A Structural Simulation Framework for Cognitive Systems

1st Teemu Saukkio*
dept. of computing
University of Turku
Turku, Finland
tjsauk@utu.fi
*Corresponding author

2nd Juha Plosila
dept. of computing
University of Turku
Turku, Finland
juplos@utu.fi

3nd Hashem Haghbayan
dept. of computing
University of Turku
Turku, Finland
mohhag@utu.fi

***Abstract*—This paper proposes a structural simulation environment for cognitive systems that effectively models both bottom-up emergent needs derived from sensory data and top-down task specifications for actuator control. The proposed simulation structure is both scalable and explainable; needs emerge concurrently. The tasks triggered to fulfil these needs follow a hierarchical top-down approach. Experimental results demonstrate that the simulation environment accurately models the *hunger* and *play needs* of a fully autonomous pet robot, all within the overall goal of the robot's survival.**

***Keywords*—Cognitive systems, Autonomous robot, Structural Simulation, Bottom-up Emergence, Pet Robot Simulation, Task Hierarchy, Needs-driven Behavior**

I. Introduction

Cognitive architectures and simulation environments are crucial parts of research in autonomous systems and robotics AI [8]. Various architectures have been developed to model cognition, rooted in the foundational work of Allen Newell, particularly his hypotheses regarding problem-solving methods in the human mind [11]. The proposed architectures and simulation environments primarily focus on classifying problem spaces, referred to as *task environments* in Newell's terminology [11], building knowledge structures, such as knowledge graphs or first/second-order logic, and employing reasoning based on these structures [6].

One challenge in developing such simulation environments is incorporating the advanced aspects of cognitive systems, including *intrinsic motivation* and *emotional dimensions* [2], [3], [5], [7], [9], [10], [12]. However, many of these approaches lack scalability and modularity, falling short in providing a structured model for the hierarchical functionalities of the cognitive mind. The vast expansion of knowledge within such architectures poses practical challenges, making it difficult to simulate these models with limited computational resources, such as those available in pet robots [4]. Another significant issue arises from the nature of *top-down* requirement specifications that often conflict with the *bottom-up* constraints present in developed methodologies. Ignoring the role of language in shaping developmental rules for biological cognitive systems, such as those involved in educational activities for humans and animals [1], it can be acknowledged that both goal definition and knowledge construction are fundamentally bottom-up developmental processes, e.g., the concept of goal-based concept in pp.90 of [1]. Overall, in cognitive models, there is no programmer to define desires, which presents a significant challenge to the current state of the art in this field regarding the joint development of both goals and knowledge, as well as the tuning of their interconnections.

In this paper, we propose a modular and structural framework for simulating cognitive systems. The primary advantage of this model lies in its inherent structure that facilitates both bottom-up goal and knowledge creation, as well as top-down task specifications. We define each goal as an objective addressed through a bottom-up process, initiated by a *need*, which is then met through top-down hierarchical task allocation. This process generalizes skill specification for the robot in both scalable and explainable fashion. The current version of our platform illustrates the overall architecture, with a fixed ontology for goals and knowledge. We have simulated the proposed platform on a pet robot scenario, with the sense of smell as the main perceptual component with two *needs*: 1) hunger, which is addressed through searching, finding, and consuming food, and 2) play, which is satisfied through interaction with an object in a 2D environment.

Results show that the distribution of the pet robot's location, based on these two activities, indicates a dense presence near food sources and a sparse presence in other areas of the room. In the subsequent sections, we will provide further details about the different components of the framework and their individual functionalities.

II. Framework Components

The framework consists of four main components: *needs*, *processes*, *steps*, and *primitives*. A *need* serves as an independent trigger signal for a *process*. A *need* is the amplitude or value of a independent system. Each *need* activates a specific *process*, which compiles a structured sequence of *steps*, *steps* are modular operations that represent reusable, high-level behaviors or skills. These *steps* are recursively composed of *child steps* and *primitives*, where *primitives* are the most fundamental, indivisible functions in the control hierarchy.

While different *needs* may lead to distinct *processes*, all *processes* share a common set of *steps* and *primitives*, ensuring *reusability*, *adaptability*, and *clarity of composition*.

To improve reproducibility and support clearer understanding of the framework, the following chapters provide detailed definitions, structural descriptions, and examples.

A. Need

A *need* is defined as a signal that represents the amplitude of a pattern or measurement, derived from the agent's self-experience in a specific situation—for example, patterns shaped through the processing of sensory observation or the internal clock of the agent. The difference between the current amplitude of the signal and a predefined reference point triggers an algorithmic process aimed at resolving this discrepancy. Throughout this paper, the term *pattern* refers to any temporal or spatial configuration of elements forming a distinct state, as perceived by either an observer or the agent's internal system.

A *need* may sometimes arise from a combination of multiple signals originating from different aspects of the agent's internal or external state. The classification of a *need* involves extracting temporal and spatial patterns from these signals and analyzing their interrelations. This classification allows for the simulation of higher-level functions observed in biological cognitive systems, albeit through simplified outputs generated by independent and modular functions within the simulation used in this paper.

For instance, in the context of the paper, hunger is implemented as linear relationship to the agents movement with in the simulated environment and its interactions with the locations containing food. In a robotics implementation hunger can be represented as the battery's state of charge.

B. Process

A *process* is a dynamic mechanism that responds to a *need* by initiating and managing functionality aimed at addressing the amplitude of the triggering signal. It serves as an execution context that accumulates the required knowledge to fulfill the task, by creating, evolving, and identifying suitable *steps* that can produce the desired change in relation to the *need* signal.

The design of *processes* enables the agent to react modularly to a variety of needs, reusing existing components while adapting the structure of execution based on the context and feedback.

When a specific *need* arises, a corresponding *process* is activated, that is designed to produce a counteracting or compensatory effect. The *process* searches through internal memory to discover appropriate *steps* capable of influencing the triggering signal. Once triggered, the process maps out a structured sequence of *steps*, known as the *process path*, that is expected to consistently lead to resolution.

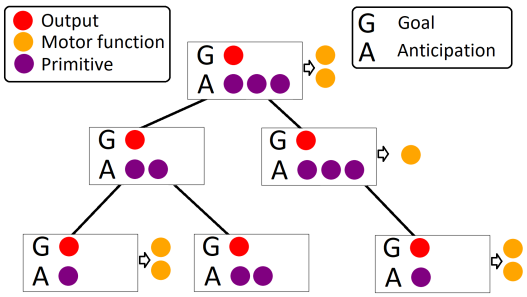

Fig. 1. Illustration of a arbitrary set of *steps*, that together form the sequentially progressing *process path* for this *process*, after each individual *goal* is reached.

This path is validated by checking whether the *anticipation* and *goal* values associated with each *step* are satisfied by the observed *outcomes* produced during execution. Through this matching mechanism, the process maintains coherence between its internal planning structure and the evolving external or internal state of the agent.

A *conflict* can occur, when an *outcome* does not match the expectation. This indicates a change in conditions and triggers an *adaptation* within the process, that can adjust the *steps* or add additional *steps* to the *process path*. This is done by isolating the parts causing the *conflict* and trying to resolve it by using the pre-existing knowledge or by acquiring new knowledge. A *process* can be simulated as a separate mechanism, or it can be incorporated into the implementation of the *steps*, in which case the *process* becomes just a term describing the act of executing the *process path*.

In Fig. 2, an example of how *adaptation* can be implemented within the *step* formalism is shown. $S2$ (step 2) in the illustration is formed by 2 *primitives* and a child *step* $S3$. *Adaptation* works in this instance through assigning the *outcome* of $p7$ (*primitive* 7) as *goal* value to the child *step*, leading to the change of the child *step* according to the *outcome* $o7$ from the previous function. This type of *adaptation* is demonstrated in the simulation section.

A *process* at a given state is defined by its *process path*, which dynamically evolves as a sequence of ordered *steps*. These *steps* represent the stages needed in the process, and reflects the current progress, adapting as the *need* changes. Using the block diagram representation of a *step*, a *process path* can be illustrated as shown in Fig. 1. In this depiction, the *process* begins execution from the bottom-left corner and progresses sequentially, moving left to right and bottom to top. Each *step* may be executed multiple times, and the process path advances only after the current *step's goal* is achieved.

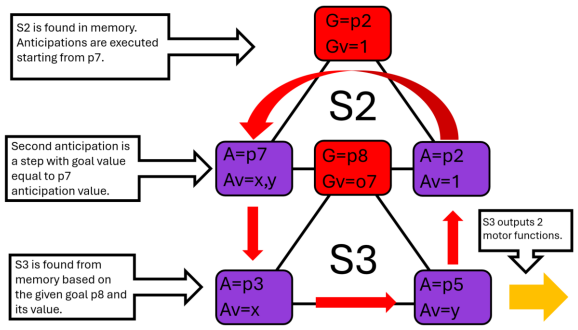

Fig. 2. Illustration of partial process path flow, of the hunger *need* used in the simulation. Illustration shows the flow with red arrows, *goals* within the red boxes, *primitives* within purple boxes, motor outputs as orange, and related details within the comment boxes.

C. Step

A *step* represents a higher-level skill, and can be constructed from a single *primitive*, a set of *primitives*, lower-level *steps* (referred to as child *steps*), or any combination thereof. The flexible composition of *steps* support modularity, reusability, and layered abstraction, enabling complex behaviors to be decomposed into traceable and explainable subroutines.

Each *step* is defined by two key parameters: *goal* and *anticipation*. Both parameters are each composed of one or more pairs containing a source and a value. Sources can be variables or functions like *steps* or *primitives* and the outcomes are expected values of the source at the moment of evaluation.

The *goal* defines the intended outcome or purpose of the *step*. It also establishes the condition for its successful completion, acting as a convergence point for causal and ontological relationships that can be inferred between the assigned *goal* and the underlying mechanisms—whether they are other *steps* or basic *primitives*—used to accomplish it. A *goal* is typically not directly attainable in isolation; its fulfillment often requires a sequence of intermediate actions, environmental interactions, or enabling conditions facilitated by subordinate components within the hierarchy.

Goal values may either naturally emerge (without explicit measurement), as is the case when a *need* signal is assigned as a *goal*, or require explicit measurement for verification. This enables *steps* to act as building blocks of knowledge. In addition to the case where a need is assigned as a *goal*, an example of a naturally emerging *goal* could be an assigned target point created through manipulation of a map through multiple different *primitives*. An explicitly measured *goal* would be, for example, a certain level of sensory reading that is measured in each iteration of the *step*, thus conducting a measurement of the progress level. A simple illustration of a *step* with an emerging *goal* is given in the form of $S3$ in Fig. 2, as well as a *step* with measured *goal* in the form of $S2$. This Figure also illustrates a partial process path for the hunger *need* used in the simulation section, where the functionality of each primitive (labeled as *p-identifiers*) is explained in more details later. The figure represents the flow of this process segment, where the system iterates through *anticipations* and *child steps* to ultimately reach the topmost *goal*.

In this illustration, the topmost *goal* is defined as $S2 = \{G, Gv\}$ where $G = p2$ represents the goal source or function, and $Gv = 1$ is the corresponding *goal* value. The process advances through *anticipations*, denoted as $\{A, Av\}$ witch includes the *child step* $S3$. This iterative process continues until the topmost *goal* is reached.

Anticipation is formed by sources and corresponding values similar to those in a *goal*. Each *anticipation* is a child *step* or *primitive* paired with a value that will either immediately or eventually cause the emergence of the *goal* value. The *anticipation* can be found through establishing consistent causality; that is, when a set of functions in an ordered sequence are executed to produce a sequence of *outcomes*, such as when these *outcomes* correspond to the *anticipation* values, the sequence will always directly or eventually lead into the *goal* value being met. Levels of the *anticipation* values, combined with the *goal*, can be used to create temporal knowledge by leveraging the amplitude of the *anticipation* values and the type of the *goal*. *Anticipation* values give out a system to track progress during the *process*, providing the mechanism needed to trigger adaptation in a dynamic environment when the *process* detects a deviation between the *anticipation* values and the *outcomes* of the functions, indicating that the *goal* is not reachable by the sequence in the current state.

D. Primitive

A *primitive* is defined as an indivisible, basic internal or external operation with the purpose of either (1) executing an action (e.g., moving an arm), or (2) enabling information gain (e.g., retrieving specific data from the sensory space). These *primitives* represent the most fundamental units of behavior or perception within the framework.

Primitives are essential for supporting the modularity required to structure complex behavior. They enable the execution of control functions, monitoring of progress, and the creation of temporal and spatial links between knowledge elements within the agent. This linking contributes to the emergence of new internal representations, effectively supporting the development of an ontological structure within the system.

The granularity of a *primitive* can vary depending on the level of abstraction used in control. At a higher abstraction level, a primitive might be defined as *"move hand"*, encompassing multiple motor actions. At a lower level of detail, it could be decomposed into *"contract the thenar muscle"*, reflecting a more biologically-inspired control resolution.

In Fig. 3, two types of *primitives* are illustrated; at the top of the image, a sensory *primitive* is used to take a sensory

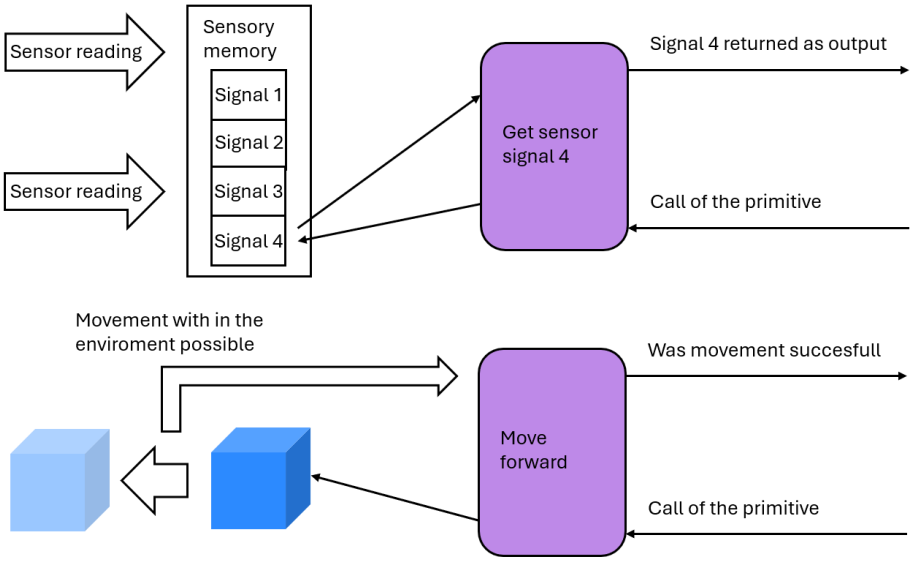

Fig. 3. Sensory *primitive* at the top part of the image, taking the sensory signal from sensory memory when called. Motor *primitive* at the bottom part of the image initiating movement within the environment, returning the confirmation if the initiated movement was successful.

signal from the independently updated sensory memory; this sensory *primitive* returns the current reading from memory as an *outcome*. At the bottom, a motor *primitive* initiates movement of the agent within the environment and returns an *outcome* indicating if the movement was successful. In a case where the agent is blocked by an obstacle the motor *primitive* would return a zero *outcome* indicating that movement was not successful.

III. Simulation

The simulation was conducted using a hardware configuration consisting of an Intel i7 (6-core) processor and 64GB of RAM. Simulation was implemented as discrete time with grid cell-based movement steps. We modelled two *needs* that are shown evolving for around 2000 time steps. The *Needs* are *hunger*, which simulates the energy consumption through actions, and *play*, which simulates the generalised time-dependent *need* for self stimulation. Both *needs* are connected to the agent's actions through linear increments. *Hunger* increases by 0.01 for each movement step the agent takes and decreases by 0.1 each time step the agent is next to a food position (agent is eating). *Play* increases on every time step by 0.05 when the agent is not interacting with the play object, thus simulating the effect of boredom and decreases by 0.1 each time step when agent is interacting with the play object (agent is playing).

Food objects are simulated as stationary locations that agent can interact with from any neighbour cell, play object was modeled as a singular moving grid cell that relocates randomly to any of its neighbour cells when agent is located in the same cell as the play object (agent interacts with play object). The two *needs* were simulated with hysteresis thresholds and with a binary prioritising policy. The *process* for *hunger* triggers when the *need* signal is over the value of 0.7 and switches off when the *hunger* signal is at 0, or if it is under 0.2 and the *play need* signal triggers. The *play need* signal triggers when the *play* signal is over 0.8 and the *hunger* signal is under 0.2.

Fig. 4 shows the evolution of *need* signal amplitudes and the trigger states of the *processes*. The threshold values are established based on an arbitrary indoor pet robot. However, the specific values for the threshold limits, time step, prioritisation policy, and constant movement step size (where the robot can move to any adjacent free cells, taking one step per time step) can be adjusted according to the physical and intellectual capabilities of the robot. This makes the proposed simulation environment a general framework capable of simulating a wide range of cognitive robots.

The pet robot the agent represents is equipped with a sense of smell, modelled as an environmental parameter, represented as a float value between 0 and 0.5. The agent is capable of distinguishing the strength of the smell in its surroundings and can select the neighbour with the strongest scent to locate food or play object and proprioceptive awareness (ability to sense movement) in a 2D space, allowing it to determine heading.

From these sensory modalities, a set of *primitives* is derived, each dedicated to specific experiences. In Tab. I, the set of *primitives*, *outcome* type, and experience used in the *processes* are shown. All *primitives* were implemented without arguments; thus, dedicated smell-based direction *primitives* for food and the toy are needed.

The *outcome* type in Tab. I specifies the measurement

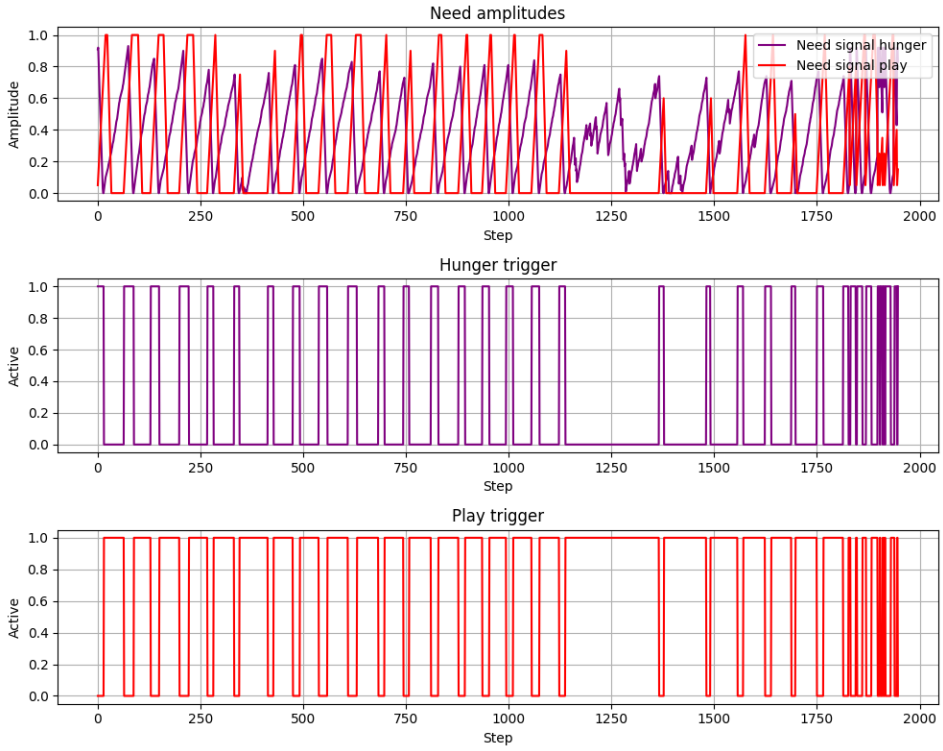

Fig. 4. Levels of *need* signals over the total amount of simulation steps are shown in the top part of the image; signal triggering eating (*hunger*) is shown with purple, and signal triggering *playing* is shown in red. In the middle part of the image, the triggering of the *process* for eating is shown with the *process* being active when the trigger is equal to one. In the bottom part, the triggering of the process for *playing* is shown.

from the *primitive*, as each primitive results in a measurable *outcome*, which, depending on the *primitive*, can be of different types, as was explained in the section about *primitives*. The *outcome* types for the 9 *primitives* that have been shown in Tab. I are as follows:

- *Amplitude* gives out the strength of the sense, which is a float number between 0 to 0.5.
- *Change* can be -1 for decrease, 0 for no change, and 1 for increase.
- *Result* type is 0 or 1 based on whether the *primitive* is executed successfully.
- *Vector* gives the x, y end coordinates for a heading vector, originating from agent's current position.

TABLE I. PRIMITIVES USED IN THE SIMULATION.

Symbol	Outcome	Experience
p1()	Amplitude	Food smell
p2()	Change	Change in p1
p3()	Result	Move to x
p4()	Result	Move to -x
p5()	Result	Move to y
p6()	Result	Move to -y
p7()	Vector	Food direction
p8()	Vector	Agent direction
p9()	Vector	Toy direction

Using this set of *primitives*, we can define the *steps* required for *Play* and *Eat* processes. As explained, the *process path* is a learnt skill, formed by the *process* specifying which sequence of *steps* leads into the desired change in *need*. Two distinct *process paths* in our experiment are as follows:

The *process path* for hunger *need*:

- S0 Goal(*hunger* need to 0) Anticipation(p1 outcome is 0.5)
- S1 Goal(p1 outcome is 0.5) follows eventually from Anticipation(p2 outcome is 1)
- S2 Goal(p2 outcome is 1) follows when Anticipation(p7 is anything, p8 matches p7 outcome, p2 outcome is 1). Where the anticipations "p8 matches p7 outcome" is the movement *step*.
- S3 Movement step Goal(p8 outcome is x,y) with the correct choice of primitives in Anticipation(p3 is x and p5 is y) exact composition of movement *step* changes according to the desired heading, thus creating *adaptation*.

The *process path* for play *need*:

- S4 Goal(play need to 0) eventually follows from Anticipation(p9 is anything, p8 matches p9 outcome) where "p8 matches p9 outcome" is the movement *step* (S3).

In Fig. 5 the resulting occurrence plot of agents activities is shown. Occurrence was plotted by area where each time the agent conducted an activity successfully in a location, the circle centred at that point grew in area; the movement of the agent when transferring from the play object to food or

back was excluded from the plot. As shown, the density of the robot's appearances is relatively higher in locations near the food, which are highlighted in purple. In contrast, due to the moving nature of the play object, the surrounding areas show a lower density of playing occurrences. As a direct consequence of the *processes*, the nature of the available objects and the agents senses, the distribution of occurrences focused in the bottom part of the available environment with relatively equal distribution of play occurrences on the area, as expected from randomly moving play object.

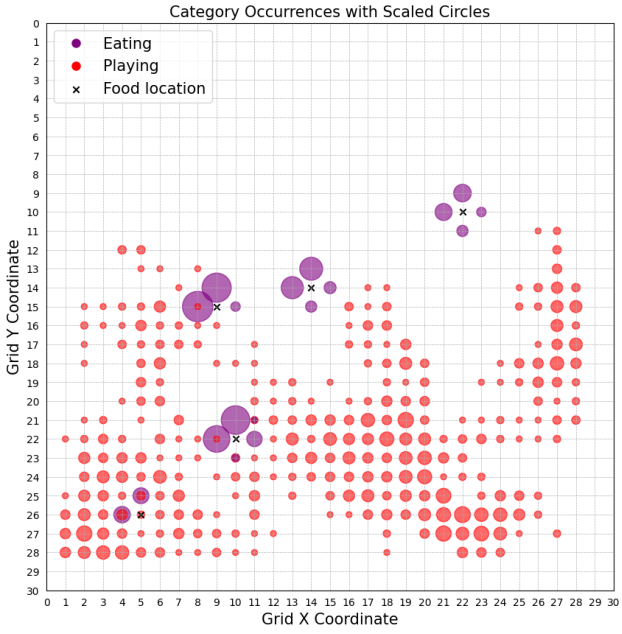

Fig. 5. The occurrences of activities (eating and playing). Occurrences are calculated by area; when the agent is eating, occurrences are shown in purple, and playing occurrences are shown in red. The fixed locations containing food are shown with a black cross.

IV. Conclusion

The results of this paper demonstrate that the proposed structural simulation framework can successfully generate intended behaviors in an artificial agent through a modular composition of cognitive components. Within the 2D simulation, the framework exhibited coherent execution of behavior based on emergent needs (such as hunger and play), fulfilling them through task decomposition of multimodal control paths. This outcome validates the framework's foundational properties: modularity, explainability, and reusability.

A key strength of this approach lies in its capacity to generalize concepts across different tasks. By representing all components in a structured form, the framework lays the groundwork for reapplying learned skills across contexts — a capability critical for future cognitive agents. However, this version of the framework intentionally omits learning mechanisms, not as a limitation of scope, but to isolate and demonstrate the structural viability and clarity of the system. Learning in unstructured or poorly abstracted architecture may obscure control flow and explainability, making this structural grounding a necessary precursor.

We acknowledge that the simulation was conducted in a simplified 2D environment and focused on relatively basic needs. These choices were intentional: they allowed us to test and expose the framework's structure without interference from complex learning dynamics. Future work will aim to expand this framework into more intricate and dynamic settings, involving overlapping and conflicting needs to evaluate its scalability and robustness under stress. This will also involve extending the primitives and steps for more broader contex.

Importantly, ongoing research explores integrating learning mechanisms into approach demonstrated in this structured framework, allowing agents to acquire and adapt knowledge while preserving explainability. Furthermore, we envision applying the framework to more physically embodied agents, enabling real-world validation and demonstrating the adaptability of reusable skills in uncertain environments.

Ultimately, this work provides a foundational contribution to the development of explainable, modular cognitive architectures and opens the path for intelligent agents capable of learning, reasoning, and adapting within a structured yet flexible cognitive scaffold.

References

[1] Lisa Feldman Barrett. *How Emotions Are Made: The Secret Life of the Brain*. Houghton Mifflin Harcourt, 2017. Kindle Edition.

[2] L. Da Costa, P. Lanillos, N. Sajid, K. Friston, and S. Khan. How active inference could help revolutionise robotics. *Entropy*, 24(3):361, 2022.

[3] N. Duminy, S. M. Nguyen, and D. Duhaut. Learning a set of interrelated tasks by using a succession of motor policies for a socially guided intrinsically motivated learner. *Frontiers in Neurorobotics*, 12:87, 2019.

[4] Marta Díaz-Boladeras. Bond formation with pet-robots: An integrative approach. *Current Psychology*, 42(4):2591–2608, 2023.

[5] S. Ivaldi et al. Learning to recognize objects through curiosity-driven manipulation with the icub humanoid robot. In *Proc. IEEE 3rd Joint Int. Conf. Develop. Learn. Epigen. Robot. (ICDL)*, pages 1–8, 2013.

[6] Pat Langley, John Laird, and Seth Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10:141–160, 06 2009.

[7] Pablo Lanillos et al. Active inference in robotics and artificial agents: Survey and challenges. *ArXiv*, abs/2112.01871, 2021. n. pag.

[8] Antonio Lieto, Mehul Bhatt, Alessandro Oltramari, and D. Vernon. The role of cognitive architectures in general artificial intelligence. *Cognitive Systems Research*, 48, 09 2017.

[9] K. Man and A. Damasio. Homeostasis and soft robotics in the design of feeling machines. *Nature Machine Intelligence*, 1:446–452, 2019.

[10] A. Manoury, S. M. Nguyen, and C. Buche. Hierarchical affordance discovery using intrinsic motivation. In *Proc. 7th Int. Conf. Human Agent Interact.*, pages 186–193, 2019.

[11] Allen Newell, J. Shaw, and Herbert Simon. The elements of a theory of human problem solving. *Psychological Review - PSYCHOL REV*, 65:151–166, 05 1958.

[12] Naoto Yoshida, Tatsuya Daikoku, Yukie Nagai, and Yasuo Kuniyoshi. Emergence of integrated behaviors through direct optimization for homeostasis. *Neural Networks*, 177, 2024.