Enumeration

Nicole Schweikardt

Humboldt-Universität zu Berlin

EDBT-Intended Summer School 2022: Data and Knowledge

Bordeaux, July 5, 2022

Databases and Queries

- σ is a finite relational signature
- a database (db) D is a finite relational σ-structure
- adom(D) is this structure's universe ("active domain")

Databases and Queries

- σ is a finite relational signature
- a database (db) D is a finite relational σ-structure
- adom(D) is this structure's universe ("active domain")
- fix a countably infinite set **dom** ("domain") and assume w.l.o.g. that adom(D) $\subseteq$ **dom**

Databases and Queries

- σ is a finite relational signature
- a database (db) D is a finite relational σ-structure
- adom(D) is this structure's universe ("active domain")
- fix a countably infinite set **dom** ("domain") and assume w.l.o.g. that adom(D) $\subseteq$ **dom**

- FO is first-order logic of signature σ
- CQ ("conjunctive queries") are FO-formulas of the form

$$\exists y_1 \cdots \exists y_\ell \, \big(\, R_1(\ldots) \wedge \cdots \wedge R_s(\ldots) \, \big)$$

Databases and Queries

- σ is a finite relational signature
- a database (db) D is a finite relational σ-structure
- adom(D) is this structure's universe ("active domain")
- fix a countably infinite set **dom** ("domain") and assume w.l.o.g. that adom(D) $\subseteq$ **dom**

- FO is first-order logic of signature σ
- CQ ("conjunctive queries") are FO-formulas of the form

$$\exists y_1 \cdots \exists y_\ell \, \big(\, R_1(...) \wedge \cdots \wedge R_s(...) \, \big)$$

- Result of query φ on db D:

$$\varphi(D) \quad := \quad \{ \, \alpha : \mathsf{free}(\varphi) \to \mathsf{adom}(D) \; : \; (D, \alpha) \models \varphi \, \}$$

Databases and Queries

- σ is a finite relational signature
- a database (db) D is a finite relational σ-structure
- $\mathrm{adom}(D)$ is this structure's universe ("active domain")
- fix a countably infinite set **dom** ("domain") and assume w.l.o.g. that $\mathrm{adom}(D) \subseteq \mathbf{dom}$

- FO is first-order logic of signature σ
- CQ ("conjunctive queries") are FO-formulas of the form

$$\exists y_1 \cdots \exists y_\ell \, \big(\, R_1(\ldots) \wedge \cdots \wedge R_s(\ldots) \, \big)$$

- Result of query φ on db D:

$$\begin{aligned} \varphi(D) \quad &:= \quad \{ \, \alpha : \mathrm{free}(\varphi) \to \mathrm{adom}(D) \; : \; (D, \alpha) \models \varphi \, \} \\ &= \quad \{ \, \bar{a} \in \mathrm{adom}(D)^k \; : \; D \models \varphi(\bar{a}) \, \} \end{aligned}$$

where $\{x_1, \ldots, x_k\} = \mathrm{free}(\varphi)$

Query evaluation

Consider a query language L (e.g., FO, CQ)

Let $\varphi(x_1, \ldots, x_k)$ be a query formulated in L.
Let D be a db.

Task:

Evaluate φ on D, i.e., compute the query result

$$\begin{aligned} \varphi(D) &= \{ \alpha : \mathsf{free}(\varphi) \to \mathsf{adom}(D) \ : \ (D, \alpha) \models \varphi \} \\ &= \{ \bar{a} \in \mathsf{adom}(D)^k \ : \ D \models \varphi(\bar{a}) \} \end{aligned}$$

Query evaluation

Consider a query language L (e.g., FO, CQ)

Let $\varphi(x_1, \ldots, x_k)$ be a query formulated in L.
Let D be a db.

Task:

Evaluate φ on D, i.e., compute the query result

$$\begin{aligned} \varphi(D) &= \{ \alpha : \mathsf{free}(\varphi) \to \mathsf{adom}(D) \ : \ (D, \alpha) \models \varphi \} \\ &= \{ \bar{a} \in \mathsf{adom}(D)^k \ : \ D \models \varphi(\bar{a}) \} \end{aligned}$$

Special case $\mathsf{free}(\varphi) = \emptyset$ (i.e., $k = 0$): Boolean queries:
Evaluate φ on D means Decide if $D \models \varphi$

Complexity of query evaluation

In his STOC'82 paper, Moshe Vardi introduced the notions

combined complexity

and data complexity

Complexity of query evaluation

In his STOC'82 paper, Moshe Vardi introduced the notions

- combined complexity: Measure the complexity of evaluating φ on D in terms of the sizes of φ and D.

and data complexity

Complexity of query evaluation

In his STOC'82 paper, Moshe Vardi introduced the notions

- combined complexity: Measure the complexity of evaluating φ on D in terms of the sizes of φ and D.

and

- data complexity: Assume the query φ to be fixed. Measure the complexity of evaluating φ on D only in terms of the size of D.

Complexity of query evaluation

In his STOC'82 paper, Moshe Vardi introduced the notions

combined complexity: Measure the complexity of evaluating φ on D in terms of the sizes of φ and D.

and data complexity: Assume the query φ to be fixed. Measure the complexity of evaluating φ on D only in terms of the size of D.

Typical results of Finite Model Theory and Database Theory:

- Boolean Conjunctive Queries: data complexity is in AC^0, combined complexity is NP-complete [Chandra & Merlin '77]

Complexity of query evaluation

In his STOC'82 paper, Moshe Vardi introduced the notions

combined complexity: Measure the complexity of evaluating φ on D in terms of the sizes of φ and D.

and data complexity: Assume the query φ to be fixed. Measure the complexity of evaluating φ on D only in terms of the size of D.

Typical results of Finite Model Theory and Database Theory:

- Boolean Conjunctive Queries: data complexity is in AC^0, combined complexity is NP-complete [Chandra & Merlin '77]
- Boolean First-Order Queries: data complexity is in AC^0, combined complexity is PSPACE-complete [Stockmeyer '74, Vardi '82]

Complexity of query evaluation

In his STOC'82 paper, Moshe Vardi introduced the notions

combined complexity: Measure the complexity of evaluating φ on D in terms of the sizes of φ and D.

and data complexity: Assume the query φ to be fixed. Measure the complexity of evaluating φ on D only in terms of the size of D.

Typical results of Finite Model Theory and Database Theory:

- Boolean Conjunctive Queries: data complexity is in AC^0, combined complexity is NP-complete [Chandra & Merlin '77]
- Boolean First-Order Queries: data complexity is in AC^0, combined complexity is PSPACE-complete [Stockmeyer '74, Vardi '82]
- Boolean Queries of Least-Fixed Point Logic LFP: data complexity is PTIME-complete, combined complexity is EXPTIME-complete [Immerman '82, Vardi '82].

Complexity of query evaluation

In his STOC'82 paper, Moshe Vardi introduced the notions

combined complexity: Measure the complexity of evaluating φ on D in terms of the sizes of φ and D.

and data complexity: Assume the query φ to be fixed. Measure the complexity of evaluating φ on D only in terms of the size of D.

Typical results of Finite Model Theory and Database Theory:

- Boolean Conjunctive Queries: data complexity is in AC^0, combined complexity is NP-complete [Chandra & Merlin '77]
- Boolean First-Order Queries: data complexity is in AC^0, combined complexity is PSPACE-complete [Stockmeyer '74, Vardi '82]
- Boolean Queries of Least-Fixed Point Logic LFP: data complexity is PTIME-complete, combined complexity is EXPTIME-complete [Immerman '82, Vardi '82].

CAVEAT: These notions & results do not handle updates of the db!

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

  For k-ary queries:

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

  For k-ary queries:
  - Compute the number of tuples in $\varphi(D)$

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

  For k-ary queries:
  - Compute the number of tuples in $\varphi(D)$
  - Test for a given tuple $\bar{a}$ whether $\bar{a} \in \varphi(D)$

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

  For k-ary queries:
  - Compute the number of tuples in $\varphi(D)$
  - Test for a given tuple $\bar{a}$ whether $\bar{a} \in \varphi(D)$
  - Enumerate the tuples in $\varphi(D)$

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

  For k-ary queries:
  - Compute the number of tuples in $\varphi(D)$
  - Test for a given tuple $\bar{a}$ whether $\bar{a} \in \varphi(D)$
  - Enumerate the tuples in $\varphi(D)$
    - without repetition
    - with guarantee on the max. delay between output tuples

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

  For k-ary queries:
  - Compute the number of tuples in $\varphi(D)$
  - Test for a given tuple $\bar{a}$ whether $\bar{a} \in \varphi(D)$
  - Enumerate the tuples in $\varphi(D)$
    - without repetition
    - with guarantee on the max. delay between output tuples
- Dynamic setting:
  Tuples may be inserted into or deleted from D

A typical scenario for DB-systems

- Input:
  - Database D
  - query $\varphi(x_1, \ldots, x_k)$
- Preprocessing:
  Build a suitable data structure that represents D and $\varphi(D)$
- Output:
  For Boolean queries:
  - Decide if $D \models \varphi$

  For k-ary queries:
  - Compute the number of tuples in $\varphi(D)$
  - Test for a given tuple $\bar{a}$ whether $\bar{a} \in \varphi(D)$
  - Enumerate the tuples in $\varphi(D)$
    - without repetition
    - with guarantee on the max. delay between output tuples
- Dynamic setting:
  Tuples may be inserted into or deleted from D

Goal: preprocessing time & delay (& update time) as small as possible

Constant delay enumeration after linear time preprocessing

- Let φ be a fixed query
- Let C be a class of databases
- Goal:
  Find an algorithm which, upon input of a db D,
  uses preprocessing time (pseudo-)linear in $\|D\|$ and then
  enumerates $\varphi(D)$ with delay $O(1)$

linear : $O(N)$

pseudo-linear : $\forall\ \epsilon > 0\ \ \exists$ algorithm with runtime $O(N^{1+\epsilon})$

Constant delay enumeration after linear time preprocessing

- Let φ be a fixed query
- Let C be a class of databases
- Goal:
  Find an algorithm which, upon input of a db D,
  uses preprocessing time (pseudo-)linear in $\|D\|$ and then
  enumerates $\varphi(D)$ with delay $O(1)$

- Question:
  For which φ, C is this possible?

linear : $O(N)$

pseudo-linear : $\forall\ \epsilon > 0\ \ \exists$ algorithm with runtime $O(N^{1+\epsilon})$

Overview

Overview

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

For any subgraph-closed class C that is not nowhere dense, evaluation of Boolean FO-queries is not possible in time $f(\|\varphi\|)\|D\|^{O(1)}$
(assuming FPT $\neq$ W[1]) [Kreutzer 2011, Dvořák, Král, Thomas 2010]

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

  For any subgraph-closed class C that is not nowhere dense, evaluation of Boolean FO-queries is not possible in time $f(\|\varphi\|)\|D\|^{O(1)}$
  (assuming FPT $\neq$ W[1]) [Kreutzer 2011, Dvořák, Král, Thomas 2010]

- C has low degree [Durand, S., Segoufin 2014]

  i.e., for each $\delta > 0$, all sufficiently large D in C have degree $\leqslant \|D\|^{\delta}$.

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

  For any subgraph-closed class C that is not nowhere dense, evaluation of Boolean FO-queries is not possible in time $f(\|\varphi\|)\|D\|^{O(1)}$ (assuming FPT $\neq$ W[1]) [Kreutzer 2011, Dvořák, Král, Thomas 2010]

- C has low degree [Durand, S., Segoufin 2014]

  i.e., for each $\delta > 0$, all sufficiently large D in C have degree $\leqslant \|D\|^{\delta}$.

  Some such C are not nowhere dense (and not subgraph-closed).

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

  For any subgraph-closed class C that is not nowhere dense, evaluation of Boolean FO-queries is not possible in time $f(\|\varphi\|)\|D\|^{O(1)}$
  (assuming FPT $\neq$ W[1]) [Kreutzer 2011, Dvořák, Král, Thomas 2010]

- C has low degree [Durand, S., Segoufin 2014]

  i.e., for each $\delta > 0$, all sufficiently large D in C have degree $\leqslant \|D\|^{\delta}$.

  Some such C are not nowhere dense (and not subgraph-closed).

Bounded degree databases

Graph $G = (V, E)$:

$$\begin{array}{rcl} \text{degree of a node } v & : & \text{the number of neighbours of } v \text{ in } G \\ \text{degree of } G & : & \max\{\text{degree}(v) : v \in V\} \end{array}$$

Database D:

$$\text{degree of } D \quad : \quad \text{degree of the Gaifman graph of } D$$

Gaifman graph of D:

the graph $G = (V, E)$ with $V = \text{adom}(D)$ and an edge between two distinct nodes $a, b \in V$ iff some tuple in some relation of D contains a and b

FO query evaluation on dbs of degree $\leqslant d$

Boolean queries:

- evaluation in linear time [Seese 1996]

FO query evaluation on dbs of degree $\leqslant d$

Boolean queries:

- evaluation in linear time [Seese 1996]
- evaluation in time $f(\varphi, d)\|D\|$, for [Frick, Grohe 2004]

$$f(\varphi, d) \quad = \quad 2^{d^{2^{O(\|\varphi\|)}}} \quad = \quad 3\text{-exp}(\|\varphi\| + \lg\lg d)$$

and the 3-fold exponential blow-up is unavoidable assuming FPT $\neq$ AW[$*$].

FO query evaluation on dbs of degree $\leqslant d$

Boolean queries:

- evaluation in linear time [Seese 1996]
- evaluation in time $f(\varphi, d)\|D\|$, for [Frick, Grohe 2004]

$$f(\varphi, d) \quad = \quad 2^{d^{2^{O(\|\varphi\|)}}} \quad = \quad 3\text{-exp}(\|\varphi\| + \lg\lg d)$$

and the 3-fold exponential blow-up is unavoidable assuming FPT $\neq$ AW[$*$].

Non-Boolean queries:

- constant-delay enumeration with linear-time preprocessing [Durand, Grandjean 2007]

FO query evaluation on dbs of degree $\leqslant d$

Boolean queries:

- evaluation in linear time [Seese 1996]
- evaluation in time $f(\varphi, d)\|D\|$, for [Frick, Grohe 2004]

$$f(\varphi, d) \quad = \quad 2^{d^{2^{O(\|\varphi\|)}}} \quad = \quad \text{3-exp}(\|\varphi\| + \lg\lg d)$$

and the 3-fold exponential blow-up is unavoidable assuming FPT $\neq$ AW[$*$].

Non-Boolean queries:

- constant-delay enumeration with linear-time preprocessing [Durand, Grandjean 2007]
- enumeration with delay $f(\varphi, d)$ and preprocessing $f(\varphi, d)\|D\|$, where $f(\varphi, d) = \text{3-exp}(\|\varphi\| + \lg\lg d)$ [Kazana, Segoufin 2011]

FO query evaluation on dbs of degree $\leqslant d$

Boolean queries:

- evaluation in linear time [Seese 1996]
- evaluation in time $f(\varphi, d)\|D\|$, for [Frick, Grohe 2004]

$$f(\varphi, d) \quad = \quad 2^{d^{2^{O(\|\varphi\|)}}} \quad = \quad \text{3-exp}(\|\varphi\| + \lg\lg d)$$

and the 3-fold exponential blow-up is unavoidable assuming FPT $\neq$ AW[$*$].

Non-Boolean queries:

- constant-delay enumeration with linear-time preprocessing [Durand, Grandjean 2007]
- enumeration with delay $f(\varphi, d)$ and preprocessing $f(\varphi, d)\|D\|$, where $f(\varphi, d) = \text{3-exp}(\|\varphi\| + \lg\lg d)$ [Kazana, Segoufin 2011]

Generalisation to the dynamic setting (update time $f(\varphi, d)$) and FO+MOD [Berkholz, Keppeler, S. 2017] and FOC($\mathbb{P}$) [Kuske, S. 2017]

FO+MOD queries and FOC($\mathbb{P}$) queries

Movie	
Name	Actor
Alien	Sigourney Weaver
Blade Runner	Harrison Ford
Blade Runner	Sean Young
Brazil	Jonathan Pryce
Brazil	Kim Greist
Casablanca	Humphrey Bogart
Casablanca	Ingrid Bergmann
Gravity	Sandra Bullock
Gravity	George Clooney
Resident Evil	Milla Jovovich
Terminator	Arnold Schwarzenegger
Terminator	Linda Hamilton
Terminator	Michael Biehn
⋮	⋮

Is the number of movies with Ingrid Bergmann even?

In FO+MOD:

$\exists^{0 \bmod 2} y \;\; \mathit{Movie}(y, \text{"Ingrid Bergmann"})$

FO+MOD = extension of first-order logic with modulo-counting quantifiers $\exists^{i \bmod m} y \; \psi(y, \bar{z})$

FO+MOD queries and FOC(ℙ) queries

Movie	
Name	Actor
Alien	Sigourney Weaver
Blade Runner	Harrison Ford
Blade Runner	Sean Young
Brazil	Jonathan Pryce
Brazil	Kim Greist
Casablanca	Humphrey Bogart
Casablanca	Ingrid Bergmann
Gravity	Sandra Bullock
Gravity	George Clooney
Resident Evil	Milla Jovovich
Terminator	Arnold Schwarzenegger
Terminator	Linda Hamilton
Terminator	Michael Biehn
⋮	⋮

Is the number of movies with Ingrid Bergmann even?

In FO+MOD:

$$\exists^{0 \bmod 2} y \quad \mathit{Movie}(y, \text{"Ingrid Bergmann"})$$

In FOC(ℙ):

$$P_{even}\big(\#(y).\mathit{Movie}(y, \text{"Ingrid Bergmann"})\big)$$

FO+MOD = extension of first-order logic with modulo-counting quantifiers $\exists^{i \bmod m} y\ \psi(y, \bar{z})$

FO+MOD queries and FOC($\mathbb{P}$) queries

Movie	
Name	Actor
Alien	Sigourney Weaver
Blade Runner	Harrison Ford
Blade Runner	Sean Young
Brazil	Jonathan Pryce
Brazil	Kim Greist
Casablanca	Humphrey Bogart
Casablanca	Ingrid Bergmann
Gravity	Sandra Bullock
Gravity	George Clooney
Resident Evil	Milla Jovovich
Terminator	Arnold Schwarzenegger
Terminator	Linda Hamilton
Terminator	Michael Biehn
⋮	⋮

Is the number of movies with Ingrid Bergmann even?

In FO+MOD:

$$\exists^{0 \bmod 2} y \;\; \mathit{Movie}(y, \text{"Ingrid Bergmann"})$$

In FOC($\mathbb{P}$):

$$P_{even}\big(\#(y).\mathit{Movie}(y, \text{"Ingrid Bergmann"}) \big)$$

FO+MOD = extension of first-order logic with modulo-counting quantifiers $\exists^{i \bmod m} y \; \psi(y, \bar{z})$

Let $\mathbb{P}$ be a collection of numerical predicates. E.g., $\mathbb{P}$ may contain the predicates $[\![P_{even}]\!] = \{i \in \mathbb{Z} : i \text{ is even}\}$ and $[\![P_{\leqslant}]\!] = \{(i,j) \in \mathbb{Z}^2 : i \leqslant j\}$.

FO+MOD queries and FOC($\mathbb{P}$) queries

Movie	
Name	Actor
Alien	Sigourney Weaver
Blade Runner	Harrison Ford
Blade Runner	Sean Young
Brazil	Jonathan Pryce
Brazil	Kim Greist
Casablanca	Humphrey Bogart
Casablanca	Ingrid Bergmann
Gravity	Sandra Bullock
Gravity	George Clooney
Resident Evil	Milla Jovovich
Terminator	Arnold Schwarzenegger
Terminator	Linda Hamilton
Terminator	Michael Biehn
⋮	⋮

Is the number of movies with Ingrid Bergmann even?

In FO+MOD:

$\exists^{0 \bmod 2} y \;\; \mathit{Movie}(y, \text{"Ingrid Bergmann"})$

In FOC($\mathbb{P}$):

$P_{even}\big(\#(y).\mathit{Movie}(y, \text{"Ingrid Bergmann"}) \big)$

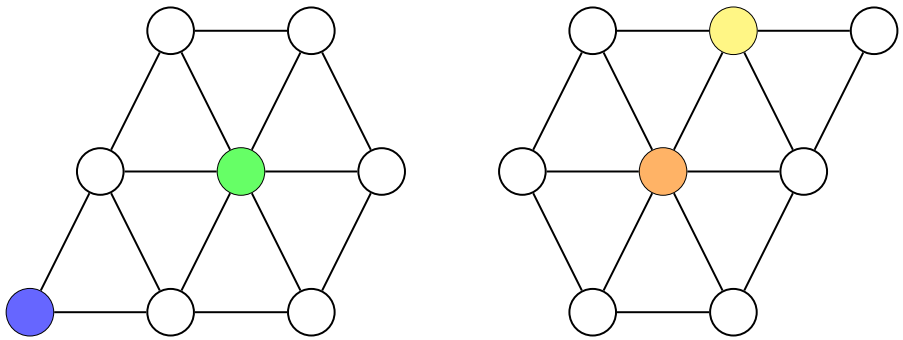
FO+MOD = extension of first-order logic with modulo-counting quantifiers $\exists^{i \bmod m} y \; \psi(y, \overline{z})$

Let $\mathbb{P}$ be a collection of numerical predicates. E.g., $\mathbb{P}$ may contain the predicates $[\![P_{even}]\!] = \{i \in \mathbb{Z} : i \text{ is even}\}$ and $[\![P_{\leqslant}]\!] = \{(i,j) \in \mathbb{Z}^2 : i \leqslant j\}$.

FOC($\mathbb{P}$) = extension of first-order logic with formulas of the form $P(t_1, \ldots, t_r)$ for $P \in \mathbb{P}$ of arity r, and where each t_i is a counting term built using integers, $+$, $\cdot$, and basic counting terms $t(\overline{x})$ of the form $\#\overline{y}.\psi(\overline{x}, \overline{y})$

Hanf normal form for FO+MOD

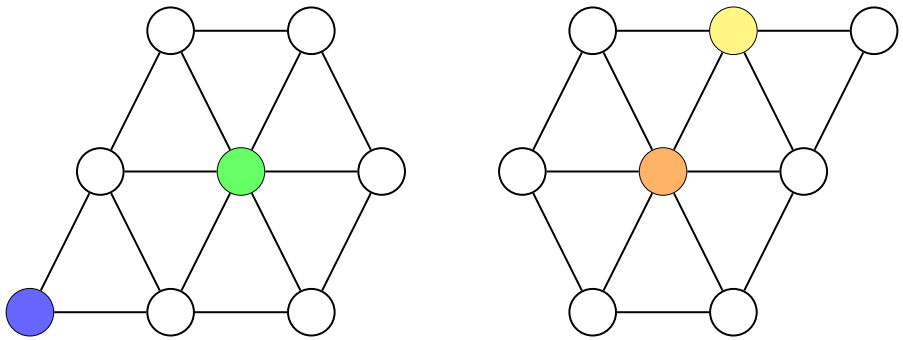
- A type τ with k centres and radius r:



Example type with $k = 4$ centres and radius $r = 1$

Hanf normal form for FO+MOD

- A type τ with k centres and radius r:



Example type with $k = 4$ centres and radius $r = 1$

- $\mathcal{N}_r^D(\bar{b})$ is the induced substructure of D on

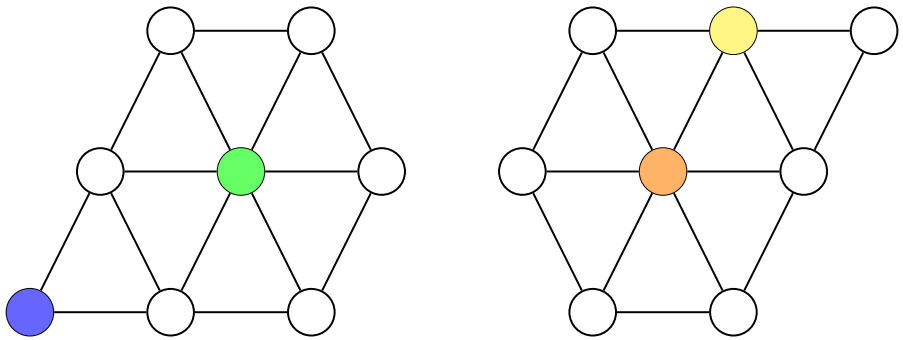
$$N_r^D(\bar{b}) \quad = \quad N_r^D(b_1) \cup \cdots \cup N_r^D(b_k)$$

where

$$N_r^D(b_i) \quad = \quad \{a \in \mathrm{adom}(D) \ : \ \mathrm{dist}^D(b_i, a) \leqslant r\}$$

Hanf normal form for FO+MOD

- A type τ with k centres and radius r:



Example type with $k = 4$ centres and radius $r = 1$

- $\mathcal{N}_r^D(\bar{b})$ is the induced substructure of D on

$$N_r^D(\bar{b}) \quad = \quad N_r^D(b_1) \cup \cdots \cup N_r^D(b_k)$$

where

$$N_r^D(b_i) \quad = \quad \{a \in \mathrm{adom}(D) \ : \ \mathrm{dist}^D(b_i, a) \leqslant r\}$$

- Sphere-formula $\mathrm{sph}_\tau(\bar{x})$:

$$(D, \bar{a}) \models \mathrm{sph}_\tau(\bar{x}) \quad \Longleftrightarrow \quad (\mathcal{N}_r^D(\bar{a}), \bar{a}) \cong \tau$$

Hanf normal form for FO+MOD

A Hanf normal form $\psi(\overline{x})$ is a Boolean combination of

- sphere-formulas $\mathrm{sph}_\rho(\overline{x})$ and
- Hanf-sentences $\exists^{\geqslant m} y\ \mathrm{sph}_\tau(y)$ and $\exists^{i \bmod m} y\ \mathrm{sph}_\tau(y)$ where τ is a type with 1 centre and radius r.

Hanf normal form for FO+MOD

A Hanf normal form $\psi(\overline{x})$ is a Boolean combination of

- sphere-formulas $\mathrm{sph}_\rho(\overline{x})$ and
- Hanf-sentences $\exists^{\geqslant m} y\ \mathrm{sph}_\tau(y)$ and $\exists^{i \bmod m} y\ \mathrm{sph}_\tau(y)$ where τ is a type with 1 centre and radius r.

Two queries $\varphi(\overline{x})$ and $\psi(\overline{x})$ are d-equivalent iff

$$(D, \overline{a}) \models \varphi \iff (D, \overline{a}) \models \psi$$

for all dbs D of degree $\leqslant d$.

Hanf normal form for FO+MOD

A Hanf normal form $\psi(\overline{x})$ is a Boolean combination of

- sphere-formulas $\mathrm{sph}_\rho(\overline{x})$ and
- Hanf-sentences $\exists^{\geqslant m} y\ \mathrm{sph}_\tau(y)$ and $\exists^{i \bmod m} y\ \mathrm{sph}_\tau(y)$ where τ is a type with 1 centre and radius r.

Two queries $\varphi(\overline{x})$ and $\psi(\overline{x})$ are d-equivalent iff

$$(D, \overline{a}) \models \varphi \iff (D, \overline{a}) \models \psi$$

for all dbs D of degree $\leqslant d$.

Theorem (Heimberg, Kuske, S., 2016)

There is an algorithm which receives as input a degree bound $d \geqslant 2$ and a FO+MOD*-formula $\varphi(\overline{x})$, and constructs a d-equivalent formula $\psi(\overline{x})$ in Hanf normal form.*
The algorithm's runtime is $f(\varphi, d) = 3\text{-}exp(\|\varphi\| + \lg\lg d)$.

Boolean queries under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Observation: There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a Boolean FO+MOD-query φ, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to return the query result $\varphi(D)$ with answer time $O(1)$.

Boolean queries under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Observation: There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a Boolean FO+MOD-query φ, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to return the query result $\varphi(D)$ with answer time $O(1)$.

Proof Idea:

1. Transform φ into Hanf normal form ψ.
2. For each type τ occurring in ψ, store the number A_τ of elements in adom(D) whose r-neighbourhood has type τ.
3. Inserting/deleting a tuple in D only affects the type of a fixed number of elements in adom(D) $\rightsquigarrow$ update affected A_τ-values.

Enumeration under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Theorem (Berkholz, Keppeler, S. 2017): There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a k-ary FO+MOD-query $\varphi(\overline{x})$, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to enumerate $\varphi(D)$ with delay $O(k^2)$.

Enumeration under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Theorem (Berkholz, Keppeler, S. 2017): There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a k-ary FO+MOD-query $\varphi(\overline{x})$, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to enumerate $\varphi(D)$ with delay $O(k^2)$.

Proof Idea:

Proof idea: Reduction to coloured graphs

Input:

Database D

FO+MOD-query $\varphi(x_1, ..., x_k)$

Same approach as in [Durand, S., Segoufin 2014],

but now we have to take care of updates!

Proof idea: Reduction to coloured graphs

Input:

Database D

FO+MOD-query $\varphi(x_1, ..., x_k)$

$\sigma_k := \{E, C_1, ..., C_k\}$

σ_k-structure $\mathcal{G}$

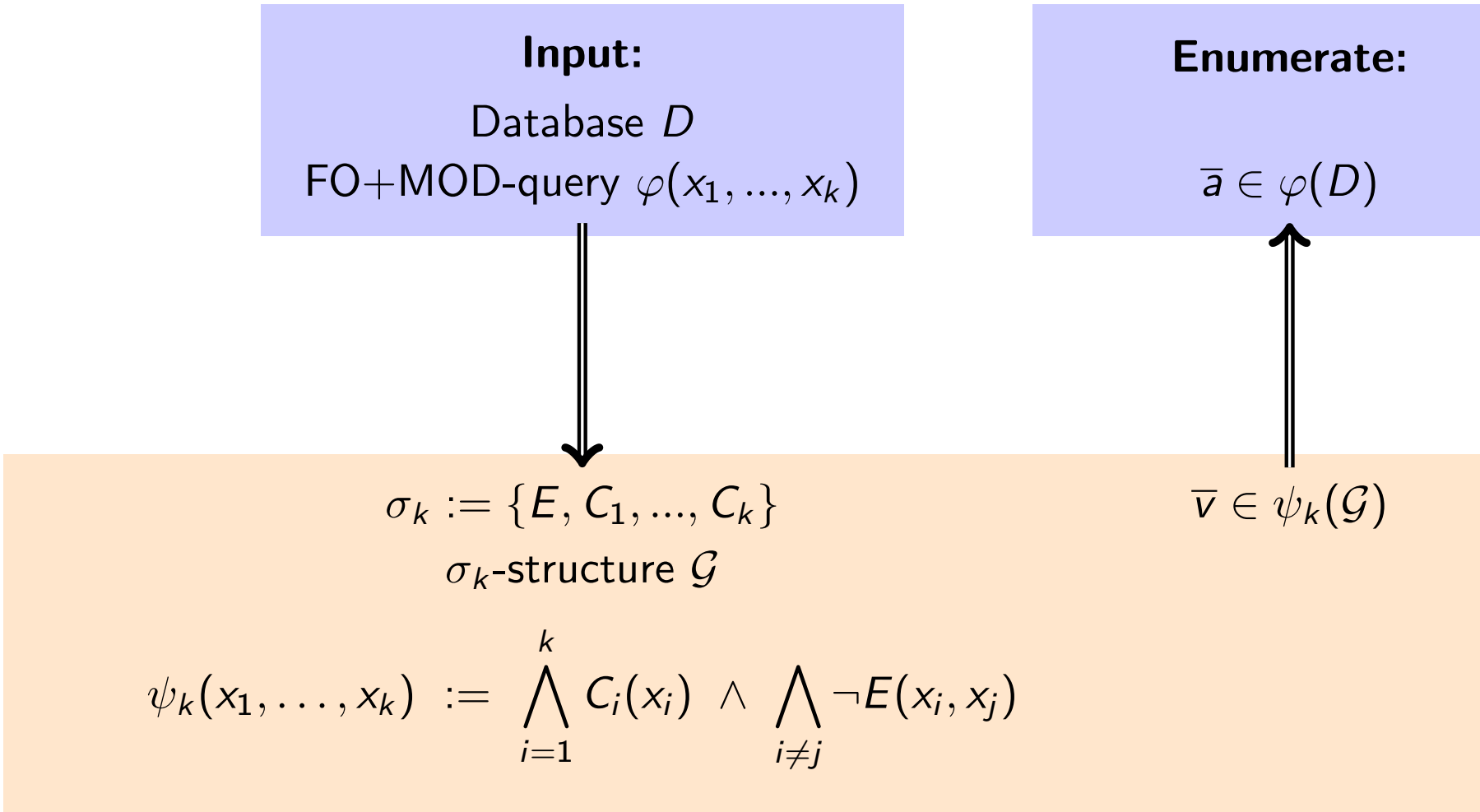
$\overline{v} \in \psi_k(\mathcal{G})$

$$\psi_k(x_1, \ldots, x_k) := \bigwedge_{i=1}^{k} C_i(x_i) \wedge \bigwedge_{i \neq j} \neg E(x_i, x_j)$$

Same approach as in [Durand, S., Segoufin 2014],

but now we have to take care of updates!

Proof idea: Reduction to coloured graphs



Same approach as in [Durand, S., Segoufin 2014],
but now we have to take care of updates!

Representing Databases by Coloured Graphs

$$\varphi(x_1, \ldots, x_k) \quad \equiv_d \quad \bigvee_{i \in \mathcal{I}} \mathrm{sph}_{\tau_i}(x_1, \ldots, x_k) \quad \& \text{ sentences}$$

Representing Databases by Coloured Graphs

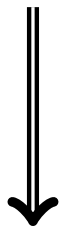
$$\varphi(x_1, \ldots, x_k) \quad \equiv_d \quad \bigvee_{i \in \mathcal{I}} \mathsf{sph}_{\tau_i}(x_1, \ldots, x_k) \quad \& \text{ sentences}$$

$$\mathsf{sph}_{\tau}(\overline{x}_1, \ldots, \overline{x}_c) \quad \equiv_d \quad \bigwedge_{j \in \{1,\ldots,c\}} \mathsf{sph}_{\tau_j}(\overline{x}_j) \quad \wedge \quad \bigwedge_{j \neq j'} \neg \mathsf{dist}_{\leqslant 2r+1}(\overline{x}_j, \overline{x}_{j'})$$

Representing Databases by Coloured Graphs

$$\varphi(x_1,\ldots,x_k) \quad \equiv_d \quad \bigvee_{i\in\mathcal{I}} \mathsf{sph}_{\tau_i}(x_1,\ldots,x_k) \quad \text{\& sentences}$$

$$\mathsf{sph}_{\tau}(\overline{x}_1,\ldots,\overline{x}_c) \quad \equiv_d \quad \bigwedge_{j\in\{1,\ldots,c\}} \mathsf{sph}_{\tau_j}(\overline{x}_j) \quad \wedge \quad \bigwedge_{j\neq j'} \neg\mathsf{dist}_{\leqslant 2r+1}(\overline{x}_j,\overline{x}_{j'})$$



$$\varphi_c(z_1,\ldots,z_c) \quad := \quad \bigwedge_{j\in\{1,\ldots,c\}} C_j(z_j) \quad \wedge \quad \bigwedge_{j\neq j'} \neg E(z_j,z_{j'})$$

Representing Databases by Coloured Graphs

$$\varphi(x_1,\ldots,x_k) \quad \equiv_d \quad \bigvee_{i\in\mathcal{I}} \mathsf{sph}_{\tau_i}(x_1,\ldots,x_k) \quad \text{\& sentences}$$

$$\mathsf{sph}_{\tau}(\overline{x}_1,\ldots,\overline{x}_c) \quad \equiv_d \quad \bigwedge_{j\in\{1,\ldots,c\}} \mathsf{sph}_{\tau_j}(\overline{x}_j) \quad \wedge \quad \bigwedge_{j\neq j'} \neg\mathsf{dist}_{\leqslant 2r+1}(\overline{x}_j,\overline{x}_{j'})$$

$$\Big\Downarrow$$

$$\varphi_c(z_1,\ldots,z_c) \quad := \quad \bigwedge_{j\in\{1,\ldots,c\}} C_j(z_j) \quad \wedge \quad \bigwedge_{j\neq j'} \neg E(z_j,z_{j'})$$

$$C_j^{\mathcal{G}} := \{v_{\overline{a}} : \overline{a} \in \mathsf{adom}(D)^{|x_j|}, (\mathcal{N}_r^D(\overline{a}),\overline{a}) \cong \tau_j\}$$

Representing Databases by Coloured Graphs

$$\varphi(x_1,\ldots,x_k) \equiv_d \bigvee_{i\in\mathcal{I}} \mathsf{sph}_{\tau_i}(x_1,\ldots,x_k) \quad \& \text{ sentences}$$

$$\mathsf{sph}_{\tau}(\overline{x}_1,\ldots,\overline{x}_c) \equiv_d \bigwedge_{j\in\{1,\ldots,c\}} \mathsf{sph}_{\tau_j}(\overline{x}_j) \wedge \bigwedge_{j\neq j'} \neg\mathsf{dist}_{\leqslant 2r+1}(\overline{x}_j,\overline{x}_{j'})$$

$$\varphi_c(z_1,\ldots,z_c) := \bigwedge_{j\in\{1,\ldots,c\}} C_j(z_j) \wedge \bigwedge_{j\neq j'} \neg E(z_j,z_{j'})$$

$$C_j^{\mathcal{G}} := \{v_{\overline{a}} : \overline{a} \in \mathsf{adom}(D)^{|x_j|}, (\mathcal{N}_r^D(\overline{a}),\overline{a}) \cong \tau_j\}$$

$$V := \bigcup_{j\in\{1,\ldots,c\}} C_j^{\mathcal{G}}$$

Representing Databases by Coloured Graphs

$$\varphi(x_1,\ldots,x_k) \quad \equiv_d \quad \bigvee_{i\in\mathcal{I}} \mathrm{sph}_{\tau_i}(x_1,\ldots,x_k) \quad \text{\& sentences}$$

$$\mathrm{sph}_{\tau}(\overline{x}_1,\ldots,\overline{x}_c) \quad \equiv_d \quad \bigwedge_{j\in\{1,\ldots,c\}} \mathrm{sph}_{\tau_j}(\overline{x}_j) \quad \wedge \quad \bigwedge_{j\neq j'} \neg\mathrm{dist}_{\leqslant 2r+1}(\overline{x}_j,\overline{x}_{j'})$$

$$\varphi_c(z_1,\ldots,z_c) \quad := \quad \bigwedge_{j\in\{1,\ldots,c\}} C_j(z_j) \quad \wedge \quad \bigwedge_{j\neq j'} \neg E(z_j,z_{j'})$$

$$C_j^{\mathcal{G}} := \{v_{\overline{a}} : \overline{a} \in \mathrm{adom}(D)^{|x_j|}, (\mathcal{N}_r^D(\overline{a}),\overline{a}) \cong \tau_j\}$$

$$V := \bigcup_{j\in\{1,\ldots,c\}} C_j^{\mathcal{G}}$$

$$E^{\mathcal{G}} := \{(v_{\overline{a}}, v_{\overline{b}}) \in V^2 : \mathrm{dist}^D(\overline{a},\overline{b}) \leqslant 2r+1\}$$

Representing Databases by Coloured Graphs

$$\varphi(x_1,\ldots,x_k) \equiv_d \bigvee_{i\in\mathcal{I}} \mathrm{sph}_{\tau_i}(x_1,\ldots,x_k) \quad \& \text{ sentences}$$

$$\mathrm{sph}_{\tau}(\overline{x}_1,\ldots,\overline{x}_c) \equiv_d \bigwedge_{j\in\{1,\ldots,c\}} \mathrm{sph}_{\tau_j}(\overline{x}_j) \wedge \bigwedge_{j\neq j'} \neg\mathrm{dist}_{\leqslant 2r+1}(\overline{x}_j,\overline{x}_{j'})$$

$$\varphi_c(z_1,\ldots,z_c) := \bigwedge_{j\in\{1,\ldots,c\}} C_j(z_j) \wedge \bigwedge_{j\neq j'} \neg E(z_j,z_{j'})$$

$$C_j^{\mathcal{G}} := \{v_{\overline{a}} : \overline{a} \in \mathrm{adom}(D)^{|x_j|}, (\mathcal{N}_r^D(\overline{a}),\overline{a}) \cong \tau_j\}$$

$$V := \bigcup_{j\in\{1,\ldots,c\}} C_j^{\mathcal{G}}$$

$$E^{\mathcal{G}} := \{(v_{\overline{a}}, v_{\overline{b}}) \in V^2 : \mathrm{dist}^D(\overline{a},\overline{b}) \leqslant 2r+1\}$$

$$(\overline{a}_1,\ldots,\overline{a}_c) \in \mathrm{sph}_{\tau}(D) \iff (v_{\overline{a}_1},\ldots,v_{\overline{a}_c}) \in \varphi_c(\mathcal{G})$$

Representing Databases by Coloured Graphs

$$\varphi(x_1,\ldots,x_k) \equiv_d \bigvee_{i\in\mathcal{I}} \mathrm{sph}_{\tau_i}(x_1,\ldots,x_k) \quad \text{\& sentences}$$

$$\mathrm{sph}_{\tau}(\overline{x}_1,\ldots,\overline{x}_c) \equiv_d \bigwedge_{j\in\{1,\ldots,c\}} \mathrm{sph}_{\tau_j}(\overline{x}_j) \wedge \bigwedge_{j\neq j'} \neg\mathrm{dist}_{\leqslant 2r+1}(\overline{x}_j,\overline{x}_{j'})$$

$$\varphi_c(z_1,\ldots,z_c) := \bigwedge_{j\in\{1,\ldots,c\}} C_j(z_j) \wedge \bigwedge_{j\neq j'} \neg E(z_j,z_{j'})$$

$$C_j^{\mathcal{G}} := \{v_{\overline{a}} : \overline{a} \in \mathrm{adom}(D)^{|x_j|}, (\mathcal{N}_r^D(\overline{a}),\overline{a}) \cong \tau_j\}$$

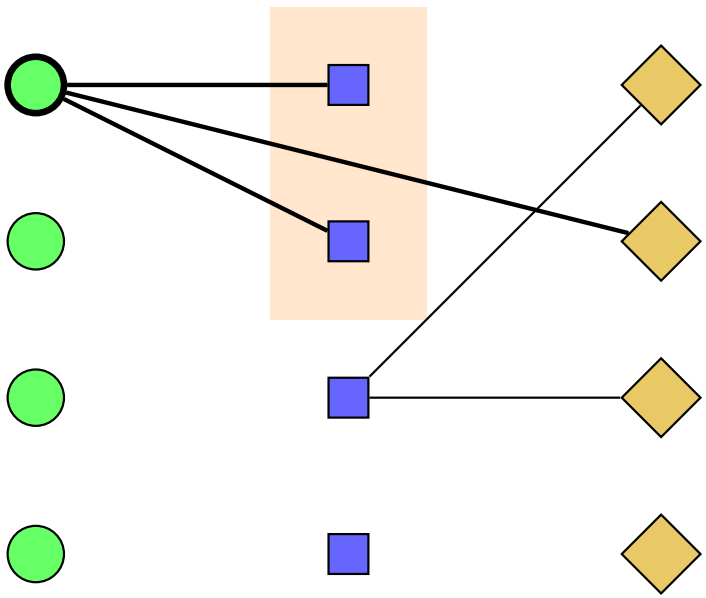
$$V := \bigcup_{j\in\{1,\ldots,c\}} C_j^{\mathcal{G}} \qquad E^{\mathcal{G}} := \{(v_{\overline{a}},v_{\overline{b}}) \in V^2 : \mathrm{dist}^D(\overline{a},\overline{b}) \leqslant 2r+1\}$$

$$(\overline{a}_1,\ldots,\overline{a}_c) \in \mathrm{sph}_{\tau}(D) \iff (v_{\overline{a}_1},\ldots,v_{\overline{a}_c}) \in \varphi_c(\mathcal{G})$$

When a tuple is inserted/deleted in D, update the coloured graph $\mathcal{G}$.

Enumeration of $\psi_k(\mathcal{G})$ with delay $O(k^3 d)$

$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



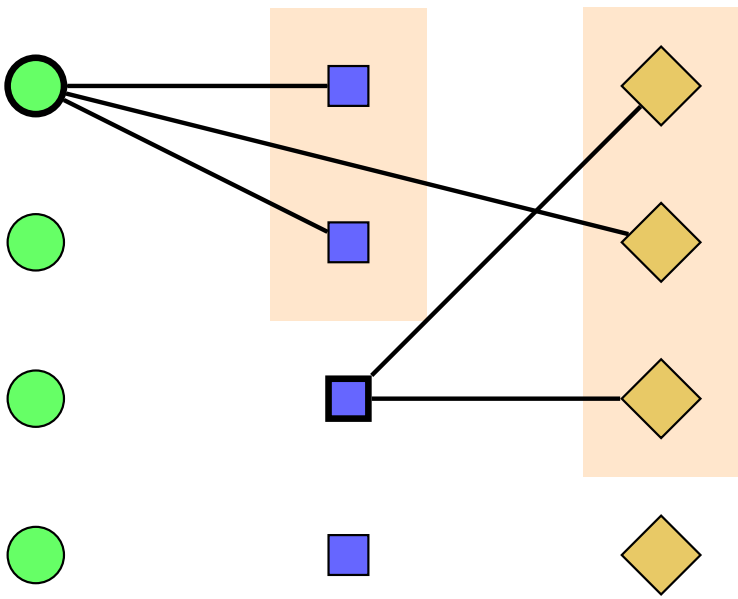
for all $u_1 \in C_1^{\mathcal{G}}$ **do**
    $\texttt{Enum}(u_1)$.

Output EOE.

function $\textsc{Enum}(u_1, \ldots, u_i)$
    if $i = k$ **then**
        Output $(u_1, \ldots, u_i)$
    else
        for all $u_{i+1} \in C_{i+1}^{\mathcal{G}}$ **do**
            if $u_{i+1} \notin \bigcup_{j=1}^{i} N^{\mathcal{G}}(u_j)$ **then**
                $\texttt{Enum}(u_1, \ldots, u_i, u_{i+1})$

Enumeration of $\psi_k(\mathcal{G})$ with delay $O(k^3 d)$

$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



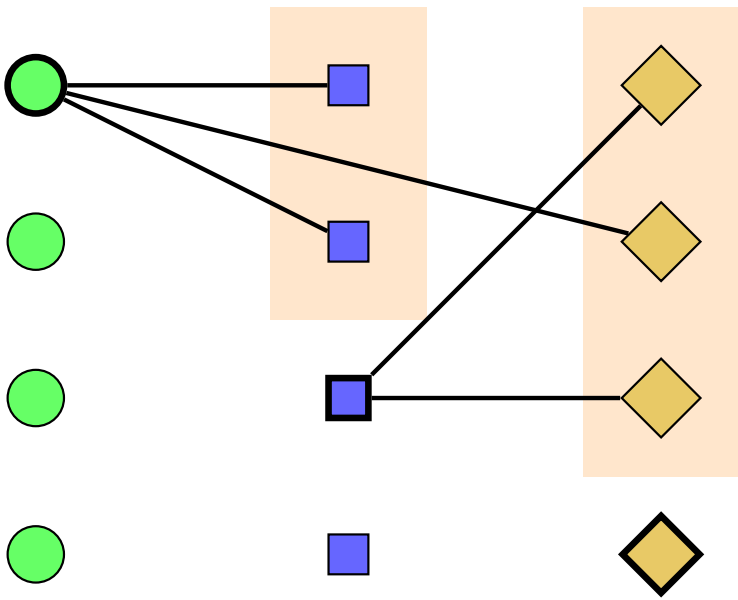
for all $u_1 \in C_1^{\mathcal{G}}$ **do**
    $\texttt{Enum}(u_1)$.

Output EOE.

function $\textsc{Enum}(u_1, \ldots, u_i)$
    if $i = k$ **then**
        Output $(u_1, \ldots, u_i)$
    else
        for all $u_{i+1} \in C_{i+1}^{\mathcal{G}}$ **do**
            if $u_{i+1} \notin \bigcup_{j=1}^{i} N^{\mathcal{G}}(u_j)$ **then**
                $\texttt{Enum}(u_1, \ldots, u_i, u_{i+1})$

Enumeration of $\psi_k(\mathcal{G})$ with delay $O(k^3 d)$

$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



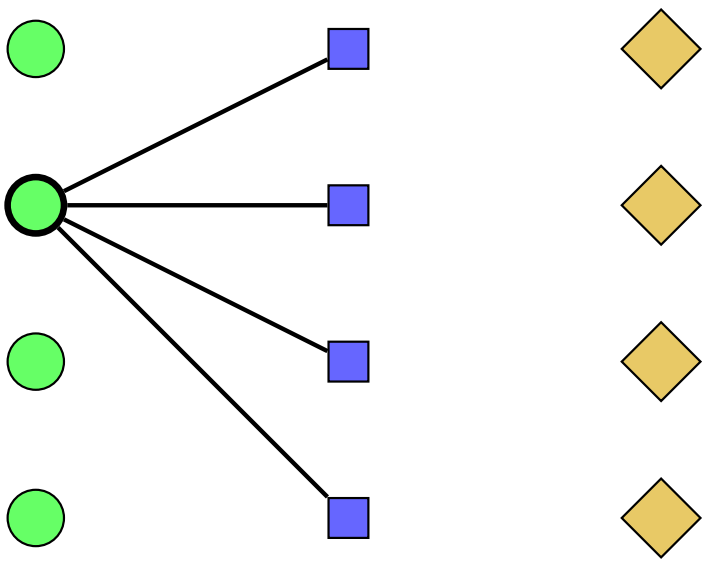
for all $u_1 \in C_1^{\mathcal{G}}$ **do**
    $\texttt{Enum}(u_1)$.

Output EOE.

function $\textsc{Enum}(u_1, \ldots, u_i)$
    if $i = k$ **then**
        Output $(u_1, \ldots, u_i)$
    else
        for all $u_{i+1} \in C_{i+1}^{\mathcal{G}}$ **do**
            if $u_{i+1} \notin \bigcup_{j=1}^{i} N^{\mathcal{G}}(u_j)$ **then**
                $\texttt{Enum}(u_1, \ldots, u_i, u_{i+1})$

Enumeration of $\psi_k(\mathcal{G})$ with delay $O(k^3 d)$

$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



Problem: Too few blue nodes

for all $u_1 \in C_1^{\mathcal{G}}$ **do**
    $\texttt{Enum}(u_1)$.

Output EOE.

function $\textsc{Enum}(u_1, \ldots, u_i)$
    if $i = k$ **then**
        Output $(u_1, \ldots, u_i)$
    else
        for all $u_{i+1} \in C_{i+1}^{\mathcal{G}}$ **do**
            if $u_{i+1} \notin \bigcup_{j=1}^{i} N^{\mathcal{G}}(u_j)$ **then**
                $\texttt{Enum}(u_1, \ldots, u_i, u_{i+1})$

Handling small colours

$$\text{A colour } \ell \in \{1, \ldots, k\} \text{ is small} \quad :\Longleftrightarrow \quad \left| C_\ell^{\mathcal{G}} \right| \leqslant dk$$

Handling small colours

$$\text{A colour } \ell \in \{1, \ldots, k\} \text{ is small} \quad :\Longleftrightarrow \quad \left| C_\ell^{\mathcal{G}} \right| \leqslant dk$$

W.l.o.g. let $I = \{1, \ldots, s\}$ be the set of small colours (with $s \leqslant k$).

Handling small colours

$$\text{A colour } \ell \in \{1, \ldots, k\} \text{ is small} \quad :\Longleftrightarrow \quad \left|C_\ell^{\mathcal{G}}\right| \leqslant dk$$

W.l.o.g. let $I = \{1, \ldots, s\}$ be the set of small colours (with $s \leqslant k$).

$$\mathcal{S} \quad := \quad \left\{ (u_1, \ldots, u_s) \in C_1^{\mathcal{G}} \times \cdots \times C_s^{\mathcal{G}} \; : \; \begin{array}{c} (u_j, u_{j'}) \notin E^{\mathcal{G}}, \\ \text{for all } j \neq j' \end{array} \right\}$$

The set $\mathcal{S}$ can be computed in time $O((dk)^k)$.

Handling small colours

$$\text{A colour } \ell \in \{1, \ldots, k\} \text{ is small} \quad :\Longleftrightarrow \quad \left|C_\ell^{\mathcal{G}}\right| \leqslant dk$$

W.l.o.g. let $I = \{1, \ldots, s\}$ be the set of small colours (with $s \leqslant k$).

$$\mathcal{S} \quad := \quad \left\{ (u_1, \ldots, u_s) \in C_1^{\mathcal{G}} \times \cdots \times C_s^{\mathcal{G}} \; : \; \begin{array}{c} (u_j, u_{j'}) \notin E^{\mathcal{G}}, \\ \text{for all } j \neq j' \end{array} \right\}$$

The set $\mathcal{S}$ can be computed in time $O((dk)^k)$.

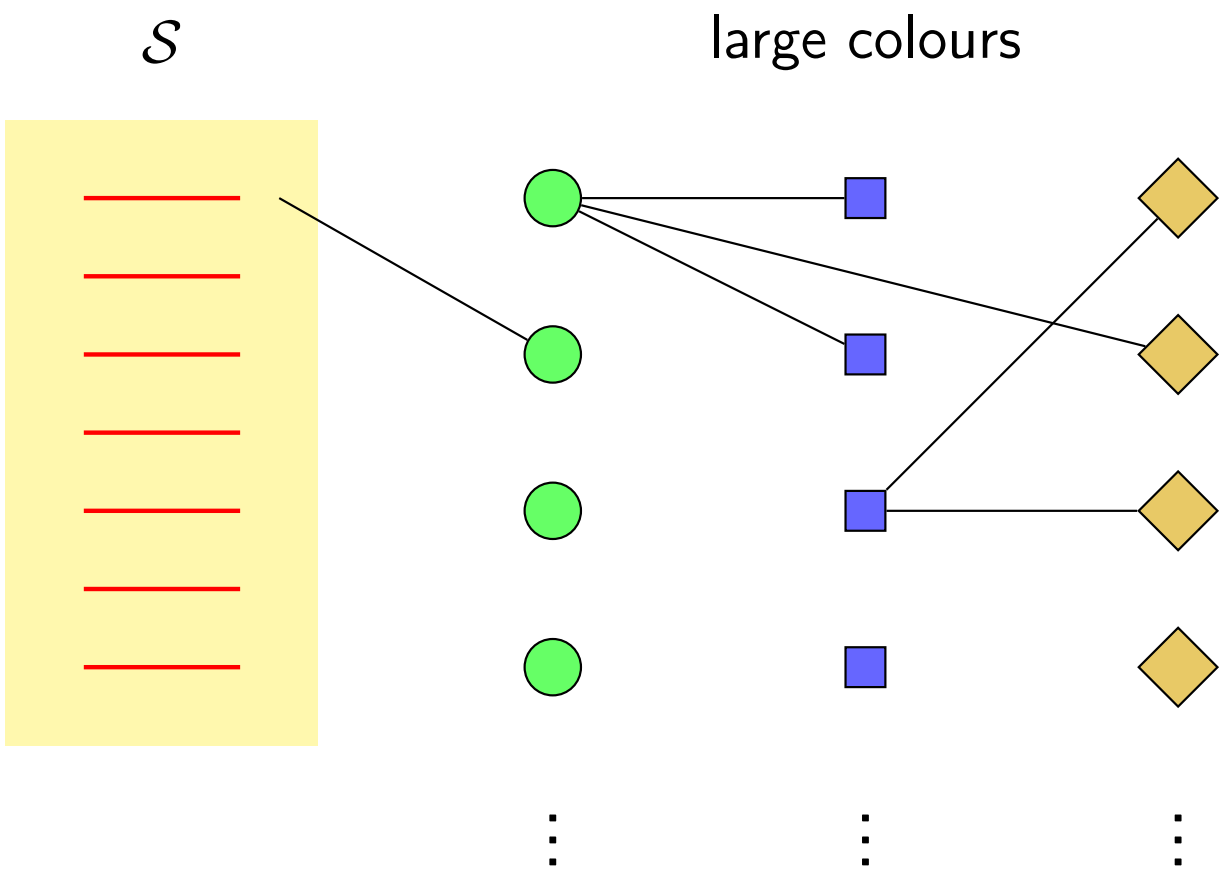
$$\overline{s} \in \mathcal{S} \quad \Longleftrightarrow \quad \text{ex. } \overline{a} \text{ such that } \; (\overline{s}, \overline{a}) \in \varphi(D)$$

The enumeration procedure

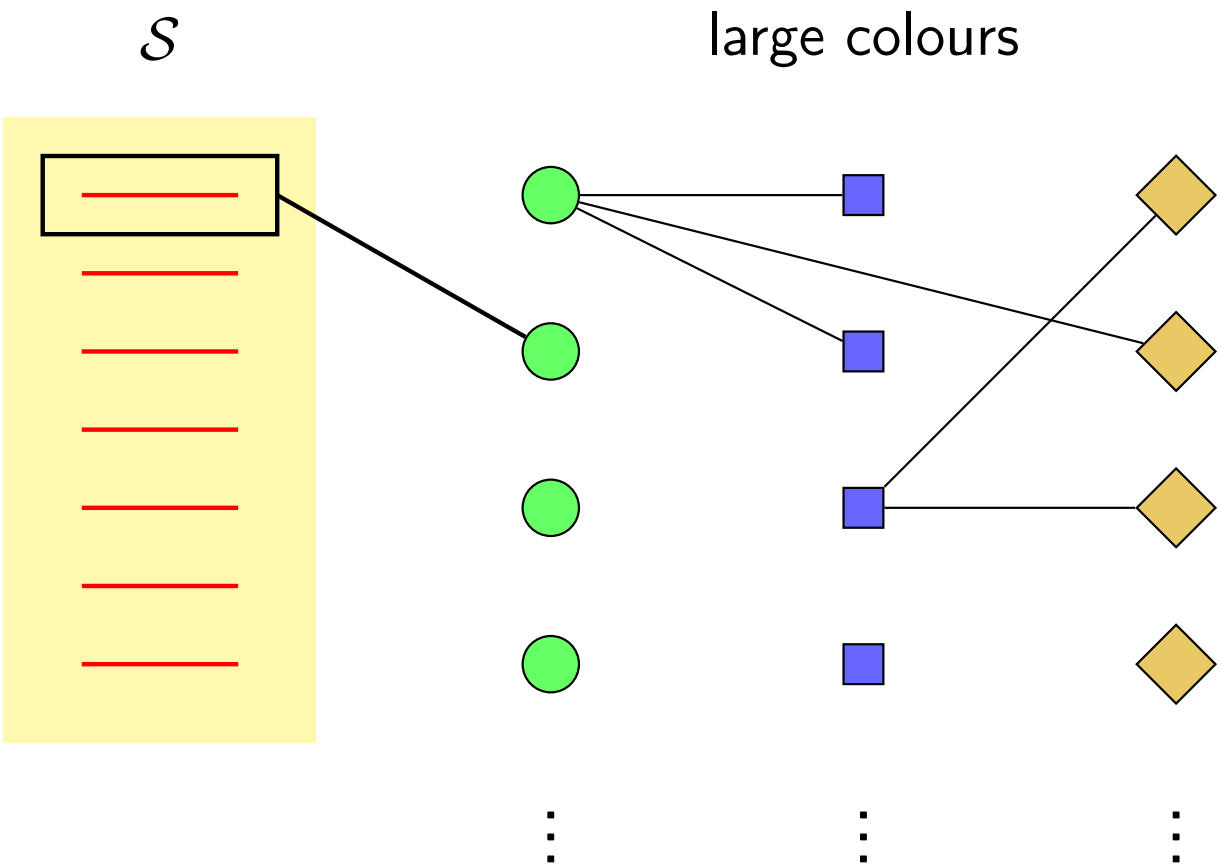
1: **for all** $(u_1, \ldots, u_s) \in \mathcal{S}$ **do**
2: $\quad$ $\texttt{Enum}(u_1, \ldots, u_s)$.
3: Output the end-of-enumeration message $\texttt{EOE}$.
4:
5: **function** $\textsc{Enum}(u_1, \ldots, u_i)$
6: $\quad$ **if** $i = k$ **then**
7: $\quad\quad$ output the tuple $(u_1, \ldots, u_i)$
8: $\quad$ **else**
9: $\quad\quad$ **for all** $u_{i+1} \in C_{i+1}^{\mathcal{G}}$ **do**
10: $\quad\quad\quad$ **if** $u_{i+1} \notin \bigcup_{j=1}^{i} N^{\mathcal{G}}(u_j)$ **then**
11: $\quad\quad\quad\quad$ $\texttt{Enum}(u_1, \ldots, u_i, u_{i+1})$

where $N^{\mathcal{G}}(u_j) := \{v \in V^{\mathcal{G}} : (u_j, v) \in E^{\mathcal{G}}\}$.

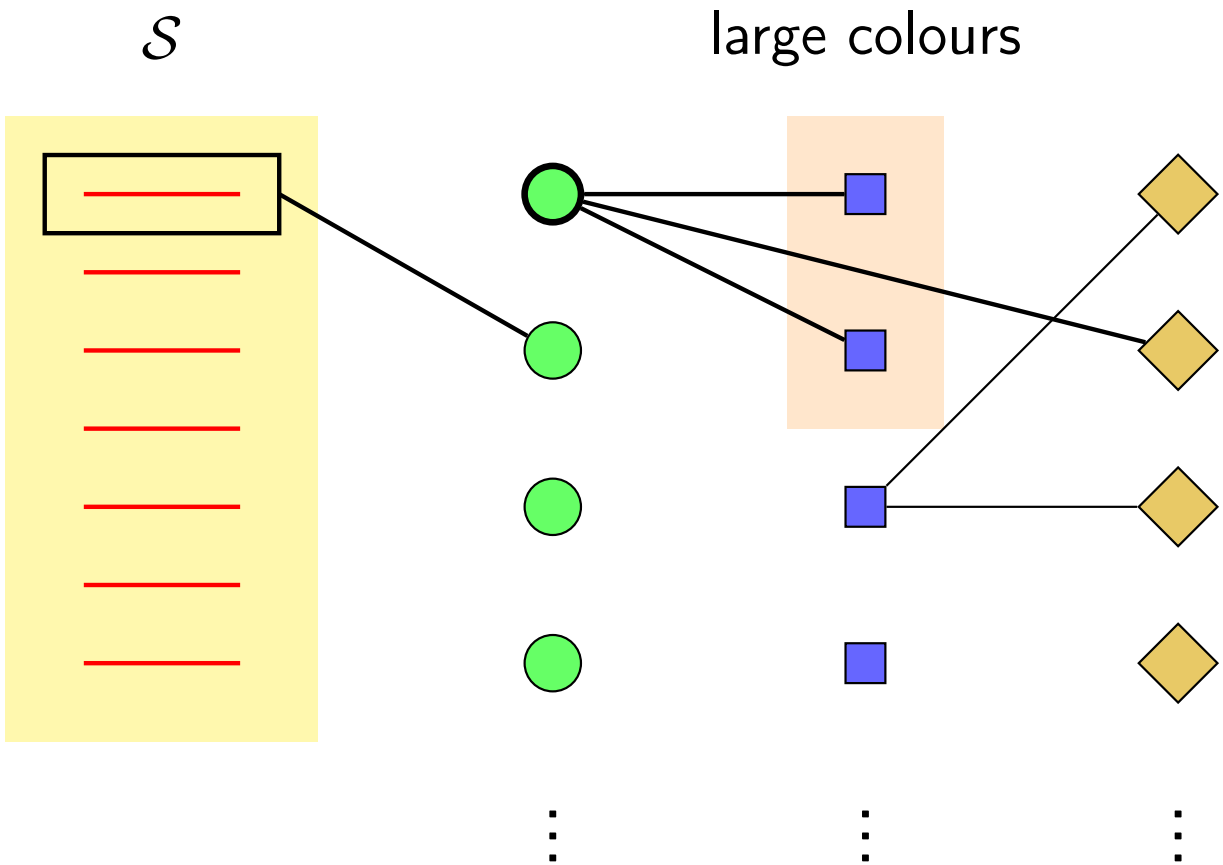
$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



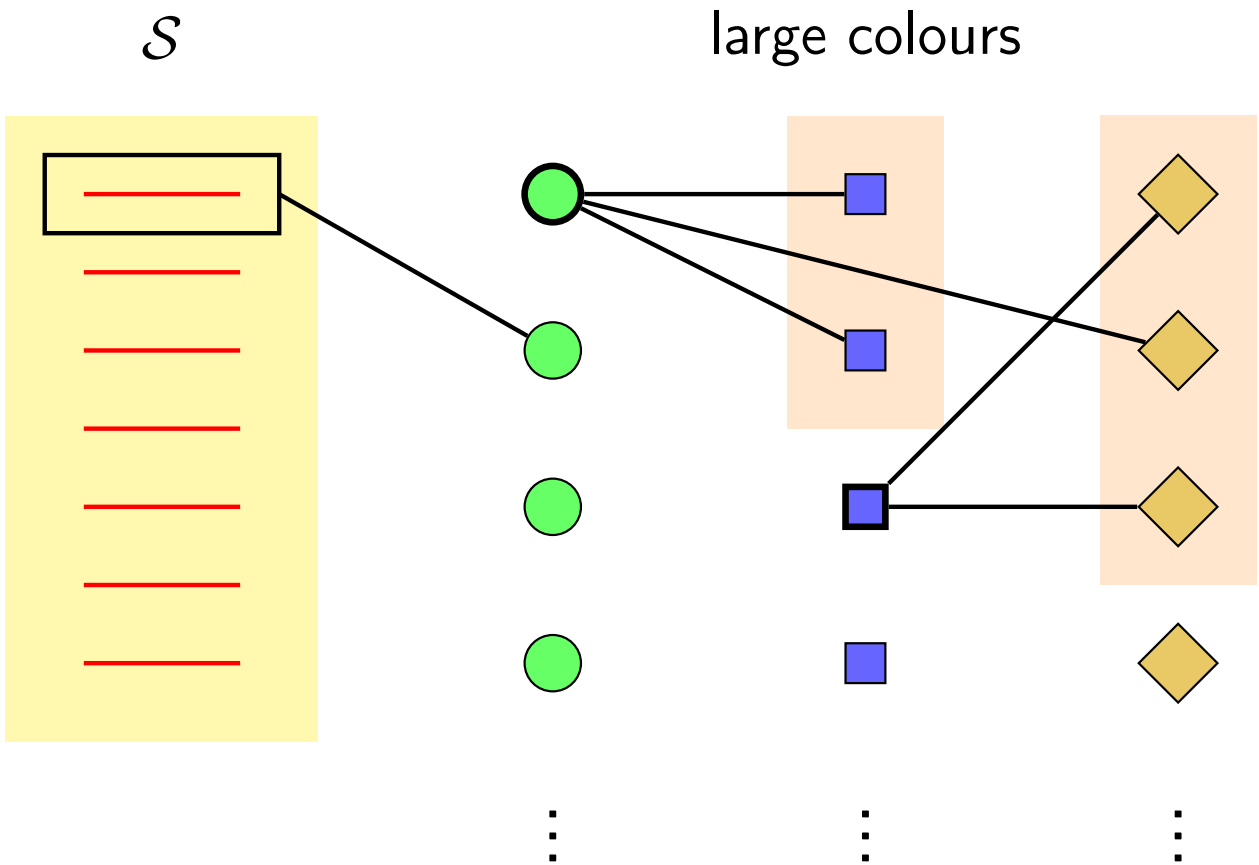
$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



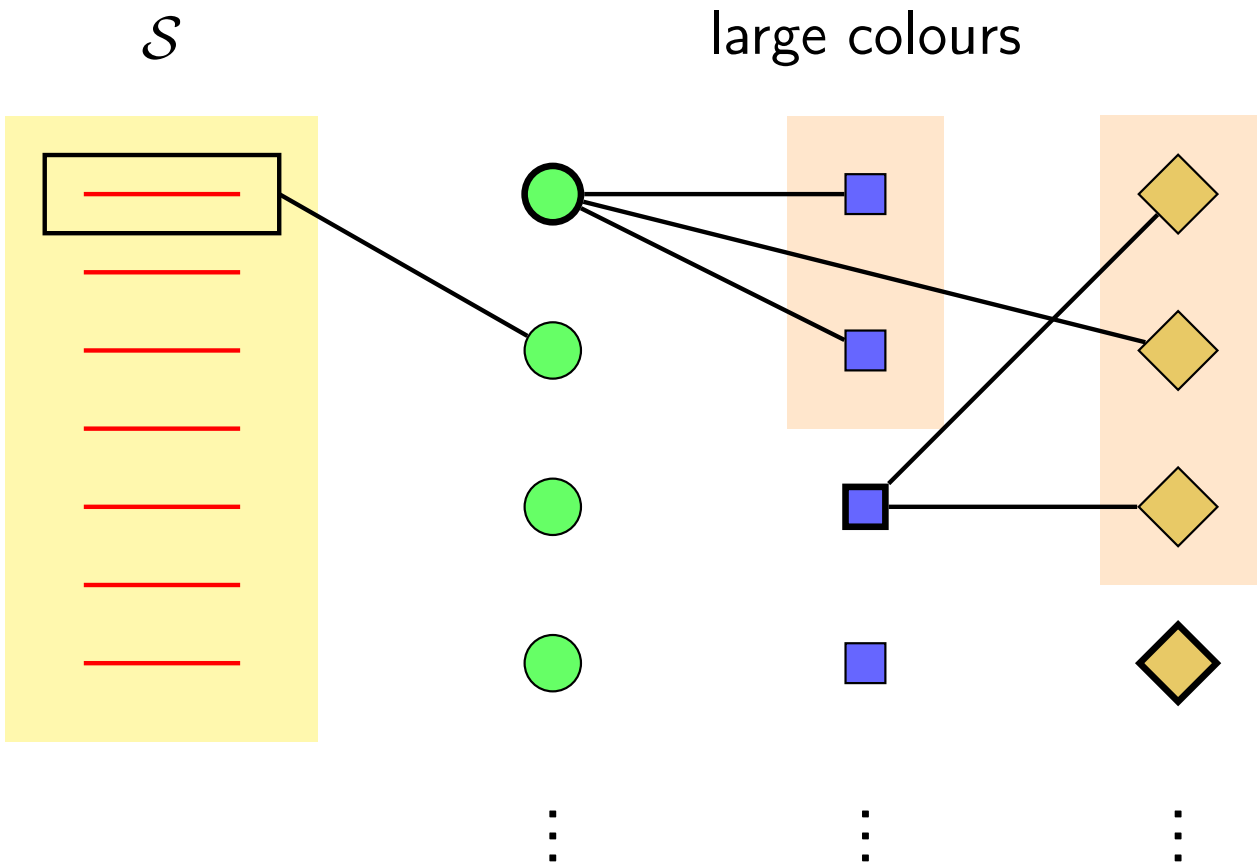
$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



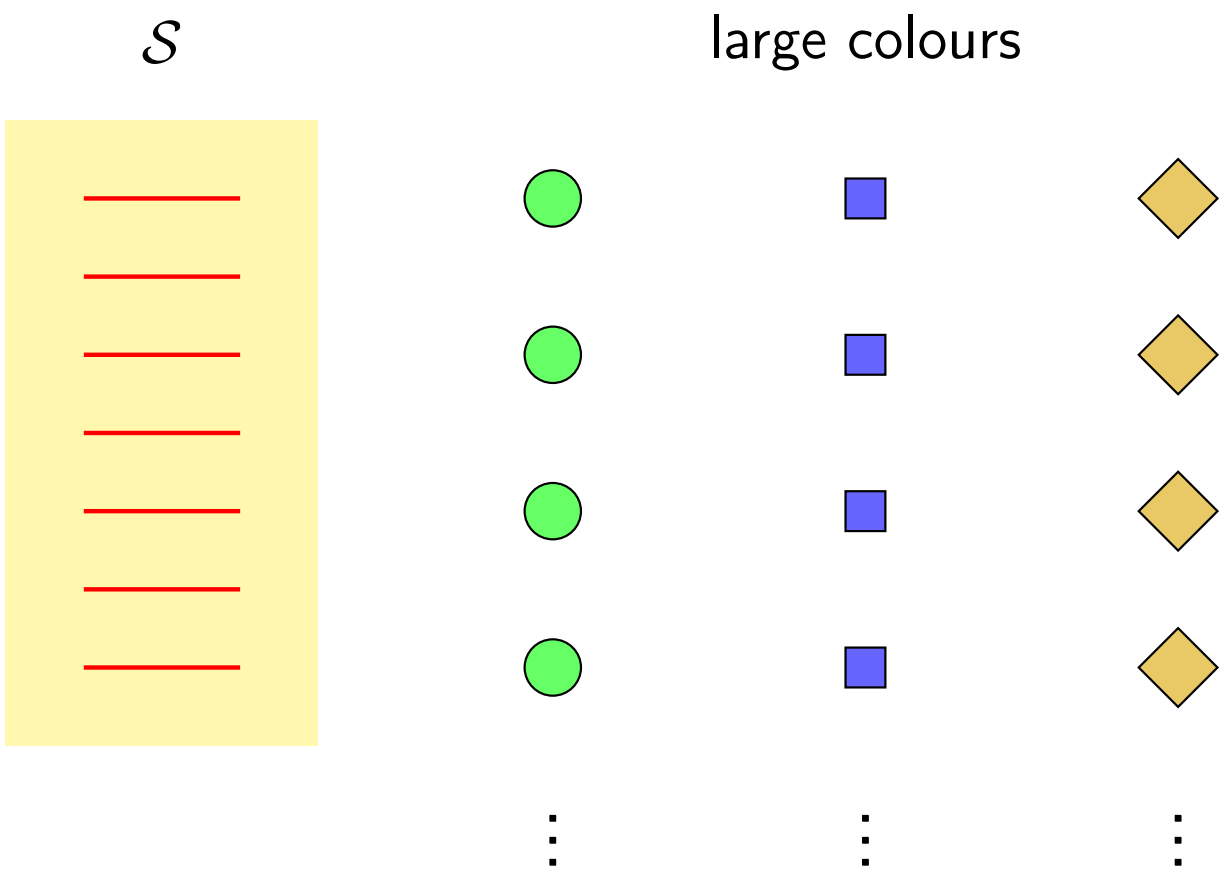
$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$

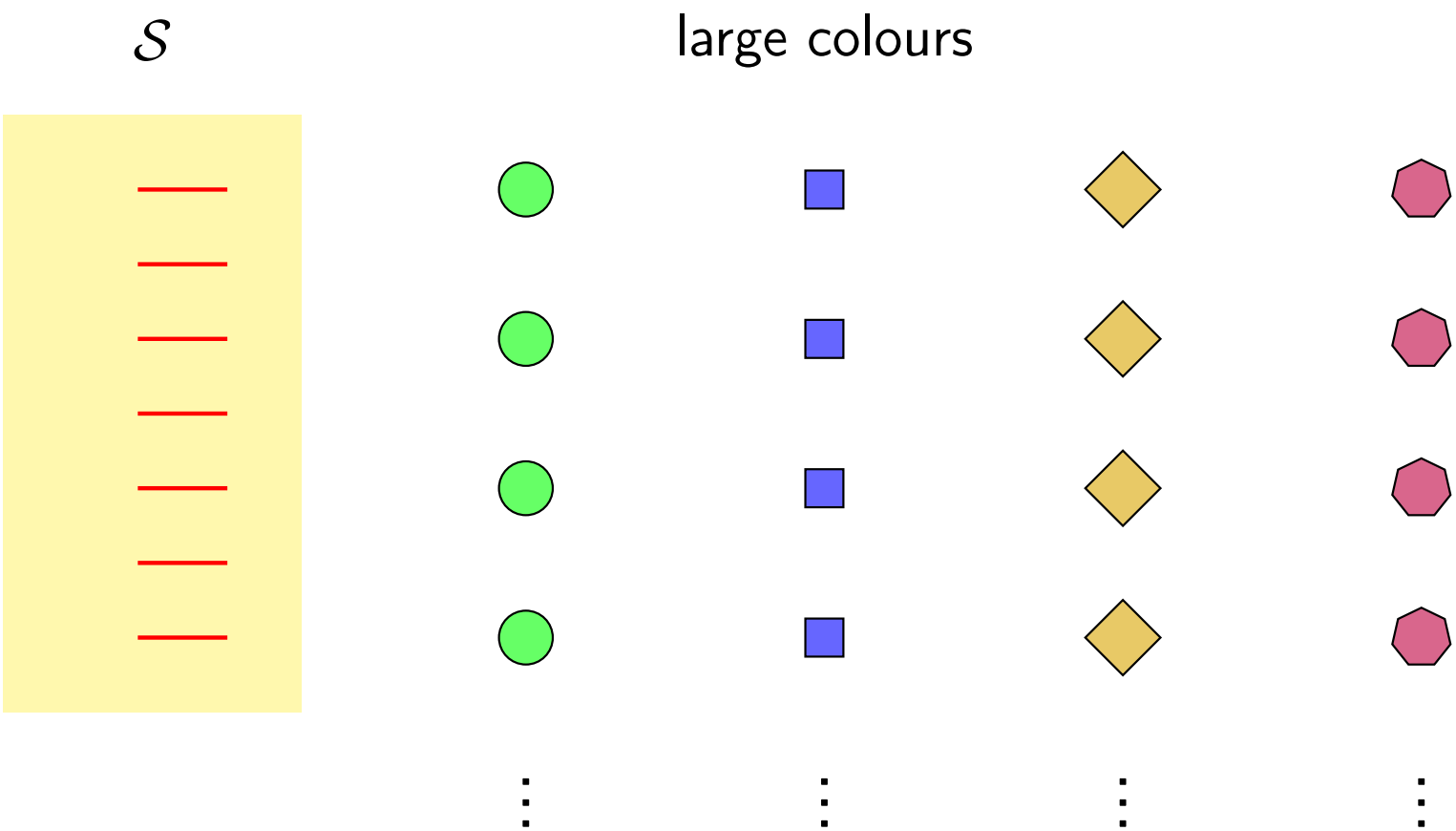


$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



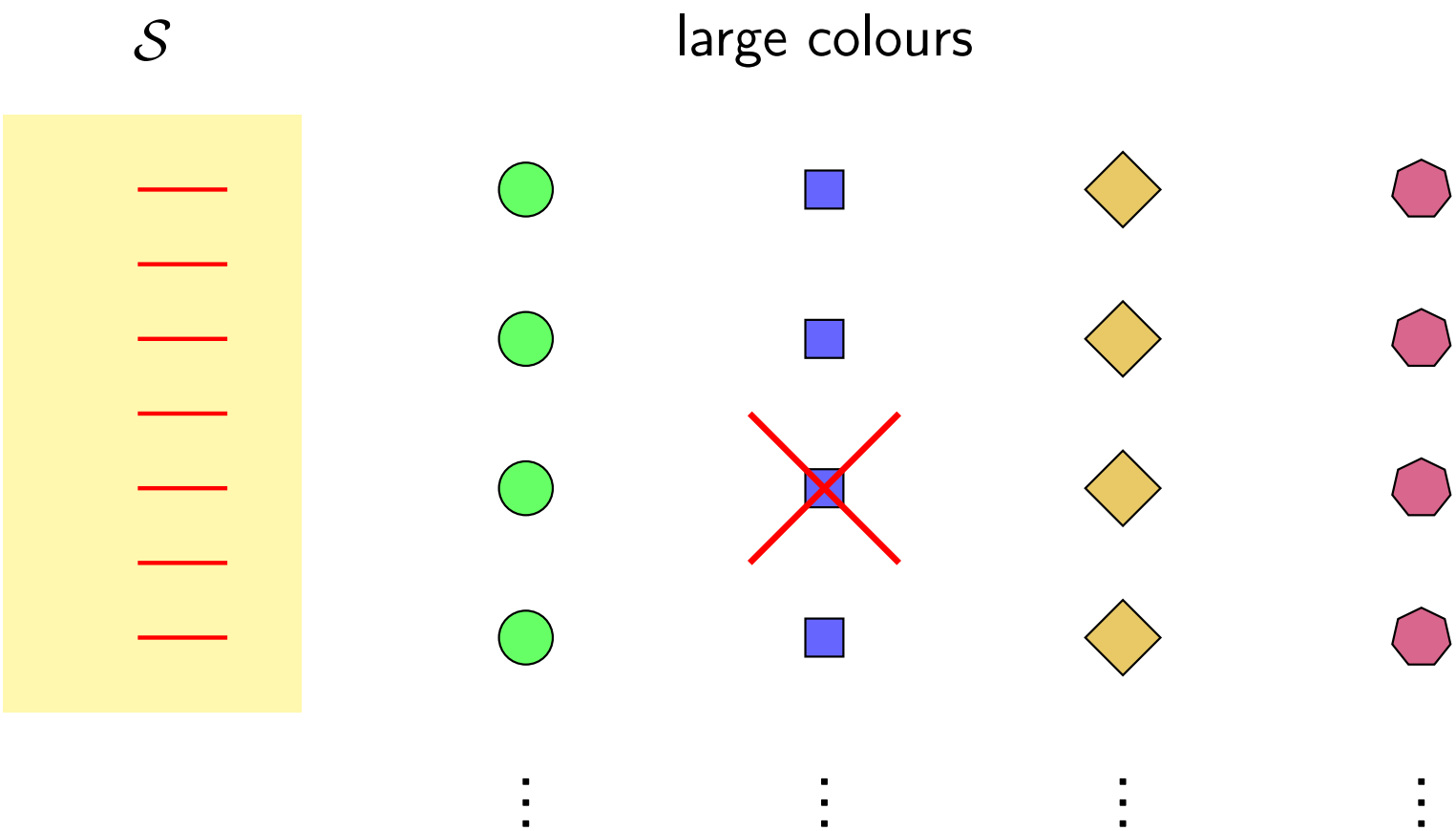
`update step:` Insert a node into a colour C_ℓ with $|C_\ell^{\mathcal{G}}| = dk$

$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



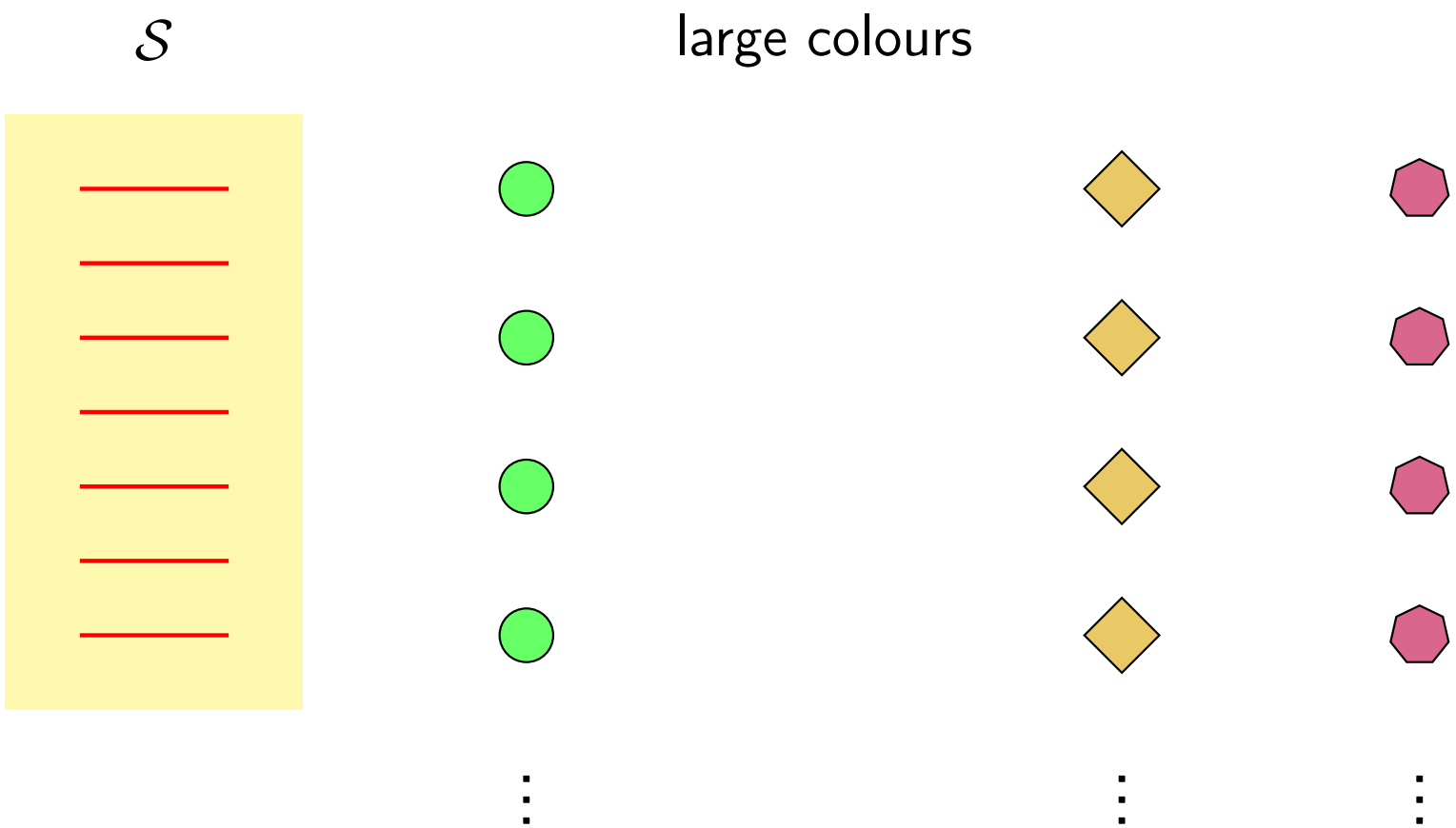
`update step:` Insert a node into a colour C_ℓ with $|C_\ell^{\mathcal{G}}| = dk$

$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



`update step:` Delete a node from a colour C_ℓ with $|C_\ell^{\mathcal{G}}| = dk + 1$

$$\psi_k(x_1, \ldots, x_k) \quad := \quad \bigwedge_{i=1}^{k} C_i(x_i) \ \wedge \ \bigwedge_{i \neq j} \neg E(x_i, x_j)$$



`update step:` Delete a node from a colour C_ℓ with $|C_\ell^{\mathcal{G}}| = dk + 1$

Enumeration under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Theorem (Berkholz, Keppeler, S. 2017): There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a k-ary FO+MOD-query $\varphi(\overline{x})$, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to enumerate $\varphi(D)$ with delay $O(k^2)$.

Enumeration under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Theorem (Berkholz, Keppeler, S. 2017): There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a k-ary FO+MOD-query $\varphi(\overline{x})$, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to enumerate $\varphi(D)$ with delay ~~$O(k^2)$~~ $f(\varphi, d)$.

Enumeration under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Theorem (Berkholz, Keppeler, S. 2017): There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a k-ary FO+MOD-query $\varphi(\overline{x})$, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to enumerate $\varphi(D)$ with delay ~~$O(k^2)$~~ ~~$f(\varphi, d)$~~ $O(k^2)$.

For enumeration with delay $O(k^2)$: Use the skip-pointers that were introduced by [Durand, S., Segoufin 2014] for the static setting and lift the approach to the dynamic setting.

Enumeration under updates $f(\varphi, d) = 3\text{-exp}(\|\varphi\| + \lg\lg d)$

Theorem (Berkholz, Keppeler, S. 2017): There is a dynamic algorithm that receives as input

- a degree bound $d \geqslant 2$,
- a k-ary FO+MOD-query $\varphi(\overline{x})$, and
- a db D of degree $\leqslant d$,

and computes

- within $f(\varphi, d)\|D\|$ preprocessing time a data structure
- that can be updated in time $f(\varphi, d)$

and allows to enumerate $\varphi(D)$ with delay ~~$O(k^2)$~~ ~~$f(\varphi, d)$~~ $O(k^2)$.

For enumeration with delay $O(k^2)$: Use the skip-pointers that were introduced by [Durand, S., Segoufin 2014] for the static setting and lift the approach to the dynamic setting.

For the generalisation to FOC($\mathbb{P}$) (FO + counting + numerical predicates from $\mathbb{P}$) use a corresponding HNF for that logic [Kuske, S. 2017].

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

  For any subgraph-closed class C that is not nowhere dense, evaluation of Boolean FO-queries is not possible in time $f(\|\varphi\|)\|D\|^{O(1)}$
  (assuming FPT $\neq$ W[1]) [Kreutzer 2011, Dvořák, Král, Thomas 2010]

- C has low degree [Durand, S., Segoufin 2014]

  i.e., for each $\delta > 0$, all sufficiently large D in C have degree $\leqslant \|D\|^{\delta}$.

  Some such C are not nowhere dense (and not subgraph-closed).

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

  For any subgraph-closed class C that is not nowhere dense, evaluation of Boolean FO-queries is not possible in time $f(\|\varphi\|)\|D\|^{O(1)}$ (assuming FPT $\neq$ W[1]) [Kreutzer 2011, Dvořák, Král, Thomas 2010]

- C has low degree [Durand, S., Segoufin 2014]

  i.e., for each $\delta > 0$, all sufficiently large D in C have degree $\leqslant \|D\|^{\delta}$.

  Some such C are not nowhere dense (and not subgraph-closed).

Open: To which extensions of FO can this be generalised?
Which results can be extended to the dynamic setting?

Overview

Constant delay enumeration after linear time preprocessing

- Let φ be a fixed query
- Let C be a class of databases
- Goal:
  Find an algorithm which, upon input of a db D,
  uses preprocessing time linear in $\|D\|$ and then
  enumerates $\varphi(D)$ with delay $O(1)$

- Question:
  For which φ, C is this possible?

Now: φ is a CQ $\qquad \varphi(x_1, \ldots, x_k) \;=\; \exists y_1 \cdots \exists y_\ell \big(R_1(\ldots) \wedge \cdots \wedge R_s(\ldots)\big)$

C is the class of all dbs

Obvious condition: Boolean φ must be tractable in linear time

$\rightsquigarrow$ restrict attention to acyclic CQs [Yannakakis 1981]

Examples (static setting for acyclic CQs)

$$\varphi_{S\text{-}E\text{-}T}(x,y) \quad := \quad S(x) \wedge E(x,y) \wedge T(y)$$

can be enumerated with constant delay after linear time preprocessing:

- For input db $D = (\mathrm{adom}(D), E^D, S^D, T^D)$
- Compute $E_1 := \{(x,y) \in E^D : x \in S^D\}$
- Compute $E_2 := \{(x,y) \in E_1 : y \in T^D\}$
- Output E_2

Examples (static setting for acyclic CQs)

$$\varphi_{S\text{-}E\text{-}T}(x,y) \quad := \quad S(x) \wedge E(x,y) \wedge T(y)$$

can be enumerated with constant delay after linear time preprocessing:

- For input db $D = (\mathrm{adom}(D), E^D, S^D, T^D)$
- Compute $E_1 := \{(x,y) \in E^D : x \in S^D\}$
- Compute $E_2 := \{(x,y) \in E_1 : y \in T^D\}$
- Output E_2

Same approach works for the acyclic CQ

$$\varphi_{E\text{-}T}(x) \quad := \quad \exists y \, \big(\, E(x,y) \wedge T(y) \, \big) \, .$$

Examples (static setting for acyclic CQs)

$$\varphi_{S\text{-}E\text{-}T}(x,y) \quad := \quad S(x) \wedge E(x,y) \wedge T(y)$$

can be enumerated with constant delay after linear time preprocessing:

- For input db $D = (\mathrm{adom}(D), E^D, S^D, T^D)$
- Compute $E_1 := \{(x,y) \in E^D : x \in S^D\}$
- Compute $E_2 := \{(x,y) \in E_1 : y \in T^D\}$
- Output E_2

Same approach works for the acyclic CQ

$$\varphi_{E\text{-}T}(x) \quad := \quad \exists y \, \big(\, E(x,y) \wedge T(y) \, \big) \, .$$

But apparently <u>not</u> for the acyclic CQ

$$\varphi_{AB}(x,y) \quad := \quad \exists z \, \big(\, A(x,z) \wedge B(z,y) \, \big)$$

Examples (static setting for acyclic CQs)

$$\varphi_{S\text{-}E\text{-}T}(x,y) \quad := \quad S(x) \wedge E(x,y) \wedge T(y)$$

can be enumerated with constant delay after linear time preprocessing:

- For input db $D = (\mathrm{adom}(D), E^D, S^D, T^D)$
- Compute $E_1 := \{(x,y) \in E^D : x \in S^D\}$
- Compute $E_2 := \{(x,y) \in E_1 : y \in T^D\}$
- Output E_2

Same approach works for the acyclic CQ

$$\varphi_{E\text{-}T}(x) \quad := \quad \exists y \, \big(\, E(x,y) \wedge T(y) \, \big) \, .$$

But apparently <u>not</u> for the acyclic CQ

$$\varphi_{AB}(x,y) \quad := \quad \exists z \, \big(\, A(x,z) \wedge B(z,y) \, \big)$$

otherwise, Boolean Matrix Multiplication (for two $n \times n$ matrices A and B) could be solved in time $O(n^2)$, contradicting an algorithmic assumption. $n = |\mathrm{adom}(D)|$

Characterisation for the static setting

Definition: A CQ φ is acyclic if there exists a join tree, i.e., a tree t whose nodes are exactly the atoms of φ, and where for any two nodes u, v the following is true for every common variable y of u and v: y occurs in every node on the path between u and v in t.

Characterisation for the static setting

Definition: A CQ φ is acyclic if there exists a join tree, i.e., a tree t whose nodes are exactly the atoms of φ, and where for any two nodes u, v the following is true for every common variable y of u and v: y occurs in every node on the path between u and v in t.

Definition: φ is free-connex if the following two queries are acyclic: φ and the query φ' obtained from φ by adding an atom of the form $R(\overline{x})$ that contains all free variables of φ [Brault-Baron 2013]

Characterisation for the static setting

Definition: A CQ φ is acyclic if there exists a join tree, i.e., a tree t whose nodes are exactly the atoms of φ, and where for any two nodes u, v the following is true for every common variable y of u and v: y occurs in every node on the path between u and v in t.

Definition: φ is free-connex if the following two queries are acyclic: φ and the query φ' obtained from φ by adding an atom of the form $R(\overline{x})$ that contains all free variables of φ [Brault-Baron 2013]

Theorem (Bagan, Durand, Grandjean 2007)

Let $\varphi(x_1, \ldots, x_k)$ be an acyclic CQ.

If φ is free-connex, then $\varphi(D)$ can be enumerated with constant delay after linear preprocessing.

Characterisation for the static setting

Definition: A CQ φ is acyclic if there exists a join tree, i.e., a tree t whose nodes are exactly the atoms of φ, and where for any two nodes u, v the following is true for every common variable y of u and v: y occurs in every node on the path between u and v in t.

Definition: φ is free-connex if the following two queries are acyclic: φ and the query φ' obtained from φ by adding an atom of the form $R(\overline{x})$ that contains all free variables of φ [Brault-Baron 2013]

Theorem (Bagan, Durand, Grandjean 2007)

Let $\varphi(x_1, \ldots, x_k)$ be an acyclic CQ.

If φ is free-connex, then $\varphi(D)$ can be enumerated with constant delay after linear preprocessing.

If φ is self-join free and not free-connex, then it is not possible to enumerate $\varphi(D)$ with constant delay after linear preprocessing, unless Boolean Matrix Multiplication can be done in time $O(n^2)$.

Dynamic setting

Queries that are easy in the static setting
can be difficult for the dynamic setting!

Example: $\varphi_{E\text{-}T}(x) \;:=\; \exists y \,\big(\, E(x,y) \,\wedge\, T(y) \,\big)\,.$

Why? — See the next few slides:

Introduce the notion of q-hierarchical CQs, which captures the easy CQs in the dynamic setting

q-hierarchical CQs

Dalvi & Suciu (2007) introduced the hierarchical CQs to characterise the Boolean CQs that can be answered in PTIME on probabilistic dbs.

q-hierarchical CQs

Dalvi & Suciu (2007) introduced the hierarchical CQs to characterise the Boolean CQs that can be answered in PTIME on probabilistic dbs.

q-hierarchical CQs are hierarchical CQs where, additionally, the quantifiers respect the query's hierarchical form.

q-hierarchical CQs

Dalvi & Suciu (2007) introduced the hierarchical CQs to characterise the Boolean CQs that can be answered in PTIME on probabilistic dbs.

q-hierarchical CQs are hierarchical CQs where, additionally, the quantifiers respect the query's hierarchical form.

Definition: A CQ $\varphi(z_1, \ldots, z_k)$ is q-hierarchical if for all variables x, y of φ the following is satisfied:

(i) $\text{atoms}(x) \subseteq \text{atoms}(y)$ or $\text{atoms}(y) \subseteq \text{atoms}(x)$ or $\text{atoms}(x) \cap \text{atoms}(y) = \emptyset$, and

(ii) if $\text{atoms}(x) \subsetneq \text{atoms}(y)$ and $x \in \text{free}(\varphi)$, then $y \in \text{free}(\varphi)$.

q-hierarchical CQs

Dalvi & Suciu (2007) introduced the hierarchical CQs to characterise the Boolean CQs that can be answered in PTIME on probabilistic dbs.

q-hierarchical CQs are hierarchical CQs where, additionally, the quantifiers respect the query's hierarchical form.

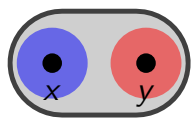
Definition: A CQ $\varphi(z_1, \ldots, z_k)$ is q-hierarchical if for all variables x, y of φ the following is satisfied:

(i) $\text{atoms}(x) \subseteq \text{atoms}(y)$ or $\text{atoms}(y) \subseteq \text{atoms}(x)$ or $\text{atoms}(x) \cap \text{atoms}(y) = \emptyset$, and

(ii) if $\text{atoms}(x) \subsetneq \text{atoms}(y)$ and $x \in \text{free}(\varphi)$, then $y \in \text{free}(\varphi)$.

Queries that are not q-hierarchical:

$$\psi_{S\text{-}E\text{-}T}() \quad := \quad \exists x \exists y \, \big(\, S(x) \wedge E(x,y) \wedge T(y) \, \big)$$



q-hierarchical CQs

Dalvi & Suciu (2007) introduced the hierarchical CQs to characterise the Boolean CQs that can be answered in PTIME on probabilistic dbs.

q-hierarchical CQs are hierarchical CQs where, additionally, the quantifiers respect the query's hierarchical form.

Definition: A CQ $\varphi(z_1, \ldots, z_k)$ is q-hierarchical if for all variables x, y of φ the following is satisfied:

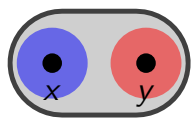
(i) $\text{atoms}(x) \subseteq \text{atoms}(y)$ or $\text{atoms}(y) \subseteq \text{atoms}(x)$ or $\text{atoms}(x) \cap \text{atoms}(y) = \emptyset$, and

(ii) if $\text{atoms}(x) \subsetneq \text{atoms}(y)$ and $x \in \text{free}(\varphi)$, then $y \in \text{free}(\varphi)$.

Queries that are not q-hierarchical:

$$\psi_{S\text{-}E\text{-}T}() \;:=\; \exists x \exists y \,\big(\, S(x) \wedge E(x,y) \wedge T(y) \,\big)$$

$$\varphi_{S\text{-}E\text{-}T}(x,y) \;:=\; S(x) \wedge E(x,y) \wedge T(y)$$



q-hierarchical CQs

Dalvi & Suciu (2007) introduced the hierarchical CQs to characterise the Boolean CQs that can be answered in PTIME on probabilistic dbs.

q-hierarchical CQs are hierarchical CQs where, additionally, the quantifiers respect the query's hierarchical form.

Definition: A CQ $\varphi(z_1, \ldots, z_k)$ is q-hierarchical if for all variables x, y of φ the following is satisfied:

(i) $\text{atoms}(x) \subseteq \text{atoms}(y)$ or $\text{atoms}(y) \subseteq \text{atoms}(x)$ or $\text{atoms}(x) \cap \text{atoms}(y) = \emptyset$, and

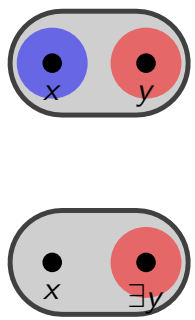
(ii) if $\text{atoms}(x) \subsetneq \text{atoms}(y)$ and $x \in \text{free}(\varphi)$, then $y \in \text{free}(\varphi)$.

Queries that are not q-hierarchical:

$$\psi_{S\text{-}E\text{-}T}() := \exists x \exists y \, (\, S(x) \wedge E(x,y) \wedge T(y) \,)$$

$$\varphi_{S\text{-}E\text{-}T}(x,y) := S(x) \wedge E(x,y) \wedge T(y)$$

$$\varphi_{E\text{-}T}(x) := \exists y \, (\, E(x,y) \wedge T(y) \,)$$



q-hierarchical CQs

Dalvi & Suciu (2007) introduced the hierarchical CQs to characterise the Boolean CQs that can be answered in PTIME on probabilistic dbs.

q-hierarchical CQs are hierarchical CQs where, additionally, the quantifiers respect the query's hierarchical form.

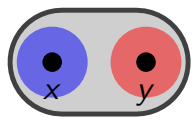
Definition: A CQ $\varphi(z_1, \ldots, z_k)$ is q-hierarchical if for all variables x, y of φ the following is satisfied:

(i) $\text{atoms}(x) \subseteq \text{atoms}(y)$ or $\text{atoms}(y) \subseteq \text{atoms}(x)$ or $\text{atoms}(x) \cap \text{atoms}(y) = \emptyset$, and

(ii) if $\text{atoms}(x) \subsetneq \text{atoms}(y)$ and $x \in \text{free}(\varphi)$, then $y \in \text{free}(\varphi)$.

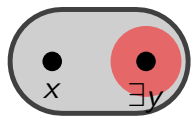
Queries that are not q-hierarchical:

$$\psi_{S\text{-}E\text{-}T}() := \exists x \exists y \, (\, S(x) \wedge E(x,y) \wedge T(y) \,)$$



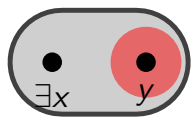
$$\varphi_{S\text{-}E\text{-}T}(x,y) := S(x) \wedge E(x,y) \wedge T(y)$$

$$\varphi_{E\text{-}T}(x) := \exists y \, (\, E(x,y) \wedge T(y) \,)$$



A q-hierarchical query:

$$\theta_{E\text{-}T}(y) := \exists x \, (\, E(x,y) \wedge T(y) \,)$$



Intractability result for enumerating CQs that are not q-hierarchical

. . . is subject to suitable algorithmic conjecture

Intractability result for enumerating CQs that are not q-hierarchical

. . . is subject to suitable algorithmic conjecture

The OMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors

Task: output Mv_ℓ before accessing $v_{\ell+1}$

Intractability result for enumerating CQs that are not q-hierarchical

. . . is subject to suitable algorithmic conjecture

The OMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors

Task: output Mv_ℓ before accessing $v_{\ell+1}$

OMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OMv-problem in total time $O(n^{3-\epsilon})$

Intractability result for enumerating CQs that are not q-hierarchical

. . . is subject to suitable algorithmic conjecture

The OMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors

Task: output Mv_ℓ before accessing $v_{\ell+1}$

OMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OMv-problem in total time $O(n^{3-\epsilon})$

Fails for offline algorithms if we receive all vectors at once: fast matrix multiplication!

Intractability result for enumerating CQs that are not q-hierarchical

. . . is subject to suitable algorithmic conjecture

The OMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors

Task: output Mv_ℓ before accessing $v_{\ell+1}$

OMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OMv-problem in total time $O(n^{3-\epsilon})$

Fails for offline algorithms if we receive all vectors at once: fast matrix multiplication!

Theorem (Enumeration): [Berkholz, Keppeler, S., 2017]

Let $\epsilon > 0$ and let $\varphi(\overline{x})$ be a self-join free CQ that is not q-hierarchical.
Then, there is no algorithm with arbitrary preprocessing time and $t_u = O(n^{1-\epsilon})$ update time that enumerates $\varphi(D)$ with $t_d = O(n^{1-\epsilon})$ delay, unless the OMv-conjecture fails.

Intractability result for enumerating CQs that are not q-hierarchical

. . . is subject to suitable algorithmic conjecture

The OMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors

Task: output Mv_ℓ before accessing $v_{\ell+1}$

OMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OMv-problem in total time $O(n^{3-\epsilon})$

Fails for offline algorithms if we receive all vectors at once: fast matrix multiplication!

Theorem (Enumeration): [Berkholz, Keppeler, S., 2017]

Let $\epsilon > 0$ and let $\varphi(\overline{x})$ be a self-join free CQ that is not q-hierarchical.
Then, there is no algorithm with arbitrary preprocessing time and $t_u = O(n^{1-\epsilon})$ update time that enumerates $\varphi(D)$ with $t_d = O(n^{1-\epsilon})$ delay, unless the OMv-conjecture fails.

Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

A lower bound for enumerating via OMv

Input: Boolean $n \times n$ matrix M and stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors.

Task: output Mv_ℓ before accessing $v_{\ell+1}$

Proof idea for $\varphi_{E\text{-}T}(x) \;:=\; \exists y \,\big(\, E(x,y) \wedge T(y) \,\big)$

A lower bound for enumerating via OMv

Input: Boolean $n \times n$ matrix M and stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors.

Task: output Mv_ℓ before accessing $v_{\ell+1}$

Given $n \times n$ matrix M, let

- $E^{D_0} \;:=\; \{\,(i,j) \in [n]^2 \,:\, M(i,j) = 1\,\}, \quad T^{D_0} \;:=\; \emptyset$

Create data structure for D_0 in time $n^2 \cdot n^{1-\epsilon}$.

Proof idea for $\varphi_{E\text{-}T}(x) \;:=\; \exists y \,\big(\, E(x,y) \wedge T(y) \,\big)$

A lower bound for enumerating via OMv

Input: Boolean $n \times n$ matrix M and stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors.

Task: output Mv_ℓ before accessing $v_{\ell+1}$

Given $n \times n$ matrix M, let

- $E^{D_0} \;:=\; \{\,(i,j) \in [n]^2 \;:\; M(i,j) = 1\,\}, \quad T^{D_0} \;:=\; \emptyset$

Create data structure for D_0 in time $n^2 \cdot n^{1-\epsilon}$.

Given n-dim. vector v_ℓ, update

- $T^{D_\ell} \;:=\; \{\, i \in [n] \;:\; v_\ell(i) = 1\,\}.$

in time $n \cdot n^{1-\varepsilon}$.

Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

A lower bound for enumerating via OMv

Input: Boolean $n \times n$ matrix M and stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors.

Task: output Mv_ℓ before accessing $v_{\ell+1}$

Given $n \times n$ matrix M, let

- $E^{D_0} := \{ (i,j) \in [n]^2 : M(i,j) = 1 \}, \quad T^{D_0} := \emptyset$

Create data structure for D_0 in time $n^2 \cdot n^{1-\epsilon}$.

Given n-dim. vector v_ℓ, update

- $T^{D_\ell} := \{ i \in [n] : v_\ell(i) = 1 \}.$

in time $n \cdot n^{1-\varepsilon}$. For $u_\ell := Mv_\ell$ we have:

- $\varphi_{E\text{-}T}(D_\ell) = \{ i \in [n] : u_\ell(i) = 1 \}$

Proof idea for $\varphi_{E\text{-}T}(x) \;:=\; \exists y \,\big(\, E(x,y) \wedge T(y) \,\big)$

A lower bound for enumerating via OMv

Input: Boolean $n \times n$ matrix M and stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors.

Task: output Mv_ℓ before accessing $v_{\ell+1}$

Given $n \times n$ matrix M, let

- $E^{D_0} \;:=\; \{\,(i,j) \in [n]^2 \,:\, M(i,j) = 1\,\}, \quad T^{D_0} \;:=\; \emptyset$

Create data structure for D_0 in time $n^2 \cdot n^{1-\epsilon}$.

Given n-dim. vector v_ℓ, update

- $T^{D_\ell} \;:=\; \{\, i \in [n] \,:\, v_\ell(i) = 1\,\}.$

in time $n \cdot n^{1-\varepsilon}$. For $u_\ell := Mv_\ell$ we have:

- $\varphi_{E\text{-}T}(D_\ell) \;=\; \{\, i \in [n] \,:\, u_\ell(i) = 1\,\}$

and can output u_ℓ after enumerating $\varphi_{E\text{-}T}(D_\ell)$ in time $n \cdot n^{1-\varepsilon}$.

Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

A lower bound for enumerating via OMv

Input: Boolean $n \times n$ matrix M and stream $v_1, \ldots, v_n$ of n-dimensional Boolean vectors.

Task: output Mv_ℓ before accessing $v_{\ell+1}$

Given $n \times n$ matrix M, let

- $E^{D_0} := \{ (i,j) \in [n]^2 : M(i,j) = 1 \}, \quad T^{D_0} := \emptyset$

Create data structure for D_0 in time $n^2 \cdot n^{1-\epsilon}$.

Given n-dim. vector v_ℓ, update

- $T^{D_\ell} := \{ i \in [n] : v_\ell(i) = 1 \}.$

in time $n \cdot n^{1-\varepsilon}$. For $u_\ell := Mv_\ell$ we have:

- $\varphi_{E\text{-}T}(D_\ell) = \{ i \in [n] : u_\ell(i) = 1 \}$

and can output u_ℓ after enumerating $\varphi_{E\text{-}T}(D_\ell)$ in time $n \cdot n^{1-\varepsilon}$.

This solves OMv in total time $O(n^{3-\varepsilon})$ ↯

Intractability result for Boolean CQs that are not q-hierarchical

The OuMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $u_1, v_1, \ldots, u_n, v_n$ of n-dimensional Boolean vectors

Task: output $(u_\ell)^\intercal M v_\ell$ before accessing $u_{\ell+1}$, $v_{\ell+1}$

OuMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OuMv-problem in total time $O(n^{3-\epsilon})$

Intractability result for Boolean CQs that are not q-hierarchical

The OuMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $u_1, v_1, \ldots, u_n, v_n$ of n-dimensional Boolean vectors

Task: output $(u_\ell)^\intercal M v_\ell$ before accessing $u_{\ell+1}$, $v_{\ell+1}$

OuMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OuMv-problem in total time $O(n^{3-\epsilon})$

Fails for offline algorithms if we receive all vectors at once: fast matrix multiplication!

Intractability result for Boolean CQs that are not q-hierarchical

The OuMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $u_1, v_1, \ldots, u_n, v_n$ of n-dimensional Boolean vectors

Task: output $(u_\ell)^\intercal M v_\ell$ before accessing $u_{\ell+1}$, $v_{\ell+1}$

OuMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OuMv-problem in total time $O(n^{3-\epsilon})$

Fails for offline algorithms if we receive all vectors at once: fast matrix multiplication!

Theorem (Boolean): [Berkholz, Keppeler, S., 2017]

Fix an $\epsilon > 0$ and let φ be a Boolean CQ whose homomorphic core is not q-hierarchical.
Then, there is no algorithm with arbitrary preprocessing time and $t_u = O(n^{1-\epsilon})$ update time that answers $\varphi(D)$ in time $t_a = O(n^{2-\epsilon})$, unless the OuMv-conjecture fails.

Intractability result for Boolean CQs that are not q-hierarchical

The OuMv-problem: [Henzinger et al. 2015]

Input: a Boolean $n \times n$-matrix M and
a stream $u_1, v_1, \ldots, u_n, v_n$ of n-dimensional Boolean vectors

Task: output $(u_\ell)^\intercal M v_\ell$ before accessing $u_{\ell+1}$, $v_{\ell+1}$

OuMv-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OuMv-problem in total time $O(n^{3-\epsilon})$

Fails for offline algorithms if we receive all vectors at once: fast matrix multiplication!

Theorem (Boolean): [Berkholz, Keppeler, S., 2017]

Fix an $\epsilon > 0$ and let φ be a Boolean CQ whose homomorphic core is not q-hierarchical.
Then, there is no algorithm with arbitrary preprocessing time and $t_u = O(n^{1-\epsilon})$ update time that answers $\varphi(D)$ in time $t_a = O(n^{2-\epsilon})$, unless the OuMv-conjecture fails.

Proof idea for $\psi_{S\text{-}E\text{-}T} := \exists x \exists y \left(S(x) \wedge E(x,y) \wedge T(y) \right)$

Intractability result for counting CQs that are not q-hierarchical

The OV-problem: [cf. R. Williams, 2005]

Input: two sets U and V of n Boolean vectors of dimension $d := \lceil \log^2 n \rceil$

Task: decide if there exist $u \in U$ and $v \in V$ with $u^\intercal v = 0$

OV-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OV-problem in time $O(n^{2-\epsilon})$

Intractability result for counting CQs that are not q-hierarchical

The OV-problem: [cf. R. Williams, 2005]

Input: two sets U and V of n Boolean vectors of dimension $d := \lceil \log^2 n \rceil$

Task: decide if there exist $u \in U$ and $v \in V$ with $u^\intercal v = 0$

OV-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OV-problem in time $O(n^{2-\epsilon})$

Theorem (Counting): [Berkholz, Keppeler, S., PODS'17]

Let $\epsilon > 0$ and let $\varphi(\overline{x})$ be a CQ whose homomorphic core is not q-hierarchical.
Then, there is no algorithm with arbitrary preprocessing time and $t_u = O(n^{1-\epsilon})$ update time that computes $|\varphi(D)|$ in time $t_c = O(n^{1-\epsilon})$, unless the OV-conjecture or the OuMv-conjecture fails.

Intractability result for counting CQs that are not q-hierarchical

The OV-problem: [cf. R. Williams, 2005]

Input: two sets U and V of n Boolean vectors of dimension $d := \lceil \log^2 n \rceil$

Task: decide if there exist $u \in U$ and $v \in V$ with $u^\intercal v = 0$

OV-Conjecture: For every $\epsilon > 0$, there is no algorithm that solves the OV-problem in time $O(n^{2-\epsilon})$

Theorem (Counting): [Berkholz, Keppeler, S., PODS'17]

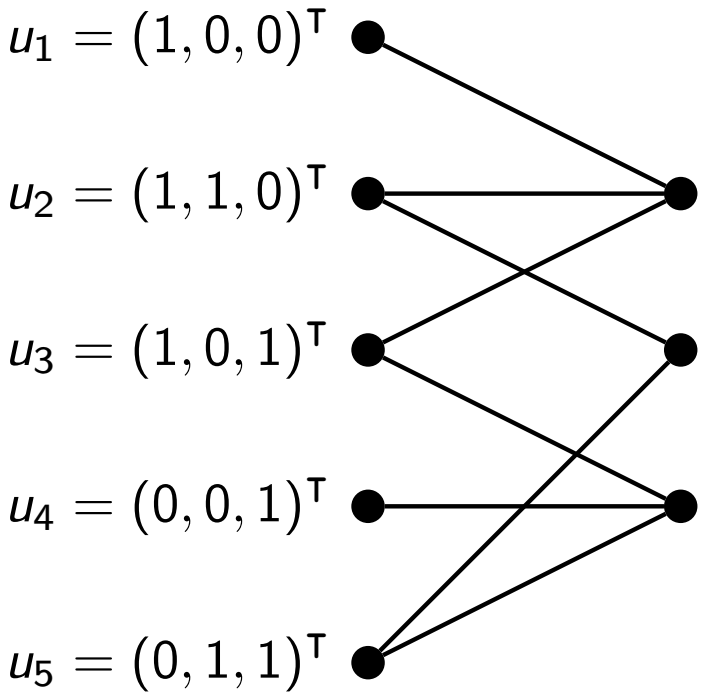
Let $\epsilon > 0$ and let $\varphi(\overline{x})$ be a CQ whose homomorphic core is not q-hierarchical.
Then, there is no algorithm with arbitrary preprocessing time and $t_u = O(n^{1-\epsilon})$ update time that computes $|\varphi(D)|$ in time $t_c = O(n^{1-\epsilon})$, unless the OV-conjecture or the OuMv-conjecture fails.

Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

A lower bound for counting via OV

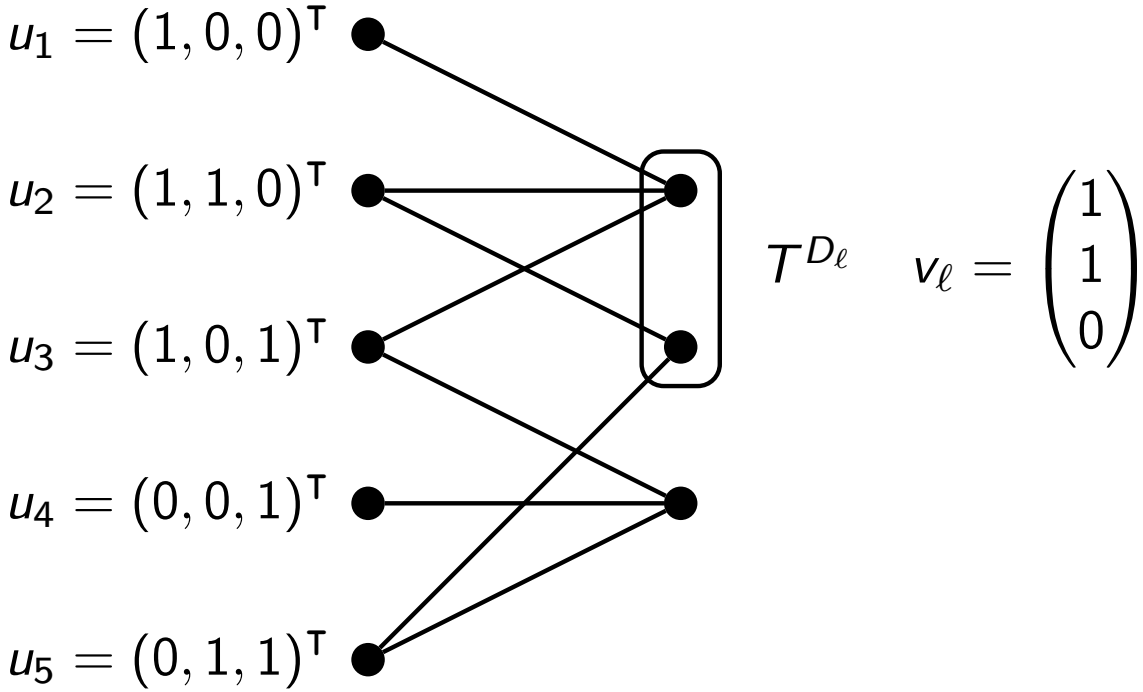
Left: n vertices for the n vectors $u \in U$
Right: $d := \lceil \log^2 n \rceil$ vertices for vector-coordinates



Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

A lower bound for counting via OV

Left: n vertices for the n vectors $u \in U$
Right: $d := \lceil \log^2 n \rceil$ vertices for vector-coordinates

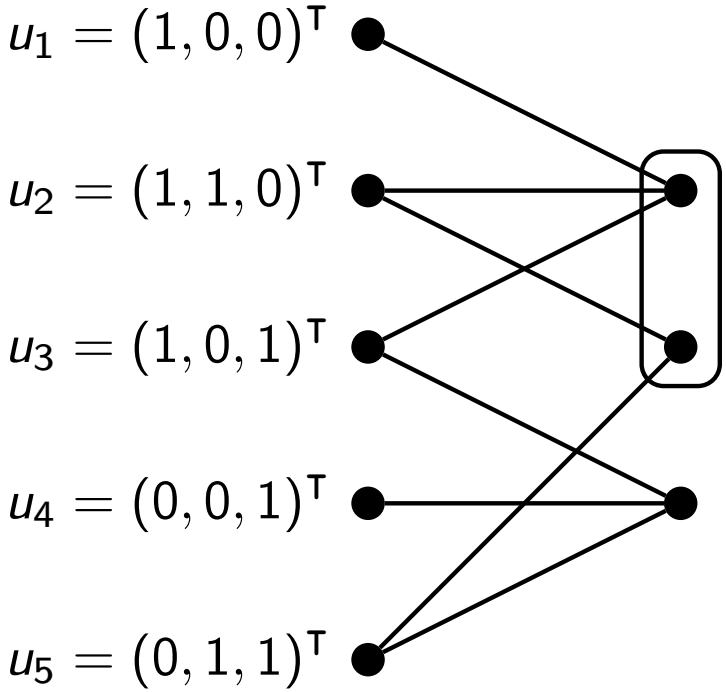


- for each $v_\ell \in V$: update T^{D_ℓ} in time $d \cdot n^{1-\epsilon} = \lceil \log^2 n \rceil n^{1-\varepsilon}$

Proof idea for $\varphi_{E\text{-}T}(x) \;:=\; \exists y \,\big(\, E(x,y) \wedge T(y) \,\big)$

A lower bound for counting via OV

Left: n vertices for the n vectors $u \in U$
Right: $d := \lceil \log^2 n \rceil$ vertices for vector-coordinates



$T^{D_\ell} \quad v_\ell = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$

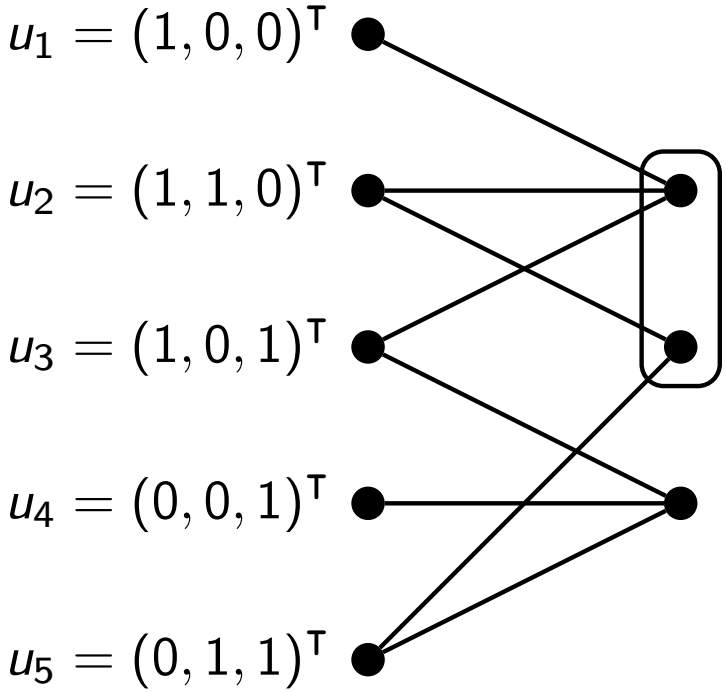
$\big|\varphi_{E\text{-}T}(D_\ell)\big| = 4 = \big|\{\, u_i \;:\; {u_i}^\mathsf{T} v_\ell \neq 0 \,\}\big|$

- for each $v_\ell \in V$: update T^{D_ℓ} in time $d \cdot n^{1-\epsilon} \;=\; \lceil \log^2 n \rceil n^{1-\varepsilon}$
- there is $u_i \in U$ with ${u_i}^\mathsf{T} v_\ell = 0 \iff |\varphi_{E\text{-}T}(D_\ell)| < n.$

Proof idea for $\varphi_{E\text{-}T}(x) := \exists y \left(E(x,y) \wedge T(y) \right)$

A lower bound for counting via OV

Left: n vertices for the n vectors $u \in U$
Right: $d := \lceil \log^2 n \rceil$ vertices for vector-coordinates



$T^{D_\ell} \quad v_\ell = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$

$\left|\varphi_{E\text{-}T}(D_\ell)\right| = 4 = \left|\{\, u_i \;:\; {u_i}^\mathsf{T} v_\ell \neq 0 \,\}\right|$

- for each $v_\ell \in V$: update T^{D_ℓ} in time $d \cdot n^{1-\epsilon} = \lceil \log^2 n \rceil n^{1-\varepsilon}$
- there is $u_i \in U$ with ${u_i}^\mathsf{T} v_\ell = 0 \iff |\varphi_{E\text{-}T}(D_\ell)| < n$.
- finished for all $v_\ell \in V$ within time $n \cdot \lceil \log^2 n \rceil n^{1-\varepsilon} = n^{2-\varepsilon'}$ ↯

Efficient evaluation of a fragment of CQs

Theorem (Upper bound) [Berkholz, Keppeler, S. 2017]:

For every CQ that is q-hierarchical, there is a dynamic data structure that has constant update time and allows to

Efficient evaluation of a fragment of CQs

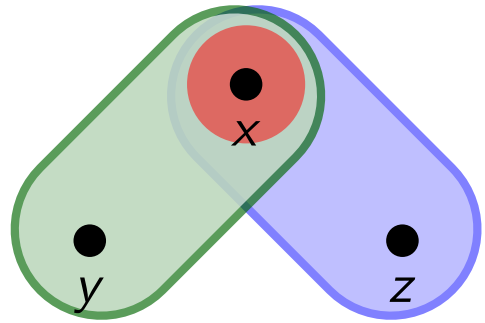
Theorem (Upper bound) [Berkholz, Keppeler, S. 2017]:

For every CQ that is q-hierarchical, there is a dynamic data structure that has constant update time and allows to

- answer a Boolean CQ,
- count the number of result tuples,
- enumerate the result relation with constant delay.

q-hierarchical queries

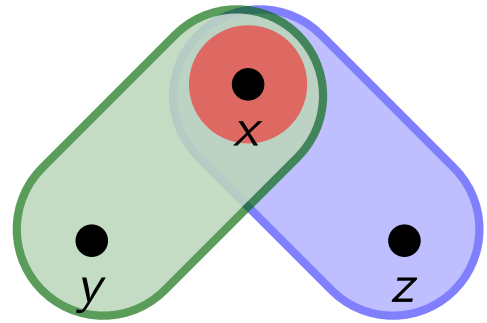
$$\varphi(x, y, z) \;:=\; R(x) \wedge E(x, y) \wedge F(x, z)$$



q-hierarchical queries

$$\varphi(x,y,z) \;:=\; R(x) \wedge E(x,y) \wedge F(x,z)$$

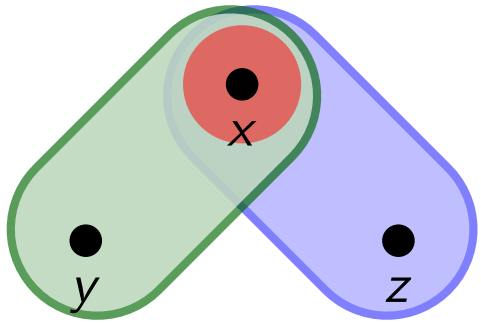
$$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$$



q-hierarchical queries

$\varphi(x, y, z) := R(x) \wedge E(x, y) \wedge F(x, z)$

$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$

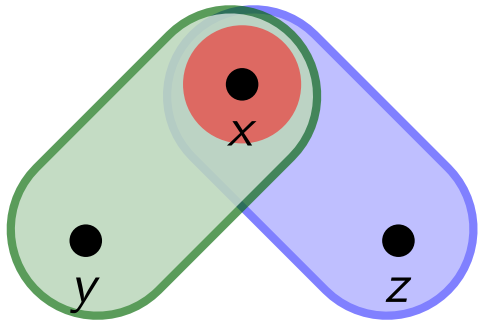


- COUNT: store $|N_E^+(v)|$, $|N_F^+(v)|$, $\sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$

q-hierarchical queries

$\varphi(x, y, z) := R(x) \wedge E(x, y) \wedge F(x, z)$

$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$

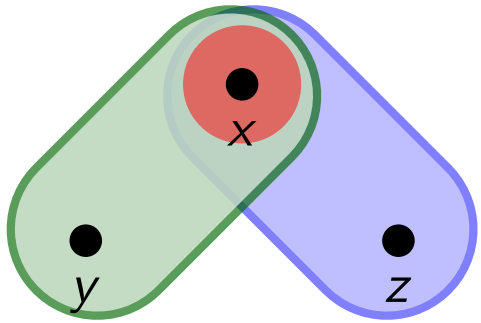


- COUNT: store $|N_E^+(v)|$, $|N_F^+(v)|$, $\sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$
- ENUM: store $N_E^+(v)$, $N_F^+(v)$ as lists with constant access, for $v \in R^D$ report $\{v\} \times N_E^+(v) \times N_F^+(v)$

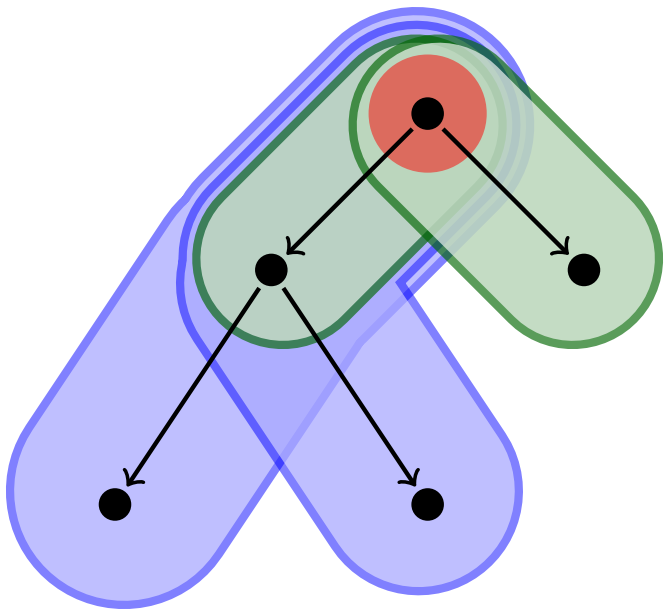
q-hierarchical queries

$\varphi(x, y, z) := R(x) \wedge E(x, y) \wedge F(x, z)$

$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$



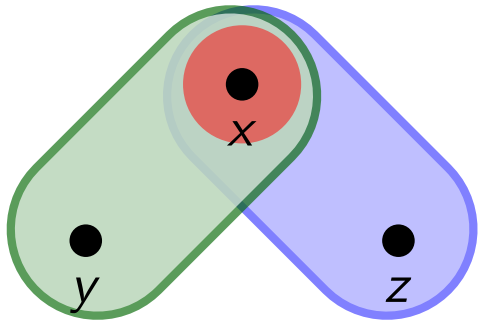
- COUNT: store $|N_E^+(v)|$, $|N_F^+(v)|$, $\sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$
- ENUM: store $N_E^+(v)$, $N_F^+(v)$ as lists with constant access, for $v \in R^D$ report $\{v\} \times N_E^+(v) \times N_F^+(v)$



q-hierarchical queries

$\varphi(x, y, z) := R(x) \wedge E(x, y) \wedge F(x, z)$

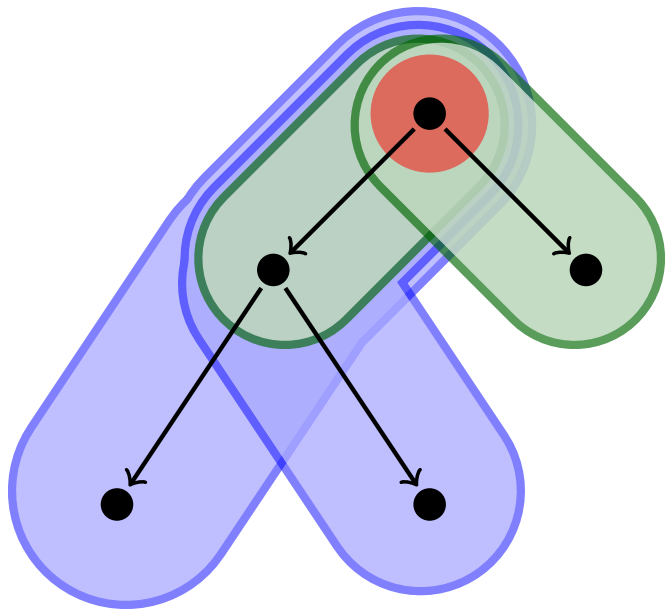
$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$



- COUNT: store $|N_E^+(v)|$, $|N_F^+(v)|$, $\sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$
- ENUM: store $N_E^+(v)$, $N_F^+(v)$ as lists with constant access, for $v \in R^D$ report $\{v\} \times N_E^+(v) \times N_F^+(v)$

Definition (q-tree):

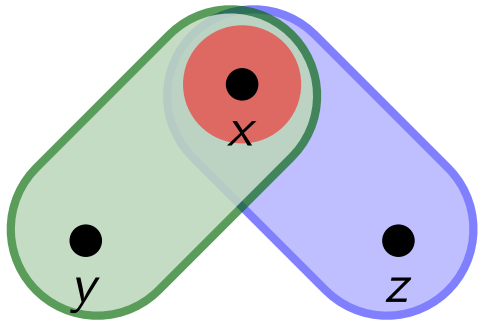
A q-tree T for a CQ $\varphi(x_1, \ldots, x_\ell)$ is a rooted tree with $V(T) = \text{vars}(\varphi)$ and



q-hierarchical queries

$\varphi(x, y, z) := R(x) \wedge E(x, y) \wedge F(x, z)$

$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$

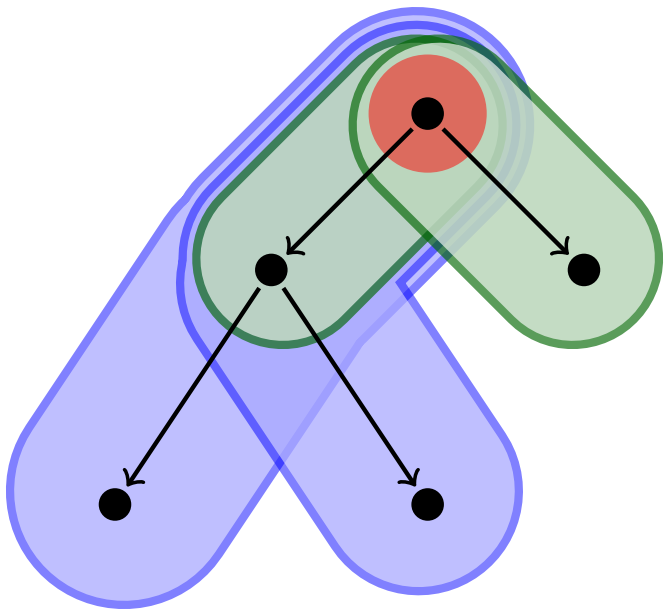


- COUNT: store $|N_E^+(v)|$, $|N_F^+(v)|$, $\sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$
- ENUM: store $N_E^+(v)$, $N_F^+(v)$ as lists with constant access, for $v \in R^D$ report $\{v\} \times N_E^+(v) \times N_F^+(v)$

Definition (q-tree):

A q-tree T for a CQ $\varphi(x_1, \ldots, x_\ell)$ is a rooted tree with $V(T) = \text{vars}(\varphi)$ and

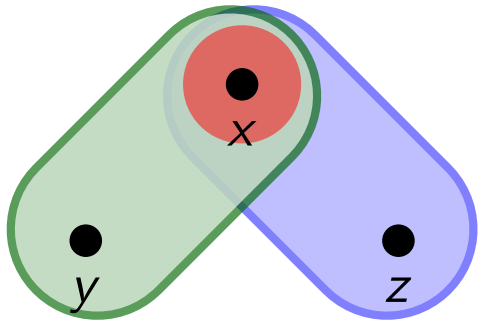
1. for every $R(y_1, \ldots, y_r)$ in φ: $\{y_1, \ldots, y_r\}$ forms a path in T that starts at the root



q-hierarchical queries

$\varphi(x,y,z) \;:=\; R(x) \wedge E(x,y) \wedge F(x,z)$

$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$

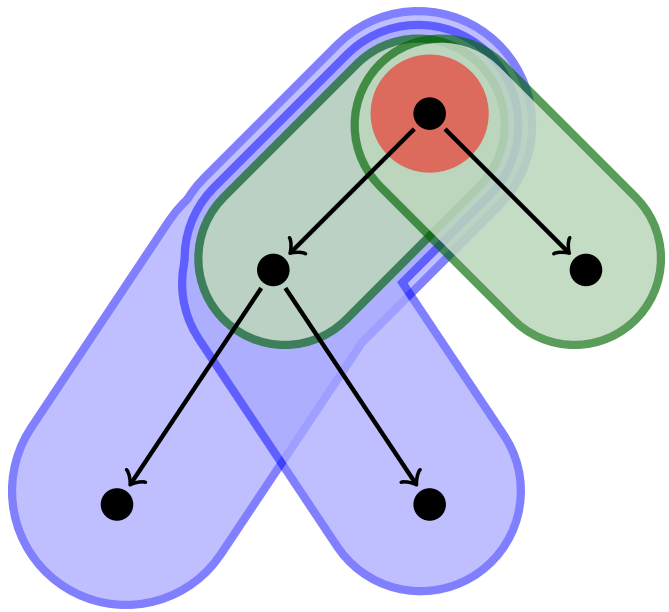


- COUNT: store $|N_E^+(v)|$, $|N_F^+(v)|$, $\sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$
- ENUM: store $N_E^+(v)$, $N_F^+(v)$ as lists with constant access, for $v \in R^D$ report $\{v\} \times N_E^+(v) \times N_F^+(v)$

Definition (q-tree):

A q-tree T for a CQ $\varphi(x_1, \ldots, x_\ell)$ is a rooted tree with $V(T) = \text{vars}(\varphi)$ and

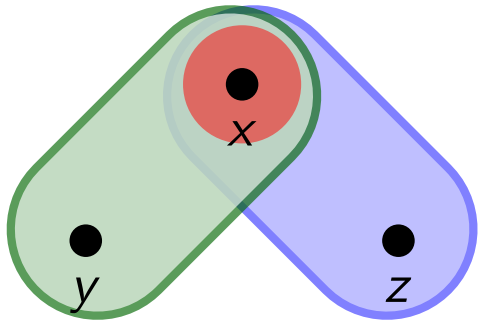
1. for every $R(y_1, \ldots, y_r)$ in φ: $\{y_1, \ldots, y_r\}$ forms a path in T that starts at the root
2. the free variables $\{x_1, \ldots, x_\ell\}$ form a connected subtree that contains the root



q-hierarchical queries

$\varphi(x,y,z) := R(x) \wedge E(x,y) \wedge F(x,z)$

$|\varphi(D)| = \sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$

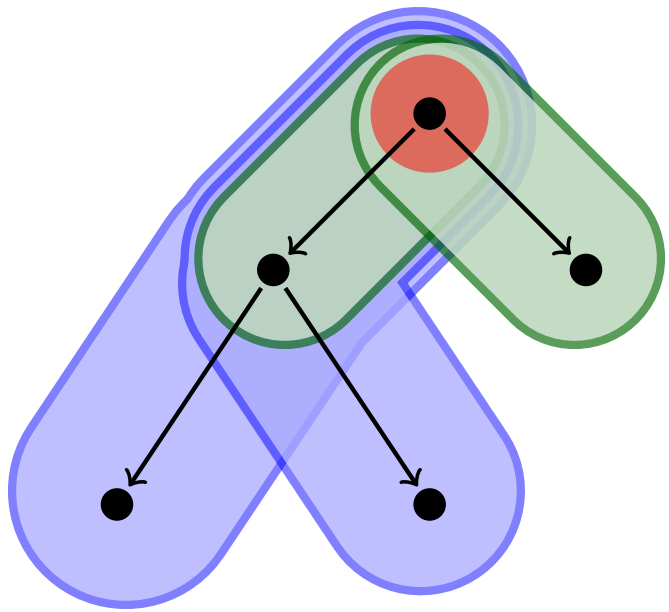


- COUNT: store $|N_E^+(v)|$, $|N_F^+(v)|$, $\sum_{v \in R^D} |N_E^+(v)| \cdot |N_F^+(v)|$
- ENUM: store $N_E^+(v)$, $N_F^+(v)$ as lists with constant access, for $v \in R^D$ report $\{v\} \times N_E^+(v) \times N_F^+(v)$

Definition (q-tree):

A q-tree T for a CQ $\varphi(x_1, \ldots, x_\ell)$ is a rooted tree with $V(T) = \text{vars}(\varphi)$ and

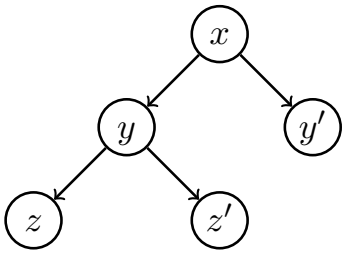
1. for every $R(y_1, \ldots, y_r)$ in φ: $\{y_1, \ldots, y_r\}$ forms a path in T that starts at the root
2. the free variables $\{x_1, \ldots, x_\ell\}$ form a connected subtree that contains the root



Lemma: A CQ $\varphi(\overline{x})$ is q-hierarchical $\iff$ every connected component of $\varphi(\overline{x})$ has a q-tree.

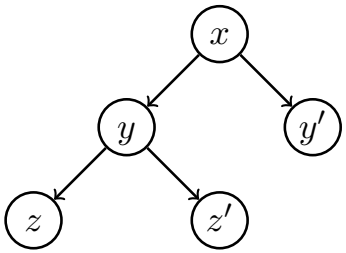
Data structure for q-hierarchical queries

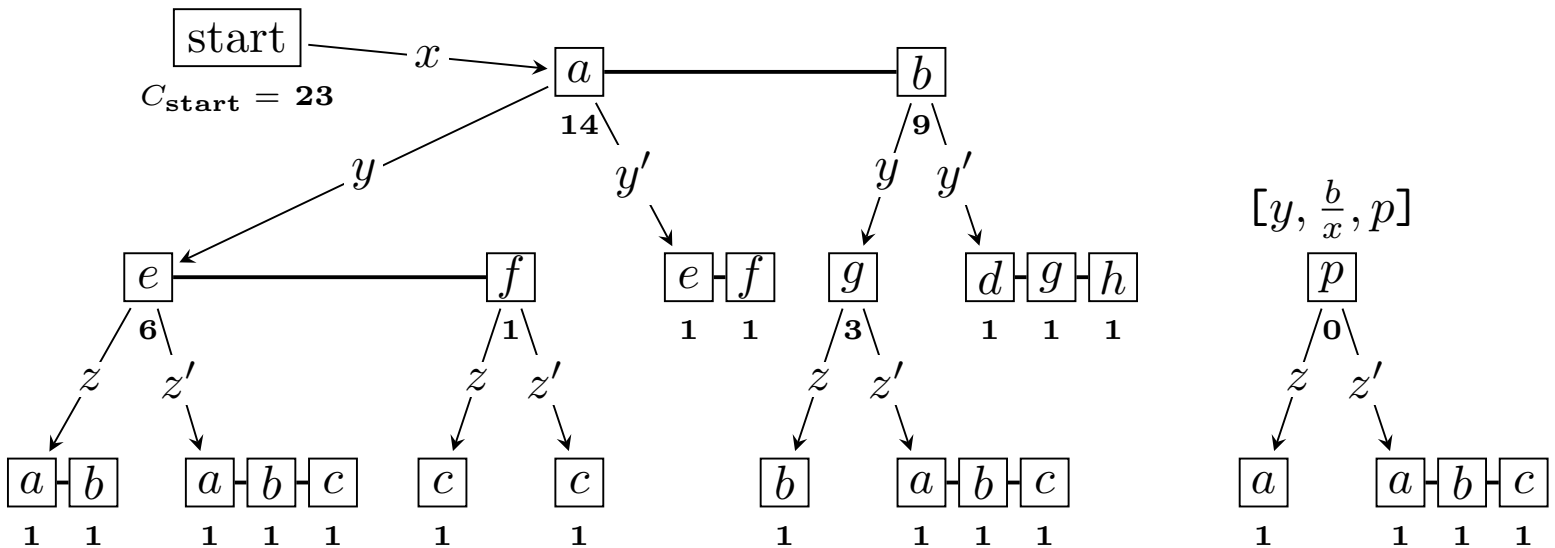
$$\varphi(x, y, z, y', z') = \big(Rxyz \wedge Rxyz' \wedge Exy \wedge Exy' \wedge Sxyz \big)$$



Data structure for q-hierarchical queries

$$\varphi(x, y, z, y', z') = \big(Rxyz \wedge Rxyz' \wedge Exy \wedge Exy' \wedge Sxyz \big)$$

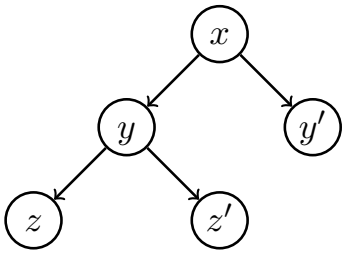


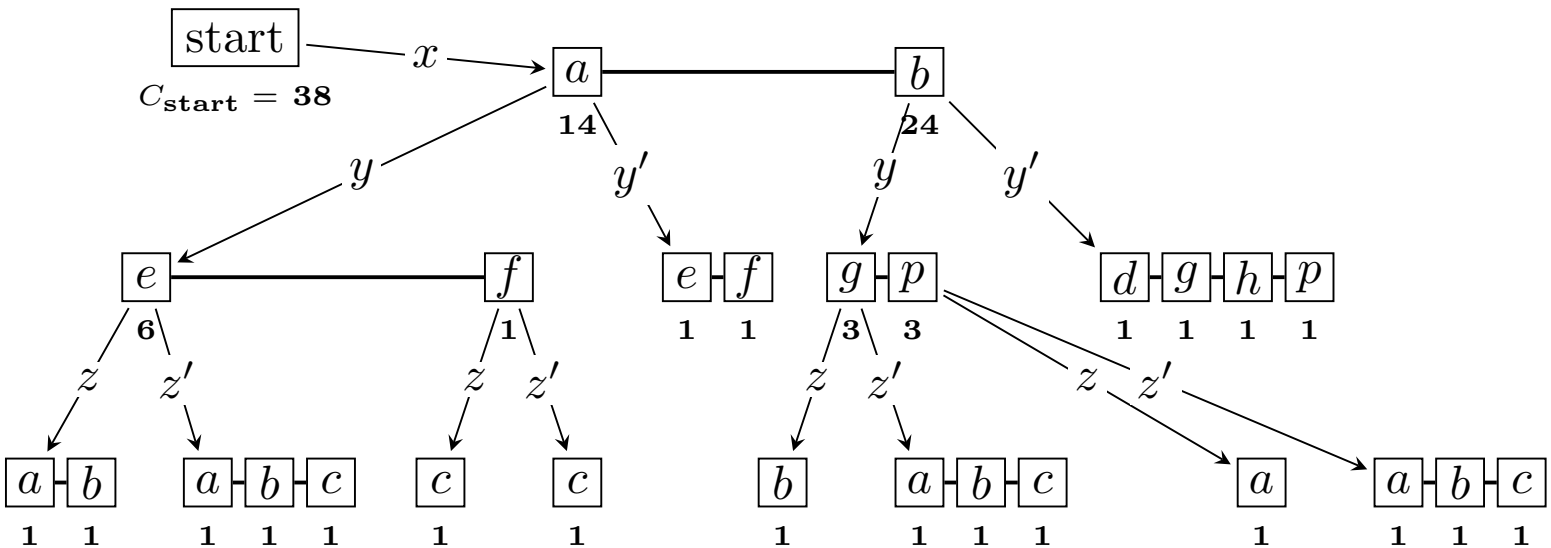


- $S(b, p, a), R(b, p, a), R(b, p, b), R(b, p, c) \in D,\ E(b, p) \notin D$

Data structure for q-hierarchical queries

$$\varphi(x, y, z, y', z') = \big(Rxyz \wedge Rxyz' \wedge Exy \wedge Exy' \wedge Sxyz \big)$$





- $S(b, p, a), R(b, p, a), R(b, p, b), R(b, p, c) \in D,\ E(b, p) \notin D$
- INSERT $E(b, p)$

Overview

Positive Results for FO-Queries (static setting: without updates)

Constant delay enumeration after pseudo-linear time preprocessing is possible for all FO-queries φ on db-class C, if

- C has bounded degree [Durand, Grandjean 2007]
- C has bounded tree-width [Bagan 2006]
- C has bounded expansion [Kazana, Segoufin 2013]
- C has locally bounded expansion [Segoufin, Vigny 2017]
- C is nowhere dense [S., Segoufin, Vigny 2018]

  For any subgraph-closed class C that is not nowhere dense, evaluation of Boolean FO-queries is not possible in time $f(\|\varphi\|)\|D\|^{O(1)}$ (assuming FPT $\neq$ W[1]) [Kreutzer 2011, Dvořák, Král, Thomas 2010]

- C has low degree [Durand, S., Segoufin 2014]

  i.e., for each $\delta > 0$, all sufficiently large D in C have degree $\leqslant \|D\|^{\delta}$.

  Some such C are not nowhere dense (and not subgraph-closed).

Open: To which extensions of FO can this be generalised?
Which results can be extended to the dynamic setting?

Want to learn more about . . .

- enumeration for CQs ?

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ?

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ?

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ? Check recent work on
  - regular document spanners:

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ? Check recent work on
  - regular document spanners: Florenzano et al. PODS'18,

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ? Check recent work on
  - regular document spanners: Florenzano et al. PODS'18, Amarilli et al. ICDT'19,

Want to learn more about ...

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ? Check recent work on
  - regular document spanners: Florenzano et al. PODS'18, Amarilli et al. ICDT'19, Schmidt & S. PODS'21&'22

Want to learn more about ...

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ? Check recent work on
  - regular document spanners: Florenzano et al. PODS'18, Amarilli et al. ICDT'19, Schmidt & S. PODS'21&'22
  - generalisations:

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ? Check recent work on
  - regular document spanners: Florenzano et al. PODS'18, Amarilli et al. ICDT'19, Schmidt & S. PODS'21&'22
  - generalisations: Amarilli et al. PODS'22, Munoz & Riveros ICDT'22

Want to learn more about . . .

- enumeration for CQs ?
  - Christoph Berkholz, Fabian Gerhardt, Nicole Schweikardt: Constant delay enumeration for conjunctive queries: a tutorial. ACM SIGLOG News 7(1): 4-33 (2020)
- the dynamic setting ? Check work by
  - Idris, Ugarte, Vansummeren on the dynamic Yannakakis algorithm (SIGMOD'17)
  - Dan Olteanu, Ahmet Kara et al. on factorised databases
  - Antoine Amarilli, Pierre Bourhis, Stefan Mengel, Matthias Niewerth: Enumeration on Trees with Tractable Combined Complexity and Efficient Updates. PODS 2019: 89-103
- enumeration for other data models ? Check recent work on
  - regular document spanners: Florenzano et al. PODS'18, Amarilli et al. ICDT'19, Schmidt & S. PODS'21&'22
  - generalisations: Amarilli et al. PODS'22, Munoz & Riveros ICDT'22

– Thank you for your attention! –