Constant delay enumeration for FO queries over databases with local bounded expansion

Luc Segoufin, **Alexandre Vigny**

INRIA Saclay, ENS Paris-Saclay, University Paris Diderot

EDBT/ICDT Conference, Venice, March 21, 2017

Outline

Introduction, Query evaluation

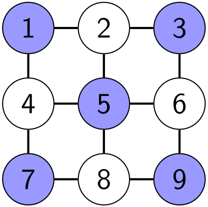
- Query q *small*
- Database D *huge*
- Compute $q(D)$ *gigantic*

Examples :

query q

$$q(x,y) := \exists z(B(x) \wedge E(x,z) \wedge \neg E(y,z))$$

database D



solutions $q(D)$

$\{(1,2)\ (1,3)\ (1,4)$
$(1,6)\ (1,7)\ \cdots$
$(3,1)\ (3,2)\ (3,4)$
$(3,6)\ (3,7)\ \cdots$
$\cdots\ \}$

Enumeration and properties

Input :

- Database : **D**
- Query : $q(\overline{x})$

Enumeration

- Compute an index to speed up the enumeration
- Compute the set of solutions with minimal *delay* between to output

  Ideally : Preprocessing: $O(\|D\|)$ delay: $O(1)$

We talk about *Constant delay enumeration after linear preprocessing*

Properties

- First solution computed in time $O(\|D\|)$
- Overall computation in time $O(\|D\| + |q(D)|)$
- Regularity in the computation

Example

Input :

- Database $D := \langle \{1, \cdots, n\}; E \rangle$ $\qquad \|D\| = |E| \quad (E \subseteq D \times D)$
- Query $q(x, y) := \neg E(x, y)$

D

$$\begin{array}{c} (1,1) \\ (1,2) \\ \boxed{\begin{array}{c} (1,6) \\ \vdots \\ (2,3) \end{array}} \\ \vdots \\ (i,j) \\ (i,j+1) \\ (i,j+3) \\ \vdots \\ (n,n) \end{array}$$

Example

Input :

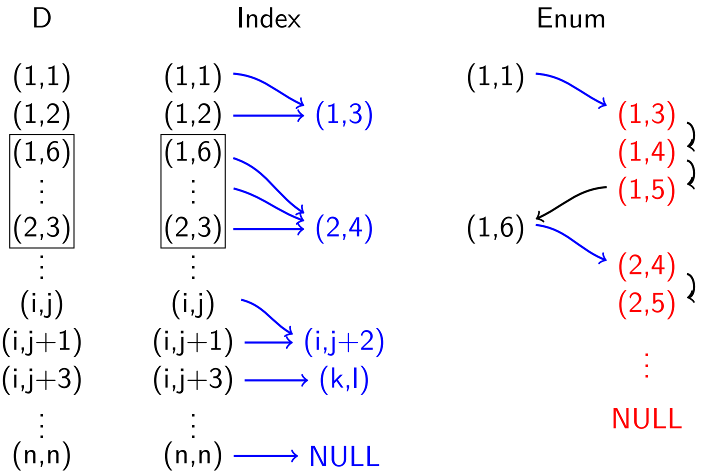
- Database $D := \langle \{1, \cdots, n\}; E \rangle \qquad \|D\| = |E| \quad (E \subseteq D \times D)$
- Query $q(x, y) := \neg E(x, y)$

D	Index	
(1,1)	(1,1) $\rightarrow$	(1,3)
(1,2)	(1,2) $\rightarrow$	(1,3)
(1,6)	(1,6) $\rightarrow$	(2,4)
$\vdots$	$\vdots$ $\rightarrow$	(2,4)
(2,3)	(2,3) $\rightarrow$	(2,4)
$\vdots$	$\vdots$	
(i,j)	(i,j) $\rightarrow$	(i,j+2)
(i,j+1)	(i,j+1) $\rightarrow$	(i,j+2)
(i,j+3)	(i,j+3) $\rightarrow$	(k,l)
$\vdots$	$\vdots$	
(n,n)	(n,n) $\rightarrow$	NULL

Example

Input :

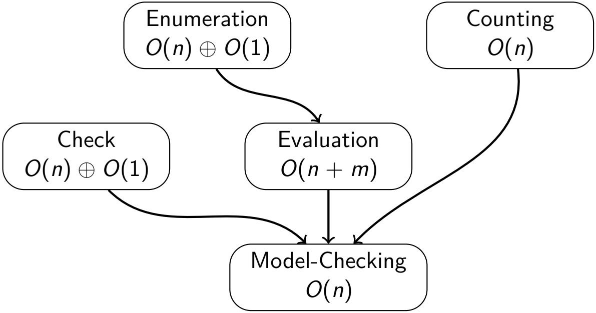
- Database $D := \langle \{1, \cdots, n\}; E \rangle \qquad \|D\| = |E| \quad (E \subseteq D \times D)$
- Query $q(x, y) := \neg E(x, y)$



Other problems

For **FO** queries over a class $\mathcal{C}$ of databases.

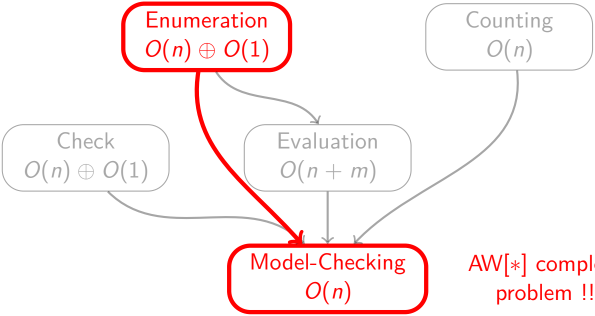
Model-Checking	:	Is there a solution ?	$O(n)$
Counting	:	How many solutions ?	$O(n)$
Check	:	Is this tuple a solution ?	$O(n) \oplus O(1)$
Evaluation	:	Compute the entire set	$O(n+m)$



Other problems

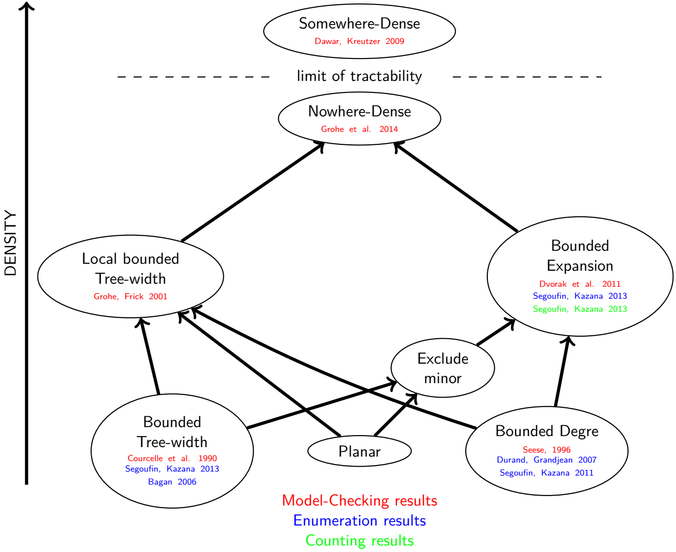
For **FO** queries over a class $\mathcal{C}$ of databases.

Model-Checking	:	Is there a solution ?	$O(n)$
Counting	:	How many solutions ?	$O(n)$
Check	:	Is this tuple a solution ?	$O(n) \oplus O(1)$
Evaluation	:	Compute the entire set	$O(n+m)$

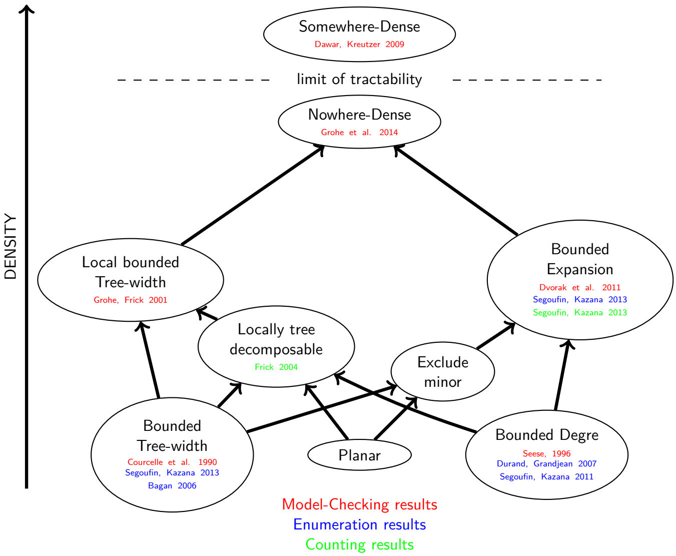


AW[$*$] complete problem !!

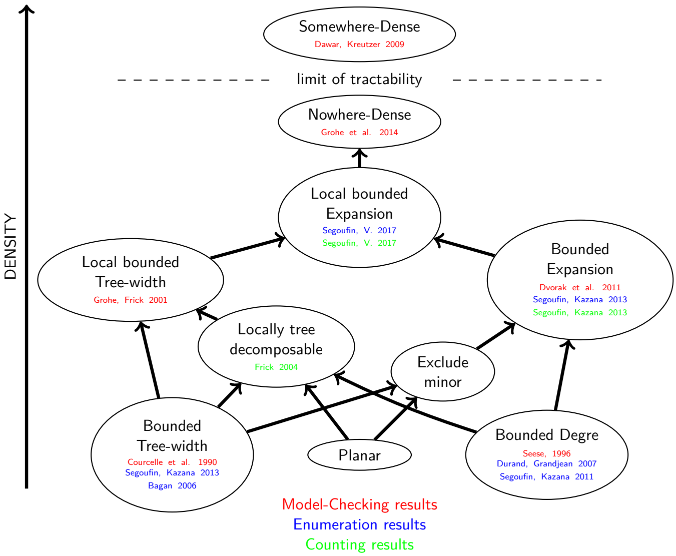
Hereditary classes of graphs and FO queries



Hereditary classes of graphs and FO queries



Hereditary classes of graphs and FO queries



Definition bounded expansion

Introduced by Nešetřil an Ossona de Mendez in '08

Definition: Bounded expansion

$\mathcal{C}$ has bounded expansion if:

$$\exists f : \mathbb{N} \mapsto \mathbb{R} \quad \forall G \in \mathcal{C} \quad \forall r \in \mathbb{N} \quad \nabla_r(G) \leq f(r)$$

Examples:

- Graphs with bounded degree
- Planar graphs
- Graphs with bounded tree width

Properties:

Bounded in-degree, linear number of edges, ...

Definitions local bounded expansion

Definition : Class of r-neighbourhoods

Let $\mathcal{C}$ be a class of graphs, $r \in \mathbb{N}$. $\mathcal{C}_r := \{N_r^G(a) \mid G \in \mathcal{C},\ a \in G\}$

Definition : Local bounded expansion

$\mathcal{C}$ has locally bounded expansion if and only if for all integer r, $\mathcal{C}_r$ has bounded expansion.

Definitions local bounded expansion

Definition : Class of r-neighbourhoods

Let $\mathcal{C}$ be a class of graphs, $r \in \mathbb{N}$. $\mathcal{C}_r := \{N_r^G(a) \mid G \in \mathcal{C},\ a \in G\}$

Definition : Local bounded expansion

$\mathcal{C}$ has locally bounded expansion if and only if for all integer r, $\mathcal{C}_r$ has bounded expansion.

Differences

The number of edges can be non-linear

Our results

Theorem (Segoufin, V. '17)

Over classes of graphs with local bounded expansion, every ***FO*** *queries can be enumerate with constant delay after a pseudo-linear preprocessing.*

Theorem (Segoufin, V. '17)

Over classes of graphs with local bounded expansion, the counting problem for ***FO*** *queries is pseudo-linear.*

Theorem (Segoufin, V. '17)

Over classes of graphs with local bounded expansion, every ***FO*** *queries can be tested in constant time after a pseudo-linear preprocessing.*

Pseudo-linear ?

A function f is pseudo linear if and only if:

$$\forall \epsilon > 0,\ \exists N_\epsilon \in \mathbb{N},\ \forall n \in \mathbb{N},\ n > N_\epsilon \implies f(n) \leq n^{1+\epsilon}$$

$$n \ll n\log^i(n) \ll \text{pseudo-linear} \ll n^{1,0001} \ll n\sqrt{n}$$

"Pseudo linear $\approx n\log^i(n)$"

"Pseudo constant $\approx \log^i(n)$"

Tools

- Gaifman theorem.
- Neighbourhood cover.[1]
- Enumeration for graphs with **Bounded expansion**.[2]
- Short-cut pointers dedicated to the enumeration.

[1] Grohe et al. STOC '14
[2] Kazana, Segoufin PODS '13

Sketch of proof (1/4) : Gaifman theorem

Theorem (Gaifman)

Every first order query is a combination of sentences and local queries.

$$q(x,y) := q_1(x) \wedge q_2(y) \wedge \mathrm{dist}(x,y) > 2r$$

Where q_1 and q_2 are r-local queries.

$$G \models q_1(a) \Longleftrightarrow N_r^G(a) \models q_1(a)$$

$$\rightsquigarrow q(x,y) := Red(x) \wedge Blue(y) \wedge \mathrm{dist}(x,y) > r$$

Sketch of proof (2/4) : Neighbourhood cover

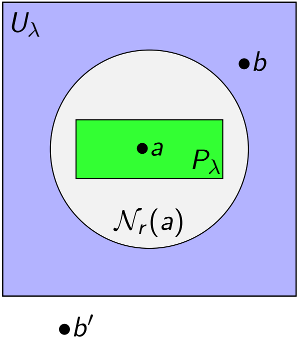
A neighbourhood cover is a set of "representative" neighbourhood.
$T := U_1, \ldots, U_\omega$ with the following properties:

- $\forall a \in G, \quad \exists U_\lambda \in T, \quad N_r(a) \subseteq U_\lambda$
- $\forall U_\lambda \in T, \quad \exists a \in G, \quad U_\lambda \subseteq N_{2r}(a)$ *(The bags have bounded expansion !)*

$\vdots$

Sketch of proof (3/4) : Using bounded expansion

$P_\lambda := \{a \in G \mid N_r(a) \subseteq U_\lambda\}$



2 kind of solutions

close enough	far enough
exemple : (a,b)	*exemple : (a,b')*
$x \in P_\lambda \wedge y \in U_\lambda$	$x \in P_\lambda \wedge y \notin U_\lambda$
Studied locally (within U_λ)	No need to check the distance

Sketch of proof (4/4) : The short-cut pointers

Let $a \in Red(G)$, let λ such that $a \in P_\lambda$ we want to enumerate :

- $b \in Blue(G) \cap U_\lambda$ Easy using the enumeration procedure within U_λ.
- $b \in Blue(G) \setminus U_\lambda$ Here we need something else.

Sketch of proof (4/4) : The short-cut pointers

Let $a \in Red(G)$, let λ such that $a \in P_\lambda$ we want to enumerate :

- $b \in Blue(G) \cap U_\lambda$ Easy using the enumeration procedure within U_λ.
- $b \in Blue(G) \setminus U_\lambda$ Here we need something else.

$$\underset{b \in U_\lambda}{NEXT(b, \lambda)} := \min\{b' \in Blue(G) \mid b' \geq b \ \wedge \ b' \notin U_\lambda\}$$

Sketch of proof (4/4) : The short-cut pointers

Let $a \in Red(G)$, let λ such that $a \in P_\lambda$ we want to enumerate :

- $b \in Blue(G) \cap U_\lambda$ Easy using the enumeration procedure within U_λ.
- $b \in Blue(G) \setminus U_\lambda$ Here we need something else.

$$\underset{b \in U_\lambda}{NEXT}(b, \lambda) := \min\{b' \in Blue(G) \mid b' \geq b \ \wedge \ b' \notin U_\lambda\}$$

For all λ with $b_{max} \in U_\lambda$, we have $NEXT(b_{max}, \lambda) \ = \ NULL$

Sketch of proof (4/4) : The short-cut pointers

Let $a \in Red(G)$, let λ such that $a \in P_\lambda$ we want to enumerate :

- $b \in Blue(G) \cap U_\lambda$ Easy using the enumeration procedure within U_λ.
- $b \in Blue(G) \setminus U_\lambda$ Here we need something else.

$$\underset{b \in U_\lambda}{NEXT(b, \lambda)} := \min\{b' \in Blue(G) \mid b' \geq b \ \wedge \ b' \notin U_\lambda\}$$

For all λ with $b_{max} \in U_\lambda$, we have $NEXT(b_{max}, \lambda) \ = \ NULL$

$$NEXT(b_i, \lambda) \in \{b_{i+1} \ , \ NEXT_\lambda(b_{i+1})\}$$

Sketch of proof (4/4) : The short-cut pointers

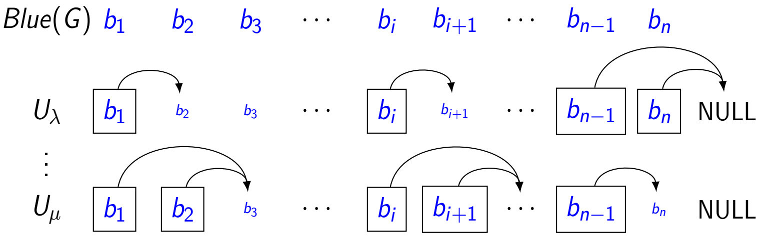
Let $a \in Red(G)$, let λ such that $a \in P_\lambda$ we want to enumerate :

- $b \in Blue(G) \cap U_\lambda$ Easy using the enumeration procedure within U_λ.
- $b \in Blue(G) \setminus U_\lambda$ Here we need something else.

$$\underset{b \in U_\lambda}{NEXT}(b, \lambda) := \min\{b' \in Blue(G) \mid b' \geq b \ \wedge \ b' \notin U_\lambda\}$$

For all λ with $b_{max} \in U_\lambda$, we have $NEXT(b_{max}, \lambda) \ = \ NULL$

$$NEXT(b_i, \lambda) \in \{b_{i+1} \ , \ NEXT_\lambda(b_{i+1})\}$$



Sketch of proof conclusion

By carefully mixing the two algorithms (bounded expansion / short-cut pointers) we can enumerate FO queries.

Moreover:

- We have lexicographical enumeration.
- Given a tuple $\overline{a}$ we can compute in $O(1)$ the smallest solution $\overline{b} \geq \overline{a}$.
- We can test in time $O(1)$ if a tuple is a solution.

Future work

- Generalize to the Nowhere-Dense case.
- Enumeration with update.
  What happens if a small change occurs after the preprocessing ?

Thank you !

Any Question ?