Faculty of Computer & Information Sciences
Ain Shams University
Subject: CSC360
Compilers' Theory

Examiners: Prof. Zaki Taha
Dr. Sally Saad
Academic year: 1st term 2021-2022
Year: 3ed undergraduate

Compilers' Theory
Milestone -1-

1. The "TINY" Language Regular Expressions:

1) Number:

Digit := [0-9]
Num_Un_Signed := (Digit)+
Num_Signed := (+|-)? Num_Un_Signed
Num_float := Num_Signed (\.Num_Un_Signed)?

2) String:

Letter = [a-z]|[A|Z]
Str := ^\".*\"$

3) Reserved_Keywords:

R_Keywords := int|float|string|read|write|repeat|until|if|elseif|else|then|return|end

4) Comment_Statement:

L_Comment := ^/*.**/$

5) Identifiers:

identifier := Letter(Letter|Digit)*

6) Function_Call:

Fun_call := identifier \(((identifier)(, identifier)*)? \)

7) Term:

Term := (Num_float | identifier | Fun_call)

8) Arithmatic_Operator:

Arth_op = (+ | - | * | /)

9) Equation:

E_unit = (Term+ Arth_op)*(Term+)$
Equ = E_unit | (Term Arth_op)* \(E_unit \)(Arth_op Term)*

10) Expression:

Exp := Term|Str|Equ

11) Assignment_Statement:

Ass_st:= (identifier := Exp)

12) Datatype:

Datatype := (int|float|string)

13) Declaration_Statement:

Dec_st := ^Datatype identifier (,identifier|,Ass_st)*;$

14) Write_Statement:

Write_st:= ^ write (EXp|\n) ;$

15) Read_Statement:

Read_st:= ^ read identifier ;$

16) Return_Statement:

Return_st:= ^ return Exp ;$

17) Condition_Operator:

Con_op:= (<|>|=|<>)

18) Condition:

Con:= (identifier Con_op term)

19) Boolean_Operator:

Boolean_Op:= (&& | ||)

20) Condition_Statement:

Condition (Boolean_Operator Condition)*

21) Set_of_Statements

Set_of_Statements := (Assignment_Statement | Declaration_Statement | Write_Statement | Read_Statement | (Return_Statement)? | Function_Call)

22) If_Statement:

If_Statement := “if” Condition_Statement “then” Set_of_Statements (Else_If_Statement | Else_Statement | end)

23) Else_If_Statement:

Else_If_Statement := “elseif” Condition_Statement then Set_of_Statements (Else_If_Statement | Else_Statement | “end”)

24) Else_Statement:

Else_Statement := ”elseif” Condition_Statement “then” Set_of_Statements (Else_If_Statement | Else_Statement | “end”)

25) Repeat_Statement:

Repeat_Statement := “repeat” Set_of_Statements “until” Condition_Statement

26) FunctionName:

FunctionName := Identifier

27) Parameter:

Parameter := Datatype Identifier

28) Function_Declaration:

Function_Declaration :=

Datatype FunctionName \((Parameter(,Parameter)*)? \)

29) Function_Body:

Function_Body := { Set_of_Statements (Return_Statement) }

30) Function_Statement:

Function_Statement:= Function_Declaration Function_Body

31) Main_Function:

Main_Function := Datatype “main“ \(\ ) Function_Body

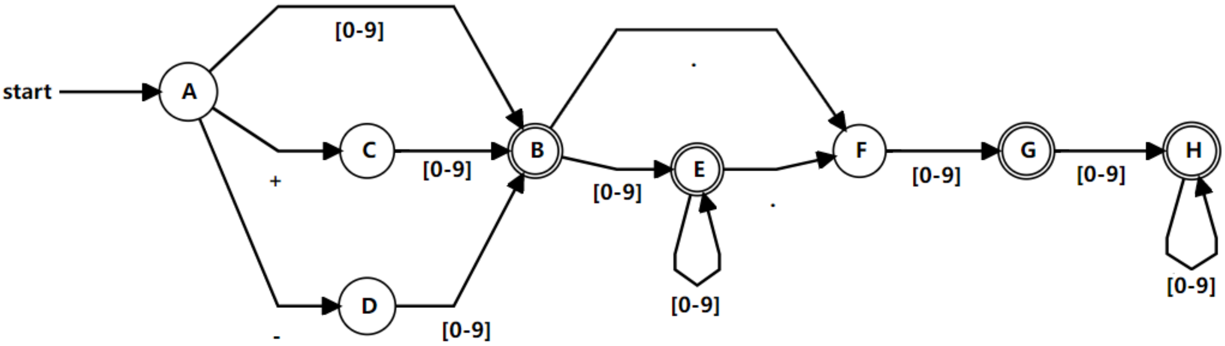
32) Program:

Program := (Function_Statment)* Main_Function

Compilers' Theory
Milestone -1-

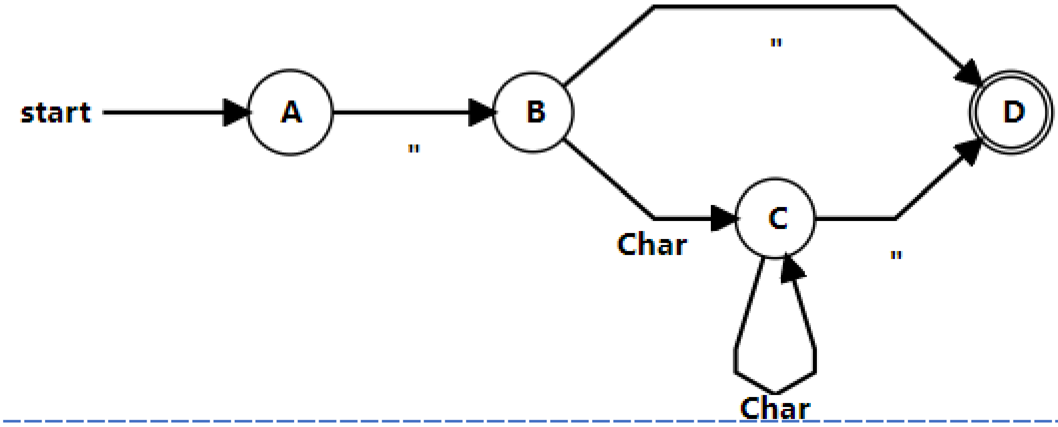
2. TINY DFAs

1 - Number:



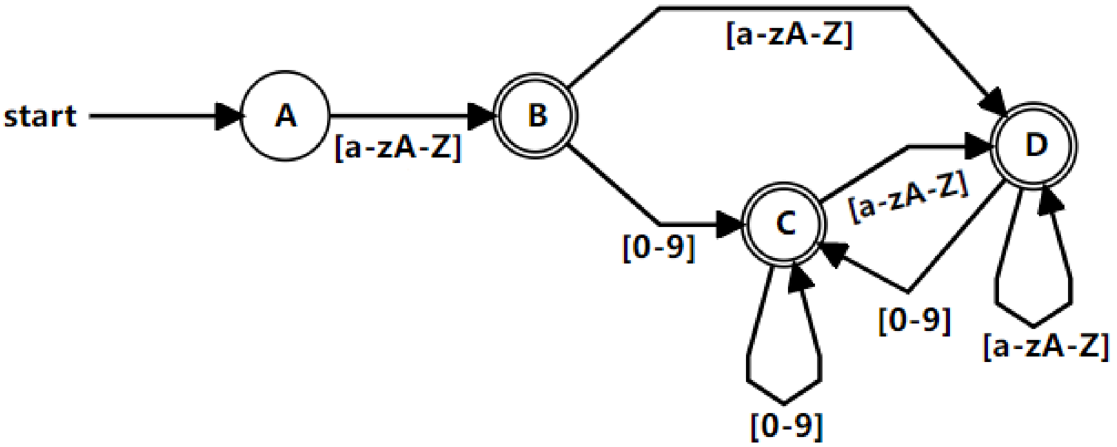
===

2) String:



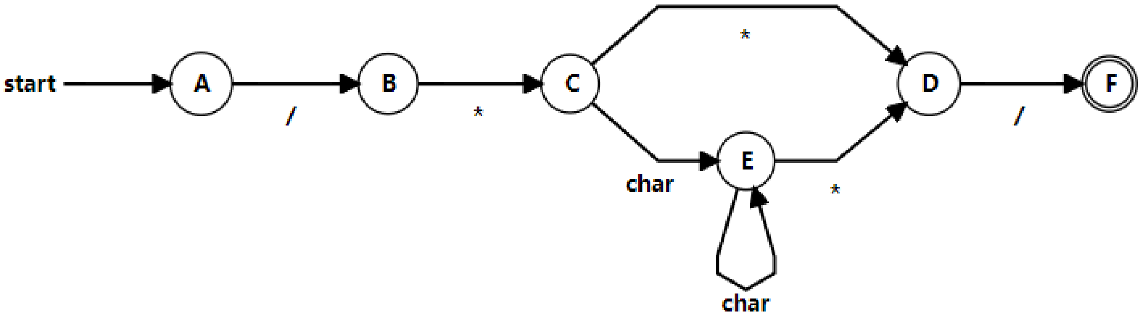
===

3) Identifiers:



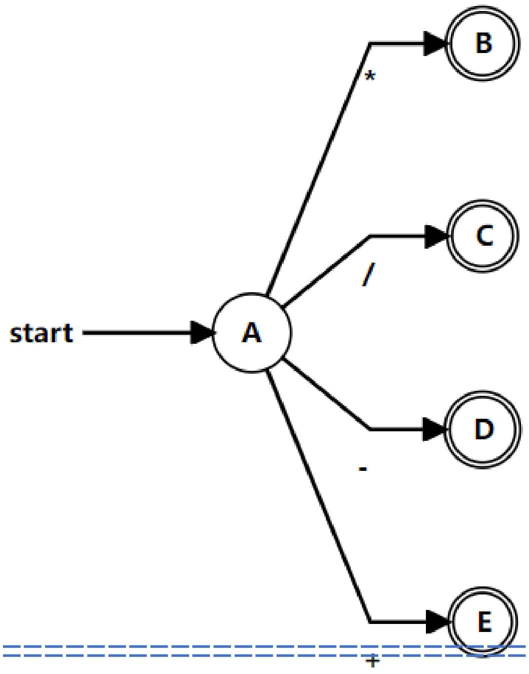
==

4) Comment statement :



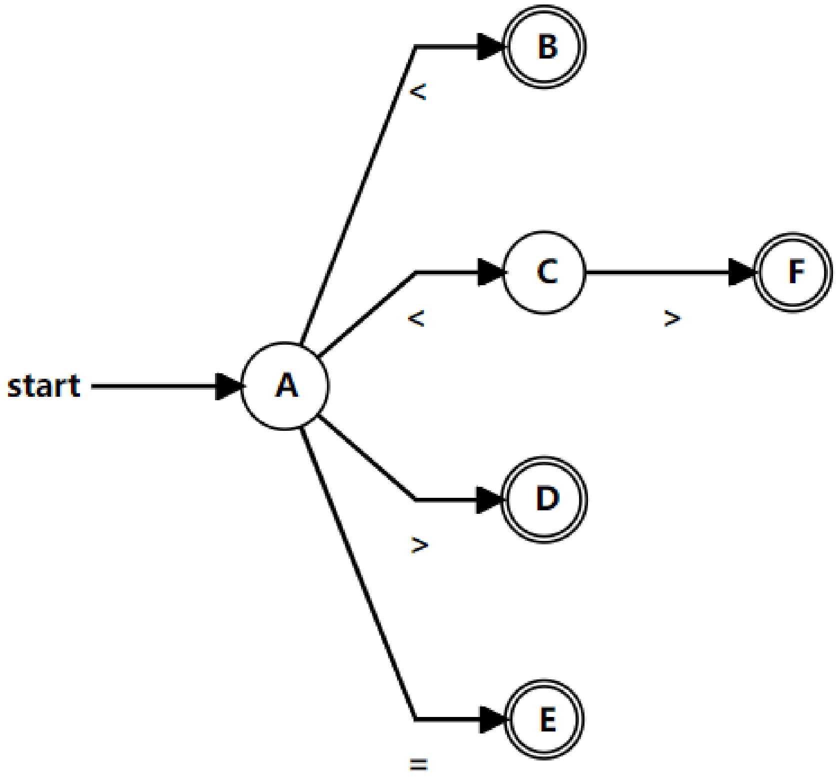
==

5) Arithmetic Operators:



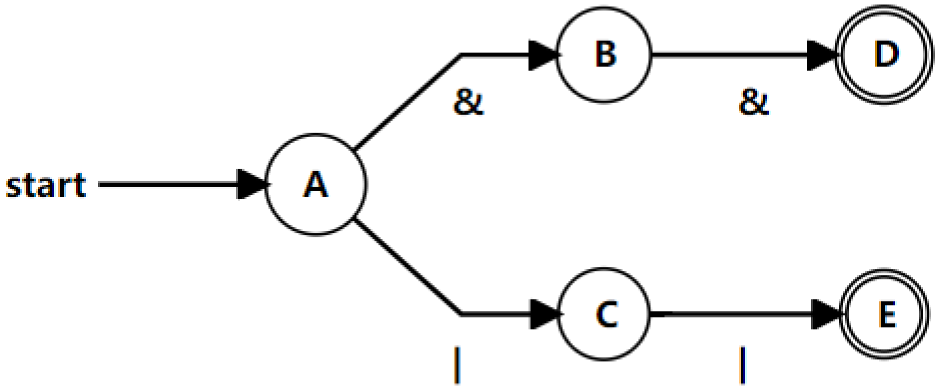
==

6) Condition Operators :



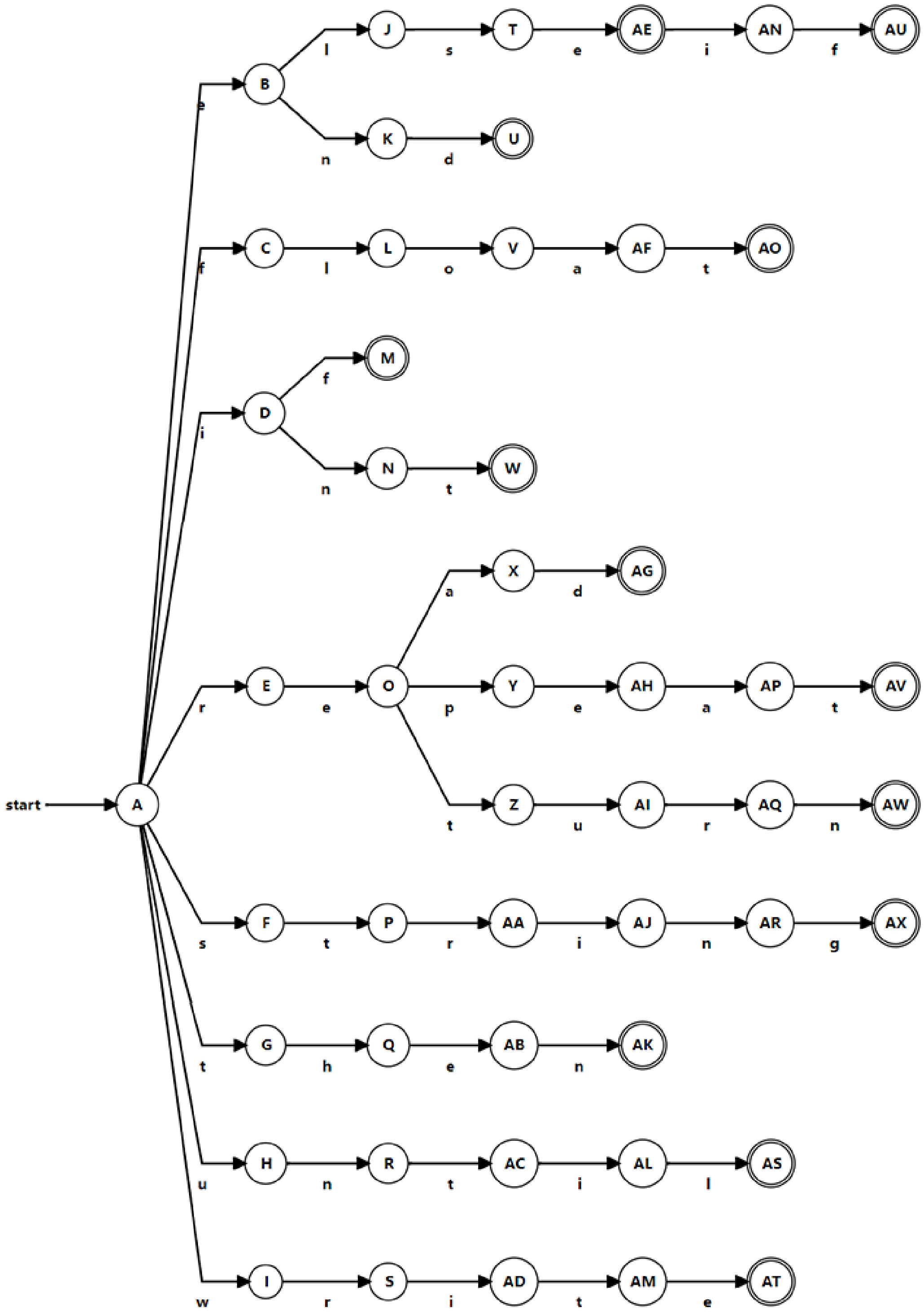
==

7) Boolean Operator



==

8) Reserved Words



start
A
e
B
l
J
s
T
e
AE
i
AN
f
AU
n
K
d
U
f
C
l
L
o
V
a
AF
t
AO
i
D
f
M
n
N
t
W
r
E
e
O
a
X
d
AG
p
Y
e
AH
a
AP
t
AV
t
Z
u
AI
r
AQ
n
AW
s
F
t
P
r
AA
i
AJ
n
AR
g
AX
t
G
h
Q
e
AB
n
AK
u
H
n
R
t
AC
i
AL
l
AS
w
I
r
S
i
AD
t
AM
e
AT

Faculty of Computer & Information Sciences
Ain Shams University
Subject: CSC360
Compilers’ Theory

Examiners: Prof. Zaki Taha
Dr. Sally Saad
Academic year: 1st term 2021-2022
Year: 3ed undergraduate

Compilers’ Theory

Milestone -2-

Note: ***Terminal_tokens*** **are written in bold and italic. non_Terminal_tokens are written in Bold.**

1. **Program → UserFunc MainFunc.**

2. **UserFunc → Function |** ***ε***

3. **MainFunc → Datetype** ***main ()*** **Body**

4. **Function → Fun_dec Body UserFunc**

5. **Fun_dec → Datatype** ***identifier*** **ArgList**

6. **Datatype →** ***int*** **|** ***float*** **|*****string***

7. **ArgList →** ***(*****Arguments*****)*** **|** ***()***

8. **Arguments → Arguments*****, identifier*** **|** ***identifier*** *(left recursive)*
   - **Arguments → Datatype** ***identifier*** **Arg**
   - **Arg →, Datatype** ***identifier*** **Arg |** ***ε***

9. **Body →** ***{*****Stat_seq return-stmt** ***}***

10. **Stat_Seq → Stat_Seq ; Statement | Statement** *(left recursive)*
    - **Stat_Seq → Statement State**
    - **State →*****;*** **Statement State |** ***ε***

11. **Statement → if-stmt | repeat-stmt | assign-or-funcallstmt | read-stmt | write-stmt| Decl-stmt| return-stmt**

12. **if-stmt →** ***if*** **Condition** ***then*** **Stat_Seq ElseClosure**

13. **elseif-stmt → *elseif* Condition *then* Stat_Seq ElseClosure**

14. **ElseClosure → *else* Stat_Seq *end* | elseif-stmt**

15. **Condition → Expression RelOp Expression ConditionClosure**

16. **ConditionClosure → ConditionOps Condition | ε**

17. **assign-or-funcallstmt → assign-stmt| fun-call** *(left factoring)*
    - **assign-or-funcallstmt → *identifier* A**
    - **A → fun-call|assign-stmt**

18. **Equation → Equation AddOp Term | Term** *(left recursive)*
    - **Equation → Term Equ**
    - **Equ → AddOp Term Equ | ε**

19. **Term → Term MultOp Factor | Factor** *(left recursive)*
    - **Term → Factor Ter**
    - **Ter → MultOp Factor Ter | ε**

20. **Factor → *constant* |*identifier* |FunCall** *(left factoring)*
    - **Factor → *constant* |*identifier* A| (Expression)**
    - **A → fun-call | ε**

21. **RelOp → <|>|=|<>**

22. **CondationOps → *“||”* | *&&***

23. **AddOp → + | -**

24. **MultOp → * | /**

25. **Expression → *String* | Term | Equation** *(left factoring)*
    - **Expression → *String* |exp**

- **exp → Term E**
- **E → Equ|ε**

26. repeat-stmt → *repeat* Stat_Seq *until* Expression

27. assign-stmt → *identifier:=* Expression

28. read-stmt → *read identifier*

29. write-stmt → *write* Expression

30. Decl-stmt → DataType Id

31. Id → *identifier* | assign-stmt IdClosure *(left factoring)*

  - **Id → *identifier* B IdClause**
  - **B → assign-stmt | ε**

32. IdClause → , Id | ε

33. fun-call → callArgList

34. CallArgList → *(*ArgumentsCall*)* | *()*

35. ArgumentsCall → ArgumentsCall, *identifier* | *identifier* ***(left recursive)***

  - **ArgumentsCall → *identifier* ArgCall**
  - **ArgCall →, *identifier* ArgCall | ε**

36. return-stmt → *return* Expression