Constant delay enumeration for FO queries and nowhere dense graphs

Alexandre Vigny[1]

Join work with: Nicole Schweikardt[2] and Luc Segoufin[3]

[1]Université Paris Diderot, Paris

[2]Humboldt-Universität zu Berlin

[3]ENS Ulm, Paris

June 1, 2018

Modelization

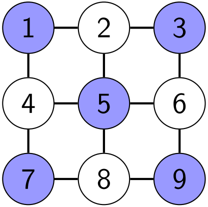
- Query q *small*
- Database D *huge*
- Compute $q(D)$ *gigantic*

Examples :

query *q*
first order logic

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

database *D*
relational structure



solutions *q(D)*
set of tuples

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

Too many solutions!

Database: A given store that contains 50 items for less than 1€

Query: What can I buy with 10€ ?

- For practical reasons:

  50^{10} solutions is not easy to store / display !

- For theoretical reasons:

  The time needed to compute the answer does not reflect the hardness of the problem !

Enumeration

Input : $\|D\| := n$ & $|q| := k$ (computation with RAM)

Goal : output solutions one by one (no repetition)

- STEP 1: Preprocessing

Prepare the enumeration : Database $D \longrightarrow$ Index I

Preprocessing time : $f(k) \cdot n \rightsquigarrow O(n)$

- STEP 2 : Enumeration

Enumerate the solutions : Index $I \longrightarrow \overline{x_1}, \overline{x_2}, \overline{x_3}, \overline{x_4}, \cdots$

Delay : $O(f(k)) \rightsquigarrow O(1)$

Constant delay enumeration after linear preprocessing

Example 1

Input :

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

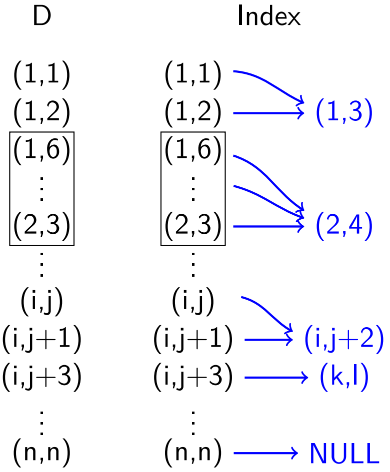
D

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

Example 1

Input :

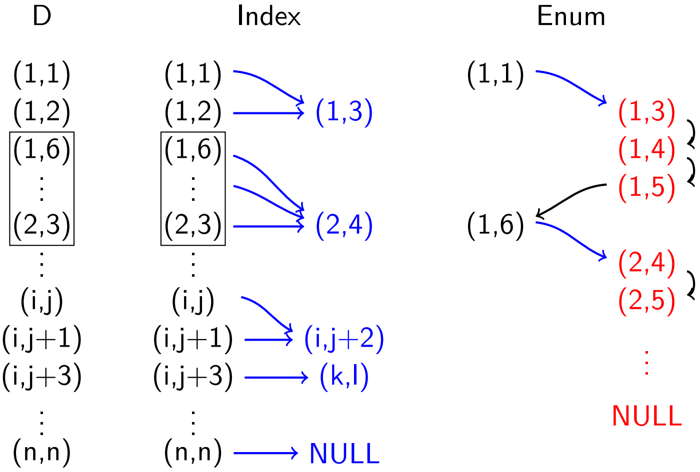
- Database $D := \langle \{1, \cdots, n\}; E \rangle$ $\quad \|D\| = |E| \quad (E \subseteq D \times D)$
- Query $q(x,y) := \neg E(x,y)$



Example 1

Input :

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



Example 2

Input :

- Database $D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle \quad \|D\| = |E_1| + |E_2| \quad (E_i \subseteq D \times D)$
- Query $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

Example 2

Input :

- Database $D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle \quad \|D\| = |E_1| + |E_2| \quad (E_i \subseteq D \times D)$
- Query $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

B : Adjacency matrix of E_2

$$\begin{pmatrix} E_2(1,1) & \dots & E_2(1,y) & \dots & E_2(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_2(z,1) & \dots & E_2(z,y) & \dots & E_2(z,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_2(n,1) & \dots & E_2(n,y) & \dots & E_2(n,n) \end{pmatrix}$$

$$\begin{pmatrix} E_1(1,1) & \dots & E_1(1,i) & \dots & E_1(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(x,1) & \dots & E_1(x,z) & \dots & E_1(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(n,1) & \dots & E_1(n,z) & \dots & E_1(n,n) \end{pmatrix} \quad \begin{pmatrix} q(1,1) & \dots & q(1,y) & \dots & q(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(x,1) & \dots & q(x,y) & \dots & q(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(n,1) & \dots & q(n,y) & \dots & q(n,n) \end{pmatrix}$$

A : Adjacency matrix of E_1

C : Result matrix

Example 2

Input :

- Database $D := \langle\{1,\cdots,n\}; E_1; E_2\rangle$ $\quad \|D\| = |E_1| + |E_2| \quad (E_i \subseteq D \times D)$
- Query $q(x,y) := \exists z,\ E_1(x,z) \wedge E_2(z,y)$

B : Adjacency matrix of E_2

$$\begin{pmatrix} E_2(1,1) & \dots & E_2(1,y) & \dots & E_2(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_2(z,1) & \dots & E_2(z,y) & \dots & E_2(z,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_2(n,1) & \dots & E_2(n,y) & \dots & E_2(n,n) \end{pmatrix}$$

$$\begin{pmatrix} E_1(1,1) & \dots & E_1(1,i) & \dots & E_1(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(x,1) & \dots & E_1(x,z) & \dots & E_1(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(n,1) & \dots & E_1(n,z) & \dots & E_1(n,n) \end{pmatrix} \begin{pmatrix} q(1,1) & \dots & q(1,y) & \dots & q(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(x,1) & \dots & q(x,y) & \dots & q(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(n,1) & \dots & q(n,y) & \dots & q(n,n) \end{pmatrix}$$

A : Adjacency matrix of E_1

C : Result matrix

Compute the set of solutions

=

boolean matrix multiplication

Example 2

Input :

- Database $D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle \quad \|D\| = |E_1| + |E_2| \quad (E_i \subseteq D \times D)$
- Query $q(x, y) := \exists z,\ E_1(x, z) \wedge E_2(z, y)$

B : Adjacency matrix of E_2

$$\begin{pmatrix} E_2(1,1) & \dots & E_2(1,y) & \dots & E_2(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_2(z,1) & \dots & E_2(z,y) & \dots & E_2(z,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_2(n,1) & \dots & E_2(n,y) & \dots & E_2(n,n) \end{pmatrix}$$

$$\begin{pmatrix} E_1(1,1) & \dots & E_1(1,i) & \dots & E_1(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(x,1) & \dots & E_1(x,z) & \dots & E_1(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ E_1(n,1) & \dots & E_1(n,z) & \dots & E_1(n,n) \end{pmatrix} \begin{pmatrix} q(1,1) & \dots & q(1,y) & \dots & q(1,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(x,1) & \dots & q(x,y) & \dots & q(x,n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ q(n,1) & \dots & q(n,y) & \dots & q(n,n) \end{pmatrix}$$

A : Adjacency matrix of E_1

C : Result matrix

- Linear preprocessing: $O(n^2)$
- Number of solutions: $O(n^2)$
- Algorithm for the boolean matrix multiplication in $O(n^2)$
- Conjecture :
  "There are no algorithm for the boolean matrix multiplication working in time $O(n^2)$."

Example 2

Input :

- Database $D := \langle\{1, \cdots, n\}; E_1; E_2\rangle \quad \|D\| = |E_1| + |E_2| \quad (E_i \subseteq D \times D)$
- Query $q(x, y) := \exists z,\ E_1(x, z) \wedge E_2(z, y)$

This query cannot be enumerated with constant delay[1]

[1] Unless there is a breakthrough with the boolean matrix multiplication.

Example 2

Input :

- Database $D := \langle \{1, \cdots, n\}; E_1; E_2 \rangle \quad \|D\| = |E_1| + |E_2| \quad (E_i \subseteq D \times D)$
- Query $q(x, y) := \exists z,\ E_1(x, z) \wedge E_2(z, y)$

This query cannot be enumerated with constant delay[1]

We need to put restrictions on queries and/or databases.

[1] Unless there is a breakthrough with the boolean matrix multiplication.

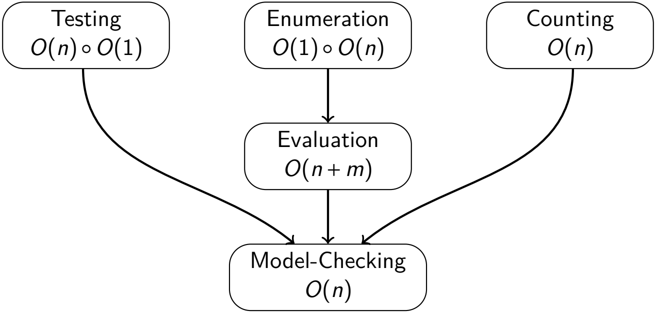
Which restrictions ?

No restriction on the database part	Highly expressive queries (MSO queries)	FO queries
$\Downarrow$	$\Downarrow$	$\Downarrow$
Only works for queries are conjunctive, acyclic and free-connex	Only works for trees (Graphs with bounded tree width)	**This talk !**
Bagan, Durand, Grandjean	Courcelle, Bagan, Segoufin, Kazana	

Other problems

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

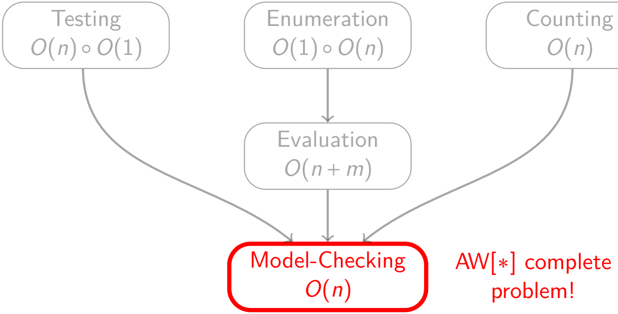
Model-Checking	:	Is this true ?	$O(n)$
Enumeration	:	Enumerate the solutions	$O(1) \circ O(n)$
Counting	:	How many solutions ?	$O(n)$
Testing	:	Is this tuple a solution ?	$O(1) \circ O(n)$
Evaluation	:	Compute the entire set	$O(n+m)$



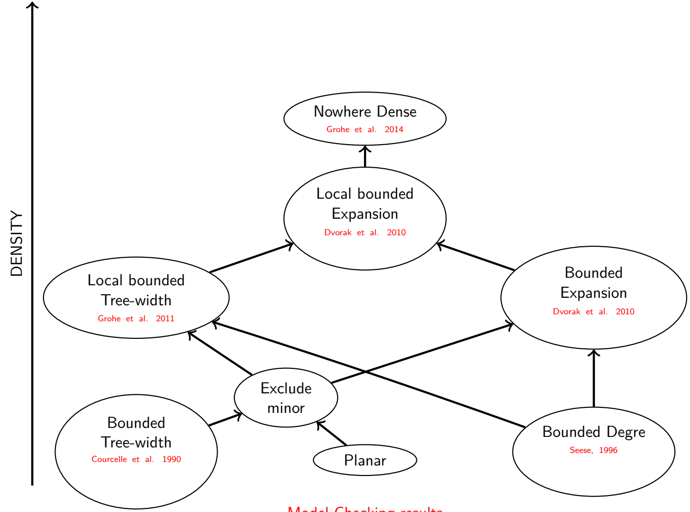
Other problems

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

Model-Checking	:	Is this true ?	$O(n)$
Enumeration	:	Enumerate the solutions	$O(1) \circ O(n)$
Counting	:	How many solutions ?	$O(n)$
Testing	:	Is this tuple a solution ?	$O(1) \circ O(n)$
Evaluation	:	Compute the entire set	$O(n+m)$

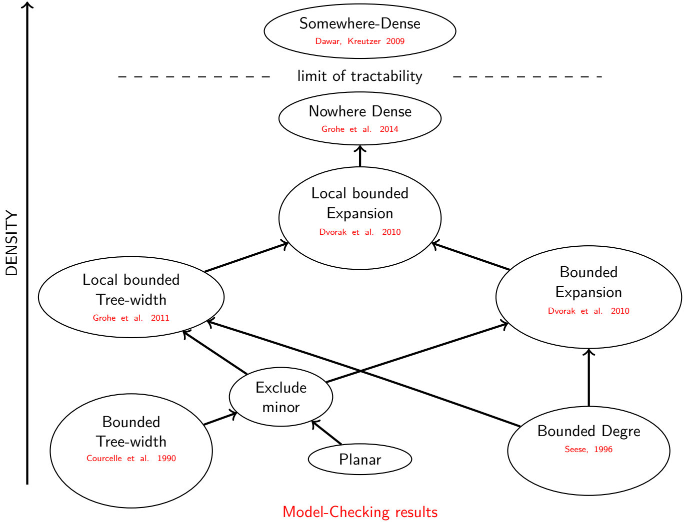


Classes of graphs closed under taking sub-graphs

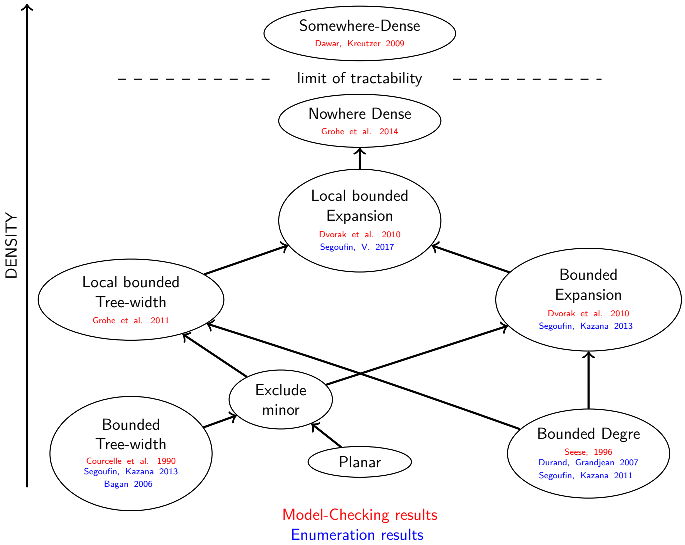


Model-Checking results

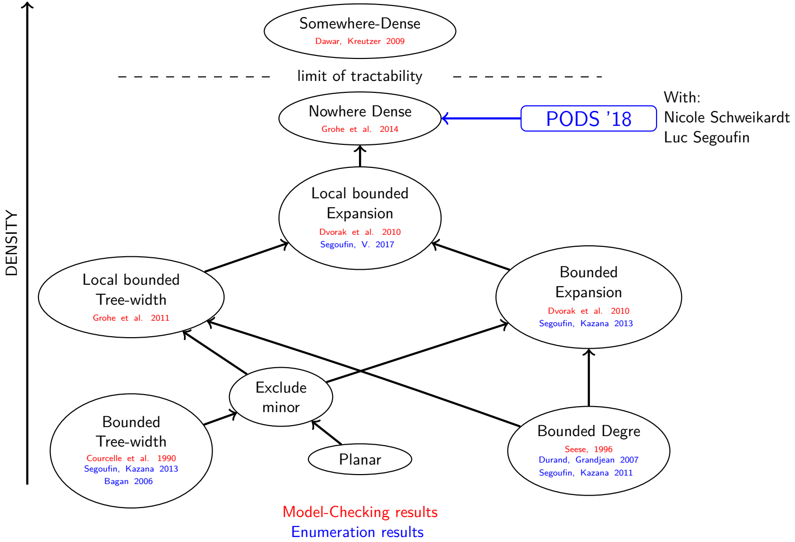
Classes of graphs closed under taking sub-graphs



Classes of graphs closed under taking sub-graphs



Classes of graphs closed under taking sub-graphs



Our results

Theorem (Schweikardt, Segoufin, V. 18')

Over *nowhere dense* classes of graphs, for every FO query, after a pseudo-linear preprocessing, we can:

- enumerate every solution with constant delay.
- test in constant time whether a given tuple is a solution.

Pseudo-linear ?

Definition

A function f is pseudo linear if and only if:

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

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

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

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

The game characterization

Definition : (ℓ, r)-Splitter game

A graph G and two players, Splitter and Connector.
Each turn:

- Connector picks a node c
- Splitter picks a node s
- $G' = N_r^G(c) \setminus s$

If in less than ℓ rounds the graph is empty, Splitter wins.

The game characterization

Definition : (ℓ, r)-Splitter game

A graph G and two players, Splitter and Connector.
Each turn:

- Connector picks a node c
- Splitter picks a node s
- $G' = N_r^G(c) \setminus s$

If in less than ℓ rounds the graph is empty, Splitter wins.

Theorem

$\mathscr{C}$ is nowhere dense if and only if there is a function $f_{\mathscr{C}}$ such that for every $G \in \mathscr{C}$ and every $r \in \mathbb{N}$:

Splitter has a wining strategy for the $(f_{\mathscr{C}}(r), r)$-splitter game on G.

How to play the (ℓ, r)-Splitter game on a graph G ?

- If G is a star, Splitter wins in 2 rounds.
- If G is a path, Splitter wins in $\log(r)$ rounds.
- If G is a tree, Splitter wins in r rounds.
- If G has degree d, splitter wins in d^r rounds.
- If G is a clique of size $> \ell$, Splitter looses the (ℓ, r)-splitter game.

Neighborhood cover

A neighborhood cover is a set of “representative” neighborhoods.

$\mathscr{X} := X_1, \ldots, X_n$ is a $(r, 2r)$ neighborhood cover if it has the following properties:

- $\forall a \in G, \quad \exists X \in \mathscr{X}, \quad N_r(a) \subseteq X$
- $\forall X \in \mathscr{X}, \quad \exists a \in G, \quad X \subseteq N_{2r}(a)$
- $\forall a \in G, \quad |\{i \mid a \in X_i\}|$ is pseudo-constant (smaller than $|G|^\epsilon$)

The examples queries

- $q_1(x,y) := \exists z \; E(x,z) \wedge E(z,y)$

  (The distance two query)

- $q_2(x,y) := \neg q_1(x,y)$

  (Nodes that are far apart)

How to use the game 1/2

G is now fixed

Goal : Given a node a we want to enumerate all b such that $q_1(a, b)$.
(Here $r = 2$)

- Base case: If Splitter wins the $(1, r)$-Splitter game on G.

  Then G is edgeless and there is no solution !

- By induction: assume that there is an algorithm for every G' such that Splitter wins the (ℓ, r)-Splitter game on G'.

How to use the game 2/2

Here, Splitter wins the $(\ell+1, r)$-game on G.

Idea :

1. Compute some new graphs on which Splitter wins the (ℓ, r) game.
2. Apply the induce algorithm for a particular query.
3. Enumerate those solutions.

How to use the game 2/2

Here, Splitter wins the $(\ell+1, r)$-game on G.

Idea :

1. Compute some new graphs on which Splitter wins the (ℓ, r) game.
2. Apply the induce algorithm for a particular query.
3. Enumerate those solutions.

For every bags X of the $(2,4)$-neighborhood cover, $X' := X \setminus \{s\}$.

For every $(a,b) \in G^2$ we have:

$$G \models q_1(a,b) \iff \bigvee_{X \in \mathscr{X}} X \models q_1(a,b) \iff \mathscr{X}(a) \models q_1(a,b)$$

How to use the game 2/2

Here, Splitter wins the $(\ell+1, r)$-game on G.

Idea :

1. Compute some new graphs on which Splitter wins the (ℓ, r) game.
2. Apply the induce algorithm for a particular query.
3. Enumerate those solutions.

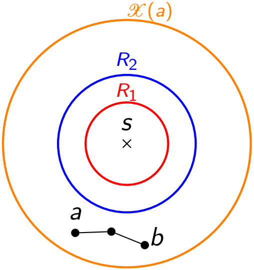
For every bags X of the $(2,4)$-neighborhood cover, $X' := X \setminus \{s\}$.

For every $(a,b) \in G^2$ we have:

$$G \models q_1(a,b) \iff \bigvee_{X \in \mathscr{X}} X \models q_1(a,b) \iff \mathscr{X}(a) \models q_1(a,b)$$

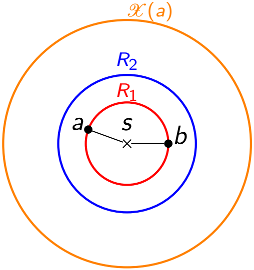
The new graph is $\mathscr{X}(a)$
Then, Splitter delete a node!

The new queries



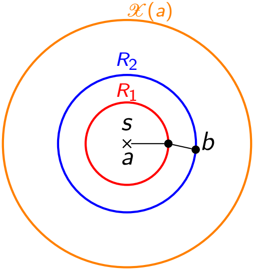
when there is still a 2-path not using s

the new query is:
$q_1(x, y)$



when s is on the only short path from a to b

the new query is:
$R_1(x) \wedge R_2(y)$



when $a = s$
(similarly for $b = s$)

the new query is:
$R_2(y)$

The second query

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

Two kinds of solutions:

- $b \in \mathscr{X}(a)$ (similar to the previous example)
- $b \notin \mathscr{X}(a)$ We need something else !

The second query

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

Two kinds of solutions:

- $b \in \mathscr{X}(a)$ (similar to the previous example)
- $b \notin \mathscr{X}(a)$ We need something else !

Goal: given a bag X, enumerate all $b \notin X$

The shortcut pointers

Given X we want to enumerate all b such that $b \not\in X$.

The shortcut pointers

Given X we want to enumerate all b such that $b \notin X$.

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

The shortcut pointers

Given X we want to enumerate all b such that $b \notin X$.

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

For all $X \in \mathscr{X}$ with $b_{max} \in X$, we have $NEXT(b_{max}, X) \ = \ NULL$

The shortcut pointers

Given X we want to enumerate all b such that $b \notin X$.

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

$$\text{For all } X \in \mathscr{X} \text{ with } b_{max} \in X, \text{ we have } NEXT(b_{max}, X) \ = \ NULL$$

$$NEXT(b,X) \in \{b+1 \ , \ NEXT(b+1, X)\}$$

The shortcut pointers

Given X we want to enumerate all b such that $b \notin X$.

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

$$\text{For all } X \in \mathscr{X} \text{ with } b_{max} \in X, \text{ we have } NEXT(b_{max},X) \ = \ NULL$$

$$NEXT(b,X) \in \{b+1 \ , \ NEXT(b+1,X)\}$$

G	b_1	b_2	b_3	$\cdots$	b_j	b_{j+1}	$\cdots$	b_{n-1}	b_n	
X	b_1	b_2	b_3	$\cdots$	b_i	b_{i+1}	$\cdots$	b_{n-1}	b_n	NULL
$\vdots$										
Y	b_1	b_2	b_3	$\cdots$	b_i	b_{i+1}	$\cdots$	b_{n-1}	b_n	NULL

Recap

We use :

- A new Hanf normal form for FO queries.[1]
- The algorithm for the model checking.[2]
- Neighbourhood cover.[2]
- Game characterization of nowhere dense classes.[2]
- Short-cut pointers dedicated to the enumeration.[3]

We can :

- Enumerate with constant delay after pseudo-linear preprocessing.
- Test in constant time after pseudo-linear preprocessing.

[1] Grohe, Schweikardt '18
[2] Grohe, Kreutzer, Siebertz '14
[3] Segoufin, V. '17

Future work

- Classes of graphs that are not closed under subgraphs
- Enumeration with update:
  What happens if a small change occurs after the preprocessing ?

  Existing results for: words, graphs with bounded tree-width or bounded degree.

Future work

- Classes of graphs that are not closed under subgraphs
- Enumeration with update:
  What happens if a small change occurs after the preprocessing ?

  Existing results for: words, graphs with bounded tree-width or bounded degree.

Thank you !

Questions ?