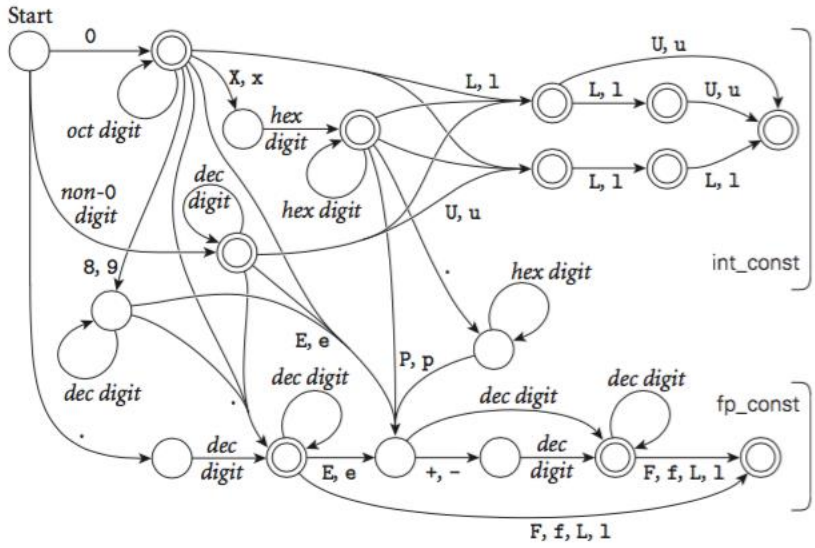
Homework Assignment 2, Theory Part

Problem 1. (i) Identify languages, i.e. describe them in the most direct way, of all variables listed below.

C_constant ⟶ *int_const* | *fp_const*
int_const ⟶ (*oct_int* | *dec_int* | *hex_int*) *int_suffix*
oct_int ⟶ `0` *oct_digit* *
dec_int ⟶ *nonzero_digit* *dec_digit* *
hex_int ⟶ (`0x` | `0X`) *hex_digit* *hex_digit* *
oct_digit ⟶ `0` | `1` | `2` | `3` | `4` | `5` | `6` | `7`
nonzero_digit ⟶ `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9`
dec_digit ⟶ `0` | *nonzero_digit*
hex_digit ⟶ *dec_digit* | `a` | `b` | `c` | `d` | `e` | `f` | `A` | `B` | `C` | `D` | `E` | `F`
int_suffix ⟶ ϵ | *u_suffix* (*l_suffix* | *ll_suffix* | ϵ)
| *l_suffix* (*u_suffix* | ϵ) | *ll_suffix* (*u_suffix* | ϵ)
u_suffix ⟶ `u` | `U`
l_suffix ⟶ `l` | `L`
ll_suffix ⟶ `ll` | `LL`
fp_const ⟶ (*dec_fp* | *hex_fp*) (`f` | `F` | `l` | `L` | ϵ)
dec_fp ⟶ *dec_digit* *dec_digit* * (`E` | `e`) *exponent*
| *dec_digit* * (`.` *dec_digit* | *dec_digit* `.`) *dec_digit* * ((`E` | `e`) *exponent* | ϵ)
hex_fp ⟶ (`0x` | `0X`) *hex_digit* * (`.` *hex_digit* | ϵ | *hex_digit* `.`) *hex_digit* *
(`P` | `p`) *exponent*
exponent ⟶ (`+` | `-` | ϵ) *dec_digit* *dec_digit* *

(ii) Explain the correspondence between the grammar of part (i) and the automaton copied below. Provide representative examples of derivations of the same strings in both, the grammar and the automaton.



Problem 2. Consider the following LL(1) and SLR(1) grammars for the language of balanced parentheses and brackets and their corresponding parse tables.

LL(1) grammar:

1. $P \longrightarrow S$ **$$**
2. $S \longrightarrow$ (S) S
3. $S \longrightarrow$ [S] S
4. $S \longrightarrow \epsilon$

SLR(1) grammar:

1. $P \longrightarrow S$ **$$**
2. $S \longrightarrow S$ (S)
3. $S \longrightarrow S$ [S]
4. $S \longrightarrow \epsilon$

LL(1) parse table:

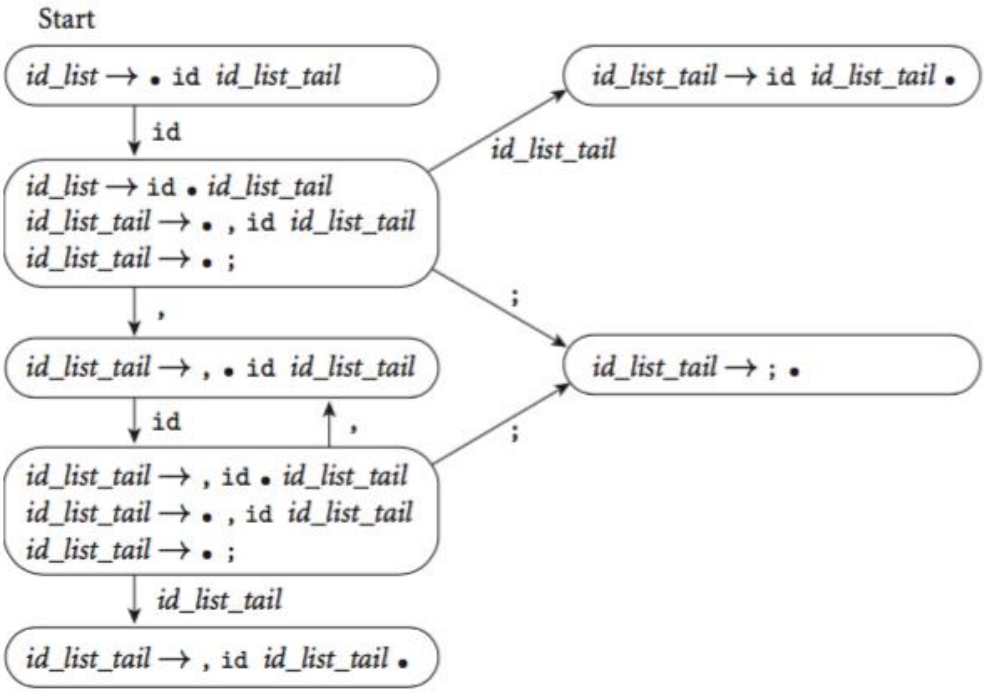
Top-of-stack nonterminal	Current input token				
	(	)	[	]	$$
P	1	–	1	–	1
S	2	4	3	4	4

SLR(1) parse table:

Top-of-stack state	Current input symbol					
	S	(	)	[	]	$$
0	s1	r4	r4	r4	r4	r4
1	–	s2	–	s3	–	b1
2	s4	r4	r4	r4	r4	r4
3	s5	r4	r4	r4	r4	r4
4	–	s2	b2	s3	–	–
5	–	s2	–	s3	b3	–

Give traces of the actions of their parsers in the format of Figure 2.21 and Figure 2.30 for the strings ([([( )])]), ( ) [ ] ( ) [ ] ( ).

Problem 3. Verify via several representative examples that the automaton copied below is a CFSM for the grammar of PLP Example 2.20 and that it can be used for parsing bottom-up with zero tokens of look-ahead.



Note that a dot appears at the end of an item in only three states, and there are no other items in these states. Lookahead is never required to resolve a shift-reduce conflict; the grammar is thus LR(0).