressed in terms of how words (i.e., English or some other natural language words) acquire and represent meaning. A word by itself does not really have that much meaning. As noted in the introductory section, Harnad (1990) gives an example of a person with no understanding of the Chinese language trying to learn Chinese from a Chinese-Chinese dictionary. If this person sees a Chinese language word, the dictionary will not be of much help. The person will unfortunately go from one string of symbols (words) to another string of symbols (words) without any meaning attached to the symbols (words). The symbols (words) are grounded in the other Chinese symbols (i.e., words) which actually provide no meaning to this person. However, if this English-speaking person, for example, saw a common English-language word written on a piece of paper, that word would be "grounded" for the person, and that word would have meaning for the person—the person understands the real-world (or internal-world, fantasy-world, and so on) objects and/or actions to which that word refers.

Enabling grounding of artificial intelligence systems that use natural language has been challenging. For example, Tellex and colleagues (2011) give an example of a human supervisor asking a robot to "put the tire pallet on the truck." The words in the command must be mapped to the external world.

One approach to ground the words in a command such as in the example above is given by Winograd (1970) and other researchers—create symbol systems where each word or term is directly connected to pre-specified actions and environments. This approach may not work well for real-world applications as there is little learning with little perceptual feedback, and the action space is quite constrained.

Another approach is for the robot to treat the words in a command as another sensory input, and to learn the meaning of the words in terms of other sensations and physical actions. For example, Marocco and colleagues (2010) describe a humanoid robot learning the meaning of commands by physical interactions with its environment.

Another approach is given by Tellex and coworkers. They take a probabilistic approach to mapping words in the command to concrete objects, places, and paths in the real-world environment. There is learning and feedback in this mapping process.

It is possible to discuss symbolic grounding problem in the CCA5 in terms of language as it does have the potential for language understanding and generation. Schneider (2022a) discusses explainability and the emergence of language in the Causal Cognitive Architecture, albeit CCA3 version. These properties remain in the CCA5 version of the architecture discussed in this paper. The explainability of the CCA5 or any intelligent system is its ability to give reasons for its behavior (Gilpin, Bau, Yuan et al., 2018). The navigation maps held by the Causal Memory Module of the CCA5 effectively provide explanations of why the CCA5 produced a particular $action_t$ output given particular sensory input values for a certain problem. A sequence of the Working Navigation Maps **WNM'** produced (and stored in the Causal Memory Module, i.e., `Causal_Mem_Mod.store_WNM_update_links(`**WNM'**$_t$`)` (70)) creates a history of the operations of the Navigation Module. This history represents an explanation of why the CCA5 produced a particular $action_t$ output given particular sensory input values for a certain problem.

An emergent property of the Causal Cognitive Architecture is that the storage of the Working Navigation Maps **WNM'** gives a reasonable explanation of why a particular answer was given for a particular problem, i.e.,

explainability emerges from the architecture. From this emerges natural language. In order to generate explanations or concepts or commands (all which occur in the architecture via navigation maps) which other CCA5 embodiments can understand, language capabilities are needed. Some way of communicating what is in a navigation map(s) in one CCA5 embodiment to another CCA5 embodiment is required. In the CCA3 this was implemented as a number of `navigation_map_to_proto_language()` instinctive primitives (Schneider, 2022a). This remains in the CCA5 but has been renamed `apply_primitive_nav_to_protolang()`.

In (106) `Nav_Mod.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 to the Output Vector Association Module (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 and then on to the actual output actuators (Figure 14).

If a brain-inspired approach to the architecture is kept, then there are engineering issues with regard to being able to communicate while simultaneously producing output actions to effect movements using the same circuitry in the CCA5 architecture (e.g., Figure 14). As well, if auditory outputs are to be used, then there also may be the need for more specialized versions of `Sequential_Mod.motion_correction()` in (86) and `Output_Vector_Mod.apply_motion_correction()` in (87). These and other details of language emergence in the CCA5 are beyond the scope of the present paper. However, if we assume high speed electronic outputs are to be used then such specialized circuitry or specialized versions of the primitives may not be needed.

In the current computer simulations of the CCA5, only a very simplistic application of equation (106) is used at present with a vocabulary of only a few nouns and verbs. However, the architecture does possess the start of a proto-language, and so we can discuss the issue of symbol grounding and language. This proto-language in the CCA5 uses the same navigation maps and largely the same equations as discussed throughout this paper. The proto-language ability is not a new module (although it could be in the future for engineering reasons) added to the architecture, but rather an emergent property that comes from having an explainable history of Working Navigation Maps that can be conveyed to other embodiments of the architecture (or humans). Thus, equations (104) and (105) which ensure grounding in the architecture also apply to language in the architecture. There are no isolated words, whether verbs or nouns, in the CCA5—everything is part of a navigation map and everything is grounded to some extent as per the above equations.

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

`Nav_Mod.apply_primitive_nav_to_protolang`(**WPR**$_t$, **WNM'**$_t$) → *action*$_t$	-`Nav_Mod` refers to the Navigation Module (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 - 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 CCA5 embodiment or a human - see Table 14 for more details of converting **WPR**$_t$ **and WNM'**$_t$ into an actionable output, which in this case represents a communication

Table 24. 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, 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 (Figure 14) where more detailed instructions are created to effect the required actuator outputs
Description:	-The Navigation Module applies the Working Primitive **WPR**$_t$ against the current Working Navigation Map **WNM'** in the Navigation Module, 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 25. Summary of the Operation of Proto-Language per Equation (106)

4.7 Non-Addressable Navigation Maps in the CCA5 and Grounding

Equations (104) and (105) above showed that for every addressable navigation map in the Causal Cognitive Architecture 5 (CCA5), *feature*s can be considered to be grounded or to be used in a grounded fashion. Thus, we considered the symbol grounding problem as largely solved within the architecture.

Above in equations (31–33) we defined an addressing scheme to address any particular cell within any particular navigation map within ***all_navmaps***$_t$. In the example given above, $\chi_{\text{modcode=*Causal\_Memory\_Mod*, } mapno=3456,2,3,4}$ is cell x=2,y=3,z=4 in map number 3456 in the Causal Memory Module. The vector ***all_navmaps*** is defined in (30). It represents all the navigation maps in all of the Input Sensory Vectors Association Modules (i.e., all of the different sensory Local Navigation Maps **LNM**s), all of the navigation maps in the Causal Memory Module (i.e., all of the multi-sensory Navigation Maps **NM**s), all of the navigation maps in the Instinctive Primitives Module (i.e., all of the Instinctive Primitives **IPM**s) and all of the navigation maps in the Learned Primitives Module (i.e., all of the Learned Primitives **LPM**s).

However, modified navigation maps are used in most of the other modules of the architecture, but are not directly addressable. For example, a modified navigation map used in an Input Sensory Vectors Shaping Module is not addressable as per equations (31–33). Given that equations (104) and (105) apply to addressable navigation maps, would such a modified navigation map used, for example, in the visual Input Sensory Vectors Shaping Module, be considered grounded?

Harnad (1990, 1994) discusses symbol grounding with reference to the sensorimotor system sensing and processing the external world. The modified navigation maps used in the visual Input Sensory Vectors Shaping Module as well as the other related input modules, the Sequential/Error Correcting Module, the Output Vector Association Module as well as the Autonomic Reflex Modules (provide simple reflexive action to external and internal stimuli) (Figure 1) are not included in the ***all_navmaps***$_{\chi,t}$ term of equation (105). However, these modified navigation maps in are indeed grounded in sensorimotor sensing or the external motor system of the architecture and its interactions with the external world. Thus, as per Harnad, the majority of the modified, non-addressable navigation maps used in the architecture should be considered grounded. However, there may be some modified navigation maps in the architecture which are not adequately grounded. For example, the modified navigation maps used in the Goal/Emotion Module may not be grounded in the sensorimotor system sensing and processing of the external world. The significance of such ungrounded navigation maps is uncertain at present. Future larger computer simulations of the architecture may better reveal the behavioral consequences.

It is interesting to note that explainability and language aspects of the modified navigation maps is also limited. Equation (106) utilizing the primitive `Nav_Mod.apply_primitive_nav_to_protolang()` will apply only to addressable navigation maps. There will be little explainability and communication by the architecture of how, for example, a visual input signal was transformed in the visual Input Sensory Vectors Shaping Module. Similarly, there will be little explainability by the architecture as to the inner workings of the Goal/Emotion Module.

5. Discussion

The Causal Cognitive Architecture is a brain-inspired cognitive architecture (BICA). With inspiration from the mammalian hippocampus, the Causal Cognitive Architecture uses the navigation map as its basic data element—information learned about the environment as well as instinctive and learned procedures, are all represented in small spatial maps, i.e., the navigation maps. Examples are shown in Figures 2, 26, 28, 29, and 30. A new version, the Causal Cognitive Architecture 5 (CCA5) is shown in Figure 1. This version of the architecture has a specialized short-term memory area, as well as much stronger feedback pathways more conducive to inductive analogical properties being able to be applied to the symbol grounding problem. The core mechanisms, represented above in equations (1) to (106), have also been better optimized for the architecture and new ones added.

Sensory features from the environment (or simulated in simulations) stream in from different perceptual sensors. Objects detected in the streams of sensory features are segmented. Visual, auditory, and other sensory features of each segmented object 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, as shown in in the example in Figures 4 and 5. The resulting map is called the Working Navigation Map **WNM'** (equations (63) – (70)).

Instinctive and learned primitives, which act as small rules or productions, and which also are stored in modified navigation maps, then operate on the Working Navigation Map **WNM'**. This causes the Navigation Module to produce an action signal (equations (71) – (82)), which goes to the Output Vector Association Module and then to the actuators of the CCA5's embodiment (Figure 6) (equations (83) – (87)).

The Causal Cognitive Architecture makes liberal use of feedback pathways—states of a downstream module can influence the recognition and processing of more upstream sensory inputs. As was noted above, in the CCA5 the feedback pathways between the Input Sensory Vectors Association Modules and the Navigation Module/ Object Segmentation Gateway Module are enhanced so that they allow intermediate results from the Navigation Module to be stored in the Input Sensory Vectors Association Modules. As shown in Figure 3, results of operations of the Navigation Module can be temporarily fed back and stored in the Input Sensory Vectors Association Modules. In the next cognitive cycle, these intermediate results from the Navigation Module are treated as the sensory inputs for that cognitive cycle, and they are propagated forward towards the Navigation Module where they can be processed again. Schneider (2022a) shows that 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 (Schneider, 2022a).

Further experimentation with the feedback pathways has revealed that inductive analogical reasoning can emerge from the Causal Cognitive Architecture (Schneider, 2022b). In this paper, we have better optimized the feedback pathways, structures, and mechanisms that allow inductive analogical reasoning. These pathways are depicted in Figures 17, 18, and 19, and formalized in equations (95) to (99). It is important to note that analogical problem solving is not a separate module of the CCA5 (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.

A detailed comparison with mammalian neurophysiology and evolution is beyond the scope of this paper. As well, as noted above, the architectures presented in this paper as well as in Schneider (2020, 2021, 2022a, 2022b) are all loosely brain inspired—there is no attempt to duplicate brain function down to the level of spiking neurons, nor is there an attempt to capture every neurophysiological element in the architecture. However, the emergence of inductive analogical reasoning in the architecture presented in this paper would be further supportive of the argument that relatively small architectural and thus genetic changes were required to go from a non-human primate brain to a human-level brain. Elements of this argument have been presented in more detail in Schneider (2020, 2021).

As noted above inductive analogical problem solving does not emerge in the Causal Cognitive Architecture as a special ability for the architecture to use when taking an intelligence test or trying to solve a difficult problem. Rather it emerges as part of the core mechanism of the architecture, that will be used ubiquitously in solving even trivial day-to-day problems. This is in fact supported by the psychological literature. As noted above, analogical reasoning has been shown to be an innate skill in 13-month-old infants (Chen et al., 1997). In adults, Reber and coworkers (2014) have shown in functional magnetic resonance imaging (fMRI) studies that analogical reasoning between a past event and a new analogous situation, occurs unconsciously. From a behavioral point of view, Hofstadter (2001) notes a plethora of evidence that supports the use of analogies as the core of human cognition. From an evolutionary point of view, Vendetti and Bunge (2014) note that small enhancements of connections in the human brain's lateral

frontoparietal network compared to primate brains, could explain much more robust analogical problem solving in humans compared to other primates.

Presented in this paper, was the finding that a full solution to the grounding problem in turn can emerge from the inductive analogical reasoning of the architecture. Most information acquired by the CCA5 is stored within a navigation map linked to other navigation maps. A grounded feature is a *feature* in a navigation map that arises from a sensory input to the architecture or related to an action produced by the architecture. The navigation map readily allows a straightforward application of Harnad's solution (1990, 1994) to the symbol grounding problem by grounding symbols with their real-world sensations. In the architecture presented in this paper, by virtue of the ***linkaddresses*** there will be grounding of most *features* in any cell of any addressable navigation map in the CCA5 architecture. This is formalized in parts of equations (104) and (105).

In a number of cases the navigation map alone will not be sufficient to allow symbol grounding of newly acquired information. Examples include when information is acquired by the CCA5 in a non-experiential manner such as it being fed in electronically, or perhaps where the architecture is reading and memorizing a table of information, or even experientially acquired information where the acquisition is poor and incomplete. The inductive analogical feedback mechanism of the architecture can provide a solution to the symbol grounding problem in these cases. This is reflected in other parts of equations (104) and (105).

Above we presented an example of leaves filling a river (Figure 27). The sensory scene is transformed into a Working Navigation Map **WNM'** consisting of cells with *features* such as "water", "solid08" (leaves), "solid32" (grass), "solid33" (trees), and/or "air" (the sky as well as perceived empty places). The CCA5 is controlling an embodiment that wants to go from one side of the river to the other side.

The CCA5 has had some information about "solid08" previously electronically fed into its visual Input Sensory Vectors Association Module (much as a human student memorizes a list for an examination without much understanding of the meaning of its contents). The CCA5 has knowledge that "solid08" (leaves) are thin sheets, causally from "solid33" (trees). It is able to recognize "solid08" in the sensory scene, and incorporates this *feature* into the Working Navigation Map **WNM'** produced (Figure 28). However, it has no actual experience with "solid08". It will be able to recognize it as a solid, and it will be able to access the ***linkaddress*** {1008} where it can obtain additional information such as that "solid08" is a thin sheet, and that it causally derives from "solid33" (trees).

Without better grounding of the "solid08" *feature* (i.e., the leaves; no formal language is used yet in the implementations or simulations of the CCA5), the Working Navigation Map **WNM'** of the sensory inputs in Figure 28 is transformed (i.e., via a Working Primitive $\mathbf{WPR}_t$ as the first step in finding a path across the water) into the Working Navigation Map **WNM'** shown in Figure 29. Cells have been simplified with *features* such as "air", "solid" or "water"—*features* the pathfinding instinctive primitive can work with. The instinctive primitive calculating a path to go from one side of the river to the other, will attempt to find a path on cells containing a "solid" *feature*. Unfortunately, some of the cells with a *feature* of "solid" will actually be leaves floating in the water. When the CCA5's embodiment attempts to walk on these leaves it will fall in the river and become damaged.

However, with full inductive analogical feedback, better grounding of the "solid08" (leaves) *features* occurs. As the example presented showed, "solid08" matches with a navigation map for "solid22" which contains information that represents it being thin sheets (actually of newspaper, but that word does not exist in the navigation map) on water. The link in that navigation map leads to another navigation map of getting wet stepping on it in a deep puddle. Getting wet stepping onto the "solid22" gets added, via the inductive analogical feedback mechanism, to the "solid08" original Working Navigation Map **WNM'**. In the next cognitive cycle, the Working Primitive $\mathbf{WPR}_t$ transforming the *features* in the Working Navigation Map **WNM'** into "air", "water", or "solid", will transform the *features* "solid08" in the Working Navigation Map **WNM'** to "water".

The transformed new Working Navigation Map $\mathbf{WNM'}_t$ is shown in Figure 30. No path across the river is now found by the pathfinding Working Primitive $\mathbf{WPR}_t$. Thus, the embodiment of the CCA5 will not cross the river at this point and become damaged. In this example, "solid08" became better grounded, with a linkage to an analogous experience when the embodiment of the CCA5 in the past stepped on the newspaper sheets floating in water. Note how better grounding provides an important problem-solving advantage.

Future work includes enhancing the current simple computer simulations of the CCA5 with an adequate collection of instinctive primitives so that a larger selection and more complex problems can be encountered by the architecture. Similarly, improvements to the learned primitives system are required for better demonstration and exploration of the architecture.

The grounding problem should be considered as an engineering issue. Ensuring information is better grounded allows the architecture presented above, and by analogy other intelligent systems, a more powerful approach to solving problems the system can encounter in its operations. From a BICA (brain-inspired cognitive architecture) point of

view, it would appear that small changes to a cognitive architecture making use of brain-inspired navigation maps allow analogical reasoning and a solution to the grounding problem to readily emerge. These small changes provide a guide to a possible evolutionary pathway to the emergence of the human mind requiring a reasonable number of genetic changes.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

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

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

Bisk, Y., Holtzman, A., Thomason, J., Andreas, J., Bengio, Y., Chai, J., Lapata, M., Lazaridou, A., May, J., Nisnevich, A. (2020). Experience Grounds Language. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8718–8735. Open-source version: *arXiv*: 2004.10151

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, **47**(1), 139–159. doi.org/10.1016/0004-3702(91)90053-M

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. doi.org/10.1098/rstb.2015.0056

Chen, Z., Sanchez, R., & Campbell, T. (1997). From beyond to within their grasp: Analogical problem solving in 10- and 13-month-olds. *Developmental Psychology*, **33**, 790-801

Chollet, F. (2019). On the Measure of Intelligence. In: *arXiv*:1911.01547.

Damasio A. R. (1998). Investigating the biology of consciousness. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, **353**(1377), 1879–1882. doi.org/10.1098/rstb.1998.0339

Damasio, A. (2003). Mental self: The person within. *Nature* **423**, 227. doi.org/10.1038/423227a

Davidsson, P. (1993). Toward a General Solution to the Symbol Grounding Problem: Combining Machine Learning and Computer Vision. *AAAI Technical Report* FS-93-04. AAAI, Palo Alto, USA.

de Vries, H., Shuster, K., Batra, D., Parikh, D., Weston, J., & Kiela, D. (2018). Talk the walk: Navigating grids in New York City through grounded dialogue. In *ICLR 2019 Conference/ OpenReview.net*, paper 389.

Dijkstra, K., Eerland, A., Zijlmans, J., & Post, L. S. (2014). Embodied cognition, abstract concepts, and the benefits of new technology for implicit body manipulation. *Frontiers in psychology*, **5**, 757. doi.org/10.3389/fpsyg.2014.00757

Dubova, M. (2022). Building human-like communicative intelligence: A grounded perspective. *Cognitive Systems Research* **72**:63-79. doi.org/10.1016/j.cogsys.2021.12.002

Epstein, S.L. (2017). Navigation, Cognitive Spatial Models, and the Mind. *AAAI 2017 Fall Symposium: Technical Report FS-17-05*.

Falkenhainer, B., Forbus, K.D., Gentner, D. (1986). The structure-mapping engine. In: *Proceedings of the National Conference on Artificial Intelligence*, AAAI, pp. 272-277.

Feldman, J. (2013). The neural binding problem(s). *Cognitive Neurodynamics*. **7**(1):1-11. doi: 10.1007/s11571-012-9219-8

Flemming, T.M., Thompson, R.K., Beran, M.J., Washburn, D.A. (2011). Analogical reasoning and the differential outcome effect: transitory bridging of the conceptual gap for rhesus monkeys (Macaca mulatta). *J Exp Psychol Anim Behav Process.* **37**(3):353-60. doi: 10.1037/a0022142.

Flemming, T.M., Thompson, R.K., Fagot, J. (2013). Baboons, like humans, solve analogy by categorical abstraction of relations. *Anim Cogn.* **16**(3):519-24. doi: 10.1007/s10071-013-0596-0.

Gick, M. L., Holyoak, K. J. (1980). Analogical problem solving. *Cogn. Psychol*. **12**, 36–355 10.1016/0010-0285(80)90013-4

Gilpin, L.H., Testart, C., Fruchter, N., Adebayo, J. (2019). Explaining Explanations to Society. *ArXiv: abs/1901.06560*

Guckelsberger, C., Kantosalo, A., Negrete-Yankelevich, S., & Takala, T. (2021). Embodiment and Computational Creativity. *arXiv:2107.00949*.

Guilford, J. P., Dunham, J. L., Hoepfner, R. (1967). Roles of intellectual abilities in the learning of concepts. *Proceedings of the National Academy of Sciences of the United States of America*, **58**(4), 1812–1817. doi.org/10.1073/pnas.58.4.1812

Hagmann, C.E, Cook, R.G. (2015). Endpoint distinctiveness facilitates analogical mapping in pigeons. *Behav Processes*. **112**:72-80. doi: 10.1016/j.beproc.2014.11.007.

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

Harnad, S. (1994). Computation Is Just Interpretable Symbol Manipulation: Cognition Isn't. *Minds and Machines* **4**:379-390

Hawkins, J., Lewis, M., Klukas, M., et al. (2019). A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex. *Frontiers in Neural Circuits* **12**:121.

Herzog M. (2008). Binding Problem. In: Binder M.D., Hirokawa N., Windhorst U. (eds) *Encyclopedia of Neuroscience*. Springer, Berlin, Heidelberg. doi.org/10.1007/978-3-540-29678-2_626

Hoffmann, M., Pfeifer, R. (2018). Robots as Powerful Allies for the Study of Embodied Cognition from the Bottom Up, in in A. Newen, L. de Bruin; & S. Gallagher, ed., *The Oxford Handbook 4e Cognition*, Oxford University Press, pp. 841-861. Open-source version: *arXiv:* 1801.04819

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.

Hofstadter, D.R., Mitchell, M. (1994). The Copycat project: A model of mental fluidity and analogy-making. In: Holyoak, K.J., Barnden, J.A. eds, *Advances in Connectionist and Neural Computation Theory*, vol **2**, 31-112. Ablex Publishing.

Kiela, D., Bulat, L., Vero, A. L., Clark, S. (2016). Virtual Embodiment: A Scalable Long-Term Strategy for Artificial Intelligence Research. *ArXiv:*1610.07432

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

Lake, B. M., & Murphy, G. L. (2021). Word meaning in minds and machines. *Psychological Review.* Advance online publication. doi.org/10.1037/rev0000297

Luzzati, F. (2015). A hypothesis for the evolution of the upper layers of the neocortex through co-option of the olfactory cortex developmental program. *Frontiers in Neuroscience.* **9**:162. doi:10.3389/fnins.2015.00162

Madl, T.; Baars, B.J.; Franklin, S. (2011). The Timing of the Cognitive Cycle. *PLoS ONE*, **6**, e14803.

Marocco, D., Cangelosi, A., Fischer, K., & Belpaeme, T. (2010). Grounding Action Words in the Sensorimotor Interaction with the World: Experiments with a Simulated iCub Humanoid Robot. *Frontiers in neurorobotics*, *4*, 7. doi.org/10.3389/fnbot.2010.00007

McCarthy, J., Minsky, M.L., Rochester, N., and Shannon, C.E. (1955). A Proposal for the Dartmouth Summer Research Project in Artificial Intelligence. Reprinted in AI Magazine, 2006, **27**, no.4, pp. 12-14.

McClelland, J. L., Hill, F., Rudolph, M., Baldridge, J., Schütze, H. (2020). Placing language in an integrated understanding system: Next steps toward human-level performance in neural language models. *Proceedings of the National Academy of Sciences*, **117**(42), 25966–25974. doi.org/10.1073/pnas.1910416117

Mikolov, T., Yih, W.T., Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 746-751. Association for Computational Linguistics.

Mitchell, M. (2021). Abstraction and analogy-making in artificial intelligence. *Ann N Y Acad Sci.* Dec;**1505**(1):79-101.

Moser, M. B., Rowland, D. C., & Moser, E. I. (2015). Place cells, grid cells, and memory. *Cold Spring Harbor perspectives in biology*, **7**(2), a021808. https://doi.org/10.1101/cshperspect.a021808

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

O'Keefe, J., Krupic, J. (2021). Do hippocampal pyramidal cells respond to nonspatial stimuli? *Physiological reviews*, **101**(3), 1427–1456. doi.org/10.1152/physrev.00014.2020

Neisser, U. (1993). Without perception, there is no knowledge: Implications for artificial intelligence. *Natural and Artificial Minds*, 174–164.

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

Penn, D.C., Holyoak, K.J, Povinelli, D.J. (2008). Darwin's mistake: explaining the discontinuity between human and nonhuman minds. *Behav Brain Sci.* **31**(2):109-30; discussion 130-178. doi: 10.1017/S0140525X08003543

Pezzulo, G., Barsalou, L. W., Cangelosi, A., Fischer, M. H., McRae, K., & Spivey, M. (2013). Computational Grounded Cognition: A new alliance between grounded cognition and computational modeling. *Frontiers in Psychology*, **3**. doi.org/10.3389/fpsyg.2012.00612

Pólya, G. (1954). *Mathematics and Plausible Reasoning Volume I: Induction and Analogy in Mathematics*. Princeton University Press.

Prade, H., Richard, G. (2011). Analogy-Making for Solving IQ Tests: A Logical View. In: Ram, A., Wiratunga, N. (eds) Case-Based Reasoning Research and Development. ICCBR 2011. Lecture Notes in Computer Science(), vol 6880. Springer, Berlin, Heidelberg. doi.org/10.1007/978-3-642-23291-6_19

Reber, T.P., Luechinger, R., Boesiger, P., Henke, K. (2014). Detecting analogies unconsciously. Front Behav Neurosci. **22**;8:9. doi: 10.3389/fnbeh.2014.00009.

Reinboth, T., Farkaš, I. (2022). Ultimate Grounding of Abstract Concepts: A Graded Account. *Journal of Cognition*. **5**(10):21. doi.org/10.5334/joc.214

Revonsuo A. (1999). Binding and the phenomenal unity of consciousness. *Consciousness and Cognition* 8(2):173-85. doi: 10.1006/ccog.1999.0384

Rodríguez, F., Quintero, B., Amores, L., Madrid, D., Salas-Peña, C., & Salas, C. (2021). Spatial Cognition in Teleost Fish: Strategies and Mechanisms. *Animals : an open access journal from MDPI*, **11**(8), 2271. doi.org/10.3390/ani11082271

Roy, N., Posner, I., Barfoot, T., Beaudoin, P., Bengio, Y., Bohg, J., ... & Van de Panne, M. (2021). From Machine Learning to Robotics: Challenges and Opportunities for Embodied Intelligence. *arXiv:2110.15245*.

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

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.

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

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

Smith, L., Gasser, M. (2005). The Development of Embodied Cognition: Six Lessons from Babies. *Artificial Life*, **11**(1–2), 13–29. doi.org/10.1162/1064546053278973

Sugar, J., & Moser, M. B. (2019). Episodic memory: Neuronal codes for what, where, and when. *Hippocampus*, **29**(12), 1190–1205. doi.org/10.1002/hipo.23132

Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., & Roy, N. (2011). Approaching the Symbol Grounding Problem with Probabilistic Graphical Models. *AI Magazine*, *32*(4), 64-76. doi.org/10.1609/aimag.v32i4.2384

Vendetti, M.S., Bunge, S.A. (2014). Evolutionary and developmental changes in the lateral frontoparietal network: a little goes a long way for higher-level cognition. *Neuron*, **84**(5), 906–917. doi: 10.1016/j.neuron.2014.09.035

Wernle, T., Waaga, T., Mørreaunet, M., Treves, A., Moser, M. B., & Moser, E. I. (2018). Integration of grid maps in merged environments. *Nature neuroscience*, **21**(1), 92–101. doi.org/10.1038/s41593-017-0036-6

Winograd, T. (1970). Procedures as a representation for data in a computer program for understanding natural language. Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA.

Wu, Y., Dong, H., Grosse, R., Ba, J. (2020). The Scattering Compositional Learner: Discovering Objects, Attributes, Relationships in Analogical Reasoning. In: *arXiv*:2007.04212.