Documentación Externa Tercera Tarea Programada

Instituto Tecnológico de Costa Rica

Compiladores e Intérpretes

Jean Carlo Cervantes Rodríguez
Luis Prado Rodríguez
Joseph Araya Rojas

Tabla de Contenido

Descripción del Problema

El objetivo de esta tarea programada consiste en desarrollar un analizador semántico para xhtml, para esto se puede utilizar el árbol sintáctico creado en la tarea anterior, o montar el analizador dentro de la gramática del parser realizado anteriormente.

El analizador deberá verificar que los valores correspondientes a los diferentes atributos HTML, estén correctos, si no lo están, el analizador debe mostrar un mensaje de error con el número de fila y columna donde se encuentra el error, además si los atributos no son validos para html5 deberá mostrar otro mensaje indicándolo.

Diseño del Programa

Analizador Semántico:

Decisión de implementación

El analizador semántico utiliza el árbol sintáctico generado en la tarea programada pasada.

No se implemento dentro de la gramática, pues eso, en este caso, implicaba una gran cantidad de trabajo adicional e innecesario pues el árbol permitía una implementación más simple.

Funcionamiento:

El analizador básicamente lo que realiza, es un recorrido por todos los elementos del árbol sintáctico.

Por cada elemento se hace un sub ciclo con los atributos del elemento, en cada iteración de este sub ciclo se verifica que el atributo sea válido para el elemento actual, si esto se cumple se verifica que el valor que contiene ese atributo sea válido.

Métodos de verificación:

Expresiones regulares

Uno de los métodos para realizar las verificaciones fue utilizar expresiones regulares, mediante la librería de unix: **regex**.

Se utilizaron las siguientes.

```
1. languageCode= "^[a-z]{2}$";
2. url= "^(http\\://)?[a-zA-Z0-9\\-
   \\.]+\\.(com|org|net|mil|edu|COM|ORG|NET|MIL|EDU)$";
3. caracter =  "^[a-zA-Z0-9]$";
4. texto = "^[a-zA-Z0-9\\-\\.]+$";
5. textoWithSpace = "^[a-zA-Z0-9\\-\\.\\ ]+$";
6. numero= "^[0-9]+$";
7. codificacion = "^(ISO\\-8859\\-(1|2|3|4|5|6|7|8|9|10|15)|ISO\\-2022\\-(P|JP-
   2|KR)|UTF\\-(8|16))$";
8. usemap = "^#[a-zA-Z0-9\\-\\.]+$";
9. fecha = "^(19|20)\\d\\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])$";
10. sizes = "^([0-9]{2}[0-9]{2})+$";
11. rgb = "^rgb\\([0-9]{1,3}, [0-9]{1,3}, [0-9]{1,3}\\)$";
12. hexNum= "^#[0-9a-fA-F]+$";
13. bgcolor = "(^[a-zA-Z0-9\\-\\.]+$|^rgb\\([0-9]{1,3}, [0-9]{1,3}, [0-
    9]{1,3}\\)$|^#[0-9a-fA-F]+$)";
```

Para realizar este proceso se compara el valor del atributo con la expresión regular asociada a esta, si la comparación resulta positiva se acepta como valido el valor, pero si no, se muestra un mensaje de error.

Comparaciones:

Otro de los métodos para realizar la verificación de los valores fue el comparar mediante or's el valor del atributo, este se compara con todos los valores validos, si la comparación resulta positiva el analizador continua con otro atributo, pero si no, muestra un error semántico.

Ejemplo:

```
if(!strcmp(atributo,"rel")){
        if( !strcmp(valorAtributo,"alternate") || !strcmp(valorAtributo,"author"))
                return 1;
        else{
                printErrorSemantico(elemento, atributo,  fila, columna);
                return -1;
        }
}
```

Mostrar errores:

Para mostrar los errores al usuario: se implementaron 2 métodos

```
void printWarnings(char* elemento, int filaValor, int columnaValor,int tipo);
void printErrorSemantico(char* elemento, char* atributo, char* valorAtributo, int
fila, int columna, int tipo);
```

1. **printWarnigs**: Muestra un mensaje de advertencia cuando los elementos solo esta disponibles para html5 o están obsoletos.
2. **printErrorSemantico**: Muestra un mensaje de error cuando corre que un elemento contiene un atributo invalido o un atributo contiene un valor invalido.

Librerías Externas

Para el desarrollo de esta tarea programada se utilizo la librería regex que permite generar expresiones regulares ejecutables en c, además de las librerías de flex y bison.

Análisis de Resultados

Rubro	Porcentaje Alcanzado
Validación de atributos posibles para cada elemento HTML	100%
Validación del formato de los valores para cada atributo HTML	100%
Mostrar advertencias y errores semánticos	100%

Manual de Usuario

1. Asegurarse de tener instalado flex y bison.
2. Abrir la Terminal.
3. Ubicarse en la dirección donde se encuentran los archivos del analizador semántico.
4. Ejecutar make en la terminal.
5. Ejecutar ./analyzer <nombre_del_archivo_de_prueba.

Conclusión Personal

Jean Carlo:

El uso de headers y sistemas de control de versiones hace la programación más ordenada y útil, para detectar problemas de forma más sencilla y rápida.

Joseph:

Se debe estar al tanto de las actualizaciones periódicas del lenguaje que se esté trabajando, para así reflejar dichos cambios adecuadamente en los informes de errores y advertencias

Luis:

La creación del analizador semántico no es un proceso estándar, para cada lenguaje se debe implementar de una forma distinta, por este motivo los programadores deben crear sus propios métodos de validación de acuerdo a las necesidades.

Bibliografía

etutorials. (2008). Recuperado el 20 de 5 de 2013, de HTML & XHTML: http://etutorials.org/Misc/html+xhtml/Appendix+A.+HTML+Grammar/Section+A.2.+The+Grammar/

GNU. (2012). Recuperado el 28 de 5 de 2013, de Bison 2.7: http://www.gnu.org/software/bison/manual/bison.html

Stackoverflow. (27 de Abril de 2011). Recuperado el 31 de Mayo de 2013, de How does flex support bison-location exactly?: http://stackoverflow.com/questions/656703/how-does-flex-support-bison-location-exactly/5811596#5811596

w3schools. (s.f.). Recuperado el 25 de Mayo de 2013, de HTML - XHTML: http://www.w3schools.com/html/html_xhtml.asp