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

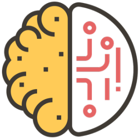
IJCAI 2019
Macau, August 15th, 2019

Running example ($\mathcal{ZOIQ}$ KB)

Database

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

Knowledge

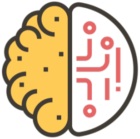


Running example ($\mathcal{ZOIQ}$ KB)

Database

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

Knowledge



$$\begin{aligned} mortal &\sqsubseteq \neg deity \\ \top &\sqsubseteq \exists HasFather.male \sqcap \exists HasMother.female \\ HasParent &\equiv HasMother \cup HasFather \\ \forall HasParent.mortal &\sqsubseteq mortal \\ deity &\sqsubseteq \forall HasParent^{*}.deity \end{aligned}$$

Positive 2-Way Regular Path Query

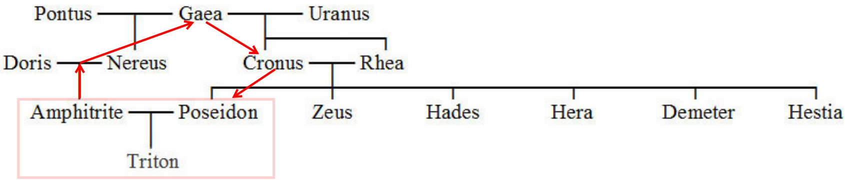
$$\exists x, y, z \; \underbrace{(\text{HasParent}^* \circ \text{HasParent}^{-^*})(x, y) \land \text{HasParent}(z, x) \land \text{HasParent}(z, y)}_{x \text{ and } y \text{ are relatives with a common children } z}$$

Positive 2-Way Regular Path Query

$$\exists x, y, z \; \underbrace{(\text{HasParent}^* \circ \text{HasParent}^{-^*})(x, y) \wedge \text{HasParent}(z, x) \wedge \text{HasParent}(z, y)}_{x \text{ and } y \text{ are relatives with a common children } z}$$

An example match π in a model $\mathcal{I}$:

$\pi(x) = \text{Amphitrite}^{\mathcal{I}}$ $\pi(y) = \text{Poseidon}^{\mathcal{I}}$ $\pi(z) = \text{Triton}^{\mathcal{I}}$



Important logical features

- Unary concepts: $\mathrm{male}, \mathrm{diety}, \neg\mathrm{mortal}$ + a little more

Important logical features

- Unary concepts: male, diety, $\neg$mortal + a little more
- Simple roles: HasSon,

Important logical features

- Unary concepts: male, diety, $\neg$mortal + a little more
- Simple roles: HasSon,   Quantifiers: $\top \sqsubseteq \exists$HasFather.male

Important logical features

- Unary concepts: $\mathrm{male}, \mathrm{diety}, \neg\mathrm{mortal}$ + a little more
- Simple roles: HasSon, Quantifiers: $\top \sqsubseteq \exists \mathrm{HasFather}.\mathrm{male}$
- Boolean role combinations b: $\mathrm{HasParent} \equiv \mathrm{HasMom} \cup \mathrm{HasDad}$

Important logical features

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

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: $\top \sqsubseteq \exists \mathrm{HasFather}.\mathrm{male}$
- Boolean role combinations b: $\mathrm{HasParent} \equiv \mathrm{HasMom} \cup \mathrm{HasDad}$
- Regular expressions reg: $\mathrm{Relatives} \equiv \mathrm{HasParent}^{*} \circ \mathrm{HasParent}^{-*}$

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

- Inverses $\mathcal{I}$: $\mathrm{HasChildren} \equiv \mathrm{HasParent}^{-}$

Important logical features

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

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

- Inverses $\mathcal{I}$: $\mathsf{HasChildren} \equiv \mathsf{HasParent}^{-}$
- Nominals (constants) $\mathcal{O}$: $\{\mathsf{Zeus}\}$

Important logical features

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

> An example logic employing all these features is called $\mathcal{Z}$.

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

Important logical features

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

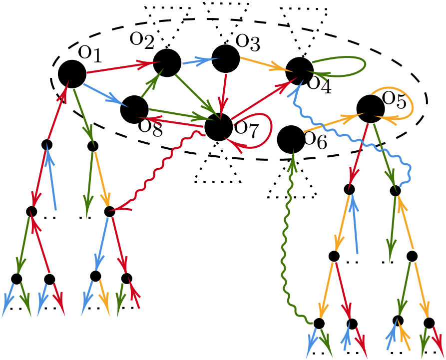
An example logic employing all these features is called $\mathcal{Z}$.

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

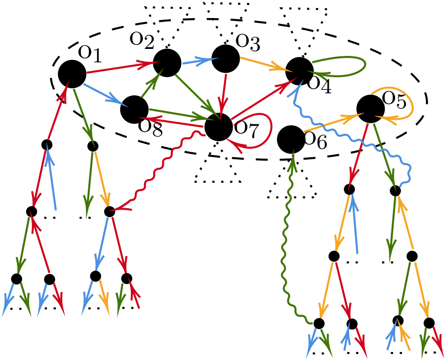
Extensions of $\mathcal{Z}$

Even more expressive logics: $\mathcal{ZIQ}$, $\mathcal{ZOQ}$ and $\mathcal{ZOI}$

Quasi-forest model property (QFMP)



Quasi-forest model property (QFMP)



QFMP for $\mathcal{Z}$ family [Calvanese et al, IJCAI'09]

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 $\mathcal{ZIQ}, \mathcal{ZOQ}, \mathcal{ZOI}$ 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 3ExpTime (2ExpTime-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)

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.

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

- For simplicity we assume that the input query q is C2RPQ
- q can be represented as a set of NFAs without ε-transitions

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.

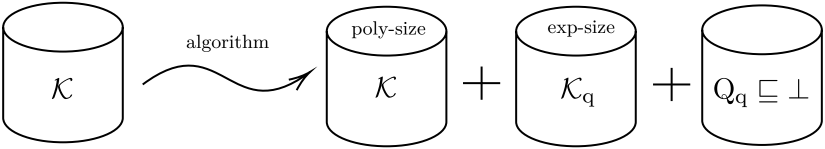
- For simplicity we assume that the input query q is C2RPQ
- q can be represented as a set of NFAs without ε-transitions
- We annotate models $\mathcal{I}$ with Q_M predicates (rolling-up):
  - Q_M indicate that there is a match M in $\mathcal{I}$

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

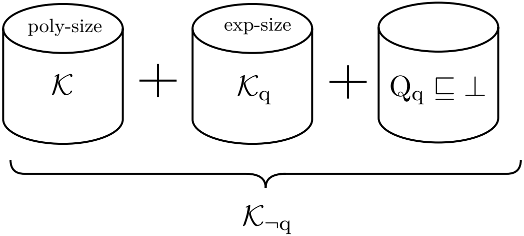


Generated with match calculus

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.

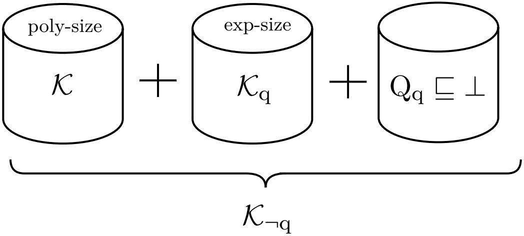


- $\mathcal{K} \models q$ iff $\mathcal{K}_{\neg q}$ is unsatisfiable

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.

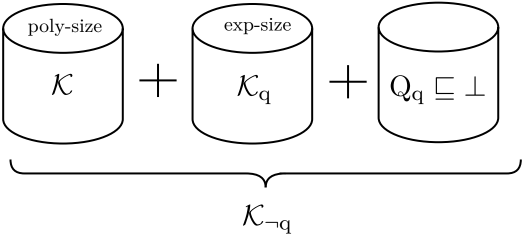


- $\mathcal{K} \models q$ iff $\mathcal{K}_{\neg q}$ is unsatisfiable
- Obtained KB $\mathcal{K}_{\neg q}$ is only exp in $|q|$ and poly in $|\mathcal{K}|$

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

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

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

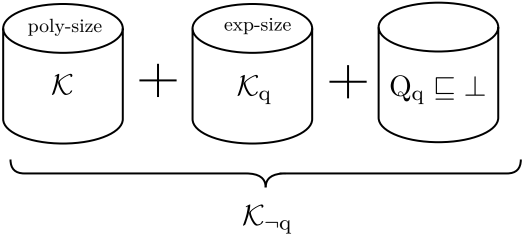


- $\mathcal{K} \models q$ iff $\mathcal{K}_{\neg q}$ is unsatisfiable
- Obtained KB $\mathcal{K}_{\neg q}$ is only exp in $|q|$ and poly in $|\mathcal{K}|$
- Testing unsatisfiability can be done in ExpTime w.r.t $|\mathcal{K}_{\neg q}|$

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

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

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



- $\mathcal{K} \models q$ 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 2ExpTime 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 2ExpTime-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 3ExpTime.

Applications to query containment

Query containment

Testing query containment $\mathcal{K} \models \mathrm{q} \subseteq \mathrm{q}'$ is:
in 2ExpTime for:

- $\mathcal{K}$ in $\mathcal{ZOQ}$ or $\mathcal{ZOI}$ and $\mathrm{q}, \mathrm{q}' \in$ P2RPQ
- $\mathcal{K}$ in $\mathcal{ZIQ}$ and $\mathrm{q} \in$ P2RPQ , $\mathrm{q}' \in$ CQ

and in 3ExpTime for:

- $\mathcal{K}$ in $\mathcal{SROQ}$ or $\mathcal{SROI}$ and $\mathrm{q}, \mathrm{q}' \in$ P2RPQ
- $\mathcal{K}$ in $\mathcal{SRIQ}$ and $\mathrm{q} \in$ P2RPQ , $\mathrm{q}' \in$ CQ

Applications to query containment

Query containment

Testing query containment $\mathcal{K} \models \mathrm{q} \subseteq \mathrm{q}'$ is:
in 2ExpTime for:

- $\mathcal{K}$ in $\mathcal{ZOQ}$ or $\mathcal{ZOI}$ and $\mathrm{q}, \mathrm{q}' \in$ P2RPQ
- $\mathcal{K}$ in $\mathcal{ZIQ}$ and $\mathrm{q} \in$ P2RPQ , $\mathrm{q}' \in$ CQ

and in 3ExpTime for:

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

Open problems

Conclusions and open problems

Our results

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

Open problems

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

Conclusions and open problems

Our results

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

Open problems

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