Tractable Approximate Counting for CQs

报告人：陈鹏宇

哈尔滨工业大学

2023 年 4 月 20 日

Content

Conjunctive Query (CQ)

CQ $Q(\bar{x}) \leftarrow R_1(\overline{y_1}), R_2(\overline{y_2}), \ldots, R_n(\overline{y_n})$, and database $D = (R_1^D, \ldots R_n^D)$

Homomorphism $h: \boldsymbol{V} \cup \boldsymbol{C} \rightarrow \boldsymbol{C}$ s.t.

1. $h(c) = c$ for all $c \in \boldsymbol{C}$
2. $h(\overline{y_i}) = \left(h(y_{i,1}), \ldots, h(y_{i,k_i})\right) \in R_i^D$ for all $i \in [n]$

Answers to a CQ

$$Q(D) = \{h(\bar{x}) \in \boldsymbol{C}^m \mid h \text{ a homomorphism}\}$$

Conjunctive Query Complexity

Query Evaluation Problem: is $Q(D) = \emptyset$?

- NP-Hard in general, poly-time if $tw(Q)$ or $htw(Q)$ bounded.
- No poly-time for if $tw(\mathcal{G})$ unbounded.

Counting Problem: Output $|Q(D)|$

- #P-Hard in general
- #P-Hard even if $tw(Q) = 1$ [Pichler, Skritek '13]

Approximate Counting: Output R s.t. $\mathrm{R} = (1 \pm \epsilon)|Q(D)|$

- No poly-time in general unless P=NP.
- What about bounded (hyper)-treewidth?

Content

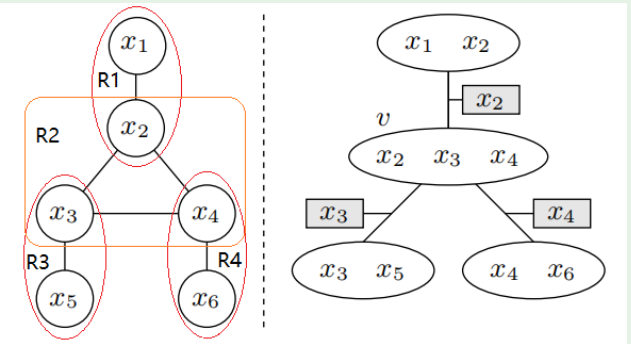
Tree Decomposition

Definition 7.2: Consider a graph $G = \langle V, E \rangle$. A tree decomposition of G is a tree structure T where each node of T is a subset of V, such that:

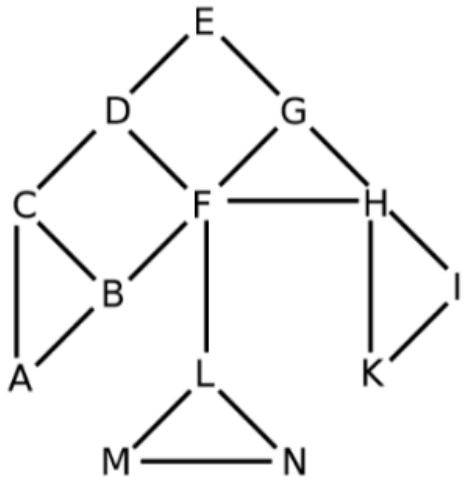
- The union of all nodes of T is V.
- For each edge $(v_1 \to v_2) \in E$, there is a node N in T such that $v_1, v_2 \in N$.
- For every vertex $v \in V$, the set of nodes of T that contain v form a subtree of T; equivalently: if two nodes contain v, then all nodes on the path between them also contain v (connectedness condition).

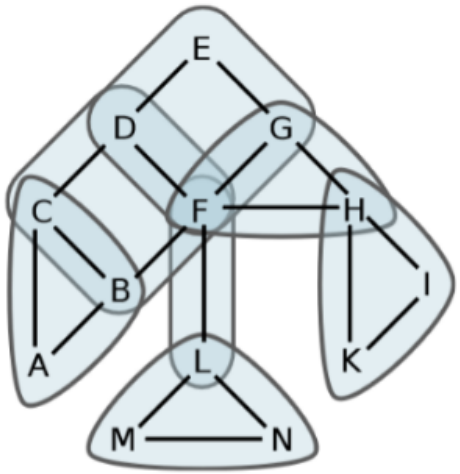
例 1 (Treewidth=2)

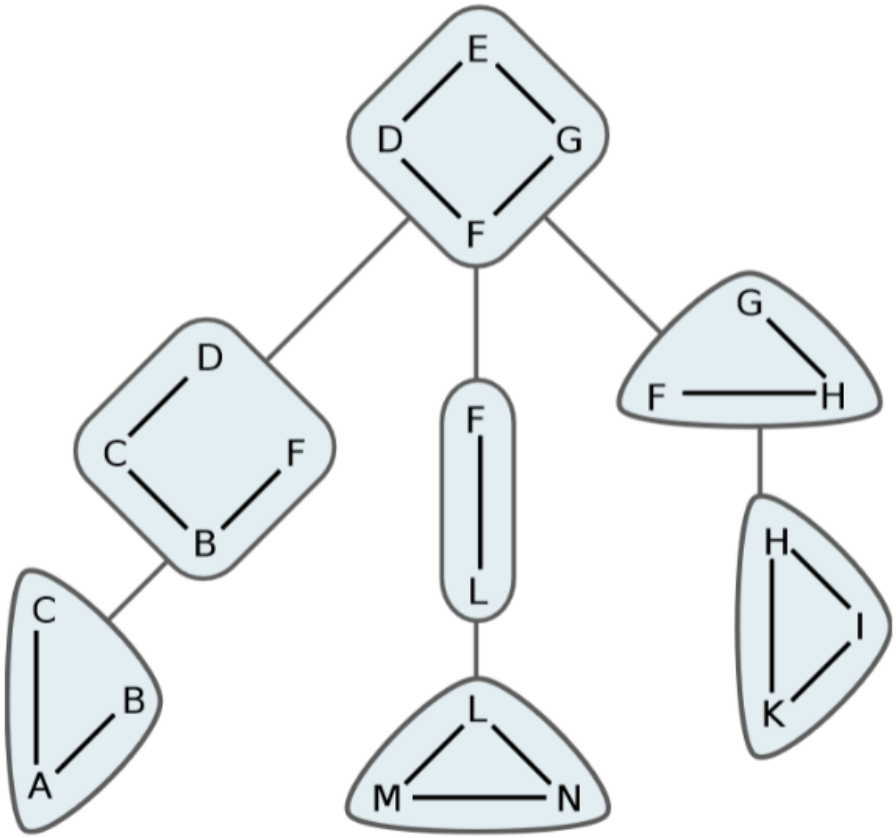
$$\varphi(x_1, x_5, x_6) \leftarrow R_1(x_1, x_2), R_2(x_2, x_3, x_4), R_3(x_3, x_5), R_4(x_4, x_6)$$



Tree Decomposition







Tree Width

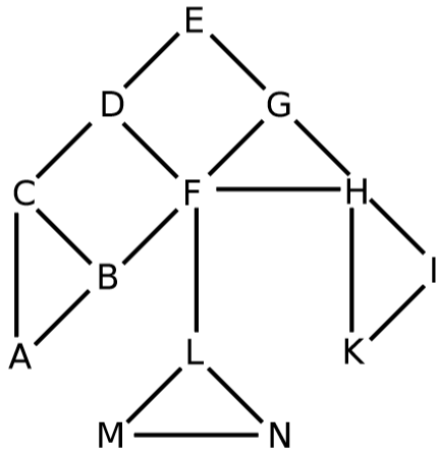
The treewidth of a graph defines how "tree-like" it is:

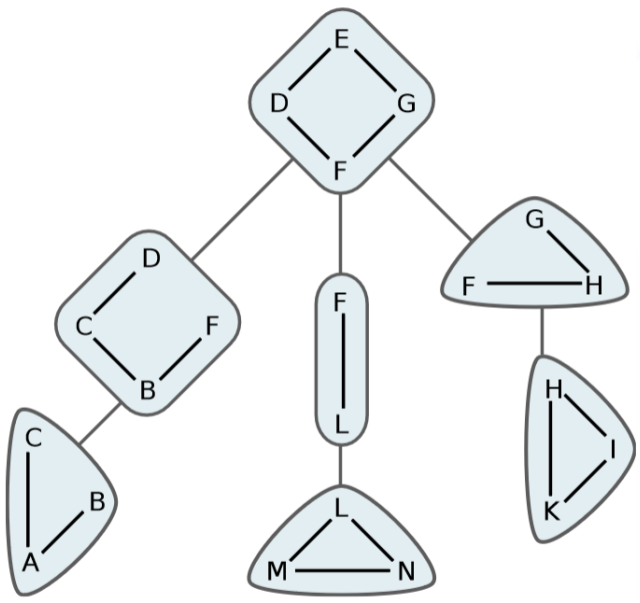
Definition 7.3: The width of a tree decomposition is the size of its largest bag minus one.
The treewidth of a graph G, denoted $\mathrm{tw}(G)$, is the smallest width of any of its tree decompositions.

Simple observations:

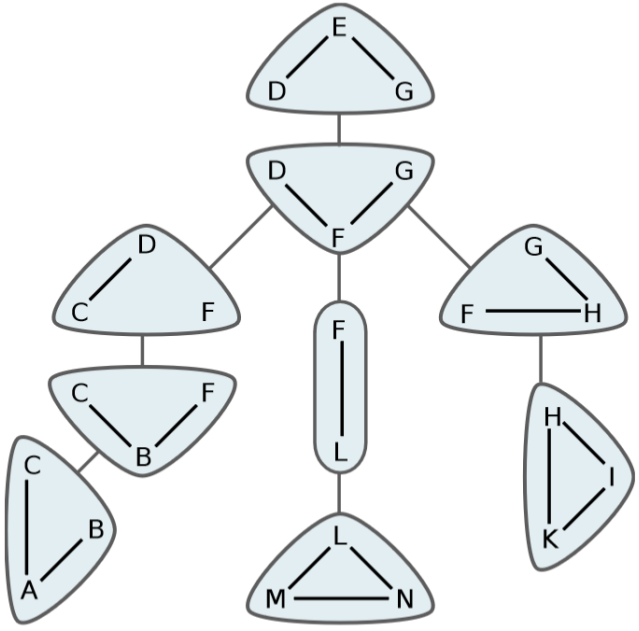
- If G is a tree, then we can decompose it into bags that contain only one edge
  $\leadsto$ trees have treewidth 1
- Every graph has at least one tree decomposition where all vertices are in one bag
  $\leadsto$ maximal treewidth = number of vertices -1

Tree Width





$\rightsquigarrow$ tree decomposition of width 3



$\rightsquigarrow$ tree decomposition of width 2 = treewidth of the example graph

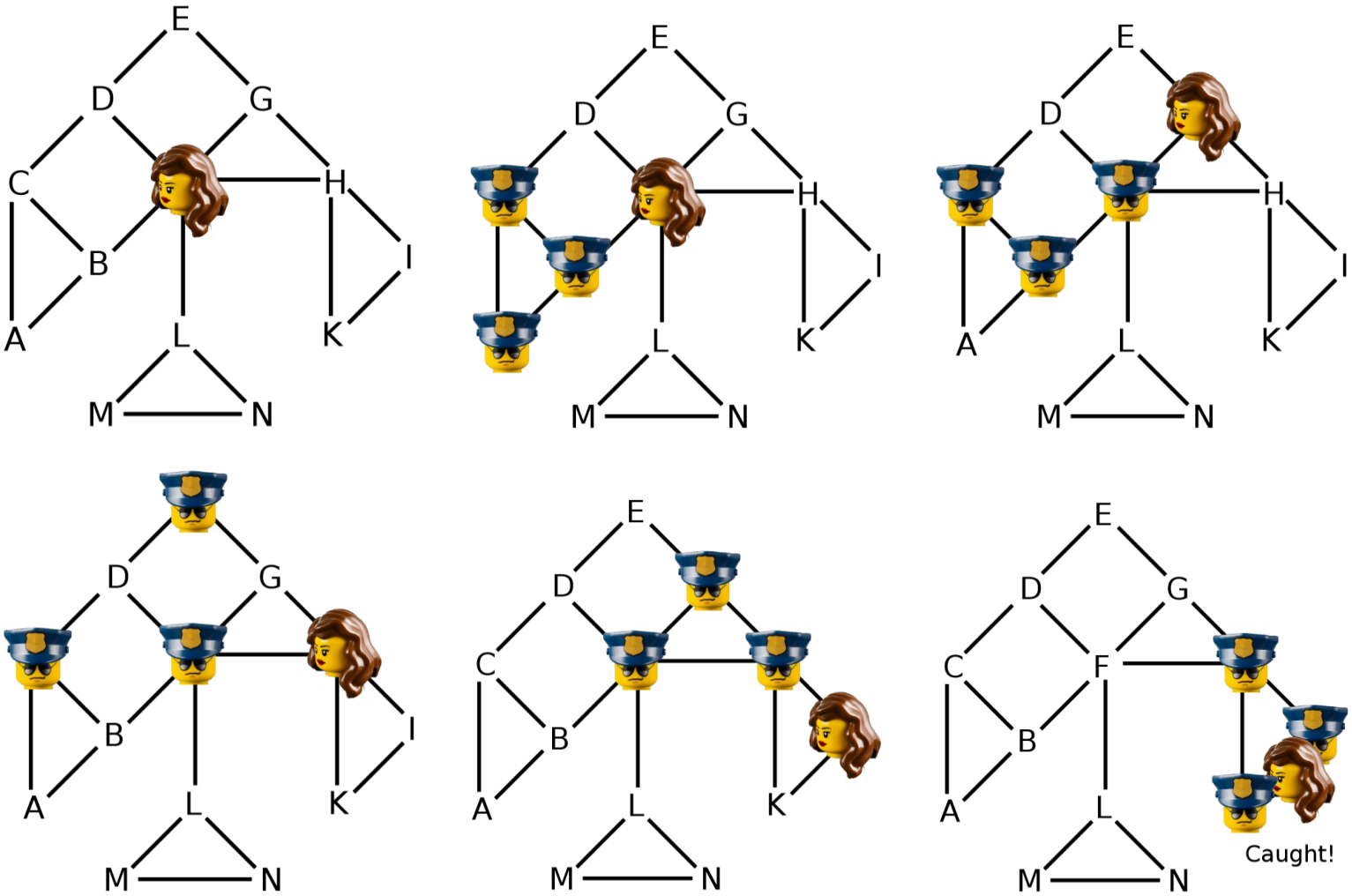
Tree Width via Games

Seymour and Thomas [1993] gave
an alternative characterisation of treewidth:

The Cops-and-Robber Game

- The game is played on a graph G
- There are k cops and one robber that may be positioned at vertices
- In the first turn, the robber places herself at an arbitrary vertex of the graph; the cops are all in a "helicopter" (i.e., not yet placed on any vertex)
- In each turn:
  - one of the cops can decide to "fly" to an arbitrary vertex in the graph
  - if the moving cop is already in the game, he is lifted from his vertex
  - before "landing" (i.e. positioning the cop at his new vertex), the target vertex is announced to the robber (the robber sees the helicopter approaching)
  - the robber can run along the edges of the graph, as far as she likes, as long as she does not use any vertex currently occupied by a cop
  - the moving cop arrives at his destination vertex
- The cops' goal is to catch the robber; the robber's goal is never to be caught

Tree Width via Games



Theorem 7.6 (Seymour and Thomas): A graph G is of treewidth $\leq k-1$ if and only if k cops have a winning strategy in the cops & robber game on G.

Summary of Tree Width

Graphs of bounded treewidth as a generalisation of (undirected) trees:

- Trees have treewidth 1
- Graphs of higher treewidth resemble trees with "thicker branches"
- It is (in theory) not hard to check if a graph has treewidth $\leq k$ for some k
- It is (in theory) not hard to answer BCQs whose primal graph has a bounded treewidth

Practically feasible only for lower treewidths

However, bounded treewidth does not generalise the notion of hypergraph acyclicity (acyclic families of hypergraphs may have unbounded treewidth)

Is there a better notion of tree-likeness for hypergraphs?

Content

Query Width

Idea of Chekuri and Rajamaran [1997]:

- Create tree structure similar to tree decomposition
- But consider bags of query atoms instead of bags of variables
- Two connectedness conditions:
  (1) Bags that refer to a certain variable must be connected
  (2) Bags that refer to a certain query atom must be connected

Query width: least number of atoms needed in bags of a query decomposition

Theorem 8.1: Given a query decomposition for a BCQ, the query answering problem can be decided in time polynomial in the query width.

Theorem 8.2 (Gottlob et al. 1999): Deciding if a query has query width at most k is NP-complete.

In particular, it is also hard to find a query decomposition

$\rightsquigarrow$ Query answering complexity drops from NP to P ...
... but we need to solve another NP-hard problem first!

Hyper Tree Decomposition[1]

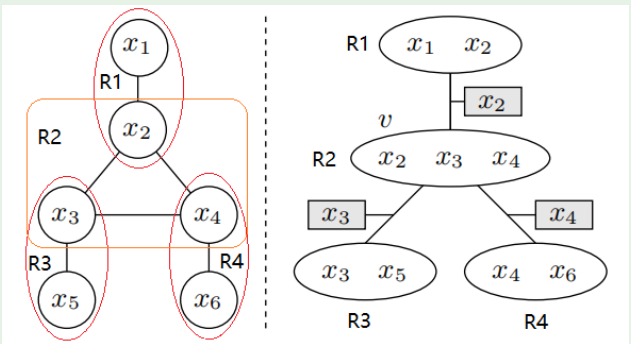
Definition 8.3: Consider a hypergraph $G = \langle V, E \rangle$. A hypertree decomposition of G is a tree structure T where each node n of T is associated with a bag of variables $B_n \subseteq V$ and with a set of edges $G_n \subseteq E$, such that:

- T with B_n yields a tree decomposition of the primal graph of G.
- For each node n of T:
  - (1) the vertices used in the edges G_n are a superset of B_n,
  - (2) if a vertex v occurs in an edge of G_n and this vertex also occurs in B_m for some node m below n in T, then $v \in B_n$.

The width to T is the largest number of edges in a set G_n.
The hypertree width of G, hw(G), is the least width of its hypertree decompositions.

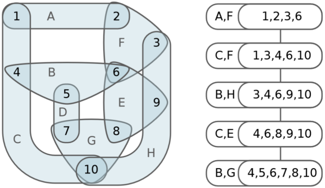
例 2 (Treewidth=2, Hyperwidth=1)

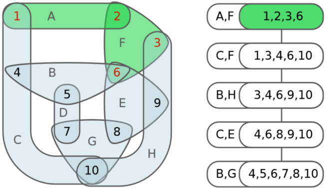
$$\varphi(x_1, x_5, x_6) \leftarrow R_1(x_1, x_2), R_2(x_2, x_3, x_4), R_3(x_3, x_5), R_4(x_4, x_6)$$

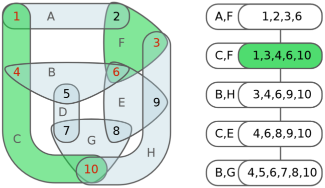


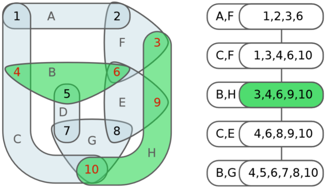
[1] (2) is the "special condition" : without it we get the generalized hypertree width

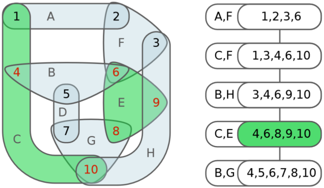
(Generalized) Hyper Tree Decomposition

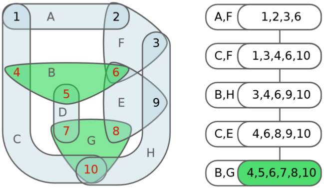




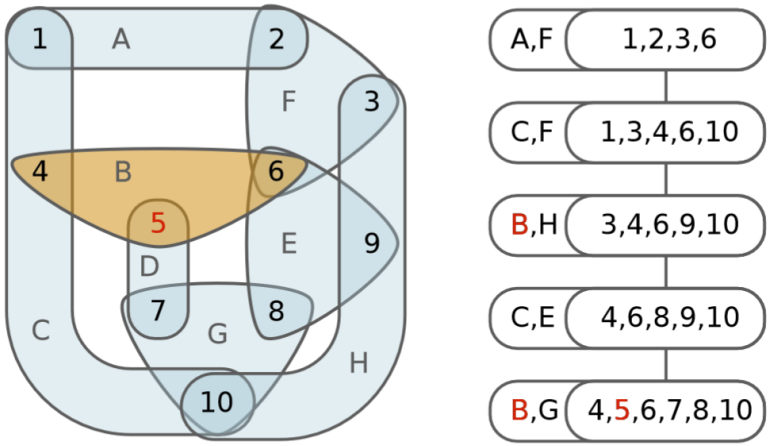




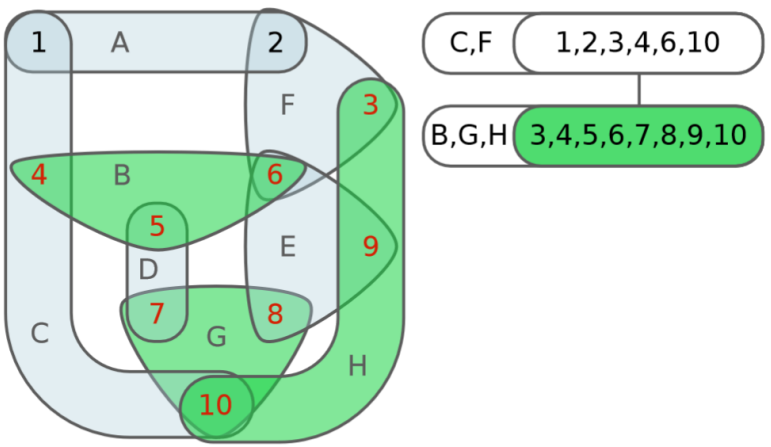




(Generalized) Hyper Tree Decomposition



Special condition violated $\rightsquigarrow$ no hypertree decomposition
$\rightsquigarrow$ But generalised hypertree decomposition of width 2



Special condition satisfied $\rightsquigarrow$ hypertree decomposition of width 3

(Generalized) Hyper Tree Results

- Relationships of hypergraph tree-likeness measures:
  generalised hypertree width $\leq$ hypertree width $\leq$ query width
  (both inequalities might be $<$ in some cases)
- Acyclic graphs have hypertree width 1
- Deciding "query width $< k$?" is NP-complete
- Deciding "generalised hypertree width < 4?" is NP-complete
- Deciding "hypertree width $< k$?" is polynomial (LOGCFL)
- Hypertree decompositions can be computed in polynomial time if k is fixed

Theorem 8.9: For a BCQ of (generalised) hypertree width k, query answering can be decided in polynomial time, and is complete for LOGCFL.

Summary of (Hyper) Tree Width

Treewidth

Graph $G(Q) = (V, E)$

- $V :=$ variables in Q
- $(u, v) \in E$ if $u, v \in \bar{y}_i$ for some i
- $tw(G) :=$measures how close G is to a tree
- $tw(G) = 1 \leftrightarrow G$ acyclic

$$tw(Q) = tw(G(Q))$$

Hyper-Treewidth

Hyper-Graph $H(Q) = (V, \mathcal{E})$

- $V :=$ variables in Q
- $\mathcal{E} := \{\{v \mid v \in \bar{y}_i\} \mid i \in [n]\}$
- $htw(G) :=$ how close H is to acyclic
  - [Gottlob, Leone, Scarcello '02]

$$htw(Q) \leq tw(Q)$$

Content

Main Results

Theorem: Let Q be a CQ with $htw(Q) = O(1)$, and D a database. Then $Q(D)$ admits an FPRAS.

- Extends to unions of CQ's with bounded htw
- $htw(Q) \leq tw(Q)$, so we obtain an FPRAS for bounded treewidth.

Proof Approach:

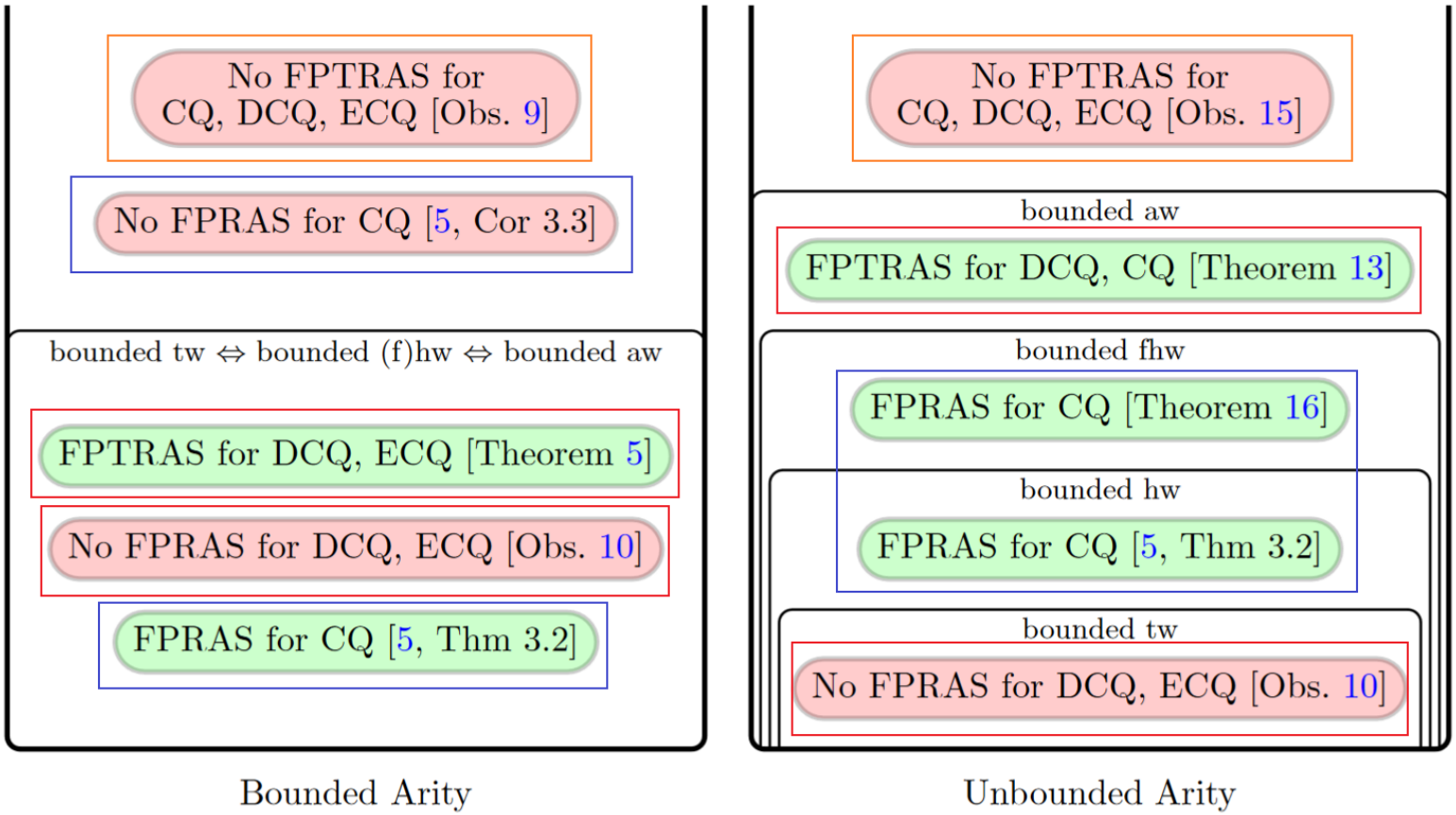
1. Construct automata A, with $|\mathcal{L}_n(A)| = |Q(D)|$, where $\mathcal{L}_n(A)$:=set of inputs of size n accepted by A.
2. Design FPRAS for $\mathcal{L}_n(A)$

- FPRAS[2]：时间为 $\mathrm{poly}(|\mathcal{D}| + |\varphi|, \varepsilon^{-1}, \log \delta^{-1})$
- FPTRAS[3]：时间为 $f(|\varphi|) \cdot \mathrm{poly}(|\mathcal{D}|, \varepsilon^{-1}, \log \delta^{-1})$

[2] fully polynomial randomised approximation scheme

[3] fixed-parameter tractable randomised approximation scheme

Main Results



- 橙色：宽松条件下的复杂性下界
- 红色：#DCQ、#ECQ 的 (参数化) 算法及复杂性下界
- 蓝色：#CQ 的 (FPRAS) 算法及复杂性下界

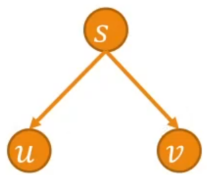
Content

Tree Automata (TA)

"Branching" Generalization of NFA's.

- Tree automata T accepts a language $\mathcal{L}(\mathrm{T})$ of *ordered trees.*
- NFA: one state to another
- TA: one state to **one or more** child states.

$(s, u, v) \in \Delta$ means:



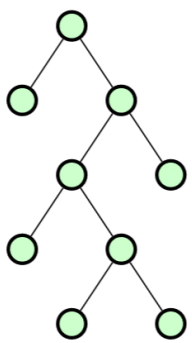
$s \in \Delta$ means s a leaf

This talk, Binary Tree Automata: $T = (S, \Delta, s_{init})$

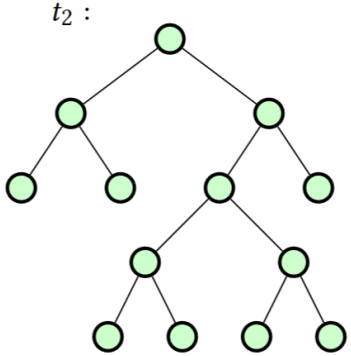
1. $S :=$ set of states
2. $\Delta \subseteq (S \times S \times S) \times S$ is a *transition relation*
3. $s_{init} \in S$ is the *initial state.*

$$
\begin{aligned}
\mathcal{T} := \quad & s \rightarrow sq \\
& s \rightarrow qs \\
& s \rightarrow rr \\
& r \rightarrow qq \\
& q \rightarrow qq \\
& q \rightarrow \cdot \\
& s_{\text{init}} = s
\end{aligned}
$$

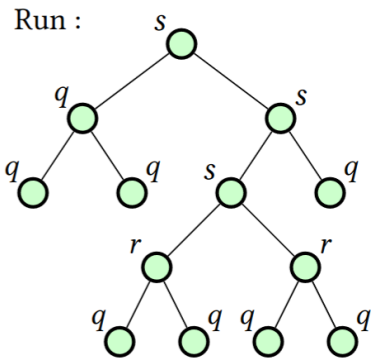
t_1 :



t_2 :



Run :

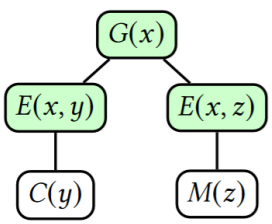


From Conjunctive Query to Tree Automata

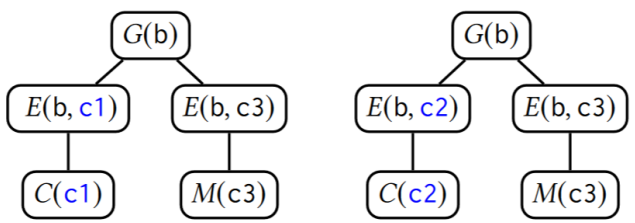
Theorem: Given Q, D with $htw(Q) = O(1)$, there is a poly($|Q|, |D|$)-time algorithm which outputs a TA T and $n = \text{poly}(|Q|, |D|)$ with:

$$|\mathcal{L}_n(T)| = |Q(D)|$$

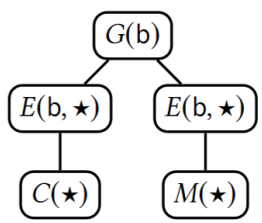
- $Q(x) \leftarrow G(x), E(x, y), E(x, z), C(y), M(z)$



(a) A join tree.



(b) Two witness trees for the answer b.



(c) An anonymous tree for b.

Content

Counting via Sampling

Dynamic Programming via sampling

- $\mathcal{L}_i(s) :=$ trees of size i accepted by T with initial state $s \in S$

$$|\mathcal{L}_i(s)| = \left| \bigcup_{\substack{(s,x,y)\in\Delta \\ j+k=i-1}} \mathcal{L}_j(x) \times \mathcal{L}_k(y) \right|$$

- **Challenge:** union non-disjoint, (size of union) $\neq$ (sum of sizes)
- **Solution:** Estimate $|\mathcal{L}_i(s)|$ via *sample sets* $\mathcal{S}_j(x)$ *of* $\mathcal{L}_j(x)$ for all $j < i, x \in S$
  - $\mathcal{S}_j(x) :=$ uniform samples from $\mathcal{L}_j(x)$.
  - Given sample sets $\mathcal{S}_j(x)$, straightforward to estimate $|\mathcal{L}_i(s)|$.
  - Similar approach in [ACJR'19] for #NFA.

Counting via Sampling

$$|\mathcal{L}_i(s)| = \left| \bigcup_{\substack{(s,x,y)\in\Delta \\ j+k=i-1}} \mathcal{L}_j(x) \times \mathcal{L}_k(y) \right|$$

- **Main Challenge:** How to generate samples from $\mathcal{L}_j(x)$ to form $\mathcal{S}_j(x)$?
  - **Karp-Luby Sampling** in QPRAS of [Gore, Jerrum, Kannan, Sweedyk, and Mahaney '97]
    1. Sample a $\mathcal{L}_j(x) \times \mathcal{L}_k(y)$ proportional to $|\mathcal{L}_j(x) \times \mathcal{L}_k(y)|$
    2. Recursively Sample $e \sim \mathcal{L}_j(x) \times \mathcal{L}_k(y)$
    3. Accept e with probability $= \frac{1}{|\text{other sets containing } e|}$
- **Our Solution:** Partition-and-Sample
  1. Partition $\bigcup \mathcal{L}_j(x) \times \mathcal{L}_k(y)$ into disjoint $T_1, T_2, \ldots, T_k$
  2. Sample T_i proportional to $\approx |T_i|$
  3. Recursively sample $e \sim T_i$

$$\sum_{j=1}^{k} \widetilde{N}(T_j) \left(\frac{\left| \widetilde{T}_j \setminus \bigcup_{j'<j} T_{j'} \right|}{\left| \widetilde{T}_j \right|} \right)$$

- **Technical contribution:** How to partition, and how to estimate $|T_i|$.
  - Much harder for TA than NFA

Summary and Open Problem

NFA

$\mathcal{L}_n(N) :=$ size n words accepted by N

- Decision problem = PTIME
- Exact counting = $\#P$-Hard
- Approximate counting = FPRAS

Tree Automata

$\mathcal{L}_n(T) :=$ size n trees accepted by T

- Decision problem = PTIME
- Exact counting = $\#P$-Hard
- Approximate counting = FPRAS

Context Free Grammar

$\mathcal{L}_n(G) :=$ size n words accepted by CFG G

- Decision problem = PTIME
- Exact counting = $\#P$-hard
- Approximate counting = ??