1. (20pt) For an SLR(1) grammar G, if there any relationship between (a) the infinity of $L(G)$ and (b) the existence of cycles in the LR-graph of G? You are required to prove which of the four possible combinations:

   (a) (5pt) finite/no cycles,

   (b) (5pt) finite/cycles,

   (c) (5pt) infinite/no cycles,

   (d) (5pt) infinite/cycles

   are possible and which are not. Prove your answers by giving examples when possible and prove the impossibility otherwise. (It is assumed that all nonterminals of G are useful for deriving strings.)

(a) Possible. If a grammar generates a finite language, its LR-graph might not contain cycles. The absence of cycles in the LR-graph is consistent with the idea that the language is finite because there is a limited number of states that the LR-parser will need to pass through to accept the strings of the language.

$S \longrightarrow a$

This grammar generates a finite language G, and its LR-graph has no cycles since parsing this string doesn't involve recursive state transitions.

(b) Not possible. If the LR-graph contains cycles, it typically indicates that there are recursive productions in the grammar, which allow for an infinite number of derivations. This would contradict the assumption that the language is finite. Hence, a finite language cannot have cycles in its LR-graph.

If a cycle exists in the LR-graph, this suggests that the parser can loop through states, implying the possibility of generating infinitely long strings. But if the language is finite, this is not possible.

(c) Not possible. If a language is infinite, the LR-graph must contain cycles. Without cycles, there would be a finite number of states and transitions, which implies the parser could only process finitely many strings. Hence, an infinite language without cycles cannot exist.

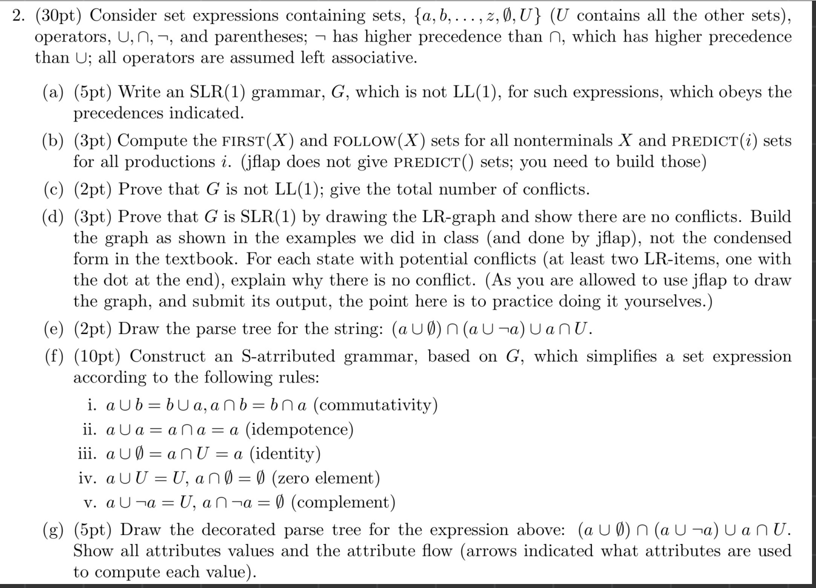
The absence of cycles means the LR-graph is acyclic, and an acyclic graph can only represent finitely many derivations. However, for an infinite language, we would require an infinite number of derivations.

(d) Possible. This is possible and expected. An infinite language generated by a grammar means that the grammar allows for recursive or repetitive constructs, which result in cycles in the LR-graph. These cycles correspond to the recursive nature of the grammar that allows the derivation of an infinite number of strings.

$S \longrightarrow aS$

$ \longrightarrow a$

This generates an infinite language G, and its LR-graph will have cycles due to the recursive production $S \rightarrow aS$



2. (30pt) Consider set expressions containing sets, $\{a, b, \ldots, z, \emptyset, U\}$ (U contains all the other sets), operators, $\cup, \cap, \neg$, and parentheses; $\neg$ has higher precedence than $\cap$, which has higher precedence than $\cup$; all operators are assumed left associative.

(a) (5pt) Write an SLR(1) grammar, G, which is not LL(1), for such expressions, which obeys the precedences indicated.

(b) (3pt) Compute the FIRST(X) and FOLLOW(X) sets for all nonterminals X and PREDICT(i) sets for all productions i. (jflap does not give PREDICT() sets; you need to build those)

(c) (2pt) Prove that G is not LL(1); give the total number of conflicts.

(d) (3pt) Prove that G is SLR(1) by drawing the LR-graph and show there are no conflicts. Build the graph as shown in the examples we did in class (and done by jflap), not the condensed form in the textbook. For each state with potential conflicts (at least two LR-items, one with the dot at the end), explain why there is no conflict. (As you are allowed to use jflap to draw the graph, and submit its output, the point here is to practice doing it yourselves.)

(e) (2pt) Draw the parse tree for the string: $(a \cup \emptyset) \cap (a \cup \neg a) \cup a \cap U$.

(f) (10pt) Construct an S-atrributed grammar, based on G, which simplifies a set expression according to the following rules:

i. $a \cup b = b \cup a, a \cap b = b \cap a$ (commutativity)
ii. $a \cup a = a \cap a = a$ (idempotence)
iii. $a \cup \emptyset = a \cap U = a$ (identity)
iv. $a \cup U = U$, $a \cap \emptyset = \emptyset$ (zero element)
v. $a \cup \neg a = U$, $a \cap \neg a = \emptyset$ (complement)

(g) (5pt) Draw the decorated parse tree for the expression above: $(a \cup \emptyset) \cap (a \cup \neg a) \cup a \cap U$. Show all attributes values and the attribute flow (arrows indicated what attributes are used to compute each value).

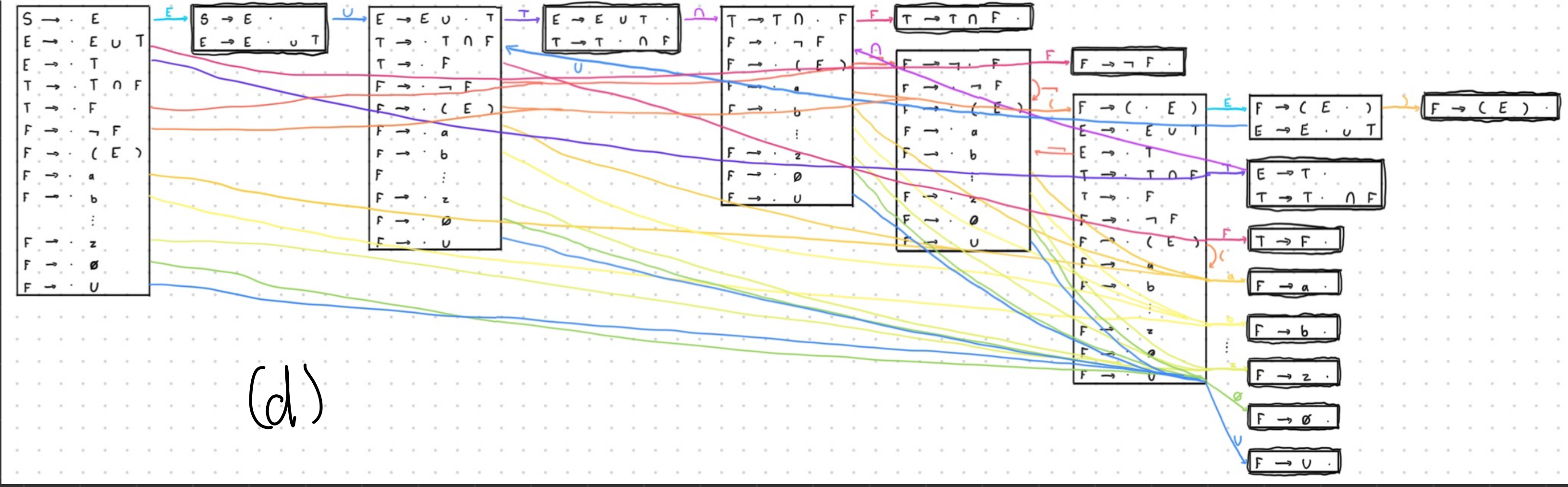
(a)
1. $S \longrightarrow E$
2. $E \longrightarrow E \cup T$
3. $\longrightarrow T$
4. $T \longrightarrow T \cap F$
5. $\longrightarrow F$
6. $F \longrightarrow \neg F$
7. $\longrightarrow (E)$
8… $\longrightarrow a \mid b \mid \ldots \mid z \mid \emptyset \mid U$

(b)

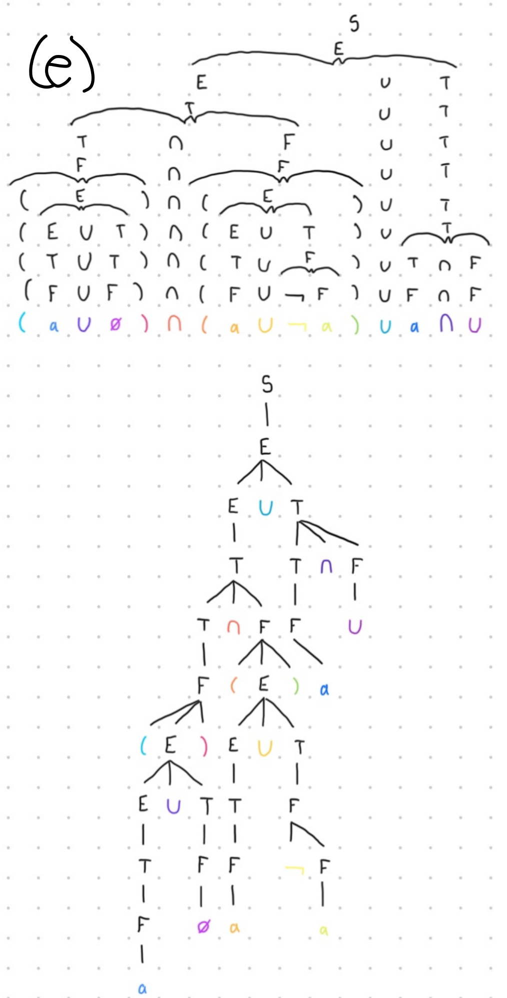
X	FIRST(X)	FOLLOW(X)
S	FIRST(E) —> ¬, (, a, b, …, z, ∅, U	∅
E	FIRST(E), FIRST(T) —> ¬, (, a, b, …, z, ∅, U	FOLLOW(S), U,) —> U,)
T	FIRST(T), FIRST(F) —> ¬, (, a, b, …, z, ∅, U	FOLLOW(E), ∩ —> U,), ∩
F	¬, (, a, b, …, z, ∅, U	FOLLOW(T), FOLLOW(F) —> U,), ∩

i	PREDICT(i)
1	FIRST(E) —> ¬, (, a, b, …, z, ∅, U
2	FIRST(E) —> ¬, (, a, b, …, z, ∅, U
3	FIRST(T) —> ¬, (, a, b, …, z, ∅, U
4	FIRST(T) —> ¬, (, a, b, …, z, ∅, U
5	FIRST(F) —> ¬, (, a, b, …, z, ∅, U
6	¬
7	(
8	a
9	b
⋮	⋮
33	z
34	∅
35	U

(c) $\neg, (, a, b, \ldots, z, \emptyset, U \in \text{PREICT}(\underbrace{2,3}_{E} + \underbrace{45}_{T})$



(d)



(e)

not the answer, just my thought process

(f) 1. $S \longrightarrow E$

$\triangleright$ $S.val = E.val$

2. $E_1 \longrightarrow E_2 \cup T$

$\triangleright$ $E.val_1 = union(E.val_2, T.val)$

3. $E \longrightarrow T$

$\triangleright$ $E.val = T.val$

4. $T_1 \longrightarrow T_2 \cap F$

$\triangleright$ $T.val_1 = intersection(T.val_2, F.val)$

5. $T \longrightarrow F$

$\triangleright$ $T.val = F.val$

6. $F_1 \longrightarrow \neg F_2$

$\triangleright$ $F.val_1 = complement(F.val_2)$

7. $F \longrightarrow (E)$

$\triangleright$ $F.val = E.val$

8. $F \longrightarrow a$

$\triangleright$ $F.val = a$

9. $F \longrightarrow b$

$\triangleright$ $F.val = b$

⋮

33. $F \longrightarrow z$

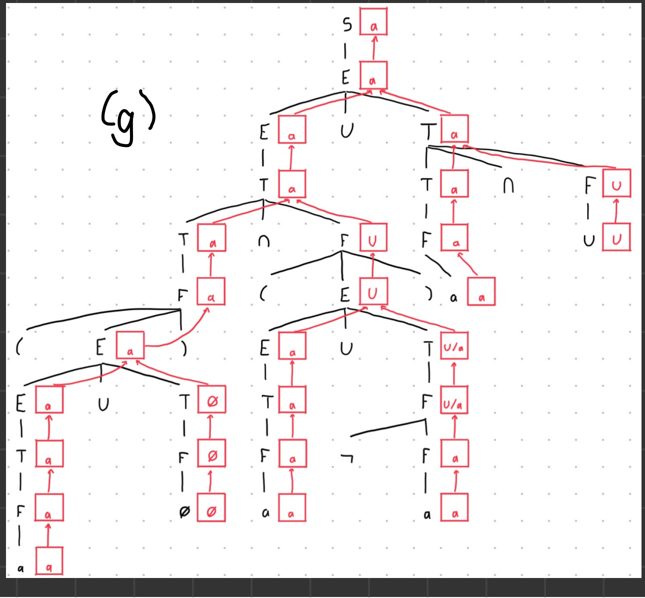
$\triangleright$ $F.val = z$

34. $F \longrightarrow \emptyset$

$\triangleright$ $F val = \emptyset$

35. $F \longrightarrow U$

$\triangleright$ $F.val = U$



(g)