Axiomathes (2006) 16:165–214
DOI 10.1007/s10516-005-5474-1
© Springer 2006

MICHAEL JOHN HEALY and THOMAS PRESTON CAUDELL

ONTOLOGIES AND WORLDS IN CATEGORY THEORY: IMPLICATIONS FOR NEURAL SYSTEMS

ABSTRACT. We propose category theory, the mathematical theory of structure, as a vehicle for defining ontologies in an unambiguous language with analytical and constructive features. Specifically, we apply categorical logic and model theory, based upon viewing an ontology as a sub-category of a category of theories expressed in a formal logic. In addition to providing mathematical rigor, this approach has several advantages. It allows the incremental analysis of ontologies by basing them in an interconnected hierarchy of theories, with an operation on the hierarchy that expresses the formation of complex theories from simple theories that express first principles. Another operation forms abstractions expressing the shared concepts in an array of theories. The use of categorical model theory makes possible the incremental analysis of possible worlds, or instances, for the theories, and the mapping of instances of a theory to instances of its more abstract parts. We describe the theoretical approach by applying it to the semantics of neural networks.

KEY WORDS: category, cognition, colimit, functor, limit, natural transformation, neural network, semantics

1. INTRODUCTION

1.1. *Neural network semantics and knowledge*

Category theory is the branch of mathematics concerned with pure structure. In recent work (Williamson et al., 2001), it has been applied to the formalization of the underlying structure of knowledge organized into ontologies for computational systems. The goal has been twofold: To produce software from a formalized knowledge representation expressing its intended semantics, and to unify knowledge-based systems developed by a multitude of different organizations and written in a variety of languages. Making programming semantics visible in an unambiguous, enforceable manner within the software development process offers a potential solution for the problems of spiraling cost and implanted logical errors haunting software development and maintenance efforts (Jullig and

Srinivas 1993; Srinivas and Julig 1995), (Williamson and Healy 1997, 2000; Healy and Williamson 2000). Another, more recent thread is perhaps closer to the spirit of the original development of category theory and its application in mathematical semantics: The exploration of ontologies for comprehensive, unambiguous, system-language-neutral knowledge representation and use in science, engineering, finance, and other areas. Our present work takes the latter thread into a new area: The incremental acquisition and representation of ontologies by adaptive neural networks. Here, we show how category theory can be applied to ontologies for understanding the many possible worlds that are internalized implicitly in the computations of adaptive neural networks.

From the earliest writings on the subject (McCullough and Pitts 1943), investigators in logic and the semantics of computation have sought an accurate understanding of the implicit meaning of neural computations; see, for example, (Arbib 1987; Fu 1992; Craven and Shavlik 1993; Bartfai 1994; Carpenter and Tan 1995; Mitra and Pal 1995; Pinkas 1995; Sima 1995; Kasabov 1996; Healy and Caudell 1997; Heileman et al., 1997; Healy 1999a), and the survey in (Andrews et al., 1995). Presumably, when a well-designed network adapts its connection weights, it is effectively modifying its responses to input stimuli to improve its future response according to some criterion. Information gained from the input data becomes internalized in the form of connection weight modifications, which affect the future response of the network to its inputs. What is this information; is it possible to understand it as knowledge expressible in a human-understandable form? This question is often addressed by attempting to decode the adapted connection-weight values as logical rules that the network is supposed to have learned from its input stream. Formal-logic-like languages are sometimes used for this, to allow declarative statements such as IF-THEN rules to be expressed without ambiguity and sometimes with full mathematical rigor (Arbib 1987; Pinkas 1995; Healy 1999a). Intuitively, the ability of any computational system to manipulate data in a systematic way is a manifestation of the knowledge represented in the system's design and in its store of already-processed data. The use of a mathematical language to understand the knowledge content of a computational system is called *mathematical semantics* (Pierce 1991).

Cognitive neuroscientists seek to understand the relationship between structure and function in the brain – the semantics of neuron/synapse organization. One of their significant findings is that neurons and their synaptic connections are organized on a larger scale into a system of interconnected functional regions (Damasio 1989; Hirsch et al., 2001). Each region is associated with one or more sensory modalities, motor control, planning, and the control of working memory (Squire and Zola-Morgan 1991; Otto et al., 1992; Rao et al., 1997; Wickelgren 1997), and a region implicated in self-referential processing has been tentatively identified in humans (Kelley et al., 2002). One way to regard this is to assign to each region a system of knowledge that appears to describe the functionalities with which it has been associated. For example, the recent "what/where" model of the visual pathway from the retinae to other areas of the brain asserts that the pathway bifurcates (Otto et al., 1992). The spatial layout of objects in the visual field appears to be extracted in successive processing stages along a pathway from the occipital to the posterior parietal lobe in the cerebral cortex (the dorsal path). In parallel with this, a pathway from the occipital lobe through the temporal lobe (the ventral path) appears to form semantic object representations in successive stages. The representations begin as simple sensory features and eventually reach the complexity of scenes and events near the juncture of parietal, temporal and occipital lobes, where multi-sensor representations of scenes and events appear to be formed. Connection pathways between regions help organize the more complex object, scene, and event representations among the modalities. Apparently, they do this by re-uniting the spatial and semantic information, now broken down into iconic representations manipulable in a flexible working memory system. The working memory system is a set of processes involving functional regions and their interconnections that organize, store and recall information from synaptic memory that is associated with current experience.

This description of the findings of cognitive neuroscience suggests a view of the brain as a knowledge-manipulation system that acquires information and forms separate representations of it, beginning with sensor-related knowledge such as visual form, auditory form, and spatial location. The storage process is more than a simple filing-away of data; the flexible use of data, involving creativity in many organisms, suggests that what is stored is an "internal model"

of the world. This model is many-faceted, capable of representing a wide variety of situations which can be associated with simultaneous inputs from several sensors. How does this multi-faceted view of knowledge representation in the brain relate to the expression through the organism's behavior of a single, unified system of knowledge? After all, an organism does not jump between visual, auditory and other knowledge representations, applying them one at a time in a disconnected fashion, for such incoherence would lead to disaster. Our mathematical model explains the interactions of the regional knowledge representations in a multi-regional network through interconnection pathways as a unifying of representations. In this unification, the whole network acts as if there were a single knowledge system guiding its behavior, a key property which we call *knowledge coherence*.

Category theory, the branch of mathematics upon which our semantic theory is based, has seen increasing use in computational semantic modeling and its applications (Meseguer 1989; Pierce 1991; Tse 1991; Goguen and Burstall 1992; Crole 1993; Jullig and Srinivas 1993; Vickers 1993; Stoltenberg-Hansen et al., 1994; Williamson et al., 2001). We have recently become aware of other applications of category theory to biological systems and their semantics (Rosen 1958a, b; Ehrevan and Vanbremeersch 1997; Gust and Kühnberger 2005). There are similarities in approach but important differences as well (see our Section 6): Ehrevan and Vanbremeerschs' limits and colimits relate loosely to ours, but our neural categories have different properties and are used differently. Gust and Kühnbergers' use of topos theory applies the notion of deduction directly to neural systems, in a somewhat similar fashion to the topological model of (Healy 1999a); here, we apply instead a more general notion of computation (or not – see our comment on Rosen) to the semantics of neural systems. Finally, it is possible that Rosen's notions of metabolism, repair and replication are emodied in our architecture transition function (called Φ_A in Section 6 for an architecture A), but more review of both approaches is needed; also, our bias is more toward the view that the brain does computation. Previous papers (Healy 1999b, 2000; Healy and Caudell 2001, 2002, 2003) have presented much of the present semantic theory in a preliminary form, and (Healy and Caudell 2004) contains a full exposition.

Section 2 of the present paper introduces a new mathematical framework for formalizing ontologies as distributed, hierarchical systems of knowledge, based upon category theory. Section 3 introduces categorical constructs used here for modeling long-term adaptation, or learning. The learning re-uses existing knowledge to form both more specialized and more abstract concepts to represent new data. Section 4 is a brief introduction to functors and natural transformations, the fundamental structure-preserving mappings of category theory. Section 5 introduces model theory, which brings mathematical rigor to the task of analyzing the "possible worlds" associated with a theory. Section 6 introduces categories that formalize neural computation. Section 7 builds on the previous sections with a discussion of neural network design principles for ontological representation based upon limits and colimits, model theory, and functors. Section 8 discusses additional design principles associated with knowledge coherence expressed through natural transformations, explicating information fusion across multiple sensor modalities in multi-region neural networks. Section 9 is the Conclusion.

2. KNOWLEDGE REPRESENTATION AS MATHEMATICAL STRUCTURE

2.1. *Encoding and using ontologies*

A mathematical model of concepts, ontologies, and knowledge representation in computational systems (including biological ones) can provide a useful adjunct to studies in computer science, neuroscience and other fields. It provides a formal language that disambiguates the expressed notions about systems, allowing analysts to achieve a common understanding of their functionality or meaning, their components and their computations. It has the potential to enable proof that a system does or does not have a proposed property, such as avoiding undesirable behaviors. It enables a designer to express desired system properties with logical precision and then synthesize a system accordingly. Establishing such a model requires a form of mathematics appropriate for both a human-readable encoding of knowledge and the expression of the knowledge in the memory store and/or the computations of a system. The class of systems discussed here poses a particular challenge in knowledge representation: Adaptive neural systems, both artificial and natural,

are computational systems that implement both "hard-wired" (designed in) and learned knowledge. We describe the knowledge as a category representing an ontology formalized in a modular hierarchy of closed knowledge fragments, or theories. We call the theories concepts, and in the hierarchy they are organized from the abstract to the specific. They are represented incrementally in a neural network as it learns through Hebbian-like (Hebb 1949) synaptic potentiation (and possibly de-potentiation as well). The relationships between concepts that organize them into a hierarchy are learned along with the concepts; they express the inheritance by the more specific concepts of the properties expressed in the more abstract concepts. This organization is meant to capture the incremental representation of an ontology in a system, especially one that is distributed over many interacting components: Beginning with certain, primitive concepts and relations representing initial knowledge and the semantics of input stimuli, specialization and abstraction processes extract information from input stimuli to form representations of new concepts and their relationships. The ontology is thereby "filled out" in both directions, specialization and abstraction, from prior knowledge and new data. This model of the learning process exposes the nature of the special challenge for the architectural design and operation of an adaptive system that is to incrementally acquire and represent an ontology. We express the challenge through constraints, or design principles, derived from our mathematical model. These can be used to analyze existing neural architectures, both biological and artificial, and can also guide the design of new artificial neural architectures.

An existing computerized system for knowledge sharing (Gruber and Olsen 1994) uses a declarative language resembling a first-order predicate calculus to express ontologies. An ontology written in the language serves as a basis for communication among knowledge-based systems or intelligent agents that perform tasks specified by a system of users (clients). Accompanying translators provide for the knowledge systems or agents to use different symbolic languages for their individual computations. This allows each agent to exploit the relative advantages of a separate knowledge representation language suitable to its intended function, and in any case is a matter of practicality given the many languages that are in use. Using the common ontology through a translator, each system has access to a version of all or part of the ontology translated into its own language,

enabling knowledge sharing and avoiding a "tower of babel" effect in multi-agent interaction.

The mathematical model for ontologies and computational systems proposed here is based in category theory. It has some similarities to the computerized ontology system just mentioned, but has capabilities far beyond the encoding of knowledge in logical statements. Expressing an ontology as a mathematical category of theories allows the machinery of category theory to be harnessed in understanding the semantics of adaptive, distributed systems. Distributed systems that implement an ontology are expressed as categories consisting of interrelated system components. The mathematical model exposes the knowledge associated with a system's operation as an embedding of the ontology category in the system category, associating with each system component the knowledge associated with the function it serves within the system at the level of abstraction or specialization at which it functions. Finally, the mathematical model provides for transformations between knowledge embeddings, or representations, to relate the semantics of the regions in a multi-regional system organization such as the brain.

2.2. *Categories*

Writing an ontology in formal logic can be cumbersome and subject to error if the statements form a single, monolithic theory of some universe of discourse. Instead, an ontology can be modularized as a collection of interrelated theories, where the relationships relate the more abstract theories to the more specific ones that share their properties, as mentioned in the last section. The relationships between theories are as important as the theories themselves, for they give the ontology its internal coherence. The encoding of an ontology in category theory (Mac Lane 1971; Adamek et al., 1990; Pierce 1991; Crole 1993; Lawvere and Schanuel 1997) addresses this fact, for it has the form of a category of theories. In a category, the relationships or *arrows* or *morphisms* represent the structural imperative, the *reason d' etre* that enables a category to represent a collection of things with a common organizing principle. Set theory is an alternative foundational theory of mathematics that is also used in expressing the semantics of formal logic. It is based upon the notion of the membership of a quantity x in a collection y, or $x \in y$. In category theory, by contrast, a morphism $f : a \longrightarrow b$ in a category

C has a *domain* a and a *codomain* b which are both *objects* of C. The morphism expresses one of the possibly many ways in which a relates to b in the context of C. This will become clearer in the examples we shall discuss.

It is apparent that the morphisms give a category an underlying directed-graph structure in which objects serve in the role of nodes and morphisms act as edges. It is important to realize that a category is not a graph. First, there can be many morphisms in either or both directions between a pair of objects a and b, hence the need for morphism labels such as f, g, h,... (as opposed to "edge (a, b)" in a graph) and the terminology "domain" and "codomain". The most important distinguishing feature of a category, however, is the notion of *composition*. In a category C, each pair of arrows having the form $f: a \longrightarrow b$ and $g: b \longrightarrow c$ (with a head-to-tail match, where the codomain b of f is also the domain of g as indicated) has a composition arrow $g \circ f : a \longrightarrow c$ whose domain a is the domain of f and whose codomain c is the codomain of g. Composition satisfies two axioms of category theory. The associative law states that in triples which have a head-to-tail match by pairs, $f: a \longrightarrow b$, $g: b \longrightarrow c$ and $h: c \longrightarrow d$, the result of composition is order-independent, $h \circ (g \circ f) = (h \circ g) \circ f$. The identity axiom states that for each object a there is an *identity morphism* $\mathtt{id}_a : a \longrightarrow a$ such that for any arrows $f: a \longrightarrow b$ and $g: c \longrightarrow a$ $\mathtt{id}_a \circ g = g$ and $f \circ \mathtt{id}_a = f$. The notion of composition derives its importance from the fact that many pathways through the morphisms leading from one object to another in a category often yield the same morphism when the composition of the morphisms is calculated along each path. This means that, unlike the situation with graphs, a path through the arrows in a category is associated with a precise notion of cumulative effect or meaning; and, further, different paths whose compositions have the same domain and codomain can have the *same* meaning. When this is true, we have a *commutative diagram* as shown in Figure 1. The category in which this diagram is formed is $\mathbf{N}_|$; its objects are positive natural numbers, 1, 2, 3,..., and a morphism from a to b, denoted $|_{a,b}$, exists exactly when a is a divisor of b. There are two apparent morphisms with domain 2 and codomain 24 in the diagram, both being compositions along a path directed through a third diagram object (4 and 6, respectively). Yet, there being at most one divisibility morphism from one natural number to

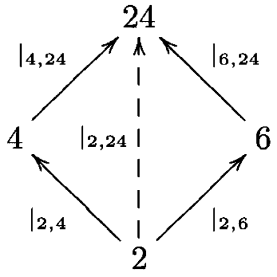


Figure 1. One of many commutative diagrams in the category $\mathbf{N}_{|}$.

another, we have the equation $|_{4,24} \circ |_{2,4} = |_{2,24} = |_{6,24} \circ |_{2,6}$, stating that the two are equal. More generally, a commutative diagram in any category has the property that any two morphisms having the same diagram objects as domain and codomain, where at least one of them is obtained as the composition of two or more diagram morphisms and the other is obtained in the same fashion or is itself a diagram morphism, are equal.

The category $\mathbf{N}_{|}$ provides an example in which the morphisms are instances of what we usually think of as a relation – the divisibility relation defined on positive natural numbers. It also exemplifies a relatively simple type of mathematical structure known as a partial order, having at most one morphism between a pair of objects, all morphisms having a shared sense of direction (that is, there are no cycles). In **Set**, which is probably the most familiar example of a category, the morphisms are functions. There are many morphisms in both directions for most pairs of objects in this category and, in particular, there are many cycles, so **Set** is definitely not a partial order. Composition in **Set** is just the familiar composition of functions, with $(g \circ f)(x) = g(f(x))$ for functions $f: a \longrightarrow b$ and $g: b \longrightarrow c$, with $x \in a$ and $(g \circ f)(x) \in c$. Function composition is associative, and for any set a there is an identity function $\mathtt{id}_a$ whose values are $\mathtt{id}_a(x) = x(x \in a)$, so **Set** is indeed a category. Commutative diagrams in **Set** can be used to express the equivalence of functions, where one of them is obtained as the composition of two or more others.

One sometimes hears a statement such as "the two [concepts, data types, program constructs, etc.] are in some sense isomorphic". Is there a precise meaning for this term? Indeed, category theory provides a mathematically rigorous notion of "isomorphism": If a, b are objects of a category C such that there

exist arrows $f: a \longrightarrow b$ and $g: b \longrightarrow a$ with $f \circ g = \mathtt{id}_b$ and $g \circ f = \mathtt{id}_a$, then the morphism f is called an *isomorphism* (as is g also) and g is called its *inverse* (and f is called the inverse of g), and the two objects are said to be isomorphic. The property of an identity morphism ensures that isomorphic objects in a category are interchangeable in the sense that they have the same relationships with all objects of the category. An isomorphism in **Set** is a one-to-one, onto function.

An *initial object* of a category C is an object i having a unique morphism $f: i \longrightarrow a$ corresponding to every object a of C. A *terminal object* t is the dual notion, obtained by reversing arrows in the definition of i – that is, it serves as the codomain of a unique morphism $f: a \longrightarrow t$ corresponding to every object a of C. It is easy to show that all initial objects in a category are isomorphic, and ditto for terminal objects. For example, suppose that i, i' are initial in C. Then, applying initiality to each object, there must be unique morphisms $f: i \longrightarrow i'$ and $f': i' \longrightarrow i$. The compositions $f' \circ f: i \longrightarrow i$ and $f \circ f': i' \longrightarrow i'$ must be unique as well, implying that $f' \circ f = \mathtt{id}_{\mathtt{i}}$ and $f \circ f' = \mathtt{id}_{\mathtt{i}'}$. Hence, i and i' are isomorphic. The empty set, $\emptyset$ is the single initial object of **Set**, since for any set a there is a unique function $f: \emptyset \longrightarrow a$ whose domain is $\emptyset$ and whose codomain is a, namely, the vacuous function, since there are no elements in $\emptyset$ to map to an element of a. There are an infinite number of terminal objects in **Set**, namely the singletons $\{x\}$, since there is a single function $f: a \longrightarrow \{x\}$ mapping the elements of any set a to x. One of the most important uses of commutative diagrams and terminal and initial objects is in the definition of a *limit* of a diagram and the dual type of quantity, a *colimit*, two categorical constructs that we use extensively.

2.3. *Ontologies as categories*

The category with which we begin a study of ontologies is one we call **Concept**. Actually, this can be any one of several categories as described in Crole (1993), Goguen and Burstall (1992) or Meseguer (1989), depending upon the kind of logic we wish to use for its theory objects, our concepts. For illustration, our theories will be expressed in a first-order multi-sorted predicate calculus with products. A **Concept** morphism $s: T \longrightarrow T'$ is an association of the syntax of a concept T with a part of the syntax of a concept T', such

that the statements of T' associated with axioms of T are either axioms themselves or are provable as theorems in T'. In general, there can be more than one such association, and, hence, **Concept** is not a partial order. To begin with an example, we can write a theory T_0 for a simple geometry of points and lines in which points are undefined primitives, as

```
Concept T0
  sorts Points, Lines
  op on: Points*Lines →Boolean
  Axiom Two-points-define-a-line is
    forall(x, y:Points) ((x not = y) implies
      (exists L:Lines) (on (x, L) and on (y, L) and
        ((forall m:lines) (on (x, m) and on (y, m))
          implies (m = L) ))
end
```

The expression `sorts Points, Lines` introduces the most basic sorts of T_0. Using a sorted logic is merely a convenience, of course: Predicates can be used in place of sorts and the explicit typing of variables, for example in the quantified sentence fragment `forall(x, y : Points)`$(\ldots)$, can be replaced by an implication involving the predicates with un-typed arguments, `forall(x, y) (( Points(x) and Points(y))` $\Longrightarrow \ldots$). The line `op on: Points*Lines` $\rightarrow$ `Boolean` specifies that the operation symbol `on` represents a predicate that maps any ordered pair (x, l) consisting of a point and a line to `true` or `false`. The term `Points*Lines` denotes a product sort whose instances are the ordered pairs. The `Boolean` sort is part of a theory of boolean operations that is implicitly included in every concept (it is an initial object of the concept category). An axiom labeled `Two-points-define-a-line` uses the predicate `on` with its two arguments, a point and a line, to decree that any two distinct points are "on" some unique line. As discussed in, for example, (Bennet 1995), this axiom serves as a basis for several different geometries. These include Euclidean and non-Euclidean geometries as well as geometries that do not involve continua. Each can be regarded as a specialization of the very abstract theory T_0, elaborating it with more sorts, axioms, predicates and so forth.

The categorical approach to concepts is very useful in modeling the declarative semantics of systems, especially systems such as neural networks, which are distributed over many components organized in a stratified or hierarchical manner. The components in the "lower" strata, having a fairly abstract semantics, may behave in a relatively simple or generalized manner. Assemblies of such components can then have a more complex semantics that builds on the abstractions; this is expressed declaratively by associating them with theories that incorporate those of the simpler components. For example, let $T_i\,(i = 1, 2, \ldots, N)$ be N copies of a theory which adds a point constant `p1` and a line constant `L1` to T_0:

```
Concept Ti
  sorts Points, Lines
  const p1: Points
  const L1: Lines
  op on: Points*Lines → Boolean
  Axiom Two-points-define-a-line is
    forall(x, y:Points) ((x not= y) implies
      (exists L:Lines) (on (x, L) and on (y, L) and
        ((forall m:lines) (on (x, m) and on (y, m))
          implies (m = L) ))
  Axiom
    on (p1, L1)
end
```

Each T_i can be associated with one of N individual sensor elements for an artificial neural network whose input nodes receive input stimuli from the individual pixels of a digital camera, in a manner loosely (very loosely) analogous to retinal cells sending input stimuli to the brain (see Figure 2). The point constant `p1` in the copy T_i associated with pixel i is interpreted as the "point stimulus" read by pixel i; when this pixel is illuminated in the image, the situation that produced the image is regarded as an instance of T_i. The axiom consisting of the single term `on (p1, L1)` specifies that the point represented by `p1` is "on" the line represented by `L1`. The pixel array being a rectangular grid, it is easy to envision lines being interpreted as the various rows, columns, and diagonal and transversal lineups of pixels so that a unique line can be

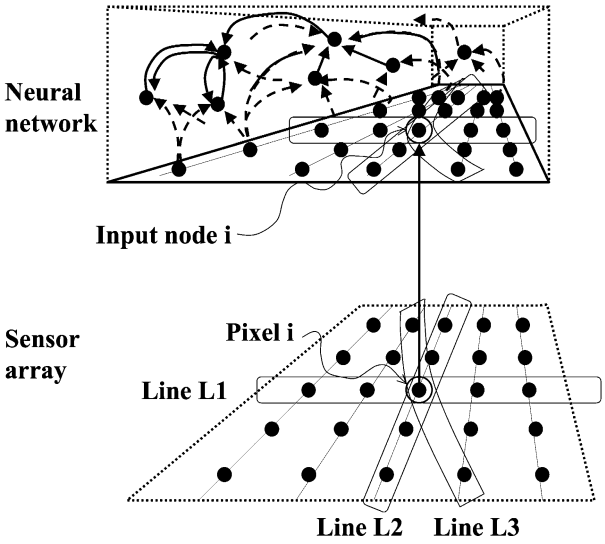


Figure 2. A neural network interpreting input from a digital array as geometry. A point (pixel stimulus) is shown at the intersection of three lines.

determined from any two active pixels. Given that the grid is finite with a row-column structure, each line will have many or few points and be long or short in length depending upon which linear array of pixels it represents. Another item that will be useful in this discussion is a theory T_L that simply adds a line constant to T_0:

```
Concept TL
  sorts Points, Lines
  const L1: Lines
  op on: Points*Lines → Boolean
  Axiom Two-points-define-a-line is
    forall(x, y:Points) ((x not= y) implies
      (exists L:Lines) (on (x, L) and on (y, L) and
        ((forall m:lines) (on (x, m) and on (y, m))
          implies (m = L) ))
end
```

The line constant in each of the identical theories T_i denotes an arbitrary line, with no location or orientation specified. The point

constant is also arbitrary. But this is a syntactic property, true of the theories as they are expressed. In any application of the mathematical semantic model in which these theories are to be associated with specific network components, their syntax is to be given a semantic in terms of the architectural structure and operation. This results in an assignment of some, but not necessarily all, theory terms to quantities associated with the activities of network components, a process which will be formalized when we introduce the categorical structural mappings. We shall call a term "grounded" when it is assigned a semantic in this way. We begin by assigning basic concepts to the neural network's input nodes and possibly other nodes directly connected to them. The concept associated with an input node is a theory about the situations associated with arbitrary stimuli that activate that node. It can be expressed at any level of detail desired and, as mentioned, the association need not ground all terms. The terms in an input concept and the associations made for them, together with the network's overall structure and operation, will determine all future explicit concept representations the network can form. The level of detail possible in these representations and their definiteness of meaning will depend upon the detail of the input concept representations and the extent to which their terms are assigned to network quantities.

We shall present the proper formal treatment for interpreting an ontology in this way following the necessary preliminaries, where we introduce categorical structure mappings and neural categories. Proceeding informally for now, we assign each T_i to a particular input node, labeled i, and regard the point constant `p1` of T_i as the symbol for the spatial location of that node's stimuli, represented at the sensor by locating pixel i in the array. No association is given for the line constant `L1`, but its use in the axiom `on (p1, L1)` constrains its future interpretations as one of the lines containing `p1`. The intent is to respect these associations in the future as we analyze the computations of the neural network and the representations of camera images it forms as it adapts its connection weights. We analyze the results of the adaptations via constructions performed in the category **Concept**. Before considering adaptation, therefore, let us discuss concept morphisms.

A concept morphism maps all quantities in one theory to quantities of the same kinds in the other; for example, sorts map to sorts. In addition, each axiom of the domain theory of the

morphism must map to a statement that is valid in the codomain theory – an axiom or theorem – when we transform the statement in the domain by substituting its symbols by their images under the symbol mapping of sorts, operations, and constants. For example, there are N morphisms $s_{0,i}: T_0 \longrightarrow T_i$. The morphism $s_{0,i}$ maps the sort symbols `Points` and `Lines` to the same sort symbols in each T_i and the `on` predicate to the same operation in T_i. We write it as

$$\begin{array}{rrcl} \text{Morphism } s_{0,i}: & \texttt{Points} & \mapsto & \texttt{Points} \\ & \texttt{Lines} & \mapsto & \texttt{Lines} \\ & \texttt{on} & \mapsto & \texttt{on} \end{array}$$

The axiom of T_0 is transformed to the identical axiom in T_i by substitution of identical symbols.[1] The morphisms $s_{0,i}$ are very simple, since the codomain theory in each differs from the domain theory T_0 only in the addition of the point and line constants `p1` and `L1` and the axiom involving these constants and the `on` predicate. The constants (hence, the axiom `on (p1, L1)`) are not involved in the symbol maplets or the translation of axioms because nothing maps to them. Most concept morphisms are more complex than this. Non-trivial symbol mappings and more symbols are typically involved. In general, if $s: T \longrightarrow T'$ is a theory morphism, t is a statement of T and u is either a sort, operation or constant symbol appearing in t, we obtain the image t' of t in T' by substituting $s(u)$ for each appearance of u in t.

By associating the theories with the activity levels in the nodes of a neural network, and theory morphisms with connection structures between active nodes, we can provide interpretations for the theories and morphisms within the computational structure of the network, equipping the network with a declarative semantics. In particular, we can associate constants with their intended semantics even with the relatively abstract geometry theories discussed here. This has two major implications: First, it provides a means of associating an ontology with a neural network, theory-by-theory, component-by-component. Second, it provides a means for grounding terms in the architecture and operation of a system; moreover, an initial grounding of selected terms will determine that part of the semantics that depends on the selected groundings, which is given by the categorical structure of the ontology. These remarks will be formalized in future sections. First, we need to introduce more categorical notions, beginning with limits and colimits.

3. BUILDING ONTOLOGIES VIA CONSTRUCTIONS IN CATEGORIES

Whether the goal of analysis is to understand the modular structure of an ontology or its incremental buildup from a collection of basic theories, the mathematical semantic model provides a tool set for analysis in the form of constructions in categories. Here, we explore constructions in a theory category and in later sections associate these with similar constructions by which we can understand the semantics of neural network substructures. The categorical constructions, chiefly colimits and limits, also provide a mathematical basis for understanding long-term adaptation, or learning. Colimits will be described first, since they have a history of use in categorical logic and computer science (Goguen and Burstall 1992; Williamson et al., 2001). A theorem in category theory can be used to derive an algorithm for calculating limits in any category that contains limits for all of its diagrams, and similarly for colimits by dualization; see The Limit Theorem in (Pierce, 1991). Colimits and limits do not exist for all diagrams in all categories, but they can be very useful where they do exist.

3.1. *Colimits*

Let Δ be a diagram in a category C as shown in Figures 3 and 4, with objects a_1, a_2, a_3, a_4, a_5 and morphisms $f_1: a_1 \longrightarrow a_3, f_2: a_1 \longrightarrow a_4, f_3: a_2 \longrightarrow a_4, f_4: a_2 \longrightarrow a_5$. The diagram $\overline{\Delta}$ in Figure 4 extends Δ to a commutative diagram with an additional object b and morphisms $g_i: a_i \longrightarrow b (i = 1, \ldots, 5)$, provided additional objects and morphisms with the requisite properties exist in C. That is, $g_1 \circ f_1 = g_2 = g_3 \circ f_2$ and $g_3 \circ f_3 = g_4 = g_5 \circ f_4$. The added cone-like structure K consisting of the *apical object* b and *leg morphisms* g_1, g_2, g_3, g_4, g_5 is called a *cocone* for diagram Δ. In general, a diagram can have many cocones or it can have few or none, depending upon the available objects and morphisms in C. Given cocones K' and K'' for Δ in Figure 3, with respective apical objects b', b'' and leg morphisms g'_i and g''_i $(i = 1,\ldots,5)$, a *cocone morphism with domain K' and codomain K''* is a C-morphism $h: b' \longrightarrow b''$ having the property

$$g''_i = h \circ g'_i (i = 1, \ldots, 5). \tag{1}$$

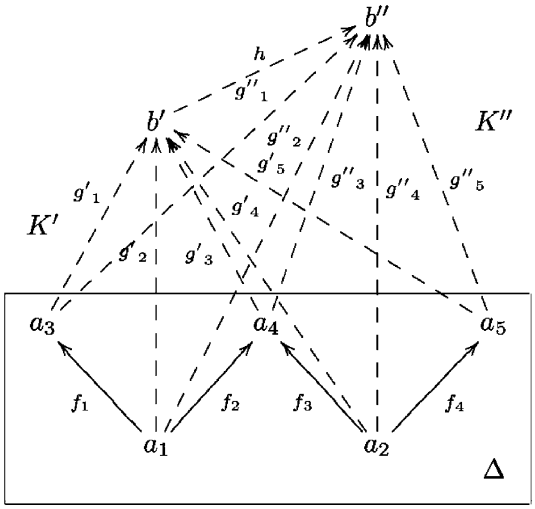


Figure 3. A cocone morphism $h : K' \longrightarrow K''$ in $\mathbf{coc}_\Delta$ is a morphism $h : b' \longrightarrow b''$ in C between the apical objects b' and b'' of cocones K' and K'', respectively, that is a factor of each leg morphism $g''_i : a_i \longrightarrow b''$ of K'', with $g''_i = h \circ g'_i$.

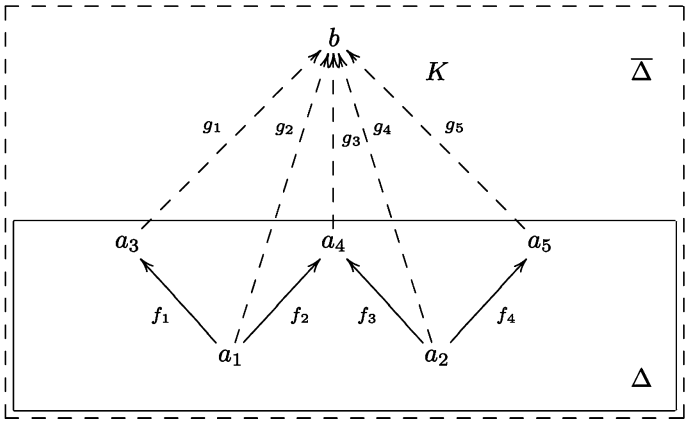


Figure 4. A colimit for a diagram Δ. The extended diagram $\overline{\Delta}$ extends Δ with a conical structure of morphisms from all diagram objects $a_1,\ldots,a_5$ pointing to an apical object b.

That is, h is a factor of each leg morphism g''_i of K'' by composition with the leg morphism g'_i of K', as illustrated in Figure 3.

Re-using the symbol h for notational efficiency, we denote the cocone morphism determined by h as $h : K' \longrightarrow K''$.

With morphisms so defined, and composition of cocone morphisms following directly from composition of C-morphisms, the cocones for Δ form a category, $\mathbf{coc}_\Delta$. A *colimit for the diagram* Δ is an initial object K in the category $\mathbf{coc}_\Delta$. That is, for every other cocone K' for Δ, there exists a unique cocone morphism $h : K \longrightarrow K'$. The original diagram Δ is called the *base diagram* for the colimit and the diagram $\overline{\Delta}$ formed by adjoining K to Δ is called its *defining diagram*. Note that, as all initial objects are isomorphic, all colimits for a given base diagram are isomorphic.

3.2. *Limits*

The notion of limits in a category can be obtained directly from notion of colimits by "reversing the arrows" and replacing initial objects with terminal objects. Let Δ be a diagram in a category C as shown in Figure 5, with objects a_1, a_2, a_3 and morphisms $f_1 : a_1 \longrightarrow a_3$ and $f_2 : a_2 \longrightarrow a_3$. The diagram $\underline{\Delta}$ extends Δ to a commutative diagram with an additional object b and morphisms $g_i : b \longrightarrow a_i (i = 1, \ldots, 3)$, provided additional objects and morphisms with the requisite properties exist in C, that is, $f_1 \circ g_1 = g_3 = f_2 \circ g_2$. The conical structure K is called a *cone*; note that its morphisms are directed into the diagram, the opposite sense of the leg morphisms of a cocone. Cone morphisms are defined appropriately by analogy with cocone morphisms. Again, composition follows directly from the composition of C-morphisms and the cones for Δ form a category, $\mathbf{cone}_\Delta$. A *limit for the diagram* Δ is a terminal object K in the category $\mathbf{cone}_\Delta$. As with colimits, the original diagram Δ is called the *base diagram* for the limit and the diagram $\underline{\Delta}$ is called its *defining diagram* and, as all terminal objects are isomorphic, all limits for a given base diagram are isomorphic.

The theory category **Concept** described in Section 3 contains colimits for all of its diagrams. This means that once an appropriate diagram is available, a colimit theory – a more specialized theory incorporating all the information in the diagram – can be calculated by an automated process. On the other hand, the theory category does not contain limits for all diagrams. Thinking of a diagram as representing a situation, real or imagined, diagrams that do have limits would seem to represent situations of greater

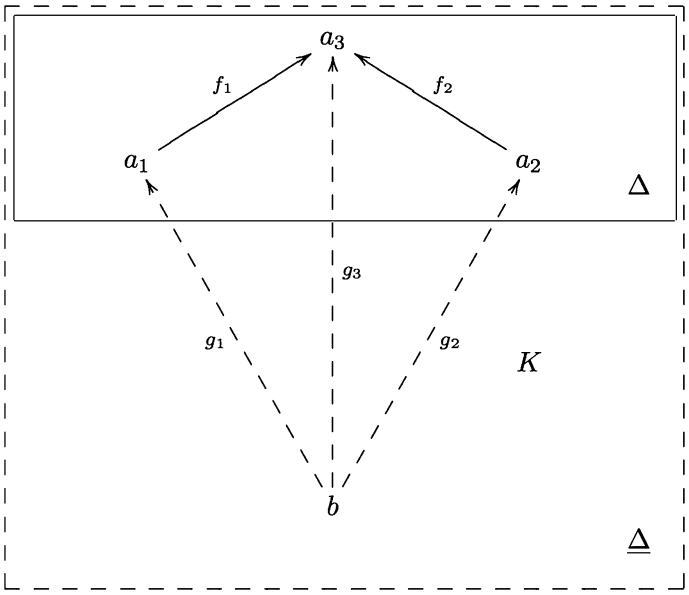


Figure 5. A limit for a diagram Δ. The extended diagram $\underline{\Delta}$ extends Δ with a conical structure of morphisms to all diagram objects $a_1,\ldots,a_3$ from an apical object b.

than average significance. The next two sections will exploit our geometry theory and pixel array example to present examples of both constructs in **Concept**.

3.3. *A Geometry colimit theory*

Geometry theories (or concepts) such as the simple theories T_0, T_L and $T_1, T_2, \ldots, T_N$ described in Section 2.3, can be used as the primitives for forming a geometry ontology within the category **Concept**. Colimits provide the means for "filling out" the ontology in the direction of greater specificity and limits provide the opposite direction, that of abstraction. To see this, let us examine two simple examples, beginning with the colimit illustrated in Figure 6.

In this simple example, theory T_L is the domain for morphisms $s_1\colon T_L \longrightarrow T_1$ and $s_2\colon T_L \longrightarrow T_2$. The morphisms have the same form, since corresponding quantities in T_1 and T_2 have the same name:

Morphism s_1 :	`Points`	$\mapsto$	`Points`
	`Lines`	$\mapsto$	`Lines`
	`on`	$\mapsto$	`on`
	`L1:Lines`	$\mapsto$	`L1:Lines`
Morphism s_2 :	`Points`	$\mapsto$	`Points`
	`Lines`	$\mapsto$	`Lines`
	`on`	$\mapsto$	`on`
	`L1:Lines`	$\mapsto$	`L1:Lines`

The theories T_L, T_1 and T_2 and morphisms s_1: $T_L \longrightarrow T_1$ and s_2: $T_L \longrightarrow T_2$ constitute the base diagram Δ in Figure 6. The colimit cocone is shown extending the base diagram to the defining diagram $\overline{\Delta}$. This cocone contains a theory T_{2pL}, the colimit object, and the morphisms r_1: $T_L \longrightarrow T_{2pL}$, r_2: $T_1 \longrightarrow T_{2pL}$ and r_3: $T_2 \longrightarrow T_{2pL}$, with

$$r_2 \circ s_1 = r_1 = r_3 \circ s_2.$$

The content of the colimit theory T_{2pL} is a consequence of the commutativity of $\overline{\Delta}$ and the initiality of the colimit cocone. To explain this, a brief comment regarding the sameness of the terms in the base diagram theories T_1 and T_2 is in order. In general, to avoid confusion when two theories such as T_1 and T_2 contain the same term u and the two copies of u are mapped to separate copies in another theory without an explicit name change, each is indexed by its theory of origin in the codomain theory of the mapping: For example, the name of the point constant `p1` from theories T_1 and T_2 is extended to `p1.T1` and `p1.T2`, respectively, in the theory T_{2pL}, which is the codomain of the morphisms r_2: $T_1 \longrightarrow T_{2pL}$ and r_3: $T_2 \longrightarrow T_{2pL}$. But why is `p1` the only term for which this is necessary; why do the other terms, identical between T_1 and T_2, appear only once in T_{2pL} and, hence, do not require showing their extended names? The answer is that the two copies of the other terms in T_1 and T_2 are "pasted" or "blended" together as a consequence of the commutativity of $\overline{\Delta}$. For the equality $r_2 \circ s_1 = r_1 = r_3 \circ s_2$ to hold, separate symbols of T_1 and T_2 that are images of the same symbol of T_L under the two diagram Δ morphisms s_1

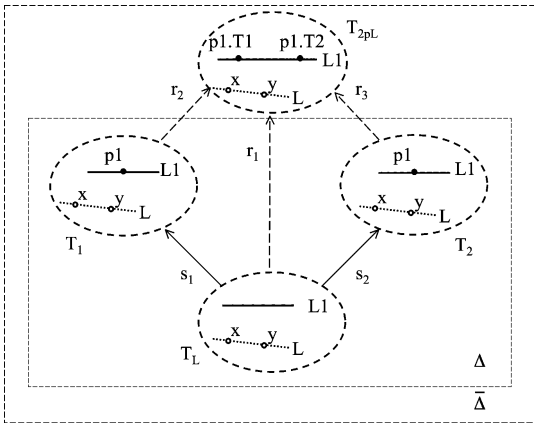


Figure 6. A pictorial illustration of the colimit base and defining diagrams Δ and $\overline{\Delta}$. The contents of the concepts involved are pictured along with the diagrammatic structure. Solid dots and lines signify point and line constants.

and s_2 must merge into a single symbol in the colimit apical concept T_{2pL}. Thus, terms such as `Points`, `Lines`, `on`, and `L1`: Lines appear in T_{2pL} only once. The point constant `p1`, however, does not appear in T_L; as a consequence, there is nothing in T_L to map to the two copies of p_1, and, hence, there is no basis for "blending" them together. The only alternative is for them to remain distinct in the colimit object T_{2pL}, as `p1.T1` and `p1.T2`. This is expressed in the calculated leg morphisms resulting from the requirement that $\overline{\Delta}$ commute:

Morphism $r_1 : T_L \longrightarrow T_{2pL}$:

`Points`	$\mapsto$	`Points`
`Lines`	$\mapsto$	`Lines`
`on`	$\mapsto$	`on`
`L1:Lines`	$\mapsto$	`L1:Lines`

Morphism $r_2 : T_1 \longrightarrow T_{2pL}$:

`Points`	$\mapsto$	`Points`
`Lines`	$\mapsto$	`Lines`
`on`	$\mapsto$	`on`
`L1:Lines`	$\mapsto$	`L1:Lines`
`p1:Points`	$\mapsto$	`p1.T1:Points`

Morphism $r_3 : T_2 \longrightarrow T_{2pL}$:

`Points`	$\mapsto$	`Points`
`Lines`	$\mapsto$	`Lines`
`on`	$\mapsto$	`on`
`L1:Lines`	$\mapsto$	`L1:Lines`
`p1:Points`	$\mapsto$	`p1.T2:Points`

As a consequence, there are two images of the term `on (p1, L1)` obtained by substitution with the two point constants: `on (p1.T1, L1)` and `on (p1.T2, L1)`. Finally, the initiality of a colimit cocone guarantees that its apical object T_{2pL} contains only those terms obtainable from its constituent theories; the non-colimit cocones have more complex theories at their apices. From these considerations, T_{2pL} can be written as

```
Concept T2pL
  sorts Points, Lines
  const p1.T1: Points
  const p1.T2: Points
  const L1: Lines
  op on: Points*Lines -> Boolean
  Axiom Two-points-define-a-line is
    forall(x, y:Points) ((x not = y) implies
      (exists L:Lines) (on (x, L) and on (y, L) and
        ((forall m:lines) (on (x, m) and on (y, m))
          implies (m = L) ))
  Axiom
    on (p1.T1, L1)
  Axiom
    on (p1.T2, L1)
end
```

To summarize, the colimit theory T_{2pL} "blends" the theories T_1 and T_2 along their common sub-theory T_L. In general, there can be many sub-theories for theories that are to be combined in this manner, and, hence, many "blending sites" can appear in a diagram.

This simple example illustrates the advantage of applying category theory in ontology development. Theories and their morphisms can be used to specify logically closed fragments of knowledge, beginning with a basic set of theories and morphisms encompassing the quantities needed. The colimit calculation provides a mathematically rigorous, automated technique for constructing ever more complex concepts starting with the basic ones, "filling out" the ontology in the direction of greater specialization.

3.4. *A geometry limit theory*

The direction opposite specialization for "filling out" an ontology is that of abstraction. This is done with limits. A very simple kind of limit is that for a discrete diagram, one having objects but no morphisms. Certain discrete diagrams yield concepts about spatial invariants. To illustrate the derivation of spatial invariants as limits, we elaborate upon the geometry example by including information about the spatial location and orientation in theories that express the presence of a line in the pixel array, with one line per theory. Referring to the neural network receiving input from a pixel array as illustrated in Figure 2, suppose that J theories $S_1, S_2, \ldots, S_J$ are associated with neural network components connected to the input nodes so that they each describe one of the lines. Without discussing the details of network operation, we can regard this representation as follows: When two or more pixels on a line are illuminated, a network node associated with one of the S_i becomes activated; moreover, only under this circumstance does that node become activated. The activation, then, signals the presence of that line in the current neural network input from the sensor array. To express this property, S_i contains a line constant as does the theory T_L. In addition, however, S_i associates its line constant with a unique, specific spatial location and orientation, expressed directly in the syntax of the theory. Just for variety, let us give the line constant a unique name in each theory – say, L_1 in theory S_1, L_2 in theory $S_2, \ldots, L_J$ in theory S_J. Combining the location and orientation into a single sort for simplicity, the theories have the following form:

```
Concept Si
  sorts Points, Lines
  sort Num
  constants 0, 1,..., operations and axioms for Num
  const Li : Lines
  op spatial: Lines -> Num
  op on: Points*Lines -> Boolean
  Axiom Two-points-define-a-line is
    forall(x, y:Points) ((x not = y) implies
      (exists L:Lines) (on (x, L) and on (y, L) and
        ((forall m:lines) (on (x, m) and on (y, m))
          implies (m = L) ))
  Axiom
      spatial(Li) = i
end
```

The sort `sort Num` represents the natural numbers, included as a subtheory of each S_i as indicated (for brevity) by the line following the sort declaration. The operation `spatial` selects a number for each line. For simplicity, we let this number serve as a code for the spatial location and orientation of the line. This could be actualized, for example, by maintaining a table uniquely associating each number with the pixel array row/column indices of the two end points of a line. Each line constant defined in the theories S_i is associated with a number, one to each theory: `spatial (L1)= 1` in theory S_1, `spatial (L2) = 2` in theory S_2, etc. A table look-up thereby provides the row-column indices, hence spatial location, of the line in each theory S_i.

The theories $S_1, S_2, \ldots, S_J$ form a discrete diagram, and a terminal cone for this diagram can be shown to exist. In fact, the apical object must be a theory isomorphic to the following (via theory morphisms that simply represent term renaming):

```
Concept R
  sorts Points, Lines
  sort Num
 constants 0, 1,..., operations and axioms for Num
  const L: Lines
  op spatial: Lines -> Num
  op on: Points*Lines -> Boolean
  Axiom Two-points-define-a-line is
    forall(x, y:Points) ((x not = y) implies
      (exists L:Lines) (on (x, L) and on (y, L) and
        ((forall m:lines) (on (x, m) and on (y, m))
          implies (m = L) ))
  Axiom
      exists (n:Num) (spatial (L) = n)
end
```

The terminal cone, or limit, for a discrete diagram is called a *product* and its leg morphisms $r_i : R \longrightarrow S_i$ are *projections*.[2] The leg morphisms have the form:

Morphism $r_i : R \longrightarrow S_i$:

Points	↦	Points
Lines	↦	Lines
Num	↦	Num
spatial	↦	spatial
on	↦	on
L:Lines	↦	L*i*:Lines

Notice that the only change between R and S_i is the Skolemization (Andrews, 1986, pp. 123–127) of the existentially quantified variable n of sort Num in R by the constant i of sort Num in theory S_i. This effectively declares that the line constant L in R is a spatial invariant, existing in an arbitrary location that is specialized to a definite location, i, for each of its correspondents L_i in S_i.

This example suggests that limits can be useful in abstraction close to the sensor level in knowledge representation: They can express spatial invariants and other primitives which can then be used in a flexible fashion through specializations of the concepts in which they appear. Unlike colimits, however, limits are not available for all diagrams in the category **Concept**. This is a mathematical way of

saying that, while a situation can always be precisely expressed in a theory comprising a blended combination of simpler theories, a precise abstraction for a situation does not always exist.

An ontology can be represented incrementally by applying limits in combination with colimits beginning at and near the sensor level. Spatial and other sensory invariants are expressed in the earliest limits. Later in the process, limits derive abstractions from colimits, increasing the flexibility in forming increasingly complex concepts.

4. STRUCTURAL MAPPINGS: FUNCTORS AND NATURAL TRANSFORMATIONS

To formalize the association of an ontology with a neural network, we make use of the categorical notion of structure-preserving mappings. The first kind of mapping to discuss is a *functor*, which transports the structure of one category into another. A functor $F : C \longrightarrow D$, with domain category C and codomain category D, associates to each object a of C a unique image object $F(a)$ of D and to each morphism $f : a \longrightarrow b$ of C a unique morphism $F(f) : F(a) \longrightarrow F(b)$ of D. Moreover, F preserves the compositional structure of C, as follows. Let $\circ_C$ and $\circ_D$ denote the separate composition operations in categories C and D, respectively. For each composition $g \circ_C f$ defined for morphisms of $C, F(g \circ_C f) = F(g) \circ_D F(f)$, and for each identity morphism of $C, F(\mathtt{id}_a) = \mathtt{id}_{F(a)}$. It follows that F preserves the commutativity of diagrams, that is, the images of the objects and morphisms in a commutative diagram of C form a commutative diagram in D. This means that any structural constraints expressed in C are translated into D and, hence, F is a structure-preserving mapping.

Not only are there structure-preserving mappings between categories, but also structure-preserving relations between the mappings themselves. A *natural transformation* $\alpha : F \longrightarrow G$ with domain functor $F : C \longrightarrow D$ and codomain functor $G : C \longrightarrow D$ consists of a system of D-morphisms α_a, one for each object a of C, such that the diagram in D shown in Figure 7 commutes for each morphism $f : a \longrightarrow b$ of C. That is, the morphisms $G(f) \circ \alpha_a : F(a) \longrightarrow G(b)$ and $\alpha_b \circ F(f) : F(a) \longrightarrow G(b)$ are actually one and the same, $G(f) \circ \alpha_a = \alpha_b \circ F(f)$. In a sense, the two functors have their morphism images $F(f) : F(a) \longrightarrow F(b), G(f) : G(a) \longrightarrow G(b)$ "stitched together" by other morphisms α_a, α_b existing in D, indexed by the

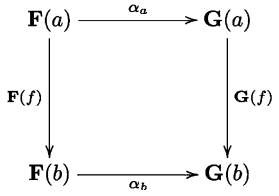


Figure 7. A commutative diagram associated with a natural transformation. The morphisms $\mathbf{G}(f) \circ \alpha_a : \mathbf{F}(a) \longrightarrow \mathbf{G}(b)$ and $\alpha_b \circ \mathbf{F}(f) : \mathbf{F}(a) \longrightarrow \mathbf{G}(b)$ are one and the same, $\mathbf{G}(f) \circ \alpha_a = \alpha_b \circ \mathbf{F}(f)$.

objects of C. Composition of the morphisms along the two paths leading from one corner $F(a)$ of a commutative square to the opposite corner $G(b)$ yields the same morphism, independently of the path traversed.

5. MODEL SPACES

Each concept T has an associated space of models, mathematically expressed situations or "possible worlds" described by T. A model of the theory T_{2pL} in the colimit example consists of any arrangement of entities corresponding to the sorts and operations and satisfying the theory axioms, with the constants serving as labels for entities in the world. An obvious example is one constructed using points and straight lines in the Euclidean plane. Another can be constructed from the grid of pixels in a video image display, with specific rules for identifying sets of pixels forming lines (such as rows, columns, and diagonals at all orientations in the pixel grid as in Figure 2). If this is to be a model of theory T_{2pL}, we must have identified a line to serve as an interpretation of the constant `L1` and two points, both on the line, to interpret `p1.T1` and `p1.T2`. In a set-theoretic model, there must also be sets which are suitable as interpretations of `Points` and `Lines`, the line must be a member of the set interpreting `Lines`, and the two points must be members of the set interpreting `Points`. The requirements on models for T_1 and T_2 are less stringent, since there need be only one point interpreting `p1` specified on the line interpreting `L1`, and the requirements on models of T_L are less stringent yet.

We denote the space of models of a concept T by $\mathtt{Mod}(T)$. For models based upon sets, the mathematical structure of a member σ of $\mathtt{Mod}(T)$ is as follows. For each sort u of T, there is a set u_σ. For

each operation p of T, where p is given the form $p : u \longrightarrow u'$, there is a function $p_\sigma : u_\sigma \longrightarrow u'_\sigma$ mapping members of u_σ to members of u'_σ (recall that if p is a predicate, u' must be `Boolean`, so that $u'_\sigma = \{\texttt{T}, \texttt{F}\}$). For each constant c with sort u, there is $c_\sigma \in u_\sigma$. Finally, each axiom of T must be valid for all quantities u_σ, p_σ, c_σ.

A model σ in the Euclidean plane for any of the theories of the colimit example has $\texttt{Points}_\sigma$ as the set of points and $\texttt{Lines}_\sigma$ as the set of straight lines in the plane, and the operation `on: Points*Lines-> Boolean` is interpreted by a function $p_\sigma : \texttt{Points}_\sigma \times \texttt{Lines}_\sigma \longrightarrow \texttt{Boolean}_\sigma$. The function p_σ has the Boolean value `T` for those point-line pairs with the property that the point lies on the line. The axiom

```
...
  Axiom Two-points-define-a-line is
    forall(x, y:Points) ((x not= y) implies
      (exists l:Lines) (on (x, l) and on (y, l) and
        ((forall m:lines) (on (x, m) and on (y, m))
          implies (m = l) ))
```

holds for all pairs of quantities $x, y \in \texttt{Points}_\sigma$ satisfying the condition that there exists a unique member of $\texttt{Lines}_\sigma$ defined by x and y. There are additional quantities, such as $\texttt{L1}_\sigma$ with $\texttt{L1}_\sigma \in \texttt{Lines}_\sigma$, required for the models of the theories T_1, T_2, and T_{2pL}.

The notion that a more specific concept T' can incorporate a less specific (more abstract) concept T has already been formalized in the definition of a concept morphism $s : T \longrightarrow T'$. Correspondingly, the notion that each model of T' corresponds to a model of T is captured in the definition of a model-space morphism $\texttt{Mod}(s) : \texttt{Mod}(T') \longrightarrow \texttt{Mod}(T)$. The morphism $\texttt{Mod}(s)$ acts as a function that maps every model σ' in $\texttt{Mod}(T')$ to a unique model σ in $\texttt{Mod}(T)$, that is, $\sigma = \texttt{Mod}(s)(\sigma')$. Notice that $\texttt{Mod}(s)$ has the reverse direction to s, as is appropriate. Since the theory T typically has less structure (fewer sorts, operations, constants, and/or axioms) than the theory T', the model σ is not required to have all the sets, functions, predicates, and specified members (constants) that σ' must have, and is only required to obey the axioms of T, which are typically fewer in number than those of T'. Nothing prevents a

model of T from having all the structure of a model of T', but an arbitrary model of T is not *required* to have all the structure that a model of T' is required to have. By the same token, there can be many models in $\mathtt{Mod}(T)$ that are not the image of a model in $\mathtt{Mod}(T')$ under the mapping $\mathtt{Mod}(s)$: *A simpler theory poses fewer restrictions on its models, hence, there can be more of them.*

Notice the use of the term "model spaces" as opposed to "sets of models". As noted previously, the demands placed upon a model in the domain of a model-space morphism are greater than those upon a member of the codomain. On the other hand, some members of a given model space can be more elaborate than is required by the corresponding theory. A model of a theory T can often be thought of as embedded within other, more elaborate models that satisfy the constraints of T. This notion is expressed through *model morphisms*, and, indeed, the model spaces $\mathtt{Mod}(T)$ are in actuality categories and the model-space morphisms $\mathtt{Mod}(s) : \mathtt{Mod}(T') \longrightarrow \mathtt{Mod}(\mathtt{T})$ are functors. We use the term "space" rather than "category" because we do not intend to explore the categorical properties of model spaces here, although it is significant that there is yet more depth to be explored in categorical model theory. See (Meseguer, 1989) or (Goguen and Burstall, 1992) for a further discussion of model categories and (Mac Lane, 1971) or (Adamek et al., 1990) for foundational issues concerning the existence of a category of categories.

6. NEURAL ARCHITECTURES AND NEURAL CATEGORIES

Applying the ideas about the incremental representation of an ontology developed in previous sections to express the semantics of neural networks requires a foundation for categories within which neural structure can be represented. Since neural activation and connection-weighted signaling computations are involved along with long-term adaptation or learning, the categorical neural model must have a sense in which it is dynamic. This means expressing not just the interconnection structure joining the nodes of a neural network, but also the rules governing the systematic activity of its nodes and connections as it interacts with its environment. On the other hand, it is not necessary to express all the details of the neural network dynamics; we need only represent the outcome at key stages of processing. The means for determining when such a stage

is present is left to the analyst who uses the semantic theory. The main requirement is that there be an unambiguous and consistently applied criterion associating states of activity and adaptation with neural category objects and their morphisms. Possible criteria for key stages of processing will be discussed here only in the most general manner. Note, however, that many artificial neural network models in the literature represent the results of neural computations at discrete steps (McCullough and Pitts, 1943; McClelland et al., 1986; Arbib, 1987; Fu, 1992; Craven and Shavlik, 1993; Andrews et al., 1995; Mitra and Pal, 1995; Pinkas, 1995; Sima, 1995; Kasabov, 1996; Healy and Caudell, 1997; Heileman et al., 1997; Healy, 1999a), and discrete neural network models have been applied to biological investigations in neuroscience (Perry, 1999; Riesenhuber and Poggio, 1999).

6.1. *Architectures*

An architecture A has nodes p_i $(i = 1, 2, \ldots, n_N)$ and connections c_k $(k = 1, 2, \ldots, n_C)$. The nodes have signal functions $\phi_i : \mathbf{R} \longrightarrow \mathbf{R}$, where $\mathbf{R}$ a convenient number system. For our purpose, this will be the commonly used real number system. A typical signal function is the sigmoid with a threshold value $\theta_{T,i}$,

$$\xi_i = \phi_i(\theta_i) = \frac{1}{1 + e^{-(\theta_i - \theta_{T,i})}} \quad (i = 1, 2, \ldots, n_N),$$

where the argument θ_i is an activation potential representing the connection-weighted algebraic sum of signals sent to node p_i from other nodes. This and other signal functions are meant to express the activation of a node when its weighted input sum θ_i exceeds $\theta_{T,i}$. Each connection c_k joins a pair of nodes, its source $p_i = \mathbf{n_S}(c_k)$ and its target $p_j = \mathbf{n_T}(c_k)$ $(1 \leq i, j \leq n_N)$. A connection can be either *excitatory*, tending to excite its target node, or *inhibitory*, tending to suppress its target. At any stage of adaptation, each connection c_k is associated with a numerical weight w_k which acts as a multiplier for the signals it transmits. The adaptation at that stage is represented by an n_C-tuple $(w_1, w_2, \ldots, w_{nC})$ called a weight state. The network transitions from one weight state to another by virtue of its connectionist or Hebbian learning algorithm. Activation and learning in a neural network occur simultaneously during these transitions in response to the input patterns the network processes,

which are presented in sequence. There are different versions of Hebbian (Hebb, 1949) learning depending upon the neural network model, but they generally involve the rule that w_k increases or decreases in magnitude depending upon the ongoing activity in $\mathbf{n_S}(c_k)$ and $\mathbf{n_T}(c_k)$.

The knowledge implicit in a neural network at a key stage of processing resides partially in the architectural design, partially in the activity of its nodes, and partially in the connection-weight values. We shall discuss the knowledge represented by the connection weights and the activations of the nodes as an interrelated system of concepts about the environment. The knowledge implicit in the network's design determines how and how well this concept structure is acquired and maintained. The concept structure begins with descriptions of what the network is capable of detecting in its inputs, and perhaps also with some concepts and relationships which, in combination with the input concepts, aid in forming new concept representations derived from the inputs. The level of detail expressed in these concepts will depend upon the analyst's level of knowledge about the properties of the sensors in relation to the environment, and the amount of detail desired for the analysis. Building upon the initial structure, colimit and limit derivations express mathematically the formation of more complex, specialized and less complex, abstract concept representations. Functors and natural transformations formalize the association of the concept structure with neural structures. Before discussing this, we need to formulate the objects and morphisms for neural categories.

6.2. *Neural categories*

A category $\mathbf{N}_{A,w}$ is associated with each pair (A, w) of architecture A and weight state w. Each input pattern e for A is presented to the *input nodes* of A, one pattern component per node, resulting in an evolving pattern of activity over the network. This activity together with the existing weight values w_k and the connection structure of A is the basis for the definitions of the objects and morphisms of $\mathbf{N}_{A,w}$. Weight adaptation, or "learning", changes one or more of the values w_k to values v_k at the next key stage of processing, a Hebbian-like response to the node activities resulting from the processing of e. This causes a transition to a neural structure expressed by a different category, $\mathbf{N}_{A,v}$. Let W_A, Θ_A, E_A

denote the sets of weight, activation and input tuples for A, respectively. To summarize formally the activation and learning algorithms associated with the architecture, we define a function $\Phi_A : W_A \times \Theta_A \times E_A \longrightarrow W_A \times \Theta_A$. The function Φ_A expresses the totality of neural processing of an input e via weighted summation of the input signals at each node, the signal function evaluations $\boldsymbol{\xi}_i = \phi_i(\theta_i)$, and any other assumptions embodied in the rules for activation in A, including feedback, recurrence, and connection weight adaptation. A transition in the neural network corresponding to a transition between neural categories is written $(v, \psi) = \Phi_A(w, \theta, e)$, with $w, v \in W_A; \theta, \psi \in \Theta_A; e \in E_A$. Objects, to be defined, are associated with network nodes. An *instance of an object* of $\mathbf{N}_{A,w}$ occurs when the output value ξ_i for its associated node p_i remains constant (to within some pre-selected tolerance value) during a transition $\Phi_A : (w, \theta, e) \mapsto (v, \psi)$. If the object represents one or more concepts, we associate ξ_i with the presence in the current input of models for some subset of them. We use the pair (θ, e) that initiates the transition to formalize the instance. Notice that, in actuality, an instance occurs globally, over the whole network; therefore, we can regard it also as an instance of the neural category $\mathbf{N}_{A,w}$. Notice that the set of all instances of any object or morphism of $\mathbf{N}_{A,w}$, and, indeed, of $\mathbf{N}_{A,w}$ itself, will be associated with transitions to many other categories $\mathbf{N}_{A,v}$.

We identify one or more concepts with the activities of an input node p_h, representing items and events that it detects in its component of the input patterns e as discussed with the colimit and limit examples associated with Figure 2. We can determine the concept or concepts represented by a node p_i in weight state w (assuming that it represents any) in one of two ways. In principle, we could simply perturb its neural network a number of times with different pairs (θ, e), each time starting its processing in weight state w and noting the items and events in the inputs to which it responds. Pursuing this to conclusion is impractical in most cases. Alternatively, we can trace backward through its input connections in the network, determining the paths leading to it from the input nodes p_h, whose semantics have been pre-assigned. This can be difficult in a complex network. Instead, the major application of the categorical model is determining whether a network has the *capability* to represent certain kinds of concept structures: What are its capabilities in representing colimits? Limits? And so forth. In describing the

neural category and related items, however, we shall proceed as if it were feasible to associate concepts and their morphisms with neural objects and morphisms for specific cases.

Because of uncertainties and other sources of variation in the input to a node p_i, we associate a concept with a set η of its output values ξ_i as opposed to a single value. If ξ_i takes real-number values, η is a real interval,[3] $\eta = \{\xi_i | \ell < \xi < u\}$ for some lower and upper bounds ℓ, u. For this reason, an obvious choice for the objects of a neural category $\mathbf{N}_{A,w}$ is the collection of pairs (p_i, η), where p_i is a node of A and η is a set of output values for p_i. A node can be associated with more than one concept; correspondingly, there can be many objects of $\mathbf{N}_{A,w}$ associated with the node, and, moreover, the intervals for its objects can overlap, sharing instances. In any case, we refer to a node p_i as the *carrier* of any object of the form (p_i,η).

The relationship between the activity in a neural network and the morphisms in its associated neural category is based upon the notion of a *signal path*. A signal path γ is a path through connections leading from a source node p_i to a target node p_j but with output intervals assigned to all nodes in the path, so that the source and target of γ are actually of the form (p_i,η) and (p_j, η'), respectively. In other words, a signal path is a string of neural category objects together with the connections joining their carrier nodes, having the form $[(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1})]$, with source and target objects $\mathbf{o_S}(\gamma) = (p_{\mu^1}, \eta_1) = (p_i, \eta)$ and $\mathbf{o_T}(\gamma) = (p_{\mu_{n+1}}, \eta_{n+1}) = (p_j, \eta')$.

Let θ denote an arbitrary n_N-tuple $(\theta_1, \theta_2, \ldots, \theta_{n^N})$ of activation states θ_i for the nodes p_i $(i = 1, \ldots, n_N)$ of $\mathbf{N}_{A,w}$. Let e denote an arbitrary n_I-tuple $(e_1, e_2, \ldots, e_{n_I})$ of input pattern values, where n_I is the number of nodes in the set S_I of input nodes (for simplicity, we are assuming that the input nodes are numbered consecutively as shown; in general, their indices can be assigned arbitrarily along with the other nodes). When the nodes p_{μ_r} in a signal path γ produce outputs lying within their corresponding intervals η_r, we say that the objects (p_{μ_r}, η_r) are activated.

The combination of a signal path γ with source and target $(p_i, \eta) = \mathbf{o_S}(\gamma)$ and $(p_j, \eta') = \mathbf{o_T}(\gamma)$ and a weight state w, has an associated set $U_{\gamma,w}$ of instances $(\theta, e) \in \Theta_A \times E_A$ satisfying the following requirements:

1. For all nodes p_{μ_r} of γ, $\phi_r(\theta_r)$ and $\phi_r(\psi_r)$ both lie in η_r, where for some $v \in W_A$, $(v, \psi) = \Phi_A(w, \theta, e)$.
2. For every connection (c_{k_r}) of γ, $v_{k_r} \neq 0$. In particular, (1) holds for the source and target nodes, $\phi_i(\theta_i), \phi_i(\psi_i) \in \eta$ and $\phi_j(\theta_j), \phi_j(\psi_j) \in \eta'$. In other words, the neural network processing associated with a pair (θ, e) in $U_{\gamma,w}$ maintains the objects of γ, regardless of the changes that occur in the connection weights of the network. The elements $(\theta, e) \in U_{\gamma,w}$ are called *instances of* γ *in weight state* w. The pairs (θ, e) associated with a node p_i generating an output within the designated interval η of a neural object (p_i,η) are called *instances* of (p_i,η).

For each set Γ of paths γ having a common source and target (p_i,η) and (p_j, η') (where now we write $(p_i, \eta) = \mathbf{o_S}(\Gamma)$, $(p_j, \eta') = \mathbf{o_T}(\Gamma)$), and each weight state w, there is an associated set of instances $U_{\Gamma,w}$ obtained by forming the intersection $U_{\Gamma,w} = \cap U_{\gamma,w}(\gamma \in \Gamma)$. Two path sets Γ and Γ' with a common source and target are *equivalent* in weight state w if $U_{\Gamma,w} = U_{\Gamma',w}$. All path sets which are pairwise equivalent have the same *closure*, a path set $\overline{\Gamma}_w$ that contains all their members. Notice that $U_{\overline{\Gamma}_w,w} = U_{\Gamma,w} = U_{\Gamma',w} = U_{\overline{\Gamma'}_w,w}$.

Finally, a morphism $m : (p_i, \eta) \longrightarrow (p_j, \eta')$ of $\mathbf{N}_{A,w}$ is given by a domain object (p_i,η), a codomain object (p_j, η'), a path set Γ with $(p_i,\eta) = \mathbf{o_S}(\Gamma)$ and $(p_j, \eta') = \mathbf{o_T}(\gamma)$, and all pairs $(\theta, e) \in U_{\Gamma,w} \subseteq \Theta_A \times E_A$. Any set Γ' equivalent to Γ defines the same morphism. Thus, a morphism can be represented by a weight state w and any of its equivalent path sets $\Gamma, \Gamma', \ldots$, each of which is called a *carrier* for the morphism in weight state w.

The definition of composition for morphisms in a category $\mathbf{N}_{A,w}$ follows easily from the foregoing. Consider a pair of morphisms m_1, m_2, with a path set Γ_1, Γ_2, respectively for each, with associated sets $U_{\Gamma_1,w}, U_{\Gamma_2,w}$ of instances and with $\mathbf{o_S}(\Gamma_2) = \mathbf{o_T}(\Gamma_1)$. We can form the intersection $U_{\Gamma_3,w} = U_{\Gamma_2,w} \cap U_{\Gamma_1,w}$, where Γ_3 contains the concatenations $\gamma;\ \gamma'$ of all pairs of paths $\gamma \in \Gamma_1$, $\gamma' \in \Gamma_2$, which is possible because $\mathbf{o_S}(\gamma') = \mathbf{o_T}(\gamma)$ for all such pairs. The pair (Γ_3, w) uniquely defines a morphism $m_3 : \mathbf{o_S}(\Gamma_1) \longrightarrow \mathbf{o_T}(\Gamma_2)$. This defines the composition $m_3 = m_2 \circ m_1$. Clearly, this scheme associates a unique morphism with the concatenation of any pair of path sets having the property that the source of one is the target of the other.

Notice that there can be path sets Γ whose source and target are one and the same object (p_i,η), with $\mathbf{o_S}(\Gamma) = (p_i, \eta) = \mathbf{o_T}(\Gamma)$. For

each w and each such path set there is a morphism $m : (p_i, \eta) \longrightarrow (p_i, \eta)$ of $\mathbf{N}_{A,w}$. For each object (p_i,η), we define a particular path set which is of fundamental importance: The singleton $\{[(p_i, \eta), (p_i, \eta)]\}$, whose only member we call the *virtual path* for (p_i, η). It is a path with no connection; instances of it are simply instances of (p_i,η). Its importance lies in its use in defining the identity morphism $\mathtt{id}_{(p_i,\eta)}$ as the morphism associated with $(\{[(p_i, \eta), (p_i, \eta)]\}, w)$. It is now a straightforward exercise to show that the composition as defined is associative and has an identity for each object. Therefore, the quantities we have been calling objects and morphisms satisfy the two axioms of composition, hence, they really are objects and morphisms and $\mathbf{N}_{A,w}$ with these definitions is indeed a category. We denote by $U_{\mathtt{id}}(p_i, \eta), w$ the instance set of the virtual path for object (p_i,η) (hence, the instance set of the identity morphism for the object). Any path set Γ equivalent to $\{[(p_i, \eta), (p_i, \eta)]\}$ has the same instances, and, hence, its members are acting collectively as the virtual path.

7. APPLYING CATEGORY-THEORETIC DESIGN PRINCIPLES

The fundamental notion in applying categorical semantics to neural networks is the representation of the concept hierarchy, our notion of an ontology, in a neural architecture at a given stage of learning. This begs the question of whether a neural network can support a representation of an ontology. An explicit representation of the entire hierarchy expressed in the category **Concept** is infeasible in any case, since this category is infinite and is meant to "contain" all possible knowledge, real or imaginary. Another difficulty is that **Concept** is meant to include knowledge associated with any sensory modality or mixture of modalities – visual, auditory, olfactory, somatosensory, millimeter-wave radar, infrared, chemical sensors used by insects, and on to an infinity of possibilities. By contrast, any realizable neural network will be finite in extent, will have a finite set of sensors, and at any finite stage of learning it cannot be expected to have absorbed all knowledge regardless of its representational power. But the semantic model is, after all, an abstract mathematical model, not a simulation of the neural networks of known organisms or other network-like entities. We can scale the ontology down to a domain of interest by using a subcategory of **Concept** – a collection of its objects and morphisms that is closed

under composition. This alternative avoids theories that are clearly not relevant to the analysis at hand, such as those involving sensorial modalities that are not present, but does not address the main difficulties. Fortunately, as shown in Section 8, a complex of functors $M : \mathbf{Concept} \longrightarrow \mathbf{N}_{A,w}, \ldots$ and natural transformations $\alpha : M \longrightarrow M', \ldots$ provides a convenient means of avoiding the difficulties with a representation of all knowledge: That which is not represented can be superimposed onto certain neural structures, effectively "compressing it out" of the representation. In this context, the use of natural transformations leads to the mathematical design principle we call *knowledge coherence*: the ability of multiple knowledge representations (functors) to act as one. Before discussing this and other design principles associated with structural mappings, let us gain experience with those suggested by the notion of commutative diagrams, which are preserved by functors and which lie at the heart of natural transformations.

7.1. *Commutative diagrams in a feedforward network*

The fact that commutative diagrams are preserved by functors, and that they play a major role in the definitions of limits and colimits, suggests that they are fundamental in the functorial representation of an ontology. To explore an example, consider a category $\mathbf{N}_{A,w}$ representing a feedforward network, a multi-layer perceptron (MLP) (McClelland et al., 1986), say, with weight state w. Let the transition function Φ_A represent a single iteration of the perceptron algorithm. Recall that an output value ξ_i for a node p_i having the property $\xi_i = \phi_i(\theta_i) \in \eta$ signifies an instance of the neural object (p_i,η).

Figure 8 shows a part of the MLP network, with selected nodes from three layers and selected connections between them. For illustrative purposes, the nodes are arbitrarily labeled p_1, p_2 and p_3 in the next layer forward, and p_4 forwardmost. The connections are also arbitrarily labeled as, for example, c_1 (from p_1 to p_2). By definition, there is a morphism corresponding to the path set $\overline{\Gamma_1}_w$, where Γ_1 is a singleton whose member γ_1 has only the single connection c_1, that is, $\gamma_1 = [(p_1, \eta), c_1, (p_2, \eta')]$. In fact, since the architecture is an MLP network, γ_1 is the only possible member of $\overline{\Gamma_1}_w$.

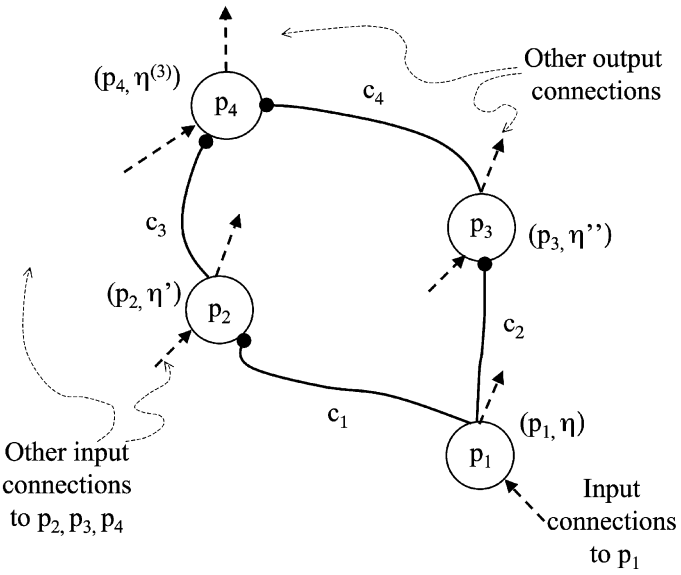


Figure 8. Two connection paths, possibly (but not necessarily) corresponding to a commutative diagram.

Denote the morphism associated with (Γ_1, w) by $m_1 : (p_1, \eta) \longrightarrow (p_2, \eta')$. The instances of m_1 are all combinations $(\theta, e) \in \Theta_A \times E_A$ of network activation states and network input patterns from the set $U_{\Gamma_1,w}$. Since $\mathbf{o_S}(\Gamma_1) = (p_1, \eta)$ and $\mathbf{o_T}(\Gamma_1) = (p_2, \eta')$, these are also instances of both (p_1,η) and (p_2, η') (that is, $\xi_1 = \phi_1(\theta_1) \in \eta$, $\xi_2 = \phi_2(\theta_2) \in \eta'$. This condition also holds in the final weight state ψ of any transition associated with m_1, with $(v, \psi) = \Phi_A(w, \theta, e)$.[4]

Similarly, having chosen the intervals indicated, the paths corresponding to connections c_2, c_3, and c_4 are $\gamma_2 = [(p_1, \eta), c_2, (p_3, \eta'')]$, $\gamma_3 = [(p_2, \eta'), c_3, (p_4, \eta^{(3)})]$, and $\gamma_4 = [(p_3, \eta''), c_4, (p_4, \eta^{(3)})]$. As with Γ_1, the path sets Γ_2, Γ_3, Γ_4 have solely the following members: $\gamma_2 \in \Gamma_2$, $\gamma_3 \in \Gamma_3$, $\gamma_4 \in \Gamma_4$. Corresponding to these, there are sets $U_{\Gamma_2,w}$, $U_{\Gamma_3,w}$, and $U_{\Gamma_4,w}$ whose members are the initial network activation and input combinations $(\theta, e) \in \Theta_A \times E_A$ that result in outputs $\xi_1 \in \eta$, $\xi_3 \in \eta''$ for the source and target nodes simultaneously for $\gamma_2, \xi_2 \in \eta', \xi_4 \in \eta^{(3)}$ for γ_3, and $\xi_3 \in \eta'', \xi_4 \in \eta^{(3)}$ for γ_4. This describes the morphisms $m_1 : (p_1, \eta) \longrightarrow (p_2, \eta')$, $m_2 : (p_1, \eta) \longrightarrow (p_3, \eta'')$, $m_3 : (p_2, \eta') \longrightarrow (p_4, \eta^{(3)})$, $m_4 : (p_3, \eta'') \longrightarrow (p_4, \eta^{(3)})$ associated with the connections c_1, c_2, c_3, c_4.

Now let us determine the requirement for the two paths from p_1 to p_4 through the array of connections c_1, c_2, c_3, c_4 to serve as a path set for a commutative diagram. Concatenating paths, we have paths $\gamma_5 = \gamma_1;\gamma_3$ and $\gamma_6 = \gamma_2;\gamma_4$, two separate paths having the same source and target. The path sets $\Gamma_5 = \{\gamma_5\}$ and $\Gamma_6 = \{\gamma_6\}$ have the property that $U_{\Gamma_5,w} = U_{\Gamma_3,w} \cap U_{\Gamma_1,w}$, $U_{\Gamma_6,w} = U_{\Gamma_4,w} \cap U_{\Gamma_2,w}$. This is the basis for the compositions $m_5 = m_3 \circ m_1$ and $m_6 = m_4 \circ m_2$. These are two morphisms with the same domain and codomain, $m_5 : (p_1, \eta) \longrightarrow (p_4, \eta^{(3)})$ and $m_6 : (p_1, \eta) \longrightarrow (p_4, \eta^{(3)})$. Now, were it the case that $U_{\Gamma_5,w} = U_{\Gamma_6,w}$, then they would be one and the same morphism $m_7 : (p_1, \eta) \longrightarrow (p_4, \eta^{(3)})$ (see Figure 9). In general, however, this will not be the case; because of the inputs to the intermediate nodes p_2 and p_3 through other connections emanating from nodes in prior layers (indicated by "Other input connections" in Figure 8), it can happen that some instances of m_5 are not instances of m_6 because they are instances of (p_2, η') but not (p_3, η''); conversely, not all instances of m_6 need be instances of m_5. Therefore, the diagram defined by the morphisms m_1, m_2, m_3, and m_4 need not be a commutative diagram, since the compositions $m_3 \circ m_1$ and $m_4 \circ m_2$ need not be equal.

This highlights an important fact: A set of paths having the same source and target objects can be involved in several different

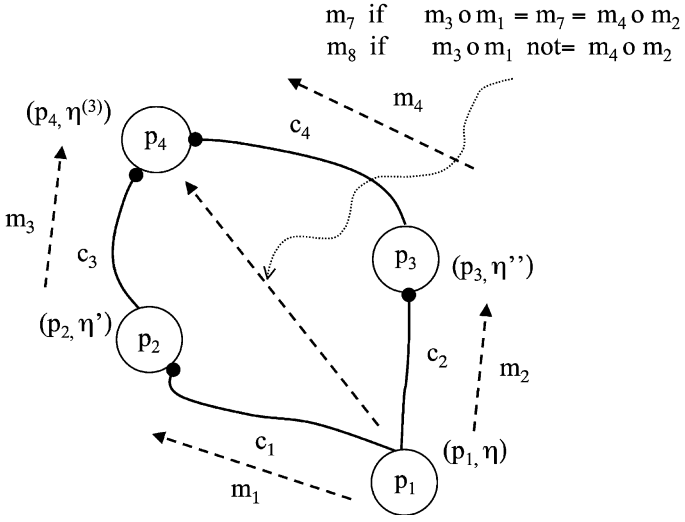


Figure 9. Two connection paths representing two morphism compositions that yield the same result correspond to a commutative diagram.

morphisms and, in fact, can be involved in several different diagrams. Further, some of these diagrams can be commutative while others are not. A morphism is uniquely defined by a path set Γ and weight state w (because the pair (Γ, w) has a unique instance set $U_{\Gamma,w}$ associated with it), but Γ can also be involved in separate morphisms defined by larger path sets that contain Γ. In this vein, although the diamond-shaped diagram defined by m_1, m_2, m_3, and m_4 need not be commutative, there is a single morphism associated with the larger path set consisting of the two paths through the connections c_1, c_2, c_3, and c_4. To see this, let Γ_8 be the union $\Gamma_8 = \Gamma_5 \cup \Gamma_6 = \{\gamma_5, \gamma_6\}$. The instances of Γ_8 lie in the intersection of the instance sets of the two path-sets in the union, $U_{\Gamma_8,w} = U_{\Gamma_5,w} \cap U_{\Gamma_6,w}$. This defines a morphism $m_8 : (p_1, \eta) \longrightarrow (p_4, \eta^{(3)})$ uniquely associated with the pair (Γ_8, w). In case the intersection is empty, $U_{\Gamma_5,w} \cap U_{\Gamma_6,w} = \emptyset$, m_8 is referred to as a vacuous morphism, having no instances. In any case, it is true that $U_{\Gamma_8,w} \subseteq U_{\Gamma_5,w}$ and $U_{\Gamma_8,w} \subseteq U_{\Gamma_6,w}$. If the subset relationship is proper in either case, the diagram does not commute. If the subset relationships are actually equalities, however, the diagram does commute, in which case $\overline{\Gamma_{5w}} = \Gamma_8 = \overline{\Gamma_{6w}}$.

This example shows that distinguishing between a diagram and a commutative diagram requires knowing whether or not concatenations of successive path sets along alternate routes in a neural network having common domain and codomain objects from the associated neural category have the same set of instances and, hence, are equivalent. Performing this kind of analysis in full is not possible in general, but the general idea can be used in analysis and design nevertheless. It can be used to either anticipate or derive useful network structures for concept and morphism representations. Explorations with this theory are at a very early stage, and experience has shown that it becomes easier to use the more we apply it. The following section describes a design principle which, when present in a neural design, ensures commutativity of the relevant diagrams without having to perform an exhaustive analysis of the instances of the path sets involved.

7.2. *Model-space morphisms imply a major design principle*

During an analysis or design exercise, we want to determine if our mathematically defined ontology can be correctly represented, at

least in part, in an architecture A in some weight state w. For this, we apply design principles implied by the presence of functors M : **Concept** $\longrightarrow \mathbf{N}_{A,w}$. Previously, we discussed one design principle: the more abstract concepts are derived as limits, and more specific ones as colimits. Since these are defined in part by commutative diagrams, and functors preserve commutative diagrams, it is important to identify situations in neural architectures under which they can exist. We now introduce a design principle that can be applied to ensure the presence of commutative diagrams.

Recall the reversal of direction between concept morphisms $s : T \longrightarrow T'$ and their model-space morphism counterparts $\mathtt{Mod(s)} : \mathtt{Mod(T')} \longrightarrow \mathtt{Mod(T)}$. Let Γ be a set of signal paths with common source and target whose instances (θ, e) define a morphism of $\mathbf{N}_{A,w}$, $m : (p_i, \eta) \longrightarrow (p_j, \eta')$. Suppose that this morphism is the image of a concept morphism via a functor M : **Concept** $\longrightarrow$ $\mathbf{N}_{A,w}$. That is, $m : (p_i, \eta) \longrightarrow (p_j, \eta') = M\ (s : T \longrightarrow T')$, or for brevity, $m = M(s)$, with $(p_i,\eta) = M(T)$ and $(p_j, \eta') = M(T')$. We argue that it is reasonable to associate an instance (θ, e) with the presence of a model for some concept whose functor image in $\mathbf{N}_{A,\ w}$ shares that instance. Applying this notion to the presence of the model-space morphism $\mathtt{Mod(s)}$: $Mod(T') \longrightarrow \mathtt{Mod}(T)$ yields the requirement that any instance (θ', e') of (p_j, η') associated with a model of T' must have a corresponding instance (θ, e) of (p_i, η), which is then associated with a model of T. Since our definitions for neural objects and morphisms are based upon the identification of persistent levels of activity in the appropriate nodes during a network transition $(v, \psi) = \Phi_A(w, \theta, e)$, formalizing a notion of simultaneity in the activations of the objects involved in a morphism, we have $(\theta', e') = (\theta, e)$.

Now, since the path set Γ for m can have many intermediate objects, and the activation of some of these could be suppressed by inhibitory inputs, the source and target nodes of Γ may not always satisfy the model-space morphism requirement. We argue that when this can happen, it signifies that (p_j, η'), the codomain of m, is the image of more than solely the concept T': It also represents a concept T'' similar to T' except that some part of $s(T)$, while a part of T', is not a part of T''.

Hence, a model of the concept T is not present and therefore the concept morphism s, hence, its associated model-space morphism $\mathtt{Mod}(s)$, does not apply to this instance. That this multiplicity of

concept representations can, in fact, occur at a single node is made possible by the fact that functors are many-to-one mappings; therefore, many objects T of **Concept** have the same image object in $M(T)$ in $\mathbf{N}_{A,w}$. Similarly, many concept morphisms can share an image morphism; when this happens, their domains and codomains must also share the domain and codomain objects of the shared image morphism.

Ensuring that the model-space morphism principle is applied correctly is difficult in a purely feedforward architecture. Since there are no reciprocal paths for feedback, there is no generally applicable means of controlling the "feedback" of instances from codomain to domain object for the appropriate morphisms. Thus, separating cases caused by the concept compression just discussed from outright representational error is at the very least a challenging task. This suggests a need for reciprocal connections to ensure the correct representation of concept morphisms. Designing an architecture with this in mind also simplifies the task of ensuring correctness of representation. Activated by a target node p_j as in our example, reciprocal connections can ensure that an instance of a neural morphism $m' : (p_j, \eta') \longrightarrow (p_i, \eta)$ in the opposite direction to $M(s)$ is present, and this morphism can serve as the carrier of the corresponding model-space morphism $\mathtt{Mod}(s) : \mathtt{Mod}(T') \longrightarrow \mathtt{Mod}(T)$, as shown in Figure 10.

7.3. *Working with limits and colimits*

The learning of more specific concepts using pre-existing concept representations is suggested by the colimit construction in Section 3, illustrated in Figure 6. The learning of more abstract concepts using pre-existing concept representations is suggested by the limit example in Section 3. To form either representation via long-term adaptation, a neural network must transition from a weight state containing a representation of the base diagram to one that yields the defining diagram. In addition, the model-space morphism requirement discussed in Section 7.2 must be satisfied. However, the fact that the latter requirement can be applied as a design principle makes it an asset rather than a liability.

As an example, consider the learning of a hypothetical limit by a neural network that has the appropriate connection structure and learning algorithm for the following scenario to take place: A

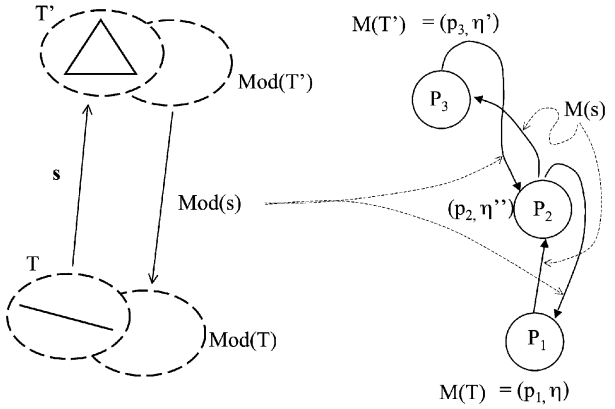


Figure 10. Reciprocal connections implement a model-space morphism.

sensor input that corresponds to a particular visual shape occurs in several input patterns generated, for example, by a pixel array as in Figure 2. Each pattern has the shape in a different location in the image space. Each time, through strengthening or weakening of the appropriate connections, the neural network has formed a separate representation of the shape accompanied by its location. This process results in the formation of colimits because the appropriate diagrams appear in the network in the form of objects and morphisms, represented by nodes producing outputs in the appropriate intervals along the paths defining the diagram morphisms. New morphisms, the colimit leg morphisms, arise by connection-weight adaptation during a neural transition $(v, \psi) = \Phi_A(w, \theta, e)$. The colimit concepts have the separate representations S_i as in the example of Section 3.4. However, they all, implicitly, can be thought of as a single shape concept R enriched with information about a specific spatial location, expressed as a term t_i, which in the example is $\texttt{spatial}(\texttt{L}i) = i$; we can express this structure in the theory as $S_i = R[t_i]$, where the term t_i is now a Skolemization of the term `(exists n:Num) (spatial (L) = n)` from R.

Each time the network receives input showing the visual shape at a new location i, it forms a new location-dependent colimit S_i. We would describe this representation as $R[t_i]$ if we were aware of the specific shape and location. The network forms this representation autonomously in a different way, however, through weight adaptation in paths leading from the component neural object carrier nodes in a diagram to an as-yet-unused node p_i.

Fortunately, our network design has provided that, also each time, the weight of a connection leading from the currently stimulated p_i to some fixed node p_L is strengthened. The new weight values are such that future activations of p_L help p_i become active within some interval $\eta^{(i)}$ representing a concept colimit provided that it receives other excitatory inputs from nodes in the appropriate intervals representing the concepts in a defining diagram. On the other hand, the new weight in a reciprocal $(p_i \rightarrow p_L)$ connection to the feedforward $(p_L \rightarrow p_i)$ path ensure that the activation of p_i will always activate p_L within an appropriate interval η regardless of any other sources of input the latter may have. An example of this scenario is that of Figure 10, with p_1 in the role of p_L, p_3 in the role of p_i, and reciprocal signal paths as indicated. When more than one input location for the visual shape has been processed, (p_L,η) will represent R alone. We now have the limit concept R and the leg morphisms $s_i : R \longrightarrow S_i$ of a cone for the discrete base diagram consisting of the concepts S_i, functorially mapped to the neural-category cone consisting of an apical object (p_L,η) and objects $(p_i, \eta^{(i)})$ together with neural morphisms $m_i : (p_L, \eta) \longrightarrow (p_i, \eta^{(i)})$. The model-space morphisms $\mathtt{Mod}(s) : \mathtt{Mod}(S_i) \longrightarrow \mathtt{Mod}(R)$ are represented by the reciprocal paths, which enforce the required property that an instance of any of the objects $(p_i, \eta^{(i)})$ is also an instance of (p_L, η).

We can generalize upon this argument and state the following rules for neural activity involving representations of limits and colimits. In a category $\mathbf{N}_{A,w}$, an instance of any one of its base diagram objects is also an instance of a limit-object node when the defining diagram is the functorial image of a concept limit defining diagram. This is the function of the paths reciprocal to those representing the limit cone leg morphisms. The paths in the opposite direction, representing the concept cone leg morphisms themselves, act differently: An instance of the limit object need not bring about the activation of its base diagram objects; their carrier nodes are merely given a boost, favoring them in any competition that may arise with nodes representing different visual shapes (enacted through inhibitory interconnections). This provides the network with a means for identifying which visual shape is relevant to the current episode of network activity regardless of its location in an input image, a potentially useful function in a vision system. For colimits, the rules are reversed just as the concept morphisms are

reversed. When a colimit defining diagram in $\mathbf{N}_{A,w}$ is a functorial image of a concept colimit defining diagram, an instance of the colimit object is also an instance of its base diagram objects given the reciprocal model-space morphism connections. An instance of one of its base diagram objects, on the other hand, need not be an instance of the colimit object although it does give it a boost, favoring it in any competition among the carrier nodes for colimit objects.

In general, the neural representations of colimits and limits can work together during successive stages of learning to form successively more complete representations of an ontology. For example, multiple colimits can be formed having a common shape representation, such as multiple, location-dependent representations. Next, a limit representation can be formed to derive an invariant shape representation. This provides a formalization for analyzing the "filling out" of the representation of an ontology by proceeding in both directions, starting with concepts at or near the sensor level. These ideas can be applied to extend the neural network representations in Riesenhuber and Poggio (1999).

8. NATURALITY: KNOWLEDGE COHERENCE ACROSS A MULTI-REGION NETWORK

Consider a multi-regional neural network, having several sensors with each sensor providing input to a region of the network. There can be other regions, such as association regions that unify the processing from two or more sensors, regions whose main function is for motor control (say, for an organism or for an autonomous vehicle controlled by the network), and regions for cognitive functions such as situation assessment and planning. The semantic theory specifies that at any stage of learning there is a system of functors: Each functor maps **Concept** into a part of the neural category $\mathbf{N}_{A,w}$ that models a subregion of the architecture A at the stage of learning represented by the weight vector w.

For example, suppose sensors S_1 and S_2 are available. Separate functors M_1 and M_2 can be used to represent the same state of learning in an architecture, but restricting concept implementations to a single sensor in each case. Thus, each concept and morphism are represented twice – once for sensor S_1 and once for sensor S_2. Each functor mathematically represents the category **Concept** in a

separate region of the network having a specialized function. Since functors can be many-to-one on both objects and morphisms, concepts T, T',... that do not relate to the function of a subregion, S_1, say, are "compressed out" by superimposition onto a common object, $M_1(T) = M_1(T') = \ldots$, and similarly for morphisms. The overriding question for this multi-regional model of network semantics is the following: How are the functions performed by the separate regions to be unified? They are described by the concepts of the hierarchy, but these are represented differently in each region, often with many compressed onto the same object of $\mathbf{N}_{A,w}$. In particular, how can the two sensors S_1 and S_2 be exploited to acquire a unified representation of their simultaneous inputs when each sensor captures only its aspect of them? *The answer lies in natural transformations* $\alpha^1 : M_1 \longrightarrow M_3$ and $\alpha^2 : M_2 \longrightarrow M_3$ from M_1 and M_2 to a third functor M_3, as shown in Figure 11. The natural transformations α^1 and α^2 are represented in $\mathbf{N}_{A,\ w}$ by neural morphisms separate from those that represent concept and model-space morphisms; unlike those representing concept and model-space morphisms, the natural transformation morphisms connect the separate regions.

In the sensor example, the M_3 region has no sensor. Instead, it serves as an association region. The semantic theory makes the association function explicit: Each triple of image objects $M_i(T_\mu)(i = 1, 2, 3)$ for a concept T_μ is connected by two morphisms $\alpha^i_{T_\mu} : M_i(T_\mu) \longrightarrow M_3(T_\mu)(i = 1, 2)$. Now consider a **Concept** morphism $s_\alpha : T_\mu \longrightarrow T_\nu$. The functorial images $M_i(s_\alpha) : M_i(T_\mu)$

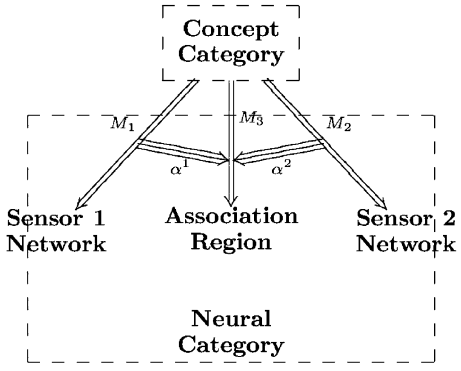


Figure 11. Functors map the hierarchy of a concept category to multiple regions. Natural transformations represent coherent interconnections between hierarchy representations.

$\longrightarrow M_i(T_\nu)(i = 1, 2, 3)$ preserve that relationship. By composition with the appropriate natural transformation components, there are two morphisms available from the sensor-S_1-based representation $M_1(T_\mu)$ of T_μ to the desired fused, dual-sensor representation $M_3(T_\nu)$ of T_ν: the compositions $\alpha^1_{T_\nu} \circ M_1(s_\alpha)$ and $M_3(s_\alpha) \circ \alpha^1_{T_\mu}$. Each of these morphisms has its associated connection paths. For knowledge coherence, however, we want the two sets of connection paths to be associated with a single morphism from $M_1(T_\mu)$ to $M_3(T_\nu)$, that is, $\alpha^1_{T_\nu} \circ M_1(s_\alpha) = M_3(s_\alpha) \circ \alpha^1_{T_\mu}$. But this is just the defining requirement of the natural transformation α^1. The same holds for α^2, corresponding to the unification of the sensor S_2 region with the association region.

9. CONCLUSION

An ontology and its incremental representation through adaptation in neural network architectures can be formalized in category theory. The categorical constructs used to model the representation of concepts and their relationships suggest architectural structures and their properties. The result is a set of principles for architectural design that apply to learning and to the combination of information from multiple sensors in a multi-region architecture. We have given examples to illustrate the application of these ideas in neural network analysis and design. By providing a mathematical vehicle for associating a hierarchy of concepts with a multi-regional neural architecture and explicating the incremental learning of both more abstract and more specific concepts with re-use of existing conceptual knowledge, the theory would seem to have a natural role as a fundamental theory for exploring knowledge representation in neural networks.

ACKNOWLEDGEMENTS

The authors would like to thank Chaouki Abdallah and Tim Goldsmith (University of New Mexico), Bob Marks (Baylor University), Mike Anderson and Keith Williamson (The Boeing Company), Andree Ehresmann (Universite de Picardi, Amiens, Fr.), Joseph

Goguen (University of California at San Diego), and Stephen Vickers (University of Birmingham, UK) for their helpful comments and suggestions during the development of the semantic theory.

NOTES

[1] The variable names are arbitrary as long as they are correctly typed and are renamed where necessary to avoid confusion, a standard substitution rule.
[2] Unlike the category **Concept**, the category **Set** has limits for all its discrete diagrams. These are the familiar Cartesian products, and the projections are the usual projections onto each component set.
[3] Real-valued quantities are used here for simplicity; others, such as complex numbers, can be used instead. Sets of the appropriate form are then used in place of real intervals.
[4] Note that it matters not whether the weight w_1 of c_1 is $w_1 > 0$ or $w_1 < 0$, that is, whether c_1 is an excitatory or an inhibitory connection. It is only required that w_1 (and also v_1) be nonzero and that the nodes along the path γ_1 (p_1 and p_2) are generating outputs within their specified intervals η, η'.

REFERENCES

Adamek, J., H. Herrlich and G. Strecker: 1990, *Abstract and Concrete Categories: The Joy of Cats*, Cambridge University Press.
Andrews, P. B.: 1986, *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof*, Academic Press, Inc.
Andrews, R., J. Diederich and A. B. Tickle: 1995, 'Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks', *Knowledge-Based Systems* **8**(6), 373–389.
Arbib, M. A.: 1987, *Brains, Machines, and Mathematics*, Springer: Berlin.
Bartfai, G.: 1994, 'Hierarchical Clustering with ART Neural Networks', in *Proceedings of the International Conference on Neural Networks*, Orlando, FL, June 28–July 2, 1994.
Bennet, M. K.: 1995, *Affine and Projective Geometry*, John Wiley and Sons.
Carpenter, G. A. and A.-H. Tan: 1995, 'Rule Extraction: From Neural Architecture to Symbolic Representation', *Connection Science* **7**, 3–27.
Craven, M. W. and J. W. Shavlik: 1993, 'Learning Symbolic Rules using Artificial Neural Networks', in *Proceedings of the 10th International Machine Learning Conference*, Amherst, MA, Morgan Kaufmann.
Crole, R. L.: 1993, *Categories for Types*, Cambridge University Press.
Damasio, A.: 1989, 'Time-Locked Multiregional Retroactivation: A Systems-Level Proposal for the Neural Substrates of Recall and Recognition', *Cognition* **33**, 25–62.
Ehresmann, A. C. and J.-P. Vanbremeersch: 1997, 'Information Processing and Symmetry-Breaking in Memory Evolutive Systems', *BioSystems* **43**, 25–40.
Fu, L. M.: 1992, 'A Neural Network for Learning Rule-Based Systems', in *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, MD.

Goguen, J. A. and R. M. Burstall: 1992, 'Institutions: Abstract Model Theory for Specification and Programming', *Journal of the Association for Computing Machinery* **39**(1), 95–146.

Gruber, T. and G. Olsen: 1994, 'An Ontology for Engineering Mathematics', in *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kauffman.

Gust, H. and K.-U. Kühnberger: 2005, 'Learning Symbolic Inferences with Neural Networks', in *Proceedings of the XXVII Annual Conference of the Cognitive Science Society (CogSci2005), Stresa, Italy*, Cognitive Science Society, ed. offices University of Indiana, Bloomington, IN.

Healy, M. J.: 1999a, 'A Topological Semantics for Rule Extraction with Neural Networks', *Connection Science* **11**(1), 91–113.

Healy, M. J.: 1999b, 'Colimits in Memory: Category Theory and Neural Systems', in J. S. Boswell (ed.), *Proceedings of IJCNN'99: International Joint Conference on Neural Networks*, Washington, DC: IEEE Press.

Healy, M. J.: 2000, 'Category Theory Applied to Neural Modeling and Graphical Representations', in M. Gori, S. -I. Amari, C. L. Giles and V. Piuri (eds.), *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks: IJCNN2000*, Como, Italy: IEEE Computer Society Press.

Healy, M. J. and T. P. Caudell: 1997, 'Acquiring Rule Sets as a Product of Learning in a Logical Neural Architecture', *IEEE Transactions on Neural Networks* **8**(3), 461–474.

Healy, M. J. and T. P. Caudell: 2001, 'A Categorical Semantic Analysis of ART Architectures', in *IJCNN'01:International Joint Conference on Neural Networks*, Washington, DC, IEEE Press.

Healy, M. J. and T. P. Caudell: 2002, 'Aphasic Compressed Representations: A Functorial Semantic Design Principle for Coupled ART Networks', in *The 2002 International Joint Conference on Neural Networks (IJCNN'02)*, Honolulu, (CD-ROM Proceedings), IEEE Press.

Healy, M. J. and T. P. Caudell: 2003, 'From Categorical Semantics to Neural Network Design', in *The Proceedings of the IJCNN 2003 International Joint Conference on Neural Networks*, Portland, OR, 2003 (CD-ROM Proceedings), IEEE Press.

Healy, M. J. and T. P. Caudell: 2004, Neural Networks, Knowledge, and Cognition: A Mathematical Semantic Model Based upon Category Theory. Technical Report EECE-TR-04-020, Department of Electrical and Computer Engineering, University of New Mexico.

Healy, M. and K. Williamson: 2000, 'Applying Category Theory to Derive Engineering Software from Encoded Knowledge', in *Proceedings of the Algebraic Methodology and Software Technology 8th International Conference, AMAST 2000*, Iowa City, IA. Published in Lecture Notes in Computer Science, Springer-Verlag.

Hebb, D. O.: 1949, *The Organization of Behavior*, John Wiley and Sons: New York.

Heileman, G. L., M. Georgiopoulos, M. J. Healy and S. J. Verzi: 1997, 'The Generalization Capabilities of ARTMAP', in *Proceedings of the International Joint Conference on Neural Networks (ICNN97)*, Houston, TX.

Hirsch, J., D. R. Moreno and K. H. S. Kim: 2001, 'Interconnected Large-Scale Systems for Three Fundamental Cognitive Tasks Revealed by Functional mri', *Journal of Cognitive Neuroscience* **13**(3), 389–405.

Jullig, R. and Y. V. Srinivas: 1993, 'Diagrams for Software Synthesis', in *Proceedings of KBSE '93: The Eighth Knowledge-Based Software Engineering Conference*, IEEE Computer Society Press.

Kasabov, N. K.: 1996, 'Adaptable Neuro Production Systems', *Neurocomputing* **13**, 95–117.

Kelley, W. M., C. N. Macrae, C. L. Wyland, S. Caglar, S. Inati and T. F. Heatherton: 2002, 'Finding the Self? An Event-Related fmri Study', *Journal of Cognitive Neuroscience* **14**(5), 785–794.

Lawvere, F. W. and S. H. Schanuel: 1997, *Conceptual Mathematics: A First Introduction to Categories*, Cambridge University Press.

Mac Lane, S.: 1971, *Categories for the Working Mathematician*, Springer. This is the standard reference for mathematicians, written by one of the two co-discoverors of category theory (S. Eilenberg being the other).

McClelland, J. L., D. E. Rumelhart and G. E. Hinton: 1986, 'The Appeal of Parallel Distributed Processing', in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, The MIT Press, pp. 3–44. This volume contains an exposition on bias nodes as a neural mechanism for varying the thresholds of a layer of nodes.

McCullough, W. S. and W. Pitts: 1943, 'A Logical Calculus of the Ideas Immanent in Nervous Activity', *Bulletin of Mathematical Biophysics* **5**, 115–133.

Meseguer, J.: 1989, 'General logics', in *Logic Colloquium '87*, Science Publishers B. V. (North-Holland), pp. 275–329.

Mitra, S. and S. K. Pal: 1995, 'Fuzzy Multi-Layer Perceptron, Inferencing and Rule Generation', *IEEE Transactions on Neural Networks* **6**, 51–63.

Otto, I., P. Grandguillaume, L. Boutkhil, Y. Burnod and E. Guigon: 1992, 'Direct and Indirect Cooperation between Temporal and Parietal Networks for Invariant Visual Recognition', *Journal of Cognitive Neuroscience* **4**(1), 35–57.

Perry, C.: 1999, 'Testing a Computational Account of Category-Specific Deficits', *Journal of Cognitive Neuroscience* **11**(3), 312–320.

Pierce, B. C.: 1991, *Basic Category Theory for Computer Scientists*, MIT Press.

Pinkas, G.: 1995, 'Reasoning, Nonmonotonicity and Learning in Connectionist Networks that Capture Propositional Knowledge', *Artificial Intelligence* **77**, 203–247.

Rao, S. C., G. Ranier and E. K. Miller: 1997, 'Integration of What and Where in the Primate Prefrontal Cortex', *Science* **276**, 821–824.

Riesenhuber, M. and T. Poggio: 1999, 'Are Cortical Models Really Bound by the Binding Problem?', *Neuron* **24**, 87–93.

Rosen, R.: 1958a, 'A Relational Theory of Biological Systems', *Bulletin of Mathematical Biophysics* **20**, 245–260.

Rosen, R.: 1958b, 'The Representation of Biological Systems from the Standpoint of the Theory of Categories', *Bulletin of Mathematical Biophysics* **20**, 317–341.

Sima, J.: 1995, 'Neural Expert Systems', *Neural Networks* **8**, 261–271.

Squire, L. R. and S. Zola-Morgan: 1991, 'The Medial Temporal Lobe Memory System', *Science* **253**, 1380–1385.

Srinivas, Y. V. and R. Jullig: 1995, 'Specware^{TM}: Formal Support for Composing Software', in *Proceedings of the Conference of Mathematics of Program Construction*.

Stoltenberg-Hansen, V., I. Lindstroem, and E. R. Griffor: 1994, *Mathematical Theory of Domains*, Cambridge University Press.

Tse, T. H.: 1991, *A Unifying Framework for Structured Analysis and Design Models: An Approach using Initial Algebra Semantics and Category Theory*, Cambridge University Press.

Vickers, S.: 1993, *Topology via Logic*, Cambridge University Press.

Wickelgren, I: 1997, 'Getting a Grip on Working Memory', *Science* **275**, 1580–1582.

Williamson, K. and M. Healy: 1997, 'Formally Specifying Engineering Design Rationale', in *Proceedings of the Automated Software Engineering Conference-1997*.

Williamson, K. and M. Healy: 2000, 'Deriving Engineering Software from Requirements', *Journal of Intelligent Manufacturing* **11**(1), 3–28.

Williamson, K., M. Healy and R. Barker: 2001, 'Industrial Applications of Software Synthesis via Category Theory-Case Studies using Specware^{TM}', *Automated Software Engineering* **8**(1), 7–30.