Compiladores 24-2
Análisis Sintáctico: parser bottom-up LR(0)

Lourdes del Carmen González Huesca
luglzhuesca@ciencias.unam.mx

Facultad de Ciencias, UNAM

13 marzo 2024

Análisis sintáctico

Parser bottom-up

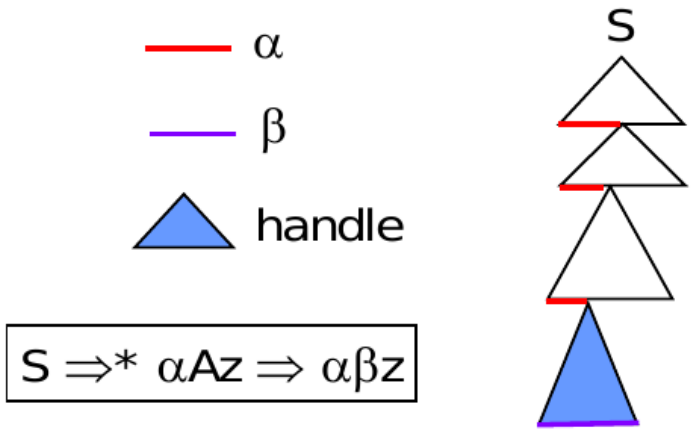
- Analizadores que construyen un árbol de sintaxis concreta (parse tree) desde las hojas y hacia la raíz.
- La técnica para generar los árboles se llama *shift-reduce* donde la reducción se realiza para obtener un proceso inverso de una derivación paso a paso.
- La clase de gramáticas de estos analizadores son las **LR** (lectura left-to-right del input y derivaciones más a la derecha).
- En cada paso del análisis decidir si se debe hacer un shift o un reduce al usar una tabla de acciones que determine lo que debe hacer el parser.
- El algoritmo se sirve de la cadena de entrada y una tabla de parsing que contiene las acciones (shift, reduce o accept) y las transiciones entre estados.

Análisis sintáctico **LR(0)**

- Decidir una acción (shift o reduce) sin considerar un símbolo por adelantado.
- La tabla de parsing definirá los estados de la máquina y las transiciones.
- Para este método, los estados serán conjuntos de producciones de la gramática donde se ha identificado una subcadena.

Análisis sintáctico **LR(0)**

- Decidir una acción (shift o reduce) sin considerar un símbolo por adelantado.
- La tabla de parsing definirá los estados de la máquina y las transiciones.
- Para este método, los estados serán conjuntos de producciones de la gramática donde se ha identificado una subcadena.
- Para identificar una subcadena se usan los mangos o asas (handle) de las cadenas.



Análisis sintáctico **LR(0)**

items

La tabla de parsing indica si se realiza un shift o un reduce, por lo tanto se usará un autómata finito para construirla.

Análisis sintáctico **LR(0)**

items

La tabla de parsing indica si se realiza un shift o un reduce, por lo tanto se usará un autómata finito para construirla.
Los estados y las transiciones de la máquina de decisión se calculan con las funciones de cerradura (estados) y goto (transiciones) las cuales requieren identificar las subcadenas que coinciden con la parte derecha de una producción.

Análisis sintáctico **LR(0)**

items

La tabla de parsing indica si se realiza un shift o un reduce, por lo tanto se usará un autómata finito para construirla.
Los estados y las transiciones de la máquina de decisión se calculan con las funciones de cerradura (estados) y goto (transiciones) las cuales requieren identificar las subcadenas que coinciden con la parte derecha de una producción.

Un **item** es una producción de la gramática con un punto (•) en alguna posición del cuerpo o parte derecha. Este punto marca la coincidencia en el proceso de análisis para identificar los símbolos analizados.

Análisis sintáctico **LR(0)**

items

La tabla de parsing indica si se realiza un shift o un reduce, por lo tanto se usará un autómata finito para construirla.
Los estados y las transiciones de la máquina de decisión se calculan con las funciones de cerradura (estados) y goto (transiciones) las cuales requieren identificar las subcadenas que coinciden con la parte derecha de una producción.

Un **item** es una producción de la gramática con un punto (•) en alguna posición del cuerpo o parte derecha. Este punto marca la coincidencia en el proceso de análisis para identificar los símbolos analizados.
Se clasifican en dos:

- item tipo kernel: es el item de la producción inicial $S \rightarrow \bullet E$ (donde E es el símbolo inicial de la gramática original) o cualquier item que **no** tenga un punto más a la izquierda
- item tipo no-kernel: es cualquier item con el punto más a la izquierda excepto el item inicial $S \rightarrow \bullet E$

Análisis sintáctico **LR(0)**

- ✓ El parser es un autómata de pila que construye el parse-tree desde las hojas hacia la raíz y con un proceso inverso, es decir de atrás hacia adelante en la generación del árbol.
- ✓ La lectura de la cadena de entrada o input se realiza de izquierda a derecha y se va almacenando la información en la pila del autómata.
- ✓ La tabla de parsing tiene dos partes:
  1. las **acciones** a realizar por el parser, shift o reduce
  2. las transiciones **goto** para decidir la regla de la gramática que se usará
- ✓ La tabla de parsing se construye con ayuda de un autómata finito para decidir las acciones el autómata de pila (parser).
  Los estados de esta máquina finita serán los conjuntos canónicos de items para abstraer las decisiones de shift-reduce.
  Los conjuntos de items **canónicos** se calculan con la función auxiliar CLOSURE.

Análisis sintáctico **LR(0)**

cerradura de items

Sea $\mathcal{I}$ un conjunto de items, la cerradura de $\mathcal{I}$ o CLOSURE($\mathcal{I}$) está definida por:

1. los items que son elementos de $\mathcal{I}$
2. Si $A \to \alpha \bullet B\beta$ está en CLOSURE($\mathcal{I}$) (se ha reconocido α)
   y $B \to \gamma$ es una producción de la gramática (falta reconocer a partir de B),
   entonces agregar $B \to \bullet\gamma$ a CLOSURE($\mathcal{I}$)

Análisis sintáctico **LR(0)**

cerradura de items

Sea $\mathcal{I}$ un conjunto de items, la cerradura de $\mathcal{I}$ o CLOSURE($\mathcal{I}$) está definida por:

1. los items que son elementos de $\mathcal{I}$
2. Si $A \to \alpha \bullet B\beta$ está en CLOSURE($\mathcal{I}$) (se ha reconocido α)
   y $B \to \gamma$ es una producción de la gramática (falta reconocer a partir de B),
   entonces agregar $B \to \bullet\gamma$ a CLOSURE($\mathcal{I}$)

El algoritmo es el siguiente:

```
let J = I;
repeat{
   for each item A → α • Bβ in J
       for each grammar production B → γ
          if (B → •γ not in J)
          then add B → •γ
}
until no more items are added to J
```

Análisis sintáctico **LR(0)**

Goto en LR(0)

Dado un conjunto de items $\mathcal{I}$ y un símbolo cualquiera de la gramática X, la función GOTO del estado $\mathcal{I}$ mediante X es la cerradura de todos los items $A \to \alpha X \bullet \beta$ tal que $A \to \alpha \bullet X \beta$ está en $\mathcal{I}$.

Análisis sintáctico **LR(0)**

Goto en LR(0)

Dado un conjunto de items $\mathcal{I}$ y un símbolo cualquiera de la gramática X, la función GOTO del estado $\mathcal{I}$ mediante X es la cerradura de todos los items $A \to \alpha X \bullet \beta$ tal que $A \to \alpha \bullet X\beta$ está en $\mathcal{I}$.

El algoritmo que construye los conjuntos canónicos de items C para una gramática extendida con el símbolo S es el siguiente:

```
C = CLOSURE({S → • E});
repeat{
   for each set of items I in C
       for each grammar symbol X
          if (GOTO(I,X) is not empty and not in C)
          then add GOTO(I,X) to C
}
until no new sets are added to C
```

Análisis sintáctico **LR(0)**

ejemplo

$$
\begin{array}{lrcl}
(0) & E' & \rightarrow & E \\
(1) & E & \rightarrow & E + T \\
(2) & E & \rightarrow & T \\
(3) & T & \rightarrow & T * F \\
(4) & T & \rightarrow & F \\
(5) & F & \rightarrow & (E) \\
(6) & F & \rightarrow & n
\end{array}
$$

- Se iniciará el cálculo de los estados y sus transiciones con el item $E' \rightarrow \bullet E$. El estado I_0 será la cerradura de este item.

Análisis sintáctico **LR(0)**

ejemplo

$$
\begin{array}{lllll}
(0) & E' & \rightarrow & E \\
(1) & E & \rightarrow & E + T \\
(2) & E & \rightarrow & T \\
(3) & T & \rightarrow & T * F \\
(4) & T & \rightarrow & F \\
(5) & F & \rightarrow & (E) \\
(6) & F & \rightarrow & n
\end{array}
$$

- Se iniciará el cálculo de los estados y sus transiciones con el item $E' \rightarrow \bullet E$. El estado I_0 será la cerradura de este item.

  $\textsc{Closure}(\{E' \rightarrow \bullet E\}) = \{E' \rightarrow \bullet E,\ E \rightarrow \bullet E+T,\ E \rightarrow \bullet T,\ T \rightarrow \bullet T*F,\ T \rightarrow \bullet F,\ F \rightarrow \bullet(E),\ F \rightarrow \bullet n\}$

- A continuación se calcularán las funciones GoTo de este estado para generar los siguientes estados del autómata finito.

Análisis sintáctico **LR(0)**

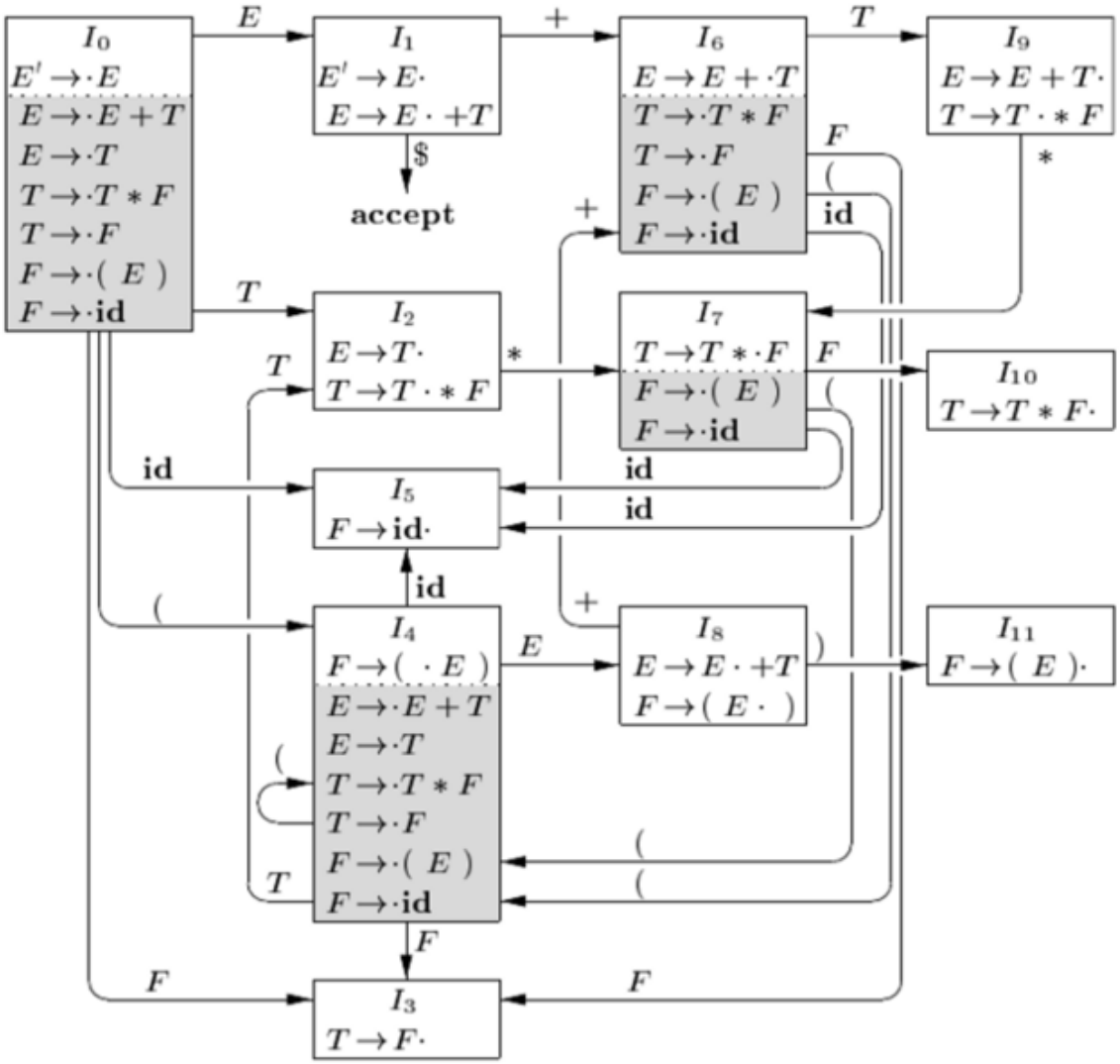
ejemplo

$$
\begin{array}{lrcl}
(0) & E' & \rightarrow & E \\
(1) & E & \rightarrow & E + T \\
(2) & E & \rightarrow & T \\
(3) & T & \rightarrow & T * F \\
(4) & T & \rightarrow & F \\
(5) & F & \rightarrow & (E) \\
(6) & F & \rightarrow & n
\end{array}
$$

- El cálculo de los estados y sus transiciones se realiza iniciando con $E' \rightarrow \bullet E$ y el estado I_0 que es su cerradura.
- El proceso anterior se repite para cada estado nuevo generado hasta completar el autómata.
- La transición a la aceptación es la que se obtiene al procesar el símbolo de fin de cadena y donde el item $E' \rightarrow E\bullet$ pertenece al estado en cuestión.

Análisis sintáctico **LR(0)**

ejemplo, AFD para tabla de parsing



Análisis sintáctico **LR(0)**

ejemplo, tabla de parsing

(1) $E \to E + T$
(2) $E \to T$
(3) $T \to T * F$
(4) $T \to F$
(5) $F \to (E)$
(6) $F \to \mathbf{id}$

STATE	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Referencias

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman.
Compilers, Principles, Techniques and Tools.
Pearson Education Inc., Second edition, 2007.

[2] Jean-Christophe Filliâtre.
Curso Compilation (inf564) école Polytechnique, Palaiseau, Francia.
http://www.enseignement.polytechnique.fr/informatique/INF564/, 2018.
Material en francés.

[3] Frank Pfenning.
Notas del curso (15-411) Compiler Design.
https://www.cs.cmu.edu/~fp/courses/15411-f14/, 2014.

[4] François Pottier.
Presentaciones del curso Compilation (inf564) École Polytechnique, Palaiseau, Francia.
http://gallium.inria.fr/~fpottier/X/INF564/, 2016.
Material en francés.

[5] Michael Lee Scott.
Programming Language Pragmatics.
Morgan-Kauffman Publishers, Third edition, 2009.

[6] Yunlin Su and Song Y. Yan.
Principles of Compilers, A New Approach to Compilers Including the Algebraic Method.
Springer-Verlag, Berlin Heidelberg, 2011.

[7] Steve Zdancewic.
Notas del curso (CIS 341) - Compilers, Universidad de Pennsylvania, Estados Unidos.
https://www.cis.upenn.edu/~cis341/current/, 2018.