Introducción al Análisis Sintáctico

El Parsing como Algoritmo

Universidad de Cantabria

Outline

El Problema de la Parsing

El problema de parsing está muy relacionado con los compiladores. Esto es, dado un programa, es necesario comprobar si “es correcto”.

El Problema de la Parsing

Esto es un *problema decisional*, por lo tanto tiene sentido intentar resolverlo con teoría de autómatas.
La novedad en este caso es que además de responder sí ó no, **querremos también tener un testigo.**

El Problema de la Parsing

Este testigo será el árbol de derivación, que da la información necesaria para construir la derivación de la palabra.
Incluso en el caso donde se ha dado una respuesta negativa podríamos dar un testigo.

Ideas

Si queremos traducir un lenguaje en otro es necesario primero de todo entender que es lo que nos dice el lenguaje fuente.
La idea fundamental en los lenguajes de programación es que **la estructura da el significado.**

Ideas

Pongamos ejemplos, ¿Qué significa en JAVA dos bucles if anidados?

¿Qué significa 2+3*4?

Ideas

Pongamos ejemplos, ¿Qué significa en JAVA dos bucles if anidados?
¿Qué significa 2+3*4?

Ideas

Volvamos al ejemplo de JAVA. String z=i+j;

- Significa que queremos sumar dos enteros.
- Significa que queremos concatenar dos strings.
- Significa que queremos convertir un entero a un string y concatenarlo.

Ideas

Volvamos al ejemplo de JAVA. String z=i+j;

- Significa que queremos sumar dos enteros.
- Significa que queremos concatenar dos strings.
- Significa que queremos convertir un entero a un string y concatenarlo.

Ideas

Volvamos al ejemplo de JAVA. String z=i+j;

- Significa que queremos sumar dos enteros.
- Significa que queremos concatenar dos strings.
- Significa que queremos convertir un entero a un string y concatenarlo.

Ideas

La estructura se encuentra en el árbol de derivación y a traves de ella podemos encontrar el significado.

Formalización

Definición

Fijado un lenguaje libre de contexto $L \subseteq \Sigma^*$, dada una palabra $\omega \in \Sigma^*$, entonces resolver:

- *Decidir si $\omega \in L$.*
- *En caso de respuesta afirmativa, dar un árbol de derivación (o una derivación) que produzca ω.*

Lenguaje Técnico

Definición (Compilador)

Un compilador es un programa (o conjunto de programas) que transforma:

- *un texto escrito en un lenguaje de programación (lenguaje fuente)*
- *en un texto escrito en otro lenguaje de programación (lenguaje objetivo)*

El texto en lenguaje fuente se llama código fuente. La salida del compilador se llama código objetivo.

Lenguaje Técnico

- **Fuente a Fuente:** transforma código entre lenguajes de alto nivel. Se suele usar el término reescritura cuando el lenguaje fuente es exactamente el lenguaje objetivo.
- **Decompilador:** el lenguaje objetivo es de mayor nivel que el lenguaje fuente. Su uso es harto infrecuente.

El Interprete

Definición

Un Intérprete es un programa (o conjunto de programas) que tienen como entrada un programa en código fuente y el programa ejecuta directamente el código fuente.

Ventajas de los Interpretes

- El programador trabaja en forma interactiva y va viendo los resultados de cada instrucción antes de pasar a la siguiente.
- El programa se utiliza sólo una vez y no importa tanto la velocidad (después se compila y se usa solamente el ejecutable).
- Se espera que cada instrucción se ejecute una sola vez.
- Las instrucciones tienen formas simples y son más fáciles de analizar.

Ejemplos de Interpretes

Ejemplo

Entre los intérpretes más habituales:

- ***shell*** *El intérprete de comandos de Unix. Es una instrucción para el sistema operativo Unix. Se introduce dando el comando de forma textual. Actúa como se ha indicado: comando a comando. El usuario puede ver la acción de cada comando.*
- ***LISP*** *Es un lenguaje de programación de procesado de Listas. Common LISP.*
- *Un Intérprete de* ***SQL****.*

Ventajas de un Intérprete

- El programador trabaja en forma interactiva y va viendo los resultados de cada instrucción antes de pasar a la siguiente.
- Las instrucciones tienen formas simples y son más fáciles de analizar.
- El manejo de la memoria suele ser a cargo del programa, lo que facilita el desarrollo de un programa.
- Suelen ser facilmente portables a otras plataformas.

Inconvenientes de los Interpretes

- La velocidad puede ser del orden de 100 veces más lento que la de un ejecutable.
- No sirve cuando se espera que las instrucciones se ejecuten frecuentemente.
- Tampoco es interesante cuando las instrucciones sean complicadas de analizar.

Compiladores de Lenguajes Interpretados

Combinan las cualidades de compiladores e intérpretes.

- Transforma el código fuente en un lenguaje intermedio.
- Sus instrucciones tienen un formato simple y fácil de analizar.
- La traducción desde el lenguaje fuente al lenguaje intermedio es fácil y rápida.

El Ejemplo de Java

En JAVA, se tiene este paradigma. Primero el código se traduce a ByteCode y este es ejecutado por la máquina virtual. Aunque tambien es posible compilarlo a lenguaje nativo de la plataforma que estemos hablando.

Reflexiones

Algunas deducciones que se plantean:

- Todos los lenguajes de programación son equivalentes al lenguaje nativo del ordenador.
- La diferencia de utilizar uno u otro lenguaje es una diferencia de eficiencia.
- Por lo tanto, tiene sentido estudiar el problema en general, como la transformación de una palabra de un lenguaje en otro.