Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2223 – Teoría de Autómatas y Lenguajes Formales
Segundo semestre de 2024
Profesor: Cristian Riveros
Ayudante: Amaranta Salas

Ayudantia 9

Transductores

Problema 1

Sea Σ y Ω dos alfabetos finitos. Diremos que una función $h : \Sigma^* \to \Omega^*$ respeta la concatenación si para todo par de palabras $u, v \in \Sigma^*$ se cumple que $h(u \cdot v) = h(u) \cdot h(v)$. Demuestre que para toda función $h : \Sigma^* \to \Omega^*$ que respeta la concatenación existe un transductor determinista $\mathcal{T}$ tal que $[\![\mathcal{T}]\!](w) = h(w)$ para toda palabra $w \in \Sigma^*$.

Solución

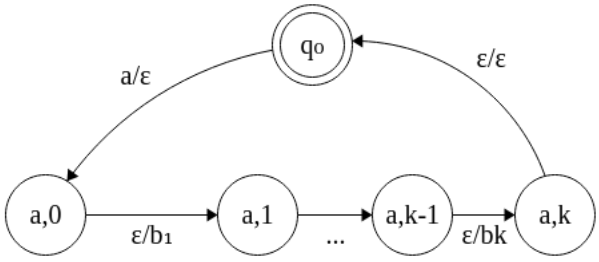
Sabemos que $h : \Sigma^* \to \Omega^*$ es una función que respeta la concatenación. Por lo tanto, para cualquiera dos palabras $u, v \in \Sigma^*$, se tiene:

$$h(u \cdot v) = h(u) \cdot h(v)$$

Luego, podemos ver que para $w = a_1 \dots a_n$,

$$h(w) = h(a_1 \dots a_n) = h(a_1) \dots h(a_n).$$

También es fácil ver que $h(\varepsilon) = \varepsilon$, necesariamente. Por lo tanto, la función h queda definida por el resultado de $h(a) \in \Omega^*$ para cada $a \in \Sigma$. Notar que el resultado de $h(a)$ es una palabra para cada $a \in \Sigma$. Entonces, intuitivamente podemos pensar en un transductor donde para cada a imprime la palabra $h(a) = b_1 \dots b_k$ con $b_i \in \Omega$ para todo $i \leq k$. Para esto, necesitamos un estado q_0 y, para cada $a \in \Sigma$, el transductor realiza la siguiente secuencia de transiciones:



Para definir esta secuencia formalmente, dado un $w = b_1 \dots b_k \in \Omega^*$, definimos $w[i] = b_i$ como la i-ésima letra de la palabra w, indexado desde 1. Luego, más formalmente, podemos definir el transductor $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, q_0, \{q_0\})$, tal que:

- $Q = \{q_0\} \cup \bigcup_{a \in \Sigma} \{(a, i) \mid 0 \leq i \leq |h(a)|\}$
- $\Delta = \bigcup_{a \in \Sigma} \left(\Big\{ \big(q_0, a, \varepsilon, (a, 0)\big) \Big\} \cup \Big\{ \big((a, i-1), \varepsilon, h(a)[i], (a, i)\big) \mid 0 < i \leq |h(a)| \Big\} \cup \Big\{ \big((a, |h(a)|), \varepsilon, \varepsilon, q_0\big) \Big\} \right)$

En primer lugar, el transductor es determinista ya que para todo $(p, a, b, q), (p, a', b', q) \in \Delta$, se tiene que si $a = a'$ entonces $b = b'$. Por otro lado para todo $(p, \varepsilon, a, q), (p, \varepsilon, a', q') \in \Delta$, $a = a'$ y $q = q'$.

En segundo lugar, demostraremos que $\mathcal{T}(w) = h(w)$ para todo $w \in \Sigma^*$. Tomemos $w = a_1 \dots a_n$. El ejecutar el transductor sobre w tendremos una ejecución:

$$\rho : q_0 \xrightarrow{a_1/\varepsilon} (a_1, 0) \xrightarrow{\varepsilon/b_{a_1,1}} (a_1, 1) \xrightarrow{\varepsilon/b_{a_1,2}} (a_1, 2) \xrightarrow{\varepsilon/b_{a_1,3}} \dots \xrightarrow{\varepsilon/b_{a_1,|h(a_1)|}} (a_1, |h(a_1)|) \xrightarrow{a_2/\varepsilon} (a_2, 0) \dots q_0$$

Lo que da como output $b_{a_1,1} b_{a_1,2} b_{a_1,3} \dots b_{a_1,k_{a_1}} b_{a_2,1} \dots b_{a_n,k_{a_n}} = h(a_1)h(a_2) \dots h(a_n) = h(w)$.

Problema 2

Sean Σ, Ω y Γ alfabetos. Recuerde que para dos relaciones $R \subseteq \Sigma^* \times \Omega^*$ y $S \subseteq \Omega^* \times \Gamma^*$ se definen las operaciones R^{-1} y $R \circ S$ como:

- $R^{-1} = \{(u, v) \in \Omega^* \times \Sigma^* \mid (v, u) \in R\}$
- $R \circ S = \{(u, v) \in \Sigma^* \times \Gamma^* \mid \exists w \in \Omega^*. (u, w) \in R \wedge (w, v) \in S\}$

Responda si las siguientes afirmaciones son verdaderas o falsas. Demuestre su respuesta.

1. Si R es una relación racional, entonces R^{-1} es una relación racional.
2. Si R y S son relaciones racionales, entonces $R \circ S$ es una relación racional.

Solución

1. Si R es una relación racional, entonces R^{-1} es una relación racional.

   La respuesta es *Verdadero*.

   Si R es una relación racional, entonces existe un transductor $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ tal que $R = [\![\mathcal{T}]\!]$. Defina $\mathcal{T}^{-1} = (Q, \Omega, \Sigma, \Delta^{-1}, I, F)$ tal que $\Delta^{-1} = \{(p, b, a, q) \mid (p, a, b, q) \in \Delta.\}$.

   Sean $u \in \Sigma^*$ y $v \in \Omega^*$. Demostramos utilizando el principio de inducción fuerte sobre el largo de u y v, que R^{-1} es un relación racional. En particular, $R^{-1} = [\![\mathcal{T}]\!]^{-1}$ y basta demostrar que $(u, v) \in [\![\mathcal{T}]\!]$ si, y solo si, $(v, u) \in [\![\mathcal{T}^{-1}]\!]$.

   Hipótesis de inducción: $(p, u, \epsilon) \vdash^*_{\mathcal{T}} (q, \epsilon, v)$ si, y solo si, $(p, v, \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, u)$.

   - Caso base: Sean $a \in \Sigma$ y $b \in \Omega$. Tenemos dos casos:
     (a) Si $(p, a, \epsilon) \vdash^*_{\mathcal{T}} (q, \epsilon, b)$, entonces $(p, b, \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, a)$ porque en la ejecución de $\mathcal{T}^{-1}$ se utilizarán las transiciones utilizadas por $\mathcal{T}$ en el mismo orden, pero la escritura invertida con la lectura. Por ejemplo, si en un instante $\mathcal{T}$ lee a y escribe b, entonces $\mathcal{T}^{-1}$ leerá b y escribirá a. O si $\mathcal{T}$ lee a sin escribir y escribe b sin leer en algún orden, entonces $\mathcal{T}^{-1}$ leerá b sin escribir y escribirá a sin leer en el mismo orden.

     (b) Si $(p, b, \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, a)$, entonces $(p, a, \epsilon) \vdash^*_{\mathcal{T}} (q, \epsilon, b)$ por un argumento simétrico al anterior.
   - Caso Inductivo: Sean $u, w \in \Sigma^*$, $a \in \Sigma$, $v, w' \in \Omega^*$ y $b \in \Omega$ tal que $u = aw$ y $v = bw'$. Tenemos dos direcciones que demostrar.

     ($\Rightarrow$) Demostraremos que si $(p, u, \epsilon) \vdash^*_{\mathcal{T}} (q, \epsilon, v)$, entonces $(p, v, \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, u)$. Sabemos que $(p, aw, \epsilon) \vdash^*_{\mathcal{T}} (q, \epsilon, bw')$ si, y solo si, $(p, aw, \epsilon) \vdash^*_{\mathcal{T}} (p', w, b) \vdash^*_{\mathcal{T}} (q, \epsilon, bw')$ por el caso base. Luego, $(p', w, \epsilon) \vdash^*_{\mathcal{T}} (q, \epsilon, w')$ por definición de ejecución de un transductor.

Por hipótesis de inducción se tiene que $(p', w', \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, w)$. Y por definición de ejecución se deduce que $(p', w', a) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, aw)$. Además, $(p, bw', \epsilon) \vdash^*_{\mathcal{T}^{-1}} (p', w', a)$ por la definición de $\mathcal{T}^{-1}$. Uniendo las últimas 2 ejecuciones mencionadas se obtiene $(p, bw', \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, aw)$, es decir, $(p, v, \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, u)$.

($\Leftarrow$) Si $(p, v, \epsilon) \vdash^*_{\mathcal{T}^{-1}} (q, \epsilon, u)$, entonces $(p, u, \epsilon) \vdash^*_{\mathcal{T}} (q, \epsilon, v)$ por un argumento simétrico al anterior.

2. Si R y S son relaciones racionales, entonces $R \circ S$ es una relación racional.

   La respuesta es *Verdadero*.

   Si R y S son relaciones racionales, entonces existen los transductores $\mathcal{T}_1 = (Q_1, \Sigma, \Omega, \Delta_1, I_1, F_1)$ y $\mathcal{T}_2 = (Q_2, \Omega, \Gamma, \Delta_2, I_2, F_2)$ tal que $R = [\![\mathcal{T}_1]\!]$ y $S = [\![\mathcal{T}_2]\!]$. Defina $\mathcal{T} = (Q_1 \times Q_2, \Sigma, \Gamma, \Delta, I_1 \times I_2, F_1 \times F_2)$ tal que:

$$\begin{aligned}\Delta \quad = \quad & \{((p_1, p_2), a, c, (q_1, q_2)) \mid \exists b \in \Omega.\ (p_1, a, b, q_1) \in \Delta_1 \wedge (p_2, b, c, q_2) \in \Delta_2\} \cup \\ & \{((p_1, p_2), \epsilon, c, (p_1, q_2)) \mid \forall p_1 \in Q_1.\ (p_2, \epsilon, c, q_2) \in \Delta_2\} \cup \\ & \{((p_1, p_2), a, \epsilon, (q_1, p_2)) \mid \forall p_2 \in Q_2.\ (p_1, a, \epsilon, q_1) \in \Delta_1\}\end{aligned}$$

   Intuitivamente, el transductor anterior hace lo siguiente. Se busca simular 2 ejecuciones. La de un transductor que genera un output y la de otro transductor que lo lee. Para hacer lo anterior en solamente una ejecución se utiliza la idea de una ejecución en paralelo. En cuanto al producto cruz, este permite almacenar la información necesaria para el paralelismo. En cuanto a Δ, la primera parte simula en un paso que T_1 y T_2 ejecuten un paso. La segunda y tercera parte de Δ simulan que un transductor avance sin la necesidad del otro. Si T_1 escribe ϵ esto no afecta el input de T_2 y puede hacerlo libremente. Si T_2 lee ϵ no es necesario que T_1 genere input él.

   Sea $u \in \Sigma^*$, $w \in \Omega^*$ y $v \in \Gamma^*$. Por demostrar, a través del principio de inducción fuerte, que $(u, w) \in [\![\mathcal{T}_1]\!]$ y $(w, v) \in [\![\mathcal{T}_2]\!]$ si, y solo si, $(u, v) \in [\![\mathcal{T}]\!]$. Esto es, $(p_1, u, \epsilon) \vdash^*_{\mathcal{T}_1} (q_1, \epsilon, w)$ y $(p_2, w, \epsilon) \vdash^*_{\mathcal{T}_2} (q_2, \epsilon, v)$ si, y solo si, $((p_1, p_2), u, \epsilon) \vdash^*_{\mathcal{T}} ((q_1, q_2), \epsilon, v)$.

Problema 3

Considere el siguiente problema:

Problema:	$\mathcal{T} - eval$
Input:	Un transductor $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ y $u \in \Sigma^*$
Output:	$v \in \Omega^*.\ v \in [\![T]\!](u)$

Escriba un algoritmo que resuelva $\mathcal{T} - eval$ en tiempo $\mathcal{O}(|\mathcal{T}| \cdot |u|)$. Recordar que los transductores no tienen transiciones de lectura con ϵ.

Solución

Modificando el algoritmo visto en clases para la evaluación de NFA *on-the-fly*, tenemos que para un transductor $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ tal que $\Delta \subseteq Q \times \Sigma \times (\Omega \cup \{\varepsilon\}) \times Q$ tenemos:

$T - eval\ (\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F),\ w = a_1 a_2 \dots a_n)$

```
S ← hash(Q, null)
for q ∈ I do
    S[q] ← ε
end for
for i := 1 to n do
    S_old ← S
    S ← hash(Q, null)
    for p ∈ Q do
        if S_old[p] ≠ null then
            for (p, a_i, b, q) ∈ Δ do
                if S[q] = null then
                    S[q] ← S_old[p] · b
                end if
            end for
        end if
    end for
end for
for q ∈ F do
    if S[q] ≠ null then
        return S[q]
    end if
end for
return null
```

La idea del algoritmo es llevar una tabla de hash con las palabras escritas por las ejecuciones actuales que se encuentran en cada estado, e ir completándola según nos indican las transiciones del transductor.

Estas tablas comienzan con un valor *null* **distinto de epsilon**, indicando que no hay ninguna ejecución en proceso, y a continuación se inicializan con ε las entradas de los estados iniciales, indicando que hay una ejecución activa, pero que no ha escrito nada aún. Teniendo lo anterior, se recorre la palabra letra por letra, visitando los estados que tengan alguna ejecución activa ($S_{old}[p] \neq null$), se revisan todas las transiciones salientes de este estado (para la letra actual) y en caso de encontrar una transición válida se concatena la letra que indica la transición a la palabra asociada al estado de llegada. Cabe señalar que solamente se concatena esta letra si es la primera transición válida que encontramos ($S[q] \neq null$), pues buscamos que el algoritmo retorne solamente una palabra, por lo que no es necesario llevar registro de todas las posibles.

Finalmente, luego de haber leído la palabra completa, visitamos todos los estados finales, retornando la palabra escrita por el transductor en caso de encontrar alguna y retornando *null* en caso de que ninguna ejecución llegue a un estado final.