Color Refinement as an Index for Evaluating Acyclic Conjunctive Queries

Cristian Riveros Benjamin Scheidt Nicole Schweikardt

AlMoTh'24

Humboldt-Universität zu Berlin

What the next 20 minutes will be about

The Problem

Query Evaluation

Color Refinement

Main Result: Using the Computed Coloring as an Index

Outlook

Lots of characters

Who could expect to become a godparent?

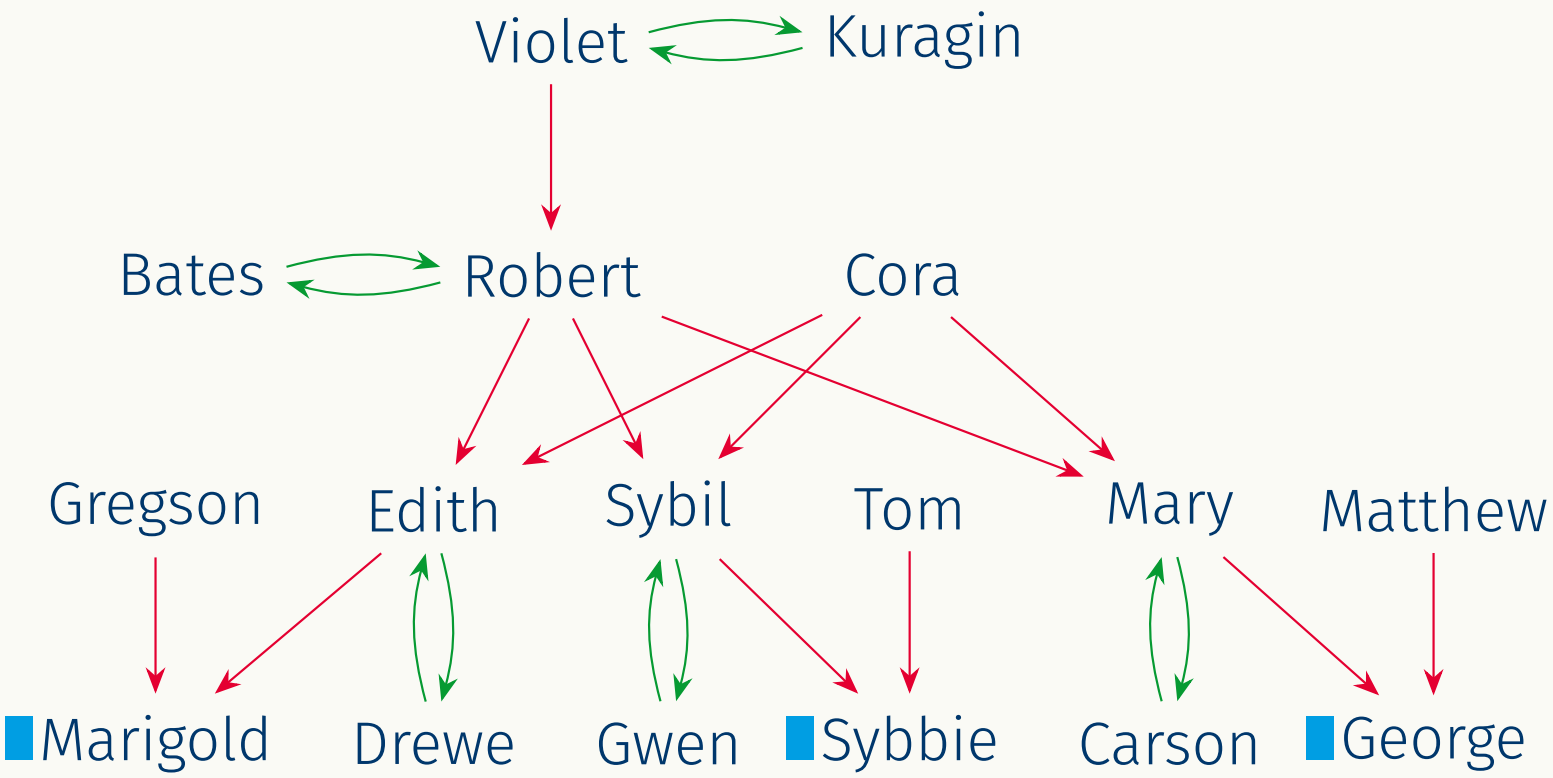
Parent	of
Violet	Robert
Robert	Mary
Robert	Edith
Robert	Sybil
Cora	Mary
Cora	Edith
Cora	Sybil

Parent	of
⋮	⋮
Mary	George
Matthew	George
Sybil	Sybbie
Tom	Sybbie
Edith	Marigold
Gregson	Marigold

Fond	of
Mary	Carson
Carson	Mary
Kuragin	Violet
Violet	Kuragin
Robert	Bates
Bates	Robert
Edith	Drewe
Drewe	Edith
Sybil	Gwen
Gwen	Sybil

Baby	George	Marigold	Sybbie

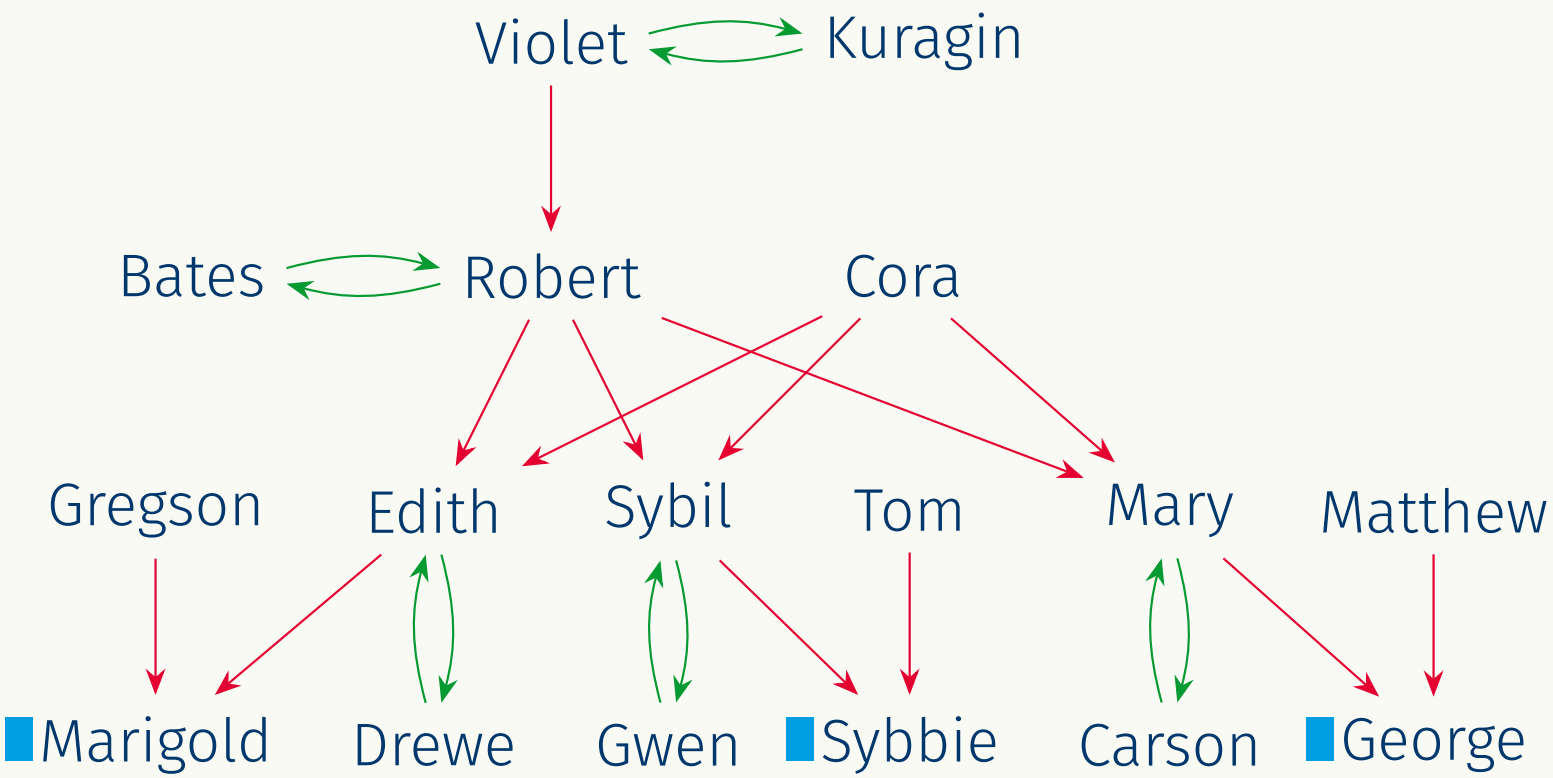
The Database as a Graph



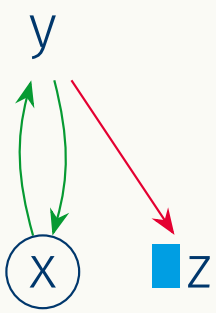
Who could expect to become a godparent?

$$Q_1 \quad := \quad \mathsf{Ans}(x) \leftarrow \mathsf{F}(x,y),\ \mathsf{F}(y,x),\ \mathsf{P}(y,z),\ \mathsf{B}(z)$$

Who could expect to become a godparent?



$\mathcal{Q}_1$:



$$[\![\mathcal{Q}_1]\!](D) \;=\; \{\text{Drewe}, \text{Gwen}, \text{Carson}\}$$

A quick (and simplified) recap on

🌟 Acyclic Conjunctive Queries 🌟

Evaluating Acyclic Conjunctive Queries

Conjunctive Queries (CQ, ACQ)

A CQ $\mathcal{Q}$ with $\text{free}(\mathcal{Q}) = \overline{\mathbf{x}}$ looks like this:

$$\text{Ans}(\overline{\mathbf{x}}) \leftarrow \alpha_1(\overline{\mathbf{x}}_1),\ \alpha_2(\overline{\mathbf{x}}_2),\ \ldots,\ \alpha_n(\overline{\mathbf{x}}_n)$$

$\mathcal{Q}$ is acyclic, if it has a join tree.
In particular, a **binary** $\mathcal{Q}$ is acyclic, if it "looks like a tree".

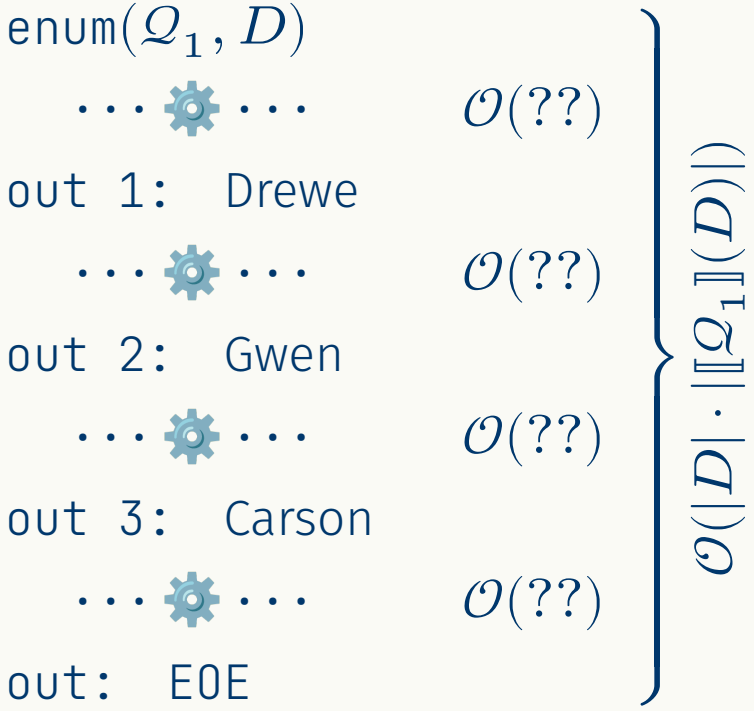
Theorem (Yannakakis 1981)

There exists an algorithm that computes $[\![\mathcal{Q}]\!](D)$ in time $\mathcal{O}(|D| \cdot |[\![\mathcal{Q}]\!](D)|)$ for every database D and every ACQ $\mathcal{Q}$.

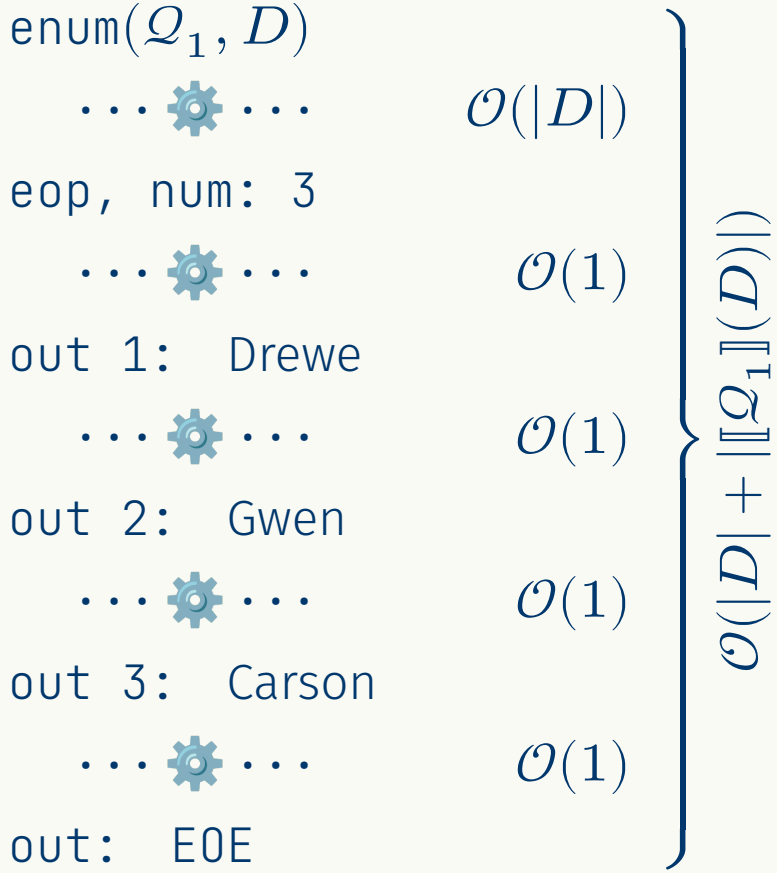
Data Complexity — Assume $\mathcal{Q}$ to be constant.

Constant Delay Enumeration: Refined runtime

Yannakakis



Bagan, Durand, Grandjean



A fancy result

Theorem (Bagan, Durand, Grandjean 2007)

There is an enumeration algorithm that enumerates $[\![\mathcal{Q}]\!](D)$ with constant delay after preprocessing in time $\mathcal{O}(|D|)$, for every D and every free-connex ACQ $\mathcal{Q}$.

Definition (Free-connex ACQs)

A CQ $\mathcal{Q}$ is free-connex acyclic, if $\mathcal{Q}$ and $\mathcal{Q} + R(\overline{\mathbf{x}})$ are acyclic, where $\overline{\mathbf{x}} = \text{free}(\mathcal{Q})$.

In particular: A **binary** $\mathcal{Q}$ is free-connex acyclic, if it is acyclic and $\text{free}(\mathcal{Q})$ induces a connected subgraph.

Can we do better?

Our Idea: Precompute an index

Indexing Phase

Build a data structure DS_D in time t.

Evaluation Phase

Enumerate $[\![\mathcal{Q}]\!](D)$ with constant delay after preprocessing in $\mathcal{O}(|\mathsf{DS}_D|)$.

Theorem

For every ***binary*** *free-connex ACQ $\mathcal{Q}$, we can enumerate $|[\![\mathcal{Q}]\!](D)|$ with constant delay after $\mathcal{O}(|D_{col}|)$ preprocessing.*

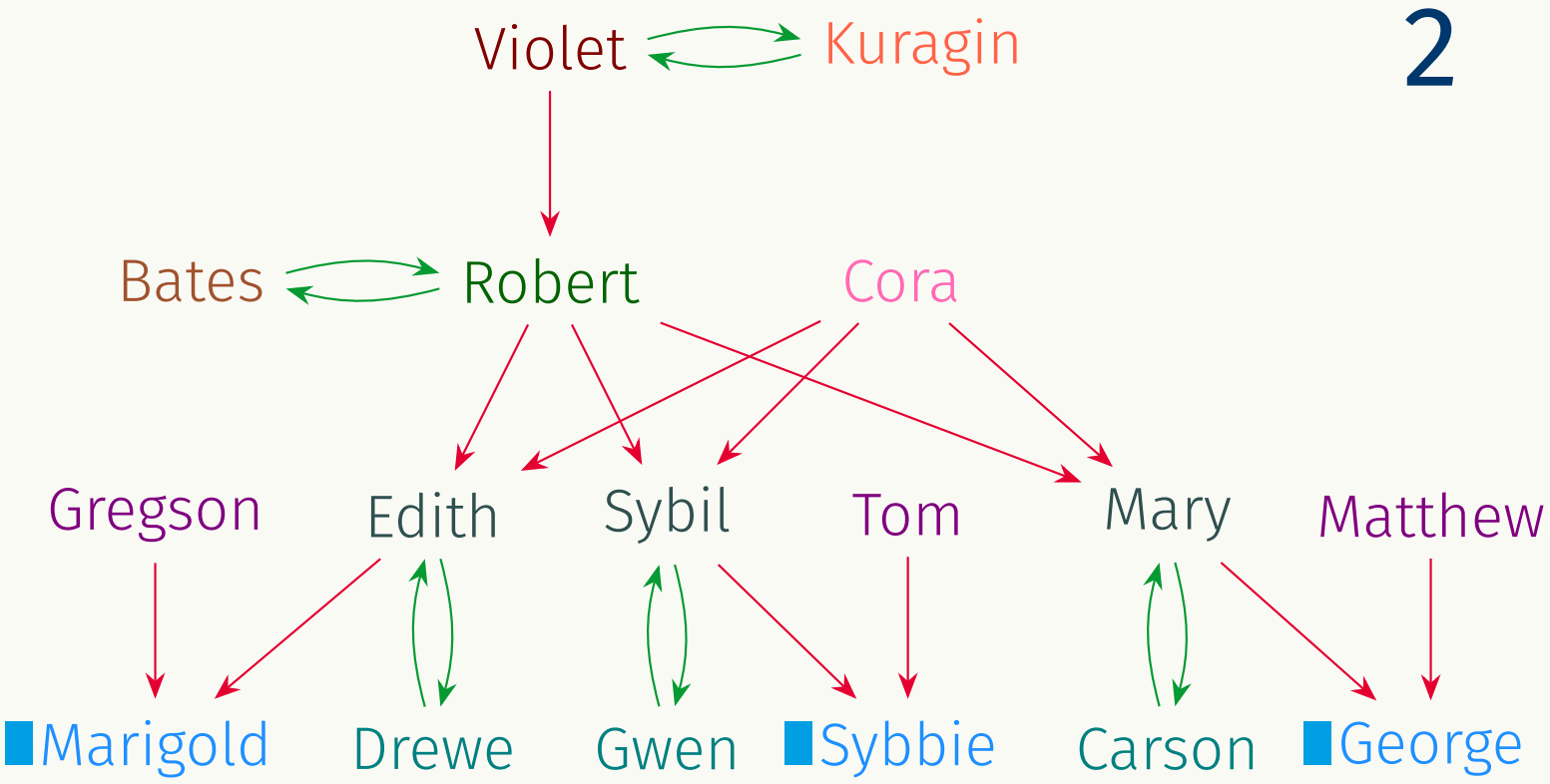
D_{col} is constructed using *Color Refinement*.

Identify similar nodes using Color Refinement

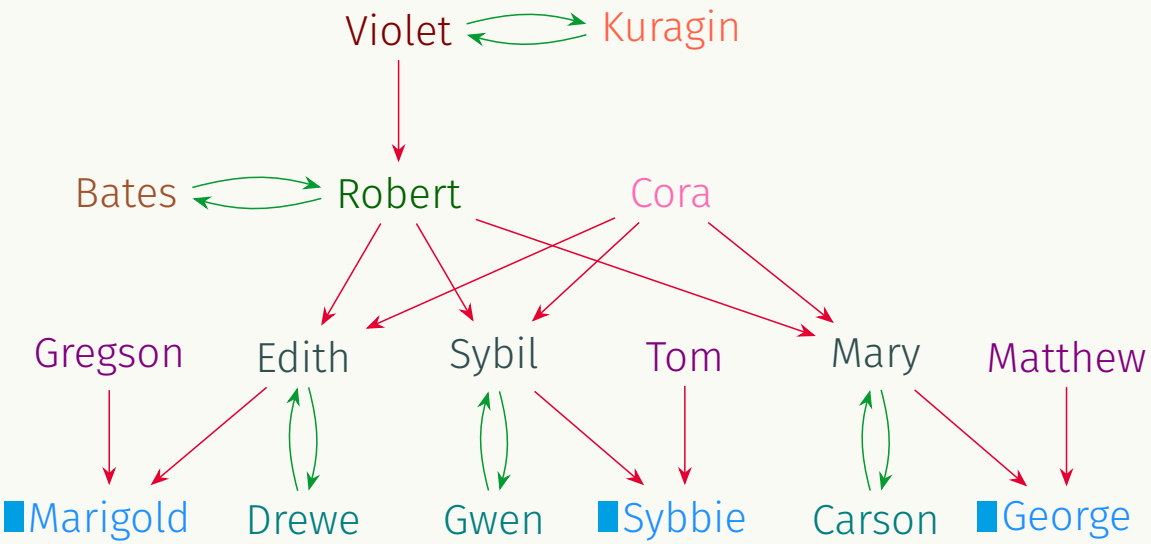
Color Refinement

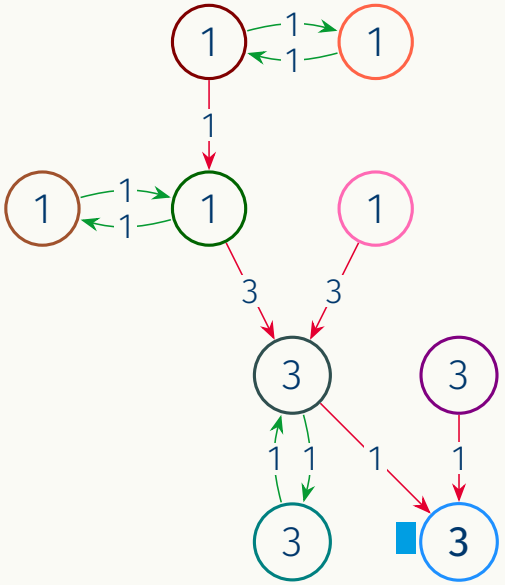
Initial: u, v get different colors, iff different unary relations.

Iterate: u, v get different colors, if colors of neighbors differ.



Create the color database D_{col}

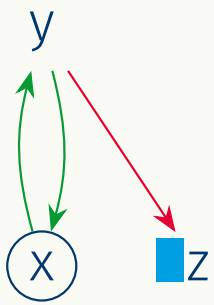




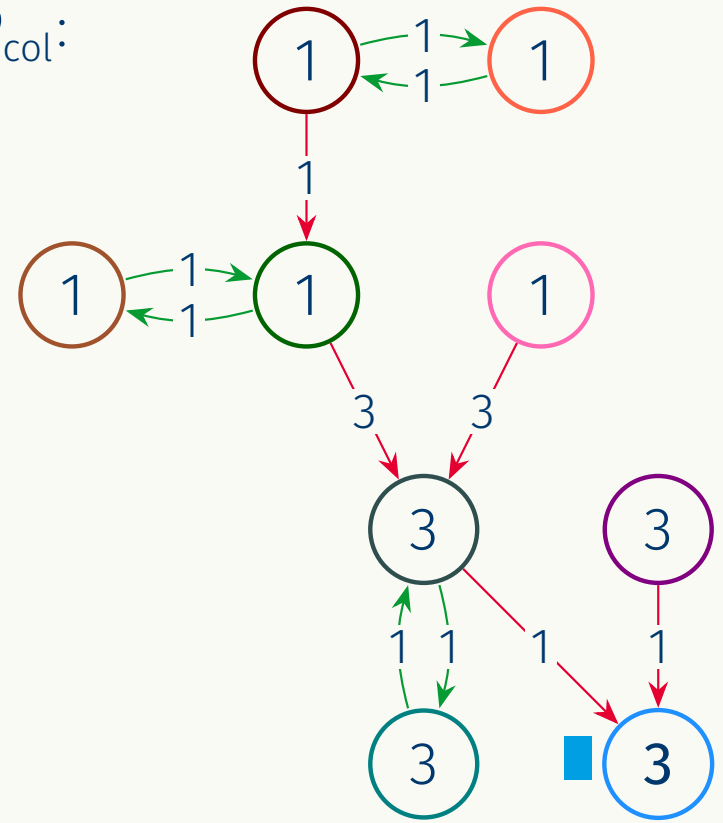
Evaluating queries using D_{col}.

Evaluate $\mathcal{Q}_1$ over D_{col}

$\mathcal{Q}_1$:



D_{col}:



`enum`$(\mathcal{Q}_1, D_{\text{col}})$

⋯ ⚙ ⋯ $\quad \mathcal{O}(|D_{\text{col}}|)$

`eop, num: 1`

⋯ ⚙ ⋯ $\quad \mathcal{O}(1)$

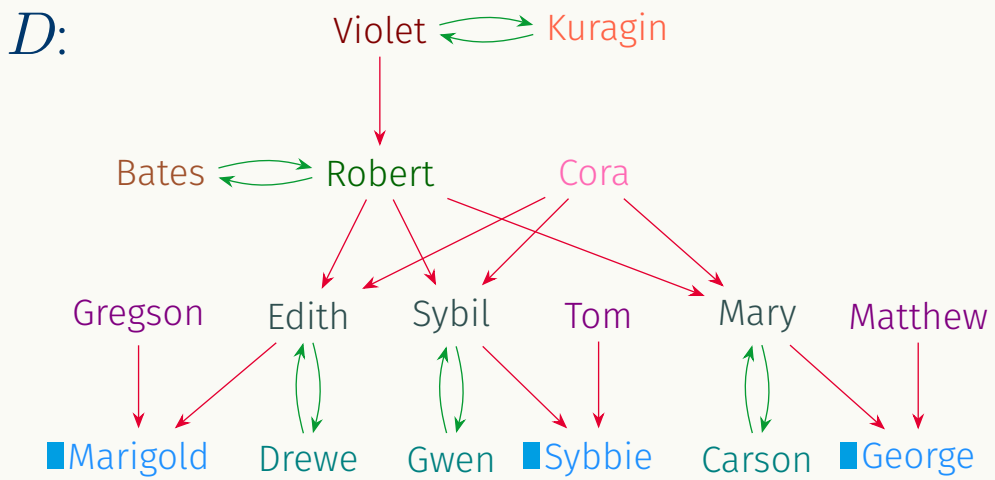
`out 1:` •

`list`$(D, \bullet)$

⋯ ⚙ ⋯ $\quad \mathcal{O}(1)$

`out: EOE`

Evaluate $\mathcal{Q}_1$ over database



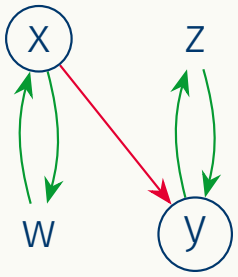
$$\texttt{list}(D, \bullet) \cdots \underset{\mathcal{O}(1)}{\text{⚙}} \cdots \text{Drewe} \cdots \underset{\mathcal{O}(1)}{\text{⚙}} \cdots \text{Gwen} \cdots \underset{\mathcal{O}(1)}{\text{⚙}} \cdots \text{Carson}^{1}$$

Enumerated $[\![\mathcal{Q}_1]\!](D)$ with const. delay after $\mathcal{O}(|D_{\text{col}}|)$ preprocessing!

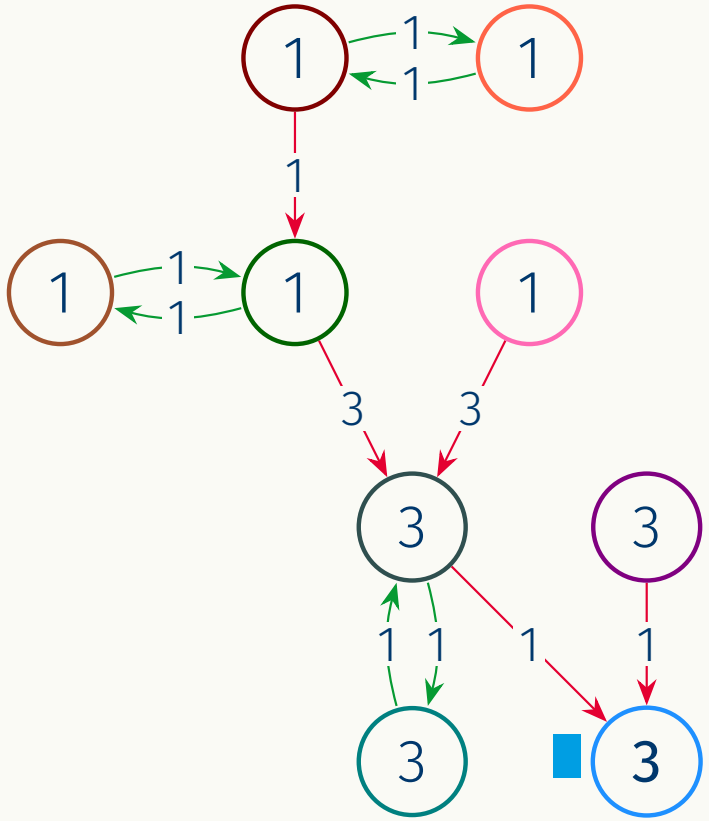
[1]: Need additional data structures to do this with constant delay.

Evaluate $\mathcal{Q}_2$ over D_{col}

$\mathcal{Q}_2$:



D_{col}:



`enum`$(\mathcal{Q}_2, D_{\text{col}})$

⋯ ⚙ ⋯ $\mathcal{O}(|D_{\text{col}}|)$

`eop, num: 2`

⋯ ⚙ ⋯ $\mathcal{O}(1)$

`out 1:` •, •

`list`$(D,$ `P`$(\bullet, \bullet))$

⋯ ⚙ ⋯ $\mathcal{O}(1)$

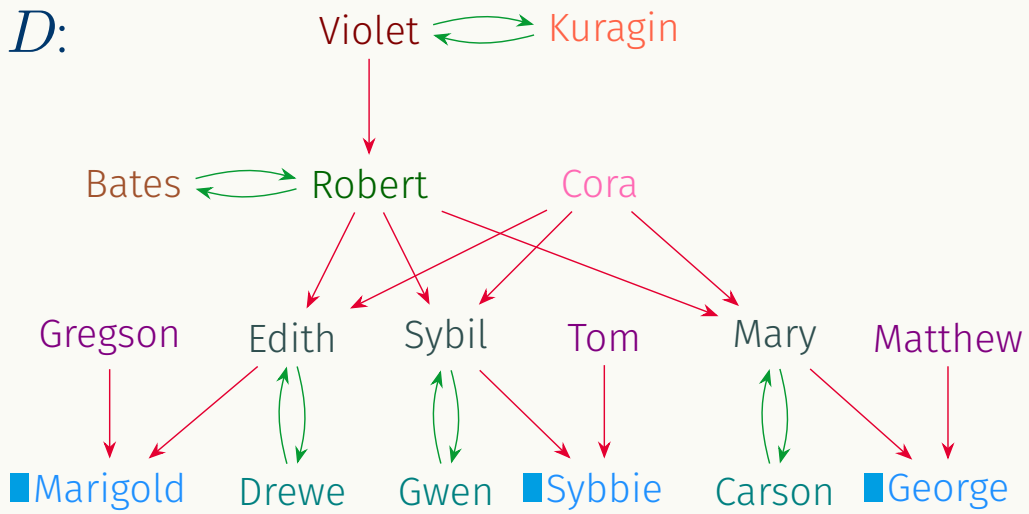
`out 2:` •, •

`list`$(D,$ `P`$(\bullet, \bullet))$

⋯ ⚙ ⋯ $\mathcal{O}(1)$

`out: EOE`

Evaluate Q_2 over database



$\texttt{list}(D, \mathsf{P}(\bullet, \bullet)) \cdots \text{⚙} \cdots (\text{Violet}, \text{Robert})$

$\texttt{list}(D, \mathsf{P}(\bullet, \bullet)) \cdots \text{⚙} \cdots (\text{Robert}, \text{Edith})$

$\cdots \text{⚙} \cdots (\text{Robert}, \text{Sybil})$

$\cdots \text{⚙} \cdots (\text{Robert}, \text{Mary})$

$\mathcal{O}(1)$

(Need additional data structures to do this with constant delay.)

I lied to you. 👉👈

This is just the idea.

It is more complicated — but not much.

Main Result & Outlook

Theorem

For every binary free-connex ACQ $\mathcal{Q}$, after $\mathcal{O}(|D_{col}|)$ preprocessing, we can return the number of result tuples $|[\![\mathcal{Q}]\!](D)|$ and enumerate them with constant delay.

- D_{col} has to be computed only once: $\mathcal{O}(|D| \cdot \log |D|)$.
- $|D_{\text{col}}|$ at most $\mathcal{O}(|D|)$, smaller if db has "good" structure.

Outlook:

- Generalize to arbitrary schemas (not trivial).
- Only d iterations works for queries of height $\leq d$.
- DBs are rarely static. How to avoid recomputing the index?

Further Reading

📝 Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt.
"Constant Delay Enumeration for Conjunctive Queries: a Tutorial".
2020. DOI: 10.1145/3385634.3385636.

📝 Sandra Kiefer.
"The Weisfeiler-Leman Algorithm: An Exploration of Its Power".
2020. DOI: 10.1145/3436980.3436982.

Some Related Work

- Jeroen Bollen, Jasper Steegmans, Jan Van Den Bussche, Stijn Vansummeren.
  “Learning Graph Neural Networks Using Exact Compression”.
  2023. DOI: 10.1145/3594778.3594878.
- Martin Grohe, Kristian Kersting, Martin Mladenov, Erkal Selman.
  “Dimension Reduction via Colour Refinement”.
  2014. DOI: 10.1007/978-3-662-44777-2_42.
- Moe Kayali, Dan Suciu.
  “Quasi-Stable Coloring for Graph Compression: Approximating Max-Flow, Linear Programs, and Centrality”.
  2022. DOI: 10.14778/3574245.3574264.

Other References

- Guillaume Bagan, Arnaud Durand, Etienne Grandjean.
  “On Acyclic Conjunctive Queries and Constant Delay Enumeration”.
  2007. DOI: 10.1007/978-3-540-74915-8_18.
- Holger Dell, Martin Grohe, Gaurav Rattan.
  “Lovász Meets Weisfeiler and Leman”.
  2018. DOI: 10.4230/LIPIcs.ICALP.2018.40.
- Zdeněk Dvořák.
  “On recognizing graphs by numbers of homomorphisms”.
  2010. DOI: 10.1002/jgt.20461.
- Mihalis Yannakakis.
  “Algorithms for Acyclic Database Schemes”.
  1981.

Color Refinement counts homomorphisms

Why does this work?

Theorem (Dvořák 2010; Dell, Grohe, Rattan 2018)

Two vertices $u, v \in V(G)$ have different colors if, and only if, there is a rooted tree (T, r) such that

$$\begin{aligned} & |\{h \in \mathsf{hom}(T, G) \mid h(r) = u\}| \\ \neq\ & |\{h \in \mathsf{hom}(T, G) \mid h(r) = v\}| \end{aligned}$$