Categorial Compositionality: A Category Theory Explanation for the Systematicity of Human Cognition

Steven Phillips[1]*, **William H. Wilson**[2]

1 Mathematical Neuroinformatics Group, Human Technology Research Institute, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Ibaraki, Japan, **2** School of Computer Science and Engineering, The University of New South Wales, Sydney, New South Wales, Australia

Abstract

Classical and Connectionist theories of cognitive architecture seek to explain systematicity (i.e., the property of human cognition whereby cognitive capacity comes in groups of related behaviours) as a consequence of syntactically and functionally compositional representations, respectively. However, both theories depend on *ad hoc* assumptions to exclude specific instances of these forms of compositionality (e.g. grammars, networks) that do not account for systematicity. By analogy with the Ptolemaic (i.e. geocentric) theory of planetary motion, although either theory can be made to be consistent with the data, both nonetheless fail to fully explain it. Category theory, a branch of mathematics, provides an alternative explanation based on the formal concept of adjunction, which relates a pair of structure-preserving maps, called functors. A functor generalizes the notion of a map between representational states to include a map between state transformations (or processes). In a formal sense, systematicity is a necessary consequence of a higher-order theory of cognitive architecture, in contrast to the first-order theories derived from Classicism or Connectionism. Category theory offers a re-conceptualization for cognitive science, analogous to the one that Copernicus provided for astronomy, where representational states are no longer the center of the cognitive universe—replaced by the relationships between the maps that transform them.

Citation: Phillips S, Wilson WH (2010) Categorial Compositionality: A Category Theory Explanation for the Systematicity of Human Cognition. PLoS Comput Biol 6(7): e1000858. doi:10.1371/journal.pcbi.1000858

Editor: Karl J. Friston, University College London, United Kingdom

Received March 15, 2010; **Accepted** June 13, 2010; **Published** July 22, 2010

Funding: The authors received no specific funding for this article.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: steve@ni.aist.go.jp

Introduction

For more than two decades, since Fodor and Pylyshyn's seminal paper on the foundations of a theory of cognitive architecture (i.e., roughly, the component processes and their modes of composition that together comprise cognitive behaviour) [1], the problem of explaining systematicity has remained unresolved [2] despite numerous Classicist and Connectionist attempts [3–7]. In general terms, the problem of systematicity for a theory of cognition is to explain why various cognitive abilities are intrinsically connected in the sense that the capacity to exhibit some abilities is indivisibly linked to the capacity to exhibit some other related abilities. Why, for example, is it the case that if one has the ability to infer that *John* is the lover from *John loves Mary*, then one also has the ability to infer that *Mary* is the lover from *Mary loves John*, where both abilities involve a common relation, *loves*? That is to ask, in general: what is it about our cognitive system that necessitates a particular group-oriented distribution of cognitive capacities, whereby you don't find people with the capacity for some but not all the behaviours pertaining to the same group (excluding, of course, individuals who lack a particular capacity for reasons clearly unrelated to normal development, because of brain damage for example)? Although the debate over what systematicity implies for a theory of cognition has many aspects (see [2]), the generally accepted common ground is that: systematicity is a property of some (though not all) components of human cognition; a complete theory of human cognitive architecture must include an explanation for this property; and no theory of cognition has a satisfactory explanation for it. In the remainder of this section, we outline the systematicity property and the main problem it still poses for existing theories, what is required for a theory to explain it, and how our approach meets those requirements.

The systematicity problem consists of three component problems:

1. *Systematicity of representation*—why is it the case that the capacity to generate some representations (e.g., the representation `John loves Mary`) is intrinsically linked to the capacity to generate some other representations (e.g., the representation `Mary loves John`)?
2. *Systematicity of inference*—why is it the case that the capacity to make some inferences (e.g., that `John` is the lover in the proposition `John loves Mary`) is intrinsically linked to the capacity to make some other inferences (e.g., that `Mary` is the lover in the proposition `Mary loves John`)?
3. *Compositionality of representation*—why is it the case that the capacity for some semantic content (e.g., the thought that `John loves Mary`, however that thought may be represented) is intrinsically linked to the capacity for some other semantic content (e.g., the thought that `Mary loves John`, however that thought may also be represented)?

These problems are logically independent—one does not necessarily follow from another [8], and so a theory is required

Author Summary

Our minds are not the sum of some arbitrary collection of mental abilities. Instead, our mental abilities come in groups of related behaviours. This property of human cognition has substantial biological advantage in that the benefits afforded by a cognitive behaviour transfer to a related situation without any of the cost that came with acquiring that behaviour in the first place. The problem of systematicity is to explain why our mental abilities are organized this way. Cognitive scientists, however, have been unable to agree on a satisfactory explanation. Existing theories cannot explain systematicity without some overly strong assumptions. We provide a new explanation based on a mathematical theory of structure called Category Theory. The key difference between our explanation and previous ones is that systematicity emerges as a natural consequence of structural relationships between cognitive processes, rather than relying on the specific details of the cognitive representations on which those processes operate, and without relying on overly strong assumptions.

to explain all three, though for some theories an explanation for one property may entail explanations for others.

Classicists and Connectionists employ some form of combinatorial representations to explain systematicity. For Classicists, representations are combined in such a way that the tokening of complex representations entails the tokening of representations of their constituent entities, so that the syntactic relationships between the constituent representations mirror the semantics ones—systematicity is a result of a combinatorial syntax and semantics [1]. For Connectionists, representations of complex entities are constructed more generally so that their tokening does not necessarily imply tokening constituent entity representations [5,6]. An example of a Classicist's representation of `John loves Mary` would be `loves (John, Mary)`, and a Connectionist representation would be a tensor product so that the vectors representing `John`, `loves`, and `Mary` do not literally appear anywhere in the tensor representation. We refer to the former as *classical compositionality*, and the latter as *connectionist* (or, *functional) compositionality*.

In general, a Classical or Connectionist architecture can demonstrate systematicity by having the "right" collection of grammatical rules, or functions such that one capacity is indivisibly linked to another. Suppose, for example, a Classical system with the following three rules:

G1:

```
P        →  Agent loves Patient
Agent    →  John | Mary
Patient  →  John | Mary.
```

G1 provides the capacities to generate all four representations (i.e., `John loves John`, `John loves Mary`, etc.), and these capacities are indivisibly linked, because absence of any one of those rules means the system cannot generate any of those representations. In no case can the system generate one without being able to generate the others. So, this Classical architecture has the systematicity of representation property with respect to this group of four propositions. A tensor product [9], or Gödel numbering [5] scheme is a functionally compositional analogue of this explanation. Systematicity of inference follows from having additional processes that are sensitive to the structure of these representations. For Classical architectures, at least, compositionality of representation also follows, because the semantic content of a complex representation is built up from the semantic contents of the constituents and their syntactic relationships [8]. Aizawa [2,8] disputes whether a Connectionist architecture can also demonstrate compositionality of representation. Regardless, though, neither Classicism, nor Connectionism can derive theories that provide a full account of systematicity [2].

A demonstration of systematicity is not an explanation for it. In particular, although grammar G1 has the systematicity of representation property, the following grammar:

G2:

```
P        → John loves Patient | Agent loves Mary
Agent    → John | Mary
Patient  → John | Mary
```

does not. This architecture cannot generate a representation of the proposition `Mary loves John` even though it can generate representations of both `John` and `Mary` as agents and patients, and the `John loves Mary` proposition. The essential problem for Classical theory—likewise Connectionist theory—is that syntactic compositionality by itself is not sufficient without some additional assumptions for admitting grammars such as G1 that have the systematicity property, while excluding grammars such as G2 that do not. An explanation for systematicity in these cases turns on the nature of those additional, possibly *ad hoc* assumptions.

An explanatory standard for systematicity

To further clarify what is required of a theory to explain systematicity [1,3], Aizawa [2] presents an explanatory standard for systematicity and the problem of *ad hoc* assumptions, which we follow, by analogy with the Ptolemean (geocentric) versus Copernican (heliocentric) explanations for the motions of the planets (see [10] for a review). The geocentric explanation for planetary motion places the Earth at the center of the other planets' circular orbits. Although this theory can roughly predict planetary position, it fails to predict periods of apparent retrograde motion for the superior planets (i.e. Mars, Jupiter, etc.) across the night sky without the assumption of *epicycles* (i.e., circular orbits with centers that orbit the Earth). This additional assumption is *ad hoc* in that it is unconnected with the rest of the theory and motivated only by the need to fit the data—the assumption could not be confirmed independently of confirming the theory. The heliocentric explanation, having all planets move around the Sun, eschews this *ad hoc* assumption. Retrograde motion falls out as a natural consequence of the positions of the Earth and other planets relative to the Sun. Tellingly, as more accurate data became available, the geocentric theory had to be further augmented with epicycles on epicycles to account for planetary motion; not so for the heliocentric theory.

The theory of planetary motion, of course, does not end there. The heliocentric theory, with its circular orbits, cannot explain the elliptical motion of the planets without further assumptions, and so was superseded by Newtonian mechanics. Newtonian mechanics cannot explain the precession of planetary orbits, and was in turn superseded by Einstein's theory of relativity. In each case, the superseding theory incorporates all that was explained by the preceding theory. Evaluating competing theories in this manner has an extensive history in science, and so one may expect it to be a reasonable standard for an explanation of systematicity in cognitive science.

Aizawa [2] notes that although philosophers of science may not have a precise definition for the concept of an *ad hoc* assumption, one can nonetheless usefully characterize the idea by analogy with generally accepted examples, such as the assumption of epicycles, which we just mentioned. Another example Aizawa uses is the Creationist versus Darwinian theory of speciation, where the appeal to a supernatural being to explain the existence of different species is an *ad hoc* assumption. The general sense in which a theory fails to provide a satisfactory explanation by its appeal to *ad hoc* assumptions is when those additional, so called auxiliary, assumptions are unconnected to the core assumptions and principles of the theory, motivated only by the need to fit the data, and cannot be confirmed independently of confirming the theory. In this sense, the core theory has no explanatory power for the particular phenomenon of interest. Note that an auxiliary assumption is not necessarily *ad hoc*, nor is it precluded from subsequent inclusion into the set of core assumptions of the modified theory. Orthogonal experiments may provide confirmatory data for an auxiliary assumption, independent of the theory in question. Observations of the Jovian moons would have been the sort of independent confirmatory evidence for epicycles, had such data been available at the time, to justifiably include it as one of the core assumptions. However, the assumption that all heavenly bodies are governed this way ultimately proved untenable. The kind of theory sought here is one where systematicity necessarily follows without requiring such *ad hoc* assumptions. This characterization guides our analysis of the problem posed by the systematicity property, and our explanation for it.

The problem for Classical and Connectionist theories is that they cannot explain systematicity without recourse to their own *ad hoc* assumptions [2]. For Classicism, having a combinatorial syntax and semantics does not differentiate between grammars such as G1 and G2. For Connectionism, a common recourse to learning also does not work, whereby systematicity is acquired by adjusting network parameters (e.g., connection weights) to realize some behaviours—training set—while generalizing to others—test set. Learning also requires *ad hoc* assumptions, because even widely used learning models, such as feedforward [11] and simple recurrent networks [12], fail to achieve systematicity [13–17] when construed as a degree of generalization [18,19]. Hence, neither Classical nor Connectionist proposals satisfy the explanatory standard laid out by Fodor and Pylyshyn [1] and Fodor and McLaughlin [3] (see also [20], Appendix), and further articulated by Aizawa [2]. Ironically, failure to meet this criterion was one of the reasons Classicists rejected Connectionist explanations for systematicity. The import of Aizawa's analysis is that the same shortcoming also befalls Classicism, and so an explanation for systematicity is still needed. In this regard, it would appear that the 90s were also the "lost decade" for cognitive science.

In hindsight, the root of the difficulty that surrounds the systematicity problem has been that cognitive scientists never had a theory of structure to start with (i.e. one that was divorced, or at least separated from specific implementations of structure-sensitive processes). In fact, such a theory has been available for quite some time, but its relevance to one of the foundational problems of cognitive science has not previously been realized. Our category-theory based approach addresses the problem of *ad hoc* assumptions because the concept of an *adjunction*, which is central to our argument, ensures that the construct we seek not only exists, but is unique. That is to say, from this core assumption and category theory principles, the systematicity property necessarily follows for the particular cognitive domains of interest, because in each case the one and only collection of cognitive capacities derived from our theory is the systematic collection, without further restriction by additional (*ad hoc*) assumptions.

Methods

Category theory is a theory of structure *par excellence* [21–23]. It was developed out of a need to formalize commonalities between various mathematical structures [24], and has been used extensively in computer science for the analysis of computation [25–28]. Yet, despite computationalism being the catchcry of many psychologists since the cognitive revolution, applications of category theory to cognitive psychology have been almost non-existent (but, see [29,30] for two examples). Our explanation of systematicity is based on the concept of an *adjunction*, which depends on the concepts of *category*, *morphism*, *product*, *functor*, and *natural transformation*. So, in this section, we provide formal definitions of these concepts. (For further explanation of some category theory concepts in the context of cognition, see [30].)

An adjunction is a formal means for capturing the intuition that a relationship between mathematical objects is "natural"—additional constructs are unnecessary to establish that relationship (see also [23], p2). The mathematical notion of being natural dates back at least to [24], and the technical aspect is given starting where we define *natural transformation*. In the current context of meeting the explanatory standard for systematicity, identifying a suitable adjunction means that no further (*ad hoc*, or arbitrary) assumptions are needed to define the relationship between a particular cognitive architecture and a desired group of cognitive capacities. Such constructs *look* natural (once understood), but it is the mathematical criterion that definitely establishes naturality.

Category

A *category* $\mathbf{C}$ consists of a class of objects $|\mathbf{C}| = (A, B, \dots)$; a set $\mathbf{C}(A,B)$ of morphisms (also called arrows, or maps) from A to B where each morphism $f : A \to B$ has A as its domain and B as its codomain, including the *identity* morphism $1_A : A \to A$ for each object A; and a composition operation, denoted "$\circ$", of morphisms $f : A \to B$ and $g : B \to C$, written $g \circ f : A \to C$ that satisfy the laws of:

- *identity*, where $f \circ 1_A = f = 1_B \circ f$, for all $f : A \to B$; and
- *associativity*, where $h \circ (g \circ f) = (h \circ g) \circ f$, for all $f : A \to B$, $g : B \to C$ and $h : C \to D$.

The most familiar example of a category is **Set**, which has sets for objects and functions for morphisms, where the identity morphism 1_A is the identity function and the composition operation is the usual function composition operator "$\circ$". Another example, where continuity is important, is the category of metric spaces and continuous functions.

Morphisms

Certain morphisms have important properties that warrant giving them names. Two such morphisms, which we will refer to later, are called isomorphisms and homomorphisms. A morphism $f : A \to B$ is an *isomorphism* if there exists a morphism $g : B \to A$, such that $g \circ f = 1_A$ and $f \circ g = 1_B$. If g exists, then it is the *inverse* of f, also denoted as f^{-1}.

Homomorphisms pertain to categories whose objects have additional internal structure, such as groups. For example, the category **Grp** has groups for objects, and the morphisms are group homomorphisms. A group consists of a set G of elements, and an associative binary operation $*$, satisfying identity and inverse axioms. That is, G has an identity element e, and for each $g \in G$, an inverse element $g^{-1} \in G$, such that $e * g = g = g * e$ and $g^{-1} * g = e = g * g^{-1}$. A *group homomorphism* is a morphism $f : (G, *) \to (H, \#)$, such that $f(g_1 * g_2) = f(g_1) \# F(g_2)$, for all $g_1, g_2 \in G$. Homomorphisms in other categories (e.g., graph homomorphisms) are defined analogously.

Product

A *product* of two objects A and B in a category $\mathbf{C}$ is an object P together with two morphisms $p_1 : P \to A$ and $p_2 : P \to B$, such that for any pair of morphisms $z_1 : Z \to A$ and $z_2 : Z \to B$, there is a unique morphism $u : Z \to P$, such that the following diagram *commutes*:

$$\begin{tikzcd} & Z \arrow[ld, "z_1"'] \arrow[d, dashed, "u"] \arrow[rd, "z_2"] & \\ A & P \arrow[l, "p_1"] \arrow[r, "p_2"'] & B \end{tikzcd} \quad (1)$$

where a broken arrow indicates that there exists exactly one morphism making the diagram commute. To say that a diagram commutes is to mean that the compositions along any two paths with the same start object and the same finish object are the same. So, in this diagram, $z_1 = p_1 \circ u$ and $z_2 = p_2 \circ u$, where p_1 and p_2 are sometimes called projection morphisms. A product object P is *unique up to a unique isomorphism.* That is, for any other product object P' with morphisms $p'_1 : P' \to A$ and $p'_2 : P' \to B$ there is one and only one isomorphism between P and P' that makes a diagram like this one commute. Hence, P is not unique, only unique with respect to another product object via isomorphism. This characteristic has an important consequence for our explanation of systematicity, which we present in the Results section. An essential characteristic of a product object is that the constituents A and B are retrievable via the projection morphisms. P is also written $A \times B$, and since u is uniquely determined by z_1 and z_2, u is often written as $\langle z_1, z_2 \rangle$, and the diagram used in defining a product then becomes

$$\begin{tikzcd} & Z \arrow[ld, "z_1"'] \arrow[d, dashed, "\langle z_1, z_2 \rangle"] \arrow[rd, "z_2"] & \\ A & A \times B \arrow[l, "p_1"] \arrow[r, "p_2"'] & B \end{tikzcd} \quad (2)$$

In **Set**, P is (up to isomorphism) the Cartesian product ($A \times B$, $p_1 : (a,b) \mapsto a$, $p_2 : (a,b) \mapsto b$), where $a \in A$, $b \in B$, and u is the product function $\langle z_1, z_2 \rangle : Z \to A \times B$, sending x to $(z_1(x), z_2(x))$, so that $p_1 \circ u = z_1$ and $p_2 \circ u = z_2$. The "maps to" arrow, $\mapsto$, indicates the action of a function on a domain element, so $f(a) = b$ is equivalent to $f : a \mapsto b$. ($A \times B$ refers both to a general product in any category with products and the more specific Cartesian product in the category **Set**.)

The categorical concept of product is a very general notion of combinatoriality. Not surprisingly, then, Classical and Connectionist notions of combinatoriality can be seen as special cases of categorical products. A grammar like G1 (Introduction), for instance, can be used to realize the Cartesian product of the set of agents and the set of patients (i.e. by employing the first production without the `loves` symbol). A categorical product can also be realized by including suitable rules for inferring the agent and patient from this Cartesian product. (A grammar like G2 cannot realize a Cartesian product, or categorical product; in fact, it realizes a union of two partial products.) Similarly, a Connectionist method such as the outer product of two vector spaces with suitable projections from the outer product space to the original vector spaces also realizes a categorical product. However, an explanation for systematicity requires more than just realization, and as we shall see, additional category theory concepts are needed.

Functor

A functor $F : \mathbf{C} \to \mathbf{D}$ is a structure-preserving map between categories $\mathbf{C}$ and $\mathbf{D}$ that associates each object A in $\mathbf{C}$ to an object $F(A)$ in $\mathbf{D}$; and each morphism $f : A \to B$ in $\mathbf{C}$ to a morphism $F(f) : F(A) \to F(B)$ in $\mathbf{D}$, such that $F(1_A) = 1_{F(A)}$ for each object A in $\mathbf{C}$; and $F(g \circ_{\mathbf{C}} f) = F(g) \circ_{\mathbf{D}} F(f)$ for all morphisms $f : A \to B$ and $g : B \to C$ for which compositions $\circ_{\mathbf{C}}$ and $\circ_{\mathbf{D}}$ are defined in categories $\mathbf{C}$ and $\mathbf{D}$, respectively. The following diagram shows the details of a functor:

$$\begin{tikzcd}[column sep=huge] \mathbf{C} & \mathbf{D} \\ A \arrow[r, mapsto, "F_0"] \arrow[d, "f"] \arrow[dd, bend right=60, "g \circ_{\mathbf{C}} f"'] & F(A) \arrow[d, "F(f)"'] \arrow[dd, bend left=60, "F(g) \circ_{\mathbf{D}} F(f)"] \\ B \arrow[r, mapsto, "F_0"] \arrow[d, "g"] & F(B) \arrow[d, "F(g)"'] \\ C \arrow[r, mapsto, "F_0"] & F(C) \end{tikzcd} \quad (3)$$

where dashed rectangles encapsulate the categories, and arrows between morphisms are omitted. The object and morphism components of a functor are sometimes explicitly distinguished as F_0 and F_1, respectively. Otherwise, the functor component is implicitly identified by its argument.

Functor composition and isomorphism are defined analogously to morphisms (above). That is, the composition of functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{E}$ is the functor $G \circ F : \mathbf{C} \to \mathbf{E}$, sending all objects A in $\mathbf{C}$ to objects $G \circ F(A)$ in $\mathbf{E}$; and morphisms $f : A \to B$ in $\mathbf{C}$ to morphisms $G \circ F(f) : G \circ F(A) \to G \circ F(B)$, such that identity and composition are respected. That is, $G \circ F(1_A) = 1_{G \circ F(A)}$; and $G \circ F(g \circ_{\mathbf{C}} f) = (G \circ F(g)) \circ_{\mathbf{E}} (G \circ F(f))$. A functor $F : \mathbf{C} \to \mathbf{D}$ is an *isomorphic functor*, if and only if there exists a functor $G : \mathbf{D} \to \mathbf{C}$ such that $G \circ F = 1_{\mathbf{C}}$ and $F \circ G = 1_{\mathbf{D}}$, where $1_{\mathbf{C}}$ and $1_{\mathbf{D}}$ are the identity functors sending objects and morphisms to themselves in the respective categories.

Theories of cognition employ some form of representation. Functors provide a theoretical basis for constructing representations. For example, computational systems often employ lists of items, such as numbers. In category theory, lists can be modeled as monoids from the category **Mon** whose objects are monoids, and morphisms are monoid homomorphisms [28]. A *monoid* $(M, \cdot, \epsilon)$ is a set M, with an associative binary operation $\cdot$, and an identity element ϵ, such that $m \cdot \epsilon = m = \epsilon \cdot m$ for all $m \in M$. A *list monoid* $(S^*, \cdot, \epsilon)$ [28] is the set S^* of all ordered lists constructed from set S by concatenation operator $\cdot$, where the identity element ϵ is the empty list (so that, e.g., $[1,2] \cdot \epsilon = [1,2]$). (It is worth noting that strings, e.g., lists of characters, of length 2 over the set S are denoted S^2, and strings of length n denoted S^n. In computer science, $*$ often means "match anything", hence the notation S^* can be read as strings of any length $n \geq 0$.) Lists can be constructed from sets by the functor $List : \mathbf{Set} \rightarrow \mathbf{Mon}$, as indicated in the example diagram

$$\begin{tikzcd}
\{1,2,3\} \arrow[r, mapsto, "lists"] \arrow[d, "sqr"'] & (\{1,2,3\}^*, \cdot, \epsilon) \arrow[d, "maplist(sqr)"] \\
\{1,4,9\} \arrow[r, mapsto, "lists"'] & (\{1,4,9\}^*, \cdot, \epsilon)
\end{tikzcd} \tag{4}$$

where *lists* is the object part of *List* (i.e., $List_0 = lists$) and *maplist* is the morphism part (i.e., $List_1 = maplist$), so that, e.g., $maplist : sqr \mapsto sqrlist$ (i.e., morphism *sqr* is mapped to monoid homomorphism *maplist*(*sqr*), which we will refer to as *sqrlist*). (For simplicity, we have omitted composition with a second morphism in each of the categories and functor mappings, as was shown in Diagram 3.) So, for example, $sqrlist([1,2]\cdot[3]) = sqrlist([1,2,3]) = [1,4,9] = [1,4]\cdot[9] = sqrlist([1,2])\cdot sqrlist([3])$. The examples pertaining to lists were adapted from [28] (Chapter 2), where $L(S)$ in [28] corresponds to our S^*. We choose to label the object component of the functor *lists* rather than *list* to emphasize the fact that the functor constructs a set of lists of numbers from a set of numbers, not just a single list containing those numbers.

The two different sorts of arrows in Diagrams 3 and 4 highlight the constructive nature of functors. The objects are (co)domains with respect to the morphisms within categories, but are themselves elements of larger objects (in general, the class $|\mathbf{C}|$) with respect to the morphisms between categories. In programmer parlance, *sqr* was "lifted" from being a function over numbers to become a function *sqrlist* over lists of numbers. In this way, functors provide a means for constructing new representations and processes from existing ones in a structurally consistent manner.

Notice that the definition of functor does not dictate a particular choice for monoid homomorphism as part of the definition of *List*. A natural choice is to define *maplist*(f) so that functions applied to one-item lists result in one-item lists (i.e., $maplist(f) : [x] \mapsto [f(x)]$). Another choice that turns out to also respect the definition of a functor includes *two* copies of each transformed element (i.e., $maplist'(f) : [x_1, \ldots, x_n] \mapsto [f(x_1), f(x_1), \ldots, f(x_n), f(x_n)]$). In this case,

$$\begin{aligned}
& maplist'(f)([x_1,\ldots,x_n]\cdot[y_1,\ldots,y_m]) \\
&= maplist'(f)([x_1,\ldots,x_n,y_1,\ldots,y_m]) \\
&= [f(x_1),f(x_1),\ldots,f(x_n),f(x_n),f(y_1),f(y_1),\ldots,f(y_m),f(y_m)] \\
&= [f(x_1),f(x_1),\ldots,f(x_n),f(x_n)]\cdot[f(y_1),f(y_1),\ldots,f(y_m),f(y_m)] \\
&= maplist'(f)([x_1,\ldots,x_n])\cdot maplist'(f)([y_1,\ldots,y_m]).
\end{aligned}$$

So, *maplist*′(f) and in particular $sqrlist' : [x_1,\ldots,x_n] \mapsto [x_1^2, x_1^2, \ldots, x_n^2, x_n^2]$ are monoid homomorphisms. In fact, there are many possible monoid homomorphisms that could be chosen to define this functor. Consequently, in the case of an architectural component of a cognitive system, there are many possible ways of constructing structurally consistent representations and processes from existing ones. We need to find a principled way to choose the "right" monoid homomorphism. In the context of explaining systematicity, a similarly principled choice is necessary. To narrow the choice down to a particular monoid homomorphisms, and hence a particular representational scheme, we need two additional category theory concepts: natural transformation and adjunction.

Natural transformation

A *natural transformation* $\eta : F \rightarrow G$ is a structure-preserving morphism from domain functor $F : \mathbf{C} \rightarrow \mathbf{D}$ to codomain functor $G : \mathbf{C} \rightarrow \mathbf{D}$ that consists of $\mathbf{D}-$maps $\eta_A : F(A) \rightarrow G(A)$ for each object A in $\mathbf{C}$, such that $G(f) \circ \eta_A = \eta_B \circ F(f)$, as indicated by the commutative diagram in the category $\mathbf{D}$

$$\begin{tikzcd}
& \mathbf{C} & \\
& A \arrow[d, "f"] & \\
& B \arrow[ddl, "F"'] \arrow[ddr, "G"] & \\
& \mathbf{D} & \\
F(A) \arrow[rr, "\eta_A"] \arrow[d, "F(f)"'] & & G(A) \arrow[d, "G(f)"] \\
F(B) \arrow[rr, "\eta_B"] & & G(B)
\end{tikzcd} \tag{5}$$

Again for expository purposes, we include the source category and functor arrows, which are usually left implicit in such diagrams. When a transformation is natural in the technical sense it seems natural in the intuitive sense, for mathematicians. In fact, category theory was founded in an attempt to formalize such intuitions [24]. We will return to this point about naturality, in the Discussion, as it pertains to an explanation of systematicity without reliance on *ad hoc* assumptions.

A natural transformation is a *natural isomorphism*, or *natural equivalence* if and only if each η_A is an isomorphism. That is, for each $\eta_A : F(A) \rightarrow G(A)$ there exists a $\eta_A^{-1} : G(A) \rightarrow F(A)$ such that $\eta_A^{-1} \circ \eta_A = 1_{F(A)}$ and $\eta_A \circ \eta_A^{-1} = 1_{G(A)}$. Natural transformations also compose, and the composition of two natural transformations is also a natural transformation. Just as there are identity morphisms mapping objects to themselves, and identity functors mapping categories to themselves, there are also identity natural transformations, $\iota_F : F \rightarrow F$, mapping functors to themselves. And, so, the composition of a natural isomorphism (isomorphic natural transformation), $\eta : F \rightarrow G$, with its inverse, $\eta^{-1} : G \rightarrow F$, is an identity natural transformation, i.e., $\iota_F = \eta \circ \eta^{-1}$.

Functors preserve structure between categories; natural transformations identify the similarities between functors. For our purposes, functors construct new representations and processes from existing ones; natural transformations identify the similarities between constructions. A simple example that is closely related to the *List* functor example, illustrating this perspective, involves list reversal as indicated by the commutative diagram

$$\begin{tikzcd}
{[1,2,3]} \arrow[r, mapsto, "rev"] \arrow[d, "sqrl"'] & {[3,2,1]} \arrow[d, "sqrl"] \\
{[1,4,9]} \arrow[r, mapsto, "rev"'] & {[9,4,1]}
\end{tikzcd} \tag{6}$$

where the domain and codomain objects of each morphism are sets of lists, such as $\{1,2,3\}^*$; and *sqrl* is essentially *sqrlist* with (co)domain the set S^* instead of the monoid $(S^*,\cdot,\epsilon)$. As the diagram illustrates, squaring a reversed list is the same as reversing a squared list. So, there is a non-trivial (i.e. non-identity) relationship between the list monoid construction functor (*List*) and itself. The functor $Lst : \mathbf{Set} \to \mathbf{Set}$ constructing the lists in Diagram 6 is closely related to $List : \mathbf{Set} \to \mathbf{Mon}$ in that the returned object S^* is just the underlying set of the monoid $(S^*,\cdot,\epsilon)$, forgetting the binary operation $\cdot$ and the identity element. The underlying set can also be extracted by a functor from the category **Mon**, as we will see in the next section. This example shows how two ways of constructing individual lists, via the *Lst* functor, are related by the list reversal natural transformation, *rev*.

Although their associated diagrams look similar, there is an important difference between functor and natural transformation pertaining to the equality constraint that defines the relationships between object elements. For a functor, the equality constraint is local to the codomain of the transformation, i.e. the relationships between object elements within the constructed category. And so, the elements of the objects in the new category are only indirectly related to the elements in the corresponding objects of the source category by the categories' common external structure (i.e. inter-object relationships). For a natural transformation, the equality constraint spans the transformation, involving object elements mapped by both domain and codomain functors. And so, the two functors are directly related to each other by the internal structure of their associated objects (i.e. the relationships between object elements within an object). As part of a theory of cognitive architecture, there is a tension between the freedom afforded by functorial construction on the one hand—allowing an architecture to transcend the specific details of the source elements to realize a variety of possible representational schemes for those elements—and the need to pin down such possibilities to specific referents on the other. This tension is resolved with adjunctions.

Adjunction

An *adjunction* consists of a pair of functors $F : \mathbf{C} \to \mathbf{D}$, $G : \mathbf{D} \to \mathbf{C}$ and a natural transformation η: $1_{\mathbf{C}} \to (G \circ F)$, such that for every $\mathbf{C}-$object X and $\mathbf{C}-$map $f : X \to G(Y)$ there exists a *unique* $\mathbf{D}-$map $g : F(X) \to Y$, such that $f = G(g) \circ \eta_X$, indicated by the following commutative diagram:

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[dr, "f"'] & G \circ F(X) \arrow[d, "G(g)"] & F(X) \arrow[d, dashed, "g"] \\
 & G(Y) & Y
\end{tikzcd} \qquad (7)$$

where the functors are implicitly identified by (co)domain categories $\mathbf{C}$ (left subdiagram) and $\mathbf{D}$ (right subdiagram). The two functors are called an *adjoint pair*, (F,G), where F is the *left adjoint* of G, and G is the *right adjoint* of F; and natural transformation η is called the *unit* of the adjunction.

The left and right functors of an adjoint pair are like "inverses" of each other, but unlike an isomorphic functor whose composition with its inverse sends all objects and morphisms to themselves, the returned objects and their elements of a composition of left and right adjoints are related to the argument (source) objects and their elements by a natural transformation. For categories **Set** and **Mon**, the adjoint pair (F,U), consisting of functor $F : \mathbf{Set} \to \mathbf{Mon}$ that constructs the *free monoid* $F(S)$ on the set S, and then "forgetful" functor $U : \mathbf{Mon} \to \mathbf{Set}$ returns the underlying set S^* of monoid $F(S)$, are related by an injection. The injection is called an *insertion of generators*, whose component at S, $i_S : S \to S^*$, sends each element of S to the corresponding element (one-item list) in S^*. The elements $i_S(s)$ together generate the set S^* (i.e. S is the alphabet from which the set S^* of all "words" is constructed where each $s \in S$ is mapped to $[s] \in S^*$). In this context, $i_S : S \to S^*$ is the unit of this adjoint pair.

The effect of F on objects has just been given; the effect on morphisms is as follows: if $\alpha : S \to T$ is a function, then $\alpha^* : S^* \to T^*$ is defined as follows:

$$\alpha^*(\epsilon_S) = \epsilon_T$$

$$\alpha^*(s_1, \ldots, s_n) = \alpha(s_1), \ldots, \alpha(s_n), \quad \forall s_1, \ldots, s_n \in S$$

(cf. [25], p.111–112). Note that F is the functor *List* defined in the Functors section.

Monoid $F(S)$ is "free" in the informal sense that there are no missing or extra bits in the construction used to satisfy commutativity. The precise definition of *free* is as follows. Given the forgetful functor U, and an object S of **Set**, $F(S)$ is free on S if there is a morphism $i_S : S \to U \circ F(S)$ such that for any morphism $f : S \to U(M)$, there exists a unique morphism $g : F(S) \to M$ such that $f = U(g) \circ i_S$, indicated in the following commutative diagram:

$$\begin{tikzcd}
S \arrow[r, "i_S"] \arrow[dr, "f"'] & U \circ F(S) \arrow[d, "U(g)"] & F(S) \arrow[d, dashed, "g"] \\
 & U(M) & M
\end{tikzcd} \qquad (8)$$

However, not just any monoid generated from a set is a free monoid. For instance, the monoid $(\{0,1\},+,0)$ (i.e. addition modulo 2) in the diagram

$$\begin{tikzcd}
S = \{x, \ldots\} \arrow[r, "i_S"] \arrow[dr, "f : x \mapsto 1"'] & \{0,1\} \arrow[d, "U(g)"] & (\{0,1\},+,0) \arrow[d, "g"] \\
 & \mathbb{N} & (\mathbb{N},+,0)
\end{tikzcd} \qquad (9)$$

is not the free monoid on any set S, because the only homomorphism, $g : (\{0,1\},+,0) \to (\mathbb{N},+,0)$, maps 0 and 1 to $0 \in \mathbb{N}$, which does not make the diagram commute for $f : x \mapsto 1$. That is, $U(g) \circ i_S(x) = 0 \neq 1 = f(x)$. (It is easy to show that the free monoid on the empty set is $\{0\}$. So $\{0,1\}$ is not the free monoid on the empty set, either.) Other free objects, such as the *free group* on a set are defined analogously (see [21]). A simple example of a free monoid as may be employed by a cognitive system is a primitive form of counting, where $(\{1\}^*,\cdot,\epsilon)$ is the free monoid counter, having elements $\{\epsilon,[1],[1,1],[1,1,1],\ldots\}$, on singleton set $\{1\}$. This monoid is isomorphic to addition over the natural numbers, i.e. the monoid $(\mathbb{N},+,0)$.

From free objects we get an alternative (equivalent) definition of adjunction: consider functor $G : \mathbf{D} \to \mathbf{C}$ from the original definition. If for every object $X \in |\mathbf{C}|$, $F(X)$ is *free* on X with morphism η_X, then functor $F : \mathbf{C} \to \mathbf{D}$, with morphism mappings defined so that $G(F(f)) \circ \eta_X = \eta_Y \circ f$, is the left adjoint of G, and G is the right adjoint of F [31].

Yet another (equivalent) definition of adjunction, favoured by category theorists for its conceptual elegance, highlights the symmetry between a pair of adjoint functors: a bijection (one-to-one correspondence) between the set of morphisms from object $F(X)$ to Y in category $\mathbf{D}$ and the set of morphisms from object X to $G(Y)$ in category $\mathbf{C}$. So, identifying the unique morphism in one category means that it is associated with one and only one morphism in the other category.

In the list construction example, the unit of the adjunction is the injection $i : x \mapsto [x]$ sending each element x in the set S to the one-item list $[x]$ in the set of all lists S^* constructed from S, as shown in the following diagram:

$$\begin{tikzcd} S \arrow[r, "i"] \arrow[dr, "i\circ f"'] & S^* \arrow[d, "mapl(f)"] & (S^*,\cdot,\epsilon) \arrow[d, dashed, "maplist(f)"] \\ & T^* & (T^*,\cdot,\epsilon) \end{tikzcd} \tag{10}$$

where the left adjoint, $F : \mathbf{Set} \to \mathbf{Mon}$, constructs the *free* monoid $(S^*,\cdot,\epsilon)$ on the set S; and the right adjoint, $U : \mathbf{Mon} \to \mathbf{Set}$, returns the underlying set, S^*, of a list monoid, as mentioned earlier. In this way, given $i \circ f : x \mapsto [f(x)]$, the only homomorphism in the constructed category making the diagram commute is $maplist(f) : [x_1, \ldots, x_n] \mapsto [f(x_1), \ldots, f(x_n)]$. The definition for arrow $mapl(f)$ is essentially the same as $maplist(f)$, except that its (co)domain is a set, not a monoid. Other monoid homomorphisms that could have been chosen as part of the *List* functor definition, such as $maplist'(f) : [x_1, \ldots, x_n] \mapsto [f(x_1), f(x_1), \ldots, f(x_n), f(x_n)]$, are excluded by $i \circ f : x \mapsto [f(x)]$ and the commutativity property of the adjunction, because $mapl'(f) \circ i(x) = mapl'([x]) = [f(x), f(x)] \neq [f(x)] = i \circ f(x)$.

Since this arrangement works for any morphism in $\mathbf{Set}$, it can also be used to define a particular list length function from a family of analogous "length" functions as indicated in the following commutative diagram:

$$\begin{tikzcd} S \arrow[r, "i_S"] \arrow[dr, "1"'] & S^* \arrow[d, "len"] & (S^*,\cdot,\epsilon) \arrow[d, dashed, "length"] \\ & \mathbb{N} & (\mathbb{N},+,0) \end{tikzcd} \tag{11}$$

where monoid $(\mathbb{N},+,0)$ is the set of non-negative integers with addition as the operator and 0 as the identity element; $1 : S \to \mathbb{N}$ is a constant function sending every element to the number 1; and $len/length$ are functions returning the number of items in a list. As in the previous example, the definition of functor affords other choices for "length", such as $length' : L \mapsto 2 \times length(L)$, where L is a list. This arrow is also a monoid homomorphism, since $length'(L_1 \cdot L_2) = 2 \times (n_1 + n_2) = (2 \times n_1) + (2 \times n_2) = length'(L_1) + length'(L_2)$, where n_1 and n_2 are the lengths of lists L_1 and L_2, respectively. Again, however, the morphism $1 : S \to \mathbb{N}$ and the commutativity property force the usual choice for length function (i.e. len), and excludes others such as len', because $len' \circ i(x) = len'([x]) = 2 \neq 1 = 1(x)$.

A general pattern emerges from this use of adjunction. Functor construction may afford multiple choices for particular morphisms (processes) in the constructed category, but a principled choice is obtained through the commutativity property of the adjunction. This arrangement means that we are not committed *a priori* to a particular representational scheme; i.e., we do not have to make an *ad hoc* assumption about what that representational format should be. Given that an architecture has the capacity for an instance of the group of computations under consideration, then necessarily it applies to all other computations in that group. In the case of list length, for example, len' may indeed be the "correct" choice when we require the length of a list of characters in number of bytes for characters that are 2-byte unicodes (i.e. the characters appearing in the extended set that includes other special symbols and language scripts requiring two bytes for unique identification). So, to paraphrase, a computational architecture with the capacity to count the length (in bytes) of some lists of 2-byte unicodes necessarily has the capacity to compute byte lengths for all other unicode lists. In this way, the explanation for the "systematicity of list length" has two parts: existence is afforded by the possible list length functions; and uniqueness is afforded by the commutativity property of the adjunction. Without the adjunction, the choice of construction is by *ad hoc* assumption. Our explanation for the systematicity of human cognition follows this pattern.

Results

With these formal concepts in hand, we now proceed to our explanation of systematicity. We apply our explanation in two domains: systematicity with respect to relational propositions, and systematicity with respect to relational schemas. Then, we contrast our explanation with the Classical and Connectionist ones.

Systematicity of relational propositions: (*diagonal*, *product*) adjoint

For expository purposes, we develop our adjoint functors explanation from its components. One may wonder whether a simpler category theory construct would suffice to explain systematicity. For this example domain, the components of this adjoint have some systematicity properties, but in and of themselves do not explain systematicity—just as for Classicism and Connectionism, having a property is not the same as explaining it. This bottom-up approach motivates the more complex category theory construct from which the systematicity properties necessarily follow. Our approach has three steps. *First*, we show a categorical product that has the systematicity of representation and systematicity of inference properties. However, a product of two objects may afford many isomorphic product objects that do not also have the compositionality of representation property. *Second*, we show that the product functor provides the principled means for constructing only those products that also have the compositionality of representation property. There may, however, be several products that have the compositionality property, but which differ in semantic content by having different orders between identical sets of constituents. So, a principled choice is needed to determine *the* product. So, *third*, we show that the diagonal functor, which is left adjoint to the product functor, provides that principled choice by the commutativity property of the (diagonal, product) adjoint functor pair. For concreteness, we refer to the category $\mathbf{Set}$, but our explanation does not depend on this category.

(If we require an explanation of systematicity with respect to ternary relational propositions, then a ternary product $(A \times B \times C, p_1, p_2, p_3)$ is employed. The explanation for systematicity extends analogously, where the diagonal and product functors involve object triples. We may also need to explicitly represent a symbol for a relation, such as Loves. In this case, an object representing the relation symbol is paired with the product object representing the related entities. We address this situation in the next section. For present purposes, we omit relation symbols, since the relation is constant across the instances considered here and nothing essentially changes by its omission.

First, suppose objects A (say, agents) and B (patients) are sets containing representations of `John` and `Mary`, denoted as $\{\texttt{J, M}\}$. Although A and B are the same set of members, we maintain distinct names to keep track of the distinction between member pairs. (The assignment of elements to objects is itself an assumption, but not an *ad hoc* one for our theory, as explained in the next section and in the Discussion.) A categorical product of these two sets is the Cartesian product of A and B, which is the set of all pairwise combinations of elements from A and B, together with projections p_1 and p_2 for retrieving the first and second constituents in each case. That is, $A \times B = \{(\texttt{J, J}),(\texttt{J, M}),(\texttt{M, J}),(\texttt{M, M})\}$, $p_1 : (a,b) \mapsto a$, and $p_2 : (a,b) \mapsto b$. By definition, the Cartesian product $A \times B$ generates all pairwise combinations of elements from A and B, therefore this Cartesian product has the systematicity of representation property. Moreover, by definition, the categorical product $(A \times B, p_1, p_2)$ affords the retrieval of each constituent from each representation (otherwise it is not a product), therefore the categorical product also has the systematicity of inference property. In this case, Z from the categorical product definition takes the role of input, so in terms of Diagram 2 inferring `John` as the lover from `John loves Mary` is just $z_1(\texttt{JM}) = p_1 \circ u(\texttt{JM})$, where `JM` is the input and u is the input-to-product object map, whose unique existence is guaranteed by definition.

The Cartesian product, however, is not the only product object that satisfies the definition of a categorical product of A and B. An alternative product has $P = \{\texttt{1,2,3,4}\}$ as the product object, and $p'_1 : \texttt{1}{\mapsto}\texttt{J},\texttt{2}{\mapsto}\texttt{J},\texttt{3}{\mapsto}\texttt{M},\texttt{4}{\mapsto}\texttt{M}$ and $p'_2 : \texttt{1}{\mapsto}\texttt{J},\texttt{2}{\mapsto}\texttt{M},\texttt{3}{\mapsto}\texttt{J},\texttt{4}{\mapsto}\texttt{M}$ as the projections. Indeed, for this example, any four-item set together with the appropriate projections for retrieving the constituents would suffice. However, these alternatives do not have the compositionality of representation property: the semantic contents of these representations, whatever they may be, are not systematically related to each other, or the semantic content of `John`, or `Mary`. Hence, categorical products, in themselves, do not necessarily provide an explanation of systematicity.

Second, for any category $\mathbf{C}$ that has products (i.e. every pair of objects in $\mathbf{C}$ has a product), one can define a product functor $\Pi : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ (or, $\Pi : \mathbf{C} \times \mathbf{C} \times \mathbf{C} \to \mathbf{C}$, in the ternary case), that is from the Cartesian product of categories, $\mathbf{C} \times \mathbf{C}$, itself a category, to $\mathbf{C}$, where $\Pi_0 : (A,B) \mapsto A \times B$, $\Pi_1 : (f,g) \mapsto f \times g$, as indicated by the following diagram:

$$\begin{tikzcd}
(A,B) \arrow[r, mapsto, "\Pi_0"] \arrow[d, "{(f,g)}"'] & A \times B \arrow[d, "f \times g"] \\
(C,D) \arrow[r, mapsto, "\Pi_0"'] & C \times D
\end{tikzcd} \tag{12}$$

recalling that our functor diagrams explicitly identify the object component, Π_0, but not the morphism component, Π_1, of the functor. In this case, the semantic contents of these elements are systematically related to each other and their constituents `John` and `Mary`. This categorical construction is an instance of Classical compositionality, whereby the constituents $a_i \in A$, $b_j \in B$ are tokened wherever the compositions $(a_i,b_j) \in A \times B$ are tokened. As such, it has the compositionality of representation property.

Although the product functor has the compositionality of representation property, it introduces a different problem: $(B \times A, p'_2, p'_1)$, where $p'_2 : (b,a) \mapsto a$ and $p'_1 : (b,a) \mapsto b$ is also a valid product, but the semantic content of (a,b) is not the same as (b,a). That is because they have different order relationships between their constituents even though the corresponding constituents are identical. Thus, a principled choice is required to determine whether, for example, `John loves Mary` should map to (`John, Mary`), or (`Mary, John`). Otherwise, one can define an architecture that does not have the systematicity of inference property by employing both products to correctly infer `John`as the lover in `John loves Mary` via $(A \times B, p_1, p_2)$, yet incorrectly infer `John` as the lover in `Mary loves John` via $(B \times A, p'_2, p'_1)$, where position within the product triple identifies the relevant projection. The assumption that architectures employ only the first product is *ad hoc* just like the assumption that Classical architectures employ grammars such as G1, but not G2. So, a principled choice is needed to determine *the* product.

Third—final step, this problem brings us to the second aspect of our explanation foreshadowed in the Introduction (i.e. uniqueness). Again, as we saw with lists, a particular construction is specified through the left adjoint functor. The left adjoint to the product functor is the *diagonal* functor $\Delta : \mathbf{C} \to \mathbf{C} \times \mathbf{C}$ (or, $\Delta : \mathbf{C} \to \mathbf{C} \times \mathbf{C} \times \mathbf{C}$, in the ternary case), where $\Delta_0 : A \mapsto (A,A)$, $\Delta_1 : f \mapsto (f,f)$ as indicated by the following diagram:

$$\begin{tikzcd}
A \arrow[r, mapsto, "\Delta_0"] \arrow[d, "f"'] & (A,A) \arrow[d, "{(f,f)}"] \\
B \arrow[r, mapsto, "\Delta_0"'] & (B,B)
\end{tikzcd} \tag{13}$$

The (diagonal, product) adjoint pair is indicated by the following commutative diagram:

$$\begin{tikzcd}
C \arrow[r, "{\eta_C = \langle 1_C, 1_C \rangle}"] \arrow[dr, "{\langle s,t \rangle}"'] & C \times C \arrow[d, "s \times t"] & (C,C) \arrow[d, dashed, "{(s,t)}"] \\
 & M \times N & (M,N)
\end{tikzcd} \tag{14}$$

(see [28] Example 2.4.6). In this manner, the `John loves Mary` family of cognitive capacities is specified by the commutative diagram

$$\begin{tikzcd}
Pr \arrow[r, "{\langle 1_{Pr}, 1_{Pr} \rangle}"] \arrow[dr, "{\langle ag,pt \rangle}"'] & Pr \times Pr \arrow[d, "ag \times pt"] & (Pr,Pr) \arrow[d, dashed, "{(ag,pt)}"] \\
 & S \times S & (S,S)
\end{tikzcd} \tag{15}$$

where ag and pt are the *agent* and *patient* maps from the set of proposition inputs Pr into the set $S = A = B$ containing all the possible constituent representations. Here, we explicitly consider the case of equality, so that $A \times B = B \times A = S \times S$. When $A \neq B$, $ag \times pt$ and $pt \times ag$ have different codomains, since $A \times B \neq B \times A$, so the conflict between these products does not come into play, therefore the adjunction is not required and the product functor is sufficient. With the understanding that sets A and B are equal, we maintain the notational distinction for clarity in the subsequent text. Given $\langle ag,pt\rangle$ as the morphism used by the architecture to map proposition inputs to their corresponding internal representations, then the definition of an adjunction guarantees that (ag,pt) is unique with respect to making Diagram 15 commute via $ag\times pt$. That is, $(ag\times pt)\circ\langle 1_{Pr},1_{Pr}\rangle(\texttt{JM})=(ag\times pt)(\texttt{JM},\texttt{JM})=(\texttt{John},\texttt{Mary})=\langle ag,pt\rangle(\texttt{JM})$, where `JM` is the input for proposition `John loves Mary`. The alternative construction $pt\times ag$ is excluded because $(pt\times ag)\circ\langle 1_{Pr},1_{Pr}\rangle(\texttt{JM})=(pt\times ag)(\texttt{JM},\texttt{JM})=(\texttt{Mary},r\neq(\texttt{John},\texttt{Mary})=\langle ag,pt\rangle(\texttt{JM})$. Having excluded $pt\times ag$ by the commutativity property of the adjunction, the only two remaining ways to map the other inputs (i.e. $\langle ag,pt\rangle$ and $(ag\times pt)\circ\langle 1_{Pr},1_{Pr}\rangle$) are equal. So, given that the architecture can represent `John loves Mary` as (`John`, `Mary`) via $\langle ag,pt\rangle$ and infer `John` as the lover via p_1 from the product $(A\times B,p_1,p_2)$, then necessarily it can represent `Mary loves John` and infer `Mary` as the lover using the same morphisms. That is, $p_1\circ\langle ag,pt\rangle(\texttt{MJ})=p_1(\texttt{Mary},\texttt{John})=\texttt{Mary}$, or $p_1\circ(ag\times pt)\circ\langle 1_{Pr},1_{Pr}\rangle(\texttt{MJ})=p_1\circ(ag\times pt)(\texttt{MJ},\texttt{MJ})=p_1(\texttt{Mary},\texttt{John})=\texttt{Mary}$.

This explanation works regardless of whether proposition `John loves Mary` is represented as (`John`, `Mary`) via $\langle ag,pt\rangle$, or (`Mary`, `John`) via $\langle pt,ag\rangle$. In the latter case, the adjunction picks out just the construction (pt,ag), and hence $pt\times ag$, because it is the one and only one that makes the following diagram commute:

$$\begin{tikzcd}
Pr \arrow[r, "\langle 1_{Pr},1_{Pr}\rangle"] \arrow[dr, "\langle pt,ag\rangle"'] & Pr\times Pr \arrow[d, "pt\times ag"] & (Pr,Pr) \arrow[d, dashed, "(pt,ag)"] \\
 & S\times S & (S,S)
\end{tikzcd} \qquad (16)$$

That is, $(pt\times ag)\circ\langle 1_{Pr},1_{Pr}\rangle(\texttt{JM})=(pt\times ag)(\texttt{JM},\texttt{JM})=(\texttt{Mary},\texttt{John})=\langle pt,ag\rangle(\texttt{JM})$, but $(ag\times pt)\circ\langle 1_{Pr},1_{Pr}\rangle(\texttt{JM})=(ag\times pt)(\texttt{JM},\texttt{JM})=(\texttt{John},\texttt{Mary})\neq(\texttt{Mary},\texttt{John})=\langle pt,ag\rangle(\texttt{JM})$. Given that the architecture can represent `John loves Mary` as (`Mary`, `John`) via $\langle pt,ag\rangle$ and infer `John` as the lover via p'_2 from the product $(B\times A,p'_2,p'_1)$, then necessarily it can do so for `Mary loves John` using the same morphisms. That is, $p'_2\circ\langle pt,ag\rangle(\texttt{MJ})=p'_2(\texttt{John},\texttt{Mary})=\texttt{Mary}$, or $p'_2\circ(pt\times ag)\circ\langle 1_{Pr},1_{Pr}\rangle(\texttt{MJ})=p'_2\circ(pt\times ag)(\texttt{MJ},\texttt{MJ})=p'_2(\texttt{John},\texttt{Mary})=\texttt{Mary}$.

Explicit (multiple) relational propositions

If we need to explicitly represent a symbol for a relation, such as `Loves`, the product object is paired with an object, say R_L, representing the context in which the entities are related. The object representing the relation in this case is $(R_L,A\times B)$. This situation may arise where we need an explanation for systematicity that involves multiple similar relations, e.g., *loves*, *likes*, *dislikes*, and *hates*, where the capacity for instances of each of these relationships is co-extensive. That is, if one can represent `John loves Mary` *and* `John likes Mary`, then one can also represent the other six combinations, such as `Mary loves John` and `Mary likes John`. If one can represent `John loves Mary`, but not `John likes Mary`, then one can represent `Mary loves John`, but not `Mary likes John`. In this case, there is a category $\mathbf{R}$ of relation symbols whose objects, R_i, are symbols referring to each relation (e.g., *loves*, *likes*, etc.), and whose morphisms, 1_{R_i}, are just the identity morphisms for each object. (Such a category is called a *discrete* category.) Each relation, in this case, is a pair $(R_i,A\times B)$. Hence, the capacity to represent instances of the *loves* and *likes* relations extends to the other instances for both relations.

For these situations, the diagonal and product functors have extensions. The extension to the diagonal functor is: $\Delta^*:\mathbf{R}\times\mathbf{C}\to\mathbf{R}\times(\mathbf{C}\times\mathbf{C})$, such that $\Delta^*_0:(R_i,C)\mapsto(R_i,(C,C))$ and $\Delta^*_1:(1_{R_i},f)\mapsto(1_{R_i},(f,f))$. The product functor is: $\Pi^*:\mathbf{R}\times(\mathbf{C}\times\mathbf{C})\to\mathbf{R}\times\mathbf{C}$, such that $\Pi^*_0:(R_i,(C,D))\mapsto(R_i,C\times D)$ and $\Pi^*_1:(1_{R_i},(f,g))\mapsto(1_{R_i},f\times g)$. The adjunction, which is an extension of the one shown in Diagram 15, is shown in the following commutative diagram:

$$\begin{tikzcd}
(R_i,Pr) \arrow[r, "(1_{R_i}\text{,}\langle 1_{Pr},1_{Pr}\rangle)"] \arrow[dr, "(1_{R_i}\text{,}\langle ag,pt\rangle)"'] & (R_i,Pr\times Pr) \arrow[d, "(1_{R_i}\text{,}ag\times pt)"] & (R_i,(Pr,Pr)) \arrow[d, dashed, "(1_{R_i}\text{,}(ag,pt))"] \\
 & (R_i,S\times S) & (R_i,(S,S))
\end{tikzcd} \qquad (17)$$

In this situation, R_i provides the explicit context in which entities are related.

Under the assumption that these relation symbols belong to a different category, then cases such as `loves loves loves` cannot be generated. Note that supposing different objects for these entities is not an *ad hoc* assumption for our theory. $\mathbf{R}$ does not contain members such as `John` or `Mary`, and likewise A (or, B) does not contain relation symbols, because they refer to different types of entities with respect to the theory—`Loves` refers to a relation, which is at the level of objects in our theory, whereas `John` and `Mary` refer to entities in a relationship, which are members of objects.

Summary

In summary, products may have the systematicity of representation and inference properties (see also Discussion), but may not have the compositionality of representation property. Product functors construct products that have the compositionality property, but there may be more than one product with this property. The possible presence of multiple products requires a principled choice for fixing *the* product. That choice is provided by the (diagonal, product) adjoint functor pair. Importantly, the unit of the adjunction, $\langle 1_{Pr},1_{Pr}\rangle$, is not a free parameter of the explanation, it defines the specific adjunction in part; and there is no choice in representational format (i.e. left-right, or right-left constituent order)—the given capacity to represent a proposition fixes the same order for all the other propositions. The same situation also applies for the explicit (multiple) relational propositions domain. Hence, systematicity is a necessary consequence of this (extended) adjoint pair without recourse to *ad hoc* assumptions, and so meets the explanatory standard set by Aizawa [2], and Fodor and Pylyshyn [1], for this domain.

Systematicity of relational schemas: (*free*, *forgetful*) adjoint

Another domain in which humans exhibit systematicity is relational schema induction. This domain is more complex than the previous one in that the intrinsic connection is between relations, rather than within one. In the relational schema induction paradigm [32], participants are required to do cue-response prediction over a set of stimuli, such as letters and shapes, whose relationships conform to a group-like structure. For example, participants are shown (trigram, shape) pairs generated from a set of four trigrams (e.g., NEJ, POB, KEF, BEJ) and two shapes (e.g., square, circle), and are required to predict the response trigram, also from the same trigram set. Suppose, for example, a participant is presented with NEJ and square. After making a prediction, the correct response trigram is presented. This procedure is repeated with a new cue-response trial. The first two responses are not predictable prior to the feedback provided by the correct trigram. Hence, the first two trials are regarded as "information" trials. Each block of eight trials (i.e. all possible trigram-shape combinations) is repeatedly presented until a certain criterion level of correct performance is reached (e.g., correct responses to all eight trials in a block). Each set of eight cue-response pairs (i.e., four trigram times two shapes) constitutes a task instance. Once participants reach criterion a new task instance of eight cue-response pairs was randomly generated from a larger pool of possible trigrams and shapes (task instance examples are shown in Tables 1 and 2). The crucial data for this paradigm are the performances on subsequent task instances. When subsequent task instances conformed to the same structure, albeit with different stimuli, mean response error over the 48 participants was at or near optimal level: 2.00 errors per eight trials for the sequence of task instances conforming to the Klein group, and 2.67 for task instances conforming to the cyclic-4 group—two information trials are needed to determine the assignment of novel stimuli to structural elements [32]. The results provide another example of systematicity of human cognition: given that a person can correctly do one task instance and the information trials from the new task instance, then necessarily they can predict trials of all others, with the usual provision for a distinction between competence and performance.

This task is modelled as the category of *sets with actions*, **ASet** (cf. [25], 6.3.1, and [33] Definition 5.2), that has objects (Q,X,δ) for task instances, where Q is a set of states indicated by trigrams, X is a set of "actions" indicated by shapes, and $\delta : Q\times X\to Q$ specifies the action of a shape on a trigram resulting in a trigram. The morphisms $(g,\rho):(Q,X,\delta)\to(R,Y,\gamma)$ in this category consist of pairs of maps $g:Q\to R$ and $\rho : X\to Y$, such that the following diagram commutes:

$$\begin{tikzcd}
Q\times X \arrow[r, "\delta"] \arrow[d, "g\times\rho"'] & Q \arrow[d, "g"] \\
R\times Y \arrow[r, "\gamma"'] & R
\end{tikzcd} \qquad (18)$$

where the identity morphism $1_{Q,X,\delta}$ is the pair of identity maps $(1_Q,1_X)$, and compositions are defined component-wise. In our example, the set Q consists of four elements representing the four trigrams, and the set X consists of two elements representing the two shapes.

For the purpose of finding a suitable adjoint, we need to see how X is naturally embedded in a monoid. Recall that a monoid $(M,\cdot,\epsilon)$ consists of a set M and a binary associative operator $\cdot$ that satisfies closure: i.e., for all $a,b\in M, a\cdot b\in M$, whenever $a\cdot b$ is defined, and there is an identity element $\epsilon\in M$, such that $\epsilon\cdot m=m=m\cdot\epsilon$. In terms of our ASets (i.e. objects in **ASet**), the monoid identity corresponds to a "shape" whose action is to do nothing at all to the trigrams on which it acts: it leaves them unchanged. (However, this shape was not included in the experiments [32].)

Table 1. First task instance.

acts-on	NEJ	POB	KEF	BEJ
square	POB	NEJ	BEJ	KEF
circle	BEJ	KEF	POB	NEJ

doi:10.1371/journal.pcbi.1000858.t001

Table 2. Second task instance.

acts-on	GUD	QAD	JOQ	REZ
cross	QAD	GUD	REZ	JOQ
triangle	REZ	JOQ	QAD	GUD

doi:10.1371/journal.pcbi.1000858.t002

The adjoint functor pair used for this domain consists of the *forgetful* functor $U:\mathbf{ASet}\to\mathbf{Set}\times\mathbf{Set}$, which returns the underlying sets, i.e. $U_0:(Q,X,\delta)\mapsto(Q,X)$ and $U_1:(\rho,\sigma)\mapsto(\rho,\sigma)$, and its left adjoint, the *free* functor $F:\mathbf{Set}\times\mathbf{Set}\to\mathbf{ASet}$, which constructs ASets. The (free, forgetful) adjoint is shown in the following commutative diagram:

$$\begin{tikzcd}
(Q,X) \arrow[r, "\eta_{Q,X}"] \arrow[dr, "{(g,\rho)}"'] & (Q\times X^*,X) \arrow[d, "U(\psi)"] & (Q\times X^*,X,\mu) \arrow[d, dashed, "\psi"] \\
 & (R,Y) & (R,Y,\gamma)
\end{tikzcd} \qquad (19)$$

where $\eta_{Q,X}(q,x)=((q,\epsilon),x)$ and, for the instance of interest to us, (Q,X) and (R,Y) are the (trigram, shape) pairs of sets for the first and second tasks (respectively), as defined for example in Tables 1 and 2 so that $g:\mathtt{NEJ}\mapsto\mathtt{GUD}$, $\rho:\mathtt{square}\mapsto\mathtt{cross}$, etc. Full details and a proof that (F,U) is an adjoint functor pair are provided in Text S1.

Our explanation for systematicity in this domain follows the now familiar pattern, where monoids model the relationships between actions in each task instance. (Though our argument employs monoids, nothing essential changes if instead we use semigroups, or groups, where for example each task instance is extended with two additional shapes, one explicitly corresponding to the identity element, and the other to the remaining element in the Klein, or cyclic-4 group. For these cases, the proofs of adjointness can be extended to involve free semigroups and free groups, respectively.) Given an ASet modelling the first task instance and an ASet modelling the second task instance, there is more than one homomorphism from the first to the second, only some of which afford the correct responses to the stimuli in the second task instance. For example, one homomorphism has the following trigram and shape mappings: $g':\mathtt{NEJ}\mapsto\mathtt{GUD}$, $g':\mathtt{POB}\mapsto\mathtt{QAD}$, $g':\mathtt{KEF}\mapsto\mathtt{GUD}$, $g':\mathtt{BEJ}\mapsto\mathtt{QAD}$, $\rho':\mathtt{square}\mapsto\mathtt{cross}$, and $\rho':\mathtt{circle}\mapsto\mathtt{cross}$. Basically, the first table collapses to a table with one row and two columns. It is straight forward to check that it is indeed a homomorphism, for example, $g'\circ\mu_{Q,X}(\mathtt{NEJ},\mathtt{circle})=g'(\mathtt{BEJ})=\mathtt{QAD}=\mu_{R,Y}(\mathtt{GUD},\mathtt{cross})=\mu_{R,Y}\circ(g',\rho')(\mathtt{NEJ},\mathtt{circle})$. However, this homomorphism does not yield the correct responses to some of the stimuli in the second task instance. For example, all predictions to trigrams REZ and JOQ are no longer possible. Thus, a principled choice is required to select only those homomorphisms that indeed result in models for the second task instance. That choice is determined by (g,ρ) and the commutative property of the adjunction. That is, having obtained the first task instance, and given the two information trials of the second task instance that identify the correspondences between task stimuli, then there is one and only one homomorphism making the diagram commute, so that correct responses are obtained from the remaining trials of the second task instance. And so, systematicity is a necessary consequence of this adjunction.

Explanatory levels: *n*-category theory

Some readers may be interested in developing alternatives, or extensions to existing theories to address the systematicity problem in light of our explanation, so it is worth formally characterizing how our approach differs from previous ones. The difference between our category theory explanation and Classical/Connectionist approaches to systematicity may be characterized as higher-order versus first-order theories. Category theory also provides a formal basis for this distinction in terms of more general *n*-category theory (see, e.g., [34]). Though the concerns of *n*-category theorists go way beyond what we need here, some elementary aspects of the theory are used to formalize the difference between why our adjoint functors explanation addresses the systematicity problem and why the Classical or Connectionist approach does not.

Notice that the definitions of functor and natural transformation are very similar to the definition of a morphism. In fact, functors and natural transformations are morphisms at different levels of analysis: a natural transformation is a morphism one level above functors as we shall see. For *n*-category theory, a category such as **Set** is a 1-category, with 0-objects (i.e. sets) for objects and 1-morphisms (i.e. functions) for arrows. A functor is a morphism between categories. The category of categories, **Cat**, has categories for objects and functors for arrows. Thus, a functor is a 2-morphism between 1-objects (i.e. 1-categories) in a 2-category. A natural transformation is a morphism between functors. The functor category, **Fun**(**C**,**D**) of functors from **C** to **D**, has functors for objects and natural transformations for arrows. Thus, a natural transformation is a 3-morphism between 2-objects (i.e. functors) in a 3-category. (A 0-category is just a *discrete* category, where the only arrows are identities, which are 0-morphisms.) In this way, the order n of the category provides a formal notion of explanatory level.

Classical or Connectionist compositionality is essentially a lower-levels attempt to account for systematicity. For the examples we used, that level is perhaps best described in terms of a 1-category. Indeed, a context-free grammar defined by a graph is modelled as the *free category* on that graph containing sets of terminal and non-terminal symbols for objects and productions for morphisms [31]. By contrast, our category theory explanation involves higher levels of analysis, specifically functors and natural transformations, which live in 2-categories and 3-categories, respectively. Of course, one can also develop higher-order grammars that take as input or return as output other grammars. Similarly, one can develop higher-order networks that take as input or return as output other networks (e.g., networks whose connectivity is dynamic, such as cascade correlation [35]). However, the problem is that neither Classical nor Connectionist compositionality delineates those (higher-order) grammars or networks that have the systematicity property from those that do not. Likewise for our category theory explanation, not just any functor, nor just any natural transformation accounts for systematicity. If the explanation was left at either of these levels, then our approach would also succumb to the same problem that befalls Classicism and Connectionism—i.e. the problem of having to stipulate, *ad hoc*, just which functors or natural transformations account for the systematicity property. Rather, it is a natural transformation between an identity functor and a composition of two other functors ($F \circ G$) that defines the adjunction that accounts for systematicity relative to the particular domain of interest. In this formal sense, a crucial difference is that there is also a between-levels aspect to our explanation.

Discussion

Our adjoints explanation of systematicity has essentially two parts: (1) existence, showing how a particular connection between cognitive capacities is possible from a functorial specification of the architecture; and (2) uniqueness, explaining why that particular connection is necessary because it is the one and only one that satisfies the commutativity property of the adjunction. In contrast, the Classical and Connectionist explanations only provide an account of existence, but not uniqueness. That is, some grammars/networks afford the required intrinsic links between capacities and some do not, just like some functorial constructions do and some do not; but, for Classicism or Connectionism, there is no further explanation determining only those grammars or networks yielding systematicity (other than by *ad hoc* assumption), whereas for the category theory explanation the adjunction specifies only the systematic functors. So, our explanation meets the explanatory standard laid out by Aizawa.

To be regarded as a theoretical explanation for systematicity, such an explanation should be potentially falsifiable. Our explanation could be challenged by an alternative theory that accounts for systematicity (without *ad hoc* assumptions) in a way that does not require, or implement an adjunction. This possibility would not falsify our explanation as such, but may provide an alternative theory that is preferred on other grounds. Alternatively, there may exist a domain in which humans exhibit systematicity but for which there does not exist a relevant adjunction. Hence, the category theory approach we have put forward is in principle falsifiable.

The unit of an adjunction is a natural transformation between functors. The sense in which a transformation is natural is that the transformation does not depend on a particular "basis". A mathematician's example is to contrast the dual of a vector space with the, natural, double dual (dual of the dual) of a vector space—the former depends on a specific set of basis vectors chosen *ad hoc*, the latter does not. The analogue, here, is that our explanation of systematicity is natural in that it does not depend on a particular representational scheme (i.e., constituent order for relational propositions). Hence, the explanation does not depend on *ad hoc* assumptions about internal representations. Contrast this explanation with the Classical one, which must assume a particular grammatical form (e.g., G1 over G2) to fit the data.

In addition to explaining systematicity, our category theory approach has further implications. According to our explanation, systematicity with respect to binary relational propositions requires a category with products. A category theory account has also been provided for the strikingly similar profiles of development for a suite of reasoning abilities that included *Transitive Inference* and *Class Inclusion*, among others [30]—all abilities are acquired around the age of five years. The difference between the difficulties of younger children and the successes of older children (relative to age five) across all these reasoning tasks was explained as their capacity to compute (co)products. (A *coproduct* is related to a product by arrow reversal—see, e.g., [28] for a formal definition.) Therefore, our explanation implies that systematicity is not a property of younger children's cognition. Some support for this implication is found on memory tasks that require binding the background context of memorized items [36], though further work is needed to test this implication directly.

Our explanation for systematicity in regard to binary relational propositions does not depend on **Set**, it only requires a category with products. For example, the categories **Top** of topological spaces and continuous mappings, and **Vec** of vector spaces and linear mappings [21] could also be used. These possibilities imply that an explanation of systematicity does not depend on a particular (discrete symbolic, or continuous subsymbolic) representational format. Thus, a further benefit is that our approach opens the way for integration of other (sub/symbolic) levels of analysis.

Though some effort is needed to provide a category theory explanation for systematicity, even for a relatively simple domain such as relational propositions, the potential payoff is that our explanation generalizes to other domains where an appropriate adjunction is identified. This sort of tradeoff has been noted elsewhere in the context of a category theory treatment of automata [25]. We sketch one possibility in the domain of context-free grammars. Languages conforming to context-free grammars can be modelled as the *free category* on the directed graph that defines the grammar, whose vertices are sets of terminal and non-terminal symbols, and edges are transitions [31]. The left adjoint is the functor $F:\mathbf{Grph}\rightarrow\mathbf{Cat}$ from the category of directed graphs and graph homomorphisms to the category of categories and functors (category homomorphisms). The right adjoint is the forgetful functor $U:\mathbf{Cat}\rightarrow\mathbf{Grph}$, which returns the underlying graph (i.e. the arrows, forgetting their compositions). The explanation here is analogous to our explanation for relational schemas. The problem Aizawa raised with respect to Classicism is avoided here because systematicity is not derived from individual grammars, but homomorphic relationships between grammars.

Having provided an explanation of systematicity in terms of the rather abstract category theory concept of adjoint functors, one may wonder what this explanation means for a more typical conception of cognitive architecture in terms of internal representations and processes, and their realization in the brain. Human cognition is remarkable in that it affords the ability to think about things that have no sensory access (e.g., "a dog that is one lightyear long …"); yet reason about such entities as if they were grounded in our everyday experience ("… is smaller than a dog that is two lightyears long"). However, these two aspects must be reconciled: unbridled abstraction means that one can no longer determine what a particular internal representation is supposed to refer to; yet blinkering the system with over-narrowly defined representations curtails one's ability to *think outside the box*. These aspects appear in the form of functors and natural transformations in category theory. The adjunction is the category theory way of bringing them into precise "synchrony", or co-ordination, so that we may think abstractly about very specific things.

The realization of computational processes in the brain is classically conceived as a *physical instantiation mapping* from computational states to brain states, where the syntactic relationships between computational states correspond to physical relationships between brain states via such maps (see [1], p13). Category theory affords a similar, but more general and formal treatment in terms of functors. Diagrams of categories are formally defined as functors that map graphs (i.e. the shape of the diagram) to categories (see, e.g., [37]). Analogously, a categorial cognitive system would involve a functor from a categorial computational model to a brain system.

Up to this point, we have not considered the relatively new Bayesian approach to cognitive modelling (see, e.g., [38,39] for summaries) because, to our knowledge, a Bayesian explanation for systematicity has not yet been articulated. Nonetheless, the hierarchical Bayesian approach offers a significant advance with the ability to learn a diverse range of structures, such as lists, trees, and other (acyclic or cyclic) graphs, from data [40]. An important aspect of this approach is that structural form (or the type of structure) is encoded as prior beliefs by hyperparameters in the higher layers, and instances of those structures are encoded as parameters in the lower layers in so far as they conform to the constraints imposed by the data (environment). In this way, the architecture is not required to presume one particular structure to induce a group of behaviours from data. The hierarchical Bayesian approach affords the sort of higher-order theory that our analysis in the previous section implies. However, the question for the Bayesians is essentially the same as for the Classicists and Connectionists: that is, to articulate the Bayesian architectural principles from which systematicity necessarily follows. As the approach currently stands, systematicity depends on a number of factors including the available data, network connectivity, and optimization parameters. A Bayesian network with independently modifiable parameters for representing the distributions of constituents in each argument position of a relation may not have the systematicity property in the absence of data with, say, `Mary` in the patient position (so called *strong systematicity* [18]), simply because there may be no (prior) information available to determine the value of the associated parameters. Hyperparameters may enable a dependency between lower level parameters so that the acquisition of one entails the acquisition of another. Still, systematicity may not necessarily follow from hyperparameters alone: for example, one can envisage a network where partial hyperparametrization links some but not all behaviours within the group, analogous to the problem that was raised with respect to classical compositionality.

All theories make certain assumptions. The question is whether those assumptions are extrinsic to the theory and carry the essential explanatory burden (i.e. they are *ad hoc*). In our case, one may question whether supposing that an object contains representations of `John` and `Mary` is not itself an *ad hoc* assumption, for the Cartesian product does not necessarily represent all possible combinations of mental representations [41] (e.g., $\{\mathtt{J},\mathtt{M}\}\times\{\mathtt{M}\}$ generates representations corresponding to `John loves Mary` and `Mary loves Mary`, but not `John loves John`). Our explanation for systematicity of binary relational propositions is a consequence of the (diagonal, product) adjoint (Diagram 15), not a specific categorical product. Though the categorical product is a component of the explanation, the particular product is derived from the adjunction, not chosen independently of it. Where the constituent entities are of the same sort, and so belong to the same object (S) in our theory, the diagonal functor generates the object pair (S,S), and the product functor takes (S,S) and generates the product object $S\times S$, hence cases like $\{\mathtt{J},\mathtt{M}\}\times\{\mathtt{M}\}$ cannot occur in this formulation. The assumption that relation symbols belong to a different category than the related arguments precludes the generation of intrinsically unconnected cases, such as `loves loves loves`. Typing, in this sense, shares some of the explanatory burden, but types are not extrinsic to our theory. An element cannot exist without belonging to an object (its type) in a category, by definition. Hence, types are intrinsic to the theory. Moreover, the explanatory burden is also born by the adjunction in our example domains. Even with typing, there must still be a principled choice for the order of those constituents, when they involve the same objects, which is provided by the adjunction. And, given that adjunctions are central to category theory, neither the assumption of types, nor our use of adjunction can be regarded as *ad hoc* for the purpose of explaining systematicity in these domains. Classicism also makes a distinction between atomic and molecular representations, as a core assumption [1]. However, even under core assumptions that are equivalent to ours—`John` and `Mary` belong to the same word classes, which differ from `loves`—systematicity does not necessarily follow, as exemplified by grammar G2. Hence, the critical difference between our explanation of systematicity and the Classical approach is the adjunction.

This assumption of typing, though, is acute for quasi-systematic domains, where cognitive capacity may extend to some but not all possible constituent combinations, which appear to be particularly prevalent in language (see [41]). For these cases, we would also need category theory-derived principled restrictions to products. *Equalizers* and *pullbacks* (see [30] for an application to cognitive development) are two ways to restrict (product) objects, in the same arrow-theoretic style. Products, pullbacks and equalizers are all instances of the general, formal concept of a *limit* in category theory. The existence of adjoint functors is closely linked to the existence of limits in the respective categories (cf. *adjoint functor theorems* [21], p210–214), which suggests that an appropriate adjunction can also be found for domains that require an explanation for quasi-systematicity.

Needless to say, our category theory explanation is not the final word on a theory of cognitive architecture. For our approach (and Classicism), where the assignment of elements to objects (and, words to word classes) is asserted, there is also the broader question of why they get assigned in a particular way. This question pertains to the acquisition of representations, whereas the systematicity problem pertains to their intrinsic connections. Incorporating category theory into the Bayesian approach may provide a more integrative theory in this regard. A connection between category theory and probability has been known for some time (see [42]), and category theory concepts have been incorporated into the development of probabilistic functional programming [43]. A potentially fruitful line of future research, then, may be to identify a suitable adjunction with respect to, say, a category of Bayesian models, if such a category exists.

From a category theory perspective, we now see why cognitive science lacked a satisfactory explanation for systematicity—cognitive scientists were working with lower-order theories in attempting to explain an essentially higher-order property. Category theory offers a re-conceptualization for cognitive science, analogous to the one that Copernicus provided for astronomy, where representational states are no longer the center of the cognitive universe—replaced by the relationships between the maps that transform them.

Supporting Information

Text S1 Proof that the free and forgetful functors for the category ASet form an adjoint functor pair.
Found at: doi:10.1371/journal.pcbi.1000858.s001 (0.10 MB PDF)

Acknowledgments

We thank the reviewers for comments that have helped improve the exposition of this work.

Author Contributions

Analyzed the data: SP WHW. Contributed reagents/materials/analysis tools: SP WHW. Wrote the paper: SP WHW.

References

1. Fodor JA, Pylyshyn ZW (1988) Connectionism and cognitive architecture: A critical analysis. Cognition 28: 3–71.
2. Aizawa K (2003) The systematicity arguments. Studies in Mind and Brain. New York: Kluwer Academic.
3. Fodor JA, McLaughlin BP (1990) Connectionism and the problem of systematicity: Why Smolensky's solution doesn't work. Cognition 35: 183–204.
4. Fodor JA (1997) Connectionism and the problem of systematicity (continued): Why Smolensky's solution still doesn't work. Cognition 63: 109–119.
5. van Gelder T (1990) Compositionality: A connectionist variation on a classical theme. Cogn Sci 14: 355–384.
6. Smolensky P (1987) The constituent structure of connectionist mental states: A reply to Fodor and Pylyshyn. Southern J Philos 26: 137–161.
7. Smolensky P (1991) Connectionism, constituency, and the language of thought. In: Loewer B, Rey G, eds. Meaning in Mind: Fodor and his critics. CambridgeMA: Blackwells, chapter 12.
8. Aizawa K (2003) Cognitive architecture: The structure of cognitive representations. In: Stich SP, Warfield TA, eds. The Blackwell guide to philosophy of mind. CambridgeMA: Blackwell, chapter 7. pp 172–189.
9. Smolensky P (1990) Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artif Intell 46: 159–216.
10. Phillips S (2007) Kenneth Aizawa, The systematicity arguments, Studies in brain and mind. Mind Mach 17: 357–360.
11. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagation of error. Nature 323: 533–536.
12. Elman JL (1990) Finding structure in time. Cogn Sci 14: 179–211.
13. Marcus GF (1998) Rethinking eliminative connectionism. Cogn Psychol 37: 243–282.
14. Marcus GF (1998) Can connectionism save constructivism? Cognition 66: 153–182.
15. Phillips S (1998) Are feedforward and recurrent networks systematic? analysis and implications for a connectionist cognitive architecture. Connect Sci 10: 137–160.
16. Phillips S (1999) Systematic minds, unsystematic models: Learning transfer in humans and networks. Mind Mach 9: 383–398.
17. Phillips S (2000) Constituent similarity and systematicity: The limits of first-order connectionism. Connect Sci 12: 1–19.
18. Hadley RF (1994) Systematicity in connectionist language learning. Mind Lang 9: 247–272.
19. Niklasson L, van Gelder T (1994) Systematicity and connectionist language learning. Mind Lang 9: 288–302.
20. Fodor JA (1987) Psychosemantics: The problem of meaning in the philosophy of mind. Explorations in cognitive science. Cambridge, MA: MIT Press.
21. Awodey S (2006) Category theory. Oxford Logic Guides. New York, NY: Oxford University Press.
22. Lawvere FW, Schanuel SH (1997) Conceptual mathematics: A first introduction to categories. Foundations of Computing. Cambridge, UK: Cambridge University Press.
23. Mac Lane S (2000) Categories for the working mathematician. Graduate Texts in Mathematics. New York, NY: Springer, 2nd edition.
24. Eilenberg S, Mac Lane S (1945) General theory of natural equivalences. Trans Amer Math Soc 58: 231–294.
25. Arbib MA, Manes EG (1975) Arrows, structures, and functors: The categorical imperative. London, UK: Academic Press.
26. Barr M, Wells C (1990) Category theory for computing science. Prentice Hall International Series in Computer Science. New York: Prentice Hall, first edition.
27. Goguen J (1991) A categorical manifesto. Math Structures Comput Sci 1: 49–67.
28. Pierce BC (1991) Basic category theory for computer scientists. Foundations of Computing. Cambridge, UK: MIT Press.
29. Halford GS, Wilson WH (1980) A category theory approach to cognitive development. Cogn Psychol 12: 356–411.
30. Phillips S, Wilson WH, Halford GS (2009) What do Transitive Inference and Class Inclusion have in common? Categorical (co)products and cognitive development. PLoS Comput Biol 5: e1000599.
31. Walters RFC (1991) Categories and computer science. Cambridge Computer Science Texts. Cambridge, UK: Cambridge University Press.
32. Halford GS, Bain JD, Maybery MT, Andrews G (1998) Induction of relational schemas: Common processes in reasoning and complex learning. Cogn Psychol 35: 201–245.
33. Wilson WH (1979) On induced representations of Lie algebras, groups, and coalgebras. J Algebra 58: 37–50.
34. Leinster T (2003) Higher operads, higher categories. London Mathematical Society Lecture Notes Series. Cambridge, UK: Cambridge University Press.
35. Fahlman SE, Lebiere C (1990) The cascade-correlation learning algorithm. Technical Report CMU-CS-90-100, Carnegie Mellon University.
36. Lloyd ME, Doydum AO, Newcombe NS (2009) Memory binding in early childhood: evidence for a retrieval deficit. Child Dev 80: 1321–1328.
37. Crole RL (1993) Categories for types. New York, NY: Cambridge University Press.
38. Griffiths TL, Kemp C, Tenenbaum JB (2008) Bayesian models of cognition. In: Sun R, ed. Cambridge Handbook of Computational Cognitive Modeling. New York, NY: Cambridge University Press, chapter 3. pp 59–100.

39. Tenenbaum JB, Griffiths TL, Kemp C (2006) Theory-based Bayesian models of inductive learning and reasoning. Trend Cogn Sci 10: 309–318.
40. Kemp C, Tenenbaum JB (2008) The discovery of structural form. PNAS 105: 10687–10692.
41. Johnson K (2004) On the systematicity of language and thought. J Philos 101: 111–139.
42. Giry M (1981) A categorical approach to probability theory. In: Banaschewski-Bernhard, ed. Categorical aspects of topology and analysis Springer-Verlag, volume 915 of Lecture Notes in Mathematics. pp 65–85.
43. Erwig M, Kollmansberger S (2006) Probabilistic functional programming in Haskell. J Funct Program 16: 21–34.