1. Introduction/Overview: Syntactic dependencies, structure, and derivational state

1 The big picture

Some important big ideas:

(1) Given an internalist/generative perspective, **grammars** are the primary object of study.
- ~~Grammars are compact summaries of observed regularities in sentences.~~
- Grammars are mental systems that determine the status of sentences, from which the observed regularities follow as consequences.
- So, what kinds of regularities arise from grammars that are structured in various ways?

(2) Grammars do interesting work by treating certain (often infinite) sets of subexpressions as **intersubstitutable** (i.e. "of the same category").

(3) The intersubstitutability of subexpressions arises when a grammar **forgets** (or ignores) the distinctions between them.

(4) The properties of a complex structure are a function of the properties of its subparts.

1.1 Very rough, incomplete historical outline

- 1950s: introduction of the systematic study of grammars (Chomsky Hierarchy) (Chomsky, 1956, 1959)
  - goal of identifying "devices with more generative capacity than finite automata but that are more structured (and, presumably, have less generative capacity) than arbitrary Turing machines" (Chomsky, 1963, p.360)
- 1960s–1990s: transformational grammar adopted as a cognitive hypothesis
  - relatively little emphasis on formal analysis or restrictiveness
  - restrictiveness in *Aspects* (Chomsky, 1965) came from the evaluation metric
  - Peters and Ritchie (1973) showed that *Aspects*-style transformational grammars were in fact formally *unrestricted*
  - restrictiveness imposed via *substantive constraints* (e.g. subjacency, parameters)
- mid 1980s: Joshi (1985) suggests (revives?) the idea of a grammar formalism as a hypothesized answer to the question "What is a human language?"
  - prompted the development of a "first wave" of *mildly context-sensitive* (MCS) grammar formalisms, e.g. Tree Adjoining Grammar (Joshi, 1985), Combinatory Categorial Grammar (Steedman, 1985)
  - surprising convergences/equivalences among these formalisms (Joshi et al., 1990)
- early 1990s: Minimalism (Chomsky, 1995) (re?)adopts the idea that *the system itself* is formally restrictive (e.g. minimality, extension condition)?
- late 1990s–2000s: Minimalism-inspired grammar formalisms (Stabler, 1997; Kobele et al., 2007) allow comparisons with the first-wave MCS formalisms, revealing both similarities and differences

1.2 Plan for the course

- Overview: Syntactic dependencies, structure and derivational state
- Classical formalisms: Finite-state automata, context-free grammars, pushdown automata
- Important "hidden" ideas: Strict locality, tree automata
- First-wave mild context-sensitivity: Tree Adjoining Grammars (TAG), Combinatory Categorial Grammars (CCG)
- Second-wave mild context-sensitivity: Minimalist Grammars (MG), Multiple Context Free Grammars (MCFG)

2 Dependency patterns

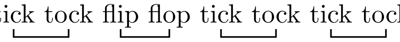
Consider three toy example languages, where

- the lexicon has just four words: 'flip', 'flop', 'tick', 'tock';
- each occurrence of 'flip' must appear with a corresponding occurrence of 'flop';
- each occurrence of 'tick' must appear with a corresponding occurrence of 'tock'.

The three languages differ in the linear arrangements of the dependent 'flip'/'flop' and 'tick'/'tock' pairs.

The first toy language exhibits what I'll call **serial dependencies**: each 'flip' must be immediately followed by a 'flop', and likewise for 'tick' and 'tock'.

(5) a. flip flop
b. tick tock
c. flip flop tick tock
d. tick tock flip flop flip flop
e. tick tock flip flop tick tock tick tock



In the second fictional language, all occurrences of 'flip' and 'tick' must appear first, followed by all corresponding occurrences of 'flop' and 'tock' in a "mirror image" order: the first word in the 'flip'/'tick' portion of a sentence is matched with the *last* word in the 'flop'/'tock' portion. This is a **nesting dependencies** pattern.

(6) a. flip flop
b. tick tock
c. flip tick tock flop
d. tick flip flip flop flop tock
e. tick flip tick tick tock tock flop tock

In the third toy language, all occurrences of 'flip' and 'tick' must again appear first, but here the corresponding occurrences of 'flop' and 'tock' are not mirrored: the first word in the 'flip'/'tick' portion is matched with the *first* word in the 'flop'/'tock' portion. This is a **crossing dependencies** pattern.

(7) a. flip flop
b. tick tock
c. flip tick flop tock
d. tick flip flip tock flop flop
e. tick flip tick tick tock flop tock tock

All three of these kinds of patterns are attested in natural language syntax.

To a pretty good first approximation:

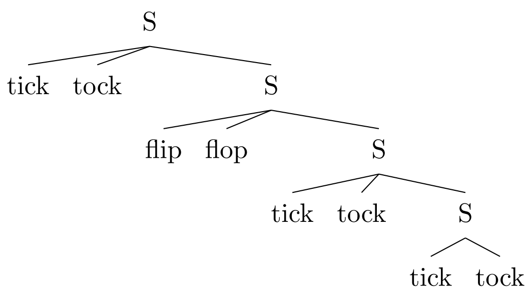
- there is wide agreement (sometimes obscured by differences in terminology and notation) about how to treat **serial** and **nesting** dependencies, but
- different syntactic formalisms express competing hypotheses about the mechanisms responsible for **crossing** dependencies.

3 Grammatical mechanisms

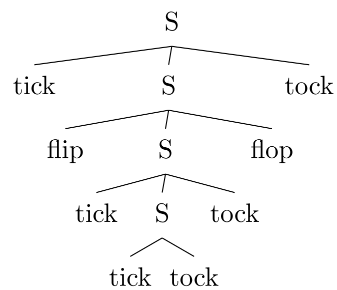
3.1 The relationship between structure and dependencies

The familiar notion of a context-free phrase-structure grammar (CFG) is one way to instantiate the common core of widely shared assumptions that can generate serial and nesting dependency patterns.

(8) S → flip flop (S)
S → tick tock (S)



(9) S → flip (S) flop
S → tick (S) tock



Important things to note about (8) and (9):

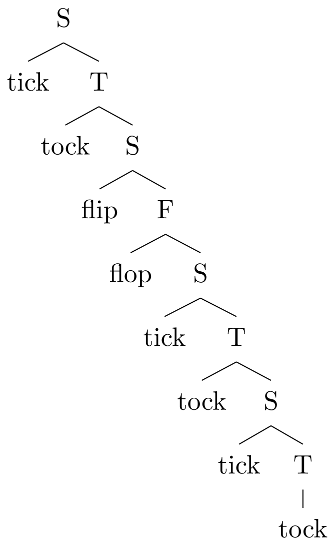
- Each rule says either
  - that an S can be comprised of one ‘flip’ and one ‘flop’ and another S, or
  - that an S can be comprised of one ‘tick’ and one ‘tock’ and another S.
- Serial dependencies arise if the “other S” is peripheral (e.g. S → flip flop S).
- Nesting dependencies arise if the “other S” is medial (e.g. S → flip S flop).

3.2 The key idea of derivational *state*

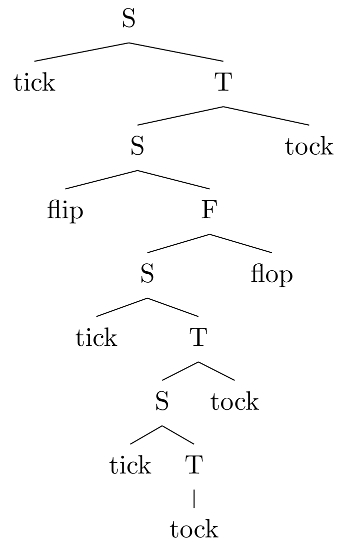
The phrase-structure rules in (8) and (9) are particularly simple in the sense that a *single rule* introduces each co-dependent pair of elements; the two members of each pair appear at exactly the same height in the tree.

In real linguistic analyses we might instead find ourselves describing such patterns using something more along the lines of (10) and (11).

(10) S → flip F
F → flop (S)
S → tick T
T → tock (S)



(11) S → flip F
F → (S) flop
S → tick T
T → (S) tock



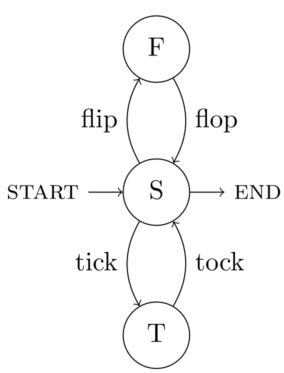
It's useful to consider the ways in which (10) and (11) do and don't differ from (8) and (9):

- The new F nonterminal serves as a record of an *unresolved* 'flip'/'flop' dependency; this record-keeping mechanism allows for applications of the separate rules that introduce 'flip' and 'flop' to be coordinated.
- The coordinated effect of the two rules involving F is to put both 'flip' and 'flop' to one side of an S in (10), and to put them on either side of an S in (11), exactly as in (8) and (9).

Another way to say this is that the F and T nonterminal symbols track the relevant **derivational state**.

This is the same notion of "state" that appears in finite-state automata (FSAs). The FSA in (12) is entirely equivalent to the CFG in (10).

(12)



From this perspective, the difference between generating serial dependencies (as in (10) and (12)) and generating nesting dependencies (as in (11)) is not a difference in the nature of derivational **state-tracking mechanisms**; the difference is in the **structure-building operations** that this derivational state controls.

3.3 The trick with crossing dependencies

CFGs generate (5e) and (6e) by combining 'tick' and 'tock' with *some other smaller expression* in which all the appropriate dependencies are already resolved.

What sets apart a crossing-dependency sentence such as (7e) is the fact that the relevant smaller expression that a 'tick'/'tock' pair needs to combine with does not correspond to a *contiguous* portion of the surface string.

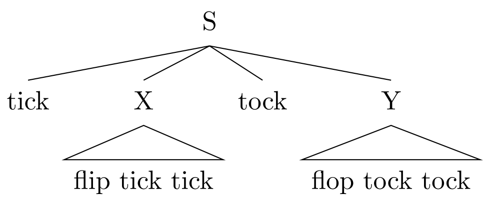
(5e) tick tock **flip flop tick tock tick tock** (serial)

(6e) tick **flip tick tick tock tock flop** tock (nesting)

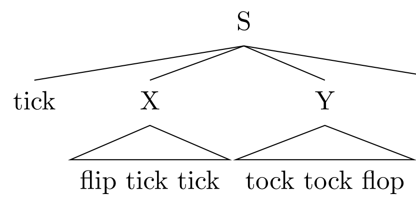
(7e) tick **flip tick tick** tock **flop tock tock** (crossing)

Notice that trying to generate crossing dependencies via a structure such as (13) will fail for the same reason that a structure like (14) fails to enforce nesting dependencies — in (9) and (11), it's crucial that the bolded portion of (6e) is "bundled up" into a single constituent.

(13)



(14)



> To generate crossing dependencies as in (7e), our grammatical mechanisms need to go beyond those of context-free grammars in some way that allows the discontiguous bolded portion 'flip tick tick . . . flop tock tock' to be somehow treated as a unit, within which all dependencies are resolved.

4 What we find in natural languages

All three of the patterns introduced above are attested in natural languages.

(15) John saw Peter let Marie swim (English, serial dependencies)

(16) . . . dass Hans Peter Marie schwimmen lassen sah
that Hans Peter Marie swim let saw
"... that Hans saw Peter let Marie swim" (German, nesting dependencies)

(17) . . . dat Jan Piet Marie zag helpen zwemmen
that Jan Piet Marie saw help swim
"... that Jan saw Piet help Marie swim" (Dutch, crossing dependencies)

The crossing-dependency pattern in (17) provides one direct argument that the simple contiguous-constituency mechanisms of CFGs are an inadequate model of natural language syntax (Huybregts, 1976, 1984; Bresnan et al., 1982; Shieber, 1985).

Chomsky (1956):

- observed that natural languages have both serial and nesting dependencies
- recognized that nesting dependencies ruled out the peripheral-constituency mechanisms of finite-state grammars[1]
- did not know about crossing dependency patterns[2]
- but made arguments of a different kind for the inadequacy of CFGs.

The essence of these other arguments is that CFGs provide no way for "selecting as elements certain discontinuous strings" (p.120), which we might think is necessary for examples like (18), (19) and (20).

(18) a. We bought **a book about linguistics** yesterday
b. We bought **a book** yesterday **about linguistics**

(19) a. We know John thinks **the girl bought the book**
b. We know **which girl** John thinks **bought the book**

(20) a. It seems **John will win**
b. **John** seems **to win**

References

Bresnan, J., Kaplan, R. M., Peters, S., and Zaenen, A. (1982). Cross-serial Dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2:137–167.

Chomsky, N. (1963). Formal properties of grammars. In Luce, R. D., Bush, R. R., and Galanter, E., editors, *Handbook of Mathematical Psychology*, volume 2, pages 323–418. John Wiley and Sons, Inc., New York and London.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.

Chomsky, N. (1995). *The Minimalist Program*. MIT Press, Cambridge, MA.

Huybregts, R. (1976). Overlapping dependencies in Dutch. In *Utrecht Working Papers in Linguistics*, number 1, pages 24–65.

Huybregts, R. (1984). The weak inadequacy of context-free phrase structure grammars. In de Haan, G. J., Trommelen, M., and Zonneveld, W., editors, *Van Periferie Naar Kern*, pages 81–99. Foris, Dordrecht.

Joshi, A. (1985). How much context-sensitivity is necessary for characterizing structural descriptions? In Dowty, D., Karttunen, L., and Zwicky, A., editors, *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, pages 206–250. Cambridge University Press, New York.

Joshi, A. K., Shanker, K. V., and Weir, D. (1990). The convergence of mildly context-sensitive grammar formalisms. University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-01.

Kobele, G. M., Retoré, C., and Salvati, S. (2007). An automata theoretic approach to minimalism. In Rogers, J. and Kepser, S., editors, *Proceedings of the Workshop: Model-theoretic syntax at 10*. Dublin.

Peters, Jr., P. S. and Ritchie, R. W. (1973). On the generative power of transformational grammars. *Information Sciences*, 6:49–83.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Stabler, E. P. (1997). Derivational minimalism. In Retoré, C., editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *LNCS*, pages 68–95, Berlin Heidelberg. Springer.

Steedman, M. (1985). Dependency and coordination in the grammar of Dutch and English. *Language*, 61:523–568.

[1] "English is literally beyond the bounds of these grammars because of mirror-image properties", p.119.

[2] "I do not know whether ... there are other actual languages that are literally beyond the bounds of phrase structure description", p.119.