

EXAM

Please read all instructions, including these, carefully

- There are 6 questions on the exam, with multiple parts. You have 3 hours to work on the exam.
- The exam is open book, open notes.
- Please write your *final* answers in the space provided on the exam. You may use the backs of the exam pages as scratch paper, or use additional pages (available at the front of the room).
- Each problem has a straightforward solution. Solutions will be graded on correctness and clarity. Partial solutions will be given partial credit.

NAME : _____

Problem	Max points	Points
1	25	
2	15	
3	15	
4	25	
5	20	
TOTAL	100	

1. Let G be the following grammar:

$$\begin{aligned} S &\rightarrow A \\ A &\rightarrow B \mid B a A \\ B &\rightarrow b C \\ C &\rightarrow C b \mid C c \mid \varepsilon \end{aligned}$$

(a) The grammar is not LL(1). Why? As part of your answer, show the FIRST and FOLLOW sets for each nonterminal symbol.

The rules for A have a common prefix and therefore the same symbol is in the first sets for both of its right hand sides, and two of the rules for C are left recursive. Both of these conditions violate the requirements for an LL(1) grammar.

FIRST and FOLLOW sets:

Symbol	First	Follow
S	b	EOF
A	b	EOF
B	b	a, EOF
C	b, c	a, b, c, \$

(b) Modify G to produce a new grammar, G' . Show that G' is LL(1). Describe the condition(s) that G' meets that makes it LL(1).

Eliminate A 's common prefix using left factoring:

$$\begin{aligned} S &\rightarrow A \\ A &\rightarrow B A' \\ A' &\rightarrow a A \mid \varepsilon \\ B &\rightarrow b C \\ C &\rightarrow C b \mid C c \mid \varepsilon \end{aligned}$$

Then eliminate the left recursion, giving G' :

$$\begin{aligned} S &\rightarrow A \\ A &\rightarrow B A' \\ A' &\rightarrow a A \mid \varepsilon \\ B &\rightarrow b C \\ C &\rightarrow b C \mid c C \mid \varepsilon \end{aligned}$$

Since the FIRST sets of the rules for each non-terminal are disjoint, the grammar is LL(1).

(c) Grammar G is not LR(0). Why?

Here are the states of the LR(0) machine:

State 0	State 5
$S' \rightarrow \cdot S$	$A \rightarrow B a A \cdot$
$S \rightarrow \cdot A$	State 6
$A \rightarrow \cdot B$	$B \rightarrow b \cdot C$
$A \rightarrow \cdot B a A$	$C \rightarrow \cdot C b$
$B \rightarrow \cdot b C$	$C \rightarrow \cdot C c$
State 1	$C \rightarrow \cdot$
$S' \rightarrow S \cdot$	State 7
State 2	$B \rightarrow b C \cdot$
$S \rightarrow A \cdot$	$C \rightarrow C \cdot b$
State 3	$C \rightarrow C \cdot c$
$A \rightarrow B \cdot$	State 8
$A \rightarrow B \cdot a A$	$C \rightarrow C b \cdot$
State 4	State 9
$A \rightarrow B a \cdot A$	$C \rightarrow C c \cdot$
$A \rightarrow \cdot B$	
$A \rightarrow \cdot B a A$	
$B \rightarrow \cdot b C$	

States 3, 6, and 7 have shift-reduce conflicts, so the grammar is not LR(0).

(d) Is grammar G SLR(1) or not? (Hint: build the SLR(1) machine, adding LL(1) FOLLOW sets to LR(0) states.) You need to draw only the relevant part(s) of the state machine; i.e., you may omit the obviously non-problematic parts.

State 3	
$A \rightarrow B \cdot$	FOLLOW(A) = EOF
$A \rightarrow B \cdot a A$	
State 6	
$B \rightarrow b \cdot C$	
$C \rightarrow \cdot C b$	
$C \rightarrow \cdot C c$	
$C \rightarrow \cdot$	FOLLOW(C) = {a, b, c, EOF}
State 7	
$B \rightarrow b C \cdot$	FOLLOW(B) = {a, EOF}
$C \rightarrow C \cdot b$	
$C \rightarrow C \cdot c$	

Using the FOLLOW sets as lookahead for the reduce moves, the LR(0) conflicts are resolved, so the grammar is SLR(1).

- (e) What is the smallest *language* category (of the standard categories) that includes the language of this grammar? Explain your answer.

The language is *regular*, it can be described with the regular expression $b(blc)^*(ab(blc))^*$

2. Consider the grammar

$$E \rightarrow B A$$

$$A \rightarrow \& B A \mid \varepsilon$$

$$B \rightarrow \text{TRUE} \mid \text{FALSE}$$

where $\{E, A, B\}$ is the set of nonterminal symbols, E is the start symbol, $\{\&, \text{TRUE}, \text{FALSE}\}$ is the set of terminal symbols, and ε denotes the empty string. The grammar is LL(1).

- (a) Construct the LL(1) parse table for this grammar.

- (b) Trace the stack and input contents for a parse of the string `TRUE & FALSE & TRUE`.

3. Indicate (with “yes” or “no”) if the following grammars fall into the listed grammar types and show why:

(a) $S \rightarrow Aa \mid Bb$
 $A \rightarrow c$
 $B \rightarrow c$

LL(1): No

LL(2): Yes

(b) $S \rightarrow Aa \mid Bb$
 $A \rightarrow cA \mid a$
 $B \rightarrow cB \mid b$

LR(1) : Yes

LL(k) : No

(c) $S \rightarrow Aca \mid Bcb$
 $A \rightarrow c$
 $B \rightarrow c$

LL(2): No

LR(1): No

4. Consider the following context-free grammar:

$$S \rightarrow Sa$$

$$S \rightarrow bS$$

$$S \rightarrow c$$

(a) Show that the grammar is ambiguous.

There are two leftmost derivations for bca :

$$S \Rightarrow Sa \Rightarrow bSa \Rightarrow bca$$

$$S \Rightarrow bS \Rightarrow bSa \Rightarrow bca$$

(b) Write the canonical collections of LR(1) items for this grammar.¹

State 0:

$$S' \rightarrow \cdot S, \$$$

$$S \rightarrow \cdot Sa, \$a$$

$$S \rightarrow \cdot bS, \$a$$

State 1:

$$S \rightarrow \cdot c, \$a$$

State 2:

$$S' \rightarrow S \cdot, \$$$

$$S \rightarrow S \cdot a, \$a$$

State 3:

$$S \rightarrow Sa \cdot, \$a$$

State 4:

$$S \rightarrow b \cdot S, \$a$$

$$S \rightarrow \cdot Sa, \$a$$

$$S \rightarrow \cdot bS, \$a$$

$$S \rightarrow \cdot c, \$a$$

State 5:

$$S \rightarrow bS \cdot, \$a$$

$$S \rightarrow S \cdot a, \$a$$

¹ No, you do not have to modify the grammar to be non-ambiguous to do this.

- (c) Identify all conflicting items, and the types of the conflicts (e.g., “shift-reduce conflict in state 3 on d”).

There is a shift/reduce conflict on a in state:

$$\begin{aligned} S &\rightarrow bS\cdot, \$a \\ S &\rightarrow S\cdot a, \$a \end{aligned}$$

This stems from the ambiguity in the grammar. E.g., should bca be parsed as $(bc)a$ (reduce at this state) or $b(ca)$ (shift the a).

- (d) Rewrite the grammar in an equivalent form that is suitable for LR parsing (i.e., does not result in conflicts). To do so, consider the strings generated by the grammar—what is their characteristic?

$$\begin{aligned} S &\rightarrow BcA \\ B &\rightarrow Bb \mid \varepsilon \\ A &\rightarrow Aa \mid \varepsilon \end{aligned}$$

- (e) Using your modified grammar, show the sequence of stack contents and inputs when parsing the input $bbcaa$.

5. Given the grammar:

$$S \rightarrow E$$

$$E \rightarrow E + E \mid (E) \mid \text{ID} \mid \text{NUM} \mid E = E$$

where S and E are non-terminals and all the other grammar symbols are terminals. Assume that the above grammar represents expressions as in C such as

$$a = (b = c)$$

$$a = (b = c + 1)$$

- (a) Extend the above grammar with attributes such that S has a boolean synthesized attribute which is TRUE iff the LHS of any assignments derived from S are simple IDs. So for all of the examples above, the attribute should be TRUE, but for $(a + b) = c$ or $(a = b) = c$ the attribute should be FALSE. Note that a statement like $(a) = 1$ should also yield TRUE.

The approach is straightforward. We add boolean synthesized attribute *isLValueOK* to both S and E , as well a boolean synthesized attribute *isLValue* to E . This results in the following grammar:

```
S(boolean $isLValueOk)
: E($isLValue, $isLValueOK)
;
E(boolean $isLValue, boolean $isLValueOK)
: E($isLValue1, $isLValueOK1) + E($isLValue2, $isLValueOK2)
  { E.isLValue= false; $isLValueOK= $isLValueOK1 && $isLValueOK2; }
| ( E($isLValue, $isLValueOK) )
| ID
  { $isLValue= $isLValueOK= true; }
| NUMBER
  { $isLValue= false; $isLValueOK= true; }
| E($isLValue1, $isLValueOK1) = E($isLValue2, $isLValueOK2)
  { $isLValue= false;
    $isLValueOK= $isLValue1 && $isLValueOK1 && $isLValueOK2;
  }
;
```

- (b) List the attributes in your extended grammar and indicate whether each is synthesized or inherited. Explain your answer.

(c) Show the parse tree for the string $a = (b = c)$ and decorate the tree with the appropriate attributes and values.

(d) Show the parse tree for the string $(a + b) = c$ and decorate the tree with the appropriate attributes and values.

,