Programming Language Principles

Programming Language Theory

Supplement

- More explanation for following topics.
  - Language from Grammar
  - Grammar from Language
- Explain Both English and Korean.

Language from Grammar

- How to describe or define language L for given grammar G?
- G = ({S, A}, {a, b}, S, P), and P is as follows.
  - $S \rightarrow aA$, $A \rightarrow bS$, $S \rightarrow \varepsilon$
- What is the language L, given grammar G generates?

Try a few steps

- $S \rightarrow aA$, $A \rightarrow bS$, $S \rightarrow \varepsilon$
- S is the start symbol - G = (V, T, **S**, P).
- Start with S.
- For $S \rightarrow \varepsilon$, we have nothing to proceed.
- So try $S \rightarrow aA$.
  - $S \Rightarrow aA$
  - Then only possible production replacing A is $A \rightarrow bS$.
  - $aA \Rightarrow a(bS)$

Try a few steps

- abS $\Rightarrow$ abaA $\Rightarrow$ ababS $\Rightarrow$ * ababab...aA (1)
    - Or
- ababS $\Rightarrow$ * ababab...abS (2)
- To eliminate variables (i.e., A, S), we need a production which leads to terminal symbols only.
- We only have S $\rightarrow \varepsilon$ (3)
- Hence we cannot finish with (1).
- Apply (3) will simply remove S from (2), hence we can get abab...ab.

Define Language L

- abab...ab is $(ab)^n$, hence $L = \{ (ab)^n : n >= 0\}$
- L = { <write simplified form> : <write conditions>}
- After a few steps, you can find simplified form.
- Writing condition is also important.
- You have to be careful ***not to put too loose or too tight conditions***.
- In the example, we can simply apply $S \rightarrow \varepsilon$, hence the empty string should be included - **n can be zero!**

What if there are more options?

- $G = (\{A, B, S\}, \{a, b\}, S, P)$
    - $S \rightarrow AB$
    - $A \rightarrow aaA|\varepsilon$
    - $B \rightarrow Bb|\varepsilon$
- Start with S, we can derive AB.
- Then what?

Multiple Choices

- Basically, we have to consider all possibilities.
- $S \rightarrow AB$, $A \rightarrow aaA|\varepsilon$, $B \rightarrow Bb|\varepsilon$
- For A, we have two options, for B we also have two options.
- However, we know that terminal symbols will not be replaced.
- Hence we can try production with variables first.
- Let's try to replace A on the left first.

Multiple Choices

- S → AB, A → aaA|ε, B → Bb|ε
- AB ⇒ aaAB
  - Now we have two choices again - replace A or B first?
  - Try both!
  - A: aaaaAB or B: aaABb
- Based on this, we can guess that,
  - every time we replace A, 'aa' is added.
  - every time we replace B, 'b' is added.

Multiple Choices

- $S \rightarrow AB$, $A \rightarrow aaA|\varepsilon$, $B \rightarrow Bb|\varepsilon$
- If we repeat the derivations, we know that **a** is always increased by 2, and **b** is increased by 1.
- So we can guess that strings generated by this grammar will be the form,
  - $a^{2n}b^m$
- Hence $L = \{ a^{2n}b^m : n,m >= 0 \}$

Quick Summary

- To define a language from a grammar,

  1. Find the start symbol.

  2. Try a few steps.

  3. Find the ***general form*** of strings in the language.

  4. Find the ***conditions*** (or constraints) for the general form.

  5. ***Language L = { <general form> : <conditions> }***

Grammar from Language

- Before start, we have to remember two things.
  - ***Productions First, Variables Last.***
    - Once you have productions, you can simply copy variables and terminal symbols used in the productions.
  - There could be ***more than one grammar*** for the same language.

Find Productions

- Always start with the start symbol.
- Normally, we can start with S.
- Productions are repeatedly applied,
  - and ***ended when the right side only has terminal symbols***.
- Let's try to find G for L = { ab^{2n} : n >= 0 }

Find Productions

- L = { ab^{2n} : n >= 0 }
- Based on the general form, we know that ***'a' should be appeared just once***.
- and ***'b' should be always added twice***.

How to add something once?

- We should add 'a' when we replace a variable appeared only once.
  - e.g.) S → aA or S → aB
  - S → aS //**Wrong!**
- Let's try S → aA.
- Then **A** should be replaced with **something containing two 'b's**.
- Also it should be repeatedly add
  - → ***need repeat? use a variable!***

Increasing Number of Terminal Symbols

- $L = \{ ab^{2n} : n >= 0 \}$
- S ➞ aA
- A ➞ ???
- We can try bbA.
  - A ➞ bbA (1)
- So every time we apply (1), 'bb' will be added.
- This is not the end!

Should Stop at Some Point

- $L = \{ ab^{2n} : n >= 0 \}$
- $S \rightarrow aA$, $A \rightarrow bbA$
- Now we can add 'a' once, then 'bb' multiple times.
- Let's try a few steps.
  - $aA \Rightarrow abbA \Rightarrow abbbbA \Rightarrow^* abb...bbA$
- Always **A** at the end.
- So we have to remove A: use ε to completely remove!

Termination

- L = { ab^{2n} : n >= 0 }
- S ➞ aA, A ➞ bbA|ε
- Now we can derive
  - abb...bbA ⇒ abb...bb
- Also, we can directly replace A after the first production applied - means no 'b's.
  - S ⇒ aA ⇒ a
- If L = { ab^{2n} : n >= 1 }, then use 'bb' instead of ε !

Converting to EBNF

- Double check if there exists anything which was correct with BNF, but not valid for EBNF.
- Remember the meaning of new notations.
  - { }: 0 or more ***repetition!***
  - [ ]: optional - 0 or 1
  - ? == [ ]