Análisis Sintáctico Ascendente

CI4721 – Lenguajes de Programación II

Ernesto Hernández-Novich
<emhn@usb.ve>

Universidad "Simón Bolívar"

Copyright ©2012-2016

UNIVERSIDAD SIMÓN BOLÍVAR

Limitaciones del reconocedor $LR(0)$

- El reconocedor debe tomar una decisión basado en lo que ha procesado hasta el punto actual en la entrada – demasiado restrictivo para gramáticas que definen lenguajes de programación.
- El reconocedor debe repetir el procesamiento a través de la Máquina Característica por cada avance que hace en la entrada – decisión de implementación.

Agregar *lookahead* generalizado vuelve a ser la solución.

UNIVERSIDAD SIMÓN BOLÍVAR

Contexto e items $LR(k)$ de la gramática

Prefijos viables con *lookahead*

Sea $G = (N, \Sigma, P, S)$ y $A \to \alpha \in P$.
Si se tiene

$$S\#^k \overset{*}{\Rightarrow} \beta A u_i v \Rightarrow \beta \alpha u_i v$$

con $|u_i| = k$, entonces

- El prefijo $\beta\alpha u_i$ es un **contexto LR(k)** para $A \to \alpha$.

UNIVERSIDAD SIMÓN BOLÍVAR

Contexto e items $LR(k)$ de la gramática

Prefijos viables con *lookahead*

Sea $G = (N, \Sigma, P, S)$ y $A \to \alpha \in P$.
Si se tiene

$$S\#^k \overset{*}{\Rightarrow} \beta A u_i v \Rightarrow \beta \alpha u_i v$$

con $|u_i| = k$, entonces

- El prefijo $\beta\alpha u_i$ es un **contexto LR(k)** para $A \to \alpha$.
- Para cualesquiera $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ y $\alpha = \alpha_1\alpha_2$

$$[A \to \alpha_1 \cdot \alpha_2, \{u_i\}]$$

es un **item LR(k)** para la gramática.

UNIVERSIDAD SIMÓN BOLÍVAR

Contexto e items $LR(k)$ de la gramática

Prefijos viables con *lookahead*

Sea $G = (N, \Sigma, P, S)$ y $A \to \alpha \in P$.
Si se tiene

$$S\#^k \overset{*}{\Rightarrow} \beta A u_i v \Rightarrow \beta \alpha u_i v$$

con $|u_i| = k$, entonces

- El prefijo $\beta\alpha u_i$ es un **contexto LR(k)** para $A \to \alpha$.
- Para cualesquiera $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ y $\alpha = \alpha_1\alpha_2$

$$[A \to \alpha_1 \cdot \alpha_2, \{u_i\}]$$

  es un **item LR(k)** para la gramática.
- Todo prefijo de $\beta\alpha$ es un *prefijo viable*.

UNIVERSIDAD SIMÓN BOLÍVAR

Items con *lookahead*

Observando que al expandir con $A \to \alpha$ tendremos

$$S\#^k \overset{*}{\Rightarrow} \beta A x \Rightarrow \beta\alpha x$$

que produce el item

$$[A \to \alpha_1 \cdot \alpha_2, \{u_i\}]$$

- El conjunto $\{u_1, u_2, \ldots, u_n\}$ es el *lookahead set* del item $LR(k)$.
- Es idéntico a $FIRST_k(x)$ – los k terminales que siguen a α.
- El símbolo inicial no es recursivo – *lookahead set* siempre será $\{\#^k\}$.

UNIVERSIDAD SIMÓN BOLÍVAR

Máquina característica $LR(k)$

Sea $G = (N, \Sigma, P, S)$. Construiremos el $\lambda - NFA$

$$M = (Q, N \cup \Sigma, \delta, q_0, Q)$$

Q contiene todos los items $LR(k)$ además del estado q_0, mientras

$$\begin{aligned}
\delta(q_0, \lambda) &= \{[S \to \cdot\alpha, \{\#^k\}] \mid S \to \alpha \in P\} \\
\delta([A \to \alpha \cdot \mathbf{a}\beta, \{u_1, \ldots, u_p\}], \mathbf{a}) &= \{[A \to \alpha\mathbf{a} \cdot \beta, \{u_1, \ldots, u_p\}]\} \\
\delta([A \to \alpha \cdot B\beta, \{u_1, \ldots, u_p\}], B) &= \{[A \to \alpha B \cdot \beta, \{u_1, \ldots, u_p\}]\} \\
\delta([A \to \alpha \cdot B\beta, \{u_1, \ldots, u_p\}], \lambda) &= \{[B \to \cdot\gamma, \{v_1, \ldots v_q\}] \mid B \to \gamma \in P\}
\end{aligned}$$

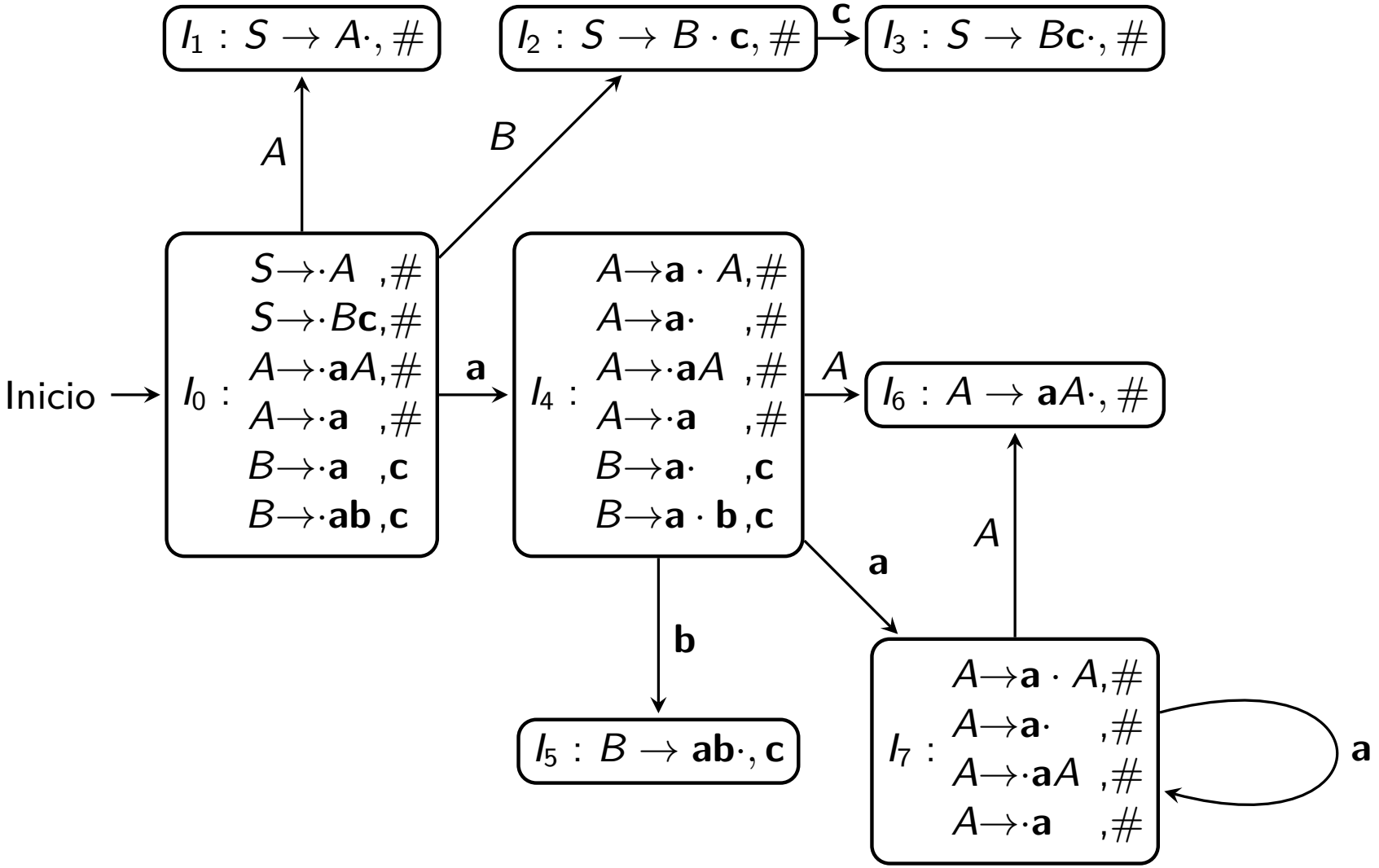
donde

$$v_i \in FIRST_k(\beta u_j) \text{ para algún } j$$

UNIVERSIDAD SIMÓN BOLÍVAR

Un ejemplo simple

Cálculo de la máquina característica determinística $LR(1)$



UNIVERSIDAD SIMÓN BOLÍVAR

¿Cómo identificar conflictos?

- Conflicto *shift/reduce LR(k)* cuando un estado I_i contiene items de la forma

$$[A \to \alpha \cdot, u]$$
$$[B \to \beta \cdot \mathbf{a}\gamma, w]$$

con $u \in FIRST_k(\mathbf{a}\gamma w)$.

UNIVERSIDAD SIMÓN BOLÍVAR

¿Cómo identificar conflictos?

- Conflicto *shift/reduce LR(k)* cuando un estado I_i contiene items de la forma

$$[A \to \alpha\cdot, u]$$
$$[B \to \beta \cdot \mathbf{a}\gamma, w]$$

con $u \in FIRST_k(\mathbf{a}\gamma w)$.

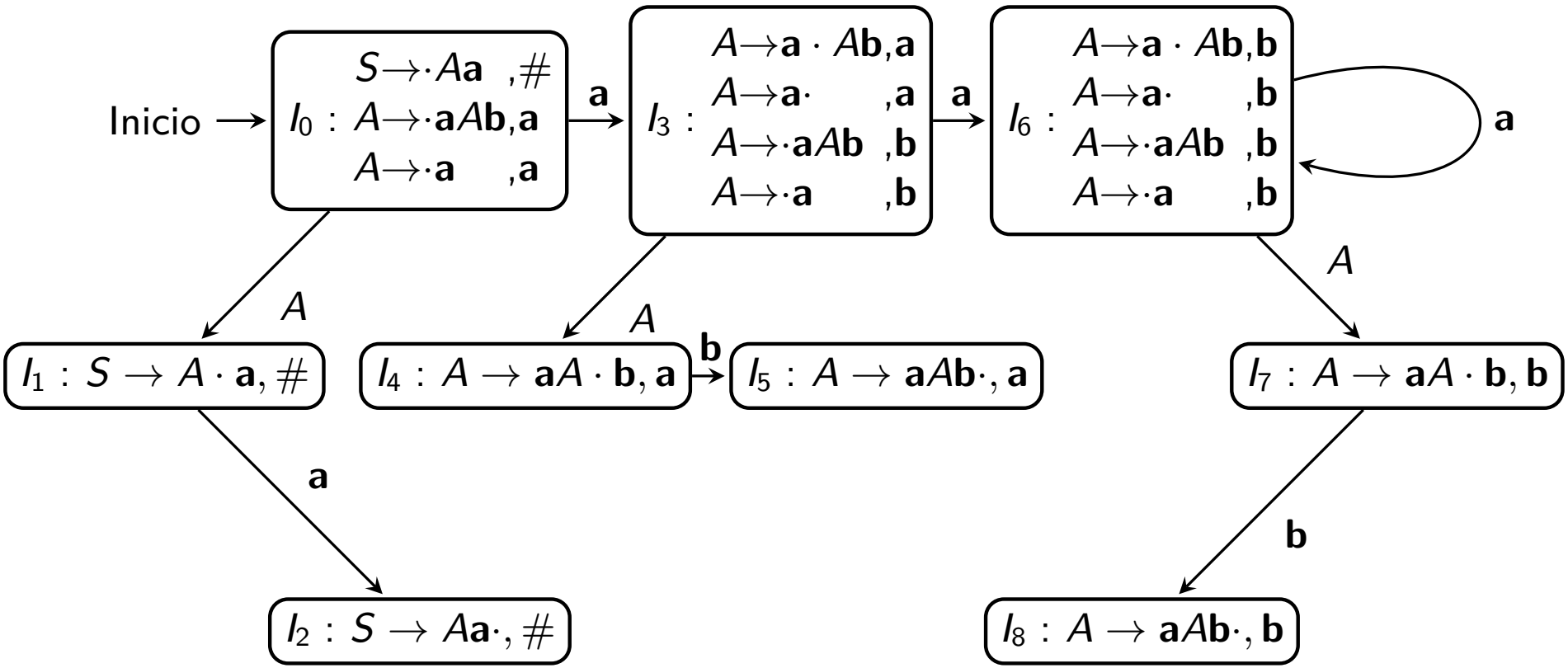
- Conflicto *reduce/reduce LR(k)* cuando un estado I_i contiene items de la forma

$$[A \to \alpha\cdot, u]$$
$$[B \to \beta\cdot, u]$$

con $A \to \alpha \neq B \to \beta$

UNIVERSIDAD SIMÓN BOLÍVAR

Un ejemplo con conflictos

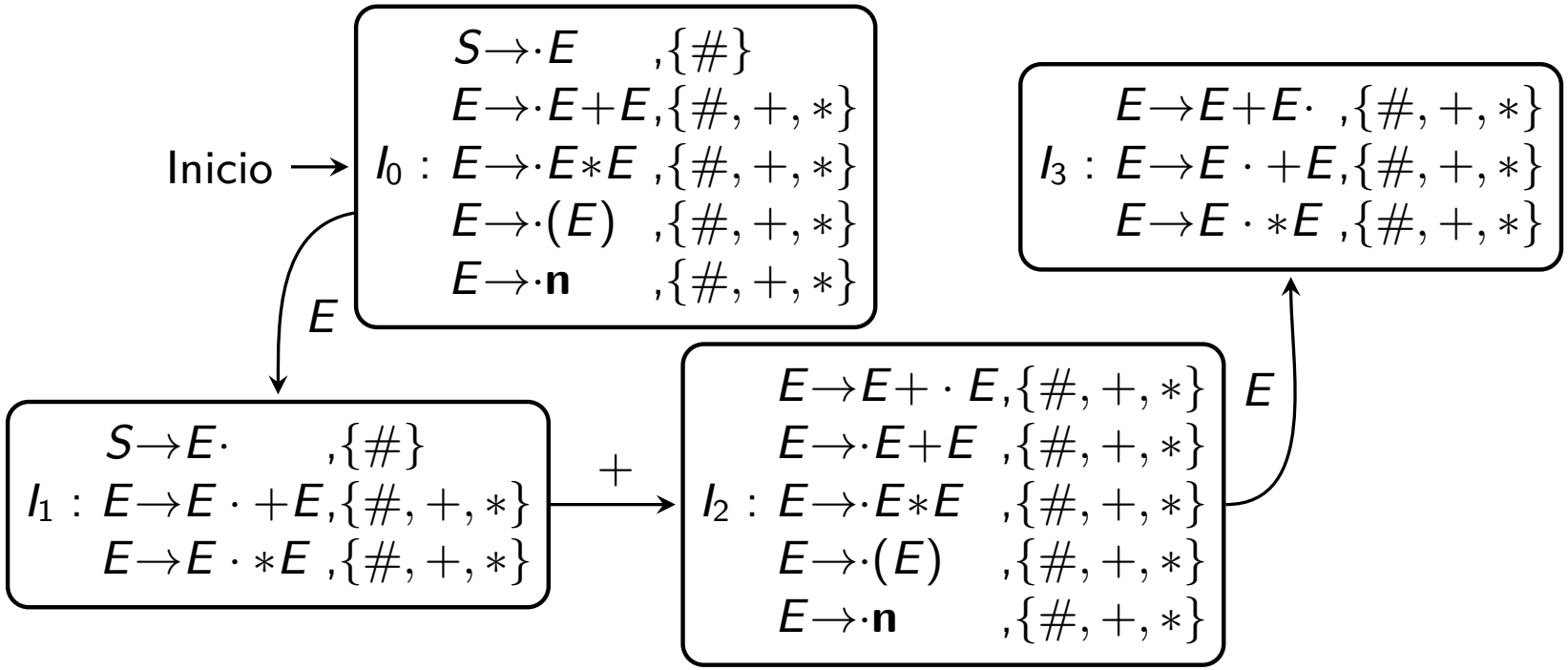


Conflictos *shift*/*reduce* en I_3

UNIVERSIDAD SIMÓN BOLÍVAR

Parte de un ejemplo con conflictos

La gramática "bonita" de expresiones



Conflictos *shift/reduce* y *reduce/reduce* en I_3

UNIVERSIDAD SIMÓN BOLÍVAR

Gramática $LR(k)$

Sea $G = (N, \Sigma, P, S)$ con símbolo inicial no recursivo y con **marcador de final (*endmarker*)** $\#^k$. Diremos que G es **LR(k)** con $k \geq 0$ si siempre que existan dos derivaciones más derechas

$$S \stackrel{*}{\Rightarrow} \alpha A u \Rightarrow \alpha \beta u$$

$$S \stackrel{*}{\Rightarrow} \gamma B x \Rightarrow \alpha \beta y$$

y además

$$FIRST_k(u) = FIRST_k(y)$$

implica $\alpha = \gamma \wedge A = B \wedge x = y$.

¿Por qué el símbolo inicial no puede ser recursivo?

UNIVERSIDAD SIMÓN BOLÍVAR

Gramáticas $LR(1)$

Definición general

Sea G una CFG con símbolo inicial no recursivo. Entonces G es $LR(1)$ si y sólo si $\hat{\delta}$ de su máquina característica determinística $LR(1)$ satisface:

1. Si $\hat{\delta}(q_s, w)$ contiene items

$$[A \to \alpha\cdot, \{u_1, \ldots, u_p\}]$$
$$[B \to \beta \cdot \mathbf{a}\gamma, \{v_1, \ldots, v_q\}]$$

   entonces $\forall i$ se cumple $\mathbf{a} \neq u_i$

2. Si $\hat{\delta}(q_s, w)$ contiene items

$$[A \to \alpha\cdot, \{u_1, \ldots, u_p\}]$$
$$[B \to \beta\cdot, \{v_1, \ldots, v_q\}]$$

   entonces $\forall i \forall j$ se cumple $u_i \neq v_j$

UNIVERSIDAD SIMÓN BOLÍVAR

Reconocedor basado en la máquina $LR(1)$

input: $G = (N, \Sigma, P, S)$ CFG $LR(1)$, $w \in \Sigma^*$ y la Máquina $LR(1)$ de G

Descomponer $w = zv$ con $z \in \Sigma \cup \{\lambda\} \wedge v \in \Sigma^*$
$u, error \leftarrow \lambda,$ **false**
repeat
  if $\hat{\delta}(q_0, u)$ contiene $[A \rightarrow \alpha\cdot, \{z_1, \ldots, z_n\}]$ con $u = p\alpha \wedge z = z_i$ **then**
    $u \leftarrow pA$
  else if $z \neq \lambda \wedge \hat{\delta}(q_0, u)$ contiene $[A \rightarrow p \cdot zq, \{\ldots\}]$ **then**
    $u \leftarrow uz$
    Descomponer $v = zv'$ con $z \in \Sigma \cup \{\lambda\} \wedge v' \in \Sigma^*$
    $v \leftarrow v'$
  else
    $error \leftarrow$ **true**
  end if
until $u = S \vee error$
if $u = S$ **then**
  aceptar
else
  rechazar
end if

UNIVERSIDAD SIMÓN BOLÍVAR

Relevancia de $LR(k)$

- Con $k > 1$, calcular el *lookahead* es tedioso y requiere mucho espacio.

UNIVERSIDAD SIMÓN BOLÍVAR

Relevancia de $LR(k)$

- Con $k > 1$, calcular el *lookahead* es tedioso y requiere mucho espacio.
- Un reconocedor $LR(k+1)$ es un *poco* más poderoso que un reconocedor $LR(k)$ con $k > 0$ – si una gramática cualquiera no es $LR(1)$ lo más probable es que **tampoco** sea $LR(2)$, $LR(3)$, etc. [*Ukkonen*1985]

UNIVERSIDAD SIMÓN BOLÍVAR

Relevancia de $LR(k)$

- Con $k > 1$, calcular el *lookahead* es tedioso y requiere mucho espacio.
- Un reconocedor $LR(k+1)$ es un *poco* más poderoso que un reconocedor $LR(k)$ con $k > 0$ – si una gramática cualquiera no es $LR(1)$ lo más probable es que **tampoco** sea $LR(2)$, $LR(3)$, etc. [*Ukkonen*1985]
- Toda gramática $LR(k+1)$ puede transformarse en una gramática $LR(k)$, con $k > 1$, hasta llegar a $LR(1)$ y **a veces** hasta $LR(0)$, pero la gramática resultante es enorme [*MLS*1976].

UNIVERSIDAD SIMÓN BOLÍVAR

Relevancia de $LR(k)$

- Con $k > 1$, calcular el *lookahead* es tedioso y requiere mucho espacio.
- Un reconocedor $LR(k+1)$ es un *poco* más poderoso que un reconocedor $LR(k)$ con $k > 0$ – si una gramática cualquiera no es $LR(1)$ lo más probable es que **tampoco** sea $LR(2)$, $LR(3)$, etc. [*Ukkonen*1985]
- Toda gramática $LR(k+1)$ puede transformarse en una gramática $LR(k)$, con $k > 1$, hasta llegar a $LR(1)$ y **a veces** hasta $LR(0)$, pero la gramática resultante es enorme [*MLS*1976].
- Cualquier gramática $LL(k)$ puede reconocerse usando $LR(k)$ – todo método de reconocimiento determinístico tiene el mismo, o menos potencia, que el método $LR(k)$ [*Nijholt*1982].

UNIVERSIDAD SIMÓN BOLÍVAR

Consideraciones

- Construya la máquina característica $LR(1)$ para G_E.
- Use el algoritmo reconocedor basado en la máquina característica de G_E para reconocer $b * (b + b)$ e identifique las formas sentenciales más derechas en el proceso.

UNIVERSIDAD SIMÓN BOLÍVAR

Bibliografía

- [*Sudkamp*]
  - Sección 20.5
  - Ejercicios 20.6 a 20.9
- [*MLS*1976] **M.D. Mickunas, R.L. Lancaster y V.B.Schneider**
  Transforming LR(k) grammars to LR(1), SLR(1) and (1,1) bounded right-context grammars
  ACM Journals Vol. 23 No. 3, July 1976
- [*Ukkonen*1985] **E. Ukkonen**
  Upper bounds on the size of LR(k) parsers
  Information Processing Letters Vol. 20 No. 2, February 1985
- [*Nijholt*1982] **A. Nijholt**
  On the relationship between the *LL(k)* and *LR(k)* grammars

UNIVERSIDAD SIMÓN BOLÍVAR