Esquemas de Traducción

CI4721 – Lenguajes de Programación II

Ernesto Hernández-Novich
<emhn@usb.ve>

Universidad "Simón Bolívar"

UNIVERSIDAD SIMÓN BOLÍVAR

L-Atribución y los reconocedores *LR*

- El reconocedor *LR* construye el árbol desde las hojas hacia arriba – al construir las hojas no se tiene acceso a la raíz.
- Acciones al final de la regla para sintetizar atributos se ejecutan al reducir – no son un problema.
- Acciones en medio de la regla son el problema
  - Sólo cuando calculan atributos heredados.
  - Transformarlas en acciones finales que sintetizan.
  - Simular los atributos heredados apoyándose en el estado mutable del meta-lenguaje de acciones.

Reescribir el esquema puede ayudar a implantarlo
aunque dificulte su comprensión.

UNIVERSIDAD SIMÓN BOLÍVAR

Heredando atributos en el reconocedor LR

La técnica "sé lo que estoy haciendo"

Heredar atributos a través de la pila intenta hacer que la información de la raíz esté disponible durante la construcción de las hojas – parece contradictorio con la operación de un reconocedor LR.

- Un reconocedor LR reduce $A \to XY$
  - Eliminando los estados XY del tope de la pila.
  - Reemplazándolos por A
- Si $X.s$ es un atributo sintetizado de X esté estará en la pila
  - **Antes** de procesar el subárbol correspondiente a la reducción de Y.
  - Podría ser *heredado* hacia Y a través de una copia simple.

UNIVERSIDAD SIMÓN BOLÍVAR

Heredando atributos en el reconocedor LR

La técnica "sé lo que estoy haciendo"

Heredar atributos a través de la pila intenta hacer que la información de la raíz esté disponible durante la construcción de las hojas – parece contradictorio con la operación de un reconocedor LR.

- Un reconocedor LR reduce $A \to XY$
  - Eliminando los estados XY del tope de la pila.
  - Reemplazándolos por A
- Si $X.s$ es un atributo sintetizado de X esté estará en la pila
  - **Antes** de procesar el subárbol correspondiente a la reducción de Y.
  - Podría ser *heredado* hacia Y a través de una copia simple.

"Ciertas restricciones aplican"

UNIVERSIDAD SIMÓN BOLÍVAR

Posiciones predecibles

Cuando ocurre, es simple de aprovechar

$$
\begin{array}{lcll}
S & \rightarrow & D\ \# & \\
D & \rightarrow & T\ L & \{L.in \leftarrow T.type\} \\
T & \rightarrow & \textbf{int} & \{T.type \leftarrow "i"\} \\
T & \rightarrow & \textbf{float} & \{T.type \leftarrow "f"\} \\
L & \rightarrow & & \{L_1.in \leftarrow L.in\} \\
 & & L_1\ ,\textbf{id} & \{addtype(\textbf{id}.lexema, L.in\} \\
L & \rightarrow & \textbf{id} & \{addtype(\textbf{id}.lexema, L.in)\}
\end{array}
$$

- Es L-Atribuida – *in* es heredado, *type* es sintetizado.
- Tiene una acción intermedia.

¿Cómo es procesada por el reconocedor *LR*?

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L
			que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L
			que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$ Aquí habría que heredar $T.type$ hacia L que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id** ¡El $L.type$ está una posición abajo! $\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L
			que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo!
			$\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	, L T S #	_ _ i _ #	

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L
			que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo!
			$\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	, L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
,baz #	**id** , L T S #	**bar** _ _ i _ #	

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo!
			$\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
,baz #	**id , L T S #**	**bar** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo!
			$\{addtype('\mathbf{bar}', '\mathbf{i}')\}$
,baz #	L T S #	_ i _ #	

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L
			que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo!
			$\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
,baz #	**id ,** L T S #	**bar** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo!
			$\{addtype('\mathbf{bar}', '\mathbf{i}')\}$
,baz #	L T S #	_ i _ #	shift **,**
baz #	**,** L T S #	_ _ i _ #	

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo!
			$\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
,baz #	**id , L T S #**	**bar** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo!
			$\{addtype('\mathbf{bar}', '\mathbf{i}')\}$
,baz #	L T S #	_ i _ #	shift **,**
baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
#	**id , L T S #**	**baz** _ _ i _ #	

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L
			que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo!
			$\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
,baz #	**id , L T S #**	**bar** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo!
			$\{addtype('\mathbf{bar}', '\mathbf{i}')\}$
,baz #	L T S #	_ i _ #	shift **,**
baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
#	**id , L T S #**	**baz** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo!
			$\{addtype('\mathbf{baz}', '\mathbf{i}')\}$
#	L T S #	_ i _ #	

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L
			que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo!
			$\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
,baz #	**id , L T S #**	**bar** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo!
			$\{addtype('\mathbf{bar}', '\mathbf{i}')\}$
,baz #	L T S #	_ i _ #	shift **,**
baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
#	**id , L T S #**	**baz** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo!
			$\{addtype('\mathbf{baz}', '\mathbf{i}')\}$
#	L T S #	_ i _ #	reduce $D \rightarrow TL$
#	D S #	i _ #	

UNIVERSIDAD SIMÓN BOLÍVAR

Una corrida

Entrada	Pila LR	Pila Atributos	Acción
int foo,bar,baz#	S #	_ #	shift **int**
foo,bar,baz #	**int** S #	_ _ #	reduce $T \rightarrow$ **int** $\{T.type \leftarrow \mathbf{i}\}$
			Aquí habría que heredar $T.type$ hacia L que aún no está en la pila . . .
foo,bar,baz #	T S #	i _ #	shift **id** – Empilar atributo intrínseco.
,bar,baz #	**id** T S #	**foo** i _ #	reduce $L \rightarrow$ **id**
			¡El $L.type$ está una posición abajo! $\{addtype('\mathbf{foo}', '\mathbf{i}')\}$
,bar,baz #	L T S #	_ i _ #	shift **,**
bar,baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
,baz #	**id , L T S #**	**bar** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo! $\{addtype('\mathbf{bar}', '\mathbf{i}')\}$
,baz #	L T S #	_ i _ #	shift **,**
baz #	**,** L T S #	_ _ i _ #	shift **id** – Empilar atributo intrínseco.
#	**id , L T S #**	**baz** _ _ i _ #	reduce $L \rightarrow L, \mathbf{id}$
			¡El $L.type$ está tres posiciones abajo! $\{addtype('\mathbf{baz}', '\mathbf{i}')\}$
#	L T S #	_ i _ #	reduce $D \rightarrow TL$
#	D S #	i _ #	shift #
#	# D S #	_ i _ #	accept

UNIVERSIDAD SIMÓN BOLÍVAR

Posiciones predecibles

Inherited attributes are so mainstream

- T siempre está en una posición conocida de la pila.
  - Por su posición relativa a L en la producción.
  - Gracias a la recursión izquierda generadora de L.
- Pueden calcularse desplazamientos apropiados en cada regla para simular el atributo heredado.

$$
\begin{array}{lcll}
S & \rightarrow & D\ \# & \\
D & \rightarrow & T\ L & \{L.in \leftarrow T.type\} \\
T & \rightarrow & \textbf{int} & \{T.type \leftarrow "i"\} \\
T & \rightarrow & \textbf{float} & \{T.type \leftarrow "f"\} \\
L & \rightarrow & L_1\ , \textbf{id} & \{addtype(\textbf{id}.lexema, stack[top-3]\} \\
L & \rightarrow & \textbf{id} & \{addtype(\textbf{id}.lexema, stack[top-1])\}
\end{array}
$$

UNIVERSIDAD SIMÓN BOLÍVAR

¿Cómo hacerlo en las herramientas?

Epic global variable is epic

```
#define T_INT   1
#define T_FLOAT 2
int l_in;
...
T : int     { l_in = T_INT; }
  | float   { l_in = T_FLOAT; }
  ;
...
L : id      { addtype($1,l_in); }
  | L , id  { addtype($3,l_in); }
  ;
```

Simularlo con variables globales – simple y claro.

UNIVERSIDAD SIMÓN BOLÍVAR

¿Cómo hacerlo en las herramientas?

Looking down the stack like a boss

```
#define T_INT   1
#define T_FLOAT 2
D : T L
  ;
T : int     { $$ = T_INT; }
  | float   { $$ = T_FLOAT; }
  ;
...
L : id      { addtype($1,$0); }
  | L , id  { addtype($3,$0); }
  ;
```

- $0 refiere al primer elemento por debajo de la regla *en proceso*
  - En la primera regla *L*, refiere al *T* debajo de **id**,
  - En la segundo regla *L*, refiere al *T* debajo de *L*,
- Puede usarse $-n con todos los riesgos que involucra.

UNIVERSIDAD SIMÓN BOLÍVAR

No obstante ...

No siempre se puede predecir la posición

$$
\begin{array}{lcll}
S & \rightarrow & \mathbf{a}\ A\ C & \{C.i \leftarrow A.s\} \\
S & \rightarrow & \mathbf{a}\ A\ B\ C & \{C.i \leftarrow A.s\} \\
C & \rightarrow & \mathbf{c} & \{C.s \leftarrow g(C.i)\}
\end{array}
$$

- En el momento de reducir $C \rightarrow c$ el valor de $C.i$
  - Podría estar en $stack[top - 1]$ – si se usó la primera regla.
  - Podría estar en $stack[top - 2]$ – si se usó la segunda regla.
- No queda más remedio que usar una variable global.

UNIVERSIDAD SIMÓN BOLÍVAR

No obstante ...

No siempre se puede predecir la posición

$$
\begin{array}{lcll}
S & \rightarrow & \mathbf{a}\ A\ C & \{C.i \leftarrow A.s\} \\
S & \rightarrow & \mathbf{a}\ A\ B\ C & \{C.i \leftarrow A.s\} \\
C & \rightarrow & \mathbf{c} & \{C.s \leftarrow g(C.i)\}
\end{array}
$$

- En el momento de reducir $C \rightarrow c$ el valor de $C.i$
  - Podría estar en $stack[top - 1]$ – si se usó la primera regla.
  - Podría estar en $stack[top - 2]$ – si se usó la segunda regla.
- No queda más remedio que usar una variable global.

¿Cómo lograr que la posición del atributo sea *invariante*?

UNIVERSIDAD SIMÓN BOLÍVAR

Hay un camino hacia el invariante

- Supongamos un Esquema de Traducción Dirigido por Sintaxis L-Atribuido con acciones en medio y al final de las reglas.
- Por cada acción en medio de una regla
  - Reemplazarla en la regla por el no terminal M_i.
  - Agregar la regla $M_i \to \lambda$
- Sea $A \to \alpha\{a\}\beta$ la producción donde M_i reemplazó la acción a. Asociar con $M_i \to \lambda$ la acción a' tal que
  - Copia aquellos atributos de A o de símbolos en α necesarios para a, como atributos heredados de M_i.
  - Cualquier atributo calculado por a se convierte en sintetizado de M_i.

UNIVERSIDAD SIMÓN BOLÍVAR

Hay un camino hacia el invariante

- Supongamos un Esquema de Traducción Dirigido por Sintaxis L-Atribuido con acciones en medio y al final de las reglas.
- Por cada acción en medio de una regla
  - Reemplazarla en la regla por el no terminal M_i.
  - Agregar la regla $M_i \to \lambda$
- Sea $A \to \alpha\{a\}\beta$ la producción donde M_i reemplazó la acción a. Asociar con $M_i \to \lambda$ la acción a' tal que
  - Copia aquellos atributos de A o de símbolos en α necesarios para a, como atributos heredados de M_i.
  - Cualquier atributo calculado por a se convierte en sintetizado de M_i.

Wait... WAT?

UNIVERSIDAD SIMÓN BOLÍVAR

La transformación

Consideremos una regla como

$$A \to \{B.i \leftarrow f(A.i)\}\ B\ C$$

La transformación consiste en hacer

$$\begin{array}{ll} A \to M\ B\ C & \\ M \to \lambda & \{M.i \leftarrow A.i; M.s \leftarrow f(M.i)\} \end{array}$$

Pero la regla de M no tiene acceso a los atributos de A, ¿o si?

UNIVERSIDAD SIMÓN BOLÍVAR

Transformación de la gramática

Partimos de

$$
\begin{array}{lcll}
S & \rightarrow & \mathbf{a}\ A\ C & \{C.i \leftarrow A.s\} \\
S & \rightarrow & \mathbf{a}\ A\ B\ C & \{C.i \leftarrow A.s\} \\
C & \rightarrow & \mathbf{c} & \{C.s \leftarrow g(C.i)\}
\end{array}
$$

UNIVERSIDAD SIMÓN BOLÍVAR

Transformación de la gramática

Partimos de

$$\begin{array}{lcll} S & \rightarrow & \mathbf{a}\ A\ C & \{C.i \leftarrow A.s\} \\ S & \rightarrow & \mathbf{a}\ A\ B\ C & \{C.i \leftarrow A.s\} \\ C & \rightarrow & \mathbf{c} & \{C.s \leftarrow g(C.i)\} \end{array}$$

y lo transformamos en

$$\begin{array}{lcll} S & \rightarrow & \mathbf{a}\ A\ C & \{C.i \leftarrow A.s\} \\ S & \rightarrow & \mathbf{a}\ A\ B\ M\ C & \{M.i \leftarrow A.s; C.i \leftarrow M.s\} \\ C & \rightarrow & \mathbf{c} & \{C.s \leftarrow g(C.i)\} \\ M & \rightarrow & \lambda & \{M.s \leftarrow M.i\} \end{array}$$

- Al reducir M, se puede **copiar** $C.i$ que *siempre* está en $stack[top-2]$.
- Al reducir $C \rightarrow c$ el valor de $C.i$ *siempre* estará en $stack[top-1]$

¿Y si el *cálculo* del atributo es complejo?

- En el esquema de traducción

$$S \rightarrow \ \mathbf{a}\ A\ \{C.i \leftarrow f(A.s)\}\ C$$

el cálculo de $C.i$ es complejo – no es meramente una copia.

- Lo transformamos en

$$\begin{array}{lcll} S & \rightarrow & \mathbf{a}\ A\ M\ C & \{M.i \leftarrow A.s;\ C.i \leftarrow M.s\} \\ M & \rightarrow & \lambda & \{M.s \leftarrow f(M.i)\} \end{array}$$

- La nueva acción al reducir M puede
  - Tener acceso a $C.i$ que *siempre* está en $stack[top-1]$
  - Calcular $f(M.i)$ para sintetizar $M.s$.
- Al reducir $C \rightarrow c$ el valor de $C.i$ *siempre* estará en $stack[top-1]$

M marca el punto del invariante

UNIVERSIDAD SIMÓN BOLÍVAR

¿Qué pasa con las acciones sin atributos?

El esquema de traducción a postfijo

$$
\begin{array}{lcll}
E & \rightarrow & T\ R & \\
R & \rightarrow & +\ T & \{\textbf{print}('+')\} \\
 & & R & \\
R & \rightarrow & \lambda & \\
T & \rightarrow & \textbf{num} & \{\textbf{print}(\textbf{num}.val)\}
\end{array}
$$

se convierte en

$$
\begin{array}{lcll}
E & \rightarrow & T\ R & \\
R & \rightarrow & +\ T\ M\ R & \\
R & \rightarrow & \lambda & \\
M & \rightarrow & \lambda & \{\textbf{print}('+')\} \\
T & \rightarrow & \textbf{num} & \{\textbf{print}(\textbf{num}.val)\}
\end{array}
$$

Las acciones ocurren en los lugares adecuados.

UNIVERSIDAD SIMÓN BOLÍVAR

L-Atribución y los reconocedores *LL*

- El reconocedor *LL* construye el árbol desde la raíz hacia las hojas – al construir las hojas se tiene acceso a la raíz.
- Heredar valores para los atributos no es problema – se pasan hacia abajo en la pila.
- Acciones al final de la regla para sintetizar atributos se ejecutan al culminar la regla son un problema – ¿cómo se asocian a sus raíces si ya fueron expandidas?
- Acciones en medio de las reglas para calcular atributos heredados también son un problema – deben ejecutarse *antes* de expandir el no terminal al cual heredan.

UNIVERSIDAD SIMÓN BOLÍVAR

Basado en un Reconocedor Recursivo Descendente

El Reconocedor Recursivo Descendente para una gramática tiene un procedimiento `A` por cada no terminal.

- Podemos extenderlo para convertirlo en traductor si hacemos
  - Atributos heredados de A – *argumentos* del procedimiento `A`.
  - Atributos sintetizados de A – valor de retorno del procedimiento `A`.
- Atributos complejos o múltiples son modelados usando apuntadores.
- Las operaciones del traductor se incorporan en el flujo del procedimiento `A` según sea conveniente.

UNIVERSIDAD SIMÓN BOLÍVAR

Método de extensión

- Decidir la producción a expandir – usar el *lookahead*.
- Variable locales por cada atributo de símbolos en la producción – no siempre son necesarias, pero simplifican el trabajo.
- Se procesa cada símbolo X del lado derecho expandido:
  - Si X es un *token*, salvar su atributo intrínseco en una variable local de ser necesario, antes de consumirlo.
  - Si X es un no terminal, escribir la llamada $c \leftarrow X(x_1, x_2, \ldots, x_k)$
    - x_i son variables locales o argumentos con atributos heredados para X.
    - c es la variable local con el atributo sintetizado por X.
- Incorporar las instrucciones para las acciones aprovechando las variables y parámetros.
- Al finalizar la regla, retornar el atributo sintetizado.

UNIVERSIDAD SIMÓN BOLÍVAR

Expresiones, una vez más

$$
\begin{array}{lcll}
S & \to & E\# & \{\textbf{print}(E.val)\} \\
E & \to & T & \{U.i \leftarrow T.val\} \\
 & & U & \{E.val \leftarrow U.s\} \\
U & \to & +T & \{U_1.i \leftarrow U.i + T.val\} \\
 & & U_1 & \{U.s \leftarrow U_1.s\} \\
U & \to & \lambda & \{U.s \leftarrow U.i\} \\
T & \to & F & \{V.i \leftarrow F.val\} \\
 & & V & \{T.val \leftarrow V.s\} \\
V & \to & *F & \{V_1.i \leftarrow V_i.i * F.val\} \\
 & & V_1 & \{V.s \leftarrow V_1.s\} \\
V & \to & \lambda & \{V.s \leftarrow V.i\} \\
F & \to & (E) & \{F.val \leftarrow E.val\} \\
F & \to & \mathbf{n} & \{F.val \leftarrow \mathbf{n}.val\}
\end{array}
$$

Vamos a implantarla según el método.

UNIVERSIDAD SIMÓN BOLÍVAR

Expresiones

Uno de los procedimientos triviales

- Convertimos la regla

$$\begin{array}{lcll} E & \rightarrow & T & \{U.i \leftarrow T.val\} \\ & & U & \{E.val \leftarrow U.s\} \end{array}$$

- Supondremos que el cálculo es entero.

```
int E()
{
  int t_val, u_i, u_s;
  t_val = T();
  u_i = t_val;
  u_s = U(u_i);
  return u_s;
}
```

Por supuesto que lo optimizamos – sólo ilustraba el método.

Expresiones

Un procedimiento más interesante

- Convertimos la regla

$$\begin{array}{lcll} U & \rightarrow & +T & \{U_1.i \leftarrow U.i + T.val\} \\ & & U_1 & \{U.s \leftarrow U_1.s\} \\ U & \rightarrow & \lambda & \{U.s \leftarrow U.i\} \end{array}$$

```
int U(int u_i)
{
  int t_val;
  switch (lookahead) {
    case '+': match('+');
              t_val = T();
              return U(u_i + t_val);
    case ')':
    case '#': return u_i;
  }
}
```

Traducción en un Reconocedor *LL*

- Sea un Esquema de Traducción Dirigido por Sintaxis L-Atribuido
  - Basado en una Gramática *LL*.
  - Con acciones dentro de las producciones – quizás resultado de haber aplicado la eliminación de recursión.
- Objetos adicionales en la pila del reconocedor.
  - Terminales con sus atributos intrínsecos.
  - No terminales con sus atributos heredados.
  - Atributos sintetizados – *Synthesize Records*.
  - Acciones por ejecutar – *Action Records*

La pila almacena registos variantes
u objetos de una clase abstracta con subclases especializadas.

UNIVERSIDAD SIMÓN BOLÍVAR

Operaciones sobre la pila

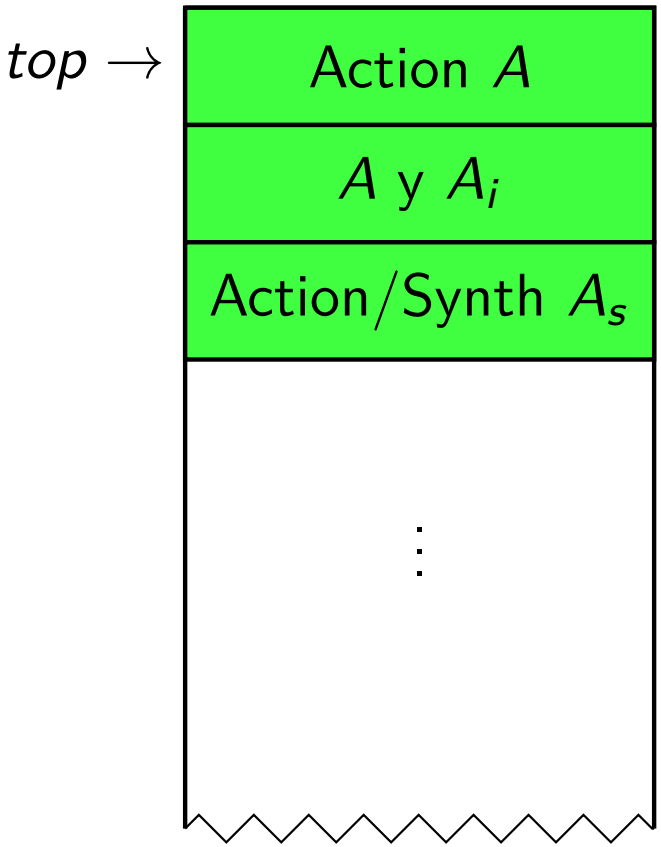
Cuatro tipos de registros en la pila

- Para cada no terminal A
  - El mismo registro contendrá los atributos heredados.
  - La *referencia* al código para evaluar sus atributos heredados se empila en un *Action Record* por **encima** del registro de A.
  - Los atributos sintetizados para A se empilan en un *Synthesize Record* por **debajo** del registro de A.
  - Un *Synthesize Record* podría contener una *referencia* a código necesario para calcular y copiar hacia otros registros.
- Para cada terminal a, el registro contendrá su atributo intrínseco.

UNIVERSIDAD SIMÓN BOLÍVAR

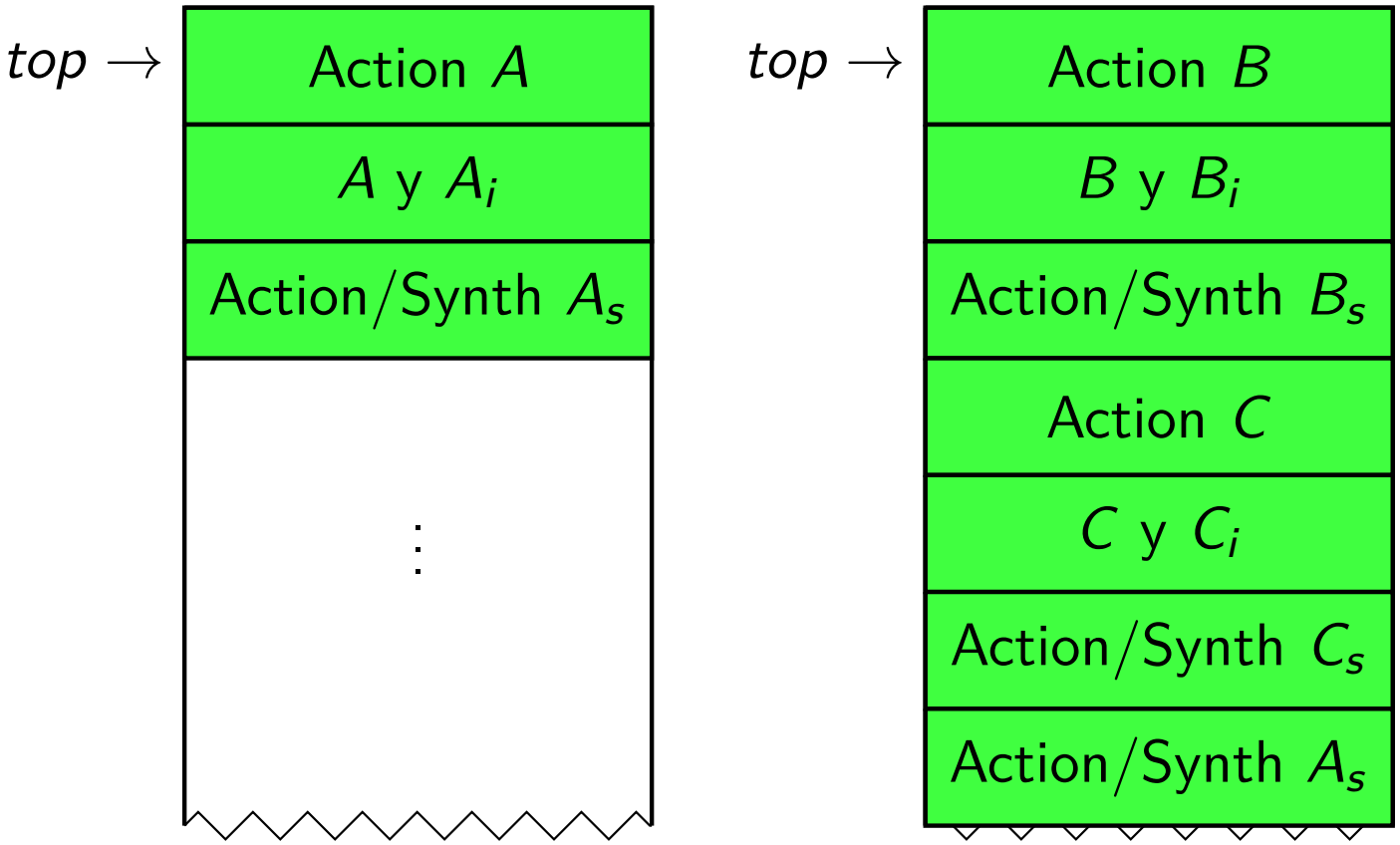
¿Cómo funciona?

El proceso de expansión $A \rightarrow BC$



¿Cómo funciona?

El proceso de expansión $A \to BC$



UNIVERSIDAD SIMÓN BOLÍVAR

Bibliografía

- [*Aho*]
  - Sección 5.5
  - Ejercicios 5.5.1 a 5.5.6
- [*Scott*]
  - Secciones 4.1 a 4.3
  - Ejercicios 4.1 a 4.5

UNIVERSIDAD SIMÓN BOLÍVAR