Compiladores 24-2
Análisis Semántico: gramáticas con atributos

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

Facultad de Ciencias, UNAM

15 abril 2024

Análisis Semántico

- Última fase del compilador en la parte de front-end que verificará la corrección del programa más alla de la forma, es decir a nivel semántico, respecto a la definición del lenguaje de programación en particular.
- El análisis semántico puede ser descrito en téminos de anotaciones o **decoraciones** en el árbol sintáctico.
- Las anotaciones son llamadas **atributos**.
- Los atributos y su relación con los tokens fueron obtenidos por el analizador léxico y está almacenado en la tabla de símbolos.

★ Una gramática con atributos es un complemento a una gramática libre de contexto al anotar las producciones con atributos.

Análisis Semántico

tipos de atributos

Atributo sintético obtiene su valor de un enunciado hacia la izquierda en una producción.

Los símbolos terminales tienen propiedades intrínsecas y es por esto que son atributos sintéticos, obtienen su valor del programa original a través de la información recabada en la tabla de símbolos desde el lexer.

Análisis Semántico

tipos de atributos

Atributo sintético obtiene su valor de un enunciado hacia la izquierda en una producción.

Los símbolos terminales tienen propiedades intrínsecas y es por esto que son atributos sintéticos, obtienen su valor del programa original a través de la información recabada en la tabla de símbolos desde el lexer.

Atributo heredado obtiene su valor cuando el mismo símbolo no-terminal está a la derecha de la producción o al usar valores o atributos de otros símbolos.

Es decir que la información contextual en el árbol sintáctico debe fluir de un símbolo en un nodo superior o en el mismo nivel.

Las reglas de producción de la gramática de atributos heredados pueden ser usadas muchas veces para obtener diferentes valores dependiendo del contexto y al heredar desde la información que está almacenada en la tabla de símbolos.

Análisis Semántico

atributos heredados Ejemplo 4.7 del libro [5], p.p.188-189.

Definir las funciones o reglas semánticas que permitan realizar una **ejecución simbólica** de las restas (expresiones de la gram. simplificada)

$$E \to \texttt{const}\, T \qquad\qquad T \to -\texttt{const}\, T \mid \varepsilon$$

Análisis Semántico

calcular atributos

- Una gráfica de dependencias establece el flujo de la información entre las instancias de los atributos en el parse tree.
- Las aristas son las restricciones de las reglas semánticas y los nodos son los diferentes atributos asociados a cada símbolo.
- Esta gráfica es particular e independiente a cada parse tree.
- Existen tres **esquemas de traducción** para aplicar las reglas de la gramática con atributos y obtener el valor de los atributos en un árbol
    - Esquema Inadvertido (*oblivious*)
    - Esquema Dinámico
    - Esquema Estático

Tipos de gramáticas con atributos

- ✓ Una gramática es **S-atribuida** si todos sus atributos son sintéticos y los argumentos de las funciones semánticas usan únicamente símbolos en la parte derecha de la producción.
  - El resultado de un atributo es la parte izquierda de la producción.
  - Estas gramáticas están asociadas a los parsers LR.
  - Este tipo de gramática favorece que los atributos puedan ser calculados al vuelo en la fase de análisis sintáctico, es decir al mismo tiempo que se calcula el parse-tree.
  - Es más eficiente.

Tipos de gramáticas con atributos

- ✓ Una gramática es **L-atribuida** si los atributos de los nodos son evaluados al visitar los nodos del parse tree de una sola vez de izquierda a derecha (**L**eft-to-right) y en un recorrido a profundidad:
  - cada atributo sintético a la izquierda depende de los heredados o de la parte derecha
  - cada atributo heredado a la derecha depende de los atributos heredados de la izquierda o de los atributos de la parte izquierda
- Estas gramáticas están asociadas a los parsers LL que generan derivaciones por la izquierda.

Ejemplo: gramática expresiones aritméticas

Gramática de alto nivel:

$$E \to E + T \mid E - E \mid E \star E \mid E/E \mid (E) \mid \texttt{const}$$

Ejemplo: gramática expresiones aritméticas

Gramática de alto nivel:

$$E \to E + T \mid E - E \mid E \star E \mid E/E \mid (E) \mid \texttt{const}$$

Gramática tipo LL:
no ambigua y sin recursión izquierda.

$$\begin{array}{rcl} E & \to & T\ T' \\ T' & \to & +T\ T' \\ T' & \to & -T\ T' \\ T' & \to & \varepsilon \\ T & \to & F\ F' \\ F' & \to & \star F\ F' \\ F' & \to & /F\ F' \\ F' & \to & \varepsilon \\ F & \to & -F \\ F & \to & (\ E\ ) \\ F & \to & \texttt{const} \end{array}$$

Gramática tipo LR: no ambigua.

$$\begin{array}{rcl} E & \to & E + T \\ E & \to & E - T \\ E & \to & T \\ T & \to & T \star F \\ T & \to & T/F \\ T & \to & F \\ F & \to & -F \\ F & \to & (\ E\ ) \\ F & \to & \texttt{const} \end{array}$$

Gramática **S-atribuida**

ejemplo

Agregar funciones semánticas a la siguiente gramática para que construya un árbol de sintaxis:

$$
\begin{array}{rcl}
E_1 & \rightarrow & E_2 + T \\
E_1 & \rightarrow & E_2 - T \\
E & \rightarrow & T \\
T_1 & \rightarrow & T_2 \star F \\
T_1 & \rightarrow & T_2 / F \\
T & \rightarrow & F \\
F_1 & \rightarrow & -F_2 \\
F & \rightarrow & (E) \\
F & \rightarrow & \texttt{const}
\end{array}
$$

Gramática **S-atribuida**

ejemplo

Agregar funciones semánticas a la siguiente gramática para que construya un árbol de sintaxis:

$$
\begin{array}{rcl}
E_1 & \rightarrow & E_2 + T \\
E_1 & \rightarrow & E_2 - T \\
E & \rightarrow & T \\
T_1 & \rightarrow & T_2 \star F \\
T_1 & \rightarrow & T_2 / F \\
T & \rightarrow & F \\
F_1 & \rightarrow & -F_2 \\
F & \rightarrow & (E) \\
F & \rightarrow & \texttt{const}
\end{array}
$$

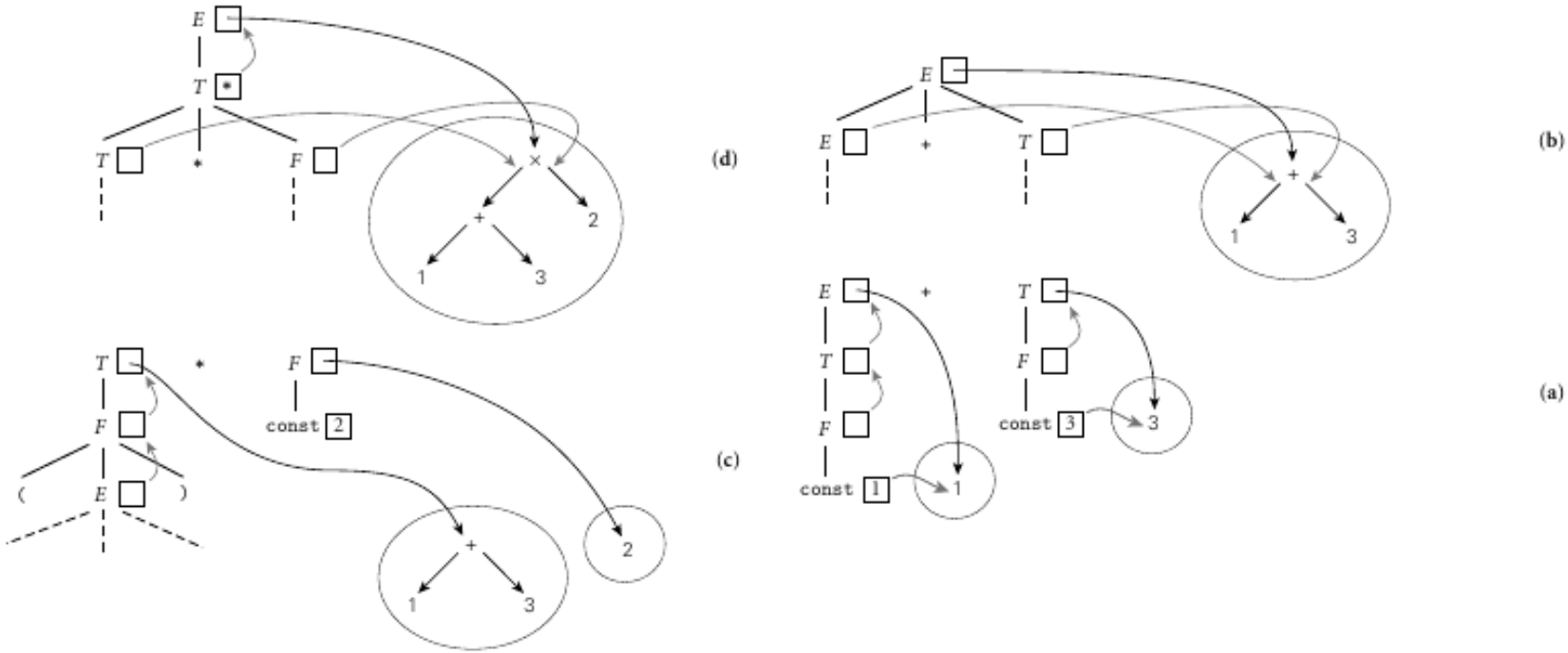
- construir hojas para los terminales
- construir nodos para las expresiones
- usar apuntadores para los nodos

Gramática **S-atribuida**

E_1	$\rightarrow$	$E_2 + T$	$E_1.\texttt{ptr} := \texttt{make\_bin\_op}(\text{“}+\text{”}, E_2.\texttt{ptr}, T.\texttt{ptr})$
E_1	$\rightarrow$	$E_2 - T$	$E_1.\texttt{ptr} := \texttt{make\_bin\_op}(\text{“}-\text{”}, E_2.\texttt{ptr}, T.\texttt{ptr})$
E	$\rightarrow$	T	$E.\texttt{ptr} := T.\texttt{ptr}$
T_1	$\rightarrow$	$T_2 \star F$	$T_1.\texttt{ptr} := \texttt{make\_bin\_op}(\text{“}\times\text{”}, T_2.\texttt{ptr}, F.\texttt{ptr})$
T_1	$\rightarrow$	T_2/F	$T_1.\texttt{ptr} := \texttt{make\_bin\_op}(\text{“}\div\text{”}, T_2.\texttt{ptr}, F.\texttt{ptr})$
T	$\rightarrow$	F	$T.\texttt{ptr} := F.\texttt{ptr}$
F_1	$\rightarrow$	$-F_2$	$F_1.\texttt{ptr} := \texttt{make\_un\_op}(\text{“}+/-\text{”}, F_2.\texttt{ptr})$
F	$\rightarrow$	$(\,E\,)$	$F.\texttt{ptr} := E.\texttt{ptr}$
F	$\rightarrow$	`const`	$T.\texttt{ptr} := \texttt{make\_leaf}(\textit{const.val})$

Gramática **S-atribuida**

ejemplo, Obtener el árbol de sintaxis para la cadena $(1 + 3) \star 2$



Estas figuras se leen de derecha a izquierda y de abajo hacia arrriba dado que el árbol se construye desde abajo hacia la raíz. Ejemplo del libro [5]

Gramática **L-atribuida**

ejemplo

Agregar funciones semánticas a la siguiente gramática para que construya un árbol de sintaxis:

$$
\begin{array}{rcl}
E & \rightarrow & T\ TT \\
TT_1 & \rightarrow & +T\ TT_2 \\
TT_1 & \rightarrow & -T\ TT_2 \\
TT & \rightarrow & \varepsilon \\
T & \rightarrow & F\ FT \\
FT_1 & \rightarrow & \star F\ FT_2 \\
FT_1 & \rightarrow & /F\ FT_2 \\
FT & \rightarrow & \varepsilon \\
F_1 & \rightarrow & -F_2 \\
F & \rightarrow & (\ E\ ) \\
F & \rightarrow & \texttt{const}
\end{array}
$$

Gramática **L-atribuida**

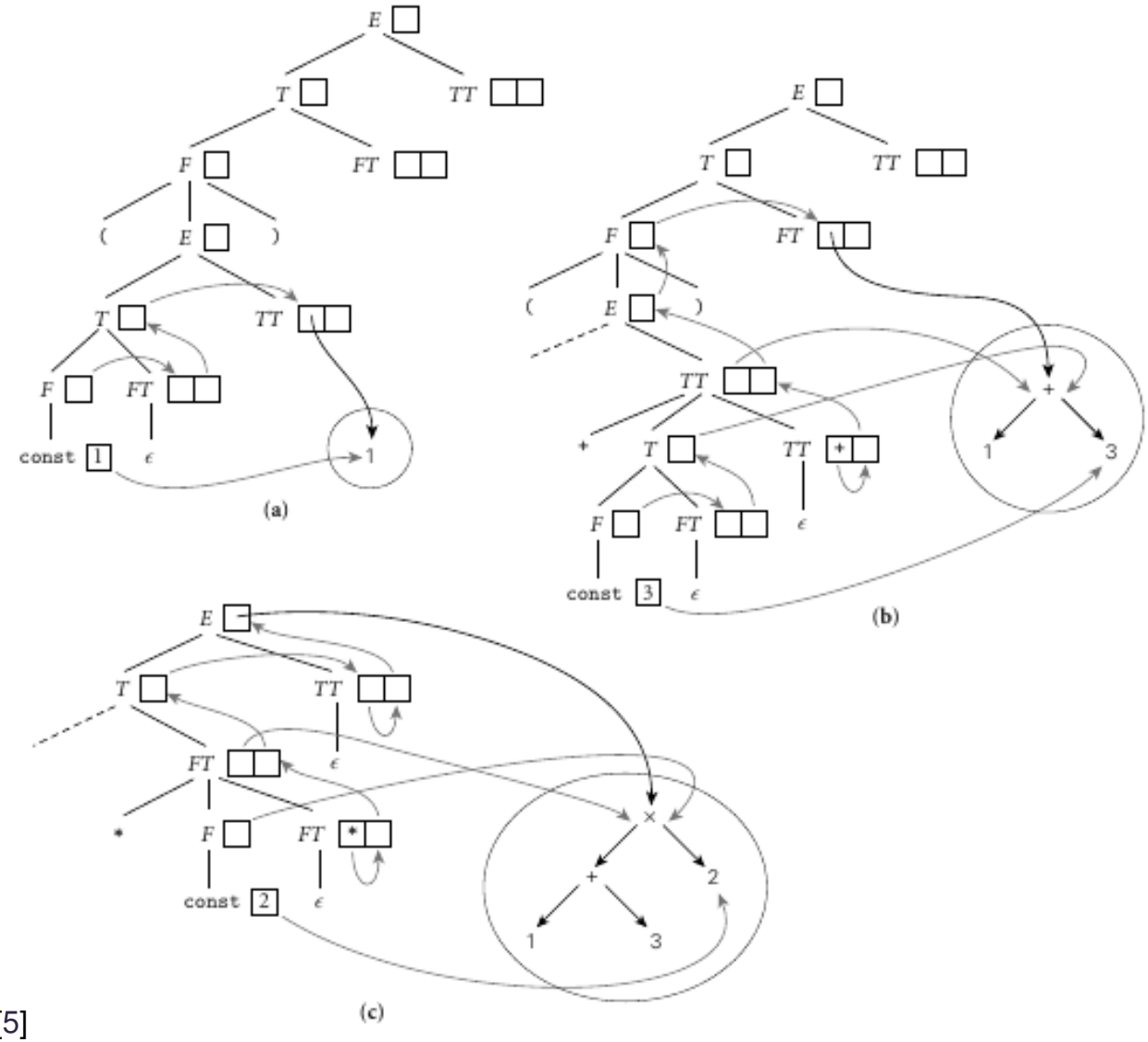
ejemplo

Agregar funciones semánticas a la siguiente gramática para que construya un árbol de sintaxis:

E	$\rightarrow$	$T\ TT$	$TT.\texttt{st} := T.\texttt{ptr}$ $\quad$ $E.\texttt{ptr} := TT.\texttt{ptr}$
TT_1	$\rightarrow$	$+\ T\ TT_2$	$TT_2.\texttt{st} := \texttt{make\_bin\_op}(\text{“}+\text{”}, TT_1.\texttt{st}, T.\texttt{ptr})$ $TT_1.\texttt{ptr} := TT_2.\texttt{ptr}$
TT_1	$\rightarrow$	$-\ T\ TT_2$	$TT_2.\texttt{st} := \texttt{make\_bin\_op}(\text{“}-\text{”}, TT_1.\texttt{st}, T.\texttt{ptr})$ $TT_1.\texttt{ptr} := TT_2.\texttt{ptr}$
TT	$\rightarrow$	ε	$TT.\texttt{ptr} := TT.\texttt{st}$
T	$\rightarrow$	$F\ FT$	$FT.\texttt{st} := F.\texttt{ptr}$ $\quad$ $T.\texttt{ptr} := FT.\texttt{ptr}$
FT_1	$\rightarrow$	$\star F\ FT_2$	$FT_2.\texttt{st} := \texttt{make\_bin\_op}(\text{“}\times\text{”}, FT_1.\texttt{st}, F.\texttt{ptr})$ $FT_1.\texttt{ptr} := FT_2.\texttt{ptr}$
FT_1	$\rightarrow$	$/F\ FT_2$	$FT_2.\texttt{st} := \texttt{make\_bin\_op}(\text{“}\div\text{”}, FT_1.\texttt{st}, F.\texttt{ptr})$ $FT_1.\texttt{ptr} := FT_2.\texttt{ptr}$
FT	$\rightarrow$	ε	$T.\texttt{ptr} := F.\texttt{st}$
F_1	$\rightarrow$	$-F_2$	$F_1.\texttt{ptr} := \texttt{make\_un\_op}(\text{“}+/-\text{”}, F_2.\texttt{ptr})$
F	$\rightarrow$	$(\ E\ )$	$F.\texttt{ptr} := E.\texttt{ptr}$
F	$\rightarrow$	$\texttt{const}$	$T.\texttt{ptr} := \texttt{make\_leaf}(\textit{const.val})$

Gramática **L-atribuida**

ejemplo, Obtener el árbol de sintaxis para la cadena $(1 + 3) \star 2$



Ejemplo del libro [5]

Gramáticas con atributos

Observaciones

- Toda gramática S-atribuida es L-atribuida pero no al revés.

Gramáticas con atributos

Observaciones

- Toda gramática S-atribuida es L-atribuida pero no al revés.
- El tipo gramática S-atribuida es la más general y es la que en la práctica se implementa junto con un parser LR.
- Si el parser es LL entonces se implementará una L-atribuida.

Gramáticas con atributos

Observaciones

- Toda gramática S-atribuida es L-atribuida pero no al revés.
- El tipo gramática S-atribuida es la más general y es la que en la práctica se implementa junto con un parser LR.
- Si el parser es LL entonces se implementará una L-atribuida.
- Los atributos decoran el árbol sintáctico y se almacenan en los nodos; esto puede realizarse de dos formas:
    1. Después del análisis sintáctico y como resultado del análisis semántico; es claro que las fases del compilador están separadas y se implementa una gramática con atributos en especial junto con un esquema de traducción apropiado.

Gramáticas con atributos

Observaciones

- Toda gramática S-atribuida es L-atribuida pero no al revés.
- El tipo gramática S-atribuida es la más general y es la que en la práctica se implementa junto con un parser LR.
- Si el parser es LL entonces se implementará una L-atribuida.
- Los atributos decoran el árbol sintáctico y se almacenan en los nodos; esto puede realizarse de dos formas:
  1. Después del análisis sintáctico y como resultado del análisis semántico; es claro que las fases del compilador están separadas y se implementa una gramática con atributos en especial junto con un esquema de traducción apropiado.
  2. Al mismo tiempo que se realiza el análisis sintáctico; es decir que la gramática tenga intercaladas funciones que permitan decorar el parse tree y generar una representación intermedia al mismo tiempo, entonces el tipo de compilador es uno *one-pass*.

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] Hanne Riis Nielson and Flemming Nielson.
Semantics with Applications: An Appetizer (Undergraduate Topics in Computer Science).
Springer-Verlag, Berlin, Heidelberg, 2007.

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

[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.