Clase Teoría de Lenguajes: Pasaje de $AFND - \lambda$ a AFD y Minimización

En lo sucesivo utilizaremos las siguientes abreviaturas:

- AFD para referirnos a un autómata finito determinísitico
- AFND para referirnos a un autómata finito no determinísitico, sin transiciones lambda
- AFND–λ para referirnos a un autómata finito determinísitico, con transiciones lambda

Motivación

En clases anteriores hablamos de que los AFND–λ y los AFD tienen el mismo poder expresivo, esto es, que el conjunto de los lenguajes aceptados por los AFND–λ es el mismo que el reconocido por los AFD. Queremos entonces, un algoritmo que dado un AFND nos dé un AFD que acepte el mismo lenguaje.

Idea

Cuando definimos el lenguaje aceptado por un AFND–λ dijimos que eran todas las cadenas para las que existía un camino a un estado final.
La propuesta para lograr determinismo es mirar todos los caminos a la vez, es decir, si tengo distintas opciones de transición, voy a mirarlas todas (por eso ahora queremos conjuntos de estados). Así, la configuración instantánea es estar en un conjunto de estados a la vez, y cada transición nos lleva de un conjunto de estados a otro.
Consumir una cadena nos llevará ahora determinísticamente a un conjunto de estados, que serán los estados a los que podíamos llegar de manera no determinística en el autómata original. Por esto, sólo aceptamos las cadenas que nos lleven a conjuntos de estados que tengan alguno final.
Si pensamos el autómata así, ya no tenemos indeterminismo, porque el cambio de configuración para un símbolo del alfabeto, es único.

Veremos dos algoritmos basados en esta idea; el primero nos permitirá obtener un AFD a partir de un AFND, mientras que el segundo generaliza al primero permitiendo obtener un AFD a partir de un AFND–λ.

Desarrollo

Observaciones y aclaraciones:

- Construiremos un *nuevo* autómata, cuyos estados se asocien a un conjunto de estados del autómata no determinístico de entrada. Notar que el AFND–λ de entrada y el AFD de salida NO tienen los mismos estados, los estados del AFD que devolverá el algoritmo serán conjuntos de estados del AFND–λ, por esto podríamos tener exponencialmente mayor cantidad de estados en el autómata de salida (a lo sumo tantos como conjuntos pueda formar, $2^{|Q|}$ si Q son los estados de AFND–λ)
- Si en el automáta no determinísitco tuviera por ejemplo 4 posibles transiciones: $p \xrightarrow{a} q$, $p \xrightarrow{a} r$, $p \xrightarrow{a} s$, $p \xrightarrow{a} t$, en el autómata de salida tendré una única transición $\{p\} \xrightarrow{a} \{q, r, s, t\}$
- Si en el autómata de salida tuviera una transición $\{p, q\} \xrightarrow{a} \{r\}$, esto implica que en el autómata de entrada tenía $p \xrightarrow{a} r$ o bien $q \xrightarrow{a} r$ o bien ambas.

AFND a AFD

Vimos en la teórica que dado un autómata finito no determinístico $N = \langle Q_N, \Sigma, \delta_N, q_0, F_N \rangle$ podemos construir un autómata finito determinístico $D = \langle Q_D, \Sigma, \delta_D, \{q_0\}, F_D \rangle$ con

$$
\begin{aligned}
Q_D &= P(Q_N) \\
F_D &= \{\{q_1, \ldots, q_n\} \in Q_D : \{q_1, \ldots, q_n\} \cap F_N \neq \emptyset\} \\
\delta_D(\{q_1, \ldots, q_n\}, a) &= \bigcup_{i=1}^{n} \delta_N(q_i, a)
\end{aligned}
$$

Podemos pensar esto de una forma algorítmica: empezamos el estado $\{q_0\}$, y vamos calculando a qué estados nos podemos mover tomando cada elemento del alfabeto. Hacemos esto con cada nuevo estado hasta que finalmente se agotan las posibilidades. Como mucho, recordemos que el conjunto $P(Q_N)$ va a tener$2^{|Q_N|}$ elementos. Para pensar: ¿Los tendrá todos? (HINT: Considerar L_1 el lenguaje de palabras en el alfabeto $\{a, b\}$ que tienen una a en la penúltima posición, el lenguaje L_2 como antes pero con a en la antepenúltima posición y ahora para k fijo considerar L_k como antes pero con a en la k-ésima posición antes del final de la palabra).

Escribamos el algoritmo. Primero, hagamos una "macro" para nuestra función de transición:

$$Mover(T, a) = \bigcup_{t \in T} \delta(t, a)$$

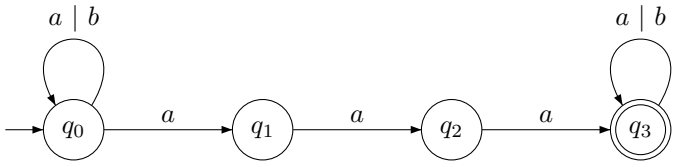
$Mover(T, a)$ nos dice que de un conjunto de estados T y consumiendo a podemos llegar a cualquier estado al que pudiéramos llegar consumiendo a y partiendo de un estado de T en N; más precisamente, llegamos a todos ellos, por eso consideramos la unión de conjuntos.

Teniendo definida la función que nos permite cambiar de un conjunto de estados a otro, podemos entonces definir el algoritmo que dado un AFND–λ nos devuelve un AFD, y que funciona haciendo cambios de configuración, partiendo del estado que solo contiene al estado inicial.

Sea $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ un AFND, construyamos $M' = \langle Q', \Sigma, \delta', q_0', F'\rangle$ tal que $L(M) = L(M')$

1. Defino $\{q_0\}$ el estado inicial de M'
2. Inicializo $Q' := \{\{q_0\}\}$ donde Q' es marcable, y el estado inicial está sin marcar
3. Mientras exista $T \in Q'$ sin marcar:
   a) Marcar T.
   b) Para cada $a \in \Sigma$
      1) Hacer $U \leftarrow Mover(T, a)$.
      2) Si $U \notin Q'$ entonces agrego sin marcar U a Q'.
      3) Hacer $\delta'(T, a) = U$
   c) Fin Para.
4. Fin Mientras.
5. Hacer $F' \leftarrow \{X \in Q' | X \cap F \neq \emptyset\}$

Resolvamos un ejercicio como ejemplo.
Sea M el AFND que reconoce cadenas de $\{a, b\}$ que contengan aaa:
$M = \langle\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\}\rangle$



Se pide calcular un autómata finito determinístico $M' = \langle Q', \Sigma, \delta', q_0', F'\rangle$ que reconozca el mismo lenguaje que M.

Vamos a calcular δ' al mismo tiempo que generamos los nuevos estados. El primer estado es $\{q_0\}$

δ'	a	b
$\{q_0\}$		

Ahora por cada símbolo del alfabeto, calculamos Mover de $\{q_0\}$:

$Mover(\{q_0\}, a) = \bigcup_{t\in\{q_0\}} \delta(t, a) = \{q_0, q_1\}$

$Mover(\{q_0\}, b) = \bigcup_{t \in \{q_0\}} \delta(t, a) = \{q_0\}$

Notar que obtuvimos un nuevo estado $\{q_0, q_1\}$

δ'	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		

Veamos un paso más con $\{q_0, q_1\}$. De nuevo por cada símbolo del alfabeto, calculamos Mover de $\{q_0, q_1\}$:

$$Mover(\{q_0, q_1\}, a) = \bigcup_{t \in \{q_0, q_1\}} \delta(t, a) = \delta(\{q_0\}, a) \bigcup \delta(\{q_1\}, a) = \{q_0, q_1, q_2\}$$

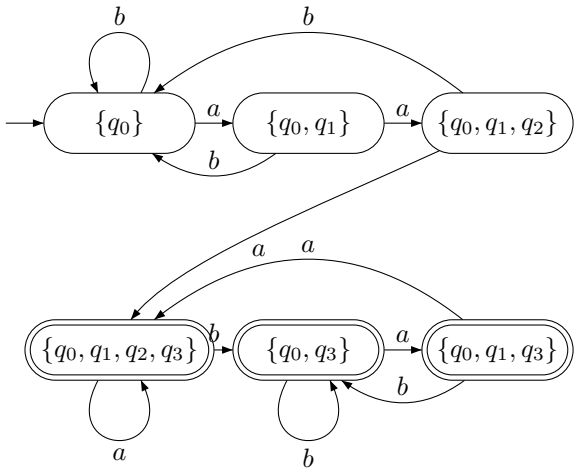
$$Mover(\{q_0, q_1\}, b) = \bigcup_{t \in \{q_0, q_1\}} \delta(t, a) = \delta(\{q_0\}, b) \bigcup \delta(\{q_1\}, b) = \{q_0\}$$

Procedemos de manera similar hasta que no queden estados a los que calcularle la función Mover.

δ'	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_3\}$
$\{q_0, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_3\}$

Entonces ya tenemos calculada δ' y ya sabemos cuál es nuestro conjunto de estados, y cuál es nuestro estado inicial. solamente nos falta calcular F', que será, como dijimos antes, el conjunto de los estados que contengan estados finales de M, ya que estarán representando los conjuntos de movimientos que terminaron en al menos algún estado final.

Luego, nuestro AFD es $M = \langle \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_1, q_2\}, \{q_0, q_1, q_2, q_3\}, \{q_0, q_3\}, \{q_0, q_1, q_3\}\}, \{a, b\}, \delta', \{q_0\}, \{q_0, q_1, q_2, q_3\}, \{q_0, q_3\}, \{q_0, q_1, q_3\}\rangle$ con δ' como sigue:



Es una buena práctica en este punto verificar que el autómata determinizado (el resultante determinístico) reconoce el mismo lenguaje que el original. Como pequeños tests, se reconocen las cadenas *ababaaaabaaa*, *aaa*, *baaa* y *baaab*, mientras que se rechazan λ, *a*, *aa*, *ababa* y *bbbb*.

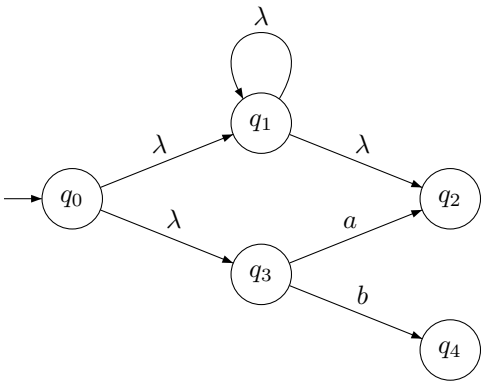
AFND–λ a AFD

Para extender la idea anterior, pensemos que ahora vamos a hacer lo mismo sólo que a cada paso vamos a contemplar también los estados a los que podremos llegar usando sólo transiciones λ, tanto "al principio" como al final de cada transición. Definamos formalmente esta noción:

Si Q son los estados del AFND–λ de entrada, sea $Clausura_\lambda : \mathbf{P}(Q) \to \mathbf{P}(Q)$:

$$Clausura_\lambda(K) = \{x \in Q | \exists q \in K \wedge (q, \lambda) \overset{*}{\vdash} (x, \lambda)\}$$

Veamos algunos ejemplos:



$Clausura_\lambda(\{q_0\}) = \{q_0, q_1, q_2, q_3\}$
$Clausura_\lambda(\{q_1\}) = \{q_1, q_2\}$
$Clausura_\lambda(\{q_2\}) = \{q_2\}$
$Clausura_\lambda(\{q_3\}) = \{q_3\}$
$Clausura_\lambda(\{q_4\}) = \{q_4\}$

Primero redefinamos Mover() para que contemple transiciones λ:
$Mover : \mathbf{P}(Q) \times \Sigma \to \mathbf{P}(Q)$:

$$Mover(T, a) = Clausura_\lambda(\bigcup_{t \in T} \delta(t, a))$$

Ahora redefinamos el algoritmo para que funcione también con transiciones lambda:

1. Hacer $Clausura_\lambda(\{q_0\})$ el estado inicial de M' (q'_0).
2. Hacer $Q' = \{Clausura_\lambda(\{q_0\})\}$ donde Q' es marcable y $Clausura_\lambda(\{q_0\}$ está sin marcar.

3. Mientras exista $T \in Q'$ sin marcar:
   a) Marcar T.
   b) Para cada $a \in \Sigma$
      1) Hacer $U \leftarrow Mover(T, a)$.
      2) Si $U \notin Q'$ entonces agrego sin marcar U a Q'.
      3) Hacer $\delta'(T, a) = U$
   c) Fin Para.
4. Fin Mientras.
5. Hacer $F' \leftarrow \{X \in Q' | X \cap F \neq \emptyset\}$

Minimización de AFD

Motivación

Dado un AFD, queremos obtener un autómata de estados mínimos que acepte el mismo lenguaje. En la clase teórica vemos que ese autómata es único salvo isomorfismos.

Idea

Como idea para obtener el autómata de estados mínimos vamos a eliminar los estados que no aporten información extra a nuestro autómata; esto es, si tenemos dos estados a partir de los cuales podemos generar el mismo lenguaje (o aceptar las mismas cadenas), vamos a eliminar uno de ellos. La demostración de que esta idea nos lleva a obtener el autómata con la mínima cantidad de estados, escapa a la clase práctica.

Vamos a recordar entonces el concepto de estados indistinguibles, y sus propiedades que usaremos para construir el algoritmo de minimización:

Indistinguibilidad

Recordemos el concepto de **lenguaje generado a partir de un estado** q_i:
$L_i = \{\alpha | \exists q_f \in F(q_i, \alpha) \vdash (q_f, \lambda)\}$.
Esto no es otra cosa que pensar al autómata como si q_i fuera estado inicial y ver qué lenguaje reconoce ahora el autómata.

Estados Indistinguibles: Sea el AFD M, y $q_p, q_r \in Q$ dos estados. Decimos que q_p es indistinguible de q_r (y lo notamos $q_p \equiv q_r$) si $L_p = L_r$ (el lenguaje generado a partir de q_p es el mismo que el generado a partir de q_r).
Otra manera de verlo es pensando en los cambios de configuración, entonces tendríamos
$\forall \alpha \in \Sigma^*(\exists q_f \in F \mid (q_p, \alpha) \overset{*}{\vdash} (q_f, \lambda) \iff \exists q'_f \in F \mid (q_r, \alpha) \overset{*}{\vdash} (q'_f, \lambda)$
es decir, si para todas las cadenas del alfabeto, los sucesivos cambios de configuracion partiendo de un estado, o bien llegan a un estado final en los dos casos, o bien no llegan en ninguno de los dos.

Indistinguibilidad de orden k

Es similar al concepto de indistinguibilidad, pero para cadenas de longitud hasta k. Sea el AFD M, y $q, r \in Q$ dos estados. Decimos que q es indistinguible de orden k de r (y lo notamos $q \equiv_k r$) si

$\forall \alpha \in \Sigma^* \wedge |\alpha| \leq k \ (\exists q_f \in F \mid (q, \alpha) \overset{*}{\vdash} (q_f, \lambda) \iff \exists q'_f \in F \mid (r, \alpha) \overset{*}{\vdash} (q'_f, \lambda)$

Más informalmente, $q \equiv_k r$ si mirando el lenguaje de q quedándome sólo con palabras de longitud hasta k, y mirando el lenguaje de r quedándome sólo con palabras de longitud hasta k, entonces esos dos conjuntos coinciden.

Propiedades

La relación de indistinguibilidad tiene las siguientes propiedades probadas en clase teórica, que vamos a utilizar para construir el algoritmo de minimización:

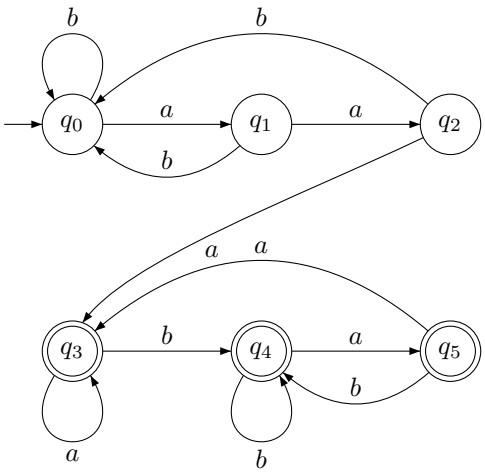
1. $\equiv_k$ es relación de equivalencia.
2. $\equiv_{k+1} \subseteq \equiv_k$
3. $Q/\equiv_0 = \{Q - F, F\}$
4. $p \equiv_{k+1} r \iff (p \equiv_k r \wedge \forall a \in \Sigma, (\delta(p, a) \equiv_k \delta(r, a))$
5. $(\equiv_p) = (\equiv_{p+1}) \Rightarrow \forall k > 0 \ (\equiv_p) = (\equiv_{p+k})$

Desarrollo

Con estas propiedades, podemos construir nuestro algoritmo. Sabemos que la relación $\equiv_k$ es relación de equivalencia y por lo tanto nos define una partición del conjunto de estados, para cada k. Lo que buscamos nosotros es construir un autómata equivalente al ingresado, pero que tenga todos sus estados distinguibles. Por la segunda propiedad sabemos que podemos ir haciendo un refinamiento progresivo de nuestra partición, y por la última sabemos que vamos a terminar. Por la cuarta propiedad sabemos cómo pasar de la indistinguibilidad de orden k a la de orden $k + 1$ y por la tercera sabemos como empezar.

Ejemplo

Veamos con el siguiente ejemplo el desarrollo del algoritmo:
Sea el AFD $M = \langle \{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \delta, q_0, \{q_3, q_4, q_5\} \rangle$



A continuación constriumos una tabla donde vamos a clasificar los estados en clases de equivalencia de orden 0 en adelante:

	$\equiv_0$	a	b
q_0	NF		
q_1	NF		
q_2	NF		
q_3	F		
q_4	F		
q_5	F		

Los estados no finales pertenecen a la clase NF y los finales a la clase F. Calculamos luego, a que clase pertenecen los estados a los que nos movemos por todos los símbolos del alfabeto; esto es, si tenemos una transición $p \xrightarrow{a} q$, completamos la entrada $[p,a]$ con la clase de equivalencia correspondiente a q en esa iteración.
Por ejemplo, consideramos $q_4 \xrightarrow{a} q_5$, como la clase de q_5 es F, en la entrada $[q_4,a]$ ponemos F. Otro ejemplo, consideramos $q_2 \xrightarrow{a} q_3$, como la clase de q_3 es F, en la entrada $[q_2,a]$ ponemos F. Otro ejemplo más, consideramos $q_1 \xrightarrow{b} q_0$, como la clase de q_0 es NF, en la entrada $[q_1,b]$ ponemos NF.

	$\equiv_0$	a	b
q_0	NF	NF	NF
q_1	NF	NF	NF
q_2	NF	F	NF
q_3	F	F	F
q_4	F	F	F
q_5	F	F	F

Ahora podemos utilizar la cuarta propiedad para calcular la indistinguibilidad de orden 1. Esto es, si los estados pertenecian a la misma clase y al moverse por todos los símbolos van a parar a las mismas clases, entonces ambos continuarán en la misma clase, si no, serán clases distintas.
En este caso, basta comparar las 3-uplas. Las 3-uplas que coincidan van a parar a la misma clase.

Para facilitar el seguimiento se recomenda usar símbolos nuevos para nombrar clases de equivalencia en cada iteración.

	$\equiv_0$	a	b	$\equiv_1$	a	b
q_0	NF	NF	NF	♡		
q_1	NF	NF	NF	♡		
q_2	NF	F	NF	♣		
q_3	F	F	F	♢		
q_4	F	F	F	♢		
q_5	F	F	F	♢		

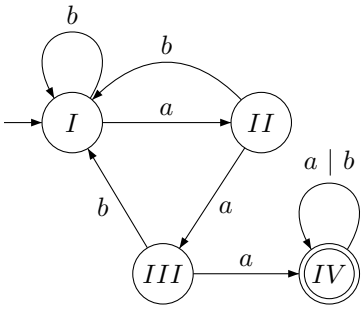
Notar que en esta iteración pasamos de dos a tres clases de equivalencia. De manera similar continuamos construyendo nuestra tabla hasta que encontremos dos indistinguibilidades consecutivas que definan la misma partición, ya que sabemos, por la quinta propiedad, que todas las demás serán iguales.
Una forma rápida de ver esto es que el número de clases de equivalencia no aumente (que sigamos usando la misma cantidad de chirimbolos para representar clases). Seguimos completando la tabla...
Por ejemplo, consideramos $q_4 \xrightarrow{a} q_5$, como la clase de q_5 es ♢, en la entrada $[q_4,a]$ ponemos ♢.
Otro ejemplo, consideramos $q_2 \xrightarrow{a} q_3$, como la clase de q_3 es ♢, en la entrada $[q_2,a]$ ponemos ♢.
Otro ejemplo más, consideramos $q_1 \xrightarrow{b} q_0$, como la clase de q_0 es ♡, en la entrada $[q_1,b]$ ponemos ♡.

	$\equiv_0$	a	b	$\equiv_1$	a	b	$\equiv_2$	a	b	$\equiv_3$
q_0	NF	NF	NF	♡	♡	♡	✠	⋆	✠	I
q_1	NF	NF	NF	♡	♣	♡	⋆	◎	✠	II
q_2	F	F	NF	♣	♢	♡	◎	⊡	✠	III
q_3	F	F	F	♢	♢	♢	⊡	⊡	⊡	IV
q_4	F	F	F	♢	♢	♢	⊡	⊡	⊡	IV
q_5	F	F	F	♢	♢	♢	⊡	⊡	⊡	IV

Notar que como en las iteraciones 2 y 3 obtuvimos la misma cantidad de clases (4 clases en este ejemplo), el algoritmo terminó.

Chequeos rápidos: Si en alguna iteración decreciera el número de clases de equivalencia, revisar las cuentas :) Hubo algún error.
Notar que si dos estados son distinguibles de orden k (hay alguna palabra de longitud menor o igual a k que *uno reconoce y el otro no*), entonces también serán indistinguibles de orden $k+1$.

Ahora podemos construir el AFD M' mínimo. El conjunto de estados serán las clases de equivalencia, el estado inicial la clase que contenga a q_0, los estados finales aquellas clases formadas por estados finales en M y δ la función calculada en la tabla en el ultimo paso. Entonces $M = \langle \{I, II, III, IV\}, \{a, b\}, \delta', I, \{IV\} \rangle$



Como antes, es buena práctica verificar que el autómata resultante acepta el mismo lenguaje que el autómata original, al menos con un conjunto representativo de cadenas.
Es importante notar que este proceso es para minimizar AFD's y que no usamos el concepto de mínimo en AFND–λ.