Automata-theoretic and Datalog-based solutions
of MSO Evaluation Problems
over Structures of bounded-treewidth

Labrini Kalantzi
University of Athens
(joint work with E. Foustoucos)

2nd Workshop on Graph Decompositions
Theoretical, Algorithmic and Logical Aspects
October 18-22, 2010
CIRM, Luminy, Marseille
The MSO evaluation problem:

- an MSO formula: defines a property that characterizes tuples of nodes/subsets of nodes
- find such tuples that satisfy this property in a given relational structure

Monadic Second Order logic (MSO): rather expressive language, e.g. defines 3-colorability of graphs

Linear algorithms over structures of bounded treewidth (Courcelle’s Theorem)

We propose a direct automata-theoretic reduction

We propose evaluation of whole MSO via datalog
(Gottlob, Pichler, Wei: a datalog solution for restricted class of unary MSO queries)
Talk Outline

1. Automata-theoretic Solution
   - The idea - Basic Notions
   - The main theorem

2. Datalog algorithms for the MSO evaluation problem
   - The datalog idea
   - The datalog approach to MSO evaluation
   - Vertex Cover
The problem

- **Input**: a relational $\tau$-structure $\mathcal{A}$, $\tau = \{R_1, \ldots, R_\ell\}$; a (binary, ordered) tree-decomposition $D$ of $\mathcal{A}$ of width $w$; an MSO[\tau] formula $\phi(X_1, \ldots, X_k)$

- **Output**: Set $\phi(\mathcal{A})$ of satisfying assignments of $\phi$ over $\mathcal{A}$

We define assignment automata that produce $\phi(\mathcal{A})$ & then use them to provide a datalog solution for the computation of $\phi(\mathcal{A})$. 
Basic facts about our automata: $w$-dec-Assign$_\phi$

A $w$-decomposition (assignment) automaton for $\phi(X_1, \ldots, X_k)$

- non-deterministic bottom-up tree automaton, runs over trees encoding a $(\mathcal{A}, \mathcal{D})$ & a decoding mechanism
- special form of state set

$$Q \times W^k;$$

$W = \{1, \ldots, w + 1\}$: set of bag positions of $\mathcal{D}$

Thus runs

$$\rho = \sigma; \varepsilon.$$

- $w$-dec-Assign$_\phi$ to produces $\phi(\mathcal{A})$ as part of its successful runs, in an encoded form $\sim$ automaton accepting $\mathcal{D}; \varepsilon$ or $(\mathcal{D}, \mathcal{A}); \varepsilon$
- the decoding is performed w.r.t. $\mathcal{D}$
An example of a run producing an assignment of $\phi(X)$

Graph $G$, ordered binary tree-decomposition $D$ ($W = \{1, 2, 3\}$)

$\begin{align*}
\text{nodes:} & \\
\text{a run } \rho = \sigma; \varepsilon: & (q_1, \{1, 3\}) & (q_2, \{2\}) & (q_3, \emptyset) \\
\text{decoding } \varepsilon: & \downarrow \pi_{\{1,3\}}(a, b, e) & \downarrow \pi_{\{2\}}(a, b, e) & \downarrow \pi_{\emptyset}(b, c) \\
B_\varepsilon: & \{a, e\} & \cup & \{d\} & \cup & \emptyset
\end{align*}$

Automata select elements via their top-most occurrence in $D$
Decoding $\varepsilon : T \mapsto W^k$ to $\overline{B}_\varepsilon \in (\mathcal{P}(A))^k$ w.r.t. $D$

The decoding procedure w.r.t. $D = (T, (\overline{a}_n)_{n \in T})$

- At each node $n$, compute $B_{\varepsilon,n}$

  $$\varepsilon(n) = \begin{pmatrix} I^1 & \ldots & I^k \end{pmatrix}$$
  $$\downarrow_D \quad \ldots \quad \downarrow_D$$

  $$B_{\varepsilon,n} = \begin{pmatrix} \pi_{I^1} \overline{a}_n & \ldots & \pi_{I^k} \overline{a}_n \end{pmatrix}$$

- Perform component-wise union

  $$\overline{B}_\varepsilon = \bigcup_{n \in T} B_{\varepsilon,n}$$

Set of computed assignments

$$w\text{-}dec\text{-}Assign_\phi(D; A) = \left\{ \bigcup_{n \in T} \overline{B}_\varepsilon \mid \varepsilon \in a\text{-}suc\text{-}runs_{\phi,D;A} \right\}$$
The input tree $\mathcal{A}; \mathcal{D}$ for our automata

For input $\tau$-structure $\mathcal{A}$ and input tree decomposition $\mathcal{D} = (\mathcal{T}, (\overline{a_n} \in \mathcal{T}))$: colored tree $\mathcal{A}; \mathcal{D} = (\mathcal{T}, c)$ has

1. **underlying tree** same as in the tree-decomposition
2. **color** at $n$ corresponds to the information about
   - the bags at $n$ and parent $n'$: $|a_n|$, $a_n \cap a_n'$
   - and substructure of $\mathcal{A}$ induced at $n$: $R^\mathcal{A}|_{a_n}$

   in terms of positions of occurrences of the corr. elements

More precisely, $c(n) = (s, P, S_1, \ldots, S_\ell) \in \Gamma_{\tau,w}$
where $\Gamma_{\tau,w} = W \times \mathcal{P}(W^2) \times \mathcal{P}(W^{r_1}) \times \cdots \times \mathcal{P}(W^{r_\ell})$,
$W = \{1, \ldots, w + 1\}$.

Very similar to tree $\mathcal{T}^*$ of Flum, Frick, Grohe, 2002.
Example

$G$ with edge relation

$E^G = \{(a, b), (b, a), (b, e), (e, b), (a, d), (d, a), (d, e), (e, d), \ldots\}$

Tree $G; D$

$E^G|_{a_{n_1}}$

$(a, b, e) \mapsto (3, \emptyset, \{(1, 2), (2, 1), (2, 3), (3, 2)\})$

$(a, d, e) \mapsto (3, \{(1, 1), (3, 3)\}, \{\ldots\})$

$a_{n_2}^i = a_{n_1}^i, i = 1, 3$

$E^G|_{a_{n_2}}$

$\ldots$
Main Automata Theorem

For every $\text{MSO}[\tau]$-formula $\phi(X_1, \ldots, X_k)$ and for every integer $w$, there exists a $w$-decomposition ($k$-assignment) automaton $w\text{-dec-Assign}_\phi$, that running over tree $A; D$ where $A$ is a $\tau$-structure with treewidth $w$ and $D = (T, (\overline{a}_n)_{n \in T})$ is an ordered tree-decomposition of $A$ with width $w$ computes the satisfying assignments of $\phi$ over $A$. i.e.

$$w\text{-dec-Assign}_\phi(A; D) = \phi(A)$$
Proof Sketch

- Construction of automata for atomic formulas $ln(x, X)$ and $R(x_1, \ldots, x_\ell)$:
  - Number of states for $ln(x, X)$: 3
  - Number of states for $E(x, y)$: $2^{w+2} + 3$
  - $q_0, q_1^I, q_2^I, q_a, q_f$ for $I \subseteq W = \{1, \ldots, w + 1\}$

  **example:**
  
  $a, b, e \rightarrow q_a, (\{1\}, \emptyset)$

  $q_2^{\{1,3\}}, (\emptyset, \{2\}) \leftarrow a, d, e$
  
  $b, c \rightarrow q_0, (\emptyset, \emptyset)$

- The construction of automata for not atomic formulas is analogous to the usual tree-automata inductive definitions

Make sure that produce proper encodings (elements of $A$ are selected via their top-most occurrences).
Counting version of our automata

The run $\rho$ of $\text{w-dec-Assign}^{\text{count}}_\phi$ over $T_{I;A}$ is a mapping

$$T \rightarrow Q \times \mathcal{P}(W) \times \{1, \ldots, |A|\}$$

Three variants of MSO evaluation problems that require evaluating the size of sets involved:

1. “$\psi \exists,k = \exists^k X \phi(X)$” which is true if there exists a set $X$ of size $k$ satisfying MSO formula $\phi(X)$.

2. “count-$\phi_{\min}/\max(X)$” whose (unique) satisfying assignment is the size of the sets satisfying $\phi_{\min}/\max(X)$

3. “$\psi(X) = \phi_{\min}/\max(X) = \min/\max X : \phi(X)$” whose satisfying assignments are those satisfying assignments of the MSO formula $\phi$ having minimum/maximum size,
From MSO to datalog

the idea of the reduction

<table>
<thead>
<tr>
<th>MSO-formula $\phi$</th>
<th>$\rightarrow$</th>
<th>datalog query $Q_\phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaluation</td>
<td>$[w\text{-dec-Assign}_\phi]$</td>
<td>evaluation</td>
</tr>
</tbody>
</table>

- Consider automaton $w\text{-dec-Assign}_\phi$
- Given $A$ and $D$, program $\Pi_{\tau,w}$ produces $A;D$
- Program $\Pi_\phi$ compute assignments encoded in runs of $w\text{-dec-Assign}_\phi$ over $A;D$ performing component-wise unions at each node $n$:

$$\overline{B}_{\epsilon,T_n} = \overline{B}_{\epsilon,n} \cup \overline{B}_{\epsilon,T_{n_1}} \cup \overline{B}_{\epsilon,T_{n_2}}$$

Then it selects the ones corresponding to successful runs
The final datalog solution is a result of **rewriting optimization method for datalog** (known as “magic sets” algorithm); thus we obtain **optimized programs** excluding the computation of non satisfying.

The resulting program correspond an efficient automata evaluation algorithm that coincides with Flum, Frick, Grohe algorithm.

Moreover we define automata construction and minimization in datalog, we thus obtain a fully database theoretic i.e. given $\phi$, $A$, $D$ there exits a datalog query computing $\phi(A)$ thats does not requires the automaton since it is able to construct it.
The optimized program $\Pi^S_{\phi}$

I. (Optimum) filter program defining predicates $suc-q$
   successful states of $\omega$-$dec$-$Assign_{\phi}$ are computed

II. $r_{\Delta_0}$ : $assign-q(x, \overline{x}) \leftarrow suc-q(x), color-\gamma(x), leaf(x),$
    $decomp-I(x, \overline{x})$

    $r_{\Delta}$ : $assign-q(x, \overline{x}) \leftarrow suc-q(x), color-\gamma(x),$
    $succ_0(x, x_1), assign-q_1(x_1, \overline{y}),$
    $succ_1(x, x_2), assign-q_2(x_2, \overline{z})$
    $decomp-I(x, \overline{v}), cw-union(\overline{x}, \overline{v}, \overline{y}, \overline{z})$

    $r_{F}$ : $\phi$-$assign(\overline{e}) \leftarrow assign-q(x, \overline{x})$

restrictions of satisfying assignments are computed
Datalog results

Theorem

Let $\tau = \{R_1, \ldots, R_\ell\}$; for every MSO[$\tau$] formula $\phi(X_1, \ldots, X_k)$ and integer $w$, we can construct a datalog query $Q_{\phi,k}^w = (\Pi_{\phi,k}^w, \text{assign})$ of arity $k+1$ s. t. for every $\tau$-structure $A$ with special tree-decomposition $D$ of width $w$, $$(B_1, \ldots, B_k) \in \phi(A) \text{ iff } (B_1, \ldots, B_k) \in Q_{\phi}^w(D^U_{A,I})$$
Complexity at most $(\ell + 2 + 2^k \cdot (w+1)+1 \cdot s^2 + |\phi(A)|) \cdot |T| + |\phi(A)|$.

Theorem

The class of $k$-ary MSO-definable queries over relational structures of bounded treewidth is $(k+1)$-datalog definable.
“MSO evaluation problems involving counting”: evaluation via datalog

Three variants that require evaluating the size of sets involved:

1. “\(\exists \psi \exists^k X \phi (X)\)” : Optimized monadic datalog program
   Complexity \((\ell + 2 + 2^{w+1} \cdot s^2 + 1) \cdot |T|\).
   (one traversal of the tree-decomposition)

2. “\(\text{count-} \phi_{\min / \max}(X)\)” : Monadic datalog program

3. “\(\psi (X) = \phi_{\min / \max}(X) = \min / \max X \phi (X)\)” : Optimized
datalog program of arity 2

Extend datalog with predicates corresponding to arithmetic operations over integers.
The vertex cover automata

**Table:** Deterministic automaton for formula \( \phi(Z) \equiv "Z \text{ is a vertex cover}". 

<table>
<thead>
<tr>
<th>Transition</th>
<th>Conditions satisfied at current node ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma, \emptyset/L ) ( \mapsto Q^I )</td>
<td>( \text{NotEdge}<em>\gamma \land \text{AdjInPar}</em>\gamma = I ) / ( L \neq \emptyset \land \text{set}(L) \land \text{NotEdge}<em>{\gamma \setminus L} \land \text{AdjInPar}</em>{\gamma \setminus L} = I )</td>
</tr>
<tr>
<td>( Q^J, Q^S, \gamma, \emptyset ) ( \mapsto Q^I )</td>
<td>( \text{NotEdge}<em>\gamma \land (J \cup S) \cap V</em>\gamma = \emptyset \land \text{SameInPar}(J \cup S) \cup \text{AdjInPar}_\gamma = I )</td>
</tr>
<tr>
<td>( Q^J, Q^S, \gamma, L ) ( \mapsto Q^I )</td>
<td>( L \neq \emptyset \land \text{set}(L) \land \text{NotEdge}<em>{\gamma \setminus L} \land (J \cup S) \cap (V</em>\gamma \setminus L) = \emptyset \land \text{SameInPar}(J \cup S) \cup \text{AdjInPar}_{\gamma \setminus L} = I )</td>
</tr>
<tr>
<td>( q_1, q_2, \gamma, I ) ( \mapsto Q_f )</td>
<td>all cases that were not considered above</td>
</tr>
</tbody>
</table>