01/02/2023

Universidad Politécnica de Chiapas

Analizador Sintáctico

Compiladores e Intérpretes

Marisol Solis López 203411

Introducción

El análisis sintáctico es una parte fundamental del proceso de compilación y asegura que un programa esté escrito de acuerdo con las reglas gramaticales y sintácticas correctas. Un analizador sintáctico es el encargado de verificar la estructura de un programa y traducirlo a una representación interna, como un árbol sintáctico abstracto (AST).

Sin embargo, a menudo se producen errores sintácticos en la escritura de programas, por lo que es importante tener estrategias de corrección de errores sintácticos. Algunas de estas estrategias incluyen la recuperación ante errores, el análisis predictivo y el análisis por recuperación ante errores.

Analizador sintáctico

Un analizador sintáctico es un componente esencial de un compilador que se encarga de verificar la estructura de un programa de acuerdo a las reglas gramaticales y sintácticas correctas. Si la estructura es válida, el analizador la traduce a una representación interna, como un árbol sintáctico abstracto (AST).

El analizador sintáctico es una parte importante del proceso de compilación y asegura que el programa esté escrito de manera correcta antes de ser ejecutado. Si se encuentra algún error en la estructura del programa, el analizador sintáctico genera un error sintáctico que describe el problema y su ubicación en el programa.

Existen dos tipos principales de analizadores sintácticos: los analizadores descendentes y los analizadores ascendentes.

- Analizadores descendentes (también conocidos como parsers LL o parsers bottom-up) empiezan por las hojas del árbol sintáctico y construyen hacia arriba hasta llegar a la raíz. Ejemplos de este tipo de analizadores son el parser LR y el parser SLR.
- Analizadores ascendentes (también conocidos como parsers LR o parsers top-down) empiezan por la raíz del árbol sintáctico y construyen hacia abajo hasta llegar a las hojas. Ejemplos de este tipo de analizadores son el parser LALR y el parser LL.

En resumen, los analizadores sintácticos son herramientas utilizadas en compiladores para verificar si una entrada es sintácticamente válida según una gramática dada.

Árbol sintáctico

Un árbol sintáctico abstracto (AST) es una representación gráfica y estructurada de la sintaxis de un programa. Es una estructura de datos que se crea a partir del análisis sintáctico y se utiliza para la generación de código y la optimización

El AST es una representación intermedia que permite a los compiladores y otras herramientas de software manipular y transformar el código de manera más eficiente y fácil. Algunos ejemplos de operaciones que se pueden realizar con un AST incluyen la optimización del código, la generación de código objeto y la detección de errores

Hay varios tipos de árboles sintácticos, algunos de los más comunes incluyen:

- Árbol de análisis sintáctico abstracto (ASA o Abstract Syntax Tree, en inglés): Es una representación estructurada y simplificada del código fuente que permite una fácil manipulación y análisis por parte del compilador.
- Árbol de análisis sintáctico concreto (CST o Concrete Syntax Tree, en inglés): Es una representación detallada del código fuente, incluyendo las reglas gramaticales y los tokens específicos.
- Árbol de análisis sintáctico descentendente (Parse Tree, en inglés): Es una representación gráfica de la gramática y cómo se aplica a la entrada. Se construye a partir de la parte inferior hacia arriba.

- Árbol de análisis sintáctico ascendente (Derivation Tree, en inglés): Es una representación gráfica de cómo se produjo la entrada a partir de la gramática. Se construye desde la raíz hacia abajo.

Estos árboles son útiles para la comprensión y el análisis de lenguajes de programación y ayudan en la implementación de compiladores y otros procesadores de lenguaje.

Estrategias de corrección de errores sintácticos

Las estrategias de corrección de errores sintácticos incluyen:

- Recuperación ante errores: La recuperación ante errores es un enfoque en el que el analizador sintáctico intenta recuperarse de un error y continuar con el análisis del programa en lugar de detenerse y generar un error. Este enfoque es útil cuando se espera que el programador haga muchos errores y se quiere permitirle corregirlos sin tener que empezar de nuevo.
- Análisis predictivo: El análisis predictivo es un enfoque en el que el analizador sintáctico utiliza una gramática formal para predecir la estructura correcta del programa. Si se encuentra un error, el analizador sintáctico puede detenerse y generar un error o intentar recuperarse.
- Análisis por recuperación ante errores: El análisis por recuperación ante errores es un enfoque en el que el analizador sintáctico combina tanto la recuperación ante errores como el análisis predictivo.

Conclusión

En conclusión, los analizadores sintácticos y los árboles sintácticos son herramientas clave en el procesamiento de lenguajes de programación. Los analizadores sintácticos verifican si una entrada es sintácticamente válida según una gramática dada, mientras que los árboles sintácticos representan gráficamente la estructura y la relación entre los elementos del código fuente. Estos árboles simplifican la tarea de comprender y manipular el código y son una parte importante de la implementación de compiladores y otros procesadores de lenguaje.

Referencias:

- Compilers: Principles, Techniques, and Tools. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman. Addison-Wesley, 2006.
- Modern Compiler Implementation in ML. Andrew W. Appel. Cambridge University Press, 1998.
- "Compiler Design". N. D. Jones, et al. In Handbook of Theoretical Computer Science, vol. B, pp. 845-972. Elsevier, 1990.