Query Evaluation With Constant Delay

Wojciech Kazana

INRIA Saclay, ENS de Cachan

PhD Thesis Defense

LSV, École normale supérieure de Cachan

September 16, 2013
Introduction

Enumeration

Examples

Results

Conclusions
Introduction – databases

Databases: storage of data and retrieval of information.

1. A store has its list of offered products.
   Can I buy orange shoes?
2. Private collection of photos.
3. Map of a metro system.
5. ...
Introduction – databases

Databases: storage of data and retrieval of information.

1. A store has its list of offered products.
2. Private collection of photos.
   On how many of my photos am I actually present?
3. Map of a metro system.
5. . . .
Introduction – databases

Databases: storage of data and retrieval of information.

1. A store has its list of offered products.
2. Private collection of photos.
3. Map of a metro system. Can I get from Château d’Eau to Bagneux with just one hop?
5. ...
Introduction – databases

Databases: storage of data and retrieval of information.

1. A store has its list of offered products.
2. Private collection of photos.
3. Map of a metro system.
4. Social network and its graph. Which pairs of people are in the 2-handshakes distance from each other?
5. . . .
Query Evaluation Problem:

Input:
- query $q(\bar{x})$
- database $D$ of size $|D|$

Output: $q(D)$

Which pairs of people are in the 2-handshakes distance from each other?

(1, 2), (1, 3), (1, Wojtek), (1, D), (1, E), (1, P), (1, Q), (1, R)
(2, 1), (2, 3), (2, Wojtek), (2, D), (2, E), (2, P), (2, Q), (2, R)
(3, 1), (3, 2), (3, Wojtek), (3, D), (3, E), (3, P), (3, Q), (3, R) ...
Query Evaluation Problem:

Input:
- query $q(\bar{x})$
- database $D$ of size $|D|$

Output:
$q(D)$

Special case: $q$ boolean = Model Checking Problem.

Are there two green friends? No
Query Evaluation Problem:

Input:
• query \( q(\overline{x}) \)
• database \( D \)

Output: \( q(D) \)

Issue: \( |q(D)| = O(||D||^k) \) if \( q \) has \( k \) free variables.

\[ ||D||^k \text{ is too big!} \]
Query Enumeration and Related Problems

Input:
- query \( q(\bar{x}) \)
- database \( D \)

**Enumeration:**
- compute first solution quickly,
- compute the rest with minimal *delay* between consecutive ones.

**Aim:** First solution in \( O(\|D\|) \), \( O(1) \) delay \( \rightarrow \) \textsc{Constant-Delay}_{lin}
Query Enumeration and Related Problems

Input:
- query \( q(\bar{x}) \)
- database \( D \)

**Enumeration:**
- compute first solution quickly,
- compute the rest with minimal *delay* between consecutive ones.

**Aim:** First solution in \( O(\|D\|) \), \( O(1) \) delay \( \rightarrow \) **CONSTANT-DELAY**

**In practice:**
- the \( O(\|D\|) \) preprocessing is a linear refactorization of the input database (usually adding to it some additional navigational power),
- the refactorized database can then be traversed efficiently, producing new solutions after only constant delays.
Query Evaluation Problem:

Input:
- query \( q(\bar{x}) \)
- database \( D \)

Output: \( q(D) \)

Which pairs of people are in the 2-handshakes distance from each other?

\((1, 2), (1, 3), (1, \text{Wojtek}), (1, D), (1, E), (1, P), (1, Q), (1, R)\)
\((2, 1), (2, 3), (2, \text{Wojtek}), (2, D), (2, E), (2, P), (2, Q), (2, R)\)
\((3, 1), (3, 2), (3, \text{Wojtek}), (3, D), (3, E), (3, P), (3, Q), (3, R) \ldots\)
Query **Enumeration** Problem:

Input:
- query \( q(\overline{x}) \)
- database \( D \)

Output:

\[ q(D) \]

Which pairs of people are in the 2-handshakes distance from each other?

\((1, 2)\)
Query Enumeration and Related Problems

Query **Enumeration** Problem:

Input:
- query \( q(\overline{x}) \)
- database \( D \)

Output:
- \( q(D) \)

Which pairs of people are in the 2-handshakes distance from each other?
(1, 2), (1, 3)
**Query Enumeration Problem:**

**Input:**
- query $q(\bar{x})$
- database $D$

**Output:**
- $q(D)$

Which pairs of people are in the 2-handshakes distance from each other?
- $(1, 2)$
- $(1, 3)$
- $(1, \text{Wojtek})$
Query Enumeration Problem:

Input:
- query \( q(\overline{x}) \)
- database \( D \)

Output:

Which pairs of people are in the 2-handshakes distance from each other?

\( (1, 2), (1, 3), (1, \text{Wojtek}), (1, D) \)
Query Enumeration Problem:

Input:
- query \( q(\bar{x}) \)
- database \( D \)

Output:
\[ q(D) \]

Which pairs of people are in the 2-handshakes distance from each other?

\( (1, 2), (1, 3), (1, \text{Wojtek}), (1, D), (1, E), (1, P), (1, Q), (1, R) \)
\( (2, 1), (2, 3), (2, \text{Wojtek}), (2, D), (2, E), (2, P), (2, Q), (2, R) \)
\( (3, 1), (3, 2), (3, \text{Wojtek}), (3, D), (3, E), (3, P), (3, Q), (3, R) \ldots \)
Counting Problem:

Input:
- query $q(\bar{x})$
- database $D$

Output: $|q(D)|$

Aim: $O(\|D\|)$

How many pairs of people are in the 2-handshakes distance from each other? 78
Testing Problem:

Input:
- query \( q(\bar{x}) \)
- database \( D \)

Dynamical output:
given \( \bar{v} \), answer \( \bar{v} \in q(D) \)

Aim:
- preprocessing (once, \( \bar{v} \) unknown) \( O(\|D\|) \)
- answering (multiple times) \( O(1) \)

Is \((1, P)\) in the 2-handshakes distance? \(\text{Yes}\)
Is \((A, E)\) in the 2-handshakes distance? \(\text{No}\)
Remark 1
\textsc{Constant-Delay}_{lin} \textit{enumeration} \rightarrow O(\|D\| + |q(D)|) \textit{evaluation}.

Remark 2
\textsc{Constant-Delay}_{lin} \textit{enumeration} \rightarrow O(\|D\|) \textit{model checking}.
Computational model – RAM machine

- Necessary, since we want to talk about linear time.
- We assume that the elements can be compared in constant time.

\[ A \leq_{\text{lex}} P \leq_{\text{lex}} \text{Wojtek} \]

In real life: user = (short) e-mail address

- We can sort lexicographically tuples of constant size in linear time.

Radix sort

- We can follow pointers in constant time.

Direct access to the \( n \)-th cell of an array.

- Coding of a graph:

List of consecutive edges.

NOT an adjacency matrix!
Introduction

Enumeration

Examples

Results

Conclusions
Example 1: enumeration of edges

Database:
\[ \text{graph } G = (V, E) \]
\[ \|G\| = |V| + |E| \]

- \textit{Constant-Delay} \textit{lin} \textit{enumeration} is not too difficult.

- \( O(\|G\|) \) \textit{counting} is not too difficult.

- \textit{Testing} requires \textit{logarithmic time}.
  - \( O(1) \) testing if \( G \) is a tree.
Example 2: complement of a graph

Database:
graph \( G = (V, E) \)
\( \|G\| = |V| + |E| \)

Query:
\( q(x, y) = \neg E(x, y) \)

- **Constant-Delay** \( \text{lin} \) enumeration already not trivial.
- \( O(\|G\|) \) counting is not too difficult.
- Testing still requires logarithmic time.
  - \( O(1) \) testing if \( G \) is a tree.
Example 2: complement of a graph

Is given by the following list:

- $(1, 1)$
- $(2, 1)$
- $(3, 2)$
- $(3, 4)$
- $(1, 4)$
- $(3, 1)$
- $(3, 3)$
- $(4, 4)$
Example 2: complement of a graph

\[
\begin{align*}
(1,1) & \rightarrow (1,2) \rightarrow (1,3) \rightarrow (1,4) \\
(2,1) & \rightarrow (2,2) \rightarrow (2,3) \rightarrow (2,4) \\
(3,1) & \rightarrow (3,2) \rightarrow (3,3) \rightarrow (3,4) \\
(4,1) & \rightarrow (4,2) \rightarrow (4,3) \rightarrow (4,4)
\end{align*}
\]
Example 2: complement of a graph
Example 3: 2-handshake distance

Database:
graph \( G = (V, E) \)
\[ \|G\| = |V| + |E| \]

Query:
\[ q(x, y) = \exists z \in E(x, z) \land E(z, y) \]

- **CONSTANT-DELAY \(_{lin}\) enumeration** not possible? (Bagan’09)
  - **CONSTANT-DELAY \(_{lin}\) enumeration** if \( G \) has bounded degree.

- **\( O(\|G\|) \) counting** not possible?
  - **\( O(\|G\|) \) counting** if \( G \) has bounded degree.

- **\( O(1) \) testing** not possible? (Bagan’09)
  - **\( O(1) \) testing** if \( G \) has bounded degree.
Introduction

Enumeration

Examples

Results

Conclusions
Classes of graphs

- Bounded tree-width
- Excluded minor
- Planar
- Excluded topological minor
- Bounded degree
- Bounded expansion
(K, Segoufin’11) gives a new proof of:

**Theorem 1 (Durand, Grandjean’07)**

*\( \mathcal{C} \) with bounded degree, \( q(\bar{x}) \in \text{FO} \), given \( G \in \mathcal{C} \), the *enumeration* of \( q \) over \( G \) is in \text{CONSTANT-DELAY}_{\text{lin}}."

Moreover, the output is returned in the lexicographical order.

*The hidden constants are triply exponential in \( |q| \) (K, Segoufin’11).*
Results - FO over structures with bounded degree

(K, Segoufin’11) gives new proofs of:

**Theorem 2 (Bagan, Durand, Grandjean, Olive’08)**

\( \mathcal{C} \) with bounded degree, \( q(\bar{x}) \in \text{FO} \), given \( G \in \mathcal{C} \), the [counting](#) \( |q(D)| \) is in \( O(\|G\|) \).

**Theorem 3 (Lindell’08)**

\( \mathcal{C} \) with bounded degree, \( q(\bar{x}) \in \text{FO} \), given \( G \in \mathcal{C} \), the [testing](#) of \( q \) over \( G \) is in \( O(1) \) after \( O(\|G\|) \) preprocessing.

The hidden constants are triply exponential in \( |q| \) (K, Segoufin’11).
Comments

• The original proof:
  
  ○ Quantifier elimination procedure based on bijective representation.
  
  ○ Non-elementary dependency on $|q| \left( 2^\left(2^{\ldots^{2}}\right) \right)$. 

• Our proof:
  
  ○ Based on Gaifman locality of FO.
  
  ○ Gives the $2^{2^{2O(|q|)}}$ dependency on $|q|$. 

Wojciech Kazana (INRIA, ENS Cachan)  Query Evaluation With Constant Delay  September 16th, 2013  28 / 46
Theorem 4 (K, Segoufin’13)

\[ \mathcal{C} \text{ with bounded expansion, } q(\vec{x}) \in \text{FO, given } G \in \mathcal{C}, \]
the \textit{enumeration} of \( q \) over \( G \) is in \text{CONSTANT-DELAY} lin.

Moreover, the output is returned in the lexicographical order.
Results - FO over structures with bounded expansion

Theorem 5 (K, Segoufin’13)

$\mathcal{C}$ with bounded expansion, $q(\overline{x}) \in \text{FO}$, given $G \in \mathcal{C}$, the counting $|q(D)|$ is in $O(\|G\|)$.

Theorem 6 (K, Segoufin’13)

$\mathcal{C}$ with bounded expansion, $q(\overline{x}) \in \text{FO}$, given $G \in \mathcal{C}$, the testing of $q$ over $G$ is in $O(1)$ after $O(\|G\|)$ preprocessing.
• The hidden constants are non-elementary \( (2^{2^\ddots^2}) \).

\[ |q| \]

• This is unavoidable already for model checking over unranked trees, unless FPT = AW[\ast] (Frick, Grohe’04).
Proof Strategy

• Quantifier elimination procedure:
  
  o For all $q(\bar{x}y)$ quantifier free,
  
  o Exists $q'(\bar{x})$ quantifier free s.t.
  
  o given $G \in \mathcal{C}$, in $O(\|G\|)$ we construct $G'$ s.t.
  
  o $(\exists yq)(G) = q'(G')$.

Graph $G'$ is an “augmentation” of $G$, which allows us to continue the inductive process.

• We then solve the quantifier free case.

Both steps are not trivial.
Results - MSO over structures with bounded treewidth

(K, Segoufin’12) gives a new proof of:

Theorem 7 (Bagan’06)

\( \mathcal{C} \) with bounded treewidth, \( q(\overline{x}) \in \text{MSO} \), given \( G \in \mathcal{C} \), the \textit{enumeration} of \( q \) over \( G \) is in \textsc{Constant-Delay} \textit{lin}. 
Results - MSO over structures with bounded treewidth

(K, Segoufin’12) gives new proofs of:

**Theorem 8 (Arnborg, Lagergren, Seese’91)**

\[ \mathcal{C} \text{ with bounded treewidth, } q(\vec{x}) \in \text{MSO}, \text{ given } G \in \mathcal{C}, \]
the **counting** \(|q(D)|\) is in \(O(\|G\|)\).

**Theorem 9**

\[ \mathcal{C} \text{ with bounded treewidth, } q(\vec{x}) \in \text{MSO}, \text{ given } G \in \mathcal{C}, \]
the **testing** of \(q\) over \(G\) is in \(O(1)\) after \(O(\|G\|)\) preprocessing.
Comments

- The original proof:
  - Allows for monadic second-order free variables.
  - No bound on total memory usage during the enumeration phase.
  - Rather complicated reasoning concerning tree automatons.

- Our proof:
  - Only first-order free variables.
  - Constant total memory usage during the enumeration phase.
  - Sequence of reduction steps.

- In both cases the hidden constants are non-elementary $\left(2^{2^{|q|}}\right)$. 
Theorem 6 (Bagan’06)

\[ C \text{ with bounded treewidth, } q(\bar{x}) \in \text{MSO}, \text{ given } G \in C, \]

the enumeration of \( q \) over \( G \) is in \( \text{CONSTANT-DELAY}_{\text{lin}} \).

The proof of (K, Segoufin’12) is a sequence of consecutive reduction steps:
MSO enumeration – outline

Reduction steps:

1. Trees instead of structures of bounded tree-width.
   Compute the tree decomposition in linear time. (Bodlaender)
   Interpret the tree decomposition in MSO. (Courcelle)

2. Binary trees instead of unranked trees.
   First child – next sibling encoding. (Rabin)

3. Ancestor-typed outputs including all the least common ancestors.

   Composition Lemma.

5. Binary queries from $\Sigma_2(<)$.
   Colcombet.

Step 3: Ancestor-typed outputs including all the least common ancestors.

\[ q(x,y) \]

\[ z = \text{lca}(x,y) \]

\[ q'(x,y,z) \]

\[ x \]

\[ y \]

\[ x \]

\[ y \]

\[ z = \text{lca}(x,y) \]

\[ q'(x,y,z) \]
Step 4: Binary queries.

\[ q(x, y, z) = \bigvee_{q', q''} q'(x, y) \land q''(x, z). \]

Disjunction is exclusive.

Composition Lemma for MSO over trees (can be proved using a simple Ehrenfeucht-Fraïssé game argument).
MSO enumeration – outline (5/6)

Step 5: Binary queries from $\Sigma_2(\prec)$.

$s'(z, x)$ and $s''(z, y)$ are of the form $\exists \overline{v} \forall \overline{u} \theta(x, y, z, \overline{v}, \overline{u})$, where $\theta$ is quantifier free.

\[
\begin{array}{c}
q'(z, x) \\
\downarrow \\
x
\end{array}
\quad
\begin{array}{c}
q''(z, y) \\
\downarrow \\
y
\end{array}
\quad
\begin{array}{c}
s'(z, x) \\
\downarrow \\
x
\end{array}
\quad
\begin{array}{c}
s''(z, y) \\
\downarrow \\
y
\end{array}
\]

Theorem 10 (Colcombet)

Over binary trees, every MSO formula $q(x, y)$ is equivalent to a $\Sigma_2^+(\prec)$ formula $q'(x, y)$. $q' = \exists \overline{v} \forall \overline{u} \theta(x, y, z, \overline{v}, \overline{u})$, where $\theta$ is a disjunction of conjunctions of atomic predicates or MSO queries with one free variable or atoms using $\prec$. 

Wojciech Kazana  (INRIA, ENS Cachan)  Query Evaluation With Constant Delay  September 16th, 2013  40 / 46
The rest is an induction on the number of free variables:

\[ q(x, y, z, u) = q'(x, y, z) \land q''(x, u) \]

- We inductively enumerate \( q'(x, y, z) \).
- For every solution \((a, b, c)\) to \( q' \) we (efficiently) extend it with all solutions to \( q'' \) of the form \((a, \_).\)
Summary

- Bounded tree-width
  - MSO - CDlin, N-Elem

- Excluded minor

- Excluded topological minor

- Bounded expansion
  - FO - CDlin, N-Elem

- Nowhere dense

- Somewhere dense
  - FO - MC not in FPT (Dawar, Kreutzer)

- Planar
  - FO - CDlin, 3-EXP

- Bounded degree
  - FO - CDlin, N-Elem
Push further along the lines of the current approach:

- Consider FO queries over classes of nowhere dense structures.
- Consider other query languages over structures with particular properties.
**Perspectives**

\textbf{\textsc{Constant-Delay}} \textit{lin} implies evaluation algorithm working in time 
\[ O(\|G\| + \|q(G)\|). \]

- When is the converse true?

When is \textbf{\textsc{Constant-Delay}} \textit{lin} enumeration impossible?

- Most of the lower bounds require complexity assumptions (matrix multiplication, \textsc{FPT} \neq \textsc{AW}[\ast], etc.).

- In the mentioned cases also \( O(\|G\| + \|q(G)\|) \) evaluation is not possible.

- How much can we prove directly?
Perspectives

Let $A, B \in \text{CONSTANT-DELAY}_{lin}$ (black boxes).

Under what assumptions

- $A \cup B \in \text{CONSTANT-DELAY}_{lin}$, (The best understood.)
- $A \cap B \in \text{CONSTANT-DELAY}_{lin}$,
- $\bar{A} \in \text{CONSTANT-DELAY}_{lin}$?
Thank You!