A Connectionist Treatment of Grammar for Generation: Relying on Emergents

Nigel Ward
Computer Science Division
University of California at Berkeley

Abstract

Parallel treatment of syntactic considerations in generation promises quality and speed. Parallelism should be used not only for simultaneous processing of several sub-parts of the output, but even within single parts. If both types of parallelism are used with incremental generation it becomes unnecessary to build up and manipulate representations of sentence structure — the syntactic form of the output can be emergent.

FIG is a structured connectionist generator built in this way. Constructions and their constituents are represented in the same network which encodes world knowledge and lexical knowledge. Grammatical output results from synergy among many constructions simultaneously active at run-time. FIG incorporates new ways of handling constituency, word order and optional constituents; and simple ways to avoid the problems of instantiation and binding. Syntactic knowledge is expressed in a simple, readable form; this representation straightforwardly defines parts of the network.

1 Introduction

Generation research has not yet fully identified the advantages offered by parallelism nor the techniques necessary to take advantage of it. This is especially true for the syntactic aspects of generation.

This paper presents a way to exploit parallelism for syntax in generation. The key points are: Syntactic constructions are encoded in the same knowledge network as words and concepts. Many constructions are active in parallel; there is synergy, and sometimes competition. The syntactic form of the output emerges from interactions among constructions at run-time — explicit syntactic choice and building up of representations of syntactic structure are unnecessary.

To see that this approach works for syntactically non-trivial examples, consider that FIG's outputs include: *"once upon a time there lived an old man and an old woman," "one day the old man went into the hills to gather wood," "a big peach bobbed down towards an old woman from upstream," "an old woman gave a peach to an old man," "John broke a dish," "John made the cake vanish,"* and *"Mary was killed;"* and when producing Japanese: *"mukashi mukashi aru tokoro ni ojiisan to obaasan ga sunde imashita," "aru hi ojiisan wa yama e shibakari ni ikimashita," "kawakami kara ookii momo ga donburiko donburako to obaasan e nagarete kimashita," "ojiisan wa meeri ni momo o agemashita,"* and *"meeri o koroshimashita."*

Section 2 discusses parallelism in syntax and presents the basic proposal. Section 3 presents a framework for connectionist generation, and Section 4 elaborates the proposal in this framework. Sections 5 through 8 discuss an implementation of these ideas: Section 5 presents a representation for grammatical knowledge, Section 6 explains how the proposal accounts for specific syntactic phenomena, Section 7 presents an example of the generator in action, and Section 8 discusses general implementation issues. Section 9 summarizes.

2 Parallel Syntax

This section discusses two types of parallelism for syntax, proposes that a generator should have both of them, and sketches out the advantages of such an approach.

Natural language generation research traditionally assumed that syntactic choices are made in a fixed (and generally top-down) order. Yet, for incremental generation at least, it is clear that a fixed order of decisions is not appropriate. This realization has led to generators which work on several parts of the input in parallel, simultaneously building several sub-trees. Recent work in this area includes (De Smedt 1990) and (Finkler & Neumann 1989). I will refer to this type of parallelism as 'part-wise' parallelism.

A second kind of parallelism involves using several constructions to generate even one part of the output. As far as I know, this 'within-part' parallelism has not been proposed in the generation literature. It has proven useful in linguistics. In Fillmore's Construction Grammar the syntactic

[1]Thanks to Daniel Jurafsky, Robert Wilensky, Dekai Wu, and Terry Regier. This research was sponsored by the Defense Advanced Research Projects Agency (DoD), monitored by the Space and Naval Warfare Systems Command under N00039-88-C-0292, and the Office of Naval Research under contract N00014-89-J-3205. An early version of this paper appears in the *Proceedings of the 12th Cognitive Science Conference*, Erlbaum, 1990.

structure of sentences is accounted for in terms of 'superimposition' of constructions (Fillmore 1989b). It has also been used in psycholinguistics, where analysis of speech errors suggests that even normal speech is the result of competing 'plans' (Baars 1980). More specifically, (Stemberger 1985) suggested that human speakers can be modeled as having many 'phrase structure units' being 'partially activated' simultaneously. That is, many syntactic alternatives for expressing some piece of meaning are considered in parallel.

I propose that a generator should exploit both part-wise and within-part parallelism.

Parallel generation is a good idea for several reasons. 1. It has been observed that part-wise parallelism is a good way to improve the speed of response, especially for incremental generation. 2. Part-wise parallelism is also useful for handling dependencies. It is not always the case that one part can be processed without consideration of 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. This can lead to better quality. 3. Given the possibility of constraints among the various syntactic choices involved in building an utterance, there is the possibility that a 'first choice' will not work out when the larger context is considered. This suggests within-part parallelism, so that a generator has available alternative ways to realize some information. Given this it can find a set of choices satisfies all the dependencies, resulting in consistent and natural utterance. 4. If a generator is indeed to consider all the possible dependencies among choices, then parallelism becomes necessary to cope with the amount of computation necessary. 5. Parallelism is the natural way to generate if the input is very complex (Ward 1989a).

3 The FIG Approach to Generation

Reduced to bare essentials, a generator's task is to get from concepts (what the speaker wants to express) to words (what he can say). On this view, the key problem in generation is computing the relevance (pertinence) of a particular word, given the concepts to express. Syntactic and other knowledge mediates this computation of relevance.

Accordingly FIG is based on word choice — every other consideration is analyzed in terms of how it affects word choice.

FIG is based on a large semantic network. Words are nodes in the network, the activation they receive represents evidence for their relevance. The basic FIG algorithm is:

1. each node of the input 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

Thus FIG is an incremental generator. Its network must be designed so that, when it settles, the node which is most highly activated corresponds to the best next word. This paper discusses only the network structures which encode syntactic knowledge.

Elsewhere I argue that FIG points the way to accurate and flexible word choice (Ward 1988), producing natural-sounding output for machine translation (Ward 1989c), and modeling the key aspects of the human language production process (Ward 1989a).

4 Connectionist Syntax: Overview

In FIG constructions and constituents also are represented as nodes in the knowledge network. Their activation levels represent their current relevance. They interact with other nodes by means of activation flow. Any number of constructions can be simultaneously active. This handles part-wise parallelism, competition, and superimposition.

Syntactic considerations manifest themselves only through their effects on the activation levels of words (directly or indirectly). An utterance is simply the result of successive word choices. FIG does produce grammatical sentences, most of the time, but their 'syntactic structure' is emergent, a side-effect of expressing the meaning. Thus we can say that the syntactic form of utterances is emergent in FIG[2]. This point will be illustrated repeatedly in Section 6.

Mechanisms developed by linguists (and often adopted by generation researchers), such as unification, are not directed to the task of generation (or parsing) so much as to the goal of explaining sentence structure. Accounting for the structure of sentences may be a worthwhile goal for linguistics, but building syntactic structures is not necessary for language generation, as subsequent sections will show.

The most common metaphor for generation is that of making choices among alternatives. For example, a generator may choose among words for a concept, among ways to syntactically realize a constituent, and among concepts to bind to a slot. Given this metaphor, organizing choices becomes the key problem in generator design. Attempts to build parallel generators while retaining the notion of explicit choice run up against problems of sequencing the choices or of doing bookkeeping so that the order of choices can vary. This appears to be difficult, judging by the general paucity of published outputs in descriptions of parallel generators. On the other hand, relying on emergents means

[2]Post hoc examination of FIG output might make one think, for example, 'this exhibits the choice of the existential-there construction.' In FIG there is indeed an inhibit link between the nodes **ex-there** and **subj-pred**, and so when generating the network tends to reach a state where only one of these is highly activated. The most highly activated construction can have a strong effect on word choices, which is why the appearance of syntactic choice arises.

```
(defp noun-phr
  (constituents (np-1  obl  article  ((article 1.2)))
                (np-2  opt  adjective((adjective .28)))
                (np-3  obl  noun     ((cnoun .47))) ))
```

Figure 1: Representation of the English Noun-Phrase Construction

```
(defp go-p
  (constituents (gp-1  obl go-w   ((go-w .2)))
                (gp-2  opt epart ((vparticle .6) (directionr .2)))
                (gp-3  opt noun  ((prep-phr .6) (destinationr .2)))
                (gp-4  opt verb  ((purpose-clause .7) (purposer .2))) ))
```

Figure 2: Representation of the Valence of *"Go"*

```
(defp ex-there
  (inhibit subj-pred passive)
  (constituents (et-1 obl therew ((therew .5)))
                (et-2 obl verb   ((verb .5)))
                (et-3 obl noun   ((noun .3))) ))
```

Figure 3: Representation of the Existential *"There"* Construction

there are no explicit choices to worry about, and thus there are no problems of ordering or bookkeeping at all(Ward 1989b).

In FIG all types of knowledge represented are uniformly in the network, and interact freely at run time. FIG not only allows this kind of interaction among various considerations when generating, it relies on it. It relies on synergy among constructions in the same way that Construction Grammar does. It relies on synergy between semantic and syntactic considerations, as seen below in Section 6.7. It also enables interaction among lexical choices and syntactic considerations.

5 Knowledge of Syntax

This section presents FIG's representation of knowledge, first presenting it in a declarative form then showing how that representation maps into network structures.

Starting with this section I will be largely describing FIG-as-implemented, as of May 1990. This is for the sake of concreteness. The theory, however, is intended to apply to parallel generators in general. Moreover, the syntactic knowledge presented in this section is purely illustrative. I do not claim that these represent the facts of English, nor the best way to describe them in a grammar. In particular, many generalizations are not captured. The examples are intended simply to illustrate the representational tools and computational mechanisms available in FIG. Many details are left unexplained for lack of space.

Figure 1 shows FIG's definition of **noun-phr**, representing the English noun-phrase construction. This construction has three constituents: **np-1**, **np-2**, and **np-3**. **np-1** and **np-3** are obligatory, **np-2** is optional. Glossing over the details for the moment, the list at the end of each constituent's definition specifies how to realize the constituent. For example, **np-1**, **np-2**, and **np-3**, should be realized as an **article**, **adjective**, and **noun**, respectively.

Figure 2 shows the construction for the case frame of the word *"go."* First comes **go-w**, for the word *"go,"* which is obligatory. Next come (optionally): a verb-particle representing direction (as in *"go* away" or *"go* back *home"* or *"go* down *to the lake"*), a prepositional phrase to express the destination, and a purpose clause.

Figure 3 shows the representation of the existential *"there"* construction, as in *"there was a poor cobbler."* The 'inhibit' field indicates that this construction is incompatible with the **passive** construction and also with **subj-pred**, the construction responsible for the basic SVO ordering of English.

Figure 4 shows knowledge about when and where constructions are relevant. Briefly, constructions are associated with words, with concepts, and with other constructions.

Constructions are associated with the meanings they can express. For example, **ex-there** is listed under the concept **introductory**, representing that this construction is appropriate for introducing some character into the story, and **purpose-clause** is listed as a way to express the **purposer** relation.

Constructions are associated with words. For example **go-p** is the 'valence' (case frame) of **go-w** and **noun-phr** is the 'maximal' of **cnoun**.

Constructions are also associated with other constructions. For example, the fourth constituent of **go-p** subcategorizes for **purpose-clause** (Figure 2); and there are negative associations among incompatible constructions, for example the 'inhibit' link between **ex-there** and **subj-pred** (Figure 3).

Figure 5 shows a fragment of FIG's network, where the numbers on the links are their weights. This is partially

```
(defw peachw
   (smallcat cnoun)(expresses momoc) (grapheme "peach") (english (consnt-initial .5)) )

(defs cnoun  (bigcat noun .4)  (maximals (noun-phr .4)))     ; common-noun

(defw go-w  (cat verb)  (expresses ikuc)  (valence (go-p .2))
      (grapheme (inf "go") (past "went") (pastp "gone") (presp "going")) )

(defc introductoryc  (properties persistent)   (english (ex-there .2) ))

(defr purposer  (english (to2w .4) (purpose-clause .1))  (japanese (ni-w .6)))
```

Figure 4: Some Knowledge Related to Constructions

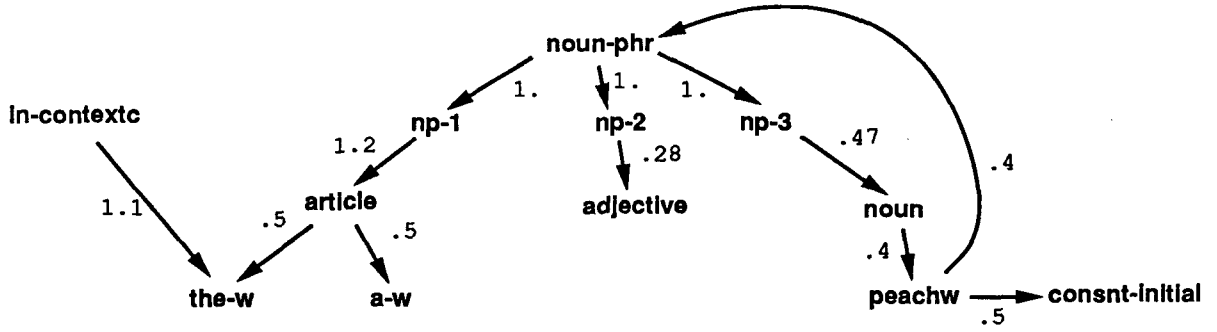

Figure 5: A Fragment of the Network

specified by the knowledge shown in the previous figures. The mapping from s-expressions to network structures is not quite trivial. For example, the link from **noun** to **peachw** comes from the statements that **peachw** has 'subcat' **cnoun** and that **cnoun** has 'bigcat' **noun**. Similarly, the link from **peachw** to **noun-phr** is inherited by **peachw** from the 'maximals' information on **cnoun**.

6 Various Syntactic Phenomena

6.1 Constituency

The links described above suffice to handle constituency. Consider for example the fact that common nouns must be preceded by articles in FIG's subset of English. Suppose that **peachw** is activated, perhaps because a **peachc** concept is in the input. Activation flows from **peachw** via **noun-phr**, **np-1**, and **article** to **a-w** and **the-w**.

In this way the relevance of a noun increases the relevance rating 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 these 'decisions' emerge as activation levels settle.

Any node can be mentioned by a constituent, thus constructions can specify: which semantic elements to include (metonymies), what order to mention things in, what function words to choose, and what inflections to use.

6.2 Subcategorization

Consider the problem of specifying where a given concept should appear and what syntactic form it should take. In FIG this is handled by simultaneously activating a concept node and a syntactic construction or category node. For example, the third constituent of **go-p** specifies that 'the direction of the going' be expressed as a 'verbal particle.' 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**. Thanks to this sort of activation flow FIG tends to select and emit an appropriate word in an appropriate form (Ward 1988). Government, for example, the way that some verbs govern case markers, is handled in the same way.

6.3 Word Order

In an incremental connectionist generator, at each time the activation level of a word must represent its *current* relevance. In particular, words which are currently syntactically appropriate must be strongly activated. In FIG the representation of the current syntactic state is distributed across the constructions. 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.' Thus word order is emergent.

In FIG the current syntactic state is encoded in constructions' activation levels and 'cursors.' The cursor of a construction points to the currently appropriate constituent and ensures that it is relatively highly activated. To be specific, the cursor gives the location of a 'mask' specifying the weights of the links from the construction to constituents. The mask specifies a weight of 1.0 for the constituent under the cursor, and for subsequent constituents a weight proportional to their closeness to the cursor. (Subsequent constituents must receive some activation so that there is partwise parallelism.) (For unordered constructions the weights on all construction-constituent links are the same.)

For example, when the cursor of **noun-phr** points to **np-1**, articles receive a large proportion of the activation of **noun-phr**. Thus, an article is likely to be the most highly activated word and therefore selected and emitted. After an article is emitted the cursor is advanced to **np-2**, and so on. Advancing cursors is described in Section 6.5.

In accordance with the intuition that a word is not truly appropriate unless it is both syntactically and semantically appropriate, the activation level for words is given by the product (not the sum) of incoming syntactic and semantic activation, where 'syntactic activation' is activation received from constituents and syntactic categories. The problem with simply summing is that it results in the the network often being in a state where many word-nodes have nearly equal activation, which makes the behavior is oversensitive to minor changes in link weights.

6.4 Optional Constituents

When building a noun-phrase a generator should emit an adjective if semantically appropriate, otherwise it should ignore that option and emit a noun next. FIG does this without additional mechanism.

To see this, suppose *"the"* has been emitted and the cursor of **noun-phr** is on its second constituent, **np-2**. As a result adjectives get activation, via **np-2**, and so to a lesser extent do nouns via **np-3**. There are two cases: If the input includes a concept linked (indirectly perhaps) to some adjective, that adjective will receive activation from it. 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. 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.

Most generators use some syntax-driven procedure to inspect semantics and decide explicitly whether or not to realize an optional constituent. 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.

6.5 Updating Constructions

Recall that FIG, after selecting and emitting a word, updates activation levels to represent the new state. There are are several aspects to this.

The cursors of constructions must advance as constituents are completed. The update mechanism can 'skip over' 'opt constituents, since, for example, if there are no adjectives, the cursor of **noun-phr** should not remain stuck forever at the second constituent. More than one construction may be updated after a word is output, for example, emitting a noun may cause updates to both the **prep-phr** construction and the **noun-phr** construction.

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 and sets temporary lower bounds on their activation levels. Thus, even though FIG does not make any syntactic plans, it tends to form a grammatical continuation of whatever it has already output. After the last constituent of a construction has been completed, the cursor is reset and the lower bound is removed.

Why is a separate update mechanism necessary? 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. Therefore constructions need this kind of feedback on what words have been output.

6.6 No Instantiation or Binding

It is not obvious that notions of instantiation, binding, embedding, or recursion are essential for the description of natural language. Nor are mechanisms for these things essential for the generation task, I conjecture. This subsection considers a problem which is usually handled with instantiation and shows how it can be handled more simply without.

Consider the problem of generating utterances with multiple 'copies,' for example, several noun phrases, or several uses of *"a"*. Note that FIG as described so far would have problems with this. For example since all words of category **cnoun** have links to **noun-phr**, that node might receive more activation than appropriate, in cases when several nouns are active. This could result in over-activation of articles, and thus premature output of *"the,"* for example.

In fact FIG uses a special rule for activation received across inherited links: the maximum (not the sum) of these amounts is used. For example, this rule applies to the 'maximal' links from nouns to **noun-phr**, thus **noun-phr** effectively 'ignores' all but the most highly activated noun. (This was not shown in Figure 5.)

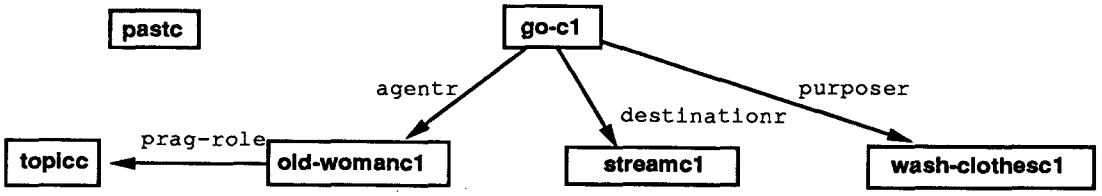

Figure 6: An Input to FIG

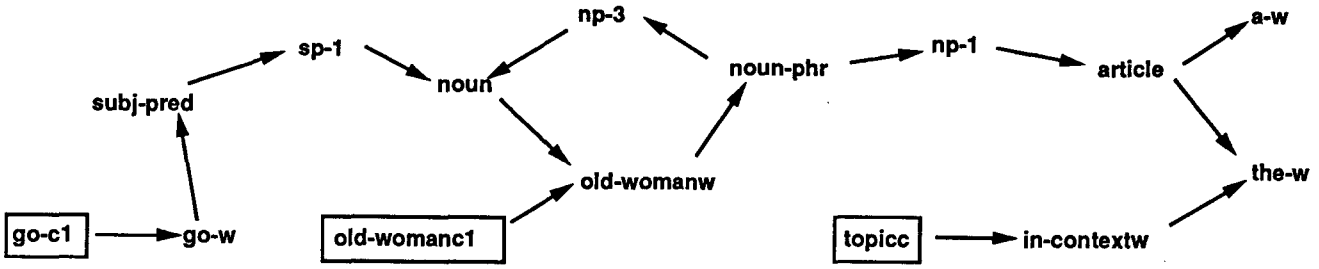

Figure 7: Selected Paths of Activation Flow Just Before Output of *"the"*

An earlier version of FIG handled this by actually making copies. For example, it would make a copy of **noun-phr** 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.

6.7 Low-level Coherence

Words must stand in the correct relations to their neighbors. For example, a generator must not produce *"the* big *man went to the mountain"* when the input calls for *"the man 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.

Most generators handle this easily with structure-mapping or pointer following. For example, a syntax-directed generator may, whenever building a noun phrase, traverse the 'modified-by' pointer to find the item to turn into an adjective. FIG, however, eschews structure manipulation and pointer following. Like all connectionist approaches, therefore, it is potentially subject to problems with crosstalk.

The way to avoid this is to ensure that related concepts become highly activated together. In the example, **bigc** should become activated together with **mountainc**, not together with **old-manc**. Using a more elaborate terminology, this means that there should be some kind of 'focus of attention' (Chafe 1980), which successively 'lights up' groups of related nodes.

This condition is met in FIG, thanks to the links among the nodes of the input. For example, if **mountainc1** is linked by a **sizer** link to **bigc1**, then **bigc1** will tend to become highly activated whenever **mountainc1** is. Thus, when **old-manc1** is the most highly activated concept-node, **bigc1** will only receive energy from it indirectly (via an inverse-**agentr** link, a **locationr** link, and a **sizer** link) and thus will not be activated sufficiently to interfere early in the sentence.

7 Example

This section describes how FIG produces *"the old woman went to a stream to wash clothes."* For this example the input is the set of nodes **go-c1**, **old-womanc1**, **wash-clothesc1**, **streamc1**, and **pastc**, linked together as shown in Figure 6. (The names of the concepts have been 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 has 11 units 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 7. Figure 8 shows the activation levels of selected nodes.

After *"the"* is emitted the update mechanism activates **noun-phr** and advances its cursor to **np-2**. The most highly activated word becomes **old-womanw**, largely due to activation from **np-3**.

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

---PATTERNS---	----WORDS-----	---CONCEPTS---
15.6 SUBJ-PRED	29.7 THE-W	19.7 OLD-WOMANC1
SP-1 sp-2	21.0 A-W	15.0 IKUC1
7.6 CAUSATIVEP	18.5 OLD-WOMANW	14.0 KAWAC1
CP-1 cp-2 cp-3	13.3 STREAMW	13.2 SENTAKUC1
6.6 NOUN-PHR	10.7 RIVERW	11.0 PASTC
NP-1 np-2 np-3	10.0 GO-W	8.3 VOWEL-INITIAL
1.8 GO-P	7.5 WASH-CLOTHESW	6.1 CONSNT-INITIAL
GP-1 gp-2 gp-3 gp-4	3.9 TO1W	5.8 TOPICC
1.4 PURPOSE-CLAUSE	3.2 MAKEW	----OTHER-----
PC-1 pc-2 pc-3	2.9 TOWARDSW	13.4 CAUSER
0.2 PREP-PHR	2.9 INTOW	10.4 AGENTR
PP-1 pp-2	2.5 TO2W	6.9 ARTICLE
---	0.4 WITHW	4.5 NOUN

Figure 8: Activation Levels of Selected Nodes Just Before Output of *"the"*

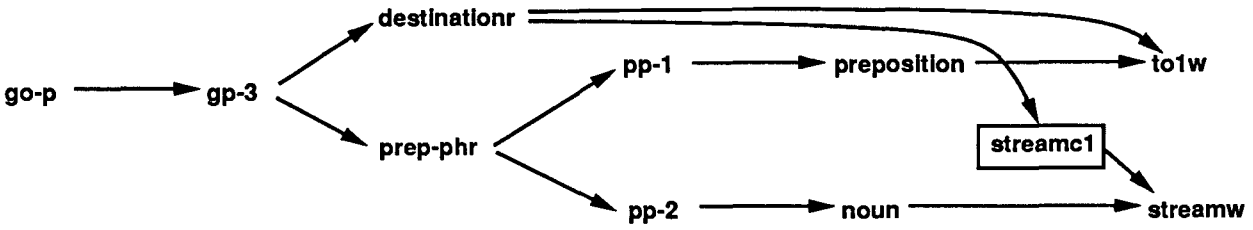

Figure 9: Selected Paths of Activation Flow Just Before Output of *"to"*

go-w is selected. Because **pastc** has more activation than **presentc**, **infinitivec** and so on, **go-w** is inflected and emitted as *"went"* (the inflection mechanism is not described in this paper). **go-p**'s cursor advances to its second constituent, thus it activates directional particles, although there is no semantic input to any such word in this case. **to1w** becomes the most highly activated word, primarily for the reasons shown in Figure 9.

After *"to"* is emitted, the cursor of **prep-phr** is advanced. The key path of activation flow is now from the second constituent of **prep-phr** to **noun** to **streamw** to **noun-phr** 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 **noun-phr** advances and *"stream"* is emitted. After this the cursor of **go-p** advances to **gp-4**. From this constituent activation flows to **purpose-clause**, and in due course *"to"* and *"wash clothes"* are emitted.

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

8 About the Implementation

I have used a connectionist model because it is a good way to explore interactivity, parallelism, emergents, not because of fondness for connectionism-for-its-own-sake.

Thus I have not attempted to develop a distributed connectionist model. Distributed models do have various advantages, such as elegant handling of generalizations and the potential for learning. Yet the current state of PDP technology does not seem up to building an interactive model of a complex task like language generation. I therefore developed FIG as a structured (localist) connectionist system.

I have also not attempted to make FIG a 'pure' connectionist model. For example, updating constructions is currently done by a special process that goes in and changes activation levels and moves the cursor. (This process uses the third elements in the constituent descriptions of Figures 1–3, not previously discussed.) FIG could be made more 'pure' by doing this connectionistically, perhaps by adding new nodes with special properties. But this change would not improve FIG's performance, since there seems no need for the update process to interact with the other processes.

A connectionist model of computation allows parallelism and emergents, but it certainly does not require them. Indeed, other generators built using structured connectionism (Kalita & Shastri 1987; Gasser 1988; Kitano 1989; Stolcke 1989) do not appear to exploit parallelism much, nor do they exhibit emergent properties. For example, Gasser's CHIE relies heavily on winner-take-all subnetworks, which cuts down on the amount of effective parallelism. Also, far from exploiting emergents, CHIE uses 'neuron firings' to model syntactic choices; these happen sequentially and the

exact order and timing of firings seems crucial.

Currently FIG has about 350 nodes and 1000 links. Before each word choice, activation flows until the network settles down, with cutoff after 9 cycles. This takes about .2 seconds per word on average, simulating parallel activation flow on a Symbolics 3670 (1.6 seconds on a Sun 3/140).

The correct operation of FIG depends on having correct link weights. I have no theory of weights, indeed finding appropriate ones is still largely an empirical process. However there are regularities, for example, all 'inhibit' links have weight .7, almost all links from syntactic categories to their members have weight .5, and so on. Many of the weights have a rationale: for example, the link from **np-1** to articles has a relatively high weight because articles get very little activation from other sources. No single weight is meaningful; the way it functions in context is. For example, the exact weight of the link from the first constituent of **subj-pred** to **noun** is not crucial, as long as the product of it and the weight on the **agentr** relation is appropriate.

FIG's knowledge is, of course, very limited. Adding new concepts, words or constructions is generally straightforward; they can be encoded by analogy to similar nodes, and usually the same link weights suffice. Occasionally new nodes and links interact with other knowledge in the system in unforeseen ways, causing other nodes to get too much or too little activation. In these cases it is necessary to debug the network. Sometimes trial-and-error experimentation is required, but often the acceptable range of weights can be determined by examination. This is a kind of backpropagation by hand; it could doubtless be automated.

9 Summary

I have proposed a new way to handle syntax for generation. The proposal also relies heavily on parallelism: partwise parallelism, competition, and cooperation. Also, syntactic considerations are used in parallel with lexical and world knowledge and there is pervasive interaction among them. This promises improved output quality without sacrificing speed, on parallel hardware. The proposal also relies heavily on emergents — it does not make syntactic choices nor build up representations of syntactic structure. The network representations of linguistic knowledge affect word choice and order directly.

This work is not traditional linguistics, artificial intelligence, or connectionism, but uses techniques from all three fields. I hope this will stimulate further work in empirical computational linguistics, modeling human language production, and building useful parallel generation systems.

References

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.

Chafe, Wallace L. (1980). The Deployment of Consciousness in the Production of a Narrative. In Wallace L. Chafe, editor, *The Pear Stories.* Ablex.

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.

Fillmore, Charles (1989a). The Mechanisms of "Construction Grammar". In *Proceedings of the Berkeley Linguistic Society*, volume 15.

Fillmore, Charles (1989b). On Grammatical Constructions. course notes, UC Berkeley Linguistics Department.

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.* Detroit.

Gasser, Micheal (1988). A Connectionist Model of Sentence Generation in a First and Second Language. Technical Report UCLA-AI-88-13, Los Angeles.

Kalita, Jugal & Lokendra Shastri (1987). Generation of Simple Sentences in English Using the Connectionist Model of Computation. In *9th Cognitive Science Conference.* Lawrence Erlbaum Associates.

Kitano, Hiroaki (1989). A Massively Parallel Model of Natural Language Generation for Interpreting Telephony: Almost Concurrent Processing of Parsing and Generation. In *Proceedings of the Second European Workshop on Natural Language Generation.*

Stemberger, J. P. (1985). An Interactive Activation Model of Language Production. In Andrew W. Ellis, editor, *Progress in the Psychology of Language, Volume 1.* Lawrence Erlbaum Associates.

Stolcke, Andreas (1989). Processing Unification-based Grammars in a Connectionist Framework. In *11th Cognitive Science Conference.* Lawrence Erlbaum Associates.

Ward, Nigel (1988). Issues in Word Choice. In *Proceedings 12th COLING.* Budapest.

Ward, Nigel (1989a). Capturing Intuitions about Human Language Production. In *Proceedings, Cognitive Science Conference.* Lawrence Erlbaum Associates. Ann Arbor.

Ward, Nigel (1989b). On the Ordering of Decisions in Machine Translation. In *Proceedings of the Third Annual Conference of the Japanese Society for Artificial Intelligence*, Tokyo.

Ward, Nigel (1989c). Towards Natural Machine Translation. In *Proceedings of the EIC Workshop on Artificial Intelligence*, Tokyo. Institute of Electronics, Information, and Communication Engineers. Published as Technical Research Report AI89-30.