Inverting the Interaction Cycle to Model Embodied Agents

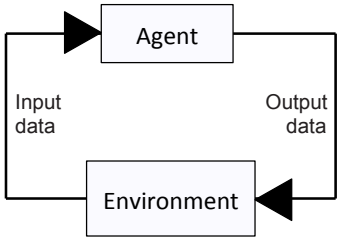
Olivier L. Georgeon[1] and Amélie Cordier[1]
[1]Université de Lyon, CNRS.
Université Lyon 1, LIRIS, UMR5205, F-69622, France.
olivier.georgeon@liris.cnrs.fr, amelie.cordier@liris.cnrs.fr

Abstract
Cognitive architectures should make explicit the conceptual begin and end points of the agent/environment interaction cycle. Most architectures begin with the agent receiving input data representing the environment, and end with the agent sending output data. This paper suggests inverting this cycle: the agent sends output data that specifies an experiment, and receives input data that represents the result of this experiment. This complies with the embodiment paradigm because the input data does not directly represent the environment and does not amount to the agent's perception. We illustrate this in an example and propose an assessment method based upon activity-trace analysis.

Keywords: Embodiment, perception-action, cognitive modeling, intrinsic motivation, activity trace.

1 Introduction

Cognitive architectures and machine-learning models generally represent the interaction between the agent and the environment as a cycle in which the agent alternatively receives *input data* from the environment and sends *output data* to the environment. Figure 1 depicts this cycle.



The agent receives input data (left arrow) from the environment and sends output data (right arrow) to the environment. The cycle rotates indefinitely; the figure does not show when the cycle begins and when it ends.

Figure 1: interaction cycle between an agent (top) and an environment (bottom).

The model in Figure 1 implies no particular commitment about the nature of the input and output data. Most models, however, make an additional commitment: they arrange the input data so that it *represents* the environment, and they implement the agent's algorithm so that it exploits this assumption. The term *represent* is used here in its etymological sense of "making present again". That is, traditional models use the input data as a *representative* sent to the agent by the environment, as if the input data made the environment's state accessible to the agent, at least partially and possibly blend with noise. Section 2 develops this argument by analyzing symbolic cognitive models (e.g., [1]) and reinforcement learning models as they are typically implemented in Partially Observable Markov Decision Processes (POMDPs, e.g., [2]).

There exist other possibilities than designing input data to represent the environment. A typical alternative has been offered by cybernetic control theory (e.g., [3]) in which the input data was a *perturbation* of a control loop. More recently, some authors have advocated an *inversion of the perception-action cycle* (e.g., [4]). Inverting the interaction cycle allows modeling the input data so that it does not directly represent the state of the environment. Instead, the input data can represent the result of an experiment initiated by the agent. In the same environment's state, different experiments may produce different results; the result itself thus does not represent the environment's state, not even partially or with noise. This paper follows this idea to propose the Experiment-Result Cycle (ERC). Section 3 examines it further.

A key concept of the embodiment paradigm (e.g., [5]) is that the agent is not a passive observer of reality but rather constructs a perception of reality through active interaction ("perception and action arise together, dialectically forming each other", [6] p5). This implies that the model should derive perception as a secondary construct resulting from experience of interaction, rather than considering the input data as the agent's perception. The term *embodiment* means that the agent must be a part of reality for this active interaction to happen.

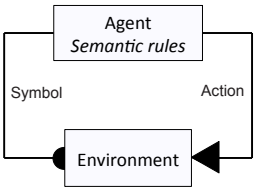
The ERC ensures that the input data is not considered as the agent's perception precisely because the input data does not represent the state of the environment. For this reason, we propose the ERC as a possible starting point to model agents according to the embodiment paradigm.

Since ERC agents do not access the environment's state, the designer cannot program them to seek a particular environment's state as a goal. Therefore, we cannot assess ERC models by measuring their performance in reaching goal states. ERC models thus require another validation paradigm. This requirement has been frequently raised in the literature of embodied robotics (e.g., [7]), intrinsic motivation (e.g., [8]), and developmental learning (e.g., [9]).

We suggest using a validation paradigm similar to that used to assess natural cognitive systems (animals): behavioral analysis (e.g., [10]). This paradigm requires the embodied-artificial-intelligence scientific community to find a consensus on how to qualify *cognitive behaviors*. To contribute to this endeavor, Section 3 gives an initial example of behavioral analysis of a simple ERC algorithm.

2 When does the cycle begin?

Figure 2 illustrates *symbolic cognitive models* (e.g., [1]) by making explicit the conceptual begin and end points of the interaction cycle.



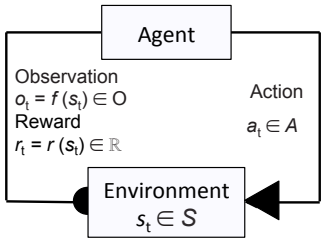
The black circle represents the begin point: a symbol is passed from the environment to the agent. The agent interprets this symbol according to semantic rules, and decides on an action to carry out in the environment. The black triangle represents the end of the cycle when the environment receives the action chosen by the agent.

Figure 2: the symbolic cognitive cycle.

In symbolic cognitive models, the agent's input data is *symbolic* in the sense that it symbolizes elements of the environment and it is interpreted according to *semantic rules* implemented in the agent. For example, we can model an animal eating apples using the rules ⟨if *input* = "apple" then *input* is edible⟩ and ⟨if *input* is edible then *action* = "eat" ⟩. Since the input symbol represents the environment's state, we can consider that the cycle begins with reading the input symbol, and ends with sending the action. Thus, Figure 2 represents symbolic models more precisely than Figure 1 because it makes explicit the begin point (black circle) and end point (black triangle) of the interaction cycle.

Symbolic models are typically used for problem solving. In this case, the designer specifies the *goal state* through rules that check for goal completion, and specifies heuristics to reach the goal through a set of rules. If the input symbol did not represent elements of the environment's state, it would be impossible to specify the goal completion rules and the heuristic rules. This validation paradigm could not apply.

Figure 3 illustrates a Partially Observable Markov Decision Processes (POMDPs, e.g., [2]) by making explicit the conceptual begin and end points of the interaction cycle.



The environment is modeled as a set of states S. At the beginning of cycle t (black circle), the agent receives an observation o_t computed as a function of the state $f(s_t)$, and a reward $r(s_t)$ associated with the state s_t resulting from the previous action a_{t-1}. At the end of cycle t (black triangle), the agent sends the action a_t to the environment. This action swaps the environment from s_t to s_{t+1}.

Figure 3: the POMDP cycle.

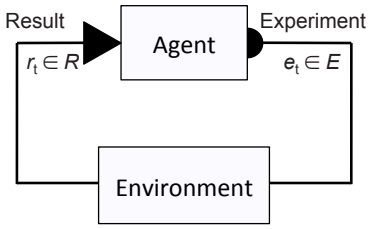
The agent's input data is called *observation* and is typically computed as a function of the state: $o_t = f(s_t)$. In contrast with symbolic models, the observation is not *symbolic* in the sense that it does not match semantic rules in the agent. Yet, it still constitutes a *representation* of the environment since it is a function of the environment's state. The function $f(s)$ may incorporate stochastic noise but the observation still represents the environment statistically, as if it was blended with noise. Since the observation is a function of the environment's state only, we can consider that the cycle begins with the observation and ends with the action. Figure 3 makes this explicit with the black circle and triangle.

The agent additionally receives a scalar reward $r(s_t)$ computed as a function of the environment's state, which defines the agent's goal. If the reward was not associated with the environment's state, the agent's goal could not be specified as a state to reach.

Note that some variations of POMDPs have been proposed in which the scope of the observation depended on the previous action, thus involving a form of active perception (e.g., [11]). However, the algorithms still assumed that the observation reflected the state of the environment (as if the environment was observed through a filter that varied with the action), and still sought reward states on the basis of the reward associated with states.

3 The Experiment-Result Cycle

In contrast with both symbolic models and reinforcement-learning models, the Experiment-Result Cycle conceptually begins with the agent sending output data, and ends with the agent receiving input data. This is illustrated in Figure 4.



At the beginning of interaction cycle t (black circle) the agent chooses an experiment e_t from amongst the set E of experiments at its disposal. At the end of interaction cycle t (black triangle), the agent receives the result r_t of the experiment. The same state of the environment may generate different results depending on the experiment.

Figure 4: the Experiment-Result Cycle.

In the ERC, the agent's input (called result r_t in Figure 4) is designed as a result of an experiment initiated by the agent. The agent does not have access to the environment's state, neither through an input symbol, an observation, or a reward value. There is no data in the agent connected with the environment's state that would offer a grip for designing an algorithm that would seek goal states. This is acceptable within the embodiment paradigm because we expect embodied agents to engage in incremental and open-ended learning rather than learning to reach predefined goal states. The environment may even not be modeled as a set of states, as in the example below, and as the real world.

Some authors in the domain of *intrinsic motivation* have already addressed the problem of implementing motivational principles without a reference to external criteria (e.g., [12, 8, 13]). Here, we report a simple algorithm in which the agent is considered *pleased* when it correctly anticipated the result of an experiment, and *pained* otherwise. This is a minimalist version of the motivational principle of becoming *in control* of one's activity, which Steels [14] called the *autotelic principle*. Additionally, our agent is *bored* when it has been *pleased* for too long, which causes it to change experiment, even though it may not be able to correctly anticipate the result. Table 1 (left) presents this algorithm.

```
01   experiment = e1
02   Loop(cycle++)
03      if (mood = BORED)
04         pleasedDuration = 0
05         experiment = pickOtherExperiment(experiment)
06      anticipatedResult = anticipate(experiment)
07      if (experiment = e1)
08         result = r1
09      else
10         result = r2
11      recordTuple(experiment, result)
12      if (result = anticipatedResult)
13         mood = PLEASED
14         pleasedDuration++
15      else
16         mood = PAINED
17         pleasedDuration = 0
18      if (pleasedDuration > 3)
19         mood = BORED
20      print cycle, experiment, result, mood
```

```
00: e1r1 PAINED
01: e1r1 PLEASED
02: e1r1 PLEASED
03: e1r1 PLEASED
04: e1r1 BORED
05: e2r2 PAINED
06: e2r2 PLEASED
07: e2r2 PLEASED
08: e2r2 PLEASED
09: e2r2 BORED
10: e1r1 PLEASED
11: e1r1 PLEASED
12: e1r1 PLEASED
13: e1r1 BORED
14: e2r2 PLEASED
15: e2r2 PLEASED
16: e2r2 PLEASED
17: e2r2 BORED
18: e1r1 PLEASED
19: e1r1 PLEASED
```

Table 1: Left: algorithm of a minimalist ERC system. Right: trace of the first twenty interaction cycles.

Table 1 (left), Lines 03 to 05: if the agent is *bored* then it picks another experiment arbitrarily from amongst the predefined list of experiments at its disposal. Line 06: the *anticipate*(*experiment*) function searches memory for a previously learned tuple that matches the chosen experiment, and returns its result as the next anticipated result. Lines 07 to 10 implement the environment: e_1 always yields r_1, and other experiments always yield r_2. Line 11: the agent records the tuple ⟨experiment, result⟩ in memory. Lines 12 to 17: if the result was anticipated correctly then the agent is *pleased*, otherwise it is

pained. Lines 18 and 19: if the agent has been *pleased* for too long (arbitrarily 3 cycles) then it becomes *bored*. Table 1 (right) shows the activity trace generated by this algorithm.

On Cycle 00, the agent chose experiment e_1 and received result r_1. It was *pained* because it could not anticipate this result. It recorded the tuple $\langle e_1, r_1 \rangle$. On Cycle 01, it chose e_1 again, anticipated r_1 on the basis of the previous experience, got r_1, and was *pleased* because the anticipation was correct. It kept e_1 during the next two cycles. On Cycle 04, it got *bored* because it had been *pleased* for too long. On cycle 05, it changed experiment to e_2 because it was bored. It was, however, unable to anticipate the result because it never tried e_2 before. It received r_2, was *pained* because it could not anticipate it, and learned the tuple $\langle e_2, r_2 \rangle$. On Cycle 09, it got *bored* again. On Cycle 10, it changed back to e_1, and anticipated result r_1 based on the experience gained on Cycle 00; it indeed received r_1, and was *pleased* because its anticipation was correct.

We reported this analysis to exemplify the validation paradigm that we propose: demonstrating cognitive behaviors using activity traces. This behavior is very rudimentary; this agent would be unlikely to remain pleased if confronted with a more complex environment. In other studies [15], we implemented another kind of self-motivation called *interactional motivation*. We called the tuple ⟨*experience*, *result*⟩ an *interaction*. Interactional motivation associates a predefined scalar valence with interactions, and implements an agent that tries to enact interactions that have a positive valence, and to avoid interactions that have a negative valence. By predefining interactions that maintain homeostasis (e.g. eating) and giving them a positive valence, this method allows implementing the motivation to regulate homeostasis without a reference to the environment's state. We also highlighted the emergence of sense-making with a more complex learning algorithm for ERC agents [16].

Another example that follows an approach similar to the ERC is the Horde architecture [17]. Horde relies on a swarm of reinforcement-learning agents to control a robot that learns hierarchical temporal regularities of interaction through experience.

4 Conclusion

Inverting the interaction cycle does not merely lead to an *action-perception cycle*. Indeed, the expression *action-perception cycle* suggests that the agent's input data would still represent the environment's state. Instead, inverting the interaction cycle allows designing the agent's input data so that it does not represent the environment's state and does not constitute the agent's perception. For this reason, we propose the new expression *Experiment-Result Cycle* (ERC).

When applied to a robot, the ERC involves implementing algorithms that exploit sensor data as a result of an experiment performed by the robot in the real world. We argue that this approach complies with the embodiment paradigm because it does not require modeling the real world *a priori* in order to design input data as a representation of the world.

We wish cognitive architectures made explicit whether they receive their input data at the beginning or at the end of the interaction cycle: at the beginning when it represents the environment, or at the end when it does not. While there may be more to embodiment than that, we believe that this could at least help clarify some of the differences between non-embodied and embodied cognitive architectures.

Since ERC agents do not incorporate data that directly represent the environment's state, we cannot program them to reach goal states, and we cannot assess them by measuring their performance in reaching goal states. We suggest other research objectives, namely designing ERC agents that exhibit smarter *cognitive behaviors*. This requires the scientific community to define *cognitive behaviors* more precisely and to specify criteria to assess their level of smartness.

Drawing inspiration from methods used in ethology to assess animals' intelligence, we suggest producing evidences of cognitive behaviors using *activity traces*. An activity trace is a stream of data that represents the agent's activity. It includes pieces of the agent's output data, input data, and

internal data. It allows reporting representative strips of activity for the community to judge the agent's level of intelligence.

Acknowledgement

Support for this work was provided by the *Agence Nationale de la Recherche* contract ANR-10-PDOC-007-01.

References

[1] Newell, A. & Simon, H. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3), 113–126.
[2] Kaelbling L., Littman M., & Cassandra A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 99-134.
[3] Foerster, H. (1960). On self-organizing systems and their environments. In Yovits, M. and Cameron, S. (Eds), *Self-Organizing Systems* (pp. 31–50). Pergamon Press, London.
[4] 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.
[5] Varela, Thompson, & Rosch (1991). The Embodied Mind: Cognitive Science and Human Experience. Cambridge, MA: The MIT Press.
[6] Clancey, W. J. (1992). "Situated" means coordinating without deliberation. McDonnel Foundation Conference. Santa Fe, NM.
[7] Pfeifer, R. & Bongard, S. (2006). How the body shapes the way we think: A new view of intelligence. Cambridge, MA: MIT Press.
[8] Oudeyer, P.-Y., Kaplan, F., & Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2), 265-286.
[9] Lungarella, M., Metta, G., Pfeifer, R., & Sandini, G. (2003). Developmental robotics: A survey. *Connection Science*, 15(4), 151–190.
[10] Martin P. & Bateson P. (1993). Measuring behavior, An introductory guide. Cambridge University Press.
[11] McCallum A. (1996). Learning to use selective attention and short-term memory in sequential tasks. The Fourth International Conference on Simulating Adaptive Behavior.
[12] 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.
[13] Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation. *IEEE Transactions on Autonomous Mental Development*, 2(3), 230–247.
[14] Steels, L. (2004). The Autotelic Principle. In I. Fumiya, R. Pfeifer, L. Steels, & K. Kunyoshi (Eds), *Embodied Artificial Intelligence* (pp. 231- 242), Springer Verlag.
[15] Georgeon, O., Marshall, J., & Gay, S. (2012). Interactional motivation in artificial systems: between extrinsic and intrinsic motivation. Second International Conference on Development and Learning and on Epigenetic Robotics (EPIROB2012), San Diego, pp. 1-2, 2012.
[16] Georgeon, O. & Marshall, J. (2013). Demonstrating sensemaking emergence in artificial agents: A method and an example. *International Journal of Machine Consciousness*, 5(2): 131-144.
[17] Sutton R., Modayil J., Delp M., Degris T., Pilarski P. M., White A., & Precup D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In: Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11), Volume 2: 761–776. IFAAMAS, Taipei.