Counting the Solutions to a Query

Marcelo Arenas

PUC & IMFD Chile

EDBT/ICDT 2022 Joint Conference

Motivation: query answering

- Three fundamental associated problems: enumeration, uniform generation and counting of solutions

Motivation: query answering

- Three fundamental associated problems: enumeration, uniform generation and counting of solutions

- Study these three problems together

Motivation: query answering

- Three fundamental associated problems: enumeration, uniform generation and counting of solutions

- Study these three problems together
  - In line with [JVV86]

Motivation: query answering

- Three fundamental associated problems: enumeration, uniform generation and counting of solutions
- Study these three problems together
  - In line with [JVV86]
- Define classes (of queries) that have good properties in terms of these three problems

Motivation: counting

- The construction of these classes required the solution to a fundamental counting problem for non-deterministic finite automata

Motivation: counting

- The construction of these classes required the solution to a fundamental counting problem for non-deterministic finite automata
- The motivating scenario comes from graph databases: counting paths conforming to a regular expression

In this talk

- Present the techniques used to solve the aforementioned counting problem

In this talk

- Present the techniques used to solve the aforementioned counting problem
- Show how these techniques can be generalized to tree automata

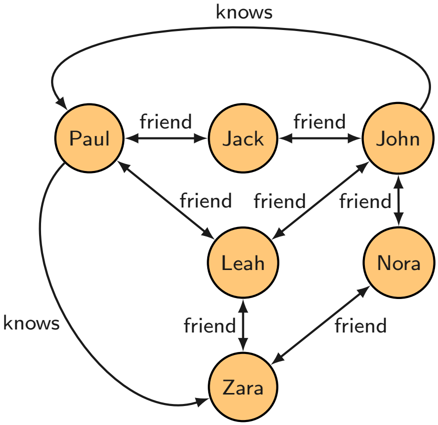
In this talk

- Present the techniques used to solve the aforementioned counting problem
- Show how these techniques can be generalized to tree automata
  - The motivating scenario comes from relational databases: counting the number of answers to an acyclic conjunctive query

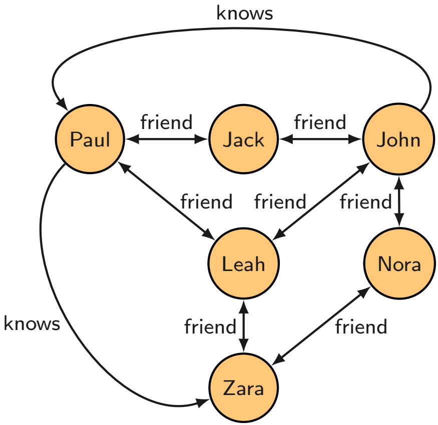
Definition of the setting

Our first motivating scenario: graph databases

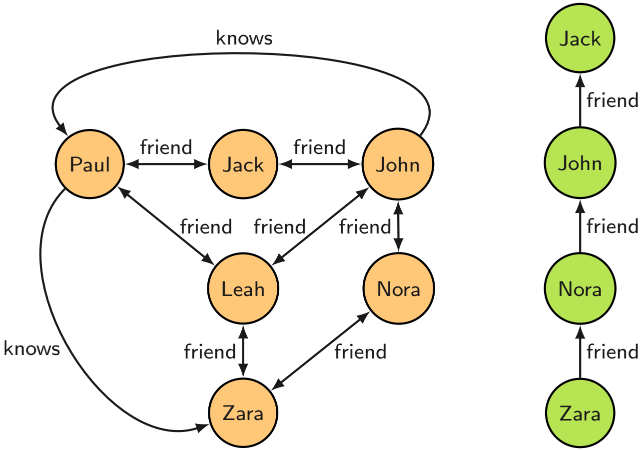
G :



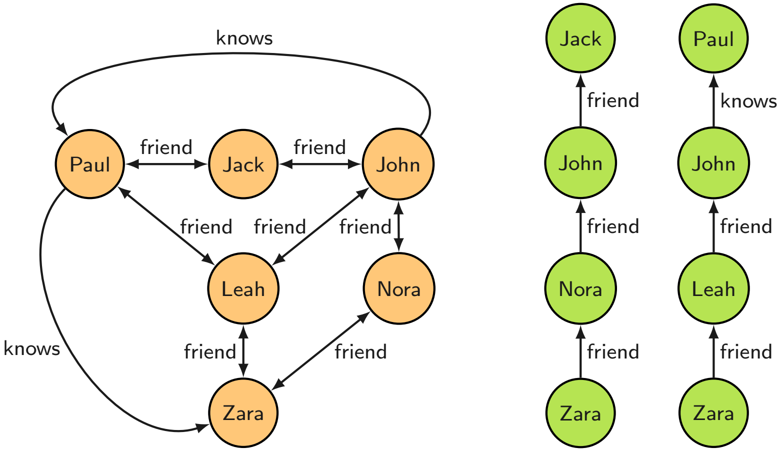
A query over G: (friend + knows)*



A query over G: $(\text{friend} + \text{knows})^*$



A query over G: (friend + knows)*



Two fundamental problems

- COUNT: count the number of paths p in G such that p conforms to regular expression r and the length of p is n

Two fundamental problems

- COUNT: count the number of paths p in G such that p conforms to regular expression r and the length of p is n

- GEN: generate uniformly at random a path p in G such that p conforms to r and the length of p is n

Is COUNT a difficult problem?

Not surprisingly the answer is yes

We would like to give a precise characterization of the complexity of this problem

Counting complexity classes

Counting complexity classes

- #P : Count the number of witnesses for a problem in NP

Counting complexity classes

- #P : Count the number of witnesses for a problem in NP
- SpanP : Count the number of distinct outputs of an NP-transducer
  - Example: given as input a graph G, count the number of subgraphs G' of G such that G' is 3-colorable

Counting complexity classes

- #P : Count the number of witnesses for a problem in NP
- SpanP : Count the number of distinct outputs of an NP-transducer
  - Example: given as input a graph G, count the number of subgraphs G' of G such that G' is 3-colorable
  - $\#P \subseteq SpanP$

Counting complexity classes

- #P : Count the number of witnesses for a problem in NP
- SpanP : Count the number of distinct outputs of an NP-transducer
  - Example: given as input a graph G, count the number of subgraphs G' of G such that G' is 3-colorable
  - $\#P \subseteq$ SpanP
- SpanL : Count the number of distinct outputs of an NL-transducer

Counting complexity classes

- $\#$P : Count the number of witnesses for a problem in NP
- SpanP : Count the number of distinct outputs of an NP-transducer
  - Example: given as input a graph G, count the number of subgraphs G' of G such that G' is 3-colorable
  - $\#\mathrm{P} \subseteq \mathrm{SpanP}$
- SpanL : Count the number of distinct outputs of an NL-transducer
  - $\#$NFA: given as input an NFA A and a length n, count the number of words of length n accepted by A

#NFA is a hard problem [AJ93]

- SpanL is contained in #P
- SpanL is a hard class: if every function in SpanL can be computed in polynomial time, then P = NP
- #NFA is SpanL-complete under parsimonious reductions

COUNT is SpanL-complete

Parsimonious reduction from #NFA to COUNT

COUNT is SpanL-complete

Parsimonious reduction from #NFA to COUNT

- Interestingly, the regular expressions used in the reduction *are not far from* the ones used in practice [BMT20]

COUNT is SpanL-complete

Parsimonious reduction from #NFA to COUNT

- Interestingly, the regular expressions used in the reduction *are not far from* the ones used in practice [BMT20]

COUNT and #NFA are equivalent problems

- In particular, in terms of the existence of efficient approximation algorithms

Existence of an efficient approximation algorithm for COUNT

SpanL-hardness of COUNT does not preclude the existence of such an efficient approximation algorithm

Existence of an efficient approximation algorithm for COUNT

SpanL-hardness of COUNT does not preclude the existence of such an efficient approximation algorithm

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for COUNT

Existence of an efficient approximation algorithm for COUNT

SpanL-hardness of COUNT does not preclude the existence of such an efficient approximation algorithm

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for COUNT

This is equivalent to constructing an FPRAS for #NFA

Existence of an efficient approximation algorithm for COUNT

SpanL-hardness of COUNT does not preclude the existence of such an efficient approximation algorithm

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for COUNT

This is equivalent to constructing an FPRAS for #NFA

- Best known approximation algorithm for #NFA worked in quasi-polynomial time [KSM95]

What about uniform generation of paths?

If we have an FPRAS for COUNT, then it can be obtained an efficient approximation algorithm for GEN

What about uniform generation of paths?

If we have an FPRAS for COUNT, then it can be obtained an efficient approximation algorithm for GEN

- A fully polynomial-time almost-uniform generator [JVV86]

Questions?

The main ideas behind the solution

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for #NFA

The definition of #NFA:

Input	:	An NFA A over the alphabet $\{0, 1\}$ and a length n (given in unary)
Output	:	Number of words w such that $w \in \mathcal{L}(A)$ and $\lvert w \rvert = n$

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for #NFA

The definition of #NFA:

Input	:	An NFA A over the alphabet $\{0, 1\}$ and a length n (given in unary)
Output	:	Number of words w such that $w \in \mathcal{L}(A)$ and $\lvert w \rvert = n$

Assume that $\mathcal{L}_n(A) = \{w \in \mathcal{L}(A) \mid \lvert w \rvert = n\}$, so that the output of #NFA is $\lvert \mathcal{L}_n(A) \rvert$

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for #NFA

The input of the approximation algorithm: A, n and $\varepsilon \in (0, 1)$

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for #NFA

The input of the approximation algorithm: A, n and $\varepsilon \in (0, 1)$

The task is to compute a number N that is a $(1 \pm \varepsilon)$-approximation of $|\mathcal{L}_n(A)|$:

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for #NFA

The input of the approximation algorithm: A, n and $\varepsilon \in (0,1)$

The task is to compute a number N that is a $(1 \pm \varepsilon)$-approximation of $|\mathcal{L}_n(A)|$:

$$(1-\varepsilon)|\mathcal{L}_n(A)| \;\leq\; N \;\leq\; (1+\varepsilon)|\mathcal{L}_n(A)|$$

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for #NFA

The input of the approximation algorithm: A, n and $\varepsilon \in (0, 1)$

The task is to compute a number N that is a $(1 \pm \varepsilon)$-approximation of $|\mathcal{L}_n(A)|$:

$$\Pr\left((1-\varepsilon)|\mathcal{L}_n(A)| \;\leq\; N \;\leq\; (1+\varepsilon)|\mathcal{L}_n(A)| \right) \quad \geq \quad \frac{3}{4}$$

Our goal is to construct a fully polynomial-time randomized approximation scheme (FPRAS) for #NFA

The input of the approximation algorithm: A, n and $\varepsilon \in (0, 1)$

The task is to compute a number N that is a $(1 \pm \varepsilon)$-approximation of $|\mathcal{L}_n(A)|$:

$$\Pr\left((1-\varepsilon)|\mathcal{L}_n(A)| \;\leq\; N \;\leq\; (1+\varepsilon)|\mathcal{L}_n(A)|\right) \quad \geq \quad \frac{3}{4}$$

Moreover, number N has to be computed in time $\text{poly}(m, n, \frac{1}{\varepsilon})$, where m is the number of states of A

Constructing an FPRAS for #NFA

Assume that $A = (Q, \{0,1\}, \Delta, I, F)$

- Q is a finite set of states
- $\Delta \subseteq Q \times \{0,1\} \times Q$ is the transition relation
- $I \subseteq Q$ is a set of initial states
- $F \subseteq Q$ is a set of final states

First component: unroll automaton A

Construct A_{unroll} from A:

- for each state $q \in Q$, include copies q^0, q^1, ..., q^n in A_{unroll}
- for each transition $(p, a, q) \in \Delta$ and $i \in \{0, 1, \dots, n-1\}$, include transition (p^i, a, q^{i+1}) in A_{unroll}

Besides, eliminate from A_{unroll} unnecessary states: each state q^i is reachable from an initial state p^0 ($p \in I$)

Second component: a sketch to be used in the estimation

Define $\mathcal{L}(q^i)$ as the set of strings w such that there is a path from an initial state p^0 to q^i labeled with w

- Notice that $|w| = i$

Besides, define for every $X \subseteq Q$:

$$\mathcal{L}(X^i) = \bigcup_{q \in X} \mathcal{L}(q^i)$$

Second component: a sketch to be used in the estimation

Define $\mathcal{L}(q^i)$ as the set of strings w such that there is a path from an initial state p^0 to q^i labeled with w

- Notice that $|w| = i$

Besides, define for every $X \subseteq Q$:

$$\mathcal{L}(X^i) = \bigcup_{q \in X} \mathcal{L}(q^i)$$

Then the task is to compute an estimation of $|\mathcal{L}(F^n)|$

Second component: a sketch to be used in the estimation

Let $\kappa = \lceil \frac{nm}{\varepsilon} \rceil$, where $m = |Q|$

Second component: a sketch to be used in the estimation

Let $\kappa = \lceil \frac{nm}{\varepsilon} \rceil$, where $m = |Q|$

We maintain for each state q^i:

- $N(q^i)$: a $(1 \pm \kappa^{-2})^i$-approximation of $|\mathcal{L}(q^i)|$
- $S(q^i)$: a multiset of uniform samples from $\mathcal{L}(q^i)$ of size $2\kappa^7$

Data structure to be inductively computed:

$$\text{sketch}[i] \quad = \quad \{N(q^j), S(q^j) \mid 0 \leq j \leq i \text{ and } q \in Q\}$$

The algorithm template

1. Construct A_{unroll} from A
2. For each state $q \in I$, set $N(q^0) = |\mathcal{L}(q^0)| = 1$ and $S(q^0) = \mathcal{L}(q^0) = \{\lambda\}$
3. For each $i = 0, \ldots, n-1$ and state $q \in Q$:
   (a) Compute $N(q^{i+1})$ given sketch[i]
   (b) Sample polynomially many uniform elements from $\mathcal{L}(q^{i+1})$ using $N(q^{i+1})$ and sketch[i], and let $S(q^{i+1})$ be the multiset of uniform samples obtained
4. Return an estimation of $|\mathcal{L}(F^n)|$ given sketch[n]

Computing an estimation $N(F^n)$ of $|\mathcal{L}(F^n)|$

We use notation $N(X^i)$ for an estimation $|\mathcal{L}(X^i)|$

Computing an estimation $N(F^n)$ of $|\mathcal{L}(F^n)|$

We use notation $N(X^i)$ for an estimation $|\mathcal{L}(X^i)|$

Such an estimation is not only needed in the last step of the algorithm, but also in the inductive construction of sketch[i]:

Computing an estimation $N(F^n)$ of $|\mathcal{L}(F^n)|$

We use notation $N(X^i)$ for an estimation $|\mathcal{L}(X^i)|$

Such an estimation is not only needed in the last step of the algorithm, but also in the inductive construction of sketch[i]:

...

3. For each $i = 0, \ldots, n-1$ and state $q \in Q$:
   (a) Compute $N(q^{i+1})$ given sketch[i]
   (b) Sample polynomially many uniform elements from $\mathcal{L}(q^{i+1})$ using $N(q^{i+1})$ and sketch[i], and let $S(q^{i+1})$ be the multiset of uniform samples obtained

...

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

Recall that $\mathcal{L}(X^i) = \bigcup_{p \in X} \mathcal{L}(p^i)$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

Recall that $\mathcal{L}(X^i) = \bigcup_{p \in X} \mathcal{L}(p^i)$

Notice that $|\mathcal{L}(X^i)| = \sum_{p \in X} |\mathcal{L}(p^i)|$ **is not true** in general

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

Recall that $\mathcal{L}(X^i) = \bigcup_{p \in X} \mathcal{L}(p^i)$

Notice that $|\mathcal{L}(X^i)| = \sum_{p \in X} |\mathcal{L}(p^i)|$ **is not true** in general

But the following holds, given a linear order $<$ on Q:

$$|\mathcal{L}(X^i)| \quad = \quad \sum_{p \in X} \Big|\mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i)\Big|$$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

We have that:

$$|\mathcal{L}(X^i)| = \sum_{p \in X} \left| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right|$$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

We have that:

$$\begin{aligned} |\mathcal{L}(X^i)| &= \sum_{p \in X} \Big| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \Big| \\ &= \sum_{p \in X} \big| \mathcal{L}(p^i) \big| \frac{\big| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \big|}{\big| \mathcal{L}(p^i) \big|} \end{aligned}$$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

We have that:

$$
\begin{aligned}
|\mathcal{L}(X^i)| &= \sum_{p \in X} \Big|\mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q<p} \mathcal{L}(q^i)\Big| \\
&= \sum_{p \in X} \big|\mathcal{L}(p^i)\big| \frac{\big|\mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q<p} \mathcal{L}(q^i)\big|}{\big|\mathcal{L}(p^i)\big|}
\end{aligned}
$$

So we will use the following approximation:

$$
\sum_{p \in X}
$$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

We have that:

$$
\begin{aligned}
|\mathcal{L}(X^i)| &= \sum_{p \in X} \left| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right| \\
&= \sum_{p \in X} \left| \mathcal{L}(p^i) \right| \frac{\left| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right|}{\left| \mathcal{L}(p^i) \right|}
\end{aligned}
$$

So we will use the following approximation:

$$
\sum_{p \in X} N(p^i)
$$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

We have that:

$$\begin{aligned} |\mathcal{L}(X^i)| &= \sum_{p \in X} \Big| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \Big| \\ &= \sum_{p \in X} \big|\mathcal{L}(p^i)\big| \frac{\big|\mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q<p} \mathcal{L}(q^i)\big|}{\big|\mathcal{L}(p^i)\big|} \end{aligned}$$

So we will use the following approximation:

$$\sum_{p \in X} N(p^i) \frac{\big|S(p^i) \setminus \bigcup_{q \in X \,:\, q<p} \mathcal{L}(q^i)\big|}{\big|S(p^i)\big|}$$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

We have that:

$$
\begin{aligned}
|\mathcal{L}(X^i)| &= \sum_{p \in X} \left| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right| \\
&= \sum_{p \in X} \left| \mathcal{L}(p^i) \right| \frac{\left| \mathcal{L}(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right|}{\left| \mathcal{L}(p^i) \right|}
\end{aligned}
$$

So we will use the following approximation:

$$
N(X^i) = \sum_{p \in X} N(p^i) \frac{\left| S(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right|}{\left| S(p^i) \right|}
$$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

$N(X^i)$ can be computed in polynomial time in the size of sketch[i]

- $S(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i)$ is constructed by checking for each $w \in S(p^i)$ whether w is not in $\mathcal{L}(q^i)$ for every $q \in X$ with $q < p$

Computing an estimation $N(X^i)$ of $|\mathcal{L}(X^i)|$

$N(X^i)$ can be computed in polynomial time in the size of sketch[i]

- $S(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i)$ is constructed by checking for each $w \in S(p^i)$ whether w is not in $\mathcal{L}(q^i)$ for every $q \in X$ with $q < p$

What guarantees that $N(X^i)$ is a good estimation of $|\mathcal{L}(X^i)|$?

The main property to maintain

$\mathcal{E}(i)$ holds if for every $p \in Q$ and $X \subseteq Q$:

$$\left| \frac{\left|\mathcal{L}(p^i) \setminus \bigcup_{q \in X} \mathcal{L}(q^i)\right|}{\left|\mathcal{L}(p^i)\right|} - \frac{\left|S(p^i) \setminus \bigcup_{q \in X} \mathcal{L}(q^i)\right|}{\left|S(p^i)\right|} \right| < \frac{1}{\kappa^3}$$

The use of the main property

...

3. For each $i = 0, \ldots, n-1$ and state $q \in Q$:
   (a) Compute $N(q^{i+1})$ given sketch[i]
   (b) Sample polynomially many uniform elements from $\mathcal{L}(q^{i+1})$ using $N(q^{i+1})$ and sketch[i], and let $S(q^{i+1})$ be the multiset of uniform samples obtained

...

The use of the main property

...

3. For each $i = 0, \ldots, n-1$ and state $q \in Q$:
   (a) Compute $N(q^{i+1})$ given sketch$[i]$
   (b) Sample polynomially many uniform elements from $\mathcal{L}(q^{i+1})$ using $N(q^{i+1})$ and sketch$[i]$, and let $S(q^{i+1})$ be the multiset of uniform samples obtained

...

Proposition

If $\mathcal{E}(i)$ holds and $N(p^i)$ is a $(1 \pm \kappa^{-2})^i$-approximation of $|\mathcal{L}(p^i)|$ for every $p \in Q$, then $N(X^i)$ is a $(1 \pm \kappa^{-2})^{i+1}$-approximation of $|\mathcal{L}(X^i)|$ for every $X \subseteq Q$

The use of the main property

$\mathcal{E}(0)$ holds and $N(p^0)$ is a $(1 \pm \kappa^{-2})^0$-approximation of $|\mathcal{L}(p^0)|$ for every $p \in Q$

- Recall that $N(p^0) = |\mathcal{L}(p^0)|$ and $S(p^0) = \mathcal{L}(p^0)$ for every $p \in Q$

The use of the main property

$\mathcal{E}(0)$ holds and $N(p^0)$ is a $(1 \pm \kappa^{-2})^0$-approximation of $|\mathcal{L}(p^0)|$ for every $p \in Q$

- Recall that $N(p^0) = |\mathcal{L}(p^0)|$ and $S(p^0) = \mathcal{L}(p^0)$ for every $p \in Q$

Then $N(X^0)$ is a $(1 \pm \kappa^{-2})$-approximation of $|\mathcal{L}(X^0)|$ for every $X \subseteq Q$

The use of the main property

$\mathcal{E}(0)$ holds and $N(p^0)$ is a $(1 \pm \kappa^{-2})^0$-approximation of $|\mathcal{L}(p^0)|$ for every $p \in Q$

- Recall that $N(p^0) = |\mathcal{L}(p^0)|$ and $S(p^0) = \mathcal{L}(p^0)$ for every $p \in Q$

Then $N(X^0)$ is a $(1 \pm \kappa^{-2})$-approximation of $|\mathcal{L}(X^0)|$ for every $X \subseteq Q$

- We want to use the values $N(X^0)$ to estimate the values $N(p^1)$

The use of the main property

For $p \in Q$, define:

$$
\begin{aligned}
Y &= \{q^0 \mid (q^0, 0, p^1) \text{ is a transition in } A_{unroll}\} \\
Z &= \{q^0 \mid (q^0, 1, p^1) \text{ is a transition in } A_{unroll}\}
\end{aligned}
$$

The use of the main property

For $p \in Q$, define:

$$\begin{aligned} Y &= \{q^0 \mid (q^0, 0, p^1) \text{ is a transition in } A_{unroll}\} \\ Z &= \{q^0 \mid (q^0, 1, p^1) \text{ is a transition in } A_{unroll}\} \end{aligned}$$

Then $\mathcal{L}(p^1) = \mathcal{L}(Y) \cdot \{0\} \uplus \mathcal{L}(Z) \cdot \{1\}$

The use of the main property

For $p \in Q$, define:

$$Y = \{q^0 \mid (q^0, 0, p^1) \text{ is a transition in } A_{unroll}\}$$
$$Z = \{q^0 \mid (q^0, 1, p^1) \text{ is a transition in } A_{unroll}\}$$

Then $\mathcal{L}(p^1) = \mathcal{L}(Y) \cdot \{0\} \uplus \mathcal{L}(Z) \cdot \{1\}$

- So that $|\mathcal{L}(p^1)| = |\mathcal{L}(Y)| + |\mathcal{L}(Z)|$

The use of the main property

For $p \in Q$, define:

$$\begin{array}{rcl} Y & = & \{q^0 \mid (q^0, 0, p^1) \text{ is a transition in } A_{unroll}\} \\ Z & = & \{q^0 \mid (q^0, 1, p^1) \text{ is a transition in } A_{unroll}\} \end{array}$$

Then $\mathcal{L}(p^1) = \mathcal{L}(Y) \cdot \{0\} \uplus \mathcal{L}(Z) \cdot \{1\}$

- So that $|\mathcal{L}(p^1)| = |\mathcal{L}(Y)| + |\mathcal{L}(Z)|$

Hence, given that $N(Y)$ is a $(1 \pm \kappa^{-2})$-approximation of $|\mathcal{L}(Y)|$ and $N(Z)$ is a $(1 \pm \kappa^{-2})$-approximation of $|\mathcal{L}(Z)|$:

$$N(Y) + N(Z) \text{ is a } (1 \pm \kappa^{-2})\text{-approximation of } N(p^1)$$

The use of the main property: a summary

$\mathcal{E}(0)$ holds and $N(p^0)$ is a $(1 \pm \kappa^{-2})^0$-approximation of $|\mathcal{L}(p^0)|$ for every $p \in Q$

The use of the main property: a summary

$\mathcal{E}(0)$ holds and $N(p^0)$ is a $(1 \pm \kappa^{-2})^0$-approximation of $|\mathcal{L}(p^0)|$ for every $p \in Q$

$$\Downarrow$$

$N(X^0)$ is a $(1 \pm \kappa^{-2})^1$-approximation of $|\mathcal{L}(X^0)|$ for every $X \subseteq Q$

The use of the main property: a summary

$\mathcal{E}(0)$ holds and $N(p^0)$ is a $(1 \pm \kappa^{-2})^0$-approximation of $|\mathcal{L}(p^0)|$ for every $p \in Q$

$\Downarrow$

$N(X^0)$ is a $(1 \pm \kappa^{-2})^1$-approximation of $|\mathcal{L}(X^0)|$ for every $X \subseteq Q$

$\Downarrow$

$N(p^1) = N(R_0(p^1)) + N(R_1(p^1))$ is a $(1 \pm \kappa^{-2})^1$-approximation of $N(p^1)$ for every $p \in Q$

where $R_b(p^1) = \{q^0 \mid (q^0, b, p^1) \text{ is a transition in } A_{unroll}\}$

The use of the main property: a summary

$N(p^1)$ is a $(1 \pm \kappa^{-2})^1$-approximation of $|\mathcal{L}(p^1)|$ for every $p \in Q$

The use of the main property: a summary

$\mathcal{E}(1)$ holds and $N(p^1)$ is a $(1 \pm \kappa^{-2})^1$-approximation of $|\mathcal{L}(p^1)|$ for every $p \in Q$

The use of the main property: a summary

$\mathcal{E}(1)$ holds and $N(p^1)$ is a $(1 \pm \kappa^{-2})^1$-approximation of $|\mathcal{L}(p^1)|$ for every $p \in Q$

$$\Downarrow$$

$N(X^1)$ is a $(1 \pm \kappa^{-2})^2$-approximation of $|\mathcal{L}(X^1)|$ for every $X \subseteq Q$

The use of the main property: a summary

$\mathcal{E}(1)$ holds and $N(p^1)$ is a $(1 \pm \kappa^{-2})^1$-approximation of $|\mathcal{L}(p^1)|$ for every $p \in Q$

$\Downarrow$

$N(X^1)$ is a $(1 \pm \kappa^{-2})^2$-approximation of $|\mathcal{L}(X^1)|$ for every $X \subseteq Q$

$\Downarrow$

$N(p^2) = N(R_0(p^2)) + N(R_1(p^2))$ is a $(1 \pm \kappa^{-2})^2$-approximation of $N(p^2)$ for every $p \in Q$

where $R_b(p^2) = \{q^1 \mid (q^1, b, p^2) \text{ is a transition in } A_{unroll}\}$

The final result

Proposition

If $\mathcal{E}(i)$ *holds for every* $i \in \{0, 1, \ldots, n\}$, *then* $N(F^n)$ *is a* $(1 \pm \varepsilon)$*-approximation of* $|\mathcal{L}(F^n)|$

Questions?

How can we maintain property $\mathcal{E}(i)$?

Sampling from a state

We need to construct the multiset $S(q^{i+1})$ of uniform samples

Recall that:

- $S(q^{i+1})$ contains $2\kappa^7$ words from $\mathcal{L}(q^{i+1})$
- $S(q^{i+1})$ is computed assuming that $N(q^{i+1})$ and $\text{sketch}[i] = \{N(q^j), S(q^j) \mid 0 \leq j \leq i\}$ have already been constructed

To recall

1. Construct A_{unroll} from A

2. For each state $q \in I$, set $N(q^0) = |\mathcal{L}(q^0)| = 1$ and $S(q^0) = \mathcal{L}(q^0) = \{\lambda\}$

3. For each $i = 0, \ldots, n-1$ and state $q \in Q$:

   (a) Compute $N(q^{i+1})$ given sketch[i]

   (b) Sample polynomially many uniform elements from $\mathcal{L}(q^{i+1})$ using $N(q^{i+1})$ and sketch[i], and let $S(q^{i+1})$ be the multiset of uniform samples obtained

4. Return an estimation of $|\mathcal{L}(F^n)|$ given sketch[n]

Sampling from q^{i+1}

To generate a sample in $\mathcal{L}(q^{i+1})$, we construct a sequence w^{i+1}, w^i, ..., w^1, w^0 such that

- $w^{i+1} = \lambda$
- $w^j = b_j w^{j+1}$ with $b_j \in \{0, 1\}$
- $w^0 \in \mathcal{L}(q^{i+1})$

Sampling from q^{i+1}

To generate a sample in $\mathcal{L}(q^{i+1})$, we construct a sequence w^{i+1}, w^i, ..., w^1, w^0 such that

- $w^{i+1} = \lambda$
- $w^j = b_j w^{j+1}$ with $b_j \in \{0, 1\}$
- $w^0 \in \mathcal{L}(q^{i+1})$

To choose $w^i = bw^{i+1}$, construct for $b = 0, 1$:

$$P_b \quad = \quad \{p^i \mid (p^i, b, q^{i+1}) \text{ is a transition in } A_{unroll}\}$$

Sampling from q^i

P_0 and P_1 are sets of states at layer i

We can compute $N(P_0)$ and $N(P_1)$ as follows:

$$N(X^i) = \sum_{p \in X} N(p^i) \frac{\left| S(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right|}{\left| S(p^i) \right|}$$

Sampling from q^i

P_0 and P_1 are sets of states at layer i

We can compute $N(P_0)$ and $N(P_1)$ as follows:

$$N(X^i) \quad = \quad \sum_{p \in X} N(p^i) \frac{\left| S(p^i) \setminus \bigcup_{q \in X \,:\, q < p} \mathcal{L}(q^i) \right|}{\left| S(p^i) \right|}$$

We choose $b \in \{0, 1\}$ with probability:

$$\frac{N(P_b)}{N(P_0) + N(P_1)}$$

We could have started from a set of states

The previous procedure works for every set of states P^{i+1}:

$$P_b \quad = \quad \{p^i \mid \exists r^{i+1} \in P^{i+1} : (p^i, b, r^{i+1}) \text{ is a transition in } A_{unroll}\}$$

In particular, we applied the procedure for $P^{i+1} = \{q^{i+1}\}$

We could have started from a set of states

The previous procedure works for every set of states P^{i+1}:

$$P_b \quad = \quad \{p^i \mid \exists r^{i+1} \in P^{i+1} : (p^i, b, r^{i+1}) \text{ is a transition in } A_{unroll}\}$$

In particular, we applied the procedure for $P^{i+1} = \{q^{i+1}\}$

The following recursive procedure summarizes the previous idea:

$$\textbf{Sample}(i+1, \{q^{i+1}\}, \lambda, \varphi_0)$$

It uses sets of states $P^{i+1} = \{q^{i+1}\}$, P^i, ..., P^1, P^0 and an initial probability φ_0

The sampling algorithm

Sample(j, P^j, w^j, φ)

1. If $j = 0$, then with probability φ return w^0, otherwise return **fail**
2. Compute $P_{j,b} = \{p^{j-1} \mid \exists r^j \in P^j : (p^{j-1}, b, r^j)$ is a transition in $A_{unroll}\}$ for $b = 0, 1$
3. Choose $b \in \{0, 1\}$ with probability $p_b = \dfrac{N(P_{j,b})}{N(P_{j,0}) + N(P_{j,1})}$
4. Set $P^{j-1} = P_{j,b}$ and $w^{j-1} = bw^j$
5. Return **Sample**$(j-1, P^{j-1}, w^{j-1}, \frac{\varphi}{p_b})$

The key observation

Let $x = x_1 \cdots x_{i+1}$ be a word in $\mathcal{L}(q^{i+1})$

The key observation

Let $x = x_1 \cdots x_{i+1}$ be a word in $\mathcal{L}(q^{i+1})$

We have that:

$$\begin{aligned}
&\mathbf{Pr}(\text{the output of } \textbf{Sample} \text{ is } x) \\
&\quad = \mathbf{Pr}(w^0 = x \wedge \text{ the last call to } \textbf{Sample} \text{ does not fail}) \\
&\quad = \mathbf{Pr}(\text{the last call to } \textbf{Sample} \text{ does not fail} \mid w^0 = x) \cdot \mathbf{Pr}(w^0 = x) \\
&\quad = \left(\left(\prod_{j=1}^{i+1} \frac{N(P_{j,x_j})}{N(P_{j,0}) + N(P_{j,1})} \right)^{-1} \cdot \varphi_0 \right) \cdot \left(\prod_{j=1}^{i+1} \frac{N(P_{j,x_j})}{N(P_{j,0}) + N(P_{j,1})} \right) \\
&\quad = \varphi_0
\end{aligned}$$

The value of the initial probability φ_0

Proposition

Assume that $\mathcal{E}(j)$ holds for each $j < i+1$. If w is the output of $\textbf{Sample}(i+1, \{q^{i+1}\}, \lambda, \frac{e^{-5}}{N(q^{i+1})})$, *then*

- $\varphi \in (0,1)$ *in every recursive call to* **Sample**
- $\textbf{Pr}(w = \textbf{fail}) \leq 1 - e^{-9}$
- $\textbf{Pr}(w = x) = \dfrac{e^{-5}}{N(q^{i+1})}$ *for every* $x \in \mathcal{L}(q^{i+1})$

The value of the initial probability φ_0

Proposition

Assume that $\mathcal{E}(j)$ holds for each $j < i+1$. If w is the output of **Sample**$(i+1, \{q^{i+1}\}, \lambda, \frac{e^{-5}}{N(q^{i+1})})$, *then*

- $\varphi \in (0,1)$ *in every recursive call to* **Sample**
- $\mathbf{Pr}(w = \mathbf{fail}) \leq 1 - e^{-9}$
- $\mathbf{Pr}(w = x) = \dfrac{e^{-5}}{N(q^{i+1})}$ *for every* $x \in \mathcal{L}(q^{i+1})$

Hence, conditioned on not failing, **Sample**$(i+1, \{q^{i+1}\}, \lambda, \frac{e^{-5}}{N(q^{i+1})})$ returns a uniform sample from $\mathcal{L}(q^{i+1})$

Bounding the probability of breaking the main assumption

Recall that $\mathcal{E}(i)$ holds if for every $q \in Q$ and $X \subseteq Q$:

$$\left| \frac{\left|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|\mathcal{L}(q^i)\right|} - \frac{\left|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|S(q^i)\right|} \right| < \frac{1}{\kappa^3}$$

Bounding the probability of breaking the main assumption

Recall that $\mathcal{E}(i)$ holds if for every $q \in Q$ and $X \subseteq Q$:

$$\left| \frac{\left|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|\mathcal{L}(q^i)\right|} - \frac{\left|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|S(q^i)\right|} \right| < \frac{1}{\kappa^3}$$

By using Hoeffding's inequality, it is possible to conclude that:

$$\mathbf{Pr}\left(\bigwedge_{j=0}^{n} \mathcal{E}(j) \right) \geq 1 - e^{-\kappa}$$

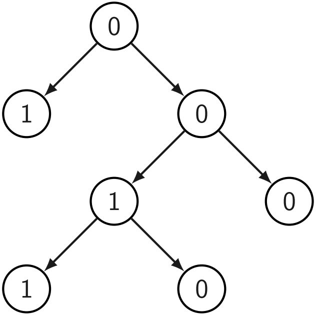
The complete algorithm: final comments

Putting all together, we obtain that the probability that the algorithm returns a wrong estimate is at most $\frac{1}{4}$

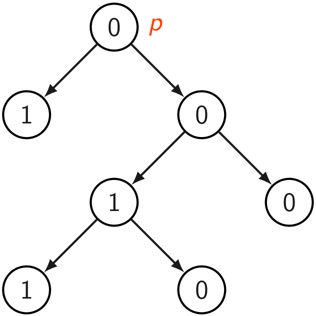
The algorithm runs in time poly$(m, n, \frac{1}{\varepsilon})$

The extension of the approach to tree automata

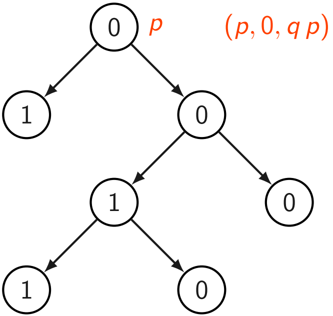
Tree automata



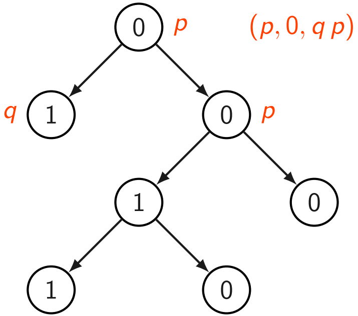
Tree automata



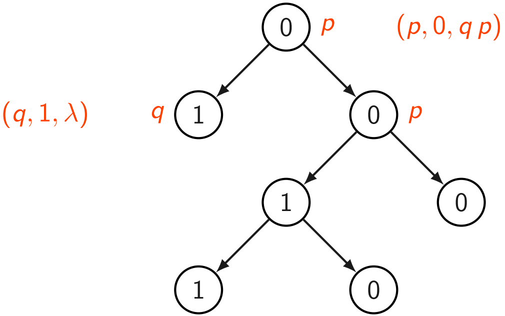
Tree automata



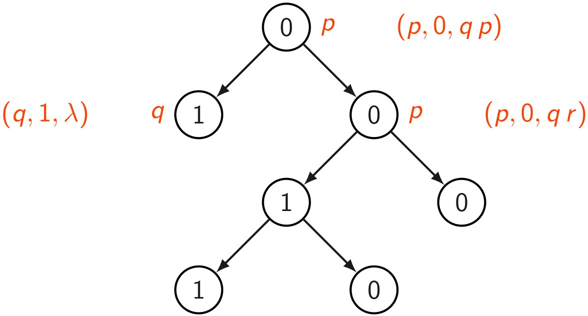
Tree automata



Tree automata



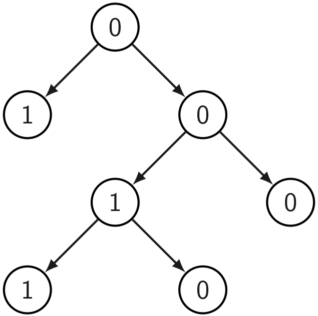
Tree automata



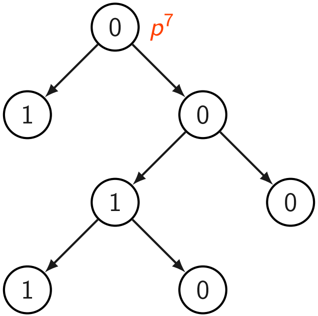
The problem $\#$TA

Input	:	A tree automaton (TA) T over the alphabet $\{0, 1\}$ and a number n (given in unary)
Output	:	Number of trees t such that $t \in \mathcal{L}(T)$ and the number of nodes of t is n

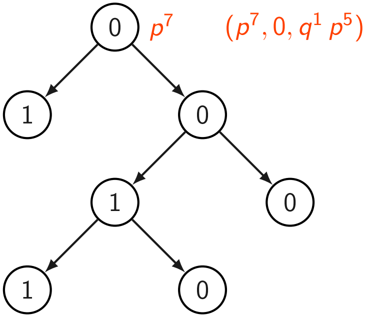
Constructing an FPRAS for #TA



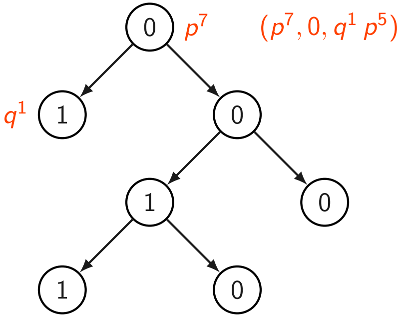
Constructing an FPRAS for #TA



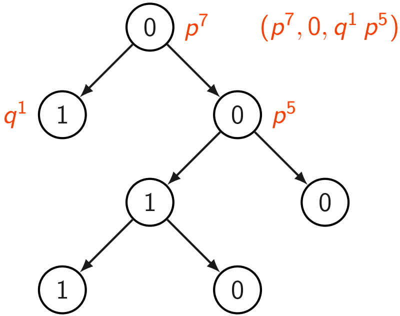
Constructing an FPRAS for #TA



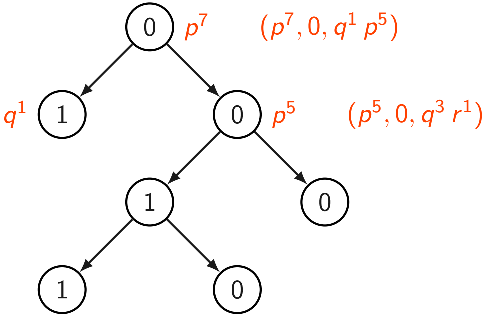
Constructing an FPRAS for #TA



Constructing an FPRAS for #TA

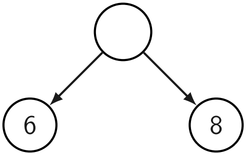


Constructing an FPRAS for #TA

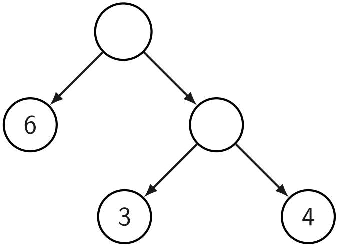


Reducing to a counting problem for *succinct* NFA

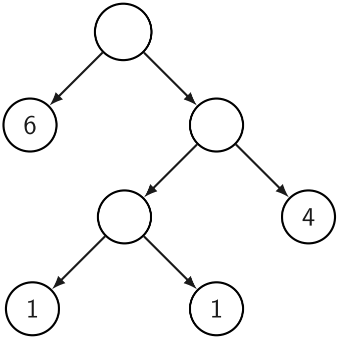
Reducing to a counting problem for *succinct* NFA



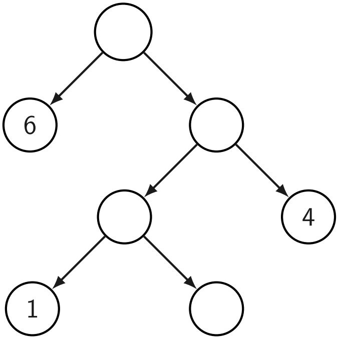
Reducing to a counting problem for *succinct* NFA



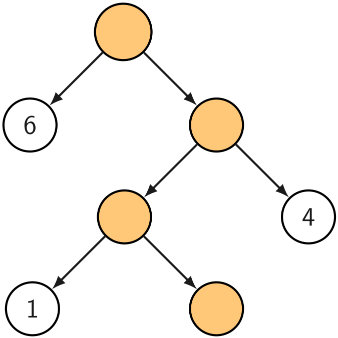
Reducing to a counting problem for *succinct* NFA



Reducing to a counting problem for *succinct* NFA



Reducing to a counting problem for *succinct* NFA



Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before
- Σ is an alphabet

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before
- Σ is an alphabet
  - Σ is assumed to be succinctly encoded via some representation

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before
- Σ is an alphabet
  - Σ is assumed to be succinctly encoded via some representation
- $\Delta \subseteq Q \times 2^{\Sigma} \times Q$ is the transition relation

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before
- Σ is an alphabet
  - Σ is assumed to be succinctly encoded via some representation
- $\Delta \subseteq Q \times 2^{\Sigma} \times Q$ is the transition relation
  - If $(p, A, q) \in \Delta$, then A is also assumed to be succinctly encoded via some representation

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before
- Σ is an alphabet
  - Σ is assumed to be succinctly encoded via some representation
- $\Delta \subseteq Q \times 2^{\Sigma} \times Q$ is the transition relation
  - If $(p, A, q) \in \Delta$, then A is also assumed to be succinctly encoded via some representation

$w_1 \cdots w_n \in \mathcal{L}(S)$ if there exists a sequence of states $q_0 q_1 \ldots q_n$ such that $q_0 \in I$, $q_n \in F$ and

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before
- Σ is an alphabet
  - Σ is assumed to be succinctly encoded via some representation
- $\Delta \subseteq Q \times 2^{\Sigma} \times Q$ is the transition relation
  - If $(p, A, q) \in \Delta$, then A is also assumed to be succinctly encoded via some representation

$w_1 \cdots w_n \in \mathcal{L}(S)$ if there exists a sequence of states $q_0 q_1 \ldots q_n$ such that $q_0 \in I$, $q_n \in F$ and for every w_i, there exists A such that $w_i \in A$ and $(q_{i-1}, A, q_i) \in \Delta$

Succinct NFA $S = (Q, \Sigma, \Delta, I, F)$

- Q, I and F as before
- Σ is an alphabet
  - Σ is assumed to be succinctly encoded via some representation
- $\Delta \subseteq Q \times 2^{\Sigma} \times Q$ is the transition relation
  - If $(p, A, q) \in \Delta$, then A is also assumed to be succinctly encoded via some representation

$w_1 \cdots w_n \in \mathcal{L}(S)$ if there exists a sequence of states $q_0 q_1 \ldots q_n$ such that $q_0 \in I$, $q_n \in F$ and for every w_i, there exists A such that $w_i \in A$ and $(q_{i-1}, A, q_i) \in \Delta$

- The length of $w_1 \cdots w_n$ is n

The main result about #Succinct-NFA

The definition of #Succinct-NFA:

Input	:	A succinct NFA S and a length n (given in unary)
Output	:	Number of words w such that $w \in \mathcal{L}(S)$ and $\lvert w \rvert = n$

The main result about #Succinct-NFA

The definition of #Succinct-NFA:

Input	:	A succinct NFA S and a length n (given in unary)
Output	:	Number of words w such that $w \in \mathcal{L}(S)$ and $|w| = n$

Theorem

#Succinct-NFA admits an FPRAS when restricted to the class of succinct NFA $S = (Q, \Sigma, \Delta, I, F)$ such that for every $(p, A, q) \in \Delta$, there exists an oracle which can:

(1) *test membership in A,*

(2) *produce an estimate of the size of $|A|$, and*

(3) *generate almost-uniform samples from A.*

The main result about #TA

Theorem

#TA admits an FPRAS

The main corollaries about query answering

Let #ACQ:

Input	:	A database D and an acyclic conjunctive query Q
Output	:	$\lvert Q(D)\rvert$

The main corollaries about query answering

Let #ACQ:

Input	:	A database D and an acyclic conjunctive query Q
Output	:	$\lvert Q(D) \rvert$

Corollary

#ACQ admits an FPRAS

The main corollaries about query answering

Given $k \geq 1$, let $\#k$-HW:

Input	:	A database D and a conjunctive query Q such that the hypertree width of Q is at most k
Output	:	$\lvert Q(D) \rvert$

The main corollaries about query answering

Given $k \geq 1$, let $\#k$-HW:

Input	:	A database D and a conjunctive query Q such that the hypertree width of Q is at most k
Output	:	$\lvert Q(D) \rvert$

Corollary

$\#k$-HW admits an FPRAS

Some final remarks

The results presented in this talk are proved in [ACJR21a] and [ACJR21b]

The results presented in this talk are proved in [ACJR21a] and [ACJR21b]

Future work:

The results presented in this talk are proved in [ACJR21a] and [ACJR21b]

Future work:

- Make algorithms practical

The results presented in this talk are proved in [ACJR21a] and [ACJR21b]

Future work:

- Make algorithms practical
  - Implementation of FPRAS for #NFA works well with smaller bounds

The results presented in this talk are proved in [ACJR21a] and [ACJR21b]

Future work:

- Make algorithms practical
  - Implementation of FPRAS for #NFA works well with smaller bounds
- Can the results be extended for semantic acyclicity?

The results presented in this talk are proved in [ACJR21a] and [ACJR21b]

Future work:

- Make algorithms practical
  - Implementation of FPRAS for #NFA works well with smaller bounds
- Can the results be extended for semantic acyclicity?
- Can the results be extended to context free grammars?

The results presented in this talk are proved in [ACJR21a] and [ACJR21b]

Future work:

- Make algorithms practical
    - Implementation of FPRAS for #NFA works well with smaller bounds
- Can the results be extended for semantic acyclicity?
- Can the results be extended to context free grammars?
    - Only quasi-polynomial time approximation are known for this problem [GJKSM97]

Questions?

Bibliography

[AJ93] C. Àlvarez, B. Jenner: A Very Hard log-Space Counting Class. Theor. Comput. Sci. 107(1): 3-30, 1993

[ACJR21a] M. Arenas, L. A. Croquevielle, R. Jayaram, C. Riveros: #NFA Admits an FPRAS: Efficient Enumeration, Counting, and Uniform Generation for Logspace Classes. J. ACM 68(6): 48:1-48:40, 2021

[ACJR21b] M. Arenas, L. A. Croquevielle, R. Jayaram, C. Riveros: When is approximate counting for conjunctive queries tractable? STOC 2021: 1015-1027

[BMT20] A. Bonifati, W. Martens, T. Timm: An analytical study of large SPARQL query logs. VLDB J. 29(2-3): 655-679, 2020

Bibliography

[GJKSM97] V. Gore, M. Jerrum, S. Kannan, Z. Sweedyk, S. R. Mahaney: A Quasi-Polynomial-Time Algorithm for Sampling Words from a Context-Free Language. Inf. Comput. 134(1): 59-74. 1997

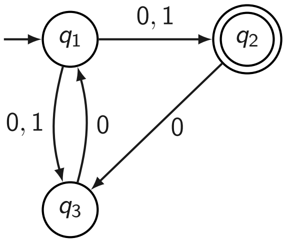
[JVV86] M. Jerrum, L. G. Valiant, V. V. Vazirani: Random Generation of Combinatorial Structures from a Uniform Distribution. Theor. Comput. Sci. 43: 169-188, 1986

[KSM95] S. Kannan, Z. Sweedyk, S. R. Mahaney: Counting and Random Generation of Strings in Regular Languages. SODA 1995: 551-557

Appendix

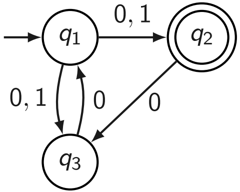
COUNT is SpanL-complete (under parsimonious reductions)

Consider the following NFA A:

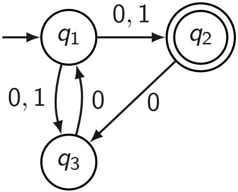


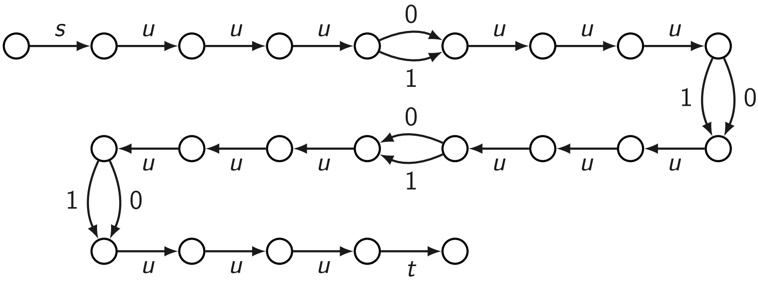
Assume we need to return the number of words of length 4 accepted by A

Reduction from #NFA to COUNT

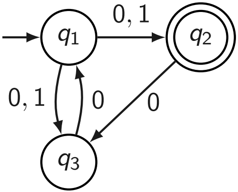


Reduction from #NFA to COUNT

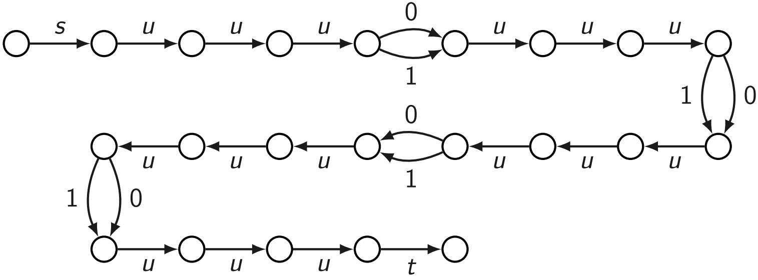




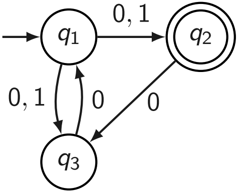
Reduction from #NFA to COUNT



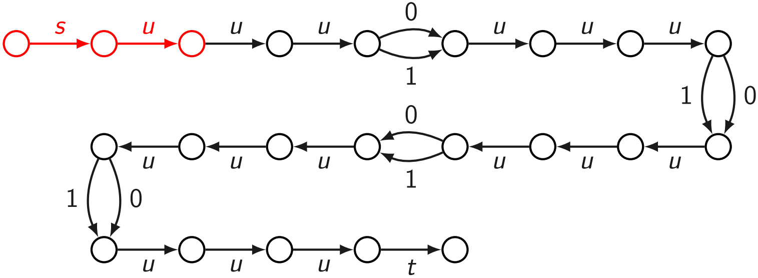
q_i is an initial state : s/u^i



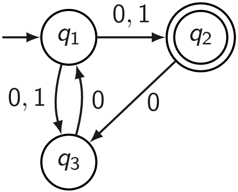
Reduction from #NFA to COUNT



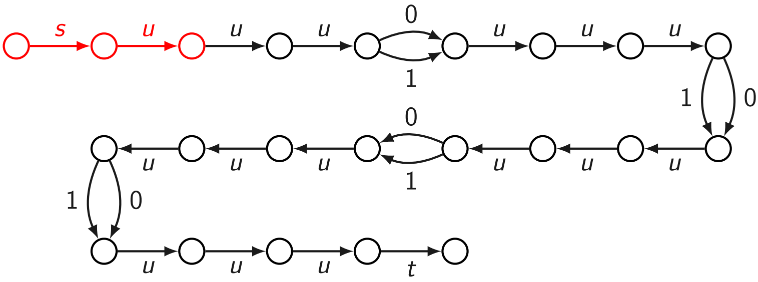
q_i is an initial state : s/u^i



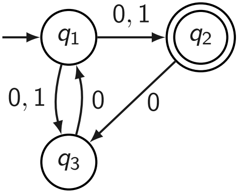
Reduction from #NFA to COUNT



q_i is an initial state	:	s/u^i
(q_i, a, q_j) is a transition	:	$u^{3-i}/a/u^j$

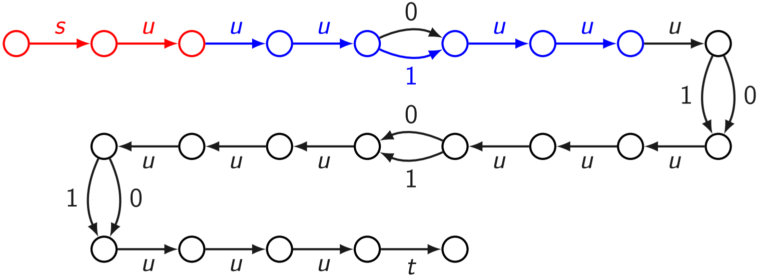


Reduction from #NFA to COUNT

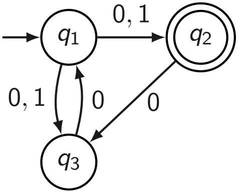


q_i is an initial state : s/u^i

(q_i, a, q_j) is a transition : $u^{3-i}/a/u^j$

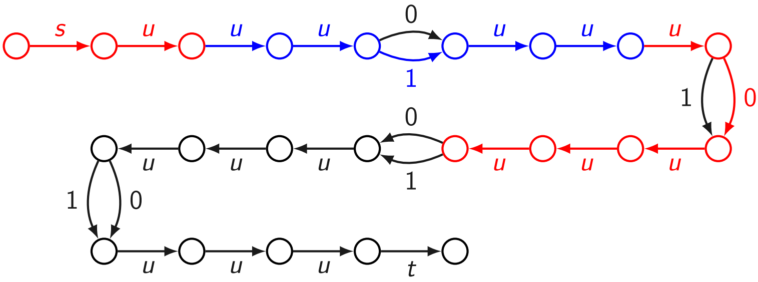


Reduction from #NFA to COUNT

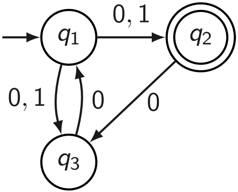


q_i is an initial state : s/u^i

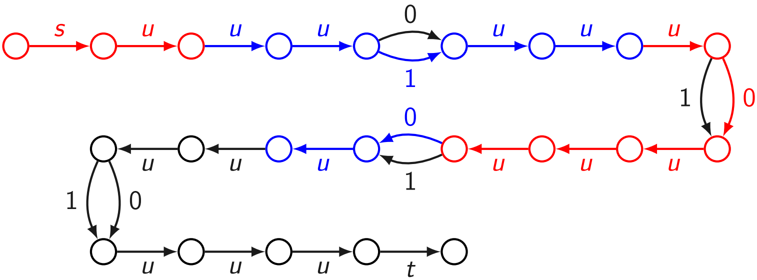
(q_i, a, q_j) is a transition : $u^{3-i}/a/u^j$



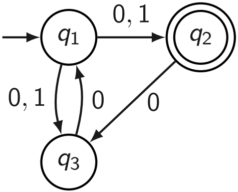
Reduction from #NFA to COUNT



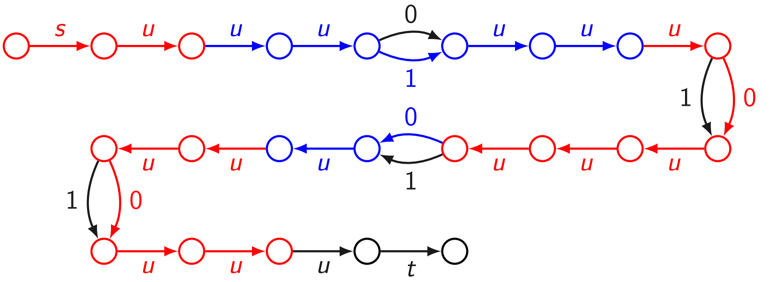
q_i is an initial state	:	s/u^i
(q_i, a, q_j) is a transition	:	$u^{3-i}/a/u^j$



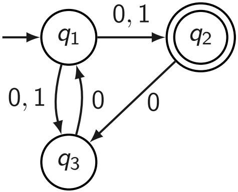
Reduction from #NFA to COUNT



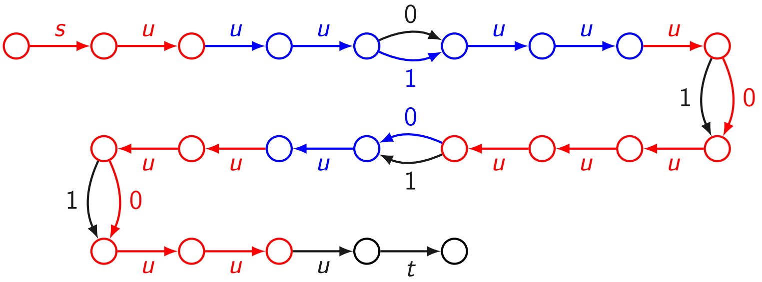
q_i is an initial state	:	s/u^i
(q_i, a, q_j) is a transition	:	$u^{3-i}/a/u^j$



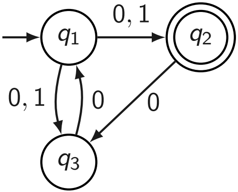
Reduction from #NFA to COUNT



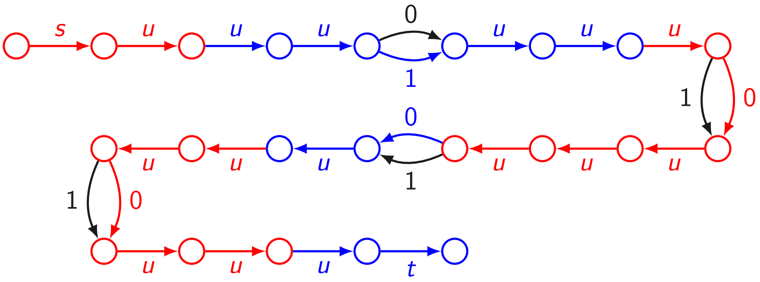
$$
\begin{array}{rcl}
q_i \text{ is an initial state} & : & s/u^i \\
(q_i, a, q_j) \text{ is a transition} & : & u^{3-i}/a/u^j \\
q_i \text{ is a final state} & : & u^{3-i}/t
\end{array}
$$



Reduction from #NFA to COUNT



q_i is an initial state	:	s/u^i
(q_i, a, q_j) is a transition	:	$u^{3-i}/a/u^j$
q_i is a final state	:	u^{3-i}/t



Reduction from #NFA to COUNT

Define $r = (s/u + u/u/1/u/u/ + u/0/u/u/u + 0/u + \ldots + u/t)^*$

Reduction from #NFA to COUNT

Define $r = (s/u + u/u/1/u/u/ + u/0/u/u/u + 0/u + \ldots + u/t)^*$

Number of words of length 4 accepted by A

=

Number of paths p in G such that p conforms to r and the length of p is $21 = (5 \times 3 + 4 + 2)$

Bounding the probability of breaking the main assumption

Recall that $\mathcal{E}(i)$ holds if for every $q \in Q$ and $X \subseteq Q$:

$$\left| \frac{\left|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|\mathcal{L}(q^i)\right|} - \frac{\left|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|S(q^i)\right|} \right| < \frac{1}{\kappa^3}$$

We know that $\mathcal{E}(0)$ holds.

Bounding the probability of breaking the main assumption

Recall that $\mathcal{E}(i)$ holds if for every $q \in Q$ and $X \subseteq Q$:

$$\left| \frac{\left|\mathcal{L}(q^i) \smallsetminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|\mathcal{L}(q^i)\right|} - \frac{\left|S(q^i) \smallsetminus \bigcup_{p \in X} \mathcal{L}(p^i)\right|}{\left|S(q^i)\right|} \right| < \frac{1}{\kappa^3}$$

We know that $\mathcal{E}(0)$ holds. We need to compute a lower bound for:

$$\mathbf{Pr}\left(\bigwedge_{j=0}^{n} \mathcal{E}(j) \right)$$

Bounding the probability of breaking $\mathcal{E}(i)$

Assume that $\bigwedge_{j=0}^{i-1} \mathcal{E}(j)$ holds

Let $q \in Q$ and $S(q^i)$ be a multiset of $2\kappa^7$ samples from $\mathcal{L}(q^i)$ computed by calling **Sample**$(i, \{q^i\}, \lambda, \frac{e^{-5}}{N(q^i)})$

- Each element of $S(q^i)$ is obtained by repeatedly calling **Sample** until the output is different from **fail**

Assume that $S(q^i) = \{w_1, \ldots, w_t\}$ with $t = 2\kappa^7$

Bounding the probability of breaking $\mathcal{E}(i)$

Let $X \subseteq Q$, and Y_i be a Bernoulli random variable for $i \in \{1, \ldots, t\}$:

$$Y_i = 1 \quad \text{if and only if} \quad w_i \in \left(\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i) \right)$$

Bounding the probability of breaking $\mathcal{E}(i)$

Let $X \subseteq Q$, and Y_i be a Bernoulli random variable for $i \in \{1, \ldots, t\}$:

$$Y_i = 1 \quad \text{if and only if} \quad w_i \in \left(\mathcal{L}(q^i) \smallsetminus \bigcup_{p \in X} \mathcal{L}(p^i)\right)$$

We have that:

$$\begin{aligned}
\mathbb{E}[Y_i] &= \frac{|\mathcal{L}(q^i) \smallsetminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|} \\
\sum_{j=1}^{t} Y_i &= |S(q^i) \smallsetminus \bigcup_{p \in X} \mathcal{L}(p^i)| \\
t &= |S(q^i)|
\end{aligned}$$

By using Hoeffding's inequality

$$\mathbf{Pr}\Bigg(\Bigg|\frac{|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|}\Bigg| \geq \frac{1}{\kappa^3} \ \Bigg| \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\Bigg)$$

By using Hoeffding's inequality

$$\mathbf{Pr}\left(\left|\frac{|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|}\right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) =$$

$$\mathbf{Pr}\left(\left|\frac{1}{t}\sum_{j=1}^{t} Y_i - \mathbb{E}\left[\frac{1}{t}\sum_{j=1}^{t} Y_i\right]\right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right)$$

By using Hoeffding's inequality

$$\mathbf{Pr}\left(\left|\frac{|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|}\right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) =$$

$$\mathbf{Pr}\left(\left|\frac{1}{t}\sum_{j=1}^{t} Y_i - \mathbb{E}\left[\frac{1}{t}\sum_{j=1}^{t} Y_i\right]\right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) \leq 2e^{-2\left(\frac{1}{\kappa^3}\right)^2 t}$$

By using Hoeffding's inequality

$$\mathbf{Pr}\Bigg(\Bigg|\frac{|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|}\Bigg| \geq \frac{1}{\kappa^3} \;\Bigg|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\Bigg) =$$

$$\mathbf{Pr}\Bigg(\Bigg|\frac{1}{t}\sum_{j=1}^{t} Y_i - \mathbb{E}\Bigg[\frac{1}{t}\sum_{j=1}^{t} Y_i\Bigg]\Bigg| \geq \frac{1}{\kappa^3} \;\Bigg|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\Bigg) \leq 2e^{-2\left(\frac{1}{\kappa^3}\right)^2 t}$$

$$= 2e^{-2\left(\frac{1}{\kappa^6}\right)2\kappa^7}$$

By using Hoeffding's inequality

$$\mathbf{Pr}\left(\left|\frac{|S(q^i) \setminus \bigcup_{p\in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p\in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|}\right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) =$$

$$\mathbf{Pr}\left(\left|\frac{1}{t}\sum_{j=1}^{t} Y_i - \mathbb{E}\left[\frac{1}{t}\sum_{j=1}^{t} Y_i\right]\right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) \leq 2e^{-2\left(\frac{1}{\kappa^3}\right)^2 t}$$

$$= 2e^{-2\left(\frac{1}{\kappa^6}\right)2\kappa^7}$$

$$= 2e^{-4\kappa}$$

By taking the union bound

$$\mathbf{Pr}\Bigg(\exists q \in Q\, \exists X \subseteq Q \left| \frac{|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|} \right| \geq \frac{1}{\kappa^3} \;\Bigg|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j) \Bigg) \leq$$

By taking the union bound

$$\mathbf{Pr}\Bigg(\exists q \in Q\, \exists X \subseteq Q \left| \frac{|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|} \right| \geq \frac{1}{\kappa^3} \;\Bigg|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j) \Bigg) \leq$$

$$\sum_{q \in Q} \sum_{X \subseteq Q} \mathbf{Pr}\Bigg(\left| \frac{|S(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p \in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|} \right| \geq \frac{1}{\kappa^3} \;\Bigg|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j) \Bigg) \leq$$

By taking the union bound

$$\mathbf{Pr}\left(\exists q \in Q\, \exists X \subseteq Q \left| \frac{|S(q^i) \setminus \bigcup_{p\in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p\in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|} \right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) \leq$$

$$\sum_{q\in Q} \sum_{X\subseteq Q} \mathbf{Pr}\left(\left| \frac{|S(q^i) \setminus \bigcup_{p\in X} \mathcal{L}(p^i)|}{|S(q^i)|} - \frac{|\mathcal{L}(q^i) \setminus \bigcup_{p\in X} \mathcal{L}(p^i)|}{|\mathcal{L}(q^i)|} \right| \geq \frac{1}{\kappa^3} \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) \leq$$

$$m2^m 2e^{-4\kappa} \leq \kappa 2^\kappa 2e^{-4\kappa} \leq 2e^{-2\kappa}$$

The conclusion

Rewriting the previous result:

$$\mathbf{Pr}\left(\mathcal{E}(i) \;\middle|\; \bigwedge_{j=0}^{i-1} \mathcal{E}(j)\right) \;\geq\; 1 - e^{-2\kappa}$$

We conclude that:

$$\mathbf{Pr}\left(\bigwedge_{j=0}^{n} \mathcal{E}(j)\right) \;\geq\; 1 - e^{-\kappa}$$

The complete algorithm

Input: NFA $A = (Q, \{0,1\}, \Delta, I, F)$ with $m = |Q|$, length n given in unary and error $\varepsilon \in (0,1)$

The complete algorithm

Input: NFA $A = (Q, \{0,1\}, \Delta, I, F)$ with $m = |Q|$, length n given in unary and error $\varepsilon \in (0,1)$

1. If $\mathcal{L}_n(A) = \emptyset$, then return 0

The complete algorithm

Input: NFA $A = (Q, \{0,1\}, \Delta, I, F)$ with $m = |Q|$, length n given in unary and error $\varepsilon \in (0,1)$

1. If $\mathcal{L}_n(A) = \emptyset$, then return 0
2. Construct A_{unroll} and set $\kappa = \lceil \frac{nm}{\varepsilon} \rceil$

The complete algorithm

Input: NFA $A = (Q, \{0,1\}, \Delta, I, F)$ with $m = |Q|$, length n given in unary and error $\varepsilon \in (0,1)$

1. If $\mathcal{L}_n(A) = \emptyset$, then return 0
2. Construct A_{unroll} and set $\kappa = \lceil \frac{nm}{\varepsilon} \rceil$
3. Remove each state q^i from A_{unroll} that is not reachable from an initial state in I^0

The complete algorithm

Input: NFA $A = (Q, \{0,1\}, \Delta, I, F)$ with $m = |Q|$, length n given in unary and error $\varepsilon \in (0,1)$

1. If $\mathcal{L}_n(A) = \emptyset$, then return 0
2. Construct A_{unroll} and set $\kappa = \lceil \frac{nm}{\varepsilon} \rceil$
3. Remove each state q^i from A_{unroll} that is not reachable from an initial state in I^0
4. For each $q^0 \in I^0$, set $N(q^0) = 1$ and $S(q^0) = \{\lambda\}$

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

    5.1 Set $R_b = \{p^{i-1} \mid (p^{i-1}, b, q^i) \text{ is a transition in } A_{unroll}\}$ for $b = 0, 1$

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

    5.1 Set $R_b = \{p^{i-1} \mid (p^{i-1}, b, q^i) \text{ is a transition in } A_{unroll}\}$ for $b = 0, 1$

    5.2 Set $N(q^i) = N(R_0) + N(R_1)$

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

   5.1 Set $R_b = \{p^{i-1} \mid (p^{i-1}, b, q^i) \text{ is a transition in } A_{unroll}\}$ for $b = 0, 1$

   5.2 Set $N(q^i) = N(R_0) + N(R_1)$

   5.3 Set $S(q^i) = \emptyset$. Then while $|S(q^i)| < 2\kappa^7$:

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

   5.1 Set $R_b = \{p^{i-1} \mid (p^{i-1}, b, q^i)$ is a transition in $A_{unroll}\}$ for $b = 0, 1$

   5.2 Set $N(q^i) = N(R_0) + N(R_1)$

   5.3 Set $S(q^i) = \emptyset$. Then while $|S(q^i)| < 2\kappa^7$:

      5.3.1 Run **Sample**$(i, \{q^i\}, \lambda, \frac{e^{-5}}{N(q^i)})$ until it returns $w \neq$ **fail**, and at most $c(\kappa) \in \Theta(\log(\kappa))$ times

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

   5.1 Set $R_b = \{p^{i-1} \mid (p^{i-1}, b, q^i) \text{ is a transition in } A_{unroll}\}$ for $b = 0, 1$

   5.2 Set $N(q^i) = N(R_0) + N(R_1)$

   5.3 Set $S(q^i) = \emptyset$. Then while $|S(q^i)| < 2\kappa^7$:

      5.3.1 Run **Sample**$(i, \{q^i\}, \lambda, \frac{e^{-5}}{N(q^i)})$ until it returns $w \neq$ **fail**, and at most $c(\kappa) \in \Theta(\log(\kappa))$ times

      5.3.2 If $w =$ **fail**, then return 0 (failure event)

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

    5.1 Set $R_b = \{p^{i-1} \mid (p^{i-1}, b, q^i) \text{ is a transition in } A_{unroll}\}$ for $b = 0, 1$

    5.2 Set $N(q^i) = N(R_0) + N(R_1)$

    5.3 Set $S(q^i) = \emptyset$. Then while $|S(q^i)| < 2\kappa^7$:

        5.3.1 Run **Sample**$(i, \{q^i\}, \lambda, \frac{e^{-5}}{N(q^i)})$ until it returns $w \neq$ **fail**, and at most $c(\kappa) \in \Theta(\log(\kappa))$ times

        5.3.2 If $w =$ **fail**, then return 0 (failure event)

        5.3.3 Set $S(q^i) = S(q^i) \cup \{w\}$ (recall that $S(q^i)$ allows duplicates)

The complete algorithm

5. For each layer $i = 1, \ldots, n$ and state q^i in A_{unroll}:

   5.1 Set $R_b = \{p^{i-1} \mid (p^{i-1}, b, q^i) \text{ is a transition in } A_{unroll}\}$ for $b = 0, 1$

   5.2 Set $N(q^i) = N(R_0) + N(R_1)$

   5.3 Set $S(q^i) = \emptyset$. Then while $|S(q^i)| < 2\kappa^7$:

      5.3.1 Run **Sample**$(i, \{q^i\}, \lambda, \frac{e^{-5}}{N(q^i)})$ until it returns $w \neq$ **fail**, and at most $c(\kappa) \in \Theta(\log(\kappa))$ times

      5.3.2 If $w =$ **fail**, then return 0 (failure event)

      5.3.3 Set $S(q^i) = S(q^i) \cup \{w\}$ (recall that $S(q^i)$ allows duplicates)

6. Return $N(F^n)$ as an estimation of $|\mathcal{L}_n(A)|$

The complete algorithm: final comments

The probability that the algorithm returns a wrong estimate is at most $\frac{1}{4}$

- Considering $c(\kappa) = \lceil \frac{2+\log(4)+8\log(\kappa)}{\log(1-e^{-9})^{-1}} \rceil$

The algorithm runs in time poly$(m, n, \frac{1}{\varepsilon})$