Worst-Case Optimal Querying of Very Expressive Description Logics with Path Expressions and Succinct Counting

Bartosz Bednarczyk & Sebastian Rudolph
firstname.lastname@tu-dresden.de

Technische Universität Dresden and University of Wrocław

IJCAI 2019
Macau, August 15th, 2019
Running example (ZOIQ KB)

Database

Knowledge

\[\text{HasParent} \ (\text{Heracles, Zeus})\]
\[\text{HasParent} \ (\text{Perseus, Zeus})\]
\[\text{male} \ (\text{Zeus})\]
\[\text{deity} \ (\text{Zeus})\]
\[\text{mortal} \ (\text{Alcmene})\]
Running example (ZOIQ KB)

**Database**

- **HasParent** (Heracles, Zeus)
- **HasParent** (Perseus, Zeus)
  - **male** (Zeus)
- **deity** (Zeus)
- **mortal** (Alcmene)

**Knowledge**

- **mortal** ⊑ ¬**deity**
- **⊤** ⊑ ∃**HasFather.male** ∩ ∃**HasMother.female**
- **HasParent** ≡ **HasMother** ∪ **HasFather**
- ∀**HasParent.mortal** ⊑ **mortal**
- **deity** ⊑ ∀**HasParent**.**deity**
Positive 2-Way Regular Path Query

$$\exists x, y, z \ (\text{HasParent}^* \circ \text{HasParent}^-^*)(x, y) \land \text{HasParent}(z, x) \land \text{HasParent}(z, y)$$

\(x \text{ and } y \) are relatives with a common children \(z\)
Positive 2-Way Regular Path Query

\[ \exists x, y, z \ (\text{HasParent}^* \circ \text{HasParent}^* )(x, y) \land \text{HasParent}(z, x) \land \text{HasParent}(z, y) \]

\( x \) and \( y \) are relatives with a common children \( z \)

An example match \( \pi \) in a model \( \mathcal{I} \):

\[ \pi(x) = \text{Amphitrite}^\mathcal{I} \quad \pi(y) = \text{Poseidon}^\mathcal{I} \quad \pi(z) = \text{Triton}^\mathcal{I} \]
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
- Simple roles: HasSon,
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
- Simple roles: HasSon, Quantifiers: ⊤ ⊑ ∃HasFather.male

Even more expressive logics: ZIQ, ZOQ and ZOI.
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
- Simple roles: HasSon, Quantifiers: ⊤ ⊆ ∃HasFather.male
- Boolean role combinations b: HasParent ≡ HasMom ∪ HasDad
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
- Simple roles: HasSon, Quantifiers: ⊤ ⊑ ∃HasFather.male
- Boolean role combinations b: HasParent ≡ HasMom ∪ HasDad
- Regular expressions reg: Relatives ≡ HasParent* ◦ HasParent−*

An example logic employing all these features is called \( \mathcal{Z} \).
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
- Simple roles: HasSon,  Quantifiers: ⊤ ⊑ ∃HasFather.male
- Boolean role combinations b: HasParent ≡ HasMom ∪ HasDad
- Regular expressions reg: Relatives ≡ HasParent* ∘ HasParent−−

An example logic employing all these features is called Z.

- Inverses I: HasChildren ≡ HasParent−
Important logical features

- Unary concepts: male, diety, \( \neg \) mortal + a little more
- Simple roles: HasSon, Quantifiers: \( \top \subseteq \exists \text{HasFather.male} \)
- Boolean role combinations \( b \): HasParent \( \equiv \) HasMom \( \cup \) HasDad
- Regular expressions \( \text{reg} \): Relatives \( \equiv \) HasParent\(^*\) \( \circ \) HasParent\(^{-*}\)

An example logic employing all these features is called \( \mathcal{Z} \).

- Inverses \( \mathcal{I} \): HasChildren \( \equiv \) HasParent\(^-\)
- Nominals (constants) \( \mathcal{O} \): \{Zeus\}
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
- Simple roles: HasSon, Quantifiers: \( \top \sqsubseteq \exists \text{HasFather}.\text{male} \)
- Boolean role combinations \( b \): HasParent \( \equiv \) HasMom \( \cup \) HasDad
- Regular expressions \( \text{reg} \): Relatives \( \equiv \) HasParent* \circ \) HasParent**

An example logic employing all these features is called \( \mathcal{Z} \).

- Inverses \( \mathcal{I} \): HasChildren \( \equiv \) HasParent\(^{-}\)
- Nominals (constants) \( \mathcal{O} \): \{Zeus\}
- Counting \( \mathcal{Q} \): \{Zeus\} \( \sqsubseteq (\geq 100 \text{HasChildren}).\top \)
Important logical features

- Unary concepts: male, diety, ¬mortal + a little more
- Simple roles: HasSon, Quantifiers: \( \top \sqsubseteq \exists \text{HasFather.male} \)
- Boolean role combinations \( b: \text{HasParent} \equiv \text{HasMom} \cup \text{HasDad} \)
- Regular expressions \( \text{reg}: \text{Relatives} \equiv \text{HasParent}^* \circ \text{HasParent}^- \)

An example logic employing all these features is called \( \mathcal{Z} \).

- Inverses \( I: \text{HasChildren} \equiv \text{HasParent}^- \)
- Nominals (constants) \( O: \{ \text{Zeus} \} \)
- Counting \( Q: \{ \text{Zeus} \} \sqsubseteq (\geq 100 \text{HasChildren}). \top \)

Extensions of \( \mathcal{Z} \)

Even more expressive logics: \( \mathcal{Z}IQ, \mathcal{ZOQ} \) and \( \mathcal{ZOI} \)
Quasi-forest model property (QFMP)
Quasi-forest model property (QFMP)

Logics $\mathcal{ZIQ}$, $\mathcal{ZOQ}$ and $\mathcal{ZOI}$ enjoy QFMP. SAT "only" ExpTime-complete (with bin encoding of $\mathcal{Q}$).
Querying \( \mathcal{Z} \) with P2RPQs (existing results)

P2RPQ entailment for \( \mathcal{Z} \) family [Calvanese et al, IJCAI’09]

Testing P2RPQ entailment for \( \mathcal{ZI}, \mathcal{ZO}, \mathcal{ZO} \) can be done in 3ExpTime (2ExpTime-c. under unary encoding).
Querying $\mathcal{Z}$ with P2RPQs (existing results)

P2RPQ entailment for $\mathcal{Z}$ family [Calvanese et al, IJCAI’09]

Testing P2RPQ entailment for $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, $\mathcal{ZOI}$ can be done in $3\text{ExpTime}$ ($2\text{ExpTime}$-c. under unary encoding).

- Quite complicated...
- Heavy machinery on automata theory...
Querying $\mathcal{Z}$ with P2RPQs (our results)

**P2RPQ entailment for $\mathcal{Z}$ family** [Calvanese et al, IJCAI’09]

Testing P2RPQ entailment for $\mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI}$ can be done in 3ExpTime (2ExpTime-c. under unary encoding).
Querying $\mathcal{Z}$ with P2RPQs (our results)

Testing P2RPQ entailment for $\mathcal{Z}_{IQ}, \mathcal{Z}_{OQ}, \mathcal{Z}_{OI}$ can be done in $3\text{ExpTime}$ ($2\text{ExpTime}$-c. under unary encoding).

P2RPQ entailment for $\mathcal{Z}$ family [this paper!]

P2RPQ entailment for $\mathcal{Z}_{IQ}, \mathcal{Z}_{OQ}, \mathcal{Z}_{OI}$ is $2\text{ExpTime}$-c, even under binary encoding. Moreover once the number of atoms in the query is bounded, entailment is in $\text{ExpTime}$. 

Reduction to satisfiability (works under binary encoding)

P2RPQ version of so-called “rolling-up” technique used for CQs

Match calculus
Querying \( \mathcal{Z} \) with P2RPQs (our results)

- P2RPQ entailment for \( \mathcal{Z} \) family [Calvanese et al, IJCAI’09]

  Testing P2RPQ entailment for \( \mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI} \) can be done in 3ExpTime (2ExpTime-c. under unary encoding).

- P2RPQ entailment for \( \mathcal{Z} \) family [this paper!]

  P2RPQ entailment for \( \mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI} \) is 2ExpTime-c, even under binary encoding. Moreover once the number of atoms in the query is bounded, entailment is in ExpTime.

- Reduction to satisfiability (works under binary enc)
- P2RPQ version of so-called ”rolling-up” technique used for CQs
- Simulate automata on quasi-forest-models = Match calculus
Querying $\mathcal{Z}$ with P2RPQs (our results)

P2RPQ entailment for $\mathcal{Z}$ family [this paper!]

P2RPQ entailment for $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, $\mathcal{ZOI}$ is $2\text{ExpTime-c}$. 
Querying $\mathcal{Z}$ with P2RPQs (our results)

- P2RPQ entailment for $\mathcal{Z}$ family [this paper!]

P2RPQ entailment for $\mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI}$ is 2ExpTime-c.

- For simplicity we assume that the input query $q$ is C2RPQ
Querying $\mathcal{Z}$ with P2RPQs (our results)

P2RPQ entailment for $\mathcal{Z}$ family [this paper!]

P2RPQ entailment for $\mathcal{ZI}_Q, \mathcal{ZO}_Q, \mathcal{ZO}_I$ is 2ExpTime-c.

- For simplicity we assume that the input query $q$ is C2RPQ
- $q$ can be represented as a set of NFAs without $\varepsilon$-transitions
Querying $\mathcal{Z}$ with P2RPQs (our results)

- For simplicity we assume that the input query $q$ is C2RPQ
- $q$ can be represented as a set of NFAs without $\varepsilon$-transitions
- We annotate models $\mathcal{I}$ with $Q_M$ predicates (rolling-up):
  - $Q_M$ indicate that there is a match $M$ in $\mathcal{I}$
Querying \( \mathcal{Z} \) with P2RPQs (our results)

- **P2RPQ entailment for \( \mathcal{Z} \) family [this paper!]**

- P2RPQ entailment for \( \mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI} \) is 2ExpTime-c.

- For simplicity we assume that the input query \( q \) is C2RPQ

- \( q \) can be represented as a set of NFAs without \( \varepsilon \)-transitions

- We annotate models \( \mathcal{I} \) with \( Q_M \) predicates (rolling-up):
  - \( Q_M \) indicate that there is a match \( M \) in \( \mathcal{I} \)
  - basically we simulate automaton on quasi-forest models.

\[
\begin{align*}
\mathcal{K} & \xrightarrow{\text{algorithm}} \mathcal{K} \\
\mathcal{K} & \xrightarrow{\text{poly-size}} \mathcal{K} + \mathcal{K}_q + Q_q \sqsubseteq \bot \\
\end{align*}
\]

Generated with match calculus
Querying $\mathcal{Z}$ with P2RPQs (our results)

P2RPQ entailment for $\mathcal{Z}$ family [this paper!]

P2RPQ entailment for $\mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI}$ is 2ExpTime-c.

\[ \mathcal{K} + \mathcal{K}_q + Q_q \sqsubseteq \perp \]

\[ \mathcal{K}_{\neg q} \]

- $\mathcal{K} \models q$ iff $\mathcal{K}_{\neg q}$ is unsatisfiable
Querying $\mathcal{Z}$ with P2RPQs (our results)

- P2RPQ entailment for $\mathcal{Z}$ family [this paper!]
- P2RPQ entailment for $\mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI}$ is 2ExpTime-c.

\[ \mathcal{K} \quad \square \quad \mathcal{K}_q \quad \square \quad Q_q \subseteq \perp \]

**\(\mathcal{K}_{\neg q}\)**

- \(\mathcal{K} \models q\) iff \(\mathcal{K}_{\neg q}\) is unsatisfiable
- Obtained KB \(\mathcal{K}_{\neg q}\) is only exp in \(|q|\) and poly in \(|\mathcal{K}|\)
Querying \( \mathcal{Z} \) with P2RPQs (our results)

P2RPQ entailment for \( \mathcal{Z} \) family [this paper!]

P2RPQ entailment for \( \mathcal{ZI}_Q, \mathcal{ZO}_Q, \mathcal{ZO}_I \) is 2ExpTime-c.

\[
\begin{align*}
\mathcal{K} & \quad \text{poly-size} \\
\mathcal{K}_q & \quad \text{exp-size} \\
\mathcal{Q}_q \sqsubseteq \bot & \quad \text{poly-size}
\end{align*}
\]

\[\mathcal{K}_{\neg q}\]

- \( \mathcal{K} \models q \) iff \( \mathcal{K}_{\neg q} \) is unsatisfiable
- Obtained KB \( \mathcal{K}_{\neg q} \) is only exp in \(|q|\) and poly in \(|\mathcal{K}|\)
- Testing unsatisfiability can be done in ExpTime w.r.t \(|\mathcal{K}_{\neg q}|\)
Querying $\mathcal{Z}$ with P2RPQs (our results)

- **P2RPQ entailment for $\mathcal{Z}$ family [this paper!]**
- P2RPQ entailment for $\mathcal{ZI}Q, \mathcal{ZO}Q, \mathcal{ZO}I$ is $2\text{ExpTime}$-c.

\[\mathcal{K} \sqcup \mathcal{K}_q \sqcup Q_q \sqsubseteq \bot\]

- $\mathcal{K} \models q$ iff $\mathcal{K}_{\neg q}$ is unsatisfiable
- Obtained KB $\mathcal{K}_{\neg q}$ is only exp in $|q|$ and poly in $|\mathcal{K}|$
- Testing unsatisfiability can be done in ExpTime w.r.t. $|\mathcal{K}_{\neg q}|$
- Thus in $2\text{ExpTime}$ w.r.t. $|\mathcal{K}| + |q|$
Applications to other logics

We presented a reduction form $\mathcal{GC}^2$ to $\mathcal{ZIQ}$, hence we conclude:

Positive regular path queries in $\mathcal{GC}^2$

P2RPQ entailment for $\mathcal{GC}^2$ is 2ExpTime-complete.
Applications to other logics

We presented a reduction form $\mathcal{GC}^2$ to $\mathcal{ZIQ}$, hence we conclude:

Positive regular path queries in $\mathcal{GC}^2$

P2RPQ entailment for $\mathcal{GC}^2$ is $2\text{ExpTime}$-complete.

By reusing exponential reduction from $\mathcal{SR}$ to $\mathcal{Z}$:

Positive regular path queries in $\mathcal{SR}$ family

P2RPQ entailment for $\mathcal{SR}(\mathcal{OI}, \mathcal{IQ}, \mathcal{OQ})$ is in $3\text{ExpTime}$. 
Applications to query containment

Testing query containment $\mathcal{K} \models q \subseteq q'$ is:
in \text{2ExpTime} for:
- $\mathcal{K}$ in $\mathcal{ZOQ}$ or $\mathcal{ZOI}$ and $q, q' \in \text{P2RPQ}$
- $\mathcal{K}$ in $\mathcal{ZIQ}$ and $q \in \text{P2RPQ}$, $q' \in \text{CQ}$
and in \text{3ExpTime} for:
- $\mathcal{K}$ in $\mathcal{SROQ}$ or $\mathcal{SROI}$ and $q, q' \in \text{P2RPQ}$
- $\mathcal{K}$ in $\mathcal{SRIQ}$ and $q \in \text{P2RPQ}$, $q' \in \text{CQ}$
Applications to query containment

Query containment

Testing query containment $\mathcal{K} \models q \subseteq q'$ is:
in 2ExpTime for:
- $\mathcal{K}$ in $\mathcal{ZOQ}$ or $\mathcal{ZOI}$ and $q, q' \in \text{P2RPQ}$
- $\mathcal{K}$ in $\mathcal{ZIQ}$ and $q \in \text{P2RPQ}$, $q' \in \text{CQ}$
and in 3ExpTime for:
- $\mathcal{K}$ in $\mathcal{SROQ}$ or $\mathcal{SROI}$ and $q, q' \in \text{P2RPQ}$
- $\mathcal{K}$ in $\mathcal{SRIQ}$ and $q \in \text{P2RPQ}$, $q' \in \text{CQ}$

Moreover once the number of the atoms from the query is bounded complexities of each problem drops by one exponential.
Conclusions and open problems

Our results

P2RPQ entailment for $ZIQ, ZOQ, ZOI$ is $2\text{ExpTime}$-c +
P2RPQ entailment for $SRIQ, SROQ, SROI$ in $3\text{Exp}$ +
P2RPQ containment in $2\text{ExpTime}$ +
One exp less for all problems when $\#\text{atoms}(q) \leq \text{Const.}$

Open problems
Conclusions and open problems

Our results

- P2RPQ entailment for $\mathcal{ZI}Q$, $\mathcal{ZO}Q$, $\mathcal{ZO}I$ is 2ExpTime-c +
- P2RPQ entailment for $\mathcal{SRI}Q, \mathcal{SRO}Q, \mathcal{SROI}$ in 3Exp +
- P2RPQ containment in 2ExpTime +
- One exp less for all problems when $\# \text{atoms}(q) \leq \text{Const.}$

Open problems

- Data complexity?
- Finite query entailment?
- Sat of $\mathcal{ZO}I Q$?
Conclusions and open problems

Our results

P2RPQ entailment for $\mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI}$ is $2\text{ExpTime-c} +$
P2RPQ entailment for $\mathcal{SRIQ}, \mathcal{SROQ}, \mathcal{SROI}$ in $3\text{Exp} +$
P2RPQ containment in $2\text{ExpTime} +$
One exp less for all problems when $\#\text{atoms}(q) \leq \text{Const.}$

Open problems

- Data complexity?
- Finite query entailment?
- Sat of $\mathcal{ZOIQ}$?