ANALIZADORES SINTÁCTICOS $LR(k)$

Víctor Manuel Darriba Bilbao

TALF
3º Grado de Informática
darriba@uvigo.es

13 de marzo de 2022

1 Introducción

- $LR(k)$: Familia de gramáticas independientes del contexto para la que pueden construírse analizadores deterministas ascendentes salto-reducción.
  L: *Left-to-right scan* (lectura de izquierda a derecha).
  R: *Rightmost derivation* (derivación derecha).
  k: tamaño del acarreo (nº de símbolos de anticipación).
- Que el algoritmo sea determinista significa que sólo puede haber una alternativa de análisis (acción) en cada paso del algoritmo.
- Que el algoritmo sea ascendente quiere decir que el análisis (la cadena de derivaciones y/o el árbol de análisis) se construye partiendo de los símbolos terminales hacia el símbolo raíz de la gramática.
- Que el algoritmo sea de salto-reducción quiere decir que en función del contenido de la pila y los k primeros símbolos de la porción de cadena de entrada por analizar, el analizador puede realizar las siguientes acciones:
  - SALTO: se mete en la cima de la pila el primer símbolo de la entrada restante.
  - REDUCCIÓN (de una regla $A \to \beta$): se sustituye β por A en la cima de la pila.
  - ACEPTAR: la cadena se ha reconocido con éxito.
  - ERROR: la cadena no pertenece al lenguaje generado por la gramática.
- Además del análisis $LR(k)$ vamos a considerar dos simplificaciones:
  - $SLR(k)$: Simple $LR(k)$.
  - $LALR(k)$: Look-ahead $LR(k)$.
- Relación entre $SLR(k)$, $LALR(k)$ y $LR(k)$:
  - Cobertura: $SLR(k) < LALR(k) < LR(k)$.
  - Tamaño analizador: $SLR(k) < LALR(k) < LR(k)$.
- Lo habitual es usar analizadores con $k = 1$.

2 Definición de Gramática *LR(k)*

Definición (Gramática Aumentada)

Sea $\mathcal{G} = \{N, \Sigma, P, S\}$ GIC, la gramática aumentada de $\mathcal{G}$ es otra GIC $\mathcal{G}' = \{N', \Sigma, P', S'\}$ con $N' = N \cup \{S'\}$, $P' = P \cup \{S' \to S\}$ y $S' \notin N$.
(Se añade un nuevo axioma S' y la regla $S' \to S$ para determinar el final del análisis) $\diamond$

Definición (Gramática $LR(k)$)

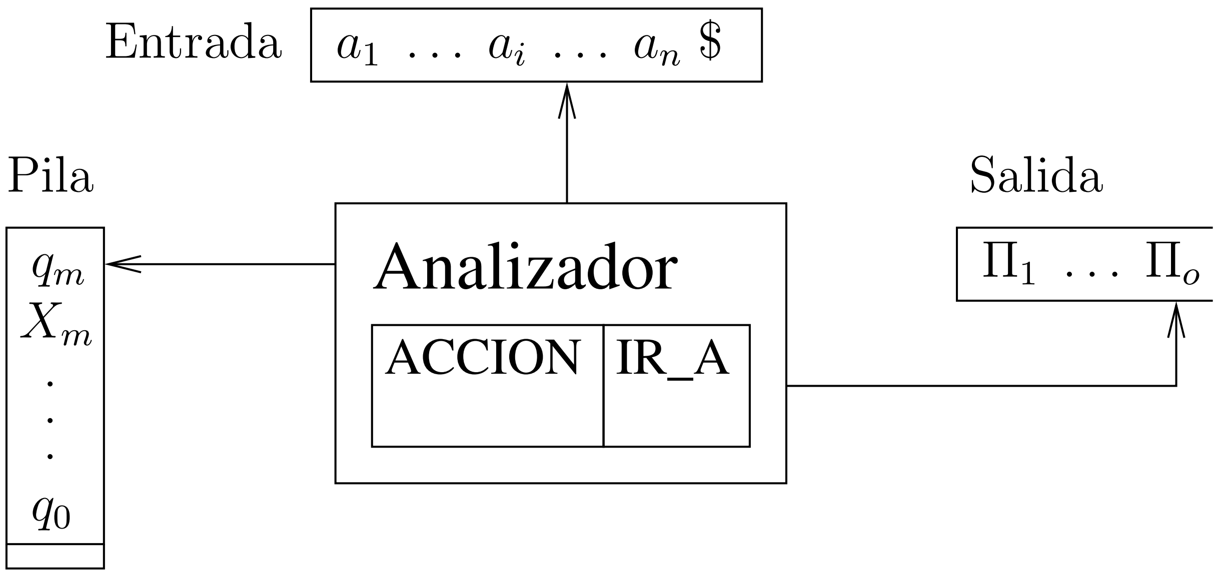
Sea $\mathcal{G} = \{N, \Sigma, P, S\}$ GIC y $\mathcal{G}' = \{N', \Sigma, P', S'\}$ su gramática aumentada. Decimos que $\mathcal{G}$ es $LR(k)$, $k \geq 0$ si y sólo si:

$$\begin{cases} S' \underset{\mathcal{G}'rm}{\overset{*}{\Rightarrow}} \alpha\, A\, w \underset{\mathcal{G}'rm}{\Rightarrow} \alpha\, \beta\, w \\ S' \underset{\mathcal{G}'rm}{\overset{*}{\Rightarrow}} \gamma\, B\, x \underset{\mathcal{G}'rm}{\Rightarrow} \alpha\, \beta\, y \\ FIRST_k(w) = FIRST_k(y) \end{cases} \Rightarrow \begin{cases} \alpha = \gamma \\ A = B \\ x = y \end{cases}$$

$\diamond$

- La explicación de la definición anterior es que si tenemos 2 derivaciones:
$$\begin{cases} \alpha\, A\, w \underset{\mathcal{G}'rm}{\Rightarrow} \alpha\, \beta\, w \\ \alpha\, A\, y \underset{\mathcal{G}'rm}{\Rightarrow} \alpha\, \beta\, y \end{cases}$$
que comparten un prefijo común $\alpha\, \beta$ y el mismo acarreo de longitud k ($FIRST_k(w) = FIRST_k(y)$) en ambos casos la reducción y el contexto izquierdo son los mismos ($A = B$ y $\alpha = \gamma$).
- Por lo tanto, conociendo $\alpha\beta$ y los primeros k símbolos de w podemos determinar que la única regla que se puede reducir es $A \to \beta$.

3 Analizador sintáctico *LR(k)*



- Q es el conjunto finito de estados del analizador (generalmente representados mediante números, con 0 como estado inicial).
- La pila acumula simbolos $X_i \in N' \cup \Sigma$ y estados q_i (el estado en el que se reconoció el símbolo).
- El intérprete LR es el mismo para todos los algoritmos de la familia (LR canónico, SLR y $LALR$). Los algoritmos se diferencian en el método de cálculo de las tablas $ACCION$ e IR_A:
  - $ACCION$: determina el tipo de movimiento (salto, reducir, aceptar o error).
  - IR_A: determina el nuevo estado destino del salto.

Configuración de un analizador *LR(k)*

- Se define como una tupla con el contenido de la pila, la porción de entrada por analizar y las reglas ya reducidas.

$$(q_0 X_1 q_1 X_2 q_2 \ldots X_m q_m, a_i a_{i+1} \ldots a_n\$, \Pi_1 \ldots \Pi_o)$$

3 Analizador sintáctico *LR(k)*

Pseudocódigo del Analizador

Entrada:	Tablas $LR(k)$ para $\mathcal{G} = \{N, \Sigma, P, S\}$ GIC $LR(k)$
	Cadena a a analizar
Salida:	Si $a \in L(\mathcal{G})$ derivación derecha, sino $ERROR$

1. Recuperar el acarreo u de longitud k y el estado actual q_m.
2. Consultar la tabla $ACCION$ usando q_m y u
   a) Si $ACCION(q_m, u) = SALTO$
      - Saltar sobre a_i, primer símbolo de la cinta de entrada
      - Identificar el estado destino del salto $IR\_A(q_m, a_i) = q_j$
      - Introducir a_i y q_j en la cima de la pila
      - Volver a 1

      $$(q_0X_1q_1 \ldots X_mq_m, a_ia_{i+1} \ldots a_n\$, \Pi_1 \ldots \Pi_o) \vdash$$
      $$(q_0X_1q_1 \ldots X_mq_ma_iq_j, a_{i+1} \ldots a_n\$, \Pi_1 \ldots \Pi_o)$$

   b) Si $ACCION(q_m, u) = REDUCIR\ \ l$, con $l \equiv A \to \alpha$
      - Eliminar $2r$ símbolos de la pila, con $r = |\alpha|$
      - Calcular el nuevo estado $IR\_A(q_{m-r}, A) = q_j$
      - Empilar A y q_j
      - Añadir l a la cinta de salida
      - Volver a 1

      $$(q_0X_1q_1 \ldots X_mq_m, a_ia_{i+1} \ldots a_n\$, \Pi_1 \ldots \Pi_o) \vdash$$
      $$(q_0X_1q_1 \ldots X_{m-r}q_{m-r}Aq_j, a_ia_{i+1} \ldots a_n\$, \Pi_1 \ldots \Pi_ol)$$

   c) Si $ACCION(q_m, u) = ERROR$, detener el análisis
   d) Si $ACCION(q_m, u) = ACEPTAR$, fin del análisis y se devuelve el conjunto de reglas

3 Analizador sintáctico *LR(k)*

Ejemplo.- Analizador $LR(1)$ para la siguiente gramática:

$$\mathcal{G}: \quad \begin{array}{l} S \to SaSb \\ S \to \varepsilon \end{array} \qquad\qquad \mathcal{G}': \quad \begin{array}{ll} (0) & S' \to S \\ (1) & S \to SaSb \\ (2) & S \to \varepsilon \end{array}$$

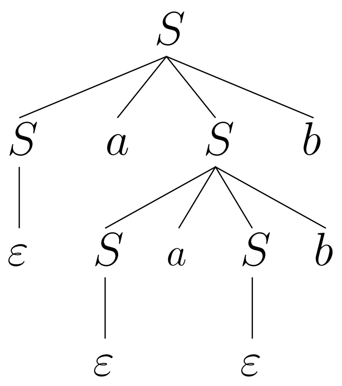
Para $k = 0, 1$ se superponen las dos tablas con la siguiente notación:
$Sl \to$ salto a estado l
$Rl \to$ reducir regla l

Las acción de las casillas en blanco es $ERROR$

	ACCIÓN			IR_A
	a	b	$\$$	S
0	$R2$		$R2$	1
1	$S2$		$ACEPTAR$	
2	$R2$	$R2$		3
3	$S5$	$S4$		
4	$R1$		$R1$	
5	$R2$	$R2$		6
6	$S5$	$S7$		
7	$R1$	$R1$		

Análisis de $w = aabb$

$$\begin{array}{l}
(0, aabb\$, \varepsilon) \underset{R2}{\vdash} (0S1, aabb\$, 2) \underset{S2}{\vdash} (0S1a2, abb\$, 2) \underset{R2}{\vdash} \\
(0S1a2S3, abb\$, 22) \underset{S5}{\vdash} (0S1a2S3a5, bb\$, 22) \underset{R2}{\vdash} \\
(0S1a2S3a5S6, bb\$, 222) \underset{S7}{\vdash} (0S1a2S3a5S6b7, b\$, 222) \underset{R1}{\vdash} \\
(0S1a2S3, b\$, 2221) \underset{S4}{\vdash} (0S1a2S3b4, \$, 2221) \underset{R1}{\vdash} \\
(0S1, \$, 22211) \vdash ACEPTAR
\end{array}$$



4 Construcción de Analizadores $LR(k)$

→ Vamos a ver cómo se construye.

Definición (Prefijo viable)

Sea $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha\ A\ w \underset{rm}{\Rightarrow} \alpha\ \beta\ w$ derivación derecha en $\mathcal{G} = \{N, \Sigma, P, S\}$ GIC, decimos que γ es un prefijo viable de $\mathcal{G}$ si $\gamma \leq \alpha\ \beta$, es decir, si $\exists \delta / \gamma\ \delta = \alpha\ \beta$. $\diamond$

Definición (Item $LR(k)$)

Sea $\mathcal{G} = \{N, \Sigma, P, S\}$ GIC, decimos que $[A \rightarrow \beta_1 . \beta_2, u]$ es un item $LR(k)$ para $\mathcal{G}$, con $A \rightarrow \beta_1\ \beta_2 \in P$ y $u \in (\Sigma \cup \{\$\})^k$. $\diamond$

→ Vamos a tener una cadena con puntos y un acarreo.

- Si $\boxed{\beta_2 \neq \varepsilon,}$ al acarreo u no tendrá efecto.
- Si $\boxed{\beta_2 = \varepsilon,}$ se podrá reducir la regla $A \rightarrow \beta_1$ sólo si los siguientes k símbolos de la cinta de entrada son u.

Lo siguiente que tenemos que hacer es reconocer el símbolo S. → 2 acarreos ≠ para 2 items ≠.

Ejemplos.- $[A \rightarrow a.S, b]$ $[S' \rightarrow .S, \$]$ $[B \rightarrow cD.\ , \underbrace{a|b}_{\text{acarreo}}]$ $\underbrace{E \rightarrow T.}_{\text{item LR 0}}$

Para las reglas $A \rightarrow \varepsilon$, son equivalentes:

Para las reglas vacías

$$[A \rightarrow .\varepsilon, u] \equiv [A \rightarrow \varepsilon.\ , u] \equiv [A \rightarrow .\ , u]$$

Definición (Item $LR(k)$ válido) → combinación de las 2 anteriores.

Un ítem $LR(k)$ $[A \rightarrow \beta_1 . \beta_2, u]$ es válido para $\alpha\ \beta_1$, prefijo viable de $\mathcal{G}$, si y sólo si existe una derivación de la forma: $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha\ A\ w \underset{rm}{\Rightarrow} \alpha\ \beta_1 \beta_2\ w$ tal que $u = FIRST_k(w)$. $\diamond$

Un ítem es válido para $\mathcal{G}$ si representa una alternativa válida para continuar el análisis. Es decir:

- Representa el análisis de $\alpha\ \beta_1$.
- El acarreo u permite que β_2 sea una alternativa para la continuación del análisis.

4 Construcción de Analizadores $LR(k)$

- Ejemplo:

$$S \rightarrow C \qquad C \rightarrow a\ C \qquad D \rightarrow a\ D$$
$$\mid D \qquad\qquad \mid b \qquad\qquad\qquad \mid b$$

$[C \rightarrow a.C, \varepsilon]$ es un ítem válido para "aaa" dado que

$$S \underset{rm}{\overset{*}{\Rightarrow}} aaC \underset{rm}{\Rightarrow} a\ a\ a\ C$$

con $\alpha = aa$, $\beta_1 = a$ y $\beta_2 = C$.

- Para reconocer los prefijos viables y sus ítems válidos correspondientes se utiliza un autómata de estado finito, a partir del cual se obtendrán las tablas de análisis.

Puede ser que haya símbolos no terminales.

el CIERRE nos ayudará a reconocer los ítems de la derecha de la regla.

Operación $CIERRE(I)$

- Sea I un conjunto de items $LR(k)$,
  1. Todo ítem de I estará en $CIERRE(I)$,
  2. Si $[A \rightarrow \alpha.B\beta, u] \in CIERRE(I)$ y $B \rightarrow \gamma \in P$, entonces $[B \rightarrow .\gamma, w] \in CIERRE(I)$, $\forall w \in FIRST_k(\beta u)$. ⇒ buscamos en la gramática donde se cumpla

ítem

- Para $k = 0$ el punto 2. se reduce a:

↳ no tenemos acarreo

$$\left.\begin{array}{l} \forall A \rightarrow \alpha.B\beta \in CIERRE(I) \\ \forall B \rightarrow \gamma \in P \end{array}\right. \Rightarrow B \rightarrow .\gamma \in CIERRE(I)$$

- Ejemplo:

$$E' \rightarrow E \qquad E \rightarrow E + T \qquad T \rightarrow T * F \qquad F \rightarrow (E)$$
$$\mid T \qquad\qquad\quad \mid F \qquad\qquad\quad \mid id$$

Calcular $CIERRE(I)$ para $LR(1)$, con $I = \{[E \rightarrow .E + T, \$]\}$.

miro si tiene símbolo no terminal. ⇒ hay 2 reglas, generando nuevos ítems

$$CIERRE(I) = \{[E \rightarrow .E + T, \$],\ [E \rightarrow .E + T, +],$$
$$[E \rightarrow .T, +],\ [T \rightarrow .T * F, +],\ [T \rightarrow .F, +],\ [T \rightarrow .T * F, *],$$
$$[T \rightarrow .F, *],\ [F \rightarrow .(E), +],\ [F \rightarrow .id, +],\ [F \rightarrow .(E), *],\ [F \rightarrow .id, *]\} =$$
$$\{[E \rightarrow .E + T, \$|+],\ [E \rightarrow .T, +],\ [T \rightarrow .T * F, +|*],\ [T \rightarrow .F, +|*],$$
$$[F \rightarrow .(E), +|*],\ [F \rightarrow .id, +|*]\}$$

Ejemplo.

- $[E \to .E + T, \$]$
  - $\to [E \to .E * T, +]$ ← como ocurre
    - $\to [E \to .E + T, +]$
    - $\text{FIRST}(+T\$) = \{+\}$
    - $\to \cancel{[E \to .T, +]}$ ⇒ no sirve para nada porque nos ha llevado a donde estábamos.
  - $\text{FIRST}(+T\$) = \{+\}$
  - $\to [E \to .T, +]$ ← como ocurre
    - $\to [T \to .T * F, +]$
      - $\to [T \to .T * F, *]$
        - $\to [T \to .T * F, *]$
        - $\text{FIRST} = *$
        - $\to [T \to .F, *]$
        - items que ya tenemos.
      - $\text{FIRST}(*F+) = \{*\}$
      - $\to [T \to .F, *]$
        - $\to [F \to .(E), *]$
        - $\text{FIRST}(*) \to \{*\}$
        - $\to [F \to .id, *]$
    - $\text{FIRST}(E+) = \text{FIRST}(+) = \{+\}$
    - $\to [T \to .F, +]$
      - $\to [F \to .(E), +]$
      - $\text{FIRST}(\varepsilon+) = \text{FIRST}(+) = \{+\}$
      - $\to [F \to .id, +]$

4 Construcción de Analizadores $LR(k)$

Operación *IR_A(I,X)*

- Sean I conjunto de items $LR(k)$ y $X \in N \cup \Sigma$ un símbolo de $\mathcal{G}$, el conjunto $IR\_A(I, X)$ se define como:

$$IR\_A(I, X) = CIERRE(\{[A \to \alpha X.\beta, u]/[A \to \alpha.X\beta, u] \in I\})$$

  es decir, para cada item $[A \to \alpha.X\beta, u] \in I$ generamos un nuevo item $[A \to \alpha X.\beta, u] \in IR\_A(I, X)$, y calculamos su $CIERRE()$.

- Ejemplo:

$$\begin{array}{llll} E' \to E & E \to E + T & T \to T * F & F \to (E) \\ & \quad\mid T & \quad\mid F & \quad\mid id \end{array}$$

  Calcular $IR\_A(I, +)$ para $LR(0)$, con $I = \{E' \to E.,\ E \to E. + T\}$.

$$\begin{array}{l} IR\_A(I, +) = CIERRE(\{E \to E + .T\}) = \\ = \{E \to E + .T, \quad T \to .T * F,\ T \to .F,\ F \to .(E),\ F \to .id\} \end{array}$$

Construcción del Autómata $LR(k)$

(Cálculo de la colección canónica *LR(k)*)

- Dada $\mathcal{G} = \{N, \Sigma, P, S\}$ GIC:
  1. Construír su gramática ampliada $\mathcal{G}'$
  2. Conjunto inicial: $I_0 = CIERRE(\{[S' \to .S,\ \$^k]\})$
  3. Para cada conjunto I_i y cada símbolo $X \in N \cup \Sigma$
     - Calcular $I_j = IR\_A(I_i, X)$
     - Almacenar el conjunto I_j en la colección

     Repetir hasta que no es posible añadir nuevos conjuntos

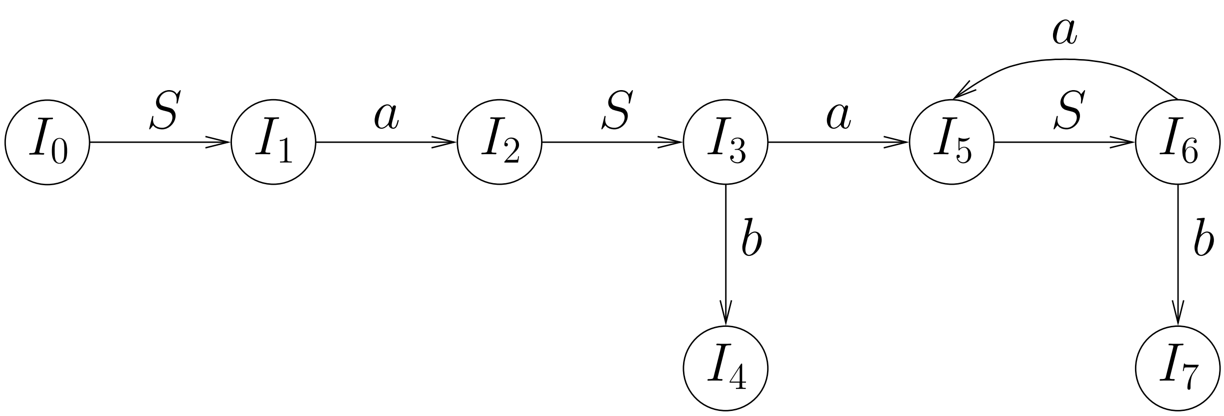
4 Construcción de Analizadores $LR(k)$

- Ejemplo:

$$\mathcal{G}: \quad \begin{array}{l} S \rightarrow SaSb \\ S \rightarrow \varepsilon \end{array} \qquad\qquad \mathcal{G}': \quad \begin{array}{ll} (0) & S' \rightarrow S \\ (1) & S \rightarrow SaSb \\ (2) & S \rightarrow \varepsilon \end{array}$$

Construír la colección canónica $LR(1)$.

$$\begin{array}{ll}
I_0 = & CIERRE(\{[S' \rightarrow .S, \$]\}) = \{[S' \rightarrow .S, \$],\ [S \rightarrow .SaSb, \$], \\
 & [S \rightarrow\ ., \$],\ [S \rightarrow .SaSb, a],\ [S \rightarrow\ ., a]\} = \{[S' \rightarrow .S, \$], \\
 & [S \rightarrow .SaSb, \$|a],\ [S \rightarrow\ ., \$|a]\} \\
I_1 = & IR_A(I_0, S) = CIERRE(\{[S' \rightarrow S., \$],\ [S \rightarrow S.aSb, \$|a]\}) = \\
 & \{[S' \rightarrow S., \$],\ [S \rightarrow S.aSb, \$|a]\} \\
I_2 = & IR_A(I_1, a) = CIERRE(\{[S \rightarrow Sa.Sb, \$|a]\}) = \\
 & \{[S \rightarrow Sa.Sb, \$|a],\ [S \rightarrow .SaSb, a|b],\ [S \rightarrow\ ., a|b]\} \\
I_3 = & IR_A(I_2, S) = CIERRE(\{[S \rightarrow SaS.b, \$|a],\ [S \rightarrow S.aSb, a|b]\}) = \\
 & \{[S \rightarrow SaS.b, \$|a],\ [S \rightarrow S.aSb, a|b]\} \\
I_4 = & IR_A(I_3, b) = CIERRE(\{[S \rightarrow SaSb., \$|a]\}) = \{[S \rightarrow SaSb., \$|a]\} \\
I_5 = & IR_A(I_3, a) = CIERRE(\{[S \rightarrow Sa.Sb, a|b]\}) = \\
 & \{[S \rightarrow Sa.Sb, a|b],\ [S \rightarrow .SaSb, a|b],\ [S \rightarrow\ ., a|b]\} \\
I_6 = & IR_A(I_5, S) = CIERRE(\{[S \rightarrow SaS.b, a|b],\ [S \rightarrow S.aSb, a|b]\}) = \\
 & \{[S \rightarrow SaS.b, a|b],\ [S \rightarrow S.aSb, a|b]\} \\
I_7 = & IR_A(I_6, b) = CIERRE(\{[S \rightarrow SaSb., a|b]\}) = \{[S \rightarrow SaSb., a|b]\} \\
 & IR_A(I_6, a) = CIERRE(\{[S \rightarrow Sa.Sb, a|b]\}) = I_5
\end{array}$$



4 Construcción de Analizadores $LR(k)$

Construcción Tablas $LR(k)$ *canónico*

1. Se contruye el autómata (colección canónica) $\underline{LR(k)}$
2. Entradas de la tabla $\boxed{ACCION}$. Recorremos cada punto de item.

   Cada conjunto I_i corresponde a un estado i:

   a) Si $[A \to \alpha\,., u] \in I_i \Rightarrow ACCION(i, u) =$ Reducir $A \to \alpha$
   b) Si $[S' \to S., \$^k] \in I_i \Rightarrow ACCION(i, \$^k) =$ Aceptar
   c) Si $[A \to \alpha.a\beta, u] \in I_i$ e $IR\_A(I_i, a) = I_j$ con $a \in \Sigma \Rightarrow$ $ACCION(i, ax) =$ Salto j, con $x = FIRST_{k-1}(\beta u)$
   d) En otro caso $ERROR$

   símbolo a continuación de punto es un símbolo terminal.

3. Entradas de la tabla IR_A.

   Se construye directamente a partir de las transiciones del autómata involucrando símbolos no terminales.

$$IR\_A(I_i, S) = I_j \Rightarrow IR\_A(i, S) = j$$

Ejemplo:

$$\mathcal{G}: \quad \begin{array}{l} S \to SaSb \\ S \to \varepsilon \end{array} \qquad \mathcal{G}': \quad \begin{array}{ll} (0) & S' \to S \\ (1) & S \to SaSb \\ (2) & S \to \varepsilon \end{array}$$

Entradas de la tabla $ACCION$:

I_0: $[S \to\ ., \$|a] \Rightarrow ACCION(0, \$) = ACCION(0, a) = R2$
I_1: $[S' \to S., \$] \Rightarrow ACCION(1, \$) = ACEPTAR$
$[S \to S.aSb, \$|a]$ e $IR\_A(I_1, a) = I_2 \Rightarrow ACCION(1, a) = S2$
I_2: $[S \to\ ., a|b] \Rightarrow ACCION(2, a) = ACCION(2, b) = R2$
I_3: $[S \to SaS.b, \$|a]$ e $IR\_A(I_3, b) = I_4 \Rightarrow ACCION(3, b) = S4$
$[S \to S.aSb, a|b]$ e $IR\_A(I_3, a) = I_5 \Rightarrow ACCION(3, a) = S5$
I_4: $[S \to SaSb., \$|a] \Rightarrow ACCION(4, \$) = ACCION(4, a) = R1$
I_5: $[S \to\ ., a|b] \Rightarrow ACCION(5, a) = ACCION(5, b) = R2$
I_6: $[S \to SaS.b, a|b]$ e $IR\_A(I_6, b) = I_7 \Rightarrow ACCION(6, b) = S7$
$[S \to S.aSb, a|b]$ e $IR\_A(I_6, a) = I_5 \Rightarrow ACCION(6, a) = S5$
I_7: $[S \to SaSb., a|b] \Rightarrow ACCION(7, a) = ACCION(7, b) = R1$

4 Construcción de Analizadores $LR(k)$

Entradas de la tabla IR_A:

$$IR\_A(I_0, S) = I_1 \Rightarrow IR\_A(0, S) = 1$$
$$IR\_A(I_2, S) = I_3 \Rightarrow IR\_A(2, S) = 3$$
$$IR\_A(I_5, S) = I_6 \Rightarrow IR\_A(5, S) = 6$$

Tablas para el analizador $LR(1)$:

	ACCIÓN			IR_A
	a	b	$\$$	S
0	$R2$		$R2$	1
1	$S2$		$ACEPTAR$	
2	$R2$	$R2$		3
3	$S5$	$S4$		
4	$R1$		$R1$	
5	$R2$	$R2$		6
6	$S5$	$S7$		
7	$R1$	$R1$		

5 Construcción de Analizadores $LALR(k)$

- Simplificación (menos estados) respecto a $LR(k)$ canónico.
- Varios modos de cálculo. Veremos el consistente en fusión de estados cuyos items sólo se diferencian en el acarreo.
- *Núcleo de un item* $LR(k)$: Regla punteada (se excluye el acarreo).

1. Se construye el autómata (colección canónica) $LR(k)$.
2. Se identifican los conjuntos cuyos items tienen los mismos núcleos.
3. Se unen dichos conjuntos:
   - El acarreo de los nuevos items es la unión de sus acarreos.
   - Las transiciones de los nuevos estados están formados por la "union" de las transiciones originales.
4. Se construye la tabla $LALR(k)$ de la misma forma que la del $LR(k)$ canónico.

- Ejemplo:

$$\begin{array}{llll} (0) & S' \to S & (2) & C \to aC \\ (1) & S \to CC & (3) & C \to b \end{array}$$

Calculamos la colección canónica $LR(1)$

$$\begin{aligned}
I_0 &= CIERRE(\{[S' \to .S, \$]\}) = \{[S' \to .S, \$],\ [S \to .CC, \$], \\
&\quad [C \to .aC, a|b],\ [C \to .b, a|b]\} \\
I_1 &= IR_A(I_0, S) = \{[S' \to S., \$]\} \\
I_2 &= IR_A(I_0, C) = \{[S \to C.C, \$],\ [C \to .aC, \$],\ [C \to .b, \$]\} \\
I_3 &= IR_A(I_0, a) = \{[C \to a.C, a|b],\ [C \to .aC, a|b],\ [C \to .b, a|b]\} \\
I_4 &= IR_A(I_0, b) = \{[C \to b., a|b]\} \\
I_5 &= IR_A(I_2, C) = \{[S \to CC., \$]\} \\
I_6 &= IR_A(I_2, a) = \{[C \to a.C, \$],\ [C \to .aC, \$],\ [C \to .b, \$]\} \\
I_7 &= IR_A(I_2, b) = \{[C \to b., \$]\} \\
I_8 &= IR_A(I_3, C) = \{[C \to aC., a|b]\} \\
&\quad IR_A(I_3, a) = I_3 \\
&\quad IR_A(I_3, b) = I_4 \\
I_9 &= IR_A(I_6, C) = \{[C \to aC., \$]\} \\
&\quad IR_A(I_6, a) = I_6 \\
&\quad IR_A(I_6, b) = I_7
\end{aligned}$$

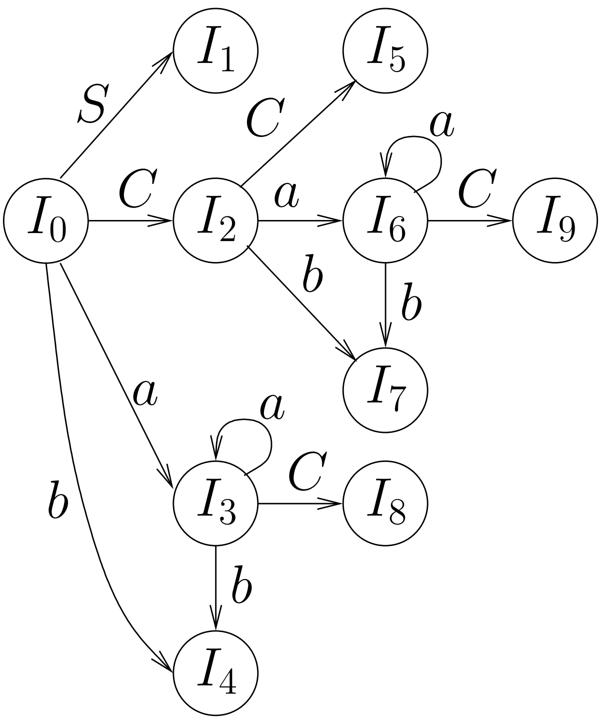
Gramática que genera secuencia de una o más as seguidas de una b.

5 Construcción de Analizadores $LALR(k)$

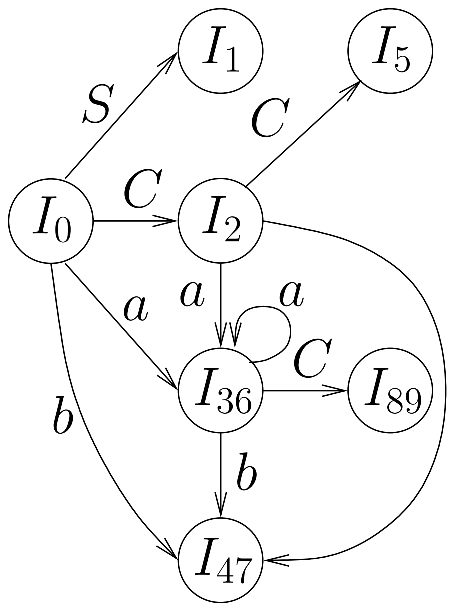
- Fusionamos los estados con ítems con el mismo núcleo:
  $I_3 - I_6$, $I_4 - I_7$ e $I_8 - I_9$

$$
\begin{array}{ll}
I_0 & = \{[S' \rightarrow .S, \$],\ [S \rightarrow .CC, \$],\ [C \rightarrow .aC, a|b],\ [C \rightarrow .b, a|b]\} \\
I_1 & = \{[S' \rightarrow S., \$]\} \\
I_2 & = \{[S \rightarrow C.C, \$],\ [C \rightarrow .aC, \$],\ [C \rightarrow .b, \$]\} \\
I_{36} & = \{[C \rightarrow a.C, a|b|\$],\ [C \rightarrow .aC, a|b|\$],\ [C \rightarrow .b, a|b|\$]\} \\
I_{47} & = \{[C \rightarrow b., a|b|\$]\} \\
I_5 & = \{[S \rightarrow CC., \$]\} \\
I_{89} & = \{[C \rightarrow aC., a|b|\$]\}
\end{array}
$$

$LR(1)$



$LALR(1)$



5 Construcción de Analizadores $LALR(k)$

- Calculamos la tabla $LALR(1)$ (aquí hemos calculado también la tabla $LR(1)$, para comparar ambas)

obtenemos la tabla de la misma manera.

Tabla $LR(1)$

	ACCIÓN			IR_A	
	a	b	$\$$	S	C
0	$S3$	$S4$		1	2
1			AC		
2	$S6$	$S7$			5
3	$S3$	$S4$			8
4	$R3$	$R3$			
5			$R1$		
6	$S6$	$S7$			9
7			$R3$		
8	$R2$	$R2$			
9			$R2$		

Tabla $LALR(1)$

	ACCIÓN			IR_A	
	a	b	$\$$	S	C
0	$S36$	$S47$		1	2
1			AC		
2	$S36$	$S47$			5
36	$S36$	$S47$			89
47	$R3$	$R3$	$R3$		
5			$R1$		
89	$R2$	$R2$	$R2$		

- Como se puede apreciar en el ejemplo, las tablas $LALR(k)$ suelen ser más pequeñas que las $LR(k)$.
- Se pierde algo de información:
  - En $LR(1)$ diferenciamos cuando se reduce una C intermendia (estados 4 y 8) o una C al final (estados 7 y 9)
  - Se añaden pasos a la detección de errores (en general $LALR(k)$ realiza más reducciones que $LR(k)$ antes de parar al encontrar un error).

6 Problemas de los Analizadores $LR(k)$

Conflictos

- Una mala elección del contexto necesario (acarreo demasiado pequeño, uso de SLR en lugar de $LALR$ o LR canónico,...) puede llevar a la aparición de **conflictos**, instancias de $ACCION(q_i, u)$ que tienen asociadas dos acciones distintas. Hay dos tipos de conflictos:
  - Conflicto Salto-Reducción
  - Conflicto Reducción-Reducción

Ambigüedad

- Definición: Sea $\mathcal{G} = \{N, \Sigma, P, S\}$ GIC, se dice que $\mathcal{G}$ es ambigua si existe alguna cadena $w \in L(\mathcal{G})$, que puede ser generada partiendo de S por **más de una** derivación por la izquierda (respectivamente por la derecha).
- Ejemplo:

$$\mathcal{G}_1 : \quad E \rightarrow E + E \quad E \rightarrow E * E \quad E \rightarrow (E) \quad E \rightarrow id$$

  Es una gramática ambigua. Por ejemplo, para la cadena $w = id + id + id$, tenemos dos posibles derivaciones por la derecha:

$$E \Rightarrow E + E \Rightarrow E + id \Rightarrow E + E + id \Rightarrow E + id + id \Rightarrow id + id + id$$

$$\begin{aligned} E &\Rightarrow E + E \Rightarrow E + E + E \Rightarrow E + E + id \Rightarrow E + id + id \Rightarrow \\ &\Rightarrow id + id + id \end{aligned}$$

- Ello no sucedería si utilizáramos una gramática no ambigua que generara el mismo lenguaje:

$$\begin{array}{llll} \mathcal{G}_2 : & E \rightarrow E + T & T \rightarrow T * F & F \rightarrow (E) \\ & E \rightarrow T & T \rightarrow F & F \rightarrow id \end{array}$$

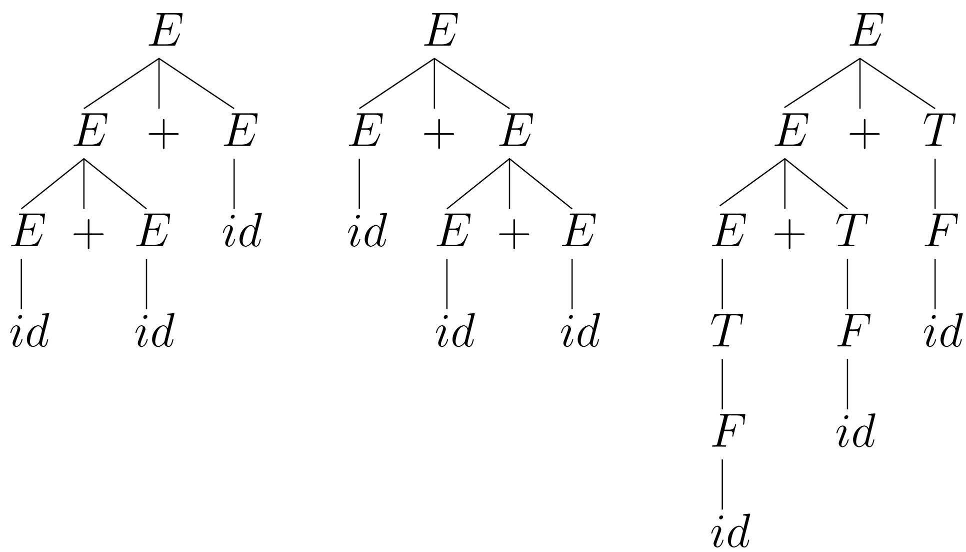
6 Problemas de los Analizadores $LR(k)$

- Para $\mathcal{G}_2$ sólo puede haber una derivación de $w = id + id + id$ por la derecha:

$$\begin{aligned} E &\Rightarrow E + T \Rightarrow E + F \Rightarrow E + id \Rightarrow E + T + id \Rightarrow E + F + id \Rightarrow \\ &\Rightarrow E + id + id \Rightarrow T + id + id \Rightarrow F + id + id \Rightarrow id + id + id \end{aligned}$$

  Las diferencias se pueden apreciar más fácilmente si dibujamos los árboles correspondientes a las derivaciones:

$\boldsymbol{\mathcal{G}_1}$ $\qquad\qquad\qquad$ $\boldsymbol{\mathcal{G}_2}$

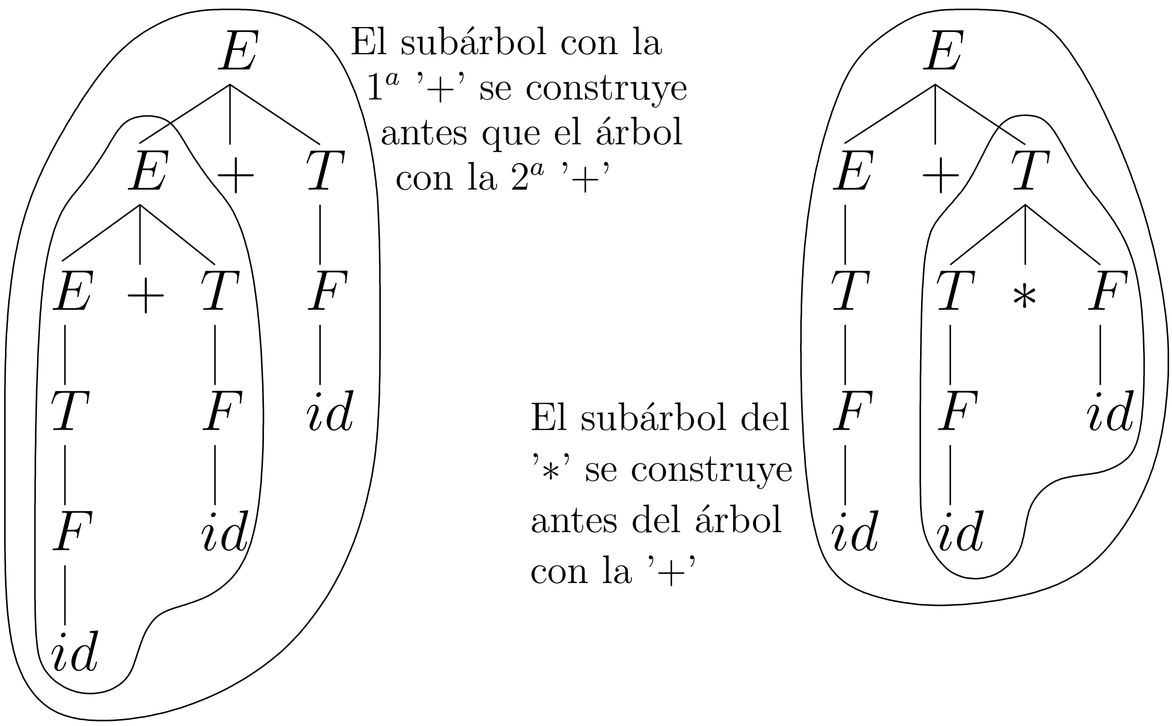


- $\mathcal{G}_2$ es no ambigua porque incluye la precedencia y asociatividad de los operadores en la propia estructura de la gramática:
  - Precedencia: El operador suma es derivado desde el axioma E ($E \to E + T$), mientras que el producto es generado desde T ($T \to T * F$), a su vez derivado desde E. Construyendo el árbol de abajo a arriba, cualquier subárbol correspondiente al producto será construído antes que el correspondiente a una suma.
  - Asociatividad: Los operadores suma y producto, asociativos por la izquierda, se definen en reglas recursivas por la izquierda. Ello significa que, de haber dos operadores suma (resp. producto) consecutivos, el subárbol de la expresión E (resp. término T) correspondiente al primero de ellos tiene que ser construído antes de poder construír el del segundo operador.

6 Problemas de los Analizadores $LR(k)$

- La explicación anterior puede entenderse más fácilmente con dos ejemplos:

'+' asociativo por la izda. '$*$' mayor precedencia que '+'



- En los analizadores $LR(k)$ la ambiguedad se manifiesta en forma de conflictos en la tabla $ACCION$.
- En muchos casos, para solventar el problema de la ambigüedad se debe rediseñar la gramática.
- Solución (en lenguajes basados en operadores). Se elige una de las acciones en cada conflicto:
  1. Se escoge la acción que permita reducir antes la regla del operador con mayor precedencia
  2. Si los dos operadores tienen la misma precedencia, se escoge la acción concordante con la asociatividad del operador (generalmente, reducir si es asociativo por la izquierda).

6 Problemas de los Analizadores $LR(k)$

- Ejemplo (gramática aumentada para $\mathcal{G}_1$):

$$\begin{array}{lll} (0)\ E' \rightarrow E & (2)\ E \rightarrow E * E & (4)\ E \rightarrow id \\ (1)\ E \rightarrow E + E & (3)\ E \rightarrow (E) & \end{array}$$

- Tabla $SLR(1)$:

	ACCIÓN						IR_A
	id	$+$	$*$	$($	$)$	$\$$	E
0	$S3$			$S2$			1
1		$S4$	$S5$			$ACEPTAR$	
2	$S3$			$S2$			6
3		$R4$	$R4$		$R4$	$R4$	
4	$S3$			$S2$			7
5	$S3$			$S2$			8
6		$S4$	$S5$		$S9$		
7		$R1/S4$	$R1/S5$		$R1$	$R1$	
8		$R2/S4$	$R2/S5$		$R2$	$R2$	
9		$R3$	$R3$		$R3$	$R3$	

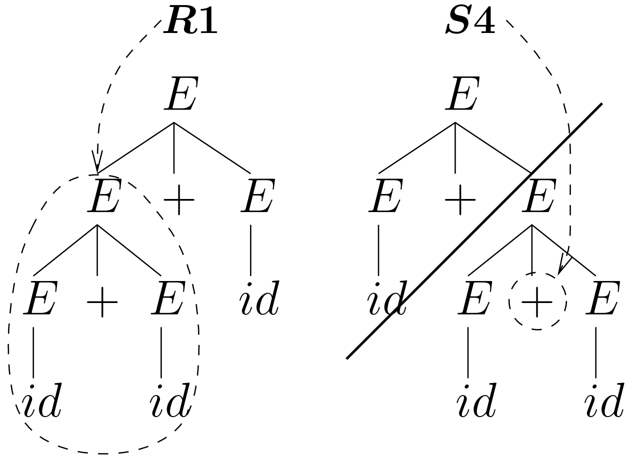
- Conflicto Salto-Reducción en $(7, +), (7, *), (8, +)$, y $(8, *)$.
- Solución: se selecciona una de las dos opciones en cada conflicto usando asociatividad y precedencia.
  - Asociatividad a la izquierda de '$*$' y '$+$'
  - Precedencia de operadores: '$*$' $>$ '$+$'

7	$ACCION(7, +) = R1$	Por asociatividad izda. de '$+$'
	$ACCION(7, *) = S5$	Por mayor precedencia de '$*$'
8	$ACCION(8, +) = R2$	Por mayor precedencia de '$*$'
	$ACCION(8, *) = R2$	Por asociatividad izda. de '$*$'

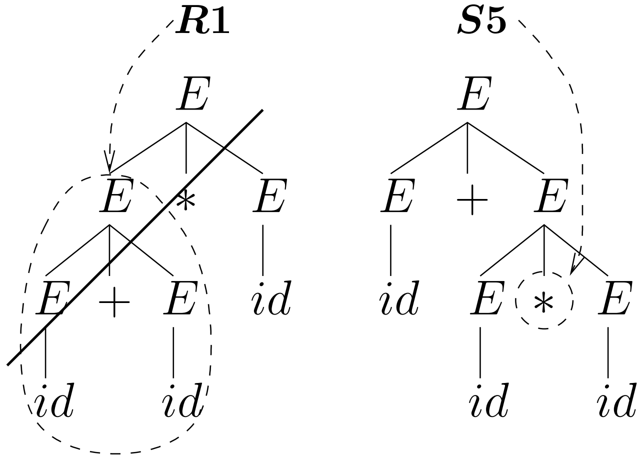
6 Problemas de los Analizadores $LR(k)$

- Ejemplos de los diferentes conflictos:
  Indicamos con las líneas de puntos el subárbol correspondiente a la regla que reduce, o el símbolo sobre el que se salta

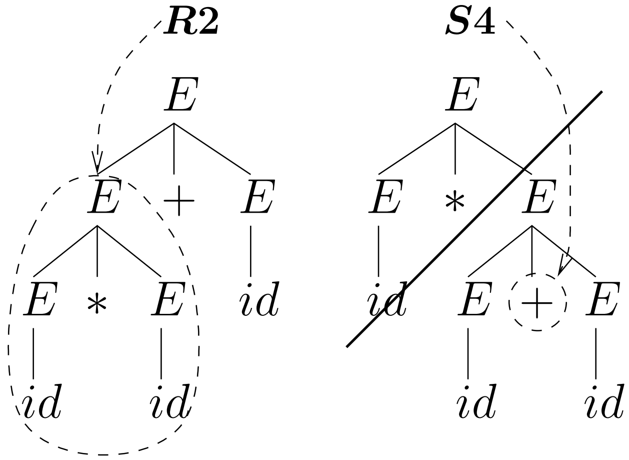
$\boldsymbol{ACCION}(7, +)$



$\boldsymbol{ACCION}(7, *)$



$\boldsymbol{ACCION}(8, +)$



$\boldsymbol{ACCION}(8, *)$