Compiladores 24-2
Análisis Sintáctico: manejo de errores en *parsers bottom-up*

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

Facultad de Ciencias, UNAM

8 abril 2024

Análisis sintáctico LR

errores

- Los errores en un analizador LR son las entradas de la tabla de parsing en donde no es posible realizar una acción para un item y un símbolo.

Análisis sintáctico LR

errores

- Los errores en un analizador LR son las entradas de la tabla de parsing en donde no es posible realizar una acción para un item y un símbolo.
- Los errores sólo serán producidos por las acciones y no por las transiciones en la parte de la tabla para la función GOTO.

M:

state	action (shift & state / reduce & rule) terminal	goto (state) variable
⋮		
$\mathcal{I}_i$	$s\mathcal{I}_k$ / rM	
$\mathcal{I}_j$	rN / rR	
⋮		

Análisis sintáctico LR

errores

- Los errores en un analizador LR son las entradas de la tabla de parsing en donde no es posible realizar una acción para un item y un símbolo.
- Los errores sólo serán producidos por las acciones y no por las transiciones en la parte de la tabla para la función GOTO.

M:

state	action (shift & state / reduce & rule) terminal	goto (state) variable
⋮		
$\mathcal{I}_i$	$s\mathcal{I}_k$ / rM	
$\mathcal{I}_j$	rN / rR	
⋮		

★ La finalidad del compilador es que al menos la etapa del análisis sintáctico termine, por lo tanto el estado del parser en donde se encuentra el error debe de adaptarse para continuar con el procesamiento del programa, se puede ignorar un parte o alterar la cadena de entrada.

Análisis sintáctico LR

errores

- Cuando un parser *detecta* un error puede informar al usuario a través de mensajes. El mínimo mensaje para esto debería ser *“input contains syntax error(s)”*

Análisis sintáctico LR

errores

- Cuando un parser *detecta* un error puede informar al usuario a través de mensajes. El mínimo mensaje para esto debería ser *“input contains syntax error(s)”*
- Algunos parsers tienen la propiedad llamada *prefijo correcto* (*correct-prefix*) que significa que se detecta un error en el primer símbolo del input que no puede usarse para construir una expresión del lenguaje.
  Esta propiedad no se detecta en el diseño de la gramática sino hasta el análisis de alguna cadena.

Análisis sintáctico LR

errores

- Cuando un parser *detecta* un error puede informar al usuario a través de mensajes. El mínimo mensaje para esto debería ser *“input contains syntax error(s)”*
- Algunos parsers tienen la propiedad llamada *prefijo correcto* (*correct-prefix*) que significa que se detecta un error en el primer símbolo del input que no puede usarse para construir una expresión del lenguaje.
  Esta propiedad no se detecta en el diseño de la gramática sino hasta el análisis de alguna cadena.
- La recuperación de errores es la adaptación del parser para terminar el análisis y entonces tratar de corregirlo.

Análisis sintáctico LR

errores

- Cuando un parser *detecta* un error puede informar al usuario a través de mensajes. El mínimo mensaje para esto debería ser *“input contains syntax error(s)”*
- Algunos parsers tienen la propiedad llamada *prefijo correcto* (*correct-prefix*) que significa que se detecta un error en el primer símbolo del input que no puede usarse para construir una expresión del lenguaje.
  Esta propiedad no se detecta en el diseño de la gramática sino hasta el análisis de alguna cadena.
- La recuperación de errores es la adaptación del parser para terminar el análisis y entonces tratar de corregirlo.

In summary, error detection, error recovery, and error correction require increasing levels of heuristics. Error detection itself requires no heuristics: a parser detects an error, or it does not. Determining the place where the error occurs may require heuristics, however. Error recovery requires heuristics to adapt the internal parser state so that it can continue, and error correction requires heuristics to repair the input. [3, Parsing Techniques]

Manejo de errores en LR

Modo pánico

Esta forma de manejo **ignora** la posible subcadena que contiene un error y continúa con el análisis, reportando el error.

Manejo de errores en LR

Modo pánico

Esta forma de manejo **ignora** la posible subcadena que contiene un error y continúa con el análisis, reportando el error.

Cuando se llega a una entrada de la tabla con error, se conoce el estado del parser:
el símbolo no-terminal que llevó a ese estado y los símbolos en la pila.

Manejo de errores en LR

Modo pánico

Esta forma de manejo **ignora** la posible subcadena que contiene un error y continúa con el análisis, reportando el error.

Cuando se llega a una entrada de la tabla con error, se conoce el estado del parser:
el símbolo no-terminal que llevó a ese estado y los símbolos en la pila.

El error se maneja al

1. Revisar la pila de estados para encontrar un estado $\mathcal{I}_j$ en donde la transición GOTO ha usado el símbolo no-terminal A.

Manejo de errores en LR

Modo pánico

Esta forma de manejo **ignora** la posible subcadena que contiene un error y continúa con el análisis, reportando el error.

Cuando se llega a una entrada de la tabla con error, se conoce el estado del parser:
el símbolo no-terminal que llevó a ese estado y los símbolos en la pila.

El error se maneja al

1. Revisar la pila de estados para encontrar un estado $\mathcal{I}_j$ en donde la transición GOTO ha usado el símbolo no-terminal A.
2. Sacar de la pila cero o más símbolos hasta encontrar un símbolo b que sigue a A.

Manejo de errores en LR

Modo pánico

Esta forma de manejo **ignora** la posible subcadena que contiene un error y continúa con el análisis, reportando el error.

Cuando se llega a una entrada de la tabla con error, se conoce el estado del parser:
el símbolo no-terminal que llevó a ese estado y los símbolos en la pila.

El error se maneja al

1. Revisar la pila de estados para encontrar un estado $\mathcal{I}_j$ en donde la transición GOTO ha usado el símbolo no-terminal A.
2. Sacar de la pila cero o más símbolos hasta encontrar un símbolo b que sigue a A.
3. Se guarda el estado resultante de $\text{GOTO}(\mathcal{I}_j, A)$ y se continúa con el análisis.

Manejo de errores en LR

Recuperación de nivel de frase

Esta opción debe examinar cada caso de la tabla de parsing en donde haya un error y decidir tomando en cuenta el uso del lenguaje el error más común que pueda ser generado por el programador.

Manejo de errores en LR

Recuperación de nivel de frase

Esta opción debe examinar cada caso de la tabla de parsing en donde haya un error y decidir tomando en cuenta el uso del lenguaje el error más común que pueda ser generado por el programador.

Estos errores se manejan al definir procedimientos especiales que usualmente modifican la pila o la cadena de entrada y complementan la tabla con otras acciones:

- agregar o eliminar símbolos de la pila o de la cadena de entrada o de ambos, por ejemplo vaciar la pila si ya se ha procesado toda la cadena de entrada;
- alterar o mover símbolos de la cadena de entrada;
- retirar un estado de la pila.

Manejo de errores en LR

Ejemplo

Considera la siguiente gramática extendida $E' \to E$ $E \to E + E \mid E * E \mid (E) \mid id$

Manejo de errores en LR

Ejemplo

Considera la siguiente gramática extendida $E' \to E \qquad E \to E + E \mid E * E \mid (E) \mid id$

Los conjuntos de items para la tabla de parsing siguiendo el método LR(0) son:

I_0: $E' \to \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot \mathbf{id}$

I_1: $E' \to E\cdot$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_2: $E \to (\cdot E)$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot \mathbf{id}$

I_3: $E \to \mathbf{id}\cdot$

I_4: $E \to E + \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot \mathbf{id}$

I_5: $E \to E * \cdot E$
$E \to \cdot E + E$
$E \to \cdot E * E$
$E \to \cdot(E)$
$E \to \cdot \mathbf{id}$

I_6: $E \to (E\cdot)$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_7: $E \to E + E\cdot$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_8: $E \to E * E\cdot$
$E \to E\cdot + E$
$E \to E\cdot * E$

I_9: $E \to (E)\cdot$

Manejo de errores en LR

Ejemplo

Considera la siguiente gramática extendida: $E' \to E \qquad E \to E + E \mid E * E \mid (E) \mid id$

La tabla de parsing correspondiente y la que maneja errores son:

STATE	ACTION						GOTO
	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	

STATE	ACTION						GOTO
	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	

Manejo de errores en LR

Ejemplo

Considera la siguiente gramática extendida: $E' \to E \qquad E \to E + E \mid E * E \mid (E) \mid id$
La tabla de parsing correspondiente y la que maneja errores son:

STATE	ACTION						GOTO
	id	+	*	(	)	$	E
0	s3			s2			1
1		s4	s5			acc	
2	s3			s2			6
3		r4	r4		r4	r4	
4	s3			s2			7
5	s3			s2			8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	

STATE	ACTION						GOTO
	id	+	*	(	)	$	E
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	

Los huecos de la tabla son errores que se pueden manejar usando subrutinas/funciones especiales para cada caso:

- *e*1 “ese símbolo terminal no debe estar ahi”
- *e*2 “paréntesis desbalanceados”
- *e*3 “se espera un operador”
- *e*4 “paréntesis derecho faltante”

Manejo de errores en LR

Ejemplo

e1: Esta subrutina es llamada en los estados I_0, I_2, I_4 e I_5 que son los estados que esperan leer el inicio de una operación, ya sea un *id* o un paréntesis izquierdo.
Pero la cadena de entrada tiene otro símbolo terminal o el fin de cadena de entrada.
El error se maneja al agregar el estado I_3 y devolver un mensaje *missing operand*.

e2: Se llama en los estados I_0, I_2, I_4 e I_5 donde se encuentra un paréntesis derecho.
El error se maneja al remover el paréntesis derecho de la cadena de entrada y devolver el mensaje *unbalanced right parenthesis*.

e3: Esta subrutina es llamada en los estados I_1 e I_6 que son los estados que esperan leer un operador y se encuentran con un *id* o un paréntesis derecho.
El error se maneja al agregar el estado I_4 y devolver un mensaje *missing operator*.

e4: Se llama en el estado I_6 cuando se encuentra el símbolo de fin de entrada o cadena.
El error se maneja al agregar el estado I_9 y devolver un mensaje *missing right parenthesis*.

Manejo de errores en LR

Ejemplo

Considera la siguiente gramática $E \rightarrow E + E \mid E * E \mid (E) \mid id$

Análisis de la cadena $id +)$

STATE	ACTION						GOTO
	id	+	*	(	)	$	*E*
0	s3	e1	e1	s2	e2	e1	1
1	e3	s4	s5	e3	e2	acc	
2	s3	e1	e1	s2	e2	e1	6
3	r4	r4	r4	r4	r4	r4	
4	s3	e1	e1	s2	e2	e1	7
5	s3	e1	e1	s2	e2	e1	8
6	e3	s4	s5	e3	s9	e4	
7	r1	r1	s5	r1	r1	r1	
8	r2	r2	r2	r2	r2	r2	
9	r3	r3	r3	r3	r3	r3	

Parsing LR

- 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 (shift-reduce), incluye transiciones (goto) y está generada por una autómata finito de conjuntos de items.
- De entre todos los tipos de parsers LR, es decir de las variantes de construcción de la tabla, el que es más eficiente es el LALR.
- La tabla se complementa para manejar errores.

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] Dick Grune and Ceriel J. H. Jacobs.
Parsing Techniques: A Practical Guide.
Ellis Horwood, USA, 1990.

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

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