Part I

Worksheet 1

1 Natural Languages

1. The following natural language sentences are ambiguous. Describe the ambiguities.

   (a) She fed her cat food

   This has syntactic ambiguity – two possible parses are: "(She fed) (her cat) food" and "(she fed) her (cat food)".

   In the first parse, "she" is feeding the "cat that she owns" some food.

   In the second parse, "she" is feeding an unspecified female "her" some "cat food".

   (b) She saw the man with one eye

   This has syntactic ambiguity – two possible parses are: "(She saw) (the man (with one eye))" and "(She saw (the man)) (with one eye)".

   In the first parse, "she" sees "a man who only has one eye".

   In the second parse, "she" sees a man; however she only sees him with one of her eyes.

   (c) She saw the queen in the garden with the telescope.

   This has syntactic ambiguity – two possible parses are: "((she saw) (the queen in the garden)) (with the telescope)" and "(she saw) (the queen (in the garden (with the telescope)))".

   In the first parse, "she" sees "the queen" using "the telescope".

   In the first parse, "she" sees the "queen who has the telescope".

2. The following natural language sentences are difficult to process. Hypothesise what is causing the difficulty.

   (a) I told the girl the rabbit knew the caterpillar would help her

   The difficulty to parse this sentence is most easily explained by Hale's model of human parsing complexity – which uses a Probabilistic Earley Parser to determine the probability of certain parses. Sentences (and words) with a high surprisal (low probability) are the hardest for humans to parse.

   This complexity arises because of the unusual context for the rabbit and the caterpillar: rabbits don't know things and caterpillars don't help people.

   An example of a sentence which has the exact same structure but is far easier to parse is "I told the girl her parents know the teacher would help her".

   (b) The twins the rabbit the girl chased liked laughed

   This can be best explained by Yngve's usage of Pushdown Automata as an approximation of parsing complexity – which states that the size of the stack required to parse a sentence is proportional to the human parsing complexity of the sentence.

   This sentence exhibits centre-embedded structures and as a result requires a large stack space to parse.

   An example of a phrase which has the exact same meaning but is far easier to parse is "The twins laughed. They were the twins the rabbit liked. The rabbit was the one chased by the girl."

(c) She shook the bottle containing the potion which had made her grow very tall up.

This is an example of a garden path sentence: "a sentence which has a prefix with a far more likely parse". The difficulty we have parsing this can be explained the easiest by Hale's probabilistic Earley Parser. The final word in the sentence has a very high surprisal and as such the probability of the entire sentence is moderately low.

2 Formal Languages

1. If $\mathcal{L}_1$ and $\mathcal{L}_2$ are regular languages prove the following are also regular:

   (a) $\mathcal{L}_1 \cup \mathcal{L}_2$

   The union of two languages of the same type in the Chomsky Hierarchy is another language of the same type. Therefore, the union of two regular languages is also a regular language.

   (b) $\mathcal{L}_1\mathcal{L}_2$

   If $\mathcal{L}_1$, $\mathcal{L}_2$ are both regular then (using Kleene's Theorem), there exist a regular expressions $\mathcal{R}_1$, $\mathcal{R}_2$ which recognise them. The regular expression $\mathcal{R}_1\mathcal{R}_2$ matches the langauge $\mathcal{L}_1\mathcal{L}_2$. Due to Kleene's Theorem, this proves that $\mathcal{L}_1\mathcal{L}_2$ is a regular langauge.

   (c) $\mathcal{L}_1 \cap \mathcal{L}_2$

   The intersection of a language of type n in the Chomksy Hierarchy with a regular language is a language of type n. So the intersection of a type 3 (regular) language with a regular langauge; is a regular language.

2. If $\mathcal{L}_1$ is regular and $\mathcal{L}_2$ is context free prove the following is also context free.

   (a) $\mathcal{L}_1 \cap \mathcal{L}_2$

   Without loss of generality, let there be a DFA $M = (\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{F})$ which recognises$\mathcal{L}_1$, and a nondeterministic PDA $M'(\mathcal{Q}', \Sigma, \Gamma, \delta', q_0', \mathcal{Z}, \mathcal{F}')$ which recognises $\mathcal{L}_2$. Notice the requirement for both machines to have the same alphabet.

   Now define the pushdown automata M'':

   $$M'' = (\mathcal{Q} \times \mathcal{Q}', \Sigma, \Gamma, \delta'', (q_0, q_0'), \mathcal{Z}, \mathcal{F} \times \mathcal{F}')$$
   $$\delta'' = \{((q_1, q_1'), \sigma, \gamma_1) \to ((q_2, q_2'), \gamma_2) \mid (q_1, \sigma) \to q_2 \in \delta, (q_1', \sigma, \gamma_1) \to (q_2', \gamma_2) \in \delta'\}$$

   M'' reaches an accepting state if and only if both M and M' reach accepting states. Therefore, there exists a pushdown automata which recognises $\mathcal{L}_1 \cap \mathcal{L}_2$. So $\mathcal{L}_1 \cap \mathcal{L}_2$ is a context free langauge.

3 Pumping Lemma for Regular and Context Free Languages

1. Use the pumping lemma for regular languages to prove the following are not regular:

   (a) $\mathcal{L} = \{ab^ncd^ne|n \geq 1\}$

   The pumping lemma for regular langauges states that for all lanaguges $\mathcal{L}$, there exists some ℓ such that for all strings $s \in \mathcal{L}$, such that $|s| \geq \ell$, s can be pumped.

For a regular langauge $\mathcal{L}$, string s can be pumped if and only if it is of the form uxv for some $|ux| \leq \ell$, $|x| \geq 1$ and for all $n \in \mathbb{N}$, $ux^nv \in \mathcal{L}$.

We can use this to derive a contradiction and prove that strings are not regular.

Assume for contradiction that $\mathcal{L} = \{ab^ncd^ne | n \geq 1\}$ is regular. Consider the string $s = ab^\ell cd^\ell e$. By the pumping lemma, s can must be pumpable. ux must be a prefix of ab^ℓ. So either $x = ab^i$ for some $i \geq 1$ or $x = b^j$ for some $j \geq 1$.

By the pumping lemma, uv must be in the language. $uv = b^{\ell-i}cd^\ell e$ or $uv = ab^{\ell-j}cd^\ell e$ – however, neither of these are in the language $\mathcal{L}$. So the original assumption that $\mathcal{L}$ is regular must have been incorrect. So $\mathcal{L}$ must not be regular.

(b) $\mathcal{L} = \{a^nb^{n+1} | n \geq 1\}$

Assume for contradiction that $\mathcal{L} = \{a^nb^{n+1} | n \geq 1\}$ is regular. Clearly $a^\ell b^{\ell+1}$ is in the langauge $\mathcal{L}$.

By the pumping lemma, $s = a^\ell b^{\ell+1}$ must be pumpable. So it must be of the form uxv for some $|x| \geq 1$, $|ux| \leq \ell$. Since the first ℓ characters of the string s are a, we can conclude that $x = a^i$ for some $i \geq 1$. By the pumping lemma if $\mathcal{L}$ is regular then $uv = a^{\ell-i}b^{\ell+1}$ must be in the language $\mathcal{L}$. However, $a^{\ell-i}b^{\ell+1}$ is not in the langauge $\mathcal{L}$. So $\mathcal{L}$ must not be regular.

(c) $\mathcal{L} = \{ww | w \in \{a, b\}^*\}$

Assume for contradiction that the langauge $\mathcal{L}$ is regular. Therefore, all strings s in the langauge can be pumped.

Consider in particular, the string $s = a^\ell b^\ell a^\ell b^\ell$. This string must be of the form uxv – where $|ux| \leq \ell$, with $|x| \geq 1$. So x is of the form a^k for some $k \in \mathbb{N}$. By the pumping lemma, we have that $s = uv$ must be in the language $\mathcal{L}$.

However, uv is of the form $a^{\ell-k}b^\ell a^\ell b^\ell \notin \mathcal{L}$. Therefore, the original assumption that the language $\mathcal{L}$ was regular must have been false.

2. Use the pumping lemma for context free languages to prove that the following are not context free:

(a) $\mathcal{L} = \{a^nb^nc^n | n \geq 1\}$

The Pumping Lemma for context-free languages states that all strings $s \in \mathcal{L}$ in a context free language such that $|s| > \ell$ are pumpable.

A string is pumpable if it is of the form $u_1xu_2yu_3$ for some $1 \leq |xy|$, $|xu_2y| \leq \ell$ and for all $n \in \mathbb{N}$, $u_1x^nu_2y^nu_3 \in \mathcal{L}$.

I prove that $\mathcal{L} = \{a^nb^nc^n | n \geq 1\}$ is not context free by proving that there exists a string $s \in \mathcal{L}$ such that s is not pumpable.

Assume for contradiction that the language $\mathcal{L}$ is context-free. So all strings $s \in \mathcal{L}$ can be pumped. Consider the string $s = a^\ell b^\ell c^\ell$. This is clearly in $\mathcal{L}$ – so by the pumping lemma for context-free languages, it must be pumpable. Consider all possible combinations for x, y for the string s.

- Case $xu_2y \subseteq a^\ell b^\ell$

  In this case, $xy = a^ib^j$ for some i, j such that $i + j \geq 1$. By the pumping lemma, we can conclude that $u_1u_2u_3 \in \mathcal{L}$. Therefore $s' = a^{\ell-i}b^{\ell-j}c^\ell \in \mathcal{L}$. However, we have $i \neq 0 \vee j \neq 0$. Therefore $\ell - i \neq \ell \vee \ell - j \neq \ell$. So the string s' must not be in $\mathcal{L}$. Contradiction!

  So in this case our original assumption that $\mathcal{L}$ is a context-free grammar must be incorrect.

- Case $xu_2y \subseteq b^\ell c^\ell$

  Similar

(b) $\mathcal{L} = \{a^n b^n c^m | n \leq m\}$

Assume for contradiction that the langauge $\mathcal{L}$ is context-free. Therefore, by the pumping lemma for context free languages, we have that all strings $s \in \mathcal{L}$, there exists u_1, x, u_2, y, u_2 such that $s = u_1 x u_2 y u_3$ and for all $n \in \mathbb{N}$, $u_1 x^n u_2 y^n u_3 \in \mathcal{L}$.

Consider in particular the string $s = a^\ell b^\ell c^\ell$. By the definition of $\mathcal{L}$, we have that $s \in \mathcal{L}$. Case split on the values of xu_2y:

- Case $xu_2y \subseteq a^\ell b^\ell$

  Therefore, xy is of the form $a^i b^j$ for some i, j such that $i + j \geq 1$. Consider now the string $s' = u_1 x^2 u_2 y^2 u_3$. By the pumping lemma for context-free languages, we have that s' must be in the language $\mathcal{L}$.

  By cardinality argument, if $xy = a^i b^i$, then $s' = u_1 x^2 u_2 y^2 u_3$ must contain $\ell + i$ as, $\ell + j$ bs and ℓ cs. Since we have the requirement that $i > 0 \vee j > 0$, we have that $\ell + i > \ell \vee \ell + j > \ell$. Therefore, we have $n > m$. However, this means s' no longer conforms to the definition of $\mathcal{L}$. So $s' \notin \mathcal{L}$.

  So $xu_2y \not\subseteq a^\ell b^\ell$

- Case $xu_2y \subseteq b^\ell c^\ell$

  Therefore, xy is of the form $b^i c^j$ for some i, j such that $i + j \geq 1$. Consider the string $s' = u_1 x^0 u_2 y^0 u_3$. By a simple cardinality argument, we have that s' contains ℓ as, $\ell - i$ bs and $\ell - j$ cs. Since all strings in $\mathcal{L}$ contain an equal number of as and bs, we have $\ell = \ell - i \implies i = 0$. Since $m \geq n$, we have $\ell - j \geq \ell \implies j = 0$. Therefore, we have $i = j = 0$. So $s' \notin \mathcal{L}$.

  So $xu_2y \not\subseteq b^\ell c^\ell$

So there is no partition of s into five substrings such that s is pumpable. Therefore, there exists a string $s \in \mathcal{L}$ such that s is not pumpable. So $\mathcal{L}$ is not a context-free language.

4 Top-Down parsing of Context Free Grammars

1. Write an implementation of the Earley Parser that can use the toy grammar from Lecture 3 to parse the sentences below: How many parses are there for each sentence?

   (a) they can fish in rivers

   There are 4 parses.

   (b) They can fish in rivers in December

   There are 9 parses.

5 Comparing grammar formalisms

1. Consider the following sentences:

   - Alice eats cakes
   - The caterpillar gives alice cakes
   - The cat with a grin disappears
   - Alice paints white roses red

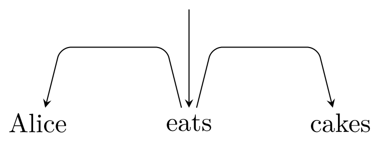
   Using examples in the notes / slides to start you off, complete the following tasks:

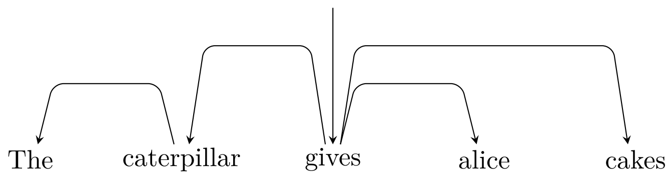
(a) Define a context free grammar that could generate the sentences.

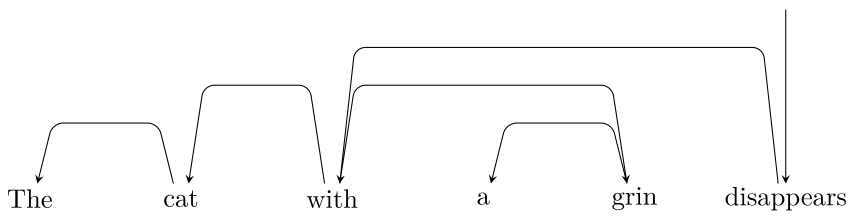
$$G = \{\mathcal{N}, \Sigma, \mathcal{P}, S\}$$

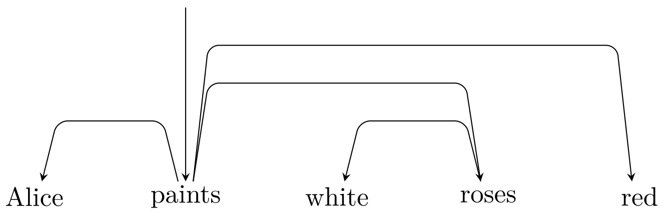
$$
\begin{aligned}
\mathcal{N} =& \{S, NP, VP, DT, PP, P, PN, N, V, ADJ\} \\
\Sigma =& \{ \\
& \quad \text{a, alice, cakes, cat, caterpillar,} \\
& \quad \text{disappears, eats, gives, grin,} \\
& \quad \text{paints, red, roses, the, white, with} \\
& \} \\
\mathcal{P} =& \{ \\
& \quad S \to NP\ VP \\
& \quad NP \to PN \mid DT\ N \mid N \\
& \quad VP \to V \mid V\ NP \mid V\ NP\ NP \mid V\ NP\ ADJ \\
& \quad DT \to \text{a} \mid \text{the} \\
& \quad PP \to P\ NP \\
& \quad P \to \text{with} \\
& \quad PN \to PN\ PP \mid \text{Alice} \\
& \quad N \to ADJ\ N \mid N\ PP \mid \text{cakes} \mid \text{cat} \mid \text{grin} \mid \text{caterpillar} \mid \text{roses} \\
& \quad V \to \text{disappears} \mid \text{eats} \mid \text{gives} \mid \text{paints} \\
& \quad ADJ \to \text{white} \mid \text{red} \\
& \}
\end{aligned}
$$

(b) Draw a dependency parse for the sentences.



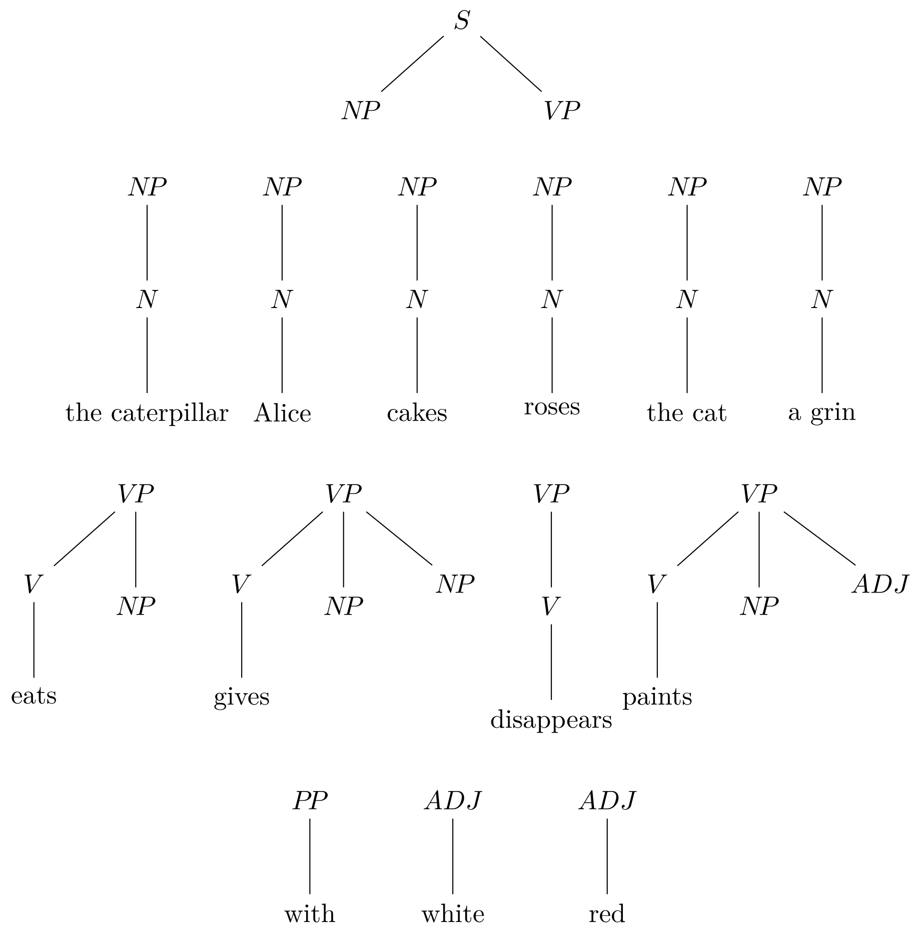




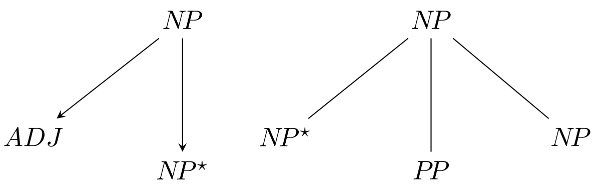


(c) Define a tree adjoining grammar that could generate the sentences.

The initial trees for the tree adjoining grammar are:



The auxiliary trees for the tree adjoining grammar are:



Part II

Exam Questions

6 2021 Paper 7 Question 4

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2021p7q4.pdf

(a) The following is a pattern for some legal strings in a langauge:

$$[a \in A]\{0,1\}\ [b \in B]\{0,1\}\ [c \in C]\{1,n\}\ [d \in D]\{1,1\}$$

where A is a finite set of characters from the alphabet, Σ; similarly for B, C, D. The sets A, B, C and D are disjoint. $\{x,y\}$ indicates that the previous bracket must match at least x times but no more than y times.

(i) Specify a Deterministic Finite Automaton, M_1, that can recognise these strings only.

It was hard to represent this. I decided the easiest way was a formal specification combined with a key which stated what each state represented.

$$M_1 = (\mathcal{Q} = \{q_s, q_a, q_b, q_{c_1}, \ldots, q_{c_n}, q_d, q_x\}, \Sigma, \delta, s = q_s, \mathcal{F} = \{q_d\})$$

Where Σ is defined as in the question

The states have the following intuitive meanings:

q_s : start state
q_a : valid so far – just seen an $a \in A$
q_b : valid so far – just seen a $b \in B$
q_{c_i} : valid so far – just seen the i^{th} $c \in C$
q_d : valid so far – just seen a $d \in D$
q_x : dead state

$$\delta \triangleq$$
$$\{((q_s,a),q_a)|a\in A\}\cup\{((q_s,b),q_b)|b\in B\}\cup\{((q_s,c),q_{c_1})|c\in C\}\cup$$
$$\{((q_s,d),q_x)|d\in D\}\cup\{((q_s,\sigma),q_x)|\sigma\in\Sigma\setminus(A\cup B\cup C\cup D)\}$$
$$\cup$$
$$\{((q_a,a),q_x)|a\in A\}\cup\{((q_a,b),q_b)|b\in B\}\cup\{((q_a,c),q_{c_1})|c\in C\}\cup$$
$$\{((q_a,d),q_x)|d\in D\}\cup\{((q_a,\sigma),q_x)|\sigma\in\Sigma\setminus(A\cup B\cup C\cup D)\}$$
$$\cup$$
$$\{((q_b,a),q_x)|a\in A\}\cup\{((q_b,b),q_x)|b\in B\}\cup\{((q_b,c),q_{c_1})|c\in C\}\cup$$
$$\{((q_b,d),q_x)|d\in D\}\cup\{((q_b,\sigma),q_x)|\sigma\in\Sigma\setminus(A\cup B\cup C\cup D)\}$$
$$\cup$$
$$\left(\bigcup_{i\in\mathbb{Z}^+_{<n}}\{((q_{c_i},a),q_b)|a\in A\}\cup\{((q_{c_i},b),q_x)|b\in B\}\cup\{((q_{c_i},c),q_{c_{i+1}})|c\in C\}\cup\{((q_{c_i},d),q_x)|d\in D\}\right)\cup$$
$$\{((q_{c_n},a),q_x)|a\in A\}\cup\{((q_{c_n},b),q_x)|b\in B\}\cup\{((q_{c_n},c),q_x)|c\in C\}\cup\{((q_{c_n},d),q_d)|d\in D\}\cup$$
$$\left(\bigcup_{i\in\mathbb{Z}^+_{\leq n}}\{((q_{c_i},\sigma),q_x)|\sigma\in\Sigma\setminus(A\cup B\cup C\cup D)\}\right)\cup$$
$$\{((q_d,\sigma),q_x)|\sigma\in\Sigma\}\cup$$
$$\{((q_x,\sigma),q_x)|\sigma\in\Sigma\}$$

(ii) Design a Regular Grammar, G_1, which generates $L(M_1)$.

Let $\underline{X}$ denote the regular expression $x_1|x_2|\dots|x_n$ for some set $X=\{x_1,x_2,\dots,x_n\}$.

$$G_1=(\underline{A}|\varepsilon)\,(\underline{B}|\varepsilon)\,\underline{C}\,\underbrace{(\underline{C}|\varepsilon)\dots(\underline{C}|\varepsilon)}_{n-1\text{ times}}\,\underline{D}$$

(iii) Describe a set of strings in a natural langauge that could be generated by G_1, given an appropriate Σ and its subsets A, B, C and D.

I was unable to think of a high-level natural language construct which was of this form. I therefore propose the set of "fake laughs".

$\Sigma=\mathcal{P}([a-z\_'] \{1,20\})$	all phrases of length less than 10 characters
$A=\{\text{oh my}\}$	
$B=\{\text{lol, lmao}\}$	
$C=\{\text{ha}\}$	
$D=\{\text{I'm dying}\}$	

(b) We can hypothesise that matches of the following pattern are always valid constructions in English:

$$[\texttt{The}\ Noun]\{n,n\}\ [Verb]\{n,n\}$$

where $Noun$ represents the coordinated members of a finite set; similarly for $Verb$.

(i) Now consider the following English sentence which matches the pattern when $n=1$:

`The vaccine worked`

Provide example sentences that extend this sentence for the case when $n=2$ and $n=3$.

```
The vaccine the researcher created worked
The vaccine the researcher the dog loved created worked
```

(c) Assuming that these constructions are part of the English language, would this mean that English is a Context-Free langauge? Justify your answer.

No. For two reasons:

- If there is a context-free construction in a grammar, this does not mean the grammar is context free. For example, the grammar [The $Noun$]* [$Verb$]* is regular, and yet accepts all constructions of the form

  $$[\texttt{The } Noun]\{n,n\}\ [Verb]\{n,n\}$$

  So, this is insufficient proof to prove that the language is *not* regular.

- We have no evidence that the English can be fully captured by a context free grammar. We have not proved that English does not contain constructions requiring context sensitivity to recognise.

  Is "respectively" an example of context-sensitivity in English? If not, why? https://aclanthology.org/C96-1028.pdf states that the examples in swiss-german are the only known extra-context free natural language syntactic phenomena. However it later references "respectively" (footnote on page 6) and gives a confusing explanation as to why it's not.

(d) Design a grammar in Chomsky Normal Form, G_2, which generates the finite matches of the pattern.

$$G_2 = (\{S, Verb, Noun\}, \Sigma, \mathcal{P}, S)$$

$$\begin{aligned}\mathcal{P} = \{ & \\ & S \to \text{The } Noun\ S\ Verb \\ & Noun \to \dots \\ & Verb \to \dots \\ \} & \end{aligned}$$

(e) Specify a Push Down Automaton, M_2, that recognises $L(G_2)$.

$$\begin{tikzcd}
{} \arrow[r] & q_s \arrow[r, bend left, "\varepsilon : \varepsilon/\$"] & q_0 \arrow[r, bend left, "\text{the} : \varepsilon/V"] & q_1 \arrow[r, bend left, "Noun : \varepsilon/\varepsilon"] & q_2 \arrow[r, bend left, "Verb : V/\varepsilon"] \arrow[ll, bend left, "\text{the} : \varepsilon/V"] & q_3 \arrow[r, "\varepsilon : \$/\varepsilon"] \arrow[loop below, "Verb : V/\varepsilon"] & \boxed{q_f}
\end{tikzcd}$$

7 2019 Paper 7 Question 5

https://www.cl.cam.ac.uk/teaching/exams/pastpapers/y2019p7q5.pdf

(a) Tree Adjoining Grammars contain two types of elementary tree.

(i) What are these trees called?

Initial trees (aka α-trees) and Auxiliary trees (aka β-trees).

(ii) If one were building a grammar for English which aspects of lanuage do the two tree types model?

I believe there are two "correct" answers to this. One intuitive and one theoretical.

- Intuitively

  Initial trees model the core parts of a sentence.

  Auxiliary trees model additional information in a sentence – i.e adjunction could be used to convert "the cat disappears" into "the cat with a grin disappears".

  This interpretation allows the grammar to more reflect human intuition about the structure of a sentence.

- Theoretically

  Initial trees model all context free parts of language. Substitution is sufficient to model all context free languages.

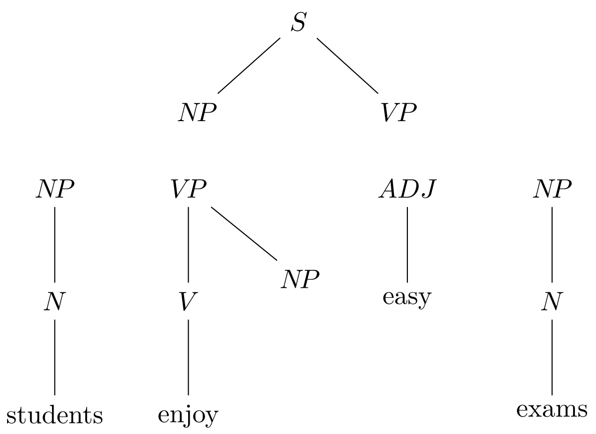
  Auxiliary trees model context-sensitive parts of natural language (such as cross-serial dependencies). Adjunction extends the grammar just enough to be a superset of natural language.

(b) Provide a Tree Adjoining Grammar that can parse the string: *students enjoy easy exams.*

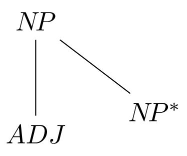
A tree adjoining grammar is specified by a quintuple $(\mathcal{N}, \Sigma, S, \mathcal{I}, \mathcal{A})$.

I define a tree adjoining grammar $G = (\{S, NP, VP, ADJ, N, V\}, \{\text{students}, \text{enjoy}, \text{easy}, \text{exams}\}, S, \mathcal{I}, \mathcal{A})$

The initial trees $\mathcal{I}$ are:



The Auxiliary trees $\mathcal{A}$ are:



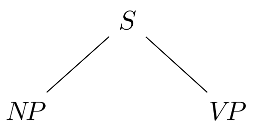
(c) Show how a parse for this string is constructed. Explain the operations.

Parses for strings are constructed by a combination of substitution and adjunction. In substitution, an initial tree with root X is attached to a leaf X in the parse tree. In adjunction, we insert an auxiliary tree with root X onto a node X in the middle of a parse tree and attach the remainder of the parse tree onto the foot of the auxiliary tree.

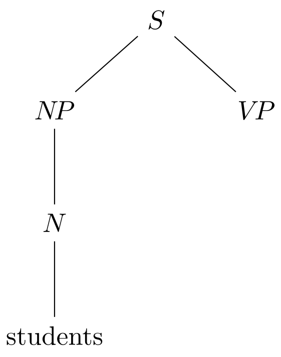
Start with the root.

$$S$$

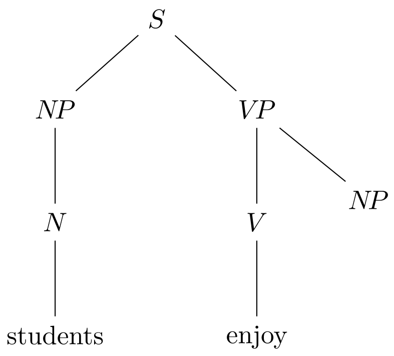
Substitute in an initial tree with root S:



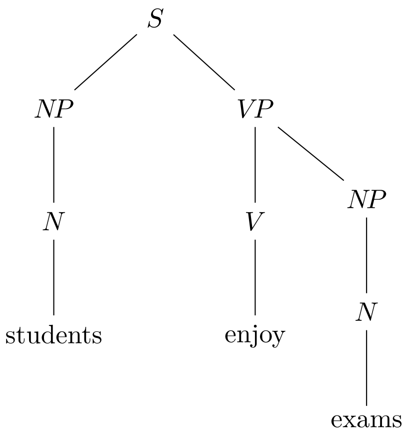
Substitute in an initial tree with root NP:



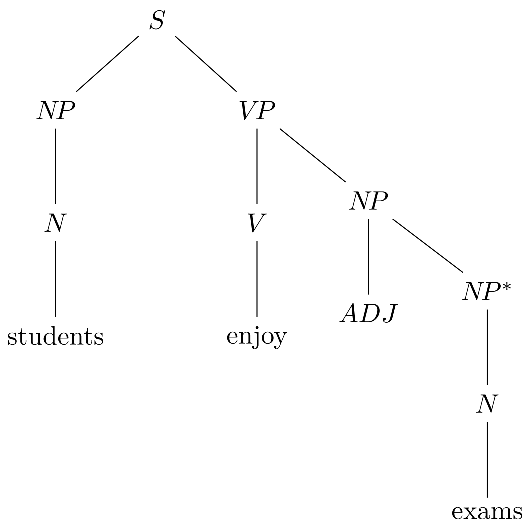
Substitute in an initial tree with root VP:



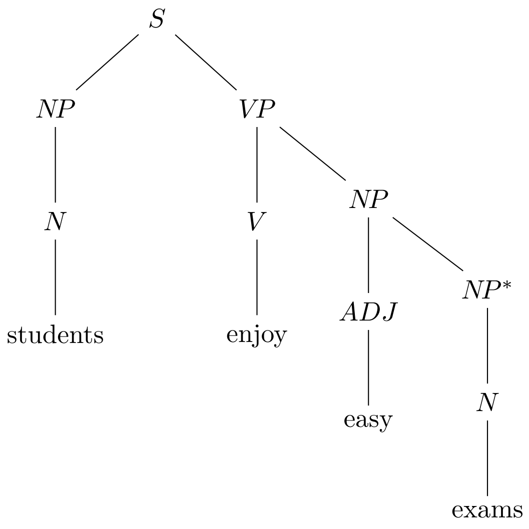
Substitute in an initial tree with root NP:



Use adjunction to insert the auxiliary tree with root *NP*:



Substitute in an initial tree with root *ADJ*:



(d) Provide a Categorial Grammar that can parse the same sentence.

A Categorial Grammar can be represented as a quadruple of $(\Sigma, P_r, S, \mathcal{R})$, where Σ is the alphabet, P_r is the set of primitive types, S is the root of completed derivations and $\mathcal{R}$ is the typing relation.

A Categorial Grammar that can parse the same sentence is given by:

$$
\begin{aligned}
M =& (\Sigma, S, P_r, \mathcal{R}) \\
\Sigma =& \{\text{students}, \text{enjoy}, \text{easy}, \text{exams}\} \\
P_r =& \{S, V, N, ADJ\} \\
\mathcal{R} =& \{ \\
& \qquad (\text{students}, S/VP), (\text{students}, NP), (\text{students}, NP\backslash ADJ) \\
& \qquad (\text{enjoy}, VP/NP) \\
& \qquad (\text{easy}, ADJ) \\
& \qquad (\text{exams}, S/VP), (\text{exams}, NP), (\text{exams}, NP\backslash ADJ) \\
& \}
\end{aligned}
$$

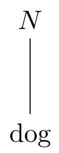
(e) When children learn their first language they usually acquire nouns before verbs before modifiers. They also usually produce single word strings before moving on to longer strings. With reference to Tree Adjoining Grammars and / or Categorial Grammars propose some hypotheses for this. Justify your proposals.

I use Tree Adjoining Grammars to approximate the complexity of sentence structure. The more complicated the structure of the parse tree, the higher the human parsing complexity of the sentence. This approach is inspired by Hale's interpretation of improbable sentences being harder to parse – rather I state that improbable sentence *structures* are harder to learn.

Children have a finite learning rate. They are therefore likely to learn the simplest sentences with which they are able to express their wants. As such, I hypothesise that children will learn sentences which can be expressed by the simplest Tree Adjoining Grammars first.

Note firstly, that longer sentences require larger and significantly more complicated parse trees to process. Therefore, children are likely to learn single-word sentences which suffice their needs first.

Nouns are associated with the simplest tree structures. Nouns don't operate on any other words and are standalone – the trees which generate nouns are simple (below). From an information-theoretic viewpoint, nouns are the most efficient way of conveying meaning, making them the most efficient words to learn first.



This contrasts with verbs – verbs operate on other words i.e "give santa the biscuit" and form complex sentences. As a result the initial trees to use verbs are more complex than for nouns as are the parse trees generated by sentences using verbs. As such, children will often learn verbs after nouns.

Modifiers not only require more advanced trees, they require an entirely new operation – adjunction. This is more complicated to generate and as such harder for young children to learn. Therefore, children often learn them later.

Single word-strings have far simpler parse trees than multiword strings and fully-formed sentences. Therefore, children find them easier to parse and create.