Limits of Intelligence and Design Implication

Son Tran(✉), Sophie Alyx Taylor, and Dan V. Nicolau Jr.

School of Mathematical Sciences, Queensland University of Technology, Brisbane, Australia
caoson.tran@qut.edu.au

Abstract. This paper presents a design framework for artificial general intelligence (AGI). The approach is guided by a simple question: if we encounter an intelligent system, what could we observe? The answer is based on the idea that intelligence emerges from simple goal-driven interactive adaptability, and the process leads to emerging properties underlying complex behaviors of any intelligent systems. These properties in turn serves as design criteria for the construction of a network architecture of AGI which is proposed here for further investigations in future studies.

Keywords: Artificial general intelligence · Cognitive redundancy · Cognitive complexity · Administrative behaviors · Composition search · Network of mind · Hypergraph categories

1 Emerging Properties of Intelligence

What could we learn about intelligence? Simon [1] approached this question by introducing the concept of interface to study intelligent systems. It is essentially a collection of properties emerging out of the system's behaviors seeking to achieve specific goals. These properties in turn impose limits on what the system could or could not do. Using the interface, an observer could predict how the system behaves when facing specific events without knowing details of its internal reasoning processes.

Minsky [2] proposed another goal-based framework focusing instead on constructing internal processes underlying an intelligent system's behaviors. In this framework, the system's intelligence relies on a set of resources that could be combined in various ways to create processes corresponding to different behaviors. A central concept is resource management which deals with how resources, diverse in their representation and capability, are acquired, evaluated, allocated, combined, and used. The system's intelligence could then be observed through these administrative operations.

In this section, these concepts are integrated into a simple computational model to explore the idea that intelligence is closely connected to computational complexity created by the process of continuous managing resources to achieve goals. By looking at the interface of this process, the model discusses three emerging properties of intelligence: cognitive redundancy, cognitive complexity, and administrative behaviors.

T. Nakano (Ed.): BICT 2021, LNICST 403, pp. 215–229, 2021.
https://doi.org/10.1007/978-3-030-92163-7_18

1.1 Cognitive Redundancy and Complexity

The computational model views an intelligent system's behaviors as goal-driven processes that use resources to achieve goals when encountering specific events. These processes are represented by the following structure.

$$\{e \rightarrow g \rightarrow r\} \tag{1}$$

where e is an *event situation* which describes an event, g is a *goal situation* that describes the goals the system wants to achieve given e, and r is a *resource situation* describing resources the system uses to achieve goals. $\rightarrow$ is an operator that maps one type of operator to another type of operator. The detailed construct of e, g, r, and $\rightarrow$ can be ignored here since the abstraction is sufficient for the purpose of this paper. In other words, we only look at the situations' interfaces. Essentially (1.1.1) says that when an event happens, the system decides whether it should set goals or not. Given a set of goals, it proceeds to figure out the resources needed and how they could be used together to achieve the goals.

What could the tiny structure in (1.1.1) tells us about intelligence? The most obvious thing is that it describes an ideal situation in which we always know precisely what goals to set and what resources to use. But the picture is less clear when we have a circumstance in which event, goal, and resource situations interact to create complexity. Consider the following three scenarios.

$$\left\{\{e \rightarrow e_1\} \rightarrow^e \{\{e \rightarrow e_1\}, \{e \rightarrow e_2\}, \{e_1 \rightarrow e_2\}\}\right\} \tag{2}$$

$$\left\{\{g \rightarrow g_1\} \rightarrow^g \{\{g \rightarrow g_1\}, \{g \rightarrow g_2\}, \{g \rightarrow g_3\}\}\right\} \tag{3}$$

$$\left\{\{\{r \rightarrow r_1\}, \{r \rightarrow r_2\}\} \rightarrow^r \{\{r_1 \rightarrow r_2\}, \{r \rightarrow r_3\}, \{r_1 \rightarrow r_3\}, \{r_2 \rightarrow r_3\}\}\right\} \tag{4}$$

The structure in (2) could be described in the following process. First, the event situation e first encountered by the system leads to another event situatione_1. This in turn leads to a new *event space* created in the system's memory to store *configurations* in the form $\{e \rightarrow e_1\}$. The system now learns that there are more possible configurations of the event space, a realization represented by the structure on the right of (2). The updating process is represented by the operator $\rightarrow^e$ constructed as a term-rewriting operation on hyper graphs [3–5]. Similar scenarios could be observed for the *goal space* in (3) and the *resource space* in (4). The structure in (1) now expands into:

$$\left\{\left\{e, \rightarrow^e\right\} \rightarrow \left\{g, \rightarrow^g\right\} \rightarrow \left\{r, \rightarrow^r\right\}\right\}, \tag{5}$$

with {e, $\rightarrow^e$}, {g, $\rightarrow^g$}, and {r, $\rightarrow^r$} representing (2), (3), and (4) correspondingly.

The system's ability to update event, goal, and resource spaces creates a cognitive redundancy that imposes a limit on how flexible the system could be in term of creating connections and switching between configurations to adapt to changes. Figure 1 (left panel) visualizes the event space as a hyper graph, showing how diverse it has become just after 6 updates from the initial configuration in (2). This leads to an increase in cognitive redundancy observed in the rapid raise in the number of edge counts (Fig. 1,

right panel). From this perspective, the concept of cognitive redundancy could explain why models based on artificial neural networks, while performing well in some specific tasks, often failed or perform poorly in tasks found simple by humans [6]. The reason is that these models have rather low cognitive redundancy since their architectures are essentially extensions of (1) which has only one updating path.

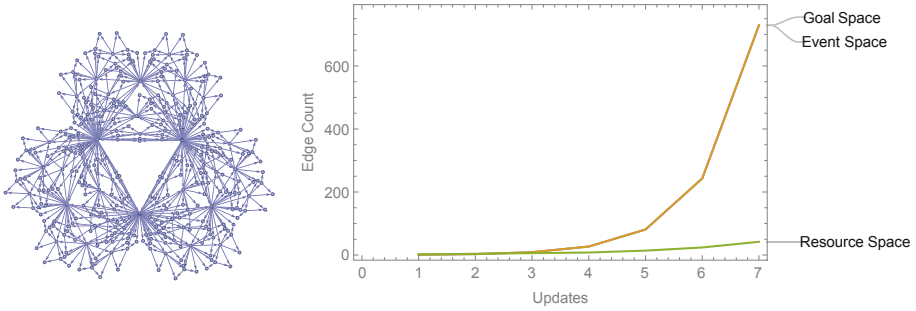

Fig. 1. Left panel: the hypergraph of the event space after 6 updates from the simple initial configuration $\{e \to e_1\}$. Right panel: an increase in cognitive redundancy as showed by a rapid increase in the number of edge count.

Cognitive redundancy, however, comes with a cost. With a space of many possible updating paths, the system must essentially perform a combinatorial task to select the desired path to act upon. Besides, the combinatorial space is much larger than that created by the hypergraph's edges since the graph does not reflect possible updating paths, instead it simply merges them into a reduced form. As a result, searching for a right combination of resources situations requires exploration of all possible path absent of effective administrative operations (Fig. 2, left panel). This combinatorial challenge leads to *cognitive complexity*, which corresponds to an exponential increase in the number of possible updating paths created by simple updating rules (Fig. 2, right panel). Cognitive redundancy and cognitive complexity are thus emerging properties that impose constraints on the system's ability to find paths to its goals.

1.2 Administrative Behaviors

What should the system do when facing cognitive complexity? A likely response is to seek for a balance between redundancy and complexity to achieve desired goals. This implies that there should be a mechanism to achieve this balance, which is essentially an administrative task. The most obvious thing for the system to do is to impose a set of constraints on the scope of the updating process. For example, it could impose limits on the number of updating steps or the type and scope of resource situation. Alternatively, it could simply restrict its attention to specific event, goal, and resource situations. For example, an organization could create an identity, focus on specific goals and work within specified budget plans to manage complexity [7].

The multipath evolution of the system hints at a more elaborate mechanism involving two phases. First, it could adopt a kind of analogical reasoning that could rapidly identify

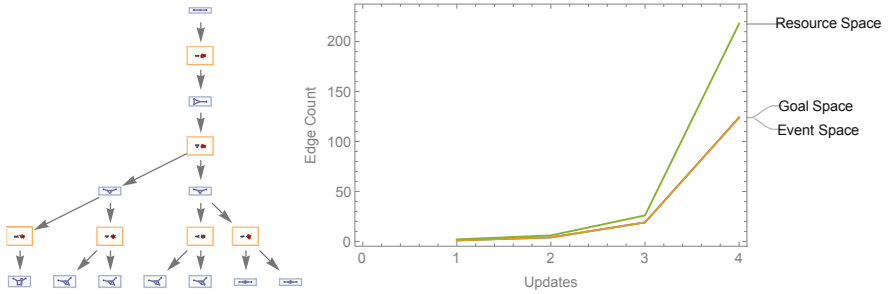

Fig. 2. Left panel: the multipath evolution of the resource space with each possible path pointing to a hyper graph having an increasing number of edges. Right panel: an increase in cognitive complexity in the event, goal, and resource spaces as reflected in the exponential increase in the number of edge counts in the multipath evolution.

equivalent combinatorial paths. Minsky [2] suggested that this type of analogy is used in human intelligence, but the general mechanism is unknown. That said, in a multipath system, analogical reasoning could be viewed as performing an operation similar to the Knuth-Bendix completion algorithm that seeks to reduce two updating paths to the same structure (Fig. 3) [5, 8]. Another possibility is the deployment of special type of resources that are essentially computational systems specializing in doing combinatorial tasks quickly. These resources thus help reduce the search space to a more manageable structure. An example is the use of a hybrid physical-biological computation network to perform massive parallel computing tasks [9].

In the second phase, the system performs domain specific credit assignment operations which are essentially weight assignment tasks. An intelligent system is thus likely to have a memory of different credit assignment methods corresponding to different configuration of event, goal, and resource situations [2]. And this implies acquisition of a large memory of decision processes made by diverse intelligent systems.

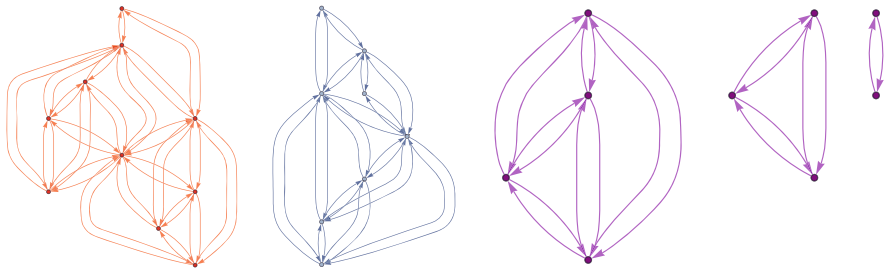

Fig. 3. Left panel: Knuth-Bendix completion rules for the event space. Middle panel: Knuth-Bendix completion rules for the goal space. Right panel: Knuth-Bendix completion rules for the resource space.

The model proposed here is far from a complete and accurate description of intelligence at an abstract level. It presents a computational way to investigate properties of intelligence that has novel architectural implications. The hypergraph structure with dynamic updating rules enables us to model intelligence at a level sufficiently abstract to accommodate different problem domains. Through the model, we discovered two important concepts, cognitive redundancy and cognitive complexity, and this in turn leads to the emergence of administrative behaviors as key mechanisms underlying the evolution of intelligence. Using these constructs as design criteria, the next section presents a design framework called network of mind (NoM) to demonstrate how complex intelligent systems could emerge from the simple task of dealing with redundancy and complexity during the search for paths to their goals.

2 The Architecture of NoM

Several conceptual constructs are used in NoM to represent three ideas. First, intelligence is simply considered as a process in which a system learns how to achieve goals by trying to be resourceful given specific circumstances. Being resourceful here means having the ability to combine and switch among different ways of doing things. Second, the learning process arises not only from internal capability but also external interaction with other systems. Finally, intelligence processes are implemented in a network structure that makes difference between layers of representation of knowledge rather superfluous while enabling massive parallel operations to deal with cognitive complexity in a fashion inspired by Nicolau et al. [9].

2.1 Key Constructs

Imagine a world in which things exist in construct called *entities*. Each entity is associated with a finite collection of *interfaces* through which it interacts with other entities. The interaction process takes place when an interface receives some input and sends out output. The interaction is feasible when the output sent out by one entity are accepted as input for at least another entity. Input and output could themselves be entities. Entities that do not have an interface are called uncertainties.

$$\text{Entity} = \{\{interface \rightarrow \{input, output\}\}\}_1^n \tag{6}$$

Some entities send out their output through the interface without having definite receiving destinations, while others deliberately send output to specific targets with the expectation of getting something back. An entity behaving like the latter are called an agent. Its interaction with other entities is called a probing process in which the agent uses a set of entities called models to extract information from the other entities through their interfaces. A model's output could be used as another entity's input, and the entity's output another model's input. A model, however, only works through a single interface, and an agent could use different models to probe the same interface of an entity. The agent also needs to update its models when no longer relevant or less effective.

$$\text{Model} = \{interface \rightarrow \{input, output\}\} \tag{7}$$

Each agent only pays attention to certain entities called entities of interest (EOI). The probing models thus play a dual role. First, they serve as a tool for the agent to discover an EOI's interfaces. Second, they are used as learning mechanisms that enable the agent to extract properties associated with the EOIs in a probing process. A property is essentially the result of a transformation of a probing process' output into constructs that remain invariant to the agent. From time to time an interface's output could be used as properties without further transformation. Different models learn about different properties of the same EOI.

When the agent wants to make changes to its EOI's properties, it needs to use some models to interact with the EOI's interfaces. This is called an affecting process in which the employed models and interfaces could be the same or different from the probing process' models and interfaces. When performing an affecting process, the agent may need to use several models at the same time to alter properties that could not be handled by a single model.

In this situation, the agent must perform a composition operation on the existing models to build a new model that could affect its EOI. Essentially, this process determines whether and how certain models could be combined to affect the EOI's properties. The composition operation consists of a parallel operator, a sequence operator[1], and a difference operator that work on multiple levels of model composition. The parallel and sequence operators work on finding ways to combine the models, while the difference operator checks how well the composition process has worked so far.

The agent could own some models and entities while having access to other models and entities through transactions made with other agents. To make this possible, the agent must have currency and suitable transaction models. Within the context of the society proposed here, it is assumed that transaction costs are zero. As a result, agents could gain access to each other's models, making the ownership cost of the models equal to zero unless they are unavailable when being requested. That said, adding transaction costs to the society would put more constraint on learning and make intelligence more dynamical as some agents will create better models and acquire more currency from selling their use to other agents. This would eventually lead to competition in building better probing models, creating diversity in the agents' probing capability. It is not unreasonable to think of a society where some agents develop models to attract other agents' attention. Whatever the situation, each agent uses an administrative process to check whether a model is available, how long it could be used for, how much it would cost, how to carry out a transaction, and how it could be accessed by the agent.

The operators act as an internal interface through which the agent learns over time, in a process called composition search to find out how to compose models to affect its EOIs' properties. Since it is possible that there could be more than one way to compose the models to achieve the result, this creating cognitive redundancy in the search process. This redundancy, however, requires the agent to deal with cognitive complexity arising from the task of finding the right model composition amongst many scenarios, especially with increasing number of models. As a result, complexity makes the composition search's solution at most a local optimum, thus leading to a diversity in

[1] Additional operators such as initialization, copying, merging, and terminating could be added to the operation when needed for complex situations.

the composition search's results even when the same search strategy is used. From time to time, the composition operation requires not only internal models, but also external ones acquired through transactions with other entities. The agent is said to be resourceful if it knows how to switch among different approaches to constructing the composition search.

The probing process could be represented by the following incidence matrix.

	$\boldsymbol{e_1}$	…	$\boldsymbol{e_l}$
$\boldsymbol{p_1}$	x_{11}	…	x_{1l}
. . .	…	…	…
$\boldsymbol{p_k}$	x_{k1}	…	x_{kl}

Fig. 4. The probing incidence matrix.

Essentially, each agent's probing process could be represented as a weighted hypergraph with it probing models' interfaces $\boldsymbol{p}_i$ as vertices and its EOI $\boldsymbol{e}_j$ as hyperedges (Fig. 4). The properties generated by the probing process are represented by the variable x which changes over time as the agent adjusts his models and the entities' interfaces are updated. x_{ij}, the weight of the vertex $\boldsymbol{p}_i$ is essentially a list of properties of entities $\boldsymbol{e}_j$ learned by the agent through model $\boldsymbol{p}_i$. In this context, an entity is a hyperedge consisting of weighted vertices that are essentially models used by the agent to construct its representation of the entity. The incidence matrix is thus a random matrix with distribution of its elements capturing the agent's probing process over time.

The agent recognizes an event when there are anticipated changes in its EOIs' property measures. Events could occur internally when the agent adjusts its probing process, causing changes in its assessment of the entities' properties. Events also occur externally when causes of changes come from without. The agent has a goal when it seeks to achieve specific property measures for its EOIs, and goals arise when the agent reacts to changes caused by events.

Intelligence arises when the agent tries to be resourceful in reaching specific goals that are set when it wants to interact with current EOI or new entities it never encountered before. The agent starts with an initial set of goals and proceeds to adjust them over time through two mechanisms. First, it reflects on how certain goals were achieved easily while others were more difficult to realize. Second, it learns how appropriate its goals are through communications with other agents.

Usually, the effecting process requires the composition operation as there could be many models involved in the probing process. This operation could be captured in the following incidence matrix (Fig. 5).

	e_1	...	e_l
a_1	y_{11}	...	y_{1l}
. . .	...	...	...
a_k	y_{k1}	...	y_{kl}

Fig. 5. The affecting incidence matrix.

The matrix's structure is like that of the probing process, but unlike x, y only takes the value of 1 or 0, indicating whether an affecting model a_i is used in the probing operation or not. Like the probing incidence matrix, the affecting matrix only indicates which models were used in the probing process while revealing little as to how the affecting process happened. This could be addressed by the introduction of a diagram wiring scheme and a composition matrix to represent the composition operation (Figs. 6 and 7).

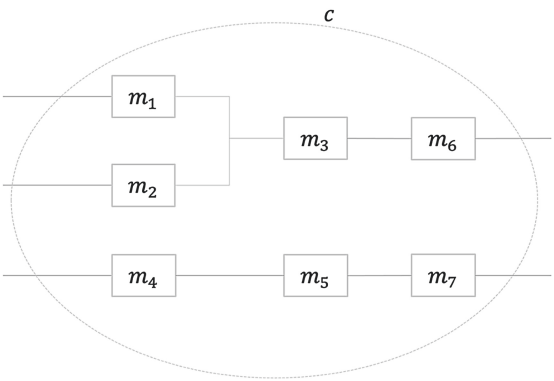

Fig. 6. A composition operation represented as a simple wiring diagram that consists of a parallel operator and a sequence operator working on 7 models. The box represents the model and the wiring the models' input and output. The relationship between the models in this composition operation could be captured by the hypergraph $\boldsymbol{c}$ for two reasons. First, all models would be used in the operation. Second, they could be combined in more than one way to achieve the same result. In this example, the hypergraph also consists of two directed graphs. This simple structure could be expanded to include other data operations such as initialization, termination, copying, merging, and feedback.

Theoretically, this operation could potentially be analyzed by using codesign theory and the concept of compact closed categories that essentially deals with composition

	$\boldsymbol{m_1}$	…	$\boldsymbol{m_k}$
$\boldsymbol{m_1}$	y_{11}	…	y_{1l}
. . .	…	…	…
$\boldsymbol{m_k}$	y_{k1}	…	y_{kl}

Fig. 7. A matrix representing to complement the composition hypergraph. In this matrix, $\boldsymbol{m}$ represents the models used in the composition operations, and $\boldsymbol{y}$ takes value of 1, 0 or nothing. When $y_{ij} = 1$, it implies the model m_i is composed sequentially with m_j in this order. When $y_{ij} = y_{ij} = 0$, it implies m_i and m_j are working in parallel in the same or different wiring operations. When y_{ij} has no value, it indicates there is no composition operation working directly on both models.

possibilities [10]. One crucial requirement for this approach to work is that the models' input and output must form preorders that enable the use of enriched categories. Intelligence, however, seems to be able to deal with things that do not seem to have this structure. A workaround would be equipping the agent with a mechanism to transform their models' inputs and outputs into preorders, thus enabling us to analyze the composition process more rigorously. Formally, this could be represented by the following proposition:

$$\bowtie (\mathcal{C}, \mathcal{E}, l) = P(\mathcal{E}) \Leftrightarrow (\mathcal{C}, \boldsymbol{I}, \otimes) iscompactclosed \quad (8)$$

Essentially, (8) states that as soon as the composition operation $\bowtie$ finds a combination of models $\mathcal{C}$ that generate desired properties $\mathcal{P}$ of some EOIs $\mathcal{E}$ under the constraint l, it could be concluded that this process be equivalent to making the symmetric monoidal category $(\mathcal{C}, \mathrm{I}, \otimes)$ a compacted closed category. What this proposition implies is that while evolution of intelligence may not be expressed in any precise mathematical formalism, the results of the evolution could be.

This proposition, however, says little about the dynamics of the composition search over time, especially when the agent must perform several operations at once to deal with different EOIs. This issue could be addressed by analyzing the evolution within the framework of hypergraph categories which represent the process as a set of $(\mathcal{C}, \mathrm{I}, \otimes)$ nested within and connected to each other via a Frobenius structure [11]. Representing the composition process in the language of hypergraphs and categories would enable appropriate use of algorithms such as the Knuth-Bendix completion rules while dealing with increasing cognitive complexity.

2.2 Learning and Intelligence

Initially, each agent is endowed with a set of EOIs a set of models to probe and affect the EOIs. To better organize knowledge as the number of EOIs increase over time,

during the probing process, the agents seek to construct relationships between entities by transforming the hypergraph's incidence matrix into the following adjacency matrix (Fig. 8).

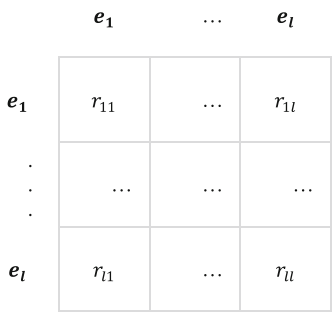

Fig. 8. The entity relationship matrix.

The value of r_{ij} is derived from the properties generated by the probing results of two entities e_i and e_j by all models available to the agent (Fig. 4). r is thus a multi-dimensional variable representing what the agents could learn about connections among the EOIs. For example, one value of r_{ij} tells the agent whether two people e_i and e_j come from the same family while another value capture their roles in the family. Looking at these values, the agents could build a family structure that consists of e_i and e_j. In another example, it could learn that both e_i and e_j are cities belonging to a larger entity e_k which is a state or a country.

In general, the agents could construct three kinds of relationship based on the measures captured in the adjacency matrix. The first type enables the agent to group entities into categories according to similarity in their interfaces. In other words, if two entities response to the probing by a similar set of models, they would belong to the same category. The second type looks at relationship within a category in more details by linking together entities that share similar properties. The third type enables the agent to build invariant structural relations that could be applied to different entities, regardless whether they belong to the same category or not, and this could be achieved via exploiting relations among properties. For instance, the agents, by looking at specific properties of e_i and e_j through the probing process, could build a structure to describe their relationship as family members (husband, wife, daughter, dog etc.). they could reuse this structure when encountering a similar context when probing other EOIs, a behavior that has been observed in experiments with mouse brain [12]. While the agents could build structures of relationships among his EOIs, the capability to do so varies as each agent has different probing models with different quality.

Having the capability to find connections and build structural relationships among EOIs, the agents only pay attention to entities likely connected to his current EOI. By doing this, it could learn to anticipate and response quickly and systematically to events. Suppose that there is an event affecting properties of an entity e_i, and the agents know that by using its knowledge of relationships among EOI, e_j and e_k will also be affected.

With this anticipation, they could know how to prepare response to changes in not only e_i but also e_j and e_k.

Even though the agents pay attention to entities related to its EOI, their attention level differs from each other. Some agents care more about specific properties of itself. Other agents start with several interests but over time become slack, while a few keeps increasing their learning capability. That said, when the agents encounter a new entity, they would use all available models to probe it. Results generated by the probing process would give the agents three choices. The first choice is to add the entity to its list of EOIs if there is a hint of a relationship between the entity and some of the existing EOIs. The second option is for the agent to broadcast the encounter to other agents through the administrator. In case some other agents have the right model to deal with the entity, the agent could proceed to make a transaction, if needed. The third choice is to simply ignore it.

When it comes to the composition operation, the agents follow four principles. First, the EOI's input and properties must form or could be transformed into preorders as this enables the agent to establish some sort of hierarchical relationships among the input and the properties. These relationships would in turn enable the composition operation to learn roughly but quickly which combination of inputs or properties will likely be feasible. Second, they do not optimize but only seek heuristically to find approximately right solutions when facing complexity, such as creating changes in an EOI's properties that are within an acceptable limit. This is to avoid the trouble of having to deal with cognitive complexity and to also utilize what the agents know about input, properties and emerging structures learned from reflection.

Third, the agents operate on the model interfaces rather than the actual model constructs. In other words, the agents, assuming that they have full knowledge of the composition search's behaviors, perform simulations of the operation instead trying to use the actual models all the time. By doing this, the agents would avoid the issue caused by unavailability of some models while still operating within a margin of safety. Finally, the composition search could benefit from transactions among the agents, especially when there is a competition among them with the introduction of currency.

To sum up, in the society just constructed, the agents start learning by using available models to probe their EOI through their interfaces. The probing process generates properties associated with the EOI, and these properties enable them to acquire knowledge of the world around them. But probing is not sufficient as they also want to change the properties, and when no existing model could be used, the agents perform a composition operation to create new models from what he has. This composition operation in turn requires an administrative operation to handle communication and transaction involved in model acquisition.

Properties and changes in their measures give rise to events and goals which in turn create a demand for intelligence. In addition, properties enable the agents to build relationships among entities, thus enabling them to learn about these entities not just through knowing individual properties but also through understanding their structural relations. As a result, they understand and respond to events with more complex dynamics. The goal setting capability is thus enhanced accordingly, with intelligence pushed to a new level. In the long term, what matters most to the development of intelligence is the

administrative capability to effectively deal with composition search under increasing transaction costs and cognitive complexity.

It is thus sensible to say that one agent is more intelligent than another if the former performs the probing process, the composition search and the administrative function faster and more effectively than the latter does. To achieve these results, the agents use three mechanisms. The first mechanism is a series of snapshots of the learning process he creates when at least one of the following four situations happens. First, the agents create new affecting models through the composition operation. Second, the agents add new entities to his list of EOIs. third, there are changes in the existing entities' properties, and the changes are so significant that they lead to different structure relationships between the entities. Finally, the agents may take a series of snapshots to learn about property relationship which informs whether there is a connection between properties of a specific entity, thus pointing to implications that may benefit the composition operation on the probing models.

The snapshots are essentially entities which would collectively form the agents' memories which could be stored in and retrieved from the following structure. From time to time, an agent could share some of their snapshots to other agents, leading to message exchange, reevaluation of learning capability, and transactions when the snapshots contain models that the other agents need to work on existing and new entities.

$$\left\{e_i^{mt} \rightarrow \left\{p^{mt}, x^{mt}, cp^{mt}, a^{mt}, y^{mt}, ca^{mt}\right\}\right\}_{i=1}^{l}, \tag{9}$$

where

$$p^{mt} = \left\{p_1^{mt}, \ldots, p_k^{mt}\right\}, x^{mt} = \begin{bmatrix} x_{11}^{mt} & \cdots & x_{1k}^{mt} \\ \vdots & \ddots & \vdots \\ x_{k1}^{mt} & \cdots & x_{kk}^{mt} \end{bmatrix}, cp^{mt} = \begin{bmatrix} cp_{11}^{mt} & \cdots & cp_{1k}^{mt} \\ \vdots & \ddots & \vdots \\ cp_{k1}^{mt} & \cdots & cp_{kk}^{mt} \end{bmatrix} \tag{10}$$

$$a^{mt} = \left\{a_1^{mt}, \ldots, a_l^{mt}\right\}, y^{mt} = \begin{bmatrix} y_{11}^{mt} & \cdots & y_{1l}^{mt} \\ \vdots & \ddots & \vdots \\ y_{l1}^{mt} & \cdots & y_{ll}^{mt} \end{bmatrix}, ca^{mt} = \begin{bmatrix} ca_{11}^{mt} & \cdots & ca_{1l}^{mt} \\ \vdots & \ddots & \vdots \\ ca_{l1}^{mt} & \cdots & ca_{ll}^{mt} \end{bmatrix} \tag{11}$$

Essentially, an agent *m's* snapshot *t* is a list in which element *i* describes an association between the entity e_i and the corresponding probing models (***p***), property matrix (***x***), composition matrix for the probing models (***cp***), affecting models (***a***), affecting matrix (***y***), and composition matrix (***ca***) used by *m* in the probing and affecting process. The snapshots are a rough representation of what happened in the past since the agent only records them in certain situations, thus making a trade-off between perfect memory and computational complexity.

The second mechanism is credit assignment which essentially ranks composition searches by reflecting on past snapshots. When probing a new EOI, the agents would favor probing models and composition solutions that have higher credit assignment. One simple strategy is to extract and give higher credit to models that remain invariant in the probing and composition operations over the reflection period. If these models turn out to be effective, they will in turn reinforce their credit ranking. This circle will allow the agents to reduce search time and avoid unnecessary computational complexity when

probing new EOIs or performing composition operation. The strategy, however, may put them in a difficult situation when they overuse the credit assignment without considering the context applied to. Regardless of the strategy used, the credit assignment essentially deals with estimating the following probability

$$P(\bowtie (\mathcal{C}, \mathcal{E}, l, \mathcal{P}(\mathcal{E}))| \bowtie (\mathcal{C}^*, \mathcal{E}^*, l^*, \mathcal{P}(\mathcal{E}^*))), \quad (12)$$

where * denotes a degree of similarity between elements of other composition operations and that of the current operation.

The third mechanism is parallel processing of the probing operation, the administrative function and the credit assignment process. The idea is rather simple. Doing things in parallel, if the outcomes could be combined in the right way, would speed up the computational process significantly, especially when the number of tasks increases. Performing the composition search concurrently, however, may be more difficult unless the agents have enough redundancy in its architecture to support the operation. The reason is that as some models are used during one instance of composition search, they will not be available for use in another instance unless the agents have some copies ready for the operation. This architecture issue is discussed in the next section.

2.3 Network Structure

Having developed the constructs for the learning process and the mechanisms to create memories that enable agents to be more intelligent, they need a place to store the constructs and operate the mechanisms. What would the architecture of such place look like? The first feature of the architecture would be a network-like structure, since the learning process relies on connections between the probing, affecting and administrative operations. Organizing these operations around networks would enable them to be carried out in parallel efficiently.

The second feature would be a separation of the architecture into two parts in term of operational speed and complexity. The first part consists of computing nodes focusing on coordinating probing, affecting and administrative tasks in parallel. The nodes do not carry out the operations but perform a supervision role instead. This part also contains nodes linked to the actual models used in the operation. They serve as a temporary memory that keeps information about the EOIs and model interfaces. This enables the supervisory functions to work on these models and entities at rapid pace and, when possible, in parallel fashion.

This process works with the principle that knowing what comes in and what comes out through the interface would be reasonably enough to make judgements as to whether the probing, affecting and administrating operations would work or not. Besides, the nodes always make the interface available when needed by these operations, while actual models may not. Doing so, the agent is sacrificing uncertainty in access and slow precision offered by operations on actual models for high availability and rapid approximation delivered by implementation on the interfaces.

The second part is where snapshots and the actual models' constructs are kept. The purpose of keeping these entities in a separate place is twofold. First, it will over time require more and more permanent memory capacity and putting these in the first part

of the architecture will create a server constraint temporal memory needed for crucial operations. Second, working directly on the models and wait for actual results is much slower than working on their interfaces with the presumption that the interfaces would do what the actual models would do, especially when many models get involved in the operations. Finally, it is less computationally complex to have a verification on actual model at the end of an operation rather than at each step in the process. That said, some models that are used frequently and have simple constructs could still be kept in the first part, especially those serving as elementary constructs upon which other models are built.

Next, what languages do the operations use to communicate and coordinate? For the first part of the architecture, it must be simple enough to enable the operations to be carried out rapidly in parallel without causing computational overhead. A binary language appears to be a sensible solution since the elements of this language are just bits. And the rules for this language should be based on basic logical and bitwise operations. Thus, at the center of the architecture are binary networks that perform the probing, affecting, and administrating operations. Recent developments in studies of binary neural networks pint to the potential of this computational model [13]. The next challenge is to construct appropriate coding for the network. One could imagine watching such networks at work when some nodes become active, they represent models that are being used in the operations. And when they become active together, they are used in a composition search, and we could say that when they fire together, they wire together. These nodes are not models that perform complicated computations, instead they serve as an interface to those models.

For the constructs stored in the second part, they could be represented in any languages that satisfy three criteria. First, the language must have an interface to the binary language used by the first part, thus enabling communications between the actual models and the representative interfaces. Second, the language must support execution of parallel operations on the constructs initiated by the binary network. Especially, it must enable parallel operations to be carried out without causing unnecessary computational overhead. Finally, it must has a common interface to enable the agents to exchange messages and make transaction with each other, and these tasks could be performed in a parallel implementation made possible by development of mature and robust message-based concurrent programming language such as Erlang [14].

Knowing how these constructs should be implemented and what performance would be expected from such implementation would allow validation of the architecture presented here. What this paper hopes to accomplish is to present new conceptual elements as core design artifacts of intelligence. Being a proposal of new ideas, the paper has not addressed issues such as how credit assignments evolve and how the agents deal with cognitive complexity during the composition search process over time. We believe that the best avenue to seek answers for these questions is in actual implementation of the architecture, and this our goal in the next endeavor.

3 Conclusion

This paper approached the study of intelligence from a network computation perspective based on abstract structures that could be applied to different contexts rather than just

specific tasks. These structures were constructed by observing emerging properties of intelligence that essentially deals with cognitive redundancy and complexity. Using the concept of agents, the study is also concerned with social aspect of intelligence. As a result, agents would be expected to exhibit administrative behaviors sooner or later in the evolution of its intelligence as they learn how to deal with cognitive redundancy and complexity effectively. From a design perspective, the basis to support these behaviors is a network architecture amenable to structural analysis based on network theories and hypergraph categories. Key conceptual constructs of this architecture were developed and discussed with the purpose of creating a sound basis for further investigations in future studies.

References

1. Simon, H.A.: The Sciences of the Artificial. MIT Press, Cambridge, Mass (1996)
2. Minsky, M.: The Emotion Machine: Commensense Thinking, Artificial Intelligence, and the Future of the Human Mind. Simon & Schuster, New York (2006)
3. Berge, C.: Hypergraphs: Combinatorics of Finite Sets. Elsevier (1984)
4. Baader, F., Nipkow, T.: Term Rewriting and all that. Cambridge University Press (2012). https://doi.org/10.1017/CBO9781139172752
5. Gorard, J.: Some quantum mechanical properties of the wolfram model. Complex Syst. **29**(2), 537–598 (2020)
6. Zador, A.M.: A critique of pure learning and what artificial neural networks can learn from animal brains. Nat. Commun. **10**(1), 1–7 (2019)
7. Simon, H.A.: Administrative Behavior: a Study of Decision-Making Processes in Administrative Organizations. Free Press, New York (1997)
8. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras, pp. 342–376. Springer, Automation of Reasoning (1983)
9. Nicolau, D.V., et al.: Parallel computation with molecular-motor-propelled agents in nanofabricated networks. Proc. Natl. Acad. Sci. **113**(10), 2591–2596 (2016)
10. Censi, A.: A mathematical theory of co-design. arXiv:151208055 (2015)
11. Fong, B., Spivak, D.I.: Hypergraph categories. J. Pure Appl. Algebra **223**(11), 4746–4777 (2019)
12. Whittington, J.C.R., et al.: The Tolman-Eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. Cell **183**(5), 1249–1263 (2020)
13. Valencia, R., Sham, C.W., Sinnen, O.: Using Neuroevolved Binary Neural Networks to solve reinforcement learning environments. In: 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp. 301–304. IEEE (2019)
14. Armstrong, J.: Making reliable distributed systems in the presence of software errors (Doctoral dissertation) (2013)