Algoritmo de Knuth-Morris-Prat

Clase 14

IIC 2223

Prof. Cristian Riveros

Problema de pattern matching de una palabra

Problema

Dado un **patrón** $w = w_1 \dots w_m$ y un **documento** $d = d_1 \dots d_n$,
encontrar todos las posiciones donde aparece w en d, o sea, enumerar:

$$\{(i,j) \mid w = d_i d_{i+1} \dots d_j\}$$

Solución ingenua

for $i = 0$ **to** $n - m$ **do**
    $j := 1$
    while $j \leq m \wedge w_j = d_{i+j}$ **do**
        $j := j + 1$
    if $j > m$ **then**
        output $(i + 1, i + m + 1)$

¿es posible hacerlo mejor?

Outline

Autómata de un patrón

k-lookahead

Algoritmo de Knuth-Morris-Prat

Outline

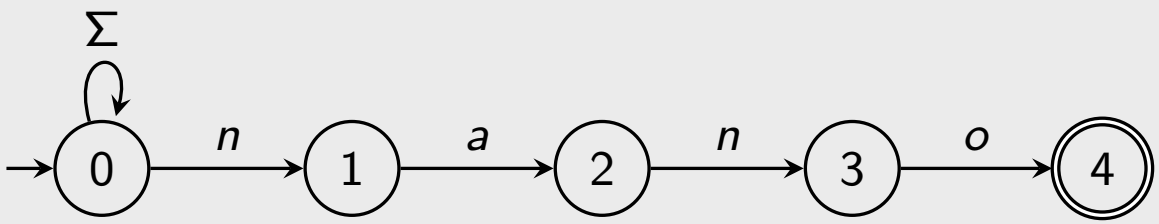
Autómata de un patrón

Definición

Dado un palabra $w = w_1 \dots w_m$, sea el NFA $\mathcal{A}_w = (Q, \Sigma, \Delta, I, F)$ tal que:

- $Q = \{0, 1, \dots, m\}$
- $\Delta = \{(0, a, 0) \mid a \in \Sigma\} \cup \{(i, w_{i+1}, i+1) \mid i < m\}$
- $I = \{0\}$ y $F = \{m\}$.

Ejemplo: palabra w = nano



¿cómo podemos usar $\mathcal{A}_w$ para encontrar todas las apariciones de w en d?

Determinización de $\mathcal{A}_w$

Sea $\mathcal{A}_w^{\text{det}} = (Q^{\text{det}}, \Sigma, \delta^{\text{det}}, \{0\}, F^{\text{det}})$ la determinización de $\mathcal{A}_w$ tal que Q^{det} contiene **solo los estados alcanzables** desde $\{0\}$.

Recordatorio

Para un autómata no-determinista $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$,
se define el autómata determinista (**determinización** de $\mathcal{A}$):

$$\mathcal{A}^{\text{det}} \quad = \quad (Q^{\text{det}}, \Sigma, \delta^{\text{det}}, q_0^{\text{det}}, F^{\text{det}})$$

- $Q^{\text{det}} = 2^Q = \{S \mid S \subseteq Q\}$
- $q_0^{\text{det}} = I$.
- $\delta^{\text{det}} : 2^Q \times \Sigma \to 2^Q$ tal que:

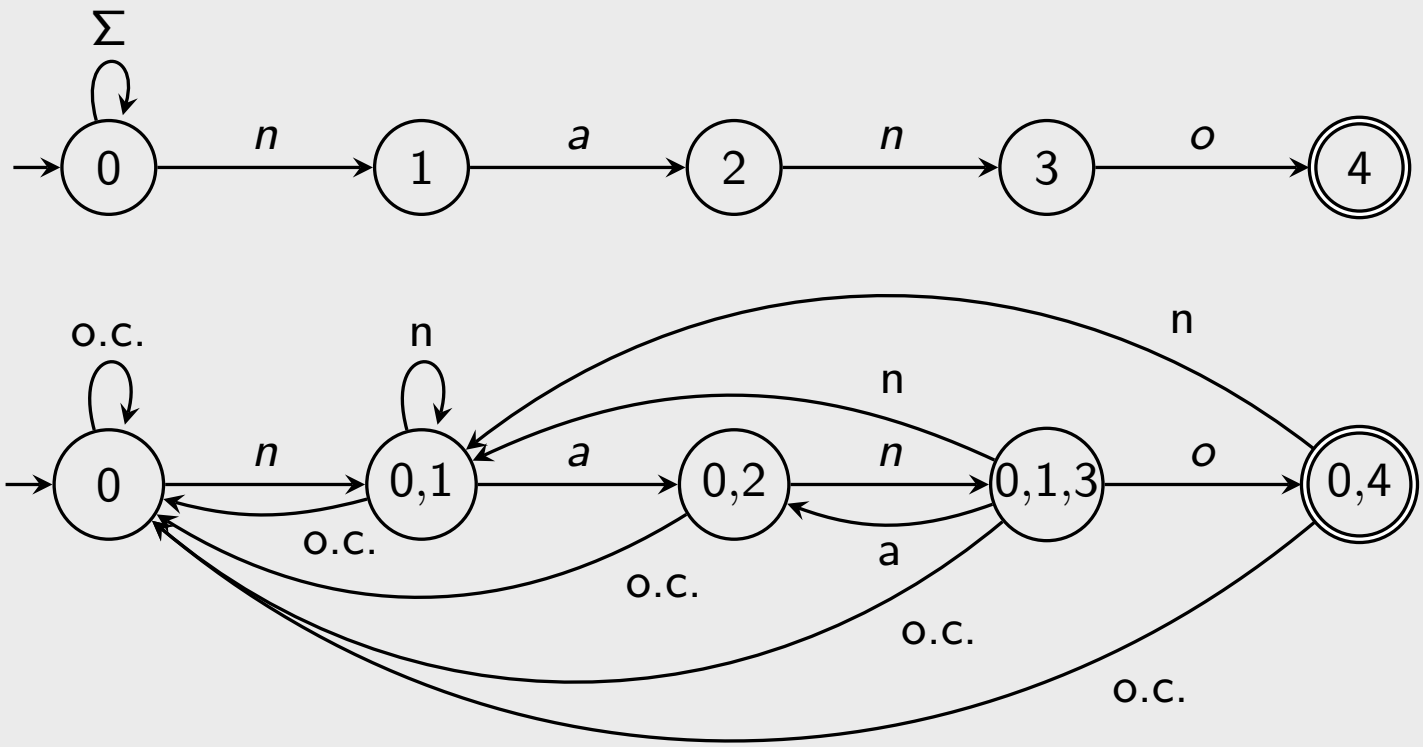
$$\delta^{\text{det}}(S, a) = \{q \in Q \mid \exists p \in S.\ (p, a, q) \in \Delta\}$$

- $F^{\text{det}} = \{S \in 2^Q \mid S \cap F \neq \varnothing\}$.

Determinización de $\mathcal{A}_w$

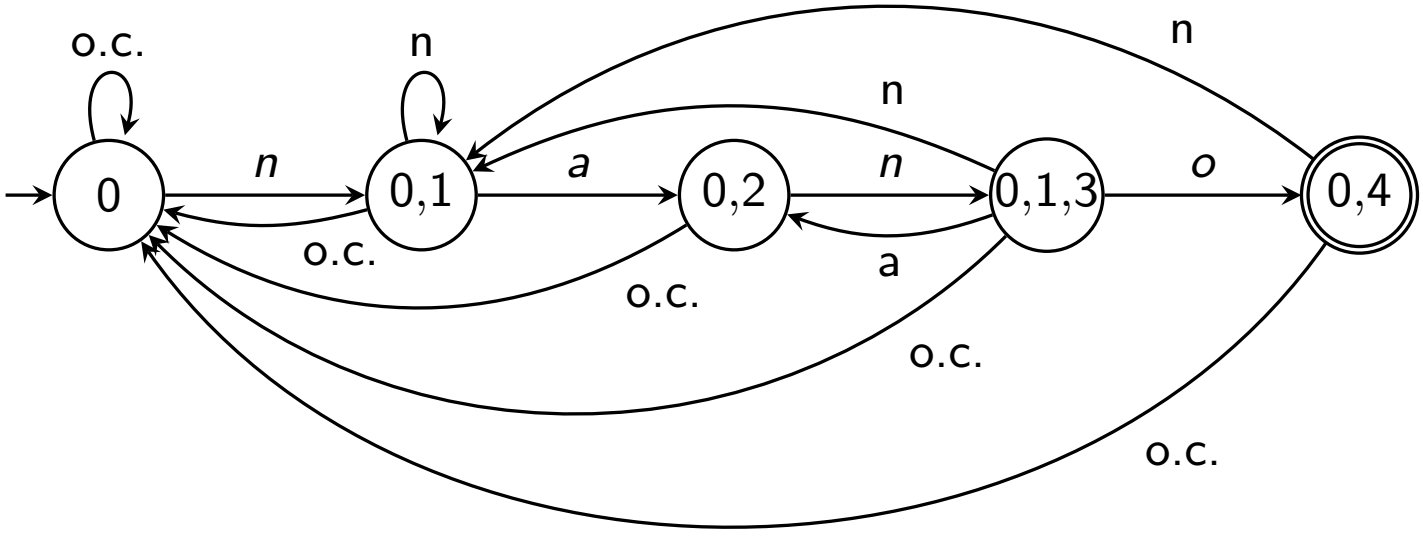
Sea $\mathcal{A}_w^{\text{det}} = (Q^{\text{det}}, \Sigma, \delta^{\text{det}}, \{0\}, F^{\text{det}})$ la determinización de $\mathcal{A}_w$ tal que Q^{det} contiene **solo los estados alcanzables** desde $\{0\}$.

Ejemplo: palabra w = nano



¿cuál es el problema de construir $\mathcal{A}_w^{\text{det}}$?

¿cómo utilizamos $\mathcal{A}_w^{\text{det}}$ para encontrar todos los matches?



q_0	q_0	q_1	q_0	q_1	q_2	q_3 ✓	q_4	q_0	q_1	q_0	q_0	q_1	q_2	q_3	q_2
u	*n*		*n*	*a*	*n*	*o*		*n*	*o*		*n*	*a*	*n*	*a*	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

¿cuál es el tiempo de este algoritmo una vez construido $\mathcal{A}_w^{\text{det}}$?

¿cuál es el tamaño de $\mathcal{A}_w^{\text{det}}$?

Sea $w = w_1 \dots w_m$ y $\mathcal{A}_w^{\text{det}} = (Q^{\text{det}}, \Sigma, \delta^{\text{det}}, \{0\}, F^{\text{det}})$ la determ. de $\mathcal{A}_w$.

Teorema

Para todo $S \in Q^{\text{det}}$ y $i \in \{0, 1, \dots, m\}$ se cumple que:

$$i \in S \quad \text{si, y solo si,} \quad w_1 \dots w_i \text{ es un sufijo de } w_1 \dots w_{\max(S)}.$$

Corolarios

- Para todo $S_1, S_2 \in Q^{\text{det}}$, si $\max(S_1) = \max(S_2)$, entonces $S_1 = S_2$.
- $\mathcal{A}_w^{\text{det}}$ tiene $|w| + 1$ estados y a lo más $\mathcal{O}(|w|^2)$ transiciones.

Por lo tanto, encontrar todos los substrings de w en d
toma tiempo $\mathcal{O}(|d| + |w|^2)$

¿cuál es el tamaño de $\mathcal{A}_w^{\text{det}}$?

Demostración teorema

Sea $S \in Q^{\text{det}}$ un conjunto de estados cualquiera alcanzable desde $\{0\}$.

Entonces existe una palabra $u = a_1 \dots a_k$ tal que $\hat{\delta}^{\text{det}}(\{0\}, u) = S$.

Por la demostración que $\mathcal{L}(\mathcal{A}^{\text{det}}) = \mathcal{L}(\mathcal{A})$ para todo NFA $\mathcal{A}$ (Clase 03), sabemos que $j \in S$ si, y solo si, existe una ejecución de $\mathcal{A}_w$ sobre u:

$$0 = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_k} q_k = j.$$

Por la definición de $\mathcal{A}_w$ esta ejecución es de la forma:

$$0 \xrightarrow{a_1} 0 \xrightarrow{a_2} \dots \xrightarrow{a_{k-j}} 0 \underbrace{\xrightarrow{a_{k-j+1}} 1 \xrightarrow{a_{k-j+2}} 2 \dots \xrightarrow{a_k} j}_{w_1 \dots w_j}.$$

Por lo tanto, $w_1 w_2 \dots w_j$ es sufijo de $a_1 \dots a_k$.

Usaremos este último hecho para demostrar **ambas direcciones**.

¿cuál es el tamaño de $\mathcal{A}_w^{\det}$?

Propiedad

Para toda $u = a_1 \dots a_k$ tal que $\hat{\delta}^{\det}(\{0\}, u) = S$, y para todo $j \leq m$:

$$j \in S \quad \text{si, y solo si,} \quad w_1 \dots w_j \text{ es sufijo de } a_1 \dots a_k$$

Demostración teorema ($\Rightarrow$)

Como S es alcanzable desde $\{0\}$,
entonces existe $u = a_1 \dots a_k$ tal que $\hat{\delta}^{\det}(\{0\}, u) = S$.

Como $\max(S) \in S$, entonces $w_1 \dots w_{\max(S)}$ es sufijo de $a_1 \dots a_k$.

Suponga que $i \in S$. Entonces $w_1 \dots w_i$ es sufijo de $a_1 \dots a_k$.

Como $i \leq \max(S)$, entonces:

$$a_1 a_2 \dots a_{k-\max(S)} \overbrace{a_{k-\max(S)+1} \dots a_{k-i} \underbrace{a_{k-i+1} \dots a_k}_{w_1 \dots w_i}}^{w_1 \dots w_{\max(S)}}$$

Por lo tanto, $w_1 \dots w_i$ es sufijo de $w_1 \dots w_{\max(S)}$.

¿cuál es el tamaño de $\mathcal{A}_w^{\det}$?

Propiedad

Para toda $u = a_1 \dots a_k$ tal que $\hat{\delta}^{\det}(\{0\}, u) = S$, y para todo $j \leq m$:

$$j \in S \quad \text{si, y solo si,} \quad w_1 \dots w_j \text{ es sufijo de } a_1 \dots a_k$$

Demostración teorema ($\Leftarrow$)

Como S es alcanzable desde $\{0\}$,
entonces existe $u = a_1 \dots a_k$ tal que $\hat{\delta}^{\det}(\{0\}, u) = S$.

Como $\max(S) \in S$, entonces $w_1 \dots w_{\max(S)}$ es sufijo de $a_1 \dots a_k$.

Suponga que $w_1 \dots w_i$ es sufijo de $w_1 \dots w_{\max(S)}$.

Como $w_1 \dots w_i$ es sufijo de $w_1 \dots w_{\max(S)}$ y $w_1 \dots w_{\max(S)}$ es sufijo de u,
entonces $w_1 \dots w_i$ es sufijo de $u = a_1 \dots a_k$.

Por la "Propiedad", concluimos que $i \in S$.

¿cuál es el tamaño de $\mathcal{A}_w^{\text{det}}$?

Sea $w = w_1 \dots w_m$ y $\mathcal{A}_w^{\text{det}} = (Q^{\text{det}}, \Sigma, \delta^{\text{det}}, \{0\}, F^{\text{det}})$ la determ. de $\mathcal{A}_w$.

Teorema

Para todo $S \in Q^{\text{det}}$ y $i \in \{0, 1, \dots, m\}$ se cumple que:

$$i \in S \quad \text{si, y solo si,} \quad w_1 \dots w_i \text{ es un sufijo de } w_1 \dots w_{\max(S)}.$$

Corolarios

$\mathcal{A}_w^{\text{det}}$ tiene $|w| + 1$ estados y $\mathcal{O}(|w|^2)$ transiciones.

Por lo tanto, encontrar todos los substrings de w en d
toma tiempo $\mathcal{O}(|d| + |w|^2)$

¿es posible hacerlo mejor?

Outline

Autómata finito con k-lookahead

Sea Σ un alfabeto finito.

Definiciones

Se definen los siguientes conjuntos de palabra:

- $\Sigma_\bullet = \Sigma^* \times \Sigma^*$
- $\Sigma_\bullet^k = \left\{ (u, v) \in \Sigma_\bullet \mid |uv| = k \right\}$

Notación

En vez de $(u, v) \in \Sigma_\bullet$, escribiremos $u.v \in \Sigma_\bullet$.

Ejemplos

Si $\Sigma = \{a, b\}$ entonces:

- $ab.ba \in \Sigma_\bullet$ y $.aba \in \Sigma_\bullet$
- $ab.ba \in \Sigma_\bullet^4$ y $.aba \in \Sigma_\bullet^3$

Autómata finito con k-lookahead

Definición

Un autómata finito determinista con k-lookahead es:

$$\mathcal{A} \quad = \quad (Q, \Sigma, \delta, q_0, F)$$

- Q es un conjunto finito de estados.
- Σ es el alfabeto de input.
- q_0 es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales.

$$+$$

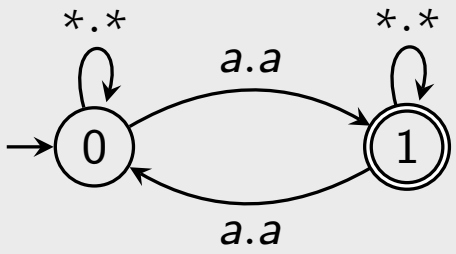
- $\delta : Q \times (\Sigma \cup \{\$\})^k_\bullet \rightharpoonup Q$ es una función parcial, tal que:

  para todo $p \in Q$ y $w \in (\Sigma \cup \{\$\})^k$: $\quad |\{\, u.v \mid \delta(p, u.v) = q \text{ y } uv = w \,\}| \leq 1.$

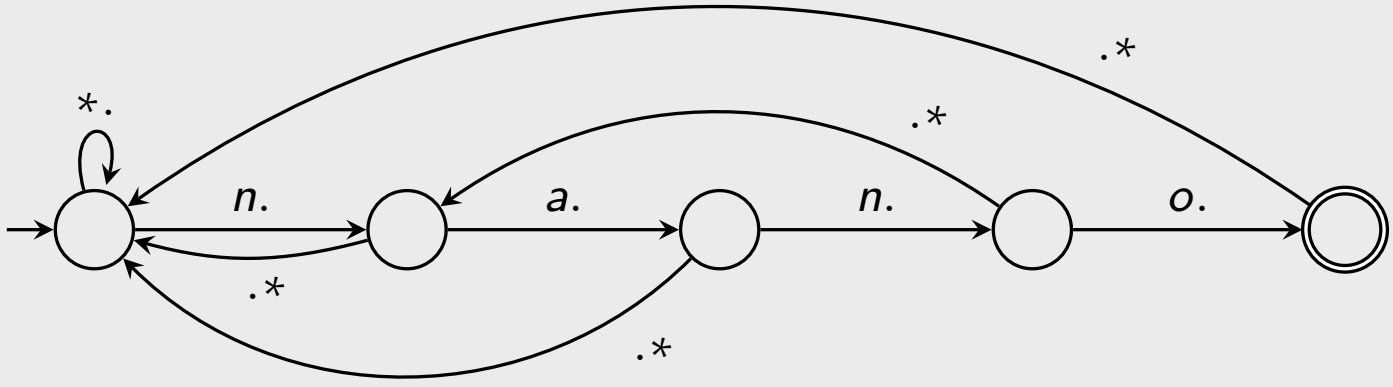
Autómata finito con k-lookahead

Algunos ejemplos

1.



2.



Ejecución de DFA con k-lookahead

Sea $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA con k-lookahead.

Definiciones

- Un par $(q, w) \in Q \times (\Sigma \cup \{\$\})^*$ es una **configuración** de $\mathcal{A}$.
- Una configuración $(q_0, w\$^k)$ es **inicial**.
- Una configuración $(q, \$^k)$ es **final** si $q \in F$.

El sufijo $\$^k$ no sirve para marcar el final del input
(y simplificar la definición de lookahead al leer el final de la palabra)

Ejecución de DFA con k-lookahead

Sea $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA con k-lookahead.

Definición

Se define la relación $\vdash_{\mathcal{A}}$ de **siguiente-paso** entre configuraciones de $\mathcal{A}$:

$$(p_1, w_1) \vdash_{\mathcal{A}} (p_2, w_2)$$

si, y solo si, $\delta(p_1, u.v) = p_2$ y existe $w \in \Sigma^*$ tal que $w_1 = uvw$ y $w_2 = vw$.

Se define $\vdash_{\mathcal{A}}^*$ como la clausura **refleja** y **transitiva** de $\vdash_{\mathcal{A}}$:

$$\text{para toda configuración } (p, w): \quad (p, w) \vdash_{\mathcal{A}}^* (p, w)$$

$$\text{si } (p_1, w_1) \vdash_{\mathcal{A}}^* (p_2, w_2) \text{ y } (p_2, w_2) \vdash_{\mathcal{A}} (p_3, w_3): \quad (p_1, w_1) \vdash_{\mathcal{A}}^* (p_3, w_3)$$

$(p, u) \vdash_{\mathcal{A}}^* (q, v)$ si uno puede ir de (p, u) a (q, v) en **0 o más pasos**.

Ejecución de DFA con k-lookahead

Sea $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA con k-lookahead.

Definiciones

- $\mathcal{A}$ **acepta** w si existe una configuración **inicial** $(q_0, w\$^k)$ y una configuración **final** $(q_f, \$^k)$ tal que:

$$(q_0, w\$^k) \vdash_{\mathcal{A}}^{*} (q_f, \$^k)$$

- El **lenguaje aceptado** por $\mathcal{A}$ se define como:

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ acepta } w\}$$

Mismas definiciones que para ϵ-NFA.

k-lookahead y lenguajes regulares

Teorema

Para todo DFA con k-lookahead $\mathcal{A}$
se tiene que $\mathcal{L}(\mathcal{A})$ es un **lenguaje regular**.

Demostración: ejercicio.

Definición

Llamaremos un **lazy automata** a un DFA con 1-lookahead.

¿cuál es la ventaja de un lazy autómata?

Outline

Construcción de un lazy autómata a partir de $\mathcal{A}_w^{\text{det}}$

Sea $w = w_1 \dots w_m$ y $\mathcal{A}_w^{\text{det}} = (Q^{\text{det}}, \Sigma, \delta^{\text{det}}, \{0\}, F^{\text{det}})$ la determ. de $\mathcal{A}_w$.

Definición

Para $i \in [0, m]$, sea S_i el **único estado** en Q^{det} tal que $i = \max(S_i)$.

(¿por qué S_i es único?)

Propiedad 2

Para todo $a \in \{w_1, \dots, w_m\}$ y $i \in [0, m-1]$:

1. $S_i - \{i\} \in Q^{\text{det}}$.
2. $a = w_{i+1}$, entonces $\delta^{\text{det}}(S_i, a) = S_{i+1}$.
3. $a \neq w_{i+1}$, entonces $\delta^{\text{det}}(S_i, a) = \delta^{\text{det}}(S_i - \{i\}, a)$.

Demostración: ejercicio.

¿cómo podemos construir un lazy autómata usando la Propiedad 2?

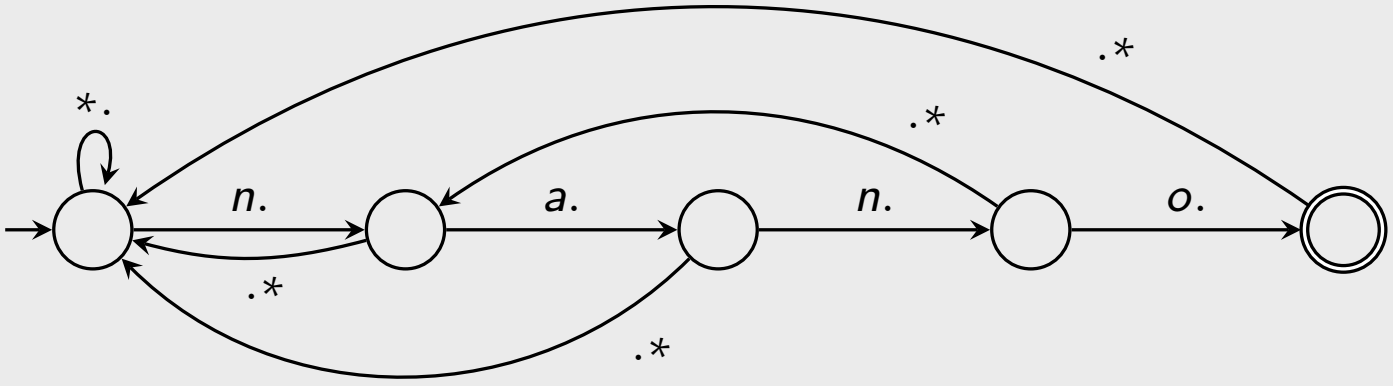
Construcción de un lazy autómata a partir de $\mathcal{A}_w^{\text{det}}$

Construcción

Se define el lazy autómata $\mathcal{A}_w^{\text{lazy}} = (Q^{\text{det}}, \Sigma, \delta^{\text{lazy}}, \{0\}, F^{\text{det}})$ tal que:

- para todo $a \neq w_1$: $\delta^{\text{lazy}}(\{0\}, a.) = \{0\}$.
- para todo $a \in \{w_1, \ldots, w_m\}$ y $i \in [0, m-1]$:
  - si $a = w_{i+1}$, entonces $\delta^{\text{lazy}}(S_i, a.) = S_{i+1}$
  - si $a \neq w_{i+1}$ y $i \neq 0$, entonces $\delta^{\text{lazy}}(S_i, .a) = S_i - \{i\}$.

Ejemplo



Construcción de un lazy autómata a partir de $\mathcal{A}_w^{\text{det}}$

Construcción

Se define el lazy autómata $\mathcal{A}_w^{\text{lazy}} = (Q^{\text{det}}, \Sigma, \delta^{\text{lazy}}, \{0\}, F^{\text{det}})$ tal que:

- para todo $a \neq w_1$: $\delta^{\text{lazy}}(\{0\}, a.) = \{0\}$.
- para todo $a \in \{w_1, \ldots, w_m\}$ y $i \in [0, m-1]$:
  - si $a = w_{i+1}$, entonces $\delta^{\text{lazy}}(S_i, a.) = S_{i+1}$
  - si $a \neq w_{i+1}$ y $i \neq 0$, entonces $\delta^{\text{lazy}}(S_i, .a) = S_i - \{i\}$.

Teorema

Para todo w se cumple que $\mathcal{L}(\mathcal{A}_w^{\text{det}}) = \mathcal{L}(\mathcal{A}_w^{\text{lazy}})$.

Demostración: ejercicio. (usando Propiedad 2)

¿cuántos pasos toma $\mathcal{A}_w^{\text{lazy}}$ sobre un documento d?

- Número de pasos que $\mathcal{A}_w^{\text{lazy}}$ **consume** letras $= |d|$
- Número de pasos que $\mathcal{A}_w^{\text{lazy}}$ **retrocede** $\leq |d|$
- Número de **pasos totales** de $\mathcal{A}_w^{\text{lazy}}$ $\leq 2 \cdot |d|$

Por lo tanto, la cantidad de pasos es **lineal** en $\mathcal{O}(|d|)$.

Algoritmo de Knuth-Morris-Prat

Algoritmo

Dado una palabra w y un documento d:

- Construimos $\mathcal{A}_w^{\text{lazy}}$ desde $\mathcal{A}_w$. $\mathcal{O}(|w|)$
- Ejecutamos $\mathcal{A}_w^{\text{lazy}}$ sobre d. $\mathcal{O}(|d|)$

Tiempo del algoritmo: $\mathcal{O}(|w| + |d|)$

Ejercicio: demuestre como construir $\mathcal{A}_w^{\text{lazy}}$ en tiempo $\mathcal{O}(|w|)$