Lecture 1: Sentence Parsing

Janet Dean Fodor

The Graduate Center, City University of New York

May 2021

Queen Mary University of London

Background reading

Comprehending Sentence Structure, Fodor (1995).

In An Invitation to Cognitive Science: Language,
Gleitman & Liberman (eds.).

- 8.1 From word string to sentence meaning
  - 8.1.1 Deducing structure
  - 8.1.2 Empty categories
  - 8.1.3 Ambiguity
  - 8.1.4 Anticipating empty categories
  - 8.1.5 Using linguistic information
- 8.2 Are empty categories real?
  - 8.2.1 Linguistic explanations
  - 8.2.2 Experimental evidence
- Suggestions for further reading

Two tasks for the parser

- PHRASE STRUCTURE PARSING:

  (The Sausage Machine; Frazier & Fodor 1978)

  Combining input words into a syntactic tree structure.

- PARSING TRANSFORMED SENTENCES:

  (Parsing strategies & constraints on transformations; Fodor 1978)

  Deriving a deep structure tree, which can feed semantic processing.

From a word string to its deep structure: ‘de-transformation’

- Transformational grammar is an implementation nightmare for building a parser.
- A heroic effort: Warren J. Plath (1973, CoLing)
- Method: De-transformation from the surface form.
  - First, compose a phrase structure grammar for surface sentences. That frequently overgenerates (as expected).
  - Then apply inverse versions of the grammar’s transformation rules, in inverse sequence (surface to deep structure).
- SURELY NOT PRACTICAL! T-rules don’t work in reverse.

Transformations don't work well in reverse

- A unique outcome if transform from DS to SS.
  Multiple alternatives (ambiguity!) if transform from SS to DS.
- Which book$_i$ did the teacher read gap$_i$ to the children?
  Which book$_i$ did the teacher read to the children from gap$_i$?
- Who$_i$ did you expect gap$_i$ to make a potholder?
  Who$_i$ did you$_j$ expect gap$_j$ to make a potholder for gap$_i$?
- Filler is in a fixed position. GOOD
  But more than one possible gap position. PROBLEM
- The word string is often structurally ambiguous until the very end.
- But mostly, only one analysis is eventually correct.

A more practical model for on-line parsing

- For a moment, set aside all theoretical commitment to transformational derivations. (We'll discuss below.)
- Assume one level of structure. Surface structure, with traces of 'transformational' movement or deletion.
- This is the parser's aim. While computing phrase structure on-line: Identify any potential 'fillers' and 'gaps', and pair them up while proceeding left-to-right through the sentence. One-pass.
- If, as can happen, there's more than one alternative, pick one ('serial' parsing). That may turn out to be a garden path (incompatible with later words). If so, back up and retry (just as for g-paths even without fillers & gaps).

A one-level simulation of a multi-level transformational derivation: pairing up fillers and gaps

- Input: Which book did the teacher read to the children from?

- The parser aims to build the right tree. It deduces:
  - *which book* = filler, needs gap;
  - *read* is optionally transitive;
  - *to* reveals that *read* has no following object;
  - so this is a possible position for the gap;
  - adopt it (semantically acceptable, ok);
  - keep parsing. Ok until *from* is sentence-final.
  - Oops! Must accept this new gap, so revise!

The Sausage Machine

- The Sausage Machine of Frazier & Fodor (1978) is one version of this. (There could be others.)
- It derives its silly name from another characteristic: it works on one chunk of the sentence (6 or 7 words, approx.) at a time.
- It shunts the chunks to another processing unit, which composes them into a complete sentential tree.
- So it's actually a 2-stage model. Why that?
- Because human performance shows a curious mix of intelligence and stupidity – as if it blanks out every now and then. (Every few words, in a single sentence.)

Sausage Machine explains the occasional shortsightedness of the human parser

- The human sentence parsing mechanism is prodigious. It works at great speed, and can cope with long complicated sentences when necessary. Almost always accurate!
- But some perfectly well-formed sentence constructions cause befuddlement. *

  Sue read the note, the memo and the newspaper to Jim.
  I met the boy who Jill took to the park's friend.

- It has something to do with <u>constituent length</u> (weight):
- John threw the apple that (Mary had discovered) was rotten out!   (√ out of the window and into the rosebush.)

*For befuddlement with center-embedding, see Lecture 4.

Phrasal chunking as the explanation

- This owes <u>much</u> to John Kimball.

- Kimball (1973) proposed a 2-stage parser.

  Stage 1: Package up 6 or 7 words into a (well-formed) phrasal unit.

  Stage 2: Combine those phrasal chunks into a complete well-formed sentence tree.

- We dubbed these the PPP (preliminary phrase packager) and the SSS (sentence structure supervisor).

- This model is efficient, because each stage is working with a relatively small number of units (words or chunks).

Where to make the breaks?

- The chunks created by the PPP are phrasal: an NP, or an Adv phrase, sometimes a whole clause.
- They are selected by length, not by syntactic category.
- Nevertheless, where exactly to make a break does seem to respect phrasal boundaries.
- Not: (I met the boy who Jill) (took to the park's friend.)
- Our hypothesis (now, though not yet in 1978!) is that the chunks are prosodic phrases. <See Lecture 2.>
- That's great! Not custom-created for syntactic parsing.√ With implicit (silent) prosody, it works for reading too.√

NEXT: How to parse transformed sentences?

- So far, we've not considered how the parser would/could re-constitute the tree structure for the whole sentence.
- In principle: the 6-7-word outputs of the Sausage Machine could be put back together and submitted for multi-stage de-transformation to recover a deep structure.
- But we know that could be a real headache for the parser.
- And there's no need to do it, if the transformational history can be folded into a single tree structure, with fillers and gaps co-indexed.
- This is precisely what Generalized Phrase Structure Grammar does. (GPSG; Gazdar et al.1982, and related work) See discussion below.

How to cope with non-reversibility of some T-rules?

- A reversed Passive transformation works well, whether by standard TG or by GSPG. Because its filler is associated with just one specific gap location (obj position).
- But not so for unbounded movement and deletion rules: one filler position, maybe many possible gap positions.
- Wh-movement can move a wh-phrase from any position (*pace* island constraints): subject, object, indirect object, obj of preposition, object of a subordinate clause….
- Only way for a parser to tell is: What's missing? Look for a 'gap' in the sentence, where a phrase would normally be.
- Must also include possible gaps (= absence of an optional constituent). Might turn out to be the true gap – or not.
- Often, there are several candidates along the way.

But nothing is gained by actually reversing the transformation. Just co-index filler and gap.

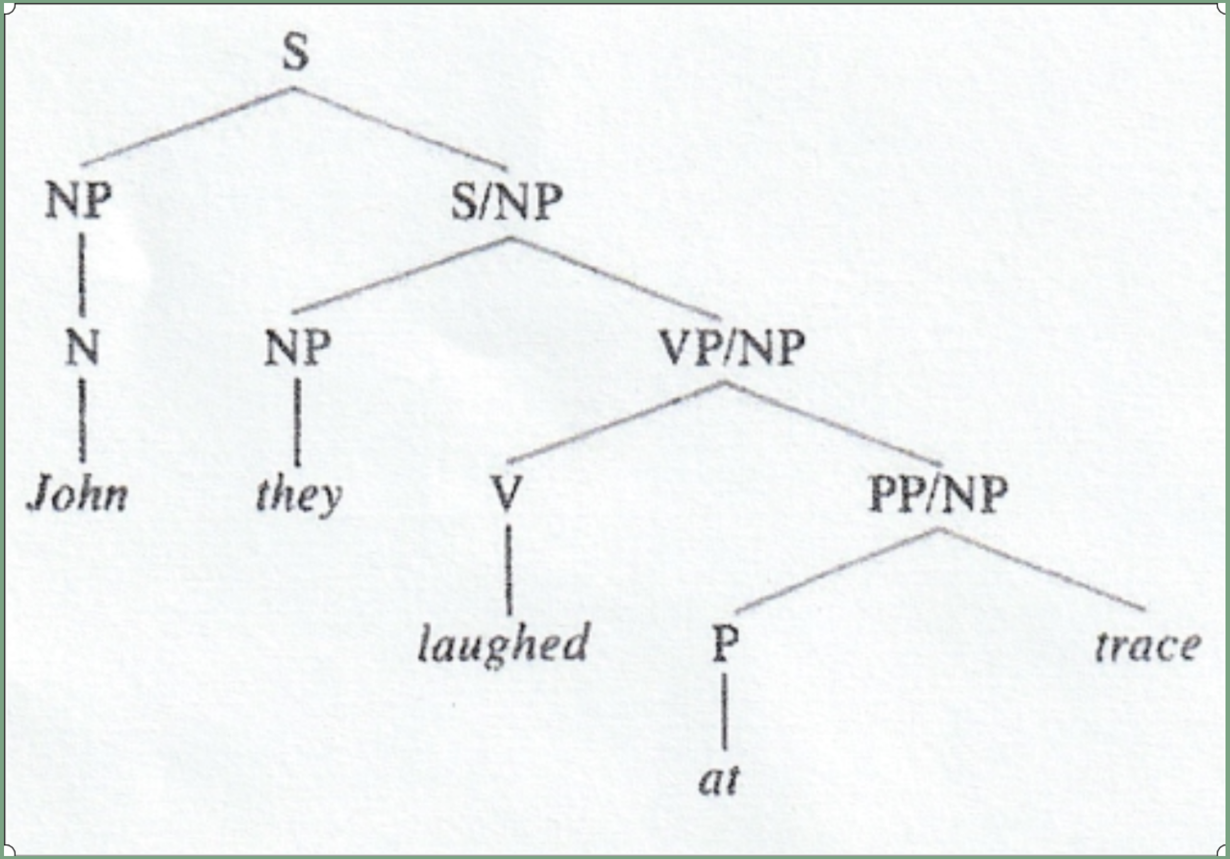
- Whether the aim is, or is not, to actually reverse the derivation to arrive at a deep structure, the parser has to do the same work. Recognize a filler! Find its gap!
- That can be laborious. (Examples below)
  But there's no way to avoid the work of doing it.
  It involves checking argument structure to detect any possibly 'missing' constituents, checking for any potential 'fillers' in legitimate positions,….
- But in that case: Why not just build a single tree structure? One pass through the word string, with any fillers tagged as such, each co-indexed with a *trace* in a legitimately related position. Best thing for the parser.

This can be implemented in GPSG

- GPSG is a single-level non-transformational theory. Now largely overlooked in linguistics, though its offspring HPSG is highly valued in computer science.
- Surface tree structures are generated with fillers and gaps in-place, and co-indexed.
- Relations between Fs and Gs are tracked by ‘slash’ features, which percolate filler-information through the nodes of the tree until an appropriate gap is found. (Examples below.)
- See Gazdar, Klein, Pullum & Sag (1985) for the theory; and Pollard & Sag (1987, 1994) for HPSG.
- Note: I’m ignoring scrambling operations here, to the extent that they go beyond the bounds of enriched PS grammars.

Simple example of a GPSG tree, with topicalization

Slash notation (/NP) indicates element 'moved' from that constituent



Filler-gap relations in a phrase structure grammar (no transformations)

- Many (perhaps all) of the familiar ‘constraints on transformations’ can be formulated simply as relations between nodes in a GPSG tree.
  - Nested Dependency Constraint
  - A-over-A constraint
  - Coordinate structure constraint
  - A range of various island constraints
- Because a filler–gap dependency runs through the tree branches, it can be sensitive to the nodes it passes through on the way.

Transformations don't work well in reverse

- A unique outcome if compute from DS to SS.
  Multiple alternatives if compute from SS to DS.
- Which book$_i$ did the teacher read EC$_i$ to the children?
  Which book$_i$ did the teacher read to the children from EC$_i$?
- Who$_i$ did you expect EC$_i$ to make a potholder?
  Who$_i$ did you$_j$ expect EC$_j$ to make a potholder for EC$_i$?
- The Wh-filler needs to find an empty NP somewhere.
  The subject of *expect* <u>may or may not</u> be an Equi filler.
- More than one <u>possible</u> gap position. The word string is often structurally ambiguous until the very end.
- But mostly, only one analysis is eventually correct.

Example: The Nested Dependency Constraint

- The NDC regulates how fillers and gaps are paired up in a sentence which has 2 fillers and 2 gaps.

  e.g. <u>What</u> are <u>boxes</u> easy to store <u>gap</u> in <u>gap</u>?

- The issue arises only if both fillers are of the same category, e.g. both are NP. If not, there's no ambiguity.

- But it can arise even if the two 'movement' rules differ, e.g. Wh-movement and Tough-movement.

- The NDC favors the <u>second filler</u> for <u>the first gap</u>.
  Answer: Boxes are easy to store in closets.
  <u>Not</u>: Pencils are easy to store in boxes.

Filler-gap relations don't need transformations

- The NDC is very simple - essentially free. It is also:
- Very useful – it eliminates ambiguity.
- Very efficient – it applies immediately, as soon as the issue arises, i.e. at the first gap position.
- Very strong – it applies even if the outcome is silly, e.g. What are warehouses easy to store in?
- And all of this can be done without any movement, or transformations at all. With an enriched context-free phrase structure grammar.

  Just: Adopt the closest filler. (Least effort on-line!)

Summary

- A single tree structure per sentence is easier for 'left-to-right' mental computation than a series of related ones.
- Therefore: As a psycholinguist, I strongly vote for some kind of enriched phrase structure grammar. I consider this case closed!
- As a linguist, I think it is elegant and explanatory. But I welcome your advice on that.

To end

- FOR FURTHER READING:

  How can grammars help parsers?
  (Crain & Fodor, in Dowty et al. 1985, sections 3.1-3.2)

- FOR FURTHER THINKING:

How can linguists help grammars help parsers?

Abstract

A sentence parsing system has two main tasks:
(i) to combine words into a syntactic tree structure, and
(ii) to deduce the deep structure, which will feed semantic interpretation. For step 2, Transformational Grammar is an implementation nightmare. Early attempts to 'de-transform' a surface string were immensely cumbersome. The major problem was that transformational rules don't work well in reverse (surface to deep), especially for a left-to-right on-line parsing system. More practical is an enriched phrase structure grammar, which delivers a single tree structure enriched with notations of 'fillers' and 'gaps'.

I propose that this is better linguistics as well as better psycholinguistics. And I add a plea to future researchers: How can linguists help grammars help parsers?