Variable Binding by Synaptic Strength Change

Christian R. Huyck

Middlesex University

The Burroughs, London

NW4 4BT, UK

c.huyck@mdx.ac.uk

Tel: 44-208-411-5412

Fax: 44-208-411-6943

Keywords: variable binding, Cell Assembly, short-term potentiation, long-term potentiation, synchrony

November 13th 2007

Abstract

Variable binding is a difficult problem for neural networks. Two new mechanisms for binding by synaptic change are presented. The first is based on a common learning mechanism of permanent change of synaptic weight, and the second on synaptic change which decays. Both are biologically motivated. Simulations of binding on a paired association task are shown with the first mechanism succeeding with a 97.5% F-Score, and the second performing perfectly. Bindings are erased and can be reused. A third simulation of a natural language parser implemented in neurons shows that the mechanisms can be combined to take advantage of the speed and capacity characteristics of each. Another common binding mechanism, synchrony, is compatible with these new mechanisms. All three mechanisms are compared and integrated with the Cell Assembly theory.

1 Introduction

Symbol systems have been enormously successful and it has been proposed that, at least at some level, humans are symbol processors [Newell, 1990]. In particular, symbolic systems, such as ACT [Anderson and Lebiere, 1998], have been very successful as models of human cognition. This success is

probably due to the rule based or at least rule like behaviour of humans in tasks such as natural language processing.

Unfortunately, symbolic systems also have problems with brittleness. The symbols are not grounded [Harnad, 1990] and it is difficult or impossible to learn new symbols that are not just some combination of existing symbols [Frixione et al., 1989].

These and other problems provided motivation for the rise of connectionism, particularly in the 80s. Connectionist systems are particulary good at learning, and thus may be able to learn new symbols. If the systems learn from an environment, the newly learned symbols might even have semantic content grounded in that environment.

Early connectionist systems were criticized for their inability to perform symbolic processes [Lindsey, 1988]. In particular they were criticised for their lack of compositional syntax and semantics [Fodor and Pylyshyn, 1988].

Variable binding offers an answer to these criticisms. A good variable binding solution allows for the implementation of rules; connectionist primitives can be combined, and variables instantiated as constants. If this can be done so that the result has compositional syntax and semantics, the criticism will have been answered.

Binding by synchrony is a well explored area. It is functional but can support a limited number of bindings, and the bindings have a restricted persistence. This may limit the effectiveness of a neural system, particularly as it relates to composition.

After some background for orientation, binding by synaptic change is introduced. This comes in two forms, binding by short-term potentiation (STP) and binding by compensatory long-term potentiation (LTP). Simulations that show these mechanisms are effective are described. A simulation that shows the two mechanisms being combined usefully is then described. Ramifications for memory formation speed and duration are also explored along with other issues in the discussion and conclusion.

2 Background

Humans have compositional syntax and semantics, so if systems based solely on neural models are to duplicate human behaviour, they must have compositional syntax and semantics. One way for neural systems to exhibit compositional syntax and semantics is by variable binding.

A good cognitive model should have compositional syntax and semantics [Fodor and Pylyshyn, 1988]. Standard symbolic cognitive architectures have

this compositionality, but it is more difficult for connectionist models to exhibit it.

Compositional semantics means that the semantics of a complex thing includes the semantics of that thing's constituents. So sentence 1

Pat loves Jody. Sentence 1

includes the semantics of *Pat*, *love*, and *Jody*. Compositional syntax means that the syntactic structure of complex things effects the underlying semantics. For example, the semantics of sentence 1 is different from the semantics of sentence 2.

Jody loves Pat. Sentence 2

So the semantics of a sentence must be more than the sum of its parts.

Variable binding can be used to solve these problems in a neural system by binding the semantics of constituents in a syntax sensitive way. Sentence 1 could be represented by a case frame [Filmore, 1968] for *love* where the actor slot is bound to *Pat*, and the object slot to *Jody*.

Below a more thorough background of the problem is presented and the nature of the variable binding problems is further explored. This paper is concerned with variable binding in neural models, so some popular neural models are introduced. One key component of neural models is how they learn, and a range of Hebbian learning mechanisms are discussed. This paper assumes that the neural representation of primitive concepts is a neural Cell Assembly (CA), so the concept is introduced. Finally, some existing solutions to the variable binding problem are described.

2.1 The Variable Binding Problems

The variable binding problem is a key neural network problem that involves combining representations. It is also called the binding problem [VonDerMalsburg, 1986], and the dynamic binding problem [Shastri and Aijanagadde, 1993].

The variable binding problem is really a host of problems rolled into one. These problem include:

1. Hetero-associative Memory
2. Binding Features in an Object
3. Frames
4. Rules

5. Unification

Hetero-associative memory refers to the association of an input with an output. This is a common and well understood form of memory [Willshaw et al., 1969]. Here items are combined, and each is linked to that combination. Presentation of one enables the system to retrieve the combined representation. This is roughly what Smolensky [Smolensky, 1990] refers to as variable binding. Of course restrictions can be placed on the inputs, and several features may be needed to activate the full set of items [Furber et al., 2004]. This is a weak form of variable binding, because it is not dynamic. Once bindings are set, they are not erased. Standard neural models can account for this problem using standard Hebbian learning rules to implement a form of LTP [Gerstner and vanHemmen, 1992]. Semantic nets [Quillian, 1967] are one use for associative memories.

Binding problems that are more dynamic require more complex solutions than simple permanent change of synaptic weight. Perhaps this is the basis of the term dynamic binding.

Perhaps the simplest variable binding problem is binding the features of an object. This is required when a new object is presented. If an object is composed of features, then when an object is presented, all of its features need to be bound together. One classic example is the *red-square blue-circle* example. If the system is presented with two objects, a *red-square* and a *blue-circle*, it can relatively easily activate the internal representation of all four of these features. The question is, how does the system know which pairs are bound.

The features being bound into an object do not need to be a variant of an existing object, but can be a combination that is novel for the system. A system can use a solution based on existing objects. For example, if there are two sets of 100 features that can be bound, the problem can be solved by having 10,000 stored bindings, but this number will grow exponentially with the number of features, and the number of potential combinations. This solution is just a form of auto-associative memory that is open to the problem of exponential growth.

Another example of this problem is binding parts into a whole, such as binding elements of a square lattice into rows or columns [Usher and Donnelly, 1998]. Another important variant of this problem is the what-where problem. If a system can recognise multiple objects simultaneously, and their locations, how does it know where each thing is and which things are in each location. To modify the above problem this is the *left-square right-circle* problem, but in this case location is one of the features.

Furthermore, unlike the standard associative memory task, binding features of an object has the associated difficulty of erasing the binding. After some time, *red* and *square* are no longer bound, and both may be bound to some other concept, for example *red-triangle*. This reuse problem is also a question of binding duration. As long as the binding persists, it can be used. Once it stops working, it can be reused (see section 3.3).

Another standard problem is filling in frames [Shastri and Aijanagadde, 1993, Henderson, 1994]. An example of this would be a verb frame [Filmore, 1968]. For example the verb *move* might take an actor, object and location. In the sentence *I moved the ball to the door. I* would be the actor, *the ball* the object, and *to the door* the location. When processing a sentence, the system would have to fill in the details by binding these objects to these slots. Perhaps frames are a common task for systems that use variable binding due to the compositional syntax and semantics problems mentioned by early critics of connectionist systems [Fodor and Pylyshyn, 1988]. Frames are a flexible knowledge representation format [Schank and Abelson, 1977], but they are just a form of simple pattern matching where data is used to fill in structures with variables.

Rules are an important case where variable binding is needed. Firstly, rule based systems are Turing complete [Hopcroft and Ullman, 1979], so a neural implementation of rules could easily lead to a Turing complete system. This is not particular surprising as others have shown connectionist systems to be Turing complete (e.g. [Siegelmann and Sontag, 1991]). Secondly, rules are widely used as a means of modelling human cognition (e.g. [Anderson and Lebiere, 1998, Laird et al., 1987]). An example rule would be *if X gave Y to Z, then Z possesses X*. Sequences can be implemented by rules and by connectionist systems. One system uses dynamic connections to learn sequences [Feldman, 1982]. These learned sequences are then automatically forgotten by a process of connection weight decay.

Unification is a more complex form of variable binding. This is done by symbolic systems such as language processing systems [Shieber, 1986] and logic programming. Complex structures are combined in complex ways. It is a complex form of pattern matching. This can lead to a case where a structure may be illegally combined with a subset of itself, known as the occurs check [Browne and Sun, 1999]. Unification in neural systems may incorporate soft constraints making the system more flexible [Kaplan et al., 1990, Hofstadter, 1979].

2.2 Neural Models

Connectionism refers to models that are inspired by the brain. A definition is beyond the scope of this paper, but involves the distribution of knowledge and process. Neural networks are often used as a synonym for connectionism, however, in this paper, Smolensky's terminology [Smolensky, 1988] is followed with neural networks referring to models that have a close resemblance to biological neurons and to some extent their topology.

Variable binding has been considered a problem for cognitive models based on connectionist systems. Since neural systems are connectionist systems, variable binding must be a problem for cognitive models based on neural systems. This paper is mainly interested in neural models.

There is a wide range of biological faithfulness of neural networks. Hodgkin Huxley models [Hodgkin and Huxley, 1952] are extremely faithful, breaking each neuron into several compartments and modelling interactions on a fine time grain ($< 1ms$).

At the other extreme, the Integrate and Fire (IF) model [McCulloch and Pitts, 1943] has a long standing history and is quite simple. Roughly, neurons are connected by uni-directional synapses. A neuron integrates activity from the synapses connected to it. If the activity surpasses a threshold, the neuron fires and sends activity to the neurons it connects to.

More recently, the Hopfield net [Hopfield, 1982] has been a popular system for modelling brain function [Amit, 1989]. It uses IF neurons, but adds the constraint that all neurons are connected and that connections are bi-directional, that is, they are symmetric. One advantage of this symmetry is that it allows the simple use of statistical mechanics to analyse the behaviour of large networks. Similarly, Boltzmann machines [Ackley et al., 1985] have this symmetry but add a random bias to the firing behaviour to avoid local minima.

The Hopfield net and Boltzmann machine are attractor nets. A third popular attractor net is the interactive activation model [Rumelhart and McClelland, 1982]. This keeps the symmetrical connection, but does not keep the constraint that neurons are well connected. It also moves away from the firing model to a continuous output model.

While these attractor net models are useful, they break key biological constraints. For instance, synapses are uni-directional not bi-directional. Another example is that the brain is far from well connected, with any neuron having connections to or from only a small percentage of other neurons. Consequently, results based on these attractor nets are suspect from the perspective of biological neural modelling. One key point that attractor nets

do show is that attractor states are important. This is a key point of CAs (see section 2.3).

Leaky Integrate and Fire (LIF) models are more biologically faithful than simple IF models [Churchland and Sejnowski, 1992]. In the IF model, if a neuron does not fire, it loses all its activity. In the LIF model, a neuron retains a portion of that activity making it easier to fire later. Typically, the neuron loses all its activity when it fires [Maas and Bishop, 2001].

This paper largely ignores larger brain features like hemispheres, brain areas, and even laminar architecture. Instead, it concentrates on relatively fine grained mechanisms that could function across these larger brain features.

As can be seen, there are a wide range of neural net models and the above list is by no means exhaustive. Simulations in this paper are based on a particular fatiguing LIF model (see section 4.1). How the binding algorithms relate to other neural net models is discussed in section 6.

2.3 Learning and Cell Assemblies

One of the key questions for any neural net model is how the neurons represent things. One particular concern is how the net represents symbols.

The long standing answer, attributable to Hebb [Hebb, 1949], is that a symbol is instantiated neurally by a CA. In the simulations discussed in this paper, a small subset of all the neurons represents a symbol. If many of the neurons in the CA are firing, the symbol is active. Hebb notes that the neurons in a CA have high mutual synaptic strength enabling neurons in the CA to persistently fire after external stimulus ceases.

CAs give a sound answer to the neural representation of two types of memory, long-term memory and short-term (or working) memory. Short-term memory is the firing of many neurons in a CA; this high frequency and persistent firing shows the CA is active. This firing is initially started by neurons outside of the CA or direct sensory stimuli. These cause a number of the neurons in the CA to fire. As the neurons in the CA have high connection strengths to other neurons in the CA, they cause other neurons in the CA to fire leading to ignition of the CA [Wennekers and Palm, 2000]; thus an ignited CA is an attractor state. After ignition, the neurons in the CA remain persistently active for a significant period of time on the order of seconds.

The *red-square* problem can be restated in terms of CAs. There is a CA for *red*, *blue*, *square* and *circle*. When a *red-square* and a *blue-circle* are presented, all four base CAs ignite. Somehow the pairs must be bound, so

that the system can ascertain, for example, the colour of the *square*. This binding is temporary and will only persist for a relatively small amount of time.

The formation of a CA is the formation of a long-term memory by synaptic modification. Typically, this synaptic modification is modelled as a form of LTP and long-term depression (LTD).

Hebbian learning rules provide the link between these long and short term memories. When neurons co-fire, they become more likely to fire together because their mutual synapses are strengthened [Hebb, 1949]. Eventually, this can lead to the formation of a CA. Hebbian learning is local; it occurs between two neurons that are connected and takes information based solely on these neurons. Hebbian learning is based on the neurons' firing behaviors. Typically the synaptic weight is increased when both the presynaptic and post-synaptic neurons fire. For all but the simplest forms of Hebbian learning, there is an associated form of forgetting that is, somewhat oddly, called anti-Hebbian learning. Here if one neuron fires and the other does not, the synaptic weight is decreased [White et al., 1988]. This prevents the weight from growing without limit. There is significant biological evidence for Hebbian learning [Brunel, 1996]. Moreover, as this learning is based on pairs of neurons, these biological experiments are relatively simple, so there is good reason to believe that Hebbian learning does occur.

However, it is not entirely clear which algorithm or algorithms are used in biological systems. There are a range of Hebbian learning algorithms that follow the above definition, but differ from each other. The simplest rule merely increases the synaptic weight when both neurons co-fire. There is no anti-Hebbian rule, and the weight may be clipped at some value [Sompolinsky, 1987]. Another rule, the simple correlatory learning rule sets the synaptic weights to the likelihood that the post-synaptic neuron fires when the presynaptic neuron does [Huyck, 2004].

Formal computational treatment of Hebbian learning shows that it is capable of independent component analysis [Fyfe, 2005] and principal component analysis [Oja, 1982]. Oja's rule [Oja, 1982] uses existing synaptic weights to modify the weight change. If the same input patterns are repeatedly presented, the weights will become a unit vector on the input space that represents the principal component of that input.

Timing is also important to learning. The Hebbian rule involves firing of neurons at the same time. In a model that uses continuous time, the same time requires some degree of flexibility. Work on Spike Timed Dependent Plasticity (e.g. [Gerstner and Kistler, 2002]) adds another dimension to the complexity of Hebbian rules.

The interaction between learning and firing leads to a complex dual dynamics [Hebb, 1949]. Once a CA is learned, it is hard to forget because any activation of it strengthens its intra-neural connections; this is a form of the stability plasticity dilemma [Carpenter and Grossberg, 1988]. Similarly, it is difficult to do anything with a CA until it has formed.

Hebbian learning rules are used to form CAs, the neural basis of concepts. Binding is not necessarily related to Hebbian learning, but if CAs can be appropriately bound, the resulting system can have compositional semantics and syntax. What mechanisms can be used to bind CAs together?

2.4 Solutions to the Problem

Typical neural simulations resolve the binding problem by synchrony, but neural binding has also been simulated using long term changes in synaptic weights. Other connectionist models bind using these mechanisms, though other mechanisms have also been used.

2.4.1 Binding via Synchrony

The most commonly used form of binding in simulated neural nets is binding via synchrony. This requires neurons that are bound together to fire together. So if two neurons are bound, they might fire at times 0.1, 0.3, 0.6, 0.9 and 1.1; then might repeat the pattern at 1.5, 1.7, 2.0, 2.3 and 2.5. Of course there is some room for variation, and the binding usually applies to a much larger number of neurons than two.

A good example of this is SHRUTI, a non-neural connectionist mechanism [Shastri and Aijanagadde, 1993]. In this model different sets of concept nodes are bound together by firing at roughly the same time. Rules can be instantiated in the nodes, and these can continue to propagate the bindings to new items.

SHRUTI has been used to develop, among other things, a syntactic parser [Henderson, 1994]. Here synchrony is used to bind slots and fillers. Unfortunately, the system only allows 10 bindings, so only relatively simple sentences can be processed.

SHRUTI actually assigns synchronous firing states, but, there is significant evidence for synchronous firing in biological neural systems (e.g. [Abeles et al., 1993, Bevan and Wilson, 1999, Eckhorn et al., 1988]). Some really convincing evidence that synchronous firing is used for biological binding is provided by a study that shows how binding is facilitated by stimulus that is presented synchronously [Usher and Donnelly, 1998].

Simulated neural models of binding via synchrony are also available (e.g. [Wennekers and Palm, 2000]). Networks of spiking neurons are used to segment a visual scene into different objects based on the firing timing of neurons associated with those objects [Knoblauch and Palm, 2001]; a scene with a triangle and a square is presented, and neurons associated with the square fire together and the triangle neurons fire together but at at different times from the square neurons. Spiking neurons are also used to parse simple text [Knoblauch et al., 2004] using binding via synchrony.

One major problem with binding via synchrony is the number of bindings that it allows. One connectionist parser [Henderson, 1994] is limited to 10 bindings. It is not entirely clear how many bindings biological neural systems allow, but as more bindings exist, there is an increased likelihood that closely related patterns will coalesce thus incorrectly combining the two sets of bound items.

2.4.2 Binding via LTP

Another option is to bind by changing synaptic weights. An earlier version of the work presented in this paper used a fatiguing LIF neural model to implement rules to count from one number to another [Huyck and Belavkin, 2006]. A Hebbian learning rule is used to change synaptic weights permanently as a form of LTP.

Unfortunately, a general binding solution based on LTP would interact with the stability plasticity dilemma [Carpenter and Grossberg, 1988]. Roughly the dilemma is how is it possible to add new knowledge without disrupting existing knowledge in a neural net [Lindsey, 1988]. With binding, base CAs would need to be stable, bindings would need to be plastic, and new CAs would still need to be formed. Any system that allowed a LTP based binding to be erased, could have the problem of erasing the base CAs that are being bound.

2.4.3 Other Connectionist Binding Mechanisms

There are other connectionist mechanisms for binding. One standard mechanism is to create a new binding element for each possible binding. As mentioned earlier (section 2.1), this has the problem of combinatorial explosion.

Another simple mechanism is to merely combine the bound representations, but this lacks compositional syntax. An example is Tensor Product binding [Smolensky, 1990] which forms a type of cross product of the

variables that are being bound. It could not solve the *red-square* problem because it could not say which object was which colour.

Dynamic connections are used to store bindings [Feldman, 1982]. These connections are activated by a pair of inputs, and then persist for a considerable period. The persistence automatically decays allowing the node to be reused later.

While some work has been done on binding via synaptic change in neural systems, most neural binding work has been done using synchronous firing. Some non-neural connectionist work is relevant to the problem. However, the possibility of binding via synaptic change is an under explored area.

3 Binding via LTP and STP

There is strong evidence that distinct features that co-occur in a particular object fire synchronously [Usher and Donnelly, 1998, Abeles et al., 1993, Eckhorn et al., 1988]. This is solid evidence for binding via synchrony[1]. However, synchronous binding has a problem with capacity and a problem with duration of binding.

It is not entirely clear how many bindings can be maintained by a network at any given time, but each binding must have its own unique pattern of synchrony. In one connectionist implementation, this limit was set to 10 separate bindings [Henderson, 1994]. When parsing a single sentence many more bindings may be needed, and people can retain information about more than one sentence at a time showing the need for yet more bindings. Of course humans do more things than process language and bindings would be needed for other things like object recognition. Since CAs cross brain areas [Pulvermuller, 1999], orthogonalizing domains (e.g. vision and language) is not a viable solution.

Also the synchronous binding only persists as long as the CAs are active. Once they stop, the binding is lost. While it is not entirely clear how long CAs persist, there is a wide range of times that a binding might persist.

Even if binding via synchrony occurs in the brain, this does not mean that there are not other types of binding. A different mechanism for binding, as is shown below, is change in synaptic weights. There are at least two variants of known biological synaptic weight change, LTP and STP.

[1]It is not conclusive proof. Synchronous firing may simply be an emergent property of the neural representation of the new object as it is an emergent property of standard long-term CAs [Wennekers and Palm, 2000].

3.1 Binding via LTP

One possible solution to the binding problem is permanent synaptic change; biologically this is LTP and LTD. Objects are bound using synaptic weight change. These weight changes remain until future learning causes them to change.

This relates closely to the stability plasticity dilemma [Carpenter and Grossberg, 1988]. The problem is that some memories need to be stable, while new memories must be formed. For LTP to be able to solve the binding problem, the binding must be able to be erased. If the same mechanism is used to form the initial memories and to do the binding, something else must prevent the initial memories from being erased when the bindings are erased.

3.2 Binding via STP

A more novel approach is to use STP to bind. In this case, the memories are bound using STP. As the STP is automatically erased, so is the associated binding.

There is extensive evidence that STP occurs in biological neural systems [Hempel et al., 2000, Buonomano, 1999]. It is still a type of Hebbian learning, based on the firing behaviour of the neurons the synapse connects with cofiring increasing the synaptic weight. However, unlike LTP, the change is not permanent.

3.3 Properties of Binding Mechanisms

Different binding mechanisms have different properties. Four important properties are:

1. Persistence of binding
2. Number of bindings supported
3. Items bound by one binding
4. Speed to bind

Persistence of binding has already been mentioned. Hetero-associative memory (section 2.1) has a binding that persists forever. At the other extreme, binding via synchrony only persists as long as at least one of the bound items is firing. This leaves a wide range of times that a binding might persist.

The number of bindings supported has also been mentioned. The solution of forming a binding node for each possible binding is impractical due to capacity limitations. Binding via synchrony limits the number of possible bindings that a system can store. Another mechanism might be based on specific binding nodes or CAs. Each node might be used to bind any one item, and in this case, as many bindings as nodes can be implemented. So, in the case of verb frames, each slot of each verb might be a binding node. The slot fillers could be simple nouns, or they could consist of other verbs, in for example the case of sentential complements, to allow an arbitrary degree of complexity. Of course complex noun phrases would also need binding sites.

Another capacity dimension is how many things can be bound by one binding. Binding by synchrony supports multiple things being bound together. A different mechanism might allow only one or two items to be bound in a particular binding.

Finally, time to bind is an important consideration. How long must CAs be coactive before they can be bound. Binding via synchrony is very fast and can occur within tens of ms [Wennekers and Palm, 2000]. Binding via LTP would take much longer.

Next some simulations involving binding via synaptic change are described. These show that both mechanisms can be used effectively for binding.

4 Simulating Binding with LTP and STP

Simulations based on fatiguing LIF (fLIF) neurons show that both binding by compensatory LTP and binding by STP are successful. First the fLIF model and general topology are described. Next, a simulation of binding by compensatory LTP is described; the task is a simple paired association task similar to the *red-square* problem (section 2.1). Finally, a simulation of the same task using binding by STP is described.

4.1 fatiguing LIF model

The neural model that is used for the simulations described in this paper is the fLIF neuron. The simulator runs in discrete time cycles. Each neuron has two variables associated with it, an array of synapses, and each subnet has four constants associated with all its neurons.

The two variables associated with each neuron i are fatigue F_i and activation A_i. As neurons fire, activation is passed to a neuron and is accumulated

in A_i.

The first constant is the firing threshold, θ. A neuron i fires if

$$A_i - F_i >= \theta \tag{1}$$

If the neuron fires, it loses all its activation.

If a neuron does not fire, some of its activation leaks away. This leak, or decay, is the second constant D where $D > 1$. Ignoring external input, activation of neuron i at time t is

$$A_i^t = A_i^{t-1}/D \tag{2}$$

When neuron i fires, it sends activation (or inhibition) along its synapses to another neuron j according to synaptic strengths w_{ij}. The neuron is an integrator, so it accumulates activity from the synapses that pass energy on to it. So, given P_j, the prior activation of neuron j, either 0 or equation 2, the activation at time $t + 1$ is

$$A_j^{t+1} = P_j + \sum_{i \in V_i} w_{ij} \tag{3}$$

where V_i is the set of all neurons that fired at time t.

These equations describe a LIF model [Maas and Bishop, 2001]. The fatigue variable is incremented by the third constant F_c in a cycle when the neuron fires, and is decremented by the fourth constant F_r in a cycle when the neuron does not fire. This makes it more difficult for neurons to fire the longer they are active. Fatigue is a property of biological neurons [Kaplan et al., 1991].

The model has a loose link with time in biological neurons. The model does not incorporate conductance delays or refractory periods. These behaviours all happen in under 10 ms., so each given cycle can be considered to be roughly 10 ms. Consequently, each neuron emits at most one spike per 10 ms. of simulated time. This is a shortcoming of the model, but enables efficient simulation of hundreds of thousands of neurons on a standard PC.

The connectivity of the network, and subnets is also important. The system uses neurons that are either inhibitory or excitatory but not both. While there is some debate over the biological behavior, this follows the strict constraint of Dale's Law [Eccles, 1986]. In the simulations described in this section, the ratio is 4 excitatory to 1 inhibitory as in the mammalian cortex [Bratenberg, 1989].

Like the mammalian brain, excitatory neurons are likely to connect to neurons that are nearby. In the simulations described in this section, excitatory neurons also have one long distance axon with several synapses. So a

neuron connects to nearby neurons and to neurons in one other area of the subnet. These connections are assigned randomly, so each new subnet is different from other subnets with the same number of neurons. Since distance is relevant, the overall subnet is toroidal to avoid edge problems. Equation 4 is used for connectivity.

$$r < (N * .8) \rightarrow connect \tag{4}$$

It is initially called for each neuron with N (distance) of one for three adjacent neurons. It is subsequently called recursively on all four adjacent neurons with distance increasing one on each recursive call. The recursion is stopped at distance 5. r is a random number between 0 and 1. The long-distance axon uses the same process though starts with distance 2. Inhibitory neurons are connected randomly within a subnet. There are approximately 60 synapses leaving a neuron to other neurons in the subnet, for both inhibitory and excitatory neurons.

4.2 Simulating Binding by Compensatory Long-Term Potentiation

The first set of simulations being reported in this paper involve binding via permanent changes of synaptic strength. This involves a compensatory Hebbian learning mechanism [Huyck, 2004] that makes permanent changes to increase a synapse's strength, akin to LTP, and permanent changes that decrease the strength, akin to LTD. The simulation also makes use of spontaneous neural activation, a known biological phenomena [Amit and Brunel, 1997], to support erasing bindings

The gross topology is shown in figure 1. There are three subnets called the *letter* subnet, the *number* subnet, and the *binding* subnet. The *letter* and *number* subnets are trained to contain 10 CAs each. Both nets consist of 1600 neurons and the *binding* subnet has 400.

The excitatory neurons are connected in a distance biased fashion (Eq. 4) within the subnet, and the inhibitory neurons are connected randomly within the subnet. Each *bind* neuron has 15 connections to both the other subnets. The neurons of the base subnets, *letter* and *number*, have 16 connections to the *bind* subnet and all inter-subnetwork connections are randomly assigned. The initial weights are initialized to a number close to 0.

The compensatory learning mechanism is another type of Hebbian learning. It forces the total synaptic strength leaving a neuron toward the desired weight, W_B.

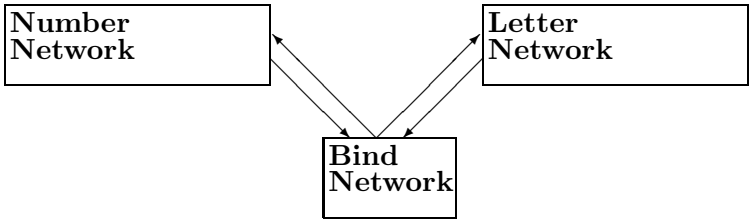

Figure 1: Gross Topology of LTP Binding Simulation

The compensatory rule modifies the correlatory learning rules to include a goal total synaptic weight W_B. R is the learning rate, which is 0.1. P is a constant and must be greater than 1. The larger it is, the less variance the total synaptic weight has from W_B. Both W_B and R are constants associated with a particular subnet. Equation 5 is the compensatory increase rule and Equation 6 is the compensatory decreasing rule; that is Equation 5 is a Hebbian rule and Equation 6 an anti-Hebbian rule. W_B is a constant which represents the desired total synaptic strength of the pre-synaptic neuron, and W_i is the current total synaptic strength. So, when the two neurons co-fire there is an increase in synaptic weight corresponding to Equation 5. If the pre-synaptic neuron fires and the post-synaptic neuron does not fire, the weight is decreased according to Equation 6.

$$\Delta_+ w_{ij} = (1 - w_{ij}) * R * P^{(W_B - W_i)} \tag{5}$$

$$\Delta_- w_{ij} = w_{ij} * -R * P^{(W_i - W_B)} \tag{6}$$

Correlatory learning is important in the erasing process described below.

A summary of the value of the constants can be found in table 1.

Name	Symbol	Base Net	Bind Net
Threshold	θ	4	7
Decay	D	1.5	5
Fatigue	F_c	1.0	1.0
Fatigue Recovery	F_r	2.0	2.0
Saturation Base	W_B	21	28
Compensatory Base	P	1.3	1.3

Table 1: Network Constants

During the entire run, there is spontaneous activation in the binding net. Spontaneous neural firing is a property of biological neurons [Abeles et al., 1993, Amit and Brunel, 1997, Bevan and Wilson, 1999]. It has been proposed as a mechanism for erasing memories [Huyck and Bowles, 2004].

In this simulation, some neurons may be spontaneously activated. This is modelled by the selection of a random number $0 <= r < 1$ for each neuron in each cycle. If the $r < 0.03$ the neuron is spontaneously active. So, roughly 3% of neurons in the bind subnet fire spontaneously each cycle.

The simulation first learns the base *number* and *letter* CAs then one of each is randomly selected to be bound. The task is a simple paired association task similar to the task performed in earlier connectionist simulations [Feldman, 1982] and those done in psychological experiments [Sakai and Miyashita, 1991]. Once bound, the binding is tested, followed by a test for an unbound *letter* and *number*. The binding is then erased by spontaneous activation; and the tests are rerun. For measurement, this binding, testing, erasing, and retesting process is repeated 10 times on each of 10 different networks.

The base CAs are learned by merely presenting components of them. As both the base nets consist of 1600 neurons, they can be divided into 10 orthogonal CAs of 160 neurons each. 50 randomly selected neurons of a particular CA are selected and presented for 10 cycles. This is akin to clamping, but these neurons are given $\theta * (1 + random)$ units of activation. After fatigue has accumulated they may not fire. After the 10 cycles of activation, the network is allowed to run for 40 more cycles. It is then reset with all activation and fatigue zeroed. Then a new CA is presented. These 50 cycles of activation, run-on, and short term variable resetting are called an epoch.

Each base CA is presented in a rotation so that all CAs are presented once every 1000 cycles. The complete training phase is 20000 cycles so that each base CA is presented 20 times. Note that spontaneous activation in the *bind* net continues throughout this time.

Figure 2 shows the formation process. A network is created with synaptic weights near 0. It is then trained. At the 45th cycle of each training epoch, the number of neurons in the presented CA is measured. This is averaged over the presentation of each of the 20 base CAs, and over 10 networks. The number of neurons outside the desired CA firing was also measured but was always zero. This measurement of persistence shows a rapid increase in neurons firing toward the end of each training epoch. There is a gradual increase after the 5000th cycle. Note, that the maximum number of neurons that could be firing is 160, but fewer are firing due to fatigue. By cycle 20000, the base CAs are quite persistent.

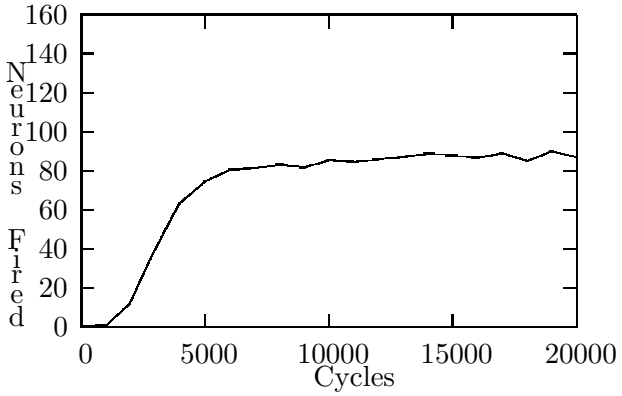

Figure 2: CA Formation

After the training phase, the epoch duration is lengthened to 1000 cycles for the binding phase. A randomly selected *letter* CA and a randomly selected *number* CA are presented simultaneously. Again, 50 neurons from both are selected at random and presented for 10 cycles. As the CAs have formed, these almost always persist for the duration of the binding epoch.

As ever, the *bind* subnet is spontaneously activated during this phase. Gradually throughout this period the synaptic weights between the subnets increase. When binding is successful, neurons in the *bind* subnet fire due to input from the ignited *number* and *letter* CA. This in turn causes the inter-subnet synapses to increase. In essence, a new CA is being formed and it includes neurons from all three subnetworks.

It is crucial that two CAs in the base subnets are simultaneously active. This is similar to the mechanism used for node activation by dynamic connections [Feldman, 1982]. Along with the spontaneously active bind neurons, these base neurons provide sufficient activation to fire some of the neurons in the bind subnet. Firing these base neurons causes the mutual synaptic strength between them and the base neurons to increase. This leads to further neural firing in the bind subnet. By the end of the binding epoch, a CA has been formed that includes the binding neurons, and this composite CA can be reignited at any time over a significant period of time.

In the second epoch, the bound *number* is presented, and in the third, the bound *letter* is presented. When successful, this leads to ignition of the binding CA and the opposite base CA. This further reinforces the inter-subnet synaptic strengths, improving the binding.

In the fourth epoch a randomly selected unbound *number* is presented,

and an unbound *letter* is presented in the fifth. The correct result here is that no neurons in the opposite subnetwork fire.

The synaptic strength from the binding subnet that supports the binding is being reduced during the test unbound phase, but four further epochs of no base presentation are run to allow the binding to be sufficiently erased. The synapses from the binding subnet that support the binding move rapidly toward zero due to the interaction of the spontaneous firing with the compensatory learning rules.

Synapses from the *bind* subnet to the base subnets are erased during the period of no presentation. During this period, neurons in the *bind* subnet fire, but no neurons in the base subnets fire. Consequently, the weights are reduced toward 0.

However, the synapses to the *bind* subnet from the base subnets are not changed during the testing of unbound items or during the period of no presentation. Instead, these synapses are reduced by the compensatory learning mechanism during the next two test epochs.

The synaptic weights from neurons in the base subnets to the *bind* subnet do not change between the last binding test, and the first bind retest. Why then does the presentation of the here to fore bound item not cause the *bind* subnet to activate as it had done during the presentation in the second and third epochs?

Firstly, there are fewer neurons firing in the just bound item. This is due to the loss of intra-subnet synaptic strength during the binding. Secondly, there is little initial feedback from the bind node since its neurons no longer have much synaptic weight to the recently bound item. During this initial phase the synaptic weights in the just bound item are changing. The weights to the bind node are being reduced while the weights within the just bound item are increasing. There is only a small part of the parameter space where this difficult task can be solved.

Finally, there are four tests to assure that the binding has been erased. The formerly bound *number* and *letter* CA are presented, followed by the formerly tested unbound *number* and *letter*.

For each network, this series of tests was run 10 times. It was run on a total of 10 networks. When the testing epoch length was 1000 cycles, 192/200 or, 96%, of the binding tests were successful, and 595/600, or 99.2%, tests of unbound CAs were successful. These measurements can be combined using a standard F-score $(2 * Bound * Unbound)/(Bound + Unbound)$. The F-score is 97.5%.

The length of the binding period is important. Substantial variations from the binding period of 1000 cycles causes decreased performance. Figure

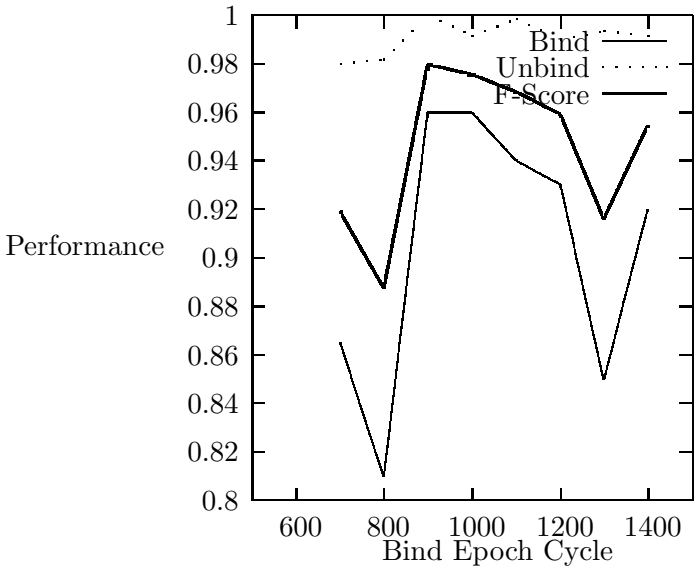

Figure 3: Effect of Binding Period on Binding

3 shows this. Performance is best around 1000 cycles, and trails off when it is shorter or longer.

It is important that the base CAs must be formed and solid before binding occurs. They need to be solid so that they can fully participate in the binding process. This solidity is supported by a low firing threshold ($\theta = 4$) and a low decay rate ($D = 1.5$); together these enable rapid recruiting of new neurons in a few presentations and high activity in the formed CA.

During base training, the binding area should not form a CA. That is, no single CA should be able to recruit many neurons from the binding area. Instead, two base CAs are needed to recruit neurons in the binding area. Consequently, little activity should be retained in the binding area ($D = 5$), and it should be difficult to fire a neuron in the binding area ($\theta = 7$). As the binding area needs to be quickly recruited when two CAs are active, the total synaptic strength is high ($W_B = 28$) so that the connections to the other areas and within can be quickly formed.

This simulation fits into a rather small part of the parameter space. This is largely due to the rather precise way that the synapses from the base CAs to the binding subnet are erased. There is no spontaneous activation in the base subnets so the connections remain the same during the erase epochs. However, during the binding epoch, the synapses between neurons in the

base CAs being bound lose strength to the synapses to the binding net. The loss of the feedback from the binding net after the erase epochs is enough to prevent the ignition of the binding net when the bound base CA ignites.

Since this is so precise, minor changes to parameters cause a rapid decrease in performance. Changing the number of synapses from each base neuron to the bind neurons from 16 to 15 gives Bound/Unbound/F-Score results of 87%/95.5%/91.1%, and changing the number from 16 to 17 gives B/U/F results of 78.5%/94.8%/85.9%. Similarly, changing the base nets' desired total synaptic strength (W_B) from 21 to 20 gives B/U/F results of 45%/99.2%/61.9%, and changing it from 21 to 22 gives results of 80.5%/89.8%/84.9%.

This is a particularly difficult binding simulation because there is no spontaneous activation in the base nets to facilitate erasing the binding. However, the lack of this spontaneous activation allows those CAs to persist indefinitely. Additionally, binding still works quite effectively.

4.3 Simulating Binding by Short-Term Potentiation

Another option to implement variable binding by synaptic modification is to change the basic mechanism of synaptic change. LTP and LTD require the synaptic weight to remain unchanged until there is another application of one those rules. Since synaptic change is caused by neural firing, the synaptic weights will remain unchanged until the appropriate neurons fire.

Another option is to have the weights automatically revert to zero over time. A rule that did this would be akin to STP. Note that the rule is still Hebbian in nature, changing the synaptic weight based solely on the firing behavior of the two neurons that a synapse connects, but in this case, the weight also changes toward 0 when there is no firing.

The binding via STP simulations reported below adhere closely to the simulations describing the binding via LTP. The topology is similar, though the binding subnet is no longer needed. A different class of neuron is introduced, and its learning rule is obviously different, though the compensatory rule is still used on the remaining neurons.

The simulation makes use of a new type of neuron, termed a fast-bind neuron. The basic properties are all the same, but synapses leaving these neurons change their weights based on a different mechanism.

The topology of the *number* and *letter* subnets is the same as in the LTP simulations, except every 10th neuron is a fast-bind neuron and these are all excitatory. Of the remaining neurons, 70% are excitatory and 20% are inhibitory. Each fastbind neuron is connected to 60 randomly selected

neurons from the opposite subnet.

The learning rule for fast bind neurons is the simplest type of Hebbian learning. For each synapse leaving a fast bind neuron, if that neuron fires in the same cycle as the neuron the synapse is connected to, the strength increases by the learning constant, which is 0.1. The weight is clipped at 1.

The rule for reducing synaptic weight is equally simple. If the neuron does not fire in a cycle, all synapses leaving it have their weight decreased by a constant (in this case 0.004).

So, a maximally weighted synapse, will return to 0 after 250 cycles of inactivity. Similarly, a minimally weighted synapse will go to 1 after 10 cycles of pre and post-synaptic co-firing.

The constants of the *letter* and *number* nets are the same as those in the LTP experiment; these are shown in Table 1. The training length is the same, 20,000 cycles, and the procedure is the same. The testing patterns are the same: binding epoch, two bind test epochs, two unbound test epochs, four epochs with no presentation, then two more tests of the formerly bound CAs, and two tests of the unbound CAs.

When the epoch lengths are 50 cycles, the system performs perfectly over 10 bindings on each of 10 nets. That is all 100 bindings were successful, and all associated 100 erasings were successful. The Bound/Unbound/F-Score results of 100%/100%/100%. The bindings only need 10 cycles to be fully established, and as they are given 50, they are firmly established. Similarly, only 250 cycles are needed for the bindings to be fully erased. As there are two unbound test epochs, and four non-presentation epochs after the binding, there are 300 cycles or erasing. So, they are also perfect.

5 Natural Language Parser

One of the advantages of these two mechanisms is that they can be combined. A system using a combination could have bindings that are quickly learned and erased using STP, and bindings that are learned and erased more slowly using LTP.

Another thing to note about the two binding mechanisms presented above is that their bindings differ in the number of items that can be bound together (see sections 3.3 and 6). The STP binding algorithm presented above allows a practically unlimited number of items to be bound in one binding. So in the number letter binding simulation above, it would be simple to bind the number *1* to both *A* and *B* simultaneously. Presentation of any one of the three would lead to ignition of all of them. Indeed a number

could be bound to all of the letters.

The compensatory LTP binding algorithm limits the strength of connections and thus limits the number of items that can be bound into one binding. It is probably not possible to bind the number *1* to all of the letters simultaneously via compensatory learning with parameters like those from section 4.2.

Binding many items in one binding, like the binding via STP algorithm from section 4.3, is not always desirable. Binding via STP is used in the Cell Assembly roBot version 1 (CABot1) agent in parsing commands. CABot1 is an agent in a video game that assists the user. The agent is implemented entirely in fLIF neurons. It consists of vision subnets, planning subnets, an action subnet, a control subnet, and parsing subnets. The parsing subnets take the user's commands in natural language and parses them. The semantic result then leads to goals being set within the agent.

The parsing component uses a verb frame structure [Filmore, 1968] to store the semantics of the sentence being parsed. In particular, each verb has slots that are filled by arguments (noun and prepositional phrases). These verb slots consist of fast bind neurons, and are bound using the STP mechanism described in section 4.3. Unfortunately, that means if a verb has two or more slots filled, and the verb is active along with all its slots and slot fillers, each slot will be bound to each filler. Subsequent activation of a single slot activates all fillers. As the semantics of an individual slot may be needed, this cross talk is not desirable.

An example of this problem would be sentence 3. In the CABot1 parser, this never occurs because the commands that are considered never have two slots filled.

Move it toward the stalactite. Sentence 3

The CABot1 parser was modified to allow this sentence to be presented, and to check the individual slots. This was a minor modification, but indeed confirmed that the system both parses correctly and that both slots are bound to both fillers. The system assigns *it* to the object slot and *toward the stalactite* to the location slot. This is evident by the sequence of rule firings.

1. $VP \rightarrow$ *VP NP-obj*
2. $PP \rightarrow$ *prep det*
3. $PP \rightarrow$ *PP noun*
4. $VP \rightarrow$ *VP PP-loc*

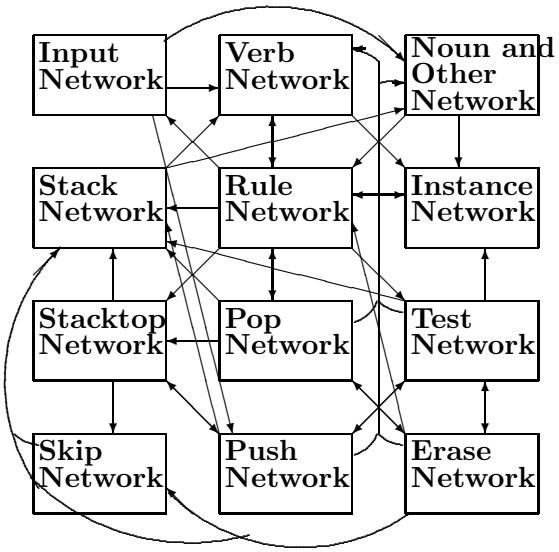

Figure 4: Gross Topology of the CABot1 Parser

5. $S \rightarrow$ *VP Period*

Due to space limitations and focus, a full description of CABot1 is not included. A more thorough description of the parser can be found in [Huyck and Fan, 2007]. A brief discussion of the CABot1 parser follows with particular attention to the use of binding.

The CABot1 parser consists of 13 subnets. 12 are shown in figure 4 as the gross topology of the noun network and other network is the same.

The parser works by taking symbolic input, and converting it into input CAs in the Input subnetwork. It then uses the *Push* subnet to push these onto the stack. The stack consists of several stack elements represented by CAs. Each element consists of standard fLIF neurons (with learning turned off), and fastbind neurons. These fastbind neurons have connections to the *Verb*, *Noun* and *Other* subnets. When an item is pushed onto the stack, the item to be pushed and the current stack CA are simultaneously ignited and they are bound. Similarly, when the stack is popped, the *Erase* subnet is run. It takes over 200 cycles to run, and ignites the items on the stack in sequence while being run. Thus the items still on the stack remain bound because the synaptic weights have been refreshed by co-firing, and the item that was just popped loses its binding.

When the *Test* subnet is run (e.g. after a push or an erase), the items on the *Stack* are ignited in sequence. These provide input to the *Rule* subnet.

If the appropriate items are ignited in sequence, a rule CA ignites and thus is applied. Some rules merely ignore the input and call Pop (e.g. → VP Period).

Other rules update the semantics. The VP → VP PP-loc rule ignites the verb via the stack, and its *location* slot. The slot contains some fastbind neurons. As the PP is already active, the *location* slot is filled by the PP. Note that the semantics of the PP and the nouns are stored in the *Instance* subnet, and the noun instances provide no input to the rules. So, when the Noun is removed from the stack, its semantics remain in the instance, but there is no longer a contribution to rule firing.

Parsing is completed when the S → VP Period rule is applied. In CABot1, parsing is followed by goal setting. In the modified version, the semantics can be tested.

The semantics of the slots can be tested by activating the slots after parsing and testing the instances that ignite. After Sentence 3 has been parsed, activation of *move*'s *object* slot or *location* slot both indicate that the *it*, and *stalactite* are active. A test of 100 successful parses of this sentence was done. In each case, the *location* slot ignited both *it* and *stalactite* instance, as did the *object* slot.

This is because the fastbind neurons in the slots have synapses to both items as well as the other nouns. When any of the nouns are coactive with the slots, they will be bound. As both *it* and *stalactite* are coactive with the slots, they are both bound to them.

Consequently, a new parser was implemented that made use of both binding mechanisms. This was a variation on the CABot1 parser that used compensatory LTP learning to bind the slots to their fillers.

A new subnet was introduced, the *VerbSlot* subnet, consisting of normal fLIF neurons. In this subnet, and no other subnets, compensatory learning was used. The slots were removed from the *Verb* subnet, though fastbind neurons were retained to bind the verbs to their slots. The connections from the *Verb* subnet to the *Instance* subnet were removed. Connections from the *Verb* subnet to the *VerbSlot* subnet and from the *VerbSlot* subnet to the *Instance* subnet were added.

Again, a complete description of the parameters involved in even the binding subnets is inappropriate here, but some values may help. The compensatory base (P) was 2, the saturation base (W_B) was 20, and the learning rate was 0.01. Connectivity is also very important. The *VerbSlot* subnet consisted of eight slots, two for each verb, with each slot consisting of 100 neurons. An additional 400 neurons in the net were in the sink (see below), but these had no connections leaving them. The slot neurons had 80/20 ex-

citatory/inhibitory ratio. Inhibitory neurons had 3 synapses to the opposite slot that had an initial weight of -0.2; as these were the only inhibitory connections from the VerbSlot net, inhibition was not particularly important in the simulation. Excitatory neurons had 20 connections to other neurons in the CA with an initial weight of 0.6. Each excitatory neuron also had 160 connections to sink neurons with an initial weight of 0.05. Each excitatory slot neuron had 20 connections to each of the noun instances, and the initial weights were 0.01.

The basic parsing process remained unchanged. Input text CAs were ignited from text input when the *Push* subnet completed. The fLIF neurons propagated activation around the network going through a test, apply, push, pop, erase circuit. Activity was completed when the $VP \rightarrow VP\ Period$ rule was applied.

When this activity was completed, activation and fatigue in the nets was cleared, and the semantics of the sentence were tested by activating the first stack element. This was done by externally stimulating the first stack element for 10 cycles. The measurement was made by counting the number of neurons on in each CA. As expected, the semantics were correct.

When this semantic test is successful on Sentence 3, it requires that both verb slots are active simultaneously. This would allow crosstalk between the slots. In the CABot1 parser, this coactivation of slots also occurred during parsing. For the modified parser, this was reduced, but did occur after the application of the $VP \rightarrow VP\ PP\text{-}loc$ rule.

After the semantic test there was a slot test. During the slot test, learning was turned off. The individual slots were activated for 10 cycles, and the instance CAs were measured. In both cases, only the correct instance was active and they were active at a high level with almost all 150 neurons firing.

Learning was turned on again, and the system tried to erase the binding. Of course, spontaneous activation was on the whole time. Unfortunately, this was not sufficient to erase the binding (see section 6). So, a set of semantically meaningless neurons in the *VerbSlot* subnet were externally activated for 2000 cycles. In combination with spontaneous activation, these neurons acted as a sink of synaptic strength. Strength in the bound neurons was pulled away from the binding, and devoted to neurons in this sink. This pulling was an interaction between the firing dynamics and the compensatory learning rule. This presentation was followed by 8000 cycles of pure spontaneous activation.

Figure 5 shows the behaviour of synaptic weights from *move*'s *object* slot in the *VerbSlot* subnet. The top solid line is the average total synaptic

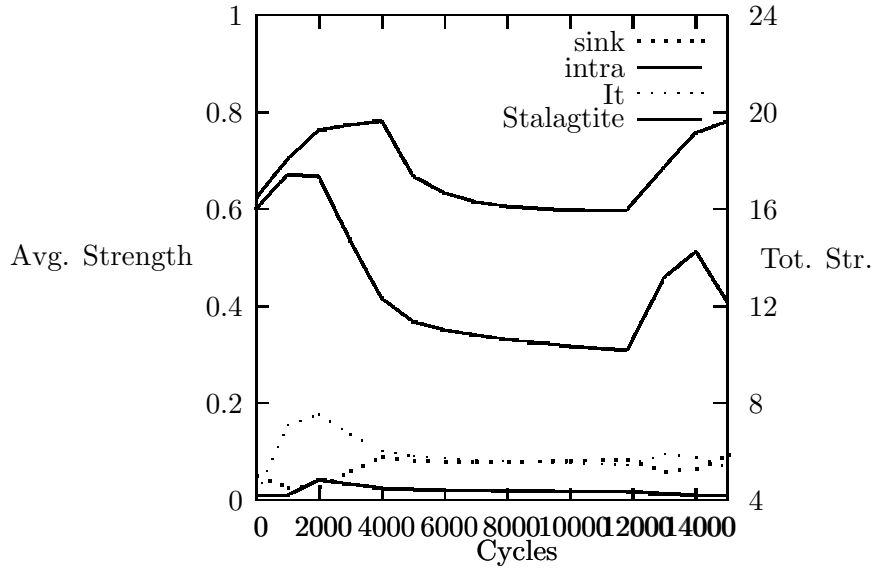

Figure 5: Avgerage Synaptic Weight of Object Neurons

weight of one of the *move object* slot's neurons; it is read off the right axis. The next solid line is the average weight of synapses from the slot to other neurons in the slot; this and the bottom three lines are read off the left axis. The bottom three lines refer to the average weight from the bind slot to the respective filler neurons showing the average synaptic weight throughout time.

Initial weights are as stated above at cycle 0. By cycle 1000, the *VP* $\rightarrow$ *VP NP-obj* rule has been applied and the connections to the *It* instance have increased from 0.01 to on average of 0.15. This weight increases to 0.17 by cycle 2000 as parsing supports it; the *Test* and *Erase* subnets both activate items on the stack and this causes neurons in the slot and filler to co-fire. Similarly, the weight to the *Stalactite* instance has increased, through co-firing due to spontaneous activity in the *VerbSlot* subnet; this is not particularly desirable. Parsing now ceases and the erase phase begins for the next 10000 cycles. Between cycles 2000 and 4000 the sink neurons fire, the weights to itself, *It* and to *Stalactite* all go down quite rapidly. Meanwhile, the weights to the sink increase. The total synaptic weight also rises in this period to near W_B. This means that any increases will be roughly the same as corresponding decreases. Between 4000 and the end of the erase phase, near cycle 12000, the sink weights remain stable while the

others go down. The total weight also goes down. This means that it will be relatively easy for the slot to bind to a new object when the next parse starts.

After this sink presentation and period of spontaneous activation, the binding was effectively erased. A second slot test left no neurons firing in any instance CA.

After sentence parsing, erasing and the three tests, a new sentence was presented. This was done to assure that the slots were able to rebind. Sentence 4 uses the same slots as sentence 3, but the fillers are different. Parsing of this sentence followed by the semantic test, slot test, erasing and the second slot test show that the slots are correctly rebound and then erased.

Move him toward the pyramid. Sentence 4

In figure 5, the next parse starts at this phase around cycle 12000. Sentence 4 is being parsed and the slots connections to itself again increase. The connections to the sink decrease as the weights are reclaimed. The weights to the *Him* instance also increase (not shown in the figure) while the weights to *It* and *Stalactite* continue to decrease.

An additional effect of the change from binding by STP to binding by compensatory LTP is the time to erase. In the original parser using binding by STP for slots, the erase phase lasted roughly 500 cycles. In this parser, it lasts 10,000 cycles. This is substantially longer, but also the binding persists for that extra period.

6 Discussion

The parsing simulations show that binding via synaptic change implements semantic and syntactic composition. Similarly, the simulations show that most of the problems of section 2.1 have be resolved using these algorithms. The use of binding via synaptic change does not contradict binding via synchrony, and it is shown how synchronous firing occurs in these simulations Below these issues, the ease of engineering with these binding mechanisms, and their possible use with other neural models is discussed. The simulations also show the times to bind, unbind, number of bindings and items per bindings (see section 3.3) of the binding mechanisms. A discussion of these values, the effects of dual dynamics, and stability vs. plasticity tradeoffs finishes the section.

It is easy to see how these mechanisms can be used to implement compositional semantics and syntax. The CABot1 parser and the parser of section

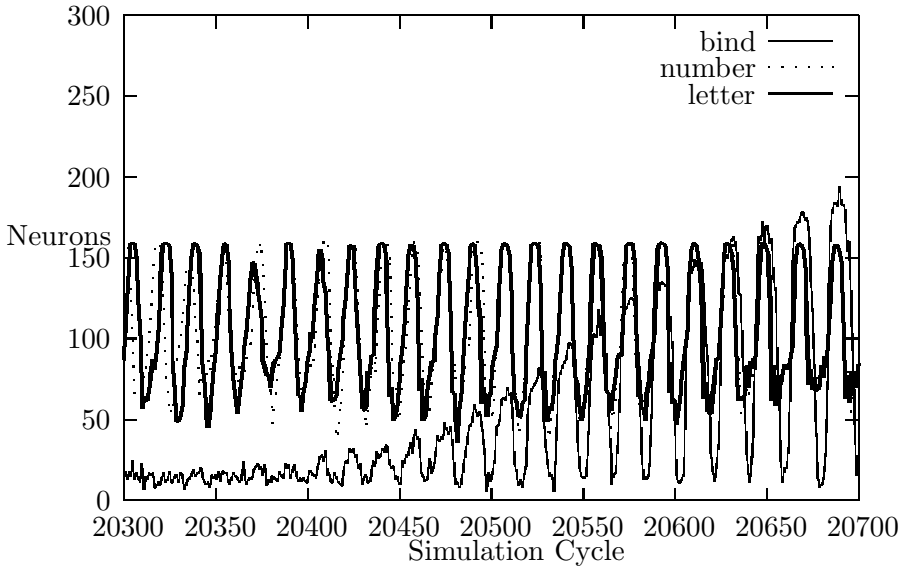

Figure 6: Firing of Bound Neurons

5 both exhibit compositional syntax and semantics. A sentence is represented by a verb frame that has slots that are dynamically filled. *Pat loves Jody.* includes the semantics of *Pat*, *love* and *Jody*, and is different from *Jody loves Pat.*

The simulations of sections 4.2 and 4.3 both execute a simple paired association task. This is a form of binding features in an object. Frames are obviously implemented via the parser (section 5). Rules are also implemented in the parser in the form of grammar rules, and they have been implemented in another simulation of counting [Huyck and Belavkin, 2006]. Complex matching or unification has not been shown in these simulations, but this is really just a form of complex rule implementation; consequently, it seems reasonable to expect binding via synaptic change to implement unification.

The alternative to binding by synaptic weight change is binding by synchrony. It must be noted that a single system can include both binding by synchrony, and binding by weight change. Indeed, the above simulations exhibit synchronous firing of bound items.

For example, figure 6 shows the firing behaviour of neurons in one run of the binding simulation described in section 4.2. This shows a section of one binding epoch. The x axis shows the number of neurons firing in

a subnetwork, and the y axis shows the cycle. Initially, the *number* and *letter* CAs are firing in different cycles. As the strength of the binding node grows, its neurons fire more frequently, and all three subnets begin to fire in synchrony; the firing is so closely correlated that the dotted *number* line disappears in the figure as it is covered by the *letter* line. The number of neurons firing in the base CAs oscillates, while the number firing in the binding CA oscillates while growing. This shows a strongly correlated firing pattern between the CAs.

It also must be noted that it has been significantly simpler to use binding by STP than to use binding by compensatory LTP. The portion of the parameter space where binding via compensatory LTP functions acceptably is quite small. This has required the use of relatively precise topologies, training, and use regimes. On the other hand, binding by STP works in a much larger range of conditions. The manipulation of learning and forgetting weights allows for a corresponding manipulation of bind and unbind times. This is a software engineering consideration, but the use of these mechanisms in large simulations of complex behaviour is encouraged and this is a software engineering task.

The binding by compensatory LTP and binding by STP models that are presented in this paper are examples of classes of learning algorithms. The use of both in the parser (section 5) is an example of a way these two classes might be combined. The use of binding by compensatory LTP enabled the system to limit the number of active bindings of a verb frame slot, and enabled that binding to persist longer. One might propose a compensatory form of STP that would also limit the number of active bindings. Ultimately, it is hoped that the neurobiological basis of neural learning will be sufficiently illuminated to say which algorithms are used for binding in the biological system. Until then, an exploration of different binding algorithms and their use in large systems to simulate complex behaviour may be a good way to explore the problems of neural binding.

The above simulations use fLIF neurons. Binding by compensatory LTP and STP should both be applicable to other neural systems. Spiking models are particularly appropriate. The rules may force breaking of the constraints of some attractor nets (e.g. Hopfield Net connections would no longer be bidirectional), but this is not incompatible from a simulation perspective. Continuous value output neural models should also be compatible with binding via STP. It is not entirely clear how spontaneous activation would work with these models, but compensatory learning should still work. It is also not clear how these mechanisms would apply to non-neural connectionist systems like Multi Layer Perceptrons.

Both binding by compensatory LTP and binding by STP as described in this paper have values associated with the properties of section 3.3. The values regarding time to bind are quite clear. The fLIF model equates 1 cycles with 10 ms. So, in section 4.2, it takes roughly 1000 cycles to bind, so roughly 10 seconds. The same mechanism with different parameters is used in the parser (section 5). Though the measurement is complex, the binding is done in between 10 and 100 cycles, which is between roughly 100 ms. and 1 second.

Compared to this, binding via STP is quite rapid. In both simulations in sections 4.2 and 5 the learning rate is set to .1 and the weight is clipped at 1; so binding happens in 10 cycles or less, and this equates to times less than 100 ms. This contradicts the statement "it is unlikely that there exist mechanisms that support widespread structural changes and growth of new links within" hundreds of milliseconds [Shastri and Aijanagadde, 1993]. There is biological evidence of STP based on short bursts of spikes that persist for seconds to minutes [Hempel et al., 2000]. So, it is biologically plausible that rapid binding can be implemented by synaptic change.

The persistence of binding also differs between binding by synchrony, binding by STP and binding by compensatory LTP. As noted, binding by synchrony will persist as long as the bound items are active. In the above simulations using binding by STP, synaptic weights are reduced by .004 each cycle they are not increased. So the weights are completely erased in 250 cycles, and may be effectively erased in less; this equates to 2.5 seconds.

The persistence of binding by compensatory LTP is more difficult to calculate. In the first binding simulation 6000 cycles (4 erase epochs and 2 unbound test epochs) were used to erase the binding, or 60 seconds. In the parser, there were 10000 cycles to erase the binding, or 100 seconds. However, this requires the activation of the sink neurons. Without this, the binding persists beyond 50,000 cycles. That is, without the activation of the sink neurons, the binding still persists after 50,000 cycles. This is due to the relative stability of compensatory LTP. When there is spontaneous activation of a small number of neurons, there are many more applications of anti-Hebbian learning than of Hebbian learning. So the total synaptic weight, W_i, is significantly below the goal weight W_B (see figure 5). This means that applications of the anti-Hebbian rule change the weights very little. This makes the original weights surprisingly stable. In the parser, the activation of the sink is needed to take the synaptic strength away from the neurons in the bound slot fillers.

The number of separate bindings differs between the three binding mechanisms. It is not clear how many bindings can be supported by synchrony,

but one simulation sets the limit at 10 [Henderson, 1994]. At the other extreme, binding by compensatory LTP supports a practically unlimited set of bindings. In the first simulation, there is only one binding, but in the parser, each slot is a binding node. There is no practical limit for the number of bindings except perhaps time to erase. The binding by STP mechanism that was used in the above simulations also supports a practically unlimited number of binding nodes, though again time is a factor. In the paired associate simulation, only one binding of two items was tested at a time, but each number could be bound to a unique letter, or indeed multiple letters. Similarly, in the parser, each stack item was bound to a different element.

The final property is the number of items per binding. The compensatory mechanism limited this, but the binding by STP mechanism did not. This was what motivated the change from the CABot1 parser to the new parser (section 5). Binding by synchrony also does not limit the number of things that can be in a binding as a host of CAs could fire in synchrony.

It should also be noted that the time courses of the binding by STP and binding by compensatory LTP are effected by the constants, topologies, and presentation mechanics. The above simulations should provide example time courses. For example, binding could take longer with STP if the increase weight was .05 instead of .1, and the binding would persist longer if the reduction was .001 instead of .004.

The three properties, speed to bind, number of bindings supported, and speed to unbind are also issues for general memory formation. Recall that CAs give an explanation for short-term memory (CA ignition and persistence) and long-term memory (stable state CA formation based on LTP). CA ignition happens quickly ($<$ 20 ms), but does not last long (seconds). CAs form more slowly, perhaps over days, but last much longer, perhaps years. CA ignition and CA formation are akin to speed to bind as all involve a memory formation. The cessation of a CA firing, and the loss of a stable state are akin to a binding being erased as all involve the loss of memory.

While there is some debate as to whether memories are lost or not, it is largely accepted that as time passes, memories become less accessible [Klatzky, 1980]. Figure 7 shows the amount of memory that can be accessed as time progresses. This figure is meant to be a qualitative guide of the process indicating that as time passes fewer memories from a particular time can be accessed. At the left of the table CA ignition (CAI) does not last long, but in a given period (say an hour) many memories can be used. On the right, CA formation (CAF) shows that memories last a long time, but not many things (relative to the number of CAs accessed) can be stored.

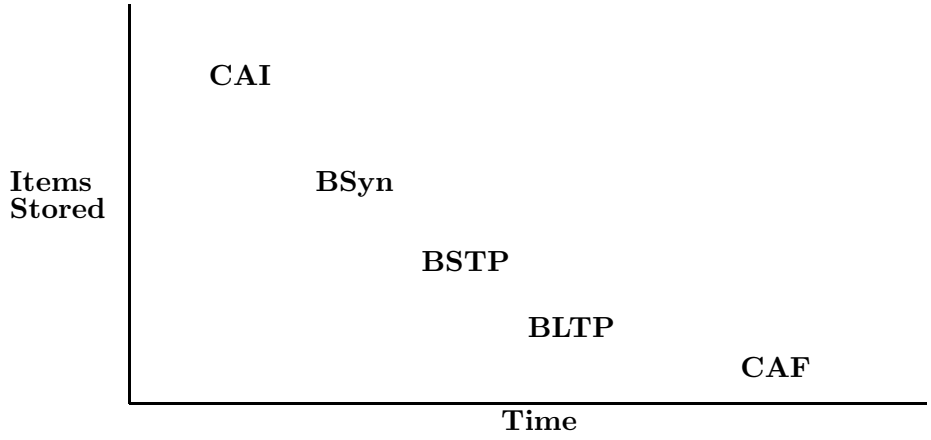

Figure 7: Memory Hierarchy

Without binding, this leaves the middle ground empty; how can something be forgotten after only a day? Binding fills in this middle area. Many items may be bound by synchrony (BSyn); clearly there are fewer than the CAs that are active, and they can persist longer as only one of the base CAs is needed to keep the binding. Binding by STP (BSTP) probably occurs less frequently, but persists longer than binding by synchrony. Finally, binding by compensatory LTP (or any LTP) has fewer items bound, but persists longer yet. So, over a given hour, 1000 CAs might ignite, 100 sets of CAs might be bound via synchrony, 20 bound by STP, 10 bound by LTP and two new CAs might be created. The ignited CAs would persist for one minute, the synchronous bindings for two, the bindings by STP for five minutes, the bindings by LTP for two hours, one new CA might last for a month and the other for 10 years.

These memory mechanisms use and are influenced by the dual dynamics of CA ignition and CA formation. One good example of the complexity of these dual dynamics is the erasing of the binding by compensatory LTP in section 4.2. The weights from the bound letter to the *bind* subnet are not changed during erasing. When the letter is presented after erasing, the synaptic weights to the *bind* subnet are high, but they go down rapidly. This rapid decline completes the erasing. The dynamics also have an effect on the stability of existing CAs and formation of new CAs.

Biological neural systems are always learning [Churchland and Sejnowski, 1992], and there is always spontaneous firing. Under these conditions, CAs must ignite relatively frequently to keep their mutual synaptic strength high. It does not seem reasonable that all CAs are ignited relatively frequently. The relative stability of compensatory LTP bindings, as shown earlier in this section, provides some hope that this problem may be resolved, but it has as yet not been resolved.

Binding by synchrony and STP have a lesser effect on CA stability and plasticity, but they still have an effect. They have less of an effect because they are not based on long term synaptic change. They still have an effect because they cause the simultaneous firing of neurons in CAs, and this will lead to increased permanent synaptic weight between the bound CAs. This might lead to the CAs recruiting each other, so that they no longer could be independently active.

Hetero-associative memory may play a part in this. CAs that are frequently bound may become more related but, perhaps due to topology, may not recruit each other. Other options for resolving stability problems include modified spontaneous activation mechanisms, subassemblies, and learning rules involving fatigue. In the above model, spontaneous activation is purely random; this might be modified to make neurons fire when they have not fired for a long time, and these neurons might co-fire based on their last activity. Subassemblies are merely sets of neurons that do not persist, but can be activated by spontaneous activation leading to synaptic support. Finally, if synaptic weights only changed significantly when neurons were fatigued, spontaneous activation would have little effect on them. These mechanisms are, of course, speculative.

7 Conclusion

This paper has introduced a new variable binding mechanism, binding by STP. It has also made use of the relatively novel variable binding by compensatory LTP.

Binding by STP is fast to bind, persists beyond the activity of the bound CAs, is relatively easy to engineer, and works consistently. Binding by compensatory LTP works, but faces the stability plasticity dilemma. It is slower to bind and the bindings persist longer.

Neither of these mechanisms faces a combinatorial explosion to bind items, and both can support a very large number of bindings. One binding can support a small number of items with compensatory LTP while a large

number of items could be supported by STP.

Binding via compensatory LTP and by STP can be used together and with binding via synchrony. These mechanisms complement each other. They each have different behaviours on time to bind, time to erase, and capacity. Along with CA ignition and CA formation, these binding mechanisms give a wide range of memory formation and retention behaviour.

Together, these mechanisms allow for a sophisticated use of compositional syntax and semantics in a simulated neural system. This allows complex symbol processing agents to be developed from simulated neurons bridging the gap between subsymbolic and symbolic systems.

Acknowledgements:
This work was supported by EPSRC grant GR/R13975/01 and EP/D059720.

References

[Abeles et al., 1993] Abeles, M., Bergman, H., Margalit, E., and Vaddia, E. (1993). Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of Neurophysiology*, 70(4):1629–1638.

[Ackley et al., 1985] Ackley, D., Hinton, G., and Sejnowski, T. (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169.

[Amit, 1989] Amit, D. (1989). *Modelling Brain Function: The world of attractor neural networks*. Cambridge University Press.

[Amit and Brunel, 1997] Amit, D. and Brunel, N. (1997). Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral Cortex*, 7:237–252.

[Anderson and Lebiere, 1998] Anderson, J. and Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence Erlbaum.

[Bevan and Wilson, 1999] Bevan, M. and Wilson, C. (1999). Mechanisms underlying spontaneous oscillation and rhythmic firing in rat subthalamic neurons. *Neuroscience*, 19:7617–7628.

[Bratenberg, 1989] Bratenberg, V. (1989). Some arguments for a theory of cell assemblies in the cerebral cortex. In Nadel, Cooper, Culicover, and Harnish, editors, *Neural Connections, Mental Computation*. MIT Press.

[Browne and Sun, 1999] Browne, A. and Sun, R. (1999). Connectionist variable binding. *Expert Systems*, 16:3:189–207.

[Brunel, 1996] Brunel, N. (1996). Hebbian learning of context in recurrent neural networks. *Neural Computation*, 8:1677–1710.

[Buonomano, 1999] Buonomano, D. (1999). Distinct functional types of associative long-term potentiation in neocortical and hippocampal pyramidal neurons. *Neuroscience*, 19:16:6748–6754.

[Carpenter and Grossberg, 1988] Carpenter, G. and Grossberg, S. (1988). The art of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, 21:77–88.

[Churchland and Sejnowski, 1992] Churchland, P. and Sejnowski, T. (1992). *The Computational Brain*. MIT Press.

[Eccles, 1986] Eccles, J. (1986). Chemical transmission and dale's principle. *Prog. Brain Research*, 68:3–13.

[Eckhorn et al., 1988] Eckhorn, R., Bauer, R., Jordan, W., Brosch, M., Kruse, W., Munk, M., and Reitboeck, H. (1988). Coherent oscillations: A mechanism of feature linking in the visual cortex? *Biological Cybernetics*, 60:121–130.

[Feldman, 1982] Feldman, J. (1982). Dynamic connections in neural networkse. *Biological Cybernetics*, 46:27–39.

[Filmore, 1968] Filmore, C. (1968). The case for case. In Back, E. and Harms, R., editors, *Universals in Linguistic Theory*. Holt, Rinheart and Winston Inc.

[Fodor and Pylyshyn, 1988] Fodor, J. and Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A criticial analysis. *Cognition*, 28:3–71.

[Frixione et al., 1989] Frixione, M., Spinelli, G., and Gaglio, S. (1989). Symbols and subsymbols for representing knowledge: a catalogue raisonne. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 3–7.

[Furber et al., 2004] Furber, S., Bainbridge, W., Cumpstey, J., and Temple, S. (2004). Sparse distributed memory using n-of-m codes. *Neural Networks*, 17:10:1437–1451.

[Fyfe, 2005] Fyfe, C. (2005). *Hebbian Learning and Negative Feedback Networks*. Springer.

[Gerstner and Kistler, 2002] Gerstner, W. and Kistler, W. (2002). Mathematical formulations of hebbian learning. *Biological Cybernetics*, 87:404–415.

[Gerstner and vanHemmen, 1992] Gerstner, W. and vanHemmen, J. (1992). Associative memory in a network of spiking neurons. *Network*, 3:139–164.

[Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D*, 42:335–346.

[Hebb, 1949] Hebb, D. O. (1949). *The Organization of Behavior*. J. Wiley & Sons.

[Hempel et al., 2000] Hempel, C., Hartman, K., Wang, X., Turrigiano, G., and Nelson, S. (2000). Multiple forms of short-term plasticity at excitatory in rat medial prefrontal cortex. *Journal of Neurophysiology*, 83:3031–3041.

[Henderson, 1994] Henderson, J. (1994). Connectionist syntactic parsing using temporal variable binding. *Journal of Psycholinguistic Research*, 23:5:353–379.

[Hodgkin and Huxley, 1952] Hodgkin, A. and Huxley, A. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544.

[Hofstadter, 1979] Hofstadter, D. (1979). *Godel, Escher and Bach: an Eternal Golden Braid*. Basic Books.

[Hopcroft and Ullman, 1979] Hopcroft, J. and Ullman, J. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[Hopfield, 1982] Hopfield, J. (1982). Neural nets and physical systems with emergent collective computational abilities. *Proc. of the Nat. Academy of Sciences USA*, 79:2554–8.

[Huyck, 2004] Huyck, C. (2004). Overlapping cell assemblies from correlators. *Neurocomputing*, 56:435–439.

[Huyck and Belavkin, 2006] Huyck, C. and Belavkin, R. (2006). Counting with neurons: Rule application with nets of fatiguing leaky integrate and fire neurons. In *Proceedings of the Seventh International Conference on Cognitive Modelling*, pages 142–147.

[Huyck and Bowles, 2004] Huyck, C. and Bowles, R. (2004). Spontaneous neural firing in biological and artificial neural systems. *Journal of Cognitive Systems*, 6:1:31–40.

[Huyck and Fan, 2007] Huyck, C. and Fan, Y. (2007). Parsing with flif neurons. In *IEEE Systems, Man and Cybernetics Society*, pages 35–40.

[Kaplan et al., 1991] Kaplan, S., Sontag, M., and Chown, E. (1991). Tracing recurrent activity in cognitive elements(trace): A model of temporal dynamics in a cell assembly. *Connection Science*, 3:179–206.

[Kaplan et al., 1990] Kaplan, S., Weaver, M., and French, R. (1990). Active symbols and internal models: Towards a cognitive connectionism. *AI and Society*, 4:51–71.

[Klatzky, 1980] Klatzky, R. (1980). *Human Memory: Structures and Processes*. W. H. Freeman & Co.

[Knoblauch et al., 2004] Knoblauch, A., Markert, H., and Palm, G. (2004). An associative model of cortical language and action processing. In *Proceedings of the Ninth Neural Computation and Psychology Workshop*.

[Knoblauch and Palm, 2001] Knoblauch, A. and Palm, G. (2001). Pattern separation and synchronization in spiking associative memories and visual areas. *Neural Networks*, 14:763–780.

[Laird et al., 1987] Laird, J., Newell, A., and Rosenbloom, P. (1987). Soar: An architecture for general cognition. *Artificial Intelligence*, 33:1.

[Lindsey, 1988] Lindsey, R. (1988). Can this treatment raise the dead. *Behavioral and Brain Sciences*, 11:1:41–42.

[Maas and Bishop, 2001] Maas, W. and Bishop, C. (2001). *Pulsed Neural Networks*. MIT Press.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[Newell, 1990] Newell, A. (1990). *Unified Theories of Cogntion*. Harvard University Press.

[Oja, 1982] Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273.

[Pulvermuller, 1999] Pulvermuller, F. (1999). Words in the brain's language. *Behavioral and Brain Sciences*, 22:253–336.

[Quillian, 1967] Quillian, M. (1967). Word concepts: A theory of simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–30.

[Rumelhart and McClelland, 1982] Rumelhart, D. and McClelland, J. (1982). An interactive activation model of context effects in letter perception: Part 2. the contextual enhancement and some tests and extensions of the model. *Psychological Review*, 89:1:60–94.

[Sakai and Miyashita, 1991] Sakai, K. and Miyashita, Y. (1991). Neural organization for the long-term memory of paired associates. *Nature*, 354:152–155.

[Schank and Abelson, 1977] Schank, R. and Abelson, R. (1977). *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum.

[Shastri and Aijanagadde, 1993] Shastri, L. and Aijanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behaviour and Brain Science*, 16:417–494.

[Shieber, 1986] Shieber, S. (1986). *An Introduction to Unification-Based Approaches to Grammar*. Center for the Study of Language an Information.

[Siegelmann and Sontag, 1991] Siegelmann, H. and Sontag, E. (1991). On the computational power of neural nets. Technical report, SYCON.

[Smolensky, 1988] Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1:1–22.

[Smolensky, 1990] Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.

[Sompolinsky, 1987] Sompolinsky, H. (1987). The theory of neural networks: The hebb rule and beyond. In *Proceedings of Heidelberg Colloquium on Glassy Dynamics*.

[Usher and Donnelly, 1998] Usher, M. and Donnelly, N. (1998). Visual synchrony affects binding and segmentation in perception. *Nature*, 394:179–182.

[VonDerMalsburg, 1986] VonDerMalsburg, C. (1986). Am I thinking assemblies? In Palm, G. and Aersten, A., editors, *Brain Theory*, pages 161–176. Springer-Verlag.

[Wennekers and Palm, 2000] Wennekers, T. and Palm, G. (2000). Cell assemblies, associative memory and temporal structure in brain signals. In Miller, R., editor, *Time and the Brain: Conceptual Advances in Brain Research (Vol. 2)*, pages 251–274. Harwood Academic Publishers.

[White et al., 1988] White, G., Levy, W., and Steward, O. (1988). Evidence that associative interactions between afferents during the induction of long-term potentiation occur within local dendritic domains. *Proceedinigs of the National Academy of Sciences*, 85:2368–2372.

[Willshaw et al., 1969] Willshaw, D., Buneman, O., and Longuet-Higgins, H. (1969). Non-holographic associative memory. *Nature*, 222:960–962.