Causality Reconstruction by an Autonomous Agent

Jianyong Xue, Olivier L. Georgeon, Mathieu Guillermin

Causality Reconstruction by an Autonomous Agent

Jianyong Xue[1], Olivier L. Georgeon[1,2] and Mathieu Gillermin[2]

[1] Université de Lyon, LIRIS CNRS UMR5205, F-69622 Villeurbanne, France,
Université Claude Bernard Lyon 1
[2] Université de Lyon, LBG UMRS 449, F-69288 Lyon, France
UCLy, EPHE
{jianyong.xue, olivier.georgeon}@liris.cnrs.fr,
mguillermin@univ-catholyon.fr

Abstract. Most AI algorithms consider input data as "percepts" that the agent receives from the environment. Constructivist epistemology, however, suggests an alternative approach that considers the algorithm's input data as feedback resulting from the agent's actions. This paper introduces a constructivist algorithm to let an agent learn regularities of actions and feedback. The agent organizes its behaviors to fulfill a form of intentionality defined independently of a specific task. The experiment shows that this algorithm constructs a Petri net whose nodes represent hypothetical stable states afforded by the agent/environment coupling, and arcs represent transitions between such states. Since this Petri net allows the algorithm to predict the consequences of the agent's actions, we argue that it constitutes a rudimentary causal model of the "world" (agent+environment) learned by the agent through experience of interaction. This work opens the way to studying how an autonomous agent car learn more complex causal models of more complex worlds, in particular by explaining regularities of interaction through the presence of objects in the agent's surrounding space.

Keywords: Feedback, Constructivist Paradigm, Interaction, Causality Reconstruction, Developmental Learning.

1 Introduction

Traditional Artificial Intelligence (AI) algorithms strongly depends on how the agent connects with the environment through actuators and sensors. As Russell & Norvig stated that "the problem of AI is to build agents that receive percepts from the environment and perform actions" [2, p. iv]. In this paradigm, hereafter referred to as the *realist paradigm*, the AI algorithm assumes that the agent's input data is a direct function of the state of the environment.

Developmental approaches in AI explore alternative paths. Following the insights of cognitive science, developmental AI draws on the seminal work of Piaget [4] to define guiding principles for designing autonomous artificial agents [5]: embodiment, situatedness and a prolonged ontogenetic developmental process [1]. Ontogenetic development refers to the development of an individual by learning through

interaction with its environment. In general, developmental AI tries to reproduce the path along which human acquire knowledge. For example, Mugan & Kuipers proposed a seminal attempt to construct a representation of the environment from sensori-motor interactions [6].

Constructivist epistemology [e.g. 8, 9] suggests an approach to make an autonomous agent iteratively construct a representation of an unknown environment. In the constructivist paradigm, the agent's input data constitutes feedback of the agent's actions with the environment [3]. By contrast with the realist paradigm, input data is not considered as directly representing elements of the world, supposedly pre-existing and available for registration. In a given state of the environment, the input data may vary according to the agent's action, and thus does not constitute a direct representation of an agent-independent presupposed reality.

Authors in psychology argue that perceiving the world consists of actively constructing a representation of the current situation through interaction, as opposed to directly receiving a representation of the world's state [10]. Georgeon, Casado and Matignon [11] model the biological beings as agents trying to perform rewarding interactions with their environment (interaction-driven tasks). Rosech et al. propose to implement a process of knowledge construction from regularities the agent learns from its interactions with the environment without any predefined knowledge [9]. Georgeon and Hassas [14] present a model that does not make of the information upon the environment directly available to registration by the agent. De Loor, Manac'h and Tisseau [12] propose an enaction-based artificial intelligence (EBAI), combined with the evolution of the environment to refine the ontogenesis of an artificial system. The achieved studies lead to the integration of human interactions into the environment to construct relevant meaning in terms of participative artificial intelligence.

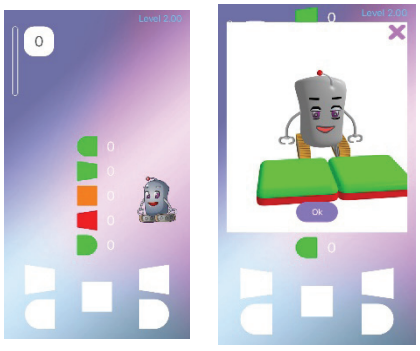
In this paper, we introduce design principles, inspired by the constructivist paradigm, to develop a self-motivated agent capable of learning regularities in a task previously introduced by Georgeon [7] in a pedagogical game called *Little AI* (next section).

2 Experimental settings

In the game Little AI, the player controls a simulated artificial agent in a simulated environment. She presses action buttons (white shapes at the bottom of the screen, Fig. 1.a) and receives feedback. She, however, has no knowledge of what these actions are doing. The tuple <action, feedback> is called an *experience*. In the game, an experience is represented as a colored shape whose shape represents the action and whose color represents the feedback. The history of experiences constitutes the *trace* (stream of colored shapes in Fig. 1.a). The purpose of the game is to learn to predict the feedback of actions. To do so, the player must infer a model that links actions and feedback, based on regularities observed in the trace.

In this paper, we focus on level 2.00 of Little AI, which provides five possible actions. Unbeknownst to the player, the agent consists of a simulated robot interacting

with a simulated environment made of two tiles, one on the right and one on the left. Fig 1.b shows the robot and the tiles but the player cannot see this screen until she wins the level. Tiles have two different sides: recto (green) and verso (red). The agent can swap the tiles and test their sides. More precisely, the set A of possible actions is $\{a_1$=*feel left*, a_2=*swap left*, a_3=*feel both*, a_4=*feel right*, a_5=*swap right*$\}$ (Fig. 2). At round t, the player chooses an action a_t from A, and then receives feedback f_t depending on this action and on the state of the environment. We say that the agent *enacts* experience $e_t = \langle a_t, f_t \rangle$ when it performs action a_t and receives feedback f_t. Actions *feel left* and *feel right* test the side of the corresponding tile; these actions yield feedbacks f_1 or f_2 whether the tile is recto or verso. Action *feel both tiles* tests the both tiles at the same time and can yield three different feedbacks: f_1: *both tiles are recto*, f_2: *both tiles are verso*, f_3: *one tile is recto and the other is verso*. Action *swap right* (or *swap left*) swaps the right (or left) tile and then returns feedback f_1 (or f_2) whether the right (or left) tile ends up recto or verso. As a result, there are eleven possible experiences. The player cannot assume that the color of feedback corresponds to the color of the side of the corresponding tile. We set the color of feedback to the same color as the side of the corresponding tile only to facilitate the understanding of this paper.



a). The initial interface b). The 3D interface

Fig. 1. Little AI's user interface and Level 2.00

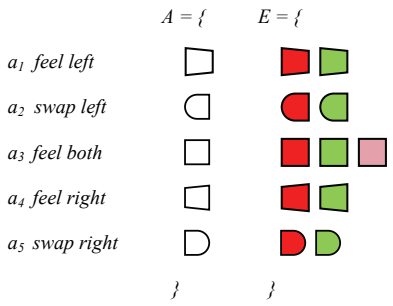


Fig. 2. Five actions and eleven experiences.

3 The algorithm

In this paper, we present an algorithm to replace the human player and control the agent. Similar to the human player, the algorithm ignores the meaning of actions and feedback. It must learn to predict feedback triggered by actions based on regularities in the trace of experiences, which amounts to inferring a model of the *agent/environment* coupling. From here on, we use the term *agent* to refer to the "simulated robot" and the term *algorithm* to refer to the algorithm that controls the behavior of this agent instead of the player.

The problem of constructing a model from sequences of events has been studied in process mining (PM) [15]. PM algorithms take a flow of events and automatically construct an automaton that can generate this flow. For instance, Van der Aalst [16] proposed the foundational α-algorithm to construct such an automaton modeled as a Petri net.

Drawing on Van der Aalst's α-algorithm and Georgeon et al.'s work [13], our algorithm is based upon the assumption that there are stable states in the agent/environment coupling. This algorithm progressively constructs a Petri net whose nodes represent hypothetical stable states and arcs represent transitions between nodes. Each arc is associated with an experience. When the Petri net has been constructed, the agent uses it to predict the consequences of enacted experiences. The agent uses the token of the Petri net to keep track of the state. That is, when the token is on a peculiar node, the agent believes that the coupling with the environment is in the state represented by this node. The position of the token thus represents the current *belief state* of the agent. When the agent enacts an experience, the agent moves the token through the arc associated with this experience to the destination node of this arc. An arc can be a self-loop if its destination node is also its origin node. In this case, its associated experience does not change the belief state of the agent. We call such an experience a *persistent* experience as opposed to *sporadic* experiences that change the state.

We divide the learning process into two parts: the *node construction stage* and the *arc construction stage*. In the node construction stage, the algorithm tries to learn which stable states exist. In the arc construction stage, the algorithm learns how to transition from one state to another.

3.1 The node construction stage

In this stage, the agent searches persistent experiences by trying to enact experiences several times in a row. When the same experience is enacted a certain number (the *excitement threshold*) of times in a row, the algorithm assumes that the experience is persistent and creates a new node. The algorithm implements a *BeliefState* class with an array *triedNumber* for each experience and a method *getLeastTriedExperience()*. When a new node is created, a new *BeliefState* object associated with this node is instantiated. Its *triedNumber* attribute is used to count the number of trials of an experience in this belief state. When the algorithm is in the "curious" mood, it uses the *getLeastTriedExperience()* function to select the least tried experience based on *triedNumber*.

3.2 The arc construction stage

When the agent learns a stable state, then it goes into the arc construction stage. The arc construction combines the new feedback the agent gets from its past interactions and the previous patterns in the stream of traces. With the changeable environment, it needs to figure out the difference context between previous and post persistent experiences after the agent experienced a sporadic experience. If the two persistent

experiences are different, then an arc between these two persistent experiences will be created through this sporadic experience. Otherwise, the agent continues try another sporadic experience until all sporadic experiences have all been tested with all persistent experiences. When all experiences are known to the agent and no changes happen in the causal structure, the agent is in the *confident* mood. Overall, we expect the algorithm to learn the Petri net in Fig. 3, which constitutes a valid representation of the structure of the agent/environment coupling of Level 2.00 of Little AI.

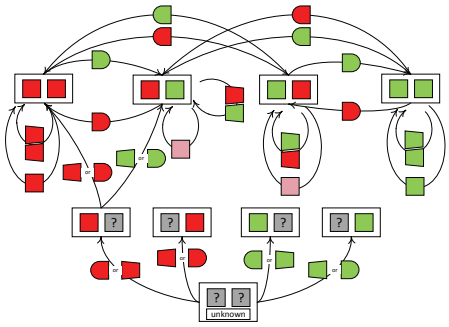


Fig. 3. Partial representation of the Petri net constructed by the algorithm. White rectangles represent nodes. Arcs with feeling experiences (trapezoids and squares) are self-loops that do not change the state. Arcs with swapping experiences (half-circles) cause transitions between belief states.

3.3 Details of the algorithm

Alg 1. The control of the agent/environment interaction cycle

```
01      initial parameters
02          currentBeliefState = "unknown", mood = "curious"
03      loop
04          if mood = "curious"
05              intendedExperience = getLeastTriedExperience(currentBeliefState)
06          if mood = "confident"
07              finish the knowledge construction and learning is done
08          if mood = "excited"
09              intendedExperience = enactedExperience
10          enactedExperience = Environment(intendedExperience)
11          if enactedExperience is unknown
12              mood = "excited"
```

The algorithm has three possible motivational moods: *curious*, *excited* and *confident*. On initialization, the current belief state: *unknown*, and it is in a *curious* mood. Lines 04 and 05: if it is in a curious mood, then it selects the least tried experience in the context as the next *intended experience*. Lines 06 and 07: if it is in a *confident* mood, then the algorithm has reconstructed the whole Petri net. Lines 08 and 09: if it is in an excited mood, then it intends to repeat the previously enacted experience. Line 10: the algorithms passe the intended experience to the subprogram that implements the

environment (Little AI Level 2.00). This subprogram processes the action associated with the intended experience and returns the enacted experience, which may be the intended experience or not. Lines 11 and 12: if the enacted experience is unknown, which means this experience has neither been marked *persistent* nor *sporadic*, then the mood becomes excited.

Alg 2. The node construction and the arc construction

```
01      if mood = "excited"
02          if intendedExperience ≠ enactedExperience
03              intendedExperience is sporadic
04              currentBeliefState = knowledgeUpdating(intendedExperience)
05          else if excitement > excitementThreshold
06            enactedExperience is persistent
07            create new beliefState and added in the beliefStateList
08              currentBeliefState = knowledgeUpdating(enactedExperience)
09          else
10            excitement++
11      updateTriedNumberOfExperience(intendedExperience)
12      knowledgeUpdating(currentBeliefState)
13      if all experiences have not been tried yet in the context of this currentBeliefState
14          mood = "curious"
15      if all experiences have been tried and knowledge isn't updating
16          mood = "confident"
```

Lines 01 to 10: if the algorithm is in the excited mood and the intended experience differs from the actually enacted experience, the intended experience is marked sporadic. Otherwise, the algorithm increments its excitement level. When the excitement reaches the preset threshold, the experience is marked *persistent*, a new belief state is instantiated. Line 11: the intended experience updated its tried numbers. Line 12: learn regularities and update the current belief state. Lines 13 and 14: if all experiences the agent has not been experienced with, the mood becomes curious. Lines 15 and 16: all experiences are known to the agent and there are no changes in the procedure of knowledge reconstruction. This means that the algorithm has reconstructed a valid representation of the structure of the agent/environment coupling of Level 2.00 of Little AI. The mood switches to *confident*.

4 Experiment

Our experiment shows that the algorithm was able to construct the Petri net in 350 interaction cycles (Fig. 4). Step 1: the algorithm intends a red left trapezoid and obtains a same red left trapezoid. Since the experience is neither yet marked sporadic or persistent, the agent gets excited (initiated black bar in Line 4). Step 2 to 5: the algorithm repeats the red left trapezoid and gets increasingly excited. Step 6: the algorithm reaches the excitement threshold; it marks the red left trapezoid as persistent and creates a new belief state associated with this experience (line 3: the current belief becomes red left trapezoid). Line 4: the algorithm becomes curious to

play with the newly created belief. Step 7: the algorithm tries the red rectangle. Step 15: the algorithm tries the red left half-circle and obtains green left half-circle. Step 16: green left half-circle is marked as sporadic since it differs from the intended experience, the algorithm enters the arc construction stage. Step 17: the algorithm encounters again the red left trapezoid, the current belief state associates with this persistent experience. Step 77: similar, green right half-circle is sporadic. Step 350: The experiences are all marked and no more possible changes in the Petri net. The algorithm mood becomes confident (green circle). Arrived at this step, the algorithm can use the constructed Petri net to predict the consequences of its actions.

Fig. 4. Trace of the first 350 interaction cycles in our experiment. Line 1: intended experiences. Line 2: enacted experiences. Line 3: belief states: unknown (grey triangle) / known state represented by its corresponding persistent experience. Line 4: mood: curious (question mark), excited (increasing black bars), or confident (green circle).

5 Conclusion

We presented an algorithm that initially ignores the structure of the agent that it is controlling, as well as the structure of the environment with which this agent is interacting. As and when the algorithm controls the agent, it learns a model of the possibilities of interaction afforded by the agent/environment coupling.

An intriguing philosophical question is whether the learned model of the agent/environment coupling constitutes the best possible model accessible to the agent. When transposed to humans and their knowledge, Kantian-like positions affirm that we can only know the world as we experience it. By contrast, those admitting a more (scientific) realist epistemology believe that knowledge can be pushed further, toward the world in itself. Returning to our discussion, this suggests the following question: would it be possible and interesting to design an agent that would try to infer theories of the world in itself (the complete implementation of the agent and of the environment)?

Technically, the question remains how the algorithm could construct a theory that involves a representation of its environment with tiles and their sides. This may be feasible if the agent presupposes the existence of space. The agent could try to construct a simpler model based on the assumption that states of the agent/environment coupling are caused by the presence of objects in some locations in space. Such simplification of knowledge will be necessary when moving on towards more complex tasks. In more complex tasks, the Petri net will be too large and complex. Explaining the regularities in terms of the presence of objects in some locations in space will provide a powerful means to deal with such complexity.

References

1. Guériau, M., Armetta, F., Hassas, S., Billot, R., EI Faouzi, N., E.: A constructivist approach for a self-adaptive decision-making system: application to road traffic control. In: 28th International Conference on Tools with Artificial Intelligence, pp. 670-677. IEEE, San Jose (2016)
2. Russell, S., Norvig, P.: Artificial intelligence, a modern approach. Pearson, New Jersey (2003).
3. Olivier, O. G., Mathieu G.: Mastering the laws of feedback contingencies is essential to constructivist artificial agents. Constructivist Foundations 13(2), 300-301 (2018).
4. Jean, P.: The construction of reality in the child. Journal of Consulting Psychology 19(1), 77 (1955).
5. Zlatev, J., Balkenius, C.: Introduction: Why “epigenetic robotics”?. In: Balkenius, C., Zlatev, J., Kozima, H., Dautenhahn, K., Breazeal, C.(eds.) Lund University Cognitive Science Series, no. 85 (2001)
6. Mugan, J., Kuipers, B.: Autonomous representation learning in a developing agent. Computational and Robotic Models of the Hierarchical Organization of Behavior. Springer, Berlin, Heidelberg (2013).
7. Georgeon, O. L.: Little AI: Playing a constructivist robot. SoftwareX 6, 161-164 (2017).
8. Von Glasersfeld, E.: "An introduction to radical constructivism." The invented reality. W. W. Norton, London (1984).
9. Roesch, E. B., Spencer, M., Nasuto, S. J., Tanay, T., Bishop, J. M.: Exploration of the functional Properties of interaction: computer models and Pointers for theory. Constructivist Foundations 9(1), 26-33 (2013).
10. Findlay, J., Gilchrist, I.: Active Vision: The Psychology of Looking and Seeing. Oxford University Press, New York (2003).
11. Georgeon, O. L., Casado, R. C., Matignon, L. A.: Modeling biological agents beyond the reinforcement-learning paradigm. Procedia Computer Science 71, 17-22 (2015).
12. De Loor, P., Manac’h, K., Tisseau, J.: Enaction-based artificial intelligence: Toward co-evolution with humans in the loop. Minds and Machines 19(3), 319-343 (2009).
13. Georgeon, O. L., Bernard, F. J., Cordier, A.: Constructing phenomenal knowledge in an unknown noumenal reality. Procedia Computer Science, 71, 11-16 (2015).
14. Georgeon, O., Hassas, S.: Single Agents Can Be Constructivist too. Constructivist Foundations 9(1), 40-42 (2013).
15. Van der Aalst, W., Van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Wei- jters, A.: Workflow mining: A survey of issues and approaches. Data & Knowledge Engineering 47(2), 237–267 (2003).
16. Van der Aalst, W., Weijters, A., Maruster, L.: Workflow Mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128-1142 (2004).