Ling 5801: Lecture Notes 7
From Programs to Context-Free Grammars

Contents

7.1 The rules we used to define programs make up a context-free grammar

A Context-Free Grammar is a tuple $\langle C, X, S, R \rangle$, where:

- C is a finite set of type or 'category' symbols
- $X \subseteq C$ is a set of terminal symbols (e.g. `if`, `:`, ...)
- $S \subseteq C$ is a set of start symbols (e.g. $\langle$program$\rangle$)
- $R \subseteq C \times C \times C$ is a set of rewrite rules of the form $c \rightarrow c' \; c''$ (or $\langle c, c', c'' \rangle$)

The symbol to the left of each arrow may be called the 'parent'

The symbols to the right of each arrow may be called the 'children'

Grammars allowing only two children per rule are called 'binary-branching'

For example:

$$\begin{aligned} G_{\text{cattoy}} = \langle\, & \{\textbf{S}, \textbf{NP}, \textbf{VP}, \textbf{the}, \textbf{cat}, \textbf{hits}, \textbf{toy}\}, \\ & \{\textbf{the}, \textbf{cat}, \textbf{hits}, \textbf{toy}\}, \\ & \{\textbf{S}\}, \\ & \{\textbf{S} \rightarrow \textbf{NP VP},\ \textbf{NP} \rightarrow \textbf{the cat},\ \textbf{NP} \rightarrow \textbf{the toy},\ \textbf{VP} \rightarrow \textbf{hits NP}\} \rangle \end{aligned}$$

7.2 Languages accepted by a CFG

For any CFG $G = \langle C, X, S, R \rangle$:

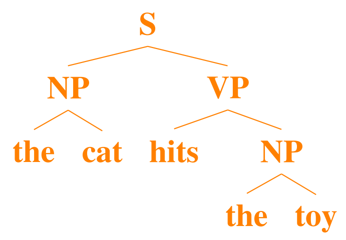
$$L(G) = \{x_1..x_T \mid \exists_{c \in S}\; c \overset{*}{\underset{G}{\rightarrow}} x_1..x_T\}$$

where c yields $x_i..x_j$ if and only if there is a tree τ w. root c, all leaves in $x_i..x_j$, branches in R

(defining trees τ as sets of constituents $\langle c, i, j\rangle$ of category c spanning input i to j):

$$c \xrightarrow[G]{*} x_i..x_j \text{ iff } \bigvee_{\tau \text{ w. root } \langle c,i,j\rangle} \ \bigwedge_{\langle c',i',j'\rangle\in\tau} \begin{cases} \text{if } i'=j' : \begin{cases}\text{if } c'=x_{i'} : \text{True}\\ \text{if } c'\neq x_{i'} : \text{False}\end{cases} \\ \text{if } i'<j' : \bigvee_{k',d',e'} R(c'\to d'\ e') \wedge \langle d',i',k'\rangle \in\tau \wedge \langle e',k'+1,j'\rangle\in\tau \end{cases}$$

For example, ‘the cat hits the toy’ is in $L(G_{\text{cattoy}})$ because this tree τ exists:



with constituents:

$$\tau = \{\langle \textbf{S},1,5\rangle, \langle \textbf{NP},1,2\rangle, \langle \textbf{the},1,1\rangle, \langle \textbf{cat},2,2\rangle, \langle \textbf{VP},3,5\rangle, \langle \textbf{hits},3,3\rangle, \langle \textbf{NP},4,5\rangle, \langle \textbf{the},4,4\rangle, \langle \textbf{toy},5,5\rangle\}$$

Now move the shared top branch conjunct outside the disjunction over trees (distributive axiom):

$$c \xrightarrow[G]{*} x_i..x_j \text{ iff } \begin{cases} \text{if } i=j : \begin{cases}\text{if } c=x_i : \text{True}\\ \text{if } c\neq x_i : \text{False}\end{cases} \\ \text{if } i<j : \bigvee_{k,d,e} R(c\to d\ e) \wedge \left(\bigvee_{\tau' \text{ w. root } \langle d,i,k\rangle} \ \bigwedge_{\langle c',i',j'\rangle\in\tau'} \{\dots\right) \wedge \left(\bigvee_{\tau'' \text{ w. root } \langle e,k+1,j\rangle} \ \bigwedge_{\langle c'',i'',j''\rangle\in\tau''} \{\dots\right) \end{cases}$$

and identify parenthesized terms as recursive instances of $c \xrightarrow[G]{*} x_i..x_j$:

$$c \xrightarrow[G]{*} x_i..x_j \text{ iff } \begin{cases} \text{if } i=j : \begin{cases}\text{if } c=x_i : \text{True}\\ \text{if } c\neq x_i : \text{False}\end{cases} \\ \text{if } i<j : \bigvee_{k,d,e} R(c\to d\ e) \wedge \left(d \xrightarrow[G]{*} x_i..x_k\right) \wedge \left(e \xrightarrow[G]{*} x_{k+1}..x_j\right) \end{cases}$$

More simply:

$$c \xrightarrow[G]{*} x_i..x_j \text{ iff } \begin{cases} \text{if } i=j : c = x_i \\ \text{if } i<j : \exists_{k,d,e}\ c\to d\ e \in R \wedge d \xrightarrow[G]{*} x_i..x_k \wedge e \xrightarrow[G]{*} x_{k+1}..x_j \end{cases}$$

And the languages recognizable by a CFG are:

$$\mathcal{L}(\text{CFG}) = \{L(G) \mid \text{CFG } G\}$$

7.3 CFGs don’t gain power from extra children

$\mathcal{L}(\text{CFG}) \subseteq \mathcal{L}(\text{CFG}_{\leq 2 \text{ children per branch}})$, so $\mathcal{L}(\text{CFG}) = \mathcal{L}(\text{CFG}_{\leq 2 \text{ children per branch}})$

CFGs which have rules R containing more than two children can be reduced to binary R':

$$R' = \{c \to c_0\ c_1\_..\_c_B \mid c \to c_0 \dots c_B \in R\}$$

$$\cup \{c_b\_..\_c_B \to c_b\ c_{b+1}\_..\_c_B \mid c \to c_0 \ldots c_B \in R,\ 1 \le b < B\}$$

$$C' = C \cup \{c_b\_..\_c_B \mid c \to c_0 \ldots c_B \in R,\ 1 \le b < B\}$$

For example:

VP $\to$ pick NP up,
. . .

would become:

VP $\to$ pick NP_up,
NP_up $\to$ NP up,
. . .

7.4 CFGs don't gain power from unary children

$\mathcal{L}(\text{CFG}_{\le 2 \text{ children per branch}}) \subseteq \mathcal{L}(\text{CFG}_{2 \text{ children per branch}})$, so $\mathcal{L}(\text{CFG}) = \mathcal{L}(\text{CFG}_{2 \text{ children per branch}})$.

CFGs which have rules R containing fewer than two children can be reduced to binary R':

1. First, obtain reflexive transitive closure $R^{(|C|)}$ of unary rules in R (like ϵ-transitions in FSA):

   $R^{(0)} = \{c \to c \mid c \in C\}$

   $R^{(k)} = R^{(k-1)} \cup \{c \to c'' \mid c \to c' \in R^{(k-1)},\ c' \to c'' \in R\}$

2. Then add these closure expansions to children of each binary expansion:

   $R' = \{c \to d'\ e' \mid c \to d\ e \in R,\ d \to d' \in R^{(|C|)},\ e \to e' \in R^{(|C|)}\}$

3. Then add these closure expansions to the set of start symbols:

   $S' = S \cup \{c' \mid c \in S,\ c \to c' \in R^{(|C|)}\}$

For example, adding the following unary rule to our cat-toy grammar:

NP $\to$ Meowy

would result in the addition of the following binary rules:

S $\to$ Meowy VP,
VP $\to$ hits Meowy

7.5 A weak but intuitive claim: $\mathcal{L}(\text{RE}) \subseteq \mathcal{L}(\text{CFG})$

We can write a CFG $G(\rho)$ implementing any RE ρ, concatenated to a start symbol.

First, remove epsilons from Kleene star:

$\rho^* = (\epsilon|\rho^+)$
$(\epsilon|\rho_1)\rho_2 = (\rho_2|\rho_1\rho_2)$
$\rho_1(\epsilon|\rho_2) = (\rho_1|\rho_1\rho_2)$

Then map regular expression syntax to CFG rules:

$$\begin{aligned}
R_{G(\rho_1\rho_2)} &= R_{G(\rho_1)} \cup R_{G(\rho_2)} \cup \{c_{\rho_1\rho_2} \rightarrow c_{\rho_1}\ c_{\rho_2}\} \\
R_{G(\rho_1|\rho_2)} &= R_{G(\rho_1)} \cup R_{G(\rho_2)} \cup \{c_{\rho_1|\rho_2} \rightarrow c_{\rho_1},\ c_{\rho_1|\rho_2} \rightarrow c_{\rho_2}\} \\
R_{G(\rho_1^+)} &= R_{G(\rho_1)} \cup \{c_{\rho_1^+} \rightarrow c_{\rho_1}\ c_{\rho_1^+},\ c_{\rho_1^+} \rightarrow c_{\rho_1}\}
\end{aligned}$$

7.6 A strong claim: $\mathcal{L}(\mathbf{RE}) \subset \mathcal{L}(\mathbf{CFG})$

There is a language that cannot be recognized by a RE: $a^n b^n$

(proof by pumping lemma)

...which can be recognized by a CFG:

(proof by existence: $\langle \{\mathbf{S}, \mathbf{a}, \mathbf{b}\}, \{\mathbf{a}, \mathbf{b}\}, \{\mathbf{S}\}, \{\mathbf{S} \rightarrow \mathbf{a}\ \mathbf{S}\ \mathbf{b},\ \mathbf{S} \rightarrow \mathbf{a}\ \mathbf{b}\} \rangle$)

7.7 We can use CFGs to model natural languages like English

Do natural languages have natural category types that we can use in CFG rules? Yes!

First, observe that natural languages use different argument structures for different verbs:

- They sleep. – one argument ahead (intransitive)
- They find pets. – one argument ahead and one argument behind (transitive)
- They give people pets. – one argument ahead, two arguments behind (ditransitive)

Next, observe natural languages *coordinate* conjunctions (combine like types): $\langle\alpha\rangle \rightarrow \langle\alpha\rangle$ and $\langle\alpha\rangle$.

- [$_\beta$ [$_\beta$ They find people] and [$_\beta$ they find pets]]. – sounds ok (β is sentence)
- They find [$_\gamma$ [$_\gamma$ people] and [$_\gamma$ pets]]. – sounds ok (γ is noun phrase)
- *They find [$_\gamma$ [$_\beta$ they find people] and [$_\gamma$ pets]]. – sounds wrong; conjuncts must match

Now, allowable conjunctions give us insight into the category structure of language:

- They [$_\delta$ [$_\delta$ sleep] and [$_\delta$ find pets]]. – sounds ok (δ is verb phrase)
- They [$_\eta$ [$_\eta$ find] and [$_\eta$ give people]] pets. – sounds ok (but what's η?)

Transitive verbs (find) match type with ditransitive verb + indirect object (give people)!

Both lack argument ahead and behind – it seems types are defined by missing arguments!

Formalize set of categories C as follows – clauses with various unmet requirements:

1. every U is in C, for some set U of primitive categories;
2. every $C \times O \times C$ is in C, for some set O of type-combining operators;
3. nothing else is in C

Define primitive categories $U = \{\mathbf{N}, \mathbf{V}\}$:

- **N**: noun-headed category with no missing arguments (noun phrase)
- **V**: verb-headed category with no missing arguments (sentence)

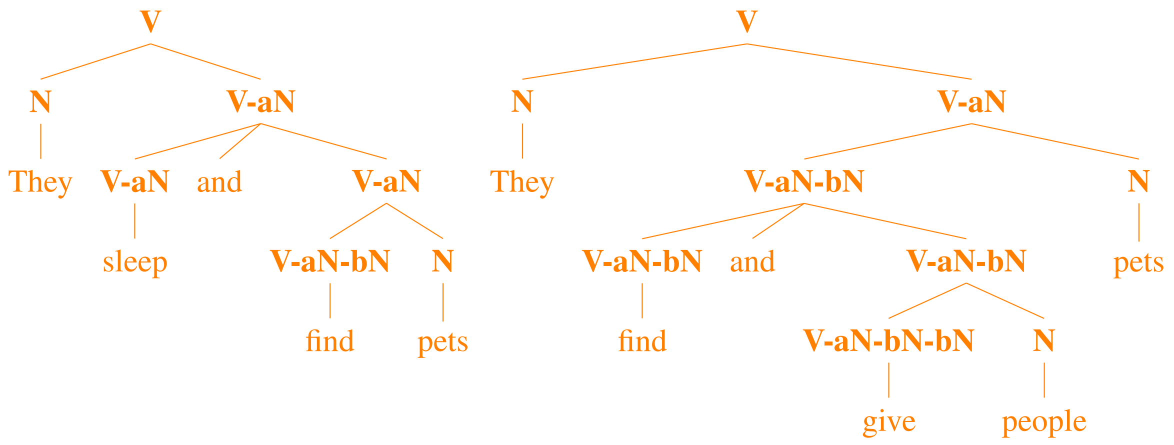
Define type-combining operators $O = \{\textbf{-a}, \textbf{-b}\}$:

- $\langle\alpha\textbf{-a}\beta\rangle$: α lacking β argument ahead (e.g. **V-aN** for intransitive δ above)
- $\langle\alpha\textbf{-b}\beta\rangle$: α lacking β argument behind (e.g. **V-aN-bN** for transitive η above)

Now we can define CFG rules R over these categories:

- $\langle\alpha\rangle \rightarrow \langle\beta\rangle\ \langle\alpha\textbf{-a}\beta\rangle$: argument attachment ahead
- $\langle\alpha\rangle \rightarrow \langle\alpha\textbf{-b}\beta\rangle\ \langle\beta\rangle$: argument attachment behind
- $\langle\alpha\rangle \rightarrow \langle\alpha\rangle$ and $\langle\alpha\rangle$: conjunction

These three rules model all of the above sentences:



Also note that the parents in these rules all have simpler types than the children.

This means for any lexicon (constraining types at tree leaves), the set of categories C is finite.

7.8 Limits of CFGs

Natural languages may also use non-local dependencies.

In English, these show up in topicalization, which seem to use a gap '_' at one argument:

- These pets, you say they found _.

These coordinate as well, but our test shows categories with gaps differ from those without:

- These pets, you [${}_\delta$ [${}_\delta$ say they found _] and [${}_\delta$ think _ gave people joy]]. – sounds ok
- *These pets, you [${}_\delta$ [${}_{\textbf{V-aN}}$ say they found pets] and [${}_\delta$ think _ gave people joy]]. – wrong

We can model this by adding a new type-combining operator for non-local dependencies:

- $\langle\alpha\textbf{-g}\beta\rangle$: α lacking non-local β argument (e.g. **V-aN-gN** for intransitive δ above) and adding rules to introduce non-local dependencies:

- $\langle\alpha\textbf{-g}\beta\rangle \rightarrow \langle\alpha\textbf{-a}\beta\rangle$: introduce non-local dependency to argument ahead
- $\langle\alpha\textbf{-g}\beta\rangle \rightarrow \langle\alpha\textbf{-b}\beta\rangle$: introduce non-local dependency to argument behind

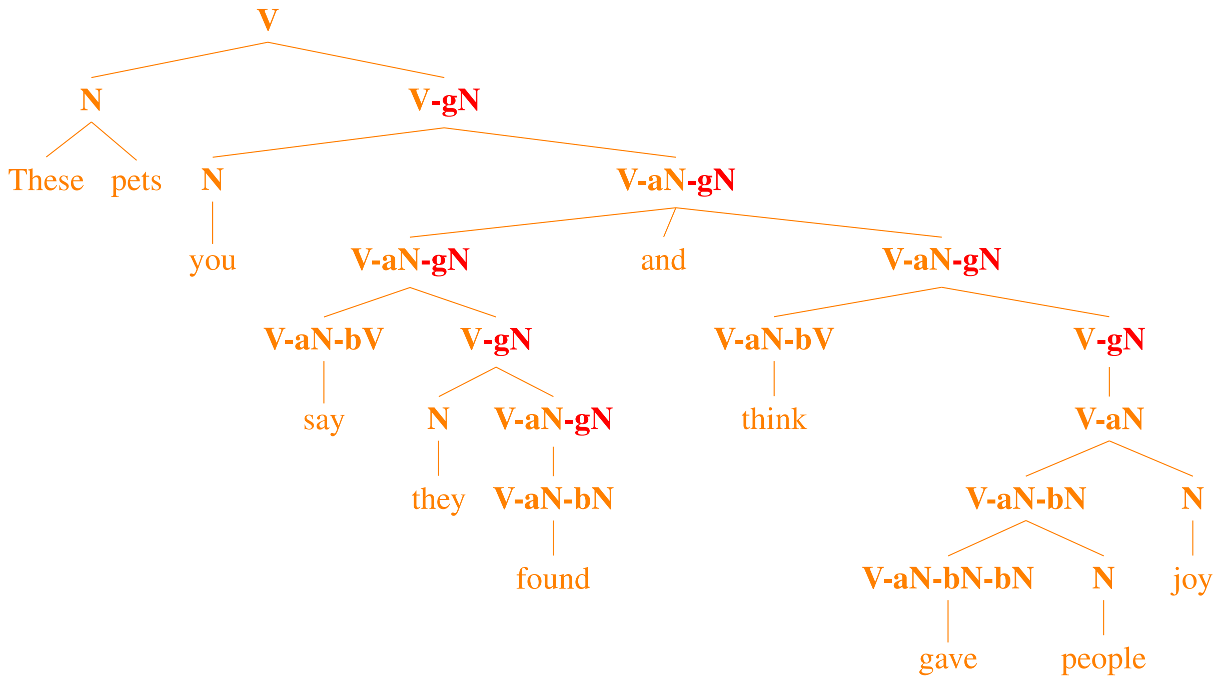
and adding rules to attach non-local dependencies:

- $\langle\alpha\rangle \rightarrow \langle\beta\rangle\ \langle\alpha\textbf{-g}\beta\rangle$: non-local dependency attachment

and modifying existing rules to propagate non-local dependencies $\psi_m \in \{\textbf{-g}\} \times C$:

- $\langle\alpha\psi_{1..M}\rangle \rightarrow \langle\beta\psi_{1..m}\rangle\ \langle\alpha\textbf{-a}\beta\psi_{m+1..M}\rangle$: argument attachment ahead, with propagation
- $\langle\alpha\psi_{1..M}\rangle \rightarrow \langle\alpha\textbf{-b}\beta\psi_{1..m}\rangle\ \langle\beta\psi_{m+1..M}\rangle$: argument attachment behind, with propagation

Here's the analysis:



Note that M above is unbounded, so our rules no longer guarantee a finite set of categories.

(Any number of arguments may be extracted and propagated up from children.)

Some use evidence like this to argue language isn't context-free but mildly context-sensitive [Shieber, 1985, Joshi, 1985, Steedman, 2000].

In practice, though, we can just constrain category sets to combinations seen in training data.

7.9 Natural language grammar: $G_{\textbf{English}} = \langle C, X, S, R\rangle$ (generatively sound, not complete)

1. Categories C defined from set of primitives U and set of type-constructing operators O:
   (a) Primitive categories U — different ways to say the same thing:

V	finite / declarative sentence	(e.g. 'they knew it')
I	infinitive clause	(e.g. 'them to know it', as in 'we expect . . . ')
B	base-form clause	(e.g. 'them know it', as in 'we let . . . ')
L	participle clause	(e.g. 'them known it' – complete form not used)
A	adjectival/predicative/'small' clause	(e.g. 'them knowing it', 'it known', as in 'we consider . . . ')
R	adverbial clause	(e.g. 'them knowingly' – complete form not used)
G	gerund clause	(e.g. 'them knowing it')
N	accus. noun phrase / nominal clause	(e.g. 'their knowledge of it', 'her', 'him', 'Kim')
N1S	nom. 1st-pers. sing. noun phrase	(e.g. 'I')
N1P	nom. 1st-pers. plur. noun phrase	(e.g. 'we')
N2	nom. 2nd-pers. noun phrase	(e.g. 'you')
N3S	nom. 3rd-pers. sing. noun phrase	(e.g. 'their knowledge of it', 'she', 'he', 'Kim')
N3P	nom. 3rd-pers. plur. noun phrase	(e.g. 'they', 'the risks')
D	determiner (clause)	(e.g. 'their knowledge of it's', 'the', 'their')
O	non-possessive genitive (clause)	(e.g. 'of their knowledge of it')
C	complementized clause	(e.g. 'that they know it', as in 'we think . . . ')
E	embedded infinitive clause	(e.g. 'for them to know it', as in 'we hope . . . ')
Q	(polar) question	(e.g. 'did they know it')
S	start symbol / sentence / utterance	(e.g. 'know it', 'yeah', 'duh')

(b) Type-constructing operators O — these allow specification of unmet requirements:

-a	lacking initial argument	(precedes lexical head / main word, e.g. subject)
-b	lacking final argument	(follows lexical head / main word, e.g. object)
-c	lacking initial conjunct	(precedes conjunction)
-d	lacking final conjunct	(follows conjunction)
-g	lacking gap argument	(missing at any position in clause)
-h	lacking right-node-raised argument	(right-node raising)
-i	lacking interrogative pronoun referent	
-r	lacking relative pronoun referent	

(c) Define categories C as follows — clauses with various unmet requirements:

i. every U is in C

ii. every $C \times O \times C$ is in C

iii. nothing else is in C

English needs categories for interrog. and relative pronouns, clauses, and gapped clauses:

V-gN	gapped finite clause / bare relative	(e.g. 'Kim gave ϵ to Chris')
B-gN	gapped base-form clause	(e.g. 'Kim give ϵ to Chris')
N-iN	interrogative pronoun	(e.g. 'what')
Q-iN	content question	(e.g. 'what did Kim give ϵ to Chris')
V-iN	embedded question	(e.g. 'what Kim gave ϵ to Chris')
N-rN	relative pronoun	(e.g. 'which')
V-rN	relative clause	(e.g. 'which Kim gave ϵ to Chris')
. . .	(and others)	

and categories for constituents within clauses:

N-aD, **N3S-aD**, **N3P-aD**	common noun, lacking determiner	(e.g. 'risk')
V-aN3S	intransitive verb, lacking subject	(e.g. 'sleeps')
V-aN3S-bN	transitive verb, lacking subject, object	(e.g. 'finds')
V-aN3S-bN-bN	ditransitive verb	(e.g. 'gives')
V-aN3S-bC	bridge verb	(e.g. 'thinks')
V-aN3S-b(I-aN)	raising verb	(e.g. 'seems')
V-aN3S-b(B-aN)	auxiliary verb	(e.g. 'would')
V-aN3S-bE		(e.g. 'hopes')
V-aN3S-b(Q-iN)		(e.g. 'wonders')
A-aN	adjective / predicative form of verb	(e.g. 'asleep', 'knowing', 'known')
A-aN-bN	adjectival preposition	(e.g. 'in', 'finding')
A-aN-b(I-aN-gN)	tough adjective	(e.g. 'tough')
R-aN	adverb	(e.g. 'knowingly')
R-aN-bN	adverbial preposition	(e.g. 'into')
R-aN-bV	discourse connective	(e.g. 'after')
...	(and others)	

2. Observation symbols: $X = \{\textbf{the}, \textbf{a}, \textbf{and}, \textbf{car}, \dots\}$

3. Start symbols: $S = \{\textbf{S}\}$

4. Rules in R:

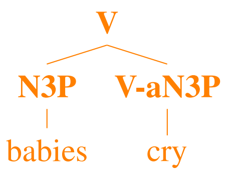
   (a) lexical items: ... (e.g. $\textbf{D} \rightarrow \textbf{the}$)

   (b) arguments:

   for $c \in U \times (\{\textbf{-a}, \textbf{-b}, \textbf{-c}, \textbf{-d}\} \times C)^*$, $d \in C$:

   $c \rightarrow d\ c\textbf{-a}d, \quad c \rightarrow c\textbf{-b}d\ d$

   e.g.:

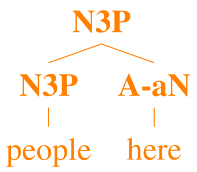


   (c) modifiers:

   for $c \in U \times (\{\textbf{-a}, \textbf{-b}, \textbf{-c}, \textbf{-d}\} \times C)^*$, $u \in U$:

   $c \rightarrow c\ u\textbf{-aN}, \quad c \rightarrow u\textbf{-aN}\ c$

   e.g.:



   (d) hypothesize gaps:

   for $c \in C$, $d \in C$:

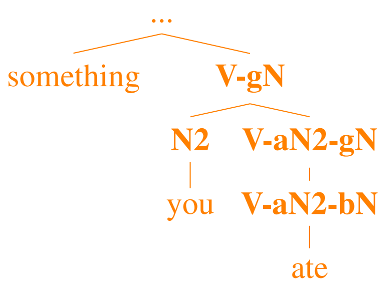
   $c\textbf{-g}d \rightarrow c\textbf{-a}d \quad c\textbf{-g}d \rightarrow c\textbf{-b}d \quad c\textbf{-g}d \rightarrow c$

(e) propagate non-local arguments:

for $a \rightarrow b\ c \in R,\ \psi \in \{\textbf{-g}, \textbf{-h}, \textbf{-i}, \textbf{-r}\} \times C$:

$a\psi \rightarrow b\ c\psi,\quad a\psi \rightarrow b\psi\ c,\quad a\psi \rightarrow b\psi\ c\psi$

e.g.:



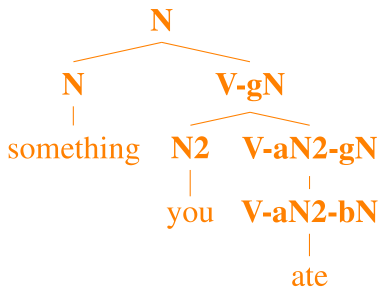
(f) connect gaps to modificands or fillers:

for $c \in C,\ d \in C,\ e \in C$:

$e \rightarrow e\ c\textbf{-r}d,\quad c\textbf{-r}e \rightarrow d\textbf{-r}e\ c\textbf{-g}d$

$e \rightarrow e\ c\textbf{-g}d,\quad c\textbf{-i}e \rightarrow d\textbf{-i}e\ c\textbf{-g}d$

e.g.:

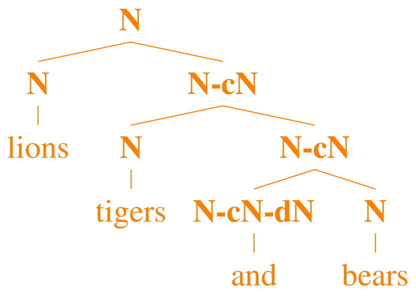


(g) conjuncts:

for $c \in C,\ d \in C$:

$c \rightarrow d\ c\textbf{-c}d,\quad c\textbf{-c}d \rightarrow d\ c\textbf{-c}d,\quad c \rightarrow c\textbf{-d}d\ d$

e.g.:

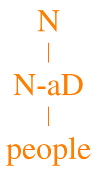


(h) elided arguments (e.g. determiners):

for $c \in C,\ d \in C$:

$c \rightarrow c\textbf{-a}d \quad c \rightarrow c\textbf{-b}d$

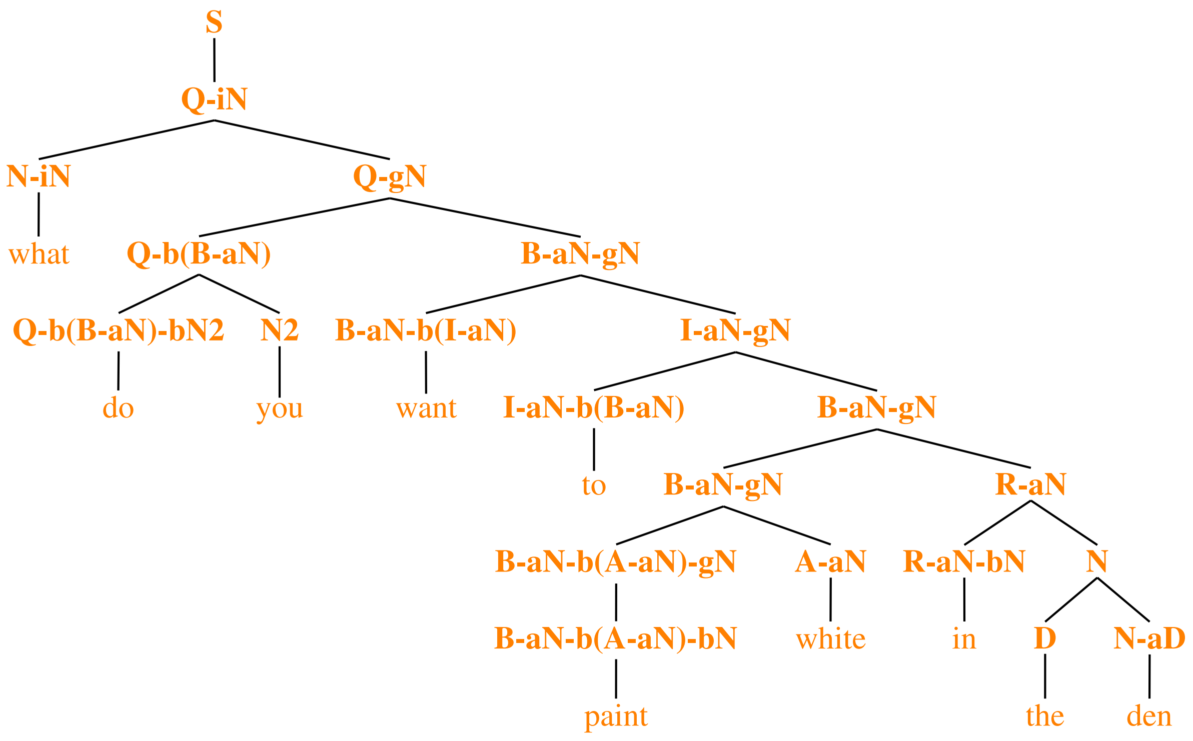
e.g.:



(i) sentential forms:

S→V	declarative sentence	(e.g. ‘Kim slept’)
S→B-aN	imperative sentence	(e.g. ‘sleep!’)
S→Q	polar question	(e.g. ‘did Kim sleep?’)
S→Q-iN	content question	(e.g. ‘what did Kim drive?’)
S→N V-gN	topicalized sentence	(e.g. ‘that park I like’)
S→S V-gS	topicalized sentence	(e.g. ‘I ate she said’)

For example:

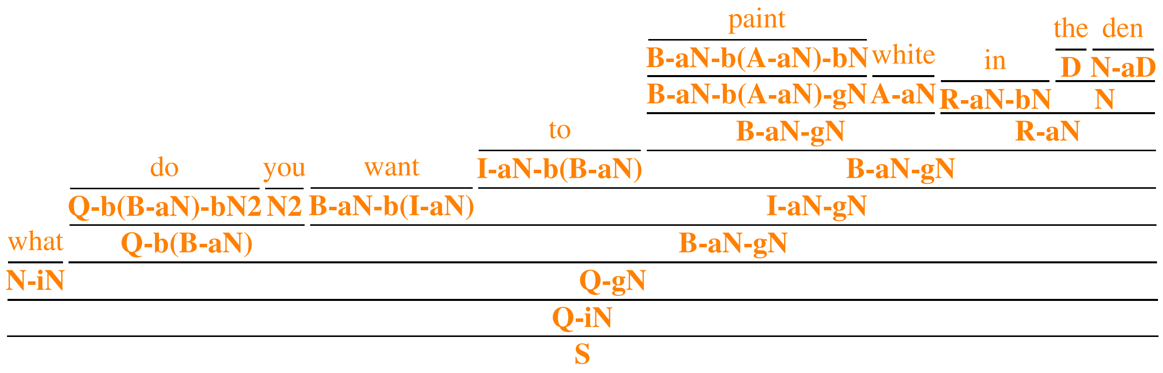


7.10 Parsing as deduction

Can also express rules as inferences in natural deduction:

$$c \rightarrow d\ e \qquad \Leftrightarrow \qquad \frac{d\ \ e}{c}$$

Parse trees are then proofs:



References

[Joshi, 1985] Joshi, A. K. (1985). How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. In D. Dowty, L. K. and Zwicky, A., editors, *Natural language parsing: Psychological, computational and theoretical perspectives*, pages 206–250. Cambridge University Press, Cambridge, U.K.

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

[Steedman, 2000] Steedman, M. (2000). *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.