Compiladores 24-2
Análisis Sintáctico: parser bottom-up SLR

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** (lectura left-to-right del input y derivaciones más a la derecha) corresponden a 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.
- Los analizadores utilizan una tabla de parsing que decide las acciones y está generada por una autómata finito entre conjuntos de items.

  El comportamiento del algoritmo es el mismo, manejar el autómata de pila, lo que estudiaremos son variantes en la construcción de la tabla y del autómata que guarda el avance en el análisis de la cadena de entrada:

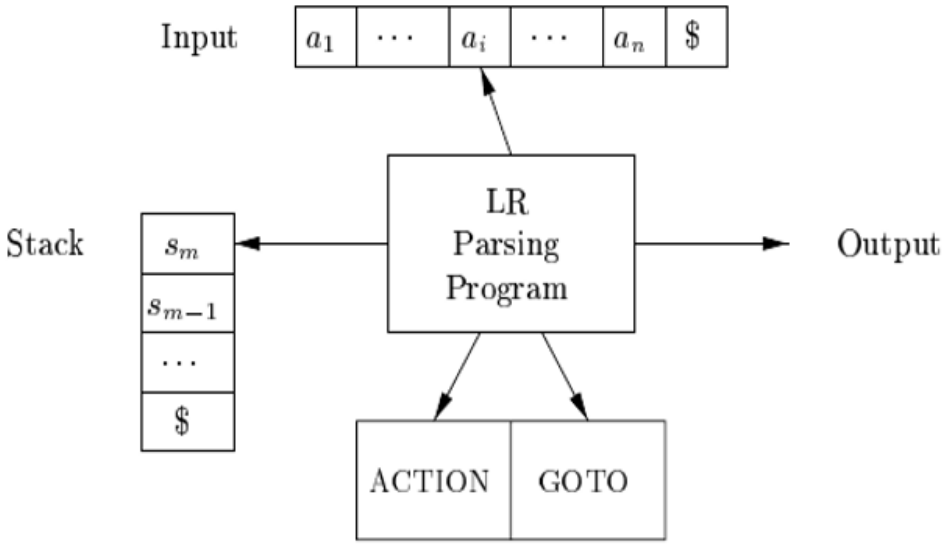
  - LR(0)
  - SLR *simple LR*

Análisis sintáctico

Parser bottom-up

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.

Una configuración del *parser* muestra
el contenido de la pila y el resto del input que falta por procesar.



Análisis sintáctico

analizador SLR

Analizador bottom-up donde la tabla de parsing es ligeramente diferente al del analizador LR(0):

Input: Una gramática aumentada.

Output: La tabla de parsing SLR con las funciones ACTION and GOTO para la gramática.

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(0) $C = \{I_0, I_1, \ldots, I_n\}$.
2. El estado i se construye desde I_i y las acciones están determinadas como sigue:
   - Si el item $A \to \alpha \bullet a\beta$ está en I_i y $\text{GOTO}(I_i, a) = I_j$ entonces $\text{ACTION}(i, a) =$ shift a donde a es un símbolo terminal.
   - Si el item $A \to \alpha\bullet$ está en I_i entonces $\text{ACTION}(i, a) =$ reduce$A \to \alpha$ para toda a en $\text{FOLLOW}(A)$ donde A no es el nuevo símbolo de inicio en la gramática aumentada.
   - Si el item $S \to A\bullet$ está en I_i entonces $\text{ACTION}(i, a) =$ 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

analizador SLR

$S' \to S$ $\quad$ $L \to \star R \mid \text{id}$ $\quad$ $S \to L = R \mid R$ $\quad$ $R \to L$

Análisis sintáctico

analizador SLR

$S' \to S$ $\quad$ $L \to \star R \mid \text{id}$ $\quad$ $S \to L = R \mid R$ $\quad$ $R \to L$

Análisis sintáctico SLR

- Un analizador LR(0) tiene conflictos si para una entrada de la tabla se pueden realizar dos acciones diferentes.
- Un analizador SLR puede construir una tabla con más entradas y posiblemente eliminar conflictos si se lee por adelantado uno o más símbolos de la cadena de entrada.

Análisis sintáctico SLR

- Un analizador LR(0) tiene conflictos si para una entrada de la tabla se pueden realizar dos acciones diferentes.
- Un analizador SLR puede construir una tabla con más entradas y posiblemente eliminar conflictos si se lee por adelantado uno o más símbolos de la cadena de entrada.
- En general, al incluir símbolos por adelantado (look-aheads) mejora la toma de decisiones:

Análisis sintáctico SLR

- Un analizador LR(0) tiene conflictos si para una entrada de la tabla se pueden realizar dos acciones diferentes.
- Un analizador SLR puede construir una tabla con más entradas y posiblemente eliminar conflictos si se lee por adelantado uno o más símbolos de la cadena de entrada.
- 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

Análisis sintáctico SLR

- Un analizador LR(0) tiene conflictos si para una entrada de la tabla se pueden realizar dos acciones diferentes.
- Un analizador SLR puede construir una tabla con más entradas y posiblemente eliminar conflictos si se lee por adelantado uno o más símbolos de la cadena de entrada.
- 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.

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.