Algoritmo de minimización

Autómatas y Lenguajes Formales

13 de marzo de 2023

Automata cociente

La idea del algoritmo de minimalización es colapsar estados, dos o más estados se reducen a uno. Eso no se puede hacer de forma deliberada, debe haber un método formal y que nos asegure que no estamos perdiendo transiciones.

Para ello debemos definir algunas cosas. Para empezar podemos notar que no es deseable colapsar un estado de aceptación con uno de rechazo, son radicalmente distintos. Cualesquiera dos o más estados que se colapsen deben mantener el determinismo (recuerden estamos minimizando un autómata finito determinista).

Definimos una relación de equivalencia entre dos estados, que como bien sospechan, serán los dos a colapsar.

Definimos una relación de equivalencia entre q y r como:

$$q \approx r \iff \{\forall \alpha . \delta^*(q, \alpha) \in F \iff \delta^*(r, \alpha) \in F\},$$

siendo $q, r \in Q$ y $\alpha \in \Sigma^*$. Esta relación de equivalencia cumple con ser reflexiva, simétrica y transitiva, y parte el conjunto en subconjuntos disconexos llamados clases de equivalencia

$$[p] \stackrel{def}{=} \{q | q \approx p\}$$

Definimos un autómata cociente. Partimos del autómata $A = (Q, \Sigma, \delta, s, F)$ y definimos al autómata cociente como $A/\approx = (Q_\approx, \Sigma, \delta_\approx, s_\approx, F_\approx)$:

$$\begin{aligned} Q_\approx &= \{[q] | q \in Q\} \\ \delta_\approx([q], a) &= [\delta(q, a)] \\ s_\approx &= [s] \\ F_\approx &= \{[f] | f \in F\} \end{aligned}$$

.

Hay un estado de $A/\approx$ por cada clase de equivalencia, es decir, todos los estados colapsados están en un estado para el autómata cociente. Sólo restaría ver si la función de transición está bien definida, pero eso pueden consultarlo en el libro[1].

Algoritmo de minimización

Entonces vamos ahora sí a dar un método sistemático para colapsar estados, tan sistemático que hasta numerado está (recuerden que trabajamos con autómatas finitos

deterministas sin estados inalcanzables, si quieren reducir uno no determinista primero deben pasarlo a determinista):

1. Escribe una tabla vacía de todos los pares $\{p, q\}$.
2. Marca $\{p, q\}$ si $p \in F$ y $q\neg \in F$ o al revés.
3. Paso recursivo, **repítelo hasta que ya no haya nada más por marcar**: si existe un par sin marcar $\{p, q\}$ tal que su par de transiciones $\{\delta(p, a), \delta(q, a)\}$ sí está marcado para algún símbolo del alfabeto, $a \in \Sigma$, entonces se marca el par $\{p, q\}$.
4. Si ya no hay más que marcar por el *loop* pasado, $p \approx q$ si $\{p, q\}$ no está marcado.

Vemos un ejemplo:

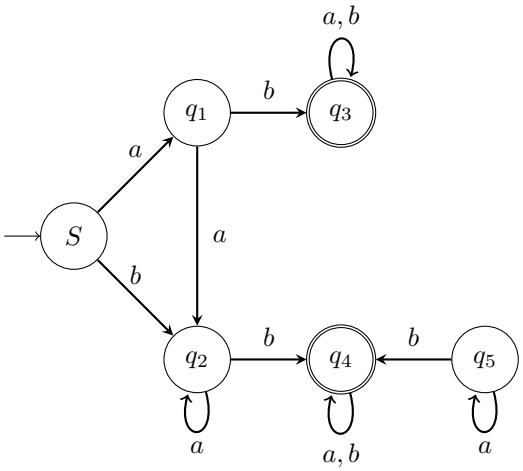

Figura 1: Autómata determinista con estados inaccesibles, por reducir.

Este autómata aún tiene un estado inaccesible, es decir, ninguna flecha entra en él, no hay transiciones hacia él, por lo tanto es inaccesible, no podemos llegar a ese estado de ninguna manera. Si lo borramos no perdemos nada pues no hay manera de llegar a él.

Ya nos deshicimos del estado inaccesible, ahora está todo para aplicar el algoritmo. Vamos paso a paso, armemos la tabla:

Estados	S	q_1	q_2	q_3	q_4
S					
q_1					
q_2					
q_3					
q_4					

Quizá este método no es el mejor visualmente, pero es más directo de lo que se escribe en el algoritmo. Noten que las parejas $\{q_i, q_i\}$ siempre se pueden colapsar, es el mismo estado colapsado a si mismo, la versión del Kozen se ahorra estos cuadros.

Marcamos todos las parejas que incluyen un estado final, los que incluyan a q_3 y q_4.

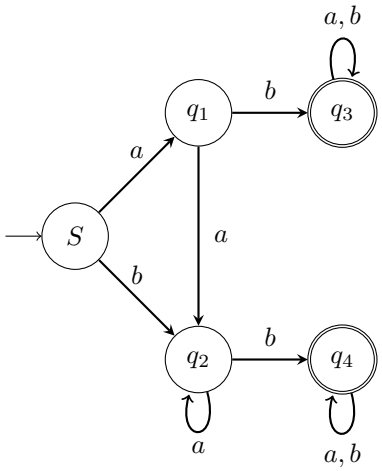

Figura 2: Autómata determinista sin estados inaccesibles, aún hay más por reducir.

Estados	S	q_1	q_2	q_3	q_4
S				✓	✓
q_1				✓	✓
q_2				✓	✓
q_3	✓	✓	✓		
q_4	✓	✓	✓		

Noten que sólo se marca si uno de los dos es final, no ambos (esos podría ser que sí colapsen, aún no lo sabemos). Vamos al siguiente paso: vemos por pares sin marcar para los valores a o b hacia donde van.

Empezamos con $\{S, q_1\}$, con a $S \xrightarrow{a} q_1$ y $q_1 \xrightarrow{a} q_2$, entonces $\{s, q_1\} \xrightarrow{a} \{q_1, q_2\}$, que no está marcado, seguimos sin marcarlo.

¿Qué pasa con b? $\{s, q_1\} \xrightarrow{b} \{q_2, q_3\}$, que sí está marcado, lo marcamos entonces (y también el par simétrico $\{q_1, s\}$).

Ahora vemos que pasa con el par $\{S, q_2\}$:

$\{S, q_2\} \xrightarrow{a} \{q_1, q_2\}$ sin marcar

$\{S, q_2\} \xrightarrow{b} \{q_2, q_4\}$ se marca ✓, así como el simétrico

Estados	S	q_1	q_2	q_3	q_4
S		✓	✓	✓	✓
q_1	✓			✓	✓
q_2	✓			✓	✓
q_3	✓	✓	✓		
q_4	✓	✓	✓		

El siguiente sería $\{S, q_3\}$, pero ya está marcado, al igual que $\{S, q_4\}$. Saltemos

hasta el $\{q_1, q_2\}$

$$\{q_1, q_2\} \xrightarrow{a} \{q_2, q_2\} \text{ no se marca}$$
$$\{q_1, q_2\} \xrightarrow{b} \{q_3, q_4\} \text{ no se marca}$$

Y ya no podemos hacer más, si volvemos a barrer la tabla llegamos a los mismos estados sin marcar.

Lo que quiere decir que q_1 y q_2 son colapsables, así como q_3 y q_4. Entonces el autómata reducido es como se ve en la imagen 3. Es casi autómatico pasar a este autómata, las trancisiones que iban al estado q_1 o q_2 ahora van al estado q_{1-2}, las que iban al estado q_3 o q_4 ahora van al estado q_{3-4}.

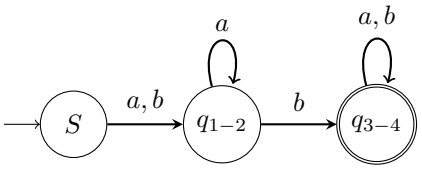

Figura 3: Autómata determinista ya reducido al mínimo.

Veamos otro ejemplo ahora a partir de una tabla.

Ejemplo: Minimaliza el siguiente autómata

Estado	a	b
$\rightarrow S$	Q_1	Q_5
Q_1	S	Q_5
Q_2	Q_6	S
Q_3	Q_7	Q_1
Q_4	S	Q_6
Q_5^*	Q_7	Q_2
Q_6^*	S	Q_3
Q_7^*	S	Q_2

Espero sea claro que S es el estado inicial y Q_5, Q_6 y Q_7, los estados marcados con asterisco, son estados finales. De analizar la tabla podemos encontrar los estados inaccesibles, aquellos estados que no aparezcan del lado derecho de la tabla. Revisando podemos ver que el estado Q_4 no aparece en ninguna transición desde otro estado. Podemos eliminarlo.

Estado	a	b
S	Q_1	Q_5
Q_1	S	Q_5
Q_2	Q_6	S
Q_3	Q_7	Q_1
Q_5^*	Q_7	Q_2
Q_6^*	S	Q_3
Q_7^*	S	Q_2

S						
	Q_1					
		Q_2				
			Q_3			
				Q_5		
					Q_6	
						Q_7

Cuadro 1: Tabla para la minimalización de estados del autómata.

Ya sin estados inaccesibles armamos la tabla **??**, empezando vacía y marcamos, como lo dice el algoritmo, las parejas {*estado final, no estado final*} o viceversa.

Una vez marcadas las parejas {*estado final, no estado final*}, revisamos las posibles transiciones de cada estado de la pareja para cada símbolo del alfabeto.

S						
	Q_1					
		Q_2				
			Q_3			
✓	✓	✓	✓	Q_5		
✓	✓	✓	✓		Q_6	
✓	✓	✓	✓			Q_7

Cuadro 2: Tabla para la minimalización de estados del autómata.

Se pueden ver las siguientes posibilidades:

$\{S, Q_1\} \xrightarrow{a} \{Q_1, S\}$, que no está marcado

$\{S, Q_1\} \xrightarrow{b} \{Q_5, Q_5\}$, que no puede estar marcado

$\{S, Q_2\} \xrightarrow{a} \{Q_1, Q_6\}$, que sí está marcado ✓

$\{S, Q_3\} \xrightarrow{a} \{Q_1, Q_7\}$, que sí está marcado ✓

$\{Q_1, Q_2\} \xrightarrow{a} \{S, Q_6\}$, que sí está marcado ✓

$\{Q_1, Q_3\} \xrightarrow{a} \{S, Q_7\}$, que sí está marcado ✓

$\{Q_2, Q_3\} \xrightarrow{a} \{Q_6, Q_7\}$, que no está marcado

$\{Q_2, Q_3\} \xrightarrow{b} \{S, Q_1\}$, que no está marcado

$\{Q_5, Q_6\} \xrightarrow{a} \{Q_7, S\}$, que sí está marcado ✓

$\{Q_5, Q_7\} \xrightarrow{a} \{Q_7, S\}$, que sí está marcado ✓

$\{Q_6, Q_7\} \xrightarrow{a} \{S, S\}$, que no puede estar marcado

$\{Q_6, Q_7\} \xrightarrow{b} \{Q_3, Q_2\}$, que no está marcado

Una vez con viendo las transiciones podemos seguir repitiendo el paso, pero vemos

que las parejas sin marcar siguen si poderse marcar. Para ese caso la tabla queda como se ve en **??**

S						
	Q_1					
✓	✓	Q_2				
✓	✓		Q_3			
✓	✓	✓	✓	Q_5		
✓	✓	✓	✓	✓	Q_6	
✓	✓	✓	✓	✓		Q_7

Cuadro 3: Tabla para la minimalización de estados del autómata.

Lo que quiere decir, que aunque demos más vueltas ya no podemos marcar más y por lo tanto las equivalencias son $S \equiv Q_1$, $Q_2 \equiv Q_3$ y $Q_6 \equiv Q_7$, estos estados pueden colapsarse. Les muestro la tabla y ya les toca a ustedes hacer el autómata:

Estado	a	b
$\{S, Q_1\}$	$\{S, Q_1\}$	Q_5
$\{Q_2, Q_3\}$	$\{Q_6, Q_7\}$	$\{S, Q_1\}$
Q_5^*	$\{Q_6, Q_7\}$	$\{Q_2, Q_3\}$
$\{Q_6, Q_7\}^*$	S	$\{Q_2, Q_3\}$

Referencias

[1] Kozen, Dexter C. “Automata and Computability” Springer (1997)

[2] Sipser, Michael “Introduction to the Theory of Computation” 2a ed., Thomson Course Tecnology (2006)

[3] Lewis, Harry R. y Papadimitriou, Christos H. “Elements of the Theory of Computation”2a ed., Prentice-Hall Inc. (1998).