Phrase-structure grammars assign a unique phrase structure (constituency) to an unambiguous sentence. Thus, for example, *John likes apples* will be bracketed as follows (ignoring the phrase labels and ignoring some brackets not essential for our present purpose):

(1) (John (likes apples))

There are systems, however, for example, Combinatory Categorial Grammars (CCGs) (Steedman, 1990) which assign multiple structures to unambiguous strings. Thus CCG assigns the following two groupings to *John likes apples*:

(2) (John (likes apples))
(3) ((John likes) apples)

The justification in CCG for such multiple structures is their use in coordination and in defining intonational phrases. Thus the bracketing (2) is necessary for (4) and the bracketing (3) for (5).

(4) (John ((likes apples) and (hates pears)))
(5) (((John likes) and (Bill hates)) beans)

Also, (2) corresponds to the intonational phrasing if the previous context is (6) and (3) if the previous context is (7).

(6) Who likes apples?
   (John (likes apples))
(7) What does John like?
   ((John likes) apples)

The work on CCG was presented by Mark Steedman in an earlier DARPA SLS Workshop (Steedman, 1989).

In this paper, we show how a CCG-like account for coordination can be constructed in the framework of lexicalized tree-adjoining grammars (TAGs) (Joshi, 1987; Schabes et al., 1988; Schabes, 1990). In particular, we show how a fixed constituency can be maintained at the level of the elementary trees of lexicalized TAGs and yet be able to achieve the kind of flexibility needed for dealing with the so-called non-constituents. This is the key significance of this contribution. In a CCG, being a constituent is the same as being a function. We show that this need not be the case and standard notions of constituency can be maintained. The key idea is that we use the lexicalized trees of TAG as structured categories with the associated functional types. Because of lack of space, we will illustrate our ideas by examples only.

Lexicalized TAGs (with substitution and adjunction) are similar to CCGs in the sense that for each lexical item the elementary tree(s) which is (are) anchored on that lexical item can be regarded as the (structured) category (categories) associated with that item. Figure 1 and Figure 2 give examples of elementary trees that can be found in lexicon for a Lexicalized TAG. The associated functional types are shown below each elementary tree.

Furthermore, each tree can be interpreted as a function on the types of arguments it requires. For example, we say that the category of the verb *gave* (see Figure 1) is the elementary tree associated with it and not the primitive category V; the functional interpretation of its category, $NP \times NP \times NP \rightarrow S$, is a function expecting three trees rooted in $NP$ and which returns an $S$-rooted tree. By combining elementary trees with substitution or adjunction, we can assign a structured category (the derived tree) and a functional interpretation to sequences of lexical items even in the cases when the sequence is discontinuous or when it does not define a constituent in the conventional sense. See

1 This research is partially supported by DARPA grant N0014-85-K0018, ARO grant DAAL03-89-C-0031PRI and NSF grant-IRI-94-10413 A02. We are grateful to Jamie Henderson, Anthony Kroch, Mitch Marcus, Stuart Shieber, Mark Steedman and K. Vijay-Shanker for providing valuable comments.

2 It is known that TAGs are weakly equivalent to CCGs, i.e., they both generate the same sets of strings but not strongly because they do not assign the same structural descriptions.
The coordination schema $(\&)$ combines two lexical strings with their structured categories and their functional types: $(l_1, \sigma_1, r_1) \& (l_2, \sigma_2, r_2) = (l, \sigma, r)$, where $l_1, l_2, l$ are lexical strings; $\sigma_1, \sigma_2, \sigma$ are structured categories (trees); and $r_1, r_2, r$ are functional types.

The lexical strings in Figure 3 are John eats, eats cookies, thinks John eats, and gave NP D book. The first three strings are contiguous but the fourth string is not contiguous, in the sense that it is interrupted by one or more non-terminals, which are the argument positions for the associated functional type. We will say that the first three strings satisfy the Lexical String Contiguity (LSC) condition and the fourth string does not satisfy LSC. Our structured categories are like un-Curried functions. LSC allows us to achieve Currying in a certain sense. Henceforth we will require that the structured categories that enter into coordination as well as the structured category resulting from coordination always satisfy LSC.

The coordination $(l_1, \sigma_1, r_1) \& (l_2, \sigma_2, r_2)$ succeeds if:

- the lexical strings $l_1$ and $l_2$ both satisfy LSC;
- the functional types are identical ($r_1 = r_2 = r$);
- the least nodes dominating $l_1$ in $\sigma_1$ and $l_2$ in $\sigma_2$ have the same label.

The resulting structured category, $\sigma = \sigma_1 \& \sigma_2$, is obtained by:

1. equating the corresponding shared arguments in $\sigma_1$ and $\sigma_2$ (preserving linear precedence of arguments in $\sigma_1$ and $\sigma_2$);
2. coordinating at the least nodes dominating $l_1$ and $l_2$;
3. collapsing the supertrees above the nodes at which coordination was made;
4. selecting the argument positions such that LSC holds for $\sigma$;
5. if the anchor of $\sigma_2$ is the same as the anchor if $\sigma_1$, then the anchor of $\sigma_2$ is erased and equated with the anchor of $\sigma_1$ (the decision to erase the anchor of $\sigma_2$ is based on the fact that the complements of the anchor must always be in the appropriate direction, on the right for English).

Now we will give a series of examples to illustrate the coordination schema. Figure 4 shows how Mary a book and Susan a flower can be coordinated to derive sentences like:

\[ \text{John gave Mary a book and Susan a flower} \]

In Figure 4, the tree corresponding to gave Mary a book and Susan a flower has been obtained by:

1. equating the NP nodes in $\sigma_1$ and $\sigma_2$;
2. coordinating the VP nodes;
3. collapsing the supertrees above the VP nodes;
4. selecting the left most NP as argument;
5. erasing the anchor (gave) in $\sigma_2$ and equating the anchor node in $\sigma_2$ with the V node in $\sigma_1$.

Similarly, the sentence,

\[ \text{John likes and Bill hates bananas} \]

is obtained by coordinating John likes and Bill hates (see Figure 5).

Note that John likes and Bill hates have been ‘coordinated’ but John likes and Bill hates have not been grouped together (i.e., bracketed as constituents). The phrase structure of the elementary trees has been preserved. This is in contrast to the analysis provided by CCG. CCG groups John likes and Bill hates as constituents and then invokes the coordination schema $X = X_1 \& X_2$ where $X$ is a constituent. John likes is turned into a constituent in a CCG by ‘type-raising’ John to a category which essentially encodes the information that John is in the subject position. In the elementary tree $\sigma_1$ the structure already encodes the information that whatever is substituted for the leftmost NP in $\sigma_1$ is in the subject position.

Some additional examples follow.

\[ \text{John eats cookies and drinks beer (see Figure 6)} \]
\[ \text{John cooked and ate the beans (see Figure 7)} \]

Examples in which $\sigma_1$ and $\sigma_2$ invoke more than one elementary tree can also be handled in a similarly fashion. We will only give the examples and not show the trees due to the lack of space.

\[ \text{John thinks Mary and Bill believes Susan will win.} \]
\[ \text{John gave Mary three and Susan four recently published novels.} \]

So far, we have not said anything about the so-called gapped sentences, for example:

\[ \text{John thinks Mary and Bill believes Susan will win.} \]
(14) John likes apples and Bill pears.

It can be shown that the gapped sentences and other sentences related to gapped sentences have to be obtained by assuming that the left conjunct (σ₁) is built up to S, i.e., its functional type is a constant, S. A structured category, σ, (where the functional type is a constant S) can be viewed retroactively as corresponding to various functional types, for example, NP × likes × NP → S.

Note that this functional type cannot be obtained by staring with σ in Figure 2, where the functional type is NP × NP → S.

We now take σ₂ to be of the same functional type as σ₁, i.e., NP × likes × NP → S and instantiate the coordination schema as before. Note that the lexical anchor of σ₂ is guaranteed to be the same as the lexical anchor of σ₁ because both have the functional type NP × likes × NP → S. Hence, the anchor in σ₂ will be erased following the specification in the coordination schema described earlier⁴. We will not discuss all the details of this retroactive approach due to lack of space. This approach also handles sentences which are closely related to gapping, for example,

(15) John likes apples and pears (too)

The too is introduced to show that the interpretation is different from the case where apples and pears is obtained by NP and NP coordination. In (15) we have S and S coordination.

In summary, we have shown how constituency and functional types can be kept apart and still the kind of flexibility in the constituent structures that CCG allow can be realized by using lexicalized TAG with an associated coordination schema. In an expanded version of this paper, we will describe several details concerning (1) how this approach extends to coordination with cross-serial dependencies (as in Dutch) as well as to languages with constituent orders different from English, (2) some processing implications and (3) the computation of the semantic interpretation using the machinery of synchronous TAG (Shieber and Schabes, 1990).

REFERENCES
Aravind K. Joshi. 1987. An Introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, Mathematics of Language. John Benjamins, Amsterdam.

---

⁴This approach is inspired by Steedman's approach to gapping, which depends on type-raising. Steedman requires an additional stipulation to rule out certain unwanted consequences of type-raising. It appears that in our approach this problem can be avoided. Space does not permit us to give all the details.
S
A
S NP
VP
NP
V NP
NP VP V NP
N V NP
eats N
I I I
John eats cookies
NP --~ S NP .-9S

S NP VP NP
V NP
thinks NP VP NP
V NP
NP
gave D N
John eats book
NP xNP --~ S NP xNP xD--~S

Figure 3: Examples of derived trees with their functional types.

σ1:
σ2:
σ:

gave N D N gave N D N gave N D N gave N D N
Mary a book Susan a flower Mary a book Susan a flower

11: gave Mary a book 12: gave Susan a flower 1: gave Mary a book and Susan a flower
τ1: NP → S τ2: NP → S τ: NP → S

Figure 4: Coordination of Mary a book and Susan a flower.

σ1:
σ2:
σ:

John likes Bill hates John likes Bill hates
NP VP NP VP NP VP NP VP
I I

11: John likes 12: Bill hates 1: John likes and Bill hates
τ1: NP → S τ2: NP → S τ: NP → S

Figure 5: Coordination of John likes and Bill hates.

198
Figure 6: Coordination of eats cookies and drinks beer in (10).

Figure 7: Coordination of cooked and ate in (11).