Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

ANÁLISIS LÉXICO

DESDE LA EXPRESIÓN REGULAR AL AFD

Índice

Presentación

El objetivo de este tema es comprender los pasos necesarios para, a partir de una expresión regular, obtener un **autómata finito determinista (AFD)** con el mínimo número de estados. Primero se obtendrá, a partir de la ER, el **autómata finito no determinista (AFND)** para obtener el AFD equivalente a partir de este y, finalmente, obtener el **AFD mínimo que reconoce el mismo lenguaje.**

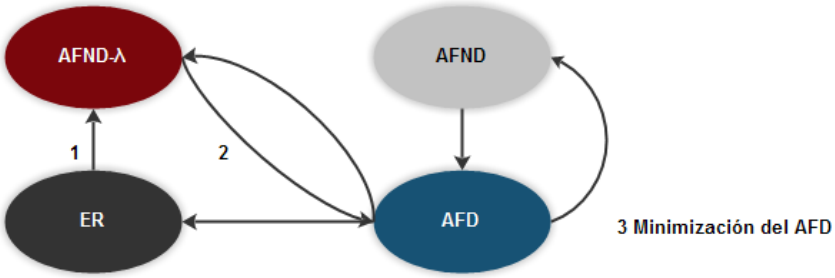
Los contenidos a tratar en este tema son:

- Paso de una ER a un AFD.
- Obtención del cierre-λ.
- Obtención del mueve (estado, entrada).
- Algoritmo para pasar de un AFND al AFD equivalente.
- Ejemplo de paso de AFND a AFD.
- Minimización del AFD.
- Ejemplo de minimización del AFD.

Con los ejemplos veremos paso a paso cada uno de los algoritmos utilizados aplicado a un caso real de obtención del AFD a partir de la ER.

Paso de una ER a un AFD

En este tema referido al analizador léxico, estudiaremos cómo pasar desde **una expresión regular (ER) hasta el autómata finito determinista (AFD) mínimo que reconoce el mismo lenguaje.** Las posibilidades de paso de uno a otro pueden observarse en la siguiente figura:



Primer paso	A partir de la ER construimos el AFND-λ (llamado así porque incorpora las transiciones-λ), utilizando las construcciones de Thomson.
Segundo paso	Para el **segundo paso**, necesitamos un algoritmo que permita pasar del AFND-λ al AFD. Este algoritmo eliminará las transiciones-λ y también las transiciones a más de un estado. Para eliminar las transiciones-λ, necesitamos construir el **cierre-λ** (Louden, 2004) y para eliminar las transiciones múltiples hay que construir los conjuntos de estados que son alcanzables a partir de cada una de las entradas. La idea general tras el paso 2 es que cada estado del AFD corresponde a un conjunto de estados del AFND, por eso a este algoritmo se le denomina **construcción se subconjuntos** (Louden, 2004 y Aho et al, 1990 y 2008).
Tercer paso	**El tercer paso** consiste en la minimización del número de estados del AFD obtenido del paso 2. Se trata de conseguirlo reconociendo el mismo lenguaje que el AFND del que proviene y para ello utilizaremos otro algoritmo que tiene como entrada el AFD y que genera, como salida, otro AFD que como hemos indicado reconoce el mismo lenguaje y tiene el número mínimo de estados posibles.

Antes de ir a los algoritmos veremos las operaciones necesarias que en ellos se utilizan, como son cierre-λ(estado) o mueve (estado, entrada).

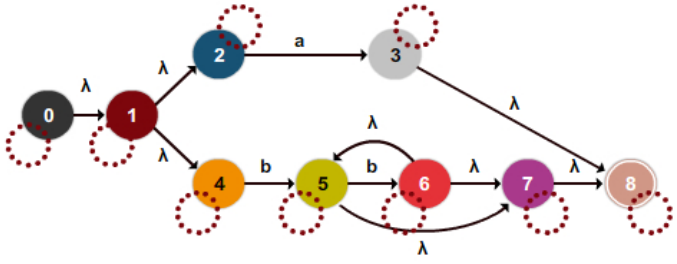
Obtención del cierre-λ

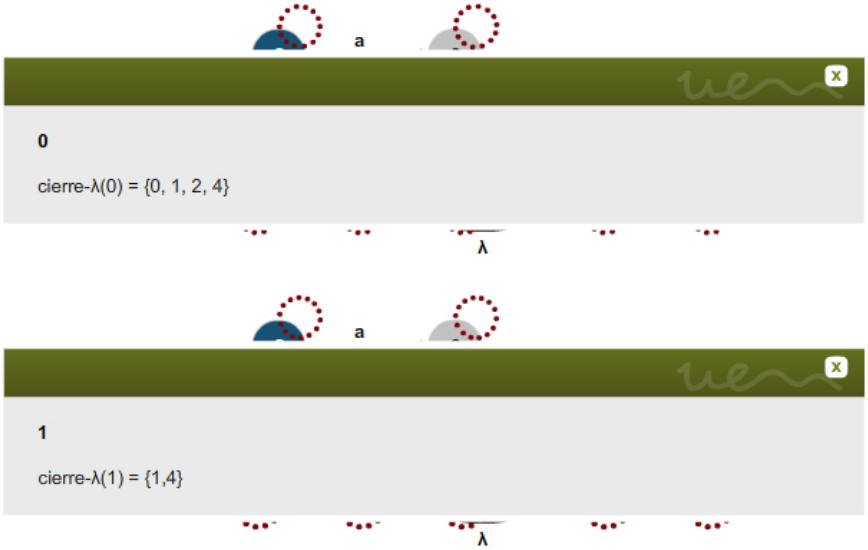
El cierre-λ de un estado es el conjunto de estados a los que se puede llegar por transiciones-λ desde ese estado incluyendo el propio estado.

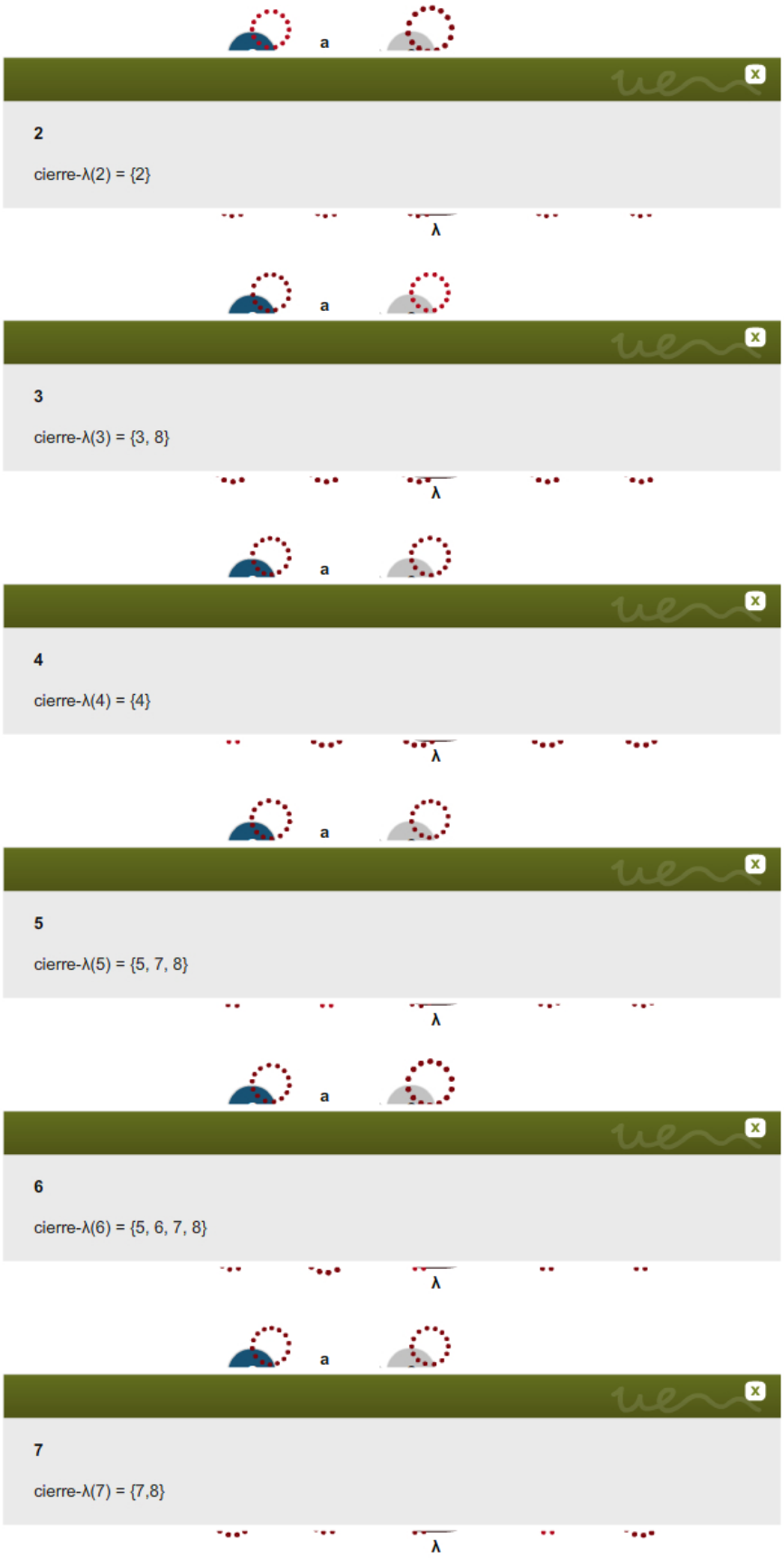
Algoritmo para el cálculo de cierre-λ (Aho et al, 1990)

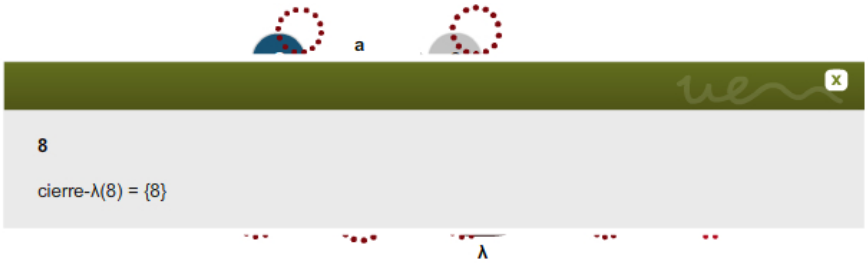
```
meter todos los estados de T en la pila;
inicializar cierre-λ(T) a T;
mientras pila no esté vacía hacer
 sacar t, el elemento del tope de la pila;
 para cada estado u con una arista desde t a u etiquetada con λ hacer
  si u no está en cierre-λ(T) hacer
   añadir u a cierre-λ(T);
   meter u en la pila
 fin-para
fin-mientras
```

Ejemplo **a|b+:**



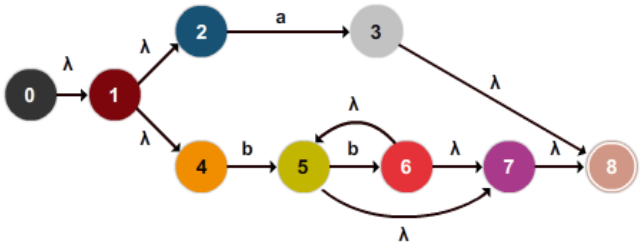






Obtención mueve (estado, entrada)

El conjunto mueve (T,a) es el conjunto de estados del AFND hacia los cuales hay transiciones con el símbolo de entrada desde los estados que partimos (T).



Continuamos con el ejemplo **a|b+,** donde los caracteres posibles que puede haber a la entrada son {a, b}.

Supongamos que al conjunto cierre-λ(0) = {0, 1, 2, 4}, lo hemos denominado A. En este caso T es A. Vamos a calcular el conjunto mueve (estado, entrada) para el carácter a, con lo que tendremos mueve (A, a). Para ello miramos el diagrama de transiciones anterior y, de ese estado A, anotamos todos los que tengan transición con a.

mueve (A, a) = {3}, puesto que con el estado 2, tenemos una transición con a al estado 3

Ahora calculamos el conjunto mueve (A, b), que es el otro posible carácter que nos podemos encontrar a la entrada:

mueve (A, b) = {5}, puesto que con el estado 4, tenemos una transición con b al estado 5

A partir de aquí, y como indican los algoritmos que vienen a continuación, se vuelve a hacer el cierre-λ de lo obtenido con el conjunto mueve (A, a), con lo que obtenemos un nuevo conjunto denominado **tranD[A, a] = B** (si es que es distinto al A en cuanto a los estados que contiene). Lo mismo sucede con mueve (A, b), obteniéndose **trandD[A, b] = C.** Estos estados tranD son a los que vamos a transitar para cada una de las entradas desde, en este caso, el estado A.

Algoritmo para el cálculo de cierre-λ

```
meter todos los estados de T en la pila;
inicializar cierre-λ(T) a T;
mientras pila no esté vacía hacer
 sacar t, el elemento del tope de la pila;
 para cada estado u con una arista desde t a u etiquetada con λ hacer
  si u no está en cierre-λ(T) hacer
   añadir u a cierre-λ(T);
   meter u en la pila
 fin-para
fin-mientras
```

T

Transiciones con el símbolo de entrada desde los estados que partimos (T)

Hay que tener en cuenta que T representa el conjunto de estados al que llegamos con transiciones-λ (cierre-λ).

Algoritmo para pasar de un AFND a un AFD

Para pasar de un AFND a un AFD, utilizaremos el algoritmo de construcciones de subconjuntos que definen Aho et al (1990), explicando algunas de las convenciones que se utilizan:

S_0	Es el estado inicial, en nuestro caso el estado 0.
T	Es el conjunto de estados de las distintas transiciones en el AFND.
tranD	Es la tabla de transiciones para el AFD. Aho et al, para abreviar, denominan D al AFD y N al AFND.
estados	Son los estados en los podría estar el AFND después de leer alguna secuencia de símbolos de entrada, incluyendo las posibles transiciones-λ que se puedan dar antes o después de la lectura de símbolos.

Todos estos conjuntos y tablas debemos ir manteniéndolos en paralelo mientras vamos haciendo las distintas operaciones que nos indica el algoritmo**: cierre-λ(S_0)**, **cierre-λ(T)** y **mueve(T, a).**

Algoritmo de construcción de subconjuntos

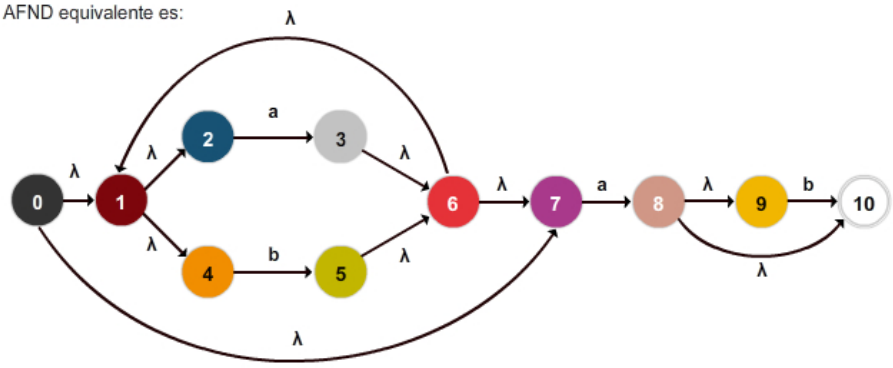
Se inicia calculando cierre-λ(S_0) e introduciendo el primer estado en estadosD, en este caso A -estado de transición T-, el cual no está marcado:

```
mientras haya un estado no marcado T en estadosD hacer
 marcar T; (en la primera iteración es A el estado que marcamos -A*-)
 para cada símbolo de entrada a hacer
  U := cierre-λ (mueve (T, a));
  si U no está en estadosD entonces
   añadir U como estado no marcado a estadosD;
  tranD[T, a] := U
 fin-para
fin-mientras
```

Ejemplo de paso de AFND a AFD

Partimos de la ER: (a|b)*ab?

El AFND equivalente es:



Estado de inicio:

- **cierre-λ(0) = {0, 1, 2, 4, 7},** a este primer estado le llamamos A (es el estado T) y se introduce en estadosD como estado no marcado.
- **estadosD = {A}** (cuando lo marquemos le pondremos un *, para saber que ya hemos hecho las operaciones mueve(T, a)).

1/7

Comienza la ejecución del algoritmo (primera iteración):

```
mientras haya un estado no marcado T en estadosD hacer (en esta primera iteración A está
no marcado)
            marcar T; (en la primera iteración es A el estado que marcamos -A*-)
            para cada símbolo de entrada a hacer (en nuestro caso los símbolos de la
            entrada son a y b)
            U:= cierre-λ(mueve (T, a))
```

estadosD ={A*}

transD={Φ} (cadena vacía)

2/7

Primero calculamos mueve (A, a)

(A,a) es el conjunto de estado del AFND que tiene transiciones con la entrada a, desde miembros de A (en este caso el estado 2 y el 7 tiene transición con a, el 2 al 3 y el 7 al 8:

A = {0, 1, 2, 4, 7}
mueve (A, a) = {3, 8} => cierre-λ(mueve (A, a)) = {1, 2, **3**, 4, 6, 7, **8**, 9, 10} = **B**

El algoritmo dice si U no está en estadosD entonces añadir U como estado no marcado a estadosD, y también transD[T, a] = U

estadosD ={A*, B}
transD[A, a] = B

Ahora calculamos mueve (A, b) y su cierre- λ

mueve (A, b) = {5} => cierre-λ(mueve (A, b)) = {1, 2, 4, 5, 6, 7} = **C**

Con esto terminamos la primera iteración y volvemos al bucle" mientras haya un estado no marcado T en estadosD hacer". Como vemos tenemos sin marcar B y C puesto que no hemos pasado por ellos.

estadosD ={A*, B, C}
transD[A, b] = C

3/7

Segunda iteración:

```
marcar T → estadosD = {A*, B*, C}
           para cada símbolo de entrada a hacer
                      U:= cierre-λ(mueve (T, a))
```

En nuestro caso hay que hacer U:= cierre-λ(mueve (B, a)) y U:= cierre-λ(mueve (B,b))

B = {1, 2, 3, 4, 6, 7, 8, 9, 10}

mueve(B, a) = {3, 8} =>
=> cierre-λ(mueve (B, a)) = {1, 2, **3**, 4, 6, 7, **8**, 9, 10} = **B**
(es el mismo estado que teníamos)

estadosD ={A*, B*, C}
transD=[B, a] = B

mueve (B, b) = {5, 10} =>
=> cierre-λ(mueve (B, b)) = {1, 2, 3, 4, **5**, 6, 7, 8, 9, **10**} = D
(nuevo estado)

estadosD ={A*, B*, C, D}
transD=[B, b] = D

Puesto que tenemos estados no marcados en T (C y D), vamos a por la tercera iteración.

4/7

Tercera iteración:

```
marcar T → estadosD = {A*, B*, C*, D}
           para cada símbolo de entrada a hacer
                      U:= cierre-λ(mueve (T, a))
```

marcar T → estadosD = {A*, B*, C*, D}

En nuestro caso hay que hacer U:= cierre-λ(mueve (C, a)) y U:= cierre-λ(mueve (C,b))
C= {1, 2, 4, 5, 6, 7}

mueve(C, a) = {3, 8} => cierre-λ(mueve (C, a)) = {1, 2, **3**, 4, 5, 6, 7, **8**, 9, 10} = **D**
(es el mismo estado que teníamos)

mueve (C, b) = {5} => cierre-λ(mueve (C, b)) = {1, 2, 4, **5**, 6, 7} = C
(nos quedamos en el mismo estado)

estadosD ={A*, B*, C*, D}
transD=[C, a] = D
transD=[C, b] = C

Como todavía nos queda un estado no marcado en T, (D), hacemos la cuarta iteración.

5/7

Cuarta iteración:

```
marcar T → estadosD = {A*, B*, C*, D*}
           para cada símbolo de entrada a hacer
                      U:= cierre-λ(mueve (T, a))
```

En nuestro caso hay que hacer U:= cierre-λ(mueve (D, a)) y U:= cierre-λ(mueve (D,b))
D = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

mueve(D, a) = {3, 8} => cierre-λ(mueve (D, a)) = {1, 2, **3**, 4, 5, 6, 7, **8**, 9, 10} = **D**
(es el mismo estado que teníamos)

mueve (D, b) = {5, 10} => cierre-λ(mueve (C, b)) = {1, 2, 3, 4, **5**, 6, 7, 8, 9, 10} = D
(es el mismo estado que teníamos)

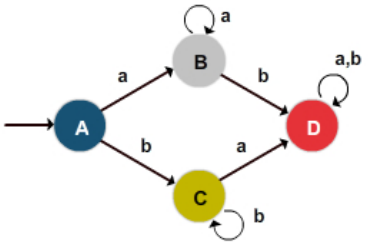
estadosD ={A*, B*, C*, D*}
transD=[D, a] = D
transD=[D, b] = D

Con esta iteración finalizamos la construcción de subconjuntos puesto que no queda ningún estado sin marcar.

6/7

Ahora ya hemos terminado puesto que tenemos todos los estados T, en estadosD marcados. Nos queda representar el AFD obtenido con su tabla y su diagrama de transiciones y para ello ponemos en la columna de estados, las entradas de la tabla transD y en entrada los dos símbolos de entrada:

ESTADO	ENTRADA	
	a	b
A	B	C
B	B	D
C	D	C
D	D	D



Teniendo en cuenta que el lenguaje que puede reconocer este autómata está formador por: a, aa, aaa, ab, ba, bab, bba, bbba, bbbab, aab, ..., los estados finales o de aceptación pueden ser B y D.

7/7

En detalle Algoritmo de construcción de subconjuntos

En detalle

Algoritmo de construcción de subconjuntos

```
mientras haya un estado no marcado T en estadosD hacer
 marcar T; (en la primera iteración es A el estado que marcamos -A*-)
 para cada símbolo de entrada a hacer
  U := cierre-λ (mueve (T, a));
  si U no está en estadosD entonces
   añadir U como estado no marcado a estadosD;
  tranD[T, a] := U
 fin-para
fin-mientras
```

Minimización del AFD

Para realizar la minimización del autómata, vamos a utilizar el algoritmo que definen Aho et al (1990) y, posteriormente, lo aplicaremos al ejemplo que hemos ido desarrollando en este tema. Consiste en lo siguiente:

- Dividir el conjunto de estados inicial en dos grupos: estados de aceptación o finales (F) y estados de no aceptación (NF).
- Aplicar este algoritmo:

```
para cada grupo hacer
        partir el grupo en subgrupos de tal forma que dos estados s y t
        están en el mismo subgrupo si, y solo si, para todos los símbolos de entrada a,
        los estados s y t tienen transiciones en a hacia estados del mismo grupo
        /*en el peor de los casos un estado estará solo en un subgrupo */
        sustituir el grupo en Pnueva por el conjunto de todos los subgrupos formados
fin-para
```

- Si P_{nueva} := P, hacer Pfinal := P y continuar con el paso 4. Si no, hacer P := P_{nueva} y repetir el paso 2.
- Escoger en cada grupo de la partición un estado representante que formará parte del AFD mínimo.
- Eliminar todos los estados inactivos (estado de no aceptación que tiene transiciones hacia el mismo con todos los símbolos de entrada). También deben eliminarse todos los estados que no sean alcanzables desde el estado inicial.

Representantes de un estado: cuando hay varios estados que tienen las mismas transiciones para los distintos símbolos de la entrada, estos estados son representados (se eliminan los otros del conjunto) por solo uno de ellos.

Escoger estados representantes

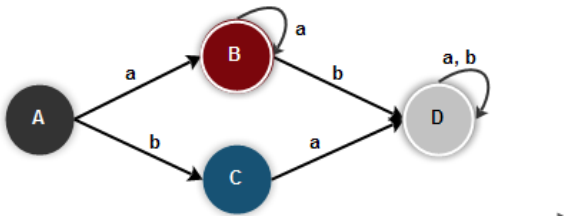
Para escoger en cada grupo de la partición un estado representante que formará parte del AFD mínimo, en primer lugar construiremos la tabla de transiciones utilizando los estados representantes y, basándonos en esta:

- El estado inicial del AFD mínimo es el representante del grupo que contiene el estado S_0 del AFD inicial.
- Los estados finales son los representantes que están en F.

Ejemplo de Minimización de un AFD

Vamos a aplicar el algoritmo a nuestro ejemplo (**(a|b)*ab?,** donde obtuvimos lo siguiente:

ESTADO	ENTRADA	
	a	b
A	B	C
B	B	D
C	D	C
D	D	D



- Dividir el conjunto de estados inicial en dos grupos: los estados de aceptación o finales (F) y los de no aceptación (NF).

En nuestro caso tendremos: F:= {B, D} y NF:= {A, C}

- Aplicar el algoritmo para obtener P_{nueva}.
- Como ya no se pueden dividir más, pasamos al paso 4.
- En este paso se trata de escoger un representante. Esto haría falta si en alguna partición hubiera más de un estado, que no es el caso pues cada partición se representa a si mismo con un solo estado.
- En los dos estados de no aceptación, A y C, no se tienen transiciones hacia ellos mismos con todos los símbolos de entrada (a y b). En nuestro caso, solo C tiene transición hacía él mismo, pero solo con b y no con a, con lo que no se puede eliminar.

Por tanto, **el AFD obtenido es el AFD mínimo** puesto que no podemos eliminar ningún estado.

En detalle Algoritmo de minimización del autómata

En detalle

Algoritmo de minimización

```
para cada grupo hacer
  partir el grupo en subgrupos de tal forma que dos estados s y t
  están en el mismo subgrupo si, y solo si, para todos los símbolos de entrada a,
  los estados s y t tienen transiciones en a hacia estados del mismo grupo
  /*en el peor de los casos un estado estará solo en un subgrupo */
  sustituir el grupo en Pnueva por el conjunto de todos los subgrupos formados
fin-para
```

Algoritmo para obtener P_{nueva}

- Los estados de F con la entrada b, que son B y D, van al mismo estado, en este caso D, pero con la entrada a van a estados diferentes, por lo que podríamos separarlos en: {B} y {D}.
- Los estados de NF con la entrada b van ambos al mismo estado, en este caso C, pero con la entrada a uno va a B y otro a D, por lo que podríamos separarlos en: {A} y {C}.

Resumen

En este tema, mediante un ejemplo completo, hemos entendido qué pasos hay que realizar para obtener: primero el AFND, después el AFD equivalente y finalmente el AFD mínimo.

Es importante entender cómo se realizan las operaciones cierre-λ y mueve (T,a) puesto que son la base del algoritmo de construcción de subconjuntos propuesto por Aho et al. (1990). A partir de estas operaciones y siguiendo el algoritmo obtenemos el AFD equivalente.

Una vez se ha obtenido este AFD, tenemos que asegurarnos de que tiene el menor número posible de estados para reconocer el lenguaje y de esta forma validar que es el más eficiente que podemos obtener y, por tanto, que no se pasa por estados inactivos o por estados inalcanzables. En estos momentos ya estamos en condiciones de programar, si fuera necesario, las expresiones regulares obteniendo el AFD mínimo.

Para finalizar, hemos realizado un ejemplo completo de paso de ER a AFND aplicando los algoritmos que hemos visto, en primer lugar para una expresión regular y su obtención del AFD y, posteriormente, para la minimización del AFD obtenido.