Universidad Nacional Autónoma de México
Facultad de Ciencias

Compiladores

Práctica 2

Laura Itzel Rodríguez Dimayuga
(nombre 2)
(nombre 3)

Ejercicio 1

Indica los valores asignados a w, x, y y z. en los siguientes dos códigos estructurados por bloques. Muestrala tabla de símbolos en cada bloque con una implementación imperativa en cada caso:

```
int w,x,y,z;
int i = 4; int j = 5;
{
    int j =7;
    i =6;
    w = i+j;
}
x = i+j;
{
    int i = 8;
    y = i+j;
}
z = i+j;
```

```
int w,x,y,z;
int i = 3; int j = 4;
{
    int i = 5;
    w = i+j;
}
x = i+j;
{
    int j = 6;
    i = 7;
    y = i+j;
}
z = i+j;
```

Solución.

Variable	Scope	Comentario
int i =4	Bloque 1	Nueva variable
int j = 5	Bloque 1	Nueva variable
int j = 7	Bloque 2	Nueva variable, solo queda en el Bloque 2
i=6	Bloque 2	Afecta al Bloque 1
x = 6 + 5 = 11	Bloque 1	Afecta al Bloque 1
int i =8	Bloque 3	Nueva variable, solo queda en el Bloque 3
y = 13	Bloque 3	Afecta al Bloque 1
z = 6 + 5	Bloque 1	El valor de i cambio, pero el de j es el del primer reglon
Final		w = 13, x = 11, y = 13, z = 11

Cuadro 1: Tabla de símbolos y valores para el primer código estructurado por bloques

Variable	Scope	Comentario
int i =3	Bloque 1	Nueva variable
int j =4	Bloque 1	Nueva variable
int i = 5	Bloque 1	Afecta solo al Bloque 1
w = 5 + 4	Bloque 2	Asignación, afecta al bloque 1
x = 3 + 4	Bloque 1	Asignación, afecta al bloque 1
j=6	Bloque 3	Afecta al Bloque 3
i= 7	Bloque 3	Afecta al Bloque 1
int i =8	Bloque 3	Nueva variable, solo queda en el Bloque 3
y = 7 + 6 = 13	Bloque 3	Afecta al Bloque 1
z = 7 + 4 = 11	Bloque 1	El valor de i cambio, pero el de j es el del primer reglón
Final		w = 9, x = 7, y = 13, z = 11

Cuadro 2: Tabla de símbolos y valores para el segundo código estructurado por bloques

■

Ejercicio 2

Divide el siguiente programa en C++ en lexemas y genera los tokens correspondientes:

```
 float limitedSquare(x) float x; {
     /* return x-squared, bit never mover than 100 */
     return(x <= -10.0 || x>=10.0) ? 100 : x*x;
 }
```

Solución.

Después del escaneo no tenemos comentarios. Ni espacios en blanco. Lexemas: `float, limitedSquare, (, x, ), float, x, ;,, return, (, x, <=, -10.0, ||, x, >=, 10.0, ), ?, 100, :, x, *, x, ;,`

Tokens:

```
<float>
<id, limitedSquare>
<(>
<id, x>
<)>
<float>
<id, x>
<;>
<{>
<return>
<(>
<id, x>
<<=>
<float,-10.0>
<||>
<id, x>
<>=>
<float,10.0>
<)>
<?>
<100>
<:>
<id, x>
<*>
<id, x>
<;>
<}>
```

■

Ejercicio 3

Define una función recursiva que compute los prefijos de una expresión regular. La base de tal función recursiva es:

$$prefix(\varepsilon) = \{\varepsilon\}$$
$$prefix(a) = \{a\}$$

Completa la definición.

Solución. Para definir los prefijos de una expresión regular, podemos considerar las siguientes reglas:

Podemos definir las reglas que aplicamos para cada una de las operaciones básicas de las expresiones regulares, si tenemos que R y S son expresiones regulares y a es un símbolo, entonces:

$$\begin{aligned} prefix(\epsilon) &= \{\epsilon\} \\ prefix(a) &= \{\epsilon, a\} \\ prefix(RS) &= prefix(R) \cup R \circ prefix(S) \\ prefix(R|S) &= prefix(R) \mid prefix(S) \\ prefix(R*) &= R^* \circ prefix(R) \end{aligned}$$

■

Ejercicio 4

Para las siguientes expresiones regulares, da el lenguaje que definen:

1. $[ab][cd\epsilon]$
2. $[a - zA - Z]^* at^*$
3. $ca[tr]$

Solución.

1. $[ab][cd\epsilon]$ define el lenguaje $\{a, b, ac, ad, bc, bd\}$
2. $[a - zA - Z]^* at^*$ define el lenguaje de (letras)at^*, es decir,todas cadenas de letras seguidas de `a` y luego de cero o más `t`. Por ejemplo: `at`, `bat`, `cat`, `a`, `rattt`, etc.
3. $ca[tr]$ define el lenguaje $\{cat, car\}$

■

Ejercicio 5

Para el siguiente automata, finito no determinista, construye el automata finito determinista equivalente.

Solución.

1. Primero, identificamos los estados iniciales:

$$S_0 = \{q_0\} \cup \{r : q \in S_0 \wedge S(q, \epsilon)\} \qquad \text{No hay epsilon transiciones}$$
$$S_0 = \{q_0\}$$

2. Ahora, construimos la función de transición para el autómata determinista. Para cada estado en S_0,

determinamos las transiciones para cada símbolo de entrada.

$$
\begin{aligned}
\delta(S_0, a) &= \{q_1\}\ cup \epsilon - closure(\{q_1\}) \\
&= \{q_1\} = S_1 \\
\delta(S_0, b) &= \varnothing \\
\delta(S_1, a) &= \{q_2\} \cup \epsilon - closure(\{q_2\}) \\
&= \{q_2\} = S_2 \\
\delta(S_1, b) &= \{q_0, q_2\} \cup \epsilon - closure(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3 \\
\delta(S_2, a) &= \{q_3\} \cup \epsilon - closure(\{q_3\}) \\
&= \{q_3, q_0\} = S_4 \\
\delta(S_2, b) &= \varnothing \\
\delta(S_3, a) &= \{q_1, q_3\} \cup \epsilon - closure(\{q_1, q_2, q_3\}) \\
&= \{q_0, q_1, q_3\} = S_5 \\
\delta(S_3, b) &= \varnothing \\
\delta(S_4, a) &= \{q_1, q_2\} \cup \epsilon - closure(\{q_1, q_2\}) \\
&= \{q_1, q_2\} = S_6 \\
\delta(S_4, b) &= \varnothing \\
\delta(S_5, a) &= \{q_1, q_2\} \cup \epsilon - closure(\{q_1, q_2\}) \\
&= \{q_1, q_2\} = S_6 \\
\delta(S_5, b) &= \{q_0, q_2\} \cup \epsilon - closure(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3 \\
\delta(S_6, a) &= \{q_2, q_3\} \cup \epsilon - closure(\{q_2, q_3\}) \\
&= \{q_0, q_2, q_3\} = S_7 \\
\delta(S_6, b) &= \{q_0, q_2\} \cup \epsilon - closure(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3 \\
\delta(S_7, a) &= \{q_1, q_2, q_3\} \cup \epsilon - closure(\{q_1, q_2, q_3\}) \\
&= \{q_0, q_1, q_2, q_3\} = S_8 \\
\delta(S_7, b) &= \varnothing \\
\delta(S_8, a) &= \{q_1, q_2, q_3\} \cup \epsilon - closure(\{q_1, q_2, q_3\}) \\
&= \{q_0, q_1, q_2, q_3\} = S_8 \\
\delta(S_8, b) &= \{q_0, q_2\} \cup \epsilon - closure(\{q_0, q_2\}) \\
&= \{q_0, q_2\} = S_3
\end{aligned}
$$

Como no tenemos nuevos estados terminamos. Ahora construimos los estados Finales

$$F = \{S_0, S_3, S_4, S_7, S_8\}$$

∎

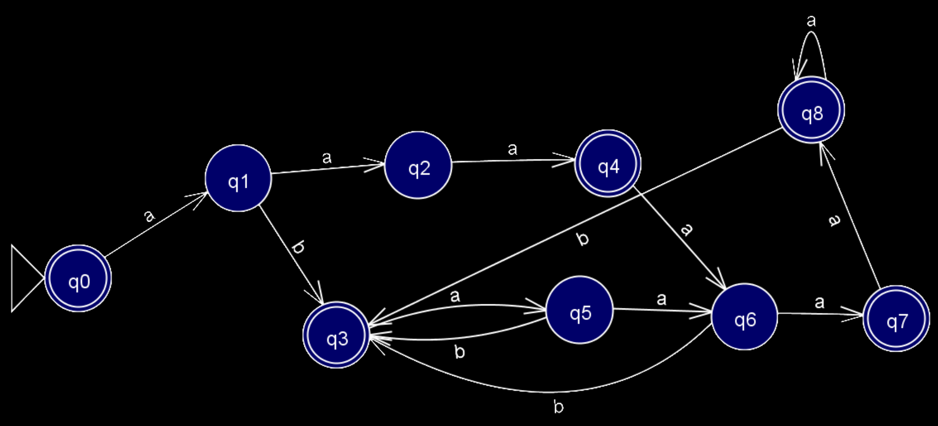

Figura 1: Automata Finito Determinista, ejercicio 5

Ejercicio 6

Para los siguientes DFA obten el DFA mínimo:

1.

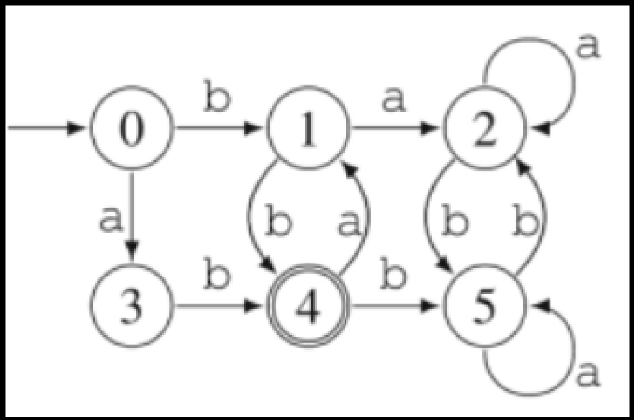

Figura 2: Automata Finito Determinista, ejercicio 6 b

2. Separemos nuestros estados en dos grupos: los estados de aceptación y los que no lo son.
   - Grupo 1 (aceptación): $\{q_4\}$
   - Grupo 2 (no aceptación): $\{q_0, q_1, q_2, q_3, q_5\}$

   Veamos como se comportarían los estados del Grupo 2 con las entradas posibles:

Estados	**Transición con a**	**Transición con b**
q_0	Grupo 1	Grupo 1
q_1	Grupo 1	Grupo 2
q_2	Grupo 1	Grupo 1
q_3	-	Grupo 2
q_5	Grupo 1	Grupo 1

Cuadro 3: Tabla de ejemplo 6×3

Vamos a agrupar los estados que se comportan igual:

- Grupo A: $\{q_0, q_2, q_5\}$
- Grupo B: $\{q_1\}$
- Grupo C: $\{q_3\}$

Veamos como se comportarían los estados del Grupo A con las entradas posibles:

Estados	**Transición con a**	**Transición con b**
q_0	Grupo C	Grupo B
q_2	Grupo A	Grupo A
q_5	Grupo A	Grupo A

Cuadro 4: Tabla de ejemplo 6×3

Ahora, agrupamos los estados que se comportan igual:

- Grupo A1: $\{q_2, q_5\}$
- Grupo A2: $\{q_0\}$
- Grupo B: $\{q_1\}$
- Grupo C: $\{q_3\}$

Veamos como se comportarían los estados del Grupo A1 con las entradas posibles:

Estados	**Transición con a**	**Transición con b**
q_2	Grupo A1	Grupo A1
q_5	Grupo A1	Grupo A1

Cuadro 5: Tabla de ejemplo 6×3

Notemos que ambos estados se comportan igual, por lo que no es posible seguir dividiendo los grupos. Por lo tanto, los grupos finales son:

- Grupo A1: $\{q_2, q_5\}$
- Grupo A2: $\{q_0\}$
- Grupo B: $\{q_1\}$
- Grupo C: $\{q_3\}$
- Grupo D (aceptación): $\{q_4\}$

Ahora, construimos el DFA mínimo usando estos grupos como estados:

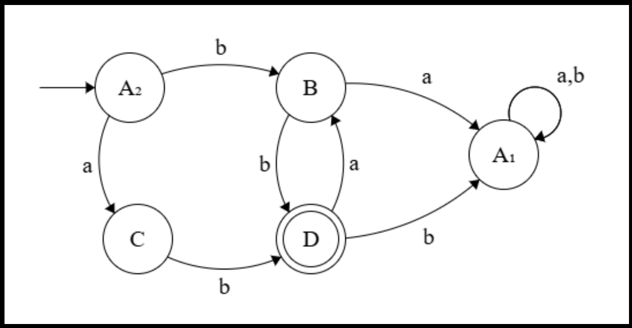

Figura 3: Automata Finito Determinista Minimo, ejercicio 6 b