# A Flexible, Parallel Model of Natural Language Generation

Nigel Ward

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| | | |
|---|---|---|
| 1. REPORT DATE<br>**APR 1991** | 2. REPORT TYPE | 3. DATES COVERED<br>**00-00-1991 to 00-00-1991** |

| | |
|---|---|
| 4. TITLE AND SUBTITLE<br>**A Flexible, Parallel Model of Natural Language Generation** | 5a. CONTRACT NUMBER |
| | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>**University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT

**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

**This thesis describes a structured connectionist system for natural language generation. 'FIG', short for 'Flexible Increment Generator', is based on a single network which encodes lexical knowledge, syntactic knowledge, and world knowledge. In the initial state, some nodes representing concepts are sources of activation; this represents the input. Activation flows from these nodes to nodes representing words via the various knowledge structures of the network. When the network settles, the most highly activated word is selected and emitted, activation levels are updated to represent the new current state. This process of settle, emit, and update repeats until all of the input has been conveyed. An utterance is simply the result of successive word choices. Syntactic knowledge is encoded with network structures representing constructions and their constituents. Constituents are linked to words, syntactic categories, relations, and other constructions. Activation flow via these links, and eventually to words, provides for constituency and subcategorization. The links to constituents are gated by 'cursors,' which are updated over time, based on feedback from the words output. This mechanism ensures that words and concepts which are syntactically appropriate become highly activated at the right time, thus causing words to appear in the right order. FIG suggests that the complexity present in most treatments of syntax is unnecessary. For example, it does not assemble or manipulate syntactic structures; constructions affect the utterance only by the activation they transmit, directly or indirectly, to words. Unlike previous generators, FIG handles arbitrarily large inputs, since the number of nodes activated in the initial state makes no difference to its operation; it handles trade-offs among competing goals without additional mechanism, since all computation is in terms of commensurate levels of activation; and it handles interaction among choices easily, since it tends to settle into a state representing a compatible set of choices, due to links among nodes representing such choices. Abstracting from the implementation leads to several design principles for generation, including explicit representation of the current state and pervasive use of parallelism. The design is a weak cognitive model and also points the way to more natural machine translation.**

| 15. SUBJECT TERMS | | | | | |
|---|---|---|---|---|---|
| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **176** | |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std Z39-18

# A Flexible, Parallel Model of Natural Language Generation

Nigel Ward


## Abstract

This thesis describes a structured connectionist system for natural language generation. 'FIG', short for 'Flexible Incremental Generator', is based on a single network which encodes lexical knowledge, syntactic knowledge, and world knowledge. In the initial state, some nodes representing concepts are sources of activation; this represents the input. Activation flows from these nodes to nodes representing words via the various knowledge structures of the network. When the network settles, the most highly activated word is selected and emitted, activation levels are updated to represent the new current state. This process of settle, emit, and update repeats until all of the input has been conveyed. An utterance is simply the result of successive word choices.

Syntactic knowledge is encoded with network structures representing constructions and their constituents. Constituents are linked to words, syntactic categories, relations, and other constructions. Activation flow via these links, and eventually to words, provides for constituency and subcategorization. The links to constituents are gated by 'cursors,' which are updated over time, based on feedback from the words output. This mechanism ensures that words and concepts which are syntactically appropriate become highly activated at the right time, thus causing words to appear in the right order. FIG suggests that the complexity present in most treatments of syntax is unnecessary. For example, it does not assemble or manipulate syntactic structures; constructions affect the utterance only by the activation they transmit, directly or indirectly, to words.

Unlike previous generators, FIG handles arbitrarily large inputs, since the number of nodes activated in the initial state makes no difference to its operation; it handles trade-offs among competing goals without additional mechanism, since all computation is in terms of commensurate levels of activation; and it handles interaction among choices easily, since it tends to settle into a state representing a compatible set of choices, due to links among nodes representing such choices. Abstracting from the implementation leads to several design principles for generation, including explicit representation of the current state and pervasive use of parallelism. The design is a weak cognitive model and also points the way to more natural machine translation.

# Contents

**Acknowledgements**

Thesis advising Robert Wilensky / linguistic wisdom Charles J. Fillmore, George Lakoff / committee chairing Lotfi Zadeh / rewrites with a lot of help from Robert Wilensky, Daniel S. Jurafsky, Terry Regier, Narciso Jaramillo, Jane Edwards / home in exile Hirochika Inoue / money the National Science Foundation, via a Graduate Student Fellowship; the Sloan Foundation, via the Berkeley Cognitive Science Program; Monbusho, via a Kokuhi Shoogakukin; the Defense Advanced Research Projects Agency (DoD), monitored by the Space and Naval Warfare Systems Command under contracts N00039-84-C-0089 and N00039-88-C-0292; the Office of Naval Research under contract N00014-89-J-3205 / other the BAIR group, the CogSci Crowd, everyone else.

# Chapter 1

# Introduction



## 1.1 Motivation

### 1.1.1 Two Basic Problems

In an ordinary day an ordinary person says many things, without noticeable strain, and successfully communicates what he wants to. Yet language production is intrinsically a difficult task, as can be seen by considering some cases where aspects of the language production task must be dealt with consciously.

One such situation is editing and revising a paper. There are cases where after changing one word it may be necessary to rewrite the rest of the sentence or even the entire paragraph. This happens because there are complex dependencies among the various choices made in producing a text.

Another such situation is editing a text written by a non-native speaker. There are cases where, to determine whether a certain word choice is correct, for example *"a"* instead of *"the"*, it is necessary to spend several minutes to determine exactly what he wants to say or the nature of the thing he is describing. This difficulty arises because the choice of words and structures can depend on nuances.

These two types of difficulty, the problem of interdependencies among choices and the problem of choices dependent on nuances of the input, can be illustrated with the simple sentence:

1.1 *One day the old man went to the hills to gather wood.*

where the context is that this is a continuation of a story begun with *"Once upon a time there was an old man"*, and the meaning behind this sentence is that the old man set off from his home to perform the day's task, which was to gather twigs and branches and bring them home to use for cooking fuel.

As far as interaction among choices: note that *"gather"* is inflected correctly. In particular, it obeys the selectional restrictions of the purpose clause construction (unlike *"the old man went to the hills to* gathering *wood"*). Note also that *"gather"* refers to an action, which sounds natural in a purpose clause (unlike a state, as in *"the old man went to the hills to* have *wood"*). Note also that *"went ... to gather"* cleanly focuses the purpose of the motion (unlike *"walked ... to gather wood"* which tries to focus both the purpose and the manner, and sounds unnatural).

As far as choice depending on nuances of meaning: Note that *"gather"* successfully refers to the entire sequence of activities (unlike *"pick up"* in *"the old man went to the hills to* pick up *wood"*), and does so concisely (unlike *"the old man went to the hills to* look for, identify, pick up, break up, bundle together, and carry *wood"*). Note also that *"gather"* is also appropriate because it conveys the facts that the wood was initially scattered and was finally in one place (unlike *"the old man went to the hills to* move *wood"*), and conveys this naturally (unlike "move *the wood* together"). Note also that *"to gather wood"* concisely expresses that going to the hills was part of a plan to get wood (unlike a purpose clause with *"because"*, such as *"the old man went to the hills* because he wanted *wood"*, which expresses the reason for but not the planful nature of his journey). Note that *"went to the hills"* focuses on the journey (unlike *"went* into *the hills"* which highlights the destination more). Note also that *"the hills"* naturally expresses the fact that the old man was going to move around at the destination (unlike *"the hill"* or *"a hill"*, which could mean that his destination was a single point).

The basic motivation for this research is simply to explore how to generate well, paying attention to these two problems. The remainder of this section situates this goal with respect to previous research and with reference to specific generation tasks.

### 1.1.2 Scaling Up

Generation, the process of text production by computer, seems easy. For example, it is simple to map *'(kill John Mary)'* to *"John killed Mary"*. For such inputs one can ignore the issues of dependencies and nuances. Most generators do just this, and perform perfectly adequately in limited domains.

However one can foresee that in the future generators will have more knowledge, in particular, they will have more lexical and syntactic options available. If so, there will be many options for expressing 'each element of meaning', and the danger of these being incompatible with options for expressing 'nearby' elements of meaning will become more serious. In other words, if a generator knows more words and syntactic constructions then it cannot ignore dependencies.

One can also foresee that the programs which use generators will have more knowledge. This knowledge will need to be accessible to the generator; that is, the effective inputs to generators will be rich in information. A generator faced with such complex inputs will not be able to use simple strategies of directly mapping concepts to words and conceptual structures to syntactic structures.

In sum, the two issues raised in the previous section will become more pressing as generators and the programs which use them become more sophisticated. Existing architectures for generation are not likely to be easy to scale up to handle these problems. This point is discussed further in §2.4.

An important consideration in building larger generators is speed. Adding more knowledge to traditional generators makes them slower. For example, if they have more linguistic resources available, they need more time to choose which to use. Worse is the problem of dependencies among choices — there is a combinatorial explosion of possibilities to consider. This strongly suggests the need for parallel computation (§2.3.3). Indeed, another pervasive trend in AI is towards parallel algorithms and parallel machines. Traditional models of generation however are fundamentally serial in many ways, as argued in §2.4. Thus a major theme of this thesis is how to take advantage of parallelism.

In order to build a parallel system it helps to keep things as simple and as uniform as possible. Simplicity is accordingly also a major theme of this thesis.

### 1.1.3 Modeling Human Performance

The fact that generation is a difficult task is not evident to human speakers. Humans seem to have a special talent for dealing with interdependent choices and for choosing words appropriate to complex meanings. This thesis is in part a computational inquiry into the nature of this talent. (Although the title includes the word 'generation', since that is the tradition in AI, I could equally well have used the word 'production', the psycholinguistic term for the same activity.)

Thus one of the goals of this work is to propose a model of generation which is cognitively plausible. This means, first of all, explaining specific, quantifiable observed phenomena, such as pauses in spontaneous speech, priming effects, and speech errors. Second, it means building a model which includes nothing more than is necessary to account for the data; this has been another reason for the drive towards simplicity, and to determine which of the mechanisms traditionally used for generation are not really necessary. Third, building a cognitively plausible model means that the model must not violate obvious facts about human language production. The remainder of this section expands on this briefly.

One obvious fact is that humans produce language with a brain. This suggests that a good cognitive model should exploit massive parallelism, as the brain seems to.

The second obvious fact is that human language production is incremental. Consider the following utterance:

> *Nigel, you've got such a we-ird answering machine message. - So anyway we need to know what time we can come and rehearse tomorrow, - a-nd, - we're assuming that, - ten o'clock in the morning we'll show up at your place, — and, — or or shortly thereafter, - and uh - if we - if we don't hear from you then we'll do that, if you - can't do that then leave a message on my machine ok? - thanks bye.*

Notation:

- \- short pause or lengthened word
- — longer pause
- ,.? clause boundaries, based on intonation

After understanding the content, a second reading will reveal two things. First, speech is full of pauses, including filled pauses such as *"uh"* and *"oh"*. The obvious and pervasive explanation for this is that people think as they speak; in particular, that 'deciding' what to say next requires thought. In other words, the 'effort' of speaking is expended 'on-line'. Second, the example illustrates that speech exhibits false starts, that is, cases where the speaker utters a few words, 'denies them', and starts over. The obvious explanation for this is that speakers choose words without completely computing the consequences for future choices. In general, a cognitive model of generation should be 'incremental'. Incremental generation is also useful for other reasons, as discussed in §2.1.3.

### 1.1.4 Machine Translation

An anecdote here may help illustrate just how hard generation can be. When I first started thinking about generation I was advised to start from the outputs of an existing parser (Norvig 1987), and, since the input to the parser was English and I was interested in machine translation, I tried to make a system to generate Japanese. (Here and throughout I will use 'parser' to include programs which do semantic as well as syntactic interpretation.)

The first sentence analyzed was *"In a poor fishing village ... a young boy ... lived with his widowed mother"*, from which the parser produced a structure including an individual concept which was simultaneously an instance of the 'widow' concept and the 'mother' concept.

From this structure it proved impossible to straightforwardly produce a good Japanese sentence. The basic problem is that the word for widow, *"miboojin"*, cannot without awkwardness simply modify a noun such as *"okaasan"* (mother). Of course, it would be possible to splice in the definition of 'widow' as a relative clause, but that would be awkward and would put much more emphasis on widowhood than was present in the original. My informant considered this problem for a while, and finally cut the Gordian knot with the phrase *"okaasan to futari de sunde imashita"*, *"lived with his mother, (just) the two of them"*, which is a perfectly natural idiomatic phrase of Japanese. Of course the meaning is not precisely the same, but this phrase can serve the same role in the discourse, namely to invoke pity for the boy, and given the context and genre would lead the reader to infer that the mother is in fact a widow.

There are several lessons to draw from this, including:

- Generation has multiple potentially conflicting goals — for example, to express exactly the same information present in the input, to produce natural-sounding text, and to produce concise text.

- A conceptual structure which is perfectly natural and motivated to express the meaning of an utterance in one language may be very inconvenient for specifying an utterance in another language. Hence for machine translation, and perhaps more generally, a generator must not rely blindly on the structure of its input to determine the organization of its output.

- To generate an utterance that sounds natural a generator needs the flexibility to take advantage of words and concepts that are specific to or well suited to the target-language, such as *"futari"* in the example.

- To discover a good utterance may require consideration of the larger context. Thus the input to the generator cannot be simply a proposition in isolation.

- A generator may require vast amounts of knowledge — for example, information regarding the relationship between death, families, widowhood, piteousness, poverty, and sons.

In general, generation is more than a question of mapping the input to the output, tidying up inflections, inserting function words, and setting the word order. These points are discussed at length in Chapters 2 and 9.

### 1.1.5 Summary of Motivations

This thesis explores the computational problem of natural language generation, paying full attention to the issues raised above. In short, it explores the question of how to produce an utterance to satisfy a multitude of conflicting goals, by using some of a vast number of linguistic resources, the choices among which are interdependent and depend on nuances of meaning. To address this problem I have applied methods and data from artificial intelligence, computational linguistics, and psycholinguistics.

Strange to say, the aims of this inquiry are rather novel. The bulk of previous research in generation has not addressed these issues. In particular, it has gone for depth, rather than breadth; addressing specific issues intensely, for example syntax or text-planning, but not considering the way they interact. Nor has it addressed problems of generating from inputs rich in information. These points are made more precise in Chapter 2.

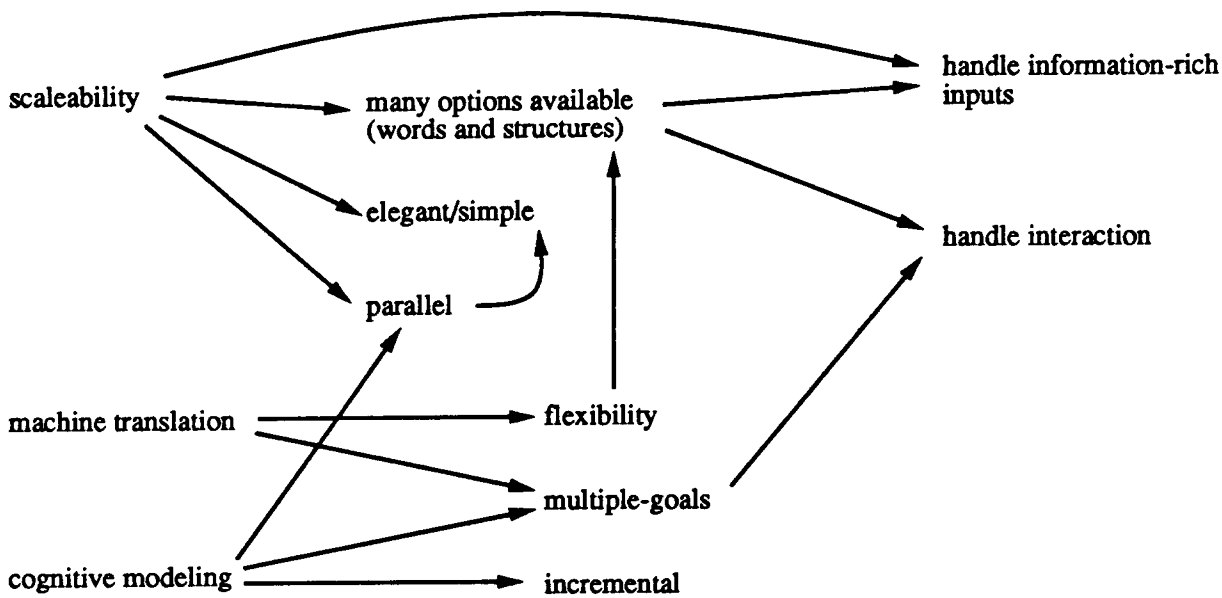


Figure 1.1: Relations Among the Motivations for This Research

Figure 1.1 summarizes the logic of this section. The arrows mean 'requires a generator to have/be/do'. One might think that these diverse motivations overconstrain the research task. Quite the contrary, they all seemed to lead in the same direction; all of them provided valuable hints which shaped the form of the final model, so much so that I regard my results to be in some ways more discoveries than inventions.

## 1.2 Preview of the Model

I have explored the above issues by writing a program and experimenting with it. The program I call 'FIG', since it is a 'Flexible Incremental Generator'. This thesis is to a large extent an explanation of this system and what I learned from it.

### 1.2.1 FIG's Task

FIG is a model of the production of spontaneous single-sentence narrative spoken utterances. There are many other generation tasks, but it seems that the same techniques apply, regardless of the communicative situation. This task is in some ways the simplest and most basic; the production of written text or careful speech, while having certain different qualities, is not enormously different (Chafe 1985). Also, it seems that polished writing can best be produced, as people do, in a two-stage process, where the first stage consists of "an architecture for generation that favors efficiency and expressibility over initial optimality of expression" (Meteer 1991). There are other issues in generation which do not appear in this task. FIG like most generators, produces text intended to convey a static meaning to a generalized, not a specific, hearer, and it has no need for revision, nor for haste.

This thesis focuses on issues of lexical choice and syntactic organization. FIG is not, however, a linguistic generator, narrowly conceived. Its inputs are intended to be more like representations of thoughts than descriptions of sentences. The significance of this distinction will become clear in §2.2.2.

FIG does not deal with issues in multi-sentence generation, morphology, or prosody. Nor does this thesis shed new light on the nature of the true facts about English or Language; my goal has been to produce a performance model, not a knowledge model. Nor have I attempted to give FIG a large number of words, nor a great inventory of syntactic knowledge, unless it seemed difficult or otherwise interesting to do so. In these respects this thesis does not open up new fields of inquiry, rather, it is an examination of some very basic issues in generation.

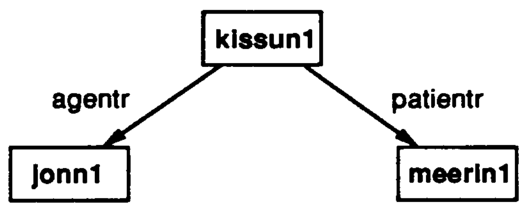


Figure 1.2: An Input to FIG

Inputs to FIG are assembled by hand or come from a simple parser of Japanese. From these representations FIG produces English and Japanese sentences. For simple cases, FIG's input-output behavior is much like that of any other generator. For example, from the input depicted in Figure 1.2, FIG produces, of course, *"John kissed Mary"*.

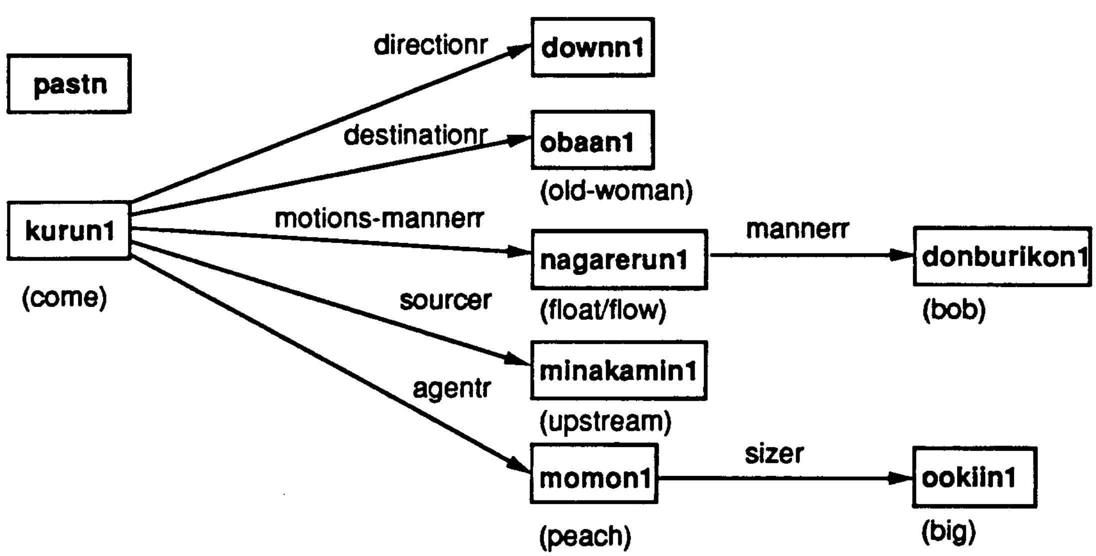


Figure 1.3: Another Input to FIG

There need not necessarily be a straightforward mapping from input to output; Figure 1.3 shows an input which causes FIG to produce *"a big peach bobbed down to an old woman from* *upstream*"; here the mapping from concepts of the input to words is not one-to-one. This is discussed further in §3.1.

Other outputs include:

> "*once upon a time there lived an old man and an old woman*"
> "*one day the old woman went to a stream to wash clothes*"
> "*a dish broke*"
> "*John broke a dish*"
> "*a cake vanished*"
> "*John made a cake vanish*"
> "*John killed Mary*"
> "*Mary was killed*"
> "*Mary was killed by John*"
> "*the wind broke a dish*"
> "*Mary died*"
> "*Mary kicked the bucket*"
> "*John made Mary kick the bucket*"
> "*an old man and an old woman ate a cake*"
> "*John went to Chicago and to a mountain*"
> "*John ate a peach from a stream in Chicago*"
> "*John ate a peach with a fork from Chicago*"
> "*John gave a big peach to Mary quickly*"
> "*John ate an old woman 's peach*"
> "*John 's very big peach was eaten*"
> "*the strong old woman ate John 's very big peach*"
> "*Mary saw John die*"
> "*the high winds*"
> "*the swift currents*"
> "*the old man went to a mountain*"
> "*the old man went into the hills to gather wood*"

And in Japanese:

> "*mukashi mukashi aru tokoro ni ojiisan to obaasan ga sunde imashita*"
> long-ago long-ago certain place LOC old-man and old-woman SUBJ live PAST-PROG
> once upon a time there lived an old man and an old woman
>
> "*aru hi obaasan wa kawa e sentaku ni ikimashita*"
> certain day old-woman TOPIC stream DEST wash-clothes PURP go-PAST
> one day the old woman went to a stream to wash clothes
>
> "*momo ga obaasan e nagarete kimashita*"
> peach SUBJ old-woman DIR float come-PAST
> a peach floated towards an old woman
>
> "*jon wa meeri ni kissu shimashita*"
> John TOPIC Mary RECIP kiss PAST
> John kissed Mary

*"ojiisan wa momo o meeri ni agemashita"*
old-man TOPIC peach PATIENT Mary RECIP give-PAST
the old man gave a peach to Mary

*"jon ga fooku de sara o kowashimashita"*
John SUBJ fork INSTRUMENT dish PATIENT break-TRANSITIVE-PAST
John broke a dish with a fork

*"sara ga kowaremashita"*
dish SUBJ break-INTRANSITIVE-PAST
the dish broke

*"ojiisan to obaasan ga keeki o tabemashita"*
old-man and old-woman SUBJ cake PATIENT eat-PAST
an old man and an old woman ate a cake

### 1.2.2 Overview of FIG

A simplistic but useful restatement of the generator's task is that it must get from concepts (what should be expressed) to words (what can be said). From this point of view the key problem in generation is computing the appropriateness of words given the concepts to express. FIG is accordingly centered on the task of word choice — every consideration is analyzed in terms of how it affects word choice. Syntactic and other knowledge mediates the computation of appropriateness of words, but that is its only role; in particular there is no explicit concept choice or syntax choice, nor any building of syntactic structure.

This idea extends naturally to generation as a process in time: FIG computes the appropriateness of each word for the current time. By doing this for each successive word choice it produces an appropriate utterance word by word.

The computational mechanism by which all this happens is the flow of 'activation' in an associative network. Network structures encode lexical, syntactic, and world knowledge. Concepts, words, and constructions are represented as nodes and relations among them are represented as links. Each node is a simple computational unit. The nodes have various amounts of 'activation', which change over time. The computation in the system is done by the individual units in parallel: based on the amounts of activation they receive, they compute their own next activation level and the amount of activation to transmit to their neighbors. This computation is simple and local.

The general direction of activation flow is from concepts to words, with nodes representing syntactic constructions 'redistributing' activation to words which are syntactically appropriate for the current situation. Constructions are represented as sets of nodes: the construction node itself is linked to nodes for constituents. These links have the special property that their weights change over time; in particular, the link to the currently relevant constituent has weight 1 and the links to the other constituents have weight 0. This is how FIG represents the current syntactic state. There is an update process which ensures that these weights are kept up to date as the generation of an utterance proceeds. The FIG algorithm is:

1. each node of the input conceptualization is a source of activation
2. activation flows through the network
3. when the network settles, the most highly activated word is selected and emitted
4. activation levels are updated to represent the new current state
5. steps 2 through 4 repeat until all of the input has been conveyed

An utterance is simply the result of the successive word choices made in each iteration of the loop; thus FIG is incremental in a strong sense. The five steps are summarized in Figure 1.4. Of

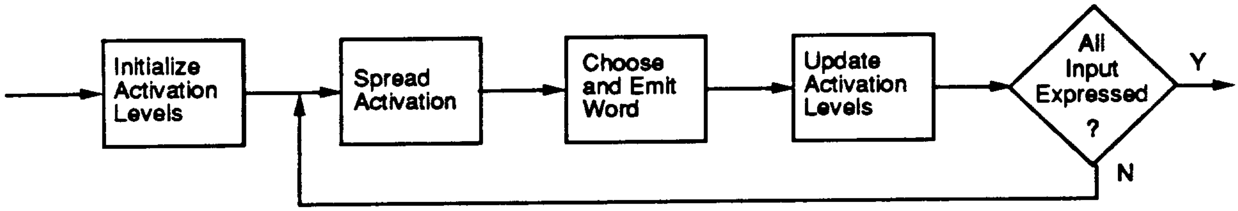


Figure 1.4: FIG Control Flow

course, this description of the control flow is not really very informative — what actually happens in each step depends on the structure of the network; this is explained in Chapters 3 and 4. Full details of each of the five steps are given in Chapter 6.

FIG is built as a spreading activation (or 'structured connectionist') system primarily for practical reasons, namely, that this technique allows parallelism, numeric combination of evidence, and easy integration, as set forth in §2.3 and Chapter 7. I have not been a purist, however; FIG includes symbolic processes for subtasks where the use of spreading activation has no advantages. This thesis is not intended to advance the state of the art in connectionist modeling in general.

To survey the results: The architecture sketched out above can cope with subtle influences of meaning on word choice, because any number of concept nodes can be linked to a word node. It scales up to handle large inputs since any number of nodes can be activated in the initial state. It handles interaction among choices easily, provided that the network structures are such that it tends to settle into a state representing a compatible set of choices. It can cope with multiple conflicting goals simply because all considerations are scored numerically. It is clearly highly parallel; and it is clearly incremental. These points are discussed in detail in subsequent chapters.

This approach also points the way to more natural machine translation. Existing systems are largely structure-bound, that is, the structure of the input closely governs that of the output. FIG is more flexible: there is no direct mapping from its input to its output. Thus it is more able to take advantage of the idiosyncrasies of the target language.

### 1.2.3 Notation Conventions

The names of nodes which represent 'notions' (concepts) generally end in 'n', syntactic constructions in 'c', constituents in '-$n$', where $n$ is a number, words in 'w', and relations in 'r'. The names of instances are generated by appending a number to the name of the corresponding generic node, following standard practice in knowledge representation, for example **yaman1** is an instance of the node **yaman** (mountain). The names of nodes and links are set in bold when they appear in the text.

Notions in FIG are usually given Japanese names. This may help remind the reader that the starting point for generation need not be something which is easy to map to the target language. The original reason is that most of FIG's inputs come from a parser of Japanese which does absolutely no more work than it has to. Node names are of course irrelevant to the operation of the program.

Graphs are used to represent portions of FIG's knowledge network, for example Figure 1.5 shows the relation between a node representing a notion and one representing a word. More often the network structures are shown in a frame-like notation, as in:

```
(defn momon (english (peachw .5)) )
```

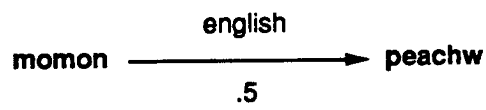


Figure 1.5: A Tiny Fraction of FIG's Knowledge

Data structures, such as this one, that are taken directly from FIG are shown in typewriter font. Function names are preceded by `#'`.

Examples of English and Japanese are in italics and set off by double quotes, as in *"the old man went"*; examples of Japanese are written in the roman alphabet, as in *"ojiisan wa ikimashita"*. Single quotes are used as 'scare quotes'.

### 1.2.4 A Worked Example

This subsection is a preliminary attempt to convince the reader that generation with spreading activation is possible. It describes how FIG produces *"the big boy ate Mary's peach with a fork in Chicago"*. There is nothing especially interesting about this example; it is used merely because it is relatively easy to explain.

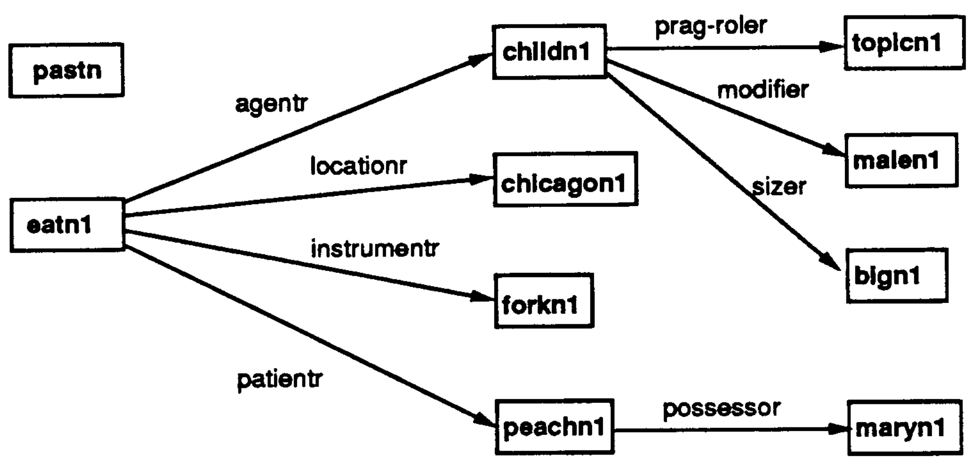


Figure 1.6: Yet Another Input for FIG

The input for this example is the set of nodes linked together as shown in Figure 1.6 (details of the input representation are given in §6.2). In this section all concept names are anglicized for the reader's convenience. Boxes are drawn around nodes in the input so that they can be easily identified in subsequent diagrams.

Initially each node of the input is a source of activation. Also each construction activates its first constitutent (constructions are described in Chapter 4). An important construction at this point is the Subject-Predicate Construction, represented by the node **subj-pred**. The first constituent of this construction has a link to **agent**, and so activation flows to **childn1**, since it is the agent in this input. Activation flows from both this and the input node **bign1** to **boyn**, and it becomes the most highly activated node. All this and more is shown in Figure 1.7. Many other nodes are, of course, also activated, and many other paths of activation flow exist but happen not to be important for this input. For example, **passivec**, representing the Passive Construction, has

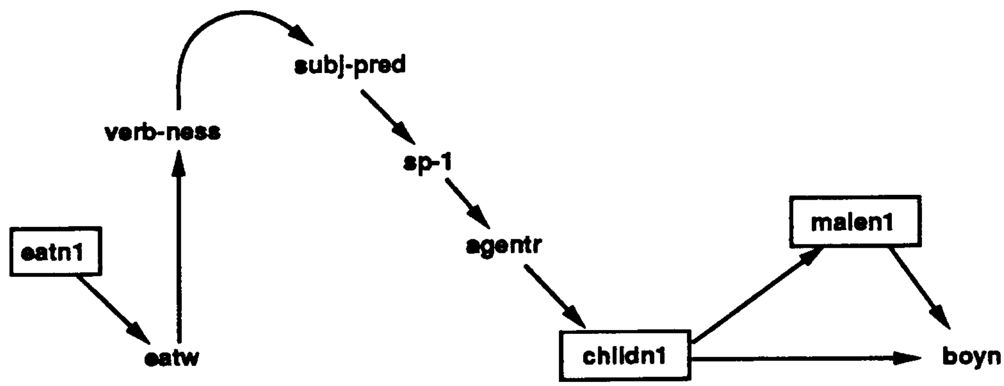


Figure 1.7: Selected Paths of Activation Flow Just Before Output of *"the"*

some small amount of activation at this point and is 'competing' (feebly) with **subj-pred**. Each construction activates words that it 'thinks' are currently appropriate; the conflict is resolved by the fact that FIG always selects the most highly activated word. Competition is discussed at length in §4.4.1.

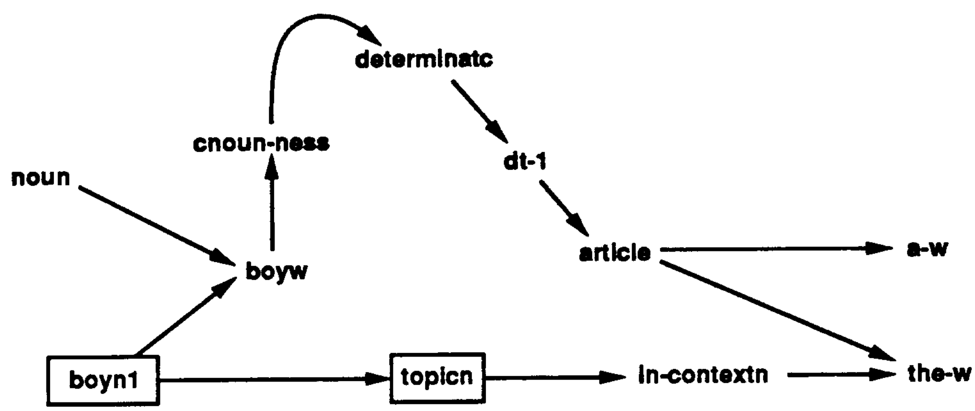


Figure 1.8: More Selected Paths of Activation Flow Just Before Output of *"the"*

At this point, therefore, **boyn** is the most highly activated concept at this point. The most highly activated word, however, is **the-w**; the reasons are shown in Figure 1.8. The basic idea is that the Determination Construction, represented as **determinatc**, ensures that articles become highly activated before common nouns. What actually happens is that activation flows from **boyn** to **boyw**, the node for the word *"boy"*, and, since **boyw** is a count noun, it activates **determinatc**, which passes activation via its first constituent to **article**. Activation flows from that syntactic category to its members, namely **a-w** and **the-w**. **the-w**, unlike **a-w**, also receives activation from **topicn**, and so it becomes the most highly activated of all word nodes.

Figure 1.9 shows some nodes of the network and their activation levels at this point, that is, after activation has flowed but before any word has been output. For each construction the relevant constituent is shown by capitalization; since generation has just begun, in each case the current constituent is the leftmost. The notions **shoonenn1** and **otokon1** are the ones that I have been anglicizing as **boyn1** and **malen1**, respectively. Notions whose names end in 'ness' can be ignored for now. As the figure shows, many words are currently significantly activated, but **the-w** has the most activation.

| —constructions— | ——words—— | —notions,insts— |
|---|---|---|
| 23.9 SUBJ-PRED | 22.2 THE-W | 25.0 THING-NESS |
| SP-1 sp-2 sp-3 | 20.8 A-W | 24.7 AGENT-NESS |
| 21.6 DETERMINATC | 20.1 BIGW | 24.2 CONSOI-NESS |
| DT-1 dt-2 | 20.1 BOYW | 24.2 ADJ-NESS |
| 21.4 ADJ-MODC | 19.7 CHILDW | 24.0 NOUN-NESS |
| AM-1 | 16.3 MALEW | 24.0 CNOUN-NESS |
| 20.1 PREP-PHR | 13.5 FORKW | 21.2 SHOONENN |
| PP-1 pp-2 | 11.3 EATW | 20.7 OOKIIN1 |
| 17.7 INTENSIFYC | 6.4 MARYW | 20.7 OTOKON1 |
| IN-1 in-2 | 6.1 CHICAGOW | —cats,rels— |
| 8.8 LOC-SETTNGC | 6.1 PEACHW | 16.0 ARTICLE |
| LS-1 | 3.0 IS2W | 15.3 ADJECTIVE |
| 8.4 GENITIVEC | — — | 14.7 NOUN |
| GE-1 ge-2 | — — | 11.5 VERB |

Figure 1.9: Activation Levels of Selected Nodes Just Before Output of *"big"*

FIG accordingly emits *"the"* and then updates the syntactic representation. In particular, the 'cursor' of **determinatc** advances to point to its second constituent and so it no longer activates articles. (The update process is described in in full in §4.2.2). **boyn** continues to be the most highly activated notion, for the same reasons as before. At this point the most highly activated syntactic category is **adjective**, due to activation from **adj-modc**, representing the Adjectival Modification Construction. This construction was also active previously, but its effects were overshadowed by the effects of **determinatc**. While all adjectives receive activation from **adjective**, **bigw** becomes the most highly activated word, since it alone receives activation also from a node present in the input, namely **bignl**. Figure 1.10 shows the state of the network at this point.

| —constructions— | ——words—— | —notions,insts— |
|---|---|---|
| 27.1 DETERMINATC | 19.2 BIGW | 24.8 THING-NESS |
| dt-1 DT-2 | 18.6 BOYW | 24.5 AGENT-NESS |
| 24.0 SUBJ-PRED | 18.2 CHILDW | 23.5 CONSOI-NESS |
| SP-1 sp-2 sp-3 | 15.4 MALEW | 23.5 ADJ-NESS |
| 20.6 ADJ-MODC | 14.1 FORKW | 23.0 NOUN-NESS |
| AM-1 | 11.0 EATW | 23.0 CNOUN-NESS |
| 19.3 PREP-PHR | 5.3 MARYW | 21.0 SHOONENN |
| PP-1 pp-2 | 5.0 CHICAGOW | 20.6 INSTR-NESS |
| 17.7 INTENSIFYC | 5.0 PEACHW | 20.4 OTOKON1 |
| IN-1 in-2 | 2.5 IS2W | —cats,rels— |
| 7.6 LOC-SETTNGC | — — | 14.6 ADJECTIVE |
| LS-1 | — — | 13.6 NOUN |
| 6.4 TRANSITIVEC | — — | 11.5 VERB |
| TV-1 tv-2 tv-3 | — — | 10.5 CAUSER |

Figure 1.10: Activation Levels of Selected Nodes Just Before Output of *"big"*

After *"big"* is output **boyw** becomes the most highly activated word and *"boy"* is output. As

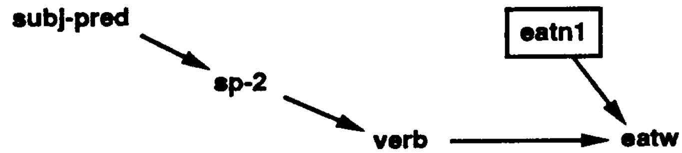


Figure 1.11: Selected Paths of Activation Flow Just Before Output of *"ate"*

seen in Figure 1.7 this word is not linked directly to any of the nodes present in the input; its selection represents a kind of inference. This and other kinds of inference are discussed in §3.4.

After *"boy"* is output, syntactic update moves the cursor of **subj-pred** its second constituent becomes highly activated, this constituent passes activation to **verb**, and **eatw** becomes highly activated, as shown in Figure 1.11. The word **eatw** is emitted as *"ate"*, due to activation from **pastn**, which is part of the input. (The inflection process is discussed in §6.4.1.)

After a verb, here *"eat"*, has been output, **transitivec**, representing the Transitive Construction, is updated so that its cursor points to its second constituent, which causes activation to reach the **patientr** relation. As a result, **peachnl**, the patient in this input, becomes highly activated. The way this notion is realized is determined by synergy with other constructions. To be specific, activation flows from the **possessor** relation, which relates **peachnl** and **marynl**, to **genitivec**, representing the Genitive Construction. This construction affects word order not by activating a syntactic category but rather by activating a relation and indirectly some concepts; its first constituent activates fillers of the **possessor** relation, in this case **marynl**. As a result **meerinl** (marynl) is the most highly activated node of the input, and **maryw** becomes the most highly activated word, as shown in Figure 1.12. After *"Mary"* is output, **genitivec**'s cursor advances and it highly activates **apostrophe-s**, and so *"'s"* is output next. Finally the word for the patient, namely *"peach"*, is emitted, there no longer being any words with more activation than it. (Synergy among constructions in general is discussed in §4.4.2.)

| —constructions— | | ——words—— | | —notions,insts— | |
|---|---|---|---|---|---|
| 26.3 | SUBJ-PRED | 21.3 | MARYW | 25.2 | NOUN-NESS |
| | sp-1 sp-2 SP-3 | 21.1 | A-W | 24.8 | CONSOI-NESS |
| 24.1 | PREP-PHR | 20.6 | PEACHW | 24.8 | CNOUN-NESS |
| | PP-1 pp-2 | 19.3 | THE-W | 24.3 | POSSE-NESS |
| 22.3 | DETERMINATC | 16.2 | CHICAGOW | 23.7 | THING-NESS |
| | DT-1 dt-2 | 16.2 | FORKW | 23.7 | PAT-NESS |
| 22.1 | ADJ-MODC | 12.6 | WITHW | 20.0 | MEERIN1 |
| | AM-1 | 11.9 | IN-W | 19.5 | LOC-NESS |
| 20.3 | LOC-SETTNGC | 10.2 | IS2W | 19.5 | INSTR-NESS |
| | LS-1 | —— | | —cats,rels— | |
| 17.2 | GENITIVEC | —— | | 16.8 | ARTICLE |
| | GE-1 ge-2 | —— | | 16.6 | NOUN |
| 11.5 | TRANSITIVEC | —— | | 16.0 | LOC-ADVERB |
| | tv-1 TV-2 tv-3 | —— | | 16.0 | ADJECTIVE |

Figure 1.12: Activation Levels of Selected Nodes Just Before Output of *"big"*

At this point the concepts which remain to be expressed are **forknl** and **chicagonl**. **prep-phr**, representing the Prepositional Phrase Construction, now receives activation from two main

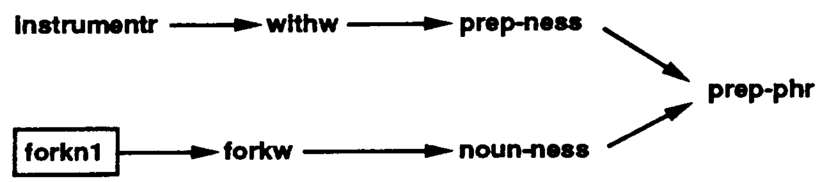


Figure 1.13: Selected Paths of Activation Flow Just Before Output of *"with"*

sources: first, from the noun **forkw** (FIG has this pathway for activation flow since it allows prepositions in front of nouns as a default); and second, from the preposition **withw**, which is activated since **forkn1** fills an **instrumentr** relation in the input (details of activation flow via relations are given in §6.2). These reasons are summarized in Figure 1.13. (That many nouns and prepositions are active does not cause **prep-phr** to become over-activated due to the special evidence-combining function used by 'collector nodes' such as **noun-ness** and **prep-ness**, as explained in §4.2.4.) **prep-phr** thus becomes highly activated and activates its first constituent, which in turn activates prepositions. Choice among prepositions is affected by activation from relations. Because the weights on the path from instruments to **withw** are higher than the weights from locations to **in-w**, *"with"* is output. The correct noun must go with the preposition just output, and this happens since, after *"with"* is output, the update process gives a minor boost to **forkn1**, since it is attached to the relation expressed by the word just output, namely **instrumentr** (this update process in described in §6.4.3). The Determination Construction, having reset after participating in the output of *"the big boy"*, causes the output of *"a"*. *"fork"* is then output.

At this point the only concept not yet expressed is **chicagon**, which fills a **locationr** relation. **in-w** being the word associated with **locationr**, FIG now produces *"in Chicago"* in the same way that it produced the previous prepositional phrase. All concepts of the input having been expressed, FIG ends, having produced *"the big boy ate Mary's peach with a fork in Chicago"*.

## 1.3 Overview of the Thesis

Describing a highly parallel system is not easy to do on paper. Ideally the reader should have a copy of FIG to experiment with while reading along. Instead FIG and the associated theory are presented in several passes.

Chapter 2 continues discussion of the issues raised in the first section of this chapter. It characterizes the nature of the generation task and presents principles for building a good generator. This chapter also criticizes common perceptions of the generation task and shows that serial approaches to generation are inadequate.

Chapter 3 discusses the most concrete part of the generation process: the choice of words. This chapter presents the basic issues, explains how FIG handles them, and surveys previous approaches.

Chapter 4 discusses how to represent and use syntactic knowledge in a network-based system and contrasts this with other approaches to syntax for generation. The main results are, contrary to received wisdom, that neither the assembly of syntactic structures nor explicit syntactic choice is necessary for generation.

Chapter 5 explains the grammars of English and Japanese incorporated in FIG.

Chapter 6 documents every last detail of FIG as implemented.

Chapter 7 discusses the role of connectionist models in the study of language processing.

Chapter 8 discusses psycholinguistic evidence on language production, showing that the design principles of FIG are compatible with what is known about human speakers.

Chapter 9 illustrates and reinforces the claims of Chapter 2 by exploring the problem of machine translation. This chapter is also a stand-alone presentation of a new model for translation, the 'F Model', and a critique of previous approaches to translation.

Chapter 10 evaluates FIG and summarizes the results.

# Chapter 2

# Design Issues



This chapter considers the nature of the generation task and proposes design principles for building successful generators. It also criticizes previous research for failing to adequately identify the task and for building inadequate models of generation. This chapter is idealistic: I point out the weaknesses of other systems even when my own system is no better.

Although this chapter mentions a great deal of previous research it is not a survey of the field. Readers seeking an overview are referred to Mcdonald (1987a) or Kempen (1989). This chapter discusses general architectural issues, for example the uses of parallelism. Specific issues in lexical and syntactic processing in generation are discussed in Chapters 3 and 4; which also exemplify the points made here.

## 2.1 Characteristics of the Generation Task

This section is an inquiry into what a generator must be able to do. The tenor of the argument is that generation is much more complex than is generally supposed. §2.2 criticizes previous research for ignoring this complexity, and §2.3 discusses how a generator can cope.

### 2.1.1 Information-Rich Inputs

genre: folk tale
previous sentence: *"The old woman went to the stream to wash clothes."*
topic of next sentence: describe woman's reaction to peach
role of this sentence in the narrative: introduce the peach into the story
important 'characters': peach and old woman, not the stream
old woman's condition: old, poor
viewpoint the story is being told from: old woman's
attitude towards old woman: sympathetic
attitude towards peach: amazing, later turns out to be enchanted

old woman's destination: stream
old woman's intention: wash clothes, in the stream
method of motion: walk
old woman's expectation: another normal day of chores
familiarity of stream to old woman: very

setting: fairly swift mountain stream, emptying into a pool
location of woman upon arrival: side of stream by pool
event enabling her to see peach: arrival at stream
time of seeing peach: immediately after arrival
result of seeing peach: old woman becomes surprised
specific surprising fact: enormous size of peach
prior probability of a peach being greater than 20 cm in diameter: 0

initial location of peach: in the stream, visible above water, upstream of the woman
direction of peach motion: downstream, coming closer to the woman
manner of peach motion: bobbing up and down and from side to side
cause of peach motion (floating, bobbing and moving): all due to current of stream

aspect of floating in a stream: continuous action
content of stream: water
speed of current in mountain streams: swift
depth of mountain streams: shallow
density of peaches: lighter than water
inanimate(peach)

Figure 2.1: A Sample Input

Most generators accept inputs which are relatively poor in information, such as feature structures, lists of propositions, logical forms, 'messages', 'realization specifications', and so on (Mann 1983; McDonald 1983a). Yet ideally the input to a generator should be 'rich' in information. Figure 2.1 illustrates the large amount of information which can (and should) be the input causing the generation of a single sentence.

Appropriate outputs for this might include

2.1 *When the old woman got to the stream, she was surprised to see a big peach bobbing down towards her.*

2.2 *She got to the stream, and, upstream, there was an enormous peach! bobbing down towards her.* (better if read aloud with dramatic pauses)

2.3 *She got to the stream, and upstream, to her great surprise, there was an enormous peach bobbing along.*

Producing these sentences, rather than something similar but wrong, can depend on nuances of meaning, as argued in §1.1.1. To give just one example here, *"a peach surprised her"* is inappropriate given that the woman *saw* the peach, rather than hearing or bumping into it. One might object that the task of a generator is merely to 'express the facts', and that it is not important for it to accurately convey nuances of meaning. But in fact these nuances can be helpful to a reader. For example, in the text

2.4 *There are 10,635,264 words available if you elect not to garbage collect.*

*"if you elect not to"* instead of *"if you do not elect to"* or *"unless you"* reflects the fact that to garbage collect is the normal choice. *"Available"* instead of nothing suggests that *"words"* are a resource for the user.

Thus a generator needs access to a lot of information in order to make correct word choices. It may not always use it all — depending on the target language — but the information must be there just in case.

It is true that there is currently no application which requires a generator to deal with such rich inputs. Yet this will change as the programs which interface with generators become more knowledge-intensive. For example, imagine an advanced expert system with a complete causal model of its domain. For a generator to express the rationale behind some conclusion, it will probably need access to the entire working memory and knowledge base of the system. As another example, consider an advanced parser/analyzer, able to infer all the implicit information in a text, including the background knowledge of the writer and the nuances the reader should pick up. Again, to ensure a faithful expression of the result into, say, French, a generator should have access to of all that information.

This discussion raises a deeper issue: that of the relation between the generator and the system within which it works. Most generators, at least as implemented, are modules which operate in isolation; they are passed a few symbols as parameters and have no other access to the knowledge of the system. It is, however, more appropriate to view a generator as a process which shares memory and knowledge with other processes. Indeed, the very word 'input' is inappropriate to the extent that it suggests that a generator can operate autonomously. Nonetheless, I will continue to speak of the generator's 'input' for the sake of convenience.

If a generator is to accept information-rich inputs, a potential problem arises: not every fact need appear explicitly in the output, because of the hearer's inference ability. For example,

2.5 *she saw a peach floating in the stream,* **being moved by the current, and moving downstream**

is redundant in that the information given by the words in bold is inferrable from the first clause. This is a serious issue, since existing generators are based on the idea that in the normal case the task is to map from input elements to output words one-for-one. They rely on a distinction between 'things to be expressed in the output' and 'the things which only bias the way those things get expressed'. But there can be no such in-principle distinction, since which information gets explicitly expressed can depend on the specific words available in the target language.

Even if such a distinction were possible, it would be necessary to retrofit existing generators with pre-processors able to take information-rich inputs and reduce them to simple 'messages' containing exactly and only the things to be expressed in the output. No one has proposed how this might be done. There are systems which decide 'what-to-say', but most only address the problem of which propositions to include. Even those which address the decision at a finer grain (Hasida *et al.* 1988) do not employ knowledge of the target language, which is clearly a mistake, since the decision of which concepts to express depends on the specific words and structures available in the target language, as seen in Examples 2.1–2.3.

Note that this, the problem of having more information available than should be explicitly expressed, can arise even for information-poor inputs, at least for generators which have, as they should, the power to augment their 'inputs' by inference at run-time. This inference ability can be needed in order to 'realize' that a certain word is applicable to describe a situation, as Hovy has pointed out (Hovy 1988). For example, from 'the peach was upstream of the woman and moving downstream' a generator should be able to infer that 'the peach was approaching the woman', and thus that words like *"approach"* and *"come"* may be appropriate. Such a generator has a 'virtual' rich input, and thus also faces the problem of choosing what information should be explicitly expressed. An approach to these problems is presented in §2.3.5.

### 2.1.2 Multiple Interacting Goals

A generator must strive to simultaneously achieve many goals, and these goals may be in conflict. For example, the goals of being clear and unambiguous inherently conflict with the goal of being concise. There are also potential conflicts among various pragmatic goals, such as the potential conflict between the goal of not offending the hearer and the goal of conveying all the intended information.

Perhaps the two most basic goals of generation are 'express the input' and 'express it in the target language'. Most descriptions of the generation task conflate these, but considering them as distinct goals makes it easier to see that a speaker must sometimes make trade-offs between even them. In other words, a generator must sometimes choose between being faithful to the original intention and sounding natural and fluent. Talmy (1976) has a nice example of this. Imagine the speaker has the goal of praising someone for being charming and observant, and that saliencing goals lead to choice of a pseudo-cleft construction, resulting in

2.6 *what I like about her is her charm and her ...*

From this point the speaker cannot continue grammatically, because a nominal is required but *"observant"* does not have a noun form. To continue requires sacrificing at least one goal: sacrificing grammaticality can lead to *"... charm and observantness"*; sacrificing fidelity to the original intention can lead to *"... charm and perceptiveness"*; and sacrificing the goal of not being too informal can lead to non-parallel conjuncts, as in *"... charm and how she's observant"*.

This is an example where the speaker's goals are not inherently in conflict, but conflict arises dues to dependencies among various possible choices of words and constructions. In general, goals can interact indirectly, in that the resources for achieving them (the words and structures of the target language) may or may not be compatible.

The notions of goal and goal conflict can also be fruitfully applied at a finer grain. One can consider each 'part' of the input as giving rise to the goal of expressing it. A generator must satisfy these goals while still achieving the orthogonal goal of producing a valid utterance by respecting the dependencies among syntactic and lexical choices. For example, the lexical causative construction can only be used if the verb allows it, as shown by the ungrammaticality of *"the magician disappeared the rabbit"*. Given the importance of the goal of arriving at a compatible set of choices, it may be helpful to think of generation as a constraint satisfaction problem.

Yet most models of generation have little place for interaction. They are based on the assumption that, in the default case, each of the various decisions involved in the process of generating a sentence can be made in isolation. Such generators map each individual portion of the input onto a word or construction, ignoring questions of compatibility among the choices. Generators which do this can be said to be 'compositional'. Context effects are treated as special cases in such systems.

An improvement is to allow a little interaction by having only certain goals interact, for example those of the same type. For instance, a generator might first work on pragmatic decisions, then on syntactic ones, and so on. This strategy leads, of course, to modular generators, which apply different kinds of knowledge at different times.

To allow a little more interaction, a generator can use an algorithm which consults different kinds of knowledge in an interleaved order. For example, a syntax-first algorithm may first choose a verb and then, depending on its valence, decide whether or not to use the lexical causative construction. This is not a general solution; as §2.4 points out, there is probably no fixed order of choices that does not violate any dependencies.

Other types of interaction, if allowed at all, are commonly treated as rare events. For example, one component is sometimes allowed to send a message to ask a query of another component.

Returning to the topic of modularity, it is interesting to consider as a case study one type of modularity and the ways in which it prevents needed interactions. One popular way of modularizing the generation process is to separate very language-y tasks from less language-y ones. Proposals in this vein include separating what-to-say from how-to-say-it, strategy from tactics, planning from realization, and concept choice from linguistic choice. The information passed from one module to the other is variously known as the 'message', 'meaning', 'content', or 'realization specification'.

As many researchers have pointed out, however, no such 'message' can contain enough information (Danlos 1984; Appelt 1985; Hovy 1988; Meteer 1989), since even seemingly low-level language decisions can depend on seemingly irrelevant information, or on the speaker's goals. For example, the language at hand might require a choice between *"a"* or *"the"* or between *"in"* or *"on"*, and such decisions might depend on any nuance of the input.

Conversely, if language-y tasks are separated from other tasks, the process responsible for choosing conceptual content has no access to information about the resources available in the language at hand, and unnatural utterances can result (Danlos 1987). The next few paragraphs give several examples of the way that the dependencies of the target language constrain the 'content' of what is conveyed — that is, examples of interaction between considerations of language and of what-to-say.

Syntax and the lexicon can 'constrain' what is usually called concept choice. For example, they constrain what can be expressed easily and compactly. For instance, it requires more effort to specify the sex of a person in Japanese than in English, where that information can be encoded concisely on a pronoun. On the other hand the language can provide opportunities to express particular types of information easily. For example, English has post-verbal particles which are convenient for expressing directional information, as in *"come straight back down from up there"*. Such information could not be encoded nearly as simply in many languages, nor could it be expressed so as to make it clearly only background information.

Language also constrains what information can appear together. To repeat an example from §1.1.1, *"went to"* can be followed by verbs describing actions, for example *"get"*, but not states, for example *"have"*. In such cases a generator may find that its first choice for a lexical item to express one concept may be syntactically incompatible with its first choice for a lexical item to express another concept. Conversely such interactions also come in the form of opportunities. For example, in response to *"where is Henry?"* the reply *"I saw the bastard in Florence"* (Levelt 1989) allows the speaker to concisely give the requested information and express his feelings about Henry.

Because English allows deprecatory referring expressions there is interaction between these two goals. Appelt (1985) also gives examples where a single noun phrase can satisfy both referential and attributive goals.

Any one of the types of dependencies among choices is easy to handle, but handling all of them together is difficult — an architecture issue. This issue can only become more important in the future, since no existing generator is complete. There are many kinds of considerations which will need to be added to any generator, and they all interact with other considerations. For example a generator should avoid accidental alliteration and duplication, as in *"I like most animals, but I can't bear bears"*; this of course interacts with other considerations in lexical choice. Another thing the ideal generator will deal with is prosody, including intonation contours; this also interacts with lexical choice, since the stress patterns and lengths of words must allow the desired overall intonation, and with clause structure, clause length and sentence type, as in the choice between a declarative statement and a rhetorical question (Cutler & Isard 1980). Another consideration is to avoid choosing words which evoke unpleasant memories in the hearer; this pragmatic consideration obviously interacts with lexical choice.

### 2.1.3 Incrementality

An incremental generator is one that performs most of the computation relevant to the output of a word not long before that word is output. It is hard to come up with a more precise definition, but easy to identify certain generation strategies as antithetical to incrementality. One such strategy is that of planning the whole text at once, possibly starting with S as a top node for a sentence. Indeed any approach that does pre-planning or relies heavily on 'lookahead' is not incremental. The use of non-incremental processing is often compounded, as in multi-pass generators, which have to complete one pass before starting the next.

Recently several researchers have argued for the value of incremental generation or have proposed incremental models (Kempen & Hoenkamp 1987; Finkler & Neumann 1989; De Smedt 1990; Kitano 1990). While for many tasks incremental generation is not needed, there are several cases where it is useful:

Sometimes a generator must begin outputting words before the input is complete. This is clearly the case when the information to express arrives in a stream from an external source, as when doing simultaneous interpretation or when narrating sports events.

Incremental generation is also useful when the intention of the speaker may change during the course of generation. Such changes may be due to remindings which occur during generation (Hovy 1988) or to signals from the listener, among other things. An incremental generator is likely not to have done a lot of pre-planning, so it is likely to be able to proceed without complicated re-planning if the intention changes.

Even if the entire input is available before generation starts and it is guaranteed not to change, incremental generation may still be useful for the sake of decreasing response time. That is, it may be a good strategy to output the first words before all the decisions about the form of the entire utterance have been made. That human speakers exploit this strategy is suggested by pauses and false starts (§1.1.3, Chapter 8).

Of course, a generator must not be completely incremental. For example, a generator cannot be "on-line" in the sense that it "converts the current token in its input stream completely into tokens of the output stream before moving on to the next input token" (McDonald 1983a). Such a generator would have no chance to consider interactions between choices; it would be completely compositional.

Parenthetically, incremental generation is possible because language is well suited to speakers with only limited pre-planning ability. Language is not such that the speaker must do arbitrary lookahead and computations, so a functional, incremental approach suffices. Cases such as Example 2.6, where the choice of a grammatical form can become inappropriate depending on a word choice much later in the sentence, are fairly rare. In other words, there is not an unreasonable amount of long-distance interaction.

## 2.2 Why Previous Research has Missed the Point

If the issues raised in the previous section are as important as I say, why has so little work addressed them? I believe the reasons lie in the way the field has developed and the way the generation task is still perceived; this section explains.

### 2.2.1 Inputs Poor in Information

Impoverished inputs are assumed without question by most generation researchers. The reasons for this lie in the historical development of generation research.

Early generators were showcases of syntactic theories. Indeed, the very word 'generate' suggests a mathematical operation to produce all and only the sentences of a language from a finite description, attesting to the origins of the field. To demonstrate the coverage of a generation mechanism it is necessary to make it exhibit grammatical variation. One way to do this is to have it randomly choose among options (Friedman 1969). A better way is to determine what governs the variation, and make that information the input to the generator. Such inputs thus serve to specify the behavior of a generator. Typically the 'specification' contains exactly enough information to specify one sentence of a specific language; this is most obviously true for the inputs to systemic generators (Davey 1978; Mann 1983).

Another important early influence on the field of generation was the idea of a 'deep structure' underlying and prerequisite to the surface realization. It was only a short step to then call this structure (or something slightly deeper, such as a 'logical form') the generator's input. If one adopts an input-driven strategy, these inputs straightforwardly control the behavior of a generator. For example, traversal of a downward link might cause a subroutine call, and arriving at a node for a verb-concept might lead to unifying slots with cases. In general it is common for the structure of the input to directly determine the sequence and constituency of the text, as McDonald has observed (McDonald *et al.* 1987). Deep-structure-like inputs have the advantage that they are often plausibly useful as representations of meaning for other reasoning and symbolic manipulation tasks.

Finally, for practical generation systems, the input only needs to contain just enough information to select the right canned text and perhaps to control variation (singular vs plural) and govern choices (Tuesday vs Wednesday).

The underlying reason behind the common acceptance of information-poor inputs seems to lie in the common practice of 'reverse-engineering' sentences to find out what goes into them, and then making that be the input to the generation process. There is nothing wrong with this in principle, but the process is never carried far enough; it typically stops far short of the 'information-rich inputs' which presumably are the true starting points of generation. Another problem with this practice is that inputs based on slightly-reverse-engineered sentences of one language are clearly not useful for generating into other languages. (This points up one advantage of studying generation in the context of machine translation: it makes it easier to avoid the temptation to start from inputs which are thinly disguised versions of the desired outputs.)

The alternative to reverse-engineering is to choose an independently motivated input format and then work from that. Building a generator to work with such inputs may result in a useful piece of software but may not say much about generation as a general problem. Descriptions of such generators tend to degenerate into discussions of how they deal with specific problems raised by the specific representation languages they use, such as NIKL's handling of definitions (Sondheimer *et al.* 1990), or Conceptual Dependency's decomposition of verb meanings into primitives (Goldman 1974). Most generators used in machine translation are degenerate in this way — they tend to focus narrowly on problems which arise when trying to express an input which comes from a specific parser for a specific source language (Nagao *et al.* 1984).

This is not intended as an argument against the general goal of finding *the* correct way to represent meaning. I merely observe that it will be at least a long time before the general representation issue is definitively solved, and generation research should not have to wait until then.

Generators have been built which explain expert systems' conclusions (Miller & Rennels 1988; Moore & Swartout 1989), answer users' questions (Wilensky *et al.* 1989), and so on. Yet the existence of such applications does not prove anything about the power of the generators. The reason that there are generators with acceptable performance is that such generators make assumptions about their inputs. In other words, they have implicit, built-in knowledge about the domain where they will be used. For example, McDonald (1987a), discussing his own previous work (McDonald 1983b), points out that his generator can produce impressive output, but does so only because it presumes certain information which is not explicit in the input. Meteer (1990) shows how many other generators do the same. There is nothing wrong with this if the generator is intended to be used for one specific application, but it is clearly not the way to build general-purpose systems.

In short, to the extent that generators generate correct utterances without the benefit of information-rich inputs, it is because they cheat.

The most common place where implicit information is hidden is in the inventory of lexical and syntactic constructions. In most generators these are severely limited. Consider the sentence:

2.7 *The Vinson, which is an aircraft-carrier, has maximum-speed 29 and current-location Sasebo.*

This is easy to produce if a generator *lacks* a lot of knowledge. But if a generator has the knowledge needed to produce also *"a slow ship"*, *"a ship whose maximum distance-covered per time-elapsed ratio is 29"*, *"a ship which can reach 29 knots"*, and so on, its input must contain enough information to allow it to choose sensibly. Having more knowledge also can lead to trouble in that a generator might mangle fragments of all of these together; in other words, a generator whose knowledge includes enough grammatical and lexical variation for there to be unanticipated interactions cannot generate compositionally.

In general, generators which incorporate implicit assumptions about the input can produce good text with only information-poor inputs, but they allow very little flexibility. That is, they can exhibit only very limited coverage of the language in question, as discussed in §2.1.1. This is appropriate in that the expressive needs of most programs are quite limited — a few basic sentence patterns with the ability to substitute-in open-class words can suffice. (Generators are generally fairly complex systems, but it is seldom clear whether the complexity discussed is necessary, or whether the outputs could just as well have been produced by a much simpler technique.) If, however, a generator is to be used for more than one domain, it should use general, linguistically motivated knowledge, rather than templates or canned phrases. This enables it to be applied to new domains without the need to build a dictionary from scratch, as argued in (Jacobs 1985a).

Many researchers dream of a stronger type of reusability: the idea of a generator that you can just plug into to any system, and have it spew out all the language you need. There is no a priori reason to think that it is possible to build a generator portable in this way — in human speakers generation densely interacts with other knowledge and processes, such as spatial reasoning, and emotion, to name just two.

There is also the practical problem that making a standard generator portable is very difficult. That is, if a generator is directly controlled by its input, as so many are, that it must receive an input in exactly the expected format. §4.5.5 gives examples of unreasonable constraints that many generators impose on the programs that use them.

One can of course declare as 'the standard meaning representation language' whatever format the generator happens to expect for its input. A less imperialistic solution is to proclaim the generator's input format to be an interface specification, and expect people wanting to use the generator to write conversion routines to convert their representations to it.

It seems to me that such positions are unreasonable in requiring too much of programs using generators. It should not be necessary to rigidly prescribe the format of the generators input; a generator should be robust in the face of variations in the structure or conceptual vocabulary of the input. In slogan form, a generator should be flexible.

I intended to build a generator flexible enough to work directly from any reasonable input, including the raw output of a Japanese parser/understander, not tidied up to make it convenient for an English generator (§9.5.2). However the idea of flexibility as a special property of a generator now seems unnecessary. If a generator accepts information-rich inputs, as it must, the possibility of simply mapping input to output is clearly absurd, so any good generator must necessarily be a 'flexible generator'. Further discussion and examples of flexibility are given in §3.4 and §6.2.

### 2.2.2 Generation as Mapping

What is the relation between the generator's input and its output? Most theories of language have a definite answer to this question, and most generators embody such a theory. To oversimplify, most 'map' the input into the output. This view implies that a generator "starts (as people usually seem to) with a complete, well-formed 'thought' to be 'said' " (Cullingford 1986). There are several problems with this view of generation.

First, there are multiple goals; expressing a 'meaning' or 'thought' is only part of the generation task, as discussed in §2.1.2. Second, even if we restrict attention to the expression of meaning, the generation process has no 'input' separate from the background knowledge and world knowledge, as discussed in §2.1.1. In other words, the impetus and initial state for generation is rich in information.

Therefore it is a mistake to view generation as starting from an input which corresponds in any simple way to the 'meaning' of the utterance finally output. For example the mapping from input concepts to output words need not be one-to-one; indeed, the way the input meaning is made up of concepts may not be reflected at all in the way that the meaning of the output arises from the words. It is well-known that the task of a parser/analyzer is not merely to 'extract' some meaning already present in the utterance, but rather to take the utterance as a sequence of clues which provoke and guide the formation of an interpretation. Conversely, in generated text "the content explicitly expressed ... constitutes a dotting of disconnected islets of information. The fully continuous 'sea' exists in the minds of the intercommunicators as a skein of knowledge and familiarities, presuppositions and expectations, presumptions and deductions. ... the manifest communication ... has interpretable significance only within the context" of the sea of information (Talmy 1976). In slogan form, speech creates meaning.

In general, the 'meaning' of the final utterance can depend on the language which the speaker happens to be producing at the time. I do not mean that there exist some inputs which are impossible to express in language X, but only that there is a continuum between things said because the speaker wants to say them and things said because the language facilitates or forces their inclusion, as illustrated in §2.1.2. Cases where the speaker or author insists on expressing his exact meaning, riding roughshod over the language, are rare enough to be noteworthy. Heidegger's philosophical writings are the only example that comes to mind; these are notoriously unreadable.

The only advantage of viewing generation as a process of mapping is that it allows formalization of the problem of generation. One can, to this end, define 'generation' to be the task of mapping 'logical forms' or other information-poor inputs to text, but this comes at the price of leaving out most of the interesting and difficult parts of generation. Fortunately, a formal definition of the generation task is not needed to produce useful text.

### 2.2.3 Modularity

In general modularity is a valid design objective. If each module has only limited interaction with other modules, they can be built one by one, and the system as a whole is easier to extend and understand. Most generators are very modular (McDonald 1988): there are, for example, commonly divisions between thinking and generating, between deciding what-to-say and deciding how-to-say-it, and between syntactic choice and word choice.

But in generation the convenience of modularity comes at the price of restricting interaction, and thereby risking poor quality output. Various types of modularity and their associated problems are discussed in §2.4.

Most researchers fail to recognize the need for interaction, probably because each typically addresses only one dimension of variation. For example, (to simplify) Goldman's BABEL (Goldman 1974) really only had to choose among words with some common element of meaning; McDonald's MUMBLE (McDonald 1983b) really only had to choose among alternative parts of speech for expressing a node; and Mann's PENMAN (Mann 1983) really only had to order words and choose among syntactic options. When studying only one issue, the problem of dependencies among diverse kinds of choices obviously does not arise.

A historical anecdote can illustrate the danger of researching generation one module at a time. The original strategy/tactics distinction was proposed by Thompson to justify addressing certain syntactic aspects of generation in isolation; these he called the "tactical aspects" (Thompson 1976). By "strategic aspects" of generation he meant everything else, including word choice and the interface of thought to language. Later McKeown addressed the problem of 'strategic generation' as that of determining "the content and structure of the discourse" (McKeown 1985). She then worked on some issues in text grammar, touching on only a fraction of the problems set aside by Thompson. Yet the terminology suggests, and many researchers believe, that the task of generation has been exhaustively decomposed (McDonald 1987a), and that the basic theoretical work in both halves is finished.

While it is reasonable to partition the *problem* in order to study it, one must take care not to define away important problems, such as interaction between syntactic and other considerations. It is fair to focus one's research on a single issue, but one can not simply assume that that issue should be handled by an autonomous module. That is, divisions of the task which are invented for the sake of organizing research should not be imposed on designs for generation.

### 2.2.4 Other Problems

Like Belgium, the field of generation is subject to frequent invasion from neighboring fields with strong ideologies, such as the fields of knowledge representation, linguistics, planning, search, expert systems, and software engineering. Thus, many of the ideas in the field are motivated not by the generation task itself, but rather by extraneous theoretical considerations. This was especially true in the early years of generation research (pre-1980), when two major tendencies ruled the field. The first was experimentation with various grammatical formalisms, including transformational grammar, ATNs (Simmons & Slocum 1972), and systemic grammar (Davey 1978). The second major tendency was experimentation with proposals for meaning representation. Goldman (1974), concerned with producing English to express Conceptual Dependency diagrams, is a prime example. These two tendencies are still very strong in the field, as discussed in §2.2.1.

Despite the limited success of such 'theory-driven' approaches, there was no general reexamination of the basic assumptions and techniques of the field. Rather, the consensus was that the shortcomings were due to the lack of knowledge of various kinds (Mann 1982). Attention accordingly turned to building bigger systems and exploring new topics, such as:

- specific issues in syntax, such as tense, reference, and connectives.
- new types of knowledge about language, notably knowledge about discourse and text structures (Mann & Thompson 1987; McKeown 1985).
- new tasks, for example planning referring expressions that the hearer can understand (Appelt 1985); producing language to satisfy pragmatic goals, such as convincing the hearer of something (Hovy 1988); coordinating generation with other ways of communicating information, such as pointing (Reithinger 1987; Feiner & McKeown 1990); and tailoring explanations to the hearer's knowledge state (Chin 1988; Moore & Swartout 1989).
- better representations for knowledge (Jacobs 1985a).
- new formalisms, such as unification (Kay 1984), tree adjoining grammar (Joshi 1987), and Patten's (1987) formalization of systemic grammar.

Thus, recent research has largely addressed 'advanced topics' rather than the fundamentals of generation: expressing the meaning at hand and doing so in a natural way.

## 2.3 Design Principles

This section argues that the ideal generator should have certain characteristics, namely: be integrated, represent the current state explicitly, be parallel, combine evidence numerically, rely on emergent choice, and use feedback. The concrete significance of these design principles is amply illustrated in Chapters 3 and 4. Evidence that the human language production mechanism exhibits these properties is presented in Chapter 8.

FIG is mentioned in this section merely to illustrate how a generator might embody these principles. There is no claim that FIG is in any sense ideal; it merely demonstrates that these design principles are workable.

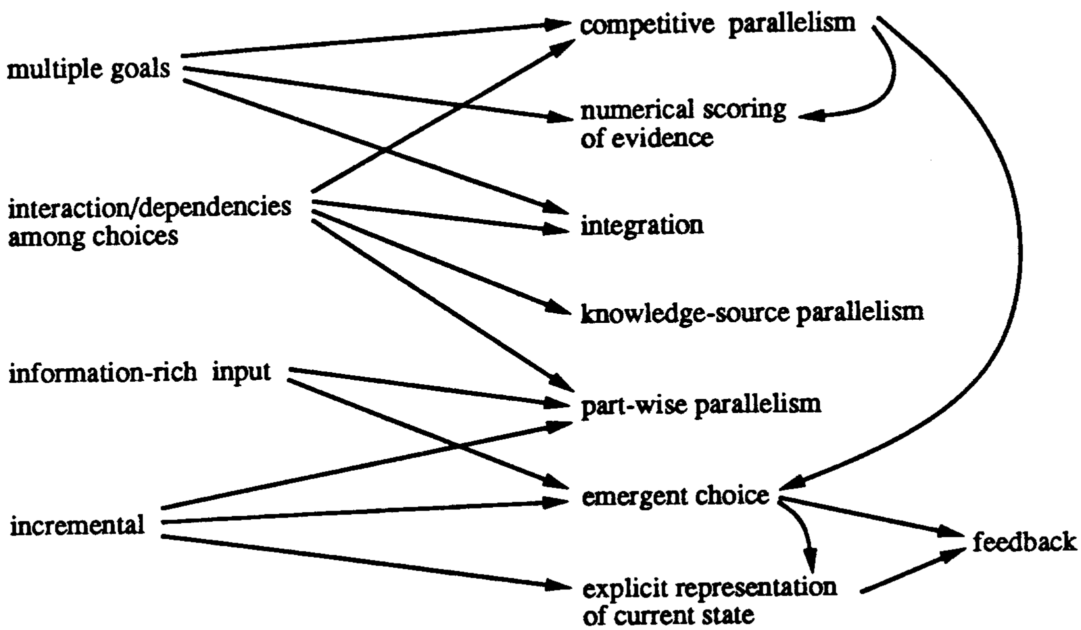


Figure 2.2: Generation Issues and Design Principles

Figure 2.2 summarizes how this section relates to §2.1. The items on the left represent characteristics of the generation task; those on the right represent design principles. The arrows mean 'requires' or 'suggests the use of' or 'becomes easier with increasing amounts of'. A cursory examination of the properties listed at the right of the figure suggests why I built FIG as a spreading activation model. This point is discussed further in Chapter 7. §10.1 discusses the extent to which FIG-as-implemented solves the problems raised in §2.1.

Some of these 'characteristics of the ideal generator' can be applied to existing designs as cost-effective ways to improve their performance, although this section does not focus on this. Some of the design principles presented here have precursors, but there is no attempt to trace their history; in any case, this is the first work to discuss all the principles, motivate them, relate them to each other, and contrast them to the principles implicitly underlying more traditional designs (§2.4).

### 2.3.1 Integration

In order to deal with the dependencies among lexical and syntactic choices, and thus interactions among diverse goals, a generator should be integrated. That is, there must be pathways for information flow that allow one consideration to affect all choices, of whatever type, that depend on it. This contrasts with modular approaches. In FIG everything is integrated — there are no modules at all. Information, in the form of activation, flows among nodes of all types. In particular all knowledge sources are active in parallel, all knowledge is encoded in the same network, and information on the current state is also represented in the network, with activation levels. It appears that no other generator is completely integrated, as discussed in §2.4.

### 2.3.2 Explicit Representation of Current State

To generate incrementally requires that a generator have a representation of the current state of the generation process at each moment. This is necessary so that all relevant information is accessible and available for each successive word choice. FIG easily meets this condition; it has a complete and explicit representation of the state, syntactic and semantic, at each moment of the generation process. This representation consists of the activation levels of many concepts, syntactic constructions, and words; and this information is globally available by means of flow of activation.

The representation should be easy to modify, both to reflect the progress of generation and to enable changes of intention, as might occur due to remindings in the course of generation. Modification of FIG's representation merely involves changing its activation levels; there are no structures to change or manipulate. For example, if some reminding occurs, causing a word to be judged taboo in the current context, its activation level may be set to zero, and the syntactic situation will be re-evaluated automatically as activation flows. This makes FIG robust in the face of changing intentions, provided only that the network is given time to settle.

Other generators generally do not maintain representations of the current state; those that do maintain representations in the form of structures, which are complicated to update, as discussed in §2.4.

### 2.3.3 Parallelism

Parallelism is useful in generation for many reasons. This subsection describes five types of parallelism and explains why each is useful.

**'Knowledge-source parallelism'** is simultaneous computation involving different types of knowledge, for example, simultaneous computation of what-to-say and how-to-say-it, or simultaneous consideration of lexical and syntactic knowledge. Knowledge-source parallelism enables interaction between diverse considerations. It is also cognitively plausible (the motivation for 'cascade models' (Stemberger 1985a)), and can improve the speed of generation (the key insight behind pipelined generators (Meteer 1990)).

Knowledge-source parallelism contrasts with stage models of generation, where different types of knowledge are applied independently at different times.

**'Competitive parallelism'** means consideration of several alternatives in parallel. Competition among alternative words is the simplest example of this. Competitive parallelism is useful in case the 'first choice' for the realization of something turns out inappropriate in the larger context; it also allows emergent choice, as discussed in §2.3.5. Competitive parallelism contrasts with considering only one candidate output word at a time, and backtracking to consider another alternative only if necessary.

**'Part-wise parallelism'** means working on several parts of the input in parallel, and thus simultaneously working on several parts of the output in parallel. A simple example is the simultaneous consideration of 'subsequent' words, for example *"big"* and *"peach"*. Part-wise parallelism contrasts with serial traversal of the input, serial lexical access, and so on.

Part-wise parallelism is a good way to improve the speed of response when used for incremental generation. That is, if it is not initially clear which information should be said first, the generator can work on several portions of the utterance in parallel, enabling it to quickly emit the portion which turns out to be an appropriate beginning. Researchers exploiting part-wise parallelism for this include (De Smedt 1990), whose generator effectively builds up several sub-trees in parallel, and (Finkler & Neumann 1989), whose generator is an object-oriented system which exhibits both knowledge-source and part-wise parallelism.

Part-wise parallelism is the natural way to generate from information-rich inputs, since the components of the input can all be goals active in parallel, each suggesting words and constructions appropriate for it. FIG can handle information-rich inputs without extra mechanism — its operation is no different no matter how many nodes are activated in the initial state.

Part-wise parallelism is also useful for handling dependencies among choices. To determine how to express one part of the input a generator must consider the way the surrounding utterance will turn out. If the various parts are generated in parallel, then knowledge about the probable output for one part is available for consideration when building another part. FIG exploits this information by having links between compatible choices. As a result, the network tends to settle into a pattern of activation in which nodes representing compatible choices are highly activated together, causing a consistent set of 'choices' to emerge. There is no need to explicitly reason about interactions.

Part-wise parallelism does not preclude incremental generation. Even if there is simultaneous processing for the whole utterance, processing can focus on just a few 'current' concepts and words. That is, there can be 'incomplete' processing for some parts of the utterance, perhaps resulting in useful probabilistic information such as 'current evidence suggests the head noun will be *"winds"*', but indelible decisions need be made only for the word about to be output. These points will be illustrated in §3.5.

For completeness I here mention two other kinds of parallelism, although their significance will only become clear in later chapters.

**'Evaluative parallelism'** is simultaneous consideration of all kinds of evidence for the appropriateness or relevance of a word, concept, or construction. Evaluative parallelism is simply the easiest way to compute this in the absence of any reason to temporally separate the considerations or to treat them differently (§3.1, §3.5, and §4.1).

**'Synergistic parallelism'** occurs when several constructions are active in the generation of one part of the output. This is discussed in §4.4.2.

FIG is computation-intensive; information, in the form of activation, flows in all directions through the network. Most paths of activation turn out to have no effect on the utterance produced. In contrast, most generators only compute what is needed. For example, one common technique is to use a checklist or discrimination net when choosing a word, and only test for the things mentioned in the checklist. FIG's 'useless computation' is an important feature: given information-rich inputs and the dependencies among choices, a great deal of computation is the price of high quality output.

### 2.3.4 Numeric Combination of Evidence

Since generation involves many goals it is necessary to make them commensurate in some way, so that the generator can effectively perform trade-offs. The easiest way to do this is to use some numerical 'scoring' mechanism. In FIG, the importance of every goal, even such humble ones as that of using a certain construction or a certain word, are represented by the activation levels of nodes. This allows all knowledge sources to communicate in the same language, namely via numbers.

It appears that no other generator, including structured connectionist ones, relies on numeric combination of evidence. Instead goals are either present or not; considerations are either active completely or not at all.

### 2.3.5 Emergent Choice

A generator can do without explicit choice except for words. That is, choice of content, concepts, syntactic structures, and word order can be emergent. Doing without explicit choice is better for the sake of parsimony and simplicity in general. A specific advantage is that it is easy to deal with information-rich inputs if there are no explicit decisions as to what content to include in the output. In other words, concept choice can also be emergent; since each word choice is a *de facto* concept choice. In FIG it doesn't matter how many concepts are initially active, nor how many words they suggest — in the end an appropriate amount of information will appear explicitly, without the need for a what-to-say preprocessor (provided that there is feedback, as discussed in the next subsection).

Trade-offs among goals can also be handled without explicitly reasoning about them. In FIG all considerations are 'scored' as to importance, and the more important have greater effects on word choice. Competition is also implicit — each candidate is scored independently, and the final decision is to take the best. There is thus no need to explicitly decide between alternatives, as discussed in Chapter 3 and §4.4.1.

Explicit decisions mean, in general, that information flow is tied up with control flow (§2.4). If information flow is separated from control flow, however, information can propagate freely, in all directions, in parallel. This also means that decisions can be made 'smoothly' over time, rather than by serial backtrack. Jacobs makes a similar, although weaker, point: he argues that 'selection' itself is only a small part of the generation task, and that much of the work should be done in a 'predisposition' phase in which certain choices are primed (Jacobs 1988).

### 2.3.6 Feedback

Feedback is necessary if many goals are simultaneously active at run-time and choice is emergent. FIG, for example, simultaneously 'tries' to express all the input, to be concise, and to be grammatical; and each construction can be seen as embodying a sub-goal to the goal of grammaticality. If choice is emergent there is no way to know in advance when any specific goal will become satisfied; goal satisfaction depends on which items happen to be available in the lexicon for the language being generated. Thus there must be processes to check and update the current status (satisfied or not) of each goal. (Feedback in this sense discussed here should not be confused with the 'feedback' among the levels of standard generators.)

One type of feedback needed is feedback on the content expressed, for the sake of avoiding redundancy. Thus a generator needs to monitor what is inferrable by the hearer. This requires feedback because, in general, a generator cannot know exactly what it will express until it has chosen the words. In an incremental generator, knowledge of what is inferrable from the words already output must be available to the component responsible for deciding whether to continue. More generally, this information serves to allow the system to keep a correct and explicit representation of what fraction of the input has been conveyed and what remains to be said.

Feedback is also needed to ensure a correct representation of the current syntactic state, at least for incremental generators which allow emergent syntactic choice. There must be a mechanism to give constructions feedback on what words have been output.

The need for feedback is, if recognized at all, typically only acknowledged for 'large' goals, such as detecting ambiguity of the generated output, or modeling the hearer's probable reaction (Clippinger 1977); the process of providing such feedback is usually called 'monitoring' and often involves user models.

In particular, most generators do without inference to determine what has been expressed. If the input is information-poor, a generator can implicitly preserve the amount of information. By using a one-to-one mapping from concepts to words, with no need or opportunity for creative word choice, the output is guaranteed to (trivially) conform to the input — feedback is not needed. That is, without flexibility there is no need for feedback.

Most generators do without syntactic feedback also. They never output a word without knowing in advance the implications for the sentence's syntactic structure — they simply choose a construction and then 'execute' it straightforwardly. Only one construction is active at a time, and its effects are predictable, so no feedback is needed. In general, most generators are crafted so as to work on only one goal at a time.

## 2.4 The Trouble with Serial Algorithms

By and large, current generators do not embody the above design principles. One reason is the structuralist legacy from linguistics, as discussed in §4.5.1. Another reason is the pervasive influence of the von Neuman architecture in AI models, the topic of this section.

An architecture for generation is typically described with: box diagrams, pseudocode, and flowcharts, describing the algorithms used; a module diagram describing the way the various algorithms interact and communicate; and descriptions of the knowledge structures consulted by and the intermediate representations built by the algorithms. The entire field, at least until 1987, can be surveyed in these terms (McDonald *et al.* 1987). Some recent generators are different in important ways, most saliently in using parallelism, but it is still worthwhile to thoroughly examine the weaknesses of traditional approaches, as this has never yet been done, and since many of the weaknesses have been carried over into parallel and even connectionist generators, as discussed in Chapter 7.

When designing a serial generator, one must order choices in a way that makes it possible to respect the dependencies among choices (McDonald 1987b). For example, since the decision of a head can constrain the choice of a modifier (recall *"high winds"* and *"fast currents"*), it apparently makes sense to choose heads first. Another strategy for ordering decisions is to make the 'most constraining decision very early' (McDonald 1991) so it can constrain other decisions. In general, a generator must not make a choice before a choice which it depends on, unless it is prepared to do backup.

There is no consensus as to the correct order of choices. For example, some generators first work out the syntactic form and use that to constrain lexical choice, while others choose words first and use their syntactic properties to constrain syntactic choice, and yet others access the lexicon twice: once to choose syntactically appropriate words, and then again to allow the chosen words to trigger transformations. Indeed, it is not obvious that there exists a total ordering of choices, since they "are all dependent on one another" (Danlos 1984), that is, there are "cross dependencies" among decisions, cases where "the synthesis of element X depends on that of another element Y and ... the synthesis of Y depends on that of X" (Danlos & Namer 1988). Even if there is a workable total ordering of decisions, an algorithm based on it will be necessarily serial, and hence hard to speed up with parallelism.

The root problem is that serial decision making allows only limited interaction. A common but inelegant way to deal with this is to allow backtracking, which allows information considered later to cause the reversal of an earlier decision. A slightly better way to allow a little more interaction is to use pattern-matching. Testing a pattern involves examining a large piece of representation, and the pattern action can alter a large piece of representation. Thus pattern matching is a way of considering a lot of information and making several choices simultaneously. Of course, as patterns get larger problems of cooperation among patterns arise.

Another way to allow interaction is to make many passes over the representation. Danlos (1988) presents a system that performs "a sequence of incomplete syntheses of X and Y". For example, to choose a verb and a direct object, it makes some decisions about the verb form, uses that information to constrain the realization of the direct object, uses that information in order to make more decisions about the verb, and so on until both words are fully realized. This approach unfortunately seems incompatible with incremental generation.

Problems with ordering choices have been a motivation for the use of unification (Kay 1984). The advantage of unification is that it is 'monotonic', that is, the final result does not depend on the order in which the unifications are done. Specifically, "how different constraints interact can be determined at run-time depending on the current context of generation" (McKeown & Elhadad 1991). Unification avoids the problems of ordering choices by simultaneously pursuing all the alternatives for each choice in parallel. While this 'non-determinism' is the conceptual ideal, it is typically implemented with backtracking, so efficiency is a problem.

Another problem with using a fixed order of decisions is that it is ill-suited to incremental generation. Since the order of words can vary from input to input, the decisions about words, at least, must also be allowed to happen in various orders. If the order of choices is made more flexible, then a new problem arises: that of scheduling the decisions at run-time so that no dependencies are violated.

One solution is to use a blackboard (Nirenburg *et al.* 1988). In a blackboard model many decisions can be pending, and, as soon as enough information is available to make some decision, that decision takes place. This has two main advantages: first, parallelism (albeit at a rather coarse grain — the units of computation are large, since they are knowledge sources rather than simply nodes in a network), and second, the separation of control flow from data flow: all information is recorded on the blackboard and is therefore accessible to all processes. Another solution is to use some kind of constraint propagation; this has the same two advantages: parallelism and pervasive flow of information.

The FIG approach also has these two advantages. Since data flow is not constrained by a serial algorithm FIG can, for example, consider the syntactic and collocational implications of a word before that word is finally 'selected'. In slogan form, FIG avoids problems with ordering decisions by simply minimizing the number of explicit decisions. Since the only decisions which are absolutely necessary are decisions of which words to output, that is all that FIG does.

All of these techniques allow one to escape from the basic problem underlying serial generators, namely that "approaches to the question of choice are inexorably tied up with approaches to control" (McDonald 1987b). When making a choice the generator accesses certain knowledge, and when it has made that choice the result becomes available to be considered for later choices. This is the reason that choices must be carefully ordered; the information resulting from a choice is not available until after the choice has been made.

A generator based on serial algorithms typically uses several intermediate representations, both to serve as interfaces among algorithms and as working memories for algorithms. In both cases they serve to record the results of decisions. Often, of course, representations and data structures are defined first, with the algorithms for manipulating and converting among representations being written later. This process of structure-manipulation and structure-building sometimes absorbs an enormous part of the generator's effort, especially if "the utterance is represented at various levels of abstraction throughout the generation process from the objects specified by the application program to the text itself" (Meteer 1990).

Using explicit representations, of syntactic structures for example, is difficult if the order of decisions is flexible. It must be possible to extend or modify the representation at any place as decisions are made or information becomes available §2.3.2. Recent work has started to address this problem, for example 'incremental grammar' (Kempen 1987) and TAGs both allow syntax trees to grow by splicing in subtrees at various points, rather than just by growing from the lower right corner. Such approaches do not appear very successful, as discussed in §4.5.1. The alternative is to dispense entirely with structure building, as in FIG.

An algorithm when running has internal state, consisting of the current position of the program counter, pointers, input parameters, and local variables. This information is implicit in that it is generally not available to other processes. For example, syntactic features and constraints are usually accumulated within the algorithm for syntactic expansion or traversal. Such implicit information is not visible to other processes, which limits interaction.

Also, the state of a given process cannot be reasonably updated except by the algorithm for that process itself. This is a problem for a generator with several processes running in an interleaved fashion; it forces the designer of the algorithm to foresee and test for all the possible things that other processes can do to change the situation. The alternative is to do without serial algorithms and ensure that all information is represented explicitly, which of course makes it more accessible.

Generators vary as to control structure. One traditional dichotomy is that between syntax-driven and input-driven generators (McDonald *et al.* 1987). In syntax-driven generators the grammar specifies what choices are to be made and in what order. The input 'message' is passive — it is merely a repository of information for syntax to access or inspect. In input-driven generators the flow of control is determined by the structure of the input; decisions are typically made as the input is traversed. In this approach, also called 'message-driven' (Meteer 1989), 'data-directed', (McDonald 1983b) or 'direct replacement', syntax is merely a set of options to consider in the course of expressing the input. (Of course, few generators are purely input-driven or purely syntax-driven. For example, Goldman's BABEL is basically input-driven until a verb is chosen, then becomes syntax-driven, using the case frame of the verb to control further processing. Similarly, Hovy advocates the interleaving of 'prescriptive planning', where the operations are guided by the input, and 'restrictive planning', where the decisions are forced by and the opportunities provided by the target language (Hovy 1988).)

McDonald has generalized the dichotomy and analyzed, for a half dozen generators, whether each stage is under the control of the 'representational level' (the input) or the 'reference knowledge' (the knowledge used) (McDonald *et al.* 1987). The diversity is striking, as is the fact that no one has proposed reasons for preferring one approach over the other. This mystery vanishes when one realizes that the entire dichotomy is an artifact of the use of algorithms whose control flow is based on the traversal, access, and manipulation of knowledge structures.

The structure of the knowledge is often determined by the organization of the computation, since control flow and data flow are wed, especially in syntax-driven (or, more generally, knowledge-driven) generators. For example, conceiving of the generation task as involving a series of choices has led to knowledge representations such as discrimination networks for lexical choice and systemic networks for syntactic choice. This is questionable as a way to design knowledge structures — it is not likely to lead to representations which can also be used for other tasks, such as parsing.

To be manageable, each algorithm should only do a few things, thus most generators usually have several different algorithms, for example, a traversal algorithm for syntax, a lookup algorithm for open-class words, and yet another algorithm for inflection.

Accordingly, the knowledge used in generation is generally divided into separate knowledge bases, for example, world knowledge, grammatical knowledge and the lexicon. Each type of knowledge is typically represented separately. Of course, in some cases the causation goes the other way: representations for lexical or syntactic knowledge were designed first, and then it became necessary to build separate algorithms to apply them.

If there are separate algorithms it is natural to divide the process of generation into stages, for example, a lexical lookup stage followed by a syntactic organization stage. Generators which operate in stages limit the availability of different types of information to different times, which is a problem for interaction. Generating in stages of course also constrains the order of decisions, which is a problem for incremental generation, as discussed above. (Both problems can be alleviated by having the various processes run in an interleaved fashion, thus relaxing the rigid separation into stages, as proposed in (Hovy 1988) for example.) If there are separate algorithms there are also typically several intermediate representations, such as a network fragment or proposition, a case frame representation, and a syntax tree.

Thus the notion of modularity comprises generation in stages, diverse knowledge sources, diverse algorithms, and the use of several intermediate representations. (For a similar analysis of the notion of modularity see (McDonald 1988).) Each of these types of heterogeneity is a bad thing for generation, due to the way it limits interaction (§2.1.2).

Every previous generator is, if not modular, heterogenous in at least one respect. The generators with a single algorithm, including Appelt's planning-based system (Appelt 1985) and Kalita's connectionist system (Kalita & Shastri 1987), employed levels of representation. Jacobs' KING, which exploited a uniform representation (Jacobs 1985a), relied on separate processes for concept choice and syntactic choice. Most generators are modular in several ways. For example, consider a traditional multi-pass generator which first plans the content, then the syntactic structure, then chooses the words, then inflects them. Such a system has several stores of knowledge, orders choices into stages, has several algorithms, and works with several intermediate representations.

The alternative to modular design is to build generators which are as homogenous, integrated, and uniform as possible, as was done in FIG.

There are vast number of linguistic options available to a generator. At first glance this suggests that the task of a generator is to 'make choices' among them. Indeed, choice has been called the key problem in natural language generation (McDonald 1983b). It is obvious that a generator must somehow choose words. It is tempting to think that a generator should also make syntactic choices, conceptual choices, or even more abstract choices, such as what concept to bind to a slot, what link of a network to traverse, what branch of a conditional to take, or what decision to make first.

If there are explicit choices then the question of how to organize them into an algorithm immediately arises. But if explicit choices (except for words) are dispensed with, the problem disappears.

Serial algorithms make a generator harder to extend — in order to add a new type of knowledge, such as prosody, it is necessary to add a new knowledge source, and new mechanisms (procedures) to apply it, each probably with its own internal representation, and then to add paths of information flow so that this new type of knowledge can be integrated with other types of knowledge, and then to work the new procedures into the control structure of the generator, that is, to order it with respect to the other processes. This is true not only for obviously procedural systems but also for systems based on ATNs or systemic grammars, where the choice systems behave as programs, being 'executed' by a special interpreter. It is, at least in principle, easier to add new types of knowledge to a declarative than to a procedural system, as pointed out by (Wilensky & Arens 1980; Jacobs 1985a).

Therefore, the idea of algorithm, involving a serial order of choices, each of which depends on some pieces of information, results in making other pieces of information available, and has its own process state, is problematic in many ways.

# Chapter 3

# Lexical Knowledge and Word Choice



This chapter discusses the issues involved in word choice, presents an idealized model based on spreading activation, and illustrates this by explaining the specifics of FIG. The basic idea is that lexical knowledge, like all knowledge, can be encoded with links which conduct activation. Table 3.1 summarizes the organization of this chapter. The details of FIG's processing are postponed to Chapter 6. A survey of lexicons for generation can be found in (Cumming 1986).

| Knowledge of a word's ... | Discussed in ... |
|---|---|
| meaning | §3.1 and §3.4 |
| syntactic properties | §3.2 and Chapter 4 |
| morphology | §3.3 |
| collocations | §3.3 and §3.5 |
| genre and register | §3.3 |
| spelling and pronunciation | §3.3 |

Table 3.1: Types of Lexical Knowledge

## 3.1 Meanings of Words

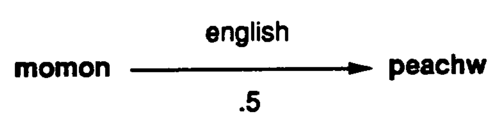


Figure 3.1: Knowledge about the Word *"peach"* and its Meaning

The fundamental task for a generator is: given some concepts to express, find appropriate words. In FIG there are links between concepts and words that can be used to express them. At run-time activation flows from concepts to words across these links. For example there is a link from **momon** (peach) to **peachw**, as seen in Figure 3.1. Relations, like other concepts, can have links to words; for example the **sourcer** relation has a link to the word **fromw**.

FIG actually has two different types of concept-word links: **english** and **japanese** ones. Links of one type are disabled when the other language is being generated. (This is not intended as a serious model of the knowledge of bilingual speakers.)

Before explaining more interesting cases, it is necessary to digress for a moment on the notion of 'word meaning'. Philosophers of language have struggled to define word meaning, and have invented distinctions between meaning and conditions of use, between dictionary knowledge and world knowledge, between denotation and connotation, and so on. None of these distinctions are particularly compelling, or I believe, necessary. FIG accordingly does not have any encoding of 'word meanings' as such. Instead there are links which represent the relationships between words and elements of meaning. Nor does FIG have a separate lexicon, unlike most other generation systems. It follows that it does not have any notion of 'the structure of the lexicon'. While linguistic studies of word meanings frequently taxonomize and organize the words of a lexicon into pairs of opposites, arrays (or 'paradigms') such as *"uncle"*, *"aunt"*, *"father"*, and *"mother"*, semantic fields such as *"small"*, *"medium"*, *"large"*, and *"extra large"*, and so on (Fillmore 1978; Evens 1988), such regularities seem to be facts about the relations among concepts, not words.

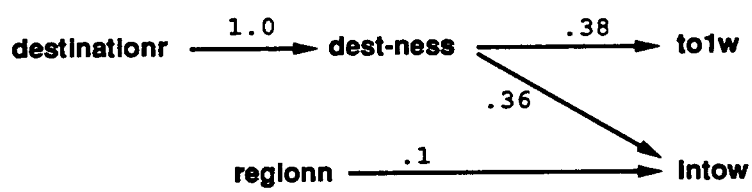


Figure 3.2: Graphical Representation of Some Knowledge About the Word *"into"*

Of course, not all relationships between concepts and words are one to one. Some concepts can be expressed in several ways; the choice can depend on syntactic constraints or on the presence of other concepts in the input. In FIG such concepts have links to more than one word. For example, there are links from **kurun** to **comew** and to **towards**; from **kawan** to **streamw** and **riverw**; and from the relation **destinationr** (used to relate motions to their destinations) to **to1w** (representing the preposition *"to"*), **intow**, and **towardsw**, as shown in Figure 3.2.

When a concept has links to more than one word, there will be a 'default' word, that is, the word to use in the absence of context, including the absence of other concepts to express and the absence of syntactic context. For example the default way to express that some location is the 'destination' of some action is with the preposition *"to"*. In other contexts different words are more appropriate. For example, if the destination of the motion is a region, the appropriate preposition is *"into"*.

In FIG defaults are represented by weights on links. For example, as shown in Figure 3.2 the weight of the link from **destinationr** to **to1w** is .38, higher than the other link weights. The activation transmitted across a link is the product of the activation of the source node and the link weight, so when generating, if no nodes other than **destinationr** are activated, **to1w** will become the most highly activated word. In this way FIG produces, for example, *"the old man went to a mountain"*.

The fact that *"into"* is appropriate if the destination of the motion is a region is encoded with a link from the concept **regionn** to the word **intow**. If both **regionn** and **destinationr** are activated, then **intow** will receive activation from both, resulting in the output of, for example, *"the old man went* into *the hills to gather wood"*. (There is a potential problem with this representation of the meaning of *"into"*: it does not specify that **regionn** must be a property of the destination. In other words, there is a chance that **intow** could be selected due to activation from some completely extraneous location that was characterized as a **regionn**. The reason that this does not happen is explained in §4.2.3.)

In general, a word with a complex meaning simply has incoming links from many concepts. At run-time such a word will receive activation from more than one node. Since the activation of a word depends on the sum of the activation it receives from all sources, the most appropriate word will get a higher 'relevance' score.

Parenthetically, there is an intrinsic bias favoring words with 'large' or 'specific' meanings. Such words become highly activated simply because they receive activation from many nodes of the input. This is in general a good thing; it allows a generator to produce utterances like *"a* floating *peach"* instead of *"a peach located at the surface of the water and supported by the water"*. FIG handles this without additional mechanism; other generators appear to need special knowledge structures or explicit rules to this effect. KING's knowledge, for example, consists of a taxonomy of concepts (Jacobs 1985a), which gives it an automatic metric of specificity, which enables it to easily choose the most 'specific' word. Hovy's PAULINE chooses the word whose meaning configuration is 'largest', that is, the one whose meaning subsumes as much of the input as possible (Hovy 1988).

In FIG a word can be accessed via any part of its meaning. All types of evidence are considered in parallel; that is, there is evaluative parallelism. This is possible since all sources of activation are active simultaneously, thanks to knowledge-source and part-wise parallelism.

FIG is not unique in treating all components of meaning equally; for example, PHRED also does so (Jacobs 1985b). Many generators, however, treat different factors differently. The classic example is BABEL (Goldman 1974). In BABEL verbs are organized into groups with similar meanings as called for by conceptual dependency theory. For example, *"drink"*, *"eat"*, *"inhale"*, and *"take"* are all listed under the primitive *'INGEST'*. Given this structure, BABEL accesses words using two distinct processes: one to choose a primitive and access the associated discrimination network, and one to traverse the discrimination network to choose among the set of words for that primitive.

Another common way to separate word choice into sequential stages which apply different types of knowledge is to distinguish indexing and evaluation. That it, it is common to first find a handful of candidates, by dictionary lookup, pointer following, or checklist evaluation, and then use a separate process to filter out the irrelevant ones (Hovy 1988) or to evaluate their appropriateness to choose the best (Nirenburg *et al.* 1988). In FIG there is no distinction between indexing and evaluation. Both are done using the same knowledge and the same mechanism. That is, the flow of activation across a link handles what other systems would call 'finding' a word and 'evaluating' its relevance.

FIG has no notion of 'necessary and sufficient conditions' that must be satisfied to use a word. A word gets chosen not because it is appropriate in any absolute sense, but simply because it is more appropriate than any other word. Many other generators have absolute criteria for the use of a word; these are often not explicitly represented but are implicit in the mechanism used to find the word. A related point is that FIG does not consider evidence *against* the use of words. This is in contrast to systems like BABEL, which can be described as doing word choice by process of elimination. FIG takes the best word, whether it is perfectly appropriate or not; this is of course a more robust strategy, in the sense that it will always say something.

FIG can be seen as performing a process of matching inputs to word profiles, although there is of course no explicit unification process. When evaluating word choices FIG ignores link labels. To use unification terminology, it only considers the values, whereas most generators consider both attributes and values. In most cases it is clear that, given the value of an attribute-value pair, the attribute is redundant. For example, in *'age=young'*, part of Nirenberg's definition of *"boy"* (Nirenburg *et al.* 1988) *'age'* seem superfluous — there is nothing else that *'young'* is likely to be the value of. In FIG terminology, all that is needed is a simple link from **youngn** to **boyw**. The effect is the same as that of 'select the best match', although the mechanism is simpler.

## 3.2 Syntactic Properties of Words

To handle classes of words with identical syntactic properties FIG has the notion of syntactic category. Syntactic categories include major categories, such as 'noun', and minor categories, such as 'count noun' and 'mass noun'. Categories are represented as nodes. Their primary functions are to provide paths for activation flow from constructions to words and from words to constructions.

An example of the first use of categories is the link from **adj-modc**, representing the Adjectival Modification Construction, to the syntactic category **adjective**. Since **adjective** is linked to all nodes representing adjectives, at run-time activation will reach all adjectives. In this way syntactic evidence for words is also handled by activation, and it is considered at the same time as other evidence — that is, there is evaluative parallelism. Unlike most other generators, there is no special mechanism for accessing syntactically appropriate words.

An example of the second use of categories is the flow of activation from every count noun to **determinatc**, representing the Determination Construction. While the details are somewhat complicated (§4.1) the basic idea is that this path of activation flow is due to the node **cnoun**, representing the syntactic category 'count noun'. In addition to such paths of activation from 'heads' to constructions, there are also direct links from words to constructions representing their valences.

In order to ensure that the words emitted are both syntactically and semantically appropriate, FIG is designed so that a word does not become strongly activated unless it receives both syntactic and semantic activation. To be specific, the activation level of a word is a function of the *product* of the semantic activation and the syntactic activation it receives. This 'multiplication rule' is probably not essential but experience has shown it to be useful, as discussed in §6.5.2.

An important characteristic of FIG is that it chooses all types of words in the same way. Everything which affects word choice is just a source of activation, and all words are chosen in this way. Many generators use different algorithms to choose different types of words, and often also choose different types of words at different 'stages' of the generation process. Commonly distinguished are open-class words and closed-class words (Pustejovsky & Nirenburg 1987), or content words and function words (Kempen & Hoenkamp 1987). Other common distinctions are those between phrase-heads and modifiers (Goldman 1974), and between words with valence and words without. Effectively (and often actually) there are separate lexicons. Of course it is true that different types of factors are more important for different types of choices. For example, activation from the nodes of the input is typically more important for open-class words, and activation from constructions and from relations is more important for closed-class words.

However, such statistical correlations need not be reflected in the generator's design.

## 3.3 Other Properties of Words

A word may be associated with a specific genre or register. In FIG these also are represented by nodes, and also affect word choice via activation flow. For example, **fairy-talen** is linked to **once-upon-a-w** (the node for *"once-upon-a-timew"*). If **fairy-talen** is activated, then **mukashin** (long ago) will probably be expressed as *"once upon a time"*, otherwise it will probably be expressed simply as *"long ago"*.

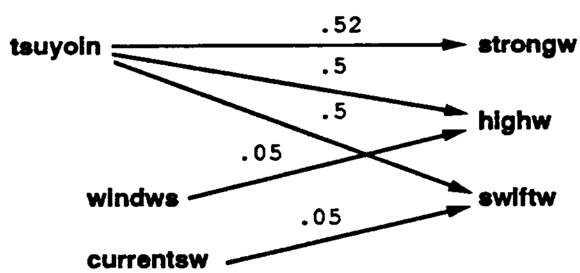


Figure 3.3: Some Knowledge of Collocations

Words also have collocational properties. For example, adjectives and nouns often collocate, as seen by the fact that "swift *currents*" and "high *winds*" sound fine, but "high *currents*" and "swift *winds*" do not. In FIG there are links among collocating words, as illustrated in Figure 3.3. Thus, for example, if **windsw** has activation some activation will spread to **highw**. Other things being equal, such activation will make **highw** be more activated than words such as **swiftw**, **strongw**, or **fastw**. Thus FIG produces *"high winds"* but *"swift currents"*.

A word also of course has a spelling, a pronunciation, and morphological properties. I have nothing new to say about these. FIG-the-program, of course, has mechanisms for retrieving spellings and handling inflection; these are discussed in §6.4.1.

At this point I should acknowledge the existence of problems in defining the notion of 'word' in the face of the existence of homonyms, polysemes, idioms, and derivational morphology. This issue is beyond the scope of this research; fortunately none of my claims depends on an exact definition of what a word is. For the record, different morphological forms are treated in the implementation as follows: *"criticize"* and *"criticizing"* are different forms of the same word, but *"criticize"* and *"criticism"* are different words. Of course, these two are still intimately related by virtue of the fact that they are linked to the same concept.

The existence of homonyms raises a interesting issue. For parsers lexical ambiguity is a key problem but for most generators it does not matter at all. However in FIG it is important to keep homonyms separate. The reason is that many nodes and words are active in parallel, because of part-wise parallelism, and it would be a problem if the evidence for one homonym were confused with evidence for another. For example, consider **to1w** the preposition and **to2w** the infinitive verb marker: if there were only one node for both of these, FIG probably could not produce *"the old man went* to *the hills* to *gather wood"*, since the activation intended for the two *"to"*s would be added together, and *"to"* would be likely to be emitted prematurely.

In FIG word sequences with non-compositional meaning are treated simply as nodes. It does not address the issue of the way in which the meaning of the whole can be motivated by the meanings of the parts. This subsumes the idea of a phrasal lexicon which includes standard ways to refer to things, (Becker 1975). The difficult issue for generation is that some idioms must be subject to syntactic manipulation. For example, the verb in *"kick the bucket"* can be inflected. In other words, idioms can have 'internal syntax' (Fillmore *et al.* 1988). FIG handles such idioms as constructions, whose productive syntactic properties are determined by synergy with other, more general, constructions, as described in §4.4.2.

## 3.4 Inference

This section discusses the role of inference in generation. I do not intend to say anything new about knowledge representation or inference in general — only to address two basic issues as they relate to generation, namely meaning decomposition and concept recognition.

There are many concepts which cannot be expressed with a single word. For example, there is no English word associated with say, 'the-upper-part-of-the-foot', as opposed to 'sole', nor is there any word for 'plane-or-train-ticket-bought-for-someone-to-come-and-visit-the-buyer'. In such cases the generator must use world knowledge in the process of producing a paraphrase.

Consider the case of the Japanese concept **nagarerun**, which means being in a body of water and moving with that water. In English there is no single word that conveys that information. In FIG this means that there is no direct link from **nagarerun** to any English word. **nagarerun** is however linked to the nodes **in-watern** and **motionn**. These nodes could be called part of its definition, components of its meaning, or merely associatively related concepts. In any case, activation will flow from **nagarerun** via these nodes to words like **floatw**, and **tolw**, **fromw**, **towards**, and **come**, enabling FIG to produce phrases such as *"float down"* or *"come floating"*. Thus FIG accesses words some set of which can indirectly express the meaning at hand.

Thus, thanks to activation flow via nodes and links representing world knowledge, FIG can generate even if the concepts in the input do not have direct links to words. This flexibility is, of course, the primary advantage of representing words simply as nodes in a network.

Many systems treat this kind of inference as 'concept choice' rather than word choice. KING, for example, does essentially all of the word choice task by means of operations over the knowledge network: it 'searches' through world knowledge to find words, traversing only links of certain types, which ensures that it only reaches words with equivalent meaning, such as *"buy"* for commercial-transaction (Jacobs 1985a).

Problems with choosing concepts in ignorance of lexical knowledge were discussed in §2.1.2. To repeat briefly, nuances of the input can affect word choice directly, and conversely, lexical availability can affect what concepts are explicitly expressed. Handling word choice as 'concept choice' does not account for ways of expressing information which differ from language to language. Another problem is the psychological implausibility of treating every case of word choice as an example of concept choice. For example, it would be strange to say that there is an *'into'* concept behind each choice of the word *"into"*, or to say that there is an *'on'* concept distinct from the *'in'* concept in *"he met her* on *the bus"*.

FIG employs competitive parallelism; it considers many candidate words in parallel, spreading activation being an excellent technique for efficient parallel search. Most generators are designed to find only one candidate word. For example, KING uses various 'preferences' to control the choice of links to traverse in the process of finding words to express concepts. This kind of scheme can be considered a heuristically guided search, where various types of evidence bias the *process* of retrieving words. In FIG, in contrast, all evidence contributes directly to the 'relevance score';

that is, considerations have their effect by affecting the score of words, not the process of retrieving them. The general problem with schemes for retrieving only one word at a time is that unless the word retrieved infallibly fits the syntactic environment, backtrack may be necessary.

The problem of building up a sequence of words to express a concept has not been adequately addressed by previous work. One reason is that the concepts used in the input to most generators are inspired by English words, so difficulties are few. Rare is the concept in any standard knowledge representation language which does not directly map to a word, and expressing such concepts, the task addressed by (Sondheimer *et al.* 1990), is rather artificial — the results are simply nouns with a few modifiers, such as *"a message that is a file"*. Such an approach is clearly inadequate for referring to more interesting objects, such as, for example, those orange-and-white-striped sawhorses with flashing yellow lights which are often used to stop pedestrians from falling into open trenches. Expressing such ideas requires world knowledge and reference to sundry properties, for example, reference to shape (either directly or by analogy to things of similar shape), function, color, places where they are usually seen, lettering found on them, clues to places where the hearer is known to have seen them lately, and so on. Of course the task of finding a string of words to express a single concept is in itself artificial — far more common is the task of expressing some configuration of concepts in an utterance.

The second basic issue in inference is dealing with an input which includes many simple concepts. In this case a generator must find words which convey a lot of information. Consider Hovy's example: *"In the primary on 20 February, Carter got 20515 votes. Kennedy got 21850."* and his comment: 'if we want good text from our generators, we have to give them the ability to recognize that *"beat"* or *"lose"* or *"narrow lead"* can be used instead of only the straightforward sentences' (Hovy 1988). In other words a generator must recognize the relevance of a concepts not explicitly in the input.

The current FIG scheme can handle simple cases of concept recognition. For example, it produces *"boy"* from an input consisting of the concepts **otokon** (male) and **ko-n** (child). This happens simply by virtue of the links representing the world knowledge that boys are male and are children. In general, thanks to part-wise parallelism, the whole input activates words for numerous subsets of it. Parenthetically, this problem, 'finding a word to express several concepts' (for example, reaching **boyw** from 'male' and 'child'), is handled in FIG in the same way as 'finding a word for a concept which suits the context' (for example, reaching **intow** from 'destinationr' and 'region').

Using one word to express several concepts is called 'conflation'. Different languages have preferences for what types of information are conflated into words (Talmy 1975). For example, English has a general preference to conflate motion and manner into the verb thus *"the peach* floated *down to the old woman"* is more natural than *"the peach* approached *the old woman floatingly"*. Japanese, among other languages, has the opposite tendency. FIG's treatment of such tendencies is somewhat complicated, since constructions play a role and inference is involved, but the basic idea is simply that the right conflations tend to be chosen simply because of relatively high weights on the links to words.

In FIG concept recognition happens in the process of word choice. It makes no difference whether the network is built to have a concept **boyn** which becomes highly activated and then activates **boyw** or whether **boyw** receives activation directly. In either case, FIG accesses the word **boyw** without requiring a previous 'concept choice' phase. That is, concept choice is emergent.

Behind word meanings stand a great of many interesting types of knowledge, such as frames

(Fillmore 1985). For example, the choice between *"to"* and *"into"* really depends on whether the destination is *thought of* as a region. Various factors come into play here, such as the size of the region compared to the length of the path taken to get there. This could be modeled by having a view-destination-as-region frame, which gets 'recognized as appropriate' because of activation flow, and then activates *"into"*. Some similar knowledge structures may allow FIG to recognize an instance of *'narcissism'* from an input involving *'love'* linked to the same node by both agent and patient links (which it cannot currently do, as pointed out by (Miezitis 1988)). In general it is clear that a lot of diverse types of structure are needed in the network, for example, preposition choice undoubtedly must tie in directly to a spatial reasoning component. In FIG-as-implemented word choice basically starts from just a flat space of features.

Together these two abilities, concept recognition and concept decomposition, give FIG the flexibility to express a complex concept by using another complex concept.

Consider for example the Japanese word *"shibakari"*, which refers to the process of going to uncultivated public land; foraging for underbrush, small twigs and the like; picking them up, breaking them off, or cutting them; bundling them; and carrying them home to use for firewood or to make into a fence. Clearly there is no single English word to describe this activity. But it is not always appropriate to express a concept by mentioning all the facts about it (despite (Tomabechi 1987) — interpolating the definition). Often a generator must find some word or short phrase which is fairly close in meaning. That is to say, perfect fidelity may be sacrificed for the sake of ease of expression.

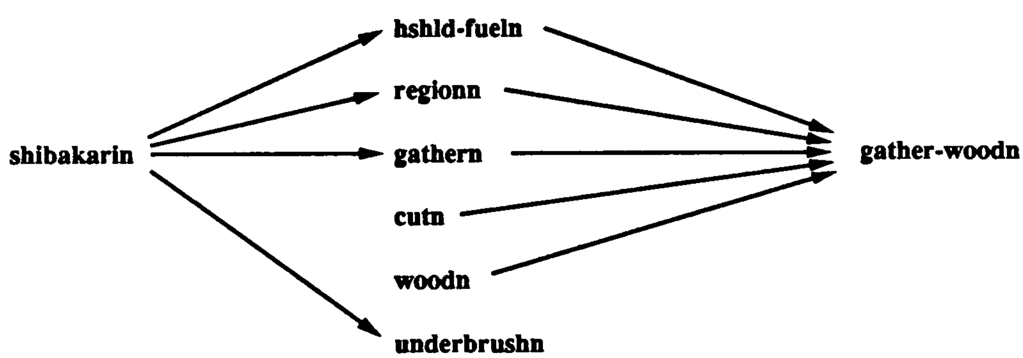


Figure 3.4: Example of Knowledge used in Paraphrase

In FIG such a word can be output thanks to activation flow via world knowledge. Continuing the example, the node **shibarakin** is linked to the nodes **hsehld-fueln** (household fuel), **regionn**, **gathern**, **cutn**, and **underbrushn**. Three of these nodes, **hsehld-fuelc**, **regionn**, and **gathern**, happen to also be linked to **gather-woodn**, which is in turn linked to **gather-woodw**. Thus activation can flow across these links from **shibarakin** to **gather-woodw**, and *"gather wood"* can be produced, as suggested by Figure 3.4. Wanton paraphrase does not occur since activation attenuates every time it crosses a link, which ensures a bias in favor of words which are 'nearer' to the nodes of the input.

The fact that **gather-woodw** can be chosen regardless of the fact that the object gathered was **woodn**, not **underbrushn**, illustrates the point that FIG never rejects a word choice outright — it only 'rejects' a choice when there is something better to choose. Thus there is a kind of 'soft match', although without the cost of matching. A similar point was made by Gasser (1988), and his generator used a similar mechanism.

These kinds of 'flexible' word choice, or 'paraphrase' ability, justify the 'F' in 'FIG'.

## 3.5 Word Choice at Run-Time

Since FIG is an incremental generator the activation level of a word reflects not only its relevance in general, but also its specific relevance at time $t$, for each moment $t$ of the process of producing an utterance. This does not complicate things at all; the scheme described above still works, provided that knowledge about the current state is also represented in the activation levels of nodes. To this end FIG represents the current relevance of concepts and syntactic constructions with activation levels. Thus, word choice at run-time happens exactly for the reasons described above: activation flows across the same links to words, with the clarification that the amounts of activation transmitted depend on the current state.

At each step FIG simultaneously computes the relevance of all words using spreading activation and then chooses the best one. In order for the activation levels of words to correctly reflect their relevance, two basic conditions must hold. First, the node for a word must be linked, directly or indirectly, to nodes representing relevant factors, primarily concepts and syntactic considerations. Second, nodes representing those considerations must be active at the appropriate times. The first condition was discussed in §3.2, the second condition is discussed in Chapter 4.

To a first approximation, all unexpressed nodes of the input are equally active, and syntactic considerations cause the 'selection' of one among those concepts. For example, suppose the input includes nodes like **womann**, **go-n**, **thursdayn**, and **rivern**, and that syntactic considerations are currently activating verbs. Then **go-w** will have the highest activation. It will have more activation than any other verb, since it also receives activation from the input; and it will have more activation than any other word suggested by the input, since it also receives activation from **verb**. Because of its high activation, it will probably be emitted next.

Thus, syntactic and semantic sources of activation are active in parallel; an instance of knowledge-source parallelism.

The approximation is not quite accurate in that it is not quite true that only nodes representing the current syntactic state are active. Were this the case, only words of one specific syntactic category would be active at any given time, which would make interaction among lexical choices impossible. In other words, there is a (superficial) contradiction between choosing words one-by-one and choosing a consistent set of words. The contradiction only holds if the generator has absolutely no idea of what will come next. In FIG, however, activation levels represent relevance in general, not just for the next output position. This is a use of part-wise parallelism. For example, even if an adjective is due to be output next, nouns will have some activation; this can affect the choice of adjective, as required for the *"high winds"* example above. As another example, since FIG may represent that *"the hills"* is likely to be emitted sometime soon, it can use this fact to infer that *"into"* is likely to be appropriate now.

Nor is it quite true that the nodes of the input are all equally activated; there is a kind of 'moving focus of activation', which causes currently relevant nodes of the input to have some extra activation; this is discussed in §4.2.3.

To summarize, at run-time words receive activation from:

– The nodes of the input. Concepts 'within the focus of activation' send out relatively more activation, and concepts which have already been expressed send out none.

– Syntactic considerations. Nodes are active to represent the syntactic categories which are currently appropriate and also those which will probably be appropriate in the near future.

– Other words. This is important for handling collocations.

## 3.6 Comments

This approach is well suited to deal with interactions among choices. Collocations, for example, are easily handled. Most systems do not address the problem of collocations, not because it is intrinsically hard, but simply because it sits ill with compositional approaches to generation.

In general, none of the things discussed in this chapter are handled only by FIG. The primary innovation here is the simplicity of the approach: all knowledge is represented in the network and all computation occurs in the form of spreading activation. There are no complex knowledge structures, algorithms, or intermediate representations. For example, since all words compete to be selected, FIG does not require knowledge structures to represent sets of alternatives, nor mechanisms to choose among words for a concept or for a syntactic role. This is novel: even other connectionist generators organize words into winner-take-all networks or otherwise use extensive inhibition to cut down on the amount of effective parallelism.

# Chapter 4

# Syntactic Knowledge and Its Use



The practical value of syntax for generation is that it causes the right words to be emitted at the right times. Accordingly, syntactic considerations manifest themselves in FIG only through their effects on the activation levels of words. This chapter explains how syntactic constructions, encoded in the network as nodes and links, affect word choice and thereby allow FIG to produce grammatical outputs.

## 4.1 Knowledge of Syntax

This section introduces FIG's representation of syntactic knowledge. The use of this knowledge is discussed in subsequent sections. The knowledge presented here is purely illustrative. I do not claim that it represents the facts of English, nor that it is the best way to describe them in a grammar. The examples are intended simply to illustrate the representational tools and computational mechanisms available in FIG. They are taken directly from FIG, so they are known to at least work. FIG's full grammars of English and Japanese are presented in Chapter 5.

FIG's treatment of syntax is loosely based on Construction Grammar (Fillmore 1988; Fillmore

1989). This is for the purely practical reason that Construction Grammar is well suited to implementation in a network, as will become apparent in subsequent sections. Construction Grammar is a theory which describes the grammar of a language "directly, in terms of a collection of grammatical constructions" (Fillmore *et al.* 1988; Fillmore 1989), rather than incorporating transformations, for example. Construction Grammar is associated with a willingness to notice very specific constructions with very specific meanings and functions and with a reflex to look closely at the semantics underlying constructions.

As a simple example, consider the Existential There Construction (Lakoff 1987), as in *"there was a poor cobbler"*. To simplify, this construction is used to introduce a new item into the discourse, and it consists of the word *"there"*, a verb, and then a description of the item being introduced, in that order.

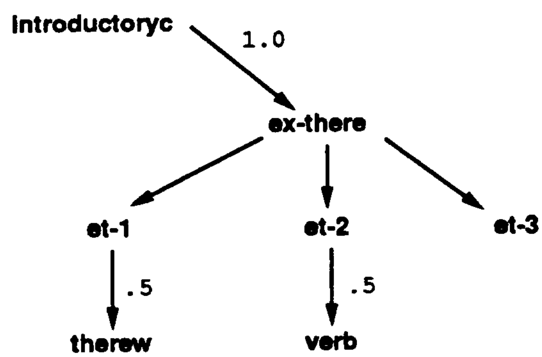


Figure 4.1: The Existential There Construction as a Fragment of the Network

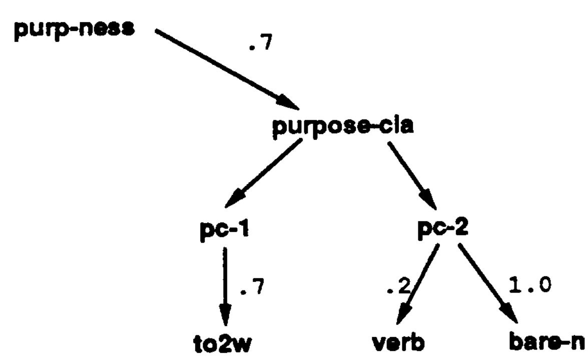


Figure 4.2: The Purpose Clause Construction

In FIG constructions and constituents are simply nodes. Knowledge about the Existential There Construction is encoded as shown in Figure 4.1. There is a link from the concept **introductoryn** to **ex-there**, the node representing the construction. In turn, there are links from **ex-there** to its three constituents; these are ordered, although this is not shown explicitly in the diagram. From constituents there are links to nodes specifying their 'content': **et-1** is linked to **therew** and **et-2** is linked to **verb**. The numbers on the links are their weights. The significance of weights is a complex issue; it is discussed in the next section and in §6.5.3.

As another example of a construction Figure 4.2 shows FIG's definition of **purpose-cla**, representing purpose clauses with *"to"*. This construction has two constituents: **pc-1** and **pc-2**, which are to be realized as the word *"to"* and a verb phrase with a bare verb, respectively. (The way that a full verb phrase can be built even though only **verb** is specified explicitly is explained in §4.4.2.) Purpose clauses are used to express the **purposer** relation (although this activation actually comes via the node **purp-ness** for reasons discussed in §6.2).

Other examples of constructions in FIG are **subj-pred**, representing the subject-predicate construction, **change-statec**, representing the intransitive valence of state-change verbs such as *"break"* and *"die"*, and **kick-bucketc**, representing the construction for *"kick the bucket"*. These and all of FIG's constructions are described in Chapter 5.

Despite FIG's the novel processing mechanism, the factors determining the syntactic form of its output are all familiar. That is, there is nothing surprising about the sources from which constructions receive activation. Briefly, constructions are associated with their meanings, with words for their constituents, and with other constructions; the remainder of this section gives examples.

There are links from concepts to constructions. For example, as seen above, the concept **introductoryn** has a link to **ex-there**. As with word choice, activation can reach constructions via multi-link paths. For example, if **kurun** (coming) is active, activation will flow via its supertype, **motionn**, to **dir-particlec**. There are also links from relations to constructions, for example, the **purposer** relation is linked (indirectly) to **purpose-cla**. Thanks to this link, whenever a node which fills a **purposer** relation is activated, **purpose-cla** will become activated. There are links from nodes representing genre and register to constructions, for example, there is a link from **slangn** to **kick-bucketc** (for *"kick the bucket"*); this causes FIG to produce *"Mary kicked the bucket"* instead of *"Mary died"* when the node **slangn** is activated.

There are links from words to constructions representing their valence, for example, from **go-w** to **go-c**, and from words to constructions representing their 'maximal projections', for example, there is a link from every count noun to **determinatc**, the determination construction.

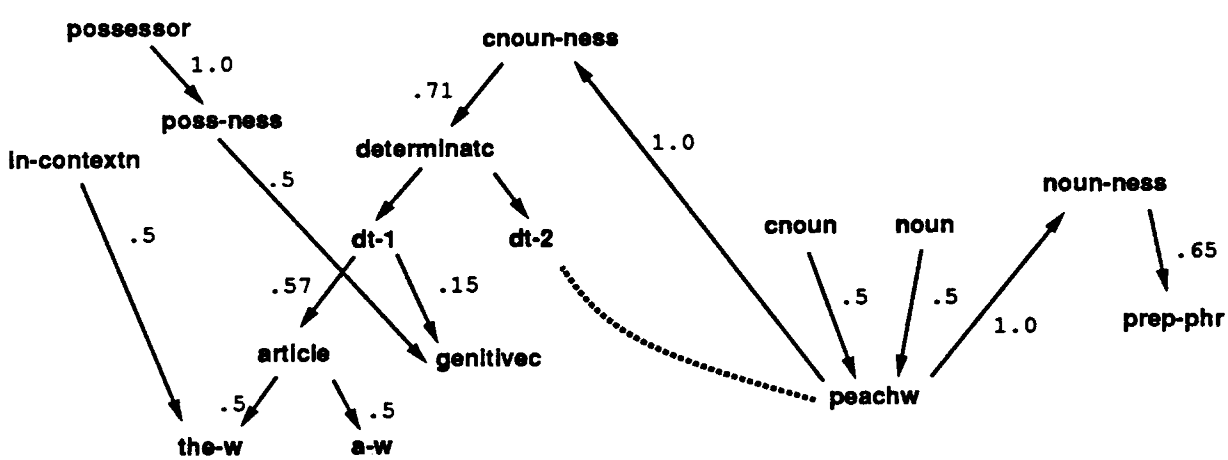


Figure 4.3: The Determination Construction

Constructions are also associated with other constructions. For example, there is a link from the first constituent of **determinatc**, the Determination Construction, to **genitivec**, the Genitive Construction, as shown in Figure 4.3. This link encodes the fact that the determiner can be a possessive noun.

In summary, FIG constructions receive activation from words, concepts, relations, genres, and other constructions. All these sources of activation are considered in parallel and summed, that is, there is evaluative parallelism. Most generators, in contrast consider one kind of factor for any given construction, that is, construction X may be chosen top-down (due to another construction) whereas construction Y may be chosen bottom-up (due to its head).

## 4.2 Syntactic Knowledge in Action: Basics

This section explains how such knowledge structures enable FIG to produce grammatical sentences. The basic idea is that, like other nodes, nodes representing constructions and constituents have activation levels which represent their current relevance. The activation sent from these nodes to various other nodes affects the syntactic form of the utterance.

This section merely explains how FIG handles some basic issues in syntax. More interesting examples and comparisons to other approaches are deferred to later sections.

### 4.2.1 Constituency and Subcategorization

The links described above suffice to handle constituency. Consider for example the fact that count nouns must be preceded by determiners (in FIG's subset of English). To be concrete, suppose that **peachw** is activated, perhaps because a **momon** (peach) concept is in the input. Activation of **a-w** and **the-w** will then happen, thanks to the path of activation flow from **peachw** via **cnoun-ness**, **determinatc**, **dt-1**, and **article**, as can be seen in Figure 4.3. In this way the activation of a count noun causes an increase in the activation levels of articles. Provided that other activation levels are appropriate, this will cause some article to become the most highly activated word, and thus be selected and emitted. Note that FIG does not first choose to say a noun, then decide to say an article; rather the 'decision' to use an article and the 'decision' to output it before the noun both result from the fact that **determinatc** causes articles to become highly activated at the right time.

As mentioned in Chapter 3, the most highly activated word typically becomes that way because it receives activation from both semantic and syntactic sources. Strong syntactic activation can compensate for relatively weak semantic activation. (By 'syntactic activation' I mean simply activation whose source is a syntactic construction; activation from nodes of the input will be called 'semantic activation'.) An obvious example of this is articles, which have little semantic motivation, but compensatory high weights on the path which brings them syntactic activation. Another example is verbal particles of direction, as in *"she went* **down** *to the stream"*. Here the direction can be expressed with one little syllable, so it is easy for speakers to include that information even if not important, whereas in languages without this option, such a concept could perhaps only be expressed in a more complex structure, and hence would not be included in the output unless important to say. Accordingly, the weights of FIG's **motion-vp** construction are such that verbal particles of direction get a great deal of syntactic activation. In this way constructions can affect what information is expressed explicitly.

Constructions can also affect the syntactic form in which information appears, so constituents can be linked to syntactic categories. These can determine what part of speech to choose, for example *"criticism"* versus *"criticize"* as the subject of a sentence. Such links can indifferently be viewed as saying 'if there's some information which can be expressed with a noun, now is the time', or 'whatever you want to say now, try to express it with a noun'. Constructions also affect inflection. For example the second constituent of **purpose-cla** is linked to the node **bare-n**, which causes the verb to appear in bare form.

Constituents can also refer to concepts and semantic relations. This is how FIG ensures that concepts appear in the right syntactic positions. For example, the second constituent of **transitivec**, representing the Transitive Construction, is linked to **patientr**, which causes patients to often be expressed as direct objects. The details of activation flow via relations to concepts are given in §6.2. Allowing constituents to refer to concepts and relations is not common outside the framework of Construction Grammar, but appears necessary to properly handle certain phenomena (Jurafsky 1990).

A constituent may specify both form and content. That is, it may activate a syntactic category node and a relation node, in parallel of course. Consider the Verbal Particle of Direction Construction, as in *"float down"*, *"go away"*, and *"come back"*. To produce such outputs the second constituent of **dir-particlec** must specify that 'the direction of the going' be expressed as a 'verbal particle'. Thus this constituent it is linked to **directionr** and **vparticle**. Activation will thus flow to an appropriate word node, such as **downw**, both via the concept filling the **directionr** slot and via the syntactic category **vparticle**. Thus FIG tends to select and emit an appropriate word in an appropriate form. More commonly, however, a constituent is linked to only one node, and either the form or the content is determined by synergy with other constructions, as discussed in §4.4.2.

Constituents can also, of course, have links to function words. For example, the second constituent of the causative construction is linked to the word *"make"*. Verbs govern case markers in the same way, thanks to links from the construction representing the valence to the appropriate preposition.

### 4.2.2 Word Order

In FIG the activation level of a word must at each time must represent its *current* relevance. In particular, words which are currently syntactically appropriate must be strongly activated, as discussed in §3.5.

This is the responsibility of constructions. To do this they must have knowledge of the current syntactic context. This knowledge is in the form of 'cursors', one per construction, which point to the currently relevant constituent. The cursor 'gates' the links to constituents, so that the link from the construction to the current constituent has weight 1 and the links to all other constituents have weight 0. As a result, appropriate words become activated at appropriate times. For example, when the cursor of **determinatc** points to **dt-1**, articles receive a relatively large amount of activation. Thus, an article is likely to be the most highly activated word and therefore selected and emitted. After an article is emitted the cursor advances to **dt-2**, and so on. The process of updating cursors is described below.

In some cases there is a single construction which is primarily responsible for specifying the ordering, as for determiners and nouns. In other cases the order is determined by the interplay of several constructions. For example, the fact that determiners precede adjectives is encoded by the fact that the links from **cnoun-ness** to **determinatc** and to **adj-modc** have weights of .71 and .7, respectively, which allows **determinatc** to activate articles more than **adj-modc** activates adjectives.

In FIG the representation of the current syntactic state is distributed across the constructions, being represented in the positions of their cursors and in their activation levels. There is no central process which plans or manipulates word order; each construction simply operates independently. More highly activated constructions send out more activation, and so have a greater effect. But in the end, FIG just follows the simple rule, 'select and emit the most highly activated word', and word order is emergent.

§3.5 indicated that activation levels of words must also represent which words are likely to be chosen in the near future, for the sake of dependencies between present and future choices; that is, there must be part-wise parallelism. In FIG there is enough 'stray' activation at all times that any word relevant to any part of the input will receive at least some syntactic activation.

In FIG all constructions are ordered — indeed, the presence of a cursor is the only thing which distinguishes constructions from other nodes. This does not mean that FIG cannot handle 'free word-order languages'; it simply means that constructions play less of a role when generating them.

After a word is selected and emitted, two things must be done to ensure that the current syntactic state of affairs is accurately represented by constructions:

First, the cursors of constructions must advance as constituents are completed. Any number of constructions may be updated after a word is output, for example, emitting a noun may cause both the **prep-phr** construction and the **determinatc** construction to update. After all the constituents of a construction have been exhausted, the construction 'resets', and the cursor returns to its original position. Details are given in §6.4.2.

Second, constructions which are 'guiding' the output should be scored as more relevant. Therefore the update process adds activation to those constructions whose cursors have changed. Thus, even though FIG does not make any syntactic plans, it tends to form a grammatical continuation of whatever it has already output. An example of the importance of this is the production of passive sentences. If, for whatever reason a patient is highly activated, it will be expressed first. If so, the cursor of **passivec** will advance, **passivec** will become highly activated, and it will then activate nodes in order to form a passive sentence. If the patient is not expressed before the verb then **passivec** will not become highly activated and will send out insignificant amounts of activation.

### 4.2.3 The Locality Principle

Words must stand in the correct relations to their neighbors. For example, a generator must not produce *"a large mountain and a small peach"* when the input calls for *"a small mountain and a large peach"*. Similarly, a generator must not produce *"the* **big** *boy went to the mountain"* when the input calls for *"the boy went to the* **big** *mountain"*. This is the problem of emitting the right adjective at the right time, or, in other words, only emitting adjectives that stand in an appropriate relation to the head noun.

This is a serious problem in FIG. Since there is 'part-wise parallelism' — that is, all parts of the input are active simultaneously — words and constructions for all parts of the output are also active simultaneously. Thus there is the potential for 'crosstalk', that is, activation of a syntactically appropriate but 'wrong' concept. The solution is to ensure that related concepts always become highly activated together. In the example, **ookiin** (large) should become activated together with **momon** (peach), not together with **yaman** (mountain). I call this the 'locality principle'.

FIG manifests the locality principle thanks to activation flow among the nodes of the input. For example, if **momon1** (peach) is linked by a **sizer** link to **ookiin1** (large), then **ookiin1** will tend to become highly activated whenever **momon1** is (activation flow among the nodes of the input is discussed further in §6.5.1). Because of this FIG exhibits a kind of 'focus of attention' which successively 'lights up' groups of related nodes, similarly to the 'focus of consciousness' of (Chafe 1980a). This differs from some notions of focus in that in FIG being in focus is not all-or-nothing, rather, a node is in focus to the extent that it receives activation from related concepts which are currently syntactically relevant.

The locality principle is important not just for word order but also for word choice. Recall the crosstalk problem mentioned in §3.1: if a **regionn** node is present anywhere in the input, it could cause the choice of *"intow"* over *"to"*, regardless of whether that region is the destination of the motion in question. Thanks to the locality principle, **regionn** will become strongly active and relevant when, and only when, related concepts, such as the concept of motion to a location that is a **regionn**, are active.

### 4.2.4 Multiple Copies

Consider the problem of generating utterances with multiple 'copies', for example, several noun phrases or several uses of *"a"*. This is handled by having constructions 'reset' once they are complete, that is, once the cursor advances past the last constituent. Some constructions need no resetting, since they have only one constituent. For example, **adj-modc** siphons activation over to the adjective whenever a **cnoun** is activated. Thus any number of adjectives can be produced, and all will appear before the noun, as in *"the strong big boy"*.

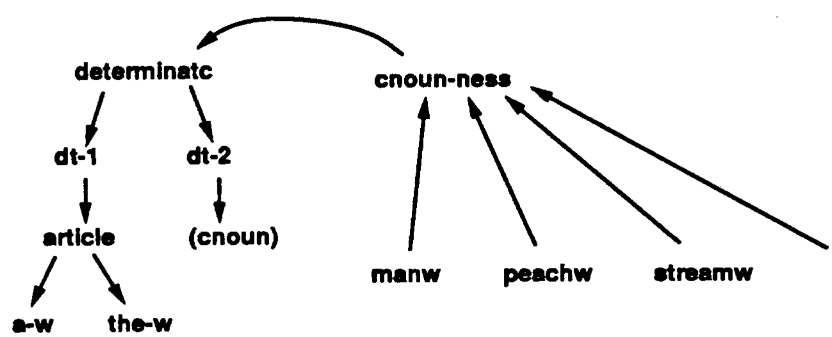


Figure 4.4: Illustrating the High Fan-in to **cnoun-ness**

However the simple reset idea has a potential problem: overactivation. For example since all words of category **cnoun** have links to **cnoun-ness** that node might receive more activation than appropriate, in cases when several count nouns are active, and of course this is the normal case, since FIG has both part-wise parallelism and competitive parallelism. Overactivation of **cnoun-ness** would in turn result in over-activation of, for example, articles, which then could result in premature output of, say, *"a"*. Figure 4.4 illustrates the problem.

To deal with this FIG uses a special evidence-combining rule for activation received by 'collector' nodes, such as **cnoun-ness**: the maximum (not the sum) of these amounts is used. For example, this 'maximum rule' applies to the activation received by **cnoun-ness**; thus it effectively ignores all but the most highly activated noun, and, since **determinatc** receives activation via **cnoun-ness**, it is also effectively only activated by one noun.

An earlier version of FIG handled this problem by actually making copies. For example, it would make a copy of **determinatc** for each noun-expressible concept, and bind each copy to the appropriate concept, and to copies of **a-w** and **the-w**. This worked but it made the program hard to extend. In particular, it was hard to choose weights such that the network would behave properly both before and after new nodes were instantiated and linked in.

Resetting handles multiple copies of constructions well except if the copies are embedded. This latter case is the classic problem of handling 'recursion' which is often raised as a shortcoming of connectionist systems. It is not clear, however, that this is a real problem, since it appears that even people can only cope with limited center embedding, and since it is not clear that it is necessary to use recursion in a grammar at all.

## 4.3 A Simple Example

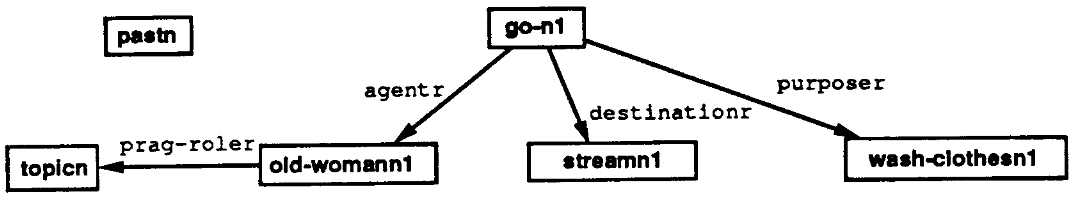


Figure 4.5: An Input to FIG

This section illustrates how various constructions participate in the production of an entire sentence. It describes FIG producing *"the old woman went to a stream to wash clothes"*. For this example the input is the set of nodes **go-n1**, **old-womann1**, **wash-clothesn1**, **streamn1**, and **pastn**, linked together as shown in Figure 4.5. (The names of the concepts have been anglicized for the reader's convenience.)

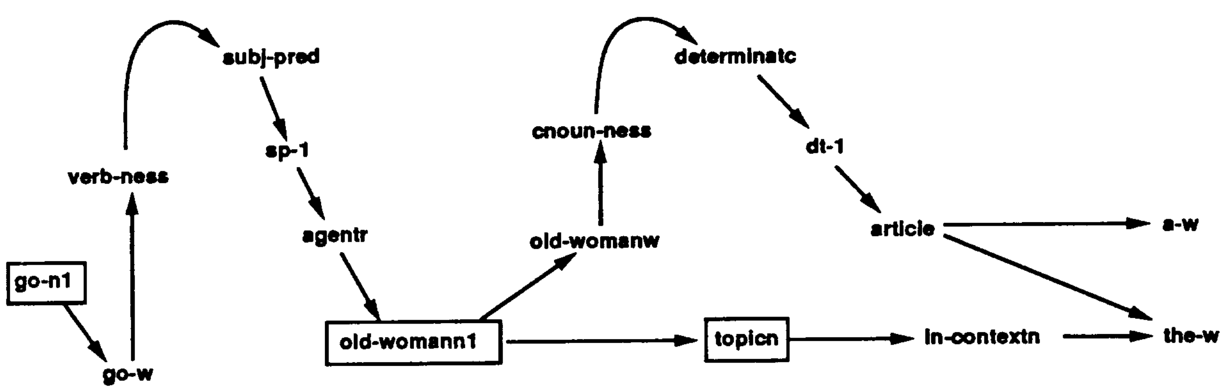


Figure 4.6: Selected Paths of Activation Flow Just Before Output of *"the"*

Initially each node of the input is a source of activation. After activation flows, before any word is output, the most highly activated word node is **the-w**, primarily for the reasons shown in Figure 4.6. That is, thanks to activation from **subj-pred**, the most highly activated concept is the agent, namely **old-womann1**. This concept activates the associated word, **old-womanw**, which in turn activates **determinatc**. The cursor of this construction being on its first constituent, this activation reaches **dt-1** and then **article**. From this, activation flows to **the-w** and **a-w**. **the-w** has more activation than **a-w** because it also receives activation from **topicn**.

Figure 4.7 summarizes the state of the network at this point. The nodes are listed by type and ordered by amount of activation. The positions of the cursors of constructions is shown by capitalizing the relevant constituent. The nodes **obaanl** and **ikunl** are the ones that are being anglicized as **go-n1** and **old-womann1**, respectively.

After *"the"* is emitted the update mechanism moves the cursor of **determinatc** to **dt-2**. The

| —constructions— | ——words—— | —notions,insts— |
|---|---|---|
| 20.7 DETERMINATC | 21.6 THE-W | 24.4 AGENT-NESS |
| DT-1 dt-2 | 20.2 A-W | 22.7 NOUN-NESS |
| 20.5 PREP-PHR | 18.0 OLD-WOMANW | 22.7 VOWELI-NESS |
| PP-1 pp-2 | 14.9 GO-W | 22.7 CNOUN-NESS |
| 20.5 ADJ-MODC | 11.0 STREAMW | 20.2 OBAAN1 |
| AM-1 | 8.9 TO2W | 19.3 VERB-NESS |
| 15.3 SUBJ-PRED | 8.4 RIVERW | 17.9 IKUN1 |
| SP-1 sp-2 sp-3 | 7.9 WASH-CLOTHESW | 16.9 DEST-NESS |
| 7.7 PURPOSE-CLA | 5.6 BY-W | 15.1 PURP-NESS |
| PC-1 pc-2 | 3.2 TO1W | —cats,rels— |
| 3.0 GENITIVEC | 3.0 INTOW | 15.5 ARTICLE |
| GE-1 ge-2 | 2.3 TOWARDSW | 14.8 ADJECTIVE |
| 2.8 GO-C | —— | 13.6 NOUN |
| GV-1 gv-2 | —— | 11.5 VERB |

Figure 4.7: Activation Levels of Selected Nodes Just Before Output of *"the"*

most highly activated word becomes **old-womanw**, largely due to activation from **subj-pred** via **agentr** and **old-womannl**.

After *"old woman"* is emitted **determinatc** is reset — that is, the cursor is set back to **dt-1** and it thereby becomes ready to guide production of another noun phrase. Also, the cursor on **subj-pred** advances to **sp-2**. As a result verbs, in particular **go-w**, become highly activated.

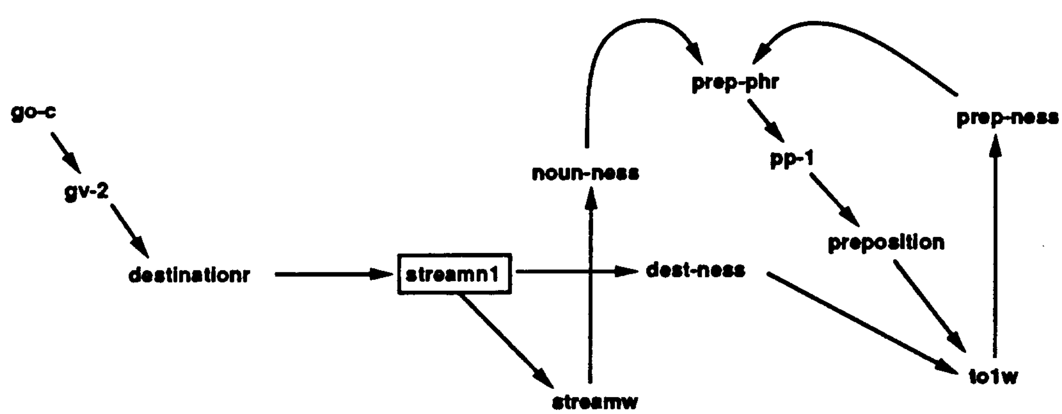


Figure 4.8: Selected Paths of Activation Flow Just Before Output of *"to"*

**go-w** is selected. Because **pastn** has more activation than **presentn, infinitiven** and so on, **go-w** is inflected and emitted as *"went"* (the inflection mechanism is described in §6.4.1). **go-c**'s cursor advances to its second constituent, which activates **destinationr**. (This construction exists exactly for this reason, to make the destination appear before the purpose.) **to1w** becomes the most highly activated word, due in part to to syntactic activation from the first constituent of **prep-phr**, which in turn receives its energy from **gv-2**. Of all the prepositions in the network, **to1w** receives the most activation since it receives semantic activation from **dest-ness**, which receives its activation, ultimately, from **gv-2**. This is shown in Figure 4.8.

After *"to"* is emitted, the cursor of **prep-phr** is advanced. Moreover the relation **destinationr** is marked as expressed, thanks to *"to"* being output. As a result, **streamnl**, the filler of

**destinationr**, enters the focus of attention, which increases its activation level. The most important path of activation flow is now from **streamnl** to **streamw** to **determinatc** to **article** to **a-w**. Thus **a** is selected. The inflection mechanism produces *"a"* not *"an"* since **consnt-initial** is more highly activated than **vowel-initial**.

Then the cursor of **determinatc** advances and *"stream"* is emitted.

At this point the only node of the input not yet expressed is **wash-clothesnl**. This activates the relation it fills, namely **purposer**. Activation flows from this via **purp-ness** to **purpose-cla**, thence to its first constituent **pc-1**, and thence to **to2w**. After *"to"* is output the cursor advances, and *"wash clothes"* is output as a bare verb, thanks to the link from **pc-2** to **bare-n**.

Now that all the nodes of the input are expressed, FIG ends, having produced *"the old woman went to a stream to wash clothes"*.

## 4.4 Implications

§4.2 completely described all the syntactic mechanisms of FIG; there are no more. There are, however, some non-obvious implications; these arise because many constructions are active at once.

### 4.4.1 Competition and Emergent Choice

This subsection illustrates how constructions affect competition among words and how constructions themselves compete.

Recall that for many concepts there is a default word to use (§3.1). In FIG a construction can override these defaults simply by strongly activating an alternative word. An instance of this is the way that some verbs govern prepositions (or post-positions). For example, although patients are usually marked with *"o"* in Japanese, the patient of the verb *"kissu suru"* (kiss) is marked with *"ni"*[1]. In FIG this is handled simply by a valence link from **kissuw** to **patient-ni**, the construction which specifies that the patient be marked with *"ni"*. Activation from this construction to *"ni"* overrides the default.

Another example of competition is the production of *"John broke the dish"* and *"John made the dish vanish"* from analogous inputs[2]. Certain verbs, including *"break"*, allow lexical causatives, whereas for other verbs, including *"vanish"*, a periphrastic causative must be used. This is an example of lexical knowledge which constrains syntactic form. In FIG the possibilities are represented by **per-causativec**, representing the Periphrastic Causative Construction, the default, and **lex-causativec**, for the Lexical Causative Construction. Certain verbs,xincluding **breakw**, have a link to **lex-causativec**. After the causer of the event has been expressed, the cursors of both constructions advance. **per-causativec** activates the word **makew**, and **lex-causativec** activates the category **verb**. The weights are chosen such that **makew** is selected as the default, but if **lex-causativec** is receiving activation (for example, from **breakw**) then the most semantically appropriate verb ( *"break"* in the example) will be selected. Thus the rivalry between

[1] I do not mean to make any general claim about deep case here; the point is simply that *"kissu suru"* patterns differently from other verbs, and, since this is a special fact about Japanese, FIG must handle this in its representation of knowledge of Japanese.

[2] The inputs are analogous in that both have a **causer** and a **patientr**. Thus the meaning behind this use of *"break"* is effectively 'cause to become broken', which one could argue to be an unnatural representation. The issue of whether this is correct is not really relevant here, however: the point is that the meanings of the two sentences both have a causative component, and, since in other languages these two meanings could be realized in exactly the same way, there must be an account of the way they are realized differently in English.

these two constructions is played out as the rivalry between two words. FIG never chooses one or the other construction explicitly — it simply doesn't need to. Thus the syntactic form of its utterances is emergent.

A similar type of competition occurs when the input only includes an action and its patient. In this case FIG will produce either an intransitive form, such as *"the dish broke"*, or a passive form, such as *"the peach was eaten"*, depending on the verb[3]. The passive form is the default, but certain change-of-state verbs, such as *"vanish"* and *"break"*, have an alternative valence which allows an intransitive, patient-as-subject utterance. In FIG there is a construction, **change-statec**, which specifies that the verb can directly follow the patient. For certain inputs **passivec** and **change-statec** are in effect rivals; the rivalry is played out as competition between **is2w**, activated by **passivec**, and 'open-class' verbs, such as **breakw**, which are activated by **change-statec**. Weights are chosen so that the passive form is the default, but if **change-statec** is activated (due to a valence link from a verb) then a patient-as-subject utterance will be produced.

Another case where competition among words results in alternative syntactic forms is the case of 'optional' constituents. By optional constituents I mean syntactic positions where the generator has the option of outputting a word, depending on whether the input includes suitable information. For example, pre-nominal adjectives appear if properties of an object need to be described. In FIG the availability of this option is represented by syntactic activation to **adjective** at the appropriate time. (The source of this activation is explained in the next subsection.) As a result adjectives get syntactic activation, and so to a lesser extent do nouns. There are two cases: If the input includes a concept linked (indirectly perhaps) to some adjective, that adjective will receive activation from that concept. In this case the adjective will receive more syntactic activation than any noun does, and hence have more total activation, so it will be selected next. Any number of adjectives can be emitted before a noun in this way. If the input does not include any concept linked to an adjective, then a noun will have more activation than any adjective (since only the noun receives semantic activation also), and so a noun will be selected next, and the adjective option will expire. In summary, in FIG, the decision to include or to omit an optional constituent (or adjunct) is emergent — if an adjective becomes highly activated it will be chosen, in the usual fashion, otherwise some other word, most likely a noun, will be. This requires no additional mechanism; it occurs simply thanks to partwise parallelism and competition between nouns and adjectives. This is in contrast to most generators, which handle optional constituents with a grammar augmented with specifications of how to test the input to decide whether or not to use an optional constituent and if so what concept it will be used to express, which of course requires a special mechanism to execute these tests.

There is also competition as to which concepts are realized in which locations in the sentence. Consider the question of determining what information gets expressed in subject position. In FIG, **sp-1**,the first constituent of **subj-pred**, which represents the Subject-Predicate Construction, has links to **causer**, **agentr**, and **instrumentr**, with weights .35, .25, and .2, respectively. The effect of this miniature agency hierarchy is that, for example, if present the causer will appear as the first noun phrase of the sentence, if none is present the agent will, and if neither causer or agent is present, the instruments will. This enables FIG to produce sentences with instrument subjects, such as *"the wind broke a dish"*, without additional mechanism. If there is no causer, agent, or instrument a patient will appear as subject, as in *"Mary was killed"*. Other constructions and other sources of activation can also affect what becomes the subject. For example, if the patient has an amount of activation large enough to override the agency hierarchy, FIG produces a by-

[3]Again, I do not claim that the meanings underlying these sentences are necessarily isomorphic, only that they share a component of meaning and could be realized the same way in languages other than English.

passive such as *"Mary was killed by John"*. Thus the 'slot-to-case' mapping is emergent in FIG; this is in contrast to most generators, which explicitly choose which concept to express in each syntactic role.

In all these cases, and also in the determination of word order, as discussed in §4.2.2, FIG does without explicit choice among alternative syntactic organizations. As an example of an implicit choice, consider a FIG output which exhibits an Existential There Construction. The reason will not be because that construction was chosen over, say, the Subject-Predicate Construction; at run-time both of these constructions will have been active. The appearance of syntactic choice arises simply from the fact that the more highly activated construction has a greater effect on word choice.

Most generator designs include explicit syntactic choices. For example, they decide which template to use, which pattern to execute, which arc to traverse, among ways to syntactically realize a constituent, among concepts to use in some grammatical role, how to order words, whether to include or omit an optional constituent, and so on. FIG is thus much simpler in all these respects.

The claim that FIG makes no explicit syntactic decisions has been challenged on the grounds that syntactic update is a kind of decision-making (Miezitis 1988). Syntactic update does involve decisions in the sense that it results in discrete changes in the state of constructions. Yet these are not planning-type decisions; they update the network only to represent the current state after a word has been output. This kind of update is in effect a form of monitoring something in the external world, namely the sounds or words that have been produced. Also, unlike the decisions in most generators, the result is only a change in the pattern of activation, not something qualitatively different, like a branch, a subroutine call, or the formation of a variable binding.

Perhaps one reason for the use of explicit choice in most generators is the linguistic tradition of identifying the structures present in a sentence. On my view constructions participate in language production but do not 'exist' in the resulting utterances. In slogan form, linguistic knowledge exists in the speaker's head, not as part of the utterances he produces. In particular, one can say that in FIG constructions have constituents but the outputs do not.

The fact that syntactic choice is emergent is the reason FIG requires a separate update mechanism. Most generators simply choose a construction and 'execute' it straightforwardly. However in FIG no construction is ever 'in control'. For example, one construction may be strongly activating a verb, but activation from other constructions may 'interfere', causing an adverbial, for example, to be interpolated. Also, considerations having nothing to do with syntax, such as salience, may cause a word to be emitted, and then the generator must continue grammatically. Therefore constructions need feedback on what words have been output.

### 4.4.2 Synergy

An important aspect of FIG is that the final output is determined by various constructions 'working in synergy'. As far as I know, this sort of 'cooperation' among constructions has not previously been used for generation, although it has been used in linguistics. For instance, in Construction Grammar, the syntactic structure of a sentence is accounted for in terms of 'superimposition' of constructions (Fillmore 1989).

A simple example of synergy is the cooperation of two constructions on the choice of a single word. For example, Figure 4.9 shows how the choice of the word *"make"* is governed by **subj-pred** and **per-causativec**. It also shows that, for *"eat"*, the position is governed by **subj-pred** and the form by **per-causativec**. FIG is here in the spirit of Construction Grammar, where

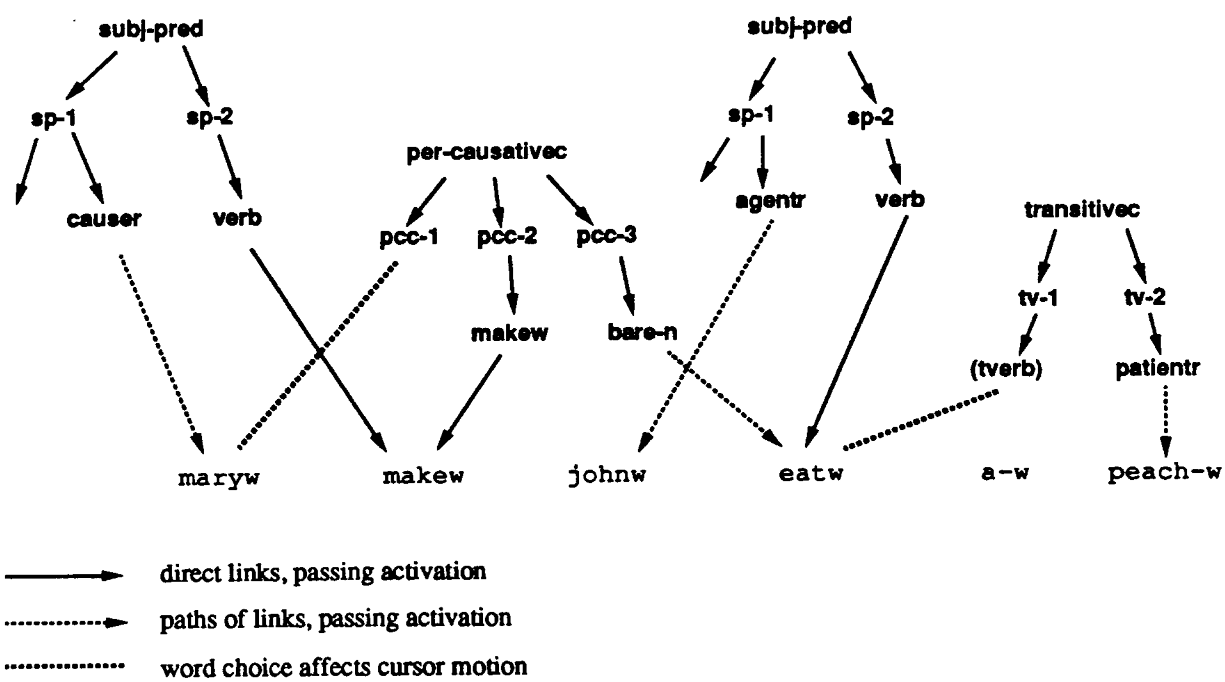


Figure 4.9: Selected Constructions Affecting Generation of *"Mary made John eat a peach"*. For of clarity **subj-pred** is drawn twice, although there is only one **subj-pred** node in FIG; reuse of constructions is discussed in §4.2.4.

"an occurring linguistic expression [can] be seen as simultaneously instantiating more than one grammatical construction at the same level" (Fillmore 1988).

Synergy also allows FIG to handle semi-frozen idioms. For example, FIG produces not only *"Mary kicked the bucket"* but also *"Mary made John kick the bucket"*, where the verb in the latter is bare. This is possible due to synergy between **kick-bucketc** and **per-causativec**, both shown in Figure 4.10. The diagram also shows that the order of *"the"* and *"bucket"* is determined in synergy with **determinatc**.

This example also illustrates a more general point: since all information is simultaneously available in FIG, there is no need to explicitly move information around. This is in contrast with most generators, which have mechanisms to propagate information around trees, to pass information when making function calls, to make copies of information, to execute routines to retrieve information when required for a syntactic decision, and so on. A good example is the treatment of tense. FIG has synergistic parallelism; this allows a construction like **per-causativec** to remain active over a period of time and affect the form of the verb in the embedded clause — there is no need for a special mechanism to transport the information that 'verb-should-be-bare' down to the point of verb choice. In FIG there is also part-wise parallelism, that is, all parts of the input are simultaneously sources of activation. For example, if the input includes the node **pastn**, this information is 'globally' available, in the form of activation sent out from this node. This information will affect the inflection of the verb chosen whenever that choice happens to occur (since such nodes are 'persistent' and never become 'expressed', as discussed in §6.4.3). There is no need for the syntax engine to invoke a procedure to retrieve the information governing tense from wherever it is present in the input, nor for the grammar to state that the tense of the verb equals the tense of the sentence.

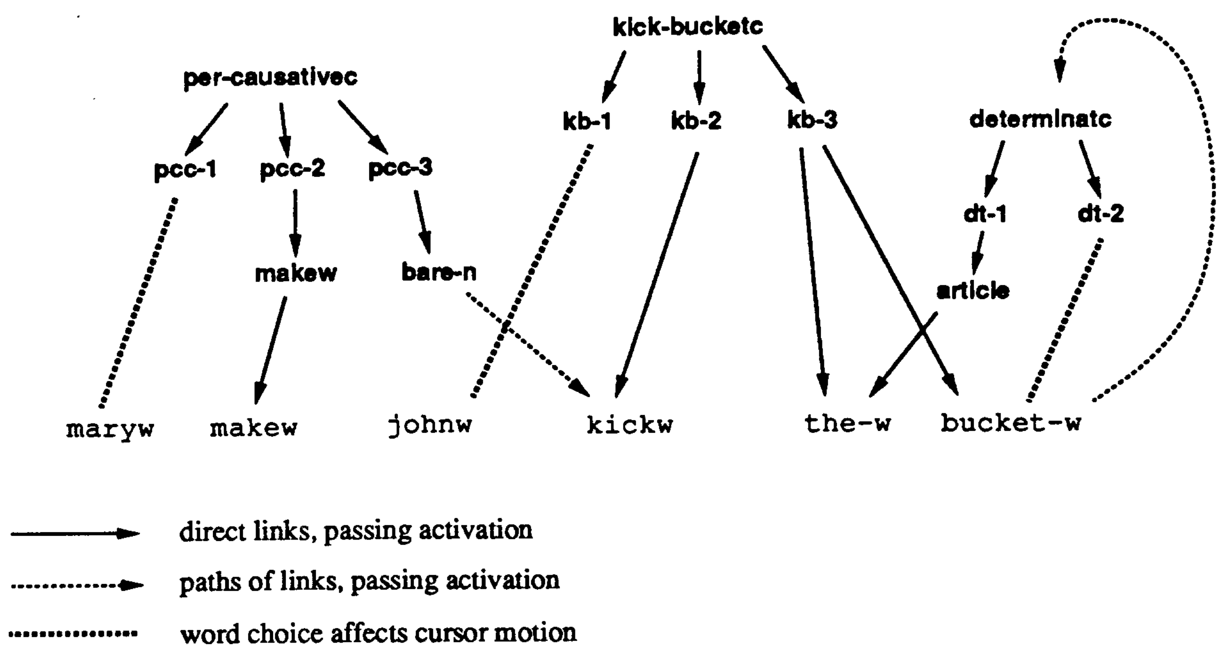


Figure 4.10: Selected Constructions Affecting Generation of *"Mary made John kick the bucket"*. The contributions of **subj-pred** are omitted here for clarity.

An advantage of using synergy is that constituents do not need to specify which constructions can instantiate them — that information is factored out. For example, the second constituent of **go-c** simply specifies **destinationr** as the inner argument of *"go"*; thanks to synergy with **prep-phr** the destination is expressed as a prepositional phrase, as shown in Figure 4.11. Similarly, the first constituent of **genitivec** specifies simply **noun**. By independent principles (namely, the existence of **determinatc**, **adj-modc**, etc) the possessor is realized as a noun phrase which is as complex as is needed, for example, as in *"a big old woman 's peach"*.

More generally, synergy allows information to be factored out into separate constructions. Figure 4.12 shows a fragment of FIG's knowledge network, drawn to suggest the way constructions affect word choice during the production of a noun phrase. This shows that there is no 'noun-phrase' construction which combines knowledge about determiners, intensifiers, and adjectives and count-nouns; instead the knowledge is expressed in separate constructions. For example, the knowledge about the adjective option is factored out into **adj-modc**. At run-time the information encoded in these constructions jointly determines the output. The way that constructions 'overlap' and the correct word order results is determined by link weights, as discussed in §4.2.2.

Using synergy among constructions allows each individual construction to be simpler, and to have its own semantic motivation, as called for by functional approaches to grammar. In FIG this means that each construction can have its own specific source of semantic activation. For example, consider the construction **dir-particlec**, which encodes the information that it is possible to use a verbal particle to express direction, as in *"go* **away***"* or *"go* **back** *home"* or *"go* **down** *to the lake"*. This construction is linked to the node **motionn**, and so whenever the input involves a motion, this construction will be activated and able to cause the interpolation of a verbal particle of direction after the verb.

The use of synergy also allows a grammar where constructions are suggested largely 'bottom-

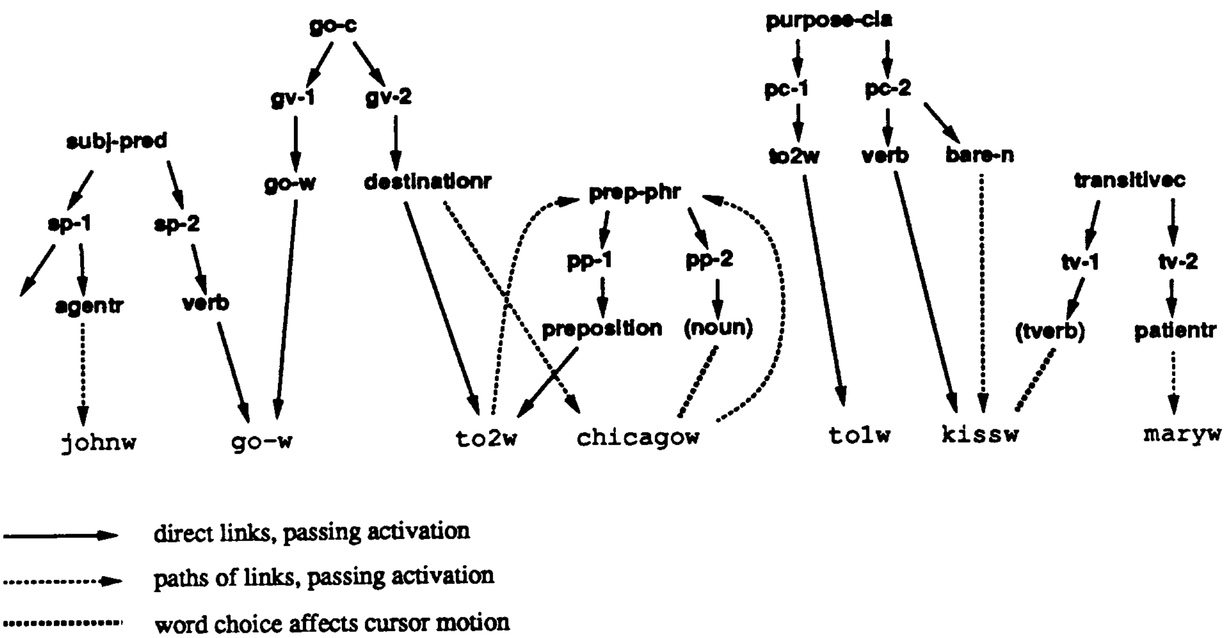


Figure 4.11: Selected Constructions Affecting Generation of *"John went to Chicago to kiss Mary"*

up'. For example, in FIG determiners are treated as being motivated by the presence of a count noun, not by the need to complete a maximal NP so as to satisfy the requirements of some superior construction. To be specific, there is a path from all count nouns via **cnoun-ness** to **determinatc**. (Evidence for and against this grammatical analysis is discussed in Chapter 5.)

The idea of factoring out information into separate constructions and having the generator combine that information when generating a sentence is not unique to FIG. For example, unification is a common way to combine information from different sources. The advantages of using synergistic parallelism to combine information are that this is well suited to incremental generation, and that it avoids the need to form binding lists or build up structures. These points are discussed further in §4.5.1.

Synergy requires that constructions operate 'in sync', that is, they must be organized with respect to each other. This is in part the responsibility of the syntactic update process, discussed in §4.2.2, and in part due to the weights on links. For example the verbal particle precedes the purpose clause, as seen in Figure 4.11, simply because of activation from **gv-2** to **destinationr**; *"once upon a time"* appears at the beginning of sentences simply because the weights to and from **time-settngc** are relatively high; and the inner arguments of the verb precede adjuncts simply because the weights are set so that the former get more activation, in particular the weights to and from **transitivec** are relatively high.

In general any construction can cause the interpolation of words 'in the midst of' any other construction. Of course this synergy is sometimes not desired. To prevent this a constituent of some construction can activate some nodes so strongly that it overwhelms the effects of all other constructions. This is the case for the second two constituents of **kick-bucketc**, responsible for the words *"the bucket"*. In effect, this construction temporarily takes exclusive control, suspending synergistic parallelism.

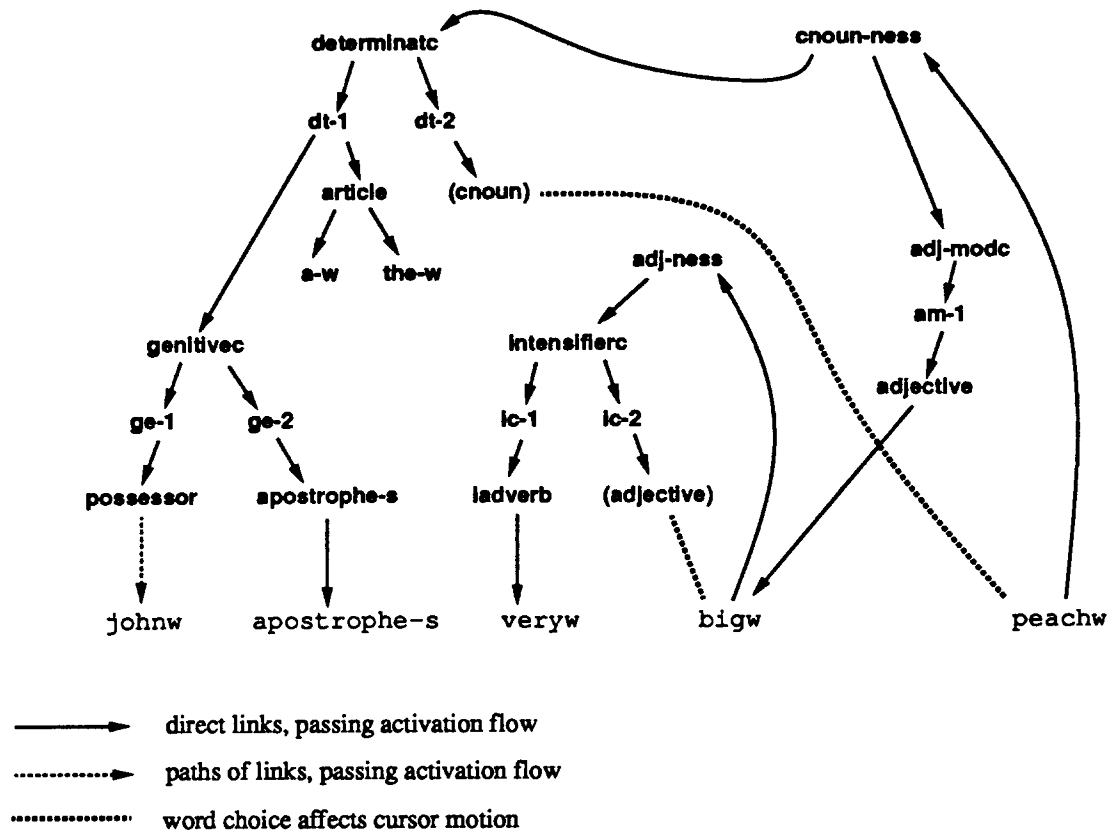


Figure 4.12: Selected Constructions Affecting the Generation of *"John's very big peach"*

## 4.5 Comments and Comparisons

### 4.5.1 Doing Without Structure-Building

FIG does not build up representations of syntactic structure; there are no mechanisms to unify constructions or organize them into a syntax tree. Instead the network representations of linguistic knowledge affect word choice and order directly. Most generators build syntactic structures in the process of generation. This section explains the reasons for this and explains why these reasons do not apply to FIG.

Structure-building goes hand-in-hand with explicit decisions — if a generator makes explicit choices it must save the results of these decisions somewhere. For example, if a generator chooses the concepts to express in various syntactic slots it must record that information, perhaps by binding variables; if a generator first chooses the words to say and then chooses their order, it requires some intermediate buffer in which to save the chosen but yet-unordered words; and if it assigns word order before actually emitting the words, it needs to a assemble a representation of the sequence of words. FIG, by avoiding explicit choice except for words (§4.4.1), avoids the need for such storage.

It is often assumed that building up syntactic structures is necessary for psychological reasons. This is an instance of the more general claim that for an organism to exhibit serial behavior it must have a representation of that behavior (for example, some kind of plan) (Lashley 1951). This is the standard technique for ensuring that actions are performed in the right order, especially if there is a need to produce more than just canned action sequences, that is, if the system must exhibit productivity. The necessity of assembling representations of behavior has recently been questioned within the AI community. Also, as discussed in Chapter 8, there is no psychological evidence for the existence of structure-building in speech and some evidence for the opposite: emergent structure.

A low-level, technical reason for structure-building is that syntax trees can be exploited for feature propagation, that is, information can be passed up and down across the links forming the tree. Such a mechanism is unnecessary if there is part-wise and synergistic parallelism (§4.4.2).

Another reason for structure building lies in the fact that generation research is largely parasitic on linguistic theories, even though the mechanisms used in theories of syntax are not, in general, intended for use in language processing. Linguists are concerned with explaining sentence structure, being as they are prisoners of their structuralist heritage. In particular, syntactic structures are often used for capturing generalizations and for accounting for grammaticality. These two issues are discussed in §4.5.2 and §4.5.4.

The idea of dispensing with intermediate representations has proven useful in linguistics. For example, Construction Grammar, among other modern theories of grammar, does without deep structure, that is, it has no intermediate level of structure between the meaning and the representation of surface structure. FIG goes one step further — it does not even represent surface structure.

The general acceptance of structure-building is perhaps a reason for the continued prevalence of top-down approaches to generation. It is relatively easy to build syntax trees top-down, since there is only one growth point at any one time. Recently several researchers have experimented with more bottom-up, lexically-driven approaches, where words suggest syntactic fragments and these fragments are then assembled into larger syntactic structures (Kempen & Hoenkamp 1987; De Smedt 1990; Finkler & Neumann 1989). There has been apparently little success in this venture, judging by the paucity of outputs shown in the descriptions of such programs. My analysis of the problem is that it is hard to assemble trees piece-by-piece from the bottom-up. FIG also relies heavily on parallel bottom-up activation of many constructions, but this has proved workable because FIG dispenses with the bother of structure-building.

### 4.5.2 Capturing Generalizations

Generators using canned text or long, domain-specific patterns are inflexible. Thus there is a practical reason to capture generalizations, or, equivalently, to factor out information, (§2.2.1), This section focuses on the goal of capturing generalizations as pursued in structuralist approaches of syntax.

The goal of capturing generalizations has been one reason to postulate 'syntactic structures' purportedly underlying sentences. For example, an active and a passive sentence might be considered to share some properties at some 'phase' of derivation in transformation grammar. Generators based on such notions recapitulate generalizations at run-time every time they produce a sentence.

More recent theories of grammar do not account for generalizations in this way. Instead, they account for generalizations in terms of the structure of the grammar — that is, the relationships among constructions (see for example (Lakoff 1987; Jurafsky 1988; Fillmore 1989)). Inheritance is a common tool for capturing such generalizations (Jacobs 1985a). An alternative to inheritance is to capture generalizations implicitly in the structure of connections in a PDP network; this appears especially useful for capturing sub-regularities. FIG, since it has very limited inheritance and is a structured connectionist system, simply fails to capture many generalizations.

One way that FIG does capture syntactic generalizations is by relegating them to semantics. Concretely, this means it relies on activation flow via general (abstract) concept nodes, for example, any concept which has a link to **motionn** will have access to the **dir-particlec** construction.

The second way FIG captures generalizations is by factoring them out into separate constructions which are used together at run-time, as explained in §4.4.2. Synergy may obviate the need for some inheritance, including multiple inheritance. For example, post-nominal modification, as in *"the peach from the stream"* occurs thanks to synergy between **prep-phr** and **determinatc** and so on; there is no need for a separate *'Postmodified-np*'* construction which inherits information from *'NP*'*, as was done in KING (Jacobs 1985a). As another example, in FIG the fourth constituent of **passivec** is only linked to the word **by-w**, and the fact that the agent is realized in a prepositional phrase is thanks to synergy with **prep-phr**, whereas in KING there is a *'by-adjunct'* pattern which inherits from *'prep-phrase'*. Thus synergy allows two pieces of information to both affect the output directly; this contrasts with inheritance, where the information in one construction must first be inherited and combined with another.

### 4.5.3 Representing Syntactic Knowledge

This thesis is a study of language processing; its claims involve the types of knowledge and computation required in generation. But FIG-as-implemented necessarily also incorporates a specific grammar and a specific syntactic representation. About these I wish to make no claims, but some comments are in order. FIG's grammar is discussed in Chapter 5. This subsection discusses representation.

One who does not like my representation is invited to criticize it on the grounds that 'some syntactic facts are unreasonably hard to encode in this formalism', or 'this formalism does not support certain kinds of processing'. In fact, I have changed the representation several times myself for these reasons, and may well change it again, to handle the issues raised in §4.6, for example. To forestall misunderstanding, I stress that I do not claim that the knowledge representable with this approach includes all and only the true facts about possible human languages. I also cannot claim that this representation can be used for other tasks, such as parsing, or that it is learnable. Researchers who proclaim the use of the same syntactic representations for both parsing and generation generally view both tasks as simply questions of mapping one representation onto another. (Reasons why it is inadequate to treat generation as a mapping problem are given in §2.2.2 and §9.1.) To say anything about the use of shared knowledge for real language understanding and real generation will require a great deal more work in this framework as in any other.

I currently have no theory of how to extend FIG's syntactic coverage. It would be nice to present a set of principles for drawing links, such as 'all constructions have links from words which could be their heads'. But of course, notions such as 'head' are notoriously tricky; what counts as a head varies from grammatical theory to grammatical theory. Moreover FIG's representation is subject to the further constraint that it work for processing. Currently FIG generally does have links from heads to constructions, for example, from **cnoun-ness** to **adj-modc** and not from **adj-ness** to **adj-modc**, but I am not ready to state any general rule as to when there should be a link or path from a word to a construction it can appear in. Perhaps after more experience encoding and debugging grammatical information it will be possible to make general claims about such questions.

FIG would be a terrible grammar-writing-and-testing tool for linguists. It is not something to which one could trivially add any number of linguistic facts, which was a motivation for, say, PATR-II (Shieber 1986). This limitation is in part a constraint imposed by the implementation: in order to make FIG work it is necessary to choose link weights, a chore which is not straightforward (§6.5.3). This limitation also reflects the fact that syntax in FIG is not autonomous; it is necessary to add world knowledge and so forth in order to extend FIG, as discussed in §4.6. Moreover, it is impossible to trace through by hand and predict what FIG will do, in part because of all the parallelism, and in part because syntactic choice is emergent and there is no structure-building. For example, in FIG the 'adjective option' arises due to synergy among constructions (§4.4.2). Many generators have instead grammars where all the rules are folded together; in these systems options are explicit branch points, so one can inspect the grammar and readily determine all the sentence structures that can result; this is most clear in ATNs and systemic generators.

### 4.5.4 What About Grammaticality?

The competence hypothesis states that a theory which can account for grammaticality judgments will also be useful in models of language performance. This seems to imply that a process model should incorporate a notion of grammaticality. The FIG experiment suggests that this is unnecessary, at least for generation.

Quite apart from theoretical motivations, clearly a generator's output ideally ought to be grammatical rather than not. But it seems that the importance of grammaticality is often overemphasized. Output which is grammatically correct is not necessarily more understandable than fragmented, ungrammatical output. Thus, producing grammatical output was only one among several considerations when designing FIG.

Yet many researchers believe that a generator should be guaranteed to produce grammatical output. It would be hard to prove that FIG's output is necessarily grammatical. Even during the production of a simple sentence, many nodes have some degree of activation; parallelism is rampant.

Also, to the extent that the output is grammatical, it is due in part to the fact that the input to be expressed constitutes a reasonable thought. For example, FIG would never have a problem of failing to satisfy a verb valence, not because there are constraints and cross-checks on various structures, but simply because no reasonable input would lead to the choice of a verb with a certain valence and also to leaving out an argument required by that verb.

More generally, in FIG syntax is not autonomous. Quite the contrary, syntactic knowledge only has the intended effects thanks to interaction with the input, for example, for the sake of locality (§4.2.3). The fact that syntax is never completely in control has a practical advantage: it ensures that all the concepts will somehow be expressed, even if the representation of the syntactic state of affairs gets messed up, as can happen if the syntactic update process has a bug.

FIG has no idea whether, when it has finished expressing some information, the result is grammatical or not. One could imagine improving the syntactic update mechanism to notice if some construction's cursor is in an odd state after generation completes; working out the details of this would require some thought.

Parenthetically, there are some problems with the standard method of accounting for grammaticality, namely the idea of matching or 'unifying' constructions together. This requires a notion of external syntax, that is, for each construction, a description of "the larger syntactic contexts in which it is relevant" (Fillmore 1988). This must match the specification of the constituent it instantiates in the encompassing construction. That is, there are "principles which govern the nesting and superimposition of constructions into or upon one another". Such principles allow a model to account for grammaticality judgements; a grammatical sentence is one where all the constructions 'fit together' snugly. In FIG a construction's privileges of occurrence are nowhere represented explicitly; rather this information is implicit in the links which point to that construction. Constructions are active together, but are never compared, coordinated, matched, or bound together.

To build a system where constructions fit together also requires otherwise unmotivated constituent labels, such as *'n-bar'*, or, equivalently, features which allow the creation of complex labels, such as *'noun +max'*. These allow 'upper' constructions like Subject-Predicate to refer explicitly to 'lower' constructions like Noun-Phrase; that is, these nodes are the points where the upper and lower constructions fit together. If such constituent labels are available they can be used for other things also. For example, an *'n-bar'* node or the feature *'+max'* can be used to ensure that determiners come before adjectives. In FIG such order information is simply encoded in link weights. Labels and features can also be used to ensure uniqueness, for example, the feature *'+max'* can also be used to ensure that no noun phrase has two determiners. In FIG this type of thing is handled simply in the syntactic update process.

### 4.5.5 Constraints on the Input

Standard approaches to syntax for generation rely too narrowly on the structural characteristics of their inputs. Historically this is partly because meaning representations have been designed in the image of syntactic representations (§2.2.1).

One example of this is that generator inputs usually have a privileged top node. This is the place from which they begin traversing the input; it corresponds to the S node of the syntax tree.

Another example is the common assumption that the input is a hierarchical structure. This is convenient because it allows generators an elegant control flow: "if ... an element [of meaning] can be said directly, in one word or frozen phrase, the generator's task is easy; otherwise, the generator has to break up the element into parts and concentrate on each part, recursively" (Hovy 1988). This notion of recursive decomposition is widely used in generation research. One possible reason is the prevalent belief that the productivity of language and thought requires that meanings be 'compositional', but, as van Gelder (1990) has pointed out, this does not imply that meanings need be represented as traditional symbolic structures, hierarchical or otherwise.

Another example is the common assumption of a conveniently labeled input. The clearest example is the idea of links from concepts to modifying concepts. Such links allow the generator to follow pointers in the input, for example to find the concept related by an agent link to the current node. This is clearly convenient for head-driven processing.

Also common is the use of matching or unification of syntactic structures or word meanings to input structures. This presumes that inputs have particular structural configurations.

Relying on such details of the input representation makes a generator inflexible, as argued in §2.2.1 and Chapter 9. FIG is rather more lenient with respect to its input and consequently more robust. Concrete examples are given in §6.2.

### 4.5.6 History of Syntax in FIG

The reader may be wondering whether there aren't less radical ways to handle syntax in a parallel-processing, network-based generator. I have tried several, and can attest that they do not work so well.

When building the earliest version of FIG I tried to do without explicit syntactic knowledge. I hoped that grammatical output would emerge from a network with a very simple mechanism, namely transitional probabilities. That is, the likelihood of outputting a word of a certain syntactic type depended on the words just output; thus generation was a kind of Markov process. There was no other mechanism for representing syntactic knowledge. It was hard to make this approach work, and even harder to extend it, so, with reluctance, I considered more powerful ways to handle syntax.

My first refinement was to represent constructions explicitly in the network, with a moving cursor to gate activation to constituents. In this version constructions were explicitly selected, just like words. Only after selection could a construction activate its constituents and affect word choice. Although only one construction could be active at a time, an activated construction could temporarily delegate this privilege to a subordinate construction. This meant that there was no longer any syntactic parallelism; this version was effectively a standard top-down generator where spreading activation only served to select which word to use for each syntactic role. This version did have competition between optional constituents and those which could follow them (§4.4.1), but because there was no synergistic parallelism it was necessary for constructions to explicitly give some 'anticipatory activation' to constituents to the right of the cursor.

I then realized that constructions need not be explicitly selected, that is, that syntactic choice could be emergent. In the next version therefore constructions were active in parallel. To organize and order constructions this version allowed nodes, especially those representing constructions, to send and receive names of other nodes, not just amounts of activation. Nodes were also allowed to have internal variables, these became bound to other nodes. For example, the internal variables of constructions were bound to concepts and words for constituents. This was thus a marker-passing or object-oriented system instead of a simple spreading activation one. It was hard to make this scheme work: by binding nodes together, the system was creating links during the course of generation; it proved most difficult to tune link weights so that nodes would become activated to the correct extent both before and after these links were added.

Finally I decided to dispense with structure-building also. Doing without binding concepts to constituents meant that cross-talk became a problem; to avoid this I gave FIG the locality principle. Doing without binding constructions together means that in this, the current version, word order is entirely due to amounts of activation. As I began to extend the system more rapidly it became more important to have a clean grammar, so I began to factor out more generalizations into separate constructions and to exploit synergistic parallelism. A pleasant side-effect of factoring out information was that the network became easier to extend and debug. Exploiting synergy more also simplified the syntactic update process; for example, when there was a complex 'noun-phase' construction it was necessary to label adjectives as 'optional', and such optional constituents had to be treated differently for the purpose of syntactic update; this is not necessary in the current version. As I extended and tided the grammar the individual constructions began to look more like those familiar from Construction Grammar. All in all I consider it a pleasant surprise that an encoding of syntactic knowledge usable for processing with spreading activation can also resemble a linguistically motivated grammar.

### 4.5.7 Similar Approaches

Previous sections have contrasted FIG with very different approaches. This section compares it with fairly similar ones. It does so by discussing, for certain claims of this chapter, the extent to which they have been noted or missed by other builders of structured connectionist generators. The focus is on Gasser's CHIE (Gasser 1988), since it is by far the most powerful connectionist generator and is also the only one explained at all clearly. The aims, significance, and shortcomings of other connectionist generation research are discussed more generally in Chapter 7.

FIG's mechanism for updating the current syntactic state, the moving cursor, is not unique. Other connectionist generators have equivalent mechanisms: Kalita has 'sequencing nodes' (Kalita & Shastri 1987), Gasser has a rather complex mechanism, and Kitano has 'V-markers' (Kitano 1990). Neither is the idea of resetting constructions rather than making copies of them novel; both Kalita and Gasser have essentially the same mechanism. The idea that word order requires no mechanism other than that for word choice was stated explicitly by (Stemberger 1985a).

FIG is unique in using the maximum rule. Recall that this is necessary to avoid overactivation of constructions (§4.2.4) as a result of part-wise parallelism or competitive parallelism plus activation from words to constructions. Other researchers in connectionist generation do not report problems of overactivation, which suggests to me that their systems have no significant part-wise or competitive parallelism.

The locality principle is not unique to FIG — CHIE also uses flow of activation among the nodes of the input. It is not clear, however, whether it suffices to deal with crosstalk.

The idea of competing constructions has been proposed in the psycholinguistic literature (Stemberger 1985a; Baars 1980), but choice among constructions is resolved relatively early in these models, and in all implemented generators. Unique to FIG is the idea that construction 'choice' is entirely emergent, that is, that competition among constructions continues even during word choice. This position is perhaps too radical; it is possible that a system with many more constructions will require some mechanism to ensure that after some point only compatible constructions are significantly activated. Such a mechanism could perhaps simply consist of inhibitory links among constructions.

FIG's trick for handling optional constituents, using competition between words for an optional position and words that can follow it, is also used in CHIE.

FIG's treatment of the slot-to-case mapping as emergent is unique. To determine what concept to express in each 'syntactic role' CHIE uses a clever mechanism involving simultaneous firing of nodes, and Kitano's system includes node names in the markers it passes around. To record the results of this process Gasser has a special role binding mechanism; similarly Kalita has 'binding nodes'. When FIG used binding it led to problems (§4.5.6). Gasser and Kalita do not report problems with the use of binding; I surmise that this is because their systems only had to produce a few outputs.

For Kitano syntactic knowledge is in large, very specific templates — that is, generalizations are not captured. CHIE has more general constructions, but they appear to be active one-at-a-time, that is, there is no synergistic parallelism

FIG is unique in having emergent syntactic choice; even CHIE has explicit syntactic choice, in the form of firings of nodes for constructions.

## 4.6 What Remains to be Done?

There is one grammatical phenomenon which is handled by most generators but not by FIG: agreement. To implement this would probably require the definition of a new class of nodes, with a slower decay rate than other nodes. This would, in effect, allow FIG to remember the person and number of the subject and allow that to affect the verb. I have not yet implemented such a scheme. This is not because I anticipate that it will be difficult, but because I am not sure of exactly what the desired behavior is. That is, it not clear to what extent agreement is a purely syntactic phenomenon or is based on meaning; nor is it clear which other syntactic phenomena also require slow-decay nodes. I prefer to keep FIG as simple as possible until I understand exactly how it should be extended.

This remainder of this section suggests the enormity of other syntactic phenomena which remain to be addressed. Of course FIG is not unique in handling none of these. The focus here is on things which probably can not be handled simply by adding more constructions; all may require major new inference abilities. This may be possible with larger network structures, although I do not as yet have any proposals. The discussion is organized in terms of the new functionality that FIG needs; for each it lists the syntactic phenomena that this would FIG to exhibit.

FIG requires detailed semantics for various domains, such as: scales, aspect, modality, causation, questions, reflexives, negative polarity, quantity, intrinsicness (for ordering adjectives), surface case, and quantification. It would be possible to simply add constructions for these to FIG, but this would be rather pointless until the semantics are understood. This will require more basic linguistic research, and also more research into the nature of the input to the generator and into the inferences appropriate for various semantic domains. In particular, FIG needs richer notions of the semantics of actions; this should enable it to to predict the valences (case frames) of verbs, including definite null complements and so on, instead of requiring each verb to have links to its own valences, as is done presently.

FIG needs some notion of a distinction between given and new information, and between background and foreground information. This would enable it to produce anaphor and referring expressions more generally, and to handle stress, end-focus, and deliberate ordering. When building an input for FIG there is currently no way to represent which node is in focus other than to give it a higher activation level. This suffices for the fronting of focused information but will probably not suffice in general.

FIG needs some way of marking a node as being of special interest in order to handle anaphor, relative clauses, distant instantiation, and perhaps agreement. Regarding relative clauses, the difficulty in handling them will probably be due not to the syntactic form but to the underlying meaning and function. That is, relative clauses are usually used either for the sake of smooth discourse or for the sake of specifying a referent, and neither notion is yet present in FIG.

FIG may need some notion of scope to handle negative polarity.

FIG needs knowledge about and mechanisms for prosody, including intonation. This will also be necessary to deal with end-weight (for example for heavy-NP movement).

FIG needs spatial reasoning ability and some notion of viewpoint for the sake of consistency so it does not not produce, for example, *"the woman arrived at the stream and saw a peach going towards her"*.

FIG needs very abstract patterns, in order to handle metonymies. For example, in English states are best expressed by describing them, rather than by using the cause or the onset metonymically (Talmy 1980), thus *"he* **is standing**" is generally better than *"he* **has** *stood up"*. Similarly, complex actions are best expressed by mentioning the onset, thus *"let's* **go** *to a restaurant"* is better than *"let's* **eat** *at a restaurant"*, if the context is 'what shall we do now?' In FIG, this could perhaps be handled by having a node such as **express-scriptal-activity**. Such a node would receive activation from the need to express, for example a scriptal activity, and would in turn activate one part of it, for example the onset.

Each of these could be handled many ways in a spreading activation system; it will require thought to decide which way is best. I expect to handle each with a mixture of language-specific functionality and more general inference-type functionality.

# Chapter 5

# FIG's Grammars



The purpose of this chapter is to present and explain the syntactic knowledge currently present in FIG. It is provided primarily in case the reader is baffled or suspicious about the production of some examples. This chapter does not raise any issues that have not been discussed in previous sections, nor does it provide analyses of English or Japanese with much validity beyond being workable in FIG.

The organization of this chapter is simple: it consists of descriptions of the constructions of FIG, with illustrations of outputs they affect, and comments on some cases where FIG fails.

## 5.1 Some Details of Syntax in FIG

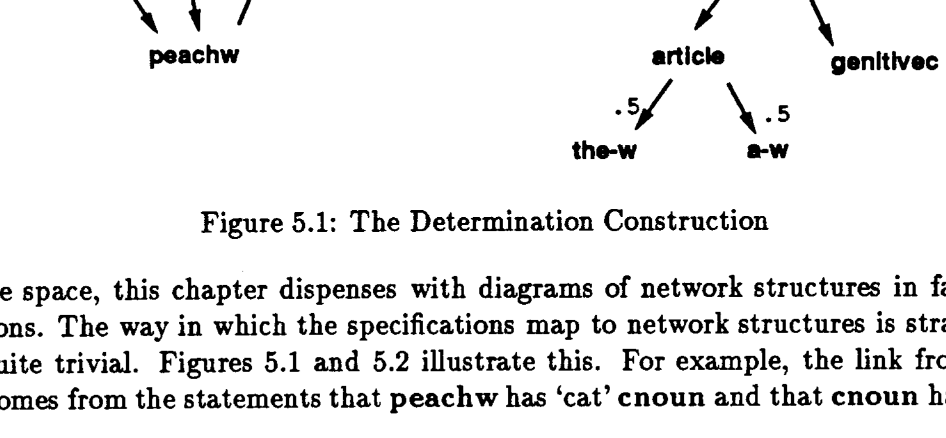


Figure 5.1: The Determination Construction

To save space, this chapter dispenses with diagrams of network structures in favor of their specifications. The way in which the specifications map to network structures is straightforward but not quite trivial. Figures 5.1 and 5.2 illustrate this. For example, the link from **noun** to **peachw** comes from the statements that **peachw** has 'cat' **cnoun** and that **cnoun** has 'supercat'

```
(defec determinatc
  (constituents (dt-1  (or article apostrophe-s)  ((article .57)(genitivec .15)))
                (dt-2  cnoun  )))

(defew peachw    (cat cnoun conso-inits)(expresses momon)  (grapheme "peach"))
(defes cnoun (cat-collector cnoun-ness) (supercat noun))
(defen cnoun-ness (english (adj-modc .7) (determinatc .71)))
(defr prag-roler (weights .2 .2))
```

Figure 5.2: The Determination Construction: Specification

**noun**, and the link from **peachw** to **cnoun-ness** is inherited by **peachw** from **cnoun**. §6.1 gives the details of this mapping.

As seen in Figure 5.2, each constituent consists of three items. The first item, for example `dt-1`, is just a label. The second item, for example `(or article apostrophe-s)` is the 'trigger'; this specifies when the cursor can advance to the next constituent, as explained in §6.4.2. The third, for example `((article .57)(genitivec .15)))`, is a list of pairs: the nodes the constituent links to and the weights on those links.

```
(defx permanentx
  (english (noun 10) (verb 9) (sadverb 4))
  (japanese (jnoun 12) (jverb 9) (agentr 5)) )
```

Figure 5.3: Nodes Permanently Activated

Finally there is something not mentioned in Chapter 4: some nodes representing syntactic categories and relations are permanently activated. The primary reason is to introduce some syntactic activation into the system. Since the activation level of words depends on the product of incoming syntactic and semantic activation, and syntactic activation comes from constructions, and constructions are activated mostly by words, there is the question of where the initial syntactic activation comes from. The answer is that certain nodes representing basic syntactic possibilities always have at least some activation, for example 10 units for **noun** and 9 units for **verb**, as shown in Figure 5.3. This activation comes from the pseudo-node **permanentx**, which itself always has exactly one unit of activation. The reasons that these particular nodes are permanently activated are discussed below where relevant.

## 5.2 English

Figure 5.2 having shown the Determination Construction, it is time to justify and explain its function. FIG has argued for a bottom-up approach to determiners, as mentioned in §4.4.2. That is, determiners are licensed by the appearance of a common noun, rather than by some superordinate construction which requires a maximal noun phrase. Evidence that this is the right analysis is the fact that, even in the absence of any syntactic context, *"a snake!"* or *"the snake!"* seems more natural English than just *"snake!"*. On this view, uses of determinerless nouns, for example as allowed by the Unique-Role Nominal Predicate Construction, as in *"he was chairman of the committee"* (Fillmore 1988), are special cases. In FIG this construction would have to inhibit the **determinatc** construction (although there would ideally be a more semantic explanation, perhaps along the lines of 'the Unique-Role Nominal Predicate Construction specifies the mental space and so there is no semantic motivation for tagging the noun as known or not').

In FIG even articles must receive some semantic activation, since the activation level of a word is the product of syntactic and semantic activation, as discussed in §3.2. FIG's inputs are too

```
(defn thingn (cat-collector thing-ness))
(defn thing-ness (nebors (definitenessr a-w .41) (definitenessr the-w .37)))

(defn topicn   (nebors (prerequisiter in-contextn .9))  (japanese (wa-w .7)) )
(defn in-contextn   (english (the-w .5)))
```

Figure 5.4: Knowledge Relating to Articles

impoverished to provide this activation, since Japanese has no articles and my Japanese analyzer has little inference ability. To compensate there is a somewhat ad hoc mechanism to provide semantic activation to articles. The idea is that, whenever you think of a physical thing, things like its plexity and definiteness are immediately salient. This idea is implemented by creating a **cat** link from all nodes representing 'objects' to **thingn**, which gives them an inherited link to **thing-ness**, which is in turn linked to **a-w** and **the-w**. This stretches the semantics of **cat**, which was originally intended to connect words to syntactic categories, but it gives the desired properties; in particular, like all inherited links, the links from notions to **thing-ness** are subject to the maximum rule (§4.2.4). This is important, otherwise the activation level of **thing-ness** would depend on the number of noun-realizable notions in the input, risking premature emission of, for example, *"a"*, for complex inputs. Activation of articles via **thing-ness** is not a wonderful solution; tuning the weights of the links involved has been a recurring problem in making FIG robust. This is a case where having richer inputs would make FIG's task easier.

The choice between *"a"* and *"the"* is determined as follows. There is a link from **in-context** to **the-w**, representing that *"the"* is appropriate for concepts which are in the context. Since FIG's inputs come from an analyzer of Japanese, sometimes the node **topicn** is part of the input, linked to the appropriate notion, via a **prag-roler** (pragmatic role) relation. **topicn** is linked to **in-contextn**, representing that whatever concept is the topic must be in the shared context of discourse. Thus activation will flow from **topicn** via **in-contextn** to **the-w**, which will become activated at the right time to go with the right noun thanks to the locality principle.

```
(defec adj-modc
  (constituents (am-1  adjective ((adjective .55))) ))

(cnoun-ness (english (adj-modc .7) (determinatc .71)))
(defr sizer     (weights (ratio 1.) nil))
(defr modifier  (weights (ratio 1.) nil))
```

Figure 5.5: The Adjectival Modification Construction

Figure 5.5 shows the Adjectival Modification Construction. This receives activation from **cnoun-ness** and sends it to **adjective**. As a result adjectives receive more syntactic activation than nouns, and are emitted first. Once an adjective is output the cursor advances and, since there is only one constituent, it immediately resets. Thus this construction allows any number of adjectives before the noun (§4.2.4). This construction currently does not receive activation from its semantics, namely the need to express a **sizer** or other **modifier** relation, although it could.

Figure 5.6 shows FIG's representation of the Prepositional Phrase Construction. This construction is activated whenever a noun is, which means that nouns are in prepositional phrases as a default. This requires, unfortunately, **subj-pred** and **transitivec** to inhibit **prep-phr** to prevent case marking prepositions on the subject and direct object, otherwise FIG might produce *"by john kicked mary"*.

**prep-phr** also receives activation from **prep-ness**, which means every time a preposition is activated this construction will be. The typical reason that a preposition becomes activated is due

```
(defec prep-phr
  (constituents (pp-1  preposition ((preposition .6)))
                (pp-2  noun  )))

(defes preposition (cat-collector prep-ness))
(defen prep-ness (english (prep-phr .2)))

(defes noun  (cat-collector noun-ness))
(defen noun-ness (english (prep-phr .65)))

(defr instrumentr (weights .18 .18) (rel-collector instr-ness))
(defn instr-ness  (english (subj-pred .3) (withw .4))  (japanese (de-w .5)))

(defr originr   (weights .0 .1) (rel-collector orig-ness))
(defn orig-ness (english  (fromw .35))          (japanese (karaw .45)) )
```

Figure 5.6: The Prepositional Phrase Construction

to activation from a relation, for example Figure 5.6 shows that **orig-ness**, the 'collector' node associated with the **originr** relation, is linked to **fromw**. There is a potentially dangerous cycle here, from **prep-ness** to **prep-phr** to **pp-1** to **preposition** to **fromw** (and all other prepositions) and back to **prep-ness**. Careful tuning of weights was required to prevent activation levels from increasing without bound.

The weights to and from **prep-phr** are relatively high so that prepositions precede adjectives, articles, possessives, and intensifiers. This makes prepositions highly activated, so much so that FIG currently cannot generate *"a peach from a stream"* — here **prep-phr** activates **fromw** so highly that it appears even before the first noun. This is another problem of crosstalk — it arises because the current implementation of the locality principle is too weak (§6.5.2). That is, **fromw** should receive almost no semantic activation until just before the origin, the stream, is to be expressed, but in fact it receives a fair bit of activation.

The fact that FIG produces any post-nominal prepositional phrases correctly is due to the fact that **transitivec** inhibits prepositions; this allows, for example, *"peach"* to appear before *"from"* in *"John ate a peach from a stream"*.

The fact that prepositional phrases come after the inner arguments of a verb is simply due to weights.

The fact that *"peach"* precedes *"stream"* in *"a peach from a stream"* is not due to syntactic knowledge at all. Rather it results from activation flow among the nodes of the input. The nodes underlying this phrase will consist of **momon1** (peach) and **kawan1** (stream) connected by an **originr** link. The weights of **originr** are asymmetric so that the object is always more highly activated than its origin, thus *"peach"* will be emitted first. In general the weights on links among nodes of the input play an important role in activating concepts to the right degree, in synergy with syntactic considerations, of course. The precise mechanism for activation flow among nodes of the input is discussed in §6.2.

The Prepositional Phrase Construction handles verbal prepositional phrases, post-nominal prepositional phrases, and by-passives:

5.1 *John went to a hill to eat a peach from a stream*

5.2 *John ate a peach with a fork from Chicago*

5.3 *John ate a peach from a stream in Chicago*

5.4 *John lived in Chicago*

5.5 *John was killed by Mary*

```
(defec intensifyc
  (constituents (in-1  iadverb  ((iadverb .48)))
                (in-2  adjective  )))

(defes adjective (cat-collector adj-ness))
(defen adj-ness (english (intensifyc .56)))

(defr degreer   (weights (ratio 1.) nil) (rel-collector degr-ness))
(defn degr-ness (english (intensifyc .25)))
```

Figure 5.7: The Intensification Construction

Figure 5.7 shows the Intensification Construction, responsible for putting degree adverbs, such as *"very"*, in front of adjectives. This construction receives syntactic activation from all adjectives (via **adj-ness**), and semantic activation from semantically appropriate modification, via **degr-ness**, the 'collector' node for concepts which fill the relation **degreer**. (Collector nodes are discussed in §6.1 and §6.2.)

5.6 *a very big boy came*

5.7 *John ate a very big peach*

5.8 *John ate a cake with a very big fork*

```
(defec genitivec
  (constituents (ge-1   possessor     ((noun .15)))
                (ge-2   apostrophe-s  ((apostrophe-s .5))) ) )

(defr possessor (weights (ratio 1.04) nil) (rel-collector posse-ness))
(defn posse-ness(english (genitivec .4)(apostrophe-s .5))
                (japanese (jgenitivec .5) (no-w .5)))
```

Figure 5.8: The Genitive Construction

Figure 5.8 shows the Genitive Construction, which produces a possessor and an apostrophe-s before a noun, for example *"John 's peach"*. This construction receives activation from the **possessor** relation and also from **determinatc**, as seen in Figure 5.2 and diagramed in Figures 4.3 and 4.12. The first constituent, **ge-1**, could also be linked to **possessor** but is not, since the possessor will be activated anyway. Note that the possessor can be complex, as in *"a big boy 's peach"*, and so can the possessee, as in *"John 's very big peach"*.

5.9 *John ate Mary 's peach*

5.10 *John ate Mary 's big peach*

5.11 *a boy 's peach ate John*

5.12 *the strong boy ate John 's very big peach*

```
(defec noun-conjc
  (constituents (ac-1 first-memr  ((first-memr .2)))
                (ac-2 andw        ((andw .6)))
                (ac-3 second-memr ((second-memr .5))) ))

(defn setn
  (english (noun-conjc .9) (andw .4))
  (japanese (jnoun-conjc .9) (jto2w .5)) )
```

Figure 5.9: The Noun Conjunction Construction

Figure 5.9 shows the Noun Conjunction Construction. This handles two nominal conjuncts; other uses of conjunction, such as verbal conjunction and the comma-list use of *"and"* for more than two conjuncts are considered different constructions and are not dealt with. If the input involves a set, **setn** activates **noun-conjc**, which will activate the first member of the set, the word *"and"*, and second member of the set, in that order. **andw** gets activation from **setn** both directly and indirectly via **noun-conjc**. Of course this construction need not play a causal role in the expression of the first member of the set; if a notion which is the first member of a set gets expressed, for whatever reason, the cursor of **noun-conjc** will advance, FIG will tend to output *"and"* next.

Interestingly, this construction also suffices for conjoined prepositional phrases, such as *"John went to the hill and to Chicago"*. The second constituent is realized as a prepositional phrase, rather than just a noun, since after *"and"* **prep-phr** gets activation at this point both from **chicagow** via **noun-ness** and from **shikagon** (Chicago) via **dest-ness** (since it is the destination), **to1w**, and **prep-ness**. On the other hand, FIG cannot produce *"to a stream and a river"*, because the semantic update function does not currently cover relations (§6.4.3).

Having completed the description of noun phrases in FIG's grammar I now turn to constructions involving verbs.

```
(defec subj-pred
  (reset-if scomp-verb)
  (constituents (sp-1 (and noun
                           (or (never first-memr) second-memr) (not possessor))
                      ((causer .35)(agentr .25)(instrumentr .15)(preposition -.5)))
                (sp-2 verb  ((verb .22)))
                (sp-3 sleep) ) )

(defr agentr    (weights .25 .15) (rel-collector agent-ness))
(defn agent-ness(english (subj-pred .4) (by-w .4)) (japanese (ga-w .4)) )

(defes verb  (cat-collector verb-ness))
(defen verb-ness    (english (subj-pred .1) (purpose-cla .1)))
```

Figure 5.10: The Subject-Predicate Construction

Figure 5.10 shows the Subject-Predicate Construction. This construction receives activation from **causer**, **agentr**, **experiencer**, **instrumentr** and **patientr**, via their collector nodes (§6.2), and from every verb, via **verb-ness**. **subj-pred** passes on this activation to notions which should precede verbs.

The mini-agency hierarchy in **sp-1**, the first constituent, was discussed in §4.4.1. In English the subject receives no case marker, hence **sp-1** inhibits **preposition**. In some grammars **sp-1** would call for a noun phrase or at least a noun. Not so here; in accordance with Fillmore's theory, where the syntactic form of the subject is not determined by this construction (Fillmore 1988), in FIG, **noun** is independently activated by other considerations, namely activation from **permanentx**.

The trigger of **sp-1** is complex, for reasons discussed in §6.4.2. The intent is simply that **subj-pred** should proceed to activate a verb as soon as the subject has been completed; this is handled with `(and noun (or (never first-memr) second-memr) (not possessor))` which means that the cursor can advance if a noun has been output which expresses neither a possessor nor the first member of a set.

**subj-pred** must not reset immediately after it completes, unlike such constructions as **determinatc**, since it is not the case that a verb should appear after every noun; otherwise FIG might produce *"John ate a peach was eaten"*. To prevent such things is the purpose of the third constituent. Its trigger is 'sleep' which matches nothing, thus after the verb this construction enters a state in which it activates nothing, since **sp-3** is linked to nothing. **subj-pred** awakens only if the 'reset-if' trigger is matched, which happens for verbs with sentential complements. In this case it resets, as illustrated in §4.4.2. This allows **subj-pred** to participate in the production of embedded clauses after verbs allowing sentential complements, such as *"see"* and *"make"*.

```
(defec ex-there
  (constituents (et-1  therew ((therew .5)))
                (et-2  verb   ((verb .5)))
                (et-3  noun      ) ))

(defn introductoryn (persistentp t) (english (ex-there 1.0))
     (japanese (jloc-settngc 1.0)) )
```

Figure 5.11: Representation of the Existential There Construction

Figures 4.1 and 5.11 show the Existential There Construction. This receives activation from **introductoryn**, the node representing that the utterance will serve to introduce some character into the discourse (Lakoff 1987).

5.13 *once upon a time there lived an old man and an old woman*

```
(defec per-causativec
  (constituents (pcc-1 causer )
                (pcc-2 makew  ((makew .7)))
                (pcc-3 verb ((bare-n 1.6))) ))

(defr causer    (weights .6 .2) (rel-collector causr-ness))
(defn causr-ness(english (makew .3) (per-causativec .3)) (japanese (ga-w .6)))
```

Figure 5.12: The Periphrastic Causative Construction

Figure 5.12 shows the Periphrastic Causative Construction. As discussed in §4.4.2 the third constituent, although simple, in synergy with **subj-pred** results in the production of an entire clause with the verb in the correct form.

Figure 5.13 shows the Lexical Causative Construction. This construction says that verbs like *"break"* are appropriate after a causer, unlike verbs like *"vanish"*. Another way to get the same effect would be to say that *"vanish"* only has an intransitive form, from which the generator could conclude that a periphrastic causative is necessary, but the current treatment, where the periphastic causative is the default, is more direct. This construction is a rival to the Periphrastic

```
(defec lex-causativec
  (constituents (lcc-1  (or causer agentr)  )
                (lcc-2  verb ((verb .2))) ) )

(defes lc-verb (cat-collector lc-verb-ness) (supercat verb))
(defen lc-verb-ness (english (lex-causativec .4)))
```

Figure 5.13: The Lexical Causative Construction

Causative Construction, as discussed in §4.4.1. The patient of this construction is handled by synergy with **transitivec**, which necessarily becomes activated since, all lexical causative verbs activate **transitivec**, since **lc-verb** has supercat **tverb**.

```
(defec passivec
  (constituents (pa-1 (and patientr (never verb))  )
                (pa-2 is2w    ((is2w .7)))
                (pa-3 verb    ((past-partn 1.0)(verb .5)))
                (pa-4 by-w    ((by-w .5))) ))

(defr patientr  (weights .1 .1)  (rel-collector pat-ness))
(defn pat-ness  (english (is2w .3) (passivec .3) (subj-pred .3))
                (japanese (o-w .5)(niiw .3)(ga-w .3)) )
```

Figure 5.14: The Passive Construction

Figure 5.14 shows the Passive Construction. This construction gets triggered and has an effect if a patient appears before a verb. This will happen if there is no notion realizable as a noun other than the patient, or if the patient is overwhelmingly more activated. The complex trigger prevents this construction from becoming activated if the subject was an agent; for example, **passivec** should not become activated after *"peach"* in *"John ate the peach in Chicago"*.

This construction operates in synergy with **subj-pred**; **is2w** being a **verb** it receives activation from **sp-2**.

There is also synergy with **prep-phr** for the production of by-passives, such as *"John was killed by Mary"*. *"By"* will receive syntactic activation from the noun for the agent via **noun-ness** and **prep-phr** and semantic activation from the agent via **agent-ness**. If no agent is present *"by"* will not be emitted since it will have neither of these sources of activation.

5.14 *Mary was killed*

5.15 *Mary was killed by John*

5.16 *a peach was given to Mary*

5.17 *John 's very big peach was eaten*

5.18 *John was discouraged by criticism*

Figure 5.15 shows the State-Change Construction, in which the patient appears in subject position, as in *"the old man died"* or *"the dish broke"* (van Oosten 1985). For this construction also the sentence subject is selected by general principles (typically simply that nothing higher on the agency hierarchy is present); this construction only determines what verb to use to continue. This construction competes with **passivec**; for example, it is appropriate to say *"John was killed"* but not *"the cake was vanished"*, as discussed in §4.4.1.

```
(defec state-changec
  (constituents (csc-1  (and patientr (never verb))  )
                (csc-2  st-ch-verb  ((st-ch-verb .2))) ))

(defes st-ch-verb (cat-collector st-ch-vb-ness) (supercat verb))
(defen st-ch-vb-ness (english (state-changec .4)))
```

Figure 5.15: The State-Change Construction

```
(defec transitivec
  (constituents (tv-1 tverb )
                (tv-2 noun  ((patientr .3)(preposition -.4)))
                (tv-3 sleep) ) )

(defes tverb (cat-collector tverb-ness) (supercat verb))
(defen tverb-ness (english (transitivec .4)))
```

Figure 5.16: The Transitive Construction

Note that a verb can have any number of valences: for example, **breakw** is both a **st-ch-verb** and a **lc-verb**.

Figure 5.16 shows the Transitive Construction. This activates **patientr** so that it will appear before any adjuncts (as in *"John ate the peach in Chicago"*). It also inhibits **preposition** so that the patient is not preceded by a preposition. A better analysis is needed here; inhibitition of **preposition** prevents FIG from producing *"Mary ate with a fork"* — instead it produces *"Mary ate a fork"*.

This construction is doesn't care what the subject was, in particular, it could have been either a causer or an agent. **transitivec** receives activation from **tverb-ness**; it could receive activation from the relation **patientr**, as an alternative or in addition.

```
(defec dir-particlec
  (constituents (mvp-1  verb )
                (mvp-2  vparticle  ((directionr .6)(vparticle .6))) ))
(defn motionn
  (english (dir-particlec .5))   (japanese (mannr-motionc .5)) )
```

Figure 5.17: The Verbal Particle of Direction Construction

Figure 5.17 shows the Verbal Particle of Direction Construction; this merely says that the direction can be realized as a particle. By virtue of the relatively high weights from **mvp-2**, the particle will come directly after the verb. The relatively high weight on the link to **directionr** means that this construction allows the direction to be expressed even if the direction is receiving little semantic activation.

5.19 *a big peach bobbed down towards an old woman from upstream*

Figure 5.18, represents the Purpose Clause Construction, as discussed in §4.1. Note that the word **to2w** receives syntactic activation from this construction and semantic activation from **purp-ness**. The patient, if any, is expressed in synergy with **transitivec**.

Figure 5.19 shows the constructions responsible for sentence-initial adverbials of space and time. If a concept filling a **timer** relation is present, activation will flow via **time-ness** to **time-settngc**, which will activate time-adverbs, such as **one-dayw** or **long-agow**. Similarly for **locationr** relations — these map to words like *"somewhere"* and *"there"*. (Incidentally, this

```
(defec purpose-cla
  (constituents (pc-1  to2w  ((to2w .7)))
                (pc-2  verb  ((verb .2) (bare-n 1.0))) ))

(defr purposer (weights .15 .15)  (collector purp-ness))
(defn purp-ness (english (to2w .45) (purpose-cla .3)) (japanese (ni2w .4)))
```

Figure 5.18: Representation of the Purpose-Clause Construction

```
(defec time-settngc
  (constituents (ts-1 time-adverb ((time-adverb .7)))))

(defec loc-settngc
  (constituents (ls-1 loc-adverb ((loc-adverb .6)))))

(defr locationr (weights .2 .1) (rel-collector loc-ness))
(defn loc-ness  (english (loc-settngc .8)(in-w .35)) (japanese (ni1w .5)) )

(defr timer     (weights .5 .2) (rel-collector time-ness))
(defn time-ness (english (time-settngc .8)) (japanese (jtime-settngc 1.)) )
```

Figure 5.19: Knowledge About Sentential Adverbs

activation to *"there"* is in synergy with **et-1** of the Existential There Construction **ex-there**.) These adverbials appear sentence-initially simply because the weights of the links to and from these constructions are relatively high. The fact that time adverbials precede location adverbials in English is also encoded in the weights: .7 is greater than .6, so time adverbials will receive more activation.

5.20 *one day the old man went into the hills to gather wood*

5.21 *once upon a time there lived an old man and an old woman*

FIG also produces sentence-final adverbials, such as *"quickly"*. These are not mentioned by any construction; but receive activation from **sadverb**, which is permanently slightly activated.

5.22 *Mary came quickly*

5.23 *John gave a big peach to Mary quickly*

Here are some verb valences not yet handled by proper generalizations:

```
(defec comec
  (constituents (cc-1  comew )
                (cc-2  destinationr ((destinationr .2)))))
```

Figure 5.20: Valence of the Verb *"come"*

Figure 5.20 shows the construction used for the valence of *"come"*. Its only purpose is to cause the destination to be expressed before the source, as in *"the peach came from upstream towards the old woman"*. Rather than handling this with a construction, this could be handled by saying that the weight of the destinationr slot of **kurun** (come) is strong, if FIG were extended to have a concept of frames.

```
(defec go-c
  (constituents (gv-1  go-w       ((go-w .2)))
                (gv-2  noun       ((destinationr .25))) ))
```

Figure 5.21: Valence of the Verb *"go"*

Figure 5.21 shows the valence of *"go"*. The primary function of this construction is to activate the destination enough so that it will precede the purpose clause and other adjuncts, but not direction particles. A secondary function is to compensate for a weakness of FIG discussed in §6.5.1, namely, that since inputs involving purpose and destination tend to be densely interlinked, there is a danger that the verb of purpose will receive more semantic activation than the main verb. The link from **gv-1** activates **go-w** enough to overrule this.

Here are a few examples to illustrate synergy of this construction with **dir-particlec**, **purpose-cla**, and **prep-phr**.

5.24 *the old woman went*

5.25 *the old woman went to wash clothes*

5.26 *the old woman went to a stream*

5.27 *the old woman went to a stream to wash clothes*

```
(defec get-toc
  (constituents (gc-1  get-to-w  )
                (gc-2  noun      ((destinationr .1)(preposition -.5))) ) )

(defew get-to-w   (cat verb)  (expresses tsukun) (valence (get-to-c .2))
      (grapheme (inf "get to") (past "got to"))
```

Figure 5.22: Valence of the Verb *"get to"*

Figure 5.22 shows the valence of *"get to"*, which FIG treats as a verb whose spelling happens to include a space. This construction therefore serves to suppress redundant prepositions.

```
(defec kick-bucketc
  (constituents (kb-1  (and patientr (never agentr))  )
                (kb-2  kickw     ((kickw .25)) )
                (kb-3  bucketw  ((the-w 1.5)(bucketw .85)(diew -.3)(kickw -.4)))))

(defn shinun (english (diew .5)(kick-bucketc .4)(kickw .4)(the-w .14)(bucketw .2)))

(defn slangn (persistentp t)       (english (kick-bucketc .6)) )
```

Figure 5.23: The Kick-the-bucket Construction

Figure 5.23 shows how the *"kick the bucket"* idiom is encoded as a simple construction. This construction become significantly activated when both of the concepts **shinun** (die) and **slangn** are actived. Having just a single node, **slangn**, govern the choice is of course too simplistic: English has the expressions *"pass away"*, *"die"*, and *"kick the bucket"*, and these vary along more than one dimension; and Japanese has only two terms *"naku naru"*, and *"shinu"*, which cover the three in English.

Like all constructions, **kick-bucketc** gives syntactic activation to its component words. The ordering of *"the"* and *"bucket"* is governed by synergy with **determinatc**.

This treatment is not completely satisfactory. Because all words require some semantic activation to be selected, there are direct links from **shinun** (die) to **kickw**, **the-w**, and **bucketw**. This causes the problem that, since **shinun** always activates **the-w**, FIG cannot produce *"an old man died"*; it always comes out as *"the old man died"*. Other symptoms of something wrong is that it has been hard to tune the weights right, that this is one of the very few cases where inhibitory links are needed, and that **bucketw** has to be listed as expressing **shinun** (die).

A probable improvement would be to have some concept to represent the 'meaning' behind this idiom, even if the meaning is only some very semantically 'bleached notion'. This concept probably should have a special evidence-combining function, perhaps taking the product of activation from **shinun** (die) and *"slangn"*. There is unfortunately no obvious or easy way to implement this.

| Category | Comments |
|---|---|
| noun | |
| cnoun | count noun |
| mnoun | mass noun |
| pnoun | proper noun |
| verb | |
| tverb | transitive verb |
| lc-verb | lexical causative verb |
| st-ch-verb | state-change verb (intransitive) |
| scomp-verb | verb taking a sentential complement |
| adjective | |
| preposition | |
| article | |
| conjunction | |
| vparticle | verbal particle |
| iadverb | intensifying adverb, for example *"very"* |
| sadverb | sentential adverb, for example *"somewhere"* |
| time-adverb | for example *"long ago"* |
| loc-adverb | for example *"somewhere"* |
| conso-inits | consonant-initial word (see §6.4.1) |
| vowel-inits | vowel-initial word (see §6.4.1) |

Table 5.1: Syntactic Categories FIG's Subset of English. **supercat** relationships are indicated by indentation.

Table 5.1 shows FIG's categories of English. The only 'words' currently without a category are **to2w**, the *"to"* of purpose clauses, and **apostrophe-s**. **vowel-inits** and **conso-inits** are syntactic categories for convenience, as explained in §6.4.1.

## 5.3 Japanese

As discussed in Chapter 3, there is a low-level switch to choose the output language; if generating English the weights of all **japanese** links become zero, and, conversely, if generating Japanese the weights of all **english** links become zero. This works not only for links to words but also for links to constructions. The same world knowledge and input structures are used regardless of the target language.

FIG's coverage of Japanese is smaller than its coverage of English.

In Japanese nouns are marked with particles. Particles serve the functions handled in other languages by case markers and those handled by prepositions. Particles come after the noun they modify, thus they are 'postpositions'. Particles mark all nouns in an utterance, including the subject and object.

```
(defjc jnoun-partc
  (constituents (jn-1 jnoun    )
                (jn-2 jparticle ((jparticle 1.4))) ))

(defjs jnoun (cat-collector jnoun-ness))
(defjn jnoun-ness (japanese (jadj-modpc .65)(jnoun-partc .2)))

(defr agentr    (weights .25 .15) (rel-collector agent-ness))
(defn agent-ness(english (subj-pred .4) (by-w .4)) (japanese (ga-w .4)) )

(defn topicn   (nebors (prerequisiter in-contextn .9))  (japanese (wa-w .7)) )
```

Figure 5.24: Noun-Particle Construction

Figure 5.24 shows FIG's encoding of the Japanese Noun Particle Construction. After a noun is output this construction activates particles. A semantically appropriate particle is chosen due to activation flow from concepts to the relations they fill, and thence to particles which mark those relations. For example, agents are marked with the particle *"ga"*, and so there is a link from **agentr-ness** to **ga-w**, as shown. Thus, at run-time, activation will flow from an agent to **agent-ness** and thence to **ga-w**.

To make this work right the principle of locality is handled slightly differently in English and in Japanese. In English the weight from the filler of a relation to the collector for that relation (such as **agent-ness**) is 1.0 but in Japanese it is .5. The higher weight means that 'case markers' become more highly activated in English, which is appropriate since a preposition precedes the filler of the relation, whereas in Japanese a particle follows the noun whose 'case' it marks. Conversely, after the noun concept is expressed, the collector gets 12 units of activation in the English case, but 18 units in the Japanese case; this helps ensure that the correct particle appears after a noun is output. If FIG didn't handle Japanese differently in these ways the particle would reflect the case of the upcoming noun, instead of the previous one, which would be wrong. This is the only place where FIG treats the two languages differently. These things are explained further in Chapter 6.

Japanese also has the particle *"wa"*. The noun which is the topic takes this particle; this rule overrides the default particle choice (in FIG's subset of Japanese).

```
(defjc jadj-modc
  (constituents (ja-1 jadj   ((jadj .63))) ))
```

Figure 5.25: Japanese Adjectival Modification Construction

Figure 5.25 shows the Japanese Adjectival Modification Construction, responsible for making adjectives precede the noun.

*"no"* is the possessive marker in Japanese. It joins two nouns, just like **apostrophe-s**. Figure 5.26 shows the Japanese Genitive Construction. Although not a standard case marker, **no-w** is listed as a particle since it appears in the same syntactic position, namely, after a noun.

```
(defjc jgenitivec
  (constituents (jg-1 possessor ((jnoun .1)))
                (jg-2 no-w  ((no-w .5))) ))

(defjw no-w  (cat jparticle) (grapheme "no"))
```

Figure 5.26: The Japanese Genitive Construction

5.28 *jon wa ookii shoonen no momo o tabemashita*
*John TOPIC big boy GEN peach PATIENT eat-PAST*
*John ate a big boy's peach*

5.29 *shoonen no momo ga jon o tabemashita*
*boy GEN peach SUBJ John PATIENT eat-PAST*
*a boy's peach ate John*

5.30 *jon wa meeri no ookii momo o tabemashita*
*John TOPIC Mary GEN big peach PATIENT eat-PAST*
*John ate Mary's big peach*

5.31 *ookii shoonen wa fooku de meeri no momo o shikago de tabemashita*
*big boy TOPIC fork INSTRUMENT Mary GEN peach PATIENT Chicago LOC eat-PAST*
*a big boy ate Mary's peach with a fork in Chicago*

```
(defjc jnoun-conjc
  (constituents (tc-1 first-memr  ((first-memr .2)))
                (tc-2 jto2w       ((jto2w .6)))
                (tc-3 second-memr ((second-memr .3))) ))
```

Figure 5.27: The Japanese Noun-Conjunction Construction

Figure 5.27 shows the Japanese Noun-Conjunction Construction. This is exactly like noun conjunction in English, except that the conjunction, *"to"* is treated as a particle. This prevents outputs like *"yama e to kawa e"* ( *"hill to and stream to"*) (where both nouns have case markers) which are ungrammatical in Japanese.

5.32 *fooku to ookii momo*
*fork CONJ big peach*
*a fork and a big peach*

5.33 *yama to kawa e*
*hill CONJ stream DEST*
*to a hill and a stream*

Figure 5.28 shows that the default particle for **patientr** is **o-w**. As mentioned in §4.4.1 the verb can affect particle choice, overriding the defaults which are given by semantics. For example, the verb *"kissu"* marks its object with *"ni"*. Thus the valence of **kissuw** is **patient-ni**. This construction says that, once the patient is output, output *"ni"*. Thus FIG produces:

5.34 *ringo o tabeta*

```
(defjc patient-ni
  (constituents (pn-1 patientr     )
                (pn-2 nilw ((nilw .6))) ) )
(defjc dest-ni
  (constituents (dn-1 destinationr     )
                (dn-2 nilw ((nilw .6))) ) )
(defjc loc-ni
  (constituents (ln-1 locationr     )
(ln-2 nilw ((nilw .8))) ) )

(defr patientr  (weights .1 .1)  (rel-collector pat-ness))
(defn pat-ness  (english (is2w .3) (passivec .3) (subj-pred .3))
                (japanese (o-w .5)(nilw .3)(ga-w .3)) )

(defjw kissuw    (cat jverb) (expresses kissun) (valence (patient-ni .2))
                (grapheme (pres "kissu shimasu") (past "kissu shimashita")) )
```

Figure 5.28: Verb Valences

```
(defjc jintransitivec
  (constituents (kc-1 (and patientr (never agentr))  )
                (kc-2 ga-w       ((ga-w .5)))
                (kc-3 ji-verb    ((ji-verb .5))) ) )

(defjw kowareruw (cat ji-verb) (expresses kowan)
                (grapheme (past "kowaremashita") (pres "kowaremasu") )

(defjs ji-verb (supercat jverb) (cat-collector jiverb-ness))
(defjn jiverb-ness (japanese (jintransitivec .4)))
```

Figure 5.29: Japanese Intransitive Construction

*apple PATIENT(o) eat*
*eat an apple*

5.35 *meeri ni kissu shita*
*kiss PATIENT(ni) kiss PAST*
*kiss Mary*

5.36 *ookii shoonen wa fooku de meeri no momo o shikago de tabemashita*
*big boy TOPIC fork INSTRUEMT Mary GEN peach PATIENT Chicago LOC(de) eat-PAST*
*a big boy ate Mary's peach with a fork in Chicago*

5.37 *jon ga shikago ni sunde imashita*
*John SUBJ Chicago LOC(ni) live PAST-PROG*
*John lived in Chicago*

Japanese has paired transitive and intransitive verbs. For example the idea of something becoming broken can be expressed with *"kowasu"* if an agent is to be expressed or with *"kowareru"* if not (in FIG's subset of Japanese). If the intransitive (*"kowareru"*) form is used then the patient is marked with *"ga"*. This information is encoded in FIG with the Japanese Intransitive Construction, as shown in Figure 5.29. If no agent has been expressed but a patient has, then

```
(defjc mannr-motionc
  (constituents (mm-1 jverb ((motions-mannr .25) (gerundn 2.)))
                (mm-2 jverb       ) ) )

(defr mannerr   (weights .3 .1) (rel-collector man-ness))
(defn man-ness  (japanese (mannr-motionc .4) (mimetic-phr .35) (jto1w .4)))

(defn motionn
  (english  (dir-particlec .5))
  (japanese (mannr-motionc .5)) )
```

Figure 5.30: The Manner-Motion Construction

the trigger will be satisfied and this construction will force the patient to be marked with *"ga"* and force an intransitive verb.

Figure 5.30 shows the Manner-Motion Construction. In Japanese a manner verb can modify a verb of motion, as in *"nagarete kuru"* (floating come). In this construction the first verb appears in gerund form. The weights on incoming links to this construction are chosen so that **mannr-motionc** becomes activated significantly only when there are two verbs to say, one a motion verb and one a manner verb; otherwise the first constituent would make all verbs be output inflected as gerunds.

The weights are also chosen so that the modifying verb comes before and adjacent to the motion verb; in particular, it appears after all noun phrases.

In a sentence like that above, *"kuru"* (come) actually expresses both the direction of motion and that the viewpoint is that of the woman. In FIG's input this information is not encoded. Thus the semantic update mechanism (§6.4.3) would mark **kurun** as completely expressed prematurely. To handle this problem the semantic update mechanism handles **kurun** as a special case; using the rule that it cannot be considered expressed as a result of inference, but must be expressed directly with a word.

5.38 *minakami kara ookina momo ga obaasan e nagarete kimashita*
*upstream SOURCE big peach SUBJ old-woman DEST floating came*
*a big peach floated down towards the old woman*

```
(defjc mimetic-phr
  (constituents (mc-1 mimetic ((mimetic .8)))
                (mc-2 jto1w   ((jto1w .8))) ) )

(defjs mimetic (cat-collector mimetic-ness))
(defjn mimetic-ness (japanese (mimetic-phr .1)))
```

Figure 5.31: The Mimesis Construction

Japanese allows certain adverbials of manner to modify verbs. The 'case marker' *"to"* is used in these cases. This information is encoded in the Mimesis Construction, as shown in Figure 5.31. Outputs include:

5.39 *donburiko donburako to nagarete kuru*
*bob-sway bob-sway QUOTATIVE float come*
*bob towards*

```
(defjc jtime-settngc
  (constituents (jt-1 jtime-adv  ((jtime-adv 1.2))) ))

(defjc jloc-settngc
  (constituents (jl-1 locationr ((locationr 1.))) ))
```

Figure 5.32: Knowledge about Japanese Sentential Adverbs

The fact that adverbs of time and location are sentence-initial and appear in that order are due to the constructions shown in Figure 5.32. **jloc-settngc** receives activation from **introductoryn**, but not from **locationr** since there is no special tendency for locations to be sentence-initial, except in sentences which serve to open a mental space and introduce something into the discourse.

5.40 *mukashi mukashi aru tokoro ni ojiisan ga sunde imashita*
*long-ago long-ago certain place LOC old-man SUBJ live PAST-PROG*
*once upon a time there lived an old man*

Japanese is verb final. All noun phrases precede the verb. This happens, not because of any construction, but simply because the permanent activation level of **jnoun** is greater than that of **jverb**, and so the verb will not appear until all nouns have been output.

In Japanese any noun phrase can be omitted — thus a verb can by itself be a sentence.

5.41 *meeri o koroshita*
*Mary PATIENT killed*
*[someone] killed Mary*

5.42 *tabeta*
*ate*

5.43 *meeri ni momo o agemashita*
*Mary RECIP peach PATIENT give-PAST*
*[someone] gave Mary a peach*

In Japanese the order of noun phrases is basically free. There are given-new and pragmatic constraints; these are modeled in FIG by its general principle that the most highly activated word is output first. There are also some defaults, for example, agents tend to come earlier than instruments. In the current implementation of FIG, this is largely determined by the weights of the links among the concepts of the input. For example, **agentr** has a relatively large weight. Thus, other things being equal, the filler of **agentr** will tend to become relatively highly activated, and therefore output relatively early. (This particular tendency is reinforced by the fact that **agentr** is permanently slightly activated).

Making FIG handle two languages has not been terribly difficult. In particular, adding the Japanese constructions took only a fraction of the the effort it took to add those for the first language, English. The need to make FIG work for both languages has forced the representation of conceptual (world) knowledge to be 'clean' in the sense that it does not pander to the idiosyncracies of one language. The only place where this was difficult was for the weights on relations among the nodes of the input (§6.2), since these affect word order by affecting the activation levels of the concepts of the input.

| Category | Comments |
|---|---|
| jnoun | |
| jverb | |
| ji-verb | intransitive verb |
| jparticle | nominal particle (case marker etc) |
| jadj | adjective |
| jtime-adv | time adverbial |
| mimetic | mimetic word |

Table 5.2: Syntactic Categories of FIG's Subset of Japanese

# Chapter 6

# Details of FIG



In previous chapters FIG's behavior was described in only enough detail to illustrate the general claims. This chapter remedies the omissions: it is the definitive reference on how FIG works. Taken together with the listing of knowledge in Appendix A this should make it possible for the interested reader to duplicate FIG's behavior exactly. This chapter is not intended to make any new theoretical claims.

## 6.1 Building the Network

This section explains how the structures shown in Appendix A map to network structures. In general, each command beginning with '(def' defines a node, and the subsequent information specifies links to other nodes.

```
(defc gather-woodn
  (nebors (methodr gathern .4)
          (patientr woodn .4)
          (purposer hsehld-fueln .4)
          (locationr regionn .2) )
  (english (gather-woodw .5)) )
```

Figure 6.1: Representation of Some Knowledge about Wood-Gathering

For example Figure 6.1 shows the definition of the node **gather-woodn**. This node has **nebor** (neighbor) links to **gathern**, **woodn**, and so on, and also an **english** link to **gather-woodn**.

Link types are distinguished for the sake of suggesting to the reader the type of knowledge they are intended to encode, but all links all functionally equivalent; that is, activation flow is indifferent to link type, with a handful of exceptions, as noted where relevant. A complete list of link types in FIG appears in §6.6.

Some links include further information. For example the **nebor** link from **gather-woodn** to **gathern** bears the additional information that the relationship between the two nodes is one of **methodr**. This information is only to clarify the rationale for the links; FIG ignores it. Thus the network is more an 'associative network' than a 'semantic network'.

Links are not bidirectional; if inverse links are required they are declared explicitly.

In order to enable concise specification of a network which has many regular structures, the network-building mechanism has a little intelligence. That is, as it goes through the definitions in the file and converts them into nodes and links, it adds some links which are not explicitly present.

```
(defew peachw (cat cnoun conso-inits) (expresses momoc) (grapheme "peach"))

(defes noun (collector noun-ness))
(defec noun-ness (english (prep-phr .65)))

(defes cnoun (collector cnoun-ness) (supercat noun))          ; count noun
(defec cnoun-ness (english (adj-modc .7) (determinatc .71)))
```

Figure 6.2: Illustration of Inheritance of Lexical Knowledge from Syntactic Categories

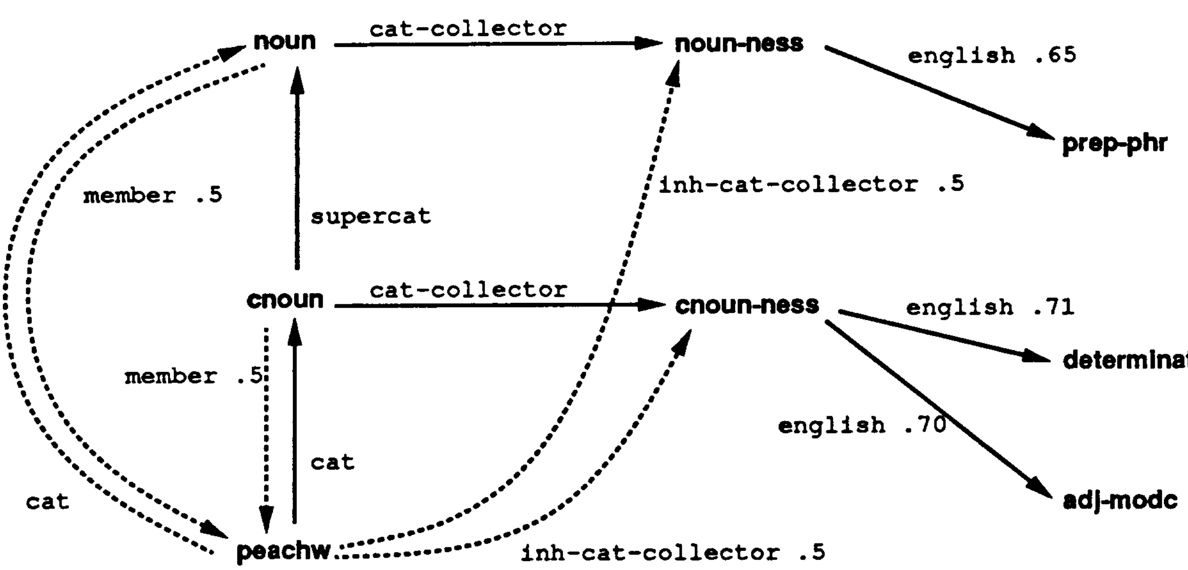


Figure 6.3: A Fragment of the Network. Links shown as solid lines are declared explicitly; dotted lines represent links formed by inheritance at network-building time.

For example member links (from each syntactic category to all words of that category, as discussed in §3.2) are not specified explicitly, rather each **cat** (syntactic category) link automatically creates a member link in the other direction. For example, there is a member link from **cnoun** to **peachw**, since **peachw** has **cat cnoun**, as seen in Figure 6.2. (The reason that **cat** links are explicit and member links are created automatically, rather than the other way around, is simply that it is more convenient to list under a word all categories to which it belongs, rather than to list under a category all the words which are members of it.)

The network-building mechanism also employs a little 'inheritance', not as a mechanism with theoretical significance but merely as a convenience. Inheritance is done when the network is built up rather than at run-time; this is purely for the sake of speed. For example, syntactic categories contain information which is inherited by words when the network is built. For example, all count nouns need a path to spread activation to **determinatc**, since all participate in the determination construction. Such paths result from inheritance of `cat-collector` (category's collector) links associated with the category. This is illustrated in Figure 6.3 by the `inh-cat-collector` (inherited category's-collector) link from **peachw** to **cnoun-ness**. (**cnoun-ness** functions as a 'collector' node for activation from all count nouns; collector nodes are used to avoid having cycles and possible non-convergence of activation levels, as discussed in §6.5.2.) As it happens, the destinations of inherited links are always collector nodes, and hence they are subject to the maximum rule, as discussed in §4.2.4 and §6.3.

Further, there are `supercat` (superordinate category) links among syntactic categories. For example, every count noun is also a noun, so there is a `supercat` link from **cnoun** to **noun**. The effect of this is that every word which is a **cnoun** inherits all the links and special properties pertaining to **noun** also; this is also shown in Figure 6.3. `cat` and `supercat` links only exist for the sake of inheritance; they do not pass activation.

## 6.2 Representing the Input

Spreading activation has several advantages as a modeling technique, as discussed in Chapter 7. On the other hand, it has some disadvantages, notably that it provides no easy way to represent conceptualizations for the input. This section describes how I have worked around this problem. This is not a perfect solution; the flaw is discussed in §6.5.1.

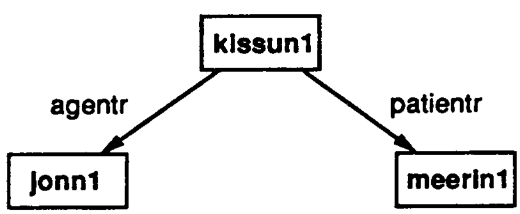


Figure 6.4: An Input to FIG

Suppose that we want to generate the sentence *"John kissed Mary"*. The traditional representation of the meaning underlying this is something like that shown in Figure 6.4 (a copy of Figure 1.2) and this is basically what FIG uses. As is traditional, the component nodes are 'instances', for example, **kissunl** is an instance (copy) of **kissun**. Concepts which are plausibly unique, such as **slangn** (the slang genre) and **presentn** (the present time), are not instantiated.

The nodes of the input are sources of activation. This activation actually comes from a 'pseudo-node' **sourcex**, which is always activated. When the input for FIG is set up, links are created from **sourcex** to the each of the nodes of the input. The reasons for using this particular way of activating the input nodes are explained in §6.5.1 and §6.5.2.

Since the nodes of the input are the sources of activation that drive the entire generation process they must be linked to the network as a whole. This could be done simply with links to the 'parent' concepts, for example from **kissunl** to **kissun**. But for the sake of efficiency it is done by copying all of kissun's links over to **kissun1**. (As long as there is only one instance of any given concept, it is perfectly fine to just use that concept itself as part of the input, and not bother to make an instance at all. In fact, FIG currently cannot handle inputs with more than one instance of the same concept anyway, since the semantic update mechanism (§6.4.3) would get confused.) With this scheme there is no need for activation flow between the generic concept and its instance, so FIG does without. Also, after an instance is made a flag is set on the parent concept which prevents it from transmitting activation, on the rationale that, having been superseded by the new instance, it is no longer relevant. For example, this applies to **kissun** after **kissun1** has been created.

Activation must flow among the nodes of the input for the sake of the locality principle (§4.2.3). For example, for *"John kissed Mary"* activation of **kissun1** must cause activation of **jonn1** and conversely. This happens thanks to the links between neighboring nodes of the input, such as, the **agentr** link in the figure.

This simple scheme for representing the input does not quite suffice; the specific relations which link the nodes of the input have a more complex role to play. (To clarify, while the specific relations among the nodes of the input do play a role, the specific relations among the nodes in world knowledge (§6.1) are ignored.) In FIG relations subsume the notions of relations, slots, and deep cases present in other systems. Relations include not only such obvious examples as **agentr**, **patientr**, and **destinationr**, but also **first-memr** (for the first member of **setn**), **contextr**, **prag-roler** (pragmatic role), **modifier**. (I am aware of the problems of defining a satisfactory set of deep cases, and do not really believe that there is a small finite set of such relations, or that they directly correspond to prepositions and syntactic structures. In fact, use of a small set of relations has been awkward in several ways: in particular, for choosing good weights for **inebor** links, and for governing syntactic update. A better treatment will have to wait for another implementation of FIG.)

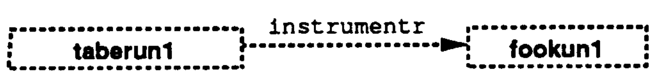


Figure 6.5: An Fragment of an Input to FIG

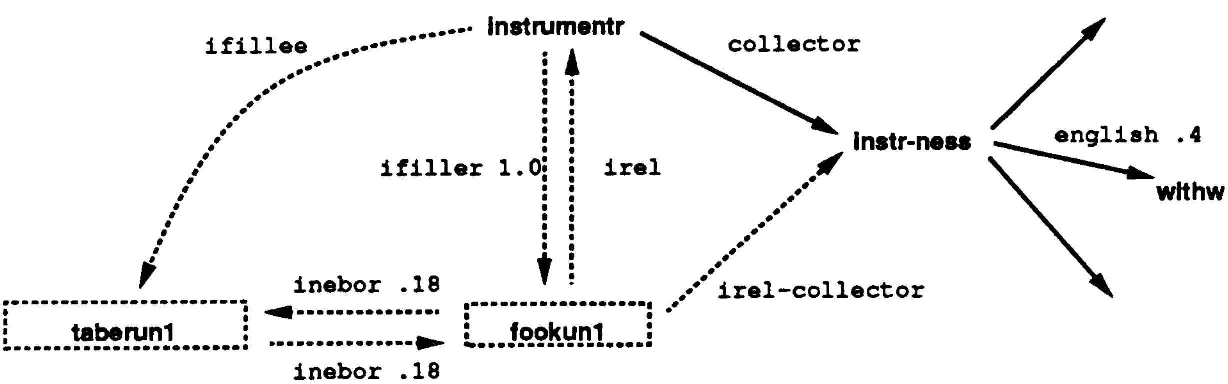


Figure 6.6: 'Compiled' Format of the Input Fragment in Figure 6.5. Dotted boxes and lines represent nodes and links created when the input is built up.

Activation must flow via relations in two cases. These can be illustrated best by an example: consider the fractional input shown in Figure 6.5; this might be created with (plug taberuni instrumentr fookun1) and might be realized as *"... eat ... with a fork"*. First, if **instrumentr** is activated **fookun1** must become active; this is useful so that syntactic constructions can affect the positions in which concepts are expressed. Conversely, if **fookun1** is activated, prepositions associated with **instrumentr** must become active.

These two needs are satisfied by having the simple structure shown in Figure 6.5 automatically map into the configuration of nodes and links shown in Figure 6.6. The latter is the structure that FIG uses for activation flow.

The first need is satisfied by an ifiller (input-relation's filler) link from **instrumentr** to **fookun1**. Activation flow via ifiller links gives the functionality which is provided in most generators by explicitly traversing the links among the nodes of the input, that is, by 'pointer-following'.

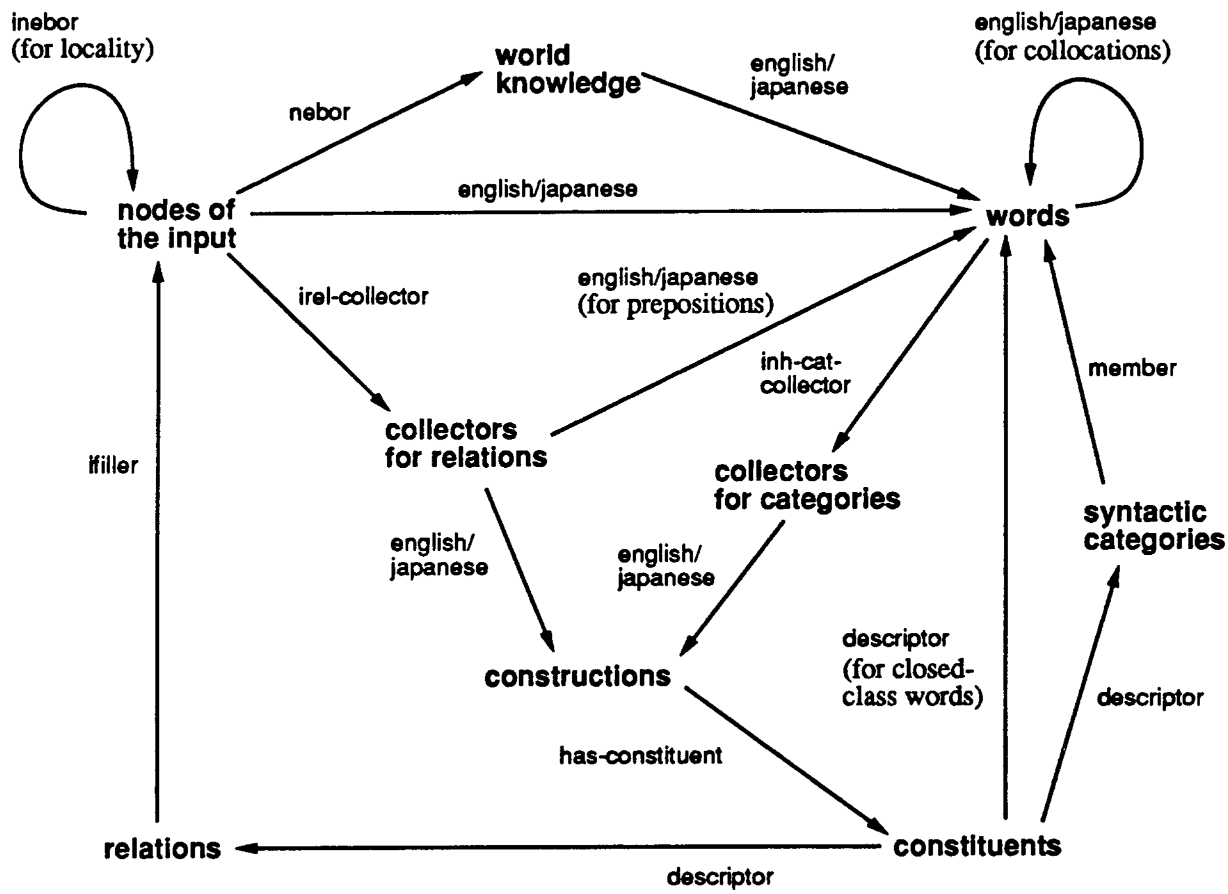


Figure 6.7: Overall Flow of Activation in FIG: Main Pathways. Groups of nodes of the network are shown as spatially separated according to function.

The second need is satisfied by an irel-collector (input-relation's collector) link from **fookun1** to **instr-ness**. In general, there is an irel-collector link from an input node to the collector nodes for every relation it fills; these links associate input nodes with information they have by virtue of filling relations. irel-collector links are subject to the maximum rule (§4.2.4). The need for collector nodes is discussed in §6.5.2. Figure 6.7 summarizes the main pathways for activation flow; it should clarify the role of relations and their collector nodes in the overall scheme of things.

Note that relations are treated as nodes, not links, at the implementation level. Each relation between two input nodes also creates a pair of **inebor** (input neighbor) links; these handle activation flow among the nodes of the input. The final structure also includes some links not used for activation flow, namely **irel**, used for syntactic update (§6.4.2), and **ifillee**, used for semantic update (§6.4.3).

This scheme also works when a relation has two fillers. For example, FIG can produce *"a big peach and a small mountain"* even when both **ookiinl** (big) and **chisaiinl** (small) are **ifillers** of the relation **sizer**. Both of these concepts receive activation from the relation; they get associated with the correct noun thanks to the principle of locality (§4.2.3), as implemented by activation flow across **inebor** links. Thus my non-standard treatment of relational information can handle standard examples also. As another example, FIG produces *"John went to Chicago and to a mountain"* from an input where both **shikagonl** (Chicago) and **yamanl** (mountain) are **ifillers** of the **destinationr** relation. One possible danger with this example is overactivation of **tolw**; this does not occur since **dest-ness**, being a collector node, uses the maximum rule to combine the activation it receives. Another example is *"criticism discouraged John"* which FIG produces from an input where **jonnl** is the patient of both **gakkarin** (become discouraged) and **hihann** (criticize).

As argued in §2.2.1 and §4.5.5, a generator should make the absolute minimum set of assumptions about the structure of its input. FIG's inputs include only two types of information: first, indications of the closeness among the various components of the input (in the form of weighted **inebor** links among the concepts), and second, indications of the semantic roles which the various components of the input fill (in the form of **ifiller** links from relations to concepts and **irel** links in the opposite direction).

This scheme makes FIG robust in the sense that it is not tripped up by minor variations in the structure of its inputs — it is not over-sensitive to the exact topology of its input. This feature complements the flexibility which arises due to activation via world knowledge, discussed in §3.4. In a sense, use of amounts of activation to represent information transmission allows a functional specification of the input, rather than a structural specification.

For example, FIG produces *"the old woman went to a stream to wash clothes"* whether or not the old woman is marked as the agent of the clothes-washing, in addition to being the agent of the going. Similarly *"criticism discouraged John"* is produced whether or not John is the patient of the criticism. Another example is that FIG produces *"once upon a time there lived an old man and an old woman"* regardless of whether the old man, the old woman, and the set consisting of both of them are all listed as experiencers, or whether the set is the only experiencer. A final example is that FIG produces *"a big peach bobbed down towards an old woman from upstream"* regardless of whether or not **downn** (the notion for that direction) is explicitly part of the input, that is, regardless of whether it is a source of activation. In general, FIG is not too sensitive to the exact activation levels of the nodes of the input, thanks to the strong contributions of syntax and activation flow among nodes of the input.

To summarize, the initial state of FIG consists of its usual knowledge network augmented with new nodes and new links, and with some of the nodes made into sources of activation thanks to links from **sourcex**. The new nodes and links are erased after every run of FIG, so that the network is ready to be augmented with the next input. FIG itself never creates nodes or links (with the exception of links from certain pseudo-nodes, as discussed in §6.6.)

```
(let ((kiss-inst (inst kissun))
      (jon-inst (inst jonn))
      (meeri-inst (inst meerin)) )
  (plug kiss-inst agentr jon-inst)
  (plug kiss-inst patientr meeri-inst)
  (list kiss-inst jon-inst meeri-inst) )
```

Figure 6.8: LISP Code to Produce the Structure in Figure 6.4

Programs which use FIG are insulated from the details of how it represents the input. They need only use two functions: `#'inst`, which returns a new 'instance node' when given a node in the network, and `#'plug`, which takes two nodes and a relation name as argument, and creates a link between the two nodes (`#'plug` is so named to invoke the idea of plugging a filler into a slot in a frame). LISP hackers will appreciate that the input shown in Figure 6.4 can therefore be built up simply with the code fragment of Figure 6.8, or by an equivalent series of function calls by, say, a parser. After creating an input network, the program passes a list of the nodes in the input to FIG. These nodes then become the source of activation.

```
(defr instrumentr  (weights .18 .18)  (collector instr-ness))
(defc instr-ness  (english (subj-pred .3) (withw .4))  (japanese (de-w .5)) )
```

Figure 6.9: Knowledge About **instrumentr**

The `#'plug` function is responsible for converting simple links among input nodes to the more complex structures as shown in Figure 6.6. For example, to build the structure shown in Figure 6.6 requires simply `(plug taberun1 instrumentr fookun1)`. The function `#'plug` combines this information with knowledge in the network; for example, it applies information about the `instrumentr` relation, shown in Figure 6.9, to produce the structure shown in Figure 6.6. To be specific, the `irel-collector` link comes from the `collector` of the relation, and the weights on the `inebor` links are determined by the `weights` of the relation. Note that these weights are not necessarily symmetric. (To preclude a possible misunderstanding, the `weights` of relations are used only for specifying the weights of `inebors`; they are not relevant to the `nebor` links in world knowledge; those weights are specified directly, as shown in Figure 6.1.)

Most of the inputs used to produce the outputs exhibited in this thesis are built by a Japanese parser; thus, FIG is part of a Japanese-to-English machine translation system (or a Japanese-to-Japanese system when it produces Japanese). Since the Japanese parser has very limited coverage its outputs do not exercise all of FIG's functionality, so some of the examples in the thesis were generated from inputs assembled by hand.

## 6.3 Activation Flow

Figure 6.10 depicts the computation unit used for nodes in FIG. To summarize, activation from neighboring nodes is multiplied by the link weight to give the amounts received; the function $e$ combines all these types of 'evidence' into a single number $x$; the activation level $a$ of the node is given by $F(x)$, and $a$ is passed through a threshold value $f$, resulting in $o$, the output transmitted to other nodes. This section explains each of these functions in turn, then discusses the initial state of the network, the stability condition, and finally the actual implementation. Some messy details are postponed to §6.5.

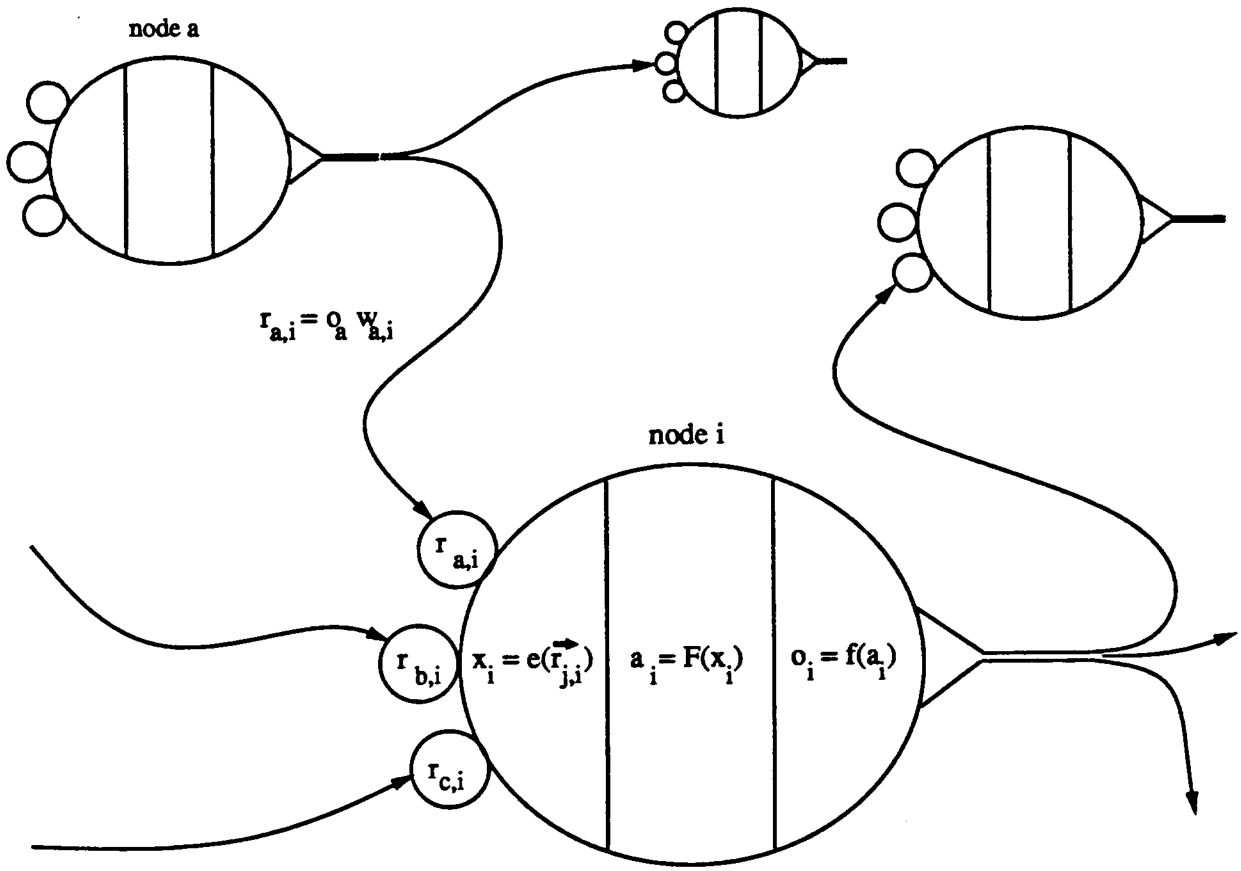


Figure 6.10: FIG's Nodes as Units of Computation

The evidence-combining function $e$ depends on the node type. Most nodes simply sum the amounts of activation received:

**Eqn. 6.1 (Evidence-combining Function** $e$**: most nodes)** $n_i = \sum_i r_{h,i}$

Collector nodes take the maximum of the amounts received, for the reasons discussed in §4.2.4 and §6.2:

**Eqn. 6.2 (Evidence-combining Function** $e$**: collector nodes)** $n_i = \max_h r_{h,i}$

Word nodes take the product of the syntactic activation and the semantic activation:

**Eqn. 6.3 (Evidence-combining Function** $e$**: word nodes)** $n_i = .2(\sum_{h \epsilon Y} r_{h,i})(\sum_{h \not\epsilon Y} r_{h,i})$

where $Y$ is the set of sources of activation which are constituents or syntactic categories. Multiplication by .2 is to prevent the activation levels of words from becoming disproportionate to those of other nodes.

The activation function $F$ is

**Eqn. 6.4 (Activation Function** $F$**)** $a_i = -10 + \frac{40}{1+e^{.15(8-x_i)}}$

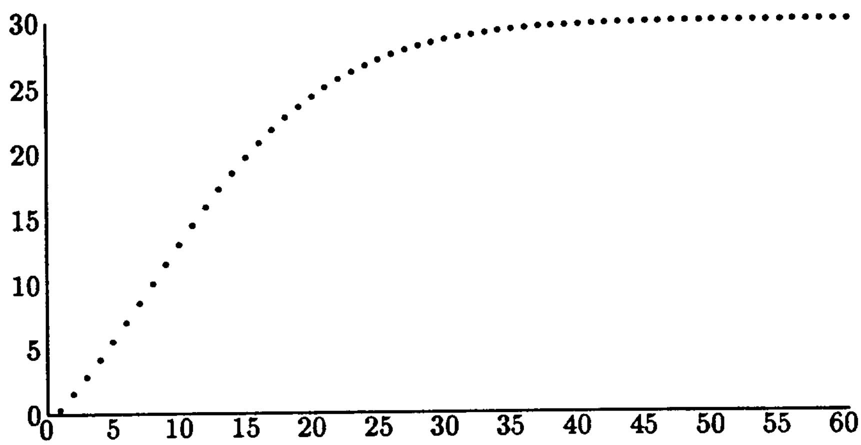


Figure 6.11: The Activation Function $F$

This function was chosen to give a curve which is monotonic, smooth, nearly linear for typical values of $x$ (5 to 15), low for values near zero, and bounded, as shown in Figure 6.11. There is nothing special about this particular function; many other functions would doubtless do equally well. (The actual reason for this function is historical: I had previously been using a piece-wise linear function, and the current $F$ is a smooth approximation to that function. Since it is a close approximation I could switch to it without having to to re-tune all the weights in the system from scratch.)

The amount of activation $o$ that a node emits is given by the threshold function $f$

**Eqn. 6.5 (Output Function $f$)** $o_j = \min(a_j, 0.)$

Thus nodes with negative activation levels send out no activation. This means that, in particular, no matter how badly inhibited a node is, it does not pass this inhibition along to neighboring nodes.

Before generation starts the activation levels of all nodes are zero, with the exception of pseudo-nodes, which always have some activation. These pseudo-nodes are the ultimate sources of activation to the system. After a word is emitted, links from pseudo-nodes are adjusted to reflect the new state, and generation continues. Activation levels are not reset after each word is output; this makes the network settle more quickly than it would if the activation levels of all nodes were set to zero after each word choice.

The network is judged to have reached a stable state if no node's activation level changes more than .5 from cycle to cycle. This number is purely empirical: a higher tolerance risks having settling be cut off prematurely and a lower tolerance makes settling continue longer than necessary, wasting time.

FIG currently generates from full inputs; the Japanese analyzer parses an entire sentence, and then invokes FIG on the result. There is nothing in FIG to prevent it from handling inputs that arrived in pieces, or changed over time. In that case, though, I'd need to add some kinds of control knowledge to make it wait until 'enough' of the input arrived. That is, the emission of a word should depend on some more sensitive measure of when enough information is available or enough computation has been done.

Time is of course discretized. Spreading activation is implemented by having all nodes re-compute their activation levels in synchrony. Each such 'cycle' involves two phases. In phase one each node in the network computes its output value $o$, and 'pushes' it across outgoing links to neighboring nodes. In phase two each node in the network computes its new activation level as a function of the amounts of activation just received.

To save space and time, the implementation does without separate storage for the $r_{j,i}$ (the little receptive circles in Figure 6.10). Instead, activation is accumulated as it arrives, according to the correct evidence-combining function. For most nodes this is just summation. This is done incrementally — scanning through all the nodes of the network, FIG computes the output activation, multiplies it by the weight on each outgoing link, and, at the destination of that link combines it with the amounts of activation already received.

Exactly how this works depends on the node type. For most nodes the activation is simply added to the activation already received, to get $x$. For collector nodes the evidence-combining rule is to take the maximum, so the newly incoming activation is compared with the previous $x$ and if it is higher it becomes the new $x$. For word nodes the accumulation of evidence is a little more complicated. Each word node has two 'inboxes', one to sum activation received from syntactic sources (namely that received across `descriptor` links (from constituents), and across `member` links (from syntactic categories), and one to sum semantic activation (activation received from all other sources.

In phase two the activation level of the node is updated as given by $F(x)$, where for words $x$ is the product of the amounts in the two inboxes.

Undeclared links have weights of zero — this is the vast majority, that is, the network is sparse. In addition, certain types of links do not transmit activation at all, serving other purposes. Such link types are listed without a 'weight' field in §6.6.

In the course of extending the network I frequently run FIG on the entire test suite to check for unforeseen effects. Thus speed has been an important consideration. Spreading activation takes up most of the time in FIG, therefore there are several efficiency hacks to speed this up. 1. FIG calls link-traversing functions (phase one) based on the node-type, to avoid making function calls to process link sets which are always `nil`. 2. Since the only links which point to constituents are those from the constructions they participate in, activation is passed directly from the construction to the descriptors of the constituent under the cursor. 3. Since gathering data on the progress of activation flow is storage-intensive, it is not done except when debugging. 4. The $F$ function is cached for the sake of speed.

## 6.4 Special Processes

This section explains the operation of the FIG's 'special processes', those which are not implemented with spreading activation. §7.2 explains why they are not implemented with spreading activation and discusses the prospects for re-implementing them with spreading activation.

Figure 6.12 is a copy of Figure 1.4 augmented with indications of which sections of this chapter discuss FIG's various operations. Figure 6.13 is another overview of FIG, focusing on how the special processes interact with the network.

### 6.4.1 Word Selection and Inflection

Once the network has settled, a special process scans all nodes of type 'word', finds the most highly activated one, and emits the associated grapheme. This is usually trivial, for example, the grapheme of **the-w** is *"the"*. It is more complex for words which can inflect. In FIG these have lists of graphemes, for example, **eatw** has `(grapheme (inf "eat")(past "ate")(pastp "eaten"))`,

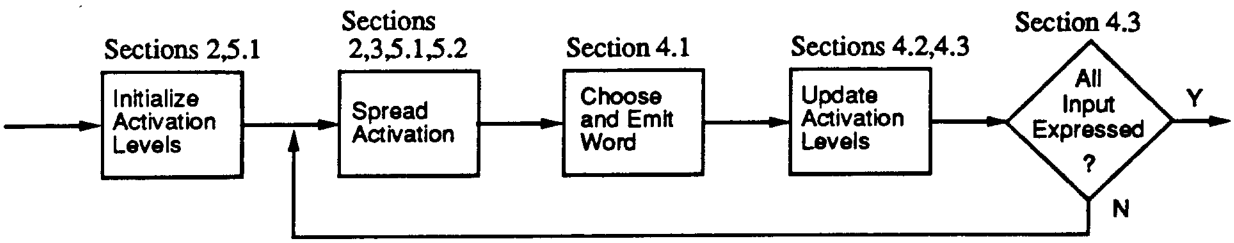


Figure 6.12: FIG Control Flow

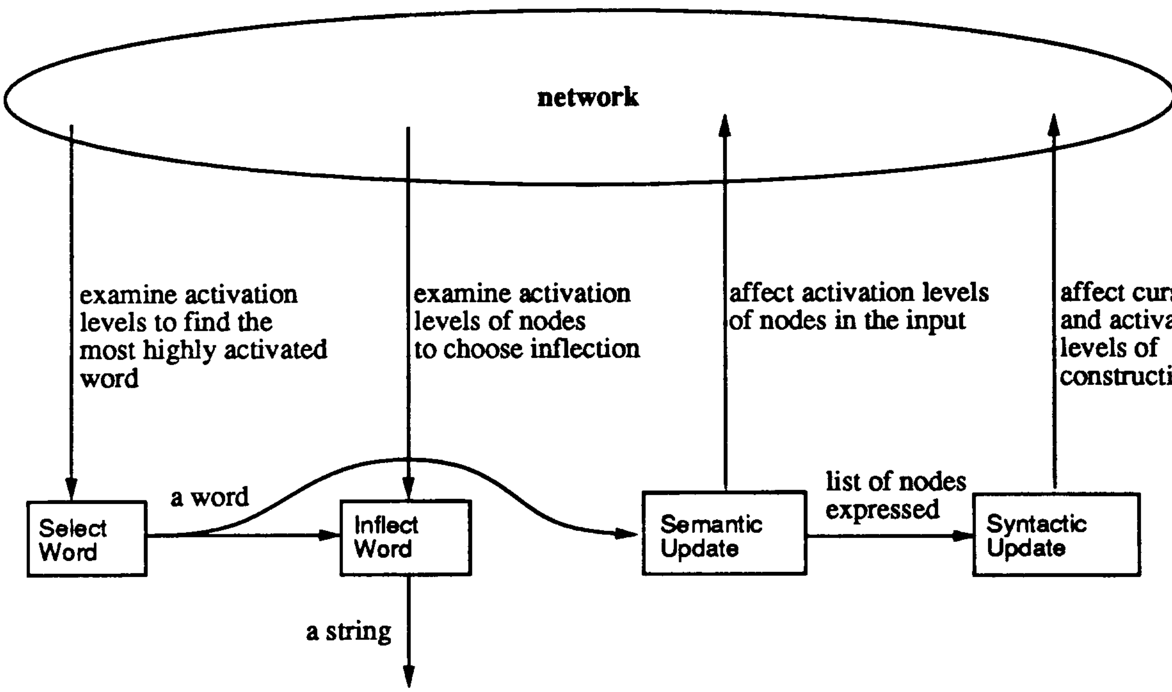


Figure 6.13: FIG Data Flow

representing the infinitive, past, and past-participle forms of the verb. The choice among them is governed by the activation levels of nodes representing syntactic features. For example, if **past-partn** is more highly activated than **presentn**, **pastn**, and **bare-n**, *"eaten"* will be emitted. The nodes which govern inflections typically receive activation from constructions, for example, **passivec** activates **pastpn**.

To summarize, the inflection process takes the list of graphemes for the chosen word, looks up the corresponding nodes, determines which node is most highly activated, and outputs the string paired with that node. FIG uses this mechanism also for choosing between *"a"* and *"an"*; this choice depends on the activation levels of the nodes **voweli-ness** and **consoi-ness**, which are linked to vowel-initial and consonant-initial words, respectively.

FIG handles inflection not to cast light on this aspect of language production, but simply to make the output more readable. While adequate for simple cases, this scheme is very limited; in fact, weak morphology is the main reason why FIG's coverage of Japanese is weaker than its coverage of English.

### 6.4.2 Syntactic Update

As discussed in §4.2.2 it is necessary to update the cursors of constructions so that they accurately represent the current syntactic state. Conceptually each construction monitors the events which occur as words are output and advances its cursor as appropriate.

To this end each constituent of a construction has a trigger. For example, the trigger of **tv-1**, the first constituent of **transitivec**, is 'tverb' (for transitive verb). If the trigger of the constituent currently under the cursor matches an 'event' just flagged, then the cursor advances to the next constituent. For example, after **eatw** is selected, the event 'tverb' is flagged, and the cursor of **transitivec** advances. Events include both syntactic and semantic happenings; to be specific, events are: the name of the word, the names of its syntactic categories, and the names of the relations filled by any concepts 'covered' by that word. For example, after emitting *"Mary"* in *"John kissed Mary"* the events are 'maryw', 'pnoun', 'noun', and 'patientr'. Triggers need not refer to just a single event; a trigger consists of an expression where the literals are events and the operators are: not, and, or, ever, and never. For example the trigger of **dt-2**, the first constituent of **determinatc**, is **(or article apostrophe-s)**, meaning that either an article or a possessive noun can count as a determiner.

A construction 'resets', that is the cursor moves back to the left-most constituent, when all constituents have been triggered. Also, a construction resets immediately if the special 'reset-if' trigger is matched by an event.

Also, FIG must boost the activation level of relevant constructions, specifically those which are 'in progress'. These are all constructions whose cursors have ever been updated but have not yet been reset.

The current syntactic update mechanism is one of the weak points of FIG. One problem is that it makes for redundancy in the representation of constructions — the trigger of a constituent is often simply the name of the node that constituent is linked to. Another problem is that, since events are very simple, triggers must be cleverly crafted to fire when the rightmost word of a constituent has been emitted. An example of a complex trigger is that for the first constituent of the **subj-pred** construction as discussed in §5.2. If events included more semantic happenings, such as 'subject completely expressed', complex triggers would not be necessary.

### 6.4.3 Semantic Update

Semantic update is needed so that the activation levels of the input concepts reflect what has been said and what remains to be said. This involves two things: determining which concepts have been 'expressed', the topic of this section, and making their activation levels be very low, which is explained in §6.5.1.

**express** links associate words with the concepts they express. For example, the word **peachw** expresses the concept **momon**. **express** links are in general inverses of **english** and **japanese** links. Once a word has been output, all the concepts to which it has **express** links are marked as expressed.

Concepts can also be expressed indirectly; to determine what has been expressed requires inference. This is a special case of the more general inference problem, often studied in connection with parsing. The problem is simpler here, since FIG knows what it wants to say, and is only interested in how much of it it has expressed. Ideally, of course, a generator should be able to detect whether its words have expressed more than intended, in particular, whether they might have led the hearer to an inappropriate inference.

To decide which concepts have been expressed, FIG starts from the concepts directly expressed

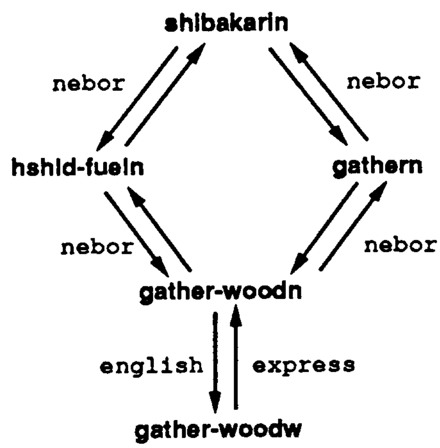


Figure 6.14: Links Used to Infer that **shibakarin** has been Expressed

by words and 'searches' backwards through the network structures representing world knowledge. The implementation uses a rule inspired by an inference method of Norvig (1987): if a concept X of the input can be reached by two paths from concepts which have been expressed, then X is also expressed. For example, after having output *"gather wood"*, the concept **gather-woodn** is directly expressed, and the concept **shibakarin** is indirectly expressed, because of the path **gather-woodn** → **hsehld-fueln** (fuel for household use) → **shibakarin**, and also a path **gather-woodn** → **gathern** → **shibakarin**, as shown in Figure 6.14. This rule is ad hoc, much too simplistic, and relies on the exact topology of the encoding of world knowledge. There is also an exception to compensate for inadequate inputs, as discussed in §5.3.

FIG ends when all 'non-persistent' input concepts have been expressed. 'Persistent' nodes are those which should affect an entire utterance. Whereas most nodes of the input can be seen as goals which can be achieved, persistent nodes represent background goals, such as 'use the **fairy-talen** genre', 'use **slangn**', or 'describe an event that happened in the **pastn**'.

FIG exhibits the 'locality principle', required as explained in §4.2.3, thanks to two entirely disparate mechanisms: Before a node is expressed, activation flows to all its **inebors**, that is, nodes which are its neighbors in the input, as explained in §6.2. But immediately after a node is expressed its neighbors should still be in the focus of activation. For example, when discussing a big peach, **momon** (peach) should become highly activated immediately after *"big"* is emitted. Therefore there is a mechanism to boost the activation of nodes 'adjacent in the input' to nodes just conveyed. These adjacent nodes are: **inebors**, such as **taberunl** (eat) for **fookunl** (fork) and vice versa; **irel-collectors**, such as **instr-ness** for **fookunl**, for the sake of particles; **ifillers**, such as **fookunl** for **instrumentr**, for the sake of nouns after prepositions; and **ifillees**, such as **momonl** (peach) for **possessor**, for the sake of outputting the correct concept after apostrophe-s. This boost lasts until new concepts are expressed. The exact amounts of activation involved in these boosts are given in §6.6.

These two kinds of update are only superficially in contradiction. Consider FIG generating a sentence involving a peach bobbing downstream. After FIG outputs *"bobbing"*, adjacent concepts reached via world knowledge, and thus considered inferrable by the hearer, such as **nagarerun** (float), are expressed and thus inhibited; but adjacent concepts reached via **inebor** links, such as perhaps **downstreamn**, enter the focus of activation, and thus are activated.

## 6.5 Getting the Correct Overall Behavior

Activation serves many roles in FIG. It is easy to say: 'let's build a system where spreading activation is used for everything'. But making such a system actually run requires some extra effort: all the uses of activation must interact sensibly. This section explains some problems and solutions.

### 6.5.1 Activation of the Input

It is particularly troublesome to arrange things so that the nodes of the input have appropriate activation levels. There are three main goals that an activation scheme must attain:

1. The activation level of a node must reflect the fact that it participates in the input and has not yet been expressed.
2. Nodes must be responsive to activation from syntax. For example if **transitivec** activates **patientr**, activation must then flow to `ifillers` of **patientr**.
3. Activation must flow among the nodes of the input (across **inebor** links), for the sake of the locality principle.

Need 1 requires the nodes of the input to always have some substantial activation. The simple rule 'clamp all nodes which are part of the input at a certain level of activation until they have been expressed' will not work, since these nodes must still be responsive to activation from syntax and from inebors. FIG's solution to this is to treat the 'bonus' for being part of the input exactly the same as any other type of activation. This is why activation to nodes of the input comes from the pseudo-node **sourcex**, as discussed in §6.2.

This scheme requires a source of negative activation (inhibition) to link to nodes of the input which have been expressed to prevent 'rebound'. For example, after emitting *"big"* it is not enough to merely erase the link from **sourcex** to **ookiinl** (big) and thereby remove its principal source of activation, since it will still receive activation from its yet-unsaid **inebor momonl** (peach). To counteract this, after a node is expressed, FIG links it to the pseudo-node **coveredx**, from which it receives inhibition.

FIG is too simplistic here — there can be cases where a node, once expressed, may still need some activity to affect some other part of the utterance, as when producing *"John washed himself"*. To handle such cases it might be appropriate to instead mark relations as covered, for example, after outputting *"John"* the generator might only inhibit the **agentr** relation, leaving the **patientr** relation to cause the output of *"himself"*. Inhibition of relations would also improve uniformity: currently the semantic sources of activation to open-class words, namely concepts, are marked as expressed, whereas the primary sources of activation for closed-class words, namely relations, are not.

There is also a fourth goal which an activation scheme must attain.

4. The activation ratio between adjacent nodes of the input should not vary widely. For example, if the input involves a 'big peach' the activation levels of the nodes **ookiinl** (big) and **momonl** (peach) must always be in approximately in the same ratio, regardless of the state of the generation process or the presence of other nodes in the input.

This is important because the activation levels of concepts affect word order. Adjectives, for example, must appear early (before the noun) but not too early (not before the determiner). Since the activation level of a word depends not only on activation received from constructions but also on activation received from concepts, the activation levels of concepts must be also be in appropriate ratios. One case where FIG often failed in the past is for sentences with two nouns and one adjective, such as *"a man ate a big peach"*, where there is the danger of outputting the adjective with the wrong noun. To fix this required making the locality principle work well enough to ensure that when **otokon1** (man) is the most highly activated concept-node, **ookiin1** (big), which receives energy from it only indirectly (via an inverse-**agentr** link, a **patientr** link, and a **sizer** link), will not be activated sufficiently to be output next to *"man"*. Tuning the weights to make this work was a major strain if the ratio between the concept for the adjective and the concept for the head noun was not constant over time.

I eventually decided to add a 'ratio mechanism' to enforce goal 4 explicitly, as follows: Certain relations, when instantiated as **inebor** links, constrain the activation levels of the nodes at the ends to be in a fixed ratio (generally 1.0). This is handled by a special 'ratio' mechanism, which, after phase two of each cycle of spreading activation, detects whether any such pair of nodes deviates from the specified ratio, and if so, boosts the activation of the one that is too low. This rule applies currently to the relations **modifier**, **sizer**, **degreer**, and **possessor**. These are the links which tie together the information which is typically realized as a noun phrase; thus this rule is not entirely unjustified: it is plausible that an entire 'complex concept cluster', including modifiers, is necessarily uniformly activated. For example, to produce *"a very big peach"* **sugokun1**, **ookiin1** and **momon1** can all be equally activated.

The ratio mechanism has been a mixed blessing — the basic problem is that need 4 conflicts with needs 2 and 3. The ratio mechanism completely prevents responsiveness to activation from relations, need 2. This is not generally a problem, since within a complex concept cluster there does not need to be any responsiveness to activation from relations, because the constructions involved in noun-phrases do not refer to relations. For example, the first constituent of **adj-modc** is linked not to the relation **modifier** but to the syntactic category **adjective**, thus ordering is done not by differential activation of concepts but of words.

There is one case in which the ratio is not 1.0: the case of the **possessor** link. The problem is that both the possessor and the possessee are realized as nouns, as in *"the boy's peach"*, hence there is no way to select among them, and order them, by syntactic activation alone. So it is necessary that the concept filling the **possessor** relation be more highly activated than the concept which is the possession. To handle this case the fixed ratio of nodes joined by **possessor** links is 1.04, which is just high enough to cause the possessor to be expressed before the possessed.

The ratio mechanism relies on direct links among nodes of the input. As such it is not adequate for cases where the input concepts do not map directly to words. Suppose the input includes **onsenn** (hot spring). Activation will flow from this concept indirectly to **hotw** and **springw**. However, there is no direct link between the concepts underlying these words, and so no way to enforce the 1.0 ratio, and thus a danger of misordering the words *"hot"* and *"spring"*. FIG works on this particular example thanks to a tricky link weight, but there is no easy or obvious way to deal with such cases in general.

Thus the current scheme for handling the activation levels of the nodes of the input is inelegant and inadequate. I take some consolation from the general lack of satisfactory connectionist treatments for the representation of conceptualizations, that is, the contents of short-term memory. The task here is even harder, since it requires the activation levels of various parts of the conceptualization to vary over time.

### 6.5.2 Activation of the Network as a Whole

This subsection discusses some desiderata for the activation levels of the network as a whole.

**The activation levels of the network should converge over time.** The first thing to avoid is oscillation. Oscillation has not recently been a problem, probably because there are currently very few inhibitory links, and because nodes with negative activation levels send out no activation or inhibition at all. Several early versions of FIG had problems with oscillation; it was suppressed by damping (adding a decay term to the activation rule) at the price of slowing down convergence, and thus the overall speed, by about a third.

The second thing to avoid is unbounded increases of activation levels. This problem sometimes arose during development, although not often, since the net is fairly sparse, and weights are generally low (to be specific, the typical weight is .5, and few weights are over 1.0). The maximum rule (§4.2.4) also helps prevent unbounded increase in activation levels.

The usual cause for an unbounded increase in activation was the presence of a cycle in the network. Sometimes this was solved by simply lowering the weights of the links comprising the cycle. Other times it was necessary to break the cycle. This is the reason for the use of 'collector' nodes, such as **instr-ness** and **cnoun-ness**. If prepositions and so on are activated in the obvious way, by the filler of a relation activating prepositions via activating the relation, there will be a tight cycle. This is because the relation will, in turn activate the filler, as discussed in §6.2. For the example shown in Figure 6.5, there would be a tight cycle with **fookunl** and **instrumentr** activating each other, which would cause the activation levels of **instrumentr** and **fookunl** to increase without ever converging.

Collector nodes allow paths from a relation or syntactic category via a concept back to nodes associated with that relation or category, without requiring a cycle. This can be seen in Figures 6.3 and 6.6 (recall that `cat`, `supercat`, and `irel` links do not transmit activation). For example, activation can flow from **instrumentr** to **fookunl** to **instr-ness** to **withw**.

The problem of cycles is also why there are few bidirectional links in world knowledge, and why the ones that do exist have low weights; and why the links from words to the concepts they express do not pass activation. Non-convergence is not intrinsic to the presence of cycles; it would doubtless be possible to eliminate the problem by replacing $F$ or $f$ with a function which is horizontal beyond a certain point.

There are of course cycles in the input, in that `inebor` links come in pairs. The weights of these links are generally low, to prevent unbounded increases in activation, but, because they are low, there is relatively little activation flow between input nodes, and thus the locality principle is not implemented as well as it should be, and crosstalk can still occur.

It would be nice to prove that the network is stable for all inputs. FIG, however, uses a non-linear activation function, and also multiplication and the maximum rule, so it would be hard to state a concise condition for convergence.

**The network must be robust.** In particular, the best word at any given time must have significantly more activation than any other word; otherwise in a slightly different context another word may become more highly activated and thus be emitted, inappropriately. In earlier versions of FIG this was a serious problem: many words would have nearly the same activation, thus making selection of the best word rather fragile, and making weight-tuning an arduous process. This problem is handled in part by careful design of the network (§6.5.3). Another fact which alleviates this problem is that the activation level for words is given by the product (not the sum) of incoming 'syntactic' and 'semantic' activation. Because of this, no word becomes highly activated unless it is both syntactically and semantically appropriate.

To implement this requires a distinction between 'syntactic' and 'semantic' activation. There are many possible ways to implement this, for example, by changing FIG into a marker-passing system with different markers for syntactic and semantic activation. In the end I found a much simpler solution. Since it is only for words that syntactic and semantic information come together, syntactic and semantic activation is only distinguished at this point; this can be done by simply dividing incoming links into two sets, as discussed in §6.3.

**The network should be easy to debug.** This suggests the use of a smooth activation function. Painful experience has shown that if the activation function has any 'kinks', that is, places where the derivative is not continuous, it is much harder to find an appropriate set of weights. In early versions of FIG the representation of the current state of the generation process was done by imposing a ceiling or a floor on the activation level of a node, such as, a construction in progress. This of course meant that such nodes had non-smooth responses to other inputs. Now the representation of the current state is in the form of links from pseudo-nodes, namely **in-progressx**, **in-focusx**, **sourcex**, and **coveredx**, and activation from these nodes is simply summed together with that from ordinary nodes.

**Activation levels in the network should have approximately the same meaning across inputs.** For example, if **adj-modc** has 25 units of activation it should count as being highly activated and should have a major effect on word choice, regardless of the activation levels of other words. The thing to avoid is over- or under-activation of the network as a whole. This is a real possibility: the most common reason is that, due to activation flow among neighboring nodes of the input, large or dense inputs cause activation levels generally to rise. This is a problem because certain amounts of activation are fixed quantities, such as activation from **in-progressx** to syntactic constructions and from **permanentx** to nouns. Thus if many nodes of the network become highly activated for some input, syntactic considerations can be 'drowned' out.

To prevent this from happening there is a special 'scaling mechanism'. This ensures that the most highly activated node of the input has about 20 units of activation, whatever the input and whatever the current context. This is done by modifying the activation of **sourcex** so that this condition holds. This solution is better than simply capping levels of activation since it does not change the ratios among the activation levels of the nodes of the input. This adjustment of the activation of **sourcex** is done on every cycle of activation flow, after phase two. (This mechanism is not used (or adequate) to compensate for cases of non-convergence.)

Another factor that contributes to making activation levels invariant with respect to input size is the maximum rule. For example, the activation level of **determinatc** does not depend on how many nouns are activated. In early versions of FIG I also experimented with the idea of conservation of activation; under this scheme activation was never created or destroyed, that is, a node could not activate another node unless it gave up some of its own activation. This idea did not work out well.

**The network should have purely local computation.** This is important for the sake of efficient parallel implementation. There are two aspects to this. First, the activation level of a node should depend on nothing but its own previous value and the activation it receives from neighboring nodes. Second, the amount of activation emitted by a node should depend only on its activation level. Unfortunately FIG violates both aspects: the ratio mechanism and the scaling mechanism both perform non-local computation.

Thus the current scheme for activation flow has some unresolved problems. It should not, however, be difficult to come up with a better scheme; I imagine that experimentation with different evidence-combining functions and activation rules could lead to a satisfactory solution.

### 6.5.3 Extending FIG

In general, the difficulty of extending FIG to handle something new depends on how different it is from what FIG can already handle. For example, it is trivial to add a new open-class word whose meaning is a single concept. This subsection focuses on more interesting cases.

Sometimes the starting point for an extension is the desire to be able to express a new input, sometimes the desire to produce a novel output, and sometimes the desire to incorporate and use some new knowledge.

In any case the first step is to think about the knowledge required. There is no particular method for this — it is based on intuition. In particular, adding syntactic knowledge is non-trivial since there is no existing grammar that FIG is based on. That is, to extend FIG's grammar involves not only encoding data and debugging it, but also empirical work on the facts of language. Of course, a lot can be learned by examining treatments of the phenomena in other theories, computational and otherwise.

The second step is to decide how to encode the knowledge in the network. Of course this overlaps with the first step. It is very helpful to use analogy to previous knowledge if appropriate, especially for choosing link weights.

The third step, which may again overlap with the other steps, is to work up appropriate test inputs. I create input structures based on my knowledge of Japanese and English, trying neither to cheat nor to make things artificially difficult.

Finally I run the extended FIG on the new input. It may fail to converge, in which case I deal with it as discussed in the previous subsection. Otherwise it either produces the desired output or it doesn't. The rest of this subsection discusses what to do if it doesn't.

FIG being an incremental generator with emergent syntactic choice, its only failure mode is to pick an inappropriate word. That is, there may be a time when the desired word has less activation than some other word. Some typical problems are: over-activation of a syntactically inappropriate word, resulting for example in *"once upon a time there* an ... *"*; over-activation of a semantically inappropriate word, resulting for example in *"*to *the hills"* instead of *"*into *the hills"*; and over-activation of a word expressing an inappropriate concept, resulting for example in *"a* big *peach and a fork"* instead of *"a peach and a* big *fork"*.

Given a problem, the solution is to change things either so as to increase the amount of activation reaching the desired word or to decrease the amount of activation reaching the inappropriate word. In either case it is necessary to decide which of the sources of activation should be boosted or attenuated, and then to decide whether this should be done by changing the weight on the incoming link or by recursing; that is, trying to boost or attenuate the activation level of the node at the other end of the link. From words we might recurse backwards, for example, to syntactic categories, then to constructions, and then to relations. It occasionally necessary to recurse all the way back to pseudo-nodes, in which case it may necessary to improve the syntactic or semantic update process or even alter an evidence-combining function.

The process of deciding which link to change is eased by a window-based environment which displays nodes and their activation levels, and provides the ability to click on a node to see the sources from which it received activation, or to edit its definition. Figure 6.15 shows the status of FIG before outputting the first word of *"the big boy ate Mary's peach with a fork in Chicago"*, the example discussed in §1.2.4. The top left pane, labeled 'Input', shows the Japanese words typed in to the parser; the pane below that, labeled 'JKodiak', shows the nodes of the structure built up by the parser; and the pane below that, labeled 'Output', shows the target language and the output so far. At the right is a list of highly activated nodes, ordered by node type, and

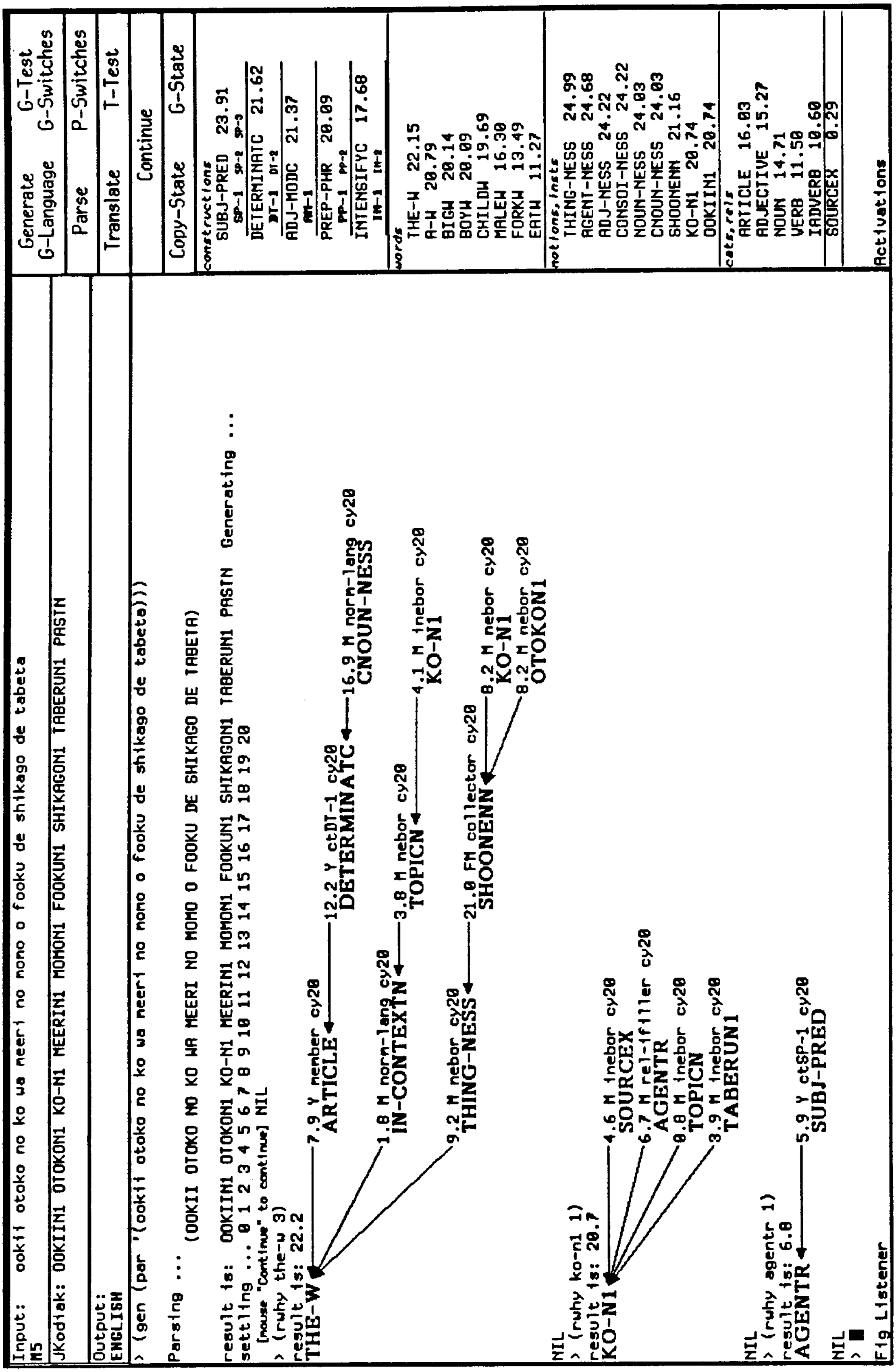


Figure 6.15: FIG Window Interface

showing the amounts of activation each has. The main pane, at the lower left, shows recursively the sources of activation for **the-w**, **ko-n1**, and **agentr**.

Another helpful technique for deciding which of the links on a path to change, is to look for a similar input on which FIG succeeds, compare the two states of the network, and so locate the link which can be blamed and safely adjusted. (I have experimented with automating this process but have not so far had much success: either the link to be changed is immediately obvious (to me) or there are conflicting indications as to which link to change and careful thought is required.)

Having chosen which link to adjust there remains the question of how much to change the weight by. This is usually not difficult, since the activation function is almost linear in the range of typical inputs. Of course this is not completely trivial, in part because of the non-linearities and in part because the network is not a tree — there are even a few cycles. It is possible that a full-blown back-propagation scheme would eliminate not only the problem of choosing weights but also the problem of deciding which link to change, although the presence of cycles and the maximum rule make standard back-propagation inapplicable.

Sometimes there are subtle problems, where the same weight should be made higher by some criteria but lower by others, as sometimes happens when trying to make different inputs both work. That is, a weight chosen to enable correct generation from one input may interact with other knowledge in the system in such a way as to cause failure for some other input. To detect such problems, after making any change, I test the network on a standard corpus to make sure that nothing has broken. A problem requiring the fine tuning of a weight is often a sign that more nodes and links need to be added. That this is sometimes necessary is not surprising — FIG contains only weak approximations to the knowledge a generator should have, and only a tiny fraction of that.

This process of adjusting weights does not lead to any great truths. Quite the contrary, the link weights found are only appropriate for the specific functions $e$, $F$, and $f$ and the specific update processes used. On the other hand, the importance of the weights must not be belittled. The correct operation of FIG depends on having correct link weights. More generally, weights are essential to the task of generation, given that generation is not a simple process of mapping (§2.2.2), but rather requires trade-offs among various goals (§2.3.4).

I have no theory of weights, indeed finding appropriate ones is still partly an empirical process. There are, however, some heuristics for weight choosing:

- Consistency. Similar links should have the same weights. As a corollary, link weights should not be ad hoc. It is not a good idea, for example, to change some weight to .8798 just to make one example work right.

- Logic. Many of the weights have a rationale: for example, the link from **determinatc** to **article** has a relatively high weight because articles get very little activation from other sources. The values of link weights are not important in themselves; their values in relation to other weights are.

- Position independence. For example, regardless of whether "*the old man and the old woman*" is to be the subject, object, object of a preposition or what-not, those words should all be activated to the same degree when this phrase is due to be output. Otherwise it is likely that the order of adjective and noun within the phrase, for example, may be scrambled depending on where the phrase appears. For example, when **sp-1** was linked to **noun**, the first noun of the sentence would become so intensely activated that it would appear before any adjective. This goal of position independence is the reason why most constituents activate semantic relations instead of syntactic categories. It is also one of the reasons that some syntactic categories are permanently activated, that is, they receive activation from **permanentx** instead of via constructions. It is also a reason for the use of the scaling mechanism (§6.5.2).

- Content independence. A strength of FIG is the fact that activation represents current relevance as well as eventual relevance. This means that there is no separate mechanism for determining word order — it is emergent. But this strongly constrains weights. For example, to produce *"John ate the old man's peach"* requires FIG to activate **ojiisannl** (old man) strongly, so that it precedes *"peach"*, and it must activate **the-w** even more strongly, so that it precedes *"old man"*, which might make the article so highly activated that it precedes the subject. To avoid such problems, weights should be chosen so that competition for position within a 'constituent' does not affect position of the constituent with respect to other constituents. For example, regardless of whether the direct object will be expressed as *"John"*, *"the strong old man's very big peach"*, or *"an old man and an old woman"*, the first word should be activated equally highly in any case. This is so that the internal structure of the 'phrase' realizing the direct object does not affect its competition with other phrases, so it is still ordered correctly with respect to them.

This is not to pretend that debugging and extending FIG can be done by following a few simple rules. To extend FIG to handle entirely new phenomena sometimes requires not just more network but actual changes to the activation function or to the special processes. For example, extending FIG to handle Japanese particles required refinements to the update mechanism, as discussed in §5.3, to make it a better implementation of the locality principle. To extend FIG to handle other things may also require major refinements; for example improved update functions and new types of nodes with different evidence combining functions, activation functions, decay rates, and so on.

The fact that there is no good algorithm for building FIG's network must not be taken as an argument against the theoretical claims of Chapter 2. The study of generation and the study of how to learn the knowledge needed for generation are largely independent. I suspect the problem of learning could best be addressed not by working with FIG-as-is but by building a PDP model exploiting the ideas explored in FIG.

## 6.6 Summary of Node and Link Types

FIG has nine types of nodes, as shown in Table 6.1.

Node types are distinguished mostly for the sake of clarifying their intended meanings. FIG pays attention to node types only for three reasons: first, to implement the 'special properties' shown in the table, second, for the sake of efficiency, since certain node types have no links of certain types, and third, so that the display routines can display the various types differently. Words, constructions, syntactic categories, and concepts are further subdivided into those used for Japanese, those used for English, and those used for both; this latter division is purely for the sake of an efficiency hack.

'Notion' nodes typically represent concepts. FIG's inventory of concepts is not particularly different from that used in any semantic network. Some concepts correspond directly to specific actions, objects, and properties, for example **kissun** (kiss), **momon** (peach), and **ookiin** (big); others are more abstract, for example **thingn**, **mannern** and **regionn**; still others represent

| node type | suffix | special properties |
|---|---|---|
| notion | n | |
| word | w | selectable for output, special evidence-combining function |
| relation | r | |
| construction | c | has a cursor |
| constituent | -1, -2 ... | |
| syntactic category | | |
| instance of notion | 1,2 ... | |
| collector | -ness | special evidence-combining function |
| pseudo | x | described in §6.5.1 and §6.5.2 |

Table 6.1: Node Types

script-type information, for example **gather-woodn**. Notion is also the catch-all category for nodes which do not seem to belong elsewhere, such as **slangn**, and **fairy-talen**.

FIG also has 'pseudo-nodes': unlike other nodes, these do not encode knowledge, but exist only for the sake of the implementing the generator. There are five such nodes:

**coveredx** exists to inhibit nodes of the input which have already been expressed. **in-focusx** exists to activate nodes which are close in the input to the nodes just expressed. **in-progressx** exists to activate constructions which are in progress. **permanentx** exists to activate nodes which should always be activated, regardless of the input and the current syntactic state. **sourcex** is responsible for activating the nodes of the input. It is unique among the pseudo-nodes in that its activation level changes over time (§6.5.2), the other four pseudo-nodes always have exactly 1 unit of activation.

The rest of this section lists all the types of 'links', broadly defined, in FIG. That is, it shows all information associated with nodes, including true links to other nodes, information used to govern inheritance of links, and information used by the special processes. This is the finest division possible: links are classified into different types if there is any difference whatsoever in their purpose, provenance, or behavior.

The fact that there is a long list of link types in FIG does not invalidate my claim that generation with spreading activation can be simple and elegant. Indeed, the proliferation of link types is largely due to the fact that FIG is not a pure spreading activation system.

As a general rule, link types beginning with the prefix 'i' are set up when the input is assembled. The exceptions are link types with the prefix 'inh', which stands for inheritance.

`nebor` from/to: notion→notion
  example: **gather-woodn→hsehld-fuel**
  purpose: mostly for world knowledge (§3.4), also the vanilla, catch-all link,
    also used in semantic update (§6.4.3)
  special properties: has an associated label (§6.1)
  weight: varies

`cat` from/to: word→syn-cat
  example: **peachw→cnoun**
  purpose: creation of `member` links, inheritance (§6.1)
    syntactic update, inflection

`member` from/to: syn-cat→word
example: **cnoun→peachw**
purpose: primarily as part of paths for activation flow from constructions to words (§3.2)
provenance: automatic inverse of `cat` (§6.1)
weight: always .5

`cat-collector` from/to: syn-cat→notion
example: **cnoun→cnoun-ness**
purpose: not used at run-time,
capture generalizations about the syntactic properties of words (§6.1)

`inh-cat-collector` from/to: word→notion
example: **peachw→cnoun-ness**
special properties: subject to the maximum rule (§4.2.4)
provenance: automatically created by `cat` and `cat-collector` (§6.1)
weight: always 1.

`supercat` from/to: syn-cat→syn-cat
example: **cnoun→noun**
special properties: make all words belonging to the first category
inherit membership in the second category (§6.1)

`english` from/to: relation, notion, word→word, construction
example: **momon→peachw**
weight: typically .5
special properties: vanishes if generating Japanese

`japanese` from/to: relation, notion, word→word, construction
example: **momon→momow**
weight: typically .5
special properties: vanishes if generating English

`has-constituent` from/to: construction→constituent
example: **determinatc→dt-1**
special properties: weight adjusted by the syntactic update process (§6.4.2)
weight: 1.0 if pointed to by the cursor, 0. otherwise (§4.2.2)

`descriptor` from/to: constituent→relation, word, syn-cat, construction
example: **tv-2** (the second constituent of **transitivec**)→**patientr**
weight: varies

`valence` from/to: word→construction (§3.2,§4.1)
example: **go-w→go-c**
weight: typically .2

Links used in making inputs:

`instance` from/to: notion→instance
example: **momon→momon1**
purpose: for semantic update (§6.4.3),
and for determining whether this concept needs to send out act (§6.2)
provenance: created by `#'plug` (§6.2)

`generic` from/to: instance→notion
example: **momonl→momon**
purpose: used for semantic update (§6.4.3)
provenance: created by `#'plug` (§6.2)

`inebor` from/to: instance,notion→instance,notion
example: **kissunl→jonnl** (as **agentr**)
purpose: locality principle (§4.2.3, §6.2)
provenance: created by `#'plug`
special properties: always paired with an `inebor` link in the opposite direction,
also used for semantic update, handled specially by the 'ratio' mechanism(§6.5.1)
weight: varies, determined by the associated relation

`weights` from/to: relation→a pair of numbers: specifying weights in both directions
example: **instrumentr**→(.18,.18)
purpose: provide the weights to use for `inebor` links (§6.2)

`irels` from/to: instance→relation
example: **fookunl→instrumentr**
purpose: locality principle, used in semantic update (§6.4.3)
provenance: created by `#'plug` (§6.2)

`rel-collector` from/to: relation→notion
example: **patientr→pat-ness**
purpose: inherited to `irel-collector` when the input is assembled with `#'plug` (§6.2)

`irel-collector` from/to: notion→notion
example: **jonnl→pat-ness**
purpose: mostly for preposition choice
provenance: created by `#'plug` (§6.2)
special properties: subject to the maximum rule (§4.2.4)
weight: 1. if generating English, .5 if Japanese

`ifillers` from/to: rel→instance
example: **patientr→jonnl**
purpose: part of a path from constructions to elements of the input
provenance: created by `#'plug` (§6.2)
weight: always 1.

`ifillees` from/to: rel→instance
example: **possessor→momonl** (peach)
purpose: the locality principle (§6.4.3)
provenance: created by `#'plug` (§6.2)

Links from pseudo-nodes. (Links from **in-progressx**, **coveredx**, and **in-focusx** are changed while FIG is running. These are the only links created or altered by FIG.):

`from-sourcex` from/to: **sourcex**→instance, notion
weight: always 16 (so that concepts in the input initially have 20 units of activation)
purpose: activate the nodes of the input (§6.5.1)

`from-coveredx` from/to: **coveredx**→instance, notion
weight: always -100
purpose: inhibit concepts already expressed
provenance: created by the semantic update process (§6.4.3) nodes

`from-in-focusx` from/to: **in-focux**→instance,notion,relation
weight: 6 to ifillers of covered relations;
6 to ifillees of covered concepts;
12 to inebors of covered concepts;
12 to irel-collectors of covered concepts if English, 18 if Japanese
(if generating Japanese **topicn** counts as a relation for this purpose)
purpose: the locality principle
provenance: created and destroyed by the semantic update process (§6.4.3)

`from-in-progressx` from/to: **in-progressx**→construction
weight: always 9
purpose: represent the current syntactic context (§4.2.2, §6.4.2)
provenance: created and destroyed by the syntactic update process (§6.4.2)

`from-permanentx` from/to **permanentx**→ syn-cat, relation
weight: varies
purpose: activate nodes which are always relevant (§6.3)

Other node properties:

`express` from/to: word→notion
example: **peachw**→**momon** (the peach concept)
purpose: for semantic update (§6.4.3)

`persistentp` from/to: notion→t, nil
example: **fairy-talen**→t
purpose: a flag to the semantic update mechanism (§6.4.3)

`grapheme` from/to: word→string, list of string-node pairs
example: **johnw**→ *"John"*
purpose: used by the inflection mechanism (§6.4.1)

## 6.7 Size and Speed

FIG's current knowledge network, shown in the Appendix, has roughly 400 nodes and 1400 links. FIG is approximately 240K bytes of source code. Table 6.2 shows the specifics.

One would not expect a parallel program simulated on a single-processor machine to be fast. Yet FIG is actually quite speedy. This is because the computation for each node is so simple and there is very little computation of any other type, and in particular, none of the storage-intensive structure-building present in most generators. FIG outputs a word about every .31 seconds on average on a Symbolics 3670. It is twice as slow on a Sun SPARCstation 1+ and 8 times slower on a Sun 3/140.

The per-word time is roughly proportional to the product of the number of nodes which become activated and the number of cycles it takes for the network to settle.

| Bytes of Heavily-commented LISP | | |
|---|---|---|
| Basic Program | 60K | |
| data structure definitions | | 9K |
| spreading activation processes | | 15K |
| special processes (inflection, update) | | 22K |
| network building code | | 14K |
| Network | 49K | |
| The Japanese Parser | 33K | |
| parser code | | 21K |
| knowledge | | 12K |
| Test Corpus | 30K | |
| Conveniences | 48K | |
| regression testing tools | | 14K |
| input/output routines | | 12K |
| window interface definition | | 22K |
| Other | 18K | |

Table 6.2: Size of FIG as a LISP Program

The former is roughly proportional to the number of nodes in the network, and the latter is roughly proportional to the longest useful path of activation flow. Since FIG is to some extent head-driven, the longest number of cycles required tends to be in cases where a head has to activate a construction which puts a word in front of it, for example, nouns activating articles via **determinatc**. To examine this I gathered some statistics on the number of cycles required to settle as a function of next word output: the averages were: 12 cycles before articles, about 9 cycles before proper nouns, and about 6 cycles before common nouns and adjectives, with 7–10 common for most other types of words. Also, the average cycles required to settle before the first word of an utterance is 14.0 cycles, while the average before subsequent words is 7.6 cycles. The longest time required to settle is about 25 cycles.

If FIG were running on a parallel machine with each node assigned to a processor I would expect the per-word time to be proportional to the longest useful path. The longest useful path in FIG-as-is is about 7 links; this is in part because there is no meaningful semantic computation, but I speculate that even an enormous network would not have much longer useful paths.

# Chapter 7

# Connectionism: Why and How



Connectionism is all the rage. This chapter considers ways in which connectionist modeling techniques can be used in the study of natural language. In the process I will explain why I built FIG as a system straddling the distinction between symbolic and connectionist approaches. I will also criticize other connectionist generators; this is not to be negative, but only because previous chapters have already mentioned their good points.

In this chapter I will refer to FIG as a 'structured connectionist' system (Feldman *et al.* 1988). I could equally well continue to use the term 'spreading activation', but 'structured connectionist' stresses that FIG is built using a modeling technique that is intermediate between traditional symbolic techniques and parallel distributed processing (PDP) approaches.

## 7.1 Symbolic versus Connectionist Models

There are several reasons to use connectionist models in general, including similarity to the brain, and various computational advantages, such as speed, robustness, and learnability (Feldman *et al.* 1988; Rumelhart *et al.* 1986).

The reason which was most important in the development of FIG is that connectionism can open up new ways to approach a problem. Much work in AI has continued in the molds set over 30 years ago. For example, generators were traditionally designed around a module diagram, a flowchart, or some pseudo-code, and were based on the creation and manipulation of structures (§2.4), and this style of design continued until 1987, largely because these ideas were the only ones available. Connectionism allows one to explore new approaches. This may be useful for scientific reasons, in particular it can provide a new descriptive vocabulary for linguistics and psychology and ultimately neurophysiology. New approaches are also worthwhile for the practical reason that they may lead to designs which are easier to implement on parallel hardware.

To take full advantage of the opportunities offered by connectionism it is necessary to be willing to use new designs and take new perspectives on the problem. Most connectionist generation research has not done so. By and large, it has accepted without challenge most of the research priorities, vocabulary, and mechanisms developed within traditional linguistic and AI models. The next three paragraphs illustrate this.

There have been many demonstrations that connectionism is no less powerful than symbolic processing. For example, Kalita re-implemented a subset of the MUMBLE generator (McDonald 1983a) in a connectionist network (Kalita & Shastri 1987). Stolcke built a neural network with the same behavior as a unification algorithm (Stolcke 1989); it then was able to generate *"Peter loves Mary"* and *"Peter is loved by Mary"*. (Rager & Berg 1990) implemented government-binding theory in a connectionist network. Success in such endeavors is not surprising — any good computer scientist can implement any arbitrary algorithm with any arbitrary technology. In particular, it is always possible to implement any theory with neurons (since it is possible to build a Turing machine out of NAND gates). Such work is unlikely to tell us anything new, nor is it likely to take advantage of any of the interesting properties of connectionism, such as emergents.

There is also some research which applies connectionism to sub-problems. The resulting systems are 'hybrids', conventional except in one module or for one algorithm. For example, Tomabechi (1987) used connectionist techniques to address some issues in word choice as they arise in the context of machine translation. If there was a concept used in the input for which the target language had no word, his system found a way to express that concept. It did so by searching for a concept which can be expressed with a target-language word; this search was performed by passing markers across the 'isa' links of the conceptual hierarchy. There was no provision for other kinds of paraphrase, and the system is otherwise quite conventional; in particular, the process of lexical choice can not affect the syntactic form of the output. In fact, the paraphrase-words found were apparently only ever used in appositives, rather than being blended into the text. Klöck (1988) similarly only addressed one issue, that of choosing the verb whose valence best matches the configuration of cases to express. Such work may help patch up existing systems but are again unlikely to lead to new ideas.

Even systems which are are not directly based on standard approaches tend to carry a lot of baggage from symbolic models. For example, the systems of Gasser (Gasser 1988) and Kitano (Kitano 1990), although original in many respects, are still largely top-down, serial, and structure-bound. In particular, in CHIE nodes fire relatively infrequently; the sequence of firings often appears to amount to parallel simulation of a serial algorithm.

Thus most connectionist research in generation (indeed in natural language in general) accepts a lot of conventional wisdom, to the extent of ruling out the possibility of discovering anything new. As witness to this fact, although most of the kinds of parallelism in §2.3.3 have been advocated in one form or another, no one has understood the ways in which they are required by the generation task. In contrast I have tried to take the constraints and opportunities of connectionism seriously. As a result I have learned specific reasons why connectionism is appropriate for generation (§2.3), and have discovered that many of the basic assumptions of previous generation research are ill-suited to or unnecessary in a connectionist approach.

In general, I suggest that a radical re-thinking of the nature of the problem under study is perhaps necessary to build a good connectionist model. However the need for new thought is deprecated by some — many hope for a clever way in which symbolic approaches will convert nicely to connectionist systems. Indeed, a great deal of effort has gone into the implementation of low-level mechanisms sufficient to allow symbolic models to be ported to run in connectionist network, thereby enabling old AI programs to be recycled into robust, trainable connectionist systems. Mechanisms so far attacked include variable binding, inheritance, instantiation, recursion, structure matching, and structure building (Touretzky & Hinton 1985; Ajjanagadde & Shastri 1989; Tomabechi & Kitano 1989; Pollack to appear). Most of this work has been inspired by the perceived needs of natural language processing. While it is possible that the existing inventory of connectionist mechanisms and idioms is inadequate, I believe that one should try to do as much as possible with the existing mechanisms. That is, extending the connectionist model to make it more like LISP should only be done as a last resort — only when it has been shown that the nature of language demands such mechanisms. The task for connectionists is to re-consider the actual phenomena of interest and find new, non-symbolic ways to handle them. Using ideas borrowed from old symbolic approaches is antithetical to making new discoveries. Self-discipline as to what to borrow from traditional accounts will enable, and force, us to discover new ways to solve old problems.

Apart from the desire to be able to directly use symbolic theories on connectionist nets, such efforts are also driven by the sniping from various philosophers that 'a connectionist model cannot handle X', where X is recursion, instantiation, variable binding, structure building, or the like. These mechanisms are often assumed to be essential to cognition or language. For example, the idea that 'natural language is recursive' is ingrained in many views of language. Similarly the use of syntactic 'structures' has been considered essential. But these are not characteristics of language; they are descriptive mechanisms used by linguistic theorists. They are undeniably useful for building simple, practical natural language systems, but they are not *necessary* for the processing of language. I conjecture that, for any real phenomenon, even if it has been previously handled with symbolic technique X, there exists an alternative, connectionist way to handle it. For this reason the connectionist researcher can and must reexamine previous theories. This may consist of reconsidering the raw data or re-considering the larger task being addressed.

The things that are easy for connectionism are different from the things that are easy for symbolic models. For example, interactive systems are easier to build using connectionist techniques, but variable binding is harder to handle. In general, it seems that the things easy for connectionism are easy for people. The things that are harder to handle, like recursion, are also harder for people. For example, people can only handle a very limited amount of center embedding in language, showing that they cannot handle recursion in general; and it has been shown that people cannot handle the general problem of instantiation, although they perform well in situated tasks (Kanwisher 1990). In general, connectionist technology suggests a different research agenda, one which is probably more appropriate for cognitive science. Connectionist generation researchers should not uncritically accept standard priorities, such as attacking sentences with grotesquely embedded relative clauses (Miikkulainen 1990), or claiming to handle 'constraint equations' or 'unification-based processing' (Kitano 1990).

This section has so far argued that connectionist research should not follow blindly the path laid down by symbolic researchers. But this is not to imply that symbolic and connectionist researchers have nothing to learn from each other. In fact, there is not a strict dichotomy between connectionist and symbolic approaches; rather, there is a continuum of system-building techniques, ranging from standard symbolic systems, through object-oriented systems, marker-passing systems, and local connectionism, to distributed connectionism. Moving towards the connectionist end of this continuum, the grain-size of parallelism gets finer, that is, the computational elements get simpler and generally more numerous.

Moreover the same information must be computed regardless of modeling technique. Thus, ideas developed using one approach may be useful to people using other modeling techniques. In particular, the 'characteristics desired for generators' discussed in Chapter 2 are relevant to all generators, regardless of the technology on which they are built. Structured connectionism is at a privileged position in the continuum of techniques; being in the middle, it can lead to ideas relevant to more extreme positions on the scale.

## 7.2 PDP versus Structured Connectionist Models

One way to do connectionist research is to model the brain as closely as possible. A problem with this is that the brain offers enough computational mechanisms to build models which are completely neurally motivated but utterly confusing. For example, the nodes of Gasser's connectionist generator each had a firing threshold, a rate of decay, a refractory period, and a recovery amount (Gasser 1988). The resulting model is too complex to understand — it is not clear what each of these parameters is being used for, or indeed whether they are anything but ad hoc (the latter seems indicated by the fact that some parameters are tuned, by hand, to three significant digits). For example, even to encode simple grammatical information seems to require many unmotivated numbers; CHIE's definition of noun-phrase has 16 numbers.

A more common connectionist technology is PDP. This does not mimic the brain so closely. Indeed, some of the mechanisms are neurologically implausible. For example, activation flow is typically in terms of continuous quantities even though real neurons fire too infrequently to transmit floating point numbers. On the other hand, these models are clean and comparatively simple to analyze. They also have nice computational properties, notably robustness and learning with automatic generalization.

Unfortunately it is currently impossible to build an interactive PDP model of a complex task like language generation. This is because existing learning mechanisms are simply not powerful enough — they can only handle single aspects of language, and then only if the input is carefully pre-processed. This is why only 'toy problems' have been handled by so many connectionist experiments, especially those which deal with time-dependent behavior. Indeed, to build a PDP model of generation it seems necessary to make many sacrifices, judging by the work of Miikkulainen (1990). To make learning tractable he had to introducing modularity, which is a pity, since use of modules restricts the interactivity that seems so important to language and which is an important advantage of connectionism. Also, his system treats generation as a simple process of mapping, can only deal with fixed-width inputs, and 'cannot generalize into sentence structures it has not seen before'.

Spreading activation systems are less like the brain but they allow one to leap over the learning barrier which limits PDP research. Since there is one node per concept, they are buildable, understandable, and debuggable by humans, which allows relatively fast development of relatively large prototypes.

Prototyping and experimentation are important. In general, I doubt that any one breakthrough in connectionism will solve all the problems of AI. For example, the network structures needed for language may well differ from those needed for vision or coordinated motion. Assuming this is the case, it is necessary to do detailed, domain-specific research, and this can probably best be done by eclectic, exploratory research. Researchers should feel free to choose from the connectionist palette whichever colors are useful to paint a model and perform experiments, including parallel computation, knowledge represented by the topology of connections, information passed as numbers, state of each object as a number, simple output functions for the objects, purely local computation, distributed representations, emergents, integration (doing without modules), and numeric combination of evidence. Whichever of the these do not appear useful should be left out; a system should not be judged on the basis of its purity as a connectionist system. The goal of such experimentation should be to learn what is and what is not necessary for the task. For example, FIG shows that some kind of moving focus of activation is useful, and that structure-building is not necessary. After the domain is well understood it should be easier to build a PDP model with the same or better behavior.

Thus there is a place for spreading activation models. But when building such systems it is important to remain faithful to the basic ideas of connectionism, otherwise the results will be irrelevant to other systems. This is a real danger — it is easy to build a system based on nodes in a network but which on close inspection turns out to be fundamentally incompatible with

| | Kalita | Gasser | Kitano | FIG |
|---|---|---|---|---|
| parallel computation | Y | Y | Y | Y |
| knowledge represented by connections | Y | Y | Y | Y |
| information passed as numbers | Y | Y | N | Y |
| state of each object is simple | Y | N | N | Y |
| objects' output functions are simple | Y | Y | N | Y |
| purely local computation | N | N | Y? | N |
| distributed | N | N | N | N |
| emergents | N | N? | N | Y |
| integrated (uses world knowledge) | N | Y | Y | Y |
| no binding or structure building | N | N | N | Y |

Table 7.1: Connectionist Qualities of Structured Connectionist, Marker-passing, and Spreading Activation Generators

connectionism. One possible incompatibility is requiring computation which intrinsically cannot be done with purely local computation, another is requiring a system to store information at run-time in a way that intrinsically cannot be handled by changing activation levels. An example of something which fails both these tests is the idea of binding concepts to syntactic slots. This requires non-local computation — the name of one node must be communicated to another. It also requires a structural change to the network (creation of a link) which cannot be modeled simply by changing an activation level.

Of course, whether or not some mechanism is 'fundamentally incompatible with connectionism' is not always obvious. Indeed, Tomabechi and Kitano have argued that non-local computation, specifically marker-passing, is justified, since it can be modeled by using the frequency of neuron firings (Tomabechi & Kitano 1989). Gasser uses simultaneous node firings to determine which nodes to bind together; to record the bindings he uses the primed state of neurons (Gasser 1988). Such proposals, while clever, are not well supported by physiological data.

Both of these proposals make use of the time dimension to implement generally accepted processing modes. FIG also uses the time dimension, being incremental, and it relies on this in several ways, for example, in using the locality principle to avoid problems of crosstalk (§4.2.3). This does not require any complex behavior by nodes, and has the advantage of being directly related to and required by the generation task, rather than being a completely general mechanism.

It is interesting to note that FIG exploits rather more of the techniques in the connectionist toolkit than any similar generator. Table 7.1 summarizes the key connectionist properties of the three generators which most closely resemble FIG in motivation and design. (The question marks represent things which are unclear from the published descriptions.)

However, FIG is connectionist because it has to be (§2.3); the parts which do not have to be connectionist (at least not for the first prototype) are not. The next few paragraphs explain why none of FIG's 'special processes' (§6.4), make it fundamentally incompatible with connectionism.

To decide what word to emit FIG searches for the most highly activated word representing a node; it is not obvious how best to handle this with spreading activation, but a simple solution would be to augment FIG with an 'output network' for this purpose. After a word is selected, a special process choses the correct inflection. This would be unnecessary if FIG were extended to handle morphological and phonological processing correctly. Indeed, Dell has implemented a network which does some of this (Dell 1986).

To determine when to advance the cursors of syntactic constructions FIG uses a matching process; this mechanism could trivially be replaced with a small sub-network with events as inputs and syntactic effects as outputs. As a result of this syntactic update process, FIG changes the weights of the links from constructions to constituents, depending on the current position of the cursor. This could be implemented by just adding extra nodes to 'gate' these links. FIG performs inference for the sake of semantic update; the current inference algorithm could easily be implemented with spreading activation, given some revision to the evidence-combining function, for example, so that nodes receiving negative activation across **express** links would necessarily have zero activation.

Thus most of the non-connectionist parts of FIG could easily be made connectionist. There are however some non-connectionist aspects which it is not clear how to do away with, namely those relating to the representation of the input: FIG uses links among instances of concepts to represent the input. These links must be set up for each input, which is not really satisfactory. Moreover, the flow of activation among these nodes is regulated in complex ways, as discussed in §6.5.1. Finally, the activation levels of the nodes of the input are manipulated by creating and altering links from pseudo-nodes to them. These problems are instances of the more general problem of how to represent structural information in short-term memory, and how to make it exhibit a focus of attention. No connectionist system yet has a satisfactory solution.

I would like to build a PDP system with the same behavior as FIG; I am convinced that PDP models are eventually the right way to go. This is in part for the sake of learning — that is, avoiding the problems of adjusting weights and of deciding what the network structures should be. Partial generalizations (sub-regularities) are important in language (Wilensky 1990), and PDP models are also good at capturing these (Rumelhart *et al.* 1986; Lakoff 1988).

However none of the weaknesses of FIG are things that would disappear simply by re-doing it as a PDP system. Also, it is far from obvious how to build a PDP system to emulate FIG's behavior, so building a powerful PDP generator will have to wait for further progress in PDP techniques, especially for the learning of time-dependent behaviors.

So convinced am I that FIG is only an experiment that I have made no effort to believe in the correctness of the ontology of the network, or indeed that it is possible to come up with a correct ontology at all. What I do believe in is my characterization of the computations that the network must perform and the functionality it must exhibit, as discussed in Chapters 2, 3, and 4.

# Chapter 8

# Psycholinguistic Evidence



Common sense about how people generate has been an inspiration in the design of FIG — for example, it is incremental and parallel. Although detailed facts about human language production were originally not considered, recent examination of the psycholinguistic literature reveals much interesting data which bears on the theoretical claims of this thesis. Much of the data can be seen as supporting FIG, or, more precisely, supporting some of the design characteristics listed in §2.3.

This chapter does not mention evidence on aspects of the generation task beyond the scope of FIG, such as evidence on the importance of phonology in lexical access, on the handling of inflected forms of words, on the role of stressed syllables in speech errors, and on pause distributions in discourse. This chapter is not even a comprehensive survey of evidence on the syntactic and lexical aspects of production — much of this data is meaningless outside the narrow theoretical framework within which it was gathered and analyzed.

## 8.1 Introspection

"If it were asked 'Do you have the thought before finding the expression?' what would one have to reply?" asks Wittgenstein (1963). In psycholinguistics, as in artificial intelligence, the usual answer is "yes". More specifically, generation is often considered to be simply a process of encoding a pre-existing thought or 'meaning', as discussed in §2.2.2. Yet writing, at least, is commonly considered a creative process. Similarly, after talking about something you may look at it differently or understand it in a different way. Introspection thus suggests that generation creates meaning.

Further, it seems that the language being spoken can catalyze the creation of meaning. In the course of speaking one can find, along with words, 'conceptual frames' and ways to think about the scene or situation. Also language sometimes constrains the speaker to choose how to think about something; a good example is the choice between singular and plural for nouns in generic statements, such *"speaker"* in the previous clause.

I do not mean to say that language constrains thought, or that there can be no thought without language. Nor do I mean to suggest that people do not think at all before they speak. However the tendency in the field has been to ascribe too much to pre-verbal thought. That is, instead of just assuming that speakers have some idea of what they want to say before they start many generation researchers seem to assume that people have thoughts organized conveniently into feature-structures or logical forms before they start to speak. Based on my introspection this seems excessive; experimental evidence, discussed below, agrees.

Other researchers emphasize that thought can develop during the process of speaking, but do not allow generation any causal role in that process, nor do they even allow it access to the evolving thoughts (Kempen & Hoenkamp 1987; Levelt 1989; De Smedt 1990). That is, these researchers assume that the generator's input arrives one chunk at a time from the thought process, and that the generator has no access to nebulous or evolving thoughts. Certainly remindings and intention-changes do exist, and the idea that the input need not be completely well formed and present at the beginning is a good one, but introspection suggests that at least a vague notion of what is coming next is usually available. In Wundt's words, "a sentence is both a simultaneous and a sequential structure" (Wundt 1900). Vygotsky (1962) argues that "thought, unlike speech, does not consist of separate units. When I wish to communicate the thought that today I saw a barefoot boy in a blue shirt running down the street, I do not see every item separately: the boy, the shirt, its blue color, his running, the absence of shoes. ... A thought may be compared to a cloud shedding a shower of words". These ideas agree with the notion of part-wise parallelism.

On the other hand, there clearly is some sense in which thought is linear and correlated with the words being output. In Wundt's words, the content of thought "changes from moment to moment ... as individual constituents move into the focus of attention and out again one after the other" (Wundt 1900), and in Chafe's, "spurts of language ... provide us with excellent evidence of how consciousness successively activates small chunks of information". (Chafe 1980b). These introspections support FIG's locality principle.

One thing that introspection often fails to reveal is that language production is hard. Speakers usually manage to get their point across, and hearers automatically edit out disfluencies. But careful listening to almost any spontaneous speech will reveal pervasive pauses, repetitions, and self-corrections; these will be discussed below. In general, people are good at producing adequate language but seldom do so optimally.

## 8.2 Pauses

There has been a vast amount of research on pauses in speech (O'Connell & Kowal 1983). Pauses have many functions. Some pauses contribute to prosody, and so convey meaning or mark clause boundaries. Some pauses serve discourse functions: for example, to indicate deference to or fear of the hearer, to allow the hearer to understand or to signal understanding, to allow the hearer a chance to take a conversational turn, or to pretend to the hearer to be doing any of the above. Some pauses provide time to monitor one's own speech.

Once all these things have been discounted, there remains the fact that some pauses are clearly involuntary and low-level. That is, some pauses are symptoms of the need for more time to formulate something. Other symptoms of this need for time include lengthened syllables, repetitions, and filled pauses, such as *"um"*, and *"uh"*, which also have the pragmatic role of staving off interruptions.

The most important fact about pauses is their existence. In Chafe's words, disfluent speech in general and hesitation phenomena in particular "provide good evidence that speaking is not a matter of regurgitating material already stored in the mind in linguistics form, but that it is a creative act ... The fundamental reason for hesitating is that speech production is an act of creation" (Chafe 1980b). This interpretation is supported by the fact that subjects reading aloud pause less than subjects speaking spontaneously (Rosenberg 1977).

What processing happens in pauses? Tip-of-the-tongue situations are clearly a problem of lexical access, but in general introspection tells us little. The best that can be done is to is study the distribution of pauses: that is, their occurrence and duration as correlated with other measures of language or thought. Examples of such research include studies of the correlation between pauses and syntactic configurations, interpreted as showing which configurations are harder to process (Holmes 1988); and studies of the correlations between pause type and length and the type of word that follows the pause, interpreted as showing that some kinds of decisions or lexical accesses are more difficult than others (Maclay & Osgood 1967; Butterworth 1980). While undeniably important, the significance of these studies is unclear in the absence of a clear notion of how lexical and syntactic processing is done.

Such studies are complicated by the fact that there is no reason to think that planning cannot be simultaneous with uttering. Thus, if a speaker pauses before a word, it is not possible to ascribe that pause to the need to retrieve or choose that particular word or the concept associated with it, since he may have been planning ahead. A further complication is the fact that any given pause may serve multiple functions (Rosenberg 1977), including simply allowing the speaker to inhale.

However there is one set of experiments carefully controlled to enable certain pauses, or portions thereof, to be ascribed to specific tasks. Lindsley (1975) showed pictures to subjects, instructing them to describe what they saw, using a two-word sentence with a subject and a verb, the SV pattern, or alternatively, to use SVO, S, or V patterns. In each case he measured the latency, that is, the time from the presentation of the picture to the onset of speech. One finding is that SV latency is greater than S latency. This implies that some of the work of choosing the verb takes place before production starts. In my terminology, this is evidence for part-wise parallelism. A second finding is that if S is known beforehand, then SV latency is greater than V latency. This implies that some of the work of choosing the V takes place after production starts, or in other words, speech is initiated before all 'planning' is completed. This is also supported by the fact that the latency for an SVO utterance is no longer than that for an SV utterance, or, in general, "the amount of advance planning of utterances is to some degree independent of the number of independent units comprising the utterance" (Lindsley 1975). This shows that generation is to some extent incremental.

Pauses appear to cluster; conversely speech exhibits stretches of relative fluency. It is hard to say anything definite about this, but there are many interesting speculations (Chafe 1980a).

## 8.3 Priming Effects

Lexical priming has been extensively studied. A pervasive finding is that priming of semantically related items can speed up output of words, for example in picture-naming tasks (Aitchison 1987). Such priming suggests, of course, an associationist or spreading activation model of memory.

Some researchers have attempted to go further, to experimentally investigate the time course of the lexical access process and the organization of memory and the lexicon. Such studies are difficult; they require clever experimentation and detailed, quantitative analysis of lexical access times (Levelt 1989). There seem to be no solid conclusions except those regarding phonological priming.

Syntactic constructions can also be primed; Bock showed that the probability of a passive being used to describe a picture increases if the speaker has previously produced a passive (Bock 1986). This suggests that syntactic constructions are handled with activation similarly to words.

Priming evidence also bears on the interaction between 'levels'. In general, "evidence from sentence production, sentence recall, and word processing experiments indicates that the retrieval of words during sentence formulation influences sentence form, partially independent of the sentence's intended substance" (Bock 1982). This shows that top-down, syntax-first, interaction-free models of generation are not cognitively plausible.

A specific example of this is the fact that, if a word is more "accessible", either due to phonological or semantic priming, it tends to be the subject (although the details are complicated) (Bock 1987). In FIG there are two possible ways in which lexical accessibility can affect syntactic form. The first is that after a word is emitted the syntactic update mechanism activates appropriate constructions, and those constructions thenceforth help guide production. The second way is that activation flows from concepts via the relations they fill to syntactic constructions. For example, if a patient is highly activated, then, thanks to activation via **pat-ness**, (which is the collector for **patientr**, the relation filled by the concept) the passive construction can be used. This is precisely how FIG produces *"Mary was killed by John"*.

## 8.4 Errors

People produce many sub-optimal sentences, for example, those involving poor choices of words. Unfortunately no one has studied this. The syntactic aspects of generation are also not perfect; people sometimes use constructions in contexts where they are not quite appropriate, as in *"it was all told me about"* (presumed target *"I was told about all of it"*, on the pattern of *"that was all decided years ago"*) (Stemberger 1982). Syntactic 'choice' has the additional problem that it requires 'lookahead'. That is, a speaker can 'talk himself into a corner' by failing to "evaluate the linguistic consequences of all [his] goals taken together", as in *"it's not a street that you can use to find a street that you don't know where it is"* (McDonald 1983a) and Example 2.6. The existence of such errors is evidence that speakers do not plan very far ahead, or, in other words, that speech is incremental.

Sometimes people continue with a faulty plan. More often they correct it — resulting in a false start. In both cases, failure indicates that what to say and when to start saying it are computed by different processes. That is, many false-starts can be considered mis-judgements of whether it is safe to initiate output. This is modeled by FIG, where the decision as to when the network has settled is separate from the computation of what needs to be said.

Detection of failures is evidence also for monitoring processes. Things speakers check for include failures of lexical access, ungrammaticality, and undesired effects on the hearer (Clippinger 1977). The existence of monitoring processes has also been shown with experimentally elicited errors (Motley *et al.* 1983). It is not clear to what extent monitoring is a separate process or just feedback within the network (Stemberger 1983).

Parenthetically, it is interesting to note that many pauses probably have the same cause as many false starts. If the processes of monitoring and editing are fast they can suppress a word before it is actually uttered, resulting in a pause; and if not then some words will be emitted before the speaker corrects himself, resulting in a false start. There is also experimental evidence to support this interpretation (Levelt 1989).

Most helpful as a source of insight into the human process of language production are errors where it is clear what the speaker intended to say instead. That is, there are cases where the intended (target) utterance and the actual utterance can be compared. From this it is sometimes possible to infer what characteristics of the generation process make it error-prone in the way observed. Of course, this methodology is not without risks: the minimal correction to make a valid sentence may not reflect the actual etiology of an error, and for certain errors it is not even clear that there was a clearly intended target.

Well known are cases where a wrong word substitutes for the intended one. The wrong word is typically appropriate for other reasons, such as being relevant to 'subconscious' thought, as Freud claimed (Dell 1989). A more recent study has shown that almost any factor of context or the environment at large can affect lexical access. For example, the utterance *"I'm going to read some yogurt"* was produced where the previous context was a discussion of literature (Harley 1984). Similar conclusions have been achieved by studies of experimentally-elicited errors (Motley *et al.* 1983). Thus there is no isolable and small 'input'; the generator is not an autonomous process, as argued in §2.2.1. Another substitution error is *"I forgot to listen to the picture of Greg"* (presumed target *"... look at ..."*) (Stemberger 1985b). This illustrates the tendency for slips to result in the output of words which are similar in meaning to the intended word or to other words in the context. This is more evidence for an associative network.

Errors are subject to the constraint that "when a wrong word is accessed ...it is almost invariably of the same part of speech as the target word; noun for noun, verb for verb, etc" (Stemberger 1985a). This tendency is weaker when the intruding word would be appropriate elsewhere in the the utterance (Stemberger 1985a). This can be explained as follows: for a nontarget word to be accessed, it needs every possible source of activation including syntactic activation. A word appropriate elsewhere, however, receives semantic activation, and so it may be able to be selected even without syntactic activation — that is, even if it is of the wrong syntactic category, as observed by (Stemberger 1985a). This is consistent with a model where syntactic activation and semantic activation are both just factors in scoring the relevance of words, rather than being absolute constraints, as in FIG (§3.5).

*"Sky is in the sky"* (presumed target *"sun is in the sky"*) (Dell 1986) is an 'anticipation error'; here a word which should come late in the sentence also appears earlier. The existence of anticipations shows that generation is not completely incremental; words for 'current' and 'future' positions are active simultaneously; using my terminology, there is part-wise parallelism.

*"this ice cream has an interesting flaste"* (presumed targets *"flavor"* and *"taste"*) (Stemberger 1985a), and *"upcited"* (presumed targets *"upset"* and *"excited"*) (Garrett 1975) are examples of 'lexical blends'. These suggest competitive parallelism among words. The two blended words are usually of the same part of speech (Levelt 1989), but not always. There are also examples such as *"I haven't found a thingle yet"* (presumed target *"...a single thing ..."*) (Stemberger 1985a) This example suggests part-wise parallelism, in that the anomalous item is a blend of two words which should have appeared in sequence.

*"Do you get ice cream"* (presumed targets *"do you feel like some ice cream?"* and *"why don't we get some ice cream"*) (my observation) is an example of a syntactic blend. Stemberger (1982) has many further examples. These examples suggest the existence of competitive parallelism among constructions.

The next few paragraphs consider evidence for the existence of separate update processes, by examining how people continue after emitting a wrong word.

*"Class will be about discussing the class"* (presumed target *"...discussing the test"*) (Dell 1986) is an example of a perseveration. These are considered to result from failure of the semantic update process to zero out the activation of the concept associated with an emitted word.

*"Writing a mother to my letter"* (presumed target *"writing a letter to my mother"*) (Dell 1986), and *"beating your brick against a head wall"* (presumed target *"beating your head against a brick wall"*) (Garrett 1975) are examples of 'exchanges'. These are considered to result from

an anticipation after which the semantic update process correctly zeroes the activation of the expressed concept. The fact that exchanges exist illustrates the presence of a separate semantic update process. If there were no such process there would be no reason that one error would cause another. Also, this shows that the 'assignment' of concepts to syntactic slots is not complete before generation starts, which is more support for the idea that generation is incremental. Exchange errors where the exchanged words are close are more common than those where they are more distant. This suggests that the entire utterance is not built up before generation starts, but rather that generation is incremental, with a relatively small focus of activation.

*"Most cities are true of that"* (presumed target *"That is true of most cities"*) (Stemberger 1985a) illustrates 'accommodation' in that the verb is *"are"*, not *"is"*, to agree with the subject. This is evidence that a syntactic framework is not chosen first; more generally, it is evidence against top-down, syntax-first generation. Rather, agreement seems to be handled by interactions among words.

*"I just wanted to that"* (presumed target *"I just wanted to ask that"*) (Dell 1986) is an example of a 'deletion'. Using FIG terminology, this is simply an anticipation after which the syntactic and semantic update processes update the situation as if the correct word had been output.

*"The only one thing ..."* (presumed targets *"the only thing I can do"* and *"the one thing I can do"*) (Dell 1986) is an example of an 'addition' of a word. An inappropriate word is selected, but the update processes do not update the situation, and the right word then comes out next. This error could alternatively be explained as a syntactic blend, that is, as being due to the simultaneous activation of different constructions (competitive parallelism) or competing 'plans' (Baars 1980).

*"Iowa's very important in Indiana, apparently"* (presumed target *"basketball's very important in Iowa, apparently"*) (Stemberger 1985a) is an example of a rare type of error apparently involving an anticipation, after which the semantic update mechanism correctly zeros the activation of the expressed concept, and then a semantically-primed concept steals into the vacant place. Another example is *"I got a paper on my test"* (presumed target *"I got an A on my paper"*).

*"You went to the same bathroom at the time"* (presumed target *"...to the bathroom at the same time I did"*) (Stemberger 1985b) is a 'crosstalk' error in that the adjective is associated with the wrong noun. Such examples, while rare, give some support to the separation of syntactic and semantic activation, and against the idea of structure-building. That is, here the adjective appears in a place which is syntactically appropriate but semantically inappropriate. Interestingly, FIG has a tendency to make errors of this kind, due to the difficulties of adequately implementing the locality principle §6.5.1.

*"The flood was roaded"* (presumed target *"the road was flooded"*) is an example of 'affix stranding', the phenomenon where "words are misordered, but usually leave associated inflections behind and take on the inflections of the words that they replace" (Stemberger 1985a). This also happens for determiners, as in *"there's a beard hanging from your hair"* (presumed target *"There's a hair hanging from your beard"*) (Stemberger 1985b). The significance of stranding is that it "implies that lexical items and inflectional categories are accessed separately and in parallel. Any failure on one of these two aspects of lexical access is usually not accompanied by failure on the other" (Stemberger 1985a). In FIG terms, stranding shows that grammatical markers are positioned by constructions. Stranding is also evidence against the view that words are assembled together into syntactic structures for constituents.

Speech error data have been used in many arguments. For example, some facts appear to suggest different processes for retrieval of open and closed-class words; but Stemberger's re-analysis shows that this assumption is not necessary to explain the data (Stemberger 1985a). Some have used error data to argue for interaction-free 'levels' of processing; but Bock has shown that the data do not support this (Bock 1987). The fact that some errors appear to involve wrong words and others the right words in the wrong order has led some to the claim that there are separate mechanisms for word choice and word ordering; but Stemberger shows the explanatory adequacy of "a single process which both accesses lexical items and establishes their serial ordering" (Stemberger 1985a), thus the data are compatible with a model where word order is emergent, as in FIG.

Data from aphasic subjects (those suffering language loss due to brain damage) have often been taken to show "dissociation of syntactic and lexical aspects of production" (Saffran *et al.* 1980), suggesting that syntactic knowledge is stored in special place in the brain or in encapsulated in a component. Given the vast amount of data on aphasia and an enormous number of ways to analyze it, this traditional conclusion cannot be accepted as proven. In fact, more recent analyses suggest that at least some aspects of aphasia differ only qualitatively from disfluencies in the speech of normals, and moreover that certain aphasic behaviors are simple to account for with spreading activation models (Stemberger 1984; Harley 1990).

## 8.5 Building Cognitive Models

This chapter has ignored the tradition of attempting to verify the reality of the representations and processes postulated in linguistics. It has also ignored research on human knowledge structures, such as the 'mental lexicon', of which overviews can be found in (Aitchison 1987) and (Schreuder & Flores d'Arcais 1989); I have not discussed this because FIG is a process model. In general, the best way to study human language processing seems to be to study the process itself, rather than to perform experiments which are not ecologically valid, or to reason tenuously from conclusions about other processes, such as parsing.

Speech errors are a much richer source of information. Early work in the modeling of speech errors explained them directly in terms of a speech 'machine' and its phases, levels, modules, stages, buffers, paths, planning units, and so on (Garrett 1975; Levelt 1989). Only recently have there been attempts to explain errors more deeply, in terms of emergents, that is, "phenomena that may not need to be encoded into the system, but that emerge automatically from the way the system is put together" (Stemberger 1985a).

Dell (1986) built a spreading activation model to account for slips of the tongue. It has activation from lexemes to their component phonemes and back from phonemes to lexemes, and inhibition among lexemes. His implementation of the phonological aspects of the model predicted, and experiment has verified, that at high speech rates exchanges become more common, relative to anticipations. (Interestingly, this prediction arose due to a property of his model which is absent in FIG. After outputting a phoneme Dell's model inhibits it only temporarily, so it is possible for its activation level to rebound, unlike in FIG (§6.5.1). If the same phenomenon also holds at the lexical level (and it might not (Stemberger 1985b)), then that one detail of FIG is incorrect.)

Stemberger (1985a) proposed a similar model to account for speech errors involving word choice and syntax. This model foreshadows FIG in two ways: it chooses when words appear in the same way it chooses which words appear, and it allows 'many phrase structure units' to be 'partially activated' simultaneously — that is, it has competitive parallelism. This model was apparently never implemented.

Gasser (1988) offers a more detailed spreading activation account of substitution errors.

FIG does not yet model human speech errors. It would not be hard to make it exhibit certain observed errors. However if noise were introduced randomly FIG would be more likely

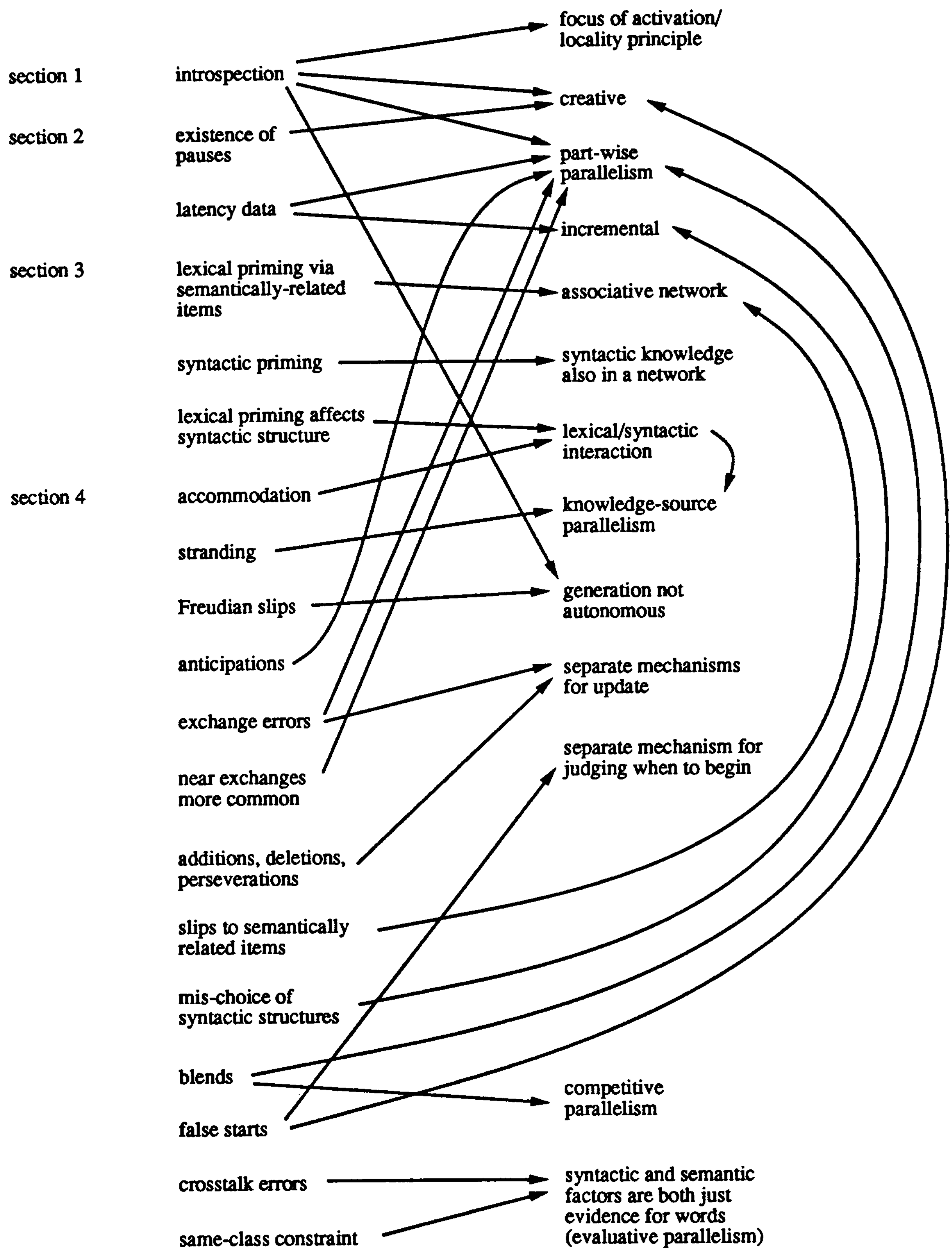


Figure 8.1: Summary of Psycholinguistic Evidence Supporting FIG

to produce strange errors than the kinds people make, since it is not yet a very robust system anyway. Nor is FIG a model of false starts, response latencies, priming effects, and so on. It would take a lot of work to make it do exactly what people do. To model timing data, in particular, would require correspondence at an incredibly accurate level, perhaps involving the modeling of individual neurons.

Thus there is a promising trend toward building spreading activation models of human language production. FIG goes further in three respects. First, it is compatible with introspection, pause data, and priming data, not just speech error data. Second, it is the only model to completely escape the legacy of structural linguistics and old-style computer models, as discussed in §2.4 and §7.1; even recent psycholinguistic models retain notions such as decision, choice, syntactic slots, and matching (Dell 1986; Stemberger 1985a; Bock 1987). Third, FIG is the only model which actually explains how generation works in the normal case — it is well-specified enough to exist as a computer program that produces non-trivial outputs.

FIG answers the question of why people are very good at producing sentences on the fly. The simple answer is lots of parallel processing at a low level. This is supported by the fact that speakers are seldom consciously aware of the many possible lexical items and syntactic forms they can choose among. Nor do they notice when goals conflict or interact. Lots of parallelism obviates the need for backtrack, scheduling decisions, structure-building, and explicit syntactic decisions; all things which have made generation hard for standard computational models.

There appear to be no data which contradict my principles of generator design (§2.3), and much to support it, as summarized in Figure 8.1. Of course, there are alternative explanations of the evidence. These have not been mentioned because my purpose here is not to prove that FIG is the best model of the human generation process, but merely to show that it is compatible with the evidence.

FIG has not yet resulted in directly testable predictions, but many of my claims are potentially falsifiable. The FIG model is incompatible with: a serial algorithm or a fixed order of decisions (vs parallel processing), explicit syntactic decisions (vs emergent word order) structure-building (vs emergent syntactic structure), binding words or concepts to syntactic slots (vs emergent organization), lexical choice by constraints (vs syntactic and semantic considerations affecting the activation of words), lexical access as a discrete event (vs the activation level of a word increasing over time), modularity (vs a unified network), copying or instantiation (vs a type-only system), and feedback-free generation (vs separate processes for update).

# Chapter 9

# A Model for Free Translation



An important goal for machine translation (MT) is to produce natural-sounding output. This requires more flexibility and 'creativity' than current technology supports. I propose a new abstract model of translation, the 'F Model', in which both expressing the meaning and producing a natural-sounding target language output are run-time goals of the system. The implementation of this model requires a richer intermediate representation, a more powerful generator, and a less powerful parser than are commonly advocated. This approach avoids some of the theoretical problems with and internal contradictions present in the interlingua model. It also accounts for several important characteristics of the process of translation as done by people.

This chapter is included to provide extra evidence for my view of the nature of the generation task. Readers who are completely convinced by the arguments of Chapter 2 and who have no interest in machine translation can profitably skip this chapter.

## 9.1 The Need for Free Translation

Assume that we are translating a certain Japanese newspaper editorial into English, and, after a sentence giving some statistics on the high cost of living in Tokyo[1], are faced with:

9.1 *keizai kikaku choo ga happyoo shita suuji ni wa aratamete odoroita.*
*economy plan agency SUBJ announce PAST [relative clause boundary] statistics CAUSE TOPIC again surprised.*

A conservative translation is:

9.2 *I was surprised anew at the statistics announced by the Economic Planning Agency.*

Freer translations are:

9.3 *The latest Economic Planning Agency report drives home just how much it costs to live in Japan.*

9.4 *These statistics from the Economic Planning Agency merely confirm the painful experience of all Japanese.*

The latter two are more natural English, and hence more readable. In each case there are several interesting deviations from the structure of the original, including:

- lack of explicit mention of surprise. This is because it is awkward in English to use a verb like *"surprise"* with a word like *"again"* or *"anew"*. Instead, the emotional impact surfaces in the words *"cost"* and *"painful"*.
- lack of an explicit word for *"anew"*. In English there are verbs which conflate the notion of 'once-again' with 'be-informed' or 'realize', namely *"drive home"* and *"confirm"* respectively.
- having *"the report"* or *"the statistics"* as subject of the main clause. This option was not available to the original author writing in Japanese. On the other hand, the choice he made, zero subject, is not available in English. Nor is it possible to maintain the same sentence structure and simply restore the elided subject, since neither *"I"*, *"we"*, nor *"we Japanese"* is altogether appropriate.
- being in present tense. Since the English sentence has the statistics as subject, it describes a continuing condition, not the report of a personal reaction, and so past tense is inappropriate.

This example illustrates two points: First, the target language constrains the translation in complex ways — the way that one part of the output is realized can depend on the way that other parts are realized. Second, the structure of the input cannot always be preserved. To the extent that the output exhibits structure (syntactic and conceptual) not present in the input, we can say that translation must 'create' new structure. The following subsections expand on these points.

[1]To be specific, the sentence is *"tookyoo o hyaku to shita bukka suijun wa beikoku no nyuuyooku ga nanajyuu ni, nishi doitsu no hamburugu ga rokujyuu hachi da to iu"*, which can be translated more or less adequately as *"On a price index with 100 being Tokyo, New York is 70 and Hamburg is 68."*

### 9.1.1 Target Language Dependencies

The realization of one part of the output can constrain the realization of other parts. Yet most approaches to MT assume that translation is essentially a compositional process. For good quality a compositional approach will not suffice: a generator expressing one concept must be sensitive to the way other choices have been made (§2.1.2).

To find a word compatible with other words may require inference to arrive at many candidate words. For example not only *"confirm"* but also *"verify"*, *"show"*, *"corroborate"*, and so on are appropriate in Example 9.1. Such a wealth of alternatives is needed to allow the generator to select from among them a word which is compatible with the other target language (TL) choices, here, for example, the use of *"statistics"* as the subject and *"painful experience"* as the object. Both the possibility of and the necessity for a proliferation of choices have been obscured by previous discussions of machine transation, perhaps because they strive to minimize the amount of computation required.

Problems of lexical gap, structural mismatch and so on have often been considered, but they are typically only studied one-by-one, never considering their interactions. For example, the problem of lexical gap is well known, but even systems which deal with this problem seem not to adequately consider the ways in which it can interact with syntax.

Recognition of the importance of 'context' in generation is a step in the right direction, but the very word 'context' presupposes that compositionality is the normal mode of processing.

The computational analog of this problem of dependencies among TL choices is that there are interactions among the the rules applied in translation (Arnold & Sadler 1990).

A partial solution to the problem of dependencies is to use larger chunks of text, parameterized texts, for example. In generation, we see this technique in the use of a phrasal lexicon, which includes not only words but also large phrases, such as *"Wall Street's securities markets"*, and *"were swept into a broad and steep decline"* (Kukich 1988), similarly for (Miller & Rennels 1988). Using large phrases allows one to produce natural text without having to deal with the low-level dependencies among syntactic and lexical choices. The weakness is that the lexicon is tuned to a specific domain — appropriateness is achieved at the expense of flexibility.

Thus there is a dichotomy: some approaches can handle general text, but only by translating compositionally, and other approaches can produce high quality output, but at the expense of flexibility. What is needed is a system that can build up an utterance out of small pieces, but which can produce smooth outputs. This requires active consideration of TL constraints and dependencies at run-time.

### 9.1.2 Creativity

As an example of the impossibility of preserving the structure of the input, consider an announcement often heard on Japanese trains:

9.5 *o wasuremono ga nai yooni go chuui kudasai.*

HONORIFIC forgotten-things SUBJ NOT-EXIST TO-THIS-END HONORIFIC attention please.

A literal translation is

9.6 *Please pay attention in such a way that there are no forgotten things.*

Again, we see that there is no straightforward way to turn this into normal English — indeed, the structure of the original gives virtually no clue as to how the information should be expressed in English. An acceptable translation might be

9.7 *Please check carefully to make sure that you have all of your personal belongings with you as you leave the train.*

although the null string would be the most pragmatically appropriate English translation in most contexts.

The need to sacrifice the structure of the original can also be argued by simple comparison of the source language (SL) lexicon with the target language (TL) lexicon. For one thing, there can be SL words for which the TL has no equivalent (that is, the TL may have a 'lexical gap' with respect to the SL (Nitta 1986)). This can be equally true of open-class words, such as *"privacy"* and *"giri"*, which has to do with a feeling of obligation or duty, and closed-class words, such as *"worth"*, *"worthwhile"*, and *"shimau"*, which has to do with completed, irreversible, regrettable events or actions. There, are similarly 'syntactic gaps' (Tsutsumi 1990). To translate SL texts which include these words or structures requires some kind of restructuring or paraphrase.

In sum, to produce a more readable sentence it is sometimes necessary to sacrifice the structure of the original, and therefore to create new structure. These are things which human translators and post-editors do freely, but existing machine translation systems are very poor at.

This need for 'creativity' has often been noted in the literature on human translation (Kelley 1979). It has unfortunately been ignored in the MT literature. There are several reasons for this. One is the often heard but never substantiated claim that creativity is necessary only when translating literature, never for technical prose. Another is that readers do not need natural translations. It is true that in many applications poor quality is acceptable because readers are able (and willing) to use unnatural output. But this is not always the case, which is why post-editing of MT output is often necessary.

There are also methodological reasons why the need for creativity has not been addressed by MT. One reason is that the problems of parsing and inference themselves present such a challenge that no one has looked beyond them. Yet even a complete solution to those issues will not suffice for the construction of good machine translation systems — the need for creativity to produce natural utterances will remain. Another reason is that practical people do not like to talk about ideas as airy as creativity.

### 9.1.3 Strategies for MT Research

The prevailing methodology in MT is what one might call the 'cumulative' approach: starting from some implemented model, such as word-for-word translation, one typically asks 'what's obviously wrong?' and 'how can we quickly fix it?' Similarly, theoreticians can be heard to say 'my model handles the simple cases, which is the obvious place to start. Harder cases obviously require extra mechanisms, and I can add those as needed'. Gradual development is questionable as a long-term strategy for software development and even more so as a research methodology. The danger is that adding ever more mechanisms may make the system beyond the limits of maintainability or the theory grow beyond the size of comprehensibility.

My strategy is to *first* figure out what the ideal is, and then figure out how to approximate it cost-effectively. I am also aiming for a general model able to clearly explain how to translate not only simple sentences like *"John kissed Mary"* but all inputs. In order to arrive at a general model, it seems necessary to start with complex cases, which was why this chapter opened with a non-trivial example.

## 9.2 Present Technologies for Machine Translation

This section discusses the basic technologies used for machine translation. More philosophical arguments about various approaches to MT are discussed in Section 9.5. The overall conclusion is that existing technologies do not exhibit creativity, nor are they directed towards the production of natural outputs. Of course it is impossible to prove that any given approach is inadequate, since for any difficult input proposed a system can be extended to handle that particular input.

### 9.2.1 Structure-Preserving Translation

Linguistic, philosophical, and semiotic theories of translation have typically focused on the question 'What is it that makes X a translation of Y?' The answer is typically in terms of qualities which are shared by X and Y. From this it is a small step to the view that a translation *process* must preserve those qualities.

This idea gives rise to the implementation technique of using 'source sentences as moulds of target sentences' (Tsujii 1986), or in other words doing 'structure-preserving translation as first choice' (Somers *et al.* 1988). As far as possible the structure of the input is preserved; adjustments are made only as needed to ensure that the result is a grammatical sentence of the TL. Structure-preserving translation usually proceeds step-by-step, applying various operations until the input is converted into the output. Examples of such operations include lexical substitution, reordering constituents, adding function words, re-organizing information to suit the case frame of the TL verb, and so on. Even if the system uses non-destructive operations, such as unification, the result is the same. (At this point I am ignoring the fact that these steps are usually clustered into stages called 'parsing', 'transfer', and 'generation'.)

The transfer and the interlingua approaches are both based on this idea of preserving structure. They differ in emphasizing different aspects. The transfer approach focuses on the differences between the input and the output. Accordingly, the heart of a transfer system is a 'transfer component', responsible for making the necessary adjustments to the structure. The interlingua approach, on the other hand, focuses on what the input and output have in common; the heart of an interlingua system is the language (the interlingua) used for representing the invariant information.

While there are other differences between the interlingua and transfer modes, as discussed in Section 9.5, they basically rely on the same technology when implemented. For example, although transfer systems usually operate on 'syntactic structures' while interlingua systems operate on 'meaning', their representations are not fundamentally different when implemented (Tsujii 1986; Nirenburg 1990). Indeed, representational tools (tree structures, network structures, case frames, features, primitives, and so on) are borrowed freely back and forth. Nor are the two approaches distinguished by the amount of knowledge applied to the parsing and generation processes, nor by the amount of information made explicit in the intermediate representations — these issues are now generally acknowledged to be independent of the choice between transfer and interlingua.

What exactly is it that is invariant between the input and the output? This is the key question for structure-preserving approaches. Some systems preserve constituency structure, others dependency structure, others deep case structure. The goal, especially among advocates of interlinguas, is to use a representation formalism which captures the 'content' of the input; irrelevant aspects of 'form' can then be ignored.

Yet there is no simple dichotomy between structural properties which translation must preserve and those which it need not. Even surface structure should be preserved if possible, since the order of presentation may, among other things, convey the distinction between given and new information. Yet this goal will often conflict with the goal of preserving dependency structure or of preserving case structure, those goals can also conflict with each other, and so on. In general, there are many properties which a translation should preserve if possible; trade-offs among these various goals are an unavoidable part of the translation task.

There is also no simple dichotomy between information which must be explicitly included in the output and that which need not be. This is clear for Example 9.1, where, although the input and the free translations mean roughly the same things to a human reader, the words in the input and the output do not refer to the same concepts. In general the language at hand affects what is said explicitly. That is, there is no rigid line between what is said because the speaker wants to and what is said because of the language at hand. For example, it is easy and almost unavoidable to specify the sex of a person in English, since that information piggy-backs on pronouns. In contrast, that information will not be included in Japanese unless it is important, since it requires an explicit modifier. Given that the information present in the *input* may not directly reflect the speaker's intention, the information present in the output certainly need not copy the content of the input.

Thus, there is no simple answer to the question 'what is preserved in translation?' nor to the philosopher's question 'what do a sentence and its translation have in common?' (short of answers like 'speaker intentions', which dodge the computational question). Fortunately these questions do not need to be answered in order to understand the process of translation or to build a good MT system.

Neither the transfer nor the interlingua model has anything to say about naturalness. This is sometimes reflected in the output quality of such systems; the output is often 'faithful' to the original text to the point of poor readability or incomprehensibility. There is no attempt to make new structure; the only restructurings done are for the sake of the goal of grammaticality, an easier goal than naturalness. In a word, structure-preserving translation systems are not 'reader-friendly'.

### 9.2.2 Translation Using Parameterized Texts

A different answer to 'what makes X a translation of Y' is that X and Y have 'the same effect on the reader'. This theory of translation gives a criterion but does not specify the process — that is, it does not give any guidance as to how to produce a translation.

Yet there is a class of utterances for which this idea can be straightforwardly implemented, namely for those utterances whose only content is their pragmatic force, as is often the case in situated dialog. Consider the indirect request *"have you seen Dan today?"*. For many purposes the only information needed to translate this is the motivation behind the utterance, namely an instance of the goal *'wanting-to-know-where-someone-is'* where the someone in question is *'Dan'*.

Translation of such sentences can be handled with a simple two-part technique. First, the system recognizes the speaker's goal, perhaps instantiating variables, such as the name of the person, and extracting parameters, such as the level of politeness. Second, it looks up the TL text for the goal, filling in the blanks according to the variables and choosing among alternatives according to the parameters. I will refer to this approach as 'translation using parameterized texts'.

Parenthetically, goal-oriented TL utterances can equally well be produced by letting a monolingual SL user directly specify his goal and the relevant variables and parameters, perhaps by using a menu (Somers *et al.* 1990).

Translation using parameterized text has also been used for translating information which fits into known structures. Newspaper stories of hostage takings or traffic accidents, for example, fall into this category. Variables include things like the name of the hostage, the ransom amount, and so on (Carbonell *et al.* 1981; Ishizaki 1988). The system need only recognize the text type, determine the values for the variables, look up the hostage-taking-newspaper-story template for the TL, and flesh it out. The parameterized text may be somewhat more flexible than a simple template — there may be branch points and optional material.

Systems using parameterized texts can produce outputs which are natural and which incorporate structure that was not present in the original. Yet this naturalness is largely due to specific, template-like generation knowledge. This technique cannot handle general texts — that is, it fails to handle the 'productivity of language'.

Parenthetically, the interlingua approach is advocated as giving both the broad coverage of a productive representation system and the high quality of parameterized texts (Carbonell *et al.* 1981). Yet translation using compositional representations, such as Conceptual Dependency, is generally structure-preserving, and translation using scripts relies of course on parameterized texts. While the two approaches can be used in tandem, each applied to different parts of a sentence (Iida 1990), they are radically different technologies, and it is not clear how to wed the two to get the advantages of both.

### 9.2.3 Translation by Analogy

The basic idea behind this approach is that a translation should be done 'similarly' to the way adequate human translations have been done in the past. This approach cleverly sidesteps questions about the nature and purpose of translation.

Translation by analogy does without knowledge of the SL and TL other than that which is implicit in the corpus of sample translations. Thus at run-time the system must, in effect, both uncover the facts of the languages and apply them to the given input. This can give good performance even when no one yet knows the facts, which is the case, for example, for polysemous words such as common verbs and prepositions (Sumita *et al.* 1990). There are, however, two problems with translation by analogy: it is not clear that it is better and it is not clear that it is possible.

Translation by analogy in effect both extracts the facts about language and applies them to an input, all at run-time. It would seem the better strategy to determine the facts of language off-line, to avoid the great deal of storage and computation at run-time required for translation by analogy. In fact pre-computation of analogy has been proposed (Nagao 1984). This position appears equivalent to use of a standard translation system whose knowledge has been obtained by machine learning, rather than from linguists and programmers. Apart from the intrinsic difficulty of language learning, this approach has the further problem that learning from translation pairs requires the machine to learn two languages together instead of one at a time, which does not seem to be an advantage.

The first step of translation by analogy is finding similar cases in memory. This requires a good metric to measure the similarity of the input to the stored cases. The problem is that similarity depends on context. For example "*takushii de no hanashi*" maps to "*conversations in the taxi*", unlike "*basu de no ...*" and "*densha de no ...*", which map to "*... on the bus*" and "*... on the train*", respectively. To handle this case "*bus*" must be rated more similar to "*train*" than to "*taxi*". Now consider the problem of translating "*basu no untenshuu*", where the system knows that "*takushii no untenshuu*" maps to "*driver of the taxi*" and "*densha no untenshuu*" maps to *"motorman of the train"*, as shown in the right side of Figure 9.1. If the same metric of similarity as before is used in this case, the system is likely to choose, incorrectly, *"the motorman of the bus"*. To get the metrics of similarity right will require making them sensitive to the context of use in complex ways; this seems as hard as solving the general inference problem in AI.

| | | |
|---|---|---|
| conversations in a taxi | taxi | driver |
| conversations on a bus | bus | ? |
| conversations on a train | train | motorman |

Figure 9.1: A Misleading Analogy

Translation by analogy has been treated as a compositional process. That is, it operates by translating word for word or phrase-by-phrase, combining 'fragments' by concatenating them (Sato & Nagao 1990). But of course it is not generally true that words from the SL map one-to-one to words of the TL. Translation by analogy does not yet allow the choice of the TL word to be constrained by, or to impose new constraints on, the translation of the rest of the information; much less does it deal with problems of lexical gap, or anything to do with syntax. In short, this approach relies on the adequacy of structure-preserving translation as the underlying model.

## 9.3 Translation as a Creative Process

Thus we see that existing approaches are not up to the task of producing natural translations. This section accordingly presents a new abstract model for translation. Computational implications of this model are discussed in the following section.

### 9.3.1 The F Model

To restate concisely the claims of Section 1, *translation has two goals*, namely:
**Goal A:** *Express the input.*
**Goal B:** *Produce a natural utterance.*

To explain the relation between these goals I propose the following
**Interaction Rule:** *Both A and B are active goals at run-time.*

These three statements constitute a new[2] theory of translation, the 'F Model'. ('F' stands for 'free' and 'flexible'.)

'Active at run-time' means that a translation system must simultaneously strive to satisfy both goals. The two goals are not independent, and it is often necessary to make trade-offs between them. This can be done most easily if they are active together.

### 9.3.2 Other Models

It is clear that the result of a translation must meet these goals, but no one has previously proposed that both be active at run-time. This section discusses how other models of translation incorporate the two goals.

In structure-preserving translation goal A is handled implicitly by preserving the structure of the input. That is, the structure is not changed except to the extent that some constraint of

[2]Actually, the F Model was first presented in (Ward 1989).

the TL requires it. Goal B is usually the focus of processing; that is, the various algorithms of the system work towards this goal (although in the weaker form of producing a grammatical, not necessarily natural, utterance).

In translation using parameterized texts goal A is the responsibility of the lookup function: it retrieves a text whose function is as close as possible to that of the input. Goal B is achieved for free, since the TL texts are essentially written in advance.

In translation by analogy goals A and B are not explicitly dealt with at all — both goals are implicit in the way the analogies are worked out.

In general, the goal of fidelity to the input has been stressed more than the goal of naturalness in MT. It is of course important to be faithful to the source text, but this is not the only goal in translation. If it were, the original text would suffice, but in fact MT output must be readable in order to be useful.

### 9.3.3 Scope of the F Model

The F Model characterizes the task of translation — it does not directly specify how to translate. In other words, the F Model is more of a framework than an architecture (although it does have architectural implications, as discussed in the next section).

The F Model does not define criteria for success in goal A and goal B. To explain when one utterance is a faithful re-expression of another requires further research in inference and human language understanding. To explain when an utterance is natural requires further research on language. Even after the research is done, there will remain the task of gathering and encoding vast amounts of world knowledge and language knowledge. These research tasks are of course important not only for the sake of machine translation.

The F Model is not a normative theory of translation. It does not explain why one translation is better than another. In particular, it does not tell for a given situation which goal is more important — that is, it does not prescribe the degree to which a translation should be free or literal. The weightings given to goal A and goal B must depend on the exact context and purpose of the translation. When implementing the F Model an important task is developing a metric for weighing the importance of preserving some nuance of SL meaning against the importance of respecting some TL preference.

The F Model accounts for some of the phenomena of human translation, as discussed in Section 9.6.

## 9.4 Design Implications

This section discusses some computational implications of the F Model. I have nothing new to say about techniques for analysis, so the next two subsections discuss only the intermediate representation and the generator.

### 9.4.1 Implications for Generation

The Interaction Rule calls for the two goals to be simultaneously active. This must obviously happen during generation, since this is where knowledge about how to build a natural TL utterance is available. Thus, a key characteristic of the F Model is that it requires an unusually powerful generator. This subsection describes briefly what this entails, recapitulating briefly some arguments of Chapters 2 and 3 with special reference to the needs of MT.

First consider the 'simple' problem of how a generator could come up with the word *"confirm"* in Example 9.4, reproduced here.

9.8 *These statistics from the Economic Planning Agency merely confirm the painful experience of all Japanese.*

Note that *"confirm"* corresponds to no single word or concept of the input, or of any plausible intermediate representation. For the generator to discover that this word is appropriate requires inference. In general a generator must use world knowledge in order to access words not directly related to the input, as discussed in §3.4. A generator with this ability does not need direct links between SL and TL words. That is, so long as the meanings of SL and TL words and world knowledge are adequately represented, there is no need for a bilingual dictionary.

Such inference can lead to many candidate words, for example not just *"confirm"* but also *"verify"*, *"show"*, *"corroborate"*, and so on. The generator must not prune the set of candidate words until all information about TL constraints (including those imposed by other words) is available. There have been other proposals for inference for the sake of flexible word choice, but these have not advocated parallel consideration of many candidate words. Also, inference is usually treated as a backup mechanism, used only if the first choice of a concept or word turns out to be unworkable due to syntactic constraints or due to a lexical gap in the TL at hand (Tomabechi 1987).

Parenthetically, the idea of inference in generation suggests a simple benchmark for MT systems. A MT system should be able to fully exploit the resources of the TL. The test is: given a TL word lacking a SL equivalent, is it possible for the MT system to produce that TL word? For example, is it possible to give a Japanese-to-English system an input that causes it to use the word *"worth"*? If not, then it is obviously handicapped when producing natural English sentences. If so then it is clearly not just performing a one-to-one mapping, but rather is doing a kind of 'opportunistic' word choice.

Previous translation systems are more or less compositional — they translate an input part-for-part, considering 'context' information only to some degree. Yet no matter how appropriate a TL word is in isolation it is worthless unless it can be part of a coherent sentence. The F Model is not compositional at all — all the information from the input must always be available to affect every choice the generator makes. (Of course, any implementation of the F Model will have a limit on the amount of information which it can consider at one time.)

If a generator pays full attention to dependencies among TL choices and can use inference to arrive at many candidate words, it will not be enslaved to the structure of the input. The syntactic and conceptual structure of the output will 'emerge' as a side effect of choosing words. That is, the 'creation of new structure' can happen as a side effect of choosing appropriate and mutually compatible words, and so there is no need for explicit concept choice and structure building.

### 9.4.2 Other Implications

To meet goal A the generator needs information from the input; to extract this information is the task of a parser. This requires inference to make explicit information which was implicit in the input. Analysis must augment the input with additional information; it need not restructure or transform the input or build new structure, and in fact it should not, as will be discussed in §9.5.2. That is, the output of the parser is still in the conceptual system of the SL; the task of coming up with a structuring appropriate to the TL lies in the province of the generator.

How much information should the parser extract? Figure 9.2 shows informally some of the information that is necessary to generate a good translation of Example 9.1. Figure 2.1 is also an example of this; it is intended to show some of the information involved in *"minakami kara ookina momo ga donburiko donburako to nagarete kimashita"*.

audience: educated, American
genre: newspaper editorial
style: faintly sarcastic, slightly playful
author's viewpoint: Japanese
author's goal: convince readers that government policies should be changed
overall point of article: regulations in Japan are pernicious
primary evidence: regulations cause high prices
previous sentence: *"Taking Tokyo's cost of living as 100, New York's is 72, and Hamburg's is 68."*
source of statistics: report
source of report: Japanese Economic Planning Agency
content of statistics: comparison of price indices in different countries, much more
period covered by statistics: recent past
release date of report: yesterday
source of author's knowledge of statistics: reading report
newness of information to the author: brand new
reliability of Economic Planning Agency statistics: good
typical way government agencies 'mtrans' statistics: report or press conference
significance of statistics: proves japanese-cost-of-living-is-excessive
duration of japanese-high-cost-of-living: many years
author's previous view of japanese-cost-of-living-is-excessive: annoying
general cause of author's annoyance: personal experience, political beliefs
effect of statistics on author's belief: provide evidence for, strengthen, justify
magnitude of relative expensiveness of Japan: large
author's opinion of japanese-cost-of-living-is-excessive: unreasonable
author's opinion of cause of japanese-cost-of-living-is-excessive: government policies
original order of presentation: J.E.P.A, announce, statistics, cause, again, surprise

Figure 9.2: Representation of Some of the Information Needed to Translate Example 9.1

Fixing requirements for the intermediate representation, or fixing a representation language, can determine the tasks of parser and generator. It is not clear how best to do this. On the one hand the parser could do very little. For example, some of the information in Figure 9.2 could quite reasonably have been left for the generator to infer as needed. On the other hand a much more complete intermediate representation would also be possible. But no matter how much information is made explicit in the intermediate representation it must be 'embedded' in a representation of world knowledge so that the generator has access to definitional and other information.

### 9.4.3 An Implementation

An implementation of the F Model exists. FIG is of course its generator — indeed, FIG was developed in tandem with the F Model. (Of course, the F Model need not be implemented with spreading activation.) I have also built a small stack-based Japanese parser. Its responsibility is to activate nodes which represent words, structures, and concepts present in the input.

Relating FIG to the needs of the F Model, goal A is met by activation flow during generation from SL concepts via TL concepts to TL words; Goal B is met by the way the structure of the network affects the words output. Activation provides the metric for weighing possibly contradictory goals — words with the greatest total amount of activation (from all sources) get selected. Thus there is no need for explicit reasoning about trade-offs between goals.

Translations done by the system include:

*"mukashi mukashi aru tokoro ni ojiisan to obaasan ga sunde imashita"*
long-ago long-ago certain place LOC old-man and old-woman SUBJ live PAST-PROG
once upon a time there lived an old man and an old woman

*"aru hi ojiisan wa yama e shibakari ni ikimashita"*
certain day old-man TOPIC hill DEST gather-wood PURP go-PAST
one day the old man went into the hills to gather wood

*"obaasan e momo ga nagarete kimashita"*
peach SUBJ old-woman DIR float come-PAST
a peach floated to an old woman

*"ojiisan wa meeri ni momo o ageta"*
old-man TOPIC peach PATIENT Mary RECIP give-PAST
the old man gave a peach to Mary

*"meeri o koroshita"*
Mary PATIENT kill-PAST
Mary was killed

*"jon ga momo o fooku de tabeta"*
John SUBJ peach PATIENT fork INSTRUMENT eat-PAST
John ate a peach with a fork

*"jon wa yama e kawa kara no momo o tabe ni itta"*
John TOPIC hill DEST stream SOURCE GEN peach PATIENT eat PURP go-PAST
John went to a mountain to eat a peach from a stream

## 9.5 Philosophical and Software Engineering Issues

The F Model primarily addresses the most basic issue in MT, namely how to produce natural outputs. This section discusses some secondary considerations in the theory and practice of translation. Most these issues have been raised in the arguments over the relative merits of the interlingua and transfer approaches (Carbonell *et al.* 1981; Nagao 1987; Nagao *et al.* 1986; Slocum 1986; Nirenburg 1990).

### 9.5.1 A Universal, General-Purpose Interlingua

There are two motivations behind the idea of an interlingua. One is that, since thought is language-free, MT should proceed via language-free representations. The second is that, since any specific natural language has idiosyncrasies and flaws, a good way to translate is via sentences of an artificial ideal language. I discuss each motivation in turn.

The belief that thought and meaning are universal and language-independent is very common, yet far from self-evident, as the Whorfians have pointed out. The problem for translation is that the meaning of the input and that of the output are necessarily situated in 'different conceptual systems', and are hence inherently not equal. In particular, the differences between languages do not simply disappear as a result of 'deeper understanding'. The F Model explains how translation is possible anyway: the generation phase has the flexibility to build up utterances exhibiting new conceptual structures.

Some claim that the same interlingua can be used for translating between any language pair. If this commitment is taken seriously, it requires the parser to do more work than necessary, for example, extracting information about gender when translating from Japanese to Chinese.

The belief in universal representations is also hard to sustain in practice. Experience building MT systems has shown that meaning structures underlying a sentence in one language can differ significantly from the structures underlying any reasonable translation. This problem has been recognized as the need to 'paraphrase' the meaning representation before generation begins (Carbonell *et al.* 1981), and concretely addressed with 'conceptual transfer' (Uchida 1988). Unfortunately this technique addresses the symptom, not the root cause — like syntactic transfer, it still basically relies on the structure of the original, creating no new structure.

At this point someone might still object that 'a translation must *mean* the same thing as the original, otherwise it's not a translation', Yet this statement cannot be true unless it is intended as the definition of either 'meaning' or 'translation'. I believe it is better to reason from observations than from definitions. When I talk about translation I mean nothing other than normal translation, as exemplified in the relation between sentences (1) and (3).

The other motivation underlying the notion of interlingua is that any given natural language is flawed, for example in allowing lexical and structural ambiguity, elision of information, and redundancy, and so it is better to work with an artificial language having none of these problems. The artificial language can have a different vocabulary and a different syntax, perhaps one based on graphs rather than string or trees. This view of interlingua surfaces in the slogan 'a representation formalism which does not represent the syntax of any language'.

This idea is problematic on two counts. First, it is not clear that escaping from the syntactic structure of the original is a worthwhile goal in itself. That is, it is only plausible as part of the process of producing a TL utterance, and should therefore be done in the generation phase, not during parsing. Second, representations in such formalisms are generally, like a natural language, concise, containing roughly as many symbols as the input has words. Yet, as Tsujii points out, any concise interlingua will be insufficient, because a great deal of syntactic, semantic, and discourse information must be available for generation to make good decisions (Tsujii 1986).

The contradiction at the heart of the notion of interlingua is that its charm comes from being an approximation to the language of thought, but its performance comes from being a tidied-up natural language. But the language-of-thought idea is fundamentally incompatible with the ideal-natural-language idea: to the extent that a representation is universal, complete, and fully disambiguated, it is not likely to be concise.

The F Model does not require its intermediate representation to be a 'universal meaning representation', to be 'fully disambiguated', or to have any other nice theoretical properties. Nor does it assume that the intermediate representation is compact, or even that a symbolic representation is necessary at all, thereby leaving an opening for a PDP connectionist approach.

### 9.5.2 General-Purpose Parsing and Generation

Ideally a parser should not depend on the TL and a generator should not depend on the SL. That is, an parser for language X should produce an output usable for generation into language Y or Z, and conversely a generator for language Z should be work regardless of whether its input comes from an parser for language W or language X. What then of the differences among languages? Advocates of interlinguas often state that the differences between languages 'go away' if 'complete understanding' is done. That is, a deeper parse results in a structure closer to the TL, which obviates the need for a transfer component.

Thus there is no place in the interlingua model for adjustments of structure in order to achieve a natural sentence of the target language. Therefore, the only way to get natural TL sentences is to have a little 'slack' in either the parsing or the generation process. Specifically, interlingua-inspired systems seem to parse into impoverished representations, discarding information in the process. This perhaps explains the unfortunate tendency for interlingua-based systems to be rather cavalier about the preservation of meaning. Distinctions made in the source language get blurred together unnecessarily. For example, the distinctions between *"my head hurts"*, *"I have a headache"*, *"there is a pain in my head"*, *"my head aches"*, and *"I have a pain in the head"* are lost when all are translated as *"atama ni itami ga arimasu"* (Tomita & Carbonell 1987). Perhaps such distinctions are considered irrelevant to 'meaning', but a patient consulting a doctor would probably wish them to be respected. Rather than having some unadvertised slack in the parser, it would seem better to cope with mismatches among languages in a more principled manner; for example the F Model calls for changes in meaning only as 'required' by the TL.

MT systems which do without a transfer component tend to have the parser do things to make the generation task easier. (This is possible since most interlingua-based systems are built with one language pair in mind.) This is often justified with the slogan 'full disambiguation', but this can be misleading — it does not justify having the parser go so far as to choose TL equivalents for words which are merely vague in the SL, despite (Wilks 1972). While more recent systems do not require quite so much of the parser, parsing is still commonly considered to be the big problem for MT (Nirenburg 1990).

More generally, the parser and the generator should ideally be usable not only for MT, but also for other natural language processing tasks. A MT parser which outputs something 'closer' in structure and conceptual vocabulary to the TL is clearly not general-purpose. As Tsujii points out, " 'to extract more kinds of information explicitly during the analysis' does not ... necessarily mean 'to express such kinds of knowledge in a language-independent framework' "(Tsujii 1986), let alone one closer to the TL. For example, the parser need not replace the parse tree with a representation in another language, such as an interlingua. In general, understanding depends on the task for which understanding is being done, just as there is no meaning of 'disambiguate' that does not refer to the TL which is to be finally produced or to the larger task of the system.

The transfer approach allows general-purpose parsers and generators, but at the price of requiring a transfer module which relies on contrastive knowledge, that is, knowledge of correspondences between the grammars and lexicons of the SL and TL languages. One problem with contrastive knowledge is that it is essentially unmotivated — it is of no use for anything but the task of translation. Another problem is the practical one that new contrastive information must be built up from scratch for each new language pair (this is the old argument that 'in order to translate among $n$ languages transfer requires $n^2 - n$ modules but interlingua requires only $2n$ modules').

The F Model does without contrastive knowledge thanks to the use of world knowledge. Words are translated on the basis of what they mean. TL equivalents are found at run-time, so there is no need for direct bilingual lexical correspondences. While the F Model does not require contrastive knowledge it can use it if available — it simply eases the task of the generator to have some initial candidate words to work with. Since the generator is required to work with a rich input anyway, it can also exploit any further scrap of information. This is equivalent to the ability to perform 'multi-level transfer' (Tsujii 1986).

The F Model explains translation among several languages in a novel way. The parsing process does not depend on the TL; its output is entirely 'within the conceptual system' of the SL. Thus the starting points for generation will be different, depending on the SL. This is possible since the generation process does not rely directly on the structure of the input. The same world knowledge and TL knowledge is used regardless of the starting point.

Extending a system based on the F Model to handle a new language requires addition of knowledge about that language and the concepts specific to it. No matter how many SLs there are, the TL knowledge and process is the same, and similarly for many TLs.

## 9.6 What People Do

Analogies to human translation or human cognition are often used in arguments about machine translation. Unfortunately there are as yet no convincing theories of human translation and interpretation, and interesting experiments are also few (Gerver 1976), so these arguments are based largely on introspection. Such analogies are nonetheless useful. This section discusses some old arguments and points out some interesting but rather neglected facts.

As part of an argument for an interlingual approach, Carbonell discusses the kind of 'translation' done informally by bilinguals who are *not* professional translators (Carbonell *et al.* 1981). This is compatible with the use of formalisms like Conceptual Dependency, which was specifically designed to represent the information which people retain after forgetting the exact words they read. But it is not what translators do — they frequently refer back to the words of the original to make sure that they do not lose any information present in the way the input is worded.

On the other hand, one of the arguments for treating MT as simply a matter of syntactic transformations and lexical substitutions is that people can translate without understanding the subject matter. While true, this is also not the ideal translation situation.

Another non-typical style of translation is 'translation as decoding' (Weaver 1955). This may be useful for describing the behavior of translators whose SL knowledge is weak, and who have to internally 'translate' into a different conceptual system to understand the input at all. But in general it seems that one can understand utterances within their own frame of reference. This is reflected by the common dictum of foreign language teachers: 'learn to think in the language you are learning' or whatever. Also, a translator is typically not troubled when initially reading an input involving some idiosyncratic SL concept — difficulty only arises when he tries to express it in the TL.

Rather than basing analysis on special cases, I believe that one should consider how good translators work when producing good translations, and the F Model is designed with this in mind.

An argument favored by advocates of interlingual approaches is that translators attain a 'language-free' state of understanding between parsing and generation. This is true trivially in that the 'thought' in the mind of the translator presumably changes from a representation of a SL text to that of a TL one, and any convenient point along this process can be labeled an interlingual representation. But it is questionable whether a brain state can be represented symbolically at all, and it is certainly not representable in a concise manner. The important point is that the intermediate 'representation' can be very rich in information and that it is necessarily embedded in world knowledge.

Recall that the two abilities required by the F Model are to:

- understand the input
- generate into the TL, respecting goals A (fidelity) and B (naturalness)

This is probably enough to account for the process of interpretation, and it is a good first step towards explaining translation, since interpretation is the underlying skill, in the same way that speaking is a more basic skill than writing (to gloss over other differences in the two tasks (Andre 1985)). Since translators have more time than interpreters and can also use paper and pencil as a memory aid, they can use other techniques. The following list of activities and their role in the translation process are based on casual observation, speculation, and introspection (see also (Gerloff 1988)). It leaves out non-run-time tasks such as deciding the purposes and parameters of the translation. The list is in no particular order.

- substitute TL words for SL words.
- alter a TL sentence to make it grammatical or more natural, that is, paraphrase the output without reference to the input. This skill is necessary for tidying up after doing word-for-word substitution. This skill and the previous one enable people without good knowledge of the SL and TL to translate 'mechanically', using a dictionary and great patience.
- brainstorm many candidate TL words. This is often used to cope with lexical gap or when free translation is necessary for some other reason. Normally candidate words are automatically accessed as part of the generation process, but probably can be done more effectively if done consciously.
- piece together candidate words to form a grammatical utterance of the TL. This is necessary to cull out and organize the candidates raised by brainstorming.
- evaluate how faithfully the output corresponds to the input. This is useful for choosing among alternative possible translations. This task is in service of goal A; conscious effort may complement the unconscious effort towards goal A in first-pass translation.
- judge the naturalness and readability of a TL sentence; this also is useful for choosing among alternative possible translations. Normally, generation processes automatically produce natural output, but when translating there can be interference from the SL. Thus this task involves conscious attention to goal B.
- debug a specific problem, that is, for some case of unnaturalness or infidelity, uncover the root cause of the difficulty, in terms of SL or TL peculiarities, and consciously decide whether it is unavoidable or how to fix it.
- re-parse the SL text in order to find some specific piece of information, for example, when the TL requires some information to be made explicit or when some TL syntactic choice hinges on a nuance of meaning. Clearly it can be more efficient to do certain kinds of inference only if necessary for generation, rather than automatically while parsing.

- use reference material. If a translator realizes that his knowledge of the SL or TL is deficient, he may consult monolingual or bilingual dictionaries, grammars, and translations of similar texts.

Some of these techniques can probably be useful for machine translation. Yet building models of them is only of academic interest until the basic mechanisms, those of doing the basic parse and of coming up with the candidate words, structures, and sentences of the TL, have been implemented. The F Model addresses this basic process.

There are some facts about human translation which seem to suggest that the F Model is on the right track.

Translation is intrinsically hard. The F Model explains this as due to the need to attempt to simultaneously attain two goals. The other models treat translation as a far more algorithmic process — slow perhaps, due to the amount of information involved, but ultimately straightforward.

When translators write about why their work is difficult, they often mention cases where goals A and B are in conflict and they must make a compromise (Steiner 1975). This is in accord with the emphasis the F Model places on trade-offs between the two goals.

Finally, there are several observations in accord with the emphasis the F Model places on generation: People translating take much more time to produce the output than they do to understand the input (as can be seen quickly from informal studies of translators at work). Gerloff, measuring the various activities involved in generation has a similar result: statistically about half of the activities performed involve solution generation, as opposed to the eighth devoted to understanding (most of the rest involve evaluation) (Gerloff 1988). The most essential requisite for would-be translators is the ability to express themselves well in the TL (Seleskovitch 1968).

## 9.7 Prospects

As MT output becomes more widely used it will have to be acceptable to readers who are less willing or able to put in the extra effort needed to understand unnatural text. 'Reader-friendly' output will become essential, and the need for a model which focuses on producing natural output, such as the F Model, will become clearer.

The F Model is an idealized model, not an immediately practical one. Indeed, building a completely adequate MT system is as hard as passing the Turing Test, and is not likely to be achieved in our lifetimes. But the F Model may be useful even today to the extent that it points out new directions for research. Whereas the commonly advocated goals for MT research are 'more knowledge' and 'better parsers', the F Model makes it clear that generators also must be greatly improved. The F Model also suggests possible modifications for existing MT systems, such as using richer intermediate representations, allowing inference while generating, and considering many candidates for TL words and structures. Some of these may actually become cost-effective in the near future.

# Chapter 10

# In Conclusion



## 10.1 How FIG Measures Up

This section discusses the extent to which FIG-as-implemented addresses the issues in generation discussed in §1.1 and §2.1.

A generator must be able to choose words with subtle meanings, to choose among a proliferation of language knowledge, and to handle rich, complex inputs (§1.1.1, §2.1.1). In FIG all this is easy, thanks to the use of activation in an associative network. A word can have arbitrarily many neighbors, and any number of input nodes can affect the choice of a word. However, I have not yet tested FIG on particularly rich inputs; indeed, the inputs are not much more than deep case representations of Japanese sentences. This is partly due in part to the lack of a powerful parser/analyzer. Also, because of the problems discussed in §6.5.1. it is not quite obvious that FIG actually could cope with such inputs

A generator is faced with a great number of interdependent lexical, syntactic, and conceptual options; to produce a good utterance it must arrive at a consistent set of choices (§1.1.1, §2.1.2). FIG does this by means of spreading activation, which provides explicit parallel consideration of all possible options and parallel computation of their interdependencies. Nodes for compatible choices of all sorts are linked and therefore mutually reinforcing. The network is thus designed to settle into a state which represents a compatible set of syntactic structures and lexical choices. One can think of this as a process of simultaneous constraint satisfaction. This has been demonstrated for collocations and other simple examples but not for anything very impressive; this is because FIG does not yet have a great variety of knowledge about language and has few options to choose among.

A generator should be robust and flexible (Chapter 9). FIG meets this need thanks to the spreading of activation via world knowledge and because it does not rely on the structure of its inputs. Some illustrations of robustness with respect to trivial variations in the organization of the input were given in §6.2. However FIG does not yet have enough knowledge to make it really flexible or robust. Indeed, it often fails when tested on a novel input that should be within its coverage. Sometimes this is simply a problem of inappropriate weights, which is easy to fix, but often the difficulty relates to one of the problems discussed in §6.5 and is not easy to solve within the current implementation.

A generator should be scalable (§1.1.2). One issue is whether the system would still work with several orders of magnitude more knowledge. FIG can, since the computation is parallel at the level of nodes. Another issue is how easy it is to add new knowledge to the system. FIG is weak on this count (§6.5.3); some kind of learning algorithm is required.

A generator which is a cognitive model should be incremental and should be in accordance with experimental evidence. FIG is both (Chapter 8).

To sum up, FIG is currently limited by lack of enough knowledge, or an algorithm for obtaining it (§6.5.3); problems of representation, particularly with the use of a small set of relations (§6.2); and problems of implementing the desired properties, such as the locality principle, with spreading activation (§6.5), Although I do not have solutions in hand there is no reason to think that these problems are insolvable.

An additional but weak motivation for building FIG was to create a useful generator with broad coverage. Instead I got sidetracked into exploring some difficult issues in generation. Thus the performance of FIG is not that impressive, compared with any symbolic generator. However it is still an enormous advance over every previous generator which is either seriously incremental or seriously parallel. For example, Kalita's connectionist generator produced only simple SVO sentences in various tenses in active and passive forms, with every noun phrase composed of one determiner and one noun, such as *"a monkey has been eating a banana"* (Kalita & Shastri 1987); Gasser's CHIE apparently only produced *"could you take the empty box to the garage?"* (Gasser 1988); Kitano's generator apparently only produced *"I want to attend the conference because I am interested in interpreting telephony"* (Kitano 1990); and Miikkulainen's system apparently only produced sentences of the form *"the boy moved"*, *"the boy ate the chicken"* and sentences produced from them by replacing one noun phrase by a relative clause based on another simple sentence (Miikkulainen 1990).

## 10.2 What Has Been Learned

When building FIG to cope with issues in generation, such as rich inputs and interactions among possible choices it became necessary to abandon the goal of having a structural theory of the relationship between the input and the output (§2.2), and instead to see the task of generation is seen as a creative one, and one not completely constrained by the detailed structure of the generator's input.

The main results of this thesis are the principles of generator design (§2.3), namely that a generator should: be integrated, represent the current state explicitly, be incremental, be parallel, combine evidence numerically, rely on emergent choice, and use feedback.

There are also some findings relating to the use of spreading activation for this task. Certain special functionality is required, such as the locality principle (§4.2.3), the maximum rule (§4.2.4), and the multiplication rule (§3.5). There are several unsolved problems for spreading activation systems capable of dealing with behavior in time (§6.5). Structured connectionist modeling is a valuable research tool, if not yet of practical value, even for generation, which is commonly considered a prototypical symbolic task (Chapter 7).

Finally, there are some negative results. The FIG experiment suggests that modularity is a mistake in generation, that a separate lexicon with complex access procedures is unnecessary, and that two commonly accepted assumptions of the field are unnecessary; to be specific, the views that 'sentences are grammatical, hence they have syntactic structures' and that 'there are many linguistic resources available hence the task of a generator is to choose among them', can be dispensed with, in favor of emergent syntactic choice and syntactic structure. This latter should perhaps not be surprising: after all, the purpose of speaking is to communicate ideas, so it is only reasonable that syntactic structure is at best a 'side-effect' of this process.

## 10.3 Prospects

FIG requires the use of a great deal of world knowledge and language knowledge. Most generators require less, at the cost of reducing the variety of inputs they can handle. FIG also requires a great deal of computation, for the sake of output quality in the form of a consistent set of choices. Thus FIG is not a prototype for practical systems in the near term; rather it is an exploration of the principles necessary for sophisticated generation, looking forward to the time when programs interfacing with generators will be smarter.

There remain decades worth of basic linguistic research to do in lexical semantics, syntax, and so on; and in reasoning and inference more generally. There is also a great deal of work to be done experimenting with computational models of these processes. FIG is but a preliminary step towards a complete, all-purpose, cognitively plausible generator.

# Appendix A

# FIG's Knowledge Network

```
;

;;; -*- Base: 10; Syntax: Common-lisp; Package: CL-USER -*-

;;; file = g:>g>network.lisp

;;; Most of the comments have been stripped out of this file.
;;; Refer to Chapter 5 etc. for more information.

;;; Organization of this file:
;;;   - relations
;;;   - notions (concept) nodes, encoding world knowledge etc.
;;;   - English: syntactic categories, constructions, words
;;;   - Japanese: syntactic categories, constructions, words
;;;   - pseudo-nodes
;;; If a node is commented out, it's because it's not needed for the test suite,
;;;   and is taken out for the sake of speeding things up in \#'clean-eternals.
;;; Node names should have at most 13 characters, so they display concisely.

;;; The network is built up in two passes:
;;;   first make the nodes, then add the links, including inherited ones
;;;   see Section 6.1 and the file "setup.lisp" for details

(setq *all-eternals nil  *non-j-eternals nil  *non-e-eternals nil)
(setq *links-to-process nil)

;;; ==============================================================================
;;; RELATIONS and their collector nodes

;;; The weights listed on relations are for the inebor links among nodes of the input.
;;;   see Section 6.2 and \#'plug and \#'weight-for-slot
;;; The weights here are utterly irrelevant to the nebor links,
;;;   also nebor links are different in that they do not necessarily have inverse links.
;;; Weights to 'case-marking' prepositions all seem to be between .35 and .46

(defr first-memr (weights .10 .00))
(defr second-memr (weights .05 .00))
(defr locationr (weights .1 .1) (rel-collector loc-ness))
(defn loc-ness  (english (loc-settngc .8)(in-w .38))                 ; also at, on ...
                (japanese (de-w .3) (ni1w .25)) )
(defr timer     (weights .5 .2) (rel-collector time-ness))
(defn time-ness (english (time-settngc .8)) (japanese (jtime-settngc 1.)) )
(defr agentr    (weights .25 .15) (rel-collector agent-ness))
(defn agent-ness(english (subj-pred .4) (by-w .4)) (japanese (ga-w .4)) )
(defr causer    (weights .6 .2) (rel-collector causr-ness))    ; deliberately asymmetric
(defn causr-ness(english (makew .3) (per-causativec .3) (subj-pred .2))
                (japanese (ga-w .6)))
```

```
;; weight of .18 slightly less than .2 of patientr
(defr instrumentr (weights .18 .18) (rel-collector instr-ness))
(defn instr-ness  (english (subj-pred .3) (withw .4))  (japanese (de-w .5)))
(defr patientr  (weights .1 .1)  (rel-collector pat-ness))
(defn pat-ness  (english (is2w .3) (passivec .3) (subj-pred .3))
                (japanese (o-w .5)(ni1w .3)(ga-w .3)) )
(defr experiencer (weights .4 .3)  (rel-collector exper-ness))
(defn exper-ness  (english (subj-pred .1)) (japanese (ga-w .8)))
(defr sourcer     (weights .2 .15) (rel-collector sourcer-ness))
(defn sourcer-ness(english (fromw .38))  (japanese (karaw .6)))
(defr destinationr(weights .2 .2)  (rel-collector dest-ness))
(defn dest-ness   (english (to1w .38) (intow .36)) (japanese (e-w .4) (ni1w .2)))
(defr landmarkr   (weights .2 .2)  (rel-collector landm-ness))
(defn landm-ness  (english (towardsw .4)) (japanese (no-hoo-e-w .4)))
;; the practical reason why recipientr is different from destinationr for weights
(defr recipientr (weights .1 .1)  (rel-collector recip-ness))
(defn recip-ness   (english (to1w .48)) (japanese (ni1w .3) ) )
(defr directionr (weights .4 .1))                             ; direction of motion
;; origin of an object --- typically a place
;; weights asymmetric so object precedes both its origin and fromw
(defr originr   (weights .0 .1) (rel-collector orig-ness)) ;alternative: (ratio .8) nil
(defn orig-ness (english  (fromw .35))          (japanese (karaw .45)) )
;; things like "for" and "in order to" not listed
;; to2w gets sem act from the direct link, and syn act via purpose-cla
(defr purposer  (weights .15 .15)  (rel-collector purp-ness))
(defn purp-ness         (english (to2w .45) (purpose-cla .3)) (japanese (ni2w .4)))
;(defr planr    (weights .2 .1) (rel-collector planr-ness))
;(defn planr-ness        (english (purpose-cla .8)) )
(defr mannerr   (weights .3 .1) (rel-collector man-ness))
;; this is tuned so that the mimetic phrase comes after nouns and before the verb
(defn man-ness  (japanese (mannr-motionc .4) (mimetic-phr .35) (jto1w .2)))
(defr motions-mannr (weights .2 .1))
(defr topologyr (weights .2 .2))
(defr contextr  (weights .2 .2))
(defr degreer   (weights (ratio 1.) nil)   (rel-collector degr-ness))
(defn degr-ness (english (intensifyc .25)))
(defr possessor (weights (ratio 1.04) nil) (rel-collector posse-ness))
(defn posse-ness(english (genitivec .4)(apostrophe-s .5))
                (japanese (jgenitivec .5) (no-w .5)))
(defr perceptr  (weights .05 .05) )
(defr sizer     (weights (ratio 1.) nil))    ; could just as well be just 'modifier'
                ;(english (adj-modc .1)))
(defr modifier  (weights (ratio 1.) nil))    ; adj-modc relation, not further specified
                ;(english (adj-modc .1)))
(defr prag-roler (weights .2 .2))            ; currently only ever filled by 'topicn

;;; The following relations are never used to link nodes of the input,
;;;  but only for knowledge in the net (at least so far)
(defr definitenessr)    ; this is inherited by various concepts from thingn
(defr rivalr)           ; label for inhibitory links
(defr subtyper)
(defr supertyper)
(defr methodr)
(defr prerequisiter)

;;; ==========================================================================
;;; NOTIONS --- concepts for world knowledge

;;; The first batch of concepts listed below are not instantiated for inputs,
;;;   though now it doesn't really matter whether they are or not.

;;; genres
(defn openningn (persistentp t)  (english (once-upon-a-w .4)) (japanese (mmukashiw .3)))
(defn fairy-talen (persistentp t)(english (once-upon-a-w .4)) (japanese (mmukashiw .3)))
```

```
(defn slangn (persistentp t)    (english (kick-bucketc .6)) )

;;; property of a sentence/utterance (like a discourse role)
(defn introductoryn(persistentp t)(english (ex-there 1.0))(japanese (jloc-settngc 1.0)))

;;; activation goes from 'topicn, often found in inputs, via 'in-contextn to 'the-w
(defn topicn    (nebors (prerequisiter in-contextn .9))  (japanese (wa-w .7)) )
(defn in-contextn    (english (the-w .5)))

;;; A gross simplification of tense problems ...
;;;   these nodes have no outbound links; they are only used by \#'inflect
;;; Pastn and presentn can also be parts of the input,
;;;   the others are only ever activated by syntactic constructions.
(defn pastn     (persistentp t))
(defn presentn (persistentp t))
(defn bare-n)           ;; bare stem, uninflected
(defn past-partn)       ;; past participle
(defn gerundn)          ;; te-form, in Japanese

;; 'thingn is used for inheritance only (since only 'cat links point to it).
;; Making sure that articles have some semantic-type energy is the ulterior motive.
(defn thingn (cat-collector thing-ness))
(defn thing-ness (nebors (definitenessr a-w .41) (definitenessr the-w .37)))

;; 'regionn is used to modify things which could be landmarks, locations, or regions.
;; 'intow is listed since it is useful when going "into" regions.
;; 'hillsw is listed since that describes a region
;; 'the-w is listed so that we don't get "a hills"; see comment on hillsw
(defn regionn    (english (intow .1) (the-w .2) (hillsw .4) ) )

;;; ------
(defn setn (english (noun-conjc .9) (andw .4))  (japanese (jnoun-conjc .9) (jto2w .5)) )

;(defn planforn)

(defn kissun    (english (kissw .5))  (japanese (kissuw .5)) )
(defn taberun   (english (eatw .5))   (japanese (taberuw .5)) )
(defn mirun     (english (see-w .5))  (japanese (miruw .5)) )
;(defn odorokun) (english (surprise .5))  ;; note: this goes to a 'passive' in English
;(defn kesun     (english (deletew .5)) )
;(defn tsukaun   (english (usew .5)) )
;; The transitive form is treated as the default in Japanese;
;;   there is absolutely no reason for this: could just as easily do it the other way
(defn kowan   (english (breakw .5))(japanese (kowareruw .4)(kowasuw .5)))
(defn agerun  (english (givew .5)) (japanese (ageruw .5)) )
(defn sentakun   (english (wash-clothesw .5)) (japanese (sentakuw .5)))
;(defn sonzain    (english (is1w .5))(japanese (iruw .5)) )
(defn sumun   (english (livew .5))    (japanese (sumuw .5)) )
(defn kierun    (english (vanishw .5)) )
(defn ikun      (english (go-w .5))    (japanese (ikuw .5)) )
;;; supertype= motionn is important for reaching floatc, bobc, and dir-particlec
(defn kurun
  (nebors (supertyper motionn .4))
  (english (comew .55)(towardsw .08)(to1w .05))
  (japanese (kuruw .5)) )

(defn tsukun    (english (get-to-w .6)) (japanese (tsukuw .5)) )

;; rather unpleasant that shinun and korosun are listed as entirely different concepts
(defn korosun    (english (killw .5))    (japanese (korosuw .5)) )
;; see comment on kick-bucketc
(defn shinun    (english (diew .5)(kick-bucketc .4)(kickw .4)(the-w .14)(bucketw .2)))
  ;(japanese (shinuw .5)) )
```

```
(defn hihann   (english (criticizew .5) (criticismw .49)))
(defn gakkarin (english (discouragew .5)))

;---
(defn momon       (cat thingn)  (english (peachw .5)) (japanese (momow .5)))
(defn ringon      (cat thingn)  (english (applew .5)) (japanese (ringow .5)))
(defn keekin      (cat thingn)  (english (cakew .5))  (japanese (keekiw .5)))
(defn saran       (cat thingn)  (english (dishw .5))  (japanese (saraw .5)))
(defn fookun      (cat thingn)  (english (forkw .5))  (japanese (fookuw .5)))
(defn underbrushn (cat thingn)  (english (underbrushw .5)) )
;(defn filen       (cat thingn)  (english (filew .5)))
(defn woodn)

(defn yaman       (cat thingn)
      (english (mountainw .5) (hillw .5) (hillsw .4))   (japanese (yamaw .5)) )
(defn kawan       (cat thingn)
      (english (riverw .4) (streamw .5))    (japanese (kawaw .5)) )

(defn kazen       (cat thingn) (english (windw .5)) (japanese (kazew .5)))
(defn nagaren     (cat thingn) (english (currentw .5)))

;; the senses of ojiisan and obaasan meaning old man and old woman
(defn ojiin  (cat thingn)  (english (old-manw .5))  (japanese (ojiisanw .5)) )
(defn obaan     (cat thingn) (english (old-womanw .5)) (japanese (obaasanw .5)))
;;; These do not inherit from 'thingn since people and places whose names we know
;;;   do not count as things whose definiteness needs to be specified
(defn jonn      (english (johnw .5))    (japanese (jonw .5)))
(defn meerin    (english (maryw .5))    (japanese (meeriw .5)))
(defn shikagon  (english (chicagow .5)) (japanese (shikagow .5)))

;;;  ---
(defn mukashin
      (english  (long-agow .7) (once-upon-a-w .6))
      (japanese (mukashiw .5)  (mmukashiw .4) ) )

(defn aru-hin  (english (one-dayw .5))   (japanese (aru-hiw .5)) )

(defn aru-tokoron (english (somewherew .7) (therew .7))   (japanese (aru-tokorow .5)))

(defn minakamin   (english (upstreamw .4)) (japanese (kawakamiw .6)) )
(defn downn       (english (downw .5)))
;(defn rmn  (english (rmw .5)))

(defn ookiin    (english (bigw .5))   (japanese (ookiiw .5)) )
(defn chiisain  (english (smallw .5)) (japanese (chiisaiw .5)) )
(defn furuin    (english (oldw .5)))
(defn tsuyoin  (japanese (tsuyoiw .5))
  (english (swiftw .5) (highw .5) (strongw .52)))      ; and many other English words

(defn sugokun  (english (veryw .5)))

(defn kyuun  (english (quicklyw .5)) );(japanese (kyuuw .5)))

;;; ------
;;; Below this point is some slightly less trivial world knowledge.

;;; 'shibakarin is really something like a script, but who cares?
;;;   since we aren't going to unify it with anything anyway.
;;;   The names of the relations have no causal role --- they are just memos to myself.
(defn shibakarin
  (japanese(shibakariw .5))
  ;(no english word, gotta do some paraphrase)
  (nebors (purposer hsehld-fueln .4)
          (locationr regionn .2)
```

```
        (methodr gathern .4)
        (methodr cutn .4)
        (patientr underbrushn .4) ) )

(defn hsehld-fueln (cat thingn)   (nebors (purposer gather-woodn .4)))
(defn gathern      (english (gatherw .5))   (nebors (subtyper gather-woodn .4)))
(defn cutn         (english (cutw .5))      (nebors (subtyper cut-woodn .4)))
(defn cut-woodn    (english (chop-woodw .5)) (nebors (patientr woodn .4)))

;;; The incoming links here are be stronger than the outgoing links.
;;;  so that this gets more activated than related concepts,
;;;  since this is rather specific, and an instance of recogntion/concretion.
(defn gather-woodn
  (nebors (methodr gathern .4) (patientr woodn .4)
          (purposer hsehld-fueln .4) (locationr regionn .2) )
  (english (gather-woodw .5)) )

;;; ------
;;; actually, the version of nagareru where the subject is a thing (ie, not water)
(defn nagarerun
  (japanese (nagareruw .45))
  (nebors (locationr in-watern .5)
          (supertyper motionn .3) ) )

(defn in-watern    ;; ie, in or on the water
  (nebors (locationr nagarerun .1)    ; weight is low to prevent non-convergence; un-good
          (locationr floatn .5) ))

(defn on-watern    ;; ie, protruding above surface
  (nebors (locationr floatn .3)  (locationr donburikon .1)) )

(defn motionn   (english  (dir-particlec .5))   (japanese (mannr-motionc .5)) )

(defn floatn
  (nebors (locationr in-watern .3) (locationr on-watern .3) )
  ; this weight is large to make up for weak inference; see comment on kick-bucketc
  (english (floatw 1.1)) )

;;; The activation going via mannern could equally well go via mannerr.
;;;   I do it this way simply to show the possibility of activation to syn constructions
;;;   which is bottom-up via concepts, and not relation-name-sensitive.
(defn donburikon
  (nebors (supertyper mannern .4))
  (english (bobw .8))
  (japanese (donburikow .7)) )

(defn mannern  (japanese (mimetic-phr .4)) )

;;; ------
;; weight to 'malew is low since adjectives for intrinsic properties come late
(defn otokon    (cat thingn) (nebors (subtyper shoonenn .4)) (english (malew .4)))
(defn ko-n      (cat thingn) (nebors (subtyper shoonenn .4)) (english (childw .5)))
(defn shoonenn (cat thingn) (nebors (supertyper otokon .1) (supertyper ko-n .1))
      (english (boyw .5)) (japanese (shoonenw .5)))

;; The weight to 'atsuin is higher than that to 'springn so we get "hot spring"
;;   and not "spring hot".
;;   This is required since the ratio mechanism can't ensure that 'atsuin
;;   has more activation, since there is no link between 'atsuin and 'springn.
;;   See Section 6.5.1.
(defn onsenn (cat thingn) (nebors (modifier atsuin .8) (supertyper springn .5))
      (japanese (onsenw .5)))
;; onsenn is a nebor only for the sake of sem update
(defn atsuin (english (hotw .5)) (nebors (subtyper onsenn .0)))
```

```
(defn springn (cat thingn) (english (springw .5)) (nebors (subtyper onsenn .0)))

;;;======================================
;;; SYNTACTIC CATEGORIES of English, and their collector nodes

;; Constructions which put stuff before the 'head' are immediately relevant,
;;   so the links to them generally have higher weights than the links to
;;   constructions which coordinate words following the head.
;; 'head' here refers to the word, if any, which is a source of activation
;;   to the construction.
;; Categories also determine inflection and are events to flag after a word is said.

(defes noun  (cat-collector noun-ness))
(defen noun-ness (english (prep-phr .65)))

(defes cnoun (supercat noun) (cat-collector cnoun-ness))    ; count noun
(defen cnoun-ness (english (adj-modc .7) (determinatc .71)))
(defes pnoun (supercat noun))                                ; proper noun
(defes mnoun (supercat noun) (cat-collector mnoun-ness))    ; mass noun
(defen mnoun-ness (english (adj-modc .65)))     ; low weight since no articles

(defes verb  (cat-collector verb-ness))
(defen verb-ness   (english (subj-pred .1) (purpose-cla .1)))

;; Verbs taking sentential complements.
;; This is a special type only so that it can be used as an event (for subj-pred)
;; An alternative would be to make it provide the inheritance of a construction,
;;   as with 'tverb and 'transitivec.
(defes scomp-verb  (supercat verb))

(defes tverb (cat-collector tverb-ness) (supercat verb))
(defen tverb-ness (english (transitivec .4)))
(defes lc-verb (cat-collector lc-verb-ness) (supercat tverb))
(defen lc-verb-ness (english (lex-causativec .4)))

(defes st-ch-verb (cat-collector st-ch-vb-ness) (supercat verb))
(defen st-ch-vb-ness (english (state-changec .4)))
;(defes copula)

(defes adjective  (cat-collector adj-ness))
(defen adj-ness  (english (intensifyc .56)))

(defes preposition (cat-collector prep-ness))
(defen prep-ness (english (prep-phr .2)))  ; potentially dangerous cycle
(defes article)
(defes iadverb)          ; intensifying adverb, eg 'very'
(defes sadverb)          ; sentential adverb, sentence-final
(defes time-adverb)      ; sentential adverb, sentence-initial
(defes loc-adverb)       ; sentential adverb, sentence-initial

(defes vparticle)        ; verb particles
(defes conjunction)      ; this does absolutely nothing at present

;; Used for choosing between "a" and "an", in concert with \#'inflect
(defes vowel-inits (cat-collector voweli-ness .1))
(defen voweli-ness)
(defes conso-inits (cat-collector consoi-ness .1))
(defen consoi-ness)

;;; ==========================================================================
;;; CONSTRUCTIONS --- English --- documented in Section 5.2

(defec adj-modc                            ; adjectival modification
  (constituents (am-1  adjective ((adjective .55))) ))
```

```
;               (am-2  cnoun  )))       ; commented out to allow several adjectives

;; The semantics underlying articles is not really there,
;;   this leads to different handling of the act from eg 'momon to "the-w" and to "bigw"
;;   via an 'nebor and an 'inebor link, respectively, which is ugly.
(defec determinatc
  (constituents (dt-1  (or article apostrophe-s)  ((article .57)(genitivec .15)))
                (dt-2  cnoun  )))

(defec prep-phr
  (constituents (pp-1  preposition ((preposition .6)))
                (pp-2  noun  )))

(defec intensifyc
  (constituents (in-1  iadverb  ((iadverb .48)))
                (in-2  adjective  )))

(defec genitivec
  (constituents (ge-1  possessor     ((noun .15)))       ; noun to beat out article
                (ge-2  apostrophe-s  ((apostrophe-s .5))) ))

(defec noun-conjc
  (constituents (ac-1 first-memr  ((first-memr .2)))
                (ac-2 andw        ((andw .6)))
                (ac-3 second-memr ((second-memr .5))) ))

(defec ex-there
  (constituents (et-1  therew ((therew .5)))
                (et-2  verb   ((verb .5)))               ; rather too general
                (et-3  noun    ) ))

(defec subj-pred
  (reset-if scomp-verb)
  (constituents (sp-1  (and noun (or (never first-memr) second-memr) (not possessor))
                       ((causer .35)(agentr .25)(instrumentr .15)(preposition -.5)) )
                (sp-2  verb  ((verb .22)))
                (sp-3  sleep) ) )

(defec per-causativec
  (constituents (pcc-1 causer )                      ; causer in synergy with sp-1
                (pcc-2 makew  ((makew .75)))         ; makes subj-pred reset
                (pcc-3 verb ((bare-n 1.6))) ))       ; clause in synergy with sp-1

;; Re the rivalry with per-causativec:
;;   This receives activation from lc-verbs whereas per-causativec receives
;;   activation only from the meaning (causer).  If this construction is not
;;   activated, makew gets piles of syntactic activation, and wins.
;;   But if this construction is active, it gives syn activation to lc-verbs,
;;   and thus reduces the syntactic-act advantage of makew,
;;   and the semantic-act advantage of 'breakw, or whatever, shines through.
;; The subject is due to synergy with subj-pred;
;;   the direct-object is due to synergy with transitivec.
(defec lex-causativec
  (constituents (lcc-1  (or causer agentr)  )
                (lcc-2  verb ((lc-verb .2))) ) )

(defec passivec
  (constituents (pa-1 (and patientr (never verb))  )
                (pa-2 is2w   ((is2w .7)))            ; 'verb in synergy with subj-pred
                (pa-3 verb   ((past-partn 1.0)(verb .5)))
                (pa-4 by-w   ((by-w .5))) ))         ; in synergy with prep-phr

(defec state-changec
  (constituents (csc-1  (and patientr (never verb))  )
```

```
                (csc-2  st-ch-verb  ((st-ch-verb .2))) ))

;; Would 'direct-objc' be a better name for this construction?
(defec transitivec
  (constituents (tv-1 (and tverb (never patientr)))       ; transitive and not passive
                (tv-2 noun  ((patientr .3)(preposition -.4)))
                (tv-3 sleep) ) )

;;; ---
;;; Here are some verb valences not yet handled by proper generalizations
(defec comec
  (constituents (cc-1  comew )
                (cc-2  destinationr ((destinationr .2)))))
(defec go-c
  (constituents (gv-1  go-w      ((go-w .2)))
                (gv-2  noun      ((destinationr .25))) ))
(defec get-to-c
  (constituents (gc-1  get-to-w  )
                (gc-2  noun     ((destinationr .1)(preposition -.5))) ) )

;;; ---
(defec dir-particlec
  (constituents (mvp-1  verb )
                (mvp-2  vparticle  ((directionr .6)(vparticle .6))) ))

(defec purpose-cla
  (constituents (pc-1  to2w  ((to2w .7)))
                (pc-2  verb  ((verb .2)(bare-n 1.0))) ))

(defec time-settngc
  (constituents (ts-1 time-adverb ((time-adverb .7)))))

(defec loc-settngc
  (constituents (ls-1 loc-adverb ((loc-adverb .6)))))

;;; ------
;; What should really happen, is that there should be a kick-the-bucket metaphor,
;;   whose activation level is given by the product of act. from shinun and slangn.
;;   This would avoid all sorts of ugliness.
;;   However this would be a serious performance hit,
;;   since the test would have to go in \#'combine-evidence.
;;   Also, the metaphor would have to list the links across which it receives
;;   activation, and whether to multiply, which would complicate the way that
;;   the network is defined.
;;   Multiplication would be useful for recognizing concepts more generally, eg floatn.
;; Note that causer and verb are also ok to precede kb-1
(defec kick-bucketc
  (constituents (kb-1   (and patientr (never agentr)) )
                (kb-2  kickw    ((kickw .25)) )        ; inflected by general principles
                (kb-3  bucketw  ((the-w 1.5)(bucketw .85) (diew -.3) (kickw -.4))) ))

;;;================================================
;;; WORDS --- English

(defew wash-clothesw (cat verb) (expresses sentakun)
       (grapheme (inf "wash clothes")(past "washed clothes")))
(defew gather-woodw  (cat verb) (expresses gather-woodn)
       (grapheme (inf "gather wood")(past "gathered wood")))
(defew gatherw       (cat verb) (expresses gathern)
       (grapheme (inf "gather") (past "gathered")))
(defew cutw          (cat tverb)(expresses cutn)
       (grapheme (inf "cut") (past "cut")))
(defew chop-woodw    (cat tverb)(expresses cut-woodn)
       (grapheme (inf "chopped wood") (past "chopped wood")))
```

```
(defew go-w     (cat verb)  (expresses ikun)  (valence (go-c .2))
      (grapheme (inf "go") (past "went") (pastp "gone") (presp "going")) )
(defew comew    (cat verb)  (expresses kurun) (valence (comec .2))
      (grapheme (inf "come") (past "came")))
(defew get-to-w   (cat verb)  (expresses tsukun) (valence (get-to-c .2))
      (grapheme (inf "get to") (past "got to")))

(defew floatw     (cat verb)  (expresses floatn)
      (grapheme (inf "float") (past "floated") (presp "floating")))
(defew bobw       (cat verb)  (expresses donburikon in-watern on-watern)
      (grapheme (inf "bob")(past "bobbed")(presp "bobbing")))

(defew kissw      (cat tverb) (expresses kissun)  (grapheme (inf "kiss")(past "kissed")))
(defew killw      (cat tverb) (expresses korosun) (grapheme (inf "kill")(past "killed")))
(defew eatw       (cat tverb) (expresses taberun)
      (grapheme (inf "eat") (past "ate") (pastp "eaten")))
(defew see-w      (cat tverb) (expresses mirun)
      (grapheme (inf "see") (past "saw")(pastp "seen")))
;(defew deletew (cat tverb) (expresses kesun) (grapheme (inf "delete")(past "deleted")))
;(defew usew     (cat tverb) (expresses tsukaun)(grapheme (inf "use") (past "used")))

(defew kickw    (cat tverb)  (grapheme (inf "kick") (past "kicked")))
(defew givew    (cat tverb)  (expresses agerun)
      (grapheme (inf "give") (past "gave") (pastp "given")))

(defew diew     (cat st-ch-verb) (expresses shinun) (grapheme (inf "die")(past "died")))
(defew vanishw  (cat st-ch-verb) (expresses kierun)
      (grapheme (inf "vanish")(past "vanished")))

(defew breakw   (cat st-ch-verb lc-verb)
      (expresses kowan)  (grapheme (inf "break") (past "broke")(pastp "broken")) )

;; there should be a livedc specifying that a location is required: see OnCG sec 5.4
;; note that livew is not an st-ch-verb,
;;   in part because sumun takes an experiencer, not a patientr (ugly)
(defew livew    (cat verb)   (expresses sumun)  (grapheme (inf "live") (past "lived")))
;(defew isiw     (cat copula) (expresses sonzain)  (grapheme (inf "is ") (past "was ")))

(defew discouragew (cat lc-verb)  (expresses gakkarin)
      (grapheme (inf "discourage") (past "discouraged")))

(defew criticismw (cat mnoun)  (expresses hihann) (grapheme "criticism"))
(defew criticizew (cat tverb) (expresses hihann)
      (grapheme (inf "criticize") (past "criticized")))

;;; ------
(defew peachw     (cat cnoun conso-inits)(expresses momon)  (grapheme "peach"))
(defew applew     (cat cnoun vowel-inits)(expresses ringon) (grapheme "apple"))
(defew cakew      (cat cnoun conso-inits)(expresses keekin) (grapheme "cake"))
(defew dishw      (cat cnoun conso-inits)(expresses saran)  (grapheme "dish"))
(defew forkw      (cat cnoun conso-inits)(expresses fookun) (grapheme "fork"))
(defew underbrushw(cat cnoun conso-inits)(expresses underbrushn)(grapheme "underbrush"))
;(defew filew      (cat cnoun conso-inits)(expresses filen)  (grapheme "file"))
;; Listing 'bucketw as expressing 'shinun is rather unpleasant.
(defew bucketw    (cat cnoun conso-inits)(expresses shinun) (grapheme "bucket"))

(defew malew      (cat adjective conso-inits)(expresses otokon) (grapheme "male"))
(defew childw     (cat cnoun conso-inits)(expresses ko-n)   (grapheme "child"))
(defew boyw       (cat cnoun conso-inits)(expresses shoonenn) (grapheme "boy"))
(defew manw       (cat cnoun conso-inits)(expresses otokon) (grapheme "man"))
(defew old-manw   (cat cnoun vowel-inits)(expresses ojiin)  (grapheme "old man"))
(defew old-womanw(cat cnoun vowel-inits)(expresses obaan)  (grapheme "old woman"))
```

```
(defew streamw   (cat cnoun conso-inits)(expresses kawan)  (grapheme "stream"))
(defew riverw    (cat cnoun conso-inits)(expresses kawan)  (grapheme "river"))

(defew mountainw (cat cnoun conso-inits)(expresses yaman)  (grapheme "mountain"))
(defew hillw     (cat cnoun conso-inits)(expresses yaman)  (grapheme "hill"))
(defew springw   (cat cnoun conso-inits)    (expresses springn)(grapheme "spring"))

(defew windw     (cat cnoun conso-inits) (expresses kazen)   (grapheme "wind")
                 (english (highw .05)))
(defew currentw  (cat cnoun conso-inits) (expresses nagaren) (grapheme "current")
                 (english (swiftw .05)))

(defew johnw     (cat pnoun) (expresses jonn)      (grapheme "John") )
(defew maryw     (cat pnoun) (expresses meerin)    (grapheme "Mary") )
(defew chicagow  (cat pnoun) (expresses shikagon) (grapheme "Chicago") )
;(defew rmw      (cat pnoun) (expresses rmn)       (grapheme "rm") )

;; there is as yet nothing to say that plural nouns take 'the' or 'some' but not 'a'
;;   hence a kludge on 'regionn
(defew hillsw    (cat cnoun) (expresses yaman)    (grapheme "hills") )

;;; ------
;; Saying that prepositions express relations only affects semantic update
;;   nothing more, since relations are not explicitly part of the input;
;;   This is not done for Japanese particles ( post-positions), since it is unnecessary,
;;   since they _follow_ the filler of the relation they mark (Section 5.3).
;; Prepositions of motion express the motion notion.
(defew to1w    (grapheme "to")      (cat preposition)
                (expresses destinationr kurun motionn))
(defew intow   (grapheme "into")    (cat preposition)(expresses destinationr))
(defew fromw   (grapheme "from")    (cat preposition)(expresses destinationr motionn))
(defew in-w    (grapheme "in")      (cat preposition)(expresses locationr))
(defew towardsw(grapheme "towards")(cat preposition)(expresses landmarkr kurun motionn))
(defew by-w    (grapheme "by")      (cat preposition)(expresses agentr))  ; for passives
;; 'withw is listed as expressing 'instr-ness to avoid "... eat with a boy 's with fork"
;;   basically it ensures that, a preposition, once said, won't pop out again.
;;   similar express links should be given to all prepositions, and postpositions ?
(defew withw   (grapheme "with")    (cat preposition) (expresses instrumentr instr-ness))

(defew hotw    (cat adjective conso-inits) (expresses atsuin)  (grapheme "hot"))
(defew bigw    (cat adjective conso-inits) (expresses ookiin)  (grapheme "big"))
(defew smallw  (cat adjective conso-inits) (expresses chiisain)(grapheme "small"))
(defew oldw    (cat adjective vowel-inits) (expresses furuin)  (grapheme "old"))
(defew strongw (cat adjective conso-inits) (expresses tsuyoin) (grapheme "strong"))
(defew swiftw  (cat adjective conso-inits) (expresses tsuyoin) (grapheme "swift"))
(defew highw   (cat adjective conso-inits) (expresses tsuyoin) (grapheme "high"))

(defew a-w   (cat article)   (grapheme (vowel "an") (consnt "a")))
(defew the-w (cat article)   (grapheme "the") )

;;; ------
(defew once-upon-a-w (cat time-adverb)(grapheme "once upon a time")(expresses mukashin))
(defew long-agow     (cat time-adverb)(grapheme "long ago")  (expresses mukashin))
(defew one-dayw      (cat time-adverb)(grapheme "one day")   (expresses aru-hin))

(defew somewherew (cat loc-adverb)  (grapheme "somewhere")   (expresses aru-tokoron))
(defew therew     (cat loc-adverb)  (grapheme "there")
      (expresses aru-tokoron introductoryn))

(defew downw (cat vparticle) (grapheme "down") (expresses downn))

;; this is listed as a noun only since it appears as the object of a preposition,
;;   as in "from upstream"
(defew upstreamw (cat pnoun) (expresses minakamin) (grapheme "upstream") )
```

```
(defew to2w  (grapheme "to ") )                    ;; the "to" in purpose clauses

;; the "was" in passivec
;; this must be a verb in the way that it inflects,
;; for 'subj-pred, "was" counts as the verb in passives,
;;   but really maybe "was killed" or "killed" should count as the verb
(defew is2w (cat verb)(grapheme (inf "be") (past "was") (pres "is") (pastp "has been")))

;; the "make" of causatives
(defew makew  (cat scomp-verb)  (grapheme (inf "make") (past "made") ))

(defew quicklyw (cat sadverb)    (grapheme "quickly") (expresses kyuun) )
(defew veryw    (cat iadverb conso-inits)    (grapheme "very")    (expresses sugokun) )

(defew andw   (cat conjunction) (grapheme "and")     (expresses setn))

;; This is a word, rather than being handled by inflection,
;;  in part so that i can easily raise a flag for determinatc.
(defew apostrophe-s  (grapheme "'s") (expresses possessor))

;;; ===========================================================================
;;; SYNTACTIC CATEGORIES of Japanese, and their collector nodes

(defjs jnoun (cat-collector jnoun-ness))
(defjn jnoun-ness (japanese (jadj-modc .65)(jnoun-partc .2)))

(defjs jverb)
(defjs ji-verb (supercat jverb) (cat-collector jiverb-ness))
(defjn jiverb-ness (japanese (jintransitivec .4)))
;(defjs fverb (japanese (verb-final .3)))    ;; verbs adopted from foreign languages
(defjs jparticle)
(defjs jadj)
(defjs jtime-adv)

(defjs mimetic (cat-collector mimetic-ness))
(defjn mimetic-ness (japanese (mimetic-phr .1)))

;;; ===========================================================================
;;; CONSTRUCTIONS --- Japanese --- documented in Section 5.3

(defjc jnoun-partc
  (constituents (jn-1 jnoun     )
                (jn-2 jparticle ((jparticle 1.4))) ))

(defjc jadj-modc
  (constituents (ja-1 jadj   ((jadj .63))) ))

(defjc jgenitivec
  (constituents (jg-1 possessor ((jnoun .1)))
                (jg-2 no-w  ((no-w .5))) ))

(defjc jnoun-conjc     ; noun conjunction
  (constituents (tc-1 first-memr  ((first-memr .2)))
                (tc-2 jto2w       ((jto2w .6)))
                (tc-3 second-memr ((second-memr .3))) ))

(defjc jintransitivec
  (constituents (kc-1 (and patientr (never agentr))  )
                (kc-2 ga-w       ((ga-w .5)))
                (kc-3 ji-verb    ((ji-verb .5))) ) )

;;; Constructions which implement the relation-case mappings,
;;;   overriding the defaults which are given by semantics.
```

```
(defjc patient-ni
  (constituents (pn-1 patientr     )
                (pn-2 niiw ((niiw .6))) ) )    ; not "o"
(defjc dest-ni
  (constituents (dn-1 destinationr     )
                (dn-2 niiw ((niiw .6))) ) )    ; not "e"
(defjc loc-ni
  (constituents (ln-1 locationr     )
                (ln-2 niiw ((niiw .8))) ) )    ; not "de"

(defjc mannr-motionc
  (constituents (mm-1 jverb ((motions-mannr .25) (gerundn 2.)))
                (mm-2 jverb      ) ) )          ; will be a motion verb

(defjc mimetic-phr
  (constituents (mc-1 mimetic ((mimetic .8)))
                (mc-2 jtoiw   ((jtoiw .8))) ) )

(defjc jtime-settngc
  (constituents (jt-1 jtime-adv  ((jtime-adv 1.2))) ))

(defjc jloc-settngc
  (constituents (jl-1 locationr ((locationr 1.))) ))

;;; ==========================================================================
;;; WORDS --- Japanese

(defjw momow    (cat jnoun) (grapheme "momo")    (expresses momon))
(defjw ringow   (cat jnoun) (grapheme "ringo")   (expresses ringon))
(defjw keekiw   (cat jnoun) (grapheme "keeki")   (expresses keekin))
(defjw saraw    (cat jnoun) (grapheme "sara")    (expresses saran))
(defjw fookuw   (cat jnoun) (grapheme "fooku")   (expresses fookun))
(defjw kawaw    (cat jnoun) (grapheme "kawa")    (expresses kawan))
(defjw yamaw    (cat jnoun) (grapheme "yama")    (expresses yaman))
(defjw onsenw   (cat jnoun) (grapheme "onsen")   (expresses onsenn))
(defjw kazew    (cat jnoun) (grapheme "kaze")    (expresses kazen))
(defjw ojiisanw (cat jnoun) (grapheme "ojiisan") (expresses ojiin))
(defjw shoonenw (cat jnoun) (grapheme "shoonen") (expresses shoonenn))
(defjw obaasanw (cat jnoun) (grapheme "obaasan") (expresses obaan))
(defjw jonw     (cat jnoun) (grapheme "jon")     (expresses jonn))
(defjw meeriw   (cat jnoun) (grapheme "meeri")   (expresses meerin))
(defjw shikagow (cat jnoun) (grapheme "shikago") (expresses shikagon))

;(defjw iruw      (cat jverb) (expresses sonzain)
;                (grapheme (past "imashita") (pres "imasu")))
(defjw sumuw     (cat jverb) (expresses sumun)  (valence (loc-ni .2))
                (grapheme (past "sunde imashita") (pres "sunde imasu")))
(defjw ikuw      (cat jverb) (expresses ikun)
                (grapheme (past "ikimashita")(pres "ikimasu")))
(defjw kuruw     (cat jverb) (expresses kurun)
                (grapheme (past "kimashita")(pres "kimasu")(gerund "kite")))
(defjw tsukuw    (cat jverb) (expresses tsukun)  (valence (dest-ni .3))
                (grapheme (past "tsukimashita")(pres "tsukimasu")) )
(defjw ageruw    (cat jverb) (expresses agerun)  (valence (dest-ni .2))
                (grapheme (past "agemashita") (pres "agemasu")) )
(defjw korosuw   (cat jverb) (expresses korosun)
                (grapheme (past "koroshimashita") (pres "koroshimasu")))
(defjw kowasuw   (cat jverb) (expresses kowan)
                (grapheme (past "kowashimashita") (pres "kowashimashita")))
(defjw kowareruw (cat ji-verb) (expresses kowan)
                (grapheme (past "kowaremashita") (pres "kowaremasu")) )
(defjw taberuw   (cat jverb) (expresses taberun)
                (grapheme (past "tabemashita") (pres "taberu")) )
(defjw miruw     (cat jverb) (expresses mirun)
```

```
                 (grapheme (past "mimashita") (pres "miru")) )
(defjw kissuw     (cat jverb) (expresses kissun) (valence (patient-ni .2))
                 (grapheme (pres "kissu shimasu") (past "kissu shimashita")) )
(defjw nagareruw (cat jverb) (expresses nagarerun)
                 (grapheme (pres "nagareru")(past "nagareta")(gerund "nagarete")) )

(defjw shibakariw (cat jnoun)  (expresses shibakarin) (grapheme "shibakari"))
(defjw sentakuw   (cat jnoun)  (expresses sentakun) (grapheme "sentaku"))

;;; Particles choice is governed by general principles of relation-case mapping,
;;;   and also by verb valences.
;;; Currently wa supplants all particles, instead of just ga, o, and ni
(defjw ga-w (cat jparticle) (grapheme "ga"))
(defjw wa-w (cat jparticle) (grapheme "wa"))
(defjw e-w  (cat jparticle) (grapheme "e"))
(defjw no-hoo-e-w  (cat jparticle) (grapheme "no hoo e"))
(defjw ni1w (cat jparticle) (grapheme "ni")) ; location, destination, recipient, patient
(defjw ni2w (cat jparticle) (grapheme "ni ")); purpose
(defjw karaw(cat jparticle) (grapheme "kara"))
(defjw o-w  (cat jparticle) (grapheme "o"))
(defjw de-w (cat jparticle) (grapheme "de"))

(defjw jto1w  (grapheme " to") )                    ; for mimetics
;; these two have category jparticle merely to advance jnoun-partc
(defjw jto2w (cat jparticle) (grapheme " to ") (expresses setn))
(defjw no-w  (cat jparticle) (grapheme "no") )

(defjw ookiiw   (cat jadj)  (grapheme "ookii")    (expresses ookiin))
(defjw chiisaiw (cat jadj)  (grapheme "chiisai")  (expresses chiisain))
(defjw tsuyoiw  (cat jadj)  (grapheme "tsuyoi")   (expresses tsuyoin))

(defjw mmukashiw   (cat jtime-adv) (grapheme "mukashi mukashi")(expresses mukashin))
(defjw mukashiw    (cat jtime-adv) (grapheme "mukashi")     (expresses mukashin))
;; these two should be compositional, probably
(defjw aru-hiw     (cat jtime-adv) (grapheme "aru hi")      (expresses aru-hin))
(defjw aru-tokorow (cat jnoun)     (grapheme "aru tokoro")  (expresses aru-tokoron))
(defjw kawakamiw   (cat jnoun)     (grapheme "kawakami")    (expresses minakamin))

(defjw donburikow  (cat mimetic) (grapheme "donburiko donburako")(expresses donburikon))

;;; ========================================================================
;;; PSEUDO-nodes --- documented in Sections 5.1, 6.2, 6.5 and 6.6

(defx permanentx
  (english (noun 10) (verb 9) (sadverb 6))
  (japanese (jnoun 12) (jverb 9) (agentr 5)) )

(defx sourcex)

(defx in-progressx)
(defx coveredx)
(defx in-focusx)

;;; ========================================================================
(process-links)
;;; ========================================================================
;
```

# Bibliography


Aitchison, Jean (1987). *Words in the Mind: An Introduction to the Mental Lexicon.* Basil Blackwell, Oxford and New York.

Ajjanagadde, Venkat & Lokendra Shastri (1989). Efficient Inference with Multi-Place Predicates and Variables in a Connectionist System. In *Program of the Eleventh Annual Conference of the Cognitive Science Society.*

Andre, Edgar (1985). Aspects of Translating and Conference Interpreting. In G. Debusscher & J. P. van Noppen, editors, *Communiquer et Traduire: Hommages à Jean Dierickx.* Editions de l'Université de Bruxelles, Belgium.

Appelt, Douglas E. (1985). *Planning English Sentences.* Cambridge University Press.

Arnold, Doug & Louisa Sadler (1990). The Theoretical Basis of MiMo. *Machine Translation*, 5.

Baars, Bernard K. (1980). The Competing Plans Hypothesis: an heuristic viewpoint on the causes of errors in speech. In Hans W. Dechert & Manfred Raupach, editors, *Temporal Variables in Speech.* Mouton.

Becker, J. D. (1975). The Phrasal Lexicon. In Roger Schank & Bonnie L. Webber, editors, *Theoretical Issues in Natural Language Processing*, Cambridge, Massachusetts.

Bock, J. Kathryn (1982). Toward a Cognitive Psychology of Syntax: Information Processing Contributions to Sentence Formulation. *Psychological Review*, 89(1).

Bock, J. Kathryn (1986). Syntactic Persistence in Language Production. *Cognitive Psychology*, 18.

Bock, J. Kathryn (1987). Exploring Levels of Processing In Sentence Production. In Gerard Kempen, editor, *Natural Language Generation.* Nijhof. distributed by Kluwer.

Butterworth, Brian (1980). Evidence from Pauses in Speech. In Brian Butterworth, editor, *Language Production, Volume 1: Speech and Talk.* Academic Press.

Carbonell, Jaime G., Richard E. Cullingford, & Anatole V. Gershman (1981). Steps Toward Knowledge-Based Machine Translation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(4).

Chafe, Wallace L. (1980a). The Deployment of Consciousness in the Production of a Narrative. In Wallace L. Chafe, editor, *The Pear Stories.* Ablex.

Chafe, Wallace L. (1980b). Some Reasons for Hesitating. In Hans W. Dechert & Manfred Raupach, editors, *Temporal Variables in Speech.* Mouton.

Chafe, Wallace L. (1985). Linguistic Differences Produced by Differences Between Speaking and Writing. In A. Hildyard D. R. Olson, N. Torrance, editor, *Language, Literacy, and Learning.* Cambridge University Press.

Chin, David N. (1988). *Intelligent Agents as a Basis for Natural Language Interfaces.* PhD thesis, University of California at Berkeley, Computer Science Division. also technical report UCB/CSD 88/396.

Clippinger, John H. (1977). *Meaning and Discourse: a computer model of psychoanalytic speech and cognition.* Johns Hopkins University Press.

Cullingford, Richard E. (1986). *Natural Language Processing.* Rowman and Littlefield, Totowa, New Jersey.

Cumming, Susanna (1986). The Lexicon in Text Generation. Technical Report ISI/RR-86-168, Information Sciences Institute, Marina del Rey, California.

Cutler, A. & S. D. Isard (1980). The Production of Prosody. In Brian Butterworth, editor, *Language Production, Volume 1: Speech and Talk.* Academic Press.

Danlos, Laurence (1984). Conceptual and Linguistic Decisions in Generation. In *Proceedings 10th COLING.*

Danlos, Laurence (1987). *The Linguistic Basis of Text Generation.* Cambridge University Press.

Danlos, Laurence & Fiammetta Namer (1988). Morphology and Cross Dependencies in the synthesis of personal pronouns in Romance languages. In *Proceedings 12th COLING.*

Davey, Anthony (1978). *Discourse Production.* Edinburgh University Press.

De Smedt, Koenrad J.M.J. (1990). Incremental Sentence Generation: a computer model of grammatical encoding. Technical Report 90-01, Nijmegen Institute for Cognition Research and Information Technology.

Dell, Gary S. (1986). A Spreading Activation Theory of Retrieval in Sentence Production. *Psychological Review*, 93(3).

Dell, Gary S. (1989). The Retrieval Of Phonological Forms in Production: Tests of Predictions from a Conectionist Model. In William Marslen-Wilson, editor, *Lexical Representation and Process.* MIT Press.

Evens, Martha W., editor (1988). *Relational Models of the Lexicon.* Cambridge University Press.

Feiner, Steven K. & Kathleen R. McKeown (1990). Coordinating Text and Graphics in Explanation Generation. In *Proceedings of the Eighth National Conference on Artificial Intelligence.*

Feldman, Jerome A., Mark A. Fanty, Nigel H. Goddard, & Kenton J. Lynne (1988). Computing with Structured Connectionist Networks. *Communications of the ACM*, 31.

Fillmore, Charles (1978). On the Organization of Semantic Information in the Lexicon. In *Chicago Linguistics Society Parasession on the Lexicon.*

Fillmore, Charles (1985). Frames and the Semantics of Understanding. *Quaderni di Semantica*, IV(2).

Fillmore, Charles (1989). On Grammatical Constructions. course notes, UC Berkeley Linguistics Department.

Fillmore, Charles, Paul Kay, & M. C. O'Connor (1988). Regularity and Idiomaticity in Grammatical Constructions: The Case of Let Alone. *Language*, 64(3).

Fillmore, Charles J. (1988). The Mechanisms of "Construction Grammar". In *Proceedings of the 14th Berkeley Linguistic Society.*

Finkler, Wolfgang & Günter Neumann (1989). POPEL-HOW: A Distributed Parallel Model for Incremental Natural Language Production with Feedback. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.*

Friedman, Joyce (1969). Directed Random Generation of Sentences. *Communications of the ACM*, 12(6).

Garrett, M. F. (1975). The Analysis of Sentence Production. In Gordon Bower, editor, *The Psychology of Learning and Motivation, Volume 9.* Academic Press.

Gasser, Micheal (1988). A Connectionist Model of Sentence Generation in a First and Second Language. Technical Report UCLA-AI-88-13, University of California, Los Angeles.

Gerloff, Pamela A. (1988). *From French to English: A look at the translation process in students, bilinguals, and professional translators.* PhD thesis, Harvard University, School of Education.

Gerver, David (1976). Empirical Studies of Simultaneous Interpretation: A Review and a Model. In Richard Brislin, editor, *Translation: Applications and Research.* Gardener Press, New York. distributed by Wiley.

Goldman, Neil (1974). *Computer Generation of Natural Language from a Deep Conceptual Base.* PhD thesis, Stanford University. also Technical Report CS-74-461.

Harley, Trevor A. (1984). A Critique of Top-down Independent Levels Models of Speech Production: Evidence from Non-plan-internal Speech Errors. *Cognitive Science*, 8.

Harley, Trevor A. (1990). Paragrammaticisms: Syntactic Disturbance or Breakdown of Control. *Cognition*, 34(1).

Hasida, Kôichi, Shun Ishizaki, & Hitoshi Isahara (1988). An Approach to Abstract Generation. *Bulletin of the Electrotechnical Laboratory*, 52(4).

Holmes, V. M. (1988). Hesitation and Sentence Planning. *Language and Cognitive Processes*, 3(4).

Hovy, Eduard (1988). *Generating Natural Language Under Pragmatic Constraints.* Lawrence Erlbaum Associates.

Iida, Hitoshi (1990). Intention Translation Method: A Spoken Dialog Translation System Using a Lexicon-Driven Grammar. In *Overview of ATR Basic Researches into Telephone Interpretation.* ATR. Technical Report TR-I-0184.

Ishizaki, Shun (1988). Generating Japanese Text from Conceptual Representation. In David D. McDonald & Leonard Bolc, editors, *Natural Language Generation Systems.* Springer-Verlag.

Jacobs, Paul (1985a). *A Knowledge-Based Approach to Language Generation.* PhD thesis, University of California at Berkeley, Computer Science Division. also report UCB/CSD 86/254.

Jacobs, Paul (1985b). PHRED: A generator for natural language interfaces. Technical Report 85/198, University of California at Berkeley, Computer Science Division. revised versions appear in *Computational Linguistics* 11, October 1985, and in *Natural Language Generation Systems*, David McDonald and Leonard Bolc, eds., Springer Verlag, 1988.

Jacobs, Paul (1988). Why Text Planning Isn't Planning. In *Proceedings of the AAAI Workshop on Text Planning and Realization.*

Joshi, Arvind K. (1987). The Relevance of Tree Adjoining Grammar to Generation. In Gerard Kempen, editor, *Natural Language Generation.* Nijhof. distributed by Kluwer.

Jurafsky, Daniel (1988). Issues in Relating Syntax and Semantics. In *Proceedings 12th COLING.*

Jurafsky, Daniel (1990). Representing and Integrating Linguistic Knowledge. In *Proceedings 13th COLING.*

Kalita, Jugal & Lokendra Shastri (1987). Generation of Simple Sentences in English Using the Connectionist Model of Computation. In *Program of the Ninth Annual Conference of the Cognitive Science Society.* Lawrence Erlbaum Associates.

Kanwisher, Nancy G. (1990). Binding and Type-Token Problems in Human Vision. In *Program of the Twelfth Annual Conference of the Cognitive Science Society.* Lawrence Erlbaum Associates.

Kay, Martin (1984). Functional Unification Grammar: a formalism for machine translation. In *Proceedings 10th COLING.*

Kelley, L. G. (1979). *The True Interpreter: A History of Translation Theory and Practice in the West.* St. Martin's Press, New York.

Kempen, Gerard (1987). A Framework for Incremental Syntactic Tree Formation. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.*

Kempen, Gerard (1989). Language Generation Systems. In István S. Bátori, Winfried Lenders, & Wolfgang Putschke, editors, *Computational Linguistics: An International Handbook.* Walter de Gruyter, Berlin and New York.

Kempen, Gerard & Edward Hoenkamp (1987). An Incremental Procedural Grammar for Sentence Formulation. *Cognitive Science,* 11.

Kitano, Hiroaki (1990). Parallel Incremental Sentence Production for a Model of Simultaneous Interpretation. In Robert Dale, Chris Mellish, & Micheal Zock, editors, *Current Research in Natural Language Generation.* Academic Press.

Klöck, Erwin (1988). Utterance Generation Without Choice. Technical Report FKI-92-88, Technische Universität München, Institut für Informatik.

Kukich, Karen (1988). Fluency in Natural Language Reports. In David D. McDonald & Leonard Bolc, editors, *Natural Language Generation Systems.* Springer-Verlag.

Lakoff, George (1987). *Women, Fire, and Dangerous Things.* University of Chicago Press.

Lakoff, George (1988). A Suggestion for a Linguistics with Connectionist Foundations. In *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann.

Lashley, Karl S. (1951). The Problem of Serial Order in Behavior. In L. A. Jeffress, editor, *Cerebral Mechanisms in Behavior.* John Wiley and Sons. reprinted in *The Neuropsychology of Lashley,* McGraw Hill, 1960.

Levelt, W. J. M. (1989). *Speaking: from intention to articulation.* MIT Press.

Lindsley, James R. (1975). Producing Utterances: How Far Ahead Do We Plan? *Cognitive Psychology,* 7.

Maclay, Howard & Charles E. Osgood (1967). Hesitation Phenomena in Spontaneous English Speech. In Leon A. Jakobovits & Murray S. Miron, editors, *Readings in the Psychology of Language.* Prentice Hall.

Mann, William C. (1982). Text Generation. *American Journal of Computational Linguistics,* 8(2).

Mann, William C. (1983). An Overview of the Penman Text Generation System. In *Proceedings of the Third National Conference on Artificial Intelligence.*

Mann, William C. & Sandra A. Thompson (1987). Rhetorical Structure Theory: description and construction of text structures. In Gerard Kempen, editor, *Natural Language Generation.* Nijhof. distributed by Kluwer.

McDonald, David D. (1983a). Description Directed Control: Its Implications for Natural Language Generation. *Computers and Mathematics with Applications*, 9(1). Reprinted in *Readings in Natural Language Processing*, edited by Barbara Grosz, Karen Sparck Jones, and Bonnie Webber, Morgan Kaufmann, Los Altos CA, 1986.

McDonald, David D. (1983b). Natural Language Generation as a Computational Problem. In Micheal Brady & Robert Berwick, editors, *Computational Theories of Discourse.* MIT Press.

McDonald, David D. (1987a). Natural-Language Generation. In Stuart C. Shapiro, editor, *Encylopedia of Artificial Intelligence, Volume 1.* Wiley.

McDonald, David D. (1987b). Natural Language Generation: Complexities and Techniques. In Sergei Nirenburg, editor, *Machine Translation.* Cambridge University Press.

McDonald, David D. (1988). Modularity in Natural Language Generation: Methodological Issues. In *Proceedings of the AAAI Workshop on Text Planning and Realization.*

McDonald, David D. (1991). On the Place of Words in the Generation Process. In Cécile L. Paris, William R. Swartout, & William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer.

McDonald, David D., Marie Vaughan, & James Pustejovsky (1987). Factors Contributing to Efficiency in Natural Language Generation. In Gerard Kempen, editor, *Natural Language Generation.* Nijhof. distributed by Kluwer.

McKeown, Kathleen R. (1985). Discourse Strategies for Generating Natural-Language Text. *Artificial Intelligence*, 27(1).

McKeown, Kathleen R. & Micheal Elhadad (1991). A Contrastive Evaluation of Functional Unification Grammar for Surface Language Generation: A Case Study in Choice of Connectives. In Cécile L. Paris, William R. Swartout, & William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer.

Meteer, Marie W. (1989). The Spokesman Natural Language Generation System. Technical Report 7090, BBN Systems and Technologies Corporation, Cambridge MA.

Meteer, Marie W. (1990). The "Generation Gap": The Problem of Expressibility in Text Planning. Technical Report 7347, BBN Systems and Technologies Corporation, Cambridge MA.

Meteer, Marie W. (1991). The Implications of Revisions for Natural Language Generation. In Cécile L. Paris, William R. Swartout, & William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer.

Miezitis, Mara (1988). Generating Lexical Options by Matching in a Knowledge Base. Technical Report CSRI-217, Computer Systems Research Institute, University of Toronto.

Miikkulainen, Risto (1990). A PDP Architecture for Processing Sentences with Relative Clauses. In *Proceedings 13th COLING.*

Miller, Perry L. & Glenn D. Rennels (1988). Prose Generation from Expert Systems. *AI Magazine*, 9(3).

Moore, Johanna D. & William R. Swartout (1989). A Reactive Approach to Explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.*

Motley, Micheal T., Bernard J. Baars, & Carl T. Camden (1983). Experimental Verbal Slip Studies: A Review and an Editing Model of Language Encoding. *Communication Monographs*, 50.

Nagao, Makoto (1984). A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In *Artificial and Human Intelligence.* Elsivier Science Publishers.

Nagao, Makoto (1987). Role of Structural Transformation in a Machine Translation System. In Sergei Nirenburg, editor, *Machine Translation.* Cambridge University Press.

Nagao, Makoto, Toyoaki Nishida, & Jun-ichi Tsujii (1984). Dealing with Incompleteness of Linguistic Knowledge in Language Translation — Transfer and Generation Stage of MU Machine Translation Project. In *Proceedings 10th COLING.*

Nagao, Makoto, Jun-ichi Tsujii, & Jun-ichi Nakamura (1986). Machine Translation from Japanese to English. *Proceedings of the IEEE*, 74(7).

Nirenburg, Sergei (1990). Treatment of Meaning in MT Systems. In *Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language.* Linguistics Research Center, University of Texas at Austin.

Nirenburg, Sergei, Rita McCardell, Eric Nyberg, Philip Werner, Scott Huffman, Edward Kenschaft, & Irene Nirenburg (1988). DIOGENES-88. Technical Report CMU-CMT-88-107, Carnegie Mellon University, Center for Machine Translation.

Nitta, Yoshihiko (1986). Idiosyncratic Gap: A Tough Problem to Structure-bound Machine Translation. In *Proceedings 10th COLING.*

Norvig, Peter (1987). *A Unified Theory of Inference for Text Understanding.* PhD thesis, University of California at Berkeley, Computer Science Division. also report UCB/CSD 87/339.

O'Connell, Daniel C. & Sabine Kowal (1983). Pausology. In Walter A. Sedelow, Jr. & Sally Yeates Sedelow, editors, *Computers in Language Research 2.* Mouton.

Patten, Terry & Graeme Ritchie (1987). A Formal Model of Systemic Grammar. In Gerard Kempen, editor, *Natural Language Generation.* Nijhof. distributed by Kluwer.

Pollack, Jordan (to appear). Recursive Distributed Representations. *Artificial Intelligence.*

Pustejovsky, James & Sergei Nirenburg (1987). Lexical Selection in the Process of Language Generation. In *Proceedings 25th Association for Computational Linguistics.*

Rager, John & George Berg (1990). A Connectionist Model of Motion and Government in Chomsky's Government-binding Theory. *Connection Science*, 2.

Reithinger, Norbert (1987). Generating Referring Expressions and Pointing Gestures. In Gerard Kempen, editor, *Natural Language Generation.* Nijhof. distributed by Kluwer.

Rosenberg, Sheldon (1977). Semantic Constraints on Sentence Production: an Experimental Approach. In Sheldon Rosenberg, editor, *Sentence Production.* Lawrence Erlbaum Associates.

Rumelhart, David E., James L. McClelland, & the PDP Research Group (1986). *Parallel Distributed Processing, Volume 1: Foundations.* MIT Press.

Saffran, E. M., M. F. Schwartz, & O. S. M. Marin (1980). Evidence from Aphasia: Isolating the Components of a Production Model. In Brian Butterworth, editor, *Language Production, Volume 1: Speech and Talk.* Academic Press.

Sato, Satoshi & Makoto Nagao (1990). Toward Memory-based Translation. In *Proceedings 13th COLING.*

Schreuder, Robert & Giovanni B. Flores d'Arcais (1989). Psycholinguistic Issues in the Lexical Representation of Meaning. In William Marslen-Wilson, editor, *Lexical Representation and Process.* MIT Press.

Seleskovitch, Danica (1968). *l'Intèrpret dans les Conférences Internationales.* Minard, Paris.

Shieber, Stuart (1986). *An Introduction to Unification-Based Approaches to Grammar.* CSLI, Stanford University.

Simmons, R. & J. Slocum (1972). Generating English Discourse from Semantic Networks. *Communications of the ACM*, 15(10).

Slocum, Jonathan (1986). MT — An American Perspective. *Proceedings of the IEEE*, 74(7).

Somers, Harold, Hideki Hirakawa, Seijii Miike, & Shinya Amano (1988). The Treatment of Complex English Nominalizations in Machine Translation. *Computers and Translation*, 3.

Somers, Harold, Jun-ichi Tsujii, & Danny Jones (1990). Machine Translation Without a Source Text. In *Proceedings 13th COLING.*

Sondheimer, Norman, Susanna Cumming, & Robert Albano (1990). How to Realize a Concept: Lexical Selection and the Conceptual Network in Text Generation. *Machine Translation*, 5.

Steiner, George (1975). *After Babel.* Oxford University Press.

Stemberger, Joseph Paul (1982). Syntactic Errors in Speech. *Journal of Psycholinguistic Research*, 11(4).

Stemberger, Joseph Paul (1983). Distant Context Effects in Language Production: A Reply to Motley et al. *Journal of Psycholinguistic Research*, 12(6).

Stemberger, Joseph Paul (1984). Structural Errors in Normal and Agrammatic Speech. *Cognitive Neuropsychology*, 1(4).

Stemberger, Joseph Paul (1985a). An Interactive Activation Model of Language Production. In Andrew W. Ellis, editor, *Progress in the Psychology of Language, Volume 1.* Lawrence Erlbaum Associates.

Stemberger, Joseph Paul (1985b). *The Lexicon in a Model of Language Production.* Garland Press.

Stolcke, Andreas (1989). Processing Unification-based Grammars in a Connectionist Framework. In *Program of the Eleventh Annual Conference of the Cognitive Science Society.* Lawrence Erlbaum Associates.

Sumita, Eiichiro, Hitoshi Iida, & Hideo Kohyama (1990). Translating with Examples: A New Approach to Machine Translation. In *Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language.* Linguistics Research Center, University of Texas at Austin.

Talmy, Leonard (1975). Syntax and Semantics of Motion. In John P. Kimball, editor, *Syntax and Semantics, Volume 4.* Academic Press.

Talmy, Leonard (1976). Communicative Aims and Means — A Synopsis. Technical Report 20, Working Papers on Language Universals, Stanford University.

Talmy, Leonard (1980). Lexicalization Patterns: Semantic Structure In Lexical Forms. In Timothy Shopen, editor, *Language Typology and Syntactic Description, Volume 3*. Cambridge University Press. also UC Berkeley Cognitive Science Program Report No. 30.

Thompson, Henry (1976). Towards a Model of Language Production: Linguistic and Computational Foundations. *Statistical Methods in Linguistics*. from an invited paper presented at COLING-76.

Tomabechi, Hideto (1987). Direct Memory Access Translation. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*.

Tomabechi, Hideto & Hiroaki Kitano (1989). Beyond PDP: the Frequency Modulation Neural Network Architecture. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*.

Tomita, Masaru & Jaime Carbonell (1987). The Universal Parser Architecture for Knowledge-Based Machine Translation. Technical Report CMU-CMT-87-101, Carnegie Mellon University, Center for Machine Translation.

Touretzky, David S. & Geoffrey E. Hinton (1985). Symbols Among the Neurons: Details of a Connectionist Inference Architecture. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*.

Tsujii, Jun-Ichi (1986). Future Directions of Machine Translation. In *Proceedings 10th COLING*.

Tsutsumi, Taijiro (1990). Wide-Range Restructuring of Intermediate Representations in Machine Translation. *Computational Linguistics*, 16(2).

Uchida, Hiroshi (1988). ATLAS II: A Machine Translation System Using Conceptual Structures as an Interlingua. In *Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*. Carnegie Mellon University, Center for Machine Translation.

van Gelder, Tim (1990). Compositionality: A Connectionist Variation on a Classical Theme. *Cognitive Science*, 14(3).

van Oosten, Jeanne (1985). *The Nature of Subjects, Topics, and Agents: A Cognitive Explanation*. Indiana University Linguistics Club, Bloomington, Indiana. revised version of a Ph.D. Thesis, University of California at Berkeley, Linguistics Department.

Vygotsky, Lev S. (1962). *Thought and Language*. MIT Press. originally 1934.

Ward, Nigel (1989). A Model for Natural Machine Translation. In *38th National Conference*, Tokyo. Information Processing Society of Japan.

Weaver, Warren (1955). Translation. In W. N. Locke & A. D. Booth, editors, *Machine Translation of Languages*. MIT Press.

Wilensky, Robert (1990). Extending the Lexicon by Exploiting Subregularities. In *Proceedings 13th COLING*.

Wilensky, Robert & Yigal Arens (1980). PHRAN — A Knowledge-Based Natural Language Understander. In *Proceedings 18th Association for Computational Linguistics*.

Wilensky, Robert, David N. Chin, Marc Luria, James Martin, James Mayfield, & Dekai Wu (1989). The Berkeley UNIX Consultant Project. Technical Report UCB/CSD 89/520, University of California at Berkeley, Computer Science Division.

Wilks, Yorick (1972). An Artificial Intelligence Approach to Machine Translation. In Roger Schank & Kenneth Colby, editors, *Computer Models of Thought and Language*. W. H. Freeman.

Wittgenstein, Ludwig (1963). *Philosophical Investigations*. Basil Blackwell, Oxford. originally 1945.

Wundt, Wilhelm (1900). *Die Sprache*. Engelmann, Leipzig. quoted in Arthur L. Blumenthal (ed.) *Language and Psychology — Historical Aspects of Psycholinguistics*, 1970, Wiley.