Cognitive architecture and second-order systematicity: categorical compositionality and a (co)recursion model of systematic learning

Steven Phillips (steve@ni.aist.go.jp)
Mathematical Neuroinformatics Group, National Institute of Advanced Industrial Science and Technology (AIST),
Tsukuba, Ibaraki 305-8568 JAPAN

William H. Wilson (billw@cse.unsw.edu.au)
School of Computer Science and Engineering, The University of New South Wales,
Sydney, New South Wales, 2052 AUSTRALIA

Abstract

Systematicity commonly means that having certain cognitive capacities entails having certain other cognitive capacities. Learning is a cognitive capacity central to cognitive science, but systematic learning of cognitive capacities—second-order systematicity—has received little investigation. We proposed associative learning as an instance of second-order systematicity that poses a paradox for classical theory, because this form of systematicity involves the kinds of associative constructions that were explicitly rejected by the classical explanation. In fact, both first and second-order forms of systematicity can be derived from the formal, category-theoretic concept of universal morphisms to address this problem. In this paper, we derived a model of systematic associative learning based on (co)recursion, which is another kind of universal construction. This result increases the extent to which category theory provides a foundation for cognitive architecture.

Keywords: cognitive architecture; systematicity; compositionality; learning; category theory; coalgebra

Introduction

The problem of systematicity for theories of cognitive science is to explain why certain cognitive capacities typically co-exist (Fodor & Pylyshyn, 1988); why, for example, having the ability to identify square as the top object in a scene consisting of a square above a triangle implies having the ability to identify triangle as the top object in a scene consisting of a triangle above a square. More formally, an instance of systematicity occurs when one has cognitive capacity c_1 if and only if c_2 (McLaughlin, 2009): thus, systematicity is the partitioning of cognitive capacities into equivalence classes. The problem is to provide an explanation for systematicity that does not rely on *ad hoc* assumptions: i.e., auxiliary assumptions that are motivated *only* to fit the data, cannot be verified *independently* of verifying the theory, and are *unconnected* to the theory's core principles (Aizawa, 2003).

Learning is another cognitive capacity. Hence, using the characterization of systematicity as equivalence classes of cognitive capacities (McLaughlin, 2009), we have another form of systematicity: i.e., the capacity to *learn* cognitive capacity c_1 if and only if the capacity to *learn* cognitive capacity c_2, which is sometimes referred to as *second-order systematicity* (Aizawa, 2003). Aizawa (2003), citing Chomsky (1980), provides an example from language: a person has the capacity to learn one natural language (say, Chinese) if they have the capacity to learn another (say, German). Importantly, the learned capacities need not be systematically related to each other. An example that is pertinent to the classical explanation is the learning (or memorization) of arbitrary associations. For instance, if one has the capacity to learn (memorize) that the *first day of the Japanese financial year is April 1st*, then one also has the capacity to learn (memorize) that the *atomic number of carbon is 6*. This example is a legitimate instance of systematicity (at the second level) given that systematicity has been characterized as equivalence classes of cognitive capacities (McLaughlin, 2009).

Elsewhere (Phillips & Wilson, submitted), we argue that associative learning is problematic for classical theory, because it involves the kinds of associative constructions that were explicitly rejected by the classical explanation. Our category theoretic explanation of systematicity resolves this problem, because both first and second-order forms of systematicity are derived from the same categorical construction: universal morphisms. Here, we derive a model of systematic associative learning based on (co)recursion, which is another kind of universal construction.

Outline of paper

The remaining four sections of this paper are outlined as follows. The second section provides the basic category theory definitions and motivating examples needed for our model. The third section provides concrete examples of anamorphisms (corecursion) as a conceptual guide to the model given in the fourth section. Discussion of this work is in the last section. Readers already familiar with the category theory approach to corecursion may prefer to skip straight to the fourth section, since the category theory employed here is already well known, with the possible exception of the recently developed adjoint extensions (Hinze, 2013). Readers not familiar with a category theory approach may prefer to skip to the third section for simple concrete examples as a way of orienting themselves for the model that follows. The second section, then, provides points of reference for technical details and motivating examples when needed.

Basic category theory

In this section, we provide the basic category theory needed for our model. In the interests of brevity and clarity, we only provide definitions and examples directly pertaining to the model, omitting the (albeit, well known) theorems and lemmas that justify statements. Deeper and broader introductions

to category theory and categorical treatments of (co)recursion can be found in many textbooks on the topic (e.g., Mac Lane, 1998; Bird & de Moor, 1997). In the context of systematicity, this paper builds upon our earlier work (see Phillips & Wilson, 2010), and particularly in the context of recursive capacities (Phillips & Wilson, 2012), where the theoretical principles of *first*-order systematicity and other technical details can be found.

Definition 1 (Category)**.** A *category* **C** consists of:

- a collection of *objects* (A, B, ...);
- a collection of *morphisms* (f, g, ...), written $f : A \to B$ to indicate that A and B are respectively the *domain* and *codomain* of f, including an identity morphism, denoted $1_A : A \to A$, for each object A in **C**; and
- a *composition* operation that sends a pair of *compatible* morphisms $f : A \to B$ and $g : B \to C$, i.e., where the codomain of f equals the domain of g, to their *composite* morphism, denoted $g \circ f : A \to C$,

that satisfy the axioms of: *associativity*—$h \circ (g \circ f) = (h \circ g) \circ f$; and *identity*—$f \circ 1_A = f = 1_B \circ f$ for each f in **C**.

Example 1 (**Set**)**.** The category **Set** has sets for objects, functions between sets for morphisms, and composition is composition of functions.

Definition 2 (Terminal object)**.** In a category **C**, a *terminal object* is an object, denoted 1, such that for every object Z there exists a unique morphism $u : Z \to 1$.

Remark. In **Set**, a singleton set is a terminal object, whose only element is denoted $*$ when its identity is not required. Other categories may have terminal objects with further internal structure, as we shall see for categories of (co)algebras.

Definition 3 (Isomorphism)**.** An *isomorphism* is a morphism $f : A \to B$ such that their exists a morphism $g : B \to A$ satisfying $f \circ g = 1_B$ and $g \circ f = 1_A$. Morphism g is called the *inverse* of f, and denoted f^{-1}.

Remark. In a category **C**, the collection of morphisms with domain object A and codomain object B is called a *hom-set*, denoted $\mathrm{Hom}_{\mathbf{C}}(A,B)$. As we shall see, hom-sets play an important role in category theory.

Definition 4 (Functor)**.** A *functor* F from a category **C** to a category **D** is a *structure-preserving map*, written $F : \mathbf{C} \to \mathbf{D}$, that maps each object A in **C** to the object $F(A)$ in **D** and each morphism $f : A \to B$ in **C** to the arrow $F(f) : F(A) \to F(B)$ in **D** such that the following axioms are satisfied:

- *identity*: $F(1_A) = 1_{F(A)}$ for each object A in **C**; and
- *compositionality*: $F(g \circ_{\mathbf{C}} f) = F(g) \circ_{\mathbf{D}} F(f)$ for each pair of compatible morphism (f,g).

Remark. An *endofunctor* is a functor $F : \mathbf{C} \to \mathbf{C}$, i.e., the domain and codomain are the same category **C**. Endofunctors are used to model (co)recursion.

Example 2 (Right product functor)**.** The *right product functor* $\Pi_B : \mathbf{Set} \to \mathbf{Set}$ sends each set X to the Cartesian product $X \times B$ and each function $f : X \to Y$ to the product of functions $f \times 1_B : X \times B \to Y \times B, (x,b) \mapsto (f(x),b)$, i.e., $f \times 1_B$ maps (x,b) to $(f(x),b)$.

Example 3 (Right exponential functor)**.** The *right exponential functor* $\Lambda_B : \mathbf{Set} \to \mathbf{Set}$ sends each set X to the function set X^B, which is the set of functions $\{f : B \to X\}$, and each function $g : X \to Y$ to the function $\Lambda(g) : X^B \to Y^B, f \mapsto g \circ f$.

Example 4 (List)**.** List-related constructions built from a set of elements A are obtained from an endofunctor on the category **Set**: i.e., $F_A : X \mapsto 1 + A \times X, f \mapsto 1_1 + 1_A \times f$, where 1 corresponds to the empty list, and $+$ and $\times$ are (respectively) the disjoint union and Cartesian product of sets or functions.

Definition 5 (Natural transformation)**.** A *natural transformation* η from a functor $F : \mathbf{C} \to \mathbf{D}$ to a functor $G : \mathbf{C} \to \mathbf{D}$, written $\eta : F \dot{\to} G$, is a family of **D**-morphisms $\{\eta_A : F(A) \to G(A) | A \text{ is an object in } \mathbf{C}\}$ such that for each morphism $f : A \to B$ in **C** we have $G(f) \circ \eta_A = \eta_B \circ F(f)$, i.e., the following diagram is *commutative* (equational):

$$\begin{array}{ccc} F(A) & \xrightarrow{\eta_A} & G(A) \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ F(B) & \xrightarrow[\eta_B]{} & G(B) \end{array} \tag{1}$$

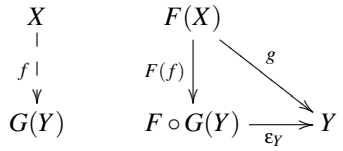
Definition 6 (Final morphism)**.** A *final morphism* from a functor $F : \mathbf{C} \to \mathbf{D}$ to an object X in **D** is a pair (A, ϕ) consisting of an object A in **C** and an morphism $\phi : F(A) \to X$ in **D** such that for every object Z in **Z** and every morphism $f : F(Z) \to X$ in **C** there exists a unique morphism $u : Z \to A$ such that $f = \phi \circ F(u)$, as indicated by the following commutative diagram (dashed arrows indicate unique existence):

$$\begin{array}{ccc} Z & F(Z) & \\ {\scriptstyle u}\,\vdots & {\scriptstyle F(u)}\,\vdots & \searrow^{f} \\ A & F(A) & \xrightarrow[\phi]{} X \end{array} \tag{2}$$

Remark. The *dual* of final morphism is *initial morphism*, whose definition is obtained by reversing the directions of the morphisms in the definition of final morphism. A universal morphism is either a final morphism or an initial morphism. In general, category theory concepts are dualized by reversing all the arrows in the definition of the original concept.

Definition 7 (Adjunction)**.** An *adjunction* from a category **C** to a category **D** is a triple, written $(F,G,\varepsilon) : \mathbf{C} \rightharpoonup \mathbf{D}$, consisting of a functor $F : \mathbf{C} \to \mathbf{D}$, a functor $G : \mathbf{D} \to \mathbf{C}$ and a natural transformation $\varepsilon : F \circ G \dot{\to} 1_{\mathbf{D}}$ such that for each object Y in **D**, the pair $(G(Y), \varepsilon_Y)$ is a final morphism from F to Y, as

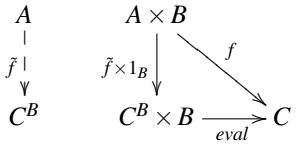
indicated by the following commutative diagram:



(3)

Remark. The functor F is called the *left adjoint* of functor G, and G is called the *right adjoint* of F. The relationship between F and G is called an *adjoint situation*, denoted $F \dashv G$. The morphism ε_Y is the component of the natural transformation ε at object Y. Definition 7 emphasizes the natural transformation and universal morphism aspects of adjunctions, cf. diagrams 3 and 2.

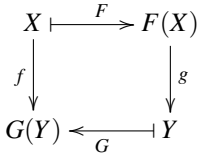
Example 5 (Product-exponential)**.** The right product functor is left adjoint to the right exponential functor, $\Pi_B \dashv \Lambda_B$, see examples 2 and 3, is indicated by commutative diagram



(4)

where $\tilde{f}$ is called the *exponential transpose* of f, and *eval* is the evaluation of each function $\tilde{f}$ in C^B at each b in B.

Remark. An equivalent definition of adjunction emphasizes the relationship between hom-sets: an adjunction is a bijection (i.e., one-to-one correspondence) between hom-sets $\mathrm{Hom}_{\mathbf{C}}(X, G(Y))$ and $\mathrm{Hom}_{\mathbf{D}}(F(X), Y)$ that is natural (in the natural transformation sense) in variables X and Y, written $\mathrm{Hom}_{\mathbf{D}}(F(X), Y) \cong \mathrm{Hom}_{\mathbf{C}}(X, G(Y))$, as indicated by diagram



(5)

Hence, one can think of an adjunction as a kind of isomorphism that is local to hom-sets, but not necessarily global to categories. This aspect will be useful when considering adjunctions in the context of corecursion.

Example 6 (Curry-uncurry)**.** The product-exponential adjoint is familiar in functional programming in the form of the curry-uncurry operator, which converts an n-ary function (i.e., a function of n arguments) to a unary function (i.e., a function of one argument). For instance, the curry of addition, written as the binary function $add : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, is the unary function $addN : \mathbb{N} \to \mathbb{N}^{\mathbb{N}}$, which takes a number x and returns the *addx* function: e.g., $addN : 1 \mapsto add1$, where $add1 : n \mapsto n+1$. Uncurry is the inverse of curry. These two operators are either sides of the bijection $\mathrm{Hom}_{\mathbf{Set}}(\mathbb{N} \times \mathbb{N}, \mathbb{N}) \cong \mathrm{Hom}_{\mathbf{Set}}(\mathbb{N}, \mathbb{N}^{\mathbb{N}})$ obtained from the product-exponential adjoint. Similarly, a state transition function is a binary function $\tau : A \times S \to S$ from inputs, $a \in A$, and states, $s \in S$, to states, or equivalently a unary function $\tau_S : A \to S^S$ from an input to function between states, as given by the bijection $\mathrm{Hom}_{\mathbf{Set}}(A \times S, S) \cong \mathrm{Hom}_{\mathbf{Set}}(A, S^S)$. We make use of this universal construction in our associative learning model.

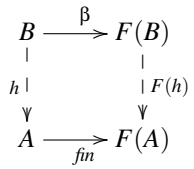
Definition 8 (F-coalgebra)**.** An F-coalgebra on an endofunctor $F : \mathbf{C} \to \mathbf{C}$ is a pair (A, α) consisting of an object A and an morphism $\alpha : A \to F(A)$ in $\mathbf{C}$.

Definition 9 (F-coalgebra homomorphism)**.** An *F-coalgebra homomorphism* from a coalgebra (B, β) to a coalgebra (A, α) is a morphism $h : (B, \beta) \to (A, \alpha)$ such that $F(h) \circ \beta = \alpha \circ h$.

Definition 10 (Category of F-coalgebras)**.** Suppose an endofunctor $F : \mathbf{C} \to \mathbf{C}$. A *category of F-coalgebras*, denoted $\mathbf{CoAlg}(F)$, has F-coalgebras for objects and F-coalgebra homomorphisms for morphisms. Composition is composition of F-coalgebra homomorphisms.

Definition 11 (Final F-coalgebra)**.** Suppose we have a category of F-coalgebras, $\mathbf{CoAlg}(F)$. A *final F-coalgebra* is an F-coalgebra, denoted (A, fin), such that for every F-coalgebra (B, β) in $\mathbf{CoAlg}(F)$ there exists a unique F-coalgebra homomorphism $h : (B, \beta) \to (A, fin)$.

Definition 12 (Anamorphism)**.** An *anamorphism* is an F-coalgebra homomorphism to a final F-coalgebra homomorphism, as indicated by the following commutative diagram:



(6)

Remark. Anamorphism h is denoted $[\![\beta]\!]$, using *lens brackets* (Meijer, Fokkinga, & Paterson, 1991), since h is completely determined by β. Anamorphism is also called *unfold*.

Definition 13 (Conditional function)**.** A *conditional function* is a function consisting of a predicate $p? : A \to \{\mathrm{False}, \mathrm{True}\}$ and two alternative functions $f : A \to B$ and $g : A \to C$, written $(p? \to f, g) : A \to B + C$, that is defined as:

$$(p? \to f, g) : a \mapsto \begin{cases} f(a) & \neg p?(a); \\ g(a) & \text{otherwise.} \end{cases}$$

That is, a function that applies alternative f to argument a if $p?(a)$ is false, otherwise alternative g. Recall that $B + C$ is the disjoint union of sets B and C.

Example 7 (List anamorphism)**.** For list-related constructions built from elements in a set A, we have a category of coalgebras on the endofunctor $F_A : X \to 1 + A \times X$. It can be shown that a final coalgebra for this category consists of conditional function $(empty? \to \mathsf{I}_*, \langle head, tail \rangle) : L \to 1 + A \times L$, where L is the set of lists constructed from elements of a set A, predicate *empty*? tests for empty list, constant function $\mathsf{I}_* : L \to 1$ returns unnamed element $*$, and product function $\langle head, tail \rangle : L \to A \times L$ returns a pair consisting of the head

and the tail of the given list. Every anamorphism to this final coalgebra is given by commutative diagram

$$\begin{array}{ccc} X & \xrightarrow{(p?\to \mathsf{I}_*,\langle f,g\rangle)} & 1+A\times X \\ {\scriptstyle [\![p?\to \mathsf{I}_*,\langle f,g\rangle]\!]}\downarrow & & \downarrow{\scriptstyle 1+1_A\times[\![p?\to \mathsf{I}_*,\langle f,g\rangle]\!]} \\ L & \xrightarrow[(empty?\to \mathsf{I}_*,\langle head,tail\rangle)]{} & 1+A\times L \end{array} \tag{7}$$

Since $(empty? \to \mathsf{I}_*, \langle head, tail\rangle)$ is an isomorphism, traversing diagram 7 from X to L clockwise yields the definition:

$$[\![p? \to \mathsf{I}_*, \langle f,g\rangle]\!] : x \mapsto \begin{cases} [\,] & \neg p?(x); \\ f(x)\cdot [\![\cdots]\!](g(x)) & \text{otherwise.} \end{cases}$$

We also write $[\![p? \to \mathsf{I}_*, \langle f,g\rangle]\!]$ as $unfold(p? \to \mathsf{I}_*, \langle f,g\rangle)$.

Remark. The preceding definitions pertaining to coalgebras have duals, which are obtained by reversing morphisms. For comparison, a *catamorphism* $k : (A, in) \to (B, \beta)$ from *initial F-algebra* (A, in) to F-algebra (B, β) as indicated by

$$\begin{array}{ccc} F(A) & \xrightarrow{in} & A \\ {\scriptstyle F(k)}\downarrow & & \downarrow{\scriptstyle k} \\ F(B) & \xrightarrow[\beta]{} & B \end{array} \tag{8}$$

Catamorphism k is denoted $(\!|\beta|\!)$, by *banana brackets* (Meijer et al., 1991), since k is determined by β. Catamorphism is also called *fold*. For lists built from elements of A, catamorphisms are given by

$$\begin{array}{ccc} 1+A\times L & \xrightarrow{[empty,cons]} & L \\ {\scriptstyle 1+1_A\times(\!|\mathsf{I}_v,f|\!)}\downarrow & & \downarrow{\scriptstyle (\!|\mathsf{I}_v,f|\!)} \\ 1+A\times X & \xrightarrow[[\mathsf{I}_v,f]]{} & X \end{array} \tag{9}$$

where $empty : * \mapsto [\,]$ returns the empty list, $cons : (h,t) \mapsto h\cdot t$ returns the list with (head) element h prepended to (tail) list t, and I_v assigns the value v to the empty list. An initial algebra is an isomorphism. Thus, traversing diagram 9 from L to X counterclockwise yields the following definition:

$$(\!|\mathsf{I}_v, f|\!) : \begin{cases} [\,] & \mapsto v; \\ h\cdot t & \mapsto f(h, (\!|\mathsf{I}_v, f|\!)(t)). \end{cases}$$

We also write $(\!|\mathsf{I}_v, f|\!)$ as $fold(\mathsf{I}_v, f)$.

Example 8 (Counting)**.** Counting list elements is computed by the catamorphism, $fold(0, inc) : L \to \mathbb{N}$, where the first argument, 0, assigns the result of zero to the empty list, and the second argument, *inc*, ignores the head (first element) of the list and increments the count of the tail (remaining elements) of the list (cf. diagram 9). In other words, the count of a list is either 0, for the empty list, or one plus the count of the rest (tail) of the list. For instance,

$$\begin{aligned} & fold(0, inc)[a,b,c] \\ &= 1 + fold(0, inc)[b,c] \\ &= 1 + (1 + fold(0, inc)[c] \\ &= 1 + (1 + (1 + fold(0, inc)[\,])) \\ &= 1 + (1 + (1 + 0)) \\ &= 3. \end{aligned}$$

Remark. Every list is entirely deconstructed before folding into a result. This approach is unrealistic as a cognitive model of learning, since it requires having seen all examples before any learning can take place. Nonetheless, list catamorphisms provide an important step in that they are closely related to the dual construction that affords a model of (on-line) learning at each input presentation, which we turn to next.

Corecursion for lists

Category theory provides a systematic treatment of corecursion in the form of anamorphisms, which is the basis for our categorical model of associative learning. Several simple examples of anamorphisms provide a guide to our model.

Repeating an item n number of times is realized as the anamorphism, $unfold(0? \to \mathsf{I}_*, \langle 1, dec\rangle) : \mathbb{N} \to L$, where 0? tests whether a number is zero, 1 is the constant function returning 1, and *dec* decrements a number by 1 (cf. diagram 7). For instance,

$$\begin{aligned} & unfold(0? \to \mathsf{I}_*, \langle 1, dec\rangle)(3) \\ &= 1 \cdot unfold(0? \to \mathsf{I}_*, \langle 1, dec\rangle)(2) \\ &= 1 \cdot 1 \cdot unfold(0? \to \mathsf{I}_*, \langle 1, dec\rangle)(1) \\ &= 1 \cdot 1 \cdot 1 \cdot unfold(0? \to \mathsf{I}_*, \langle 1, dec\rangle)(0) \\ &= 1 \cdot 1 \cdot 1 \cdot [\,] \\ &= [1,1,1]. \end{aligned}$$

Notice that the anamorphism just given is a state-less (or, memory-less) computation. To count list items, we need to maintain a state for the number of previously counted items. For example, $unfold(e? \to \mathsf{I}_*, \langle incl, tailr\rangle) : \mathbb{N} \times L_X \to L_{\mathbb{N}}$ takes the number of items counted so far, $n \in \mathbb{N}$, and a list $l \in L_X$ of elements from X, and returns the progressive count of list items $c \in L_{\mathbb{N}}$. In this example, the conditional $e?$ tests for an empty list (at the second component of a given pair), i.e., no more items to be counted, effectively terminating the count when the list of remaining items is empty, via I_*, or incrementing the count and removing the counted item from the list, via product function $\langle incl, tailr\rangle$. The function $incl : (n,l) \mapsto n+1$ increments the counter (left component) and ignores the list; the function $tailr : (n, h\cdot t) \mapsto (n+1, t)$ maintains the new count and removes the counted item from the list of items to be counted. Compare diagram 7: object A is now the set of natural numbers $\mathbb{N}$, and X is the Cartesian

product $\mathbb{N} \times L_X$ of the natural numbers with the set of lists of elements from a set X. For instance,

$$\begin{aligned}
&\mathit{unfold}(e? \to \mathsf{I}_*, \langle \mathit{incl}, \mathit{tailr} \rangle)(0, [a,b,c]) \\
&\quad = \quad 1 \cdot \mathit{unfold}(e? \to \mathsf{I}_*, \langle \mathit{incl}, \mathit{tailr} \rangle)(1, [b,c]) \\
&\quad = \quad 1 \cdot 2 \cdot \mathit{unfold}(e? \to \mathsf{I}_*, \langle \mathit{incl}, \mathit{tailr} \rangle)(2, [c]) \\
&\quad = \quad 1 \cdot 2 \cdot 3 \cdot \mathit{unfold}(e? \to \mathsf{I}_*, \langle \mathit{incl}, \mathit{tailr} \rangle)(3, [\,]) \\
&\quad = \quad 1 \cdot 2 \cdot 3 \cdot [\,] \\
&\quad = \quad [1,2,3].
\end{aligned}$$

Notice, further, that this count anamorphism returns a list of counts, not a single count. The elements of such output (likewise, input) lists are commonly interpreted as being indexed by steps in time for corecursive models of data streams, i.e., infinite lists (Rutten, 2000). We invoke a similar temporal interpretation of lists for our learning model.

Categorical (corecursion) model

We develop our model in two steps for expository purposes. The first step treats the association network as an explicit input. This approach is simpler, but unrealistic since memory is treated as external input. The second step treats memory as internal using adjoint anamorphisms (adjoint unfolds).

Network state as external input

The capacity for learning associations is modeled as a function from a list of pairs (associates) to an association network. Recall, from the counting example, that a simple anamorphism does not maintain a state, and so does not suffice as an associative learning model, since previous associations are lost. A memory is maintained by passing the results of earlier items as an explicit input to the model. Accordingly, associative learning is modeled as a function from a list of pairs and an association network to an updated association network. The anamorphism (model) is indicated by the diagram

$$\begin{array}{ccc}
P \times G & \xrightarrow{(e? \to \mathsf{I}_*, \langle \mu, \nu \rangle)} & 1 + G \times (P \times G) \\
{\scriptstyle [e? \to \mathsf{I}_*, \langle \mu, \nu \rangle]} \downarrow & & \downarrow {\scriptstyle 1 + 1_A \times [e? \to \mathsf{I}_*, \langle \mu, \nu \rangle]} \\
L & \xrightarrow[(p_{empty} \to \mathsf{I}_*, \langle head, tail \rangle)]{} & 1 + G \times L
\end{array} \tag{10}$$

where:

- P is a list of pairs of associated items;
- G is the set of (labeled) directed graphs (association networks), where each graph $g \in G$ is a pair (E,V) consisting of a set of edges E and a set of vertices V, and each edge is a triple (s,σ,t), where s and t are the source and target vertices and σ is the strength of association; hence
- L is the set of lists of association networks;
- $e?$ tests whether the list of associates is empty;
- $\mu : P \times G \to G$ is a function that merges the current pair of associated elements with the current association network, returning an association network; and
- $\nu : P \times G \to P \times G$ is the next state function that returns the list of remaining pairs, and the merged association network: i.e., $\nu = \langle \tau, \mu \rangle$, where $\tau : P \times G \to P$ returns the tail of the pairs list, which ignores the association network.

An example illustrates the mechanism. The anamorphism given by diagram 10 is relabeled m_{ext}, the model with external memory. Suppose the initial list of pairs: [(bread, butter), (knife, fork), (knife, butter)]. The initial state of the association network is set to the empty graph e. We denote pair and network lists at time t as p_t and g_t, respectively. Hence, the initial pair list p_0 contains three pairs, and the inital network $g_0 = e$. One time step is the mapping $m_{ext} : (p_0, g_0) \mapsto g_1 \cdot m_{ext}(p_1, g_1)$, where g_1 is the association network containing the single edge σ_1 : bread $\to$ butter (i.e., an association from bread to butter with strength of association σ_1), and p_1 is the pairs list [(knife, fork), (knife, butter)]. This process continues corecursively until we obtain $g_1 \cdot g_2 \cdot g_3 \cdot m_{ext}(p_3, g_3)$ at which point the model returns the empty list (of networks) and terminates with the list $[g_1, g_2, g_3]$. That is the evolution of association networks over time steps, with g_3 being the final network state indicated by the following diagram:

$$\begin{array}{ccc}
\text{bread} & \xrightarrow{\sigma_1} & \text{butter} \\
& \overset{\sigma_3}{\nearrow} & \\
\text{knife} & \xrightarrow{\sigma_2} & \text{fork}
\end{array} \tag{11}$$

An important feature of the anamorphism approach, in contast to a catamorphism approach, is that the computation at each (time) step proceeds independently of the remaining steps. For example, the first item of the list $g_1 \cdot m_{ext}(p_1, g_1)$, i.e., g_1, is not affected by the computation of the rest of the list. This property of anamorphisms justifies the temporal interpretation of lists. Effectively, then, there is only one association graph produced by the model, whose state is indexed by time step t: i.e., the network g_t in the list $g_0 \cdots g_t \cdot m_{ext}(p_t, g_t)$.

Network state as internal memory

The previous model depends on treating network state as a kind of external memory. The theory of adjoint catamorphisms and anamorphisms—adjoint folds and unfolds (Hinze, 2013) allows us to treat network state as internal to the model. We make use of the product-exponential adjoint introduced earlier. Recall that this construction effectively provides a universal means of transforming the external state map into an internal state map, as indicated by the following diagram (highlighting the bijection aspect of this adjunction):

$$\begin{array}{ccc}
P & \xmapsto{\Pi_G} & P \times G \\
{\scriptstyle \widetilde{[e? \to \mathsf{I}_*, \langle \mu, \nu \rangle]}} \downarrow & & \downarrow {\scriptstyle [e? \to \mathsf{I}_*, \langle \mu, \nu \rangle]} \\
L^G & \xmapsto[\Lambda_G]{} & L
\end{array} \tag{12}$$

The internal model, which we denote m_{int}, is the exponential transpose of the external model m_{ext}. That is, $m_{int} = \widetilde{m_{ext}}$.

A final algebra is a universal construction. Thus, we have shown that the (second-order) systematicity of associative learning follows from the same category theoretical principles as other (first-order) forms of systematicity.

Discussion

The advantage of a category theory approach is that it provides a principled approach to (co)recursive cognitive capacities. Symbol systems admit arbitrary recursion, but not every recursive formulation is systematic, in the sense of being well-defined over all possible inputs. Well-definedness, categorically, depends only on the well-definedness of the given F-(co)algebra, since the (unique) existence of an anamorphism (or, catamorphism) is guaranteed by the universal property. Existence and uniqueness were motivations for taking a category theory approach to recursion in the first place (see, e.g., Bird & de Moor, 1997).

The anamorphic (corecursion) explanation for second-order systematicity of learning is quite general. Moreover, extensions to other forms of learning and associated models are straightforward. For instance, in the case of supervised learning, P is a list of pairs of input-target representations, and G is a set of neural (feedforward, error backprogagation) networks. Supervised learning proceeds in the same corecursive manner as already described, except that we replace the coalgebra updating associative networks with one for updating (feedforward) neural networks. Unsupervised learning, which omits the target, is similarly straightforward.

The generality of anamorphisms may leave some people wondering whether it is too general. In particular, since many species have the capacity for simple associative learning, why then do they not also have the capacity for more advanced forms of learning, such as learning via analogy? Recall that the systematicity problem is at the level of the complex entities, not at the level of their components. For example, the capacity to understand that *John loves cricket* implies the capacity to understand *John loves baseball* given that one understands that *John* refers to a person, and that *cricket* and *baseball* refer to games. The capacity to understand that *John loves cricket* does not imply the capacity to understand *John loves hanafuda* when one does not understand the meaning of *hanafuda*—a Japanese card game. Likewise, we don't expect a capacity for learning associations to imply a capacity for learning by analogy, because association and analogy involve different kinds of underlying structures (Halford, Wilson, Andrews, & Phillips, 2014); categorically, they involve the existence of different objects. Rather, we expect that if a subject has the capacity for the underlying structures, then a capacity for learning with respect to one kind of structure implies a capacity for learning with respect to the other kind of structure, because they both involve the same form of (co)recursion.

To test such claims of systematicity empirically, we need three things: (1) a test for the base capacities, analogous to a test for the understanding of *John* and *Mary*, (2) a test for the understanding of a complex entity case, analogous to a test for the understanding of *John loves Mary*, and (3) a test of the prediction for another complex entity belonging to the same equivalence class, analogous to a test for the understanding of *Mary loves John*. Failure to exhibit capacity to other members of the equivalence class counts against the theory of systematicity that predicts that class. Hence, the category theory explanation that we have put forward is testable, and amenable to further empirical work.

Acknowledgments

We thank the reviewers for helpful comments. This work was supported by a Japanese Society for the Promotion of Science Grant-in-aid (26280051).

References

Aizawa, K. (2003). *The systematicity arguments*. New York: Kluwer Academic.

Bird, R., & de Moor, O. (1997). *Algebra of programming*. Harlow, England: Prentice Hall.

Chomsky, N. (1980). *Rules and representations*. New York, NY: Columbia University Press.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, *28*(1–2), 3–71.

Halford, G. S., Wilson, W. H., Andrews, G., & Phillips, S. (2014). *Categorizing cognition: Toward conceptual coherence in the foundations of psychology*. Cambridge, MA: MIT Press.

Hinze, R. (2013). Adjoint folds and unfolds—an extended study. *Science of Computer Programming*, *78*, 2108–2159.

Mac Lane, S. (1998). *Categories for the working mathematician* (2nd ed.). New York, NY: Springer.

McLaughlin, B. P. (2009). Systematicity redux. *Synthese*, *170*, 251–274.

Meijer, E., Fokkinga, M., & Paterson, R. (1991). Functional programming with bananas, lenses, envelopes and barbed wire. In *Proceedings on the conference on functional programming and computer architecture* (Vol. 523, pp. 125–144). Berlin, Germany: Springer-Verlag.

Phillips, S., & Wilson, W. H. (2010). Categorial compositionality: A category theory explanation for the systematicity of human cognition. *PLoS Computational Biology*, *6*(7), e1000858.

Phillips, S., & Wilson, W. H. (2012). Categorial compositionality III: F-(co)algebras and the systematicity of recursive capacities in human cognition. *PLoS ONE*, *7*(4), e35028.

Phillips, S., & Wilson, W. H. (submitted). *The second-order systematicity of associative learning: a paradox for classical compositionality and a coalgebraic resolution.*

Rutten, J. J. M. M. (2000). Universal coalgebra: a theory of systems. *Theoretical Computer Science*, *249*, 3–80.