El análisis descendente LL(1)

Cristina Tîrnăucă

6, 7 y 13 de abril de 2011

Analizadores sintácticos (repaso)

Los analizadores descendentes:

- Corresponden a un autómata de pila determinista.
- Construyen un árbol sintáctico de la raíz hacia las hojas (del símbolo inicial de la gramática hacia los símbolos terminales).
- Por ejemplo: el analizador LL o predictivo lee los datos de izquierda a derecha (*Left to right*) y construye la derivación izquierda (*Leftmost*).
- Emplea una pila para mantener un resumen de lo que espera ver a continuación hasta el final de los datos.
- La recursividad izquierda les puede causar problemas.

Los analizadores ascendentes (shift-reduce):

- También corresponden a un autómata de pila determinista.
- Pero siguen la estrategia inversa: construyen un árbol sintáctico de las hojas hacia la raíz (de los terminales hacia el símbolo inicial de la gramática).
- Por ejemplo: los analizadores LR leen los datos de izquierda a derecha (*Left to right*) y construyen la derivación derecha (*Rightmost*). ("al revés")
- Emplean una pila para mantener un resumen de lo que llevan visto hasta el momento.
- Son más eficientes con recursividad izquierda.

Herramientas conceptuales auxiliares

Tres nociones importantes

Sobre palabras formadas por símbolos de la gramática, tanto terminales como no terminales.

- Anulabilidad de una palabra,
- FIRST de una palabra (terminales por los que puede empezar una parte de la entrada que derive de esa palabra),
- FOLLOW de una palabra (terminales que pueden aparecer en una entrada válida justo a continuación de una parte de la entrada que derive de esa palabra).

Anulabilidad

Es decir, capacidad para "desaparecer"

Una palabra es anulable si puede derivar en la palabra vacía.

- Palabras que contienen algún símbolo terminal:
  Nunca pueden derivar en la palabra vacía, porque un símbolo terminal que participa en una derivación ya no puede desaparecer de ella.
- Palabras que sólo contienen símbolos no terminales:
  - λ es anulable;
  - si α y β son anulables, $\alpha\beta$ son anulables;
  - si la gramática contiene una regla $X \to \alpha$ y α es anulable, X es anulable.

Lo calculamos de manera iterativa, alternativamente para símbolos no terminales y para partes derechas de las reglas: nada es anulable hasta que se demuestra lo contrario.

¿Por qué nos interesa la anulabilidad?

Ejemplo

$S \to SS \mid aSb \mid bSa \mid \lambda$

Entrada: $w = aabb$

$S \Rightarrow SS$
$S \Rightarrow SS \Rightarrow SSS$
$S \Rightarrow SS \Rightarrow aSbS$
$S \Rightarrow SS \Rightarrow bSaS$
~~$S \Rightarrow SS \Rightarrow bSaS$~~
$S \Rightarrow SS \Rightarrow S$

$S \Rightarrow aSb$
$S \Rightarrow aSb \Rightarrow aSSb$
$S \Rightarrow aSb \Rightarrow aaSbb$
$S \Rightarrow aSb \Rightarrow abSab$
~~$S \Rightarrow aSb \Rightarrow abSab$~~
$S \Rightarrow aSb \Rightarrow ab$
~~$S \Rightarrow aSb \Rightarrow ab$~~

$S \Rightarrow bSa$
~~$S \Rightarrow bSa$~~

$S \Rightarrow \lambda$
~~$S \Rightarrow \lambda$~~

FIRST

FIRST(α) $= \{a \mid \alpha \text{ puede derivar en } a\gamma\}$, donde a es un símbolo terminal y α y γ son palabras formadas por símbolos terminales o no terminales.

FIRST(α): conjuntos lo más pequeños posible tales que

- FIRST(a) $= \{a\}$ para cada símbolo terminal a;
- si $\alpha = X\beta$ y X no es anulable, FIRST(α) $=$ FIRST(X);
- si $\alpha = X\beta$ y X es anulable, FIRST(α) $=$ FIRST(X) $\cup$ FIRST(β);
- si la gramática contiene una regla $X \to \alpha$, FIRST(X) incluye FIRST(α).

Cálculo iterativo: inicialmente no sabemos de ningún terminal y empezamos por $\emptyset$; y alternamos entre símbolos no terminales y partes derechas de reglas.

FOLLOW

FOLLOW(X) $= \{a \mid S$ puede derivar en $\alpha X a \gamma\}$, donde X es un símbolo no terminal, a es un símbolo terminal, α y γ son palabras formadas por símbolos terminales o no terminales, y S es el símbolo inicial de la gramática.

FOLLOW(X): conjuntos lo más pequeños posible tales que

- FOLLOW(S) incluye el fin de datos (representado aquí \$);
- si la gramática contiene una regla $X \to Y_1 \dots Y_k$ y $Y_{i+1} \dots Y_{j-1}$ es anulable, con $1 \leq i < j \leq k$, FOLLOW(Y_i) incluye FIRST(Y_j) (prestemos atención al caso $i + 1 = j$).
- si la gramática contiene una regla $X \to Y_1 \dots Y_k$ y $Y_{i+1} \dots Y_k$ es anulable, con $1 \leq i \leq k$, FOLLOW(Y_i) incluye FOLLOW(X) (prestemos atención al caso $i = k$).

Un ejemplo (Anulabilidad, FIRST y FOLLOW)

Consideremos $G = (\{E, E', T, T', F\}, \{+, *, (,), id\}, E, P)$, donde P consiste en las reglas siguientes:

$E \to TE'$
$E' \to +TE' \mid \lambda$

$T \to FT'$
$T' \to *FT' \mid \lambda$

$F \to (E) \mid id$

	Anulable	FIRST	FOLLOW
E	no	$(, id$	$\$,)$
E'	sí	$+$	$\$,)$
T	no	$(, id$	$\$,), +$
T'	sí	$*$	$\$,), +$
F	no	$(, id$	$\$,), +, *$

Definition

Una gramática libre de contexto $G = (V, \Sigma, S, P)$ es $LL(1)$ si

- $S \Rightarrow^*_{lm} wX\gamma \Rightarrow w\alpha\gamma \Rightarrow^*_{lm} wu$,
- $S \Rightarrow^*_{lm} wX\delta \Rightarrow w\beta\delta \Rightarrow^*_{lm} wv$,
- $FIRST(u) = FIRST(v)$

implican $\alpha = \beta$, donde $u, v, w \in \Sigma^*$ y $X \in V$.

Se dice sobre un lenguaje que es LL(1) si se puede generar con una gramática LL(1).

Ejemplos

Gramáticas LL(1)

- $G = (\{S\}, \{(,)\}, S, P = \{S \rightarrow (S)|\lambda\})$
  En esta gramática es obvio que la primera producción se usa cuando aparece un paréntesis abierto y la segunda cuando aparece el primer paréntesis cerrado.
- $G = (\{S\}, \{a, b\}, S, P = \{S \rightarrow aAb|b, A \rightarrow aSAa|b\})$

Una gramática LL(2) que no es LL(1)

$G = (\{S\}, \{a, b\}, S, P = \{S \rightarrow abSba|aa\})$

Ejemplo de lenguaje que no es LL(k) para ningún k

$L = \{a^n cb^n \mid n \geq 1\} \cup \{a^n db^{2n} \mid n \geq 1\}$

El analizador LL descendente

El analizador LL está utilizando:

- un "buffer" para la entrada
- una pila, con símbolos terminales y no terminales
- una tabla de análisis

En cada paso, el analizador lee un símbolo del buffer y el símbolo que está en la cima de la pila.

- si coinciden, el analizador los elimina del buffer y de la pila
- si en la cima de la pila hay un símbolo terminal distinto, devuelve ERROR (la palabra no está aceptada)
- si en la cima de la pila hay un símbolo no terminal X, el analizador "mira" la tabla para ver que regla se debe aplicar, y substituye X por la parte derecha de esa regla.

Al principio la pila contiene el símbolo inicial S y el símbolo especial \$ (el fondo de la pila).

Ejemplo

$G = (\{S, F\}, \{a, (,)\}, S, P)$ con P dado por las reglas siguientes:

1. $S \rightarrow F$
2. $S \rightarrow (S + F)$
3. $F \rightarrow a$

Cuadro: FIRST y FOLLOW

	Anulable	FIRST	FOLLOW
S	no	$a, ($	$+$
F	no	a	$+,)$

Cuadro: Tabla de análisis

	(	)	a	$+$	\$
S	2	-	1	-	-
F	-	-	3	-	-

Entrada: $w = (a + a)$

Tabla de análisis

Para cada símbolo no terminal X y cada símbolo terminal a, añadimos la regla $X \to \alpha$ si:

- $a \in \text{FIRST}(\alpha)$, o
- α es anulable y $a \in \text{FOLLOW}(X)$

Si la tabla contiene a lo sumo una regla para cada una de sus celdas, entonces el analizador siempre sabe que regla se debe utilizar en cada momento.

Conflictos

- FIRST/FIRST conflict (dos alternativas empiezan igual)

$$A \to X \mid XYZ$$

$$A \to XB$$
$$B \to YZ \mid \lambda$$

- FIRST/FOLLOW conflict (el FIRST y el FOLLOW de un símbolo no terminal anulable tienen algo en común)

$$S \to Aab$$
$$A \to a \mid \lambda$$

$$S \to aab \mid ab$$

- Recursividad izquierda

$$E \to E + T \mid T$$

$$E \to TZ$$
$$Z \to +TZ \mid \lambda$$

Conflictos (II)