

General problem solving with category theory.

Francisco J. Arjonilla, Tetsuya Ogata

September 15, 2017

Abstract

This paper proposes a formal cognitive framework for problem solving based on category theory. We introduce cognitive categories, which are categories with exactly one morphism between any two objects. Objects in these categories are interpreted as states and morphisms as transformations between states. Moreover, cognitive problems are reduced to the specification of two objects in a cognitive category: an outset (i.e. the current state of the system) and a goal (i.e. the desired state). Cognitive systems transform the target system by means of *generators* and *evaluators*. Generators realize cognitive operations over a system by grouping morphisms, whilst evaluators group objects as a way to generalize outset and goals to partially defined states. Meta-cognition emerges when the whole cognitive system is self-referenced as sub-states in the cognitive category, whilst learning must always be considered as a meta-cognitive process to maintain consistency. Several examples grounded in basic AI methods are provided as well.

1 Introduction

Unification of Artificial Intelligence (AI) has been a long pursued goal since the early days of computation, however it remains elusive. Here we propose a novel framework that lays the grounds of a formal description of general intelligence. This framework is a generalization and formalization of the concepts presented in [1], which claimed that cognitive systems learn and solve problems by trial and error: the attempts at reaching a goal, termed variants, are first *generated* by some heuristics and then assessed. The

better the models a cognitive system has, the more accurate the generation of variants is and the fewer the mistakes made. Random variants are inevitable when there are no models, and mistake-free variants are used when the model is complete.

Many theories have been proposed that attempt general problem solving, yet the final goal of achieving human-level intelligence has been unsuccessful. Some authors have proposed guidelines and roadmaps for this search [17, 12]. One of the earliest theories that focus on general problem solving was proposed in [14] and focuses in decomposing recursively goals in subgoals and separating problem content from problem solving strategies. It evolved later into the cognitive architecture SOAR [10, 11] as an example of a unified theory of cognition [15]. Furthermore, [8, 9] proposed a general theory of universal intelligence that combines Solomonoff induction with sequential decision theory realized in a reinforcement learning agent called AIXI. However, AIXI is incomputable [22] and relies on approximations. A formal measure of general intelligence was proposed in [13] and related it to AIXI.

On the other hand, category theory has been seldom applied to modeling general cognitive processes. Rather, these efforts have been directed towards knowledge representation and specific cognitive processes. [4] proposed a general framework for representation based on category theory to advance in the understanding of brain function. Other authors have focused in modeling the semantics of cognitive neural systems [6], describing certain aspects of cognition such as systematicity [5, 16], or modeling theories about human consciousness such as Integrated Information Theory [20, 19].

2 Category Theory

Category theory is a relatively new field of mathematics and the theory of structure *par excellence*. It raises the importance of relations between objects to that of the objects themselves. We now sketch the categorical entities that stand as the formal skeleton of the cognitive theory developed in this paper.

Definition: A category \mathcal{C} consists of two entities:

1. A class $Obj(\mathcal{C})$ of elements. These elements are called *objects*. An object $A \in Obj(\mathcal{C})$ is also written $A \in \mathcal{C}$.
2. Morphisms (Also maps or arrows): For each $A, B \in \mathcal{C}$, a hom-set $hom_{\mathcal{C}}(A, B)$ whose elements $f \in hom_{\mathcal{C}}(A, B)$ are called the *morphisms* from A to B . A is called the *domain* of f and B the *codomain*. f is also written $f : A \rightarrow B$ or f_{AB} . The class of all morphisms in \mathcal{C} is denoted as $Mor(\mathcal{C})$.

With the following properties:

1. For each object $A \in \mathcal{C}$, there is a morphism $1_A \in hom_{\mathcal{C}}(A, A)$ called the identity morphism with the property that, for any morphism $f \in hom_{\mathcal{C}}(A, B)$, then $1_B \circ f = f \circ 1_A = f$.
2. Composition: If $f \in hom_{\mathcal{C}}(A, C)$ and $g \in hom_{\mathcal{C}}(C, B)$, then there is a morphism $g \circ f \in hom_{\mathcal{C}}(A, B)$. $g \circ f$ is called the composition of g with f .
3. Composition is associative: $f \circ (g \circ h) = (f \circ g) \circ h$. □

There is no restriction on what elements can \mathcal{C} hold. In this article we will use small categories, i.e. $Obj(\mathcal{C})$ and $Mor(\mathcal{C})$ are sets. The elements of \mathcal{C} may be abstract mathematical entities, daily objects or even other categories. In the latter case, the morphisms between categories receive a special treatment and are called functors:

Definition: A (covariant) functor $F : \mathcal{C} \rightarrow \mathcal{D}$ is a morphism between categories \mathcal{C}, \mathcal{D} with the following two components:

1. A function $F : Obj(\mathcal{C}) \rightarrow Obj(\mathcal{D})$ that maps objects in \mathcal{C} to objects in \mathcal{D} .

2. A function $F : Mor(\mathcal{C}) \rightarrow Mor(\mathcal{D})$ that maps morphisms in \mathcal{C} to morphisms in \mathcal{D} .

With the following properties:

1. If $A, B \in \mathcal{C}$, F maps a morphism $f : A \rightarrow B$ to $Ff : FA \rightarrow FB$.
2. Identity is preserved: $F1_A = 1_{FA}$.
3. Composition is preserved: $F(g \circ f) = Fg \circ Ff$. □

Other important concepts in category theory are natural transformations, limits and adjunctions, which are not needed to introduce the concepts presented here.

3 Core

Let us specify an *independent system* as an entity that has no relationship whatsoever with other systems. This definition grants self-containment to systems that will undergo cognitive processing. Independent systems are abstract, but will serve as an idealization that will ease the study of non-independent systems. Moreover, we assume that they abide at exactly one state in any given context. We will also refer to cognitive systems. Cognitive systems will only intervene on an independent system when the latter fails to fulfill the desired goal. The other assumption is that the purpose of cognitive systems is to solve problems specified as states that fulfill some condition. We now present the key concepts along with some examples grounded on AI.

3.1 Cognitive categories

Cognitive categories are the cornerstone of this framework. The generality of category theory enables to apply this framework in a wide variety of applications. We will first give the definition of cognitive category and then study the properties.

Definition: A cognitive category \mathcal{S} is a category that satisfies $|hom_{\mathcal{S}}(A, B)| = 1$ for all $A, B \in \mathcal{S}$. □

All objects in cognitive categories are initial and terminal objects. Also, all morphisms are invertible

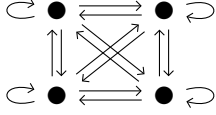


Figure 1: A cognitive category consisting of 4 states. The number of morphisms is $4^2 = 16$.

and there are exactly $|Obj(\mathcal{S})|^2$ morphisms. Let us create an intuition for cognitive categories. The objects $Obj(\mathcal{S})$ of a cognitive category \mathcal{S} associated to an independent system, which we will also denote as \mathcal{S} , correspond to each of the possible unique states that the system can be found at. On the other hand, $hom_{\mathcal{S}}(A, B)$ with $A, B \in \mathcal{S}$ represents a transformation from state A to state B . Other than that, we leave the specific mechanics of transformations undefined. We will justify that this transformation is unique as follows: It is always possible to conceive a transformation between any two states by replacing the system in state A with another undistinguishable system in state B . For practical purposes, transformation and replacement are equally effective because the relationships between states of the system remain equal. Therefore $|hom_{\mathcal{S}}(A, B)| \geq 1$. Now, consider any $t_1, t_2 \in hom_{\mathcal{S}}(A, B)$. Since A and B are states, the effects on the system of any transformation $t_1 \in hom_{\mathcal{S}}(A, B)$ will be undistinguishable from any other transformation $t_2 \in hom_{\mathcal{S}}(A, B)$, hence $t_1 = t_2$ and so, $|hom_{\mathcal{S}}(A, B)| = 1$. Thus, the category of states of an independent system is a cognitive category. With this intuition in mind, identity morphisms are null transformations and composition of $t_1 : A \rightarrow B$ and $t_2 : B \rightarrow C$ is a direct transformation $t_{21} : A \rightarrow C$. Morphisms in cognitive categories allow for the consideration of atomic transformations between single states, but they are less useful when considering real world problems because transformations are generally defined as operations rather than a replacement of one specific state into another.

3.2 Cognitive problems

The next step is to characterize the functionality of a cognitive system. We assumed that an independent

system \mathcal{S} may only be found in one of its possible states at any one time. Consider that this state is $O \in \mathcal{S}$ and consider another state $T \in \mathcal{S}$ that holds some desired condition. We will refer to O as the *outset* and T as the *goal*. The cognitive system receives these states as fixed: O is given by the current state of the independent system and T is given externally. The role of the cognitive system is to find some method or operation that transforms the independent system to a goal-complying state, and optionally executing this operation. Once the cognitive system transforms the outset state into the goal state, it becomes superfluous until another cognitive problem is presented.

We can define for now a restricted definition of cognitive problem that is valid when there is no uncertainty, which is covered in Section 3.4 where O and T will be generalized to partially known outset and abstract goals by considering sets of outset and sets of goals.

Definition: A *deterministic cognitive problem* B is a triple (\mathcal{S}, O, T) , where \mathcal{S} is a cognitive category, $O \in \mathcal{S}$ is the outset and $T \in \mathcal{S}$ the goal. \square

The solution to cognitive problems involves two stages, as mentioned above:

First, find a morphism t that guarantees that, if followed, the solution will be reached. Since $|hom_{\mathcal{S}}(O, T)| = 1$, t is the singleton morphism in $hom_{\mathcal{S}}(O, T)$ and is fully defined by $t \approx B = (\mathcal{S}, O, T)$. However, the cognitive system will not be able to provide t reliably unless stored as prior knowledge, which makes the base of model-based cognitions, as we will see in section 3.3.

$$O \xrightarrow{?} T$$

Second, follow t to actually transform \mathcal{S} from O to T . Let us analyze further this stage. Assume that \mathcal{S} is in state O and that there is a morphism $t : O \mapsto X$, with X unknown, that is, the cognitive system knows the outset O and knows that there is a morphism t whose domain is O , but has no knowledge about what state X will be reached if morphism t is followed. The only way of finding X is by following t . However, there is an important drawback that originates from transforming \mathcal{S} : The cognitive

system might not have the ability to return \mathcal{S} to O after having followed t , that is, it might be incapable of solving (\mathcal{S}, T, O) . Graphically,

$$O \xrightarrow{t} ?$$

The rest of the paper builds on how to find t under various conditions and shows through examples how this formulation is sufficient to describe a variety of AI methods.

Example: A Turing machine [21] is a formal model of computation that manipulates symbols in an infinite string on a tape following a table of rules. Mathematically, it is a 7-Tuple $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ [7] where the symbols denote, respectively, Q : states (not to confuse them with states in independent systems), Γ : tape alphabet symbols, b : blank symbol, Σ : input symbols, δ : transition function, q_0 : initial state and F : Accepting states. However, the control element of Turing machines alone lack some characteristics of independent systems, specifically that strings are not considered part of the state of the Turing machine. Actually, the only discernible change between the initial and halting conditions is that the internal state changes from q_0 to $q_G \in F$. Hence the states of the independent system do not need to describe the Turing machine. Moreover, we will assume that it always halts and only consider the initial and halting contents of the tape. So, we define the cognitive category of Turing machine strings \mathcal{S}_{TM} whose objects are the set of all possible strings in the tape. The morphisms replace one string with another, but hold no information about what processes lie behind this transformation. That is the job of the generators: A Turing machine is a component of a cognitive system, specifically a generator, that operates on the tape and transforms one string into another following the instructions set by its transition function δ . As purely computational machines, Turing machines solve the second stage of cognitive problems as stated above. \square

One more comment: there is no single way of designing a cognitive category. It depends on how a cognitive problem is best described. For instance, we could have described instead the cognitive states of a Turing machine in the previous example as the string

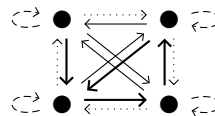


Figure 2: An omnipotent set of 4 generators over a cognitive category with 4 states. Dashed: Do not transform. Dotted: Cycle through all states.

after executing each instruction. By way of composition of morphisms between the initial tape and the halting tape, this category can be reduced to the previous.

3.3 Generators

Morphisms in cognitive categories are well suited to study the atomic transformations of independent systems, but they are most useful for theoretical analyses. Generally, it is not possible to transform freely a system. Rather we are constrained by a limited set of operations. In practice, a cognitive system will perform operations on an independent system regardless of its outset, with the resulting state depending on the operation and the outset. Generators are cognitive solvers that produces transformations for any state of \mathcal{S} in the scope of cognitive categories and stand as the first abstraction of cognitive processes.

Definition: A generator $gt : Obj(\mathcal{S}) \rightarrow Obj(\mathcal{S})$ over a cognitive category \mathcal{S} is an endomorphism in the category of sets. \square

gt stands for generate and transform. Generators generalize transformations defined by a morphism to the whole set of possible states. Intuitively speaking, a generator is the application of an operation over \mathcal{S} from an unspecific outset. More specifically, it provides the morphism that is equivalent to running the operation conveyed by the generator, and then follows the morphism. Generators cover the first stage of problem solving as stated in section 3.2. An isomorphic function is $g : Obj(\mathcal{S}) \mapsto hom_{\mathcal{S}}(O, T)$ such that g is equivalent to gt except that it does not follow the morphism. If a state $X \in \mathcal{S}$, the relation between g and gt is $gt(X) = \text{codom}(g(X))$. We will use g or gt depending on which one fits best, knowing

that they are isomorphic.

$$O \xrightarrow{t_{O=g(O)}} T = gt(O)$$

The elements of g are morphisms in \mathcal{S} that have the same properties as set functions (because they are set functions). Specifically:

- g provides one morphism for each state in \mathcal{S} .
- g is either an automorphism or both a non-injective and non-surjective morphism in the category of sets: there are generators that cannot transform \mathcal{S} to every state.

Example: Consider the cognitive category \mathcal{S}_M of mathematical statements and a cognitive problem $B = (\mathcal{S}_M, O \approx (x-1)^2 = 0, G \approx x = 1)$. The solution involves finding a function $\theta_M : (\mathcal{S}, O, G) \rightarrow g$ with $g : (x-1)^2 = 0 \mapsto x = 1$. θ is well known for B (for example, [2]) and computationally implemented with Computer Algebra Systems. θ_M recognizes O as a polynomial, providing and calling a one-variable polynomial solver for the generator g . Alternatively, a student θ_H provides g by writing a sequence of steps required to solve for x . \square

Consider the set \mathcal{G}_S of all possible generators in \mathcal{S} . With respect to its cardinality, if $n = |\mathcal{S}|$ then for every object $A \in \mathcal{S}$ there are n possible morphisms, hence $|\mathcal{G}_S| = n^2$. In general, cognitive systems will not have access to \mathcal{G}_S , but rather to a subset of \mathcal{G}_S that constitute the set of operations over \mathcal{S} available to the cognitive system. Let us study a special case of $G \in \mathcal{G}_S$ that greatly simplifies the study of generators.

Definition: An *omnipotent* set of generators over \mathcal{S} is a set of generators $G \subset \mathcal{G}_S$ that can generate any morphism in \mathcal{S} . \square

Example: Consider an independent system with n possible states associated to the cognitive category $\mathcal{S} = \{S_1, \dots, S_n\}$. Consider as well a cognitive system with a unique generator g over \mathcal{S} that cycles over all states:

$$g(S_i) = \begin{cases} S_{i+1} & \text{if } i \in \{1 \dots n-1\} \\ S_1 & \text{if } i = n \end{cases}$$

An omnipotent set of generators is constructed by composing g with itself iteratively: $G = \{g_i = g \circ \dots \circ g | i \in \{1 \dots n\}\}$ \square

Given any outset O , a cognition with an omnipotent set of generators can transform it to any other state and thus take complete control of \mathcal{S} .

Theorem: Consider an omnipotent cognition over a cognitive category \mathcal{S} with a set of available generators $G \subset \mathcal{G}_S$. Then, $|G| \geq |\mathcal{S}| = n$.

Proof: The number of morphisms that each generator can produce is equal to $n = |\mathcal{S}|$. Taking the definition of cognitive category, the number of morphisms in \mathcal{S} is n^2 . Therefore, covering the full set of morphisms in \mathcal{S} requires a minimum of $n^2/n = n$ generators. \square

We will refer to G as a reduced set of generators when $|G| = n$. These generators have interesting properties:

1. For each $t \in Mor(\mathcal{S})$ such that $t : A \rightarrow B$ there exists a unique generator $g \in G$ such that $g : A \mapsto B$.
2. For each $A \in \mathcal{S}$, and for each $g_1, g_2 \in G$, if $g_1(A) = g_2(A)$ then $g_1 = g_2$.

The second property states that given an outset O , there exists one generator for transforming to each state T . However, the opposite is not true: The number of generators that transform each O to a fixed T range from 0 to n . Generators are more interesting when they transform any outset to a small number of states, which leads to the next definition:

Definition: A *purposeful* generator g_T is the contravariant hom functor $\text{hom}(\cdot, T) : \mathcal{S} \mapsto \mathbf{set}$ defined by $g_T(O) = \text{hom}(\cdot, T)(O) = \text{hom}(O, T) = \{t_{OT}\}$. \square

We leave the analysis of the morphism component of this functor for future works. The *generate and transform* variety of g may also be used for an equivalent but simpler definition:

Definition: A *purposeful* generator gt_T is the constant function $gt_T(O) = T$ where $O \in \mathcal{S}$. \square

The latter definition clearly shows that purposeful generators are strongly related to states in the cognitive system. This is consistent with the fact that the minimum number of generators required in an omnipotent cognition matches the number of states

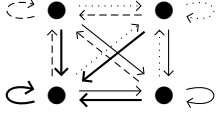


Figure 3: The canonical set of generators over a cognitive category with 4 states. Morphisms with the same line style correspond to elements of the same generator.

in \mathcal{S} . Actually, it is possible to build an omnipotent cognition with purposeful generators exclusively from an omnipotent cognition with non-purposeful generators: Consider a cognition with an initial omnipotent set of non-purposeful generators G . Then, for every outset O and every goal T , there is a generator $g_i \in G$ such that $g_i(O) = t_{OT}$. Now, consider a generator g_T that uses the appropriate non-purposeful generator for each possible outset, such that

$$g_T(X) = \begin{cases} g_i(X) = t_{O_1T} & \text{if } X = O_1 \\ \vdots \\ g_j(X) = t_{O_nT} & \text{if } X = O_n \end{cases}$$

g_T is indeed a purposeful generator because $gt_T(X) = T$ for all $X \in \mathcal{S}$. We can repeat this argument for each state in \mathcal{S} and therefore build a set of n distinct purposeful generators:

Definition: The *canonical* set of generators for the cognitive system \mathcal{S} is the set $G_{\mathcal{S}} = \{g_X = \text{hom}(\cdot, X) | X \in \mathcal{S}\}$. \square

$G_{\mathcal{S}}$ is a reduced set of generators because $|G_{\mathcal{S}}| = |\text{Obj}(\mathcal{S})| = n$. Moreover, $G_{\mathcal{S}}$ is unique: Consider a reduced set of purposeful generators $G'_{\mathcal{S}}$ that is built using a different set of generators than $G_{\mathcal{S}}$. We have that for all $X, Y \in \mathcal{S}$, there is a $g_T \in G_{\mathcal{S}}$ such that $g_T(X) = t_{XY}$ and a $g'_T \in G'_{\mathcal{S}}$ such that $g'_T(X) = t_{XY}$. Taking into account that both sets have the same number of generators, we conclude that $G_{\mathcal{S}}$ and $G'_{\mathcal{S}}$ are the same set.

Morphisms in cognitive categories represent intuitively transformations of states, however these transformations are too specific to be useful in real problems. Normally a cognitive system will have a set

of operations to work with an independent system. The above developments show how we can build operations that are specialized in pursuing a single goal from a set of operations with unspecific results. More importantly, we can assign one of these operations to each state such that instead of specifying a goal as a state in \mathcal{S} , we can specify a goal as an operation defined by a purposeful generator and extend this result to the resolution of cognitive problems. Thus, any omnipotent cognition over \mathcal{S} straightforwardly solves any arbitrary cognitive problem $B = (\mathcal{S}, O, T)$ by calling the generator $g_T \in G_{\mathcal{S}}$. Hence, at first sight B could alternatively be specified by the triple (\mathcal{S}, O, g_T) .

Example: Following the example above, the control element of a Turing Machine M is a generator over the infinite string of symbols: For each state $O_i \in \mathcal{S}_{TM}$, M produces another string $g_{TM}(O_i) = T_i$. Turing machines that overwrite the contents of the tape with a constant string are pseudo-purposeful generators: $g_{TM}(O_i) = T$ for all $O_i \in \mathcal{S}_{TM}$, as long as O_i and T have a finite number of distinct elements. Therefore, it is not possible to construct an omnipotent cognition with Turing machines unless Q is infinite, but by definition Q is finite. \square

We have introduced the first steps towards formally grounding abstract cognitive processing by showing how elementary transformations between states are related to operations performed by cognitive systems on independent systems. Generators generalize operations in the context of category theory and provide a more natural way of approaching transformations of independent systems than morphisms, which are limited to one specific outset and goal each. On the contrary, the domain of generators include every state in a cognitive category. Moreover, we have seen a class of generators related to a single state each that take the target system \mathcal{S} to that state regardless of the outset and allows to work with operations instead of states in cognitive problems.

3.4 Evaluators

Until now we have assumed that the states of \mathcal{S} are fully known. In general, this is not the case, however desirable. Evaluators cover those situations where

$$\begin{array}{ccccc}
S_{A1} & \xleftarrow{g(S_{A2})} & S_{A2} & \xrightarrow{ES_{A1}=ES_{A2}} & V_A & \curvearrowright_{g_{\mathcal{V}}(V_A)} \\
g(S_{A1}) \downarrow & & & & \downarrow_{g_{\mathcal{V}}(V_A)} & \\
S_{B1} & \xleftarrow{g(S_{B2})} & S_{B2} & \xrightarrow{ES_{B1}=ES_{B2}} & V_B & \curvearrowright_{g_{\mathcal{V}}(V_B)}
\end{array}$$

Figure 4: Objects S_{A1} , S_{A2} , S_{B1} and S_{B2} belong to the cognitive category \mathcal{S} , and $V_A, V_B \in \mathcal{V}$. Only the morphisms generated by g and its counterpart $g_{\mathcal{V}}$ are shown. Evaluator $E : \mathcal{S} \rightarrow \mathcal{V}$ sends objects S_{A1} and S_{A2} to V_A , and S_{B1} and S_{B2} to V_B . Generator g cannot be transferred to \mathcal{V} because its $g_{\mathcal{V}}$ is not a function.

information about the state is needed, yet there is partial or no access to this state. In a sense, evaluators *query* the independent system and return some evaluation on the state, without actually referencing any particular state. The power of evaluators stem from the fact that they can group many states with a common property into a single object of another category, resulting in an abstraction of unknown states into a cognitive category with known states.

Consider a cognitive category \mathcal{V} whose objects represent abstract properties. The objects in \mathcal{V} group states according to some pattern, characteristic or common property. Also, consider a functor $E : \mathcal{S} \Rightarrow \mathcal{V}$ that sends each state in \mathcal{S} to an object in \mathcal{V} . E establishes the relation between states in \mathcal{S} and abstract objects in \mathcal{V} .

Definition: An evaluator is a functor $E : \mathcal{S} \Rightarrow \mathcal{V}$, with \mathcal{S}, \mathcal{V} being cognitive categories. \square

\mathcal{V} is a partition of \mathcal{S} , therefore the number of evaluators is precisely the Bell number B_n [18] with index $n = |\text{Obj}(\mathcal{S})|$:

$$B_n = \frac{1}{e} \sum_{k=0}^{\infty} \frac{k^n}{k!}$$

Example: Neural networks for classification [24] take input vectors and output an element in a finite set. It is possible to construct an evaluator from these neural networks. If \mathcal{S} is a cognitive category whose states \mathbf{s}_i are described with vectors and \mathcal{V} is another cognitive category whose objects $V \in \mathcal{V}$ are the out-

put elements of a neural network $N : \mathbf{s}_i \rightarrow \text{Obj}(\mathcal{V})$, then an evaluator $E : \mathcal{S} \rightarrow \mathcal{V}$ is constructed in the following way: taking $\mathbf{s} \in \mathcal{S}$, the object component of E is the neural network: $E(\mathbf{s}) = N(\mathbf{s})$ and the morphism component is trivial: $E(t_{\mathbf{rs}}) = t_{E(\mathbf{r})E(\mathbf{s})}$. \square

Example: Consider a cognitive problem where the desired goal is a positive assessment from all binary evaluators $E_1 \cap \dots \cap E_n$, i.e. with two outcomes *pass* and *fail*. In this case, the appropriate evaluator to use as goal for the cognitive problem is $E(S) = E_1(S) \wedge \dots \wedge E_n(S)$. \square

3.4.1 Generators in evaluators.

Evaluators are only useful if we can apply transformations to their outcomes. Let us transfer the generators over \mathcal{S} to generators over \mathcal{V} . \mathcal{V} is an abstract category, so it does not make sense to talk about operations in \mathcal{V} unless they are grounded somehow. For that reason, the operations conveyed by generators g'_i over \mathcal{V} are the same operations as generators g_i over \mathcal{S} , hence gt' (generate and transform) actually transforms the outset in \mathcal{S} . If E is an evaluator, we say that $g' = Eg$ and take that g' is the set of codomains of E restricted to the morphisms generable by g .

Definition: Given a set of generators $G_{\mathcal{S}}$ over \mathcal{S} , an evaluator $E : \mathcal{S} \Rightarrow \mathcal{V}$ is *controllable* by $G_{\mathcal{S}}$ if $G'_{\mathcal{S}} = EG_{\mathcal{S}} = \{g'_i = Eg_i | g_i \in G_{\mathcal{S}}\}$ forms an omnipotent set of generators over \mathcal{V} . \square

Cognitive problems expressed with controllable evaluators allow for complete control over a system while allowing uncertainty in the definition of the outset and goal.

Example: John loves Mary, but he does not know if she loves him back. John evaluates all possible states in the universe \mathcal{U} in 2 outcomes: Those where Mary loves John \heartsuit and those that do not \heartsuit . His love is so deep that he does not even consider an universe where either himself or Mary do not exist. His cognitive problem is $(\mathcal{U}, \heartsuit \cup \heartsuit = \text{Obj}(\mathcal{U}), \heartsuit)$. John decides to take action g =“confess his love in public” to make her fall in love with him, unaware that g is not a purposeful generator because Mary will feel embarrassed if she loved him. John loses the love of her life.

$$\heartsuit \begin{array}{c} \xrightarrow{\text{confess}(\heartsuit)} \\ \xleftarrow{\text{confess}(\heartsuit)} \end{array} \heartsuit$$

This example shows that it is possible to model abstract concepts as cognitive categories with the use of generators and evaluators, without requiring extensive knowledge of the underlying independent system. \square

3.4.2 Hidden states.

By definition, a generator assigns one morphism (or state) to each state. However, due to uncertainties, some operations do not always led to the same result, even if that operation is repeated from the same outset. Therefore, the operation does not yield a generator because it is not a function. The solution is to consider hidden states (see Figure 5).

Consider an outset $O \in \mathcal{S}$ and a putative generator $g \in \mathcal{G}_{\mathcal{S}}$ that such that

$$g(O) = \begin{cases} T_1 & P(g(O) = T_1|O) = p_1 \\ T_2 & P(g(O) = T_2|O) = p_2 = 1 - p_1 \end{cases}$$

where p_1 and p_2 are the probabilities that the outcome of $g(O)$ are T_1 and T_2 , respectively. g is not a function so it is not a generator. This is the trick: We split the outset O into O'_1 and O'_2 in another cognitive category \mathcal{S}' such that the new category has the same states as \mathcal{S} plus an additional one that represents the second outcome from $g(O)$, and construct a proper generator g' such that if $X \neq O$ and $g(X) = Y$, then $g'(X') = Y'$, and if $X = O$, then $g'(O'_1) = T'_1$ and $g'(O'_2) = T'_2$. That is, O is equivalent to O_1 when $g(O) = T_1$ and analogously to O_2 . We now have grounded the generator in a more accurate cognitive category and removed the uncertainty posed by $g(O)$. By application of an evaluator $E : \mathcal{H} \implies \mathcal{S}$

$$E(X') = \begin{cases} O & \text{if } X' = O'_1 \text{ or } X' = O'_2 \\ X & \text{Otherwise} \end{cases}$$

we can verify that Eg' is not a generator in \mathcal{S} . O'_1 and O'_2 are hidden states in O .

The bottom line is that an operation over a cognitive category \mathcal{S} that yields two different outcomes

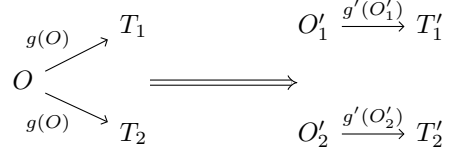


Figure 5: Outset O presents one hidden state because $g(O)$ is not a function. O is splitted into O'_1 and O'_2 and g adapted to function g' .

from the same outset is an indication that there is some deeper structure that we are not aware of, yet it is still possible to be modeled as a cognitive category by designing a more detailed model that takes the different outcomes as a substate of the outset. Hidden states are reciprocal to evaluators: Evaluators reduce the number of states in a category whilst hidden states increments them.

We are now ready to define cognitive problems in a general form.

Definition: Given a cognitive category \mathcal{S} , a *cognitive problem* B is a triple $(\mathcal{S}, \mathbf{O}, \mathbf{T})$, with $\mathbf{O}, \mathbf{T} \subset \mathcal{S}$ subsets denoting outset and goal, respectively. \square

This definition has the advantage of expressing a cognitive problem with partial information about states of \mathcal{S} . An equivalent formulation is to define the outsets and goals implicitly as objects $V_O, V_T \in \mathcal{V}$ in the codomain of an evaluator $E : \mathcal{S} \longrightarrow \mathcal{V}$. The solution to B is a generator that sends every object $O_i \in \mathbf{O}$ to any object $T_j \in \mathbf{T}$. This way, the actual outset, whose uncertainty is represented by the set \mathbf{O} , is guaranteed to be sent to any state T_j that complies with the desired goal.

Example: In this example we will model genetic algorithms with the proposed framework. Genetic algorithms (see for example [23] or [3]) are iterative processes that are composed of (1) a population of n *chromosomes* \mathbf{v}_i generally represented with bit vectors, (2) genetic operators that alter the chromosomes, and (3) a fitness function f . The adaptation is as follows: (1) The population of chromosomes \mathbf{v}^n maps to one object in the cognitive category \mathcal{P} of populations, (2) genetic operators map to gener-

ators over \mathcal{P} , and (3) fitness functions map to evaluator $E : \mathcal{P} \rightarrow \mathcal{V}$, where $Obj(\mathcal{V}) = \{V_{opt}, V_{not}\}$ representing, respectively, better optimization than the outset population and not better. However, mutation and crossover operators yield random morphisms in \mathcal{P} , which indicate the presence of hidden states that encode the future outcomes of the mutation operators, i.e. the seed in pseudo-random generators. To overcome this uncertainty, we pose the cognitive problem by specifying outset and goal in \mathcal{V} : $B_{opt} = (\mathcal{P}, V_{not}, V_{opt})$. Now, mutation and crossover operations have a high probability of being proper generators in \mathcal{V} . Note that we are not constructing proper generators from pseudo-generators, but rather bypassing the randomness presented in \mathcal{P} by implicitly considering the cognitive category of hidden states $\mathcal{H}_{\mathcal{P}}$, an evaluator $F : \mathcal{H} \rightarrow \mathcal{P}$ and the composition $E \circ F : \mathcal{H} \rightarrow \mathcal{V}$, which allows to send generators from \mathcal{H} to \mathcal{V} if the parameters of the genetic algorithm are chosen adequately to converge.

$$\mathcal{H} \begin{array}{c} \xrightarrow{F} \mathcal{P} \xrightarrow{E} \mathcal{V} \\ \xrightarrow{E \circ F} \end{array}$$

□

3.5 Dynamic systems

We have previously stated that independent systems keep no relation to other systems whatsoever. This also applies to time. In order to conserve generality, we give time no special consideration and incorporate it into states, which yields the expected independency. If \mathcal{S}_D is a dynamic independent system, let us describe a state $S \in \mathcal{S}_D$ as a vector of static substates $S = \{S_{t_0}, S_{t_1}, \dots\}$. Thus, S holds the complete timeline of \mathcal{S}_D in a single state. The set of objects in the cognitive category \mathcal{S}_D holds all the possible timelines. We deal with uncertainty in time by considering deterministic (causal determinism) and indeterministic dynamic systems.

3.5.1 Time in deterministic systems.

Consider a dynamic independent system whose state is defined by $S = \{S_{t_0}, S_{t_1}, \dots\}$. A deterministic independent system is completely determined by the

state in one instant t_0 . The remaining instants are calculated inductively from $S_{t+1} = f(S_t)$. Hence, we can rewrite the state of a deterministic system with just $S = S_0$ and use it for the objects in the cognitive category.

3.5.2 Time in indeterministic systems.

In this case, the states $S \in \mathcal{S}$ must include the temporal evolution of the system in all instants to completely and uniquely describe each possible timeline, i.e. two timelines that split at instant t_1 are described by states $S_A = \{S_{t_0}, S_{t_1}^A, \dots\}$ and $S_B = \{S_{t_0}, S_{t_1}^B, \dots\}$. If a cognition has access to evaluating \mathcal{S} only at instant t_0 , then we can construct an auxiliary cognitive category \mathcal{V} whose objects are all the possible states of \mathcal{S} only at instant t_0 : $\mathcal{V} = \{V_i = S_{t_0}^i\}$ and an evaluator $E : \mathcal{S} \rightarrow \mathcal{V}$ such that $E(S_i) = S_{t_0}^i$. The result is that we have a cognitive category, i.e. \mathcal{V} , of an indeterministic dynamical system with uncertainty in future instants, but subject to cognitive processing with all the formal tools of the previous sections.

3.6 Agents

The agent-environment paradigm is prevalent in artificial intelligence. In this paper we drop the assumption that agents and environments should be treated as separated components with the definition of independent systems. This proposal integrates the agent-environment paradigm as a special case of independent systems. To start with, consider an independent system \mathcal{S} consisting of two subsystems Agent \mathcal{S}_A and Environment \mathcal{S}_E . Then all states $S \in \mathcal{S}$ have the form $S = A \times E$, where $A \in \mathcal{S}_A$ and $B \in \mathcal{S}_B$ are substates that describe the agent and the environment, respectively. A putative cognitive system acts on \mathcal{S} as a whole, rather than controlling agent and/or environment separately.

Example: A neural network that controls an automatic car is described by a set of weights $\bar{w} = \{w_{i,j}\}$. The timeline of the neural network, car and roadways constitute an independent system $\mathcal{S} = A \times E$ where $A = \bar{w} \times A_C \times A_M$ describes the neural net-

work \bar{w} , the computing system A_C that executes the neural network, and the mechanical components that drive the car and sense the environment A_M , whilst E describes the roadway. A drives safely along E with no need for a cognitive system. Even so, we would like to improve the reliability of the system against unexpected situations, so we take a deep learning algorithm to train the neural network for better performance. This algorithm executes independently to driving the car and transforms \mathcal{S} by modifying the values in \bar{w} , therefore the deep learning algorithm is a generator over the category of neural networks that drive A , and it solves the cognitive problem $B = (\mathcal{S}, \mathbf{O}, \mathbf{T})$, where $\mathbf{O} = \bar{w} \times \mathbf{A}_C \times \mathbf{A}_M \times \mathbf{E} \subset \mathcal{S}$ is all the possible states that describe the independent system with a fixed \bar{w} and $\mathbf{T} \subset \mathcal{S}$ is the set of all states where the neural network performs better than any $O \subset \mathbf{O}$. Moreover, consider the evaluator $F : \bar{w} \times A_C \times A_M \times E \rightarrow \bar{w}$, or equally, $F : \mathcal{S} \rightarrow \mathcal{V}_{\bar{w}}$. Any generator over \mathcal{S} will only be an omnipotent generator over at most $\mathcal{V}_{\bar{w}}$ because there are no generators that transform A_C , A_M or E independently to \bar{w} . \square

4 Discussion

We have succinctly introduced the principles of a new way to understand artificial intelligence. There is a long path until we can fully understand how category theory may contribute to cognitive theories. Indeed, just from applying two of the most basic concepts in category theory, i.e. categories and functors, we have shown that turing machines, neural networks and evolutionary algorithms admit a single formalization under the proposed framework. This framework considers operations as *black boxes* and constructs methods to manage them and operate with them. The flexibility of categories allows to consider many different kinds of systems which are not limited to the target system, but can also represent abstract systems grounded on the target system that ease the theoretical study of the operations.

It remains to study how other central concepts in category theory can contribute to extend this framework further, for example by analyzing the role of

the Yoneda Lemma in deepening our understanding of canonical sets of generators (section 3.3). Moreover, one of the central cognitive abilities that has been left out of this article is analogies. It seems plausible to generalize evaluators and hidden states to analogies using adjunctions instead of functors, which would enable to construct a network of adjunctions between cognitive categories of representations. Furthermore, we postulate that cognitive categories of generators and of evaluators enable a cognitive system to cognitively process and improve its own methods, opening the path to formally describing meta-cognition by fusing the cognitive system with the independent system it controls. We leave these topics for future developments, as well as more detailed cognitive-categorical models of the examples presented.

We have challenged some prior assumptions traditionally rooted in AI. Firstly, we have dropped the distinction between agent and environment in order to process them as a whole. The advantage is that any behaviour that emerges from interactions agent-environment requires no special treatment. Secondly, we considered time as an additional state variable by including the timeline of the system as states. For this reason, we claim that the study of general AI demands that we approach it from several perspectives, challenging the assumptions that may hinder the progress in the field, to unblock the switch from narrow to general AI.

References

- [1] Francisco J Arjonilla. *A three-component cognitive theory*. Msc. thesis, Universiteit Utrecht, 2015.
- [2] Keith O. Geddes, Stephen R. Czapora, and George Labahn. *Algorithms for Computer Algebra*. Springer, 1992.
- [3] David E. Goldberg. *Genetic Algorithms*. Pearson Education India, 2006.

- [4] Jaime Gómez-Ramírez. *A New Foundation for Representation in Cognitive and Brain Science*. Springer Netherlands, Dordrecht, 2014.
- [5] Graeme S. Halford, William H. Wilson, and Steven Phillips. Relational knowledge: the foundation of higher cognition. *Trends in Cognitive Sciences*, 14(11):497–505, nov 2010.
- [6] M.J. Healy. Category theory applied to neural modeling and graphical representations. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, pages 35–40 vol.3, 2000.
- [7] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, 1979.
- [8] Marcus Hutter. A Theory of Universal Artificial Intelligence based on Algorithmic Complexity. 2000.
- [9] Marcus Hutter. *Universal Artificial Intelligence*, volume 1 of *Texts in Theoretical Computer Science An EATCS Series*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [10] John E. Laird. *The Soar Cognitive Architecture*. 2012.
- [11] John E Laird, Allen Newell, and Paul S Rosenbloom. SOAR: An integrative architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.
- [12] John E. Laird and Robert E. Wray. Cognitive architecture requirements for achieving AGI. In *Proc. of the Third Conference on Artificial General Intelligence*, pages 79–84, 2010.
- [13] Shane Legg and Marcus Hutter. Universal Intelligence: A Definition of Machine Intelligence. *Minds and Machines*, 17(4):391–444, dec 2007.
- [14] A. Newell, J. C. Shaw, and H. A. Simon. Report on a general problem-solving program. In *IFIP Congress*, volume 224, 1959.
- [15] Allen Newell. Unified theories of cognition and the role of Soar. In *Soar A cognitive architecture in perspective A tribute to Allen Newell Studies in cognitive systems Vol 10*, pages 25–79 ST – Unified theories of cognition and the. 1992.
- [16] Steven Phillips and William H. Wilson. Categorical compositionality: A category theory explanation for the systematicity of human cognition. *PLoS Computational Biology*, 6(7):7, 2010.
- [17] Marek Rosa, Jan Feyereisl, and The GoodAI Collective. A Framework for Searching for General Artificial Intelligence. 2016.
- [18] Gian-Carlo Rota. The Number of Partitions of a Set. *The American Mathematical Monthly*, 71(5):498–504, 1964.
- [19] Giulio Tononi. An information integration theory of consciousness. *BMC neuroscience*, 5:42, nov 2004.
- [20] Naotsugu Tsuchiya, Shigeru Taguchi, and Hayato Saigo. Using category theory to assess the relationship between consciousness and integrated information theory. *Neuroscience Research*, 107:1–7, 2016.
- [21] Alan Turing. On computable numbers. *Proceedings of the London Mathematical Society*, 25(2):181–203, 1936.
- [22] Joel Veness, Kee Siong Ng, Marcus Hutter, William Uther, and David Silver. A Monte-Carlo AIXI approximation. *Journal of Artificial Intelligence Research*, 40:95–142, 2011.
- [23] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2), jun 1994.
- [24] G.P. Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.