2 parcial

Sunday, 24 November 2019 7:01 PM

Teoría

- Existen 3 tipos de análisis sintácticos

① Universal → Cualquier tipo de gramática
  → Es lento

② Descendente → Va de raíz a hojas
  → Derivación por la izquierda
  → Tipos → Recursivo
    → LL(k)

③ Ascendente → SLR
  → LR
  → LALR (Bison)

- Recuperación errores

a) Modo pánico b) Gramática con producción de errores c) Correcciones
  → Por frase
  → Por forma global

Examen Tipo B

① Sea la gramática

a) $S \rightarrow SS \mid id = E$

b) $E \rightarrow E^* \mid E\ 'l'\ E \mid (E) \mid EE \mid a \mid b$

② Eliminar Ambigüedad (Jerarquía)

1 Disyunción $E \rightarrow E\ 'l'\ C \mid C$
Concatenación $C \rightarrow CK \mid K$
* $K \rightarrow K^* \mid P$

Concatenación

*

( ) ab

K → K* | P

P → (E) | a | b

c) Derivación por izquierda x = a b | (a)*

E → E T //Sólo recorremos

// Gramática Nueva

S → ss | id = E ;

E → E '|' C | C

C → C K | K

K → K* | P

P → (E) | a | b

S => x = E ; => x = E | C ;

=> x = E | C ; => x = C | C ; =>

=> x = C K | C ; => x = C K | C ;

=> x = K K | C ; => x = P K | C ;

=> x = a K | C ; => x => a P | C ;

=> x = a b | C ; => x = a b | K* ;

=> x = a b | (E)* ; => x = a b | (C)* ;

=> x = a b | (K)* ; => x = a b | (P)* ;

=> x = a b | (a)* //

∴ Si acepta por la izquierda

// Derivación derecha x = a b | (a)*

x = E | K

S => x = E ; => x = E | C ; => x = E | K* ;

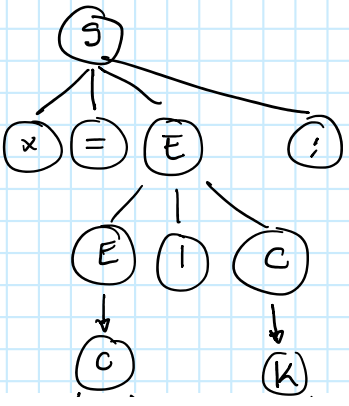
=> x = E | P* ; => x = E | (E)* ; => x = E | (C)* ;

=> x = E | (K)* ; => x = E | (a)* ; => x = C | (a)* ;

=> x = C K | (a)* ; => x = C P | (a)* => x = C b | (a)*

=> x = K b | (a)* ; => x = P b | (a)* => x = a b | (a)*

d) Árbol de derivación x = a b | (a)*



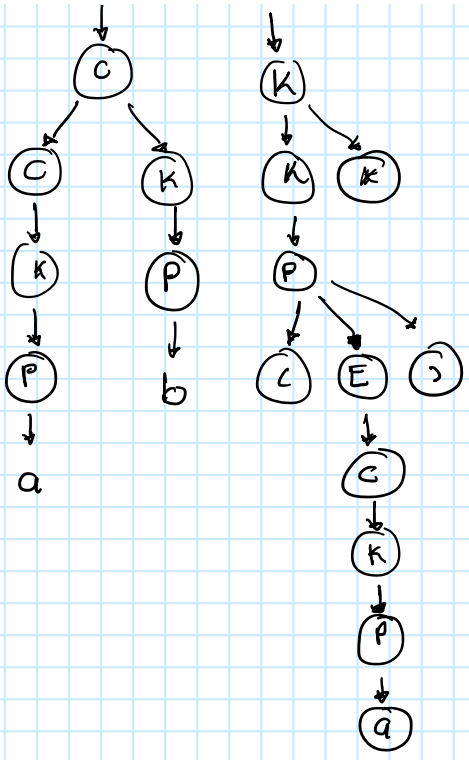
S → ss | id = E ;

E → E '|' C | C

C → C K | K

K → K* | P

P → (E) | a | b



5 EBNF $A \to A\alpha \mid \beta$ $A \to \beta\{\alpha\}$

$S \to \underbrace{SS}_{A\alpha} \mid \underbrace{id = E;}_{\beta}$ $S \to id = E\{S\}$

$E \to \underbrace{E\,'+'\,C}_{A\ \alpha} \mid \underbrace{C}_{\beta}$ $E \to C\{+C\}$

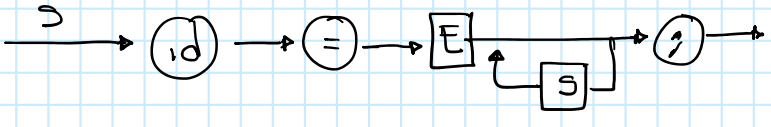
$C \to \underbrace{CK}_{A\ \alpha} \mid \underbrace{K}_{\beta}$ $C \to K\{K\}$

$K \to \underbrace{K*}_{A\alpha} \mid \underbrace{P}_{\beta}$ $K \to P\{*\}$

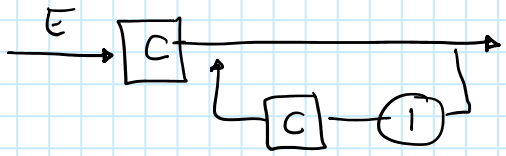
$P \to \underbrace{(E)}_{\beta} \mid \underbrace{a}_{\beta} \mid \underbrace{b}_{\beta}$ $P \to (E) \mid a \mid b$

6 Diagramas C i

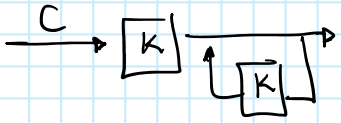
$S \to id = E; \{S\}$ |



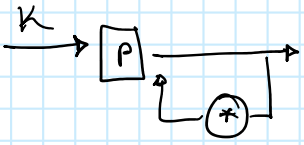
$E \rightarrow C \{ | C \}$



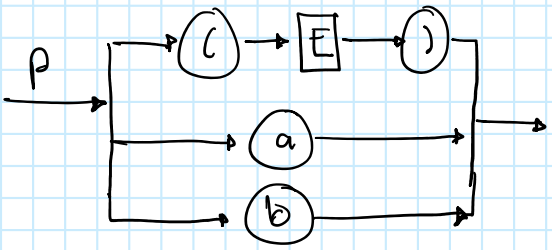
$C \rightarrow K \{ K \}$



$K \rightarrow P \{ * \}$



$P \rightarrow (E) \mid a \mid b$



3 Parcial

Sunday, 24 November 2019 8:25 PM

$E \to E \text{ or } T \mid T$
$T \to T \text{ and } F \mid F$
$F \to \text{not } F \mid id$

a) Obten gramática LL(1)

$A \to A\alpha \mid \beta$

$A \to \beta A'$
$A' \to \alpha A' \mid \varepsilon$

} Recursividad Izquierda

$E \to \underbrace{E \text{ or } T}_{\alpha} \mid \underbrace{T}_{\beta}$ | $E \to T\,E'$, $E' \to \text{or } T\,E' \mid \varepsilon$

$T \to \underbrace{T \text{ and } F}_{\alpha} \mid \underbrace{F}_{\beta}$ | $T \to F\,T'$, $T' \to \text{and } F \mid \varepsilon$

$F \to \text{not } F \mid id$ | $F \to \text{not } F \mid id$

↑ No recursividad izquierda

b)

	Anulable	First: or	and	not	id	ε	Follow: or	and	not	id	$
E				✓	✓						✓
E'	✓	✓				✓					✓
T				✓	✓		✓				✓
T'	✓		✓			✓	✓				✓
F				✓	✓		✓	✓			✓

$E \to TE'$
$E' \to \text{or } TE' \mid \varepsilon$
$T \to FT'$

(1) Si A es símbolo inicial

$Follow(A) = Follow(A) \cup \{\$\}$

E → or T E' | ε
T → F T'
T' → and F T' | ε
F → not F | id

Follow(A) = Follow(A) ∪ {$}

② Si B → α A β

Follow(A) = Follow(A) ∪ First(β)

③ Si B → α A | B → α A β
β ≟ Anulable

Follow(A) = Follow(A) ∪ Follow(B)

Follow(E) = { / ∪ $ }

Follow(E') = { //No hay nada ∴ Follow(E) = { $ } }

Follow(T) = { E ∴ First(E') = { or, ε } Anulable

Anulable; E → T E'

∴ Follow(E) = { $ }

E' ∴ First(E') = { or, ε } Anulable

Anulable; E' → or T

∴ Follow(E') = { $ }

Follow(T') = { T // No hay nada

∴ Encabezado Follow(T) = { or, $ }

T' // No hay nada

∴ Encabezado Follow(T') = { or, $ }

Follow(F) = { T First(T') = { and, ε }

Anulable T → F

∴ Follow(T) = { or $ }

∴ Follow(T) = { or, $ }

F | // No hay nada
∴ Encabezado Follow(F) = { or, and, $ }

c) Construir la tabla

	or	and	not	id	$
E			E → TE'	E → TE'	
E'	E' → or TE'				E → ε
T			T → FT'	F → FT'	
T'	T' → ε	T' → and FT'			T' → ε
F			F → not F	F → id	

E → TE' | First(TE') = { not, id }

E' → or TE' | First(or TE') = { or }

E' → ε | First(ε) = { ε }
∴ Follow(E') = { $ }

T → FT' | First(FT') = { not, id }

T' → and FT' | First(and FT') = { and }

T' → ε | First(ε) = { ε }
∴ Follow(T') = { or, $ }

F → not F | First(not F) = { not }

$F \to not\ F$ | $First(not\ F) = \{not\}$

$F \to id$ | $\{id\}$

d) Análisis de "id or id and not id"

Pila	Entrada	Acción
$E\$$	id or id and not id $	$E \to TE'$
$TE'\$$	id or id and not id $	$T \to FT'$
$FT'E'\$$	...	$F \to id$
$idT'E'\$$	...	avanzar
$T'E'\$$	or id and not id $	$T' \to \varepsilon$
$E'\$$	...	$E' \to or\ TE'$
$or\ TE'\$$	...	avanzar
$TE'\$$	id and not id $	$T \to FT'$
$FT'E'\$$	...	$F \to id$
$idT'E'\$$	...	avanzar
$T'E'\$$	and not id $	$T' \to and\ FT'$
$and\ FT'E'\$$	...	avanzar
$FT'E'\$$	not id $	$F \to not\ F$
$not\ FT'E'\$$	not id $	avanzar
$FT'E'\$$	id $	$F \to id$
$idT'E'\$$	... $	avanzar
$T'E'\$$	$	$T' \to \varepsilon$
$E'\$$	$	$T \to \varepsilon$
$\$$	$	Acepta //

---

c) Gramática

$E \to \overset{\alpha}{T}\ \overbrace{or\ E}^{\beta} \mid \overset{\alpha}{T}$

$T \to F\ and\ T \mid F$

$A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \alpha A' \mid \varepsilon$

} Recursividad Izquierda

$T \rightarrow F \text{ and } T \mid F$

$F \rightarrow \text{not } F \mid id$

$A' \rightarrow \alpha A' \mid \varepsilon$ } Izquierda

// Factorizar

$E \rightarrow TE'$

$E' \rightarrow \text{or } E \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow \text{on } T \mid \varepsilon$

$F \rightarrow \text{not } F \mid id$

$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$

$A \rightarrow \alpha A' \mid \gamma_1 \mid \gamma_2 \mid \ldots \mid \beta_n$

$A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n \mid \varepsilon$

} Factorizar

b) Parser Recursivo

```
void E(){  // E → TE'
    T();
    Ecom();
}

void Ecom(){ // E → or E/ε
    if(token == or){
        eattoken(or);
        E();
    }
}
```

```
void F(){
  switch(token){
    case 'not';
        eattoken(not);
        F();
    case 'id';
        eattoken(id);
    default;
        error();
  }
```

4 parcial

Sunday, 24 November 2019 10:57 PM

// Gramática

$S \to S ; id = E \mid id = E$

$E \to E + id \mid id$

$S' \to S$

$S \to S ; id = E \mid id = E$

$E \to E + id \mid id$

Automata

cerradura($S' \to \cdot S, \$$)

$= \{ S' \to \cdot S, \$ \quad First(\$) = \{\$\}$

$S \to \cdot S ; id = E, \$/; \quad First(; id = E) = \{;\}$ // Agrego lo directo mismo NT

$S \to \cdot id = E, \$/; \ \} = I_0$

$goto(I_0, S) = \{ S' \to S \cdot, \$$ // Acepta → I_1 con $

$S \to S \cdot ; id = E, \$/; \ \} = I_1$

$goto(I_0, id) = \{ id \cdot = E, \$/; \} = I_2$

$goto(I_1, ;) = \{ S \to S ; \cdot id = E, \$/; \} = I_3$

$goto(I_2, =) = \{ id = \cdot E, \$/; \quad First(\$/;) = \{\$, ;\}$

$E \to \cdot E + id, \$/; / + \quad First(+ id) = \{+\}$

$E \to \cdot id, \$/; / + \ \} = I_4$

$goto(I_3, id) = \{ S \to S ; id \cdot = E, \$/; \} = I_5$

$goto(I_4, E) = \{ S \to id = E \cdot, \$/; \}$

$E \to E \cdot + id, \$/; / + \ \} = I_6$

$goto(I_4, id) = \{ E \to id \cdot, \$/; / + \} = I_7$

$goto(I_5, =) = \{ S \to S ; id = \cdot E, \$/; \quad First(\$/;) = \{\$/;\}$

$E \to \cdot E + id, \$/; / + \quad First(+ id) = \{+\}$

$E \to \cdot id, \$/; / + \ \} = I_8$

$goto(I_6, +) = \{ E \to E + \cdot id, \$/; / + \} = I_9$

goto(I_6, +) = { E → E + · id, $/;/+ } = I_9

goto(I_8, E) = { S → S ; id = E ·, $/;
E → E · + id, $/;/+ } = I_{10}

goto(I_8, id) = { E → · id ·, $/;/+ } = I_7

goto(I_9, id) = { E → E + id ·, $/;/+ } = I_{11}

goto(I_{10}, +) = { E → E + · id, $/;/+ } = I_9

2) Tabla de análisis sintáctico

	;	=	id	+	$	S	E
I_0			d2			1	
I_1	d3				Acept		
I_2		d4					
I_3			d5				
I_4			d7			6	
I_5		d8					
I_6	r2			d9	r2		
I_7	r4			r4	r4		
I_8			d7			10	
I_9			d11				
I_{10}	r1			d9	r1		
I_{11}	r3			r3	r3		

de las R

// Gramática

1. S → S ; id = E
2. S → id = E
3. E → E + id
4. E → id

3) Realizar la cadena "x = y + z"

Stack	Entrada	Acción
$0	x = y + z $	d2
$02	= y + z $	d4

$0	~		
$02	= y + z $	d4	
$024	y + z $	d7	
$0247	t z $	r4	E → id
$0246	+ z $	d9	
$02469	z $	d11	
$024691 1	$	r3	S → S ; id = E
$01	$	Acepta	5

Derivación

Friday, 6 December 2019 2:42 PM

//Entrada

ent [5] Hola, Hola ;

~~Func ent~~ Cuqui (sin) inicio

~~escribir 5~~

~~devolver -- -fHola~~

<***>

fin