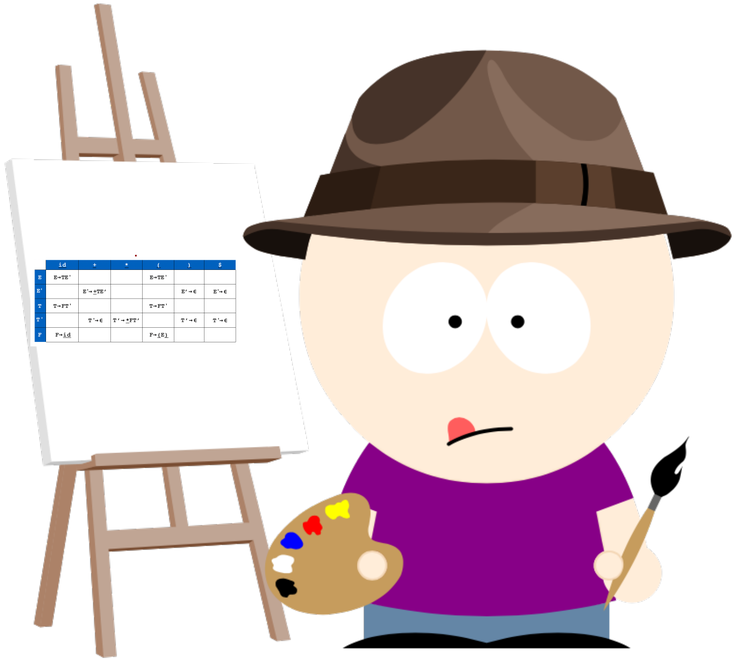


In-Class Exercises

Version: 2026-01-03

Parsing 3- Predictive Parsing

Download exercises

1. Go to

   https://ligerlabs.org/compilers.html

2. Download the file

   `parse-3-exercises.zip`

3. Open up a terminal and Unzip the file

   ```
   > unzip parse-3-exercises.zip
   > cd parse-3-exercises
   > ls
   ```

Task 1

- For this grammar
  1. Generate the predictive parse table
  2. Show the actions of the interpreter for the input strings "xyz" and "y".

1. S → A B C
2. A → x
3. A → ϵ
4. B → y
5. C → z
6. C → ϵ

FIRST(S) = {x, y, ε}
FIRST(A) = {x, ε}
FIRST(B) = {y}
FIRST(C) = {z, ε}

FOLLOW(S) = {$}
FOLLOW(A) = {y}
FOLLOW(B) = {z,$}
FOLLOW(C) = {$}

Task 2

- For this grammar
  1. Generate the predictive parse table
  2. Show the actions of the interpreter for the input strings "yyy".

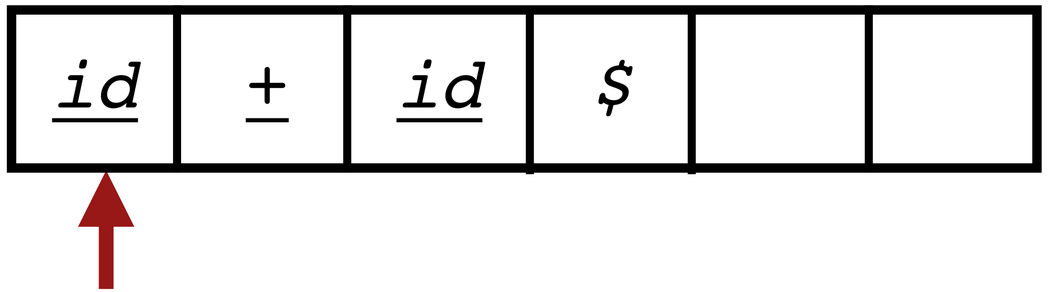
```
1.S → yAB
2.A → y
3.A → z
4.B → y
5.B → ϵ
```

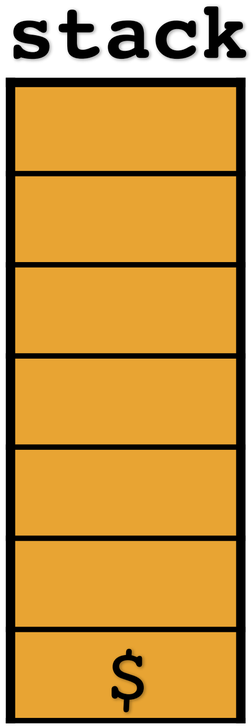
```
FIRST(S) = {y}
FIRST(A) = {y,z}
FIRST(B) = {y,ϵ}

FOLLOW(S) = {$}
FOLLOW(A) = {y,$}
FOLLOW(B) = {$}
```

Task 3

- Show the actions of the interpreter on the expression grammar for this input.

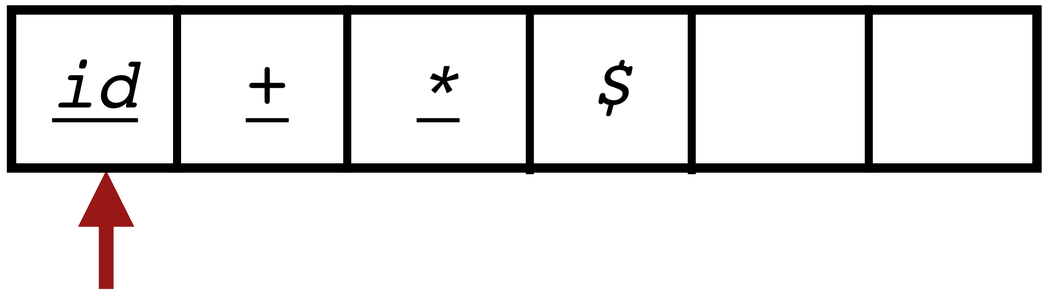


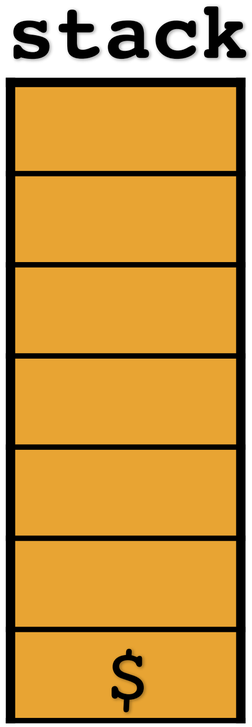


	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→∈	E'→∈
T	T→FT'			T→FT'		
T'		T'→∈	T'→*FT'		T'→∈	T'→∈
F	F→id			F→(E)		

Task 4

- Show the actions of the interpreter on the expression grammar for this input.





	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→∈	E'→∈
T	T→FT'			T→FT'		
T'		T'→∈	T'→*FT'		T'→∈	T'→∈
F	F→id			F→(E)		

Task 5

- For this grammar
  1. Compute FIRST and FOLLOW
  2. Generate the predictive parse table
- Do you notice something weird about this parse table? Explain!
- This grammar models C's if-then-else statement: `i`=`if`, `t`=`then`, `e`=`else`, and `A` represents the options `else` part.

1. S → iEtSA
2. S → x
3. A → eS
4. A → ϵ
5. E → y

Task 6

- For this grammar
  1. Compute FIRST and FOLLOW
  2. Generate the predictive parse table

1. S A B C
2. S → ϵ
3. A → A S
4. A → ϵ
5. B → x B y
6. B → ϵ
7. C → z C
8. C → w

Task 7

- For this grammar
  1. Compute FIRST and FOLLOW
  2. Generate the predictive parse table
  3. Show the actions the parser takes on the input "bcba"

1. S → x A x
2. S → B A x
3. S → ϵ
4. A → z A
5. A → y A
6. A → ϵ
7. B → y

Algorithms

Computing FIRST Sets (no $A \rightarrow \epsilon$ rules)

1. **FOR** each non-terminal A **DO**

   FIRST(A) = {}

2. **FOR** each terminal t **DO**

   FIRST(t) = {t}

3. **REPEAT** until no more changes:

   FOR each production $A \rightarrow Y_1 \ldots Y_k$ **DO**

   FIRST(A) ∪= FIRST(Y_1);

Computing FIRST Sets (with A→ϵ rules)

1. **FOR** *each non-terminal A* **DO** FIRST(*A*) = {}
2. **FOR** *each terminal t* **DO** FIRST(*t*) = {*t*}
3. **FOR** *each production A → ϵ* **DO** FIRST(*A*) = {ϵ}
4. **REPEAT** until no more changes:

```
    FOR each production A → Y1 ... Yk except A→ϵ DO
        FIRST(A) ∪= FIRST(Y1) - {ϵ};
        FOR i = 1 to k−1 DO
            IF ϵ is in FIRST(Y1) ∧ ... ∧ ϵ is in FIRST(Yi) THEN
                FIRST(A) ∪= FIRST(Yi+1) - {ϵ};
        IF ϵ is in FIRST(Y1) ∧ ... ∧ ϵ is in FIRST(Yk) THEN
            FIRST(A) ∪= {ϵ};
```

Computing FOLLOW Sets

1. **FOR** each non-terminal A **DO** FOLLOW(A) = {}
2. FOLLOW(S) = {$}

REPEAT until no more changes:

3. **FOR** each production $A \rightarrow \alpha B \beta$ **DO**

   FOLLOW(B) ∪= (FIRST(β) – {ϵ})

4. **FOR** each production $A \rightarrow \alpha B$ **DO**

   FOLLOW(B) ∪= FOLLOW(A)

5. **FOR** each production $A \rightarrow \alpha B \beta$ **WHERE** ϵ is in FIRST(β) **DO**

   FOLLOW(B) ∪= FOLLOW(A)

Predictive Parsing Interpreter

```
curr_tok := first token
push($); push(start symbol)
REPEAT
    X := top()
    IF X is curr_tok terminal OR $ THEN
        IF X=curr_tok THEN
            pop()
            curr_tok := next token
        ELSE
            error
    ELSE
        IF M[X,curr_tok]=X→Y1 Y2 ... Yk THEN
            pop()
            push(Yk); ... push(Y1)
        ELSE error
UNTIL X=$
```

Case 3

Case 2

Case 4

Case 1

Building the Parse Table

```
FOR each production A→α DO
    FOR each terminal x in FIRST(α) DO
        M[A,x] ∪= {A→α}

    IF ∈ is in FIRST(α) THEN
        FOR each terminal x in FOLLOW(A) DO
            M[A,x] ∪= {A→α}

    IF ∈ is in FIRST(α) AND $ is in FOLLOW(A) THEN
        M[A,$] ∪= {A→α}

FOR all undefined entries M[A,x] DO
    M[A,x] := error
```

collberg.cs.arizona.edu

ligerlabs.org

Supported by

NSF SATC/TTP-1525820

SATC/EDU-2029632

Christian Collberg

University of Arizona