Sintaxis y Semántica de los lenguajes

# Métodos ascendentes de análisis sintáctico

2020

Facultad Regional Delta,
Universidad Tecnológica Nacional

*Prof. Juan Miguel Santos*

Métodos ascendentes de análisis sintáctico.

- Introducción.
- Forma sentencial. Forma sentencial derecha. Forma sentencial izquierda.
- Pivote.
- Algoritmo general de los analizadores sintácticos por desplazamiento y reducción.
- Relaciones de precedencia.
- Gramática de precedencia.
- Gramática de precedencia simple.
- Algoritmo de análisis sintáctico por precedencia simple.
- Gramáticas LR.
  - Prefijo viable.
  - Item. Item completo.Item válido.
  - Autómata finito no determinístico lambda para los prefijos viables de un lenguaje. (T)
  - Analizador sintáctico LR.
    - Algortimo de construcción para el AFD de un analizador sintáctico LR.
    - Función clausura. Función goto.
    - Construcción de la matriz de acción y desplazamiento para una analizador LR(0).
    - Algoritmo de un analizador sintáctico LR(0).
    - Conflictos desplazamiento-reducción y reducción-reducción.
  - Analizador sintáctico SLR.
    - Construcción de la matriz de acción y desplazamiento para una analizador SLR.
  - Analizador sintáctico LR(1).
    - Item LR(1).
    - Item válido LR(1).
    - Función clausura LR(1). Función goto LR(1).
  - Analizador sintáctico LALR(1).
    - Núcleo de un conjunto de items LR(1).
    - Construcción del AFD para un analizador LALR.
    - Persistencia de conflictos en un analizador LALR.

## Métodos de análisis sintáctico ascendentes

Estos métodos, a la inversa de como lo hacen los descendentes, reconstruyen el árbol de derivación a partir de la base hacia el nodo raíz.

Consideremos el siguiente ejemplo:
Sea G = < $V_N$, $V_T$, P, E > con P = { E $\rightarrow$ E+T / T, T $\rightarrow$ T*F / F, F$\rightarrow$ (E) / i / c }
y la cadena w = a + b * c.
Derivemos a partir de E usando la derivación más a la derecha:

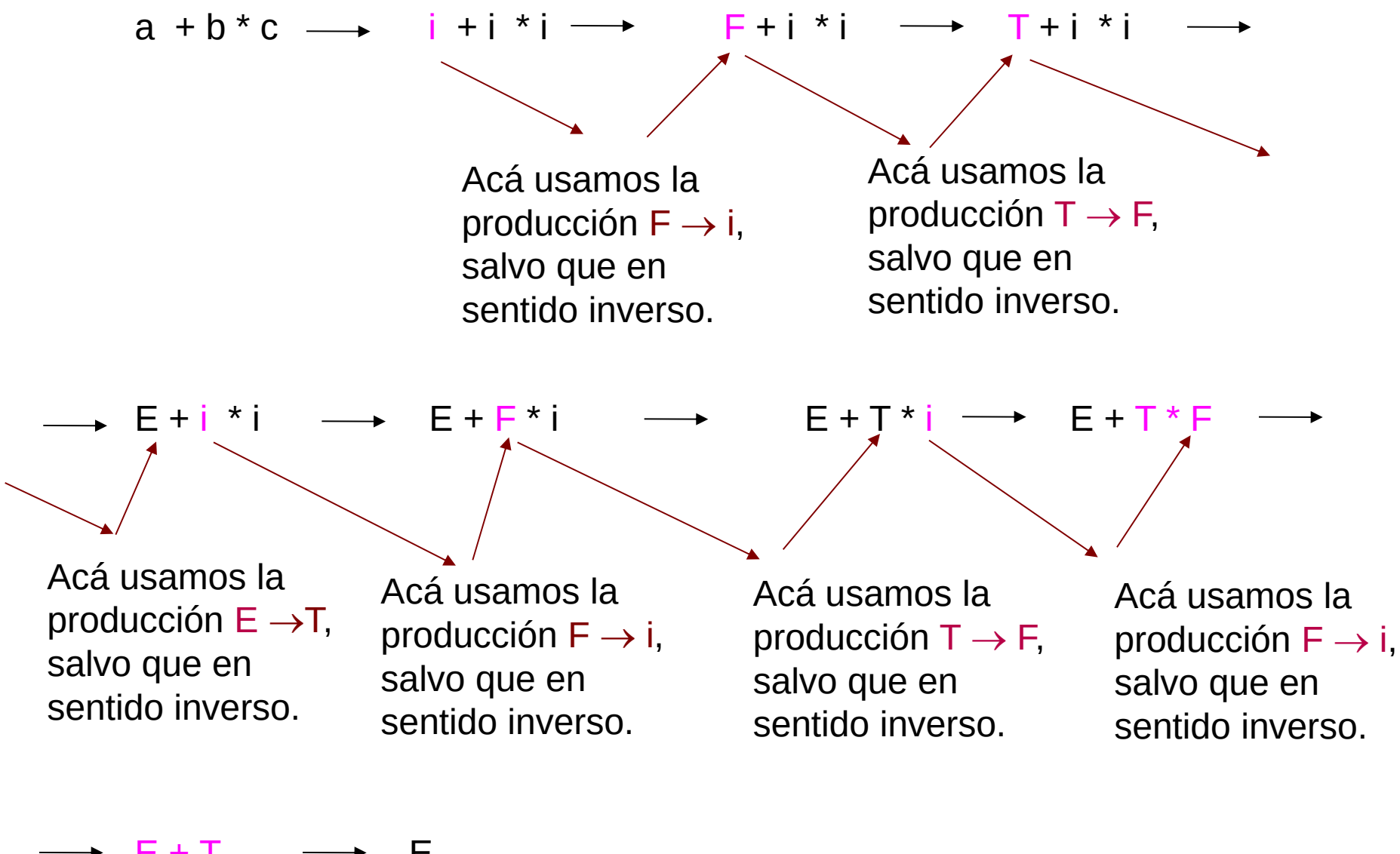
E
E + **T**
E + T * **F**
E + **T** * i
E + **F** * i
**E** + i * i
**T** + i * i
**F** + i * i
i + i * i
a + b * c

Fijarse que el último no terminal que ha sido expandido es el no terminal más a la izquierda.

Por lo tanto, si quisiera realizar el camino inverso, la primera expansión que debería intentar revertir es la última.

Una primera aproximación podría ser: “intentar identificar la parte de derecha de alguna producción y reemplazarla por la parte izquierda de la misma”.

(Veremos luego si este procedimiento siempre nos conduce o no a nuestro objetivo).


a + b * c
i + i * i
F + i * i
T + i * i
Acá usamos la producción F → i, salvo que en sentido inverso.
Acá usamos la producción T → F, salvo que en sentido inverso.
E + i * i
E + F * i
E + T * i
E + T * F
Acá usamos la producción E →T, salvo que en sentido inverso.
Acá usamos la producción F → i, salvo que en sentido inverso.
Acá usamos la producción T → F, salvo que en sentido inverso.
Acá usamos la producción F → i, salvo que en sentido inverso.
E + T
E

En la secuencia anterior se llevaron a cabo dos operaciones

**DESPLAZAMIENTO** (cuando al considerar las subcadenas desde el símbolo apuntado en la forma sentencial no es suficiente para reducir)

**REDUCCION** (cuando identificabamos lo que puede ser la parte derecha de un producción y usabamos esa misma producción pero al revés para obtener la forma sentencial anterior).

Pero **atención**, no alcanza con identificar sólo la parte derecha de una producción para poder reducir.

Definición (pivote):
Sea $G = < V_N, V_T, P, S >$ y $\alpha\rho\beta$ una forma sentencial derecha de dicha gramática, entonces $\rho$ es pivote si y sólo si

$$\exists\, (N \rightarrow \rho) \in P \wedge S \rightarrow^* \alpha N \beta \rightarrow \alpha\rho\beta,$$

donde todas las derivaciones son más a la derecha.

Determinar *cuál es el pivote* es el problema central de los analizadores sintácticos ascendentes.
Estos analizadores son llamados de desplazamiento – reducción.

Una forma de implementar este método es mediante el uso de una pila.

Cada vez que tenemos el pivote en el tope de la pila reducimos, sino desplazamos.

**Algoritmo general de un analizador por desplazamiento – reducción**.

Sea w una cadena a analizar / w = $a_1\ a_2\ ..\ a_n$

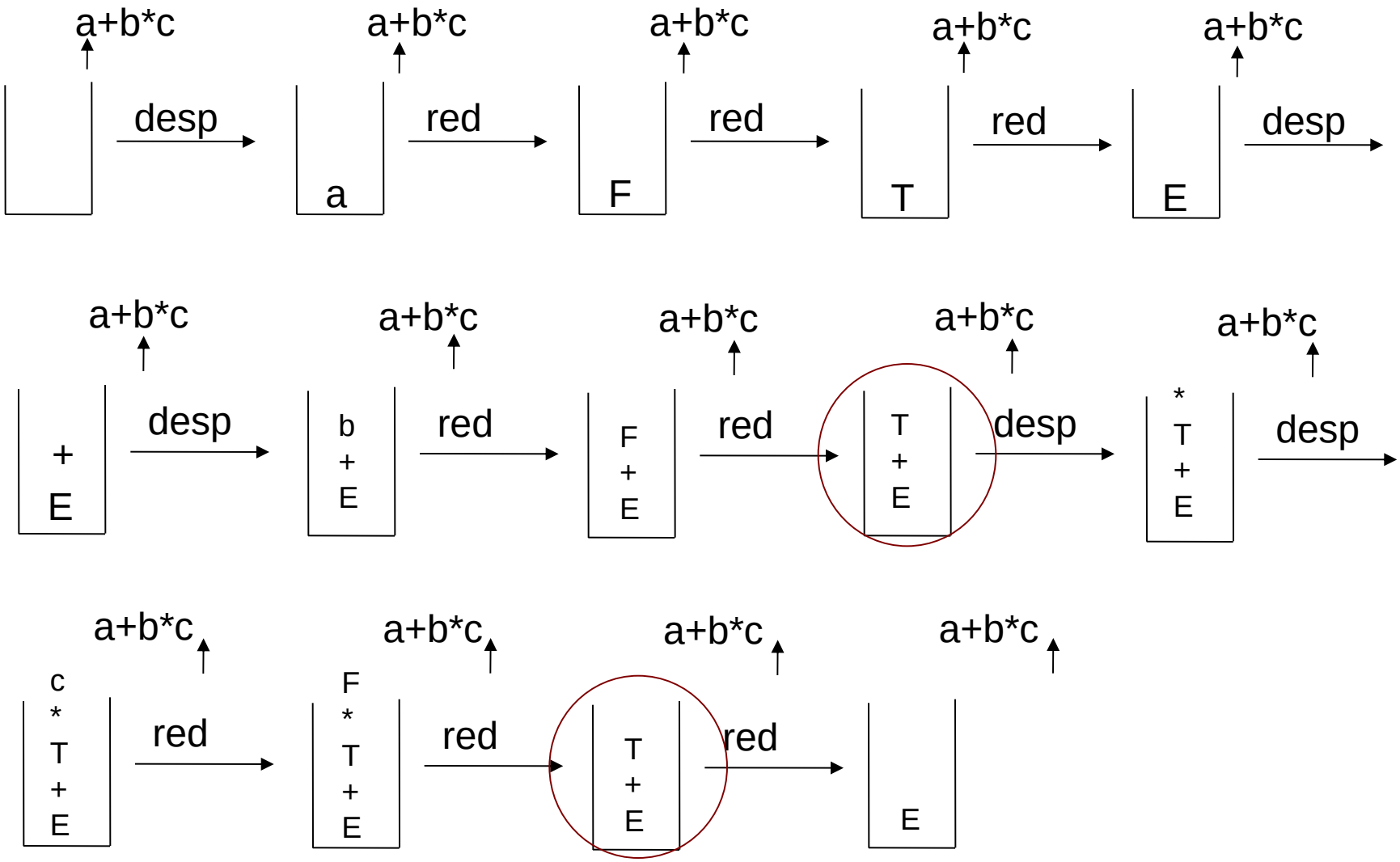
pc el símbolo apuntado en w

pila = $\lambda$

tope $\in$ V* una secuencia de símbolos en el tope de la pila

1. el_fin $\leftarrow$falso;  error $\leftarrow$ falso
2. repetir hasta el_fin
    Según
        . tope es pivote
            Desapilar pivote
            Apilar A (donde (A $\rightarrow$ pivote) $\in$ P)
        . pila ==  S $\wedge$ fin_de_cadena(w)
            Aceptar w
            el_fin $\leftarrow$ verdadero
        . pc $\neq \lambda$
            Apilar pc
            Avanzar puntero en w
        . otro caso
            error $\leftarrow$ verdadero
            el_fin $\leftarrow$ verdadero
3. Fin

Ejemplo: consideremos la misma gramática y cadena w = a + b * c del ejemplo anterior.



## Analizadores sintácticos ascendentes por precedencia

## Relaciones de precedencia

Sea G = < $V_N$, $V_T$, P, S >, para cada par de símbolos (terminales o no terminales) consecutivos en una forma sentencial derecha se puede definir una relación que llamaremos de precedencia.

Hay tres tipos de relación de precedencia “>”, “<” e “=”.

Dichas relaciones están definidas en

**=** : V x V,
**<** : V x V, y
**>** : V x $V_T$.

$\forall$ R, S $\in$V,

R > S si y sólo si $\exists \alpha$ f.s.d. / $\alpha = ..$ RS .. $\wedge$ R está en su pivote y S no.

R < S si y sólo si $\exists \alpha$ f.s.d. / $\alpha = ..$ RS .. $\wedge$ S está en su pivote y R no.

R = S si y sólo si $\exists \alpha$ f.s.d. / $\alpha = ..$ RS .. $\wedge$ ambas están en su pivote.

Ejemplo: Sea G = < $V_N$, $V_T$, P, S > con P = { S $\rightarrow$ (S) / i }.

| | ( | S | ) | i | # |
|---|---|---|---|---|---|
| ( | < | = | | < | |
| S | | | = | | |
| ) | | | > | | > |
| i | | | > | | > |
| # | < | | | < | |

Usaremos la posibilidad que las relaciones binarias pueden ser representadas en tablas de doble entrada. Las tres relaciones la representaremos en la misma tabla.

| | ( | S | ) | i | # |
|---|---|---|---|---|---|
| ( | < | = | | < | |
| S | | | = | | |
| ) | | | > | | > |
| i | | | > | | > |
| # | < | | | < | |

Consideremos la cadena

w = ( ( ( i ) ) )

< < < > > >

( ( ( S ) ) )

< < = = > >

( ( S ) )

< = = >

( S )

= =

S

Definición (relaciones de precedencia de Wirth-Weber)

Sea $G = < V_N, V_T, P, S >$

A) $< : (V_N \cup V_T) \rightarrow (V_N \cup V_T)$

$X < Y$ si $\exists (A \rightarrow \alpha X B \beta) \in P / B \rightarrow^+ Y\gamma$

b) $= : (V_N \cup V_T) \rightarrow (V_N \cup V_T)$

$X = Y$ si $\exists (A \rightarrow \alpha X Y \beta) \in P$

c) $> : (V_N \cup V_T) \rightarrow V_T$

$X > a$ si $\exists (A \rightarrow \alpha B Y \beta) \in P / B \rightarrow^+ \gamma X \wedge Y \rightarrow^* a \delta$

Nota: Consideraremos dos símbolos demarcadores denotados con # al principio y final de la cadena.

Así, $\# < X$ si $S \rightarrow^+ X \alpha$

$Y > \#$ si $S \rightarrow \alpha Y$

Definición (gramática de precedencia):

G es de precedencia si es propia y además, entre cualquier par de símbolos hay definida, a lo sumo, una relación de precedencia.

Definición (gramática de precedencia simple):

G es de precedencia simple si es de precedencia y es univocamente inversible $((A \rightarrow \alpha),(B \rightarrow \beta) \in P \Rightarrow (\alpha = \beta \Rightarrow A = B)$.

**Algortimo de análisis sintáctico para precedencia simple**.

Sea w la cadena de entrada
tc el símbolo apuntado en w
tope el tope de la pila (la pila es incializada con #)
>, < , = las relaciones de precedencia ya calculadas

Repetir
  Si tope > tc
    $\alpha \leftarrow \lambda$
    repetir ( t $\leftarrow$ tope; $\alpha \leftarrow$ t $\alpha$ ; desapilar tope ) hasta tope < t
    Si $\exists$ A / (A $\rightarrow \alpha$) $\in$ P en G
      apilar A
    sino
      retornar Falso
  Si no (tope < tc) $\wedge$ no (tope = tc)
      retornar Falso
  Si pila == #S $\wedge$ tc == #
      retornar verdadero
  En otro caso
    apilar tc
    avanzar el puntero de tc en w

Ejemplo: Sea $G = < V_N, V_T, P, S >$ con $P = \{ S \rightarrow (S) / i \}$ y la cadena

W = # ( ( ( i ) ) )#.

Definición (prefijo viable):

Sea $\alpha\rho\beta$ una forma sentencial derecha donde $\rho$ es pivote, $\gamma$ es prefijo viable si $\gamma$ es prefijo de $\alpha\rho$

Nota 1: un prefijo viable es cualquier prefijo de una forma sentencial derecha tal que su último símbolo no este a la derecha del último símbolo del pivote.

Nota 2: el contenido de la pila de un analizador sintáctico por desplazamiento-reducción siempre es un prefijo viable.

*Ejemplo*: Sea $G = < V_N, V_T, P, S >$ con

$P = \{ S \rightarrow ABC, A \rightarrow aa, B \rightarrow bb, C \rightarrow cc \}$,

¿cuáles son prefijos viables?

A AB ABC ABcc

aab Abbc Abb

aa a Ab

Definición (ítem):

Sea $A \rightarrow \alpha$ una producción en $G = < V_N, V_T, P, S >$ con $\alpha = a_1\ a_2\ ..\ a_n$, un ítem es una cadena de la siguiente forma

$$[\ A \rightarrow \alpha_1\ .\ \alpha_2\ ]$$

donde $\alpha_1$ es la cadena formada por los símbolos $a_1\ ..\ a_k$ y

$\alpha_2$ es la cadena formada por los símbolos $a_{k+1}\ ..\ a_n$.

*Ejemplo*: Sea $G = < V_N, V_T, P, S' >$ con

$P = \{S' \rightarrow Sc,\ S \rightarrow SA / A,\ A \rightarrow aSb / ab\ \}$

Los ítems de esta gramática son

$[S' \rightarrow .\ Sc\ ]$ $[S' \rightarrow S\ .\ c\ ]$ $[S' \rightarrow Sc\ .\ ]$

$[S \rightarrow .\ SA\ ]$ $[S \rightarrow S\ .\ A\ ]$ $[S \rightarrow SA\ .\ ]$

$[S \rightarrow .\ A\ ]$ $[S \rightarrow A\ .\ ]$

$[A \rightarrow .\ aSb\ ]$ $[A \rightarrow a\ .\ Sb\ ]$ $[A \rightarrow aS\ .\ b\ ]$ $[A \rightarrow aSb\ .\ ]$

$[A \rightarrow .\ ab\ ]$ $[A \rightarrow a\ .\ b\ ]$ $[A \rightarrow ab\ .\ ]$

Definición (ítem completo):

Un ítem es completo si el punto está después del último símbolo de la parte derecha de la producción que dio origen al mismo.

Definición (ítem válido):

Dado un prefijo viable $\gamma = \delta\alpha$, $[A \rightarrow \alpha . \beta]$ es un ítem válido para $\gamma$

si y sólo si $S \rightarrow^* \delta Aw \rightarrow \delta\alpha\beta w$.

*Ejemplo*: considere la gramática del último ejemplo. Los prefijos viables de SaSbc en G son

S y su ítem válido es $A \rightarrow .aSb$

Sa y su ítem válido es $A \rightarrow a.Sb$

SaS y su ítem válido es $A \rightarrow aS.b$

SaSb y su ítem válido es $A \rightarrow aSb.$

*Ejemplo*:

Sea S’ $\rightarrow^*$ Ac $\rightarrow$ abc una derivación en la gramática de los últimos dos ejemplos, vemos que el ítem [A $\rightarrow$ab.] es válido para el prefijo viable ab,

[A $\rightarrow$a.b] es válido para el prefijo viable a, y

[A $\rightarrow$.ab] es válido para el prefijo viable $\lambda$.

*Ejemplo*:

Consideremos la gramática $G = < V_N, V_T, P, S >$ que genera el lenguaje $L = \{ a^n b^n, n > 0 \}$, con $P = \{ S \rightarrow aSb / ab \}$ y consideremos la siguiente cadena perteneciente a L

aaabbb

¿Dónde está el pivote?

aa**ab**bb

¿Qué prefijos viables existen a partir de aaabbb?

a, aa, aaa y aaab.

Consideremos ahora las cadenas, siempre de L, siguientes:

ab, aabb, aaaabbbb, aaaaabbbbb

¿Qué prefijos viables existen a partir de esas cadenas?

ab, aab, aaaa, aaaab, aaaaa y aaaaab.

Si quisieramos caracterizar los prefijos viables que pueden aparecer en las cadenas analizadas de L podríamos escribir:

$$a^{+} (b/\lambda).$$

**Lo interesante de esto es que usamos una expresión regular para caracterizar dicho conjunto.**

Definición (AFND-$\lambda$ que reconoce a los prefijos viables de un lenguaje dado G):

Sea $G = < V_N, V_T, P, S >$, definimos $M = < K, V_N \cup V_T, \delta, q_0, K >$

donde K es el conjunto de items para $G \cup \{q_0\}$.

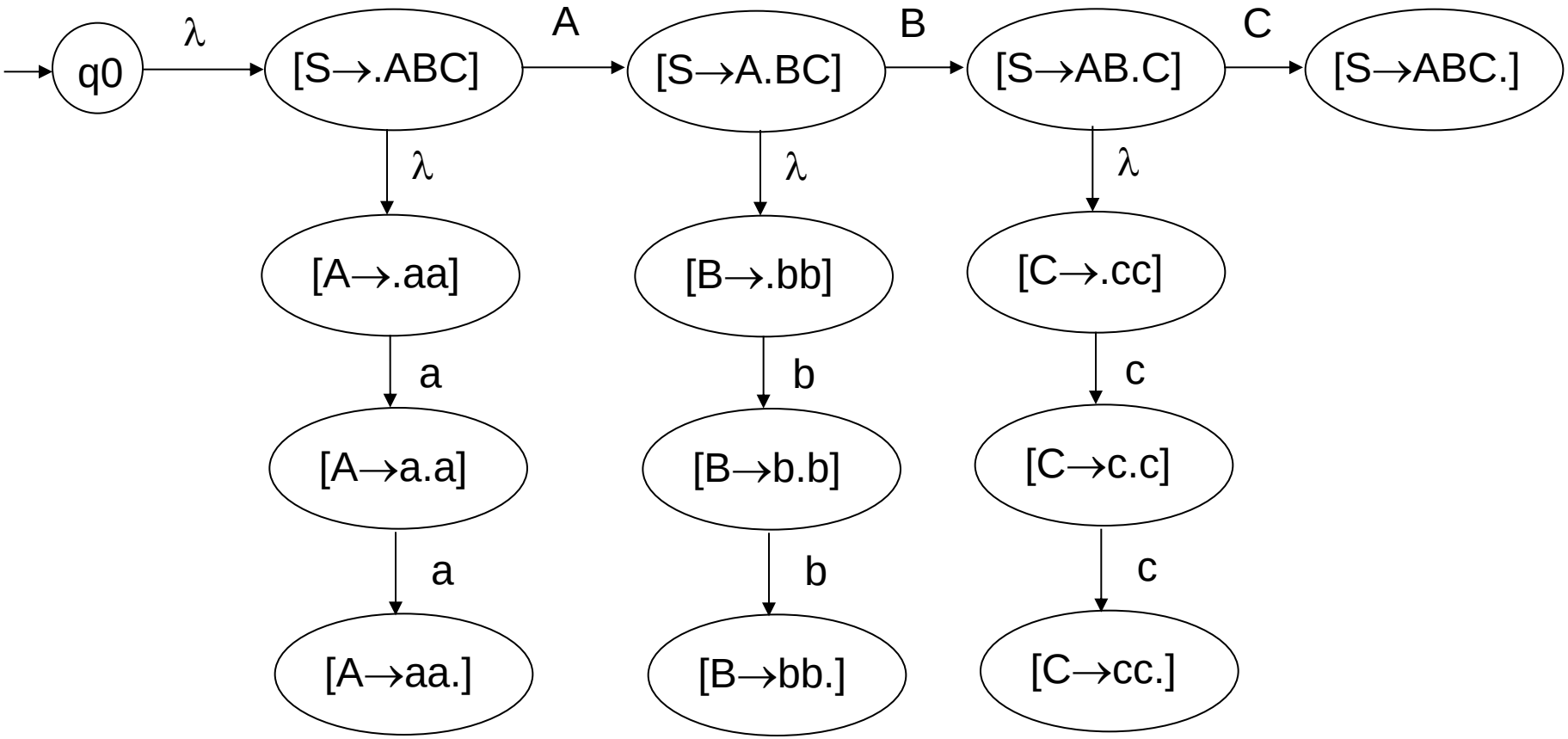
La función $\delta$ sigue

i) $\delta(q_0, \lambda) = \{ [S \rightarrow . \alpha] / S \rightarrow \alpha \in P \}$

ii) $\delta([A \rightarrow \alpha.B\beta], \lambda) = \{ [B \rightarrow . \gamma] / B \rightarrow \gamma \in P \}$

iii) $\delta([A \rightarrow \alpha.X\beta], X) = \{ [A \rightarrow \alpha X.\beta] \}$

*Ejemplo*: Sea G = < $V_N$, $V_T$, P, S > con
P = { S→ABC, A →aa, B →bb, C →cc },



Teorema:

Sea G = < $V_N$, $V_T$, P, S > y sea el AFND-$\lambda$ M como recién se definió, se cumple que

$\delta(q_0, \gamma)$ contiene a [A $\rightarrow \alpha.\beta$] si y sólo si [A$\rightarrow \alpha.\beta$] es ítem válido para $\gamma$ pv

donde $\gamma \in V^*$.

Demostración:

$\Rightarrow$) Si [A$\rightarrow \alpha.\beta$] es ítem válido para $\gamma$, por definición de ítem válido y prefijo viable debe suceder que

$S \rightarrow^* \delta Aw \rightarrow \delta\alpha\beta w$ con $\gamma = \delta\alpha$ y $(A \rightarrow \alpha\beta) \in P$ en G.

Haremos inducción en el camino de $\gamma$.

Caso base (longitud del camino es 1):

La transición estará etiquetada con $\lambda$ y por construcción del AFND- $\lambda$, $\delta(q_0, \lambda)$ contiene a [S $\rightarrow .\alpha$] y [S $\rightarrow .\alpha$] es ítem válido para $\gamma = \lambda$.

Paso inductivo (consideramos un camino de longitud k asumiendo que la hipótesis inductiva es verdadera para caminos de longitud menor a k).

El último tramo del camino puede ser 1) una transición etiquetada con algún símbolo o 2) una transición $\lambda$.

Caso 1) Si $\delta(q_0,\gamma)$ contiene a $A\rightarrow\alpha.\beta$ quiere decir que $\delta(q_0,\gamma')$ contiene a $A\rightarrow\alpha'.X\beta$ con $\alpha = \alpha'.X$ y $\gamma = \gamma' X$.

Así, $A\rightarrow\alpha'.X\beta$ es ítem válido para $\gamma'$ (por hipótesis inductiva) y por lo tanto

$\exists$ una forma sentencial derecha obtenida según

$$S \rightarrow^* \delta Aw \rightarrow \delta\ \alpha' X\beta w$$

por definición de prefijo viable, con $\alpha' X\beta$ pivote.

Luego $A \rightarrow\ \alpha' X.\beta$ es ítem válido para el prefijo viable $\delta\alpha' X$.

Pero $A \rightarrow\ \alpha' X\beta$ es $A \rightarrow\ \alpha\beta$ y $\delta\alpha' X$ es $\gamma$, entonces $A \rightarrow\ \alpha.\beta$ es ítem válido para $\gamma$ prefijo viable.

Caso 2.

$A \rightarrow \alpha.\beta$ es en realidad $A \rightarrow .\beta$ y el estado anterior en el camino es $B \rightarrow \alpha_1.A\beta_1$.

Pero como para llegar al estado anterior se sigue un camino de longitud menor a k y la última transición estuvo etiquetada con $\lambda$, resulta que

$B \rightarrow \alpha_1.A\beta_1$ es ítem válido para $\gamma$ prefijo viable.

Luego sucede que

$S \rightarrow^* \delta Bw \rightarrow \delta\alpha_1 A\beta_1 w \rightarrow \delta\alpha_1\beta\beta_1 w$ con $\beta$ pivote.

Pero entonces, $A \rightarrow .\beta$ es ítem válido para $\delta\alpha_1 = \gamma$ prefijo viable.

Lema: si $A \rightarrow .\alpha$ es ítem válido para $\gamma$ prefijo viable,

entonces $A \rightarrow .\alpha \in \delta(q_0, \gamma)$

Demostración:

$A \rightarrow .\alpha$ es ítem válido para $\gamma$ prefijo viable si y sólo si

$S \rightarrow^* \gamma A w \rightarrow \gamma\alpha w$.

Hagamos inducción sobre $\rightarrow^*$ en

$S \rightarrow^i \gamma A w \rightarrow \gamma\alpha w$.

Caso base (i = 0):

Esto quiere decir que la derivación a considerar es en realidad

$S \rightarrow \alpha$

y entonces el prefijo viable es $\lambda$.

Por construcción sabemos que

$\delta(q_0, \lambda)$ contiene a $S \rightarrow .\alpha$ ,

Luego $S \rightarrow .\alpha$ es ítem válido para $\lambda$ prefijo viable y $A \rightarrow .\alpha \in \delta(q_0, \lambda)$.

Paso inductivo (supongamos la hipótesis verdadera para i):

Sea $S \rightarrow^{i+1} \gamma A w \rightarrow \gamma \alpha w$,

lo podemos re-escribir como

$$S \rightarrow^{i} \gamma' A' w' \rightarrow \gamma' \gamma'' A w'' w' \text{ y}$$

donde $A' \rightarrow \gamma'' A w'' \in P$ en G, $\gamma = \gamma' \gamma'$ y $w = w'' w'$.

Luego, por hipótesis inductiva,

$$A' \rightarrow . \gamma'' A w'' \in \delta(q_0, \gamma'),$$

lo cual implica que $A' \rightarrow \gamma''.A w'' \in \delta(q_0, \gamma' \gamma'') = \delta(q_0, \gamma)$.

Pero como $A \rightarrow . \alpha \in \delta(A' \rightarrow \gamma''.A w'', \lambda)$ entonces

$$\delta(q_0, \gamma) \text{ contiene a } A \rightarrow .\alpha .$$

Continuación de la demostración del teorema:

$\Leftarrow$) Tenemos que $A \rightarrow \alpha.\beta$ es ítem válido para $\gamma = \mu\alpha$ prefijo viable.

y queremos ver que $A \rightarrow \alpha.\beta \in \delta(q_0, \gamma)$.

Pero dado que $A \rightarrow \alpha.\beta$ es ítem válido para $\gamma = \mu\alpha$ prefijo viable v

entonces

$A \rightarrow .\alpha\beta$ es ítem válido para $\mu$ prefijo viable.

Luego, por el lema anterior

$A \rightarrow .\alpha\beta \in \delta(q_0, \mu)$

y por lo tanto $A \rightarrow \alpha.\beta \in \delta(q_0, \mu\ \alpha) = \delta(q_0, \gamma)$.

**Construcción del autómata finito determinístico que reconoce los prefijos viables de L(G) para G = < $V_N$, $V_T$, P, S >.**

Función Clausura

Entrada: I conjunto de ítems

Salida: C conjunto de ítems

1. Hacer C $\leftarrow$ I (con sus elementos sin marcar)
2. Mientras haya algún ítem [ $A \rightarrow \alpha.B\beta$ ] sin marcar, $B \in V_N$

    Marcar el ítem

    Para cada producción $B \rightarrow \gamma$

        Si [ $B \rightarrow .\gamma$ ] $\notin$ C

            Agregar [ $B \rightarrow .\gamma$ ] a C sin marcar
3. Retornar C

Función goto

Entrada: (I conjunto de ítems, X símbolo de V)

Salida: conjunto de ítems

1. Retornar Clausura( { [ $A \rightarrow \alpha X.\beta$ ] / [ $A \rightarrow \alpha.X\beta$ ] $\in$ I })

Procedimiento Construir conjunto de ítems

1. $q_0 \leftarrow$ Clausura( { [ S' $\rightarrow$ .S$] })
2. K $\leftarrow$ { $q_0$ } sin marcar
3. Mientras haya un conjunto de ítems I $\in$ K sin marcar

    Marcar I

    Para cada x $\in$ V

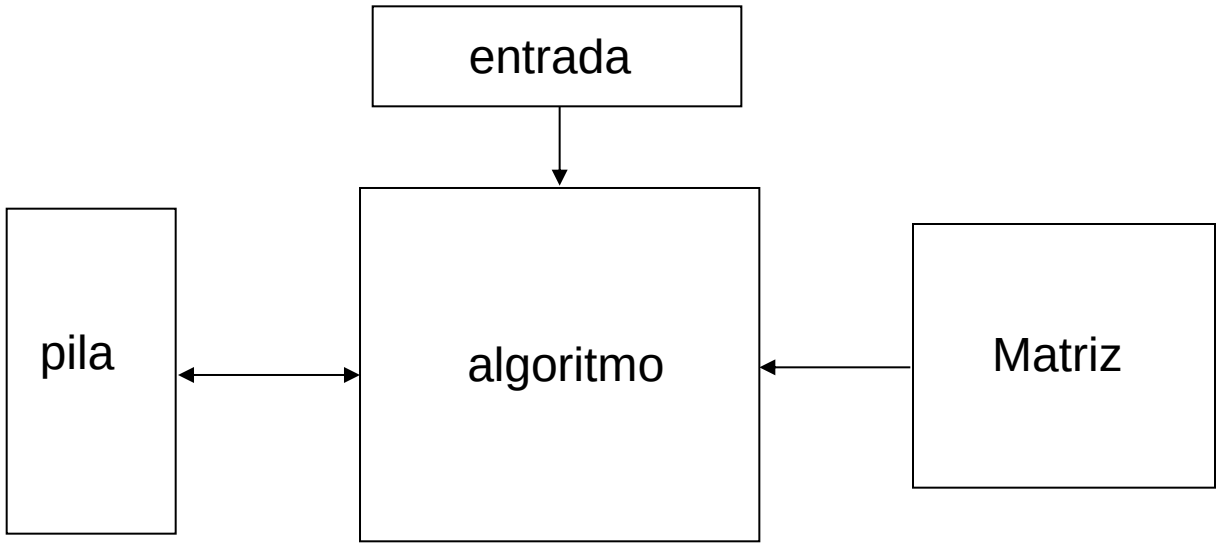
        J $\leftarrow$ goto(I, x)

        Si J $\notin$ K

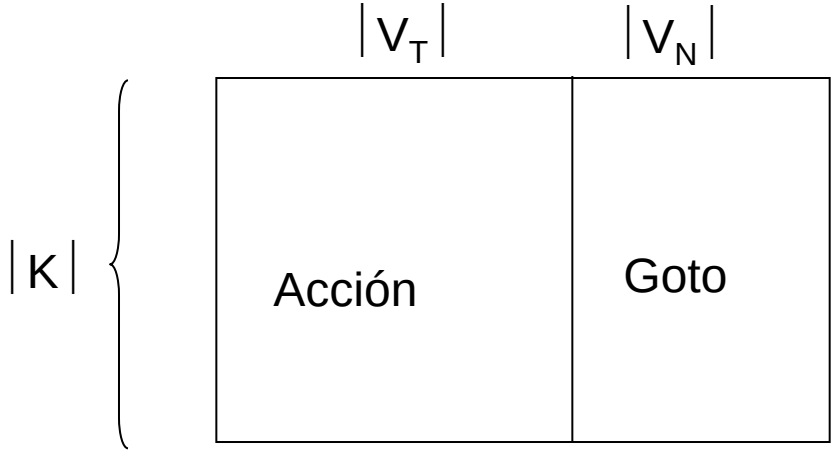
            Agregar J a K sin marcar

4. Fin

## Estructura de un analizador sintáctico LR



## Construcción de la matriz de análisis sintáctico LR(0)




$$
\text{Acc}[I, a] = \begin{cases}
\textbf{desplazar a J} & \text{si } [A \to \alpha.a\beta] \in I \wedge \text{goto}(I, a) == J \\
\textbf{reducir según } \textbf{A} \to \boldsymbol{\alpha} & \text{si } [A \to \alpha.] \in I \wedge A \neq S \\
\textbf{aceptar} & \text{si } [S' \to S.\$] \in I \wedge a = \$ \\
\textbf{error} & \text{en otro caso}
\end{cases}
$$

$$\text{Goto}[I, B] = \text{goto}(I, B)$$

**Algoritmo del analizador LR (común a todos los tipos de analizadores LR(k), SLR y LALR).**

Sea w$ la cadena de entrada

| | |
|---|---|
| tc | símbolo apuntado en w$ |
| estado_tope | el estado en el tope de la pila |
| Acc | matriz de acciones |
| Goto | matriz de goto |
| M | AFD |

1. Repetir indefinidamente
   Según Acc[estado_tope, tc]
   - desplazar q: Apilar q
     Avanzar tc
   - reducir $A \rightarrow \beta$: Desapilar $|\beta|$ veces de la pila
     q $\leftarrow$goto(estado_tope, A)
     Apilar q
   - aceptar: Fin (Aceptar)
   - error: Fin(Rechazar)

Ejemplo: Sea G = < $V_N$, $V_T$, P, S > con P = { S $\rightarrow$ (S) / a, S' $\rightarrow$ S$ }, obtener el AFD y la matriz de acción y goto.

$q_0$ = Clausura( { [ S' $\rightarrow$ .S$ ] } ) = { [ S' $\rightarrow$ .S$ ]

[ S $\rightarrow$ .(S) ],

[ S $\rightarrow$ .a ] }

¿Con qué símbolos me puedo mover desde $q_0$? Con "S", "(" y "a".

goto($q_0$, S ) = Clausura( { [ S' $\rightarrow$ S.$ ] } ) (Lo llamaremos $q_1$)

goto($q_0$, "(" ) = Clausura( { [ S $\rightarrow$ (.S) ] } ) (Lo llamaremos $q_2$)

goto($q_0$, a ) = Clausura( { [ S $\rightarrow$ a. ] } ) (Lo llamaremos $q_3$)

Calculemos cada clausura:

$q_1$ = { [ S' $\rightarrow$ S.$ ] }

$q_2$ = { [ S $\rightarrow$ (.S) ],

[ S $\rightarrow$ .(S) ],

[ S $\rightarrow$ .a ] }

$q_3$ = { [ S $\rightarrow$ a. ] }

De $q_3$ no me puedo mover. De $q_1$ tampoco.

¿Con qué símbolos me puedo mover desde $q_2$? Con “S”, “(“ y “a”.
goto($q_2$, S ) = Clausura( { [ S $\rightarrow$ (S.) ] } )  (Lo llamaremos $q_4$)
goto($q_2$, “(“ ) = Clausura( { [ S $\rightarrow$ (.S) ] } ) = $q_2$
goto($q_2$, a ) = Clausura( { [ S $\rightarrow$ a. ] } ) = $q_3$

Calculemos la clausura de $q_4$ :

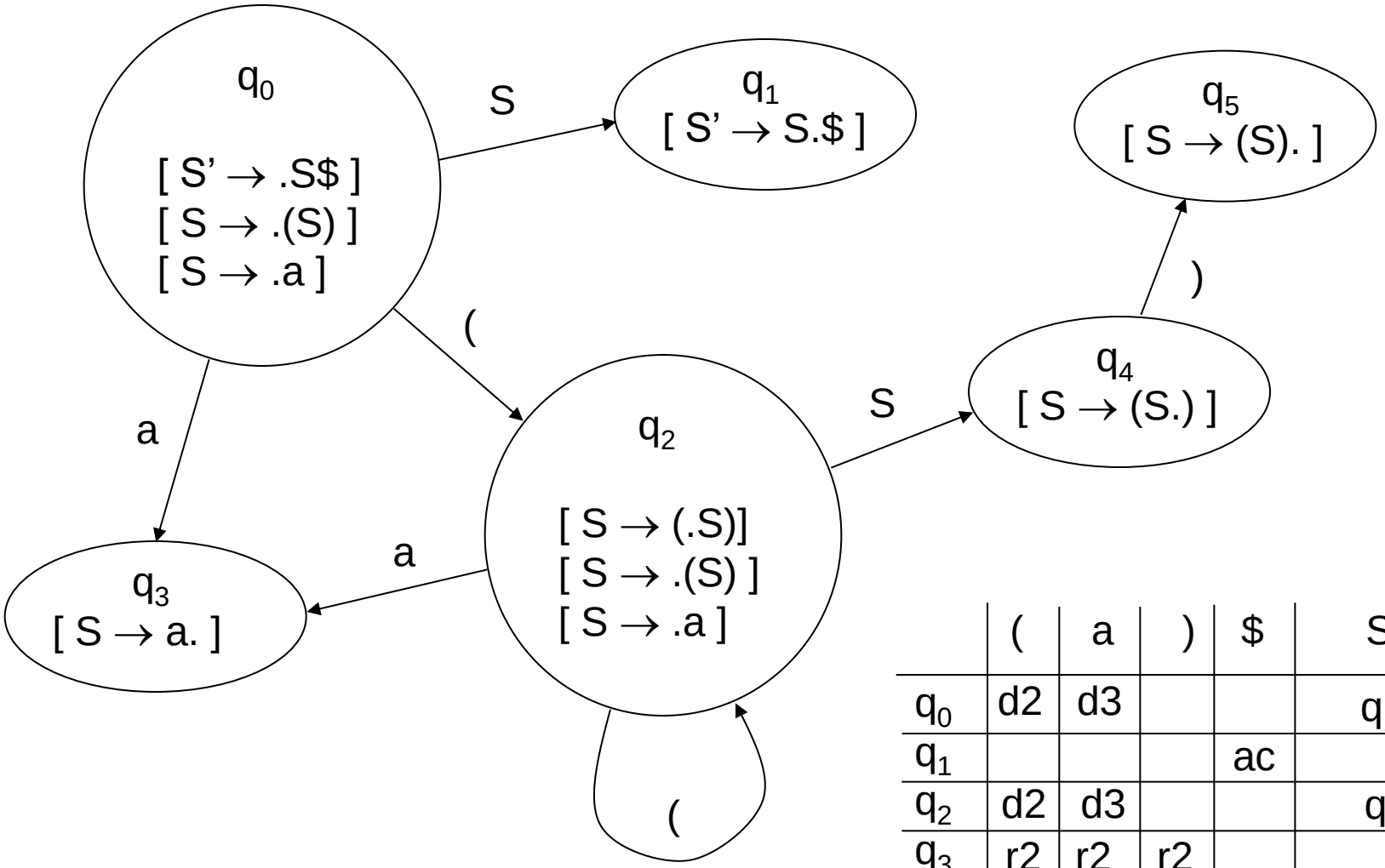
$q_4$ = { [ S $\rightarrow$ (S.) ] } .

¿Con qué símbolos me puedo mover desde $q_4$? Con “)”.
goto($q_4$, “)” ) = Clausura( { [ S $\rightarrow$ (S). ] })  (Lo llamaremos $q_5$)

Calculemos la clausura de $q_5$ :

$q_5$ = { [ S $\rightarrow$ (S). ] }

¿Cómo quedó el AFD?




| | ( | a | ) | $ | S |
|---|---|---|---|---|---|
| $q_0$ | d2 | d3 | | | $q_1$ |
| $q_1$ | | | | ac | |
| $q_2$ | d2 | d3 | | | $q_4$ |
| $q_3$ | r2 | r2 | r2 | | |
| $q_4$ | | | d5 | | |
| $q_5$ | r1 | r1 | r1 | | |

Goto[$q_0$, S ] = $q_1$
Acc[$q_0$, ( ] = d2
Acc[$q_0$, a ] = d3

Acc[$q_1$, $] = aceptar
Goto[$q_2$, S] = $q_4$
Acc[$q_2$, ( ] = d2
Acc[$q_2$, a ] = d3

Acc[$q_3$, - ] = r2

Acc[$q_4$, ) ] = d5

Acc[$q_5$, - ] = r1

| | ( | a | ) | $ | S |
|---|---|---|---|---|---|
| $q_0$ | d2 | d3 | | | $q_1$ |
| $q_1$ | | | | ac | |
| $q_2$ | d2 | d3 | | | $q_4$ |
| $q_3$ | r2 | r2 | r2 | | |
| $q_4$ | | | d5 | | |
| $q_5$ | r1 | r1 | r1 | | |

Ir al estado $q_4$

Reducir usando la producción 1

Desplazar al estado $q_2$

Producciones

1) $S \rightarrow (S)$
2) $S \rightarrow a$
3) $S' \rightarrow S\$$

Defínción (gramática LR(0)):

Una gramática es LR(0) cuando al construir la matriz de Acción, para un mismo par (estado, símbolo terminal) no hay mas de 1 acción posible.

Defínción (gramática LR(0)): (bis)

Una gramática es LR(0) si:

1) Su símbolo distinguido no aparece en la parte derecha de ninguna producción
2) Para cada prefijo viable $\gamma$, cada vez que $A \rightarrow \alpha.$ es un ítem completo válido para $\gamma$, entonces no hay otro ítem completo o no completo válidos para $\gamma$.

Nota: puede suceder que varios items no completos sean válidos para $\gamma$. Lo que no puede suceder es que simultáneamente a eso un ítem completo sea válido.

## Gramáticas SLR(1)

Si al construir la matriz de acciones para un analizador LR(0) aparecen conflictos reducción-reducción o desplazamiento-reducción, se puede considerar tener en cuenta el próximo símbolo a ser analizado (el símbolo apuntado en la cadena de entrada) para establecer si se debe reducir o no, y en tal caso decidir cuál producción usar.

Un aspecto interesante es que esto no afecta la forma de construir los conjuntos de ítems (AFD) que sigue siendo la misma que para LR(0).

Sólo hay un pequeño cambio en la construcción de la matriz de acciones:

$$
Acc[\ I, a\ ] = \begin{cases}
\textbf{desplazar a J} & \text{si } [A \rightarrow \alpha.\beta] \in I \wedge goto(I, a) == J \\
\textbf{reducir según } \mathbf{A \rightarrow \alpha} & \text{si } [A \rightarrow \alpha.] \in I \wedge A \neq S \wedge a \in Siguientes(A) \\
\textbf{aceptar} & \text{si } [S' \rightarrow S.\$] \in I \wedge a = \$ \\
\textbf{error} & \text{en otro caso}
\end{cases}
$$

**Gramáticas LR(1)**

Definición (ítem LR(1)):

Sea G = < $V_N$, $V_T$, P , S >, [A $\rightarrow \alpha$ . $\beta$, t ] es un ítem LR(1) si A $\rightarrow \alpha\beta$ es una producción de G y t $\in V_T$.

Nota: la idea es incluir en cada ítem el símbolo (terminal) que permite la reducción. Por ejemplo, si [B $\rightarrow \alpha$. , t] reducimos por B $\rightarrow \alpha$ si t coincide con el símbolo apuntado en la cadena de entrada.

Definición (ítem válido LR(1)):

[A $\rightarrow \alpha$ . $\beta$, t ] es un ítem válido LR(1) para $\gamma = \delta\alpha$ pv si y sólo si

$S \rightarrow^* \delta Aw \rightarrow \delta\alpha\beta w \wedge (t \in \text{Primeros}(w) \vee ( w = \lambda \wedge t = \$) )$.

El sentido de la segunda componente del ítem LR(1) es obtener con más precisión el terminal que habilita la reducción. Con los analizadores SLR hacíamos algo parecido pero el conjunto de Siguientes es inespecífico.

Las definiciones de ítem e ítem válido obligan a cambiar el algoritmo de Clausura y goto.

Función Clausura LR(1)
Entrada: I conjunto de ítems LR(1)
Salida: C conjunto de ítems LR(1)
1. Hacer C $\leftarrow$ I (con sus elementos sin marcar)
2. Mientras haya algún ítem [ A $\rightarrow \alpha.B\beta$, t ] sin marcar, B $\in V_N$
    Marcar el ítem
    Para cada producción B$\rightarrow \gamma$
        Para cada a $\in$ Primeros($\beta$ t)
            Si [ B$\rightarrow .\gamma$, a ] $\notin$ C
            Agregar [ B$\rightarrow .\gamma$, a ] a C sin marcar
3. Retornar C

Función goto LR(1)
Entrada: (I conjunto de ítems LR(1), X símbolo de V)
Salida: conjunto de ítems LR(1)
1. Retornar Clausura( { [ A$\rightarrow\alpha$X.$\beta$, t ] / [ A$\rightarrow\alpha$.X$\beta$, t ] $\in$ I })

**Gramática LALR**

Un problema de las gramáticas LR(1) es que el cardinal del conjunto de estados del AFD M para sus prefijos viables puede ser excesivamente grande. Una forma de tratar con esta cuestión es crear otro AFD M' a partir de M con un conjunto de estados cuyo cardinal sea menor.

Esto se puede lograr agrupando en clases a los estados de M.

Definición:

Los conjuntos de ítems $I_1$ e $I_2$ tienen el mismo núcleo si y sólo si:

$$\forall\ A \rightarrow \alpha.\beta \Rightarrow (\exists\ t\ /\ [A \rightarrow \alpha.\beta,\ t\ ] \in I_1 \Leftrightarrow \exists\ t'\ /\ [A \rightarrow \alpha.\beta,\ t'] \in I_2).$$

Esto es notado como $I_1 \equiv I_2$.

La relación $\equiv$ es de equivalencia y la partición inducida da lugar al conjunto de estados LALR. Es decir, cada clase será un estado del AFD de la gramática LALR.

Notación: Si $[ A \rightarrow \alpha.\beta, t_1 ], [ A \rightarrow \alpha.\beta, t_2 ], \ldots, [ A \rightarrow \alpha.\beta, t_k ]$ entonces notamos $[ A \rightarrow \alpha.\beta, \{ t_1, t_2, .., t_k \} ]$

**Construcción del conjunto de estados del AFD para un analizador LALR**

1. Construir el conjunto de estados K del AFD LR(1).
2. Construir un nuevo conjunto de estados K' de la siguiente forma:

   Si dos estados $k_i$, $k_j$ en K tiene todos sus ítems respectivamente con el mismo núcleo

   Crear un nuevo estado I en K' y para cada $A \rightarrow \alpha.\beta$ en $k_i$

   agregar a I el ítem $[ A \rightarrow \alpha.\beta, U ]$

   donde U es el conjunto de todos los t / $[ A \rightarrow \alpha.\beta, t ]$ esten en $k_i$ o $k_j$.

**Conflictos LALR**

Al construir un AFD M' (para un analizador LALR) a partir de un AFD M correspondiente a una gramática LR(1) no se agregan conflictos desplazamiento-reducción ya que estos dependen de los núcleos de los ítems.

En cambio sí pueden aparecer conflictos reducción-reducción que antes no estaban presentes en el analizador LR(1):