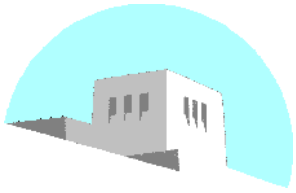
6-25-2004

Neural Networks, Knowledge and Cognition: A Mathematical Semantic Model Based upon Category Theory

Michael John Healy

Thomas Preston Caudell

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING



SCHOOL OF ENGINEERING

UNIVERSITY OF NEW MEXICO

Neural Networks, Knowledge, and Cognition: A Mathematical Semantic Model Based upon Category Theory

Michael John Healy
Department of Electrical and Computer Engineering
The University of New Mexico
Department of Electrical Engineering
The University of Washington
13544 23rd Place NE
Seattle, WA 98125, USA
e-mail: mjhealy@ece.unm.edu

Thomas Preston Caudell
University of New Mexico
Department of Electrical and
Computer Engineering
Albuquerque, New Mexico 87131, USA
e-mail: tpc@ece.unm.edu

UNM Technical Report: EECE-TR-04-020

Report Date: June 25, 2004

Abstract

Category theory can be applied to mathematically model the semantics of cognitive neural systems. We discuss semantics as a hierarchy of concepts, or symbolic descriptions of items sensed and represented in the connection weights distributed throughout a neural network. The hierarchy expresses subconcept relationships, and in a neural network it becomes represented incrementally through a Hebbian-like learning process. The categorical semantic model described here explains the learning process as the derivation of colimits and limits in a concept category. It explains the representation of the concept hierarchy in a neural network at each stage of learning as a system of functors and natural transformations, expressing knowledge coherence across the regions of a multi-regional network equipped with multiple sensors. The model yields design principles that constrain neural network designs capable of the most important aspects of cognitive behavior.

Keywords

category, cognition, colimit, functor, limit, natural transformation, neural network, semantics

1 Introduction

1.1 Neural Network Semantics and Knowledge

The search for a deeper understanding of neural networks has led investigators to the notion that the distributed pattern of connection weights formed within a network as a result of its processing of input patterns can be explained in terms of human- understandable rules ([2], [3], [4], [7], [8], [9], [13], [20], [18], [25], [31], [37], [40], [44], [45]). Often, the rules are regarded as knowledge made explicit by a symbolic-rule-extraction technique. The knowledge typically has the form IF (input stimulus) THEN (output response), providing an explicit description of the relationship between meaningful items in the input patterns and desired responses to those items. The symbolic expressions in the rules are meant to describe significant items associated with the stimuli as well as items expressed through the network responses. Some investigators have used mathematics to remove ambiguity in the descriptions and to associate with rigor the relationship of rule expressions to connection weights. Typically, they do this by expressing the rules as symbolic formulas in a formal logic. This use of symbolic statements to model the knowledge content of a computational system is called *mathematical semantics*[39].

The original inspiration for neural networks is the highly interconnected, massively parallel, distributed, adaptive neuron-and-synapse structure of the brain. In contemporary modeling, the neurons and other architectural elements are organized on a larger scale in a system of interconnected functional regions([11], [27]). Each region is associated with one or more sensory modalities, motor control, planning, the control of working memory ([38], [41], [47], [52]), and possibly self-referential processing, at least in humans[32]. The unique functionality of each region can be described by a system of knowledge with which it has been associated. Neuroscientists work to understand the content and organization of the knowledge systems by carefully analyzing activity in different brain regions during the perfomance of tasks, and by studying the task performance deficits associated with damage to different regions and/or their interconnections found in patients. In general, the knowledge asssociated with a region exists as a system of interconnected partial descriptions of sensed items and events. A goal of neuroscience is to be able to use these partial descriptions in different combinations to explain the brain's activity in response to a given situation. For example, visual processing regions in the primate brain contain representations of visual objects in terms of basic features of color, form and texture. Descriptions of visual objects, their form, color and other features, and the relationships between objects and features constitute a knowledge system. Region-specific knowledge representations act together through interconnections between regions to produce a response to each input [43] and, in the brains of some animals, cognitive behavior([11], [15], [46], [26], [43], [29], [27], [48]). For example, visual and auditory representations are unified in an association region that shares interconnections with the vision- and sound-specific processing regions.

This paper addresses the question of where and how knowledge is stored in connectionist systems organized to greater or lesser extent like the brains of various organisms. It presents a mathematical model of knowledge as acquired by and stored within multi-region neural networks. Because it directly addresses the knowledge represented in the adaptive structure of a neural network, the model is called a semantic model. As such, it is consistent with previous work in logical rule modeling, but transcends rule-modeling by *explaining the structure* of the knowledge contained in the rules and the component statements that form the rules. That is, the model explains not only what the rules are saying, but how the rules arise from a hierarchical system of concepts, with the hierarchy directed from the abstract to the specific. It explains this process as an incremental re-use of existing concept representations in the neural network to form new concept representations as the network processes input patterns. The model provides a mathematically rigorous yet natural explanation of the combining of the interacting regional knowledge structures so that the network, if well-designed, acts as if there were a single knowledge structure guiding its behavior. This is a key property studied with the model which we call *knowledge coherence*.

The intended scope of the semantic model is the full range of neural network architectures, artificial or biological, from single-layer perceptrons to highly complex, multi-regional, mutli-sensor networks with recurrent connections. The semantic model provides an analytical framework for understanding the capability of any neural network design in terms of the knowledge it is capable of acquiring from its input data. It provides an analysis

vehicle for the systematic design of neural networks having a desired knowledge-acquisition capability. Its use in analysis and design can lead to a fundamental understanding of neural networks and a quantum jump in the state of the art of biological neural system analysis and artificial neural network design.

The mathematical semantic model is based upon category theory. Category theory, the mathematical theory of structure, is seeing increasing use in representing the hierarchical structure of knowledge and its manifestations in computational structures—thus achieving a mathematical model of their semantics ([10], [12], [16], [28], [36], [39], [49], [50], [51], [53]). This approach is now being applied to investigate the acquisition of knowledge through adaptation in neural networks, and in this paper we describe the theoretical basis for this investigation. Previous papers ([17],[19],[21], [22], [23]) have presented much of the theory in a preliminary form along with an initial application to architectural design.

The paper is organized as follows. Since a grounding in category theory is as yet uncommon, Section 2 provides a brief review. Along with this, it introduces the fundamental categorical quantities used in semantic analysis. Section 3 introduces categories of concepts and their instances, borrowing heavily from categorical logic and model theory. Section 4 introduces the categories used in modeling neural architectural structure, and Section 5 describes the association of concept categories with neural categories. Section 6 contains a discussion of the implications for neural architecture design resulting from categorical constructions that model the learning of specializations and abstractions derived from prior knowledge. Section 7 introduces information fusion across multiple modalities as an interconnected system of concept hierarchy representations associated with network sub-regions, and Section 8 presents an initial neural network design based upon this analysis. Section 9 contains the final discussion and conclusion.

2 Category Theory: A Brief Introduction

2.1 A Mathematical Theory of Structure

A brief, straightforward introduction to category theory is contained in [39]. There are other good introductions with varying mathematical detail, among them [1], [10], [33], and [34]. Category theory has been proposed as an alternative to set theory as a foundation for mathematics; however, the two are normally used together, since each has unique advantages in representing mathematical quantities in the most fundamental or abstract terms. The primitive notion in set theory, in terms of which all others are defined, is that of the membership of a quantitiy x in a collection y, denoted $x \in y$. in category theory, on the other hand, the primitive notion is that of an arrow, or *morphism*—a relationship between two *objects* in a *category*. A category can be thought of as a system of mathematical structures of some kind, concrete (such as algebras called groups) or abstract, and the relationships (morphisms or arrows) between them that express that type of structure (in the cased of groups, the arrows are the group homomorphisms). Each morphism $f{:}a \longrightarrow b$ has a *domain* object a and a *codomain* object b; that is, f serves as a sort of directed relationship between a and b. In a *category* C, each pair of arrows $f : a \longrightarrow b$ and $g : b \longrightarrow c$ (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 the associative law: 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, $h \circ (g \circ f) = (h \circ g) \circ f$. Also, 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 $g{:}c \longrightarrow a$ $\mathtt{id}_a \circ g = g \quad$ and $\quad f \circ \mathtt{id}_a = f$.

The *principle of duality* is a fundamental notion in category theory. The dual or opposite C^{op} of a category C has the same objects, and the arrows reversed. The *dual of a statement* in category theory is the statement with the words "domain" and "codomain" reversed and compositions $f \circ g$ in place of compositions $g \circ f$ (note that the reversed arrows are by convention given the same names as the originals). If a statement is true of a category C, then its dual is true of C^{op}; if a statement is true of all categories, then, because every category is dual to its dual, the dual statement is also true of all categories. This phenomenon means that, roughly speaking, "half

the theorems of category theory are obtained for free", since proving a theorem immediately yields its dual as an additional theorem (see any of [1], [39], [34]).

Category theory provides a mathematically rigorous notion of "isomorphism", a term which is often used in a loose, intuitive sense in discourse. For example, in a discussion, one sometimes hears a statement such as "the two [concepts, data types, program constructs, etc.] are in some sense isomorphic". If we can formalize the entities under discussion and include them in a category as objects, we can apply a more rigorous notion: 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.

Certain notions in category theory are key to semantic modeling. One is the notion of an *initial object* in a category and the dual notion of a *terminal object*. 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 in a category. 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 $f' \circ f = \mathtt{id}_i$ and $f \circ f' = \mathtt{id}_{i'}$.

There is a category **Set** whose objects are sets, whose morphisms are functions, and for which composition 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 X there is an identity function $\mathtt{id}_X$ whose values are $\mathtt{id}_X(x) = x (x \in X)$, so **Set** is indeed a category. 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. Finally, notice that a **Set** isomorphism is just a one-to-one, onto function.

A second example of a category dispels any notion that the morphisms must represent mappings. Consider the category $\mathbf{N}_{|}^{+}$, in which the objects are nonzero natural numbers and in which there is a morphism $|_{n,m}{:}n \longrightarrow m$ exactly when n is a divisor of m, $n \mid m$. Notice that the transitive property of the divisor relation yields a composition operation, which is associative. Identities exist since every nonzero natural number divides itself, $n \mid n$. In fact, natural numbers are the objects in more than one category. Consider the category $\mathbf{N}_{\leq}$, which has all the natural numbers as objects and in which there is a morphism $\leq_{n,m}{:}n \longrightarrow m$ exactly when the inequality $n \leq m$ holds. Again, the transitivity of the relation used to define the category, in this case $\leq$, yields an associative composition operation, and again identities exist, in this case since every nonzero natural number is related to itself, $n \leq n$.

A second fundamental notion in category theory is that of a *commutative diagram*. In Figure 1 , the diagram Δ expresses the fact that 4 and 6 are both divisible by 2. Diagram Δ' expresses also the additional knowledge that 24 is divisible by 4 and 6. The dashed arrow represents the consequential knowledge that (because 24 is divisible by 4 and 6 while 4 and 6 are divisible by 2) 24 is divisible by 2. Notice that there are two morphisms from 2 to 24 that are compositions along a path directed through a third object (4 and 6, respectively), yet there is at most one divisibility morphism from one natural number to another. Therefore, $|_{4,24} \circ |_{2,4} = |_{2,24} = |_{6,24} \circ |_{2,6}$. The diagram Δ' is said to be a *commutative diagram*.

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. In other words, a commutative diagram is associated with a system of equations involving compositions of morphisms. This expresses a system of constraints on mathematical structures associated with the category.

Diagram Δ Diagram Δ'

Figure 1: **Two diagrams in the category Nat$_|$.**

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*. Colimits have long been used ([16], [28], [53]) in categorical logic and computer science as a means of expressing complex formal logic theories or specifications of theories in terms of their simpler, component theories or specifications. In a later section, we apply colimits and limits to model the semantics of the Hebbian-like learning in a neural network, where the network learns by adapting the synaptic connections strengths between neuron-like elements called nodes .

2.2 Colimits

Let Δ be a diagram in a category C as shown in Figures 2 and 3 , 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 3 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 2 , 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

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

That is, h is a factor under composition of each leg morphism g''_i of K'' with respect to the corresponding leg morphism g'_i of K'. This is illustrated in Figure 2 . Re-using the symbol h for notational efficiency, we also 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.

A *coproduct* (Figure 4) is the colimit of a discrete diagram, one having objects but no morphisms among them. In **Set**, for example, coproducts are disjoint unions of the component sets.

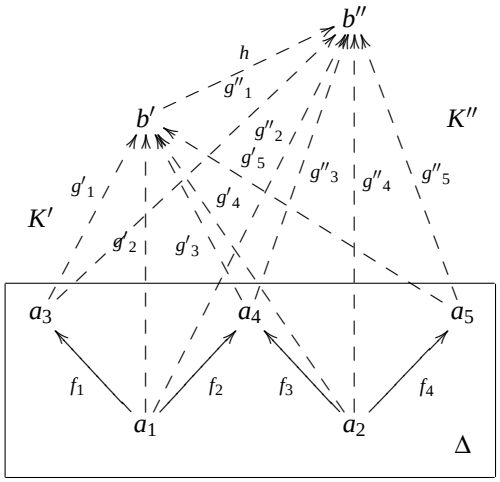


Figure 2: **A cocone morphism $h{:}K' \longrightarrow K''$ in $\mathbf{coc}_\Delta$ is a morphism $h{:}b' \longrightarrow b''$ in C between the apical objects b' and b'' of cocones K' and K'', respectively, that is a factor of each leg morphism $g''_i{:}a_i \longrightarrow b''$ of K'', with $g''_i = h \circ g'_i$.**

2.3 Limits

Limits are the dual notion to colimits; that is, the one notion is obtained from the other by "reversing the arrows" and interchanging "initial" and "terminal" (for objects). The reason for discussing colimits first is that they have a history of use in categorical logic and computer science ([16], [53]) and, perhaps more importantly, colimits for all diagrams exist in certain categories of interest to us—but limits do not exist for all diagrams in these categories. The diagrams for which limits do exist are in that sense special, and we shall have more to say about

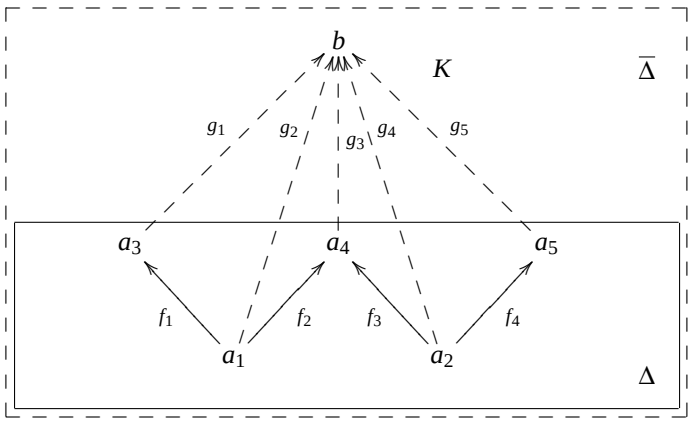


Figure 3: **A colimit for a diagram Δ. The extended diagram $\overline{\Delta}$ extends Δ with a conical structure of morphisms from all diagram objects $a_1, \ldots, a_5$ pointing to an apical object b**

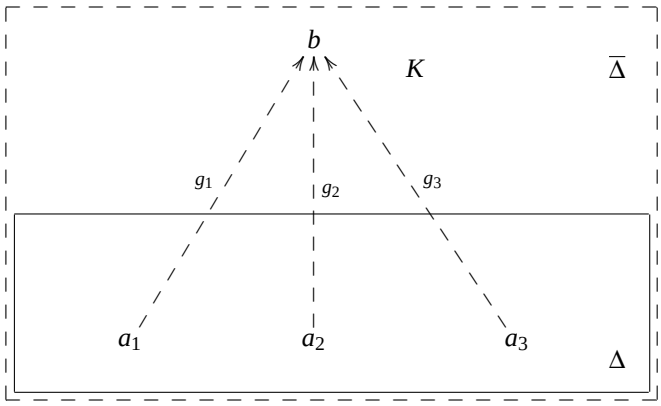


Figure 4: **A coproduct, where the base diagram Δ is discrete.**

this when we discuss the concept hierarchy representation of neural network semantics.

Duality makes a discussion of limits quite simple, now that colimits have been defined. Our example will have a correspondingly simple diagram. 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, which are directed out of the diagram. Cone morphisms are defined appropriately by analogy with cocone morphisms, and again composition follows directly from 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 $\mathbf{cone}_\Delta$. That is, for every other cone K' for Δ, there exists a unique cone morphism $h{:}\, K' \longrightarrow K$. Again, the original diagram Δ is called the *base diagram* for the limit and the diagram $\underline{\Delta}$ formed by adjoining K to Δ is called its *defining diagram*. Note that, as all terminal objects are isomorphic, all limits for a given base diagram are isomorphic. A limit for a discrete diagram is called a *product*, and the leg morphisms are called *projections*. The familiar cartesian product of sets is an example in the category **Set**.

2.4 Structural Mappings

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 house plan exists as a complex of forms either inscribed in ink on paper or electronically within a computer. The plan can be implemented (mapped) many times, with variations in the fine details of construction, to build houses. Each instance of building a house from the plan can be thought of as a mapping from the structure detailed in the architectural plan to a structure made of wood, brick, stone, metal, wallboard, and other materials. The material substances of the plan and the house are different but the structure given in the plan is essentially unchanged in the constructed house. In category theory, the notion of a structure-preserving mapping is formalized in the definition of a *functor*. 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,

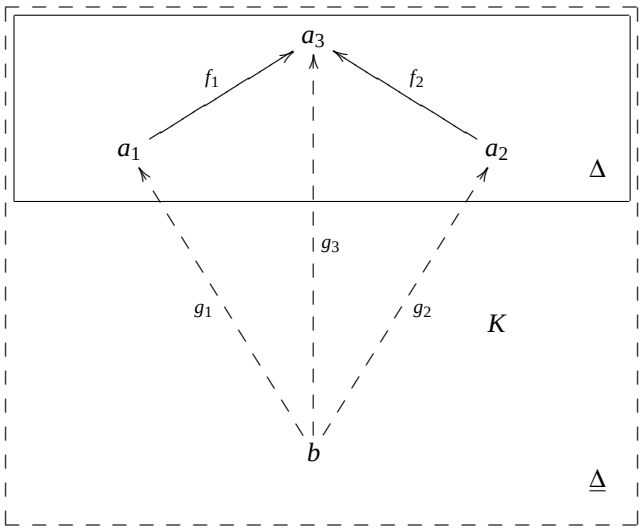


Figure 5: **A limit for a diagram Δ. The extended diagram $\underline{\Delta}$ extends Δ with a conical structure of morphisms to all diagram objects $a_1, \ldots, a_3$ from an apical object b.**

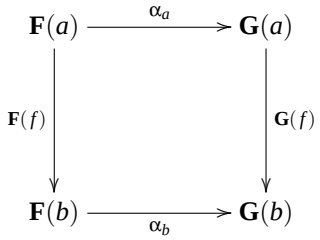


Figure 6: **A commutative diagram associated with a natural transformation. The morphisms $\mathbf{G}(f) \circ \alpha_a : \mathbf{F}(a) \longrightarrow \mathbf{G}(b)$ and $\alpha_b \circ \mathbf{F}(f) : \mathbf{F}(a) \longrightarrow \mathbf{G}(b)$ are one and the same, $\mathbf{G}(f) \circ \alpha_a = \alpha_b \circ \mathbf{F}(f)$.**

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.

The two categories $\mathbf{N}_{|}^{+}$ and $\mathbf{N}_{\leq}$ yield an example of this important kind of structural relation. Define a functor $F : \mathbf{N}_{|}^{+} \longrightarrow \mathbf{N}_{\leq}$ as follows. The image of each positive natural number n is itself, that is, $F(n) = n$. The image of each morphism $|_{n,m}$ is the morphism $F(|_{n,m})$, where $F(|_{n,m} : n \longrightarrow m) = \leq_{n,m} : n \longrightarrow m$, which works because $n \mid m$ implies $n \leq m$. Notice that the compositional structure of $\mathbf{N}_{|}^{+}$ is appropriately preserved.

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 6 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 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.

3 Categorical Semantics

3.1 Concept Morphisms: The Structure of Knowledge

Few neural network researchers would argue with the notion that meaningful representations of the items that are sensed to form input patterns for an adaptive network become associated with the connection weights through Hebbian-like adaptation, or learning. That is how the network builds an ”internal model” leading to its ability to perform a useful task associated with the input examples it has processed. That the many components of the network are highly distributed and interdependent suggests that the represented items can be decomposed into parts. For example, an item represented by an input pattern can be regarded as a combination of features that stimulate individual input nodes to varying levels of activity. Symbolic descriptions for the parts or features of input items can be associated with input patterns that express the features. Since the patterns are responsible for the network activity that results in connection weight adaptation, it is conceivable that a system for manipulating symbolic descriptions of items can be associated with the representation of input items in the distributed connection weight pattern of the network. This representation can be organized into a hierarchical system that relates the more abstract items such as simple, pattern-expressed features to the more complex, more specific items that are *composed* of the more abstract items. Categorical constructs involving a category of concepts and concept relationships will be our system language for expressing the association of symbolic descriptions with adaptively-formed, or learned, distributed representations of items.

A category **Concept** provides the hierarchical structure of descriptions associated with a distributed system of item representations. Its objects are concepts, or symbolic descriptions of items, and its morphisms are concept relationships. A concept morphism $s: T_i \longrightarrow T_j$ is an association of the description constituting concept T_i with a subconcept, or logical part, of the description constituting concept T_j. We use an already-available mathematical convenience, a category of formal logic theories and theory morphisms[10], for the category **Concept**. An example will serve to describe the objects and morphisms of this category, as well as provide a concrete example of a colimit. The example is presented in full in the Appendix, and an overview of it is given in the next section sufficient to follow its use as an illustration of the analysis of neural network semantics.

3.2 Example: Concepts, Morphisms and a Colimit

The definitions and axioms of a theory, as well as all formulas that can be proved from them, are symbolic statements in a language sufficiently expressive and unambiguous that it can be used to phrase propositions for proof by a mechanical theorem-prover. A theory describes a domain of items and their properties; a theory of geometry is an example, where the items are geometric objects. A complete listing of the statements of a theory includes the definitions of the quantities whose properties it describes, the axioms that assert the properties assumed in the theory, all the quantities whose existence one can derive from the defined quantities, and all the further properties of the quantities that one can prove from the axioms—that is, the theorems. It is not necessary to always have in hand the entirety of a theory’s quantities and theorems. A simple alternative which we use in most of our discussion is to list only its definitions and axioms. The inferred quantities can be derived and theorems proved as they are needed. A theory can appear in several different forms, depending upon which of its quantities and statements are regarded as definitions and axioms, for there are often many alternative choices. For our formalization of concepts and their morphisms, we shall use the alternative of listing a single choice. Any such list is called a *formal specification*. Even though specifications are not the same as theories, the distinction will not be important for the purposes of this presentation.

We represent concepts as theories (or their specifications). A concept morphism is a theory morphism. The following example illustrates concepts, concept morphisms, commutative diagrams and colimits. It shows how the concept of a triangle can be derived from simpler concepts as the apical object of a colimit for a diagram. The concepts and morphisms are expressed in a syntax similar to that used in [53]. For simplicity, a triangle is regarded as a construct obtained by joining three distinct, intersecting lines (the angles often associated with a triangle are not discussed). The example is only sketched here, but the Appendix contains a full exposition. The

discussion in subsequent sections describes the association of the example colimit with the learning of complex concepts and morphisms by a neural network through the re-use of prior conceptual knowledge.

Concept T_1 is a very basic theory of points and lines. In this presentation, points are regarded as undefined quantities and lines are quantities defined in terms of points. This is done through a logical predicate `on` that has two arguments, a point and a line, and is true just in case the point is associated with (or "lies on") the line. The `on` predicate is used in an axiom to express the notion that any two distinct points are "on" some unique line (see [5] for the use of this axiom in several different geometries).

```
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 statement line `sorts Points,Lines` introduces the most basic sorts of the concept T_1. Sorts are "logical containers" which are used to distinguish between the different types of things referred to by the variables or constants in logical formulas. For example, the universal quantifier (`forall`) portion of the axiom in T_1, `forall(x,y : Points)`, makes it clear that the axiom is a formula applying to all things x and y of type "Points". As a consequence, the antecedent of the first implication of the axiom, `(x not = y)`, is understood to mean that x and y represent two distinct *points*, as opposed to lines, circles, widgets, or any other kind of thing. The line `op on:  Points*Lines -> Boolean` specifies that there is an operation that maps any ordered pair (x,l) consisting of a point and a line to a member ("true" or "false") of the `Boolean` sort. The meaning of `on (x,l)` is "Point x lies on line l". An operation which takes values in `Boolean` is a called a predicate. The `Boolean` sort is part of a concept of logical operations that is implicitly included in every concept (it is an initial object of the concept category). Versions of formal logic containing predicates allow for highly expressive formulas, or statements, that employ functions and quantifiers such as `forall` and `exists`, corresponding to the usual universal and existential quantifiers $\forall$ and $\exists$, respectively. There are many derived sorts in a theory, such as products of the given sorts. If the sort symbols represent sets, then product sorts represent cartesian products of the sets. For example, the product sort `Points*Lines` represents the set of ordered pairs (x,l) of points and lines. Combinations of sorts and operations associated with them are similar to abstract data types in software specifications.

Notice that T_1 also states the existence of three labelled, but otherwise unspecified, points `p1`, `p2` and `p3`. This is done through the use of the statement form `const X : Points`, which is a way of stating that there exists a specific (but otherwise indefinite) point with the label `X`. The three constants may or may not represent distinct points: Their separate nature must either be stated as an axiom of T_1, or provable from the axioms of the concept.

We next express three concepts T_2, T_3 and T_4 by making and modifying three copies of T_1. In each new concept, we add a line constant, re-name the three point constants (for clarity in this presentation—otherwise, the specific names are not important), and associate two of the point constants with the line constant via the `on` predicate. Notice the additional inclusion of an axiom stating that the three point constants denote distinct points. A specification for the first of the three concepts, T_2, is as follows:

```
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
```

The other two concepts are identical in form but differ in the naming of the point and line constants. They are listed in full in the Appendix.

A concept morphism $s_1: T_1 \longrightarrow T_2$ maps the sort symbols `Points` and `Lines` to sort symbols in T_2. These symbols are left unchanged by mapping `Points` to `Points` and similarly for lines, and also the `on` predicate. All statements are reformulated in accordance with the symbol mapping to form their image statements in T_2. The resulting mapping of formulas is truth-preserving: The single axiom of T_1 relating points to lines maps to itself as an axiom of T_2, and similarly for the implicit axioms for booleans. Finally, the point constants `p1`, `p2` and `p3` map to the point constants `pa1`, `pa2` and `paext`, respectively. Here, `pa1` and `pa2` are associated with the line `la` via the `on` predicate in T_2 and `paext` is intended as a point "external to" `la`. This intention is not stated in T_2 because it is not necessary to make it explicit as yet. The individual symbol mapping relationships are expressed using *maplet* notation:

Morphism s_1:

```
Points ↦ Points
Lines  ↦ Lines
on     ↦ on
p1     ↦ pa1
p2     ↦ pa2
p3     ↦ paext
```

Hereafter, all maplets that leave symbols unchanged will be omitted from morphism descriptions. Since every sort and operation symbol must map to something, this will not result in any ambiguities.

Morphisms $s_2: T_1 \longrightarrow T_3$ and $s_3: T_1 \longrightarrow T_4$ have a similar form, but with different mappings of point constants in place of those of s_1:

Morphism s_2:

```
Points ↦ Points
Lines  ↦ Lines
on     ↦ on
p1     ↦ pbext
p2     ↦ pb1
p3     ↦ pb2
```

Morphism s_3:

```
Points ↦ Points
Lines  ↦ Lines
on     ↦ on
p1     ↦ pc2
p2     ↦ pcext
p3     ↦ pc1
```

The morphisms s_2 and s_3 indicate that the images of the point constants `p1`, `p2` and `p3` are used differently in T_3 and T_4. In T_3, for example, 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. This is so that the morphisms can properly define the "concept blending" that forms a triangle from the three lines. This is accomplished by including exactly the objects T_1, T_2, T_3, T_4 and morphisms s_1, s_2, s_3 in a diagram Δ. A colimit for Δ has the requisite cocone, as shown in Figure 7 , 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, $\overline{\Delta}$ as shown in the figure, is commutative, with

$$\ell_4 = \ell_2 \circ s_1 = \ell_3 \circ s_2 = \ell_4 \circ \ell_3 . \tag{2}$$

Figure 8 is a pictorial illustration of the colimit defining diagram of Figure 7 . The illustration depicts the contents of the concepts involved along with the diagram structure. The resulting colimit object, the specification T_5, is as follows (see the Appendix for a complete exposition of the concepts and morphisms):

```
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
```

Spec T_5 is a "blending" or "pasting together" of T_2, T_3 and T_4 along their common sub-concept T_1. This is because of the commutativity of the defining diagram $\overline{\Delta}$ of the colimit. For the equality (2) 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, we have re-assigned the name of the common T_1 symbol to the merged-image symbol in T_5 for each such case. 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 appears in the definition of two different lines. This is because each of them appears in concept T_1 but is mapped to three points, one in each concept T_2, T_3, T_4, via the three morphisms from T_1 to those concepts. In two of these concepts, its image point appears in the definition of a line, but as a different point on a different line in each concept. In the remaining concept, it appears as an "external" point, not on the line named in that concept. For example, `p1` 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`. See concept T_5, pictured in Figure 8 , for an illustration of the triangle. The other concepts pictured in the defining diagram yield the names of the various constants which together yield the vertices and sides of the triangle.

A theorem in category theory can be used to derive an algorithm for calculating colimits in any category that contains colimits for all of its diagrams (the dual to The Limit Theorem—see [39]). The category **Concept** is one such category. Thus, the apical object T_5 and leg morphisms ℓ_1, ℓ_2, ℓ_3, and ℓ_4 can be derived automatically from the objects and morphisms of the base diagram, Δ. This confers a great advantage on the use of category theory in knowledge-based system development. Theories and their morphisms (or formal specifications and their

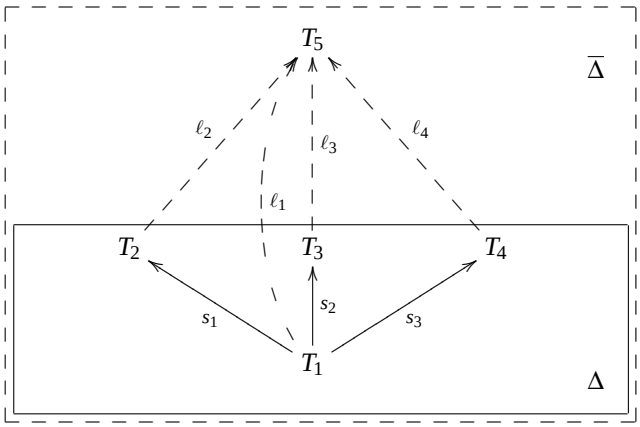


Figure 7: **A colimit for the diagram Δ in the triangle geometry example in category Concept. The extended diagram $\overline{\Delta}$ extends the point-and-line concepts in Δ to form a concept about triangles formed from triples of lines in the apical concept of the colimit, T_5, as discussed in the text.**

morphisms) can be used to specify the intended semantics of software or other kinds of system components. The colimit calculation and the structure-preserving mappings of category theory together provide a mathematically rigorous as well as automated technique for constructing the full system from diagrams[53].

Here, we shall apply the same kind of mathematics, but regard the theories as concepts that describe the stimuli that activate the nodes of a neural network, and the morphisms as relationships that describe the embedding of one concept in another by virtue of the transmission of stimuli through paths of connections in a network. This will provide an analytical framework for determining, unambiguously and precisely, when a neural architecture has the capability to learn a hierarchy of concepts of varying degrees of complexity. It will also explicate the combining of information from multiple sensors and the integration of cognitive functions such as memory storage and retrieval, planning, and decision-making. It will make it possible to determine why a network generates the input-to-output behaviors it does, and how to design a network with more desirable behavior.

3.3 Model Spaces

Each concept T has an associated space of *instances*. An instance can be thought of as a situation described by T, where the entities and relationships of the situation satisfy the axioms of T. For example, an instance of the colimit theory T_5 just discussed consists of any geometrically-describable arrangement of entities satisfying the theory axioms and in which the constants $\texttt{p1}, \texttt{p2}, \texttt{p3}, \texttt{la}, \texttt{lb}, \texttt{lc}$ serve as labels for three distinct entities considered to be points and three distinct entities considered as lines, each one at a specific location in some "space", real or imagined. Examples of these "spaces" include the Euclidean plane (imagined) and a collection of pebbles on a sidewalk, arranged in intersecting straight lines so as to satisfy the theory axioms (real). A consequence of the axioms of T_5 that involve its point and line constants is that three lines labelled $\texttt{la}, \texttt{lb}, \texttt{lc}$ form a triangle with points labelled $\texttt{p1}, \texttt{p2}, \texttt{p3}$, at their intersections as specified. An instance of T_1 is any example of the geometry of points and lines, and an instance of any one of T_2, T_3 or T_4 is an example of geometry augmented by a line defined by two points and a third point not specified as being on the line, with all quantities labelled appropriately.

Instances of theories can be related, as the theories are related, by morphisms. The triangle example provides an illustration. An instance of the geometry theory T_5, which describes a triangle axiomatically, consists of

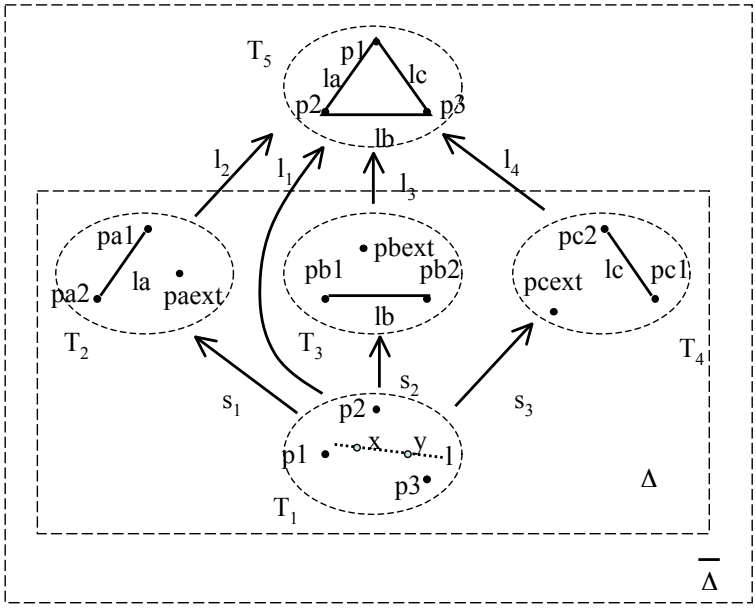


Figure 8: **A pictorial illustration of the colimit base and defining diagrams Δ and $\overline{\Delta}$. The contents of the concepts involved are pictured along with the diagrammatic structure. Solid dots and lines signify point and line constants, respectively, labelled as in the corresponding concepts. Certain point constants are equivalenced by the composition morphisms with domain T_1 and codomain T_5, all of which are equal since $\overline{\Delta}$ commutes. For example,** `pa1`, `pbext` **and** `pc2` **are all mapped to the triangle vertex** `p1` **in T_5 by the colimit leg morphisms ℓ_2, ℓ_3 and ℓ_4 because** `p1` **in T_1 maps to** `pa1`, `pbext` **and** `pc2` **via morphisms s_1, s_2 and s_3, respectively. 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.**

some mathematically-expressable means of organizing the space containing its entities— for example, points and straight lines drawn on paper on the Euclidean plane or pebble arrangments on a sidewalk—together with a triangle at a specific location and orientation to the observer with the vertices and sides labelled as are the point and line constants. Each instance of T_5 corresponds to an instance of T_2, T_3 and T_4, which describe only points and lines but provide partial information about the quantities and laws governing them which are necessary for constructing triangles. Each instance of T_2, and also each instance of T_3 and T_4, corresponds to an instance of T_1. These relationships are composable: Each instance of T_5, by virtue of its correspondence with an instance of any of T_2, T_3 and T_4, corresponds to an instance of T_1 as well. This shows that there are mappings from instances of the more specific theories to instances of less specific theories embedded in them, and the mappings have a law of composition. It is also true that the less specific theories can have more instances than those that are more specific. This is because a theory is more specific when it has more sorts, operations, constants and/or axioms, any of which provide constraints upon its instances.

This leads us to our system for analyzing the instances of concepts and relating them to neural network activities. We denote the space of instances of a concept T, called *models of* T, by $\mathtt{Mod}(T)$. 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'_σ is the set of Boolean values `T`, `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 $u_\sigma, p_\sigma, c_\sigma$. In other words, the theory T must be valid for the structure σ obtained by replacing its sorts, operations and constants with the indicated quantities.

For an example of a model, consider the previously-discussed instances in the triangle example. A model σ constructed with the Euclidean plane in mind has u_σ as the set of points in the plane if u is the sort `Points`. If u is the sort `Lines`, on the other hand, then u_σ is the set of straight lines in the plane. In theory T_5, for example, the operation $\texttt{on}:\texttt{Points}*\texttt{Lines}->\texttt{Boolean}$ can be represented by a function $p_\sigma{:}u_\sigma \longrightarrow u'_\sigma$, where $u_\sigma = (\texttt{Points}*\texttt{Lines})_\sigma = \texttt{Points}_\sigma \times \texttt{Lines}_\sigma$ and $u'_\sigma = \texttt{Boolean}_\sigma$. The predicate p_σ has the value `T` for those point-line pairs having 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 quantities $x, y \in \texttt{Points}_\sigma$ satisfying the stated pre-condition: That is, there exists a unique member of $\texttt{Lines}_\sigma$ defined by x and y. The same holds in any model of the theories T_1—T_4. However, notice that there can be members σ of $\texttt{Mod}(T_1)$ that do not have specifically identified members of $\texttt{Lines}_\sigma$ for any of the line constants `la`, `lb`, `lc` because T_1 does not specify any line constants. However, specific lines serving in the roles of the three line constants must exist in any member of $\texttt{Mod}(T_5)$.

Another example can be formed from the pebbles-on-a-sidewalk example. Consider a region delineated on a sidewalk with specific positions marked for all possible placements of pebbles. A model of T_5 then has a specific layout of pebbles representing the point constants, with the point constant representatives labelled, with paint, say. It can also contain other pebbles (shaped or colored differently, perhaps) to represent all the other points on the three lines chosen to represent the line constants; the three line constant representatives can be indicated by placing an appropriately-labelled piece of tape at one end of each. The specified points and lines must be configured, collectively, to satisfy the axioms of T_5. Interestingly, notice that the pebbles could be either pixels or groups of pixels in a video image. The lines could likewise represent linear configurations of pixels, and so forth. Models constructed in this "image space" are suggestive of the use of model theory in analyzing the relationship between the processing in a neural network trained to recognize geometric objects and the structure of its input environment consisting of sensor images.

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 $s{:}T \longrightarrow T'$. In correspondence with this, 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'; it is just the case that an arbitrary model of T is not *required* to have all the structure that an arbitrary model of T' is required to have. By the same token, there can be many models in $\texttt{Mod}(T)$ that are not the image of a model in $\texttt{Mod}(T')$ under the mapping $\texttt{Mod}(s)$: *A simpler theory poses fewer restrictions on its models, hence, there can be more of them.*

We can extend the examples of models just discussed to obtain examples of model-space morphisms. In the pebbles-on-a-sidewalk example, any model σ'' in $\texttt{Mod}(T_5)$ maps to a model σ' in $\texttt{Mod}(T_2)$, thence to a model σ in $\texttt{Mod}(T1)$, via the composition $\texttt{Mod}(s_1) \circ \texttt{Mod}(\ell_2)$ of the model-space morphisms $\texttt{Mod}(\ell_2){:}\texttt{Mod}(T_5) \longrightarrow \texttt{Mod}(T_2)$ and $\texttt{Mod}(s_1){:}\texttt{Mod}(T_2) \longrightarrow \texttt{Mod}(T_1)$. At each stage, the model derived from σ'' can have the same appearance: Although models of T_2 and T_1 are not required to contain all that is specified for models of T_5, those derived directly from models such as σ'' via the model-space morphisms can retain all the structure of σ'',

or be part of a system that does so, while other models not so derived have less structure. Thus, although the model σ may retain the triangle built from pebbles as in σ'' (but perhaps with differently-colored pebbles and perhaps with some deleted), other members of $\mathtt{Mod}(T_1)$ need have no triangle, nor any specified lines. However, they must have at least one designated pebble serving in the role of the point constants.

Notice the use of the term "model spaces" as opposed to "sets of models". As noted previously, a different amount of structure is required of arbitrary models of two theories serving as the domain and codomain of a model-space morphism. This suggests that some models in a given model space are more elaborate than others, and that a less elaborate model can be thought of as embedded in another model that happens to satisfy more constraints. In actuality, this notion is expressed through *model morphisms*, and, indeed, the model spaces $\mathtt{Mod}(T)$ are in actuality categories and the model-space morphisms $\mathtt{Mod}(s)\!:\!\mathtt{Mod}(T') \longrightarrow \mathtt{Mod}(T)$ are functors. On the other hand, we use the term "space" rather than "category" because we do not intend to explore the categorical properties of model spaces in this report. However, for completeness, it is desirable to point out that the categorical formulation applies within as well as between model spaces. Overall, there is a functor $\mathtt{Mod}\!:\!\mathbf{Concept} \longrightarrow \mathbf{Cat}$, where **Cat** is a category whose objects are categories and whose morphisms are functors. The concept object images $\mathtt{Mod}(T)$ are objects in **Cat** and the concept morphism images $\mathtt{Mod}(s)\!:\!\mathtt{Mod}(T') \longrightarrow \mathtt{Mod}(T)$ are morphisms of **Cat**. See [36] or [16] for a further discussion of model categories and [34] or [1] for foundational issues concerning the existence of a category of categories.

We shall relate the model spaces and their morphisms to neural architectures following a discussion of our categorical formulation for the latter. Several items must be discussed before doing this.

4 Neural Architectures and Neural Categories

Since the point of our categorical concept representation is to express the semantics of neural networks, we need a category within which neural structure can be represented. By neural structure, we mean not just the interconnection structure joining the nodes of a neural network, but the systematic activity of its nodes and connections in interacting with its environment. The objective is to explain the activity of any neural network arising from its inputs in terms of the concept structure that network is capable of acquiring, and also to explain how neural activity is related to the acquisition of a concept structure. In accomplishing this, a category representing a neural architecture need not include all the details of the neural network computations; it need only represent the outcome of the computations in expressing and learning concepts and concept morphisms.

4.1 Architectures

A neural network architecture A has nodes $p_i\,(i = 1, 2, \ldots, n_N)$ and connections $c_k\,(k = 1, 2, \ldots, n_C)$, where n_N, n_C are positive integers. The nodes have signal functions $\phi_i\!:\!\mathbf{R} \longrightarrow \mathbf{R}$, where $\mathbf{R}$ is the set of real numbers[1]. For example, the commonly-used sigmoid with a threshold value $\theta_{T,i}$ is

$$\xi_i = \phi_i(\theta_i) = \frac{1}{1 + e^{-(\theta_i - \theta_{T,i})}}\,(i = 1, 2, \ldots, n_N), \tag{3}$$

where the argument θ_i is an activation potential. Each connection c_k joins a pair of nodes, its *source* $p_i = \mathbf{n_S}(c_k)$ and its *target* $p_j = \mathbf{n_T}(c_k)\,(1 \le i, j \le n_N)$. If $i = j$, c_k is an *autoconnection* through which p_i either stimulates or inhibits itself, depending upon whether c_k is *excitatory* or *inhibitory*.

[1]The system $\mathbf{R}$ of real numbers is used here purely for simplicity, since it has familiarity in the connectionist literature. Certain subsets of $\mathbf{R}$ could have been specified instead to represent the domain and codomain of a signal function. In fact, the semantic model does not depend upon the use of the real number system at all: functions defined over the rationals, the complex numbers, or other algebraic structures with additive and multiplicative structures can be used instead, with the proper care taken to ensure consistency of the resulting analysis. For example, $\mathbf{R}$ is a complete, ordered field; many other algebraic systems do not have this combination of properties. One result of this is that the intervals used here in defining neural objects must be replaced by other kinds of sets when using these systems.

In the usual discrete neural network model, such as the perceptron, the activation potential θ_i is

$$\theta_i = \Sigma_{\mu \in d_i} w_\mu \delta_\mu \tag{4}$$

where each δ_μ is an output signal ξ_ℓ emitted by some node $p_\ell (\ell \in d_i)$ where d_i is the set of input connections to p_i, $\delta_\mu = \xi_\ell^- = \phi_\ell(\theta_\ell^-)$, and $p_\ell = \mathbf{n_S}(c_\mu)$, $p_i = \mathbf{n_T}(c_\mu)$, and w_μ is the current weight value for connection $(c_\mu$. A fixed set of *input nodes* $p_h (h \in S_I \subseteq \{1, \ldots, n_N\})$ is typically defined for an architecture, and in many cases also a set of *output nodes* $p_\omega (\omega \in S_O \subseteq \{1, \ldots, n_N\})$. Input nodes require an additional term in their input sum to account for the external input, *viz.*,

$$\theta_i = \Sigma_{\mu \in d_i} w_\mu \delta_\mu + I_i. \tag{5}$$

In architectures such as perceptrons, input nodes receive external inputs only; in this case, the input node has input sum

$$\theta_i = I_i. \tag{6}$$

As soon as the current activation values θ_i for the nodes are computed, the signal functions are evaluated to compute an output,

$$\xi_i = \phi_i(\theta_i). \tag{7}$$

A neural architecture A has an infinite number of possible weight states w. A weight state is an n_C-tuple $(w_1, w_2, \ldots w_{n_C})$ of numerical values w_k for the connections $c_k (k = 1, 2, \ldots, n_C)$. The network transitions from one weight state to another by virtue of its conectionist learning algorithm, usually based upon some variant of Hebb's law[24]. Activation and learning in a neural network occur simultaneously in response to the input patterns the network processes. Activation and learning are combined mathematically to define global state transitions as part of the definition of a neural category. Before discussing neural categories, let us explain in more detail what it means to have a semantic model for neural networks.

4.2 Neural Processing and Semantics

A basic assumption of any neural network model is that a stimulus pattern I presented to the input nodes represents an event — an object, entity, situation, visual scene, or some representative of the network's environment that has significance "for the network". Another assumption is that the representation is consistent across all events; that is, the algorithm or sensing method that converts events to input patterns remains fixed as the input stream progresses. These assumptions are fundamental to the analyst's ability to make sense of the phenomena observed as a network processes its input stream, whether the network is artificial or biological. In the analyst's view, each input pattern I associates, or bind, the network to its environment. The network is supposed to adapt from a sequence of inputs so that it can respond to future inputs from the environment in a manner that the analyst considers reasonable. Semantic modeling is a vehicle for understanding this process by employing the precision of a mathematical language to describe the contents of the information supplied by environmental events as represented by the input patterns, the consequent representation of the environment learned by the network and stored in its connection weight array, and the outputs the network generates. The explicit representation of knowledge about entities and their properties and relationships contained in the descriptions can make visible the assumptions implicit in a neural network model so that they, too, become amenable to analysis.

To even begin to understand the information content of events and the consequent neural representation of the environment, we need an unambiguous, precise language capable of expressing the properties of and relationships among arbitrary entities. To be useful in analysis, the language must be flexible, able to express the properties of many entities in combination and the properties of complex entities in terms of the properties and relationships of their parts. For example, the property of balance in a painting arises from the spatial relationships of the shapes, colors and textures within it; a blue patch of water to the lower left is reflected in a scattering of blue flowers on the river bank at the upper right, and so forth. In many cases involving the "emergent" property of a system of entities, on the other hand, the association of the system property with those of its components may

not be obvious on the surface, but if it arises systematically the association must be expressible mathematically. In any case, the key to semantic modeling is to be able to associate, again unambiguously and with precision, the descriptions in our language with the structure and operations of a computational entity such as a neural network. An important notion we use for this association is that the description associated with an event can be seen as a part of a longer, more detailed description associated with that which produced the event. Thus, the descriptions associated with the inputs to the network exist at various levels of detail, from the abstract to the specific. Furthermore, the network, given a sufficiently rich input format, can reconstruct much of the complexity of the environment in its internal representation of the information it gains from the input representation of the events. Again, the descriptions of simple and complex entities are related. We wish to associate this hierarchy of descriptions with the learning of environmental representations and their storage in and retrieval from a neural network's connection-weight memory.

This line of thought has two consequences for our purposes: First, one or more of the pattern values in an input I represent a concept, a description of something which, when sampled (say, by apprehending examples of it), generates events. A single pattern may represent several such concepts. Second, a concept has sub-concepts, or embedded concepts, representing different aspects of the event-generator. For example, the pattern values in I represent pieces of information about an event, such as boundary segments of shapes appearing in visual images; each is describable as a boundary segment, an aspect of a boundary; each description is a concept. The full boundary helps to delineate a shape appearing in the image, so it is itself an aspect (the shape) of a yet more complex object captured in the image. Other aspects of the object are its colors, textures, and shading or the indication of three- dimensionality; all have descriptions, and all descriptions are concepts. The boundary segment concepts are sub-concepts of the boundary shape concept, and it, in turn, is a sub-concept of a full description of an object, along with other sub-concepts describing color, texture and so forth. Further, an object is more than it appears in a visual image: It can be an entity that also produces sounds, has a touch and feel, and other properties; therefore, the visual object concept is a sub-concept of a complex concept describing the whole object. Objects of any possible kind may exist in a system of interacting entities, such as people in a social situation or vehicles in a traffic network. The individual object concepts are therefore sub-concepts of a description of a system, which also contains descriptions of the ways in which individuals can interact. From this, it is evident that concepts normally exist in a hierarchy given by a sub-concept relation. This is a re-statement of the fact that the descriptions of simple and complex entities are related.

A concept can be transformed significantly when used as a sub-concept in more complex concepts, or it can be used in several ways. Therefore, although we often describe a concept relation as a sub-concept relation, the term "concept morphism" is more generally applicable. This brings us to the use of the category **Concept** for representing the semantics of a neural network. *A network forms an internal representation of the concept category to greater or lesser extent depending upon its ability to learn and represent concepts at all levels of the hierarchy*. The part of the category that it can fully represent in its connection weight array at any point in its learning history constitutes a "knowledge base", one which it updates incrementally. Our objective is to provide a mathematical framework that explains the output responses and the connection weights of an arbitrary neural network in terms of an internal knowledge base, and explains the weight modifications in terms of updates to the knowledge base. This explanation is an understanding of the semantics of the neural network.

If the connection weight changes are to be effective in accumulating the knowledge implicit in a stream of input patterns, they must incorporate previously obtained information in the formulation of new information as each pattern is processed. This systematic change, therefore, involves the re-use of prior knowledge and can be seen as the derivation of concepts from previously-learned concepts. Each network response to an input pattern either represents a system of concepts pre-existing in the network's internal knowledge base or else prescribes the derivation of a concept new to the network's representation. A prescription for a derivation is represented by the activated nodes and connections between them that represent pre-existing concept representations and relationships between them, beginnning with the input nodes. This can be represented by the mapping of a diagram in **Concept** to a diagram in a neural category, once we have defined what is meant by a neural category.

We measure the activation values θ_i of the nodes $p_i\,(i = 1, 2, \ldots, n_N)$ of a neural network indirectly, by observing their outputs $\xi_i = \phi_i(\theta_i)$. The value ξ_i represents a concept in the existing knowledge base. For

example, a particular output value ξ_h for an input node p_h might be associated with a concept such as "event X produces a horizontal contour segment at location L in the image", where the "strength" of the contour at location L is associated with a range of output values for p_h. Leaving open the possibility that a range of values ξ_h might be associated with a horizontal contour segment occurring at location L for "noise immunity", it is best to associate the contour segment at L with a set η of values $\eta = \{\xi \mid \ell < \xi < u\}$ (if ξ_h takes real values, η is a real interval and may be arbitrarily small). The occurrence of a value lying within η signifies an instance of the concept of a horizontal bar occurring at location L. Thus, in general, concepts about events are represented by node/interval pairs (p_i, η), where η is a range (for example, a real interval) of output values for p_i.

Now, for any node p_i, we can trace backward through its input connections in the network, eventually determining how its output value ξ_i is formed by combining and transforming input node values through the weighted connection pathways leading to it. If we can describe how this combining works in terms of the input node/interval concepts, then, outputs ξ_i for node p_i can be related to a derivation of the concept associated with them in terms of concepts and relationships associated with the nodes and connections relating p_i to nodes such as p_h. As with the input nodes, the values ξ_i are best regarded as points within a set (for example, a real interval) η.

Examining this from the opposite perspective, what tells us that (p_i, η) represents a given concept T, or that it represents any concepts at all? For example, suppose T describes the layout of a room seen from some viewpoint (e.g., "The room contains a chair and a table. The chair sits behind the table.") We might be led to the supposition that (p_i, η) represents T by noticing that an output ξ_i for p_i falls within the interval η when an input pattern representing a camera image of the room is presented to the input nodes. However, this conclusion is not necessarily an accurate assertion about the semantics of (p_i, η). For if p_i is an input node, it could simply be that a particular region of the camera image yields a segment of an image contour that indicates the presence of the chair, or the table, but does not describe either entity in any more detail than that. The same contour segment (such as a horizontal bar) could just as easily describe any of hundreds of other items in the room, or in other, completely unrelated camera images. To represent the entire room view, (p_i, η) must represent "downstream" processing of the image contour segment together with many other image features.

The need to establish the semantics of quantities such as (p_i, η) unambigously and with precision is the reason for having a mathematical semantic model. The outputs a node p_i will generate at a given time depends upon its connection-weighted inputs, its signal function, and its current activity. We can account for the other quantities by observing the node's properties, but its inputs through connections involve its relationship with its immediate neighbors. Semantics involves relationships and, categorically, relationships are the morphisms in a category. We have a declarative basis for expressing semantics with concept morphisms; to understand the declarative semantics of a neural network, we need to have a mathematical model that includes neural morphisms and how the two kinds of morphisms are related.

4.3 Neural Categories

There is an infinite number of possible neural network architectures A, and each has an infinite number of possible weight states w. In analyzing or designing a specific neural network, artificial or biological, we either assume or provide a structure for A that contains enough nodes and connections to account for the appearance of previously-unused nodes or connections as the network evolves through learning or growth. In an artificial network, for example, many connections can initially be assigned weight values close to zero. Their values can then increase if they happen to be recruited as significant contributors to network processing by the learning algorithm for A. In a biological system, an increase from an initial near-zero value can be used to represent either adaptation (learning) or the growth of a new connection. In any case, the semantic model associates each combination of A and w with a category $\mathbf{N}_{A,w}$. Each input pattern e for A results in activity in its nodes and connections, and this together with the connectionist structure of A is involved in the definitions of objects and morphisms for $\mathbf{N}_{A,w}$. Weight adaptation, or "learning", changes one or more of the components w_k of w, a consequence of the node activities resulting from the processing of e. This is represented in the semantic model as a transition between categories.

Our mathematical model addresses the semantics of the network processing explicitly, but regards neural dynamics and notions of time and state change only as a means of carrying out the computations involved. We shall assume that the computations proceed, and confine our interest to the knowledge representation scheme that we use to express the semantics. To model the processing of an arbitrary neural network, therefore, we use a transformation that expresses only the neural network computational states of significance for semantic analysis. Let W_A, Θ_A, E_A denote the spaces of weight tuples, activation tuples and input tuples for A, respectively. To represent 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 maps an $n_C + n_N + n_I$-tuple of initial weights, activation values and input values to a resultant $n_C + n_N$-tuple of weights and activation values. It expresses the totality of neural processing of the input via weighted summation of the inputs at each node, the signal function evaluations $\xi_i = \phi_i(\theta_i)$, and any other assumptions embodied in the rules for activation in A, including feedback, recurrence, and connection weight adaptation. It is meant to represent the processing in a significant computational step in an arbitrary architectural model, whether discrete or continuous. We write $(v, \psi) = \Phi_A(w, \theta, e)$, with $w, v \in W_A; \theta, \psi \in \Theta_A; e \in E_A$.

Based upon the discussion in the previous sub-section, 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 that has some significance in our analysis. There may be many such output sets, so a single node can be associated with many objects, depending upon the needs of the analysis to be perfomed. In any case, we refer to the node p_i as the *carrier* of any object of the form (p_i, η).

Let two concepts T_1, T_2 be represented by objects $(p_i, \eta), (p_j, \eta')$, respectively, in the category $\mathbf{N}_{A,w}$. Although the morphisms of $\mathbf{N}_{A,w}$ have not been defined as yet, one would expect that any concept morphisms between T_1, T_2 would be represented as morphisms of $\mathbf{N}_{A,w}$ by virtue of the activity in the weighted network connections between the nodes p_i, p_j. To begin relating activity in a neural architecture to morphisms in its associated neural category, we define a *signal path* γ with source (p_i, η) and target (p_j, η') as a connection path $c_{k_1}, c_{k_2}, \ldots c_{k_n}$ with nonzero weights, $w_{k_r} \neq 0$ $(r = 1, \ldots, n)$, together with a sequence of nodes $p_{\mu_1}, p_{\mu_2}, \ldots, p_{\mu_n}, p_{\mu_{n+1}}$ and intervals $\eta_1, \eta_2, \ldots, \eta_n, \eta_{n+1}$, where $p_i = p_{\mu_1} = \mathbf{n_S}(c_{k_1})$, $p_{\mu_r} = \mathbf{n_S}(c_{k_r})$ and $p_{\mu_{r+1}} = \mathbf{n_T}(c_{k_r})$ $(r = 1, \ldots, n)$, $p_j = p_{\mu_{n+1}} = \mathbf{n_T}(c_{k_n})$. Thus, a signal path is a connection path between (p_i, η) and (p_j, η') which, if it has any intermediate nodes (i.e., if there is more than one connection in the path), has specified output intervals for those nodes. We can represent γ by the string $[(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, (p_{\mu_3}, \eta_3), \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1})]$, which shows the layout of objects and connections along the path. The path has source and target objects $(p_i, \eta) = (p_{\mu_1}, \eta_1) = \mathbf{o_S}(\gamma)$ and $(p_j, \eta') = (p_{\mu_{n+1}}, \eta_{n+1}) = \mathbf{o_T}(\gamma)$, the others being referred to as intermediate objects.

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, as before, the number of nodes in the input node set S_I, with $n_I = \mathtt{card}(S_I) \leq n_N$. Matching input pattern components to the input nodes that receive them is done by associating each input pattern index $h (1 \leq h \leq n_I)$ to a unique node index $i (1 \leq i \leq n_N)$ such that $e_h = I_i$, where p_i is the appropriate input node. When the nodes p_{μ_r} in a path γ produce outputs lying within the intervals η_r, where $\gamma = [(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, (p_{\mu_3}, \eta_3), \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1})]$, 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 $(p_j, \eta') = \mathbf{o_T}(\gamma)$, together with a weight state w, has an associated set $U_{\gamma,w}$ of pairs $(\theta, e) \in \Theta_A \times E_A$ satisfying the following requirements:

1. For all nodes p_{μ_r} of γ $(r = 1, \ldots, n)$, $\phi_r(\theta_r), \phi_r(\psi_r) \in \eta_r$, where for some $v \in W_A$, $(v, \psi) = \Phi_A(w, \theta, e)$.
2. For every connection (c_{k_r}) of γ, $v_{k_r} \neq 0$ $(r = 1, \ldots, n)$.

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 pairs (θ, e) in $U_{\gamma,w}$ are combinations of initial activation values and inputs for the network whose subsequent processing does not change the objects that are activated along the path γ, even with accompanying changes in some or all network connection weights (as long as the weights do not become zero). The elements $(\theta, e) \in U_{\gamma,w}$

are called *instances of* γ *in weight state* w. The totality of elements (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 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 $U_{\Gamma,w}$. This is obtained by simply forming the intersection $U_{\Gamma,w} = \bigcap U_{\gamma,w}(\gamma \in \Gamma)$. We say that two path sets Γ and Γ' with a common source and target are *equivalent* in weight state w, denoted $\Gamma \equiv_w \Gamma'$, 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. Thus, when $\Gamma \equiv_w \Gamma'$, we can write $\overline{\Gamma_w} = \overline{\Gamma'_w}$. A path set of the form $\overline{\Gamma_w}$, where Γ is an arbitrary path set, is called *closed in weight state* w. Notice, finally, that $U_{\overline{\Gamma_w},w} = U_{\Gamma,w}$.

We are now ready to say what we mean by a neural morphism. A morphism $m\colon (p_i, \eta) \longrightarrow (p_j, \eta')$ in a neural category $\mathbf{N}_{A,w}$ is given by the following:

1. A domain object (p_i, η),
2. a codomain object (p_j, η'),
3. a path set Γ having $(p_i, \eta) = \mathbf{o_S}(\gamma)$ and $(p_j, \eta') = \mathbf{o_T}(\gamma)$ $(\gamma \in \Gamma)$, and
4. the set $U_{\Gamma,w}$ of pairs (θ, e), with $(\theta, e) \in U_{\Gamma,w} \subseteq \Theta_A \times E_A$, where for each path $\gamma \in \Gamma$ with $\gamma = [(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, (p_{\mu_3}, \eta_3), \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1})]$, we have $\phi_{\mu_r}(\theta_{\mu_r}) \in \eta_r$, $\phi_{\mu_r}(\psi_{\mu_r}) \in \eta_r$ for $(r = 1, \ldots, n)$, where $(v, \psi) = \Phi_A(w, \theta, e)$ for some $v \in W_A$.

Any set Γ' with $\Gamma \equiv_w \Gamma'$ defines the same morphism. Thus, a morphism can be represented by a weight state w and any of its path sets $\Gamma, \Gamma', \ldots$, which have the same source and target nodes and are associated with the same set of instances, $U_{\Gamma,w} = U_{\Gamma',w} = \ldots$. Given any of its equivalent path sets Γ and the weight state w of its category, we refer to a morphism m as *the morphism associated with the pair* (Γ, w), and we write $U_m = U_{\Gamma,w}$ (the dependence of U_m on w is appropriate because a morphism is specific to its category, in this case $\mathbf{N}_{A,w}$). Conversely, each of its path sets is called a *carrier* for the morphism in weight state w.

The definition of composition for morphisms in a category $\mathbf{N}_{A,w}$ is really rather obvious, given our notions of what constitutes a morphism. Consider a pair of morphisms m_1, m_2, with a path set $\Gamma_1, \Gamma_2,$, respectively for each, with associated sets $U_{\Gamma_1,w}, U_{\Gamma_2,w}$ of instances, where $\mathbf{o_S}(\Gamma_2) = \mathbf{o_T}(\Gamma_1)$. 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. For example, if
$\gamma = [(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, (p_{\mu_3}, \eta_3), \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1})]$ and
$\gamma' = [(p_{\mu_{n+1}}, \eta_{n+1}), c_{k_{n+1}}, (p_{\mu_{n+2}}, \eta_{n+2}), c_{k_{n+2}}, (p_{\mu_{n+3}}, \eta_{n+3}), \ldots, (p_{\mu_{n+n'}}, \eta_{n+n'}), c_{k_{n+n'}}, (p_{\mu_{n+n'+1}}, \eta_{n+n'+1})]$, then

$$\begin{aligned}\gamma;\gamma' = [&(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, (p_{\mu_3}, \eta_3), \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1}),\\ &c_{k_{n+1}}, (p_{\mu_{n+2}}, \eta_{n+2}), c_{k_{n+2}}, (p_{\mu_{n+3}}, \eta_{n+3}), \ldots, (p_{\mu_{n+n'}}, \eta_{n+n'}), c_{k_{n+n'}}, (p_{\mu_{n+n'+1}}, \eta_{n+n'+1})].\end{aligned}$$

With these definitions, the pair (Γ_3, w) uniquely defines a morphism $m_3\colon (p_{\mu_1}, \eta_1) \longrightarrow (p_{\mu_{n+n'+1}}, \eta_{n+n'+1})$, that is, $m_3 = m_2 \circ m_1$, having as one of its path sets $\Gamma_3 = \{\gamma;\gamma' \mid \gamma \in \Gamma_1, \gamma' \in \Gamma_2\}$, with $U_{\Gamma_3,w} = U_{\Gamma_2,w} \cap U_{\Gamma_1,w}$ and with $(p_{\mu_1}, \eta_1) = \mathbf{o_S}(\Gamma_3)$ and $(p_{\mu_{n+n'+1}}, \eta_{n+n'+1}) = \mathbf{o_T}(\Gamma_3)$. The operation $\circ$ for morphisms associated with the concatenations of path sets, such that the source of one set is the target of the other, can always be defined and yields a unique result.

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 $m\colon (p_i, \eta) \longrightarrow (p_i, \eta)$ 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 $\{[(p_i,\eta),(p_i,\eta)]\}$, that is, $\Gamma \equiv_w \{[(p_i,\eta),(p_i,\eta)]\}$, has the same instances, hence, is acting collectively as the virtual path, $U_{\Gamma,w} = U_{\mathtt{id}_{(p_i,\eta),w}}$. Notice that the instance set of an identity morphism is just the instance set of its object, $U_{\mathtt{id}_{(p_i,\eta),w}} = U_{(p_i,\eta)}$.

4.4 Example: A Simple Multi-Layer Perceptron Network

Consider a category $\mathbf{N}_{A,w}$ representing a multi-layer feedforward network with weight state w. Here, the function Φ_A represents all processing that occurs in a single iteration of a learning algorithm for feedforward networks such as the perceptron algorithm[35]. For simplicity, let the nodes p_i act as binary nodes in representing concepts, where positive outputs signify events associated with a concept represented by the node; all outputs not sufficiently positive are regarded as having either uncertain or no significance. Given this one-to-one correspondence between nodes and objects, substituting p_i for (p_i,η) simplifies notation, and any output $\xi_i = \phi_i(\theta_i) > 0$ is an instance of the object p_i.

Figure 9 shows a part of the MLP network, with selected nodes in what we shall refer to as layers $n, n+1$ and $n+2$, and selected connections between them. Although there can be many other nodes present, we refer to the nodes shown in the figure as p_1 (in Layer n), p_2 and p_3 (in Layer $n+1$) and p_4 (in Layer $n+2$). The connections are also labelled as, for example, c_1 (from p_1 to p_2). From the definition of neural morphism just given, there is a morphism corresponding to the set $\overline{\Gamma_1}_w$, where Γ_1 contains a path associated with the single connection c_1, that is, $\gamma_1 = [(p_1,\eta),c_1,(p_2,\eta)]$. In fact, since the architecture is a feedforward network and c_1 connects nodes across adjacent layers (n and $n+1$), γ_1 is the only possible member of $\overline{\Gamma_1}_w$, hence, of Γ_1, unless we allow there to be multiple connections between two nodes. Denote the morphism associated with the pair (Γ_1,w) by $m_1\colon(p_1,\eta) \longrightarrow (p_2,\eta)$. Because of our one-to-one node-to-object representation, we can simplify the notation for our example and instead write $\gamma_1 = [p_1,c_1,p_2]$ and $m_1\colon p_1 \longrightarrow p_2$.

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 $U_{\Gamma_1,w}$. Since $\mathbf{o_S}(\Gamma_1) = (p_1,\eta) = p_1$ and $\mathbf{o_T}(\Gamma_1) = (p_2,\eta) = p_2$, these are instances of both (p_1,η) and (p_2,η) (that is, $\xi_1 = \phi_1(\theta_1) \in \eta$, $\xi_2 = \phi_1(\theta_2) \in \eta$, and similarly for the outputs evaluated at ψ, where $(v,\psi) = \Phi_A(w,\theta,e)$). Note that it matters not whether the weight w_1 of c_1 is $w_1 > 0$ or $w_1 < 0$, only that w_1 (and also v_1) is nonzero and that the nodes along the path γ_1 (p_1 and p_2) are generating outputs within their specified intervals (the interval η).

Again using the fact that each node corresponds to a single object in the current example, we can express the paths corresponding to connections c_2, c_3, and c_4 as $\gamma_2 = [p_1,c_2,p_3]$, $\gamma_3 = [p_2,c_3,p_4]$ and $\gamma_4 = [p_3,c_4,p_4]$, respectively. Assuming that there is at most one connection between each pair of nodes in adjacent layers, the path sets $\Gamma_2,\Gamma_3,\Gamma_4$ 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 $\xi_1,\xi_3 \in \eta$ (i.e., $\xi_1 = \phi_1(\theta_1) > 0, \xi_3 = \phi_1(\theta_3) > 0$) for the source and target nodes simultaneously for γ_2, $\xi_2,\xi_4 \in \eta$ for γ_3, and $\xi_3,\xi_4 \in \eta$ for γ_4. This describes the morphisms associated with the connections c_1,c_2,c_3,c_4, which we shall call $m_1\colon p_1 \longrightarrow p_2$, $m_2\colon p_1 \longrightarrow p_3$, $m_3\colon p_2 \longrightarrow p_4$, and $m_4\colon p_3 \longrightarrow p_4$, respectively.

Now let us examine some compositions and characterize any commutative diagrams that might involve the objects and morphisms associated with the array of connections c_1,c_2,c_3,c_4. Concatenating paths, we have paths $\gamma_5 = [p_1,c_1,p_2,c_3,p_4]$ and $\gamma_6 = [p_1,c_2,p_3,c_4,p_4]$, two separate paths with the same source and target. Letting $\Gamma_5 = \{\gamma_5\}$ and $\Gamma_6 = \{\gamma_6\}$, with $U_{\Gamma_5,w} = U_{\Gamma_3,w} \cap U_{\Gamma_1,w}$, $U_{\Gamma_6,w} = U_{\Gamma_4,w} \cap U_{\Gamma_2,w}$, we have morphisms $m_5 = m_3 \circ m_1$ and $m_6 = m_4 \circ m_2$. These are two morphisms with the same domain and codomain, $m_5\colon p_1 \longrightarrow p_4$, associated with the pair (Γ_5,w), and $m_6\colon p_1 \longrightarrow p_4$, associated with the pair (Γ_6,w). Now, were it the case that $U_{\Gamma_5,w} = U_{\Gamma_6,w}$, i.e., that $\Gamma_5 \equiv_w \Gamma_6$, then they would be one and the same morphism with domain p_1 and

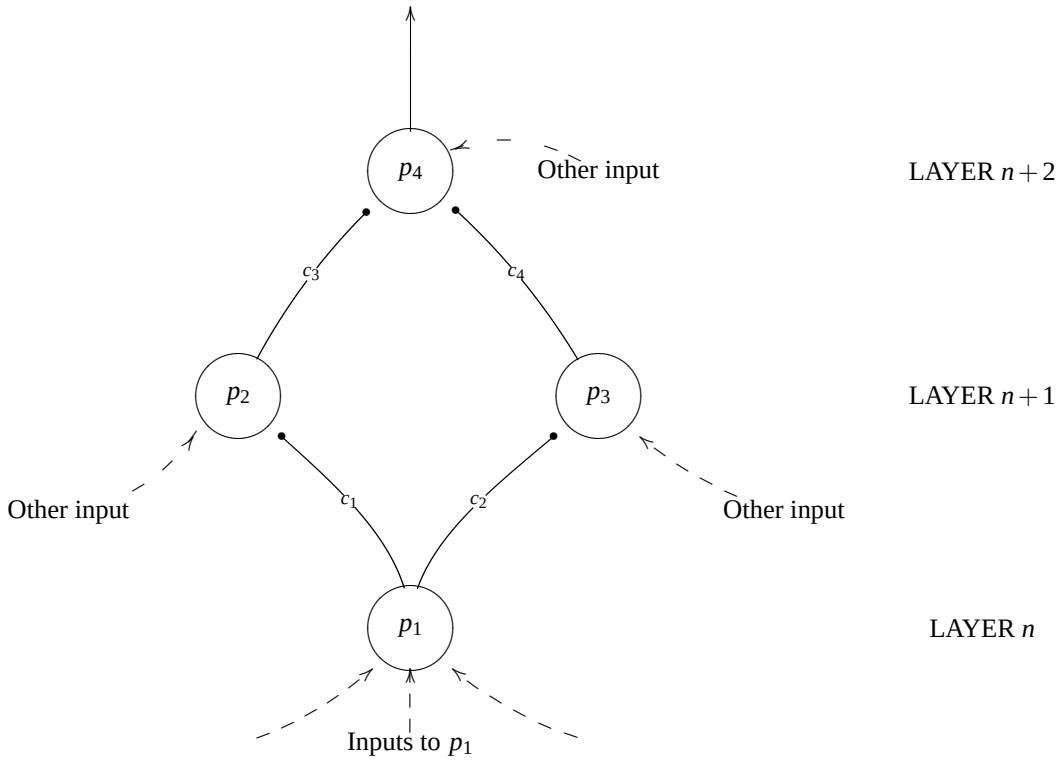


Figure 9: **Two connection paths corresponding to a commutative diagram. The bottom pair of connections is seen emanating from a single node p_1 serving as the carrier for diagram object (p_1, η_1). Similarly, the top pair of connections is seen terminating upon a single node p_4 serving as the carrier for diagram object (p_4, η_4). The commutativity of the diagram indicates that the two paths are involved in an instance of the same morphism with domain (p_1, η_1) and codomain (p_4, η_4). All instances that define the diagram are instances of this morphism.**

codomain p_4, that is, it would be the case that $m_7 = m_5 = m_6 : p_1 \longrightarrow p_4$. However, in general, this will not prove to 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, it can happen that some instances of m_5 are not instances of m_6 because they are instances of p_2 but not p_3; 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 examplifies an important fact: A set of paths having the same source and target objects can be involved in several different morphisms and, in fact, can be involved in several different diagrams. Further, some of these diagrams can be commutative while others are not. This is to say, although 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), Γ can also be involved in separate morphisms defined by larger path sets that contain it. The diamond-shaped diagram defined by $m_1, m_2, m_3,$ and m_4 need not be commutative if we define these morphisms using the instance sets $U_{\Gamma_1,w}, U_{\Gamma_2,w}, U_{\Gamma_3,w}, U_{\Gamma_4,w}$ as in this example. However, 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: Simply let Γ_7 be the union of path sets $\Gamma_7 = \Gamma_5 \cup \Gamma_6 = \{\gamma_5, \gamma_6\}$. The instances of this union are the instances of the intersection, $U_{\Gamma_7,w} = U_{\Gamma_5,w} \cap U_{\Gamma_6,w}$. This defines a morphism $m_7 : p_1 \longrightarrow p_4$ uniquely associated with the pair (Γ_7, w). In case the intersection is empty, $U_{\Gamma_5,w} \cap U_{\Gamma_6,w} = \emptyset$, m_7 is referred to as a vacuous morphism, having no instances. In any case, it is true that $U_{\Gamma_7,w} \subseteq U_{\Gamma_5,w}$ and $U_{\Gamma_7,w} \subseteq U_{\Gamma_6,w}$. The subset relationship is proper in either or both cases

unless the diagram commutes. Thus, there is a morphism associated with the two paths through the connections $c_1, c_2, c_3,$ and c_4 regardless of whether the diagram involving $m_1, m_2, m_3,$ and m_4 is commutative.

Without being too specific about the total connectivity structure or the learning algorithm of our perceptron-like network example, or even the total range of neural computation represented by the activation function Φ_A used in defining morphisms, we have shown how morphisms, diagrams and commutative diagrams can be found in neural network architectures. We have shown how to distinguish between a diagram and a commutative diagram, given the knowledge of whether or not certain path sets in the network have equal sets of instances (hence, are equivalent) with respect to the array of network weights associated with the category under consideration. With the categorical model for neural architectures in hand, we can proceed to investigate concept representation and learning.

5 Functors: Transporting Structures Across Categories

5.1 Defining Functors Incrementally

We have defined categories **Concept** for an organized system of concepts and their relationships, and $\mathbf{N}_{A,w}$ to represent the system consisting of a neural network architecture A, a weight tuple w, and the activities associated with the pair (A, w), which are represented by the network activity function Φ_A. We now describe the categorical representation of the concept system in the activities of the neural system, that is, the semantics of $\mathbf{N}_{A,w}$. Fundamental in the representation is the notion of a functor M:**Concept** $\longrightarrow \mathbf{N}_{A,w}$. It maps each **Concept** morphism $s_\kappa: T_\mu \longrightarrow T_\nu$ to an appropriate $\mathbf{N}_{A,w}$ morphism $M(s_\kappa): M(T_\mu) \longrightarrow M(T_\nu)$ with domain $(p_i, \eta) = M(T_\mu)$ and codomain $(p_j, \eta') = M(T_\nu)$. Consider the consequences of there being concept morphisms, such as $s_{\kappa'}: T_\nu \longrightarrow T_\lambda$, whose domain is the codomain of s_κ. We can form the composition $s_{\kappa'} \circ s_\kappa$, whose functorial image under M is $M(s_{\kappa'} \circ s_\kappa): M(T_\mu) \longrightarrow M(T_\nu)$. Because of the functorial property, we need to ensure that our definition of M has as a consequence that $M(s_{\kappa'} \circ s_\kappa) = M(s_{\kappa'}) \circ M(s_\kappa)$.

To see what this entails, recall that a pair (Γ, w) can uniquely represent $M(s_\kappa)$, where Γ is a set of paths of the form $\gamma = [(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, (p_{\mu_3}, \eta_3), \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1})]$. If we assign functorial images to concepts such that the domain and codomain of $M(s_\kappa)$ are $M(T_\mu) = (p_i, \eta)$ and $M(T_\nu) = (p_j, \eta')$, then the common source and target objects of all of the paths γ in Γ must be $(p_{\mu_1}, \eta_1) = (p_i, \eta) = \mathbf{o_S}(\gamma)$ and $(p_{\mu_{n+1}}, \eta_{n+1}) = (p_j, \eta') = \mathbf{o_T}(\gamma)$, respectively. The elements of the set of instances $U_{\Gamma,w}$ are the combinations of initial state and input, $(\theta, e) \in \Theta_A \times E_A$, that initiate the neural network activities of A associated with the $\mathbf{N}_{A,w}$ morphism $M(s_\kappa)$. Similarly, $M(s_{\kappa'}): M(T_\nu) \longrightarrow M(T_\lambda)$ has domain $M(T_\nu) = (p_j, \eta')$ and also a codomain $M(T_\lambda) = (p_\ell, \eta'')$, and can be represented by a pair (Γ', w) where Γ' has elements of the form $\gamma' = [(p_{\nu_1}, \eta'_1), c_{k'_1}, (p_{\nu_2}, \eta'_2), c_{k'_2}, (p_{\nu_3}, \eta'_3), \ldots, (p_{\nu_{n'}}, \eta'_{n'}), c_{k'_{n'}}, (p_{\nu_{n'+1}}, \eta'_{n'+1}),$, where $(p_{\nu_1}, \eta'_1) = (p_j, \eta) = (p_{\mu_{n+1}}, \eta_{n+1})$ (so that $\mathbf{o_S}(\gamma') = \mathbf{o_T}(\gamma)$) and $(p_{\nu_{n'+1}}, \eta_{n'+1}) = (p_\ell, \eta'')$.

By the definition of composition in the category $\mathbf{N}_{A,w}$, $M(s_{\kappa'}) \circ M(s_\kappa)$ is representable by some pair (Γ'', w), where $\Gamma'' = \Gamma; \Gamma'$ contains the path concatenations

$$\begin{aligned}\gamma;\gamma' = [&(p_{\mu_1}, \eta_1), c_{k_1}, (p_{\mu_2}, \eta_2), c_{k_2}, (p_{\mu_3}, \eta_3), \ldots, (p_{\mu_n}, \eta_n), c_{k_n}, (p_{\mu_{n+1}}, \eta_{n+1}), c_{k_{n+1}}, (p_{\mu_{n+2}}, \eta_{n+2}),\\ &c_{k_{n+2}}, (p_{\mu_{n+3}}, \eta_{n+3}), \ldots, (p_{\mu_{n+n'}}, \eta_{n+n'}), c_{k_{n+n'}}, (p_{\mu_{n+n'+1}}, \eta_{n+n'+1})],\end{aligned}$$

where we have re-labelled as follows:

$$\begin{aligned}(p_{\nu_1}, \eta'_1) &= (p_{\mu_{n+1}}, \eta_{n+1}),\\ c_{k'_1} &= c_{k_{n+1}},\\ (p_{\nu_2}, \eta'_2) &= (p_{\mu_{n+2}}, \eta_{n+2}),\\ c_{k'_2} &= c_{k_{n+2}},\\ (p_{\nu_3}, \eta'_3) &= (p_{\mu_{n+3}}, \eta_{n+3}),\end{aligned}$$

$$\begin{aligned} &\ldots, \\ (p_{\nu_{n'}}, \eta'_{n'}) = &\quad (p_{\mu_{n+n'}}, \eta_{n+n'}), \\ c_{k'_{n'}} = &\quad c_{k_{n+n'}}, \\ (p_{\nu_{n'+1}}, \eta'_{n'+1}) = &\quad (p_{\mu_{n+n'+1}}, \eta_{n+n'+1})]. \end{aligned}$$

The set of instances of $M(s_{\kappa'}) \circ M(s_\kappa)$ is the set $U_{\Gamma'',w}$, which is just the intersection $U_{\Gamma'',w} = U_{\Gamma,w} \cap U_{\Gamma',w}$. Thus, we obtain the composition $M(s_{\kappa'}) \circ M(s_\kappa)\colon (p_i, \eta) \longrightarrow (p_\ell, \eta'')$. We must take care in defining M so that each such composition of concept morphism images in $\mathbf{N}_{A,w}$ is actually the same as the image of the corresponding composition of concept morphisms, $M(s_{\kappa'} \circ s_\kappa)$. This can be done by defining M recursively so that if $M(s_\kappa)$ is representable by a pair (Γ, w) and $M(s_{\kappa'})$ is representable by a pair (Γ', w), then $M(s_{\kappa'} \circ s_\kappa)$ is representable by the pair (Γ'', w), where $\Gamma'' = \Gamma;\Gamma'$.

5.2 Example: Applying a Functor to a Diagram

Applying a functor $M\colon \mathbf{Concept} \longrightarrow \mathbf{N}_{A,w}$ to the colimit defining diagram $\overline{\Delta}$ requires the identification of neural objects and morphisms to serve as the images $M(T_1), M(T_2), M(T_3), M(T_4), M(T_5)$ and

$$\begin{aligned} M(s_1)&\colon M(T_1) \longrightarrow M(T_2), \\ M(s_2)&\colon M(T_1) \longrightarrow M(T_3), \\ M(s_3)&\colon M(T_1) \longrightarrow M(T_4), \\ M(\ell_1)&\colon M(T_1) \longrightarrow M(T_5), \\ M(\ell_2)&\colon M(T_2) \longrightarrow M(T_5), \\ M(\ell_3)&\colon M(T_3) \longrightarrow M(T_5), \\ M(\ell_4)&\colon M(T_4) \longrightarrow M(T_5). \end{aligned}$$

In addition, the functorial property requires that the resulting neural diagram, denoted $M(\overline{\Delta})$, must commute. This requires that $M(\ell_2) \circ M(s_1) = M(\ell_3) \circ M(s_2) = M(\ell_4) \circ M(s_3) = M(s_1)$. If we can find the nodes p_i and paths γ that supply the basic structural shape for the diagram, and if we can then claim that there is a connection weight array and network initial activations and inputs w, θ, e that help define the morphisms such that the diagram commutes, we can claim to have found that part of a functor consistent with the mapping of $\overline{\Delta}$ into a category $\mathbf{N}_{A,w}$ for the architecture A. To define the entire functor requires a scheme for calculating the images of arbitrary concepts and concept morphisms. The functorial property facilitates this because it enables the calculation of the images of all concept morphisms $M(u)$ obtainable by composition, where $M(u) = M(s) \circ M(t)$,given that the factors $M(s), M(t)$ are known. In particular, applying the functoriality property to define functors in this incremental fashion guarantees the preservation of diagram commutativity. Applying functoriality requires first specifying a basic collection of concept and morphism images in terms of neural objects and morphisms associated with architectural items such as input nodes and the connection paths from them to other nodes.

There are other requirements in addition to functoriality for the images of the defining diagrams of colimits and limits. Initiality and terminality are among these. For example, in addition to the commutativity of diagram $M(\overline{\Delta})$, its cocone must be initial. Now, functors that preserve initiality and terminality of colimit cocones and limit cones are of a special kind. Ensuring that the functors discussed here have this property poses an additional design constraint for neural network architectures.

But there is another constraint whose severity depends upon how explicitly we want the functor M to represent the concepts and morphisms of **Concept** through the objects and morphisms of $\mathbf{N}_{A,w}$. Requiring the representation to be very explicit places great demands upon the architectural constructions we use in A. The discussion of Section 6 explores this "explicitness" constraint in more detail.

At this juncture in our presentation, it is apparent that applying the categorical model of neural network semantics is a complex undertaking. Its use poses many constraints upon architectural analysis and design. It is

important to observe that this is an advantage more than it is a burden. Design constraints are the conceptual tools that make it possible to fully understand and to create neural network architectures based upon the knowledge structures they purport to represent.

5.3 Neural Models

A functor M:**Concept** $\longrightarrow \mathbf{N}_{A,w}$ maps concepts T, T' and morphisms s:$T \longrightarrow T'$ to neural category objects and morphisms $M(T), M(T')$ and $M(s)$:$M(T) \longrightarrow M(T')$. As discussed previously, there is also a functor $\mathtt{Mod}$:**Concept** $\longrightarrow$ **Cat** which maps the concepts and morphisms to model spaces and model-space morphisms $\mathtt{Mod}(T), \mathtt{Mod}(T')$ and $\mathtt{Mod}(s)$:$\mathtt{Mod}(T') \longrightarrow \mathtt{Mod}(T)$, respectively. The functor $\mathtt{Mod}$ is called *contravariant* because it reverses the directions of the arrows of **Concept**.

Now, the instances of a neural object (p_i, η) are pairs $(\theta, e) \in \Theta_A \times E_A$. There is a difficulty in regarding an instance (θ, e) as a model σ of a theory. As defined, instances do not have the correct mathematical form to be models: A model σ is an object of a model space $\mathtt{Mod}(T)$ and has a structure obtained by substituting sets and functions for the sorts and operations of the theory T such that the axioms of T are valid for the resulting system of sets and functions . Although not in the mathematical form of a theory model, however, an instance (θ, e) corresponds to a model σ. Consider a theory morphism such as ℓ_2:$T_2 \longrightarrow T_5$ in our triangle example. The concepts T_2 and T_5 are mapped to neural category objects $(p_i, \eta), , (p_j, \eta')$, respectively, by a functor M—for example, $(p_i, \eta) = M(T_2)$, with instances (θ, e), where $(\theta, e) \in U_{(p_i,\eta)} \subseteq \Theta_A \times E_A$. Since each such instance represents, through its component e, an input from the environment E_A, it can also be associated with a situation, or event, within the environment as represented by the sensors providing input to the network. Each event that produces a member of $U_{(p_i,\eta)}$, on the other hand, can be regarded as a model of T_2, that is, as some member σ of $\mathtt{Mod}(T_2)$. To construct this model from the environment requires a knowledge of the quantities constituting the environment sufficient to enable the proper substitutions to be made in terms of sets and functions for the quantities in the theory T_2. The model-space morphisms associated with concept morphisms such as $\mathtt{Mod}(\ell_2)$:$\mathtt{Mod}(T_5) \longrightarrow \mathtt{Mod}(T_2)$ can be indexed by neural morphism instances in similar fashion.

As important as the correspondence between the functors M and $\mathtt{Mod}$ is in fully understanding the association of neural object and morphism instances with the corresponding concept models and model-space morphisms, there is not space here to discuss it in detail. Instead, the present analysis will proceed informally. Nevertheless, using the mathematical semantic model in this way makes it possible to gain understanding and anticipate neural structures one might not have discovered otherwise.

6 Applying Category-Theoretic Design Principles

To fully represent the concept hierarchy expressed in the category **Concept** is a daunting undertaking. First of all, there is an infinite number of concepts and morphisms in **Concept** but any realizable neural architecture A is finite. Therefore, if a neural category $\mathbf{N}_{A,w}$ is to represent only that which is possible to construct from A with any weight array w, any functor **Concept** $\longrightarrow \mathbf{N}_{A,w}$ is inherently a many-to-one mapping on both objects and morphisms. This will be explored further in discussing concept compression in Section 9. An accurate representation of even a finite number of concepts in their entirety can be intractable. Many concepts are simply not representable in many architectures, and many are not represented at a given stage of learning using the current weight array w even though they are representable: The network may not have learned them yet. One of the main issues in representability is the complexity of the architecture. In the concepts T_1—T_5 of the triangle example, the axiom relating points and lines appears in all the concepts. It is a basic notion in geometry [5] and a neural network that fully expresses it must therefore possess the ability to reason about geometry, a cognitive skill apparently possessed only by highly complex systems. The many predicates and functions used in the axioms of concepts, such as the $\mathtt{on}$ predicate of the triangle example, are similarly difficult to represent in an explicit form. It is, however, possible to overcome the latter difficulty in many cases short of designing a fully cognitive neural

network as long as the representation is not required to be explicit. The triangle example serves to illustrate this as we demonstrate in this section.

The main point of this section is that the more one wishes to represent explicitly with a neural network, the more complex the design task. Our main thrust is to illustrate neural network design to represent certain concept constructions, with some items in the concepts necessarily represented only implicitly. This is consistent with the discussion of logical rules and knowledge representation in the Introduction, for knowledge represented implicitly can guide behavior in the same sense that the knowledge implicit in the design of any machine guides its operation. The desired end in neural network design is that the knowledge representation be as explicit as possible and acquired through experience, so that the machine is useful in applications requiring highly complex computations including adaptability.

6.1 Representing a Colimit with a Feedforward Network

It is instructive to attempt an association of the triangle colimit example with the learning of the triangle concept by a specific type of neural network. It is easy enough to match the shape of the triangle diagram with a diagram in a category $\mathbf{N}_{A,w}$ if one is not concerned with the other details of representation. A functorial mapping of the diagram of figures 7 8 from **Concept** to $\mathbf{N}_{A,w}$ requires only that we assume that $\mathbf{N}_{A,w}$ has the necessary objects and morphisms. The difficulties lie in showing that the architecture A has an associated function Φ_A and weight array w that allows a representation of the morphisms in terms of objects (p_i, η), signal paths γ and prior activation-input pairs (θ, e), using our previously- introduced notation. But even this is not enough: it must also be shown that w can be an output of a learning episode represented with Φ_A, that is, that for some triple (w', θ, e), we can obtain $(w, \psi) = \Phi_A(w', \theta, e)$ for some ψ. Fortunately, the main use envisioned for the semantic model is more generally in network analysis and design. The objective is then to show that the representation and learning of certain kinds of concepts is possible, not to exhibit all the details of individual concept representations and learning episodes.

The diagram shape illustrated in Figures 7 and 8 can be achieved in a category $\mathbf{N}_{A,w}$ representing a feedforward architecture such as the MLP discussed previously. This requires only a simple modification to the diamond-shaped perceptron diagram of Figure 9. The result, shown in Figure 10 , has a node p_1 in Layer n as before, but now has three nodes p_2, p_3, p_4 in Layer $n+1$ and a node p_5 shown at the apex of the structure. We intend that p_5 serve as the neural colimit object. Retaining our binary model in which nodes are treated as objects, the $p_i\,(i = 1, 5)$ serve as objects as well as nodes. We intend that p_5 serve as the neural colimit object. We further intend that the connections $c_k\,(k = 1, 6)$ be involved in the signal paths corresponding to the colimit defining diagram morphisms

$$\begin{aligned} m_1 &: p_1 \longrightarrow p_2\,, \\ m_2 &: p_1 \longrightarrow p_3\,, \\ m_3 &: p_1 \longrightarrow p_4\,, \\ m_4 &: p_2 \longrightarrow p_5\,, \\ m_5 &: p_3 \longrightarrow p_5\,, \\ m_6 &: p_4 \longrightarrow p_5\,. \end{aligned} \tag{8}$$

Specifically, morphism $m_k\,(k = 1, 6)$ is associated with a signal path set $\Gamma_k\,(k = 1, 6)$. At least one member of each path set is a single-connection path, where its connection is indicated in Figure 10. That is,

$$\gamma_k \in \Gamma_k\,(k = 1, 6)\,, \tag{9}$$

where

$$\begin{aligned} \gamma_1 &= [p_1, c_1, p_2]\,, \\ \gamma_2 &= [p_1, c_2, p_3]\,, \\ \gamma_3 &= [p_1, c_3, p_4]\,, \end{aligned} \tag{10}$$

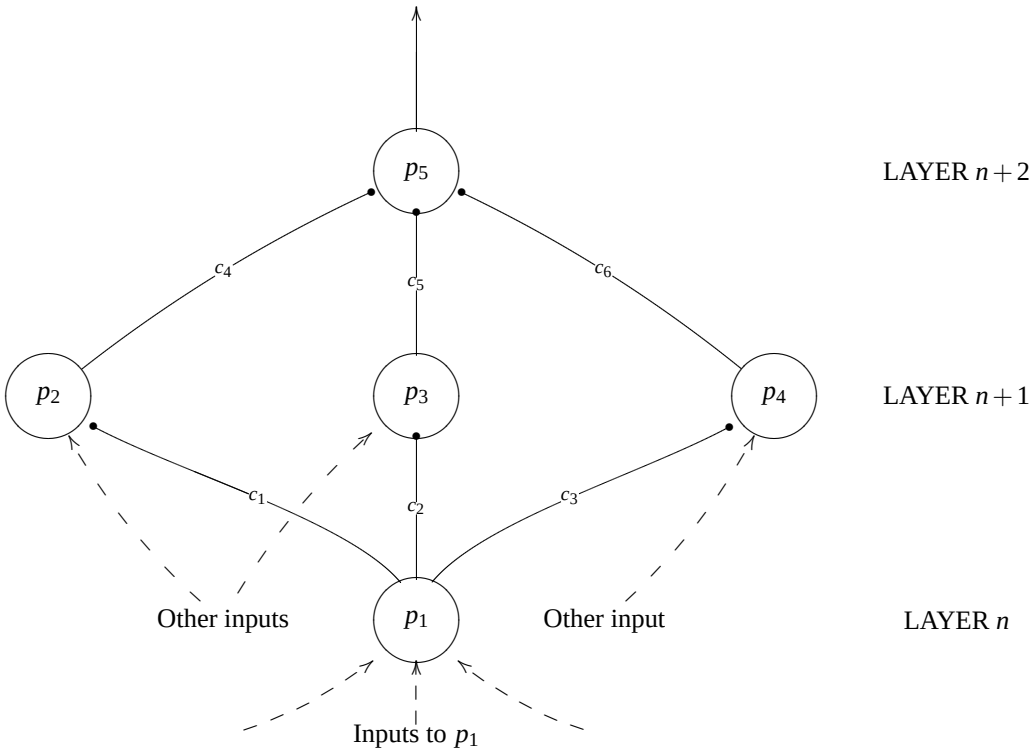


Figure 10: **Connection paths corresponding to a commutative diagram intended to serve as a neural colimit diagram in a feedforward architecture. This diagram is, in turn, intended as the functorial image of the defining diagram for the triangle concept example.**

$$\begin{aligned}\gamma_4 &= [p_2, c_4, p_5]\,,\\ \gamma_5 &= [p_3, c_5, p_5]\,,\\ \gamma_6 &= [p_4, c_6, p_5]\,.\end{aligned}$$

The neural category diagram is suggestive of the following assignments of functor M images for the concept objects and morphisms from the triangle example (see figures 8, 10 and 11):

$$p_\mu = \quad M(T_\mu)\,(\mu = 1, \ldots, 5)\,; \tag{11}$$

$$m_\kappa{:}\, p_1 \longrightarrow p_{\kappa+1} = \quad M(s_\kappa){:}M(T_1) \longrightarrow M(T_{\kappa+1})\,(\kappa = 1, 2, 3)\,; \tag{12}$$

$$m_\kappa{:}\, p_{\kappa-2} \longrightarrow p_5 = \quad M(\ell_{\kappa-2}){:}M(T_{\kappa-2}) \longrightarrow M(T_5)\,(\kappa = 4, 5, 6)\,. \tag{13}$$

For the assignments to be a part of a functor, the neural morphisms $m_1, m_2, m_3, m_4, m_5, m_6$ must form a commutative diagram, that is, there must be a neural morphism $m_7 : p_1 \longrightarrow p_5$ such that

$$m_7 = m_4 \circ m_1 = m_5 \circ m_2 = m_6 \circ m_3\,. \tag{14}$$

so that defining

$$m_7 = M(\ell_1)\,. \tag{15}$$

yields the desired result,

$$M(\ell_1) = M(\ell_2) \circ M(s_1) = M(\ell_3 \circ M(s_2) = M(\ell_4) \circ M(s_3)\,. \tag{16}$$

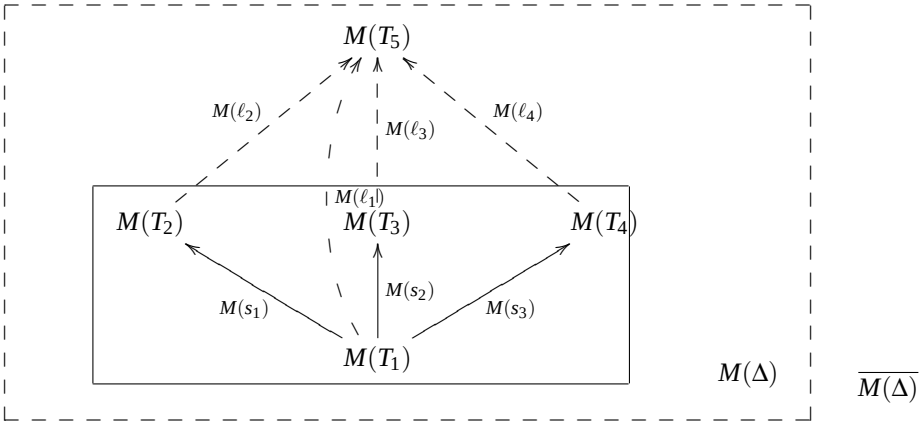


Figure 11: **The image under the functor M of the objects and morphisms of the defining diagram of the colimit for the diagram Δ in Concept, mapped to a commutative diagram in the category $\mathbf{N}_{A,w}$.**

This requires that

$$\begin{aligned} &\Gamma_7 \equiv_w \Gamma_8 \equiv_w \Gamma_9, \quad \text{where} \\ &\quad \Gamma_7 = \Gamma_1;\Gamma_4\,, \\ &\quad \Gamma_8 = \Gamma_2;\Gamma_5\,, \\ &\quad \Gamma_9 = \Gamma_3;\Gamma_6\,. \end{aligned} \tag{17}$$

It is a trivial matter to establish that a weight array w exists so that the diagram involving $m_1, m_2, m_3, m_4, m_5, m_6$ commutes if that is the only requirement. Since the MLP network example relies on binary nodes, let the node signal functions produce a value $\phi_i(\theta) = 1$ when $\theta > \theta_{T,i}$, and arbitrarily assign $\theta_{T,i} = 0.5$ for each node. Then, p_i is activated when

$$\theta_i = \Sigma_{\mu \in d_i} w_\mu \delta_\mu > 0.5\,, \tag{18}$$

where the values δ_μ are output signals ξ_ℓ emitted by nodes from which p_i has input connections. Suppose that the only input connections targeting nodes p_2, p_3 and p_4 are c_1, c_2 and c_3, respectively; that is, the "other inputs" indicated in Figure 10 are not present. Then, Γ_k is a singleton, $\Gamma_k = \{\gamma_k\}\,(k = 1, 6)$. Assigning weights $w_1 = w_2 = w_3 = 0.6$ and $w_4 = w_5 = w_6 = 0.2$ ensures that p_5 is active exactly when p_2, p_3, p_4 are active, which in turn occur exactly when p_1 is active. Therefore, the paths γ with $p_1 = \mathbf{o_S}(\gamma)$ and $p_5 = \mathbf{o_T}(\gamma)$ are all equivalent, or $\Gamma_7 \equiv \Gamma_8 \equiv \Gamma_9$ as desired. (Note that in the earlier discussion associated with Figure 9, we could have forced commutativity of its diamond-shaped diagram in a similar fashion given the freedom to choose an appropriate weight array.)

A weight array can easily be found to make the indicated diagram commute, but this is a wholly inadequate means of ensuring a representation of colimits. First, commutativity is only half the requirement: Initiality is the other half. More basic to our discussion, however, is that this approach, if followed in neural network design, would make colimits superflous. Notice that the neural diagram indicated in Figure 10 has all nodes (hence, signal paths $\gamma_1, \gamma_2, \gamma_3$) activated in any instance in which p_1 is activated. This precludes the re-use of p_1 in other colimit diagrams when those diagrams are supposed to represent concepts independent of T_5. But one of the major advantages of the colimit construction is that it allows diagram objects and morphisms to be re-used in other diagrams, representing distinct concepts, that is, concepts which have different model spaces. The model

spaces will very likely have nonempty intersections. However, if the concepts that share diagram components are to be non-redundant, they must have some non-shared models.

Having other connections to the nodes as indicated in Figure 10 allows the flexibility to overcome the redundancy and other shortcomings of the connections and weights discussed here. A balance of inhibitory as well as excitatory connections allows greater flexibility. Better still, replacing the feedforward architecture with one that employs feedback connections allows a colimit object node to provide control over its associated diagram. An example is shown in Figure 12. Utilizing the presence of the "other input" connections, the input connections to p_2, p_3, p_4 from p_1 can have weight values smaller in magnitude, hence, weaker, yet p_2, p_3, p_4 can be activated individually by appropriate inputs (we shall continue to view all connections as excitatory for simplicity in this example, although this is not a design suggestion). This allows their associated objects to be used in many separate but overlapping diagrams defining a wide variety of colimits that happen to share some concepts. The connections from p_2, p_3, p_4 to p_5 can, on the other, be strengthened, so that the activation of, say, p_2 and p_3, is enough to bring about the activation of p_5; this allows partial evidence for the concept represented by p_5 to highlight it through network activity as a hypothesis for representing the current input. Conversely, suppose that the feedback connections c_4^R, c_5^R, c_6^R, c_7^R in Figure 12 all have weights of sufficient magnitude so that the activation of p_5 necessitates the activation of p_1, p_2, p_3, p_4. Then, for example, any instance of path set Γ_7 (which is an instance of both Γ_1 and Γ_4, hence, requires that nodes p_1, p_2, p_5 be activated) will also be an instance of Γ_8 and Γ_9. Thus, the reciprocal connections ensure that the diagram commutes. Having inhibitory interconnections among colimit object nodes— another type of reciprocity— provides yet another degree of flexibility: it allows partially-activated nodes to compete for continued activation, with the winners finally suppressing the losers. This serves as a selection mechanism, allowing the colimits with the greatest support from the current network input pattern to be selected to represent the input.

6.2 Explicit versus Implicit Representations of Concept Morphisms

In principle, the ability to construct a functor M:**Concept** $\longrightarrow \mathbf{N}_{A,w}$ to establish with mathematical certainty the kinds of information content, or concepts, a neural architecture can represent, or to produce an architecture that can acquire and represent the kind and complexity of concepts desired. This makes it possible to *ground* a given concept representation by exposing its relationship to other concept representations involved in the processing of inputs by the neural architecture, ultimately relating it to the concepts represented at the input nodes. This raises the issue of the relative complexity of the morphisms of **Concept** and $\mathbf{N}_{A,w}$. A concept morphism $s: T \longrightarrow T'$ can be highly complex, since it is a symbol mapping that shows which items of each type — sort, operation, and constant— are related in the domain and codomain concepts, preserves the structure of the logical expressions of T in the substitutions of mapped symbols to obtain their s-images in T' and, finally, preserves the truth of the axioms of T in their images in T'. A single connection between nodes cannot be expected to represent this amount of information explicitly.

Depending upon the complexity of s, an explicit representation of its image $M(s): M(T) \longrightarrow M(T')$ under a functor M:**Concept** $\longrightarrow \mathbf{N}_{A,w}$ can require a multiplicity of connections and intermediate nodes forming several signal paths. In the triangle example, $M(s_1): M(T_1) \longrightarrow M(T_2)$ would need to represent the various sort, operation and constant mappings from T_1 to T_2. For example, s_1 maps point constants `p1` and `p2` of T_1 to `pa1` and `pa2` of T_2, respectively. There is another morphism that maps `p1` and `pa2` and `p2` to `pa1`, reversing the associations of points specified in the first morphism. An explicit representation of s_1, in the neural structure defining a morphism of $\mathbf{N}_{A,w}$ must make it clear which of the two is intended. This can be accomplished by having separate paths $\gamma_{10}, \gamma_{11}, \gamma_{12}, \ldots$ with $p_1 = \mathbf{o_S}(\gamma_{10}) = \mathbf{o_S}(\gamma_{11}) = \mathbf{o_S}(\gamma_{12}) = \ldots$ and $p_1 = \mathbf{o_T}(\gamma_{10}) = \mathbf{o_T}(\gamma_{11}) = \mathbf{o_T}(\gamma_{12}) = \ldots$, where each path represents one of the associations specified in s. For example, the association `p1` $\mapsto$ `pa1` can be represented using a node $p_1 0$ that is a coproduct object $M(T) + M(T')$ for the two neural objects $M(T)$ and $M(T')$ that represent two simple theories T and T' sharing the sort `Points`, with T containing the constant `p1` and T' containing the constant `pa1` (recall that in the neural architecture under discussion, we assume binary nodes so that nodes = objects). Figure 13 illustrates this. The fact that it is a coproduct object ensures that $p_1 0$ is active only if $M(T)$ and $M(T')$ are active (as with the original defining diagram for p_5, where $p_5 = M(T_5)$,

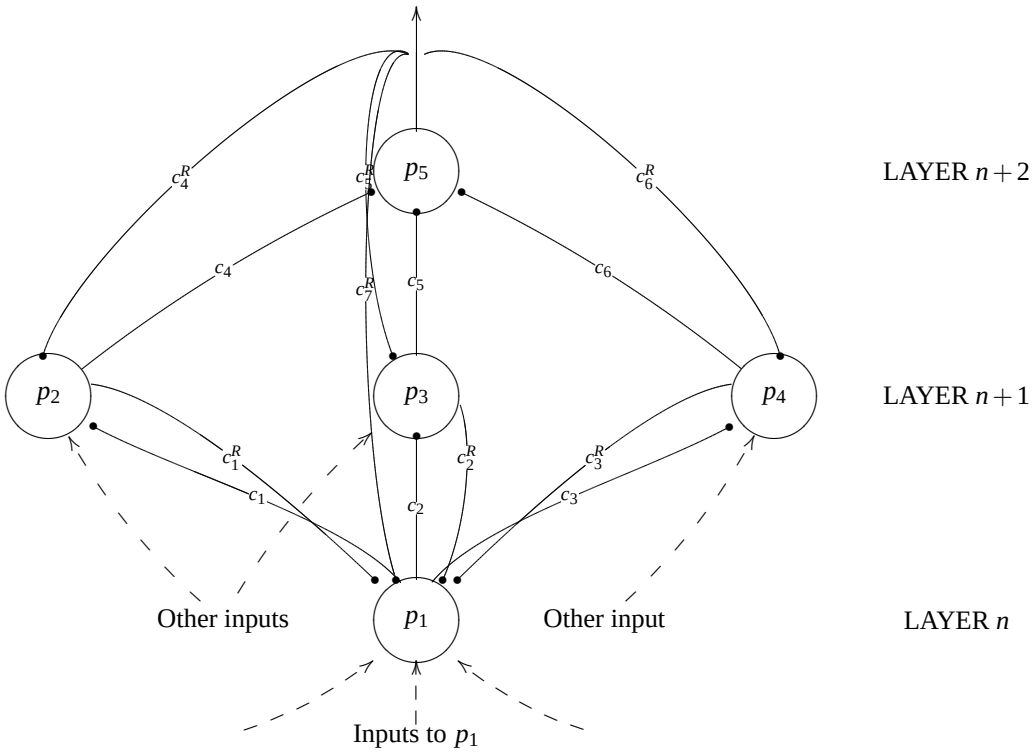


Figure 12: **Connection paths corresponding to a commutative diagram intended to serve as a neural colimit diagram in an architecture having feedback connections. The connections reciprocal to the connection paths of the colimit leg morphisms implement the model-space morphisms from a colimit concept to the concepts in its base diagram.**

we see that reciprocal connections are highly recommended as a design element). A path γ_{10} representing the association of `p1` with `pa1` is $\gamma_{10} = [p_1, c_{10}, p_{10}, c_{11}, p_2]$. Here, c_{10} and c_{11} are connections whose sole purpose is to form γ_{10}: They do not represent concept morphisms—nor can they, since their source and target nodes represent concepts which are mismatched with regard to the functorial property. Let Γ_{10} be the path set defining the morphism $M(s_1)$, $\Gamma_{10} = \{\gamma_{10}, \gamma_{11}, \gamma_{12}, \ldots\}$. It is the totality of the path set Γ_{10} that defines the morphism $M(s_1)$, not any one path within it. None of the paths individually, nor any of their single-connection components (such as $[p_1, c_{10}, p_{10}]$ or $[p_{10}, c_{11}, p_2]$ for γ_{10}), need be the functor image of a concept morphism.

Finally, there are two strategies for ensuring that the truth of the axioms of T_1 is maintained in their s_1-images in T_2. One strategy is to apply theorem-proving techniques (see [53]). The other is a model-based approach, checking to ensure that the model-space morphism $\mathtt{Mod}(s_1)\!:\!\mathtt{Mod}(T_2) \longrightarrow \mathtt{Mod}(T_1)$ is properly associated with $M(s_1)\!:\!M(T_1) \longrightarrow M(T_2)$. Given the earlier discussion of the indexing of models by the instances of the nodes representing their corresponding concepts, ensuring that the model-space morphism is properly represented can be achieved by ensuring that any instance of the object $M(T_2)$ that is also an instance of the morphism $M(s_1)\!:\!M(T_1) \longrightarrow M(T_2)$ is an instance of $M(T_1)$.

We have seen that an explicit representation of a concept morphism is a neural morphism with possibly many paths, constructed in a special way but apparently requiring knowledge of only those concept representations (such as $p_{10} = M(T) + M(T')$) directly involved in defining the associations along its paths. An alternative to an explicit representation is an implicit representation, with most of the morphism semantics hidden as with the single-connection path γ_1 associated with $M(s_1)$ in Figure 12. To ensure that this is a valid functorial representation, however, the mapping of sorts, operations and constants specified by s_1 must be discernible by

identifying the place that the purported $M(s_1)$ occupies within the network. Fortunately, as mentioned before, a theorem in category theory guarantees that in the concept category, colimit cocones can be automatically derived given their base diagrams. This makes it possible to "decode" concept object and morphism representations in $\mathbf{N}_{A,w}$ by tracing connection paths backward to the input nodes where these paths are involved in morphisms of the defining diagrams of colimits. The functorial property can be applied along with the knowledge of concept colimit derivations to determine the semantics of objects and morphisms in colimit derivations leading to the morphism in question. Thus, the connections to be traced are all those in paths leading to nodes $M(T_1)$ and $M(T_2)$ that are involved in the defining diagrams of colimits and/or limits. Although it allows a much simpler neural representation of the concept morphism s_1 as a neural morphism $M(s_1)$, it requires examining a potentially much greater part of the neural network to verify the representation. Ensuring that the truth of the axioms of T is maintained is the same as with the explicit representation.

Implementing concept colimits as neural colimits places several constraints on neural network design. These constraints require that nodes be interconnected through multiple pathways to form commutative diagrams and to provide some degree of accuracy in representing concept morphisms. The ability to enforce colimit representations also suggests that the pathways have reciprocal connections or some equivalent mechanism. There also seems to be a requirement for inhibitory connections, at least occurring between colimit object nodes, to prevent a "tower of babel" effect when multiple nodes share large portions of their base diagrams. Finally, representing concepts of a generally- applicable nature, such as those involving spatially invariant geometric entities, requires a means of representing the association of a spatially invariant entity with a class of spatially fixed, or location-dependent, entities that it characterizes. In the next section, we address a general notion of invariance, which we call abstraction. This involves a deeper understanding of the relationship between colimits and limits.

6.3 Grounding the Colimit Representations

The discussion of explicit versus implicit concept morphism representations in the previous sub-section raised the issue of grounding the representation of a concept morphism by ensuring that a functor actually exists that confers the semantics of the concept morphism upon a neural morphism purported to serve as its image. This can be done with either an explicit or an implicit neural representation. The explicit representation demands only local information about concept and morphism images directly connected with the neural morphism but requires that these be involved in multiple paths defining a rather complex neural morphism, an example being the path representing the maplet $\texttt{p1} \mapsto \texttt{pa1}$. The implicit representation requires analyzing the representations connecting the domain and codomain images of the concept morphism through many stages of colimit representation going all the way back to the input node concept representations, but can be implemented as a single connection in many cases.

More generally, the ability to claim that a functor exists that can be applied to a diagram such as that of Figures 7 and 8 raises the following issues:

1. There must be input nodes whose activations represent concepts from which T_1, T_2, T_3, T_4 can be derived via appropriate diagrammatic constructions in **Concept**.
2. It must be shown that the functor extends to these constructions, spanning the network from the input nodes to p_1, p_2, p_3, p_4.
3. It must be shown that the functor preserves colimits and other needed constructions from **Concept**; in particular, the neural cocone in Figure 11 must be initial: There must be a unique cocone morphism in $\mathbf{N}_{A,w}$ from it to any other cocone for the base diagram $M(\Delta)$.

Item 3 is a reflection of the fact that having a functor does not guarantee the preservation of colimits as such. However, it does preserve the commutativity of all the defining diagrams for cocones, and it also preserves cocone morphisms. As long as $\mathbf{N}_{A,w}$ does not offer additional cocones or cocone morphisms for the functor image $M(\Delta)$

of the base diagram Δ from **Concept**, colimits are indeed preserved. A functor that preserves colimits in general is explicitly referred to as a colimit-preserving functor.

To address the other two items, consider the concept images $M(T_1), M(T_2), M(T_3), M(T_4)$. They must be represented directly by input nodes or, more likely, constructed from other concepts associated with neural network input nodes. The most basic items required are representations of single points and lines. Linked properly to a visual sensor, for example, the input nodes can be made to respond to stimuli occurring in separate regions of the image space, such as a retinal visual field or a field of video image pixels (their receptive fields, or RFs). The RFs can overlap, but each appears to the network as a separate input, since it is associated with a separate input node. Input nodes having very small RFs, or "blobs", represent individual points. The detection of a point by a blob node is signified by a binary 1 (a sufficiently large accumulation of image brightness values occurring in its RF, yielding a positive stimulus exceeding its threshold). Individual lines can be represented as coproducts, where their base diagrams are discrete diagrams containing input nodes (remember that nodes are equivalent to objects in our binary representation). Connections from the input nodes to their line nodes are the paths associated with the coproduct leg morphisms; note that a point node can participate in many different diagrams, hence, can have connections projecting to many line nodes.

At this juncture, it might seem that the next step is to form colimits involving the point and line nodes to obtain $M(T_1), M(T_2), M(T_3)$ and $M(T_4)$. However, notice that the concepts T_1, T_2, T_3, T_4 all involve the `on` operation, and an explicit representation of this using point and line nodes is difficult to envision. Instead, an implicit representation of `on` as well as the axiom that uses it to relate points and lines in general will be assumed present in the manner in which the network is applied to represent geometry. For example, one can observe that a specific point is "on" a line when the nodes representing both are active simultaneously, generating positive outputs. Also, .all models of any of the concepts containing the axiom will be assumed to associate a unique line with any pair of points. In a pebbles-on-a-sidewalk model, for example, any pair d, d' of distinct pebbles will appear in the layout within one line l; any set of pebbles q containing both d and d' that is distinct from l cannot be a line. Another consideration in constructing $M(T_1), M(T_2), M(T_3)$ and $M(T_4)$ using point and line nodes as objects is that the represented points and lines are not spatially invariant—each point is associated with a fixed location in the image space, and each line is a coproduct of a specific collection of these fixed points. Thus, even if the diagram Δ could be represented using these nodes, it would express a triangle with a single, fixed location and orientation in the image space. Many such triangles would be required to represent arbitrarily-placed triangles. Finally, notice that the objects and constructions discussed so far leave the network with no means of expressing "pointness", "lineness" or "triangle-ness"—that is, all concepts representing fixed points share the property that they represent points, and similarly for those representing other fixed geometric entities, such as lines and triangles. In other words, the ability to represent spatially invariant geometric concepts is missing.

This calls forth a more general issue: In order for neural colimits to represent generally-applicable concepts, their base diagram objects must represent abstractions derived from the specific information represented at the sensor level. Spatial invariance is one such abstraction; there are many others.

6.4 Designing with Limits: Learning Invariants and Forming Abstractions

The desired representations of spatially-invariant quantities, and abstractions in general, already exist within the semantic model. It is the capability to derive limits for certain concept diagrams. As with colimits, the commutative diagrams for these constructions are preserved by functors. For limits, the only caveat remaining is that the cones must be terminal. As with initiality for colimit cocones, we address this with the general statement that terminality can be established if the requisite category of cones and cone morphisms can always be found where the limit of a represented concept diagram exists.

Future reports will explore the construction of limits and their interaction with colimits. For the present, suffice it to say that spatial invariance for a particular kind of object can be derived via a limit for a discrete diagram. Indeed, suppose the neural network has developed several representations for the same entitiy—a simple shape in a visual image space, say, described by a concept T. Suppose that the shape occurs several times in

input patterns to the network derived from images presented by a visual imaging sensor, and in each occurrence it appears without the image analysis pre-processing that would be necessary to remove spatial information. That is, each representation of the shape formed by adapting the neural network weights contains spatial information fixing the entity at a specific position in the visual field. Each representation is a concept T_i, say, associated with a neural category object (p_i, η). Each T_i can be thought of as T enriched with sorts, objects, constants and axioms which themselves constitute a concept S_i describing a spatial location relative to a fixed reference in the space, such as a coordinate center. For example, T_i might be a coproduct, $T_i = T + S_i$. Whether or not T_i is a coproduct, we express the representation via a functor M as $(p_i, \eta) = M(T_i)$.

Consider the discrete diagram consisting solely of a collection of spatially dependent concepts of the form T_i. A limit for the discrete diagram has an apical object, a concept, that contains only the description that is common to all the T_i. Since the location-specific descriptions S_i vary, this common description is essentially the shape concept T (some sorts and other information constituting a theory of space may also be present, but this is of no concern). A neural limit object $M(T)$ for the discrete diagram of spatially-dependent neural representations $M(T_i)$ is a spatially-invariant shape representation.

In the case of the triangle example, the base diagram components are to be spatially invariant. This can be accomplished by a network that can represent limits as just described. The use of limits and colimits in combination to form complex representations from spatially-invariant and other abstract representations, as well as the formation of abstractions by the network, is explored next.

6.5 Feedforward Networks versus Networks with Feedback

Recall that the model-space morphisms $\mathtt{Mod}(s):\mathtt{Mod}(T') \longrightarrow \mathtt{Mod}(T)$ are directed opposite their corresponding concept morphisms $s:T \longrightarrow T'$ and neural morphisms $M(s):M(T) \longrightarrow M(T')$. This suggests that a neural network implementing concepts must have an instance for each concept T (represented by $M(T)$) in every case that it has an instance for a more complex concept T' (represented by $M(T')$) which incorporates it when the latter instance is accompanied by an instance of a morphism $s:T \longrightarrow T'$ (represented by $M(s):M(T) \longrightarrow M(T')$). This must be true independently of whether there is a neural morphism $m:M(T') \longrightarrow M(T)$ available in the opposite direction to $M(s)$ to represent the model-space morphism $\mathtt{Mod}(s):\mathtt{Mod}(T') \longrightarrow \mathtt{Mod}(T)$. In particular, this applies to limits and colimits: Every instance of a colimit object T must be accompanied by an instance of its base diagram, as implied by the model-space morphisms $\mathtt{Mod}(\ell_i):\mathtt{Mod}(T) \longrightarrow \mathtt{Mod}(T_i)$ associated with its leg morphisms $\ell_i:T_i \longrightarrow T$. The concept morphisms are represented in the neural category by the functorial images $M(\ell_i):M(T_i) \longrightarrow M(T)$, which in turn are represented by bundles of signal paths directed from each $M(T_i)$ to $M(T)$ (recall that if represented implicitly, the leg morphisms can have singleton bundles whose sole elements are single-connection paths). To begin making the point of this discussion, the simplest means of assuring that the neural network behaves correctly in accordance with the model-space morphisms $\mathtt{Mod}(\ell_i):\mathtt{Mod}(T) \longrightarrow \mathtt{Mod}(T_i)$ is for the neural network to have reciprocal paths directed from $M(T)$ to each $M(T_i)$. Otherwise, the desired behavior is difficult to ensure: The many diagrams possible in the concept category overlap in many places, and this makes it highly improbable that a feedforward architecture representing concept morphisms alone and trained on an arbitrary set of input patterns will perform in the synchronized manner implied by the model-space morphisms adjoined to colimit leg morphisms. Having a feedforward architecture represent the model-space morphisms instead of the concept morphisms is no solution to this dilemma, for then complex concept representations could not be learned as combinations of simple, input feature representations.

On the other hand, an instance of some of the diagram objects and morphisms may provide excitatory input to the colimit object but without activating it: These objects and morphisms may also be part of another diagram, and activate instead the colimit of that diagram. An instance of the leg morphisms $M(\ell_i):M(T_i) \longrightarrow M(T)$ acts to "prime" the colimit object, giving it a competitive advantage over colimit objects with lesser or no input from active diagram representations in the the network. This is where network synchronization is essential: The many, overlapping network representations of the many, overlapping diagrams that are available to represent a current input pattern, and thereby retireve its representation from the connectionist memory of the network, must be

sorted through in some fashion and one or more choices made among them. Intuitively, this can require rather sophisticated neural circuitry.

By duality, similar comments hold for a limit object for a diagram, but with the directions of all morphisms reversed. An instance of any of its defining diagram objects in $\mathbf{N}_{A,w}$ will activate a limit object for the diagram as long as there is activity in the signal paths defining the corresponding leg morphism. That is, an output ξ_i by an appropriate node p_i lying within an appropriate interval η_i, representing a diagram object (p_i, η_i), will always activate a limit object (p, η) for the diagram, with an output ξ lying within the interval η being emitted by the node p, given that the nodes in the network's representation of the neural morphism representing the corresponding model-space morphism have activity in their corresponding intervals. That this must be the case is a consequence of the model-space morphism associated with the leg morphism. The model-space morphisms $\mathtt{Mod}(\ell_i)\colon \mathtt{Mod}(T_i) \longrightarrow \mathtt{Mod}(T)$, directed opposite their corresponding concept morphisms $s\colon T \longrightarrow T_i$ and neural morphisms $M(s)\colon M(T) \longrightarrow M(T_i)$, are represented in the neural network architecture by connections reciprocal to the signal path bundles representing the concept morphism images in the neural category. On the other hand, the limit object provides input to its defining diagram objects but need not activate them.

To summarize, the rules for activation of limits versus activity in their defining diagrams have just the opposite sense of the rules for activation of colimits versus activity in *their* defining diagrams. This is appropriate, since the corresponding limit leg morphisms are directed from limit to diagram object, just the opposite of the sense of the colimit leg morphisms.

An example of this behavior of neural network representations of limits can be seen in the relationship between spatially invariant representations and the spatially-dependent items they represent. More generally, the formation in the neural network of a representation of a limit for a diagram is characteristic of the derivation of an abstraction when a diagram of special cases of the abstract concept is represented by neural network activity. In the concept category, that which is common to the diagram concepts as they are related by diagram morphisms is the content of the limit object, provided a limit for the diagram exists. The fact that not all diagrams have limits is a reflection of the fact that an abstraction may not be derivable in many situations. This confers a value on diagrams that do lend themselves to limit formation. Appropriately, when it exists, the network's limit object representation will be activated when any of its defining diagram object representations is activated. This is a reflection of the fact that an instance of an abstraction occurs when any of its special cases occurs.

The full range of behaviors with regard to limits and colimits just described can be realized in the most straightforward manner by a neural network that contains reciprocal connections between nodes associated with neural objects representing concepts that are connected via morphisms. Suppose, by contrast, that a feedforward neural network is designed to learn invariant representations in the manner of the Neocognitron (originally proposed in [14]). This architecture expresses invariant representations, of visual shapes, say, by having a layer of simple (S) cells learn spatially-dependent shapes by virtue of having input connections from visual feature-representing cells localized to image regions (for example, edge detectors). The S cells for all spatially-dependent representations of a single shape supply input to a single cell in a layer of complex (C) cells. The C cells represent all the spatially invariant shapes. In our categorical model, the input connections to a C cell are directed opposite the direction of concept morphisms, where the S cells represent the spatially- dependent shape descriptions, or concepts, and the C cells represent the corresponding spatially-invariant shape descriptions. The S-to-C cell connections are, in fact, associated with the model-space morphisms. Notice that, as described in [14], any one S cell can excite a C cell, as is the case with the nodes associated with our discrete diagram objects and the corresponding diagram limit (or in the discrete case, product) objects.

Similarly to but opposite the situation with colimits, on the other hand, there are no connections in the reciprocal direction—from C to S cells. But these connections are necessary to represent the limit leg morphisms $\ell_i\colon T \longrightarrow T_i$, where T_i is a diagram concept object and T is a limit object for the diagram. These morphisms have the limit object as domain and the diagram objects as codomains. Since there are no C to S cell connections, there can be no stimulus transmission in the network associated with the activation of a C cell. But this transmission would be useful in cases where an explicit manifestation of the spatially invariant representation at all locations is desirable. For example, it might be desired to call forth a shape representation and have the neural network

"prime", or stimulate, all past localized representations for the shape. This would give the shape representation an advantage over other, competing shapes in all those regions. This would serve as a " shape filter" for input patterns, and could be seen as a manifestation of the semnantics of the invariant representation in the neural network's behavior. In any case, a feedforward network has connections in only a single direction, in particular, solely for the directions toward (or efferent to) a C cell from its spatially-dependent counterparts. Therefore, a feedforward network does not have the flexibility in function reciprocal connections would provide the C cells as limit objects. More generally, a feedforward neural network does not properly represent limits.

By duality, a similar analysis with opposite activation properties can be made for S cells at higher levels which might receive inputs from a variety of spatial-invariance C cells, to form complex shape representations using the simpler spatially-invariant C-cell-based representations as features. This use of S and C cells, by placing them in alternating layers in a feedforward network, is discussed in [42]. The higher-level S cells correspond to colimit objects in the categorical model, but the feedforward neural network does not properly represent them as colimit objects.

In summary, the Neocognitron is an architecture that has important and highly useful properties for representing invariant features and complex, multi-feature objects. The paper of Riesenhuber and Poggio [42] provides an informative and useful analysis of these properties. The claim made in this section is simply that there is more to be gained by heeding the consequences of the categorical semantics applied to neural networks.

7 Naturality: Knowledge Coherence Across a Multi-Region Network

7.1 Stages of Learning in a Multi-Region Network

Consider, now, 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 autonomous vehicle controlled by the network), and regions for cognitive functions such as situation assessment and planning. The semantic model 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. We can design an architecture to learn concepts describing those aspects of sensed items representable by any particular collection of sensors, or concepts which involve many sensors for their full expression but must be implemented with a limited sensor array for economy reasons.

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 models the concept hierarchy as the category **Concept**, represented in a separate region of the network having a specialized function. Since functors can be many-to-one on both objects and morphisms, those parts of concepts that do not relate to the function of the subregion corresponding to a functor are "compressed out". This will be explained in more detail. First, let us ask the overriding question for this multi-regional network semantic model: How are the functions of the separate regions to be unified? The functions are described by the concepts of the hierarchy, but this is represented differently in each region. In the current example, how can the two sensors be exploited to acquire a unified concept representation in the neural network, when each sensor region expresses only that aspect of the concepts specific to that sensor representation? *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 14 . This system of functors and natural transformations works together as follows.

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: The latter are instead specific to each network region. via reciprocal connections as usual. A third network region contains the representation of the objects and morphisms of the image of M_3;. However, the M_3 region has no sensor. Instead, it serves as an association region. The se-

mantic model makes the association function explicit: Each triple of objects $M_i(T_\mu)\,(i = 1,2,3)$ is connected by the two individual morphisms $\alpha^1_{T_\mu}: M_i(T_\mu) \longrightarrow M_3(T_\mu)\,(i = 1,2)$. This connects the three functorial images of the concept T_μ. The defining property of a natural transformation specifies that the network connections that support the natural transformation morphisms are the appropriate ones to unify the two sensors in the dual-sensor representation of functor M_3. For, consider a **Concept** morphism $s_\alpha: T_\mu \longrightarrow T_\nu$ specifying a subconcept relationship. The functorial images $M_i(s_\alpha): M_i(T_\mu) \longrightarrow M_i(T_\nu)\,(i = 1,2,3)$ 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.

7.2 Diagrams in a Functor Category

The categorical model, with functors from a category of concepts to a category of neural network components and natural transformations between these functors, provides a mathematical model for neural structures consistent with concept-subconcept relationships. Colimits of diagrams show how concepts can be combined, and how a concept can be re-used many times in forming more complex concepts. Functors map commutative diagrams to commutative diagrams, capturing this aspect of the colimit structure. Natural transformations express the fusion of single-mode sensor representations of concepts in the same neural architecture, connecting the different implementations of the concept hierarchy at all levels and in a consistent fashion. This mathematical model appears to be compatible with a model of the primate brain proposed by Damasio[11]. In this section, we explore the multi-regional ramifications of these notions.

Functors can be many-to-one on either or both objects and morphisms; for two objects a and b, for example, it can be that $\mathbf{F}(a) = \mathbf{F}(b)$. Because of its significance in architecture design([21],[22]), we refer to this "merging of objects and morphisms" as *compression*.

A natural transformation $\alpha : \mathbf{F} \longrightarrow \mathbf{G}$ between functors $\mathbf{F}, \mathbf{G} : \mathcal{C} \longrightarrow \mathcal{D}$ consists of $\mathcal{D}$-morphisms α_a, one for each object a of $\mathcal{C}$, such that for each morphism $f: a \longrightarrow b$ of $\mathcal{C}$, $\mathbf{G}(f) \circ \alpha_a = \alpha_b \circ \mathbf{F}(f)$. The square-shaped commutative diagrams for two natural transformations $\alpha^1, \alpha^2 : \mathbf{F}, \mathbf{G} \longrightarrow (\mathbf{F} + \mathbf{G})$ evaluated at two objects a and b and a morphism $f: a \longrightarrow b$ are shown in Figure 15 . This figure also illustrates the notion of a coproduct for two functors $\mathbf{F}$ and $\mathbf{G}$. Given categories $\mathcal{C}$ and $\mathcal{D}$, there is a category $\mathcal{D}^\mathcal{C}$ whose objects are the functors $\mathbf{F} : \mathcal{C} \longrightarrow \mathcal{D}$ and whose morphisms are the natural transformations, such as α^1, α^2. A coproduct functor $\mathbf{F} + \mathbf{G}$ for two functors $\mathbf{F}, \mathbf{G}$ in $\mathcal{D}^\mathcal{C}$ (provided the required injection morphisms exist) is one that compresses a pair of objects or of morphisms from $\mathcal{C}$ if and only if the pair is compressed by both $\mathbf{F}$ and $\mathbf{G}$. The coproduct has the effect of "pasting together" the corresponding commutative squares of the natural transformation injections as shown in Figure 15 .

Functors in the category $\mathbf{N}^{\mathbf{Concept}}_{A,w}$ preserve the colimit construction, in part because functors preserve diagram commutativity. A given functor M_A: **Concept** $\longrightarrow \mathbf{N}_{A,w}$, therefore, effectively transports the structure of all possible knowledge— represented by the concept category—into the categorical architecture representation. The colimit construction also indicates how concepts are formed through adaptation. Of necessity, M_A entails a large amount of compression; in fact, it represents a single state of learning in a finite network, with all the other knowledge contained in the infinitely-large concept category hidden by compression. Many such functors are required to express the concept hierarchy representation in the several network regions at the many stages of learning.

To see how we apply this in analyzing existing neural network designs, consider some of the ART mapping networks (for example, as analysed in a rule-base analysis in [20]). These couple separate, unsupervised ART networks (sometimes with an intermediate "mapping field" subnetwork) to provide supervised ART systems. We

have shown[21] that each ART network has solely coproduct formation as a means of learning complex concepts in terms of the simple concepts represented by its input nodes. This stems from the lack of connections within the input (F_0) and matching (F_1) layers. The more general colimits required to capture knowledge representation are missing, and limits have no explicit representation at all in these networks. Also, there are no nodes available for compression of yet more complex (or less complex) concepts, or input concepts not represented by the present input layer. The effect is to limit the concept hierarchy representation to a functor image fragment containing at most coproducts over a single layer. Further, the interconnections between ART networks required for commutative squares are not present, so any natural transformations would be only partially represented[21].

On the positive side, ART networks have important advantages for a study of knowledge representation. The explicit learning procedure and the presence of feedback (top-down) connections facilitates the identification of intervals η of the F_2 nodes as coproduct objects: The bottom-up connections to a previously-committed F_2 node that have nonzero weights are the connection paths of the coproduct injection morphisms. Using an ART 1 network as an example[6], the coproduct objects are the previously-mentioned intervals η_2 of the committed F_2 nodes. Suppose that the template of a winner-take-all node in the F_2 competition to represent an input pattern is a subset template of "sufficient size" (see [6]; "subset" means that each nonzero template connection weight corresponds to a nonzero input pattern value, and "sufficient size" means that there are "enough" template nonzeros). The top-down, unit-weight template connections then ensure the activation in their η_2 intervals of exactly the F_1 nodes in the discrete diagram upon which the currently-expressed coproduct is based, thus ensuring the continued activation of the coproduct morphism connections. This largely describes the semantics of resonance in a typical ART network.

The interconnects between F_2 layers of the separate ART networks is suggestive of natural transformation components corresponding to the concepts represented by the F_2 η_2 intervals. What is missing are the components corresponding to the concepts at the F_0 (or, equivalently, the F_1) level. We decided to exploit the properties of network templates and interconnections by using ART networks as a point of departure in developing new architectures more compatible with knowledge representation and coherence.

8 A Categorically-Motivated Multi-Regional Design

In this section, we apply the design principle of knowledge coherence to derive a neural network design that improves upon multi-ART networks. We do not address improvements to allow the derivation of colimits more general than coproducts, and we do not address the derivation of limits. The design principles corresponding to these fundamentally important considerations will be addressed in future analyses. Further discussion of them does, however, appear in later sections of this document.

We have given a very brief example of applying the categorical semantic theory using ART mapping networks. The example emphasized binary nodes for simplicity, using the binary intervals η_2. Our other example, the proposed categorical neural architecture that is the subject of this paper, will be described next. Although this initial architecture is designed to operate in a binary fashion, we use the more general notation η for arbitrary intervals for simplicity and also to emphasize that the semantic model applies to other than binary-node networks.

Figure 14 shows the overall scheme for an architecture that represents and coherently fuses the information from two sensors according to our theory. There are three regions, two receiving input from sensors and the third, an association region, receiving input from the other two regions. In terms of the semantic model, the architecture at a given stage of learning is represented by the category $\mathbf{N}_{A,w}$ and the sensor regions are represented by the images of the functors M_1 and M_2. The association region is represented by the image of M_3. Two natural transformations $\alpha^i : M_i \longrightarrow M_3$ $(i \in \{1,2\})$ are coproduct morphisms for the discrete diagram containing M_1, M_2 in $\mathbf{N}_{A,w}^{\mathbf{Concept}}$, and M_3 is the corresponding coproduct object, $M_3 = M_1 + M_2$. Figure 16 shows an initial architecture designed in accordance with this scheme and using properties of ART networks where these can be exploited. We call the sensor regions *primaries* and denote these by $\mathbf{P}_1$ and $\mathbf{P}_2$, corresponding to the functors M_1 and M_2. We call the region corresponding to M_3 an *associator*, which we will denote by $\mathbf{A}$. Adaptive

connections are indicated by arrows ending in bullets and non-adaptive connections have standard arrowheads. Connection polarity is indicated by + (excitatory), - (inhibitory), or R (a bundle of non-phasic reset connections).

An immediate problem occurs in implementing the scheme of Figure 14 as a multi-sensor architecture. Let us suppose that the two sensors are logically independent. For example, let $\mathbf{P}_1$ be coupled to a visual sensor, with its F_0 layer representing concepts describing visual primitives of some kind, and let the F_0 nodes at $\mathbf{P}_2$ represent tactile information provided by an array of pressure sensors. Then, since the two kinds of sensor primitives convey entirely different kinds of infornmation, each subnetwork must omit the input concepts associated with the other subnetwork's sensor. However, our semantic model requires that each functor map the entire knowledge space of **Concept** into its subnetwork region, including all of the concepts that describe both kinds of sensor primitives. To satisfy these competing requirements, there must be $\mathbf{N}_{A,w}$ objects and morphisms available to represent the functorial compression of sensor primitives for each subnetwork within the other subnetwork. We choose to represent these as a sort of auxiliary F_1 node, because it is at F_1 that the discrete diagrams form via bottom-up/top-down matching (ART-based, although the gain control subsystem is not shown). We include one of these compression nodes in each subnetwork to serve as the target for the functor images of the missing sensor primitives. They are labelled C_1 (for $\mathbf{P}_1$) and C_2 (for $\mathbf{P}_2$) in Figure 16 . The coproduct functor M_3, on the other hand, fully represents the non-compressed knowledge represented by M_1 and M_2. Therefore, region $\mathbf{A}$ has two F_1 layers whose nodes receive inputs from their corresponding F_1 nodes in $\mathbf{P}_1$ and $\mathbf{P}_2$, respectively. We use the term "proxies" for the two F_1 layers in $\mathbf{A}$; however, their bottom-up/top-down connections with the $\mathbf{A}$ F_2 layer operate independently of $\mathbf{P}_1$ and $\mathbf{P}_2$.

We connect each compression node to the proxy of the opposite subnetwork's F_1 layer as shown. There are pairs of connection paths emanating from the F_1 nodes of $\mathbf{P}_1$ and $\mathbf{P}_2$ to the appropriate $\mathbf{A}$ F_2 nodes. These form the commutative squares of the two natural transformations $\alpha^1{:}M_1 \longrightarrow M_3$ and $\alpha^2{:}M_2 \longrightarrow M_3$. For example, let S be an object in **Concept** that descibes a sensor primitive via the functor M_1, so that for some $\mathbf{P}_1$ F_1 node $F^1_{1,k}$ and an appropriate interval η, $(F^1_{1,k}, \eta) = M_1(S)$. To simplify notation, we shall simply use the functorial representation, in the present case $M_1(S)$. Proceeding, let S be represented in the $\mathbf{A}$ subnetwork's $\mathbf{P}_1F_1$-proxy layer via the functor M_3 as $M_3(S)$. This yields two F_1 representations of S, connected across subnetworks by a $\mathbf{P}_1$ F_1 to $\mathbf{A}$ F_1 connection that represents the natural transformation component $\alpha^1(S){:}M_1(S) \longrightarrow M_3(S)$. Let T be a coproduct concept with a coproduct injection morphism $m{:}S \longrightarrow T$, and let T be represented in the $\mathbf{P}_1$ and $\mathbf{A}$ F_2 layers by $M_1(T)$ and $M_3(T)$, respectively. Then, there are two connection paths from $M_1(S)$ to $M_3(T)$ forming the sides of a commutative square, as follows. One path consists of two connections: a $\mathbf{P}_1$ F_1 to $\mathbf{P}_1$ F_2 connection, the coproduct morphism image $M_1(m){:}M_1(S) \longrightarrow M_1(T)$, followed by the $\mathbf{P}_1$ F_2 to $\mathbf{A}$ F_2 connection for the α^1 component $\alpha^1(T){:}M_1(T) \longrightarrow M_3(T)$, yielding the composition morphism $\alpha^1(T) \circ M_1(m){:}M_1(S) \longrightarrow M_3(T)$. The second path consists of two connections: a $\mathbf{P}_1$ F_1 to $\mathbf{A}$ F_1 connection for the α^1 component $\alpha^1(S){:}M_1(S) \longrightarrow M_3(S)$, followed by an $\mathbf{A}$ F_1 to $\mathbf{A}$ F_2 connection, the coproduct morphism image $M_3(m){:}M_3(S) \longrightarrow M_3(T)$, yielding the composition morphism $M_3(m) \circ \alpha^1(S){:}M_1(S) \longrightarrow M_3(T)$. The composition morphisms are the same morphism, $\alpha^1(T) \circ M_1(m) = M_3(m) \circ \alpha^1(S)$, hence, in the architecture, the two connection paths must represent the same $\mathbf{N}_{A,w}$ morphism. In other words, the square whose sides are the four factors in the two compositions is commutative. Thus, the relationship between the primary sensor primitive concept representation $M_1(S)$ and the associator colimit concept representation $M_3(T)$, where concept T includes concept S as one of its parts, is independent of the path that expresses the relationship across the primary $\mathbf{P}_1$ and the associator $\mathbf{A}$. This is the basis for knowledge coherence and is one of the theoretical considerations that help determine the operational rules for the architecture.

Joined to the preceding commutative square is one for the natural transformation $\alpha^2{:}M_2 \longrightarrow M_3$. In the latter, we obtain two connection paths from $M_2(S)$ to $M_3(T)$; however, there is no $\mathbf{P}_2$ F_1 node representing S, but instead the compression node C_2. Defining the appropriate functor images and natural transformation components as before, but this time using C_2 instead of a $\mathbf{P}_2$ F_1 node (but using again the same $\mathbf{A}$ F_1 node from the proxy layer for $\mathbf{P}_1$ F_1), we obtain the commutative square associated with the equality $\alpha^2(T) \circ M_2(m) = M_3(m) \circ \alpha^2(S)$. The fact that the two commutative squares are "pasted together" along $M_3(m){:}M_3(S) \longrightarrow M_3(T)$ implies that in the operational architecture, all four connection paths must become activated for the same inputs. The details of operation must be defined to be consistent with this.

8.1 Discussion of Operation for the Multi-Region Design

There is not space here to describe the operational flow of the new architecture in full, but we can point out a few guiding principles. First, the fact that the architecture is to implement coproducts, functors and natural transformations and, in particular, coproducts in $\mathbf{N}_{A,w}^{\mathbf{Concept}}$, serves as a basis for the algorithm for the activation and connection-weight modifications. Thus, the appropriate commutative square diagrams must be activated connecting the $\mathbf{P}_1$ and $\mathbf{P}_2$ F_1 and compression nodes to the currently-active $\mathbf{A}$ F_2 coproduct node. This operational consideration is a realization of a theoretical implication of the semantic model: The fusion of information from multiple sensors is a consequence of knowledge coherence, fully expressed as the calculation of colimits at two levels. First, colimits in **Concept** are mapped to their neural representations by functors, which preserve the commutativity of diagrams. Second, natural transformations representing interconnects to an association region in a neural network define the latter as a coproduct of functors; in the presented architecture, this results in the appropriate "pasting together" of the commutative squares some of whose sides are the colimit (in this case, coproduct) morphism images for the functors.

More operational detail is provided by the desire to stay as close as possible to ART network design. The F_2 choice and template modification operations within each subnetwork proceed according to the ART gain control and connectionist adaptation mechanisms. For example, **Neural** morphisms such as $M_1(m)\colon M_1(S) \longrightarrow M_1(T)$ and $M_3(m)\colon M_3(S) \longrightarrow M_3(T)$ in the illustration of commutative squares for $\alpha^1\colon M_1 \longrightarrow M_3$ are implemented as bottom-up connections, but the latter are assisted in this role by their corresponding top-down template connections, which ensure their activation during the appropriate periods.

The principle of connections not directly involved in a morphism supporting those that are in their role is extended throughout the new architecture. For example, the morphism $\alpha^1(T)\colon M_1(T) \longrightarrow M_3(T)$ is implemented as a $\mathbf{P}_1$-F_2-to-$\mathbf{A}$-F_2 connection (an on-center (OC) connection in Figure 16), but the latter receives support in that role from its attendant F_2-to-F_2 off-surround (OS) connections to the rest of the $\mathbf{A}$ F_2 layer and also from a reciprocal connection with which it is paired. Thus, coproduct nodes in $\mathbf{P}_i$ and $\mathbf{A}$ $(i = 1,2)$ that represent the same coproduct concept form mutually-supportive pairs. Sustained activation of a chosen $\mathbf{P}_1$ F_2 node, for example, depends upon this, for its support from its own template weakens during the time the input it is representing is presented to the network.

The $\mathbf{P}_2$-to-$\mathbf{A}$ commutative square that shares a side with the previously-described $\mathbf{P}_1$-F_1-to-$\mathbf{A}$-F_2 square has the $\mathbf{P}_2$ compression node C_2 in the role of the third representation of concept S, $M_2(S)$, and its connections to/from the $\mathbf{P}_1$ F_1 proxy in $\mathbf{A}$ are inhibitory. To examine the effect this has in a typical operational scenario, let us assume that for the current episode of presentation of input patterns to the $\mathbf{P}_1$ and $\mathbf{P}_2$ F_0 layers, $\mathbf{P}_1$ was the first to reach its candidate for its F_2 choice, a node representing the functor image $M_1(T)$ of a concept T. Suppose that, next, a node representing the image $M_2(T')$ of a concept T' becomes activated. This causes the activation of C_2 (see the appropriate top-down connection in Figure 16), which then acts to suppress the $\mathbf{P}_1$ F_1 proxy layer in $\mathbf{A}$; but $M_1(T)$ has already provided excitation to C_1, which acts in a similar manner to suppress the $\mathbf{P}_2$ F_1 proxy. Under an ART-like "2/3 Rule" operating in $\mathbf{A}$, the proxies can only sustain their activities if the nodes representing $M_3(T')$ and $M_3(T)$ can provide the top-down excitatory stimulation through the proxy template connections. However, if $M_3(T') \neq M_3(T)$, then two distinct $\mathbf{A}$ F_2 nodes will be competing for activation as winner-take-all nodes. This results in their mutual suppression through the OS inhibitory connections from $\mathbf{P}_1$ and $\mathbf{P}_2$, resulting in loss of top-down support to their proxies. The final result is a loss of activity in the node representing $M_1(T)$, for its support from its template has weakened with the passage of time, as mentioned before (in the current architectural design, $M_2(T)$, having reached its choice later, has support from its "fresh" nonzero template bottom-up connections, hence, remains near full activation). The weakened state of the $M_1(T)$ node allows the reset connection from C_1 to the $\mathbf{P}_1$ F_2 layer to have its full effect (C_1, like $\mathbf{P}_2$, lags behind $\mathbf{P}_1$ F_1 and F_2 in its cycle of activation). This results in the stimulation of the F_2 competitors for $M_1(T)$, which then compete to choose a new winner. Thus, the functor coproduct is enforced by rejecting the choice of the primary subnetwork that made the original prediction.

If, on the other hand, $M_3(T') = M_3(T)$, then the primary subnetwork predictions are consistent and the entire network is said to be in resonance. The proper associations between concept representatives have been established

and enforced at both the F_1 and F_2 levels across all three subnetworks. Thus, resonance as here described corresponds to coherence at both concept levels. That is, a concept representation at F_1 in both primaries is related to a single associator F_2 representation through a morphism representation, a bundle of connections representing a commutative square. The two commutative squares express the notion that the primary-to-associator subconcept-to-concept relationships are independent of path, hence, are expressed without ambiguity.

This architecture is currently in preliminary testing for a series of experiments that will follow. Experience gained with it so far has led to some design refinements. The refinements are not completely determined by expediency: When a difficulty arises, it leads us to consider the semantic model, and to ask if we are being fully consistent with it and to what extent does it determine the architectural details.

9 Conclusion

We have presented a mathematical semantic model for neural networks. The semantic model is based upon category theory, the mathematical theory of structure. The categorical constructs used to model the representation of concepts and their relationships suggests architectural structures and their properties. These apply to learning and to the combination of information from multiple sensors in a multi-region architecture. Overall, the analysis with the semantic model performed to date suggests the following neural network design principles.

1. Functorial mappings guide network design by posing constraints upon the representation of concepts and their relationships, where the relationships are the morphisms in a category of theories.
2. The learning algorithm of the network must be capable of expanding an existing functor image representation into a larger one through network activation.
3. The expansion of the representation will consist in part of the derivation of colimits to represent complex entities and situations through the re-use of already-represented concepts and their morphisms. It consists also of the derivation of limits to represent the abstractions that are derivable from diagrams representing their special cases. Reciprocal connections are strongly suggested as architectural elements to enforce the properties of limits and colimits.
4. Natural transformations represent knowledge coherence, the association of information from different regions of a multi-region architecture in such a manner that the separate representations of the concept hierarchy act as one.
5. Reciprocal connections are again suggested, to enforce the commutative squares of the natural transformations.
6. Concept compression is suggested as a means of representing missing information—concepts and morphisms that are represented in one region by a functor but not in another region with its separate functor. The commutative diagrams of the natural transformations ensure that the missing concepts, compressed by a many-to-one functor onto a node representing many concepts at once, are properly connected to their representations in other regions.

We have given examples to illustrate the semantic model and its applications in neural network analysis and design. We have applied it to achieve an understanding of the semantics of ART networks. We have followed this with an application to the design of a multi-region, multi-sensor neural network capable of coherent knowledge representation based upon the new design principles. In the new design, nodes and connections are explicitly organized to implement coproducts, functors and natural transformations. Experimentation with the new architecture is under way.

There are still details implied by the semantic model that are missing in the new architecture introduced here. First, as in ART, the colimits represented in each subnetwork are only two-layer coproducts. Limits are

not represented at all. Second, the compression nodes and morphisms for the infinite variety of concepts not representable in any of the three subnetworks are missing, with the sole exception of the compression nodes for the alternate sensor in each sensor subnetwork. Nevertheless, the architecture described here is a beginning at applying design principles obtained from the semantic model. More generally, the present document introduces the model and delineates many of its basic features as a guide to neural network semantic analysis. Finally, 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 model would seem to have a natural role as a fundamental model for exploring cognition in neural networks.

10 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) and Joseph Goguen (University of California at San Diego) for their helpful comments and suggestions during the development of the semantic model.

Appendix

A An Introduction to Some Formal Logic Terminology

Quantification of variable symbols is something not normally considered unless one is specifically concerned with a study of logic, the establishment of truth by argument. An example of the need for quantification occurs when analyzing statement forms "X is a dog". Here, the meaning of the term "dog" is clear: It is a symbolic string representing a class of animals, which can be described in a theory with as much detail as one wishes. In the description, of course, the theory will resort to undefined quantities: One can include as many definitions and axioms as one likes in an attempt to describe what exactly is meant by the term "dog", but eventually one finds that something in the description must simply be understood without further definition. A theory of geometry, for example, resorts to introducing "points" without defining them, to avoid circularity in definitions. One cannot define everything, and at some place in any discussion, formal or informal, there must be an understanding of a common meaning of terminology with no further explanation required. The symbol "X" in the statement form "X is a dog", on the other hand, has no fixed meaning other than that it represents an arbitrary member of the class "dog": That is, it is a variable. Clearly, the use of this variable in the statement form leaves the validity of the statement form ambiguous. No justification exists for stating "X is a dog" without further information about the context in which "X" is being used. What if X is really a cat or an elephant? It is clear, however, what meaning is intended in the statement "There exists a dog". In formal logic, one introduces a variable such as X and restates this as "There exists an X such that X is a dog", so that the form "there exists" can be re-use as a general symbolic form for the statement of existence, with the context of its use supplied by the variable ("X" in this case) and the statement form to which it is adjoined.

In symbolic logic, or formal logic as it is now called, predicates are used together with logical operators and quantifiers to form propositions. In the text, propositions are referred to simply as "statements". A proposition has a form such as "There exists X such that X is a dog". A statement form of this kind is called *closed* because its variables are all quantified. Axioms, definitions and theorems, the statements making up a theory, must all be closed.

Predicates are statement forms intended as descriptions of the properties of things. The things are like the subjects in sentences of natural language, and the predicates are like verb phrases or statements of membership: In "X is a dog", X represents an arbitrary subject and "is a dog" is, of course, a statement form for membership in a class of animal. In formal logic, this would be restated symbolically in a form such as $\mathtt{dog}(x)$. Existentially quantified, it would be written in a form such as $\exists(x)(\mathtt{dog}(x))$.

Normally, one needs to supply some qualification for the variables in a proposition. After all, given common knowledge about life on the planet Earth, it is a bit redundant to assert that there exists a dog. A more useful statement would declare that some member of a class of thing related somehow to dogs, but defined differently, is actually, in fact, a dog. For example, one might have the class `WA`, with some definitions and axioms that amount to establishing `WA` as the class of all "working animals"—animals that perform some function guided by and useful to humans in performing certain tasks (such as police patrol horses, seeing-eye dogs, and so forth). It might then be useful to have a theory concerning working animals and to include within it (possibly as a theorem, provable from other statements) $\exists(x)(\mathtt{WA}(x)\,\mathtt{and}\,\mathtt{dog}(x)$. This asserts that some things are working animals and are also dogs—that is, there are working dogs. Here, two predicates— `WA` and `dog`— have been combined using a logical operator (`and`) and a quantifier ($\exists$) to form a statement which is relatively useful and can be evaluated with respect to its validity. In logic, statement forms and statements (propositions) are all often referred to as formulae, or more specifically, "well-formed formulae".

B Constructing Triangles from Point and Line Primitives: An Example of a Concept Colimit

This exposition presents the details of a concept diagram and a colimit derived from it. Concepts and their morphisms exist either in a category of theories or of theory presentations (formal specifications). The presentation here is meant to be self-contained. As such, it repeats some of the main text, although only a minor part.

A specific example of concepts, concept morphisms, commutative diagrams and colimits can be seen in expressing the concept of a triangle as a geometric construct obtained by joining three line segments by pairs. The example begins with a concept T_1, a very basic theory of points and lines. In this presentation, points are regarded as undefined quantities and lines are quantities defined in terms of points. This is done through a logical predicate `on` that has two arguments, a point and a line, and is true just in case the point is associated with (or "lies on") the line. The `on` predicate is used in an axiom to express the notion that any two distinct points are "on" some unique line (see [5] for the use of this axiom in several different geometries).

```
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 first line contains the declaration `Spec T1`, announcing a specification for concept T_1. The statement line `sorts Points,Lines` introduces the most basic sorts of the concept T_1. Sorts are "logical containers" which are used to distinguish between the different types of things referred to by the variables or constants in logical formulas. The things referred to are an example of a specific part of a model for the concept. For example, the universal quantifier (`forall`) portion of the axiom in T_1, `forall(x,y:Points)`, makes it clear that the axiom is a formula applying to all things x and y of type "Points". As a consequence, the antecedent of the first implication of the axiom, `(x not = y)`, is understood to mean that x and y represent two distinct *points*, as opposed to lines, circles, widgets, or any other kind of thing. Aside from listing only the axioms of a concept, a specification lists only the most basic sorts and operations (such as `op on:  Points*Lines -> Boolean`) used in the axioms. There are also many derived sorts, such as products of the given sorts. If the sorts are represented by sets in a model of the concept, then the product sorts are represented by the corresponding cartesian products of the sets. The concept so specified includes all the statement lines one can write that follow as a consequence of those in the specification, such as all theorems one could prove from the given axioms, and declarations of all the derived sorts. Derived sorts are any sorts that can be formed by combining the basic ones in sums, products or other sort constructs (we will require only product sorts in our examples, to provide "containers" for the pairs, triples or higher-order tuples of things "contained in" the basic sorts). The notion of sorts, together with the operations associated with them in concepts, is similar to the notion of abstract data types in software program specifications.

Notice that T_1 also states the existence of three labelled, but otherwise unspecified, points `p1`, `p2` and `p3`. This is done through the use of the statement form `const X:Points`, which is a way of stating that there exists a specific (but otherwise indefinite) point with the label `X`. The three constants may or may not represent distinct points: Their separate nature must either be stated as an axiom of T_1, or provable from other axioms of the concept. However, there are no axioms in T_1 that would apply.

An equivalent to the use of sorts in sorted theories is the use of one-variable predicates in un-sorted theories. For example, the predicate `point` (in place of the sort `Points`) has the truth value `true` if its argument represents a point. Were we using un-sorted theories, the statement lines

```
const p1
const p2
const p3
point (p1) and point (p2) and point (p3)
```

would appear in T_1 in place of the lines

```
sorts Points, ...
const p1:  Points
const p2:  Points
const p3:  Points .
```

The line `op on : Points * Lines -> Boolean` in the specification of T_1 introduces an *operation symbol* denoting a function, in this case a predicate which maps an ordered pair (x,l) consisting of a point and a line to a truth value `T` or `F` in the sort `Boolean`. The meaning of `on (x,l)` is the statement "point x lies on line l". That is, for specified values of the variables x and l, `on (x,l)` evaluates to `T` (true) if x represents a point on l, `F` if not. The sort `Boolean` is part of a concept of logical operations that is implicitly included in every concept. Versions of formal logic containing predicates allow for highly expressive formulas, or statements, that employ functions and quantifiers (`forall` and `exists` correspond to the usual universal and existential quantifiers $\forall$ and $\exists$, respectively).

Theorem-proving software programs use formal specifications such as the one for concept T_1 above in applications such as the formal verification of electronic hardware component designs. The use of sorts with an associated type-checking mechanism is one of many mathematically valid devices that can improve clarity in the expression of theories as well as efficiency in using them.

We next express three concepts T_2, T_3 and T_4 by making and modifying three copies of T_1. In each new concept, we add a line constant, re-name the three point constants (for clarity in this presentation—otherwise, the specific names are not important), and associate the latter with the line constant via the `on` predicate. Notice the additional inclusion of an axiom stating that the three point constants denote distinct points. A specification for the first of the three concepts, T_2, is as follows:

```
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
```

A concept morphism $s_1 : T_1 \longrightarrow T_2$ maps the sort symbols `Points` and `Lines` to sort symbols in T_2. Since we want to leave these symbols unchanged, we map `Points` to `Points` and similarly for lines. We also map the

`on` predicate symbol to itself, and the same for the variables. However, the point constants have been renamed in the mapping. All formulas are reformulated with the symbol mapping images to form their images in T_2. Notice that the axiom relating points and lines maps to itself. Obviously, the resulting mapping of formulas is truth-preserving: The only truth in question is that of the image of the single axiom of T_1, whch maps to itself as an axiom of T_2 (of course, all truths of the implicit concept of booleans are also unchanged). Finally, we map the point constants `p1`, `p2` and `p3` to the point constants `pa1`, `pa2` and `paext`, respectively. Here, `pa1` and `pa2` are associated with the line `la` via the `on` predicate and `paext` is intended as a point "external to" `la`. This intention is not stated in T_2 because it is not necessary to make it explicit as yet. The individual symbol mapping relationships are expressed using *maplet* notation:

Morphism s_1:

```
Points ↦ Points
Lines  ↦ Lines
on     ↦ on
p1     ↦ pa1
p2     ↦ pa2
p3     ↦ paext
```

Hereafter, all maplets that leave symbols unchanged will be omitted from morphism descriptions. Since every sort and operation symbol must map to something, this will not result in any ambiguities.

Concept T_3 is the same as T_2 except with point constants `pb1`, `pb2` and `pbext` and line constant `lb` in place of `pa1`, `pa2`, `paext` and `la`, respectively. Similarly, concept T_4 is the same as T_2 except with constants `pc1`, `pc2`, `pcext` `lc` in place of `pa1`, `pa2`, `paext`, `la`. The axiom of T_2 defining `la` in terms of the two points `pa1`, `pa2`, via the `on` predicate is replaced in T_3 by an axiom with the same syntactic structure, but defining `lb` in terms of the two points `pb1`, `pb2`,. Similarly, it is replaced in T_4 by the axiom defining `lc` in terms of the two points `pc1`, `pc2`,. The concepts T_3 and T_4 are:

```
Concept T3
    sorts Points, Lines
    const pb1:  Points
    const pb2:  Points
    const pbext:  Points
    const lb:  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 (pb1, lb) and on (pb2, lb) and (pb1 not= pb2)
end


Concept T4
    sorts Points, Lines
    const pc1:  Points
    const pc2:  Points
    const pcext:  Points
    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 (pc1, lc) and on (pc2, lc) and (pc1 not= pc2)
end
```

We now define morphisms $s_2: T_1 \longrightarrow T_3$ and $s_3: T_1 \longrightarrow T_4$, both the same as s_1 but with the following mappings of point constants in place of those of s_1:

Morphism s_2:

```
Points ↦ Points
Lines  ↦ Lines
on     ↦ on
p1     ↦ pbext
p2     ↦ pb1
p3     ↦ pb2
```

Morphism s_3:

```
Points ↦ Points
Lines  ↦ Lines
on     ↦ on
p1     ↦ pc2
p2     ↦ pcext
p3     ↦ pc1
```

The objects T_1, T_2, T_3, T_4 and morphisms s_1, s_2, s_3 form a diagram Δ. A colimit for Δ has the requisite cocone, as shown in Figure 7 , 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, $\overline{\Delta}$ as shown in the figure, is commutative, with

$$\ell_4 = \ell_2 \circ s_1 = \ell_3 \circ s_2 = \ell_4 \circ \ell_3 . \tag{19}$$

The specification 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
```

Spec T_5 is a "blending" or "pasting together" of T_2, T_3 and T_4 along their common sub-concept T_1. This is because of the commutativity of the defining diagram $\overline{\Delta}$ of the colimit. For the equality (19) 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, we have re-assigned the name of the common T_1 symbol to the merged-image symbol in T_5 for each such case. 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 appears in the definition of two different lines. This is because each of them appears in concept T_1 but is mapped to three points, one in each concept T_2, T_3, T_4, via the three morphisms from T_1 to those concepts. In two of these concepts, its image point appears in the definition of a line, but as a different point on a different line in each concept. In the remaining concept, it appears as an "external" point, not on the line named in that concept. For example, `p1` 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`.

A theorem in category theory can be used to derive an algorithm for calculating colimits in any category having colimits for all diagrams (the dual to The Limit Theorem—see [39]). The category **Concept** is one such category. Thus, the apical object T_5 and leg morphisms ℓ_1, ℓ_2, ℓ_3, and ℓ_4 in the example can be derived from the objects and morphisms of the base diagram, Δ. This confers a great advantage on the use of category theory in knowledge-based system development. Theories and their morphisms (or formal specifications and their morphisms) can be used to specify the intended semantics of software or other kinds of system components. The colimit calculation and the structure-preserving mappings of category theory together provide a mathematically rigorous as well as automated technique for constructing the full system from diagrams[53]. The same kind of mathematics can be applied to an analysis of neural network representations of knowledge, where the knowledge is analyzed as a structure of concepts and morphisms acquired incrementally from the network's input environment. This knowledge structure describes the semantics of the network as a distributed system of interconnected computational components.

References

[1] Jiri Adamek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories*. Cambridge University Press, 1990.

[2] 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.

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

[4] G. 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.

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

[6] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

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

[8] G. A. Carpenter and A-H. Tan. Rule extraction: From neural architecture to symbolic representation. *Connection Science*, 7:3–27, 1995.

[9] 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.

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

[11] Antonio Damasio. Time-locked multiregional retroactivation: A systems-level proposal for the neural substrates of recall and recognition. *Cognition*, 33:25–62, 1989.

[12] A. C. Ehresmann and J.-P. Vanbremeersch. Information Processing and Symmetry-Breaking in Memory Evolutive Systems. *BioSystems*, 43:25–40, 1997.

[13] 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.

[14] K. Fukushima. Neocognitron: A self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[15] Michael S. Gazzaniga. Organization of the human brain. *Science*, 245:947–952, 1989.

[16] 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.

[17] Michael J. Healy. Colimits in memory: Category theory and neural systems. In J. S. Boswell, editor, *Proceedings of IJCNN'99: International Joint Conference on Neural Networks, Washington, DC*, pages 492–496. IEEE Press, 1999.

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

[19] Michael J. Healy. Category theory applied to neural modeling and graphical representations. In M. Gori S.-I. Amari, C. L. Giles and V. Piuri, editors, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks: IJCNN2000, Como, Italy*, volume 3, pages 35–40. IEEE Computer Society Press, 2000.

[20] 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.

[21] 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.

[22] Michael J. Healy and Thomas P. Caudell. 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)*, page 2656. IEEE,INNS, IEEE Press, 2002.

[23] Michael J. Healy and Thomas P. Caudell. From categorical semantics to neural network design. In *The Proceedings of the IJCNN 2003 International Joint Conference on Neural Networks. Portland, OR, USA. (CD-ROM Proceedings)*, pages 1981–1986. IEEE,INNS, IEEE Press, 2003.

[24] Donald O. Hebb. *The Organization of Behavior*. John Wiley and Sons, New York, 1949.

[25] 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.

[26] Bernd Heinrich. *Mind of the Raven*. HaperCollins Publishers, Inc., New York, 2000. Paperback edition.

[27] Joy Hirsch, Diana Rodriguez Moreno, and Karl H. S. Kim. Interconnected large-scale systems for three fundamental cognitive tasks revealed by functional mri. *Journal of Cognitive Neuroscience*, 13(3):389–405, 2001.

[28] R. Jullig and Y. V. Srinivas. Diagrams for software synthesis. In *Proceedings of KBSE '93: The Eighth Knowledge-Based Software Engineering Conference*, pages 10–19. IEEE Computer Society Press, 1993.

[29] Eric R. Kandel. From nerve cells to cognition: The internal cellular representation required for perception and action. In Kandel et al. [30], pages 381–403.

[30] Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell, editors. *Principles of Neural Science*. McGraw-Hill, New York, fourth edition, 2000.

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

[32] W. M. Kelley, C. N. Macrae, C. L. Wyland, S. Caglar, S. Inati, and T. F. Heatherton. Finding the self? an event-related fmri study. *Journal of Cognitive Neuroscience*, 14(5):785–794, 2002.

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

[34] 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.

[35] James L. McClelland, David E. Rumelhart, and Geoffrey E. Hinton. The appeal of parallel distributed processing. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 3–44. The MIT Press, 1986. This volume contains an exposition on bias nodes as a neural mechanism for varying the thresholds of a layer of nodes.

[36] J. Meseguer. General logics. In *Logic Colloquium '87*, pages 275–329. Science Publishers B. V. (North-Holland), 1987.

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

[38] Isabelle Otto, Philippe Grandguillaume, Latifa Boutkhil, Yves Burnod, and Emmanuel Guigon. Direct and indirect cooperation between temporal and parietal networks for invariant visual recognition. *Journal of Cognitive Neuroscience*, 4(1):35–57, 1992.

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

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

[41] S. C. Rao, G. Ranier, and E. K. Miller. Integration of what and where in the primate prefrontal cortex. *Science*, 276:821–824, 1997.

[42] Maximilian Riesenhuber and Tomaso Poggio. Are cortical models really bound by the binding problem? *Neuron*, 24:87–93, 1999.

[43] Clifford B. Saper, Susan D. Iversen, and R. S. J. Frackowiak. Integration of sensory and motor function: The association areas of the cerebral cortex and the cognitive capabilities of the brain. In Kandel et al. [30], pages 349–380.

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

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

[46] Larry R. Squire. Declarative and nondeclarative memory: Multiple brain systems supporting learning and memory. *Journal of Cognitive Neuroscience*, 4(3):232–243, 1992.

[47] Larry R. Squire and S. Zola-Morgan. The medial temporal lobe memory system. *Science*, 253:1380–1385, 1991.

[48] Klaas E. Stephan, John C. Marshall, Karl J. Friston, James B. Rowe, Afra Ritzl, Karl Zilles, and Gereon R. Fink. Lateralized cognitive processes and lateralized task control in the human brain. *Science*, 301:384–386, 2003.

[49] Viggo Stoltenberg-Hansen, Ingrid Lindstroem, and Edward R. Griffor. *Mathematical Theory of Domains*. Cambridge University Press, 1994.

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

[51] Stephen Vickers. *Topology via Logic*. Cambridge University Press, 1993.

[52] Ingrid Wickelgren. Getting a grip on working memory. *Science*, 275:1580–1582, 1997.

[53] 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.

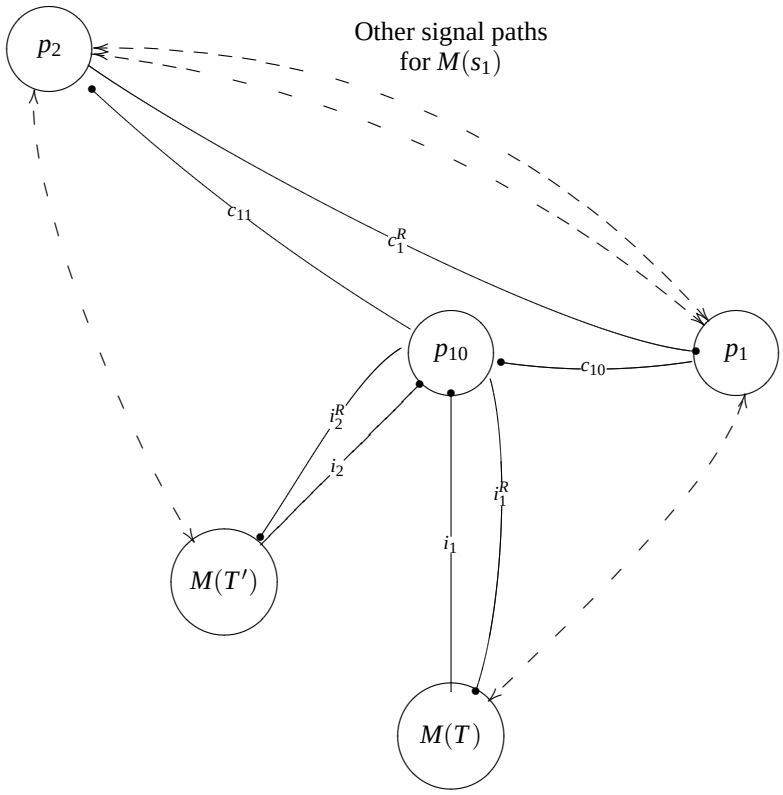


Figure 13: **A connection path representing one of the symbol maplets,** $\texttt{p1} \mapsto \texttt{pa1}$ **(see text), in an explicit representation of a concept morphism image** $M(s_1)$**. The intermediate node** p_{10} **is shown representing a coproduct** $M(T)+M(T')$ **of the nodes** $M(T)$ **and** $M(T')$ **representing concepts** T **and** T' **that contain essentially only the point constant symbols** `p1` **and** `pa1`**, respectively. In addition to its injection morphism** i_1 **to the coproduct object** p_{10}**, the neural object** $M(T)$ **is also the domain of a morphism with codomain** p_1**. Similarly, in addition to being the domain of the injection** i_2**,** $M(T')$ **is also the domain of a morphism with codomain** p_2**. The latter morphisms are images of concept morphisms, since the symbols of the simple concepts** T **and** T' **are in the concepts** T_1 **(represented by** p_1**) and** T_2 **(represented by** p_2**), respectively. However, there is no concept morphism represented solely by the paths containing connections** c_{10} **and** c_{11}**. Also, a coproduct was used to describe** p_{10} **instead of a more general colimit that would unify the two copies of the sort** `Points` **in** $M(T)$ **and** $M(T')$**. This is because to represent the maplet it is sufficient to associate the path with the symbols** `p1` **and** `pa1`**. The proper unification of concepts is not necessary for this. The connection** c_1^R **is the reciprocal for the morphism** $M(s_1)$**. It represents the model-space morphism** $\texttt{Mod}(s_1)$**.**

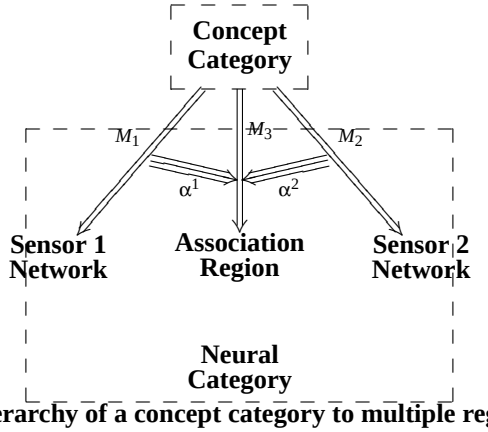


Figure 14: **Functors map the hierarchy of a concept category to multiple regions. Natural transformations represent coherent interconnections between hierarchy representations.**

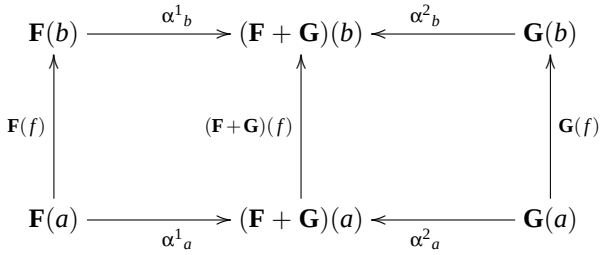


Figure 15: **A coproduct in $\mathcal{D}^{\mathcal{C}}$ "pastes together" commutative squares along the morphism images of the coproduct functor.**

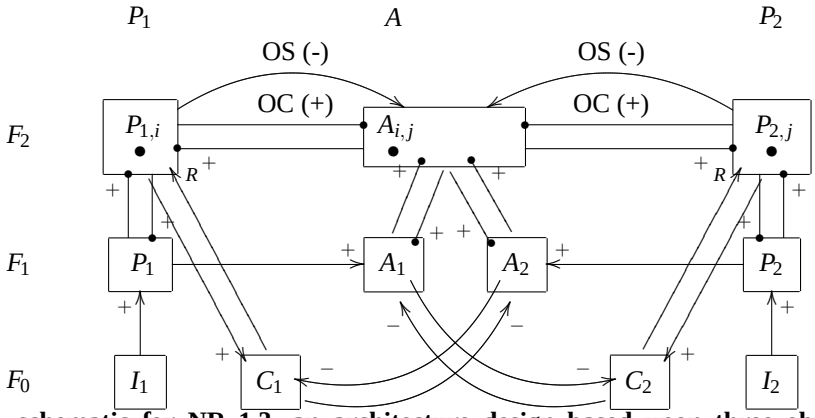


Figure 16: **A schematic for NR 1.3, an architecture design based upon three objects (functors) P_1, A and P_2 and two morphisms (natural transformations) $\alpha^1 : P_1 \longrightarrow A$ and $\alpha^2 : P_2 \longrightarrow A$ in the functor category Neural$^{\text{Concept}}$.**