CFL2 - Solution Sketches

March 2, 2020

We sketch here the solutions for the exercises of lecture CFL2 in a brief manner. Note that a proper solution would require more detailed descriptions, explanations, and in some cases examples. Some of the exercises may have more than one solution, and we just show one of them. When the exercise requests a context-free grammar, we provide one in a very compact way (as done in the slides) where we just provide the productions, with the implicit assumption that non-terminals start with capital letters, and that the initial symbol is the on corresponding to the first production.

Exercise 2.1

$$
\begin{array}{rcl}
\texttt{EMAIL} & \rightarrow & \texttt{<email>}\ \texttt{FROM}\ \texttt{TOP}\ \texttt{BCCN}\ \texttt{SUBJECT}\ \texttt{BODY}\ \texttt{</email>} \\
\texttt{FROM} & \rightarrow & \texttt{<from>\#PCDATA</from>} \\
\texttt{TOP} & \rightarrow & \texttt{\#TO} \mid \texttt{\#TO}\ \texttt{\#TOP} \\
\texttt{TO} & \rightarrow & \texttt{<to>\#PCDATA</to>} \\
\texttt{BCCN} & \rightarrow & \epsilon \mid \texttt{\#BCC}\ \texttt{\#BCCN} \\
\texttt{BCC} & \rightarrow & \texttt{<bcc>\#PCDATA</bcc>} \\
\texttt{SUBJECT} & \rightarrow & \texttt{<subject>\#PCDATA</subject>} \\
\texttt{BODY} & \rightarrow & \texttt{<body>\#PCDATA</body>}
\end{array}
$$

Exercise 2.2

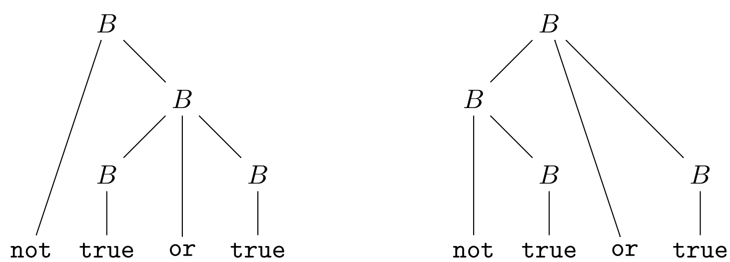
$$
\begin{array}{rcl}
\texttt{Obj} & \rightarrow & \{\} \mid \{\texttt{Pairs}\} \\
\texttt{Pairs} & \rightarrow & \texttt{Pair} \mid \texttt{Pair}, \texttt{Pairs} \\
\texttt{Pair} & \rightarrow & \texttt{String} : \texttt{Value} \\
\texttt{Array} & \rightarrow & [] \mid [\texttt{Values}] \\
\texttt{Values} & \rightarrow & \texttt{Value} \mid \texttt{Value}, \texttt{Values}
\end{array}
$$

Exercise 2.3 The are two sources of ambiguity: (1) precedence between `not` and `or` and (2) associativity of the binary operator `or`.

An example of (1) can be observed in string

`not true or true`

for which we can give two parse trees:



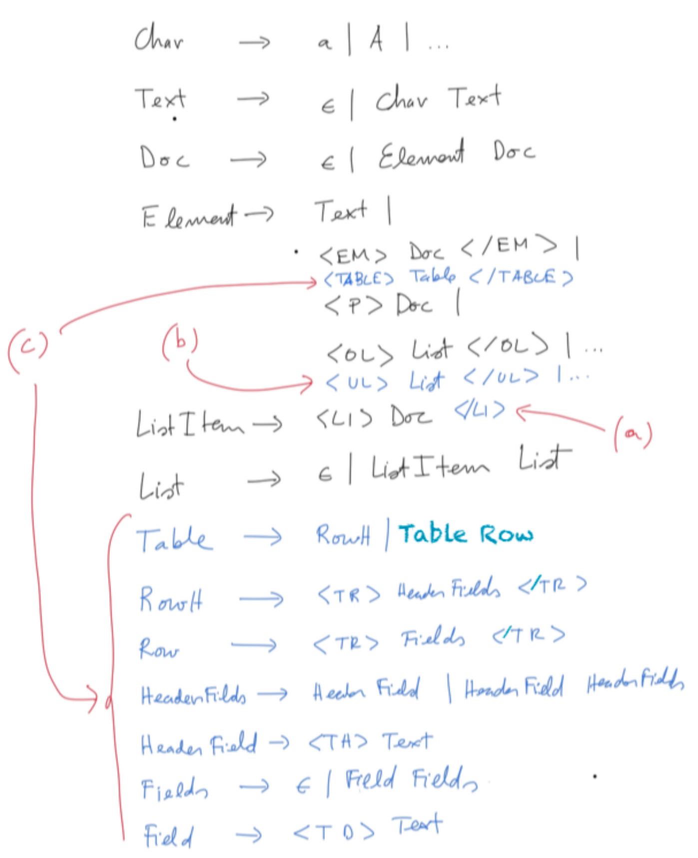
We begin solving (1) by giving precedence to `not` over `or` using the stratification technique:

$$\begin{array}{lcl} B_0 & \rightarrow & B_0 \ \texttt{or} \ B_0 \mid B_1 \\ B_1 & \rightarrow & \texttt{true} \mid \texttt{not} \ B_1 \mid (B_0) \end{array}$$

Next we address (2) by choosing left-associativity and transforming the grammar as follows:

$$\begin{array}{lcl} B_0 & \rightarrow & B_0 \ \texttt{or} \ B_1 \mid B1 \\ B_1 & \rightarrow & \texttt{true} \mid \texttt{not} \ B_1 \mid (B_0) \end{array}$$

Exercise 2.4



(c) (b)

Char → a | A | ...
Text → ϵ | Char Text
Doc → ϵ | Element Doc
Element → Text |
<EM> Doc </EM> |
<TABLE> Table </TABLE>
<P> Doc |
<OL> List </OL> | ...
<UL> List </UL> | ...
ListItem → <LI> Doc </LI> (a)
List → ϵ | ListItem List
Table → RowH | **Table Row**
RowH → <TR> HeaderFields </TR>
Row → <TR> Fields </TR>
HeaderFields → HeaderField | HeaderField HeaderFields
HeaderField → <TH> Text
Fields → ϵ | Field Fields
Field → <TD> Text

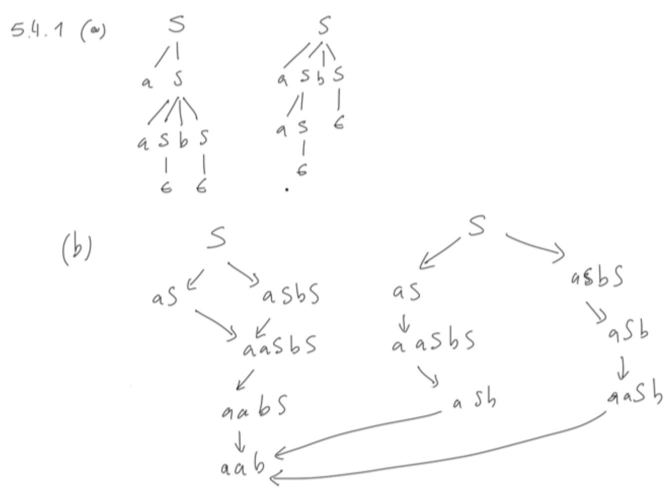
Exercise 2.5

COURSESOPECS	→	<COURSESPECS> COURSES </COURSESPECS>
COURSES	→	COURSE | COURSE COURSES
COURSE	→	<COURSE> CNAME PROFSTUDENTS TAP </COURSE>
CNAME	→	<CNAME> #PCDATA </CNAME>
PROF	→	<PROF> #PCDATA </PROF>
STUDENTS	→	ϵ | STUDENT STUDENTS
STUDENT	→	<STUDENT> #PCDATA </STUDENT>
TAP	→	ϵ | TA
TA	→	<TA> #PCDATA </TA>

Exercise 2.6 / 5.4.1 (d) The ambiguity is similar to the *dangling else* problem. Think of a being `if...then...` and b being `else`. The ambiguity means that in general the grammar does not tell us to which `if` an `else` belongs, e.g. in `aab` the `b` could refer either to the first or the second `a`.

A common convention in programming languages is that each `else` belongs to the closest syntactically possible `if`. E.g. in `aaabb`, the first `b` belongs to the third `a`, the second `b` belongs to the second `a`, the first `a` has no `b`, i.e. as writing `aaabb` in a programming language with braces for structuring.

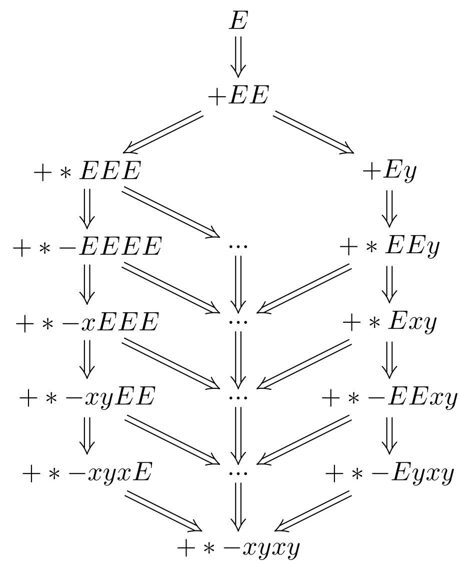
Exercise 2.6 / 5.4.1 (a)–(b)



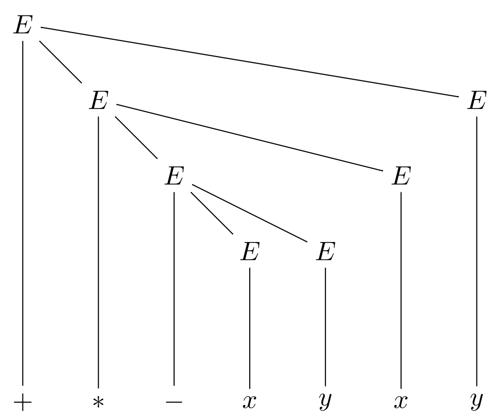
Exercise 2.6 (d) / 5.4.3 Idea: the production $S \to aSbS$ allows more than we want: the first S could be replaced with a string that has more a's then b's, then we could attach the b of this production also to an a produced by S, and that is exactly the ambiguity we want to avoid. Therefore, we want to restrict this first S to "well-balanced" strings, represented by the new symbol S_0:

$$
\begin{array}{rcl}
S & \to & aS \mid aS_0bS \mid \epsilon \\
S_0 & \to & aS_0bS_0 \mid \epsilon
\end{array}
$$

Exercise 2.7 (a) The derivation tree would be as follows, where, for brevity, we just draw the left-most and right-most derivation branches, and leave the rest underspecified with ...:



(b) The unique parse tree for $+*-xyxy$ is



(c) The grammar is unambigous. To see this observe that for any string w generated by the grammar, and each possible initial terminal symbol of w, w can only be obtained by one production. This means that the left-derivation of E and any string derived by E is always unique, and corresponds hence to a unique parsing tree.