Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Technical
Memo
TM-89-01

Connectionist Models and Figurative Speech

Susan Holbach-Weber

September 1989

Deutsches Forschungszentrum für Künstliche Intelligenz
GmbH

Postfach 20 80
D-6750 Kaiserslautern
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Connectionist Models and Figurative Speech

Susan Holbach-Weber

DFKI-TM-89-01

Connectionist Models and Figurative Speech

Susan Hollbach-Weber

Abstract

This paper contains an introduction to connectionist models. Then we focus on the ques-tion of how novel figurative usages of descriptive adjectives may be interpreted in a structured connectionist model of conceptual combination. The suggestion is that infe-rences drawn from an adjective's use in familiar contexts form the basis for all possible interpretations of the adjective in a novel context. The more plausible of the possibilities, it is speculated, are reinforced by some form of one-shot learning, rendering the inter-pretative process obsolete after only one (memorable) encounter with a novel figure of speech.

CONTENTS:

Part 1: A Tutorial Introduction to Connectionist Models.

1 Introduction and Overview

Connectionist models are the first attempt to provide a computational model of the human brain. Historically, the subfield of computer science known as artificial intelligence has been inspired and motivated by the performance of the human brain. The computational characteristics of the brain are very different from those of computers however. Where computer processors are very fast, measurable in nanoseconds, the processors used in the brain (neurons) are very slow, with response times on the order of several milliseconds. Communication channels between computer processors are typically restricted, and of course in a single processor architecture nonexistent. Neurons, however, are interconnected with each other by tens of thousands of communication channels, known as dendrites and axons.

The fact that the processing speed of neurons is so slow has something interesting to say about the computational style of neural systems. The human brain can perform such complex tasks as object recognition in a matter of a few hundred milliseconds. This means that entire complex behaviors are carried out in about one hundred time steps, an observation known as Feldman's 100 step rule. This implies that neural computation is massively parallel and organized into relatively shallow computation trees.

The attempt to model biological neural systems raises two distinct questions. One question is how the neural signals are generated and transmitted: neurobiology supplies the answer to this question. The second question is what algorithm is used in neural computation, that is, what function is computed by an individual neuron in response to its inputs. This second question is very difficult to answer. The approach adopted by connectionist models is to run simulations of networks whose units execute user-supplied functions, an approach which completely begs the question.

Since connectionist models are neurally inspired, an understanding of neural mechanics helps provide a basis for understanding connectionist models. Neurons are a biological cell, whose walls extend in long filiamentary processes known as dendrites. Dendrites are used to both transmit and receive signals to and from other neurons. A specialized form of dendrite is the axon; wrapped in myelin for insulation and equipped with nodes of Ranvier for amplification, the axon is used to transmit signals over long distances. The point at which the tip of an axon or dendrite makes contact with the dendrite of another neuron is called the synapse.

Neurons transmit electrical pulses of varying rates, where the rate of transmission is a function of the inputs currently being received by the neuron. A mechanical analogy to help explain the notion of neural computing is to imagine a neuron to be a tub, with water spouts draining into it, and an output valve at the bottom of the tub. The output valve is regulated by a float mechanism, so that when the water level in the tub rises to the level of the float the value opens, draining the tub. The output thus generated is transmitted to any number of other such tubs by means of branching drainage channels.

A refinement to this model takes into account the notion of inhibitory inputs. Certain neural inputs tend to block the activity of a neuron rather than stimulating it. These inhibitory signals can be modelled by a moveable dam across the input channels, whose position is regulated by the inhibitory input. If the inhibitory input is very strong, it will cut off all other inputs to the tub; if it is only moderately strong, it will merely reduce the input flow.

The water-and-tub analogy completely breaks down with respect to network topology. Neural systems have no inherent restrictions on the pattern of connectivity between units; the tub analogy only permits downhill flow of signals, a restricted form of topology known as the feedforward network.

Having now obtained an intuitive grasp of how neural computations behave, we can focus on the classification of neural models. Neural models can be coarsely divided into two camps: analytical models and empirical models.

Relatively little work has been done on developing analytical approaches to neural modelling, in part due to the analytic complexity of neural systems. A realistic model of the biological neuron has twenty or more parameters; this fact, coupled with the high degree of connectivity between neurons, leads to a dearth of analytical models of completely general neural systems. In order to develop an analysis, the system must be modellable with a very small number of parameters, enforcing a degree of simplification that necessarily leads to a loss of generality. Accordingly, a greater emphasis is placed in the literature on empirical models of network behavior. Empirical models are used not as formal analyses of system performance characteristics, but rather as existence proofs for a specific capability. The idea is to simulate the operation of a working neural network, be it by direct implementation in VLSI circuits, the hardware approach, or indirect implementation on existing computers, the software approach. To date, although the hardware option seems to afford the most speed and accaracy, it is still a premature solution. The problem is that very little light has been shed as yet on the actual algorithms used by neurons in their various computations, so the designer of a neural model is forced to choose them almost arbitrarily. Until more is known about the interaction between network computing characteristics and choice of activation functions, the greater flexibility afforded by software simulations of network activity more than compensates for the reductions in speed.

At this point in the discussion we are finally in a position to define the term "connectionist models”. A connection model is the empirical simulation on a (serial) computer of a neural network. The simulation need not be on a serial architecture machine, but for reasons of availability and ease of programming they generally are. The two elements that are invariant across all connectionist models are *units*, computational simulations of neurons, and *links*, the communication channels corresponding to dendrites and axons. Connectionist units compute a specified *activation function* of their input values, generating a single positive output value for the given time. As this output value is propagated across all outgoing links from the unit, this value may be modified by the *weight.* A positive link weight greater than 1 amplifies the signal, a positive weight between O and 1 attenuates signal, and a negative link weight transforms the signal into an inhibitory one.

At the level of the operating environment, that is, the external context surrounding and imparting meaning to the computation of the connectionist net, there are input and output patterns. Input patterns are a function not only of the network architecture and semantics but also of the environmental constraints on the function being computed. For example, a connectionist network in the form of an retinotopic map of the visual field would have individual input units corresponding to positional points in the visual field, much like pixels in a digitized camera image. The input to these units would depend on the visual scene being looked at, that is, input is environmentally determined. One can also imagine a network to compute a partial function on its input field, that is, not all input patterns are “valid", since not all input patterns have a corresponding known output. For such functions a popular approach is to "train" the network by presenting it with sequences of valid inputs accompanied by the expected output pattern, having first specified a *learning rule* for link weight adjustment to minimize the difference between the function actually computed by the network and the desired function. Network output, like network input, is a function of the design choices made by the architect of the network. One popular convention is to keep three distinct populations of units,

- input units,
- “hidden” units receiving input from the input units and sending output to other hidden units or to output units, and
- output units. In this case, system output is defined to be the activation of the output units. Output can also be defined to be the activity of the entire network, in which case some measure of stability or convergence is required before the output can be sampled.

The details of the internal structure and operation of connectionist units are the most discretionary, as they are determined neither by the strictures of the model nor by the environmental constraints on the functions being computed. For any given function to be computed, there exist an infinite number of connectionist architectures to compute it. One possible metric to choose between this infinity of options is minimality, but as there are so many variables interacting that minimizing one will always tend to maximize others. Different minimization metrics like biological accuracy, computational efficiency or analytic simplicity all lead to different minimal models.

The two principal variables at the level of the connectionist unit are the pattern of connectivity and the activation function. For a fixed activation function, different patterns of connectivity compute different functions, and for a given pattern of connectivity, different activation functions, obviously enough, compute different functions. The fundamental question to be explored by connectionist models is the interaction between these two variables.

To specify the architecture and operation of a connectionist network, one must tabulate the pattern of connectivity and the individual link weights, and provide the activation function for each unit in the network. Exploiting the fact that a link weight of O is functionally equivalent to having no link at all (assuming the multiplicative application of link weights to signals), a convenient notation for specifying pattern of connectivity is the *weight matrix.* Activation functions operate on the inputs to the units, inputs which have been optionally preprocessed by one or more *site functions.* The result of an activation function is a single positive value known as the *activation value* or *unit potential.* This value is then optionally transformed into a distinct output value by the *output function.* These multiple functions lead to a certain redundancy in specifying the operation of a unit. Suppose the unit is to compute the binary threshold of the sum of its inputs. One approach would be to have the sum of inputs computed by a site function, and the thresholding operation performed by the activation function; the output function is then simply the identity function. Alternatively, the activation function could compute the sum of inputs, obviating the need for a site function, and the output function could be the thresholding operation. A third alternative is to compress

both the summation and thresholding into the activation function, leaving both the site and output functions transparent (the identity function). If biological plausibility is a design factor, the electrical capacitance of the neuron is modelled by the activation value and the firing rate by the output value, with the preprocessing of inputs performed in the dendritic trees captured by various site functions. If biological plausibility is not a factor, and the connectionist model is seen merely as a vehicle for exploring massive parallelism in computation, then any function at all can be chosen for the site, activation and output functions, including probabilistic functions where the binary output of the unit is specified probabilistically with respect to the activation value.

Given this plethora of choice in specifying a network, it is clear that network semantics can be difficult to define. Since the function computed by a network depends primarily on the pattern of connectivity but also on link weights and activation functions, one possible grounding for network semantics is in the semantics of the individual connectionist units. For example, one popular interpretation of a unit's function is as a *feature detector.* If the unit is on, the feature is present in the environment. If it is off, the feature is either absent or simply has not been detected. Intermediate levels of activation correspond to degrees of belief in the presence of the feature. If two such feature detectors form a conjunctive input to a third node, then the semantics of this node is the conjunction of the two features in the environment. Thus node semantics can be used to recursively determine network semantics.

A final element of connectionist models that has nothing to do with the modelling constraints imposed by biological neural systems but is purely an artifact of the empirical simulation on conventional computers is the choice of simulation strategy or update rules. One option is to enforce global synchro-nization of all units by sequentially updating all units at each clock tick. This imposes uniform propagation delay (equal link lengths) among all units, and can lead to deadlock in cases where two equally valid network states are in oppo-sition and must be chosen between. Another option, less commonly used, is to update only a certain percentage of the units in a given simulation step. This necessitates certain fairness guarantees, guaranteeing that a given unit will have been updated by a fixed number of simulation steps, lest a unit get "unlucky" and never be updated. Although this approach solves the potential problem of systemwide deadlock, it implements random and variable length propagation delays between units, to sometimes curious computational effect. The extreme form of this algorithm is to update only one unit per Clock tick, again with fairness guarantees that every unit will eventually be updated.

This concludes the introduction to and overview of the general principles underlying neural networks in general and connectionist networks in particular. Neurally inspired networks can be classified with respect to each other along three mutually orthogonal axes. Two have already been mentioned: degree of biological plausibility and mode of analysis (empirical v. analytical). The third axis, the origin of network structure, or whether the network is pre-structured or self-structuring, forms the basis for the rest of this discussion of connec-tionist networks.

2 Self-Organizing nets: the PDP approach

The PDP ("Parallel Distributed Processing") research group at the University of California, San Diego have popularized and advanced the connectionist paradigm now known as the PDP model. In a PDP network, structure is initially uniform, with complete connectivity between units or layers of units, small random weights on the links, and one common activation function for all units. The network is then ”trained” to compute the desired function by supplying it with a *learning rule* to alter link weights and optionally a *teaching signal,* or the expected output for the function to be learned. Thus the network is in some sense self-structured, since the learning process eventually produces structural differentiation from the initially uniform state. The positive side to this approach is that a network can be trained on any one-to-one function whose input and output pattern are reasonably orthogonal. The drawbacks are that the resultant structure can be quite opaque and difficult to assign a semantics to, and the learning algorithms currently employed are biologically implausible in their behavior. Nevertheless, it is a rich and promising paradigm, and one that has recently generated considerable interest in the research community.

Historically, the PDP approach is based on the simple linear models of the 1950's. The structural characteristics of a simple linear model include a two layer feedforward (no feedback links) architecture whose layers are fully connected with initially small random weights. The link weight on a link connecting unit A to unit B multiplied with the output signal of A being propagated along the link to produce the modified signal received as input by B. The activation function sums these weighted inputs, and the output value is equal to the activation value.

These simple linear models can "learn" according to Hebb's rule, which states that when input unit A and output unit B are simultaneously excited, the strength of their interconnection increase. Training a simple linear model in this fashion involves clamping on the input units for a given input pattern and the output units in the pattern expected for the given input, and applying Hebb's rule. This process is repeated many times for all possible input patterns. This model, known as a *linear assoCiator,* can learn multiple pattern associations if the pattern are orthogonal. If the patterns are overlapping, learning performance can be improved by using the delta rule, where rather than clamping the output units on to the expected pattern, the expected values are supplied to the output units as a distinct signal and they are allowed to compute the function of their inputs (the actual output). The difference between the actual output and expected output is used to determine the change in link weight on all incoming links.

Two useful applications of the simple linear model are pattern association, for example mapping upper case letters to lower case letters, and pattern retrieval, where the network is trained to auto associate patterns (i. e. the input and expected output are identical), a function useful in retrieving known exemplars from degraded representations. Pattern retrieval can be seen as a form of content addressable memory.

A third application of the model can be developed by installing mutual inhibition links between all units in the output layer, and adopting the learning rule that if an output unit wins the competition on a given input, each of its input links gives up some proportion of its weight, and that weight is then distributed equally among its active input lines. The weights of all links coming in to a unit must sum to 1. This algorithm for *competitive learning is* a form of unsupervised learning, as there is no explicit teaching signal, no particular expected output. The task that such a network performs *is pattem classification,* where the population of input patterns is broken down into n or fewer classifications, where n is the number of competitive output units. If structural clustering exists within the input population, the network will find it; if the population is more evenly distributed, however, the classification will be arbitrary and possibly non-optimal.

Simple linear models are limited to computing linear functions, that is, functions whose outputs increase in direct proportion to their inputs. Feedback links and hidden or multiple layer architectures do not add to the power of these models. Accordingly, the next stage in the historical development of the PDP paradigm was to introduce non-linearities into the model.

The linear threshold model possesses the same structural characteristics as simple linear models, namely, a two layer feedforward architecture with full connectivity, but the activation functions changes from a weighted sum of inputs to a binary thresholded weighted sum of inputs. That is, if the weighted sum of inputs is less than the threshold, the output is 0, and if it is greater than the threshold, the output is 1. Such networks can compute any boolean function, including non linear functions such as exclusive-or. This model is also known by the name of *perceptron.*

The Hebbian learning rules and the delta rule can both be applied to the perceptron, as can the competitive learning rule, leading to both supervised and unsupervised learning. The interesting feature of this model, however, is that adding "hidden~~ layers of units between the input and output layers actually increases the power of the network, due to the non-linearity of the activation function. For example, while it is still impossible to compute exclusive-or with a two layer perceptron, it can be computed by any number of three-layer (one hidden layer) linear threshold networks. For twenty years, from the late 1 950~s to the late 1 970's, interest in multi-layered perceptrons atrophied since there was no known way to apply the delta rule to these networks, thus enabling them to learn. The delta rule is only directly applicable to the layer of links coming in to the output units, since these units receive the training signal needed to compute the change in weight.

The generalized delta rule, appearing in the early 1980´s, revitalized interest in multilayer feedforward networks. By adopting a differentiable approximation to the linear threshold function called a *quasi-linear* threshold function, an algorithm, popularly known as *error backpropagation,* was devised to recursively compute the change in link weights for hidden layers from the basis provided by applying the delta rule to the output layer. Unfortunately the resulting training algorithm is very slow, requiring thousands of presentations of each inputoutput pair.

At about the same time that the generalized delta rule for error backpropagation was invented, a unique and somewhat offbeat model known as the Boltzmann machine was devised. Inspired by the physical systems known as "spin glasses", a Boltzmann machine has binary unit output and symmetric links, so that if unit A is linked to unit B with weight w, then unit B must also be linked to unit A with the same weight w. The activation functions are probabilistic, that is, the inputs determine the probability with which the output of the unit will be O or 1. The goal of the learning algorithm is to minimize global system energy, where energy is a function of the sum of all link weights connecting two active units plus the total number of active units. The algorithm, starting with initially random activations, is to sequentially update units to locally minimize the global energy. In order to overcome the problem of local minima endemic to all gradient descent algorithms, the gimmick known as *simulated annealing is* used. A parameter known as the *temperature is* introduced into the probabilistic activation function such that at higher temperatures the unit will turn on and off more or less randomly, while at lower temperatures the probability of the unit turning on becomes more strongly associated with the amount of input to the unit. At high temperatures the system is unstable and will never converge to a solution. At low temperatures the system is too stable and local minima in the energy space become a problem. Thus in simulated annealing the temperature of the system is initially high, and as the temperature is gradually lowered, the optimum point of balance between instability and stability is crossed, thus usually ensuring that a global minimum will be found. The learning algorithm cumbersome and slow, requiring tens of thousands of simulation "sweeps", or annealings in order to collect the statistics needed to approximate the implemented probabilities. Boltzmann machines are highly implausible, biologically speaking, and are currently regarded as more of a curiosity than a useful connectionist model.

There are certain drawbacks to the PDP approach. The approach sheds no light on the question of the tradeoffs between patterns of connectivity and choice of activation functions, since one activation function is generally shared by all units in the network. The structure that arises from applying a given learning algorithm can be quite obscure, with no readily derived semantics, particularly for multiple layers of units. But the compensations are impressive: PDP networks can be trained on any of a large class of functions, and the typically highly parallel and redundant computations that emerge from such training leads to graceful performance degradation on incomplete or erroneous input.

3 **Structured connectionist networks**

The structured approach to connectionist models differs from the PDP approach in that the pattern of connectivity is specified by the network designer, rather than emerging incrementally through some training procedure. This approach leads to greater design flexibility and clearer network semantics, but also tends to produce network designs of a less robust nature. The highly distributed computational style of PDP networks with their resultant obscure semantics are difficult to design by hand, so the tendency is often to adopt a somewhat more *localist* style of network design. Interestingly, structured networks are the favored medium for cognitive scientists building computational models of their favorite cognitive phenomenon, particularly low level perceptual tasks like vision, speech and hearing.

One of the commonest design rule adopted by the architects of structured networks is the *unit-value principle*, which states that a separate unit exists for each distinct value of a feature. This principle leads to clear node semantics, but can also lead to *crosstalk,* if two concepts with overlapping feature sets are considered concurrently. It is not yet clear how to dynamically associate a value with a variable, a technical difficulty known as the binding problem. One way to circumvent the binding problem in the case of potential crosstalk is to allocate nodes to explicitly represent the conjunction of two feature values. This simplistic approach has the obvious disadvantage of being combinatoric. Another possible solution to the binding problem is to allocate distinct time slices of the simulation to each object under consideration, thus adopting a temporal, rather than a spatial, metric of separation between the two feature spaces. This idea has promise, but the details are still being worked out, and it is too early to tell if it presents a viable solution.

The semantics of a connectionist network can be defined recursively with respect to the semantics of its component units. If a node is regarded as a feature detector, then binary output values have obvious truth value mappings and continoous values can represent degrees of belief in the proposition represented by the unit. In general, multiple unit activation is considered to be conjunctive, so that explicit mechanisms to represent disjunctions are needed.

One disjunctive mechanism is the mutual inhibition subnet. In such a structure, there are inhibitory links fully connecting every unit in the subnet. Then concurrent activation of units results in the suppression of all units but the strongest, whose activation is dampered somewhat but not extinguished. Another mechanism is the winner-take-all subnet, in which an auxiliary node computes the maximum of the values in the subnet. This maximum value is transmitted back to each participant in the winner-take-all competition, and on receiving a value greater than its own, a node voluntarily turns itself off.

The Rochester Connectionist Simulator, soon to be acquired by DFKI, presents a useful tool for simulating structured connectionist networks. It runs on a Sun Workstation under both Suntools and X, and is written in the C programming language. It has a powerful graphics interface with iconic representation of network units that can be selected by mouse clicks. Once a unit has been selected, information about the unit can be displayed, information such as incoming and outgoing links, and the values of all local variables such as state, potential and output. To program an application on the Simulator, the user writes a subroutine in C with appropriate calls to predefined functions such as MakeUnit and MakeLink. This subroutine is linked at compile time to the simulator, creating an application specific executable. Running the executable creates a graphics window for the application from which you can run the network, display individual nodes, and alter node parameters.

Part II: A structured connectionist model of figurative adjective-noun phrases

1 Introduction

This section of the report focuses on the question of how novel figurative usages of descriptive adjectives may be interpreted in a structured connectionist model of conceptual combination. The suggestion is that the inferences drawn from an adjective's use in familiar contexts form the bases for all possible interpretations of the adjective in a novel context. The more plausible of the possibilities, it is speculated, are reinforced by some form of one-shot learning, rendering the interpretive process obsolete after only one (memorable) encounter with a novel figure of speech.

What interpretive mechanisms are used in the attempt to understand a figure of speech for the first time? More specifically, what mechanisms are used to determine the meaning of a figurative adjective-noun phrase? A figurative interpretation of an adjective-noun phrase is required when the adjective names a property or a property value not possessed by the noun, as in, for example, the expressions 'green idea' or 'blue tree': ideas do not have colors, and trees are not naturally colored blue. Since it is safe to assume that a speaker (or writer) will not intentionally employ phrases that are meaningless to the listener (or reader), a figurative interpretation for such apparently anomalous phrases must somehow be found. The successful interpretation of a novel adjective-noun phrase hinges on selecting plausible values for the salient properties of the noun. It is argued here that the connotations of the adjective in previously encountered contexts supplies a basis for any such interpretation.

In order to define what is meant by adjectival connotations, it is first necessary to devise a model of category representation. The model's design is constrained by the choice of a connectionist implementation. Assuming that a connectionist unit can represent feature values, a category is incarnated as varying degrees of concurrent activation of a subset of its allowable feature values. In a literal adjective-noun phrase, the descriptive adjective names a property value of the category denoted by the noun, eg. 'green peach' or 'novel idea'; in other words, it indexes a feature value unit associated with the noun. As correlations often exist between feature values, indexing one will tend to excite others, thus supplying the adjective with characteristic connotations, known as *direct inferences.* For example, while a (ripe) peach is normally pink on the outside and juicy and sweet on the inside, a *green* peach is unripe: dry and sour in addition to being green in color. These direct inferences suggest themselves so strongly that perceptual values are often used metonymically to stand for certain (non-observable) functional or constitutive property values. For example, *unripe* is an extended sense of the adjective 'green'.

When the modifying adjective indexes a feature value not possessed by the noun, however, the semantics of the situation are more complex. The idea is to indirectly index feature values of the noun by establishing a mapping from feature values associated with the adjective in literal noun contexts. For instance, since green is so commonly associated with the notion of unripeness in the domain of fruits, it seems plausible that a ‘green idea’ is one that is somehow premature, or not fully developed. This interpretation hinges on finding mappings not only between the properties of fruit maturity and idea development, but also between their respective values, *immature* and *underdeveloped*.

These ideas have been implemented in a structured connectionist knowledge representation and inferencing system. The implementation, written in the C programming language, runs on a Sun™ Workstation as an application of the Rochester Connectionist Simulator, the results of which are represented as graphical icons by the Graphics Interface for the Simulator [Goddard et al., 1988]. The resulting system is known as DIFICIL, for Direct Inferences and Figurative Interpretation in a Connectionist Implementation of Language comprehension.

The basic building block of a connectionist net is the unit. For each unit in the network the simulator keeps track of the activation function used and the connectivity to other units, unidirectional links coming into and going out of the unit from and to other units in the network. The activation function typically computes unit potential from the input values coming in across the links from other units. This potential is sent as the unit’s output across the outgoing links.

The units in a connectionist network, like atoms in LISP, have no integral semantics apart from the function they compute. The labels they carry have meaning only for the human observer of the network’s operation. Labels nevertheless serve a useful function in keeping track of the real-world constraints that exist between the concepts being represented. Accordingly, certain nodes of the network are named for the known nouns and adjectives (categories and property values respectively), and others are named for the properties implied by the property values named by adjectives. The category and property value nodes serve as input nodes: an adjective-noun phrase is input to DIFICIL by pegging the activation values of the relevant category and property value nodes to their maximum value. All nodes in the network serve as output nodes, in the sense that the pattern of activation over all nodes defines the state of the network. However, the property values are usually of greatest interest, so they are the best candidates for the title of output units. For example, if the input phrase is ‘green peach’, then the expected output will include the values *dry, sour, hard* and *unripe*. These distinctions are more suited to layered feedforward networks than to structured nets with feedback: a given property value node will be an input node in one network run, an output node in another, and a ‘hidden’ unit in a third. These operational characteristics of a DIFICIL network will become clearer as the discussion of network architecture unfolds.

A widely accepted model of metaphor interpretation is to establish a mapping from a source domain to a target domain. For example, in the metaphor “marriage is a zero-sum game” the source domain is games, the target domain marriage, and the mapping to establish correspondences between contestants and spouses, winning and personal fullfillment, and so on. In accordance with this view, the process of interpreting a novel figurative adjective-noun phrase in DIFICIL occurs in three phases. First the need for a figurative interpretation is signalled by literal semantic anomaly. Then all the literal connotations of the adjective are primed to establish a source field of property values. Finally, mappings are established from values in the source domain to values in the target domain, namely, permissible values of category denoted by the noun. This requires not only establishing mappings between source properties and target properties, but also between the relevant property values. Before considering the form taken by these interpretive mechanisms, however, the prior question of how to interpret a literally intended adjective-noun phrase must be addressed.

2 Preamble: a functional aspect model of category representation

The interpretation of novel figurative adjective noun phrases in DIFICIL is based on the behavior of the system when presented with literally interpretable adjective-noun phrases. When an adjective modifies a noun in a literal context, the category denoted by the noun is cast in a new light. Sometimes the shift in perspective is a minor one: the phrase ‘green car’ carries little additional information over the selection of a specific color. The phrase ‘green banana’, however, entails a significant modification to the default values of bananas: in addition to the color changing from yellow to green, one also infers that a green banana is unripe, dry, bitter, difficult to peel, and so on. Since it seems untenable that this (completely different) view of the property values of bananas constitutes a proper subcategory, this information must exist within the category itself.

This habit of inferring changes in property value settings from the knowledge of one property value is called *direct inferencing*. Direct inferences can be either *immediate* or *mediated*. Immediate inferences are the direct inferences available at the level of the category under consideration. They are performed quickly, in a few hundred milliseconds, and without conscious thought. These immediate inferences must reflect the structure of stored knowledge, as they are available too quickly and effortlessly to involve any complex form of information retrieval. The argument is that the patterns of immediate inferences reflect the structure of connections in the underlying spreading activation model, implemented here as a structured connectionist network. Mediated inferences are the second form of direct inference, where knowledge about a more abstract category is used to supply the information necessary to understand discourse.

Mediated inferences take somewhat longer than immediate inferences, as they require chaining up the subcategorization (or 'property inheritance') hierarchy. Mediated inferences are not exploited in the figurative interpretation process, however; only the immediate inferences available for a category are used.

At the heart of the proposed model of figurative adjective-noun interpretation lies a functionally organized context sensitive model of the internal structure of categories. Given that physical objects possess certain properties with characteristic values, the question is how the correlations between values is represented. Functional properties of the object supply the necessary organizational structure, as each value of a functional property participates in a distinct aspect, or informal coalition of property values, of the category. For example, the functional property *ripeness* of fruit motivates three distinct aspects of bananas: unripe bananas are green, hard, dry and bitter, while ripe bananas are yellow, soft, moist and tangy-sweet, and rotten bananas are black, mushy, wet and sickly-sweet.

The choice of functional properties as the organizing principle of the internal structure of categories is based on the assumption that *categories, as mental constructs of active agents, are inseparably linked with the agent's goals.* To a watchmaker, it is the extreme hardness of diamonds that is of paramount importance, while to a socialite it is their more intangible property of conferring status on the wearer that governs the representation. The two representations of the category, although based on the same external substance, are quite different. Diamonds to a watchmaker are very hard and durable. Diamonds to a socialite are brilliant, clear and expensive. The two sets of properties both hold true of diamonds, but their relevance is contextually determined. As well as determining relevance, context can also actively determine the property values themselves; an industrial diamond is generally tiny and hence relatively cheap, while an ornamental diamond is much larger and more expensive (see Figure 1).

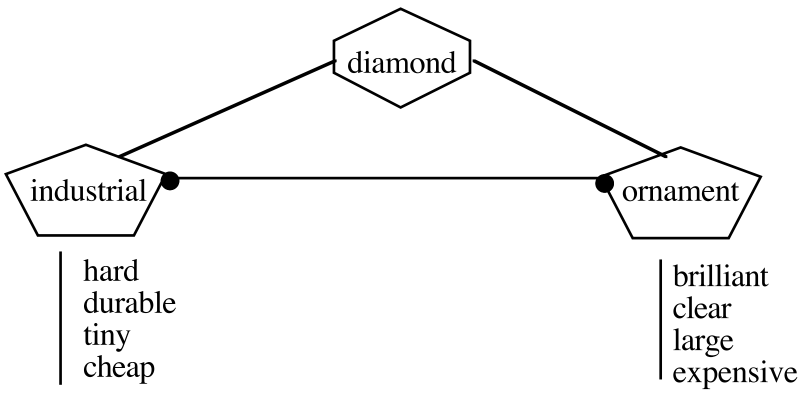

Figure 1: Two competing aspects of the diamond category.

Aspects of a category can be established by several mechanisms. The direct approach is to name a property value participating in the aspect. For example, the phrase 'tiny gem' supplies evidence for the relevance of the *industrial* aspect. A similar effect is achieved by actually naming the aspect (eg. 'industrial gem'), as the motivating functional property value which gives the aspect its name also participates in the coalition. Sometimes one aspect will be subsumed by another, as, for example, when one planning goal is a subgoal of a higher level goal. When this happens, invocation (by whatever means) of the high level goal supplies activation to the subgoal, thus indirectly exciting the subordinate aspect. Property inheritance can also supply indirect activation to an aspect. If a category happens to excite a property value that also participates in an aspect of a superordinate category, then when activation propagates up from the category to the super-category, that aspect will be preferred. Finally, an aspect can be established by default; in the absence of information to the contrary, the most typical aspect of the category will predominate.

Knowledge bases in DIFICIL are not built by hand; an input language exists with which to specify information to be incorporated into the system. The relationship between a category, a property and relevant property values is given by

hasslot (category: property; value$_1$, ..., value$_n$).

The internal structuring of property values into aspects are established with

aspect (category: goal [default]; value$_1$, ..., value$_n$).

Property inheritance and abstraction hierarchies are specified respectively by

subcat (category: subcategory$_1$, ..., subcategory$_n$)

and

abstracts (property: subproperty$_1$, ..., subproperty$_n$).

A new network unit is allocated and appropriately named on first reference to a term, and is indexed by subsequent references. For example (refer to the key appearing in Figure 2), the DIFICIL network fragment built by the statements

hasslot (gem: cost; cheap, expensive)

hasslot (gem: look; ugly, pretty)

hasslot (gem: function; industrial, ornament)

aspect (gem: ornament [default=] TRUE; expensive, pretty)

is shown in Figure 3. Details of the mapping from this simple input language to the connectionist structures of a DIFICIL knowledge base appear in [Weber 1989].

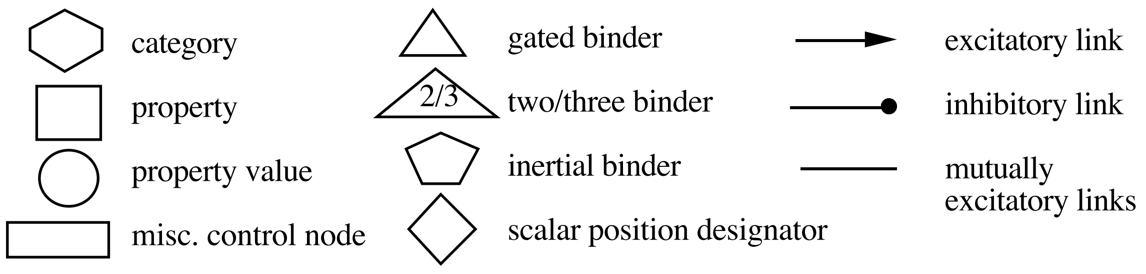

Figure 2: Key to symbols used to depict the various units in network structure diagrams. All category, property and value nodes set their potential and output equal to the sum of their inputs, unless wired high externally, in which case output is set to the given potential, and further inputs (or lack thereof) are ignored until reset. This latter case arises when a category or value is being used as an input node.

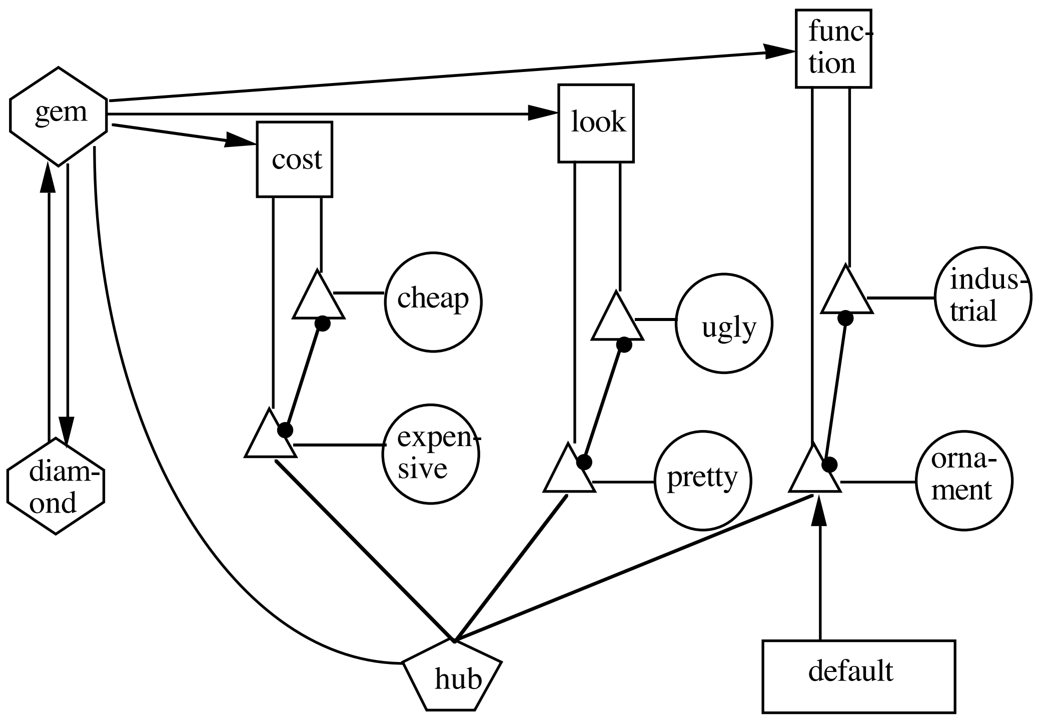

Figure 3: Connectionist structures implementing the *ornamental* aspect of gems.

3 Detecting Semantic Anomaly

When thinking about physical objects, the mental representation of the object includes various properties with associated property values. For example, beans can be either green or yellow in color. Thus the phrase ‘green beans’ has a readily available semantic interpretation. But what about the phrase ‘green recruit’ or, even odder, ‘green idea’? It is assumed (see Section 7) that both adjectives and nouns have only one word sense; in this case, ‘green’ is used exclusively in the *color* sense. Object properties can be organized into a hierarchy of applicability whose structure is isomorphic to the structure of the agent’s ontological knowledge, in much the same spirit as Keil’s hierarchy of predicability. Keil defines predicability to be the knowledge of which predicates can be combined with which terms in a natural language. A predicate is said to span a term when the predicate can meaningfully be applied to the term and the resulting phrase can be assigned a truth value, be it true or false. A category error occurs when a predicate is used in conjunction with a term it does not span. For example, ‘green idea’ is a category error, since only physical objects can have color as a property. Thus it is neither true that the idea is green nor that it is not green, since the latter statement implies that the idea has a color, the value of which is something other than green.

The phrase ‘green recruit’ also involves a form of semantic anomaly, again assuming that the adjective ‘green’ is used to denote color. People, as physical objects, can take on various colors, so there is no category error. The anomaly occurs instead at the level of an unexpected property value: people are not naturally colored green. This is an example of an *expectation violation*, the second form of semantic anomaly exploited in this study.

The mechanisms to detect the need for a figurative interpretation currently implemented in DIFICIL are limited to category error and expectation violation detection, although nothing precludes the possibility of eventually incorporating other signals of figurative usage, such as contextual cues. Category errors are detected on a property-by-property basis. A detection node is created for each new property named in the database. This node receives excitation from the property-value nodes and inhibition from the category, so if a property value should be activated while the category is inactive, a semantic anomaly will be reported as required. For example, in the DIFICIL network fragment created by the statements

    hasslot (diamond: size; tiny, small)

    hasslot (desk: size; large, enormous)

and

    hasslot (idea: novelty; novel, familiar),

the phrase ‘large idea’ raises a category error. Excitatory links are established by the first two hasslot statements from the various size values to the size category error node, as well as two inhibitory connections, one from *diamond*, the other from *desk*. So any phrase utilizing a size value such as ‘large’ in the context of any noun other than ‘diamond’ or ‘desk’ will raise a size category error.

Expectation violations are detected at the level of the category-property conjunction. A key element of the network’s structure is the *binder unit*, a connectionist unit that represents the semantic association (binding) of a set of other units. The gated binder, which requires that each of a distinguished set of inputs be active before summing and broadcasting all other inputs, is used to establish the association of a property and one of its values with a category. Only if both the category and the property are activated will this gated binder node compute the sum of its inputs. The detection node receives inhibition from the property-value binders, and excitation from the property, so if any property value not possessed by the category is named, an expectation violation results. Thus for the (somewhat unrealistic) network fragment given above, the phrases 'large diamond' and 'tiny desk' both result in expectation violations.

These semantic anomaly detection structures are depicted in Figure 4. The category error detection mechanism shown works for all references to size in any noun context, while the expectation violation mechanism works exclusively for the size of diamonds. Both forms of semantic anomaly, when detected, transmit their activation to the global *metaphor* control node. When all possible immediate and mediated inferences have been drawn, if there is still an anomaly being reported, the *metaphor* node is activated, signalling a network-wide change of state, from literal interpretive mechanisms to figurative.

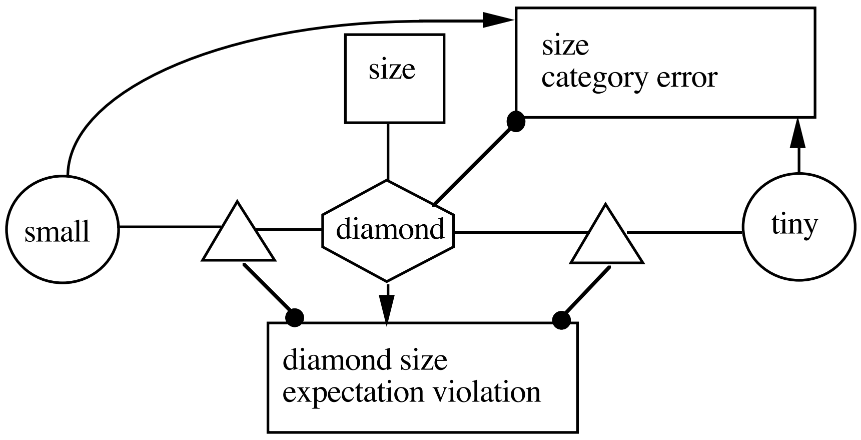

Figure 4: category error and expectation violation detection structures

4 Priming the Source Field

Once a semantic anomaly has been detected, signalling the need for a figurative interpretation of the adjective noun phrase, a source domain for possible meaning mappings must be established. This domain consists of the adjective's connotations in all allowable literal contexts. The connotations of an adjective arise from its associations within the literally allowable noun contexts. The connotations considered by DIFICIL for the purposes of figurative interpretation are the immediate inferences arising from the modification of an arbitrary noun by the given adjective. For any category known to the system, if the adjective names a property value that participates in an aspect of that category, then it will trigger a characteristic set of immediate inferences, as activation spreads from the named value to the aspectual hub and from there propagates to all related property values.

This idea is implemented by a special purpose node, call it the priming node, that is connected to all concepts in the knowledge base. On the rising edge of the metaphor detection signal, that is, when semantic anomaly is first detected, the priming node is enabled. The activation of this node decays linearly and fairly rapidly, so that after fewer than 10 time steps the excitatory (or priming) signal has changed to an increasingly inhibitory one (see Figure 5). (Actual inhibition is required because the local feedback within an aspect gives it stability; this stable coalition must then be actively defeated.)

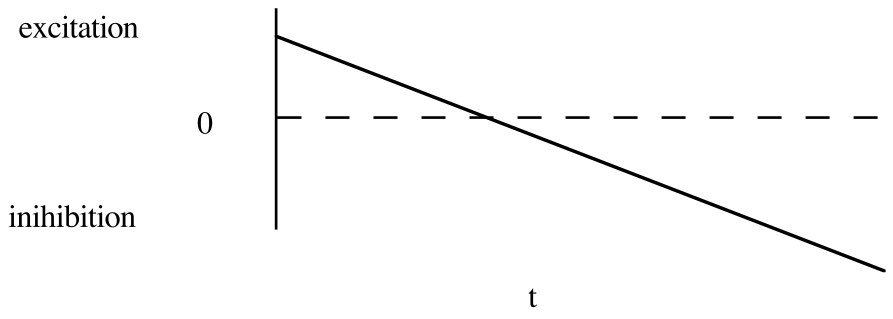

Figure 5: Activation function of the concept priming node. This unit is linked to all concepts in the knowledge base.

Thus for a short time following the detection of semantic anomaly all the categories in the knowledge base are stimulated. In this state the system is effectively consi-dering the combination of the currently active adjective with every known noun in parallel. This state is maintained long enough for activation to propagate to any relevant conceptual aspects. For example, suppose the input phrase was ‘aggressive diamond’. On detection of the semantic anomaly in this phrase, the system briefly considers (in parallel) all possible additional phrases involving ‘aggressive’, such as ‘aggressive fruit’, ‘aggressive weapon’, ‘aggressive person’, ‘aggressive flower’, and so on. Not all of these phrases convey literal meaning of course, but those that do, such as ‘aggressive weapon’ and ‘aggressive person’, establish the relevant aspect of the category. The property values participating in these aspects form the interpretive basis, or source field, for the eventual figurative interpretation of the target phrase.

5 Establishing the property and value mappings

The final stage in the process of figuratively interpreting an adjective noun phrase involves setting up mappings from the values primed by the literal connotations of the adjective in various contexts (the source domain) to values of the category denoted by the noun (the target domain). The set of all values activated in this manner form the interpretive basis for understanding a figurative usage. Semantic correspondences must somehow be established between the anomalous property value denoted by the adjective and property values belonging to the category denoted by the noun. There are two correspondences to be established, the first being property to property, the second, value to value. For example, when interpreting the phrase ‘green idea’, one must not only establish that the color property in fruit corresponds somehow to development in ideas, but also that the value *green* maps to *underdeveloped*. This holds true not only in the case of category errors, where the property implied by the value does not belong to the category, but also for expectation violations, where it does: the phrase ‘green recruit’ requires mappings between the properties *color* and *experience* and the values *green* and *experienced*.

These mappings are discovered by two interlocking interpretive processes, one to establish all property to property mappings, the other to set up the value to value correspondences within related properties.

In order to establish a semantic correspondence between the property named by the adjective (eg. ‘green’ names a color) and properties of the noun (eg. the experience of recruits), a property abstraction hierarchy relates all the properties in the knowledge base. As soon as the need for a figurative interpretation has been recognized, activation is permitted to spread throughout the abstraction hierarchy from the property named by the adjective. Activation will eventually spread to every property in the knowledge base, so in some sense the hierarchical arrangement is unnecessary: one could simply stimulate all properties in parallel, and achieve the same end result. But with the hierarchical spread of activation the timing delays between the various meaning hypotheses reflect their plausibility: later suggestions are increasingly implausible, as the semantic distance between the properties increases.

Thus the real interpretive work is done at the level of the value to value mappings. There are three methods used to establish semantic correspondences: (direct) value transference, (indirect) scalar correspondence, and (very indirect) qualitative correspondence.

The most straightforward interpretations arise when a property value of the target category is made available through an immediate inference associated with another category. For example, suppose it was ‘known’ to the system that aggressive people are also large in size, i.e. *large* and *aggressive* both participate in the same aspect of *person*, then the unfamiliar figure of speech ‘aggressive diamond’ would be interpreted as denoting a large diamond (see Figure 6). This form of mapping value-to-value mapping, known as *property value transference*, is applicable when a property value (and hence its associated property) is common to both the source and target fields.

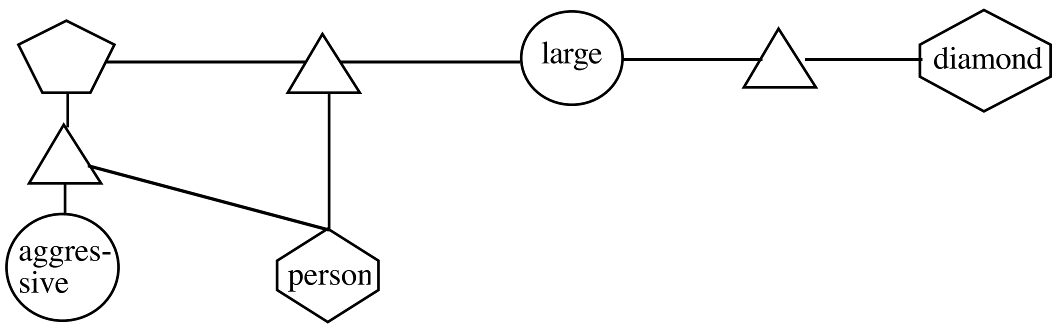

Figure 6: Direct value transference: the phrase 'aggressive diamond' is interpreted to mean a large diamond, due to the correlation of *aggressive* and *large* within an aspect of the category *person*.

To establish correspondences in the non-overlapping areas of the source and target fields, however, more indirect mappings must be resorted to. One possible indirect mapping exploits the scalar nature of many properties, particularly properties of physical objects. Properties that are quantitative in nature, such as size, weight, density, malleability and so on, tend to impose a natural scalar ordering on their values. Thus assuming that two such properties have somehow been placed in correspondence, the value mappings become obvious. For instance, if *size* in the source field corresponds to *weight* in the target field, then *small* maps to *lightweight, large* to *heavy*, and so on. This method of value mapping is called *scalar correspondence*.

This approach is based on the intuition that quantitative physical property values tend to be scalar in nature, that is, the allowable values for a given property can be strictly ranked with respect to each other, from least to greatest. One example of this behavior is the *temperature* property, whose values range from freezing through cold, cool, warm, hot and finally to boiling/burning/blistering etc. There will generally be two values that typify the positive and negative extremes (eg. hot and cold) with a third value typifying the neutral setting. For example, the *intensity* scale may have the positional designators +, 0 and –. Unranked but mutually exclusive properties are handled in the same way, as there is no attempt to actually impose the scalar ordering implied by the choice of designators. There are many possible scales to choose from, the most obvious of which is the intensity scale: temperature ranges from cold (–) to hot (+), size from small (–) to large (+), hydration from dry (–) to juicy (+). Another common scale is the emotional connotations scale: anger is negative, reasonableness positive; sadness (–) opposes happiness (+), and so on. As a property value can participate in any number of scales, it is necessary to distinguish the scales by their scalar position designators. The various scales are considered to be incomparable, so no contradiction is perceived by the system with the assignment of anger, for example, to the positive end of the intensity scale as well as to the negative end of the emotions scale.

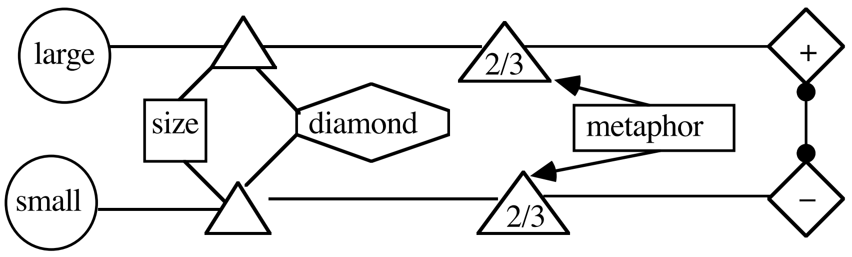

Figure 7: Control structures for scalar correspondence established by the statement hasslot (diamond: size; small (–), large (+)).

The scalar positions of property values are established by a binder node known as a two/three node [Shastri 1987] that links the value–property–category binder to the position designator. The third input to the two/three binder is supplied by the metaphor global control node (see Figure 7). A two/three node becomes active when two of its three inputs are significant.

As one of the three inputs to a two/three binder, the input from the scalar position designator, can only become active via input received from a two/three binder, the first such binders to be activated will be those receiving activation from a gated (value–property–category) binder as well as from the metaphor node. This includes all property–value pairs activated by immediate inferences during the initial priming of the network for figurative interpretation when all categories are briefly stimulated. The positional designators thus activated will then transmit their activation to all attached two/three binders. Activation then spreads from these binders to all gated binders affiliated with the positional designator, thus implementing the notion of scalar correspondence.

As shown in Figure 7, mutual inhibition is enforced between competing positional designators. All designators on a given scale participate in a winner-take-all competition, to suppress any potential inconsistencies arising from having two values with contradictory scalar positions participating in the same aspect.

Properties that are more qualitative in nature, such as shape, color, texture and their ilk, are not so readily compared. The best one can do is to group similar values with a given property, a process known as *qualitative correspondence*. For example, a marquise cut diamond, with its two pointy ends can be (imaginatively) classified along with certain tools and weapons as objects having a 'pointy' shape. This is the loosest and most tenuous form of property value mapping, and leads to the most imaginative (and least defensible!) figurative interpretations.

Qualitative correspondence is implemented in DIFICIL exactly in the same manner as scalar correspondence. Rather than having scalar designators such as + and –, shape designators such as *pointy* and *sharp* are used, but the end result is indistinguishable at the implementation level.

6 An example of figurative interpretation

The way in which all these mechanisms interact is best brought out in an extended example. Given the database established by the input file containing (in part) the following statements:

```
hasslot (diamond: size; tiny (-), small (0), large (+))

hasslot (damond: shape; marquise (pointy), diamond-cut (round))

hasslot (diamond: cost; expensive (+), cheap (-))

aspect (diamond: ornamental [default=]TRUE; brilliant, diamond-cut,
        expensive, small)

aspect (diamond: industrial [default=]FALSE; cheap, cloudy, tiny)

hasslot (person: aggression; aggressive (+), passive (-))

aspect (person: aggressive [default=]FALSE;  hostile, is-threat)

hasslot (weapon: shape; sharp (pointy), blunt (round))

aspect (weapon: aggresive [default=]TRUE; sharp, large)
```

A schematic of the network structures built by these statements appears in Figure 8. The hasslot statements create the gated binders linking the categories with their property values, and the two/three binders linking the gated binders to the named scalar position designator (eg. +, *pointy*). Each two/three binder has an additional input, not shown in the figure, from the global *metaphor* control node. All gated binders occuring in the same hasslot statement are placed in a winner-take-all competition with each other, depicted as mutual inhibition in the figure. This implements the intuition that values at different points on the same scale are mutually contradictory. The aspect statements create the inertial binders (shown as pentagons in the figure) linking the correlated category-value binders.

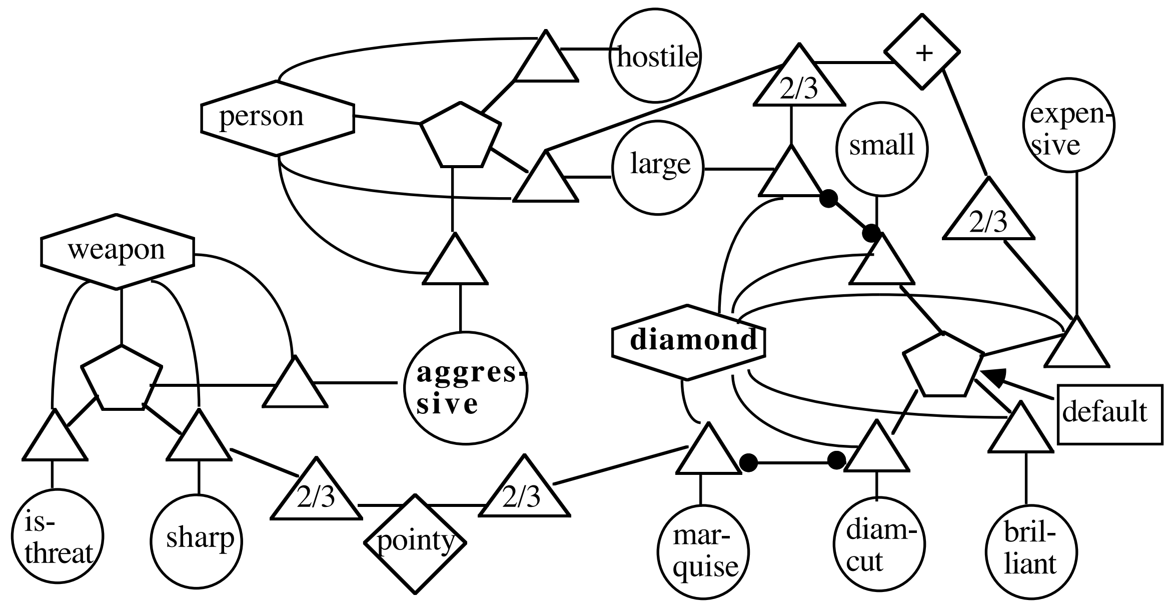

Figure 8: Simplified schematic of network structures built by the example input file pertinent to the figurative interpretation of the target phrase ' aggressive diamond'. Property nodes are not shown, nor is the *metaphor* control node with its links to all two/three binders.

This example traces the interpretation of the phrase 'aggressive diamond'. The expression is input to DIFICIL by setting the outputs of the *diamond* category node and the *aggressive* property value node to their maximum value. The excitation provided to *diamond* is sufficient to establish the default aspect of diamonds, namely, *small, diamond-cut, expensive* and *brilliant*. However, the fact that *aggressive* is active in conjunction with *diamond* will eventually cause a category error to be reported: aggression is not one of the listed properties of diamonds, so there is no inhibitory link from *diamond* to the aggression category error detection node to overcome the activation supplied by the excitatory link from *aggressive*.

When a category error is reported, the *metaphor* node is activated. This prompts the brief stimulation in parallel of all categories in the knowledge base. The default aspects of person and weapon, and all other known categories, are established at this time. The feedback within an aspect lends it stablility, so the irrelevant aspects must be actively defeated. This is accomplished by the linear decay of the stimulation signal: after a relatively short time all the categories in the network are receiving active inhibition, which is eventually sufficient to defeat these default aspects, leaving only the diamond category active. At an intermediate state of the network, the aggressive aspects of weapon (large and sharp) and person (hostile and is-threat) are active as well as a partial figurative interpretation of 'aggressive diamond'.

This partial interpretation is due to direct value transference and scalar correspondence. Since aggressive people are large and large is one of the allowable values of diamonds, activity can flow directly from the large-people binder to the large-diamond binder, mediated only through the property *size* and the value *large*. This is an example of direct transference. Scalar correspondence operates similarly: since aggressive people are large and large in the context of people is considered to be a positive scalar value (it is associated with the scalar position designator '+'), activation flows from the large-person binder to its dedicated two/three binder. All two/three binders are by this time receiving input from the global *metaphor* node, so this two/three binder now has the requisite two inputs, and fires, transmitting activation to the '+' designator node. The '+' designator node in turn supplies a second input to all two/three binders associated with it, triggering the eventual transmission of activation to such positive scalar values as *brilliant* and *expensive*.

Qualitative correspondence also occurs in this example. As aggressive weapons are sharp, and sharp in the context of weapons is considered to be of a pointy nature, the qualitative designator *pointy* is established in the same fashion as the scalar designator '+'. However, the expected effect, to select the value *marquise-cut* over the default *diamond-cut*, is not evident in the partial interpretation available midway through the interpretation. The explanation for this lies with the organization of values into aspects of the category's representation. The effect of stimulating the *brilliant* and *expensive* values of diamonds is to establish anew the ornamental aspect of diamonds, namely, *small, diamond-cut, expensive* and *brilliant*. The value *small*, however, is being actively defeated not only by scalar correspondence but also by direct transference, so is readily suppressed. The value *diamond-cut* is also receiving some inhibition from its competitor *marquise-cut*, but only through scalar correspondence, so at the intermediate stage shown the excitation of *diamond-cut* from the ornamental aspect of diamonds still outweighs the excitation of *marquise-cut* from scalar correspondence.

The final figurative interpretation is available after a total of 40 simulation steps. All concepts but the target *diamond* have been inhibited, shutting down all value–property pairs not directly associated with the target category. Conversely, all properties in the knowledge base are active to varying degrees due to the spread of activation through the abstraction hierarchy, but as properties are incidental to their values, this is unimportant. The relevant output values are: *large, expensive, brilliant* and *marquise-cut*. The ornamental aspect of diamonds is still partially active, due to the activation of two of its four component values, but two "free floating" property values, namely *large* and *marquise-cut*, are also active. It is easy to imagine that some form of node recruitment could organize these four values into a new aspect of diamonds, called the *aggressive* aspect, thus learning the new figure of speech.

References

[Goddard et al. 1988] Nigel Goddard, Kenton Lynne and Toby Mintz, “The Rochester Connectionist Simulator User Manual”, University of Rochester Computer Science Department Technical Report 233, March 1988.

[Weber 1989] Susan H. Weber, “A Structured Connectionist Approach to Direct Inferences and Figurative Adjective-Noun Combinations”, University of Rochester Computer Science Department Technical Report 289 (PhD. thesis), May 1989.

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymem ftp von ftp.dfki.uni-kl.de (131.246.241.100) unter pub/Publications bezogen werden.

Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or via anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) under pub/Publications.

The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-93-10
Martin Buchheit, Francesco M. Donini, Andrea Schaerf: Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-11
Bernhard Nebel, Hans-Jürgen Bürckert:
Reasoning about Temporal Relations:
A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-12
Pierre Sablayrolles: A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-13
Franz Baader, Karl Schlechta:
A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-14
Joachim Niehren, Andreas Podelski, Ralf Treinen:
Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-15
Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster: PLUS - Plan-based User Support
Final Project Report
33 pages

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz: Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-17
Rolf Backofen:
Regular Path Expressions in Feature Logic
37 pages

RR-93-18
Klaus Schild: Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-20
Franz Baader, Bernhard Hollunder:
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-22
Manfred Meyer, Jörg Müller:
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy:
Comparative Study of Connectionist Simulators
20 pages

RR-93-24
Rainer Hoch, Andreas Dengel:
Document Highlighting —
Message Classification in Printed Business Letters
17 pages

RR-93-25
Klaus Fischer, Norbert Kuhn: A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-26
Jörg P. Müller, Markus Pischel: The Agent Architecture InteRRaP: Concept and Application
99 pages