Integrating Symbolic and Sub-symbolic Reasoning

Claes Strannegård[1,2(✉)] and Abdul Rahim Nizamani[3]

[1] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Gothenburg, Sweden
[2] Department of Applied Information Technology, Chalmers University of Technology, Gothenburg, Sweden
claes.strannegard@chalmers.se
[3] Department of Applied Information Technology, University of Gothenburg, Gothenburg, Sweden
abdulrahim.nizamani@gu.se

Abstract. This paper proposes a way of bridging the gap between symbolic and sub-symbolic reasoning. More precisely, it describes a developing system with bounded rationality that bases its decisions on sub-symbolic as well as symbolic reasoning. The system has a fixed set of needs and its sole goal is to stay alive as long as possible by satisfying those needs. It operates without pre-programmed knowledge of any kind. The learning mechanism consists of several meta-rules that govern the development of its network-based memory structure. The decision making mechanism operates under time constraints and combines symbolic reasoning, aimed at compressing information, with sub-symbolic reasoning, aimed at planning.

Keywords: Autonomous agent · Bounded rationality · Survival · Symbolic reasoning · Sub-symbolic reasoning

1 Introduction

Symbolic reasoning connects linguistic statements via syntactic rules, whereas sub-symbolic reasoning connects sensory concepts via association links [6]. These forms of reasoning have been studied since antiquity, e.g. by Euclid [2], who designed systems for axiomatic reasoning, and by Aristotle [10], who investigated associative as well as axiomatic reasoning. These forms of reasoning are closely related to James' division into associative and symbolic reasoning [4] and Kahneman's dichotomy of System 1 and System 2 processes [5].

The ability to do symbolic and sub-symbolic reasoning and to combine the two seems to be an essential feature of human intelligence [6]. In contrast, AI systems rarely support more than one of the two processes. For example, neural networks and reinforcement learning systems support sub-symbolic, but usually

B. Steunebrink et al. (Eds.): AGI 2016, LNAI 9782, pp. 171–180, 2016.
DOI: 10.1007/978-3-319-41649-6_17

not symbolic reasoning, while automatic theorem provers and logic-based systems are the other way around. In particular, deep networks are good at recognizing faces or evaluating go-positions, but not at arithmetic, while automatic theorem provers have the opposite strengths.

Several cognitive and agent architectures combine symbolic and sub-symbolic reasoning to varying degrees. Examples include Soar [8], ACT-R [1], OpenCog [3], AERA [14], and NARS [17]. Some of the architectures with this capacity are hybrid systems with juxtaposed subsystems operating on separate knowledge bases. Certain others are not fully autonomous, in that they depend on engineers for manually preparing the system for new domains, e.g. for updating the set of production rules.

Despite the progress made, the following quote by Yoshua Bengio, one of the deep learning pioneers, suggests that the two types of reasoning have not been sufficiently integrated for artificial general intelligence purposes [7]:

> Traditional endeavors, including reasoning and logic—we need to marry these things with deep learning in order to move toward AI.

Schmidhuber proposed to combine long-term memory compression with reinforcement learning [11]. This idea has been successfully used in several AI-programs, including Alphago [12].

In this paper we present a computational model that compresses the long-term memory as well as the working memory. In both cases we use compression with bounded cognitive resources in order to make the compression computationally feasible. This reflects our belief that compression is key to natural intelligence and also that working memory compression is the sole purpose of symbolic reasoning. Our system has a fixed set of needs, whose levels of satisfaction are computed on the basis of sensory data (interoception). The sole goal of the system is to survive as long as possible by satisfying its needs. This will in turn cause the system to take different actions, e.g. to ambulate between a water source and a food source.

In contrast to many cognitive architectures, our system does not use the notion of task. Instead, the planning process continuously searches for action sequences aimed at increasing the probability of survival. The behavior of the system can be altered by external agents who provide reward, similarly to how dogs can be trained by humans who reward tricks with treats. The system can also learn on its own without interacting with other agents.

This paper focuses entirely on artificial general intelligence and aims for a fully autonomous artificial system without regard to biological or psychological realism. We combine and extend our previous work on long-term memory compression [16], working memory compression [9], and transparent networks [15].

Section 2 presents the components of our system and Sect. 3 describes the update mechanisms of the components. Section 4 presents the reasoning mechanism in greater detail. Section 5 presents a proof-of-concept prototype implementation of the system. Section 6 draws some conclusions.

2 System Components

The system consists of several components that develop over time.

2.1 Status Signals

Time is modeled using the natural numbers $\mathbb{N}$. For $t \in \mathbb{N}$, let $Status(t)$ be the vector $(\sigma_1(t), \ldots, \sigma_N(t)) \in [0,1]^N$. Here N is a fixed positive integer that models the number of needs of the system and $\sigma_i(t) \in [0,1]$, for $0 \leq i \leq N$. For instance, $\sigma_1(t)$ and $\sigma_2(t)$ might reflect the glucose and water concentration in the blood stream, or perhaps the oil and gasoline levels of a vehicle. Intuitively, $Status(t)$ measures the status of the different needs of the system, i.e. its well-being.

2.2 Long-Term Memory

We will use a labeled graph $LTM(t)$ for encoding the system's long-term memory at time t. Intuitively, $LTM(t)$ is a Markov Decision Process (MDP) that the system uses for decision making at t. The states and actions of $LTM(t)$ are described in the vocabulary of transparent neural networks [15]. Our reason for using this formalism is that it facilitates the definition of learning rules that develop $LTM(t)$ over time, as we shall see in Sect. 3.

Definition 1 (LTM). *Let $LTM(t)$ be a graph consisting of a finite set of labeled nodes $D(t)$ and a finite set of labeled edges $E(t) \subseteq D(t)^2$.*

- *Each node of $D(t)$ has exactly one label from the following list: $SENSOR$, $MOTOR$, NOT, AND, OR, x, y, z, $DELAY$, and $ACTION$.*
- *Each edge of $E(t)$ has exactly one label from the following list: $ACTIVITY$, $DECISION$, and $PREDICTION$.*

Edges labeled $PREDICTION$ are also labeled with a probability in $[0,1]$ and an expected reward in $[-1,1]^N$. Here N is the fixed number of needs that was mentioned above.

Figures 1 and 2 provide examples of some graphs that could be part of an LTM. Oval shapes represent $SENSOR$ nodes and squares represent $ACTION$ nodes. Solid, dashed, and annotated arrows represent $ACTIVITY$, $DECISION$, and $PREDICTION$ edges, respectively. Many more examples can be found in [15].

Remark 1 (Intended interpretation). $SENSOR$-nodes model sensors, e.g. receptor cells with ion channels sensitive to cold temperature, mechanical pressure, or acidity. $MOTOR$-nodes model muscle-controlling motor neurons. NOT, AND, and OR-nodes model nerve cells that compute the corresponding boolean functions. The first of these is binary operators and the other two binary. x, y, or z-nodes are abstraction nodes that are used for pattern matching purposes (using temporary assignments of nodes to variables). $DELAY$-nodes model nerve cells

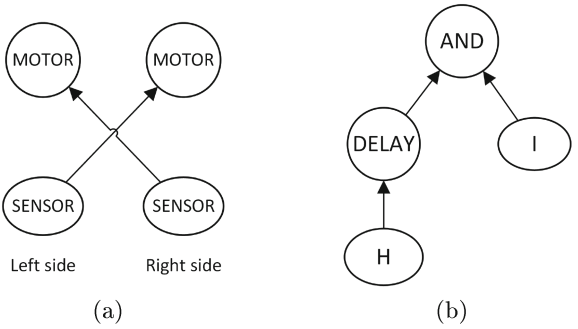

Fig. 1. (a) A Braitenberg vehicle with two light sensors and two motors for wheels on the left and right sides, respectively. (b) The AND-node of this graph recognizes the sequence *HI*, i.e. letter H immediately followed by the letter I. The sensors of this graph recognize letters. Alternatively, the sensors could be replaced by top nodes of more complex network that recognize letters.

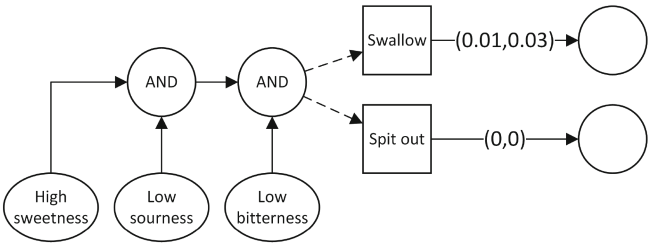

Fig. 2. We assume that the system has two needs: water and glucose. When the system receives a familiar combination of tastes (of an apple), it chooses between two actions, one of which (swallow) leads to reward.

that re-transmit action potentials with a delay. $ACTION$-nodes model neurons that activate motor sequences as a result of (conscious) decisions. $ACTIVITY$-edges model connections in nerve systems where activity propagates in the network. $DECISION$-edges model connections (choices) in MDPs that go from states to actions. $PREDICTION$-edges model connections in MDPs that go from actions to states. The number is a probability that will be learned and the vector is the expected reward w.r.t. N fixed needs of the system. This vector will also be learned.

2.3 Activity

Let $Activity(t)$ be a subset of the nodes of $LTM(t)$. This set models the neurons that fire at time t. Our view in this connection is essentially that humans are capable of symbolic as well as sub-symbolic information processing and that both processes run on systems of neurons that are binary in the sense that they either fire all-out or not at all at any given moment.

2.4 Attention

Let $Attention(t)$ be a subset of $D(t)$ consisting of at most one node. This set models the node under attention at time t, if any.

2.5 Working Memory

Let $WM(t)$ be a sequence consisting of a finite number of nodes of $D(t)$. This sequence models the content of the system's working memory at time t.

2.6 Decision

Let $Decision(t)$ be a subset of $D(t)$ consisting of at most one node. This set models the decision at time t, if any.

Remark 2 (Bounded rationality). In line with biological organisms with cognitive resources all of which are limited, we impose firm resource bounds on all components of our system. Thus the capacity of $LTM(t)$ is limited (e.g. to 10^5 nodes) and so is the capacity of $WM(t)$ (e.g. to 5 nodes). As we shall see later, the processing time that the system uses for decision-making is also bounded, although in this case the limit depends on the status of the system. Thus the system has bounded rationality in several ways, leading it to a *satisficing* rather than an *optimizing* behavior [13].

3 Update Functions

In this section we will specify how the system components are updated in response to the input that the system receives.

Definition 2 (Input stream). *The function* $input(t)$ *is an assignment of values in* $\{0, 1\}$ *to each* $SENSOR$ *node of* $LTM(t)$.

The components of the knowledge base can largely be initialized arbitrarily at $t = 0$. Thus it will start as a *genotype* and gradually develop *phenotypes* depending on the input stream. Now suppose the system receives the input $input(t + 1)$. Then the components of the knowledge base will be updated as described below and in the order specified.

3.1 Activity Update

Here we define $Activity(t+1)$ via a function A that selects which nodes to shall be put into the set:

Definition 3 (Activity update). *Let $A(a,t)$ be defined by recursion on t as follows. Here $L(a)$ denotes the label of node a. Let $A(a,0) = 1$ if $a \in Activity(0)$ and $A(a,0) = 0$ otherwise. Moreover, let*

$$A(a,t+1) = \begin{cases} 1 \text{ if } a \in Attention(t) \\ 1 \text{ if } L(a) = ACTION \text{ and } a \in Decision(t) \\ input(t+1)(a) \text{ if } L(a) = SENSOR \\ A(a',t+1) \text{ if } L(a) = MOTOR \text{ and } (a',a) \in ACTIVITY \\ min\{A(a',t+1) : (a',a) \in ACTIVITY\} \text{ if } L(a) = AND \\ max\{A(a',t+1) : (a',a) \in ACTIVITY\} \text{ if } L(a) = OR \\ 1 - A(a',t+1) \text{ if } L(a) = NOT \text{ and } (a',a) \in ACTIVITY \\ A(a',t) \text{ if } L(a) = DELAY, (a',a) \in ACTIVITY \end{cases}$$

This definition should be read as a case statement in programming that selects the first case that applies. It describes how activity propagates through nodes of different types. The nodes x, y, and z are activated via an additional mechanism.

3.2 Status Update

The vector $Status(t+1)$ is computed directly from $input(t+1)$ using an arbitrary fixed function. For instance, $\sigma_1(t+1)$ could be the fraction of the insulin receptors that fire at $t+1$. Now, reward can be defined as status changes:

Definition 4 (Reward vector). *Let $Reward(t+1) = Status(t+1) - Status(t)$.*

Note that $Reward(t+1) \in [-1,1]^N$, since each σ_i takes values in $[0,1]$.

3.3 Attention Update

To be able to describe how attention is updated we need to introduce a couple of concepts.

Definition 5 (Most urgent need). *The* most urgent need *at t is defined as* $\arg\min_i(\sigma_i(t))$.

Definition 6 (Top active node). *A node $a \in LTM(t)$ is top active at t if $a \in Active(t)$ and there is no $b \in Active(t)$ such that $(a,b) \in ACTIVITY$.*

Together the top active nodes constitute a description of the present situation at a maximum level of detail in terms of previously experienced situations. Therefore they are important from the perspective of decision making and attention. For instance, if the system sees a green snake, then the nodes representing "green", "snake", and "green AND snake", might become activated. Among those, "green AND snake" would be top active if it had no other active nodes above it, whereas the other two would only be active.

Definition 7 (Flashing node). *A node* $a \in LTM(t)$ *is flashing at* $t+1$ *if* $a \in Active(t+1) - Active(t)$.

When attention moves from no node or an old node to a new node, this new node will always be top active and flashing. In general there are many such nodes to choose from and in that case, attention will go to the node that seems to be the most promising when it comes to satisfying its most urgent need.

3.4 WM Update

The sequence $WM(t)$ stores the nodes that have been under attention most recently.

3.5 Decision Update

The system makes decisions by continuously computing $Decision(t+1)$. If $Decision(t+1) = \emptyset$, then no action is taken. Otherwise $Decision(t+1)$ contains a single node, which is labeled $ACTION$. When this node gets activated, motor activity will follow. The one and only goal of decision making is to prolong the system's life, i.e. avoiding that some σ_i reaches 0. The decision update process may use several consecutive cycles for making a decision. In the mean time $Decision(t+1) = \emptyset$. The processing time is constrained by a time limit that depends on $Status(t)$. In general, less time is available if some σ_i is low. A new situation that brings along dramatic status changes can cause the processing time to run out and make the system return to the main loop and be ready to deal with the new situations.

Our system combines sub-symbolic reasoning, aimed at planning for survival, with symbolic reasoning, aimed at improving the planning process by compressing the information contained in $WM(t)$. Both types of reasoning is based on knowledge that is stored in $LTM(t)$. The reasoning mechanisms, which are used for decision making, are described in greater detail in Sect. 4. In addition, the system engages in exploration by testing new or old actions for different nodes under attention.

3.6 LTM Update

Here we outline the learning mechanisms (or meta-rules) that govern the transfer from $LTM(t)$ to $LTM(t+1)$. For reasons of space, these mechanisms are not described in full detail. Several detailed definitions can be found in [9,15].

1. Hebbian learning at random moments (so that frequently occurring sensory combinations or sequences of sensory combinations will be remembered)
2. Status-driven learning of state-action pairs leading to reward (so that good actions can be repeated) or to punishment (so that bad actions can be avoided)

3. Repetition-driven learning (so that motor patterns that cause sensory patterns to be repeated can be learned, e.g. in the context of sensor-motor development and language learning)
4. Novelty-driven learning (so that actions leading to sensory changes are remembered, e.g. for learning sensory-motor patterns that lead to locomotion and manipulation of the environment)
5. Abstraction (so that general patterns can be remembered via the use of abstraction nodes, e.g. in the context of symbolic pattern learning)
6. Forgetting (so that memory structures that are rarely used and not associated with strong reward or punishment are eventually removed)
7. Adjustment of the expected rewards and transition probabilities of the $PREDICTION$-edges (so that experience will be properly encoded).

4 Reasoning Mechanisms

Now let us consider the system's reasoning mechanisms.

4.1 Sub-symbolic Reasoning

The fundamental building blocks of MDPs are association rules that lead from a state-action pair (s, a) to a resulting state s' with probability p and expected reward r. Our LTM was designed for representing rules of exactly this kind with the convention that the node under attention defines the state of the system. Thus we can represent MPDs in our framework and do sub-symbolic reasoning with all kinds of reinforcement learning methods, including Monte-Carlo methods, Q-learning and Dyna-Q. Our prototype implementation uses Q-learning. Since we have a mechanism for creating memories of sequences of events, the system has the Markov property, but is nevertheless able to take history into account. Figure 3 shows an example of a sub-symbolic computation to the right.

4.2 Symbolic Reasoning

We view rewrite rules as association rules. For instance, we view the rewrite rule $2 * 3 \mapsto 6$ as an association rule that leads from the node representing the sequence $2 * 3$ and the motor action "write 6" to the node "read 6" with probability 1 and some positive expected reward. Another rewrite rule leads from $\top \wedge x$ and the action "write x" to the node "read x". Symbolic reasoning is done by means of computations, i.e. successive transformations of $WM(t)$, by means of rewrite rules. A central rule is Chunk, which is built-in to the system. This rule enables contents in the working memory to be chunked into a sequence, provided the sequence in question is an element of $LTM(t)$. For instance, suppose $WM(t)$ contains the sequence $(2, *, 3)$. Also suppose $LTM(t)$ contains the sequence $2*3$. Then Chunk enables the transition from $(2, *, 3)$ to $(2 * 3)$. Figure 3 shows an example of a symbolic computation to the left.

$$\dfrac{\dfrac{\dfrac{\dfrac{(2+3,*,4)}{(5,*,4)}\text{ Rewrite } 2+3 \mapsto 5;1.0;(0,0)}{(5*4)}\text{ Chunk; } 1.0;(0,0)}{(4*5)}\text{ Rewrite } x*y \mapsto y*x;1.0;(0,0)}{(20)}\text{ Rewrite } 4*5 \mapsto 20;1.0;(0,0)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{(\text{Big rock})}{(\text{River})}\text{ Walk north; } 0.8;(0,0)}{(\text{River})}\text{ Drink; } 1.0;(0.4,0)}{(\text{Apple tree})}\text{ Walk west; } 0.9;(0,0)}{(\text{Apple tree})}\text{ Eat apple; } 1.0;(0,0.2)$$

Fig. 3. Two computations by a system with two needs: water and glucose. The annotations show the action name, the transition probability and the expected reward in terms of water and glucose. The left panel shows a symbolic computation (compression) and the right panel a sub-symbolic computation (a simple plan for drinking and eating).

Remark 3 (Mixed computations). The ability to combine symbolic and sub-symbolic reasoning can be critical. For instance, if you hear the voices of three burglars in your house and later see two of them leave, then you could use symbolic reasoning to conclude that one burglar is still in the house and then sub-symbolic reasoning to conclude that you should act in a certain way, e.g. remain still. In the present framework symbolic and sub-symbolic computation steps can be mixed arbitrarily.

5 Prototype Implementation

We have implemented a proof-of-concept prototype of the system described above. The code is available at `github.com/arnizamani`. Several of the mechanisms for learning and reasoning that are mentioned in Sect. 3 have been implemented in our earlier work on transparent networks [15], inductive learning [16], and symbolic reasoning [9]. Now we have also implemented a simple prototype of the system with all the components described in Sect. 2. For simplicity the number of needs N was set to 1. An MDP was used together with a policy for decision making. Q-learning was used to update the policy. A class named Environment was implemented to simulate a real environment that provides an input stream and a reward signal to the agent. The implemented system is limited in features. In particular, the update function for $LTM(t)$, which was partly developed in [15], has not been added yet. A simple mechanism moves the attention to a top-active flashing node. The rule Chunk has been added to the compression mechanism of $WM(t)$.

6 Conclusion

We have proposed a way of bridging the gap between symbolic and sub-symbolic reasoning. More precisely, we have presented a system together with a prototype implementation that combines symbolic reasoning, aimed at compressing information, with sub-symbolic reasoning, aimed at planning. The system subsumes and extends our previously developed computational models of symbolic and

sub-symbolic reasoning. Much work remains, however, for turning the present proof-of-concept implementation into a fully functional autonomous system.

Acknowledgement. This research was supported by The Swedish Research Council, grants 2012-1000 and 2013-4873. We would like to thank José Hernández-Orallo for many helpful suggestions.

References

1. Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychol. Rev. **111**(4), 1036 (2004)
2. Fitzpatrick, R., Heiberg, J.: Euclid's Elements (2007)
3. Goertzel, B., Pennachin, C., Geisweiller, N.: The OpenCog Framework. In: Engineering General Intelligence, Part 2, pp. 3–29. Springer (2014)
4. James, W.: The Principles of Psychology. American Science Series: Advanced Course. H. Holt, New York (1918)
5. Kahneman, D.: A perspective on judgment and choice: mapping bounded rationality. Am. Psychol. **58**(9), 697 (2003)
6. Kelley, T.D.: Symbolic and sub-symbolic representations in computational models of human cognition what can be learned from biology? Theory Psychol. **13**(6), 847–860 (2003)
7. Knight, W.: Will machines eliminate us? MIT Technol. Rev. **119** (2016)
8. Laird, J.: The Soar Cognitive Architecture. MIT Press, Cambridge (2012)
9. Nizamani, A.R., Juel, J., Persson, U., Strannegård, C.: Bounded cognitive resources and arbitrary domains. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 166–176. Springer, Heidelberg (2015)
10. Ross, G.: Aristotle: De Sensu and De Memoria. The University Press, Cambridge (1906)
11. Schmidhuber, J.: Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity. In: Corruble, V., Takeda, M., Suzuki, E. (eds.) DS 2007. LNCS, vol. 4755, pp. 26–38. Springer, Heidelberg (2007)
12. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
13. Simon, H.A.: Models of Bounded Rationality: Empirically Grounded Economic Reason, vol. 3. MIT Press, Cambridge (1982)
14. Steunebrink, B.R., Koutník, J., Thórisson, K.R., Nivel, E., Schmidhuber, J.: Resource-bounded machines are motivated to be effective, efficient, and curious. In: Kühnberger, K.-U., Rudolph, S., Wang, P. (eds.) AGI 2013. LNCS, vol. 7999, pp. 119–129. Springer, Heidelberg (2013)
15. Strannegård, C., Cirillo, S., Wessberg, J.: Emotional concept development. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 362–372. Springer, Heidelberg (2015)
16. Strannegård, C., Nizamani, A.R., Sjöberg, A., Engström, F.: Bounded kolmogorov complexity based on cognitive models. In: Kühnberger, K.-U., Rudolph, S., Wang, P. (eds.) AGI 2013. LNCS, vol. 7999, pp. 130–139. Springer, Heidelberg (2013)
17. Wang, P., Hammer, P.: Assumptions of decision-making models in AGI. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 197–207. Springer, Heidelberg (2015)