Universidad de La Laguna

E. T. S. DE INGENIERÍA INFORMÁTICA
Departamento de E.I.O. y Computación
Compiladores

La construcción de Thompson

Factor de ponderación [0-10]: 9

7.1. Objetivos

La Construcción de Thompson es un algoritmo que a partir de una expresión regular (ER) permite construir un Autómata Finito no Determinista (NFA) que reconoce el lenguaje regular representado por la ER.

El objetivo de esta práctica es diseñar y desarrollar un programa que dada una ER genere el NFA correspondiente utilizando para ello la Construcción de Thompson.

7.2. Descripción

Considérese la siguiente gramática:

1. <rexp> ::= <rexp> | <rexp>
2. <rexp> ::= <rexp> <rexp>
3. <rexp> ::= <rexp> *****
4. <rexp> ::= **(** <rexp> **)**
5. <rexp> ::= **letra**

Se trata de una gramática (ambigua) que genera expresiones regulares y en la que el símbolo no-terminal <rexp> representa una ER.

La primera fase de la práctica consiste en obtener una gramática no ambigua equivalente a la anterior. Para ello, se ha de modificar la gramática de forma que refleje la precedencia y asociatividad de los operadores de ERs (disyunción, concatenación y asterisco). Estos operadores son todos asociativos a izquierdas y el orden de precedencia, de mayor a menor es: asterisco, concatenación y disyunción.

A continuación se escribirá un analizador léxico y un analizador sintáctico descendente recursivo para la gramática que se obtenga. El analizador léxico se puede construir fácilmente simplificando el que se ha utilizado en prácticas anteriores, puesto que el lenguaje de las expresiones regulares, tal como lo hemos definido tiene un número pequeño de símbolos terminales.

Modificar el analizador sintáctico de forma que como resultado del análisis, si la ER es sintácticamente correcta devuelva un puntero al árbol sintáctico (AST) de la expresión regular. Para ello bastará con definir una estructura (dinámica) adecuada para almacenar

los diferentes tipos de nodos del AST correspondiente a la ER, y hacer que cada uno de los procedimientos de análisis devuelva un puntero al nodo correspondiente. Por ejemplo, el prototipo de la función *factor()* sería:

```
nodo *factor(void);
```

En caso de que la ER no sea sintácticamente correcta, el analizador sintáctico debería indicar en qué posición de la cadena de entrada (las ERs se escribirán siempre en una única línea de texto) se encuentra el error. El analizador sintáctico no realizará ningún tipo de recuperación de errores.

La construcción de Thompson consiste en recorrer el AST desde las hojas hacia la raíz (utilizar para ello una función `recorrer_arbol()`) construyendo el autómata asociado a cada nodo en función de los autómatas asociados a sus hijos y del valor de la raíz del árbol (función `crear_nfa()`).

Implementar una función que recorra el AST de la expresión y almacene además en cada nodo del AST un puntero al NFA correspondiente a la subexpresión del nodo. Será necesario disponer de funciones que creen los NFAs correspondientes a nodos de disyunción, concatenación, estrella y a nodos hojas. Los nodos hojas en el AST corresponderán con símbolos individuales (letras en nuestro caso) o bien con la ER vacía (ϵ). Los prototipos para estas funciones podrían ser los siguientes:

```
automata *nfa_disyun(nodo *raiz);
automata *nfa_concat(nodo *raiz);
automata *nfa_estre(nodo *raiz);
automata *nfa_hoja(nodo *raiz);
```

donde El identificador *raiz* se refiere a la raíz del nodo correspondiente a un determinado subárbol.

Para unificar resultados (para conseguir que el NFA correspondiente a una ER dada sea el mismo para todos los equipos de prácticas), estableceremos el convenio de que cuando construyamos el NFA asociado a la ER $a|b$, incluiremos primero los estados del NFA correspondiente a a y luego los de b, análogamente haremos con la ER ab.

El algoritmo de Thompson garantiza que cada uno de los autómatas construidos tiene un único estado de entrada, y uno solo de aceptación, verificándose además que el estado de aceptación no tiene transiciones.

Finalmente, una vez construido el NFA correspondiente a la ER, se imprimirá en un fichero en formato `.dot`.

El formato `.dot` es el que utiliza la aplicación Graphviz (`http://www.graphviz.org/`) para representar gráficos. Para instalar Graphviz en Ubuntu basta ejecutar (como usuario root): `apt-get install graphviz`.

En el servidor `ftp` de la ETSII

`ftp://ftp.etsii.ull.es/asignas/AUTOMALF/img/AF/`

pueden encontrar diversos ejemplos de NFAs en formato gráfico `SVG` con su correspondiente fichero `.dot` de "código fuente".

La Figura 7.1 muestra un ejemplo de un fichero en formato `.dot`.

```
digraph finite_state_machine {
  rankdir=TB;
  start [shape = point, width="0", height="0"];
  node [shape = doublecircle]; 1;
       node [shape = doublecircle]; 2;
       node [shape = doublecircle]; 3;
  node [shape = circle, charset = UTF-8];
  start -> 0
  0 -> 1 [ label = "~" ];
  0 -> 2 [ label = "~" ];
  0 -> 3 [ label = "~" ];
  1 -> 1 [ label = "a,b" ];
  2 -> 2 [ label = "b,c" ];
  3 -> 3 [ label = "a,c" ];
}
```

Figura 7.1: Fichero .dot del NFA
`ftp://ftp.etsii.ull.es/asignas/AUTOMALF/img/AF/nfa_david_galles_04-18.svg`

Para generar un fichero gráfico a partir del fichero de especificación (`.dot`) basta con utilizar `dot -Tjpg -o fichero.jpg fichero.dot` (mirar `man dot` para una explicación completa de las diferentes posibilidades).

Una ϵ-transición se representará mediante el carácter ~ (código ASCII 126).

Todos los ficheros de NFAs que se creen deben llevar en las primeras líneas un comentario que indique la ER que representa al lenguaje que el NFA reconoce. Recomendamos que todos los ficheros de NFAs que se generen lleven al menos tres líneas de comentarios como primeras líneas del fichero, para que el usuario escriba allí cualesquiera otros comentarios oportunos sobre el fichero en cuestión.

El programa resultante (`thompson`) se invocará en la línea de comandos del siguiente modo:

```
thompson fich_er fich_nfa
```

Donde `fich_er` será el fichero que contenga la expresión regular y `fich_nfa` será el fichero de salida que almacenará el NFA resultante.

7.3. Notas

Nótese que una vez construido el autómata asociado a la raíz, no son necesarios los autómatas asociados a sus hijos y por tanto podremos liberar la memoria ocupada por ellos.

Recomendamos la utilización de expresiones regulares de complejidad creciente para comprobar y depurar el funcionamiento del programa que se ha de diseñar.

7.4. Bibliografía

De especial ayuda en la realización de esta práctica resultará la lectura del Algoritmo 3.3 (*Construcción de Thompson*) del texto de Aho, Sethi y Ullman *Compiladores: principios, técnicas y herramientas*, páginas 124-128.

`http://www.graphviz.org/`

Drawing Graphs with Dot. `http://www.graphviz.org/Documentation/dotguide.pdf`