Categorial compositionality continued (further): A category theory explanation for the systematicity of recursive cognitive capacities

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

Human cognitive capacity includes recursively definable concepts, which are prevalent in domains involving lists, numbers, and languages. Cognitive science currently lacks a satisfactory explanation for the systematic nature of recursive cognitive capacities. The category-theoretic constructs of initial F-algebra, catamorphism, and their duals, final coalgebra and anamorphism provide a formal, systematic treatment of recursion in computer science. Here, we use this formalism to explain the systematicity of recursive cognitive capacities without ad hoc assumptions (i.e., why the capacity for some recursive cognitive abilities implies the capacity for certain others, to the same explanatory standard used in our account of systematicity for non-recursive capacities). The presence/absence of an initial algebra/final coalgebra implies the presence/absence of all systematically related recursive capacities in that domain. This formulation also clarifies the theoretical relationship between recursive cognitive capacities. In particular, the link between number and language does not depend on recursion, as such, but on the underlying functor on which the group of recursive capacities is based. Thus, many species (and infants) can employ recursive processes without having a full-blown capacity for number and language.[1]

Keywords: systematicity; category theory; endofunctor, *F*-(co)algebra; (final) initial algebra; universal construction; catamorphism; anamorphism; classicism; connectionism

Introduction

Many cognitive domains include recursively definable concepts (i.e., concepts defined with reference to themselves), such as domains involving lists, or languages. In card games, for example, a deck of cards can be defined (recursively) as a top card (perhaps turned face up to reveal its value) and a (remaining) deck of cards. To include finite decks, the definition has an alternative clause specifying an empty deck; that is, a deck is either empty, or contains a top card and a (smaller) deck. Operations on recursively defined concepts may also be defined recursively. For example, removing jokers from a deck of cards can be defined (recursively) as removing the top card if it is a joker and then removing jokers from the remaining deck of cards. Given that you don't find people who can remove the jokers from a hand of seven cards without being able to remove jokers from a deck of fifty-three, recursion-related capacities are further instances (see below) of the systematic nature of human cognition.

Systematicity is a property of human cognitive architecture (i.e., the basic processes and modes of composition that together afford cognition) whereby cognitive capacity is organized around groups of related abilities. A standard example since the original formulation of the problem (Fodor & Pylyshyn, 1988) has been that you don't find people with the capacity to infer John as the lover from the statement *John loves Mary* without having the capacity to infer Mary as the lover from the related statement *Mary loves John*. An instance of systematicity is when a cognizer has cognitive capacity c_1 *if and only if* the cognizer has cognitive capacity c_2 (see McLaughlin, 2009). So, e.g., systematicity is evident where one has the capacity to remove the jokers if and only if one has the capacity to remove the aces (assuming, of course, one has the capacity to identify jokers and aces).

The classical explanation for systematicity has two components: (1) combinatorial syntactically structured representations; and (2) processes that are sensitive to (i.e., compatible with) those syntactic structures. In a classical cognitive architecture, mental representations of constituent entities (e.g., *John*, *Mary*) are tokened (instantiated) whenever the mental representations of their complex hosts (e.g., *John loves Mary*) are tokened, with the meaning of a complex host representation obtained (recursively) from the meaning assigned to its constituent mental representations and their syntactic relationships. By analogy to language, this form of mental representation is called a *language of thought* (LoT) (Fodor, 1975).

The three aspects of systematicity, i.e., *systematicity of representation*, *systematicity of inference*, and *compositionality of representation* (Fodor & Pylyshyn, 1988), can often be derived from classical cognitive architectures, because the same component processes are often used for each and every member of a group of systematically-related capacities. For instance, a classical system has the capacity to represent *John loves Mary* if and only if the system has a capacity to represent *Mary loves John* when the common component process is something like a production rule: $S \rightarrow Agent\ loves\ Patient$ (where *John* and *Mary* are produced from *Agent* and *Patient* by other production rules)—systematicity of representation. Likewise, a classical system has the capacity to infer John as the lover in *John loves Mary* if and only if it has the capacity to infer Mary as the lover in *Mary loves John* given a common process that is sensitive to the syntactic structure whereby the lover constituent is represented by the first token—systematicity of inference. And, again, the capacity

[1] This paper is a short version of Phillips and Wilson (2012).

to assign the semantic content of John being the lover of Mary to the representation *John loves Mary* if and only if there is the capacity to assign the semantic content of Mary being the lover of John to the representation *Mary loves John* derives from the tokening principle (above) mediating classical representations and processes: the process for juxtaposing tokens (symbols) *John*, *loves*, and *Mary* to form *John loves Mary* with corresponding semantic content is the same process that is used to form *Mary loves John* with corresponding content.

Classical compositionality would seem to provide an elegant explanation for systematicity with regard to recursive capacities, even though it fails to provide a full account of systematicity generally (Aizawa, 2003).[2] For recursive definitions, like the deck of cards, one self-referencing rule typically covers all cases (bar the terminating case, such as the empty deck). For example, removing jokers from a single hand, or an entire deck invokes the same component process. The two tasks only differ in the number of recursive steps.

Classical, but not systematically recursive

However, the classical explanation with regard to recursive capacities still suffers the same general problem that it suffers for non-recursive capacities. To illustrate, suppose one card game requires removing the lowest value card in the hand dealt, while another card game requires removing the highest value card. Suppose the following recursive procedure, *lowest*, for identifying the lowest valued card in a deck of cards (containing at least one card):

$$\begin{aligned} \mathit{lowest}(c:cs) &= \mathit{lower}(c, \mathit{lowest}(cs)) \\ \mathit{lowest}(c:[]) &= c \end{aligned}$$

where a deck of cards $c:cs$ is represented by a recursively defined list with c as the top card and cs as the remaining deck, $[]$ is the empty deck, and *lower* returns the lower of two cards. Suppose, also, the following classical non-recursive procedure, *highest*, for identifying the highest valued card:

$$\begin{aligned} \mathit{highest}(cs) \quad = \quad & (i, \mathit{high}) \leftarrow (0, \mathit{undefined}) \\ & \textbf{while } i < n \textbf{ do} \\ & \quad (i, \mathit{high}) \leftarrow (i+1, \mathit{higher}(\mathit{high}, cs_i)) \\ & \textbf{return } \mathit{high} \end{aligned}$$

where deck cs is represented by an array of n cards with position indexed by i (i.e., cs_i is the ith card), *high* maintains a representation of the (currently) highest card, *higher* returns the higher of two cards (*undefined* is some value guaranteed to be lower than any card), and $\leftarrow$ indicates variable-value assignment. Clearly, the two procedures do not share any component processes, and so do not provide a basis for systematicity, even though systematicity could be supported when both tasks are implemented in either the first style only, or the second style only. Notice that we are not unfairly stressing classical theory by apportioning capacity at the level of constituents—systematicity concerns "molecular", not "atomic" capacities (Fodor & Pylyshyn, 1988). Rather, given constituent capacities *lower* and *higher*, classical theory admits two independent compositional forms, as the example illustrates.

In this paper, we extend our category theory explanation to recursive capacities using universal constructions called an *initial F-algebra* and a *final F-coalgebra*, which have been extensively developed in computer science as a theoretical basis for recursive computations (Manes & Arbib, 1986). Our previous work (Phillips & Wilson, 2010, 2011) dealt with non-recursive domains using a kind of universal construction called *adjoint functors*—a *functor* is a way relating categories, which can be viewed as a way of constructing objects and morphisms from one category based on those in another. The current work uses *endofunctors*, which relate categories to themselves, hence their relevance to recursion.

Category theory: *F*-(co)algebras

A category theory treatment of recursion starts with the concept of an *F-algebra* constructed on an endofunctor F.[3]

Definition (*F*-algebra). For an endofunctor $F : \mathbf{C} \to \mathbf{C}$, an *F-algebra* is a pair (A, α), where A is an object and $\alpha : F(A) \to A$ is a morphism in $\mathbf{C}$.

For example, if $F(A) = A \times A$, then the addition operator $+ : A \times A \mapsto A$ is an F-algebra.

Definition (*F*-algebra homomorphism). An *F-algebra homomorphism* $h : (A, \alpha) \to (B, \beta)$ is a morphism $h : A \to B$ (in $\mathbf{C}$) such that the following diagram *commutes*:

$$\begin{array}{ccc} F(A) & \xrightarrow{\alpha} & A \\ {\scriptstyle F(h)}\downarrow & & \downarrow{\scriptstyle h} \\ F(B) & \xrightarrow[\beta]{} & B \end{array} \qquad (1)$$

That is, $h \circ \alpha = \beta \circ F(h)$.

Definition (*F*-algebra category). For endofunctor $F : \mathbf{C} \to \mathbf{C}$, an F-algebra category $\mathbf{Alg}(F)$ has F-algebras (A, α) for objects, and F-algebra homomorphisms $h : (A, \alpha) \to (B, \beta)$ for morphisms.

Definition (Initial algebra). An *initial F-algebra* (A, in), also called *initial algebra*, is an *initial object* in the category

[2] Classical theory fails to provide a complete explanation because one can construct syntactically compositional systems that support some but not all members of a group of systematically-related cognitive capacities. Additional (*ad hoc*) assumptions are needed to derive only those classical cognitive architectures that support systematicity (Aizawa, 2003). This problem echoes the one originally raised against connectionism as a *theory* of cognitive architecture (Fodor & Pylyshyn, 1988; Fodor & McLaughlin, 1990).

[3] We omit definitions of *category*, *functor*, and *initial object* (see Phillips & Wilson, 2010, 2011, for introductions tailored to the systematicity problem—short versions appear in CogSci10/CogSci11 proceedings). An example category is **Set**, whose *objects* are sets, *morphisms* are functions, and *composition* of morphisms is function composition. Functors are like generalized functions, but map morphisms to morphisms as well as objects to objects. An *endofunctor* is a functor whose domain and codomain are the same category.

of F-algebras $\mathbf{Alg}(F)$. I.e., there exists a unique F-algebra homomorphism from (A, in) to every F-algebra in $\mathbf{Alg}(F)$.

Definition (Catamorphism). A *catamorphism* $h : (A, in) \to (B, \beta)$ is the unique F-algebra homomorphism from initial F-algebra (A, in) to F-algebra (B, β). That is, $h \circ in = \beta \circ F(h)$, and the uniquely specified h for each such β is denoted cata β (i.e., $h =$ cata β). In Diagram 1, replace α with *in* and h with cata β for a commutative diagram indicating a catamorphism.

Hence the importance of initial algebras to the systematicity of recursive capacities: every algebra (process) factors through an initial algebra in a category $\mathbf{Alg}(F)$ that has one.

Duals: F-algebra, initial algebra, and catamorphism have dual constructs called *F-coalgebra*, *final coalgebra*, and *anamorphism* (respectively), which we shall also use.

Systematicity: List-related capacities

Returning to the example raised as a problem for classical theory: a common task is to select the smallest or largest item in a collection. Systematicity, in this case, means that if one has the capacity to distinguish the relative sizes of items, and one has the capacity to identify the smallest item in a list, then one also has the capacity to identify the largest item in a list. (Notation: 1_A is the identity morphism for A; l_v is a constant function returning v.) List-related capacities are constructed from the functor $F_A : S \mapsto A \times S, f \mapsto 1_1 + 1_A \times f$. ($\phi : x \mapsto y$ means $\phi(x) = y$.) The algebras are the pairs $(S, [\mathsf{l}_v, f])$. An initial algebra for lists is the pair $(L, [empty, cons])$, where L is a set of lists, and $[empty, cons] : 1 + A \times L \to L$ is the list-constructing morphism, consisting of the constant function $empty : 1 \to L$ for constructing the empty list $[]$, and the function $cons : A \times L \to L; (a, l) \mapsto a \cdot l$ for constructing the list with element $a \in A$ prepended ($\cdot$) to list $l \in L$. Here $1 + A \times L$ is the disjoint union of a 1-element set with the cartesian product of A and L. If, for example, A is the natural numbers $\mathbb{N}$, then L is the set of all finite natural number lists. Catamorphisms from this initial algebra have the form $foldL[\mathsf{l}_v, f] : L \to S$, where $foldL[\mathsf{l}_v, f]$:

$$[] \mapsto v$$
$$(a, l) \mapsto foldL[\mathsf{l}_{f(v,a)}, f](l)$$

The catamorphism for identifying the smallest number is $foldL[1_\infty, min]$, where $min : (x, y) \mapsto x$, *if* $x \leq y$, *else* y returns the smaller of two items, indicated in commutative diagram

$$\begin{array}{ccc} 1 + \mathbb{N} \times L & \xrightarrow{[empty, cons]} & L \\ {\scriptstyle 1_1 + 1_{\mathbb{N}} \times foldL[\mathsf{l}_\infty, min]} \downarrow & & \downarrow {\scriptstyle foldL[\mathsf{l}_\infty, min]} \\ 1 + \mathbb{N} \times \mathbb{N} & \xrightarrow[[\mathsf{l}_\infty, min]]{} & \mathbb{N} \end{array} \tag{2}$$

E.g., $foldL[\mathsf{l}_\infty, min]([2, 1, 3]) = min(2, min(1, min(3, \infty))) = 1$. By replacing *min* in Diagram 2 with $max : (x, y) \mapsto x$, *if* $x \geq y$ *else* y, and ∞ with 0 (or, $-\infty$ for lists of integers or reals), we have the catamorphism that corresponds to identifying the largest number. For example, $foldL[\mathsf{l}_0, max]([2, 1, 3]) = max(2, max(1, max(3, 0))) = 3$. Since the two computations have the morphism $[empty, cons]$ as the common component, this arrangement accounts for systematicity with respect to these capacities. Since the catamorphisms are uniquely determined, we have an account of systematicity without further (*ad hoc*) assumptions.

Systematicity: language-related capacities

In this domain, we use an artificial grammar (for arithmetic expressions) to illustrate our explanation for systematicity with regard to language-related capacities. Artificial grammars are often used, because their forms are more easily adapted to the question at hand. Up to this point, we have addressed systematicity with respect to inference, e.g., why the capacity to infer the smallest list item is systematically related to the capacity to infer the largest list item—*systematicity of inference*. This aspect of systematicity assumes that the cognitive system also has the capacity to systematically represent the entities from which such inferences are made—*systematicity of representation*. Here, we also provide a category theory explanation for systematicity of representation, using the closely related, dual notion of an F-coalgebra.

Arithmetic expressions: systematicity of inference

The example in this section is based on Hutton (1998), but adapted to model cognitive capacity for evaluating numerical expressions. The category of F-algebras that includes language-related capacities is constructed from the functor $F_A : S \mapsto A + S \times S, f \mapsto 1_A + f \times f$. The F-algebras for the category $\mathbf{Alg}(F_A)$ are the pairs $(S, [f, g])$, where $[f, g] : A + S \times S \to S$, $f : A \to S$ is a unary function, and $g : S \times S \to S$ is a binary function. An initial algebra in this category is $(T, [leaf, branch])$, where T is the set of trees of type A, $[leaf, branch] : A + T \times T \to T$, $leaf : A \to T; a \mapsto \langle a \rangle$ returns a tree consisting of a single leaf $a \in A$, and $branch : T \times T \to T; (l, r) \mapsto \langle l, r \rangle$ returns a tree consisting of a left branch l and a right branch r, where $l, r \in T$. For example, a binary tree of numbers $\langle \langle 1 \rangle, \langle \langle 2 \rangle, \langle 3 \rangle \rangle \rangle$ has a leaf 1 as its left branch, and a tree, with left leaf 2 and a right leaf 3, as its right branch. A catamorphism from initial algebra $(T, [leaf, branch])$ to an arbitrary F-algebra $(S, [f, g])$ in $\mathbf{Alg}(F_A)$ is the recursive function *foldT* (i.e., fold for trees), defined as follows. The (higher-order) function *foldT* takes a unary function $f : A \to S$ and a binary function $g : S \times S \to S$ and returns the recursive function $foldT\,[f, g] : T \to S$, where

$$\langle a \rangle \mapsto f(a)$$
$$\langle l, r \rangle \mapsto g(foldT\,[f, g](l), foldT\,[f, g](r))$$

and T is a set of trees of type A, indicated in diagram

$$\begin{array}{ccc} A + T \times T & \xrightarrow{[leaf, branch]} & T \\ {\scriptstyle 1_A + foldT\,[f,g] \times foldT\,[f,g]} \downarrow & & \downarrow {\scriptstyle foldT\,[f,g]} \\ A + S \times S & \xrightarrow[[f,g]]{} & S \end{array} \tag{3}$$

Suppose participants are given arithmetic expressions involving a particular operator, say, *addition*, e.g., $(1+2)+(2+3)$, which they are required to evaluate. Given that participants can correctly evaluate such expressions, there is a host of other capacities that are also afforded provided that they have some other basic knowledge. For example, given knowledge of another binary operator, say, *subtraction*, participants can also evaluate the related expression $(4-2)-(2-1)$ as 1. The specific catamorphism for the addition case is given by replacing A in Diagram 3 with the set of numbers N, f with identity morphism 1_N, and g with addition $+$. For the case of *subtraction*, the binary operator $(+)$ for addition is replaced with $(-)$ in Diagram 3. Hence, the second task is computed as $foldT\,[1_N,(-)]$. The universal construction common to these two capacities is the morphism $[leaf, branch]$. So, the explanation for systematicity is essentially the same as the explanations we provided for list- and number-related capacities, albeit based on a different underlying functor—the capacities for evaluating expressions involving addition and subtraction contain $[leaf, branch]$ as the common factor.

Arithmetic expr.: systematicity of representation

Evaluating trees is an example of *systematicity of inference* (Fodor & Pylyshyn, 1988; Aizawa, 2003; Phillips & Wilson, 2011). However, such expressions are not given to the cognitive system in tree-form. Typically, such trees are assumed to be constructed from an input (list of characters) by another process. The input may take on several different formats: e.g., numeric/symbolic, as in "1+(2+3)", or word form, as in *one plus (two plus three)*, which correspond to the same tree. Again, these two forms are systematically related: one has the capacity to represent the expression "1+(2+3)" if and only if one has the capacity to represent the expression *one plus (two plus three)* assuming, of course, a person knows that *one*, *two* and *three* denote the same things as 1, 2 and 3 (respectively), and *plus* denotes the same thing as $+$. This form of systematicity is called *systematicity of representation* (Fodor & Pylyshyn, 1988; Aizawa, 2003; Phillips & Wilson, 2011). In this section, we show how systematicity of representation is addressed using coalgebras.

Constructing trees from lists is achieved by a dual construction called an *F-coalgebra* (Hutton, 1998). The explanation for systematicity in this case proceeds in a "dual" manner: i.e., every morphism in a category of F-coalgebras with a terminal (dual to initial) object, called a *final coalgebra* (dual to initial algebra) is composed of a unique *anamorphism* (dual to catamorphism) and a common final coalgebra.

Final coalgebras derive from their dual definitions of initial algebras in the category of F-algebras $\mathbf{Alg}(F_A)$ on the functor $F_A : \mathbf{Set} \to \mathbf{Set}; S \mapsto A + S \times S, f \mapsto 1_A + f \times f$. A final coalgebra in this category is $(T, (p_{\langle\rangle} \to fmleaf, fmbranch))$, where conditional $p_{\langle\rangle} \to fmleaf, fmbranch$ consists of a condition $p_{\langle\rangle} : T \to Bool$ that tests whether $t \in T$ is a leaf (i.e., $t = \langle a\rangle, a \in A$), or a branch (i.e., $t = \langle l, r\rangle, l, r \in T$), and associates functions $fmleaf : T \to A, \langle a\rangle \mapsto a$, for retrieving a value from a leaf, and $fmbranch : T \to T \times T, \langle l, r\rangle \mapsto (l, r)$, for retrieving a pair of left and right subtrees from a branch. The dual category $\mathbf{CoAlg}(F_A)$ has F-coalgebras $(S, (p \to f, g))$ as objects, and F-coalgebra homomorphisms as morphisms. The anamorphism associated with this final coalgebra is called *unfoldT* (i.e., unfold for trees), defined recursively as $unfoldT\,(p \to f, g) : S \to T$

$$\begin{array}{ll} s \mapsto \langle f(s)\rangle & \text{if } p(s) \\ s \mapsto \langle unfoldT\,p?\,(p_1 \circ g(s)), unfoldT\,p?\,(p_2 \circ g(s))\rangle & \neg p(s) \end{array}$$

where $p?$ abbreviates $(p \to f, g)$. The final coalgebra and anamorphism are indicated in commutative diagram

$$\begin{array}{ccc} S & \xrightarrow{\quad p \to f,g \quad} & A + S \times S \\ {\scriptstyle unfoldT\,p?}\downarrow & & \downarrow{\scriptstyle 1_A + unfoldT\,p? \times unfoldT\,p?} \\ T & \xrightarrow[p_{\langle\rangle} \to fmleaf, fmbranch]{} & A + T \times T \end{array} \tag{4}$$

Diagram 4 indicates the general form of the anamorphism from which we specify a particular $p?$ (i.e., $p \to f, g$) for our domain of arithmetic expressions. That is, we need to define the test function $p : L \to Bool$, where $Bool = \{\text{True}, \text{False}\}$ that determines whether an expression indicates a simple (value) or complex expression, and associated functions $f : L \to N$ and $g : L \to L \times L$ for transforming simple and complex expressions into numbers and expression pairs (respectively).

Specifications of f and g (in Diagram 4) are obtained from case analysis. Examples of simple expressions, which indicate values, are: "1", "(2)", and "((3))", i.e., any well-formed expression that does not contain the "+" character. A complex expression is any well-formed expression that is not simple. So, p is the function $isVal : l \mapsto$ "+" $\notin l$. Since f is associated with $p(l)$ being true, we require a function to convert a string into a (internal) representation for the corresponding number, i.e., f is the function $str2num : L \to N$. Finally, we need a function g for complex expressions. Examples of complex expressions include: "1+2", "1+(2+3)", "(1+2)+3", "(1+2)+(3+4)", and so on. The purpose of g is to split an expression into two subexpressions, one corresponding to the left branch of the tree, and the other to the right branch. That is, g must split the expression at the topmost operator into two subexpressions containing the strings before and after the "+" symbol, after stripping off the outer brackets. Identifying the split point is also determined by case analysis: Basically, the split point is the first instance of "+" in the absence of an unmatched right bracket ")". So, one simply maintains a counter, starting from 0 (i.e., no unmatched brackets, or top level), which is incremented/decremented on every occurrence of a left/right bracket, when read from left to right. So, g is the function $split : L \to L \times L$. Thus, the function for parsing expressions into trees is the anamorphism $unfoldT(isVal \to str2num, split)$.

Systematicity of representation (in this example, constructing trees) is obtained in the same way as systematicity of inference ("destructing" trees). To represent the same tree from

the expressions in word form, one simply replaces the argument $isVal \rightarrow str2num, split$ as appropriate. For example, the function *str2num* is replaced with, say, *word2num* which converts numbers in word form (e.g., "one", "two", etc.) to their corresponding internal representation of number. In any case, the resulting anamorphism factors through the same universal morphism, i.e., $p_{\langle\rangle} \rightarrow fmleaf, fmbranch$ from Diagram 4.

Given initial algebra $in : F(A) \rightarrow A$ in a category $\mathbf{Alg}(F)$, the corresponding final coalgebra $fin : A \rightarrow F(A)$ is guaranteed to exist, because $F(A) \cong A$, so *in* has as inverse *fin*. Thus, further (*ad hoc*) assumptions are not required to guarantee a correspondence between expressions and evaluations since they are indivisibly bound by the (final) initial (co)algebra. By contrast, classical theory assumes that the processes for constructing syntactically compositional representations and the processes for systematically transforming those representations correspond (Phillips & Wilson, 2011).

Discussion

Our explanation in regard to recursive domains employs the same category theory construct (i.e. universal construction) as our previous explanations for (quasi-)systematicity in regard to non-recursive domains (Phillips & Wilson, 2010, 2011), albeit with different kinds of functors: here, for recursive domains, the universal constructions involved endofunctors (i.e., where the domain and codomain are the same category), whereas for non-recursive domains, the universal constructions involved adjoint functors (which are reciprocating, though not necessarily inverse, functorial maps between categories that are not necessarily the same. Every composition of left and right adjoints is an endofunctor, but not every endofunctor can be decomposed into a pair of adjoint functors. So, having some (primitive) form of systematicity over a recursive domain does not imply having systematicity for non-recursive domains. Nor, for that matter, does having the systematicity property for one recursive domain (e.g., numbers) imply the having the systematicity property for another recursive domain (e.g., lists), when the universal constructions involve functors not related by a natural isomorphism (Manes & Arbib, 1986)—this distinction also applies to non-recursive domains. This functorial distinction has implications for comparative and developmental psychology (discussed later).

Limitations

Our theory may be incomplete at two points: one point is where competence meets performance, such as when supposed systematically related capacities span memory or cognitive complexity limits. The other point is where systematic cognition meets non-systematic cognition: not all cognition is regarded as systematic; idioms (e.g., *John kicked the bucket*—i.e., he died—is not systematic with *Mary kicked the bucket [with her foot]*) are a paradigm (Fodor & Pylyshyn, 1988). We discuss our theory in the context of both cases.

Competence versus performance: In the case of lists where the morphism f is not associative (e.g., subtraction), computing with a right-fold version of list fold means keeping all list items in memory (if presented once only), so systematicity would not extend beyond lists of more than a few items. Such cases are generally not regarded as evidence against the systematicity property—human cognition is *ceteris paribus* (e.g., memory requirements being the same) largely systematic (see McLaughlin, 2009). Nonetheless, a more complete theory will address both aspects of cognition. Category theory may also provide independent principles for performance, since cognitive development-related limits in children were identified with the arity of the (co)product underlying the task (Phillips, Wilson, & Halford, 2009): e.g., the ability of children older than the median age of five years to perform transitive inference and class inclusion in the more difficult condition versus children younger than five was related to (co)product arity, i.e., binary versus unary (co)products. Note that here, too, the difference in "complexity" of the endofunctors for number (no/unary product of functors, not given), list (binary product of constant and identity functors) and tree (binary product of two identity functors). However, performance related differences are beyond the scope of our theory as it currently stands.

Systematic versus non-systematic cognition: Category theory also provides a principled means for joining two cognitive (sub)systems via (co)products of categories, where one category models systematic cognitive capacity and the other non-systematic capacity, and (say) the coproduct category models both. However, as Aizawa (2003) explains, the required explanatory standard for hybrid theories is higher, because one must also explain why/when component theories are invoked. A possible reason is efficiency. A primitive form of addition is supported (systematically) by the category of F-algebras that included number-related capacities via *foldN* (not presented here), where the number of iterations is proportional to the size of the addends. The time needed to add numbers can be reduced by memorizing the addition table for small numbers, which is what children are taught to do. However, addition via memorized associations is not a systematic process: one can memorize part of a table without memorizing the other part; this is an analog of the phrase-book example in language (Fodor & Pylyshyn, 1988). Utility may drive the cognitive system to employ a faster, but non-systematic process, but it is also outside the scope of our current theory.

Perspective

At the core of our category theory explanation for systematic recursive capacity is a special pair of dual constructions: an (final) initial (co)algebra in a category of (co)algebras on a polynomial functor F. Although one can reverse the direction of any collection of arrows to form a dual, such duals may not exist in the category of interest (e.g., some categories have initial objects but no final objects). Yet, for categories of (co)algebras on a polynomial functor (final) initial (co)algebras are guaranteed to exist (Manes & Arbib, 1986), and an initial algebra $in : F(A) \rightarrow A$ is guaranteed to have an inverse $fin : A \rightarrow F(A)$, because the component objects are

isomorphic (i.e., $A \cong F(A)$), which constitutes a final coalgebra for the domains we have investigated. So, the systematic relationship between representation and inference is guaranteed without further (*ad hoc*) assumptions, in contrast to the classical explanation where the link between the two is just assumed (Phillips & Wilson, 2011). Notice that this dual relationship between systematicity of representation and systematicity of inference is more general (and more useful) than an inverse. In the arithmetic expressions example, lists were represented as trees (systematicity of representation), but trees were evaluated as numbers (systematicity of inference). This form of duality goes beyond the simple inverse relationship between sentence recognition and generation found in parsing/production rules in a classical approach to language.

The capacity for recursion has been a contentious issue in the broader interests of cognitive science, which includes comparative and developmental psychology. Some argue that recursion is specific to humans and depends on language (Hauser, Chomsky, & Fitch, 2002); more particularly, a fully inductive (recursive) basis for number is specific to adults and distinct from infants' non-inductive basis (Rips, Bloomfield, & Asmuth, 2008). In contrast, others claim a human language-like capacity for recursion in songbirds (Gentner, Fenn, Margoliash, & Nusbaum, 2006) (but, see Corballis, 2007), and that adult understanding of number (in its fully induced form) is founded on a more primitive infant conception (Carey, 2009). See also Gelman and Butterworth (2005), for a review of the debate over the link between number and language. Our category theory treatment of recursive cognitive capacities provides a different perspective on this issue: specifically, the particular systematic capacities for recursion depend on the underlying functor, not a general capacity for recursion, as such. In particular, one can have a basic recursive capacity for number without having a full-blown capacity for language, because the functor underlying recursive number-related capacities does not provide a systematic basis for recursive language-related capacities, though by our account language-related recursive capacities afford number-related recursive capacities. Analysis of the songbird evidence (Gentner et al., 2006) for supposed center-embedded recursion suggested that these birds were using a simple *counting* strategy (Corballis, 2007), which accords with our F-(co)algebraic basis for recursion in cognition, where simple counting involves a fold for numbers, not trees. Thus, other species (and infants) can have elementary recursive capacities without a full-blown capacity for number and language as available in adult humans.

The classicist's approach to cognitive architecture is fundamentally limited not in advocating syntax, but in placing syntax at the foundation of their theory. Given the often *ad hoc* and idiosyncratic choices that go into programming language design, computer scientists in recent decades have turned to category theory for a deeper syntax-free understanding of the principles of computation. Cognitive science, as couched within the framework of computationalism, can likewise do better than lay foundations on the shifting sands of syntax.

Acknowledgments

This work was supported by a Japanese Society for the Promotion of Science Grant-in-aid (22300092).

References

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

Carey, S. (2009). *The origins of concepts*. New York, NY: Oxford University Press.

Corballis, M. C. (2007). Recursion, language, and starlings. *Cognitive Science*, *31*, 697–704.

Fodor, J. A. (1975). *The language of thought*. New York, NY: Crowell.

Fodor, J. A., & McLaughlin, B. P. (1990). Connectionism and the problem of systematicity: Why Smolensky's solution doesn't work. *Cognition*, *35*, 183–204.

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

Gelman, R., & Butterworth, B. (2005). Number and language: how are they related? *Trends in Cognitive Sciences*, *9*(1), 6–10.

Gentner, T. Q., Fenn, K. M., Margoliash, D., & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature*, *440*(7088), 1204–1207.

Hauser, M. D., Chomsky, N., & Fitch, W. T. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*, *298*(5598), 1569–1579.

Hutton, G. (1998). Fold and unfold for program semantics. In *Proceedings of the 3rd ACM SIGPLAN International Conference on Functional Programming.*

Manes, E. G., & Arbib, M. A. (1986). *Algebraic approaches to program semantics*. New York, NY: Springer-Verlag.

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

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. (2011). Categorial compositionality II: Universal constructions and a general theory of (quasi-)systematicity in human cognition. *PLoS Computational Biology*, *7*(8), e1002102.

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., & Halford, G. S. (2009). What do Transitive Inference and Class Inclusion have in common? Categorical (co)products and cognitive development. *PLoS Computational Biology*, *5*(12), e1000599.

Rips, L. J., Bloomfield, A., & Asmuth, J. (2008). From numerical concepts to concepts of number. *Behavioral and Brain Sciences*, *31*(6), 623–687.