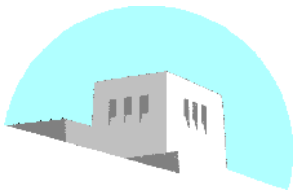
3-23-2016

The Neural Representation of Concepts at the Sensor Level

Michael John Healy

Thoms Preston Caudell

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING



SCHOOL OF ENGINEERING
UNIVERSITY OF NEW MEXICO

The Neural Representation of Concepts at the Sensor Level

Michael John Healy
Department of Electrical and Computer Engineering
e-mail: mjhealy@unm.edu

Thomas Preston Caudell
Department of Electrical and Computer Engineering and
Department of Computer Science
e-mail: tcaudell@unm.edu

University of New Mexico
Albuquerque, New Mexico 87131, USA

UNM Technical Report: ECE-TR-16-0001R

Report Date: March 23, 2016

Abstract

This report[1] presents a mathematical model of the semantics, or meaning, of the connectionist structure and stimulus activity of a neural network, whether artificial or biological. The mathematical model associates concepts about sensed objects with the neuron-like nodes in a neural network and composable concept relationships with the connection pathways in the network. Category-theoretic constructs, specifically colimits, limits, and functors, organize the concept structure and map it to a formal neural network in a structure-preserving manner. Starting with a simple example of a neural vision system, we show that this mathematical model of neural network structure and activity can be used to derive connectionist architectures that work as intended. We also claim an additional advantage of this approach: A properly-functioning connectionist architecture has an accompanying concept representation and this representation is both local and distributed. These properties are derived from the category-theoretic formalism described here.

Keywords

category, colimit, concept, diagram, function, functor, limit, morphism, neural network, set, theory

[1]Note: This is revision ECE-TR-16-0001R to the technical report ECE-TR-16-0001. Aside from minor corrections, the revision addresses a change to the definition of the adjacency of sensor pixels in Section 6.3.

1 Introduction

The purpose of this report is twofold: First, it provides a gradual introduction to a mathematical approach for understanding the semantics of neural networks. Second, it shows with an example that the use of this approach can improve the design of artificial neural systems. The mathematical approach has been presented before in several sources including [9, 7, 8, 6, 10]. It is based upon category theory and has recently been called the Categorical Neural Semantic Theory (CNST). The provided example is a series of neuron-and-synapse or connectionist structures within a neural network receiving sensor stimuli from objects sensed by a simple vision system. The neural network's objective is to process the sensed objects in a manner that credibly provides not only object recognition but object understanding: At issue in investigating concept representation in neural networks—and ultimately the brain—is the question of whether a connectionist network can be shown to acquire human-understandable descriptions of the world around it by processing the stimuli that activate its sensors.

Some in cognitive science have argued against representations in neural networks as concepts, or descriptions which can be expressed in symbols. This view largely centers upon the argument that neural networks are incapable of symbol processing in the sense of Newell and Simons' Physical Symbol System hypothesis [15, 16]. A classic example of this argument is presented by Fodor and Pylyshyn [5]. They define symbol processing as the manipulation of expressions which form a linear syntax with a grammar, as in a natural language. A key property of a symbol-processing system is that of having a *combinational constituent structure* (the syntax, a collection of grammatical rules for combining symbolic expressions to form more complex expressions) wedded to a *combinatorial semantics* (where the meaning of a complex symbolic expression depends upon the meanings of its constituent expressions and the syntactic rules used in combining them). For example, the meaning of the expression `A and B`, where `A` and `B` symbolize propositions, is that an instance in which `A` is true is also an instance in which `B` is true, and conversely. Thus, the meaning of `A and B` depends not only upon the separate meanings of `A` and `B`, but also the meaning of the conjunctive symbol `and`, as in `birds are warm-blooded and birds have feathers`. Even if neural networks are capable of representing conjunctions in which `A` and `B` are specific propositions, Fodor and Pylyshyn doubt that they can represent conjections `A and B` in which `A` and `B` are propositional variables. The use of propositional variables allows for the substitution of specific propositions into propositional variables to express different situations. The argument against this capability is intimately connected with the local-distributed issue for the discipline of neural networks, or connectionist systems.

The local-distributed issue is the question of whether single neural cells can represent concepts about objects or whether each specific concept representation requires the activity of a population of neurons. At one extreme, "local" means that *only* a single cell is required to represent an object, and at the other extreme "distributed" means that there is *no* single

cell capable of signifying an object's presence. Clearly, there is room for a middle ground between these two extremes.

Here, we show that both the concept representation and local-distributed controversies can be addressed simultaneously and, in fact, must be. We show this by applying the CNST. The simulations performed show that category-theory-based neural network design can resolve the two issues and also lead to significant improvements in neural network performance.

1.1 The contents

It is important that we be clear from the beginning about what we are claiming in this paper. First, we argue that representation involves *both* local and distributed processing; that is, our claim lies in the often-excluded middle. On the one hand, the CNST models a representation as local, represented by a single cell. However, a cell carrying the representation serves as a unique *indicator* that the object is present in the network's inputs, not the lone representative of the *semantics*, or full meaning of the object's representation. In fact, the indicator cell must interact with an interconnected array of cells, and they and the way they are interconnected collectively provide the semantics associated with the indicator cell's representation. The array cells, in turn, act as indicators by interacting with their own interconnected retinues of cells, or ultimately with sensor, actuator or pattern-generating cells. The precise structures these inerconnected cell assemblies form will be described in detail, but for the moment notice that we are describing a system that is both localist and distributed.

This brings us to a second claim. The interactions of and between groupings of cells, with each group associated with an indicator cell, can be seen as a sort of symbolic processing. This type of system is distinct from the traditional, classical model of symbolic processing. This will be explained in detail in the final section of this paper. Hence forth, to emphasize this point and because it better expresses the mathematical model of the CNST, we call a representation a *concept* as an alternative to the term "symbol", which can be quite misleading because of its historical connotations. Finally, it is important to be clear that the use of the term "concept" is not meant to imply that the neural network is "thinking". Instead, a concept is a human-understandable *description* or *explanation* of the stimulus to which an active neural cell is responding. In this context, the "thinking" is done by the neuroscientists and neural network analysts who use concepts to express the semantics of a neural network.

Finally, we claim that the mathematical approach to concept representation taken here can be used to not only understand how neural structures form concept representations, but also to create neural network architectures that actually satisfy stated design objectives in concept representation. This claim is the impetus behind development of the CNST, and evidence for it is shown in following sections.

The organization of the report is as follows. In Section 2 we introduce our main example and use it to show how the CNST can address the local versus distributed issue. The method for doing so employs category-theoretic constructs known as *colimits* and*limits*. Colimits are treated in some detail beginning with Section 3. Section 4 introduces category theory. This branch of mathematics is unfamiliar to many, having been employed only in recent decades in applications outside mathematics. To allow for this, Section 3 prepares the way with an initial mathematical model for the example in a more familiar, set-theoretic setting in which concepts are expressed as sets of sensor elements. Using a simple neural vision example, we show in a step-by-step fashion how a certain neural structure solves the problem of attaining an unambiguous representation of a simple geometric shape object appearing in the visual field of an imaging sensor. After formally introducing category theory in Section 4, in Section 5 we describe a category which we use to formalize the structure and operation of a neural network. We then show how a hierarchical concept structure can be mathematically mapped into this neural category to achieve a partial formalization of concept representation. In Section 6 we take a step toward the full formalization of concept representation in a category more expressive of concepts and their relationships. Section 7 is the conclusion.

2 A Neural Vision System and Its Semantics

Let us establish a goal on which to base our arguments. The objective is a neural network that processes its inputs from its environment in a way that convinces us that it contains the basis for "understanding" the environmental stimuli it processes. By "understanding" we mean that we the analysts can trace through the network and see that its connectionist structure and processing allows us to make unambiguous, informed statements about the objects that produce the stimuli it detects. Also—and this is important—we can do this without prior knowledge of what objects in the environment the network is currently sensing. What we do have is the knowledge related to the modality of each sensor—whether it is visual, auditory, somatic, or processes microwave, reflected radar illumination, and so on. So, based upon our knowledge of how the network acquires its stimuli but not where they came from, what it does with them can be said to be human-understandable—it can be expressed as a system of concepts.

In fact, our running example throughout the remainder of this paper is a simple vision system in a neural network "brain". Of course, simplicity can be deceptive, and we can expect complexity to arise when attempting to make the system useful. Because of this, the system we initially propose will require improvement to achieve the desired functionality.

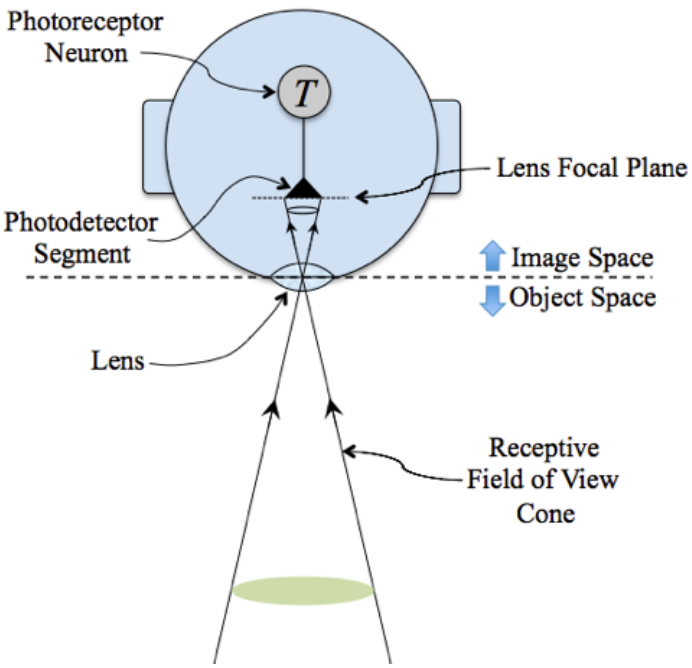


Figure 1: **The agent R1.0 with its simple "brain" consisting of a single sensory neuron receiving brightness stimuli from a visual focal plane.**

2.1 A robot vehicle

The inital model is illustrated in Fig. 1, where a top view is shown suggestive of a "head" containing an "eye", a visual sensor. This is the head of intelligent agent R, version 1.0, a robot vehicle within which the visual sensor and brain are located. Two-dimensional shapes appear in R's visual field, shown as a cone within the Object Space, forward of its head. This cone illustrates the gathering of light by the lens as shown. Behind the lens, within the Image Space, is another cone within which the light from the Receptive Field of View cone is focused by the lens, impinging upon a photocell at the Lens Focal Plane. Let us suppose the robot brain consists of a single neuron T connected to the photocell. The neuron T has a *receptive field* (RF), the Object Space cone. The single neural cell responds by becoming active if the photocell's output to it exceeds a threshold; T then emitts an output signal. To narrow this down further, the particular stimulus the network is to represent as a concept is a triangle with a particular shape and orientation as shown in Fig. 2.

By examining the object shown in Fig. 2, we humans easily identify it as a triangle. We also understand triangles because we know how they are composed as geometric figures. But does the robot? Its single-neuron brain is simply responding to the appropriate stimulus; this does not in and of itself constitute knowledge. A single neuron processing a brightness stimulus cannot be said to recognize a triangle as an object distinct from other stimuli, to say nothing of having knowledge of triangles. That is, it does not represent the

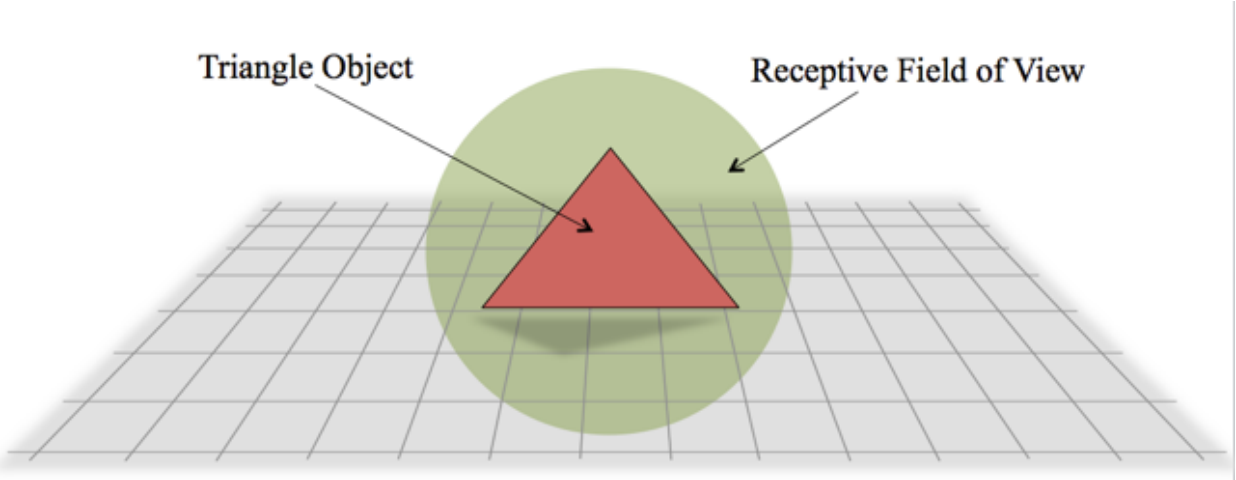


Figure 2: **The receptive field (RF) for the neuron T of agent R1.0, showing a triangle object in the RF. The RF, in turn, lies within the object space.**

concept of a triangle. The semantics of the neuron T is simply the response to a stimulus. If R is an autonomous vehicle and T has an axon through which it can provide a "Go" signal to, say, its wheels, R can be designed so that it will move depending upon whether T receives a sufficent stimulus from the photodetector. But T is acting only as a simple off-on switch, switched to "on" by sufficient light intensity. Braitenberg in his book *Vehicles: Experiments in Synthetic Psychology* [3] discusses at this most basic level the semantics of vehicles controlled by a series of incrementally more complex systems, beginning with one as simple as the one presented here. One way of looking at the semantics of machines like this is to ask how much they "know". A convenient definition of "Knowledge" for our present purpose is "the ability to manipulate data" [18], another way to define concept representation. This ability can be applied in a robot to transform sensor data into action commands. Unfortunately, our robot cannot manipulate anything; it can only respond in a fixed fashion to a stimulus. Obviously, our vehicle R needs to have a brain with more structure if it is to respond in an effective and flexible manner to a variety of inputs.

2.2 A local-distributed neural architecture

Of course, it is no secret that knowledge about an item depends upon a familiarity with its constituents. The knowledge of what it means for an object to be a triangle, how to recognize one and perhaps reason about it or manipulate it, depends upon knowledge of its geometric properties. This can be expressed in various ways, but a simple one is that a triangle is a three-sided polygon, a closed plane curve composed of three line segments enclosing three angles. Since there are only three angles, the sides of each angle must be colinear with two of the sides, otherwise there would be an extra angle at the point of intersection of the angle and a side. These facts suggest that a seemingly simple definition of a visual object can be rather complex if one discards all that we humans

implicitly assume because of the familiarity with plane geometry hidden away in our own brains. Evidently, agent R needs a brain with more neurons to separately and uniquely represent lines and/or angles, combine them appropriately, and then signal the presence of a triangle to T. For T to reliably signal the detection of a triangle would require that it have a high enough activation threshold that the presence of all constituents in an appropriate combination would be required. In fact, this would contribute to the ability to "understand" the concept of a triangle. The activation of not only T but also the cells representing its constituents would provide R with the ability to "parse" the stimulus into its component stimuli, a first step in knowledge representation.

Our second design, R1.1, improves upon R1.0 by providing six "feature-detector" neurons, three specialized for edges or sides and three specialized for vertices or angles. We shall refer to vertices and angles interchangeably; our definition of the relationship between edges and sides will be given presently (Fig. 3 shows a triangle whose sides each consist of a single edge). The receptive fields of the "feature detector" cells lie largley within the RF of R1.0's single triangle-detector neuron (Fig. 4). To support this design, we replace the single triangle-detector photocell with six photodetectors, the appropriate one for each of the six feature-detector neurons. The feature-detector output axons fan into and synapse upon the triangle-detector neuron T (see the R1.1 design in Fig. 5). Let us assume just for now that each feature-detector neuron of R1.1 has a stimulus threshold value so that it will produce a unit output if its intended feature (either edge or angle) appears within its RF, and a zero otherwise. The layout of this architecture is shown in Figs. 4 and 5. We fix all of the input synaptic weights of the connections from these cells to T to unity and provide T with an output threshold tuned to a high value—say, 5.1. Using a simple neuron model in which T sums its inputs and tests the sum against its threshold, T will be activated if and only if all six of its inputs are sufficiently active so that their outputs sum to a value greater than 5.1. As before, T is to signal the detection of a triangle by becoming active. Satisfying the definition of "triangle" has forced us to produce R1.1, somewhat more complex than R1.0.

Notice that the robot R1.1's brain now has six neurons helping to define a triangle and one neuron that represents a triangle based upon its inputs from the six. Because of this arrangement, we can claim that this neural arcitecture is an example of a system that is both local and distributed. It is local because the output of the cell T indicates whether or not a triangle has been detected, and it is distributed because it is the feature detectors and their inputs through the connections to T that really determine that the components of a triangle, hence a triangle, are present. Conceivably, this local-distributed system could be providing not only detection, but the desired analysis of the detected object based upon the network's representation of its features. Cen we regard R1.1 as signaling and, further, understanding "triangle" when one appears in its visual field?

This question can be rephrased in terms of semantics, or meaning: Does neuron T in R1.1's brain actually mean "triangle" when it is active; that is, can R manipulate the data, can we claim that it registers the concept of a triangle because the object appears

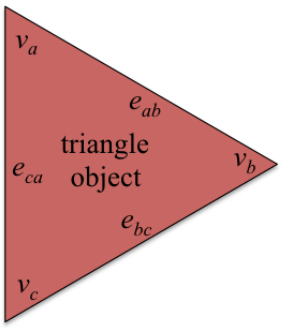


Figure 3: **A world object triangle with labeled components.**

with its components identified? Although we can trace backward through the neural network and find active neurons which we know represent the required elements of a triangle because we agreed that R has the needed angle and side detectors, in the end this is not very convincing. First of all, the original concept representation problem has been moved only a step back, because it has not been specified just how the feature detectors work. What guarantees that other kinds of features appearing within their receptive field do not activate the feature-detector neurons and thereby neuron T? This design for R requires feature-detectors with more sophistication than simply the ability to detect a brightness stimulus, and incorporating these as input devices does not provide an understanding of the semantics of *neural network* architectures. This defect stems from the following fact. Because the six neural cells receiving input from the photoreceptors are simply responding to a brightness stimulus, the knowledge represented by the photoreceptors' designs for detecting specific image features (angles and sides) is inaccessible to the neural network. In fact, many issues remain to be addressed in object representation in brightness stimuli. Among these are shape representation for a variety of image objects, part-to-whole relationships for a range of assemblies from elementary brightness stimuli through shape features such as edges and angles to complex shapes, the separation of contiguous bodies of brightness stumuli (i.e., image objects) from each other and from a background stimulus across image space, the significance assigned to bodies having varying brightness, leading to boundary contours within image objects, and potential occurrences such as shape distortion and sensor noise.

Although all of these issues require attention, in this paper we shall focus upon simple geometric shapes and subsume brightness variation by averaging over stimulus regions.

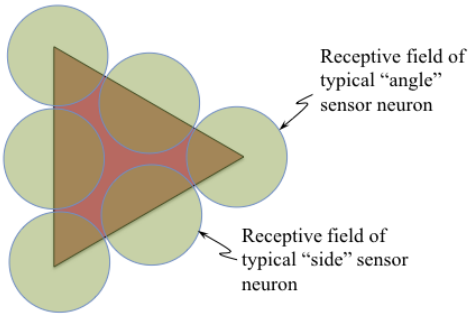


Figure 4: **The receptive fields of sensory neurons that detect the components of a triangle in the visual field.**

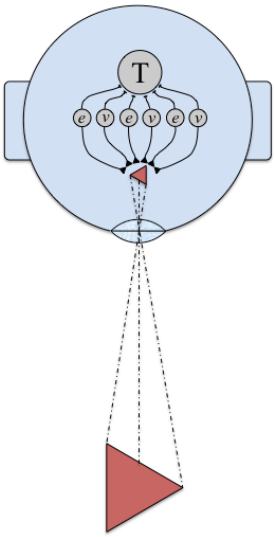


Figure 5: **Agent R1.1 in a world with nearby objects. It has an eye consisting of a lens, six sensory neurons labeled with the type of feature they detect projected onto the focal plane, and the triangle-detector neuron labeled "T".**

However, we will address part-to-whole relationships and shape distortion and noise.

A second reason R1.1 lacks credibility for our design goal is that all this apparatus works only if a triangle of the correct shape appears in the correct position in R's visual field. One could argue that if the triangle and its features were required to be fixed at the indicated locations in the visual field and oriented properly, the sides and angles would have to be arranged exactly as required. But this argument requires that only a particular triangle, with angles and sides sized appropriately, would be detectable. Otherwise, the architecture must have the capability of detecting objects with invariance to location and orientation. Invariance is a subject to be deferred to a later section.

Third, and this is our primary focus for now, merely connecting some feature detector cells as inputs to T says nothing about how the features are arranged within the image. With the inputs specified, neuron T is not capable of discriminating between a triangle and other patterns that contain three edges and three angles within the feature-detectors' RFs. In fact, Fig. 6 shows three possible arrangements of three edges and three angles with the required triangle features present in the appropriate RFs but not aligned properly. The photoreceptors could be designed to rule out such misalignments, but as stated before, such complexity in responding to stimuli was the purpose in giving R a brain. The fact is that in R1.1, neuron T is simply responding based upon whether it receives sufficiently strong stimuli from six sources. Moreover, as will be shown, their response in a unified manner to the sensory stimuli is not reliable. If R1.1 were an organism whose survival depended upon its correctly distinguishing between triangles and other objects, our "improved" robot still would be in trouble. A further improvement in R is needed, but this calls for a better understanding of the semantics of systems like R.

2.3 Booleans—an attempt at a formal semantic model

Answering questions about the semantics of computational systems is the purpose of formal or mathematical semantic models. To begin framing this discussion, consider a Boolean model of connectionist computation. The earliest known such model is that of McCullough and Pitts [13], who developed a full symbolic logic calculus for neural networks. Other Boolean models have appeared occasionally in the literature (for example, see the discussion in [1]).

Boolean operations commonly appear as operations upon bit patterns generated by electrical circuit elements. In formal logic, they are performed upon symbolic quantities, where the symbols are propositional formulas which serve as codes for statements about items in some domain of discourse. If the domain is the input/output environment of a neural system whose cells produce Boolean outputs 1 (meaning "true") and 0 ("false"), the symbols might represent cats, dogs, animals in general, the furniture in a room, airplanes, and other items that can serve as stimuli for the sensory system of the network. Each symbol can correspond to a proposition such as "A triangle has been detected", and the

Figure 6: **Three shapes using angles and sides that are not triangles.**

binary 1 or 0 output of its cell corresponds to the proposition currently having the value "true" (1) or "false" (0). Suppose that A and B represent two properties the items may have in such a system, with each represented by cells appropriately connected within the network. If a description of some item is input to the system, say as images for our agent R, each Boolean expression takes on a truth value which can be read as the output of the neurons in R's brain. For example, A might represent the property "The item has the color red". Then in a neural system, a cell appropriately thresholded so that it represents A would output a value 1 if the item currently being sensed by the system is indeed red in color, and a 0 if not. The Boolean operations enter the picture when queries to the system can be made concerning combinations of properties. The Boolean expression $A \wedge B$ yields the value "true" if the current input item has both properties A and B, otherwise it yields the value "false". Similarly, the expression $A \vee B$ yields "true" if the item has either property A, property B, or both, otherwise it yields "false". Finally, $\neg A$ yields the value "true" if the item does not possess property A, and "false" if it does. Boolean expressions like this can be combined recursively and involve many of the basic symbols $A, B, \ldots$, called *atoms* or *primitives*, to yield arbitrarily complex expressions. By the rules just given and certain others for Boolean propositions, an expression can be evaluated recursively (tracing through the recursive construction of the expression) to transform the current truth values of its individual atoms into a single truth value for the expression.

The neurons shown in Fig. 7 have input connections, connection weights and thresholds designed to implement each of the operations $\wedge, \vee$ and $\neg$. Denote the thresholded outputs of the triangle-detector cell by T and the six feature-detector (edge- and angle-detector) cells by s_j $(j = 1, \ldots, 6)$. Each output value is therefore a binary 1 or 0 de-

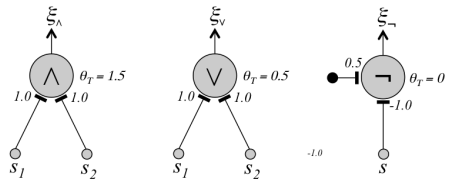


Figure 7: **Neurons that implement three operations in Boolean algebras; (a) conjunction, $\wedge$, (b) disjunction, $\vee$, and (c) negation, $\neg$. Note that negation requires a bias neuron with a fixed unity output.**

pending upon whether the input to the cell is above or below its threshold value. Correspondingly, denote by ξ_T and ξ_{s_j} $(j = 1, \ldots, 6)$ the Boolean truth values for the symbolic quantities represented by the cells, "true" for a binary 1 and "false" for a binary 0 as before. Then, if the feature represented by the cell output s_j is present in the image, so that $s_j = 1$, the corresponding truth value is $\xi_{s_j} =$ true and is false if the binary output is 0, meaning that the feature is not present. Similarly, $T = 1$ corresponds the presence of the triangle, with 0 corresponding to false otherwise.

Now, the binary output value T expressed as a thresholded weighted sum of the binary inputs s_j $(j = 1, \ldots, 6)$ is

$$T = [\Sigma_{j=1}^{6} w_j s_j]_+ , \quad Eq.1$$

where w_j is the connection weight for the input s_j to the T cell and $[\,]_+$ represents the thresholding operation, where the function $[x]_+$ is defined as

$$[x]_+ = \begin{cases} x, & x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

If the cells and their weighted interconnections in R1.1's brain represent a Boolean conjunction operation, as shown for the $\wedge$ cell with its two input cells in Fig. 7, then the conjunction truth value ξ_T in terms of the truth values ξ_{s_j} for the features of the triangle is expressed

$$\xi_T = \wedge_{j=1}^{6} \xi_{s_j} . \quad Eq.2$$

This is intended as a formalization of the design illustrated in figures 5 and 8, and it illustrates the formalization of the semantics of a computational system by a symbol system. It also exemplifies one of the problems raised by attempting such a formalization. The cell T is not in actuality a Boolean circuit element unless it has carefully-selected connection weight values w_j and cell outputs s_j in the thresholded, weighted summation of Eq. 1. In fact, assuming the cell outputs all have exact binary values, the connection weights and cell T's threshold value must be selected with precision. Assuming the appropriate weights and T threshold, the network's *intended* semantics have been formalized because a mathematical Boolean operation upon symbols yields Boolean true-false statements about the objects that appear as sensor stimuli. Even so, this formalization does not express the needed information on precisely how the components are joined to form the triangle shape. Also, it relies upon the feature-detectors to perform the task of compensating for shape distortion and noise in the image.

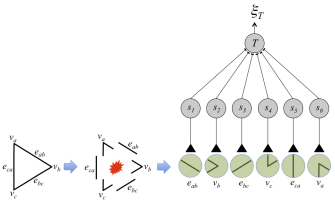


Figure 8: **Conjunction of parts.**

Having examined Boolean modeling, let us pass on to an alternative semantic model. But before proceeding, it is important that we make a further stipulation on our modeling of the neural network's activity. Since our objective is the analysis of neural network semantics, we shall simplify the discussion by dispensing with the discrete dynamic modeling required for a detailed analysis of neural processing, which typically includes the spiking behavior of neurons and the mathematics of dynamic systems. The input-sum-and-signal function formulation we shall use is commonly referred to as a rate-coding model. This model has been assumed in Eq. 1, which expresses the effect of the pixel brightness values on a cell such as T as a one-step process. Unless stepping through time is important in

analyzing network behavior, we shall follow this convention and simply express the effect of the inputs on the output signals at each cell.

2.4 Semantics—going deeper

Because its brain now has six neurons helping to define a triangle and one neuron that represents a triangle based upon its inputs from the six, the robot R1.1 is an example of a system that is both local and distributed. However, it leaves unanswered the questions posed in regard to the semantics of the system: How can we analyze meaning in a neural network with mathematical rigor? A Boolean formal model does not seem to provide the answer we need. We need to begin with a model that provides greater precision in understanding the semantics of representations such as "edge" and "angle" using neurons and synaptic connections. Is there a formalization for this?

The mathematical language of sets and functions, which map one set into another, is used in formalizations such as formal concept analysis (FCA [4]). However, the relationships between the concepts expressed in FCA do not offer the complexity we need. We shall show how sets and functions can be used for our purpose to begin combining the parallel tracks of numerical and symbolic processing to formally express the semantics of neural processing.

Although the discussion is now becoming more mathematical, it will be kept relatively simple if we imagine how objects within our robot's visual receptive field look when projected upon the focal plane. Imagine the focal plane subdivided into tiny cells called pixels, sensor elements which transmit brightness values to agent R's neural network. They are analogous to (but a gross simplification of) the rod and cone cells in a mammalian retina. Think of the (now mathematically analyzed) T cell and in fact each of its edge-detector cells as having several pixels within its RF in Fig. 4, where each receptive field covers an assigned area. Each of the edge-detector cells, or e-cells, receives stimuli as input from the pixels in its RF and, as in Eqs. 1–2, responds with a signal that is a weighted sum over the pixel brightness values. The RF of a typical edge-detector cell is a set of pixels grouped together, referred to as D_e (for edge-detector cell e) in Fig. 9.

Notice that D_e has been subdivided into two regions or subsets, D_e^+ and D_e^-. Two of the potentially many pixels (very small squares) from each region along with their weighted connections to e are shown. The connections from the pixels in D_e^+ to the edge-detector cell e are accompanied by plus signs where they synapse upon e, meaning that these connections transmit excitatory input to e. On the other hand, the connections from the pixels in D_e^- to e are accompanied by minus signs, meaning that these connections tend to inhibit e. Thus, the summation of the brightness values transmitted through the input connections to e emanating from the pixels in D_e^+ tend to activate e, while the input connections to e from the elements of D_e^- subtract the brightness values of these pixels from the overall sum of inputs at e. Because it responds as described to the positive and

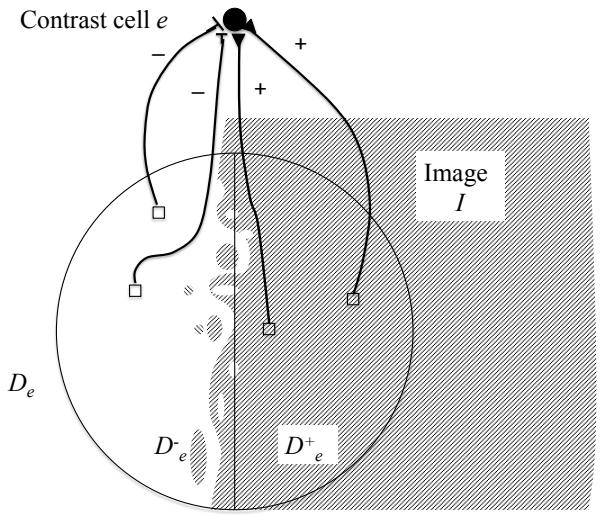


Figure 9: **An edge-detector cell, or contrast cell, with its sensory inputs.**

negative inputs from its RF, an edge detector cell is sometimes referred to as a contrast cell.

The input sum processed by e is expressed

$$\theta_e = \Sigma_{j \in D_e^+} w_j s_j - \Sigma_{j \in D_e^-} w_j s_j \quad (0 \leq s_j \leq 1)\,, \quad Eq.3$$

where the w_j are unsigned connection weight magnitudes representing synaptic strengths and the s_j are pixel brightness values. A cell's output can be expressed in terms of its input sum θ and thresholded activation as $\phi(\theta)$, where ϕ is the cell activation or signal function. Let θ_T denote a typical cell's threshold, the value which θ must exceed for the cell to become activated and produce an output signal. There are several forms of signal function in use; three examples are as follows:

- Heaviside step function:

$$\phi(\theta) = \begin{cases} a, & \theta \geq \theta_T \\ b, & \theta < \theta_T \end{cases} \quad . \quad Eq.4$$

  where a and b are the upper and lower step values the cell can produce as output, with $a > b$.

- Recified linear function:

$$\phi(\theta) = \begin{cases} c\,\theta, & \theta > \theta_T \\ 0, & \theta \leq \theta_T \end{cases} \quad , \quad Eq.5$$

  where c is a constant of proportionality, with $c > 0$,

- Linear function:

$$\phi(\theta) = c\,\theta. \quad Eq.6$$

Hence forth, let the pixel output values be assumed to range between the lower and upper bounds 0 and 1. The upper bound of unity is a simplifying assumption, partly for convenience but also based upon the idea that the dynamic range of each sensor element must be somehow bounded to avoid saturation. With saturation, a subset of the pixels' stimulus magnitudes become so large that they overpower the variation in magnitudes across the sensor array, at best smearing and at worst hiding the information in the image. Now, for a linear neuron the signal function ϕ simply expresses the proportionality of the output signal with respect to the input sum, and a rectified linear function is simply a linear function whose output at or below its threshold is zero. Assume for simplicity that all edge-detector cells have the same signal function ϕ_{edge} and that ϕ_{edge} has the form of Eq. 5, the rectified linear function. Further simplifying, assume that the common activation threshold for edge-detector cells is $\theta_{T,e} = 0$ so that a net positive value in the input summation activates e. Assume, finally, that the constant in Eq. 5 is $c_{\text{edge}} = 1$ for all edge-detector cells. Presently it will become evident that this assumption is reasonable because of the assumption that the pixel outputs are bounded by unity. Under the stated assumptions, then,

$$\phi(\theta) = [\theta]_+. \quad Eq.7$$

These assumptions, a rectified linear neuron with the stated coefficient $c_{\text{edge}} = 1$ and with its output expressed through Eq. 7, will be used throughout this paper for all cells. This provides a baseline model for the investigation of the properties of the neural structures to be proposed here. Nonlinearities and other changes or elaborations can always be imposed to further improve performance if the baseline model can be shown to work for us.

Further constraints can be imposed at this point, again for a baseline model. The edge detector functionality is to be such that if an image, a region of nonzero pixel brightness values, pretty well covers D_e^+ but shows up hardly at all in D_e^-, e will be activated and emit a strong output signal, signaling an edge occurring along the midline separating D_e^+ from D_e^-. Fig. 9 illustrates this for an image I (for ease of illustration, the shaded region contains the brightness stimuli while the clear region has none). Of course, the balance of excitatory versus inhibitory inputs required to activate e can vary widely depending upon the connection weight magnitudes w_j. We can simplify the analysis somewhat by assuming, for connections emanating directly from the pixels, that there is little or no intrinsic bias toward any particular set of pixels in the transmission of stimuli. Therefore, we assume that the magnitudes of the weights over D_e^+ are all approximately the same and are normalized, and the same for D_e^-. The simplest type of normalization is an ℓ_1-like summation with the value unity, where in this case $\Sigma_{j\in D_e^+} w_j = 1 = \Sigma_{j\in D_e^-} w_j$. This type of expression will be understood when we refer to normalization unless specified otherwise. To further simplify the analysis, assume that both the excitatory and inhibitory regions

have the same number of pixels, and that this number is the same for all edge-detectors. Given these assumptions, the edge-detector output will be restricted within the range

$$0 \leq \phi_{\text{edge}}(\theta_e) \leq 1 \,. \quad Eq.8$$

Further, in case each pixel's output value is a binary 0 or 1, merely subtracting the area covered by an image I within D_e^- from that within D_e^+ in an appropriately-scaled map of the visual field represents the value of the input sum θ_e.

2.5 Vertex detection (angle detection)

Now that we have a basic design for the edge-detection e cells, how would a vertex- or angle-detection cell v work? A relatively simple design for this can be attempted by combining edge-detection cells. Imagine two e-cells e and e' with overlapping RFs, with the midlines of the RFs meeting at an angle at an intersection point on their circumferences as shown in Fig. 10. Again, this is for simplicity. In a more realistic model, the relative placement of cells need not be exact; after all, we have been allowing for some latitude in the relative amounts of excitatory and inhibitory input required to activate the edge-detection cells, and some tolerance in the location and relative orientations of a pair of these cells does reflect the natural variation in biology (in fact, in living organisms the circular regions and straight midlines will be "warped"). Now, set-theoretically, an angle in the Euclidean plane is defined as the set of points between two rays emanating from a common point, its vertex. We substitute pixels for mathematical points and define the set of points as the intersection of the sets of excitatory inputs, $D_e^+ \cap D_{e'}^+$, for two edge detectors e and e' whose orientations represent the directions of the two rays as shown in Fig. 10. The regions D_e^+ and $D_{e'}^+$ within D_e and $D_{e'}$ are shown in the Figure as overlapping, corresponding to the set intersection. This is the configuration of RFs we wish for vertex detection and suggests the form of the neural architecture required. The midlines of D_e and $D_{e'}$ enclose the overlap region, $D_e^+ \cap D_{e'}^+$. They also enclose the larger region enclosed by extending them out to infinity on the right (shown as shaded in Fig. 10). This larger region is an angle having the vertex v at its apex. Because the midlines are edges of the pixel region $D_e^+ \cap D_{e'}^+$ and are aligned with the edges of the angle, the activation of v by activity within the pixel region $D_e^+ \cap D_{e'}^+$ is at the core of our vertex-detection scheme. Because of this correspondence of vertex detection with the activation of pixels within the angle containing the set $D_e^+ \cap D_{e'}^+$, we shall hence forth refer to vertex detection as angle detection.

An image I that perfectly fits within the region $D_e^+ \cap D_{e'}^+$, with nothing appearing in the intersections $D_e^+ \cap D_{e'}^-$ and $D_e^- \cap D_{e'}^+$, would register as the angle through the simultaneous activation of e and e'. Suppose that a v-cell that is based upon this combination of the two e-cells takes its input from them rather than directly from the pixels in its RF. The resulting configuration with the connections from the pixels in D_e and $D_{e'}$ to e and e' and from there to v is shown in Fig. 11. With this design we intend that if e and e'

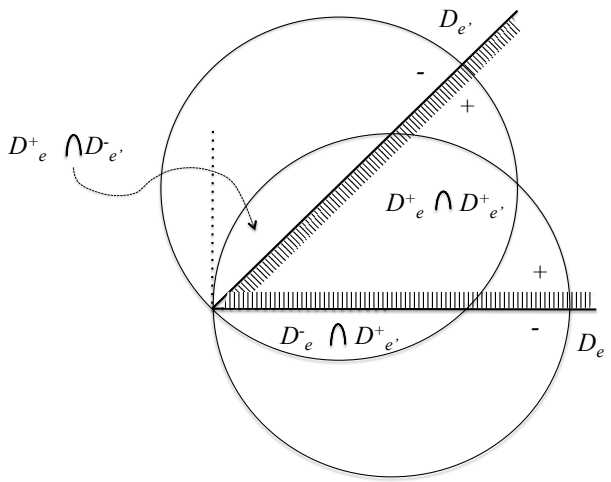


Figure 10:

are both sufficiently active, the angle depicted in Fig. 10 has been detected at the shaded region, otherwise not. The input sum for v in terms of e and e' is

$$\theta_v = w_e \phi_{\text{edge}}(\theta_e) + w_{e'} \phi_{\text{edge}}(\theta_{e'}) \; . \quad Eq.9$$

Here, w_e and $w_{e'}$ are unsigned weight magnitudes. As with the edge-detectors, we shall assume that all angle-detectors have the same signal function, producing signals $\phi_{\text{angle}}(\theta)$ based upon their edge-detector inputs.

The variation in image shapes to which the e-cells respond in signaling edge detection affects the responses of the v-cells in angle detection. Continuing with the assumption of rectified linear cells, where $\phi_{\text{angle}}(\theta) = [\theta - \theta_T]_+$, if $\theta_T = 0$ (an assumption already made in a previous section) then an angle-detector cell (hereafter called a v-cell) simply adds the inputs from its two e-cells as in Eq. 9. This allows any positive weighted signal, whether from e, e' or both, to activate v with the sum of magnitudes of the weighted signals. This is clearly inadequate for angle detection, for a balance of pixel activity in favor of either D^+_e alone or $D^+_{e'}$ alone, activating either e or e' with a positive signal, could be mistaken for an angle. The so-called "angle detector" v could not distinguish between an angle and either of its edges because either e-cell alone can activate v. Therefore, to avoid false positives ϕ_{angle} must either be nonlinear or thresholded-linear with a threshold $\theta_{T,v} > 0$. Further, the threshold value must be large enough to require that the v-cell output will be nonzero only if *both* of the e-cell inputs to v are nonzero.

For simplicity, suppose as with the e-cells that the v-cell's input sources (the two e-cells) have equal weights, with $w_e = 1/2 = w_{e'}$ in Eq. 9. Then,

$$\theta_v = \frac{1}{2} \phi_{\text{edge}}(\theta_e) + \frac{1}{2} \phi_{\text{edge}}(\theta_{e'}) \; . \quad Eq.10$$

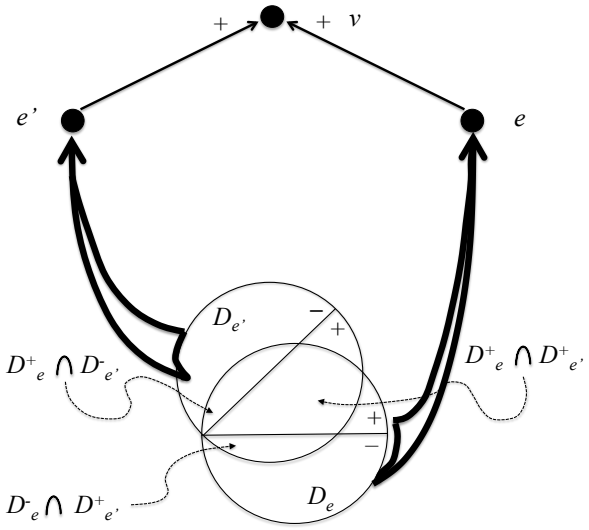


Figure 11:

An angle detection requires that $\theta_v > \theta_{T,v}$, which in turn requires that

$$\frac{1}{2}\left(\phi_{\text{edge}}(\theta_e) + \phi_{\text{edge}}(\theta_{e'})\right) = \frac{1}{2}\phi_{\text{edge}}(\theta_e) + \frac{1}{2}\phi_{\text{edge}}(\theta_{e'}) > \theta_{T,v},$$

or

$$\phi_{\text{edge}}(\theta_e) + \phi_{\text{edge}}(\theta_{e'}) > 2\theta_{T,v}. \quad Eq.11$$

But as previously shown the e-cell outputs are bounded above by unity, with $\phi_{\text{edge}}(\theta_e), \phi_{\text{edge}}(\theta_{e'}) \leq 1$. To avoid having v activated by either of its e-cells acting alone, we must have it that if for example $\phi_{\text{edge}}(\theta_e) = 1$ while $\phi_{\text{edge}}(\theta_{e'}) = 0$, then $\theta_v \leq \theta_{T,v}$, in which case $\phi_{\text{angle}}(\theta_v) = 0$. From Eq. 11, this implies $1 + 0 \leq 2\theta_{T,v}$, or

$$\theta_{T,v} \geq \frac{1}{2} = \theta_{T,v,\text{min}} .$$

With its threshold value in this range, activating v will require that

$$\phi_{\text{edge}}(\theta_e) + \phi_{\text{edge}}(\theta_{e'}) > 1. \quad Eq.12$$

To provide a baseline for the analysis in the following sections, we shall make the simplest assumption for all signal functions ϕ and use the recified-linear form in Eq. 7. In effect, this has the property of linearity $\phi(\theta) = \theta$ where $\theta \geq 0$, but in a sense is nonlinear because of the threshold $\theta_T = 0$.

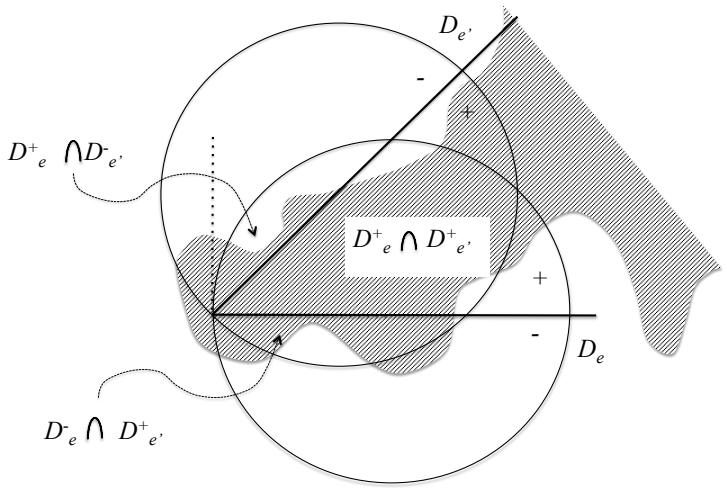


Figure 12:

2.6 Testing the angle detector

Let us examine some cases in depth. Using Eq. 7, Eq. 10 can be rewritten

$$\theta_v = \frac{1}{2}[\theta_e]_+ + \frac{1}{2}[\theta_{e'}]_+ . \quad Eq.13$$

Then the maximum input to the v-cell is $\theta_{v,\max} = \frac{1}{2}(\theta_{e,\max} + \theta_{e',\max}) = \frac{1}{2}(1+1) = 1$. From the assumption that all signal functions have the form of Eq. 7, the angle-detector signal function is $\phi_{\text{angle}}(\theta) = [\theta - \theta_{T,v}]_+$ and therefore the maximum value that is output by a v-cell is

$$\phi_{\text{angle,max}} = \theta_{v,\max} - \theta_{T,v,\min} = 1 - \frac{1}{2} = \frac{1}{2}. \quad Eq.14$$

A nonlinear angle-detector cell with a threshold level and input weights designed to prevent its activation by an input from a single edge-detector does not solve all problems, for there remains a great deal of variation in image boundaries. The constraints imposed so far ensure only that the two edge-detectors representing the two adjacent sides of the angle must be active; they do not ensure this activity represents an angle shape very closely. For example, suppose that D^+_e and $D^+_{e'}$ are both 2/3 covered by an image (in Fig. 10 they overlap considerably and, hence, may well have similar coverage), where $\Sigma_{i \in D^+_e} s_i = \Sigma_{i \in D^+_{e'}} s_i = \frac{2}{3}$. Suppose also that D^-_e and $D^-_{e'}$ each has slightly less than 1/6 coverage,

where $\Sigma_{i\in D_e^-} s_i = \Sigma_{i\in D_{e'}^-} s_i = \frac{1}{6} - \varepsilon$, with ε small, $\varepsilon > 0$. Then with $\theta_{T,v} = 1/2$, both e-cells exceed the value necessary for v to exceed its threshold since $\theta_e = \theta_{e'} = \frac{2}{3} - \frac{1}{6} + \varepsilon = \frac{1}{2} + \varepsilon > \frac{1}{2}$, yielding

$$\theta_v = \frac{1}{2}([\theta_e]_+ + [\theta_{e'}]_+) = \frac{1}{2}(\frac{1}{2} + \varepsilon + \frac{1}{2} + \varepsilon) = \frac{1}{2} + \varepsilon$$

and therefore (assuming $\theta_{T,v} = \frac{1}{2}$)

$$\phi_{\text{angle}}(\theta_v) = [\theta_v - \frac{1}{2}]_+ = [\frac{1}{2} + \varepsilon - \frac{1}{2}]_+ = \varepsilon > 0\,.$$

Yet, the shape is not a good approximation to an angle. In fact, a variety of shapes can activate both e and e' under these assumptions, and most of them are arguably poor approximations to an angle as illustrated in Fig. 12.

The problem is this: Enforcing the shape control sufficient to ensure a reasonable approximation to the angle that v is supposed to represent requires that $\theta_{T,v}$ have a high value, and the question becomes one of just how high this must be to force the image shape to be a reasonable approximation to the specified angle. Alternatively, ϕ_{edge} could have a high nonzero threshold, or the form of the signal function ϕ_{edge} could be changed to a different kind of nonlinearity than the simple rectified linear form. However, either of these two alternatives would constitute trying to force the e-cells to solve the angle-detection problem at the potential cost of their more general edge-detection function. The root of the problem is that the simple neural network design so far proposed has no explicit mechanism for shape control; it is difficult to force the putative angle-detector to identify an angle without responding also to many other shapes. Perhaps too much is being expected of ϕ_{edge}, ϕ_{angle} and the parameters $\theta_{T,e}, \theta_{T,v}, w_e,$ and $w_{e'}$.

3 Semantic analysis applied to neural network design

The angle-detector problem turns out to be a shape control problem at a very elementary level of image processing, where simple feature shapes must be represented with enough precision to ensure their efficacy as components of more complex representations. This takes us back to the semantics of the neural network, pursued through our set-theoretic analysis.

3.1 A disjoint union

Notice that the first consideration encountered in angle detection in Section 3 was the need to enforce the requirement that the edge-detector cells e and e' must be activated simultaneously. The second consideration was that simply working with thresholds and

nonlinearities, while it might address the simultaneity requirement, does not ensure shape control. The problem is that connecting e and e' to v in effect simply combines the separate influences of the two sets D_e and $D_{e'}$, with each set acting independently upon v. This ignores the fact that the sets may have a nonempty intersection $D_e \cap D_{e'}$ in which each pixel plays a dual role in its influence upon v since it acts separately through e and e'. The influence of each element of D_e and $D_{e'}$ depends upon whether it is a member of D_e^+ or D_e^- in the case of D_e, and whether it is a member of $D_{e'}^+$ or $D_{e'}^-$ in the case of $D_{e'}$. When considering a set of pixels in its relationship to a cell to which its elements have feedforward connections, we call it an "influence set". For example, each pixel influences cell v twice positively if it lies in $D_e^+ \cap D_{e'}^+$. This is because it is an element of D_e^+ and thereby has an *excitatory* effect upon e which relays its activity to v via the excitatory connection with weight w_e, and it is an element of $D_{e'}^+$ and thereby also has an excitatory effect on the activation of e' which relays its activity to v via the excitatory connection with weight $w_{e'}$. This means that if the weights w_e and $w_{e'}$ are equal the excitatory influence of the pixel upon v is effectively doubled. However, if it lies in the region described as $D_e^+ \cap D_{e'}^-$ its influence is nil. This is because it is an element of D_e^+ and thereby has an excitatory effect on the activation of e which relays its activity to v via the excitatory connection with weight w_e but also it is an element of $D_{e'}^-$ and thereby has an *inhibitory* effect on the activation of e' which relays its inhibitory activity to v via the excitatory connection with weight $w_{e'}$. This means that if the weights w_e and $w_{e'}$ are equal the influence of the pixel upon v is effectively cancelled to zero. A similar statement applies to a pixel that is a member of $D_e^- \cap D_{e'}^+$. We can summarize this in the set-theoretic model by stating that this architecture combines the two sets D_e and $D_{e'}$ in a *disjoint union* $D_e + D_{e'}$ (see Fig. 13) because it allows each pixel in the intersection $D_e \cap D_{e'}$ to play two independent roles.

Set-theoretically, each element of a disjoint union of two sets is considered to be an element of one of them but not both. In effect, the elements of the intersection $D_e \cap D_{e'}$ of the two sets appear twice, once for each set. The usual way to express this mathematically is to incorporate each element p of D_e and $D_{e'}$ in either one or two ordered pairs, as follows: If $p \in D_e$, it is represented by the ordered pair $(p, 1)$ in $D_e + D_{e'}$; if $p \in D_{e'}$, it is represented by $(p, 2)$. Thus if $p \in D_e \cap D_{e'}$ it is represented by not one but *two* ordered pairs $(p, 1)$ and $(p, 2)$, and otherwise only by *one* ordered pair, either $(p, 1)$ or $(p, 2)$. Mathematically, there are *injections* of the two sets into their disjoint union. These are functions $i_e : D_e \longrightarrow D_e + D_{e'}$ and $i_{e'} : D_{e'} \longrightarrow D_e + D_{e'}$ such that elements $p \in D_e$ and $p' \in D_{e'}$ have images $i_e(p) = (p, 1)$ and $i_{e'}(p') = (p', 2)$. It is easily seen that i_e and $i_{e'}$ are one-to-one correspondences. Formalizing the combining of sets of pixels through the connections from D_e and $D_{e'}$ to v in the neural network as a disjoint union is an expression of the actual semantics of our putative angle detector.[2]

Describing the angle-detection architecture as a disjoint union $D_e + D_{e'}$ is a way of expressing the influence an arbitrary pixel—an element of either D_e or $D_{e'}$—has on the

[2] As we shall see, category theory offers a full expression of the semantics of a neural network.

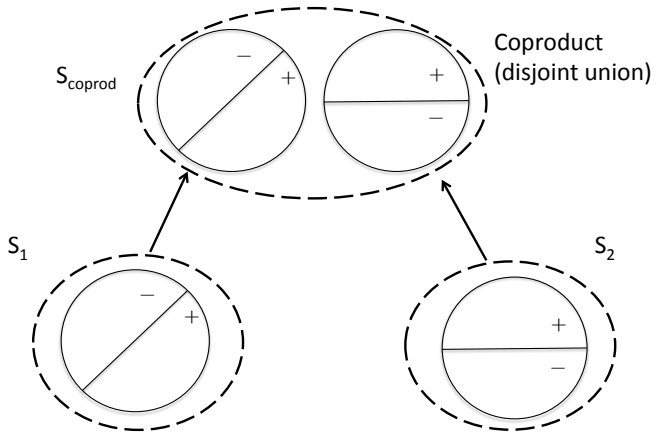


Figure 13: **A disjoint union of the edge-detector receptive fields.**

v-cell. If it lies in the intersection region $D_e^+ \cap D_{e'}^+$ its influence is excitatory and thereby provides evidence for the presence of an angle in an image, but is being counted twice and is thereby magnified. On the other hand, if a pixel lies in $D_e^+ \cap D_{e'}^-$ or $D_e^- \cap D_{e'}^+$ its influence is cancelled out, yet as indicated in Fig. 10 it would have provided an invaluable contribution to the evidence against an angle's presence in an image. Notice that the linearity of ϕ_{edge} is not the problem; similar statements about the *effects* of the intersections would be true if ϕ_{edge} were nonlinear. And as previously suggested, forcing the form of ϕ_{edge} to help in shape control for angle detection is an added burden upon its role in edge detection. The gist of this discussion is that $D_e + D_{e'}$ does not express the *intended* semantics of angle detection with the design in Fig. 11, but in an incomplete yet revealing way it does express something about what the design *actually does*. It expresses the fact that the inputs from the pixel field are all treated separately, and a pixel with two connection pathways to v has its action either magnified or ignored.

3.2 Reconsidering the connectionist structure

Perhaps the problem lies in simply combining the two edge detector inputs to the v-cell. We have seen that this results in a disjoint union of their sets of input pixels, which overuses the positive evidence for an angle in $D_e^+ \cap D_{e'}^+$ and cancels the negative evidence in $D_e^+ \cap D_{e'}^-$ and $D_e^- \cap D_{e'}^+$. Let us start over and simplify our analysis by first assuming that ϕ_{edge} and ϕ_{angle} are rectified linear functions with thresholds of zero. Our purpose is simply to explore the effects upon the activation of e, e' and v of alterations we shall make to the

connectionist structure of the angle detector network. The hope is that by reformulating the inputs to the v-cell, we can find a neural structure that distinguishes precisely between angles and non-angles. We can then apply a threshold value for it below which it shuts down and above which it becomes active, signaling the presence of an angle.

We first express the quantities θ_e and $\theta_{e'}$ in Eq. 12 in greater detail by expanding the summation quantities in terms of the subregions $D_e^+ \cap D_{e'}^+, D_e^+ \cap D_{e'}^-, D_e^+ \cap D_{e'}^0, D_e^- \cap D_{e'}^+, D_e^- \cap D_{e'}^0, D_e^- \cap D_{e'}^-, D_e^0 \cap D_{e'}^+$ and $D_e^0 \cap D_{e'}^-$. The notation for subregions is as in figures 10 and 12 but with the addition of D_e^0 and $D_{e'}^0$, which are the regions complementary to D_e and $D_{e'}$, respectively, within $D_e \cup D_{e'}$. Using the linear signal function form $\phi(\theta) = \theta$ and assuming as before that there is no bias in favor of either e or e' so that $w_e = w_{e'}$, Eq. 12 becomes

$$\begin{aligned}
\theta_v &= w_e(\theta_e + \theta_{e'}) \\
&= w_e[(\Sigma_{j\in D_e^+\cap D_{e'}^+} w_j s_j + \Sigma_{j\in D_e^+\cap D_{e'}^-} w_j s_j + \Sigma_{j\in D_e^+\cap D_{e'}^0} w_j s_j \\
&\quad -\Sigma_{j\in D_e^-\cap D_{e'}^+} w_j s_j - \Sigma_{j\in D_e^-\cap D_{e'}^0} w_j s_j - \Sigma_{j\in D_e^-\cap D_{e'}^-} w_j s_j) \\
&\quad +(\Sigma_{j\in D_e^+\cap D_{e'}^+} w_j s_j + \Sigma_{j\in D_e^-\cap D_{e'}^+} w_j s_j + \Sigma_{j\in D_e^0\cap D_{e'}^+} w_j s_j \\
&\quad -\Sigma_{j\in D_e^+\cap D_{e'}^-} w_j s_j - \Sigma_{j\in D_e^0\cap D_{e'}^-} w_j s_j - \Sigma_{j\in D_e^-\cap D_{e'}^-} w_j s_j)],
\end{aligned}$$

where the s_j are the current activities of the pixels and the w_j are the weights, normalized for each region D_e^+ or D_e^- according to the equation $\Sigma_{j\in D_e^+} w_j = 1 = \Sigma_{j\in D_e^-} w_j$ from Section 3.1. Just to simplify expressions, assume equal numbers of cells $\text{card}(D_e^+)$ and $\text{card}(D_e^-)$ in D_e^+ and D_e^-; then we can use a single weight value $w_j = \frac{1}{\text{card}(D_e^+)} = \frac{1}{\text{card}(D_e^-)}$. Now, grouping like summation terms, cancelling like terms with opposite signs, and noting that $D_e^- \cap D_{e'}^- = \emptyset$ for angles of size not greater than 180^o,

$$\begin{aligned}
\theta_v &= w_e(2\cdot\Sigma_{j\in D_e^+\cap D_{e'}^+} w_j s_j + \Sigma_{j\in D_e^+\cap D_{e'}^0} w_j s_j + \Sigma_{j\in D_e^0\cap D_{e'}^+} w_j s_j \\
&\quad -\Sigma_{j\in D_e^-\cap D_{e'}^0} w_j s_j - \Sigma_{j\in D_e^0\cap D_{e'}^-} w_j s_j). \qquad Eq.15
\end{aligned}$$

We can achieve greater clarity by a change in notation: In a weighted summation with weights w_j and activities s_j of the pixels represented by the elements of a set X, let us substitute the expression ΣX for $\Sigma_{j\in X} w_j s_j$. With this simplification, we can rewrite Eq. 15 as

$$\theta_v = w_e(2\cdot\Sigma D_e^+ \cap D_{e'}^+ + \Sigma D_e^+ \cap D_{e'}^0 + \Sigma D_e^0 \cap D_{e'}^+ - \Sigma D_e^- \cap D_{e'}^0 - \Sigma D_e^0 \cap D_{e'}^-). \; Eq.16$$

We need to remember that the pixel weights w_j and activities s_j are implicit in these summation terms. Eq. 16 expresses in detail that in the disjoint union the effects of the pixels in $D_e^+ \cap D_{e'}^+$ are doubled and the effects of those in $D_e^+ \cap D_{e'}^-$ and $D_e^- \cap D_{e'}^+$ are cancelled out.

Now consider the following change in the neural network design. A new type of cell called a u-cell is shown in Fig. 14 in a layer intermediate between the pixel layer and the

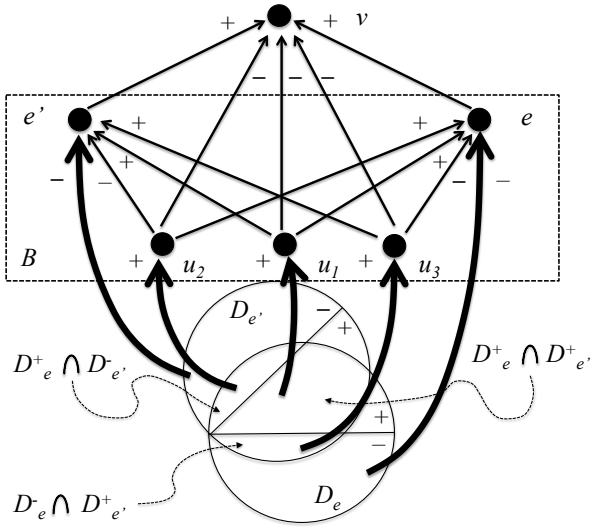


Figure 14: **A modified architecture for angle detection (vertex detection).**

e-cell layer. Each of the u-cells u_1, u_2, u_3 has an RF covering a smaller region of the pixel layer and receives excitatory inputs from the pixels in its RF. The RF for u_1 in terms of the RFs $D^+_e, D^-_{e'}, \ldots$ is $D_{u_1} = D^+_e \cap D^+_{e'}$. For the others we have $D_{u_2} = D^+_e \cap D^-_{e'}$ and $D_{u_3} = D^-_e \cap D^+_{e'}$. The labels for the sets in Fig. 15 have been simplified to, for example, u_1 as opposed to D_{u_1}. This figure will be used to introduce the notion of a *colimit* in the nect section.

The three u-cell RFs together form the total intersection $D_e \cap D_{e'}$ of the RFs D_e and $D_{e'}$ (Fig. 14). The purpose of the new cells is to represent the overlap in the RFs D_e and $D_{e'}$ by supplying an additional input to v, correcting for the lack of information about the overlap in the original architecture. As pointed out in Section 4.1, the original architecture, uninformed about the overlap, represents the disjoint union of the two RFs. The u-cells correct for this by supplying inhibitory inputs to v, accounting for the intersection by subtracting from the input summation. The activity of v is now discounted for any active pixels in $D^+_e \cap D^-_{e'}$ and $D^-_e \cap D^+_{e'}$, and the positive effect of the excitatory inputs from $D^+_e \cap D^+_{e'}$ is represented once instead of twice. This increases the sensitivity of the angle detector to shape distortions outside the region $D^+_e \cap D^+_{e'}$ by (1) allowing any shape distortions occurring in $D^+_e \cap D^-_{e'}$ and $D^-_e \cap D^+_{e'}$ to lower the activity of v and (2) lowering the overall activity of v by discounting for one copy of its positive input from $D^+_e \cap D^+_{e'}$. The angle detector remains sensitive to shape distortions from inactive pixels within $D^+_e \cap D^+_{e'}$ because there remains a net of one excitatory input per pixel from that region, but now each inactive pixel has a greater relative effect in lowering the activation level of v. This opens up the possibility that, now, v can have a relatively *low* threshold and yet require a net excitatory input from *both* e-cells, as before, in order to become active. The effect of

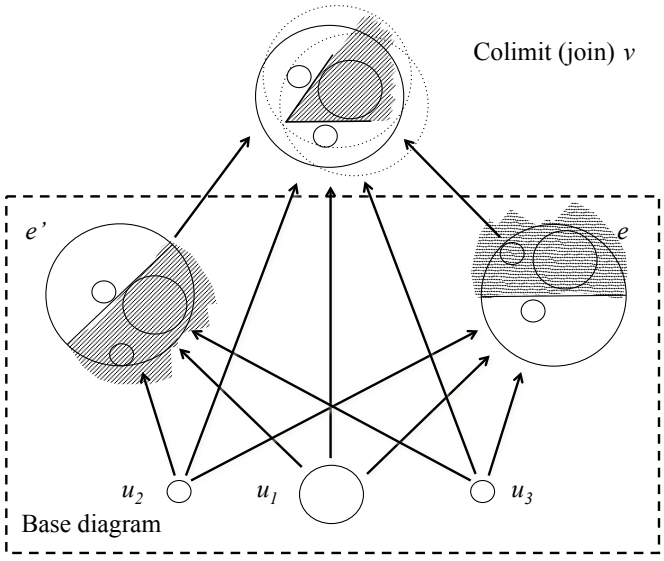


Figure 15: **A colimit for a base diagram of sets and functions with an angle as colimit object.**

the new architecture upon cell v in angle detection is illustrated in Fig. 15.

As with the e-cells, it seems reasonable also to let the u-cells have a uniform weight value,

$$w_u = w_{u_i}\ (i = 1, 3)\ .Eq.17$$

Using the u-cell modifications to the architecture, Eq. 16 is replaced by

$$\begin{aligned}
\theta_v &= w_e(\theta_e + \theta_{e'}) - w_u(\Sigma D_e^+ \cap D_{e'}^+ + \Sigma D_e^+ \cap D_{e'}^- + \Sigma D_e^- \cap D_{e'}^+) \\
&= w_e(2 \cdot \Sigma D_e^+ \cap D_{e'}^+ + \Sigma D_e^+ \cap D_{e'}^0 + \Sigma D_e^0 \cap D_{e'}^+ - \Sigma D_e^- \cap D_{e'}^0 - \Sigma D_e^0 \cap D_{e'}^-) \\
&\quad - w_u(\Sigma D_e^+ \cap D_{e'}^+ + \Sigma D_e^+ \cap D_{e'}^- + \Sigma D_e^- \cap D_{e'}^+) \\
&= (2w_e - w_u)\Sigma D_e^+ \cap D_{e'}^+ + w_e(\Sigma D_e^+ \cap D_{e'}^0 + \Sigma D_e^0 \cap D_{e'}^+ - \Sigma D_e^- \cap D_{e'}^0 - \Sigma D_e^0 \cap D_{e'}^-) \\
&\quad - w_u(\Sigma D_e^+ \cap D_{e'}^- + \Sigma D_e^- \cap D_{e'}^+) \\
&= (2w_e - w_u)\Sigma D_e^+ \cap D_{e'}^+ + w_e(\Sigma D_e^+ \cap D_{e'}^0 + \Sigma D_e^0 \cap D_{e'}^+ - \Sigma D_e^- \cap D_{e'}^0 - \Sigma D_e^0 \cap D_{e'}^-) \\
&\quad - (w_u(\Sigma D_e^+ \cap D_{e'}^- + \Sigma D_e^- \cap D_{e'}^+)\ .\ Eq.18
\end{aligned}$$

Notice that step 2 in the derivation shown in Eq. 18 exploits the consolidation of terms in $\theta_e + \theta_{e'}$ already performed in deriving Eq. 16. For example, $D_e^+ \cap D_{e'}^-$ and $D_e^- \cap D_{e'}^+$ have been cancelled out as noted before. Now, however, the inhibitory inputs to v from u_2 and u_3 provide a negative summation term for each of these regions.

3.3 A colimit architecture

As we did with the disjoint union architecture illustrated in figs. 11 and 13, let us examine the semantics of the new architecture illustrated in Fig. 14 using our partial formalization via sets, as shown in Fig. 15. The shaded regions indicate the excitatory inputs to each e-cell when it acts alone as an edge detector, and the net excitatory input to the angle-detector colimit v-cell when both e and e' are acting upon v along with the inhibitory input from the u-cells. The set formalization being only partial is indicated by the fact that the shading is superimposed upon the sets to represent the net excitatory nature of the inputs to each cell, a feature not expressed by the sets themselves. The illustration with the shading added provides a conceptual, semantic view of the functionality of the architecture illustrated in Fig. 14.

Indeed, Fig. 15 can be interpreted as a diagram in *category theory* that defines a *colimit*. The sets and the functions which map pixels in one set to pixels in another are *objects* and arrows, or *morphisms*, in a *category*. These terms will be defined in detail in a future section. They are illustrated by the circular disks and the arrows joining them, which are *inclusion functions*. The inclusions define $D_{u_1}, D_{u_2}, D_{u_3}$ as subsets of D_e and $D_{e'}$ and $D_{u_1}, D_{u_2}, D_{u_3}, D_e$ and $D_{e'}$ as subsets of D_v. An inclusion is simply a one-to-one mapping of the pixels, that is the elements, of the set at the base of the arrow to themselves as pixels in the set at the head of the arrow. This is a special case of an injection function, another case of which was discussed previously for the disjoint union. An inclusion is a way of expressing a subset relationship $A \subseteq B$ as a function mapping a subset A to itself as an image in the superset B. The images of $D_{u_1}, D_{u_2}, D_{u_3}$ are shown as smaller disks inside the disks representing D_e and $D_{e'}$. The disks for the latter two sets are shown in their union D_v as lightly-dashed circles, with the smaller disks for $D_{u_1}, D_{u_2}, D_{u_3}$ inside their intersection. A separate disk for the angle-detector v is superimposed over the union of the dashed disks to emphasize that the result is an RF for a single angle-detector cell, with its shaded "arrowhead" region illustrating the intersection of the excitatory RF regions of the e-cells.

The entire figure, containing the *defining diagram* of the colimit, shows a cone-like structure attached to the RFs inside the dashed-rectangle box, the *base diagram* of the colimit. The cone-like structure, called a *cocone*, consists of the v-cell RF at the heads of five arrows together with the arrows themselves and the sets at the base of each arrow. There is one such arrow for each of $D_{u_1}, D_{u_2}, D_{u_3}, D_e$ and $D_{e'}$. The cocone is attached to the structure consisting of the sets and functions inside the dashed-rectangle box. Separately, the latter sets together with the six arrows in the base diagram map $D_{u_1}, D_{u_2}, D_{u_3}$ into D_e and $D_{e'}$. This diagram can be thought of as a *specification* for constructing the "Colimit (join) v". In the construction, the union of D_e and $D_{e'}$ is properly expressed by blending them so that they are merged where they share the images of the u-cell RFs that represent the three parts of the intersection $D_e \cap D_{e'}$. But why can we claim that D_e and $D_{e'}$ are "blended" by this colimit architecture?

The claim of the "blending" is justified by the following property of the defining diagram, a property known as *commutativity*: Choosing any one of D_{u_1}, D_{u_2} or D_{u_3} and following all three pathways from that set through the mapping arrows to D_v results in one and the same composite function regardless of the pathway followed; that is, the three mappings given by tracing the three different pathways are equivalent. One of the three pathways consists of a single arrow mapping the u-cell's pixels directly into D_v. The other two pathways map each of the u-cell's pixels through D_e and $D_{e'}$ and on into D_v. In doing so, the base diagram arrows based at the u-cell are composed with arrows from the cone-like structure. The three pathways, two of them forming the function compositions, yield one and the same function, the same function represented by the single arrow directly from the u-cell to v. This states mathematically that each u-cell intersection region is represented only once in the union of D_e and $D_{e'}$, for the commutativity guarantees that all pathways from any D_{u_i} to D_v ($i \in \{1, 2, 3\}$) correspond to a single inclusion.

3.4 The colimit as a local-distributed concept representation

The colimit is a categorical structure that is used in concept analysis in the CNST. By combining concept representations (discussed so far in terms of pixel sets) to form more complex concepts, the colimit can be seen as a representation that is both local and distributed. First, the sets and functions in the defining diagram in Fig. 15 suggest an interconnected array—a diagram—of neural cells in the architecture. The interconnections are links in pathways from the "blending cells" of the diagram, through the cells being "blended", to the colimit cell at the apex of the neural diagram. Thus, the corresponding, blended concept representation is distributed within the neural network. The colimit cell's representation is a concept that blends the concepts represented by the "blended" cells based upon the concepts represented by the "blending" cells. Thus, the colimit cell amalgamates the separate representations of the other cells in the defining diagram and therefore the concept representation is local—provided there is a sense in which the neural structure commutes.

A future section will show how a neural architecture can itself be formalized as a category and, hence, how it is correct to say that the interconnected colimit array of cells is a commutative diagram in correspondence with the concept commutative diagram. As the discussion continues further, the formalization of concept representation in a neural network will become complete.

We have just introduced three important notions that will be explored further as we complete the semantic formalization. First, the equivalence of pathways through the arrows in a diagram from one set to another, with at least one of the pathways involving the composition of two or more arrows meeting at intermediate sets along the way, is a property of *commutative diagrams*. Second, the commutativity of the defining diagrams arises because the arrows in the cocone are directed out of the base diagram, and attaching the cone-like structure to the base diagram achieves commutative diagrams within the

composite diagram (one commutative diagram for each of the three u-cells with its three pathways leading to the v-cell). Third, there is one other property which, if it holds, defines the cocone to be a *colimit cocone*: the property of *initiality*, which for colimits states essentially that the colimit is the simplest blended combination of the objects in its base diagram; the colimit object contains no extraneous information. It is when commutativity and initiality both hold that the composite diagram is called the defining diagram for a colimit . Finally, it is worth emphasizing that the colimit structure is a form of local-distributed concept representation for connectionist networks.

An additional property of the colimit suggests the importance of feedback in a neural network. Because it is a distributed concept representation, it is important that all cells representing the defining diagram are active whenever the colimit cell is active. In particular, this must be the case if the colimit cell is activated by inputs from elsewhere in the network. For example, a subnetwork processing auditory stimuli from a microphone could, based upon past adaptation, have formed strong connections representing an expectation of detecting a visual object corresponding to a particular sound. Perhaps more commonly, the colimit cell may increase its activation in concert with feedback to its base diagram which increases the activation of those cells in a sort of "bootstrapping" mode. Each of these occurences can be seen as an instance of the network "parsing" its colimit concept representation with the aid of feedback through connections which reciprocate its feedforward connections.

3.5 Testing the modified architecture

We wish to compare the results for angle detection using the colimit architecture with the results from the disjoint sets architecture. As usual, we shall assume that all cells have the same linear signal function, so that input sums are translated directly into cell activation values. Section 3.6 explains how a simulation of the architecture was performed and presents the results. But first, let us examine some cases by simply specifying a distribution of activities over the terms in Eq. 17.

First let us reconsider the distorted-angle example in which D_e^+ and $D_{e'}^+$ each is 2/3 covered by an image while D_e^- and $D_{e'}^-$ each has a coverage of $\frac{1}{6} - \varepsilon$, with ε small, $\varepsilon > 0$. Since the e-cells, and hence their copies of the intersection regions, have weight $w_e = \frac{1}{2}$, it seems reasonable also to let the u-cell weight value be

$$w_u = w_{u_i} = \frac{1}{2} \; (i = 1, 3) \; .Eq.19$$

Using the weight values from Eqs. 11 and 16, the v-cell input sum for the new architecture is

$$\begin{aligned} \theta_v \;\; = \;\; & (2w_e - w_u)\,\Sigma D_e^+ \cap D_{e'}^+ + w_e\,(\Sigma D_e^+ \cap D_{e'}^0 + \Sigma D_e^0 \cap D_{e'}^+ - \Sigma D_e^- \cap D_{e'}^0 - \Sigma D_e^0 \cap D_{e'}^-) \\ & - w_u\,(\Sigma D_e^+ \cap D_{e'}^- + \Sigma D_e^- \cap D_{e'}^+) \end{aligned}$$

$$
\begin{aligned}
&= (2\cdot\frac{1}{2}-\frac{1}{2})(\frac{1}{2}+\varepsilon)+\frac{1}{2}\cdot(0+0-0-0)-\frac{1}{2}\cdot\{(\frac{1}{6}-\varepsilon)+(\frac{1}{6}-\varepsilon)\}\\
&= (\frac{1}{2})\cdot(\frac{1}{2}+\varepsilon)-\frac{1}{2}\cdot 2\cdot(\frac{1}{6}-\varepsilon)\\
&= \frac{1}{12}+\frac{3}{2}\varepsilon\,.
\end{aligned}
$$

With the threshold for v used previously, the resulting activation would be

$$\theta_{\text{angle}}(\theta_v) = [\frac{1}{12}+\frac{3}{2}\varepsilon-\frac{1}{2}]_+ = [-\frac{5}{12}+\frac{3}{2}\varepsilon]_+ = 0\,.$$

Recall that with the threshold value of $\theta_{T,v}=\frac{1}{2}$, the v-cell in the disjoint sets architecture attains a positive (albeit small) activity value for the distorted image illustrated in Fig. 12. In fact, with a threshold value of $\theta_{T,v}=\frac{1}{12}+\frac{3}{2}\varepsilon$ its activation would be even larger. By contrast, the colimit architecture with *either* threshold value is blocked from activating v with an image as distorted as this.

Now let us see what would happen with a near-perfect fit to the angle represented by v. Again using the small positive number ε to represent imperfection (although in fact none of the numbers used here is exact), suppose that $\theta_e=\theta_{e'}=\frac{2}{3}$, this time due to the total activation $\frac{1}{2}+\varepsilon$ over the region $D_e^+\cap D_{e'}^+$ and $\frac{1}{6}-\varepsilon$ each over the regions $D_e^+\cap D_{e'}^0$ and $D_e^0\cap D_{e'}^+$. Then

$$
\begin{aligned}
\theta_v &= (2w_e-w_u)\Sigma D_e^+\cap D_{e'}^+ + w_e(\Sigma D_e^+\cap D_{e'}^0+\Sigma D_e^0\cap D_{e'}^+-\Sigma D_e^-\cap D_{e'}^0-\Sigma D_e^0\cap D_{e'}^-)\\
&\quad -w_u(\Sigma D_e^+\cap D_{e'}^-+\Sigma D_e^-\cap D_{e'}^+)\\
&= (2\cdot\frac{1}{2}-\frac{1}{2})(\frac{1}{2}+\varepsilon)+\frac{1}{2}\cdot\{(\frac{1}{6}-\varepsilon)+(\frac{1}{6}-\varepsilon)-0-0\}-\frac{1}{2}\cdot(0+0)\\
&= (\frac{1}{2})\cdot(\frac{1}{2}+\varepsilon)+\frac{1}{2}\cdot 2\cdot(\frac{1}{6}-\varepsilon)\\
&= \frac{5}{12}-\frac{\varepsilon}{2}\,.
\end{aligned}
$$

Therefore, considering just these two cases, the v-cell for the colimit architecture could be an effective angle detector if it possessed a threshold value in the range $\frac{1}{12}+\frac{3}{2}\varepsilon\le\theta_T<\frac{5}{12}-\frac{\varepsilon}{2}$.

3.6 The simulation results for the angle detector

Edge detection via contrast

To quantify the detection properties of edge cells, we created a numerical simulation of a contrast cell responding to a variety of visual images in the focal plane. As stated in Section 2, we assumed that the focal plan of the agent's eye is flat and covered by a rectangular grid of photoreceptor neurons (or pixels). The cell has a rectified linear response

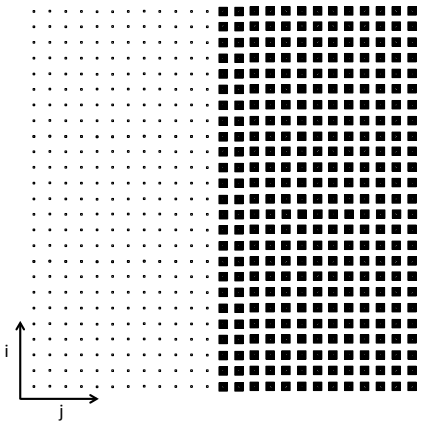


Figure 16: **A grid of photoreceptor neurons where the size of the pixel indicates the magnitude of response. Small points represent zero magnitude while full black squares represent a maximum magnitude of one. A vertical edge of an object, oriented at 90^o to the horizontal, is shown.**

up to a maximum intensity of unity at which it saturates. Fig. 16 shows the pixel grid in a close up view of a region of the focal plane centered on an edge of an object.

The output of an edge-detector or e-cell is given by the weighted input sum and signal function of eqs. 3 and 4, with weights normalized, a threshold value of zero and the linear function constant set to unity as described in Section 2.4. But recall that in Section 3.2 we introduced u-cells, a layer of cells intermediate between the focal plane photodetectors, or pixels, and other cells which would otherwise have received inputs from the pixel layer. The u-cells receive only excitatory inputs from their receptive fields (RFs) in the pixel layer, and they have output connections to "higher-up" cells which are either excitatory or inhibitory as the situation demands. In the following, then, when we refer to a positive or negative region in a cell's RF, we mean in actuality the excitatory or inhibitory weights in the connections emanating from the u-cells in that region to the cell in question. As explained in sections 6.5, this use of u-cells is crucial for the neural network's ability to represent adjacency in the visual space. That having been said, much of the following discussion will refer to cell input summations in which the pixels themselves in an RF region are represented as opposed to the u-cells that cover that region. This is purely a computational simplification that does not affect the results of the simulations.

For the numerical simulations, we must assign specific receptive field pixel locations to the sets D_e^+ and D_e^-, the positive and negative regions which together make up the RF of an e-cell. Fig. 17 shows examples of circular RF excitatory and inhibitory weight patterns for edge cells that are tuned for maximum output when edges in an image are ori-

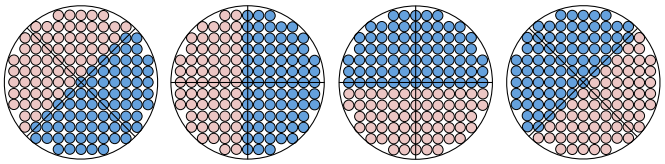


Figure 17: **Examples of receptive fields (RFs) 13 pixels in diameter for edge cells tuned for several edge orientations. Blue circles represent excitatory input weights while pink ones represent inhibitory weights. In this example, each RF contains approximately 75 excitatory and 62 inhibitory pixel weights totaling 137. In all cases, excitatory and inhibitory weight RF subfields are independently normalized to +/-1 respectively.**

ented at 45^{o}, 90^{o}, 180^{o}, and 225^{o}, respectively, in the left-to-right direction in the figure. Because the excitatory and inhibitory weighted u-cell inputs in the two halves of each receptive field are balanced by the normalization to have equal influence, the diameter of the receptive field sets the spatial scale at which the neuron reaches full sensitivity. Image features smaller than the RF diameter will produce decreased activation of the edge cell, as will features that cover most or all of the RF.

To avoid any confusion, the following note about contrast versus edge orientation in the edge cell RFs may be helpful. The orientation of directed contrast in each RF is that of increasing net activation of its edge cell by moving a hypothetical image feature from coverage of mostly the negative region to coverage of mostly the positive region based upon the terminology "contrast cell". This is the direction for which orientation angles are commonly assigned to these cells. However, we have been and will continue to assign orientation based upon the contrast boundary, which defines the edge and is perpendicular to the contrast direction. The assignment of angles to various edge cell RFs can be deduced by referring to Fig. 17 and the accompanying discussion in the previous paragraph.

A two-dimensional "tuning surface" for a cell quantifies its output response over a range of stimuli. To calculate the tuning surface, the center of its RF is first located to match the image feature it detects. The image feature is then systematically translated and rotated relative to the RF center and the input sums calculated each translation and rotation sampled. Fig. 18 shows examples of an image edge at different rotations and translations

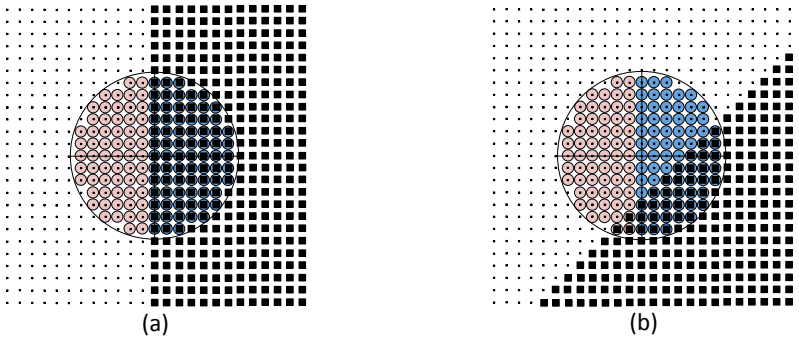


Figure 18: **Examples of edge stimuli used in computing a tuning surface for a** 90^{o} **edge cell. a) An image object with an edge aligned with the contrast boundary, or edge, of the cell's RF. This image object, placed to cause maximal activation of the cell, is at zero translation and rotation. b) The same object translated by** $+4$ **pixels and rotated** -45^{o}**.**

with respect to the center of a 90^{o} edge cells RF. Fig. 19 shows the tuning surface for this cell calculated over a grid of relative image translations and rotations. Note that for this plot, the RF's diameter has been defined as 101 pixels to create a smoother response surface than is possible with smaller diameters.

We can see from Fig. 19 that the edge cell's response has a sharp peak at zero pixels of translation and zero degrees of rotation. Were we to substitute the Heaviside output function of Eq. 4 for the rectified linear function used throughout this paper, the discrimination of the canonical edge could be controlled by adjusting the threshold up from zero, $b > 0$ using the Heaviside parameter b. For reasonable threshold values, this cell will provide an unambiguous signal for the presence of a contrast edge at a particular location in the agent's field of view. Because of the earlier assumption that the connection weight distributions of an edge-detector cell's positive and negative input fields are normalized, the cell's input sum θ will always be $\theta \leq 1$, so the parameter a might as well be set to unity, $a = 1$.

In addition to considering tuning surfaces generated by translating and rotating an image object, it is also instructive to study the effect on an edge cell's response due to varying levels of image degradation. There are many ways to model this, but in the following simulations we will treat it as pixel noise. In the discussion so far, we have considered images with binary valued pixels, whose output is either 0 or 1. A simple noise model for this type of image randomly toggles a pixel's value biased on a notional biased coin flip. The bias of the coin will be referred to as the "noise level". For a noise level of zero the value

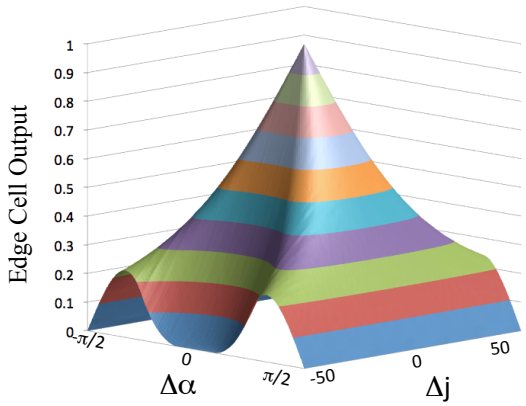


Figure 19: **The tuning surface for a 90^o edge cell with RF diameter equal to 101 pixels. The units of translation of the image object are pixels and the units of rotation are radians. A rectified linear activation function was used in these calculations. For this study, the photoreceptor grid consisted of an array of 400x400 pixels.**

of a pixel is that generated by a noise-free image, while a noise level of 0.5 toggles each pixel of the same image either from 0 to 1 or from 1 to 0 with a 50 % probability. Fig. 20 shows examples of a single noise trial with noise levels 0.1 and 0.5 applied to the pixel grid shown in Fig. 16.

To calculate a noise level tuning curve for an edge-detector cell, we perform a simplified form of a Monte Carlo simulation. The RF of a 90^o edge-detector cell is first stimulated to its maximum response on an uncorrupted contrast edge also oriented at 90^o. Successive points on the noise tuning curve are found by calculating the edge cells output when the image is corrupted with increasing level of noise. At each noise level, the average cell output and standard deviation are computed over 1000 trials with the same noise level, but with a new distribution of corrupted pixels specified by a pseudo-random number generator for each trial. Fig. 21 shows the resulting tuning curve with the noise level varied from 0.0 to 0.5 in increments of 0.05. The Figure shows a linear reduction in average output as a function of noise level, implying the error in edge localization is linear in noise level. Therefore as above, if we substitute a Heaviside output function the discrimination of the canonical edge may be controlled by adjusting a simple threshold. Based upon the results made visable in both the tuning surface of Fig. 19 and the noise tuning curve of Fig. 21, we can have confidence that when an edge cell becomes activated, a contrast edge is most likely positioned appropriately or nearly so in its receptive field.

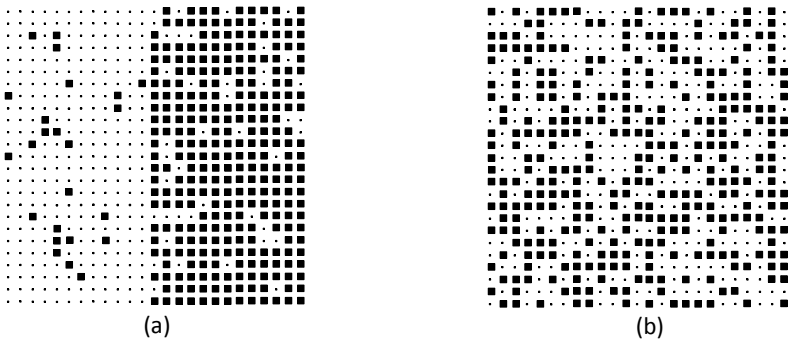


Figure 20: **Two examples of the application of the biased coin noise model to an image contrast boundary. a) noise level = 0.1, b) noise level = 0.5**

Angle detection via coproduct and colimit neurons

Paralleling the analysis for edge-detector cells, we performed simulations to quantify the discrimination properties of angle-detector cell models. Both the coproduct and colimit models were evaluated to compare their performance. A sufficient performance improvement in the colimit versus coproduct type of cell would be evidence toward verifying our claim that the full-on colimit is a significant improvement in angle detection over the coproduct. The colimit has a diagram with blending objects that specify how the parts of an object fit together whereas the coproduct, while also a colimit, has no blending objects and therefore no scheme for specifying how the parts of an object fit together. To perform the evaluation, we performed simulations in which each type of angle-detector cell was stimulated with a variety of visual images. The baseline stimulus was an image of an angle at one vertex of a triangle with a subtended angle of 45^{o} angle, illustrated in Fig. 22.

Recall that an angle-detector cell has attendant cells and connections representing a colimit defining diagram with the following components. In Section 3 the diagrams contain pixel sets and functions, and in the neural network these are represented by cells and positively- or negatively-weighted connections. To simplify the discussion, in discussing the simulations we will describe the cells and connections as if they themselves are the diagram objects and arrows. In the next section we will correct for this inaccuracy by describing a formal colimit model for the neural network that corresponds to the formal colimit model for the pixel sets and functions.

In the base diagram for the colimit are two edge-detector cells e' and e with RFs tuned for contrasts at 45^{o} and 180^{o}, respectively. The two RFs with their positive and negative regions highlighted (positive shown in blue, negative in pink) are shown separately in

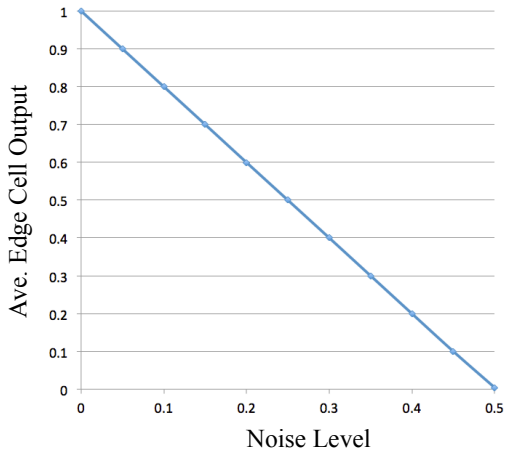


Figure 21: **Monte Carlo simulated noise level tuning curve for a 90^o edge-detector cell. The average over 1000 trials for each noise level is plotted at each noise level simulated. The standard deviation at each level is less than the line width of the curve and, hence, is not shown. A 400x400 pixel grid was used for this study and the edge-detector cell's RF diameter was 101 pixels. Points on the tuning curve were calculated using a rectified linear activation function.**

Fig. 23. In actuality, the two RFs overlap, or intersect, on the pixel grid as shown in Fig. 24 where a 45^o angle is to be detected should one occur in an image. The other components in the base diagram are theoretically the u-cells covering the intersection of the e-cell RFs and their connections to the e-cells, positive or negative depending upon which half of an e-cell's RF a u-cell covers. As mentioned earlier, in the simulations the pixels themselves substitute for the the u-cells. Finally, the defining diagram of the colimit is a cocone attached to the base diagram. The cocone has at its apex a v-cell, an angle-detector cell, and it also contains excitatory connections from every cell in the base diagram to the v-cell. The formal neural network model of the next section will explain how the commutative property of a neural diagram is defined.

Notice in Fig. 24 that the intersection of positive regions forms an "arrowhead", the desired angle (colored in dark red). The pixels (substituting in the simulations for u-cells) in this region have positive weights in their connections to both e-cells, making their inputs to the e-cells excitatory. This contribution to the v-cell's input summation is conveyed to it by the excitatory input connections from each e-cell. The net effect of this is to double the excitatory contribution to the v-cell's input coming from the arrowhead region. On the other hand, the two "lobes" (in green) are the intersection of the positive half of one e-cell's RF and the negative half of the other. Thus, the lobes' inputs to the

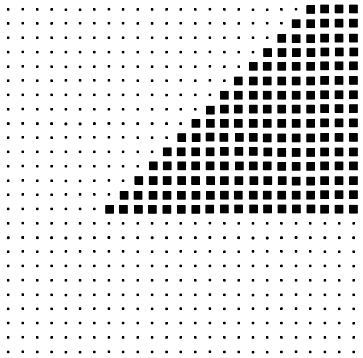


Figure 22: **A pixel grid of photoreceptors responding to one vertex of a polygon. The diagonal edge-detector is oriented at 45^{o} while the horizontal edge-detector is oriented at 180^{o}. The angle is defined as the pixel field lying between the two edges (black pixels). The vertex is located at the grid coordinates of the apex. The photoreceptor grid was 400x400 pixels.**

v-cell, conveyed through the excitatory e-cell-to-v-cell connections, provide a net input of zero. The other regions are disjoint from the rest, and, hence, their inputs conveyed through the e-cells, whether positive or negative, are exclusive. As was discussed in Section 3, the effect of this is that the v-cell recieves two copies of the excitation from the "arrowhead", zero from the "lobes" due to the cancellations, and a single copy from everywhere else in the two e-cells' RFs. Thus, the coproduct cell, which has no u-cells as "blending objects" in its defining diagram, has unbalanced inputs from the "arrowhead" and "lobes", where the former is overrepresented and the latter are not represented at all in their effects on the coproduct v-cell. On the other hand, the colimit v-cell receives an extra copy of inhibitory input from its "blending object" u-cells. This has the effect of cancelling one of the excitatory copies of the "arrowhead" and supplying a net inhibitory input from the "lobes". This enforces a sort of balance among the different RF regions of the colimit v-cell in that it requires a greater balance of excitatory over inhibitory active pixels than is required by the coproduct cell to reach the same level of activity.

The activation of the coproduct and colimit angle-detector cells was computed using Eq. 16 and Eq 18, respectively. Rectified linear activation functions were used for the cells e', e, v. A two dimensional translation tuning curve was produced for both the coproduct and colimit cells. The conventions used for the two-dimensional image translations along with examples is shown in Fig. 25. The image was not rotated in this study. The tuning

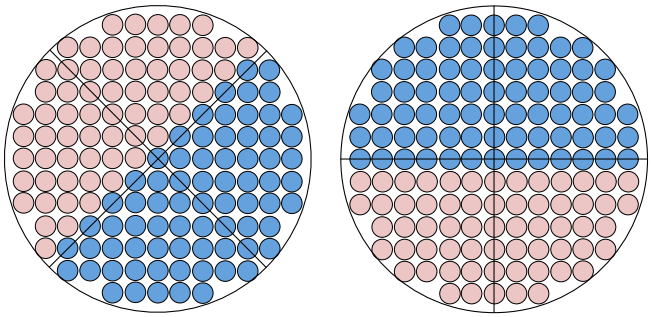


Figure 23: **The receptive fields for edge-detector cells e' and e. In the simulations, the RF diameter is 101 pixels for both cells.**

surfaces for both the coproduct and the colimit v-cells is shown in Fig. 26. To enable a comparison of the two cells' performance free of the bias of the individual cells' output distributions, the output for each cell has been normalized to unity at zero translation.

The first thing to observe when comparing the two surfaces in Fig. 26 is the qualitative differences in their shapes. Because the composite RF of the coproduct cell has some areas that over-emphasize the contributions of excitatory stimuli and other areas that cancel to zero the contributions of mixed excitatory and inhibitory stimuli, the resulting tuning surface is unevenly distributed over a large range of translation parameters. By contrast, the colimit cell with its "blending cell" inputs has corrected the unbalanced contribution effects, making the curve more symmetrical and with activation values in the region of its peak concentrated in a smaller range of translations. This confers upon the colimit call a sharper discriminatory ability. In fact, the most striking difference is the shape of the peaks near their maxima. The coproduct cell peak is noticeably broader than that of the colimit cell. Therefore, if we substitute a Heaviside output function for the rectified linear output function of each cell, the colimit cell will better localize the vertex for a given threshold value.

As with the edge-detector cells, it is also instructive to study the effect on an angle-detector cell's response due to varying levels of image degradation. Using the same noise model described above, a Monte Carlo simulation was performed to calculate noise level tuning curves for both types of vertex cells. At each noise level, the average cell output and standard deviation were computed over 1000 trials. The noise level was varied from

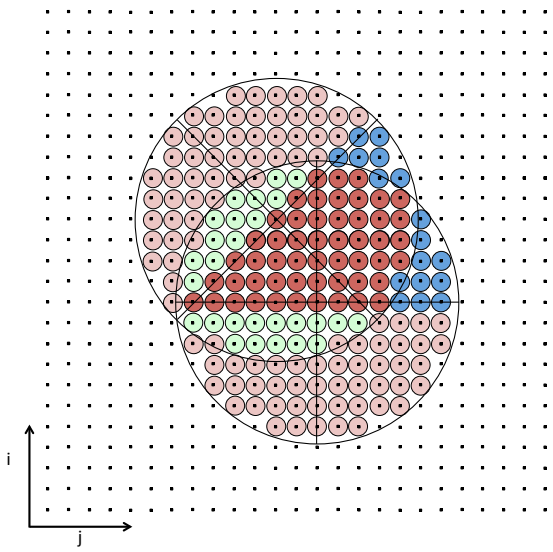


Figure 24: **An illustration of the overlapping receptive fields of edge-detector cells e' and e. The outputs of these cells are combined to form the input for the angle cell. Non-overlapping excitatory and inhibitory weights are blue and pink respectively. The region containing only overlapping excitatory weights to form the "arrowhead" is in red while the regions containing overlapping excitatory and inhibitory weights to form the "lobes" are light green. The RF diameter for each edge-detector cell is 101 pixels.**

0.0 to 0.5 in increments of 0.05. The results are shown in Fig. 27.

Comparing the two tuning curves, first notice that the colimit curve approaches a zero v-cell output level much earlier than the coproduct curve. For example, the colimit curve ordinate is just slightly in excess of 0.2 at a noise level of 2.5, whereas the coproduct curve ordinate reaches 0.2 at a noise level of 4.5, with both curves reaching 0 at a noise level of 5.0. As in the translational tuning curves, this difference is due to the differences in the input structure for the two cells. The second difference, and perhaps more significantly, can be seen in the slopes of the two curves as they approach the ordinate 0 at the zero noise level of 5.0: The colimit drops in value much faster than the coproduct. Therefore, as before, if we substitute a Heaviside output function for the rectified linear output function of each v-cell, the colimit cell will better discriminate between an angle and noise for a given threshold value. That is, it will reject the presence of the angle it represents at a much lower level of noise than will the coproduct cell, thus reducing false positive detections.

In summary, this result supports our claim that colimit angle-detector cells have generally better discrimination properties than coproduct cells. When a colimit cell becomes active, we can be more confident that the angle the cell represents is present in the visual

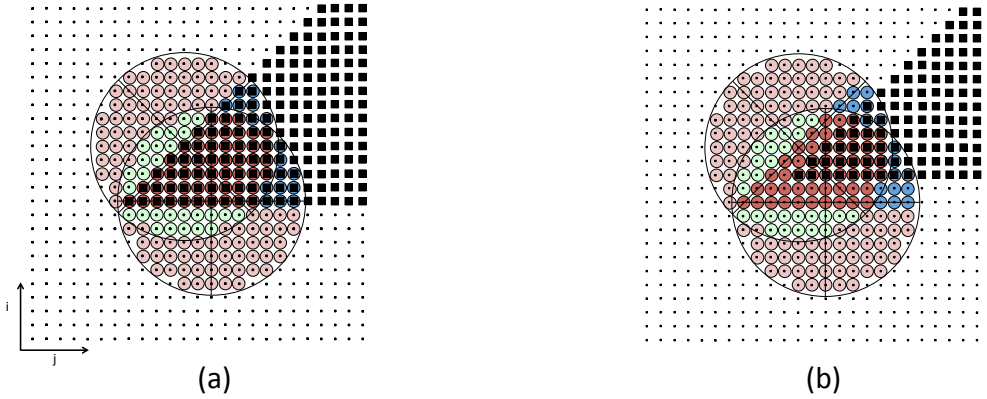


Figure 25: **Two of the translations of the image used in computing both the coproduct and colimit angle-detector cells' translation tuning surface. Tranlations in the vertical (i) and horizontal (j) coordinate directions were sampled in the simulations. The positive sense of the translations is both up and to the right. The translations are expressed in units of pixels. (a) The location of the image designed to give maximum angle-detector cell output, locating the baseline for translations $\Delta i, \Delta j$ as $\Delta i = 0, \Delta j = 0$. (b) The translation $\Delta i = 2, \Delta j = 4$. As usual, the edge cell RF diameters is 101 pixels and the photoreceptor grid is 400x400 pixels.**

field of Agent R. Moreover, this performance improvement has been obtained without the need for nonlinearities or high threshold values applied to either edge-detector cells or angle-detector cells.

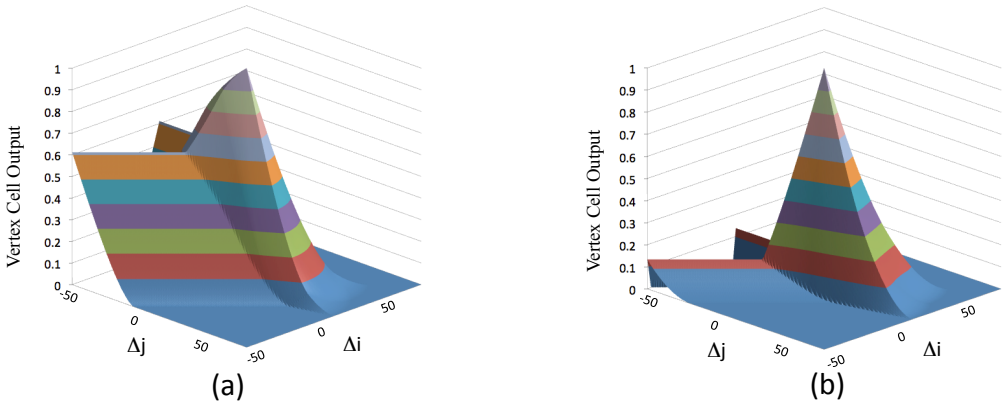


Figure 26: **The results for calculating the translation tuning curve for a range of stimulus translational parameters, where a) is for the coproduct neuron and b) is for the colimit neuron. The output for each neuron has been normalized to unity at zero translation to ease comparison**

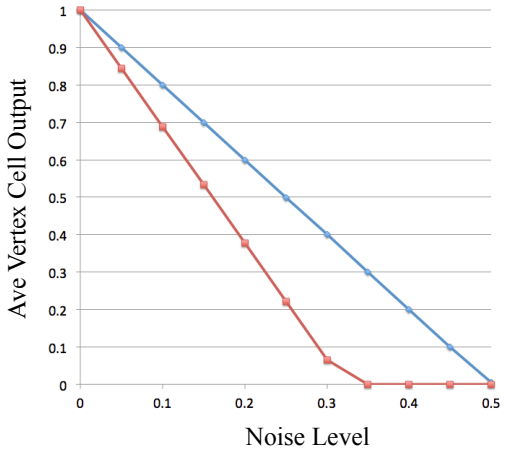


Figure 27: **Monte Carlo-simulated noise level tuning curves for the coproduct (blue) and colimit (red) angle-detector cells. The average over 1000 trials for each noise level is plotted for each curve. As with the edge-detector cells, the standard deviations are not shown because they were less than the line width in each case. The output for each neuron has been normalized to unity at the zero noise level to support an unbiased comparison.**

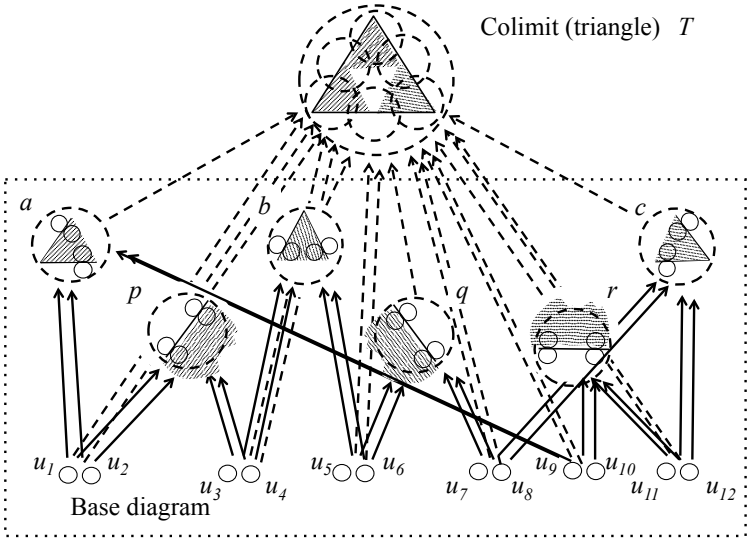


Figure 28: **A triangle as a colimit formed by blending angles along common edges.**

3.7 Triangle detection

The method of putting things together described in the previous sections can be applied to the construction of more complex concept representations in a neural network. In this way, previously-constructed objects are blended together to form a many-tiered, hierarchical, local-distributed system of concept representations of increasing complexity.

Applying this idea to the triangle, u-cells can be recruited to blend together angles to make a triangle. As mentioned in Section 4.2 and illustrated in Fig. 10, an angle is defined set-theoretically as the set of points between two rays emanating from a common point. In fact, this set of points is the intersection of the sets of pixels in the excitatory regions of the two e-cells' RFs. In similar fashion, a triangle is defined set-theoretically as the intersection of three angles whose sides overlap in pairs, with the overlapped sides forming line segments joining the vertices. Correspondingly, we define a triangle representation as a colimit of angle representations, where the blending occurs along aligned edges forming the sides. A construction for this is shown in Fig. 28.

Notice that in addition to the three angles involved, three additional edge-detectors not used by the three angle-detectors are also involved in the colimit base diagram. Their RFs overlap the RFs of those used in defining the angles; they join the pairs of angles by matching their aligned edges, resulting in the triangle represented by the T-cell. The three angles are labelled $a, b,$ and c in Fig. 28 and the added edges are labelled $p, q,$ and r. In the base diagram, the blending of two angles and the edge aligned with them is specified using two pairs of u-cells. For example, in Fig. 28 the angles a and c are

joined along edge r, where a and r are blended along the intersection of their RFs. The intersection has two parts: a region whose pixels provide excitatory inputs to a and r and a region whose pixels provide inhibitory inputs to them. The excitatory and inhibitory regions provide their inputs through the blending u-cells u_9 and u_{10}, respectively. In similar fashion, r and c are blended along the intersection of their RFs, whose excitatory and inhibitory regions provide their inputs through the blending u-cells u_{11} and u_{12}. Recall that the pixel inputs to all u-cells are uniformly excitatory. Therefore, as with the colimit construction for the v-cell, the mapping of each u-cell's RF into its v- and e-cell RFs is accomplished via the neural network with excitatory weights for the connections from a u-cell representing an excitatory RF region and inhibitory weights for the connections from a u-cell representing an inhibitory RF region. On the other hand, the mapping of each blending u-cell's RF directly into the T-cell RF is accomplished with an inhibitory weight, as was the case for the v-cell colimits. Maintaining our use of linear signal functions for simplicity, the net effect of this is to subtract one copy of each blending u-cell RF in the total input summation of the T-cell. As with the angle colimits, this inhibition coming from the blending u-cells acts as an adaptive threshold making it difficult for the T-cell to become activated by only a part of its colimit base diagram.

The three angles and edges are thereby joined by blending to form the triangle object. Because the set intersection of the excitatory subsets of the six RFs of $a, b, c, p, q,$ and r defines the triangle, the colimit object is a formal representation of the triangle. A neural architecture for the triangle colimit consisting of angles blended along common edges is shown in Fig. 29. As with the angle colimit, the feedforward excitatory connections have excitatory reciprocals, making it possible for an active colimit T-cell to prime its diagram. As before, this priming allows the neural network to "parse" its inputs; in this case, the triangle can be understood as a composite of its priming-activated components, its angles and sides.

The set-theoretic illustration of Fig. 28 suggests the intended semantics of the triangle construction. But does the neural network really have this semantics? And are the u-cells really needed; could the v- and e-cells a, b, c and p, q, r have done this just as well? That is, could a disjoint union of the RFs of the latter cells represent the intended effect of a T-cell, to respond if and only if a triangle appears in the composite RF?

The subtraction of one copy of an excitatory intersection region via a blending u-cell from the T-cell input sum causes a net decline in the T-cell's activity. As it does with the v-cells, this overall decline in activity makes it more difficult for the T-cell to respond to any shape that appears in its RF in comparison with the disjoint union architecture, which has no blending u-cells. The net effect of the inputs to T is that there remains an excitatory region inside the triangle, but only a net inhibitory input to the T-cell outside the triangle, effectively "trimming off" the external region. Therefore, as with the v-cells, shape distortions arising from a non-triangle object or excessive imaging noise can be more readily dissociated from the correct triangle image without the necessity of a high value for the T-cell's threshold.

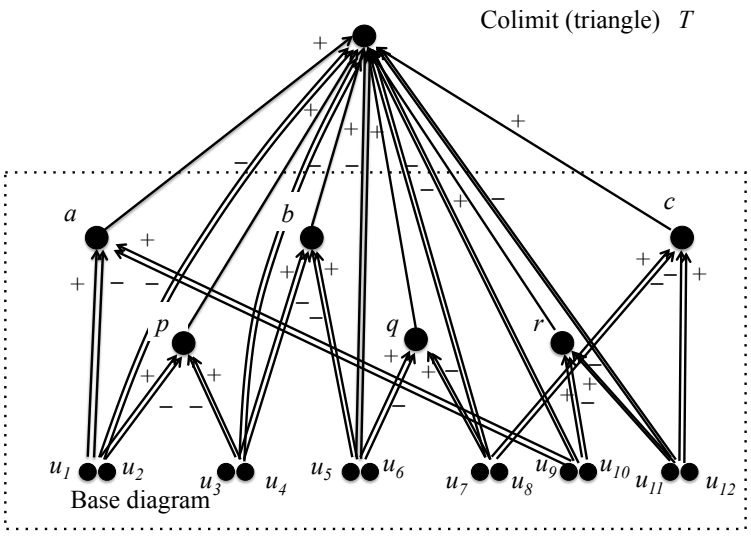


Figure 29: **A neural architecture for a triangle as a colimit of angles blended along common edges.**

The architecture for triangle detection for Agent R 1.2 uses angle-detector or vertex colimit cells along with edge-detector or contrast cells and blending u-cells in its base diagram. It is represented in a summary fashion in figs. 30 and 31.

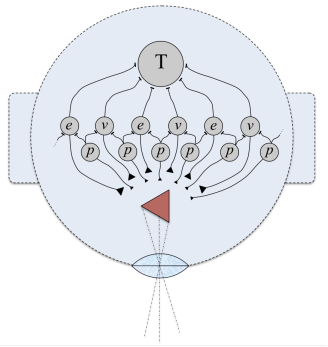


Figure 30: **Agent R1.2, illustrated with an icon for the triangle colimit, where the angle cells are labelled "v" (for "vertex"), the sides are labelled "e" (for "edge"), and the blending cells are labelled "p" (for "pasting").**

3.8 Limits: Positional Invariance as a Form of Abstraction

We have shown that colimits allow agent R1.2 to detect acute angles represented as a blended combination of edges and triangles as blended combinations of angles and edges.

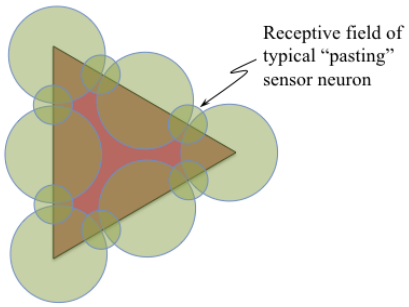


Figure 31:

This local-distributed form of representation can be generalized to ever more complex objects appearing in the stimulus input to the neural network. However, this depends entirely upon the activity in location-specific fields of pixels in the focal plane in Fig. 1. Many different colimit subnetworks are required to represent angles or triangles at the many possible locations in the visual field, to say nothing of changes in scale and orientation. Triangles, since they are composed of triples of angles joined by sides represented through the use of additional edge-detectors, present an even greater range of shapes, locations, scales and orientations. Agent R 1.2's brain must provide a separate base diagram and cocone combination for the colimit representation of every one of these possibilities. As objects can be yet more complex than triangles, the proliferation of colimit subnetworks can lead to an unwieldy and difficult-to-analyze neural network.

An objective in the design of agent R is that we want it to have a brain with a credible capability for *understanding* what it is seeing. Admittedly, R1.2 does have the ability to "parse" its stimulus input, representing angles and then triangles as colimits. However, each parsed object is an entity unrelated to others except where they share edges or angles. As it stands, the robot has no means of recognizing a deconstructed colimit object as being a member of a class of objects with a particular shape. Lacking some piece of additional information, the object's identity depends upon its location, to say nothing of scale and orientation. How are all the fixed-in-place colimit representations of a type of object to be related through the neural network in a way that convinces us that the network has a means of resolving its pure shape identity? R 1.2's dilemma is illustrated for several copies of the same-shaped triangle at different locations in the visual field in Fig. 32, where the triangles are all labelled as different "widgets" to emphasize the "knowledge gap".

This dilemma is often called "object invariance" or "invariance detection", that is, invariance with respect to location, scale, and/or orientation. It is a key issue in *image understanding*. More specifically, the example presented here involves the neural network's ability to express knowledge of plane geometry.

In previous work, we have introduced the category-theoretic notion of a *limit* and shown that it can provide the opposite function to that of the colimit construct: A limit can express abstraction as opposed to specialization. Specialization has been discussed: It is the process of deriving a more complex concept such as a triangle from simpler ones, such as angles and edges. The triangle is a more specialized concept, hence with fewer manifestations than have angles and edges, which appear in many other geometric shapes. We propose that specialization occurs through synaptic learning by forming colimits in a neural network. Abstraction is the opposite, deriving simpler concepts from many complex concepts in which the simpler ones are a common component by observing a concept in which many of the complex concepts appear. Again we propose synaptic learning for this process, but abstraction is much more difficult and applies in very special contexts; after all, it involves the complexity in bringing to activation many concepts containing the shared abstract component.

To understand limits, first recall that the colimit has a cocone adjoined to a base dia-

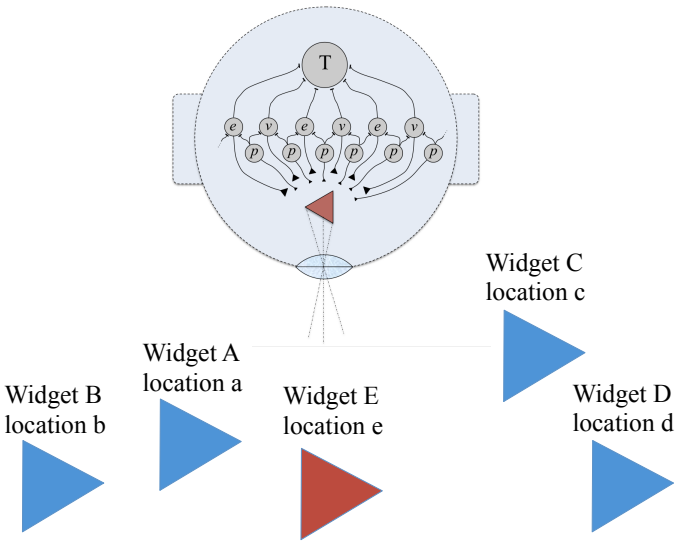


Figure 32: **Agent R1.2 can detect all triangles with its v-cells, but it has no way of identifying them as copies of a single type of object. Instead, to R1.2 they are just different widgets at different locations in visual space.**

gram, and the resulting defining diagram is commutative and has the intiality property so that the apex of the colimit cocone represents a more complex, more specialized concept containing more information than any of the concepts or their functional connectivity in the base diagram. Again there is a base diagram, but this one is a schematic showing how to express all of the information shared by the base diagram concepts when they are part of a more complex concept that incorporates them all. We regard this shared information as an abstraction that defines them as members of a class. It is obtained as the apex of a *cone* attached to the base diagram, making it commutative as does the colimit cocone attached to its base diagram. However, the cone has its arrows directed *into* its base diagram as opposed to the cocone, which has its arrows directed *out of* its base diagram. A cone arrow exists from the apical object to each of the objects in the base diagram; since the defining diagram formed by attaching the cone to the base diagram commutes, the apical object must map to the same part of each of these objects. As with the colimit, there can be many cones for the base diagram. A limit cone has the property of *terminality*, which effectively means its apical or limit object contains the most information of any cone for that base diagram.

For a simple example, let us restrict the discussion to the problem of location invariance for angles. Suppose R1.2 is to contain a representation an angle which appears in two different locations. In Fig. 33 a putative abstraction is shown with arrows pointing to it from two location-specific copies of an angle. The arrows indicate the abstraction process and also indicate the direction of a pair of neural network connections from the location-

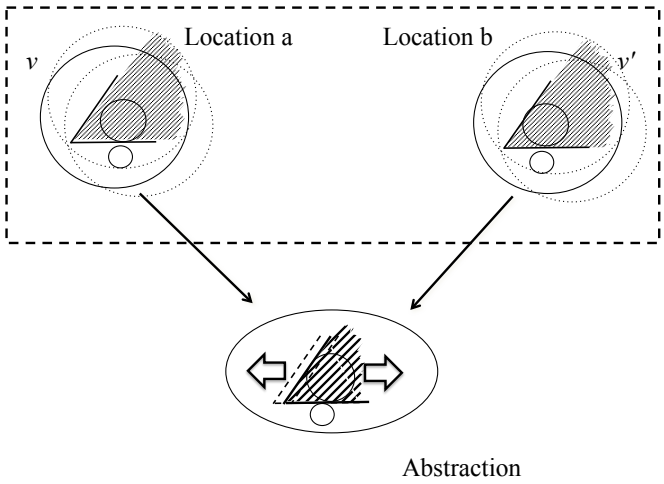


Figure 33: **An angle invariant as an abstraction from two location-specific angles.**

specific v-cells to an abstraction cell. As has been the case, we show the more abstract concept below the other two, the direction of specialization being shown as upward. Each connection back down, to the abstraction cell, must be strong enough that its v-cell source *alone* can activate the abstraction cell when it becomes active in response to the angle appearing in its RF. The objective in this is that the abstraction represent an angle free of location information. Given that, we can argue that it represents in pure form the concept of an angle.

The problem with this is twofold. First, this construct leaves out an essential component of the information about what is being abstracted, for there is nothing else in the network architecture representing the information that was to be shared by the two location-specific "widgets". Some indication of their common semantics in the context of an image must be present.

The second problem concerns the neural architecture itself. Suppose that there is partial activation of each v-cell due to non-angle active groups of pixels overlapping with their RFs. If their connections to the "abstraction" have sufficiently-strong connection weights so that either cell can activate the abstraction cell, how can we prevent the partial activation of both from having the same consequence even though no angle is present?

When faced with a similar problem in combining edges to form an angle, we tried setting an activation threshold θ_T for each v-cell. But it proved difficult to set an effective theshold for a simple disjoint combination of RFs. We solved this problem through the use of colimits, which allowed flexibility in setting a threshold that required both edge-detector cells to be active. The ensuing discussion pointed out that the colimit notion also

solved the problem of properly expressing the information that defined the "blending" of two edges to achieve an angle. Is there a similar solution for achieving abstractions instead of specializations?

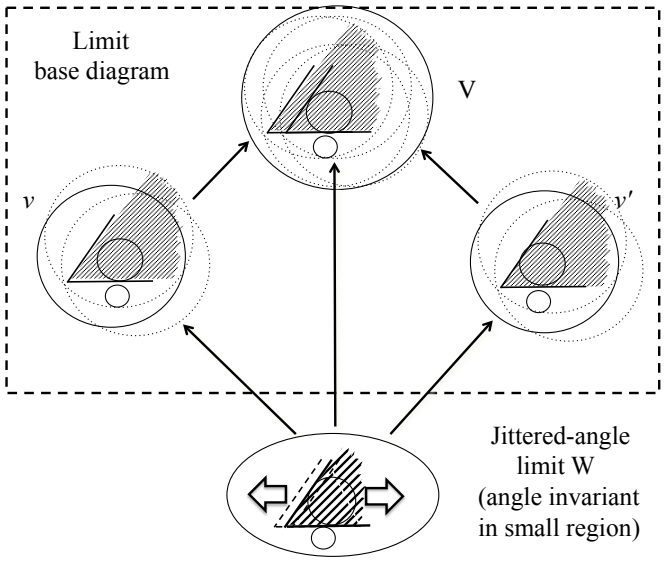


Figure 34: **A limit derived from the "jittered-angle" representation by "pulling back" an invariant. The invariant is an angle formed as a limit of a base diagram containing the fixed angles and the "jittered-angle" colimit together with the fixed-angle-to-colimit injection morphisms.**

Indeed there is. To derive a neural network representing an angle which is invariant with respect to location in the visual field, we proceed in stages. Consider again the example with an angle having the same shape at two closely-spaced locations. The base diagram for abstracting the angle contains influence sets representing two angle-detector cells v and v'. The limit construct illustrated in Fig. 34 uses three previously-formed colimit objects in its base diagram. Two of them are copies of an angle and the third is a superposition of the two that blends them together in a single image object as shown. For illustration purposes, we have labelled the angle invariant in Fig. 34 a "jittered-angle limit". One might envision the agent's eye jittering as it sees the image, causing the visual field to slide back and forth—or the angle changing its location slightly from side-to-side. The colimit construction for building this "jittered-angles colimit" is indicated in Fig. 35. The two base diagram arrows map the two fixed-location angle copies into a composite image containing a single copy of the angle. The composite is a union of sets of pixels with excitatory and inhibitory inputs to the two v-cells and contains the superposition of the two copies. As for the limit, corresponding elements in the intersections of the two angle RFs are mapped by the two base diagram arrows of Fig. 34 to a single element in the composite. But this is true only for the elements common to both v-cell RFs. All the other

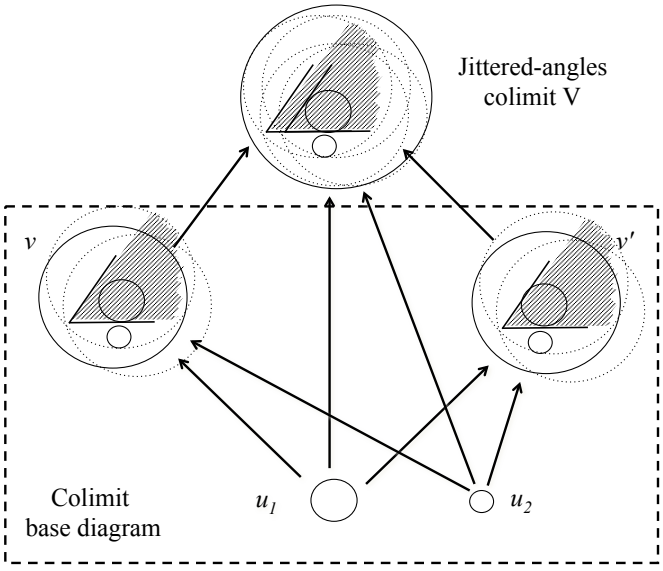


Figure 35: **A superimposed image formed by a vision sensor "jittering" over a small region. The neural network represents this as a colimit formed by blending angle representations along u-cells.**

elements of the v-cell influence sets are mapped separately, since they do not represent the same pixels. Finally, the base-diagram-plus-cone commutes: the compositions of the arrows along the two pathways through the angles and the direct arrow from limit object to the composite object are all the same function. That is, they map each element of the limit object to the same element of the composite.

The information retained in the limit concept is that which is shared between the two fixed-angle concepts based upon their joint use in the composite concept. The composite arises in the neural network as a consequence of persistent activity in one angle cell as the sensor "jitters over" to the other angle; the activation of both angle cells activates the composite cell in turn, bringing to activity the base diagram for the limit. During learning, as with the colimit, connection strengths adapt because of the activity, which now also includes the newly-recruited limit cell. The shared information represented by the limit cell includes the excitatory region and part of the inhibitory region of each angle cell's RF. This helps retain the angle shape independently except that the retained, shared pixels continue to express location information. The shape information and the range of occurrence of the shape are reinforced by the fact that when the abstraction is active each of the base diagram cells remains partially active, albeit the activity of the limit cell is dominant.

Augmenting our agent once more, this time with the capability to form the invariant angle representations as limits, results in Agent R1.3 (Fig. 36). A vision for the agent's

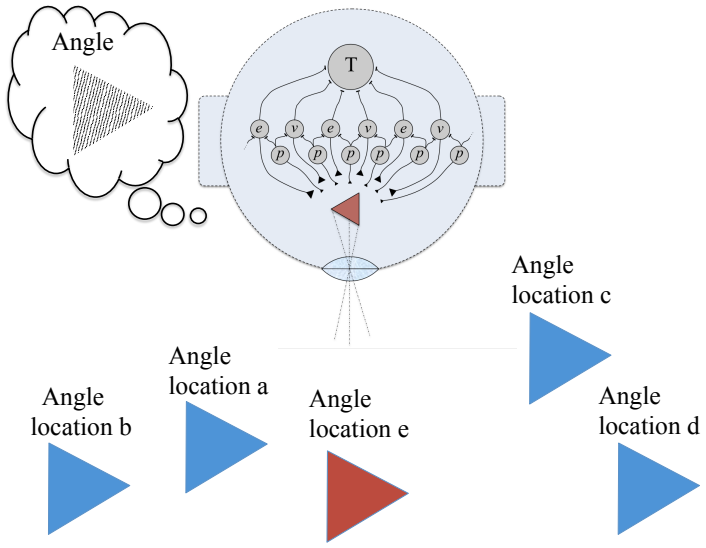


Figure 36: **Agent R1.3 can detect angles invariant to location in its visual field, and can thereby classify them as exemplars of the same shape. The incorporation of limits into the neural architecture makes this possible.**

neural network design, then, is that with limits in addition to colimits, it can go on to recognize and classify more shapes invariant to location in its visual field. A beginning at the complete formalization of colimits for our example will be provided in the final sections of this paper. The detailed discussion of limits will end with the example just presented.

3.9 Concept representations form a hierarchy

The final sections of this paper explain the use of category theory in completing the formalization of neural semantics that has been begun here using influence sets of pixels. The problem-solving analysis presented here suggests that there exists a hierarchy of representation in biological brains, and this will heretofore will be regarded as a hierarchy of *concepts*. The hierarchy progresses from simple representations of stimuli at the sensor level to complex representations farther along based in part upon the formation of colimits. This serves as a local-distributed knowledge system, formed adaptively, that expresses the sensory information represented by the structure and activity of a neural network. In the mammalian brain, particularly in humans, a region called the medial temporal lobe (MTL) evidently contains such a hierarchy [11, 14]. Beginning with the notion of colimits, we can proceed further into the theory and perhaps analyze the MTL hierarchy of representation. The analysis discussed here also points the way toward the design of artifical neural

networks capable of serving as artificial brains for autonomous robots.

4 Compositional Relations and Categories

The formalization presented in preceding sections shows that a representation of an angle can be obtained as a colimit of pixel sets. The colimit diagram explains an angle as a particular combining of edge-detector pixel sets. Further along, colimits combine angles and sides (strings of edges) to represent triangles. A colimit is defined mathematically by a diagram of sets and functions that commutes (and has another property not discussed yet). Commutativity means that the composition of the functions along all pathways from a set A to a set B in a diagram yield the same function. If the diagram defines a colimit, this property is the basis for the blending of pixel sets to properly represent combinations of shapes. This suggests that composition is a key aspect of functions in the hierarchy of sets.

The consequence of commutativity in the corresponding neural architecture is that there are terms in the input sum for a particular cell which bring about the blending of the representations carried by certain other cells. For example, an angle-detector cell will not work properly unless both edge-detector cells reach a sufficient level of activity to ensure that both edges are present simultaneously in the input image, defining the two rays enclosing the angle. As has been shown, adding the blending u-cell terms from the angle's comutative diagram to the input sum enforces this requirement.

Although the pixel-sets-and-functions model has proven useful in solving the neural network design problems addressed in preceding sections, it is only a partial formalization of the network's semantics. The colimit defining diagram of Fig. 15 introduced in Section 3.2 has served mainly as a support to intuition. It is useful in guiding the design of the neural architectures that determine each detector cell's response to stimuli. This utility stems from the fact that the compositions of its arrows along separate paths from a blending concept to the apical, colimit concept must commute, together with the fact that the commuting pathways show where connections are to go in formulating the input sums for the neural network cells. In the neural network, commutative diagrams correspond to connection paths that always fire simultaneously. However, the correspondence between commutative diagrams and the neural cell input sums that are based upon the correspondence between commutativity and the simultaneity of activity in the paths has not been fully formalized. A full formalization would not only provide a deeper understanding, but also might lead to new discoveries.

A mathematically-defined mapping of semantic expressions such as those illustrated in Fig. 15 to Agent R1.3's neural network, but with greater explanatory capability, is desired. The illustration in Fig. 37 suggests a mapping from a network of semantic expressions such as pixel sets and functions to a neural network. Is there a way to mathematically define a mapping that formalizes the transfer of the properties of the sets, functions, and

neuron input summations to the neural architecture? Can such a mapping ensure that commutative diagrams of sets and functions correctly correspond to the simultaneity of activation in neural structures? In order to answer such questions, it is necessary to first define what is meant by a commutative diagram in a neural network.

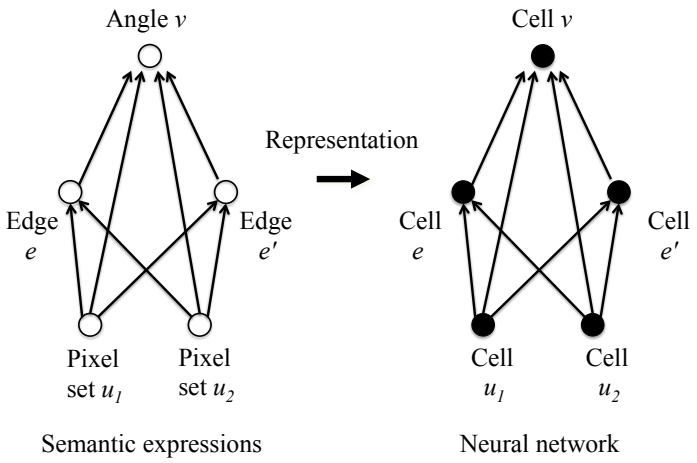


Figure 37: **A mapping of semantic expressions into a neural architecture, mathematically defined.**

Another consideration arises as we proceed from edges to angles to triangles and then to more complex imagery: It becomes increasingly difficult to express the semantics of the cells and their synaptically-weighted connections strictly in terms of sets of pixels in a diagram. For example, the pixel sets that help define triangles and other shapes become unwieldy and of questionable explanatory value in expressing complex stimuli. And what about non-visual information, such as auditory and somatic inputs and non-sensorial neural structures such as those involved in decision-making, planning, and motor control? If we wish to increase the capabilities of Agent R to render it more generally useful, all these functionalities will surely become necessary. Is there a mathematical framework more all-encompassing than pixel sets and injections for the expression of the semantics of neural network structure and activity?

4.1 A deeper look at sets, functions and composition

In the following, the mathematical terminology, the accompanying notation, and how to use it are well-known to those who apply mathematics, but as stated in the Introduction

this paper is meant for a wider audience. Therefore, we continue to introduce the needed information.

The analysis of commutative diagrams depends by definition upon the notion of composition. Composition is an operation upon pairs of functions. It is a *partial* operation because it applies only to function pairs that are properly aligned. That is, two functions can be composed to obtain a third function if and only if the image set of function values of the first function f is a subset of the set the second function g maps from; Fig. 38 provides an illustration. The set a function maps from—the set it is defined upon—is called its *domain*, and for example we denote the domain of f by $\mathrm{dom}(f)$. In like fashion, the set of images $f(x)$ is denoted by $\mathrm{ran}(f)$. An arbitrary element x being an element of $\mathrm{dom}(f)$ is denoted by $x \in \mathrm{dom}(f)$. Let $y = f(x)$. Then $y \in \mathrm{ran}(f)$. The composition evaluated at a value x is $g(f(x))$, and this is evaluated in stages as $g(f(x)) = g(y)$, that is f is evaluated first to obtain y and then g is evaluated on y. There is an algebraic notation for composition which will be important in discussing category theory. In the present case, the composition of f and g is the function denoted as $g \circ f$, defined by the expression $(g \circ f)(x) = g(f(x))$.

To see the usefulness of all this notation, suppose that $\mathrm{ran}(f)$ is not a subset of $\mathrm{dom}(g)$; that is, $\mathrm{ran}(f) \not\subseteq \mathrm{dom}(g)$. Then there is at least one element $f(x_0)$ of $\mathrm{ran}(f)$, where $x_0 \in \mathrm{dom}(f)$, that lies outside $\mathrm{dom}(g)$, or $f(x_0) \notin \mathrm{dom}(g)$. Then if we try to evaluate the composition $g(f(x))$ where x takes the value x_0, we can get as far as the evaluation $f(x_0)$ and no farther. Since $f(x_0) \notin \mathrm{dom}(g)$, $g(f(x_0))$ does not make sense and therefore the putative function $g \circ f$ is not well-defined. Paying attention to facts like this is essential in applying mathematics successfully. In the case of neural representations, x_0 could be a pixel, $f(x_0)$ its image in the RF of an edge-detector cell, D_e, and g a function mapping D_e into the RF of an angle-detector, D_v.

In Fig. 39, without the shading in Fig. 15 but with other added detail, the three functions that map D_{u_1}, D_{u_2}, and D_{u_3} into D_v are labelled i_1, i_2, and i_3, respectively (notice that as in Fig. 15 the set names have been simplified to u_1, u_2, and u_3). The sets D_{u_1}, D_{u_2}, and D_{u_3} are the domains $\mathrm{dom}(i_1), \mathrm{dom}(i_2)$ and $\mathrm{dom}(i_3)$ and the circles of same size inside D_v are their ranges $\mathrm{ran}(i_1), \mathrm{ran}(i_2), \mathrm{ran}(i_3)$, as indicated. Separately (see Fig. 40, which is Fig. 15 with now *all* the functions labelled), D_{u_1}, D_{u_2}, and D_{u_3} are mapped first into the edge-detector RF $D_{e'}$ via the functions f_1, f_3, and f_5 and into D_e via the functions f_2, f_4, and f_6. The functions i_4 and i_5 map $D_{e'}$ and D_e, respectively, into D_v. As a consequence, the composition $i_5 \circ f_2$ maps D_{u_1} into D_v; the other compositions map the other u-cell RFs into D_v as is evident from the diagram in Fig. 40. Now, following the pathways in the diagram, the path through f_2 followed by i_5 commutes with i_1 and therefore these paths form a commutative diagram which looks like a triangle of arrows. We can call this diagram a commutative triangle (not to be confused with the visual triangles being represented by colimits); it corresponds to the function equation $i_5 \circ f_2 = i_1$. Notice that the pathway through f_1 and i_4 also commutes with i_1 and therefore $i_4 \circ f_1 = i_1$. In fact, by the same reasoning from an inspection of the full diagram (or simply from the

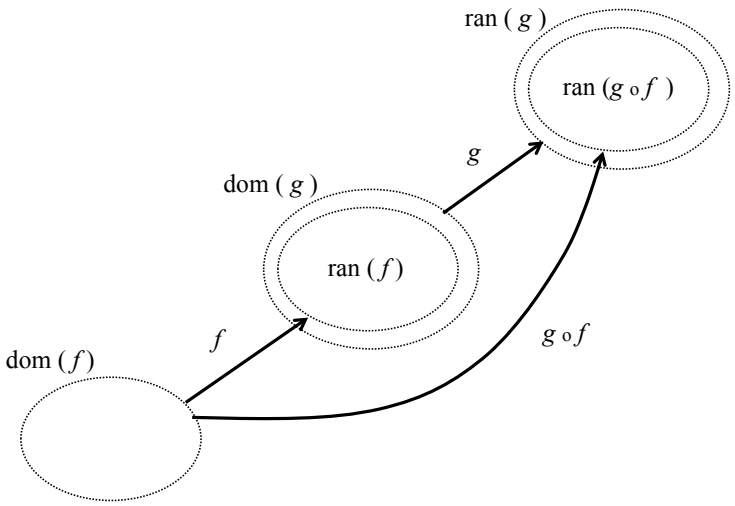


Figure 38: **A function composition** $g \circ f$**.**

rule that two things equal to the same thing are equal to each other), $i_5 \circ f_2 = i_4 \circ f_1$. In order for this and the other composition equalities in the diagram to hold, D_{u_1}, D_{u_2}, and D_{u_3} must each map to a single image in D_v. This proves that the diagram as drawn, with the single images of the u-sets and the e-set images containing them within D_v, is correct. Along with this, the overlap between the e-set images is also correct as drawn. The images have been properly blended in the colimit representation.

Now, since the ranges of i_1, i_2, i_3 and their commuting composites all are subsets of the same set D_v, and D_v is an important set since it is the RF of the angle-detector v, it is convenient to have a term that emphasizes this. We say that D_v is the *codomain* of i_1, i_2, i_3 and their composite twins, $D_v = \text{cod}(i_1)$, $D_v = \text{cod}(i_2)$, and $D_v = \text{cod}(i_3)$. This implies that the ranges of the three functions are subsets of their common codomain, $\text{ran}(i_j) \subseteq \text{cod}(i_j)\,(j \in \{1,2,3\})$. In fact, $D_{e'} = \text{cod}(f_j), (j \in \{1,3,5\})$ and $D_e = \text{cod}(f_j), (j \in \{2,4,6\})$. This is convenient because also $D_{e'} = \text{dom}(i_4)$ and $D_e = \text{dom}(i_5)$, so each composite $i_4 \circ f_1, i_4 \circ f_3, i_4 \circ f_5$ involves a codomain-to-domain match at $D_{e'}$ and each composite $i_5 \circ f_2, i_5 \circ f_4, i_5 \circ f_6$ involves a codomain-to-domain match at D_e, that is, the codomain of the first function in each composition is the same as the domain of the second. This simplifies things since it allows dispensing with function ranges, which vary with each function, and allows different functions with the same *domain* to also have the same *codomain* that contains their different ranges. This simplification makes it easier to visualize compositions of functions. In constructing further representations such as triangles, for example, as the codomain of the three functions i_1, i_2, i_3, D_v will match the domain of any function mapping it into the colimit set in a commutative diagram for a triangle or other polygon.

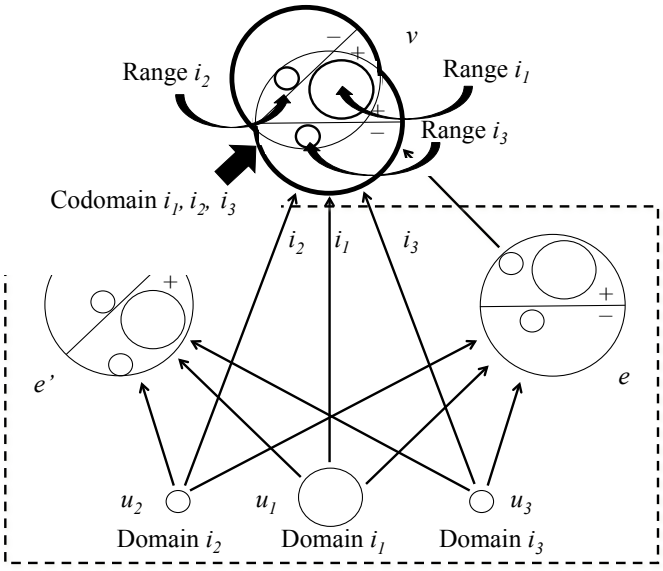


Figure 39: **The domains, ranges and shared codomain (inside the bold boundary) of the three functions $i_1, i_2,$ and i_3.**

A commonly-used notation for functions which is explicit about the domain and codomain

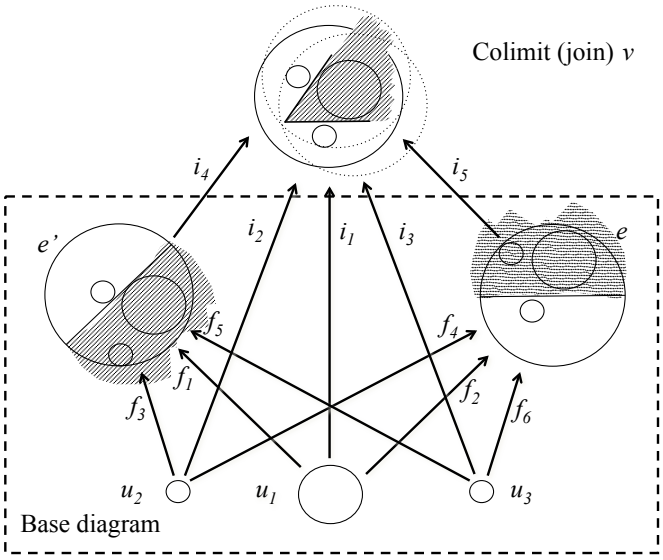


Figure 40: **A colimit for a base diagram of sets and functions with an angle as colimit object.**

is $f{:}D \longrightarrow C$. This puts the function arrow and the domain and codomain right in the label for the function and can be an aid in associating text with diagrams, which contain arrows. We use the notation f, $f{:}D \longrightarrow C$ and $\mathrm{dom}(f) = D, \mathrm{cod}(f) = C$ interchangeably, as convenient. Another convenience is that when two or more functions have the same domain and codomain, for example $f{:}D \longrightarrow C$ and $h{:}D \longrightarrow C$, this can be expressed in a single sentence as follows: $f,h{:}D \longrightarrow C$. This annotated-arrow notation is helpful in discussions involving multiple sets and functions, especially in connection with our focus on function composition and commutative diagrams. In computer science the specificity of using domains and codomains is referred to as "strong typing". In fact, with the foregoing discussion we have begun transitioning into category theory, a form of mathematics that is the ultimate in strong typing. Now considering an additional function $g{:}C \longrightarrow B$, with $\mathrm{dom}(g) = C, \mathrm{cod}(g) = B$, there is a domain/codomain match $\mathrm{dom}(g) = C = \mathrm{cod}(f)$, so the composition $g \circ f$ is well-defined. Notice that $\mathrm{dom}(g \circ f) = D = \mathrm{dom}(f)$ and $\mathrm{cod}(g \circ f) = B = \mathrm{cod}(g)$.

4.2 A new look at pixel sets and functions

Let us apply these ideas to the sets and functions in Part 1. The shaded regions in Fig. 40 indicate the excitatory inputs to each e-cell when it acts alone as an edge detector, and the net excitatory input to the angle-detector colimit v-cell when both e and e' are acting upon v along with the u-cell inputs (as before, only $D_e^+ \cap D_{e'}^+$ is shaded; remember, the excitatory inputs to the other intersection regions, $D_e^+ \cap D_{e'}^-$ and $D_e^- \cap D_{e'}^+$, are far outweighed by the inhibitory inputs). Recall that the illustrations in the accompanying figures replace the names of the cell RFs, for example D_e, with the cell names, such as e—for simplicity. We now reinterpret the quantities involved.

As in Chapter 5, the sets $D_{u_1}, D_{u_2}, D_{u_3}, D_e, D_{e'}, D_v$ and the function arrows labelled $f_j (j \in \{1, 2, \ldots, 6\})$ and $i_j (j \in \{1, 2, \ldots, 5\})$ form two diagrams, one inside the other. Recall that the functions can also be written as, for example, $f_1{:}D_{u_1} \longrightarrow D_{e'}$ and $i_4{:}D_{e'} \longrightarrow D_v$. The inner diagram, inside the dashed box labelled "Base diagram", has a cone-like structure attached to it to make the larger diagram. The latter is the defining diagram of the colimit, which commutes. To see an example of this commutativity, notice first that for the two functions i_1 and $i_4 \circ f_1$, a composition, $\mathrm{dom}(i_1) = D_{u_1} = \mathrm{dom}(i_4 \circ f_1)$ and $\mathrm{cod}(i_1) = D_v = \mathrm{cod}(i_4 \circ f_1)$, or in a more form, $i_1, i_4 \circ f_1{:}D_{u_1} \longrightarrow D_v$. Because they have the same domain and codomain and at least one of them is formed by composition along a path in a commutative diagram, i_1 and $i_4 \circ f_1$ are one and the same function, $i_1 = i_4 \circ f_1$. By the same reasoning, $i_1 = i_5 \circ f_2$. Since the two compositions are equal to the same function i_1, they are equal to each other, $i_4 \circ f_1 = i_5 \circ f_2$. If we wish, we can summarize the foregoing facts all in one compound mathematical sentence as $i_4 \circ f_1 = i_1 = i_5 \circ f_2{:}D_{u_1} \longrightarrow D_v$.

Another way of stating this is that the two triangular diagrams consisting of sides i_1, i_4, f_1 and i_1, i_5, f_2, respectively, commute and they share a common side i_1, so they

can be "pasted together" to form a larger commutative diagram, which appears as a trapezoidal shape with the arrow i_1 along the middle in Fig. 40. The equivalence of the three morphisms $i_4 \circ f_1, i_1, i_5 \circ f_2 : D_{u_1} \longrightarrow D_v$ states that the sets $D_e, D_{e'}$ are blended along their common subset D_{u_1} since the latter set is mapped to the same subset of D_v by the morphisms defined by the three pathways from D_{u_1} to D_v. The same statements hold for each of the other pairs of triangular diagrams based at D_{u_2} and D_{u_3}. The result is three commutative trapezoidal-shaped diagrams and thus D_e and $D_{e'}$ are blended along $D_{u_1}, D_{u_2}, D_{u_3}$. Since the functions involved in the diagrams simply map the elements of one set to themselves as elements of another set, $D_{u_1}, D_{u_2}, D_{u_3}$ taken together form the intersection $D_e \cap D_{e'}$ which is contained within the union $D_e \cup D_{e'}$.

It may seem that the foregoing discussion of commutative diagrams is a complicated way to describe a set union with an embedded set intersection. If the sole topic were set unions and intersections, that would be the case. However, recall that in Part 1 it was pointed out that the commutativity of the three trapezoidal diagrams, taken together to form the larger diagram of Fig. 40 (originally Fig. 15), was used to introduce the notion of a colimit. The colimit is a mathematical derivation of the combining without duplication of the influence of sets of pixels in an RF. As expressed through the input sum to the angle detector cell v, this combining is what makes possible a properly-functioning angle detector.

Yet as was also pointed out, this view of the RF colimit as a set union does not adequately express the semantics of the diagram as mapped into the neural network. For example, the effect of the inhibitory connections is not represented; for this, we rely on the input sum to v, which is an adjunct to the set-theory model that is our current means of formalization. The problem with this formalization is that it is not complete: The set-theory model does not explicitly incorporate the input sum model because it does not make explicit the excitatory and inhibitory effects of the pixels, so the two models are only informally bound together. In large part, this is not a fault of set theory but of the highly-simplified form of the functions used here, since they are merely injections operating upon pixel sets. Another issue is that there is no obvious way to formalize the association of sets and functions with the neurons and their synaptically-weighted connections in a neural network. How can this be done for neural architectures more complex than the ones so far discussed, where pixel sets are not adequate representatives of the inputs to neurons?

The next section introduces category theory, the field of mathematics acually used for the formal model of this book. Along with this introduction, we sketch the overall scheme for applying the mathematics. Before replacing the pixel set model of semantics with a more comprehensive formalization, we shall first introduce category theory and show how we use it to formalize neural networks and their semantics in processing sensor stimuli as indicated by Fig. 37.

4.3 Categories

The existence of commutative diagrams and their usefulness in defining quantities such as colimits suggests that composition is a valuable aspect of computing with functions. In fact, there are two key properties of function composition that are generalized to mathematical structures called *categories*, which more fully exploit the composition idea. First, notice that for each set D there is an *identity function* $\mathrm{id}_D{:}D \longrightarrow D$, defined by $\mathrm{id}_D(x) = x\,(x \in D)$; this makes id_D the simplest of all injections operating upon D. Identity functions and their categorical generalization are of prime importance. This is because of a key property of identities: composition with an identity leaves a function unchanged. For example, consider the sine function $\sin{:}\mathbf{R} \longrightarrow [-1, 1]$ defined on the set of real numbers $\mathbf{R}$, with values $\sin(x)$ ranging over the real interval $-1 \leq x \leq 1$, alternately written as $[-1, 1]$. Notice that $(\mathrm{id}_{[-1,1]} \circ \sin)(x) = \mathrm{id}_{[-1,1]}(\sin(x)) = \sin(x)$ and $(\sin \circ \mathrm{id}_{\mathbf{R}})(x) = \sin(\mathrm{id}_{\mathbf{R}}(x)) = \sin(x)$. This property can be expressed independently of the elements of sets. That is, using the function names only, $\mathrm{id}_{[-1,1]} \circ \sin = \sin$ and $\sin \circ \mathrm{id}_{\mathbf{R}} = \sin$. In fact, for any sets D, A and a function $f{:}A \longrightarrow D$, it is always true that $\mathrm{id}_D \circ f = f$ and $f \circ \mathrm{id}_A = f$; that is, the composition of a function with an identity simply returns that function. The other key property of composition, which is easily shown to hold, is the *associative law of composition*: Given functions $f{:}A \longrightarrow D$, $g{:}D \longrightarrow B$ and $h{:}B \longrightarrow E$, the two possible ways of computing the composition of all three by computing compositions of two functions at a time yield the same function (note that the domains and codomains all match as they should): $(h \circ g) \circ f = h \circ (g \circ f){:}A \longrightarrow E$. It can be shown that associativity generalizes to chains consisting of more than three functions.

The point is that for many purposes, the notation including function evaluations $f(x)$ on set elements x can be replaced by a notation which is explicit about how functions relate to sets and to one another, such as $f{:}A \longrightarrow D$ and $(h \circ g) \circ f = h \circ (g \circ f){:}A \longrightarrow E$. The element-free notation is a significant development: It suggests the generalization from sets and functions to categories. A *category* C consists of a collection of *objects* and *morphisms* (or *arrows*), with a composition for morphisms that has the identity and associative properties. The objects are all examples of a type of mathematical structure (such as sets) and the arrows, or morphisms, are a type of relation between pairs of objects (such as functions). As with sets, the objects related by a morphism $f{:}a \longrightarrow b$ in C are called its *domain* a and *codomain* b and we write $\mathrm{dom}(f) = a, \mathrm{cod}(f) = b$. Also as with sets, a composition of two morphisms is defined if and only if the domain of one of them is the codomain of the other. In the case of identities and also other morphisms of the form $f{:}a \longrightarrow a$ where $\mathrm{dom}(f) = \mathrm{cod}(f)$, the domain and codomain consist of two copies of the same object. Thus, each object a of the category has an identity morphism $\mathrm{id}_a{:}a \longrightarrow a$ with the same property with respect to composition as the identity functions just discussed, so that if $f{:}a \longrightarrow d$ is a morphism of C, then $\mathrm{id}_a \circ f = f$, and so forth. Notice how expressing identities without explicitly mentioning set elements, relying only on the key identity property of composition with explicit domains and codomains, enables this generalizing to categories in which the objects need not be sets. The same is true for

the associative property: Merely replace the sets and functions in the previous paragraph with objects and morphisms of C.

4.4 Examples of categories

Any reference book on category theory contains numerous instructive examples of categories and their many interesting properties (among the more accessible are [17],[12], and [2], but there are others). Let us briefly examine some examples of categories. First of all, the totality of sets and functions form a category and a very important one, called **Set** or **Sets** depending upon the author's preference. In fact, there are many categories whose objects are sets, either finite or unrestricted in cardinality. The category **Set** appears most often in writings, but a category of finite sets is common in applications such as the analysis using pixel sets in Part 1. Hereafter we shall regard our pixel sets as objects in a category of finite sets which we call **FSet**. Another category having sets as objects is the category **Rel** of binary relations. Its objects are sets but its morphisms are relations between pairs of sets which, unlike functions, can be many-to-many mappings; functions are strictly many-to-one relations (thus, relations include functions). Another difference from **Set** is that relations are not required to map all elements of the domain into the codomain, whereas functions are required to do so. A relation $R{:}A \longrightarrow B$ is often expressed as a subset R of the cartesian product $A \times B$ of sets A and B, $R \subseteq A \times B$ because a relation is given by a set of ordered pairs (a, b), $a \in A, b \in B$, with a related to b by R.

In yet other categories the objects are sets with an associated structure such as an algebra or a topology and the functions preserve that structure. In the category **Grp**, the objects are algebras called groups and the morphisms are a special type of function called a homomorphism. An example of a group is $(\mathbf{Z}, +)$, where $\mathbf{Z}$ is the set of integers $\ldots -2, -1, 0, 1, 2 \ldots$ and + is the binary operation of the addition of integers. A property called commutativity holds because the order of addition is irrelevant, $x + y = y + x$, and associativity holds because the addition of several integers can always be performed by accumulating the sum by pairs of operands in any order, for example $x + (y + z) = (x + y) + z$. Notice the similarity to the associative law for categories. The element 0 serves as a very important group element called the group identity. This is due to the fact that, like the identity morphism for an object in a category, operating with it leaves elements unchanged: for example, $1 + 0 = 1$. As another property, true of certain groups called Abelian groups, each element has an additive inverse. Adding an element to its inverse yields the group identity; for example, $1 + -1 = 0$, so -1 is the additive inverse for 1 (and vice versa, $1 = -(-1)$, so 1 is the additive inverse of -1). Another group is the additive group of even integers $(2\mathbf{Z}, +)$, where the binary operation is still the addition of integers, but restricted to even integers (note that the sum of even integers is another even integer, so $+$ is a valid binary operation upon the set $2\mathbf{Z}$). An example of a homomorphism is a function we can call $\times 2{:}\mathbf{Z} \longrightarrow 2\mathbf{Z}$ which doubles an integer, for example $\times 2(1) = 2$, $\times 2(2) = 4,$, etc. The function $\times 2$ is called a homomorphism because it preserves the

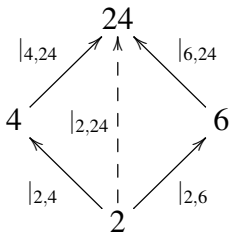


Figure 41: One of many commutative diagrams in the category $\mathbf{N}_{|}^{+}$.

structure of the addition operation: for example, $\times 2(0) = 0$, the identity of both groups, and $\times 2(1+1) = 4 = 2+2 = \times 2(1) + \times 2(1)$, so that the image of a sum is the sum of the images of the operands. To see another example of a homomorphism, consider the two groups $(\mathbf{R}, +)$ and $(\mathbf{R}^{+}, \cdot)$, the real numbers under the binary operation of addition and the positive reals under the binary operation of multiplication, respectively. Note that the commutativity, associativity, identity and inverse properties of addition hold for the reals as with the integers, which are a subset of the reals. Note also that the same holds for the positive reals under multiplication. The identity element for $(\mathbf{R}^{+}, \cdot)$ is 1 and each element $x \in \mathbf{R}^{+}$ has an inverse $x^{-1} \in \mathbf{R}^{+}$, where $x \cdot x^{-1} = x \cdot (1/x) = 1$. Now consider the function $\exp\colon (\mathbf{R}, +) \longrightarrow (\mathbf{R}^{+}, \cdot)$ given by $\exp(x) = e^{x} (x \in \mathbf{R})$. By the rules $e^{0} = 1$ and $e^{x+y} = e^{x} \cdot e^{y}$ this function is a homomorphism; for example, $\exp(x + -x) = \exp(0) = e^{0} = 1 = e^{x} \cdot (1/e^{x}) = e^{x} \cdot e^{-x} = \exp(x) \cdot \exp(-x)$. Hence, the image of a sum is the product of the images of the operands, and the example also shows that the image of the additive inverse of an element is the multiplicative inverse of the image of the element. There is an infinite number of groups and homomorphisms, including permutation groups with their homomorphisms, linear spaces with linear transformations and so forth.

The examples presented so far have objects that are sets or sets with structure (algebras). A distinctly different example of a category, the category $\mathbf{N}_{|}^{+}$, is represented by one of its commutative diagrams in Fig. 41. The objects of $\mathbf{N}_{|}^{+}$ are the positive natural numbers, 1, 2, 3, ... , . Unlike the previous examples, the morphisms here are not like functions at all. A morphism from n to m, denoted $|_{n,m}\colon n \longrightarrow m$, exists 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, $n \mid n$. There are two apparent morphisms with domain 2 and codomain 24 in the diagram of Fig. 41, both being compositions along a path directed through a third diagram object (4 and 6, respectively). However, the fact that there is at most one divisibility morphism from one natural number to another implies the equation $|_{4,24} \circ |_{2,4} = |_{2,24} = |_{6,24} \circ |_{2,6}$. Thus, the diagram commutes.

The category $\mathbf{N}_{|}^{+}$ provides an example in which the morphisms are instances of what we usually think of as a relation—the divisibility relation defined on positive natural numbers. It also exemplifies a relatively simple type of mathematical structure known as a

partial order, having at most one morphism between a pair of objects, all morphisms having a shared sense of direction (that is, there are no cycles).

The same objects can exist in a variety of categories with the morphisms differing between the categories. Categories of sets with either functions or relations as morphisms have been discussed. Another example concerns two categories with natural numbers as objects. The category $\mathbf{N}_{|}^{+}$ based upon the divisibility relation has the positive natural numbers as objects. The category $\mathbf{N}_{\leq}$ has all the natural numbers 0, 1, 2, 3, ... , as objects and a morphism $\leq_{n,m}: n \longrightarrow m$ exactly when the inequality $n \leq m$ holds between two of them. 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, $n \leq n$. This category is a partial order in which the objects "line up" because any two natural numbers n and m are related by the transitive relation $\leq$.

Many categories are derived from other categories. A simple example of this is a subcategory: Given a category C, a subcollection D of its objects and morphisms is a subcategory of C if (1) for each object d of D, the morphism id_d of C is also a morphism of D, and (2) for each pair of morphisms $f: a \longrightarrow b, g: b \longrightarrow c$ of D, their composition $g \circ f: a \longrightarrow c$ is also in D as well as in C. Obviously, the laws of identity and associativity are inherited directly from C, so if it satisfies the two conditions (1)–(2) D is indeed a category. Other examples of derived categories will be described in the process of defining colimits and limits.

4.5 Categorical structure

It is apparent that the morphisms give a category an underlying directed-graph structure in which objects serve in the role of nodes and morphisms act as edges. It is important to realize that a category is more than a graph. First, there can be many morphisms in either or both directions between a pair of objects a and b, hence the need for the notation $f: a \longrightarrow b$ and $g: b \longrightarrow a$ (as opposed to "edge (a, b)" in a graph) and the terminology "domain" and "codomain". The most important distinguishing feature of a category, however, is the notion of composition with its identity and associative properties. Composition derives its importance from the existence of commutative diagrams, the fact that in many cases (depending upon the category and the particular diagram) two alternative pathways through the arrows leading from one object to another yield the same morphism when the composition of the arrows is calculated along each path. This means that, unlike the situation with graphs, a path through the arrows in a category is associated with a precise notion of cumulative effect or meaning; and, further, different paths whose compositions have the same domain and codomain can have the *same* meaning, and when this is true we have a commutative diagram.

One sometimes hears a statement such as "the two [concepts, data types, program

constructs, etc.] are in some sense isomorphic". Is there a precise meaning for this term? Indeed, category theory provides a mathematically rigorous notion of "isomorphism": If a, b are objects of a category C such that there exist arrows $f: a \longrightarrow b$ and $g: b \longrightarrow a$ with $f \circ g = \mathtt{id}_b$ and $g \circ f = \mathtt{id}_a$, then the morphism f is called an *isomorphism* (as is g also) and g is called its *inverse* (and f is called the inverse of g), and the two objects are said to be isomorphic. The property of an identity morphism ensures that isomorphic objects in a category are interchangeable in the sense that they have the same relationships with all objects of the category. An isomorphism $f: A \longrightarrow B$ in **Set** is a one-to-one, onto function, meaning that the image y is unique for every x in $y = f(x)$ and also that every $y \in B$ is the image of some element $x \in A$. Because of this, $f: A \longrightarrow B$ has an inverse $g: B \longrightarrow A$, with $x = g(y)$ for every $y = f(x)$ so that $(g \circ f)(x) = x = \mathrm{id}_A(x)$ and $(f \circ g)(y) = y = \mathrm{id}_B(y)$. An isomorphism in **Grp** is a one-to-one, onto homomorphism; an example is the isomorphism $\exp: (\mathbf{R}, +) \longrightarrow (\mathbf{R}^+, \cdot)$, whose inverse is the isomorphism $\ln: (\mathbf{R}^+, \cdot) \longrightarrow (\mathbf{R}, +)$, where ln is the natural logarithm operation, with $(\ln \circ \exp)(x) = \ln(e^x) = x = \mathrm{id}_{(\mathbf{R}, +)}(x)$ and $(\exp \circ \ln)(y) = e^{\ln(y)} = y = \mathrm{id}_{(\mathbf{R}^+, \cdot)}(y)$.

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

4.6 Colimits and Limits

Before discussing limits and colimits, key structures of great importance, it is necessary to dispense with a mathematical formality. Our concern is solely to limit the discussion in this book in order to avoid making false statements. Because of a logical trap known as Russell's Paradox (after the famous twentieth century mathematical philosopher Bertrand Russell), it is common practice to sort mathematical collections into two kinds: sets (which can be either finite or infinite) and proper classes. Proper classes are those whose definitions make them "too large" to be called sets—the finite/infinite distinction is insufficient

to cover this notion of largeness. To avoid a lengthy discussion, we refer the interested reader to any reference on set theory and logic. Categorical authors often label the two kinds as "small sets", which we shall call, simply, "sets", and "large sets", the proper classes. It happens that in many categories of interest such as **Set** the objects and morphisms do not form a set—they form a proper class. Such categories are called "large categories", and those whose objects and morphisms form a set are called "small categories". To get to the point, large sets are of no concern here, and in fact we could have written this book based upon only finite categories (therefore definitely small) which are subcategories of the ones we do discuss. We have avoided this to forego more discussion. Therefore, regardless of the "size" of a category under discussion, we shall consider among its many diagrams only those whose objects and morphisms form a "small set". When we say "diagram", we mean "small diagram" and in fact we are really interested only in "finite diagrams".

Let Δ be a diagram in a category C as shown in figures 42 and 43 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 Fig. 43 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 Fig. 42 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 of each leg morphism g''_i of K'' by composition with the leg morphism g'_i of K', as illustrated in Fig. 42 Re-using the symbol h for notational efficiency, we denote the cocone morphism determined by h as $h: K' \longrightarrow K''$.

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

The defining diagram from the angle colimit illustrated in Fig. 40 is an example. Notice that as drawn, the angle object has only the information that can be gleaned from the two edge objects and the u-cells that blend them. That is, there is no extraneous information to be had from the base diagram when the cocone is initial. For our purpose, this is the significance of the fact that the cocone formed by the morphisms i_1, i_1, i_1, i_1, i_1 is an initial

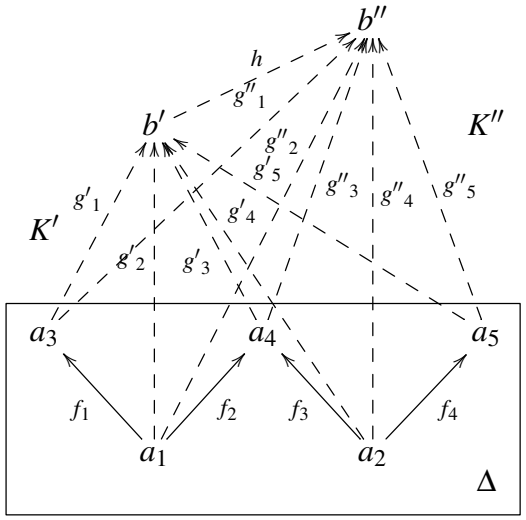


Figure 42: A cocone morphism $h: K' \longrightarrow K''$ in $\mathbf{coc}_\Delta$ is a morphism $h: b' \longrightarrow b''$ in C between the apical objects b' and b'' of cocones K' and K'', respectively, that is a factor of each leg morphism $g''_i: a_i \longrightarrow b''$ of K'', with $g''_i = h \circ g'_i$.

cocone. Any other cocones must add information not in the base diagram; for example, they might be part of a cocone attached to a larger diagram.

The notion of limits in a category can be obtained directly from the notion of colimits by reversing the arrows and replacing initial objects with terminal objects. Let Δ be a diagram in a category C as shown in Fig. 44 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

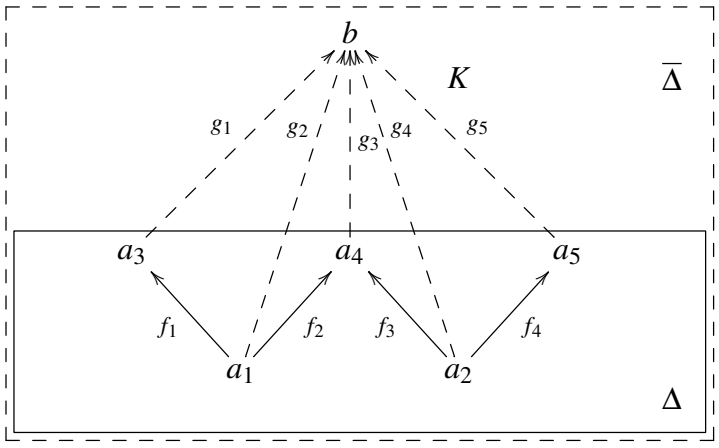


Figure 43: A colimit for a diagram Δ. The extended diagram $\overline{\Delta}$ extends Δ with a conical structure of morphisms from all diagram objects $a_1, \ldots, a_5$ pointing to an apical object b.

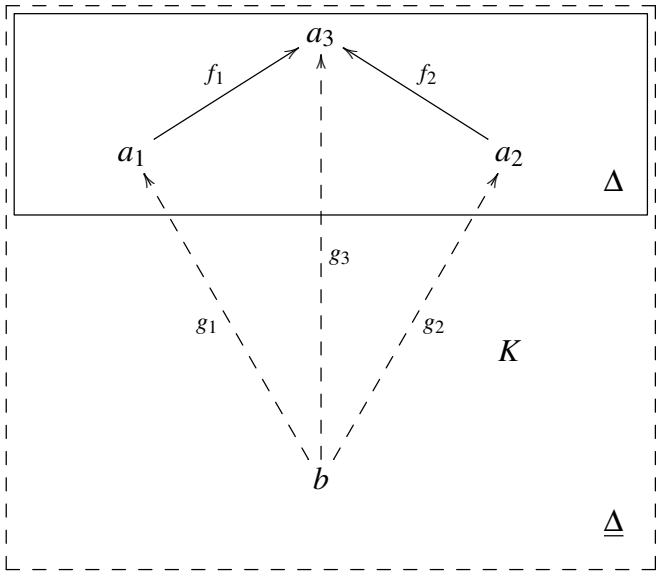


Figure 44: A limit for a diagram Δ. The extended diagram $\underline{\Delta}$ extends Δ with a conical structure of morphisms to all diagram objects $a_1, \ldots, a_3$ from an apical object b.

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

The category **Set** contains limits and also colimits for all of its diagrams. The base diagram of pixel sets and functions in Fig. 40 is an example in which the colimit object is a set union because the functions simply embed subsets in the sets of which they are a part. In general, once one expresses any diagram in **Set**, a colimit — a blending of sets incorporating all the information in the diagram — can be calculated by an automated process. On the other hand, not all diagrams in **Set** have limits. A theorem in category theory can be used to derive an algorithm for calculating limits in any category that contains limits for all of its diagrams, and similarly for colimits. Colimits and limits do not exist for all diagrams in all categories, but as will be shown they can be very useful where they do exist.

4.7 Functors

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. The functorial image of each composition $g \circ_C f$ defined for morphisms of C is a composition of the images of its factors in the same order, $F(g \circ_C f) = F(g) \circ_D F(f)$. Also, each identity morphism $\mathtt{id}_a$ of C maps to the identity morphism of the image $F(a)$ of its domain a, that is, $F(\mathtt{id}_a) = \mathtt{id}_{F(a)}$. It follows from these properties that F preserves the commutativity of diagrams. That is, the images of the objects and morphisms in a commutative diagram of C form a commutative diagram in D. Since any constraints on the structure of a category are expressed in commutative diagrams, this means that any structural constraints expressed in C are translated intact into D and, hence, F is a structure-preserving mapping.

The two categories $\mathbf{N}_{|}^{+}$ and $\mathbf{N}_{\leq}$ yield a simple example of this fundamentally important structural mapping. 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 $F(|_{n,m})$; that is, $F(|_{n,m}) = \leq_{n,m}$. It is easy to see that this mapping of objects and morphisms is a functor because $n \mid m$ implies $n \leq m$. Notice that the compositional structure of $\mathbf{N}_{|}^{+}$ is appropriately preserved. This functor is partially illustrated in Fig. 45.

How can a neural network be analyzed as a category and related to a category expressive of its semantics? The groundwork for this discussion has now been established.

5 Neural Architectures and Neural Categories

The description of the semantics of a neural network, Agent R1.3's brain, has been developed in a semi-formal way through pixel sets, which are objects in the category **Set**, and injection functions, a special case of the morphisms. However, a formal statement

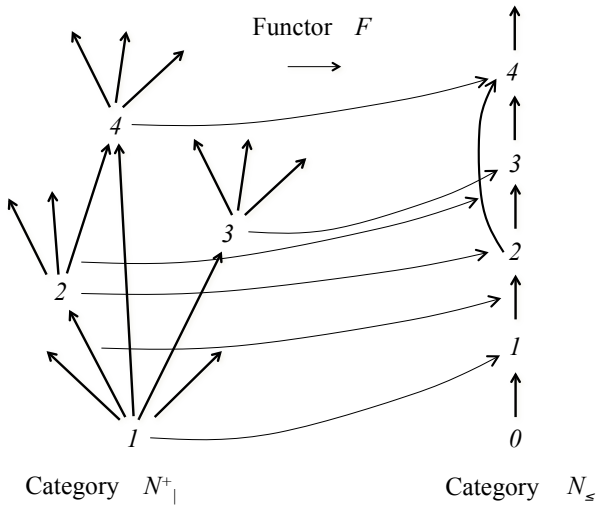


Figure 45: **A functor $F : \mathbf{N}^+_| \longrightarrow \mathbf{N}_\leq$, illustrated by showing a few maplets of numbers to numbers ($1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3, \ldots$) and divisor relations to less than or equal relations ($|_{1,2} \mapsto \leq_{1,2}, |_{2,4} \mapsto \leq_{2,4}, \ldots$).**

of semantics requires more than a language for describing meaning (such as the language of sets and functions): it also requires a mathematical way of mapping the semantic descriptions to the system whose semantics is being described. Since categories have been suggested as the mathematical vehicle for the formalization of the semantics of Agent R1.3's brain, the desired mapping of semantic descriptions (a category of pixel sets and functions) to Agent R1.3's brain would logically be a functor. This requires that Agent R1.3's brain be transformed into a type of category within which neural structure can be represented as suggested by Fig. 37. The arrow in the Figure is a functor which maps semantic objects, the pixel sets, and their morphisms, the pixel-set functions, into some part of an appropriate neural category. In this scheme, the semantic description category is the domain of the semantic functor and the neural category is its codomain.

It is intended that the neural category construction proposed here will be applicable to any neural network architectural model. For simplicity, however, the present construction applies to rate-coding neural models in which spike trains and other biological phenomena, although important, are subsumed. This simplification will be appropriate for finding solutions to Agent R1.3's brain-design problems, a process we began earlier. For biological neural systems the situation is more complex, and the mathematical model must undergo some refinement to address this issue.

A semantic analysis of Agent R1.3's brain starts with the structure of its neural network, its neural cells and their synaptic connections, the cell thresholds, the synaptic

weights, and the differentiation between excitatory and inhibitory synapses. The analysis also must take account of the possible ranges of activity of the neurons and the connection-weighted signals through which they communicate. Finally, the analysis includes something that was only just mentioned: The long-term synaptic adaptation that changes the connection weights to form semantic structures such as the limits and colimits that represent what the network has learned from its history of activity in reponse to sensory stimuli. Thus, in order to support a semantic analysis the categorical neural model must have a sense in which it represents the dynamics of the network at some level of detail, including not only neural activity but long-term synaptic adaptation. On the other hand, for our purposes it is not necessary to apply a full dynamic system model. Although such models are essential in the detailed study of neural network processing, the semantics of episodes of neural activity and adaptation for the most part can be studied with a summary of what has occurred during an episode. The interest here is in modeling the outcome at key stages of neural processing and weight adaptation. Our main requirement is twofold: First, that there be an unambiguous and consistently-applied criterion associating significant states of activity with objects and morphisms in a neural category; second, that there be equally rigorous criteria for associating changes to the array of network weights with changes to the neural category. Possible criteria for what constitutes a significant state of processing will be discussed here only in the most general manner. Note, however, that many artificial neural network models in the literature pass over a detailed dynamic model in favor of the expression of neural/synaptic processing at discrete stagesand some of these models have been applied to biological investigations in neuroscience .

5.1 Neural objects

The earlier discussion suggests that a neural cell in a sufficiently high state of activity is representing some item by virtue of its connection-weighted inputs, either directly from a sensor element or through afferent connections from other cells. For example, an edge detector for Agent R1.3 represents a contrast along the edge of an object appearing in the robot's visual field. This is because the edge-detector cell responds appropriately to the balance of excitatory versus inhibitory inputs from pixel elements on opposite sides of the edge. The semantics of the edge detector are manifest neurally in not only the presence of the edge detector cell, but also a range of levels of activity that the cell has when an edge is present at the appropriate place in the visual field. We have been regarding objects in the category of semantic expressions in part as pixel sets, which being finite can be regarded as objects of the category **FSet** mentioned earlier. However, fully expressing the semantics demands that the balance of excitatory versus inhibitory weighted sums input to the representing cells be included in the analysis along with the pixel sets. Correspondingly, the manifestation of the semantics in the neural network, expressed mathematically, has two parts. In the neural category, a cell such as an edge-detector is not, in and of itself, an object. It is the cell together with the range of activities signifying an edge that is an object.

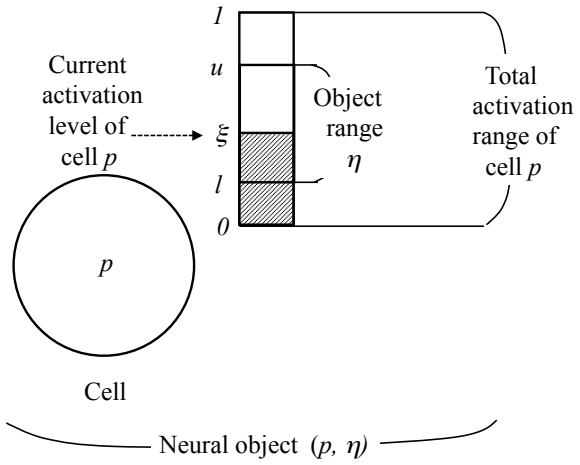


Figure 46: **An object in a neural category.**

A *neural object* (p, η), is illustrated in Fig. 46. The cell has the label p and η is the range of real-number activity values assigned to the object, where $\eta = \{\xi \in \mathbf{R} \mid \ell < \xi \leq u\}$ for real numbers ℓ, u which are the lower and upper bound of the range with $0 \leq \ell < u$. By activity level we mean that the internal activity θ of the cell, regarded as its connection-weighted input sum, has exceeded the cell's firing threshold θ_T, in which case the cell's signal function ϕ transforms the internal activity into an activity level ξ. For our purposes, this is the output signal which (p, η) transmits. Objects also can be defined by levels of activity at or below the carrier cell's threshold if desired, in which case the single value 0 is normally used in place of η by labelling the object $(p, 0)$. We say that the object (p, η) (or $(p, 0)$) is *carried by* the cell p. There can be as many objects as desired associated with a given cell. In particular, the definition of a neural category need not require objects to be given by binary, "off-on" states of a cell. If binary neural objects are used, η would include either the entire range of above-threshold levels of activity (the "on" state) or the single value 0 (the "off" state). When using this convenience, we shall denote an "on-state" object by the cell p that carries it if η has nonzero width, representing the above-threshold activity range, and an "off-state" object by $(p, 0)$; when not being specific, however, the notation (p, η) will suffice as it does for non-binary objects.

Thus, the neural category objects defined for a given neural network are determined not only by the network cells, but also by the choice of objects associated with its cells. An important point about this choice is the following: Since the reason for defining a neural category is to formulate a mathematical model of the network's semantics, the category does need to include the appropriate objects (p, η) with each cell p so that the activity ranges η cover the cell's response to objects of interest in its RF.

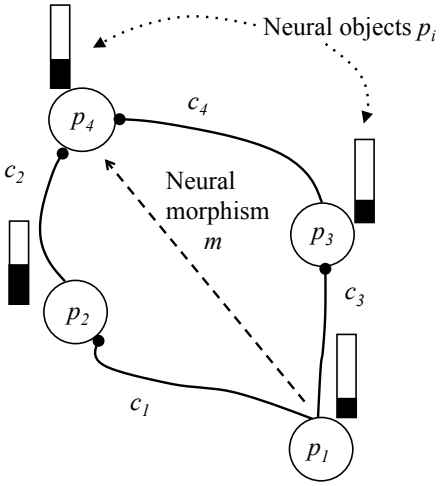


Figure 47: **A morphism in a neural category. To simplify, the objects are binary, encompassing the entire range of potential activity of their carrier cells, so the η's are omitted. The black bars illustrate an example of simultaneous activity levels of the cells.**

5.2 Neural morphisms

Defining a *neural morphism* $m\colon (p,\eta) \longrightarrow (p',\eta')$ between two objects (p,η), (p',η') is slightly more involved. Briefly, a morphism corresponds to the instances of simultaneous activity that can occur in a set of connection paths through the neural network connecting the cells p, p', which are the carriers of the domain (p,η) and codomain (p',η') objects of the morphism. A connection is active when its source and target cells are active, and a path is active when all of its connections are active. A path can have multiple connections along the way and therefore contain intermediate cells, or can consist of a single connection joining the cells p, p'. A set of simultaneously-active paths can have several members or can consist of a single path. All connections within each of the paths must have the same direction, away from the cell p and toward the cell p'. A morphism $m\colon p_1 \longrightarrow p_4$ between two binary "on-state" objects p_1 and p_4 is illustrated in Fig. 47, where p_2 and p_3 are intermediate cells within the two paths depicted. The bundle of connection paths together with the weights of the connections is *the carrier of the morphism, but not the morphism itself.*

To completely describe a morphism we must replace its bundle of connection paths with a kind of pathway that combines the *structure* of a connection path with its *set of potential states of activity*. These are obtained as follows. Replacing each cell along a connection path with a neural object that it carries, we obtain a *signal path*. Activity in

a signal path has a more restricted definition than activity in a connection path: A signal path is active when the activity of the source and target cells of each of its connections is an instance of activity of the objects for which all of the cells along the path are carriers. With this revision, a neural morphism is defined as a set of signal paths that are always simultaneously active. The objects at either end of each path—the domain and codomain of the morphism—are of course included in each signal path. The signal path set of a morphism can be represented by any subset of its members since all are active simultaneously, but whether explicitly mentioned or not all are a part of the morphism. Because of the simultaneity-of-activation property, any of the signal paths defining a morphism is called a carrier of the morphism.

To be more precise, we define a morphism step-by-step as follows (see Fig. 47). When a cell's activity persists for a significant amount of time within the specified range η of an object (p,η) carried by a cell p, the object (p,η) is said to be *highlighted*. Similarly, a signal path is highlighted when its objects are all highlighted. For two objects $(p,\eta),(p',\eta')$, a morphism $m\colon(p,\eta)\longrightarrow(p',\eta')$ is highlighted when any of its signal paths is highlighted (and, hence, when all of them are highlighted). And, finally: *A neural morphism m is a set of signal paths having a common source (the domain) and target (the codomain) and which always become highlighted simultaneously.*

To mitigate any confusion, let's illustrate with an example. For simplicity here and in much of the book, we shall use binary neural objects. A binary neural object is highlighted when its carrier cell is active and emitting a signal in case it is an "on-state" object, or is highlighted when inactive, represented by the single value 0, in case it is an "off-state" object. When using this convenience, we shall denote an object by the cell that carries it if η has nonzero width and by $(p,0)$ otherwise. In Fig. 47, a set of two signal paths containing such objects is shown having the object p_1 as their common source (the domain of the morphism for this example) and p_4 as their target (the codomain). The connections c_1, c_2, c_3, c_4 in the paths are all oriented with the same sense of direction, from source to target. As it happens, each path consists of two connections through an intermediate cell. The intermediate objects (cells in our binary model) are p_2 and p_3 as shown. The black bar inside the white bar for each object indicates its current activity level. In the illustration all four objects have positive activity levels; if the objects are defined as having positive activites (as opposed to objects representing zero activity), they are therefore highlighted. The two signal paths, defined using these objects and the connections, are therefore highlighted. If it happens that the two signal paths are always highlighted simultaneously, they define a morphism m which, in the illustration, is therefore highlighted. In concert with the category-theoretic notation introduced previously, m can be written $m\colon p_1\longrightarrow p_4$. The individual morphisms defined by the single-connection signal paths, m_1, m_2, m_3 and m_4, can be written $m_1\colon p_1\longrightarrow p_2, m_2\colon p_1\longrightarrow p_3,\ m_3\colon p_2\longrightarrow p_4$ and $m_4\colon p_3\longrightarrow p_4$.

5.3 Commutative diagrams

The definition of a neural morphism has been developed to be compatible with the categorical notion of composition as well as with the definition of a neural object. Notice that the illustration of a neural category commutative diagram in Fig. 48 looks very similar to that for a neural morphism in Fig. 47. This is no accident, for if neural morphisms are to form commutative diagrams then compositions along different signal paths with a common source and target object must result in a single morphism in such a diagram. In fact, Fig. 48 is just Fig. 47 enriched with the details of individual single-connection morphisms m_1, m_2, m_3 and m_4 whose compositions lie along the two multi-connection signal paths. In the general case, compositions along separate signal paths with the same source and target object may well define distinct neural morphisms. This could have been true of the compositions $m_3 \circ m_1$ and $m_4 \circ m_2$ in Fig. 48 were their signal paths not active simultaneously in all cases in which one of them is active. If they are always active simultaneously then we find that the two compositions are equal and we know that the diagram commutes and therefore represents a single neural morphism. Conversely, if we were designing a neural network and stipulated that as part of the design the two two-connection signal paths of Fig. 48 must form a commutative diagram and, hence, a single neural morphism, it follows that care must be taken to ensure that the two compositions are indeed equal, which in turn requires that their signal paths are always highlighted simultaneously. Ensuring that this is the case requires that any inputs to the cells in the two pathways, whether they occur through the pathway connections or originate outside the pathways, help to enforce the simultaneity of their activity. Thus, intentionally including a neural morphism in a neural network requires careful network design. The angle and triangle colimits discussed earlier are examples of this.

Finally, let us summarize while observing a few more details about the example of Fig. 48. Using the simplification of labelling binary neural objects as their carrier cells, morphism m_1 is represented by a single-connection signal path with $\mathrm{dom}(m_1) = p_1$, $\mathrm{cod}(m_1) = p_2$ and the weighted connection c_2. As with the morphism illustration, the connection weights are not shown. The other single-connection paths have connections c_2, c_3, c_4 and domain/codomain pairs as shown; for example, m_3 is defined by the signal path p_2, c_2, p_4. Both compositions $m_3 \circ m_1$ and $m_4 \circ m_2$ have the same domain and codomain, p_1 and p_4, respectively. The diagram involving both compositions commutes if the compositions are equal to one and the same morphism. In Fig. 48 this is illustrated by supplying the added arrow m_5. Correspondingly, we can write $m_3 \circ m_1 = m_5 = m_4 \circ m_2$.

In general, diagrams can be much more complex than this and contain multiple pairs of equivalent compositions, but the principle is the same. Thus, our definition of a neural morphism corresponds naturally to our definition of a commutative diagram.

As implied in an earlier paragraph, objects and morphisms other than those in the signal paths of a neural morphism can be highlighted along with it and can contribute to

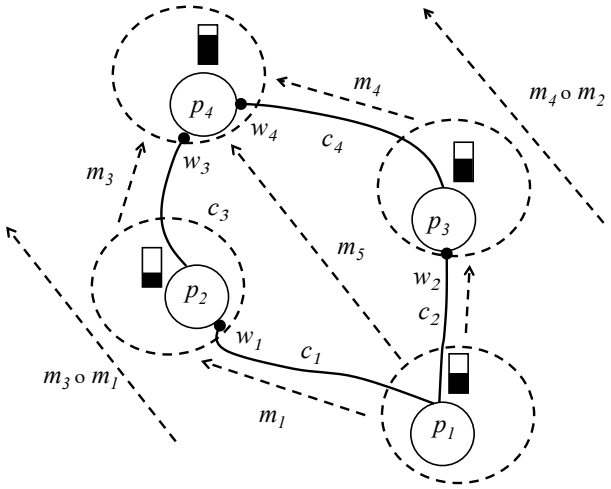


Figure 48: **A neural category diagram, along with the two compositions $m_3 \circ m_1$ and $m_4 \circ m_1$ that are implicit and the single morphism m_5 that the diagram defines provided that it commutes. Again the objects are binary, with their "on"-cell activity ranges simplified to the entire ranges of the cells.**

its existence by supplying either excitatory or inhibitory weighted inputs to its signal path objects as illustrated in Fig. 49. Even so, only the signal paths in the bundle between the domain and codomain are regarded as carriers of the morphism. The inputs from outside the signal paths may well be essential in enforcing the simultaneity of their activation instances, however. As mentioned earlier, this is an important consideration in network design.

5.4 Functors

Now that we have a method for constructing a neural category from any given neural network, how can we formalize the expression of its semantics? The mapping illustrated in Fig. 37 suggests a relationship between two categories: one for semantic expressions, the other for a neural network. The semantic category will be referred to eventually as a concept category, where concepts contain human-understandable semantic expressions that describe that which is represented in neural structure and processing.

But before we begin, why map from the semantic category to the neural category as suggested by Fig. 37 instead of mapping in the opposite direction, first selecting neural structures and mapping these to semantic descriptions? The latter strategy might seem more consistent with the piecemeal identification of a brain structure's semantics in a

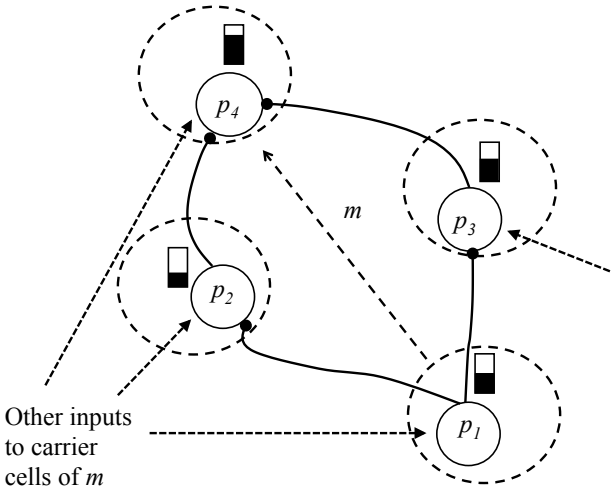


Figure 49: **A neural category diagram, along with the two compositions $m_3 \circ m_1$ and $m_4 \circ m_1$ that are implicit. Again the objects are binary "off" and "on" states.**

neuroscience investigation. An immediate answer to this question is that nothing in our approach precludes associating neural structure and activity to semantic expressions. Explorations can be conducted in either direction with a single functor regardless of the direction of the functorial mapping itself, much in the manner that functions on sets can be analyzed by exploring the set of elements in the domain that map to a given element in the codomain (i.e., inverse images). In any case, the important consideration in associating semantics with neural structure and activity is what the investigator can understand given the information available. The formalization (the functor), as with any mathematical model of physical phenomena, only aids the understanding by expressing it with precision to achieve greater depth and utility.

In fact, mapping objects and morphisms from the semantic category to the neural category has two advantages. First, it can be useful to have a multitude of functors mapping the same semantic expressions to different parts of the network. This allows the same semantic expressions to be associated with different regions of the network. This is an important consideration in expressing the unification, or coherence, of a brain or an artificial neural network for multisensor and/or sensor-actuator information fusion. The different regions for vision, hearing, somatic sensation and other senses, and also the association regions, planning, motor control and other functions, must operate in a unified fashion for a brain to work properly. Mapping from a common structure of semantic expressions into a neural category is a convenient analysis tool in seeking an understanding of this unity of function in biological systems and in designing artificial ones. We call the unity of function *knowledge coherence* (see for example [9]).

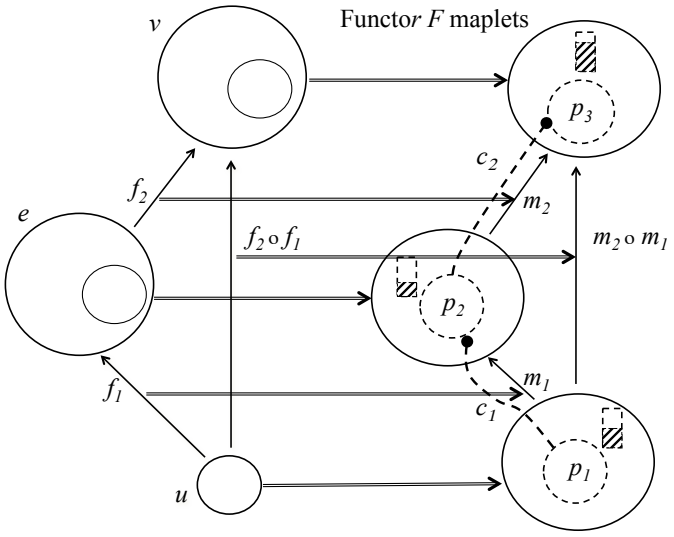


Figure 50: **Six of the maplets for a functor F:Fset $\longrightarrow \mathbf{N}_w$ are shown. The maplets are $u \mapsto p_1$, $e \mapsto p_2$, $v \mapsto p_3$, $f_1 \mapsto m_1$, $f_2 \mapsto m_2$, and $f_2 \circ f_1 \mapsto m_2 \circ m_1$. These map two semantic-expression morphisms, the functions f_1 and f_2, along with their composition $f_2 \circ f_1$ into the neural category. Again the neural objects carried by the cells are binary for this example.**

Functors provide a mathematical way to map between categories in a way that preserves composition (see Fig. 50), which implies that they preserve commutative diagrams. The commutativity of a neural diagram is given by the equivalence of morphisms defined by composition along parallel paths between pairs of objects. This is convenient because the defining diagrams of various categorical structures such as colimits and limits are commutative. This is of the utmost importance because it suggests that semantic expressions such as colimits and limits of pixel sets and functions can be mapped to similar structures within a neural category.

Let us proceed with this idea as applied to Agent R1.3. For illustration, we shall continue to use the category **FSet** in the formalization of the semantics of neural structure and activity. Also, we now have a method for defining a neural category $\mathbf{N_w}$ for a neural network with a given array $\mathbf{w}$ of connection weights. Commutative diagrams within $\mathbf{N_w}$ can express colimits and other structures which become highlighted as a consequence of neural activity. Mapping the semantic expression structures (diagrams of sets and functions in our present formulation, to form structures such as colimits) to connection-weighted neural structures, then, expresses the semantics of the potential activity occurring in the neural network at a specific stage of weight adaptation represented by $\mathbf{w}$.

5.5 Agent R1.3 and its Functors

If the connection weight array for Agent R1.3 is as required for the edge, angle, and triangle detector colimits and limits described in Part 1, a functor can map the diagams of sets and functions expressing these into the appropriate neural network structures. This is the mathematical basis for constructing the neural diagrams which were simulated to evaluate their effectiveness as described in earlier sections of this report. A functor $M\colon \mathbf{FSet} \longrightarrow \mathbf{N}_w$ maps each set A to a neural object (p, η) and each function $f\colon A \longrightarrow B$ to a neural morphism $m\colon (p, \eta) \longrightarrow (p', \eta')$. We can symbolize this using maplet arrows $(\mapsto)$ to show how individual objects and morphisms are mapped, for example $A \mapsto (p, \eta)$, $B \mapsto (p', \eta')$, and $f \mapsto m$ or, written out longhand, $(f\colon A \longrightarrow B) \mapsto (m\colon (p, \eta) \longrightarrow (p', \eta'))$. An example of the mapping of a composition by a functor is illustrated in Fig. 50 for a functor $F\colon \mathbf{FSet} \longrightarrow \mathbf{N}_w$ from a finite set category to a neural category. The functor's maplets for objects and morphisms are shown. Note that the example assumes the binary or "on-off" neural objects referred to earlier.

With this, all the mathematical tools are in place. However, the sets-and-functions semantic model is not using them to fullest effect to formalize the semantic expressions. It is still necessary to augment the objects and morphisms in **FSet** by including the input sums to the neural cells. It remains to complete the formalization by introducing a new category for formalizing the semantic expressions.

6 Semantic Expressions as Explanations

An effective formalization of neural network semantics requires a mathematical language for expressions that describe the meaning of neural structure (the cells and weighted connections of a network) and the processing—the activity and synaptic weight adaptation—that occurs in that structure. The language must provide for unambiguous descriptions of things and their relationships in the sensed environment of the network. To be most effective in describing a neural network as a cognitive system, it must also provide for an "inner world", the invented and introspective universe within which an organism projects possible futures, possibly contemplates its own capabilities, makes predictions, solves problems and concocts plans for future action. Humans have the full list of capabilities. In any organism, they depend ultimately upon the ability to extract information from data supplied by sensing.

Natural languages such as English could be used for semantic expressions were it not for their context dependence. Sentences in everyday discourse contain many ambiguities of expression unless they are spoken in a particular world context or appear within a descriptive discourse that adequately sets the context. An example of this is the sentence "Joseph carried it forward." . Who's Joseph? Carried what? What does it mean to "carry it forward"? Understanding what is said demands an agreed-upon real-world interpretation

of the terms used, either by (within) an individual mind or between minds engaged in discourse. Graphic illustrations are often worth many words, as an oft-repeated saying goes, and text combined with graphics is yet more powerful in communicating or thinking with a clear understanding. The fact remains, however, that a context-free symbolic language combined with agreed-upon interpretations of a set of basic symbols is the all-powerful type of formal language, since it allows an unambiguous interpretation in a wide variety of contexts. This is why scientists rely on mathematical models: The same familiar set of symbols and well-known operations upon them (e.g., numbers, arithmetic operations, algebraic equations, functions, differentials and integrals in calculus, etc.) can be used in a wide variety of scientific theories with the symbols interpreted appropriately in each. A famous example of a sentence in such a formal (mathematical) model is $E = mc^2$.

Explanatory mathematical descriptions of things can be expressed in logical theories. A theory is a closed system of propositions that can be manipulated according to certain rules of logic. The logical manipulations, such as deductive inference, operate upon propositions to form more complex propositions or to deduce or infer a proposition from a set of propositions. A simple example of logical inference is the syllogism "All men are mortal" and "socrates is a man", therefore "Socrates is mortal". A theory can be expressed in a compact form known as a *presentation* by listing its *axioms*. The axioms are the members of a set of assumptions from which all other propositions of the theory can be deduced. The theory is a closed system because only the axioms together with those propositions obtainable beginning with the axioms, by operations such as "and", "or", deductive inference (resulting in a "therefore" as in the syllogism example), and others are regarded as parts of the theory. Other examples of propositions are $E = mc^2$ and "All A's are B's". A statement such as "All men are mortal" is interpreted to mean that the set of humans is a subset of the set of all mortal beings. If theory T contains the axioms $E = mc^2$ and "m is the mass of an object at rest" and "c is the speed of light", then anything expressible by combining these three propositions or through deduction from them (not much in this case) is a valid proposition of the theory.

6.1 Interpretations

But although it would seem obvious, it is easy to overlook the important fact that the propositions forming a theory are meaningless without an agreed-upon meaning for the symbols E, m, c, A and B and mathematical operations and relations such as equality ($=$), multiplication and squaring (mc^2), sets and subsets and so forth. That is, the symbols and the operations upon them need an *interpretation*. In part, adding more propositions supplies this, for example "m is the mass of an object at rest" and "c is the speed of light" ($c = d_{photon}/T$ where d_{photon} is the distance a photon travels in a vacuum during some standard time interval T). This is to say that further explanations serve to enrich a theory, making it more specific while explaining some symbols' intended meanings by relating them to others. The operation of substitution in mathematical expressions is a familiar

example of the replacement of the symbols in a symbol-string with other symbol strings to make the original more specific according to a systematic procedure of interpretation. It results in a more complex, more detailed symbol string with a more specific interpretation, or semantics. However, as the above example shows, this does not fully resolve the interpretation because the symbols in the strings which are substituted for symbols in the original string also need an interpretation. Thus, mass (m) and the speed of light (c) can be defined through further propositions, but that merely pushes the interpretation problem back one step.

Logicians have formalized the notion of symbol interpretation to make the problem amenable to analysis, as follows. Ultimately, everything in a theory relates back to a set of undefined symbols called "primitives". For the theory to have meaning, then, its primitives need an interpretation, but an interpretation of the primitives must be of a different kind than simply adding more propositions. This brings to mind the natural-language need for "context". What is the context for $E = mc^2$, what kind of world of understanding are we talking about? This is the same problem as with the sentence "Joseph carried it forward." The answer involves a second kind of interpretation: a domain over which the theory is valid given an appropriate mapping of its primitive symbols to objects and relationships that exist within that domain. That is, a mathematical model for the domain of a theory is needed. Where we started out by regarding the theories containing propositions as our semantic expressions, we now need a domain which expresses, or *grounds*, the semantics. Going back to the syllogism example, the domain could be the set of all mortal beings (if it can be agreed what these are) together with all the distinguished subsets of mortals the theory refers to. The closest thing to a domain in the earlier discussion of pixel sets is the sets and functions themselves. More properly, the objects and relationships in the domain are the sets of stimulus patterns over the pixels and the relationships between these sets. These are represented by the activities of the neurons and synaptic or weighted connecitons in a neural network. Thus, in our formalization we propose logical theories and their domains of sensor stimuli together with relationships between the theories based upon mappings between their domains, and a mapping that relates all this to the structure and activity of a neural network.

The full formalization of theories and domains is called *categorical model theory*, which will not be discussed here. Other papers on the CNST do discuss this topic [9, 7]. Suffice it to say for now that categorical model theory is a part of the CNST.

6.2 Why theories?

If everything depends upon a context in the form of a mathematical semantic domain, why bother with theories? The important fact about a theory—a justification for taking the trouble to create and use it—is that it is a way of packaging propositions that work together logically to describe a world. While the symbols of the theory require an interpretation to relate to the world (universe, domain), the propositions and their logical relationships

are a mathematically precise, unambiguous way of encapsulating information about it in a human-readable form amenable to logical inference. Given the availability of a common understanding among analysts of a particular interpretation of its symbols, a theory is a way of resolving ambiguities and deducing properties of the things it describes by using the logical relationships it expresses. One important kind of theory is that which describes the fundamental processes underlying natural phenomena: once validated on sufficient data, it becomes a scientific theory, describing a domain such as that of particle physics. Knowing the speed of light and the mass of an electron and positron (all expressed in a consistent system of units), one can predict the energy released in the resulting matter-antimatter annihiliation when these two particles collide head-on. From this, one calculates the wavelength of the gamma-ray photons that are produced: $E = (m_{\text{electron}} + m_{\text{positron}})c^2 + E_K$, where E_K is the combined kinetic energy of the two particles, and $\lambda = (hc)/E$ where h is the Planck constant.

By now it ought to be clear that one theory can be more abstract, or less specific, than another. A theory that contains Einstein's famous equation and nothing else is the ultimate abstraction because it is really just an equation with no other information relating its symbols to anything. It might as well read $a = bc^2$, where a, b, c are just numerical quantities of some sort, and that is assuming that a theory of number is understood even though not specified (so that multiplication and squaring a quantity are interpretable operations). If we replace $a = bc^2$ with the even less specific $a = d$, the possible interpretations can be any two things that are identical: a and d are just two names for the same thing. For example, they can be two of the three functions mapping any one of the sets $D_{u_1}, D_{u_2}, D_{u_3}$ into the angle colimit set D_v as illustrated in Fig. 15. The usefulness of the equation $E = mc^2$, stated in isolation, stems from the fact that we all understand that it relates matter and energy and we have some idea what those things are; we supply the context. A more specific theory can be produced by combining $E = mc^2$ with the added equations $m = (m_{\text{electron}} + m_{\text{positron}})c^2$ and $\lambda = (hc)/E$; through the indicated substitution for m in the equation $E = mc^2$, the more-specific theory with the three equations can logically infer the annihiliation equation in the previous paragraph.

Regardless of how specific a theory is, however, a domain of interpretation is required if a full formalization is needed. To illustrate this point, a more specific theory than the one expressing the matter-antimatter annihiliation can be obtained by adding the propositions " m is the mass of an object at rest", $c = d_{photon}/T$, and "d_{photon} is the distance a photon travels in a vacuum during some standard time interval T". This is more specific since with the addition of these propositions more is known about what the original symbols mean. But now the notions of "mass", "photon", "distance" and "time" and the state of "rest" or "zero motion" require interpretation. If we settle upon these notions as primitives for the theory, then to be fully formal the items that these notions represent in a domain for the theory must be identified.

6.3 An adjacency relation for sensor elements

Can everything be interpreted in terms of sets of pixels and operations upon them? Expressing everything with sets of pixels is difficult, as mentioned earlier. In any sensor modality—vision, auditory, smell, touch, or others such as magnetic sensing—there are sensor elements of some kind through which elementary parts of that modality register the apropriate stimuli for input to a neural network. As a convenience, then, we generalize the use of the word "pixels" to mean "sensor elements". In the section following, we shall address the "pixel set problem": How do we transition in our theory expressions from pixel sets to meaningful objects such as the visually-sensed edges, angles and triangles, or to sounds, scents, pressure, etc.?

An answer lies in the diagrams introduced earlier for colimits and limits. These utilized the u-cells as the most basic cells, receiving excitatory stimulus from the pixels; through the u-cells, the neural network begins organizing its inputs. The u-cell RFs, or "u-sets", group the pixels into local stimulus regions (see Fig. 51) which can be represented by their activities. A pixel may receive stimuli from a wide region in, say, visual space and thereby confuse the many objects providing the stimuli. Moreover, the only useful information in this "smearing" of stimuli, the adjacency of stimulus features detected by adjacent pixels, is inaccessible to the network without help from its own organizing of its inputs. The u-cells provide this organizational function. The stimulus response of a pixel is blended with that of the other pixels in any u-cell RFs that contain it. This has the effect of constraining the pixels' many different stimuli so that a u-cell becomes active only when its RF lies largely within the area receiving illumination from an object. In this way, the u-set represents a small region of the space of stimuli. It is as if the pixels were voting upon the significance of stimuli in their region and a sufficient plurality—the sum of their inputs exceeding the threshold of their representing cell—would result in a "Yes" vote. Other sensory modalities' stimulus regions are dealt with in a similar fashion. A variation on this theme uses cells with inhibitory as well as excitatory input to provide local contrasts, with a cell becoming active only if its excitatory inputs outweigh the net inhibition. Here, this function has been implemented by the edge-detectors defined earlier which receive their input from u-cells. In any case, a u-set is appropriately called an influence set because it represents the cumulative influence of stimuli of sufficient magnitude to activate its u-cell. Because of this and the fact that the u-sets are small and effectively cover the pixel array, they serve as primitives, basic elements in the formal language of the formalization of the semantics of a neural network.

The neural network's semantic expression is dependent upon its connectivity and activity in response to the stimuli. Therefore, if there are intrinsic relationships among its pixels, its connectivity must capture these at the input level. In a vision sensor, adjacency between pairs of pixels is such a relationship because it is the most basic organization of space. In this type of sensor, we can assume that each pair of u-cell RFs, or u-sets, that lie next to each other in the pixel array have a non-empty intersection. This allows a neural network to use its u-cells to represent a stimulus as a spatial entity. In line with this, we

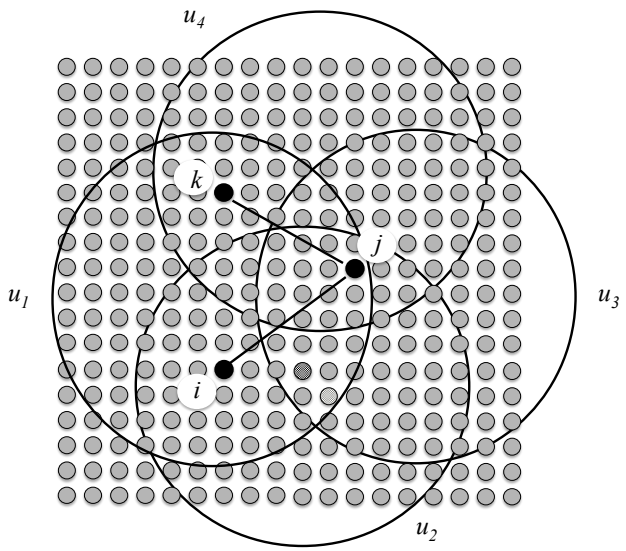


Figure 51: **Every open set containing pixel i contains pixel j, so they are adjacent. The same is true of pixels k and j. However, the same is not true of i and k.**

define an adjacency relation on the pixels as follows.

If the smallest u-set intersection containing a pixel i also contains a pixel j, or conversely, we say that i and j are *adjacent*, denoted $i \sim j$. Note that pixels in a u-set that do not lie in an intersection with another u-set fit this definition because a set is the same as its intersection with itself, $S = S \cap S$. Hence, $i \sim i$ for all pixels i. Three examples in which $i \neq j$ can be seen with the three pixels i, j, k which are labelled in Fig. 51. First, notice that the smallest intersection containing j is that of all four u-sets shown, $j \in D_{u_1} \cap D_{u_2} \cap D_{u_3} \cap D_{u_4}$, but this set does not contain i. However, the smallest intersection containing i is $D_{u_1} \cap D_{u_2}$, which does contain j and therefore $i \sim j$. Similarly, although as before $j \in D_{u_1} \cap D_{u_2} \cap D_{u_3} \cap D_{u_4}$, the smallest intersection containing k is $D_{u_1} \cap D_{u_4}$, which also contains j and therefore $k \sim j$. Since we have defined two points adjacent if the smallest intersection containing one of them contains the other, we can make the symmetric statements $j \sim i$ and $j \sim k$. On the other hand, since $i \in D_{u_2}$ but $k \notin D_{u_2}$ and $k \in D_{u_4}$ but $i \notin D_{u_4}$, we must conclude that $i \not\sim k$. Since a u-set has a nonempty intersection with its closest 4 neighbors (or 2 or 3 neighbors at the boundary of the pixel array), it is always possible to trace a curve through adjacent pixels as far as desired. Most importantly, the neural network has the informaton necessary for such a trace through its u-cells, which can be further grouped in hierarchical fashion by edge-detectors and other connectionist structures.

In vision, the spatial adjacency relation is a primitive relation for the elements of the u-sets, the pixel or sensing elements. Because they sample different modalities, the pixels

in other sensors may or may not be arranged in proximity as they are in a visual sensor. In an auditory sensor such as the mammalian cochlea they are so arranged: The pixels are hair cells whose proximity corresponds to similarity of sound frequency. In this case, the ordering of frequencies lies in a 1-dimensional as opposed to a 2-dimensional space and the u-cells exist as 1-dimensional intervals with nonempty intersections where they lie next to one another in the ordering along the cochlea. A sensor modality with an ordering such as vision or auditory is often referred to as "topographic". Not all modalities are of this kind; for example, for the olfactory modality the spatial adjacency of its sensing elements has no obvious meaning. Yet in sensors of this kind with no obvious continuity of spatial ordering, if there is any order at all in the domain of that modality u-cells can be defined that group together its sensor pixels. The grouping of sensing elements by their input to u-cells defines a similarity relationship between the types of items sensed. Thus in the olfactory modality the different molecules of various chemicals are sensed by cells having receptors for those molecules, and the similarities in receptor type can be taken as an "adjacency" relation. In such modalities the domain may be quite complex and the spatial notion of dimensionality may not be efficaceous.

Adjacency, or similarity in some sense, is fundamental in beginning an analysis of stimuli. How this and the pixel sets and functions are incorporated in the concept model is the topic of the next section.

6.4 From Stimulus to Theory

Sets and functions will now be formalized as types of entities that are more abstract. The actual sets and functions will be contained in the interpretations of theories. Interpreting theories is the subject matter of model theory, which will not be discussed here. Instead, in discussing a theory we shall often refer to its interpretation in terms of sets, functions and relationships between these entities. One of the assumptions implicit in all that follows is that every theory contains the entities necessary for the Boolean operations "and", "or", etc., relations "implies", "is equivalent to", etc., and any necessary numerical expressions such as $u = ax + 1; xy \leq 3.5; y = f(x)$, etc. To start, there is a theory for a sensor K:

```
Theory TSensorK
    contains TBool, TList
    sort SensorK
    op Kloc1:  SensorK --> Integer
    op Kloc2:  SensorK --> Integer
    op Kpixval:  SensorK --> Real
end
```

Here and in all that follows, the symbol `T` begins the name of a theory. A name for the sensor can be substituted for the symbol `K: Visual` or just `Vis`, or `V` for a vision

system, for example. The expression `contains TBoolean, TList` states that the theory `TSensorK` contains two theories which are very basic and will not be defined in detail here: A theory `TBool` of Boolean quantities (such as truth values `true` and `false`) and a theory `TList` of lists which has statements that express operations upon lists of items. In turn, `TList` contains a theory of numbers, `TNum`. More will be said of these presently; use of the `contains` statement indicates that the contents of the named thories are present without having to reproduce them in the theory at hand. The expression `sort SensorK` introduces the most basic sort, or type, of the quantities described in the theory `TSensorK`. It is an expression for the sensor K itself; in an interpretation of the theory, it can be interpreted as the set of all of sensor K's sensing elements, or pixels. The statement `op Kpixval : SensorK - - > Real` declares that `Kpixval` is an operation that maps entities of type `SensorK` (the pixels) to entities of type `Real` (the current numerical values of the stimulus-generated activities of the pixels in any instance of network input activity). Operations in a theory are single-valued just as is the case with functions between sets, so it is already assumed that an operation maps a symbol of one type to a single, unique symbol of another type. The operations `Kloc1` and `Kloc2` have been included in case it is desired to formally include pixel array position in the two-dimensional pixel array of visual sensor elements. There will be a further comment on whether this is actually needed, but more explanation is required first.

Each sensor element of sensor K has its own theory:

```
Theory TK[k]
    contains TSensorK
    const K[k]:  SensorK
    Axiom K[k]-location is
        (Kloc1 (K[k])  =  [j])  and  (Kloc2 (K[k])  =  [l])
end
```

As was the case with `contains TBool, TList`, the statement `contains TSensorK` indicates that the theory `TK[k]` includes the theory `TSensorK` (and, hence, any theories it includes, in this case the theory `TBool` of Boolean quantities such as truth values `true` and `false`, numbers `TNum`, and lists `TList`). Therefore, as with the other theories, the statements of theory `TSensorK` need not be listed, for they are assumed present. The symbol string `TK[k]` indicates that the theory includes a designated sensor element, or pixel, `K[k]` of sensor `K`. The symbol for the pixel `K[k]` includes a positive integer k, denoted by the symbol `[k]` in brackets. It serves as the unique identity of a particular pixel. Heretofore, a symbol in brackets in a theory always denotes a positive integer that for the purpose of illustration is represented by a Roman letter.

The expression `const K[k] : SensorK` of theory `TK[k]` decrees that the symbol `K[k]` is a constant of type `SensorK` (in an interpretation, it would be a specific pixel), just as the theory `TBool` declares that the symbols `true` and `false` are constants of type `Boolean`. In case it is desired by the analyst, `TK[k]` includes the pixel location operations

together with an axiom which defines pixel K[k]'s location coordinates Kloc1(K[k]) and Kloc2(K[k]) as a pair of nonnegative integers. In that case, the integers Kloc1(K[k]) and Kloc2(K[k]) must lie within the range of row and column indices of the sensor pixel array. In sensors other than visual, it may happen that the sensor elements do not follow a two-dimensional layout; in such a case it might be that only the operation Kloc1, or perhaps operations in addition to Kloc1, Kloc2 such as Kloc3 would apply. It may in fact be the case that position coordinates for a sensor do not follow a natural order and only the integer value [k] identifies the pixel K[k] uniquely in the sensor array. The u-sets, the basis for defining the adjacency relation $\sim$, are probably the best way to incorporate ordering or other pixel relations in the theories.

The theory TBool is needed as a basis for Boolean expressions and for expressing the truth values true and false, which are constants of a sort Boolean. A constant of any sort is an entity that plays a special role in a theory. The two truth value constants are declared within the theory TBool with the expressions const true : Boolean and const false : Boolean. These two constants are often used in a Boolean equation to specify that an expression that forms a complete sentence—that is, a proposition—is to be regarded as true or false. The theory TBool also contains the familiar Boolean operations and, or and not for combining or modifying expressions. For example, TBool contains statements of the form op and : Boolean, Boolean $-->$ Boolean; this defines and to be a binary operation on Booleans that combines a pair of Boolean expressions, joining them with an and symbol to form a compound expression. Finally, this theory has axioms which state the properties of the various sorts and their operations. More will be said of the axioms presently.

In similar fashion to TBool, TList contains the sorts, operations, constants and axioms necessary to form lists of items and perform operations upon them. A key operation is list, which forms a list given its contents; for example, list a b c yields the list (a b c). Another is maplist, which applies an operation with a single argument over a list, yielding a list of operation results. For example, maplist f (a b c) yields the list (f(a) f(b) f(c)). The final operation we need discuss is reduce, which applies a binary operation recursively to a list to obtain a single item. For example, reduce + (1 2 3) yields the single integer 6, the sum of 1, 2 and 3. Note that these operations are similar to, but not the same as, the Lisp operations with the same names. Aside from the fact that the notation differs—for example, we do not have to quote quantities to signify literals as in reduce‘ + ‘(1 2 3)—Lisp has higher-order types and is based upon the λ-calculus, in which everything is a function. In the first-order theories we use, list operations must be defined recursively in the theory TList and the logic must allow for this.

Note that in the example reduce + (1 2 3), TLisp assumes the availability of a binary numerical operation, +. This requires that TLisp contain a theory TNum which contains the sorts, operations, constants (such as 1 and 0), and axioms necessary to perform numerical operations. The theory TNum contains sorts such as Complex, Real, Integer, PosReal, NonnegReal (for the nonnegative real numbers, which is PosReal augmented by the

number 0), PosInteger, etc. for the various kinds of numbers such as complex, real, and integer, and special sorts for positive integers, etc. Having all these sorts simplifies the expression of numbers with the appropriate ranges of values for a particular application. Finally, TNum also contains all the required numerical operations ($+$, $-$, $\cdot$) and relations ($=$, le for $=, \leq$, etc.) and axioms.

Statements derivable from the axioms by deduction are called *theorems* of the theory—these, like the axioms, are always true within the theory. The axioms and theorems together are often collectively called "theorems", since a theory can often be expressed by interchanging axioms with proved theorems. Because all of the truths of the theory can be derived from them, the use of axioms allows a compact expression of it. As a practical matter, a properly-chosen set of axioms clarify the important facts about what is assumed in the theory, such as the properties of the operations it contains. The axiom specifying the location coordinates associated with the pixel K[k] is one example. Another is an axiom for Booleans expressing the fact that for propositions A and B, (A and B) implies A (in any instance in which the proposition A and B is true, it must be that A is true). This is one of the axioms that defines the properties of the and operation. Another axiom states that (A and B) iff (B and A), where iff means "if and only if", so we can infer from these two axioms and the rules for logical inference that (A and B) implies B. The two axioms appear in the theory TBool as follows:

```
Axiom Dropping-and-rule is
    forall  (A:Boolean, B:Boolean)
    (A  and  B)  implies  A
Axiom and-commutes is
    forall  (A:Boolean, B:Boolean)
    (A  and  B)  iff  (B  and A)
```

There is a theory $TK[k]$ for each pixel $K[k]$ in the sensor array. In an interpretation in sets, SensorK maps to the set of pixels, Integer maps to the set $\mathbf{Z}$ of integers, and Kloc1 and Kloc2, if included, would map to functions which together assign an ordered pair of integers to each pixel, expressing its location in the array. Other quantities in a theory are interpreted in like fashion.

6.5 Resolving issues in expressing theories

A theory to represent a subset S of pixels defines a sort each of whose members corresponds to a unique member of the sort SensorK. If the subset is a u-set we might denote its sort in the theory representing it by U[i], where [i] represents a positive integer acting as an index, as in the case of the index $[k]$ for the pixel $K[k]$. But note that this poses a possible dilemma, for now the pixel symbols have two different types, x : U[i] and

x : SensorK. Yet, the theories are supposed to be carefully typed. If a symbol is given two distinct types, one representing a set and the other one of its subsets, this could introduce an ambiguity. Therefore, in place of the notion that an element of a subset is also an element of its superset, strict formality demands that the theory include an explicit correspondence between a symbol of type U[i] and a symbol of type SensorK that in an interpretation would be the same element in the larger set. For this purpose, there is an operation op U[i]K : U[i] − − > SensorK accompanied by the following axiom:

```
Axiom U[i]-members-unique is
forall  (x:U[i], y:U[i], z:SensorK)
    ((U[i]K (x)  =  z)  and  (U[i]K (y)  =  z))  implies  (x  =  y)
```

This axiom states that if two symbols x and y of type U[i] map to the same symbol z of type SensorK, then x and y are just two names for the same thing. That is, operation U[i]K is a one-to-one correspondence and, since operations in a theory are single-valued, the operation U[i]K is a one-to-one symbol map. This use of one-to-one mappings is an abstract way of expressing the subset relation: in an interpretation, U[i]K is interpreted as a function $f_i : u_i \longrightarrow K$ mapping each element of u_i to itself as an element of the set K of sensor pixels, that is, U[i]K is a way of expressing the subset relation, $u_i \subseteq K$.

This use of different symbols x and y of one type to represent the same element in two sets is meant to distinguish between symbols of different types (sorts). It is a common practice to simplify notation by overloading symbols, however, and we could have just used the same symbol x with two different types, as in K[k] : U[i] and K[k] : SensorK, and hide the operation U[i]K. The operation U[i]K made it clear that *every* symbol of type U[i] represents a unique member of the sort SensorK. Since that operation is now hidden, we use a new notation, the *subsort* relation, to clarify that every symbol of type U[i] corresponds to the same unique symbol with type x : SensorK:

```
subsort  U[i]->SensorK
```

which can be thought of as a theory "macro" that hides the axiom U[i] − members − unique. This is in the same spirit as the hiding of a theory, such as TBool, TNum or FSet.

Building on the notion that for each x : U[i] there is a member x : SensorK, the expression op U[i]K : U[i] − − > SensorK takes on an added significance as part of the hidden statements of a theory. Since the sorts x : SensorK and x : U[i] are to be interpreted as a set and one of its subsets, respectively, it must be the case that every property of x : SensorK is also a property of x : U[i] since in an interpretation they are the same element. Properties of the members of a sort can be represented by operations, such as op Kloc1 : SensorK − − > Integer. In this case, each member x : U[i] must share the location of its corresponding member x : SensorK. But this requirement is satisfied, since in reality

$$\texttt{Kloc1}(\texttt{x} : \texttt{U[i]}) \;=\; \texttt{Kloc1}(\texttt{U[i]K}(\texttt{x} : \texttt{U[i]})) \;=\; \texttt{Kloc1}(\texttt{x} : \texttt{SensorK})$$

and similarly for all other properties of the members of sort `SensorK`.

Hereafter we shall use the `contains` statement only to refer to included theories other than `TBool`, `TNum` and `TList` and just assume that these theories are included in every other theory dealt with here. Hiding the operations `U[i]K` and using the statement `subsort U[i]->SensorK` instead shows how we can substantially shorten the statement of theories. So, now we have three simplifications we can make in expressing theories:

1. Cease using the `contains TBoolean, Tlist` statement and just assume that those two theories are always present.
2. For any subsort `Y` of a sort `X`, include the statement `subsort Y->X` and hide the operation `op YX : Y -> X` and accompanying axiom

```
Axiom Y-members-unique is
forall  (x:Y, y:Y, z:X)
    ((YX(x)  =  z)  and  (YX(y)  =  z))  implies  (x  =  y)
```

   The hidden operation and its axiom will be assumed to be present in the theory.
3. For any property P of the sort `X`, expressed as an operation `op P : X -> W` where `W` is a different sort whose members can be used as values for the property, the subsort `Y` automatically inherits P via the hidden statement `op YX : Y -> X`.

A theory for a u-cell u_i can now be stated:

```
Theory TU[i]
    contains TSensorK
    subsort U[i]->SensorK
    const ulist[i]:List
    const uact[i]:Real
    const uthreshold[i]:Real
    const ubool[i]:Boolean
```

Include copies of the pixel theories
`TK[j1], ... , TK[jn]` representing the pixels
$x_{j_1}, \ldots, x_{j_n}$ of D_{u_i}
but DELETE from these copies the statement
`contains TSensorK`
to avoid multiple duplications.

Next, assert that the activity θ_{u_i} of cell u_i
is the summation over the list `ulist[i]` of its pixels,
where the pixel constants are given in the theories

```
    TK[j1], ...  , TK[jn] :
    ------------------------------------------------
    Axiom U[i]-summation is
        (ulist[i] = (list K[j1]  ...   K[jn]))  and
        (uact[i] = reduce + (maplist Kpixval ulist[i]))
    ------------------------------------------------
```

Finally, assert that the `U[i]` predicate value `ubool[i]` is `true` if and only if the `U[i]`-summation exceeds a threshold. Reminder: An expression of the form `(a > b)`, `(a = b)`, `(a < b)`, etc., is a predicate; i.e., it is of type `Boolean`:

```
    ------------------------------------------------
    Axiom U[i]-predicate-has-value is
        ubool[i] = (uact[i] > uthreshold[i])
end
```

The subsort `U[i]` of `SensorK` formally represents the u-set u_i as a subset of the sensor array K. The subsort statement indicates that each item of the subsort `U[i]` maps to a unique item of the sort `SensorK`. The `uact[i] : Real` constant is the summation over the activities of the u-cell's pixels. The summation is obtained as the `reduce` operation applying the addition operator $+$ to the list of pixel activity values (stimulus values) over D_{u_i}. The pixel activity values are the application of the `Kpixval` operation over the list `ulist[i] : List` of the pixel constants of type `U[i]` via the `maplist` operation. The expression $(\texttt{uact[i]} > \texttt{uthreshold[i]})$ is a predicate since it yields a Boolean value. It yields the Boolean value `true` for the Boolean constant `ubool[i] : Boolean` if and only if the `uact[i] : Real` sum exceeds a threshold value `uthreshold[i] : Real`. The subsort `U[i]` represents an object in the visual space and the Boolean constant `ubool[i]` indicates whether or not it is present given the current activity in the neural network (in this case, the object's presence is determined solely by the aggregate activity of its pixels). This scheme is typical of the theories; a new sort (here a subsort) and predicate appear with each new theory, beginning with the theories `U[i]`.

Finally, notice that the inclusion of the theories $\texttt{TK[j1]}, ..., \texttt{TK[jn]}$, hence of the pixels $\texttt{K[j1]}, ..., \texttt{K[jn]}$. This suggests the formalization of the relation $\sim$ defined earlier. However, this depends upon there being theories that group each u-set with its immediate neighbors and identify the pixels that lie in the intersections. By definition, $\texttt{K[i]} \sim \texttt{K[j]}$ would be valid only if the smallest intersection containing one of them contained the other. The formal use of the symbol $\sim$ will not be included here; instead, adjacency will be implicit in the contents of the theories.

Equipped with this formalization of a sensor and the most basic groupings of its pixels represented in the neural network by the u-cells and in the theories as objects `U[i]`, how do we now formalize the partition of an edge detector's RF into positive and negative inputs to the e-cell, which are then added algebraically? A theory for this is as follows. Let $\texttt{TU[j1]}, ..., \texttt{TU[jm]}$ be theories for m u-sets u_j whose union forms D_e^+, where $D_e^+ =$

$\cup_{j=j_1}^{j_m} u_j$: these are the RFs of the u-cells providing excitatory input to e (such as u_1 and u_2 in Fig. 14). Similarly, let TU[k1],...,TU[kn] be theories for n u-sets u_k whose union forms D_e^-, with $D_e^- = \cup_{k=k_1}^{k_n} u_k$; these are the RFs of the u-cells providing inhibitory input to e (such as u_3 in Fig. 14). Then, a theory for the edge-detector e is

```
Theory Te[i]
    contains TSensorK
    sort E[i]
```

Include copies of the theories `TU[j1], ..., TU[jm]`,
where $u_{j1}, \ldots, u_{jm}$ have excitatory inputs to e_i,
and the theories `TU[k1], ..., TU[kn]`,
where $u_{k1}, \ldots, u_{kn}$ have inhibitory inputs to e_i,
but DELETE from these copies the statements

```
sort SensorK,
op Kloc1:  SensorK --> Integer
op Kloc2:  SensorK --> Integer
op Kpixval:  SensorK --> Real
```

to eliminate duplication, and also delete the statements

```
subsort U[j1] -> SensorK, ..., U[jm] -> SensorK,
subsort U[k1] -> SensorK, ..., U[kn] -> SensorK,
```

and because the U[... subsorts are being replaced by `U[...]+` and `U[...]-` subsorts.

```
subsort U[j1]+ -> SensorK, ..., U[jm]+ -> SensorK,
subsort U[k1]- -> SensorK, ..., U[kn]- -> SensorK,
const elist[i]+:List
const elist[i]-:List
const eact[i]:Real
const ethreshold[i]:Real
const ebool[i]:Boolean
```

Next express two lists of constants representing the input
weights to the e-cell. In an interpretation, the weights
will be given by the current state of the neural network.

```
const welist[i]+:List
const w[j1]:Real
...
const w[jm]:Real
const welist[i]-:List
const w[k1]:Real
...
```

```
    const w[kn]:Boolean
    Axiom e[i]-weights is
        (welist[i]+ = (list w[j1]  ...   w[jn]))  and
        (welist[i]- = (list w[k1]  ...   w[kn]))
```

Assert that the activity θ_{e_i} of cell e_i
is the summation over the weighted list of excitatory inputs minus
the summation over the weighted list of inhibitory inputs
from the appropriate u-cells as expressed in the
theories `TU[j1], ..., TU[jm]` and `TU[k1], ..., TU[kn].`
`vinnerprod list1 list2` multiplies two lists of numerical values
element-by-element and sums the results to get the inner product:

```
    Axiom e[i]-summations is
        (elist[i]+ = (list uact[j1]  ...   uact[jn]))  and
        (elist[i]- = (list uact[k1]  ...   uact[kn]))  and
        (eact[i] = (vinnerprod welist[i]+ elist[i]+)
        - (vinnerprod welist[i]- elist[i]-))
```

Finally, assert that the `U[i]` predicate value `ebool[i]` is
`true` if and only if the summation exceeds a threshold:

```
    Axiom e[i]-predicate-has-value is
        ebool[i] = (eact[i] > ethreshold[i])
end
```

In the sets-and-functions model of neural semantics there are input sums to the u-cells

$$\theta_{u_i} = \Sigma_{D_{u_i}} s_j \ ,$$

where the s_j are pixel stimulus values, and input sums to the e-cells,

$$\theta_{e_i} = \Sigma_{j \in D_{e_i^+}} w_j s_j - \Sigma_{j \in D_{e_i^-}} w_j s_j \ ,$$

where the stimulus values are accompanied by connection weights w_j. This scheme of expressing summations is an adjunct to the sets-and-functions model and carries through to the colimits and limits derived from the u- and u-cells, such as the triangle detectors. In the theories, on the other hand, the important items are the objects represented, such as the `U[j]` and `E[i]` sorts, triangle-detector sorts represented similarly in the triangle theories, and so on for other colimit and limit objects. The input sums to the corresponding cells can be represented in the theories as illustrated by the examples; their importance there is solely to show the relationships of theory items inherited from the base diagrams of colimits and limits to the object defined in the colimit or limit theory. They indicate when the

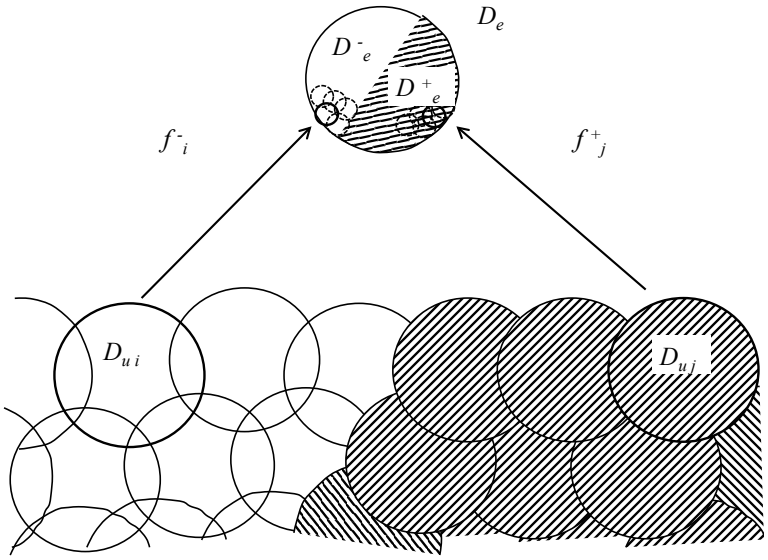


Figure 52: **The bottom half of this schematic picture shows the receptive fields D_{u_i} and D_{u_j} of two u-cells u_i and u_j which, as with all u-cells, receive excitatory input from their pixels. The arrows labelled f_i^- and f_j^+ illustrate the mappings of the two RFs into the inhibitory and excitatory fields, respectively, of the RF D_e of the edge-detector cell e, a consequence of the inhibitory and excitatory connections from u_i and u_j to e.**

object is present in the input space based upon the presence of its constituent base diagram objects, and the predicate values `ubool[j]`, `ebool[i]`, etc. summarize this information.

Recalling the set-theoretic model of edge detectors built from u-cell inputs in Section 5, Fig. 52 illustrates the following discussion of the u-cell excitatory and inhibitory inputs to the e-cell. We assume that the u-cell indices in D^+_{uj} and $D^-_{u_j}$ are selected so that the cells' RFs $D_{u_j}(j \in J^+)$ and $D_{u_j}(j \in J^-)$ are separated along a more-or-less straight line, so that their RFs actually do form D^+_e and D^-_e as $D^+_e = \cup_{j \in J^+} u_j$ and $D^-_e = \cup_{j \in J^-} u_j$. This can be defined formally in terms of the adjacency relation $\sim$. As mentioned, however, the chain of $\sim$ relationships will not be shown explicitly in the formalization. Implicitly, this chain forms the actual boundary between D^+_e (the shaded region) and D^-_e. Now, an edge can be defined as the algebraic sum of RFs depicted in the theory `Te[i]`, where the RFs $D_{u_j}(j \in J^+)$ and $D_{u_j}(j \in J^-)$ lie on either side of a line joining pixels $b_{i_1}, bi2, \ldots b_{i_n}$ such that $b_{i_1} \sim b_{i_2}$, $b_{i_2} \sim b_{i_3}$ (where b_{i_2} lies in the overlap of two u-cell RFs, so that it shares an RF with both b_{i_1} and b_{i_3}), $b_{i_3} \sim b_{i_4}$ (where b_{i_3} again lies in the overlap of two u-cell RFs) and so forth until the line reaches b_{i_n} via $b_{i_{n-1}} \sim b_{i_n}$. Expressing in a theory that the line is a straight line would require more description. As with $\sim$, we leave the straightness property as implicit for now and regard it as the task of the neural network

designer or the biological processes through which a neural vision system forms to ensure this. In any case, we do not regard it as necessary that the line be perfectly straight. The important requirement is that b_{i_1} and b_{i_n} be maximally separated by a chain of adjacency relations with no crossover points in the chain.

6.6 Theory morphisms

If the theories are to be interpreted as sets, something is needed to interpret to the functions between sets. In fact, as we have have seen, the theory interpretations are not just sets: they are systems of sets which interpret the sorts together with functions upon the sets which interpret the operations upon the sorts and designated elements of the sets which interpret the constants. This brings us to the relationships between the theories, which completes the formalism for the purpose of this paper.

We shall exemplify theory morphisms by describing a typical relationship between a pixel theory and a u-cell theory, and then between the u-cell theory and the e-cell theory from the above example. First, the morphism whose domain is pixel theory `TK[k]` and whose codomain is theory `TU[i]` representing the neural connection from pixel K_k to the u-cell u_i is

```
Morphism TK[k]-to-TU[i] is
    sort Boolean --> sort Boolean
    sort Integer --> sort Integer
    sort List --> sort List
```

This continues, mapping the other sorts from the theories `TBool`, `TNum` and `TList` to themselves in `TU[i]`

```
(op and:Boolean,Boolean --> Boolean) -->
    (op and:Boolean,Boolean --> Boolean)
(op or:Boolean,Boolean --> Boolean) -->
    (op or:Boolean,Boolean --> Boolean)
```

This continues, mapping the operations from the theories `TBool`, `TNum` and `TList` to themselves in `TU[i]`

```
const true:Boolean --> const true:Boolean
const false:Boolean --> const false:Boolean
```

This continues, mapping the constants from the theories `TBool`, `TNum` and `TList` to themselves in `TU[i]`

```
    Axiom and-commutes --> Axiom and-commutes
    Axiom and-is-associative --> Axiom and-and-is-associative
```

... and this continues, mapping the axioms from the theories `TBool`, `TNum` and `TList` to themselves in `TU[i]`

```
    sort SensorK --> sort SensorK
    (op Kloc1:  SensorK --> Integer) --> (op Kloc1:  SensorK --> Integer)
    (op Kloc2:  SensorK --> Integer) --> (op Kloc2:  SensorK --> Integer)
    (op Kpixval:  SensorK --> Real) --> (op Kpixval:  SensorK --> Real)
    const K[k]:  SensorK --> const K[k]:  SensorK
    Axiom K[k]-location --> Axiom K[k]-location
end
```

This shows in detail the expression of a theory morphism. However, as withe the contents of included theories via the `contains` statement, theory morphisms can be simplified by assuming that items that appear unchanged in both the domain and codomain of a morphism are mapped to themselves. In fact, that is true of the entire contents of the morphism $\texttt{TK[k]} - \texttt{to} - \texttt{TU[i]}$. So, instead of having to undergo a lengthy writing process, the assumption concerning unchanged items can be applied to write this morphism as, simply,

```
Morphism TK[k]-to-TU[i]
end
```

As for the morphism $\texttt{TU[j]} - \texttt{to} - \texttt{Te[i]}$, most items are unchanged as with the previous example. However, note that the item $\texttt{subsortU[j]} - > \texttt{SensorK}$ must be mapped to $\texttt{subsortU[j]} + - > \texttt{SensorK}$ if $[\texttt{j}]$ is one of $[\texttt{j1}], ..., [\texttt{jm}]$, and to $\texttt{subsortU[j]} - - > \texttt{SensorK}$ if $[\texttt{j}]$ is one of $[\texttt{k1}], ..., [\texttt{kn}]$. Therefore, the assumption concerning unchanged items can be applied to write the morphism as either

```
Morphism TU[j]-to-Te[i] is
    subsort U[j] -> SensorK --> subsort U[j]+ -> SensorK
end
```

or

```
Morphism TU[j]-to-Te[i] is
    subsort U[j] -> SensorK --> subsort U[j]- -> SensorK .
end
```

Given morphism like this, we can form compositions like $\texttt{TK[k]} - \texttt{to} - \texttt{Te[i]}$, as

$$\texttt{TK[k]} - \texttt{to} - \texttt{Te[i]} = (\texttt{TU[j]} - \texttt{to} - \texttt{Te[i]}) \circ (\texttt{TK[k]} - \texttt{to} - \texttt{TU[j]}) .$$

With this formalism, diagrams corresponding to the diagrams of sets and functions at the beginning of this paper can be formed and limits and colimits derived.

7 Conclusion

We have presented a neural network semantic model using a simple example at the basic stimulus level to illustrate it. The model is the Categorical Neural Semantic Theory (CNST), a mathematical model based in category theory. By first formulating a system of semantic expressions starting with visual sensor stimuli in terms of a category of sets and functions,we showed that the CNST can be applied to improving the design of neural networks. Initially, the model was expressed in terms of pixel sets, and functions which served as relationships between the sets. Using categorical constructs called colimits and limits, a visual-sensing neural network design which is effective in recognizing a type of geometric object was derived. In fact, the neural network design represents the objects in a *parsed* format wherein an object is expressed not only holistically, but also broken down into its constituent features together with the information for joining them to form the represented object.

Although the pixel set model provides a simple, illustrative example and shows that the neural network functions as desired, the set-and-function formalism is regarded as too simplistic. Hopefully, it is now clear to the reader why this was the case. The sets and simple inclusion functions used required the accompanying information about the excitatory and inhibitory connections between cells and cell input summations that expressed the resulting effects upon the target cells of the connections. An improved model uses a category of theories and theory morphisms to represent concepts and relationships between the concepts that express the semantics of the neural network in terms of its input stimuli. The theories express object representations in a human-understandable, formal language in place of the pixel sets. The theory morphisms the inheritance of one theory by another, possibly with some modifications while maintaining the validity of the inherited theory. Together, theories and theory morphisms incorporate all the information required to understand the semantics of the neural network.

Colimits and limits in the theory category can be mapped to a neural category that expresses the connectionist structure and stimulus-generated activity of a neural network. Functors, the structure-preserving mappings of category theory, provide this association of theories and morphisms with neural structure and activity. Properly designed in concert with the theory colimits and limits, the neural network yields a neural category whose diagrams represent parsed object representations, as did the sets-and-functions model, but with a comprehensive formalism. In addition, we propose that the CNST can be a useful mathematical base for the study of biological neural networks.

References

[1] Michael A. Arbib. *Brains, Machines, and Mathematics*. Springer-Verlag, 1987.

[2] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Centre de Recherches Math?ematiques, third edition, 1999.

[3] Valentino Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT Press, Cambridge, MA, 1984.

[4] Richard Cole, Peter Eklund, and Gerd Stumme. Document retrieval for email search and discovery using formal concept analysis. *Applied Artificial Intelligence*, 17(3), 2003.

[5] Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.

[6] M. J. Healy, R. D. Olinger, R. J. Young, S. E. Taylor, T. P. Caudell, and K. W. Larson. Applying category theory to improve the performance of a neural architecture. *Neurocomputing*, 72:3158–3173, 2009.

[7] Michael J. Healy and Thomas P. Caudell. Knowledge representation and possible worlds for neural networks. In *2006 IEEE-INNS International Joint Conference on Neural Networks. Vancouver, BC, Canada. (CD-ROM proceedings)*, pages 5354–5361. IEEE,INNS, IEEE Press, 2006.

[8] Michael J. Healy and Thomas P. Caudell. Generalized lattices express parallel distributed concept learning. In Vassilis G. Kaburlasos and Gerhard X. Ritter, editors, *Computational Intelligence Based on Lattice Theory*, volume 67 of *Studies in Computational Intelligence*, pages 59–77. Springer, 2007.

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

[10] Michael John Healy, Thomas Preston Caudell, and Timothy E. Goldsmith. A model of human categorization and similarity based upon category theory. Technical Report EECE-TR-08-0010, Department of Electrical and Computer Engineering, University of New Mexico, June 2008.

[11] Pierre Lavenex and David G. Amaral. Hippocampal-neocortical interaction: A hierarchy of associativity. *Hippocampus*, 10:420–430, 2000.

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

[13] W. S. McCullough and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[14] Florian Mormann, Simon Kornblith, Rodrigo Quian Quiroga, Alexander Kraskov, Moran Cerf, Itzhak Fried, and Christof Koch. Latency and selectivity of single neurons indicate hierarchical processing in the human medial temporal lobe. *The Journal of Neuroscience*, 28:8865–8872, 2008.

[15] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[16] Nils J. Nilsson. The physical symbol system hypothesis: Status and prospects. In M. Lungarella et al., editor, *50 Years of AI, Festschrift, LNAI 4850*, pages 9–17. 2007.

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

[18] S. L. Tanimoto. *The Elements of Artificial Intelligence: An Introduction Using Lisp*. Computer Science Press, Rockville, Maryland, 1987.