Principles of Programming Languages

CS 314

Recitation 8

Midterm Review

Topics Today

- Homework Review
- Complementary Examples

Describe Regular Expression

3(c) (00|11)* ((01|10)(00|11)*(01|10)(00|11)*)*

Wrong Answer:
binary strings that are even in decimal base.

example: 11

Correct Answer:
binary strings contain even numbers of 0’s and 1’s

Write Regular Expression

4(a) All strings of a’s, b’s, and c’s that contain no a’s following any b’s.

Wrong Answer:
(a*(bc|c)*)*

bca
There is still an a following an b, even though the a doesn’t immediately follow b

Correct Answer:
(a|c)∗(b|c)∗

Write Regular Expression

4(b) All strings of a’s, b’s, and c’s that do not contain more than 2 a's and 2 b’s.

Wrong Answer:

c*(a|ε)c*(a|ε)c*(b|ε)c*(b|ε)c*|
c*(a|ε)c*(b|ε)c*(a|ε)c*(b|ε)c*|
c*(b|ε)c*(a|ε)c*(b|ε)c*(a|ε)c*|
c*(b|ε)c*(b|ε)c*(a|ε)c*(a|ε)c*

Make sure you have all permutation

Correct Answer:

c*(a|ε)c*(a|ε)c*(b|ε)c*(b|ε)c*|
c*(a|ε)c*(b|ε)c*(a|ε)c*(b|ε)c*|
c*(a|ε)c*(b|ε)c*(b|ε)c*(a|ε)c*|
c*(b|ε)c*(a|ε)c*(a|ε)c*(b|ε)c*|
c*(b|ε)c*(a|ε)c*(b|ε)c*(a|ε)c*|
c*(b|ε)c*(b|ε)c*(a|ε)c*(a|ε)c*

Write Grammar in BNF Notation

2(a) $\{a^m b^n c^o | m>n\geq 0, o>0\}$, with alphabet $\Sigma = \{a, b, c\}$

Wrong Answer:

```
<S>::=<A><B><C>
<A>::=a<A>|a
<B>::=<B>b|ε
<C>::=c<C>|c
```

Violate the constraint m>n

Correct Answer:

```
<S>::=<A><B><C>
<A>::=a<A>|a
<B>::=a<B>b|ε
<C>::=c<C>|c
```

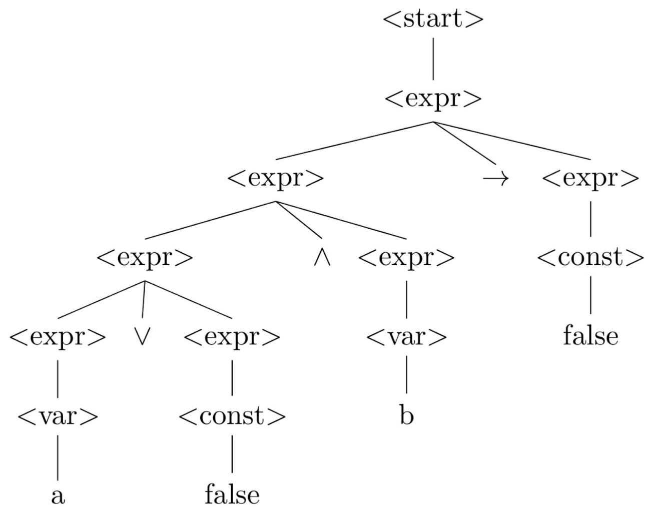
Abstract Syntax Tree

3(c) Give the corresponding abstract syntax tree (AST)

<start> ::= <expr>
<expr> ::= <expr>∨<expr>|<expr>∧<expr>|<expr>→<expr>|<const>|<var>
<const> ::= true|false
<var> ::= a|b|c|. . .|z

Wrong Answer:

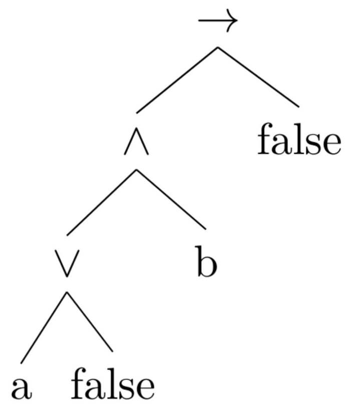
Right-most parse tree:



Parse tree is not AST

Correct Answer:

AST for right-most derivation:



FRIST Sets And PREDICT Sets

FIRST sets are for sequences of symbols while PREDICT sets are for rules.

Wrong Answer:

PREDICT(<morevars>)=...
FIRST(<morevars>::=ϵ)=...

Correct Answer:
FIRST(<morestmts>)={\n,ϵ}
FIRST(\n<stmtlist>)={\n}
PREDICT(<morevars>::=ϵ)=
FOLLOW(<morevars>)={)}

Recursive Descent parser

Remember to return value or raise exception to reject an input

Wrong Answer:

```
bool funcname() {
    switch(token) {
        case f:
        case g:
                token = next_token();
        default:
    }
}
```

Correct Answer:

```
bool funcname() {
    switch(token) {
        case f:
        case g:
                token = next_token();
                return true;
        default:
                return false;
    }
}
```

C++ Memory Management

3 Deallocate singly-linked list

Wrong Answer:

```
current_cell = head;
while (1)
{
    list_cell *temp = current_cell->next;
    free(current_cell);
    current_cell = temp;
    if (current_cell == NULL)
        break;
}
```

head can be NULL!

Correct Answer:

```
for (current_cell = head; current_cell != NULL;;)
{
    list_cell *temp = current_cell->next;
    free(current_cell);
    current_cell = temp;
}
```

Lexical And Dynamic Scoping

1 what is the output?

```
procedure main():
    int var = 10;
    procedure set_var(int val):
          var = val;
    end set_var
    procedure proc1():
          set_var(1);
    end proc1
    procedure proc2():
          int var = 2;
          set_var(4);
          print var;
    end proc2
    print var;
    set_var(41);
    proc1();
    print var;
    proc2();
end main
```

Lexical: 10, 1, 2

```
procedure main():
    int var = 10;
    procedure set_var(int val):
          var = val;
    end set_var
    procedure proc1():
          set_var(1);
    end proc1
    procedure proc2():
          int var = 2;
          set_var(4);
          print var;
    end proc2
    print var;
    set_var(41);
    proc1();
    print var;
    proc2();
end main
```

Dynamic: 10, 1, 4

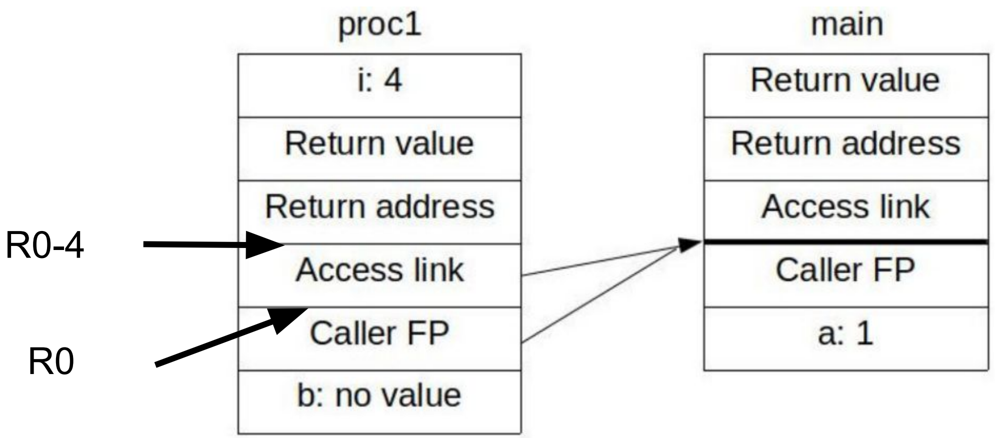
```
procedure main():
    int var = 10;
    procedure set_var(int val):
          var = val;
    end set_var
    procedure proc1():
          set_var(1);
    end proc1
    procedure proc2():
          int var = 2;
          set_var(4); (use var)
          print var;
    end proc2
    print var;
    set_var(41); (use var)
    proc1(); (use var)
    print var;
    proc2();
end main
```

Dynamic!
It can be different var for different call

Stack Frame

2(c) RISC instructions for b=a+1;

```
procedure main():
    int a;
    procedure proc1(int i):
      int b;
      b = a + 1;
```



proc1

i: 4

Return value

Return address

Access link

Caller FP

b: no value

main

Return value

Return address

Access link

Caller FP

a: 1

Correct Answer:

```
LOADI R1, #-4;
ADD R2, R0, R1; //main's access pointer
LOAD R3, R2; //main
LOADI R4, #4;
ADD R5, R3, R4; //address of a
LOAD R6, R5
```

Wrong Answer:

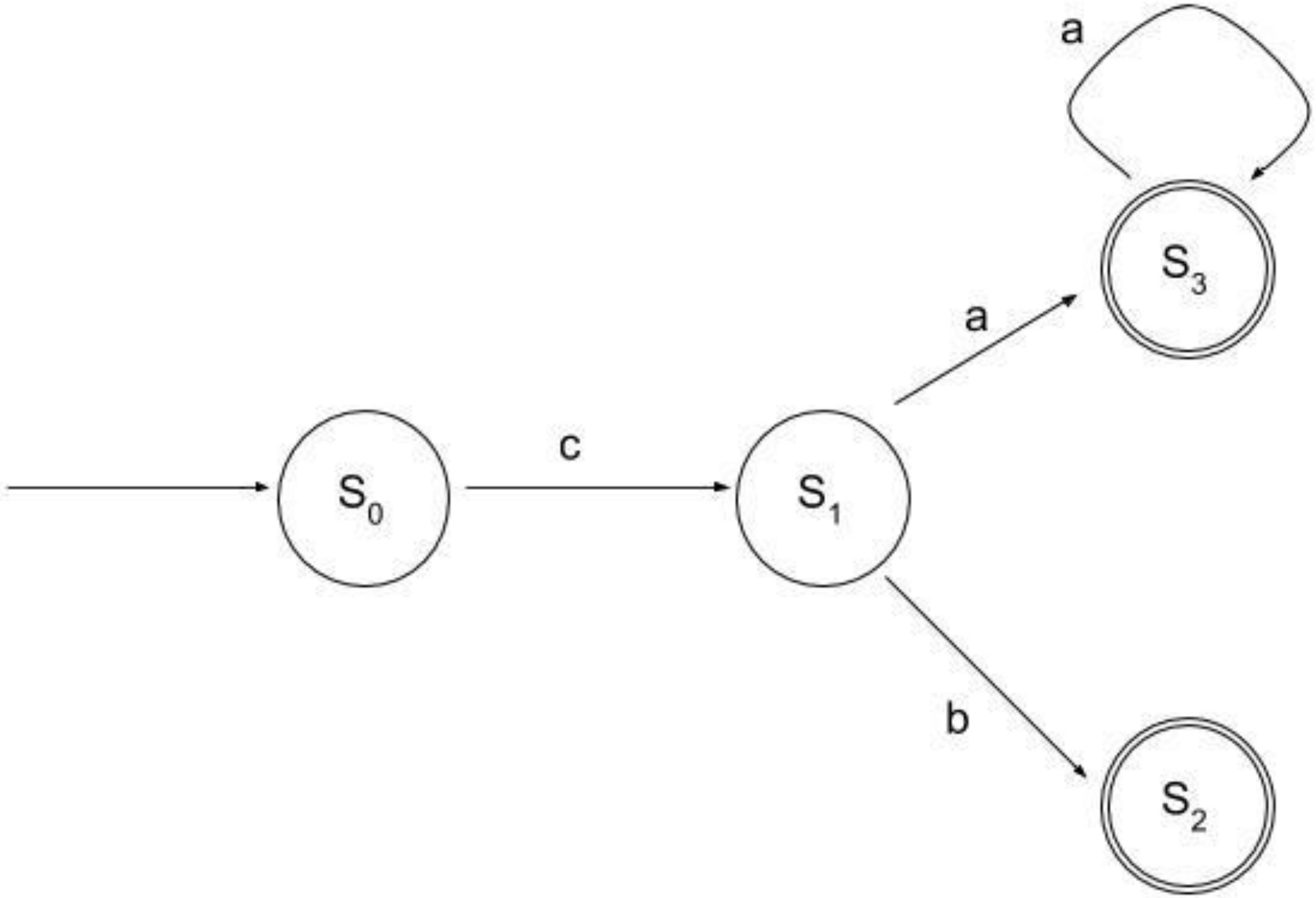
```
LOAD R3, R0;
LOADI R4, #4;
ADD R5, R3, R4; //address of a
LOAD R6, R5
```

Should look at R0-4 for the access link

DFA-Example

Create DFA For Regular expressions

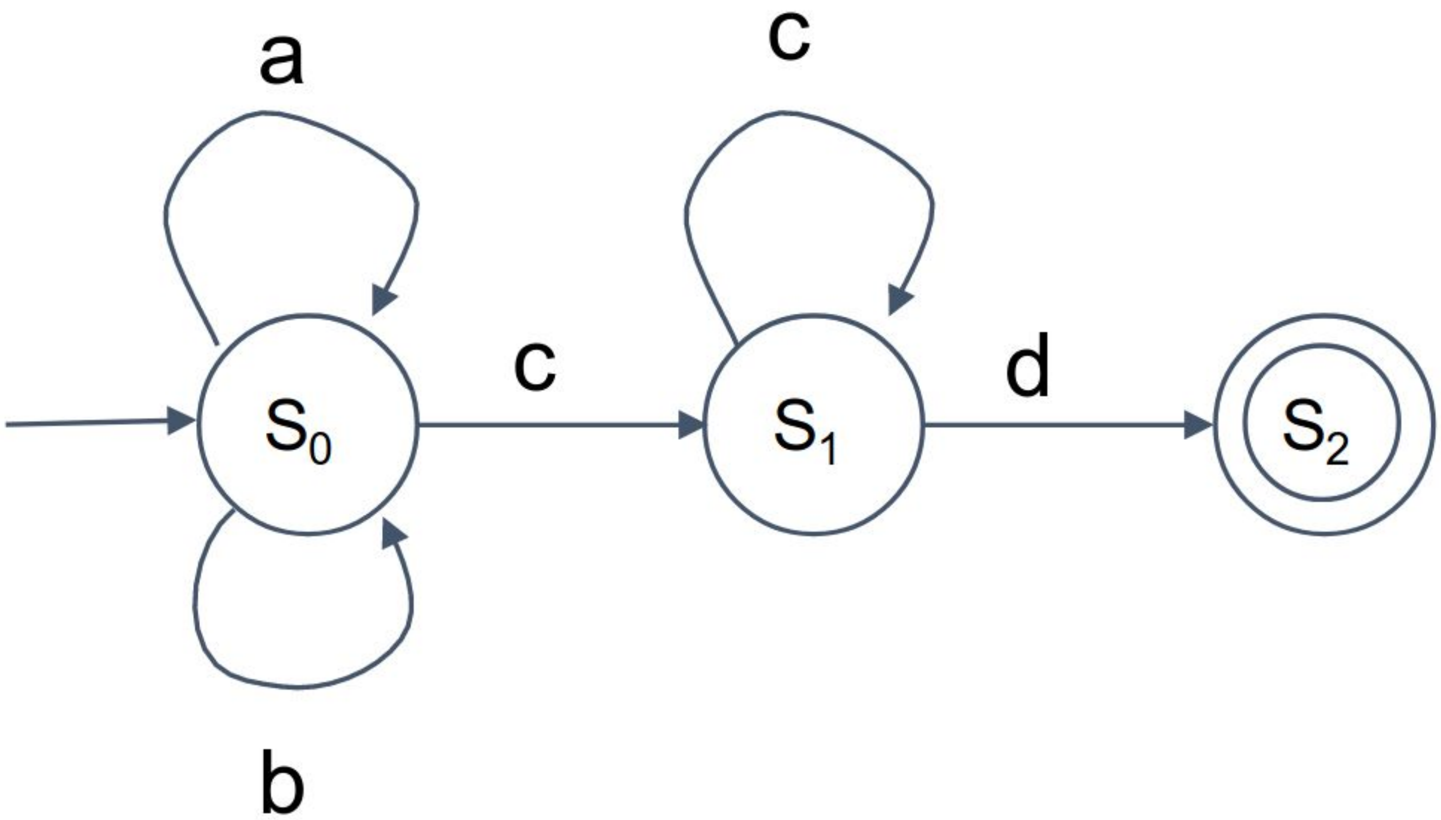
$c(b|a^{+})$



DFA-Example

Create DFA For Regular expressions

$(a|b)^*c^+d$



Ambiguity-Example

Is the grammar ambiguous?

S->aSA|∈
A->bA|∈

Two leftmost derivation for aabb

Derivation 1:
S => aSA
=> aaSAA
=> aaAA
=> aabAA
=> aabbAA
=> aabbA
=> aabb

Derivation 2:
S => aSA
=> aaSAA
=> aaAA
=> aabAA
=> aabA
=> aabbA
=> aabb