Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

A CONNECTIONIST MODEL OF SENTENCE GENERATION IN A FIRST AND SECOND LANGUAGE

Michael Edward Gasser

July 1988
CSD-880050

UNIVERSITY OF CALIFORNIA

Los Angeles

A Connectionist Model of Sentence Generation in a First and Second Language

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Applied Linguistics

by

Michael Edward Gasser

1988

To

Martha

Mayumi

and Naomi

Table of Contents

ACKNOWLEDGMENTS

I have been lucky to find co-advisors who share the willingness to see beyond the traditional boundaries that divide disciplines, the sort of willingness that makes Cognitive Science possible. I owe lots of things to Prof. Michael Dyer, not the least of them a new career. In the true spirit of cognitive science, Michael saw fit to allow outsiders like myself into his group. The result was an extra four years to finish my Ph.D, but it was definitely worth it. The last part of this process involved having every word in this thesis scrutinized; Prof. Dyer really reads a dissertation, and as painful as this may be, you learn a lot in the process.

Prof. Evelyn Hatch has spent much of her professional life challenging applied linguists to look at new methodologies and areas of research, including conversational analysis, language simplification, path analysis, neurolinguistics, processing models, and scripts. In agreeing to go along with me on a project involving an AI approach to second language acquisition, she again demonstrated her recognition of what is to be gained by borrowing from other fields. If second language acquisition research is to join in the "cognitive revolution", it will be because of people like Prof. Hatch.

I'd also like to thank the other members of my committee for their time, Prof. Edward Keenan, Prof. Dean Richards, and Prof. George Bedell.

My thesis research was conducted in the UCLA AI Laboratory, the brainchild of Professors Michael Dyer and Margot Flowers. The lab is a place where people have the audacity to work on language, memory, reasoning, creativity, vision, daydreaming, and learning, often at the same time. Seriously, the variety is probably unequalled in any AI group anywhere.

A lot of the progress I made in my work came through discussion with other lab members, in particular Uri Zernik (now at GE) and Ric Feifer. Seth Goldman and his T Flavors Graphics package had a lot to do with the appearance of demo versions of my program. Other lab members include Sergio Alvarado (now at UC Davis), Erik Mueller (now at Morgan Stanley), Stephanie August, Charlie Dolan, Maria Fuenmayor, Jack Hodges, Valeriy Nenov, Mike Pazzani, Alex Quilici, Walter Read, John Reeves, Ron Sumida, and Scott Turner.

My research was supported by grants from the Initial Teaching Alphabet Foundation and the Joint Tactical Fusion program of the Department of Defense. I am especially grateful to Betty Thompson of the ITA Foundation for recognizing the role that her small organization could play in basic research in AI, as well as in the careers of a few graduate students. Equipment in the lab is also paid for in part by grants from the Keck Foundation.

This thesis is dedicated to my mother, Martha Dickman, who has always thought I knew what was best for me, even when I clearly didn't; to my wife, Mayumi Koide, who has supported me in all sorts of ways and in addition endured the humiliation of having her English mistakes pointed out to hundreds of people she never met; and to my daughter, Naomi Koide-Gasser, who has done her best to lighten up the last 14 1/2 months of this process.

VITA

July 18, 1948	Born, Camp Lejeune, North Carolina
1969	B.A., Mathematics, San Diego State College
1969-1973	Peace Corps, Ethiopia
1976-1986	ESL Instructor, American Language Center, UCLA Extension
1979	M.A., Teaching English as a Second Language, University of California, Los Angeles
1985-1988	Research Assistant, Artificial Intelligence Laboratory, Computer Science Department, University of California, Los Angeles
Spring 1988	Visiting Lecturer, Applied Linguistics Department, University of California, Los Angeles

PUBLICATIONS

Gasser, M., & Dyer, M. G. (1988). Sequencing in a connectionist model of language processing. *Proceedings of the Twelfth Conference on Computational Linguistics.*

Gasser, M. (1987). Memory organization in the bilingual/second language learner: a computational approach. *Proceedings of the Eastern States Conference on Linguistics 1986.*

Gasser, M. (1987). Dynamic lexical memory. *Thirteenth LACUS Forum*, 421-431.

Gasser, M., & Dyer, M. G. (1986). *Speak of the devil:* Representing deictic and speech act knowledge in an integrated lexical memory. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 388-398.

Gasser, M. (1985). Amharic *-m* and *-ss* : Morphology, theme, and assumed knowledge. *Lingua, 65*, 51-106.

Gasser, M. (1985). *Second language production: Coping with gaps in linguistic knowledge* (Technical Report UCLA-AI-85-18). Los Angeles: University of California, Los Angeles, Computer Science Department.

Gasser, M. (1983). Topic continuity in written Amharic narrative. In T. Givón (Ed.), *Topic continuity in discourse: A quantitative cross-language study* (pp. 99-139). Amsterdam: John Benjamins.

Rossi, L. D., & Gasser, M. (1983). *Academic English.* Englewood-Cliffs, NJ: Prentice-Hall.

ABSTRACT OF THE DISSERTATION

A Connectionist Model of Sentence Generation

in a First and Second Language

by

Michael Edward Gasser

Doctor of Philosophy in Applied Linguistics

University of California, Los Angeles, 1988

Professor Evelyn M. Hatch, Co-Chair

Professor Michael G. Dyer, Co-Chair

This thesis presents the Connectionist Lexical Memory model, a psychologically plausible computational approach to sentence generation. The system's knowledge is contained entirely in a network of simple processing units and weighted connections. All linguistic knowledge is in the form of subnetworks representing schemas; there are no rules as such. Processing consists entirely in the parallel spread of activation over the network; there is no top-level control structure guiding it, and there are no processes specific to language. The thesis demonstrates not only that it is possible to accomplish generation within this constrained framework but also that the approach offers significant advantages over conventional models in modelling features of human generation. In particular, the model exhibits priming effects, speech errors, robustness with respect to incomplete input and gaps in linguistic knowledge, flexibility in sequencing, and cross-linguistic transfer effects. In addition, the same memory and processing mechanism are usable in comprehension.

The input to generation is a set of activated network nodes representing the speaker's goal and the output a set of activated nodes representing words. The three major aspects of the generation process are the selection of linguistic schemas in memory which match input concepts, the temporary association of syntactic roles in selected schemas with semantic roles in the input, and the sequencing of words and constituents being output. All are implemented through spreading activation.

Role binding and sequencing are usually difficult for connectionist models to handle. In the present approach, role binding is represented by the simultaneous firing of the nodes being "bound". A path of primed nodes connecting the two bound nodes remains to make the association temporarily accessible. Sequencing is handled using explicit sequencing connections, special nodes to represent the ends of sequences, and mutual inhibition among nodes to implement competition for a particular output position.

For multilingual contexts it is shown that there are at least four different types of associations between linguistic units for different languages. When a speaker does not have an appropriate unit in the second language, the existence of a matching first language schema may bias the selection of a particular second-language schema.

Chapter 1

Overview

1.0 Introduction

In this dissertation I examine the problem of language generation. I present a theory of conceptual and linguistic memory and processing called the Connectionist Lexical Memory (CLM) model. The theory is implemented in a computer program called CHIE which generates English and Japanese sentences, exhibiting features of human language generation including priming, speech errors, robustness with respect to incomplete input and gaps in linguistic knowledge, flexibility in sequencing, speech errors, and cross-linguistic transfer effects.

The CLM model offers an approach to generation radically different from existing models. The system's knowledge is contained entirely in a network of simple processing units. All linguistic knowledge is in the form of declarative form-function associations; there are no rules as such. Processing consists entirely in the parallel spread of activation over this network; there is no top-level control structure guiding it. The dissertation demonstrates not only that it is possible to accomplish generation within this highly constrained framework but also that the approach offers significant advantages over conventional models in modelling features of human generation.

Input to the generation process consists of a set of network nodes representing features of the speaker's goal. These nodes "fire", sending activation to other nodes. Nodes which receive enough activation, in particular those where activation converges from two or more sources, fire and propagate activation further through the network. A sentence is uttered as nodes representing words fire in an appropriate sequence.

The major process implemented by spreading activation is the selection of one or more subnetworks which characterize the input. During generation subnetworks representing linguistic patterns are selected. Patterns compete with one another as ways to refer to a given input concept. A pattern is selected when enough activation converges on it for its head node to fire. Pattern selection can be viewed as a process of simultaneous constraint satisfaction implemented with spreading activation and mutual inhibition. The process is robust in that some pattern is normally selected even where there is none that matches the set of input features precisely. Errors in pattern selection may result when only a subset of the input features are activated and there is a pattern other than the correct one which matches these features. Such errors resemble those that people make.

Two general problems for other parallel models making use of spreading activation are the temporary binding of variables and sequential behavior. In the CLM model, there are no variables, and there is no way to actually bind one node to another temporarily. What corresponds to binding instead is the more or less simultaneous firing of the nodes to be associated. Sequencing is handled through the use of explicit sequence connections between nodes representing constituents and competition between constituent nodes for particular output positions. This approach not only forces sequential behavior on the parallel process, but it allows sequencing to be flexible because the relative ordering of two constituents may vary with the amount of activation they receive from other nodes.

While the focus of this study is on generation, the network has been designed so that the knowledge can be used in parsing the same sorts of the sentences that the system generates. Parsing is implemented with exactly the same spreading activation mechanism used for generation. The only difference is that the input to parsing consists of firing nodes representing words rather than conceptual features.

1.1 Requirements for a Generation System

Language generation is the process which takes a speaker (or writer) from some conceptual input to an utterance or set of utterances. In this section I look at what general types of knowledge a generation system needs in order to carry out this process and what sorts of features characterize human generation.

1.1.1 Representation of Conceptual and Linguistic Knowledge

Consider what kinds of knowledge are involved in the generation of an ordinary sentence such as (1.1).

(1.1) *Mary deposited $100 in the bank.*

First, there needs to be some way of representing the input to the utterance process, in this case the notion of Mary's depositing some money in the bank. This notion must be represented in terms of the general concept of depositing money, which should in turn provide access to the verb *deposit*.[1] The system also needs knowledge about this verb, in particular knowledge that tells the speaker to refer to particular elements of the input in particular slots in the sentence. For this example, the speaker must know that Mary is to be referred to in the subject position and the deposited money in the direct object position. Thus the generation process requires three kinds of knowledge, knowledge about concepts, such as DEPOSIT-MONEY-IN-BANK; knowledge about linguistic patterns, such as the one associated with the verb *deposit*; and knowledge about associations between concepts and patterns, such as the fact that the actor of the action is referred to in the subject of the sentence.

For generation, the most important concepts to be represented are those directly associated with linguistic forms. These include not only concepts which are linked to lexical items such as *deposit* but also concepts which allow the speaker to select a general pattern for an utterance. Consider sentence (1.2), for example.

(1.2) *Could you deposit this money for me?*

Here the speaker selects the *could you* pattern, one of a set of request forms. Speakers perform requests as a means of getting someone else (the hearer) to perform an act on their behalf. In general, morphosyntactic patterns such as the *could you* form are associated with types of speaker goals.

A generation system, then, must make reference to concepts associated both with general patterns and with lexical items. These concepts and the inputs to the generation process need to be represented in terms of a set of common conceptual features. Since the inputs are non-linguistic, these features must also be non-linguistic.

Linguistic patterns specify the syntax of the utterances that are generated and fill in the slots in the utterances with words or grammatical morphemes. Patterns need to include information about constituency, that is, how the positions in larger patterns are filled by smaller patterns of particular types, and about sequencing, that is, how the constituents of a pattern are ordered. For example, the *could you* pattern has positions for a MODAL-AUXILIARY, filled by the word *could*, and a SUBJECT, filled by the word *you*, and the MODAL-AUXILIARY is specified as preceding the SUBJECT.

---

[1] In this thesis I will use *italics* for linguistic forms, SMALL CAPITALS for concepts or nodes in the network memory, boldface for new terms, and underlining for emphasis.

The concept-pattern associations that are needed can be divided into two types, those joining types of speaker goals with general patterns, such as the *could you* pattern, and those joining concepts with relatively specific patterns, such as the pattern with *money* in its noun position or the one with *deposit* in its verb position. Within these basic associations there need to be connections between parts of the concept and parts of the pattern. For example, in order to generate sentence (1.1), the *deposit* pattern needs to contain information associating the SUBJECT position in the pattern with the DEPOSITOR role in the concept referred to.

1.1.2 Accommodating Both Top-Down and Bottom-Up Processing

In sentence (1.2) we saw the role of the speaker's goals and plans. However, utterances also arise in response to the speaker's perception of contextual events. Consider sentence (1.3). The speaker has been talking about Mary when unexpectedly she arrives on the scene.

(1.3) *Speak of the devil.*

In sentences such as this one, the context-driven (bottom-up) aspect of generation predominates; the speaker is more concerned with what is appropriate to say under the circumstances than with satisfying a goal. Knowledge in the system needs to be organized in such a way that the speaker's perception of the arrival of the person being discussed causes the expression to occur to her[2].

1.1.3 Robustness: Generation in the Face of Deficient Linguistic Knowledge

Sometimes the speaker does not find a word or pattern which matches the concept she wants to convey. Consider sentence (1.4), a made-up example. The speaker wants to leave her jewelry in a bank for safekeeping but has never done this before and does not know how to refer to the process.

(1.4) *I'd like to deposit this jewelry.*

She selects the verb *deposit*, though for her this word has the sense of transferring money to a bank. However, she is willing to relax this constraint in the hopes that the word will be appropriate in this slightly different situation. It turns out that the word does apply in this case. In another case it might not be as appropriate. For example, if the speaker wanted to leave some money for safekeeping with a friend rather than putting it in a bank, she might also choose *deposit*, but here it would sound odd. The point is that a speaker can often find a word for an input notion even though the meaning of the word that she has stored doesn't precisely match that notion. Thus human generation is robust: it does not necessarily break down in cases like this.

1.1.4 Flexibility: Multiple Representations of Concepts

The same input or similar input may result in very different structures. This phenomenon is especially clear when we look at translation equivalents across languages. Sentences (1.5a) and (1.5b), for example, mean more or less the same thing, yet their structure is quite different. In the Japanese version the concepts of CAUSE and PREPARE-FOOD are combined in a single verb which takes three arguments, expressing the CAUSER, the ACTOR, and the OBJECT of the cooking.

---

[2]For convenience, I will adopt the convention, as Clark (1979) does, of referring to speakers with feminine pronouns and hearers with masculine pronouns.

(1.5a) *Mary had John cook dinner.*

(1.5b) *Mary wa John ni yuuhan o tukur-ase-ta.*
Mary TOPIC John AGENT dinner ACCUS make-cause-PAST

Examples like these show that generation needs to be flexible enough to map a single input concept onto more than one linguistic structure.

1.1.5 Parallelism

At the output level generation is obviously a sequential process, but it is apparently not so at deeper levels. Speech errors involving exchanges of elements (Dell, 1986) provide evidence for one kind of parallelism in generation. Sentence (1.6) is an example of a switching error.

(1.6) *I wrote a mother to my letter.*

In sentences like (1.6) the speaker seems to be working on two pieces of the utterance at once when one of the pieces somehow replaces the other. Thus generation involves some parallelism in the formulation of the constituents of clauses.

Another type of parallelism comes in the selection of linguistic patterns on the basis of conceptual features. The number of features involved increases with the specificity of the concept. If these features were checked sequentially, we would expect more time to be required for finding lexical items for specific concepts than for more general concepts. For example, it would take longer to select the word *water* than the word *liquid*. It is in fact the reverse ordering that occurs; basic-level terms such as *water* are accessed more readily than their superordinates (Clark & Clark, 1977, p. 470). Thus features must be checked in parallel.

1.1.6 Competition Between Linguistic Forms

In selecting the linguistic form which best conveys an input concept, we can expect a speaker to begin with a set of candidate forms which can be viewed as competing with one another. Speech errors involving substitutions (Dell, 1986) provide evidence for this competition. Sentence (1.7) is an example of a substitution error. The speaker intends for the hearer to pass the pepper.

(1.7) *Please pass the salt.*

Here the words *salt* and *pepper*, or their meanings, are competing as ways of characterizing the input concept. There is strong pressure to select only one of a competing set of items; for example, we do not find both *salt* and *pepper* used to refer to the same object.

A different sort of competition is illustrated in sentences (1.8), in which there is variation in the relative position of the direct and indirect objects.

(1.8a) *Mary sent John a letter.*
Mary sent a letter to John.

(1.8b) *Mary didn't phone John. She sent him a letter.*
?Mary didn't phone John. She sent a letter to him.

(1.8c) *Mary didn't throw the letter away. She sent it to John.*
?Mary didn't throw the letter away. She sent John it.

These sentences can be seen as reflecting competition between two constituents, the direct and indirect objects, for the position following the verb.

1.1.7 Priming Effects

In choosing among candidate items to convey a particular input concept or to fill a particular output position, speakers are influenced by previous processing. For example, the error in (1.7) may have resulted from priming on the word *salt* or its meaning, either because the speaker had been thinking about salt or because someone else had been talking about salt.

1.1.8 Cross-Linguistic Transfer Effects

When a speaker knows more than one language, knowledge of one language may influence or interfere with generation in another language. Sentences (1.9-12) illustrate this phenomenon.

In example (1.9), speaker A is a native speaker of Japanese. She is referring to the cold water faucet.

(1.9) A: *The faucet's broken.*
B: *Which one?*
A: *Water.*

Speaker A's choice of words appears to be related to the fact that Japanese has separate words for 'hot water' and 'cold, or default, water'. That is, for her the English word *water* is in some way associated with the 'cold water' meaning.

In sentence (1.10) the speaker, a native speaker of Japanese, is trying to describe a person's leaving some property with a friend for safekeeping.

(1.10) *He...deposit...his property his friend.*

Exhibiting a good deal of uncertainty, she chooses the word *deposit* to refer to this act. While the selection of *deposit* in this sentence is similar to that which a native speaker might make under similar circumstances (see example (1.4)), it may also be related to the fact that Japanese has a verb, *takusu*, with the general meaning 'leave for safekeeping', including the special case of 'deposit money in a bank'. That is, the speaker may feel that *deposit* can be generalized in the same way as *takusu*.

Example (1.11) is an Amharic[3] sentence spoken by a native speaker of Amharic who also knows English very well.

(1.11) *lela kɨfɨl wessede.*
other class he:took
'He took another class.'

The Amharic verb *wesede* is the appropriate translation of English *take* only in the sense of 'achieve control', as in *John took the book from Mary* (*John mes'hafun kɛMary wesede*). This verb is not appropriate for referring to the 'taking' of a class. Here, however, the speaker is apparently influenced by the fact that English uses the same verb for these two concepts. She seems to be translating the English verb *take* directly (and inappropriately) into Amharic.

For example (1.12) (from Uyekubo, 1972) the speaker and hearer are both Japanese-English bilinguals.

[3]Amharic is a Semitic language spoken in Ethiopia.

(1.12) *ano okanemoti wa ozyoosan o spoil sita*
that rich:person TOP daughter ACC spoil did
'That rich man spoiled his daughter.'

The sentence produced is ordinary Japanese except for the intrusion of the English verb *spoil*. The implication of examples like this is that bilingual speakers cannot simply "turn off" their knowledge of one language when they are generating in another language.

1.1.9 Common Knowledge for Generation and Analysis

As far as possible, the knowledge used in generation should also be usable in language analysis. Shared knowledge makes more efficient use of memory, and it also makes sense from an acquisition perspective. Since the units of knowledge used in generation got into the system originally through the process of parsing, they should be in a form which is accessible to both processes.

1.2 The Connectionist Lexical Memory Approach

1.2.1 Features of the Model

The Connectionist Lexical Memory (CLM) model is a psychologically plausible computational account of sentence generation. The model takes as its input a representation of a speaker's goal or the perception by the speaker of an event motivating an utterance and yields as output a sequence of words. The model is partially implemented in a demonstration computer program called CHIE, which generates English and Japanese sentences.

The main features of the model are the following:

A. All knowledge as a network
   1. Memory consists of a network of nodes joined by pairs of directed, weighted connections. The system's knowledge is embodied entirely in these connections.
   2. Concepts are represented as schemas consisting of subnetworks of the memory.

B. All processing as spread of activation
   1. Processing consists entirely in the parallel spread of activation through the memory network starting with sets of nodes representing inputs. The amount of activation spreading along a connection depends on the weight associated with the connection and may be either positive (excitatory) or negative (inhibitory).
   2. Competition, implemented through sets of mutually inhibiting nodes, is an important aspect of processing. Competition forces the selection of one of a set of candidate nodes and imposes sequential output behavior on the system.
   3. The activation on each node decays over time, representing short-term forgetting. Priming is modelled as activation remaining on nodes as a result of recent processing.
   4. Spreading activation and competition together implement a form of lenient pattern matching which makes generation robust.
   5. Flexibility derives from the fact that spreading activation automatically finds alternate ways of conveying particular concepts.

6. Processing is interactional rather than strongly modular. Pragmatic, semantic, and syntactic information may be involved simultaneously in the selection of units of linguistic knowledge.

C. Linguistic knowledge as associations mapping language to concepts

1. The basic units of linguistic knowledge are schematic subnetworks associating form directly with function. These form-function mappings comprise an inventory from which selections are made during processing.

2. With respect to form, the linguistic units are composed of surface-level patterns of varying degrees of generality. Relatively specific (lexical) patterns are given priority in processing.

3. Since linguistic knowledge consists of form-function associations and since connections come in pairs, one for each direction, the knowledge in linguistic memory is usable for analysis as well as generation.

4. There are connections between linguistic schemas for different languages. During generation a schema for the wrong language may be accessed and then translated directly into the target language as a result of the traversal of these cross-linguistic connections.

5. Another type of transfer may result when a speaker selects a partially matching target language schema on the basis of the existence of a matching schema in another language. Sentence (1.10) is an example of such a case. In general the activation of one concept may lead to the activation of other concepts which overlap with it. For examples like (1.10), the availability of a suitable lexical item in the wrong language predisposes the speaker to select a similar, but not completely suitable lexical item in the appropriate language.

1.2.2 Representation of Concepts

Concepts are represented as schemas, that is, structured chunks of memory. Each schema has an associated type and a set of roles, each with a particular value. For example, the BANK-DEPOSIT schema has as its type the general notion of TRANSFER-OF-CONTROL, and among its roles is an OBJECT with MONEY as its value and a RECIPIENT with BANK as its value. Inputs to generation are represented similarly. Thus the input to sentence (1.10) has the concept TRANSFER-OF-CONTROL as its type and BANK as the value of its RECIPIENT role.

The schema-based memory is implemented in the form of a network of nodes joined by weighted uni-directional connections. Separate nodes represent schema heads, roles, and values. Pairs of connections, one for each direction, represent the relationships between schemas and their types, heads and roles, roles and their values, and roles and their types, e.g., the OBJECT of BANK-DEPOSIT and the OBJECT of TRANSFER-OF-CONTROL. The network implementation has three advantages. First, it makes representations more flexible because roles are treated as concepts not fundamentally different from schema heads. Second, the degree of associativity between any two concepts (heads, roles, and values) can vary with differences in connection weight. Third, the network implementation permits the use of a general spreading activation mechanism for processing. This mechanism is described in the next section.

Figure 1.1 shows a portion of the network representing the concept of DEPOSIT-IN-BANK. Node names and schema boundaries are indicated for convenience only; these are not used by the processing mechanisms in the model.

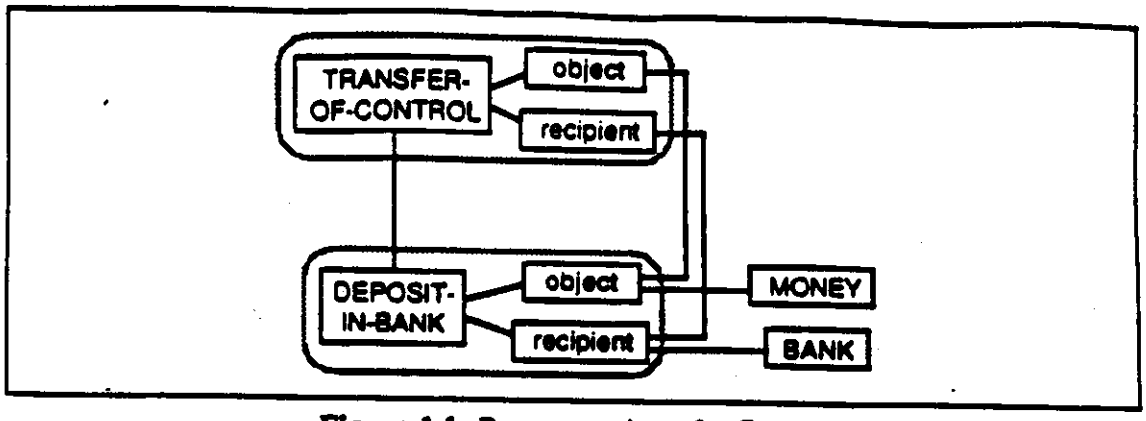

Figure 1.1: Representation of a Concept

In the figures, nodes representing schemas are surrounded by fuzzy rounded rectangles. Schema head nodes appear at the left side of schemas. Role nodes are labeled in lower-case. Only two roles are shown for these schemas; in general figures will not display all of the roles in schemas. Role names are meant to be abbreviations; the two roles labeled OBJECT in the figure represent the OBJECT of TRANSFER-OF-CONTROL and the OBJECT of DEPOSIT-IN-BANK. I will often refer to roles using the head and role names separated by a colon, e.g., DEPOSIT-IN-BANK:OBJECT. Each of the lines in the figure denotes a pair of directed connections. The connections joining TRANSFER-OF-CONTROL to DEPOSIT-IN-BANK and TRANSFER-OF-CONTROL:OBJECT to DEPOSIT-IN-BANK:OBJECT represent is-a (generalization) relations. The connections joining the two schema heads to their roles represent has-a (constituency) relations. The connections joining DEPOSIT-IN-BANK:OBJECT to MONEY and DEPOSIT-IN-BANK:RECIPIENT to BANK represent role-value relations.

1.2.3 Linguistic Knowledge

Linguistic knowledge is represented in terms of generalizations of utterances. An utterance is a meaningful sequence of words, such as a clause or a noun phrase, within a particular context. An utterance has roles for (a) the words which realize it; (b) a set of contextual elements, including a speaker and hearer; and (c) a content, the notion that the speaker wants to convey with the utterance.

The basic units of linguistic knowledge are **generalized utterances** (GUs), schemas which generalize in some way over the utterance roles. A GU defines a linguistic pattern consisting of a set of constituent roles, which have as their values particular morphemes or classes of morphemes. For example, there is a GU representing clauses with the word *deposit* in their VERB role and the concept of DEPOSIT-IN-BANK in their CONTENT role. I will refer to this GU as *DEPOSIT-IN-BANK; in general I will use asterisks to distinguish GU names from concept names. A GU may also have information associating pattern roles with aspects of the content. For example, the *DEPOSIT-IN-BANK GU includes an association between its DIRECT-OBJECT role and the semantic OBJECT role of its CONTENT. Other GUs define more general patterns such as the passive structure.

Figure 1.2 shows a portion of the *DEPOSIT-IN-BANK GU. The GU consists of the schema at the bottom of Figure 1.2. The CONTENT of the GU, that is, the event that a clause of this type refers to, represents the concept of DEPOSIT-IN-BANK shown in Figure 1.1. The shaded portion in Figure 1.2 corresponds to what appears in Figure 1.1. Notice that DEPOSIT-IN-BANK is actually embedded in the GU in a sense. In addition to its CONTENT, the GU has roles for two constituents, the VERB and the DIRECT-OBJECT. The

DIRECT-OBJECT has its own CONTENT, that is, the thing that it refers to. Thus roles can have their own roles. For this GU the DIRECT-OBJECT refers to the thing that is being deposited, so the nodes for the CONTENT of the DIRECT-OBJECT and the OBJECT of the clause CONTENT are merged; that is, they behave like a single node. This relationship is indicated in figures by a "tube" connecting the nodes. The verb of the clause has as its value the word node "DEPOSIT".[4]

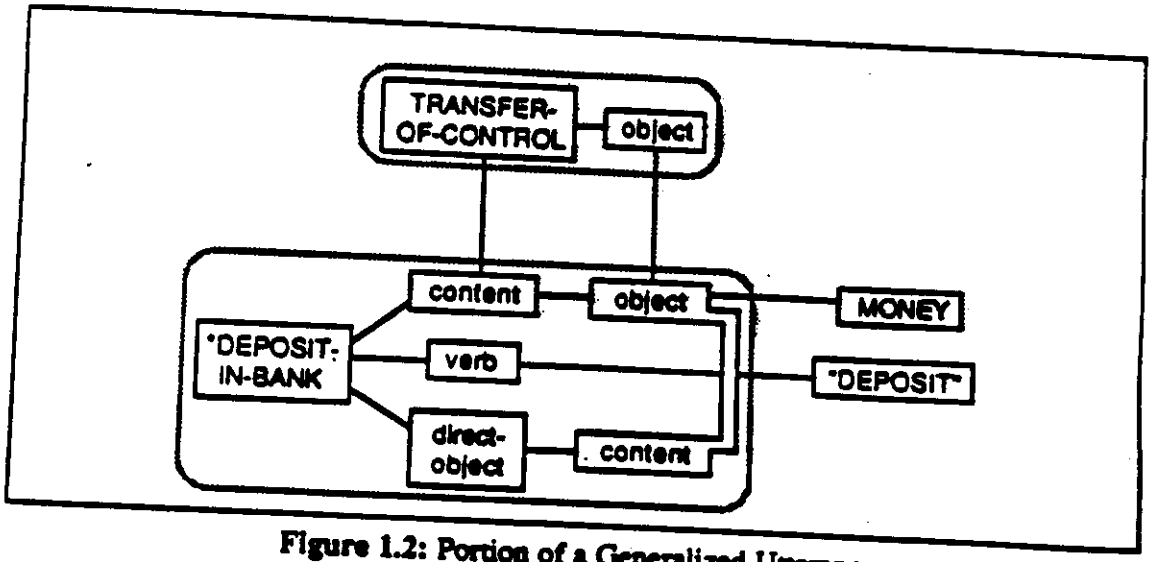

Figure 1.2: Portion of a Generalized Utterance

The generalized utterance in Figure 1.2 would be used in the generation of a specific utterance such as *Mary deposited $100 in the bank* or *could you deposit this money for me?*. Speakers use utterances such as these for particular reasons. That is, an utterance can be viewed as a plan to achieve a goal of the speaker. The system needs general knowledge about what sorts of utterances satisfy what sorts of goals. For this purpose the model has another type of unit, the **generalized illocution** (GI). Each GI is a specialization of the general INTEND schema, which has a GOAL and a PLAN role. The GOAL of a GI is a mental state or behavior of the hearer which the speaker wants to achieve, and the PLAN is a general type of utterance which can accomplish the goal. The GOAL-PLAN relationship represented in GIs is the same general relationship as that encoded in planning schemas for non-linguistic actions such as buying groceries and mailing a letter.

Figure 1.3 shows a simple GI representing the fact that a plan for getting someone to undertake something is to produce a *could you* question referring to the intended act. The GOAL of the speaker in this case is that someone else UNDERTAKE a particular ACT. The PLAN which achieves this goal is the production of a YES-NO-QUESTION with SUBJECT *you* and MODAL verb *could*. The sequencing information for these constituents is not shown in the figure. The person who is to undertake the act is the HEARER of the plan utterance (that is, the utterance should be addressed to that person), and the CONTENT of the utterance to be produced is just the ACT that the HEARER is to do. This GI specifies some of the constituents for the utterance to be generated, but others will be filled in by GUs on the basis of the CONTENT, that is, the nature of the intended act.

[4]This node actually represents different forms of the word *deposit*.

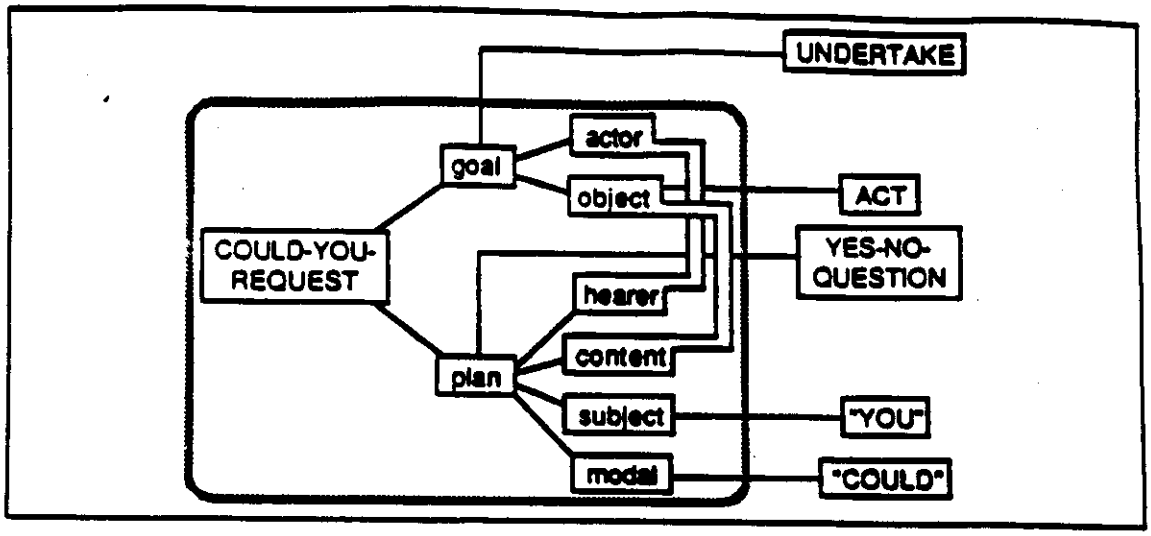

Figure 1.3: GI for Requests

Figure 1.4 shows how GIs and GUs fit into the rest of memory. Note that utterances are treated as acts and their CONTENT may be any type of concept.

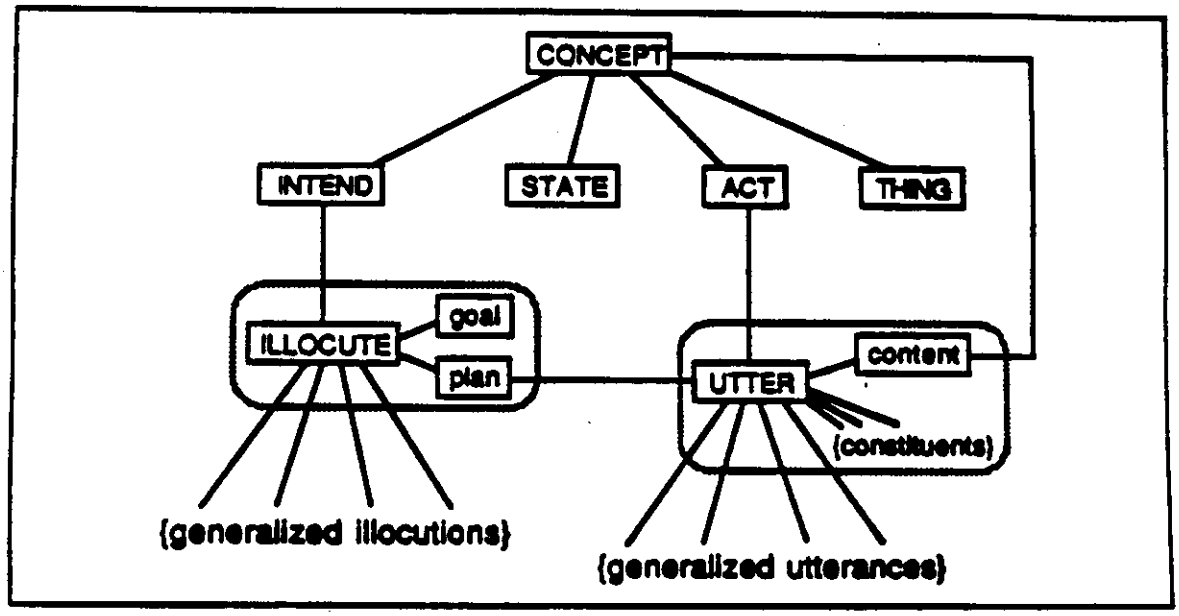

Figure 1.4: Relationship of GIs and GUs to Rest of Memory

1.2.4 Knowledge about Multiple Languages

For speakers of more than one language, the linguistic schemas for different languages need to be distinguished. In the CLM model this distinction is made primarily through the inclusion of restrictions on the HEARER role in the schemas. For example, in GIs and GUs for Japanese the HEARER role has JAPANESE-SPEAKER as its value.

The GI at the top of Figure 1.5 is just the one shown in Figure 1.4, but with the additional constraint that the person whom the speaker wants to perform the act is an

ENGLISH-SPEAKER. The GI at the bottom of Figure 1.5 is one possible corresponding GI for Japanese. This schema has the same GOAL as the English GI except that the ACTOR, that is, the person who is to do the act for the speaker, is a JAPANESE-SPEAKER. The GI PLAN roles differ of course. The PLAN for the Japanese GI takes the form of a yes-no question with the auxiliary *kurenai* and the main verb in the *te*-form.

Figure 1.5 shows examples of English and Japanese GIs for requests.

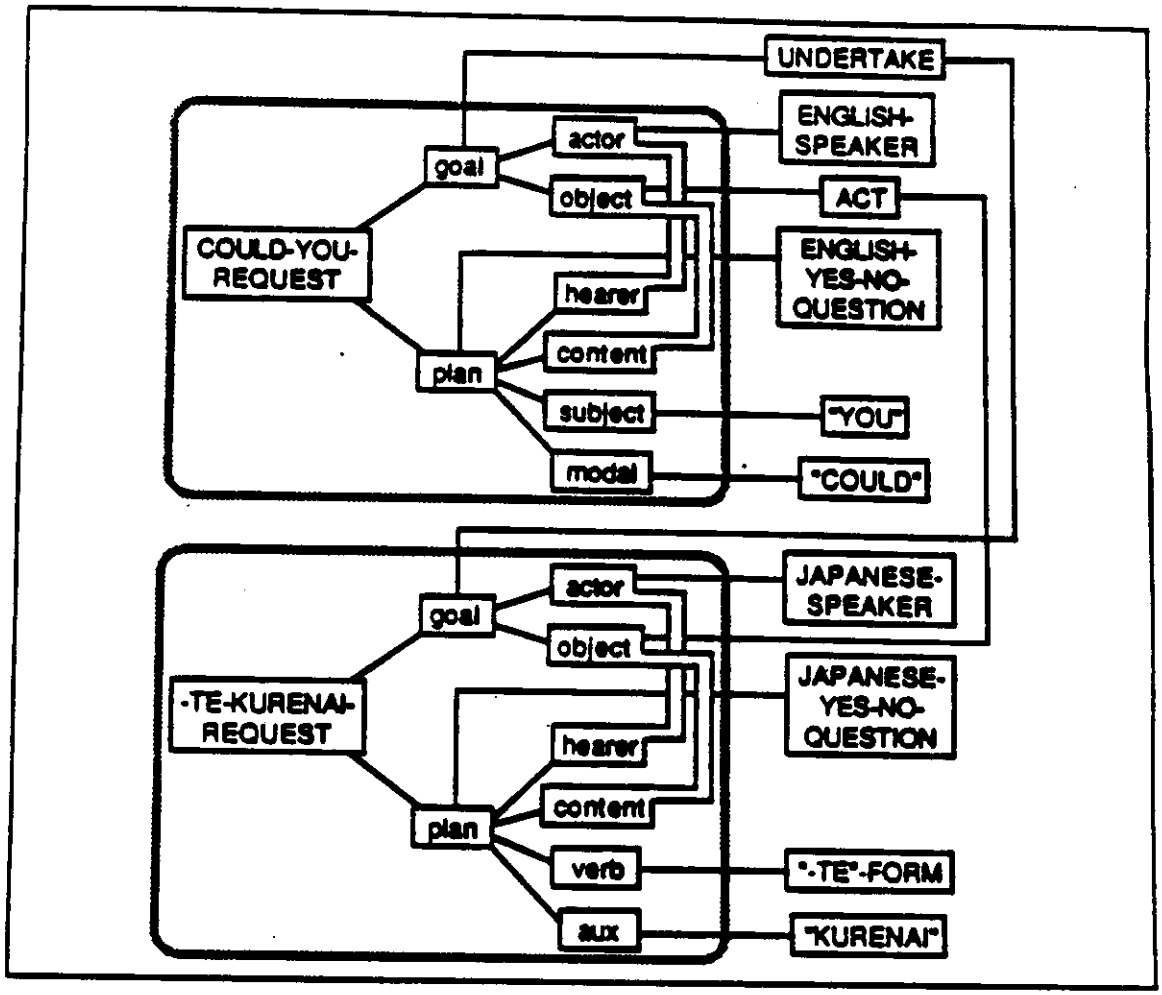

Figure 1.5: English and Japanese GI for Requests

GIs and GUs for different languages can be associated with one another to varying degrees. They may be linked indirectly through their GOAL or CONTENT roles as for the GIs in Figure 1.5; that is, they may represent patterns or words with the same function or meaning. Another possibility is a direct pointer from a GI or GU in one language to one in another language. This type of association enables a speaker to use a translation strategy in generation.

1.2.5 Use of Linguistic Knowledge

Consider how GUs and GIs are used in the generation of the sentence *could you deposit this money?*. The process begins with a goal of the speaker that the hearer carry out the requested act. The act is represented as an instance of the TRANSFER-OF-CONTROL

schema. The sum of money is represented as an instance of the MONEY schema. A portion of the input is shown in the lower left corner of Figure 1.6.

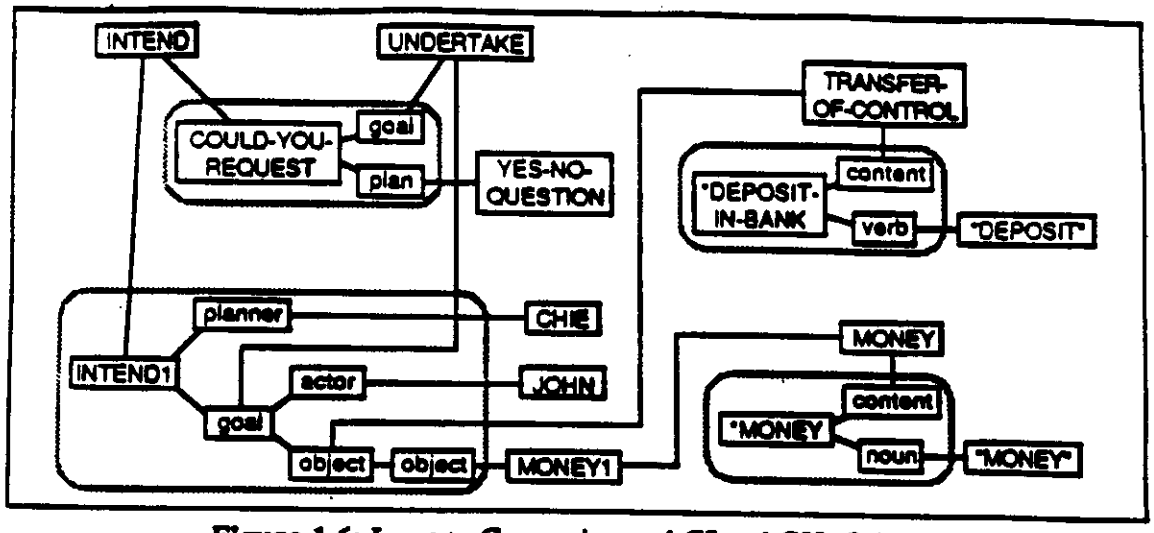

Figure 1.6: Input to Generation and GI and GUs Selected

The input is an instance of the general INTEND schema. The head node for the input is labeled "INTEND1"; in general I will use names of schemas followed by arbitrary numbers to indicate instances. The PLANNER for this intention is CHIE, the system itself. The GOAL is that JOHN UNDERTAKE a TRANSFER-OF-CONTROL of a particular instance of the concept MONEY, MONEY1. Missing in the figure is the information that the DESTINATION of the intended transfer is a BANK.

Starting from the elements of this input, the system selects one GI and three separate GUs. The GI is chosen on the basis of the type of goal and the GUs on the basis of the semantic features of the propositional content of the goal. Figure 1.6 shows the GI and two of the GUs and how they are connected to elements of the input.

1. The COULD-YOU-REQUEST GI (Figure 1.3) is selected because the goal is that someone UNDERTAKE something. This GI has as its PLAN the *could you* question pattern, shown only partially in Figure 1.6.
2. The *DEPOSIT-IN-BANK GU is selected to convey the nature of the requested transfer. It is the combination of the features of the transfer that results in the selection of this GU over others, in particular, the fact that the OBJECT is MONEY and the DESTINATION is a BANK. The GU pattern specifies that the verb is *deposit* and that the direct object of the clause should refer to the money being transferred, that is, to the semantic OBJECT of the transfer. This relationship is not shown in Figure 1.6, but it appears in Figure 1.2.
3. The *MONEY GU is selected to refer to the money.
4. The THIS-NP GU is also selected for the money noun phrase. This selection is made on the basis of the speaker's showing the hearer which money is intended by the noun phrase. THIS-NP does not appear in the figure.

1.2.6 Spreading Activation

Generation can be thought of as consisting of three processes: 1) the selection of linguistic schemas on the basis of input features, 2) the association of schema syntactic roles with input semantic roles, and 3) the sequencing of words and constituents in the selected patterns. In the CLM model all three of these processes are implemented by the spread of activation through the memory network.

At any time every network node has an associated activation that consists of a value between -1 and +1. Each node has a characteristic threshold activation, and when the node's activation reaches its threshold, the node "fires". A firing node sends activation along all of its output connections to neighboring nodes. The amount of activation sent on a given connection is proportional to the weight of the connection. Activation on nodes decays with time; this decay implements short-term forgetting.

Input to the generation process is a set of firing nodes representing features of the speaker's intention or of perceived events. Activation from these nodes converges on other nodes, resulting in their firing. This process continues until a set of nodes representing words fires in an appropriate order. This constitutes the system's output.

1.2.7 Schema Selection via Winner-Take-All Networks

Schema selection can be viewed as a matching of the pattern of input features against features of schemas in memory. A schema is selected when its head node fires. This node fires when it has received enough activation from its roles. The role nodes themselves fire in response to corresponding features of the input. For example, if the input includes the feature that the OBJECT of a TRANSFER-OF-CONTROL is MONEY, the OBJECT roles in all schemas with this same feature will also fire, lending a measure of support to all of those schemas.

Figure 1.7 shows simultaneous activation of the OBJECT roles in DEPOSIT-IN-BANK (the CONTENT of the *DEPOSIT-IN-BANK GU) and WITHDRAW-FROM-BANK (the CONTENT of another GU).

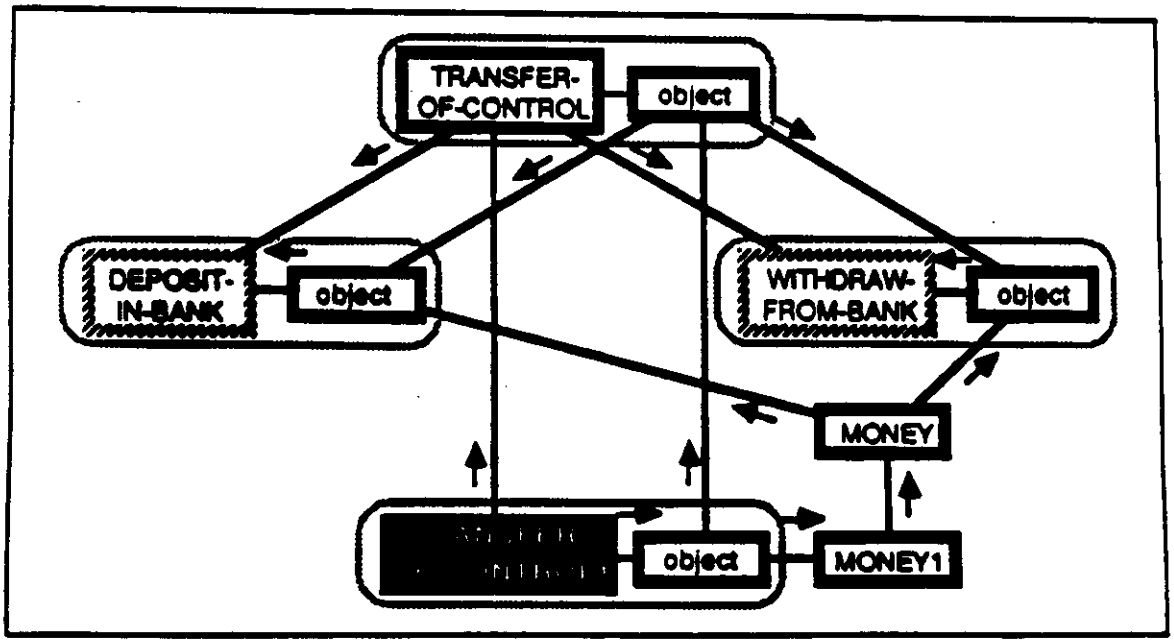

Figure 1.7: Activation of Roles During Schema Selection

Activation spreads from the node representing an instance of TRANSFER-OF-CONTROL. Inital activation sources are shown in black in figures, and the spread of activation is indicated by arrows and thick connection lines. The firing of TRANSFER-OF-CONTROL1 leads to a succession of firing nodes, indicated in the figure with thick borders. Once a node fires, it is inhibited for a period of time, during which activation from other nodes does not affect it. Thus activation in the figure moves in one direction only, even though each line represents a bi-directional pair of connections. Activation converges on DEPOSIT-IN-BANK:OBJECT and WITHDRAW-FROM-BANK:OBJECT from two sources, TRANSFER-OF-CONTROL:OBJECT and MONEY. Once these OBJECT roles fire, they send activation to their head nodes. At this point these nodes do not have enough activation to fire. Activation below the firing threshold is indicated by a hashed border. Additional features of the input would be needed to enable a selection to be made between DEPOSIT-IN-BANK and WITHDRAW-FROM-BANK. Since the spread of activation is parallel, information from all role nodes normally comes into schema heads simultaneously.

It is this process that implements the selection of GUs for particular conceptual inputs. Activation starting from the input eventually converges on the CONTENT role of a suitable GU, and this node passes activation on to the GU head node. An important feature of this process is competition among groups of nodes. Competition is implemented through winner-take-all (WTA) networks, subnetworks whose members inhibit each other through connections with negative weights. When one of these networks is activated, it will tend to cause one and only one of its members to fire. WTA networks implement decision making in the system. The firing of the WTA network node with the highest activation constitutes the decision. For example, the schemas representing noun lexical entries compete with one another in such a way that only one can be selected for a given input. This competition prevents a person from being referred to as *Mary*, *the woman*, and *she* in the same phrase.

Figure 1.8 shows a portion of the WTA network representing the competition between different ways of referring to the notion of TRANSFER-OF-CONTROL. Inhibitory connections are indicated by fuzzy lines. Competition between the CONTENT roles of the GUs forces the system to select just one GU to refer to a given instance of TRANSFER-OF-CONTROL. Not shown in the figure are the features which distinguish the different entries.

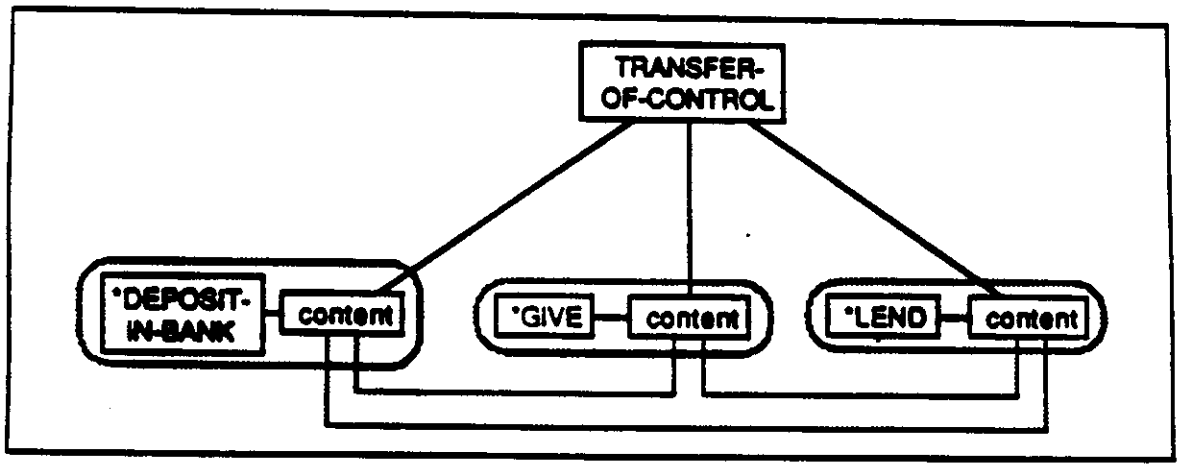

Figure 1.8: Competition Among Generalized Utterances

The schema selection process does not require all of the input features to be matched. All that is necessary is for one schema head to receive enough activation from its

roles to win over other competing schemas. Thus spreading activation and winner-take-all networks implement a kind of lenient pattern matching.

For multilingual speakers the schema selection process involves not only elements of the speaker's goal and its propositional content but also the type of hearer. If the hearer is monolingual, schemas for the language that the hearer understands will tend to win out over schemas for other languages. However, since the speaker has no way of turning off knowledge of other languages, we can expect it to have an effect under certain circumstances. For example, for a given concept, a schema for one language may be more familiar to the speaker than that for another, and that schema is likely to be selected over the other. This may result in language, as in sentence (1.12). Figure 1.9 shows how the two competing GUs would be related for this example.

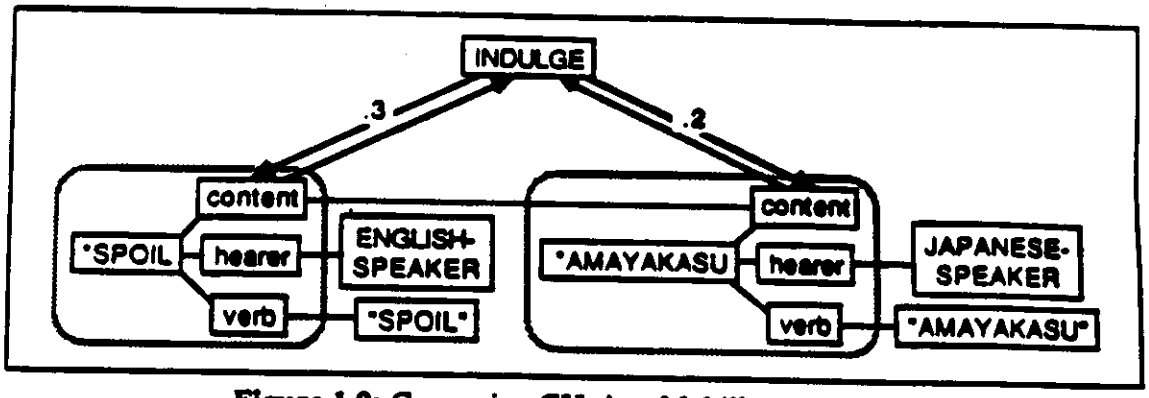

Figure 1.9: Competing GUs in a Multilingual Context

The speaker intends to refer to an instance of the concept of INDULGE. For simplicity the figure does not indicate how this concept is represented. The speaker has two GUs associated with this concept, one each for English and Japanese. Apparently the English GU is more familiar to her. In the model this is reflected in the higher weight on the connection from INDULGE to the CONTENT role of the English GU. In the figure both directed connections are shown for the these relationships, and possible weights are indicated for the downward connections.

Sentence (1.12) was produced in a bilingual setting. When the hearer does not know one of the speaker's languages and a schema in that language is selected, on the other hand, the speaker may translate it directly into the appropriate language if this is possible. This kind of translation is what we see in sentence (1.11).

1.2.8 Role Binding

Once a schema has been selected on the basis of some of its roles, other roles in the schema fire, providing information that the system needs in order to proceed. When a GU is selected on the basis of the content being conveyed, the pattern elements specified in the GU fire, yielding words or subpatterns that will be part of the utterance.

As noted above, the information in a pattern often includes role associations. For example, the *DEPOSIT-IN-BANK pattern associates the semantic OBJECT of the transfer with the DIRECT-OBJECT of the clause (see Figure 1.2). For the generation of sentence such as *could you deposit the money?*, the system needs to be able to use this information to have the direct object of the utterance refer to the money, which is the OBJECT in this case. In a connectionist approach such as this one there is no mechanism for actually binding one node to another temporarily. What corresponds to binding instead is the more

or less simultaneous firing of the nodes to be associated, for this example, the DIRECT-OBJECT and the node representing the money. Figure 1.10 shows how this happens.

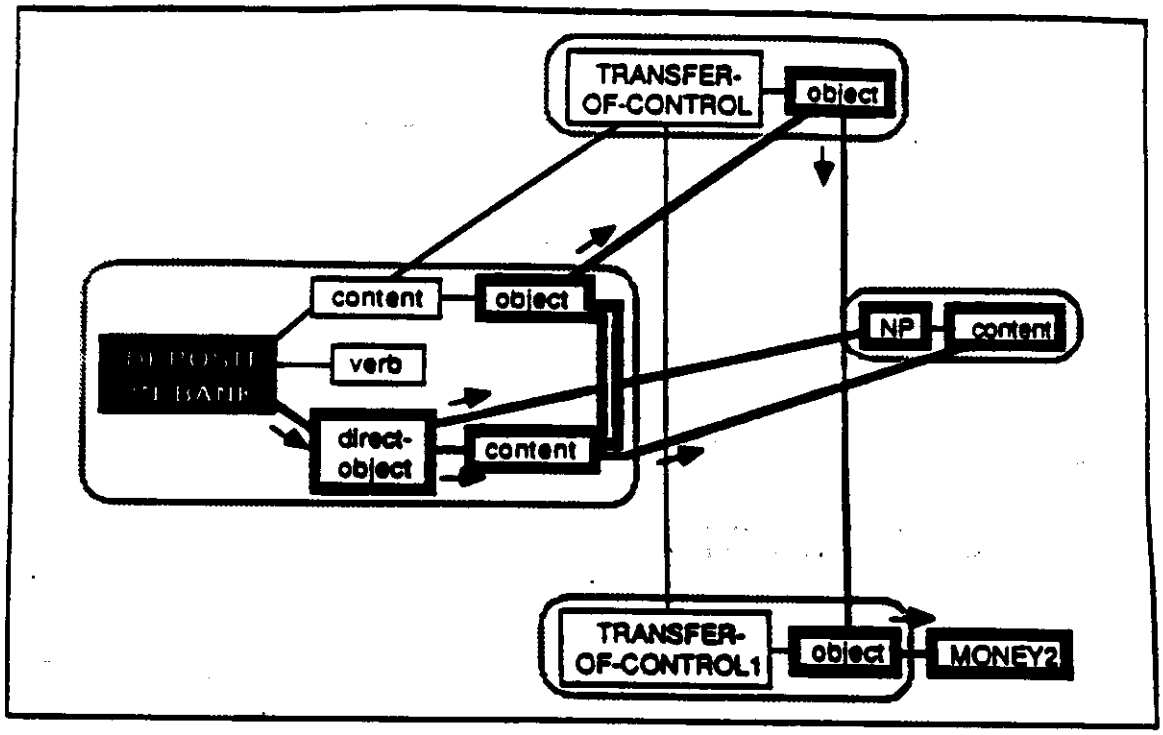

Figure 1.10: Binding of Syntactic and Semantic Roles

The DIRECT-OBJECT node fires as activation spreads through the *DEPOSIT-IN-BANK pattern. This causes the merged node in the schema to fire, spreading activation in two directions. In one direction the spread reaches the MONEY2 node. In the other the NP GU is activated, starting off the generation of the DIRECT-OBJECT constituent. The firing of MONEY2 and NP initiates the GU selection process of the DIRECT-OBJECT constituent which will eventually access the GU *MONEY (Figure 1.6).

1.2.9 Sequencing

Because processing in the model is parallel, the sequencing of words to be output is not a trivial matter. Consider sentences (1.8): *Mary sent him a letter, Mary sent it to John*, etc. There must be a way to represent the fact that either the DIRECT-OBJECT, the constituent referring to the letter in this case, or the RECIPIENT-REFERENCE constituent can appear immediately after the verb. To handle this, the verb sends some activation to both constituents, and these compete via a WTA network to fill the position. When the constituent appearing first is complete, it sends activates the other one. For this to happen, however, there needs to be a way for the system to signal when a constituent is complete. For this reason each constituent or phrase has a separate node representing its end.

Figure 1.11 shows some of the sequencing relationships in the *SEND-MAIL GU, the one selected for sentences (1.8).

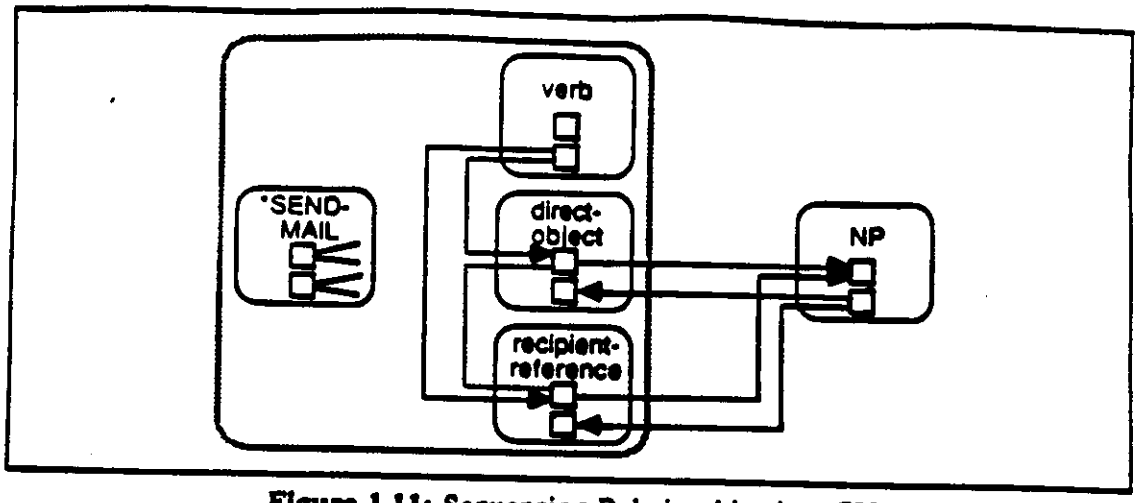

Figure 1.11: Sequencing Relationships in a GU

Each of the squares with rounded corners corresponds to a single phrase or constituent node in the previous figures. The pairs of small squares denote the start and end nodes for each phrase or constituent. Note that many of the sequence relationships are represented by single directed connections rather than pairs of bi-directional connections. The end node for the VERB sends activation to the start nodes for the two constituents which may follow, and these nodes inhibit each other. One will normally have activation from a semantic or pragmatic source that will allow it win out over the other and fire first. The start nodes for the DIRECT-OBJECT and the RECIPIENT-REFERENT activate the start node for the NP GU, which contains general sequencing information for NPs. The end node for the NP GU activates the end nodes for the two phrasal constituents. This activation effectively signals whichever constituent is being generated that it is complete.

1.3 Method

1.3.1 Cognitive Modelling

Current methodologies in applied linguistics follow not only those in linguistics, but also experimental methods from educational psychology, social psychology, and psycholinguistics and, to a lesser extent, traditions deriving mainly from anthropology and literary criticism. This thesis takes a relatively novel approach, one that is most closely associated with the new field of cognitive science, and in particular, with one of its central disciplines, artificial intelligence (AI) and with the newly emerging field of connectionism.

This methodology will be referred to as cognitive modelling. The name is often used for a subfield of AI, distinguished from other subfields in terms of its concern with understanding and simulating human intelligence, regardless of whether such simulations have immediate or obvious engineering applications. The approach involves at least the following:

1. a cognitive phenomenon to be modelled, defined in terms of its input-output behavior and evidenced in a body of data
2. the model itself, consisting of a more or less rigorous description of a set of procedures and the data that they operate on

3. a computer program, which demonstrates for a limited range of inputs and outputs that the model performs as claimed.

1.3.2 The Computer Program

What distinguishes cognitive modelling most clearly from other methods is the program, often referred to as the "implementation". Unless the primary goal of the research is to produce intelligent software, one might ask what function the program serves. At least three functions can be singled out. First, the program provides a means of checking the accuracy of the model. For a complex task such as language processing, it is simply too difficult to perform such a check using ordinary pencil-and-paper methods. The program generates behavior, and the researcher can then compare this behavior with the predictions of the theory. Second, the act of writing the program invariably brings up theoretical deficiencies in the model which the researcher would not have discovered otherwise. As Brandt Corstius (1978) maintains in his Second Law of Computational Linguistics: "Any linguistic description, no matter how precise, if it is itself not a computer program, will turn out to contain a mistake if one tries to make a program of it" (p. 125). The discovery of problems leads the researcher to make revisions and then attempt to implement the model in its new form. From the point of view of the day-to-day activities of the researcher, cognitive modelling is thus a continual process of theorizing, implementing, revising, reimplementing, and so on. Third, the program, once it is complete, can also serve as a tool for further research, either by the same investigator or by others, who might modify the program to suit their needs.

However useful the program may be, there are good reasons for distinguishing it from the model itself. First, the range of inputs that are handled by the program is necessarily restricted, and simplifying assumptions often need to be made in designing the program. Second, some aspects of the program are irrelevant to the model. These include the computer language it is written in, the type of computer it runs on, and various features of programming style. These aspects are not necessarily irrelevant to the project as a whole, however, because, as noted above, the program may be intended to serve functions other than simply demonstrating the validity of the model. For example, if the program is to be used by other researchers in evaluating their own hypotheses or if it is to be the basis for a practical system of some type, then it is important that the code be both readable and adaptable.

The implementation of a model in the form of a computer program does not necessarily imply that the "computer metaphor" of human cognition is being maintained. It is possible to program a computer to behave in ways which are not particularly efficient for the computer but which reflect aspects of human processing. For example, the parallel processing which seems to characterize much of human cognition can be simulated on a conventional serial computer by breaking the processes into time steps of arbitrarily small length and running each process through one time step at a time.

1.3.3 Constraints on Cognitive Models

It is not enough to demonstrate that a model (or its implementation in a program) yields the expected output given particular input. In fact, by itself this is of little interest since it is generally accepted in automata theory that any well-defined behavior can be simulated by a very simple automaton, a Turing Machine (Church's thesis). A model needs to be constrained in ways other than the accuracy of its output. Types of constraints, or equivalently, criteria for evaluating models, include the following:

1. Parsimony: Is duplication of knowledge avoided?

2. Memory limitations: Are the memory requirements of the model reasonable considering the storage capacity of the human brain?

3. Temporal limitations: Does the model perform within the range of humanly possible speeds, or equivalently, does the model perform using more or less the same number of computational steps as a human might require for the task?

4. Uniformity/generality: Are the mechanisms and representational structures which are posited applicable to various domains? Is there an attempt to limit the types of mechanisms and structures which are involved?

5. Neurophysiological factors: Is the behavior of the primitive units in the model in line with what is known about the behavior of neurons or ensembles of neurons?

6. Learnability: Does the model include mechanisms by which the knowledge is acquired, or is the knowledge that the model is provided with in a form that might have been acquired?

Some of these constraints agree or conflict with others. Parsimony (constraint 1) limits the amount of memory required (constraint 2). On the other hand, redundancy in the representation of knowledge may result in faster processing (constraint 3). The savings come when there are compiled chunks of knowledge which allow the system to avoid accessing each of the pieces from which these chunks are composed. For the human brain, unlike the conventional computer, relative slowness of processing is a more important limiting factor than memory capacity (Feldman & Ballard, 1982). That is, redundancy is desirable when it can speed up processing. For this reason parsimony is not accorded much importance in the present model. Specific knowledge, which may involve duplication, has priority over general knowledge because it leads to greater fluency.

Memory limitations cannot be ignored, however. A general concern is whether a cognitive model will "scale up", that is, whether the memory requirements would be too great if the model's memory were extended to encompass all of the knowledge needed for a wide variety of contexts. This sort of consideration has not guided the design of the CLM model, and I return to the problem in Section 11.1.1.

Cognitive models differ considerably in the importance attached to uniformity (constraint 4). For modular approaches to cognition (Fodor, 1983), exemplified in particular by Chomskyan linguistics, there is little attempt to seek commonalities across the different components of language or between linguistic and non-linguistic knowledge. By contrast, one general aim of the present approach has been to develop a cognitive architecture and a set of processes which can be used in language analysis and non-linguistic processing as well as in language generation.

Another class of models for which uniformity is not particularly important is those which are relatively unconstrained in terms of the primitive constructs used. For example, a model may be built out of the set of functions provided by a programming language such as LISP or out of a wide range of computational devices such as production systems, demons, and discrimination networks.[5] Models such as these have their place. They excel at producing desired output behaviors on a computer, and they have yielded many insights into high-level cognitive processes. However, there is also much to be gained by attempting to implement these same high-level processes using a more limited set of computational constructs. In particular, a more restricted approach may bridge the gap between the neural and the cognitive levels, an area of considerable current interest in

[5]See Charniak, Riesbeck, McDermott, & Meehan (1987) for an overview of these and other AI programming concepts.

cognitive science. The present model belongs to the more constrained category. While no claims are made here about direct correspondences between theoretical entities and biological entities, the model is "neurally inspired" in the sense that none of the basic processes seems to be incompatible with what neurons are capable of doing (Feldman & Ballard, 1982).

It has not been a goal of this project to develop an explicit learning mechanism (constraint 6). In Section 11.1.2 some suggestions are made regarding how acquisition might take place within the general framework of the model.

1.4 Organization of the Thesis

Chapter 2 of the thesis deals with knowledge representation in general. The schema framework is discussed, and some of the concepts involved in the generation of example sentences (1.1-12) are considered in detail. Particular attention is paid to the representation of goals and plans and to the way in which alternative views of a single concept are handled. Finally, the implementation of the schema memory in the form of a network is described.

Chapter 3 is concerned with the representation of linguistic knowledge. The hierarchical organization of linguistic memory is motivated through a discussion of a bottom-up view of language acquisition. The nature of utterances and generalized utterances is described. Emphasis is placed on the semantic aspects of GUs, such as the representation of semantic composition and the way in which indexical features, such as properties of the speaker or hearer, can be referred to in GUs. Generalized illocutions are also treated, with examples coming mainly from the category of directives, that is, requests, commands, and the like. There is also a consideration of two other types of linguistic knowledge, generalizations about vocatives, and generalizations associating patterns directly with the perception of particular event types (as for *speak of the devil*).

Chapters 4, 5, and 6 deal with the process of generation itself. In chapter 4 the algorithm for the propagation of activation, including the special behavior of winner-take-all networks, is described, and the implementation of schema selection in terms of this mechanism is discussed. The generation of substitution errors within the framework of the model is also discussed. Chapter 5 examines role binding and how it is handled using spreading activation and merged nodes. The problem of crosstalk, that is, confusion among various paths of activation, is handled using winner-take-all networks to stagger the firing of nodes which might lead to confusion. The place of role binding in schema selection is also considered. It figures particularly in selection which is guided by paradigms in the language rather than by distinctions which the speaker intends to make. An example of such a paradigm is the singular-plural distinction in English. Finally, the generation of exchange errors is discussed. Chapter 6 describes how sequencing is dealt with in the model. Explicit connections are used to represent sequencing relationships, and winner-take-all networks implement competition for a particular output position. A further feature of the approach is the use of special nodes to represent the ends of phrases and constituents.

Chapter 7 looks at the organization of linguistic memory and the process of generation for speakers with knowledge of more than one language. Five possible relationships between GUs for different languages are considered, ranging from none at all to direct translation associations. Evidence is presented for four types of relationships. The behavior of speakers who perceive a gap in their knowledge of the target language during generation is also examined. In such cases the robust matching mechanism may

allow them to find a partially matching schema. They may also be aided in this process by the existence of a matching schema from some other language.

Chapter 8 considers ways in which the network and spreading activation mechanism are applicable to comprehension as well as generation. Schema selection, role binding, and sequencing are discussed.

Chapter 9 examines related research. This is discussed under three headings, unified cognitive models, generation models, and connectionist models of language processing.

Chapter 10 discusses the implementation of the model in the program CHIE. A trace from a sample run of the program generating a sentence is included.

Chapter 11 summarizes the findings of the thesis and suggests future work. The most promising future directions concern the development of an acquisition component; explorations in the use of distributed representations, that is, representations in which a concept is not localized in a single node; expansion of the coverage of the model to other areas of discourse, syntax, and semantics; and the application of the model to machine translation.

There are also an appendix containing a listing of the program itself.

Chapter 2

Representing Concepts

2.1. A Schema-Based Memory

As we saw in Chapter 1, an account of language generation must build on a system for representing concepts. In producing a noun phrase such as *the chair*, a speaker has in mind a particular object, and she needs to be able to get from her representation of that object to the general concept of CHAIR in order to locate the word *chair*. In producing a sentence such as *Mary deposited $100 in the bank*, a speaker is thinking of an event, and she must be able to classify it as an instance of the concept of DEPOSIT-IN-BANK in order to find the word *deposit*. In uttering the expression *your honor*, a speaker must know something about the concept of TRIAL and in particular about the role that the presiding judge plays in it in order to select this expression over some other such as *excuse me*.

This chapter looks at the way in which concepts are represented in memory in the CLM model. The memory is characterized by a simple, uniform architecture; all of the system's knowledge is contained in a network of nodes joined by directed connections. This network can be viewed as an implementation of a higher-level type of knowledge representation, a schema-based or semantic network memory. Thus there are two levels at which the system's memory can be described. In this section I look at the representation of concepts as schemas, and in Section 2.2, I describe how schemas and their interrelationships are implemented in the memory network of the model.

2.1.1. Schemas

One of the most fundamental properties of human cognition to be accounted for is the ability to abstract out the essential properties in situations and to use this knowledge later in recognition, inferring, and planning. When a person sees a chair, it is enough for him to identify a few surface features, say, a leg or two and the general shape of the back, to recognize that what he sees is a chair and to know that it is for sitting on. When a person hears a story about a trial, he is able to recognize it as such even without hearing the word *trial* by identifying a small number of key features. Then on the basis of this recognition, he is able to fill in missing details or to build expectations about what is to follow by using the default knowledge that he has about trials, for example, that the defendant will be found guilty or innocent at the end of the trial. When a person answers a telephone and the caller asks for someone she has never heard of, the answerer is able to quickly find the plan she has stored for this set of circumstances and to execute the plan by uttering, "I'm afraid you have the wrong number."

The structured generalizations that are the basis for recognition, inference, and planning are known as schemas (Minsky, 1986; Rumelhart, 1980; Schank & Abelson, 1977). The central ideas in schema-oriented approaches concern the internal structure of schemas, the relations among schemas, and the use of schemas in processing.

2.1.2. Schema Structure

Schemas organize essential knowledge about concepts. The fundamental relation involved is one of "belongingness", the association between a head concept and its roles. A chair is expected to have four legs, a relatively sturdy construction, and the function of providing a place for people to sit. A trial is expected to have a set of participants, including the judge, lawyers, witness, and defendant; a series of characteristic episodes, including the reading of the charges, the witnesses' testimony, and the announcing of the verdict; and a set of functions, including the protection of society from dangerous criminals. A situation involving a phone call to a wrong number is expected to have a

caller, an answerer, and an appropriate response made by the answerer. The schemas for CHAIR, TRIAL, and WRONG-NUMBER-CALL have roles for these characteristic features. A role associates the head concept with another concept functioning as the "value" of the role. For example, the PRESIDING-JUDGE role associates a trial with the general category of JUDGE, which functions as the value of PRESIDING-JUDGE.

Roles and their values are not necessarily defining properties for schemas. A chair without legs, a beanbag chair, for example, is still a chair, and a trial in which there are no witnesses is still a trial. Schemas are selected for use in processing not through the identification of any particular set of properties but through the identification of a significant number of the typical properties in the schema. This view of schemas is most closely associated with the work on prototypes by Rosch (1978) and her colleagues.

2.1.3. Schema Relationships

Schemas are associated with each other in two ways. First, schemas may be composed of other schemas. For example, the sentencing episode in the TRIAL schema has its own roles for the ACTOR and OBJECT of the sentencing. This property gives schemas a hierarchical structure.

Figure 2.1 shows a portion of the TRIAL schema. This figure, like most others, shows only a portion of the information that an ordinary person would have access to for the concept.

```
(TRIAL is-a ELABORATE-PROCEDURE
    (presiding-judge (JUDGE ?J)
        (garment ROBE))
    (defendant (PERSON ?D))
    (prosecuting-attorney ATTORNEY)
    (witnesses SET
        (member (PERSON ?W)))
    (episode1 ASSERT                          ;;testimony
        (actor ?W))
    (episode4 ASSERT                          ;;sentencing
        (actor ?J)
        (object CAUSE
            (actor GOVERNMENT)
            (object SUFFER
                (experiencer ?D)
                (time FUTURE)))))
```

Figure 2.1: Portion of the TRIAL Schema

Role values follow the role names. Values include names for other schemas and variables. Schema names are capitalized in the figure. Variable names, consisting of a letter preceded by a question mark, are used to indicate role-to-role equivalences. For example, the schema needs to include the information that the ACTOR of the act of reading the sentence is the same as the presiding judge. To show this relationship, the roles are assigned the same variable name, ?J. Some roles group in types; for example, in the TRIAL schema there is a set of roles representing EPISODES in the trial. I will denote such roles by the role type name followed by a number, e.g., EPISODE1. Two episodes appear in the figure, the testimony, in which a member of the set of WITNESSES ASSERTs something, and the sentence, in which the PRESIDING-JUDGE ASSERTs that the GOVERNMENT will

CAUSE the DEFENDANT to SUFFER some punishment. I will sometimes add comments to indicate what a particular role represents. These follow pairs of semicolons.

Schemas may also be related through what is known as the is-a relation (Brachman & Schmolze, 1985; Fahlman, 1979); that is, one schema may be a specialization of a higher-order schema. For example, the CHAIR schema specializes the schema for FURNITURE (CHAIR "is-a" FURNITURE), and the WRONG-NUMBER-CALL schema specializes the schema for PHONE-CALL. The is-a relation organizes the schemas in a hierarchical network. The most important aspect of this relation is the correspondence between the roles at different levels of the is-a hierarchy. For example, the DEPOSIT-IN-BANK schema is associated through an is-a relation with the more abstract schema TRANSFER-OF-CONTROL, and there is a mapping between the roles of these two schemas. The DEPOSITOR in the DEPOSIT-IN-BANK is just the TRANSFERRER in the TRANSFER-OF-CONTROL schema, who is in turn just the ACTOR in the even more abstract ACT schema. Thus we can also refer to the DEPOSITOR as the ACTOR of the DEPOSIT-IN-BANK schema.

Figure 2.2 shows a portion of the DEPOSIT-IN-BANK schema. The relevant is-a relationship is indicated following the schema name. In addition to the basic participants, this schema show a set of expansions and two purposes. Expansions are subacts which together realize the larger act; for example, the act of filling out a deposit slip is an expansion for DEPOSIT-IN-BANK. Purposes are events which the ACTOR hopes to achieve through performing the act. For DEPOSIT-IN-BANK, one purpose is keeping the deposited money safe.

Is-a hierarchies permit the system to avoid unnecessary redundancy. Any information appearing in a particular schema is implicitly shared by all of the specializations of that schema, and this information can be inherited if it is needed. Thus each schema need only record information which does not appear at a higher level of the is-a hierarchy. For this reason the BANK-DEPOSIT schema shown in Figure 2.2 does not include information which belongs in the TRANSFER-OF-CONTROL schema, for example, the fact that the ACTOR is a HUMAN and a PARTICIPANT and the fact that the effect of the depositing is that the recipient, that is, the bank, now has control over the object, that is, the deposited money. Inheritance also applies to the relation between roles and their values. For example the PROCESSOR of a BANK-DEPOSIT implicitly inherits all of the properties in the TELLER schema.

At the bottom of an is-a hierarchy are the representations for instances, that is, particular objects, events, and states. Instances have the general structure of schemas, except that their roles tend to have other instances rather than schemas for values. Consider how a person might encode the memory for a particular trial, say one that he has observed. Information about this trial that duplicates what is found in the TRIAL schema does not need to be recorded since it can be inherited when needed. What the observer needs to store about the particular trial is what is unusual or interesting about it. For example, the judge might have been a dwarf, a fact which an observer would be likely to take note of. Or the judge might have suffered a heart attack during the trial, an event of considerable importance and hence again one which an observer might explicitly encode.

Figure 2.3 shows how this trial, call it TRIAL34, would be represented in the schema notation we are using.

```
(DEPOSIT-IN-BANK is-a TRANSFER-OF-CONTROL
    (actor ?A)
    (object (MONEY ?O))
    (source ?A)
    (recipient (BANK ?B))
    (processor (TELLER ?P))
    (duration TEMPORARY)
    (expansion1 PHYSICAL-TRANSFER      ;;get to bank
        (actor ?A)
        (object ?A)
        (destination ?B))
    (expansion2 WRITE                  ;;fill out deposit
        (actor ?A)                     ;;   slip
        (location (PAPER ?L)))
    (expansion3 TRANSFER-OF-CONTROL    ;;give to teller
        (actor ?A)
        (object (?L & ?O))
        (source ?A)
        (recipient ?P))
    (expansion4 TRANSFER-OF-CONTROL    ;;receive receipt
        (actor ?A)
        (object PAPER)
        (source ?P)
        (recipient ?A))
    (purpose1 PREVENT                  ;;keep money safe
        (object LOSE
            (object ?O)))
    (purpose2 INCREASE-VALUE           ;;earn interest
        (object ?O)))
```

Figure 2.2: Portion of Schema for BANK-DEPOSIT

```
(TRIAL34 is-a TRIAL
    (presiding-judge (DWARF ?J))
    (episode8 HEART-ATTACK
        (victim ?J)))
```

Figure 2.3: An Instance of the TRIAL Schema

2.1.4. Schema Use

Schemas function like templates. During processing, appropriate schemas are "activated" on the basis of particular features present in the environment. These schemas are then instantiated, yielding instances representing particular entities in the processing context. For example, an instance might represent a particular character or event in a story, instantiating the schema for a general category of event or person. As a part of processing, some of the roles in the instantiations which do not already have particular values have these values filled in. For example, a chair (like any physical object) has a particular composition; in instantiating the CHAIR schema, a system might specify that the chair in question is composed of rattan; that is, the COMPOSITION role is associated with the value

RATTAN. Alternately, if this information is not available, the system might simply fill in, or inherit, the value of the COMPOSITION role with a default value specified in the schema, say, WOOD.

Schemas need to be indexed in such a way that they can be rapidly accessed during processing. On seeing a beanbag chair, a person quickly recognizes it as such, without apparently having to traverse the hierarchy of schemas from PHYSICAL-OBJECT to FURNITURE to CHAIR. On receiving a phone call dialed to a wrong number, a person rapidly locates the schema for this situation, without necessarily being aware of the schemas for other kinds of phone calls or dialogue openings. Schema indexing and access is a complex issue which has generated a good deal of research (e.g., Fahlman, 1979; Kolodner, 1984; Pazzani, 1988; Rumelhart, Smolensky, McClelland, & Hinton, 1986). Like many other network models of memory, the present approach makes use of a spreading activation mechanism for schema access. This mechanism is discussed in Chapter 4.

Up to this point I have illustrated the basic representational notions with knowledge about trials and bank deposits, but the same points hold for schemas and instances in other categories. The schema for CHAIR, for example, has roles for parts such as the back and the legs, roles for its functions, and a role for the way in which a chair is used. There are also schemas for specializations of CHAIR, for example, one for ROCKING-CHAIR, which has a role for the rockers and additional information associated with its MANNER-OF-USE role.

2.1.5. Representation of Acts and Procedural Knowledge

For the purposes of this thesis, it will not be necessary to go into details of the representation of most categories of concepts. What will be important is the representation of acts because all knowledge of language will consist of specializations of knowledge about what it means to act. The representation I will describe is based in part on ideas of Schank and Abelson (1977), Dyer (1983), and Wilensky (1983).

Consider first some of the general knowledge that people have about acts. Acts (like all events) have what I will call **effects**; that is, they result in changes in the world. For example, one effect of giving something to someone is that the object is no longer in the "control" of the giver. Acts, unless they are viewed as primitive acts, are composed of a set of **expansions** which realize the act. For example, the act of depositing money in a bank (Figure 2.2) expands to a set of sub-acts including filling out a deposit slip, giving the slip and the money to a teller, and taking the receipt from the teller. In most cases, including this one, there is a sequence imposed on the set of expansions. Thus filling out the deposit slip must precede handing it to the teller.

What distinguishes acts from other events is that they are intentional. Acts have an actor, an animate participant who has a reason for performing the act. The actor of an act is of course also the actor of each of the expansions of the act. An act results from the actor's intention to achieve a **goal**. The actor cannot always be sure whether the act will succeed; it constitutes a **plan** to achieve the goal. Given a particular goal, a plan may be chosen either because it is directly associated with that type of goal or because there is a match between the goal state and one of the effects of the act that makes up the plan. The decision to carry out a plan in order to satisfy a goal is also complicated by the fact that certain preconditions may need to be satisfied before the plan can be realized. For example, say a planner wants to have a new VCR. She will probably decide that a way to achieve that goal is to buy one. However, she will then discover that buying something requires having an amount of money equal to the price of the article (assuming for simplicity's sake

that credit is not available). If this is more than she has, she will have to plan to get the money, seek an alternative plan to get the VCR, or abandon her goal.

Knowledge about particular types of acts may exist at two different levels. At the more general level the actor is unspecified, that is, the knowledge applies to acts made by the system itself as well as by others. Such knowledge is usable in recognizing (understanding, explaining) the acts of others as well as in planning one's own acts. This would generally be knowledge acquired through observation or formal study. Much knowledge about language appears to be of this type. That is, it is learned through the process of understanding language, and it can be used in both comprehension and generation. For other kinds of acts, however, knowledge may only be available at the level of the system's own actions. For most people, for example, knowledge about how to drive a car is largely of this type. They are able to apply the knowledge when it needed, but it is difficult for them to explain it to others, and it not usable in the recognition of others' actions. Such knowledge is learned through practice rather than through observation or formal study. It is what is normally referred to as **procedural knowledge**. Procedural knowledge is distinguished from **declarative knowledge**, consisting of facts which the system can describe and which are acquired mainly through the process of being told.

Carrying out (or recognizing) a complex act may require the system to access any number of schemas in memory. For example, in the case of making a bank deposit, there may be a general schema for filling out forms, another for handing things to people, and another for taking things from people. On the other hand, if a person often makes deposits, we would expect the knowledge to be available in a more compact form. In such a case there might be a special schema for depositing money in the particular bank where the planner has her account. This would include particular knowledge about the expansions of the act: how to get to that bank, precisely what needs to be filled in on the deposit slip, and so on. Such a schema is very context-specific, and it does not permit much variation in the features that it will match. For example, it would not apply if the goal were to withdraw money from the bank (though it might help the system make an analogous plan for this situation). Knowledge in this "compiled" form is faster to use than the more general knowledge, although it may be redundant in the sense that the system could derive everything in the context-specific schema using the general schemas. Thus there is a tradeoff between processing efficiency and memory requirements.

In the CLM model the basic schemas used in planning acts and understanding the acts of others are specializations of the INTEND schema, which has roles for a GOAL and a PLAN to achieve the goal. A schema of this type encodes the knowledge that a particular kind of act is a way to satisfy a particular kind of goal. It may be accessed via its GOAL role (in planning) or its PLAN role (in recognizing or understanding). The PLAN role may have as its value a schema for a particular type of act, or it may have a set of EXPANSIONS specifying the steps to be executed as part of the plan. A PLAN may also be another INTEND with its own goal; that is, the way to achieve one goal may be to plan to achieve another goal. An example is the knowledge that a person can get someone to like him by getting that person to pity him. In this case, the PLAN for the goal to be liked is a further INTEND with a GOAL of being pitied.

Figure 2.4 illustrates how the planning knowledge related to making a bank deposit might be represented in the CLM network. The schema represents the knowledge that a goal to prevent money from being lost is associated with a plan to deposit the money in a bank.

```
(INTEND-PROTECT-MONEY is-a INTEND
    (planner ?P)
    (goal PREVENT
        (object LOSE
            (object (MONEY ?M))))
    (plan DEPOSIT-IN-BANK
        (object ?M)))
```

Figure 2.4: Portion of Schema for Depositing Money

Rather than satisfying her goal through an act of her own, a planner may choose to get another person to perform an act that will satisfy the goal on her behalf. Thus the planner's original goal may lead in turn to an agency goal, a goal that someone else perform an act to satisfy the planner's original goal. In the CLM model the actual goal of the planner in such cases is that the agent *wants* to perform the desired act since this is what the planner has more direct influence on. An agency goal may be satisfied in various ways, the most important of which involve speech acts such as requests and threats. Figure 2.5 shows how AGENCY is represented in the CLM model. AGENCY is a kind of INTEND for which the GOAL is that an ASSIGNEE UNDERTAKE an ASSIGNMENT on behalf of the PLANNER. In this form the schema is not actually usable because it does not specify its PLAN. In Chapter 3 we will see how different types of request schemas fill in the PLAN role.

```
(AGENCY is-a INTEND
    (assignee (HUMAN ?X))
    (assignment (ACT ?Y))
    (goal UNDERTAKE
        (actor ?X)
        (object ?Y)))
```

Figure 2.5: Schema for AGENCY

2.2. A Localized Connectionist Implementation of Schemas

2.2.1. Connectionist Models

Most computational models, including those making use of schemas, are examples of symbolic approaches based to a large extent on the operation of conventional digital computers. These models are characterized by the manipulation of symbols, which are entities with inherent semantic content, and usually by serial processing. A typical symbolic model (Newell, 1980) has a memory composed of symbol structures such as schemas, a control, a set of primitive operators which take symbols as inputs and yield other symbols as outputs (and may also have other effects), and input and output contacts with the outside world. Symbolic models are expressed in high-level languages, including logical formalisms such as predicate calculus, rule-based formalisms such as production systems, and programming languages such as LISP. These approaches have been quite successful at simulating some of the high-level properties of natural language processing, problem solving, and memory organization.

However, there are good reasons for questioning the computer metaphor for human cognition. Feldman and Ballard (1982) have pointed out the striking differences in the processing speeds of the basic units of the brain and a conventional computer. They argue that the only way the brain can carry out the complex behaviors that it does given the sluggishness of its neurons is for it to operate in a massively parallel fashion, quite unlike a computer. At the same time, some of the pervasive aspects of human cognitive processing are simply not handled well by symbolic methods. These include the effect of priming on processing; robustness, the ability to process even impoverished input and to fill in the missing portions of an incomplete input pattern; the shading of concepts into one another; the ability to rapidly integrate a range of knowledge sources in perception and problem solving; and "graceful degradation", the ability of the system to survive the destruction of portions of its memory.

In response to these problems with the traditional approach, researchers from cognitive psychology and computer science have in the past few years been developing (or reviving) a novel way of viewing and modelling human memory and cognitive processes. This approach, generally referred to as connectionism, posits a memory consisting of units modeled to a certain extent on neurons. Unlike the symbols of more conventional models, these units get their semantic significance entirely from the connections they have to other similar units. These connections are weighted and may be either excitatory or inhibitory. Processing involves the activation of portions of this network in response to inputs and the parallel spread of the activation along the connections in proportion to their weights, with inhibitory connections inhibiting the spread. After some time the network settles into a stable configuration of activation which represents the system's response to the input.

Connectionist models are clearly better candidates than symbolic models for low-level processes such as vision and speech recognition. They are able to respond to a variety of inputs simultaneously in a way that resembles human processing and to respond appropriately when the input is incomplete or ill-formed. What is less clear is the usefulness of these models for higher-level processes such as parsing and playing chess. One possibility is that both connectionist and symbolic processing occur in cognition; the question that arises then is what the interface between the two might look like. A more attractive possibility, at least from the point of view of uniformity, is that higher-level processes can also be handled in connectionist terms.

The usual approach in designing connectionist models is to start from the bottom up, that is, to let the networks themselves develop their own representations of a domain given a set of primitive features. The representations that result in such cases are distributed; concepts do not appear at particular places in memory. An alternative, and complementary, approach is to begin with the types of representations that have been found by symbolic models to support basic cognitive processes and attempt to implement these representations in a connectionist network. The resulting representations are localized; each concept has a single network node associated with it. The present model is an example of the latter type.

2.2.2. Nodes and Connections

Memory in the CLM model is an example of a semantic network, a network which represents the knowledge embodied in schemas. It differs from other semantic networks in being composed of a smaller set of primitive entities, enabling it to make use of a content-free spreading activation mechanism for processing.

Memory consists of a network of nodes joined by pairs of connections, one for each direction. Each node is actually a simple processing unit, which computes a current

activation level by summing its inputs and, when this level passes its threshold, sends activation to other nodes along its output connections. Each connection has an associated weight which determines the amount of activation that is spread along it. Weights may be positive or negative. Details of spreading activation are discussed in Chapter 4.

Each schema and instance consists of a subnetwork of memory. The head and roles are represented by separate, connected nodes, and the is-a and role-value relationships are also realized as connection pairs. Figure 2.6 shows a portion of the representation of the TRIAL schema using the graphical notation described in Section 1.2.2. Note how merged roles are used to represent the between-role relationships that require variables in the notation used in previous figures. The ACTOR node for EPISODE4, the sentencing of the defendant, is merged with the PRESIDING-JUDGE node, representing the fact that these refer to the same person for any instantiation of the schema. Figure 2.8 below presents another example of a merged node.

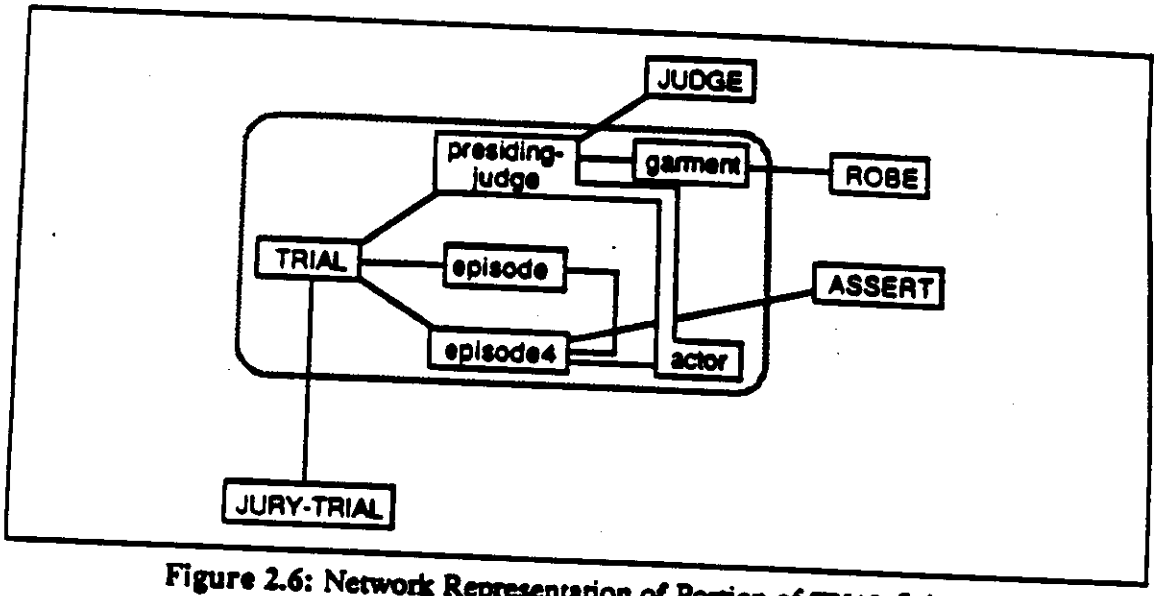

Figure 2.6: Network Representation of Portion of TRIAL Schema

When there is information to be associated with the role of a schema or instance, a separate node is needed to represent this role. That is, it is necessary to distinguish a high-level role such as ACTOR as it appears in the ACT schema from its instantiation in a lower-level schema such as TRANSFER-OF-CONTROL. Like the is-a, role-value, and head-role relations, the relation between the ACTOR of ACT and the ACTOR of TRANSFER-OF-CONTROL is represented by a pair of connections. Figure 2.7 illustrates this type of relation for three levels of the hierarchy. Note that there are alternative ways of referring to roles such as TRANSFERRER. Calling the TRANSFERRER the "ACTOR of TRANSFER-OF-CONTROL" emphasizes the fact that it shares all of the properties that ACTORs have. Again, however, it is not the particular name chosen that matters, but rather the connections that a node has to other nodes. In general I will follow the convention of referring to roles by very general names such as ACTOR and OBJECT.

Now we can see how merged roles get their meaning. A single role at one level of the is-a hierarchy is connected to two separate roles at a higher level. Figure 2.8 illustrates the example of the ACTOR and OBJECT of a SUICIDE.

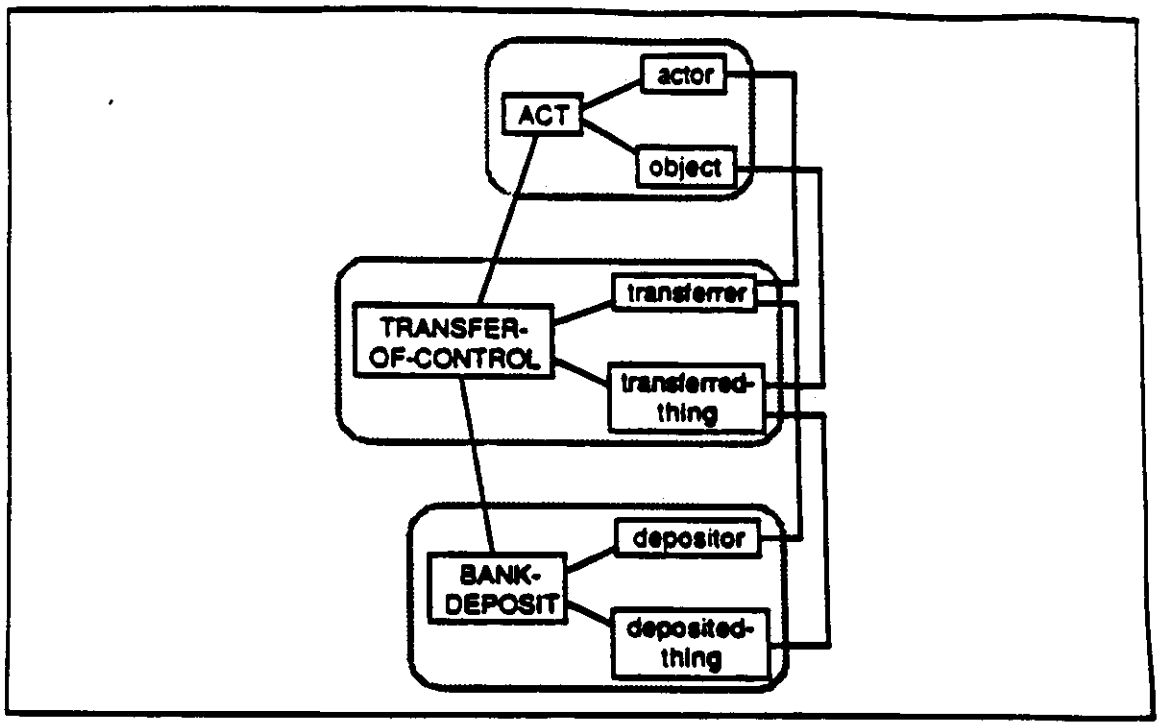

Figure 2.7: Role Hierarchies

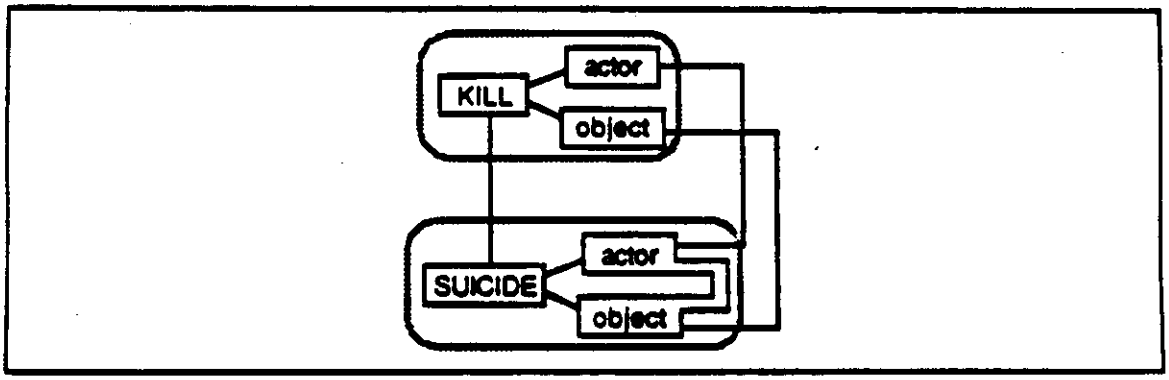

Figure 2.8: Merged Roles

Merged roles are used in two ways. If, for a given instance of killing, the ACTOR and the OBJECT are the same person, the system will automatically access the SUICIDE schema as activation converges on the merged role from two different sources. The selection of the SUICIDE schema allows the system to make use of other information in this schema (and not shown in the figure), for example, information about typical reasons for suicides. On the other hand, if it is known that a particular event is an instance of the SUICIDE schema and that a particular person is the OBJECT of that instance, the system will automatically determine that that person is also the ACTOR because activation spreads from the merged node to both KILL:ACTOR and KILL:OBJECT.

2.2.3. Inhibitory Connections

We have seen how subtypes of a schema are represented in the model; each subtype is a schema joined by a connection to the supertype schema. It will also be useful to represent the notion of mutually exclusive subtypes. For example, the system may have knowledge of various types of trials. We would like to be able to represent the knowledge that a trial may be a civil trial or a criminal trial but not both, that is, that civil trials are distinct from criminal trials. Exclusivity is also a property we want to associate with sets of instances. We would like the system to know, for example, that if the person John married is Mary, then she can't be Sue. Roles within a schema can also be related in this way. For example, among the possible constituents in a Japanese clause are a topic, a subject, and a direct object. The system needs to know that while both subjects and direct objects can be topics—that is, the same constituent may fill two roles at once—subjects cannot be direct objects.

To represent exclusivity, pairs of inhibitory connections, that is, connections with negative weights, are used. For example, the nodes CIVIL-TRIAL and CRIMINAL-TRIAL are joined by a pair of inhibitory connections. In most cases, sets of mutually inhibitory nodes group into what are called **winner-take-all (WTA) networks** (Feldman & Ballard, 1982). During processing, if one or more members of one of these networks become activated, then one of them will eventually dominate over the others, in effect winning all of the activation for itself.

In the CLM memory is-a hierarchies are represented using combinations of excitatory and inhibitory connections. Nodes at different levels of the hierarchy are joined by excitatory connections, while nodes at the same level tend to be connected by inhibitory connections. These relationships are easiest to see for biological taxonomies. Figure 2.9 illustrates some of the connections involved in representing the hierarchy under the ANIMAL schema. Inhibitory connections are denoted by fuzzy lines. Four WTA networks are illustrated, one for general categories of animals, one for species of birds, one for birds classified by gender, and one for individual birds.

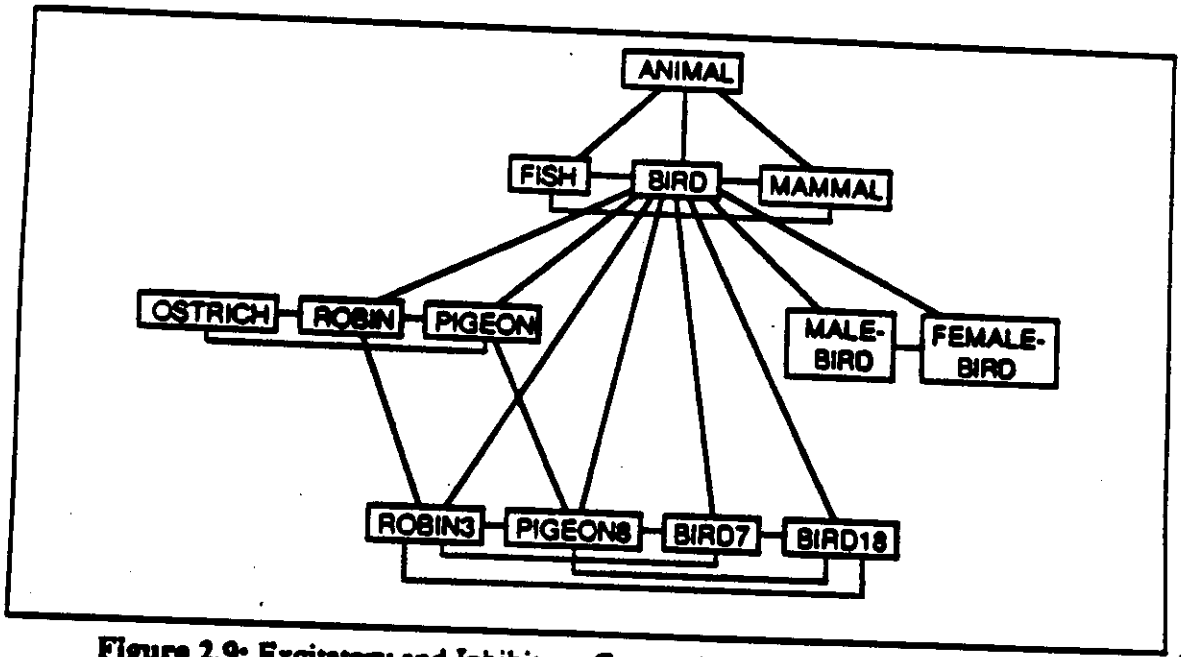

Figure 2.9: Excitatory and Inhibitory Connections in the ANIMAL Hierarchy

2.2.4. Connection Weights

The links connecting nodes in the network represent associations between concepts, but obviously concepts have degrees or strengths of association. Consider the TRIAL schema again. Presented with the word *trial*, a person would probably be much more likely to think of the judge than of the bailiff and more likely to think of the reading of the verdict than of the attendees' standing when the judge enters the courtroom. We find the same variation with roles of instances. In a particular trial the prosecuting attorney may have been more prominent than the defendant, and the sentencing of the defendant may have been more noteworthy than the testimony of the witnesses.

Thus strength of association may be thought of as accessibility of one node, given the other. Another way to view the notion is in terms of likelihood or confidence. A trial normally has both a judge and a defense attorney, but the accused may defend himself, in which case there is no defense attorney. Thus given a trial, a person may have more confidence that there is a judge than that there is a defense attorney.

Strength of association also applies to is-a relationships. One basic characteristic of this relationship is that there is greater accessibility in one direction than in the other. Given a particular trial, we expect to access general knowledge about trials, but thinking about trials in general does not usually cause us to remember all of the individual trials that we know about. Given a particular species of bird, one is quite likely to think of the general BIRD schema, while the reverse process is much less likely. There is also variation from one is-a relationship to another. Anyone who has studied elementary biology will know that people are mammals, but we are much less likely to access general mammalian properties when thinking about people than when thinking about dogs. A person may know of any number of species of birds, but given the concept BIRD, some subtypes are much more likely than others to be accessed.

Strength of association varies similarly for the role-value relation. A person that the system is quite familiar with will fill many different roles. For example, a person may know of John as Mary's father, Bill's boss, somebody who likes science fiction, and somebody who once ate two pizzas at a sitting, but given the instance JOHN, these roles will probably not be equally accessible to the person.

In the CLM model, connections have associated weights between -1 and 1 to reflect the strength of association between the connected nodes. There are default weights associated with the connection pairs for the various relation types. In CHIE connection weights may also be assigned explicitly within the procedures that create schemas (see Chapter 8 and Appendix B). Figure 2.10 shows some examples of connection weights for knowledge about trials. In this figure pairs of connections, usually denoted by single lines, are indicated by pairs of arrows, with numbers on the arrows denoting the weights. TRIAL18 sends more activation to TRIAL than TRIAL does to TRIAL18. PRESIDING-JUDGE sends more activation to TRIAL than DEFENSE-ATTORNEY does, and TRIAL activates PRESIDING-JUDGE more strongly than DEFENSE-ATTORNEY.

A question that always arises when numbers are added to a system such as this is how the numbers are arrived at in a principled way. In connectionist systems which learn (e.g., McClelland & Kawamoto, 1986; McClelland & Rumelhart, 1986), this is not an issue because it is the general mechanisms in the model itself which assign the weights. In the present case, however, as in most other localized models, the weights must be specified. In this regard, it should be emphasized that no claims are being made about the particular values assigned to connections. The point being made is that there is a need for a means of indicating relative strengths of association between nodes in the network. In selecting particular values for connection weights in the program, there are two considerations to be made, just as there are for other decisions concerning the particular

structure of the network. First, what makes intuitive sense? Does it seem plausible, for example, that the judge is associated more strongly with a trial than the bailiff is? A preferable alternative here would be experiments to measure with some precision the relative strengths of association. Second, what works? What set of connection weights allows the model to perform as it is supposed to? Obviously, by itself this criterion would amount to an ad hoc approach. Therefore, it is important to justify the relative link weights chosen on the grounds of psychological plausibility.

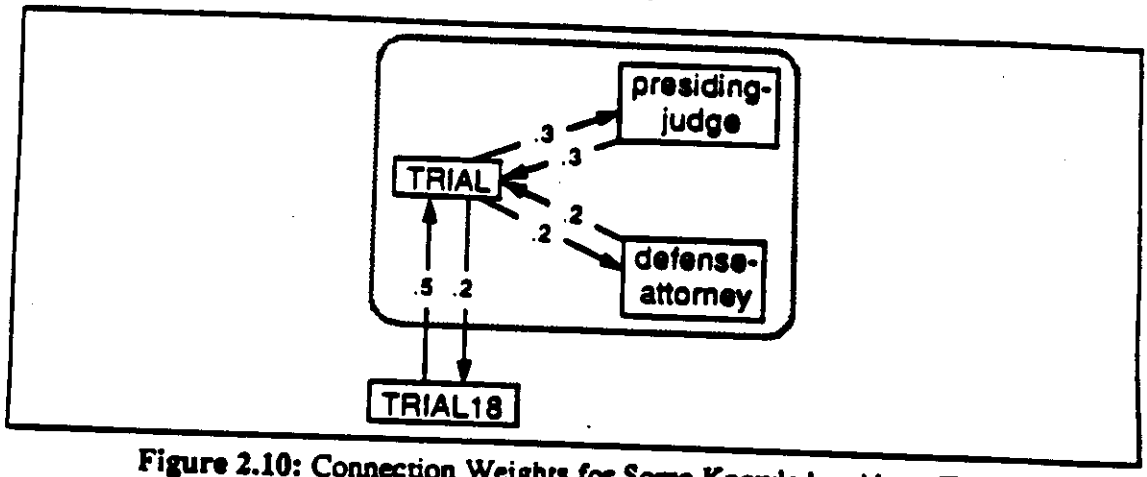

Figure 2.10: Connection Weights for Some Knowledge About Trials

2.3. Flexibility of Representation

The flexibility of human language as well as many of the differences between languages can be attributed informally to the potential to view a notion in more than one way. This potential is central to language generation, which involves the process of "transforming" a given input representation into a form that enables it to be conveyed in the target language. Many strictly symbolic representational formats have what amounts to a "standard" way of representing a particular notion and are relatively inflexible when it comes to alternative views. With such an approach one is forced to make ad hoc decisions about which representational possibility is most basic, and it is impossible to represent directly the meanings and functions of many lexical items and structures.

The CLM approach to this issue is that the formalism should be flexible enough to represent in a straightforward fashion any concept which is expressable by a lexical item or surface structure in a human language. Alternate views of a notion are treated as separate concepts, with no priority assigned to any particular one. In this section I discuss two areas in which flexibility is desirable, the representation of attributes such as gender and the possibility of equivalences among the roles of a single concept.

2.3.1. Attributes

Consider first the representation of an attribute such as nationality or biological gender. A natural way of treating these is as roles of concepts. For example, GENDER would be a role in the ANIMAL schema which could take the values FEMALE or MALE, and NATIONALITY would be a role in the HUMAN schema which could take values such as CANADIAN and SOVIET. This possibility corresponds to the use of an adjective like *female* or *Canadian*. Another alternative, however, is to have separate schemas representing

entities with particular values of the attributes. For example, we might have schemas for FEMALE-ANIMAL, MARE, CANADIAN-PERSON, and SOVIET-BALLET-DANCER. This possibility corresponds to the use of nouns like *woman*, *mare*, *Canadian*, and *Soviet*. There is no point in arguing about which of these representations is right. Both need to be part of the system because both are ways of thinking about gender and nationality, and both are reflected in ways of referring to these notions.

Figure 2.11 shows how some of the concepts involved in the gender example would be related in the CLM network.[6] GENDER appears first at the ANIMAL level of the hierarchy. Not shown of course are the features which define the genders. Note that MARE has two types, HORSE and FEMALE-ANIMAL. What is important is that, given a particular instance of an animal, it is possible to access both views of gender for purposes of generation.

```
(ANIMAL is-a ORGANISM
    (gender))

(FEMALE-ANIMAL is-a ANIMAL
    (gender FEMALE))

(HORSE is-a ANIMAL)

(MARE is-a (HORSE FEMALE-ANIMAL))
```

Figure 2.11: Alternate Views of Gender

2.3.2. Role Correspondences

A further type of representational flexibility concerns the possibility of a single concept playing several different roles within a schema. This fundamental feature of human cognition is behind analogy and metaphor, and it provides the basis for theories of semantic case (e.g., Fillmore, 1968; Foley & Van Valin, 1984; Langacker, 1987; Schank & Abelson, 1977). The claims of these theories amount to associations of specific roles with a small set of abstract roles. For example, in an act of transferring control, the TRANSFERRER might be assigned to an ACTOR and the TRANSFERRED-THING to an OBJECT role. As we have already seen, however, there is no reason to stop with two levels of the hierarchy. A specialization of TRANSFER-OF-CONTROL such as the DEPOSIT-IN-BANK schema described above will have roles corresponding to TRANSFERRER/ACTOR and TRANSFERRED-THING/OBJECT. These relationships are represented in the CLM network as shown in Figure 2.7.

More interesting cases are those in which there are sets of mutually distinct roles within a given schema and more than one possible mapping between the roles in the sets. Jacobs (1985b, pp. 56ff.) discusses the example of acts of giving and taking, which are specializations, or "views", of the TRANSFER-OF-CONTROL schema described above. For such acts, there are, in addition to the OBJECT, that is, the thing transferred, the mutually distinct sets SOURCE/RECIPIENT and ACTIVE-PARTICIPANT/PASSIVE-PARTICIPANT. When the transfer is viewed as giving, the source is the active-participant and the recipient the passive-participant. When the act is viewed as taking, the reverse mappings hold. For

[6] I will continue to use the schema notation in figures when it is more convenient. It should be remembered, however, that what it represents is a network structure of the type described in section 2.2.

particular subtypes of transfer acts, the giving and taking views and mappings also hold. A commercial transaction, for example, may be viewed as either selling or buying, depending on whether the merchant or the consumer is perceived as the active participant. Figure 2.12 shows the way in which these relationships are represented in the CLM model. A WTA network encodes the fact that a transfer event cannot be viewed as both giving and taking at the same time. For the sake of readability, the nodes and links associated with TAKE are highlighted with thick lines.

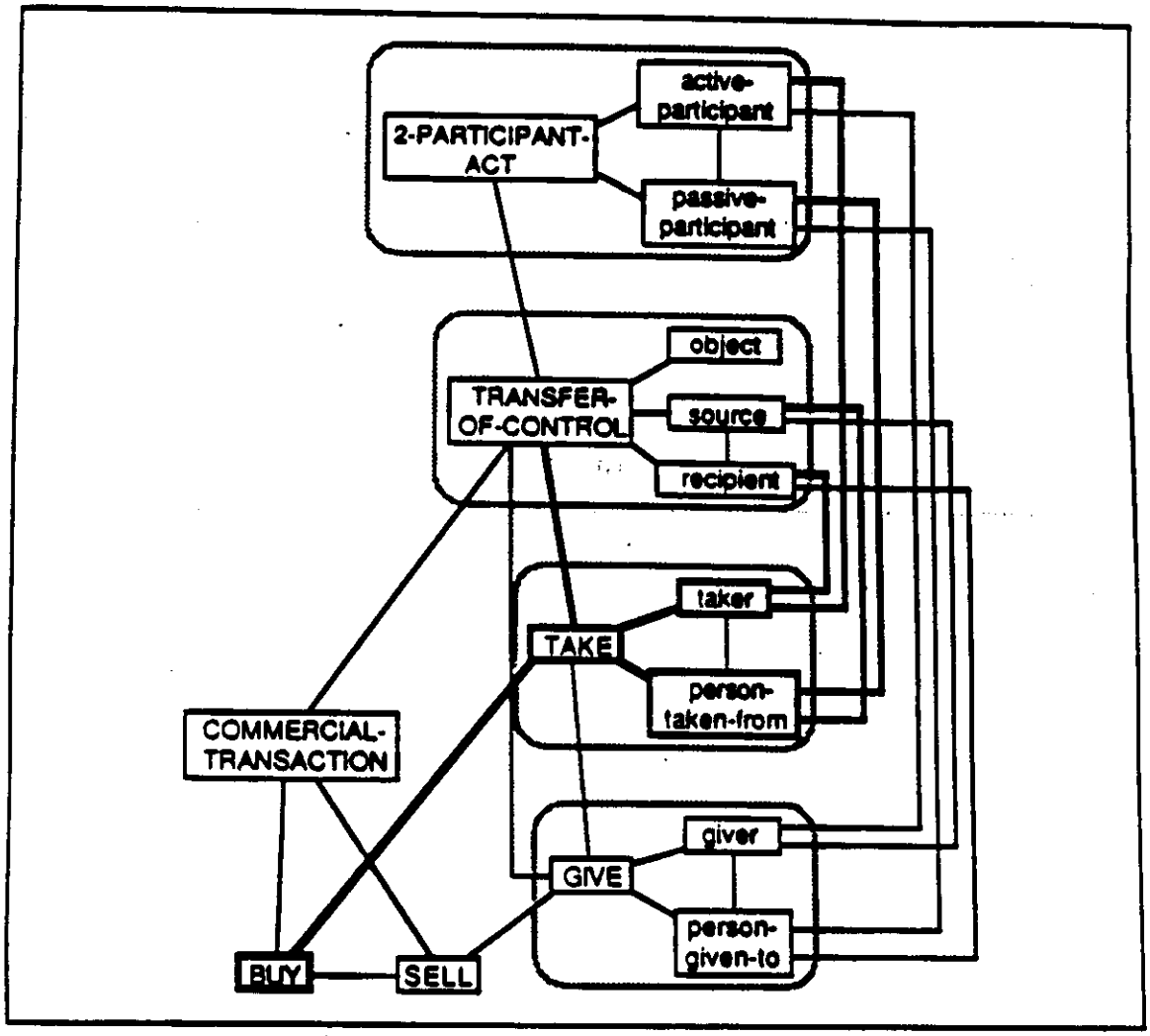

Figure 2.12: Representation of GIVE and TAKE

In sum, the present approach to the representation of roles is flexible in three ways. First, it permits any number of levels of abstraction in the role hierarchy. Second, it allows a particular role, GIVER, for example, to correspond to more than one other role. Third, as noted in Section 2.2.4, the strength of head-role and role-value relations can smoothly vary.

2.4. Summary

In this chapter I have argued that both schema-based approaches and connectionist approaches to memory organization and access exhibit features that seem to be basic to human cognition. I have described a representation formalism consisting of a localized connectionist implementation of a schema memory. Like other schema-based frameworks, the model incorporates two fundamental types of organization, constituency and generalization hierarchies. In addition, the CLM approach has the advantage of allowing more flexibility in representations than many other schema-based models. This flexibility derives in particular from the network structure of the memory. Unlike other semantic network approaches, memory in the CLM model is composed of nodes functioning as simple processing units and joined by weighted connections of a single type. This scheme permits memory to make use of connectionist processing mechanisms, as discussed in Chapter 4. In the next chapter I show how the features of the formalism prove useful in the representation of knowledge about language use.

Chapter 3

Representing Linguistic Knowledge

3.1. The Organization of Linguistic Memory

3.1.1. Language as Action

In the last chapter we looked at the representation of acts and the goals behind them. Consider now how linguistic behavior can be incorporated into this framework. An act such as the depositing of money in a bank starts with a goal that the act can satisfy, in this case a goal to protect the money or increase its value. The uttering of a sentence, say,*could you turn down the TV?*, is also an act which is made in response to a goal, in this case a goal that the hearer undertake to turn down the television in question. An act itself expands into a set of subacts, in the bank deposit example, getting to the bank, filling out a deposit slip, etc. Likewise in the language case, there is an expansion into subacts: the utterance of words such as *could* and phrases such as *the TV*. The subacts themselves are expanded further; for example, the filling out of the deposit slip includes the picking up of a pen, the writing of numbers in the appropriate spaces on the form, etc. In some cases this expansion involves the further consideration of a goal and a plan to satisfy it. For example, getting to the bank involves a goal to be at the bank and a plan to accomplish this goal, for example, walking there. Similarly, in the language case the expansions may be directly realizable or may require reference to a goal and a plan to achieve it. The word *could*, for example, expands directly into a series of phonemes, while the final noun phrase in the sentence requires that the speaker come up with a way to realize the goal to have the speaker be aware of a particular television.

The planning and executing of an action, linguistic or non-linguistic, involves the selection of schemas from memory. Some schemas are of the INTEND type; they associate goals with plans in the form of particular acts or sequences of acts. For language the most important category of this type I will call **generalized illocutions** (GIs). These associate goals to affect a hearer in some way with plans in the form of utterances. For example, there is a GI which associates a goal to get the hearer to undertake an act on behalf of the speaker with a plan to produce an utterance in the form of a command. Other schemas are subtypes of the ACT schema; they specify the features of the participants in particular types of acts. Linguistic schemas which are subtypes of ACT I will call **generalized utterances** (GUs). They associate the different features of an utterance–its form, speaker, hearer, and context–with its content, the fact or object it is intended to refer to. For example, there is a GU associating the concept of TELEVISION (the content) with a noun phrase having the word *TV* as its head noun. Other GUs specify the structure of syntactic patterns such as passive clauses.

Usually the generation of a sentence involves the selection of at least one GI and one GU. For example, in generating the sentence *sit down*, the system would select the GI specifying an imperative utterance to satisfy an agency goal and the GU which associates the concept of sitting down with a clause that has *sit* as main verb and *down* as a post-verbal adverbial particle.

Because schemas are non-directional associations (implemented as nodes joined by connections in both directions), the schemas used in language generation are also usable from the analysis direction. For example, a GI can be accessed from it PLAN role (the utterance realizing the speaker's goal) as well as from its GOAL role. The selection of a GI in language analysis on the basis of the form of the utterance would allow the hearer to infer the goal of the speaker.

3.1.2. Representation of Sequence

Both language analysis and language generation require attention to the temporal properties of constituent sequences. For example, during generation all of the elements of a constituent need to be uttered before any element of a following constituent is uttered, and in the same fashion during analysis a new element should be recognized as either belonging to the current constituent or signalling the beginning of a new constituent.

A considerable body of research has been dedicated to problems of this sort, in particular for parsing, and a variety of computational techniques have been developed to handle sequencing along with other aspects of analysis. These include augmented transition networks (Woods, 1980), definite clause grammars (Pereira & Warren, 1980), and demon-based parsers (Dyer, 1983). The goal in the present model has been to deal with these problems within the connectionist framework and to represent the necessary relationships in such a way that they are usable in both generation and analysis. The basic idea is that some degree of sequentiality is imposed on the otherwise parallel processing through the use of winner-take-all networks and connections representing sequencing relations. In addition, each constituent or phrasal unit needs to be represented as a pair of nodes, one for the start and the other for the end of the sequence. The details of how this works will be spelled out in Chapter 4. For now it will suffice to note that the use of start and end nodes, sequence connections, and winner-take-all networks allows one to represent that a particular constituent is the first or last in a sequence and that one or more constituents follow a given constituent. In this chapter I will simply abbreviate each of these relations and will treat constituents and phrases as though they were single nodes rather than start-end pairs.

3.2. Sources of Utterances

Behind most utterances is a goal to influence the hearer in some way. There are two general types of goals, those involving a change in the hearer's knowledge, beliefs, goals, or overt behavior and those involving directing the attention of the hearer towards a particular entity. The former type, **illocutionary acts** (Searle, 1969), usually result in clauses. The latter, which include **reference acts** and **vocative acts**, normally result in noun phrases. The CLM model currently handles illocutionary acts and vocative acts.

In some situations utterances arise out of a set of contextual features rather than as a result of an explict speaker goal. I will discuss this type under the heading of **context-engendered utterances.**

3.2.1. Illocutionary Acts

Generalizations about the ways in which utterances can achieve goals involving hearer's beliefs, goals, or behavior make up the network of generalized illocutions (GIs). GIs vary in terms of specificity; the more specific a GI, the more restricted the context in which it applies and the form in which it is realized.

In its present state, the model has knowledge about directives and assertives. I will discuss mainly the former, which are in any case more interesting. A directive is an illocutionary act with an agency goal, an intention that the hearer (the ASSIGNEE) undertake some act (the ASSIGNMENT) on the speaker's behalf. There is a variety of ways of achieving such a goal, not all of them linguistic. For example, if the goal is to get an unwelcome guest to leave, a host may try yawning or looking bored. Linguistic means to achieve agency goals include requesting, ordering, and persuading.

3.2.1.1. Context-Specific Conventions

A basic assumption of this thesis is that speakers have a good deal of relatively specific knowledge about language use. If so, we would expect knowledge about how directives are realized in particular contexts. Figure 3.1 shows a simple example representing the knowledge that in military contexts an imperative utterance is used to get someone with lower rank to do something.

```
(MILITARY-COMMAND is-a AGENCY
    (planner (MILITARY ?P)
        (rank ?PR))
    (assignee (MILITARY ?H)
        (rank ?HR))
    (assignment ?A)
    (plan IMPERATIVE
        (content ?A)
        (speaker ?P)
        (hearer ?H))
    (constraint1 RANKING
        (higher-rank ?PR)
        (lower-rank ?HR)))
```

Figure 3.1: GI for Military Commands

IMPERATIVE, like other types of utterances, is treated as a kind of act, which is what is required as the value of the PLAN role. This GI includes a CONSTRAINT, a fact which should be true at the time an act is executed. In this case, there is a RANKING relationship between the PLANNER (speaker) and ASSIGNEE (hearer) which must be satisfied. Note how the schema provides the information needed for the selection of a verb lexical schema: the CONTENT of the utterance PLAN is specified as equivalent to the ASSIGNMENT that the hearer is to undertake. Thus if the hearer is to peel some potatoes, the peeling is treated as the content of the utterance. The lexical schema (generalized utterance) for the verb *peel* has as its content the concept of peeling, and on this basis the GI can be selected for the utterance being generated.

In some contexts particular wordings are appropriate. An example is a request like *I'll have an X*, used in the context of ordering in a restaurant. In this case the GI has in its CONTEXT role the restaurant script. This allows the SPEAKER and HEARER roles to be related in the proper way; the speaker is the customer and the hearer the waiter in the instantiation of the restaurant script that fills the CONTEXT role. This GI is shown in Figure 3.2. The PLAN is specified as a declarative utterance with MOOD (subject + auxiliary) realized as *I'll* and verb as *have* and with the direct object referring to the OBJECT of the act that the speaker wants done, that is, the food being ordered. Of course, this GI specifies only one of the ways in which ordering is done.

The GIs in Figures 3.1 and 3.2 are conventions: they represent knowledge about the use of particular forms for particular functions, without including the reasons these particular forms are appropriate. Thus the MILITARY-COMMAND schema does not indicate why it is that an imperative is a reasonable way to issue a military command, and the REQUEST:I'LL-HAVE schema does not indicate what *I'll* and *have* have to do with ordering food in a restaurant. Note that these conventions are not really idioms in the usual sense, however, because the literal meanings of the expressions are not inconsistent with their functions here. The existence of these conventions means rather that these expressions are

acceptable forms for satisfying these functions, whereas other possible expressions are not. For example, on the basis of general knowledge about directives, one might conclude that a military order could be executed just as well using the expression *I want you to VERB*. The fact that there is no GI specifying this expression for military orders means that the imperative is preferred.

```
(REQUEST:I'LL-HAVE is-a AGENCY
    (context RESTAURANT-SCRIPT
        (customer ?C)
        (waiter ?W))
    (planner ?C)
    (assignee ?W)
    (assigment PHYSICAL-TRANSFER
        (actor ?W)
        (object (FOOD ?F))
        (destination ?C))
    (plan DECLARATIVE
        (speaker ?C)
        (hearer ?W)
        (mood "I'LL")
        (verb "HAVE")
        (direct-object ()
            (content ?F))))
```

Figure 3.2: A GI for Ordering in a Restaurant

Speakers probably have such context-specific knowledge for many situations. For example, a speaker might have the knowledge that a way to make a routine request of her husband is to use the *could you VERB?* request form. While such knowledge has the advantage of saving precious processing time, it is of little use when novel situations arise. A system also needs more general knowledge, knowledge which applies across contexts.

3.2.1.2. Cross-Contextual Conventions

Many considerations may be involved in selecting an appropriate form for a directive. These can be divided into two general types. On the one hand, there may be factors which the speaker perceives as potential obstacles to the success of the directive (Gibbs, 1986), that is, factors which might prevent compliance on the part of the hearer. One is the hearer's belief regarding his ability to perform the requested act. For example, the hearer might not have, or might not believe he has, the physical strength to carry out the act, or might not have access to some object which is necessary for the act. Another consideration is the hearer's willingness to perform the requested act, based on his belief that the potential benefits outweigh the detriments. Benefits to the hearer include the release of an obligation, the avoidance of punishment, and benefits to the speaker, if the speaker is someone the hearer cares about. Detriments include the time and effort required to perform the act. Gibbs (1986) argues that speakers select request forms which focus on the perceived obstacles to the hearer's compliance. For example, in a situation where the speaker has doubts about the hearer's ability to comply, a request with *can you*, *could you*, or *are you able* would be likely.

Figure 3.3 shows an example of a GI representing a convention concerning a type of obstacle. This schema encodes the knowledge that a speaker who wants a hearer to perform an act, but feels that the hearer's lack of ability may impede the success of the

request, can convey his intention by uttering a *could you* question regarding the hearer's ability to perform the requested act. An OBSTACLE role is meant to represent a state which the speaker feels might interfere with the success of the plan.

```
(REQUEST/ABILITY-OBSTACLE:COULD-YOU is-a AGENCY
    (planner ?P)
    (assignee ?H)
    (assignment ?A)
    (plan YES-NO-QUESTION
        (speaker ?P)
        (hearer ?H)
        (modal "COULD")
        (subject "YOU")
        (content ?A
            (modality ABILITY)))
    (obstacle1 ABLE
        (actor ?H)
        (object ?A)))
```

Figure 3.3: GI for Request Involving Doubt About Hearer's Ability

Another class of considerations, discussed by Brown and Levinson (1978), relate more to the social appropriateness of the forms used. Three independent factors combine to form what Brown and Levinson call the "weightiness" of a directive (or, in fact, any speech act), that is, the degree to which the act is seen as threatening the speaker's and/or the hearer's "face". These are the social distance between the speaker and hearer, the relative power of the speaker and hearer, and the extent to which the requested act is likely to be viewed as an imposition by the hearer. In Brown and Levinson's analysis, weightiness is the sum of the values associated with these three variables, and it is this sum which determines the extent of the politeness strategies which the speaker will use to minimize the threat of the act. That is, the choice of form depends on the combination of social distance, power, and imposition but not on the particular contributions of each factor. For example, a somewhat polite request form like *I was wondering if you could* is appropriate if either the social distance is great, or the hearer has considerable power over the speaker, or the requested act is thought to be a significant imposition. When two of the factors have relatively high values, an even more polite form (or combination of forms) is called for.

Figure 3.4 shows a GI for a convention involving a politeness level. This schema represents the knowledge that one way to perform a request, when weightiness is high, is to produce an utterance with *I was wondering if you could*, where the *if*-clause refers to the hearer's ability to perform the requested act.

3.2.1.3. General Knowledge About Illocutionary Acts

Speakers probably also have more general knowledge about the strategies used in performing speech acts and the kinds of inferences that hearers are able to make. Like the more specific knowledge, this knowledge can relate to either obstacles or to politeness. For example, the GI in Figure 3.3 can be seen as a specialization of a more general schema encoding the knowledge that the hearer's inability to perform the requested act would be an obstacle to the success of the request and that a speaker wants to know if there are any such obstacles.

Figure 3.5 shows a relatively general GI for requests involving politeness. This schema encodes the knowledge that one way to make a polite request (one with relatively high weightiness) is to let the hearer know what is desired but to convey pessimism about the chance of success.

```
(REQUEST/POLITE:I-WAS-WONDERING is-a AGENCY
    (planner ?P)
    (assignee ?H)
    (assignment ?A)
    (weightiness HIGH)
    (plan DECLARATIVE
        (speaker ?P)
        (hearer ?H)
        (subject "I")
        (auxiliary "WAS")
        (verb "WONDERING")
        (direct-object NOMINALIZED-YES-NO-QUESTION
            (conjunction "IF")
            (subject "YOU")
            (modal "COULD")
            (content ?A
                (modality ABILITY)))))
```

Figure 3.4: GI for Polite Request Using *I was wondering if you could*

```
(REQUEST/PESSIMISTIC is-a AGENCY
    (planner ?P)
    (assignee ?H)
    (assignment ?A)
    (weightiness MEDIUM-TO-HIGH)
    (plan ()
        (expansion1 ILLOCUTE        ;;get hearer to believe
            (goal BELIEVE           ;;  that speaker intends
                (believer ?H)       ;;  hearer to undertake
                (object INTEND      ;;  assignment
                    (planner ?P)
                    (goal UNDERTAKE
                        (actor ?H)
                        (object ?A)))))
        (expansion2 ILLOCUTE        ;;get hearer to believe
            (goal BELIEVE           ;;  that speaker doubts
                (believer ?H)       ;;  that hearer will
                (object DOUBT       ;;  undertake assignment
                    (actor ?P)
                    (object UNDERTAKE
                        (actor ?H)
                        (object ?A)))))))
```

Figure 3.5: GI for Polite Request Conveying Pessimism

The PLAN for this GI has no type specified (hence the "()") because the speaker's goal can be realized in more than one way. Two speech acts are involved in the PLAN, both intended to affect the hearer's beliefs. One seeks to cause the hearer to believe that the speaker intends for the hearer to undertake the requested act, and the other attempts to get the hearer to believe that the speaker has doubts about whether the hearer will undertake the act. In the schema both of the subacts are specified only as ILLOCUTE; that is, they may be finally realized as any type of speech act which satisfies the given goals. Examples include relatively straightforward instantiations of the GI such as *I'd like you to proofread this for me, but I guess that's too much to ask* and those in which the first speech act is missing and inferrable such as *I don't suppose you could proofread this for me* and *I was wondering if you could proofread this for me.* This type of knowledge about a very general strategy for directives would probably only rarely be used to actually generate a sentence; a particular convention such as the GI shown in Figure 3.4 would be selected instead. On the other hand, this GI would come in handy in the comprehension of novel utterances of this type.

It is doubtful that speakers or hearers would ever need knowledge more general than that shown in Figure 3.5. Thus the CLM model does not include a GI for directives in general. For assertives, however, there is the general GI shown in Figure 3.6. This specifies that a way to get someone to believe a fact is to produce a declarative utterance with that fact as content.

```
(ASSERTIVE is-a INTEND
    (planner ?P)
    (goal BELIEVE
        (believer ?B)
        (object ?O))
    (plan DECLARATIVE
        (planner ?P)
        (hearer ?B)
        (content ?O)))
```

Figure 3.6: GI for Assertives

3.2.2. Vocatives

Noun phrases have two sorts of functions. Referential NPs such as *the judge* and *the defendant* in the sentence *the judge sentenced the defendant to hard labor* function to direct the hearer's attention to particular entities. Other NPs are desgined to get the attention of the hearer. An example is *your honor* in the sentence *your honor, may I question the witness?*. Expressions such as *your honor* are known as vocatives. As Zwicky (1974) has pointed out, vocatives appropriate for getting the attention of a particular person may not be identical to the type of NPs that would be appropriate in reference to the same person. In English one may use the word *madam* as a polite vocative for a woman, but this word is hardly appropriate for referring to the woman, unless she happens to run a bordello. Like directives, vocatives also depend strongly on the nature of the speaker-hearer relationship and on the social setting. A judge is addressed with *your honor* by a lawyer in the courtroom, but the same lawyer may address the judge by his first name outside the courtroom if they are friends.

It is a straightforward matter to represent the use of vocatives within the framework developed thus far. Figure 3.7 shows the schema for the expression *your honor*.

```
(VOCATIVE:YOUR-HONOR is-a INTEND
    (planner ?P)
    (context TRIAL
        (presiding-judge ?J))
    (goal ATTEND
        (actor ?J)
        (destination ?P))
    (plan NP
        (content ?J)
        (det "YOUR")
        (noun "HONOR")))
```

Figure 3.7: Schema for Vocative *your honor*

This, and other vocative schemas, is a subtype of the general INTEND schema. Its goal is for the hearer, who is the PRESIDING-JUDGE of the TRIAL which forms the CONTEXT for the performance of the act, to ATTEND to the speaker (the PLANNER). The PLAN for achieving this goal is to produce an NP which refers to the HEARER, that is, the PRESIDING-JUDGE, and takes the explicit form *your honor*.

3.2.3. Context-Driven Utterances

Thus far we have assumed that all utterances arise out of speaker goals. Goals themselves originate in the perception of external or internal events or states by the speaker. Thus a goal to inform the hearer of some fact may result from a question from the hearer regarding that fact, or, more precisely, from the recognition by the speaker that the hearer wants to know some aspect of the fact. A goal to get the hearer to close a window may be motivated by the speaker's perception that she is cold, together with the knowledge that the window in question is open and general knowledge about how the temperature of a room can be affected by the opening and closing of the room's windows.

There are cases, however, in which the perception of an event or state, together with a set of appropriate contextual features, seems to lead directly to an utterance, or at least to a situation in which a form is available to the speaker. In such cases the utterance need not be seen as satisfying a goal of the speaker. Consider the generation of the expression *speak of the devil*. This expression is appropriate when, in a context in which a particular person is under discussion, one of the conversants sees or hears that person. This conversant might then utter *speak of the devil*, but without the necessity of an intervening stage in which a search is made for a means of satisfying a particular goal. It is true that the expression can perform a function of sorts, that of making the hearer aware of a particular kind of coincidence, but the speaker need not explicitly want to achieve this before the expression makes itself available to her. I will refer to such utterances as **context-driven**. In order for context-driven generation to work, a phrase such as *speak of the devil* needs to be associated in memory directly with the event which leads to it.

In the CLM model events may have other events as CONSEQUENCEs. A subtype of the PERCEIVE schema brings together the features necessary for the utterance of *speak of the devil* and has such an utterance as a CONSEQUENCE. This schema is shown in Figure 3.8.

```
(PERCEIVE:SPEAK-OF-THE-DEVIL is-a PERCEIVE
    (perceiver ?P)
    (context CONVERSATION
        (topic ?T))
    (object (HUMAN ?T))
    (consequence *SPEAK
        (speaker ?P)
        (subject ZERO)          ; represents null subject
        (verb "SPEAK")
        (matter PP
            (prep "OF")
            (pp-obj *DEVIL
                (det "THE")
                (noun "DEVIL")))))
```

Figure 3.8: Schema Specifying Conditions for Utterance of *speak of the devil*

The schema has as its CONTEXT a CONVERSATION and as its OBJECT (the perceived thing) a person who is the TOPIC of the conversation. The utterance that is the CONSEQUENCE of this set of circumstances is spoken by the PERCEIVER and is realized as the sequence of words *speak of the devil*. This sequence is represented as an instance of the generalized utterance *SPEAK. *SPEAK provides information about the meaning of the word *speak*, but this information can be inherited so does not need to appear in the schema in Figure 3.8. The SUBJECT of the clause is specified as ZERO; this is an unpredictable peculiarity of this idiom. The MATTER constituent of the pattern represents the thing spoken about. It is a prepositional phrase with an NP as its object which is represented as an instance of the generalized utterance *DEVIL. *DEVIL provides information about the meaning of the word *devil*. Note that the literal meaning of *speak of the devil* does not appear in the schema in Figure 3.8, but the representation of the pattern in terms of the *SPEAK and *DEVIL schemas enables the speaker or hearer to infer the origin of the expression from the semantic information appearing in these other schemas.

Other examples of context-engendered utterances are greetings, apologies, and similar conventional expressions. It is possible, however, to view much of what takes place in conversational contexts as a combination of the top-down, goal-driven kind of processing and the bottom-up, context-driven kind of processing. In an argument setting, for example, a speaker has a constantly active goal to defend her argument against attack. This has the effect of priming various ways of achieving this defense, including some relatively high-level plans and some relatively specific expressions such as *you're putting words in my mouth* and *that's exactly my point*. When the other speaker says something that satisfies other conditions for one of these expressions or high-level plans, the relevant schema receives activation from this contextual change and may then be able to fire.

3.3. Utterances

The schemas for illocutionary acts and vocatives are subtypes of INTEND; they associate a communicative goal with a plan. The plan for each of these schemas takes the form of an act I will refer to as an utterance. An utterance is realized in turn as a combination of words or morphemes. The general schema for utterances is shown in Figure 3.9.

```
(UTTER ACT
     (speaker HUMAN)
     (hearer HUMAN)
     (time)
     (context)
     (content CONCEPT)
     (expansions))
```

Figure 3.9: General Schema for Utterances

Three of its roles, the ACTOR (referred to as SPEAKER for convenience), TIME, and CONTEXT, are inherited from higher-level schemas. In addition, utterances are characterized by a HEARER and, most importantly, some semantic CONTENT. The EXPANSIONs of an utterance, indicated with braces in the figure because there is generally more than one, are either words or other utterances. Most of the system's knowledge about language consists of schemas in the hierarchy under this general schema. These I call **generalized utterances** (GUs). Of course most of them will not need to make reference to all of the roles shown in Figure 3.9. In particular, the deictic roles, SPEAKER, HEARER, TIME, and CONTEXT, will not generally be needed, but they are available when they are.

3.3.1. Hierarchy of Generalized Utterances

GUs are arranged in a generalization hierarchy which brings together the aspects of language that are normally dealt with under the headings of syntax and semantics. Figure 3.10 illustrates the basic structure of the hierarchy of GUs. Missing in the figure are the roles which specify the semantics and morphosyntax for each schema.

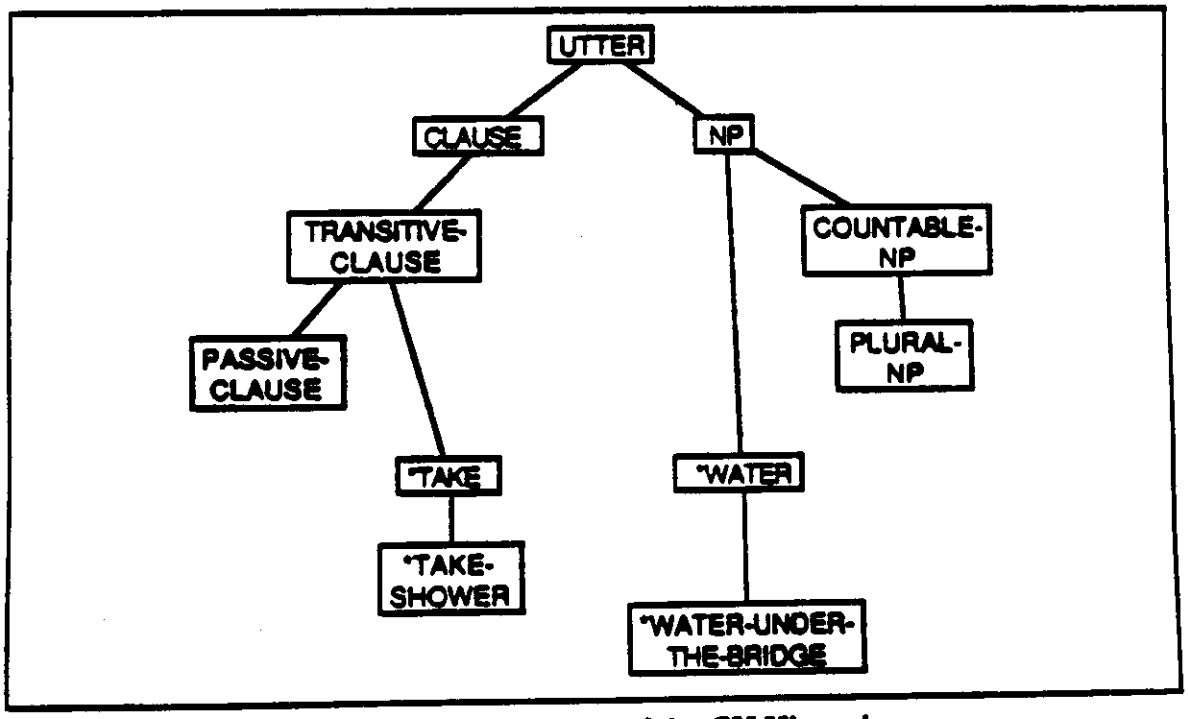

Figure 3.10: Structure of the GU Hierarchy

The most general, or abstract, GUs are those specifying syntactic structures such as NPs and clauses. Further down the hierarchy come GUs for more specific structures such as imperative clauses, transitive clauses, relative clauses, proper names, countable NPs, and plural NPs. At this point we begin to see grammatical morphemes being filled in in some of the patterns. For example, the GU for plural NPs specifies that the morpheme *-s* is suffixed to the head noun. Further down the hierarchy begins what is normally thought of as the lexicon. Here we have GUs for the verb *take* and the noun *water*. Note that names of lexical GUs are preceded by an asterisk to distinguish them from concept names. Still further down are GUs specifying entire phrases, with or without some variable slots. There are, for example, GUs for the expressions*take a shower* and *tap water*. At the bottom of the hierarchy are GUs for completely specified patterns such as *how's it going* and *water under the bridge*.

An important aspect of the organization of the hierarchy of GUs is that the schemas which correspond to lexical entries represent entire phrases rather than single words (Jacobs, 1986a; Wilensky & Arens, 1980; Zernik & Dyer, forthcoming). Thus the schema for *water* is a generalization about NPs which have the word *water* as their head noun rather than about the word *water* in isolation. Likewise the GU for *take* is a generalization about clauses with the verb *take* as their head. This feature of the representation allows lexical GUs to be related directly both to syntactic structures and to idioms: lexical GUs represent specializations of more general structures and generalizations of more specific idioms.

It should also be noted that terms like clause and noun phrase are not being used in their ordinary senses. The difference is that here these are more than structures; each has associated content, speaker, and hearer roles, though these may not always be explicitly referred to in particular schemas.

3.3.2. Utterance Content

An utterance in the sense I am using the term is a meaningful act; that is, it has associated with it a concept, an instance of a concept, or a set of features, which I will call its content. The immediate goal behind an utterance could be seen as the activation in the hearer's memory of the content node or nodes. However, the speaker never has to be concerned with this goal because utterances are usually produced as plans to satisfy higher-level pragmatic goals, that is, the goals contained in schemas for illocutionary acts and vocatives. Figure 3.11 shows an example of a noun GU. The content of an NP with the word *water* as head noun is specified as an instance of the non-linguistic concept WATER.

```
(*WATER is-a NP
    (content WATER)
    (noun "WATER"))
```

Figure 3.11: GU for *water*

During generation GUs at the same level of the hierarchy compete with one another for selection. For example, in the generation of an NP to refer to a particular entity, only one noun GU must be chosen. In order to represent this competition, the CONTENT roles of lexical GUs which are similar in meaning form a WTA network. A fragment of such a WTA network appears in Figure 3.12. The figure shows only the inhibitory connection joining the CONTENT roles of two schemas. These roles are also connected to the CONTENT roles for other potentially competing GUs such as those for *BEVERAGE and *JUICE.

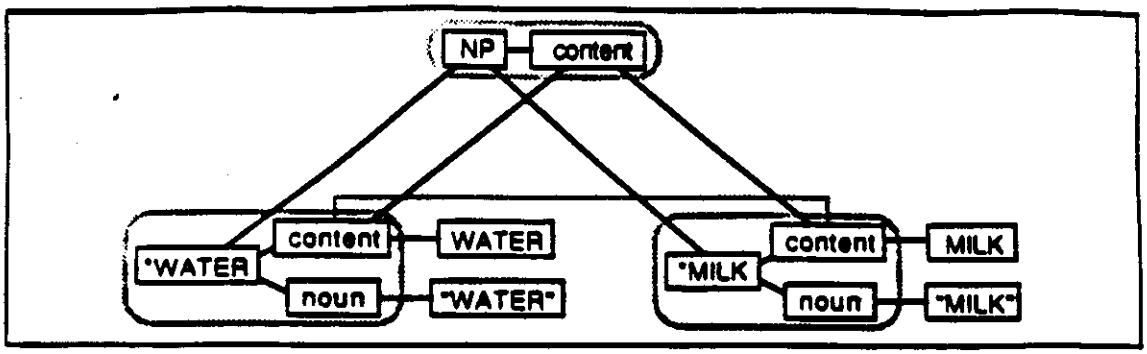

Figure 3.12: Inhibition Between CONTENT Roles of NP GUs

Analogous to the competition between CONTENT roles in generation is competition between the head nodes of GUs in analysis. This competition prevents more than one sense of a particular word from being selected for a given instance of the word. Figure 3.13 shows an example.

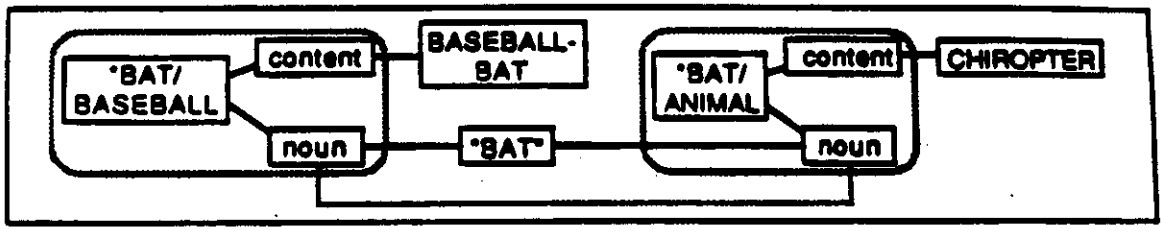

Figure 3.13: Inhibition Between GUs for Homonyms

Two GUs have the word *bat* as head NOUN, one referring to the baseball implement, the other to the animal (chiropter), so nodes for the whole schemas inhibit one another, forming a WTA network. Such WTA networks are required only for homonyms, that is, sets of GUs whose head NOUNs, VERBs, or ADJECTIVEs are identical.

3.3.3. Deictic Information in Generalized Utterances

In many cases there is a deictic, or indexical, element in the content (Anderson & Keenan, 1985); that is, the content must make reference to one of the elements of the speaking context itself. This presents no problem for the CLM framework because the SPEAKER, HEARER, TIME, and CONTEXT roles are implicitly part of all GUs. First and second person pronouns are a simple example. Figure 3.14 shows the schema for first person singular pronouns. The merged role indicates that the CONTENT and SPEAKER are the same entity; that is, the NP refers to whoever the current speaker is.

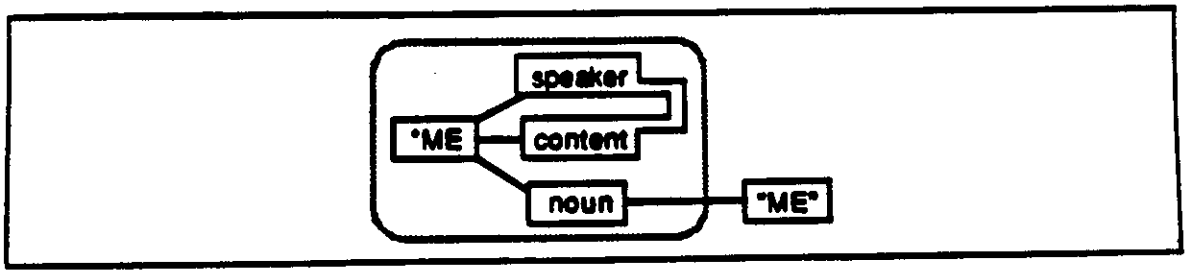

Figure 3.14: GU for *me*

Other examples are the GUs for the verbs *bring* and *come*, which are appropriate when the direction of the movement referred to is toward the speaker or hearer. A portion of the GU for *come* is shown in Figure 3.15.

```
(*COME is-a CLAUSE
    (speaker ()
         (location ?SL))
    (hearer ()
         (location ?HL))
    (content SELF-PHYSICAL-TRANSFER
        (destination (?SL ?HL)))
    (verb "COME"))
```

Figure 3.15: GU for *come*

In this schema, the CONTENT:DESTINATION role has connections to both the SPEAKER:LOCATION and HEARER:LOCATION roles. These connections represents the fact that the DESTINATION of the transfer for a clause with verb *come* is equivalent to either the current LOCATION of the SPEAKER or the HEARER. This relationship is what causes this GU to be selected over *GO, which is the default GU for reference to SELF-PHYSICAL-TRANSFER.

The ability to make reference to deictic elements in lexical GUs is also useful for defining lexical items as suitable for particular kinds of hearers. Figure 3.16 gives two examples.

```
(*BUNNY is-a NP
    (content RABBIT)
    (hearer CHILD)
    (noun "BUNNY"))

(*GLOTTAL-STOP is-a NP
    (content GLOTTAL-STOP)
    (hearer LINGUIST)
    (adj-modifier "GLOTTAL")
    (noun "STOP"))
```

Figure 3.16: Two GUs Specifying Appropriateness for Types of Hearers

The *BUNNY GU, the schema for NPs with the word *bunny* as their head noun, has as its CONTENT the concept of RABBIT and the restriction that the HEARER be a CHILD. Note that, like all other schemas, this represents a prototype. That is, there is nothing preventing the *BUNNY GU from being used when the hearer is not a child, but the impression given in such a case is that the hearer is somehow being treated as a child. The second GU in the figure is the schema for the phrase *glottal stop*. This refers to the concept of GLOTTAL-STOP and stipulates that the HEARER be a LINGUIST.

Reference to the hearer is also necessary in the GU for the third person pronouns *he*, *she*, *it*, and *they*. These are appropriate when the referent is presumed to be currently active in the hearer's consciousness. The hearer becomes conscious of an entity either because it has just been referred to in the discourse or because he is currently or has just

been observing or listening to it. Thus a speaker may say, "he's an old friend of mine" just after she has said, "you know the guy who called while I was out?" or when the hearer is apparently looking at the person spoken about. Otherwise another form of reference is appropriate: "John Smith is an old friend of mine".

In the CLM model, there is a type of state called CONSCIOUS-OF to represent this condition on the use of third person pronouns. The GU for these pronouns is shown in Figure 3.17. It includes the contraint that the HEARER be currently conscious of the CONTENT of the NP.

```
(3RD-PERSON-PRONOUN is-a NP
    (hearer ?H)
    (content ?C)
    (constraint1 CONSCIOUS-OF
        (conscious ?H)
        (object ?R)))
```

Figure 3.17: Entry for Focused Reference

Of course, giving a name to a concept such as 'being in someone's consciousness' doesn't solve the problem of how this concept gets used. The system needs to be able to infer that such a fact is true of the hearer and the referent under certain circumstances. One way in which the hearer becomes conscious of an entity is through reference to it. That is, the production of an NP has as an effect the fact that the hearer (and the speaker) are now conscious of the thing referred to. The general GU for NP needs to include this information so that it can be used in determining when a pronoun is appropriate in later NPs. Figure 3.18 shows the portion of the NP schema containing this information.

```
(NP is-a UTTER
    (content ?C)
    (speaker ?S)
    (hearer ?H)
    ...
    (effect CONSCIOUS-OF
        (conscious (?S ?H))
        (object ?C)))
```

Figure 3.18: Portion of GU for NPs Showing Consciousness Effect

The EFFECT in this GU is an instance of the concept of CONSCIOUS-OF with both the SPEAKER and the HEARER as the conscious entities and the CONTENT of the NP the OBJECT of their conscious state. When an NP is produced, the presence of this EFFECT in the NP entry leads to the priming of a set of nodes representing the fact that the hearer is conscious of the entity. If another NP referring to the same entity follows soon, this priming will be strong enough to establish that the CONSTRAINT in the 3RD-PERSON-PRONOUN entry holds, leading to the selection of that GU and the generation of a pronoun.

3.3.4. Constituency and Semantic Composition

Like other acts that are not primitives, utterances expand into subacts, that is, constituents. Constituents in turn may expand into further utterances or be realized as

words. In the CLM model, words are treated as primitives, but it is a straightforward matter to take this recursive expansion into the domain of phonology and phonetics.

Phrasal constituents, including both obligatory and optional ones, are defined at the level of the basic phrase types, clause, noun phrase, adjective phrase, and prepositional phrase. A portion of the GU for English NPs is shown in Figure 3.19.

```
(NP is-a UTTER
    (content THING)
    (det ?D)
    (adj-modifier (ADJ-PHRASE ?A)
        (<- ?D))
    (noun ?N
        (<- ?D)
        (<- ?A))
    (relative-clause CLAUSE
        (<- ?N)))
```

Figure 3.19: Portion of GU for NPs Showing Constituents

Four constituents appear in the NP schema, DET, NOUN, ADJECTIVE-MODIFIER, and RELATIVE-CLAUSE. The symbol "<-" indicates a sequence relationship; the head NOUN of the NP, for example, is specified as following the DETERMINER (?D) and the ADJECTIVE-MODIFIER. Sequence relationships are discussed in detail in Chapter 6. Two of the constituents, ADJECTIVE-MODIFIER and RELATIVE-CLAUSE, are optional, though this fact is not indicated in the figure. Optional roles differ from obligatory ones in having lower weights on their input connections from the schema head.

Figure 3.20 shows a portion of the GU for English clauses. In addition to the SUBJECT and VERB, there is a set of optional constituents, each taking the form of a prepositional phrase. Three of these are shown here, LOCATIVE (*we chatted on the subway*), TEMPORAL (*we chatted on Tuesday evening*), and MATTER (*we chatted about the elections*). Other clause constituents are introduced in subtypes of the basic CLAUSE schema. DIRECT-OBJECT, for example, appears in the TRANSITIVE-CLAUSE schema.

```
(CLAUSE is-a UTTER
    (content FACT)
    (subject (NP ?S))
    (verb ?V
        (<- ?S))
    (locative PP)
    (temporal PP)
    (matter PP))
```

Figure 3.20: Portion of GU for Clauses Showing Constituents

An important type of information is missing in Figures 3.19 and 3.20, an indication of the semantic composition of the phrase types, that is, how the phrasal constituents relate semantically to the phrase as a whole. Consider first the clause. What we need is

something that tells us that the SUBJECT typically refers to the ACTOR of the clause CONTENT, that the LOCATIVE constituent refers to the LOCATION of the clause CONTENT, and so on. This kind of information is necessary for the temporary **role binding** that goes on in generation, the process by which clause constituents are associated with arguments of the predicate signalled by the clause itself. For example, in generating the sentence *Mary loves John*, the system must know that the ACTOR of the loving is to be referred to in the SUBJECT and the OBJECT of the loving in the DIRECT-OBJECT. In the CLM model, role binding information can appear at any level of the hierarchy of clause GUs. Some of it belongs in the general CLAUSE schema, for example, that concerned with the LOCATIVE constituent. Figure 3.21 shows how this knowledge is represented.

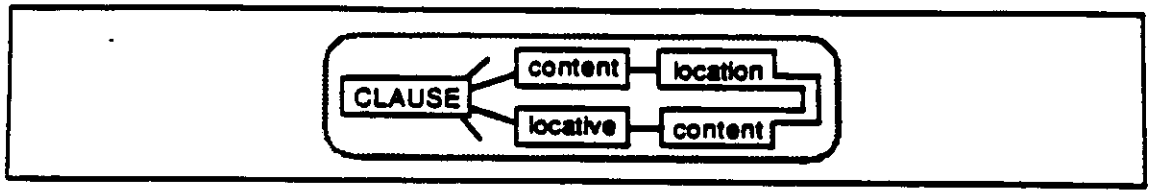

Figure 3.21: Role Binding Information for Clause LOCATIVE

The merged role represents the equivalence of the CONTENT of the LOCATIVE phrase and the LOCATION of the clause CONTENT. Thus in the sentence *we chatted on the subway*, the LOCATIVE constituent *on the subway* refers to the LOCATION of the act of chatting.

Other role binding information appears in lexical GUs. For example, the GU for *deposit* specifies associations for two clause constituents, the LOGICAL-SUBJECT and LOGICAL-DIRECT-OBJECT. These are the constituents which are realized as the SUBJECT and DIRECT-OBJECT respectively of an active clause and the (optional) AGENT and SUBJECT respectively of a passive clause. For example, *Mary* is the LOGICAL-SUBJECT and *$100* the LOGICAL-DIRECT-OBJECT in both of the following sentences:

(3.1a) *Mary deposited $100 in the bank.*
(3.1b) *$100 was deposited in the bank by Mary.*

The associations are shown in Figure 3.22.

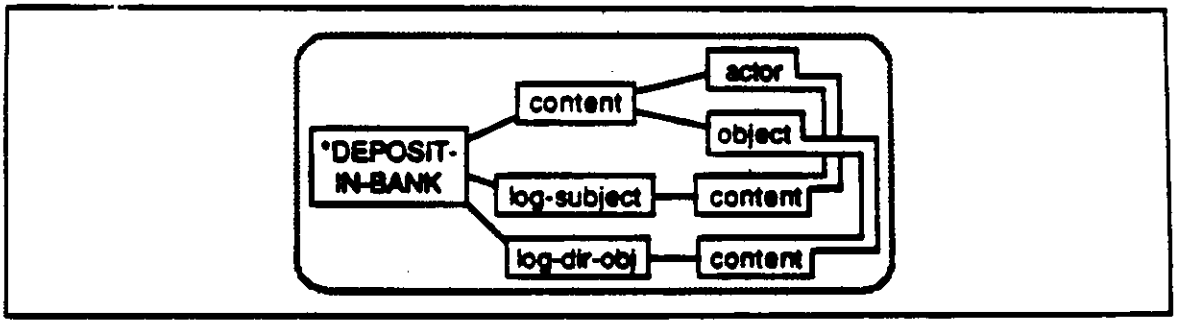

Figure 3.22: Role Binding Information for *deposit*

The two merged roles represent the fact that the LOGICAL-SUBJECT refers to the ACTOR and the LOGICAL-DIRECT-OBJECT to the OBJECT of the clause CONTENT. That is, in sentences (3.1), MARY, the ACTOR of the transfer, is referred to in the LOGICAL-SUBJECT, and $100, the OBJECT of the transfer, is referred to in the LOGICAL-DIRECT-OBJECT.

Role binding information for the LOGICAL-SUBJECT and LOGICAL-DIRECT-OBJECT is also found at a more general level, in the GU for TRANSITIVE-CLAUSE, where the CONTENT would be a general TRANSITIVE-EVENT, that is, any event with an ACTOR and an OBJECT. If the system did not have the necessary role binding information in the GU for a particular verb, the information at the level of TRANSITIVE-CLAUSE would be available.

Consider now the modification relationships in the noun phrase. In the CLM model adjective phrases have as their CONTENT facts of type ATTRIBUTE , which take a single ATTRIB-OBJECT argument. For example, in the phrase *the green box*, the adjective *green* is taken to refer to an instance of the concept GREEN with the box in question as its ATTRIB-OBJECT. The way this relationship is represented in the NP schema is shown in Figure 3.23. The ATTRIB-OBJECT of the CONTENT of each ADJECTIVE-PHRASE-MODIFIER is the same as the CONTENT (the referent) of the NP.

```
(NP is-a UTTER
    (content ?C)
    (adj-modifier ADJ-PHRASE
        (content ATTIBUTE
            (attrib-object ?C))))
```

Figure 3.23: Semantics of Adjective Modifiers in the NP GU

3.4. Summary

In this chapter I have outlined an approach to the representation of knowledge about language use. This knowledge is implemented using the same network primitives as are used for conceptual memory. Linguistic knowledge is hierarchically organized, not only in terms of constituency but also in terms of abstraction/generality. With respect to the first type of hierarchy, there is a lower level, that of utterances, at which linguistic knowledge is about how patterns are associated with particular concepts, and a higher level, that of illocutionary acts and vocatives, at which linguistic knowledge is about how particular utterance types can satisfy particular goals of the speaker. The abstraction hierarchies integrate schematic units of knowledge ranging from those specifying fixed lexical patterns to those specifying very general syntactic structures. The inheritance mechanism that is implicit in these hierarchies permits the sharing of semantic and morphosyntactic properties among GUs and also allows deictic features to be represented in a straightforward way in GUs.

The most important feature of the CLM approach to the representation of linguistic knowledge, however, is that it permits the use of a general spreading activation mechanism for all processing. This mechanism is the subject of the next two chapters.

Chapter 4

Language Generation: Schema Selection

4.1. Processing in General

In Chapers 2 and 3 we saw how knowledge is represented in the CLM model. In order for this knowledge to be useful, there must be a means of accessing appropriate pieces of knowledge and then taking actions based on what is found. In the CLM model a single spreading activation process is responsible for locating and making use of knowledge in memory. This process is completely domain-independent; that is, it may be used in the classification of a bird, given a set of features of the bird in question, or in the generation of a sentence, given the features comprising the goal behind the sentence. This chapter describes the algorithm for the spread of activation through the network and how it is used in accessing schemas of the types described in Chapter 3.

4.1.1. Spread of Activation

Each node in the network has at any given time an activation level varying from -1 to +1. When the activation of a node reaches its **threshold activation**, the node fires. A firing node sends activation along all of its output connections. The amount of activation spreading is equal to the weight on the connection from the firing node to the destination node. The weight on a connection may be high enough to cause the destination node to fire on the basis of activation along that connection alone; that is, the weight may be greater than the threshold of the destination node. Many connections representing upward is-a relations, for example, have weights of this type. In most cases, however, the destination node requires activation from more than one source to fire.

Figure 4.1 shows the state of a small network over a short stretch of time. In the first time interval, node A has just fired as a result of activation from a source not shown. In the second time interval shown in the figure, we see the spread of activation to the nodes connected to A. The connection to node D is high enough to permit it fire immediately, while node B is now activated to a level below its threshold, indicated by the hatched border pattern. Connection weights may also be negative. This relationship is then an inhibitory one because the negative activation spread lessens the likelihood of the destination node's firing. Inhibitory connections are indicated in the figures by fuzzy lines. When node D fires in the third time interval in the figure, it inhibits the activated node G. Negative activation on a node is indicated by a fuzzy border pattern.

Following firing, a node is inhibited for a length of time referred to as its **refractory period**. During this interval, the node's state is unaffected by inputs from other nodes. This type of inhibition is also characteristic of neurons which have fired (Kuffler & Nicholls, 1976). In the CLM model, inhibition during the refractory period prevents activation from spreading back along the path from which it originated and reverberating indefinitely. This inhibition is represented in the figures by a fuzzy node border. In the second time interval in Figure 4.1, node A is inhibited because it has just fired. Once its refractory period has passed, a node retains a small amount of positive activation and can be further activated from other nodes. This property is also a characteristic of neurons (Kuffler & Nicholls, 1976). In the fourth time interval in Figure 4.1, node A is no longer inhibited and displays residual activation. It can now fire again if it receives additional activation from another node. Note that priming results in two ways: when a node has received some activation, but not enough to fire (e.g., node B in the second time interval in Figure 4.1), and when a node has fired and its refractory period has passed (e.g., node A in the fourth time interval in Figure 4.1).

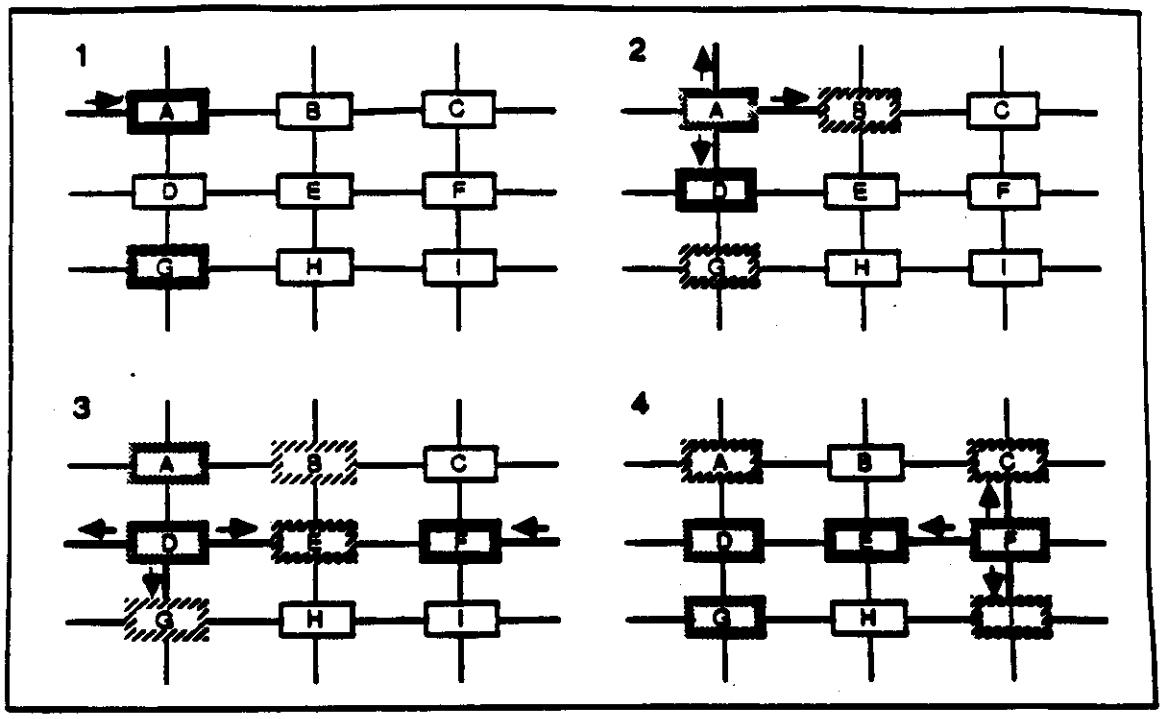

Figure 4.1: Firing, Spread of Activation, and Decay

The model also has a decay mechanism used to represent short-term forgetting and the importance of recency. The activation level of all nodes decreases at a rate which may vary from node to node. In Figure 4.1 note the changes in activation on node B between the second and fourth time intervals. Spacing differences in the border hatch pattern indicate different activation levels. The reduction of activation on this node is due to decay.

Since CHIE runs on an ordinary serial computer, the spread of activation must be broken down into discrete time steps. Ideally the time required for activation to traverse a connection would be allowed to take on a range of values, but this is complicated to simulate. I have found it necessary to make use of two propagation speeds. Fast connections are needed so that is-a hierarchies are traversed quickly. In CHIE four fast connections but only one slow connection can be traversed in a single time step.

During each time step three things happen (ignoring complications due to fast connection traversal).

1. Activation propagates from all of the nodes which have just fired.
2. Each node updates its activation level. This is the sum of its previous activation and the new inputs. If its activation exceeds its threshold, it fires.
3. The activation level of each node is multiplied by a decay factor (<1).

For more details see Chapter 9 and Appendix B.

4.1.2. Winner-Take-All Networks

In previous figures winner-take-all networks have been represented by sets of inhibitory connections joining the members of the networks. In fact, for each WTA

network there are two other nodes with special significance. The **WTA parent** is the node which sends activation to all members of the WTA network. For the ANIMAL example illustrated in Figure 2.9, the BIRD node is the WTA parent for the three WTA networks below it. When BIRD fires, all of the members of these networks receive some activation from it. In addition, each WTA network has a special node, the **WTA hub**, which is responsible for providing the additional activation that may be necessary for one of the network members to fire. WTA hub nodes fire under different circumstances than other nodes, as described below. The basic configuration of WTA networks is shown in the first diagram in Figure 4.2. Node P is the WTA parent, the node in the shape of a pentagon is the WTA hub, and nodes M1, M2, and M3 are the network members.

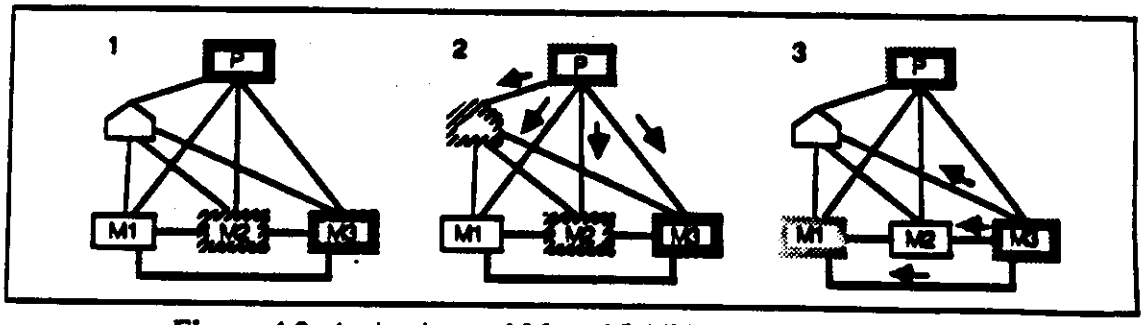

Figure 4.2: Activation and Mutual Inhibition in WTA Networks

The most important property of WTA networks is mutual inhibition among the network members. When one member node reaches its firing threshold, it inhibits all of the other members, preventing them from also firing. The WTA hub node is also inhibited when a member node fires. Figure 4.2 illustrates this basic process. The parent node P fires in the first time step, activating the three member nodes. Two of these already are primed as a result of activation from other sources. The additional activation from P is enough to allow node M3 to fire in the second time step, and it then inhibits M1, M2, and the hub node in the third time step. Fuzzy arrows in the figures indicate the spread of inhibition. Note that what determines which member node will fire is the total activation each node has received from all sources, including the WTA parent node. Thus different nodes will fire under different circumstances.

If, following the firing of the WTA parent, none of the WTA members has enough activation to fire, an interval is allowed to pass, and then the hub node fires. This "timing out" feature is what distinguishes WTA hubs from other nodes. The interval which must pass before a WTA hub fires may vary from node to node. This interval is meant to represent the hesitation that occurs when difficulty is experienced in reaching a decision. When it fires, the WTA hub sends activation to the network members, usually providing enough for the one which had the highest activation to fire. The firing element then inhibits the other elements in the normal fashion. This is the sense in which the "winner takes all". This possibility is shown in Figure 4.3.

The parent node P fires and sends activation to the members, but in this case none has enough priming to allow it fire immediately. In the fifth time step the hub node times out and fires, providing additional activation to the member nodes. Now M2 reaches its threshold, fires, and inhibits the other two member nodes. It is also possible that no node will achieve its threshold when the hub node fires. This represents situations in which there is not enough information for a decision to be reached.

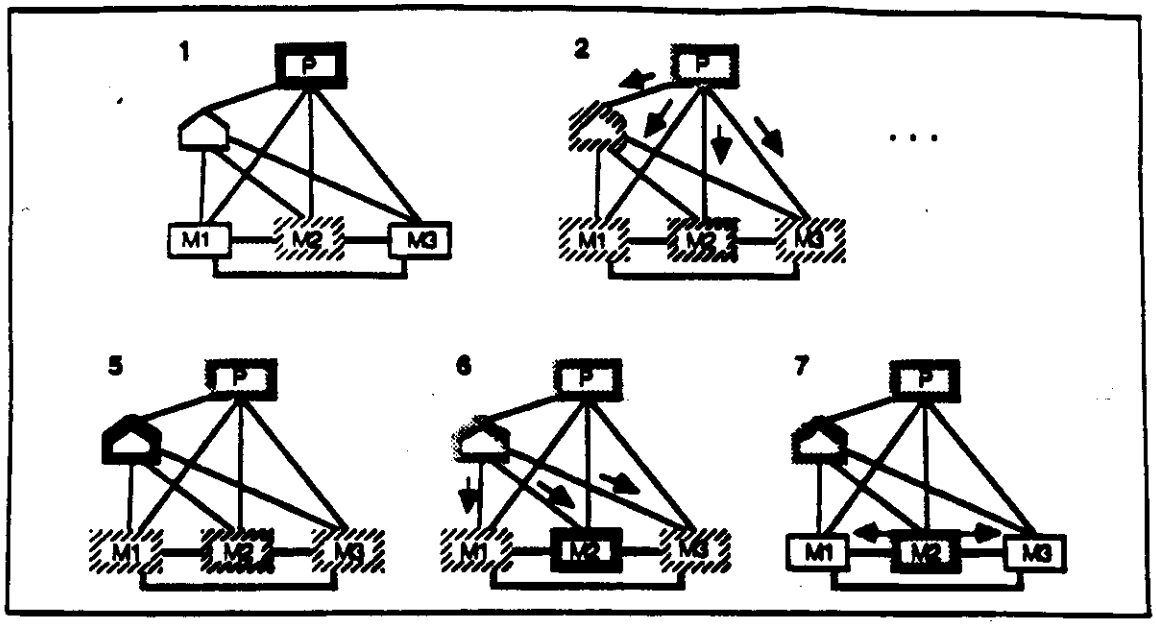

Figure 4.3: Role of the Hub Node in WTA Networks

4.1.3. Phases of Processing

Processing in the CLM model can be thought of as consisting of three phases, schema selection, role binding, and sequencing, though only the first is involved in every process. These three phases are not distinguished formally in the model; they are a consequence of the way in which nodes are connected to one another and the way in which spreading activation responds to these connections.

Schema selection amounts to classification. The entity represented by a set of input features is classified as an instance of the schema which is eventually selected. The input feature set is normally missing some crucial elements, and the selected schema allows the system to complete the feature pattern. Say the system is interested in recognizing birds, as a birdwatcher might. It begins with a set of features of a particular bird and on the basis of these features, selects a schema for a particular bird species if one can be found. This schema yields a species name such as *mockingbird* or *robin*. For language generation, the input features characterize a goal of the speaker, and the missing information is the linguistic pattern that can be used to achieve this goal. The schema that is selected in response to the input (a generalized illocution) provides the speaker with some or all of the missing information in the form of a general structure for the utterance to be generated and possibly some of the actual words.

Schema selection begins with the spread of activation from a set of firing input nodes. Each feature of the input is actually represented by two firing nodes, one for the role that the feature plays in the input structure and one for the value of the feature. In the bird recognition example, one feature might be represented by the firing of the nodes for HEAD-COLOR and BLACK. In the generation case, a feature might be represented by the firing of the nodes for GOAL and BELIEVE if the speaker's goal is that the hearer believe some fact. Activation spreads from the firing input nodes and converges on roles in schemas which agree with the features represented by the firing nodes. These candidate schemas compete against one another via a WTA network. The schema whose head node receives the most activation from its roles, that is, the one which is closest agreement with

the input features, is selected. Once the head node of the schema fires, activation spreads to roles of the schema which were not represented in the input. These normally provide the missing information sought by the system.

In the case of language processing, the initial schema selection usually does not suffice. This schema may in turn call for further schema selections to be made on the basis of particular elements of the input. For example, when a verb GU is selected, it specifies that the noun phrases in the positions in the clause refer to particular semantic roles in the input. This role binding information translates to a role binding process once the schema is selected. If John is the ACTOR of a particular event to be referred to, the system needs to associate JOHN with the SUBJECT position in the clause. Role binding works through the spread of activation beginning from the two associated roles, in this case the ACTOR and SUBJECT roles.

Finally, for processes resulting in actions taken by the system, such as language generation, there is the problem of turning the mainly parallel processing into sequential output. Sequencing makes use of connections representing sequencing relations and WTA networks to force the selection of one and only one constituent to fill a particular output position.

4.2. Basic Schema Selection

4.2.1. A Generalized Utterance

We will first consider the selection of a lexical GU for the generation of a noun phrase. We will assume that the speaker is attempting to generate a noun phrase referring to some pepper, for example, as part of the sentence *could you pass the pepper?*. The input at this point consists of a set of firing nodes representing this intent: the nodes for NP, the CONTENT of NP, and SUBSTANCE4 (see Figure 4.4), the pepper to be referred to. Note that the input concept has not been classified as an instance of the concept of BLACK-PEPPER. One simplifying assumption is made here: the issue of constituent sequencing is ignored. For the time being, it is assumed that the head noun of the NP can be uttered without concern for the constituents that must precede it.

Figure 4.4 shows the state of the network at the beginning of the process. SUSTANCE4, the input concept, is represented as an instance of the general concept SUBSTANCE. SUBSTANCE has many roles, only two of which are shown in the figure, USE and COLOR. SUBSTANCE4 has as values for these roles ADD-FLAVOR and BLACK respectively. Other features that might be activated in this context include TASTE, TEXTURE, and the usual LOCATION of the pepper. The *PEPPER GU has as its CONTENT an instance of SUBSTANCE which has as its USE the value ADD-FLAVOR and as its COLOR the value BLACK, that is, just the features specified for the input.

The initially firing nodes for the selection process are those representing the input concept and its USE and COLOR and those for the general NP GU and its CONTENT role. Figure 4.5 shows what happens during the first time step. Activation spreads down the is-a hierarchy from NP and NP:CONTENT. The CONTENT roles of all noun lexical GUs receive activation from NP:CONTENT, and GUs compete with one another through WTA networks. For *PEPPER the WTA network is indicated in the figures by the inhibitory connections

emanating from *PEPPER:CONTENT.[7] Activation also spreads from the nodes representing the input concept, resulting in the firing of SUBSTANCE, SUBSTANCE:USE, SUBSTANCE:COLOR, ADD-FLAVOR, and BLACK.

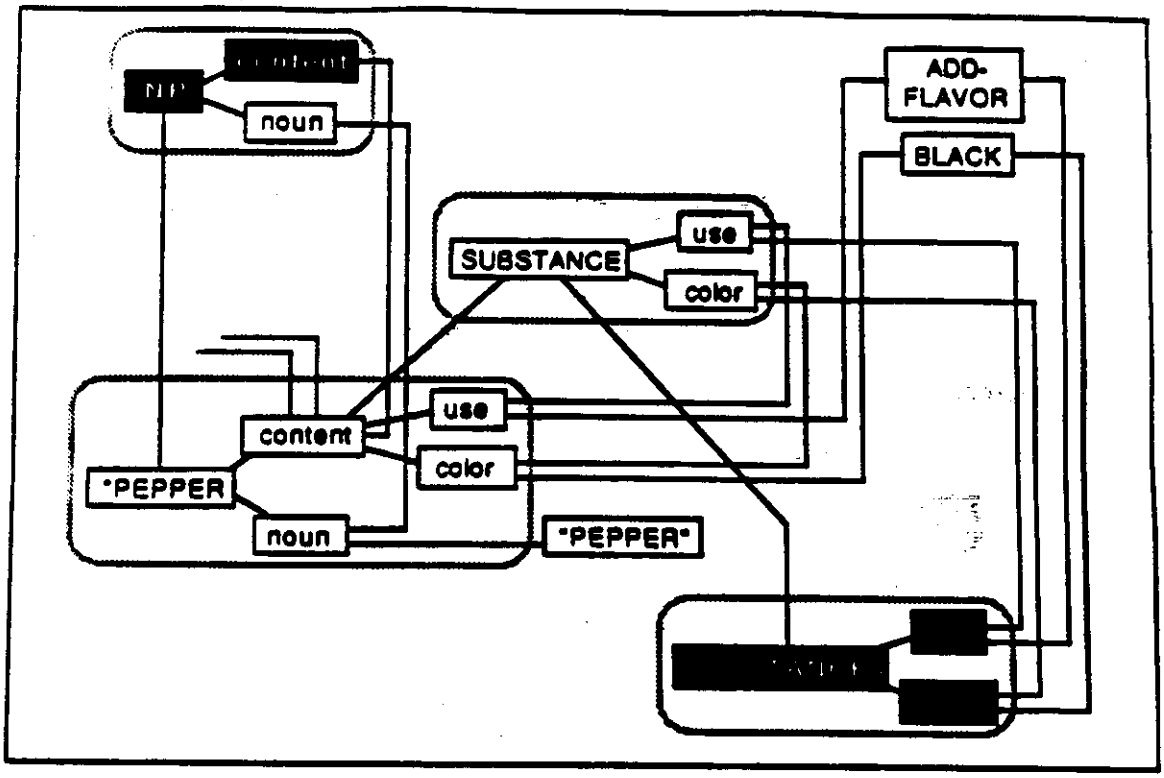

Figure 4.4: Lexical Selection for an NP (1)

In the next two time steps, activation spreads down from the nodes representing features of the input and results in the firing of *PEPPER:CONTENT. This process is shown in Figure 4.6. Activation first converges on the roles of *PEPPER:CONTENT, causing them to fire. This intersection of paths of activation is a basic mechanism in schema selection (Quillian, 1968), and intersections on the roles of GU CONTENT nodes are the usual means by which lexical GUs are accessed in generation. *PEPPER:CONTENT also receives activation from SUBSTANCE, but the weight on this input connection is small, and the node cannot yet fire. However, activation from the roles of *PEPPER:CONTENT during the next time step is enough to bring the node over its threshold, and it fires.

During the next three time steps, we see the selection of the *PEPPER GU, the inhibition of GU competitors, and the utterance of the word *pepper*. These steps are shown in Figure 4.7. The firing of *PEPPER:CONTENT results in the spread of inhibition to the CONTENT roles of competing GUs such as *SALT and *SUGAR, preventing these nodes from firing. Activation from *PEPPER:CONTENT also causes *PEPPER to fire, and this GU has now been "selected". The firing of *PEPPER leads in turn to the firing of *PEPPER:NOUN, which has been primed from the NOUN role in the NP schema.

[7] If each noun GU CONTENT role inhibits every other, and there are n noun GUs, n^2 connections would be required, probably an unmanageable number. Section 11.1.1 discusses one possible way around this problem.

*PEPPER:NOUN then activates the word node "PEPPER". The firing of "PEPPER" represents the utterance of the word *pepper*.

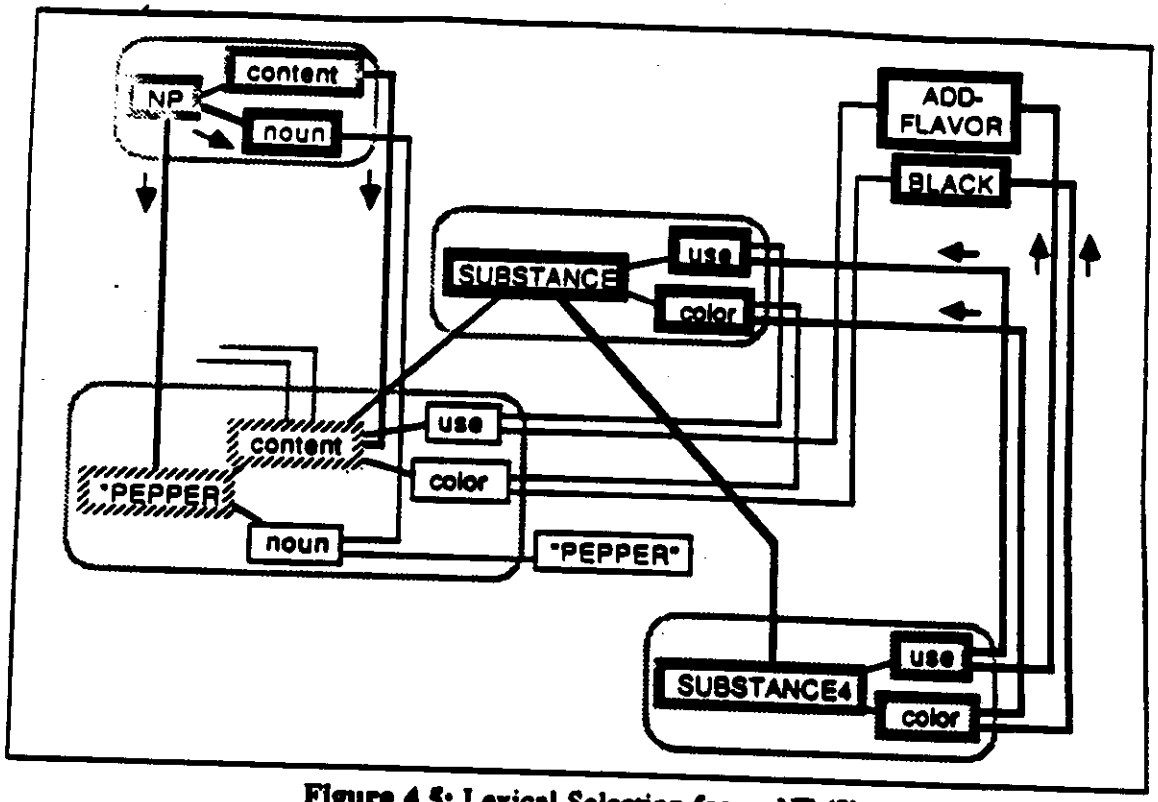

Figure 4.5: Lexical Selection for an NP (2)

Note how the spread of activation implements pattern matching. Activating an input with the feature USE = ADD-FLAVOR leads to the partial activation of all schemas with this same feature. For a given set of role-value pairs in the input, competition via WTA networks selects the best matching schema. Note also that, because the features are checked in parallel, an increase in the number of features does not lead to a slowing down of the process. Lexical selection in the CLM model capitalizes on a well-known property of connectionist approaches, the **content addressability** of memory: the semantic portion of lexical GUs is directly accessible from semantic features of the input.

4.2.2. A Generalized Illocution

The generation of an utterance in the CLM model begins with an instance of a schema of the INTEND type, with the speaker (SELF) as PLANNER. In the case of directives, the input is an instance of AGENCY and the GOAL is to get an ASSIGNEE (eventually the hearer) to want to do something. By means of the same bottom-up procedure that accesses lexical GUs, this input instance is associated with a GI in memory.

Consider the generation of the sentence *could you take out the trash?* We assume that the speaker is the wife of the hearer and that, for her, the *could you* request form is a conventional way of asking her husband to perform a task which involves a medium degree of imposition (Brown & Levinson, 1978). The GI is shown in Figure 4.8.

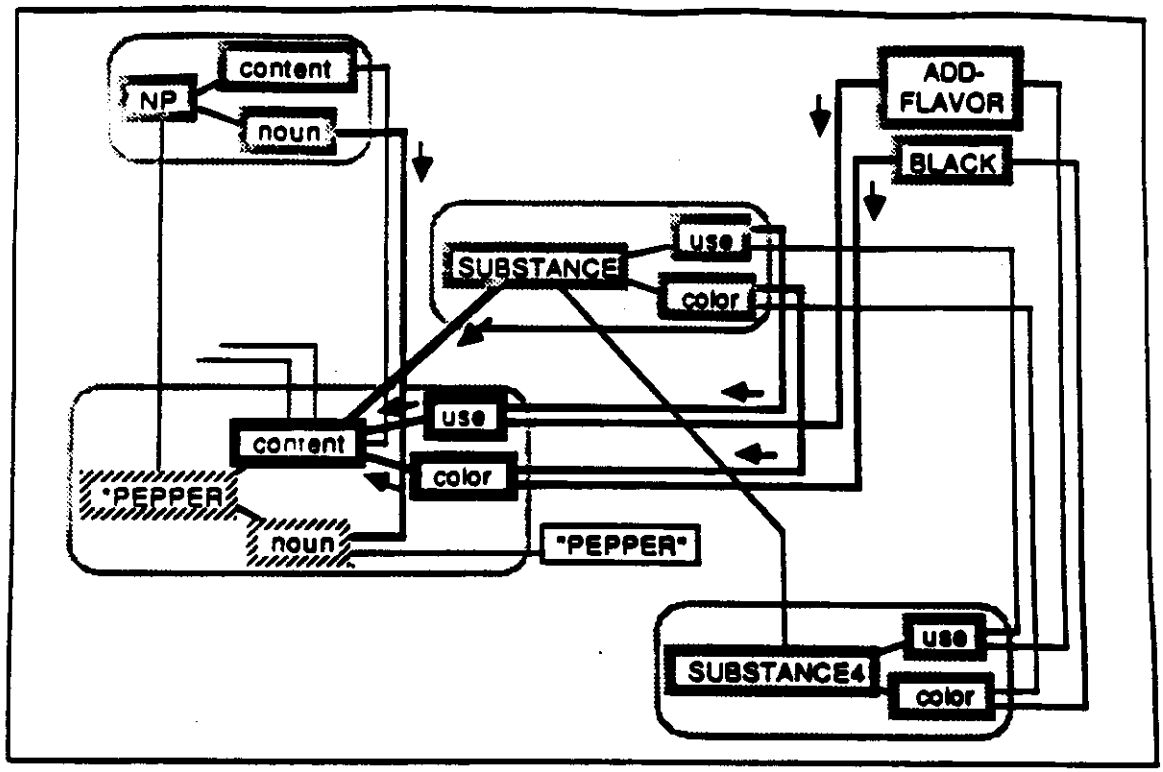

Figure 4.6: Lexical Selection for an NP (3)

The speaker might even have conventionalized a form for the specific case of requesting her husband to take out the trash, but to make the example more interesting, we will assume that this is not the case. In order for the GI in Figure 4.8 to be used, the sytem needs to be able to determine when an act is of medium imposition for a particular ASSIGNEE. We will assume that there is special knowledge about taking out heavy objects, in particular, the explicit knowledge that for the speaker's husband to take moderately heavy objects out of the speaker's house is an act of medium imposition. This schema is shown in Figure 4.9.

As the input to generation, we take the intention of the speaker (SELF) to have the hearer (OWN-HUSBAND) act as the speaker's agent (ASSIGNEE) in taking out the trash. This instance of AGENCY is shown in Figure 4.10.

The input to the generation of the sentence is the firing of the nodes representing features of AGENCY17. Two of the role-value pairs match those in the GI REQUEST/OWN-HUSBAND_MEDIUM-IMPOSITION directly: PLANNER = SELF and ASSIGNEE = OWN-HUSBAND, and through the basic pattern-matching mechanism implemented by spreading activation, the PLANNER and ASSIGNEE roles in the GI fire. The ASSIGNMENT portion of the GI is matched indirectly, first through the recognition that the ASSIGNMENT in the input matches the HUSBAND-TAKE-OUT-MEDIUM-WEIGHT schema shown in Figure 4.9. This latter process is again a matter of pattern matching. In this case it depends on the knowledge (not shown in the figures) that the WEIGHT of TRASH4 is MEDIUM. The firing of the head node of OWN-HUSBAND-TAKE-OUT-MEDIUM-WEIGHT leads to the firing of the MEDIUM-IMPOSITION node. This in turn provides the extra activation needed for the ASSIGNMENT role in the GI to fire. Together the PLANNER, ASSIGNEE, and ASSIGNMENT roles send enough activation to the GI head node for it to fire. Next the PLAN role fires,

yielding a portion of the pattern (*could you*) and initiating the selection process for a verb GU. Figure 4.11 summarizes the steps in the GI selection process.

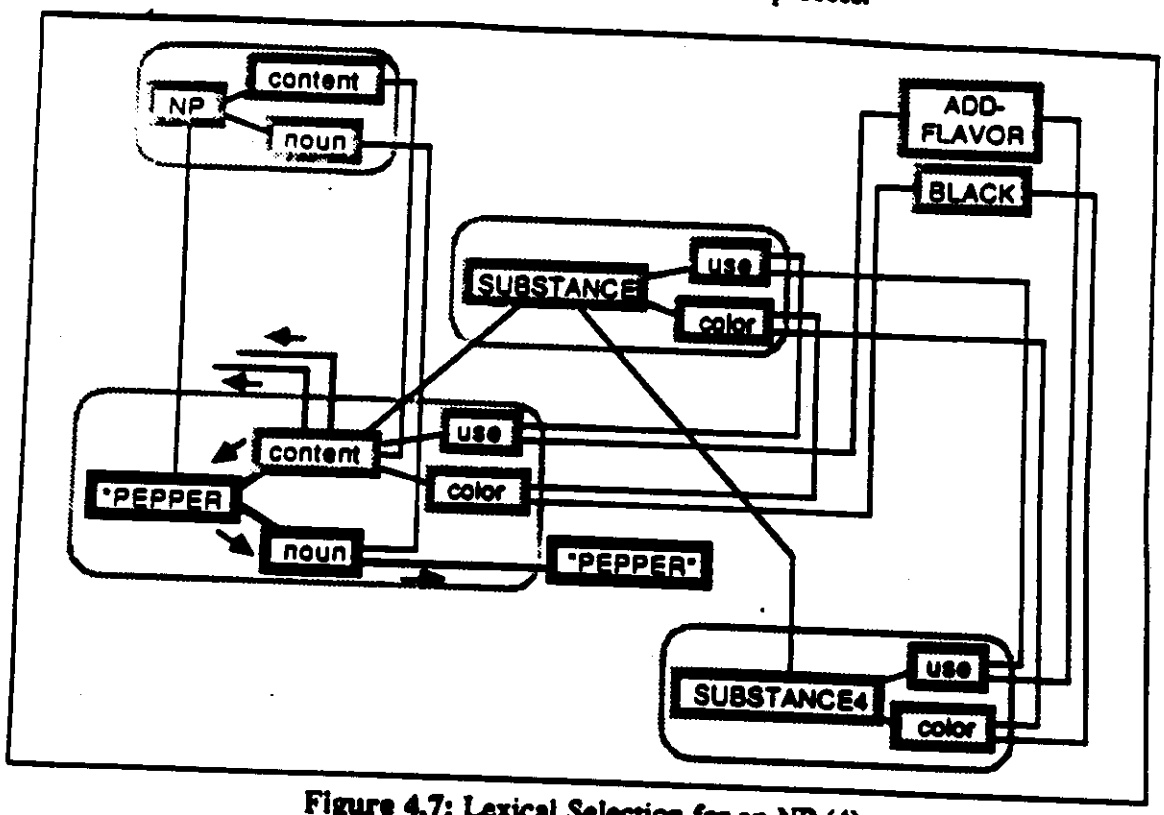

Figure 4.7: Lexical Selection for an NP (4)

```
(REQUEST/OWN-HUSBAND_MEDIUM-IMPOSITION is-a AGENCY
    (planner SELF)
    (assignee OWN-HUSBAND)
    (assignment (MEDIUM-IMPOSITION ?A))
    (plan YES-NO-QUESTION
        (speaker SELF)
        (hearer OWN-HUSBAND)
        (content ?A)
        (modality ABILITY)
        (subject "YOU")
        (modal "COULD")))
```

Figure 4.8: GI for an Individual Request Convention

```
(OWN-HUSBAND-TAKE-OUT-MED-WT-OBJ is-a (PHYSICAL-TRANSFER
                                        MEDIUM-IMPOSITION)
    (actor OWN-HUSBAND)
    (object PHYSICAL-OBJECT
        (weight MEDIUM))
    (source OWN-HOUSE-INSIDE)
    (destination OWN-HOUSE-OUTSIDE))
```

Figure 4.9: Specific Knowledge about Level of Imposition

```
(AGENCY17 is-a AGENCY
    (planner SELF)
    (assignee OWN-HUSBAND)
    (assignment PHYSICAL-TRANSFER
        (object TRASH4)
        (source OWN-HOUSE-INSIDE)
        (destination OWN-HOUSE-OUTSIDE)))
```

Figure 4.10: Input to Generation of a Request

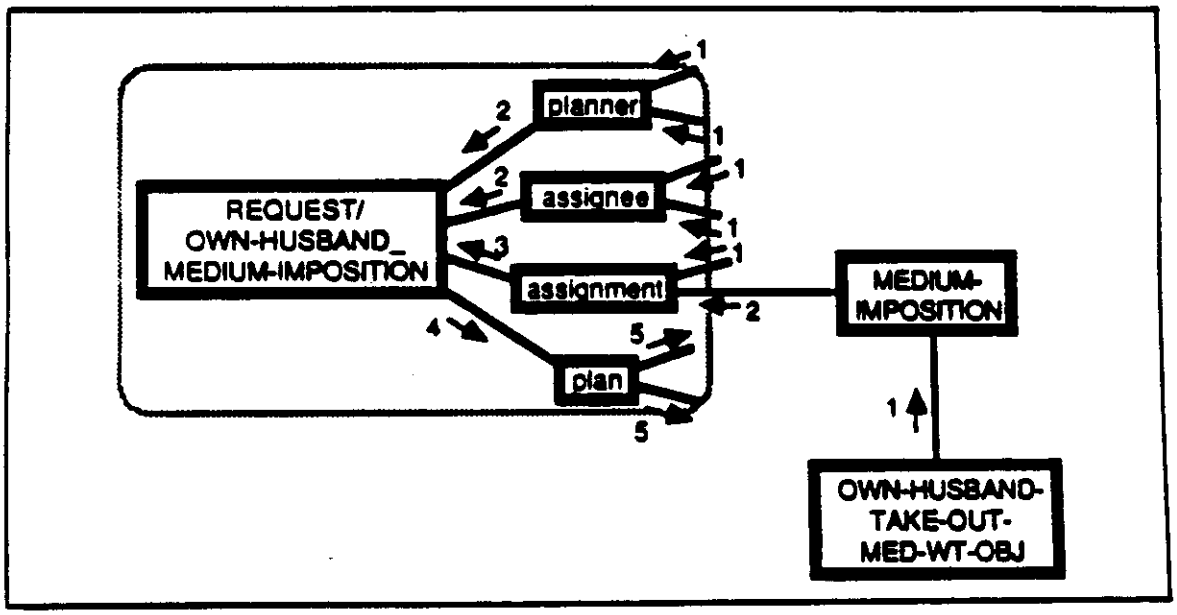

Figure 4.11: Selection of a GI

The activation of the PLANNER, ASSIGNEE, and ASSIGNMENT roles of the GI takes place in parallel, though the last takes somewhat longer because it involves an intermediate step. The numbers in the figure indicate the order in which the connections are traversed. The PLANNER role fires as a result of activation from PLANNER (in the AGENCY schema) and SELF. ASSIGNEE fires after activation from ASSIGNEE (in the AGENCY schema) and OWN-HUSBAND. OWN-HUSBAND-TAKE-OUT-MEDIUM-WEIGHT-OBJECT fires because it gets activated by its ACTOR, OBJECT, SOURCE, and DESTINATION roles. Activation from

PLANNER, ASSIGNEE, and ASSIGNMENT causes the head node REQUEST_OWN-HUSBAND/MEDIUM-IMPOSITION to fire. This sends activation to its PLAN role, representing the utterance to be produced.

4.2.3. Merged Nodes in Schema Selection

Sometimes it is necessary to match merged nodes in a schema. For example, the SUICIDE schema has a merged node representing the fact that the ACTOR and OBJECT of the killing are the same person, and an instance of killing should be classified as SUICIDE in case this condition holds for it. Dealing with merged nodes in schema selection involves several complications.

Consider first the case of an instance of killing which should qualify as suicide. Note that we are not dealing here with a linguistic classification, but the schema selection process is the same, and the example will suffice to illustrate the points. Figure 4.12 shows how selection operates in this situation.

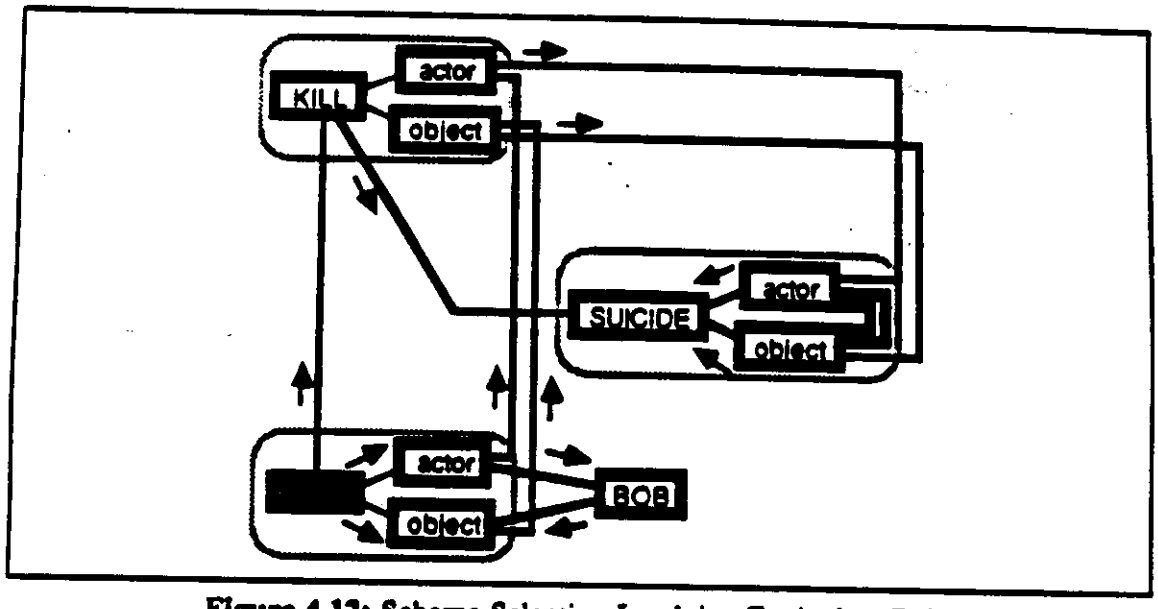

Figure 4.12: Schema Selection Involving Equivalent Roles

Assume that the head node of the killing instance, KILL1, fires and activates both of its roles but that only the ACTOR role fires. The firing of KILL1:ACTOR would represent the fact that this participant in the event was relatively salient. Activation from KILL1:ACTOR spreads to the node for the value of the role, BOB, which fires and causes KILL1:OBJECT to fire. The nodes representing KILL1 activate the nodes in the general KILL schema. KILL:ACTOR and KILL:OBJECT both send activation to the merged node in the SUICIDE schema, and it fires as a result, sending activation to SUICIDE. This is already primed as a result of activation from KILL and it fires, completing the selection process.

Consider now what happens when the ACTOR and OBJECT are distinct. Given the input from Figure 4.12, the OBJECT role in the instance of KILL would fail to fire, resulting eventually in less activation to the merged node in the SUICIDE schema and no firing of the SUICIDE head node. But what if the ACTOR and the OBJECT of the instance fired simultaneously as a result of their salience or some external sources of activation? Clearly this would again lead to the selection of SUICIDE, in this case, wrongly. To prevent an invalid classification like this, the ACTOR and OBJECT roles in an instance of KILL inhibit

each other slightly so that only one can fire at a time. Figure 4.13 shows the case where the ACTOR node in the instance fires. It inhibits the OBJECT node, preventing it from firing also. Activation reaching the SUICIDE schema is then not enough for it to be selected for this instance.

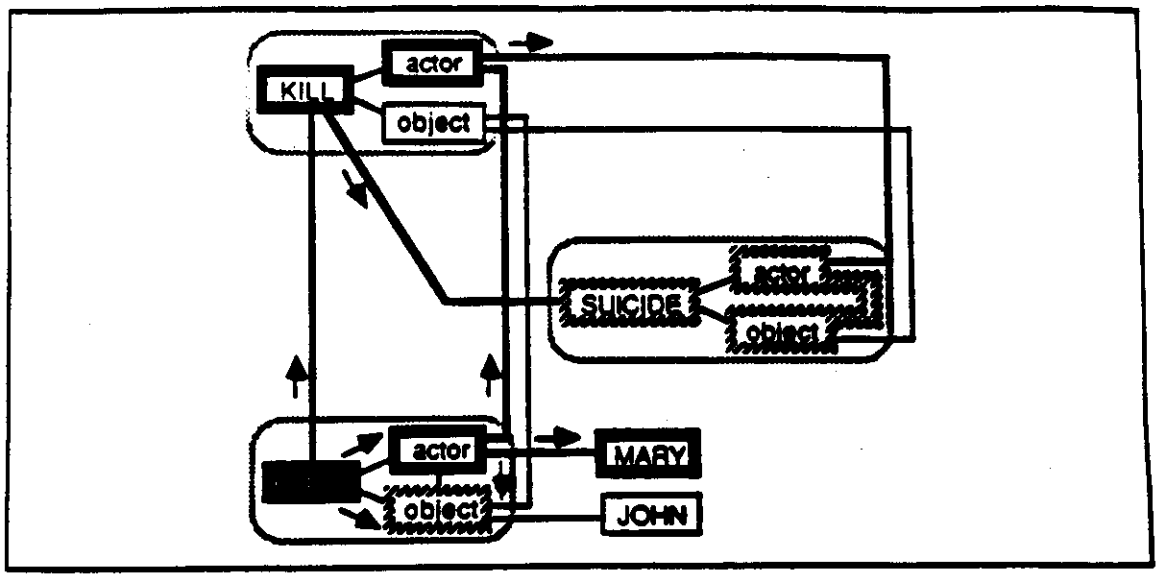

Figure 4.13: Schema Selection Involving Inhibition Between Roles

However, we would like to permit the OBJECT role in KILL2 to fire soon after the firing of ACTOR so that it and its value can participate in the classification process. For example, if JOHN is an instance of the concept POLITICAL-FIGURE, the ASSASSINATE schema should be selected for KILL2. In order to prevent the merged node in the SUICIDE schema from firing at this time, the priming remaining on it from initial activation must have already decayed significantly. For this reason, the merged SUICIDE:ACTOR+OBJECT node, and merged nodes in general, have a relatively rapid decay rate. The rapid rate represents the requirement that the two sources of activation come in close succession in order for the node to fire. That is, if the gap between the arrival of the activation sources is long, activation from the first source will have decayed to the point that the sum of the two inputs will not exceed the node's threshold.

4.3. Specific and General Schemas

4.3.1. Basic-Level Categories

It is a well-known feature of human language generation (Rosch, 1978) that speakers have a default level of reference for each hierarchy of concepts. For example, given a particular dog to refer to, most speakers will prefer the word *dog* over the more general *mammal* or *animal* and the more specific *dachshund*, unless there is some reason to assert the referent's membership in one of the other categories. *Dog* is the basic-level term for this hierarchy.

Figure 4.14 shows some of the knowledge needed to select the GUs for the nouns *animal*, *dog*, and *dachshund*.

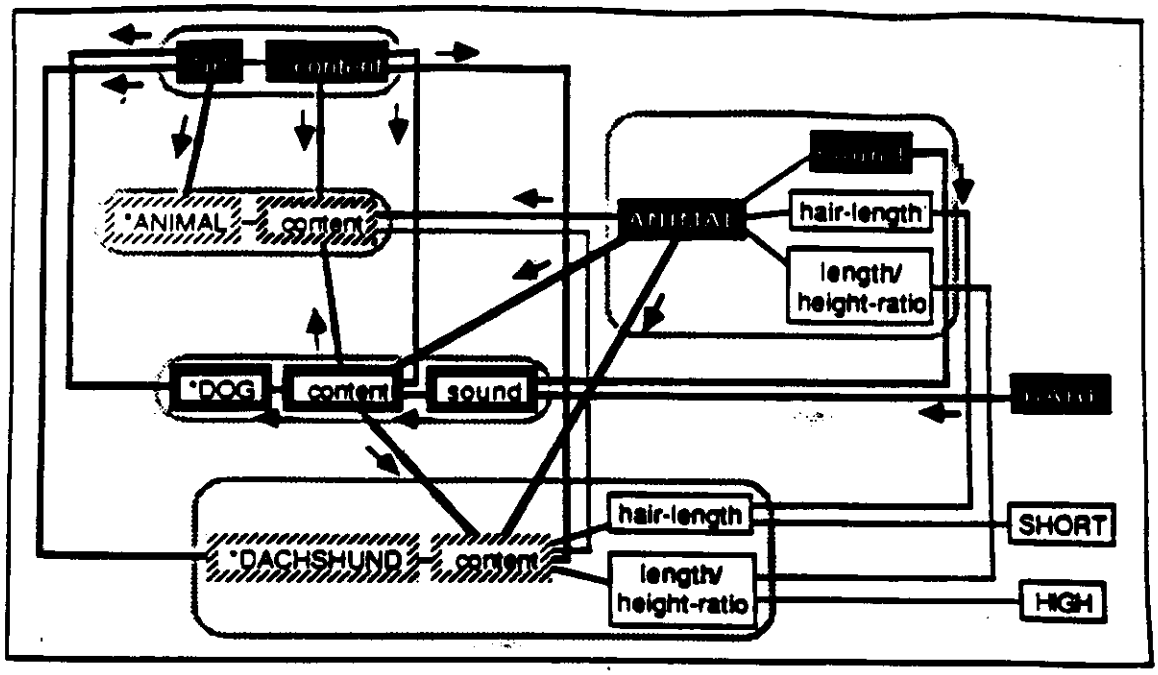

Figure 4.15: Selection of *DOG Over *ANIMAL

What happens when features characteristic of dachshunds but not all dogs are activated in the input? Figure 4.16 shows how it is still possible for the *DOG GU to win out.

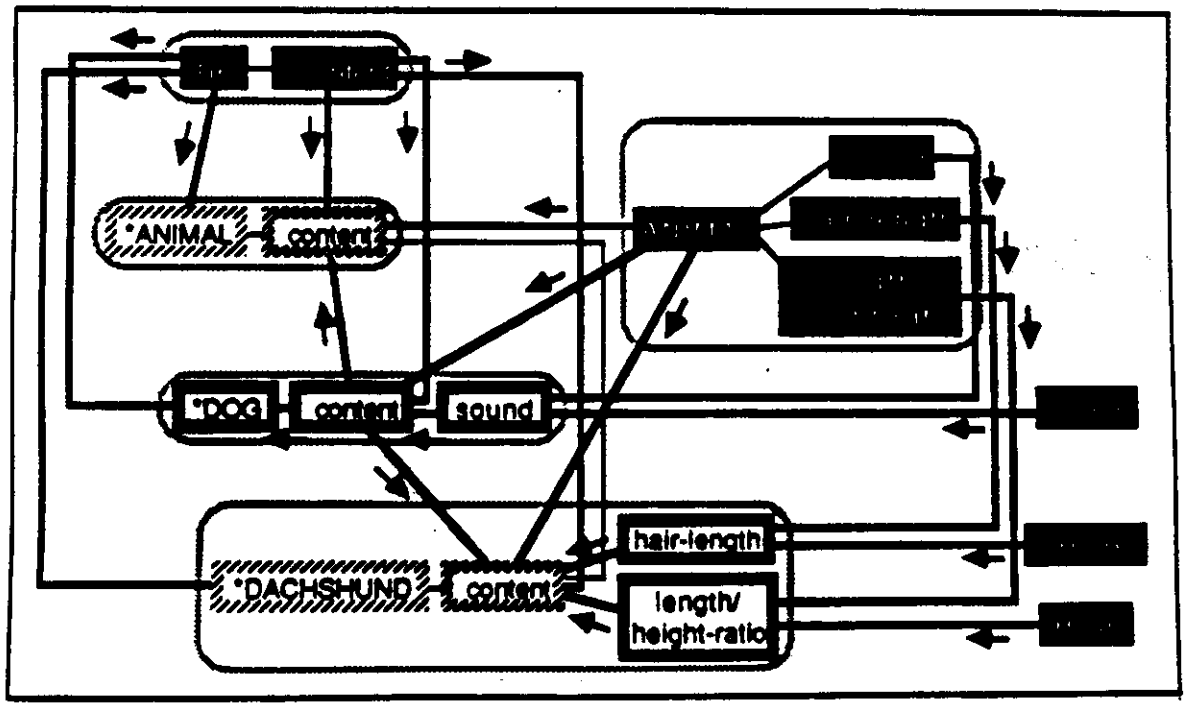

Figure 4.16: Selection of *DOG Over *DACHSHUND

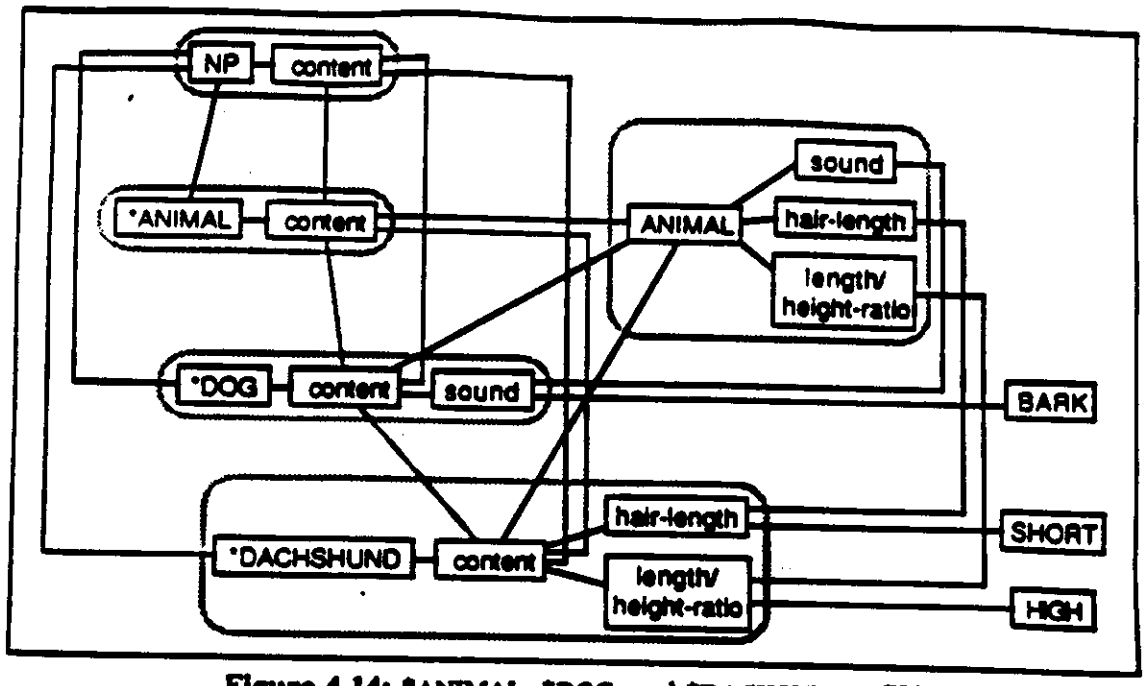

Figure 4.14: *ANIMAL, *DOG, and *DACHSHUND GUs

The *ANIMAL GU has as its CONTENT the concept ANIMAL, shown with three of its roles. The CONTENT of *DOG also points to ANIMAL, but in addition it specifies features specific to dogs, one of which is shown in the figure. The characteristic SOUND of a dog is a BARK. The CONTENT of *DACHSHUND likewise points again to ANIMAL and has features which distinguish dachshunds from other dogs, e.g., the fact that their HAIR-LENGTH is SHORT and their LENGTH/HEIGHT-RATIO HIGH. The figure does not show morphological portions of the GUs, that is, the information that the head NOUNs for the schemas are the words *animal*, *dog*, and *dachshund*.

Consider now the generation of an NP to refer to a particular ANIMAL for whom a salient feature is the fact that its SOUND is a BARK. The GU selection process is shown in Figure 4.15. The input for this example (not shown in the figure) initially activates ANIMAL and the two nodes representing the barking feature. In addition, as usual for the generation of NPs, NP and NP:CONTENT fire. ANIMAL and NP:CONTENT send activation to the CONTENT roles of all three GUs. If the conection from ANIMAL to *ANIMAL:CONTENT were relatively strong, then *ANIMAL:CONTENT would fire at this point. We assume, however, that the connection has a relatively low weight, a reflection of the fact that ANIMAL is not a basic-level category. Activation from ANIMAL:SOUND and BARK leads to the firing of the SOUND role of *DOG:CONTENT, sending additional activation to *DOG:CONTENT and causing it to fire. This results in the inhibition of competing GUs and the selection of *DOG. If the barking feature had not been present, the WTA hub node for this set of GUs (not shown in the figure) would have timed out and caused *ANIMAL:CONTENT to fire, leading to the selection of *ANIMAL instead.

joined by excitatory connections. The possibility of both GUs firing allows the system to store some of the information needed for *BREAK-BONE under *BREAK. In particular we would like to avoid having special schemas for *broke a BONE* and *broken a BONE* because the irregular forms of *break* are properties of the verb in all its uses. Knowledge about the forms of the verb *break* is stored in subtypes of *BREAK. For example, the GU *BROKE, which is also a subtype of the general GU PAST-CLAUSE, provides the irregular form *broke*. To get *broke a BONE*, both *BREAK-BONE and *BROKE fire. Figure 4.18 shows how this process takes place.

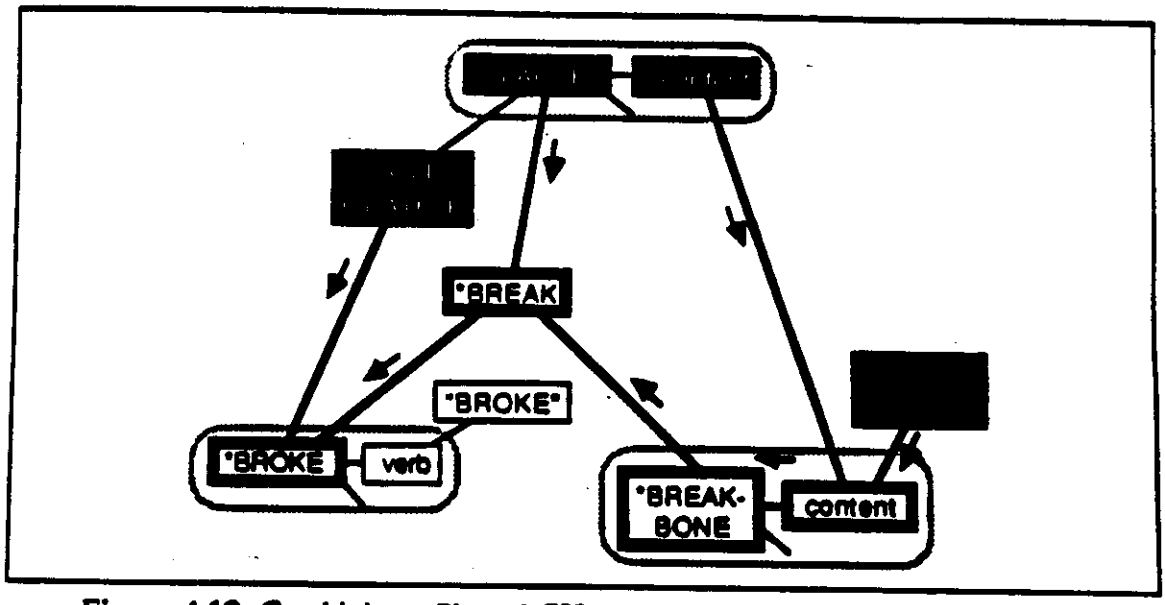

Figure 4.18: Combining a Phrasal GU with One Specifying an Irregular Form

It is assumed that the PAST-CLAUSE schema has already fired as a result of activation spreading from nodes representing temporal features of the input. The *BREAK-BONE GU is selected on the basis of input features. The head node for this schema activates the more general *BREAK GU. *BREAK and PAST-CLAUSE both send activation to the *BROKE GU, which fires as a result of this convergence of activation. This later yields the appropriate form of the verb for the clause being generated.

4.4. Features of the Schema Selection Mechanism

4.4.1. Robustness: Coping with Novel Input

In the examples thus far the input notions perfectly matched the content roles of particular schemas. However, we cannot always count on perfect matches. In cases where there is no matching schema, competition between two or more schemas may result.. Consider sentence (1.4), repeated here:

(1.4) *I'd like to deposit this jewelry.*

The system ends up selecting *deposit* to refer to the desired act even though the meaning associated with this verb involves the transfer of money to a bank. A portion of the GU for *deposit* is shown in Figure 4.19.

For the example, assume the nodes representing the short-hair and long-body features of dachshunds are activated in the input. This leads to the firing of the nodes ANIMAL;HAIR-LENGTH and SHORT and ANIMAL:LENGTH/HEIGHT-RATIO and HIGH and as a result the firing of the HAIR-LENGTH and LENGTH/HEIGHT-RATIO roles of *DACHSHUND:CONTENT. These nodes activate *DACHSHUND:CONTENT, but the weights on the connections are not high enough to cause the node to reach its threshold. *DOG:CONTENT wins out again, and the basic-level term is used.

How might it ever be possible to generate the noun *dachshund* then? If DOG is the basic-level category, this should happen only when there is an explicit intent to make reference to the breed of the dog. The node for BREED would be activated in the input in such cases. This node activates nodes such as *DACHSHUND:CONTENT and *COLLIE:CONTENT, but more importantly, it inhibits *DOG:CONTENT, causing it to lose out in the winner-take-all competition which selects one lexical CONTENT node. Figure 4.17 illustrates how this would result in the selection of *DACHSHUND with the same set of input dog features as in the last example. *DACHSHUND:CONTENT fires either as a result of the additional activation it receives from BREED or as a result of activation from the firing hub node (not shown in the figure) for the WTA network linking it to related CONTENT nodes.

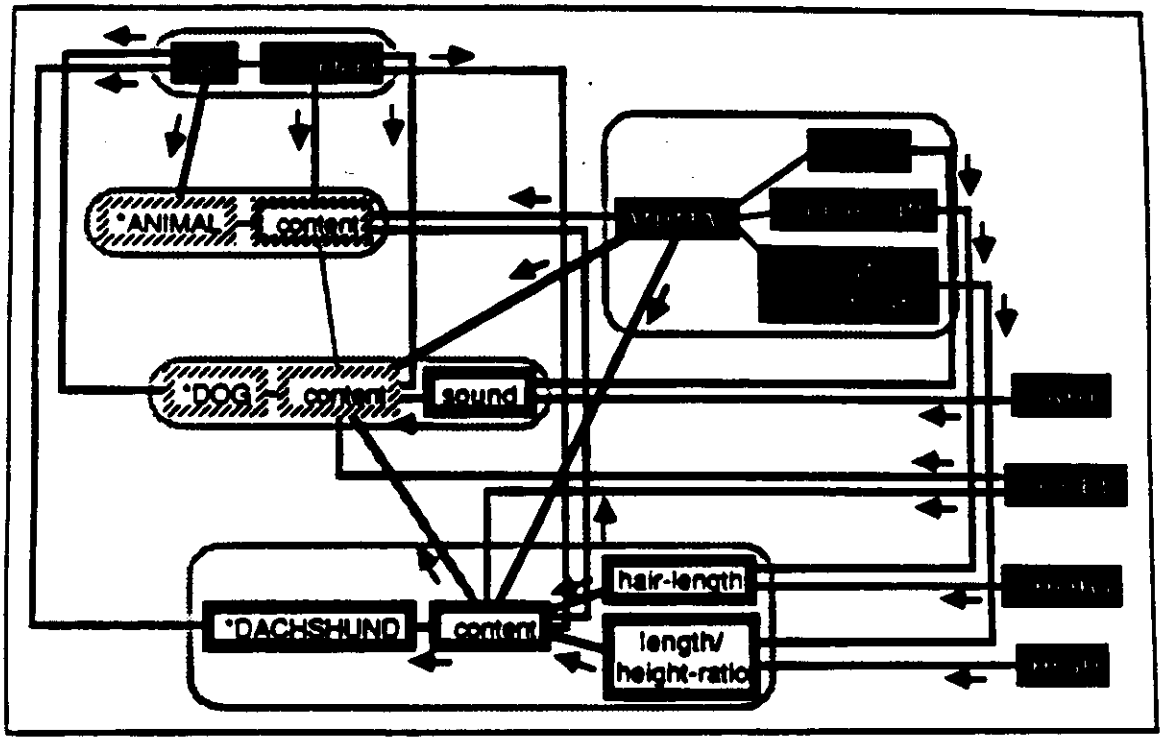

Figure 4.17: Selection of *DACHSHUND Over *DOG

4.3.2. Pattern Specificity

There is another sense in which relative generality is an issue in schema selection. Linguistic memory may contain schemas for patterns of varying degrees of specificity. For example, there may be a GU for the expression *break a BONE* as well as a more general one for *break*. In order to be useful, the specific GU must be selected when appropriate. This type of situation differs in an important way from that discussed in the last section, however. The GUs *BREAK-BONE and *BREAK do not inhibit each other; in fact, they are

The CONTENT role of *GIVE, like that of *DEPOSIT-IN-BANK, is an instance of TRANSFER-OF-CONTROL. It also includes the information that the ACTOR and SOURCE of the transfer are the same and the tendency for the DURATION of the transfer to be PERMANENT. For the example, *GIVE:CONTENT receives input from TRANSFER-OF-CONTROL and from the firing merged role representing the ACTOR and SOURCE. The CONTENT of *DEPOSIT-IN-BANK, on the other hand, receives input from its RECIPIENT and DURATION roles (and others not shown in the figure) in addition to TRANSFER-OF-CONTROL and the merged ACTOR+SOURCE node. *DEPOSIT:CONTENT wins out over *GIVE:CONTENT because of the activation received from a greater number of matching roles.

Of course, a GU other than *DEPOSIT could also be selected in this situation. A speaker might use the verb *leave* instead, for example. The point is that the system is able to cope with input that does not precisely match any of the schemas in memory.

4.4.2. Generating Substitution Errors

A common type of speech error involves the substitution of one element for another, as in *pass the salt* for *pass the pepper*.[8] Like other errors, these have been a major source of data for research on human language generation. The assumption has been that errors are a reflection of the basic processes that are behind error-free generation.

Like most speech errors, those involving substitution can be accounted for in terms of priming. When it comes time for the target element to be selected, other elements belonging to the same general category compete for selection. One of these has more activation than the target and is selected (fires, that is) in its place. Consider what goes on in the substitution of *salt* for *pepper*. The normal selection process for the *PEPPER GU was illustrated in Figures 4.4, 4.5, 4.6, and 4.7. The CONTENT role of the *PEPPER GU receives activation from the firing SEASONING node. This role competes through a WTA network with the CONTENT roles of other noun GUs with similar CONTENT, including the one for the *SALT GU. *SALT:CONTENT will also have received some activation from the firing of the representation of the desired pepper. In particular the node for SEASONING will send some activation to *SALT:CONTENT. *PEPPER:CONTENT will still have more activation, however, because of the features of the input concept unless either 1) the nodes in the *SALT schema already had some activation left over from other processing or 2) random noise in the system left *PEPPER:CONTENT with less activation and/or *SALT:CONTENT with more activation than normal. In either of these unusual cases, *SALT:CONTENT would win out over *PEPPER:CONTENT, leading to the selection of the *SALT GU and eventually the production of the the word *salt*.

Figures 4.21 and 4.22 illustrate two stages in the generation of the error. In 4.21 we see some of the ways in which the CONTENT of the two GUs is related. They share a general type, SUBSTANCE, and their USE feature has the same value, ADD-FLAVOR. For another feature, their COLOR, the two concepts differ. In this figure the *SALT:CONTENT node is shown as primed, that is, already activated for one reason or another. Figure 4.22 shows the spread of activation the NP and NP:CONTENT nodes and also from the conceptual nodes that are associated with the input instance of pepper, which is not shown in the figure. The conceptual activation sources shown in the figure represent the fact that the input concept is a SUBSTANCE with ADD-FLAVOR as its USE and BLACK as its COLOR. Activation converges on the role nodes of the CONTENT roles of the two GUs. Both the USE and COLOR roles fire in the *PEPPER schema, but only the USE role in the *SALT schema matches the input. Both *PEPPER:CONTENT and *SALT:CONTENT receive

[8] See Dell (1986) for a more complete classification of errors. The examples discussed in this thesis are from Dell's list (p. 285), which includes some errors gathered by other researchers.

```
(*DEPOSIT-IN-BANK TRANSITIVE-CLAUSE
    (verb "DEPOSIT")
    (content TRANSFER-OF-CONTROL
        (actor ?A)
        (object (MONEY ?O))
        (source ?A)
        (recipient BANK)
        (duration TEMPORARY)
        (purpose1 PREVENT          ;;keep money safe
            (object LOSE
                (object ?O)))
        (purpose2 INCREASE-VALUE   ;;earn interest
            (object ?O))))
```

Figure 4.19: Portion of GU for *deposit*

At the point in generation where the verb GU selection is to take place, a number of nodes representing the input concept will have fired. As activation spreads from these nodes, it will intersect on the roles of some GUs. Within the CONTENT of *DEPOSIT, the ACTOR, SOURCE, RECIPIENT, DURATION, and PURPOSE1 (protecting the transferred property) roles will fire, while the OBJECT role, which is an instance of JEWELRY rather than money, and the PURPOSE2 (earning interest) role, which is not applicable to the jewelry, will fire. The firing roles send activation to the CONTENT node of the GU, which also receives activation from TRANSFER-OF-CONTROL. The CONTENT node competes via a WTA network with the CONTENT roles of all other verb lexical GUs, including some which will also have significant activation, such as the one in the GU for *give*. The WTA network sees to it that only one of the CONTENT nodes fires. Figure 4.20 shows a portion of two of the GUs whose CONTENT roles are competing.

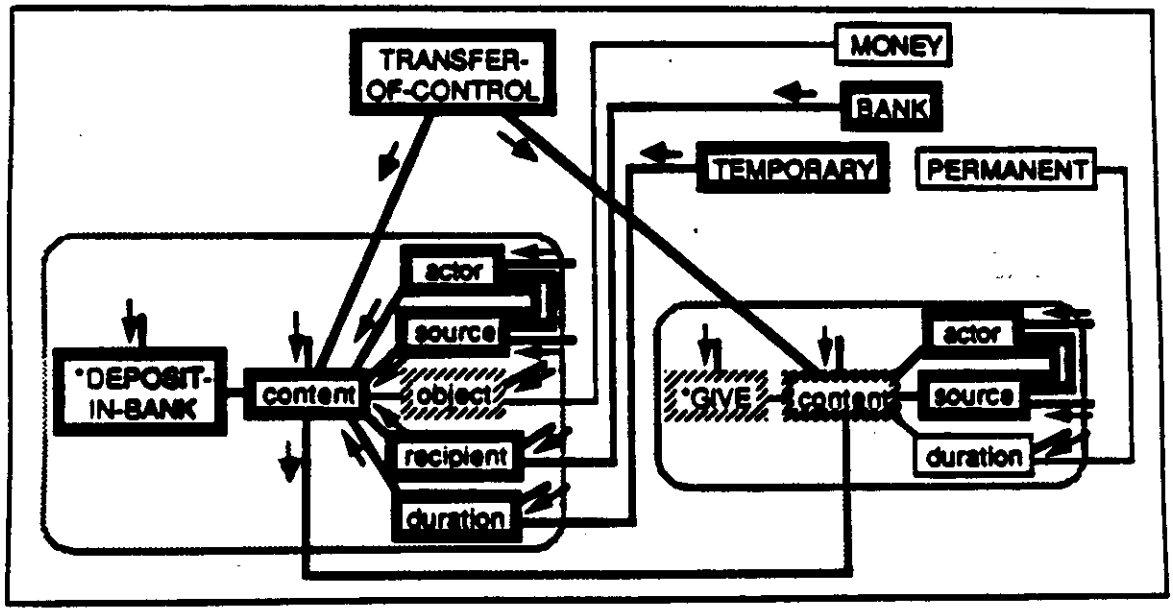

Figure 4.20: Selection of a Partially Matching GU

selection process works in the comprehension as well as the generation direction using the same network of conceptual and linguistic knowledge.

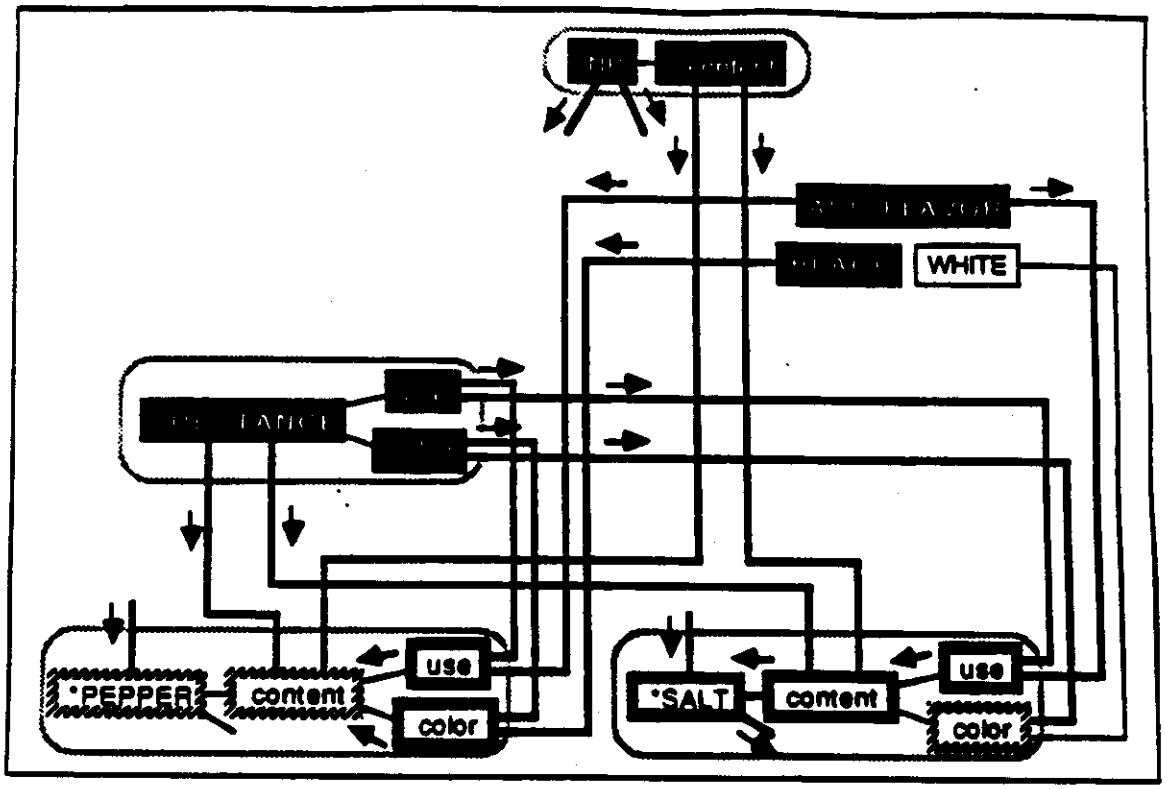

Figure 4.22: Generation of a Substitution Error (2)

activation from NP:CONTENT, from SUBSTANCE, and from their firing role nodes. *PEPPER:CONTENT receives more because two of its roles have fired, but the priming on *SALT:CONTENT gives it enough activation to predominate anyway, resulting in the selection of the wrong GU.

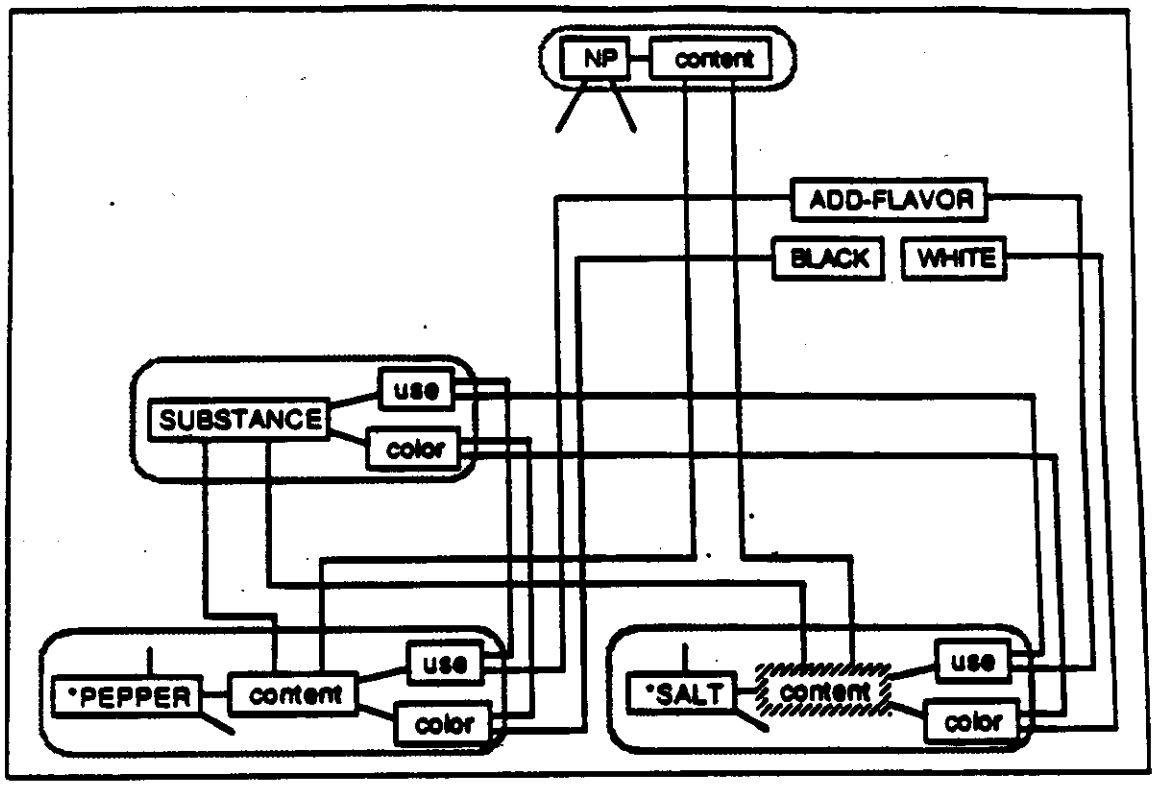

Figure 4.21: Generation of a Substitution Error (1)

4.5. Summary

In this chapter I have described the processing characteristics of the nodes in the CLM network and shown how they enable units of linguistic knowledge to be accessed during language generation. Important features of processing are (1) firing thresholds on nodes to check the spread of activation and implement the making of decisions, (2) refractory periods following firing to prevent the repeated firing of nodes, (3) priming resulting both from activation below the firing threshold and firing itself, (4) winner-take-all networks to implement competition among groups of related concepts, and (5) a varying decay rate both to implement short-term forgetting and variable constraints on merged nodes.

The schema selection process is accomplished without any of the domain-specific rules that would be part of a traditional symbolic account. More importantly, the mechanism exhibits five properties which are desirable in a model of human generation. (1) It accesses schemas directly from conceptual features of the input. (2) It permits the system to find a schema even when the input does not match it perfectly. (3) It produces occasional substitution errors as people do. In addition, as will be shown in Chapter 7, the

Chapter 5

Language Generation: Role Binding

5.0. Introduction

Some aspects of language generation are trivial in conventional symbolic models. The process which associates syntactic arguments with semantic arguments, for example, SUBJECT with MARY in the generation of the sentence *Mary deposited $100 in the bank*, can take the form of a low-level rule which explicitly binds local variables to values. Similarly, the process which sequences words and constituents can be carried out by a rule which simply prints out the items which are currently bound to the constituents in a pattern sequence. For example, if the sequence specifies SUBJECT + VERB, the rule would generate whatever is the current subject and then whatever is the current verb.

Things are not so simple in a connectionist approach. No variables are available, so there is no way to temporarily link the SUBJECT node to the node representing the entity that should be referred to in the SUBJECT position. Sequencing requires not only some means of binding variables, but also a way of representing sequencing information and of making it available at the appropriate time. It is not at all obvious how this might be done in a system where processing is basically parallel. This chapter and the next one are concerned with the way in which role binding and sequencing are implemented in the CLM model.

5.1. The Basic Role Binding Process

Role binding in generation is the process which associates particular roles in a schema with particular values. The value can then be used in the selection process for further schemas. In the CLM approach the explicit variable binding that would be used in a symbolic model is replaced by the firing of the relevant nodes in relatively close temporal proximity. If the direct object of a clause is to refer to the semantic object of the fact being predicated, then the DIRECT-OBJECT node and the node representing the object fire at more or less the same time. Because there is a path of primed nodes connecting the role and value nodes, the binding remains temporarily accessible.

Consider the association of the direct object with the pepper in the generation of the sentence *could you pass the pepper?* Role association information may be available at different levels, that is, in the general GU for transitive clauses or in a lexical GU. For this example, we will assume that it is found in the lexical GU for *pass*. We start at the point where this schema has just been selected. The realization of the subject of the clause as *you* is taken care of by a GI specifying the *could you* pattern for the goal of getting the hearer to do something for the speaker. At this point in the generation, a number of nodes have either fired or been activated. For our purposes, it is important that the node for the semantic OBJECT of the PHYSICAL-TRANSFER instance being referred to is primed at this point because it has received some activation from its head node during the selection process for the verb GU. Figure 5.1 shows the relevant portion of the network. In this figure and Figure 5.2, node activation and activation spread are only indicated where they are of interest.

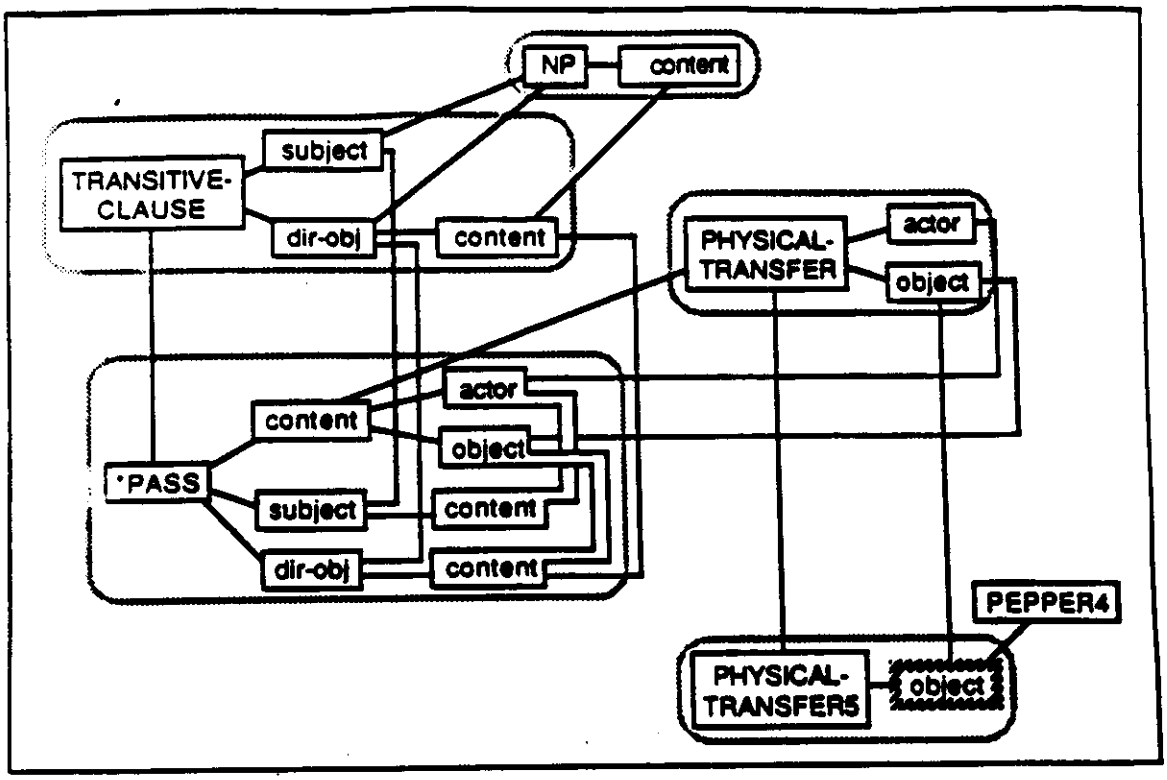

Figure 5.1: Binding a Referent to a Syntactic Role (1)

The passing action that the hearer is expected to perform is represented in Figure 5.1 as PHYSICAL-TRANSFER5, an instance of the general concept of PHYSICAL-TRANSFER. This concept in turn is associated with the *PASS GU. This GU also contains the role association information necessary to generate the subject and direct object.[9] For the direct object this information is contained in the merged node which equates the CONTENT of the DIRECT-OBJECT with the semantic OBJECT of the CONTENT of the clause. *PASS is in turn a subtype of the general TRANSITIVE-CLAUSE GU. This schema is where general properties of the SUBJECT and DIRECT-OBJECT are stored, though this information does not appear in the figure. Both the SUBJECT and DIRECT-OBJECT point off to the general GU for NPs.

Figure 5.2 shows how the activation of the role association information for the direct object leads to the firing of PEPPER4 and DIRECT-OBJECT. When the head node of the *PASS GU fires, it activates *PASS:DIRECT-OBJECT, and this in turn activates the merged node representing the role association information for that constituent. The merged node sends activation out in two directions. In one direction this results in the firing of the DIRECT-OBJECT, NP and NP:CONTENT nodes. In the other direction it leads to the firing of PHYSICAL-TRANSFER:OBJECT, PHYSICAL-TRANSFER5:OBJECT, and PEPPER4. The firing of NP, NP:CONTENT, and PEPPER4 at more or less the same time provides the input to the lexical selection process for the final NP in the sentence.

[9]For simplicity, I will assume that the verb entries apply only to active clauses and have roles such as SUBJECT and DIRECT-OBJECT rather than LOGICAL-SUBJECT and LOGICAL-DIRECT-OBJECT.

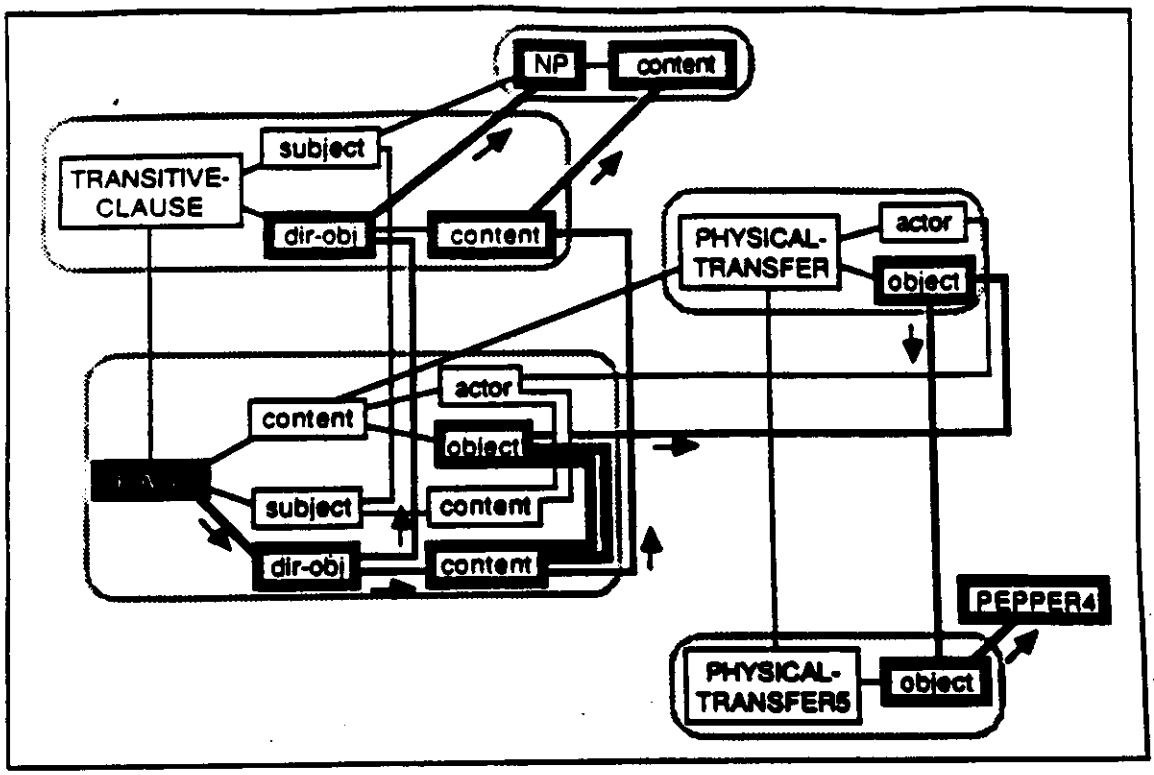

Figure 5.2: Binding a Referent to a Syntactic Role (2)

Recall that the *PEPPER GU is selected as a result of activation intersecting on its CONTENT role from two paths, one beginning with NP:CONTENT, the other with the instance of PEPPER being referred to. In the example, if PEPPER4 is activated long before NP:CONTENT, then the activation which PEPPER4 provides to *PEPPER:CONTENT will not be great enough for the firing threshold to be reached when additional activation arrives from NP:CONTENT. Thus the limiting factor for the time difference in the firing of the nodes representing bindings is the decay of activation.

Note that the success of the binding process here depends on the existence of an explicit connection from PHYSICAL-TRANSFER5:OBJECT to PEPPER4. A general problem for localized connectionist models is where such links come from in the first place. There is no account of the origin of these role-binding connections in the CLM model, and I speculate on possible mechanisms in Chapter 9.

5.2. Using Binding Paths in Schema Selection

In other cases, a binding that is made at one point must be accessed later on. For example, the selection of some GUs depends on the current values of the deictic roles SPEAKER and HEARER. Examples of such schemas which we have already encountered include the GUs for ME, BUNNY, and BRING. In this section I look at four types of situations in which bindings are used after they are first created.

5.2.1. Paradigm-Driven Schema Selection: Pronouns

As we saw in Chapter 4, finding schemas in linguistic memory that match the pragmatic/semantic input features is an essentially bottom-up process. Activation propagating from input features intersects on roles of one or more schemas, one of which finally predominates. The process need not be guided by rules or paradigms built into the language. If the generation of an utterance were to stop here, however, the result would have the character of pidginized speech. This is because, within the system of each language, some distinctions are made which are in a sense imposed on speakers, whether they intend to make such distinctions or not. These take the following general form: 'a phrase of type X must be either of subtype Y or subtype Z' or, from a generation point of view, 'if you want to use a phrase of type X, you will have to decide whether it should be of subtype Y or Z'. Thus, in English a clause must be either perfect, progressive, or unspecified for aspect; a noun phrase must be either definite or indefinite; a countable noun phrase must be either singular or plural. This is the **paradigmatic** aspect of language which linguistic theories have tended to focus on; I will refer to this type of processing as **paradigm-driven**. Note that it appears not only in the grammar of the language; a similar type of knowledge forces speakers to choose between *come/bring* (roughly 'towards the speaker or hearer') and *go/take* when generating clauses referring to a physical transfer.

A portion of the paradigm for English NPs is shown in Figure 5.3. An NP is either a pronoun referring to the speaker (*me*), a pronoun referring to the hearer (*you*), a pronoun referring to something other than the speaker and hearer which the hearer is currently conscious of (*him, her, it, them*), or a full NP (*the pepper*).[10]

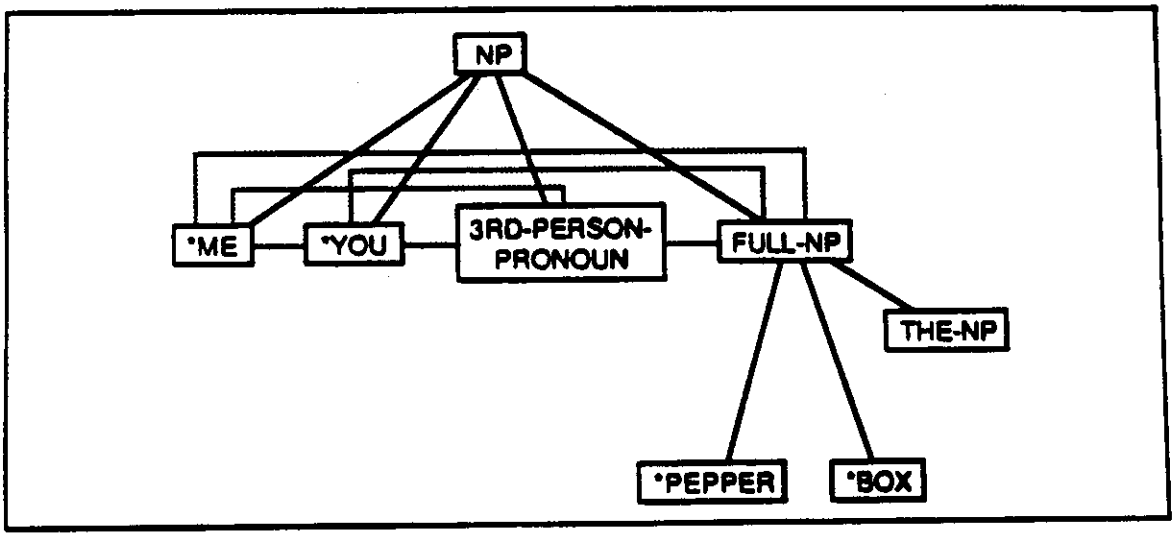

Figure 5.3: Portion of English NP Paradigm

The way the paradigm comes into play in the generation of an NP is roughly as follows. If the referent (the thing currently bound to NP:CONTENT) is the speaker, then *ME will be selected, and it will inhibit the alternatives. If the referent is the hearer, then *YOU will be selected, and it will inhibit the alternatives. If the referent is something which the hearer is thought to be currently conscious of, then 3RD-PERSON-PRONOUN will be selected, and the alternatives will be inhibited. This GU has subtypes specifying *him*, *her*, *it*, and *them*. If neither *ME, *YOU, nor 3RD-PERSON-PRONOUN fires, then the hub node

[10] In most of the examples in this thesis, the hierarchy is simplified, and it is assumed that GUs such as *PEPPER inherit directly from NP.

for the WTA network joining the nodes fires. This sends activation to all four members, but FULL-NP, the default, gets the most and fires, inhibiting the others.

In what follows I illustrate the selection of *YOU and *HIM for the sentences *Mary loves you* and *Mary loves him.*

Early on in the generation of any sentence the HEARER role for that sentence is bound. The process is essentially the same as that described in the last section, but here the role association is based on information contained in a GI rather than a GU. If the speaker's goal is to get a person to perform an act for him, then that person becomes the hearer of a request or command. If the speaker's goal is to get a person to believe some fact, then that person becomes the hearer of a declarative utterance.

Consider the binding of the HEARER role for the sentence *Mary loves you*, an example of the latter type. Figure 5.4 shows the representation of the input and the role association process that takes place once the GI is selected.

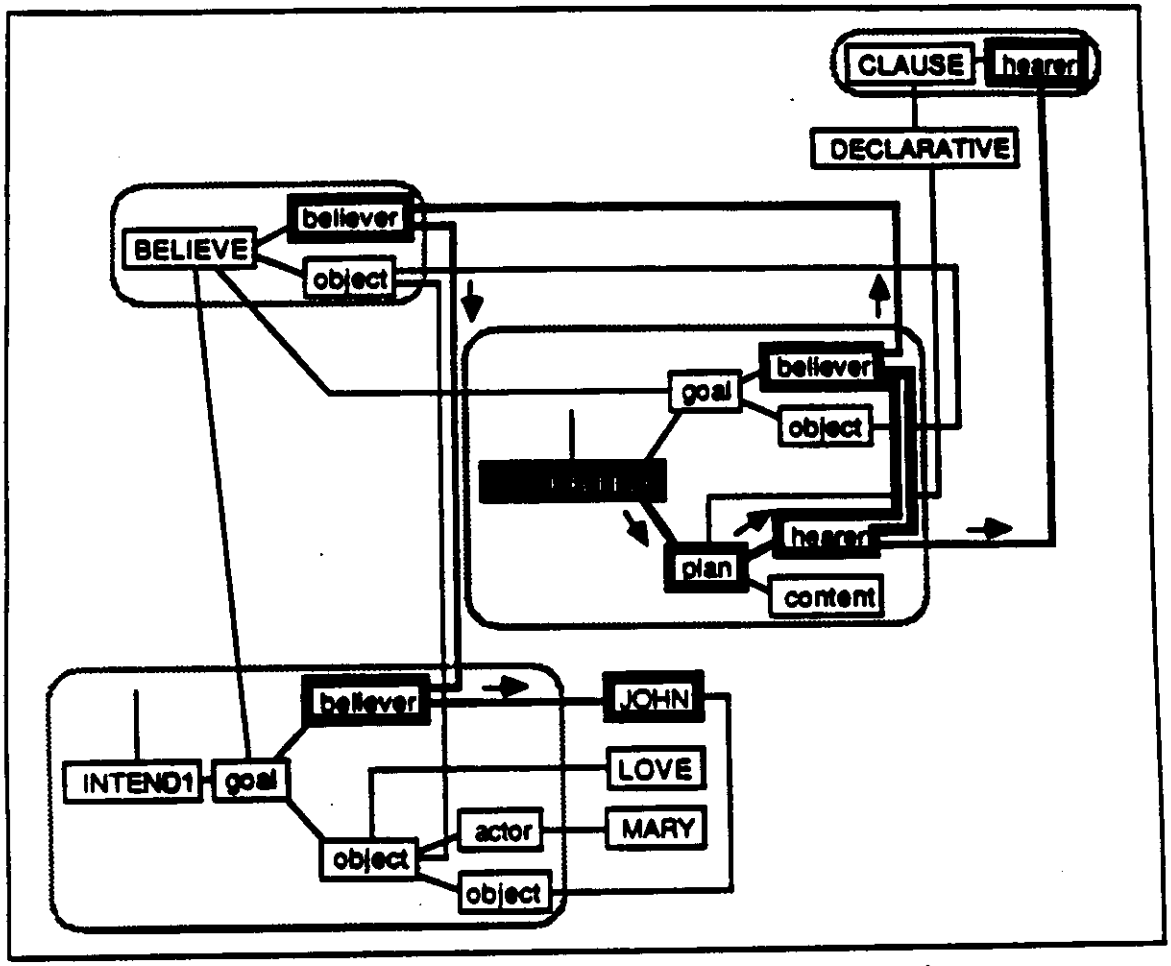

Figure 5.4: Binding of the HEARER Role for an Assertive

The input to the generation is INTEND1, an instance of the general INTEND schema which has as its GOAL that JOHN believe that MARY loves JOHN. Note that JOHN fills two roles in INTEND1; he is the intended BELIEVER of the fact and the OBJECT of MARY's love. It is this dual role that later results in the use of *you* to refer to JOHN. The GI selected for

the utterance is ASSERTIVE, which associates a GOAL that someone believe a particular fact with a PLAN to produce a DECLARATIVE utterance with the fact as its CONTENT and the person who is to believe the fact as the HEARER. The merged node in the ASSERTIVE GI represents the association of the HEARER with the intended BELIEVER. When ASSERTIVE is selected, the merged node fires, resulting in a role binding process. Activation spreading in one direction leads to the firing of JOHN and in the other direction to the firing of CLAUSE:HEARER.

The binding of JOHN to CLAUSE:HEARER serves no function at this point. It plays a role later in the selection of a GU for the final NP in the sentence, the one that refers to the person that MARY loves. The binding is accessible then because there is a temporary path of primed nodes connecting JOHN and CLAUSE:HEARER. I will call this the **binding path**. In this case it consists of JOHN, INTEND1:GOAL:BELIEVER, BELIEVE:BELIEVER, ASSERTIVE:GOAL:BELIEVER (the merged node), and CLAUSE:HEARER. These nodes are primed because the firing of a node always results in residual activation following the refractory period of the node. Figure 5.5 shows how the binding path is used in the selection of the *YOU GU for the last NP in the sentence.

The selection process begins as always with the firing of the node for the referent, JOHN, and the nodes NP and NP:CONTENT. The NP GU has a subtype *YOU which includes the information that the HEARER and CONTENT are the same person. A merged node encodes this knowledge. Some activation spreads from NP:CONTENT to the merged node at this point but not enough for it to fire. Activation from JOHN reactivates the binding path, resulting eventually in the firing of CLAUSE:HEARER. In the figure we see in the CLAUSE GU the additional information that the CLAUSE:HEARER is the same person as the HEARER for each of the constituents of the CLAUSE. Thus CLAUSE:HEARER itself is a merged node. The firing of CLAUSE:HEARER then sends enough activation to NP:HEARER for it to fire, and this node in turn sends additional activation to the merged node in *YOU, which can now fire. The merged node then leads to the firing of the head node *YOU and eventually to the production of the word *you*.

Note that the generation of this sentence involves two separate role equivalences. One, represented by the merged node in the ASSERTIVE schema, equates the BELIEVER, the person who is supposed to believe that Mary loves John, and the HEARER, the person the sentence is being spoken to. This equivalence is true for any assertive, and in generating an assertive, a binding path will always be created which joins the CLAUSE:HEARER role to the value of the BELIEVER role in the input intention, e.g., JOHN in Figures 5.4 and 5.5. The other equivalence, represented by the merged node in the *YOU schema, equates the NP:CONTENT, the thing currently being referred to, and whoever the HEARER is. In generation this equivalence is used in the opposite way from the other one. Rather than being accessed once a schema has been selected, it is a condition for selecting the schema. In comprehension, on the other hand, the information represented by the merged node in the *YOU schema is used after that schema is accessed, resulting in a role binding process associating the hearer with the referent.

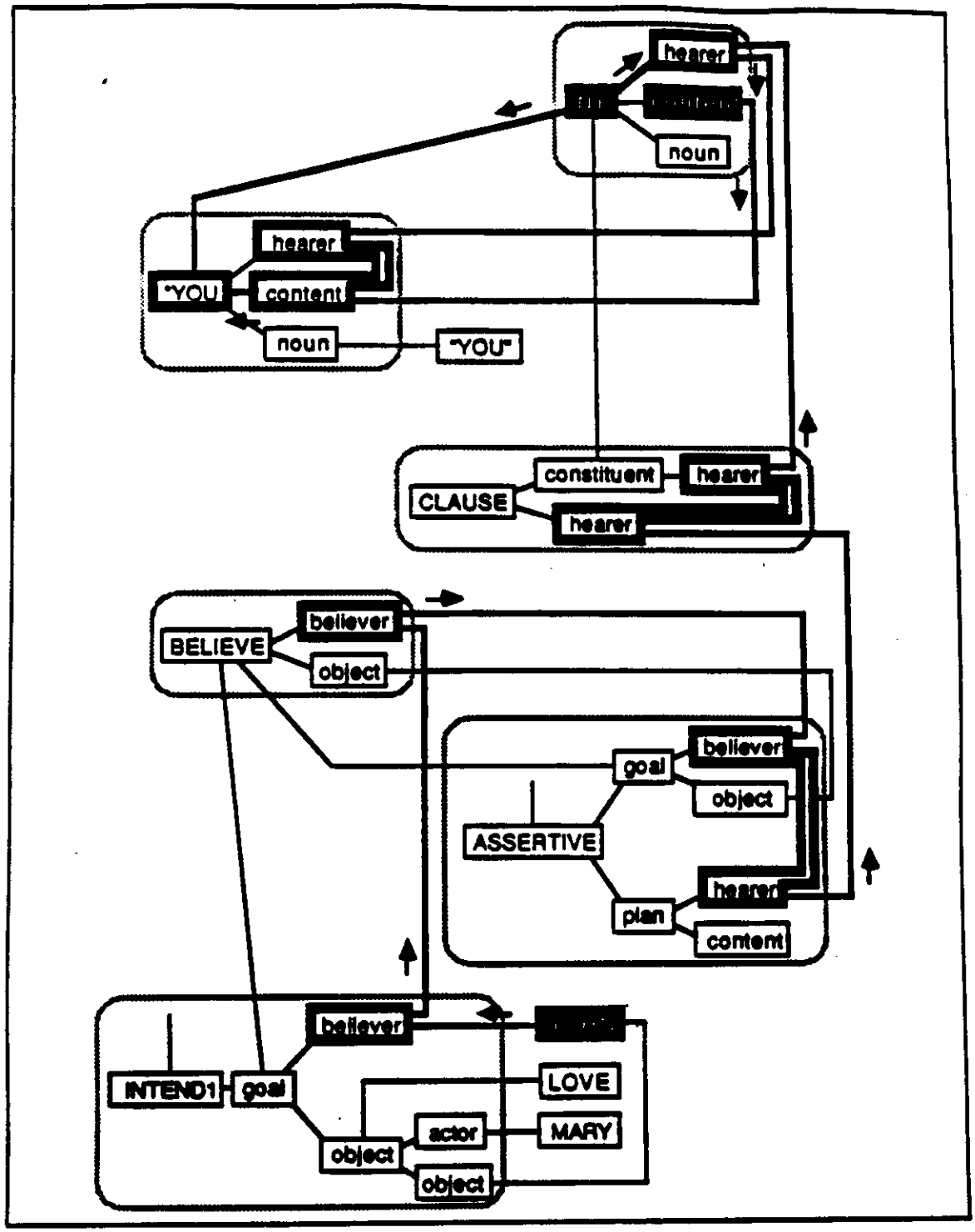

Figure 5.5: Using the Binding of the HEARER Role in Selecting *YOU

Now consider how an anaphoric pronoun is selected. This decision is based on whether the referent is currently in the hearer's consciousness, for example, because it has just been referred to. Thus in generating the sentence *Mary loves him*, the speaker selects

him over *John* if the previous sentence was, say, *remember John?*. The assumption is that the first reference to John makes the hearer temporarily conscious of him and that while this consciousness lasts, a pronoun is appropriate for another reference to John. The speaker needs a way of recognizing this when she wants to refer to John in the second sentence. In brief, the process works as follows. The production of every NP has as its EFFECT a state of the type CONSCIOUS-OF with the current HEARER as its CONCIOUS argument and the current referent (NP:CONTENT) as its OBJECT. This state is not explicitly instantiated, however; it is represented as a pair of binding paths for the roles of the general CONSCIOUS-OF schema. Thus for a short time the fact is accessible. If, during this time, the speaker chooses to refer to the same referent again, the CONSCIOUS-OF schema will be reactivated and will send activation to the 3RD-PERSON-PRONOUN schema, causing it to be selected over the alternatives.

Figure 5.6 shows how the CONSCIOUS-OF fact is created during the generation of the first NP.

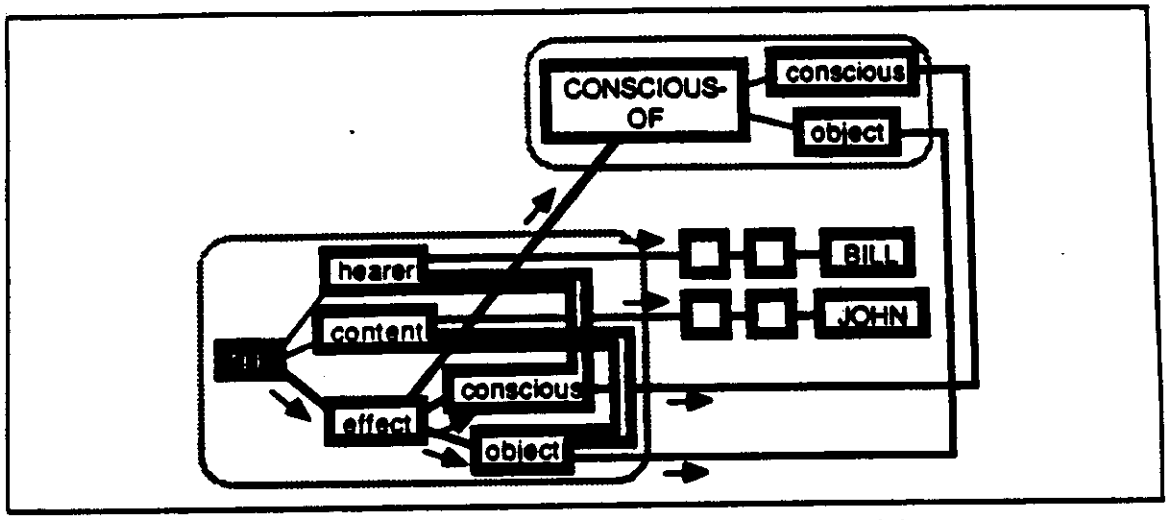

Figure 5.6: EFFECT of an NP: The HEARER is Conscious of the CONTENT

We have already seen how the CONTENT of an NP (the example in Figures 5.1 and 5.2) and the HEARER of an NP (the example in Figures 5.4 and 5.5) are associated with particular values during the generation of a sentence. In both cases a binding path of primed nodes is left behind, making the associations temporarily retrievable from one end of the path or the other. To simplify the remaining figures, I will indicate a binding path by a pair of connected squares joining the role and its value. This is not meant to indicate that the path contains two nodes; it may contain any number of nodes.

The NP GU has an EFFECT role which points off to the CONSCIOUS-OF schema. The roles of the EFFECT are merged with the HEARER and CONTENT roles of the NP. When an NP is produced, the EFFECT role fires, causing the roles in CONSCIOUS-OF to fire and, via the binding paths, the nodes representing the current HEARER and CONTENT. For our example, the CONTENT is JOHN, and the HEARER is BILL. The result is two somewhat longer binding paths connecting the CONSCIOUS and OBJECT roles of CONSCIOUS-OF to BILL and JOHN respectively. As long as the nodes on these paths remained primed, the associations are retrievable.

Figure 5.7 shows these paths and the relevant portion of the 3RD-PERSON-PRONOUN schema.

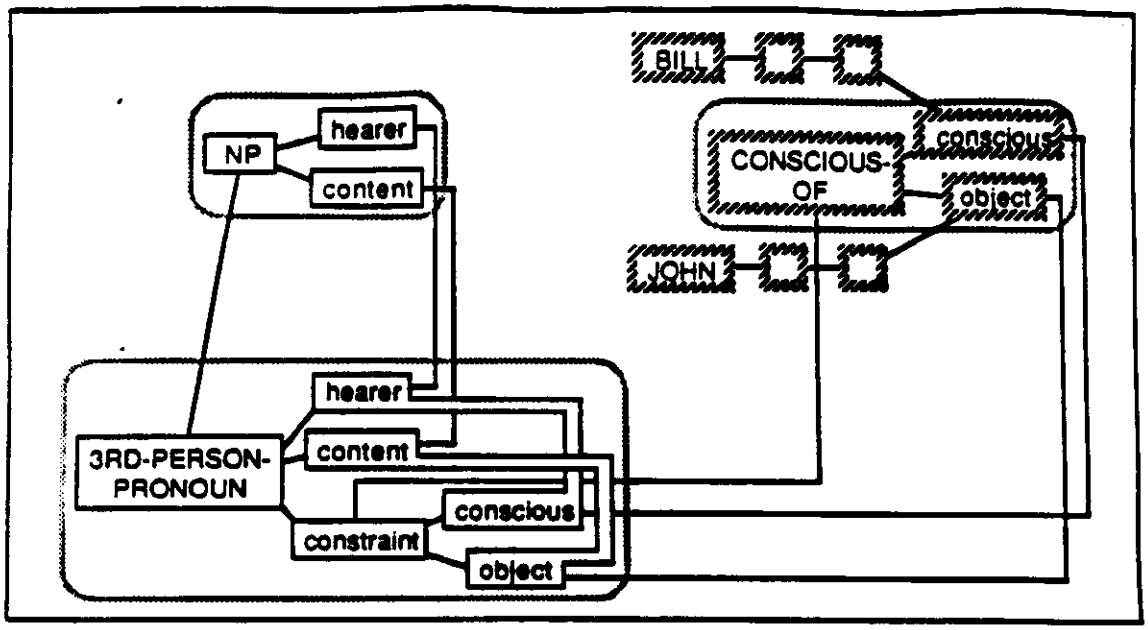

Figure 5.7: 3RD-PERSON-PRONOUN GU and Binding Paths Remaining After an NP

The 3RD-PERSON-PRONOUN schema has a CONSTRAINT role with the same form as the EFFECT role in the NP schema, shown in Figure 5.6. In this case, however, the information will be used in matching the GU rather than as a consequence of selecting it.

Figure 5.8 shows some of the network involved in the generation of the sentence *Mary loves him*. As in the example shown in Figure 5.4 and 5.5, the ASSERTIVE GI has been selected on the basis of the speaker's goal. In this case, however, the CONTENT of the last NP is not the same as the current HEARER, so we do not get the activation of a path of nodes leading to NP:HEARER and eventually the selection of the *YOU schema.

Figure 5.9 shows what happens when the generation of the NP is initiated. Activation from NP and NP:CONTENT primes the head and one of the merged nodes in the 3RD-PERSON-PRONOUN GU. Activation spreading from JOHN, the referent of the NP, along the primed binding path causes CONSCIOUS-OF:OBJECT and then CONSCIOUS-OF to fire. CONSCIOUS-OF:OBJECT sends additional activation to the primed merged role in the 3RD-PERSON-PRONOUN GU. At this point half of the constraint is satisfied.

The remainder of the process that selects 3RD-PERSON-PRONOUN appears in Figure 5.10. The head node of the CONSCIOUS-OF schema sends some activation to the CONSTRAINT role in the 3RD-PERSON-PRONOUN GU and also causes CONSCIOUS-OF:CONSCIOUS to fire. This node activates the other merged node in the GU and initiates the activation of the other binding path. This results in the firing of NP:HEARER, providing the additional activation needed for the second merged node to fire. Now both parts of the constraint are satisfied, the CONSTRAINT role fires, leading to the firing of 3RD-PERSON-PRONOUN and the inhibition of the other subtypes of NP. The generation of *him* results from the later selection of the *HIM schema, a subtype of 3RD-PERSON-PRONOUN which specifies that its CONTENT be MALE and SINGULAR.

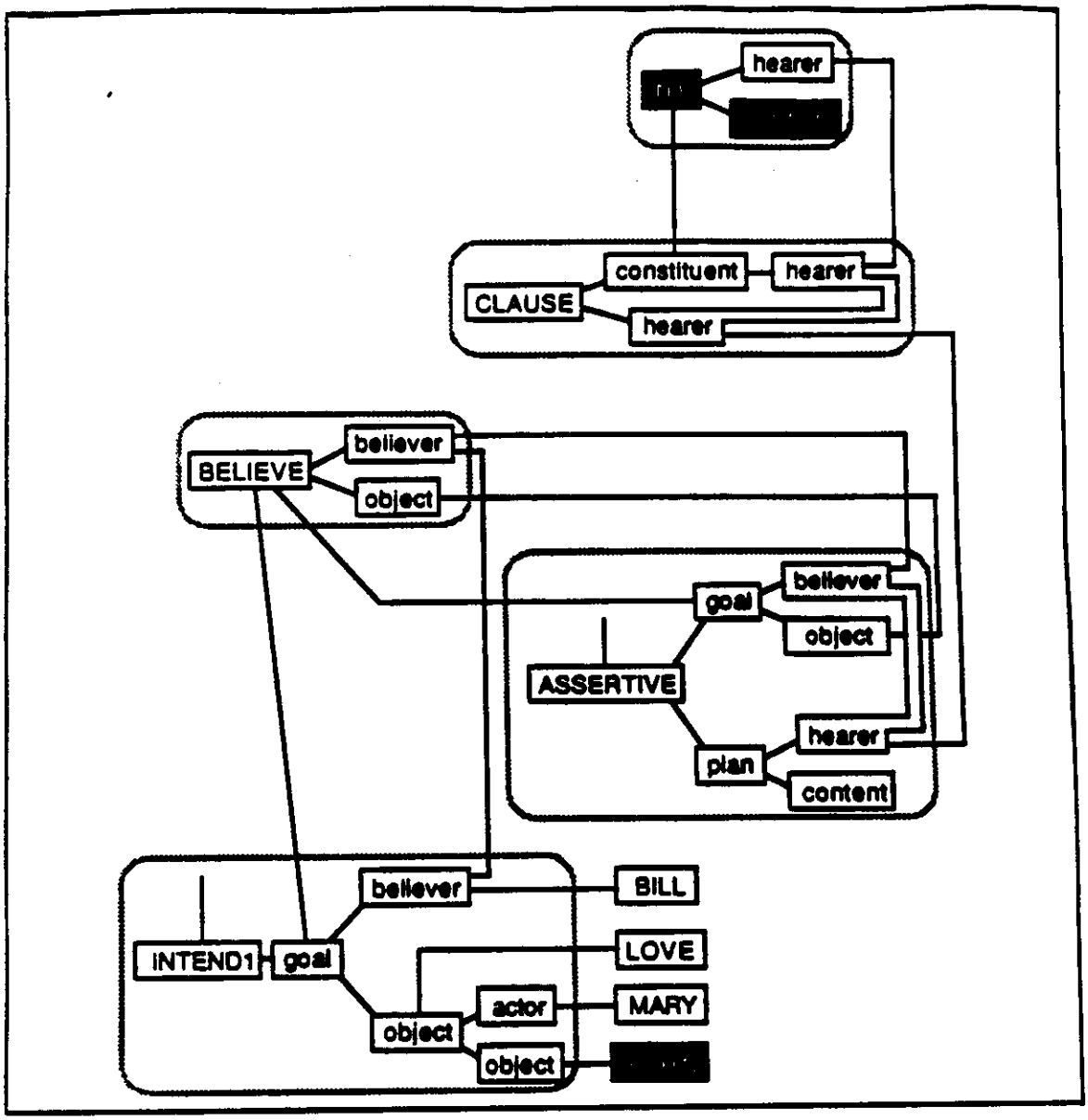

Figure 5.8: Generation of an Anaphoric Pronoun (1)

5.2.2. Flexibility: Coping with Alternative Input Representations

A concept can in general be represented in various ways, and particular languages may prefer one over another. In cases when the input is in the form which does not match any schemas in linguistic memory, there needs to be a conceptual "translation" of sorts. Spreading activation, including the role association process I have described, can accomplish this without any additional mechanisms.

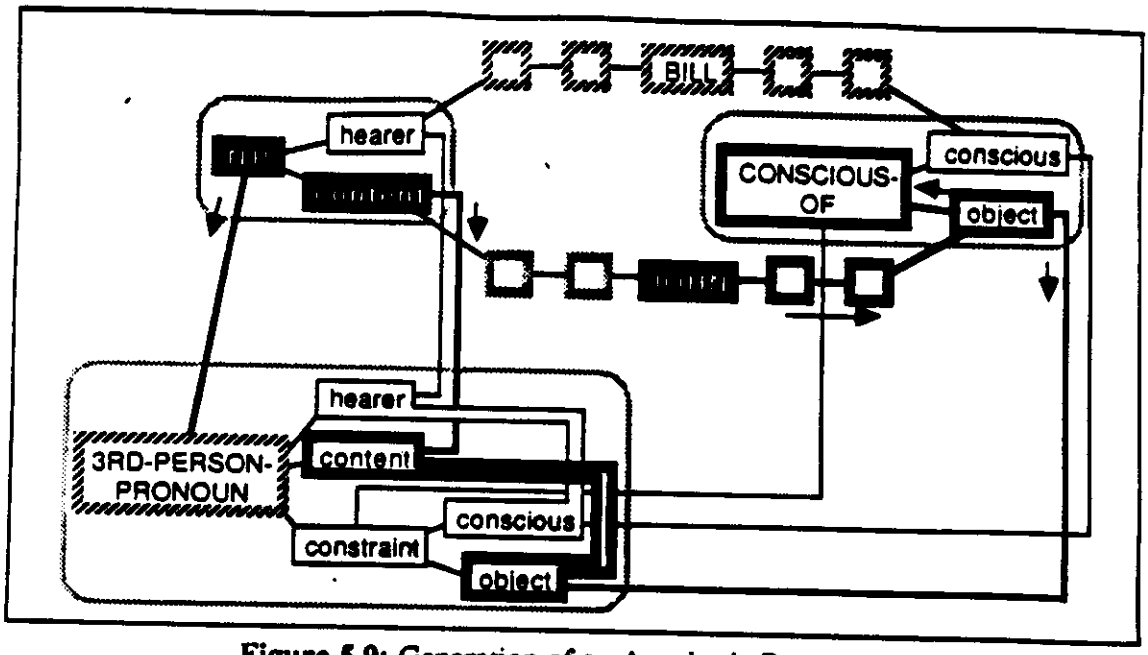

Figure 5.9: Generation of an Anaphoric Pronoun (2)

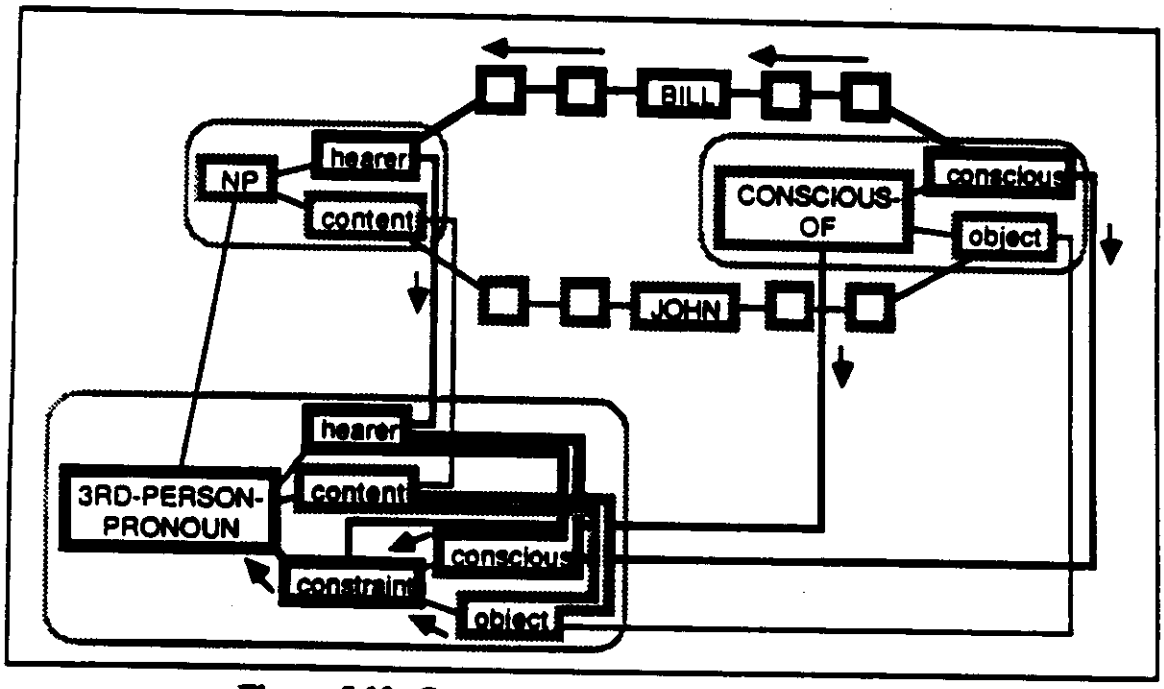

Figure 5.10: Generation of an Anaphoric Pronoun (3)

Consider again sentences (1.5), repeated here as (5.1).

(5.1a) *Mary had John cook dinner.*

(5.1b) *Mary wa John ni yuuhan o tukur-ase-ta.*
Mary TOPIC John AGENT dinner ACCUS make-cause-PAST

Whereas English uses *have* with an embedded complement clause to refer to the cooking, Japanese has a special causative form of the verb meaning 'cook'. There is no embedded structure in the Japanese sentence, which would translate literally into English as 'Mary caused-to-cook dinner by John'.

Figure 5.11 shows how the different GUs are represented. The causative form of the Japanese verb is given its own schema, *TUKURASERU. Both GUs have as their CONTENT an instance of the concept CAUSE-ACT, which has an ACT as its OBJECT. For the Japanese GU the OBJECT of the CONTENT points to the PREPARE-FOOD schema.[11] The English GU has a COMPLEMENT constituent, whose CONTENT is the OBJECT, that is, the ACT being caused. This information is represented by a merged node. The figure also shows the English GU *COOK, which is also needed for sentence (5.1a).

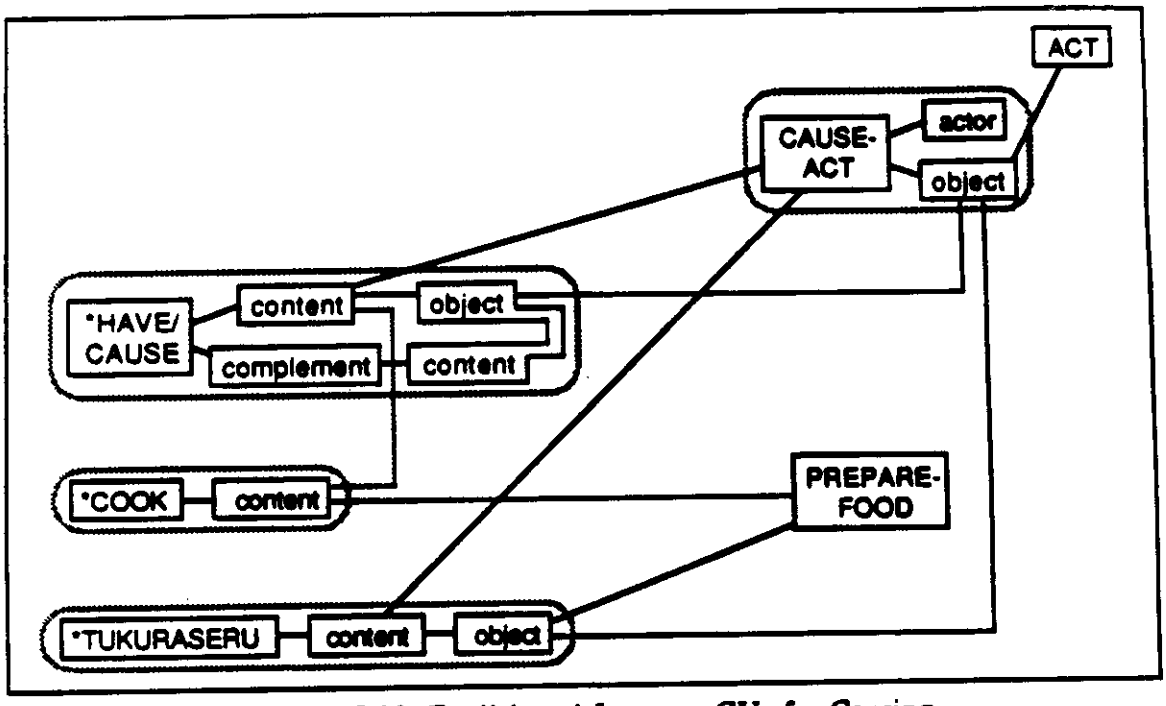

Figure 5.11: English and Japanese GUs for Causing

Figures 5.12 and 5.13 show how schema selection occurs for sentence (5.1a). The input to the selection process is the instance of CAUSE-ACT shown in the lower right corner of the figures. The CAUSE-ACT node is activated, and this leads to the firing of *HAVE/CAUSE:CONTENT and the selection of that GU. On the basis of the OBJECT of the input concept, *COOK:CONTENT also receives some activation, but this arrives somewhat later than for *HAVE/CAUSE:CONTENT, so *HAVE/CAUSE wins out. Note that the two CONTENT roles inhibit one another.

In Figure 5.13 we see the selection process initiated for the complement of the clause. The merged node representing the equivalence of the COMPLEMENT:CONTENT and the OBJECT of the causing fires spreading activation to CLAUSE:CONTENT in one direction and to the input OBJECT and PREPARE-FOOD in the other. Activation from these two nodes converges on *COOK:CONTENT, leading to the selection of this GU.

[11] The verb *tukuru* (causative *tukuraseru*) is actually much more general than this, but it is reasonable to assume that there is a specific GU for this sense.

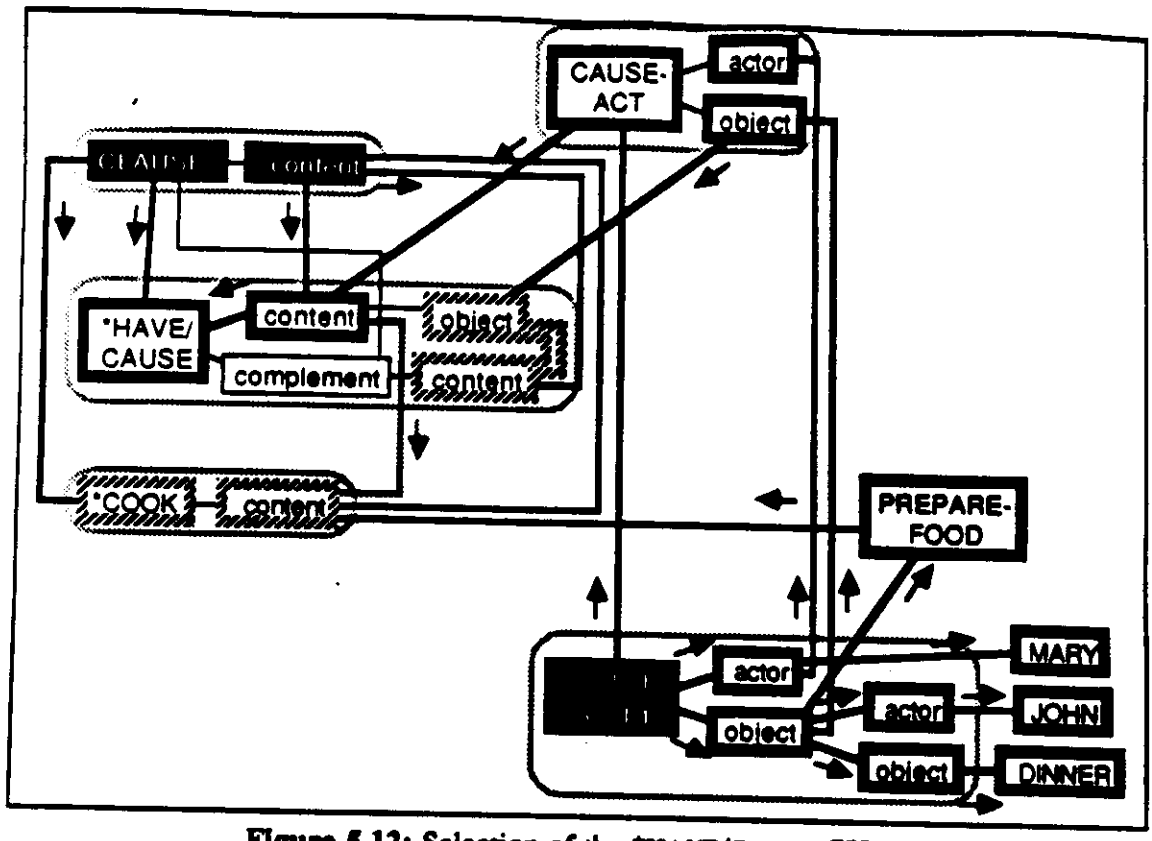

Figure 5.12: Selection of the *HAVE/CAUSE GU (1)

Figure 5.14 illustrates the comparable process for the Japanese sentence, (5.1b). On the basis of CAUSE-ACT, the CONTENT roles of all causative verb GUs are activated, but none can fire at this point. Activation starting from the OBJECT of the input concept, however, provides the extra amount needed for *TUKURASERU:CONTENT to fire and inhibit its competitors.

5.2.3. Context-Driven Generation

In Section 3.2.3 I discussed the type of generation which is driven by contextual features rather than by a goal of the speaker. Consider again the example of *speak of the devil*. Briefly, what we want to happen is the following. During a conversation, an association is set up between the node for the thing under discussion and the TOPIC node. When that thing is then seen by one of the conversants, activation converges on the PERCEIVE:SPEAK-OF-THE-DEVIL schema from two directions. One source of activation is the general PERCEIVE schema. The other is the TOPIC role which fires because the node for the thing seen has fired and its association with TOPIC is still accessible via a primed binding path.

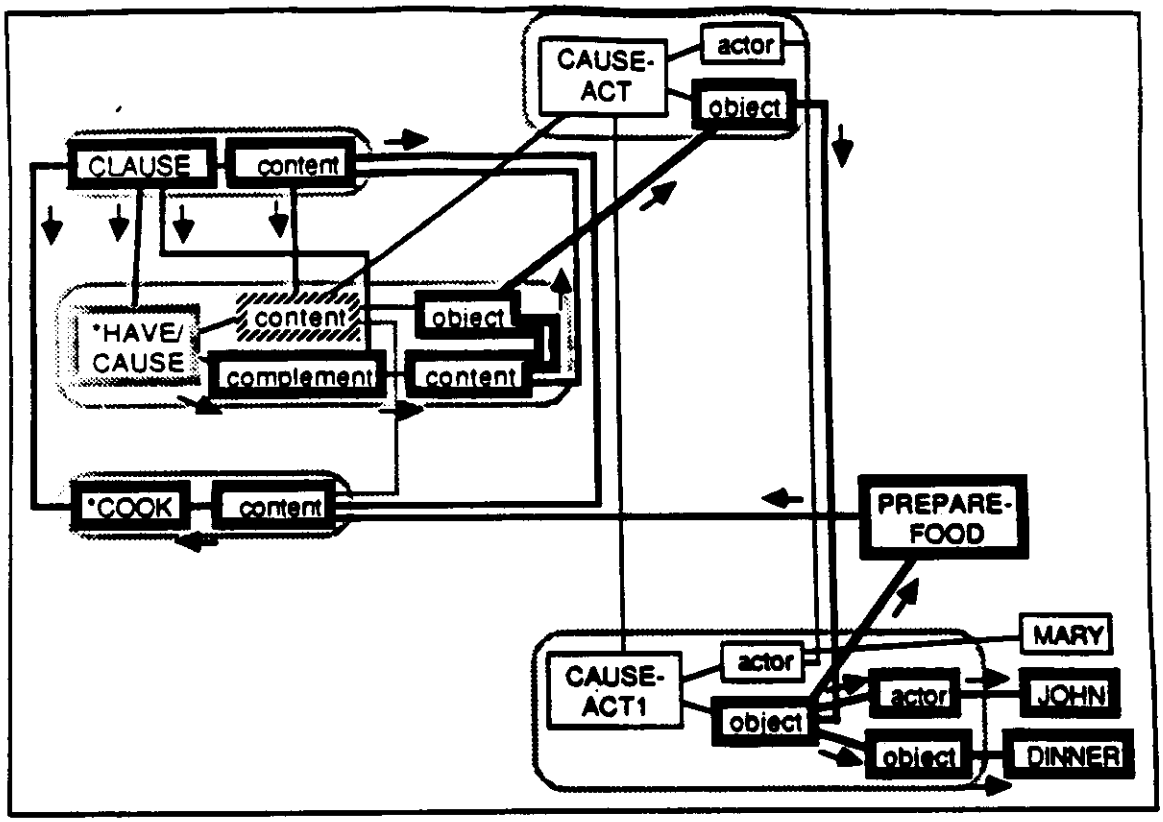

Figure 5.13: Selection of the *HAVE/CAUSE GU (2)

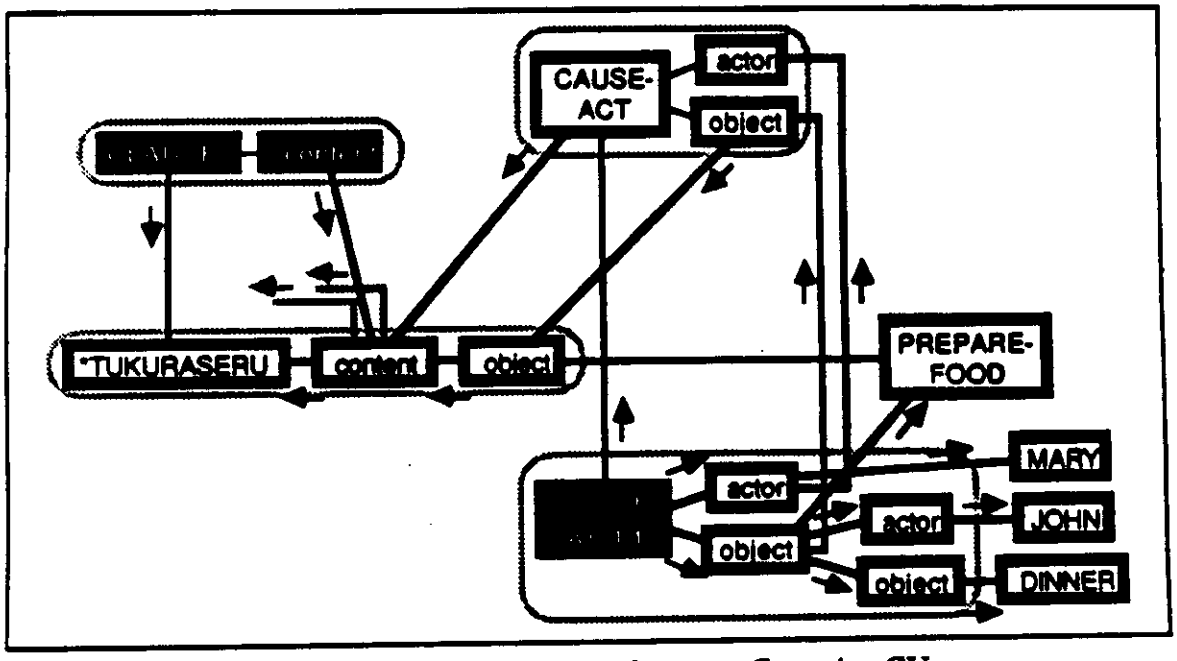

Figure 5.14: Selection of a Japanese Causative GU

In order for this process to take place, we need to add to the model the means by which the current context is updated to reflect the occurrence of and recognition of new external events. First, the system needs to be frequently reminded of what sort of context it is in. This reminding consists of causing the generic node for CONTEXT (a role in the EVENT schema) and a set of nodes representing the current context to fire at fixed intervals. Contexts are situational entities such as BASEBALL-GAME, CLASSROOM, and CONVERSATION. During a conversation, the generic CONVERSATION node fires along with the CONTEXT node.

When a conversation is taking place, the conversants are aware of changes in topic. This awareness corresponds to a role association process of the type described above: the node representing the new topic and the TOPIC node in the CONVERSATION schema both fire, and a path of activated nodes is temporarily left behind, representing the association. Precisely what a conversational topic is has not been addressed in the model; that is, the connections that enable the association process do not actually appear in memory. Presumably they would associate the content of certain types of noun phrases with the TOPIC node. For the purposes of generating *speak of the devil*, I will simply assume that the association of TOPIC has somehow taken place.

We also need a way to represent new events that the system is aware of. In the model this is simulated by causing the nodes in the schema for a general event type to fire along with the appropriate values for the roles in the schema. For instance, perceptual events are represented by the firing of the following nodes: PERCEIVE, PERCEIVE:OBJECT plus the value for this role, and PERCEIVER plus SELF (the node for the system itself). Because of the spreading activation, which automatically takes place as a result of the firing of these nodes, the system in effect classifies the new event as an instance of a lower-level schema on the basis of the features of the event.

It is through this process of automatic classification that the system accesses the schema which yields *speak of the devil*. Figure 5.15 shows a portion of the network just before the event which triggers this expression. A conversation is underway, so the CONVERSATION and CONTEXT nodes have fired. The current topic of conversation is John, so JOHN is temporarily associated with TOPIC via an unspecified binding path. Missing in the figure are the details of the CONSEQUENCE role of PERCEIVE:SPEAK-OF-THE-DEVIL. This is the utterance which results from the perception.

The perception of John is represented in the system by the firing of PERCEIVE, PERCEIVER, PERCEIVE:OBJECT, SELF, and JOHN.[12] What follows is shown in Figure 5.16. PERCEIVE:OBJECT activates the merged node representing the equivalence of the OBJECT role and the CONTEXT:TOPIC role in the PERCEIVE:SPEAK-OF-THE-DEVIL schema, and JOHN activates TOPIC because of the existing association. When TOPIC fires, it sends enough additional activation to the merged node in the PERCEIVE:SPEAK-OF-THE-DEVIL schema for this to fire. This causes the head node of the schema to fire and then the node for the CONSEQUENCE of the perception, which leads to the utterance of *speak of the devil* by the perceiver.

[12] Actually, to avoid crosstalk, there must be a delay between (a) the firing of one of the roles and its value and (b) the firing of the other role and its value to avoid crosstalk. Section 5.3 discusses this problem and a way in which the necessary delay can be implemented. For the purposes of the discussion here, we can ignore this complication.

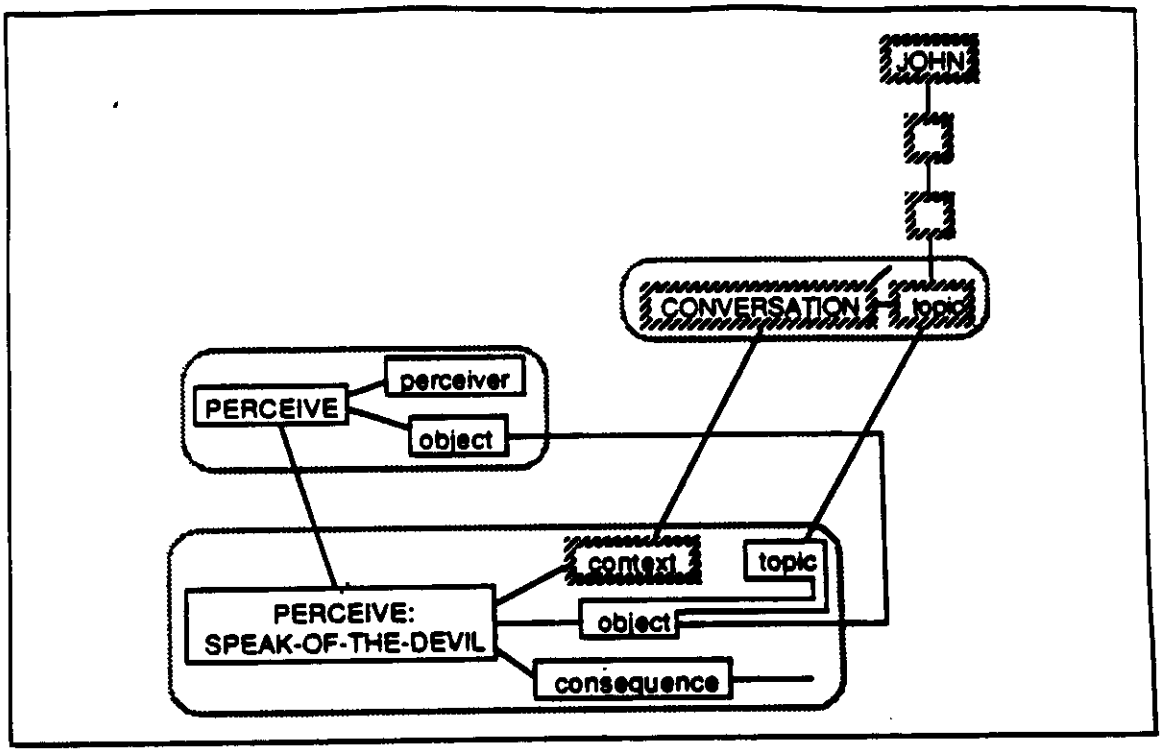

Figure 5.15: Context-Driven Generation: *speak of the devil* (1)

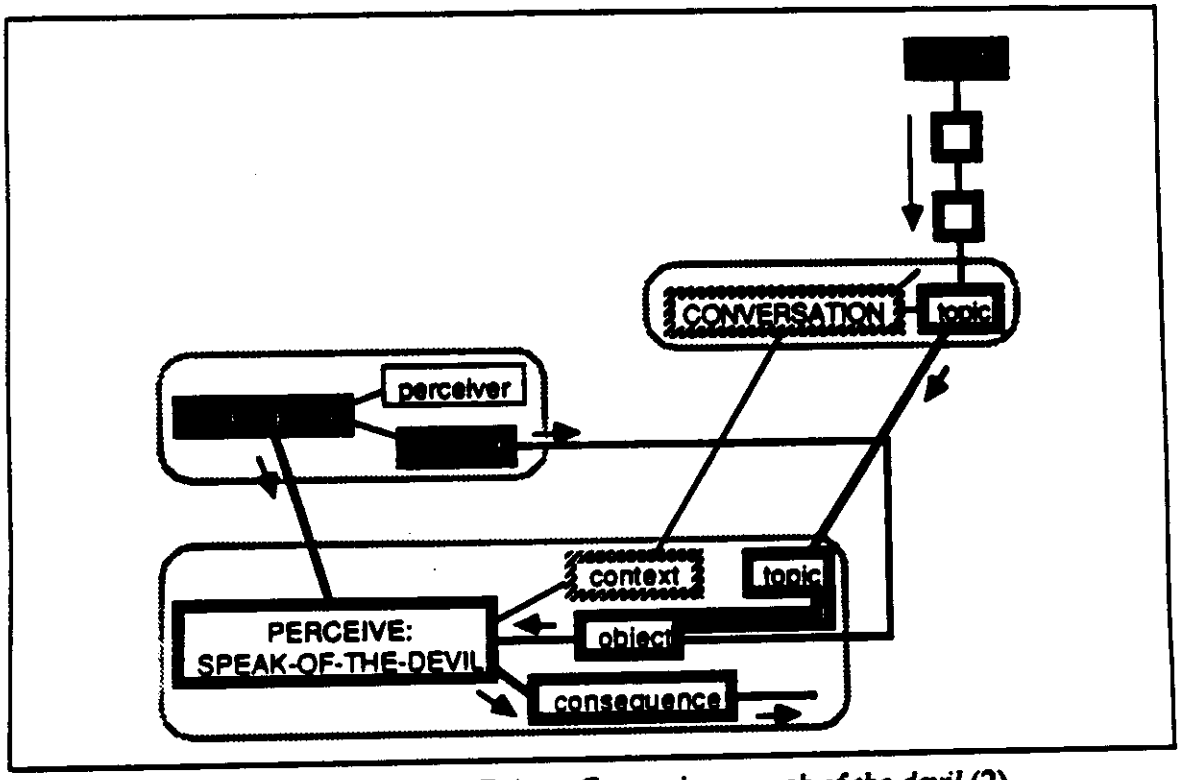

Figure 5.16: Context-Driven Generation: *speak of the devil* (2)

5.2.4. Generating Exchange Errors

A common type of speech error involves the switching of two elements of a particular type. For example, in the following sentence the speaker exchanges two nouns:

(5.2) *I wrote a mother to my letter.*

Errors such as this can be explained if we assume that features relating to elements later in the sentence are primed relatively early on, presumably at a stage when the eventual order of the elements is not yet known. Thus in (5.2) the speaker begins thinking about her mother before she gets to the direct object of the sentence. When she reaches the direct object position and looks for a noun GU to fill it, the GU for mother has more priming than the one for letter and it wins out. The schema for letter retains some activation, however, and is selected over the correct alternative for the final position.

Let us consider the error in more detail. Figure 5.17 shows what takes place as the GU selection process begins for the clause.

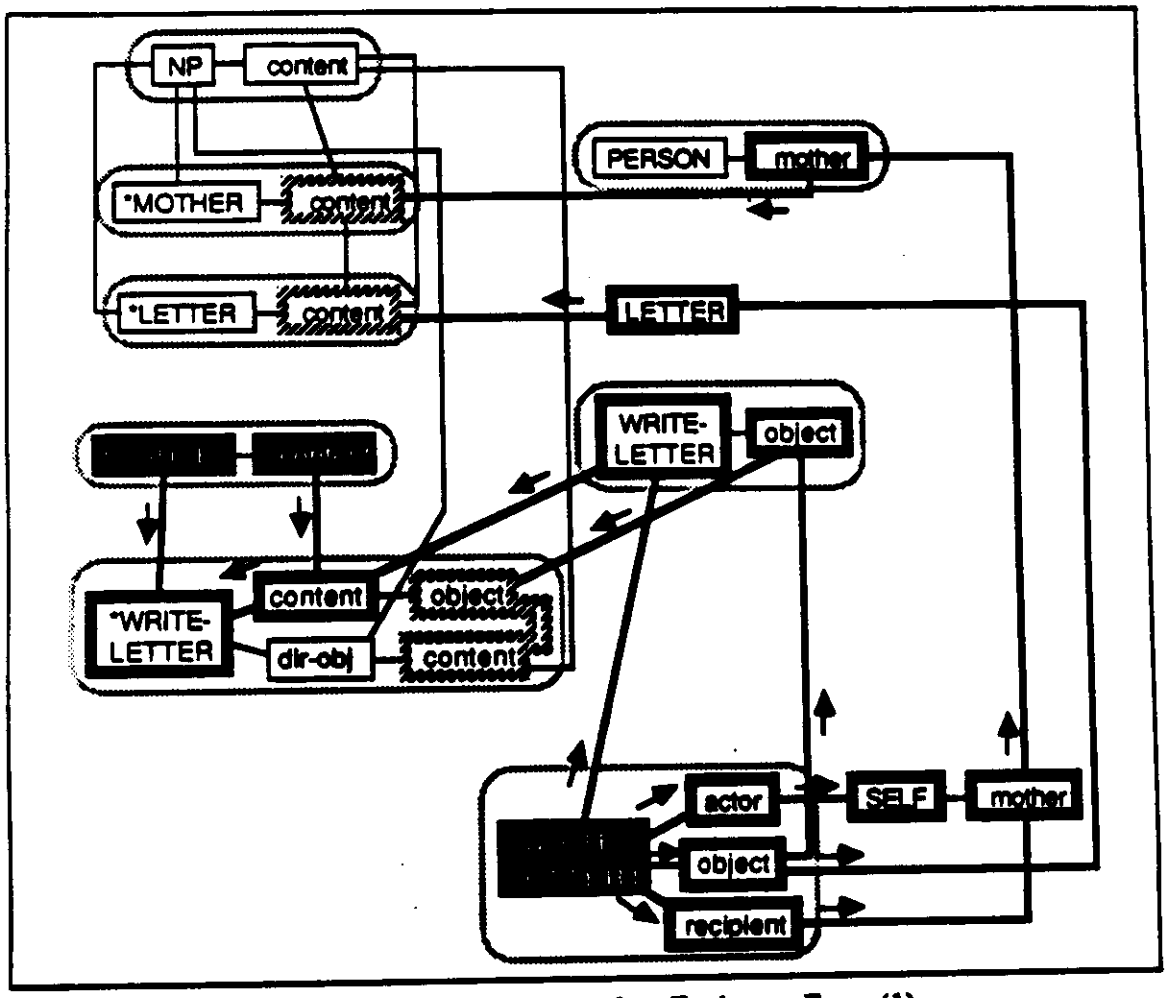

Figure 5.17: Generation of an Exchange Error (1)

Activation spreads from the input concept WRITE-LETTER1. For simplicity, this is represented as an instance of a concept which is directly associated with a GU. The roles of WRITE-LETTER1 all receive enough activation to fire, sending activation to the nodes

representing their values. Both LETTER and MOTHER fire, priming the CONTENT roles for the GUs *LETTER and *MOTHER. For the clause the GU *WRITE-LETTER is selected on the basis of the type for the input concept.

For the generation of the direct object, activation spreads first to the DIRECT-OBJECT role in *WRITE-LETTER. What follows appears in Figure 5.18.

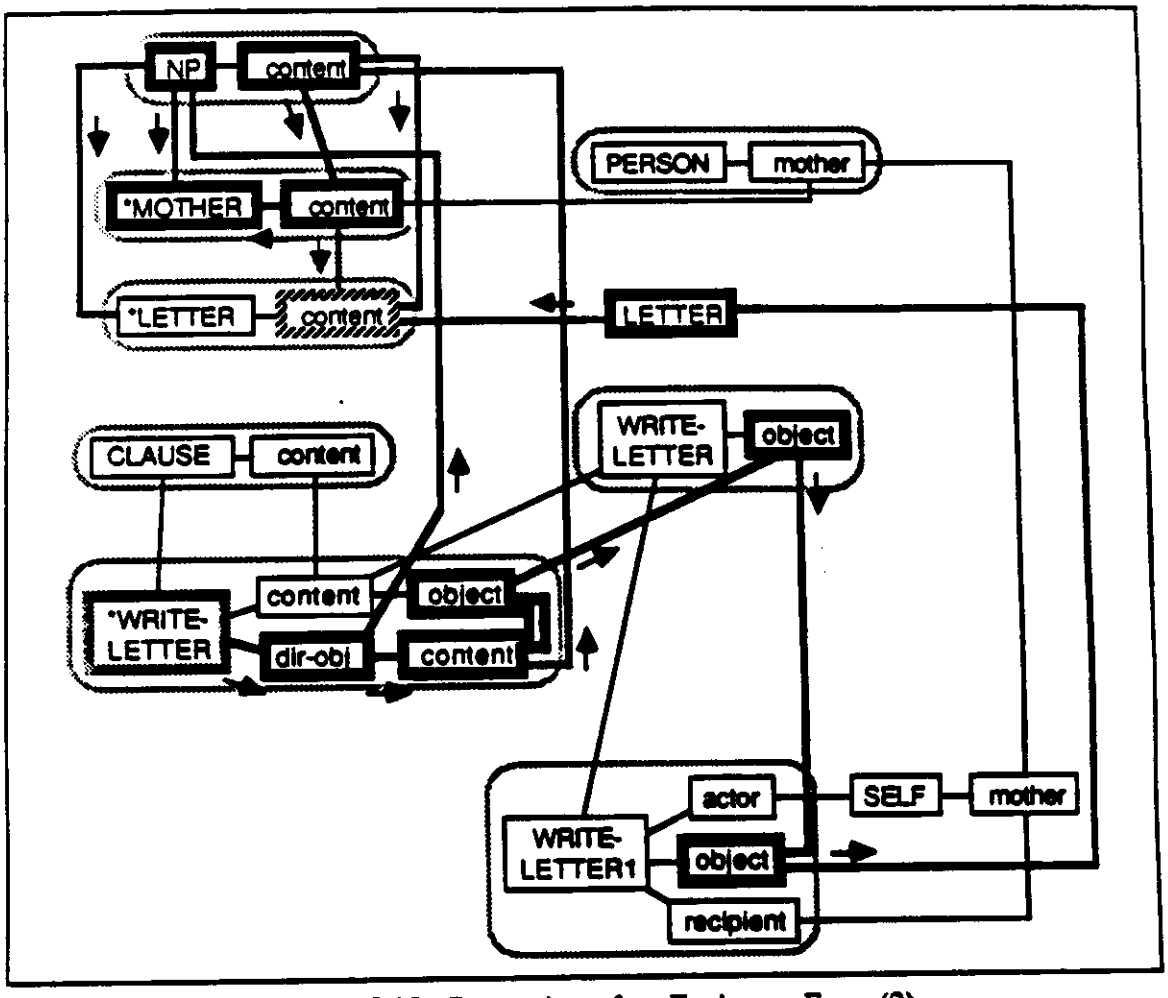

Figure 5.18: Generation of an Exchange Error (2)

Activation reaches the merged node representing the role association, here the fact that the direct object should refer to the OBJECT of the writing. This leads to the firing of NP and NP:CONTENT. NP:CONTENT activates the CONTENT roles of all noun GUs, including *MOTHER and *LETTER. Normally neither of these would have enough activation to fire at this point, but if *MOTHER:CONTENT started out at the beginning of the sentence with some priming, then it might fire. If it does, it inhibits *LETTER:CONTENT (and the CONTENT roles of other noun GUs), and the error results. Note that *LETTER:CONTENT would have fired otherwise because of the additional it would receive from LETTER as activation spread from the merged node via the instance concept.

5.3. Dealing with Crosstalk

There is a serious difficulty with the approach to role binding outlined above. In the example, we assumed that the subject had already been taken care of; that is, the SUBJECT role of the lexical GU had fired and was inhibited. In many cases, however, both the subject and direct object would need to be associated with semantic roles at this point, and there would be nothing to prevent the two processes from happening simultaneously. That is, in the above example, *PASS:SUBJECT and *PASS:DIRECT-OBJECT would fire at more or less the same time. The problem arises because the nodes for the referents of the two NPs would also fire at more or less the same time, and the system would have no way of knowing which was the appropriate referent for which clause argument. It is easier to appreciate the confusion for sentences in which the arguments are of the same general semantic type, for example, *Mary loves John*. Figure 5.19 illustrates the problem.

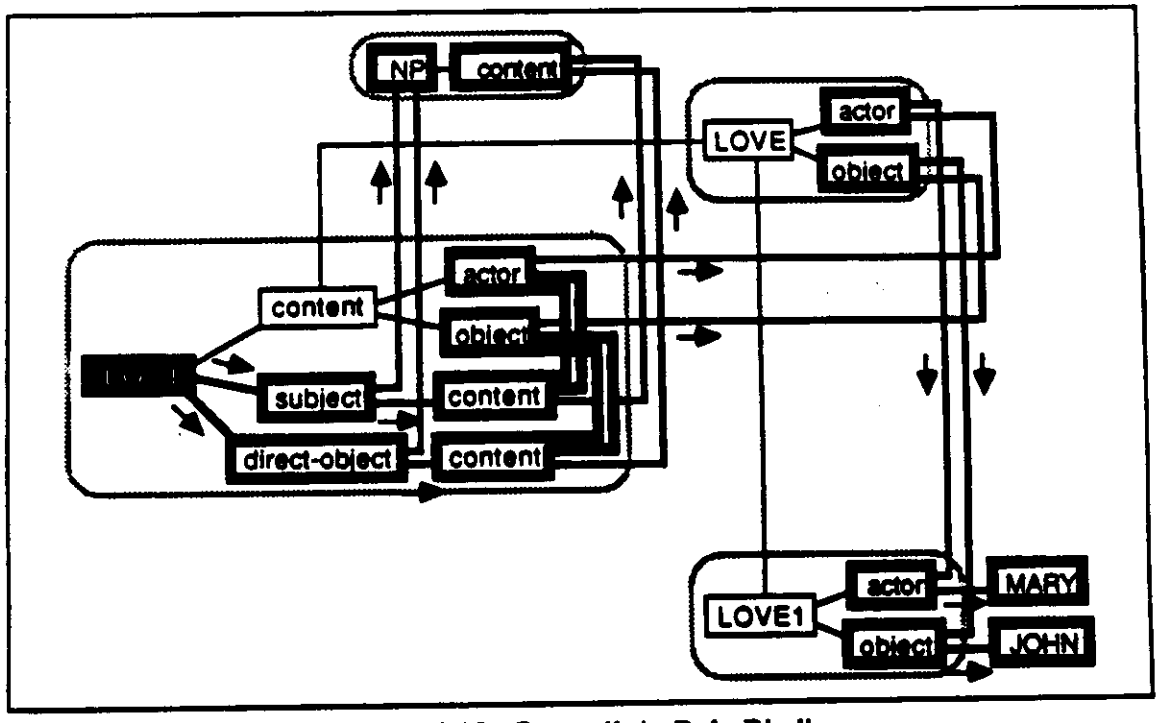

Figure 5.19: Crosstalk in Role Binding

With activation spreading to the SUBJECT and DIRECT-OBJECT at the same time and from there to the two merged roles, the result is the firing more or less simultaneously of MARY and JOHN. Clearly there would be nothing to prevent the system from generating *John loves Mary*.

This problem is known as crosstalk; it is a familiar one in connectionist models (Feldman, 1986). It surfaces in comprehension as well, where, given a sentence such as *Mary loves John*, the system may be unable to decide who loves who. Crosstalk may also arise in schema selection when two or more roles of an input concept are of the same general type. In selecting a GU to refer to a transfer of money, for example, the fact that the RECIPIENT is a BANK contributes to the selection of the GU *DEPOSIT-IN-BANK. If the SOURCE and RECIPIENT roles in the input fire at more or less the same time, however, the system will be unable to distinguish information about the two roles. Thus if the SOURCE

of the transfer is a bank, this information could wrongly contribute to the selection of *DEPOSIT-IN-BANK. Figure 5.20 illustrates the confusion.

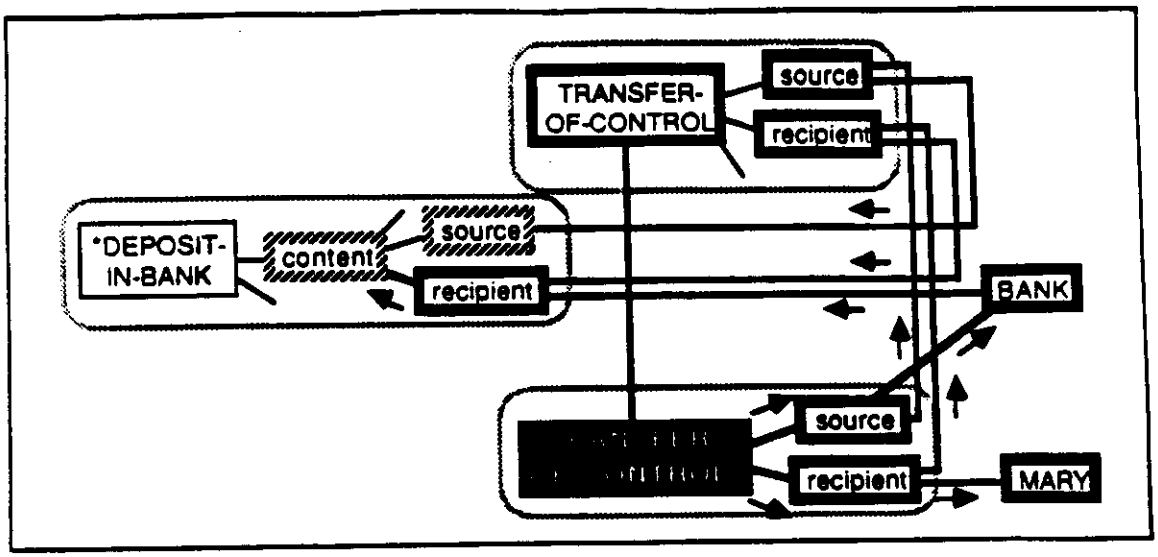

Figure 5.20: Crosstalk in Schema Selection

In this figure an instance of TRANSFER-OF-CONTROL has as its SOURCE a BANK and its RECIPIENT MARY. An appropriate GU for this concept would be *WITHDRAW (if the RECIPIENT is also the ACTOR) but certainly not *DEPOSIT-IN-BANK. Activation spreads simultaneously from TRANSFER-OF-CONTROL3 to its SOURCE and RECIPIENT roles, and these activate the nodes for their values, BANK and MARY, and their general role types, SOURCE and RECIPIENT in the TRANSFER-OF-CONTROL schema. Activation converges on the RECIPIENT role of *DEPOSIT-IN-BANK:CONTENT from BANK and TRANSFER-OF-CONTROL:RECIPIENT, but these two nodes are not actually associated with one another in the input concept. The result is that *DEPOSIT-IN-BANK:CONTENT:RECIPIENT fires and sends activation to its head, providing some support for the use of *deposit* to refer to the transfer. Together with activation from other roles, for example, the one representing the temporary nature of the transfer, the CONTENT role might fire, leading to the erroneous selection of *DEPOSIT-IN-BANK.

A way to handle crosstalk is to arrange for the interfering processes to be staggered. In the example of role binding for the sentence *Mary loves John*, this would mean the firing of MARY and SUBJECT together and then, slighly later, the firing of JOHN and DIRECT-OBJECT. When there is a convention for the ordering of the constituents, the staggering can be accomplished by explicitly representing the sequence relationships, roughly as shown in Figure 5.21. The arrow heads indicate the directions of single connections. The connection from PHRASE to CONSTITUENT1 has a greater weight than that to CONSTITUENT2, so CONSTITUENT1 is more likely to fire first. Once it has fired, it sends additional activation to CONSTITUENT2, causing it to fire. Sequencing in the CLM model is discussed in detail in Chapter 6.

In other cases, sequencing is inappropriate as a way of avoiding crosstalk because there is no convention for which role should precede. For the firing of conceptual roles in schema selection, there is no reason to force a particular sequence on the consideration of these roles. All that is necessary in such cases is something to prevent the roles from firing simultaneously and to insure that all do fire. A way to accomplish for the instances of TRANSFER-OF-CONTROL is shown in Figure 5.22.

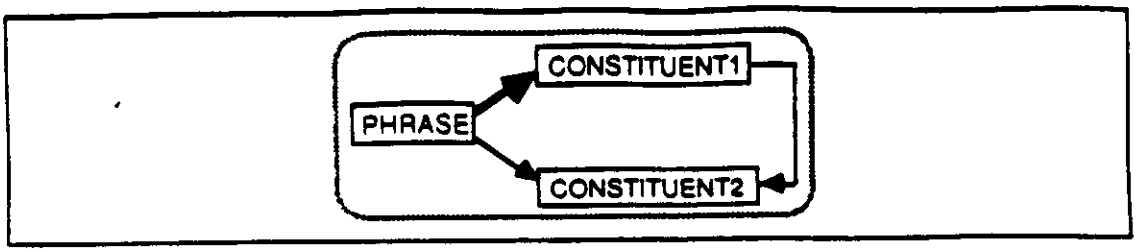

Figure 5.21: Preventing Crosstalk Using Sequencing

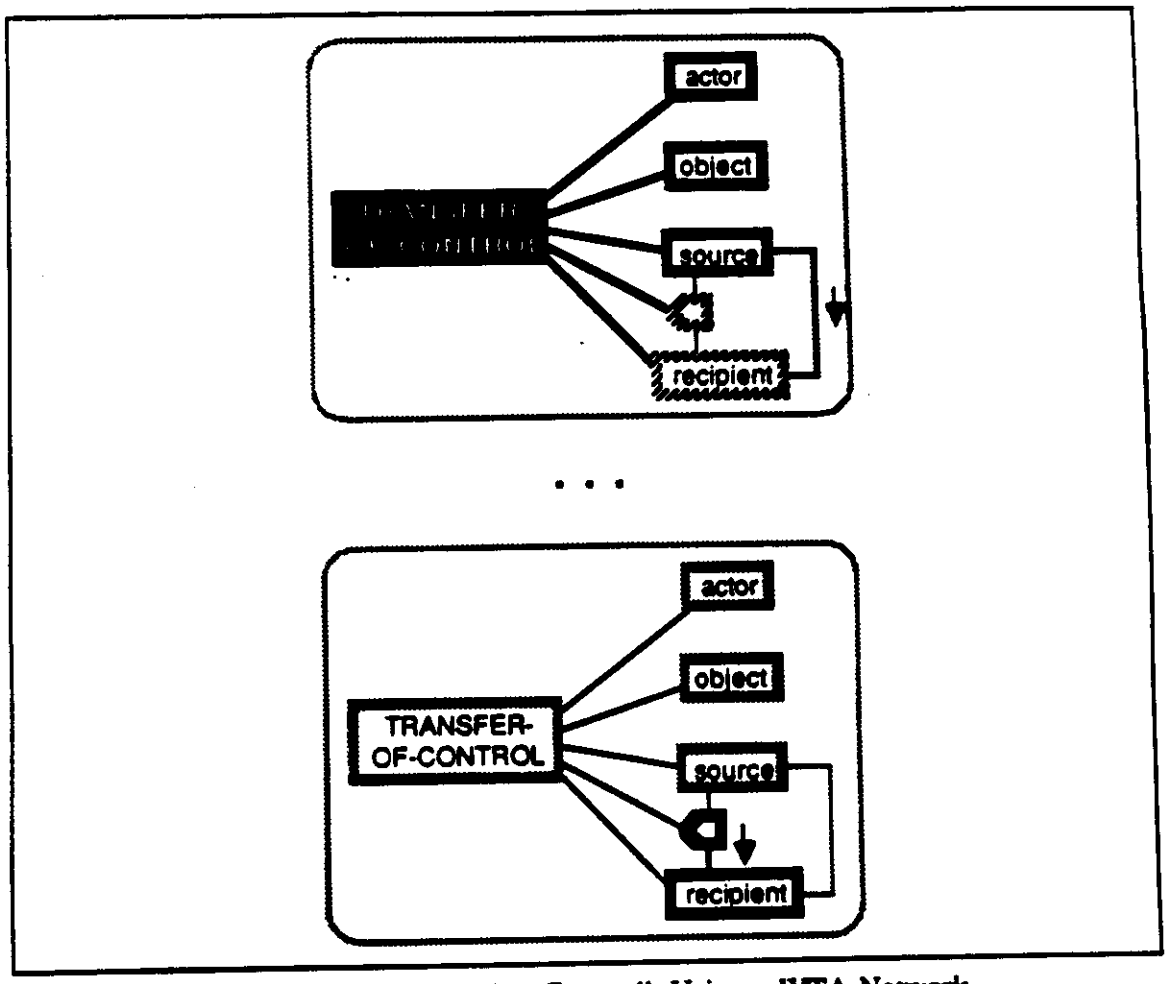

Figure 5.22: Preventing Crosstalk Using a WTA Network

The SOURCE and RECIPIENT roles form a WTA network so that only one can fire at a time. This type of WTA network differs from others, however, in that each of the members must fire. When the first role fires, it does not inhibit the WTA hub, as in most WTA networks. Instead the hub fires when it times out, sending activation to the network member(s), causing one to fire. At this point the first role will not fire again because it is in its refractory period. This process continues until all of the roles have fired. The WTA hub is inhibited only when all of the member roles have fired and sent inhibitory activation to it. Note that this approach would fail for large numbers of roles because the refractory

periods would not last long enough to prevent roles from refiring. In most of the cases encountered, however, the networks had only two members.

5.4. Summary

This chapter has examined the process of temporary role binding in generation and how it can be implemented in a processing model with no variables and no dynamic creation of links. Role binding takes part in two sorts of processes. On the one hand, when a GI or GU is selected, it normally contains information in the form of merged nodes which associate pairs of roles. These associations provide the input to further schema selections. On the other hand, schemas may required particular constraints to be satisfied before they can be selected. The matching of these constraints often requires accessing bindings which are already available.

In the CLM model role binding is implemented in terms of the firing of role and value nodes in relatively close temporal proximity rather than through the formation of explicit links. The binding remains temporarily accessible in the form of a path of primed nodes connecting the role and value nodes. The crosstalk problem, which arises when there is confusion as to the source of a path of activation, is handled through the use of winner-take-all networks which force the source nodes to fire in sequence.

Chapter 6

Language Generation: Sequencing

6.1. The Problem of Sequencing in Generation

The order in which the constituents of an utterance appear depends on two kinds of factors: language-specific conventions and more or less universal tendencies. Examples of conventions are the placement of relative clauses after nouns in English and the reverse ordering in Japanese. Some conventions are absolute: relative clauses always follow nouns in English. Others are only tendencies and can be overridden. For example, in English direct objects usually follow verbs, but they may also come at the beginnings of clauses, for example, in the sentence *that I like*. Universal tendencies include in particular the appearance relatively early in a clause of material which is primed in some way (Bock, 1982). Such psychological considerations may be the sole factor determining an item's position, as often happens in languages with relatively free word order, such as Finnish. But they also come into play when there is a linguistic sequencing convention which is a tendency rather than an absolute constraint.

Sentences (1.8), repeated below as (6.1), illustrate some of the variation that is possible in the ordering of constituents in English. Ditransitive sentences such as these generally refer to an instance of a TRANSFER-OF-CONTROL from one person to another. The argument referring to the semantic OBJECT may precede or follow the argument referring to the RECIPIENT of the transfer. Other things being equal, if one of these arguments refers to something which has been mentioned recently, it will tend to come first. This tendency explains the strangeness of sentences (6.2b) and (6.3b).

(6.1a) *Mary sent John a letter.*

(6.1b) *Mary sent a letter to John.*

(6.2a) *Mary didn't phone John. She sent him a letter.*

(6.2b) *?Mary didn't phone John. She sent a letter to him.*

(6.3a) *Mary didn't throw the letter away. She sent it to John.*

(6.3b) *?Mary didn't throw the letter away. She sent John it.*

One way to view this variation is in terms of competition between the two arguments to fill the position following the verb. One argument may have a head start if it has been primed in some way, in particular if its referent has just been mentioned. In the example sentences, the givenness of the referent results both in the priming that leads that NP to come first and in the realization of the NP as a pronoun rather than a full noun phrase.

This explanation, in terms of competition for output positions, can account for other types of constituent order variation as well. An example is the alternative orders possible with transitive verb-plus-particle combinations in English: *take out the trash, take the trash out.*

Thus sequencing is a phenomenon involving competition and quantitative tendencies rather than absolute constraints. These properties make it reasonable to deal with sequencing within the framework of connectionist models. At the same time, it is not a straightforward matter to implement sequential behavior within the confines of a system consisting of simple processing units that are activated in parallel. Alongside the basic problem of creating emergent sequential behavior from a fundamentally parallel process,

there is the need for sequencing information of two types to be transmitted. When it is time for a constituent to be produced, it needs to signal its own daughter constituents to be produced in the appropriate sequence and, when these are completed, to signal sister constituents, which follow it, to be produced.

6.2. Sequencing in the CLM Model

The thrust of the CLM approach to sequencing is that it can be modelled like the rest of language processing, that is, as a series of selections made on the basis of interacting quantitative factors. Consider first how the parallel activation spread is turned into a sequential process during generation. Activation spreads initially from nodes representing the semantics and pragmatics of the utterance to nodes representing the lexical and grammatical patterns to be used, but the constituent nodes of these patterns often require activation along connections specifying sequencing relations for their firing thresholds to be reached. When more than one constituent may follow a given constituent, there are sequencing connections to all of the alternatives. The weights on these connections represent degrees of syntactic expectation regarding which constituent will follow, and the constituent nodes inhibit each other through a WTA network which permits only one at a time to fire. It is the combination of the activation from syntactic and other sources which determines which constituent wins out over the others and fires. The firing of the winning constituent represents the selection of an item to fill the next output position.

A second problem involves the need to know when a constituent begins and when it ends. This problem is handled by having two nodes for each constituent or phrase, one representing the start and the other the end of the unit. The start node signals daughter constituents to be produced, and the end node signals remaining sister constituents to be produced. I will refer to start and end nodes using the name of the phrase or constituent followed by a slash and the word "start" or "end", e.g., NP/start.

Figure 6.1 shows some of the basic sequencing connections that are part of a phrasal GU, in this case the one for the phrase *over the hill*. Start-end node pairs are denoted by pairs of small squares surrounded by rectangles with rounded corners. The upper square represents the start, the lower square the end of the word or phrase. Single directional connections are indicated by arrows. For example, there is a connection from *OVER-THE-HILL/start to CONSTITUENT1/start but no corresponding connection in the opposite direction. As elsewhere, lines without arrow heads indicate connections in both directions.

The start node for the whole phrase activates the start node for the initial constituent, and the end node for each constituent activates the start node for the next constituent or the phrase end node. Thus in the example CONSTITUENT1/end has a connection to CONSTITUENT2/start, and CONSTITUENT3/end has a connection to *OVER-THE-HILL/end because it is the final constituent in the phrase. Each constituent start node activates a node for the word or phrase filling that constituent, and this node in turn activates the end node for the constituent. For example CONSTITUENT3/start has a connection to the word node "HILL", and this node has a connection to CONSTITUENT3/end. A phrase such as *over the hill* is the simplest type of example because there is no variation in the word order of the pattern.

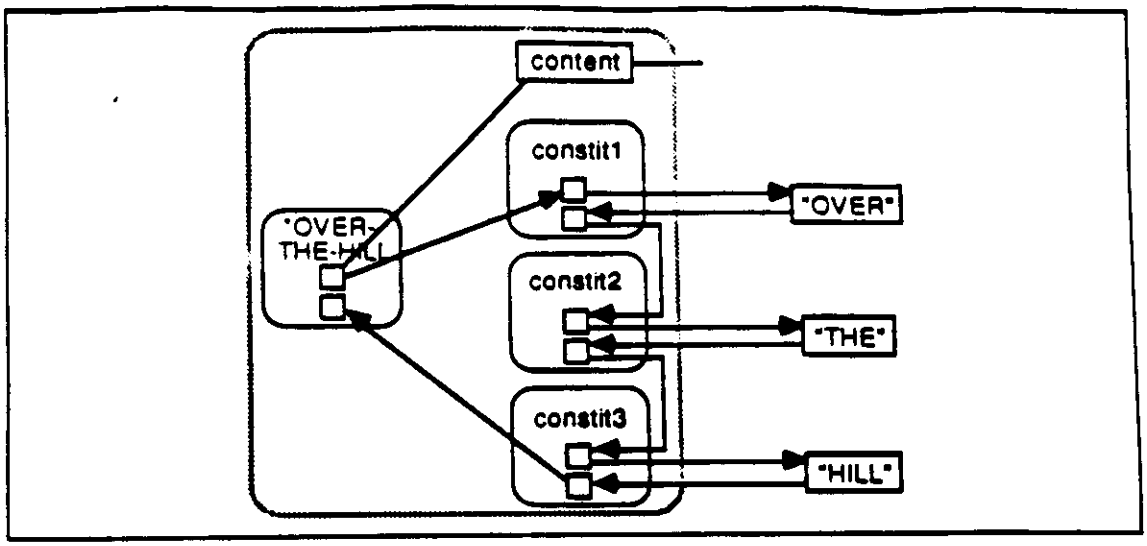

Figure 6.1: Sequencing Connections in GU for a Fixed Pattern

At the other extreme are cases where all possible orderings of constituents are possible, for example, among clause constituents in a language such as Finnish.[13] Figure 6.2 shows the connections needed to handle sequencing in such cases.

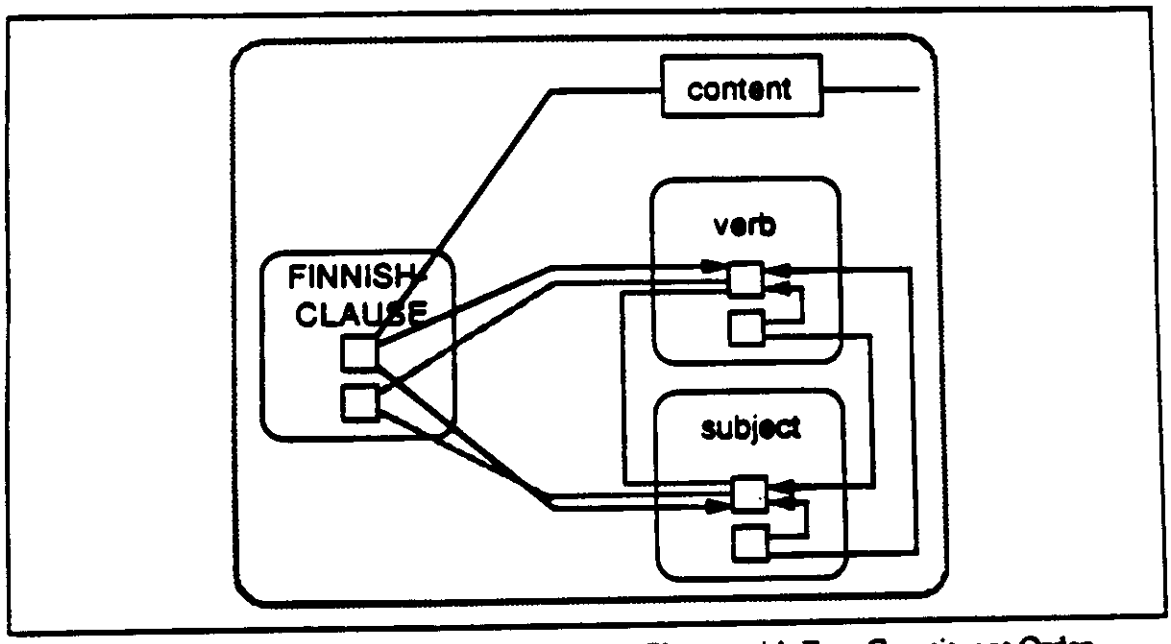

Figure 6.2: Sequencing Connections for a Phrase with Free Constituent Order

[13] Actually there are more extreme cases, the so-called "non-configurational" languages such as the Australian language Warlbiri (Hale, 1981). Not only may clause constituents appear in any order in these languages, the elements of each constituent may be distributed throughout the clause. Morphological marking indicates which constituent a word belongs to. In these languages there is no real sense in which constituents begin and end at all, so start and end nodes could be dispensed with in a Warlbiri version of the CLM model.

The start node for the entire phrase activates all constituents which can appear in initial position, for this example, the VERB and SUBJECT. Because only one constituent may appear in the clause-initial position, the constituent start nodes form a WTA network, indicated in the figure by the inhibitory connections joining them. The start node with the highest activation fires and inhibits the other start nodes. Even a language such as Finnish has preferential orderings, so the weights on the phrase-start-to-constituent-start connections will not be equal. However, activation on constituent start nodes may also come from non-syntactic sources, so we find variability in the final ordering.

When a constituent is complete, its end node sends activation to the start nodes for all constituents which may follow it. These connections are shown on the right side of the figure. For example, VERB/end has a connection to SUBJECT/start. Note that these are unidirectional connections rather than pairs of bi-directional connections. Constituent end nodes also activate the end node for the whole phrase. These connections represent the fact that any constituent can be the final one. The end node for the entire phrase also participates in the WTA network linking the constituent start nodes. That is, following the end of a constituent of the phrase, the beginnings of other constituents compete with the end of the whole phrase.

The network must be arranged in such a way that every obligatory constituent is produced and that no constituent is produced twice (unless it is an iterating constituent). To ensure that every constituent is produced, we must make sure that the end node for the whole phrase does not fire prematurely. Thus the weights on the connections into this node must be small enough so that its threshold is not reached before it has received input from the end nodes of all obligatory constituents.

To ensure that no constituent is repeated, the refractory periods for the start nodes will do for relatively short intervals. But a constituent such as SUBJECT may be very long. That is, if the VERB in the GU in Figure 6.2 appears first, by the time the SUBJECT/end node fires, it is likely that the refractory period for VERB/start will have ended.[14] Two features of the model work to prevent the repetition of constituents in such cases. First, the residual activation, that is, the activation that remains on a node after its refractory period ends, is set very low for constituent start nodes. Second, the end node for each constituent inhibits the start node for that constituent. These inhibitory connections are shown in Figure 6.2. Both of these features have the effect of producing relatively low level of activation on constituent start nodes which have already fired in the current phrase.

A final constraint concerns the need to have general syntactic networks, such as the one in Figure 6.2, reusable following a short interval. For example, the NP schema may be required five or more times for a single sentence. Since there is no mechanism for initializing the activation on sets of network nodes, the nodes making up schemas such as NP and CLAUSE must have their parameters set in such a way that their activation returns quickly to values close to zero. Two parameters are involved, the refractory periods and decay rates of the nodes. The refractory periods must be relatively short to prevent the nodes from being inhibited when they are needed, and the decay rates must be relatively fast to prevent a constituent from firing out of turn simply because it recently appeared in another phrase of the same type.

Thus the behavior of networks such as the one shown in Figure 6.2 is sensitive not only to the connections and their weights, but also to the decay rate, refractory period, and resumptive activation of the nodes. A good deal of careful tuning is necessary before these syntactic networks yield appropriate behavior. The need for meticulous tuning is a problem for a system in which weights and other node and connection parameters are set solely by

[14]The default refractory period in CHIE is 6 time steps.

the programmer. But in a learning system, correct values for the parameters would be discovered through experience and reinforcement (see Section 11.1.2).

6.3. Competition for an Output Position

Consider now the generation of the second sentence in (6.2a): *She sent him a letter.* As always, generation begins with the firing of a set of network nodes representing a goal of the speaker. In this case the goal is that the hearer believe that a letter was sent. This type of goal results in the selection of the ASSERTIVE GI, which has as its PLAN the generation of a DECLARATIVE sentence referring to the sending.

The lexical GU that is selected for this clause, *SEND-MAIL, includes information regarding the ordering of the OBJECT-REFERENCE and RECIPIENT-REFERENCE constituents. A portion of this schema is shown in Figure 6.3.

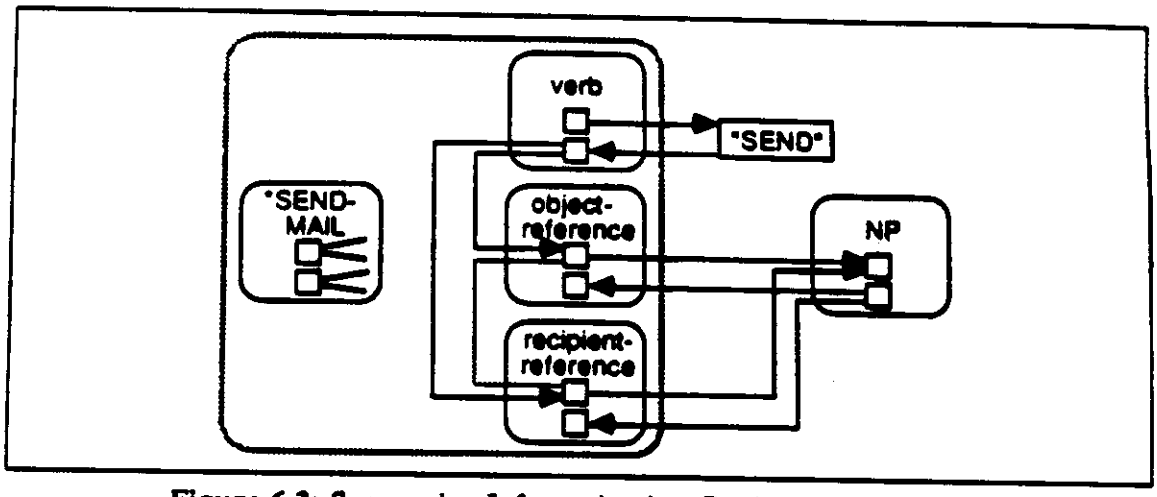

Figure 6.3: Sequencing Information in a Portion of *SEND-MAIL

Both the OBJECT-REFERENCE and RECIPIENT-REFERENCE can appear immediately after the VERB, so there are connections from VERB/end to the start nodes for the two constituents, but only one can appear in this position, so the two start nodes compete through a WTA network. Note also that the start nodes for these constituents activate the NP/start node, and the NP/end node activates the end nodes for these constituents. These connections provide a way for each constituent to "call" the NP GU when it is ready to be produced and for the NP GU to signal the constituent that the phrase is complete.

The event to be referred to in the clause is represented as an instance of the general TRANSFER-OF-CONTROL predicate with MARY as the ACTOR, an instance of the concept LETTER as the OBJECT, JOHN as the RECIPIENT, and MAIL as the MEANS of the transfer. We ignore time and tense in order to simplify the discussion. Figure 6.4 shows this portion of the network.

The sentence which preceded the one we are generating was *Mary didn't phone John.* The utterance of that sentence resulted in processing involving the concepts MARY and JOHN, so there is residual activation on these nodes and the nodes immediately connected to them. Other nodes in the figure would also be activated, but in this figure and those following, activation (and inhibition) will only be indicated for nodes which are relevant to the discussion.

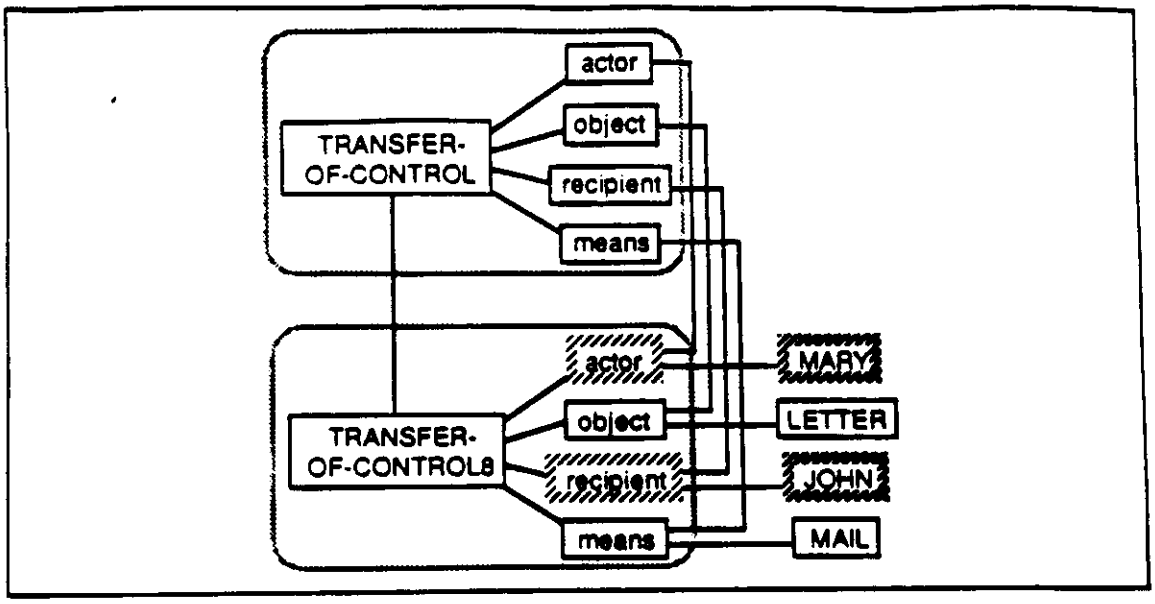

Figure 6.4: Portion of Input to Generation of *she sent him a letter*

Activation spreading from TRANSFER-OF-CONTROL8 (i.e., the specific transfer instance) converges on a set of verb lexical GUs that may be used to describe the input notion. Competition among the CONTENT roles of these schemas eventually forces one to win out. For this example, we assume that the *SEND-MAIL GU would predominate because it specifies that the MEANS of the transfer is by MAIL, as in the input. A simplified view of this lexical GU selection process is shown in Figure 6.5.

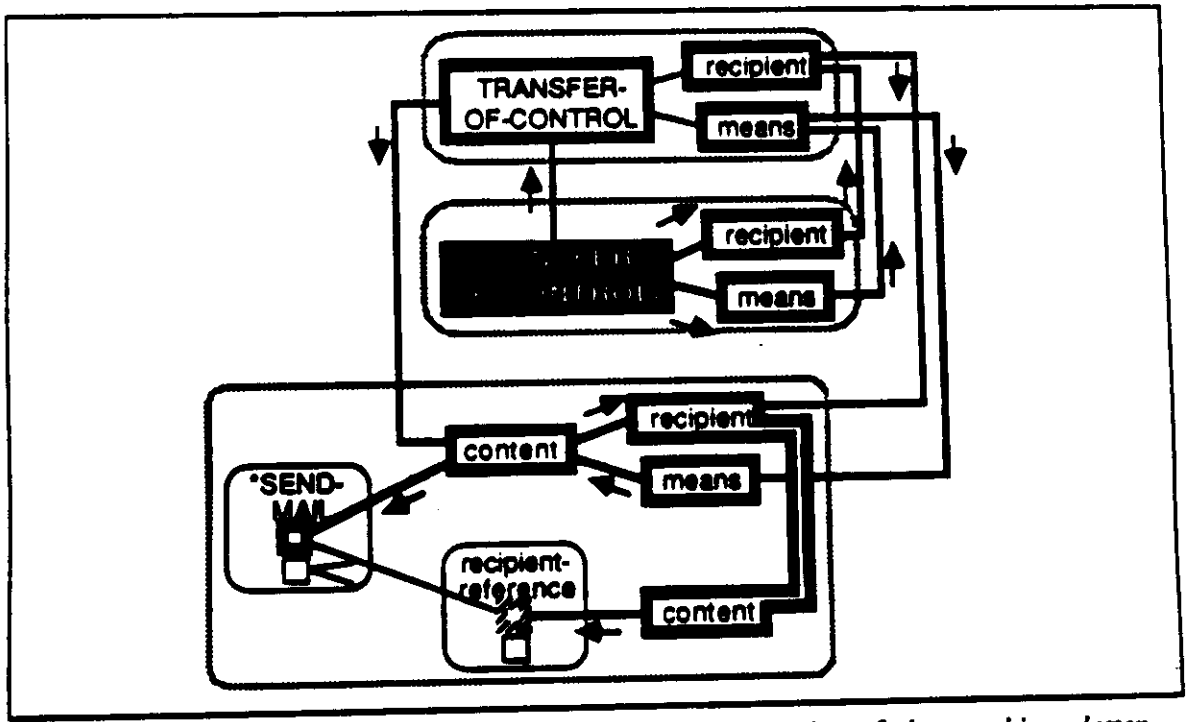

Figure 6.5: Selection of *SEND-MAIL Schema for Generation of *she sent him a letter*

Figure 6.5 also shows another process taking place at the same time, which results in the priming of the RECIPIENT-REFERENCE constituent. Activation spreading from TRANSFER-OF-CONTROL8 causes the primed RECIPIENT node to fire, leading to the firing of the merged RECIPIENT/CONTENT node in *SEND-MAIL which represents the role association information for the RECIPIENT-REFERENCE constituent. Activation then spreads from this node to RECIPIENT-REFERENCE/start.

Thus the fact that JOHN was referred to in the previous sentence led to the priming of nodes representing JOHN's roles in other facts, and when the speaker chose to assert one of these facts (the sending of the letter), the constituent that is to refer to JOHN was primed. In this way the givenness of JOHN has provided an advantage to the constituent which will refer to JOHN. Note that the OBJECT-REFERENCE constituent will not be primed because there was no initial activation on the OBJECT node in PHYSICAL-TRANSFER8.

Once the *SEND-MAIL GU has been selected, activation spreads through it, resulting in the priming of the nodes representing the constituents of the clause. At the same time activation has also spread to the constituent nodes of the higher-level CLAUSE GU. The connections within this schema determine the order of the SUBJECT and VERB in the sentence. The fact that the event referred to occurred before the time of speaking also leads to the selection of the PAST-CLAUSE GU, and this in combination with the *SEND-MAIL schema results in the firing of the node representing the word *sent*. For the purposes of this discussion, we ignore the details of these processes.

When the verb has been produced, the VERB/end node in the *SEND-MAIL schema fires. From here activation spreads to the nodes representing the beginnings of the two possible following constituents: RECIPIENT-REFERENCE/start and OBJECT-REFERENCE/start. These nodes compete with one another via a WTA network. In this case the priming on RECIPIENT-REFERENCE/start leads this constituent to win out over OBJECT-REFERENCE/start. The situation at this point is shown in Figure 6.6. The portion of the sentence already generated is given below the network in this and the next two figures.

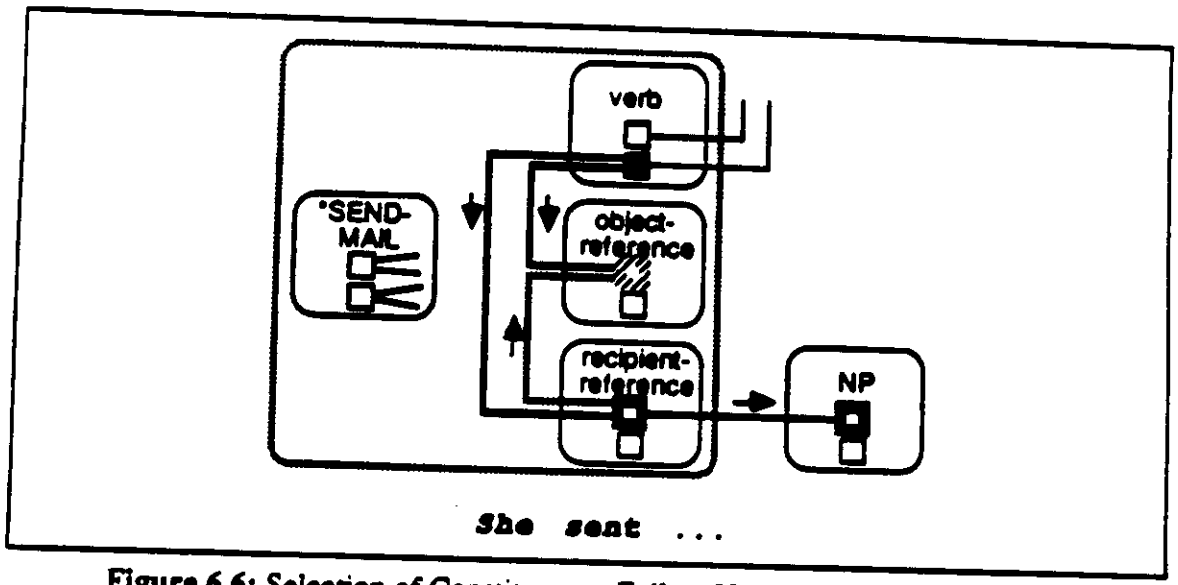

Figure 6.6: Selection of Constituent to Follow Verb in *she sent him a letter*

Next the NP schema takes over. At this point there is competition between the GU for third person pronouns and that for full NPs. Since John has just been referred to in the previous sentence, the pronoun GU wins out.

When the NP is complete, activation is sent back to RECIPIENT-REFERENCE/end, which fires, leading to the beginning of the OBJECT-REFERENCE constituent. Figure 6.7 shows what then takes place.

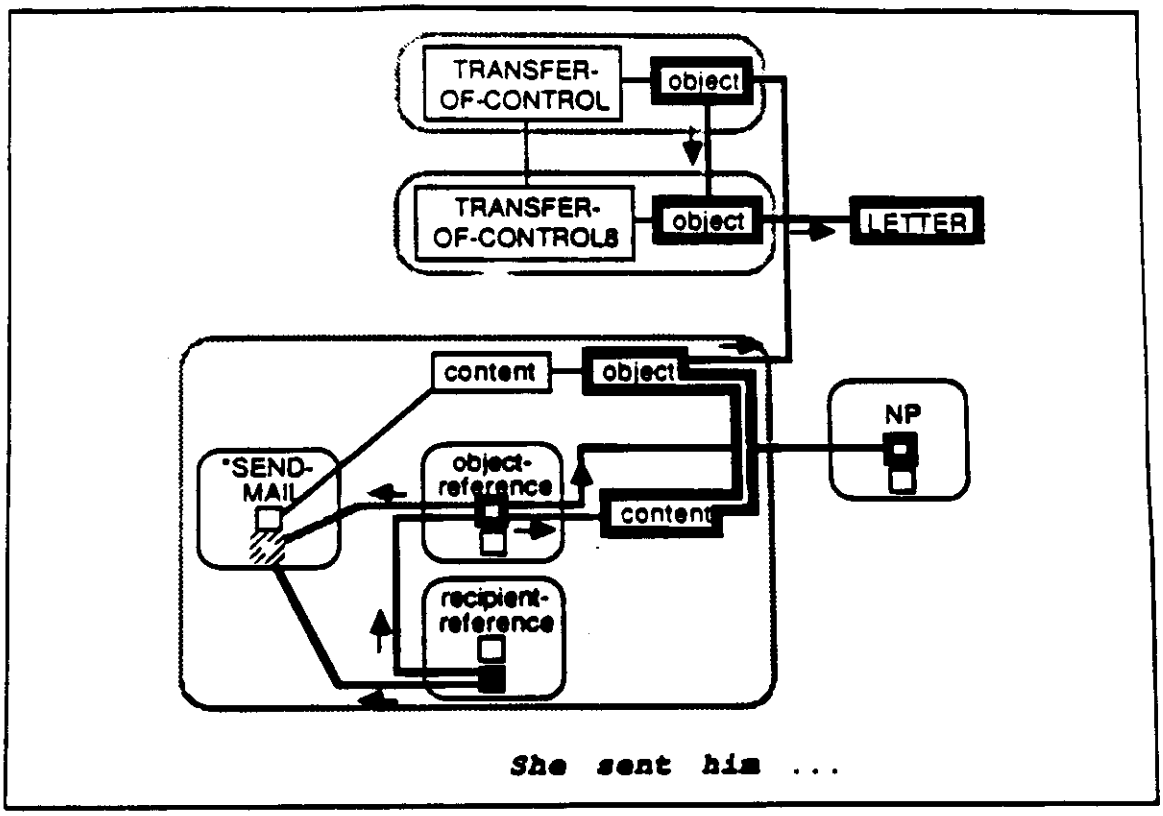

Figure 6.7: Start of Generation of Reference to OBJECT in *she sent him a letter*

In general, there are two possibilities for what follows the RECIPIENT-REFERENCE constituent in an instance of the *SEND-MAIL schema. In a sentence such as *she sent a letter to John*, the RECIPIENT-REFERENCE is the last constituent, and the clause is complete. In a sentence such as the one we are generating, on the other hand, the RECIPIENT-REFERENCE is followed by the OBJECT-REFERENCE. RECIPIENT-REFERENCE/end activates the nodes representing these two possibilities, the end node for the whole clause and the OBJECT-REFERENCE/start node. Note that it is not possible to encode this information in the form of a rule such as "if the OBJECT-REFERENCE has already been produced, then go to *SEND-MAIL:END" because the system has no explicit memory for what has or has not already been generated. All of its decisions are made completely locally.

The weights on the two output connections from RECIPIENT-REFERENCE/end are such that OBJECT-REFERENCE/start is the default and will be preferred in this case. Thus OBJECT-REFERENCE/start fires. This initiates a role co-activation process, resulting in the firing of the LETTER node.

Again control is passed to the NP schema. Here two further selections take place. As there is no evidence that the hearer knows the referent, the INDEFINITE-NP schema is

selected over the DEFINITE-NP schema by default.[15] INDEFINITE-NP provides the article *a*. Finally, the lexical GU *LETTER is selected as a result of activation spreading from the LETTER node. This schema provides the noun *letter* for the OBJECT NP.

Once the final constituent is complete, activation is sent back to the OBJECT-REFERENCE/end node, and the clause is complete. Since the system has no memory for the fact that the RECIPIENT constituent has already been produced, however, there must be a way to prevent it from following at this point. Figure 6.8 shows how this is done.

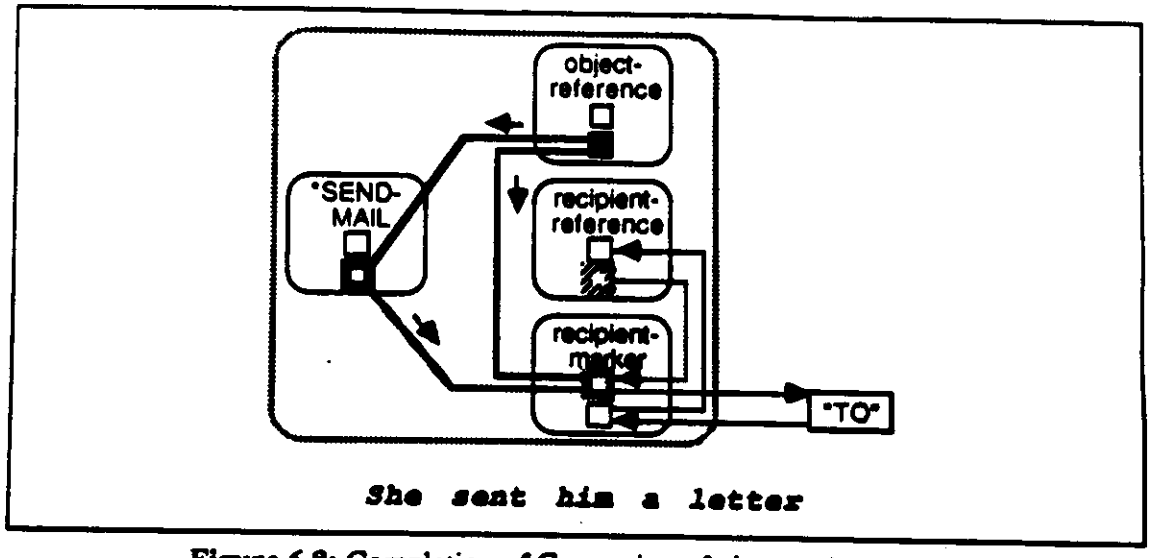

Figure 6.8: Completion of Generation of *she sent him a letter*

In general, there are two possibilities for what may follow the OBJECT-REFERENCE constituent, the end of the clause or the *to* case marker and the following RECIPIENT-REFERENCE constituent. The case marker is represented as a separate constituent of the clause, RECIPIENT-MARKER. As before, the nodes representing the two alternatives, here *SEND-MAIL/end and RECIPIENT-MARKER/start, form a WTA network. We need a way to prevent the RECIPIENT-MARKER from being generated in the current sentence; otherwise we might get *she sent him a letter to him.* The appearance of the RECIPIENT-REFERENCE must prevent the later occurrence of the RECIPIENT-MARKER and a reappearance of the RECIPIENT-REFERENCE. To accomplish this, there is an inhibitory connection from RECIPIENT-REFERENCE/end to RECIPIENT-MARKER/start. When RECIPIENT-REFERENCE/end fires, it inhibits RECIPIENT-MARKER/start, and this inhibition allows *SEND-MAIL/end to win out later when OBJECT-REFERENCE/end fires in the current example. The fuzzy border on RECIPIENT-MARKER/start in the figure indicates that the node is inhibited.

6.4. Using Sequencing Information in Syntactic Schemas

In the previous example we have made use of sequencing information found in the lexical GU *SEND-MAIL. This sort of information also appears in more general lexical GUs such as *SEND and in non-lexical schemas such as DITRANSITIVE-CLAUSE, the

[15] This process is not yet completely implemented in the model because there is no adequate means of dealing with inferences about what the hearer knows.

general GU for clauses with both OBJECT-REFERENCE and RECIPIENT-REFERENCE constituents. If a specific schema lacks the required information, a more general one is used automatically.

Most noun GUs have no sequencing information, so generation operates somewhat differently for NPs than for clauses. The generation of an NP involves selecting one or more GUs on the basis of the content of the NP. For example, in generating *the box*, the speaker selects the *BOX schema because what is referred to is an instance of the concept BOX and the THE-NP schema because the box is known to the hearer. Each of these GUs provides a part of the morphology of the NP, but general information about how the constituents of the NP are sequenced is not repeated in each GU. Rather it appears in one place at the level of the general NP schema. The NP schema is activated whenever an NP is generated (normally through its input connections from constituent nodes such as SUBJECT and DIRECT-OBJECT), and its function is to guide the selected specific GUs in sequencing the words that they provide.

Figure 6.9 illustrates the connections that enable the NP GU to control the output of words in other GUs. For simplicity, however, it does not show the sequencing information in the schema. Two lower-level GUs appear here, (1) THE-NP, the schema for NPs with the determiner *the*, and (2) *BOX, the lexical schema for the noun *box*. The THE-NP schema is matched when the CONTENT is something that is explicitly associated with the KNOWLEDGE-BASE of the HEARER. This characterization is an oversimplification of the semantics of *the*, but this will not affect our discussion of sequencing.

The connections from start nodes in the NP schema to start nodes in the lower-level schemas carry activation which signals the words when it is time for them to be uttered. The connections from end nodes in the lower-level schemas to end nodes in the NP schema carry activation which signals that a word has been uttered and that a new constituent can begin. In the next section there is an example which illustrates these processes.

6.5. Handling Optional and Iterating Constituents

The mechanisms which handle competition for a particular output position are also used in the generation of optional constituents, such as adjective phrase modifiers in NPs. In such cases, there is competition between the optional constituent and one or more other constituents which either follow it or replace it.

Here we will consider the generation of the NP *the empty box*. Figure 6.10 shows more of the NP schema, including information about the semantics of adjective phrase modifiers[16]. The attribute that the adjective refers to has as its ATTRIBUTE-OBJECT the referent of the NP. This information is embodied in the connection from the NP:CONTENT node to the ADJECTIVE-MODIFIER:CONTENT:ATTRIBUTE-OBJECT node. It is through this latter node that activation reflecting the need to generate an adjective enters the NP schema.

Figure 6.10 also shows some of the directed connections needed for sequencing in NPs. There are two alternatives for what follows the determiner directly, an adjective modifier or the head noun of the phrase. These possibilities are represented as connections from DET/end to both ADJECTIVE-MODIFIER/start and NOUN/start. These two nodes form a WTA network so that only one can follow the determiner. If an NP does have an adjective modifier, it may be followed either by another adjective phrase or by the head noun. Again

[16]The model currently handles only single-word adjective phrases like *empty*; that is, there is no role for an adjective phrase modifier such as *completely*.

the possibilities are handled by explicit sequence connections, in this case from ADJECTIVE-MODIFIER/end to ADJECTIVE-MODIFIER/start and NOUN/start. Note that the same WTA network can handle the decision regarding what follows the determiner and what follows an adjective.

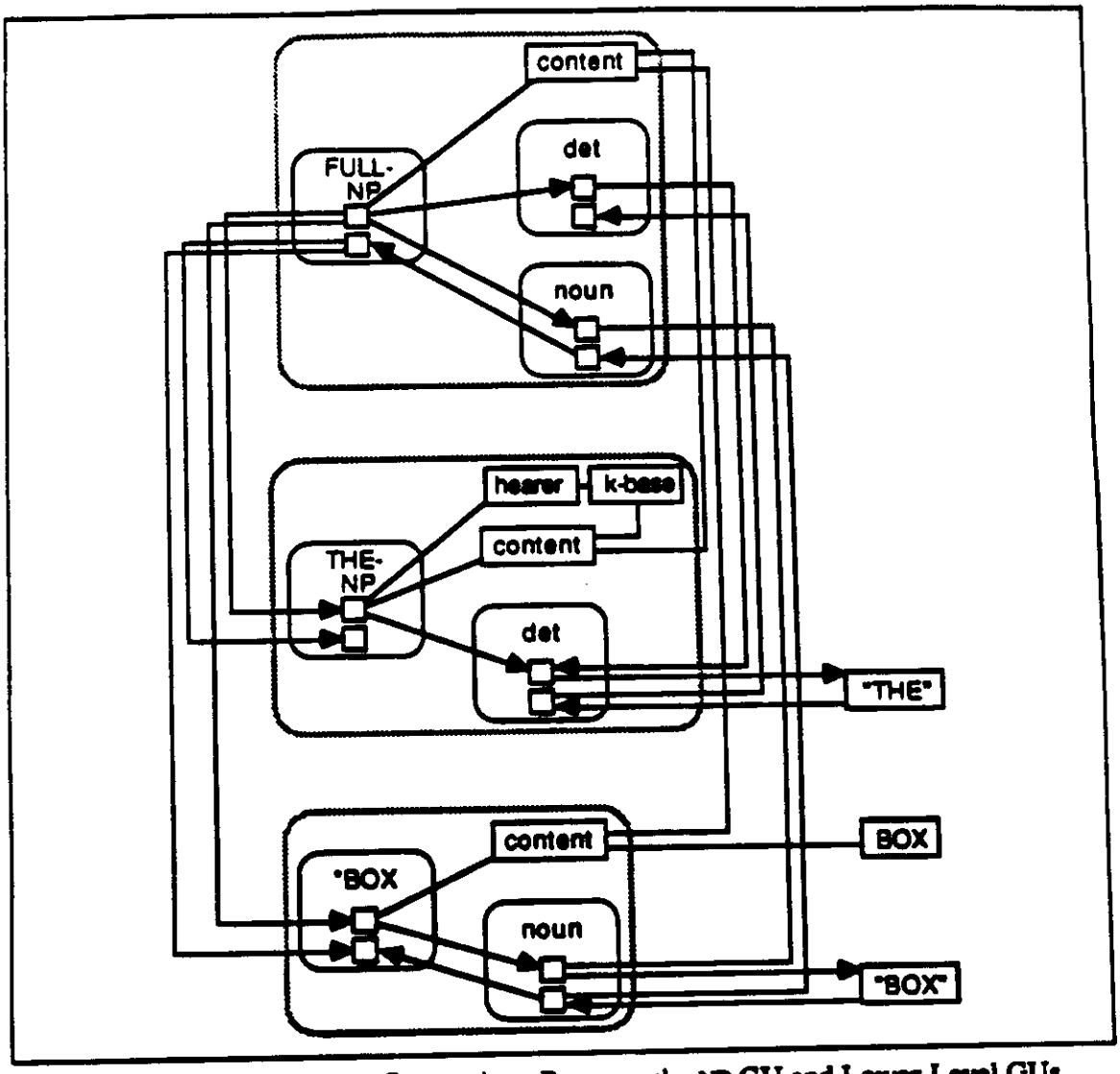

Figure 6.9: Sequencing Connections Between the NP GU and Lower-Level GUs

During the selection phase for an NP lexical GU, features of the referent receive activation. One of these may be an ATTRIBUTE-OBJECT role of an ATTRIBUTE fact. That is, the referent may be an argument of a fact of type EMPTY, GREEN, or the like. In such cases the ADJECTIVE-MODIFIER constituent in the NP GU is primed. Figure 6.11 shows how this process occurs for the phrase *the empty box*.

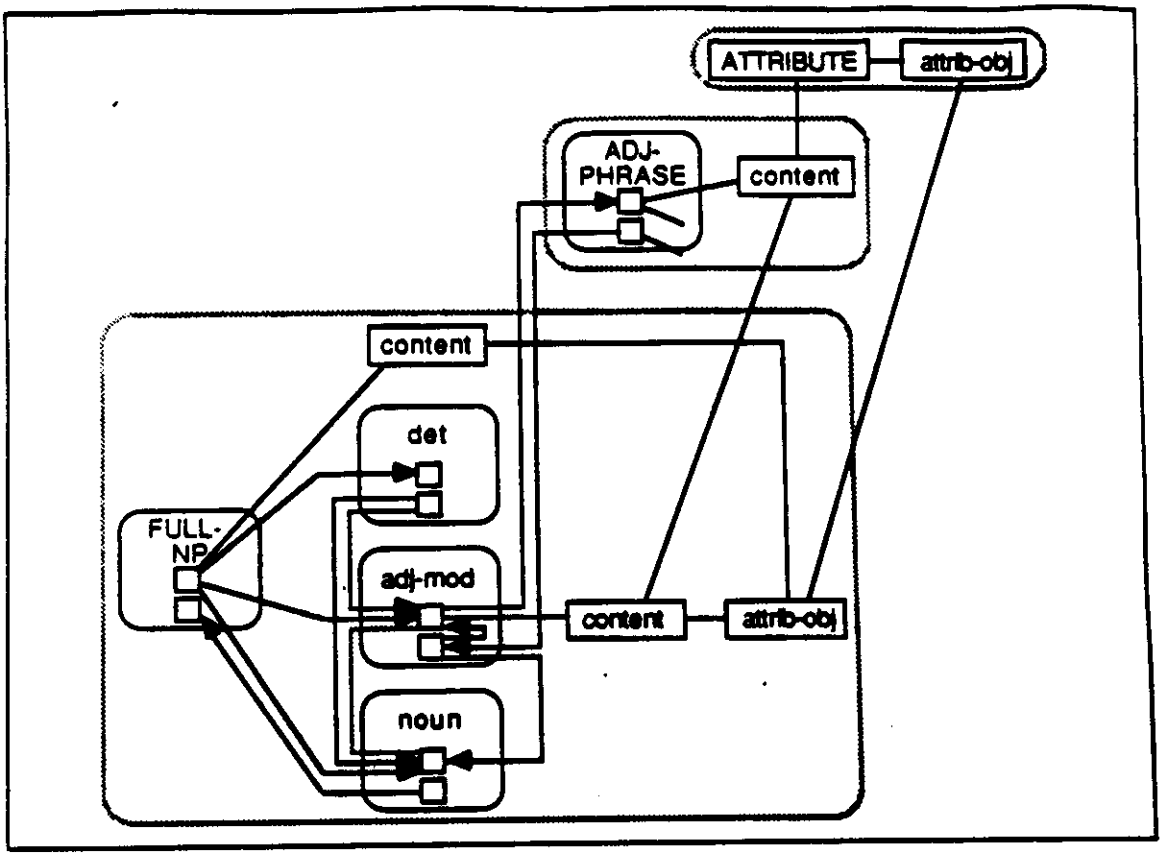

Figure 6.10: Adjective Phrase Modifiers in the NP GU

BOX4, the referent of the NP being generated, is represented as empty, that is, the BOX4 node is connected to the EMPTY:ATTRIBUTE-OBJECT node. When generation begins, activation spreads from BOX4 and from NP:CONTENT, as for all NPs. From these two sources, activation converges on the ADJECTIVE-MODIFIER:CONTENT:ATTRIBUTE-OBJECT role in the NP GU. The firing of this node leads to the firing of ADJECTIVE-MODIFIER:CONTENT and the priming of ADJECTIVE-MODIFIER/start.

At the same time, activation spreading through the NP schema causes the determiner for the phrase to be produced. At this point the THE-NP GU has already been selected on the basis of the fact that BOX4 is known to the hearer. In the current implementation this knowledge is represented as an explicit connection from BOX4 to the KNOWLEDGE-BASE role associated with JOHN, the hearer for this NP. The DET/start role in the THE-NP schema is primed, and when it receives activation from DET/start in NP, it fires. This in turn initiates the generation of the adjective. Figure 6.12 shows the sequencing processes.

Activation spreads from NP/start to the nodes representing the start nodes of the constituents of the phrase. DET/start fires because the connection to it has the highest weight. This sends activation to the DET/start roles of all subtypes of NP. In this case, we assume that the THE-NP GU has been activated on the basis of the fact that the referent is known to the hearer, so its DET/start node has been primed. This node now fires, activating the word node "THE". The firing of this node represents the utterance of the word. The word node sends activation back to DET/end in THE-NP, and this node in turn activates DET/end in the NP schema, signalling the end of this constituent. DET/end activates two nodes, ADJECTIVE-MODIFIER/start and NOUN/start, which compete for the

next position. In this case the priming on the former node leads it to win out. ADJECTIVE-MODIFIER/start then activates ADJECTIVE-PHRASE/start, and the ADJECTIVE-PHRASE GU takes over.

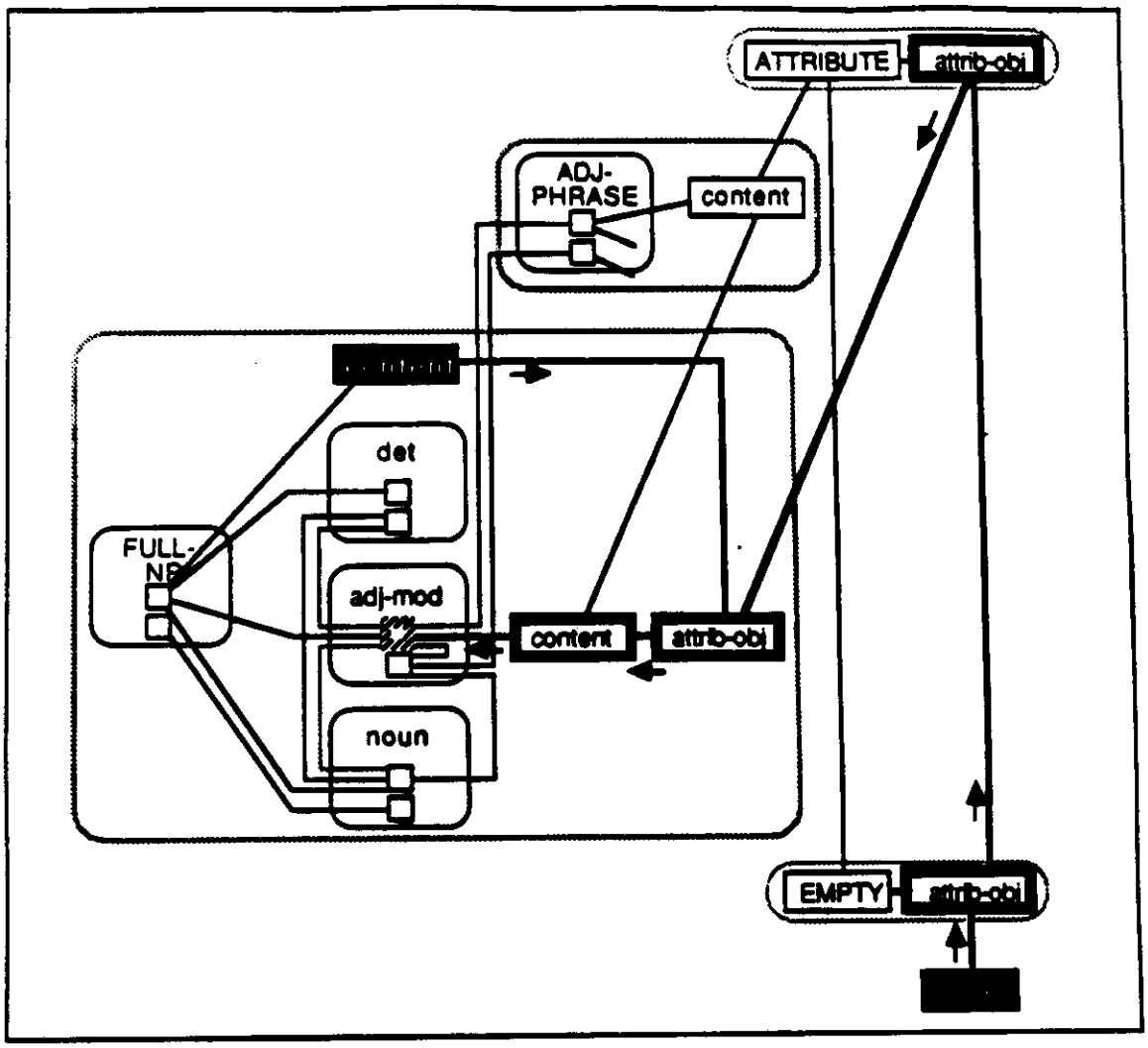

Figure 6.11: Priming of ADJECTIVE-MODIFIER/start in the Generation of an NP

The GU for the adjective *empty* has been primed by this time as a result of activation coming from the referent, and additional activation from the ADJECTIVE-PHRASE GU now results in the production of the word *empty*.

Figure 6.13 shows what happens when the adjective phrase is complete. ADJECTIVE-PHRASE/end fires and send activation back to ADJECTIVE-MODIFIER/end in the NP schema. There are again two possibilities for what may follow, either another adjective or the head noun of the phrase, and the two connections from ADJECTIVE-MODIFIER/end represent the alternatives. Another adjective follows if there is additional priming on ADJECTIVE-MODIFIER/start from a further ATTRIBUTE of the input concept. For example, if the box is red, then RED:ATTRIBUTE-OBJECT would fire, leading to activation of ADJECTIVE-MODIFIER/start in the same way that it happened for EMPTY:ATTRIBUTE-OBJECT. However, there are still unsolved problems in the generation of iterating

constituents such as ADJECTIVE-MODIFIER. The major stumbling block is that some of the same nodes are involved in the paths representing the different attributes. Thus unless the activation for different attributes is separated quite widely, the paths will interfere and only one adjective will be generated. For this reason currently it is difficult for the model to produce adjective sequences, such as in *the big red empty box*.[17]

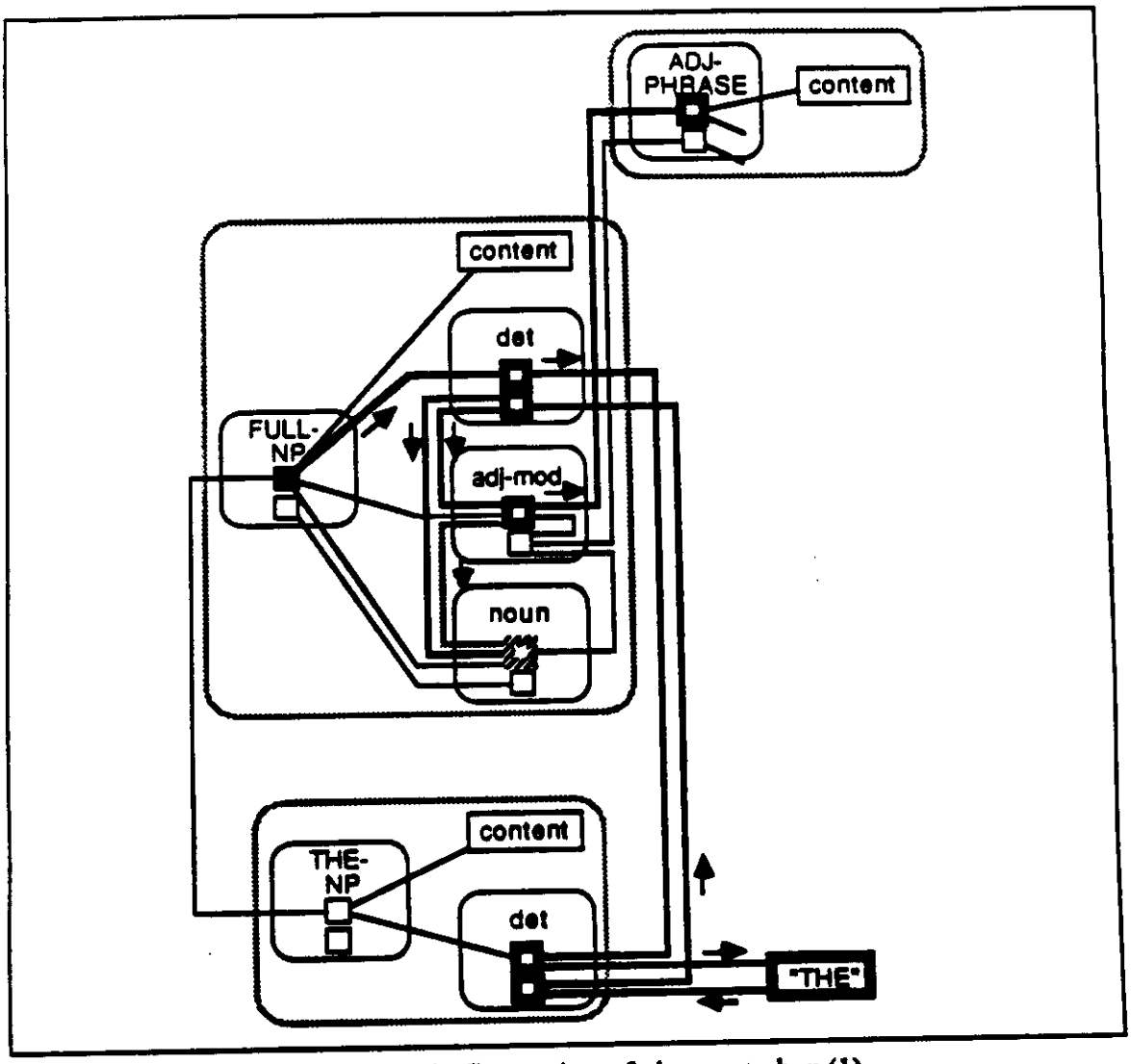

Figure 6.12: Generation of *the empty box* (1)

[17]One possible way to handle sequences of adjectives is to treat each adjective as a separate constituent as is done, for example, in the Nigel generation system (Mann & Matthiessen, 1983). Thus there might be a COLOR-ADJECTIVE-MODIFIER constituent and a SIZE-ADJECTIVE-MODIFIER constituent, among others. That way there would be no harmful overlap among the paths by which the adjective constituent nodes are primed from conceptual features. This possibility is currently being pursued. Note, however, that this solution does not actually make use of iteration.

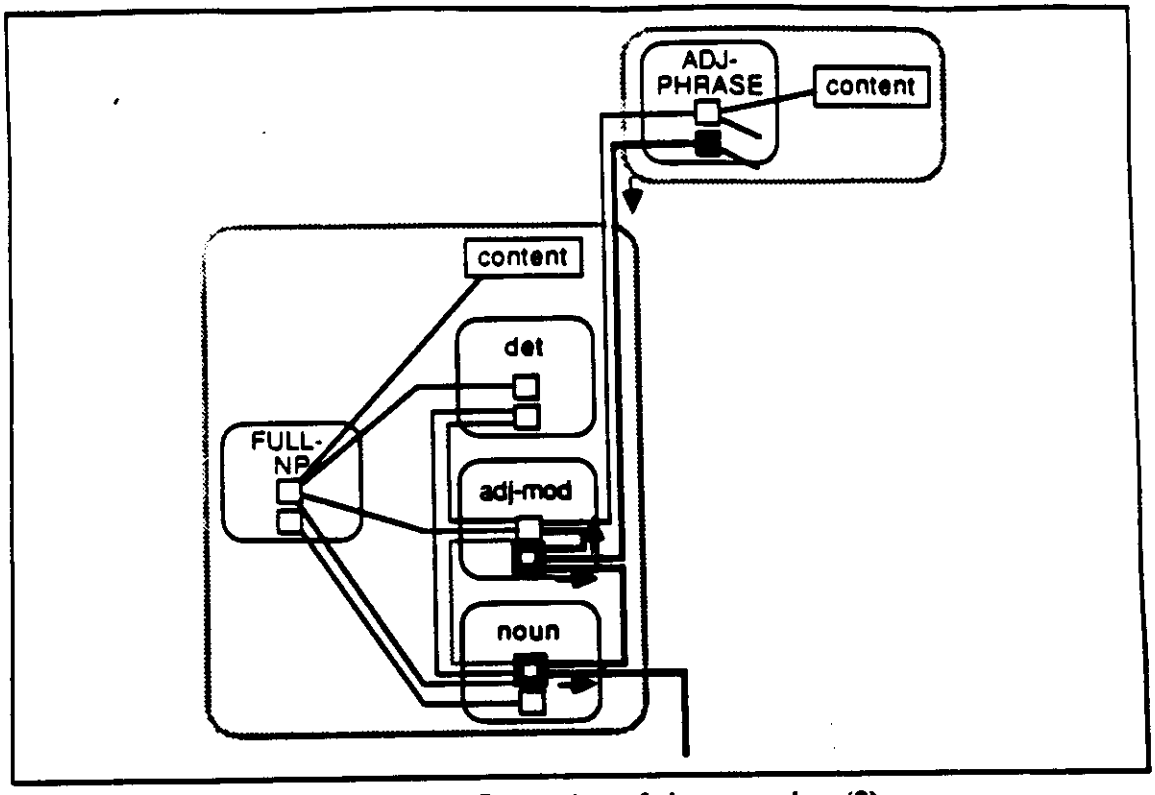

Figure 6.13: Generation of *the empty box* (2)

In this example NOUN/start wins out because ADJECTIVE-MODIFIER/start has received no additional activation on the basis of other salient attributes of the referent. The firing of NOUN/start results eventually in the utterance of the head noun *box* and the end of the phrase.

6.6. Summary

This chapter has looked at the problem of achieving sequential behavior in a model of language generation which is fundamentally parallel. Sequencing involves competition among various quantitative factors, so it can be profitably modelled within a connectionist model, such as the CLM model. Key features of the CLM approach to sequencing are (1) the representation of phrasal units and their constituents as pairs of network nodes, one for the start and one for the end of the sequence; (2) the representation of ordering constraints and tendencies as uni-directional weighted connections; and (3) the use of winner-take-all networks to impose sequentiality on the parallel spreading activation mechanism and to implement competition between constituents for particular output positions.

Chapter 7

Multilingual Generation

7.0. Introduction

A basic assumption of the present approach is that generation in a language other than the speaker's first[18] works more or less like generation in the speaker's first language (L1). It involves the selection of appropriate units of knowledge from an inventory in memory by means of the spread of activation from nodes representing input. If the speaker is not well acquainted with the target language, the inventory will be a relatively skimpy one, and we will see an unusual amount of hesitation and other evidence of difficulty in achieving particular communicative goals, but these are processes that also occur, to a lesser degree, in L1 generation. Of course, the units in the speaker's inventory for the L2 cannot be expected to correspond precisely to those in the memory of a native speaker of the L2. For that matter, no two native speakers of a language will have the same set of units. The point is that the units are used in essentially the same way regardless of whether the language is the speaker's (or hearer's) first, second, or eighth.

What is special about L2 generation is that there are units of linguistic knowledge in memory for the L1 as well. Thus, at a minimum, schemas need to be distinguished for each language. Because there is no need to make a sharp distinction between L1 and L2 generation, we can consider the general case of generation in any of a speaker's languages, given knowledge of more than one. The evidence from errors in L1 generation in bilinguals (Sharwood Smith, 1983) seems to bear out the usefulness of this general categorization. Thus what I will be talking about in this chapter is **multilingual generation**, the memory organization that supports it, and some of the consequences of this organization.

7.1. Distinguishing Schemas for Different Languages

In multilingual generation the speaker has two special sorts of problems, selecting a target language for an utterance, and selecting schemas from linguistic memory which match the target language. In order to look at these processes, we first need to consider the question of how knowledge of different languages is to be distinguished in memory.

In the early stages of language acquisition, children clearly have no notion of "a language". That is, while a child may be able to understand and use appropriately any number of words, phrases, and structures, she does not yet have the sort of meta-knowledge required to talk and reason about a language as an entity in itself. In fact, when this sort of knowledge is acquired, it will probably have little or no effect on how language is used by the learner. In other words, there is no real sense in which a language learner/user has "a language", as opposed to pieces of knowledge about how to do things with words.

When a young child is first exposed to two or more languages, the units that the child acquires are certainly not filed under categories like ENGLISH and JAPANESE, categories which she does not yet have. What the child is concerned with learning is the circumstances under which the words and phrases that she hears are appropriate. She may make generalizations on the basis of any aspects of the context that tend to co-occur with the items she is learning, including features of the physical setting and perhaps even the time of day. In particular, however, she soon discovers that some of the words and

[18] I will generally refer to any language other than the speaker's first as a "second language" or "L2".

phrases are used by particular speakers and not others and that those items get their message across when spoken to those same people. For example, we might imagine a child with an English-speaking mother and a Japanese father learning schemas like those in Figure 7.1.

```
(*DOG is-a NP
     (speaker MOMMY)
     (content DOG)
     (noun "DOG"))

(*INU is-a NP
     (speaker DADDY)
     (content DOG)
     (noun "INU"))
```

Figure 7.1: Possible Early GUs in a Bilingual Learning Context

The child then makes generalizations regarding the sorts of people the different words and phrases are associated with. In the end she arrives at a category of speaker/hearer for each of the languages she is learning. The units now begin to look like the one shown in Figure 7.2.

```
(*DOG is-a NP
     (speaker ENGLISH-SPEAKER)
     (hearer ENGLISH-SPEAKER)
     (content DOG)
     (noun "DOG"))
```

Figure 7.2: GU for *dog* in a Bilingual Context

As the child begins to generalize aspects about the syntax of the languages, a schema such as that shown in Figure 7.2 will not suffice because it associates the lexical schema with a generic notion of NP rather than with the category of English NPs, which of course have their own special properties. Thus, for higher-level categories such as NP, there need to be language-specific schemas. One possible representation of these relationships is shown in Figure 7.3. There is a cross-linguistic NP schema representing the general notion of NPs and subtypes for each language providing language-specific properties such as where the various constituents go.

In second language learning (as opposed to the simultaneous acquisition of two or more languages) the situation is a little different. Here the pieces of knowledge for the L1 cannot indicate that the speaker and hearer are speakers of that language since this information would not have been relevant when the units were learned. It is also unreasonable to assume that this information is somehow added to all L1 schemas when the learner begins to acquire the L2. Therefore, only the L2 schemas will have the speaker/hearer information, and the L1 will behave as the default during processing. That is, unless there is reason to assign the hearer to the category of L2-speaker, L1 schemas will be selected.

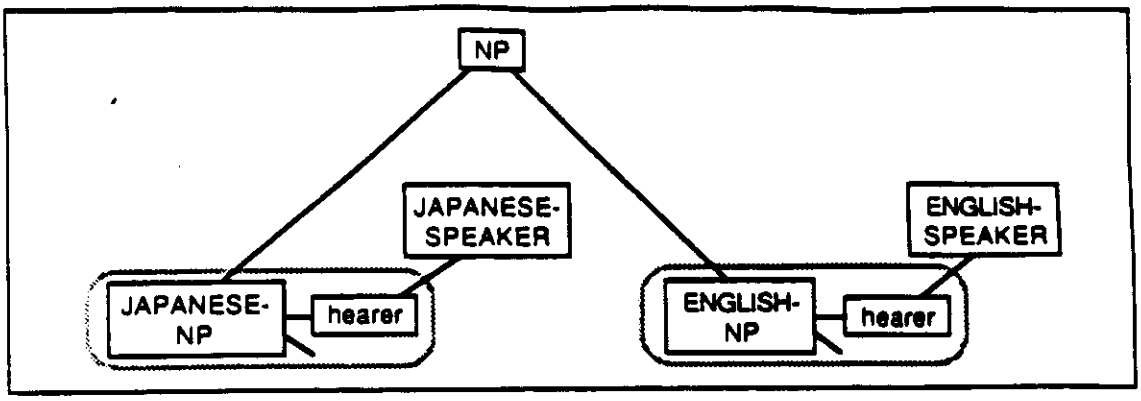

Figure 7.3: GUs for NPs in a Bilingual

Note that the inclusion of category restrictions on the speaker and hearer in schemas extends to expressions within a language as well. Baby talk provides one example; here the hearer or speaker is specified as a child. Other cases involve terms known only to people in particular areas of expertise. Thus a dentist might have separate GUs for *caries* and (*tooth*) *decay*, both pointing to the same concept. The former would include the additional information that the speaker and hearer are dentists. These restrictions would normally prevent the word *caries* from being produced when the speaker knows that the hearer is not a dentist.

7.2. Schema Selection by Multilinguals

In Chapter 4 we saw how the selection of GI and GUs in generation can be viewed as a pattern matching process. The schema or schemas which are selected are the ones which best match the set of input features. The most important of these features are aspects of the concepts to be referred to and of the goal behind the utterance, but they also include features associated with the hearer and the speaking context. In the multilingual case the latter categories become more significant because they distinguish schemas in one language from those in another.

Thus schema selection by multilinguals differs from schema selection in the monolingual case only in the number of input features that are matched. No explicit decision needs to be made about the target language; appropriate schemas are selected automatically as activation converges on those schemas whose HEARER roles match the current hearer.

During generation in a language other than the speaker's first, there is the possibility that a GU will be found which matches the input adequately except for the restriction on the HEARER. In such cases, the speaker may either ignore this schema and continue searching for a GU for the appropriate language or may choose to use it, either by translating it word-for-word into the target language or integrating it directly into the target language utterance. We shall see examples of these alternatives later.

Another possibility is that contextual factors will outweigh characteristics of the hearer, resulting in the selection of an inappropriate schema. A native speaker of Amharic, whom I am acquainted with, is a fluent speaker of English but has a tendency to speak Amharic in informal settings, even when the hearers know no Amharic. In the terms of the CLM model, this speaker would have some degree of formality associated with the

contexts in which English is appropriate. When the context is informal, there is less likelihood of English schemas being matched and her default language sometimes wins out even when there are other features that would call for English, such as the language abilities of the hearers.

When both speaker and hearer are bilingual, **code switching**, that is, the mixing of languages both between and within sentences, often occurs. This mixing is a straightforward consequence of the way in which schemas are distinguished on the basis of the HEARER role. Since the hearer in a bilingual context is a speaker of both languages, all schemas in linguistic memory will match on their HEARER roles, and selection will be based entirely on relative appropriateness or strength.

In a monolingual context, GUs have to inhibit one another through their CONTENT roles in order that only one gets selected for a given input concept. These inhibitory connections represent the knowledge that there is a single best lexical item for referring to a particular concept. When there are GUs for more than one language, on the other hand, there is usually an appropriate lexical item for a given input in each language. Thus the CONTENT roles of the schemas do not inhibit one another across languages. They may in fact support one another in cases of lexical transfer. That is, the CONTENT role of a GU in the L1 may tend to activate the CONTENT role of a corresponding GU in the L2. We will see an example of this phenomenon in Section 7.4.1.

We still must ensure, however, that only one lexical item actually is produced for an input concept. Thus a speaker who knows both Japanese and English may activate both *DOG and *INU in referring to a dog, but either *dog* or *inu* is finally generated in the NP. Two features in the model achieve this behavior. First, as we saw in Chapter 6, words are actually generated only when consituent nodes in a lexical GU receive activation from constituent nodes in a high-level GU such as NP. There are syntactic schemas for each language. Thus when it time for the noun in a English NP to appear, it is the NOUN/start node in ENGLISH-NP which fires. This activates the NOUN roles in English lexical GUs only, preventing activated GUs for other languages from having an effect. Figure 7.4 shows these relationships for the GU *DOG and the corresponding Japanese GU *INU.

Second, the model has inhibitory connections between words of different languages with the same or similar meanings. For example, when noun GUs for more than one language are activated for a given NP, only one of the word nodes filling the NOUN slots may fire. In multilingual contexts, words are distinguished for language. Thus in Figure 7.4, the word node "DOG" is joined to the general ENGLISH-WORD node. When a native speaker of Japanese is speaking English, the English words will all receive some priming and will tend to win out over Japanese words.

7.3. Relationships Between Schemas From Different Languages

When generating in one language, a multilingual speaker may find herself influenced by knowledge of another language. Cross-linguistic effects in generation are reflected in the use of target language words and structures which appear to be direct translations of non-target language words and structures, and in the actual intrusion of words from the non-target language.

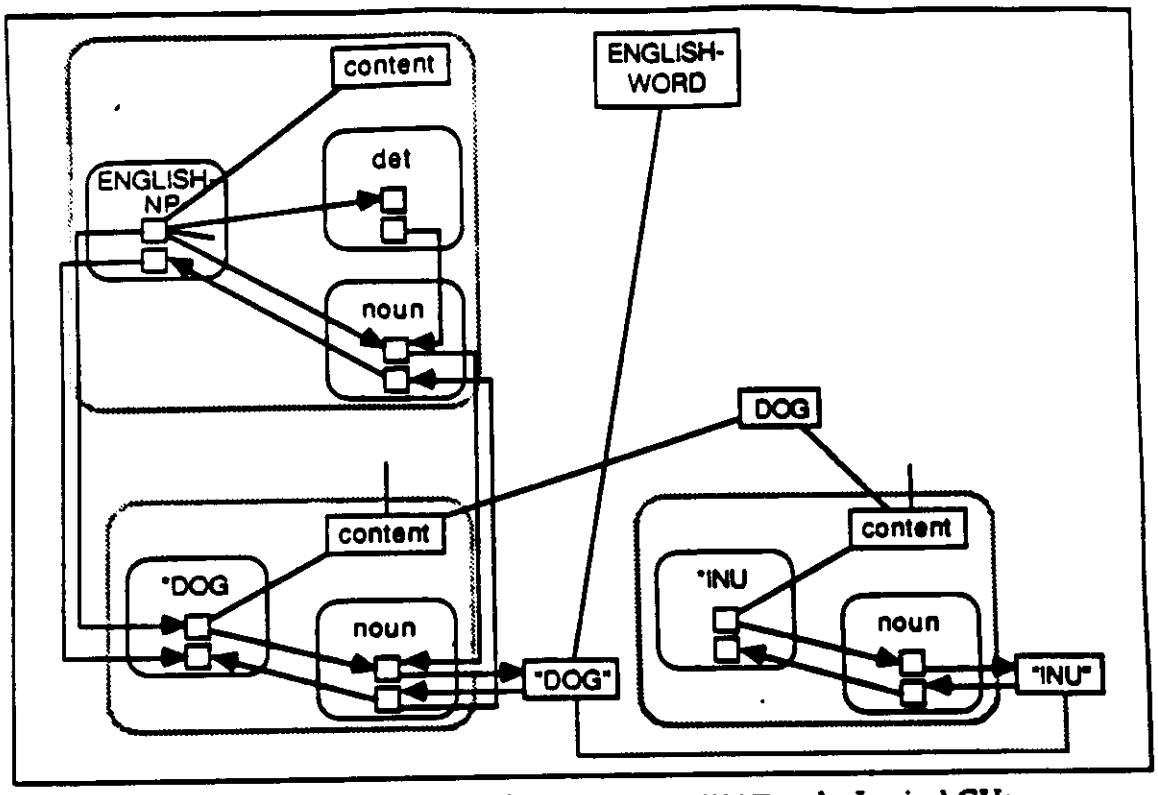

Figure 7.4: Relationships Between ENGLISH-NP and a Lexical GUs

Accounting for these cross-linguistic transfer effects requires consideration of the ways in which schemas from different languages can be associated. This section looks at five logical possibilities for these relationships, ranging from none at all to direct word-to-word translation associations. I examine evidence from multilingual generation that bears on which types of relationships actually occur. Note that, for a given speaker, there is no reason to assume that all L2 schemas would be related to L1 schemas in one or the other of these possible ways. There could be variation according to the type of item, the context in which the item was learned, or the stage in the overall acquisition process at which the item was learned.

7.3.1. No Direct Relationship

Given a schema in the L1, there is the possibility that a speaker will have no schema in her L2 with any direct correspondence to it. In such cases lexical items in the L1 must be rendered as entire phrases in the L2. In Amharic, for example, there is a verb *tɛgdɛrɛddərə* which could be defined as 'to refuse, out of excess politeness, something which one really wants'. Other examples are represented in the CLM model as GIs appropriate in highly specific situations. Consider the English phrase *tell me about it* in the sense 'I've had a similar unpleasant experience'. According to one Japanese informant, there is no corresponding expression in Japanese. In a situation where *tell me about it* would be appropriate in English, a Japanese speaker would apparently need to generate an utterance built up out of more than one linguistic unit rather than a pre-fabricated idiom such as the English expression. For example, the speaker might say something that would translate as *I can understand your complaining because I've been through the same thing*

myself. Expressions like *tell me about it* are interesting because of the challenge they present to second language learners.

7.3.2. Schemas with Common Content

When two schemas in different languages do correspond, at least to at certain extent, there are three possibilities for how they might be related: (1) Their CONTENT roles (or GOAL roles for GIs) might point off to the same concept or the same set of features. (2) Both schemas might inherit from a general schema which applies to both languages. (3) One schema might be treated as the translation of the other.

Figure 7.5 illustrates the general picture for GUs with common CONTENT. Here the CONTENT roles point to the same general concept (A), and corresponding roles (B and C) of the CONTENT roles have the same concept (D) as their values.

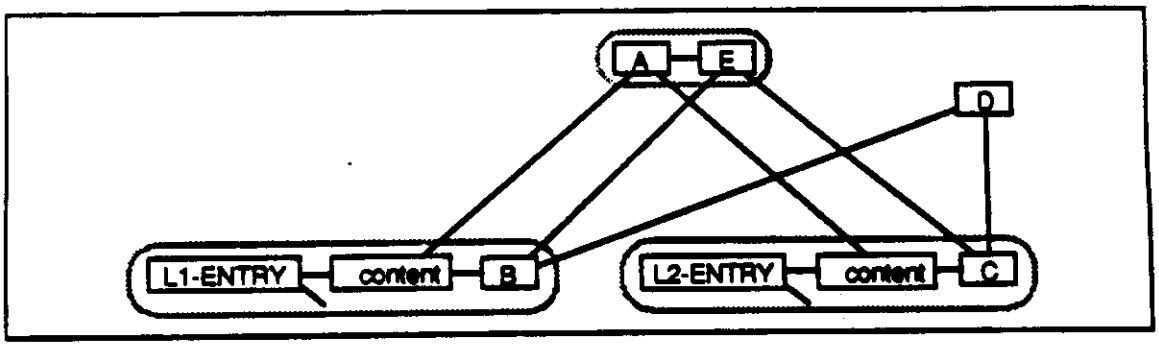

Figure 7.5: GUs with Common Content

Consider a case where an error does seem to be due to a common-content relationship. In example (1.9), repeated here as (7.1), A is a native speaker of Japanese. Her error is apparently related to the fact that Japanese has one single-morpheme word, *yu*, meaning 'hot water' and another word, *mizu*, meaning something like 'default water'.[19] Clearly *water* for this speaker means much the same as *mizu*.

(7.1) A: *The faucet's broken.*
B: *Which one?*
A: ***Water.*** (= 'cold water')

For Japanese speakers, unlike English speakers, the concepts associated with *yu* and *mizu* are basic-level categories (Rosch, 1977). That is, a speaker will always select *yu* when it is appropriate even when the hotness of the water is not at issue. For example, in generating a sentence corresponding to the English *you put too much water in the tub*, Japanese requires *yu* when the water is hot even when this is obvious to the hearer. Compare this to the distinction between ice and water for English speakers. Figure 7.6 shows how the knowledge that Japanese speakers have is represented in the CLM model.

The concept WATER has two subtypes, HOT-WATER, defined in terms of its TEMPERATURE, and NOT-HOT-WATER, defined in opposition to HOT-WATER. A WTA network represents the fact that WATER must be either HOT-WATER or NOT-HOT-WATER. COLD-TAP-WATER is viewed as a specialization of NOT-HOT-WATER. The linguistic knowledge consists of two GUs, *YU, with CONTENT pointing to HOT-WATER, and *MIZU, with CONTENT pointing to NOT-HOT-WATER. In generating a Japanese utterance

[19]Unless otherwise indicated, examples were collected by the author.

corresponding to the English one above, the speaker would access *MIZU via COLD-TAP-WATER and NOT-HOT-WATER.

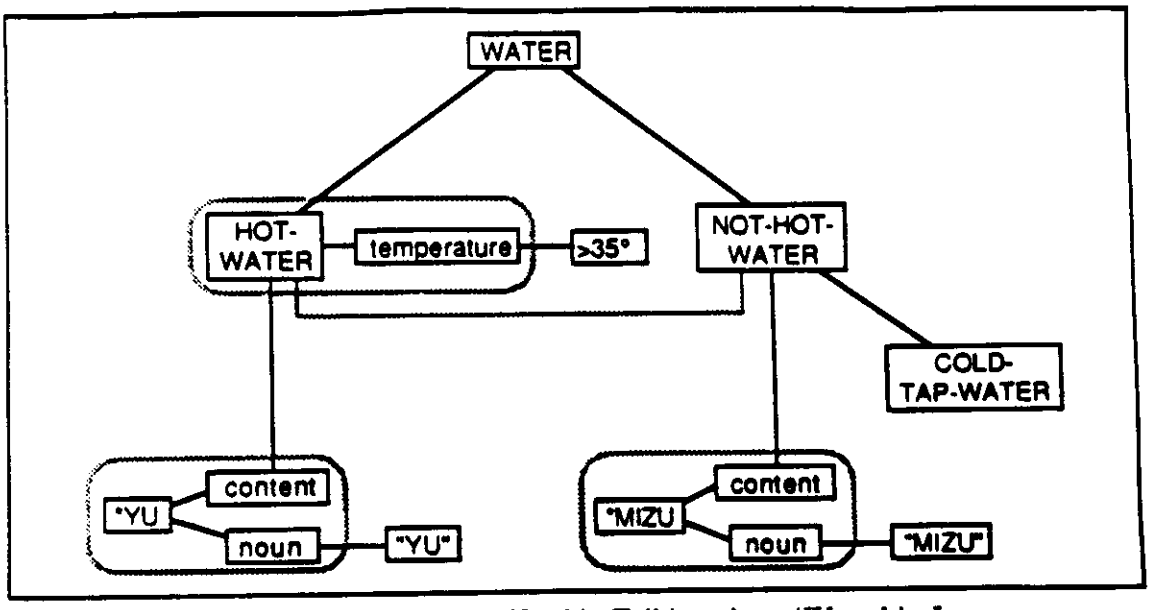

Figure 7.6: Some Knowledge Used in Talking about 'Water' in Japanese

What then is the source of the English error? One possibility is that the speaker accesses the English word via the Japanese word, that is, that she first finds the GU for *mizu* and there finds an association with *water*. This translation approach appears unlikely in this case, however, because the speaker is an advanced learner of English who normally does not rely on her first language. She has made the same error a number of times and does so without any apparent hesitation. It is also clear that the problem behind the error is a deep-seated one. The speaker is in fact fully aware that this is an error, but this awareness seems to have no effect on her behavior (except when she is clearly monitoring her speech and is able to catch the error before it is uttered).

In the CLM model this speaker's error is accounted for in terms of a GU for *water* which points through its CONTENT role to the same concept as the Japanese GU for the word *mizu*. That is, for this speaker *water* means NOT-HOT-WATER, rather than WATER. The relationships are shown in Figure 7.7. Note that the L1 schema is not accessed at all during the generation process, so we are not really seeing linguistic transfer in the usual sense, but rather the influence of a different conceptual categorization system.

7.3.3. Cross-Linguistic Schemas

A somewhat closer association than the common-content relationship is one involving a cross-linguistic schema which has schemas of the two languages as specializations. Figure 7.8 illustrates the general case. Here there is information about the CONTENT and one of the constituents in the cross-linguistic schema, and this information is shared by the two subtypes.

One use of cross-linguistic schemas is to represent knowledge about words that are recognized as cognates. For example, a person who knows both English and French would have a cross-linguistic GU for *important*, which would record the CONTENT and the spelling of the word. The different phonological realizations of the word would either

appear in language-specific subtypes of the cross-linguistic GU or be derivable from general grapheme-to-phoneme associations for the two languages. Figure 7.9 shows the relationships among these schemas. Here I assume that both words refer to a concept *IMPORTANT, which I have not bothered to analyze.

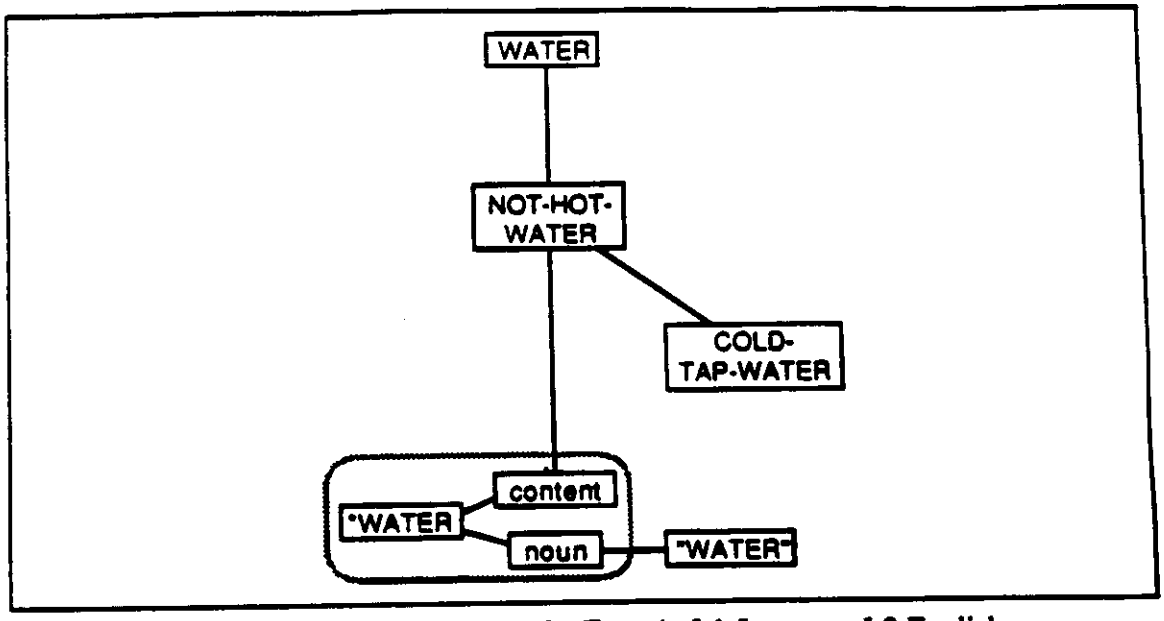

Figure 7.7: Source of Transfer Error in L1 Japanese, L2 English

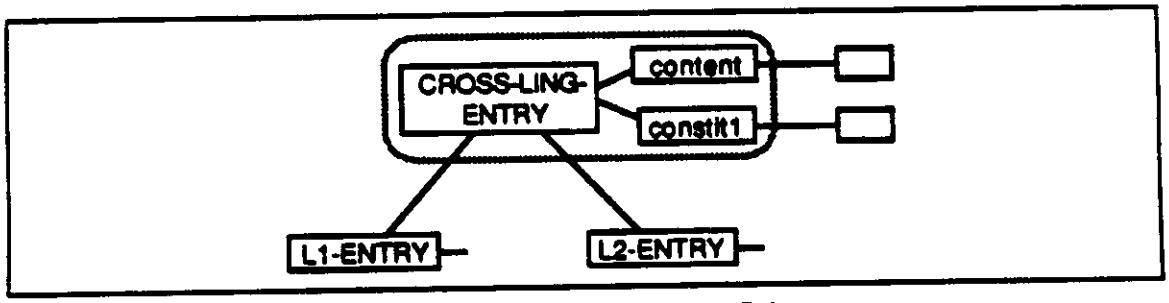

Figure 7.8: Cross-Linguistic Schema

Intra-sentential code-switching provides some evidence for a cross-linguistic notion of particular syntactic roles. Sentence (1.12) (Uyekubo, 1972), repeated here as (7.2), is an example.

(7.2) *ano okanemoti wa ozyoosan o **spoil** sita*
that rich:person TOPIC daughter ACCUS spoil did
'That rich man spoiled his daughter.'

Other than the intrusion of the English verb *spoil*, this is an acceptable Japanese sentence. In a code-switching context such as this, both speaker and hearer are bilinguals. For one reason or another (Hatch, 1976), the speaker selects a particular language in which to produce an utterance. In the example, the target language is Japanese. When a suitable GU in the other language is accessed during generation, however, there is no reason to discard it as there would be for a monolingual hearer. In the example, the speaker has in

mind a concept which I'll call INDULGE (without attempting to represent it). The English GU *SPOIL/INDULGE is selected on the basis of this concept, because for this speaker it is more strongly associated with the concept than the appropriate Japanese GU. The result is the integration of GUs from separate languages into a single utterance.

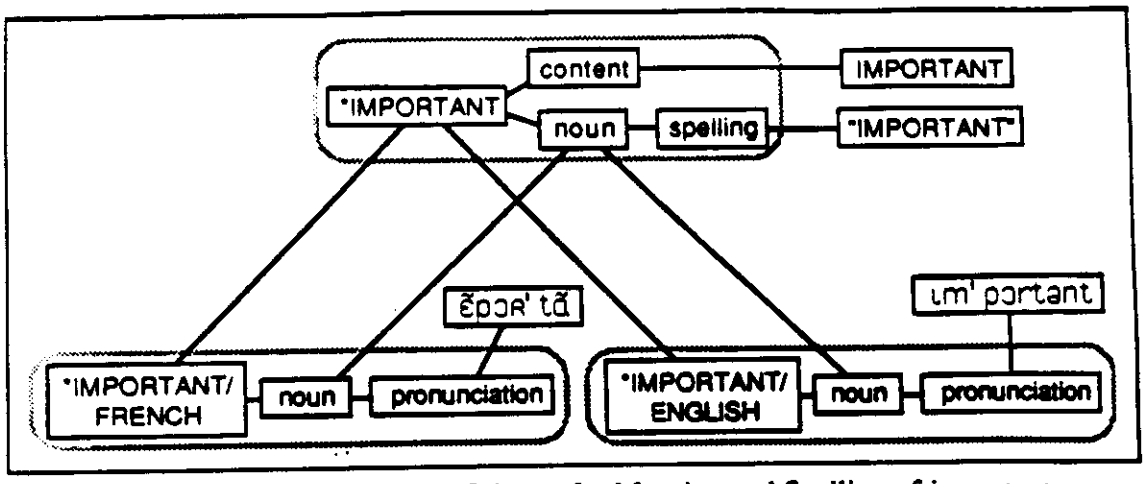

Figure 7.9: English-French Schema for Meaning and Spelling of *important*

What we are interested in here is the fact that the reference to the daughter in the sentence is made (appropriately) in the accusative argument of the clause. Knowledge about which constituent in a clause should refer to which semantic argument is embodied in the role binding information found in clause-level schemas. For example, the *SPOIL/INDULGE GU includes the information that the DIRECT-OBJECT refers to the semantic OBJECT, that is, the person being indulged. However, this is information about English. How does the speaker end up with the Japanese accusative marker on the phrase referring to the OBJECT?

Apparently the speaker recognizes a general correspondence between the English DIRECT-OBJECT and the Japanese ACCUSATIVE. Both appear in clauses referring to what we might call TRANSITIVE-EVENTs, that is, events with semantic OBJECTs in addition to ACTORs. This English-Japanese correspondence is represented in the cross-linguistic GU for TRANSITIVE-CLAUSE. Figure 7.10 shows the spread of activation during the role binding process for the argument referring to the daughter in sentence (7.2).

The sequencing of clause constituents is guided by the JAPANESE-CLAUSE (not shown) and JAPANESE-TRANSITIVE-CLAUSE GUs because Japanese is the target language for this sentence. When it comes time for the ACCUSATIVE constituent to be produced, the ACCUSATIVE node[20] fires, sending activation to the OBJECT-REF role in the cross-linguistic schema. This in turn sends activation to the DIRECT-OBJECT node in the ENGLISH-TRANSITIVE-CLAUSE schema, which activates the DIRECT-OBJECT role in the *SPOIL/INDULGE GU, initiating the role association process. The merged node activates the OBJECT role in the INDULGE schema and, though this is not shown in the figure, eventually the node representing the daughter to be referred to. Note that there is a seeming conflict because both the English DIRECT-OBJECT and Japanese ACCUSATIVE nodes have fired. The only real conflict, however, is in the position of the constituent, before the verb

[20]Actually what fires at this point is the ACCUSATIVE/start node, but the start-end node distinction will be ignored here and elsewhere in this chapter since it is not relevant to the issues being discussed.

for Japanese, after for English. With respect to position the Japanese schemas will dominate because Japanese is the target language.

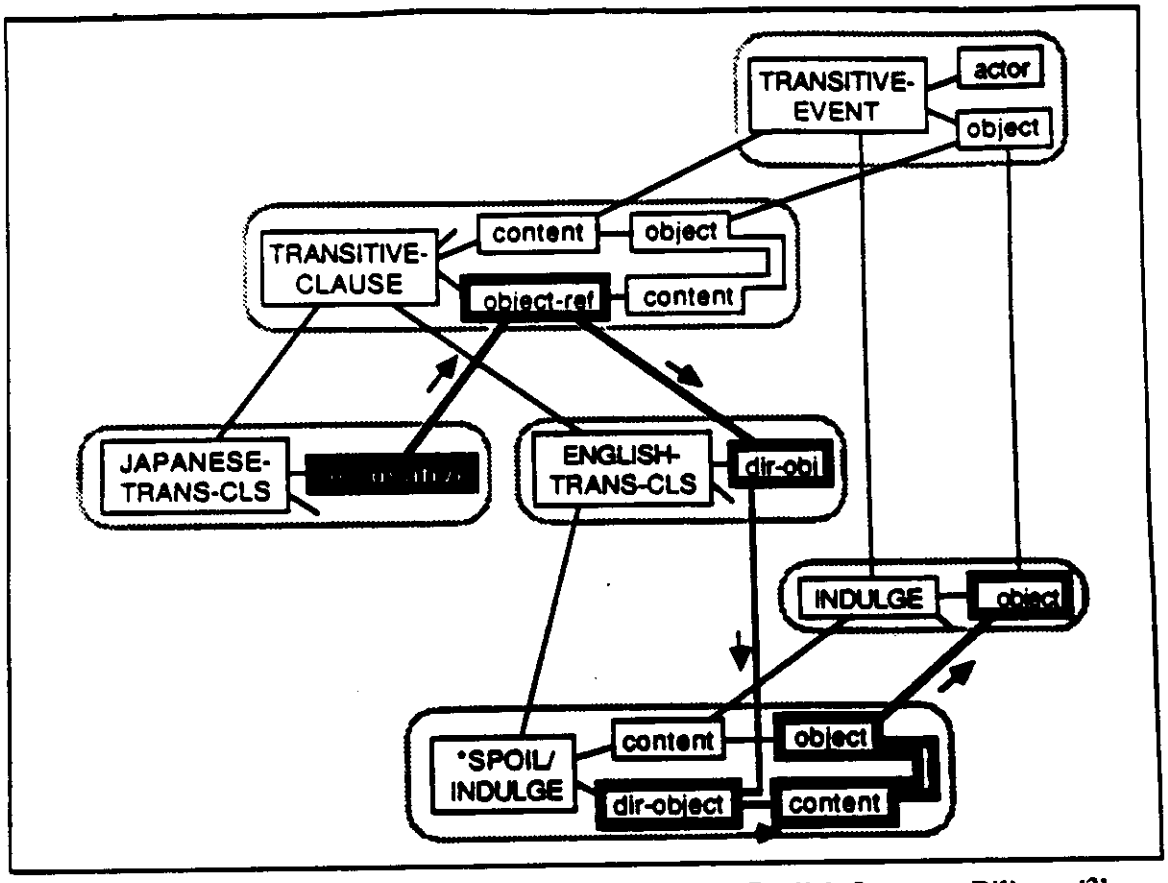

Figure 7.10: Cross-Linguistic Syntactic Role for an English-Japanese Bilingual[21]

7.3.4. Schema-to-Schema Translation

An even closer association between schemas from two different languages is one which treats a schema in one language as a translation of a schema in another language. This is implemented using TRANSLATION roles in schemas. Figure 7.11 shows the general relationship. Note that there is a directionality to the association. The relationships shown in Figure 7.10 would be used to get from the L1 schema to the L2 schema but not in the reverse direction. However, for fluent speakers of a second language, there may also be translation relationships in the other direction.

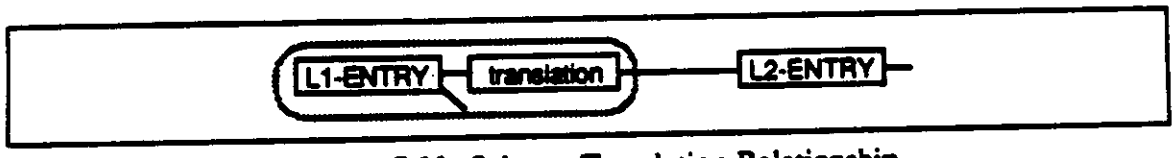

Figure 7.11: Schema Translation Relationship

[21]Not shown in the figure are the morphosyntactic properties of the corresponding constituents in English and Japanese, in particular the fact that the Japanese ACCUSATIVE takes the case marker *o*.

One place where we might expect to find schema translations is in the representations used by experienced translators. The TRANSLATION role would allow a translator to get from input in one language to output in the other in a relatively direct fashion, in some cases without making any reference to the content of the input utterance.

The translation assocations also seem to have a role in generation contexts where explicit translation is not called for. The process is similar to the kind of intra-sentential code-switching seen in sentences like (7.2), except that, instead of directly making use of the schema from the wrong language, this schema is translated into a target language schema. Say a Japanese speaker wants to generate an English NP referring to some pepper. She can either access the English schema directly, access it through the corresponding Japanese, or use a combination of these paths. Figure 7.12 shows what would happen in the second case.

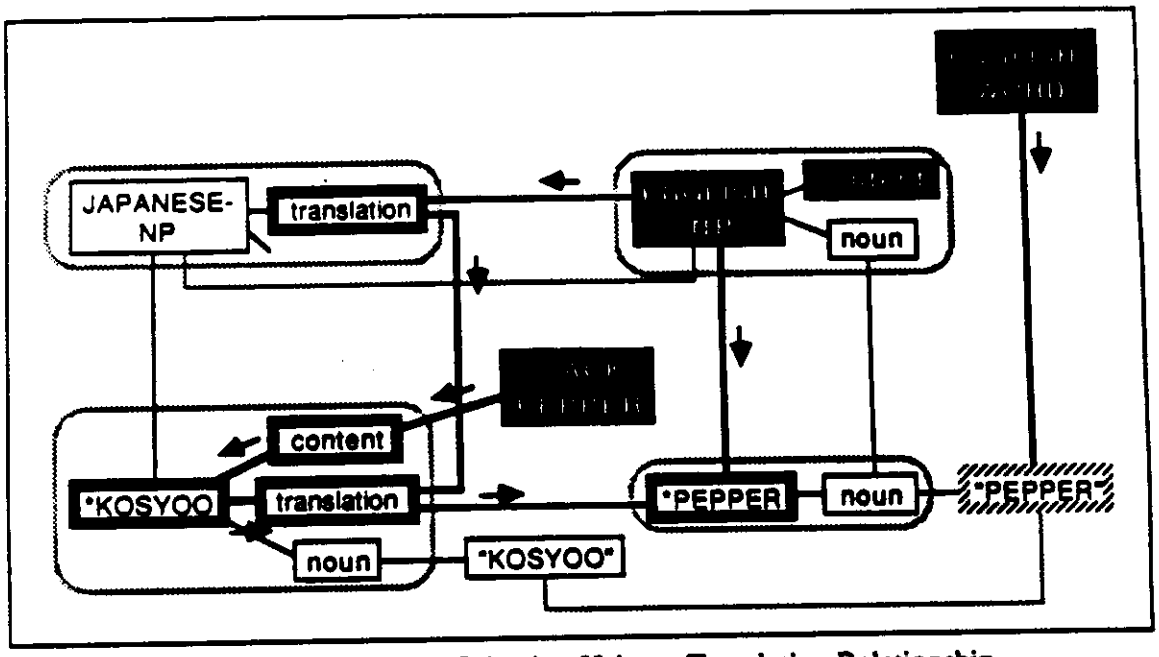

Figure 7.12: Lexical Selection Using a Translation Relationship

The schema selection process starts with the firing of a node representing the referent (not shown in the figure), resulting in the firing of the general node for BLACK-PEPPER (or a set of feature nodes). For this speaker this concept is associated with only a Japanese GU. The firing of the ENGLISH-NP node activates the TRANSLATION node in the JAPANESE-NP GU, and this primes all TRANSLATION roles of Japanese noun GUs. In this case the TRANSLATION role in the *KOSYOO GU can fire and activate the appropriate English GU, *PEPPER. Since two GUs have been selected, two word nodes, "KOSYOO" and "PEPPER", are both candidates for the head noun of the NP. These inhibit one another via a WTA network, and the English word predominates because of the priming that all English words receive when the hearer is an English speaker.

For fluent bilinguals the translation strategy may also be used in the other direction, from the L2 to the L1. Sentence (1.11), repeated here as (7.3), is an example where the L1 is Amharic and the L2 English, but this type of process occurs among Japanese speakers who have been immersed in an English-speaking environment as well. The utterance is ordinary Amharic except for the use of the verb *wessede*. This verb corresponds to one of

the general senses of English *take*, roughly 'transfer control of an object to the actor'. Thus *wɛssɛdɛ* would be appropriate in the Amharic versions of the English sentences *Mary took John's watch* and *take my coat, I don't need it*, but not in the translations of the sentences *I'll take you to the airport, he took a shower*, and *he took another class*.

(7.3) *lela kɨfɨl wɛssɛdɛ.*
other class he:took
'He took another class.'

Apparently the speaker has accessed the English GU for taking a class while intending to produce an Amharic utterance and then literally translated the English expression into Amharic. Figure 7.13 shows the path that is followed in the schema selection process for the verb.

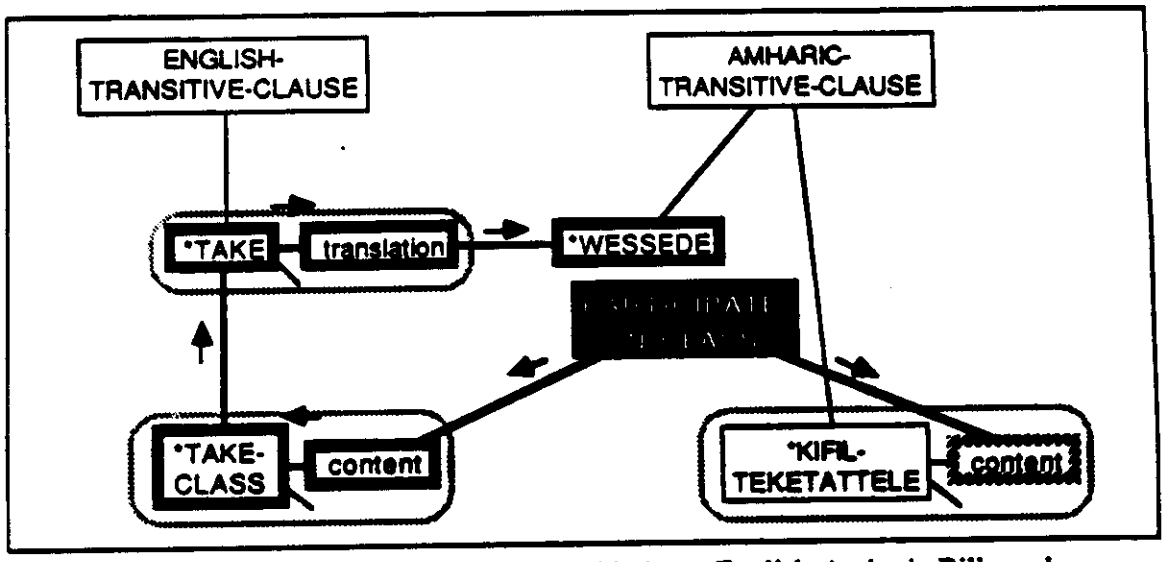

Figure 7.13: A Translation Relationship in an English-Amharic Bilingual

Activation from the input concept leads to the firing of the PARTICIPATE-IN-CLASS node (or a comparable set of features), and this in turn sends activation to the two GUs which this speaker has for the concept, one for each language. In this case, the connection to the Amharic GU, *KIFIL-TEKETATTELE, is too weak for it to fire immediately..The connection to *TAKE-CLASS, on the other hand, is strong, and *TAKE-CLASS fires as a result. There is no translation specified for this GU, but there is one in the more general *TAKE GU. Activation from *TAKE-CLASS spreads to this *TAKE GU, and the translation association yields the Amharic GU *WESSEDE. In this case the association is a faulty one because it implies equivalence between two verbs whose senses are quite different despite some overlap.

7.3.5. Word-to-Word Translations

The closest association possible between units of knowledge in two languages is a translation association between words, as opposed to schemas. Figure 7.14 shows the what this would look like for a person who knows the English noun *pepper* only as the translation of the Japanese noun *kosyoo*. Note this relationship leaves the English word *pepper* completely isolated from any general knowledge of English noun phrases. That is,

knowledge in this form would only be useful for a word-for-word replacement of L1 patterns.

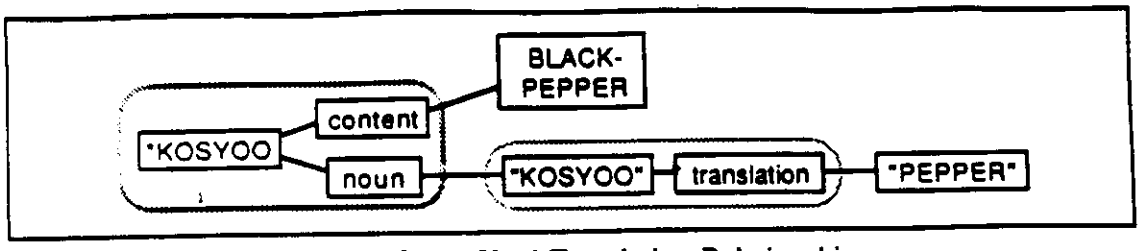

Figure 7.14: Word Translation Relationship

We might expect word translation relationships to develop in very low-level learners, in particular those studying in formal contexts where the memorization of lists of translation pairs is encouraged. The early utterances of second language learners and pidgin speakers are often described as relexifications of first language utterances, and we might look there for evidence of word translation relationships. The following is an example of a Hawaiian Pidgin English sentence produced by a native speaker of Japanese (Bickerton, 1981).

(7.4) *da pua pipl awl poteito it.*
'The poor people only ate potatoes.'

Japanese verb-final word order is apparent in this and other of Bickerton's examples. One possible account would have this sentence generated using Japanese schemas entirely, including the high-level GU JAPANESE-CLAUSE, specifying clause-final position for the verb, and the lexical GU *TABERU, specifying a VERB which translates to English *eat*. This relexification view of the generation of sentence (7.4) is shown in Figure 7.15.

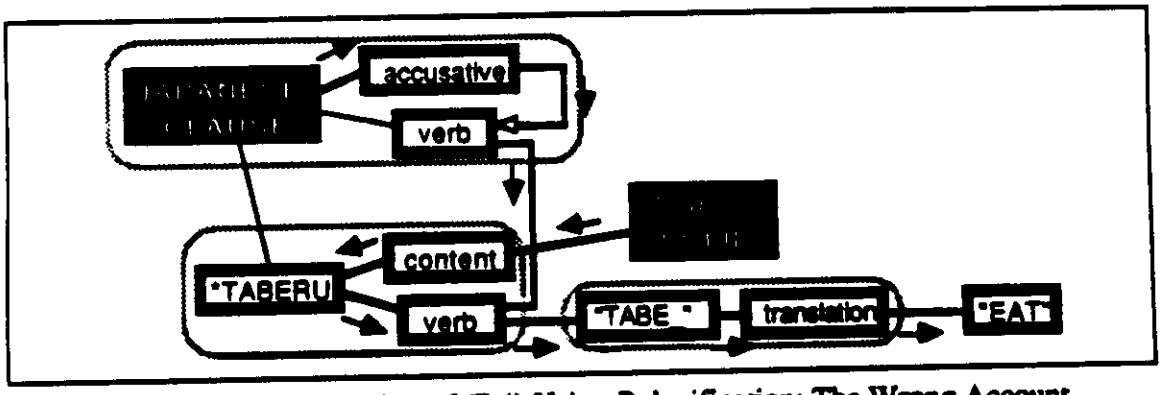

Figure 7.15: Generation of (7.4) Using Relexification: The Wrong Account

The open-headed arrow is an abbreviation for a sequence relationship; here the ACCUSATIVE constituent of the Japanese clause is specified as preceding the verb. The generation is driven by the JAPANESE-CLAUSE and *TABERU GUs. When it comes time for the VERB to be produced, it is replaced by the English translation *eat*.

There is a problem with this account, however. These associations leave no room for the later integration of the knowledge into the syntax of the L2. If we hold to the view that syntax is just generalized lexical patterns (Peters, 1983; Wong Fillmore, 1977), then there is no way for syntax to ever arise in a system like the one in Figure 7.15.

A more reasonable account is to have full-fledged phrasal schemas in the L2 which contain relatively little syntactic information in the early stages of acquisition. This alternative is shown in Figure 7.16 for the present example.

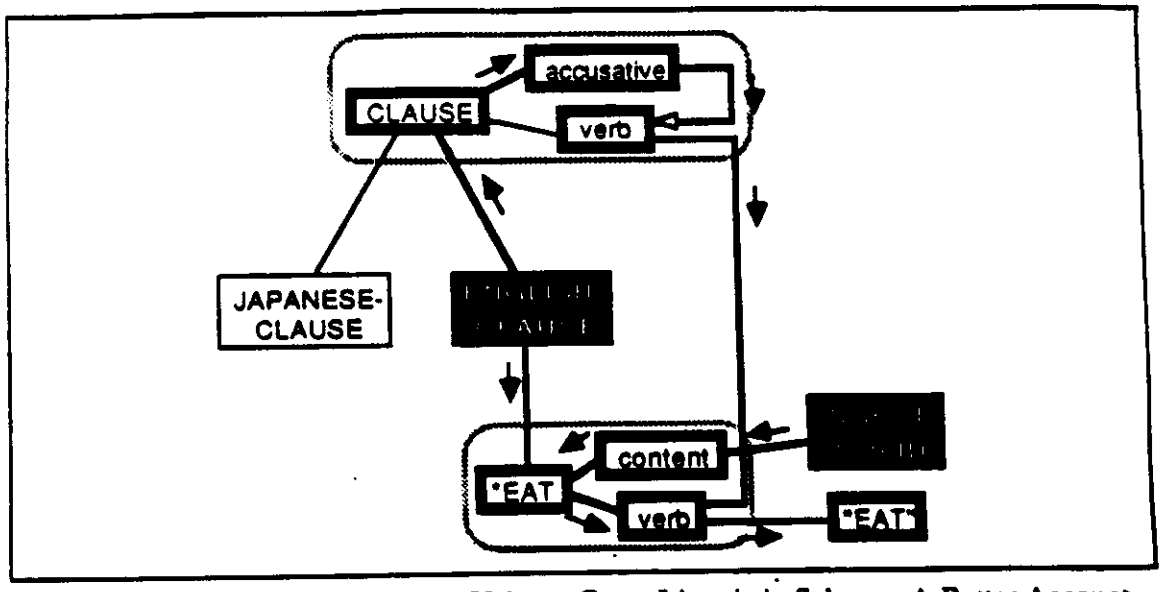

Figure 7.16: Generation of (7.4) Using a Cross-Linguistic Schema: A Better Account

The speaker has a schema for *eat* pointing to the schema ENGLISH-CLAUSE, but this high-level GU currently has nothing to say about constituent order. ENGLISH-CLAUSE is in turn a specialization of the cross-linguistic schema CLAUSE, which at this stage still specifies Japanese word order. On this view, the utterance gets its ordering from what is currently known about constituent ordering in general, namely, what is known about constituent ordering in Japanese clauses.

7.4. Coping with Gaps in L2 Knowledge

In Section 7.3 we saw how some of the errors made by second language learners can be accounted for in terms of faulty associations which have been acquired. There is another possible source or errors: a speaker may have no association at all for a given concept and will have to cope with this deficit in a way that is likely to result in an error. In many cases an error can be interpreted in more than one way. In this section I consider some errors which apparently result from missing linguistic knowledge.

Gaps in linguistic knowledge lead to two sorts of errors. On the one hand, there are errors with grammatical morphemes which express largely redundant meanings in the target language. Speakers usually seem to be unaware of such errors, thus they need no particular strategies to deal with the gaps that result in these errors. There are two of them in the following utterance; note that they do not interfere significantly with communication.

(7.5) *He* **wonder** *because there are only two* **basket**.

In terms of the CLM model, the missing knowledge in such cases is of the paradigmatic type, which forces the speaker to make particular decisions whether or not they contribute to her communicative goals. Thus *basket* is not plural in this utterance either because there

is no WTA network which forces a COUNTABLE-NP to be either a PLURAL-NP or a SINGULAR-NP or the WTA connections exist but are still weak. I will not be concerned further with this type of error.

7.4.1. Lexical Gaps

On the other hand, there are errors which are potentially more serious because a misunderstanding may result. The problem may be lexical or grammatical. Sentence (1.10), repeated here as (7.6), is an example of a lexical gap. The speaker is referring to a person's leaving property (some iron bars) with a friend for safekeeping.

(7.6) *He uh keep uh he kept his ... property so he* ***deposited*** *uh .. his friend. Do you understand?*

This appears to be an example of transfer from L1. Japanese has a verb *takusu* which has the general sense of 'leave something for safekeeping', including money in a bank. Interestingly, the speaker of (7.6) used this verb when describing the same event in Japanese.

The following, from Takahashi (1985), is another example of lexical overgeneralization, apparently under the influence of the L1.

(7.7) *I don't like this tea because it's too* ***thick*** *(= 'strong').*

Japanese has an adjective *koi* which, when used to describe a solution, has the general sense 'concentrated', corresponding to both English *thick* and *strong*.

Let us consider (7.6) in some detail. As already noted, the error may be a result of an existing faulty GU. This possibility is shown in Figure 7.17. The GU for *deposit*, like the GU for the Japanese verb *takusu*, is associated with the general notion of LEAVE-FOR-SAFEKEEPING, which includes the type of act referred to in (7.6) as well as the depositing of money in the bank. With this GU in memory, the generation of (7.6) is straightforward since the event referred to is an instance of the concept LEAVE-FOR-SAFEKEEPING. This account would explain the error in the same way as I explained the error in (7.1), for which the speaker has an English GU *WATER which points to the wrong concept, NOT-HOT-WATER.

```
(*DEPOSIT is-a ENGLISH-CLAUSE
    (content LEAVE-FOR-SAFEKEEPING)
    (noun "DEPOSIT"))
```

Figure 7.17: Faulty GU for *deposit*

On the other hand, based on the speaker's dysfluencies, a more likely account is that the error results from some sort of coping behavior. Let us assume that the speaker has learned the appropriate meaning for *deposit* in the sense of putting money in a bank. A portion of the GU is given in Figure 7.18.

```
(*DEPOSIT is-a ENGLISH-CLAUSE
    (hearer ENGLISH-SPEAKER)
    (verb "DEPOSIT")
    (content TRANSFER-OF-CONTROL
        (actor ?A)
        (object (MONEY ?O))
        (source ?A)
        (recipient BANK)
        (duration TEMPORARY)
        (purpose1 PREVENT
            (object LOSE
                (object ?O)))
        (purpose2 INCREASE-VALUE
            (object ?O))))
```

Figure 7.18: GU for *deposit*

In order to see how this GU might have been selected, we need to consider what features there are to be matched in the input. A possible representation for the input concept for (7.6) is shown in Figure 7.19.

```
(TRANSFER-OF-CONTROL3 is-a TRANSFER-OF-CONTROL
    (actor MAN6)
    (object IRON-BARS9)
    (recipient MAN8)
    (duration TEMPORARY)
    (purpose1 PREVENT
        (object LOSE
            (object IRON-BARS9))))
```

Figure 7.19: Possible Content for Sentence (7.6)

The GU in Figure 7.18 matches this instance on some features. Both are of type TRANSFER-OF-CONTROL, and both have TEMPORARY DURATION. A purpose of the act in both cases is the prevention of the loss of the transferred object. Where the GU does not match is for the stipulations that the object is money, that the recipient is a bank, and that a further purpose of the transfer is the earning of interest.

Given a particular set of connection weights, activation from this input structure may not be enough to cause the CONTENT of *DEPOSIT to fire. However, recall that this role is linked to the CONTENT roles of other clause GUs through a WTA network. When the WTA hub node is activated, it first waits for one of the network elements to fire on its own and if this does not happen, it sends additional activation to the network members, and the one with the highest activation normally fires. In this case, this is the CONTENT of *DEPOSIT. This delay in firing is in part a way of modelling the uncertainty experienced by the speaker in generating a sentence such as this. Note that this account is identical to that used in explaining the comparable example for L1 generation described in section 4.4.1: *I'd like to deposit jewelry*.

This explanation of the error in (7.6) makes no reference to the L1 system, though. That is, the fact that there is a Japanese GU for a verb with the general meaning 'leave for

safekeeping' had no bearing on the speaker's decision to use the verb *deposit* in more or less this sense. Based on the frequency of errors like those in (7.6) and (7.7), it appears that the existence of the Japanese word does have an effect. That is, all other things being equal, we would expect an error like that in (7.6) to be more likely from a Japanese speaker than from a speaker of a language which does not have a verb like *takusu* with the general sense of 'leave for safekeeping'. How can this effect be achieved within the context of the CLM approach to L2 generation?

The explanation depends on the sharing of conceptual features between the GUs for *deposit* and *takusu* and also on the possibility that, given a particular input notion, only a subset of the relevant features of the notion will result in firing nodes. Figure 7.20 shows some of the relationships between the two schemas.

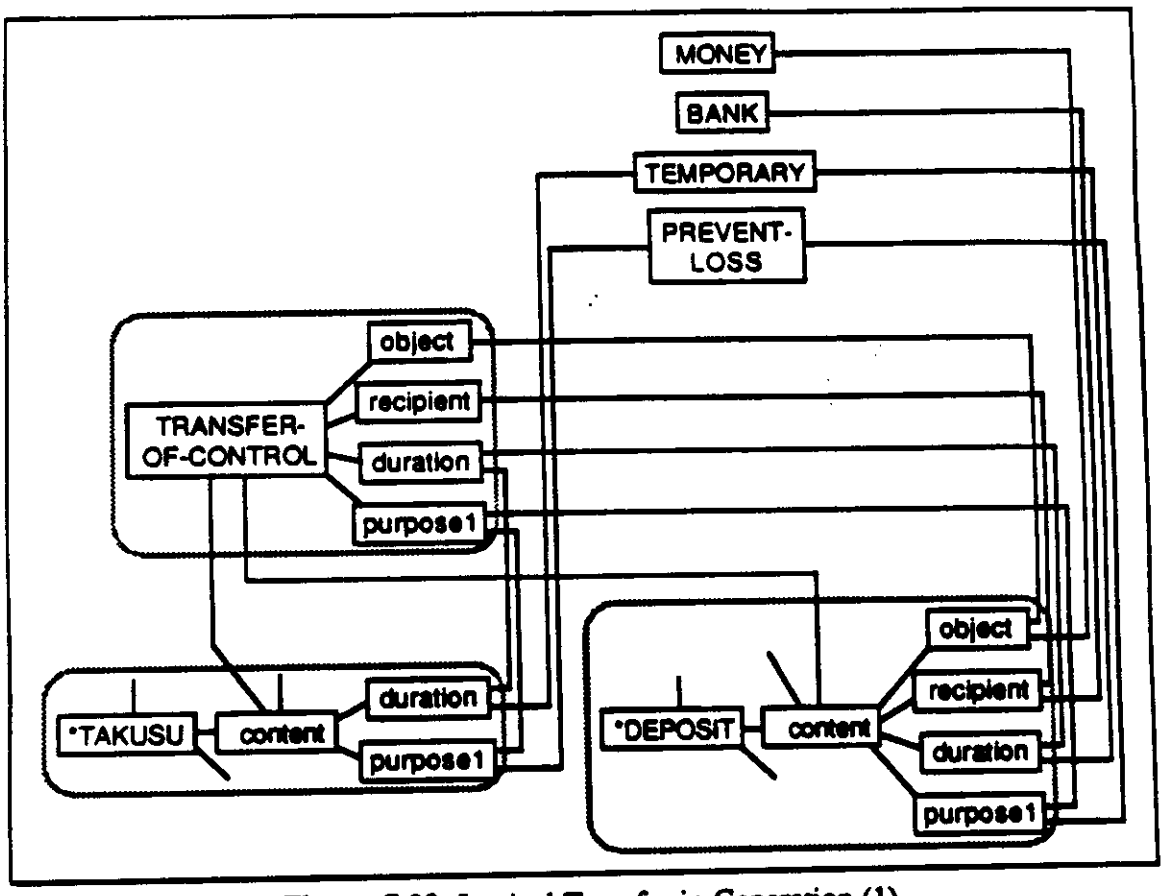

Figure 7.20: Lexical Transfer in Generation (1)

Only four features are shown, the OBJECT, the RECIPIENT, the DURATION, and one PURPOSE. For both GUs the DURATION is TEMPORARY, and one PURPOSE is the prevention of loss (here abbreviated as a single PREVENT-LOSS node). *DEPOSIT also has the feature that the OBJECT be MONEY and the RECIPIENT be a BANK.

Figures 7.21 and 7.22 show the series of events that would lead to *DEPOSIT firing for sentence (7.6) in a way that is dependent on the existence of *TAKUSU.

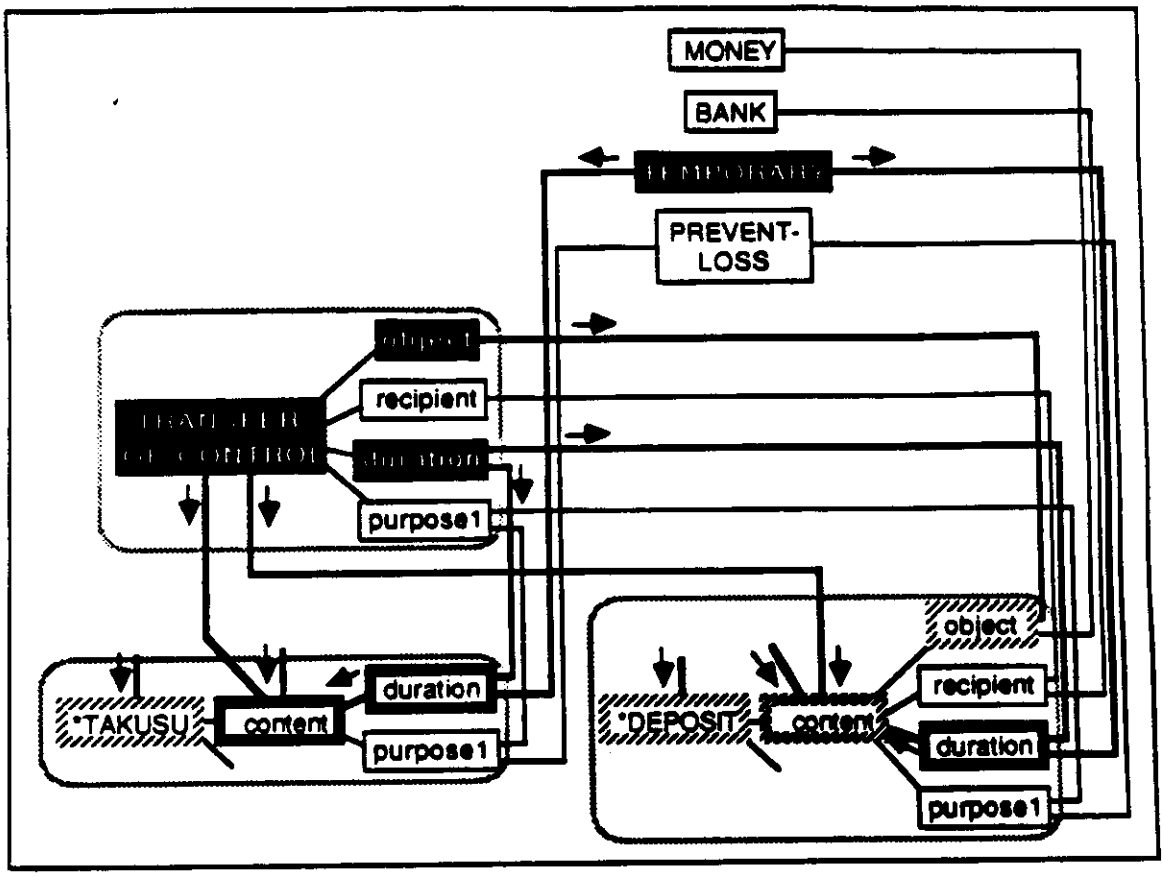

Figure 7.21: Lexical Transfer in Generation (2)

Assume first that only a subset of the relevant features is activated. To keep the example simple, we will assume that the only activated features are those for the OBJECT and the DURATION of the input concept, as well as the fact that it is an instance of TRANSFER-OF-CONTROL. The result is the firing of the blackened nodes in the figure. Activation from these nodes converges on the CONTENT nodes of both GUs and on some of the roles of the CONTENT nodes. Note that the OBJECT role in the *DEPOSIT GU does not fire because it received no activation from its value node MONEY. At this point *TAKUSU:CONTENT is more likely to fire than *DEPOSIT:CONTENT because a greater proportion of its roles have fired.

Figure 7.22 shows how the firing of *TAKUSU:CONTENT can result in the selection of *DEPOSIT. Once *TAKUSU:CONTENT fires, it sends activation to those of its roles which have not yet fired: in the figure, the PURPOSE1 role. The firing of this node and the node for its value, PREVENT-LOSS, in turn lead to the firing of the corresponding role of *DEPOSIT:CONTENT, increasing the likelihood that *DEPOSIT:CONTENT itself will fire. If it does, the *DEPOSIT GU will be selected, as indicated in the figure. Note that the Japanese GU *TAKUSU is also selected because its CONTENT role has fired, but it will not compete with *DEPOSIT because it is a the high-level GU ENGLISH-CLAUSE which is guiding the generation of the sentence. Thus when the verb of the clause is to be generated, the VERB/start node in ENGLISH-CLAUSE will activate VERB/state in *DEPOSIT, but not in *TAKUSU.

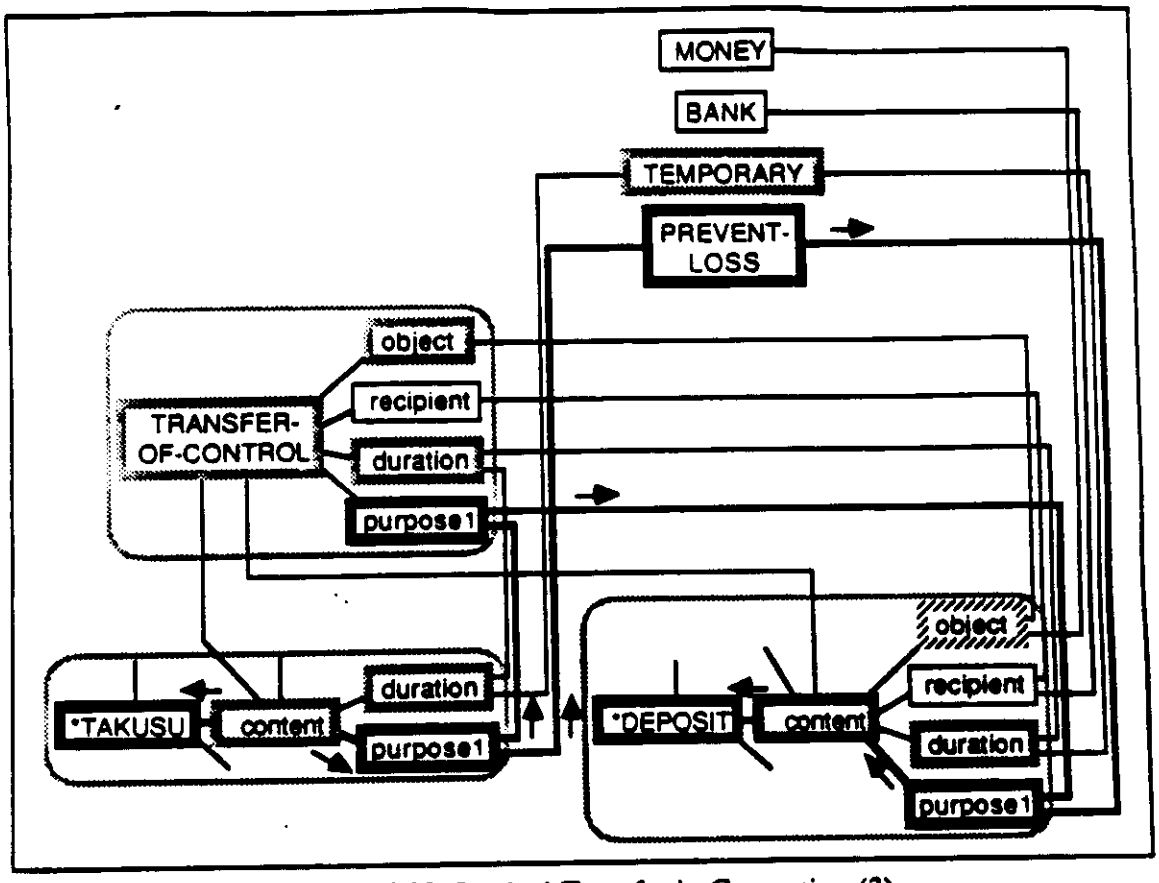

Figure 7.22: Lexical Transfer in Generation (3)

Note that it is only the probability of this sort of transfer that is affected by the presence of the more general L1 schema. In the example, if the full set of features common to the two schemas had been activated from the input, *DEPOSIT would have fired whether or not *TAKUSU had been available.

7.4.2. Syntactic Gaps

In other cases where there is a perceived gap in the speaker's knowledge of the L2, the problem is the unavailability of an appropriate syntactic pattern to express a particular relationship. Sentences (7.8) illustrate such a problem with the English relative clause pattern. The native Japanese speaker has read a story in Japanese and is retelling it in English and in Japanese. The story concerns two men who have found a hatchet lying on the ground. At this point the narrator wants to assert the fact that the man who had lost the hatchet was searching for it. In relating the story in Japanese, she uses a relative clause to distinguish this man from the other two:

(7.8a) *Sosite sono teono o otosita otoko ga*
then that hatchet ACC dropped man NOM
'Then the man who dropped the hatchet

sono teono o sagasite imasita.
that hatchet ACC was seeking
was looking for the hatchet.'

In the English version, however, she seems to be unable to use a relative clause to perform the function and instead produces two sentences:

(7.8b) *And then that saw is somebody's.*
Somebody's looking for its saw.

There are, of course, problems with these sentences other than the one which interests us here: *saw* is used for 'hatchet' and *its* for 'his', and *somebody* in the second sentence incorrectly indicates a definite reference. To facilitate comparison with (7.8a), then, it will help to correct these other errors and recast (7.8b) in the following form:

(7.8b') *And the hatchet was somebody's.*
(Somebody had dropped the hatchet.)
He was looking for his hatchet.

The effect of the two sentences is to imply to the hearer that he cannot infer the existence of the owner of the hatchet or of the act of losing. In this context this effect is clearly not what the speaker intended.

In what follows I will sketch an account of this error in terms of the CLM model. There are complications related to the planning of a description for the NP, however, which have not yet been worked out in the framework of the model.

What we have in (7.8) is an intent to produce a reference to a person to whom a particular fact is to be attributed. A fact attributed to a referent as part of an NP normally takes the form of an adjective phrase or a relative (adjective) clause. The form in which the modifier is realized depends on the way in which the fact is lexicalized. If there is an adjective lexical GU for the fact, the modifier will take the form of an adjective phrase. If there is a verb (clause) lexical GU for the fact, the modifier will take the form of a relative clause.

For this example, the referent, whom I will call MAN4, has several facts associated with him in the speaker's memory. One of these represents his ownership of the hatchet. In the generation of the NP in question, activation spreading from MAN4 activates the OWNER role of this fact, and I will assume that this activation constitutes the decision to attribute the ownership fact to the referent in the NP. Figure 7.23 shows how the NP might be generated correctly by a speaker with all of the requisite facts of the language.

Activation spreading from the ENGLISH-NP node primes the SUBJECT-RELATIVE-CLAUSE node, representing relative clauses in which the referent of the subject is the same as the referent of the NP that the clause is embedded in. This relationship is indicated by the connection from ENGLISH-NP:CONTENT to ENGLISH-NP:SUBJECT-RELATIVE-CLAUSE:SUBJECT:CONTENT. Activation from the referent of the NP being generated, MAN4, leads to the firing of OWN7:OWNER, then the general OWNER role, and then the merged role in the *OWN GU representing the equivalence of the SUBJECT:CONTENT and the CONTENT:OWNER for this schema. This node fires because it has also received activation from ENGLISH-NP:CONTENT. At this point the *OWN GU is selected. This provides one way to talk about ownership in English: *he owned the hatchet.* Activation from *OWN:SUBJECT leads to the firing of ENGLISH-CLAUSE:SUBJECT, which then supplies extra activation for SUBJECT-RELATIVE-CLAUSE.

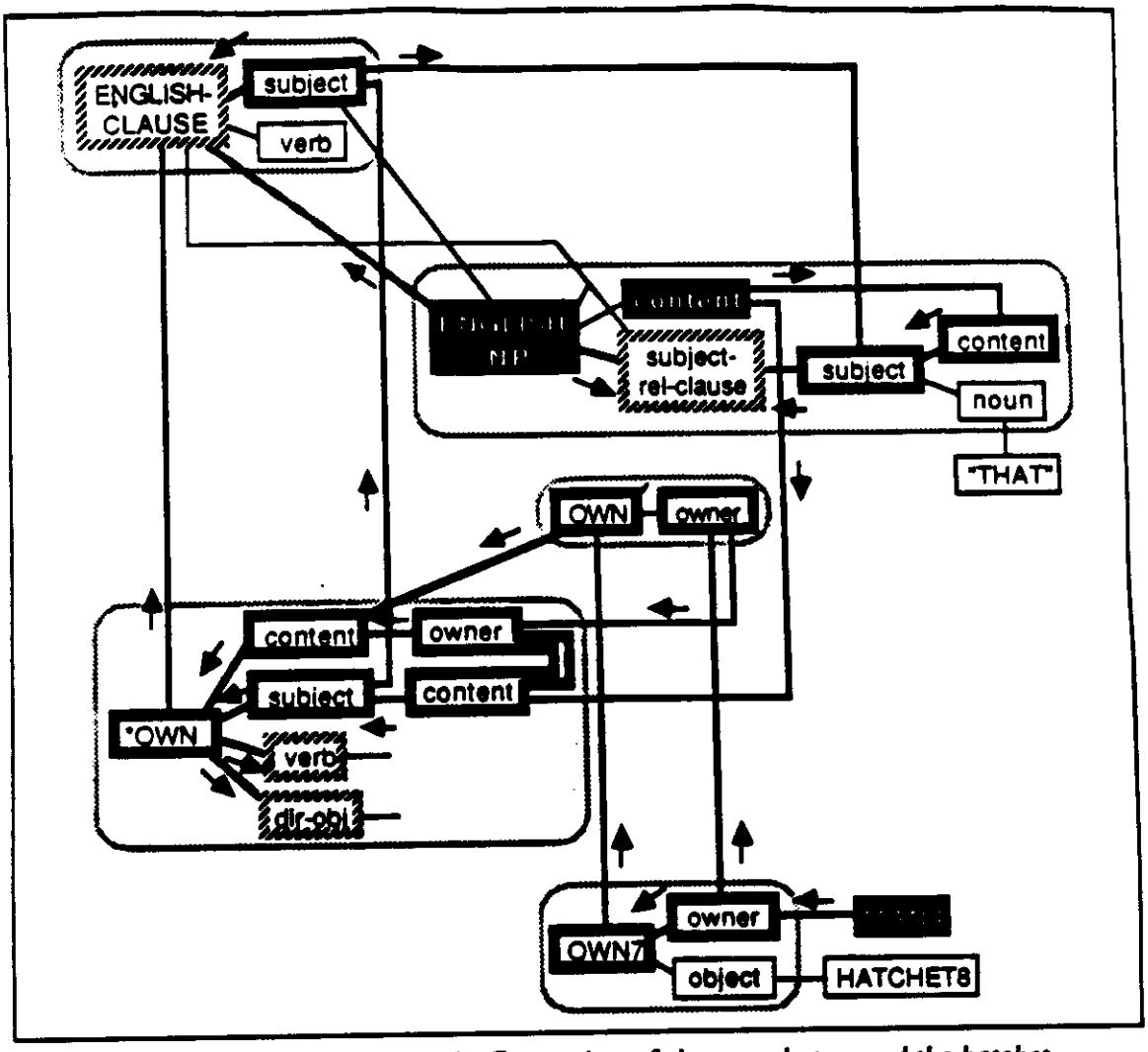

Figure 7.23: Initial Stages in Generation of *the man that owned the hatchet*

Since this process takes place within the context of the generation of an NP, the clause should not be generated immediately, but should appear in the appropriate relative clause slot following the head noun of the NP. ENGLISH-NP inhibits ENGLISH-CLAUSE, preventing the *own* clause from being generated when *OWN is selected. Activation spreading through ENGLISH-NP sequences the constituents in the NP. When the head noun has been produced activation passes to the start nodes for optional following constituents, including prepositional phrases and relative clauses. Since the SUBJECT-RELATIVE-CLAUSE has priming, it fires at this point, sending activation to ENGLISH-CLAUSE and initiating the generation of the relative clause *that owned the hatchet*. Note that SUBJECT-RELATIVE-CLAUSE specifies that its subject take the form of a relative pronoun *that*.

The problem in (7.8b) could be due to one or more gaps in the speaker's knowledge of English. One possibility is that the speaker has no knowledge of English relative clauses. Another is that she is unfamiliar with the *own* pattern and instead knows only the pattern found in the actual utterance, *X be Y's*. This is, of course, a legitimate way to convey ownership in English, but in this sentence it would force the use of the relatively complex possessive relative clause form: *the man whose hatchet it was*. A

speaker at this level of proficiency would be very unlikely to know such a pattern. Figure 7.24 shows a possible picture of what this speaker knows and part of the path of activation resulting in the first clause in (7.8b).

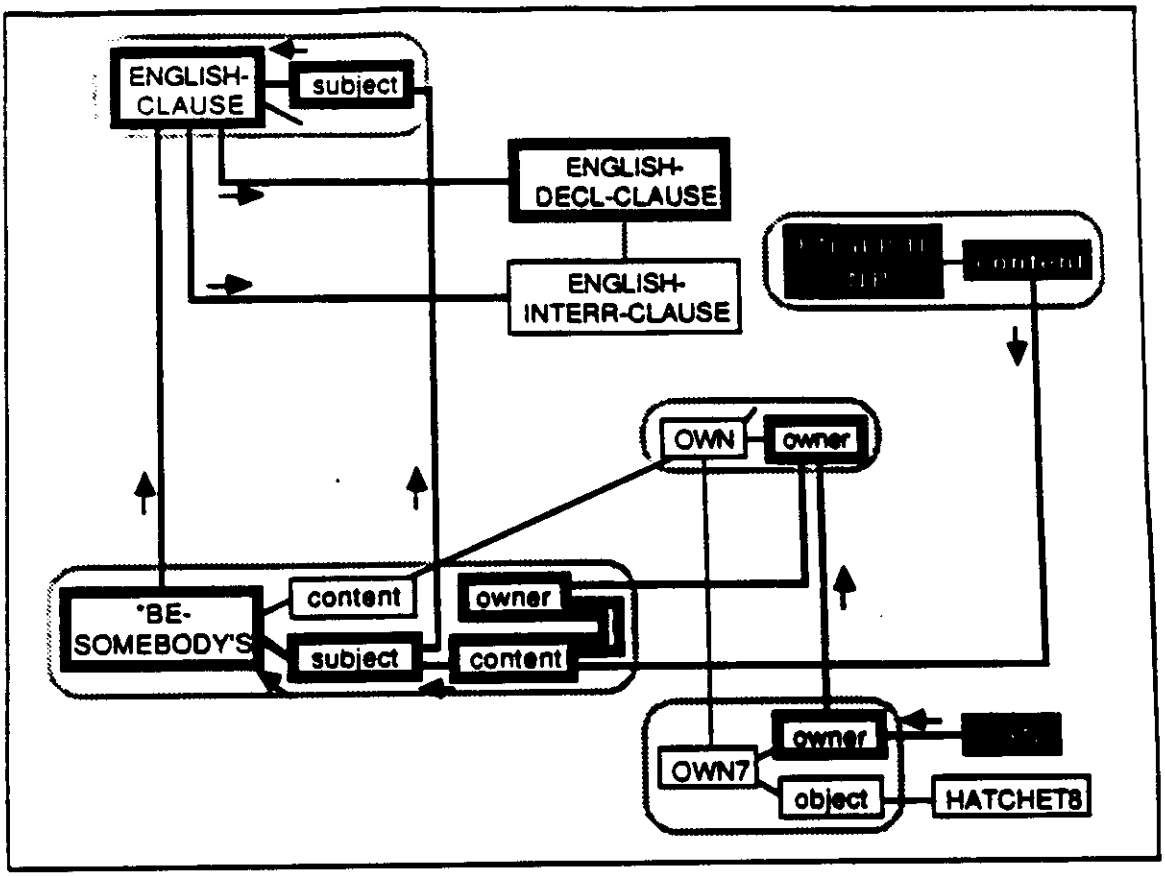

Figure 7.24: Intial Stages in the Generation of (7.8b)

With *OWN missing, the *BE-SOMEBODY'S GU is the only way to refer to the ownership of the hatchet. With the various relative clause patterns missing, the only way to realize the clause is through the available subtypes of CLAUSE. Let us assume that this speaker has knowledge of two clause patterns: declarative and interrogative clauses. The schemas for these types would be part of a WTA network that forces the selection of one or the other. For this example, there is no reason to produce an interrogative clause, so the default form, the declarative, wins out.

Assuming the speaker has not yet begun to generate the NP, the declarative clause will be uttered at this point: *the hatchet was somebody's*. Once this has happened, the generation of the reference can begin again, but now the conditions have changed. The referent is now assumed to be activated for the hearer because it has just been referred to. As a result, the reference is now appropriately realized as a pronoun. Here another gap in the speaker's knowledge of the language leads to a further error, the repetition of *somebody* in place of *he*. This gap is of the type illustrated in example (7.5). It represents a part of a linguistic paradigm from which a native speaker of this language is forced to choose, even when the selection has nothing to do with her communicative intent.

Note that the strategy used in (7.8) is not the only possibility for dealing with a gap in syntactic knowledge of this type. The main concern of the speaker in this case is the position of the relative clause (rather than any sort of morphological marking it might have), and she might simply choose to place the English relative clause in the Japanese position, that is, before the head noun. This occurs in the example in (7.9b), in which the speaker has been asked to translate the Japanese sentence in (7.9a).

(7.9a) *pittyaa wa booru o nageru sensyu desu.*
pitcher TOP ball ACC throw player is
'The pitcher is the player who throws the ball.'

(7.9b) *The pitcher is throw the ball player.*

Whether this sort of transfer would occur in a more natural context is not certain.

7.5. Summary

In this chapter I identified four types of cross-linguistic memory relationships suggested by the CLM approach, one involving only a shared concept, one a shared schema, and two direct translation connections of different types. Evidence from second language errors, code-switching data, and first language "loss" indicates that three of these types of associations are needed to account for the behavior of second language learners and bilinguals, while the direct word-to-word translation association seems to be unnecessary. I also discussed ways in which the CLM model handles the behavior of speakers whose knowledge of the L2 is deficient some way. In this regard it is important to distinguish gaps in the knowledge which the speaker needs to realize her goals from gaps in the grammatical knowledge which is built into the language but which often has nothing to do with the realization of the speaker's goals. In the former case speakers are generally aware of the deficiency and must cope with it in some way. For lexical gaps one option is to select a schema which partially matches the input. In doing so, the speaker may be guided by any available L1 schemas. The result may be the typical errors of overgeneralization. For syntactic gaps, one option is the selection of an alternate realization of a high-level pattern type, what would be a "sister" of the desired structure in the generalization hierarchy.

Chapter 8

Comprehension

The focus of this thesis is the generation of language, but an attempt has been made to develop a system which allows much of the same knowledge to be used in comprehension as well. Corresponding to the three aspects of generation in the CLM model, there are accounts of schema selection, role binding, and sequencing in comprehension. These are the topics of this chapter.

8.1. Schema Selection

Schema selection for comprehension in the model makes use of exactly the same knowledge structures and spreading activation mechanism as it does for generation. The only difference is in the nature of the input.

Consider what would be involved in lexical selection for the word *bank* in the sentence *Mary deposited $100 in the bank*, assuming the system has GUs for two senses of *bank*, 'financial institution' and 'river shore'. The selection process is illustrated in Figures 8.1 and 8.2.

The input at this point in the sentence consists of the firing of the word node "BANK". This is strongly connected to the NOUN nodes in both GUs, *BANK/RIVER and *BANK/FINANCE, and both of these nodes fire, sending activation to the head nodes for the GUs. These compete with one another via a WTA network so that only one will be selected for a given context. The situation at this point is shown in Figure 8.1.

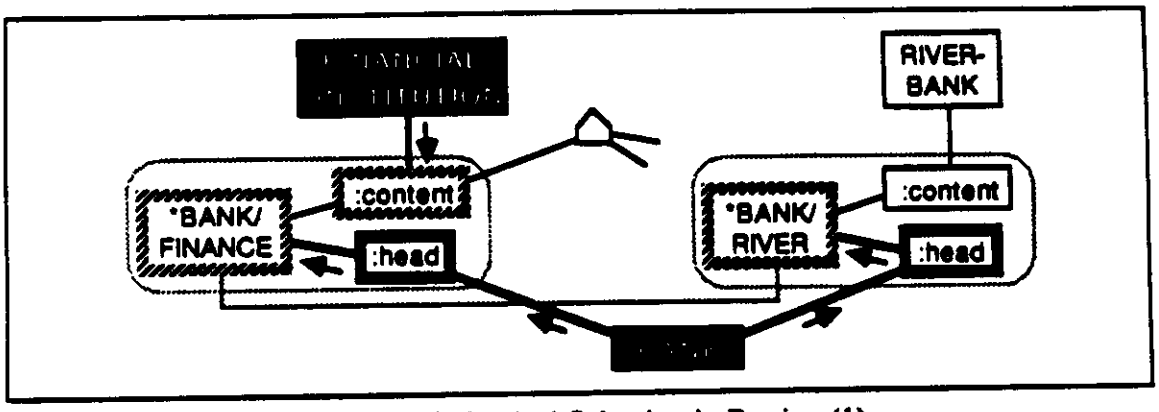

Figure 8.1: Lexical Selection in Parsing (1)

At the same time, the CONTENT role of the *BANK/FINANCE GU will have received activation on the basis of the semantics of the clause. The *DEPOSIT-IN-BANK GU specifies that the DESTINATION of the depositing appears in a prepositional phrase with *in* and that this DESTINATION is a FINANCIAL-INSTITUTION. Hence at this point we would expect the FINANCIAL-INSTITUTION node to have fired and sent activation to *BANK/FINANCE:CONTENT. The situation at this point is shown in Figure 8.1.

*BANK/FINANCE:CONTENT is a member of a WTA network, and when the WTA hub for this network times out and sends activation to all of the members, *BANK/FINANCE:CONTENT is most likely to fire. It then sends additional activation to the *BANK/FINANCE node, causing it to win over *BANK/RIVER (Figure 8.2).

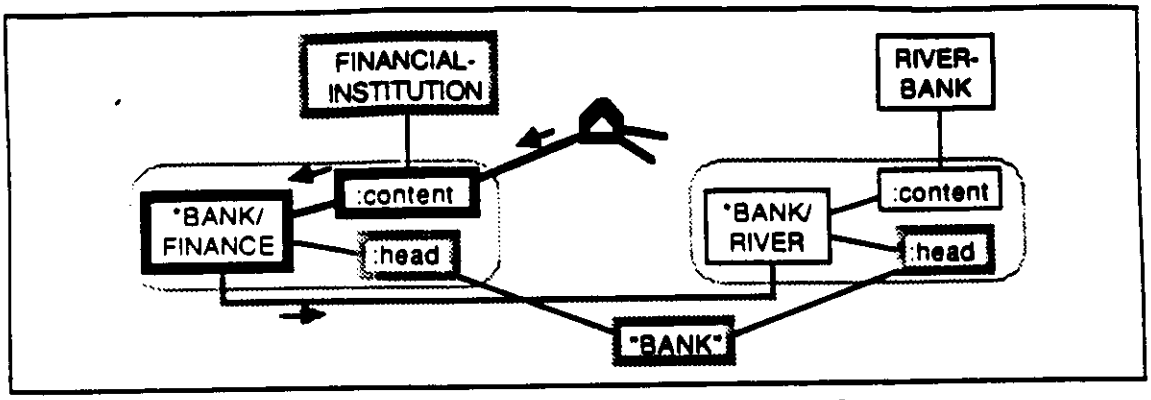

Figure 8.2: Lexical Selection in Parsing (2)

Note that whereas generation requires extensive WTA networks linking CONTENT roles, comprehension seems to need WTA networks only for homonyms such as *bank*.

8.2. Role Binding

Role binding comprehension is somewhat more complicated than in generation. Here conceptual entities are assigned to roles in other conceptual entities on the basis of the role association information in accessed schemas. The CLM approach works here as well, but it is limited in two ways. First, intersecting paths of activated nodes may prevent a set of associations from being available simultaneously. Second, there is currently no way of permanently storing the associations in long-term memory.

Consider the comprehension of the sentence *Mary loves John*. The word *Mary* leads to the firing of the nodes in the *MARY GU and the MARY node so that these nodes are primed when the word *loves* appears. Figure 8.3 shows a portion of the network at this point. The *LOVE GU includes the role association information for both the SUBJECT and DIRECT-OBJECT constituents, one merged node for each association. For simplicity I have omitted the TRANSITIVE-CLAUSE schema shown in Figures 5.1 and 5.2 and shown the SUBJECT and DIRECT-OBJECT of *LOVE connected directly to the NP schema.

Figure 8.4 shows what happens after the word *love* is recognized. The word *loves* activates the *LOVE GU, resulting in the firing of the node for the concept LOVE. Since the SUBJECT is expected to precede the verb, the SUBJECT role in the schema also fires at this point. This activates the role association information for the SUBJECT, represented by the node merging the CONTENT of the SUBJECT with the ACTOR. Activation spreads from the merged node in two directions, leading in one direction to the firing of the ACTOR node in the LOVE schema and in the other to the firing of the MARY node. It is the more or less simultaneous firing of these two nodes which implements the role binding.

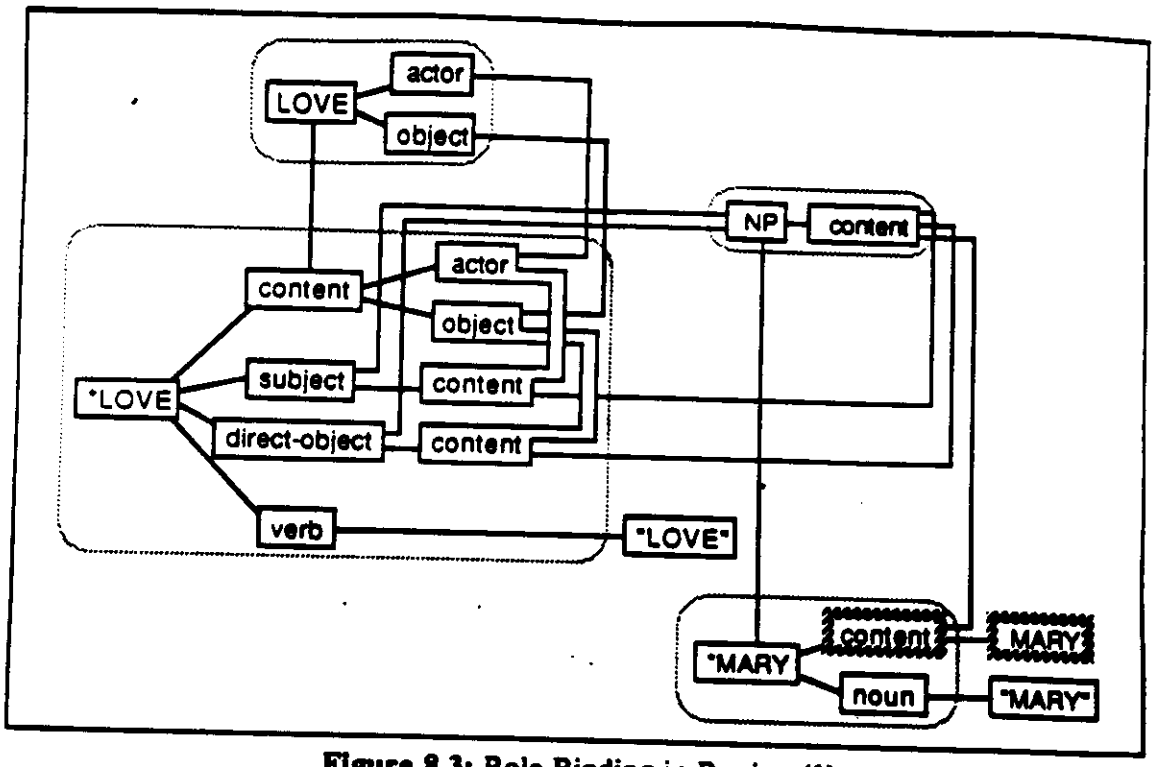

Figure 8.3: Role Binding in Parsing (1)

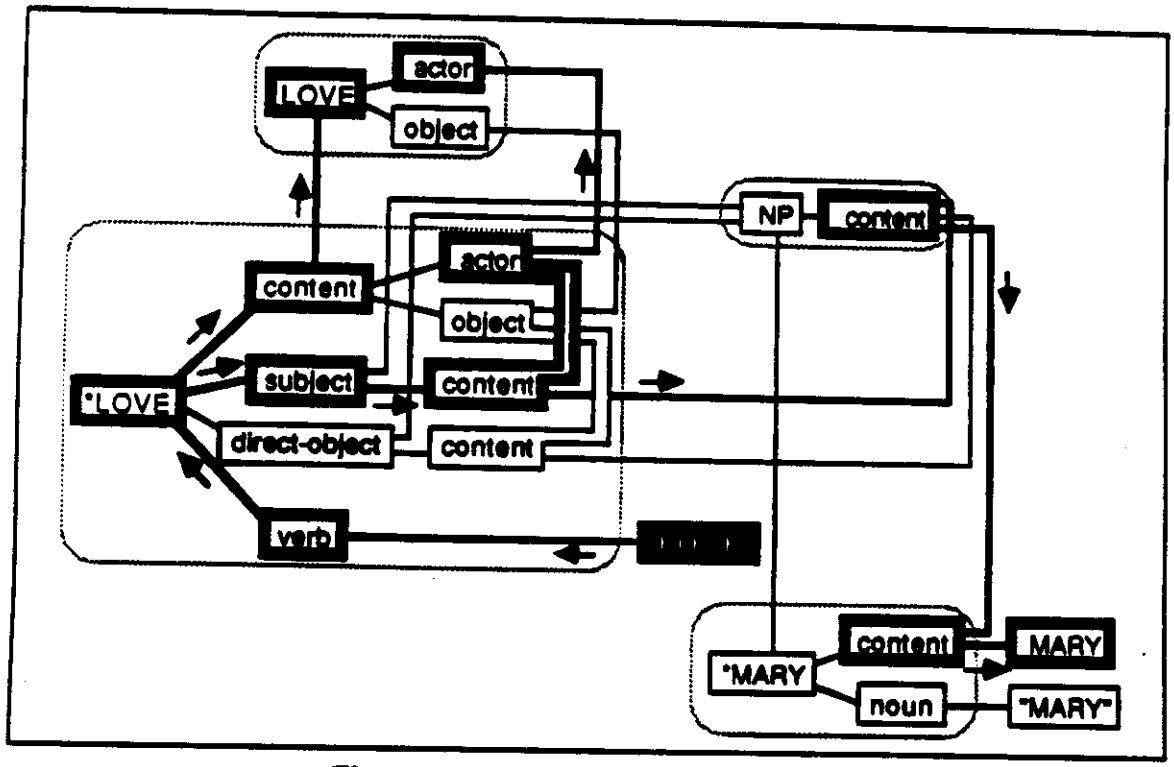

Figure 8.4: Role Binding in Parsing (2)

This binding should be accessible later because it will leave a binding path connecting MARY and LOVE:ACTOR. That is, after the refractory periods of the nodes on the path have ended and before their residual activation has disappeared, the system can access MARY by activating LOVE:ACTOR or LOVE:ACTOR by activating MARY. For this example, there is a problem with crosstalk, however. Role binding must next take place for the DIRECT-OBJECT of the sentence as well. This will correctly associate the LOVE:OBJECT and JOHN nodes, but the binding path for these two nodes includes NP:CONTENT, which was also on the path for the binding of LOVE:ACTOR with MARY. That is, the earlier binding is now inaccessible. Within the present framework there is no way around this problem. In Chapter 11, I will discuss an adaptation of the model that would enable the two bindings to co-exist.

8.3. Sequencing

In Chapter 6, I discussed the use of sequencing information in the generation of the sentence *she sent him a letter*. Consider how the same information would be used in the parsing of the sentence. Input in this case consists of the spaced firing of nodes representing the words of the sentence. The firing of "MARY" leads to the selection of the *MARY GU and the consequent firing of the MARY node. The firing of "SENT" results in the selection of the *SEND GU, which is similar to the more specific *SEND-MAIL GU described in Chapter 6. The latter schema would also receive some activation and might fire if there were expectation-based priming regarding the writing of a letter. Activation is sent immediately to the SUBJECT constituent of the *SEND (and/or *SEND-MAIL) schema, resulting eventually in the firing of the ACTOR node. It is the close proximation of the firing of MARY and ACTOR which represents role binding, though there is currently no way to record this binding permanently in the system's memory.

The firing of VERB/end in the *SEND schema leads, as in the generation of the same sentence, to the activation of nodes for both of the constituents which may follow. At this point neither of these constituents has enough activation to fire. The activation that is present represents the expectation that there will now be a reference to either the RECIPIENT or the OBJECT. The state of the network at this point is shown in Figure 8.5.

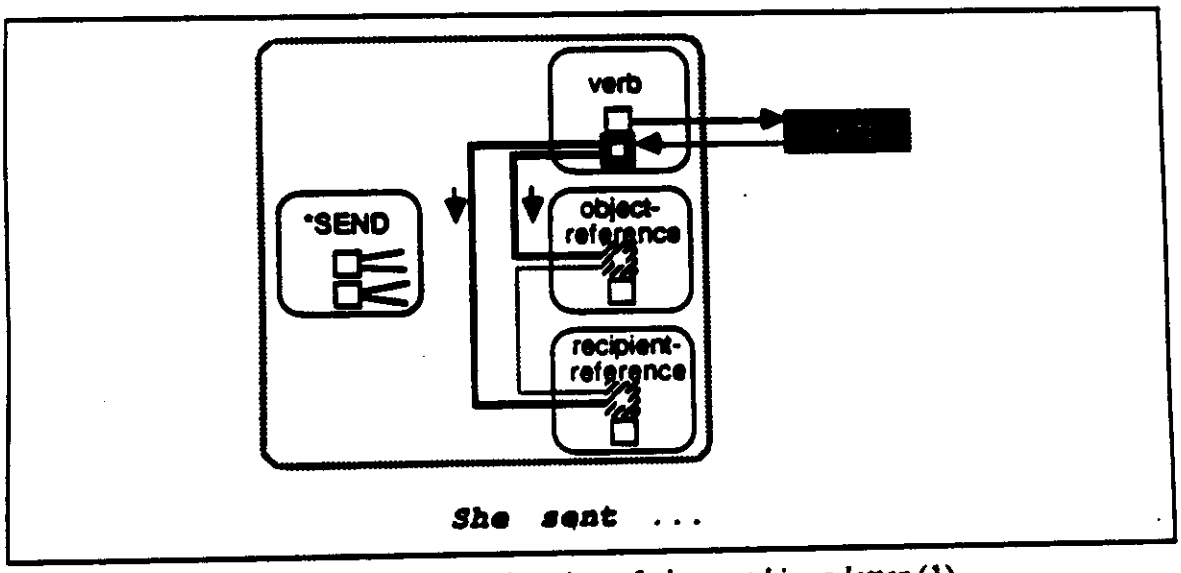

Figure 8.5: Comprehension of *she sent him a letter* (1)

Next "HIM" fires, leading to the activation of nodes representing all male humans that are currently in focus. There is only one such entity, John, and the JOHN node then fires. Activation spreads to nodes for features of John including the HUMAN node. Since humanness is a default property of the RECIPIENT of a TRANSFER-OF-CONTROL, this last node is connected to the RECIPIENT node, which can now fire, sending activation in turn eventually to RECIPIENT-REFERENCE/start in the *SEND entry. The additional activation now causes this node to fire, representing the system's recognition that the current constituent refers to the RECIPIENT rather than the OBJECT of the TRANSFER-OF-CONTROL. Figure 8.6 illustrates this sequence of events.

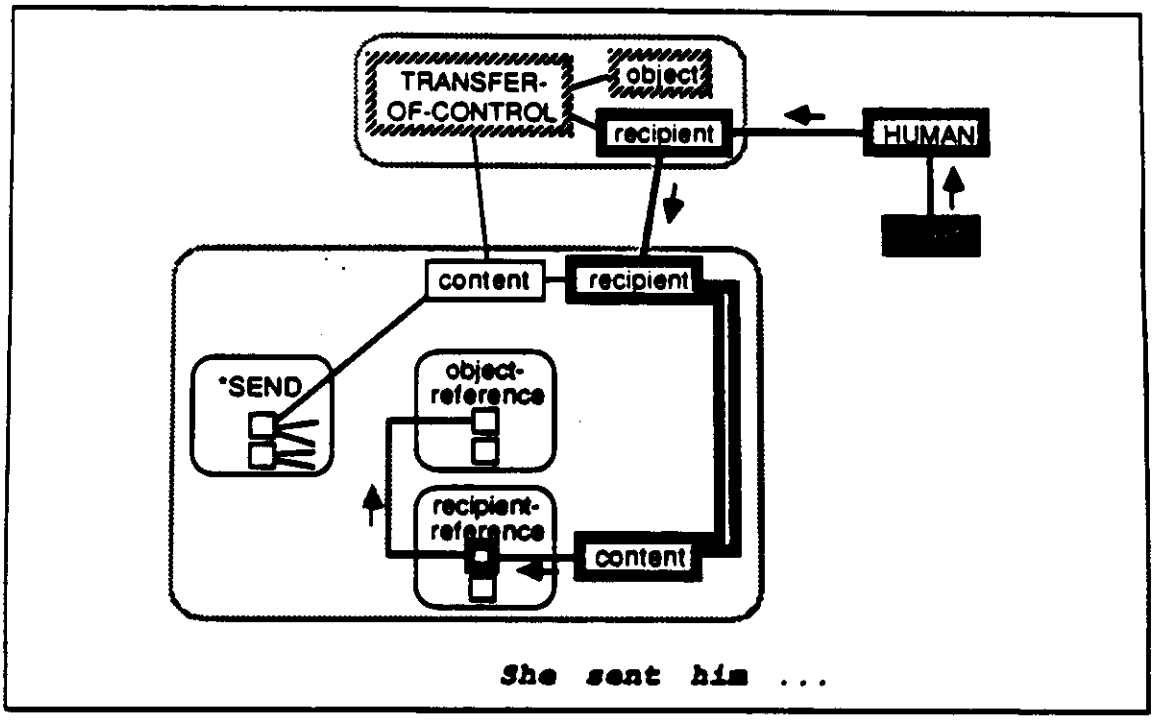

Figure 8.6: Comprehension of *she wrote him a letter* (2)

From this point on, the process, at least with respect to sequencing, is similar to what goes on during generation. After the firing of "HIM", activation spreads from RECIPIENT-REFERENCE/end to the nodes representing the two possible alternatives, the end of the clause or the appearance of the OBJECT-REFERENCE. The latter will predominate for this example once the beginning of the NP *the letter* is recognized. Following the completion of this NP, there will again be two alternatives. In this case the CLAUSE/end option will win out, as in the generation case, because of inhibition on RECIPIENT-MARKER/start.

8.4. Summary

This chapter has looked briefly at ways in which the CLM network and processing algorithm apply to comprehension as well as generation. Schemas are accessed in comprehension through words and the roles for constituents rather than CONTENT or GOAL roles, as in generation. There is competition between schemas with identical forms for particular constituents, e.g., the two BANK GUs, rather than between schemas with similar

meanings as in generation. Otherwise schema selection in comprehension proceeds exactly as in generation. In comprehension, role association information in a selected schema allows the system to bind roles in the concept that is behind the utterance. Sequencing information in comprehension comes in part from the ordering that is built into the input and in part from the same sequencing connections and WTA networks which play a part in generation.

Chapter 9

Related Work

9.1. Overview of Related Work

The CLM model is concerned primarily with explaining human language generation, but the goal has been to carry out generation using mechanisms and representational primitives that are common to other domains. Thus there is a large body of research which is related in one way or another to the present work. I will discuss it under two headings: computational models of language generation and general models of cognitive processing. Rather than surveying the work, I will treat three illustrative models from each category. The models chosen for discussion are those resembling the CLM model most in terms of goals or general approach.

9.2. Language Generation Models

Existing work on generation can be conveniently divided into two categories, that done in linguistics and psycholinguistics and that done in AI and computational linguistics. In general the linguistic and psycholinguistic models tend not to be formalized to the extent that they could be implemented as computer programs.. Dell (1986) and Kempen and Hoenkamp (1987) are partial exceptions in this regard. On the other hand, for the models in AI and computational linguistics the main concern is generally the development of a working program, and psychological plausibility takes a backseat. Again there are partial exceptions; for Conklin (Arbib, Conklin, & Hill, 1987) in particular, psychological issues are an important consideration. The present model is in the middle of the two extremes. The twin concerns here have been to produce a model which generates language in a way that is consistent with human behavior and to implement the model as a program.

In this section I examine three computational models of generation, each sharing certain important features with the CLM model.

9.2.1. Jacobs

Paul Jacobs' KING generator (Jacobs, 1985b), in many ways a descendant of his earlier PHRED system (Jacobs, 1985a), focuses on the importance of knowledge representation in generation.

The KING system makes use of a semantic network scheme called Ace. Both conceptual and linguistic knowledge are represented in terms of Ace structures. In addition to the usual is-a and has-a relationships, Ace includes VIEW and REF relations. VIEWs are designed to add representational flexibility. For example, GIVE and TAKE are related through separate VIEW constructs to the more general TRANSFER-EVENT. REF links associate concepts with linguistic forms. For example, the lexical category LEX_SELL is linked to the concept SELL.

Jacobs' work, in particular that on the earlier PHRED system, is within the Phrasal Lexicon approach (see also Wilensky & Arens, 1980; Zernik & Dyer, forthcoming). The basic units of linguistic knowledge are pattern-concept associations. That is, linguistic knowledge is oriented around phrases rather than single words, and the patterns are linked directly with meanings.

The generation process in KING consists of three phases. First, input concepts are mapped onto alternative concepts until one or more are found which have associated with

them the linguistic forms needed to generate a sentence. The mapping process may traverse is-a links, but more importantly it may also make use of VIEW links. Thus an input represented as an instance of TRANSFER-EVENT would first be transformed to an instance of GIVE or TAKE because there is no linguistic form associated with TRANSFER-EVENT. Next, mapping continues with the traversal of REF links, yielding a set of linguistic relations and constraints. For example, for the sentence *John gave Mary a hug*, the set includes: LEX_GIVE, VOICE_ACTIVE, VERB-INDIR-RELATION (IOBJ MARY1), and VERB-OBJ-RELATION (OBJ HUGGING1). Note that at this point much of the linguistic work of the generator has already been done.

The next phase is **pattern selection**. The relations and constraints resulting from mapping could be potentially realized in more than one way, and pattern selection chooses a syntactic pattern which best satisfies them. Patterns, like the rest of knowledge in the system, are represented in an is-a hierarchy, which is searched during this phase. For the sentence *John gave Mary a hug*, the DATIVE-VP pattern is selected. The final phase is **restriction**, which takes the output of both mapping and pattern selection and fills in the elements of the patterns.

Like Jacobs's KING system, the CLM model emphasizes commonalities between linguistic and conceptual knowledge, direct links between linguistic and conceptual units, and alternative representations of basic conceptual notions. However, whereas Jacobs makes use of specific VIEW and REF links, these relationships are handled in the CLM model using the same simple connections that implement is-a and has-a relations. Thus GIVE and TAKE are joined by excitatory connections to TRANSFER-OF-CONTROL and to each other by inhbitory connections. This arrangement results in the selection of at most one of these concepts when TRANSFER-OF-CONTROL is activated. Alternative views, then, are accessed not through a special mapping procedure but through the spread of activation. The same is true for Jacobs' REF links. In the CLM model concepts are associated with patterns through the GOAL roles of GIs and the CONTENT roles of GUs, but these nodes are treated like any others by the spreading activation mechanism.

The CLM model also shares with Jacobs' work and other work within the Phrasal Lexicon tradition the representation of linguistic units as phrases rather than isolated words and the organization of these units in is-a hierarchies. However, in the CLM model the patterns are associated not only with concepts (their CONTENT) but also with the deictic elements of an utterance, the speaker, hearer, time, and context.[22] I have shown how the availability of these roles allows the direct representation of the meanings of words such as *me* and *come*. In addition, the CLM model expands the hierarchy of units to include phrases at the level of illocutionary acts. This enables the generation system to take into account the goals of the speaker, as is done in systems that handle speech acts (e.g., Appelt, 1985; Perrault & Allen, 1980).

The CLM approach differs from Jacobs' systems, and from other AI models of language generation, most significantly in the use of spreading activation to achieve all aspects of the process. The mapping of concepts to alternative views, the selection of patterns, the binding of roles such as DIRECT-OBJECT:CONTENT to particular input entities, and the sequencing of pattern constituents, which require special mechanisms in a symbolic model such as Jacobs', are all side effects of the spread of activation from input nodes in the CLM model. Furthermore, no instantiations of intermediate concepts or patterns are made during processing.

[22] Zernik (1988) also includes a context in his phrasal units.

9.2.2. Dell

Gary Dell's work on language generation (Dell & Reich, 1980, 1981; Dell, 1985; Dell, 1986) has focused on modeling human speech errors, particularly those at the morphological and phonological levels. He has carried out several computer simulations of his theory.

Dell makes two fundamental distinctions in his model. First, as in nearly all linguistic models, he distinguishes between levels of linguistic knowledge: semantic, syntactic, morphological, and phonological. Dell posits a representation at each of these levels for the generation of a given utterance. Second, he distinguishes between the "lexicon", and "generative rules". The lexicon is composed of a network associating elements from the different levels. Thus it includes elements which would not be considered lexical in most models, for example, nodes representing phonemes. Each level has its own set of generative rules defining possible combinations of elements at that level. The rules apparently look a good deal like phrase structure rewrite rules, except that they operate at levels other than syntax as well. For example, at the phonological level there are rules defining the possible combinations of phonemes in the syllable.

During generation, the same basic process takes place at each level. Using the rules for that level, a "frame" is generated which is consistent with information from the next higher level. (How the rules operate for the semantic and syntactic levels is not clearly spelled out.) Next a lexical selection process accesses a set of items from the lexicon to fill the slots in the frame. For example, at the syntactic level, words are selected, and at the phonological level, phonemes or phonological features are selected. Each of the nodes in the lexicon is marked for its class so that the selection process will be able to choose items which fit in the slots in the frame. Thus a word node with the feature NOUN would be a possible filler for a NOUN slot in a syntactic frame. There is a separate set of "insertion rules" responsible for associating lexical nodes with slots in frames.

The lexical selection process operates through spreading activation. Activation spreads from lexical nodes when they are selected at a particular level to lexical nodes belonging to the next lower level. For example, the morphological node "SWIM" activates the phonological nodes /SW-onset/, /I-nucleus/, and /M-coda/, representing the segments making up the morpheme. When the phonological frame for the morpheme *swim* is being filled, these three nodes will normally exceed other phonological nodes in activation and will be selected for the three slots in the frame.

The representation that is built at each level is not actually an instantiation of a frame. Rather it consists of a set of lexical nodes that are tagged for their positions in the frame. For example, in the generation of the sentence *some swimmers sink*, the word nodes SOME, SWIMMER, PLURAL, and SINK would have the tags 1, 2, 3, and 4 respectively to indicate their positions in the current syntactic frame. There is also a tag to indicate the current node at each level that is being expanded into a frame at the next level.

An important feature of the model is feedback between the levels, which plays a role in "output bias" in errors. Output bias is the tendency for phonological errors to result in genuine words and for these words to be related semantically to the content of the sentence. For example, in the production of the error *rude lip* for *lewd rip*, the morpheme nodes "LEWD" and "RIP" activate the set of phonological nodes for these words, and these nodes in turn send some activation back to the morphological level. In this case the result is the activation of the morpheme nodes "RUDE" and "LIP". The model predicts correctly that errors such as this one are more common than those which result in non-words such as *ruke lisk* for *Luke risk*.

Dell's model bears a number of similarities to the CLM model. In particular, both models make use of spreading activation over a network of nodes representing linguistic constructs. Both are what Dell calls type-only models; rather than instantiate patterns, the models use the patterns themselves to represent particular pattern tokens. While Dell does not make use of inhibitory connections, he does use WTA networks to select from among candidates to fill a frame slot. For Dell's syntactic level, this is analogous to the selection of lexical GUs in the CLM model.

There are important differences, however. Most significantly, Dell's model requires two sets of rules and two types of markers in addition to the spreading activation mechanism. In the CLM model, patterns are selected, pattern slots are "filled" (through the role binding process described in Chapter 5), and items are sequenced appropriately using the same spreading activation mechanism. Thus the CLM model has the advantage of greater simplicity.

The CLM model does not make the distinction between generative rules and static lexical knowledge which is basic to Dell's model. Instead, GUs, which correspond to lexical entries in other models, incorporate syntactic information because they are phrasal units. Thus the selection of a GU entails not only the selection of one or more morphemes but also of a syntactic pattern. Patterns also appear as the PLAN roles of GIs which are selected on the basis of the speaker's goals.

The two models also differ in terms of emphasis. While the CLM model focuses on illocutionary acts and lexical selection, Dell is more concerned with morphology and phonology, and he has nothing at all to say about pragmatics. Dell is interested mainly in accounting for speech errors, while the present model handles other phenomena such as robustness, flexibility, and cross-linguistic transfer, in addition to some types of speech errors. In future work it will need to be demonstrated that the model can handle the range of errors that Dell's model can. With the exception of certain perseveratory phonological errors (see Section 11.1.1 for a discussion of a possible account within a modified CLM model), there is no reason to believe that the present model will fail in this regard. In particular, output bias effects are accommodated because activation flows in both directions across linguistic levels.

A final difference concerns the fact that the CLM network and processing mechanism are usable for comprehension as well as generation, an issue which Dell does not attempt to address.

9.2.3. Kalita and Shastri

Jugal Kalita and Lokendra Shastri (1987) have developed a localized connectionist approach to language generation which is apparently the only example of such a model other than the present model. This research appears to be at a relatively early stage of development, and not all of the details are clear from the single published paper on it.

Kalita and Shastri's network has five levels, each corresponding to a separate aspect of processing. At the input level are concept nodes representing notions such as EAT and BANANA1 and constraint-specification nodes representing syntactic features such as ACTIVE-VOICE. There is no conceptual memory as such, however; the concept nodes are not connected to each other. Input level nodes connect to nodes at the realization class level representing clauses and NPs, and these are connected to nodes at the choice level representing more specific structures such as passive clauses and indefinite NPs. Choice level nodes activate nodes at the constituent level representing constituents such as subject and determiner, and these are in turn connected to nodes at the morphological level representing morphemes.

As in other work within the localist connectionist framework defined by Feldman and Ballard (1982), nodes exhibit relatively complex behaviors. Connections join with nodes at "input sites", each of which has an associated site function. For example, one type of site computes the maximum of the inputs coming in at that site. Nodes in the network have several input sites. Nodes are either active or in one of two types of inactive states. Nodes are in the "initial-inactive" state when processing begins. A node becomes active as a result of input activation on the appropriate connections, and it remains so until its "expansion-completed" site is activated. At this point it becomes "final-inactive". Most final-inactive nodes cannot be re-activated until the network is initialized for the generation of a new sentence. The nodes in the NP network, however, have a special "reuse site". When this is activated, these nodes change from "final-inactive" to "initial-inactive" and can be used for the generation of further NPs in the same sentence.

As discussed in previous chapters, a connectionist model of language generation must concern itself with three aspects of the problem, the selection of linguistic units, the binding of syntactic and semantic roles, and the sequencing of items being output. With respect to the selection of units, Kalita and Shastri's model has little to offer. Since there is no real conceptual memory, lexical selection involves a direct mapping from input concepts to words. This deficiency in the model is recognized by the researchers. Selections of syntactic patterns are made, but this is done in part on the basis of rather implausible syntactic input constraints such as the stipulation that the current clause be active.

A good deal of the special machinery designed by Kalita and Shastri is involved in the work of role binding. The mechanism is quite complicated and requires at least four special node types in addition to the nodes actually representing the semantic and syntactic roles themselves. Each combination of a possible input object and a semantic role has its own special binder node (e.g., OBJECT-BANANA1). Each semantic role also has a "global enable" node with connections to the set of binder nodes for that role. When processing starts, certain of these connections are turned on to specify input semantic role assignments. In addition, there is a "driver" node for each syntactic role, an "NP expansion driver" node, and a "binding completion" node which finally transmit activation to the NP network and initiate lexical selection for the constituent.

As discussed in Chapter 5, sequencing in a connectionist model involves two separate problems, a way to represent sequential relationships and a way to have constituents signal others when they are complete. For the first problem, Kalita and Shastri make use of special sequencing nodes. The operation of these nodes is not made completely clear in the paper, but they function not only to order output words but also to deactivate units which have already been used. The second problem is handled by the special "expansion-completed" sites on nodes which were referred to above. Once a constituent is complete, the node for that constituent is activated at its expansion-completed site, the node becomes final-inactive, and a sequencing node is activated to start off the generation of the next constituent. In the case of NPs, the sequencing node also turns on the NP nodes again by activating their reuse sites.

There are two reasons for wanting to implement a particular cognitive task using a connectionist approach. One is to demonstrate that it is possible to accomplish what is accomplished in symbolic models within the highly constrained framework of connectionism. Alternately, or in addition, one may wish to demonstrate that the connectionist account has advantages over the symbolic account for the task being modeled. While both goals are behind the CLM model, the work of Kalita and Shastri seems only to be based on the first. Their model does not exhibit any interesting features which would not be exhibited by symbolic generation systems. It simply shows that, for the range of structures they consider, it is possible to generate sentences by propagating activation through a network of simple processing units.

As in other work within the localist connectionist framework defined by Feldman and Ballard (1982), nodes exhibit relatively complex behaviors. Connections join with nodes at "input sites", each of which has an associated site function. For example, one type of site computes the maximum of the inputs coming in at that site. Nodes in the network have several input sites. Nodes are either active or in one of two types of inactive states. Nodes are in the "initial-inactive" state when processing begins. A node becomes active as a result of input activation on the appropriate connections, and it remains so until its "expansion-completed" site is activated. At this point it becomes "final-inactive". Most final-inactive nodes cannot be re-activated until the network is initialized for the generation of a new sentence. The nodes in the NP network, however, have a special "reuse site". When this is activated, these nodes change from "final-inactive" to "initial-inactive" and can be used for the generation of further NPs in the same sentence.

As discussed in previous chapters, a connectionist model of language generation must concern itself with three aspects of the problem, the selection of linguistic units, the binding of syntactic and semantic roles, and the sequencing of items being output. With respect to the selection of units, Kalita and Shastri's model has little to offer. Since there is no real conceptual memory, lexical selection involves a direct mapping from input concepts to words. This deficiency in the model is recognized by the researchers. Selections of syntactic patterns are made, but this is done in part on the basis of rather implausible syntactic input constraints such as the stipulation that the current clause be active.

A good deal of the special machinery designed by Kalita and Shastri is involved in the work of role binding. The mechanism is quite complicated and requires at least four special node types in addition to the nodes actually representing the semantic and syntactic roles themselves. Each combination of a possible input object and a semantic role has its own special binder node (e.g., OBJECT-BANANA1). Each semantic role also has a "global enable" node with connections to the set of binder nodes for that role. When processing starts, certain of these connections are turned on to specify input semantic role assignments. In addition, there is a "driver" node for each syntactic role, an "NP expansion driver" node, and a "binding completion" node which finally transmit activation to the NP network and initiate lexical selection for the constituent.

As discussed in Chapter 5, sequencing in a connectionist model involves two separate problems, a way to represent sequential relationships and a way to have constituents signal others when they are complete. For the first problem, Kalita and Shastri make use of special sequencing nodes. The operation of these nodes is not made completely clear in the paper, but they function not only to order output words but also to deactivate units which have already been used. The second problem is handled by the special "expansion-completed" sites on nodes which were referred to above. Once a constituent is complete, the node for that constituent is activated at its expansion-completed site, the node becomes final-inactive, and a sequencing node is activated to start off the generation of the next constituent. In the case of NPs, the sequencing node also turns on the NP nodes again by activating their reuse sites.

There are two reasons for wanting to implement a particular cognitive task using a connectionist approach. One is to demonstrate that it is possible to accomplish what is accomplished in symbolic models within the highly constrained framework of connectionism. Alternately, or in addition, one may wish to demonstrate that the connectionist account has advantages over the symbolic account for the task being modeled. While both goals are behind the CLM model, the work of Kalita and Shastri seems only to be based on the first. Their model does not exhibit any interesting features which would not be exhibited by symbolic generation systems. It simply shows that, for the range of structures they consider, it is possible to generate sentences by propagating activation through a network of simple processing units.

The most serious deficiency of the model is its failure to deal with the issue of unit selection. This is an aspect of generation involving the satisfaction of multiple constraints, and hence, as I have argued, connectionist models are well suited to the task. Thus it is somewhat surprising that Kalita and Shastri have chosen to ignore the selection issue.

Kalita and Shastri have developed a novel approach to the role binding problem, but it is much more complex than the mechanism I described in Chapter 5. The CLM approach depends crucially on the fact that node activation decays over time and that recently activated nodes are more readily reactivated than those activated earlier. Kalita and Shastri's model does not have a decay mechanism, however, and they are forced to resort to their complicated network of binding nodes. The most serious problem with this approach, as they themselves note, is the use of separate binding nodes for each concept-role combination. This greatly increases the memory requirements of the system.

On the other hand, there are some similarities in the way in which sequencing is handled in the two models. The expansion-completed sites on Kalita and Shastri's nodes perform the same basic functions as the end nodes of the CLM model. Both signal that a constituent has been completed so that the next constituent can begin and at the same time prevent the overlap of constituents. One advantage to the CLM approach is that start and end nodes can have separate output connections to other nodes. For example, some constituent end nodes activate the end nodes for the phrases which they belong to. That is, the completion of a constituent may provide evidence that the larger phrase has been completed or is near completion. The division into start and end nodes also makes special sequencing nodes unnecessary; the start and end nodes do the sequencing themselves through sequencing connections joining end nodes to start nodes of different constituents. A further weakness of Kalita and Shastri's model, which the authors recognize, is the inability to handle optional constituents. In the CLM these are simply constituents which fire only when they receive activation from semantic sources.

A final drawback to Kalita and Shastri's model is the need to reinitialize the network each time a sentence is to be generated. All nodes must be returned to the initial-inactive state, and connections representing role assignments must be enabled. The need for separate initial-inactive and final-inactive states in the model is not made clear in the paper. A type-only model such as this should allow nodes of all types, not just those used in generating NPs, to be reused in the generation of a single sentence.

9.3. General Cognitive Models

In recent years there have been several attempts to handle a wide range of cognitive phenomena within the framework of a single theory. As in the CLM model, these theories emphasize commonalities between language processing and other types of cognitive tasks such as the solving of geometry problems. In this section I discuss three models which share other aspects of the CLM model as well: all make extensive use of spreading activation over a network memory.

9.3.1. Anderson

John Anderson's ACT* model of human cognitive processing (Anderson, 1983; 1985) is a computational theory developed over more than a decade. The model accounts for an impressive array of data from recognition, recall, planning, and learning, and Anderson's research includes many computer simulations of the key aspects of his model. ACT* is certainly unparalleled in the psychological literature, and this is no place to attempt a comprehensive review.

Anderson makes a fundamental distinction between **declarative** and **procedural knowledge**, and I will focus on this claim in the discussion here. Declarative knowledge is knowledge about facts and things. It can be acquired through observation or by being told. Declarative knowledge is represented using three different "codes": abstract propositions, spatial images, and temporal strings.

Propositions, comprising the semantic knowledge that is involved in interpreting and generating language, are embodied in a network of nodes with associated strengths. Facts in the network are accessed through the spread of activation starting with a set of source nodes. The degree of spread is governed by node strengths. There is a decay mechanism to keep activation in check.

Procedural knowledge is knowledge about how tasks are performed. Procedural knowledge about driving, for example, would be what is used during the act of driving, whereas declarative knowledge about driving would be what is necessary to explain how to drive to someone else. Note that the one does not presuppose the other; it is possible to drive perfectly well without being able to describe what one is doing.

Procedural knowledge takes the form of **productions**, which are rules each composed of a condition and an action component. The condition specifies a pattern of data, often including a type of goal, that must be available in the system's working memory for the production to apply, and the action specifies the procedure that is executed when the production applies. An action may add elements to working memory or carry out some sort of motor activity. For example, the following is a production that would be used in generating a sentence (Anderson, 1983, p. 262):

```
Condition   the goal is to describe RELATION
            and RELATION describes ongoing ACTION
            and ACTION is currently happening

Action      set as subgoals
            1. to generate is
            2. to generate the name for ACTION
            3. to generate ing
```

When the speaker wants to describe a relation which describes an ongoing action, this production would "fire", resulting in a goal to say *is* and the verb representing the action together with an *ing* suffix.

Productions make use of facts in declarative memory. For example, one of the actions taken by the above production is to access a name for the concept being referred to. This retrieval is effected through the spread of activation from the node representing the concept. By itself declarative knowledge is useless. It is only through the retrieval processes (that are part of productions) that it is put to use.

Procedural knowledge is acquired differently from declarative knowledge. Declarative knowledge is learned on an all-or-none basis. For example, on being told that more men are bald than women, the system might simply add this fact to its database. Productions, on the other hand, are acquired over time. With practice, the system is better able to match the conditions in productions, and the procedures become more and more automatic.

A further difference between declarative and procedural knowledge concerns directionality. Procedural knowledge is asymmetric; it operates from conditions to actions, but not vice versa. A declarative fact, on the other hand, can be accessed from any of its component elements.

Like Anderson's ACT* model, the CLM model makes use of a semantic network and a spreading activation mechanism for memory access. The major difference lies in the fact that the CLM model has no separate procedural component; the functions of Anderson's productions are carried out by pieces of the network and spreading activation.

Collapsing the two components simplifies the system and at the same time makes it easier to conceive of its implementation in the hardware of the brain. It is a trivial matter to encode rules in the form of a schema: the condition and action would correspond to roles in a rule schema. But is such a schema usable in a way that simulates human procedural knowledge? One problem is that schemas are normally accessible from all of their roles. That is, there is no specified direction to a schema; a rule could proceed from action to condition just as well as from condition to action. A direction can be enforced, however, if there are weights on the role-to-head connections. If the weights on connections from condition roles were higher than those from action roles, the rule schema would normally be accessed more readily via its condition than its action.

At the same time, the lack of directionality of schemas is an advantage, even for certain types of procedural knowledge. Consider language generation, an example of a skill which Anderson would like to represent in terms of productions. The ability to generate language is acquired mainly through the observation of others. In this process the learner makes generalizations about what sorts of intents are behind the linguistic actions of speakers. In other words, the learner creates schemas that can account for the behavior of other speakers but which can also be used by the learner herself in generating language. These schemas resemble productions in that they associate particular goals (conditions) with particular actions, but they differ in two significant ways. First, they are usable in either direction. Given a particular type of linguistic act, a hearer recognizes it as the action component of a schema and infers the goal behind the act on that basis. Given a particular goal, a speaker accesses the same schema and executes the action specified in it to realize the goal. Secondly, these linguistic schemas represent knowledge about how speakers behave, not only how the learner herself behaves. This is clear from the fact that people have knowledge about the speech patterns of groups to which they do not belong, knowledge which they are quite capable of using when they are quoting or mimicking speakers who are members of those groups. Thus Japanese women have no difficulty in using the personal pronouns which are characteristic of Japanese men's language when they are imitating or quoting men. At least for linguistic knowledge, schemas seem to do a better job than rules.

It appears that Anderson may be making an important distinction, but making it in the wrong place. There are two ways to learn a general fact, such as that men are more often bald than women. One is to be told, in which case the fact either is or is not added to memory. The other is to build up a generalization about human baldness over time by observing a number of men and women. This is the type of knowledge that would be encoded in a schema. There is no reason to believe that the former type of knowledge would have the same form as the latter. The generalization alternative would correspond closely to Anderson's procedural knowledge, and like procedural knowledge it might be relatively inaccessible to consciousness. Yet it does not represent knowledge about doing in the usual sense. The distinction Anderson makes is an important one, but it does not argue against the use of schemas to represent rule-like knowledge.

9.3.2. MacKay

Donald MacKay's Node Structure theory of human action and perception (1987) has its origins in MacKay's work on language generation, but the theory in its present form has been extended to encompass not only language perception, but non-linguistic action and perception as well.

MacKay posits a network memory containing three types of nodes. "Content nodes" represent concepts and linguistic constructs, "sequence nodes" control the serial output of actions, and "timing nodes" handle variations in overall tempo. A spreading activation mechanism operates on the network memory. Nodes must be primed from two sources before firing (which MacKay calls "activation") can take place. After firing, nodes are initially inhibited and later recover some residual activation. WTA networks ("domains" operating under the "most-primed-wins principle") implement competition between nodes.

Consider how the model generates the subject of the sentence *theoretical predictions guide research*. A content node representing the meaning of the entire sentence fires, and this activates a node representing the meaning of the subject NP. The node for the NP then activates lexical content nodes for the adjective *theoretical* and the noun *predictions*. These nodes do not fire immediately but pass some activation along to the sequence nodes for ADJECTIVE and NOUN, which are members of the domain (WTA network) of sentential sequence nodes. ADJECTIVE inhibits NOUN. Next additional activation is sent to the sequence nodes from firing timing nodes for the sentential domain, but only the most primed wins, in this case, ADJECTIVE. ADJECTIVE then sends activation back to the lexical node for *theoretical*, which can now fire. Activation on ADJECTIVE is reduced, releasing inhibition on the NOUN node. Thus when the timing node fires again, NOUN will be the most primed sequence node in the sentential domain. The process is repeated at the phonological level, initiated in this case through activation from the firing lexical node for *theoretical*. This level has its own content nodes representing syllables, phonemes, and features; its own sequence nodes; and its own timing nodes.

Node Structure Theory provides an account of a wide range of phenomena from language generation and perception as well as from non-linguistic skills such as typing. However, MacKay has not run simulations of his theory, presumably because it is not yet formalized to the point where this is possible.

Similarities between Node Structure Theory and the CLM model include the use of spreading activation, the use of the same network and processing mechanisms for generation and comprehension, and the refractory period and residual activation following the firing of nodes.

Sequencing in Node Structure Theory also bears similarities to the mechanism used in the CLM model. MacKay's sequence nodes serve some of the function of the start nodes for syntactic roles such as DETERMINER, SUBJECT, and NOUN. For example, in the generation of the NP *theoretical predictions* in the CLM approach, the NOUN role in the GU for the noun *prediction* is primed when this GU is selected, and when it comes time for the head noun to be uttered, this node receives additional activation from the generic NOUN node in the NP schema, much as in MacKay's account. It is not clear, however, what does the work of the end nodes of the CLM model. How would MacKay's system know, for example, when the subject NP in the sentence *theoretical predictions guide research* was complete so that the verb could begin to be uttered? It seems that in MacKay's model there is the possibility of overlap between the constituents. A further problem with MacKay's approach to sequencing is that there seems to be no way for sequencing to depend on semantic or pragmatic factors, as it can in the CLM model.

Detailed timings are not treated in the current CLM model. One straightforward way to add timing, however, would be to require nodes governing sequencing to have activation from one more source before their thresholds are reached and to have separate timing nodes, as MacKay does, to provide this additional source of activation.

Nodes in MacKay's network fire on the basis of how many sources of activation they have received rather than the sum of the input activation. That is, the connections

joining nodes have no numerical weights associated with them. Thus it is impossible to represent differences in degrees of associativity between concepts.

MacKay's model lacks an explanation of the higher-level aspects of generation, for example, how the phrase *theoretical predictions* is selected in the first place. In general, MacKay has little to say about the semantic and pragmatic aspects of processing, which are a major focus of the CLM model.

A further deficiency in Node Structure Theory, shared by Dell's generation model and many connectionist models, is the lack of anything representing roles. Thus MacKay requires separate nodes for /M-onset/ and /M-coda/ to represent the phoneme /m/ in syllable-initial and syllable-final position. A model with roles would have ONSET and CODA role nodes which could take phoneme nodes such as /M/ as their values. This lack is more serious when we attempt to use the same approach for representing concepts, something MacKay does not discuss. For example, he would apparently need a separate JOHN-ACTOR node to represent the fact that JOHN is the ACTOR of some event because there is no ACTOR role which can take on different values. Such an approach may make sense in distributed models, but in a localized network such as MacKay's it will obviously result in an extremely inefficient use of memory.

9.3.3. Shastri

Lokendra Shastri (1987) has developed a localized connectionist implementation of a semantic network which can be used for two basic kinds of inferencing: **recognition**, the assignment of an input concept to a particular type on the basis of its features, and **inheritance**, the determination of the value for a feature of a concept on the basis of the concept's types. Shastri argues convincingly that these two basic processes are at the heart of all intelligent behavior.

Like other connectionist networks, Shastri's network consists of nodes joined by weighted connections. Connection weights reflect relative frequency. For example, the weight on the connection from REPUBLICAN to PERSON is the just the proportion of people who are Republicans. As in the model of Kalita and Shastri described above, nodes have input sites with their own input functions. Nodes are either "inert" or "active". Inert nodes produce no output, while active units transmit an activation level ("potential") computed by an output function which varies from one node type to another.

The network embodies is-a hierarchies, head-role relations, and role-value relations using several node types. A general role such as TASTE (actually HAS-TASTE in Shastri's system) is represented by a single "property node", while lower-level instantiations of such a concept, e.g., HAM:TASTE and PEA:TASTE, are represented by pairs of "binder nodes", one each for use in recognition and inheritance.

Shastri presents the network with inheritance or recognition problems. An example of a simple inheritance problem is one requiring the network to decide whether the value of the TASTE role of some food is SALTY or SWEET, given the fact that the food is an instance of HAM. In this case activation spreads from the nodes representing the food, TASTE, SALTY, and SWEET, and the result is different levels of output from the SALTY and SWEET. These reflect the degree of evidence that the system has for the two possible responses. An example of a recognition problem is one asking the network to determine whether some food is HAM, PEA, or something else on the basis of the fact that its TASTE is SALTY and its COLOR PINK. Here activation spreads from the nodes representing the food, TASTE, SALTY, COLOR, PINK, HAM, and PEA, and the result is output from the HAM and PEA nodes reflecting confidence in assigning the food to the one or the other category.

Like Shastri's model, the CLM model is designed to perform both recognition and inheritance. Recognition is what is involved in selecting a GI or GU on the basis of input features. Inheritance is what is involved in accessing word forms, role binding information, and sequencing information from a schema or its types once the schema has been selected. The result in both cases is a decision represented by the firing of a node. No information regarding relative confidence in two or more alternatives is provided as in Shastri's model. At the same time, Shastri's approach is considerably more complicated because it makes use of node sites with their own input functions and a variety of output functions.

Although the CLM model appears to be unable to solve some of the complex problems which Shastri handles, at least some of Shastri's problems are accommodated by the model. One type involves multiple inheritance, in which a single concept inherits properties from two or more parent concepts. Consider the following problem:

> Given that DICK is a REPUBLICAN and a QUAKER and that known proportions of REPUBLICANs and QUAKERs are PACIFISTs (i.e., have for their BELIEF roles the value PACIFIST), is DICK more likely to be a PACIFIST or a NON-PACIFIST?

Figure 9.1 shows the network used to represent the necessary knowledge and the paths of activation.

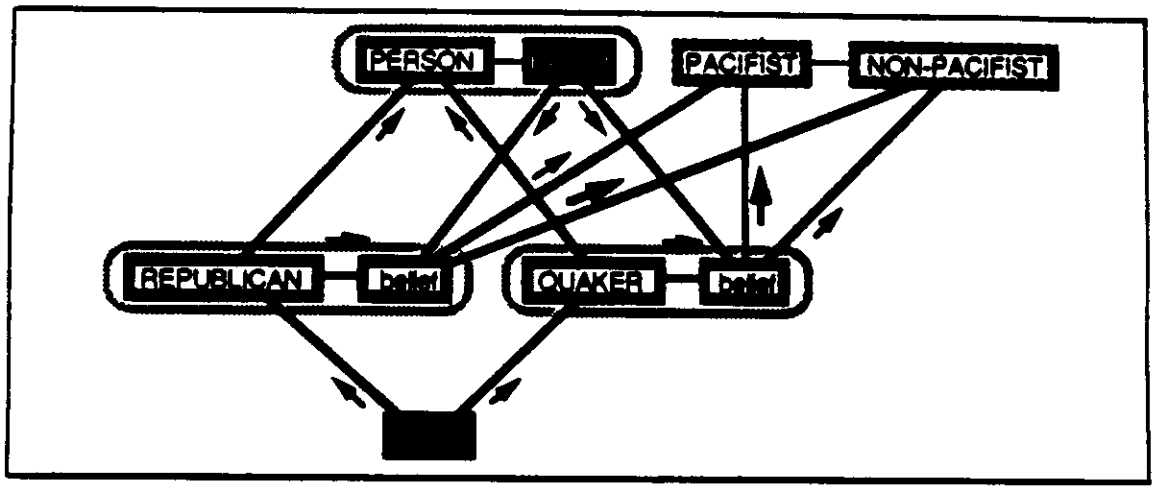

Figure 9.1: Multiple Inheritance

Activation spreads initially from DICK and BELIEF. Both REPUBLICAN:BELIEF and QUAKER:BELIEF fire because they receive activation from two sources. Each is connected to both of the nodes representing the values but with different weights on the connections. These weights would be based on the known proportions. In the figure this is indicated by the thicker arrows from REPUBLICAN:BELIEF to PACIFIST and from QUAKER:BELIEF to NON-PACIFIST. PACIFIST and NON-PACIFIST compete with each other in a WTA network. Either one will receive enough activation from the BELIEF nodes to fire, or the firing WTA hub node (not shown) will cause one to fire. In either case one node will win on the basis of information from both of the types associated with DICK, as we would like.

One feature of the WTA model not found in Shastri's model is a decay mechanism. This is not necessary for the types of problems he considers, but any processing which takes place over time should take into account short-term forgetting as well as errors resulting when activation remains on interfering nodes.

It is not clear how (or whether) Shastri can handle crosstalk that can interfere with recognition. For example, if the task is to determine whether a particular instance of killing is an instance of the concept ASSASSINATE, the OBJECT of the killing must be a POLITICAL-FIGURE. If the ACTOR is a POLITICAL-FIGURE, a system might erroneously classify the killing as assassination because activation from POLITICAL-FIGURE would spread to both the ACTOR and OBJECT roles of the ASSASSINATE schema. In the CLM model this problem is dealt with through the use of WTA networks which stagger the firing of the interfering role nodes in the instance.

9.4. Other Relevant Work

9.4.1. The Lexicon in Linguistics

Linguistics has tended to draw lines between more general syntactic knowledge and more specific lexical knowledge, but once one recognizes that lexical knowledge exists at different levels of generality, it is a logical next step to attempt to integrate grammar and lexicon into a single hierarchy as is done in the CLM model. While this is still not a popular view in linguistics, it can be found in the work of Hudson (1984) and Langacker (1987), who have developed network models of linguistic knowledge.

Linguistics and the fields that it influences have tended to downplay the significance of knowledge at the level of lexical patterns. There are recent signs, however, that relatively specific linguistic knowledge is finally receiving its due. Within linguistics Fillmore (Fillmore, 1979; Fillmore, Kay, & O'Connor, 1986) has devoted much of his recent energy to elucidating the special pragmatic, semantic, and syntactic properties of particular constructions and expressions such as *let alone*. Langacker (1987) has proposed a hierarchical organization for linguistic knowledge which incorporates all levels of generality, including specific phrases. Pawley & Syder (1983) have argued convincingly that a good deal of a speaker's knowledge must be in the form of pre-compiled phrasal chunks in order for speakers to achieve the fluency they do. Peters (1983) and Wong Fillmore (1977) have demonstrated the role of relative specific phrase-length units in first and second language acquisition.

The CLM model differs from these models in representing lexical and syntactic schemas as types of acts with roles for the participants as well as the constituents and the meaning. The linguistic and psycholinguistic theories also suffer from a lack of formalization and a disregard for the processing issues which are central for the present effort.

9.4.2. Speech Acts

Many of the ideas regarding illocutionary acts and reference in the CLM model are related to computational approaches to the generation and recognition of speech acts, in particular the work of Appelt (1985) and Perrault and Allen (1980). This research builds on the basic work on speech acts by Austin (1962) and Searle (1969), who taught the importance of viewing utterances in terms of the effect they have on the world rather than (or in addition to) their truth value. The distinction between the level of illocutions and utterances in the present model corresponds to that made between "illocutionary acts" and "surface speech acts" by Appelt and by Perrault and Allen. The CLM approach differs in making greater use of relatively specific schemas and in the use of spreading activation for accessing schemas.

9.4.3. Language Transfer and Bilingualism

Research on transfer in second language acquisition has focused mainly on the issue of when transfer will occur and when it will not. Among the hypotheses made, for example, are claims that the extent of transfer depends on the "perceived distance" between the L1 and the L2 (Kellerman, 1977), that lexical transfer tends to occur only for the "core meanings" of words (Kellerman, 1978), and that transfer presupposes the potential within the L2 input for generalization along the lines of the transfer (Andersen, 1983). While these researchers have provided what appear to be interesting constraints on the process of transfer, the problem with their work is that the principles they have arrived at do not relate directly to processing. For example, it is not clear how the "semantic equivalence hypothesis" that is ascribed to L2 learners by Ijaz (1986) would translate to a heuristic that is applied during comprehension or generation. Sharwood Smith (1979) is an exception in this regard. Taking the perspective of cognitive modeling, he considers how transfer might take place during comprehension.

Several researchers (e.g., Corder, 1983; Adjémian, 1983) make a distinction between transfer as an acquisition process and transfer as a generation strategy. The latter tends to be viewed as relatively unimportant for learning or even theoretically interesting (Adjémian, 1983; Krashen, 1983). The present study, by contrast, views the transfer and other processing that are involved in coping with gaps during generation as a source of progress as well as potentially harmful misgeneralizations.

Of course the CLM model currently does not offer a theory of transfer in second language acquisition. It is concerned only with some aspects of transfer during generation and with the nature of the associations that are the result of acquisition processes. However, there is the assumption that transfer can be accommodated within the confines of the model. In this regard a significant difference between this and other approaches is the attempt to have transfer arise as a natural consequence of the basic mechanisms involved in comprehension and generation. There is no place in such a model for rules or heuristics peculiar to transfer in second language acquisition.

Research on bilingualism that is relevant to the present model concerns intrasentential code switching, the organization of bilingual memory, and the nature of the "switch" mechanism which permits speakers to change rapidly from one language to another.

Linguistic and psycholinguistic work on code switching (e.g., Joshi, 1985; Nishimura, 1986; Pfaff, 1979) has focused on issues such as whether there needs to be a separate grammar for switching, in what ways switching is constrained, and whether a sentence involving code switching can be assigned unambiguously to one "matrix language" or another. Some of the research is oriented towards the formulation of formal models. Joshi (1985), for example, has posited a set of constraints on which syntactic categories are switchable. However, there has been little or no attempt to integrate the models into theories of generation and analysis.

As with speech errors and transfer, the CLM model was not designed to handle the range of code switching phenomena. Again the approach differs from others in beginning with a general model of processing and memory and attempting to account for behavior of a specific type in these general terms. Within the CLM framework there is no explicit code switching mechanism: switching is simply a consequence of the way in which memory is organized and the way in which the spreading activation mechanism responds to conceptual input and contextual features.

It remains to be seen how well the model will handle the purported "constraints" on switching, but preliminary considerations are encouraging. One of the striking restrictions

appears to be the lack of instances in which only a closed-class item such as an article is switched. For example, a Spanish-English bilingual would apparently rarely produce a sentence such as *put it on la table*. Within the CLM model, this follows from the fact that the schémas specifying such words are normally selected in a top-down, paradigm-driven fashion (see Section 5.2.1) and the fact that the processing of NPs in a clause is driven in part by the verb GU. That is, given an English verb, all top-down selections for NPs in the clause will be English GUs.

Another interesting phenomenon is the tendency for switched items to take on the syntactic properties of their translations in the other language. For example, when Spanish-English bilinguals insert English nouns in an otherwise Spanish sentence, they tend to take articles with the grammatical gender of the Spanish translations of the nouns (Cornejo, 1973), e.g., *una bike* (*una bicicleta*). But this is just what would be expected in a spreading activation model of processing. That is, in the generation of *una bike*, though the English GU *BIKE is the one finally selected, the Spanish GU *BICICLETA would also receive activation, sending some activation in turn to the general GU for feminine noun phrases. This priming would enable the feminine schema to win out over the masculine schema with which it competes.

Psycholinguistic research on the organization of memory in bilinguals and second language learners (e.g., Dalrumple-Alford, 1984; Hummel, 1986; Kirsner, 1986) has focused on the degree to which the lexicons for the two languages are integrated. In general, the orientation of the CLM model is towards a view of the bilingual lexicon as not fundamentally different from the monolingual lexicon. To a certain extent, this is in agreement with Kirsner's (1986) position that there is only one lexicon in bilingual memory. On the other hand, lexical GUs from different languages that correspond roughly to the same concept can point to distinct semantic features through their CONTENT roles. This would help to explain experimental results which require some degree of independence for the linguistic systems. For example, Hummel (1986) showed that bilinguals remember the content of a text much better if it appears in a bilingual rather than unilingual form. As Hummel argues, this is probably related to the degree of elaboration involved in processing in two languages. When schemas from two languages are used in comprehension, contact should be made with a greater variety of concepts than would be the case with only one language.

9.5. Summary

This chapter has contrasted the CLM model with related research, in particular in the areas of language generation and more general models of cognitive processing. Unlike most theories of generation, the CLM model has its generation knowledge embodied in the pattern of connections in its network memory. All processing is carried out through a procedure which spreads activation without regard to the sorts of concepts represented by network nodes. Two spreading activation models of generation have also been discussed, Dell (1986) and Kalita and Shastri (1987). These are closer to the CLM model in approach, but neither accounts for the range of effects or of linguistic phenomena covered by the CLM model. For example, neither of these models deals with the problem of robustness with respect to deficient linguistic knowledge or the representation of linguistic knowledge in a form that permits it to be used in comprehension as well as generation.

In comparison to other general cognitive models, the CLM approach is unusually simple in terms of the primitive objects and operations it works with. The view taken is that knowledge is representable in a single declarative format, making extensive use of

winner-take-all networks, and that spreading activation and short-term decay can implement the recognition and inheritance that underlie complex processing.

Chapter 10

Implementation

10.1. Running the Program

The CLM model has been implemented in a computer program called CHIE[23]. The program consists of three parts: (1) the functions for creating memory nodes and connections, (2) the procedure for spreading activation through the network, and (3) the actual memory network used in generating the examples.

To generate a sentence, the program requires general conceptual and linguistic knowledge and, in addition, pieces of the network representing particular people, objects, and facts to be referred to. The user calls the top-level function `propagate`, which spreads activation through the network. The argument to this function is a list of lists of nodes to be activated "externally". Each sublist of nodes is activated on a different time step. The interval between the external activations is a variable which the user can set.

The spreading activation algorithm can be used for recognition (classification) tasks as well as for tasks involving action. For example, the user can define schemas for different species of birds, then propagate activation from a set of bird feanures and have one of the schemas selected.

10.2. Sample Trace

What follows is a trace of the program as it generates the sentence *could you take the empty box to the garage?*. Some lines have been left out to shorten the example, but otherwise the trace appears exactly as it is printed out by the program.

This example illustrates GI and GU selection, role binding, and sequencing. The generated sentence is intended to be addressed to the speaker's husband. The speaker has a very specific GI for the purpose of getting her husband to carry heavy objects out of the house, and this schema is matched by the input. This schema is shown in Figure 10.1. The GI has the *could you* question form as its PLAN.

```
(REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ is-a AGENCY
    (planner CHIE)
    (assignee JOHN)
    (assignment (OTHER-PTRANS ?A)
        (actor JOHN)
        (object MEDIUM-WEIGHT-OBJECT)
        (source CHIES-HOUSE:INSIDE)
        (destination CHIES-HOUSE:VICINITY))
    (plan YES-NO-QUESTION
        (subject *YOU)
        (aux "COULD")
        (content ?A)
        (hearer JOHN)))
```

Figure 10.1: GI for Generating *could you take the empty box to the garage?*

[23] *Chie* is the Japanese word for 'wisdom' and also a women's given name.

The GU selected for the clause is *TAKE/PTRANS (*take* in its meaning of PHYSICAL-TRANSFER rather than TRANSFER-OF-CONTROL). This selection takes place through the GU *OTHER-PHYSICAL-TRANSFER, a schema whose CONTENT is the general concept of transferring something other than the actor of the transfer himself. This GU has two subtypes, *TAKE/PTRANS and *BRING. *BRING is selected if the DESTINATION of the transfer is toward the LOCATION of the SPEAKER. Otherwise, *TAKE/PTRANS, the default is selected. For the DIRECT-OBJECT and DESTINATION constituents, there are role binding processes to determine what concept is to be referred to in each constituent. Figure 10.2 shows the GUs mentioned.

```
(*OTHER-PTRANS is-a TRANSITIVE-CLAUSE
    (content OTHER-PTRANS
        (actor ?A)
        (object ?O)
        (destination ?D))
    (subject ()
        (content ?A))
    (dir-obj ()
        (content ?O))
    (destination-constit PP
        (destination-marker "TO")
        (destination-np NP
            (content ?D))))

(*TAKE/PTRANS is-a *OTHER-PTRANS
    (verb "TAKE"))

(*BRING/PTRANS is-a *OTHER-PTRANS
    (speaker ()
        (location ?L))
    (content ()
        (destination ?L))
    (verb "BRING"))
```

Figure 10.2: GUs for PHYSICAL-TRANSFER With ACTOR Different From OBJECT

For this example, there is specific knowledge in memory about CHIE herself; JOHN, her husband; BOX4, the box which is to be taken out; CHIES-HOUSE, CHIE's current location; and CHIES-GARAGE, the intended location of BOX4. The act that the hearer (JOHN) is to perform is represented as an instance of the concept OTHER-PHYSICAL-TRANSFER. The representation for this act is shown in Figure 10.3.

```
(PHYSICAL-TRANSFER6 is-a PHYSICAL-TRANSFER
    (actor JOHN)
    (object BOX4)
    (source CHIES-HOUSE:INSIDE)
    (destination CHIES-GARAGE:INSIDE))
```

Figure 10.3: Intended Act for Request *could you take the empty box to the garage?*

The sentence is generated with a call to the function `propagate`. The nodes to be activated externally are in two groups. The first group is activated on the first time step, the second group on the second time step. The nodes in this case represent the speaker's intention. AGENCY is the general type for the intention. The ASSIGNEE (the person who is to perform the act) is JOHN. This relationship is represented by the co-activation of these nodes. The ASSIGNMENT is PTRANS6, the instance of OTHER-PHYSICAL-TRANSFER described above. This relationship is also represented by co-activation. Nodes in PTRANS6 are also activated. Figure 10.4 shows the intention represented by the firing input nodes.

```
(AGENCY3 is-a AGENCY
    (planner CHIE)
    (assignee JOHN)
    (assignment PTRANS6))
```

Figure 10.4: Input to Generation of *could you take the empty box to the garage?*

One feature of the implementation has not been discussed elsewhere in the thesis. Sometimes there is a need to quickly obtain the current value of the HEARER or SPEAKER roles. For example, in order to decide whether an NP can take the article *the*, the system must recognize that the referent is known to, or at least identifiable to, the hearer. The role access mechanism described in Section 5.2.1 could be used in such cases, but it is slow since it requires the traversal of a long path of nodes. Agre and Chapman (1988) have argued that for deictic notions such as HEARER, the cognitive system needs perceptual routines which can access the values of the roles directly. I have assumed that the SPEAKER and HEARER roles in some schemas call such routines. Of course, the program does not actually carry out the routines; instead the user is simply asked to type in the name of the value for the given role. Other role bindings are handled in the way described in Chapter 5.

Since the spreading activation mechanism is a very general one, the comments that the program prints out are not as informative as they would be for a conventional symbolic AI program, where there are procedures for specific symbolic processing operations. To clarify what is happening, I have added additional comments to the trace. These appear in italics.

```
>   (propagate '((agency
                  assignee john
                  assignment ptrans6
                  ptrans6:actor
                  ptrans6:source)
                 (ptrans6:destination
                  ptrans6:obj
                  agency:planner chie))
                'action)

TIME  STEP  1

At the beginning of each time step activation on all nodes decays.
Decaying...
  done
```

Each time step is broken into four increments. During each increment activation can traverse one "fast" connection. Only one "slow" connection can be traversed in an entire time step.

```
Time increment 1
```

The 7 nodes listed below fire here because they make up the head of the list of nodes passed to the function PROPAGATE.

```
   AGENCY firing from external activation
   ASSIGNEE firing from external activation
   JOHN firing from external activation
   ASSIGNMENT firing from external activation
   PTRANS6 firing from external activation
   PTRANS6:ACTOR firing from external activation
   PTRANS6:SOURCE firing from external activation

Time increment 2
```

For each firing node which is not activated "externally", the trace prints out a list of activation sources. For example, INTEND fires as a result of activation from AGENCY.

```
   INTEND firing from
     (AGENCY)
   UNDERTAKE:WANTER firing from
     (ASSIGNEE)
   ACTOR firing from
     (ASSIGNEE)
   MAN firing from
     (JOHN)
```

The GI REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ is a very specific schema which applies when the speaker (CHIE) wants her husband to take something heavy out of the house. Here the ASSIGNEE role in this schema fires as a result of the co-activation of ASSIGNEE and JOHN.

```
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNEE firing from
     (ASSIGNEE JOHN)
   UNDERTAKE:WANTED firing from
     (ASSIGNMENT)
   OTHER-PTRANS firing from
     (PTRANS6)
   OTHER-PTRANS:ACTOR firing from
     (PTRANS6:ACTOR)
   SOURCE firing from
     (PTRANS6:SOURCE)
   CHIES-HOUSE:INSIDE firing from
     (PTRANS6:SOURCE)
   AGENCY:GOAL firing from
     (AGENCY ASSIGNEE ASSIGNMENT)

Time increment 3
   WANTER firing from
     (UNDERTAKE:WANTER)
   HUMAN firing from
     (MAN ASSIGNEE)
```

The HEARER and the ASSIGNEE in a request GI are the same person.

```
   HEARER firing from
     (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNEE)
   WANTED firing from
     (UNDERTAKE:WANTED)
   PTRANS-ACT firing from
```

```
      (OTHER-PTRANS)
    PTRANS:ACTOR firing from
      (ACTOR OTHER-PTRANS:ACTOR)
```

The ACTOR of the ASSIGNMENT is JOHN, which matches the condition in the GI.

```
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:ACTOR firing from
      (OTHER-PTRANS:ACTOR JOHN)
    INSIDE firing from
      (CHIES-HOUSE:INSIDE)
```

The SOURCE of the ASSIGNMENT is CHIES-HOUSE:INSIDE, which also matches the GI.

```
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:SOURCE firing
      from
      (SOURCE CHIES-HOUSE:INSIDE)
```

Activation often converges on nodes high in the is-a hierarchy such as ACT. The firing of such nodes usually is not crucial to the workings of the system because these schemas contain only very general information. These nodes pass very little activation back down to their subtypes (since the weights on the connections are very low), so their firing does not disturb processing. To shorten the trace, I will eliminate further comments on the firing of very general nodes.

```
    ACT firing from
      (INTEND ACTOR UNDERTAKE:WANTED ASSIGNMENT)
    UNDERTAKE firing from
      (UNDERTAKE:WANTER UNDERTAKE:WANTED)

Time increment 4
    PTRANS firing from
      (PTRANS-ACT PTRANS:ACTOR SOURCE)


T I M E   S T E P   2

Decaying...
  done

Time increment 1
```

In this time step the second sublist of nodes passed to PROPAGATE is activated. In the case of the DESTINATION node for PTRANS6, this staggering is necessary in order to avoid crosstalk from the activation of the SOURCE node for the same concept.

```
    CHIE firing from external activation
    AGENCY:PLANNER firing from external activation
    PTRANS6:OBJ firing from external activation
    PTRANS6:DESTINATION firing from external activation
    GOAL firing from
      (INTEND AGENCY:GOAL ARG)

Time increment 2
    WOMAN firing from
      (CHIE MAN HUMAN)
```

In some time increments no nodes at all are able to fire.

```
Time increment 3

Time increment 4


T I M E   S T E P   3
```

```
Decaying...
  done

Time increment 1
   PLANNER firing from
     (AGENCY:PLANNER INTEND ACTOR)
   OTHER-PTRANS:OBJ firing from
     (PTRANS6:OBJ OTHER-PTRANS OTHER-PTRANS:ACTOR)
   BOX4 firing from
     (PTRANS6:OBJ)
   DESTINATION firing from
     (PTRANS6:DESTINATION ARG PTRANS)
   CHIES-GARAGE:INSIDE firing from
     (PTRANS6:DESTINATION INSIDE)

Time increment 2
```

The second role in the GI fires, but this GI requires activation from its ASSIGNMENT role as well before it is selected.

```
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLANNER firing from
     (PLANNER CHIE)
   PTRANS:OBJ firing from
     (OTHER-PTRANS:OBJ OBJ PTRANS)
   BOX firing from
     (BOX4)
```

BOX4 is an empty object.

```
   EMPTY:ATTRIB-OBJ firing from
     (BOX4)
```

BOX4 is represented as an instance of an object of medium weight.

```
   MED-WT-OBJ firing from
     (BOX4)
   CHIES-GARAGE firing from
     (CHIES-GARAGE:INSIDE)

Time increment 3
   SPEAKER firing from
     (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLANNER ACTOR)
   ATTRIB-OBJ firing from
     (EMPTY:ATTRIB-OBJ ARG)
   PHYS-OBJ firing from
     (BOX MED-WT-OBJ INSIDE)
```

The OBJECT of the ASSIGNMENT is an object of medium weight. This matches the corresponding role in the GI.

```
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:OBJ firing from
     (MED-WT-OBJ OTHER-PTRANS:OBJ)
   EMPTY firing from
     (EMPTY:ATTRIB-OBJ)

Time increment 4


T I M E   S T E P   4

Decaying...
  done

Time increment 1
```

```
    CHIES-HOUSE:OUTSIDE:PART firing from
      (CHIES-GARAGE)
```

CHIE's garage is in the vicinity of CHIE's house.

```
    CHIES-HOUSE:VICINITY:PART firing from
      (CHIES-GARAGE)
    JOHN:K-BASE firing from
      (BOX4 CHIES-GARAGE JOHN)
    GARAGE firing from
      (CHIES-GARAGE PHYS-OBJ)
    ATTRIBUTE firing from
      (ATTRIB-OBJ EMPTY)

Time increment 2
    K-BASE firing from
      (JOHN:K-BASE HUMAN)
    CHIES-HOUSE:OUTSIDE firing from
      (CHIES-HOUSE:OUTSIDE:PART)
    CHIES-HOUSE:VICINITY firing from
      (CHIES-HOUSE:VICINITY:PART)

Time increment 3

Time increment 4


T I M E   S T E P   5

Decaying...
  done

Time increment 1
    OUTSIDE firing from
      (CHIES-HOUSE:OUTSIDE PHYS-OBJ)
    VICINITY firing from
      (CHIES-HOUSE:VICINITY PHYS-OBJ)
```

The DESTINATION of the intended transfer is a place which is in CHIES-HOUSE:VICINITY, matching the DESTINATION role for the ASSIGNMENT in the GI.

```
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:DESTINATION
      firing from
      (CHIES-HOUSE:VICINITY DESTINATION)

Time increment 2
```

Now the ASSIGNMENT role in the GI has enough activation from its roles and its parent, OTHER-PTRANS, to fire.

```
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT firing from
      (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:DESTINATION
       REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:OBJ
       ASSIGNMENT OTHER-PTRANS
       REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:ACTOR
       REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:SOURCE)

Time increment 3

Time increment 4


T I M E   S T E P   6
```

```
Decaying...
  done

Time increment 1
```

CLAUSE:CONTENT fires because the intended ASSIGNMENT is the same as the CONTENT of the utterance which is the PLAN of the GI.

```
    CLAUSE:CONTENT firing from
      (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT FACT)
```

The head of the GI now has enough activation to fire. This represents the selection of this schema.

```
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ firing from
      (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT
       REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLANNER
       AGENCY REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNEE)

Time increment 2
```

*The CONTENT role for the general *OTHER-PTRANS GU fires as a result of activation from OTHER-PTRANS, the general type of the intended act in the input, and CLAUSE:CONTENT.*

```
    *OTHER-PTRANS:CONTENT firing from
      (CLAUSE:CONTENT OTHER-PTRANS)

Time increment 3
```

**OTHER-PTRANS is a supertype of the GUs for* bring *and* take. *Selection between these two GUs is made on the basis of the DESTINATION of the transfer. If it is toward the SPEAKER, then *BRING is selected; otherwise *TAKE/PTRANS is selected. The DESTINATION role in the *OTHER-PTRANS GU fires to start off this selection process.*

```
    *OTHER-PTRANS:CONTENT:DESTINATION firing from
      (*OTHER-PTRANS:CONTENT DESTINATION)

Time increment 4


T I M E   S T E P   7

Decaying...
  done

Time increment 1
```

The () in the list of activation sources indicates that the node fires partly on the basis of residual activation following a refractory period.

```
    AGENCY firing from
      (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ ())
    OTHER-PTRANS firing from
      (*OTHER-PTRANS:CONTENT ())
    DESTINATION firing from
      (*OTHER-PTRANS:CONTENT:DESTINATION ())
```

The PLAN role in the GI fires. This represents the utterance that will be used to accomplish the goal.

```
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN firing from
      (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ
       REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT
       REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLANNER
       REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNEE)
```

```
Time increment 2

Time increment 3

Time increment 4


T I M E   S T E P   8

Decaying...
  done

Time increment 1
   PTRANS6:DESTINATION firing from
     (DESTINATION ())
```

The general syntactic type for the utterance is YES-NO-QUESTION. This GU includes the sequencing information for the AUXILIARY and SUBJECT constituents.

```
   Y-N-QUESTION firing from
     (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN)

Time increment 2
   CLAUSE firing from
     (Y-N-QUESTION CLAUSE:CONTENT)
```

The start node for the AUXILIARY in the YES-NO-QUESTION schema fires. In the trace start nodes are indicated simply by the node name, while end nodes have "_end" at the end of the name.

```
   Y-N-QUESTION:AUX firing from
     (Y-N-QUESTION)

Time increment 3
   UTTER firing from
     (CLAUSE ACT SPEAKER HEARER)
```

*The head node of the *OTHER-PTRANS GU fires. This activates its two subtypes, *BRING and *TAKE/PTRANS, which inhibit each other via a WTA network. The WTA hub node is also activated at this time.*

```
   *OTHER-PTRANS firing from
     (CLAUSE *OTHER-PTRANS:CONTENT)

Time increment 4
```

**OTHER-PTRANS is a subtype of the general TRANSITIVE-CLAUSE schema. This GU has general information about the DIRECT-OBJECT constituent.*

```
   TRANSITIVE-CLAUSE firing from
     (*OTHER-PTRANS CLAUSE)
```

*The head node of *OTHER-PTRANS is connected strongly to its SPEAKER role. This fires to test for an intersection with the DESTINATION. If there is one, then *BRING will be selected over *TAKE/PTRANS.*

```
   *OTHER-PTRANS:SPEAKER firing from
     (*OTHER-PTRANS SPEAKER)


T I M E   S T E P   9

Decaying...
  done
```

```
Time increment 1
```
*It is assumed that the *OTHER-PTRANS:SPEAKER node starts up a perceptual routine that determines the current speaker, activating the node for that person. In the program, the user is simply asked to type in the name.*

```
   Give the value of *OTHER-PTRANS:SPEAKER:  chie
   CHIE firing from
     (*OTHER-PTRANS:SPEAKER ())
```

*Activation beginning with the *OTHER-PTRANS:CONTENT:DESTINATION node has reached the DESTINATION of the activated instance of OTHER-PTRANS. If this were the current location of the speaker, this would lead to a chain of firing nodes which would activate the *BRING schema.*

```
   CHIES-GARAGE:INSIDE firing from
     (PTRANS6:DESTINATION ())
   AUX firing from
     (CLAUSE Y-N-QUESTION:AUX)
   SPEAKER firing from
     (UTTER *OTHER-PTRANS:SPEAKER ())

Time increment 2
   WOMAN firing from
     (CHIE ())

Time increment 3
   HUMAN firing from
     (WOMAN ())

Time increment 4


T I M E   S T E P   10

Decaying...
  done

Time increment 1
```

The start node for the AUXILIARY in the GI PLAN fires. This node is directly connected to the word node /COULD.

```
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:AUX firing from
     (AUX REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN)
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLANNER firing from
     (CHIE SPEAKER ())

Time increment 2
```

***The word node fires, representing the utterance of* could.**

```
   /COULD firing from
     (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:AUX)

      /COULD  u t t e r e d


Time increment 3
   WORD firing from
     (/COULD AUX ACT)
```

The end node for the AUXILIARY in the GI fires.

```
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:AUX_END firing from
     (/COULD REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:AUX)
```

```
Time increment 4
```
The end node for the AUXILIARY in the general CLAUSE schema fires.
```
    AUX_END firing from
      (REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:AUX_END AUX)

T I M E   S T E P   11

Decaying...
  done

Time increment 1
```
The end node for AUXILIARY in the YES-NO-QUESTION schema fires. This node sends activation to the SUBJECT/start node in the same schema.
```
    Y-N-QUESTION:AUX_END firing from
      (AUX_END Y-N-QUESTION:AUX)

Time increment 2
```
The SUBJECT constituent is initiated.
```
    Y-N-QUESTION:SUBJECT firing from
      (Y-N-QUESTION:AUX_END Y-N-QUESTION)

Time increment 3
```
The start node for SUBJECT in the CLAUSE schema fires.
```
    SUBJECT firing from
      (Y-N-QUESTION:SUBJECT CLAUSE)

Time increment 4
```
The general NP schema fires as a result of activation from SUBJECT.
```
    NP firing from
      (SUBJECT UTTER)
    *OTHER-PTRANS:SUBJECT firing from
      (SUBJECT *OTHER-PTRANS)
```
The SUBJECT/start node in the GI PLAN fires.
```
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:SUBJECT firing from
      (SUBJECT REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN)

T I M E   S T E P   12

Decaying...
  done

Time increment 1
```
*The SUBJECT role in the GI is associated directly with the *YOU schema, so no selection process is required for this constituent.*
```
    *YOU firing from
      (NP REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:SUBJECT)
    NP:HEARER firing from
      (NP HEARER)
    NP:CONTENT firing from
      (NP)
    DET firing from
      (NP)
```

```
    *OTHER-PTRANS:SUBJECT:CONTENT firing from
      (*OTHER-PTRANS:SUBJECT)

Time increment 2
```
*In the *YOU GU there is a strong connection to the NOUN/start role because this is the only constituent in the NP.*
```
    *YOU:NOUN firing from
      (*YOU)

Time increment 3

Time increment 4
```
*Activation from the SPEAKER and DESTINATION has not converged, so *BRING has not fired. The hub node for the WTA network composed of *BRING and *TAKE/PTRANS fires, sending activation to the two members. The default member, *TAKE/PTRANS, receives more.*
```
      TAKE-BRING-WTA timing out and firing


T I M E   S T E P   13

Decaying...
  done

Time increment 1
```
It is assumed that there is a perceptual routine which accesses the current hearer every time the NP:HEARER node fires. This is required to determine whether the referent is known to the hearer, though this is not relevant for the current NP.
```
    Give the value of NP:HEARER:  john
    JOHN firing from
      (NP:HEARER ())
    OTHER-PTRANS:ACTOR firing from
      (*OTHER-PTRANS:SUBJECT:CONTENT OTHER-PTRANS ())
    NOUN firing from
      (*YOU:NOUN NP)
```
The word node for you fires.
```
    /YOU firing from
      (*YOU:NOUN WORD)

       /YOU  u t t e r e d
```
**TAKE/PTRANS fires as a result of new activation from the WTA hub node.*
```
    *TAKE/PTRANS firing from
      (TAKE-BRING-WTA *OTHER-PTRANS)
    *YOU:CONTENT firing from
      (*YOU NP:CONTENT)

Time increment 2
```
A number of nodes fire as a result of the firing of JOHN, the current hearer. None of these is important for the generation of the sentence.
```
    MAN firing from
      (JOHN HUMAN ())
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNEE firing from
      (JOHN ASSIGNEE ())
    REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:ACTOR firing from
```

```
      (JOHN OTHER-PTRANS:ACTOR ())
    PTRANS6:ACTOR firing from
      (JOHN OTHER-PTRANS:ACTOR ())
```
*The end node for the NOUN in *YOU fires.*
```
    *YOU:NOUN_END firing from
      (/YOU *YOU:NOUN)

Time increment 3
   NOUN_END firing from
     (*YOU:NOUN_END NOUN)

Time increment 4
```
The end node in the NP schema fires because the NOUN is complete. (The NP network does not include post-nominal modifiers.)
```
   NP_END firing from
     (NOUN_END NP)


T I M E   S T E P   14

Decaying...
 done

Time increment 1
```
The SUBJECT of the clause is complete. SUBJECT/end activates VERB/start.
```
   SUBJECT_END firing from
     (NP_END SUBJECT)
   *YOU_END firing from
     (*YOU:NOUN_END NP_END *YOU)

Time increment 2
```
The VERB of the sentence is initiated.
```
   VERB firing from
     (SUBJECT_END CONSTITUENT AUX_END CLAUSE)
   Y-N-QUESTION:SUBJECT_END firing from
     (SUBJECT_END Y-N-QUESTION:SUBJECT)
   *OTHER-PTRANS:SUBJECT_END firing from
     (SUBJECT_END *OTHER-PTRANS:SUBJECT)
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:SUBJECT_END firing from
     (SUBJECT_END REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:PLAN:SUBJECT)

Time increment 3
   TRANSITIVE-CLAUSE:VERB firing from
     (VERB TRANSITIVE-CLAUSE)
```
*The start node for VERB in the *TAKE/PTRANS schema fires. This is connected directly to the /TAKE word node.*
```
   *TAKE/PTRANS:VERB firing from
     (VERB *TAKE/PTRANS)
   Y-N-QUESTION:VERB firing from
     (VERB Y-N-QUESTION:SUBJECT_END Y-N-QUESTION)

Time increment 4
```
Take ***is uttered.***
```
   /TAKE firing from
     (*TAKE/PTRANS:VERB WORD)
```

```
      /TAKE  u t t e r e d


T I M E   S T E P   15

Decaying...
  done

Time increment 1
   *TAKE/PTRANS:VERB_END firing from
     (*TAKE/PTRANS:VERB /TAKE)

Time increment 2
   VERB_END firing from
     (*TAKE/PTRANS:VERB_END VERB CONSTITUENT_END)

Time increment 3
   Y-N-QUESTION:VERB_END firing from
     (VERB_END Y-N-QUESTION:VERB)
```

The firing of VERB/end in the TRANSITIVE-CLAUSE schema activates the DIRECT-OBJECT/start node.

```
   TRANSITIVE-CLAUSE:VERB_END firing from
     (VERB_END TRANSITIVE-CLAUSE:VERB)

Time increment 4
```

The DIRECT-OBJECT is initiated.

```
   DIR-OBJ firing from
     (TRANSITIVE-CLAUSE:VERB_END CONSTITUENT NP TRANSITIVE-CLAUSE)


T I M E   S T E P   16

Decaying...
  done

Time increment 1
```

The NP/start node fires again to be reused for the second NP in the sentence. This node has a short refractory period and a low resumptive activation level to make this reuse possible.

```
   NP firing from
     (DIR-OBJ ())
   *OTHER-PTRANS:DIR-OBJ firing from
     (DIR-OBJ *OTHER-PTRANS)

Time increment 2
```

NP:HEARER fires whenever NP is selected. Together with the firing of NP:CONTENT, this will determine whether the referent is known to the hearer. If it is, THE-NP is selected.

```
   NP:HEARER firing from
     (NP ())
   NP:CONTENT firing from
     (NP ())
```

The DETERMINER is the initial constituent in the NP so it receives enough activation to fire now.

```
   DET firing from
     (NP ())
```

*The CONTENT role for the DIRECT-OBJECT of the *OTHER-PTRANS GU fires, initiating a role binding process. This is a merged node combining the DIRECT-OBJECT:CONTENT with the CONTENT:OBJECT of the GU.*

```
    *OTHER-PTRANS:DIR-OBJ:CONTENT firing from
      (*OTHER-PTRANS:DIR-OBJ NP:CONTENT)

Time increment 3

Time increment 4


T I M E   S T E P   17

Decaying...
  done

Time increment 1
    Give the value of NP:HEARER:  john
    HEARER firing from
      (NP:HEARER ())
    JOHN firing from
      (NP:HEARER)
```

Next step in the role binding process for the direct object.

```
    OTHER-PTRANS:OBJ firing from
      (*OTHER-PTRANS:DIR-OBJ:CONTENT ())

Time increment 2
```

PTRANS6:OBJ fires as a result of activation from OTHER-PTRANS:OBJ and resumptive activation following firing. The role binding process has reached the input concept.

```
    PTRANS6:OBJ firing from
      (OTHER-PTRANS:OBJ ())

Time increment 3
```

Role binding for the direct object is complete.

```
    BOX4 firing from
      (PTRANS6:OBJ ())

Time increment 4
```

Nodes representing features of BOX4 fire.

```
    BOX firing from
      (BOX4 ())
    EMPTY:ATTRIB-OBJ firing from
      (BOX4 ())
    MED-WT-OBJ firing from
      (BOX4 ())
```

The node representing JOHN's KNOWLEDGE-BASE fires as a result of activation from BOX4 (the referent) and JOHN (the hearer). This represents the fact that the hearer knows of the referent, so the THE-NP GU will be selected.

```
    JOHN:K-BASE firing from
      (BOX4 JOHN ())


T I M E   S T E P   18

Decaying...
  done
```

```
Time increment 1
```
*The CONTENT role of the lexical GU *BOX fires.*

```
   *BOX:CONTENT firing from
     (BOX NP:CONTENT)
```

The general ATTRIBUTE-OBJECT node in the ATTRIBUTE schema fires because BOX4 has an attribute (emptiness) which has fired.

```
   ATTRIB-OBJ firing from
     (EMPTY:ATTRIB-OBJ ())
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:OBJ firing from
     (OTHER-PTRANS:OBJ MED-WT-OBJ ())
   K-BASE firing from
     (JOHN:K-BASE HUMAN ())
   EMPTY firing from
     (EMPTY:ATTRIB-OBJ ())

Time increment 2
```

The firing of this role in the NP schema signals that an adjective is appropriate.

```
   ADJ-MOD:CONTENT:ATTRIB-OBJ firing from
     (ATTRIB-OBJ NP:CONTENT)
```

This node fires when the CONTENT is known to the HEARER. The THE-NP schema is selected as a result.

```
   THE-NP:HEARER:K-BASE firing from
     (K-BASE NP:HEARER)
```

*The *BOX GU is selected.*

```
   *BOX firing from
     (*BOX:CONTENT NP)

Time increment 3
   THE-NP:CONTENT firing from
     (THE-NP:HEARER:K-BASE NP:CONTENT)
```

The firing of ADJECTIVE-MODIFIER:CONTENT primes ADJECTIVE-MODIFIER/start in the NP schema. Later this priming will cause the adjective to win out over the noun for the position following the determiner.

```
   ADJ-MOD:CONTENT firing from
     (ADJ-MOD:CONTENT:ATTRIB-OBJ)
   THE-NP:HEARER firing from
     (THE-NP:HEARER:K-BASE HEARER)

Time increment 4
```

The THE-NP schema is selected.

```
   THE-NP firing from
     (THE-NP:CONTENT NP *YOU)


T I M E   S T E P   19

Decaying...
  done

Time increment 1
   ADJ-PHRASE:CONTENT firing from
     (ADJ-MOD:CONTENT ATTRIBUTE)
   ATTRIBUTE firing from
     (ATTRIB-OBJ EMPTY ())
```

The DETERMINER/start node in the THE-NP GU fires. This node is connected to the node for the word the.

```
    THE-NP:DET firing from
      (THE-NP DET)

Time increment 2
```

*The CONTENT role for the *EMPTY GU fires on the basis of a property of BOX4.*

```
    *EMPTY:CONTENT firing from
      (ADJ-PHRASE:CONTENT EMPTY)

Time increment 3

Time increment 4


T I M E   S T E P   20

Decaying...
  done

Time increment 1
```

The word* the *is uttered.

```
    /THE firing from
      (THE-NP:DET WORD)

       /THE  u t t e r e d


Time increment 2
    WORD firing from
      (/THE ())
    THE-NP:DET_END firing from
      (/THE THE-NP:DET)

Time increment 3
    DET_END firing from
      (THE-NP:DET_END DET NOUN NP_END)

Time increment 4
```

Activation from DETEREMINER/end activates both ADJECTIVE-MODIFIER/start and NOUN/start, which inhibit one another via a WTA network. In this case ADJECTIVE-MODIFIER/start wins out because of the activation is received along the path beginning with EMPTY:ATTRIBUTE-OBJECT.

```
    ADJ-MOD firing from
      (DET_END ADJ-MOD:CONTENT NP)
    NOUN can't fire because of competition


T I M E   S T E P   21

Decaying...
  done

Time increment 1
```

The general ADJECTIVE-PHRASE schema is activated from ADJECTIVE-MODIFIER/start.

```
    ADJ-PHRASE firing from
```

```
      (ADJ-MOD ADJ-PHRASE:CONTENT)

Time increment 2
```
*ADJECTIVE-PHRASE/start causes the primed *EMPTY GU to be activated.*
```
   *EMPTY firing from
     (ADJ-PHRASE *EMPTY:CONTENT)
```
The ADJECTIVE constituent in the ADJECTIVE-PHRASE schema (the only constituent in the current version of CHIE) is activated.
```
   ADJ firing from
     (ADJ-PHRASE)

Time increment 3

Time increment 4


T I M E   S T E P   22

Decaying...
  done

Time increment 1
```
*ADJECTIVE/start in the *EMPTY GU fires. This is connected to the /EMPTY word node.*
```
   *EMPTY:ADJ firing from
     (*EMPTY ADJ)

Time increment 2

Time increment 3

Time increment 4


T I M E   S T E P   23

Decaying...
  done

Time increment 1
```
The word empty is uttered.
```
   /EMPTY firing from
     (*EMPTY:ADJ WORD)

      /EMPTY  u t t e r e d


Time increment 2
   *EMPTY:ADJ_END firing from
     (/EMPTY *EMPTY:ADJ)

Time increment 3
   ADJ_END firing from
     (*EMPTY:ADJ_END ADJ)

Time increment 4
```

The end of the adjective phrase. Control is now sent back to the NP schema.

```
   ADJ-PHRASE_END firing from
     (ADJ_END ADJ-PHRASE)
```

```
T I M E   S T E P   24

Decaying...
  done

Time increment 1
```

The end of the ADJECTIVE-MODIFIER constituent in the NP schema.

```
   ADJ-MOD_END firing from
     (ADJ-PHRASE_END ADJ-MOD NP_END)
   *EMPTY_END firing from
     (*EMPTY:ADJ_END ADJ-PHRASE_END *EMPTY)

Time increment 2
```

Activation from ADJECTIVE-MODIFIER/end causes NOUN/start to fire.

```
   NOUN firing from
     (ADJ-MOD_END DET_END NP NOUN_END)

Time increment 3
```

*The primed NOUN/start node in the *BOX GU now fires.*

```
   *BOX:NOUN firing from
     (NOUN *BOX)

Time increment 4
```

The word **box** *is uttered.*

```
   /BOX firing from
     (*BOX:NOUN WORD)

      /BOX  u t t e r e d


T I M E   S T E P   25

Decaying...
  done

Time increment 1
   *BOX:NOUN_END firing from
     (*BOX:NOUN /BOX)

Time increment 2
```

The end of the NOUN portion of the NP.

```
   NOUN_END firing from
     (*BOX:NOUN_END NOUN CONSTITUENT_END)

Time increment 3
```

The NOUN is the last constituent in this simplified NP, so the NP/end node fires.

```
   NP_END firing from
     (NOUN_END UTTER_END ADJ-MOD_END DET_END)

Time increment 4
```

```
T I M E   S T E P   26

Decaying...
  done

Time increment 1
```
The end node for the DIRECT-OBJECT in the TRANSITIVE-CLAUSE schema fires.
```
   DIR-OBJ_END firing from
     (NP_END CONSTITUENT_END DIR-OBJ)
   THE-NP_END firing from
     (NP_END THE-NP:DET_END THE-NP)
   *BOX_END firing from
     (*BOX:NOUN_END NP_END *BOX)

Time increment 2
```
*The DIRECT-OBJECT/end node in the *OTHER-PTRANS GU fires, sending activation to the constituent which refers to the DESTINATION of the transfer.*
```
   *OTHER-PTRANS:DIR-OBJ_END firing from
     (DIR-OBJ_END *OTHER-PTRANS:DIR-OBJ)

Time increment 3
```
*The start node for the DESTINATION-CONSTITUENT in the *OTHER-PTRANS schema fires. This constituent has two of its own constituents, the DESTINATION-MARKER, realized as* to, *and the DESTINATION-NP, an NP referring to the location.*
```
   DESTINATION-CONSTIT firing from
     (*OTHER-PTRANS:DIR-OBJ_END *OTHER-PTRANS)

Time increment 4
```
The start node for the DESTINATION-MARKER fires. This is connected to the word node /TO.
```
   DESTINATION-MARKER firing from
     (DESTINATION-CONSTIT)


T I M E   S T E P   27

Decaying...
  done

Time increment 1
```
The word to is uttered.
```
   /TO firing from
     (DESTINATION-MARKER WORD)

      /TO  u t t e r e d


Time increment 2
   WORD firing from
     (/TO ())
   DESTINATION-MARKER_END firing from
     (/TO DESTINATION-MARKER CONSTITUENT_END)

Time increment 3
```

The start of the DESTINATION-NP.

```
   DESTINATION-NP firing from
     (DESTINATION-MARKER_END CONSTITUENT DESTINATION-CONSTIT)
```

```
Time increment 4
```

The NP schema is used once more.

```
   NP firing from
     (DESTINATION-NP NOUN)
```

The CONTENT node for the DESTINATION-NP fires, starting another role binding process.

```
   DESTINATION-NP:CONTENT firing from
     (DESTINATION-NP NP:CONTENT)
```

```
T I M E   S T E P   28

Decaying...
  done

Time increment 1
```

The next step in the role binding process for the DESTINATION-NP.

```
   *OTHER-PTRANS:CONTENT:DESTINATION firing from
     (DESTINATION-NP:CONTENT)
   NP:HEARER firing from
     (NP ())
   NP:CONTENT firing from
     (NP DESTINATION-NP:CONTENT ())
   DET firing from
     (CONSTITUENT NP ())

Time increment 2
```

Part of the role binding process.

```
   DESTINATION firing from
     (*OTHER-PTRANS:CONTENT:DESTINATION PTRANS)

Time increment 3
```

The role binding process reaches the input concept.

```
   PTRANS6:DESTINATION firing from
     (DESTINATION ())

Time increment 4
```

The role binding process is complete. CHIES-GARAGE:INSIDE is "bound" to the CONTENT of the DESTINATION-NP.

```
   CHIES-GARAGE:INSIDE firing from
     (PTRANS6:DESTINATION ())

T I M E   S T E P   29

Decaying...
  done

Time increment 1
```

The user is asked for the HEARER again. This will determine whether the referent, CHIES-GARAGE:INSIDE, is known to the hearer.

```
   Give the value of NP:HEARER:  john
```

```
   HEARER firing from
     (NP:HEARER ())
   JOHN firing from
     (NP:HEARER ())
   INSIDE firing from
     (CHIES-GARAGE:INSIDE PHYS-OBJ)
   CHIES-GARAGE firing from
     (CHIES-GARAGE:INSIDE)

Time increment 2
   MAN firing from
     (JOHN HUMAN ())
   PTRANS6:ACTOR firing from
     (JOHN ())

Time increment 3
   HUMAN firing from
     (MAN HEARER ())
   OTHER-PTRANS:ACTOR firing from
     (PTRANS6:ACTOR ())

Time increment 4
   PTRANS:ACTOR firing from
     (OTHER-PTRANS:ACTOR PTRANS ())
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:ACTOR firing from
     (OTHER-PTRANS:ACTOR JOHN ())


T I M E   S T E P   30

Decaying...
  done

Time increment 1
   GARAGE firing from
     (CHIES-GARAGE PHYS-OBJ)
   CHIES-HOUSE:OUTSIDE:PART firing from
     (CHIES-GARAGE)
   CHIES-HOUSE:VICINITY:PART firing from
     (CHIES-GARAGE)
```

CHIES-GARAGE is known to JOHN.

```
   JOHN:K-BASE firing from
     (JOHN CHIES-GARAGE ())

Time increment 2
```

*The CONTENT role of the *GARAGE GU fires as a result of activation from the general GARAGE schema.*

```
   *GARAGE:CONTENT firing from
     (GARAGE NP:CONTENT)

Time increment 3
```

*The *GARAGE GU is selected for the NP.*

```
   *GARAGE firing from
     (*GARAGE:CONTENT NP)

Time increment 4
```

```
T I M E   S T E P   31

Decaying...
  done

Time increment 1
   K-BASE firing from
     (JOHN:K-BASE HUMAN ())
   REQUEST-HUSBAND-TAKE-OUT-HEAVY-OBJ:ASSIGNMENT:DESTINATION
     firing from
     (CHIES-HOUSE:VICINITY DESTINATION)

Time increment 2
   THE-NP:HEARER:K-BASE firing from
     (K-BASE NP:HEARER ())

Time increment 3
   THE-NP:CONTENT firing from
     (THE-NP:HEARER:K-BASE NP:CONTENT ())
   THE-NP:HEARER firing from
     (THE-NP:HEARER:K-BASE HEARER ())

Time increment 4
```
The THE-NP GU is again selected on the basis of activation originally from JOHN:KNOWLEDGE-BASE.
```
   THE-NP firing from
     (THE-NP:CONTENT NP THE-NP_END)


T I M E   S T E P   32

Decaying...
  done

Time increment 1
   THE-NP:DET firing from
     (THE-NP DET THE-NP:DET_END)

Time increment 2

Time increment 3

Time increment 4


T I M E   S T E P   33

Decaying...
  done

Time increment 1
```
The word* the *is uttered.
```
   /THE firing from
     (THE-NP:DET WORD ())

     /THE  u t t e r e d
```

```
Time increment 2
   WORD firing from
     (/THE ())
   THE-NP:DET_END firing from
     (/THE THE-NP:DET THE-NP_END)

Time increment 3
   DET_END firing from
     (THE-NP:DET_END DET NP_END)

Time increment 4


T I M E   S T E P   34

Decaying...
  done

Time increment 1
```

For this NP ADJECTIVE-MODIFIER/start does not receive activation on the basis of a feature of the referent , so NOUN/start fires directly after DET/end.

```
   NOUN firing from
     (DET_END NP NOUN_END)

Time increment 2
   *GARAGE:NOUN firing from
     (NOUN *GARAGE)

Time increment 3
```

The word garage is uttered.

```
   /GARAGE firing from
     (*GARAGE:NOUN WORD)

      /GARAGE  u t t e r e d


Time increment 4
   *GARAGE:NOUN_END firing from
     (/GARAGE *GARAGE:NOUN)


T I M E   S T E P   35

Decaying...
  done

Time increment 1
   NOUN_END firing from
     (NOUN *GARAGE:NOUN_END)

Time increment 2
   NP_END firing from
     (NOUN_END DET_END)

Time increment 3
```

```
Time increment 4

T I M E   S T E P   36

Decaying...
  done

Time increment 1
```
The end of the DESTINATION-NP.
```
    DESTINATION-NP_END firing from
      (NP_END DESTINATION-NP)
    THE-NP_END firing from
      (NP_END THE-NP:DET_END THE-NP)
    *GARAGE_END firing from
      (NP_END *GARAGE:NOUN_END *GARAGE)

Time increment 2
    DESTINATION-CONSTIT_END firing from
      (DESTINATION-NP_END DESTINATION-MARKER_END
       DESTINATION-CONSTIT)

Time increment 3

Time increment 4


T I M E   S T E P   37

Decaying...
  done

Time increment 1
    *OTHER-PTRANS_END firing from
      (DESTINATION-CONSTIT_END *OTHER-PTRANS)

Time increment 2

Time increment 3

Time increment 4


T I M E   S T E P   38

Decaying...
  done

Time increment 1
    TRANSITIVE-CLAUSE_END firing from
      (*OTHER-PTRANS_END TRANSITIVE-CLAUSE)
```
The end of the CLAUSE.
```
    CLAUSE_END firing from
      (*OTHER-PTRANS_END CLAUSE)

Time increment 2
```

```
Time increment 3

Time increment 4
```

Without further external activation, no more nodes fire beyond this point.

10.3. The Program

CHIE is written in the T language (Slade, 1987), a lexically scoped dialect of LISP, and runs on Apollo workstations. The memory building portion of the program takes up about 700 lines of code, the activation spreading portion about 300 lines of code, and the memory necessary for the generation of the sentence in the trace above about 700 lines of code. The memory created in order to generate the example sentence consists of 292 nodes and 1374 connections. Of course much of this memory is general knowledge usable in generating many sentences. The generation of the example sentence requires approximately 200 seconds in real time on an Apollo 3010.

CHIE is currently capable of generating phrases and sentences of the types listed as examples in section 1.1. In terms of structure, it can handle clauses with any number of NP or PP arguments, NPs with or without single adjectives, and personal pronouns.

A version of CHIE is given in the Appendix.

Chapter 11

Conclusions

11.1. Limitations and Future Work

This section discusses some of the shortcomings of the CLM model and possible ways in which these can be remedied in future research.

11.1.1. Distributing the CLM Network

Localized representations such as those used in the CLM model suffer from two general drawbacks, representational brittleness and memory limitations (Feldman, 1986). While localized connectionist models are robust with respect to incomplete or faulty input, they are fragile with respect to damage to the network itself. The destruction of a single node in such a model means the elimination of an entire concept. Consider the absurdity, for example, of the elimination of all of the system's knowledge of itself through the removal of the single SELF node. The second problem relates to the demands placed on memory by a system in which each concept has an associated node. While the supply of neurons in the human brain dwarfs the memories in computers, it is of course not unlimited, and the supply might very well run out. A particularly troublesome aspect of this issue is the question of where the nodes and connections come from which are needed to represent new facts and objects that the system learns about.

There are two distinct senses in which representations may be distributed. On the one hand, each concept and formal linguistic entity (phrase, word, syllable, etc.) needs to be represented as a pattern across a group of units rather than a single unit. This type of representation achieves graceful degradation. On the other hand, each unit must participate in the representation of a number of concepts or linguistic entities. This type of representation, sometimes called coarse coding (Hinton, McClelland, & Rumelhart, 1986), achieves economy of storage.

Rather than scrapping the localized approach, one solution lies, as Feldman (1986) suggests, in superimposing some sort of distributed representation scheme on the network. In what follows I consider one way in which this might be achieved and what advantages it would bring in addition to those related to robustness and memory economy.

In order to solve the problem of representational brittleness, each CLM node could simply be distributed over a set of units. However, this move does nothing to solve the problem of constraints on memory size; it actually compounds the problem by multiplying the number of units required. To deal with this issue, each unit must participate in the representation of multiple CLM nodes.

One useful idea in this regard, which has been suggested by several researchers (e.g., Feldman & Ballard, 1982; Hinton, McClelland, & Rumelhart, 1986; McClelland & Kawamoto, 1986), is conjunctive coding, by which each unit represents a conjunction of primitive features. One possibility is to have units represent conjunctions of role and value features. For a high-level schema role, such as ACTOR (in the general ACT schema) or NP:CONTENT, there would be a large number of units, each representing an ACTOR or NP:CONTENT of a very specific type. Roles further down the hierarchy such as LOVE:ACTOR and *PEPPER:CONTENT would be represented by subsets of these larger sets, that is, just those units not inconsistent with the more specific senses of these roles. Roles of instances, such as the ACTOR of a particular instance of LOVE, or the CONTENT of a particular use of the noun *pepper*, would correspond to subsets of these smaller sets. Figure 11.1 illustrates this for the ACTOR example.

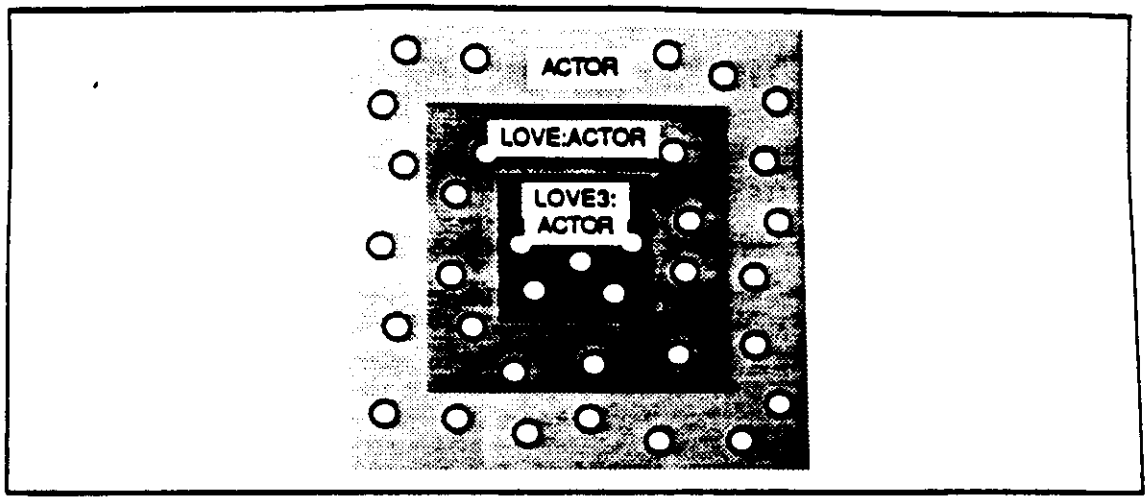

Figure 11.1: Distributed Representation of Role Hierarchy

Since the number of nodes involved in the representation increases with the generality of the concept, the loss of a group of nodes is more likely to disrupt memory and processing for a more specific than a more general concept. Note that the general ACTOR group in the figure would include units representing features which are not necessarily true of all actors but which could be true of a given actor, for example, MALE.

Recall that in the CLM model there is also the need for a prohibitively large number of connections, in particular joining the CONTENT roles of GUs. A way around this problem, in a distributed model like the one illustrated in Figure 11.1, would be to replace these connections with connections that join the units representing constrasting role-value pairs. For example, a small set of mutually inhibiting units might represent the conjunctions of a single role with different values. With this approach, we would still get the inhibition we need between similar concepts such as *DEPOSIT:CONTENT and *GIVE:CONTENT (as in Figure 4.20) as well as dissimilar concepts such as *MOTHER:CONTENT and *LETTER:CONTENT (as in Figure 5.18). The former would be based on the inhibition between units such as DURATION=TEMPORARY and DURATION=PERMANENT, and the latter would be based on many different competing pairs, for example, AGENTIVITY=HIGH and AGENTIVITY=NIL. Fewer connections would be required than in the localized approach because units would only need to inhibit those representing similar conjunctions of concepts.

It is desirable to be able to represent new facts in terms of the units and connections that already exist because the store of neurons does not grow in adults. Given the approach illustrated in Figure 11.1, we might represent the fact that Mary loves John as shown in Figure 11.2. The subset of units in the LOVE:ACTOR group representing features of MARY have relatively strong connections to the subset of units in the LOVE:OBJECT group representing features of JOHN. These units and connections are highlighted in the figure. There are also sets of units representing features of MARY and JOHN independent of their being ACTORs in particular events. These units, including role-value combinations such as HAIR-COLOR=RED, would also participate in the representation of other people.

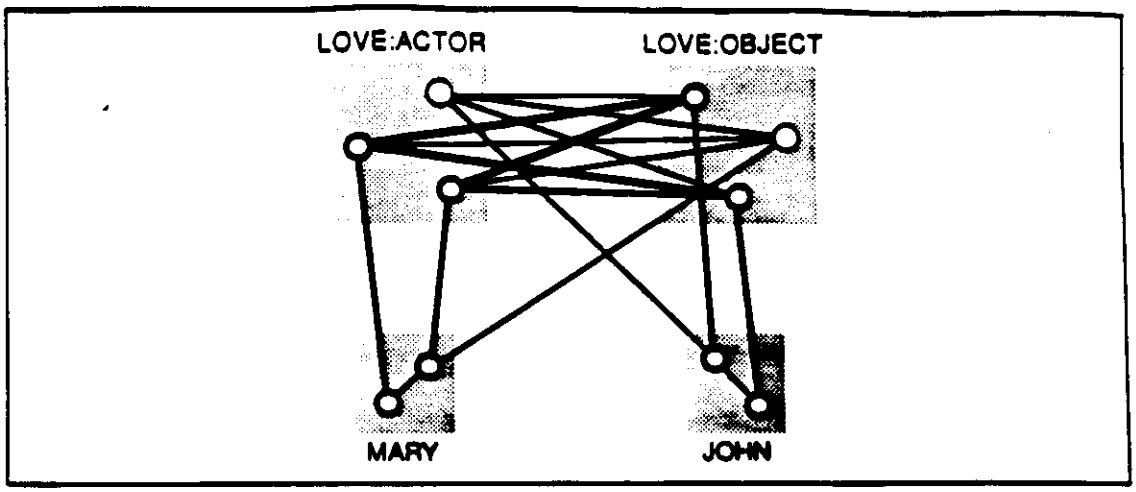

Figure 11.2: Distributed Representation of a Fact

Note that, as in other distributed approaches, there is no need for a schema head node or a set of units representing the head node. A schema or a schema instantiation here would just consist of the set of units representing the role-value pairs that characterize the schema and the relatively strong connections linking these units.

The distributed analogue of the CLM network would exhibit some special properties. For example, there would be nothing corresponding to the all-or-none selection process that is implemented by the firing of nodes in the localized model. For a concept to be selected in the distributed version, a significant subset of the nodes representing it would need to fire, but there would be no threshold in the sense that there is for the individual concept nodes in the localized model. An advantage of this approach is that it permits a concept to activate associated concepts even when it has not been selected. This property is necessary to handle some speech errors involving anticipatory effects.

Phonological exchange errors are of this type, for example, *flow snurries* for *snow flurries*. The effect is apparently due to the priming of a component of the second word before the first word is produced. In the example, there must be priming on the role for the initial consonant cluster (/fl/) in the second noun (*flurries*) when the first noun (*snow*) is to be produced. However, this priming can take place only if the role for the second noun itself has already fired. If it had fired, then the word *flurries* would have been produced intact before the word *snow*. The problem is that activation cannot spread along a path of nodes unless each fires in succession. There is no activation leakage. Within the CLM model, the only way to get exchange errors is to posit extra, redundant connections which skip over intermediate nodes, for example, one from the head node of the *SNOW-FLURRY entry to the consonant cluster node /FL/. In this case, two intermediate role nodes would be passed over by the new connection.

In the distributed alternative, on the other hand, some activation leakage would be the norm. Figure 11.3 shows two sets of units corresponding to localized nodes. The firing of node A in the localized network is represented in the distributed network by the firing of some threshold number of units in the A set. But even before this threshold is attained, some activation can spread to units representing parts of other localized nodes. In the figure, only one of the A units has fired, but this is enough for one of the B units to receive some activation. In this fashion, activation could even spread beyond B before A's "threshold" is reached. Thus the distributed version seems better able to handle anticipatory phenomena such as exchange errors.

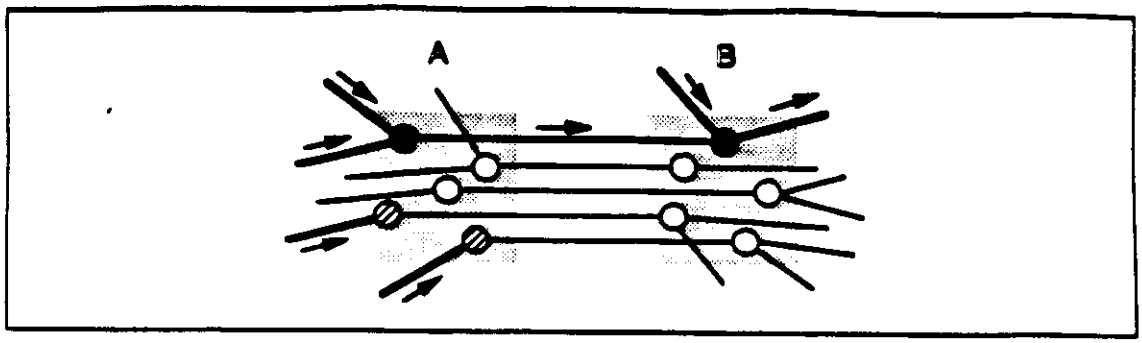

Figure 11.3: Distributed Alternative to Activation Spread Between Localized Nodes

The distributed network we are considering also offers a solution to the problem of crosstalk in parsing discussed in Section 8.2. Figure 11.4 illustrates the problem for the sentence *Mary loves John.*

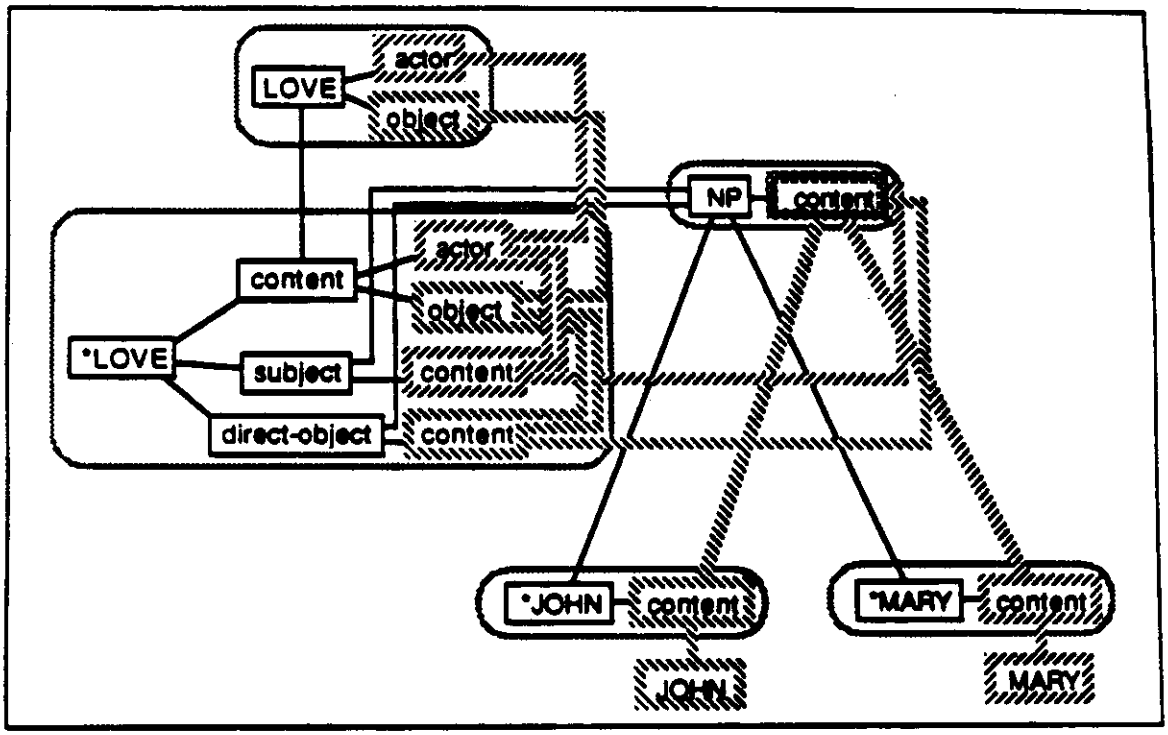

Figure 11.4: Crosstalk in Parsing

Once the sentence has been parsed, there are role binding paths for both MARY and JOHN. These are indicated in the figure by the hatched nodes and connections; those with upper-right-to-lower-left hashing pattern are on the MARY-ACTOR path, and those with the other pattern on the JOHN-OBJECT path. The problem is that NP:CONTENT is on both paths, so the bindings interfere with one another.

In the distributed alternative, on the other hand, NP:CONTENT would comprise a set of units, only a subset of which would participate in each of the two binding paths. Figure 11.5 illustrates how this arrangement solves the crosstalk problem. The two paths are indicated by the hashed units and connections.

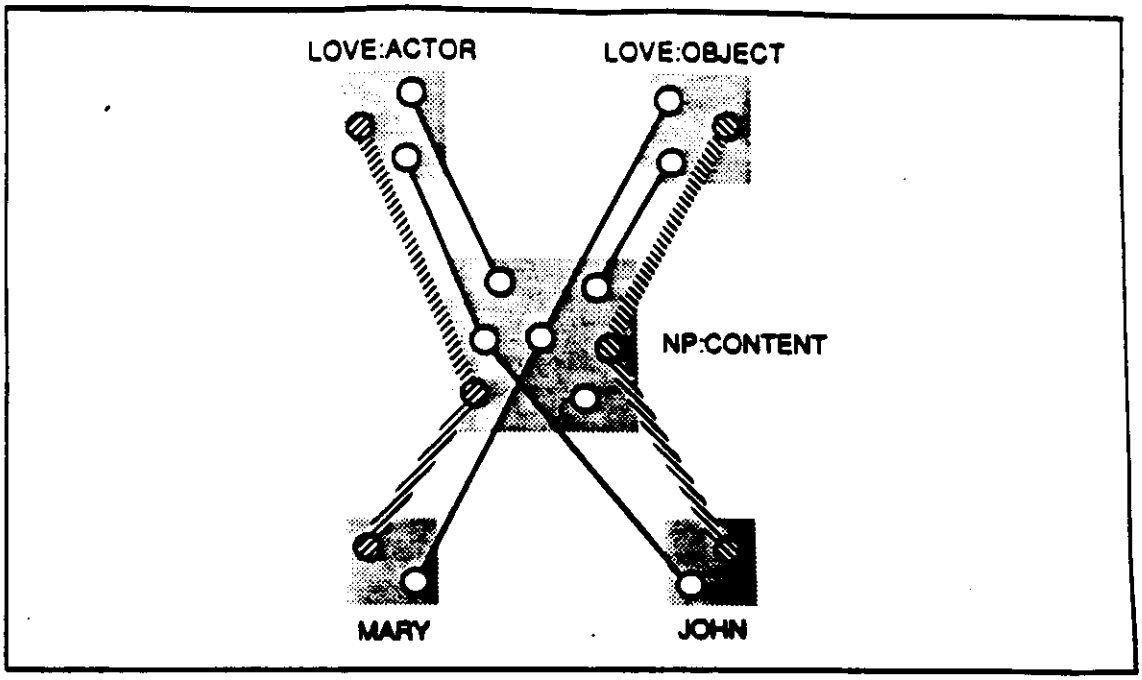

Figure 11.5: Distributed Representation for Handling Crosstalk

Note also that some of the intermediate nodes in the localized network are eliminated here. The CONTENT roles of the specific entries *JOHN and *MARY are represented by subsets of the NP:CONTENT set of units, and the merged nodes encoding the role binding information are replaced by connections joining a subset of the NP:CONTENT units to subsets of the LOVE:ACTOR and LOVE:OBJECT units.

The role bindings shown in Figure 11.5 are still temporary because they depend on the priming that remains on the shaded units. Once this activation decays, the bindings will become inaccessible, and for much of the information that the system receives, this short-term forgetting is an accurate reflection of human processing. However, people are also capable of storing new facts in long-term memory for later use. Without the addition of new nodes and connections, the localized model has no way of modelling this learning. How might it be dealt with in the distributed alternative?

Recall from Figure 11.2 that there are connections joining the units in LOVE:ACTOR to those in LOVE:OBJECT and joining the units in both of these groups to the MARY and JOHN groups. During the role binding process a subset of the nodes in each of the four groups will fire, and there will be connections joining these nodes and representing the fact that Mary loves John, as shown in Figure 9.2. A simple learning algorithm (Hebb, 1949) used in some connectionist models involves strengthening a connection whenever the units at both ends of it fire. If applied in this case, the strengthening would amount to an increase in the likelihood that this fact would be recalled later on. Thus the learning is a consequence of the role binding that takes place automatically during parsing.

One problem that must be solved before the distributed version of the CLM network can be implemented is that of what the individual units actually represent. What set of features, for example, would suffice to distinguish LOVE:ACTOR, say, from ACTOR in general?

11.1.2. Learning

Certainly the major weakness of the CLM model is the lack of a formal learning component. There are two reasons that the development of this component should be the major thrust of future work. First, the assignment of connections weights by hand is a tedious task, and it slows down the development of such models and renders them impractical for real-world applications. More importantly, any cognitive model should include an account of how the knowledge that is used got into memory in the first place.

One possible direction to take is to build a learning mechanism directly into the current localized model. This could operate in much the same way as back propagation in some distributed models (Rumelhart & McClelland, 1986). For each set of firing input nodes, it would require a teaching pattern representing the appropriate response for this input. Then if an output node fired when it should not (according to the teaching pattern), the weights on connections leading into it would be decremented by some factor, with this weight adjustment process propagating back as far as the input nodes.

Learning about new facts and objects would require a more elaborate mechanism involving the creation of new nodes and connections. This would not be difficult to implement. Consider, for example, the case where the sentence *Mary loves John* has is being parsed. As a role binding path is created, a permanent trace of it is recorded by copying the generic role node and connecting it to the value node and the generic role node. For the LOVE example, a new ACTOR instance would be created and connected to the LOVE:ACTOR (or ACT:ACTOR) and MARY nodes. That is, for each binding, one new node and two new connections would be required. In addition, once a node or is-a hierarchy of nodes is activated for the general fact type, a further node would be called for to represent the whole fact, say, LOVE23. This node would be connected to a parent, LOVE in this example, and to all of its existing role nodes.

While the algorithm would be fairly straightforward, there are problems with such a learning mechanism from the point of view of neurophysiological plausibility. Beyond infancy the number of neurons declines rather than increases (Kuffler & Nicholls, 1976), so we cannot assume a store of unused units to be recruited for the learning of new concepts. Similarly, while new connections are created, this clearly does not happen often enough for it to represent the learning that goes on whenever a new sentence is comprehended (Kuffler & Nicholls, 1976). If a learning model is ever to be implemented in brain-like hardware, then the system must make do with the units and, for the most part, with the connections that already exist.

Another possibility is to handle learning within the framework of the distributed version of the CLM model discussed in Section 11.1.1. This would have the clear advantage of avoiding the need for new nodes and connections as knowledge is added to the system. In addition, the research could rely on the considerable body of work done on learning in distributed paradigms.

In either case, the learning procedure could start as a simple aid in the tuning of the network. But later it could provide the basis for a long-term model of language acquisition. At this stage it is only possible to sketch out what would be involved. The system would begin with knowledge of concepts and some of the goals that are realized by linguistic means. Early acquisition would consist of GIs representing associations of goals specific to particular contexts with very specific patterns, e.g., *give me water*. Later the system would begin to associate patterns with concepts, for example, the expression *take a bath* with the concept of bath-taking. That is, the first GUs would be acquired. In both cases, acquisition would proceed from more specific patterns and contexts to more general patterns and contexts. Thus syntax would arise out of the learning of lexical patterns (Peters, 1983).

Most linguistic knowledge would be acquired as a side-effect of the comprehension process. Utterances would be presented to the system together with their contexts, and when unfamiliar words or structures appeared, the system would often be able to decode them through the use of contextual information. Generation goals would also be provided as input, and these would sometimes stretch the linguistic capacity of the system in such a way that it is forced to generalize existing knowledge. In this way, generation would also result in learning, though the generalizations would not always be appropriate.

Within the context of this general model, two specific directions suggest themselves. One is to attempt to simulate the development of pidgins by continually presenting the system with communicative needs, beyond its capacity, in the second language. Another is to test the effects of various types of input on the development of the different types of associations discussed in Chapter 6.

11.1.3. Coverage of Linguistic Phenomena

In terms of the actual variety of forms that is handled, the model falls short of existing generation systems. The emphasis in this project has been on determining whether simple sentences could be generated within this new framework. As a result, many of the linguistic concerns associated with a typical generation project have been neglected. Only a small fragment of English and Japanese grammar is covered by the model. Simple clauses and noun phrases including adjective or prepositional phrase modifiers are all that the model currently generates. Aspects not adequately treated include reference, quantification, negation, coordination, and long-distance dependencies such as in English relative clauses, though there is no reason to believe that these areas of language cannot be accommodated within the model.

11.1.3.1. Reference

In the present version of the model, it is assumed that constituents such as SUBJECT and DIRECT-OBJECT are realized directly as NPs. The firing of the node for the referent of the NP leads to the selection of one or more appropriate GUs for the NP being generated.

But the formulation of NPs is clearly more involved than this, as the work of Appelt (1985) has shown. The problem for the speaker is arriving at a description, the set of semantic features of the referent which will be the basis for entry selection for the NP. Thus there needs to be a separate planning level above that of the actual entry selection. This division corresponds to what the system already has for clauses, the GI level, representing knowledge about which types of plans are associated with particular communicative goal types, and the (verb) GU level, which realizes GI plans. For NPs there would be a level of generalized references for knowledge about which types of plans are associated with goals to make the hearer aware of particular referents. Noun GUs would then realize the plan roles of generalized references.

Particularly complex are definite NPs, such as *Arnold*, *him*, and *the guy I introduced you to last night*, which are designed to get the hearer to identify the referent. Two basic problems are involved. One is that of determining which features of the referent are known to the hearer. It is not enough for the speaker to have a name associated with a human referent; she must also believe that the hearer knows that that is the name associated with that person. The second aspect of formulating a suitable referent description is that of determining what set of features is sufficient for the hearer to identify the referent. This often reduces to determining whether there are any reference competitors, that is, entities other than the referent which the hearer knows of and which match the description in the NP. In one context, *the woman* may be enough for the hearer to know who is being referred to. In another, *the other woman* may suffice. In a third context, *the woman that*

we met on the plane last week may be required to distinguish that woman from others the hearer might think of.

Handling these two key aspects of definite reference, hearer knowledge and referent competition, seems to require further additions to the basic CLM framework. It needs to be possible to indicate (for any relation represented in the network) whether that relation is known to a particular person and to make inferences about what someone knows when there is no explicit indication. One promising approach is to make use of connection "scoping" (Fahlman, 1979; Feldman & Ballard, 1982). A special scope connection would join a given connection to the node representing the knowledge base of a person, and the scoped connection could only be activated when the knowledge base node is active. Note that this mechanism requires that connections have their own "handle" so that other connections can be attached to them. Many of the details of this approach need to be worked out, in particular, how inferences could be made on the basis of knowledge that a whole category of people are assumed to have (Clark & Marshall, 1981). For example, we would like to be able to infer from the fact that a person lives in a particular town that he will understand a reference by name to the mayor of that town.

Referent competition is also tied up with the speaker's representation of what the hearer knows or has in mind. In deciding whether *the car* suffices for the hearer to know which car is being talked about, the speaker does not need to concern herself with all of the cars that she might think of, but only those which the hearer might think of (or which the hearer might expect the speaker to be referring to in the current context). Thus only certain of the set of possible competitors should be "turned on". Again some sort of scoping mechanism seems called for.

11.1.3.2. Syntax

The CLM model is currently limited in the variety of syntactic constructs it can generate. Structures involving long-distance dependencies, such as relative clauses; coordination; and recursion have not been implemented. The difficulty of handling certain types of structures in this model comes from two facts. First, nodes in the network have no memory other than their decaying activation levels. Once the NP node fires, for example, it does not remember from what direction it was activated. Second, structures are not instantiated during processing; this is what is known as a type-only model (Dell, 1985). For example, in the NP *my older sister's husband*, the same NP entry must do for both the embedded phrase *my older sister* as well as for the entire phrase. Thus the basic processing mechanism in this and similar models is significantly less powerful than those in other computational approaches. It is possible, however, that this weakness may turn out to be a better reflection of the kind of language that people are capable of processing. There are questions, for example, about the extent to which people can comprehend and generate recursive structures (McClelland & Kawamoto, 1986). One area of future work will be concerned with extending the model into uncovered areas of syntax. In this regard, it will be interesting to determine how well the model handles structures that are known to be difficult for people, such as center embeddings.

11.1.3.3. Figurative Language

With respect to the lexicon, a fruitful area in which to extend the model is that of figurative language. Network models of the lexicon have been shown to be effective ways of looking at metaphor and figurative language (Jacobs, 1985a; Makkai, 1972; Martin, 1986). In the context of the present study, metaphor is especially interesting because it is related to the issue of what gets transferred in second language generation. Kellerman (1978) has shown that second language speakers are sensitive to the degree of idiomaticity of expressions in their first language when they are trying to decide whether the expressions can be translated literally into their second language. It should be possible to

represent degrees of idiomaticity with varying weights on the connections from idiomatic entries to the higher-level entries from which they are composed. That is, the extent to which an expression like *take a shower* is viewed as idiomatic can be translated into the question of how strong the connection is from the *TAKE-SHOWER entry to the general *TAKE entry.

11.1.3.4. Morphology and Phonology

The CLM model currently treats words as the primitive units of linguistic form. However, it should be possible to extend the model into the areas of morphology and phonology using the same basic ideas. It is encouraging that other work within similar frameworks (Dell, 1986; Stemberger, 1985) has been relatively successful at modelling phonological and morphological processes, including typical errors

Figure 11.6 shows how the GU for *water* might be expanded to incorporate phonological information.

```
(*WATER NP
    (content WATER)
    (noun TWO-SYLLABLE-WORD
        (first-syllable SYLLABLE
            (onset /w/)
            (coda /ɔ/))
        (second-syllable SYLLABLE
            (onset /t/)
            (coda /ɚ/))))
```

Figure 11.6: GU for *water* Showing Phonological Features

The NOUN role points to the node for TWO-SYLLABLE-WORD and it has roles of its own for the two syllables it is composed of. Each syllable role in turn has roles for the segments of which it is composed, that is, the ONSET and CODA positions in the syllable.

11.1.4. Repairs

Speakers are often aware of errors soon after they have made them, and there are characteristic ways in which they correct themselves (Levelt, 1983). The CLM model currently has no way to recognize that an error has been produced, nor to correct itself if one is discovered.

Consider what might be involved in getting the model to recognize that it has produced a substitution error, such as the production of *salt* in place of *pepper*. When a GU such as *SALT is selected, its CONTENT role fires early on. But it is reasonable to assume that it would fire again in response to activation from the node representing the end of the phrase. The firing of the CONTENT role would in turn activate role-value node pairs representing the features of the concept associated with the entry. In this case, pairs such as COLOR and WHITE, CONSISTENCY and GRANULAR, and TASTE and SALTY would be expected to fire. But a subset of these features would conflict with features of the input notion, here the pepper to be passed. Precisely how the perception of this conflict would be implemented is not clear, but it is apparently at this point that a speaker becomes aware of an error like this.

11.1.5. Planning What to Say

The CLM model in its present form has little to offer concerning decisions that speakers make about what they will talk about. The notions of relevance and interestingness are very complex ones (Schank, 1979; Sperber & Wilson, 1986). Initial attempts at selecting relevant topics for generation through the use of spreading activation show promise, however (Fuenmayor, 1988; Hasida, Ishizaki, & Isahara, 1987).

Some way of representing the hearer's knowledge is the key to decisions about what aspects of a referent to mention, but is also involved at higher levels. Thus decisions about what facts to include in a narrative depend on what the hearer can be expected to know already about the participants and the setting. Even more important is what the speaker believes the hearer can infer, but this too is related to the kind of general knowledge the hearer is thought to have about comparable situations.

Knowledge about knowledge is special in the sense that it can potentially modify anything that is encoded in the network. As noted in the discussion of reference in Section 11.1.3.1, a possible way of accomplishing this is through the use of scoping connections attached to connections themselves. There is also the need to be able to represent two or more embedded layers of knowledge or beliefs. Thus a speaker seems to be at least capable of reasoning about what the hearer believes the speaker believes. There is considerable literature on this area (e.g., Appelt, 1985; Clark & Marshall, 1981; Sperber & Wilson, 1986). As with syntax, this is an area in which the relatively constrained approach adopted here may offer insights into the limits of the human capacity to deal with embedding and recursion of belief contexts.

11.1.6. Conscious Knowledge

An important distinction made in some approaches to second language acquisition (e.g., Krashen, 1981) is that between formally learned knowledge and knowledge acquired through experience. The learner is conscious of the former type and can sometimes make use of it during generation when there is time to monitor the output, but it tends to be relatively ineffective. The latter type includes nearly all of what people know of their first languages and of second languages if they have been learned in natural settings. A major question has been whether formal knowledge can through practice become indistinguishable from the naturally acquired type of knowledge.

Within the CLM approach, the units of linguistic knowledge do not really represent knowledge about language at all; rather they encode knowledge about how certain kinds of actions accomplish certain goals for speakers. There are no explicit rules referring to linguistic objects such as words, morphemes, or phrases. Formal knowledge, on the other hand, looks more like the rules of a transformational grammar. It encodes procedures such as "if you want to produce a NOUN, then check to see whether it is a COUNTABLE-NOUN and whether it refers to more than one thing; if so, make sure the noun has the PLURAL form". In the CLM framework, such rules can be encoded as schemas like the rest of knowledge in memory. However, when so encoded, they would differ strikingly from the GUs and GIs in linguistic memory. In particular, they must operate on the output of generation as it is currently handled in the model. That is, they would have to check words that are about to be uttered.

11.1.7. Translation

Machine translation researchers have generally shown little interest in the behavior of human translators. Thus this is an area that is ripe for the insights to be gained from looking at what people do. An understanding of human translation, at the level that would be required, involves questions of the type addressed in Chapter 7: How are units of

knowledge for the different languages related in memory, and in what way do the units interact during generation?

A general contribution that might be made is the development of a flexible approach to machine translation. Existing systems tend to divide into two categories: those employing some form of direct transfer between the lexical items or structures in the languages (e.g., Nagao & Tsujii, 1986) and those which parse all input into an interlingua from which the output is generated (e.g., Carbonell & Tomita, 1986) Certainly human translation has both types of processes. While it will often be necessary to achieve a deep understanding of the input in order to translate it, there will also be shortcuts of the type described in Chapter 6, which can speed up processing by providing direct associations between units of the languages. A machine translation system, based on human translation processes, would need to be able to find the cross-linguistic associations if they exist but be ready to parse the input into a non-linguistic conceptual form for generation when no such associations are available. A spreading activation mechanism like that in the CLM model would provide a way of performing this kind of processing without the need for special procedures to guide the system. That is, the entire process of translation would be a consequence of activation spreading through the network encoding the system's knowledge of the source and target languages and the content domain. Of course, significant additions need to be made to the model before it could be applied to machine translation, particularly in the area of parsing.

11.2. Summary and Implications

The subfields of cognitive science are currently in the throes of what some consider a revolution. Radical connectionists (e.g., Rumelhart & McClelland, 1986; Smolensky, 1988) suggest that their models will one day replace all or most of what has been done by the practitioners of symbolic cognitive science. Proponents of symbolic models counter that the new approach is really just a revival of discredited associationism, that it will never be able to handle fundamental aspects of cognition such as recursion and the type-token distinction, and that at best it will be viewed as a way of implementing symbolic notions (Fodor & Pylyshyn, 1988; Pinker & Prince, 1988).

Those in the middle believe that both sides have something to offer and that a fruitful approach is to attempt a synthesis of the two types of models (e.g., Feldman, 1986; Waltz & Pollack, 1985). The model presented in this dissertation is an example of such a synthesis. It starts with conceptual and linguistic categories based on those developed in symbolic models, but it treats them as nodes in a network of neuron-like processing units. The result has advantages over models at both extremes. It performs a more complex task than any distributed connectionist model, and it does this in a more human-like fashion than any symbolic model that performs the task.

In addition to features that all generation models must exhibit (e.g., the selection of lexical items on the basis of input concepts and the outputting of words and phrases in an appropriate order) the CLM approach models other phenomena characteristic of human generation that have been generally ignored in other work. These include (1) the direct access of linguistic units on the basis of features of their meanings, (2) robustness with respect to incomplete input or linguistic gaps, (3) accommodation of context-driven as well as goal-driven processing, (4) flexibility in sequencing, (5) transfer effects in multilingual contexts, and (6) substitution and exchange errors. In addition, the same knowledge used in generation supports parsing.

The simplicity of the model can be appreciated best in terms of what it does not make use of. There are no special data structures for particular types of linguistic units and no global variables such as CURRENT-HEARER or CURRENT-REFERENT to be bound during processing. There are no rules to be applied in a fixed sequence by a central control mechanism. No temporary phrase structures are built during processing. There is in fact no generation algorithm whatsoever. The model accomplishes what it does using an approach in which all knowledge resides in the pattern of connections among simple threshold processing units. Generation is a side effect of a mechanism (spreading activation) that is neutral with respect to generation, i.e., it can be (and is) used for non-linguistic processing.

Likewise, the features that the model exhibits are not a consequence of a set of domain-specific rules. Rather they emerge out the connectivity of the network and the way activation spreads over it. The system is robust with respect to incomplete input or gaps in linguistic knowledge simply because of the way in which nodes respond to the sum of their inputs. Variability in sequencing is achieved because output ordering depends, not on a set of all-or-none rules, but on the total weight that constituent nodes receive from syntactic and semantic sources. The tendency for L2 speakers to make generalizations in the L2 that resemble those in the L1 is based, not on a specific language acquisition rule, but on the overlap between concepts that is a natural property of the CLM network.

Two problems that arise in connectionist models concern the association of temporary values with roles such as HEARER and SUBJECT and the sequencing of output forms. Both are handled here in novel ways. Role binding is implemented in terms of the firing of nodes representing a role and its value in close temporal proximity. The result of this process is a path of primed nodes which makes the binding temporarily accessible from either end of the path. The problem of crosstalk in role binding is avoided by forcing certain role nodes to fire in sequence rather than simultaneously. Sequential output is achieved by (1) using winner-take-all networks to prevent more than one constituent from being produced at a time and (2) introducing special end nodes whose firing represents the completion of a constituent or phrase.

The model has implications for theories of linguistic knowledge and processing and for general theories of human action. As in speech act theories, a basic distinction is made between knowledge about illocutions, that is, how language satisfies communicative goals, and knowledge about utterances, that is, how language maps onto concepts. However, the units within each of these components do not correspond neatly to any of the traditional categories. They represent generalizations that associate features of a pattern with semantic or pragmatic features. Thus syntax and morphology are bound together intimately with pragmatics and semantics.

Another traditional distinction (which serves no function in the model) is that between lexicon and grammar. Both syntactic and lexical patterns make up the hierarchy of GUs. Syntactic patterns are those at the more general end of the hierarchy, while lexical patterns are those at the more specific end, but there is no clear line separating them. This relationship, of lexicon to grammar, is assumed to reflect what goes on in acquisition, a process that begins with the highly specific patterns at the bottom of the hierarchy and yields more and more general patterns as knowledge of specific patterns accumulates (Peters, 1983; Wong Fillmore, 1977). Thus the pattern [PERSON *BREAK* GLASS] generalizes to the pattern [ANIMATE-THING *BREAK* BREAKABLE-OBJECT], which later generalizes to the pattern [ANIMATE-THING *TRANSITIVE-VERB* PHYSICAL-OBJECT].

The integration of lexicon and grammar in a single network also has important implications for processing. A more general entry can control the sequencing of elements appearing in a more specific entry because corresponding elements in the two entries are

joined directly. Similarly, when a relatively specific entry is selected, the system automatically has access to the features of all of the more general entries it is associated with.

Language generation in the model is treated as a form of action. Speakers have goals which they satisfy by producing utterances that match up with the contextual and semantic features of the goals. Utterances are just acts that are realized as sequences of subacts consisting either of words or further utterances. Thus, in this model, there is no separate language faculty. Linguistic behavior differs from other types of human action in degree rather than in kind.

Connectionist models and other approaches making use of spreading activation and marker passing have been applied mainly to tasks involving perception or recognition. What this dissertation has demonstrated is that one such approach can also be applied successfully and fruitfully to a complex type of action. The implication is that other tasks involving planning and acting may also be modelled similarly, as a process of the simultaneous satisfaction of multiple constraints.

The challenge now for cognitive science is to bridge the gap between the symbolic and the neural. On the one hand, cognitive models will have to be capable of achieving all of the behavior that the symbolic approaches do, for example, in the area of complex syntactic structures. On the other, the models will need to "scale down", that is, to map onto the level of highly distributed approaches, with their greater neurophysiological plausibility and straightforward learning algorithms. The model presented in this thesis is a promising step in this direction.

References

Adjémian, C. (1983). The transferability of lexical properties. In S. Gass & L. Selinker (Eds.), (pp. 250-268)..

Agre, P. E., & Chapman, D. (1988). Indexicality and the binding problem. *Proceedings of the AAAI 1988 Spring Symposium Series.*

Andersen, R. W. (1983). Transfer to somewhere. In S. Gass & L. Selinker (Eds.), (pp. 177-201).

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1985). *Cognitive psychology and its implications*. (2nd ed.). New York: W. H. Freeman.

Anderson, S. R., & Keenan, E. L. (1985). Deixis. In T. Shopen (Ed.),*Language typology and syntactic description: Vol. 3. Grammatical categories and the lexicon* (pp. 259-308). Cambridge: Cambridge University Press.

Appelt, D. E. (1985). *Planning English sentences*. Cambridge: Cambridge University Press.

Arbib, M. A., Conklin, E. J., & Hill, J. C. (1987). *From schema theory to language*. New York: Oxford University Press.

Austin, J. L. (1962). *How to do things with words*. Oxford: Oxford University Press.

Bickerton, D. (1981). *The roots of language*. Ann Arbor: Karoma.

Bock, J. K. (1982). Toward a cognitive psychology of syntax: Information processing contributions to sentence formulation. *Psychological Review, 89*, 1-47.

Brachman, R. J., & Schmolze, J. (1985). An overview of the KL-ONE representation system. *Cognitive Science, 9*, 171-216.

Brandt Corstius, H. (1978). *Computer-taalkunde*. Muiderberg, Netherlands: Coutinho.

Brown, P., & Levinson, S. (1978). Universals in language usage: Politeness phenomena. In E. N. Goody (Ed.), *Questions and politeness: Strategies in social interaction* (pp. 56-289). Cambridge: Cambridge University Press.

Carbonell, J., & Tomita, M. (1986). Knowledge-based machine translation: The CMU approach. In S. Nirenburg (Ed.), *Machine translation*. Cambridge: Cambridge University Press.

Charniak, E., Riesbeck, C., McDermott, D., & Meehan, J. R. (1987). *Artificial intelligence programming* (2nd ed.). Hillsdale, NJ: Erlbaum.

Clark, H. H., & Clark, E. V. (1977). *Psychology and language: An introduction to psycholinguistics*. New York: Harcourt Brace Jovanovich.

Clark, H. H., & Marshall, C. (1981). Definite reference and mutual knowledge. In A. K. Joshi, B. L. Webber, & I. A. Sag (Eds.), *Elements of discourse understanding* (pp. 10-63). Cambridge: Cambridge University Press.

Corder, S. P. (1983). A role for the mother tongue. In S. M. Gass & L. Selinker (Eds.), (pp. 85-97).

Cornejo, R. J. (1973). The acquisition of lexicon in the Spanish of bilingual children. In R. P. Turner (Ed.), *Bilingualism in the Southwest*. Tucson, AZ: University of Arizona Press.

Dalrymple-Alford, E. C. (1984). Bilingual retrieval from semantic memory. *Current Psychological Research and Reviews, 3*(2), 3-13.

Dell, G. S. (1985). Positive feedback in hierarchical connectionist models: Applications to language production. *Cognitive Science 9*, 3-23.

Dell, G. S. (1986). A spreading-activation theory of retrieval in sentence production. *Psychological Review, 93*, 283-321.

Dell, G. S., & Reich, P. A. (1980). Toward a unified model of slips of the tongue. In V. A. Fromkin (Ed.), *Errors in linguistic performance*. London: Academic.

Dell, G. S., & Reich, P. A. (1981). Stages in sentence production: An analysis of speech error data. *Journal of Verbal Learning and Verbal Behavior, 20*, 611-629.

Dyer, M. G. (1983). *In-depth understanding: A computer model of integrated processing for narrative comprehension*. Cambridge, MA: MIT Press.

Fahlman, S. E. (1979). *NETL: A system for representing and using real-world knowledge*. Cambridge, MA: MIT Press.

Feldman, J. A. (1986). *Neural representation of conceptual knowledge* (Technical Report TR189). Rochester, NY: University of Rochester, Department of Computer Science.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science, 6*, 205-254.

Fillmore, C. G. (1968). The case for case. In E. Bach and R. Harms (Eds.), *Universals of linguistic theory* (pp. 1-90). New York: Holt, Rinehart, and Winston.

Fillmore, C. G. (1979). Innocence: A second idealization for linguistics. *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistic Society*, 63-76.

Fillmore, C. G., Kay, P., & O'Connor, M. C. (1986). Regularity and idiomaticity in grammatical constructions: The case of *let alone*. Unpublished manuscript.

Fodor, J. A. (1983). *The modularity of mind*. Cambridge, MA: MIT Press.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition, 28*, 1-71.

Foley, W. A., & Van Valin, R. D., Jr. (1984). Functional syntax and universal grammar. Cambridge: Cambridge University Press.

Fuenmayor, M. E. (1988). *A connectionist approach for modelling creative planning in story invention*. Unpublished manuscript, Computer Science Department, University of California, Los Angeles.

Gass, S., & Selinker, L. (Eds.). (1983). *Language transfer in language learning*. Hillsdale, NJ: Erlbaum.

Gibbs, R. W., Jr. (1986). What makes some indirect speech acts conventional? *Journal of Memory and Language, 25*, 181-196.

Hale, K. L. (1981). *On the position of Warlbiri in a typology of the base*. Bloomington, IN: Indiana University Linguistics Club.

Hasida, K., Ishizaki, S., & Isahara, H. (1987). A connectionist approach to the generation of abstracts. In Kempen (Ed.), *Natural language generation*. (pp. 149-156). Dordrecht: Martinus Nijhoff.

Hatch, E. M. (1976). Studies in language switching and mixing. In W. C. McCormack & S. A. Wurm (Eds.), *Language and man. Anthropological issues* (pp. 201-214). The Hague: Mouton.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel Distributed Processing. Explorations in the microstructures of cognition: Vol. 1: Foundations* (pp. 77-109). Cambridge, MA: MIT Press.

Hudson, R. (1984). *Word grammar*. Oxford: Blackwell.

Hummel, K. M. (1986). Memory for bilingual prose. In J. Vaid (Ed.), *Language processing in bilinguals: Psycholinguistic and neuropsychological perspectives* (pp. 47-64). Hillsdale, NJ: Lawrence Erlbaum.

Ijaz, I. H. (1986). Linguistic and cognitive determinants of lexical acquisition in a second language. *Language Learning, 36*, 401-451.

Jacobs, P. S. (1985a). A knowledge-based approach to language generation (Technical Report UCB/CSD 86/254). Berkeley, CA: University of California at Berkeley, Computer Science Division.

Jacobs, P. S. (1985b). PHRED: A generator for natural language interfaces. *Computational Linguistics, 11*, 219-242.

Joshi, A. K. (1985). Processing of sentences with intrasentential code switching. In D. R. Dowty, L. Karttunen, & A. M. Zwicky (Eds.), *Natural language parsing: Psychological, computational, and theoretical perspectives* (pp. 190-205). Cambridge: Cambridge University Press.

Kalita, J., & Shastri, L. (1987). Generation of simple sentences in English using a connectionist model of computation. *Proceedings of the Ninth Meeting of the Cognitive Science Society*, 555-565.

Kellerman, E. (1977). Towards a characterisation of the strategy of transfer in second language learning. *Interlanguage Studies Bulletin, 2*, 58-145.

Kellerman, E. (1978). Giving learners a break: Native speaker intuitions as a source of predictions about transferability. *Working Papers on Bilingualism, 15*, 59-92.

Kempen, G., & Hoenkamp, E. (1987). An incremental procedural grammar for sentence formulation. *Cognitive Science, 11*, 201-258.

Kirsner, K. (1986). Lexical function: Is a bilingual account necessary? In J. Vaid (Ed.), *Language processing in bilinguals: Psycholinguistic and neuropsychological perspectives* (pp. 21-45). Hillsdale, NJ: Lawrence Erlbaum.

Kolodner, J. L. (1984). *Retrieval and organizational strategies in conceptual memory*. Hillsdale, NJ: Lawrence Erlbaum.

Krashen, S. (1981). *Second language acquisition and second language learning*. London: Pergamon.

Kuffler, S. W., & Nicholls, J. G. (1976). *From neuron to brain: A cellular approach to the function of the nervous system*. Sunderland, MA: Sinauer.

Langacker, R. W. (1987a). *Foundations of cognitive grammar (Vol. 1)*. Stanford, CA: Stanford University Press.

Levelt, W. J. M. (1983). Monitoring and self-repair in speech. *Cognition, 14*, 41-104.

MacKay, D. G. (1987). *The organization of perception and action: A theory for language and other skills*. New York: Springer-Verlag.

Makkai, A. (1972). *Idiom structure in English*. The Hague: Mouton.

Mann, W. C., & Matthiessen, C. M. I. M. (1983). Nigel: A systemic grammar for text generation (Research Report RR-83-105). Los Angeles: USC/Information Sciences Institute.

Martin, J. H. (1986). Views from a hill. *Proceedings of the Eighth Annual Conference of the Cognitive Society*, 728-733.

McClelland, J. L., & Kawamoto, A. H. (1986). Mechanisms of sentence processing: Assigning roles to constituents of sentences. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), (pp. 272-325)

McClelland, J. L., Rumelhart, D. E., & the PDP Research Group (Eds.). (1986). *Parallel Distributed Processing. Explorations in the microstructures of cognition: Vol. 2: Psychological and biological models*. Cambridge, MA: MIT Press.

Minsky, M. (1986). *The society of mind*. New York: Simon and Schuster.

Nagao, M., & Tsujii, J. (1986). The transfer phase of the Mu machine translation system. *Proceedings of the 11th International Conference on Computational Linguistics*, 97-103.

Newell, A. (1980). Physical symbol systems. *Cognitive Science, 4*, 135-183.

Nishimura, M. (1986). Intrasentential code-switching: The case of language assignment. In J. Vaid (Ed.), *Language processing in bilinguals: Psycholinguistic and neuropsychological perspectives* (pp. 123-143). Hillsdale, NJ: Lawrence Erlbaum.

Pawley, A., & Syder, F. H. (1983). Two puzzles for linguistic theory: Nativelike selection and nativelike fluency. In J. C. Richards & R. W. Schmidt (Eds.), *Language and communication*. London: Longman.

Pazzani, M. J. (1988). Learning causal relationships: An integration of empirical and explanation-based learning methods (Technical Report UCLA-AI-88-10). Los Angeles: University of California, Computer Science Department.

Pereira, F., & Warren, D. (1980). Definite clause grammars for language analysis--a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence, 13*, 231-278.

Perrault, C. R., & Allen, J. F. (1980). A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics, 6*, 167-182.

Peters, A. M. (1983). *The units of language acquisition*. Cambridge: Cambridge University Press.

Pfaff, C. W. (1979). Constraints on language mixing: Intrasentential code-switching and borrowing in Spanish/English. *Language, 55*, 291-318.

Pinker, S., & Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition, 28*, 73-193.

Quillian, M. R. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic information processing*. Cambridge, MA: MIT Press.

Rosch, E. (1978). Principles of categorization. In E. Rosch & B. B. Lloyd (Eds.), *Cognition and categorization*. Hillsdale, NJ: Erlbaum.

Rumelhart, D. E. (1980). Schemata: The building blocks of cognition. In R. Spiro, B. Bruce, & W. Brewer (Eds.), *Theoretical issues in reading comprehension* (pp. 33-58). Hillsdale, NJ: Erlbaum.

Rumelhart, D. E., & McClelland, J. L. (1986). On learning the past tenses of English verbs. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), (pp. 216-271).

Schank, R. C. (1979). Interestingness: Controlling inferences. *Artificial Intelligence, 12*.

Schank, R. C., & Abelson, R. (1977). *Scripts, plans, goals and understanding*. Hillsdale, NJ: Erlbaum.

Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language*. Cambridge: Cambridge University Press.

Sharwood Smith, M. A. (1979). Strategies, language transfer and the simulation of the language learner's mental operations. *Language Learning, 29*, 345-361.

Sharwood Smith, M. A. (1983). On first language loss in the second language acquirer: Problems of transfer. In S. M. Gass & L. Selinker (Eds.), (pp. 222-231).

Shastri, L. (1987). A connectionist encoding of semantic networks. *Proceedings of the Ninth Meeting of the Cognitive Science Society*, 143-154.

Slade, S. (1987). *The T programming language: A dialect of LISP*. Englewood Cliffs, NJ: Prentice-Hall.

Smolensky, P. (forthcoming). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11*.

Sperber, D., & Wilson, D. (1986). *Relevance: Communication and cognition*. Cambridge, MA: Harvard University Press.

Stemberger, J. P. (1985). An interactive activation model of language production. In A. W. Ellis (Ed.), *Progress in the psychology of language: Vol. 1*. London: Erlbaum.

Stemberger, J. P., & MacWhinney, B. (1986). Frequency and the lexical storage of regularly inflected forms. *Memory and Cognition, 14*, 17-26.

Uyekubo, A. (1972). *Language switching of Japanese-English bilinguals*. Unpublished master's thesis, University of California, Los Angeles.

Waltz, D. L., & Pollack, J. B. (1985). Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science, 9*, 51-74.

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Reading, MA: Addison-Wesley.

Wilensky, R., & Arens, Y. (1980). *PHRAN--A Knowledge-based Approach to Natural Language Analysis* (Memorandum UCB/ERL M80/34). Berkeley, CA: University of California, Electronics Research Laboratory.

Woods, W. A. (1980). Cascaded ATN grammars. *American Journal of Computational Linguistics, 6*, 1-12.

Zernik, U. (1987). *Strategies in language acquisition: Learning phrases from examples in context* (Technical Report UCLA-AI-87-1). Los Angeles: University of California, Computer Science Department.

Zernik, U., & Dyer, M. G. (forthcoming). The self-extending phrasal lexicon. *Computational Linguistics*.

Zwicky, A. M. (1974). Hey, whatsyourname! *Papers from the Tenth Regional Meeting of the Chicago Linguistic Society*, 787-801.

Appendix:

CHIE: the Program

CHIE currently resides in the directory `/ucla/ai_research/gasser/Chie` on the Apollo COGNET network at UCLA (`cognet.ucla.edu` on the Arpanet). The program CHIE consists of three components: a set of functions for creating memory nodes and connections, located in the file `net.t`; procedures for spreading activation through the network, located in the file `propagation.t`; and sample memory files which build the network needed for generation of certain sentences, located in the directory `Memory`. The file `misc.t` contains miscellaneous help functions.

What follows is a listing of the program, including the memory files needed for generating the sentence *could you take the empty box to the garage?* (see the trace in Chapter 10). Each file begins with a `herald` clause.

```
(herald net)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;    THIS FILE CONTAINS THE FUNCTIONS USED IN CREATING NETWORK     ;;;
;;;    NODES AND CONNECTIONS.                                          ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                       N O D E   STRUCTURE TYPE                       ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Creates the structure type for network nodes.  Defines the
;; function MAKE-NODE, which creates new nodes (but it is MAKE-NODE!
;; that is actually used).

(define-structure-type node
```

```
connections ; A list of CONNECTION-LISTS, each representing an
            ; excitatory or inhibitory connection to another node.
            ; Each connection-list is of the form
            ;
            ;   (DESTINATION WEIGHT CONNECTION-TYPE DECAY . SPEED)
            ;
            ; DESTINATION is the node at the other end of the
            ;    connection.
            ; WEIGHT is a number between -1 and 1.
            ; CONNECTION-TYPE is PARENT, CHILD, HEAD, ROLE, AFTER,
            ;    BEFORE, EQUIV, INHIB, or PRIME.  The connection-
            ;    type has no function in processing.
            ; DECAY is the rate at which activation coming into
            ;    the destination node along the connection decays
            ;    on each time step.
            ; SPEED is an optional argument.  If present, the
            ;    connection is "fast"; that is, it can be
            ;    traversed in 1/4 of a time step (a "time
            ;    increment").

activation  ; A list of the following form
            ;
            ;   (ACTIVATION-KEY  ACTIVATION-LIST*)
            ;
            ; ACTIVATION-KEY is
            ;   a negative integer if the node is in its
            ;      refractory period
            ;   a positive integer if the node is a WTA hub which is
            ;        waiting to time item
            ;   NIL otherwise
            ; Each ACTIVATION-LIST represents activation remaining
            ;   from what came in on a single connection.
            ;   ACTIVATION-LIST has the following form:
            ;
            ;   (ACTIVATION DECAY SOURCE)
            ;
            ; ACTIVATION is the current value of the activation
            ;   due to this connection.
            ; DECAY is the decay rate for this connection.
            ; SOURCE is the source of the activation, that is, the
            ; node
            ; at the other end of the connection.

attribs     ; An association list specifying properties of the node,
            ; including properties used in processing:
            ; REFRACTORY,
            ;    a negative integer, representing the numbe of time
            ;    steps the node remains inhibited following firing
            ; RECOVERY,
            ;    the amount of activation remaining on the node
            ;    following its refractory period
            ; DECAY,
            ;    the decay rate for the recovery activation on the
            ;    node
            ; LIFE (for WTA hub nodes only),
            ;    the number of time steps a hub node waits before
            ;    timing
            ;    out,
```

```
                ; and properties which have no role in processing: MAP,
                ; INDIV, SEQ, START, END, and SEQ-END (the associated end
                ; for a start node), and SEQ-START (the associated start
                ; for an end node).

   name         ; An atom associated with the node to make the node
                ; accessible for the network building functions.
                ; Names have no role in processing.

   (((print self stream)                ; Sees to it that a node's
     (print (node-name self) stream)    ; name will also be its
     repl-wont-print)))                 ; print name.

;; Initializes the components of nodes to NIL.

(let ((m (stype-master node-stype)))
  (set (node-connections m) nil)
  (set (node-activation m) nil)
  (set (node-attribs m) nil))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;    CREATING NODES AND CONNECTIONS USING  S C H E M A !  AND    ;;
;;                      S E Q - S C H E M A !.                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; The functions SCHEMA! and SEQ-SCHEMA! are used to create subnetworks
;; consisting of a "head" and a "role" hierarchy, along with the
;; connections joining them to each other and to nodes in other
;; subnetworks.  (Note that
;; all functions which add or delete nodes or connections are followed
;; by "!".)  Calls to SCHEMA! and SEQ-SCHEMA! are of the form:
;;
;;   (FUNCTION  NAME  PARENT  .  OPTIONS)
;;
;; FUNCTION is either SCHEMA! or SEQ-SCHEMA!
;; NAME is a name for the head of the schema.  This is checked first to
;;    see whether it is already assigned to an existing node.
;; PARENT is the name of a node which the schema head is to be connected
;;    to.
;; OPTIONS is an optional association list with sublists specifying
;;    1. attributes of the head
;;    2. connections joining the head to other nodes
;;    3. roles associated with the head.
;;
;; Examples:
;;
;;   (SCHEMA! 'BOX 'PHYS-OBJ)
;;
;;      This creates a new node BOX and connects it to its parent
;;      PHYS-OBJ.
;;      Two connections are created, one from BOX to PHYS-OBJ, another
;;      from PHYS-OBJ to BOX.
;;
;;   (SCHEMA! 'MARY 'WOMAN
;;        '(INDIV)
;;        '(REFRACTORY -4))
;;
```

```
;;      This creates a node MARY with the attribute INDIV and a
;;      refractory period of 4 time steps.
;;
;;   (SEQ-SCHEMA! '*DOG 'NP)
;;
;;      This creates a start node named *DOG and a node named *DOG_END.
;;      *DOG is connected up to NP and *DOG_END to NP_END.  (Of course
;;      the schema has nothing in it.  We will see a complete *DOG
;;      schema below.)
;;
;; The functions are recursive in that OPTION sublists specifying
;; connections or roles may have their owner OPTION arguments specifying
;; further connections, roles, or attributes.
;;
;; SEQ-SCHEMA! differs from SCHEMA! in that a start and end node are
;; created for the schema head.
;;
;;
;; CREATING ROLES
;;
;; Roles are created within calls to SCHEMA! and SEQ-SCHEMA! using
;; option sublists beginning with the keywords ROLE, TYPE-ROLE, MAP,
;; SEQ-ROLE, and SEQ-MAP.
;;
;; For the purposes of creating connections and assigning default
;; parameters, two sorts of distinctions are made for roles.  First,
;; a role is either a top-level role, e.g., the ACTOR in the ACT schema
;; or the NATIONALITY in the PERSON schema, or a "map", that is a
;; copy of a high-level role in a schema lower in the is-a hierarchy,
;; e.g., the ACTOR in the PHYSICAL-TRANSFER schema or the NATIONALITY
;; in the ILLEGAL-ALIEN schema.  Second, a role is either a type, that
;; is, a prototype for a set of roles within the schema, e.g., ORGAN
;; in the ANIMAL schema or PLAYER in the GAME schema, or an individual,
;; e.g., HEART in the ANIMAL schema or NET in the VOLLEYBALL-GAME
;; schema. Very little is currently made of the type-individual
;; distinction for roles, but it is felt that the two kinds of nodes may
;; require different default values for node parameters.  Both
;; distinctions are based on those made by Fahlman (1979).
;;
;; Top-level roles are created using sublists beginning with ROLE,
;; TYPE-ROLE (if they are types), and SEQ-ROLE (if they are
;; constituents).  The sublists have the following form:
;;
;;    (ROLE-TYPE NAME PARENT . OPTIONS)
;;
;; ROLE-TYPE is either ROLE, TYPE-ROLE (if it is a type), or SEQ-ROLE
;;    (if it is the constituent of sequence).  SEQ-ROLE creates a start
;;    and an end node and connects them in the appropriate ways to the
;;    start and ends nodes for the head of the schema. (The individual-
;;    type distinction is not made for sequence roles.)
;; NAME is the name of the role.  This is checked to determine whether
;;    it is already assigned to another node.
;; PARENT is a parent node which the role is to be connected to.  The
;;    for a top-level role is always a type-role.  For example, the
;;    parent for HEART is ORGAN, and the parent for ACTOR is ARG.
;; OPTIONS is an optional association list of the same type as used
;;    with SCHEMA! and SEQ-SCHEMA!.  The OPTIONS list following SEQ-ROLE
;;    may contain (FIRST), (LAST), (ITERATE), or (OPTIONAL). These have
```

```
;;    different sets of default parameters for the sequence connections.
;;
;; Examples:
;;
;;   (SCHEMA! 'FISH 'ANIMAL
;;        '(TYPE-ROLE FIN ORGAN)
;;        '(ROLE DORSAL-FIN FIN))
;;
;;      This creates a node FISH, connected to ANIMAL; a type-role
;;      FIN, connected to FISH (its head) and ORGAN (its parent);
;;      and an individual role DORSAL-FIN, connected to FISH and FIN.
;;
;;   (SEQ-SCHEMA! 'CLAUSE 'UTTER
;;        '(SEQ-ROLE VERB CONSTITUENT))
;;
;;      This creates the CLAUSE schema and two roles, VERB and VERB_END
;;      and 9 connections.
;;
;; Maps are created using option sublists of the following form:
;;
;;   (MAP-TYPE PARENT . OPTIONS)
;;
;; MAP-TYPE is either MAP or SEQ-MAP.  SEQ-MAP creates a start and an
;;   end node and the appropriate connections.
;; PARENT is the role parent of the map role.  For example, for the
;;   HEART in the ELEPHANT schema, the parent is the top-level HEART
;;   role in the ANIMAL schema.
;; OPTIONS is an association list of the same type as the one used with
;;   SCHEMA and SEQ-SCHEMA.  As with SEQ-ROLE, the OPTIONS list
;;   following SEQ-MAP may contain (FIRST), (LAST), (ITERATE), or
;;   (OPTIONAL).
;;
;; The name assigned to a map role is a concatenation of the head name
;;   and the role parent name with a separating colon.  For example, the
;;   HEART map in the ELEPHANT schema is assigned the name
;;   ELEPHANT:HEART.
;; Maps of individual roles automatically inherit the individual
;;   property so that they can be distinguished from type maps (such as
;;   the ORGAN role in the ELEPHANT schema).
;;
;; Examples:
;;
;;    (SCHEMA! 'CLYDE 'ELEPHANT
;;         '(INDIV)
;;         '(MAP TRUNK))
;;
;;       This creates a schema for ELEPHANT CLYDE and a role for
;;       his TRUNK.  The role is connected to the CLYDE node and
;;       the general TRUNK role in the ELEPHANT schema.
;;
;;    (SEQ-SCHEMA! *DOG NP
;;         '(MAP CONTENT
;;              (RECOVERY .05))
;;         '(SEQ-MAP NOUN))
;;
;;       This creates a schema for *DOG and three map roles in it,
;;       *DOG:CONTENT (with recovery level .05), *DOG:NOUN, and
```

```
;;        *DOG:NOUN_END.
;;
;;
;; CREATING CONNECTIONS
;;
;; Some connections are built automatically when schemas or roles are
;; created.  For example, a call to SCHEMA! creates a connection from
;; the new node to the specified parent and another from the parent to
;; to the new node.
;;
;; Other connections are created using option sublists beginning with
;; keywords specifying the type of connection.  It should be noted that
;; the these types have no role in processing.  They simplify the
;; setting of connection parameter because different types have
;; different default value for the parameters.  The types are also
;; useful in debugging. The general form for option sublists creating
;; connections is as follows:
;;
;;    (CONNECTION-TYPE DESTINATION . OPTIONS)
;;
;; CONNECTION-TYPE is of the following: IS-A, INSIDE-IS-A, MAP-IS-A,
;;   SEQ-IS-A, EQUIV, INSIDE-EQUIV, DISTINCT, FOLLOWS, MERGE.
;;   INSIDE-IS-A and INSIDE-EQUIV connect roles to aother roles in the
;;   same schema.  MERGE differs from other types in that the
;;   DESTINATION node is merged with the source node (see MERGE! below).
;;   FOLLOWS creates sequencing connections from start and end source to
;;   start and end destination nodes.
;; DESTINATION specifies the node at the other end of the connection.
;;   This may either be the name of a node which already exists or a
;;   "path" to a map role which is then created before the connection is
;;   created.  A role path is a list with the head of a schema as its
;;   CAR and a list of role names as its CDR.  For example, (TRIAL27
;;   PRESIDING-JUDGE NAME) is the path for the role TRIAL27:PRESIDING-
;;   JUDGE:NAME, that is, the NAME of the PRESIDING-JUDGE of TRIAL27.
;; OPTIONS is an association list of the same type as the one used with
;;   SCHEMA and SEQ-SCHEMA.
;;
;; Examples:
;;
;;   (SCHEMA! 'PHYSICAL-TRANSFER7 'PHYSICAL-TRANSFER
;;        '(INDIV)
;;        '(MAP ACTOR
;;             (EQUIV MARY))
;;        '(MAP OBJ
;;             (IS-A BOX))
;;        '(MAP DESINATION
;;             (EQUIV (TABLE3 TOP-SURFACE))))
;;
;;   This creates a schema for an instance of PHYSICAL-TRANSFER.  The
;;   ACTOR of the event is MARY, and the OBJECT is a BOX.  The
;;   connection from BOX to PHYSICAL-TRANSFER7:OBJ has weight .3.  The
;;   DESTINATION of the event is TABLE3:TOP-SURFACE, a map role which is
;;   created here.
;;
;;   (SEQ-SCHEMA! '*DOG
;;        '(MAP CONTENT
;;             (IS-A DOG))
;;        '(SEQ-MAP NOUN
```

```
;;              (SEQ-IS-A "DOG")))
;;
;;    This creates the schema *DOG.  Its CONTENT role is created and
;;    connected to the DOG node.  Its NOUN and NOUN_END roles are also
;;    created and connected to the node "DOG".
;;
;;    (SCHEMA! 'SUICIDE 'KILL
;;         '(MAP ACTOR
;;               (MERGE (SUICIDE OBJ))))
;;
;;    This creates the SUICIDE schema.  The ACTOR and OBJ roles within
;;    the schema are merged into a single role which has connections to
;;    both the general ACTOR node (in the ACT schema) and the OBJ node
;;    (in the general FACT schema).
;;
;;
;; CREATING WINNER-TAKE-ALL NETWORKS
;;
;; WTA networks are created using an option sublist within the code
;; which creates the WTA parent of the network.  The sublist has the
;; following form:
;;
;;    (WTA NAME . OPTIONS)
;;
;; NAME is a name to be assigned to the WTA hub node.  It is checked
;; first to determine whether it already belongs to an existing node.
;; The hub node is created and connected to the WTA parent node.
;;
;; OPTIONS is an options association list.  It is used for assigning a
;; LIFE to the WTA hub or setting one of the connection weights.
;;
;; Nodes are made members of a WTA network using the following option
;; sublist:
;;
;;    (WTA-MEMBER HUB-NAME . OPTIONS)
;;
;; HUB-NAME is the name of the hub node for the WTA network.  The new
;; member is connected to the hub and to all existing members of the WTA
;; network (with inhibitory connections.
;;
;; OPTIONS is an options association list.  It is used for setting the
;; weights on the connections joining the member to the hub node.
;;
;; Examples:
;;
;;   (SCHEMA! 'ANIMAL 'ORGANISM
;;      '(WTA MALE-FEMALE))
;;   (SCHEMA! 'MALE-ANIMAL 'ANIMAL
;;      '(WTA-MEMBER MALE-FEMALE))
;;   (SCHEMA! 'FEMALE-ANIMAL 'ANIMAL
;;      '(WTA-MEMBER MALE-FEMALE))
;;
;;   This set of schemas represents the knowledge that an ANIMAL is
;;   either a MALE-ANIMAL or a FEMALE-ANIMAL.  In addition to ANIMAL,
;;   MALE-ANIMAL, and FEMALE-ANIMAL, the WTA hub node MALE-FEMALE is
;;   created and connected to ANIMAL, MALE-ANIMAL, and FEMALE-ANIMAL.
;;   Inhibitory connections are also created between MALE-ANIMAL and
;;   FEMALE-ANIMAL.
```

```
;;
;;
;; SETTING CONNECTION WEIGHTS AND DECAY RATES.
;;
;; Connection weights and decay rates are specified in option sublists
;; with the following form?
;;
;;    (PARAMETER-TYPE PARAMETER-LIST*)
;;
;; PARAMETER-TYPE is either WEIGHTS or DECAYS.
;; Each PARAMETER-LIST consists of a CONNECTION-KEY and a VALUE.
;;   A CONNECTION-KEY is atom identifying the connection that the
;;   weight or decay rate is to be assigned to.  For example, the
;;   connection-key ->PARENT assigns the parameter to the connection
;;   from the node being defined to its parent.  When a weight or
;;   decay rate for a connection is not specified, it is assigned a
;;   default value for the particular connection type.
;;
;; Examples:
;;
;;   (SCHEMA! 'PHYSICAL-TRANSFER7 'PHYSICAL-TRANSFER
;;        '(DECAYS (->PARENT .5))
;;        '(INDIV)
;;        '(MAP ACTOR
;;             (WEIGHTS (->HEAD .2))
;;             (EQUIV MARY))
;;        '(MAP OBJ
;;             (IS-A BOX
;;                  (WEIGHTS (<-PARENT .3)))))
;;
;;   In this schema all connections have default weights and
;;   decay rates except the connection from PHYSICAL-TRANSFER7
;;   to PHYSICAL-TRANSFER (decay .5), the connection from
;;   PHYSICAL-TRANSFER7:ACTOR (weight .2), and the connection
;;   from BOX to PHYSICAL-TRANSFER7:OBJ (weight .3).


;; The general function for defining schemas.

(define (schema! name parent . options)
   (let ((node (node2! name parent .5 .2 options)))
    (and node
         (do-options node options))))

;; Defines a "sequence schema", in which the head actually consists
;; of a start-end pair of nodes.

(define (seq-schema! name parent . options)
  (let ((ns (seq-node! name parent 'head options)))
    (and ns
         (do-options (car ns) options)
         (car ns))))

;; Roles and maps are normally created in option sublists within
;; calls to the schema family of functions.  The following three
;; functions are used when there is reason to create a role or map
;; outside the function creating the head.  These functions take
;; OPTIONS arguments like the schema-type functions.
```

```
(define (+role! name head parent . options)
  (role! 'role name (get-node head) parent options))

(define (+type-role! name head parent . options)
  (role! 'type-role name (get-node head) parent options))

(define (+map! head parent . options)
  (map-role! (get-node head) parent options))

;; Handles the OPTIONS argument of functions such as SCHEMA!,
;; ROLE!, and IS-A!.  OPTIONS is an association list, each sublist of
;; which either creates a role/map in NODE, creates connections joining
;; NODE with another node, or adds an attribute to NODE's attribute
;; list.  Complains if it doesn't understand a sublist keyword.

(define (do-options node options)
   (do ((options options (cdr options)))
       ((null? options)
        node)
       (let ((next (car options)))
       (case (car next)
             ((role type-role)
              (role! (car next) (cadr next)
                     node (caddr next) (cdddr next)))
             ((map) (map-role! node (cadr next) (cddr next)))
             ((wta-member) (wta-member! node (cadr next) (cddr next)))
             ((wta) (wta! node (cadr next) (cddr next)))
             ((is-a) (is-a! node (cadr next) (cddr next)))
             ((merge) (merge! node (cadr next) (cddr next)))
             ((equiv) (equiv! node (cadr next) (cddr next)))
             ((ins-equiv) (equiv! node (cadr next)
                                  (cons '(inside) (cddr next))))
             ((ins-is-a) (is-a! node (cadr next)
                                (cons '(inside) (cddr next))))
             ((distinct) (distinct! node (cadr next) (cddr next)))
             ((map-parent map-is-a)
              (map-is-a! node (cadr next) (cddr next)))
             ((exec action) (executable node (cadr next)))
             ((follows) (sequence! (cadr next) node (cddr next)))
             ((seq-role)
              (seq-role! (cadr next) node (caddr next)
                         'role (cdddr next)))
             ((seq-map)
              (seq-role! (cadr (assq 'name (cddr next)))
                         node (cadr next) 'map (cddr next)))
             ((seq-is-a) (seq-is-a! node (cadr next) (cddr next)))
             ((seq-end end) (do-options (seq-end node) (cdr next)))
             ((recovery refractory life decay ->perception role indiv
               action)
              (add-atrib node next))
             ((weights decays first last optional iterate only name)
               t)
             (else (msg "I don't know what " (car next) " means" t)
                   t)))))


;; The following functions are not normally called by the user.
```

```
;; They get called within the schema-type functions defined above.

;; Creates node with name NAME and with basic connections (either
;; PARENT--CHILD or MAP-PARENT--MAP-CHILD) connecting it to the node
;; named PAR.  WEIGHTS specifies the weights on these connections.  This
;; procedure checks first to see whether there is already a node with
;; name NAME; if so, it asks the user whether to kill that node.  If
;; NAME is NIL, a gensymed name is created from PAR.  If there is no
;; node with name PAR, the procedure returns NIL.

(define (node! name par type1 type2 weight1 weight2
               decay1 decay2)
  (and (check-name name)
       (node1! name par type1 type2 weight1 weight2
               decay1 decay2)))

;; Like NODE!, except it does not check the name first.

(define (no-check-node! name par type1 type2 weight1 weight2
                        decay1 decay2)
  (let ((node (get-node name)))
    (if (node? node)
        node
        (node1! name par type1 type2 weight1 weight2
                decay1 decay2))))

(define (node1! name par type1 type2 weight1 weight2
                decay1 decay2)
   (let ((node (make-node!
                  (or name (generate-symbol (symbol->string par)))))
         (parent (get-node par)))
      (if (or (not parent) (check-node parent))
          (block (if parent
                     (link! node parent type1 type2 weight1 weight2
                            decay1 decay2 '(fast))
                     (set (node-connections node) nil))
                 node)
          nil)))

(define (node2! name parent def->par def<-par options . map?)
  (let ((weights (cdr (assq 'weights options)))
        (decays (cdr (assq 'decays options))))
    (node! name parent (if map? 'map-parent 'parent)
           (if map? 'map-child 'child)
           (find-weight weights '->parent def->par)
           (find-weight weights '<-parent def<-par)
           (find-decay decays '->parent nil)
           (find-decay decays '<-parent nil))))

;; Creates a start-end pair of nodes.  Each node is accessible
;; from the other through the SEQ-START and SEQ-END attribute lists
;; in the NODE-ATTRIBS component.  Several other connections are
;; created in addition to the usual PARENT--CHILD connections.

(define (seq-node! name par type options)
  (and (check-name name)
       (let ((parent (get-node par)))
         (and (check-node parent)
```

```
            (let ((n1 (make-node! name))
                  (n2 (make-node!
                           (concatenate-symbol name '_end)))
                  (parent2 (seq-end parent)))
              (link1! n1 parent 'parent 'child
                      '->parent (if (eq? type 'role) .15 .44)
                      '<-parent
                      (case type ((role) .05)
                                 ((map) '(.44 . .7))
                                 (else .3))
                      options 'fast)
              (if parent2
                  (link1! n2 parent2 'parent 'child
                          'end->parent-end
                          (if (eq? type 'role) .15 .5)
                          'end<-parent-end
                          (case type ((role) .05)
                                     ((map) '(.44 . .7))
                                     (else '(.2 . .7)))
                          options 'fast))
              (link1! n1 n2 'start->end 'end->start
                      'start->end '(.2 . .98)
                      'end->start '(-.15 . .9) options 'fast)
              (set (node-attribs n1)
                   (list '(seq start) (list 'seq-end n2)
                         '(recovery .05)))
              (set (node-attribs n2)
                   (list '(seq) (list 'seq-start n1)
                         '(recovery .05)))
              (cons n1 n2))))))

;; Creates a top-level role node, linking it to its parent and its head.
;; This function is used only for top-level roles such as the ACTOR
;; in the ACT schema.
;; TYPE is either TYPE-ROLE (e.g., PARTICIPANT, PART, LEG) or ROLE
;; (e.g., ACTOR, NAME, NOSE).  This function is normally called via
;; an option sublist beginning with ROLE or TYPE-ROLE within a call
;; to SCHEMA! or SEQ-SCHEMA!.

(define (role! type name head parent options)
  (let ((role (node2! name parent .15 .05 options)))
    (and role
         (link1! role head 'head 'role
                 '->head .3 '<-head .3 options)
         (if (eq? type 'role)
             (push (node-attribs role) '(indiv))
             t)
         (do-options role options)
         role)))

;; Creates a "map" (a copy of a role) joined to PARENT by MAP-PARENT-
;; MAP-CHILD connections and to HEAD by HEAD--ROLE connections.  This
;; function is normally called via an option sublist beginning with MAP
;; within a call to SCHEMA! or SEQ-SCHEMA!.

(define (map-role! head parent options)
  (let ((role (node2! (or (cadr (assq 'name options))
                          (map-name head parent))
```

```
                    parent .5 .3 options t))
        (par (get-node parent))
        (indiv-role? (role-of-indiv? head)))
    (and role
         (set (node-attribs role)
              (append (if (and indiv-role?
                               (assq 'indiv (node-attribs par)))
                          '((recovery .35) (decay .98)) nil)
                      (if (assq 'indiv (node-attribs par))
                          '((indiv)) nil)
                      '((map) (role))))
         (link1! role head 'head 'role
                 '->head (if indiv-role? .15 .3)
                 '<-head (if indiv-role? .1 .3) options)
         (do-options role options)
         role)))

;; Creates a sequence role start-end pair of nodes
;; joined by a set of connections to its head.  This function is
;; normally called via an option sublist beginning with SEQ-ROLE within
;; SEQ-SCHEMA!.  TYPE is either ROLE or MAP.

(define (seq-role! name head par type options)
  (and (or (eq? type 'map) (check-name name))
       (let ((parent (get-node par)))
         (and (check-node parent)
              (let ((nodes (seq-node!
                                (or name (map-name head par parent))
                                parent type options)))
                (seq-has-a! (car nodes) (cdr nodes) head
                            (seq-end head) type options)
                (seq-iterate! (car nodes) (cdr nodes) options)
                (if (eq? type 'role)
                    (walk (lambda (x)
                                  (push (node-attribs x) '(indiv)))
                          (list (car nodes) (cdr nodes)))
                    (block (set (node-attribs (car nodes))
                                (cons (list 'seq-end (cdr nodes))
                                      '((map) (seq start) (role)
                                        (recovery .05))))
                           (set (node-attribs (cdr nodes))
                                (cons (list 'seq-start (car nodes))
                                      '((map) (seq) (role)
                                        (recovery .05))))))
                (push (node-attribs (car nodes)) '(refractory -3))
                (do-options (car nodes) options)
                (car nodes))))))


;; Creates the map of PARENT for HEAD using default weights for the
;; connections.

(define (map-role1! head parent)
  (let ((node (no-check-node! (map-name head parent parent)
                              parent 'map-parent 'map-child
                              .5 .3
                              *default-decay* *default-decay*)))

    (if node
```

```
        (block (link! node head 'head 'role .3 .3
                      *default-decay* *default-decay* nil)
               (set (node-attribs node) '((map) (role)))
               node)
        nil)))

;; Creates either a sequence role or an ordinary role depending on
;; the attributes of PARENT.

(define (general-map! head parent)
  (if (assq 'seq (node-attribs parent))
      (seq-role! nil head parent 'map nil)
      (map-role1! head parent)))

;; Creates a map for the role "path" PATH.  PATH is a list beginning
;; with a head and followed by one or more roles.  For the first role
;; in the list a map is created for the head node.  For each
;; successive role the map just created becomes the head for the next
;; map to be created.  This function gets called when a map is
;; referred to by a path (list) rather than by its name in an option
;; sublist.

(define (recursive-map! path)
  (let ((nodes (map get-node path)))
    (and (every check-node nodes)
         (do ((ns (cddr nodes) (cdr ns))
              (current (general-map! (car nodes) (cadr nodes))
                       (general-map! current (car ns))))
             ((null? ns) current)))))

;; Like RECURSIVE-MAP! except the last map is not created.  Instead
;; the last head and role (map parent) are returned.  This is used
;; in MERGE!, where the map is not wanted.

(define (recursive-map-but-1! path)
  (let ((nodes (map get-node path)))
    (and (every check-node nodes)
         (do ((ns (cdr nodes) (cdr ns))
              (current (car nodes)
                       (general-map! current (car ns))))
             ((null? (cdr ns))
              (cons current (car ns)))))))

;; Creates a winner-take-all hub node linked to PARENT.
;; "Members" of the network are added with WTA-MEMBER!.  This function
;; may be called via an option sublist beginning with WTA in SCHEMA!
;; or SEQ-SCHEMA!.

(define (wta! parent name options)
  (and (check-name name)
       (let ((node (make-node!
                    (or name (concatenate-symbol parent '-wta)))))
         (set (node-attribs node)
              (list '(wta-hub)
                    (or (assq 'life options) (list 'life 3))))
         (link1! node parent 'wta-parent 'wta-child
                 '->parent 0 '<-parent .3 options 'fast)
         node)))
```

```
;; Like WTA! except that it checks to determine whether a node named
;; SRC exists.

(define (+wta! src name . options)
  (let ((source (get-node src)))
    (and (check-node source)
         (wta! source name options))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                        CONNECTION CREATION                             ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Creates a pair of connections joining NODE1
;; and NODE2.  WEIGHT1, TYPE1, and DECAY1 are associated with the
;; connection from NODE1 to NODE2;; WEIGHT2, TYPE2, and DECAY2
;; with the connection from NODE2 to NODE1.  Types include PARENT,
;; CHILD, ROLE, HEAD, INHIBIT, AFTER, and BEFORE.  Connection
;; types play no role in processing but are convenient for building
;; the network.

(define (link! node1 node2 type1 type2 weight1 weight2
               decay1 decay2 fast?)
    (connect! node1 node2 type1 weight1 decay1 fast?)
    (connect! node2 node1 type2 weight2 decay2 fast?))

(lset *connection-count* 0)

;; Creates a connection from SOURCE to DEST with WEIGHT, DECAY, and
;; TYPE.

(define (connect! source dest type weight decay fast?)
   (increment *connection-count*)
   (push (node-connections source)
         (cons* dest weight type decay fast?)))

;; Like LINK! but finds weights and decays in OPTIONS list, the
;; final argument to functions like SCHEMA! and IS-A!.

(define (link1! node1 node2 type1 type2
                key1 def1 key2 def2 options . fast?)
  (let ((weights (cdr (assq 'weights options)))
        (decays (cdr (assq 'decays options))))
    (link! node1 node2 type1 type2
           (find-weight weights key1 (if (atom? def1) def1 (car def1)))
           (find-weight weights key2 (if (atom? def2) def2 (car def2)))
           (find-decay decays key1
                       (and (list? def1) (cdr def1)))
           (find-decay decays key2
                       (and (list? def2) (cdr def2)))
           fast?)))

;; Copies the connections of types CONNECTION-TYPES from node DEST to
;; node SOURCE.  Currently used only in MERGE!

(define (copy-connections! dest source connection-types)
  (walk (lambda (x)
```

```
            (let ((lt-pair (assq (connection:type x) connection-types))
                  (conn1 (assq dest (node-connections source)))
                  (conn2 (assq source (node-connections dest))))
              (and lt-pair
                   (connection! dest (car x) (car lt-pair) (cdr lt-pair)
                                (connection:weight conn1)
                                (connection:weight conn2)
                                (connection:decay conn1)
                                (connection:decay conn2)
                                (connection:speed conn1)))))
          (node-connections source)))

;; Makes NODE a member of winner-take-all network with hub HUB.
;; Creates inhibition connectionss joining NODE to other members of
;; network.  This function may be called via an option sublist
;; beginning with WTA-member within a call to SCHEMA! or SEQ-SCHEMA!.

(define (wta-member! node hub options)
  (let ((wta (get-node hub)))
    (and (check-node wta)
         (link1! node wta 'wta-parent 'wta-member
                 '->hub -.3 '<-hub .4 options 'fast)
         (walk (lambda (conn)
                 (and (eq? (connection:type conn) 'wta-member)
                      (not (eq? (connection:end conn) node))
                      (distinct! node (connection:end conn)
                                 options)))
               (node-connections wta))
         node)))

;; The following 7 functions create particular types of connections.
;; They are normally called via option sublists within calls to SCHEMA!
;; and SEQ-SCHEMA!.  The sublists begin with the connection type
;; names, e.g., IS-A.

(define (is-a! child par options)
  (let ((parent (get-node* par)))
    (and (check-node parent)
         (let ((inside? (assq 'inside options))
               (child-map? (assq 'map (node-attribs child))))
           (link1! child parent (if inside? 'inside-parent 'parent)
                   (if inside? 'inside-child 'parent)
                   '->parent (if child-map? .5 .15)
                   '<-parent (if child-map? .3 .05) options 'fast))
         child)))

(define (map-is-a! child par options)
  (let ((parent (get-node par)))
    (and (check-node parent)
         (link1! child parent 'map-parent 'map-child
                 '->parent .5 '<-parent .15 options 'fast)
         child)))

(define (seq-is-a! child par options)
  (let ((parent (get-node par)))
    (and (check-node parent)
         (let ((parent-end (seq-end parent)))
           (link1! child parent 'parent 'child
```

```
                    '->parent .5
                    '<-parent (if parent-end '(.3 . .7) '(.5 . .7))
                    options 'fast)
          (if parent-end
              (link1! (seq-end child) parent-end 'parent 'child
                      'end->parent-end .5 'end<-parent-end '(.42 . .7)
                       options
                      'fast)
              (link1! (seq-end child) parent 'parent 'child
                      'end->parent 0 'end<-parent .5 options
                      'fast))))))

;; ROLE2-ATTRIB is either the name of an existing role or a map path
;; to a map which has not yet been created.

(define (merge! role1 role2-attrib options)
  (let* ((r2 (get-node* role2-attrib t))
         (role2 (if (atom? r2) r2
                    (get-node
                         (map-name (car r2) (cdr r2) (cdr r2))))))
    (if (node? role2)
        (block (copy-links! role1 role2
                            '((head . role) (map-parent . map-child)
                              (parent . child) (equiv . equiv)))
               (kill-node! role2)
               role1)
        (block (if (not (connected? role1 (car r2)))
                   (link1! role1 (car r2) 'head 'role
                           '->head .1 '<-head .1 options))
               (link1! role1 (cdr r2) 'map-parent 'map-child
                       '->parent .5 '<-parent .3 options 'fast)))))

(define (equiv! role f options)
  (let ((filler (get-node* f)))
    (and (check-node filler)
         (let ((type (if (assq 'inside options)
                         'inside-equiv 'equiv)))
           (link1! role filler type type '->equiv .5 '<-equiv .3
                   options 'fast))
         role)))

(define (distinct! node1 n2 options)
  (let ((node2 (get-node* n2)))
   (and (check-node node2)
        (link1! node1 node2 'inhibit 'inhibit
                '->inhib '(-.2 . .5) '<-inhib '(-.2 . .5) options
                'fast)
        node1)))

(define (sequence! bef after options)
  (let ((before (get-node* bef)))
    (and (check-node before)
         (link1! (seq-end before) after 'after 'before
                 '->after .3 '<-after .08 options 'fast))))

;; Used for sequence roles which can iterate, for example, adjective
;; NP modifiers.
```

```
(define (seq-iterate! r1 r2 options)
  (let ((iter (assq 'iterate options)))
    (and iter
         (link1! r2 r1 'iterate 'iterate
                 'end->start .3 'start->end 0 options 'fast))))

;; Creates the various connections joining a sequence role pair to its
;; head pair.  This gets called in SEQ-ROLE!.

(define (seq-has-a! role1 role2 head1 head2 type options)
  (if head2
      (let ((first? (assq 'first options))
            (optional? (assq 'optional options))
            (last? (assq 'last options))
            (only? (assq 'only options)))
        (link1! role1 head1 'head 'role
                '->head (cond (first? '(.35 . .5)) (optional? .05)
                              (else '(.2 . .5)))
                '<-head (cond (first? .5) (optional? .15)
                              (else .24))
                options)
        (link1! role2 head2 'head 'role
                'end->head-end (cond (last? .5) (only? .25)
                                     (else .1))
                'end<-head-end (if last? .3 .2) options)
        role1)
      (block
       (link1! role1 head1 'head 'role
               '->head .08 '<-head .24 options)
       (link1! role2 head1 'head 'role
               'end->head .01 'end<-head .1 options))))

;; Changes the weight on the connection from SOURCE to DESTination to
;; WEIGHT.  Assumes the connection already exists.

(define (change-weight! source dest weight)
  (let* ((conns (node-connections source))
         (old-list (assq dest conns)))
    (if old-list
        (block (set (node-connections source)
                    (subst alikev?
                           (list dest weight
                                 (connection:type old-list)
                                 (connection:decay old-list))
                           (assq dest conns) conns))
               weight)
        (block (msg "No existing connection for destination" t)
               nil))))

(define (change-decay! source dest decay)
  (let* ((conns (node-connections source))
         (old-list (assq dest conns)))
    (if old-list
        (block (set (node-connections source)
                    (subst alikev?
                           (list dest
                                 (connection:weight old-list)
                                 (connection:type old-list)
```

```
                                        decay)
                                   (assq dest conns) conns))
                          decay)
                   (block (msg "No existing connection for destination" t)
                          nil))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                    KILLING NODES AND CONNECTIONS                       ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Removes NODE and its associated connections and roles from the
;; network.

(define (kill-node! node)
  (msg "   Killing " node t)
  (walk (lambda (x)
                      (if (not (eq? (connection:type x) 'role))
                          (kill-connection! node (car x))))
             (node-connections node))
  (walk kill-node! (connection:ends/type node 'role))
  (let ((node-end (seq-end node)))
    (if node-end (kill-node! node-end)))
  (kill-name node)
  t)

;; NAME is no longer associated with any node.

(define (kill-name node)
   (set (table-entry *nodes* (node-name node)) nil))

;; Removes connection joining NODE to OTHER-NODE from the network.

(define (kill-connection! node other-node)
  (and other-node
       (node? other-node)
       (let ((connections (node-connections other-node)))
          (set (node-connections other-node)
               (del alikev? (assq node connections) connections)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(herald propagation)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                      ;;;
;;;   THIS FILE CONTAINS THE FUNCTIONS USED IN PROPAGATING               ;;;
;;;   ACTIVATION THROUGH THE NETWORK, DECAYING ACTIVATION,               ;;;
;;;   AND HANDLING THE PECULIARITIES OF WTA NETWORKS, AND                ;;;
;;;   REFRACTORY PERIODS.                                                ;;;
;;;                                                                      ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                          GLOBAL VARIABLES                                 ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;; GLOBAL VARIABLES USED IN PROCESSING.

;; When *NO-INPUT-FROM-USER?* is true, the user is not asked
;; to provide input every *INPUT-INTERVAL* time steps.

(lset *no-input-from-user?* t)

;; When *BREAK?* is true, PROPAGATE breaks after each time step.

(lset *break?* nil)

;; Propagation halts after *TIME-OUT* time steps have passed.

(define *time-out* 50)

;; At each *INPUT-INTERVAL* time steps either a new list of input
;; nodes is taken off of the INPUT list and passed to PROPAGATE, or the
;; user is asked to specify input.

(define *input-interval* 1)

;; The number of fast connections that can be traversed in one time
;; step.

(define *fast-conns-per-ts* 4)


;;; GLOBAL VARIABLES FOR PARAMETERS.

;; Activation level above which nodes fire.

(define *fire-threshold* .5)

;; Negative of the default number of time steps which nodes are
;; inhibited following firing.  Other values are specified in
;; nodes' attribute lists.  Gets incremented on each time step
;; (in DECAY).

(define *default-refractory* -6)

;; When activation on a connection goes below this, that connection
;; is removed from the activation list for the node.

(define *activation-min* .04)

;; Activation which a node assumes following inhibition due to
;; firing.

(define *default-recovery* .15)


;;; GLOBAL VARIABLES FOR ACTIVATED NODES.
```

```
;; *ACTIVE* is a list of currently activated nodes.

(lset '*active* nil)

;; *ACTIVE-WTAS* is a list of currently activated winner-take-all
;; hub nodes.  These nodes are also kept on the *ACTIVE* list.

(lset *active-wtas* nil)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                PROPAGATION                               ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;
;; PROPAGATE spreads activation beginning either from the nodes
;; specified in INPUT or from those which the user types in.
;;
;; INPUT is a list of lists.  Each sublist is a list of nodes which
;; fire through "external" activation on a particular time step.
;; This happens every *INPUT-INTERVAL* time steps. If there are no
;; more INPUT sublists left at a given time step, the user is asked
;; to enter the input (or No if there are none) unless the global
;; variable *NO-INPUT-FROM-USER?* is true.
;;
;; On each time step
;; 1) all activated nodes decay
;; 2) activation is propagated from all nodes which fired on the last
;;    time step
;; 3) activation continues to propagate on fast connections
;; 4) newly firing nodes become sources of activation for the next
;;    time step.

(define (propagate input)
  (deactivate-all)
  (do ((time-step 1 (1+ time-step))
       (sources nil))
      ((= time-step *time-out*)
       t)
      (msg t t "T I M E   S T E P   " time-step t t)
      (msg "Decaying..." t)
      (decay)
      (msg "  done" t)
      (set sources
           (get-next-sources
                sources
                (and (or (= *input-interval* 1)
                         (= 1 (remainder time-step *input-interval*)))
                     (external-excite
                         (if input
                             (map get-node (pop input))
                             (get-input-from-user))))))
      (if *break?* (breakpoint))))

;; Propagates activation from SOURCE-NODES, causes those nodes with
;; enough activation to fire, and returns the set of newly firing nodes.
;; EXT-FIRE is a set of nodes which have fired as a result of user
```

```
;; input on the current time step.  As many as *FAST-CONNS-PER-TS* fast
;; connections can fire in one time step.

(define (get-next-sources source-nodes ext-fire)
  (msg t "Time increment 1" t)
  (iterate -*- ((sources source-nodes)
                (fast-new ext-fire)
                (slow-new nil)
                (sources-next-ts ext-fire)
                (iteration 1))
     (cond (sources
            ;; Get nodes activated from (CAR SOURCES)
            (receive (fast slow)
                     (propagate1 (car sources)
                                 (node-connections (car sources))
                                 fast-new slow-new)
                     (-*- (cdr sources) fast slow sources-next-ts
                          iteration)))
           ((= iteration (1+ *fast-conns-per-ts*))
            ;; No more time left, even for fast connections;
            ;; Take care of WTA hubs and return sources for next time
            ;;step
            (append (do-wtas (append source-nodes sources-next-ts))
                             sources-next-ts))
           (else
            ;; Get nodes which can fire, make the ones that were
            ;; activated only along fast connections new sources for the
            ;; next time increment
            (receive (next-sources new-sources)
                     (fire-nodes fast-new slow-new ext-fire)
                     (if (= (1+ iteration) (1+ *fast-conns-per-ts*)) t
                         (msg t "Time increment " (1+ iteration) t))
                     (-*- next-sources nil nil
                          (append new-sources sources-next-ts)
                          (1+ iteration)))))))

;; Activates from SOURCE along CONNections.  The newly activated nodes
;; are FAST-NEW (those which have been activated along fast connections)
;; and SLOW-NEW.

(define (propagate1 source conns fast-new slow-new)
  ;; If SOURCE is hooked up to a perceptual routine, activate it
  ;; and add it to FAST-NEW
  (let ((percept-fire (percept-activate source)))
    (iterate -*- ((conns conns)
                  (fast-new (if percept-fire
                                (cons percept-fire fast-new)
                                fast-new))
                  (slow-new slow-new))
      (if (null? conns)
          (return fast-new slow-new)
          (let* ((dest (connection:end (car conns)))
                 (actv (node-activation dest)))
            (cond ((no-activate? dest source (car conns) actv)
                   (-*- (cdr conns) fast-new slow-new))
                  (else
                    (activate dest source
                              (connection:weight (car conns))
```

```
                              (connection:decay (car conns))
                              actv)
                        (cond ((slow-connection? (car conns))
                               (-*- (cdr conns) (delq dest fast-new)
                                    (if (memq dest slow-new) slow-new
                                        (cons dest slow-new))))
                              ((memq dest slow-new)
                               (-*- (cdr conns) (delq dest fast-new)
                                    slow-new))
                              ((memq dest fast-new)
                               (-*- (cdr conns) fast-new slow-new))
                              (else (-*- (cdr conns)
                                         (cons dest fast-new)
                                         slow-new)))))))))

;; Checks to see whether DEST can be activated along CONN from SOURCE.
;; A node in its refractory period can receive negative activation only.
;; (This does not actually affect the node until the refractory period
;; ends.) Also handles WTA nodes.  Activates them if they are not
;; already active, deactivates them if they are being activated from
;; members of the WTA network.

(define (no-activate? dest source conn curr-actv)
   (or (and (refracted? curr-actv)
            (positive? (connection:weight conn)))
       (and (eq? (connection:type conn) 'wta-child)
            (not (memq dest *active-wtas*))
            ;; DEST is WTA hub for network for which SOURCE is parent;
            (activate-wta dest source))
       (and (eq? (connection:type conn) 'wta-parent)
            (memq dest *active-wtas*)
            ;; DEST is an activated WTA hub.  Since member of network
            ;; (SOURCE) has fired, deactivate hub.
            (deactivate dest 'wta))))

;; Determines whether newly activated nodes FAST-NEW (those activated
;; along fast connections) and SLOW-NEW have enough activation to fire.
;; If so, checks to see whether they are inhibited through a WTA network
;; by a node with more activation.  If so, they fail to fire.
;; Otherwise, they fire (print out a message and enter their refractory
;; period). Returns (as multiple values) the nodes that can be further
;; sources on the current time step (because the activation has been
;; along fast connections) and those which will be sources on the next
;; time step.

(define (fire-nodes fast-new slow-new ext-fire)
   (iterate -**- ((activated-nodes (append fast-new slow-new))
                  (new-sources nil)
                  (new-sources-next-ts nil))
     (cond ((null? activated-nodes)
            (return new-sources new-sources-next-ts))
           (else (let ((actv (node-activation (car activated-nodes))))
                   (cond ((not (activation>threshold actv))
                          (-**- (cdr activated-nodes)
                                new-sources new-sources-next-ts))
                         ((inhib-compet?
                              (car activated-nodes)
                              (activation-sum (cdr actv))
```

```
                                (append new-sources-next-ts
                                        (cdr activated-nodes)))
                               (msg 3 (car activated-nodes)
                                    " can't fire because of competition" t)
                               (-**- (cdr activated-nodes)
                                     new-sources new-sources-next-ts))
                              (else
                                (fire (car activated-nodes)
                                      (sources-list (cdr actv))
                                      (memq (car activated-nodes) ext-fire))
                                (-**- (cdr activated-nodes)
                                      (if (memq (car activated-nodes)
                                                fast-new)
                                          (cons (car activated-nodes)
                                                new-sources)
                                          new-sources)
                                      (cons (car activated-nodes)
                                            new-sources-next-ts)))))))))

;; Goes through the list of currently activated winner-take-all hub
;; nodes, determines which should fire because of timing out and
;; which should be deactivated because one of the WTA members has fired.
;; SOURCES-NEXT-TS is a list of nodes which have just fired (those
;; which will be activation sources on the next time step).

(define (do-wtas sources-next-ts)
  (do ((wtas-left *active-wtas* (cdr wtas-left))
       (firing-wtas nil))
      ((null? wtas-left) firing-wtas)
      (cond ((= 1 (car (node-activation (car wtas-left))))
             (wta-time-out (car wtas-left))
             (push firing-wtas (car wtas-left)))
            ((intersection sources-next-ts
                           (wta-elems (car wtas-left)))
             (deactivate (car wtas-left) 'wta))
            (else t))))

;; Queries the user for a "value" for a particular role, and returns
;; the node representing the value after giving it enough activation
;; to fire.  This replaces what should actually be a call to a
;; perceptual routine.

(define (percept-activate source)
  (if (assq '->perception (node-attribs source))
      (block
          (msg 3 "Give the value of " (node-name source) ":  ")
          (let ((dest (get-node (user-response))))
            (activate dest source .5 .9 (node-activation dest))))
      nil))

;; Queries the user for a set of one or more input nodes which are to
;; fire automatically and returns the list of these nodes.  If there
;; are none, the user responds with "NO" or "N".

(define (get-input-from-user)
  (if *no-input-from-user?* nil
      (do ((curr-response (block (msg "Any new input?  ")
                                 (user-response))
```

```
                          (block (msg "Any more?  ")
                                 (user-response)))
                 (input nil (cons curr-response input)))
                ((null? curr-response)
                 (map get-node (reverse input))))))

;; Removes nodes from *ACTIVE* list (and *ACTIVE-WTAS* list if WTA?
;; is true) and sets NODE-ACTIVATION component of node to NIL.

(define (deactivate node . wta?)
  (set (node-activation node) nil)
  (set *active* (delq node *active*))
  (if wta? (set *active-wtas* (delq node *active-wtas*)))
  node)

;; Deactivates all active nodes.  This happens each time PROPAGATE
;; is called (though this should later be changed to permit priming
;; to be left over from previous sentences).

(define (deactivate-all)
  (walk (lambda (x) (set (node-activation x) nil))
        *active*)
  (set *active* nil)
  (set *active-wtas* nil))

;; Adds new activation (from SOURCE) to NODE's current activation.
;; If NODE is already activated from SOURCE, replaces old activation-
;; list for SOURCE with new activation-list

(define (activate node source weight decay curr-actv)
  (cond ((not curr-actv)
         (set (node-activation node)
              (list nil (list weight decay source)))
         (push *active* node))
        (else
          (let ((actv-same-source
                 (iterate -*- ((actv-left (cdr curr-actv)))
                     (cond ((null? actv-left) nil)
                           ((eq? source (caddr (car actv-left)))
                            (car actv-left))
                           (else (-*- (cdr actv-left)))))))
            (set (node-activation node)
                 (cons (car curr-actv)
                       (cons (list weight decay source)
                             (if actv-same-source
                                 (del alikev? actv-same-source
                                      (cdr curr-actv))
                                 (cdr curr-actv))))))))
  node)

;; Activates winner-take-all HUB from SOURCE.

(define (activate-wta hub source)
    (set (node-activation hub)
         (list (cadr (assq 'life (node-attribs hub)))))
    (push *active* hub)
    (push *active-wtas* hub)
    hub)
```

```
;; Each node on INPUT list is activated to .5 (enough to fire).

(define (external-excite input)
  (walk (lambda (n)
              (set (node-activation n)
                   (list nil (list .5 (node-decay n))))
              (if (not (memq n *active*)) (push *active* n)))
            input)
  input)

;; Prints firing message for NODE and sets its activation to its
;; refractory period (either one specified for the node or the
;; default).  If NODE is "executable", i.e., if it has an action
;; on its ATTRIBS list, a special "uttered" message is printed out.

(define (fire node sources indent?)
  (cond (indent? (msg 3 node " firing from external activation" t))
        (else
         (msg 3 node " firing from " t 5 (map car sources) t)))
  (set (node-activation node)
       (list (or (cadr (assq 'refractory (node-attribs node)))
                 *default-refractory*)))
  (if (not (memq node *active*)) (push *active* node))
  (if (and sources (assq 'action (node-attribs node)))
      (msg t 6 node "  u t t e r e d" t t))
  node)

;; Prints out a firing message for a WTA HUB and removes
;; it from the *ACTIVE-WTAS* list.

(define (wta-time-out hub)
  (msg 3 hub " timing out and firing" t)
  (set *active-wtas* (delq hub *active-wtas*))
  hub)

;; Implements decay in the activation of active nodes.

(define (decay)
  (walk (lambda (n)
          (let* ((actv (node-activation n))
                 ;; Get the activation lists with decayed activation
                 ;; values.
                 (decayed-actvs
                   (decay-activation (cdr actv))))
            (cond ((not (car actv))
                   ;; N is a node which is not refracted.
                   ;; If it has any activation left after decay,
                   ;; this becomes the new activation for the node.
                   (if decayed-actvs
                       (set (node-activation n)
                            (cons nil decayed-actvs))
                       (deactivate n)))
                  ((> (car actv) 1)
                   ;; N is a WTA hub;, activation integer decremented.
                   (set (node-activation n)
                        (list (-1+ (car actv)))))
                  ((= 1 (car actv))
```

```
                    ;; N is a WTA hub which has timed out; ignore it.
                    t)
                   ((< (car actv) -1)
                    ;; N is a refracted node not ready to recover;
                    ;; increment the activation key.
                    (set (node-activation n)
                         (cons (1+ (car actv)) decayed-actvs)))
                   (else
                    ;; N is a refracted node ready to recover;
                    ;; activation set to node's specified recovery
                    ;; activation or to default.
                    (set (node-activation n)
                         (cons nil
                               (cons (list (recovery n) (node-decay n))
                                     decayed-actvs)))))))
        *active*))

;; Goes through an list of activation-lists, replacing the activation
;; in each list with the new, decayed value.  A separate decay value
;; is used for eachlist.

(define (decay-activation actv-lists)
  (do ((actv-left actv-lists (cdr actv-left))
       (result nil
               (let ((new-actv
                      (* (actv-list:activation (car actv-left))
                         (actv-list:decay (car actv-left)))))
                 (if (< (abs new-actv) *activation-min*)
                     result
                     (cons (cons new-actv (cdr (car new-actv)))
                           result)))))
      ((null? actv-left) (reverse result))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;              MISCELLANEOUS HELP FUNCTIONS FOR PROPAGATION              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; The recovery activation for NODE.

(define (recovery node)
   (or (cadr (assq 'recovery (node-attribs node)))
       *default-recovery*))

;; The decay value for NODE.

(define (node-decay node)
   (or (cadr (assq 'decay (node-attribs node)))
       *default-decay*))

;; Is node with ACTIVATION-LIST refracted?

(define (refracted? activation-list)
     (and (car activation-list) (negative? (car activation-list))))

;; Is the sum of the activations in ACTIVATION-LISTS greater
;; than the default threshold?
```

```
(define (activation>threshold activation-lists)
    (>= (activation-sum (cdr activation-lists)) *fire-threshold*))

;; Has NODE fired or is it now ready to fire?

(define (fired? node)
   (let ((actv (node-activation node)))
     (or (refracted? actv)
         (activation>threshold actv))))

;; The sum of the activations in ACTIVATION-LISTS.

(define (activation-sum activation-lists)
   (apply + (map car activation-lists)))

;; The list of sources from ACTIVATION-LISTS.

(define (sources-list activation-lists)
   (map (lambda (a) (list (caddr a) (car a)))
        activation-lists))

;; Determines whether NODE1 cannot fire because of competition
;; from an inhibiting node which has more activation.

(define (inhib-compet? node1 strength1 others)
  (let ((inhib (connection:ends/type node1 'inhibit)))
    (any (lambda (a)
                 (let ((actv (node-activation a)))
                    (and (memq a inhib)
                         actv
                         (or (refracted? actv)
                             (>= (activation-sum (cdr actv))
                                 strength1)))))
         others)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(herald misc)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;         MISCELLANEOUS FUNCTIONS FOR ACCESSING NODES                ;;;
;;;         AND THEIR COMPONENTS.                                      ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;; Nodes are stored in a T hash table under their names.  The function
;; GET-NODE returns a node given its name.  There is a also a read
;; macro which converts %NAME to (GET-NODE NAME).

(lset *nodes* (make-table))

(lset *node-count* 0)
```

```
;; Creates a node with name NAME.

(define (make-node! name)
  (let ((node (make-node)))
    (increment *node-count*)
    (set (table-entry *nodes* name) node)
    (set (node-name node) name)
    node))

(define (name->node name)
  (table-entry *nodes* name))

(define ami-read-table
   (make-read-table standard-read-table 'ami-read-table))

(set (port-read-table (standard-input)) ami-read-table)

(set (read-table-entry ami-read-table #\%)
     (lambda (port ch read-table)
        (ignore ch)
        (ignore read-table)
        (list 'name->node (list 'quote (read-refusing-eof port)))))

;; Returns the node associated with NAME.

(define (get-node x)
  (if (node? x) x
      (name->node x)))

;; ATTRIBS is either a node name, in which case the associated node
;; is returned, or a list of names, in which case this is treated as
;; a path to a map, which is created, unless MERGE? is true, in which
;; case the last head and role are returned instead.

(define (get-node* attribs . merge?)
  (cond ((atom? attribs) (get-node attribs))
        (merge? (recursive-map-but-1! attribs))
        (else (recursive-map! attribs))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                          WEIGHTS AND DECAYS                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Returns the weight on the connection from SOURCE to DESTination.

(define (get-weight source dest)
   (connection:weight (assq dest (node-connections source))))

;; Returns the decay value for the connection from SOURCE to
;; DESTination.

(define (get-decay source dest)
   (connection:decay (assq dest (node-connections source))))

;; Returns a list of lists, each specifying a source and a weight
;; on a connection into NODE.
```

```
(define (weights-into-node node)
  (map (lambda (x) (list (car x) (get-weight (car x) node)))
       (node-connections node)))

;; Returns a weight to be assigned to  a connection being created.
;; WEIGHTS is an a-list of weights specified in a function like
;; SCHEMA! or IS-A!. KEY-NAME is the key used if the weights sought
;; are included in WEIGHTS.  KEY-NAME is also used for finding the
;; weights in the global variable *WEIGHTS* if they are not given
;; explicitly.  Explicit weights are either numbers or atoms such as
;; LOW or HIGH. The latter yield values from *WEIGHTS*.  If VALUE is
;; present, it is used as the default.

(define (find-weight weights key-name value)
  (let ((attrib (cadr (assq key-name weights))))
     (cond ((and attrib (number? attrib))
            attrib)
           (attrib
            (cdr (assq key-name
                       (cdr (assq attrib *weights*)))))
           (else value))))

;; A global variable specifying default weights for connections of
;; various types.

(define *weights*
        '((high (->parent . .5)
                (<-parent . .35)
                (->after . .4)
                (<-source . .5))
          (low (->parent . .35)
               (<-parent . .08))
          (very-low (->parent . .08)
                    (<-parent . .05))))

;; Analogous to FIND-WEIGHT.  Finds a decay value to be assigned to
;; a connection being created.  Uses the same key-names as FIND-WEIGHT.

(define (find-decay decays key-name default)
   (or (cadr (assq key-name decays))
       default
       *default-decay*))

(define *default-decay* .9)

;; These three functions return the components of the lists
;; representing connections.

(define connection:end car)
(define connection:weight cadr)
(define connection:type caddr)
(define connection:decay cadddr)
(define connection:speed cddddr)

;; Returns the other ends of all connections from SOURCE of type
;; CONN-TYPE.
```

```
(define (connection:ends/type node conn-type)
  (do ((conns (node-connections node) (cdr conns))
       (result nil
               (if (eq? (connection:type (car conns)) conn-type)
                   (cons (connection:end (car conns)) result)
                   result)))
      ((null? conns) result)))

;; Returns the connection type for the connection from SOURCE to
;; DEST.

(define (connection:type/s-d source dest)
  (connection:type (assq dest (node-connections source))))

;; Is NODE1 directly connected to NODE2?

(define (connected? node1 node2)
  (any (lambda (c) (eq? (connection:end c) node2))
       (node-connections node1)))

;; Is CONNection list a slow connection?

(define (slow-connection? connection)
  (not (connection:speed connection)))

;; Functions for accessing components of an activation-list.

(define actv-list:activation car)
(define actv-list:decay cadr)
(define actv-list:source caddr)

;; Returns the "end" node associated with "start" sequence node NODE.

(define (seq-end node)
  (cadr (assq 'seq-end (node-attribs node))))

;; Returns the elements of the winner-take-all network with hub HUB.

(define (wta-elems hub)
  (connection:ends/type hub 'wta-elem))

;; Gives the important connections from NODE and, recursively, from
;; the roles of NODE.  Used only in debugging.

(define (describe nd . no-roles)
  (let ((node (get-node nd)))
    (msg "Description of " node t)
    (do ((e (node-connections node) (cdr e))
         (rls nil)
         (pars nil)
         (equivs nil)
         (afters nil)
         (inhib nil))
        ((null? e)
         (if pars (msg 3 "Parents: " pars t))
         (if equivs (msg 3 "Equivs: " equivs t))
         (if rls (msg 3 "Roles: " rls t))
         (if inhib (msg 3 "Inhibits: " inhib t))
```

```
          (if afters (msg 3 "Afters: " afters t))
          (if (and (not no-roles) rls)
              (walk describe rls))
          (let ((end (seq-end node)))
            (if end (describe end t))))
        (case (connection:type (car e))
          ((parent ins-parent map-parent) (push pars (caar e)))
          ((role) (push rls (caar e)))
          ((map-parent) (set m-par (caar e)))
          ((equiv ins-equiv) (push equivs (caar e)))
          ((inhibit) (push inhib (caar e)))
          ((after) (push afters (caar e)))
          (else t)))
      node))

;; Creates a name for the map of PARENT for OWNER.  The name consists
;; of the name of OWNER and the name of PARENT (or PARENT's parent if
;; PARENT is itself a map) separated by a colon.

(define (map-name head par . parent-node)
   (let ((parent (cond ((car parent-node))
                       ((node? par) par)
                       (else (get-node par))))
         (parent-name (if (node? par) (node-name par) par))
         (head-name (if (node? head) (node-name head) head)))
     (concatenate-symbol head-name ':
                         (if (assq 'map (node-attribs parent))
                             (symbol-after parent-name 58)
                             parent-name))))

(define (symbol-after sym ch)
   (string->symbol (string-after (symbol->string sym)
                                 (ascii->char ch))))

(define (string-after s ch)
  (iterate -*- ((curr-str s))
      (cond ((string-equal? curr-str "") s)
            ((char= (char curr-str) ch)
             (string-after (chdr s) ch))
            (else (-*- (chdr curr-str))))))

;; Checks to see whether there is already a node called NAME.
;; If so, asks the user whether she wants to redefine it.  If the
;; response is YES (or Y), the old node is killed are the new node
;; created with the name.

(define (check-name name)
  (or (not name)
      (let ((node (get-node name)))
        (cond ((not (node? node))
               t)
              ((or *redefine?*
                   (and (msg name " is already a node" t
                             "Do you want to redefine it?  ")
                        (user-response)))
               (kill-node! node))
              (else nil)))))
```

```
(define (user-response)
   (let ((r (read (terminal-input))))
     (case r
     '     ((no n) nil)
           (else r))))

;; When *REDEFINE?* is true, nodes are redefined without the user's
;; being asked.

(lset *redefine?* nil)

;; Checks whether ARG is a node.  If not returns NIL.

(define (check-node arg)
  (and arg
       (if (not (node? arg))
           (block (msg arg " is not a node" t) nil)
           t)))

;; Makes NODE "executable" and makes ACTION its "action" (prints out
;; when the node executes).

(define (executable node action)
   (set (node-attribs node)
        (cons (cons 'action action)
              (node-attribs node))))

;; Adds ATTRIB to NODE's attribute list.

(define (add-attrib node attrib)
  (push (node-attribs node) attrib))

;; Does ROLE belong an INDIVIDUAL concept, at any level?

(define (role-of-indiv? role)
  (let ((heads (connection:ends/type role 'head)))
    (or (and (not heads)
             (assq 'indiv (node-attribs role)))
        (and heads (any role-of-indiv? heads)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(herald conc_memory)

;;; This file creates portions of the memory network for some basic
;;; conceptual knowledge.  In no sense is this meant to include a
;;; complete representation of any concept (if such a thing is
;;; possible).

;; CONCEPT, the most general node in the network.

(schema! 'concept nil
     '(type-role role ())
     '(type-role prop role)
     '(type-role part role))
```

```
(push (node-attribs (get-node 'role)) '((role)))

;; FACTs are CONCEPTs with ARGs, one of which is the OBJ of the FACT.

(schema! 'fact 'concept
     '(weights (->parent very-low))
     '(type-role arg role)
     '(role obj arg
          (weights (->head .1)))
     '(role modality prop))

;; STATEs are a kind of fact.  Note that nothing is said here about
;; what makes a STATE a STATE.  This will be true for many of the
;; concepts in the simple network created here.

(schema! 'state 'fact
     '(weights (->parent low)))

;; A TIME may be either an INSTANT or a PERIOD, the latter having an
;; associated BEGINNING, END, and DURATION.

(schema! 'time 'concept
     '(weights (->parent very-low)))

(schema! 'instant 'time)

(schema! 'period 'time
     '(role beginning part
          (is-a instant))
     '(role end part
          (is-a instant)
          (distinct beginning))
     '(role duration prop))

;; PLACEs, CONTEXTs, and THINGs are other very general types of
;; CONCEPTs.

(schema! 'place 'concept
      '(weights (->parent .05)))

(schema! 'context 'concept
     '(weights (->parent very-low))
     '(role context-location prop
          (is-a place)))

(schema! 'thing 'concept
     '(weights (->parent very-low)))

;; MATTER is a THING with an associated LOCATION and COLOR
;; (among many other omitted roles).

(schema! 'matter 'thing
     '(weights (->parent very-low))
     '(role location prop
          (is-a place))
     '(role color prop))

;; NAMED things have NAMEs.
```

```
(schema! 'named 'thing
     '(role name prop))

;; WEIGHT-VALUEs are abstractions not associated with any
;; particular object.

(schema! 'weight-value 'concept)

;; A PHYS-OBJ is a kind of MATTER with a set of properties
;; including a WEIGHT, an INSIDE, an OUTSIDE, and a VICINITY.

(schema! 'phys-obj 'matter
     '(role weight prop
          (is-a weight-value))
     '(role inside prop
          (is-a place))
     '(role outside prop
          (is-a place))
     '(role vicinity prop
          (is-a place)))

(schema! 'substance 'matter)

;; ANIMALs are PHYS-OBJs.  Note that levels of the hierarchy may be
;; left out, e.g., ORGANISMs.

(schema! 'animal 'phys-obj
    '(weights (->parent low)))

;; HUMANs are ANIMALs with names, Knowledge-BASEs, and sets-of
;; KNOWN-CONCepts, among other things.

(schema! 'human 'animal
     '(weights (->parent very-low))
     '(is-a named)
     '(type-role k-base prop))

;; NON-HUMAN-ANIMALs are distinct from HUMANs.

(schema! 'non-human-animal 'animal
     '(distinct human))

;; WOMEN and MEN are distinct.

(schema! 'woman 'human)

(schema! 'man 'human
     '(distinct woman))

;; CHIE is an individual WOMAN.

(schema! 'chie 'woman
     '(indiv))

;; ATTRIBUTEs are STATEs associated with an ATTRIBUER.

(schema! 'attribute 'state
```

```
        '(weights (->parent very-low))
        '(role attrib-obj arg
             (is-a thing)))

;; EXPERIENCEs are FACTS with a HUMAN EXPERIENCER argument.

(schema! 'experience 'fact
      '(weights (->parent very-low))
      '(role experiencer arg
           (is-a human)))

;; Having a concept ACTIVE is a kind of EXPERIENCE (the one you
;; have when you hear that concept referred to).

(schema! 'active 'experience
      '(weights (->parent low)))

;; PROXimity is a kind of STATE.

(schema! 'prox 'state
      '(weights (->parent very-low))
      '(type-role prox-obj arg
           (is-a phys-obj)))

;; CONTROL is a state with a HUMAN CONTROLLER argument.

(schema! 'control 'state
      '(weights (->parent very-low))
      '(role controller arg
           (is-a human))
      '(map obj
           (is-a phys-obj
                (weights (->parent .1) (<-parent .05)))))

;; BELIEVE is a STATE with a HUMAN BELIEVER argument and a
;; FACT as OBJ.

(schema! 'believe 'state
      '(weights (->parent very-low))
      '(role believer arg
           (is-a human))
      '(map obj
           (name believed)
           (is-a fact
                (weights (->parent .1) (<-parent .05)))))

;; KNOW-OF is a STATE with a HUMAN KNOWER for the OBJ is a
;; KNOWN-CONC.  If some of these "definitions" seem circular, this
;; is because they are.

(schema! 'know-of 'state
      '(weights (->parent very-low))
      '(role knower arg
           (is-a human))
      '(map obj
           (name known)
           (is-a thing)
           (is-a (knower k-base))))
```

```
;; LOCATED is a STATE with OBJECT and a LOC arguments.

(schema! 'located 'state
     '(weights (->parent very-low))
     '(map obj
          (is-a matter
               (weights (->parent .1) (<-parent .05))))
     '(role loc arg
          (is-a place)
          (merge (located:obj matter-location))))

;; An EVENT has a CONTEXT, a set of EXPANSIONs, a LOCATION,
;; a set of PRECONDIONS, some EFFECTS (STATES), and a CONSEQUENCE
;; (an EVENT).

(schema! 'event 'fact
     '(weights (->parent low))
     '(role event-context prop
          (is-a context))
     '(type-role expansion prop
          (is-a event))
     '(role event-location prop
          (is-a place))
     '(type-role precond prop
          (is-a state))
     '(type-role effect prop
          (is-a state)
          (distinct precond))
     '(role main-effect effect)
     '(type-role side-effect effect)
     '(role consequence prop
          (is-a event)))

;; An ACT is an EVENT with an ACTOR.

(schema! 'act 'event
     '(weights (->parent low))
     '(type-role participant arg
          (is-a human))
     '(role actor participant)
     '(map expansion
          (is-a act)))

;; The NULL ACT.

(schema! '0act 'act
     '(action ""))

;; PHYSICAL-TRANSFER, a kind of ACT with a SOURCE and a
;; DESTINATION.  The MAIN-EFFECT of a PTRANS is that
;; the OBJECT is now LOCATED at the DESTINATION.

(schema! 'ptrans 'act
     '(weights (->parent low))
     '(map actor
          (weights (<-parent .1) (->parent .3)
                   (->head .15)))
```

```
     '(map obj
          (weights (->parent low) (<-parent .1)
                   (<-head .1) (->head .15))
          (is-a matter))
     '(role source arg
          (is-a place) )
     '(role destination arg
          (is-a place)
          (refractory -2))
     '(map main-effect
          (is-a located)
          (map located
               (ins-equiv ptrans:obj))
          (map loc
               (ins-equiv destination))))

;; A TRANSFER-OF-CONTROL is an ACT with a
;; CONTROL-SOURCE and a RECIPIENT.  The MAIN-EFFECT
;; is that the OBJECT is CONTROLled by the RECIPIENT.

(schema! 'trans-of-control 'act
     '(weights (->parent low))
     '(map actor)
     '(map obj
          (weights (->parent low)))
     '(role c-source arg)
     '(role recipient arg)
     '(map main-effect
          (is-a control)
          (map controller
               (ins-equiv trans-of-control:recipient))
          (map control:obj
               (ins-equiv trans-of-control:obj))))

;; Other primitive acts.

(schema! 'grasp 'act
     '(weights (->parent low)))
(schema! 'ingest 'act
     '(weights (->parent low)))
(schema! 'expel 'act
     '(weights (->parent low)))
(schema! 'apply-force 'act
     '(weights (->parent low)))
(schema! 'mbuild 'act
     '(weights (->parent low)))
(schema! 'move-body-part 'act
     '(weights (->parent low)))
(schema! 'attend 'act
     '(weights (->parent low)))

;; PHYSICAL-TRANSFER where the ACTOR and OBJECT are the same,
;; as with "GO".

(schema! 'self-ptrans 'ptrans-act
     '(map ptrans:actor
          (weights (<-parent .1))
          (merge (self-ptrans ptrans:obj)
```

```
                (weights (<-parent .1)))))

;; PHYSICAL-TRANSFER where the ACTOR and OBJECT are different,
;; as with "TAKE".

(schema! 'other-ptrans 'ptrans-act
     '(map ptrans:obj
          (weights (->head .25)))
     '(map ptrans:actor
          (distinct other-ptrans:obj)))

;; PERCEIVE is an EXPERIENCE.

(schema! 'perceive 'experience
     '(weights (->parent low))
     '(role perceive->know effect
          (is-a know-of))
     '(map experiencer
          (name perceiver)
          (merge (perceive->know knower)))
     '(map obj
          (name perceived)
          (weights (->parent low))
          (distinct perceiver)
          (ins-equiv (perceive->know known))))

;; SEE is a kind of PERCEIVE

(schema! 'see 'perceive
     '(weights (->parent low)))

;; WANT is a STATE with a WANTER.

(schema! 'want 'state
     '(weights (->parent very-low))
     '(role wanter arg
          (is-a human))
     '(map obj
          (name wanted)
          (weights (->parent low)))
     '(role motivation prop
          (is-a event)))

;; UNDERTAKE is a kind of WANT.  The
;; OBJECT is an ACT.

(schema! 'undertake 'want
     '(weights (->parent low))
     '(map wanted
          (is-a act
               (weights (->parent .15))))
     '(map wanter
          (merge (undertake:wanted actor))))

;; WANT with STATE as its OBJECT.

(schema! 'want-state 'want
     '(weights (->parent low))
```

```
     '(map wanted
          (is-a state)))

;; INTEND has a PLANNER, a GOAL, and a
;; PLAN, which is an ACT that satisfies the
;; GOAL of the PLANNER.

(schema! 'intend 'act
     '(weights (->parent very-low))
     '(map actor
          (name planner)
          (weights (<-head .3) (<-parent .1)))
     '(role plan expansion)
     '(role goal arg
          (is-a state)))

;; AGENCY is a kind of INTEND with an ASSIGNEE
;; and an ASSIGNMENT.  The PLANNER's GOAL is that
;; the ASSIGNEE UNDERTAKE the ASSIGNMENT.

(schema! 'agency 'intend
     '(map goal
          (is-a undertake))
     '(role assignment arg
          (is-a act)
          (merge (agency:goal undertake:wanted)))
     '(role assignee arg
          (is-a human)
          (weights (<-head .25))
          (merge (agency:goal undertake:wanter))
          (merge (assignment actor)
               (weights (<-parent .1))))
     '(map planner
          (weights (<-head .3))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(herald ling_memory)


;; Schemas for WORD and UTTER.  Here mainly for completeness.

(schema! 'word 'act
     '(weights (->parent .1)))

(seq-schema! 'utter 'act
     '(seq-role constituent part)
     '(map actor
          (is-a human
               (weights (->parent .08) (<-parent .035)))
          (name speaker)
          (weights (<-head .1) (->head .1)))
     '(role hearer arg
          (is-a human)
          (weights (->head .1)))
     '(role content prop)
```

```
     '(seq-role head constituent
          (weights (<-head .1))
          (is-a word)))

;; ADJECTIVE-PHRASE GU.  Currently this has only one
;; constituent, the ADJECTIVE.  Thus phrases like "awfully old"
;; aren't handled.

(seq-schema! 'adj-phrase 'utter
     '(weights (->parent .15) (<-parent low))
     '(map content
          (weights (->parent .1))
          (is-a attribute))
     '(seq-role adj constituent
          (first)
          (last)))

;; NP GU

(seq-schema! 'np 'utter
     '(refractory -5)
     '(weights (->parent .15) (<-parent low)
               (end->start 0))
     '(map hearer                    ;; NP:HEARER has a low refractory
          (weights (<-head .5))      ;; because it has to fire often.
          (refractory -3)            ;; There is also a "perceptual
          (->perception))            ;; routine" which yields the
     '(map content                   ;; current value of HEARER
          (refractory -4)            ;; (actually gets it from the user).
          (weights (->head .01) (<-head .5) (->parent .1))
          (wta np-lex                          ;; WTA hub node for network
               (life 6))                       ;; joining NP CONTENT roles
          (is-a thing
               (weights (->parent .1))))
     '(seq-role det constituent          ;; DETERMINER:
          (first)                        ;;
          (decays (end->start .7))       ;; the end node is the WTA
parent
          (end                           ;; for a network joining
               (wta after-det-wta)))     ;; NOUN/start and ADJ-MOD/start
     '(seq-role adj-mod constituent      ;; ADJECTIVE-MODIFIER:
          (follows det)                  ;;
          (optional)                     ;;
          (wta-member after-det-wta)     ;;
          (seq-is-a adj-phrase)          ;; an instance of ADJ-PHRASE
          (map adj-phrase:content        ;; the ATTRIB-OBJ of its
               (weights (->head .34))    ;; CONTENT is the same as the
               (map attrib-obj           ;; CONTENT of the whole NP
                    (weights (->head .5))    ;; (in "the empty box" the
                    (ins-equiv np:content))));; argument of the EMPTY
     '(seq-role noun constituent              ;; predicate is the box
          (weights (<-head .3))
          (decays (<-head .98))
          (follows adj-mod
               (weights (->after high)))
          (follows det)
          (wta-member after-det-wta)
          (last)))
```

```
;; CLAUSE GU

(seq-schema! 'clause 'utter
     '(weights (->parent .15) (<-parent low) (end<-parent-end .05)
               (start->end .1))
     '(seq-role subject constituent          ;; SUBJECT
          (weights (end->head-end .05))      ;; no order specified in this
          (seq-is-a np))                     ;; GU
     '(seq-role aux constituent
          (weights (end->head-end .05))
          (is-a word))
     '(seq-role verb constituent             ;; VERB follows both SUBJECT
          (follows subject                   ;; and AUX
               (weights (->after .25)))
          (follows aux
               (weights (->after .25))))
     '(map content
          (weights (->parent .1))
          (is-a fact
               (weights (->parent .1)))))

;; DECLARATIVE clause; SUBJECT first

(seq-schema! 'declarative 'clause
     '(seq-map subject
          (first)))

;; YES-NO-QUESTION; AUX precedes SUBJECT

(seq-schema! 'y-n-question 'clause
     '(seq-map aux
          (first))
     '(seq-map subject
          (weights (->head .15))
          (follows y-n-question:aux
               (weights (->after .45))))
     '(seq-map verb
          (weights (->head .15))
          (follows y-n-question:subject
               (weights (->after .45)))))

;; TRANSITIVE-CLAUSE; DIRECT-OBJECT defined, following VERB

(seq-schema! 'transitive-clause 'clause
     '(seq-map verb
          (weights (->head .15)))
     '(seq-role dir-obj constituent
          (weights (->head .15) (start->end .25))
          (seq-is-a np
               (weights (<-parent .15) (end<-parent-end .45))
               (decays (end<-parent-end .3)))
          (follows transitive-clause:verb
               (weights (->after .4)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
(herald memory1)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                     ;;
;;                   MEMORY NEEDED TO GENERATE                         ;;
;;       "COULD YOU TAKE THE EMPTY BOX TO THE GARAGE?"                 ;;
;;                                                                     ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                   SOME CONCEPTUAL KNOWLEDGE                         ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; JOHN and his KNOWLEDGE-BASE.

(schema! 'john 'man
     '(indiv)
     '(map k-base))

;;; Some semantic and episodic knowledge.

(schema! 'house 'phys-obj)

;; CHIES-HOUSE with OUTSIDE and INSIDE roles.

(schema! 'chies-house 'house
     '(indiv)
     '(map inside
           (weights (<-parent .1)))
     '(map outside
          (map part
               (weights (->head .5))))
     '(map vicinity
          (map part
               (weights (->head .5)))))

(schema! 'garage 'phys-obj)

;; CHIES-GARAGE, with a role for its INSIDE
;; and is-a connections to CHIES-HOUSE:OUTSIDE:PART,
;; CHIES-HOUSE:VICINITY:PART, and JOHN's KNOWLEDGE-BASE.

(schema! 'chies-garage 'garage
     '(indiv)
     '(map inside
          (weights (->head .5)
                   (<-parent .1)))
     '(is-a chies-house:outside:part
          (weights (->parent .5)))
     '(is-a chies-house:vicinity:part
          (weights (->parent .5)))
     '(is-a! john:k-base
          (weights (->parent .35) (<-parent .1))))
```

```
(schema! 'med-wt-obj 'phys-obj)

(schema! 'box 'phys-obj)

(schema! 'able 'state
     '(weights (->parent low)))

;; EMPTY and its ATTRIBUTE-OBJECT.

(schema! 'empty 'attribute
     '(map attrib-obj
          (weights (->head .5) (<-head .05))))

;; BOX4, which is EMPTY and a MEDIUM-WEIGHT-OBJECT
;; and in JOHN's KNOWLEDGE-BASE.

(schema! 'box4 'box
     '(indiv)
     '(is-a empty:attrib-obj
          (weights (->parent .5)))
     '(is-a med-wt-obj
          (weights (->parent .5)))
     '(is-a john:k-base
          (weights (->parent .35) (<-parent .1))))

;; CHIE is currently at CHIES-HOUSE:INSIDE

(equiv! (+map! 'chie 'location)
        (get-node 'chies-house:inside)
        nil)

;; An instance of OTHER-PTRANS with JOHN as ACTOR,
;; BOX4 as OBJECT, CHIES-HOUSE:INSIDE as SOURCE, and
;; CHIES-GARAGE:INSIDE as DESTINATION.

(schema! 'ptrans6 'other-ptrans
     '(indiv)
     '(weights (<-parent .2))
     '(map other-ptrans:actor
          (equiv john))
     '(map other-ptrans:obj
          (equiv box4))
     '(map source
          (equiv chies-house:inside))
     '(map destination
          (equiv chies-garage:inside)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                          LINGUISTIC  KNOWLEDGE                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;; Some words

(schema! '/garage 'word
     '(exec "garage"))
```

```
(schema! '/box 'word
     '(exec "box"))

(schema! '/empty 'word
     '(exec "empty"))

(schema! '/the 'word
     '(exec "the"))

(schema! '/take 'word
     '(exec "take"))

(schema! '/bring 'word
     '(exec "bring"))

(schema! '/out 'word
     '(exec "out"))

(schema! '/in 'word
     '(exec "in"))

(schema! '/to 'word
     '(exec "to"))

(schema! '/you 'word
     '(exec "you"))

(schema! '/could 'word
     '(exec "could"))


;; The GU for "YOU".  Does not include the
;; HEARER-CONTENT equivalence.

(seq-schema! '*you 'np
     '(map np:content
          (wta-member np-lex))
     '(seq-map noun
          (weights (<-head .5))
          (only)
          (seq-is-a /you)))


;; GU for general facts of the OTHER-PTRANS type
;; (physical transfer where object is not actor).
;; Has as subtypes *BRING/PTRANS and *TAKE/PTRANS.
;; CONTENT:DESTINATION and SPEAKER roles are activated
;; to determine whether *BRING/PTRANS should be selected
;; (if the DESTINATION is the LOCATION of the SPEAKER).

(seq-schema! '*other-ptrans 'transitive-clause
     '(wta take-bring-wta              ;; the WTA for *TAKE/PTRANS
          (life 5))                    ;; and *BRING/PTRANS
     '(seq-is-a clause
          (weights (<-parent .15)))
     '(map speaker                     ;; this node is activated
          (weights (<-head .5))        ;; together with the
          (->perception))              ;; CONTENT:DESTINATION to
```

```
     '(map clause:content                ;; determine whether *BRING/PTRANS
          (weights (->head .45))         ;; should be selected
          (decay .95)
          (is-a other-ptrans
               (decays (<-parent .95)))
          (map destination
               (weights (<-head .5)
                        (->head .1))))
     '(seq-map subject
          (weights (<-head .45) (->head .1))
          (map np:content
               (weights (<-head .5))
               (merge (*other-ptrans:content  ;; role binding
                       other-ptrans:actor)    ;; information for
                    (weights (<-parent .1)    ;; the SUBJECT
                             (<-head .05)
                             (->head .05)))))
     '(seq-map dir-obj
          (weights (<-head .45) (->head .1))
          (decay .95)
          (map np:content
               (weights (<-head .5))
               (merge (*other-ptrans:content  ;; role binding
                       other-ptrans:obj)      ;; information for
                    (weights (<-parent .05)   ;; the DIRECT-OBJECT
                             (->head .05)))))
     '(seq-role destination-constit constituent    ;; the constituent
          (last)                                   ;; referring to the
          (decays (<-head .98))                    ;; DESTINATION
          (follows *other-ptrans:dir-obj
               (weights (->after .44)))
          (seq-role destination-marker constituent ;; the preposition
               (first)                             ;; "TO"
               (seq-is-a /to))
          (seq-role destination-np constituent     ;; the NP part of the
               (last)                              ;; constituent
               (weights (start->end .1))
               (seq-is-a np)
               (follows destination-marker
                    (weights (->after 44)))
               (map np:content
                    (weights (<-head .5))
                    (ins-equiv *other-ptrans:content:destination)
                    (weights (<-parent .1))))))

;; The default for the WTA network it shares with *BRING/PTRANS.

(seq-schema! '*take/ptrans '*other-ptrans
     '(decays (<-parent .9))
     '(wta-member take-bring-wta)
     '(weights (<-parent .25))
     '(seq-map verb
          (weights (end->start -.2))
          (only)
          (seq-is-a /take)))

;; Shares a WTA network with *TAKE/PTRANS.
```

```
(seq-schema! '*bring/ptrans '*other-ptrans
     '(decays (<-parent .9))
     '(wta-member take-bring-wta
          (weights (<-hub .2)))
     '(seq-map verb
          (weights (end->start -.2))
          (only)
          (seq-is-a /bring))
     '(map *other-ptrans:content
          (map destination                    ;; the information used to
               (weights (<-parent .24))       ;; select this GU over
               (merge (*bring/ptrans          ;; *TAKE/PTRANS
                       speaker location)
                    (weights (<-parent .24))))))

;; GU for NPs with "THE".
;; Selected on the basis of the CONTENT being in the HEARER's
;; KNOWLEDGE-BASE.

(seq-schema! 'the-np 'np
     '(weights (->parent .4))
     '(decays (->parent .3))
     '(map hearer
          (weights (->head .05))
          (map k-base
               (refractory -3)))
     '(map np:content
          (weights (->parent .1) (->head .4))
          (decays (->head .2))
          (ins-is-a the-np:hearer:k-base
               (weights (<-parent .5))))
     '(seq-map det
          (weights (<-head .45))
          (only)
          (seq-is-a /the)))

;; NP:HEARER sends some activation to THE-NP:HEARER:KNOWLEDGE-BASE.

(connect! (get-node 'np:hearer)
          (get-node 'the-np:hearer:k-base)
          'prime .3 *default-decay* '(fast))

;; The *YOU GU inhibits the THE-NP GU

(connect! (get-node '*you) (get-node 'the-np) 'inhibit -.3
          .8 '(fast))

;; The *BOX GU.

(seq-schema! '*box 'np
     '(weights (->parent .4))
     '(map np:content
          (weights (->head .5))
          (wta-member np-lex)
          (is-a box))
     '(seq-map noun
          (decays (<-head .98))
          (only)
```

```
          (seq-is-a /box)))

;; the *GARAGE GU

(seq-schema! '*garage 'np
     '(map np:content
          (weights (->head .45))
          (wta-member np-lex)
          (is-a garage))
     '(seq-map noun
          (decays (<-head .98))
          (only)
          (seq-is-a /garage)))

;; the GU *EMPTY, a subtype of ADJECTIVE-PHRASE

(seq-schema! '*empty 'adj-phrase
     '(map adj-phrase:content
          (weights (<-head .35) (->head .45))
          (is-a empty
               (weights (<-parent .45))))
     '(seq-map adj
          (weights (<-head .45))
          (only)
          (seq-is-a /empty)))

;; a very specific request GI

(schema! 'request-husband-take-out-heavy-obj 'agency
     '(weights (<-parent .08))
     '(seq-map plan                                  ;; the utterance PLAN:
          (weights (->head .5) (<-head .5))          ;; a YES-NO-QUESTION with
          (only)                                     ;; SUBJECT an instance of
          (is-a y-n-question)                        ;; *YOU and AUXILIARY
          (seq-map subject                           ;; "COULD"
               (seq-is-a *you))
          (seq-map aux
               (seq-is-a /could)))
     '(map planner                                   ;; the PLANNER is CHIE
          (weights (->head .2) (<-head .15))
          (equiv chie)
          (merge (request-husband-take-out-heavy-obj:plan speaker)
               (weights (->head .1) (<-parent .25))))
     '(map assignee                                  ;; the ASSIGNEE is JOHN
          (weights (->head .2) (<-head .05))
          (equiv john)
          (merge (request-husband-take-out-heavy-obj:plan hearer)
               (weights (<-head .05) (<-parent .05)
                        (->head .1))))
     '(map assignment
          (weights (<-parent .1) (->head .2) (<-head .15))
          (is-a other-ptrans                         ;; the ASSIGNMENT is an
               (weights (<-parent .1))               ;; instance of OTHER-
               (decays (<-parent .95)))              ;; PTRANS
          (map other-ptrans:actor                    ;; with ACTOR JOHN,
               (weights (->head .1) (<-head .15));;
               (decays (->head .95))                 ;;
               (equiv john))                         ;;
```

```
        (map other-ptrans:obj                          ;; OBJECT a
             (weights (->head .1) (<-head .15));; MEDIUM-WEIGHT-
             (decays (->head .95))                     ;; OJBECT
             (is-a med-wt-obj))                        ;;
        (map source                                    ;;
             (weights (->head .1) (<-head .15));; SOURCE the INSIDE OF
             (decays (->head .95))                     ;; CHIES-HOUSE, and
             (equiv chies-house:inside))               ;;
        (map destination                               ;; DESTINATION the
             (weights (->head .1) (<-head .15));; VICINITY of CHIES-
             (decays (->head .95))                     ;; HOUSE
             (equiv chies-house:vicinity))
        (is-a able)
        (merge (request-husband-take-out-heavy-obj:plan
          clause:content))))

;; Input to generate "COULD YOU TAKE THE EMPTY TO THE GARAGE?"

;(propagate '((agency
;              assignee john
;              assignment ptrans6
;              ptrans6:actor
;              ptrans6:source)
;             (ptrans6:destination
;              ptrans6:obj
;              agency:planner chie)))
```