# Using Tree Adjoining Grammars in the Systemic Framework*

**Kathleen F. McCoy, K. Vijay-Shanker, Gijoo Yang**
University of Delaware
Newark, DE 19716

## Abstract

In this paper we investigate the incorporation of Tree Adjoining Grammars (TAG) into the systemic framework. We show that while systemic grammars have many desirable characteristics as a generation paradigm, they appear to have problems in generating certain kinds of sentences (e.g., those containing discontinuity or long-distance dependencies). We argue that these problems can be overcome with an appropriate choice of structural units of realization.

We show that TAG provides appropriate units of structural realization because they localize all dependencies and allow the realization of two independent subpieces to be interspersed with each other. We go on to show how TAG can be incorporated without affecting the basic tenants of systemic grammar. Finally, we indicate how the incorporation of TAG yields several benefits to the systemic framework.

## Introduction

As pointed out by many researchers (e.g., [Davey 1978; Mann 1983; Matthiessen & Kasper 1985; Patten 1988; Bateman & Paris 1989]), systemic linguistics offers many advantages to a sentence generation component of a text generation system. Perhaps the strongest asset of systemics is its view of the generation process as a goal directed enterprise. Its emphasis is on function rather than form [Halliday 1985; Fawcett 1980; Hudson 1971], where the functional distinctions that are required in the grammar manifest themselves in the eventual output form.

While systemic linguists have remained agnostic with respect to certain processing decisions, a computer implementation of a systemic grammar requires that explicit decisions be made concerning realization operators and the structures available for manipulation at each point in the processing. The explicit decisions that were made in previous implementations of systemic grammar (e.g., [Mann 1983; Mann & Matthiessen 1985; Matthiessen & Kasper 1985]) have proven to be problematic in some respects. In particular, the current implementations have difficulty in generating certain sentences which exhibit discontinuities or long distance dependencies. To date, these can only be handled in a limited fashion, and the solutions provided are not very satisfying.

We argue that Tree Adjoining Grammar (TAG) provides a structural unit that is precisely appropriate for the implementation of a systemic grammar for the generation task. Moreover, we believe our use of TAG for this purpose is completely consistent with the systemic paradigm and helps to overcome the above difficulties.

In this paper we first introduce the notion of a systemic grammar and the processing paradigm it espouses. We indicate problems with current implementations of this paradigm. Next, we introduce the notion of lexicalized Tree Adjoining Grammars, emphasizing their structural domains of locality, and justify that the basic structures of TAG are appropriate structures to be used in an implementation of a systemic grammar. Following this we indicate how a tree adjoining grammar can be used as the basis for an implementation of systemic grammar indicating the differences between the approach of current implementations of systemic grammar and that which would result from the incorporation of TAG. Finally, we indicate potential gains resulting from the incorporation of TAGs and the scope of current work.

## Generating in Systemic Paradigm: Problems?

Systemic linguistics deals with the meaning and function of an utterance. It is a semantics driven approach rather than a syntax driven approach. In systemics, form follows function. The grammar itself is factored into three metafunctional domains (each of which affect the final text): ideational (concerning the characteristics of the conceptual situation to be presented), interpersonal (concerning the interaction between speaker and hearer) and textual (concerning the coherence of

*This work is supported in part by Grant #H133E80015 from the National Institute on Disability and Rehabilitation Research. Support was also provided by the Nemours Foundation.

the text as a whole).

A systemic functional grammar consists of networks of grammatical *choice* alternatives, where individual networks are concerned with one of the metafunctional domains. In generation, these networks are traversed and fragments of grammatical structure are built up and manipulated using a set of *realization operators* which are associated with the various choice alternatives. The correct choice is made by consulting the information concerning the planned utterances. As the choices are made, the associated realization statements in the network are evaluated in order to realize the final structure.

For instance, figure 1 contains a small fragment of a systemic grammar for clauses. The network indicates that a clause may either be simple or complex. If it is simple and full, then the grammatical function process is inserted (indicated by +process). The realization operation "^ subject process" indicates that the subject should be ordered before the process in the final realization.

Systemics deals with communicative function and its eventual surface manifestation at many levels. The basic processing starts with a semantically meaningful piece of representation which is decomposed into its component pieces via network traversal. The component pieces may then be further specified by re-entering the network. Given this rank-based decomposition (or stepwise decomposition), it is not unreasonable to assume that 1) decisions at a higher rank are made prior to decisions at a lower rank – that is, the decomposition of a particular semantic unit may not be influenced by the eventual decomposition of its component pieces, and 2) the structural realizations of the component pieces of a decomposed unit must be handled independently. These criteria, which we call the independence criterion, are implicitly followed by current computer implementations of systemic grammar.

Implementing a systemic grammar on computer has forced researchers to be very explicit about certain representational issues. Such explicitness (coupled with an implicit following of the independence criterion), has enabled the uncovering of certain constructions which appear to be problematic for the systemic framework. For instance, Matthiessen points out that: "There are various structural relationships (e.g., discontinuity...) that do not pose a problem for the informal box diagram representation [used by systemic linguists] but prove to be a problem for explicit realization statements [necessary for computer implementations]."[Matthiessen & Kasper 1985, p. 6]. We believe that the same applies to (so called) long distance dependencies.

It can be argued that the problems uncovered by Matthiessen are not due to explicit realization statements, but rather result from using explicit realization statements coupled with the implicit assumption that grammatical functions are realized as atomic strings. We further argue that if the independence criterion is to be followed, the choice of realization operators, the scope of the realization operators, and the choice of appropriate units of realization must be considered together.

Consider, for example, the problems which arise in generating a sentence containing a long distance dependency when atomic strings are taken as the structural unit of realization. Consider the sentence: "It was Mary that John thought Bill liked". A natural decomposition would result in the semantic subpieces which correspond to "john thought" and the dependent clause "bill liked mary". The extraction of "mary" will be done subsequent to this decomposition; this extraction should influence the realization of the structural unit for the dependent clause (and should not affect the functional decomposition). But notice, if we assume the structural units of realization are atomic strings, we can get "John thought it was Mary that Bill liked" but not our desired utterance "It was Mary that John thought Bill liked" because to do so would necessitate inserting one atomic string ("John thought") within another ("It was Mary that Bill liked").

Two possible ways to get around this problem are: (1) to say that the intended sentence is made up of several independent but smaller functions (e.g., realized as "john thought", "bill", "liked", "mary"). But this solution goes against the step-wise decomposition into semantically meaningful units. Moreover, this method is not sufficient because we can always consider a sentence that requires one more level of embedding which would necessitate a revision of units. (2) not to break up the sentence into the two functional constituents mentioned, but use some other decomposition. But this is not a correct solution because then one functional decision at a lower level (extraction of "mary") influences another which logically precedes the first (decomposition in the clause complex network).

In order to follow the independence criterion (i.e., realization of different independent functions do not influence each other), the structural units being built as the realization of a grammatical function must be capable of localizing all structural dependencies because they must embody all constraints specified with that function. In addition, the chosen structural units must be composable in such a way as to allow the surface string of one unit to be interspersed with the surface string for another (as was required in our example). If this is the case, then it makes sense that these structural units should be taken as the bounding domain for the realization operators. The structures used by TAGs have precisely these qualities and are thus an appropriate choice for a structural unit of realization in an implementation of systemic grammar. The structures used in a TAG have an enlarged domain of locality that factor all dependencies (e.g., agreement constraints, predicate-argument structure, filler-gap depen-

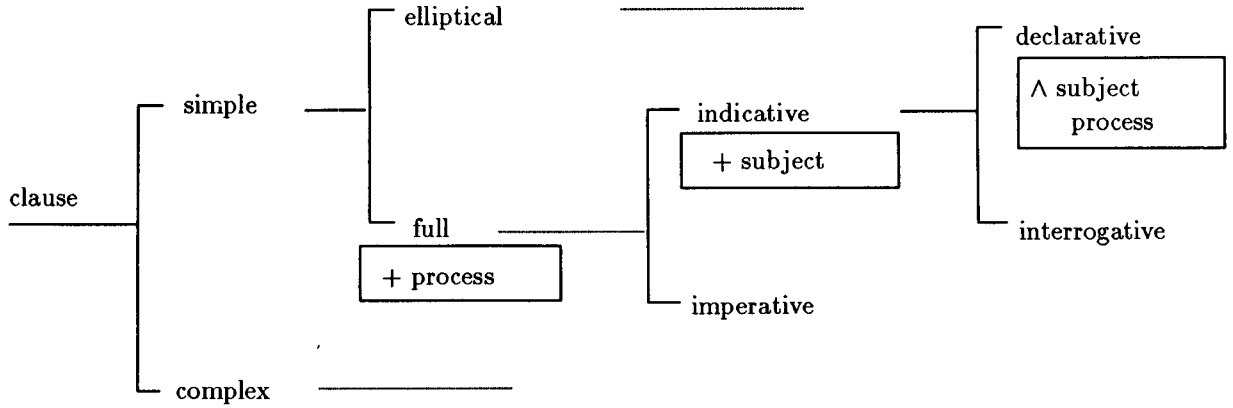


Figure 1: A Simple Systemic Network

dencies or movement). Also the composition operations allow for the interspersing of words to obtain the ordering of utterances with or without discontinuity.

In this paper we argue that if the structures used by TAG are taken as the structural units for systemic grammar with the operations of composition of TAG used to fit structures built at lower ranks into the evolving structures, then the generation of the above sentence can be done in a straight forward and general manner while at the same time maintaining the independence criterion. This sentence could be generated by adjoining a tree corresponding to "John thought" into the third tree shown in figure 3 which, after lexical insertion, corresponds to the string "It was Mary that Bill liked". As we will show later, this tree (rather than the other two trees in figure 3) is considered because of the decision to extract "mary". Notice that the adjoining operation allows the strings for the two independently derived clauses to be interspersed with each other.

## Lexicalized TAGs: An Introduction

The grammatical formalism that we have chosen is a Tree Adjoining Grammar (TAG) [Joshi 1985]. The enlarged domain of locality in a TAG grammar enables the kind of processing necessary to handle the identified problems in the systemic grammar. The lexicalized and feature-based aspects of the formalism enables ease of processing within the systemic framework.

TAG is a tree generating system where the basic building blocks are tree structures. It is over these tree structures that constraints may be placed. A TAG grammar consists of two finite sets of elementary trees: the initial trees and auxiliary trees. The initial trees represent a minimal linguistic structure. The auxiliary trees represent a minimal recursive structure that might be brought into a derivation. The frontier of each auxiliary tree contains one node, termed the foot node, which is a non-terminal identical to the root node of the tree.

Structures are derived in a TAG using the operations

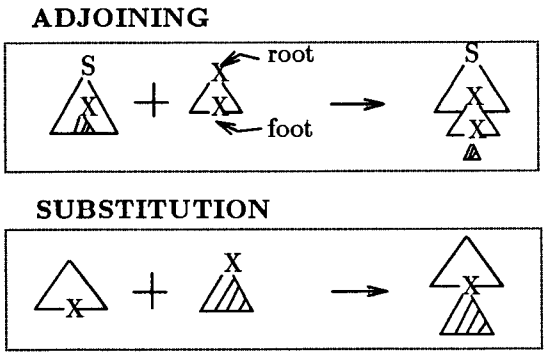


Figure 2: Adjoining and Substitution Operations

*adjunction* and *substitution* which are described pictorially in figure 2. As shown in the figure the adjoining operation first excises the subtree below the node of adjoining; inserts the auxiliary tree at this position; the excised subtree is now inserted at the foot node of the auxiliary tree used for adjunction. The substitution operation replaces a node in the frontier of a tree with a tree whose root is identical to that node.

We will use a *feature structure based* TAG[Vijay-shanker & Joshi 1988]. That is, feature structures may be associated with various nodes in the tree. When adjoining or substitution is done, the feature structures associated with the affected nodes must be unified. The operation may only be done if the required unification is successful. This will not be discussed any further here.

The TAG that we used is *lexicalized* in the sense described in [Schabes *et al.* 1988]. This means that the TAG is organized around the lexicon with lexical items being associated with sets of elementary trees in which that lexical item participates. The lexical item which chooses a given elementary tree is termed the *anchor* of that tree.

Trees are grouped into *tree groups* where each tree in a tree group has the same underlying (logical form) structure. Trees within a tree group are distinguished

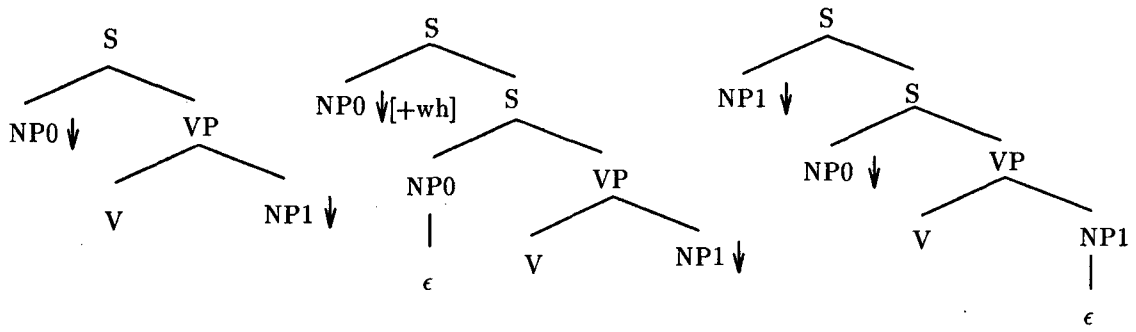


Figure 3: A Tree Group Selected by *Like*

from one another by syntactic variations (which we shall call transformations). For example, the verb *like*, which takes a nominal subject and a nominal object, selects the transitive tree group. Some of the members of this tree group are shown in Figure 3. The figure contains three initial trees, the first corresponds to a declarative sentence, the second to a wh-question on the subject, and the third to an it-cleft construction on the object.

## S-TAGs

The processing within a Systemic Tree Adjoining Grammar (S-TAG) is similar to that in systemic grammar (e.g., the networks are traversed and the realization operators associated with the choices taken are evaluated). In S-TAG we have already stated that the individual grammatical functions are realized structurally by elementary trees and that elementary trees provide the bounding scope for application of realization operators. Thus, the "functions" which are inserted into the systemic structure will be associated with elementary trees in the TAG formalism. While the types of realization operators required by S-TAG will be the same as for general systemic grammars, the individual operators will be tailored to the TAG formalism.

### Regions in S-TAG

The basic processing within a systemic grammar must take into account two dimensions of processing decisions:

1. Metafunctional domains. Structures are built in the three metafunctional domains (ideational, interpersonal, and textual) simultaneously. Certain realization operators are used to "conflate" the independently built structures.
2. Processing from one "rank" to another. It is through changes in rank that semantic structures are eventually realized as surface form. The general methodology is to insert functional units into a structure. Following this, these functional units are refined by re-entering the network at a lower rank. This process continues until a surface structure has been fleshed out.

While the processing in the S-TAG grammar follows the same principles, we differ in some implementation issues to accommodate TAG. One of the major contributions of this work is in the processing from one rank to another. In particular, this work makes explicit the bounding domains for the realization operators which are responsible for realizing a given grammatical function. Thus it becomes clear what is available for manipulation when a network is entered (and re-entered for specifying a function inserted during the initial network traversals). We employ the notion of a region for this purpose.

In general a region is created to expand a grammatical function. Since we have said that elementary trees are appropriate structural units for realizing the functions and for the bounding domains for the realization operators, we state that an elementary tree will eventually be associated with every region. The appropriate elementary tree will be chosen after a decision has been made to insert a lexical item. Informally, this lexical item will be the lexical anchor of the elementary tree that will be chosen in the region. For this reason we will call this lexical item the lexical anchor of the region also.[1] The region serves as the bounding domain on the realization operations. All realization operations

[1]This approach has interesting consequences, such as adopting a head driven generation strategy within the sys-

used within a region are applicable on the features of the lexical anchor of the region or the tree selected by this anchor. In the section on "Lexicon and Tree Groups" we will discuss how the features of the anchor, the tree groups selected, and trees selected will be maintained in a region.

Once a lexical anchor is picked, the tree groups associated with that anchor will be considered. The choices in the network will cause realization operators to be evaluated which will narrow this set of trees to one. This single tree is then said to be associated with the region and will be the structural realization of the grammatical function being expanded (whose expansion was the reason for the creation of the region). This filtering will be done by using realization operations that select between tree groups and those that select tree members within tree groups. Such realization operations will be discussed in the section on "Realization Operators".

Based on the characteristics of the tree associated with a region and directives from the networks being traversed, decisions to expand certain grammatical functions (previously inserted in the region) will be made. Sub-regions will be created to expand these functions and the network will be re-entered to determine the expansion. Notice that this will cause an elementary tree to be associated with each sub-region. These sub-regions must eventually be combined with the super-regions which spawned them using realization operations for adjoining and substitution.

This view of the process is potentially complicated since some of the realization operations to be evaluated in the region may not be applied before the set of trees being considered is narrowed down sufficiently. For example, an operator which selects a particular tree need not be applied until the tree groups have been narrowed to one. If at the point an operator which selects a tree is called for, if the number of tree groups has not been reduced to one, it serves no purpose to apply the operation on each tree group. Hence, in such cases, within a region we will maintain a record of realization operations that have to be completed later. These operations will be applied at the appropriate time.

## Lexicon and Tree Groups

In the lexicon, a lexical item will be associated with a set of tree groups, each of which contains a set of elementary trees. We choose to represent a tree group in the lexicon as a feature-structure/tree-group-name pair. The feature-structure includes all of the common features of the trees within the tree groups, the features of the lexical item itself, and its lexical idiosyncrasies, if any. The tree group name can be thought of as a pointer to the actual tree group kept in a separate area (allowing for sharing of tree groups by lexical items).

temics framework. It will also have implications on the design of the network.

For example, with the lexical item, *walk*, we will associate the pairs $(f_{trans}, trans)$, $(f_{intrans}, intrans)$, $(f_{noun}, noun)$. *trans*, for instance, is the name of the tree group for transitive verbs. All trees in this group share the information that the lexical anchor is a transitive verb. Thus, this information is stored in $f_{trans}$ along with any other features that are common to the trees in the group. The other two pairs represent the fact that *walk* can be used as a intransitive verb as well as a noun.

The trees that constitute a tree group are kept together. Some realization operators which will be evaluated in a region will refer to certain grammatical functions that are represented as nodes in the tree associated with that region. Hence we will use a mapping table that maps abstract positions (grammatical functions) to actual nodes in an elementary tree.

## Realization Operators

Having set up the notion of a region (and its associated elementary tree) as the bounding domain over which the realization operators can function, we are now in a position to discuss some of the realization operators that will be used in S-TAG. These operators parallel those found in other systemic grammar implementations, although they are particular to the use of TAG. According to [Matthiessen & Kasper 1985] the realization operators used in a systemic network can be viewed along three dimensions: (1) Structuring (which defines the structure and its organization within one rank and within one functional domain), (2) Rank (which "organizes the grammar into a scale of units: clause - group/phrase - word - morpheme" [Matthiessen & Kasper 1985, p. 25]), and (3) Metafunctional layering (which integrates the structures developed within the various metafunctional domains (e.g., interpersonal, ideational, and textual)).

We concentrate on the rank and structuring operators because they appear to be most affected by the addition of TAG. Aside from the nature of the actual structural units, a major difference between S-TAG and previous implementations of systemics is that previous implementations have built up structures of minimal import: upon proper evidence a functional unit is added to the current structure, ordered with respect to the other elements, accumulates features, and is then expanded so as to satisfy those features. There appears to be no automatic mechanism for carrying out syntactic implications of decisions that have been made. In S-TAG we take an opposite approach. In the TAG we have precompiled minimally complete packages of syntactic structure. Rather than building up structure only when we have enough evidence to know that it is correct (as has previously been done), our operation can be characterized as deciding between the syntactic possibilities that are consistent with what is known

at the given point in the network.[2] As a result many of the structuring operators we introduce are designed to narrow down the trees that could possibly realize a particular function.

**Introducing the Lexical Anchor: Insert(Lex-item)** When the lexical anchor is identified in the network, this operation will be used. The purpose of this operation is not only to introduce the lexical anchor into the region but also to bring the associated set of feature-structure/tree-group-name pairs. Thus the tree group itself is not brought in but is indirectly accessible. A tree is brought into the region only after the narrowing process is completed. The anchor is then inserted into the tree.

**Filtering Tree Groups in a Region** We will use one realization operation to choose among the the tree groups being considered in a region. This choice is made on the basis of some features that become known during the traversal of the network and is basically a decision about the functional units the realization must represent. Thus, in some sense it is analogous to the "insert" operator in Nigel. For example, the insertion of a particular lexical item, say *walk*, will bring into consideration all possible tree groups it can participate in. If it becomes known (in the transitivity network) that the recipient function will have to be realized, then among the various tree groups of the lexical anchor of the region, only the appropriate tree groups (such as those corresponding to transitive verb form) will have to be considered.

For current purposes, the realization operation that filters the tree groups will be called *Select-Group* which takes a feature as an argument. In the above example, the network may cause the operation: Select-Group(transitive) to be evaluated. Recall that the three tree-groups referenced for this lexical item are represented by the pairs: $(f_{trans}, trans)$, $(f_{intrans}, intrans)$, and $(f_{noun}, noun)$. Since the feature-structures $f_{trans}, f_{intrans}, f_{noun}$ are kept in the lexicon itself rather than with the tree group, these tuples will be brought into the region on lexical insertion. If the realization operation Select-Group(transitive) (which is analogous to insert process and recipient in the Nigel grammar) is evaluated in the region, the feature transitive is unified with the three feature-structures $f_{trans}, f_{intrans}, f_{noun}$. Since this feature is only consistent with the features in $f_{trans}$, only the pair $(f_{trans}, trans)$ will remain in the region.

**Selecting Trees from a Tree Group** The realization operation used to narrow down the choice of elementary trees within a tree group considered in a region is called *Select-Tree*. We had described a tree group to correspond to a specific semantic entity with all of its relevant semantic features inserted. The group itself represents all syntactic realizations of this entity. Therefore the purpose of this operation is to choose among different syntactic forms possible. Its effect is somewhat analogous to that of the "order" operators in Nigel. For example, if during the traversal of the network it is realized that the object is to be topicalized then the Select-Tree operation will be evaluated. Among the various syntactic variations possible, the tree(s) which realize this thematization will thus be identified.

**Composing Trees** Recall that sub-regions are created to expand grammatical functions. The elementary trees associated with the sub-regions are to be composed with the tree associated with the super-region either by substitution or by adjunction. Expansion of a grammatical function is done, in the Nigel grammar, when a function is preselected with a set of features. The preselected features determine where to re-enter the network in order to expand the given function. The resulting realization will replace the original function in the eventual realization of the input. In S-TAG this is accomplished by using the realization operation *Expand(function, features)*. This will cause the creation of a sub-region (which is named by the function). The realization of the function will occur in this sub-region by re-entering the network at a point determined by the preselected feature (as in Nigel).

The tree which eventually realizes the function must be composed (by substitution or adjoining) with the tree in the super-region at the node corresponding to the function (as given by the mapping table). The decision to adjoin or substitute is made based on the types of the trees that are picked in the sub- and super-regions.

## Discussion

The strongest asset of systemic grammar is its view of generation as a goal-directed enterprise with emphasis laid on function rather than form. While our work involves the incorporation of a syntactic formalism into systemic grammar, we have not departed from the general approach of systemic's view of generation. Systemic linguists, however, have not been interested in the details of the mapping between functional choices and the resulting form. In particular, they are not concerned with the details of the structural units that are realized. In a computer implementation, a programmer needs to be concerned about the details of the structural units, how they are realized, and how the constraints of systemic grammars are translated as principles of implementation. It is in this context that we propose the use of TAG trees as appropriate structural units and examine the processing paradigm (and its logical con-

[2]Note it is not necessary to bring all of the syntactic structures into the region, rather much of this processing can be done based on the features stored with the lexical anchor.

sequences) that follows from such a choice. Thus, the incorporation of TAG is more than just a simple addition of a syntactic formalism to the systemic framework. We argue that the incorporation of TAGs enriches a systemic grammar implementation for the following reasons:

First, systemic linguists have stressed the notion of stepwise semantic decomposition as a constraint on any implementation of a systemic grammar. Hence it is not unreasonable to expect the realization of the form to conform to the decomposition of the semantic units. We have called this the independence criterion, indicating that independent decomposed functional units be realized independently in any implementation of systemic grammar. We argued that in order to be consistent with this paradigm, we have to choose appropriate structural units as realizations of semantic/functional pieces. These units must capture all necessary structural relationships, and should be the bounding domains for the realization operators that build them. Under these conditions, we argued that the structural units should have a "large" enough notion of locality to be able to factor out all structural dependencies. Since the elementary structures of TAGs are "minimally complete" to allow for the factoring of dependencies, we have argued that they are appropriate structures that can be built and manipulated in an implementation of systemic grammar. They also form appropriate bounding domains for the realization operators. Our preliminary work on incorporating TAGs in the systemic framework gives us encouragement to believe that this is indeed the case.

Second, in addition to justifying the use of TAG structures for systemics, we can show that we can handle the discontinuity (which we did not explicitly discuss for lack of space) and long distance dependency problem which plague other implementations of systemic grammars. The key point to make here is that not only are these handled but that generation of utterances with discontinuity or long distance dependencies is conceptually no different than generation of utterances without any form of discontinuity.

Third, systemic grammar places emphasis on function over form and makes clear that functional distinctions in the input manifests themselves in the different available forms. It is clear that our approach brings this aspect of systemics to the fore front. Note that the different trees in a tree group yield various syntactic realizations of a single predicate-argument structure. As we step through the network, various choices are made on the basis of the functional content of the planned utterance. These choices will result in choosing one syntactic realization over another.

Finally, what we have suggested calls for putting together two formalisms so that a mainly semantics driven processor (systemics) is able to reap some of the advantages of a syntax driven (TAG) approach. Currently a systemic network employs limited mechanism for carrying through the syntactic consequences of the decisions that it makes. Thus one of two things has to happen:

1. the network designer must anticipate all syntactic consequences and explicitly state each of them at the time a decision is made. This is not an ideal solution, especially when the network becomes very large.
2. the system must depend on the environment being consistent in order to carry out the desired consequences. In this case the syntactic consequences are strung throughout the network (perhaps prompted by different questions that are asked). The environment must be counted on to answer those questions in a consistent fashion. Even if the information is straightforwardly captured in the environment (which is unclear), due to lack of the ability to carry out syntactic consequences, it becomes necessary to ask questions of the environment (to make choices) that are redundant. In addition, this arrangement goes against the systemic enterprise in which the environment keeps track of semantic content.

The addition of TAG allows an independent mechanism (e.g., the TAG processing) to maintain consistent syntactic consequences of decisions made. For example, agreement constraints are precompiled into the tree. Also, for example, given the choice of a particular lexical item – once the trees for that item have been narrowed to one, the tree itself will contain the information about what functions must be expanded. Thus this information need not be included in the network as well.

We believe that network design will be simpler because the incorporation of TAG makes possible clear demarcation of semantic choices from syntactic consequences. Also it allow for the separation of lexical idiosyncrasies into the lexicon rather than the network.

Our work so far has been concerned with identifying the TAG structures as appropriate structural units in a computer implementation of a systemic grammar. The implementation decisions that have been discussed are given to indicate the logical consequences of incorporating the TAG formalism in the systemic paradigm. These consequences would necessarily be handled in any actual implementation (e.g., breaking processing into regions, associating elementary trees with regions, the nature of realization operators). Considerable work remains to be done. We need to investigate the consequences of using the TAG formalism on the design of the systemic network especially in terms of uncovering redundancy and separation of syntax, semantics, and lexicon design. While currently used networks will be helpful in this task, we anticipate considerable revisions in the network design due to the incorporation of TAG. Furthermore, we have only examined some functional domains and a subset of realization operations that will be required. These topics are the focus of our current research.

## References


[Bateman & Paris 1989] Bateman, J. and Paris, C. 1989. Phrasing a text in terms the user can understand. In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, IJCAI-89, Detroit, Michigan: 1511–1517.

[Davey 1978] Davey, A. 1978. *Discourse Production.* Edinburgh University Press, Edinburgh.

[Fawcett 1980] Fawcett, R.P. 1980. *Cognitive linguistics and social interaction.* Julius Groos Verlag Heidelburg and Exter University.

[Halliday 1985] Halliday, M. A. K. 1985. *An introduction to functional grammar.* Edward Arnold, London England.

[Hudson 1971] Hudson, R.A. 1971. *English Complex Sentences: An Introduction to Systemic Grammar.* North Holland.

[Joshi 1985] Joshi, Aravind K. 1985. How Much Context-Sensitivity is Necessary for Chracterizing Structural Descriptions : Tree Adjoining Grammar. In: D. Dowty, L. Karttunen, and A. Zwicky, Eds., *Natural Language Processing : Theoretical, Computational and Psychological Perspectives.* Cambridge University Press, New York.

[Mann 1983] Mann, William C. 1983. *A Linguistic Overview of the Nigel: Text Generation Grammar.* Technical Report ISI/RS-83-9, ISI/USC.

[Mann & Matthiessen 1985] Mann, W. and Matthiessen, C. 1985. Nigel: A systemic grammar for text generation. In: O. Freedle, Ed., *Systemic Perspectives on Discourse.* Norwood, NJ.

[Matthiessen & Kasper 1985] Matthiessen, Christian and Kasper, Robert. 1985. Representational Issues in Systemic Functional Grammar -and- Systemic Grammar and Functional Unification Grammar. In: *12th International Systemic Workshop*, Ann Arbor, Michigan, Also appears as : ISI/USC Technical Note RS-87-179, May 1987.

[Patten 1988] Patten, T. 1988. *Systemic Text Generation as Problem Solving.* Cambridge University Press, Cambridge.

[Schabes *et al.* 1988] Schabes, Y., Abille, A., and Joshi, A. 1988. Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. In: *Proceedings of the 12th International Conference on Computational Linguistics (COLING' 88)*, Budapest, Hungary.

[Vijay-shanker & Joshi 1988] Vijay-shanker, K. and Joshi, Aravind K. 1988. Feature Structure Based Tree Adjoining Grammar. In: *Proceedings of the 12th International Conference on Computational Linguistics (COLING' 88)*, Budapest, Hungary.