Monadic Datalog Containment on Trees

André Frochaux\(^1\) Martin Grohe\(^2\) Nicole Schweikardt\(^1\)

\(^1\)Goethe-University Frankfurt am Main, Germany
\(^2\)RWTH Aachen, Germany

AMW 2014
Universidad Nacional de Colombia
June 5th, 2014
Monadic Datalog

A program $\mathcal{P}$ in monadic Datalog (for short: mDatalog) is a finite set of Datalog rules $r$ of the form

$$h(x) \leftarrow b_1(\vec{x}_1), b_2(\vec{x}_2), \ldots, b_n(\vec{x}_n).$$

The semantics is defined by the immediate consequence operator $T_\mathcal{P}$.

**Example:**

$$\text{Reach}(x) \leftarrow \text{Red}(x)$$
$$\text{Reach}(x) \leftarrow \text{Reach}(y), \ E(y, x)$$
Monadic Datalog

A program \( \mathcal{P} \) in *monadic Datalog* (for short: *mDatalog*) is a finite set of Datalog rules \( r \) of the form

\[
h(x) \leftarrow b_1(\vec{x}_1), b_2(\vec{x}_2), \ldots, b_n(\vec{x}_n).
\]

The semantics is defined by the *immediate consequence operator* \( \mathcal{T}_\mathcal{P} \).

**Example:**

\[
\begin{align*}
\text{Reach}(x) & \leftarrow \text{Red}(x) \\
\text{Reach}(x) & \leftarrow \text{Reach}(y), E(y, x)
\end{align*}
\]

\[
\mathcal{T}_\mathcal{P}^0(G) = \left\{ \begin{array}{l}
\text{Red}(a), E(a, b), E(a, d), \\
E(b, a), \ldots, E(i, h)
\end{array} \right\}
\]

G:

```
g --- h ----- i
  |       |
  v       v
  d       f
    |     |
    v     v
  e --- c
    |     |
    v     v
  b --- a
```

A. Frochoux, M. Grohe, N. Schweikardt

Monadic Datalog Containment on Trees

2/8
Monadic Datalog

A program $\mathcal{P}$ in *monadic Datalog* (for short: *mDatalog*) is a finite set of Datalog rules $r$ of the form

$$h(x) \leftarrow b_1(\vec{x}_1), b_2(\vec{x}_2), \ldots, b_n(\vec{x}_n).$$

The semantics is defined by the *immediate consequence operator* $\mathcal{T}_\mathcal{P}$.

**Example:**

$$\text{Reach}(x) \leftarrow \text{Red}(x)$$
$$\text{Reach}(x) \leftarrow \text{Reach}(y), E(y, x)$$

$$\mathcal{T}_\mathcal{P}^1(G) = \mathcal{T}_\mathcal{P}^0(G) \cup \{ \text{Reach}(a) \}$$
Monadic Datalog

A program $\mathcal{P}$ in *monadic Datalog* (for short: *mDatalog*) is a finite set of Datalog rules $r$ of the form

$$
 h(x) \leftarrow b_1(\vec{x}_1), b_2(\vec{x}_2), \ldots, b_{n_r}(\vec{x}_{n_r}).
$$

The semantics is defined by the *immediate consequence operator* $T_\mathcal{P}$.

**Example:**

$$
\begin{align*}
\text{Reach}(x) & \leftarrow \text{Red}(x) \\
\text{Reach}(x) & \leftarrow \text{Reach}(y), E(y, x)
\end{align*}
$$

$$
T^2_\mathcal{P}(G) = T^1_\mathcal{P}(G) \cup \{ \text{Reach}(b), \text{Reach}(d) \}
$$

G:

```
 a -- b -- c
|    |    |
|    |    |
|    |    | d
|    | e -- f
|    |    |
|    |    | g
|    |    |
|    |    | h
|    |    |
|    |    | i
```

A. Frochaux, M. Grohe, N. Schweikardt
Monadic Datalog Containment on Trees

2/8
Monadic Datalog

A program $\mathcal{P}$ in *monadic Datalog* (for short: *mDatalog*) is a finite set of Datalog rules $r$ of the form

$$h(x) \leftarrow b_1(\vec{x}_1), \ b_2(\vec{x}_2), \ldots, \ b_n(\vec{x}_n).$$

The semantics is defined by the *immediate consequence operator* $T_{\mathcal{P}}$.

**Example:**

- $\text{Reach}(x) \leftarrow \text{Red}(x)$
- $\text{Reach}(x) \leftarrow \text{Reach}(y), \ E(y, x)$

$$T^3_{\mathcal{P}}(G) = T^2_{\mathcal{P}}(G) \cup \{ \text{Reach}(c) \}$$
Monadic Datalog

A program \( \mathcal{P} \) in \textit{monadic Datalog} (for short: \textit{mDatalog}) is a finite set of Datalog rules \( r \) of the form

\[
h(x) \leftarrow b_1(\vec{x}_1), b_2(\vec{x}_2), \ldots, b_n(\vec{x}_n).
\]

The semantics is defined by the \textit{immediate consequence operator} \( \mathcal{T}_\mathcal{P} \).

\textbf{Example:}

\[
\begin{align*}
\text{Reach}(x) & \leftarrow \text{Red}(x) \\
\text{Reach}(x) & \leftarrow \text{Reach}(y), E(y, x)
\end{align*}
\]

\[
\mathcal{T}^4_\mathcal{P}(G) = \mathcal{T}^3_\mathcal{P}(G) \cup \{ \text{Reach}(f) \}
\]
Monadic Datalog

A program $\mathcal{P}$ in *monadic Datalog* (for short: *mDatalog*) is a finite set of Datalog rules $r$ of the form

$$ h(x) \leftarrow b_1(\vec{x}_1), b_2(\vec{x}_2), \ldots, b_n(\vec{x}_n). $$

The semantics is defined by the *immediate consequence operator* $T_\mathcal{P}$.

**Example:**

$$ \text{Reach}(x) \leftarrow \text{Red}(x) $$
$$ \text{Reach}(x) \leftarrow \text{Reach}(y), \ E(y,x) $$

$$ T^5_\mathcal{P}(G) = T^4_\mathcal{P}(G) \cup \{ \text{Reach}(h), \text{Reach}(i) \} $$
A program \( \mathcal{P} \) in *monadic Datalog* (for short: *mDatalog*) is a finite set of Datalog rules \( r \) of the form

\[
h(x) \leftarrow b_1(\vec{x}_1), b_2(\vec{x}_2), \ldots, b_{n_r}(\vec{x}_{n_r}).
\]

The semantics is defined by the *immediate consequence operator* \( \mathcal{T}_\mathcal{P} \).

**Example:**

\[
\begin{align*}
\text{Reach}(x) & \leftarrow \text{Red}(x) \\
\text{Reach}(x) & \leftarrow \text{Reach}(y), \ E(y,x)
\end{align*}
\]

\[
\mathcal{T}^6_\mathcal{P}(G) = \mathcal{T}^5_\mathcal{P}(G) =: \mathcal{T}^\omega_\mathcal{P}(G)
\]
Monadic Datalog

A program \( P \) in \textit{monadic Datalog} (for short: \textit{mDatalog}) is a finite set of Datalog rules \( r \) of the form

\[
h(x) \leftarrow b_1(\vec{x}_1), \ b_2(\vec{x}_2), \ldots, \ b_n(\vec{x}_n_r).
\]

The semantics is defined by the \textit{immediate consequence operator} \( \mathcal{T}_P \).

\textbf{Example:}

\[
\begin{align*}
\text{Reach}(x) & \leftarrow \text{Red}(x) \\
\text{Reach}(x) & \leftarrow \text{Reach}(y), \ E(y,x)
\end{align*}
\]

\[
\mathcal{T}_P^6(G) = \mathcal{T}_P^5(G) =: \mathcal{T}_P^\omega(G)
\]

Query: \( Q = (P, P), \ P \in \text{idb}(P), \ Q(A) := \{ a \mid P(a) \in \mathcal{T}_P^\omega(A) \} \)
Monadic Datalog

A program $\mathcal{P}$ in *monadic Datalog* (for short: *mDatalog*) is a finite set of Datalog rules $r$ of the form

$$h(x) \leftarrow b_1(\overrightarrow{x}_1), b_2(\overrightarrow{x}_2), \ldots, b_n(\overrightarrow{x}_n).$$

The semantics is defined by the *immediate consequence operator* $\mathcal{T}_\mathcal{P}$.

**Example:**

$$\text{Reach}(x) \leftarrow \text{Red}(x)$$
$$\text{Reach}(x) \leftarrow \text{Reach}(y), \ E(y,x)$$

$$\mathcal{T}_\mathcal{P}^6(G) = \mathcal{T}_\mathcal{P}^5(G) =: \mathcal{T}_\mathcal{P}^\omega(G)$$

Query: $Q = (\mathcal{P}, P), \ P \in \text{idb}(\mathcal{P}), \ Q(A) := \{a \mid P(a) \in \mathcal{T}_\mathcal{P}^\omega(A)\}$

$$Q = (\mathcal{P}, \text{Reach}), \ Q(G) = \{a, b, c, d, f, h, i\}$$
Σ-labeled Trees
$\Sigma$-labeled Trees
Σ-labeled Trees
$\Sigma$-labeled Trees

$\Sigma$: finite, unranked alphabet
Σ-labeled Trees

Σ: finite, unranked alphabet

- \text{label}_\alpha(x): \text{ node } x \text{ carries label } \alpha \in \Sigma
Σ-labeled Trees

Σ: finite, unranked alphabet

- $\text{label}_\alpha(x)$: node $x$ carries label $\alpha \in \Sigma$

Unordered trees

- $\text{child}(x, y)$: $y$ is child of $x$
- $\text{root}(x)$: node $x$ is the root
- $\text{leaf}(x)$: node $x$ is a leaf
- $\text{desc}(x, y)$: $y$ is descendant of $x$
Σ-labeled Trees

Σ: finite, unranked alphabet

- **label**\(\alpha(x)\): node \(x\) carries label \(\alpha \in \Sigma\)

Unordered trees

- **child**\((x, y)\): \(y\) is child of \(x\)
- **root**\((x)\): node \(x\) is the root
- **leaf**\((x)\): node \(x\) is a leaf
- **desc**\((x, y)\): \(y\) is descendant of \(x\)
$\Sigma$-labeled Trees

$\Sigma$: finite, unranked alphabet

- $\text{label}_\alpha(x)$: node $x$ carries label $\alpha \in \Sigma$

Unordered trees

- $\text{child}(x, y)$: $y$ is child of $x$
- $\text{root}(x)$: node $x$ is the root
- $\text{leaf}(x)$: node $x$ is a leaf
- $\text{desc}(x, y)$: $y$ is descendant of $x$

Ordered trees

- $\text{fc}(x, y)$: $y$ is the first child of $x$
- $\text{ns}(x, y)$: $y$ is the next sibling of $x$
- $\text{ls}(x)$: $x$ is the last sibling
- $\text{root}(x), \text{leaf}(x), \text{child}(x, y), \text{desc}(x, y)$
Query Containment Problem (QCP)

Let $\tau$ be a schema for $\Sigma$-labeled trees.
Let $Q_1$ and $Q_2$ be queries in mDatalog($\tau$). Then we say:

$$Q_1 \subseteq Q_2,$$  iff  $$Q_1(T) \subseteq Q_2(T)$$  for every $\Sigma$-labeled tree $T$

**QCP for unary queries in mDatalog($\tau$) on $\Sigma$-labeled trees**

*Input:* Queries $Q_1$ and $Q_2$ in mDatalog($\tau$).

*Output:* Yes, if $Q_1 \subseteq Q_2$,
No, otherwise.
Results

Previously known:

Containment of mDatalog over arbitrary finite structures:

Results

Previously known:

Containment of mDatalog over arbitrary finite structures:


Gottlob/Koch (2002): mDatalog($\tau_{GK}$) on ordered $\Sigma$-labeled trees is $\text{EXPTIME}$-hard and decidable.
Results

Previously known:

Containment of mDatalog over arbitrary finite structures:


Gottlob/Koch (2002): \( \tau_{GK} : fc, ns, ls, root, leaf, (label_\alpha)_{\alpha \in \Sigma} \)

mDatalog(\( \tau_{GK} \)) on ordered \( \Sigma \)-labeled trees is EXPTIME-hard and decidable.
Results

Theorem:
\[ \tau_u : \text{child}, (\text{label}_\alpha)_{\alpha \in \Sigma} \]

The \text{QCP} for Boolean \text{mDatalog}(\tau_u) on unordered \Sigma-labeled trees is \text{EXPTIME-hard}.

Corollary:
\[ \tau_o : \text{fc}, \text{ns}, (\text{label}_\alpha)_{\alpha \in \Sigma} \]

The \text{QCP} for Boolean \text{mDatalog}(\tau_o) on ordered \Sigma-labeled trees is \text{EXPTIME-hard}.

Previously known:

Containment of \text{mDatalog} over arbitrary finite structures:

- Cosmadakis et al (1988): \text{EXPTIME-hard} and in \text{2EXPTIME}.
- Gottlob/Koch (2002): \text{mDatalog}(\tau_{\text{GK}}) on ordered \Sigma-labeled trees is \text{EXPTIME-hard} and decidable.
Results

Theorem: $\tau_u : \text{child}, (\text{label}_{\alpha})_{\alpha \in \Sigma}$

The QCP for Boolean mDatalog($\tau_u$) on unordered $\Sigma$-labeled trees is EXPTIME-hard.

Corollary: $\tau_o : \text{fc}, \text{ns}, (\text{label}_{\alpha})_{\alpha \in \Sigma}$

The QCP for Boolean mDatalog($\tau_o$) on ordered $\Sigma$-labeled trees is EXPTIME-hard.

Theorem: $\tau_{\text{child}}^{\text{GK}} : \text{fc}, \text{ns}, \text{ls}, \text{child}, \text{root}, \text{leaf}, (\text{label}_{\alpha})_{\alpha \in \Sigma}$

The QCP for unary mDatalog($\tau_{\text{child}}^{\text{GK}}$) on ordered $\Sigma$-labeled trees belongs to EXPTIME.

Corollary: $\tau_{u}^{\text{root,leaf}} : \text{child}, \text{root}, \text{leaf}, (\text{label}_{\alpha})_{\alpha \in \Sigma}$

The QCP for unary mDatalog($\tau_{u}^{\text{root,leaf}}$) on unordered $\Sigma$-labeled trees belongs to EXPTIME.
Sketching the Proof of the 2nd Theorem

Theorem:
The QCP for unary mDatalog($\tau_{GK}^{\text{child}}$) on ordered $\Sigma$-labeled trees belongs to \text{EXPTIME}.

Proof (sketch):
Given: unary $Q_1$ and $Q_2$ in mDatalog($\tau_{GK}^{\text{child}}$).
Question: decide, whether $Q_1 \subseteq Q_2$
Use the \textit{automata-theoretic method}:
Sketching the Proof of the 2nd Theorem

Theorem:

The QCP for unary mDatalog($\tau_{GK}^{\text{child}}$) on ordered $\Sigma$-labeled trees belongs to $\text{EXPTIME}$.

Proof (sketch):

Given: unary $Q_1$ and $Q_2$ in mDatalog($\tau_{GK}^{\text{child}}$).

Question: decide, whether $Q_1 \subseteq Q_2$

Use the automata-theoretic method:

(1) $Q_1, Q_2 \rightsquigarrow$ Boolean queries $Q'_1, Q'_2$ on binary trees, such that

$$Q_1 \subseteq Q_2 \iff Q'_1 \subseteq Q'_2$$
Sketching the Proof of the 2nd Theorem

Theorem:
The QCP for unary mDatalog($\tau_{GK}^{\text{child}}$) on ordered $\Sigma$-labeled trees belongs to $\text{EXPTIME}$.

Proof (sketch):
Given: unary $Q_1$ and $Q_2$ in mDatalog($\tau_{GK}^{\text{child}}$).
Question: decide, whether $Q_1 \subseteq Q_2$
Use the automata-theoretic method:
(1) $Q_1, Q_2 \mapsto$ Boolean queries $Q_1', Q_2'$ on binary trees, such that
$$Q_1 \subseteq Q_2 \iff Q_1' \subseteq Q_2'$$
(2) $Q_1', Q_2' \mapsto$ tree automata $A_{\text{yes}}^1$ and $A_{\text{no}}^2$, such that
$A_{\text{yes}}^1$ accepts $T \iff Q_1'(T) = \text{yes}$ and $A_{\text{no}}^2$ accepts $T \iff Q_2'(T) = \text{no}$
Sketching the Proof of the 2nd Theorem

Theorem:
The QCP for unary mDatalog($\tau^\text{child}_{GK}$) on ordered $\Sigma$-labeled trees belongs to EXPTIME.

Proof (sketch):
Given: unary $Q_1$ and $Q_2$ in mDatalog($\tau^\text{child}_{GK}$).
Question: decide, whether $Q_1 \subseteq Q_2$
Use the automata-theoretic method:
(1) $Q_1, Q_2 \Rightarrow$ Boolean queries $Q'_1, Q'_2$ on binary trees, such that
\[ Q_1 \subseteq Q_2 \iff Q'_1 \subseteq Q'_2 \]
(2) $Q'_1, Q'_2 \Rightarrow$ tree automata $A^\text{yes}_1$ and $A^\text{no}_2$, such that
\[ A^\text{yes}_1 \text{ accepts } T \iff Q'_1(T) = \text{yes} \quad \text{and} \quad A^\text{no}_2 \text{ accepts } T \iff Q'_2(T) = \text{no} \]
(3) Construct the product automaton $B : \mathcal{L}(B) = \mathcal{L}(A^\text{yes}_1) \cap \mathcal{L}(A^\text{no}_2)$.
Test: $\mathcal{L}(B) = \emptyset$ ?

Note that $\mathcal{L}(B) \neq \emptyset$ if, and only if, $Q_1 \nsubseteq Q_2$.  

A. Frochoux, M. Grohe, N. Schweikardt  

Monadc Datalog Containment on Trees  

6/8
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A_1^{\text{yes}}$: $A_1^{\text{yes}}$ accepts $T \iff Q'_1(T) = \text{yes}$.

(2b) Construct tree automaton $A_2^{\text{no}}$: $A_2^{\text{no}}$ accepts $T \iff Q'_2(T) = \text{no}$.
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A_1^{\text{yes}}$: $A_1^{\text{yes}}$ accepts $T \iff Q'_1(T) = \text{yes}$.

(2b) Construct tree automaton $A_2^{\text{no}}$: $A_2^{\text{no}}$ accepts $T \iff Q'_2(T) = \text{no}$.

Translate $Q'_2$ into equivalent MSO-sentence $\varphi_2$ of the form

$$\forall X_1 \cdots \forall X_n \exists z_1 \cdots \exists z_\ell \bigvee_{j=1}^m \xi_j$$
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A_1^{\text{yes}}$: $A_1^{\text{yes}}$ accepts $T \iff Q'_1(T) = \text{yes}$.

(2b) Construct tree automaton $A_2^{\text{no}}$: $A_2^{\text{no}}$ accepts $T \iff Q'_2(T) = \text{no}$.

Translate $Q'_2$ into equivalent MSO-sentence $\varphi_2$ of the form

\[
\forall X_1 \cdots \forall X_n \exists Z_1 \cdots \exists Z_\ell \bigvee_{j=1}^{m} \xi_j
\]

\[
\neg \varphi_2 \equiv \exists X_1 \cdots \exists X_n \neg \exists Z_1 \cdots \exists Z_\ell \bigvee_{j=1}^{m} \xi_j
\]
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton \( A_1^{\text{yes}} \): \( A_1^{\text{yes}} \) accepts \( T \iff Q_1'(T) = \text{yes} \).

(2b) Construct tree automaton \( A_2^{\text{no}} \): \( A_2^{\text{no}} \) accepts \( T \iff Q_2'(T) = \text{no} \).

Translate \( Q_2' \) into equivalent MSO-sentence \( \varphi_2 \) of the form

\[
\forall X_1 \cdots \forall X_n \exists Z_1 \cdots \exists Z_\ell \bigvee_{j=1}^{m} \xi_j
\]

\[
\neg \varphi_2 \equiv \exists X_1 \cdots \exists X_n \neg \exists Z_1 \cdots \exists Z_\ell \bigvee_{j=1}^{m} \xi_j
\]

If query in TMNF (cf., Gottlob, Koch: PODS 2002)

\( \leadsto \) construction of \( A_2^{\text{no}} \) in 1-fold exponential time
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A_1^{\text{yes}}$: $A_1^{\text{yes}}$ accepts $T \iff Q_1'(T) = \text{yes}$.

(2b) Construct tree automaton $A_2^{\text{no}}$: $A_2^{\text{no}}$ accepts $T \iff Q_2'(T) = \text{no}$.

Translate $Q_2'$ into equivalent MSO-sentence $\varphi_2$ of the form

$$
\forall X_1 \cdots \forall X_n \exists z_1 \cdots \exists z_\ell \bigvee_{j=1}^{m} \xi_j
$$

$$
\neg \varphi_2 \equiv \exists X_1 \cdots \exists X_n \neg \exists z_1 \cdots \exists z_\ell \bigvee_{j=1}^{m} \xi_j
$$

If query in TMNF (cf., Gottlob, Koch: PODS 2002)

$$
\leadsto \text{construction of } A_2^{\text{no}} \text{ in 1-fold exponential time}
$$

Problem:

By following this approach to construct $A_1^{\text{yes}}$, a second complementation leads to 2-fold exponential time.
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A_1^{yes}$: $A_1^{yes}$ accepts $T \iff Q'_1(T) = yes$.

(2b) Construct tree automaton $A_2^{no}$: $A_2^{no}$ accepts $T \iff Q'_2(T) = no$.

Key idea:

Boolean TMNF-query $Q'_1 \leadsto$ two-way alternating tree automaton (2ATA) $\hat{A}_1^{yes}$

(in polynomial time)
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A_1^{\text{yes}}$: $A_1^{\text{yes}}$ accepts $T$ $\iff$ $Q'_1(T) = \text{yes}$.

(2b) Construct tree automaton $A_2^{\text{no}}$: $A_2^{\text{no}}$ accepts $T$ $\iff$ $Q'_2(T) = \text{no}$.

Key idea:

Boolean TMNF-query $Q'_1$ $\rightsquigarrow$ two-way alternating tree automaton (2ATA) $\hat{A}_1^{\text{yes}}$ (in polynomial time)

$\hat{A}_1^{\text{yes}}$ $\rightsquigarrow$ tree automaton $A_1^{\text{yes}}$ (in 1-fold exponential time)

Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A^{yes}$ accepts $T$ $\iff$ $Q'(T) = yes$

(2b) Construct tree automaton $A^{no}$ accepts $T$ $\iff$ $Q'(T) = no$

Key idea:
Boolean TMNF-query $Q'$ $\mapsto$ two-way alternating tree automaton (2ATA) $\hat{A}^{yes}$ (in polynomial time)
$\hat{A}^{yes}$ $\mapsto$ tree automaton $A^{yes}$ (in 1-fold exponential time)


$\delta(X, c) = ((Y, \text{stay}) \land (Z, \text{stay})) \lor \ldots \lor ((\text{is\_lc}, \text{stay}) \land (Y, \text{up}))$
Sketch (cont.): Zoom into Step (2)

(2a) Construct tree automaton $A_{yes}^1$ accepts $T$ iff $Q'_{yes}^1(T) = yes$.

(2b) Construct tree automaton $A_{no}^2$ accepts $T$ iff $Q'_{no}^2(T) = no$.

Key idea:

Boolean TMNF-query $Q'$ \(\Rightarrow\) two-way alternating tree automaton (2ATA) $\hat{A}_{yes}^1$ (in polynomial time) \(\Rightarrow\) tree automaton $A_{yes}^1$ (in 1-fold exponential time) (implicit in Vardi: ICALP'98 / Maneth, Friese, Seidl: 2010)

\[
X(x) \leftarrow \text{lc}(y, x), \ Y(y) \\
\vdots \\
X(x) \leftarrow Y(x), \ Z(x)
\]

\[
\delta(X, c) = (((Y, \text{stay}) \land (Z, \text{stay})) \lor \ldots \lor (\text{is\_lc}, \text{stay}) \land (Y, \text{up})))
\]
Final Remarks

Let $\tau$ be a schema for $\Sigma$-labeled trees and let $\text{desc} \in \tau$.

**Theorem:**

The QCP for unary mDatalog$(\tau)$ on $\Sigma$-labeled trees can be solved in 2-fold exponential time.
Let $\tau$ be a schema for $\Sigma$-labeled trees and let $\text{desc} \in \tau$.

**Theorem:**

The QCP for unary mDatalog($\tau$) on $\Sigma$-labeled trees can be solved in 2-fold exponential time.

**Current work:**

- Close the gap between the EXPTIME lower and the 2EXPTIME upper bound for the case where the descendant-axis is involved.

Thank you for your attention!
Final Remarks

Let $\tau$ be a schema for $\Sigma$-labeled trees and let $\text{desc} \in \tau$.

**Theorem:**
The QCP for unary $\text{mDatalog}(\tau)$ on $\Sigma$-labeled trees can be solved in 2-fold exponential time.

**Current work:**
- Close the gap between the $\text{EXPTIME}$ lower and the $2\text{EXPTIME}$ upper bound for the case where the descendant-axis is involved
- Extend the results to related problems like Emptiness and Equivalence.
Let $\tau$ be a schema for $\Sigma$-labeled trees and let $\text{desc} \in \tau$.

**Theorem:**
The QCP for unary mDatalog$(\tau)$ on $\Sigma$-labeled trees can be solved in 2-fold exponential time.

**Current work:**
- Close the gap between the EXPTIME lower and the 2EXPTIME upper bound for the case where the descendant-axis is involved
- Extend the results to related problems like Emptiness and Equivalence.

Thank you for your attention!