Technical Report OUCS-2005-09

Implementation of a model of the relation between language and sensorimotor cognition

Authors:

Maarten van der Veen, Jeroen van Dijk
Department of Computer Science, University of Otago

Status: Report for a project by two visiting students from the University of Groningen

Implementation of a model of the relation between language and sensorimotor cognition

M. van der Veen J. van Dijk

June 17, 2005

Contents

Chapter 1

Introduction

This report is about the relation between natural language and sensorimotor cognition. While traditionally there has not been much collaborative effort between linguists and cognitive scientists to search for a relationship between language and sensorimotor cognition, this issue has recently been tackled by a number of researchers (Feldman, Lakoff, Bailey, Narayanan, Regier & Stolcke, 1996; Siskind, 1995; Rizzolatti & Arbib, 1998; Givón, 2002; Corballis, 2002; Hurford, 2003; Dominey, Hoen, Blanc & Lelekov-Boissard, 2003; Knott, 2005). Some of these researches have worked on solving the L0 task 'a touchstone for cognitive science', which was introduced by (Feldman et al., 1996). The L0 task is to build a system that simulates the task of a child learning a language: it has to take pairs of (a) video of an action or state and (b) sentence describing the action/state, and learn to generate new sentences given new video inputs.

Research on this topic is uncovering interesting evidence that suggests there is overlap between theories of language and theories of sensorimotor cognition and that we may find some possible answers to questions like: the origin of universal grammar, by considering it's relation to sensorimotor capabilities.

The general hypothesis in many of the above works is that the human capacity to use natural language has emerged from prelinguistic sensorimotor capacities. The specific hypothesis which we will consider in this project is one proposed by (Knott, 2005) that when a human agent observes a concrete physical event (for instance, a man picking up a cup), that the syntactic structure of the sentence he uses to describe this event can be understood as a trace of the sensorimotor processes via which the information is acquired. These sensorimotor events happen in several fairly independent neural pathways; for instance, a pathway which works out what are salient points in the environment to attend to, a pathway which categorises objects which are attended to, and a pathway which categorises actions which are observed. In Knott's model of action recognition and action execution, the main suggestion is that events in these separate pathways happen in a characteristic sequence, and that the cognitive representation of an event is thus a sequence of sensorimotor activations in different areas of the brain. The hypothesis is that these sequences are encoded in episodic memory representations, and these memory representations map onto the syntactic structure of sentences.

The goal of this research is to develop two neural networks: one which implements a model of episodic memory for sensorimotor sequences, and one which maps sensorimotor sequences onto the surface structure of sentences. The first network will take as input the sequence of sensorimotor activations which occurs when an agent observes one simple physical event, such as "the man grabs a cup", and will learn to reproduce this sequence when given a contextually appropriate cue. The second network takes pairs of sensorimotor sequences and sentences, and learns to generate the appropriate sentence for new sensorimotor sequences.

For each network, there were four things to do: first to read and understand Knott's model and the relevant background literature, second to design a concrete neural network to implement the model, third to produce the implementation itself, and finally to evaluate the implementation and the plausibility of the model. The design phase often involved suggesting changes to or additional specifications for Knott's original model.

The understanding, revision, implementation and evaluating of the sensorimotor model has been done by M. van der Veen. The understanding of Knott's syntactic framework, and the construction and evaluation of a sentence generation model has been done by J. van Dijk.

In this report we start by giving an overview of the model proposed by Knott in Chapter 2. This includes an introduction to both the sensorimotor model and the syntactic model and includes the changes made to the original models. In Chapter 3 the implementation of the different parts of both the sensorimotor model and the syntactic model is discussed. Experiments and results for both models will also be discussed in this chapter. Finally we draw conclusions about the models and evaluate to what extent it offers an explanation for the relationship between natural language and sensorimotor cognition in Chapter 4. We present some ideas for future work in Chapter 4.

Chapter 2

A model of the relationship between language and sensorimotor cognition

This chapter introduces a model of the relationship between language and sensorimotor cognition proposed by Knott (2005). The model that is developed has two parts: a sensorimotor model of action processing and episodic memory, which is discussed in Section 2.2, and a model of sentence syntax, discussed in Section 2.5. Both these models use the basic Elman network architecture. An overview of the general idea behind this type of network is given in Section 2.1 to give the reader a better understanding of the model under discussion. The model of sentence syntax is based on Chomskyan Minimalism, which is discussed in section 2.3. An interpretation of how the sensorimotor model and sentence syntax model relate is given in 2.4.

2.1 Preliminaries: Elman networks for sequence learning

Both in word sequencing and sensorimotor sequencing tasks, it is necessary to learn a sequence of items. To learn a sequence, one common neural network architecture is an **Elman recurrent network** (Elman, 1990). Elman networks are normal multiple feedforward neural networks extended with a context layer. Elman networks have the specific ability to encode sequences because the context in which an input item is presented to the network is remembered by copying the hidden layer to the context layer; see Figure 2.1.

For an Elman network to learn a sequence, it uses the current item as an input and the next item as a target for the output of the network. After the network has learned a sequence, or multiple sequences, it is able to predict the next item in the sequence when given the current item and context. This is why such a network can be refered to as a next-state function.

Elman networks can not only be used for remembering sequences, but also for recognizing sequences. (Elman, 1990) points out that it is possible to use an Elman network to remember multiple sequences and predict with increasing probability the items in each sequence. Because the first item in each sequence is never preceded by the same item structurally, the probability of it being predicted correctly corresponds to one divided by the number of different items in the first position of all training sequences. However, the second item in each

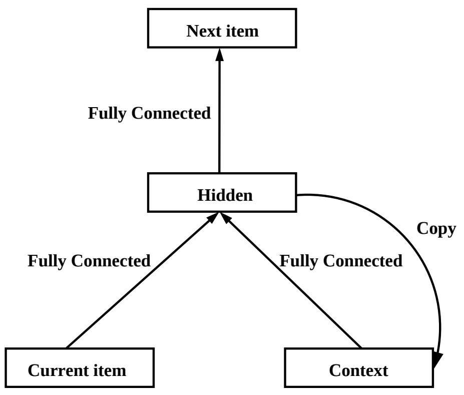


Figure 2.1: Simple Elman Network

sequence is more often preceded by these first items than by any other item. The third item in the sequence is even more often preceded by these first two items than by other items. As the sequences get longer, the probability that the same items precede an item further down the sequence decreases. Thus, the network can predict the items further down the sequence with more accuracy than the first items in the sequence. The more unique a sequence is, the more accurate an Elman network can predict its items. The following steps show how an Elman network learns a sequence:

1. Initialize context layer with random weights
2. Present item A to current item layer
3. Calculate network output at the next item layer
4. Compare predicted next item with target item B and backpropagate the error
5. Copy new context from hidden layer to context layer
6. Start again at 2, but present B to the current item layer, with C as target value
7. Stop training when the error over all items is sufficiently small

In Figure 2.2 dotted lines show which weights are being updated when backpropagating the error between the predicted next item and the target item. For a more detailed description on Elman Networks, see Elman (1990). However, there are a few important points worth noting here.

Weight initialisation When presenting the first item in a sequence there is no sensible context to use, since the context represents the previous items in the sequence. Thus, the context layer is initialized with random values in the range [-1,1]. This random initialization is important to prevent a bias in learning.

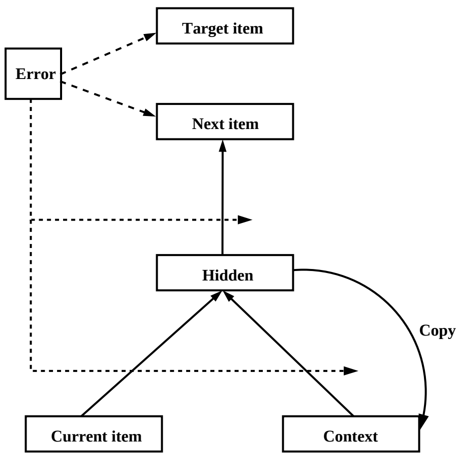


Figure 2.2: Backpropagation in a simple Elman Network

Update function It is important, for the outcome of a cycle, to visit the neurons in a network in a specific order. In an Elman Network a pattern is propagated from the input to the hidden layer and then to the output layer. After this the context layer is updated by copying the hidden layer to the context layer.

Learning algorithm The weights from the hidden layer to the context layer are fixed. The weights are set to 1.0 and are not changed by the learning algorithm. This is a neural network equivalent of a 'copy' operation. Since the weights from the hidden layer to the context layer are fixed, the network can be seen as a normal feedforward network. That is why it is safe to train the network with a normal error backpropagation algorithm.

2.2 Knott's sensorimotor model of action processing, episodic memory and reinforcement learning

In this section we review Knott's argument that a transitive action is cognitively encoded as a sequence of attentional operations, in which agent, patient and action are defined as operations occurring in characteristic serial positions.

In Sections 2.2.1 and 2.2.2, evidence is presented for a decomposition of events into sub-processes and saccadic eye movements as an explanation of how different processes in the brain use the same information. In Section 2.2.3, event perception is elaborated by stepping through an example of the perception of an event. Finally, in Sections 2.2.4 to 2.2.7, we look at the different parts of the suggested model and how they can account for the steps in event perception.

2.2.1 Decomposition of event perception into sub-processes

Knott (2005) develops a sensorimotor model in which the visual perception of an event in the world invokes several relatively separate sub-processes. This idea is motivated from the well accepted hypothesis that the brain's visual processing is organized in separate pathways (see e.g. Milner & Goodale, 1995). Each of these pathways extracts different information from the visual input. Among others, one pathway categorises objects according to their static form (Riesenhuber & Poggio, 1999), another delivers representations about the motor affordances of objects (Fagg & Arbib, 1998; Rizzolatti, Fogassi & Gallese, 2000), another delivers objects and actions derived from motion information (Johansson, 1973) and another delivers information about the most salient objects in the scene (Itti & Koch, n.d.).

2.2.2 Deictic representations

There is also evidence for a temporal decomposition of perceptual processes. Humans perceive the world via a sequence of separate snapshots, each resulting from a separate eye fixation (Ballard, 1991). The eye moves from one position to another in a fast ballistic motion called a saccade. There are roughly three saccades every second for the human eye. Each snapshot delivers **deictic representation** -i.e. a transitory representation of what the eye is currently looking at. Each visual pathway can compute its own deictic representation to encode e.g. the template, motor affordances of, or action associated with, the currently attended to object.

The notion of deictic representations suggest that cognitive processing proceeds by iteratively executing an action of attention and interpreting its results. Two examples of this process are given below.

Object categorization

(2.1) (i) The agent performs a saccade to an object
(ii) The agent categorizes the object it is looking at

Grasp action

(2.2) (i) The agent performs a saccade to an object
(ii) The agent grasps the object it is looking at

In each case, the second action invokes a deictic representation which gets its meaning from the immediately preceding direction of attention. We cannot understand what is going on until we look at a sequence of operations; first a direction of attention, and then some other operation. If the next direction of attention can itself be a function of the current sensory state, we end up with a picture where complex cognitive operations have a strong sequential structure. Such a sequence of deictic representations is a **deictic routine** (Ballard, Hayhoe, Pook & Rao, 1997).

2.2.3 Event perception

Knott's suggestion is that event perception can be decomposed into a strict sequence of deictic operations, which constitute a deictic routine (Knott, 2005). To give an example of such a deictic routine, consider the perception of the transitive event: 'I grab a cup'.

At **stage 1**, the observer is in an attentional state where objects in the world, including the observer himself, compete for the observer's attention.

At **stage 2**, the observer selects an object to attend to, by considering both the bottom-up availability of objects in the world and the top-down preferences derived from its current desires. In our example, the observer selects himself. This initial attented to object will end up being the agent of the action.

At **stage 3**, the observer creates a new attentional environment, now centred on the attended-to-object (in this case his own body). This biases attention to objects which are close to the agent, and within reach (Tipper, Lortie & Bylis, 1992). In this new environment, these objects compete for the observer's attention.

At **stage 4**, the observer selects one of these objects to attend to, in our example a cup. This object will end up being the patient of the action.

At **stage 5**, the observer is in an attentional state where several possible alternative actions on the cup are represented, and compete for selection. These actions are represented as motor goals - goals to get the hand / arm into some positions in relation to the cup.

At **stage 6**, one of these motor actions is selected. In our example, "grab". This triggers a physical motion. As a side effect of the changes in the agent's motor state during the execution of the action, the observer once more attends to the agent (in this case the observer himself) via the biological motion pathway: when we observe an agent perform an action, we cannot help establishing the agent as an object at the same time (Giese, 2000).

At **stage 7**, tracking of the agent's motor action is finished when the agent successfully reaches its goal motor state. In this case, when the agent is holding the cup. While executing this motor action, the agent re-attends to the target object, in this case the cup (Jeannerod, 1996).

In stages 2 and 4 the observer selects an attentional action. In stage 6 the observer selects a goal motor state. This selection process is performed by a combination of two separate sub-processes: a **decision function**, which provides a top-down bias, and the **sensorimotor system**, which provides a bottom-up interaction with the world. Both these processes are explained in the next paragraph.

2.2.4 Architecture for sensorimotor control

We can model the process which transits from one stage to the next in the above sequence as a function, whose input is a deictic representation (which we will call the "current sensorimotor state") and whose output is firstly an attentional operation and secondly a new sensorimotor state.

The **decision function** is a sub-process that uses the current sensorimotor state (which comprizes the currently active object template and the currently active motor state) to decide on the next action. This action can be either an attentional action or a motor action, or both. We will use the term **deictic operation** to refer to both kinds of actions. In the case of an attentional action this deictic operation is a top-down bias on the visual system to look for objects with certain low-level visual features (Treisman & Gelade, 1980). In the case of a motor action this deictic operation is a motor goal state to activate. The decision function chooses a deictic operation which is most likely to get the observer into a state which has benefits for the observer (which we will call a **reward state**). To be able to perform this task, the decision function has to be trained. How this training operates is explained in Section 2.2.7.

The **sensorimotor function** models certain aspects of the sensorimotor system as embedded in a real world context. It receives a deictic operation from the decision function[1]. Since the agent operates in an environment which does not necessarily conform perfectly to

[1]In Knott's original model, the decision function was directly connected to the hidden layer of the episodic memory, this has been changed since the episodic memory would then learn not to consider the context and only use the decision function output to produce the next remembered state.

the observer's desires, an object or motor goal state with these preferred features cannot always be selected. This selection process is one of the functions of the sensorimotor function.

The sensorimotor function chooses an object template and motor goal state to activate[2]. This selection is dependent on both the bottom-up preferred features from the deictic operation and the features of the actual objects in the world. The mapping from deictic operation onto object templates is done by comparing these features. The template that shares the most features with the bottom-up preferred features is chosen. It is not necessary that an object exists in the world which shares all the features with the deictic operation. If no such object exists, the object which shares most of the features is chosen. The sensorimotor function will then determine what object to attend to, or what action to execute.

There is also the matter of **exploration versus exploitation**. When the decision function is untrained, the observer should explore the world and find the reward states in that world. If the observer starts exploiting previously learned sequences too soon, he will not be able to find most or all reward states possible. The sensorimotor function handles this problem in two separate ways. First, the observer is less sensitive to the decision function's advice in the early stages of learning. Second, there is the notion of **inhibition of return**. When the observer has attended to a certain object and executed an action on this object, the object is inhibited so as to make sure it is not attended to again in the near future. Inhibition of return also makes sure that the observer chooses different actions when the decision function has learned all sequences.

The object template and motor state which are output of the episodic memory have to be mapped onto deictic operations, in order to train the decision function. The feature decomposition function maps object templates and motor states back onto deictic operations. See Section 2.2.7 to learn how this functionality is used in practice

2.2.5 Episodic memory

In Knott's model, episodic memories of sensorimotor sequences play a crucial role. Why is episodic memory considered to be a part of the proposed model? The goal is to link the syntactic structure of a sentence to the sensorimotor system of humans. But, sentence creation and event perception are not parallel processes; a sentence is not necessarily pronounced while perceiving the action. Rather, the sentence describing an event can be formulated long after the event was perceived. The hypothesis is thus that the syntactic structure of a sentence can be seen as a trace of an episodic memory operation which rehearses or simulates the original sensorimotor experience (see also Bergen & Chang, 2003).

In Knott's model, when an event is perceived, the sequence of activated object templates and motor goal states is stored in episodic memory. When this event is recalled, the sequence in which the objects and actions were originally perceived is reactivated. It is this reactivation of the sequence, rather than the original sequence, which is linked to sentence creation. For a more detailed description of the link between the episodic memory and sentence structure, see Section 2.4.

In Knott's model, the episodic memory is implemented as an Elman style recurrent network. The basic architecture of Elman Networks was discussed in Chapter 2.1 since it is basis of both the sensorimotor model and the word sequencing model discussed later.

[2]Knott's original model did not make a distinction between deictic operations and object templates / motor goal states.

2.2.6 The model of episodic memory: experience mode and recall mode

The model has three modes. In **experience mode** mode the observer stores new events in episodic memory when object templates and motor states are activated by the sensorimotor function. In **recall** mode, the observer recalls an event by replaying the sequence of object templates and motor states stored in episodic memory. In **rewind mode**, the decision function is trained to produce sequences which are more likely to lead to reward states. The first two will be discussed in this section, the latter is discussed in Section 2.2.7.

The model consists of three sub-processes, all discussed earlier: the decision function and sensorimotor function to determine what object the agent will attend to or what action he will execute and the episodic memory to remember in what sequence these events were perceived. The episodic memory model is an Elman Network. See Figure 2.3 for how these sub-processes are connected in both experience mode and recall mode.

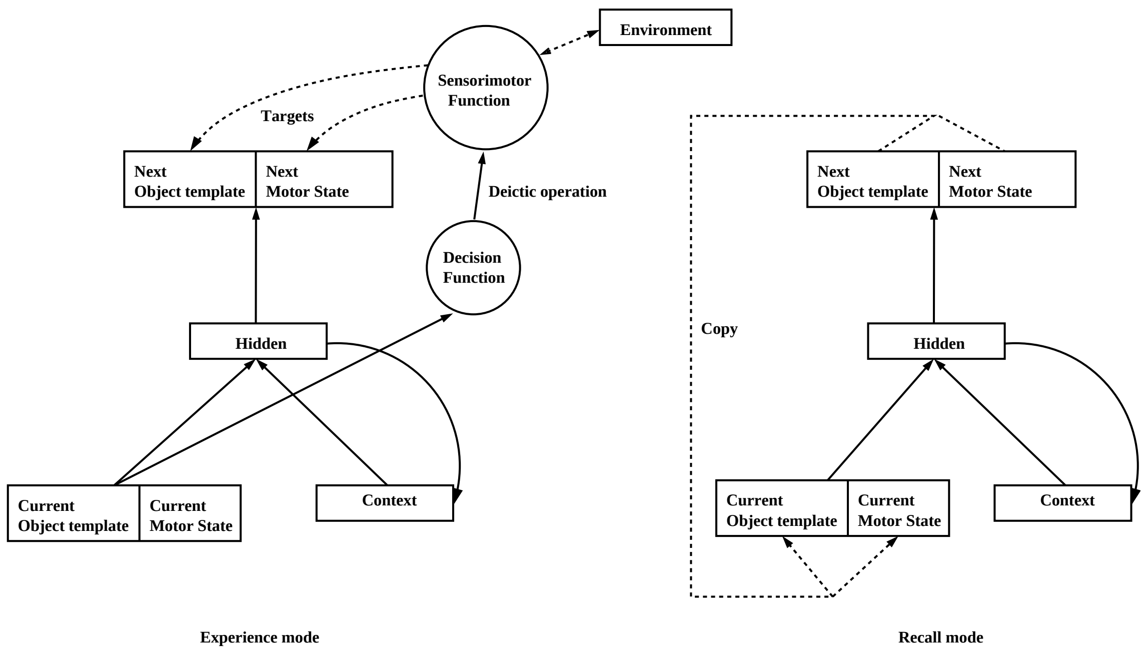


Figure 2.3: Experience mode and Recall mode

In experience mode, assume that the observer has no previous memories. An object in the world catches the attention of the observer and the object template is activated by the sensorimotor system[3]. The observer's attentional context now changes. This perceptual event has to be remembered, so the episodic memory is trained to predict this object from the initial context and object, which are both random since the observer had no previous memories. The decision function now chooses a preferred next deictic operation on the basis of this new context and activated object template. The sensorimotor function interacts with the world to see if an object with the features of this deictic operation exists. If so, this results in a new object template being activated. If a motor goal state is present in the

[3]The sensorimotor function has been added to Knott's original model to account for the mapping of deictic operations onto templates and motor states.

deictic operation, the motor action is executed. The episodic memory now stores the new object template and motor state and the whole process is repeated.

In recall mode, an event has to be remembered. Assume that a series of events is stored in episodic memory. The observer activates the sequence of events by triggering the network with an initial context, object template and motor state. The episodic memory will then give the next template that was activated. This predicted template is made the current template at the next iteration of the network. The hidden layer is copied to the context layer at the next iteration, and the network then predicts the next item in the sequence. This way, the whole sequence of events can be recalled.

2.2.7 Learning the decision function: rewind mode

The agent's intrinsic goal is to execute motor actions and attentional actions in such an order that it leads to a reward state. An example of a successful sequence of these actions is shown in Table 2.1. In this case, the reward state is the observer holding the cup.

{attend to oneself, attend to cup, cup-grasp}

Table 2.1: Sequences of deictic operations that lead to a reward

Attentional actions by themselves do not lead to reward states. Neither do motor actions without the proper prior attentional actions lead to a reward state, because the agent will not successfully perform the action on an object. However, the right sequence of attentional actions and motor goal states can lead to a reward state. In the example shown in Table 2.1, the attentional action to the cup generates the proper motor goal state for a cup-grasp action, which is centered on the observer because of the first attentional action on the observer himself.

When the decision function is untrained it will generate random deictic operations, that do not necessarily move the agent closer to a reward state. But when a specific sequence of actions does happen to result in a reward state, the agent wants to train the decision function in order to generate similar sequences in the future. The decision function thus not only has to learn to generate the deictic operation that resulted in the reward state, but also the previous deictic operations that resulted in the activations of the previous object templates and motor states. That is why the model moves into **rewind mode** to train the decision function on the whole sequence. To be able to remember sequences and predict next deictic operations that are more likely to result in a reward state, we use an Elman Network. See Figure 2.4 for the connection of sub-processes in rewind mode.

Once a deictic operation from the decision function leads to a reward state, the episodic memory is used to retrieve the previous object templates and motor states which the sensorimotor function has activated. By rewinding through the episodic memory the network sets itself to the beginning of the sequence.

The decision function has to learn each next item in the sequence from the current item. The first item is presented to the decision function. At the same time, the episodic memory generates the next item, which is then used as a target for the decision function. Next, the next item is copied to the current item, and the new context to the context layer. This way the episodic memory forwards through the items of the sequence and passes each current item and predicted next item to the decision function, which can now learn the whole sequence[4].

[4]Knott's original model used a delay function to retrieve the next state, but since the episodic memory has

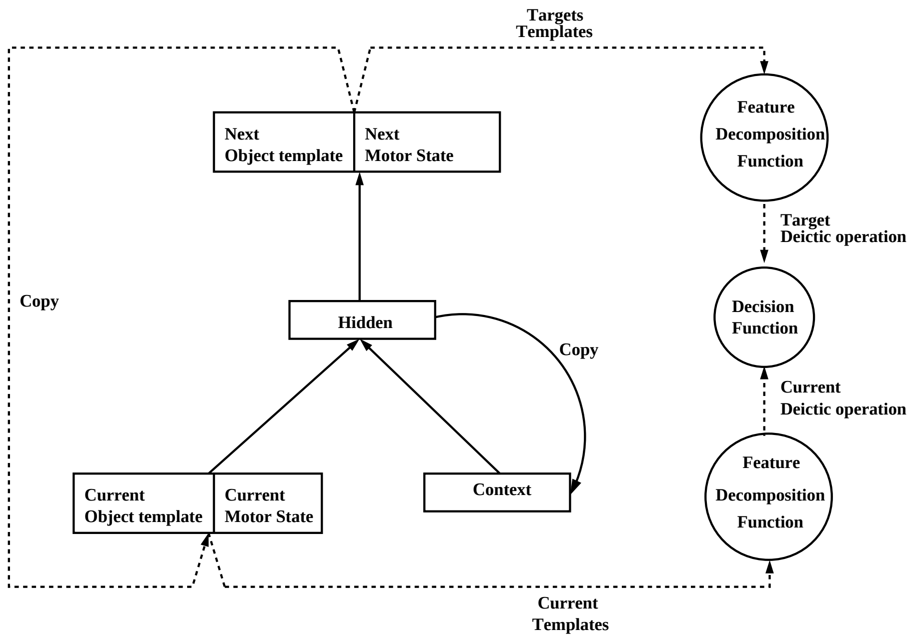


Figure 2.4: Rewind mode network

Figure 2.5 shows how the decision function learns a sequence. The decision function learns to generate sequences of deictic operations, not of object templates. The object templates are thus decomposed into features by a new function called the **feature decomposition function**. The best target deictic operation that can be used to learn the decision function is a deictic operation which shares all the features with the rewarded object template and motor state. This way it will be more likely that the correct object template and motor state are actually activated by the sensorimotor function when the decision function chooses that deictic operation.

After the reinforcement process is finished, the agent goes into experience mode again until another reward state is reached. After encountering several reward states, the agent is now able to perform the right action on a certain object in order to reach a reward state.

2.3 A model of sentence syntax: Chomsky's Minimalism

We now have a sensorimotor model. The next stage is to outline a model of natural language syntax, so that we can begin to consider possible ways of mapping between sensorimotor sequences and syntactic representations.

There are several theories of language which can be used. For example, unification-based grammars, used by computational linguists (e.g. HPSG (Pollard & Sag, 1994)) or construction grammar used by many psycholinguists and developmental linguists (which focuses on how language is learned using general-purpose learning mechanisms (Goldberg, 1995)). The theory which Knott uses is **Minimalism** (Chomsky, 1995), the successor to **Government-**

learned the next states, we use the episodic memory to get the next object template and motor state in the sequence.

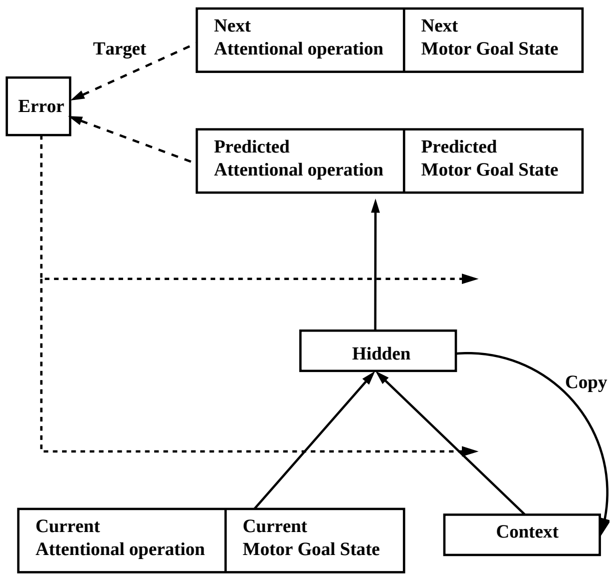

Next Attentional operation
Next Motor Goal State
Target
Error
Predicted Attentional operation
Predicted Motor Goal State
Hidden
Copy
Current Attentional operation
Current Motor Goal State
Context

Figure 2.5: Decision function

and-Binding (**GB**) theory (Chomsky, 1981). The goal of this theory is to express the relationship between the **logical form** (**LF**) and the **phonological structure** (**PF**) of a sentence. Minimalism argues that an adequate linguistic model of sentences requires their analysis not only at a surface level, but at an underlying level at which many of the surface differences between languages disappear as well. This make Minimalism a good candidate for characterisation in sensorimotor terms.

The hypothesis of Knott (2005) is that the structure of any transitive sentence is grounded in the sensorimotor memory architecture described in the previous section. This hypothesis makes a strong assumption about universal grammar: if we assume (as seems plausible) that all humans have the same sensorimotor algorithm regardless of what language they speak, then the linguistic structure of a transitive sentence in any language must be the same at some level of abstraction. The suggestion in Minimalism, that the syntactic representation of sentences in different languages is the same at some underlying level, is clearly consistent with this idea, and therefore Minimalism is the right kind of framework to explore. In the following sections this theory will be elaborated more to illustrate the arguments for the hypothesis. All the information about Minimalism in the next sections if not cited otherwise comes from Knott (2005) (an overview) and Radford (1997) (detailed).

2.3.1 Universal grammar

In Minimalism it is assumed that people possess a **Universal Grammar** (**UG**), which is basically a facility for learning the kind of languages that humans speak. An important argument for this is that children in principle are able to learn any language as their native language. For example, Dutch orphan babies brought up by English-speaking foster parents in an English-speaking community will acquire English as their first language. Moreover, children are able to learn a language in a short amount of time more easily than you would expect if language learning would be just a general learning ability. Another example is

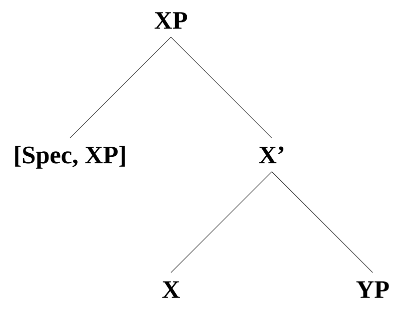


Figure 2.6: The X-bar schema

given by (Baker, 1979), referred to as Baker's paradox (Gropen, 1989): children are able to generalize verbs and nouns in novel frames. Instead of the generalization on irregular verbs (e.g. "doed" instead of "did" and "drinked" instead of "drank") children are able to recover from this overgeneralization. This behaviour could be explained by saying that children gradually learn to constrain the generalization because of negative evidence of their parents. However, parents and adults in general do not give negative evidence that is detailed enough for children to avoid overgeneralization, and so it remains unclear how children learn the constraints. From these observations it is argued that there must be a language faculty which provides a set of **principles of Universal Grammar** so that children are able to learn any natural language if enough linguistic experience in that specific language is provided. This experience should contain examples of words, phrases and sentences in this language produced by native speakers. Therefore, an important feature of a theory of language is learnability. The UG theory developed by Chomsky (1995) accounts for a rapid grammatical development. This theory will be used in the next sections. In the next paragraph we will introduce the basic structure of Minimalism.

2.3.2 Building blocks of Minimalism

Minimalism uses **X-bar schema's** as the basic building block in syntactic representations. The X-bar theory is introduced by Jackendoff (1977) and is still used as a basic element of most theories of grammar. This is because it is one of the most stable components of generative grammar, actually of all theories of grammar. The main idea is that at a certain level of abstraction all phrases have the same structure. The central element is its **head**. This head creates a grammatical constituent, called a phrase, which has certain standard slots which can be filled by other elements. The basic examples: a noun projects a noun phrase (NP), a verb projects a verb phrase (VP), a preposition projects a prepositional phrase (PP).

The X-bar schema is given in Figure 2.6. X is the head of the phrase. This can be a lexical head - an open-class lexical item - such as a noun, verb or adjective, or it can be a closed-class item, such as a preposition or an inflection. In Figure 2.6 the complement of X is labelled 'YP'. This is intended to stand for 'any maximal projection', as XP does. Hence, the X-bar schema is recursive: one maximal projection has slots in it which are to be filled by other maximal projections. The relationship between two X-bar schema's is given in Figure 2.7. The X-bar schema is the building block for the hypothesis of Knott.

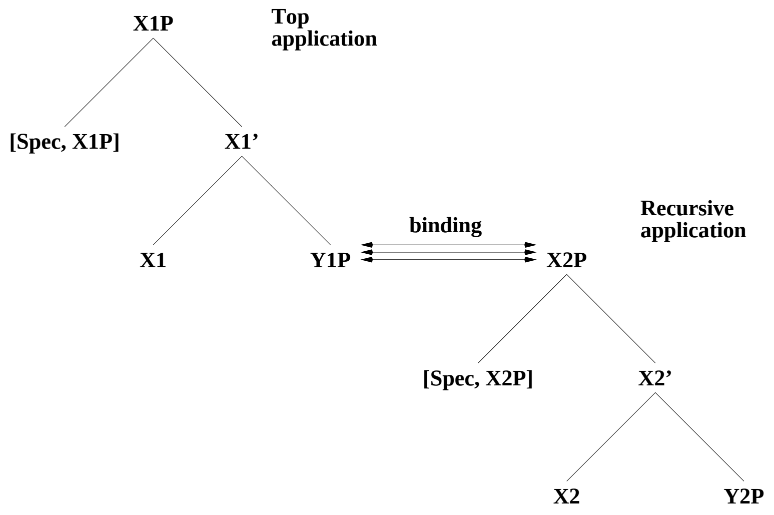


Figure 2.7: recursive application of the X-bar schema

2.3.3 The structure of the logical form of a sentence

In the introduction of this section the term **logical form** (**LF**) of a sentence was mentioned briefly. To put it simply, the LF is the 'underlying' representation of a sentence, which abstracts away from the surface differences in word order between different languages. It is supposed to be the syntactic representation which 'interfaces' with semantics. In Minimalism sentences are described by trees, built up by X-bar schema's. An example of the LF in English of the sentence *A man grabbed the cup* is given in Figure 2.8.

Note that the tree is built from recursive X-bar schema's. In the figure are also a few terms which we have not seen before and need some more explanation. (We will not go into this deeply, however the following ideas are all well-established in the syntactic literature.)

First, on top of the tree there is an element IP. This stands for inflection phrase, the GB term for 'sentence'. The whole clause is seen as a projection of the inflection of its main verb, which is termed I. In Minimalist syntax a verb (V) appears at a separate point in a syntactic tree from its inflection. The left-hand daughter of IP (Spec, IP) is associated with the subject of the sentence.

Second, noun phrases are relabelled as **DP**s or **determiner phrases**. Abney (1987) argued that determiners should be the heads of the phrases which plug into the subject and object positions in a sentence structure. Following an argument by Koopman & Sportiche (1991), the subject DP appears in two positions; it originates at [Spec, VP], and raises to [Spec, IP] as shown by the dotted arrows.

Third, there is an extra XP intervening between IP and VP called **AgrP** (or **agreement phrase**). This projection was originally suggested by Pollock (1989). Later, Chomsky (1995) proposed that the object DP should also appear in two positions: first as the complement of V, and second as [Spec, AgrP].

A key idea in Minimalism is that movement of constituents is invoked at LF. The top diagram in Figure 2.8 shows LF after this movement, and the lower diagram shows LF before movement has taken place. The suggestion is, for instance, that the subject *the man* is originally generated at the specifier of VP, but for various reasons has to move to the specifier

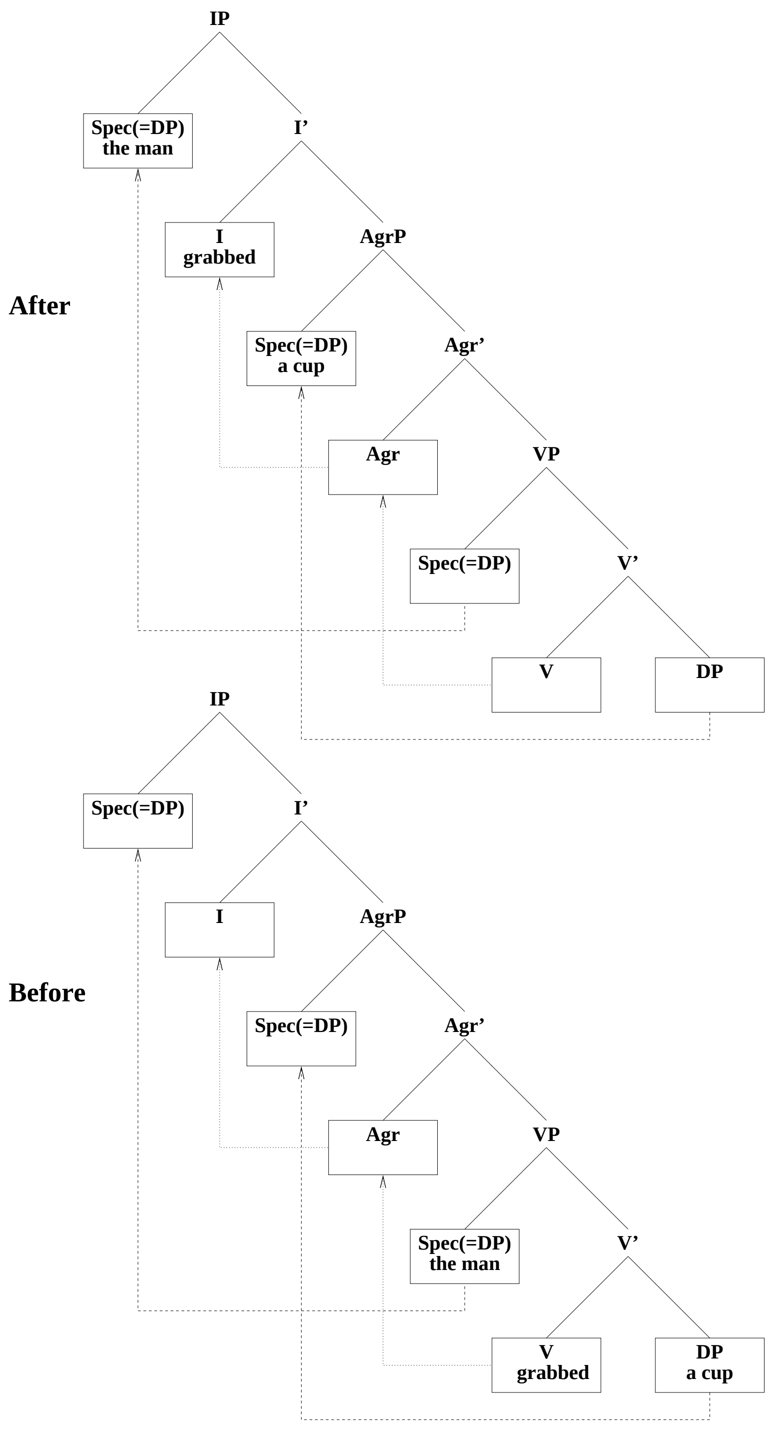


Figure 2.8: The logical form (LF) of *the man grabbed a cup* after movement and before movement

of IP. The object undergoes a similar kind of movement. The verb undergoes a different kind of movement, not to a specifier position, but successively to the heads of AgrP and IP, i.e. Agr and I. In fact, the verb is not always subject to movement. In finite sentences, it moves as shown, and *picks up* the verb inflection at Agr and I, in non-finite sentences, it remains at V, and hence is uninflected. Movements will be discussed further in Section 2.3.4.

We will discuss some of these features elsewhere in this report. At this moment it is only important to have a notion of the representation of a sentence in Minimalism. In the next section we will talk more about movements of constituents.

2.3.4 Mapping of LF to PF

In the last section we have introduced the LF. In this section, the **phonological structure** (**PF**) of a sentence will be introduced by some examples. It is well known that languages differ from one another, not only in their vocabulary, but also in their syntactic structure. In particular, different languages have different word orderings. However, there are groups of languages which have similar word or phrase ordering, and these have been extensively studied (see e.g. Greenberg, 1963). The following sentences illustrate all the possible orderings in natural languages.

(2.3) Subject - Object - Verb (SOV)

John	ga hon o	yonde-iru.	(Japanese)
John	the book ACC	read-Present	

"John read a book"

(2.4) Subject - Verb - Object (SVO)

Weiyo	tau	moru.	(One)
Weiyo	build	house	

"Weiyo build a house."

(2.5) Verb - Object - Subject (VOS)

No-sai	te	kolikoli	na	La Udi.	(Tukang Besi)
make		canoe	NOM	Mr Udin	

"Udin's making a canoe."

(2.6) Verb - Subject - Object (VSO)

Lladdodd	y	dyn	y	ddraig.	(Welsh)
killed	the	man	the	dragon	

"The man killed the dragon."

(2.7) Object - Verb - Subject (OVS)

Kaikuxi	etapa-ã	toto.	(Apalaí)
jaguar	kill-TENSE	they	

"They killed a jaguar."

(2.8) Object - Subject - Verb

Manaiin	Subih	a-wa.	(Nadëb)
cara	Subih	eat	

"Subih is eating cara."

The most common orderings are from the first examples, the Subject-Verb-Object (SVO) order (e.g. English, Dutch) and the Subject-Object-Verb (SOV) (e.g. Korean, Japanese) order. Minimalism claims that these different orderings are caused by a difference in the derivation from the underlying structure to the PF. Minimalism calls this derivation '**spell-out**'. In figure 2.9 this generative mechanism is shown schematically.

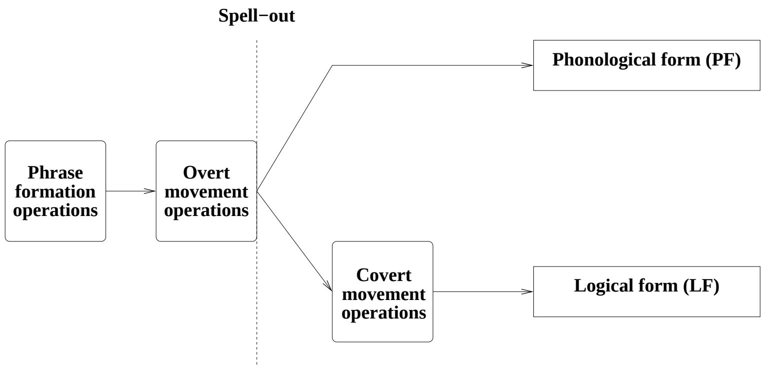

Spell−out
Phonological form (PF)
Phrase formation operations
Overt movement operations
Covert movement operations
Logical form (LF)

Figure 2.9: derivation in Minimalist syntax

In Minimalism the generative mechanism begins with a series of phrase-formation operations which create the basic LF tree structure associated with a sentence. Then a series of movement operations are executed on this basic structure, as was already mentioned. At some point during these movement operations, the PF of the sentence is *read off.* Movement operations which occur before this point are called **overt**, because they are reflected at PF. Those occur afterwards are called **covert**, because PF does not reflect them.

The spell-out can also be explained by showing a sentence as an X-bar schema. Assume that the tree shown in Figure 2.10 is the tree generated after a series of phrase-formation operations. Note the numbered arrows: these are movements. These movements can occur before or after spell-out. Minimalism says that the different word orders of different languages are caused by these movements, because they can occur before or after spell-out as we have described in the previous paragraph. For example, in English and One (see example 2.4) only movement 1 occurs before spell-out, hence, this movement is overt. For the SOV-order as in Japanese, the movements 1 and 2 are overt.

For non-linguists the figures of the syntactic structures we have seen so far might look complicated. One might ask why the structure has to be so complex. Linguists argue that a language model must explain all well-formed sentences in all languages; therefore a model has to obey a high number of constraints, and hence a language model cannot be simple. On the other hand, if we try to model brain functions nobody will argue against a high complexity of this model. Recall that the hypothesis of Knott (2005) says that the language faculty is grounded in the sensorimotor apparatus. If this hypothesis is correct it means that linguistics are actually modelling parts of the sensorimotor apparatus. This could explain why a syntactic model is so complex. In the next section we will talk more about this relationship.

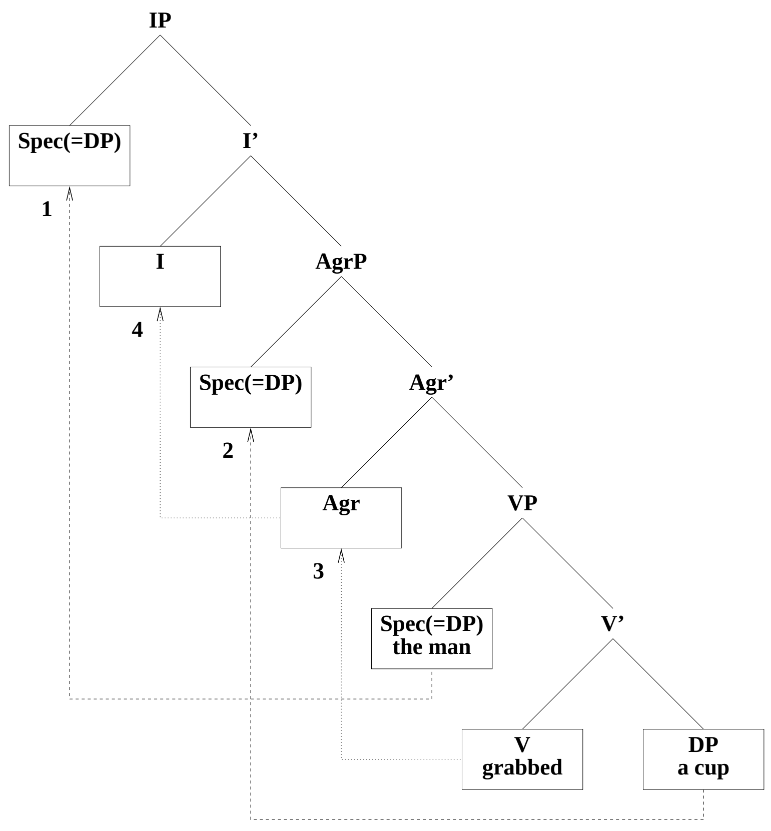


Figure 2.10: The possible movements after phrase-formations operations

2.4 Knott's sensorimotor interpretation of LF

In this section Knott's hypothesis about the relationship between sensorimotor and syntactical representations will be outlined. We will discuss his idea of the relation between the model of sensorimotor learning and memory (Section 2.2), and the model of natural language syntax from the previous section. The ideas we are talking about in this section are all adopted from Knott (2005). To begin with, we introduce Knott's general proposal:

Principle 1 The LF of a sentence describing an eventuality encodes the structure of the episodic memory representation of the sensorimotor process via which this eventuality was witnessed.

The idea is that the structure of the sensorimotor sequence has overlap with the structure linguists have given to a sentence at LF. In other words, Knott believes that the LF of a sentence is an encoding of the sensorimotor sequence. Knott's specific proposal is given below:

Principle 2 An application of the X-bar schema in LF can be understood as a description of the processing that occurs at a single time point in the execution of the episodic memory network.

- XP denotes the **current sensory state/context**;
- YP denotes the **next sensory state/context**;
- X denotes an application of the **decision function**, whose input is the current state/context and whose output is a **deictic operation**;
- [Spec, XP] the next object template generated during the current iteration of the episodic memory network.

This principle states an overlap between a syntactic structure (an LF X-bar schema) and a sensorimotor structure (the sensorimotor model). In Figure 2.11 this principle is given graphically. Recall that an X-bar schema is recursive (see Figure 2.7). Every time point of the sensorimotor sequence can be understood as an encoding of a single X-bar schema. An X-bar schema application is taken to be a description of what happens at one time point in the sensorimotor model; in other words, it links the current state to the next state.

Knott applies this principle to a whole clause. For example, the transitive sentence *The man grabbed a cup* is a right-branching structure of X-bar schema's. According to Principle 2, this can be understood as a description of a sequence of several iterations of the episodic memory network. Recall, the seven stages of event perception discussed in Section 2.2.3. In Figure 2.12 these stages are mapped onto the syntactic structure. The figure shows that the order of the description at sensorimotor level is the similar to the order at syntactic level. In particular, the DP-movement for the subject and the object is very much the same as the

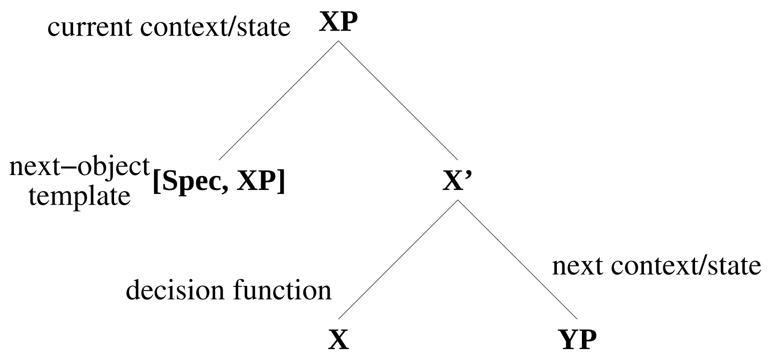


Figure 2.11: Reinterpretation of the X-bar schema (adapted from Knott (2005))

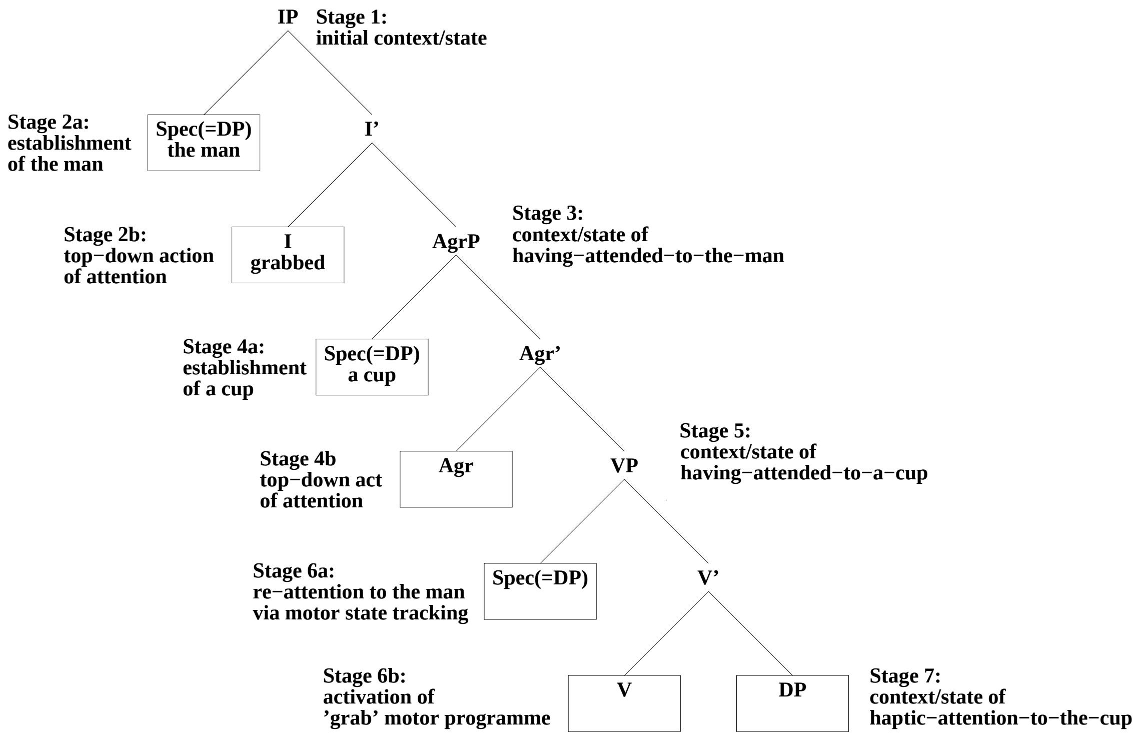

IP
Stage 1: initial context/state
Stage 2a: establishment of the man
Spec(=DP) the man
I’
Stage 2b: top−down action of attention
I grabbed
AgrP
Stage 3: context/state of having−attended−to−the−man
Stage 4a: establishment of a cup
Spec(=DP) a cup
Agr’
Stage 4b top−down act of attention
Agr
VP
Stage 5: context/state of having−attended−to−a−cup
Stage 6a: re−attention to the man via motor state tracking
Spec(=DP)
V’
Stage 6b: activation of ’grab’ motor programme
V
DP
Stage 7: context/state of haptic−attention−to−the−cup

Figure 2.12: Sensorimotor interpretation of the LF of *The man grabbed a cup* (adapted from Knott (2005))

re-attention to the objects on sensorimotor level.

Verb movement can be associated with the rewind mode described in Section 2.2.7. Examples 2.9 and 2.10 are two examples of non-finite sentences.

(2.9) I want to grab a cup

(2.10) I can grab a cup

In these sentences the main verb *grab* does not move from *V* to *Agr* to *I*, but remains in its current position *I*[5]. Only finite sentences exhibit verb movement. Since finite sentences describe actions, the sensorimotor model gives an explanation of what is happening. In rewind mode only actual actions can lead to a reward. This rewinding of the model is associated with the movements of a verb in a finite sentence. In non-finite sentences no actual action is executed; there is no reward situation and thus no rewinding. Without rewind mode there is no verb movement in the sentence. For this reason, Knott proposes that *V -> Agr -> I* movement might be a reflex of rewind mode.

In summary, what we have seen so far is that the syntactic structure of Minimalism maps nicely onto the idea of iterative sensorimotor operations. First, the general right-branching X-bar structure of LF maps onto the idea that a transitive sentence is encoded as a sequence of of actions of attention, as each phrase on the LF structure corresponds to an appropriate operation of the sensorimotor sequence. Second, the idea that subject and object DPs are each found in two locations as LF maps neatly onto the idea that the agent and patient are each attended to at two points during the sensorimotor sequence.

In particular the multiple representation of the agent and the patient in the sensorimotor sequence, which are also multiple represented in the X-bar schema as a DP for the subject and a DP for the object. Second, both from the syntactic point of view as the sensorimotor point of view is the observation that sequences make sense since in both models each state is bound to the next one. Finally, Knott suggests that rewind mode gives a natural interpretation for verb movement.

Now we have described Knott's interpretation of the sensorimotor sequence of the LF. In the next section we introduce a model for mapping the sensorimotor sequence onto surface phonological structure.

2.5 A model for mapping the sensorimotor sequence to a word sequence

Knott's model expresses a relationship between the sensorimotor sequence and LF, but the mapping from LF to PF still needs to be explained. In this section, we will describe this mapping, and then propose a concrete model.

2.5.1 Elman networks for sequencing words

Humans never produce or receive a sentence as a whole; instead we attend to it step by step, we produce one word at a time. Therefore, it is not biologically plausible to use a normal feedforward network when making a language model. Instead, a network that can learn

[5]We will not describe why Minimalism argues there is no movement here.

sequences is needed. Elman (1990) described a network which is able to extract temporal patterns from sequences (see Section 2.1 for a short introduction on Elman networks).

Elman (1990)'s original network was in fact designed to encode word sequences. Elman used the network described in Section 2.1. The goal of the network was to predict the next word given the current context. A generator program was used to create training data consisting of short (two- and three-word) sentences. Thirteen different classes of nouns and verbs were chosen. These nouns and verbs were put randomly into 15 sentence frames. This way the generator created a data set of 10000 sentences. These sentences were concatenated. Each word was presented as a binary vector. When trained, the network did not have an impressive low error rate. However, the task is nondeterministic. Successors cannot be predicted with absolute certainty; in the task is a built-in error which is inevitable. In spite of this, any given word had only a limited number of successors. Suspecting that the network did learn something useful, Elman inspected the internal representations of the hidden units. Elman found that the network had developed internal representations for the input vectors which reflected syntactic and semantics similarities for words, which were the factors which it learned were responsible for facts about the possible ordering of the inputs. The network was not able to make exact predictions about the word order, but it did recognized that different words belong to different classes of inputs. This property of an Elman network will be used in the model proposed in Section 2.5.3.

2.5.2 Adding semantics to an Elman network

Although Elman (1990) showed that Elman networks are useful when learning sequences, the original Elman network has no treatment of semantics, it is just a device for predicting the next word of a sentence, or at least, the type of the next word. It seems clear human syntactic competence is not just an ability to predict word sequences statistically (Fodor & Pylyshyn, 1988; Marcus, 1998; Chang, 2002).

Chang (2002) presented a model, the so-called dual-path model that both had a semantic representation of a sentence and additional input to an Elman-style word-prediction network. Training data consisted of pairs of semantic representations and word sequences. When trained, the network was able to take unseen semantic representations and create an appropriate sequence of words.

In Chang's network, the message is stored in two pairs of connections between thematic role units (agent, patient, verb) and the object/action representations (man, grab, cup). Chang's model (see figure 2.13) has two subsystems (pathways); the message-lexical system and the sequencing system. The first subsystem is a feedforward network which has two separate layers to represent respectively the thematic and the semantic contents of the message (as weights). The second subsystem is a simple recurrent network with some extra inputs. This network creates a message independent representation of the message content because it has no access to the message. The message-lexical system and the sequencing system have interactions via a connection from the hidden units of the sequencing system to the "where" units of the message-lexical system. The second interaction points are the word units. Here the message-lexical system activates meaning related possibilities, and the sequencing system activated syntactically-appropriate possibilities. In short, the dual-path model is biologically plausible since it simulates the learn-curve of children (overgeneralization and overcoming this overgeneralization without explicit reinforcement) and it's able to explain the double dissociation's in aphasia. The model has a subsystem for representation of the semantics and a subsystem for representations of the syntax. The model explains some linguistic puzzles and has a natural explanation of how children avoid the Baker's paradox; children are able to overcome the overgeneralization without negative evidence from their parents.

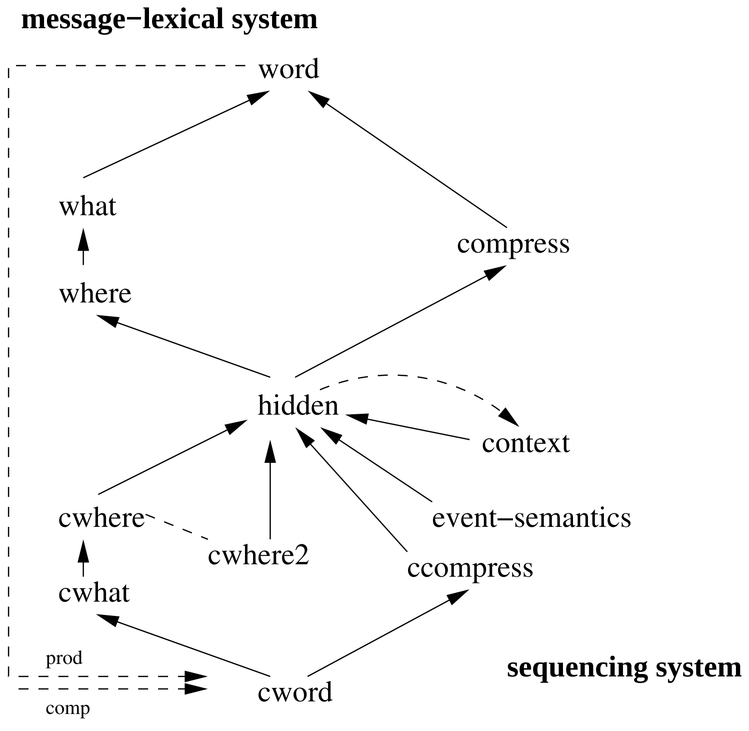


Figure 2.13: The dual-pathway model (adopted from Chang (2002))

We will follow Chang in augmenting a word-sequencing Elman network with a representation of semantics, and in providing training data consisting of pairs of word sequences and semantic representations. However, our encoding for semantics is quite different from Chang's: for us, the semantics of a sentence is itself a sequence, rather than a static mapping of concepts to thematic roles. The key task for the LF-to-PF mapping function, then, is to map a sensorimotor sequence onto a sequence of words.

2.5.3 The word-sequencing model

We have now come to the part where a concrete model of the mapping between LF and PF is proposed. In the previous sections we have outlined the problem and have mentioned a few constrains the model must have. In Section 2.3 we have briefly described Baker's paradox. The model we will propose, ideally, will be able to explain the self-constraining of overgeneralization on verbs. However, the focus is first to explain the different orderings of languages as we have talked about in Section 2.3.4. We want to map a word sequence onto a sensorimotor sequence from the model of episodic memory described in Section 2.2. Figure 2.14 illustrates a simplification of this mapping. Note that stage 1, 3, 5 and 7 are not mentioned in this figure, because these stages are binding the states and so are only implicitly part of the sequence. Since children learn their language without negative evidence our goal is to design a model that only uses positive evidence.

The model we need to build has to take as input a sensorimotor sequence of the states generated by the episodic memory network, and produce as output a sequence of words. This mapping is shown in Figure 2.14. In this figure we talk about gaps. These gaps represent

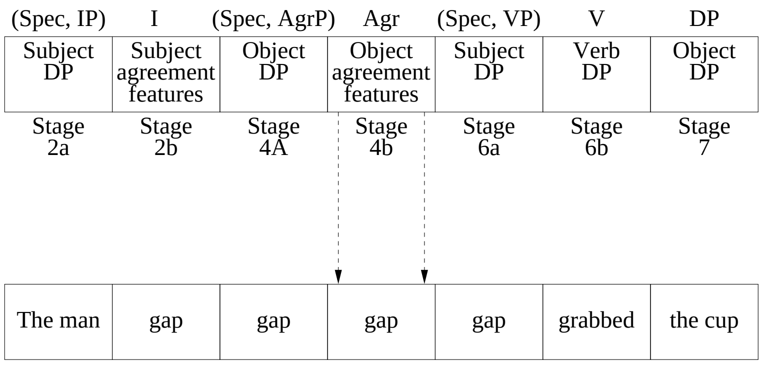


Figure 2.14: The mapping of a sensorimotor sequence onto a word sequence in English

lexical items that are not visible in the word sequence. These gaps are necessary to explain the difference in length between a sensorimotor sequence produced by the episodic memory and a word sequence. In literature the first reference to gaps in sentences is from Lakoff (1970).

The model we propose has too stages. In the first stage, the model will learn the mapping between a random sensorimotor item and a word. This stage represents the stage of word learning with children. The next stage is trying to produce complete sentences. At this stage, the model has to learn in what context to actually propounce the word associated with a sensorimotor item, as when to generate a gap.

The model is an extended Elman network as shown in Figure 2.15. The difference with a normal Elman network is the extra input for the sensorimotor sequence. We expect the model is able to link the different categories of the sensorimotor sequence to different states in the network, so that the model can predict the next state. Recall the task in the experiment of Elman (described in Section 2.5.1), this task was nondeterministic. However, our task is deterministic. The sensorimotor sequence combined with the current context gives enough information to know absolutely which word to predict. There is no built-in error as in the task of Elman. We should expect that when the model is trained the error on a test set which contains familiar words therefore tends to zero. During the first stage the model is trained on random sensorimotor items. This stage can be seen as the stage when a child's parent looks at an object structure, and names it for the child. The model will learn the relation between the sensorimotor items and words. It will ignore the context since that does not yet deliver anything meaningful. The second stage is when the episodic memory model is becoming better at its task. Sensorimotor items have a certain order at this stage. The model will produce the same word mappings; however in this case the sensorimotor sequences are descriptions of events in the world. Thus the sequence can be mapped on a correct sentence. When producing the word sequences the model will now compare the prediction with the desired word. If this is not correct the model will assume that it has to ignore[6] this sensorimotor item in the current context, and generate a new piece of training data for itself

[6]Ignoring a sensorimotor item is equivalent to predicting a gap.

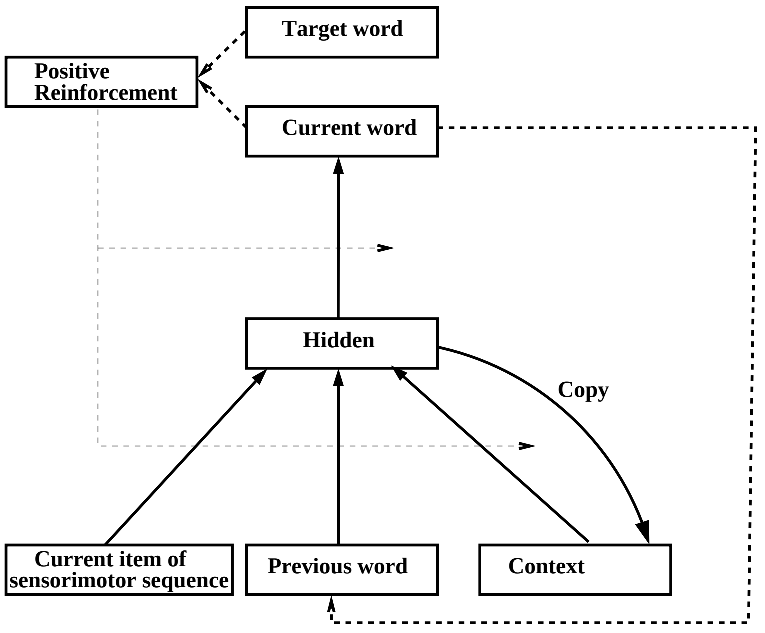


Figure 2.15: The word-sequencing model

to reflect this fact. If it is correct, the model will also generate a new piece of training data for itself, recording the context in which an explicit word is expected.

This procedure is summed up below:

1. Initialize the context.
2. Train the model to map a sensorimotor item on a word (there is no order in the training sequence).
3. Try to predict the word when representing the sensorimotor sequence of a concrete event.
4. If the prediction is correct add the prediction to the new training patterns.
5. If the prediction is wrong add a gap to the new training patterns.
6. Go back to step 2 until all sensorimotor sequences are seen.
7. Train on all the new training patterns in step 4 and 5.

The hypothesis is that for children this is the process where they are producing their first sentences. Although not corrected by their parents or adults, they will see that they are not understood very well when using the original mapping. Therefore, they have to adjust the first learned rules about the word mapping so that they can use the current context as an extra fact whether or not to ignore the current sensorimotor item. These rules are different for the different language orderings (SVO, SOV, OVS, OSV, VOS and VSO). The model will simulate this by training on new patterns created on-the-fly while predicting the words. After learning the patterns the model should be able to predict the next word given the current sensorimotor item. It must be able to learn different mappings between sensorimotor sequences and word sequences, to reflect the fact that different languages have different constituents ordering. In the implementation section we will describe this model in more detail.

Chapter 3

Implementation and evaluation

This chapter discusses the implementation of both the sensorimotor model (Section 3.2) and the syntactic word sequencing model (Section 3.3). Since learning in both models involves the learning of sequences, there is the problem of catastrophic interference to overcome. The implications of this problem and possible solutions are addressed in Section 3.1.

As far as the implementation, we started by searching for a good neural network simulator tool which could help us with building and evaluating different kinds of networks easily. Packages we looked at were: SNNS, JavaNNS, Lens and Interact. After experiencing disappointing results with these tools, we decided to build the whole model by hand in the programming language PHP. This programming language proved to be too slow to learn the networks in a descent amount of time. That is why the final implementation of the networks and supporting functions are written in Java.

3.1 Preliminaries: Catastrophic interference

To learn patterns sequentially in a feedforward network, one has to overcome the problem of catastrophic interference. In short, when a pattern is presented to the network it approximates the function between inputs and outputs and stores this as a set of weights in the hidden layer. Once a second pattern is presented, the weight changes will override the function learned for the previous pattern. The network will not be able to produce the correct output for the previous pattern and has thus completely forgotten it. This is exactly what catastrophic interference is.

One solution would be to train the network on both the new pattern and all the previous patterns the network has seen so far. Unfortunately, in real world situations it is unlikely that all these previous patterns are still available when the new pattern is being presented.

A solution proposed by Robins (1995) is to use pseudopatterns to simulate previous learned data. Pseudopatterns are like normal patterns, but generated by the network to approximate the function learned by the network. A pseudopattern thus consists of both an input and a target value. A pseudopattern is created by using a random value as an input for the learned network. The output of the network is stored as the target value for this random input value. The generated pseudopatterns are then used together with the new pattern to learn the new function which explains both the new and previous input patterns without the need of these previous patterns when learning a new pattern.

In order to be able to build a model of episodic memory, the network must not only be able to learn patterns sequentially, but must also learn multiple sequences. A sequence can be learned by Elman networks. But Elman networks are also subject to catastrophic interference when a new sequence overwrites the function learned for the previous sequence. A model using pseudopatterns in Elman networks is proposed by Ans, Rousset, French & Musca (2002). This network generates and uses pseudopatterns in the same way as the standard feed-forward network discussed earlier.

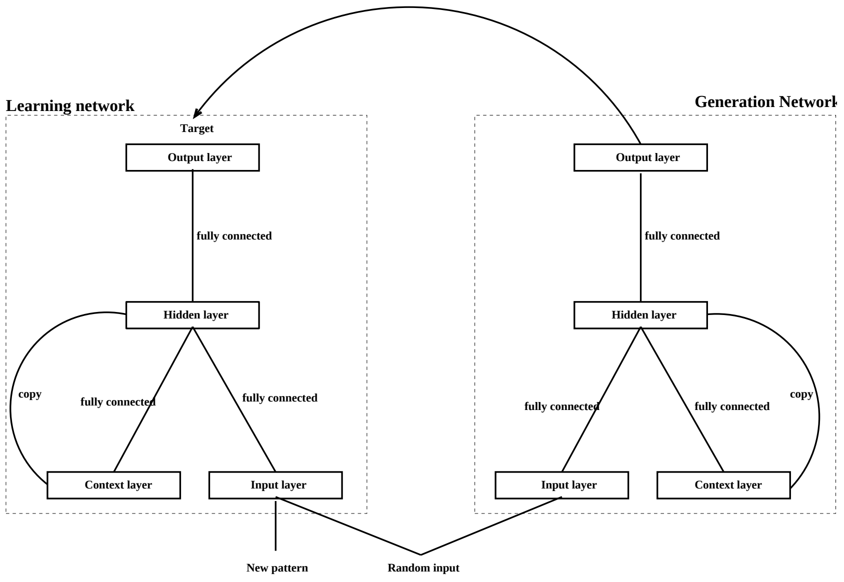


Figure 3.1: Coupled reverberating Elman Networks

Learning with pseudopatterns requires two parallel networks as shown in Figure 3.1. One network to represent the learned function for all the previous patterns, which is called the ‘generation network’, and another network to learn the new pattern, called the ’learning’ network. The pseudopatterns are generated by the generation network and used, together with the new input pattern as input for the learning network. Once the learning network has learned these patterns the network needs to be copied to the generation network. This can be achieved by using a new set of pseudopatterns, generated by the learning network, as an input for the generation network.

Because of the possible implausibility of using dual networks, there are other solutions to overcoming the catastrophic interference problem. One solution (Robins, 1997) is to use two weights on a connection: one slow learning and one fast learning weight. This way the network can prevent catastrophical forgetting with just a single network. Another single network solution (Ans, 2004) is to use clamp states on input and output neurons, to switch the network between a ’new pattern mode’ and a ’pattern generation mode’.

There are a few solutions to the problem of catastrophic forgetting. The two models that will be discussed in the next sections, overcome this problem by training the new pattern in combination with all the previous patterns. Any of the other solutions would have worked as well, but would take longer to implement.

3.2 Implementation of the sensorimotor model

This section discusses the implementation of the model described in Section 2.2. First we look at the training data and how it can be put in the right format to be used in this model. Next we discuss both the experience mode and rewind model. The implementation of the recall mode will not be discussed. We discuss the implementation of the last two sub-processes of the model: the decision function and the sensorimotor function. Experiments on all the different parts of the model show us in what sense the model is plausible and what problems arise.

3.2.1 Training data

Assume the agent operates in a world as displayed in 3.2. The observer can attend to three

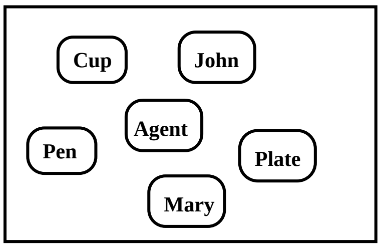


Figure 3.2: Simplified worldmodel

different objects in the world: a cup, a pen and a plate. There are also three agents in the world the observer can attend to: John, Mary and the observer himself. The agent has an action repertoire of four actions: raise arm, cup-grasp, plate-grasp and pen-grasp. Each of these objects and actions are represented as a template. A template is a sensory state if it concerns the perception of an agent or an object. A template is a motor state if it concerns the perception or execution of an action. See Table 3.1 for an overview of the templates in this world. The episodic memory is a neural network implementation which has to learn to remember sequences of object templates and motor states. For a network to be able to learn these templates, they have to be encoded into bytes. Table 3.2 shows a simplistic example of this encoding. Object templates and motor states always go together. If the observer perceives an object on which an action is executed, the next state of the model contains both the object template and the motor state. On the other hand, if the observer perceives an object on which no action is executed, the next state will be an object template and an empty motor state.

A pair of an object template and a motor state is used by the decision function to choose the next state. The decision function outputs a deictic operation, which contains two parts: an attentional operation and a motor goal state. The attentional operation is presented as a set of preferred features, which the visual system uses to find such an object. The motor goal state is presented as a set of goal states for joints and muscles. In the current implementation, the motor goal states are simplified. Table 3.3 shows the deictic operations for this world.

Deictic operations also have to be encoded in bytes for the decision function to be able to produce them. Each byte stands for a specific feature. An example encoding is shown in Figure 3.3. Some features, or groups of features, within the attentional operation are

Object templates		
T_{cup}	=	cup
T_{plate}	=	plate
T_{pen}	=	pen
$T_{oneself}$	=	oneself
T_{john}	=	John
T_{mary}	=	Mary
Motor states		
$T_{raise-arm}$	=	raising arm
$T_{plate-grasp}$	=	grasping the plate with a plate-grasp
$T_{pen-grasp}$	=	grasping the pen with a pen-grasp
$T_{cup-grasp}$	=	grasping the cup with a cup-grasp
T_{none}		no action selected

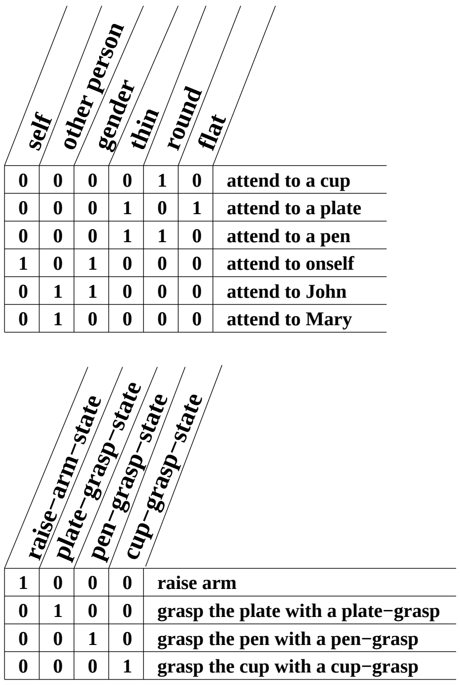
Table 3.1: Object and action templates

Object templates		
1 0 0 0 0 0	=	cup
0 1 0 0 0 0	=	plate
0 0 1 0 0 0	=	pen
0 0 0 1 0 0	=	oneself
0 0 0 0 1 0	=	John
0 0 0 0 0 1	=	Mary
Motor states		
1 0 0 0	=	raising arm
0 1 0 0	=	grasping the plate with a plate-grasp
0 0 1 0	=	grasping the pen with a pen-grasp
0 0 0 1	=	grasping the cup with a cup-grasp

Table 3.2: Byte encoding of templates

Attentional operations		
O_{cup}	=	attend to a cup
O_{plate}	=	attend to a plate
O_{pen}	=	attend to a pen
$O_{oneself}$	=	attend to oneself
O_{john}	=	attend to John
O_{mary}	=	attend to Mary
Motor goal states		
$A_{raise-arm}$	=	raise arm
$A_{plate-grasp}$	=	grasp the plate with a plate-grasp
$A_{pen-grasp}$	=	grasp the pen with a pen-grasp
$A_{cup-grasp}$	=	grasp the cup with a cup-grasp

Table 3.3: Deictic operations. 'O' stands for objects and 'A' stands for actions



self	other person	gender	thin	round	flat	
0	0	0	0	1	0	attend to a cup
0	0	0	1	0	1	attend to a plate
0	0	0	1	1	0	attend to a pen
1	0	1	0	0	0	attend to onself
0	1	1	0	0	0	attend to John
0	1	0	0	0	0	attend to Mary

raise−arm−state	plate−grasp−state	pen−grasp−state	cup−grasp−state	
1	0	0	0	raise arm
0	1	0	0	grasp the plate with a plate−grasp
0	0	1	0	grasp the pen with a pen−grasp
0	0	0	1	grasp the cup with a cup−grasp

Figure 3.3: Byte encoding of attentional operations and motor goal states

competing. This means that only one of them can be activated in a deictic operation before it is mapped onto an object. In this world the following groups of features are competing: {self, other person, gender} and {thin, round, flat}. And the following single features are competing: {self} and {other person}.

The **reward states** in our model are all the states in which an agent successfully grasps an object. Table 3.4 shows the sequences that lead to a reward state for the world described in Figure 3.2. Observing another agent in the world performing a deictic operation could also result in a reward situation.

$\{O_{oneself}, O_{cup}, A_{cup-grasp}\}$
$\{O_{oneself}, O_{plate}, A_{plate-grasp}\}$
$\{O_{oneself}, O_{pen}, A_{pen-grasp}\}$
$\{O_{mary}, O_{cup}, A_{cup-grasp}\}$
$\{O_{mary}, O_{plate}, A_{plate-grasp}\}$
$\{O_{mary}, O_{pen}, A_{pen-grasp}\}$
$\{O_{john}, O_{cup}, A_{cup-grasp}\}$
$\{O_{john}, O_{plate}, A_{plate-grasp}\}$
$\{O_{john}, O_{pen}, A_{pen-grasp}\}$

Table 3.4: Sequences of deictic operations that lead to a reward

3.2.2 Experience mode

The experience mode network is shown in Figure 3.4. Below follows a procedure for how to step through this model. It explains in what sequence the different processes of the model are executed and what data is used.

1. Initialize context layer with random values and choose random object template and motor state.
2. Present the current object template and motor goal state to the episodic memory
3. Decompose the object template and motor state into a deictic operation which is input for the decision function
4. Calculate preferred next deictic operation in the decision function
5. Present this deictic operation to the sensorimotor function
6. Determine what object template to activate and what motor action to execute in the sensorimotor function
7. Calculate predicted next object template and motor state in the episodic memory
8. Compare the next object template and motor state with the target object template and motor state from the sensorimotor function in 5
9. Backpropagate the error between output and target

10. Copy new context from hidden layer to context layer
11. Evaluate the agent's current state, if this is a reward state, switch to rewind mode
12. Else, start again at 2, but use the sensorimotor function output as the new current object template and motor state

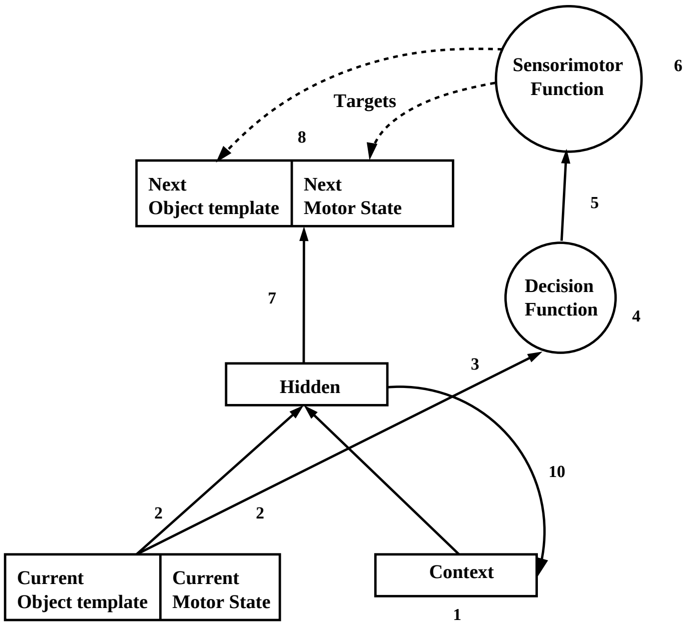


Figure 3.4: Experience mode network

3.2.3 Rewind mode

The rewind mode network is shown in Figure 3.5. Below follows a procedure for how to step through this model. Just as in the previous section, this describes the sequence in which the processes are activated and what data is used.

1. A reward situation occurs
2. Rewind the episodic memory to the start of the sequence that has to be reinforced
3. Present current object template and motor goal to the episodic memory.
4. Use the object template and motor state from episodic memory as an input to the feature decomposition function to get the features of the current deictic operation
5. Generate next item in the episodic memory
6. Present next item to the feature decomposition function to get the features of the next deictic operation

7. Present current deictic operation as an input to the decision function
8. Present next deictic operation as a target to the decision function
9. Train the decision function to generate the next deictic operation from the current deictic operation
10. Copy the next object template and motor state to the current object and motor state
11. Copy the hidden layer output to the context layer
12. If the end of the sequence is reached, exit rewind mode

    The episodic memory has to rewind to the start of the sequence that has to be reinforced. So how do we know how far to rewind the episodic memory? It turns out that it is not really important how far back the episodic memory rewinds as long as we rewind further than the position where the sequence actually started. Motor sequences that can be described by transitive sentences are in general not longer than 4 to 5 attentional actions and or motor actions. After reinforcing the same sequence multiple times, the decision function will learn that only overlapping items at the end of each sequence are important. The other items in the sequences will eventually be regarded as noise.

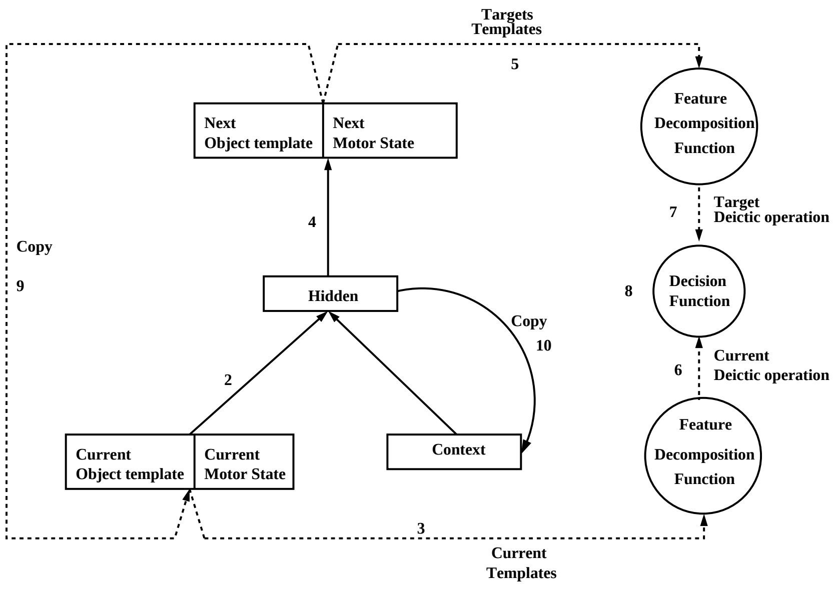


Figure 3.5: Rewind mode network

3.2.4 Decision function

The decision function is implemented as an Elman Network as shown in Figure 3.6. To learn multiple sequences in an Elman Network, all previous sequences have to be learned again (see Section 3.1). Besides this, the network has to learn that each sequence is independent of the other sequences. This can be achieved by creating a new long sequence which contains multiple instances of each sequence randomly concatenated. Below follows a description of the training regime for this network.

1. Present the features from the first item in the sequence to the decision function
2. Generate predicted deictic operation in the decision function
3. Use the object features and motor goal state (deictic operation) from the next item in the sequence as a target for the decision function's predicted object features and motor goal state
4. Calculate error between predicted deictic operation and its target
5. Change the weights of the network by backpropagating the error
6. Go back to 1 and train on the other items in the sequence

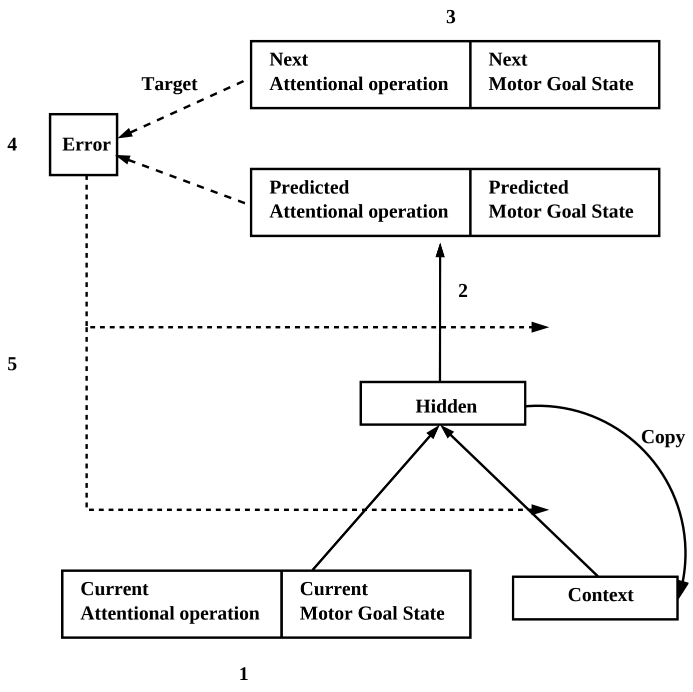

3
Next Attentional operation
Next Motor Goal State
Target
4
Error
Predicted Attentional operation
Predicted Motor Goal State
2
5
Hidden
Copy
Current Attentional operation
Current Motor Goal State
Context
1

Figure 3.6: Decision function

3.2.5 Sensorimotor function

The sensorimotor function implements the mapping of deictic operations onto object templates and motor states. When the sensorimotor function receives a deictic operation from the decision function it finds the object in the world which shares the most features with the attentional operation part of the deictic operation. Before the attentional operation is mapped onto the objects, the competing features are evaluated. The sensorimotor function first competes between groups of features. If the sum of the activation of the group its features are larger than that sum of another group its features, then the first group remains activated, all the features from the other group are deactivated. After this the features themselves compete. The feature with the largest value remains activated, the other is deactivated. Finally, all features, including the features that do not compete with any other features, are evaluated by a threshold function to see if that feature should be present in the actual deictic operation.

This deictic operation is now mapped onto all the objects. When an object is found, the template for this object is activated by the sensorimotor function. It also activates the motor action that results from the motor operation part of the deictic operation. The sensorimotor function is implemented symbolically in an algorithm, not in a neural network. See Figure 3.7. The **feature decomposition** function maps object templates and motor states back onto

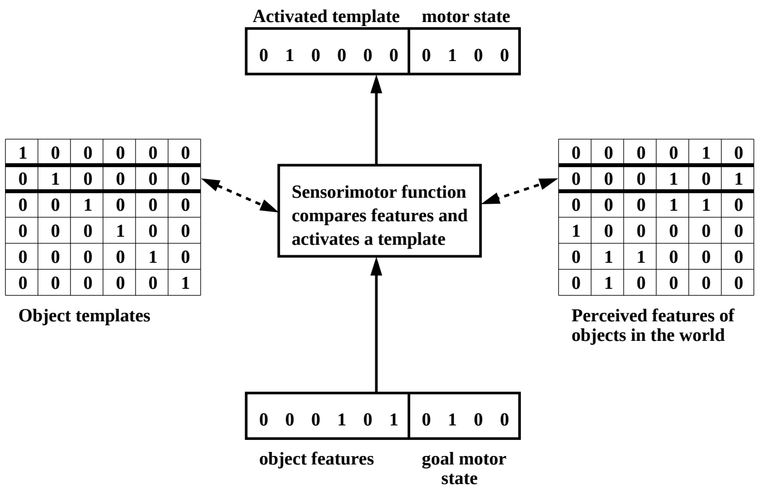


Figure 3.7: Deictic operation on Template mapping

deictic operations, by looking up all the features for the object and the joint and muscle positions for the motor action necessary to get into this motor state. This results in a new deictic operation. The procedure is the same as the procedure of the sensorimotor function and can be seen as reading the Figure 3.7 upside down.

3.2.6 Experiments and Results

Episodic Memory Results

The episodic memory learns a sequence with an Elman Network. In theory, we want the episodic memory to store all the previous object templates and motor states that were activated. To achieve this, the Elman Network capacity must be infinite. The **Elman network capacity** is directly related to the size of the hidden layer and context layer. We conducted a number of experiments which showed that there is a positive relationship between the size of the hidden layer and the number of patterns that can be stored in the network. However, the time it takes to learn a single pattern increases exponentially with the size of the hidden layer; an exponential number of weights have to be adjusted. An infinite memory would thus require an infinite amount of time to learn each pattern. This suggests that Elman Networks are not ideally suited for learning long sequences. Using up to 6 nodes in the hidden layer and context layer is feasible for fast learning and will enable us to store up to 5 patterns.

When we say the episodic memory can store up to 5 patterns, this means that a sequence of the last 5 object templates and motor states can be recalled from memory. Figure 3.8 shows the average error over 100 runs for learning a single sequence of 3 patterns by using 6 hidden and 6 context nodes. Also displayed in this figure are the 95% confidence boundaries based on two times the standard deviation over these 100 runs. The same experiment using different sequences of 3 patterns shows the same trend, only the initial errors are higher.

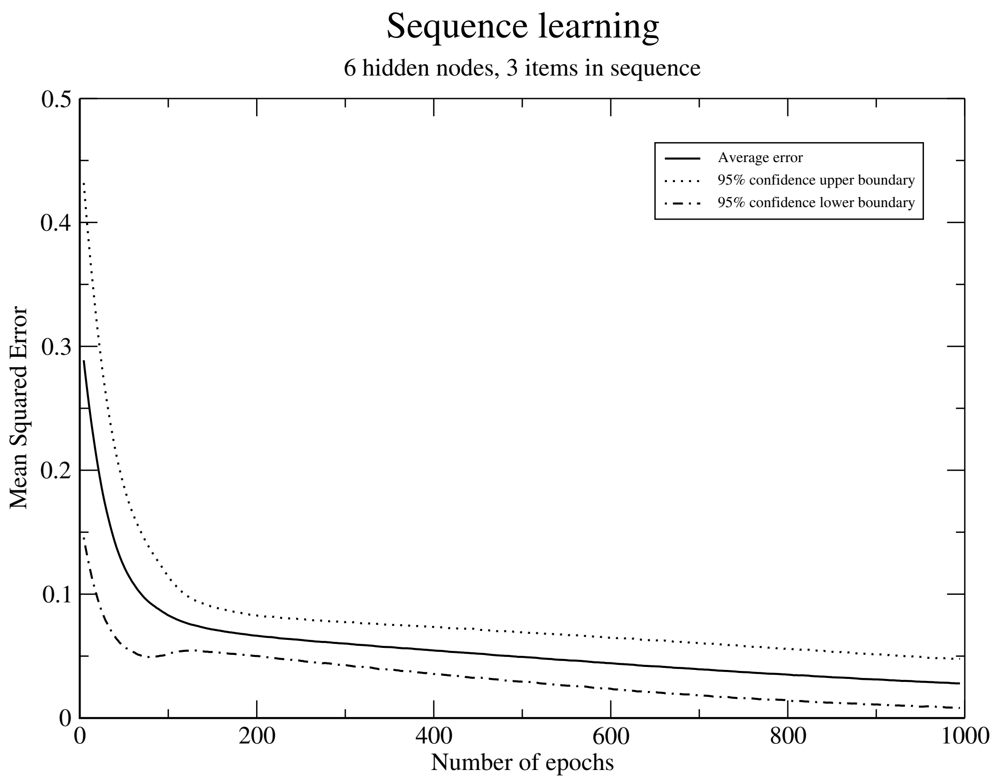


Figure 3.8: Episodic memory learning error

It has proven to be quite difficult to pinpoint a specific error value that would indicate that the whole sequence has been learned. To overcome this, every 500 epochs, the episodic memory tests its learned sequence by running the episodic memory and comparing the episodic memory trace with the sequence that has to be learned. If the sequence matches, the training stops.

Decision function results

Testing the performance of an Elman Network that learns multiple sequences can be tricky, since each sequence that has to be learned is independent of the other sequences. To make sure that the sequences are learned as independent sequences, we first make a single sequence which contains 100 instances of each sequence. Next, the sequences are randomly shuffled, in such a way that the dependent items in each sequence are kept together and only complete sequences are shuffled. We have now created a very long sequence of sequences which can be learned by an Elman Network. Due to the random ordering of the sequences and the limited capacity of the Elman Network, it will not be able to learn the newly created long sequence as a single sequence and will only generalize over multiple sequences within that long sequence.

In the current experiment we will show that an Elman Network can learn all nine reward situations as shown in Table 3.4. The Elman Network used in this experiment has 15 hidden nodes and 15 context nodes. The deictic operations is a combination of an attentional action and a motor goal state as shown in Figure 3.3. The resulting 10 bit vector is presented to the input layer which has 10 input nodes. The output of the network is again a deictic operation of 10 nodes.

The network has to learn a sequence of 100 instances of each of the nine sequences, concatenated into one big sequence as discussed in the previous paragraph. This results in a long sequence of 900 sequences, each sequence containing 3 deictic operations. The result is a training set of 1800 patterns.

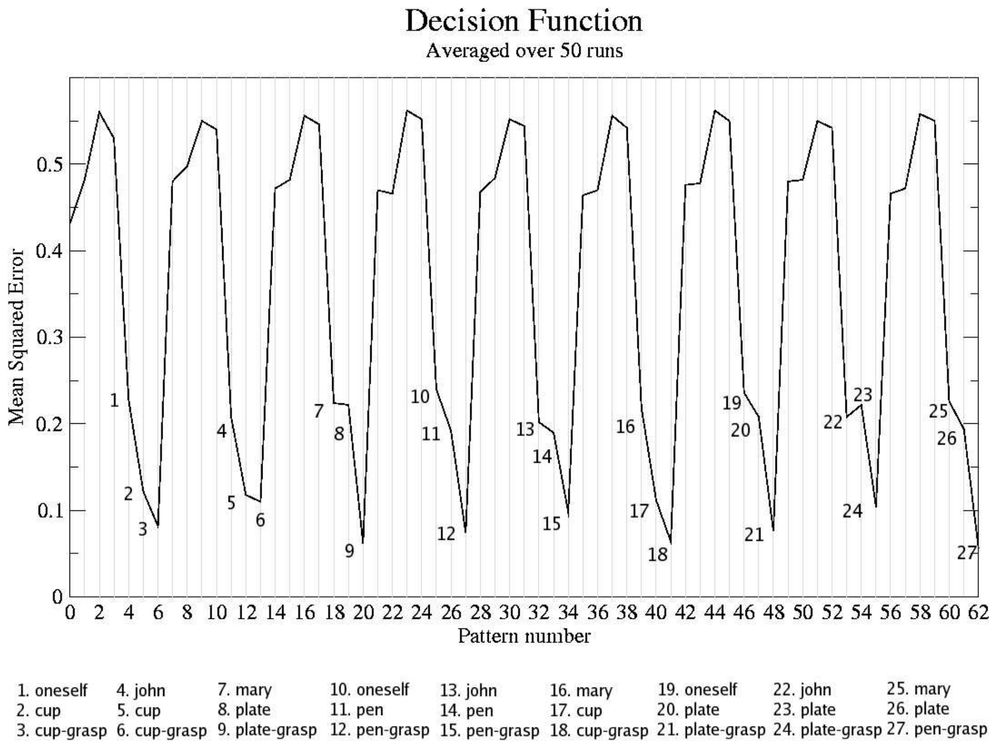


Figure 3.9: Decision function learning error

The network is tested by training the network for 500 epochs and then testing it on the reward states shown in Table 3.4. Each sequence is preceded by a noise pattern of 4 deictic operations to make sure that the prediction of the sequences is correct even if it is preceded by unseen inputs. This noise patterns of course cannot be predicted by the network and have the same output every time, shown as a high peak. The training and testing is done 50 times and the results have been averaged. Figure 3.9 show the mean squared error for all nine sequences preceded by the noise deictic operations.

We will explain the results by looking at the sequence: ‘oneself cup cup-grasp’. The first deictic operation of each sequence has a probability of one divided by it’s frequency in that position. The attentional operation ”oneself” has a probability of 1/3 of being selected. The second attentional operation, ‘cup’, has the same probability of being correctly predicted because we have already seen ‘oneself’. Since there are only 3 sequences which starts with ‘oneself’ the probability of cup being correctly predicted is 1/3 . The third item in the sequence, ‘cup-grasp’, has a probability of 1 of being correctly predicted, since the observer has already seen ”oneself cup”. In other words, the decision function does indeed manage to learn to generate the sequences of deictic operations which move the agent to a reward state.

Since there are 10 output nodes and a threshold output function is used, a 0.1 mean square error means that on the average 1 node provides an incorrect output, 0.2 means that 2 nodes are incorrect, etc. The reason that in the results shown in Figure 3.9 the mean square error of for instance onself, or cup is not equal to 1/3, is that the deictic operations of the objects are really close in Hamming distance. The reason that the mean square error of the motor goal states is not zero, is because once in so many runs the decision function doesn’t perfectly learn the problem. Figure A.1 shows the standard deviation over these 50 runs.

The decision function in experience mode

We’ve seen that the decision function can learn to recognize sequences, even when noise is present in the testing sequence. However, we would like to prove that an observer using the sensorimotor model as described in Section 3.2.2 is able to explore the world it is in. More importantly, after exploring the world, the observer should exploit the decision function to improve the sequence in which it attends to objects and activates motor actions.

The performance of the decision function in experience mode is measured in an experiment in which an observer experiences the world for 20000 cycles. Figure 3.10, shows the average number of sequences the observer evaluates before finding a reward state. Also shown is the number of sequences it takes the observer to find a reward sequence when the decision function is not trained. Learning the decision function does definitely improve the performance of the agent. After finding the first reward state by randomly choosing attentions to objects and motor actions, successful sequences follow each other up more rapidly. Every time the decision function finds a new sequence, the frequency of finding new reward states increases. Eventually, the observer chooses actions in such a sequence that it reaches a reward state at the end of each sequence. This experiment has been done several times, but due to the time variability and time of onset of the first reward state there was no point in averaging the results.

Another interesting question is how many different sequences the observer will explore and exploit. When the observer first starts to experience the world, it doesn’t pay much attention to the decision function since it doesn’t bring the observer to reward states at all. The observer chooses random actions of attention and motor actions until it finds a reward state. The observer starts listening to the decision function more and more as it finds more reward states. By increasing the influence of the decision function just slightly, the observer

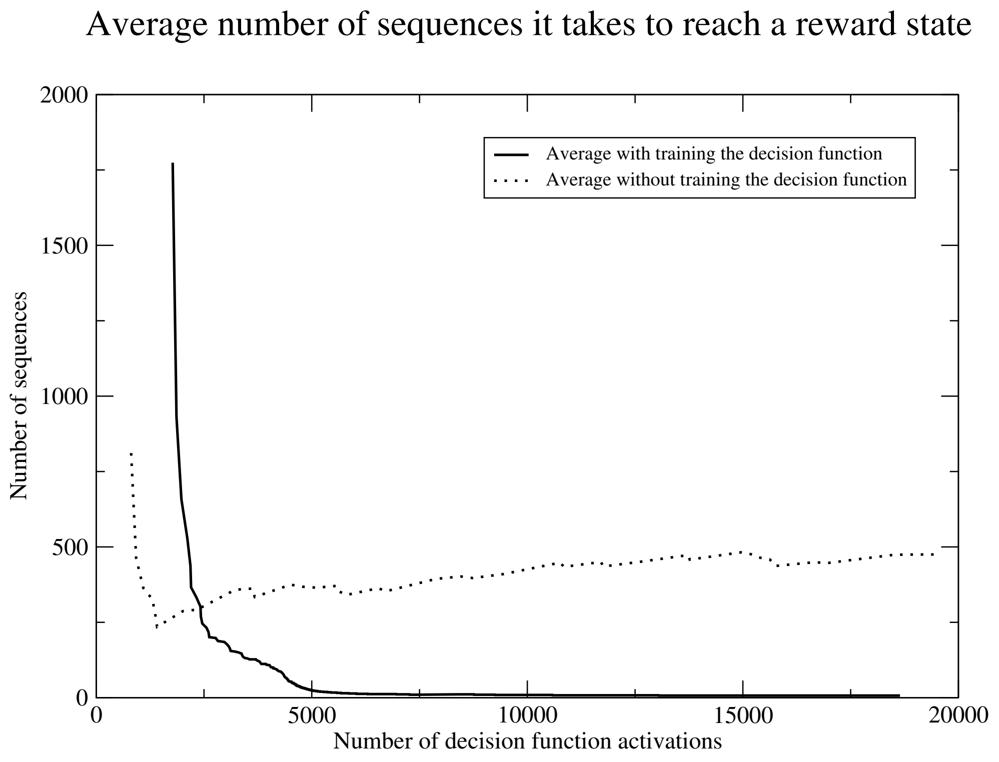


Figure 3.10: This graph show how many sequences it takes on the average, to bring the observer in a reward state

has enough time to find all the reward sequences in the world by random attentional actions and motor actions.

As soon as the decision function is fully trained on all successful sequences in the world the observer listens to the decision function's advice more and more. The agent now starts to exploit its knowledge. As shown in Figure 3.11 the observer does not equally exploit all learned sequences. Due to the fact that the objects are not described by enough different features in the deictic operation, some objects tend to be chosen over others. This is not as such a failure of the decision function and could be solved by adding more features to describe objects and actions. Figure 3.11 shows that 5 of the 9 sequences are chosen frequently, the other sequences are hardly ever chosen.

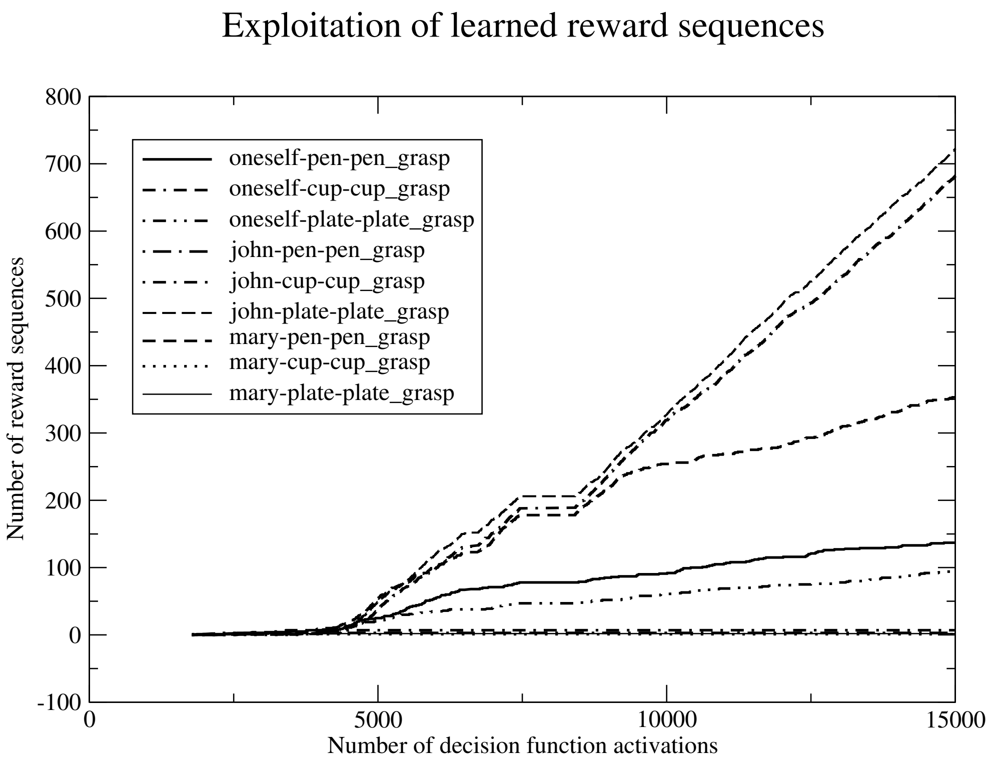


Figure 3.11: Shows how often the different rewards states are chosen during experience mode

3.2.7 Conclusions

Both the episodic memory and the decision function networks are able to learn the sequences necessary for the experience mode to improve over time. However, both these networks have their limitations. As a result the model is, at this stage, not scalable enough.

The episodic memory is able to learn sequences, but to keep the speed of the sensorimotor model within limits, only short sequences are feasible. Elman Networks are just not suited for remembering long sequences. The Elman Network used in this implementation is rather a short term memory, then a long term episodic memory.

The decision function is trained every time a reward state is encountered. The performance of the decision function after it is trained is dependent on the initial state of the network and how soon it can find the function that describes the sequences in the training set. It is not even certain that from this initial state of the network this function can be found at all. This poses a problem for the sensorimotor model in experience mode. As the plasticity of the observer decreases, he tends to take the advice of the decision function more seriously. When the decision function doesn't learn the function that describes the sequences in this stage, it is possible that the observer will not reach any reward states again in the future. The decision function does in this case not give any sensible advice on the best next deictic operation and the randomness in the system is minimal due to the decreasing plasticity.

The decision function also doesn't exploit all learned sequences. Because the deictic operations tend to be really alike, the decision function biases towards some deictic operations more than others.

In Chapter 4 we will propose a number of improvements on the current model which might be able to overcome these problems.

3.3 Implementation: the word-sequencing model

This section discusses the implementation of the word-sequencing model described in Section 2.3. The first phase, when ... word meanings are learned, is presented in Section 3.3.1. The second phase, when the system learns to generate whole sentences, is presented in Section 3.3.2.

3.3.1 Word mapping

Training data

The training data for the model is a sequence of random sensorimotor items, each associated with an appropriate word. In this phase the word-sequencing model learns to map sensorimotor items onto words. A part of this sequence with the target word is shown in Table 3.5.

Sensorimotor sequence	THE DOG	THE CUP	GRABBED	BIKE	SEE
Word sequence	the dog	the cup	grabbed	bike	see

Table 3.5: An example of a target mapping between the sensorimotor sequence and the word sequence

The training data consisted of 11 nouns and 10 verbs. The sensorimotor items and the word items were translated into bit vectors. Each word and sensorimotor item was randomly replaced by a bit vector each training run . Each vector was 22 bits long and had only one bit flipped on. This encoding guaranteed that each vector was orthogonal to every other vector (as in an experiment of Elman (1990)). Examples of this encoding is shown in Table 3.6.

Experiment and results

The model was trained on all the 21 sensorimotor-word mappings. Each sensorimotor-word pattern was added more than once to the training set so that the model was not generalizing

Sensorimotor item	Target word
0000000001000000000000 (BREAK)	0000000000000000100000 (break)
0000000000010000000000 (MAN)	0100000000000000000000 (man)
0000000000000000000001 (WOLF)	1000000000000000000000 (wolf)
0000010000000000000000 (CAT)	0010000000000000000000 (cat)
0000100000000000000000 (MOUSE)	0001000000000000000000 (mouse)
0000000000000000000100 (SEE)	0000000000000000001000 (see)
0100000000000000000000 (SLEEP)	0000000000001000000000 (sleep)
0000000000001000000000 (ROCK)	0000010000000000000000 (rock)
0000000000000000010000 (SEE)	0000000000000100000000 (see)
0000000000000000000010 (SMASH)	0000000000000000000100 (smash)
0000000000000000100000 (PLATE)	0000000001000000000000 (plate)
0000001000000000000000 (CHASE)	0000000000000010000000 (chase)
0000000100000000000000 (BOOK)	0000100000000000000000 (book)
0010000000000000000000 (GLASS)	0000000010000000000000 (glass)
0000000000000100000000 (MOVE)	0000000000000001000000 (move)
0000000000000010000000 (EAT)	0000000000000000000010 (eat)
0000000000000001000000 (THINK)	0000000000010000000000 (think)
0000000000100000000000 (SMELL)	0000000000000000010000 (smell)
0000000000000000001000 (DRAGON)	0000001000000000000000 (dragon)
0000000010000000000000 (COOKIE)	0000000000100000000000 (cookie)

Table 3.6: Training data for word-sequencing model

on the random order of the training set. To prevent the model from getting a bias to certain words each word had the same frequency in the training set. The training set contained a total of 200 sensorimotor-word patterns. The model had 22 output units, 30 hidden units, 30 context units and 44 input units (22 for the current sensorimotor item, 22 for the previous word). We trained the network over 1000 epochs. We repeated the experiment over 100 times. In Figure the learn curve average over 100 runs can be seen in Figure 3.12. The final average MSE was 1.049 x 10_{-4}. This seems really low, but an error of 0.18[1] means that it has chosen for the wrong word. So we should be carefull by making to optimistic conclusions. Therefore, we used the model to predict words which were seen but in an unseen order. The test set was a sequence of 5000 items. Every run after training the network was tested on the test set. On average, the trained model predicted 4999.51 items correct with a standard deviation of 1.25.

These results are important because this part of the model influences the results on the other experiments. This trained model produces the pseudo patterns. If the words are learned incorrect the pseudo patterns will be wrong. As result, the prediction on the final test sentences will be wrong as well.

3.3.2 Predicting sentences

Training data

The training data for this experiment is partly the same as in the previous experiment. The words and sensorimotor items are represented again as a randomly assigned bit vector of length 22. The difference with the previous experiment is that *gap* is also represented as a bit vector. The vector also has only one bit flipped on, so it is orthogonal to the other words. This representation is used as new training data when the model is unable to predict the right word.

[1]If the model chooses the wrong item using the winner-takes-all function the difference is 2. Taking the square of this and dividing it by the number of output units, 22, we have a mean squared error of 0.18

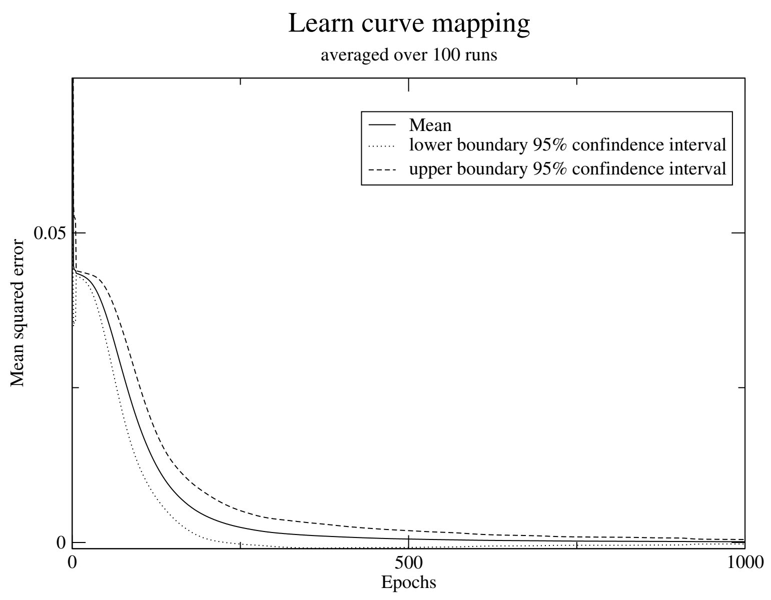


Figure 3.12: The learn curve of word mapping, averaged over 100 runs

Sensorimotor sequence	THE MAN	THE CUP	THE MAN	GRABBED	THE CUP
Word sequence	the man	gap	gap	grabbed	the cup

Table 3.7: An example of a target mapping in SVO order between the sensorimotor sequence and the word sequence

Another example, Table 3.8, shows the sequence in OSV order.

Sensorimotor sequence	THE MAN	THE CUP	THE MAN	GRABBED	THE CUP
Word sequence	gap	the cup	the man	grabbed	gap

Table 3.8: An example of a target mapping in OSV order between the sensorimotor sequence and the word sequence

Experiment and results

As in the first experiment the model is trained on sensorimotor-word mappings. After that the model is used to create its own patterns with gaps. A complete description of the this procedure is given in Section 2.5.3. The experiment was run 100 times. The mapping phase had 1500 epochs and the sentence learning had 4000 epochs. As in the first experiment the model had 22 output units, 30 hidden units, 30 context units and 44 input units. We have run the experiments for the SVO, SOV and OSV order. The results are given in Table 3.9.

	SOV	SVO	OSV
Average MSE	2.298 x 10^{-2}	1.27 x 10^{-2}	2.91 x 10^{-2}
Standard deviation MSE	1.080 x 10^{-2}	7.57 x 10^{-3}	9.25 x 10^{-3}
Average sentences predicted correct	14.67	23.97	6.78
Standard deviation predictions	8.39	8.71	4.69
Total sentences	44	44	44
Words predicted correct	164.38	189.27	149.47
Standard deviation word predictions	26.14	18.32	22.39
Total words	220	220	220
Quality pseudo patterns	99.01%	99.02%	99.99%

Table 3.9: The results for predicting sentences for the orders SOV, SVO, OSV averaged over 100 runs

The table shows the performance of the model averaged over 100 runs. It seems the model roughly works. Individual words are predicted with an accuracy of 68% for OSV, 75% for SOV to 86% for SVO. Complete sentences[2] are predicted with an accuracy of 15% for OSV, 33% for SOV and 50% for SVO. These results might not be convincing enough without showing what the likelihood is of getting words and sentences correct by chance. The chance to predict a word correct is given in Equation 3.1. N_w is the number of words per sentence. N_s is the number of items in a sentence. N_g is the number of gaps in a sentence and f_w is the number of possible words. P_w is the probability of predicting a correct word.

$$P_w = \frac{N_w\left(\frac{N_w}{N_s}\right)\left(\frac{1}{f_w}\right) + N_g\left(\frac{N_g}{N_s}\right)}{N_s} \tag{3.1}$$

When we compute the chance for our problem with $N_w = 3$, $N_s = 5$, $f_w = 21$ and $N_g = 2$ we get a probability of 1.77 x 10^{-1}. This would mean that on average a random model would

[2]A complete sentence means that all the words are correct

predict 38.97 words from a set of 220 words. This means that our model is definitely not random for all the orderings. Equation 3.2 gives the formula for the chance to predict a complete sentence. P_s is the probability of predicting a correct sentence.

$$P_s = \left[\left(\frac{N_w}{N_s} \right) \left(\frac{1}{f_w} \right) \right]^{N_w} \left[\frac{N_g}{N_s} \right]^{N_g} \tag{3.2}$$

When we use the previous result for this formula we get a probability of 3.73 x 10^{-6} for P_s. This means that a random model on average over 44 sentences can predict 8.21 x 10^{-4} sentences. Therefore, our model is performing considerably better than chance, even for the worst case of OSV.

Now we know that the results are pretty good it is interesting to speculate why there is a difference between the different orders. The table shows that the OSV order seems to be more difficult than the two other orderings, SVO and SOV. This error could have occurred in the first phase while creating the new training patterns. Therefore, we have looked at the quality of the created training patterns. These results are also shown in the Table 3.9. The table shows that the quality of for the OSV order is even better than for the other two orders. This is thus not the cause of the worse performance for OSV.

Now we have seen that the problem is far from randomly solveable and wrongly created training data is not the cause, it can be concluded that the cause must lie in learning the OSV order itself. This is interesting because there are in fact very few OSV languages - these are by far the rares group of languages. The poor performance of the model on OSV languages might therefore be taken as positive evidence for the model (See Lupyan & Christiansen, 2002, for a similar argument).

3.3.3 Conclusions

The experiments show that the word-sequencing model is able to learn the mapping between sensorimotor items and words. Although the results are not optimal. A more optimal learning rule than our basic back-propagation without momentum would probably be more powerful. Which would lead to better results. This will be left as future work. The experiments have shown that the model is able to learn the problem pretty well if you compare it to a random model. More interesting is that the model is performing worse on OSV than on the other two orders. This is interesting because Lupyan & Christiansen (2002) confirm this result while the representation of the language problem in our experiment differs from their representation of the language order problem. However, the model is not really biologically plausible at this moment and the results are not totally convincing. Therefore, future work will have to show that it is significant.

Chapter 4

Conclusion and Future Work

All the work done in this project should be seen in the context of Knott's (2005) programme to build a model of sensorimotor cognition and language, which solves the L0 task (Feldman et al., 1996). The full model takes into account the whole process from perception to action execution and generating the corresponding sentences that describe the events that took place.

In this project we implemented two parts of Knott's model. The first part is a model of action selection and memory for events, using a decision function to make decisions on the next object to attend to, or motor action to execute, and an episodic memory, to store the events that happened in the past. With this model an observer can experience the world and learn in what sequence to attend to objects or execute actions, in order to reach states which are beneficial to the observer. The second part of Knott's model that we implemented is a word sequencing network, which takes as input an event as a trace of the episodic memory and outputs sentences in different word orderings, suitable different types of languages.

The sensorimotor model and the word sequencing model have been designed with the idea to combine these two models. These two models combined can be used to evaluate the hypothesis that natural language has emerged from prelinguistic sensorimotor capacities. This hypothesis can be tested by evaluating the model against claims by both linguists and cognitive scientists.

In the current project we have unfortunately been unable to evaluate the combination of the two models. It is probably a bit too early to connect the two models in order to make claims about this hypothesis. Further work on the models can improve the results significantly to reach the point where a combination is actually possible. The sensorimotor model must be capable of remembering more events in episodic memory. The decision function in the sensorimotor function should be able to generalize over the problem every single time it encounters a reward state. The variability in training performance would become an issue when the two models are combined. The word sequencing model is only able to produce correct sentences in at most 50% of the time. This is a problem, since human speakers make hardly any errors in word ordering in sentences in their own language. In order to explain a linguistic finding like word and verb movement as an effect of rewind mode in the sensorimotor model, both models should perform as well as their human counterparts. Another hypothesis we would have liked to evaluate is that word mapping in the word sequencing model can be achieved by using the random sensorimotor output in the early stages of experiencing in the world.

Future work

Although both the sensorimotor model experiments and the word sequencing model experiments show us that the problems we discussed are learnable, there are still quite a few improvements possible. First we propose solutions to the problems with the sensorimotor model, especially the episodic memory and the decision function; next we discuss the improvements that can be made on the word sequencing model.

In order for experience mode to work better, the sensorimotor model could be extended with an extra 'long term memory', which stores context of sequences generated by the 'short term' Elman Network. Added to that, the decision function should not be unstable when it comes to learning the reward sequences. Future work on this problem might include extending the Elman Network with multiple hidden layers and context layers, increasing the context layer size and probably spend some extra time on finding the optimal parameters for learning the decision function.

The word sequencing model is now only capable of using the sentences ordering of SVO, SOV and OSV. To make a more plausible model it is necessary to model the other orderings, OVS, VOS VSO, as well. This can be done by using the real sensorimotor sequence which contains seven items instead of five which we used in the word-sequencing model experiments. Since the sensorimotor sequences gives information whether an item is an action or an object another improvement would be for the word sequencing model to use this information so it can make better predictions. When this is achieved it might be possible to present sequences that do not have a fixed length of five.

Finally, the two models should be combined. Both models should be trained on-the-fly. This means that both models improve over time. When both models are combined the next challenge will be to predict sensorimotor sequences from word sequences.

Appendix A

Training patterns for the decision function

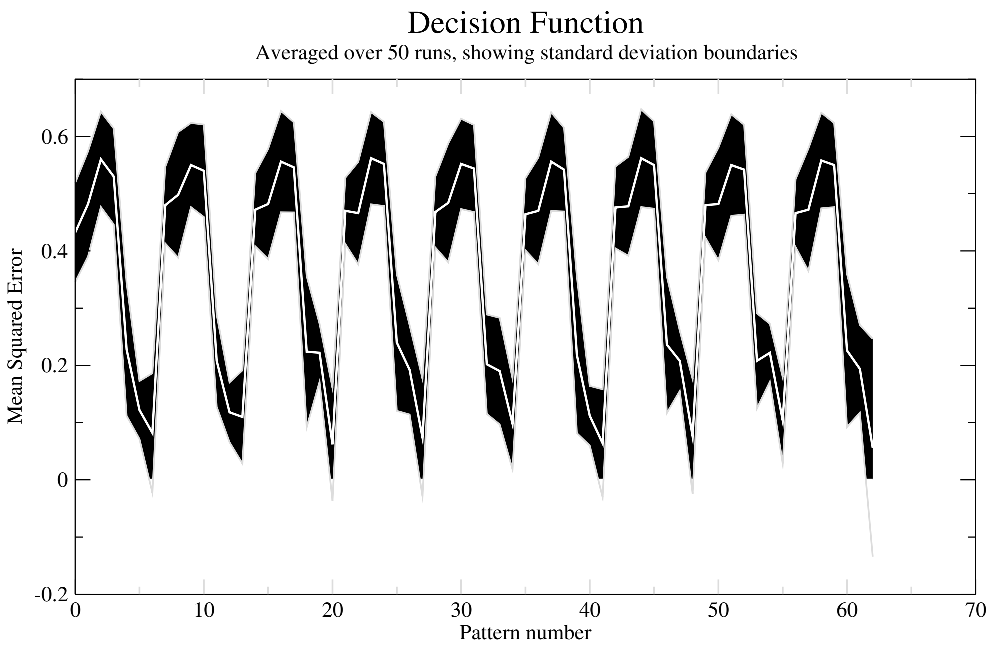


Figure A.1: Decision function learning error, coloured area is the standard deviation upper and lower boundary from the average, which is shown by the white line

Bibliography

Abney, S. (1987), *The English noun phrase in its sentential aspect*, MIT, Cambridge, MA. unpublished MIT Ph. D.

Ans, B. (2004), 'Sequential learning in distributed neural networks without catastrophic forgetting: A single and realistic self-refreshing memory can do it', *Neural Information Processing Letters and Reviews* **4**, 27–32.

Ans, B. & Rousset, S. (2000), 'Subscribed content neural networks with a self-refreshing memory: knowledge transfer in sequential learning tasks without catastrophic forgetting', *Connection Science* **12**(1), 1–19.

Ans, B., Rousset, S., French, R. M. & Musca, S. (2002), Preventing catastrophic interference in multiple-sequence learning using coupled reverberating elman networks, *in* 'Proceedings of the 24th Annual Conference of the Cognitive Science Society', number NJ:LEA.

Baker, C. L. (1979), 'Syntactic theory and the projection problem', *Linguistic Inguiry* **10**(4), 533–581.

Bakker, B. & van der Voort van der Kleij, G. (2000), Trading off perception with internal state: Reinforcement learning and analysis of q-elman networks in a markovian task, *in* C. L. Amari, M. Giles & G. V. Piuri, eds, 'Proceedings of the International Joint Conference on Neural Networks', Vol. 3, pp. 213–218.

Bakker, P. B. (2004), The State of Mind Reinforcement Learning with Recurrent Neural Networks, PhD thesis, Universiteit van Amsterdam, The Netherlands.

Ballard, D. (1991), 'Animate vision', *Artificial Intelligence.*

Ballard, D., Hayhoe, M., Pook, P. & Rao, R. (1997), 'Deictic codes for the embodiment of cognition', *Behavioral and Brain Sciences* **20**(4), 723–767.

Bergen, B. K. & Chang, N. (2003), 'Embodied construction grammar in simulation-based language understanding', in press J.-O. Ostman and M. Fried (eds.), Construction Grammar(s): Cognitive and Cross-Language Dimensions. (updated 6/2003).

Bruske, J., Ahrns, I. & Sommer, G. (96), Practicing q-learning, *in* 'ESSAN', Proceedings of the 1996 Conference, Bruges, pp. 25–30.

Chang, F. (2002), 'Symbolically speaking: a connectionist model of sentence production', *Cognitive Science* **26**(5), 609–651.

Chomsky, N. (1981), *Lectures on government and binding*, Foris publications, Dordrecht.

Chomsky, N. (1995), *The minimalist program*, Mit Press, Cambridga, MA.

Corballis, M. (2002), *From hand to mouth: the origins of language*, Princeton University Press, Princeton.

Dominey, P., Hoen, M., Blanc, J. & Lelekov-Boissard, T. (2003), 'Neurological basis of language and sequential cognition: Evidence from simulation, aphasia and erp studies', *Brain and Language* **86**, 207–225.

Elman, J. (1990), 'Finding structure in time', *Cognitive Science* **14**, 179–211.

Fagg, A. & Arbib, M. (1998), 'Modeling parietal-premotor interactions in primate control of grasping', *Neural Networks* **11**(7/8), 1277–1303.

Feldman, J., Lakoff, G., Bailey, D., Narayanan, S., Regier, T. & Stolcke, A. (1996), 'L_0: The first five years of an automated language acquisition project', *Artificial Intelligence Review* **10**(1–2), 103–129.

Fodor, J. A. & Pylyshyn, Z. W. (1988), 'Connectionism and cognitive archictecture - a critical analysis', *Cognition* **28**(1-2), 3–71.

French, R. (1999), 'Catastrophic forgetting in connectionist networks', *Trends in Cognitive Sciences* **3**(4), 129–135.

Giese, M. (2000), Neural model for the recognition of biological motion, *in* G. Baratoff & H. Neumann, eds, 'Dynamische Perzeption', Infix Verlag, Berlin, pp. 105–110.

Givón, T. (2002), The visual information-processing system as an evolutionary precursor of human language, *in* T. Givón & B. Malle, eds, 'The evolution of language out of prelanguage', John Benjamins, Amsterdam.

Goldberg, A., ed. (1995), *Constructions. A Construction Grammar approach to argument structure*, University of Chicago Press, Chicago.

Greenberg, J. (1963), *Universals of Language*, 2nd edn, The MIT Press, chapter Some universals of grammar with particular reference to the order of meaningful elements, pp. 73–113.

Gropen, J. (1989), 'The learnability and acquisition of the dative alternation in english', *Language* **65**(2), 203–257.

Hurford, J. (2003), 'The neural basis of predicate-argument structure', *Behavioral and Brain Sciences* **26**(3), 261–283.

Itti, L. & Koch, C. (n.d.), 'Computational modelling of visual attention.', *Nature Reviews—Neuroscience* **2**, 1–11.

Jackendoff, R. (1977), 'X-bar syntax: A study of phrase structure', *Linguistic Inguiry Monograph 2*.

Jeannerod, M. (1996), *Handbook of perception and action. Vol. 2: motor skills*, Vol. 2, Academic Press, London, chapter Reaching and grasping. Parallel specification of visuomotor channels, pp. 405–460.

Johansson, G. (1973), ‘Visual perception of biological motion, and a model for its analysis’, *Visual Perception and Psychophysics.*

Knott, A. (2003*a*), ‘Do sensorimotor processes have reflexes in sentence syntax as well as sentence semantics?’, *Behavioral and Brain Sciences.*

Knott, A. (2003*b*), Grounding syntactic representations in an architecture for sensorimotor control, Technical report, OUCS.

Knott, A. (2005), Sensorimotor cognition and natural language syntax, Unpublished.

Koopman, H. & Sportiche, D. (1991), ‘The position of subjects’, *Lingua* **85**(2-3), 211–258.

Lakoff, G. (1970), ‘Global rules’, *Language* **46**, 627–639.

Lupyan, G. & Christiansen, M. H. (2002), Case, word order and laguage learnability: Insights from connectionist modeling., *in* ‘Proceedings of the 24th Annual Conference of the Cognitive Science Society’, pp. 596–601.

Marcus, G. F. (1998), ‘Rethinking eliminative connectionism’, *Cognitive psychology* **37**(3), 243–282.

Milner, R. & Goodale, M. (1995), ‘The visual brain in action’, *Oxford University Press.*

Oram, M. & Perrett, D. (1996), ‘Integration of form and motion in the anterior superior temporal polysensory area (STPa) of the Macaque monkey’, *Journal of Neurophysiology* **76**(1), 109–129.

Pollard, C. & Sag, I. (1994), *Head-Driven Phrase Structure Grammar*, University of Chicago Press.

Pollock, J. Y. (1989), ‘Verb movement, universal grammar, and the structure of ip’, *Linguistic Inguiry* **20**(3), 365–424.

Radford, A. (1997), *Syntactic theory and the structure of English - A minimalist approach*, Cambridge University Press.

Riesenhuber, M. & Poggio, T. (1999), ‘Hierarchical models of object recognition in cortex’, *Nature Neuroscience.*

Rizzolatti, G. & Arbib, M. (1998), ‘Language within our grasp’, *Trends in Neurosciences* **21**, 188–194.

Rizzolatti, G., Fogassi, L. & Gallese, V. (2000), Cortical mechanisms subserving object grasping and action recognition: a new view on the cortical motor functions, *in* M. Gazzaniga, ed., ‘The new cognitive neurosciences’, MIT Press, pp. 539–552.

Robins, A. (1995), ‘Catastrophic forgetting, reheursal and pseudorehearsal’, *Connection Science* **7**, 301–329.

Robins, A. (1997), Mainaining stability during new learning in neural networks, *in* ‘Proceedings of the IEEE International Conference on Systems, Man and Cybernetics’, IEEE Society Press, Los Alamos, pp. 3013–3018.

Sag, I. A. & Fodor, J. D. (1994), Extraction without traces, *in* 'Proceedings of West Coast Conference on Formal Linguistics', Vol. 13, Stanford University: CSLI Pulbications, pp. 365–384.

Siskind, J. (1995), 'Grounding language in perception', *Artificial Intelligence Review* **8**, 371–391.

Thornton, I., Cavanagh, P. & Labianca, A. (2000), 'The role of attention in the processing of biological motion', *Perception* **29**(Suppl), 114.

Tipper, S., Lortie, C. & Bylis, G. (1992), 'Selective reaching: Evidence for action-centred attention', *Journal of Experimental Psychology: Human Perception and Performance* **18**(4), 891–905.

Treisman, A. & Gelade, G. (1980), 'A feature integration theory of attention', *Cognitive Psychology* **12**(1), 97–136.

Watkins, C. J. C. H. (1989), Learning from Delayed Rewards, PhD thesis, University of University of Cambridge, England.