Lecture 13

Syntactic Dependencies Over Trees

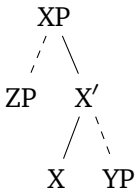
We found out that CFGs, strictly local tree grammars, and refined strictly local tree grammars all have the same weak generative capacity but differ in their strong generative capacity such that CFG $\subsetneq \mathrm{SL}_2^{\mathrm{T}} \subsetneq \mathrm{SL}_3^{\mathrm{T}} \subsetneq \cdots \subsetneq \mathrm{SL}_2^{\mathrm{T,R}} = \mathrm{SL}^{\mathrm{T,R}}$. This leaves us with the question which one of them is adequately powerful to express syntactic dependencies over trees. Although this issue involves barely any mathematics, it turns out to be a lot harder to answer than anything we have looked at so far.

1 The Complexity of Syntactic Dependencies

1.1 Headedness and Projection

Both CFGs and strictly local tree languages are egalitarian in the sense that all nodes are on equal footing without one enjoying a special status. For example, a rewrite rule like AP → B C is no less natural than AP → A C. This contrasts with one of the important findings of 20th century linguistics: sentences don't just have tree structures, these structures are the result of combining phrases, and each phrase is projected by a head.

These ideas were formalized in terms of X′-theory, where each lexical item X is the head of a phrase XP (Jackendoff 1977) as shown below.



Note that every phrase is at most unary branching. X′-theory has been subjected to extensive criticism over the years (Kornai and Pullum 1990; Chomsky 1993) and is no longer entertained in its canonical form, with current models positing that I) all phrases are strictly binary branching, ruling out unary branching nodes, and II) nodes are labeled in a less elaborate manner. The basic idea that trees are combinations of headed phrases, however, is still upheld across a variety of formalisms.

X′-theory can be defined as a restriction on the shape of rewrite rules, so it does not fall outside the purview of strictly 2-local tree languages.

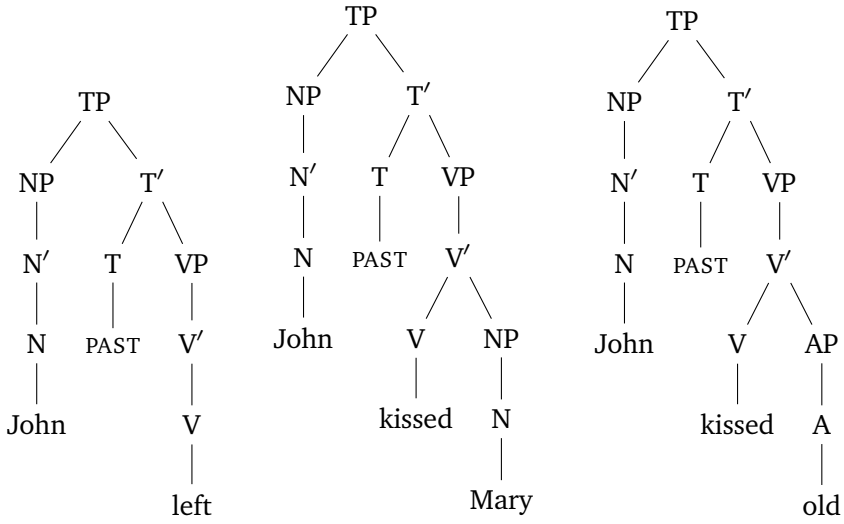
X′-template Every rewrite rule $A \to A_1 \cdots A_n$ must satisfy the following rules:

- $n \leq 2$,
- if $A = XP$ for some X, then either $A_1 = X'$ and $n = 1$, or $A_2 = X'$ and $A_1 = YP$ for some Y.
- if $A = X'$ for some X, then $A_1 = X$ and either $n = 1$ or $A_2 = YP$ for some Y.
- if $A = X$, then $n = 1$ and A_1 must be a lexical item.

Note that every CFG can be rewritten to comply with the X′-template if one adopts the standard assumption that the empty string can be the head of a phrase. Therefore X′-structure has no effect on weak generative capacity.

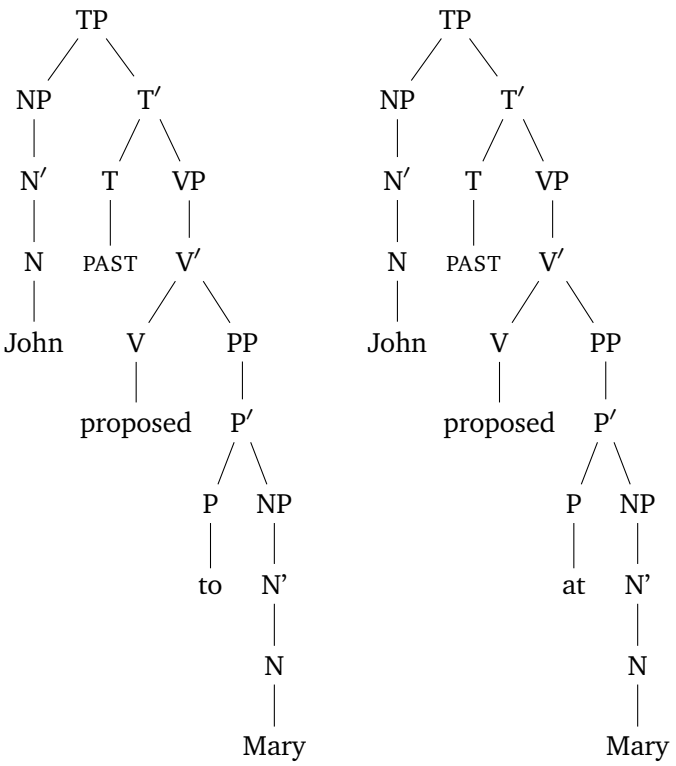
1.2 Subcategorization, Argument Structure, and Lexicalization

Even though X′-theory revolves around the notion of heads and how they regulate structure building, X′-theory by itself is not enough to capture another essential property of language, namely that heads can allow or block the presence of other phrases. More precisely, the head X in an XP controls whether YP and ZP are allowed to be present, and if so, what they must look like. For example, the intransitive verb *left* allows neither ZP nor YP, whereas transitive *kissed* allows YP to be realized as an NP but not as an AP. This property is also known as *subcategorization*, and linguists sometimes speak of a word's *subcategorization frame*.



Sometimes it is not enough to know the general type of YP or ZP and one must also look at the heads of those phrases, as is the case in the following example. Crucially, though, there seem to be no cases in natural language where XP allows a YP depending on what other phrases are contained by the YP.

This is a rather sloppy statement, and under its most literal interpretation it is actually false: long-distance dependencies can indeed prompt behavior where it looks like a head rejects a YP based on what other phrases it contains. A more careful generalization would have to factor out these aspects, which seem to be independent of subcategorization.



With the labeling mechanism of X′-theory, subcategorization is strictly k-local only if the distance between the root of an XP and its head is fixed. This should be the case given everything we have seen so far, where a subtree of depth 3 can always span from XP to the head. A strictly 6-local grammar thus suffices to encode subcategorization — the highest point is XP, the lowest point is the head of YP, and the smallest subtree containing these two nodes has depth 5.

Of course we can also handle subcategorization in CFGs by moving to more elaborate labels. One of the earliest strategies is the slashed category mechanism of *Categorial Grammar* (CG). Instead of simply treating *left* as a verb, it is instead assigned the type *VP* to indicate that it forms a VP all by itself. The transitive verb *kissed*, on the other hand, has the type *VP*/*NP*, which encodes that it must take an NP to its right in order to form a VP. And the past tense head T has the type (*NP**TP*)/*NP*, so that when it is combined with an NP to the right it reduces to *NP**TP*, which returns a TP after it has taken an NP as its left argument.

The CG approach has the advantage that it grounds context-free rewriting in an explicit type system. As a result one need not specify a whole rewrite grammar, a general type reduction scheme is enough:

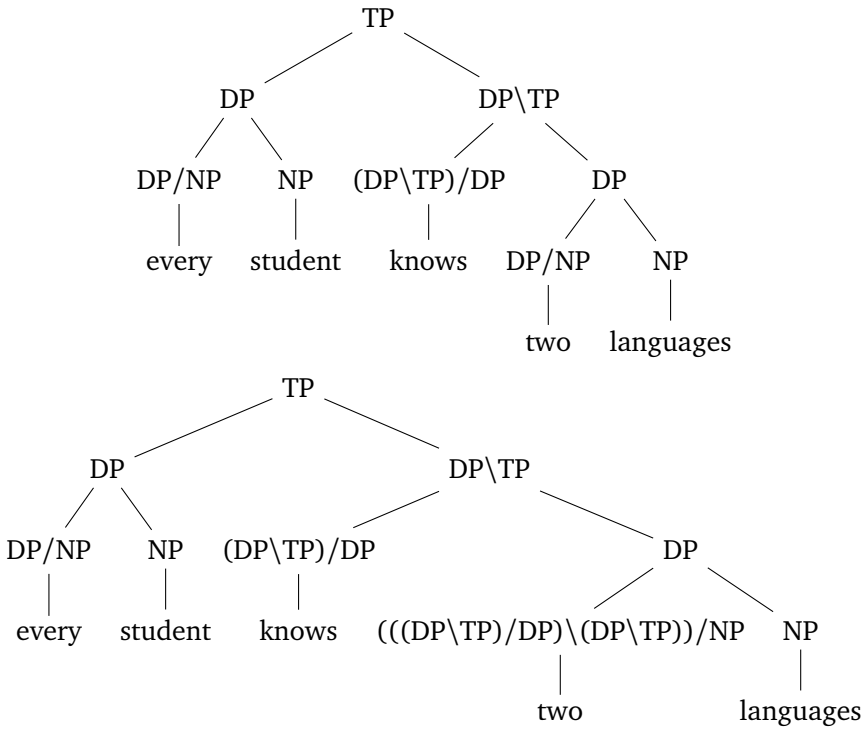
$$(\sigma/\tau) \circ \tau = \tau \circ (\tau\backslash\sigma) = \sigma$$

The language generated by the grammar is thus fully described by the lexicon, which is a list of words and their corresponding types. The CG approach thus *lexicalizes* the grammar. This can be seen as an improvement on X′-theory in the sense that the basic phrasal template is much simpler and the lexicon acts as a centralized storage for all relevant information.

The term *lexicalized grammar* also has a different usage where it refers to a restriction on the distribution of empty heads. We will look at this usage more carefully at a later point.

In addition, the use of a type system makes it easy to read the semantics of a sentence directly off its tree structure. For example, the sentence *Every student speaks*

two languages has two readings. In the narrow scope reading, it is the case that every student speaks two languages, although these languages may differ between students. In the wide scope reading, there are two specific languages that are spoken by every student, e.g. Quenya and Klingon. This difference is represented by two trees that share the same geometry but differ in their type assignment.



This higher-order type can either be assigned manually in the lexicon, or the grammar can be equipped with an additional rule to perform type shifts of this kind. As long as the number of types that can be generated this way is finitely bounded, CGs are weakly equivalent to CFGs.

1.3 Features and Agreement

A downside of the CG approach is that the type system must be very fine-grained in order to express the fact that *propose* occurs with PPs headed by *to* but not PPs that are headed by *at*. A natural evolution of the CG system is to modularize the types/non-terminal symbols by turning them into feature bundles, also known as *attribute value matrices* (AVMs).

For example, the lexical item PAST could be associated with a category feature CATV and the subcategorization features SUBCAT1 and SUBCAT2, whose respective values are $\langle \text{DP}, r \rangle$ and $\langle \text{DP}, l \rangle$ to indicate that the first argument is a DP to the right of the head and the second one a DP to the left of the head. This information can then be passed around in a local fashion via very general rewrite rules like the ones below (in $M \rightarrow M[\alpha]$, M is the result of removing α from the feature matrix):

$M \rightarrow l$ where M is the feature matrix of lexical item l

$M \rightarrow M[\text{SUBCAT1}\ \langle XP, r \rangle]\ XP$

$M \rightarrow XP\ M[\text{SUBCAT2}\ \langle XP, l \rangle]$

Note that these are no longer rewrite rules, but rather meta-rules that encode finite sets of rewrite rules. Only by plugging in the appropriate values for M, l, and XP do we eventually end up with a finite set of context-free rewrite rules. The primary purpose of the meta-rules is to link each feature to syntactic well-formedness conditions.

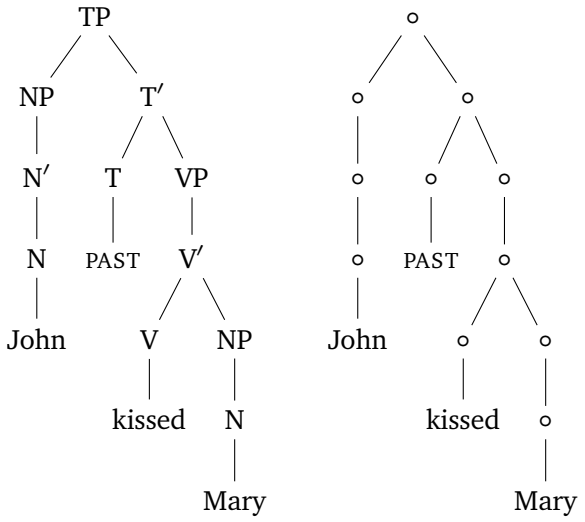
Since features can be added as needed, this system is incredibly flexible and can easily be adapted for various phenomena besides subcategorization. In English, for instance, the finite verb agrees with the subject in person and number. We can capture this by assigning nouns and verbs person and number features whose values must match. This match is enforced by expanding the subcategorization features of the T-head so that they can reference the feature make-up of the arguments. Rather than writing multiple entries for T that differ only in the values of the arguments' person and number features, we may use feature variables.

$$\begin{bmatrix} \textsc{Phon} & \varepsilon \\ \textsc{Cat} & \text{T} \\ \textsc{Subcat1} & \left\langle \begin{bmatrix} \textsc{Cat} & \text{V} \\ \textsc{Num} & \alpha \\ \textsc{Pers} & \beta \end{bmatrix}, r \right\rangle \\ \textsc{Subcat2} & \left\langle \begin{bmatrix} \textsc{Cat} & \text{N} \\ \textsc{Num} & \alpha \\ \textsc{Pers} & \beta \end{bmatrix}, l \right\rangle \end{bmatrix}$$

Other common features include tense, gender, animacy, definiteness, finiteness, and TYPE which distinguishes subtypes of common categories, e.g. mass and count nouns. In NLP, it is also useful to add semantic and pragmatic information like instrument, human, that allows for more accurate automatic analysis of sentences like *John hit the man with the black eye* versus *John hit the nail with the hammer*.

In an AVM-based system, the interior nodes of trees are no longer simple labels but complex feature matrices that keep track of all relevant information. This allows them to stay within the boundaries of CFGs, but also introduces a lot of redundancy to the tree structures — for a human, it suffices to know the feature specifications of the heads, this information need not be repeated for every node of the projected phrase. But as already discussed before, the distance between a head and the head of one of its arguments is finitely bounded, so we can omit the AVMs on all interior nodes and simply move from CFGs to some strictly k-local grammar. In fact, all interior node labels are redundant for strictly k-local grammars since every lexical item can be implicitly associated with an AVM (which might involve non-deterministic guessing in case a word has multiple entries in the lexicon), and the AVM fully determines the syntactic behavior of the word.

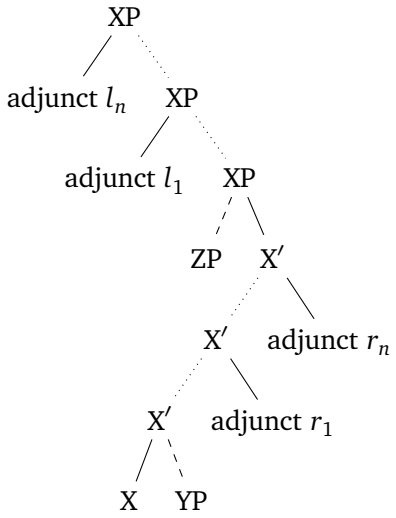
So strictly k-local grammars provide a more minimal labeling algorithm than X′-theory (or any of its recent revisions) while doing a lot more work since they also implicitly handle subcategorization and feature matching.



1.4 Adjunction

So far we have seen that three central linguistic properties — headedness, subcategorization, and feature matching — can be enforced by strictly k-local grammars without refining the labels (quite the opposite, interior node labels can be completely omitted). But a crucial assumption we made is that these relations are restricted to heads and their arguments, and that the size of a phrase is finitely bounded. At least the former, though, is likely to be wrong due to the unbounded nature of *adjunction*.

Adjuncts are elements that attach to a phrase without being explicitly licensed by the head of the phrase. They are usually optional and iterable. Adjectives are prototypical NP-adjuncts, while adverbs are adjuncts of verbs, adjectives, and other adverbs. In X′-theory, adjuncts occupy positions where their mother and their sibling have the same label.



An alternative analysis, which originated in CG, treats adjuncts as category-preserving heads. Under this view an adjunct is a phrase of category X that selects a phrase of category X. In CG terms, it is of type σ/σ or $\sigma\backslash\sigma$.

For our purposes it doesn't really matter which analysis one adopts, as the problems for strictly local grammars stay the same. Suppose that head h can select an argument

XP with head h_a, but not with head h'_a. Then we can add XP-adjuncts to XP until the distance between h and $h_a \backslash h'_a$ exceeds k. Unless the adjuncts provide sufficient information to deduce whether the head of XP is h or h_a, the grammar either allows both or neither. In both cases the grammar no longer generates the intended tree language.

This brief argument already makes it clear that adjuncts pose no challenge if we use elaborate AVMs as interior node labels, or equivalently, if we use a refined strictly local grammar to keep track of non-local information via states. But in the absence of such local coding mechanisms, they invariably render local syntactic dependencies non-local.

1.5 Displacement

Slash feature percolation from GPSG (Gazdar et al. 1985)

Movement dependencies

The fallacious argument for punctuated paths (Abels 2003)

2 Against Tree Language Dogmatism

The choice of tree structures is one of the most contentious issues in syntactic theory. Debates along these lines tend to revolve around two independent factors that unfortunately are often conflated. One pertains to what one might call the macro-constituency of the sentence, i.e. how the major phrases are positioned with respect to each other. This is the domain of constituency tests, locality effects, and so on. The other one focuses on how the macro-constituency is encoded in the tree, and it is dominated by the discussion of branching factors, labels, features, and other highly technical topics with much more abstract evaluation criteria. While syntactic formalisms mostly agree on the former (although tests are sometimes called into doubt and locality effects may be attributed to extra-grammatical factors), there is tremendous variation regarding the latter. The preceding discussion sheds some light on why that is the case.

CFGs, strictly local tree grammars and refined tree grammars define exactly the same tree languages *modulo* relabelings. Node labels can be altered as needed to fit a tree language into the desired level of expressivity. At the same time, there are no syntactic tests to determine the label of a node — whatever test one may concoct, the behavior of formalism A under such a test can also be emulated with formalism B. The observed behavior might appear more natural or plausible under formalism A, but this is not conclusive proof as it may indicate not so much a shortcoming of formalism B as a gap in our understanding of its internals. The same goes for psycholinguistic and neurolinguistic experiments. The alternatives under discussion are so close to being notational variants that a unique winner will arguably never be determined.

This is no reason for despair though — quite the opposite. Rather than trying to discern a unique correct formalism for tree languages, we should take advantage of the fact that it is so easy to translate between the different formats. Just like it causes us little concern that the computational object of a set has multiple algorithmic instantiations in Python in the form of sets, lists, and dictionaries, each one of them serving a specific purpose, we should embrace the diversity of tree languages and continuously switch between these different perspectives to further our goals.

Questions of computational complexity are best addressed by focusing on tree languages without interior node labels, where the differences between local and

unbounded dependencies surface most clearly. Strictly 2-local representations with rich interior labels do away with this distinction and thus are best suited for areas where directly handling non-local dependencies is likely to create a lot of extra work with little foreseeable pay-off. Depending on one's goals, parsing is one such area, as we will see next.