15/10/2013

Proyecto programado 2

Creación compilador de Prolog

Yaxiri Azofeifa García
Carlos Campos Fuentes
Fabricio Soto Mejías
Michelle García Campos
Daniela Valerio Cerdas

Contenido

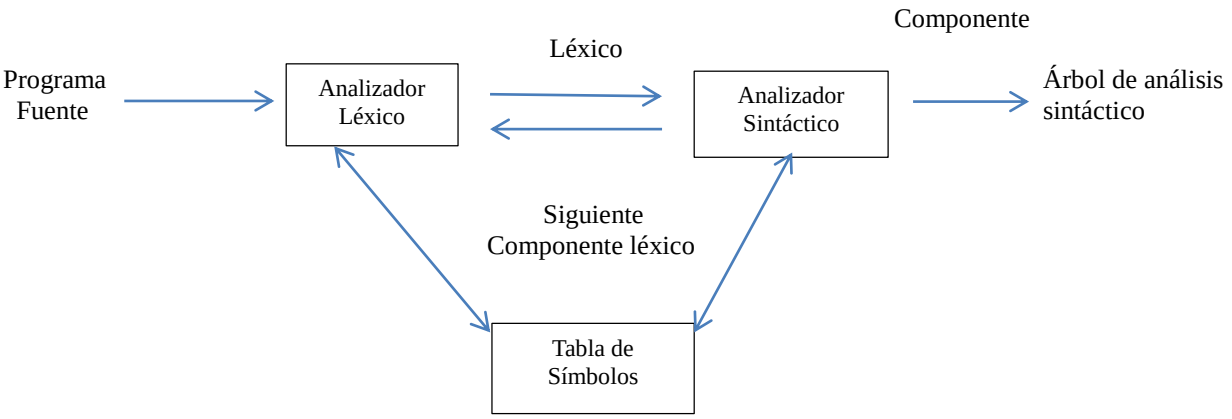
Descripción del programa:

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que son llevados a cabo por un computador. Pueden ser usados para crear programas con un fin específico. Existen diferentes paradigmas que clasifican los lenguajes, en este caso se deberá implementar uno perteneciente al paradigma lógico llamado Prolog.
En el desarrollo del programa se deberá simular el comportamiento de áreas específicas de Prolog. Por lo que se deberá implementar una base de conocimientos, que deberá ser administrada por el usuario de manera que pueda agregar la cantidad ilimitada de hechos. De la misma manera, un motor de inferencia que se encargara de revisar las reglas de inferencia solicitadas y contestar al usuario según así sea la respuesta requerida.
Modo consulta es otra característica que deberá estar implementada en el programa, en donde el usuario interactuara con el sistema por medio de consultas.
En el caso de que una regla tenga antecedentes, deberán validar los antecedentes antes de poder responder. Esto deberá ser implementado manteniendo un tipo de pila con los diferentes predicados. Se deberá implementar backtracking, para el proceso de resolución de metas.
Tanto en modo consulta como a la hora de ingresar hechos a la bese se deberá de proceder con el análisis léxico y sintáctico correspondientes a la estructura permitida por Prolog.
El programa será implementado en el lenguaje del paradigma Orientado a Objetos, Java.

Métodos Estudiados

Análisis Léxico: La fase de rastreo (scanner), tiene las funciones de leer el programa fuente como un archivo de caracteres y dividirlo en tokens. Los tokens son las palabras reservadas de un lenguaje, secuencia de caracteres que representa una unidad de información en el programa fuente. En cada caso un token representa un cierto patrón de caracteres que el analizador léxico reconoce, o ajusta desde el inicio de los caracteres de entrada. De tal manera es necesario generar un mecanismo computacional que nos permita identificar el patrón de transición entre los caracteres de entrada, generando tokens, que posteriormente serán clasificados. Este mecanismo es posible crearlo a partir de un tipo específico de máquina de estados llamado autómata finito

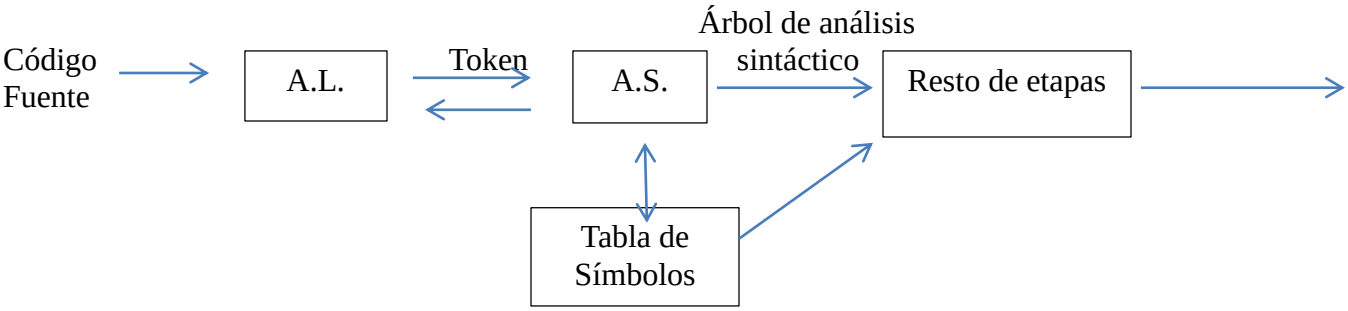
El analizador léxico opera bajo petición del analizador sintáctico devolviendo un componente léxico conforme el analizador sintáctico lo va necesitando para avanzar en la gramática. Los componentes léxicos son los símbolos terminales de la gramática. Suele implementarse como una subrutina del analizador sintáctico. Cuando recibe la orden “obtén el siguiente componente léxico”, el analizador léxico lee los caracteres de entrada hasta identificar el siguiente componente léxico.



Análisis sintáctico: El análisis sintáctico es un análisis a nivel de sentencias, y es mucho más complejo que el análisis léxico. Su función es tomar el programa fuente en forma de tokens, que recibe del analizador léxico, y determinar la estructura de las sentencias del programa. Este proceso es similar a determinar la estructura de una frase en castellano, determinando quien es el sujeto, predicado, el verbo y los complementos. El análisis sintáctico agrupa a los tokens en clases sintácticas (denominadas no terminales en la definición de la gramática), tales como expresiones, procedimientos, etc. El analizador sintáctico o parser obtiene un árbol sintáctico
(otra estructura equivalente) en la cual las hojas son los tokens, y cualquier nodo que no sea una hoja, representa un tipo de clase sintáctica (operaciones). Los árboles sintácticos se construyen con un conjunto de reglas conocidas como gramática, y que definen con total precisión el lenguaje fuente.

Al proceso de reconocer la estructura del lenguaje fuente se conoce con el nombre de análisis sintáctico (parsing). Hay distintas clases de analizadores o reconocedores sintácticos, pero en general se clasifican en 2 grandes grupos: A.S.
Ascendentes y A.S. Descendentes.

La principal tarea del analizador sintáctico no es comprobar que la sintaxis del programa fuente sea correcta, sino construir una representación interna de ese programa y en el caso en que sea un programa incorrecto, dar un mensaje de error. Para ello, el analizador sintáctico (A.S.) comprueba que el orden en que el analizador léxico le va entregando los tokens es válido. Si esto es así significará que la sucesión de símbolos que representan dichos tokens puede ser generada por la gramática correspondiente al lenguaje del código fuente.



Bactraking: Backtracking (o búsqueda atrás) es una técnica de programación para hacer búsqueda sistemática a través de todas las congelaciones posibles dentro de un espacio de búsqueda. Para lograr esto los algoritmos de tipo Backtracking construyen posibles soluciones de manera sistemática. En general, dado una solución candidata *S*:

- Verifican si *S* es solución. Si lo es, hacen algo con ella (depende del problema).

- Construyen todas las posibles extensiones de *S*, e invocan recursivamente al algoritmo con todas ellas

A veces los algoritmos de tipo backtracking se usan para encontrar una solución, pero otras veces interesa que las revisen todas.

Diseño del programa:

A continuación de especificar el diseño usado en el programa, además de la manera en que se maneja las diferentes funciones implementadas.

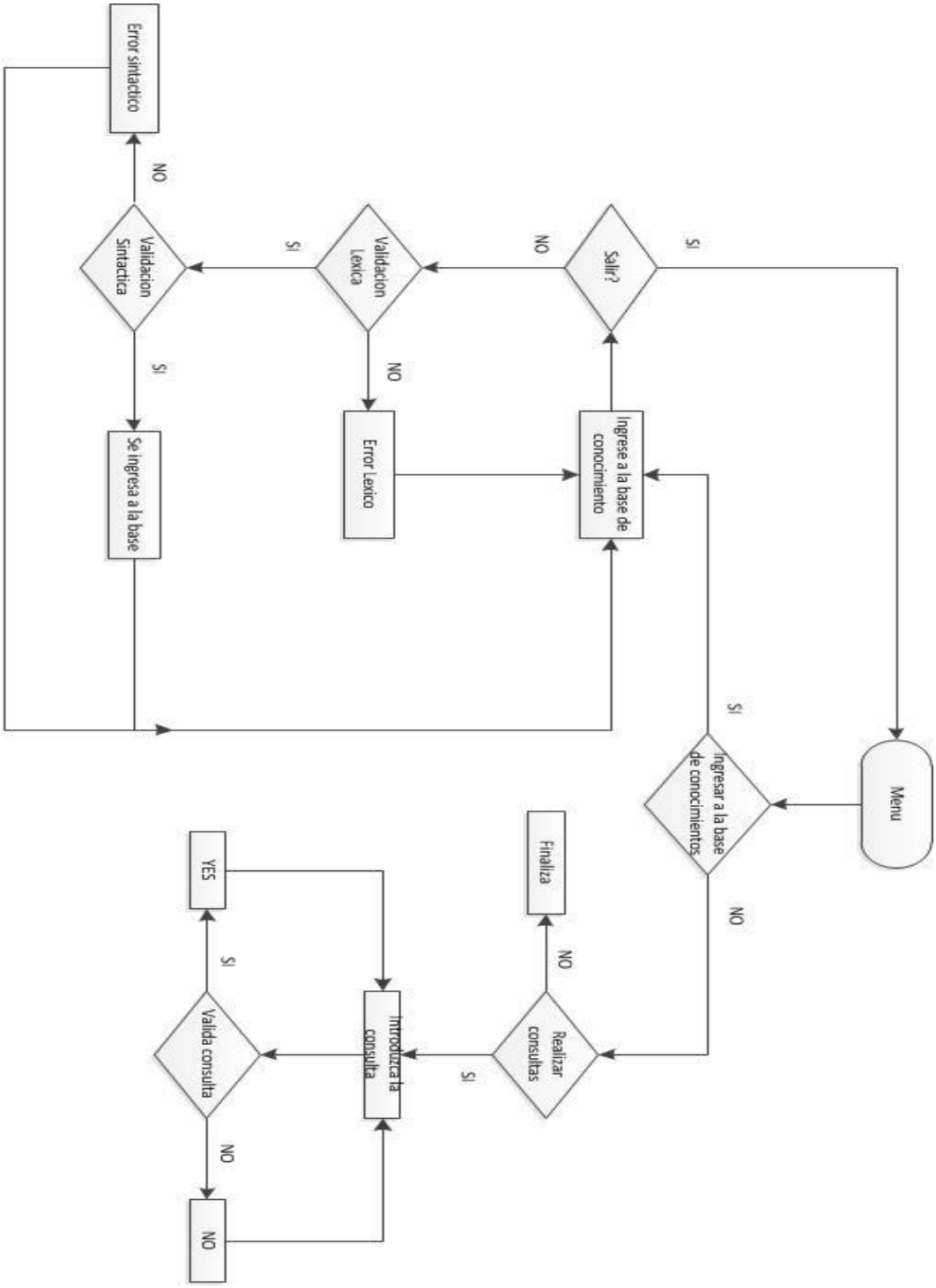
Decisión de diseño

- Se desarrolló en el lenguaje de Programación Java porque consideramos la estructura del manejo de datos de Java, más apropiada para la implementación del software.
- Se utilizó un menú en el cual se le solicita al usuario insertar una selección mediante una serie de números para acceder a las diferentes funciones del programa.
- Se diseñó el programa de manera tal que pudiera correrse desde consola solamente.
- La implementación del manejo de validaciones léxicas y sintácticas fueron implementadas de manera manual sin uso de librerías mediante el manejo de string y listas.
- Se usó el manejo de listas de adyacencia para el manejo de reglas en la base de conocimientos.

Algoritmos usados (Funciones principales del programa):

Nombre de Función	Funcionamiento	Valor de retorno
Léxico()	Consiste en recorrer el String de entrada y revisar según seas las reglas del análisis léxico de Prolog. De la misma manera se p[repara en una lista para el manejo en la lista de adyacencia.	El valor de retorno es un boolean que retornara de acuerdo a si el análisis se terminó con éxito y llamando consecuentemente al análisis sintáctico.
Sintaxis()	Consiste en la revisión de la estructura de lo ingresado por el usuario después de haber pasado por el análisis léxico, validando que se cumplan las reglas de estructura de Prolog.	El valor de retorno es un boolean que retornara de acuerdo a si el análisis se terminó con éxito y permitiendo la continuidad del proceso.
Menu()	Es la encargada de presentar al usuario las funcionalidades de la aplicación y dar la opción de escoger la deseada.	Habilitará la funcionalidad solicitada.
Consultar()	Espera la entrada del usuario para aplicar el procedimiento correspondiente de la consulta en la base de conocimientos.	Devuelve lo correspondiente al proceso de consulta en consola

Diagrama Lógico



Librerías de Java utilizadas

java.util.Scanner: De forma resumida podemos decir que cuando se introducen caracteres por teclado, la librería Scanner se encarga de tomar la cadena introducida y la divide en elementos llamados tokens. El carácter predeterminado que sirve de separador de tokens es el espacio en blanco.

Análisis de resultados

Objetivos alcanzados

- Se logró implementar las validaciones de reglas léxicas que revisa token por token antes de pasar al análisis sintáctico.
- Se logró la implementación de análisis sintáctico que se encarga de validar la estructura de lo ingresado por el usuario
- Se logró los almacenamientos de datos en la base de conocimientos.
- Se lograron las consultas .básicas que serán introducidas en el modo consulta
- En la revisión de consultas para la verificación de hechos y reglas se logró que le diera la salida al usuario según sea la solicitud.
- Se obtuvo la implementación de las funciones básicas como write, nl y fail

Objetivos no alcanzados

- No se logro las consultas que demandaran la verificación de reglas que verifican otras reglas.
- El backtracking solo funciona con reglas

Manual de usuario

Se encuentra adjunto con el nombre Manual de Usuario.pdf.

Conclusión Grupal

El proyecto programado 2, requirió de mucha investigación y análisis de comportamientos tanto del lenguaje Prolog como el manejo de datos de Java. Además se necesitó de empeño y tiempo para alcanzar objetivos.

Se implementaron muchos de los temas vistos en clase y se aumentó el conocimiento de la gran mayoría de estos.
Se notó claramente con la investigación, los diferentes comportamientos de prolog por lo que era necesario para su implementación.
Cabe mencionar que uno de las implementaciones más difíciles del proyecto fue el motor de inferencia y el manejo del backtracking.