

Constant-delay Enumeration with self-joins & projections

Clément Rouvroy

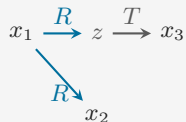
DaST UZH

Nofar Carmeli, David Carral, Luc Segoufin



Conjunctive queries and enumeration

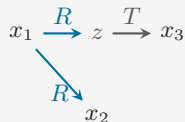
Conjunctive query (CQ)



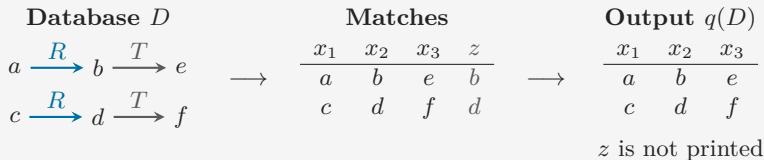
$$q(\underbrace{x_1, x_2, x_3}_{\text{Free variables}}) \leftarrow \underbrace{\exists z}_{\text{Projected}} . \overset{\text{self-join}}{R(x_1, x_2) \wedge R(x_1, z) \wedge T(z, x_3)}$$

Conjunctive queries and enumeration

Conjunctive query (CQ)

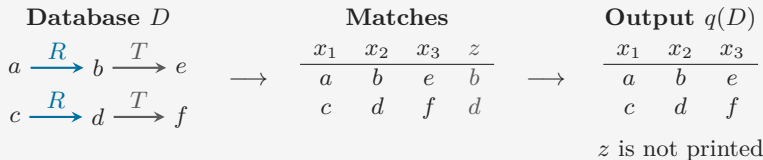
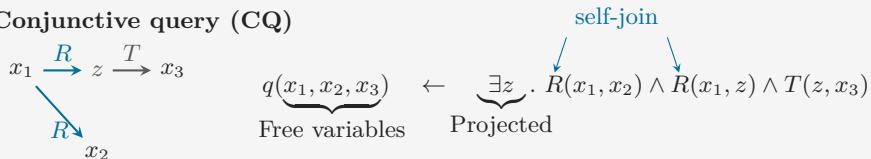


$$q(\underbrace{x_1, x_2, x_3}_{\text{Free variables}}) \leftarrow \underbrace{\exists z}_{\text{Projected}} \cdot \overset{\text{self-join}}{R(x_1, x_2) \wedge R(x_1, z) \wedge T(z, x_3)}$$



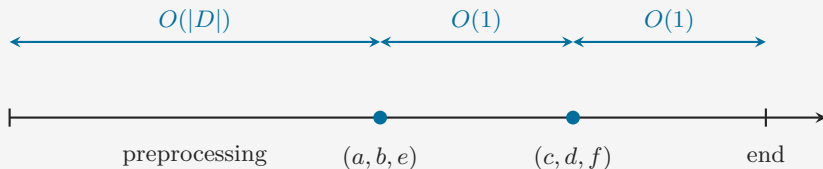
Conjunctive queries and enumeration

Conjunctive query (CQ)



CD $\circ$ Lin

Linear preprocessing, then constant-delay enumeration without duplicates:



Without self-joins

Theorem (2007, 2013) [1, 2]

For a CQ q without self-joins, assuming sBMM and sHyperclique,

$$q \in \text{CD} \circ \text{Lin} \iff q \text{ is acyclic and free-connex}$$

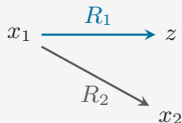
Without self-joins

Theorem (2007, 2013) [1, 2]

For a CQ q without self-joins, assuming sBMM and sHyperclique,

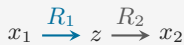
$$q \in \text{CD} \circ \text{Lin} \iff q \text{ is acyclic and free-connex}$$

Free-connex



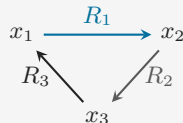
$\in \text{CD} \circ \text{Lin}$

A free-path



$\notin \text{CD} \circ \text{Lin}$

A cycle



$\notin \text{CD} \circ \text{Lin}$

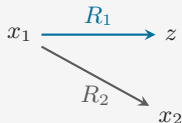
Without self-joins

Theorem (2007, 2013) [1, 2]

For a CQ q without self-joins, assuming sBMM and sHyperclique,

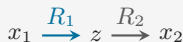
$$q \in \text{CD} \circ \text{Lin} \iff q \text{ is acyclic and free-connex}$$

Free-connex



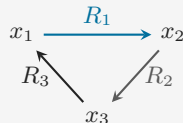
$\in \text{CD} \circ \text{Lin}$

A free-path



$\notin \text{CD} \circ \text{Lin}$

A cycle



$\notin \text{CD} \circ \text{Lin}$

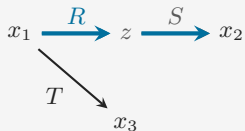
With [self-joins](#), some cyclic CQs belong to $\text{CD} \circ \text{Lin}$
(2020) [3]

**Can a CQ with a free-path
be in $CD \circ Lin$?**

An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge T(x_1, x_3)$$

The query q

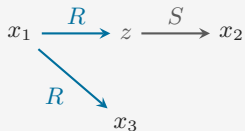


Free-path $x_1 \rightarrow z \rightarrow x_2 \implies q \notin \text{CD} \circ \text{Lin}$ under sBMM [1, 2]

An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge R(x_1, x_3)$$

The query q



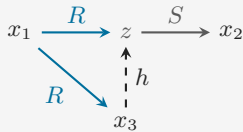
$$T \longrightarrow R$$

Now there is a self-join

An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge R(x_1, x_3)$$

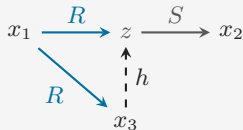
The query q



An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge R(x_1, x_3)$$

The query q



The full query q'

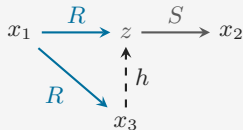


$$q'(x_1, x_2, x_3) \leftarrow R(x_1, x_3) \wedge S(x_3, x_2)$$

An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge R(x_1, x_3)$$

The query q



The full query q'



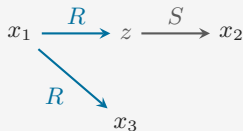
$$q' \in \text{CD} \circ \text{Lin}$$

$$q'(x_1, x_2, x_3) \leftarrow R(x_1, x_3) \wedge S(x_3, x_2)$$

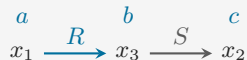
An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge R(x_1, x_3)$$

The query q



The full query q'



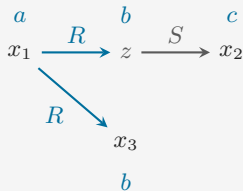
$$q' \in \text{CD} \circ \text{Lin}$$
$$(a, c, b) \in q'(D)$$

Database facts: $R(a, b)$ and $S(b, c)$

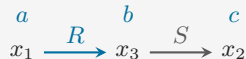
An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge R(x_1, x_3)$$

The query q



The full query q'



Witness $z = b$
 $(a, c, b) \in q(D)$

← injective

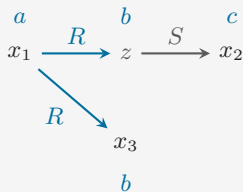
$q' \in \text{CD} \circ \text{Lin}$
 $(a, c, b) \in q'(D)$

Database facts: $R(a, b)$ and $S(b, c)$

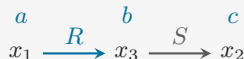
An easy free-path

$$q(x_1, x_2, x_3) \leftarrow \exists z. R(x_1, z) \wedge S(z, x_2) \wedge R(x_1, x_3)$$

The query q



The full query q'



Witness $z = b$
 $(a, c, b) \in q(D)$

← injective

$q' \in \text{CD} \circ \text{Lin}$
 $(a, c, b) \in q'(D)$

Database facts: $R(a, b)$ and $S(b, c)$

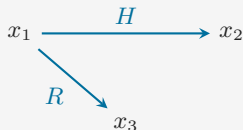
Cheater's lemma [4]

Any algorithm with $O(1)$ linear-time processing phases, $O(1)$ delay otherwise, and $O(1)$ repetitions per solution can be transformed into a $\text{CD} \circ \text{Lin}$ algorithm.

An easy free-path

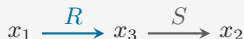
$$q_H(x_1, x_2, x_3) \leftarrow H(x_1, x_2) \wedge R(x_1, x_3)$$

The query with H



$$q_H \in \text{CD} \circ \text{Lin}$$

The full query q'



$$q' \in \text{CD} \circ \text{Lin}$$

We **enumerate** q' to **materialize** $H(x_1, x_2)$ such that

$$H(x_1, x_2) \iff \exists z. R(x_1, z) \wedge S(z, x_2)$$

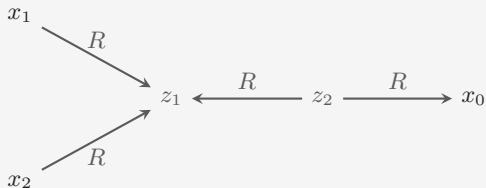
Cheater's lemma [4]

Any algorithm with $O(1)$ linear-time processing phases, $O(1)$ delay otherwise, and $O(1)$ repetitions per solution can be transformed into a $\text{CD} \circ \text{Lin}$ algorithm.

**Are free-paths the right granularity
with self-joins?**

Is handling every free-path enough?

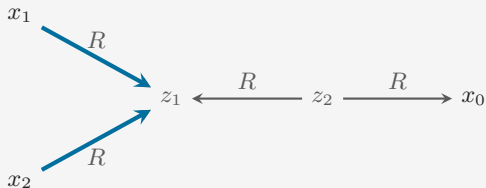
$$Q_2(x_0, x_1, x_2)$$



Three free-paths, all atoms use R

Is handling every free-path enough?

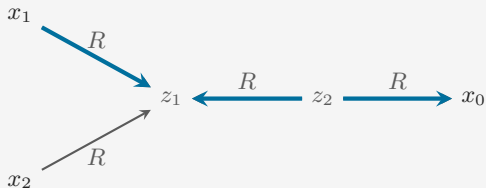
$$Q_2(x_0, x_1, x_2)$$



$H_1(x_1, x_2)$: the path (x_1, z_1, x_2)

Is handling every free-path enough?

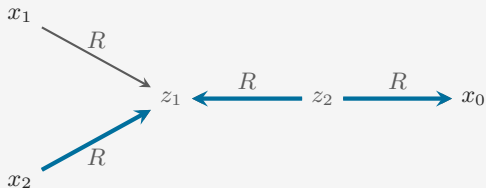
$$Q_2(x_0, x_1, x_2)$$



$H_2(x_1, x_0)$: the path (x_1, z_1, z_2, x_0)

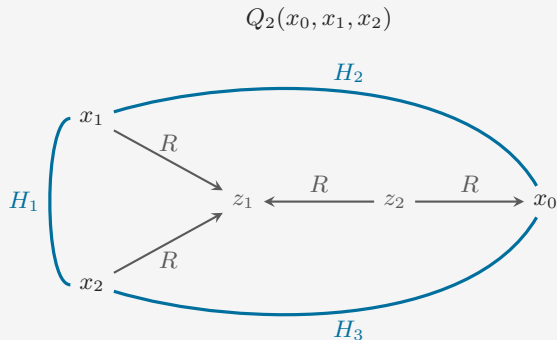
Is handling every free-path enough?

$$Q_2(x_0, x_1, x_2)$$



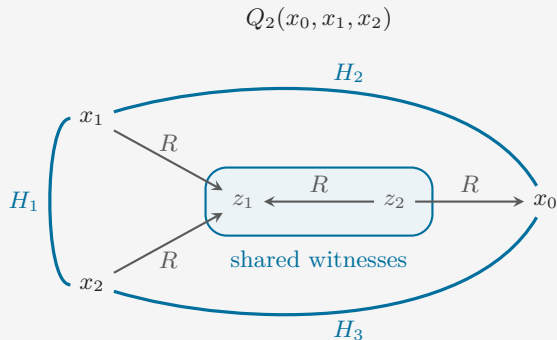
$H_3(x_2, x_0)$: the path (x_2, z_1, z_2, x_0)

Is handling every free-path enough?



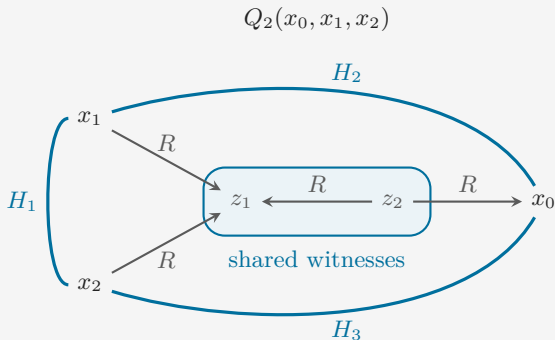
Claim. We can build H_1, H_2, H_3 for free while enumerating Q_2 .
They contain all free-path endpoint pairs.

Is handling every free-path enough?



Let $(x_1, x_0) \in H_2$ and $(x_2, x_0) \in H_3$,
can we test in $O(1)$ **whether they have common witnesses?**

Is handling every free-path enough?

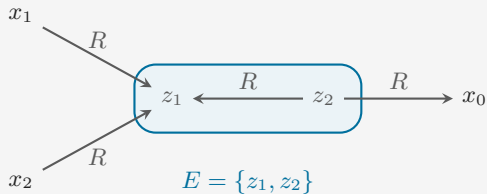


Let $(x_1, x_0) \in H_2$ and $(x_2, x_0) \in H_3$,
can we test in $O(1)$ **whether they have common witnesses?**

If you can do this, you can find a 4-clique in $O(n^3)$,
contradicting the 4-clique hypothesis [5].

Multi-frontier components

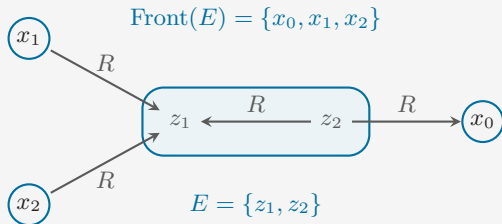
$$Q_2(x_0, x_1, x_2)$$



A **maximal** connected set of quantified variables

Multi-frontier components

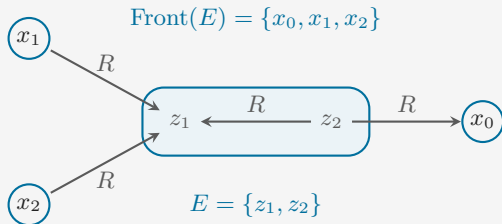
$$Q_2(x_0, x_1, x_2)$$



Its **frontier** consists of the **adjacent free variables**

Multi-frontier components

$$Q_2(x_0, x_1, x_2)$$

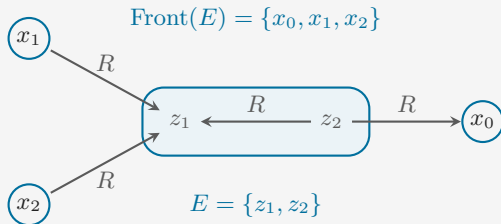


Definition

A **multi-frontier component** is a maximal connected set of quantified variables with at least two adjacent free variables.

Multi-frontier components

$$Q_2(x_0, x_1, x_2)$$



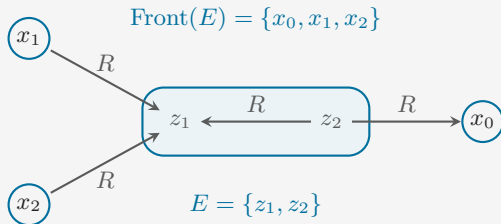
Definition

A **multi-frontier component** is a maximal connected set of quantified variables with at least two adjacent free variables.

- An isolated free-path forms a multi-frontier component.

Multi-frontier components

$$Q_2(x_0, x_1, x_2)$$



Definition

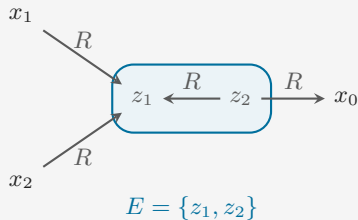
A **multi-frontier component** is a maximal connected set of quantified variables with at least two adjacent free variables.

- An isolated free-path forms a multi-frontier component.
- Free-paths that **share quantified variables** belong to the **same multi-frontier component**.

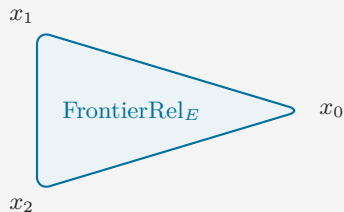
Multi-frontier components

$$Q_2(x_0, x_1, x_2)$$

Original query



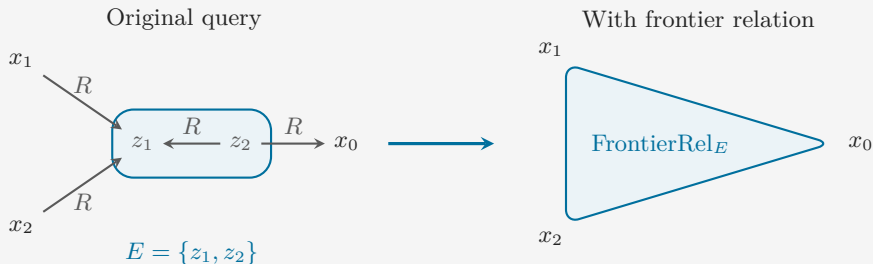
With frontier relation



The relation we would need: $\text{FrontierRel}_E(D) = q_{|\text{Front}(E)}(D)$

Multi-frontier components

$$Q_2(x_0, x_1, x_2)$$



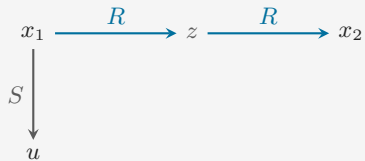
If E is **materializable inside q** , replace it by its frontier relation

Sufficient condition

Let q be an acyclic CQ with binary atoms. If **every multi-frontier component** is **materializable** inside q , then $q \in \text{CD} \circ \text{Lin}$.

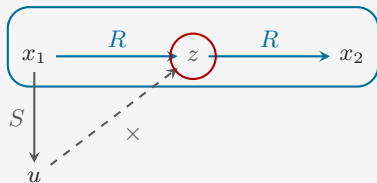
When self-joins do not help

What makes a free-path hard?



What makes a free-path hard?

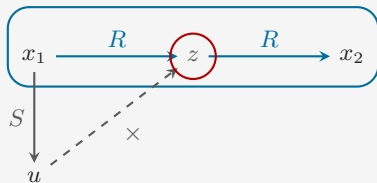
A path left alone



No outside output can fold
onto this free-path.

What makes a free-path hard?

A path left alone

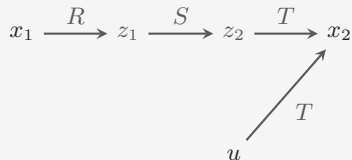
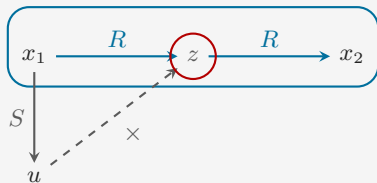


No outside output can fold
onto this free-path.

$q \notin \text{CD} \circ \text{Lin}$ under BMM

What makes a free-path hard?

A path left alone

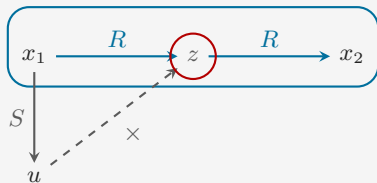


No outside output can fold
onto this free-path.

$q \notin \text{CD} \circ \text{Lin}$ under BMM

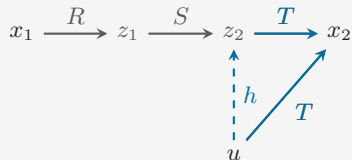
What makes a free-path hard?

A path left alone



No outside output can fold onto this free-path.

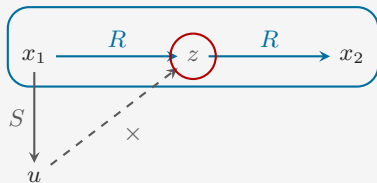
$q \notin \text{CD} \circ \text{Lin}$ under BMM



u can fold onto z_2 .

What makes a free-path hard?

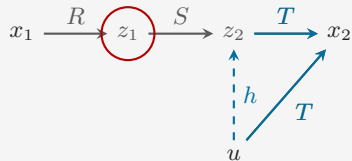
A path left alone



No outside output can fold onto this free-path.

$q \notin \text{CD} \circ \text{Lin}$ under BMM

One witness left hidden

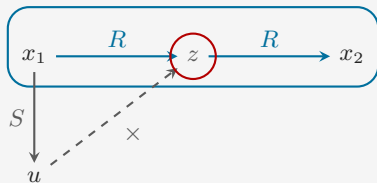


u can fold onto z_2 .

No output can fold onto z_1 .

What makes a free-path hard?

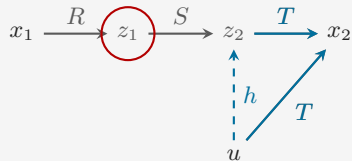
A path left alone



No outside output can fold onto this free-path.

$q \notin \text{CD} \circ \text{Lin}$ under BMM

One witness left hidden



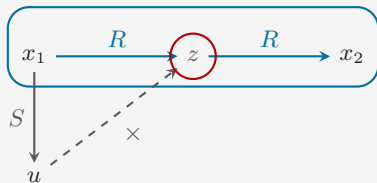
u can fold onto z_2 .

No output can fold onto z_1 .

$q \notin \text{CD} \circ \text{Lin}$ under VUTD [6]

What makes a free-path hard?

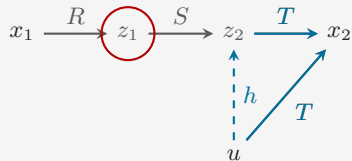
A path left alone



No outside output can fold onto this free-path.

$q \notin \text{CD} \circ \text{Lin}$ under BMM

One witness left hidden



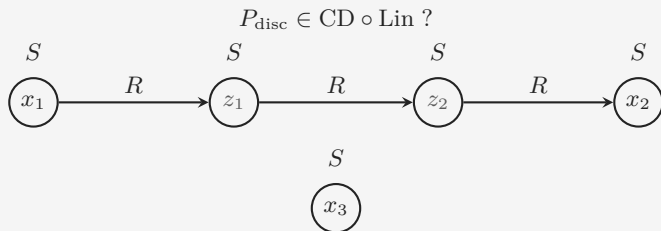
u can fold onto z_2 .

No output can fold onto z_1 .

$q \notin \text{CD} \circ \text{Lin}$ under VUTD [6]

Do we have a dichotomy?

An open case



An open case

$$P \in \text{DomLin} \circ \text{Lin} ?$$

$$x_1 \xrightarrow{R} z_1 \xrightarrow{R} z_2 \xrightarrow{R} x_2$$

Remove x_3 : the equivalent question allows $O(n)$ delay.

Linear preprocessing, $n = |\text{dom}(D)|$, no duplicates.

An open case

$$P \in \text{DomLin} \circ \text{Lin} ?$$

$$P \quad x_1 \xrightarrow{R} z_1 \xrightarrow{R} z_2 \xrightarrow{R} x_2$$

$$\textbf{Lemma.} \quad P \in \text{DomLin} \circ \text{Lin} \iff P_{\text{sf}} \in \text{DomLin} \circ \text{Lin}$$

$$P_{\text{sf}} \quad x_1 \xrightarrow{R_1} z_1 \xrightarrow{R_2} z_2 \xrightarrow{R_3} x_2$$

Linear preprocessing, $n = |\text{dom}(D)|$, no duplicates

An open case

$$P \in \text{DomLin} \circ \text{Lin} ?$$

$$P \quad x_1 \xrightarrow{R} z_1 \xrightarrow{R} z_2 \xrightarrow{R} x_2$$

$$\textbf{Lemma. } P \in \text{DomLin} \circ \text{Lin} \iff P_{\text{sf}} \in \text{DomLin} \circ \text{Lin}$$

$$P_{\text{sf}} \quad x_1 \xrightarrow{R_1} z_1 \xrightarrow{R_2} z_2 \xrightarrow{R_3} x_2$$

Linear preprocessing, $n = |\text{dom}(D)|$, no duplicates

Open Question

Under BMM, one hidden witness already rules out $O(1)$ delay.
Can we rule out $O(n)$ delay with two hidden witnesses?

Bibliography I

- [1] Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. “On Acyclic Conjunctive Queries and Constant Delay Enumeration”. In: *Computer Science Logic*. Springer, 2007, pp. 208–222. DOI: [10.1007/978-3-540-74915-8_18](https://doi.org/10.1007/978-3-540-74915-8_18).
- [2] Johann Brault-Baron. “De la pertinence de l’énumération : complexité en logiques propositionnelle et du premier ordre”. PhD thesis. Université de Caen, 2013. URL: <https://hal.science/tel-01081392>.
- [3] Christoph Berkholz, Fabian Gerhardt, and Nicole Schweikardt. “Constant Delay Enumeration for Conjunctive Queries: A Tutorial”. In: *ACM SIGLOG News* 7.1 (2020), pp. 4–33. DOI: [10.1145/3385634.3385636](https://doi.org/10.1145/3385634.3385636).
- [4] Nofar Carmeli and Luc Segoufin. “Conjunctive Queries With Self-Joins, Towards a Fine-Grained Enumeration Complexity Analysis”. In: *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS ’23. Association for Computing Machinery, 2023, pp. 277–289. DOI: [10.1145/3584372.3588667](https://doi.org/10.1145/3584372.3588667).
- [5] Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. “New Algorithms and Hardness for Incremental Single-Source Shortest Paths in Directed Graphs”. In: *STOC*. 2020, pp. 153–166. DOI: [10.1145/3357713.3384236](https://doi.org/10.1145/3357713.3384236).

Bibliography II

- [6] Karl Bringmann and Nofar Carmeli. “Unbalanced Triangle Detection and Enumeration Hardness for Unions of Conjunctive Queries”. In: *Logical Methods in Computer Science* 21.1, 29 (2025), pp. 1–33. DOI: [10.46298/lmcs-21\(1:29\)2025](https://doi.org/10.46298/lmcs-21(1:29)2025). URL: [https://doi.org/10.46298/lmcs-21\(1:29\)2025](https://doi.org/10.46298/lmcs-21(1:29)2025).