Schema Mechanisms 2.0 for Developmental Artificial Intelligence

Olivier L. Georgeon[1][0000−0003−4883−8702],
Filipo S. Perotto[2][0000−0003−2283−4703], Kristinn R. Thórisson[3],
Arash Sheikhlar[3], and Paul Robertson[4][0000−0002−4477−0379]

[1] UR CONFLUENCE: Sciences et Humanites (EA 1598), UCLy, France
ogeorgeon@univ-catholyon.fr
[2] ONERA – The French Aerospace Lab – DTIS, Toulouse, France
filipo.perotto@onera.fr
[3] Center for Analysis and Design of Intelligent Agents, Reykjavik University, Iceland
thorisson@ru.is, arash@iiim.is
[4] DOLL Labs, Lexington, MA, USA
paulr@dollabs.com

Abstract. Schema mechanisms are software frameworks designed to reflect learning theories like genetic epistemology, constructivist epistemology, and developmental learning. Building on Gary Drescher's pioneering implementation from 1991, this paper reviews four contemporary schema mechanisms: CALM, LIDA, ECA, and AERA. While maintaining Drescher's Piagetian-inspired view that sensorimotor schemas serve as the foundation for knowledge construction, these new mechanisms extend the constructivist approach by grounding schemas in interactional events rather than static world perceptions. This shift introduces an interactional motivation framework and integrates hierarchical abstraction, internal simulation, and physical space into schemas, paving the way for their application in autonomous robotics operating in open environments. We believe that these advancements mark the emergence of a new generation of schema mechanisms, which we refer to as Schema Mechanisms 2.0.

Keywords: schema mechanism · genetic epistemology · constructivist learning · intrinsic motivation · cognitive architectures.

1 Introduction

Throughout his life, Jean Piaget developed and popularized the theory of *genetic epistemology* [38] to account for the genesis of intelligence and knowledge. In parallel, he pioneered *developmental psychology* by establishing the foundational methods for studying mental development in children [37, 36]. Toward the end of his life, he connected genetic epistemology with constructivist epistemology and the work of Ernst von Glasersfeld [18]. Genetic epistemology rests on the key concept of "schema", closely related to the concept of "functional circles" proposed by Jakob von Uexküll. We refer to Ziemke [53] for a broader discussion

on the philosophical roots of constructivist epistemology in Kant's philosophy, and Uexküll's biology. Piaget defines a schema as follows:

> "A schema is a structure or organization of actions as they are transferred or generalized by repetition in similar or analogous circumstances." ([35], p. 4).

A schema is the basic unit of knowledge that encapsulates the action and its circumstances, that is, a *pattern of interaction*. Genetic epistemology insists on the primacy of interaction as a condition for the emergence of perception and knowledge, surpassing innatist and empiricist explanations:

> "Knowledge does not originally arise either from a subject conscious of itself or from objects already constituted (from the subject's point of view) that would impose themselves on the subject. Knowledge results from interactions occurring halfway between the subject and the objects, and thus involving both, but due to a complete un-differentiation and not from exchanges between distinct forms.
>
> If, at the beginning, there is neither a subject, in the epistemic sense of the term, nor objects, conceived as such, nor, above all, invariant instruments of exchange, then the initial problem of knowledge will be to construct such mediators. Starting from the contact zone between one's own body and the objects, these mediators will progressively engage more deeply in both complementary directions toward the exterior and the interior. It is from this dual progressive construction that the joint elaboration of both the subject and the objects depends.
>
> The initial instrument of exchange is not perception, as rationalists too easily conceded to empiricism, but rather action itself, with its much greater plasticity. Certainly, perceptions play an essential role, but they partly depend on action as a whole, and some perceptual mechanisms that one might have thought to be innate or very primitive only emerge at a certain level of object construction." (translated from [39], p. 14-15)

For Piaget, the intelligence is related to the ability to gradually organize and complexify the set of schemas. That increasing organization of internal structures are guided by two forces: by assimilating new experiences to existing schemas, the mind tries to integrate observations and interactions in terms of already constructed cognitive structures, extending or generalizing their scope, and by accommodating its set of schemas to surprising or disruptive experiences, differentiating, specializing, complexifying its internal relations, the mind creates new intellectual resources for better understanding and mastering its interactions:

"[...] Organization is inseparable from adaptation: they are two complementary processes of a single mechanism, organization being the internal aspect of the cycle of which adaptation constitutes the external aspect. [...] that adaptation is an equilibrium between assimilation and accommodation. [...] Intelligence is assimilation to the extent that it incorporates all the given data of experience within its framework. [...] That mental life is also accommodation to the environment. Assimilation can never be pure because by incorporating new elements into its earlier schemata the intelligence constantly modifies the latter in order to adjust them to new elements." ([37], p. 7)

"In their initial directions, assimilation and accommodation are obviously opposed to one another, since assimilation is conservative and tends to subordinate the environment to the organism as it is, whereas accommodation is the source of changes and bends the organism to the successive constraints of the environment. But if in their rudiment these two functions are antagonistic, it is precisely the role of mental life in general and of intelligence in particular to intercoordinate them." ([36], p. 352)

"Gradually, as the child's thought evolves, assimilation and accommodation are differentiated and become increasingly complementary. In the realm of representation of the world this means, on the one hand, that accommodation, instead of remaining on the surface of experience, penetrates it more and more deeply, that is, under the chaos of appearances it seeks regularities and becomes capable of real experimentations to establish them. On the other hand, assimilation, instead of reducing phenomena to the concepts inspired by personal activity, incorporates them in the system of relationships rising from the more profound activity of intelligence itself." ([36], p. 385)

"The study of sensorimotor or practical intelligence in the first two years of development has taught us how the child, at first directly assimilating the external environment to his own activity, later, in order to extend this assimilation, forms an increasing number of schemata which are both more mobile and better able to intercoordinate. [...] The increasing coherence of the schemata thus parallels the formation of a world of objects and spatial relationships, in short, the elaboration of a solid and permanent universe." ([36], p.*xi*)

Schema mechanisms attempt to reflect the key insights of genetic epistemology effectively expressed by these words of Piaget regarding the primary role of sensorimotor experience and organization of mental structures. Indeed, genetic epistemology appears to provide a materialistic explanation of how the human

mind develops from birth, potentially lending itself to computational implementation.

Over the past decades, Piaget's theories have remained central to contemporary theories of knowledge development, continually questioned and shaped by active debates. Some shortcomings have been identified, including an underestimation of infants' abilities and an insufficient consideration for cultural and social factors (e.g., [1]). The relationship between constructivist and nativist theories remains a topic of debate, raising questions about the innate knowledge and prerequisites that should be incorporated into any computational implementation [5, p. 41]. The ongoing research on schema mechanisms examined in this paper contributes to these debates.

2 Schema mechanisms

Implementing genetic epistemology as a mechanistic process involves reducing Piaget's psychological concept of schema into data structures handled by a computer. This computer controls an artifact (robot or virtual agent) that interacts with its environment. The computer must process these data structures through some mechanism that will support the generation of increasingly intelligent behavior demonstrating the internal development of the robot. Guerin and McKenzie [19] proposed the graphical representation of this developmental process shown in the Figure 1.

Notably, the developmental process is not targeted at reaching a predefined goal-state in a predefined problem space, nor does it aim at maximizing a predefined scalar value as in reinforcement learning. Instead, the problem of cognitive development in artificial systems is intertwined with the problem *intrinsic motivation*, as has been argued for example by Oudeyer and colleagues [28].

2.1 Drescher's Schema Mechanism

In the early 1990s, Gary Drescher pioneered the first schema mechanism as a result of his PhD thesis [4] and associated publications [3, 6], which gave rise to a successful book [5], proposing a framework to reproduce key constructivist notions in artificial intelligence. Drescher formalized schemas as the tuple $\langle context, action, result \rangle$ depicted in Fig. 2. The *context* and the *result* are sets of binary *items* that can be *positively* or *negatively* included. A schema's context is satisfied when all the positively included items are *On* and all the negatively included items are *Off*. A schema's result asserts that, if the *action* is taken when the context is satisfied, then the positively included items of the results will be turned *On* and its negatively included items will be turned *Off*.

The agent is initialized with a set of *primitive* items and actions that represent builtin elementary sensorimotor variables. For each primitive action, a *bare schema* with empty context and result is created. The designer may also add other initial schemas that encode the agent's "innate" behaviors.

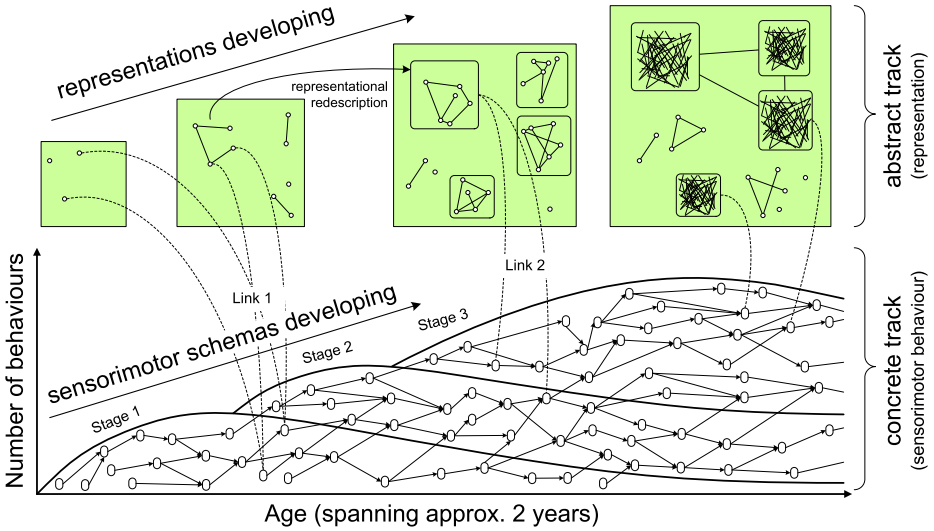


Fig. 1: Conceptual diagram of infant development from [19] Fig. 1. The lower (concrete) track shows a directed acyclic graph of sensorimotor schemas. A node represents a newly created schema. An edge has the meaning "is a necessary precursor". Stage 1: behaviors without objects. Stage 2: behaviors with single objects. Stage 3: object-object behaviors. The schemas now involve relationships among objects, and locations and transforms within space. The higher (abstract) track represents representations of objects by schemas and physical properties influencing their interactions.

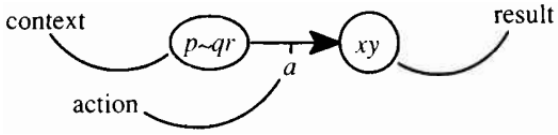


Fig. 2: Schema from [5] Fig. 3.2. The schema noted $p \sim qr/a/xy$ asserts that if action a is taken in the context where item p is On, q is Off, and r is On then the items x and y will be turned On.

The *activation* of a schema consists of initiating its action. Upon completion of the action, the predicted result is compared with the actual result. The schema is said to *succeed* if its predicted results are all obtained, and to *fail* otherwise. The ratio of success and failure is called *reliability* and is memorized for each schema.

The learning of new schemas is triggered by the identification of *relevant results* made of items that are slightly more frequent than average. When a relevant result is identified, the agent seeks conditions under which it follows reliably. Such conditions are found through new schemas that allow recovering the context leading to the relevant result, and then performing the action again to asses the reliability of these new schemas.

After learning new schemas, the agent can learn new "concepts" when it identifies schemas that act as "probes" that may evoke the manifestation of a "thing". These schemas are only reliable if the "thing" is present. When such a schema is found, it is considered a *host schema* for a newly spawned *synthetic item* that represents this thing. Eventually, the agent finds out that a single "thing" may manifest itself through different host schemas. Drescher notes that "a synthetic item thus works backward from a thing's manifestation to define the very thing manifested" [5, p. 83]. This addresses the fundamental problem of *concept invention* in that "a synthetic item is a new element of the system's ontology—an element fundamentally different from the prior contents of the system's conceptual vocabulary" [5, p. 81].

The designer defines the agent's goals by assigning *primitive values* (positive if desirable, negative if dreaded) to values of items (*On* or *Off*). The mechanism can identify subgoals as specific sets of items to be activated or deactivated; and it constructs *composite actions* as sequences of schemas used to reach a subgoal. The values of a target item may be propagated to other items in the form of *instrumental values* and *delegated values* when attaining these other items improves the chances to reach the target item. The agent randomly alternates between explicit goal-pursuit and exploration, with probability defined by the designer.

Drescher demonstrates its schema mechanism in the "microworld" made of the two-dimensional grid shown in Fig. 3. "The mechanism controls a simulated robot that has a body, a single hand, and a visual system. The hand can touch and grasp objects, and move them about. The visual system maps a visual field onto a region of the world in the immediate vicinity of the robot body" [5, p. 114].

The mechanism initially ignores the topological relations among sensory items, but the demonstration shows that it gradually learns these relations in a process that Drescher calls the "elaboration of the spatial substrate". Such learning of a cognitive map from sensorimotor sequences has recently been further investigated by Raju et al. [40].

Once the spatial substrate is learned, the mechanism manages to construct synthetic items that designate objects as distinct from their current perceptions. Drescher gives the example of "a synthetic item, which we might call

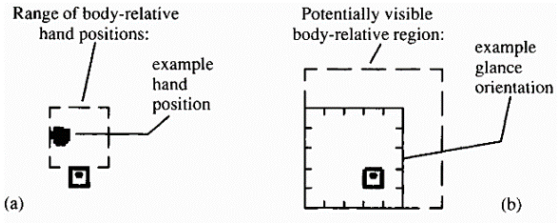


Fig. 3: Drescher's benchmark from [5, Fig. 6.1]. The simulated robot's body is represented by the small square with a black dot. (a): The hand (black spot) can move in the 3×3 grid in front of the body. (b): The visual field is the 5×5 grid in the environment made of a 7×7 grid.

`PalpableObjectAt(1,3)` (the name, of course, has no meaning to the mechanism). The item is defined to represent whatever condition makes the schema reliable; we observers know (but the mechanism does not) that this condition is that there be an object at (1, 3)" [5, p. 13]. Synthetic item `PalpableObjectAt(1,3)` may later be associated with `VisibleObjectAt(1,3)` to account for the discovery that they refer to the same thing at position (1, 3) in the world.

In summary, Drescher's schema mechanism stands among the first attempts to investigate Piaget's question of how the child's mind can dynamically construct its own knowledge of reality from experience of interaction. In our view, Drescher's main contribution to this question rests on the construction of synthetic items to account for "things in the world" that the agent can discover through regularities in sequences of sensorimotor experience. Drescher goes beyond his predecessors (e.g., [20]) who merely categorized passively-received inputs. His schema mechanism was the first that "grounds its synthetic items in the reification of counterfactual assertions" [5, p. 90].

Drescher identified some limitations that may explain why nobody has made his mechanism work in a more complex environment since its creation almost 40 years ago. These limitations follow from the acknowledgment that "whatever experiences an organism or mechanism has had, there are infinitely many mutually contradictory generalizations that are consistent with those experiences. Therefore, any induction apparatus must impose a choice among those consistent generalizations" [5, p. 174].

Due to specific design choices, Drescher's schema mechanism may lack generality. For example the assumption that a sensory item "is a state element" and "each item represents some condition in the world" [5, p. 56] presupposes that the world is defined as a predefined set of states, which is acceptable in a microworld but difficult in the physical world. Another assumption was to model the agent's motivation through positive and negative values associated with sensory items. The exploration mechanism based on hysteresis and habituation may also lack generality; other motivational principles can be imagined. Conversely, a schema mechanism may need additional built-in assumptions to cope with the complexity of the physical world. This paper reviews other schema mechanisms

that make other choices in the hope to move forward towards a general schema mechanism that could work in the real world.

2.2 Constructivist Anticipatory Learning Mechanism (CALM)

In the early 2010s, Filipo S. Perotto proposed his own version of a schema mechanism, called *Constructivist Anticipatory Learning Mechanism* (CALM), as a result of his PhD thesis [29, 30] and related publications [31–34]. Perotto tackled the same challenges as Dresher, namely: learning a model of the regularities observed by an agent while interacting with the environment, and extending the representational vocabulary to allow the agent to overpass the raw sensorimotor perception by the creation of more abstract concepts. However, while Dresher's mechanism is closer to the 1980s symbolic planning languages and structures, Perotto's background lied on reinforcement learning [47] and multiagent systems. In this way, CALM can be seen as a kind of model-based RL algorithm, trying to learn a model of the transitions (i.e. how the observations change depending on the context and actions), then, at each time step, computing a policy of actions that maximizes the expected discounted sum of rewards, similar to adaptive dynamic programming methods. Like Drescher's schema mechanism, the elementary knowledge structures in CALM are schemas, illustrated in the Figure 4, represented by tuples associating context, action, and expected result.

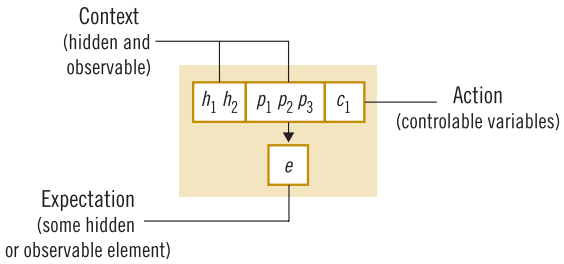


Fig. 4: Perotto's schema from [31] Fig. 2. A schema is composed of three vectors: context containing hidden variables and observable properties (h_1, h_2, p_1, p_2, p_3), action (c_1), and expected result (e).

The main drawback is that CALM can only identify regularities if they are deterministic, differently from the original schema mechanism, which can maintain unreliable regularities within the system in order to spinoff reliable schemas afterwards. To compensate that limitation, CALM introduces a new distinction between two kinds of context variables: hidden variables and observable variables as shown in the Fig. 4. The justification is that some regularities appear as non-deterministic because the agent cannot perceive all the contextual elements necessary to precisely determine its transition function. In this way, the accent is placed on the discovering of abstract or non-observable elements, that once

introduced and integrated in the anticipatory system, are able to explain the observation, making the transition deterministic. The idea is, in fact, quite similar to what Dresher called synthetic items in the original mechanism, but the way to create them is slightly different: when an expected result is contradicted, and no elements from context or action can explain the incoherence between the prediction and the observed result, then the mechanism supposes the existence of an abstract variable that, if known, would be able do distinguish between the two situations. The observed result then serves as a mean to deduce the value of that abstract variable, i.e. when the result corresponds to the old expectation, it is because the non-observable condition was present, and if the result is the new one, then the non-observable condition was absent, which corresponds to the state of the abstract variable perceived a posteriori. Finally, by searching in the previous step, the mechanism can try to anticipate the contexts and actions that changes that abstract variable, like illustrated in the Figure 5.

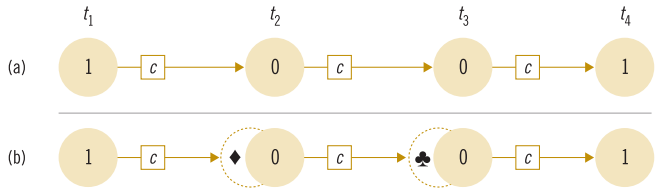


Fig. 5: Predicting the dynamics of a non-observable property. (a): An actually experienced sequence including $0 + c \rightarrow 0$ and $0 + c \rightarrow 1$. (b): The use of a synthetic element to explain the logic behind the observed transformations: $\blacklozenge 0 + c \rightarrow 0$, $\clubsuit 0 + c \rightarrow 1$, $1 + c \rightarrow \blacklozenge$, and $0 + c \rightarrow \clubsuit$.

For dealing with the combinatorial complexity, at the beginning, the only variables that CALM tries to understand and anticipate are those associated to affective values (what Drescher called primitive values). It is like a built-in reward function that associate positive (desirable) or negative (undesirable) values that the agent can "feel" immediately depending on the value of the variable (i.e. its observation, e.g. true or false if boolean). Staying focused on these variables avoids wasting energy by creating models about everything. Gradually, the variables needed to anticipate the evolution of some important variable (relation of causality) are then considered also important, and the mechanism will seek to model their transformation function too.

Differently from the original mechanism, which creates one initial schema for each action with empty context and result, CALM creates one initial schema for each item on the result vector (which corresponds to a perceptive variable) that has a non neutral affective value, with empty context and action. Each one of these initial schemas will grow up, giving rise to an anticipatory tree, responsible for identifying the relevant elements and the rules that describe the transition of its associated item, allowing in this way to anticipate it. Each anticipatory

tree is constructed step by step using 4 operations: *differentiation*, *adjustment*, *integration*, and *abstraction*. Each node in the anticipatory tree is identified by a vector containing observations and actions. Starting from a completely generic root node, where all elements on that vector are set as *undefined*, the agent tries to verify if some regularity persists from the current time-step to the next. The idea is defining a structure that is coherent with the past experience. The expectation of that first schema will be equivalent to the first observation. In this way the schema tries to represent as many transitions as they can be predicted deterministically with the minimum of context (i.e. the minimum of dependencies).

When an observed result contradicts what was expected, the tree is expanded by differentiation, like illustrated in Fig. 6: the contradicted generic schema gives rise to two different children, specializing either one element of context or one element of the action vector, in order to explain the surprise, making the model coherent again. When no primitive action or primitive context element can be used to differentiate a contradicted schema, the mechanism creates a synthetic element of context, representing some abstract or non-observable variable, that, if exists, can explain the incoherence. The schema is then differentiated by this new element. Sometimes, it is not possible to differentiate the schema anymore, in this case, an adjustment is done, setting the contradicted expectation to undefined. Finally, when several schemas are adjusted, some differentiations become unnecessary, then the tree can be pruned, and some leaf schemas integrated in a more general one. In this way, the schemas can be organized in a structure called *anticipatory tree*, which defines a hierarchy, like a decision tree, starting from a completely general context and action at the root node, to most specialized ones, the leaf nodes.

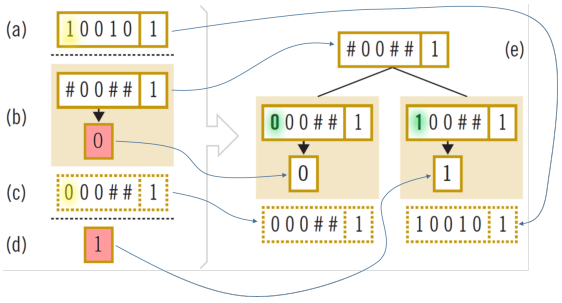


Fig. 6: Differentiation example. (a): An actually experienced situation (with five variables) and executed action (one variable). (b): Activated schema (with compatible context, action, and expectation). (c): Associated episodic memory (representation of actual situations where the scheme has been activated, with no interdependencies between variables). (d): Actually observed result after the execution of the action. (e): Sub-tree generated by differentiation in order to compensate the divergence observed between expectation and result.

To be able to make differentiations, beyond the context and action vectors that identify the schema, the mechanism must store a kind of memory of the effective experienced contexts and actions observed during the past activations of the schema. The identity of the schema corresponding to a maximally generic situation, and the memory corresponding to a maximally specific situation. In order to create a new schema by differentiation, the mechanism search for an item or action in the memory of the schema that is different from the current situation that is causing the incoherence. The differentiation will preserve the past experience in the old schema, identifying it with a more specialized context or more precise action, separating the new situation in another schema. In terms of computational complexity, the problem is how to manage that memory, because it needs to remember co-occurrences of items and actions. The solution given by Perotto is to explicitly limit the number of variables to observe together, starting from 1, and gradually increasing that number, which in any case will always be low, at the order of few units. The original mechanism proposed by Drescher faces the same problem, and is limited to observe only one additional variable at time.

Fig. 7 shows the simple flip problem [45], used to demonstrate CALM's capability to create an abstract variable for representing a relevant non-observable condition of the environment [34]. The set of possible actions is {`l`, `r`, `u`}, standing for *go left*, *go right*, and *unchanged*. The set of possible sensory signals is {`0`, `1`}, corresponding to a single primitive item in Drescher's terms. The difficulty comes from the fact that the agent cannot directly perceive the underlying state of the world. The agent constructed the graph in Figure 8.

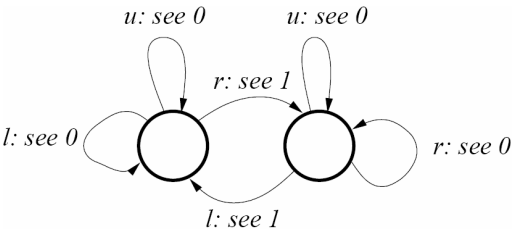


Fig. 7: The flip problem initially proposed by Singh et al. [45]. The `r` action taken from the left states activates the right state and vice versa, yielding sensory signal `1`. All other cases leave the state unchanged and yield sensory signal `0`.

In summary, CALM was used to control an agent that received only a binary sensory signal that does not directly represent the state of the environment. This contrasts with Drescher's experiment in which primitive items represent elements of the state of the world. CALM is able to reconstruct *non observable properties* from sequences of actions and feedback. Similar to Drescher's synthetic items, non observable properties describe the underlying state of the world that cause this feedback. Another addition to Dresher's mechanism is that CALM

organizes non observable properties (i.e., synthetic items) hierarchically in the anticipatory trees, which are finally mixed together, as shown in the Figure 8, which represents the final stable solution for the flip problem.

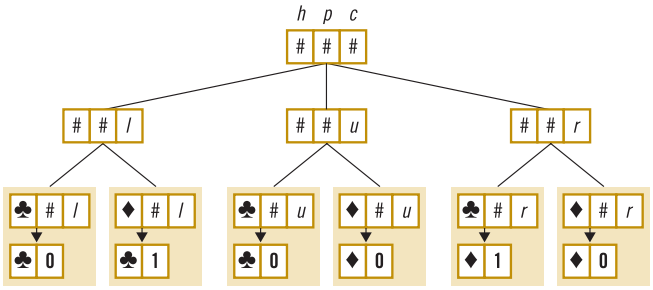


Fig. 8: Final schematic tree for deterministically describing the transition function corresponding to the flip problem [31]. Note that the underlying non-observable left and right states of the problem are represented by ♦ and ♣ values of the abstract sythetic variable. A *Markovian Decision Process* with 4 states ($\blacklozenge 0, \blacklozenge 1, \clubsuit 0, \clubsuit 1$) and a deterministic transition function can be derived from that tree if an immediate reward function is given (e.g. $R(0) = -1, R(1) = +1$), allowing to extract an optimal policy of actions using dynamic programming techniques.

2.3 Learning Intelligent Decision Agent (LIDA)

The *Learning Intelligent Decision Agent* (LIDA) is a broad cognitive architecture serving both as a conceptual and as a computational framework for modeling the mind [7, 8]. A recent version of the LIDA cognitive architecture [24] implements a schema mechanism inspired by Drescher on top of a perceptual memory module called the *Perceptual Associative Memory* (PAM). Percepts are not directly received from the environment but constructed in the PAM through an active sensorimotor process. Other memory modules are implemented including episodic memory, declarative memory, and spatial memory. Knowledge is represented as a directed graph called the *node structure*.

LIDA's schemas work as templates with unbound variables in their context, action, and result. Actions may have unbound variables that qualify and modulate how an action is executed. Schemas are instantiated as *behaviors*.

LIDA implements an *affective valence* which drives behavior selection towards desirable outcomes. Liking and disliking are distinguished from wanting and dreading with the argument that they relate to distinct neural pathways in the brain [22]. Wanting and dreading are qualified by *incentive salience* of behaviors.

Kugele [23] recently presented an improved version of LIDA into which the primitive items are replaced by *amodal nodes* organized in the node structure

in Perceptual Associative Memory. An important difference between LIDA's amodal nodes and Drescher's primitive items is that amodal nodes may not only represent sensory data but also events. Consequently, LIDA can not only represent a particular context in terms of elements of the world state, but also in terms of sensorimotor events.

Fig. 9 shows the cognitive cycle. The dashed rounded rectangle at the bottom shows the schema mechanism as a part of the architecture. The Global Workspace (right) stores a description of the current situation as a graph node. The Procedural Memory module (bottom right) is the memory of schemas. The graph node from Global Workspace activates relevant schemas. The Action Selection module selects an action, primitive or composite, from among the actions of the relevant schemas. Once the action is selected, it is instantiated as a *behavior*, or a *stream of behavior* for composite actions, and their unbound variables are qualified. The behavior is enacted in the Environment (left) and sensory stimuli (signals) are received by the Sensory Memory module (upper left). Sensory signals are processed to produce amodal nodes that represent the current situation. The Attention Codled module selects some amodal nodes to broadcast in the Global Workspace module.

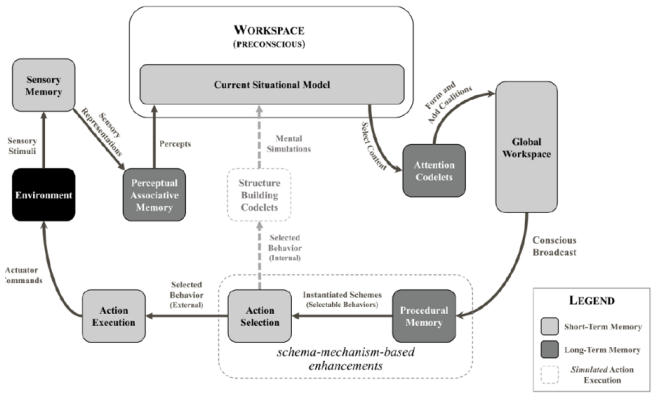


Fig. 9: Simplified depiction of the LIDA cognitive cycle [23, Fig. 4].

Kugele [23] demonstrated constructivist procedural learning within LIDA in a multi-armed bandit environment. Different versions of k-armed-bandit environments have been used previously, for example to demonstrate the theory of *active inference* [46], but Kugele added new difficulties by requiring the agent to sit and to make a deposit before playing. When the agent is standing (State S), it can take the action to sit at a one-armed bandit machine, for example Machine 1 (State M_1). The agent can then pay a deposit (state M_1P), and then play, which may yield a win (State M_1W) with probability $p(W|M_1P)$, or a loss (State M_1L) with probability $p(L|M_1P)$. The agent can choose to stand from

any state. Fig. 10 represents the state diagram of this environment in the case of two machines.

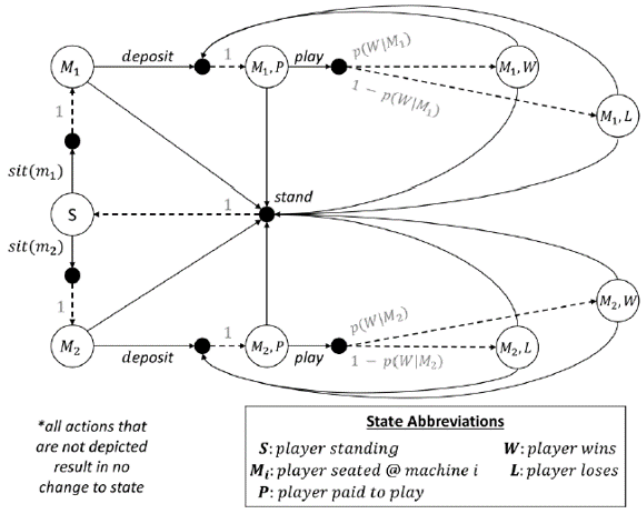


Fig. 10: State diagram of the 2-armed bandit environment [23, Fig. 5]. White nodes depict environmental states. Black nodes depict actions. Links going from white state nodes into black action nodes indicate that this action is taken from that state. Dashed links leaving action nodes are labeled with the probabilities of transitioning to possible states resulting from this action.

The set of possible actions for k machines is {`stand`, `sitM1`, ..., `sitMk`, `deposit`, `play`}. The set of initial nodes is {`Win`, `Lose`}, corresponding to a single primitive item in Drescher's terms. Node `Win` is associated with the affective valence (i.e., primitive value) of +1, and Node `Lose` of −1. As in the flip problem presented above, the agent cannot directly perceive the state of the environment. The results obtained in an 8-armed-bandit environment show that after 25,000 interaction cycles, the agent successfully constructed the amodal nodes (i.e., synthetic items) {`S`, `M1`, ..., `M8`, `M1P`, ..., `M8P`} that represent the states of standing or sitting at a machine without or with a paid deposit, along with the schemas representing the transition between states. Once this was learned, the agent determined and chose the machine that yielded the highest payoff.

In summary, LIDA allowed an agent that receives only a binary sensory signal (`Win` or `Lose`) to learn the structure of this environment, and to find adapted behaviors (sitting at the most favorable machine, and then continuing to deposit and to play) satisfying the motivation of maximizing the value won. This is similar to CALM but in a more complex environment.

The agent's adaptation to the k-armed bandit environment was made possible by LIDA's distinction between sensory signal and perceptual memory that allows

treating sensory signal as mere events rather than elements of the state. Notably, the agent did not learn the states `MiW` and `MiL` but rather the aggregated state `Mi` representing sitting at Machine i. This is because the sensory item (`W` or `L`) is not useful in itself to differentiate significant states from the agent's viewpoint.

Another difference is that LIDA treats composite schemas as procedural knowledge (stream of behavior) that are handled as a whole action by the Action Execution module, which facilitates the learning of the sequence `deposit–play`, before finding the machine that yields the highest payoff.

2.4 Enactive Cognitive Architecture (ECA)

Olivier Georgeon and colleagues also studied problems in which the agent receives little sensory information and must find behaviors that satisfy its predefined motivational criteria [16] . Similar to CALM and LIDA, Georgeon's schemas are based on events of interaction that do not necessarily represent elements of relevant states.

Formally, schemas are tuples ⟨pre-subschema, decision, post-subschema⟩ in which the pre-subschema and the post-subschema are other schemas. The mechanism is initialized with a set of *primitive schemas* that define the primitive possibilities of interaction of the agent with its environment. Primitive schemas are initialized with a predefined valence (equivalent to Drescher's primitive value and LIDA's affective valence) that specify the agent's "inborn" preferences of interaction.

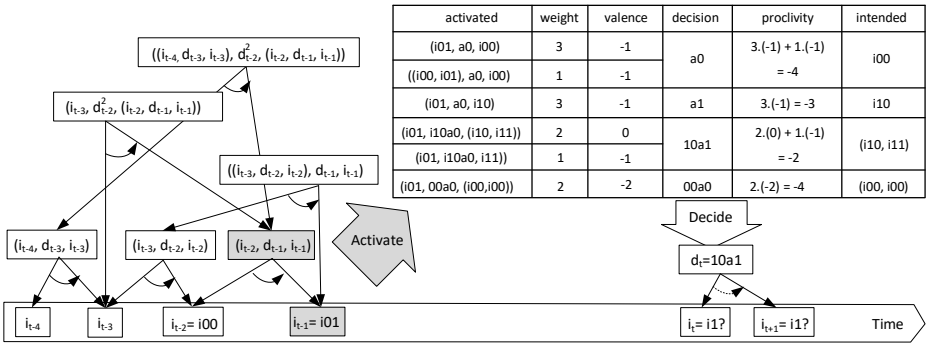
In the case of a simulated agent, a primitive schema is the association of a *primitive action*, for example `move-forward` or `turn`, with a *primitive outcome*, for example `bump` or `no-bump`. The primitive schema ⟨`move-forward`, `bump`⟩ initialized with a negative valence specifies an agent that "dislikes" bumping into walls when it tries to move forward.

The mechanism learns new schemas from the bottom up, with higher-level schemas made of a sequence of two previously-learned subschemas, as illustrated in Fig. 11. Schemas are noted with the letter i because they are sometimes called *interactions*. For example, if the primitive schema $i00$, made of action $a0$ and outcome $o0$, was enacted at time $t-2$, and the primitive schema $i11$, made of action $a1$ and outcome $o1$, was enacted at time $t-1$, then the composite schema $\langle i00, a1, i11\rangle$ is learned (added to memory) or its *weight* is incremented if it already existed. The weight of schemas plays a similar role as their reliability in Drescher's implementation.

When a new schema's post-subschema is primitive, the new schema's decision consists of performing the post-subschema's action. When the post-subschema is composite, the new schema's decision consists of enacting the whole sequence specified by the post-subschema. This mechanism constructs a hierarchy of schemas similar to LIDA's hierarchical streams of behaviors.

At time t, the previously learned schemas whose pre-subschemas match the current situation are *activated*. Activated schemas compete to propose their decisions according to a *proclivity value* computed from the activated schema's weight and the post-subschema's valence.

A decision to enact a composite schema generates a stream of behavior consisting of sequentially enacting the whole sequence of primitive schemas that constitute this composite schema. If the enaction of any of its primitive subschemas fails (i.e., the actual outcome differs from the schema's outcome), then the enaction of the sequence is interrupted.



activated	weight	valence	decision	proclivity	intended
(i01, a0, i00)	3	-1	a0	3.(-1) + 1.(-1) = -4	i00
((i00, i01), a0, i00)	1	-1			
(i01, a0, i10)	3	-1	a1	3.(-1) = -3	i10
(i01, i10a0, (i10, i11))	2	0	10a1	2.(0) + 1.(-1) = -2	(i10, i11)
(i01, i10a0, i11))	1	-1			
(i01, 00a0, (i00,i00))	2	-2	00a0	2.(-2) = -4	(i00, i00)

Fig. 11: Schema learning and selection. Schemas are nested tuples: ⟨pre-subschema, decision, post-subschema⟩. Over time, new decisions and new schemas are learned from the bottom up. Recently enacted schemas (in gray) activate the previously-learned higher-level schema whose pre-subschema they match. Activated schemas propose their post-subschemas with a proclivity value calculated from the activation weight and the expected valence. The schema with the highest proclivity is selected to try to enact.

Georgeon [12] reports an experiment in the environment shown in Fig. 12 and shared in a video [10]. The set of primitive actions is {`move-forward`, `turn-left`, `turn-right`, `feel-front`, `feel-left`, `feel-right`}. The set of possible outcomes is {`no-wall`, `wall`} corresponding to a single primitive item. The schema ⟨`move-forward`, `no-wall`⟩ has a valence of +5 making it the only "enjoyable" interaction to the agent. The schema ⟨`move-forward`, `wall`⟩, corresponding to bumping into wall, has a valence of −10 making it strongly dreaded. The 6 schemas that have a `feel` action have a valence of −1 making them slightly costly. The schemas that have a `turn` action have a valence of −3 making them mildly costly.

Like in the experiments with CALM and LIDA presented above, the difficulty comes from the fact that the agent only has a single binary sensory signal and cannot directly sense the state of the environment. The agent thus characterizes the state of the environment through enacted schemas. Results show that after approximately a hundred interaction cycles, the agent learns to use the `feel` schemas to actively sense the environment. This active sensing provides a situation awareness that makes the agent turn to empty cells when necessary, avoid bumping into walls, and optimize its moving forward.

Fig. 12: The small loop Challenge [12]. The agent can move forward to an empty (white) cell, bump into a wall (green cell), turn to the left or to the right by 90°, or "feel" the cell in front, to the left, or to the right. The sensory signal is a single-bit feedback from the action ("empty" or "wall"). The agent progressively learns to used the "feel" schemas to sense its surrounding environment, not move forward when it feels a wall in front, and turn to a direction where it feels no wall, thus maximizing the "move forward" interaction that has a predefined positive valence.

In a robot, primitive actions are defined as hard-coded control loops that regulate actuators and monitor sensory feedback over a predetermined period. For example, the `move-forward` action turns on the wheel engines and runs a loop that monitors the elapsed time and the collision detection sensor. If a collision is detected, the schema is terminated before its predetermined duration. Upon termination, the wheel engines are turned off and the outcome is sent to the schema mechanism, for example `no-bump` or `bump`.

To deal with the complexity of the physical world, Georgeon and colleagues encapsulated the schema mechanism in a cognitive architecture called Enactive Cognitive Architecture (ECA) [14]. ECA incorporates a spatial memory module that can simulate the enaction of schemas to help select the most desirable schema in the current spato-temporal context. Like in LIDA, schemas may include unbound variables that can be qualified upon their instantiation. For example, the `turn` action has the variable `angle` that specifies the rotation span in the horizontal plane (yaw). When the schema mechanism module selects the `turn` action, the spatial memory module qualifies the angle to the desired direction.

ECA was demonstrated with the robot shown in Fig. 13 [11]. When the robot finds reliable schemas in a consistent spatial localization, it creates a datastructure called a *phenomenon* that represents the "thing" in the world that affords these schema. This is analogous to the construction of synthetic items in Drescher's schema mechanism.

In summary, ECA's schema mechanism uses schemas as atomic elements for constructing higher-level schemas instead of sensory signals, synthetic items, and actions. This design choice facilitates the hierarchical leaning of composite schemas because they are not based on abstract representations of elements in

Fig. 13: Robot learning a phenomenon (synthetic item) representing a cylindrical object [17, Fig. 5]. Left: Photo of the robot showing its ultrasonic echo-localization sensor mounted on its pivoting head. Right: Representation of the *cylindrical phenomenon*. The phenomenon is aggregated from many sensorimotor "manifestations" of the cylindrical object through echo-localization measures. Blue triangles represent the cones of the ultrasonic signals sent by the robot over time from different positions around the moneybox. The black line is the shape of the cylindrical object reconstructed from the echoes.

the world. In the experiment reported in Fig. 12, the agent manages to adapt to its environment using only schemas but no synthetic items. Like Drescher and Kugele, Georgeon implemented an effect of *habituation* that causes reliable schemas to be preferably reused, which facilitates the construction of higher-level schemas on top of them. This habituation principles has echoes in Woolford's *sensorimotor sequence reiterator* [52].

ECA makes a strong design choice by assuming that the environment has a three-dimensional Euclidean structure and is populated with physical objects. This assumption is coded in the cognitive architecture through the spatial memory module and by adding unbound variables to schema that encode spatial parameters. This design choice allows ECA to construct phenomena to represent "things" in the physical world. We hypothesize that it can help avoid the scalability limitation of Drescher's implementation of synthetic items. Including spatial variable in schemas and qualifying them adequately, however, raises many questions that remain open.

2.5 Autocatalytic Endogenous Reflective Architecture (AERA)

AERA's models [49, 50, 27], similar to Drescher's schemas [5] and the neuro-symbolic approach presented in [21], represent knowledge by contextualized programs that capture antecedental and successional states of important relation-

ships (e.g. the relationship between a hand and an object that one wants to pick up). Unlike these, AERA places explicit representation of cause-effect relations at its core [51]. During its continuous learning, driven by a self-maintained dynamic goal hierarchy, the most useful (effective and efficient) models of cause-effect relations encountered in the environment are retained, while others are discarded. Based on predictions, informed experimentation, and strategic induction, the system autonomously creates new knowledge cumulatively, effortlessly unifying new information with old, even in light of contradictions. Further separating AERA from other approaches are its atomic elements for knowledge creation, whose operational semantics are transparent to the system's own reasoning, meaning that they can be inspected and learned by the same explicit, defeasible unified reasoning mechanisms as used for learning about the world.

Key representational components of this approach include [41, 43]:

- **Facts** are statements that represent a cognitive event, e.g. operations happening in AERA itself; facts are the only entity in AERA that have complete certainty (corresponding to Descartes' axiom "I think, therefore I am").
- A **composite State model (CST)** captures a set of simultaneous (timeless) Facts that the AERA system considers to be (potentially) useful and/or necessary for properly predicting an event's consequences; used to e.g. represent a variety of subsets in the world such as spatial relations of objects, the values of a perceived object's properties, such as its position and color, etc.
- A **cognitive event model (CEM)** captures a (hypothesized) causal influence of a Fact on another Fact; used to represent the effect that an action or event in particular circumstances, e.g. a command initiated by AERA at some timestep, has on the state of the world at a later timestep. When a CEM fires it produces a prediction (while the production of a prediction is represented as a Fact, the confidence in the prediction itself is defeasible and is in part based on the confidence of the CEM).
- A **requirement model (M_{req})** captures the (predicted) requirements/conditions that must be met for a CEM to be relevant in a particular context at a particular time.

AERA's programming paradigm allows the system itself to inspect these types of information structures, use them "for parts" to create variations, as well as create new ones from scratch. AERA's programming language, Replicode, allows it to autonomously use these building blocks for self-programming, in the form of coherent plans, predictions, and goals that it can pursue [26, 25, 48].

Both *CEM*s and M_{req}s have a left-hand side (LHS) and a right-hand side (RHS), composed of sets of Facts and constraints (e.g., math equations). The LHS Facts describe (assumed) conditions under which the RHS Facts would be produced. Figure 14 shows how the above components lead to making a prediction. When some sensory Facts match everything in a *CEM*'s LHS, a relevant M_{req} allows the *CTS*'s instantiation, which means the conditions for making a prediction via a *CEM* are met. A successful instantiation of a *CEM* means that everything in its RHS will happen. As the *CEM* predicts a future

state after an event's occurrence, this is verified and marked as a "successful prediction" by AERA only if the prediction matches the state that is observed.

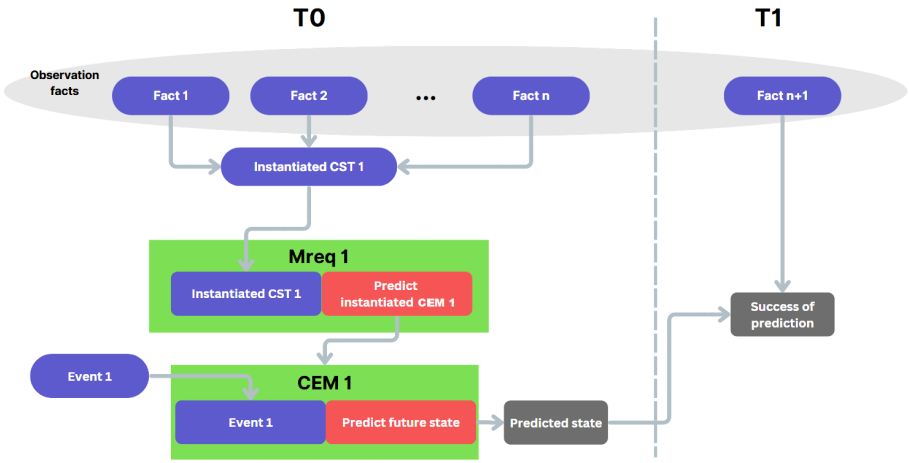


Fig. 14: The connection between CST, M_{req}, and CEM. Time T0 (left): The observation facts matching $CST1$ allow M_{req} to trigger a prediction by instantiating $CEM1$. Time T1 (right): AERA checks whether the observations match the predicted state.

As an example, the triad of $CST1$, $M_{req}1$, and $CEM1$ can represent the physical movement behavior of an entity when it collides with another entity (e.g., a hand). $CEM1$ represents the dynamics of the movement, connecting the collision event (left-hand side of $CEM1$) to the moved entity's new position (right-hand side of $CEM1$). $CST1$ represents the preconditions of the behavior, such as the entities' position, weight, etc., and the fact that the point of collision and the moved entities' position must initially be the same. $M_{req}1$ predicts that if a striking entity hits another entity with the conditions described by $CST1$, the stricken entity's position will change a specific amount in the next time step.

The interactions of the operational principles of AERA are quite intricate in their entirety; important aspects that are covered elsewhere include induction – the creation of new knowledge – unified abduction-deduction – the creation of plans and explanations – and runtime relevance computations [26]. The unified integration of these operational principles in a single coherent architecture is what gives AERA its power, allowing its seed – initial human-provided knowledge – to be orders of magnitude smaller than what AERA can create on its own through learning from experience.

These mechanisms have been demonstrated for control of virtual robots, learning and using language, and multimodal interaction [50]. Fig. 15 shows the latest experiment reported in [41] and shared in video [42]. The task has a *guided experimentation* phase, during which the experimenter tele-operates the

robot, and a *task solving* phase, during which the robot must autonomously learn to grasp new objects it has never grasped before, which are novel in shape but familiar in size.

During the guided phase, commands and successful grasping events makes AERA learn triads of Causal Event Models (CEMs), Composite State models (CSTs), and requirement models (Mreqs) for each command. During the task-solving phase, the analogy process induces new grasping models based on the objects' familiarity. Results show that, after several trials unsuccessfully testing the learned models, AERA induces new CSTs and Mreqs capable of triggering the appropriate actions.

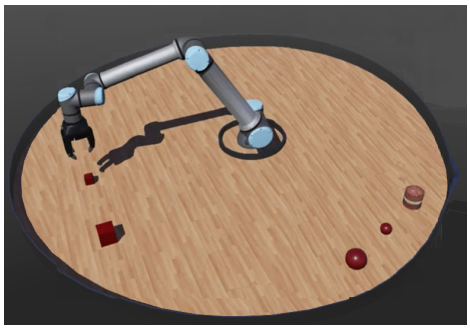


Fig. 15: Experimentation from [44, Fig. 2]. In the guided phase, the robot is tele-operated to grab the small and big cubes on the left; in the task solving phase, it quickly learns to autonomously grab the small ball, big ball, and big cylinder on the right by analogy.

In summary, AERA introduces the capacity for the schema mechanism to process events that have en external cause in addition to those caused by the actions initiated by the agent. This is done by distinguishing the context in which the event occurs, represented by Composite State models (CSTs), from the hypothetical causal effects of the event stored in Cognitive Event Models (CEMs). The agent can thus construct hypothetical causal models as it passively observes other agents interacting with the world or as it observes its own interactions while being tele-operated. The agent can then test the hypothetical causal models by itself to evaluate their reliability.

Both the events caused by the agent's actions and those passively observed generate the instantiation of cognitive events and the generation of *prediction facts*. AERA considers both external events and cognitive events as actual facts that AERA can internally process, which opens the way to reflexivity.

3 Conclusion

Schema mechanisms are characterized by their dynamic approach to constructing concepts from experience of interaction. Compared to more classical techniques such as statistical symbol grounding [20] or active inference [9], this dynamic approach takes into account counterfactual sequences of interactions for the construction of concepts. This makes schema mechanisms particularly reflect Piaget's core idea that sensorimotor schemas serve as the initial instruments for constructing knowledge.

The four new schema mecanisms presented here, CALM, LIDA, ECA, and AERA, bring an additional idea to the picture: the capacity to deal with sensory signals (primitive items) that are not "percepts representing elements of the world", but mere feedback from action or events of interaction. In so doing, these four new schema mechanisms address a limitation pointed out by Bettoni when he wrote that "Drescher's constructivism is not Piaget's constructivism, mainly because of its tacit acceptance of cognitive dogmatism" [2, p. 6]. We interpret this critique as suggesting that using primitive items that represent elements of the state falls within cognitive dogmatism because it implements the assumption that the agent has direct access to the ontology of the world. In contrast, constructing schemas from interactional events better reflects Piaget's idea that knowledge arises from an initial "complete un-differentiation between the subject and the object".

Basing schemas on interactional events is particularly important when designing the agent's control system. Drescher's agents sought goal states defined in the environment—a form of *extrinsic motivation* that is inherently prone to scalability issues as the complexity of the environment grows. In contrast, the new schema mechanisms seek rewarding interactions as illustrated in the experiments above: CALM: moving left and right, LIDA: winning, ECA: moving forward, and AERA: grabbing objects. This is a form of *interactional motivation* [13] whose complexity may not grow as the complexity of the environment grows, allowing easier transfer to robots in the open world.

LIDA and ECA have demonstrated interactional motivation using both negatively valenced primitive schemas (e.g., LIDA: `deposit`, ECA: `feel` and `turn`) and positively valenced ones (LIDA: `win`, ECA: `move-forward`). They learn composite actions made of sequences of both kinds of schemas (LIDA: ⟨`deposit`, `win`⟩, ECA: ⟨`turn`, `move-forward`⟩), which allows the emergence of behaviors adapted to delayed reward. These sequences work as short programs that the agent can re-execute, which leads to self-programming—an important aspect of constructivist epistemology [15].

CALM has focused on the construction of hierarchies of synthetic items, exploring Piaget's idea of assimilation and accommodation. LIDA, ECA, and AERA have encapsulated the schema mechanism within a cognitive architecture. They handle schemas with unbound variables, reducing the number of schemas and helping manage complexity, albeit at the cost of potentially diminishing generality. These cognitive architectures internally simulate the enaction of the schemas before selection and qualification. AERA has further explored

this direction by allowing reflexivity upon internal events. Altogether, the four new schema mechanisms extend our vision of how Piaget's ideas can be translated into artificial intelligence. Because they incorporate significantly more of constructivist epistemology than Drescher's original version, we propose to call them schema mechanisms 2.0. Fig. 16 synthesizes their contributions.

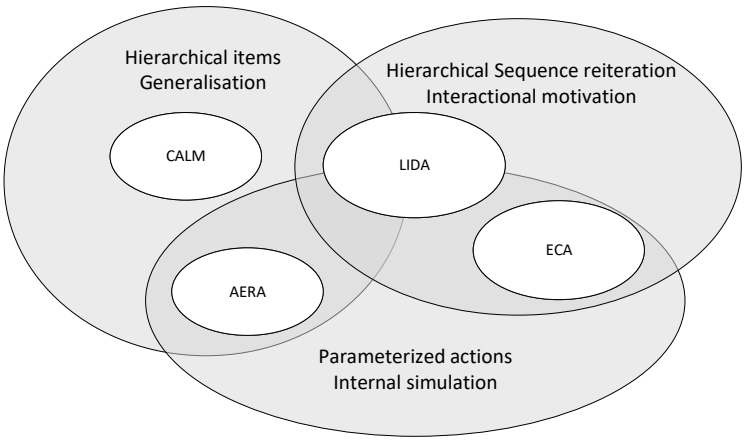


Fig. 16: Coverage of principal dimensions by schema mechanisms 2.0.

Schema Mechanisms 2.0 have still not passed the sensorimotor stage of development—the first defined by Piaget. The question whether their implementation in computers is overly reductionist compared to biological systems remains open. They nonetheless demonstrate some promising results with robots. LIDA and ECA have implemented an Euclidean spatial memory module and an intuitive physics engine to handle schemas that encode physical properties. This is a strong commitment that reduces generality but may open the way to developmental artificial intelligence for autonomous robots in the open world.

Acknowledgments. This article benefited from discussions with Gary Drescher, Sean Kugele, and Jeff Thompson.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Babakr, Z.H., Mohamedamin, P., Kakamad, K.: Piaget's cognitive developmental theory: Critical review. Education Quarterly Reviews **2**(3) (2019). https://doi.org/10.31014/aior.1993.02.03.84
2. Bettoni, M.C.: Made-up minds: A constructivist approach to artificial intelligence - a book review. AI Communications **6**(3), 234–240 (1993). https://doi.org/10.3233/AIC-1993-63-413

3. Drescher, G.L.: A mechanism for early piagetian learning. In: Forbus, K.D., Shrobe, H.E. (eds.) Proceedings of the 6th National Conference on Artificial Intelligence. Seattle, WA, USA, July 1987. pp. 290–294. Morgan Kaufmann (1987)
4. Drescher, G.L.: Made-up minds: a constructivist approach to artificial intelligence. phdthesis, MIT (1989)
5. Drescher, G.L.: Made-up minds: a constructivist approach to artificial intelligence. Artificial intelligence, MIT Press (1991)
6. Drescher, G.L.: The schema mechanism. In: Hanson, S.J., Remmele, W., Rivest, R.L. (eds.) Machine Learning: From Theory to Applications - Cooperative Research at Siemens and MIT. Lecture Notes in Computer Science, vol. 661, pp. 125–138. Springer (1993). https://doi.org/10.1007/3-540-56483-7_27
7. Franklin, S.: A foundational architecture for artificial general intelligence. In: Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms, pp. 36–54. IOS Press (2007)
8. Franklin, S., Madl, T., D'mello, S., Snaider, J.: Lida: A systems-level architecture for cognition, emotion, and learning. IEEE Transactions on Autonomous Mental Development **6**(1), 19–41 (2013)
9. Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., Pezzulo, G.: Active inference: A process theory. Neural Computation **29**(1), 1–49 (2017). https://doi.org/10.1162/NECO_a_00912
10. Georgeon, O.L.: The small loop challenge (2012), https://youtu.be/Mstl8BRCbSI
11. Georgeon, O.L., Lurie, D., Robertson, P.: Artificial enactive inference in three-dimensional world. Cognitive Systems Research **86**, 101234 (2024). https://doi.org/10.1016/j.cogsys.2024.101234
12. Georgeon, O.L., Marshall, J.: The small loop problem: A challenge for artificial emergent cognition. In: International Conference on Biologically Inspired Cognitive Architectures. pp. 137–144 (2012)
13. Georgeon, O.L., Marshall, J.B., Gay, S.: Interactional motivation in artificial systems: Between extrinsic and intrinsic motivation. In: 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL). pp. 1–2. IEEE (2012). https://doi.org/10.1109/DevLrn.2012.6400833
14. Georgeon, O.L., Marshall, J.B., Manzotti, R.: ECA: An enactivist cognitive architecture based on sensorimotor modeling. Biologically Inspired Cognitive Architectures **6**, 46–57 (2013). https://doi.org/10.1016/j.bica.2013.05.006
15. Georgeon, O.L., Riegler, A.: CASH only: Constitutive autonomy through motorsensory self-programming **58**, 366–374 (2019). https://doi.org/10.1016/j.cogsys.2019.08.006
16. Georgeon, O.L., Ritter, F.E.: An intrinsically-motivated schema mechanism to model and simulate emergent cognition. Cognitive Systems Research **15-16**, 73–92 (2012). https://doi.org/10.1016/j.cogsys.2011.07.003
17. Georgeon, O.L., Vidal, J.R., Knockaert, T., Robertson, P.: Simultaneous localization and active phenomenon inference (SLAPI). In: Thórisson, K.R. (ed.) Proceedings of the Third International Workshop on Self-Supervised Learning. Proceedings of Machine Learning Research, vol. 192, pp. 77–88. PMLR (2022)
18. Glasersfeld, E.v.: Radical constructivism: a way of knowing and learning. No. 6 in Studies in mathematics education series, Falmer Press, reprinted edn.
19. Guerin, F., Kruger, N., Kraft, D.: A survey of the ontogeny of tool use: From sensorimotor experience to planning. IEEE Transactions on Autonomous Mental Development **5**(1), 18–45 (2013). https://doi.org/10.1109/TAMD.2012.2209879
20. Harnad, S.: The symbol grounding problem. Physica D: Nonlinear Phenomena **42**(1), 335–346 (1990). https://doi.org/10.1016/0167-2789(90)90087-6

21. Komrusch, S., Minsky, H.: Symbolic guidance for constructivist learning by neural model. In: Proceedings of the Third International Workshop on Self-Supervised Learning. pp. 63–76. PMLR, https://proceedings.mlr.press/v192/komrusch22a.html, ISSN: 2640-3498
22. Kringelbach, M., Berridge, K.: The functional neuroanatomy of pleasure and happiness **9**, 579–87 (2010)
23. Kugele, S.: Constructivist procedural learning for grounded cognitive agents. Cognitive Systems Research **90**, 101321 (2025). https://doi.org/10.1016/j.cogsys.2025.101321
24. Kugele, S., Franklin, S.: Learning in LIDA **66**, 176–200 (2021). https://doi.org/10.1016/j.cogsys.2020.11.001
25. Nivel, E., Thórisson, K.R.: Replicode: A constructivist programming paradigm and language. Technical RUTR-SCS13001, Reykjavik University School of Computer Science (2013)
26. Nivel, E., Thórisson, K.R.: Towards a programming paradigm for control systems with high levels of existential autonomy. In: International Conference on Artificial General Intelligence. pp. 78–87. Springer (2013)
27. Nivel, E., Thórisson, K.R., Dindo, H., Pezzulo, G., Rodriguez, M., Corbato, C., Steunebrink, B., Ognibene, D., Chella, A., et al.: Autocatalytic endogenous reflective architecture (2013)
28. Oudeyer, P.Y., Kaplan, F., Hafner, V.V.: Intrinsic motivation systems for autonomous mental development **11**(2), 265–286 (2007). https://doi.org/10.1109/TEVC.2006.890271
29. Perotto, F.S.: Um Mecanismo Construtivista para Aprendizagem de Antecipações em Agentes Artificiais Situados. phdthesis, Universidade Federal do rio Grande do Sul (UFRGS), Porto Alegre, Brazil (2010)
30. Perotto, F.S.: Un Mécanisme Constructiviste d'Apprentissage Automatique d'Anticipations pour des Agents Artificiels Situés. phdthesis, National Polytechnic Institute of Toulouse (INP), France (2010), http://www.theses.fr/2010INPT0041/document, thèse de doctorat dirigée par Buisson, Jean-Christophe et Campos Álvares, Luís Otávio Intelligence artificielle Toulouse, INPT 2010
31. Perotto, F.S.: A computational constructivist model as an anticipatory learning mechanism for coupled agent environment systems. Constructivist foundations **1**(9), 46–56 (2013)
32. Perotto, F.S., Alvares, L.O.: Incremental inductive learning in a constructivist agent. In: Bramer, M., Coenen, F., Tuson, A. (eds.) Research and Development in Intelligent Systems XXIII, Proceedings of AI-2006, the Twenty-sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, 11-13 December, 2006. pp. 129–142. Springer (2006). https://doi.org/10.1007/978-1-84628-663-6_10
33. Perotto, F.S., Alvares, L.O.: Learning regularities with a constructivist agent. In: Nakashima, H., Wellman, M.P., Weiss, G., Stone, P. (eds.) 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006. pp. 807–809. ACM (2006). https://doi.org/10.1145/1160633.1160778
34. Perotto, F.S., Buisson, J.C., Alvares, L.O.C.: Constructivist anticipatory learning mechanism (calm): Dealing with partially deterministic and partially observable environments. In: International Conference on Epigenetic Robotics (EpiRob), Piscataway, NJ, USA. pp. 117–127. Lund University Cognitive Science (2007)

35. Piaget, J., Inhelder, B.: The Psychology of the Child. Basic Books, 2 edn. (1969)
36. Piaget, J.: The Construction of Reality in the Child. Routledge and Kegan Paul (1955)
37. Piaget, J.: The Origin of Intelligence in the Child. Routledge and Kegan Paul (1955)
38. Piaget, J.: The Principles of Genetic Epistemology. Psychology Press (1997)
39. Piaget, J.: L'épistémologie génétique. Quadrige, PUF (2011)
40. Raju, R.V., Guntupalli, J.S., Zhou, G., Lázaro-Gredilla, M., George, D.: Space is a latent sequence: Structured sequence learning as a unified theory of representation in the hippocampus (arXiv:2212.01508) (2022), http://arxiv.org/abs/2212.01508
41. Sheikhlar, A.: Autonomous causal generalization (2024)
42. Sheikhlar, A.: Sizemattersgrab experiment (2024), https://youtu.be/JXgdSjU-7OI
43. Sheikhlar, A., Thórisson, K.R.: Causal generalization via goal-driven analogy. In: International Conference on Artificial General Intelligence. pp. 165–175. Springer (2024)
44. Sheikhlar, A., Thórisson, K.R.: Causal generalization via goal-driven analogy. In: Thórisson, K.R., Isaev, P., Sheikhlar, A. (eds.) Artificial General Intelligence. vol. 14951, pp. 165–175. Springer Nature Switzerland (2024). https://doi.org/10.1007/978-3-031-65572-2_18
45. Singh, S., Littman, M.L., Jong, N.K., Pardoe, D., Stone, P.: Learning predictive state representations. In: Proceedings of the Twentieth International Conference on International Conference on Machine Learning. p. 712–719. ICML'03, AAAI Press (2003)
46. Smith, R., Friston, K.J., Whyte, C.J.: A step-by-step tutorial on active inference and its application to empirical data. Journal of Mathematical Psychology **107**, 102632 (2022). https://doi.org/10.1016/j.jmp.2021.102632
47. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, 2 edn. (2018)
48. Thórisson, K.R.: A new constructivist ai: from manual methods to self-constructive systems. In: Theoretical Foundations of Artificial General Intelligence, pp. 145–171. Springer (2012)
49. Thórisson, K.R., Bieger, J., Li, X., Wang, P.: Cumulative learning. In: Artificial General Intelligence: 12th International Conference, AGI 2019, Shenzhen, China, August 6–9, 2019, Proceedings 12. pp. 198–208. Springer (2019)
50. Thórisson, K.R., Nivel, N., Steunebrink, B.R., Helgason, H.P., Pezzulo, G., Sanz Bravo, R., Schmidhuber, J., Dindo, H., Rodríguez Hernández, M., Chella, A., et al.: Autonomous acquisition of natural situated communication. IADIS International Journal on Computer Science And Information Systems **9**(2), 115–131 (2014)
51. Thórisson, K.R., Talbot, A.: Cumulative learning with causal-relational models. In: Artificial General Intelligence: 11th International Conference, AGI 2018, Prague, Czech Republic, August 22-25, 2018, Proceedings 11. pp. 227–237. Springer (2018)
52. Woolford, F.M.G., Egbert, M.D.: A precarious sensorimotor sequence reiterator for modelling enactive habits. In: ALIFE 2020: The 2020 Conference on Artificial Life. pp. 771–779. MIT Press (2020). https://doi.org/10.1162/isal_a_00249
53. Ziemke, T.: The construction of 'reality' in the robot: Constructivist perspectives on situated artificial intelligence and adaptive robotics **6**(1), 163–233 (2001). https://doi.org/10.1023/A:1011394317088