Compiladores 24-2
Análisis Sintáctico: parser bottom-up LR(1)

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

Facultad de Ciencias, UNAM

20 marzo 2024

Análisis sintáctico

Parser bottom-up

- La clase de gramáticas **LR** corresponden a analizadores que construyen un árbol de sintaxis concreta desde las hojas y hacia la raíz.
- Los analizadores utilizan una tabla de parsing que decide las acciones y está generada por una autómata finito entre conjuntos de items.
- 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.

  Estudiaremos las variantes de construcción de la tabla:

  - LR(0)
  - SLR *simple LR*
  - LR(1) *canonical-LR*
  - LALR *lookahead-LR*

En general, al incluir símbolos por adelantado (look-aheads) mejora la toma de decisiones:

- en LR(1) los items serán parejas de item L(0) y el look-ahead:

$$[A \to \alpha \bullet \beta, \quad b]$$

ya se ha reconocido a α y se espera encontrar a βb

- extender las funciones de CLOSURE() y GOTO para este tipo de items.

Análisis sintáctico LR(1)

funciones closure y goto

closure Sea I un conjunto de items, para cada elemento del conjunto $[A \to \alpha \bullet B\beta, \quad b]$ y por cada producción $B \to \gamma$ agregar a I

$[B \to \bullet\gamma, \quad \textsc{First}(\beta b)]$
$[B \to \bullet\gamma, \quad \varepsilon]$ si $\textsc{First}(\beta b) = \varnothing$

Análisis sintáctico LR(1)

funciones closure y goto

closure Sea I un conjunto de items, para cada elemento del conjunto $[A \to \alpha \bullet B\beta,\ \ b]$ y por cada producción $B \to \gamma$ agregar a I

$[B \to \bullet\gamma,\ \ \textsc{First}(\beta b)]$
$[B \to \bullet\gamma,\ \ \varepsilon]$ si $\textsc{First}(\beta b) = \varnothing$

```
repeat{
   for each item [A -> α•Bβ,b] in I
       for each grammar production B -> γ
          for each terminal symbol x in FIRST(βb)
                    add [B -> •γ,x] to I
}
until no more items are added to I
```

Análisis sintáctico LR(1)

funciones closure y goto

closure Sea I un conjunto de items, para cada elemento del conjunto $[A \to \alpha \bullet B\beta,\ \ b]$ y por cada producción $B \to \gamma$ agregar a I

$[B \to \bullet\gamma,\ \ \textsc{First}(\beta b)]$
$[B \to \bullet\gamma,\ \ \varepsilon]$ si $\textsc{First}(\beta b) = \varnothing$

```
repeat{
   for each item [A -> α•Bβ,b] in I
       for each grammar production B -> γ
          for each terminal symbol x in FIRST(βb)
                    add [B -> •γ,x] to I
}
until no more items are added to I
```

goto La misma función que en LR(0) pero sobre pares.

Análisis sintáctico LR(1)

algoritmo para calcular items

Para generar todo el autómata finito:

```
initialize C to { 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(1)

algoritmo para la tabla de parsing

Input: Una gramática aumentada.

Output: La tabla de parsing LR(0) con las funciones ACTION and GOTO para la gramática aumentada.

S' es el nuevo símbolo inicial y S es el símbolo inicial de la gramática original.

1. Construir la colección de conjuntos de items según el método LR(1) $C = \{I_0, I_1, \ldots, I_n\}$.
2. El estado i se construye desde I_i y las acciones están determinadas como sigue:
   - Si el item $[A \to \alpha \bullet a\beta, b]$ está en I_i y $\text{GOTO}(I_i, a) = I_j$ entonces $\text{ACTION}(i, a) =$ shift j donde a es un símbolo terminal.
   - Si el item $[A \to \alpha\bullet, a]$ está en I_i donde $A \neq S'$, entonces $\text{ACTION}(i, a) =$ reduce$A \to \alpha$.
   - Si el item $S' \to S\bullet$ está en I_i entonces $\text{ACTION}(i, \#) =$ accept.
3. Las transiciones GOTO para un estado i están construidas para todos los símbolos no-terminales de la gramática aumentada.
4. Todas las entradas no definidas por los pasos 2. y 3. son un error.
5. El estado inicial del parser está construido por el conjunto de items que contiene a $[S' \to \bullet S, \#]$.

Análisis sintáctico LR(1)

ejemplo

Autómata finito (cerradura de items y goto)

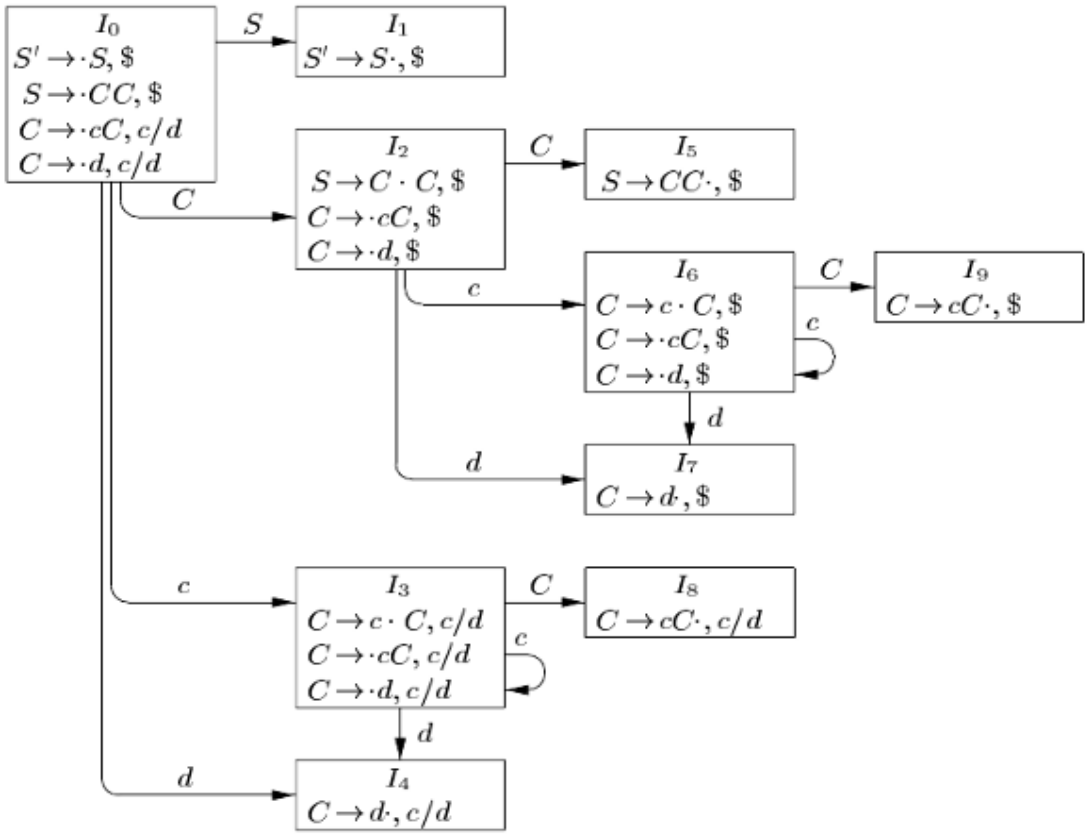
$$S' \to S \qquad S \to CC \qquad C \to cC \mid d$$

Análisis sintáctico LR(1)

ejemplo

Autómata finito (cerradura de items y goto)

$$S' \to S \qquad S \to CC \qquad C \to cC \mid d$$



Análisis sintáctico LR(1)

ejemplo

Tabla de parsing para la gramática

$$S' \to S \qquad S \to CC \qquad C \to cC \mid d$$

Acciones

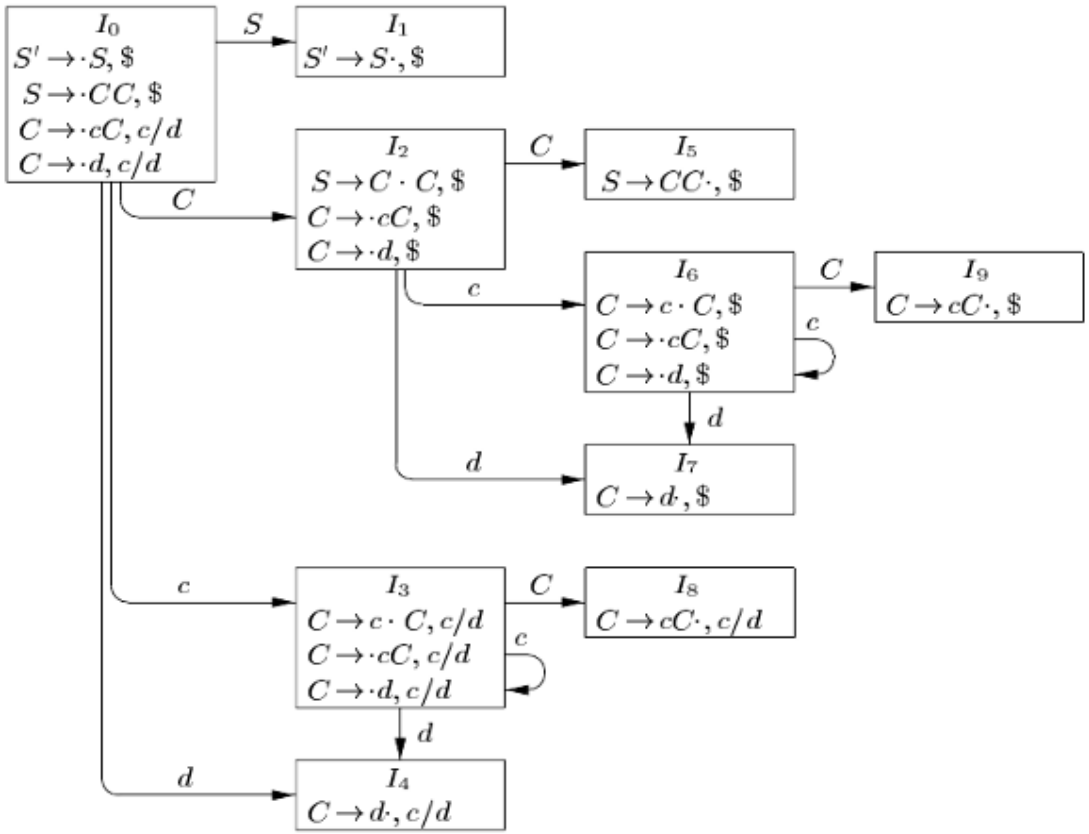
- Si el item $[A \to \alpha \bullet a\beta, b]$ está en I_i y $\textsc{GoTo}(I_i, a) = I_j$ entonces $\textsc{Action}(i, a) =$ shift j donde a es un símbolo terminal.
- Si el item $[A \to \alpha\bullet, a]$ está en I_i donde $A \neq S'$, entonces $\textsc{Action}(i, a) =$ reduce$A \to \alpha$.
- Si el item $S' \to S\bullet$ está en I_i entonces $\textsc{Action}(i, \#) =$ accept.

Análisis sintáctico LR(1)

ejemplo

Autómata finito (cerradura de items y goto)

$$S' \to S \qquad S \to CC \qquad C \to cC \mid d$$



STATE	ACTION			GOTO	
	c	d	\$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

Referencias I

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

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

[5] Tim Teitelbaum.
Introduction to compilers.
http://www.cs.cornell.edu/courses/cs412/2008sp/, 2008.

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