Gramática extendida

Las gramáticas extendidas y los métodos **ELL** permiten escribir usando ERs y generar parsers descendentes iterativos-recursivos.

Definición

Una *gramática extendida* es una tupla $< V_N, V_T, p, S >$ con
$p : V_N \rightarrow \mathrm{ER}(V)$
p puede ser una función porque se pueden combinar las producciones para un mismo no terminal con la unión.

Operadores: *, +, ?

* se puede implementar con `while`, el + con `do while` y el ? con un `if`.

Generación de código (incompleto)

Definimos por inducción la función $\mathrm{Cod}(E)$ que recibe una expresión regular sobre V y devuelve el código correspondiente:

E	$\mathrm{Cod}(E)$
λ	`skip;`
a	`match(a);`
$R?$	`if(tc in xxx){` $\mathrm{Cod}(R)$ `}`
$R*$	`while(tc in xxx){` $\mathrm{Cod}(R)$ `}`
$R+$	`do{` $\mathrm{Cod}(R)$ `}while(tc in xxx)`
$R_1 R_2$	$\mathrm{Cod}(R_1)$; $\mathrm{Cod}(R_2)$

$R_1 \mid R_2 \mid \ldots \mid R_n$	`if(tc in xxx)` $\mathrm{Cod}(R_1)$
	`eif(tc in xxx)` $\mathrm{Cod}(R_2)$
	`...`
	`else error`

¿Qué representa xxx en cada caso?

Transformación de gramáticas

Pasaje de gramática extendida a gramática común

E	$\mathrm{Cod}(E)$	E *no extendida*
$R?$	`if(tc in xxx){` $\mathrm{Cod}(R)$ `}`	$A \to R \mid \lambda$
$R*$	`while(tc in xxx){` $\mathrm{Cod}(R)$ `}`	$A \to RA \mid \lambda$
$R+$	`do{` $\mathrm{Cod}(R)$ `}while(tc in xxx)`	$A \to RR*$

$R_1 \mid R_2 \mid \ldots \mid R_n$	`if(tc in xxx)` $\mathrm{Cod}(R_1)$
	`eif(tc in xxx)` $\mathrm{Cod}(R_2)$
	`...`
	`else error`
		$A \to R_1 \mid R_2 \mid \ldots$

Transformación de gramáticas

Cálculo de condiciones

E	$\mathrm{Cod}(E)$	E *no extendida*
$R?$	`if(tc in` $\mathrm{Prim}(R)$`){` $\mathrm{Cod}(R)$ `}`	$A \to R \mid \lambda$
$R*$	`while(tc in` $\mathrm{Prim}(R)$`){` $\mathrm{Cod}(R)$ `}`	$A \to RA \mid \lambda$
$R+$	`do{` $\mathrm{Cod}(R)$ `}while(tc in` $\mathrm{Prim}(R)$`)`	$A \to RR*$

$R_1 \mid R_2 \mid \ldots \mid R_n$	`if(tc in` $\mathrm{SD}(A \to R_1)$`)` $\mathrm{Cod}(R_1)$
	`eif(tc in` $\mathrm{SD}(A \to R_2)$`)` $\mathrm{Cod}(R_2)$
	`...`
	`else error`
		$A \to R_1 \mid R_2 \mid \ldots$

Notar que al agregar producciones de esta manera evitamos generar conflictos LL(1).

Por ejemplo, si para producir β^* usáramos $B \rightarrow B\beta|\lambda$, estaríamos agregando una recursión a izquierda.

¿Cómo se usa?
El procedimiento consiste en reemplazar cada subexpresión (empezando por las más externas) por un no terminal nuevo y agregando las producciones que hagan falta.

Ejemplo 1

Tenemos la producción:

$A \to a(bA?c)^*$

$A \to aA_1$
$A_1 \to bA?cA_1 | \lambda$

$A \to aA_1$
$A_1 \to bA_2cA_1 | \lambda$
$A_2 \to A | \lambda$

¿Es **ELL(1)?**
Si tenemos una Gramática Extendida EG y su GLC derivada es **LL(1)**, decimos que EG es **ELL(1)** y el parser generado reconocerá exactamente las cadenas de $L(EG)$.

Ejemplo 2

Gramática común

$E \rightarrow E + T \mid T$
$T \rightarrow T * F \mid F$
$F \rightarrow id \mid (E)$

Gramática extendida

$E \rightarrow T(+T)*$
$T \rightarrow F(*F)*$
$F \rightarrow id \mid (E)$