Ex. 1

The two following grammars engender a subset of arithmetic expressions (for instance $1 + 3 \times 7$). What is the difference between the two?

$$
\begin{array}{ll}
E \to E + E \mid E \times E & E \to E + T \mid T \\
E \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 & T \to T \times F \mid F \\
 & F \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

Propose 3 new grammars, taking inspiration from those two grammars:

1. One that engenders the same language but yields a structural decomposition that gives priority to addition over multiplication;
2. One that engenders a sub-language of the previous one: namely only arithmetic expressions completely parenthesized (expressions where there is a pair of parenthesis for each binary operator);
3. One that engenders postfix arithmeric expressions (reversed Polish notation, where $7 \times 2 + 3$ is written $7\ 2 \times 3 +$).

.... Answer ....

0. The difference is that the first grammar is ambiguous, and not the second one: for instance $7 \times 2 + 3$ can be analysed as $[_E\ 7 \times [_E\ 2 + 3]]$ or as $[_E\ [_E\ 7 \times 2] + 3]$ by the first grammar, and only as $[_E[_T\ 7 \times 2] + 3]$ by the second one.

   About the second grammar, note that since the language has no brackets, there is no way to force an analysis of, say, $7 \times 2 + 3$ which would be equivalent to $7 \times (2 + 3)$. In other words, any multiplication will always have priority over additions. This is the reason why the more common version of ETF comprises an additional rule allowing to introduce brackets around sub-expressions:

$$
\begin{array}{l}
E \to E + T \mid T \\
T \to T \times F \mid F \\
F \to (\,E\,) \\
F \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

1. To engender a language where addition has higher probability it is enough to invert the role of the two operators in the second grammar:

$$
\begin{array}{l}
E \to E \times T \mid T \\
T \to T + F \mid F \\
F \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

2. The statement of the exercise was not consistent: since I chose to work with languages without brackets, one cannot say in this question that the language is a sub-language of the previous one. It's just a new language where the presence of brackets for every occurrence of a binary operator removes all ambiguity. It's not necessary to take the second grammar as a starting point:

$$
\begin{array}{l}
E \to (E + E) \mid (E \times E) \\
E \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

3. This language is not ambiguous either. We don't need the ETF device, a much simpler grammar is sufficient:

$$
\begin{array}{l}
E \to EE+ \ \mid EE\times \\
E \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

Ex. 2

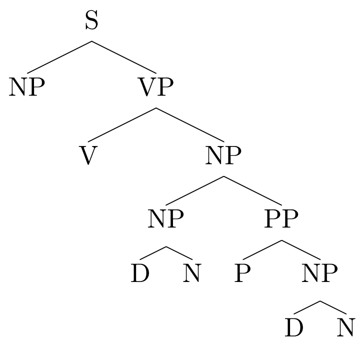
Let's consider the sentence (1), which is well-known for its being syntactically ambiguous.

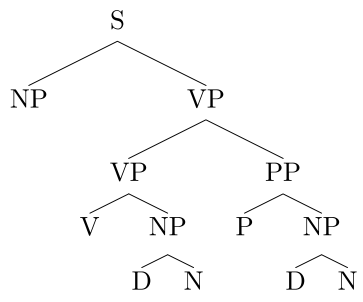
(1) Sam saw a girl with his telescope.

1. Show the syntactic ambiguity by providing two distinct syntactic trees for the sentence (to avoid dealing with a lexicon, we can consider lexical categories (N, Det, Prep, V...) as terminal symbols).
2. Give an ambiguous CFG grammar capable of generating the two possible syntactic analyses.
3. Give a CFG in which the ambiguity is removed by assuming a systematic low attachment strategy, according to which prepositional phrases will get attached to the closest noun.

Answer

1. On the left an analysis where the PP is adjoined to the NP, on the right a version where it is adjoined to the VP.



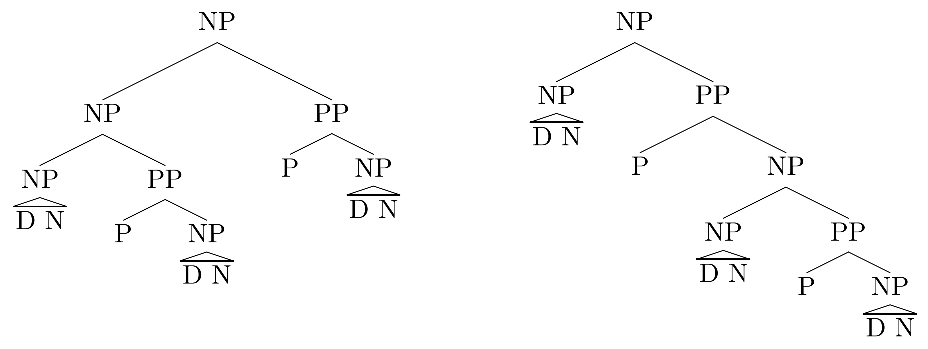


2. A typical cfg grammar covering the sentence would be the following one, where adjunction rules are used to introduce optional PPs (an adjunction rule is a rule of the form XP → XP YP).

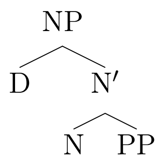
   An other option, less X-bar-like, would be to have ternary rules of the form VP → V NP PP.

S	→	NP	VP
NP	→	PN	
	|	D	N
	|	NP	PP
PP	→	P	NP
VP	→	V	NP
	|	VP	PP

3. An easy option to just get rid of the right hand-side analysis above would be to remove the last rule in the previous grammar. However, this would not warrant that in the case of embedded NPs we would get only the low attachement strategy: a sequence like D N P D N P D N would give rise to two different structures:



The left-hand analysis would be appropriate for a phrase like *the mastery of the language by this teacher*, while the second structure would be more appropriate for a phrase like *the cat of the neighbour of my brother*. To ensure that a low attachment strategy always prevail, we need to find a way to attach PPs very low. An option would be to have rules of this format:



(We need also to have a singleton rule N′ → N)