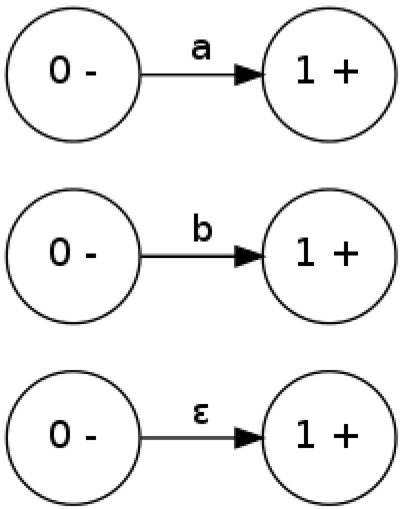
De ER a AFN

- Es útil en particular para utilitarios como el lex, que a partir de la descripción del LR mediante una ER genera el autómata para su reconocimiento.
- Si bien hay veces que la conversión se puede hacer de modo simple e intuitivo lo que queremos es un algoritmo genérico (para poder usarlo en un utilitario).
- Ken Thompson (co-autor de unix, el lenguaje Go y autor de la codificación utf-8) en 1968 desarrolló el algoritmo que vamos a explicar.
- Una de las particularidades a resaltar es que mantiene siempre un único estado final

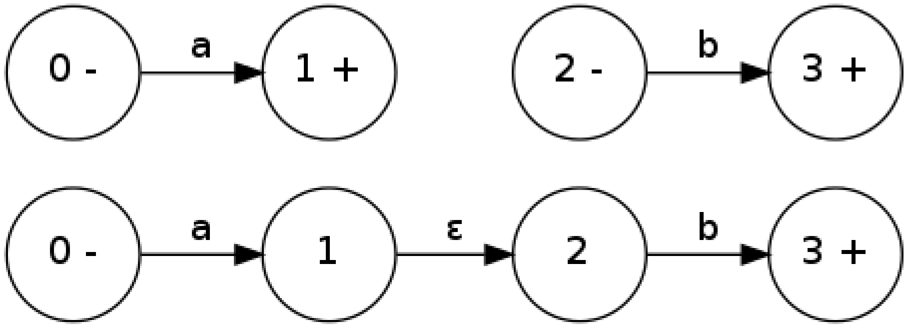
Algoritmo de Thompson

- Casos elementales
  - Para cada símbolo del alfabeto construye el autómata que lo reconoce, con un estado inicial, un arco con el símbolo en cuestión y un estado final
    - Lo mismo hace con ε



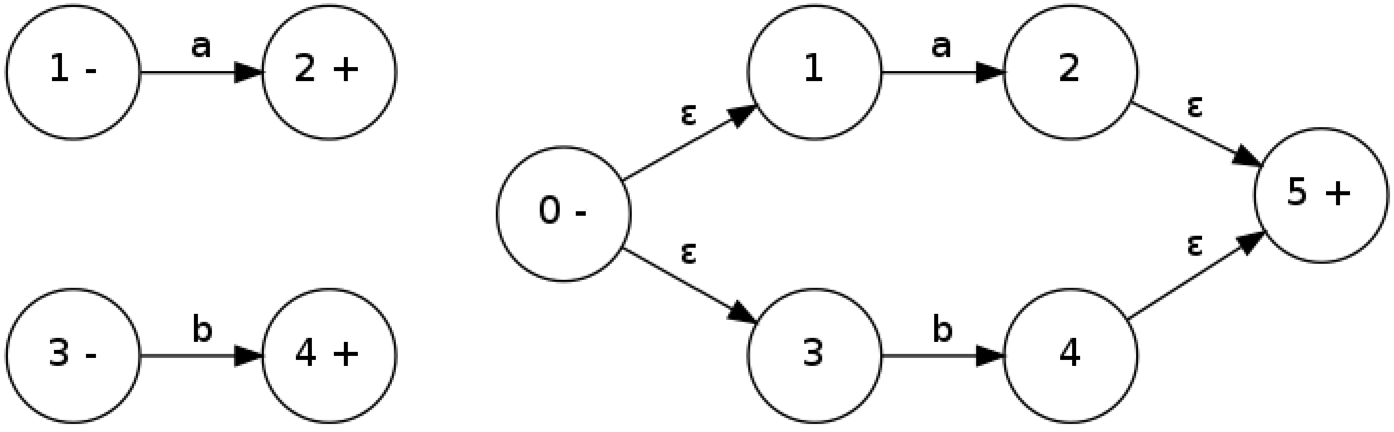
Algoritmo de Thompson

- Concatenación
  - El nodo inicial del autómata de la izquierda pasa a ser el nodo inicial de la concatenación
  - El nodo final del autómata de la derecha pasa a ser el nodo final del la concatenación
  - Desde nodo final del autómata de la izquierda agrego un arco etiquetado con ε al nodo inicial autómata de la derecha.
  - El nodo final del autómata de la izquierda y el inicial del autómata de la derecha pierden su condición especial.



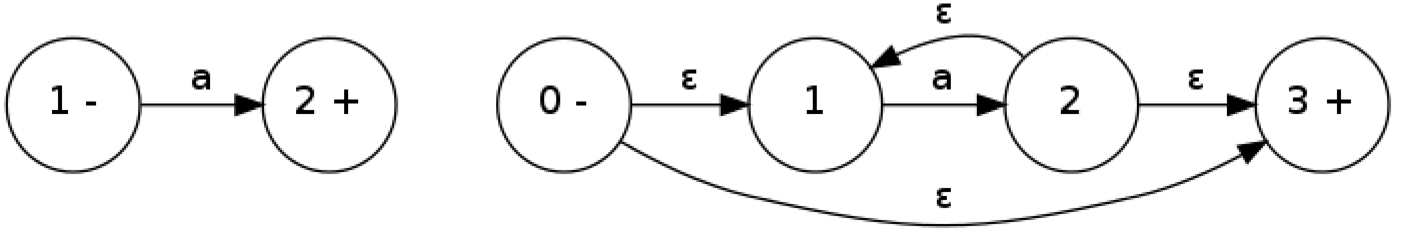
Algoritmo de Thompson

- Unión
  - Se crea un nuevo nodo inicial que se une con arcos etiquetados con ε a los anteriores nodos iniciales (que pierden esta condición)
  - Se crea un nuevo nodo final y se unen los anteriores nodos finales (que pierden esta condición) al nodo creado con arcos etiquetados con ε



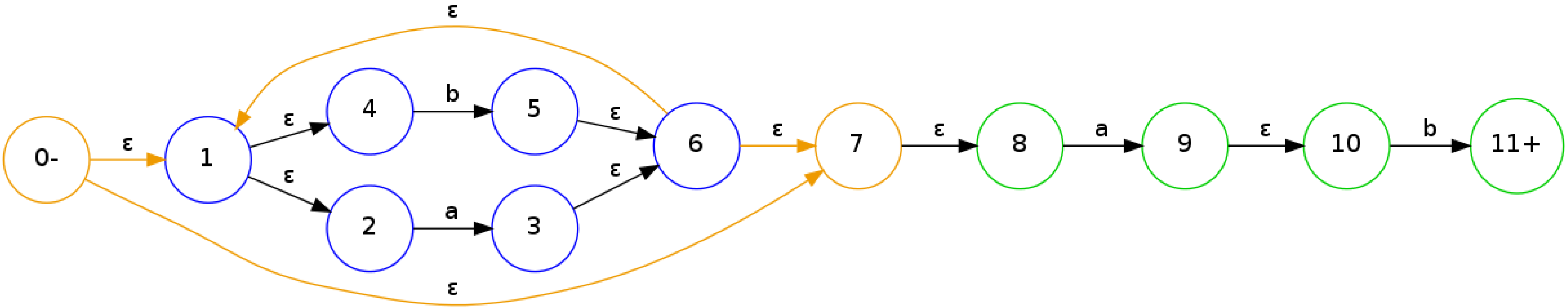
Algoritmo de Thompson

- Clausura de Kleen
  - Nuevo nodo inicial que se une a anterior nodo inicial con arco ε.
  - Nuevo nodo final y se une con arco ε desde el anterior nodo final.
  - Uno con arco ε desde el nuevo nodo inicial al nuevo nodo final.
  - Uno con arco ε desde el viejo nodo final al viejo nodo inicial.



Ejemplo

(a+b)*ab



Del AFN-ε al AFD

- La idea es “simular” todas las posibles transiciones en el AFN “en paralelo” en un AFD
- Para ello veo el conjunto de estados a donde puedo ir con un carácter y eso lo tomo como un estado del AFD equivalente que voy construyendo
- Para avanzar debemos definir dos conceptos previos
  - Función Clausura-ε
  - Función Conjunto Hacia

Clausura-ε

- Clausura-ε **de un estado** es el **conjunto** de estados al que puedo llegar a través de arcos etiquetados con ε (0, 1 o varios) incluyendo al estado del cual parto.
  - Ejemplo: clausura-ε(e_2) = {e_2, e_3, e_4}
- Algunos lo representan como ε* ya que es el conjunto de estados que puedo visitar con cero, uno o más arcos etiquetados con ε
- Clausura-ε de un **conjunto de estados** es la **unión** de los conjuntos clausura-ε de cada unos de los estados del conjunto de partida

Conjunto Hacia

- Conjunto Hacia: es una función que va del par (conjunto de estados, carácter) en conjunto de estados
  - Formalmente Hacia: $P(Q) x \Sigma \rightarrow P(Q)$
- Sea **R** un conjunto de estados y **x** un carácter del alfabeto, el conjunto Hacia(R,x) es el conjunto de estados a donde puedo llegar atravesando un arco etiquetado con **x** desde cada estado perteneciente a **R**

Algoritmo de conversión

- Se toma como estado inicial la clausura-ε del estado inicial original
- Para cada símbolo del alfabeto se calcula el conjunto Hacia desde estado inicial, a dicho conjunto se le hace la clausura-ε y se lo coloca en la columna correspondiente.
- Para cada conjunto creado, se agrega una fila etiquetada con ese conjunto y se calculan los conjuntos Hacia desde dicho conjunto y sobre el conjunto obtenido se hace la clausura-ε
- El proceso finaliza cuando al calcular los conjuntos Hacia de una fila no aparecen conjuntos nuevos
- Se marca como estados finales a **TODOS** aquellos conjuntos que contienen **al menos un estado final** del autómata original
- Para simplificar se renombran los conjuntos de estados a un nro de estado nuevo

Ejemplo

TT	a	b	ε
0-	{3}	{0,2}	{4}
1	-	{2}	-
2	{3}	-	-
3	{0,2}	{4}	{1}
4+	-	{2}	-

Clausura-ε({0}) = **{0,4}**
Hacia({0,4},a) = {3}
Clausura-ε({3}) = **{1,3}**
Hacia({0,4},b) = {0,2}
Clausura-ε({0,2}) = **{0,2,4}**

Hacia({1,3},a) = {0,2}
Hacia({1,3},b) = {2,4}
Clausura-ε({2,4}) = {2,4}

Hacia({0,2,4},a) = {3}
Hacia({0,2,4},b) = {0,2}

Hacia({2,4},a) = {3}
Hacia({2,4},b) = {2}
Clausura-ε({2}) = {2}

Hacia({2},a) = {3}
Hacia({2},b) = -

TT	a	b
{0,4}±	{1,3}	{0,2,4}
{1,3}	{0,2,4}	{2,4}
{0,2,4}+	{1,3}	{0,2,4}
{2,4}+	{1,3}	{2}
{2}	{1,3}	-

TT	a	b
0±	1	2
1	2	3
2+	1	2
3+	1	4
4	1	-

Del AFD a la ER

- Básicamente 3 algoritmos
  - Método de la clausura transitiva, provisto por Stephen Kleene (el mismo del operador *) al demostrar la equivalencia entre AFD y ER
    - Es complicado
    - Usa método propios de grafos, enfoque similar al problema de camino más corto
    - Quedan ER muy largas
  - Método de eliminación de estados
    - Medianamente intuitivo para hacer a mano a partir del diagrama de transición.
    - Hay que tener cuidado de no “olvidar” ningún camino del grafo al hacer la reducción
  - Método algebraico (de Janusz Brzozowski - 1964)
    - Es el que se explica en el libro de la cátedra
    - Genera ER razonables

Eliminación de estado erróneos

- Hay dos tipos de estados erróneos
  - Los que no conducen a un estado final (estados de error)
    - Los absorbentes que queda en si mismos
    - Los no finales que tienen todas las columnas como indefinidos
    - Los cíclicos (de 2 a 3 a 5 a 2 pero ninguno es final)
  - Los estados inalcanzables, es decir aquellos a los que no puedo llegar desde el estado inicial (típicamente no figuran en ninguna columna de la tabla de transición (eliminarlos recursivamente)

Ejemplo

Erróneos
Inalcazables

TT	a	b
0-	1	2
1	1	4
2	4	5
3	4	6
4+	5	4
5	5	5
6	2	0
7	-	-
8	5	-

TT	a	b
0-	1	2
1	1	4
2	4	-
4+	-	4
6	2	0

TT	a	b
0-	1	2
1	1	4
2	4	-
4+	-	4

Armado de las ecuaciones

- Se plantea una ecuación por cada fila
- El lado izquierdo es el estado de la fila
- El lado derecho se forma con la unión para cada columna del carácter que realiza la transición seguido del estado al que pasa.
  - Error habitual: poner estado-letra, o sea, al revés
- Si el estado es un estado final se agrega además ε
- El objetivo es despejar la ecuación del estado inicial

Reducciones

- Cuando hay ciclos o bucles en la ecuación vemos que el estado que figura a izquierda aparece también a derecha.
- Este tipo de ecuaciones se denominan recursivas, la forma general de estas ecuaciones es:
  - $e = \alpha e + \beta$ (1)
  - Donde e es el estado considerado, α es un ER y β es una expresión que puede estar formada por caracteres del alfabeto terminal, estados y ε
- Se resuelve (según por la regla o lema de Arden):
  - $e = \alpha^*\beta$ (2)
- Notar que si reemplazo (2) en (1)
  - $e = \alpha(\alpha^*\beta) + \beta = \alpha\alpha^*\beta + \beta = (\alpha^*\alpha + \varepsilon)\ \beta = \alpha^*\beta$

Ejemplo

TT	a	b
0-	1	2
1	1	4
2	4	-
4+	-	4

Ecuaciones

0 = a1 + b2

1 = a1 + b4

2 = a4

4 = b4 + ε

Desarrollo

4 = b*.ε ⇒ **4 = b***
2 = a4 ⇒ **2 = ab***
1 = a1 + bb*
1 = a*bb*

0 = a1 + b2 = **aa*bb* + bab***

Otro Ejemplo

TT	a	b
0-	{1}	-
1+	-	{2}
2	{2}	{1,2}

Ecuaciones

0 = a1

1 = b2 + ε

2 = a2 + b1 + b2

Desarrollo

2 = (a+b)2 + b1
2 = (a+b)*b1
1 = b2 + ε = b(a+b)*b1 + ε = (b(a+b)*b)*ε
1 = (b(a+b)*b)*
0 = a1 = **a(b(a+b)*b)***

Licencia