6. Tree-Adjoining Grammar (TAG)

1 Reminder: stack-based memory and “working in the middle”

We saw that an unbounded stack-based memory can allow for nesting dependencies in a string that is constructed left-to-right (i.e. categorizing only prefixes).

<table>
<thead>
<tr>
<th>Transition</th>
<th>String</th>
<th>Stack Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 0</td>
<td>ε</td>
<td>X</td>
</tr>
<tr>
<td>Step 1</td>
<td>(X,flip,FX) flip</td>
<td>FX</td>
</tr>
<tr>
<td>Step 2</td>
<td>(X,tick,TX) flip tick</td>
<td>FTX</td>
</tr>
<tr>
<td>Step 3</td>
<td>(X,flip,FX) flip tick flip</td>
<td>FTFX</td>
</tr>
<tr>
<td>Step 4</td>
<td>(X,flip,FX) flip tick flip flip</td>
<td>FTFFX</td>
</tr>
<tr>
<td>Step 5</td>
<td>(X,ε,Y) flip tick flip flip</td>
<td>FTFFY</td>
</tr>
<tr>
<td>Step 6</td>
<td>(FY,flop,Y) flip tick flip flop flop</td>
<td>FTFY</td>
</tr>
<tr>
<td>Step 7</td>
<td>(FY,flop,Y) flip tick flip flop flop flop</td>
<td>FTY</td>
</tr>
<tr>
<td>Step 8</td>
<td>(TY,tock,Y) flip tick flip flop flop tock flop tock flop tock flop tock flop tock flop</td>
<td>TY</td>
</tr>
<tr>
<td>Step 9</td>
<td>(FY,flop,Y) flip tick flip flop flop flop tock flop tock flop tock flop tock flop tock flop</td>
<td>Y</td>
</tr>
</tbody>
</table>

We can get the same effect with bounded memory if we can “work in the middle” as we construct a string (i.e. categorizing infixes).

(2) $S \rightarrow \text{flip } F$
$F \rightarrow (S) \text{flop}$
$S \rightarrow \text{tick } T$
$T \rightarrow (S) \text{tock}$

This relationship between a PDA and a CFG is analogous to the relationship between an LIG and a TAG.

<table>
<thead>
<tr>
<th>Strings</th>
<th>Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unbounded stack-based memory, working “end to end”</td>
<td>PDA (1)</td>
</tr>
<tr>
<td>Bounded memory, working “in the middle”</td>
<td>CFG (2)</td>
</tr>
</tbody>
</table>
2 TAG basics

One of the two basic tree-building operations in TAG is substitution. This does the boring stuff.

(4)

TP

VP

DP

V

DP

N

N

chased

John

The other, more distinctive operation is adjunction. As you might expect, this is used for introducing optional modifiers.

(5)

TP

VP

DP

V

DP

N

N

chased

John

PP

P

on

Tuesday

But this same adjunction operation is also used for “stretching” long-distance dependencies.

(6)

CP

DP

C

TP

V

DP

N

N

chased

John

C

TP

V

C'

think

bought

$\tau_1$

See Kroch and Joshi 1985 or Frank 2004 for broader introductions to the use of TAG as a linguistic formalism. Chapter 1 of Frank 2002 gives a fascinating account of how TAG relates to the various changes in the architectures of pre-Aspects, post-Aspects and minimalist generative theories. (Spoiler: The clausal trees that get combined via adjunction are kind of like kernel sentences!)
3 Non-context-free string languages

The TAG for $a^n b^n c^n$ is a lexicon of just two elementary trees. The NA subscripts indicate that adjoining is not allowed at those nodes.

\[(7) \quad S \quad S_{NA} \quad S_{NA} \quad S \quad S_{NA} \quad c \quad S_{NA} \quad b \quad S_{NA} \quad a \quad S \quad \varepsilon \]

A derivation using this grammar proceeds by repeatedly adjoining the tree with the three terminal nodes into the middle S node.

\[(8) \quad S \quad \varepsilon \quad S_{NA} \quad S \quad S_{NA} \quad b \quad a \quad \text{adjoin} \quad S \quad S_{NA} \quad S \quad S_{NA} \quad b \quad b \quad S_{NA} \quad c \quad \text{adjoin} \quad S \quad S_{NA} \quad S \quad S_{NA} \quad b \quad b \quad b \quad S_{NA} \quad c \quad \varepsilon \]

Kroch and Santorini (1991, pp.310–312) use this kind of “vertical nesting” derivation to generate the Dutch crossing-dependency construction.

\[(9) \quad \text{Initial tree} \quad \text{Auxiliary trees} \]

\[\text{I:} \quad S \quad S^* \quad V_3 \quad \text{NP} \quad \text{VP} \quad \text{zwemmen} \quad \text{NP} \quad \text{VP} \quad \text{larten} \quad \text{NP} \quad \text{VP} \quad \text{S} \quad \text{V}_2 \quad \text{N} \quad \text{S} \quad \text{V}_1 \quad \text{Jan} \quad \text{zag} \]

\[\text{A:} \quad S \quad S^* \quad V_3 \quad \text{NP} \quad \text{VP} \quad \text{Piet} \quad \text{NP} \quad \text{VP} \quad \text{e} \quad \text{NP} \quad \text{S} \quad \text{NP} \quad \text{S} \quad \text{V}_1 \quad \text{Marie} \quad \text{e} \quad \text{S} \]

\[\text{B:} \quad S \quad \text{NP} \quad \text{VP} \quad \text{Marie} \quad \text{NP} \quad \text{S} \quad \text{NP} \quad \text{S} \quad \text{V}_1 \quad \text{Piet} \quad \text{NP} \quad \text{S} \quad \text{NP} \quad \text{S} \quad \text{V}_1 \quad \text{Jan} \quad \text{zag} \]
4 Non-finite-state tree languages

Are those the “right” tree structures for the Dutch crossing dependencies?

This is a somewhat thorny question. Many others have instead analyzed these sentences using a “double right-branching” structure, following Bresnan et al. (1982).

This kind of structure cannot be generated by a TAG (Joshi, 1985, pp.245–249) — one easy way to see why is to think about it in terms of an LIG.

Are there analogous constructions where we might be more confident that the TAG-style structure is what we want?
Perhaps examples like the following, in languages like Bulgarian where embedded questions are not islands? As long as the pattern extends unboundedly? (Frank and Hunter, 2021)

(12) Koja kniga te popita učitelja kogo2 [ubedi Ivan t2 da publikiva t1]
    which book you asked teacher who convinced Ivan to publish
    “Which book did the teacher ask you who Ivan convinced to publish?”

(13) Koj kontinent te popita učitelja koj2 [t2 e otkril t1]?
    which continent you asked teacher who has discovered
    “Which continent did the teacher ask you who discovered?”

(14) . . .

(15) . . .

References


