Contando las palabras aceptadas por un autómata

IIC3810

Una primera motivación

¿Cuál es la complejidad de este problema?

Dado un grafo G, queremos contar el número de subgrafos de G que son 3 colorables

La clase de funciones SpanP

Extendemos la noción de MTC para el caso no determinista

- Una MTC no determinista tiene un conjunto de estados finales, y una salida se considera válida si la máquina termina en un estado final

La clase de funciones SpanP

Extendemos la noción de MTC para el caso no determinista

- Una MTC no determinista tiene un conjunto de estados finales, y una salida se considera válida si la máquina termina en un estado final

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ *está en SpanP si y sólo si existe una MTC no determinista* M *con alfabeto de entrada* Σ*, que funciona en tiempo polinomial y tal que para cada* $w \in \Sigma^*$*:*

$f(w)$ *es igual al número de salidas válidas de* M *con entrada* w

La clase de funciones SpanP

Ejercicios

1. Sea #SUB-3-COL una función tal que, dado un grafo G, cuenta el número de subgrafos de G que son 3-coloreables. Muestre que #SUB-3-COL $\in$ SpanP

2. Demuestre que #P $\subseteq$ SpanP

$\# \cdot \mathcal{C}$: una visión unificada

$\mathcal{C}$ es una clase de complejidad para problemas de decisión

$\# \cdot \mathcal{C}$: una visión unificada

$\mathcal{C}$ es una clase de complejidad para problemas de decisión

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ está en $\# \cdot \mathcal{C}$ si existe una relación $R \in \mathcal{C}$ y un polynomio p tal que:

- *Si $(x, y) \in R$, entonces $|y| \leq p(|x|)$*
- *$f(x) = N_R(x)$ para todo $x \in \Sigma^*$*

$\# \cdot \mathcal{C}$: una visión unificada

Claramente tenemos que $\#\mathrm{P} = \# \cdot \mathrm{P}$

$\# \cdot \mathcal{C}$: una visión unificada

Claramente tenemos que $\#\mathrm{P} = \# \cdot \mathrm{P}$

Proposition

$SpanP = \# \cdot NP$

$\# \cdot \mathcal{C}$: una visión unificada

Claramente tenemos que $\#\mathrm{P} = \# \cdot \mathrm{P}$

Proposition

$SpanP \; = \; \# \cdot NP$

Ejercicio

Demuestre la proposición

Un problema completo para SpanP

¿Qué variante del 3-CNF-SAT es completo para SpanP?

- Bajo reducciones parsimoniosas

Un problema completo para SpanP

¿Qué variante del 3-CNF-SAT es completo para SpanP?

- Bajo reducciones parsimoniosas

Dada una formula proposicional φ en 3-CNF con variables $\{x_1, \ldots, x_n\}$ y $k \in \{1, \ldots, n\}$, defina $\#$SUB-3-CNF-SAT(φ, k) como:

$$\left|\left\{\sigma : \{x_1, \ldots, x_k\} \to \{0,1\} \;\middle|\; \text{existe } \sigma' : \{x_{k+1}, \ldots, x_n\} \to \{0,1\} \text{ tal que } (\sigma \cup \sigma')(\varphi) = 1\right\}\right|$$

Un problema completo para SpanP

¿Qué variante del 3-CNF-SAT es completo para SpanP?

- Bajo reducciones parsimoniosas

Dada una formula proposicional φ en 3-CNF con variables $\{x_1, \ldots, x_n\}$ y $k \in \{1, \ldots, n\}$, defina $\#$SUB-3-CNF-SAT(φ, k) como:

$$\left|\left\{\sigma : \{x_1, \ldots, x_k\} \to \{0,1\} \;\middle|\; \text{existe } \sigma' : \{x_{k+1}, \ldots, x_n\} \to \{0,1\} \right.\right.$$
$$\left.\left. \text{tal que } (\sigma \cup \sigma')(\varphi) = 1 \right\}\right|$$

Teorema

$\#$SUB-3-CNF-SAT es SpanP-completo bajo reducciones parsimoniosas

La clase de funciones SpanL

¿Recuerda las clases de complejidad L y NL?

- ¿Cómo se define el uso de espacio en una Máquina de Turing?

La clase de funciones SpanL

¿Recuerda las clases de complejidad L y NL?

- ¿Cómo se define el uso de espacio en una Máquina de Turing?

Extendemos la noción de MTC para el caso no determinista y con uso restringido de espacio

- Nuevamente consideramos un conjunto de estados finales, y una salida se considera válida si la máquina termina en un estado final

La clase de funciones SpanL

¿Recuerda las clases de complejidad L y NL?

- ¿Cómo se define el uso de espacio en una Máquina de Turing?

Extendemos la noción de MTC para el caso no determinista y con uso restringido de espacio

- Nuevamente consideramos un conjunto de estados finales, y una salida se considera válida si la máquina termina en un estado final

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ está en SpanL si y sólo si existe una MTC no determinista M con alfabeto de entrada Σ, que funciona en espacio logarítmico y tal que para cada $w \in \Sigma^*$:

$$f(w) \text{ es igual al número de salidas válidas de } M \text{ con entrada } w$$

La clase de funciones SpanL

Ejercicios

1. Demuestre que si $f \in \text{SpanL}$, entonces $L_f \in \text{P}$
2. Demuestre que $\text{SpanL} \subseteq \#\text{P}$
3. Demuestre que $\#\text{DNF-SAT} \in \text{SpanL}$
4. Demuestre que si $\text{SpanL} \subseteq \text{FP}$, entonces $\text{FP} = \#\text{P}$

¿Podemos utilizar la notación $\# \cdot \mathcal{C}$ para caracterizar SpanL?

¿Es cierto que SpanL $= \# \cdot$ NL?

¿Podemos utilizar la notación $\# \cdot \mathcal{C}$ para caracterizar SpanL?

¿Es cierto que SpanL $= \# \cdot$ NL?

- Imponemos una restricción a la forma en que se procesa el segundo argumento y de una entrada (x, y)

¿Podemos utilizar la notación $\# \cdot \mathcal{C}$ para caracterizar SpanL?

¿Es cierto que SpanL $= \# \cdot$ NL?

- Imponemos una restricción a la forma en que se procesa el segundo argumento y de una entrada (x, y)

Una 2-1-MT es una MT que tiene dos cintas de entrada

- La primera funciona de manera usual
- La segunda solo se puede leer una vez de izquierda a derecha

¿Podemos utilizar la notación $\# \cdot \mathcal{C}$ para caracterizar SpanL?

¿Es cierto que SpanL $= \# \cdot$ NL?

- Imponemos una restricción a la forma en que se procesa el segundo argumento y de una entrada (x, y)

Una 2-1-MT es una MT que tiene dos cintas de entrada

- La primera funciona de manera usual
- La segunda solo se puede leer una vez de izquierda a derecha

Usamos 2-1-MT cuando aceptamos relaciones en espacio logarítmico

¿Podemos utilizar la notación $\# \cdot \mathcal{C}$ para caracterizar SpanL?

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ está en $\# \cdot NL$ si existe una relación R y un polynomio p tal que:

- *R es aceptada por una 2-1-MT no determinista y que funciona en espacio logarítmico*
- *Si $(x, y) \in R$, entonces $|y| \leq p(|x|)$*
- *$f(x) = N_R(x)$ para todo $x \in \Sigma^*$*

¿Podemos utilizar la notación $\# \cdot \mathcal{C}$ para caracterizar SpanL?

Definición

Una función $f : \Sigma^* \to \mathbb{N}$ está en $\# \cdot NL$ si existe una relación R y un polynomio p tal que:

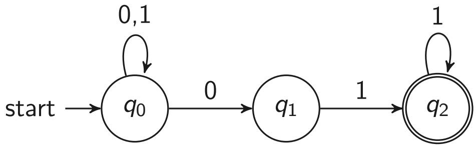
- *R es aceptada por una 2-1-MT no determinista y que funciona en espacio logarítmico*
- *Si $(x, y) \in R$, entonces $|y| \leq p(|x|)$*
- *$f(x) = N_R(x)$ para todo $x \in \Sigma^*$*

Ejercicio

Demuestre que SpanL $= \# \cdot NL$

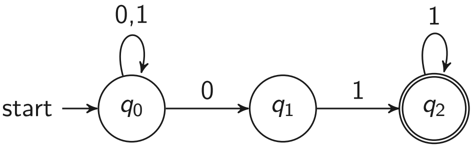
El problema #NFA

Para recordar: un automata finito no deterministico (NFA) con alfabeto $\{0, 1\}$



El problema #NFA

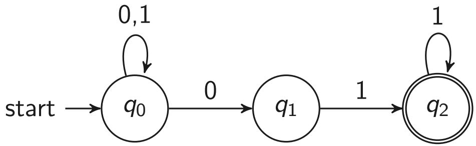
Para recordar: un automata finito no deterministico (NFA) con alfabeto $\{0, 1\}$



Queremos contar el número de palabras de un largo dado aceptados por un NFA

El problema #NFA

Para recordar: un automata finito no deterministico (NFA) con alfabeto $\{0,1\}$

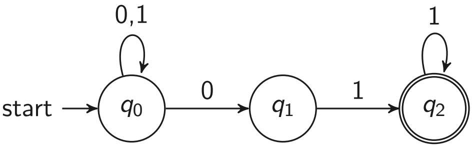


Queremos contar el número de palabras de un largo dado aceptados por un NFA

- ¿Cuántos palabras de largo 10 acepta el autómata mostrado arriba?

El problema #NFA

Para recordar: un automata finito no deterministico (NFA) con alfabeto $\{0,1\}$



Queremos contar el número de palabras de un largo dado aceptados por un NFA

- ¿Cuántos palabras de largo 10 acepta el autómata mostrado arriba? 511

El problema #NFA

Dado un NFA $\mathcal{A}$ con alfabeto Σ y un número natural n en unario, defina $\#\text{NFA}(\mathcal{A}, n)$ como:

$$|\{w \in \Sigma^* \mid |w| = n \text{ y } w \text{ es aceptado por } \mathcal{A}\}|$$

El problema #NFA

Dado un NFA $\mathcal{A}$ con alfabeto Σ y un número natural n en unario, defina $\#\text{NFA}(\mathcal{A}, n)$ como:

$$|\{w \in \Sigma^* \mid |w| = n \text{ y } w \text{ es aceptado por } \mathcal{A}\}|$$

En la entrada de #NFA:

- ¿Cómo es entregado $\mathcal{A}$?
- ¿Como es entregado n? ¿Por qué debe estar en unario?

La complejidad de #NFA

Teorema

#NFA es SpanL-completo bajo reducciones parsimoniosas

La complejidad de #NFA

Teorema

#NFA es SpanL-completo bajo reducciones parsimoniosas

Corolario

#NFA es #P-completo

La complejidad de #NFA

Ejercicios

1. Demuestre que #NFA está en SpanL

2. Demuestre que #NFA es SpanL-hard bajo reducciones parsimoniosas.

   2.1 Demuestre el corolario a partir de este resultado

Aproximando #NFA

En esta capítulo vamos a demostrar que #NFA admite un FPRAS

Aproximando #NFA

En esta capítulo vamos a demostrar que #NFA admite un FPRAS

Vamos a concluir que cada problema en SpanL admite un FPRAS

Aproximando #NFA

En esta capítulo vamos a demostrar que #NFA admite un FPRAS

Vamos a concluir que cada problema en SpanL admite un FPRAS

- Esto nos da una manera alternativa para construir un FPRAS para una función

Aproximando #NFA

En esta capítulo vamos a demostrar que #NFA admite un FPRAS

Vamos a concluir que cada problema en SpanL admite un FPRAS

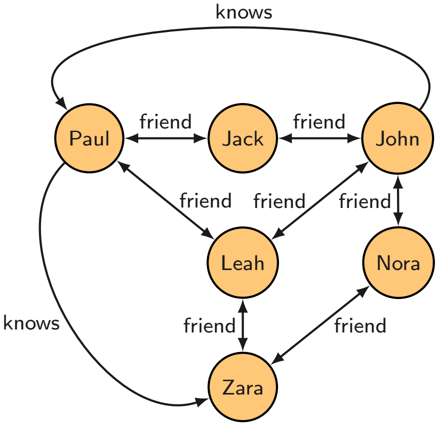
- Esto nos da una manera alternativa para construir un FPRAS para una función

Una de las motivaciones para demostrar que #NFA admite un FPRAS es tener una clase de complejidad definida por un modelo de máquina donde cada función admite un FPRAS

- Pero esta no es la única motivación

Una segunda motivación: bases de datos de grafos

G :



Una segunda motivación: bases de datos de grafos

L es un conjunto de etiquetas

Una segunda motivación: bases de datos de grafos

L es un conjunto de etiquetas

Un grafo con arcos etiquetados en L es una tupla $G = (N, A)$ donde

- N es un conjunto de nodos
- $A \subseteq N \times L \times N$ es un conjunto de arcos etiquetados en L

Una segunda motivación: bases de datos de grafos

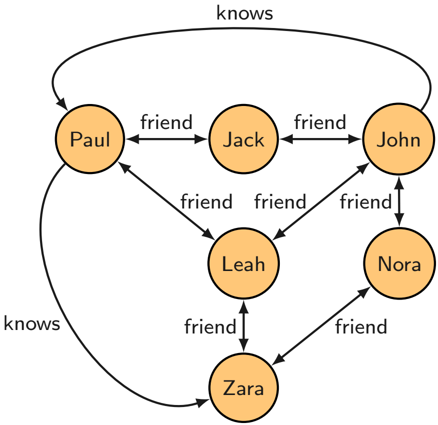
L es un conjunto de etiquetas

Un grafo con arcos etiquetados en L es una tupla $G = (N, A)$ donde

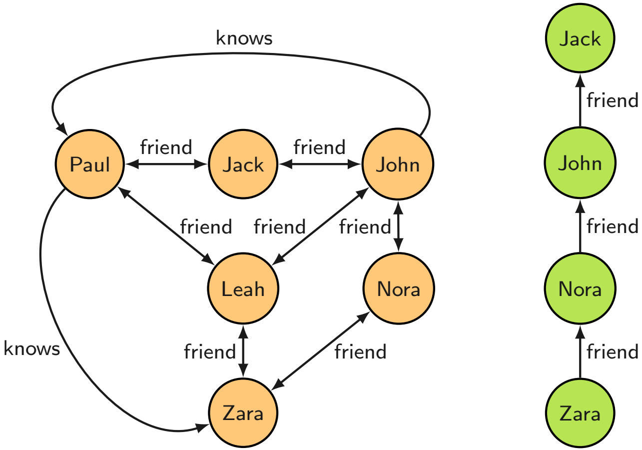
- N es un conjunto de nodos
- $A \subseteq N \times L \times N$ es un conjunto de arcos etiquetados en L

Una base de datos de grafos sobre L es un grafo con arcos etiquetados en L

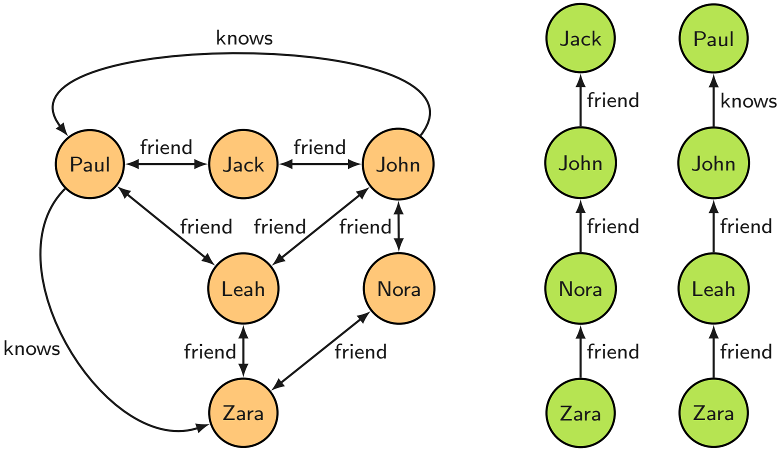
Una consulta sobre G: $(\text{friend} + \text{knows})^*$



Una consulta sobre G: (friend + knows)*



Una consulta sobre G: $(\text{friend} + \text{knows})^*$



Una consulta sobre G: $(\text{friend} + \text{knows})^*$

Sea L un conjunto de etiquetas y $G = (N, A)$ una base de datos de grafos sobre L

Una consulta sobre G: (friend + knows)*

Sea L un conjunto de etiquetas y $G = (N, A)$ una base de datos de grafos sobre L

La secuencia $\pi = u_0, \ell_1, u_1, \ell_2, \ldots, \ell_n, u_n$ es un camino en G si

- $n \geq 0$
- $(u_i, \ell_{i+1}, u_{i+1}) \in A$ para cada $i \in \{0, \ldots, n-1\}$

El camino π va desde u_0 a u_n y su largo es n

Una consulta sobre G: $(\text{friend} + \text{knows})^*$

Una consulta sobre G es una expressión regular r sobre L

- Esta consulta es una regular path query (RPQ)

Una consulta sobre G: $(\text{friend} + \text{knows})^*$

Una consulta sobre G es una expressión regular r sobre L

- Esta consulta es una regular path query (RPQ)

Un camino $\pi = u_0, \ell_1, u_1, \ell_2, \ldots, \ell_n, u_n$ satisface la expresión regular r si y sólo si $\ell_1 \cdots \ell_n$ es un palabra en el lenguaje definido por r

Una consulta sobre G: $(\text{friend} + \text{knows})^*$

Una consulta sobre G es una expressión regular r sobre L

- Esta consulta es una regular path query (RPQ)

Un camino $\pi = u_0, \ell_1, u_1, \ell_2, \ldots, \ell_n, u_n$ satisface la expresión regular r si y sólo si $\ell_1 \cdots \ell_n$ es un palabra en el lenguaje definido por r

Utilizamos la siguiente notación:

$$[\![r]\!]_G \quad = \quad \{\pi \mid \pi \text{ es un camino en } G \text{ y } \pi \text{ satisface } r\}$$

Dos problemas fundamentales en bases de datos de grafos

- REACH: Dado una base de datos G, un nodo de partida s, un nodo de llegada t y una expresión regular r, determinar si existe un camino p desde s a t tal que $p \in [\![r]\!]_G$

Dos problemas fundamentales en bases de datos de grafos

- REACH: Dado una base de datos G, un nodo de partida s, un nodo de llegada t y una expresión regular r, determinar si existe un camino p desde s a t tal que $p \in [\![r]\!]_G$

- #PATH: Dado una base de datos G, un nodo de partida s, un nodo de llegada t, una expresión regular r y un largo n, contar el número de caminos p desde s a t tales que el largo de p es n y $p \in [\![r]\!]_G$

Y esto no es sólo teoría

neo4j

SQL/PGQ

Amazon Neptune

PuppyGraph

TigerGraph

DuckDB

GQL Standard

La complejidad de REACH

Teorema

$REACH \in P$

La complejidad de REACH

Teorema

$REACH \in P$

Ejercicio

Demuestre la proposición considerando primero el caso sin una expresión regular como entrada

La complejidad de #PATH

Teorema

#PATH es SpanL-completo bajo reducciones parsimoniosas

La complejidad de #PATH

Teorema

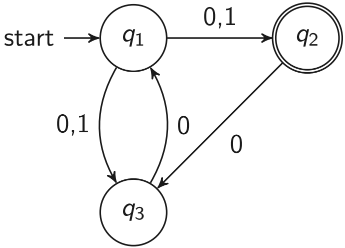
#PATH es SpanL-completo bajo reducciones parsimoniosas

Ejercicio

Demuestre que #PATH está SpanL suponiendo que la expresión regular r de entrada es dada como un NFA

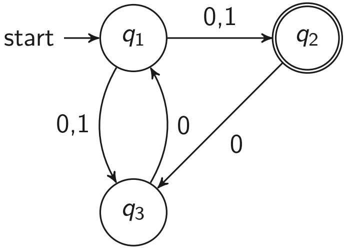
#PATH es SpanL-hard bajo reducciones parsimoniosas

Considere el siguiente NFA $\mathcal{A}$:



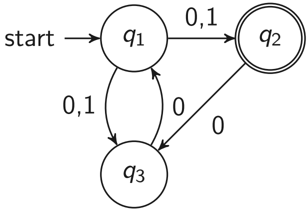
#PATH es SpanL-hard bajo reducciones parsimoniosas

Considere el siguiente NFA $\mathcal{A}$:

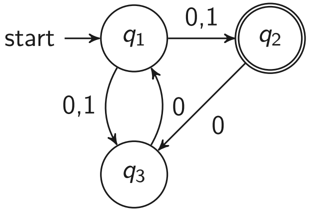


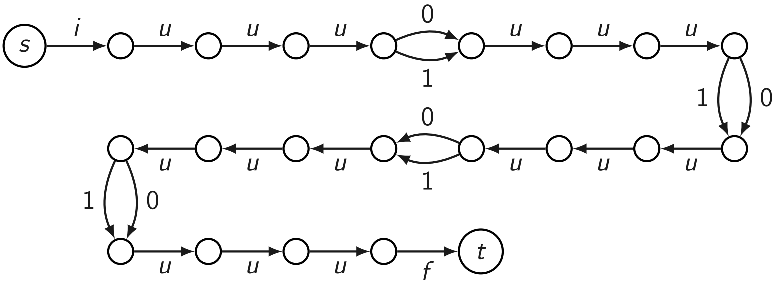
Suponga que necesitamos retornar el número de palabras de largo 4 aceptados por $\mathcal{A}$

Reducción parsimoniosa desde #NFA a #PATH

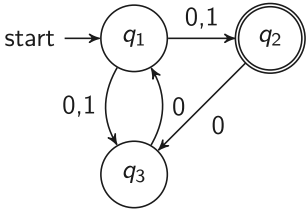


Reducción parsimoniosa desde #NFA a #PATH

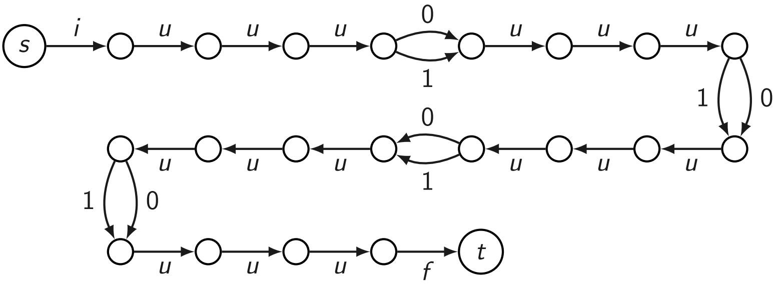




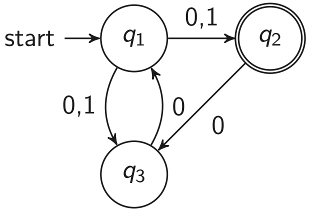
Reducción parsimoniosa desde #NFA a #PATH



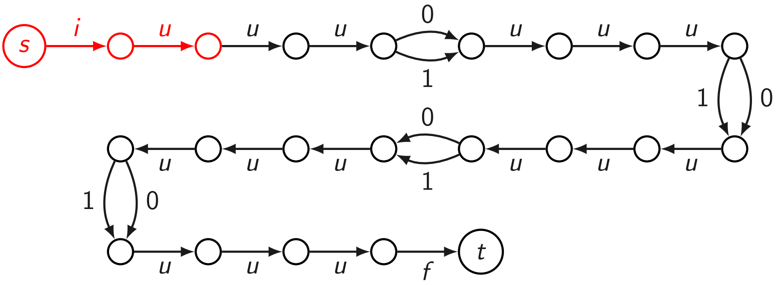
q_j es el estado inicial: $\quad i/u^j$



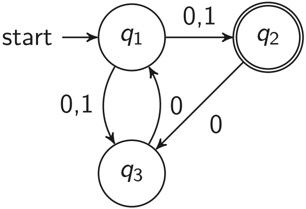
Reducción parsimoniosa desde #NFA a #PATH



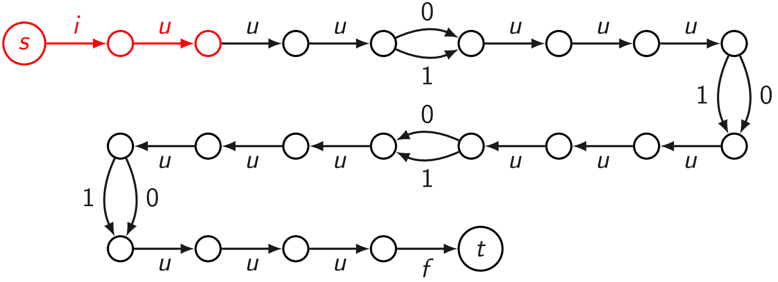
q_j es el estado inicial: $\quad i/u^j$



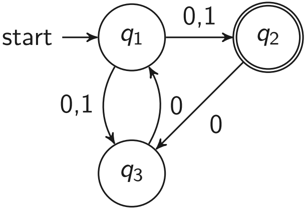
Reducción parsimoniosa desde #NFA a #PATH



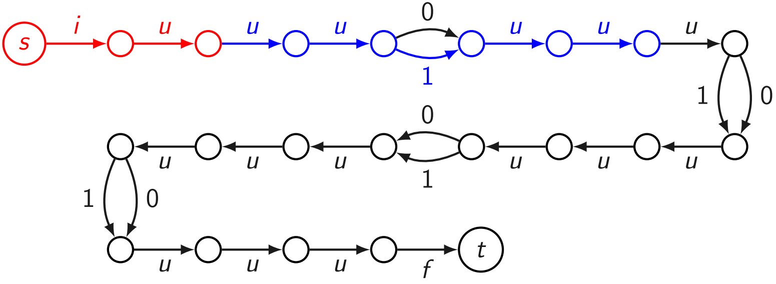
q_j es el estado inicial:	i/u^j
(q_j, a, q_k) es una transición:	$u^{3-j}/a/u^k$



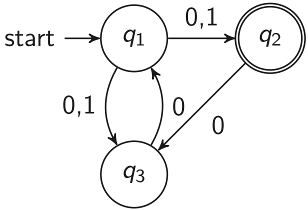
Reducción parsimoniosa desde #NFA a #PATH



q_j es el estado inicial:	i/u^j
(q_j, a, q_k) es una transición:	$u^{3-j}/a/u^k$

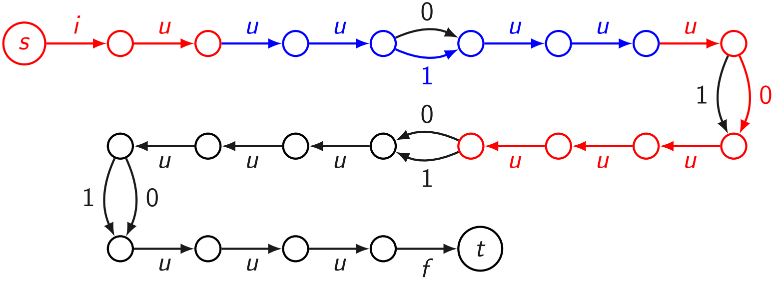


Reducción parsimoniosa desde #NFA a #PATH

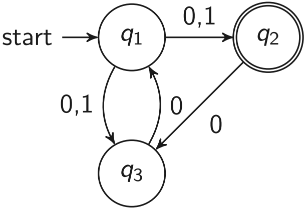


q_j es el estado inicial: i/u^j

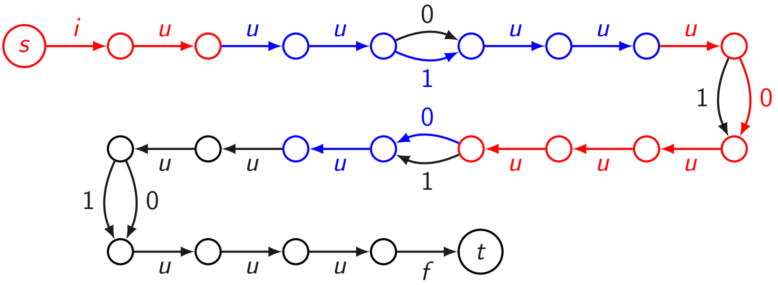
(q_j, a, q_k) es una transición: $u^{3-j}/a/u^k$



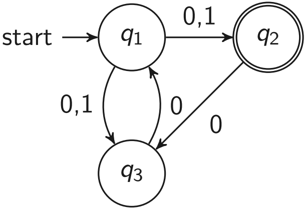
Reducción parsimoniosa desde #NFA a #PATH



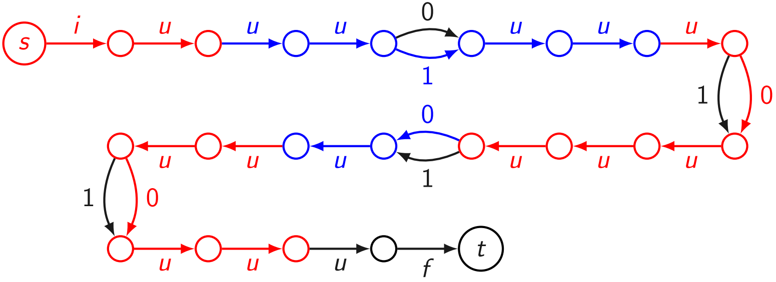
q_j es el estado inicial:	i/u^j
(q_j, a, q_k) es una transición:	$u^{3-j}/a/u^k$



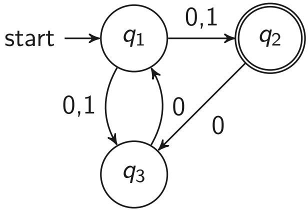
Reducción parsimoniosa desde #NFA a #PATH



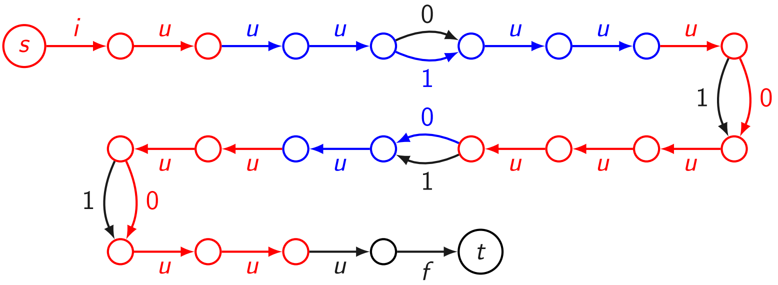
q_j es el estado inicial:	i/u^j
(q_j, a, q_k) es una transición:	$u^{3-j}/a/u^k$



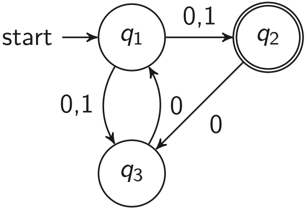
Reducción parsimoniosa desde #NFA a #PATH



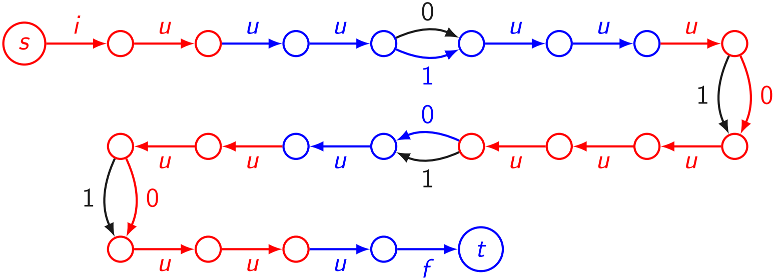
q_j es el estado inicial:	i/u^j
(q_j, a, q_k) es una transición:	$u^{3-j}/a/u^k$
q_j es el estado final:	u^{3-j}/f



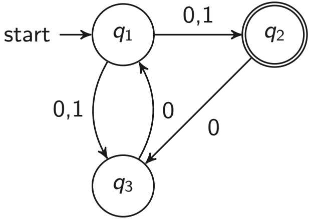
Reducción parsimoniosa desde #NFA a #PATH



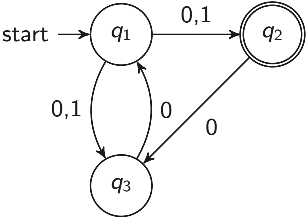
q_j es el estado inicial:	i/u^j
(q_j, a, q_k) es una transición:	$u^{3-j}/a/u^k$
q_j es el estado final:	u^{3-j}/f



Reducción parsimoniosa desde #NFA a #PATH

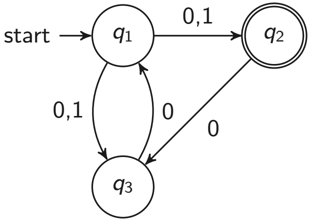


Reducción parsimoniosa desde #NFA a #PATH



Tenemos que $r = (i/u + u/u/1/u/u/ + u/0/u/u/u + 0/u + \ldots + u/f)^*$

Reducción parsimoniosa desde #NFA a #PATH



Tenemos que $r = (i/u + u/u/1/u/u/ + u/0/u/u/u + 0/u + \ldots + u/f)^*$

Número de palabras de largo 4 aceptado por $\mathcal{A}$

$=$

Número de caminos $p \in [\![r]\!]_G$ desde s hasta t de largo $21 = (5 \times 3 + 4 + 2)$

Aproximando #NFA: una segunda motivación

Como vamos a demostrar que #NFA admite un FPRAS, vamos a concluir que #PATH admite un FPRAS

Aproximando #NFA: una segunda motivación

Como vamos a demostrar que #NFA admite un FPRAS, vamos a concluir que #PATH admite un FPRAS

Tenemos un algoritmo de aproximación eficiente para un problema fundamental en bases de datos de grafos

Aproximando #NFA: una segunda motivación

Como vamos a demostrar que #NFA admite un FPRAS, vamos a concluir que #PATH admite un FPRAS

Tenemos un algoritmo de aproximación eficiente para un problema fundamental en bases de datos de grafos

- Este resultado es una segunda motivación para encontrar un FPRAS para #NFA