Next Up Previous Contents

Siguiente: Abreviaciones para el uso **Subir:** Expresiones regulares **Anterior:** Síntaxis y semántica
Índice General

Equivalencia entre autómatas finitos y expresiones regulares

La semántica de una expresión regular define un lenguaje.

Dado una expresión regular α (sobre un alfabeto Σ). ¿Qué tiene que ver el lenguaje $L(\alpha)$ con un lenguaje $L(M)$ aceptado por un autómata finito M?

Veremos: para cada expresión regular α existe un autómata no-determinista con transiciones ϵ M, o sea un AFND-ϵ, que acepta el mismo lenguaje (es decir, $L(\alpha) = L(M)$).

Ya sabemos: entonces también existe un autómata finito determinista, o sea un AFD, aceptando el mismo lenguaje.

De hecho, comprobaremos algo más: para cada α sobre Σ existe un AFND-ϵ $M = (\Sigma, Q, \delta, q_0, F)$ con $L(\alpha) = L(M)$ y

- no existe ninguna transición hacia el estado inicial, es decir

$$\forall q \in Q, \sigma \in \Sigma : q_0 \notin \delta(q, \sigma) \cup \delta(q, \epsilon)$$

- M tiene exactamente un estado final del cual no sale ninguna transición, es decir,

$$|F| = 1 \text{ y } \forall \sigma \in \Sigma, f \in F : \delta(f, \sigma) \cup \delta(f, \epsilon) = \emptyset$$

La comprobación sigue la definición inductiva de la expresión regular, lo describimos solamente con los grafos de los autómatas. Entonces, sean α, β, y γ expresiones regulares sobre algún alfabeto Σ.

1. 1. $\alpha = \emptyset$

   regexprafnde1

2. $\alpha = \epsilon$

   regexprafnde2

3. $\alpha = a$

   regexprafnde3

2. 1. $\alpha = \beta\gamma$

      regexprafnde4

   2. $\alpha = (\beta + \gamma)$

      regexprafnde5

3. 1. $\alpha = (\beta)$

      regexprafnde6

   2. $\alpha = (\beta)^*$

      regexprafnde7

Ejemplo: construimos el AFND-ϵ para $\alpha = (((a.b)^* + a) + b.b)$

regexprafndeej

La otra dirección, es decir, comprobando que para cada autómata finito existe una expresión regular que describe el mismo lenguaje, nos costará un poco más de trabajo.

Sea $M = (\Sigma, Q, \delta, q_0, F)$ un AFD (sabemos que cualquier AFND o AFND-ϵ se puede convertir en un AFD).

Describimos un algoritmo que sucesivamente construye la clausura transitiva del autómata dado y así construye finalmente--como atributos de las aristas entre q_0 y un nuevo estado f--la expresión regular.

Por eso permitimos que se pueden escribir expresiones regulares a las aristas de un autómata, es decir, para $\delta(p, \sigma) = q$ escribimos (p, σ, q) (pues, la arista del estado p al estado q con atributo a), o teniendo expresiones regulares (p, α, q) (pues, una arista de p a q con atributo α), o con dibujo:

aristaexpr

1. añadimos un nuevo estado f y conectamos todos los estados en F con transiciones ϵ a f, es decir, cambiamos M por $M' = (Q \cup \{f\}, \Sigma, \delta', q_0, \{f\})$ donde $\delta' = \delta$ para estados en Q y además $\forall q \in F : \delta'(q, \epsilon) = f$. Así no hemos cambiado el lenguaje aceptado por M. (Pero siguimos escribiendo abajo simplemente M, δ, y Q para simplificar la notación.)
2. para todos los estados $q \neq q_0$ y $q \neq f$
   1. para cada pareja de aristas (p, β, q) y (q, γ, r) y arista reflexiva (q, φ, q) (nota, puede ser $p = r$)

      añade arista $(p, \beta\varphi^*\gamma, r)$
   2. elimina q con todas sus aristas adyacentes

      pqr

   3. agrupa las aristas construidas $(p, \alpha_1, r), \ldots, (p, \alpha_k, r)$ escribiendo

      $(p, \alpha_1 + \ldots + \alpha_k, r)$
3. cuando termina el proceso, es decir, solamente existen aristas entre q_0 y f, precisamente (q_0, α, q_0) y/o (q_0, β, f), la expresión regular final es $\alpha^*\beta$.

(Observa: si $q_0 \in F$ entonces existe una arista con ϵ entre q_0 y f, por eso, $\epsilon \in L(\beta)$, y entonces no hay que considerar un caso especial para contemplar lazos reflexivos en q_0 porque $\alpha^*\beta + \alpha^* = \alpha^*\beta$.)

Ejemplo:

XXX

Una comprobación formal de la corrección del algoritmo es bastante técnica. Principalmente hay que realizar una inducción estructural con propiedades de dichos autómatas extendidos (que tienen expresiones regulares en sus aristas). No lo detallamos aquí, cae en la categoría: lo creemos (en estos momentos).

Como vimos en el ejemplo, hemos construido una expresión regular totalmente diferente a la de partida. Debemos transformar dicha expresión regular sin cambiar el lenguaje que define para conseguir finalmente una expresión regular igual a la de partida. Por eso:

Dos **expresiones regulares** α **y** β **son equivalentes** ($\alpha \equiv \beta$) si definen el mismo lenguaje, es decir, si

$$L(\alpha) = L(\beta).$$

Obviamente hay operaciones con expresiones regulares que mantienen la equivalencia, por ejemplo:

Asociatividad:

$$\begin{aligned} (\alpha + (\beta + \gamma)) &\equiv ((\alpha + \beta) + \gamma) \\ \alpha.(\beta.\gamma) &\equiv (\alpha.\beta).\gamma \end{aligned}$$

Conmutatividad:

$$(\alpha + \beta) \equiv (\beta + \alpha)$$

Elementos neutros:

$$\begin{aligned} (\alpha + \emptyset) &\equiv (\emptyset + \alpha) \\ &\equiv \alpha \\ (\alpha.\epsilon) &\equiv (\epsilon.\alpha) \\ &\equiv \alpha \end{aligned}$$

Eliminación:

$$\begin{aligned} (\alpha.\emptyset) &\equiv (\emptyset.\alpha) \\ &\equiv \emptyset \end{aligned}$$

Distributividad:

$$\begin{aligned} \alpha.(\beta + \gamma) &\equiv (\alpha.\beta + \alpha.\gamma) \\ (\alpha + \beta).\gamma &\equiv (\alpha.\gamma + \beta.\gamma) \end{aligned}$$

Simplificación:

$$\begin{aligned} ((\alpha)^*)^* &\equiv (\alpha)^* \\ (\emptyset)^* &\equiv \emptyset \\ (\epsilon)^* &\equiv \epsilon \end{aligned}$$

Con eso y un poco de ímpetu podemos transformar sucesivamente la expresión regular obtenida para obtener al final la expresión regular que era la base para el autómata finito inicial.

XXX

El problema de comprobar en general si dos expresiones regulares son equivalentes no es nada fácil. Dicho problema cae en la clase de los problemas PSPACE que contiene problemas aún más complejos que los problemas de la clase NP que (a lo mejor) veremos hacia el final del curso (un problema NP es el problema del viajante). Aquí nos basta constatar que un algoritmo determinista que resuelve el problema necesita un tiempo que crece más que exponencial en la longitud de la expresión regular.

Next Up Previous Contents

Siguiente: Abreviaciones para el uso **Subir:** Expresiones regulares **Anterior:** Síntaxis y semántica
Índice General