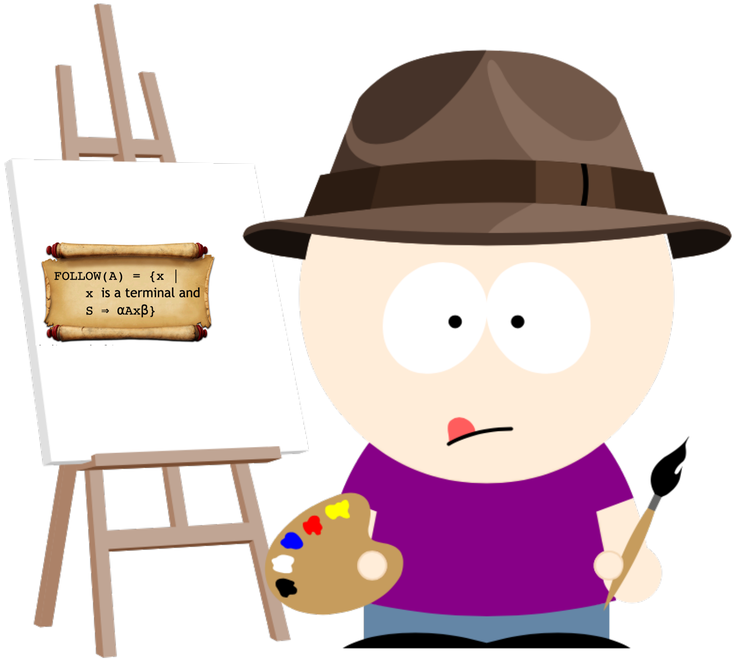


In-Class Exercises

Version: 2026-01-02

Parsing 2- Top-Down Parsing

Download exercises

1. Go to

   https://ligerlabs.org/compilers.html

2. Download the file

   `parse-2-exercises.zip`

3. Open up a terminal and Unzip the file

   ```
   > unzip parse-2-exercises.zip
   > cd parse-2-exercises
   > ls
   ```

Task 1

- For this grammar
  1. Compute FIRST and FOLLOW for all non-terminals
  2. Provide derivations for S, A, and B that show that your results are correct.

1. S → A B S
2. A → x
3. B → y
4. S → y

Task 2

- Compute FIRST for all non-terminals:

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

Task 3

- For this grammar
  1. Compute FIRST for all non-terminals
  2. Provide derivations that show that your results are correct.

1. $A \rightarrow \underline{x}\ B$
2. $A \rightarrow \underline{y}$
3. $A \rightarrow \epsilon$
4. $B \rightarrow \underline{y}$

Task 4

- For this grammar compute FIRST and FOLLOW for all non-terminals:

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

Task 5

- For this grammar compute FIRST for all non-terminals:

1. S → x S y
2. S → y x
3. S → ϵ

Task 6

- For this grammar compute FIRST for all non-terminals:

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

Task 7

- For this grammar compute FIRST for all non-terminals:

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

Task 8

- For this grammar compute FIRST and FOLLOW for all non-terminals:

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

Task 9

- For this grammar compute FIRST for all non-terminals:

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

Task 10

- For this grammar
  1. Remove left recursion
  2. Compute FIRST for all non-terminals
  3. Construct transition diagrams for all non-terminals

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

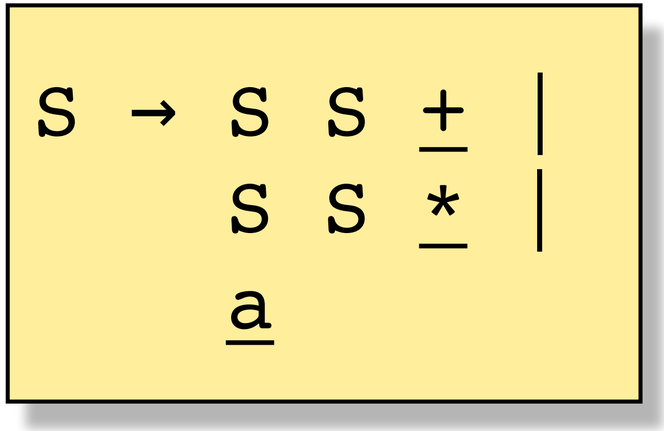
Task 11

- For this grammar compute FIRST and FOLOW for all non-terminals

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

Task 12

- Consider this grammar:



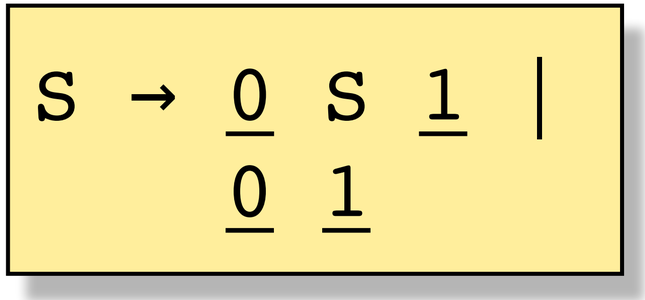
1. Compute FIRST
2. Compute FOLLOW

Dragonbook, Exercise 4.4.3

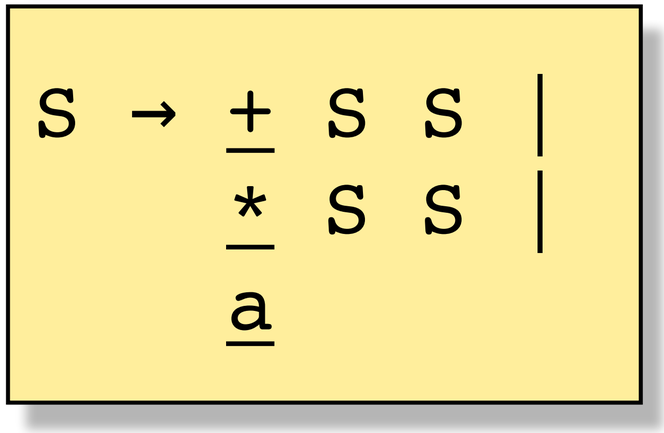
Task 13

- Compute FIRST and FOLLOW for the following grammars:

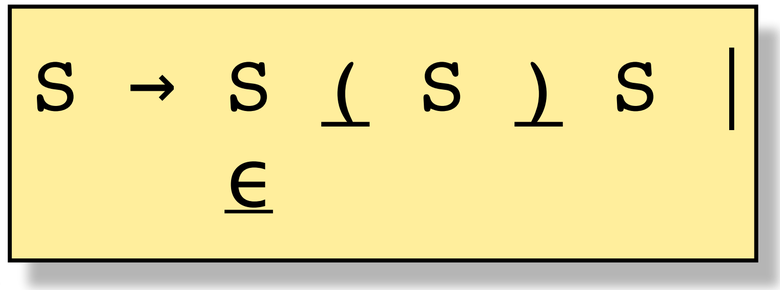
1.



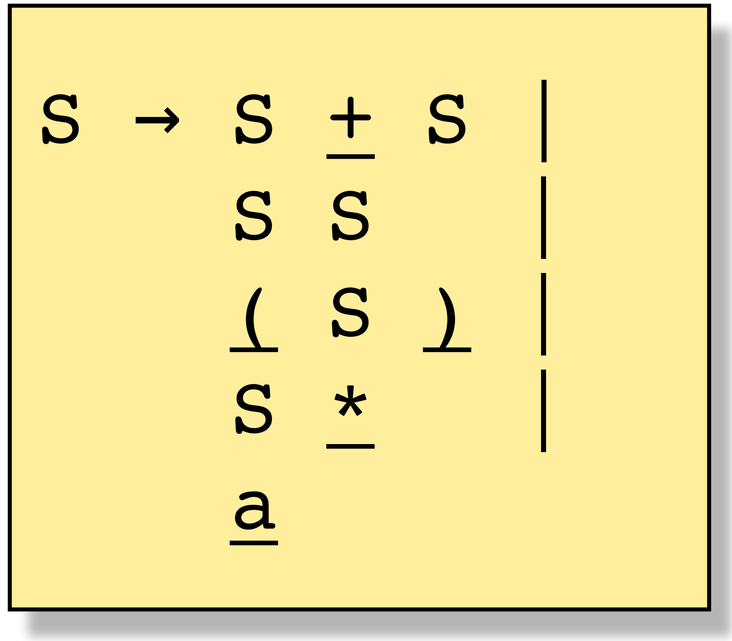
2.



3.



4.

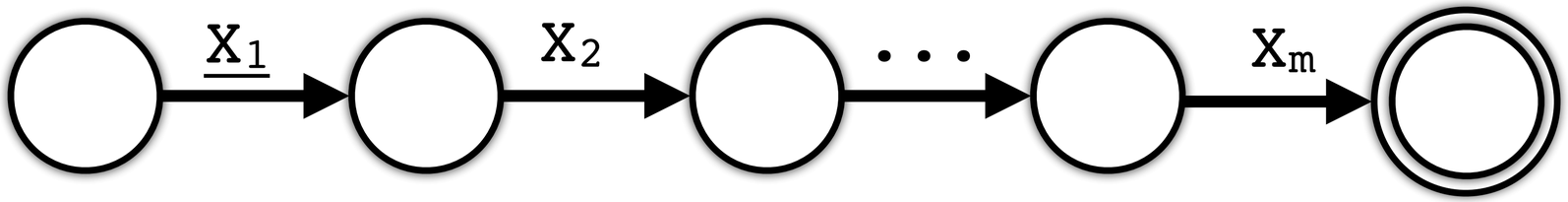


Dragonbook, Exercise 4.4.4

Definitions and Algorithms

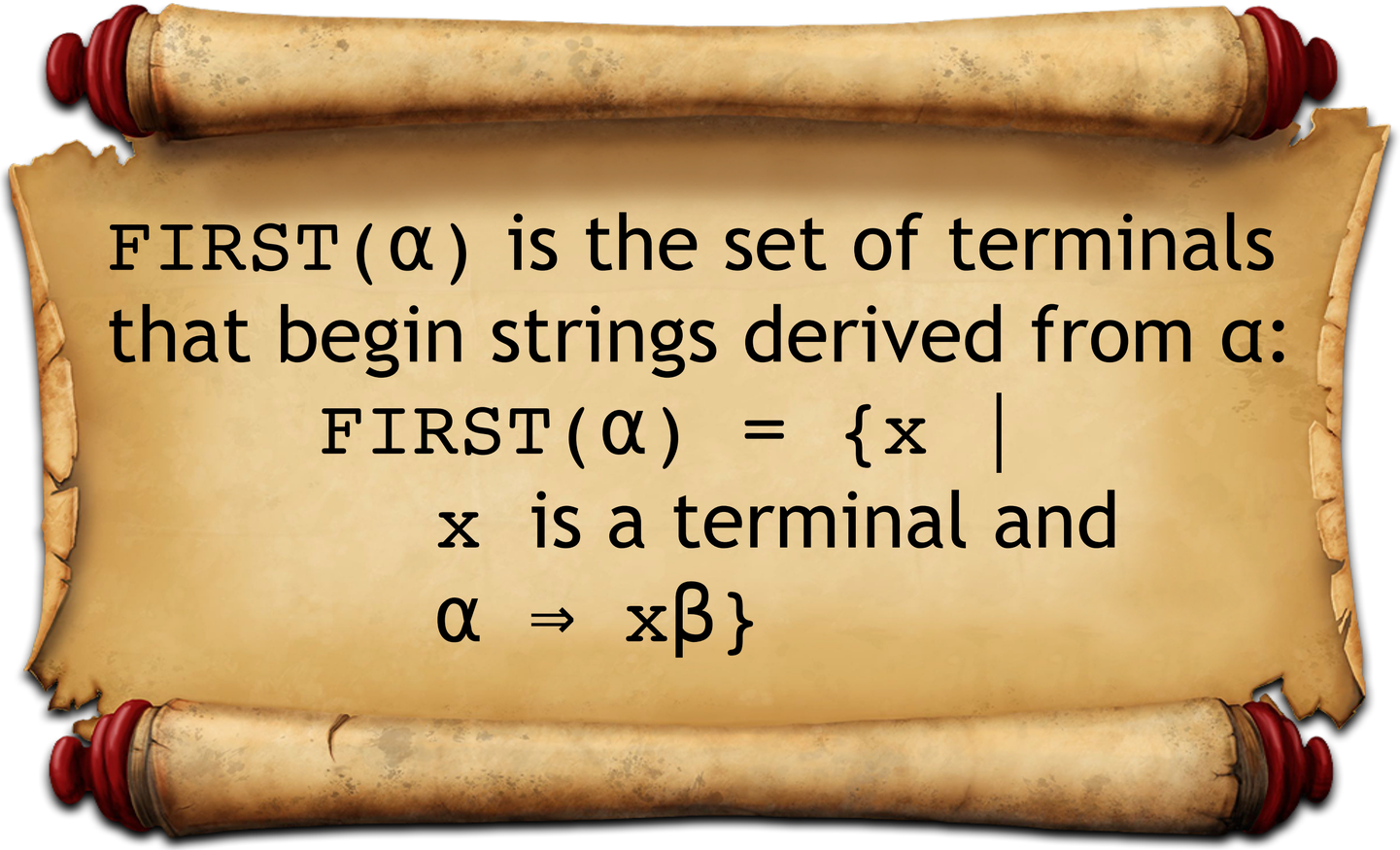
Construct Top-Down Parser

1. **FOR** each non-terminal A **DO**
   create initial ◯ and final ◎ states.
2. **FOR** each production A → $X_1, X_2, \ldots X_m$ **DO**
   create a path from A's initial to final node:



3. Simplify the transition diagrams.
4. **FOR** each transition diagram P **DO**
   Create a procedure P that ‘‘traverses’’
   the diagram guided by the input

FIRST Sets


FIRST(α) is the set of terminals
that begin strings derived from α:
FIRST(α) = {x |
x is a terminal and
α ⇒ xβ}

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**

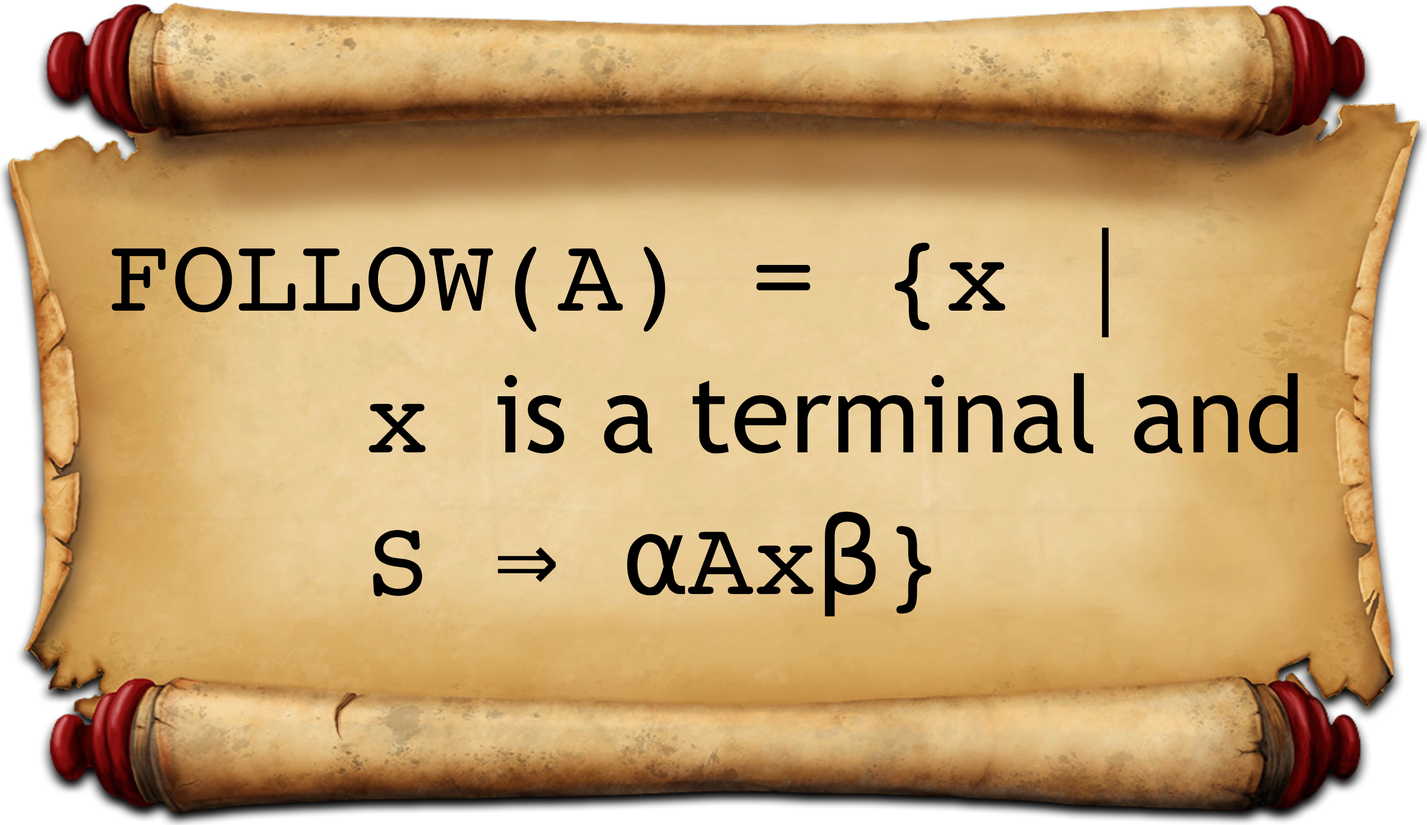
   FIRST(A) ∪= FIRST(Y_1);

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};
```

FOLLOW Sets



- Let $ symbolize end-of-input.
- $ ∈ FOLLOW(A) if A is the rightmost symbol in a sentential form, i.e. S ⇒ αA.

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)

collberg.cs.arizona.edu

ligerlabs.org

Supported by

NSF SATC/TTP-1525820

SATC/EDU-2029632

Christian Collberg

University of Arizona