scanner = ตัด input ที่รับเข้ามาให้เป็นคำๆ (stream of character, tokens) → output คือ stream of tokens

write Regular Expression to cover set of words

grammar
  given sample of sentences in a language, write its grammar
  parsing
    parse tree
    recursive descent
    LL(1)
      compute First and Follow set
      construct parsing table
    given a grammar, compute First, Follow, draw parsing table

paser = ตัวที่เรียก scanner ให้ scanner ส่ง token มาทีละตัวๆ → เช็คว่า input กลายเป็น sequence of token แล้วใช่ไหม + input ถูกต้องตามไวยากรณ์ (grammar)

code generation
  interpreter
  code generator

---

① Scanner

- Regular Expression

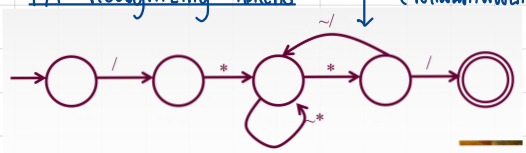
a character or symbol in the alphabet
λ : an empty string → ตัว e หรือ empty เฉยๆ
ϕ : an empty set
if *r* and *s* are regular expressions
- *r* | *s* (หรือ)
- *rs*
- *r** → ไม่มี r หรือมีกี่ตัวก็ได้
- (*r*)

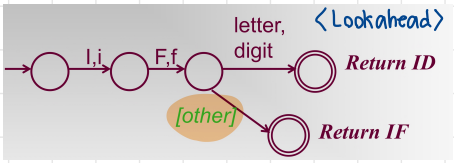
- [a-z]
  - any character in a range from a to z
- .
  - any character
- r^+ → มี r อย่างน้อย 1 ตัว
  - one or more repetition
- *r*?
  - optional subexpression
- ~(*a* | *b* | *c*), [^*abc*]
  - any single character NOT in the set

Examples of Patterns

- (**a** | **A**) = the set {**a**, **A**}
- [**0-9**]$^+$ = (**0** |**1** |...| **9**) (**0** |**1** |...| **9**)* → อย่างน้อย 1 ตัว
- (**0-9**)? = (**0** | **1** |...| **9** | λ) → 0 หรือ 9 หรือไม่เอาก็ได้
- [**A-Za-z**] = (**A** |**B** |...| **Z** |**a** |**b** |...| **z**)
- **A** . = the string with **A** following by any one symbol
- ~[**0-9**] = [^**0123456789**] = any character which is not **0**, **1**, ..., **9**

FA Recognizing Tokens (ใช้แผนภาพอธิบาย)





Production Rules:

S -> aBDh

B -> cC

C -> bC | Є

D -> EF

E -> g | Є

F -> f | Є

FIRST set

FIRST(S) = { a }

FIRST(B) = { c }

FIRST(C) = { b , Є }

FIRST(D) = FIRST(E) U FIRST(F) = { g, f, Є }

FIRST(E) = { g , Є }

FIRST(F) = { f , Є }

FOLLOW Set

FOLLOW(S) = { $ }

FOLLOW(B) = { FIRST(D) – Є } U FIRST(h) = { g , f , h }

FOLLOW(C) = FOLLOW(B) = { g , f , h }

FOLLOW(D) = FIRST(h) = { h }

FOLLOW(E) = { FIRST(F) – Є } U FOLLOW(D) = { f , h }

FOLLOW(F) = FOLLOW(D) = { h }

Production Rules:

S -> ACB|Cbb|Ba

A -> da|BC

B-> g|Є

C-> h| Є

Main First(S)

FIRST set

FIRST(S) = FIRST(A) U FIRST(B) U FIRST(C) = { d, g, h, Є, b, a}

FIRST(A) = { d } U {FIRST(B)-Є} U FIRST(C) = { d, g, h, Є }

FIRST(B) = { g, Є }

FIRST(C) = { h, Є }

FOLLOW Set

FOLLOW(S) = { $ }

FOLLOW(A) = { h, g, $ }

FOLLOW(B) = { a, $, h, g }

FOLLOW(C) = { b, g, $, h }

Production Rules:

S -> aBDh

B -> cC

C -> bC | ϵ

D -> EF

E -> g | ϵ

F -> f | ϵ

FIRST set

S → aBDh

FIRST(S) = { a }

FIRST(B) = { c }

FIRST(C) = { b , ϵ }

FIRST(D) = FIRST(E) U FIRST(F) = { g, f, ϵ }

*** เพราะเป็น ϵ

FIRST(E) = { g , ϵ }

FIRST(F) = { f , ϵ }

FOLLOW Set

** เพราะ ถัดจากเป็น ϵ ได้

FOLLOW(S) = { $ }

FOLLOW(B) = { FIRST(D) - ϵ } U FIRST(h) = { g , f , h }

FOLLOW(C) = FOLLOW(B) = { g , f , h }

FOLLOW(D) = FIRST(h) = { h }

FOLLOW(E) = { FIRST(F) - ϵ } U FOLLOW(D) = { f , h }

FOLLOW(F) = FOLLOW(D) = { h }

Computing Follow

1. If A is start symbol, put $ in FOLLOW(A)
2. Productions of the form B → α A β,
   FOLLOW(A)= FIRST(β) ϵ ไม่เอา λ นะ!
3. Productions of the form B → α A or
   B → α A β where β ⇒* ε
   Add FOLLOW(A)= FOLLOW(B)

S → ACB

∴ Follow(A) = First(C) U First(B) U Follow(S)

เพราะ C เป็น ϵ ได้

เพราะ B,C เป็น ϵ ได้

S → ACB

∴ Follow(C) = First(B) U Follow(S)

S → ACB

∴ Follow(B) = Follow(S)

A → BC

∴ Follow(C) = Follow(A)

A → BC

∴ Follow(B) = First(c) U Follow(A)

***

Production Rules:

S -> ACB|Cbb|Ba

A -> da|BC

B-> g|ϵ

C-> h| ϵ

FIRST set

FIRST(S) = FIRST(A) U FIRST(B) U FIRST(C) = { d, g, h, ϵ, b, a}

FIRST(A) = { d } U {FIRST(B)-ϵ} U FIRST(C) = { d, g, h, ϵ }

FIRST(B) = { g, ϵ }

FIRST(C) = { h, ϵ }

ต้องมี! เพราะ C,B เป็น λ ได้

FOLLOW Set

FOLLOW(S) = { $ }

FOLLOW(A) = { h, g, $ }

FOLLOW(B) = { a, $, h, g }

FOLLOW(C) = { b, g, $, h }

Production Rules:

S -> ACB|Cbb|Ba

A -> da|BC

B-> g|ϵ

C-> h| ϵ

EXAMPLE

Consider the expression grammar (4.11), repeated below:

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \varepsilon$
$F \rightarrow (\ E\ ) \mid \mathbf{id}$

Then:

FIRST(E) = FIRST(T) = FIRST(F) = {(, **id**}
FIRST(E') = {+, ε}
FIRST(T') = {*, ε}

FOLLOW(E) = FOLLOW(E') = {) , $}
FOLLOW(T) = FOLLOW(T') = {+,), $}
FOLLOW(F) = {+, *,), $}

Examples of Quiz in compiler (old questions from yesteryear) (only two examples)

1) This is an input source for our scanner

```
fac(n)
  if( n == 0 ) return 1
  else return n * fac(n-1)
```

the kind of "words" in this program are:

reserve word : if else return
symbol: ( ) == * -
constant: 0 1
identifier: fac n

write regular expression to capture all "words" in this program
(there may be sereval REs)

2) Given this grammar, compute First, Follow set and draw Parsing Table

```
dcl = ID dcl2
dcl2 =  ( formal ) STMT |  [ NUMBER ]
formal = ID formals | nil
formals = , formal | nil
```

EXAMPLE

Consider the expression grammar (4.11), repeated below:

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \varepsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \varepsilon$
$F \rightarrow (\ E\ ) \mid \textbf{id}$

Computing Follow

1. If A is start symbol, put $ in FOLLOW(A)
2. Productions of the form $B \rightarrow \alpha A \beta$, FOLLOW(A)= FIRST(β) ยกเว้น λ นะ!
3. Productions of the form $B \rightarrow \alpha A$ or $B \rightarrow \alpha A \beta$ where $\beta \Rightarrow^* \varepsilon$, Add FOLLOW(A)= FOLLOW(B)

Then:

FIRST(E) = FIRST(T) = FIRST(F) = {(, **id**}
FIRST(E') = {+, ε}
FIRST(T') = {*, ε}

FOLLOW(E) = FOLLOW(E') = {) , $}
FOLLOW(T) = FOLLOW(T') = {+,), $}
FOLLOW(F) = {+, *,), $}

Examples of Quiz in compiler (old questions from yesteryear) (only two examples)

1) This is an input source for our scanner

```
fac(n)
  if( n == 0 ) return 1
  else return n * fac(n-1)
```

the kind of "words" in this program are:

reserve word : if else return
symbol: ( ) == * -
constant: 0 1
identifier: fac n

write regular expression to capture all "words" in this program
(there may be sereval REs)

MDU
reserve word = (if | else | return)
symbol = ()| == | * | -
contants = 0|1
identifier = (fac |n)

```
reserve_word = (if | else | return)
symbols = ([(]|[)]|[==]|[*]|[-])
contants = [0-9]+
identifier = [A-Za-z][A-Za-z0-9]*
```

ยากเพื่อนในชั้น

2) Given this grammar, compute First, Follow set and draw Parsing Table

```
dcl = ID dcl2
dcl2 = ( formal ) STMT | [ NUMBER ]
formal = ID formals | nil
formals = , formal | nil
```

เพราะ NUMBER ไม่ใช่ terminals

Computing Follow

1. If A is start symbol, put $ in FOLLOW(A)
2. Productions of the form $B \rightarrow \alpha A \beta$, FOLLOW(A)= FIRST(β) ยกเว้น λ นะ!
3. Productions of the form $B \rightarrow \alpha A$ or $B \rightarrow \alpha A \beta$ where $\beta \Rightarrow^* \varepsilon$, Add FOLLOW(A)= FOLLOW(B)

First(dcl) = {ID}
First(dcl2) = {(,[}
First(formal) = {ID, nil}
First(formals) = {',',nil}

Follow(dcl) = {$} เพราะเป็นคีย์แรก
Follow(dcl2) = Follow(dcl) = {$}
Follow(formal) = {)} ∪ Follow(formals) = {)}
Follow(formals) = Follow(formal) = {)}

	ID	(	)	[	]	NUMBER	,	STMT
dcl	①							
dcl2		②		③				
formal	④		formal = nil ⑤					
formals			formals = nil ⑦				⑥	

*มี nil ดู follow set

ยากเพื่อนในชั้นเรียนอีก

```
                                first set       follow set
dcl -> ID dcl2                  : {ID}          : {$}
dcl2 -> (formal) STMT | [ NUMBER ]  : {(,[}         : {$}
formal -> ID formals | nil      : {ID,nil}      : {)}
formals -> , formal | nil       : {,nil}        : {)}

    parsing table
                ID                  (                   )                   [
,
dcl     | dcl -> ID dcl2     |                   |                   |
|
dcl2    |                   | dcl2 -> (formal) STMT |               | dcl2 -> (formal) STMT |
|
formal  | formal -> ID formals |                 | formal -> nil     |                   |
|
formals |                   |                   | formals -> nil    |                   |
formals -> , formal |
```

Quiz compiler (section 1) 21 March 2022, time 1:30 hour
Only one question, I think it will probably takes one hour.

Parser (LL1)

Here is the grammar for a subset of BASIC language. Computer First and Follow set and construct a parsing table. (You can submit photo of your parsing table). Example of a parsing table is in slide page 25 of parsing.
You can also find it here

https://www.cp.eng.chula.ac.th/~prabhas//teaching/prolang/2022/write-parser-from-parsing-table.htm

```
pgm := line | EOF
line := LNUM stmt line | empty
stmt := asgmt | goto
asgmt := ID = exp
exp := ID exps
exps := + exp | empty
goto := GOTO LNUM
```

Example of the program is:

```
10 A = B
30 B = A + A + A
40 GOTO 10
```

Note: the only variable name is "A". EOF is end of file (or end of input).
You can read more about BASIC here. (It was a project of this class many years ago).

https://www.cp.eng.chula.ac.th/~prabhas//teaching/prolang/2018/retro-basic.htm

Computing Follow

1. If A is start symbol, put $ in FOLLOW(A)
2. Productions of the form B → α A β,
   FOLLOW(A)= FIRST(β) (ยกเว้น λ นะ!)
3. Productions of the form B → α A or
   B → α A β where β ⇒* ε (หรือไม่ก็ได้)
   Add FOLLOW(B)= FOLLOW(A)

① pgm → line
② pgm → EOF
③ line → LNUM stmt line
④ line → empty
⑤ stmt → asgmt
⑥ stmt → goto
⑦ asgmt → ID = exp
⑧ exp → ID exps
⑨ exps → + exp
⑩ exps → empty
⑪ goto → GOTO LNUM

asgmt → ID = exp tmp
tmp → empty
จริงๆก็ยังสลัดได้

First(pgm) = {LNUM, empty, EOF}
First(line) = {LNUM, empty}
First(stmt) = {ID, GOTO}
First(asgmt) = {ID}
First(exp) = {ID}
First(exps) = {+, empty}
First(goto) = {GOTO}

Follow(pgm) = {EOF} หรือ $
Follow(line) = {EOF}
Follow(stmt) = First(line) ∪ Follow(line) = {LNUM, EOF} (ไม่เอาสตริงว่าง (empty))
Follow(asgmt) = Follow(stmt) = {LNUM, EOF}
*** Follow(exp) = Follow(exps) = {LNUM, EOF} = Follow(asgmt)
Follow(exps) = Follow(exp) = {LNUM, EOF}
Follow(goto) = Follow(stmt) = {LNUM, EOF}

*ไม่มี ε กับ EOF ใช่

Parsing Table

	LNUM	ID	=	+	GOTO	EOF
pgm	①					②
line	③					④
stmt		⑤			⑥	
asgmt		⑦				
exp		⑧				
exps	⑩			⑨		⑩
goto					⑪	

*** อย่าลืม!

พิจารณา first set ของ row (ที่มี λ ก็พิจารณา follow)

① pgm → line
② pgm → EOF
③ line → LNUM stmt line
④ line → empty
⑤ stmt → asgmt
⑥ stmt → goto
⑦ asgmt → ID = exp
⑧ exp → ID exps
⑨ exps → + exp
⑩ exps → empty
⑪ goto → GOTO LNUM

① pgm → line (First ของ line คือ ①)
② pgm → EOF
③ line → LNUM stmt line
④ line → empty
⑤ stmt → asgmt (First ของ asgmt)
⑥ stmt → goto
⑦ asgmt → ID = exp
⑧ exp → ID exps
⑨ exps → + exp
⑩ exps → empty (ที่เป็น empty จะพิจารณา Follow(exps))
⑪ goto → GOTO LNUM

```
pgm := line pgm | EOF
line := lnum stmt line | empty
stmt := asgmt | goto
asgmt := id = exp
exp := id exps
exps := + exp | empty
goto := GOTO lnum
id := A | B
```

	First set	Follow set
pgm	{lnum, empty, EOF}	{EOF}
line	{lnum, empty}	First(pgm) = {EOF, lnum}
stmt	{A, B, GOTO}	First(line) ∪ Follow(line) = {lnum, EOF}
asgmt	{A, B}	Follow(stmt) = {lnum, EOF}
exp	{A, B}	Follow(asgmt) ∪ Follow(exps) = {lnum, EOF}
exps	{+, empty}	Follow(exp) = {lnum, EOF}
goto	{GOTO}	Follow(stmt) = {lnum, EOF}
id	{A, B}	{=} ∪ First(exps) ∪ Follow(exp) = {=, +, lnum, EOF}

```
pgm := line pgm
pgm := EOF
line := lnum stmt line
line := empty
stmt := asgmt
stmt := goto
asgmt := id = exp
exp := id exps
exps := + exp
exps := empty
goto := GOTO lnum
id := A
id := B
```

```
pgm := line pgm | EOF
line := lnum stmt line | empty
stmt := asgmt | goto
asgmt := id = exp
exp := id exps
exps := + exp | empty
goto := GOTO lnum
id := A | B
```

return 0 - error, 1 - OK

pgm()

Grammar of ASM

```
asm = op oprnd asm | EOF
op = CLRA | MOV | ADD | MOVA
oprnd = reg , num | reg | empty
reg = R[0..31]
```

Parser

return 0 - error, 1 - OK

```
reg()
  match('R')
  ret num()

op()
  switch tokentype()
    mov: match('MOV')
    ...
    default: ret 0

oprnd()
  if reg() != 0
     if token == ','     // lookahead
        match(',')
        ret num()
     else
        ret 1
   ret 1

asm()
   if token == EOF  ret 1
   if op() != 0
      oprnd()
      asm()
   ret 0
```

Quiz compiler (section 1) 21 March 2022, time 1:30 hour
Only one question, I think it will probably takes one hour.

Parser (LL1)

Here is the grammar for a subset of BASIC language. Computer First and Follow set and construct a parsing table. (You can submit photo of your parsing table). Example of a parsing table is in slide page 25 of parsing. You can also find it here

https://www.cp.eng.chula.ac.th/~prabhas//teaching/prolang/2022/write-parser-from-parsing-table.htm

```
pgm := line | EOF
line := LNUM stmt line | empty
stmt := asgmt | goto
asgmt := ID = exp
exp := ID exps
exps := + exp | empty
goto := GOTO LNUM
```

Example of the program is:

10 A = B
30 B = A + A + A
40 GOTO 10

Note: the only variable name is "A". EOF is end of file (or end of input).
You can read more about BASIC here. (It was a project of this class many years ago).

https://www.cp.eng.chula.ac.th/~prabhas//teaching/prolang/2018/retro-basic.htm


```
pgm := line pgm | EOF
line := lnum stmt line | empty
stmt := asgmt | goto
asgmt := id = exp
exp := id exps
exps := + exp | empty
goto := GOTO lnum
id := A | B
```