Sampling in Query Evaluation

Marcelo Arenas
PUC & IMFD Chile and RelationalAI
The goals of this tutorial

- Show some fundamental problems that motivate the use of sampling in databases
- Explain the difficulties behind these problems
- Show some tools that are used to do sampling in this context
- Explain how these tools can be used to provide (partial) solutions to these problems
- Convince the audience that there are interesting open problems in the area, and also that sampling tools could be very useful 😊
Motivation: Three related problems
Problem 1: query optimization

The task is to compute $R[A, B] \bowtie S[B, C] \bowtie T[C, D]$

$$(R \bowtie S) \bowtie T \quad R \bowtie (S \bowtie T) \quad (R \bowtie T) \bowtie S$$
\[(R \bowtie S) \bowtie T\]
\[ R \bowtie (S \bowtie T) \]
\[ R \bowtie (S \bowtie T) \]

\[ \begin{array}{|c|c|}
\hline
R & A \quad B \\
\hline
1 & 2 \\
1 & 4 \\
\vdots \\
n & 4 \\
\hline
\end{array} \]

\[ \begin{array}{|c|c|c|c|}
\hline
S & B \quad C \\
\hline
2 & 0 \\
4 & 1 \\
\vdots \\
4 & n \\
\hline
\end{array} \]

\[ \begin{array}{|c|c|c|c|}
\hline
T & C \quad D \\
\hline
0 & 3 \\
\hline
\end{array} \]
Query optimization

Now the task is to compute $\sigma_{B=4}(R[A, B] \bowtie S[B, C] \bowtie T[C, D])$

$\sigma_{B=4}((R \bowtie S) \bowtie T)$

$R \bowtie (\sigma_{B=4}(S) \bowtie T)$
\[
\sigma_{B=4}( (R \bowtie S) \bowtie T )
\]

**Table R**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>n</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

**Table S**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>n</td>
<td>4</td>
<td>n</td>
</tr>
</tbody>
</table>

**Table T**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>n</td>
<td>4</td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram**

\[\sigma_{B=4}( (R \bowtie S) \bowtie T ) = \emptyset\]
$R \bowtie (\sigma_{B=4}(S) \bowtie T)$
$R \bowtie (\sigma_{B=4}(S) \bowtie T)$
Cardinality estimation

To compare query plans we need estimations of the cardinalities of the intermediate results

- Such estimations should be computed (very) efficiently
Problem 2: approximate query processing [HHW97, HH99]

The task is to compute the aggregate query
\[
\text{COUNT}(R[A, B] \bowtie S[B, C] \bowtie T[C, D])
\]

Not a good strategy to solve this task by first computing
\[
R[A, B] \bowtie S[B, C] \bowtie T[C, D]
\]

- We can approximate the answer by doing a cardinality estimation
Problem 2: approximate query processing [HHW97, HH99]

Can we also approximate \( \text{SUM}_D(R[A, B] \bowtie S[B, C] \bowtie T[C, D]) \) and \( \text{AVG}_A(R[A, B] \bowtie S[B, C] \bowtie T[C, D]) \)?

What kind of guarantees can be offered about the results of these approximations?

- How can such guarantees be obtained?
Problem 3: query exploration

The answer to a query can be very large

It can be more informative to:

- Return the number of answers
- Enumerate the answers with polynomial (constant) delay
- Generate an answer uniformly at random
Problem 3: query exploration

Returning the number of answers to a query can be solved again by using cardinality estimation.
Problem 3: query exploration

Returning the number of answers to a query can be solved again by using cardinality estimation.

Cardinality estimation can also help to generate at random an answer to a query:

- Can we sample with uniform distribution?
- Can sampling be used for cardinality estimation?
What do these problems have in common?

**Sampling** plays a central role in the development of solutions for these problems.
The complexity of counting and uniform generation
Hardness of counting

The problem of counting the number of answers to a join query is \#P-complete.

This can be easily shown by reducing from the problem of counting the number of 3-colorings of a graph.
The problem of counting the number of answers to a join query is \#P-complete.

\[ E(x_1, x_2) \land E(x_2, x_3) \]
\[ \land E(x_3, x_4) \land E(x_4, x_1) \land \]
\[ E(x_4, x_2) \]
Hardness of counting

The problem of counting the number of answers to a join query is #P-complete

\[ Q(x_1, x_2, x_3, x_4) = E(x_1, x_2) \land E(x_2, x_3) \land E(x_3, x_4) \land E(x_4, x_1) \land E(x_4, x_2) \]
Hardness of counting

The problem of counting the number of answers to a join query is \#P-complete

\[
Q(x_1, x_2, x_3, x_4) = E(x_1, x_2) \land E(x_2, x_3) \\
\land E(x_3, x_4) \land E(x_4, x_1) \land \\
E(x_4, x_2)
\]

Number of 3-colorings: \(|Q(E)|\)
Hardness of uniform generation

There is no randomized polynomial-time algorithm for uniform generation of the answers to a join query (unless NP = RP)

If such an algorithm exists, then there exists an FPRAS for the problem of counting the number of answers to a join query (by Jerrum-Valiant-Vazirani)

Then there exists a BPP algorithm problem of verifying whether a join query has a non-empty set of answers
Hardness of uniform generation

There is no randomized polynomial-time algorithm for uniform generation of the answers to a join query (unless NP = RP)

Then there exists a BPP algorithm problem of verifying whether a join query has a non-empty set of answers

But the problem of verifying whether a join query has a non-empty set of answers is NP-complete
How can we get better complexity?

Consider acyclic queries

- Or a class of queries with a bounded degree of acyclicity, such as bounded treewidth or bounded hypertree width
Counting in the acyclic case

\[ R[A, B] \bowtie S[B, C] \bowtie T[C, A] \]
Counting in the acyclic case

\[ R[A, B] \bowtie S[A, C] \bowtie T[A, D] \bowtie U[C, E, F] \]
Counting in the acyclic case

\[ R[A, B] \bowtie S[A, C] \bowtie T[A, D] \bowtie U[C, E, F] \]
Counting in the acyclic case

\[ S \begin{array}{ccc}
  A & C \\
  4 & 1 & 2 \\
  5 & 2 & 1 \\
  4 & 3 & 0 \\
\end{array} \]

\[ R \begin{array}{ccc}
  A & B \\
  4 & 6 \\
  5 & 7 \\
\end{array} \]

\[ U \begin{array}{ccc}
  C & E & F \\
  1 & 3 & 6 & 1 \\
  1 & 4 & 7 & 1 \\
  2 & 5 & 8 & 1 \\
\end{array} \]

\[ T \begin{array}{ccc}
  A & D \\
  4 & 1 & 1 \\
  4 & 2 & 1 \\
  4 & 3 & 1 \\
  5 & 4 & 1 \\
  5 & 5 & 1 \\
\end{array} \]
Counting in the acyclic case

\[
\begin{array}{c|c|c|c}
S & A & C \\
\hline
4 & 1 & 2 \\
5 & 2 & 1 \\
4 & 3 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
R & A & B \\
\hline
4 & 6 & 2 \\
5 & 7 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
T & A & D \\
\hline
4 & 1 & 1 \\
4 & 2 & 1 \\
4 & 3 & 1 \\
5 & 4 & 1 \\
5 & 5 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
U & C & E & F \\
\hline
1 & 3 & 6 & 1 \\
1 & 4 & 7 & 1 \\
2 & 5 & 8 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\end{array}
\]

\[
\begin{array}{c|c|c|c}
U[C, E, F] \\
\end{array}
\]
Counting in the acyclic case

\[
\begin{array}{c|ccc}
S & A & C \\
\hline
4 & 1 & 2 \\
5 & 2 & 1 \\
4 & 3 & 0 \\
\end{array}
\]

\[
\begin{array}{c|cc}
R & A & B \\
\hline
4 & 6 & 2 \\
5 & 7 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
T & A & D \\
\hline
4 & 1 & 1 \\
4 & 2 & 1 \\
4 & 3 & 1 \\
5 & 4 & 1 \\
5 & 5 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
U & C & E & F \\
\hline
1 & 3 & 6 & 1 \\
1 & 4 & 7 & 1 \\
2 & 5 & 8 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c}
R[A, B] \\
\hline
2 \\
1 \\
\end{array}
\]

\[
\begin{array}{c|c}
S[A, C] \\
\hline
2 \\
1 \\
0 \\
\end{array}
\]

\[
\begin{array}{c|c}
T[A, D] \\
\hline
1 \\
1 \\
1 \\
1 \\
\end{array}
\]

\[
\begin{array}{c|c}
U[C, E, F] \\
\hline
1 \\
1 \\
1 \\
1 \\
\end{array}
\]
Counting in the acyclic case

\[
\begin{array}{c|ccc}
S & A & C \\
\hline
4 & 1 & 2 \\
5 & 2 & 1 \\
4 & 3 & 0 \\
\end{array}
\quad
\begin{array}{c|cc}
R & A & B \\
4 & 6 & 2 \cdot 3 \\
5 & 7 & 1 \cdot 2 \\
\end{array}
\quad
\begin{array}{c|cc}
T & A & D \\
4 & 1 & 1 \\
4 & 2 & 1 \\
4 & 3 & 1 \\
5 & 4 & 1 \\
5 & 5 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
U & C & E & F \\
\hline
1 & 3 & 6 & 1 \\
1 & 4 & 7 & 1 \\
2 & 5 & 8 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
S[A, C] \\
R[A, B] \\
T[A, D] \\
U[C, E, F] \\
\end{array}
\]
Counting in the acyclic case

\[
\begin{array}{c|cc}
S & A & C \\
4 & 1 & 2 \\
5 & 2 & 1 \\
4 & 3 & 0 \\
\end{array}
\]

\[
\begin{array}{c|cc}
R & A & B \\
4 & 6 & 6 \\
5 & 7 & 2 \\
\end{array}
\]

\[
\begin{array}{c|cc}
T & A & D \\
4 & 1 & 1 \\
4 & 2 & 1 \\
4 & 3 & 1 \\
5 & 4 & 1 \\
5 & 5 & 1 \\
\end{array}
\]
Counting in the acyclic case

\[
\begin{array}{c|cc}
S & A & C \\
4 & 1 & 2 \\
5 & 2 & 1 \\
4 & 3 & 0 \\
\hline
U & C & E & F \\
1 & 3 & 6 & 1 \\
1 & 4 & 7 & 1 \\
2 & 5 & 8 & 1 \\
\end{array}
\]

\[
\begin{array}{c|cc}
R & A & B \\
4 & 6 & 6 \\
5 & 7 & 2 \\
\hline
T & A & D \\
4 & 1 & 1 \\
4 & 2 & 1 \\
4 & 3 & 1 \\
5 & 4 & 1 \\
5 & 5 & 1 \\
\end{array}
\]

\[
\begin{array}{c}
R[A, B] \\
S[A, C] \\
T[A, D] \\
U[C, E, F] \\
\end{array}
\]
Uniform generation in the acyclic case
Uniform generation in the acyclic case
Uniform generation in the acyclic case

\[ R[A, B] \]
\[ S[A, C] \]
\[ U[C, E, F] \]
\[ T[A, D] \]
Uniform generation in the acyclic case

\[
\begin{align*}
R & \quad A \quad B \\
R[A, B] & \quad 4 \quad 6 \quad 6/8 \\
\quad 5 \quad 7 \quad 2/8
\end{align*}
\]

\[
\begin{align*}
S & \quad A \quad C \\
S[A, C] & \quad 4 \quad 1 \quad 2 \\
\quad 5 \quad 2 \quad 1 \\
\quad 4 \quad 3 \quad 0
\end{align*}
\]

\[
\begin{align*}
T & \quad A \quad D \\
T[A, D] & \quad 4 \quad 1 \quad 1 \\
\quad 4 \quad 2 \quad 1 \\
\quad 4 \quad 3 \quad 1 \\
\quad 5 \quad 4 \quad 1 \\
\quad 5 \quad 5 \quad 1
\end{align*}
\]

\[
\begin{align*}
U & \quad C \quad E \quad F \\
U[C, E, F] & \quad 1 \quad 3 \quad 6 \quad 1 \\
\quad 1 \quad 4 \quad 7 \quad 1 \\
\quad 2 \quad 5 \quad 8 \quad 1
\end{align*}
\]

\[\frac{3}{4}\]
Uniform generation in the acyclic case

\[
\begin{array}{cccc}
A & B & C & D \\
4 & 6 & & \\
\end{array}
\quad \frac{3}{4}
\end{array}
\]

\[
\begin{array}{cccc}
S & A & C \\
4 & 1 & & \\
4 & 3 & & \\
\end{array}
\quad \frac{2}{2}
\]

\[
\begin{array}{cccc}
U & C & E & F \\
1 & 3 & 6 & 1 \\
1 & 4 & 7 & 1 \\
2 & 5 & 8 & 1 \\
\end{array}
\quad \frac{1}{2}
\end{array}
\]

\[
\begin{array}{cccc}
R & A & B \\
4 & 6 & & \\
5 & 7 & & \\
\end{array}
\quad \frac{6}{8}\quad \frac{2}{8}
\]

\[
\begin{array}{cccc}
T & A & D \\
4 & 1 & 1 \\
4 & 2 & 1 \\
4 & 3 & 1 \\
5 & 4 & 1 \\
5 & 5 & 1 \\
\end{array}
\]
Uniform generation in the acyclic case

\[
\begin{bmatrix}
A & B & C & D & E & F \\
4 & 6 & 1 & & & \\
\end{bmatrix} \quad \frac{3}{4} \cdot 1.
\]

\[
\begin{bmatrix}
R & A & B \\
4 & 6 & 6/8 \\
5 & 7 & 2/8 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
S & A & C \\
4 & 1 & 2/2 \\
4 & 3 & 0/2 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
U & C & E & F \\
1 & 3 & 6 & 1 \\
1 & 4 & 7 & 1 \\
2 & 5 & 8 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
T & A & D \\
4 & 1 & 1 \\
4 & 2 & 1 \\
4 & 3 & 1 \\
5 & 4 & 1 \\
5 & 5 & 1 \\
\end{bmatrix}
\]
Uniform generation in the acyclic case

\[
\frac{3}{4} \cdot 1
\]

\[
\begin{array}{cccccc}
A & B & C & D & E & F \\
4 & 6 & 1 & & & \\
\end{array}
\]

\[
\begin{array}{ccc}
S & A & C \\
\hline
4 & 1 & \circled{2/2} & \\
4 & 3 & \circled{0/2} & \\
\end{array}
\]

\[
\begin{array}{cccc}
R & A & B \\
\hline
4 & 6 & \circled{6/8} & \\
5 & 7 & \circled{2/8} & \\
\end{array}
\]

\[
\begin{array}{cccccc}
T & A & D \\
\hline
4 & 1 & 1 & \\
4 & 2 & 1 & \\
4 & 3 & 1 & \\
5 & 4 & 1 & \\
5 & 5 & 1 & \\
\end{array}
\]

\[
\begin{array}{cccc}
U & C & E & F \\
\hline
1 & 3 & 6 & \circled{1/2} \\
1 & 4 & 7 & \circled{1/2} & \\
\end{array}
\]
Uniform generation in the acyclic case

\[ \frac{3}{4} \cdot \frac{1}{2} \cdot \frac{6}{8} \cdot \frac{1}{2} \cdot \frac{3}{4} = \frac{9}{32} \]

\[ R[A, B] \]

\[ S[A, C] \]

\[ T[A, D] \]

\[ U[C, E, F] \]
Uniform generation in the acyclic case

\[
\begin{array}{cccccc}
A & B & C & D & E & F \\
4 & 6 & 1 & 4 & 7
\end{array}
\quad \frac{3}{4} \cdot 1 \cdot \frac{1}{2}
\]

\[
\begin{array}{cccc}
R & A & B \\
4 & 6 & 6/8 \\
5 & 7 & 2/8
\end{array}
\]

\[
\begin{array}{ccc}
S & A & C \\
4 & 1 & 2/2 \\
4 & 3 & 0/2
\end{array}
\]

\[
\begin{array}{ccc}
U & C & E & F \\
1 & 3 & 6 & 1/2 \\
1 & 4 & 7 & 1/2
\end{array}
\]

\[
\begin{array}{ccc}
S & A, C \\
4 & 1 & 2/2 \\
4 & 3 & 0/2
\end{array}
\]

\[
\begin{array}{ccc}
T & A & D \\
4 & 1 & 1/3 \\
4 & 2 & 1/3 \\
4 & 3 & 1/3
\end{array}
\]
Uniform generation in the acyclic case

\[
\begin{array}{cccccc}
A & B & C & D & E & F \\
4 & 6 & 1 & 2 & 4 & 7 \\
\end{array}
\]

\[
\frac{3}{4} \cdot \frac{1}{2} \cdot \frac{1}{3}
\]

\[
\begin{array}{cccc}
R & A & B \\
4 & 6 & 6/8 \\
5 & 7 & 2/8 \\
\end{array}
\]

\[
\begin{array}{cc}
S & A & C \\
4 & 1 & 2/2 \\
4 & 3 & 0/2 \\
\end{array}
\]

\[
\begin{array}{ccc}
T & A & D \\
4 & 1 & 1/3 \\
4 & 2 & 1/3 \\
4 & 3 & 1/3 \\
\end{array}
\]

\[
\begin{array}{cccc}
U & C & E & F \\
1 & 3 & 6 & 1/2 \\
1 & 4 & 7 & 1/2 \\
\end{array}
\]

\[
\begin{array}{ccc}
U & [C, E, F] \\
\end{array}
\]

\[
\begin{array}{ccc}
S & [A, C] \\
\end{array}
\]

\[
\begin{array}{ccc}
T & [A, D] \\
\end{array}
\]

\[
\begin{array}{ccc}
R & [A, B] \\
\end{array}
\]
Uniform generation in the acyclic case

\[
\begin{array}{cccccc}
A & B & C & D & E & F \\
4 & 6 & 1 & 2 & 4 & 7
\end{array}
\]

\[
\frac{3}{4} \cdot \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{8}
\]

\[
R[A, B]
\]

\[
\begin{array}{cc}
R & A & B \\
4 & 6 & 6/8 \\
5 & 7 & 2/8
\end{array}
\]

\[
S[A, C]
\]

\[
\begin{array}{cc}
S & A & C \\
4 & 1 & 2/2 \\
4 & 3 & 0/2
\end{array}
\]

\[
U[C, E, F]
\]

\[
\begin{array}{cccc}
U & C & E & F \\
1 & 3 & 6 & 1/2 \\
1 & 4 & 7 & 1/2
\end{array}
\]

\[
T[A, D]
\]

\[
\begin{array}{ccc}
T & A & D \\
4 & 1 & 1/3 \\
4 & 2 & 1/3 \\
4 & 3 & 1/3
\end{array}
\]
Does this work with other operators?

The previous approach for acyclic queries can be extended to consider the selection operator $\sigma$.

But it does not work if the projection operator $\pi$ is included.
Hardness of counting with projection [PS13]
Hardness of counting with projection [PS13]

The problem of counting the number of perfect matchings in a bipartite graph is $\#P$-complete
Hardness of counting with projection [PS13]
Hardness of counting with projection [PS13]

\[
\begin{array}{cccc}
I_1 & I_2 & I_3 & D \\
4 & 4 & 6 & 4 \\
5 & 5 & 6 & 5 \\
5 & 6 & & 5 \\
6 & 4 & & 6 \\
6 & 5 & & 6 \\
\end{array}
\]

\[
F(x_1, x_2, x_3) = I_1(x_1) \land I_2(x_2) \land I_3(x_3)
\]

\[
W(x_1, x_2, x_3) = I_1(x_1) \land I_2(x_2) \land I_3(x_3) \land \exists y \left( D(x_1, y) \land D(x_2, y) \land D(x_3, y) \right)
\]
Hardness of counting with projection [PS13]

\[ W(x_1, x_2, x_3) = I_1(x_1) \land I_2(x_2) \land I_3(x_3) \land \exists y \ (D(x_1, y) \land D(x_2, y) \land D(x_3, y)) \]
Hardness of counting with projection [PS13]

\[
\begin{array}{c|c|c|c}
I_1 & I_2 & I_3 & D \\
4 & 4 & 6 & 4 5 \\
5 & 5 & 6 & 4 6 \\
5 & 4 & & 5 4 \\
6 & 6 & & 5 6 \\
6 & & & 6 4 \\
& & & 6 5 \\
\end{array}
\]

Number of perfect matchings:
\[|F(I_1, I_2, I_3)| - |W(I_1, I_2, I_3, D)|\]

\[
F(x_1, x_2, x_3) = I_1(x_1) \land I_2(x_2) \land I_3(x_3)
\]

\[
W(x_1, x_2, x_3) = I_1(x_1) \land I_2(x_2) \land I_3(x_3) \land \exists y (D(x_1, y) \land D(x_2, y) \land D(x_3, y))
\]
Does this rule out efficient uniform generation?

No, the argument for join queries does not apply here

- The problem of verifying whether an acyclic conjunctive query has a non-empty set of answers can be solved in polynomial time
For practical applications

- We need to consider both acyclic and cyclic queries
- We need to include all relational algebra operators
- We need to consider aggregation
Part I: join, selection and aggregation
A bit of notation

- \( \text{dom}(A) \): domain of attribute \( A \)
- Given a tuple \( r \) and an attribute \( A \), \( r[A] \) is the value of \( r \) in the attribute \( A \)
- \( r \sim s \): \( r \) and \( s \) have the same values in their common attributes
- \( R \bowtie S = \{ r \in R \mid \exists s \in S : r \sim s \} \)
  - If \( X \) is the set of attributes of \( R \), then \( R \bowtie S = \pi_X( R \bowtie S ) \)
Uniform generation
Sampling with uniform distribution [093,CMN99]

We would like to generate uniformly at random a tuple in
\[ R[A, B] \bowtie S[B, C] \]

Ideally, the probability of choosing a tuple \( t \in R \bowtie S \) should be

\[ \frac{1}{|R \bowtie S|} \]
Sampling with uniform distribution: first attempt

To produce a sample do the following:

1. Generate uniformly at random $r \in R$
2. Generate uniformly at random $s \in S$
3. If $r \sim s$, then return $(r, s)$
Sampling with uniform distribution: first attempt

Tuples in the join are generated uniformly. If \( r \sim s \):

\[
\Pr((r, s) \text{ is generated}) = \frac{1}{|R||S|}
\]

The probability that a tuple is generated is

\[
\frac{|R \bowtie S|}{|R||S|}
\]

If \( |R \bowtie S| \ll |R||S| \), then this probability can be very small
Sampling with uniform distribution: second attempt

To produce a sample do the following:

1. Generate uniformly at random \( r \in R \)
2. Generate uniformly at random \( s \in \sigma_{B=r[B]}(S) \)
3. Return \((r, s)\)
Sampling with uniform distribution: second attempt

But in this cases the tuples in the join are not generated uniformly.

Assuming \( r \sim s \):

\[
\Pr((r, s) \text{ is generated}) = \Pr(r \text{ is generated}) \Pr(s \text{ is generated} \mid r \text{ is generated})
\]

\[
= \frac{1}{|R|} \frac{1}{|S \times \{r\}|}
\]
Sampling with uniform distribution: second attempt
Sampling with uniform distribution: second attempt

How do we solve this problem?
Sampling with uniform distribution: third attempt [093]

Let \( M_B(S) = \max_{v \in \text{dom}(B)} |\sigma_B=v(S)| \)

To produce a sample do the following:

1. Generate uniformly at random \( r \in R \)
2. Reject with probability
   \[
   1 - \frac{|S \times \{r\}|}{M_B(S)}
   \]
3. Generate uniformly at random \( s \in \sigma_B=r[B](S) \)
4. Return \((r, s)\)
Sampling with uniform distribution: third attempt [093]

The tuples in the join are generated uniformly.

Assuming $r \sim s$:

$$\Pr((r, s) \text{ is generated})$$

$$= \Pr(r \text{ is generated}) \Pr(s \text{ is generated} \mid r \text{ is generated})$$

$$= \frac{1}{|R|} \frac{|S \times \{r\}|}{M_B(S)} \frac{1}{|S \times \{r\}|} = \frac{1}{|R| \cdot M_B(S)}$$

Upper bound for $|R \bowtie S|$
A general framework for sampling [ZCLHY18]

Consider the join query $R_1[A_1, A_2] \bowtie R_2[A_2, A_3] \bowtie \cdots \bowtie R_n[A_n, A_{n+1}]$

Given $t \in R_i$, define

$$w(t) = |\{t\} \bowtie R_{i+1} \bowtie \cdots \bowtie R_n|$$

Besides, let

$$w(R) = \sum_{t \in R} w(t)$$
A general framework for sampling [ZCLHY18]

For each $t \in R_i$, we have that $w(t) = w(R_{i+1} \times \{t\})$

$$w(t) = |\{t\} \bowtie R_{i+1} \bowtie r_{i+1} \bowtie \cdots \bowtie R_n|$$

$$= \sum_{t' \in R_{i+1}} |\{t\} \bowtie \{t'\} \bowtie R_{i+2} \cdots \bowtie R_n|$$

$$= \sum_{t' \in R_{i+1} : t \sim t'} |\{t'\} \bowtie R_{i+2} \cdots \bowtie R_n|$$

$$= \sum_{t' \in R_{i+1} \times \{t\}} w(t') = w(R_{i+1} \times \{t\})$$
A general framework for sampling [ZCLHY18]

We do not have access to the values $w(t)$ when sampling, but instead we have some approximations of them.

Assume given an approximation $W$ of $w$ that satisfies the following properties:

1. $W(t) \geq w(t)$
2. $W(t) = w(t) = 1$ for each $t \in R_n$
3. $W(t) \geq W(R_{i+1} \times \{t\})$ for each $t \in R_i$
A general framework for sampling [ZCLHY18]

To produce a sample, do the following:

1. Generate $r_1 \in R_1$ with probability $\frac{W(r_1)}{W(R_1)}$

2. For $i = 2$ to $n$:

   2.1. Reject with probability $1 - \frac{W(R_i \times \{r_{i-1}\})}{W(r_{i-1})}$

   2.2. Generate $r_i \in R_i \times \{r_{i-1}\}$ with probability $\frac{W(r_i)}{W(R_i \times \{r_{i-1}\})}$

3. Return $(r_1, r_2, \ldots, r_n)$
A general framework for sampling [ZCLHY18]

The tuples in the join are generated uniformly

\[ \Pr((r_1, r_2) \text{ is generated}) \]

\[ = \Pr(r_1 \text{ is generated}) \Pr(r_2 \text{ is generated} \mid r_1 \text{ is generated}) \]
The tuples in the join are generated uniformly

\[ \Pr((r_1, r_2) \text{ is generated}) = \Pr(r_1 \text{ is generated}) \Pr(r_2 \text{ is generated} \mid r_1 \text{ is generated}) = \frac{W(r_1)}{W(R_1)}. \]
A general framework for sampling [ZCLHY18]

The tuples in the join are generated uniformly

\[
\Pr((r_1, r_2) \text{ is generated}) = \Pr(r_1 \text{ is generated}) \Pr(r_2 \text{ is generated} \mid r_1 \text{ is generated}) = \frac{W(r_1)}{W(R_1)}.\]

1. Generate \( r_1 \in R_1 \) with probability \( \frac{W(r_1)}{W(R_1)} \)
A general framework for sampling [ZCLHY18]

The tuples in the join are generated uniformly

\[ \Pr((r_1, r_2) \text{ is generated}) = \Pr(r_1 \text{ is generated}) \Pr(r_2 \text{ is generated} \mid r_1 \text{ is generated}) \]

\[ = \frac{W(r_1)}{W(R_1)} \cdot \frac{W(R_2 \times \{r_1\})}{W(r_1)} \cdot \]

2.1. Reject with probability \( 1 - \frac{W(R_i \times \{r_{i-1}\})}{W(r_{i-1})} \)
A general framework for sampling [ZCLHY18]

The tuples in the join are generated uniformly

\[
\Pr((r_1, r_2) \text{ is generated}) = \Pr(r_1 \text{ is generated}) \Pr(r_2 \text{ is generated} \mid r_1 \text{ is generated})
\]

\[
= \frac{W(r_1)}{W(R_1)} \cdot \frac{W(R_2 \times \{r_1\})}{W(r_1)} \cdot \frac{W(r_2)}{W(R_2 \times \{r_1\})}
\]

2.2. Generate \( r_i \in R_i \times \{r_{i-1}\} \) with probability \( \frac{W(r_i)}{W(R_i \times \{r_{i-1}\})} \)
A general framework for sampling [ZCLHY18]

The tuples in the join are generated uniformly

\[ \Pr((r_1, r_2) \text{ is generated}) = \Pr(r_1 \text{ is generated}) \Pr(r_2 \text{ is generated} \mid r_1 \text{ is generated}) \]

\[ = \frac{W(r_1)}{W(R_1)} \cdot \frac{W(R_2 \times \{r_1\})}{W(r_1)} \cdot \frac{W(r_2)}{W(R_2 \times \{r_1\})} = \frac{W(r_2)}{W(R_1)} \]
A general framework for sampling [ZCLHY18]

The tuples in the join are generated uniformly

\[ \Pr((r_1, r_2, \ldots, r_n) \text{ is generated}) = \frac{W(r_n)}{W(R_1)} \]
A general framework for sampling [ZCLHY18]

The tuples in the join are generated uniformly

\[
\Pr((r_1, r_2, \ldots, r_n) \text{ is generated}) = \frac{W(r_n)}{W(R_1)} = \frac{1}{W(R_1)}
\]
A generalization of the idea of [093]

Assume that:

\[ W(r_1) = M_{A_2}(R_2) \text{ for each } r_1 \in R_1 \]
\[ W(r_2) = 1 \text{ for each } r_2 \in R_2 \]
A generalization of the idea of [093]

Then:

\[
W(R_1) = \sum_{t \in R_1} W(t) = \sum_{t \in R_1} M_{A_2}(R_2) = |R_1| M_{A_2}(R_2)
\]

\[
W(R_2) = \sum_{t \in R_2} W(t) = \sum_{t \in R_2} 1 = |R_2|
\]

Therefore:

\[
\text{Pr}((r_1, r_2) \text{ is generated}) = \frac{W(r_1)}{W(R_1)} \cdot \frac{W(R_2 \times \{r_1\})}{W(r_1)} \cdot \frac{W(r_2)}{W(R_2 \times \{r_1\})}
\]
A generalization of the idea of [093]

Then:

\[ W(R_1) = \sum_{t \in R_1} W(t) = \sum_{t \in R_1} M_{A_2}(R_2) = |R_1| M_{A_2}(R_2) \]

\[ W(R_2) = \sum_{t \in R_2} W(t) = \sum_{t \in R_2} 1 = |R_2| \]

Therefore:

\[ \Pr((r_1, r_2) \text{ is generated}) = \frac{M_{A_2}(R_2)}{W(R_1)} \cdot \frac{W(R_2 \times \{r_1\})}{M_{A_2}(R_2)} \cdot \frac{W(r_2)}{W(R_2 \times \{r_1\})} \]
A generalization of the idea of [093]

Then:
\[
W(R_1) = \sum_{t \in R_1} W(t) = \sum_{t \in R_1} M_{A_2}(R_2) = |R_1|M_{A_2}(R_2)
\]
\[
W(R_2) = \sum_{t \in R_2} W(t) = \sum_{t \in R_2} 1 = |R_2|
\]

Therefore:
\[
\Pr((r_1, r_2) \text{ is generated}) = \frac{M_{A_2}(R_2)}{|R_1|M_{A_2}(R_2)} \cdot \frac{W(R_2 \times \{r_1\})}{M_{A_2}(R_2)} \cdot \frac{W(r_2)}{W(R_2 \times \{r_1\})}
\]
A generalization of the idea of [093]

Then:

\[ W(R_1) = \sum_{t \in R_1} W(t) = \sum_{t \in R_1} M_{A_2}(R_2) = |R_1| M_{A_2}(R_2) \]

\[ W(R_2) = \sum_{t \in R_2} W(t) = \sum_{t \in R_2} 1 = |R_2| \]

Therefore:

\[ \Pr((r_1, r_2) \text{ is generated}) = \frac{M_{A_2}(R_2)}{|R_1| M_{A_2}(R_2)} \cdot \frac{|R_2 \times \{r_1\}|}{M_{A_2}(R_2)} \cdot \frac{W(r_2)}{|R_2 \times \{r_1\}|} \]
A generalization of the idea of [093]

Then:

\[
W(R_1) = \sum_{t \in R_1} W(t) = \sum_{t \in R_1} M_{A_2}(R_2) = |R_1| M_{A_2}(R_2)
\]

\[
W(R_2) = \sum_{t \in R_2} W(t) = \sum_{t \in R_2} 1 = |R_2|
\]

Therefore:

\[
\Pr((r_1, r_2) \text{ is generated}) = \frac{M_{A_2}(R_2)}{|R_1| M_{A_2}(R_2)} \cdot \frac{|R_2 \times \{r_1\}|}{M_{A_2}(R_2)} \cdot \frac{1}{|R_2 \times \{r_1\}|}
\]
A generalization of the idea of [093]

Then:

\[
W(R_1) = \sum_{t \in R_1} W(t) = \sum_{t \in R_1} M_{A_2}(R_2) = |R_1| M_{A_2}(R_2)
\]

\[
W(R_2) = \sum_{t \in R_2} W(t) = \sum_{t \in R_2} 1 = |R_2|
\]

Therefore:

\[
\Pr((r_1, r_2) \text{ is generated}) = \frac{1}{|R_1|} \cdot \frac{|R_2 \times \{r_1\}|}{M_{A_2}(R_2)} \cdot \frac{1}{|R_2 \times \{r_1\}|}
\]
A generalization of the idea of [093]

Then:

\[ W(R_1) = \sum_{t \in R_1} W(t) = \sum_{t \in R_1} M_{A_2}(R_2) = |R_1|M_{A_2}(R_2) \]

\[ W(R_2) = \sum_{t \in R_2} W(t) = \sum_{t \in R_2} 1 = |R_2| \]

Therefore:

\[
\Pr((r_1, r_2) \text{ is generated}) = \frac{1}{|R_1|} \cdot \frac{|R_2 \times \{r_1\}|}{M_{A_2}(R_2)} \cdot \frac{1}{|R_2 \times \{r_1\}|} = \frac{1}{|R_1|M_{A_2}(R_2)}
\]
We can use better bounds

Define $W$ as:

- $W(t) = \text{AGM}(R_{i+1} \bowtie \cdots \bowtie R_n)$ for every $t \in R_i$ with $1 \leq i < n$
- $W(t) = 1$ for every $t \in R_n$

$W$ satisfies the three properties
Sampling in the acyclic case

Consider an acyclic join query $R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$

Fix a join tree for this query

- $R_i \prec R_j$ indicates that $R_i$ is an ancestor of $R_j$ in this tree
Sampling in the acyclic case

Given \( t \in R_i \), define

\[
w(t) = \left\{ t \right\} \Join \left( \bigotimes_{R_j : R_i \prec R_j} R_j \right)
\]

Besides, if \( R_j \) is a child of \( R_i \):

\[
w(t, R_j) = \left\{ t \right\} \Join R_j \Join \left( \bigotimes_{R_k : R_j \prec R_k} R_k \right)
\]
Sampling in the acyclic case

Assume given an approximation $W$ of $w$ that satisfies the following properties

1. $W(t) \geq w(t)$
2. $W(t, R_j) \geq w(t, R_j)$ if $t \in R_i$ and $R_j$ is a child of $R_i$
3. $W(t) = w(t) = 1$ if $t \in R_i$ and $R_i$ is a leaf
4. $W(t) \geq W(t, R_{k_1}) \cdot W(t, R_{k_2}) \cdot \ldots \cdot W(t, R_{k_\ell})$ if $t \in R_i$ and the children of $R_i$ are $R_{k_1}, R_{k_2}, \ldots, R_{k_\ell}$
5. $W(t, R_j) \geq W(R_j \times \{t\})$ if $t \in R_i$ and $R_j$ is a child of $R_i$
Sampling in the acyclic case

Sample with probability: \( \frac{W(r_1)}{W(R_1)} \)

Reject with probability: \( 1 - \frac{W(r_1, R_2)W(r_1, R_3)}{W(r_1)} \)

Reject with probability: \( \frac{W(R_2 \times \{r_1\})}{W(r_1, R_2)} \)

Sample with probability: \( \frac{W(r_2)}{W(R_2 \times \{r_1\})} \)

Reject with probability: \( \frac{W(R_3 \times \{r_1\})}{W(r_1, R_3)} \)

Sample with probability: \( \frac{W(r_3)}{W(R_3 \times \{r_1\})} \)
Sampling in the acyclic case

Sample with probability: \[ \frac{W(r_1)}{W(R_1)} \]

Accept with probability: \[ \frac{W(r_1, R_2)W(r_1, R_3)}{W(r_1)} \]

Accept with probability: \[ \frac{W(R_2 \times \{r_1\})}{W(r_1, R_2)} \]

Accept with probability: \[ \frac{W(R_3 \times \{r_1\})}{W(r_1, R_3)} \]

Sample with probability: \[ \frac{W(r_2)}{W(R_2 \times \{r_1\})} \]

Sample with probability: \[ \frac{W(r_3)}{W(R_3 \times \{r_1\})} \]
Sampling in the acyclic case

\[
\Pr((r_1, r_2, r_3) \text{ is generated}) = \\
= \frac{W(r_1)}{W(R_1)} \cdot \frac{W(r_1, R_2)W(r_1, R_3)}{W(r_1)} \cdot \frac{W(R_2 \times \{ r_1 \})}{W(r_1, R_2)} \cdot \frac{W(r_2)}{W(R_2 \times \{ r_1 \})} \cdot \frac{W(R_3 \times \{ r_1 \})}{W(r_1, R_3 \times \{ r_1 \})} \cdot \frac{W(r_3)}{W(R_3 \times \{ r_1 \})} \\
= \frac{W(r_2)W(r_3)}{W(R_1)} \\
= \frac{1}{W(R_1)}
\]
Sampling in the cyclic case

Consider the join query $Q = R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n$

Split $Q$ into join queries $Q_{\text{acyclic}}$ and $Q_{\text{rest}}$ such that $Q = Q_{\text{acyclic}} \bowtie Q_{\text{rest}}$

- Assume that $\{A_1, \ldots, A_k\}$ is the set of attributes that queries $Q_{\text{acyclic}}$ and $Q_{\text{rest}}$ have in common
Sampling in the cyclic case

Let
\[ M_{\text{rest}} = \max_{(v_1, \ldots, v_k) \in \text{dom}(A_1) \times \cdots \times \text{dom}(A_k)} |\{ t \in Q_{\text{rest}} \mid \forall i \in \{1, \ldots, k\} : t[A_i] = v_i \}| \]

To produce a sample do the following:

1. Use the sample algorithm for the acyclic case to generate a tuple \( t \in Q_{\text{acyclic}} \)
2. Reject with probability
\[ 1 - \frac{|Q_{\text{rest}} \times \{t\}|}{M_{\text{rest}}} \]
3. Generate uniformly at random \( t' \in Q_{\text{rest}} \)
4. Return \((t, t')\)
Sampling in the cyclic case

The tuples in the join are generated uniformly

\[ \Pr((t, t') \text{ is generated}) = \Pr(t \text{ is generated}) \Pr(t' \text{ is generated} \mid t \text{ is generated}) \]

\[ = \frac{1}{W(R_1)} \cdot \frac{|Q_{\text{rest}} \times \{t\}|}{M_{\text{rest}}} \cdot \frac{1}{|Q_{\text{rest}} \times \{t\}|} = \frac{1}{W(R_1)M_{\text{rest}}} \]
Estimation of cardinality and aggregates
Properties of estimators

Bias of an estimator $\hat{\theta}$ relative to $\theta$ is defined as

\[
\text{Bias}(\hat{\theta}, \theta) = E[\hat{\theta}] - \theta
\]

- $\hat{\theta}$ is unbiased if $\text{Bias}(\hat{\theta}, \theta) = 0$

$\hat{\theta}_n$ is consistent if $\hat{\theta}_n \xrightarrow{p} \theta$

- For every $\varepsilon > 0$:
  \[
  \lim_{n \to \infty} \Pr(|\hat{\theta}_n - \theta| > \varepsilon) = 0
  \]

We would like $\hat{\theta}_n$ to be computable in polynomial time in $n$
Confidence intervals

We would like to provide the following guarantee:

$$\Pr (\theta \in [f(\hat{\theta}), g(\hat{\theta})]) \geq 1 - \delta$$

Which is usually translated into the following:

$$\Pr (\theta \in [\hat{\theta}_n - \varepsilon(n), \hat{\theta}_n + \varepsilon(n)]) \geq 1 - \delta$$
Confidence intervals

Two fundamental tools to construct confidence intervals:

1. Central Limit Theorem
   - The confidence interval depends on the convergence rate, so it would be an approximation if we consider a fixed value $n$
   - A way to deal with this is to use the Berry–Esseen theorem, which gives a precise bound on the difference with the standard normal distribution
Confidence intervals

Two fundamental tools to construct confidence intervals:

2. Concentration inequalities: Chebyshev, Hoeffding, ...
   • The bounds produced are not approximations, but they are looser

In both cases it is convenient to have a small variance
Confidence intervals

Chebyshev inequality:

\[ \Pr(|X - E[X]| \geq \varepsilon) \leq \frac{\text{Var}[\hat{\theta}]}{\varepsilon^2} \]

Assuming \( \hat{\theta} \) is an unbiased estimator of \( \theta \), we can rewrite Chebyshev inequality as:

\[ \Pr (\theta \in (\hat{\theta} - \varepsilon, \hat{\theta} + \varepsilon)) \geq 1 - \frac{\text{Var}[\hat{\theta}]}{\varepsilon^2} \]
Warming up [LWYZ16]

Consider the following SQL query $Q$ over the schema $R[A, B]$:

$$\text{SUM}_D(R[A, B] \bowtie S[B, C] \bowtie T[C, D])$$

We would like to construct an estimator for the answer to this query
Warming up [LWYZ16]

$$R[A, B] \quad S[B, C] \quad T[C, D]$$

$$r_1 \quad s_1 \quad t_1$$

$$r_2 \quad r_1[B] = s_2[B] \quad s_2 \quad t_2$$

$$r_3 \quad s_3 \quad t_3$$

$$t_4$$
Warming up [LWYZ16]

\[ R[A, B] \quad S[B, C] \quad T[C, D] \]

\[
\begin{align*}
\text{r}_1 & \quad \text{s}_1 \\
\text{r}_2 & \quad \text{s}_2 \\
\text{r}_3 & \quad \text{s}_3 \\
\end{align*}
\]

\[
\begin{align*}
\text{t}_1 & \\
\text{t}_2 & \\
\text{t}_3 & \\
\text{t}_4 & \\
\end{align*}
\]
Warming up [LWYZ16]

\[ R[A, B] \quad S[B, C] \quad T[C, D] \]

\[ r_1 \quad r_2 \quad r_3 \]

\[ s_1 \quad s_2 \quad s_3 \]

\[ t_1 \quad t_2 \quad t_3 \quad t_4 \]
Warming up [LWYZ16]

\[ R[A, B] \quad S[B, C] \quad T[C, D] \]

Diagram:
- \( r_1 \) connected to \( s_1 \)
- \( r_2 \) connected to \( s_2 \)
- \( r_3 \) connected to \( s_3 \)
- \( s_1 \) connected to \( t_1 \)
- \( s_2 \) connected to \( t_2 \)
- \( s_3 \) connected to \( t_3 \) and \( t_4 \)
Warming up [LWYZ16]

\[ R[A, B] \quad S[B, C] \quad T[C, D] \]

\[
\begin{align*}
\text{Pr}((r_1, s_2, t_4) \text{ is generated}) &= \frac{1}{18} \\
v(r_1, s_2, t_4) &= t_4[D]
\end{align*}
\]
Warming up [LWYZ16]

\[ R[A, B] \quad S[B, C] \quad T[C, D] \]

\[ \Pr((r_1, s_1) \text{ is generated}) = \frac{1}{6} \]

\[ v(r_1, s_1) = 0 \]
Warming up [LWYZ16]

How do we estimate \( \text{SUM}_D(R[A, B] \bowtie S[B, C] \bowtie T[C, D]) \)?

Given a path \( \gamma \), define \( X(\gamma) = v(\gamma) \)

We can use \( X \) as an estimator

- But this is a biased estimator, as it does not consider that different paths can have different probabilities

How can we transform \( X \) into an unbiased estimator?
Warming up [LWYZ16]

Horvitz–Thompson idea:

\[ Y(\gamma) = \frac{v(\gamma)}{\Pr(\gamma \text{ is generated})} \]
Warming up [LWYZ16]

Horvitz-Thompson idea:

\[ Y(\gamma) = \frac{v(\gamma)}{\Pr(\gamma \text{ is generated})} \]

\( Y \) is unbiased:

\[ E[Y] = \sum_{\gamma} \Pr(\gamma \text{ is generated}) \cdot Y(\gamma) \]

\[ = \sum_{\gamma} \Pr(\gamma \text{ is generated}) \cdot \frac{v(\gamma)}{\Pr(\gamma \text{ is generated})} \]

\[ = \sum_{\gamma} v(\gamma) \]
The Horvitz–Thompson estimator [HT52, T12]

Suppose that we have a list of values $(v_1, \ldots, v_N)$, and we need to estimate:

$$\tau = \sum_{i=1}^{N} v_i$$

To do this estimation, we construct a sample of size $n$ of elements from $\{1, \ldots, N\}$

- With or without replacement
The Horvitz–Thompson estimator \([HT52,T12]\)

\(X_i\): number of times element \(i \in \{1, \ldots, N\}\) appears in the sample

- If we sample without replacement, then \(X_i\) can be 0 or 1

Let \(\pi_i = E[X_i]\)
The Horvitz–Thompson estimator [HT52,T12]

The Horvitz–Thompson (HT) estimator of $\tau$:

$$Y = \sum_{i=1}^{N} \frac{X_i v_i}{\pi_i} = \sum_{i \in \text{sample}} \frac{X_i v_i}{\pi_i}$$

inverse weighting
The Horvitz–Thompson estimator [HT52, T12]

The Horvitz–Thompson (HT) estimator of $\tau$:

$$Y = \sum_{i=1}^{N} \frac{X_i v_i}{\pi_i} = \sum_{i \in \text{sample}} \frac{X_i v_i}{\pi_i}$$

HT is unbiased:

$$E[Y] = E \left[ \sum_{i=1}^{N} \frac{X_i v_i}{\pi_i} \right] = \sum_{i=1}^{N} \frac{E[X_i] v_i}{\pi_i} = \sum_{i=1}^{N} \frac{\pi_i v_i}{\pi_i} = \tau$$
An example of HT

We sample uniformly with replacement: \( p = \frac{1}{N} \)

We can think of \( X_i \) as

\[
X_i = \sum_{k=1}^{n} Z_{i,k},
\]

where \( Z_{i,k} \) is 1 if \( i \) is the \( k \)-th element sampled, and 0 otherwise

\( X_i \sim \text{Binomial}(n, p) \) since each \( Z_{i,k} \sim \text{Bernoulli}(p) \) and these random variables are mutually independent
An example of HT

$$\pi_i = E[X_i] = np$$
An example of HT

\[ \pi_i = E[X_i] = np \]

HT estimator in this case:

\[ Y = \sum_{i=1}^{N} \frac{X_i \pi_i}{\pi_i} = \sum_{i=1}^{N} \frac{X_i v_i}{n p} = \frac{N}{n} \sum_{i \in \text{sample}} X_i v_i \]
What is the variance of HT?

Let $\pi_{i,j} = E[X_i X_j]$

$E[X_i X_j]$ is not necessarily equal to $E[X_i]E[X_j]$

- $X_i$ and $X_j$ are not independent random variables since $X_1 + \cdots + X_N = n$
What is the variance of HT?

\[
\sigma^2(Y) = E[Y^2] - E[Y]^2 = E \left[ \left( \sum_{i=1}^{N} \frac{X_i v_i}{\pi_i} \right)^2 \right] - \tau^2
\]

\[
= E \left[ \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{X_i X_j}{\pi_i \pi_j} v_i v_j \right] - \left( \sum_{i=1}^{N} \frac{v_i}{\pi_i} \right)^2
\]

\[
= \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{E[X_i X_j]}{\pi_i \pi_j} v_i v_j - \sum_{i=1}^{N} \sum_{j=1}^{N} v_i v_j
\]

\[
= \sum_{i=1}^{N} \sum_{j=1}^{N} \left( \frac{\pi_i, j}{\pi_i \pi_j} - 1 \right) v_i v_j
\]
But an estimation of $\sigma^2(Y)$ is usually needed in practice.

How do we estimate $\sigma^2(Y)$? We use HT again!

Define $X_{i,j} = X_i X_j$ and

$$v_{i,j} = \left( \frac{\pi_{i,j}}{\pi_i \pi_j} - 1 \right) v_i v_j$$

We have that

$$\sigma^2(Y) = \sum_{(i,j) \in \{1, \ldots, N\} \times \{1, \ldots, N\}} v_{i,j}$$
But an estimation of $\sigma^2(Y)$ is usually needed in practice

The HT estimator of $\sigma^2(Y)$ is

$$\hat{\sigma}^2(Y) = \sum_{(i,j)\in\{1,\ldots,N\} \times \{1,\ldots,N\}} \frac{X_{i,j}v_{i,j}}{\pi_{i,j}},$$

given that $E[X_{i,j}] = E[X_iX_j] = \pi_{i,j}$
But an estimation of $\sigma^2(Y)$ is usually needed in practice

Replacing the values of $v_{i,j}$, we obtain:

$$\hat{\sigma}^2(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{X_i X_j}{\pi_{i,j}} \left( \frac{\pi_{i,j}}{\pi_i \pi_j} - 1 \right) v_i v_j = \sum_{i,j \in \text{sample}} \frac{X_i X_j}{\pi_{i,j}} \left( \frac{\pi_{i,j}}{\pi_i \pi_j} - 1 \right) v_i v_j$$
The idea behind the HT estimator can be used to define unbiased estimators in many different scenarios.

In this sense, we can talk about a family of HT estimators.
Estimation in databases
Let's put what we learned into practice [CGHJ12]

Consider the following SQL query $Q$ over the schema $R[A, B]$:

$$\text{SUM}_B(R[A, B])$$

The result $Q(R)$ of this query is $\sum_{r \in R} r[B]$, so we need an estimator for this amount.
Simple random sampling with replacement (SRSWR)

To produce the sample repeat \( n \) times the following steps:

1. Generate uniformly at random \( r \in R \)
2. Add \( r \) to the sample
Simple random sampling with replacement (SRSWR)

$X_r$: number of times tuple $r$ appears in the sample

- $\pi_r = E[X_r] = \frac{n}{|R|}$

The HT estimator of $Q(R)$:

$$Y = \sum_{r \in R} \frac{X_r \cdot r[B]}{\pi_r} = \frac{|R|}{n} \sum_{r \in \text{sample}} X_r \cdot r[B]$$
The variance for SRSWR

For \( i \in \{1, \ldots, n\} \), let \( W_i \) be a random variable such that for each possible value \( v \) of attribute \( B \):

\[
\Pr(W_i = v) = \frac{|\{r \in R \mid r[B] = v\}|}{|R|}
\]

We have that:

\[
Y = \frac{|R|}{n} \sum_{r \in \text{sample}} X_r \cdot r[B] = \frac{|R|}{n} \sum_{i=1}^{n} W_i
\]
The variance for SRSWR

\[ E[W_i] = \sum_v v \cdot \Pr(W_i = v) = \frac{1}{|R|} \sum_v v \cdot |\{ r \in R \mid r[B] = v \}| = \frac{Q(R)}{|R|} \]
The variance for SRSWR

\[ E[W_i] = \sum_v v \cdot \Pr(W_i = v) = \frac{1}{|R|} \sum_v v \cdot |\{r \in R \mid r[B] = v\}| = \frac{Q(R)}{|R|} \]

Random variables \( W_i \) are mutually independent:

\[ \sigma^2(Y) = \sigma^2\left(\frac{|R|}{n} \sum_{i=1}^N W_i\right) = \frac{|R|^2}{n^2} \sum_{i=1}^N \sigma^2(W_i) \]
The variance for SRSWR

\[ E[W_i] = \sum_v v \cdot \Pr(W_i = v) = \frac{1}{|R|} \sum_v v \cdot |\{r \in R \mid r[B] = v\}| = \frac{Q(R)}{|R|} \]

We have that:

\[ \sigma^2(W_i) = E[(W_i - E[W_i])^2] = \sum_{r \in R} \frac{1}{|R|} \left( r[B] - \frac{Q(R)}{|R|} \right)^2 = \sigma^2(R) \]

We conclude that:

\[ \sigma^2(Y) = \frac{|R|^2}{n^2} \sum_{i=1}^n \sigma^2(W_i) = \frac{|R|^2}{n^2} \sum_{i=1}^n \sigma^2(R) = \frac{|R|^2 \sigma^2(R)}{n} \]
Simple random sampling without replacement (SRSWoR)

To produce the sample repeat \( n \) times the following steps:

1. Generate uniformly at random \( r \in R \)
2. Add \( r \) to the sample and remove it from \( R \)
Simple random sampling without replacement (SRSWoR)

$X_r$: number of times tuple $r$ appears in the sample, which can be 0 or 1

$X_r \sim \text{Bernoulli}(p)$, where $p$ is the following probability

Assume that $s_k$ is the $k$-th element sampled, so that:

$$p = \Pr(X_r = 1) = \Pr \left( \bigvee_{i=1}^{n} s_i = r \right)$$
Simple random sampling without replacement (SRSWoR)

\[
\Pr\left(\bigvee_{i=1}^{n} s_i = r \right) = \Pr\left(\bigvee_{i=1}^{n} \left[ s_i = r \land \bigwedge_{j=1}^{i-1} s_j \neq r \right] \right)
\]

\[
= \sum_{i=1}^{n} \Pr\left( s_i = r \land \bigwedge_{j=1}^{i-1} s_j \neq r \right)
\]

\[
= \sum_{i=1}^{n} \frac{\binom{|R|-1}{i-1}}{\binom{|R|}{i-1}} \cdot \frac{1}{|R| - (i - 1)}
\]

\[
= \sum_{i=1}^{n} \frac{|R| - (i - 1)}{|R|} \cdot \frac{1}{|R| - (i - 1)} = \frac{n}{|R|}
\]
Simple random sampling without replacement (SRSWoR)

\[ \pi_r = E[X_r] = \frac{n}{|R|} \]

The HT estimator of \( Q(R) \):

\[ Y = \sum_{r \in R} \frac{X_r \cdot r[B]}{\pi_r} = \frac{|R|}{n} \sum_{r \in \text{sample}} X_r \cdot r[B] = \frac{|R|}{n} \sum_{r \in \text{sample}} r[B] \]

This is a similar estimator to the one for the case with replacement. But what is the variance of \( Y \)?
The variance for SRSWoR

The variance is lower than for the case of SRSWR:

\[ \sigma^2(Y) = \frac{|R|(|R| - n)\sigma^2(R)}{n} \]
A second group of estimators [VMZC15, HYPM19]

Now consider the following SQL query $Q$ over the schema $R[A, B], S[B, C]$: 

$$\text{SUM}_C(R[A, B] \bowtie S[B, C])$$
Bernoulli sampling: first alternative

To produce the sample do the following for each $(r, s) \in R \times S$:

1. Generate uniformly at random $x \in [0, 1]$
2. If $x \leq p$, then add $(r, s)$ to the sample
Bernoulli sampling: first alternative

\(X_{r,s}: \text{number of times } (r, s) \in R \times S \text{ appears in the sample}\)

- \(X_{r,s} \sim \text{Bernoulli}(p), \text{ so that } \pi_{r,s} = E[X_{r,s}] = p\)

HT estimator of \(Q(R, S)\):

\[
Y = \sum_{(r,s) \in R \times S} \frac{X_{r,s} \cdot v_{r,s}}{\pi_{r,s}} = \frac{1}{p} \sum_{r \in \text{sample}} v_{r,s}
\]

But how is \(v_{r,s}\) defined? It cannot always be \(s[C]\)

- \(v_{r,s} = s[C]\) if \(r \sim s\), and \(v_{(r,s)} = 0\) otherwise
Bernoulli sampling: first alternative

The random variables $X_{r,s}$ are mutually independent, so $\sigma^2(Y)$ is easy to compute.

But we have a problem: the loop considers all the tuples, so we could just compute the exact answer to the query.

How do we solve this problem?
Independent Bernoulli sampling

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$b_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$b_2$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$a_N$</td>
<td>$b_N$</td>
</tr>
</tbody>
</table>

$p_R \rightarrow \text{sample}_R$

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b'_1$</td>
<td>$c_1$</td>
</tr>
<tr>
<td>$b'_2$</td>
<td>$c_2$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$b'_M$</td>
<td>$c_M$</td>
</tr>
</tbody>
</table>

$p_S \rightarrow \text{sample}_S$

$\text{sample} = \text{sample}_R \bowtie \text{sample}_S$
Independent Bernoulli sampling

To produce the sample do the following:

1. For each $r \in R$, generate uniformly at random $x \in [0, 1]$, and add $r$ to $\text{sample}_R$ if $x \leq p_R$
2. For each $s \in S$, generate uniformly at random $x \in [0, 1]$, and add $s$ to $\text{sample}_S$ if $x \leq p_S$
3. Let $\text{sample} = \text{sample}_R \bowtie \text{sample}_S$
Independent Bernoulli sampling

$X_{r,s}$ and $v_{r,s}$ are defined as before

- $X_{r,s} \sim \text{Bernoulli}(p_R p_S)$, so that $\pi_{r,s} = E[X_{r,s}] = p_R p_S$

HT estimator of $Q(R, S)$:

$$Y = \sum_{(r,s) \in R \times S} \frac{X_{r,s} \cdot v_{r,s}}{\pi_{r,s}} = \frac{1}{p_R p_S} \sum_{r \in \text{sample}} v_{r,s}$$
The variance of independent Bernoulli sampling

Random variables $X_{r,s}$ are not mutually independent

- If $s \neq s'$, then $\Pr(X_{r,s'} = 1 \mid X_{r,s} = 1) = p_S \neq \Pr(X_{r,s'} = 1)$
The variance of independent Bernoulli sampling

We have that:

\[ \text{Var}[Y] = \sum_{(r,s) \in R \times S} \left( \frac{1}{p_{RPS}} - 1 \right) v_{r,s}^2 + \]

\[ \sum_{r \in R} \sum_{s_1, s_2 \in S : s_1 \neq s_2} \left( \frac{1}{p_R} - 1 \right) v_{r,s_1} v_{r,s_2} + \]

\[ \sum_{r_1, r_2 \in R : r_1 \neq r_2} \sum_{s \in S} \left( \frac{1}{p_S} - 1 \right) v_{r_1,s} v_{r_2,s} \]
The variance of independent Bernoulli sampling

And we also have a simple HT estimator of the variance:

\[
\hat{\text{Var}}[Y] = \sum_{(r,s) \in R \times S} \frac{X_r X_s}{p Rp s} \left( \frac{1}{p Rp s} - 1 \right) v_{r,s}^2 + \\
\sum_{r \in R} \sum_{s_1, s_2 \in S : s_1 \neq s_2} \frac{X_r X_s}{p Rp s} \left( \frac{1}{p R} - 1 \right) v_{r,s_1} v_{r,s_2} + \\
\sum_{r_1, r_2 \in R : r_1 \neq r_2} \sum_{s \in S} \frac{X_r X_s}{p Rp s} \left( \frac{1}{p S} - 1 \right) v_{r_1,s} v_{r_2,s}
\]
The variance of independent Bernoulli sampling

And we also have a simple HT estimator of the variance:

\[
\hat{\text{Var}}[Y] = \sum_{r \in \text{sample}_R} \sum_{s \in \text{sample}_S} \frac{X_r X_s}{p_{RPS}} \left( \frac{1}{p_{RPS}} - 1 \right) v_{r,s}^2 + \\
\sum_{r \in \text{sample}_R} \sum_{s_1, s_2 \in \text{sample}_S : s_1 \neq s_2} \frac{X_r X_s}{p_{RPS}} \left( \frac{1}{p_R} - 1 \right) v_{r,s_1} v_{r,s_2} + \\
\sum_{r_1, r_2 \in \text{sample}_R : r_1 \neq r_2} \sum_{s \in \text{sample}_S} \frac{X_r X_s}{p_{RPS}} \left( \frac{1}{p_S} - 1 \right) v_{r_1,s} v_{r_2,s}
\]
Join size estimation

Consider the schema $R[A, B], S[B, C]$

We can reuse the techniques presented in the previous slides to estimate $|R \bowtie S|$

If we add a column $aux$ to $S$ with value 1 in each tuple, then estimating $|R \bowtie S|$ corresponds to the problem of estimating the answer to the following SQL query:

$$\text{SUM}_aux(R[A, B] \bowtie S[B, C, aux])$$
Universe sampling [VMZC15]

\[
\begin{array}{c|cc}
R & A & B \\
\hline
a_1 & b_1 \\
\vdots & \\
a_N & b_N \\
\end{array}
\]

sample \_R

\[
\begin{array}{c|cc}
S & B & C \\
\hline
b'_1 & c_1 \\
b'_2 & c_2 \\
\vdots & \\
b'_M & c_M \\
\end{array}
\]

sample \_S

\[
\text{sample} = \text{sample} \_R \bowtie \text{sample} \_S
\]
Universe sampling [VMZC15]

\[
\begin{array}{c|cc}
R & A & B \\
\hline
a_1 & b_1 & \\
a_2 & b_2 & \\
\vdots & & \\
a_N & b_N & \\
\end{array}
\quad\quad
\begin{array}{c|cc}
S & B & C \\
\hline
b'_1 & c_1 & \\
b'_2 & c_2 & \\
\vdots & & \\
b'_M & c_M & \\
\end{array}
\]

\[\text{sample}_R \bowtie \text{sample}_S\]
Universe sampling [VMZC15]

Assume given a (perfect) hash function $h : \text{dom}(B) \rightarrow [0, 1]$

To produce the sample do the following:

1. For each $r \in R$, if $h(r[B]) \leq p$, then add $r$ to $\text{sample}_R$
2. For each $s \in S$, if $h(s[B]) \leq p$, then add $s$ to $\text{sample}_S$
3. Let $\text{sample} = \text{sample}_R \bowtie \text{sample}_S$
Universe sampling [VMZC15]

\[X_{r,s}: \text{number of times } (r, s) \text{ appears in the sample}\]

- \(X_{r,s} \sim \text{Bernoulli}(p), \text{ so that } \pi_{r,s} = E[X_{r,s}] = p\)

HT estimator of \(Q(R, S)\):

\[
Y = \sum_{r \in R} \sum_{s \in S} \frac{X_{r,s} \cdot v_{r,s}}{\pi_{r,s}} = \frac{1}{p} \sum_{r \in \text{sample}_R} \sum_{s \in \text{sample}_S} v_{r,s}
\]

where \(v_{r,s} = 1\) if \(r \sim s\), and \(v_{r,s} = 0\) otherwise
The variance of universe sampling

Random variables $X_{r,s}$ are not mutually independent

- If $s \neq s'$ and $s[B] = s'[B]$, then $\Pr(X_{r,s'} = 1 \mid X_{r,s} = 1) = 1$
The variance of universe sampling

But the variance of $Y$ can be computed considered the following representation of this random variable

For $v \in \text{dom}(B)$, let

$$N_R(v) = |\{ r \in R \mid r[B] = v \}|$$
$$N_S(v) = |\{ s \in S \mid s[B] = v \}|$$
The variance of universe sampling

$X_v$: random variable such that $X_v = 1$ if $v$ is included as the value of attribute $B$ for some tuple in the sample, and 0 otherwise

- $X_v \sim \text{Bernoulli}(p)$

Then we can represent $Y$ as the following HT estimator:

$$Y = \sum_{v \in \text{dom}(B)} \frac{X_v N_R(v) N_S(v)}{E[X_v]} = \frac{1}{p} \sum_{v \in \text{dom}(B)} X_v N_R(v) N_S(v)$$
The variance of universe sampling

Random variables $X_v$ are mutually independent:

\[
\text{Var}[Y] = \text{Var} \left[ \frac{1}{p} \sum_{v \in \text{dom}(B)} X_v N_R(v) N_S(v) \right]
\]

\[
= \frac{1}{p^2} \sum_{v \in \text{dom}(B)} \text{Var}[X_v] N_R^2(v) N_S^2(v)
\]

\[
= \frac{1}{p^2} \sum_{v \in \text{dom}(B)} p(1 - p) N_R^2(v) N_S^2(v)
\]

\[
= \left( \frac{1}{p} - 1 \right) \sum_{v \in \text{dom}(B)} N_R^2(v) N_S^2(v)
\]
What about other operators?

The previous techniques can be easily extended to consider the selection operator

- We leave this as an exercise for the reader

But the inclusion of projection is more challenging
Part II: Adding
projection
What is left?

We now consider the operators join, selection and projection

- We consider conjunctive queries

Our goal is to show how to do *efficient* cardinality estimation for acyclic conjunctive queries
\[ R[A, B] \bowtie S[A, C] \bowtie T[A, D] \bowtie U[C, E, F] \]
\[ R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w) \]
\[ Q(x, y, z, u, v, w) = R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w) \]
\[ Q(x, y, z, u, v, w) = R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w) \]
\[
Q'(x, w) = \exists y \exists z \exists u \exists v [R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w)]
\]
\[ Q'(x, w) = \exists y \exists z \exists u \exists v \left[ R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w) \right] \]
\[ Q'(x, w) = \exists y \exists z \exists u \exists v \left[ R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w) \right] \]
The main ingredient in the solution: Tree automata

This is the right representation for the problem of counting the number of answers to an acyclic conjunctive query
Tree automata

\[(q, b, \lambda) \quad q \quad b \quad r \quad (r, a, qr)\]

\[(p, a, qr)\]
Tree automata: $(Q, \Sigma, \Delta, I)$

- $Q = \{p, q, r\}$ is the set of states
- $\Sigma = \{a, b\}$ is the alphabet
- $I = \{p\}$ is the set of initial states
- $\Delta = \{(p, a, qr), (q, b, \lambda), (r, a, qr)\}$ is the transition relation
Tree automata
Tree automata

174
Tree automata

(a, e, (e, a, eo))

(e, b, (e, b, λ))

(b, a, (o, a, oo))
Tree automata

\begin{tikzpicture}
  \node (a) at (0,0) {$a$} edge [above, bend left, draw=black, thick] node [left] {$e$} node [right] {$(e, a, eo)$} (a);
  \node (b) at (-1,-2) {$b$} edge [above, bend left, draw=black, thick] node [left] {$(e, b, \lambda)$} (a);
  \node (c) at (1,-2) {$a$} edge [above, bend left, draw=black, thick] node [right] {$o$} node [left] {$(o, a, ee)$} (a);
  \node (d) at (0,-4) {$b$} edge [above, bend left, draw=black, thick] node [left] {$(e, b, ee)$} (c);
  \node (e) at (1,-4) {$a$} edge [above, bend left, draw=black, thick] node [right] {$e$} (c);
\end{tikzpicture}
\(Q'(x, w) = \exists y \exists z \exists u \exists v [R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w)]\)

<table>
<thead>
<tr>
<th>(Q')</th>
<th>(x)</th>
<th>(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(R)</th>
<th>(x)</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(S)</th>
<th>(x)</th>
<th>(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(T)</th>
<th>(x)</th>
<th>(u)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(U)</th>
<th>(z)</th>
<th>(v)</th>
<th>(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

**Alphabet:**
- \(R(4, \star)\)
- \(S(4, \star)\)
- \(T(4, \star)\)
- \(U(\star, \star, 6)\)
- \(R(5, \star)\)
- \(S(5, \star)\)
- \(T(5, \star)\)
- \(U(\star, \star, 7)\)
- \(U(\star, \star, 8)\)

**States:**
- \(R(4, 6)\)
- \(S(4, 1)\)
- \(U(1, 3, 6)\)
- \(R(5, 7)\)
- \(S(5, 2)\)
- \(U(1, 4, 7)\)
- \(S(4, 3)\)
- \(U(2, 5, 8)\)
\[ Q'(x, w) = \exists y \exists z \exists u \exists v [R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w)] \]

<table>
<thead>
<tr>
<th>(Q')</th>
<th>(x)</th>
<th>(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(R)</th>
<th>(x)</th>
<th>(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(S)</th>
<th>(x)</th>
<th>(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(T)</th>
<th>(x)</th>
<th>(u)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(U)</th>
<th>(z)</th>
<th>(v)</th>
<th>(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
\[ Q'(x, w) = \exists y \exists z \exists u \exists v [R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w)] \]
\[ Q'(x, w) = \exists y \exists z \exists u \exists v \left[ R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w) \right] \]
\[ Q'(x, w) = \exists y \exists z \exists u \exists v [R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w)] \]

<table>
<thead>
<tr>
<th>Q'</th>
<th>x</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>x</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>x</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U</th>
<th>z</th>
<th>v</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

\[(R(4, 6), \ R(4, *), \ S(4, 1)T(4, 3)) \quad R(4, *) \quad R(4, 6)\]

\[(S(4, 1), \ S(4, *), \ U(1, 3, 6)) \quad (T(4, 3), \ T(4, *), \ \lambda)\]

\[U(*, *, 6)\]
\[ Q'(x, w) = \exists y \exists z \exists u \exists v [R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w)] \]
\[ Q'(x, w) = \exists y \exists z \exists u \exists v [R(x, y) \land S(x, z) \land T(x, u) \land U(z, v, w)] \]

<table>
<thead>
<tr>
<th>( Q' )</th>
<th>( x )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 6</td>
<td>5 7</td>
<td>5 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( R )</th>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 6</td>
<td>5 7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( S )</th>
<th>( x )</th>
<th>( z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 1</td>
<td>4 3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( T )</th>
<th>( x )</th>
<th>( u )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 1</td>
<td>4 3</td>
<td></td>
</tr>
<tr>
<td>4 2</td>
<td>4 3</td>
<td></td>
</tr>
<tr>
<td>5 4</td>
<td>5 5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( U )</th>
<th>( z )</th>
<th>( v )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3 6</td>
<td>1 4 7</td>
<td>2 5 8</td>
<td></td>
</tr>
</tbody>
</table>

The problem to solve: **count the number of trees with 4 nodes accepted by the tree automaton**
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 \( t \in L(T) \) and the number of nodes of \( t \) is \( n \)

What is the complexity of this problem?
A detour: graph databases
Graph databases

$G$: 

Paul -- friend --> Jack

Paul -- knows --> Leah

Leah -- friend --> Zara

Leah -- friend --> John

Paul -- friend --> John

John -- friend --> Nora

John -- friend --> Zara

Leah -- knows --> Zara

Zara -- friend --> Nora
A query: \((\text{friend} + \text{knows})^*\)
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 $n$

• GEN: generate uniformly at random a path $p$ in $G$ such that $p$ conforms to $r$ and the length of $p$ is $n$
COUNT is a difficult problem

COUNT is \#P-complete

The decision version of the problem can be solved in polynomial time, so this problem could admit an FPRAS
The connection with \#TA

The problem \#NFA:

- **Input:** A non-deterministic finite automaton (NFA) $A$ over the alphabet $\{0, 1\}$ and a number $n$ (given in unary)
- **Output:** Number of words $w$ such that $w \in L(A)$ and the length of $w$ is $n$
The connection with #TA

COUNT and #NFA are polynomially equivalent under parsimonious reductions

- This implies that if an FPRAS exists for one of them, then it exists for the other

#TA is #P-complete

- The construction of an FPRAS for #NFA seems to be a natural step to construct an FPRAS for #TA
Existence of an FPRAS for \#NFA

How do we obtain such an approximation algorithm?

- We use the techniques learned in the previous part of the tutorial!
An FPRAS for \#NFA

An NFA $A$ over the alphabet \{0, 1\} and a number $n$ (given in unary)

Number of words $w$ such that $w \in L(A)$ and the length of $w$ is $n$

Assume that $L_n(A) = \{w \in L(A) \mid |w| = n\}$, so that the output of \#NFA is $|L_n(A)|$
An 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:

\[
N \left( 1 \pm \varepsilon \right) \left| L_n(A) \right| \nonumber
\]

Moreover, number \( N \) has to be computed in time poly\((m, n, \frac{1}{\varepsilon})\), where \( m \) is the number of states of \( A \)
An FPRAS for \#NFA

If we think of the approximation algorithm as an estimator \( \hat{N} \) for \( |L_n(A)| \), then we need to construct the following confidence interval:

\[
\Pr \left( |L_n(A)| \in \left[ \frac{\hat{N}}{1 + \epsilon}, \frac{\hat{N}}{1 - \epsilon} \right] \right) \geq \frac{3}{4}
\]
Constructing an FPRAS for \#NFA [ACJR21a]

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 the set of initial states
- $F \subseteq Q$ is the set of final states
First component: unroll automaton $A$

Construct $A_{\text{unroll}}$ from $A$:

- for each state $q \in Q$, include copies $q_0, q_1, \ldots, q_n$ in $A_{\text{unroll}}$
- for each transition $(p, a, q) \in \Delta$ and $i \in \{0, 1, \ldots, n-1\}$, include transition $(p_i, a, q_{i+1})$ in $A_{\text{unroll}}$

Besides, eliminate from $A_{\text{unroll}}$ 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 $L(q_i)$ 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 $X \subseteq Q$:

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

Then the task is to compute an estimation of $|L(F^m)|$
Second component: a sketch to be used in the estimation

From now assume that $m = |Q|$, and let

$$\kappa = \left\lceil \frac{nm}{\varepsilon} \right\rceil$$

We maintain for each state $q_i$:

- $N(q^i)$: a $(1 \pm \kappa^{-2})^i$-approximation of $|L(q^i)|$
- $S(q^i)$: a multiset of uniform samples from $L(q^i)$ of size $2\kappa^7$
Second component: a sketch to be used in the estimation

Data structure to be inductively computed:

\[
\text{Sketch}[i] = \{N(q^j), S(q^j) \mid 0 \leq j \leq i \text{ and } q \in Q\}
\]
The algorithm template

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

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

\[ N(F^n) \text{ of } |L(F^n)| \]

We use notation \( N(X^i) \) for an estimation \( |L(X^i)| \)

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 \( i \in \{0, \ldots, n-1\} \) and state \( q \in Q \):
   
   3.1. Compute \( N(q_{i+1}) \) given Sketch\([i]\)
   
   3.2. Sample polynomially many uniform elements from \( L(q_{i+1}) \) 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 $|L(X^i)|$

Recall that

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

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

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

$$|L(X^i)| = \sum_{p \in X} |L(p^i) \setminus \bigcup_{q \in X : q < p} L(q^i)|$$
Computing an estimation $N(X^i)$ of $|L(X^i)|$

We have that:

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

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

So we will use the following approximation:

$$= \sum_{p \in X} |L(p^i)| \frac{|L(p^i) \setminus \bigcup_{q \in X : q < p} L(q^i)|}{|L(p^i)|}$$
Computing an estimation 
\[ N(X^i) \text{ of } |L(X^i)| \]

We have that:
\[
|L(X^i)| = \sum_{p \in X} |L(p^i) \setminus \bigcup_{q \in X : q < p} L(q^i)| \\
= \sum_{p \in X} |L(p^i)| \frac{|L(p^i) \setminus \bigcup_{q \in X : q < p} L(q^i)|}{|L(p^i)|}
\]

So we will use the following approximation:
\[
= \sum_{p \in X} |L(p^i)| \setminus \bigcup_{q \in X : q < p} L(q^i)
\]
Computing an estimation

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

We have that:

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

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

So we will use the following approximation:

$$= \sum_{p \in X} \mathcal{N}(p^i) \frac{|L(p^i) \setminus \bigcup_{q \in X : q < p} L(q^i)|}{|L(p^i)|}$$
Computing an estimation $N(X^i)$ of $|L(X^i)|$

We have that:

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

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

So we will use the following approximation:

$$N(X^i) = \sum_{p \in X} N(p^i) \frac{|S(p^i) \setminus \bigcup_{q \in X : q < p} L(q^i)|}{|S(p^i)|}$$
Computing an estimation
\[ N(X^i) \text{ of } |L(X^i)| \]

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

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

What guarantees that \( N(X^i) \) is a good estimation of \( |L(X^i)| \)?
An invariant to be maintained

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

$$\left| \frac{|L(p^i) \setminus \bigcup_{q \in X} L(q^i)|}{|L(p^i)|} - \frac{|S(p^i) \setminus \bigcup_{q \in X} L(q^i)|}{|S(p^i)|} \right| < \frac{1}{\kappa^3}$$
The use of the main property

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

Lemma: If $E(i)$ holds and $N(p^i)$ is a $(1 \pm \kappa)^i$-approximation of $|L(p^i)|$ for every $p \in Q$, then $N(X^i)$ is a $(1 \pm \kappa^{-2})^{i+1}$-approximation of $|L(X^i)|$ 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 $|L(p^0)|$ for every $p \in Q$

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

Then $N(X^0)$ is a $(1 \pm \kappa^{-2})$-approximation of $|L(X^0)|$ 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:

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

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

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

- So that \( |L(p^1)| = |L(Y)| + |L(Z)| \)
The use of the main property

For $p \in Q$, define:

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

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

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

$$N(Y) + N(Z) \text{ is a } (1 \pm \kappa^{-2})\text{-approximation of } N(p^1) = |L(Y)| + |L(Z)|$$
Main property: a summary

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

\[ \downarrow \]

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

\[ \downarrow \]

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

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

\[ N(p^1) \text{ is a } (1 \pm \kappa^{-2})^1\text{-approximation of } |L(p1)| \text{ for every } p \in Q \]
Main property: a summary

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

\[ \Downarrow \]

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

\[ \Downarrow \]

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

where $R_b(p^2) = \{ q^1 \ | \ (q^1, b, p^2) \text{ is a transition in } A_{\text{unroll}} \}$
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 $|L(F^n)|$.

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 $L(q^{i+1})$
- $S(q^{i+1})$ is computed assuming that $N(q^{i+1})$ and 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 $N(q^0) = |L(q^0)| = 1$ and $S(q^0) = L(q^0) = \{\lambda\}$
3. For each $i \in \{0, \ldots, n-1\}$ and state $q \in Q$:
   
   3.1. Compute $N(q_{i+1})$ given $\text{Sketch}[i]$
   
   3.2. Sample polynomially many uniform elements from $L(q^{i+1})$ using $N(q^{i+1})$ and $\text{Sketch}[i]$, and let $S(q^{i+1})$ be the multiset of uniform samples obtained

4. Return an estimation of $|L(F^n)|$ given $\text{Sketch}[n]$
Sampling from $q^{i+1}$

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

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

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

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

$P_0$ and $P_1$ are sets of states at layer $i$
Sampling from $q^{i+1}$

$P_0$ and $P_1$ are sets of states at layer $i$

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

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

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

Previous procedure works for every set of states $P^{i+1}$:

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

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

1. prob = $\varphi_0$
2. $P^{i+1} = \{q^{i+1}\}$
3. for $j = i + 1$ to 1 do
   3.1. $P_{j,0} = \{p^{j-1} | \exists r^j \in P^j : (p^{j-1}, 0, p^j) \text{ is a transition in } A_{unroll}\}$
   3.2. $P_{j,1} = \{p^{j-1} | \exists r^j \in P^j : (p^{j-1}, 1, p^j) \text{ is a transition in } A_{unroll}\}$
   3.3. Generate $b \in R_i \in \{0, 1\}$ with probability $\frac{N(P_{j,b})}{N(P_{b,0}) + N(P_{b,1})}$
   3.4. $w^{j-1} = bw^j$
   3.5. $P^{j-1} = P_{j,b}$
   3.6. prob = prop \cdot $\frac{N(P_{j,b})}{N(P_{b,0}) + N(P_{b,1})}$
4. reject with probability $1 - \text{prob}$
5. return $w^0$
As before ...

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

\[ \Pr(\text{the output of the procedure is } x) \]

\[ = \Pr(w^0 = x \land \text{the procedure does not reject}) \]

\[ = \Pr(\text{the procedure does not reject} \mid w^0 = x) \Pr(w^0 = x) \]

\[ = \left( \prod_{j=1}^{i+1} \frac{N(P_j, x_j)}{N(P_j, 0)\phi + N(P_j, 1)} \right)^{-1} \cdot \phi_0 \cdot \left( \prod_{j=1}^{i+1} \frac{N(P_j, x_j)}{N(P_j, 0) + N(P_j, 1)} \right) \]

\[ = \phi_0 \]
The value of the initial probability $\varphi_0$

**Lemma:** Assume that $\mathcal{E}(j)$ holds for each $j < i + 1$. If $\varphi_0 = \frac{e^{-5}}{N(q^{i+1})}$, then

- $\text{prob} \leq 1$ in each step in the loop
- $\Pr(\text{procedure rejects}) \leq 1 - e^{-9}$
- $\Pr(w^0 = x) = \frac{e^{-5}}{N(q^{i+1})}$ for every $x \in L(q^{i+1})$
Bounding the probability of breaking the main assumption

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

$$\left| \frac{|L(q^i) \setminus \bigcup_{p \in P} L(p^i)|}{|L(q^i)|} - \frac{|S(q^i) \setminus \bigcup_{p \in P} L(p^i)|}{|S(q^i)|} \right| < \frac{1}{\kappa^3}$$
Bounding the probability of breaking the main assumption

By using Hoeffding’s inequality, it is possible to obtain that:

$$\Pr(E(0) \land \cdots \land E(n)) \leq 1 - e^{-\kappa}$$
The complete algorithm: final comments [ACJR21a]

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 $\text{poly}(m, n, \frac{1}{\varepsilon})$
Back to conjunctive queries

The ideas used for the case of NFA can be extended to the case of TA

**Theorem [ACJR21b]:** #TA admits an FPRAS

**Theorem [ACJR21b]:** The problem of counting the number of answers to an acyclic conjunctive query admits an FPRAS

- The same holds for each class of conjunctive queries with bounded hypertree width
Research questions
• Development of a general theory for estimation in query optimization [HYPM19]
  • Which estimator should be used given a budget? What is an appropriate notion of budget? What are optimal estimators?

• Understand for which relational algebra operators and aggregates it is possible to develop sampling techniques with (strong) guarantees
  • Develop (very) efficient algorithms to compute these estimators
  • Understand the complexity of computing such estimators (fine-grained complexity)
• Understand for which relational algebra operators and aggregates it is not possible to develop sampling techniques with (strong) guarantees
  • What can of guarantees can be provided in these cases?

• Could sample techniques be used for some fundamental tasks for $K$-relations? For first-order logic with semiring semantics?

• Does #CFG admits an FPRAS?
Thanks!
Bibliography


Bibliography


Bibliography
