ECA: An enactivist cognitive architecture based on sensorimotor modeling

Olivier Georgeon, James Marshall, Riccardo Manzotti

ECA: An enactivist cognitive architecture based on sensorimotor modeling

Olivier L. GEORGEON[a,b], James B. MARSHALL[c], and Riccardo MANZOTTI[d]
[a] *Université de Lyon, CNRS*
[b] *Université Lyon 1, LIRIS, UMR5205, F-69622, France*
[c] *Sarah Lawrence College, Bronxville, NY 10708, USA*
[d] *IULM University, Milan, Italy*

Abstract. A novel way to model an agent interacting with an environment is introduced, called an Enactive Markov Decision Process (EMDP). An EMDP keeps perception and action embedded within sensorimotor schemes rather than dissociated, in compliance with theories of embodied cognition. Rather than seeking a goal associated with a reward, as in reinforcement learning, an EMDP agent learns to master the sensorimotor contingencies offered by its coupling with the environment. In doing so, the agent exhibits a form of intrinsic motivation related to the *autotelic principle* (Steels, 2004), and a value system attached to interactions called interactional motivation. This modeling approach allows the design of agents capable of autonomous self-programming, which provides rudimentary constitutive autonomy—a property that theoreticians of enaction consider necessary for autonomous sense-making (e.g., Froese & Ziemke, 2009). A cognitive architecture is presented that allows the agent to autonomously discover, memorize, and exploit spatio-sequential regularities of interaction, called Enactive Cognitive Architecture (ECA). In our experiments, behavioral analysis shows that ECA agents develop active perception and begin to construct their own ontological perspective on the environment.

Keywords. Enaction; self-motivation; cognitive architecture; developmental learning.

Introduction

In cognitive science, there has been a customary and traditional tripartite division of the mind between perception, the control system, and motor action. This view has been nicely dubbed the "classic sandwich model" by Susan Hurley (1998). Many control architectures are built in this way. Since the 1980s there have been many attempts to challenge this traditional picture particularly in the field of robotics (e.g., Brooks, 1991) but also from a more psychological and theoretical perspective (e.g., Hirose, 2002; Shanahan, 2010; Ziemke, 2001). In particular, the idea emerged that it might be a mistake to consider sensation independently from action and that we should design cognitive systems on the basis of *low-level sensorimotor loops* that represent *sensorimotor patterns* of interaction. This intuition gained momentum from other related views such as embodied cognition (e.g., Anderson, 2003; Holland, 2004), ecological psychology (Chemero & Turvey, 2007; Gibson, 1979), sensorimotor theories (O'Regan & Noë, 2001; O'Regan, 2012), morphological robotics (Paul, 2006;

Pfeifer & Bongard 2006; Pfeifer, 1999), developmental robotics (Lungarella, Metta, Pfeifer, & Sandini, 2003), and epigenetic robotics (Berthouze & Ziemke, 2003; Zlatev, 2001). Here, we introduce a modeling approach that goes a step beyond the notion of low-level sensorimotor loops by simply considering sensorimotor patterns—also called *sensorimotor schemes* by Piaget (1951)—as the atomic elements manipulated by our algorithms.

Varela and his coauthors (1991) coined the term *enactive perception* to suggest that organism and environment are coupled together. The features of the environment to which an organism responds are singled out by the ongoing activity in the organism. The domain that defines this coupling has been called the *relational domain* (e.g., Froese & Ziemke, 2009). The theory of enaction, initiated by Varela, stresses that the relational domain evolves over the organism's life in a manner that is codetermined by the organism and the environment. The fact that the relational domain is not predefined makes possible the organism's *constitutive autonomy*—the capacity of the organism to "self-constitute its identity" (Froese & Ziemke, 2009). These authors argue that constitutive autonomy is an important aspect of organisms because it is a precondition of sense-making and intrinsic teleology, and is thus a property that we should seek to obtain in artificial agents.

Furthermore, the term enaction also incorporates the idea that perception involves physical activity, or action. A model of reference was offered by O'Regan & Noë's (2001) *sensorimotor contingencies* theory. To perceive the world is to master the sensorimotor contingencies between the body and the world. Every sensor modality is characterized by "the structure of the rules governing the sensory changes produced by various motor actions, that is, what we call the sensorimotor contingencies" (O'Regan, 2001, p. 941).

The enactivist approach suggests modeling a cognitive agent on the basis of sensorimotor interactions with the environment. This paper is an attempt in that direction. In the next section, we introduce a new type of algorithm that does not separate perception from action, called an Enactive Markov Decision Process (EMDP). An EMDP provides a useful conceptual framework for designing agents capable of intrinsically-motivated self-programming as they interact with their environment. We qualify such self-programming as *sensorimotor* because it consists of learning a series of sensorimotor schemes that are subsequently executed as programs. We argue that sensorimotor self-programming opens the way to constitutive autonomy.

While acknowledging that EMDP problems are intractable in the general case, we present two instances in which the coupling with the environment allows the agent to learn to master sensorimotor contingencies within a reasonable frame. The first is called a hierarchical sequential EMDP problem. The second is called a Spatial Enactive Markov Decision Process (SEMDP). A SEMDP is intended to model an agent interacting with an environment that has a Euclidian spatial structure, such as the real world. This work leads us to propose a cognitive architecture dedicated to agents confronted with SEMDP problems, called the Enactive Cognitive Architecture (ECA).

Formalism for enactive learning problems

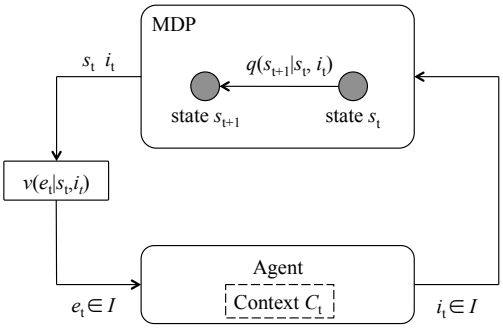
The philosophy of an EMDP is that the agent tries to enact an *intended sensorimotor scheme*, and is informed by the environment whether this intended scheme was indeed enacted, or whether another scheme was enacted instead. In the former case, the

intended scheme is considered *successfully enacted*; in the latter case, the intended enaction failed and another scheme was *actually enacted* instead. While EMDP problems differ from reinforcement learning problems, we present them using a similar formalism to allow for comparison.

Enactive Markov Decision Process (EMDP)

Formally, we define an EMDP as a tuple (S, I, q, v) in which S is the set of environment states; I is the set of *primitive interactions* offered by the coupling between the agent and the environment; q is a probability distribution such that $q(s_{t+1}|s_t,i_t)$ gives the probability that the environment transitions to state $s_{t+1} \in S$ when the agent chooses interaction $i_t \in I$ in state $s_t \in S$; and v is a probability distribution such that $v(e_t|s_t,i_t)$ gives the probability that the agent receives the input $e_t \in I$ after choosing i_t in state s_t. We call i_t the *intended interaction* because it represents the sensorimotor scheme that the agent intends to enact at the beginning of step t; i_t constitutes the agent's output sent to the environment. We call e_t the *enacted interaction* because it represents the sensorimotor scheme that the agent records as actually enacted at the end of step t; e_t constitutes the agent's input received from the environment. If the enacted interaction equals the intended interaction ($e_t = i_t$) then the attempted enaction of i_t is considered a *success*, otherwise, it is considered a *failure*.

As an example, primitive interactions may represent tactile sensorimotor schemes, which consist of the combination of a movement and the sensory stimulation generated by the movement. When the agent tries to enact a tactile interaction, this may result in a success if the agent indeed touched something, or in a failure if the agent touched nothing, in which case the actually enacted interaction represents the sensorimotor scheme of touching nothing. Figure 1 shows the EMDP cycle and algorithm. A complete EMDP example will be presented later in Figures 7a and 7b.



1. Let $t = 0$ and let $s_0 \in S$ denote the initial environment state.
2. Choose the intended interaction $i_t \in I$.
3. Transit to the new state $s_{t+1} \in S$ with probability $q(s_{t+1}|s_t,i_t)$.
4. Generate the enacted interaction $e_t \in I$ with probability $v(e_t|s_t,i_t)$.
5. $t = t + 1$.
6. Goto 2.

Figure 1: Left: diagram of an Enactive Markov Decision Process (EMDP). Right: the EMDP algorithm. At time t, the agent chooses an *intended primitive interaction* i_t. The attempt to enact i_t generates a transition of the environment (represented by the Markov Decision Process MDP) from state s_t to state s_{t+1}. The agent then receives the *enacted primitive interaction* e_t. If $e_t = i_t$ then the attempt to enact i_t is considered a success, otherwise, a failure. The agent's "perception of its environment" is an internal construct C_t rather than the input e_t.

There are four formal differences between an EMDP and a Partially Observable Markov Decision Process (POMDP) (Kaelbling, Littman, & Cassandra, 1998): (a) the agent's input and output belong to the same set I rather than two different sets

(observations and actions); (b) there is no reward defined as a function of the environment, and the goal is not that the system reaches a reward state; (c) the cycle does not start from the environment but from the agent, making the agent's input e_t a consequence of the agent's output i_t rather than a premise. Notably, some authors (e.g., Pfeifer & Scheier, 1994) have called for this conceptual inversion of the perception/action cycle, but, to our knowledge, our algorithm is the first attempt to formalize it. More precisely, what we consider an EMDP agent's "perception" is an internal construct created through interaction rather than merely the input e_t. (d) The agent's input e_t is computed from the state s_t that precedes the tentative enaction of i_t rather than from the state that follows the previous action in a POMDP.

To complete the definition of an *EMDP problem*, we now need to define the objectives that we seek to achieve in using the EMDP formalism. In contrast to reinforcement learning (e.g., Sutton & Barto, 1998), our objective is not to design agents that learn to maximize a reward function over time. Neither is it to implement an agent that finds a goal state by exploring a problem space, as in problem-solving. Instead, an EMDP agent is driven by a more complex form of self-motivation, which we address next.

Self-motivation

Our approach to agent motivation resonates with Dreyfus's (2007) vision of a "Heideggerian Artificial Intelligence" that suggests that "we are drawn to move so as to achieve a better and better grip on our situation". Dreyfus notes: "for this movement towards maximal grip to take place, one doesn't need a mental representation of one's goal nor any subagential problem solving". In accordance with this vision, we designed an algorithm to control EMDP agents that learn to successfully enact sequences of interactions (following our definition of a *successful enaction* introduced above). This tendency involves neither a reward nor a goal; it is intrinsically encoded in the algorithm through discovering, recording, and re-enacting sequences of interactions that capture regularities in the coupling with the environment, as we will explain in the next section.

Since the algorithm does not use a reward function or a problem representation, it constitutes neither a reward maximization algorithm nor a problem-solving algorithm, but is better described—using Dreyfus's term—as a "skillful coping" algorithm. This skillful coping principle relates to the autotelic principle (Steels, 2004) and to the principle of optimal experience (Csikszentmihalyi, 1990) because, to an external observer, the agent seems to enjoy being *in control* of its activity. Here, we call this motivational principle *autotelic motivation*. More broadly, autotelic motivation falls within the area of *intrinsic motivation* (e.g., Oudeyer, Kaplan, & Hafner, 2007; Blank, Kumar, Meeden, & Marshall, 2005; Schmidhuber, 2010) because the agent's preferences are defined independently of any reference to the environment's states. Since autotelic motivation does not involve a reward or problem-solving, but is rather a "way of being in the world", it is not assessed through a synthetic scalar value, but is rather demonstrated through an analysis of the agent's behavior, as we will do in the experiment reported below.

In addition to implementing a tendency to successfully enact interactions, we found that the learning process required an innate value system so that all the interactions are not equal to the agent. Such an innate value system relates to fundamental constraints that, metaphorically, involve the agent's survival. Examples of such constraints are *eating* and avoiding *being hurt*. This is metaphorical because we

are talking about virtual agents or robots that do not really eat or get hurt. This innate value system, nonetheless, provides a reason why the agent should even learn to cope with the environment in the first place: the agent should be able to efficiently enact interactions that favor its survival and avoid interactions that jeopardize its survival.

We found that the EMDP model allowed us to encode metaphorical survival preferences by associating a value with the interactions that we define as concerning the agent's survival needs. These constraints generate a form of motivation that we call interactional motivation (Georgeon, Marshall, & Gay, 2012): the motivation to enact interactions with predefined positive values and to avoid interactions with predefined negative values. We predefine a slightly negative value for interactions that do not directly concern the agent's survival needs, to represent a light cost of enacting them. We expect an EMDP agent to learn to skillfully cope with the environment using all the interactions at its disposal, and to demonstrate that it can use these skills for its own good by eventually enacting interactions that have positive values and avoiding interactions that have strong negative values.

We formally define interactional motivation through a function r: $I \rightarrow \mathbb{R}$ that associates a scalar value $r(e)$ with each primitive interaction $e \in I$. In addition to learning to successfully enact interactions, the agent tends to try to enact interactions whose value $r(e)$ is positive and to try to avoid interactions whose value $r(e)$ is negative. Note that interactional motivation differs from reinforcement learning with intrinsic reward (e.g., Singh, Barto, & Chentanez, 2005) by the fact that the value function $r(e)$ is defined independently of any state of the system (either considered internal or external to the agent). An EMDP agent is motivated to enact an interaction for the sake of enacting it rather than for the sake of the outcome of the interaction. As a concrete example, an interactionally motivated agent would seek to ingest food whereas an intrinsic-reward agent would seek to get a full stomach. The proclivity to eat is not acquired from previous experience of eating but is instead primitive. This view accounts for the fact that newborn mammals are drawn towards their mother's milk even before having ever eaten. We feel that this central role given to interactions conforms to Dreyfus's view that we should "program this experiential aspect of being drawn in by an affordance", and, more generally, to Heidegger's philosophy that behavior is prior to knowledge (e.g., cited by Sun, 2004). We also find some resonance with Dennett's (1991) *inversion of reasoning* argument, which he uses to develop his theory of consciousness. Because of this difference and its implications in the algorithm design, we do not call r a reward function but rather a *value function* associated with interactions.

To fulfill both its autotelic and interactional motivations, an EMDP agent must learn to actively recognize situations in which interactions with positive values can be successfully enacted, and to place itself in such situations, while staying away from situations in which negative interactions cannot be avoided. Because the agent's knowledge of the situation is only obtained from regularities learned as the agent interacts with the environment, we expect the agent to discover, memorize, and exploit such regularities when they exist. The next subsection formally defines a regularity of interactions as a series of interactions that can be learned through experience and enacted as a whole sequence.

Self-programming EMDP agents

We define a *serial interaction* i_s as a series of k primitive interactions $i_s = \langle i_{p1}, \ldots i_{pk} \rangle$, with $i_{p1}, \ldots i_{pk} \in I$. We let X_t be the set of all interactions, primitive or serial, known by

the agent at time *t*. X_t is initialized with I (i.e., at time t_0, $X_0 = I$), and extended as the agent learns new serial interactions. We extend r to be a function from $X_t \rightarrow \mathbb{R}$ that gives the motivational value of a serial interaction as the sum of the values of its primitive interactions, meaning that enacting a serial interaction has the same motivational value as separately enacting all of its primitive interactions. A self-programming agent can choose to enact any interaction in X_t, primitive or serial. We call the mechanism that chooses an interaction the *decisional mechanism*, the point in time t_d when this choice is made a *decision time*, and the time lapse during which an interaction is enacted a *decision cycle*. Decision steps thus do not occur on each time step but rather between each decision cycle.

At decision time t_d, trying to enact a serial interaction $i_s \in X_d$ consists of sequentially trying to enact the k primitive interactions that compose i_s over the next k time steps t_d+1, … t_d+k. If all the primitive interactions are successfully enacted, then the enaction of i_s is a success; the decision cycle ends at time t_d+k; and the actually enacted serial interaction is $e_s = i_s = \langle e_{p1} \ldots e_{pk} \rangle = \langle i_{p1} \ldots i_{pk} \rangle$. If the enaction of the j^{th} element of i_s fails, then the decision cycle is interrupted at time t_d+j and the actually enacted serial interaction is the series of the j actually enacted primitive interactions: $e_s = \langle e_{p1} \ldots e_{pj} \rangle = \langle i_{p1} \ldots i_{pj-1}, e_{pj} \rangle$. Figure 2 illustrates this principle. The dashed lines represent the *decision cycle* and the solid lines the *primitive cycle*. A full circuit of the decision cycle involves several circuits of the primitive cycle. As the agent learns longer sequences, the decisional mechanism thus ascends to higher levels of time scales. Such a capacity to cover different time scales has often been called for, particularly in the reinforcement learning (e.g., Sutton, Precup, & Singh, 1999) and cognitive architectures communities (e.g., Sun, 2004; Albus, 1993).

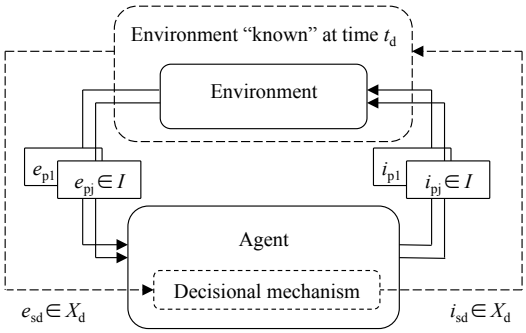


Figure 2: Diagram of a self-programming EMDP agent. At the beginning of decision cycle t_d (dashed loop), the agent's decisional mechanism chooses the intended serial interaction $i_{sd} = \langle i_{p1}, \ldots i_{pk} \rangle$ from amongst the set X_d of serial interactions known at time t_d. The enaction of i_{sd} consists of trying to enact the k intended primitive interactions $i_{p1} \ldots i_{pk}$ one after another (solid loops). If the enaction of i_{pj} fails ($e_{pj} \neq i_{pj}$) then the enaction of i_{sd} is interrupted. The decisional mechanism then receives the actually enacted serial interaction $e_{sd} = \langle e_{p1}, \ldots e_{pj} \rangle$, $j \leq k$. From the perspective of the decisional mechanism, e_{sd} thus seems to be enacted as a single interaction in a virtual "environment known by the agent at time t_d". Because the decisional mechanism ignores the primitive loop, the learning algorithm can apply recursively, independently of the length of the enacted serial interaction.

Self-programming EMDP agents are designed to discover regularities of interactions through trial and error and to encode such regularities as serial interactions. Once a serial interaction is learned, the agent tries to enact it again in contexts in which the agent anticipates that it can be successfully enacted. We describe such an agent as self-programming because serial interactions work as programs that the agent learns

and subsequently executes. Rather than being written in a conventional programming language, such programs are written in the "agent's programming language" in the sense that they are made of sequences of instructions that the agent knows how to execute.

Since, at decision step t_d, the agent knows its situation through serial interactions that were enacted recently, and since such recently enacted serial interactions were learned earlier in the agent's own singular experience, the decision is taken as if it were based on the "virtual environment known by the agent at time t_d". Over time, a given instance of agent develops its own singular "vision of itself interacting with the environment" based on its own experience. This results in the evolution of the relational domain that represents the coupling of the agent with the environment, which leads to the construction of an individual identity, and thus implements a form of constitutive autonomy.

Notably, learning regularities of interactions is an intractable problem in the general case, as in solving a general POMDP problem (Kaelbling et al., 1998); the time required to discover regularities is likely to grow exponentially with their length. An EMDP agent will thus only "survive" if the coupling with its environment offers regularities that the agent can find and exploit before it runs out of resources. In previous studies (Georgeon & Ritter, 2012; Georgeon & Marshall, in press), we implemented a learning algorithm to control agents confronted with EMDP problems in which regularities of interactions had a hierarchical sequential structure. A hierarchical structure of sequential regularities means that short sequences of interactions representing low-level regularities constitute subsequences of longer sequences of interactions representing higher-level regularities. In this case, the agent can start by discovering and mastering short regularities, then continue to learn higher-level regularities from sequences of lower-level regularities.

In the present article, we investigate a different class of problems designed to represent situations in which the coupling between the agent and the environment offers both hierarchical sequential regularities and spatial regularities. The next subsection presents such problems.

Spatial Enactive Markov Decision Process (SEMDP)

We formally define a Spatial Enactive Markov Decision Process (SEMDP) as an EMDP in which additional information σ_t and τ_t is provided to inform the agent about the spatial properties of the enacted interactions e_t. σ_t specifies a point in the space surrounding the agent where e_t can be approximately situated. In a two-dimensional environment, $\sigma_t \in \mathbb{R}^2$ represents the Cartesian coordinates of this point in the agent's egocentric referential. τ_t specifies a geometrical transformation that approximately represents the agent's movement in space resulting from the enaction of e_t. In a two-dimensional environment, $\tau_t = (\theta_t, \rho_t)$ with $\theta_t \in \mathbb{R}$ being the angle of rotation of the environment relative to the agent, and $\rho_t \in \mathbb{R}^2$ the two dimensional vector of translation of the environment relative to the agent. Figure 3 represents the SEMDP formalism.

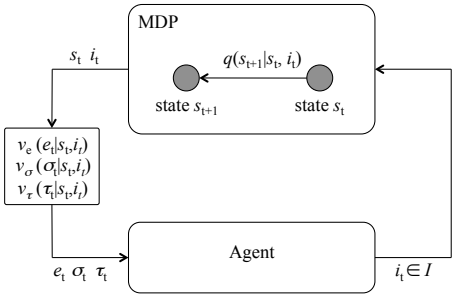


Figure 3: The Spatial Enactive Markov Decision Process (SEMDP) formalism. Compared to an EMDP (Figure 1), a SEMDP provides additional spatial information σ_t and τ_t when a primitive interaction e_t is enacted at time t. σ_t represents the position of the interaction e_t relative to the agent, and τ_t the spatial displacement of the agent generated by the enaction of e_t.

The intuition for σ_t is that the agent has sensory information available that helps it situate an interaction in space. For example, humans are known to use eye convergence, kinesthetic information, and interaural time delay (amongst other information) to infer the spatial origin of their visual, tactile, and auditory experiences. The intuition for τ_t is that the agent has sensory information available that helps it keep track of its own displacements in space. Humans are known to use vestibular and optic flow information to realize such tracking. In robots, σ_t and τ_t can be obtained through telemeters and accelerometers.

To replicate how humans and animals learn to infer spatial information from sensory inputs, σ_t and τ_t should ideally reflect sensory inputs from which spatial information could be inferred, rather than directly providing metric values of positions and displacements. Since, in the SEMDP formalism, σ and τ directly provide metric values, we acknowledge that SEMDPs do not allow studying such learning mechanisms. Instead, SEMDPs reduce the scope of research to studying how agents may use this spatial information, assuming it is available.

We propose SEMDP problems to study how agents learn the existence of physical entities from the experience of interacting with these entities in space. We call this problem the problem of *autonomous ontology construction*. The need for autonomous ontology construction is supported by pragmatic epistemology (e.g., Hume, 1739) that posits that the knowledge of objects is constructed through experience rather than given *a priori*. More specifically, some phenomenological philosophers (e.g., Merleau-Ponty, 1976) have suggested the idea that the knowledge of objects follows from the sense of space. We borrowed the term *bundle* from Hume's bundle theory of objects (Hume, 1739) to refer to the collection of interactions that are afforded by a type of entity present in the world. When a bundle of interactions consistently overlap in space, the agent infers the existence of a kind of entity that affords these interactions. We use the term *phenomenon* to refer to an instance of entity, in accordance with the general definition of this term as an *observable occurrence*. To be concrete, a physical object would be a phenomenon that is solid and persistent. We intend a SEMDP agent to learn to categorize the phenomena with which it can interact, according to the bundles of interactions that these phenomena afford.

The Enactive Cognitive Architecture (ECA)

Now that we have proposed a formalism to represent a spatio-sequential coupling between an agent and an environment (the SEMDP formalism), and have stated our objectives (designing agents that fulfill both their autotelic and interactional motivation when confronted with such a coupling), we can present the architecture that we designed to address this objective. Figure 4 gives an overview of this architecture, which we refer to as the Enactive Cognitive Architecture (ECA).

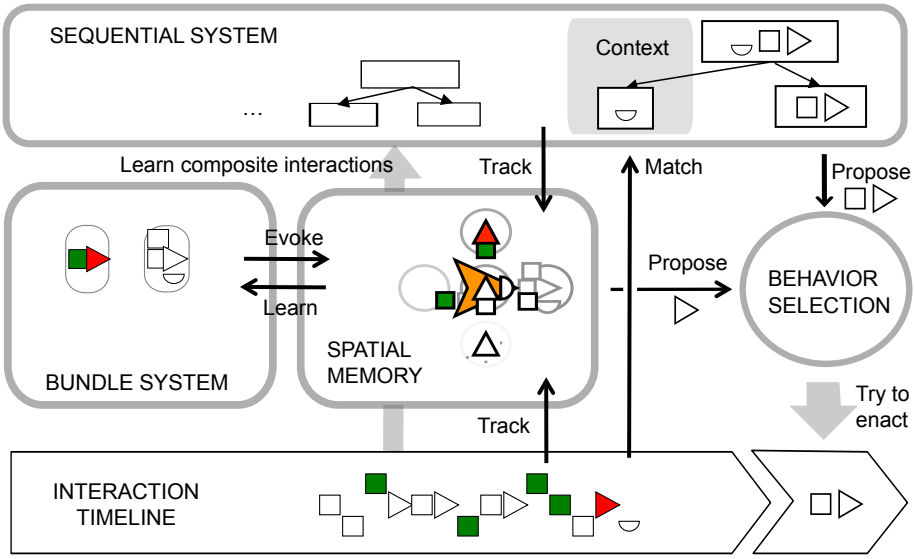


Figure 4 : The Enactive Cognitive Architecture (ECA). Interaction Timeline (bottom): stream of interactions enacted over time represented by symbols further described in Figure 7. Sequential System (top): learns hierarchical sequential regularities of interactions that can subsequently be enacted as a whole sequence. Spatial Memory (center): keeps track of the position (relative to the agent) of enacted interactions over the short term. Bundle System (left): records bundles of interactions based on their spatial overlap observed in spatial memory. Once constructed, bundles allow the evocation of phenomena in spatial memory. In turn, evoked phenomena propose the interactions that they afford. Behavior Selection (right): balances the propositions made by the sequential system and the spatial memory and selects the next sequence of interactions to try to enact.

ECA was built upon a previous algorithm implementing sensorimotor self-programming agents in hierarchical sequential EMDP problems (Georgeon & Ritter, 2012). This previous algorithm remains as a part of ECA in the form of what is now called the Sequential System (SS). The SS is responsible for the sensorimotor self-programming effect by learning hierarchical sequences of interactions that can subsequently be executed as a whole sequence. We begin the description of ECA with the SS because of this important role.

Sequential System (SS)

We define a composite interaction as a sequence of two interactions $i_c = \langle i_{pre}, i_{post} \rangle$, where i_{pre} and i_{post} may be primitive interactions or other composite interactions. We refer to i_{pre} as i_c's *pre-interaction*, also denoted $pre(i_c)$, and to i_{post} as i_c's *post-*

interaction, also denoted *post*(i_c). We define K_t as the set of composite interactions known by the agent at time t, and $J_t = I \cup K_t$ as the set of all interactions, primitive or composite, known by the agent at time *t*. Interactions in J_t are thus hierarchically organized in a pairwise manner, all the way down to primitive interactions. We define the *serialization* function *ser*: $K_t \rightarrow X_t$ such that *ser*(i_c) gives the serial interaction $i_s \in X_t$ (defined in previous section) that consists of the series of primitive interactions of $i_c \in K_t$. Trying to enact a composite interaction i_c consists of trying to enact *ser*(i_c) as defined previously. The agent uses the hierarchical structure of i_c to reconstruct the hierarchical structure of the actually enacted composite interaction $e_c \in K_t$ from e_s through a mechanism used in the previous version of the algorithm (Georgeon & Ritter, 2012). In fact, ECA agents do not actually record the set X_t; rather, K_t supersedes X_t as a hierarchically organized way of recording sequences of interactions. Figure 5 synthesizes the principles of the SS mechanism.

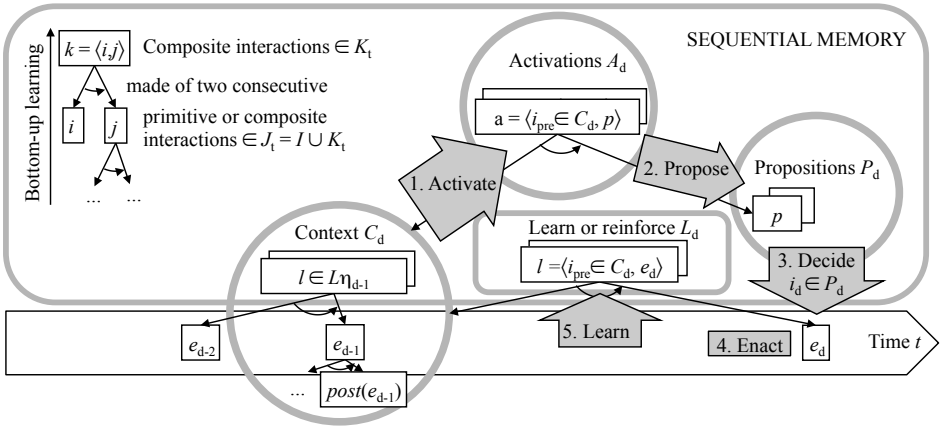


Figure 5: Schematic representation of the Sequential System (SS) at decision time t_d. e_{d-1} is the interaction (primitive or composite) enacted during the previous decision cycle t_{d-1}. The situation of the agent at t_d is represented by a set of interactions $C_d \subset J_d$ referred to as the *context*. Step 1: previously learned composite interactions whose pre-interaction belongs to the context are activated, forming the set $A_d \subset K_d$. Step 2: activated interactions in A_d propose their post-interaction for enaction, forming the set $P_d \subset J_d$. Propositions are weighted relative to the weight of the activated interactions. Step 3: the intended interaction i_d is selected from amongst the proposed interactions in P_d, based on the weight of the proposition and the values of the proposed interactions. Step 4: the agent tries to enact the intended interaction i_d, which results in the actually enacted interaction e_d. Step 5: new composite interactions are constructed or reinforced with their pre-interaction belonging to the context C_d and their post interaction being e_d, forming the set of learned or reinforced interactions L_d to be included in K_{d+1}. Step 6 (not represented in the figure): the context C_{d+1} is constructed to include *stabilized* interactions in L_d, e_d, and *post*(e_d) if it exists. The set of stabilized interactions $L\eta_d$ is the subset of interactions in L_d whose weight passed a fixed threshold η. Note that this mechanism does not depend on the length of sequence of the enacted interactions e_{d-2}, e_{d-1}, e_d. It can, therefore, apply recursively to learn increasingly higher-level composite interactions that capture longer sequential regularities of interaction.

By learning sequences of behaviors, the SS relates to adaptive history methods (e.g., Dutech, 2000; McCallum, 1996). However, it differs from these methods in that the selection of behavior is not driven by the search for rewarding states. Instead, the behavior selection mechanism (Step 3 in Figure 5) balances the various proposition weights and the motivational values of the proposed interactions *r*(*p*). The weight of a

proposition—based on the reinforcement value of the activated interactions—reflects the confidence that the agent has in the various regularities of interaction that match the context at time t_d. The selection mechanism thus results in the agent choosing the intended interaction i_d that offers the best balance between the expected value obtained if the enaction of i_d succeeds, and the alternate value $r(e_d)$ if it fails, as far at the agent can tell at time t_d.

Over time, C_d tends to represent the agent's situation in terms of the sequences of interactions that are the most representative of the situation at time t_d. This tendency of C_d to capture representative regularities is emergent in the sense that it is not directly specified by the algorithm but rather is observed in experiments. Notably, since the selected interaction i_d may contain subsequences with negative values, this mechanism does not drive the agent to the highest immediate value but rather allows the agent to enact subsequences of negative interactions to reach even greater positive interactions. The agent can also avoid immediate positive interactions that likely would lead subsequently to even more negative interactions.

The SS can work autonomously in the absence of the other elements of ECA. An agent solely equipped with the SS, however, can only learn to master sequential regularities of interactions, and is unable to handle the spatial information available in SEMDP problems. The effect of the SS alone was demonstrated in several examples of hierarchical sequential EMDP problems: the Small Loop Problem presented in the next section, other forms of loop-shaped grid environments (Georgeon & Ritter, 2012), an environment that provides the agent with a rudimentary visual system (Georgeon, Cohen, & Cordier, 2011), and a continuous two-dimensional environment (as opposed to a discrete grid) (Georgeon & Sakellariou, 2012). We refer the reader to these articles for a comprehensive description of this algorithm and for example analyses of the resulting agent behavior. Interactive demonstrations are also available online[1].

Spatial system

We define spatial memory as a set of *places* where primitive interactions were enacted. A place γ is defined as $\gamma = (e, \lambda)$ with $e \in J_t$, and λ being a location in space defined by its Cartesian coordinates relative to the agent. When the agent enacts primitive interaction e_p, a place $\gamma = (e_p, \lambda)$ is added to spatial memory with λ being initialized to the position σ where e_p was enacted. If e_p is the last primitive interaction of an enacted composite interaction e_c then another place $\gamma_d = (e_c, \lambda)$ is added to spatial memory to also keep track of the enaction of e_c. Subsequently, the transformation τ that resulted from the enaction of e_p is applied to all places in spatial memory, meaning that spatial memory keeps track of the places where interactions were enacted relative to the agent's position as the agent moves. Generally, the location σ and transformation τ could be imprecise and noisy, causing the position of interactions to become unreliable after several displacements. To reduce spurious behavior due to imprecision in spatial memory, we implemented decay in spatial memory so that older places would be removed after several interaction cycles. The agent, therefore, does not construct a map of the environment; rather, it uses spatial memory only to detect spatial overlaps of interactions in its surrounding local space over the persistence time of spatial memory.

[1] http://e-ernest.blogspot.fr/2012/03/small-loop-challenge.html

In the experiment presented next, the persistence in spatial memory was set to 10 time steps.

The agent learns *bundles* of interactions when the enaction of interactions overlaps in space. Bundles are defined as sets of interactions $b \subset J_t$. The experimental environment presented in the next section offers two types of phenomena: *walls* and *empty cells*. In this case, we expect two bundles to be constructed: the bundle that gathers the interactions afforded by walls and the bundle that gathers the interactions afforded by empty cells. Environments like this one, which only contain phenomena affording mutually exclusive bundles, allow a simplification of the bundle construction algorithm. Table 1 reports the simplified bundle construction algorithm currently implemented in ECA. In the general case, we expect more difficulties to arise in representing different kinds of objects that may afford some interactions in common.

Table 1: Simplified bundle construction algorithm.

```
when a new place γ = (e, λ) is added to spatial memory
   for each place γj =(ej, λj) previously in spatial memory at the same location as γ (i.e., λj = λ)
      if there exists a bundle b to which ej already belongs, then add e to b
      else, create bundle b = {e, ej}
check the bundle memory and merge bundles that share a common interaction
```

Bundles are constructed gradually and are merged when they have an interaction in common. For example, the agent may first learn to represent empty cells by bundle $b_0 = \{i_1, i_7\}$, then learn bundle $b_1 = \{i_4, i_7\}$; the simplified algorithm assumes that b_0 and b_1 represent the same kind of phenomenon because both bundles afford i_7, therefore, b_0 and b_1 are merged to form bundle $\{i_1, i_4, i_7\}$. Another limitation is that this algorithm is not resistant to large errors in σ and τ that generate erroneous overlaps of interactions, which may result in erroneous bundle construction. To address such cases, future versions of ECA should allow the agent to eliminate erroneous bundles.

In addition to the Sequential System, spatial memory also proposes interactions that are activated by the evocation of a phenomenon in the surroundings of the agent. Table 2 reports the phenomenon evocation algorithm.

Table 2: Phenomenon evocation algorithm.

```
// Enacted interactions evoke the phenomena that afford them
for each place γ = (e, λ) in spatial memory
   for each bundle b that contains interaction e
      add a "phenomenon place" in spatial memory ϕ = (b, λ) if it does not yet exist
// Evoked phenomena propose to enact the interactions that they afford
for each "phenomenon place" ϕ = (b, λ) evoked in spatial memory
   for each interaction e ∈ b
      if λ = σ(e)
         Generate a proposition to enact e with weight r(e) × CONST
```

Over time, certain interactions evoke certain phenomena, and evoked phenomena prompt the agent to enact the other interactions that they afford if the phenomenon's position matches the interaction's position relative to the agent. The weight of the proposition is proportional to the interaction's value. For example, when the agent recognizes an empty cell through feeling, this empty cell incites the agent to move towards it because moving to an empty cell has high value. Conversely, walls dissuade

the agent from moving towards them because bumping has a negative value. Bundles not only contain primitive interactions but may also contain composite interactions, which allows the agent to associate sequences of behaviors with types of phenomena (e.g., in the run reported in the experiment, the agent learns to turn and move forward when it recognizes an empty cell on its side). Because bundles may contain learned sequences of interactions, they support the agent's self-programming and thereby contribute to the agent's constitutive autonomy.

Behavior selection mechanism

On each decision cycle, the Sequential System proposes interactions based on sequential regularities, and the spatial system proposes interactions based on spatial regularities. The behavior selection mechanism selects the interactions with the highest cumulative proposition weight. Once an interaction is selected, the agent tries to enact it.

CONST (in Table 2) is a constant of proportionality that balances the weight of the spatial memory relative to the weight of the sequential system. This constant was adjusted empirically. We chose CONST = 10, meaning that a proposition generated by the spatial memory had the same weight as a proposition generated by the sequential system based on an activated interaction that had been reinforced 10 times. Automatically balancing the spatial and sequential systems would probably require more complex underlying mechanisms that are still open to research. We envision implementing spatio-sequential simulation of behavior in future versions of ECA, using additional modules perhaps inspired by the hippocampus.

Currently, ECA represents the agent's context as the union of the Sequential System context C_d and of the Spatial Memory. This context can be thought of as the agent's perception of its environment at time t_d, considering perception as an internal construct elaborated through interaction. This understanding of perception follows Gibson's (1977) idea that the agent perceives the world as possibilities of interaction called *affordances*. This context constitutes a directly actionable representation of the situation as opposed to a "Cartesian representation" that would require subsequent interpretation (Dennett, 1991). It also relates to the theory of enaction because the agent perceives its environment in a way that depends on the agent's individual previous experiences interacting with it. Indeed, both the sequential and the spatial contexts contain previously learned composite interactions, meaning that two different instances of agents will not "see" the same situation in the same way, depending on their previous experience.

This behavior selection mechanism can be compared to the operator selection mechanism in Soar 9 (Laird & Congdon, 2009). Soar 9 supports reinforcement learning: rules that match the context create weighted proposals for operators, and the highest-weighted operator is selected for firing. There are, however, two main differences: (a) ECA's context matching involves temporal pattern matching (over arbitrarily long sequences of interactions) rather than instantaneous context matching; (b) since proposed interactions can also be arbitrarily long sequences, the decision engages the agent for several subsequent time steps rather than only the next time step. Moreover, this mechanism contrasts with symbolic modeling as it is typically done in rule-based architectures (Newell & Simon, 1975) by the fact that the behavior selection mechanism does not involve a predefined semantics associated with symbols by the modeler.

Experiment

We demonstrate ECA using a benchmark proposed previously: the Small Loop Problem (SLP) shown in Figure 6 (Georgeon & Marshall, in press).

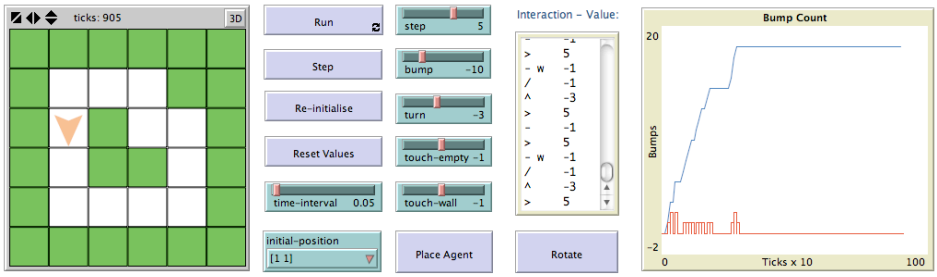


Figure 6. The Small Loop Problem (SLP) in NetLogo. The environment (left) is a loop of white squares surrounded by green walls. The brown arrowhead represents the agent. The agent can try to move one cell forward, turn to the left or to the right, feel in front, to the left or to the right, but it ignores the meaning of interactions. The experimenter can preset the values of the primitive interactions using the slider controls (center). The *Interaction-Value* window shows a trace of ASCII codes representing the primitive interactions enacted by the agent over time next to their values. The *Bump Count* graph (right) displays the number of times the agent bumps into a wall (cumulative total in blue), showing that the agent gradually learns to avoid bumping into walls. When the agent touches/feels a cell, the cell flashes yellow, and when the agent bumps into a wall, the wall flashes red, making the agent's behavior intelligible to the experimenter.

The SLP was originally used as a hierarchical sequential EMDP problem in which hierarchical sequential regularities of interaction were induced by the loop-shaped pathway that constrained the agent's behavior. Now, we use the two-dimensional spatial structure to provide the agent with additional spatial information. Since ECA agents exploit spatial information, we expect them to perform better than SS-only agents. ECA agents, however, still have to learn the meaning of interactions and to discover that certain sets of interactions are consistently afforded by certain categories of phenomena present in the environment. The possibilities of interactions are summarized in Figure 7.

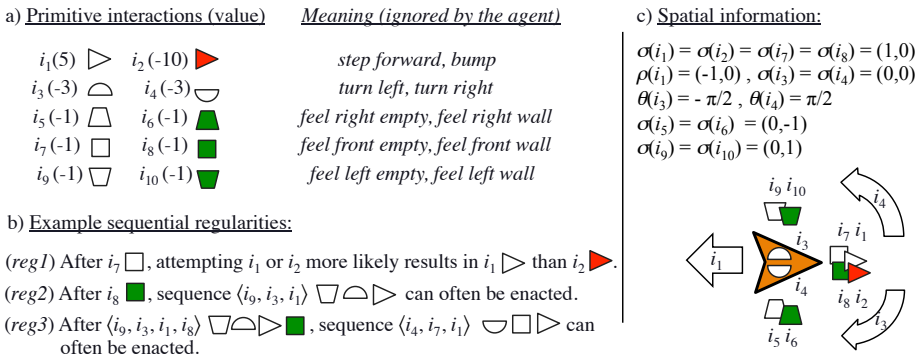


Figure 7 : Interactions offered by the Small Loop Problem modeled as a Spatial Enactive Markov Decision Process. a) The agent has 10 primitive interactions at its disposal but ignores their semantics. Each primitive interaction has a predefined value (in parentheses) set by the experimenter. b) The coupling offers hierarchical regularities of interactions. For example, we expect the agent, in discovering and exploiting (*reg1*), to choose interaction i_7, and if this effectively results in i_7, to subsequently choose i_1 so as to safely

enact i_1 which has a positive value, thus avoiding i_2 which has a very negative value. Sequential regularities have a hierarchical structure: $\langle i_9, i_3, i_1 \rangle$ in (*reg2*) is a subsequence of the (*reg3*) sequence $\langle i_9, i_3, i_1, i_8 \rangle$. c) For each enacted interaction e, the agent receives the position $\sigma(e)$ in an egocentric referential, and the transformation $\tau(e)$ consisting of the translation $\rho(e)$ and the rotation $\theta(e)$ (represented by arrows when non-zero). Interactions that are afforded by empty cells are represented in white and interactions that are afforded by walls are represented in green and red; the agent originally ignores this distinction and must learn that some interactions inform it about the presence of phenomena in its surrounding space, while simultaneously learning to categorize these phenomena.

The environment is deterministic, meaning that the corresponding probability distributions q and v presented in Figure 1 implement no stochasticity. The agent is nonetheless confronted with uncertainty because it cannot initially predict the consequences of its intended interactions until it starts learning the regularities afforded by the environment. For example, when circling the loop counterclockwise, if the agent feels a wall in front, it can often feel an empty cell to the left, but not always. The agent's algorithm is also deterministic, meaning that two runs lead to the same behavior. Different behaviors can nonetheless be observed by starting the agent from different initial positions.

Results show that the agent generally learns to avoid bumping into walls by adopting the behavior of feeling in front before trying to move forward within a hundred steps. Then, when the agent feels a wall in front, it progressively learns to feel to the side before deciding on which direction to turn. This behavior generally leads the agent to start to indefinitely circle the loop after approximately 150 steps. Figure 8 presents the trace of an example run. A video is available online that shows the entire run, the sequential trace, and the content of the spatial memory dynamically[2].

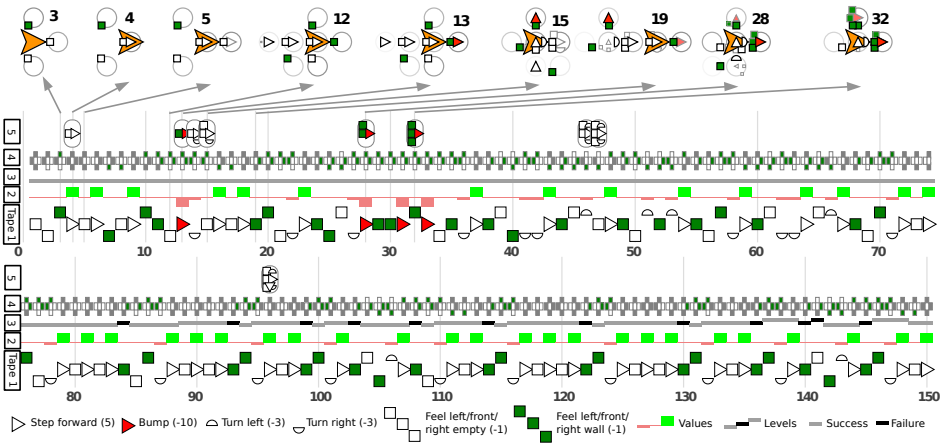


Figure 8: First 150 steps of an example trace of an ECA agent in the SEMDP version of the SLP. Tape 1 represents the primitive interactions enacted over time with the same symbols as in Figure 7 except that *feel to the sides* are represented by squares above (left) and below (right) the axis rather than trapezoids. Tape 2 represents the values of the enacted primitive interactions as a bar graph (green when positive, red when negative). Tape 3 represents the level of the enacted composite interaction in the hierarchy (gray when the primitive enaction was successful, black when it failed, thus interrupting the decision cycle). Tape 4 represents the four adjacent cells in the agent's spatial memory (cells whose content is unknown to the agent are gray). Tape 5 represents the construction of bundles over time (gray rounded rectangles that contain

[2] http://e-ernest.blogspot.fr/2012/04/ernest-112.html

interactions). The top of the figure shows snapshots of the agent's spatial memory at different steps. Gray circles represent bundles localized in spatial memory. These circles are faded to represent decay in spatial memory. This trace shows that the agent bumped only four times (red triangles on steps 13, 28, 31, and 33). The agent enacted more consistently positive interactions from step 70 on (less red in Tape 2). The agent started to try to exploit the composite interaction *feel front empty – step forward* for the first time on step 84, but the enaction of this interaction was interrupted due to an unexpected feeling of a wall (black second-level segment in Tape 3). The agent successfully enacted this sequence as a whole for the first time on steps 89-90 (gray second-level segment in Tape 3). It enacted the third-level sequence *feel left wall – turn right – step forward* for the first time on steps 137-139 (gray third-level segment in Tape 3).

In this run, the first instance of bundle construction occurred on step 4. On steps 1 to 4, the agent felt the three cells surrounding it, then stepped forward. Because the *feel front empty* on step 1 and the *step forward* on step 4 overlapped in space, these interactions were bundled together, to initiate the *empty cell bundle* (rounded rectangle on step 4, Tape 5). On step 5, a new *feel front empty* evoked the *empty cell bundle* in front of the agent. The interaction *step forward*, now belonging to this bundle, generated additional positive weight to step forward again.

In a similar way, the interaction *feel front wall* and *bump* were bundled together on step 13. On step 19, the interaction *feel front wall* evoked the newly-created *wall bundle* in front of the agent. The *bump* interaction, now belonging to the *wall bundle*, generated negative support for trying to step forward, preventing the agent from bumping into the wall. On step 96, the learned composite interaction *turn right – step forward* was added to the *empty cell bundle*, which led the agent to subsequently enact this sequence when an empty cell was again felt on the right. Note that this behavior does not rest on the construction of a map of the environment but on the fact that the agent learns to recognize surrounding phenomena through feeling interactions. Once this behavior was learned, the agent engaged in indefinite tours of the loop. The learning was improved by the ECA architecture: it involved only 4 bumps and took 150 steps as compared to 18 bumps and 300 steps with the SS-only algorithm (Georgeon & Marshall, in press).

This experiment illustrates how the agent categorized two types of phenomena afforded by the environment: the walls and the empty cells, and simultaneously learned to adapt its behavior to these phenomena. This result constitutes a starting point in addressing the autonomous ontology construction problem.

Notably, the current bundle construction mechanism works passively with regard to the agent's motivation. In future versions, we anticipate implementing other forms of motivation to incite the agent to actively try different possibilities of interaction afforded by different types of phenomena.

Conclusion

We have simultaneously introduced: (a) a new approach to model an agent interacting with an environment while keeping perception and action embedded (the EMDP and SEMDP formalisms); (b) an approach to self-motivation based on an association of autotelic motivation and interactional motivation; (c) a new cognitive architecture (ECA) to control an agent that learns to fulfill its autotelic and interactional motivation; and (d) a way to assess the agent's learning through behavioral analysis.

We report experiments that show that certain interactions (e.g., *feel*) become meaningful to the agent because it learns to use them to inform its future behavior. This result demonstrates that the agent learns to perform active perception, that is, the agent

actively uses certain interactions as a form of perception to inform its knowledge of the current situation. Additionally, the agent addresses the autonomous ontology construction problem at a rudimentary level. It learns to actively distinguish between two types of phenomena afforded by its environment and to cope with these phenomena by successfully enacting learned sequences of interactions (Figure 8).

In the description of the architecture, we point out many questions that remain to be addressed in moving towards more sophisticated agents confronted with couplings that offer more complex spatio-sequential regularities of interaction. In its current version, we acknowledge that ECA relies upon too many hard-coded functions, which should ultimately be removed in order to provide the agent with more flexibility to scale up to more complex environments. Some of these functions should be autonomously constructed by the agent, which would leave room for even more constitutive autonomy.

In spite of its current limitations, we believe that ECA offers a useful framework in which to study and advance the theory of enaction for the following reasons: (a) ECA uses sensorimotor schemes as the atomic elements of cognition rather than separating perception and action. (b) ECA supports studying how the agent constructs its own ontology of the environment from its experience interacting with it, in sharp contrast to traditional rule-based cognitive architectures that require the modeler to specify the semantics of symbols, which amounts to defining the ontology of the environment *a priori*. (c) ECA allows implementing self-motivation in the agent. In the future, we envision implementing other behavior-selection mechanisms to generate additional forms of motivation such as curiosity. (d) ECA allows the agent to program itself by learning a series of sensorimotor interactions and executing them as a single composite interaction. Self-programming allows constitutive autonomy, which theoreticians of enaction have identified as an important requirement for autonomous sense-making and intrinsic teleology.

Acknowledgement

This work was supported by the French *Agence Nationale de la Recherche* (ANR) contract ANR-10-PDOC-007-01. We gratefully thank Agnar Aamodt and Frank Ritter for their useful comments on this article.

References

Albus, J. S. (1993). A reference model architecture for intelligent systems design. In P. J. Antsaklis and K. M. Passino (Eds.), *An introduction to intelligent and autonomous control* (Chapter 2, pp. 27-56). Kluwer Academic Publishers.

Anderson, M. (2003). Embodied cognition: A field guide. *Artificial Intelligence*, *149*, 91–130.

Berthouze, L. & Ziemke, T. (2003). Epigenetic robotics—modelling cognitive development in robotic systems. *Connection Science*, *15*(4), 147–150.

Berthoz, A. (1997). *Le sens du mouvement*. Paris: Odile Jacob.

Blank, D.S., Kumar, D., Meeden, L., & Marshall, J. (2005). Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture. *Cybernetics and Systems, 32*(2), 125-150.

Brooks, R. A. (1991). New Approaches to Robotics. *Science, 253*, 1227–1232.
Chemero, A. & Turvey, M. (2007). Gibsonian Affordances for Roboticists. *Adaptive Behavior*, *15*(4), 473–480.
Csikszentmihalyi, M. (1990). *Flow. The Psychology of Optimal Experience*. New York: Harper and Row.
Dennett, D. (1991). *Consciousness explained*. New York: The Penguin Press.
Dreyfus, H. (2007). Why Heideggerian AI failed and how fixing it would require making it more Heideggerian. *Philosophical Psychology, 20*(2), 247-268
Dutech, A. (2000). Solving POMDPs using selected past events. In proceedings of European Conference on Artificial Intelligence (ECAI-2000), Berlin, pp.281-285.
Froese, T. & Ziemke, T. (2009). Enactive artificial intelligence: Investigating the systemic organization of life and mind. *Artificial Intelligence, 173*(3-4), 466–500.
Georgeon, O. & Marshall, J. (in press). Demonstrating sensemaking emergence in artificial agents: A method and an example. *International Journal of Machine Consciousness*.
Georgeon, O., Marshall, J., & Gay, S. (2012). Interactional motivation in artificial systems: between extrinsic and intrinsic motivation. In proceedings of the 2nd International Conference on Development and Learning and on Epigenetic Robotics (EPIROB2012), San Diego, pp. 1-2.
Georgeon, O. & Ritter, F. (2012). An intrinsically-motivated schema mechanism to model and simulate emergent cognition. *Cognitive Systems Research, 15-16*, 73-92.
Georgeon, O., Cohen, M., & Cordier, A. (2011). A model and simulation of early-stage vision as a developmental sensorimotor process. In proceedings of the Conference on Artificial Intelligence Applications and Innovations (AIAI), Corfu, Greece, pp. 11-16.
Georgeon, O. & Sakellariou, I. (2012). Designing Environment-Agnostic Agents. In proceedings of the Adaptive Learning Agents workshop (ALA), at the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Valencia, Spain, pp. 25-32.
Gibson, J. (1979). *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin.
Gibson, J. (1977). The theory of affordances. In R. E. Shaw, and J. Bransford (Ed.) *Perceiving, Acting, and Knowing*. Hillsdale, NJ: Lawrence Erlbaum Associates.
Gross, C. & Graziano, M. (1995). Multiple representations of space in the brain. *The Neuroscientist, 1*(1), 43-50.
Hirose, N. (2002). An ecological approach to embodiment and cognition. *Cognitive Systems Research, 3*, 289–299.
Holland, O. (2004). The Future of Embodied Artificial Intelligence: Machine Consciousness? In F. Iida (Ed.), *Embodied Artificial Intelligence* (pp. 37–53). Berlin: Springer.
Hume, D. (1739). *A treatise of human nature*. Oxford University Press.
Hurley, S. (1998). *Consciousness in action*. Cambridge, MA: Harvard University Press.
Kaelbling, L., Littman, M., & Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101*, 99-134.
Laird, J. & Congdon, C. (2009). The Soar User's Manual Version 9.1, University of Michigan.
Lungarella, M., Metta, G., Pfeifer, R., & Sandini, G. (2003). Developmental robotics: a survey. *Connection Science*, *15*(4), 151–190.

McCallum, A. (1996). Learning to use selective attention and short-term memory in sequential tasks. In proceedings of the Fourth International Conference on Simulating Adaptive Behavior.

Merleau-Ponty, M. (1976). *Phénoménologie de la perception*. Paris: Gallimard.

Newell, A. & Simon, H. 1976. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM, 19*(3), 113-126.

O'Regan, K. (2012). How to Build a Robot that is Conscious and Feels. *Minds and Machines*, *22*(2), 117–136.

O'Regan, K., & Noë, A. (2001). A sensorimotor account of vision and visual consciousness, *Behavioral and Brain Sciences, 24*(5), 939–1031.

Oudeyer, P.-Y., Kaplan, F., & Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation, 11*(2), 265-286.

Paul, C. (2006). Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, *54*, 619–630.

Pfeifer, R. (1999). *Understanding Intelligence*. Cambridge, MA: MIT Press.

Pfeifer, R. & Bongard, S. (2006). *How the body shapes the way we think: A new view of intelligence*. Cambridge, MA: MIT Press.

Pfeifer, R. & Scheier, C. (1994). From perception to action: The right direction? In P. Gaussier and J.-D. Nicoud (Eds.), *From Perception to Action* (pp. 1-11). IEEE Computer Society Press.

Piaget, J. (1951). *The psychology of intelligence*. London: Routledge and Kegan Paul.

Schmidhuber, J. 2010. Formal theory of creativity, fun, and intrinsic motivation. *IEEE Transactions on Autonomous Mental Development, 2*(3), 230-247.

Shanahan, M. (2010). *"Embodiment and the Inner Life," Cognition and Consciousness in the Space of Possible Minds*. Oxford: Oxford University Press.

Singh, S., Barto, A., & Chentanez, N. (2005). Intrinsically motivated reinforcement learning. In L. K. Saul, Y. Weiss, & L. Bottou (Eds), *Advances in Neural Information Processing Systems* (pp. 1281-1288). Cambridge, MA: MIT Press.

Steels, L. (2004). The Autotelic Principle. In I. Fumiya, R. Pfeifer, L. Steels, & K. Kunyoshi (Eds), *Embodied Artificial Intelligence* (pp. 231-242), Springer Verlag.

Sun, R. (2004). Desiderata for cognitive architectures. *Philosophical Psychology*, 17(3), 341-373.

Sun, R. & Giles, C. L. (2000). *Sequence learning - Paradigms, algorithms, and applications*. Berlin Heidelberg: Springer.

Sutton, R. & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence, 112*, 181-211.

Varela, F., Thompson, E., & Rosch, E. (1991). *The embodied mind: Cognitive science and human experience*. Cambridge, MA: MIT Press.

Ziemke, T. (2001). The construction of reality in the robot: Constructivist perspective on situated artificial intelligence and adaptive robotics. *Foundations of Science, 6*, 163–233.

Zlatev, J. (2001). The Epigenesis of Meaning in Human Beings, and Possibly in Robots. *Minds and Machines*, *11*, 155–195.