LL(1)

	(	)	{	}	eof
S	(S)	ε	{S}	ε	ε

input: ({ }) eof

2. Left-factor 去X

1\. Remove Left recursion.

$A \to \underbrace{\alpha_1 | \alpha_2 | \ldots | \alpha_n}_{①} | \underbrace{A\beta_1 | A\beta_2 | \ldots | A\beta_m}_{②}$

$\Rightarrow A \to \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_n A'$

$A' \to \beta_1 A' | \beta_2 A' | \ldots \beta_m A' | \varepsilon$

① 左递归部分全部去掉

② 没递归的部分 ~~加上~~ ~~变成为~~ ~~递归~~ 加上新的NonTerminal

③ 右递归（新Non-Terminal）.

{ S } (S S) S))) eof eof eof eof

① ~~[illegible]~~ START symbol S

② S (→ (S), ~~[illegible]~~ Left→Right ⇒ Up→Down.

push from right to left ☆.

③ check the first symbol < terminal, match current token, call next! ☆
NonTerminal, ~~scan next token~~, read current token! ☆

FIRST / FOLLOW. FIRST 有两种，左边先定义，再定义右边！两种规则不一样！

Eg.

S → BcDB

B → ab | cS

D → d | ε

FIRST(S) = ~~FIRST(BD) = FIRST a~~

= FIRST(B) + FIRST(D) = {a, c} + {d}

= {a, c, d}

FIRST(cS) = {c}

FIRST(a,b) = {a}

FIRST(DB) = FIRST(d) + FIRST(B) = {d, a, c}

FOLLOW(S) = {EOF, c}

(B) = {c, EOF}.

(D) = {a, c}

FOLLOW only defined for single nonterminals! RHS, no ε! has EOF!

C1. if A start, add eof, C2 Add FIRST(β) − {ε} C3 Add Follow(X) if ε in β

$\boxed{X \to \alpha A \beta}$

~~X → α~~

~~S → Bc~~

Table: each production X → α

for each terminal t in FIRST(α)

put α in Table [X][t]

if ε is in FIRST(α)

for each terminal t in FOLLOW(X)

put α in table [X][t]

Eg.

X → α

S → BC

FIRST(Bc) = {a, c}

t = a

Put Bc in [S][a]

t = c

Bc in [S][c]

D → ε.

⋮

FOLLOW(D)

Table collision ≠ LL(1). ⇒ LR Parsers! < Left to Right scan of input
~~Rever~~ Reverse rightmost derivation!

Parse Table.

switch action[t, a] {

case shift s:

push(s)

a = scan(); //next token.

case reduce A → α

pop length(α) times

t' = new stack of symbol!

push(goto[t', A]); //A be pushed!

⇒ Closure(I) X → α.Bβ 如果B还有 ~~[illegible]~~ ~~[illegible]~~ 规则，就加入进去！

if B → γ, ~~add~~

add B → .γ

GoTo(I, X)

= Closure({A → αX.β | A → α.Xβ is in I}) (post, pre)

~~[illegible]~~ move dot!

LL(1)

	(	)	{	}	eof
S	(S)	ε	{S}	ε	ε

input: ({ }) eof

S eof

(S) eof

S) eof

{ S }) eof

① ~~...~~ START symbol S   push from right to left ☆.

② S (→ (S), ~~...~~ Left → Right ⇒ Up → Down.

③ check the first symbol < terminal, match current token, call next! ☆
NonTerminal, ~~scan next token~~, read current token! ☆

2. Left-factor 去X

1. Remove Left recursion.

$A \to \underbrace{\alpha_1 | \alpha_2 | \ldots | \alpha_n}_{①} | \underbrace{A\beta_1 | A\beta_2 | \ldots | A\beta_m}_{②}$

$\Rightarrow A \to \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_n A'$

$A' \to \beta_1 A' | \beta_2 A' | \ldots \beta_m A' | \varepsilon$

① 左递归部分全部去掉

② 没递归的部分 ~~...~~ 加上新的 NonTerimal

③ 右递归 (新 Non-Terimal).

FIRST/FOLLOW.   FIRST 有两种，左边先定义，再定义右边！两种规则不一样！

Eg.

$S \to BcDB$

$B \to ab | cS$

$D \to d | \varepsilon$

FIRST(S) = ~~FIRST(BD) = FIRST a~~

= FIRST(B) + FIRST(D) = {a, c} + {d}

= {a, c, d}

FIRST(cS) = {c}

FIRST(a, b) = {a}

FIRST(DB) = FIRST(d) + FIRST(B) = {d, a, c}

FOLLOW(S) = {EOF, c}

(B) = {c, EOF}.

(D) = {a, c}

FOLLOW only defined for single non-terminals! RHS, no ε! has EOF!

C1. if A start, add eof, C2 Add FIRST(β) − {ε} C3 Add Follow(X) if ε in β

$\boxed{X \to \alpha A \beta}$ ~~X → α~~ ~~S → Bc~~

Table: each production $X \to \alpha$
  for each terminal t in FIRST(α)
    put α in Table[X][t]
  if ε is in FIRST(α)
    for each terminal t in FOLLOW(X)
      put α in table[X][t]

Eg.
$X \to \alpha$
$S \to BC$
FIRST(BC) = {a, c}
t = a
  Put BC in [S][a]
t = c
  BC in [S][c]

D → ε.
⋮
FOLLOW(D)

Table collision ≠ LL(1). ⇒ LR Parsers! < Left to Right scan of input
~~Rever~~ Reverse rightmost derivation!

Parse Table.

```
switch action[t, a] {
  case shift s:
    push(s)
    a = scan(); //next token.
  case reduce A->α
    pop length(α) times
    t' = new stack of symbol!
    push(goto[t', A]); //A be pushed!
```

⇒ Closure(I) $X \to \alpha . B\beta$ 如果B还有~~...~~规则，就加进去！

if $B \to \gamma$, ~~add~~

add $B \to .\gamma$

GoTo(I, X) = Closure({$A \to \alpha X . \beta$ | $A \to \alpha . X\beta$ is in I})   post pre

move dot!

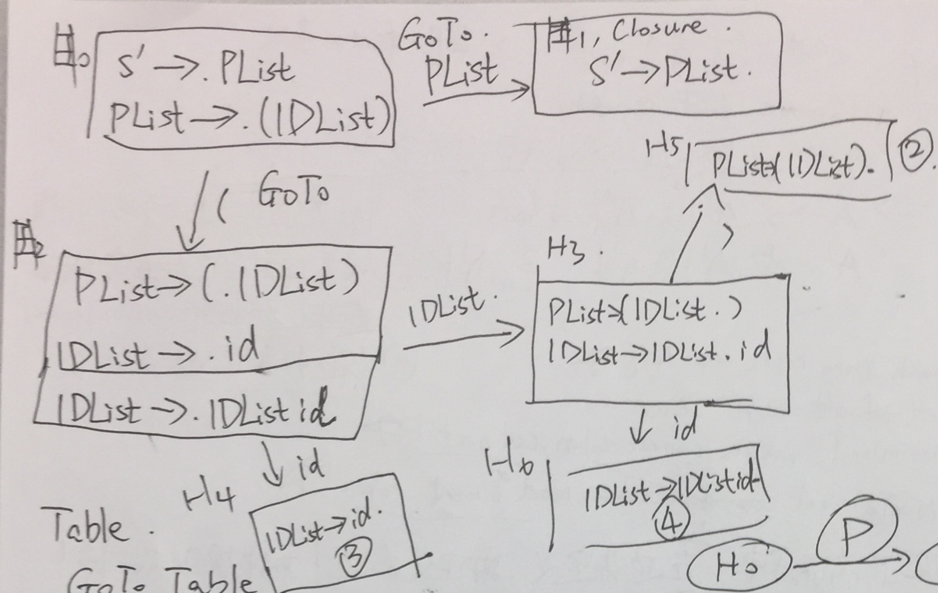


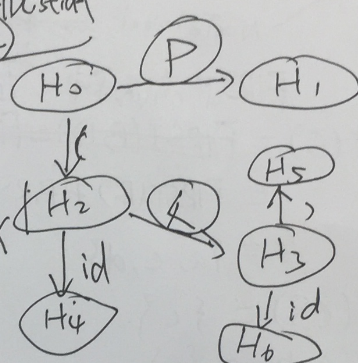
Table.

GoTo Table

for every nonterminal X

if there is an $[H_j, H_k]$ edge on X

set $GoTo[H_j, X] = H_k$.



Non-Terimal

	P	L
H_0	H_1	
H_1		
H_2		H_3.
H_3		
H_4		
H_5		
H_6..		

Action Table: {Shift/Reduce/Accept}.

Shift. $A \to \alpha \cdot t \beta$ (H_0, H_3, H_2)

t is terminal

$Action[H_j, t] = shift\ H_k$.

Reduce. $A \to \alpha \cdot$ (H_4, H_5, H_6)

A is not S' (start)

for each t is FOLLOW(A)

$Action[H_j, t] \Rightarrow reduce$ 规则. $A \to \alpha$.

FOLLOW (L) = {), id }

FOLLOW (P) = { eof }

Accept : $S' \to S \cdot$

$[H_j, eof]$ = accept!

	(	)	id	eof
H_0	SH_2			
H_1				✓.
H_2			SH4	
H_3		SH_5	SH_6	
H_4		R③	R③	
H_5				R②
H_6		R④	R④	