2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Knowledge Representation and Possible Worlds for Neural Networks

Michael J. Healy†§
§Corresponding author
13544 23rd Place NE
Seattle WA 98125
E-mail: mjhealy@ece.unm.edu

†Department of Electrical and Computer Engineering
‡Department of Computer Science
University of New Mexico
Albuquerque, New Mexico 87131

Thomas P. Caudell†‡
E-mail: tpc@ece.unm.edu

Abstract— **The semantics of neural networks can be analyzed mathematically as a distributed system of knowledge and as systems of possible worlds expressed in the knowledge. Learning in a neural network can be analyzed as an attempt to acquire a representation of knowledge. We express the knowledge system, systems of possible worlds, and neural architectures at different stages of learning as categories. Diagrammatic constructs express learning in terms of pre-existing knowledge representations. Functors express structure-preserving associations between the categories. This analysis provides a mathematical vehicle for understanding connectionist systems and yields design principles for advancing the state of the art.**

I. Introduction

A sizeable body of research has had the objective of explaining the semantics of neural computation, particularly with respect to learning ([2], [5], [9], [7], [3], [6], [1], [21], [23], [17], [24], [25], [12], [16], [11]). Much of this research is an attempt to decode the adapted connection-weight values as "if-then" rules that a network is supposed to have learned from its input stream. Formal-logic-like languages are often used to express rules unambiguously and sometimes with full mathematical rigor ([2], [11], [23]). Intuitively, the ability of a computational system to manipulate data in a systematic way is a manifestation of the knowledge represented in the system's structural design and operation. In the present paper, we describe a more comprehensive and wholly mathematical semantic model that fully explores this notion. Instead of a partial expression of the semantics of a neural network in terms of a single body of logical rule forms, we propose a hierarchical structure of interrelated knowledge modules, or concepts, expressed as formal logic theories. The concepts are symbolic descriptions of objects and events, observable or imagined, at any arbitrary level of generality or specificity. They are organized as a many-threaded hierarchy ordered from the abstract to the specific. In this context, the semantics of a neural network can be expressed as an evolving representation of a distributed system of concepts, many of them learned from data via weight adaptation. The concepts declare the meaning of the operations performed by each part of the system in terms of its inputs and outputs. Learning begins with a predetermined system of concepts at and near the sensor/actuator level that describe the most basic properties of inputs, outputs and internally-generated stimuli that are associated with the activation of the network. The neural network's learning algorithm effectively re-uses the sensor concepts in many ways in combination with the input data to form concepts not yet represented in the connection weights.

In the semantic theory, category theory provides a mathematical model of this process of knowledge representation within a neural network. Along with a mathematical system expressing the knowledge hierarchy, or ontology, the theory employs structure-preserving mappings to show how the knowledge is or can be represented (if at all) in a neural network's connection structure and weight array. Along with this, other structural mappings describe the possible worlds, or scenes, events, and situations, that the concepts describe. Taken together, the concept, possible world, and neural representations enable a comprehensive, mathematically rigorous analysis of the capabilities of neural networks in terms of their input environments and task objectives. An initial result of this new semantic model is a set of design principles [15] some of which have been applied to achieve improved performance in a neural network application [14].

II. Category Theory: A Brief Introduction

Introductions to category theory at varying levels of detail are available in [8], [18], [19], [22] and [15]. It is a theory of mathematical structure based upon the notion of an arrow, or *morphism*—a relationship between two *objects* in a *category*. A morphism $f\colon a \longrightarrow b$ has a *domain* object a and a *codomain* object b, and serves as a sort of directed relationship between a and b. In a category C, each pair of arrows $f : a \longrightarrow b$ and $g : b \longrightarrow c$ (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 is associative, that is, for three arrows of the form $f\colon a \longrightarrow b$, $g\colon b \longrightarrow c$ and $h\colon c \longrightarrow d$, the result of composing them is order-independent, with $h \circ (g \circ f) = (h \circ g) \circ f$. For each object a, there is an *identity morphism* $\mathtt{id}_a\colon a \longrightarrow a$ such that for any arrows $f\colon a \longrightarrow b$ and $g\colon b \longrightarrow a$, $\mathtt{id}_a \circ g = g$ and $f \circ \mathtt{id}_a = f$. A familiar example of a category is one called **Set**, which has sets as its objects, functions as its morphisms, and whose composition is just the composition of functions, $(g \circ f)(x) = g(f(x))$.

0-7803-9490-9/06/$20.00/©2006 IEEE

A diagram is a collection of objects and morphisms of C. In a *commutative diagram*, any two morphisms with the same domain and codomain, where at least one of the morphisms is the composition of two or more diagram morphisms, are equal. An *initial object*, where one exists in C, is an object i for which every object a of C is the codomain of a unique morphism $f: i \longrightarrow a$. A *terminal object* t has every object a of C as the domain of a unique morphism $f: a \longrightarrow t$. Every category C has an *opposite category* C^{op} obtained by reversing the arrows and the order of composition of C. That is, for each arrow $f : a \longrightarrow b$ of C there is an arrow $f^{\mathrm{op}} : b \longrightarrow a$ of C^{op}, and for any $g: b \longrightarrow c$ that is also an arrow of C, the composition $g \circ f : a \longrightarrow c$ in C is matched by a composition in C^{op} with order reversed, $f^{\mathrm{op}} \circ g^{\mathrm{op}} : c \longrightarrow a$. This category is associated with the *Principle of Duality*, by which a theorem in category theory yields an additional theorem through reversal of the arrows, the order of composition and the terms "initial" and "final", and the addition or deletion of the prefix "co".

These key notions are fundamental in the definition of *limits* and *colimits*. Colimits express mathematically the learning of more complex, specialized concepts through the re-use of simpler concepts already represented in the connection-weight memory of a neural network ([13], [15]). Here, we define limits, which express the learning of more abstract concepts; reversing the arrows and substituting "initial" for "terminal" and "cocone" for "cone" suffices to define colimits, the dual notion. Let Δ be a diagram in a category C as shown in Fig. 1, with objects a_1, a_2, a_3, a_4 and morphisms $f_1: a_1 \longrightarrow a_3$, $f_2: a_2 \longrightarrow a_3$, $f_3: a_1 \longrightarrow a_4$, $f_4: a_2 \longrightarrow a_4$. The diagram $\underline{\Delta}$ extends Δ to a commutative diagram by adding a *cone* K. The latter consists of an *apical object* b and morphisms $g_i: b \longrightarrow a_i \ \ (i = 1, \ldots, 4)$ such that $f_1 \circ g_1 = g_3 = f_2 \circ g_2$ and $f_3 \circ g_1 = g_4 = f_4 \circ g_2$, provided additional objects and morphisms satisfying these equations exist in C. Cones for Δ are the objects of a category $\mathbf{cone}_{\Delta}$ (whose morphisms are described in [15]). A *limit for the diagram* Δ is a terminal object K in $\mathbf{cone}_{\Delta}$, in which case $\underline{\Delta}$ is called the *defining diagram* for the limit and Δ is called its *base diagram*. A *discrete* diagram has objects a_i but no morphisms, in which case its limit is called a *product*. The dual notion is a colimit for a discrete diagram, called a *coproduct*.

The importance of category theory lies in its ability to formalize the notion that things that differ in substance can have an underlying similarity of "structural" form. A mapping between categories that preserves compositional structure, called a *functor*, formalizes this notion. A functor $F : C \longrightarrow D$ 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, and is such that (1) for each composition $g \circ_C f$ in C, $F(g \circ_C f) = F(g) \circ_D F(f)$, where $\circ_C$ and $\circ_D$ denote the respective compositions in C and D; (2) for each object a of C, $F(\mathtt{id}_a) = \mathtt{id}_{F(a)}$. It follows that F maps commutative diagrams of C to commutative diagrams in D. This means that any structural constraints expressed in C are translated into D. Finally, *natural transformations* are structure-preserving mappings between functors; although they are not discussed here, we mention them because of their fundamental role in the semantic model [15].

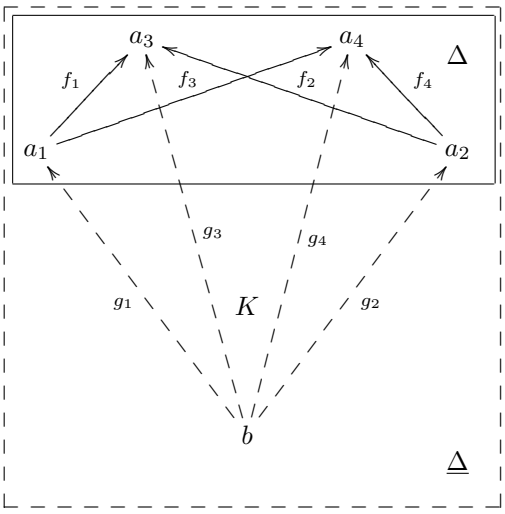


Fig. 1. **A limit for a diagram Δ.**

III. From Ontology to Neural Network

The category with which we begin a study of ontologies is one we call $\mathbf{Concept}$. For convenience, we express it as a category of theories in a formal logic ([10], [20]), and, hence, its objects (our concepts) are theories and its morphisms are theory morphisms. For illustration, our theories will be expressed in a first-order multi-sorted predicate calculus. A theory morphism $s: T \longrightarrow T'$ is a mapping of the syntax of a theory T into the syntax of a theory T', such that the axioms of T are mapped to axioms or theorems of T'. This category has colimits for all diagrams and limits for some. We shall show by example that a colimit can be used to derive a more complex, specialized theory from the theories and morphisms in its base diagram, along with morphisms (the cocone leg morphisms) by which it relates to them. Dually, a limit derives a more abstract theory along with morphisms that relate it to its base diagram objects.

The concept of a triangle can be derived as a colimit for a diagram involving concepts about points and lines (the alternative of defining triangles in terms of angles would complicate the example). We can start with a rather abstract concept, a theory that defines lines in terms of undefined quantities, or primitives, called points. The definition is expressed using a predicate `on` with two arguments, a point and a line, and is true just in case the point "lies on" the line (see [4] for a discussion of geometries based upon this definition).

```
Concept T1
  sorts Points, Lines
  const p1: Points
  const p2: Points
  const p3: Points
  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 sort symbols `Points` and `Lines` stand for the types of entities, points and lines, described by the theory. A multi-sorted logic such as this is a convenient alternative to an unsorted logic, in which the sorts would be replaced by expressions `point (x)` and `line (l)` using predicates `point` and `line`, and every formula would be required to incorporate expressions like this.

Notice that in addition to the definition of a line in terms of points, T_1 also contains constants representing three arbitrary points `p1`, `p2` and `p3`. Three more specialized concepts T_2, T_3 and T_4 share T_1 except with different names for the point constants. The latter concepts specialize T_1 by including a line constant and associating two of the point constants with it via the `on` predicate. The association is specified in an additional axiom (name omitted) which also states that the two point constants denote distinct points. For example:

```
Concept T2
  sorts Points, Lines
  const pa1: Points
  const pa2: Points
  const paext: Points
  const la: 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) ))
  on (pa1, la) and on (pa2, la)
   and (pa1 not= pa2)
end
```

Concepts T_3 and T_4 are identical, except with the names `pa1`, `pa2`, `paext`, `la` replaced with `pb1`, `pb2`, `pbext`, `lb` in T_3 and `pc1`, `pc2`, `pcext`, `lc` in T_4. A morphism $s_1 : T_1 \longrightarrow T_2$ maps the sort symbols `Points` and `Lines` and the `on` predicate symbol to the corresponding symbols in T_2, which happen to be identical. We reformulate all statements of T_1 by term replacement in accordance with the symbol mapping to form their image statements in T_2. As a consequence, the axiom of T_1 relating points to lines maps to itself as an axiom of T_2. The point constants `p1`, `p2` and `p3` map to the point constants `pa1`, `pa2` and `paext`. In T_2, `pa1` and `pa2` are associated with the line `la` via the `on` predicate, and `paext` is intended as a point "external to" `la`. Omitting the symbol maps for those symbols that are identical to their images, as is customary, morphism s_1 is as follows:

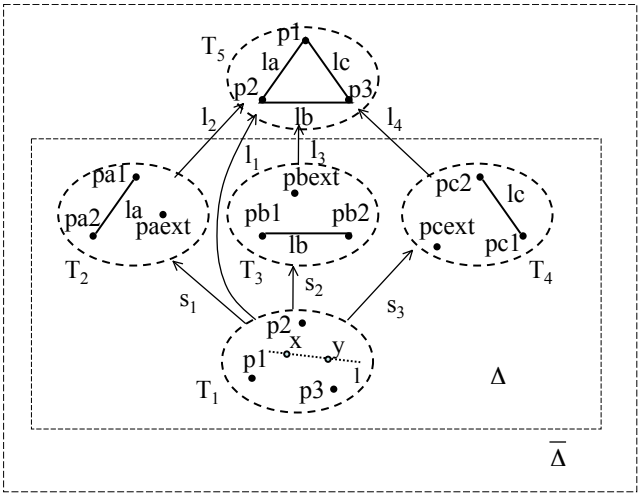


Fig. 2. **A pictorial illustration of the colimit base and defining diagrams Δ and $\overline{\Delta}$. The contents of the concepts involved are pictured along with the diagrammatic structure. Solid dots and lines signify point and line constants. Concept T_1 has no line constant, so it is shown containing a dashed line with two open dots, representing the axiom relating points and lines which is present in all the concepts.**

Morphism s_1 :

`p1`	$\mapsto$ `pa1`
`p2`	$\mapsto$ `pa2`
`p3`	$\mapsto$ `paext`

Morphisms $s_2 : T_1 \longrightarrow T_3$ and $s_3 : T_1 \longrightarrow T_4$ are similar to s_1 but with different point constant targets `pb1`, `pb2`, `pbext` and `pc1`, `pc2`, `pcext`:

Morphism s_2 :

`p1`	$\mapsto$ `pbext`
`p2`	$\mapsto$ `pb1`
`p3`	$\mapsto$ `pb2`

Morphism s_3 :

`p1`	$\mapsto$ `pc2`
`p2`	$\mapsto$ `pcext`
`p3`	$\mapsto$ `pc1`

In T_3, it is the images `pb1` of `p2` and `pb2` of `p3` that are associated with the line constant, `lb`, while the image `pbext` of `p1` is the "external" point. The associations are similarly reordered in T_4; the point-to-line associations in each concept can be seen by noticing which points are the targets of those of T_1 under the appropriate morphism and applying term substitution. A colimit for the diagram Δ with

objects T_1, T_2, T_3, T_4 and morphisms s_1, s_2, s_3 (which always exists in the category **Concept**) has a cocone as shown in Figure 2, with apical object T_5 and leg morphisms $\ell_1: T_1 \longrightarrow T_5$, $\ell_2: T_2 \longrightarrow T_5$, $\ell_3: T_3 \longrightarrow T_5$, and $\ell_4: T_4 \longrightarrow T_5$. With Δ as the base diagram, the defining diagram of the colimit, $\overline{\Delta}$, is commutative, with

$$\ell_1 = \ell_2 \circ s_1 = \ell_3 \circ s_2 = \ell_4 \circ s_3 . \qquad (1)$$

The resulting colimit apical object, T_5, is as follows:

```
Concept T5
  sorts Points, Lines
  const p1: Points
  const p2: Points
  const p3: Points
  const la: Lines
  const lb: Lines
  const lc: 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) ))
  on (p1, la) and on (p2, la) and
   (p1 not= p2)
  on (p2, lb) and on (p3, lb) and
   (p2 not= p3)
  on (p3, lc) and on (p1, lc) and
   (p3 not= p1)
end
```

Because the defining diagram $\overline{\Delta}$ of the colimit is commutative, the apical concept T_5 is a "blending" or "pasting together" of T_2, T_3 and T_4 along their common sub-concept T_1. That is, for the equality (1) to hold, separate symbols of T_2, T_3 and T_4 that are images of the same symbol of T_1 under the three diagram Δ morphisms s_1, s_2 and s_3 must merge into a single symbol in the colimit apical concept T_5. To make this clear, each symbol in T_5 that is a merging of symbols has been assigned the name of the T_1 symbol underlying the merging. Thus, symbols such as `Points`, `Lines` and `on` appear in T_5, and appear only once, since they are mapped to themselves by each of the morphisms s_1, s_2 and s_3. The point constants `p1`, `p2`, `p3` also appear. However, in T_5, each one represents a merging of two point constants from T_2, T_3 and T_4 and as a consequence appears in the definition of two different lines. In two of these concepts, the image of each single point constant appears in the definition of a line, but as a different point on a different line in each of the two concepts. In the third concept, it appears as an "external" point, not on the line named in that concept. For example, `p1` in T_1 is mapped to `pa1` in T_2 via s_1, to `pbext` in T_3 via s_2, and to `pc2` in T_4 via s_3. In T_5, therefore, it forms the point `p1` at the intersections of lines `la` and `lc`, and lies external to line `lb`. Because of the initiality of the colimit cocone, any other cocone for Δ is the codomain of a unique cocone morphism whose domain is the cocone containing T_5. Therefore, T_5 adds no extraneous information to that in Δ.

Because it is more complex, the colimit apical object is more specific, or more specialized, than any of the concepts in its base diagram. Because it expresses the "pasting together" of the base diagram concepts around their shared sub-concepts, it expresses the concept relationships (morphisms) as well as the concepts in its base diagram. Because of the Colimit Theorem, the calculation of concept colimits can be automated. For example, the apical object T_5 and leg morphisms ℓ_1, ℓ_2, ℓ_3, and ℓ_4 above can be derived automatically from the objects and morphisms of the base diagram, Δ. Other examples are given in [26] for an application of engineering software synthesis via category theory. Where they are available, limits can be calculated also, yielding less complex or abstract theories and their attendant morphisms. This facilitates the generalization of learned concepts to new contexts by extracting useful invariants from them. The ability to calculate concept colimits and limits suggests the ability to "flesh out" an ontology in an incremental fashion. This process can begin with a collection of concepts and morphisms describing the most basic properties of observable quantities and also any desired assumptions about the environment and the operation of a system within it. More specialized concepts and morphisms can be calculated as colimits, and more abstract ones as limits, through the re-use of concepts and morphisms already defined.

IV. Neural Categories and Functors

Each neural network architecture A, together with an array w of connection weight values, has an associated category $\mathbf{N}_{A,w}$. This category is a mathematical representation of the structure comprising the network connections, its current weights, and its potential activation states initiated by inputs given the weight array w. The objective in defining $\mathbf{N}_{A,w}$ is to determine to what extent a functor $M: \mathbf{Concept} \longrightarrow \mathbf{N}_{A,w}$ can be defined for a given pair (A, w) mapping concepts and their morphisms to the objects and morphisms of $\mathbf{N}_{A,w}$. The ability to perform this analysis brings mathematical rigor to the understanding of knowledge representations in neural networks.

An object of $\mathbf{N}_{A,w}$ is defined by a pair (p_i, η), where p_i is a node of A and η is a set of output values for p_i. The set η can be regarded as an interval of real values if the analysis at hand considers p_i to have a real-valued signal function. However, this is not a requirement of the semantic theory. For example, complex signal functions (and complex connection weights) can be used in an analysis and η can be a region in the complex plane. In any case, a morphism $m: (p_i, \eta) \longrightarrow (p_j, \eta')$ of $\mathbf{N}_{A,w}$ is defined by a set Γ of connection paths which share common source and target nodes p_i and p_j, together with the specified output sets η and η' along with a specified output set for each intermediate

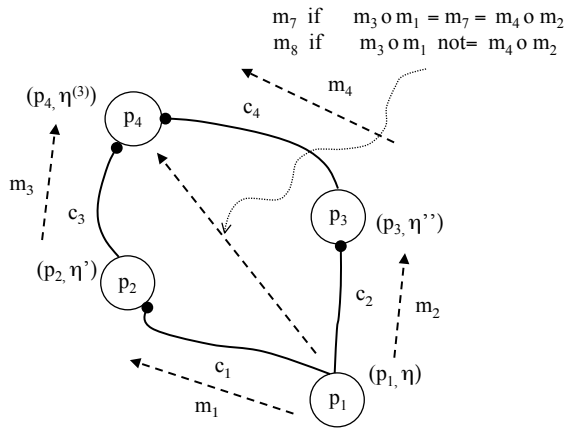


Fig. 3. **There are four neural morphisms m_1, m_2, m_3, m_4 defined by the single-connection paths c_1, c_2, c_3, c_4 between the neural objects $(p_1, \eta), (p_2, \eta'), (p_3, \eta''), (p_4, \eta^{(3)})$. The compositions $m_3 \circ m_1$ and $m_4 \circ m_2$ defined by the two signal paths from (p_1, η) to $(p_4, \eta^{(3)})$ along c_1, c_3 and c_2, c_4 through (p_2, η') and (p_3, η'') are morphisms between (p_1, η) and $(p_4, \eta^{(3)})$. Another morphism m_7 between the latter two objects is defined by the set containing both of these paths. But if the diagram formed by m_1, m_2, m_3, m_4 and the four objects commutes, there is only one morphism, m_8, between (p_1, η) and $(p_4, \eta^{(3)})$, since $m_3 \circ m_1 = m_8 = m_4 \circ m_2$.**

node in the paths of Γ. Therefore, a path can have many nodes and connections, each node having an assigned set $\eta, \eta', \eta^{(3)}, \eta^{(4)}, \ldots$. Because each node in a path has an assigned output set, we call it a *signal path* instead of a connection path. Figure 3 shows a path set containing two paths with source object (p_1, η), target object $(p_4, \eta^{(3)})$, and two connections per path through intermediate objects, (p_2, η') and (p_3, η''). Associated with each combination of path set Γ and weight array w is a set $U_{\Gamma,w}$ of ordered pairs (θ, e), where θ is an array of activation states for all network nodes at the time an input is received by A and e is the input. If the network activity resulting from the pair (θ, e) produces outputs varying within $\eta, \eta', \eta^{(3)}, \eta^{(4)}, \ldots$ by the nodes of Γ, it is called an *instance* of Γ. The set $U_{\Gamma,w}$ of instances together with Γ uniquely determine a morphism $m_{\Gamma,w}$; in terms of the previous name for the morphism $m\colon (p_i, \eta) \longrightarrow (p_j, \eta')$ defined by Γ, then, $m = m_{\Gamma,w}$. There can be many path sets $\Gamma', \Gamma'', \ldots$ with the same source and target objects *and* the same set of instances, where $U_{\Gamma,w} = U_{\Gamma',w} = U_{\Gamma'',w} = \ldots$. All define the same morphism, $m = m_{\Gamma,w} = m_{\Gamma',w} = m_{\Gamma'',w} = \ldots$, and, hence, their union also defines the same morphism. Finally, we define the instance set for an object (p, η), $U_{(p,\eta)}$, as the instance set for its identity morphism $\mathtt{id}_{(p,\eta)}$, which in turn is defined by the instances of the empty path with (p, η) as source and target; these instances, in turn, are simply those that result in p producing outputs varying within η.

Notice that in Figure 3, the two paths from (p_1, η) to $(p_4, \eta^{(3)})$ each consists of two connections through a third object. In fact, this suggests our definition of composition of morphisms for $\mathbf{N}_{A,w}$. In general, for two paths which meet end-to-end, one concatenates the end-to-end paths to obtain a single path, and forms the intersection of the two instance sets. Concatenating the single-connection paths γ_1 consisting of (p_1, η), c_1, (p_2, η') and defining a morphism m_1, and γ_3 consisting of (p_2, η'), c_3, $(p_4, \eta^{(3)})$ and defining a morphism m_3, yields a two-connection path γ_5, consisting of (p_1, η), c_1, (p_3, η''), c_3, $(p_4, \eta^{(3)})$ and defining a morphism m_5. The instance set of γ_5, and, hence, of m_5, is $U_{\gamma_5,w} = U_{\gamma_1,w} \cap U_{\gamma_3,w}$. We also define composition in this manner, so that $m_5 = m_3 \circ m_1$. Similarly, another two-connection path is evident from (p_1, η) to $(p_4, \eta^{(3)})$ via (p_3, η'') and defining a morphism $m_6 = m_4 \circ m_2$. If we can establish that $U_{\gamma_5,w} = U_{\gamma_6,w}$, the diagram formed by m_1, m_2, m_3, m_4 is commutative, $m_3 \circ m_1 = m_7 = m_4 \circ m_2$ where Γ_7 is the two-path set $\Gamma_7 = \{\gamma_5, \gamma_6\}$, with $U_{\Gamma_7,w} = U_{\gamma_5,w} \cap U_{\gamma_6,w}$. The equality of path sets as shown is one of two alternatives in Figure 3. The other alternative, with m_8 in place of m_7, $\Gamma_8 = \{\gamma_5, \gamma_6\}$, with $U_{\Gamma_8,w} = U_{\gamma_5,w} \cap U_{\gamma_6,w}$ has the path-sets not equal, $U_{\gamma_5,w} \neq U_{\gamma_6,w}$, in which case the diagram is not commutative.

Notice that some weights of w, and, hence, the network's overall response to its inputs, may be changing under the influence of an instance (θ, e) of a morphism. The network activity and consequent weight-change resulting from (θ, e) result in a transition within A to a different stage of learning, where the weight array has transitioned from w to v. We express this as a change from a category $\mathbf{N}_{A,w}$ to a different category $\mathbf{N}_{A,v}$. There can be many categories $\mathbf{N}_{A,v}$ associated with a single morphism $m_{\Gamma,w}$ of $\mathbf{N}_{A,w}$ because the instances (θ, e) in $U_{\Gamma,w}$ can be associated with many different transitions.

The mathematical machinery associated with functors of the form $M\colon \mathbf{Concept} \longrightarrow \mathbf{N}_{A,w}$ enables the study of the incremental changes in the network's knowledge representation. The determination of the colimit and limit derivations possible in the neural categories associated with each stage of learning reveal how and at what stage these constructs are or can be represented in an existing or proposed neural architecture. This provides a mathematical semantic analysis of the connectionist learning performed by the network as it advances in both directions — specialization and abstraction — beginning at the sensor-percept level. If the derivation of a functor $M\colon \mathbf{Concept} \longrightarrow \mathbf{N}_{A,w}$ is blocked for a pair (A, w), we have identified a problem in forming a knowledge representation at the stage of learning represented by w. Since functors are many-to-one mappings on objects and morphisms, it is not necessary for an explicit representation of all concepts to be available— which is fortunate, since the category $\mathbf{Concept}$ is infinite and any stage of learning cannot be expected to have absorbed all knowledge in any case. Since many concepts and morphisms can be "compressed" in certain network regions by the many-to-one mapping, the inability to construct a functor signifies either a shortcoming in the design of A or that a transition to weight

array w driven by an input stream is unrealistic: the array w cannot be expected to occur, and if it has occurred, the problem is with A. If we can determine that a functor cannot exist for most or perhaps even for any w that might result from adaptation in A, the architecture has a clear deficit in its ability to represent its environment. On the other hand, we can apply the analysis to determine an A that is capable of learning a knowledge representation to the extent desired.

There is yet more to be obtained from the use of category theory. This derives from an analysis of the "possible worlds" that can occur in the environment of A. These worlds, represented by the theories of $\mathbf{Concept}$, are the subject of the next section.

V. Model Spaces

Each concept T has an associated space of *models*, $\mathtt{Mod}(T)$. A model is a mathematically-expressed situation or system of entities, a "possible world" described by T, that is, a system that satisfies the axioms of T. For example, a model of the theory T_5 in the colimit example consists of an arrangement of entities corresponding to the sorts and operations and satisfying the theory axioms, with the constants $\mathtt{p1}$, $\mathtt{p2}$, $\mathtt{p3}$, $\mathtt{la}$, $\mathtt{lb}$, $\mathtt{lc}$ serving as labels for three distinct entities considered to be points and three distinct entities considered as lines. What are these entities? An obvious example is one constructed using points and straight lines in the Euclidean plane. Another can be constructed on the surface of a sphere, where lines are great circles. Yet another can be constructed from the grid of pixels in a video image display, with pixels as points and with specific rules for identifying sets of pixels as lines (such as rows, columns, and diagonals at all orientations in the pixel grid). Still another is any set S of people such that any two members of S share a unique set L of attributes; if the people are the points, those sharing the attributes in L (e.g., dark hair, height between 5 feet 6 inches and 6 feet) form a line. Any other line L' must contain an attribute which is shared by at most one point (person) which (who) shares the attributes in L.

The variety of such examples is infinite. However, a model-theoretic formulation demands that the entities in any of these examples belong to a mathematical system of some kind. This provides a means of studying the structure imposed upon a "world" by a theory. Conversely, given certain models, theories can be derived to describe them.

In traditional model theory the systems are sets with structure. For a multi-sorted logic such as that used in Section III, the mathematical structure of a set model σ in $\mathtt{Mod}(T)$ incorporates a multiplicity of sets and functions, the objects and morphisms of the category $\mathbf{Set}$. For each sort u of T, there is a set u_σ. For each operation p of T, where p is given the form $p\colon u \longrightarrow u'$, there is a function $p_\sigma\colon u_\sigma \longrightarrow {u'}_\sigma$ mapping every member of the set u_σ interpreting the sort u to some member of the set ${u'}_\sigma$ interpreting the sort u'. A predicate is expressed as an operation p such that the sort u' is $\mathtt{Boolean}$, so that ${u'}_\sigma$ is the set of Boolean values $\mathtt{T}, \mathtt{F}$. For each constant c with sort u, there is a specific member c_σ of u_σ. Finally, each axiom of T must be valid for all quantities u_σ, p_σ, c_σ. For example, a model σ in the Euclidean plane for any of the theories of the triangle example has $\mathtt{Points}_\sigma$ as the set of points of the planes and $\mathtt{Lines}_\sigma$ as the set of straight lines in the plane. The operation $\mathtt{on} : \mathtt{Points} * \mathtt{Lines} -> \mathtt{Boolean}$ signifying a relation predicate—which would be interpreted by a subset of the cartesian product $\mathtt{Points}_\sigma \times \mathtt{Lines}_\sigma$ in an unsorted logic model—is instead interpreted by a function $p_\sigma\colon (\mathtt{Points} * \mathtt{Lines})_\sigma \longrightarrow \mathtt{Boolean}_\sigma$, where $(\mathtt{Points} * \mathtt{Lines})_\sigma = \mathtt{Points}_\sigma \times \mathtt{Lines}_\sigma$. 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 quantities $x, y \in \mathtt{Points}_\sigma$ satisfying the precondition $x \neq y$: That is, there exists a unique member of $\mathtt{Lines}_\sigma$ defined by x and y. If lines are interpreted as sets of points that can be identified uniquely by any two of their members, then $\mathtt{Lines}_\sigma$ is a set of sets that satisfy this very demanding condition.

Whether it is defined to lie within the Euclidean plane, a pixel grid, or some other context, a model of a theory can be expressed as a collection of sets and functions. The many different models a theory T can have are objects in its model space $\mathtt{Mod}(T)$. These ideas can be used to analyze mathematically the input environment of a neural network. Entities of interest are collected into sets which represent sorts. Relationships between the entities become functions, and entities of special interest become named elements of the sets, representing sorted constants. Since there can be many theories represented explicitly by objects (p, η) through a functor $M\colon \mathbf{Concept} \longrightarrow \mathbf{N}_{A,w}$, and each theory can have many different models in $\mathbf{Set}$, which we hypothesize have correspondents in instance sets $U_{(p,\eta)}$ (and similarly for morphisms), a given entity or collection of entities in the environment can be represented in many different ways (as can relationships between the entities). The importance of our semantic theory is evident here, however, because the different representations of entities and their relationships are related as the theories are related via theory morphisms. In addition, the model spaces have structure relating the models. The next section will address the first point, by showing how theory morphisms have corresponding morphisms that map model spaces to model spaces.

VI. Model-Space Morphisms

Notice that a model σ in $\mathtt{Mod}(T_1)$, while it must have specific members of $\mathtt{Points}_\sigma$ identified as interpretations for its point constants, is not required to have specifically identified members of $\mathtt{Lines}_\sigma$ because T_1 does not specify any line constants. On the other hand, specific lines serving in the roles of its three line constants must exist in any model σ' in

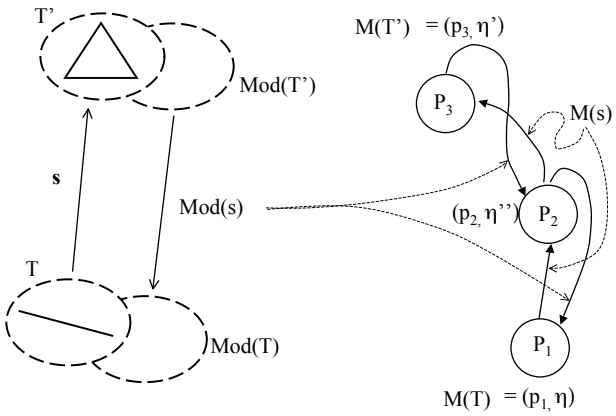


Fig. 4. Reciprocal connections implement a model-space morphism.

$\texttt{Mod}(T_5)$. The more specific theory T_5 puts more constraints on its models than does the more abstract theory T_1. Since it imposes fewer constraints, there can be more models of T_1 than there are of T_5. In fact, every model of T_5 can serve as a model of T_1 by tracing backward through the theory morphism $\ell_1 : T_1 \longrightarrow T_5$ of the colimit example. Every item in the presentation of T_1, that is, `Points`, `Lines`, `on`, and the constants `p1, p2, p3`, is mapped to T_5. Further, the axiom `Two-points-define-a-line`, transformed by symbol substitution (in this case, the identical symbols), must hold for the mapped items in T_5 as well as in T_1, and every model σ' of T_5 must include interpretations for the mapped items. Therefore, σ' must satisfy the constraints of T_1 in the form in which they appear in T_5 (identical in this case). This implies that σ' has a correspondent σ in $\texttt{Mod}(T_1)$. In fact, σ can appear as σ' minus some of its structure—in the present case, the interpretations of the line constants of T_5 in σ' need not be included in σ.

The foregoing implies the existence of a function $\texttt{Mod}(\ell_1) : \texttt{Mod}(T_5) \longrightarrow \texttt{Mod}(T_1)$ that preserves the model structure imposed by the constraints of T_1, mapped into T_5 by ℓ_1. In general, every theory morphism $s : T \longrightarrow T'$ has an accompanying *model-space morphism* $\texttt{Mod}(s) : \texttt{Mod}(T') \longrightarrow \texttt{Mod}(T)$, where for every model σ' in $\texttt{Mod}(T')$ there exists a unique model $\sigma = \texttt{Mod}(s)(\sigma')$ in $\texttt{Mod}(T)$. This relationship between concept and model-space morphisms is illustrated in Figure 4. A further insight can be found in the analysis of model spaces and their morphisms $\texttt{Mod}(s) : \texttt{Mod}(T') \longrightarrow \texttt{Mod}(T)$, which are in correspondence with concepts and their morphisms $s : T \longrightarrow T'$. We postulate that there is a correspondence between the instances of an object $M(T) = (p_i, \eta)$ of $\mathbf{N}_{A,w}$ and the models in $\texttt{Mod}(T)$. Similarly, we postulate that there is a correspondence between the inputs associated with each morphism $M(s) : M(T) \longrightarrow M(T')$ and the mapping of models through $\texttt{Mod}(s) : \texttt{Mod}(T') \longrightarrow \texttt{Mod}(T)$. *This has great significance for neural networks*, for it says that an instance of activity associated with an object $(p_j, \eta') = M(T')$ must also be associated with an object $(p_i, \eta) = M(T)$ when there is a **Concept** morphism $s : T \longrightarrow T'$, as illustrated in Figure 4. *This yields a design principle for neural networks:* There must be a mechanism to ensure that every instance of the codomain object of a neural morphism is also an instance of its domain object. The surest way to ensure this is for the associated neural network to have *reciprocal connections*, allowing activity to propagate backward from the node associated with the codomain object.

One aspect of this principle is that in a neural network (A, w) whose associated category $\mathbf{N}_{A,w}$ has colimits and limits, we can predict the network behavior associated with these diagrammatic structures as follows. First, an instance (θ, e) of a colimit object must also be an instance of the objects in its base diagram. That is, a stimulus promoting feedforward signaling from a few of the diagram object nodes, or from another source, may cause a colimit object node to become active within the appropriate interval. This may or may not occur; the feedforward connection weights might be insufficient, the colimit object node may be in competition with other colimit object nodes as a representative of the same input e, and so forth. However, if it does occur and the colimit object node's activity persists (one of our neural transitions occurs), then the resulting feedback to the diagram must be sufficient to ensure that (θ, e) is also an instance of all of the base diagram objects. For limits, the situation is reversed: The feedback occurs *from* each base diagram object node *to* the limit object node, forcing (θ, e) to be an instance of the limit object when it is an instance of *any one* of its diagram objects; on the other hand, an instance of a limit object need not be an instance of any of its diagram objects; the limit object node may, however, provide enough input to make its base diagram object nodes serious competitors as possible representatives of the current input e.

VII. The Structure of Model Spaces

Similarities between models of a theory give each model space an internal structure: A model σ of a theory T can sometimes be mapped in a structure-preserving fashion to another model τ of T, and these structural maps have a composition. A *model homomorphism* $h : \sigma \longrightarrow \tau$ is defined as follows. For each sort u of T, there is a function $h_u : u_\sigma \longrightarrow u_\tau$ such that for each constant $c : u$, the member c_σ of u_σ maps as $h_u(c_\sigma) = c_\tau \in u_\tau$; and for each operation $p : u \longrightarrow u'$ with its associated functions $p_\sigma : u_\sigma \longrightarrow {u'}_\sigma$ and $p_\tau : u_\tau \longrightarrow {u'}_\tau$ and each member a of u_σ, we have $h_{u'}(p_\sigma(a)) = p_\tau(h_u(a)) \in u'_\tau$. The composition of two homomorphisms h and h' is defined by the compositions of the individual functions, $(h' \circ h)_u = (h'_u \circ h_u)$, where function composition is as usual given by $(h'_u \circ h_u)(x) = h'_u(h_u(x))$. The identity homomorphism $\texttt{id}_\sigma : \sigma \longrightarrow \sigma$ is easily shown to exist for each model σ, where for arbitrary homomorphisms $h : \mu \longrightarrow \sigma$ and

$k\colon\sigma \longrightarrow \tau$, and any $a \in u_\mu$ and $b \in u_\sigma$, we have $((\mathtt{id}_\sigma)_u \circ h_u)(a) = h_u(a) \in u_\sigma$ and $(k_u \circ (\mathtt{id}_\sigma)_u)(b) = k_u(b) \in u_\tau$. It also follows from the corresponding law for function composition that homomorphism composition is associative. Therefore, with models as its objects and homomorphisms as its morphisms, the model space $\mathtt{Mod}(T)$ has been shown to be a category.

We have seen that model spaces are categories in their own right. A similar line of reasoning yields the fact that the model-space morphisms are functors between the model categories. A more comprehensive introduction, involving categorical logic and model theory, reveals yet more underlying structure (see any of [8], [20], [10]). If we can establish an explicit mathematical relationship between model categories and neural network input environments, we can achieve greater depth in analyzing the semantics of neural architectures and study the details of neural computation. This in turn will aid in enlarging the set of design principles for neural networks derived from the semantic theory. For example, an analysis of model categories and functors is expected to shed light on the relationship between the organization of the input environment of a neural network and the effect of sequences of inputs on learning via successive weight changes.

VIII. Conclusion

The mathematical semantic theory presented here can aid researchers in neural networks and neuroscience in achieving a deep understanding of neural computation. This understanding is made possible by viewing neural computation as driven by the derivation of a distributed ontology representing knowledge associated with the network's environment. The current understanding achieved with the semantic theory has already provided design principles as well as insights for investigations in neural and cognitive science. Some insights derive from associating concept structures such as colimits and limits with neural architectures via the category $\mathbf{Concept}$ and functors of the form $M\colon\mathbf{Concept} \longrightarrow \mathbf{N}_{A,w}$: For example, colimit and limit derivations suggest a study of learning as the long-term potentiation of weights within connection structures that represent the defining diagrams. Another insight arises from the view of multi-regional or multi-sensor neural networks as knowledge-driven systems with a coherence requirement [15]. Model theory yields yet another insight, suggesting that an effective neural design must provide feedback so that concept morphisms can be correctly represented by including model-space morphisms within the connectionist structure. The colimit-limit and model-space morphism insights have been successfully applied [14]. Further depth in the theory, including an anlaysis of model spaces as categories, shows promise of yielding further insights.

References

[1] R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.

[2] Michael A. Arbib. *Brains, Machines, and Mathematics*. Springer, New York, 1987.

[3] Guszti Bartfai. Hierarchical clustering with ART neural networks. In *Proceedings of the IEEE International Conference on Neural Networks, June 28–July 2, 1994*, volume II, pages 940–944. IEEE, 1994.

[4] M. K. Bennet. *Affine and Projective Geometry*. John Wiley and Sons, New York, 1995.

[5] Gail A. Carpenter, Stephen Grossberg, and John H. Reynolds. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, 4:565–588, 1991.

[6] Gail A. Carpenter and Ah-Hwee Tan. Rule extraction: From neural architecture to symbolic representation. *Connection Science*, 7:3–27, 1995.

[7] M. W. Craven and J. W. Shavlik. Learning symbolic rules using artificial neural networks. In *Proceedings of the 10th International Machine Learning Conference, Amherst, MA*, pages 73–80, San Mateo, CA, 1993. Morgan Kaufmann.

[8] R L Crole. *Categories for Types*. Cambridge University Press, 1993.

[9] L. M. Fu. A neural network for learning rule-based systems. In *Proceedings of the International Joint Conference on Neural Networks, Baltimore*, pages 343–348, 1992.

[10] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[11] Michael J. Healy. A topological semantics for rule extraction with neural networks. *Connection Science*, 11(1):91–113, 1999.

[12] Michael J. Healy and Thomas P. Caudell. Acquiring rule sets as a product of learning in a logical neural architecture. *IEEE Transactions on Neural Networks*, 8(3):461–474, 1997.

[13] Michael J. Healy and Thomas P. Caudell. A categorical semantic analysis of ART architectures. In *IJCNN'01:International Joint Conference on Neural Networks, Washington, DC*, volume 1, pages 38–43. IEEE,INNS, IEEE Press, 2001.

[14] Michael J. Healy, Richard D. Olinger, Robert J. Young, Thomas P. Caudell, and Kurt W. Larson. Modification of the ART-1 architecture based on category theoretic design principles. In *IJCNN 2005 Conference Proceedings: International Joint Conference on Neural Networks. Montreal, Quebec, Canada. (CD-ROM Proceedings)*, pages 457–462. IEEE,INNS, IEEE Press, 2005.

[15] Michael John Healy and Thomas Preston Caudell. Ontologies and worlds in category theory: Implications for neural systems. *Axiomathes*, 16(1-2):165–214, 2006.

[16] Gregory L. Heileman, Michael Georgiopoulos, Michael J. Healy, and Stephen J. Verzi. The generalization capabilities of ARTMAP. In *Proceedings of the International Joint Conference on Neural Networks (ICNN97), Houston, TX*, 1997.

[17] N. K. Kasabov. Adaptable neuro production systems. *Neurocomputing*, 13:95–117, 1996.

[18] F. W. Lawvere and S. H. Schanuel. *Conceptual Mathematics: A First Introduction to Categories*. Cambridge University Press, 1995.

[19] S. Mac Lane. *Categories for the Working Mathematician*. Springer, 1971. This is the standard reference for mathematicians, written by one of the two co-discoverors of category theory. The other was S. Eilenberg.

[20] J. Meseguer. General logics. In *Logic Colloquium '87*, pages 275–329. Science Publishers B. V. (North-Holland), 1989.

[21] S. Mitra and S. K. Pal. Fuzzy multi-layer perceptron, inferencing and rule generation. *IEEE Transactions on Neural Networks*, 6:51–63, 1995.

[22] B. C. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, 1991.

[23] G. Pinkas. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77:203–247, 1995.

[24] Rudy Setiono. Extracting rules from neural networks by pruning and hidden-unit splitting. *Neural Computation*, 9:205–225, 1997.

[25] J. Sima. Neural expert systems. *Neural Networks*, 8:261–271, 1995.

[26] Keith Williamson, Michael Healy, and Richard Barker. Industrial applications of software synthesis via category theory-case studies using specware. *Automated Software Engineering*, 8(1):7–30, 2001.