The Price of Selfishness
Conjunctive Query Entailment for $\mathcal{ALC}_{\text{Self}}$ is 2ExpTime-hard

19th of September 2021, DL Workshop 2021

Bartosz “Bart” Bednarczyk, Sebastian Rudolph

TU Dresden & University of Wrocław
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

The DL encompasses all these features is called $\mathcal{ALC}_{\text{Self}}$.

Conjunctive Queries: Give me IDs of all candidates who applied for "computer science".

$\phi(i) \Rightarrow \phi(i) = \exists n \exists s \text{Candidate}(i, n, s) \land \text{Appl}("\text{Computer Science"}, i)$

A knowledge base $K$ entails a conjunctive query $q$ (written: $K | q$) if $q$ matches all models of $K$.

Bartosz “Bart” Bednarczyk  The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

Knowledge (TBox)

Conjunctive Queries: Give me IDs of all candidates who applied for “computer science”.

$\phi(i) \rightarrow \phi(i) = \exists n \exists s \text{Candidate}(i, n, s) \land \text{Appl}(\text{"Computer Science"}, i)$

A knowledge base $K$ entails a conjunctive query $q$ (written: $K|_{=q}$) if $q$ matches all models of $K$.

Bartosz “Bart” Bednarczyk  The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

Knowledge (TBox)

$\text{hasParent}(\text{Heracles}, \text{Zeus})$
**Running example: Greek mythology** $\mathcal{ALC}_{Self}$ knowledge base

**Database (ABox)**

- $\text{hasParent}$(Heracles, Zeus)
- $\text{Diety}$(Zeus), $\text{Female}$(Rhea)

**Knowledge (TBox)**

- $\text{Male} \sqcap \exists \text{hasParent}$
- $\text{Female} \sqcap \exists \text{hasParent}$
- $\text{Narcissist} \sqsubseteq \exists \text{loves} \text{Self}$

The DL encompasses all these features is called $\mathcal{ALC}_{Self}$.

Conjunctive Queries: Give me IDs of all candidates who applied for "computer science".

$$\varphi(i) \Rightarrow \varphi(i) = \exists n \exists s \text{Candidate}(i, n, s) \land \text{Appl} \text{("Computer Science", i)}$$

A knowledge base $K$ entails a conjunctive query $q$ (written: $K | = q$) if $q$ matches all models of $K$.

Bartosz “Bart” Bednarczyk

The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{Self}$ is $2\text{ExpTime}$-hard
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

hasParent(Heracles, Zeus)

Diety(Zeus), Female(Rhea)

Narcissist(Narcissus)

Knowledge (TBox)

Conjunctive Queries: Give me IDs of all candidates who applied for "computer science".

$\phi(i) \mapsto \phi(i) = \exists n \exists s \text{Candidate}(i, n, s) \land \text{Appl}(\text{"Computer Science"}, i)$

A knowledge base $K$ entails a conjunctive query $q$ (written: $K \models q$) if $q$ matches all models of $K$.

The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

- hasParent(Heracles, Zeus)
- Diety(Zeus), Female(Rhea)
- Narcissist(Narcissus)

Knowledge (TBox)

- Mortal $\sqsubseteq \neg$Diety

A knowledge base $\mathcal{K}$ entails a conjunctive query $q$ (written: $\mathcal{K} | q$) if $q$ matches all models of $\mathcal{K}$. The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard.
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

- $\text{hasParent}$(Heracles, Zeus)
- $\text{Diety}$(Zeus), $\text{Female}$(Rhea)
- $\text{Narcissist}$(Narcissus)

Knowledge (TBox)

- $\text{Mortal} \sqsubseteq \neg \text{Diety}$
- $\top \sqsubseteq \exists \text{hasParent} \cdot \text{Male} \sqcap \exists \text{hasParent} \cdot \text{Female}$
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

$\text{hasParent}(\text{Heracles, Zeus})$

$\text{Diety}(\text{Zeus}), \text{Female}(\text{Rhea})$

$\text{Narcissist}(\text{Narcissus})$

Knowledge (TBox)

$\text{Mortal} \sqsubseteq \neg \text{Diety}$

$\top \sqsubseteq \exists \text{hasParent} \cdot \text{Male} \sqcap \exists \text{hasParent} \cdot \text{Female}$
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

- $\text{hasParent}(\text{Heracles, Zeus})$
- $\text{Diety}(\text{Zeus}), \text{Female}(\text{Rhea})$
- $\text{Narcissist}(\text{Narcissus})$

Knowledge (TBox)

- $\text{Mortal} \sqsubseteq \neg \text{Diety}$
- $\top \sqsubseteq \exists \text{hasParent}.\text{Male} \sqcap \exists \text{hasParent}.\text{Female}$
- $\text{Narcist} \sqsubseteq \exists \text{loves}.\text{Self}$

Icons downloaded from icon-icons.com by ©Rena Xiao and ©Eucalyp Studio (both under CC BY 4.0). No changes have been made.
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

- hasParent(Heracles, Zeus)
- Diety(Zeus), Female(Rhea)
- Narcissist(Narcissus)

Knowledge (TBox)

$\text{Mortal} \sqsubseteq \neg \text{Diety}$

$\top \sqsubseteq \exists \text{hasParent}.\text{Male} \sqcap \exists \text{hasParent}.\text{Female}$

$\text{Narcist} \sqsubseteq \exists \text{loves}.\text{Self}$

The DL encompasses all these features is called $\mathcal{ALC}_{\text{Self}}$. 

The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard.
Running example: Greek mythology $\mathcal{ALC}_{\mathit{Self}}$ knowledge base

Database (ABox)

- hasParent(Heracles, Zeus)
- Diety(Zeus), Female(Rhea)
- Narcissist(Narcissus)

Knowledge (TBox)

- Mortal $\sqsubseteq \neg$Diety
- $\top \sqsubseteq \exists \text{hasParent}.\text{Male} \sqcap \exists \text{hasParent}.\text{Female}$
- Narcist $\sqsubseteq \exists \text{loves}.\text{Self}$

The DL encompasses all these features is called $\mathcal{ALC}_{\mathit{Self}}$. 

The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\mathit{Self}}$ is 2ExpTime-hard
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

The DL encompasses all these features is called $\mathcal{ALC}_{\text{Self}}$.

\texttt{hasParent(Heracles, Zeu\text{s})}
\texttt{Diety(Zeus), Female(Rhea)}
\texttt{Narcissist(Narcissus)}

\textbf{Conjunctive Queries:} Give me IDs of all candidates who applied for "computer science".

Knowledge (TBox)

\texttt{Mortal \sqsubseteq \neg \text{Diety}}
\texttt{T \sqsubseteq \exists \text{hasParent.Male} \land \exists \text{hasParent.Female}}
\texttt{Narcist \sqsubseteq \exists \text{loves.Self}}
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

Database (ABox)

hasParent(Heracles, Zeus)  
Diety(Zeus), Female(Rhea)  
Narcissist(Narcissus)

The DL encompasses all these features is called $\mathcal{ALC}_{\text{Self}}$.

Knowledge (TBox)

Mortal $\subseteq \neg\text{Diety}$  
$\top \subseteq \exists\text{hasParent}.\text{Male} \sqcap \exists\text{hasParent}.\text{Female}$  
Narcist $\subseteq \exists\text{loves}.\text{Self}$

Conjunctive Queries: Give me IDs of all candidates who applied for “computer science”.

```
SELECT CandID  
FROM Candidate  
WHERE Major = "Computer Science"
```
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

**Database (ABox)**

- $\text{hasParent}(\text{Heracles}, \text{Zeus})$
- $\text{Diety}(\text{Zeus}), \text{Female}(\text{Rhea})$
- $\text{Narcissist}(\text{Narcissus})$

**Knowledge (TBox)**

- Mortal $\sqsubseteq \neg \text{Diety}$
- $\top \sqsubseteq \exists \text{hasParent} \cdot \text{Male} \sqcap \exists \text{hasParent} \cdot \text{Female}$
- Narcist $\sqsubseteq \exists \text{loves} \cdot \text{Self}$

**Conjunctive Queries:** Give me IDs of all candidates who applied for "computer science".

```sql
SELECT CandID
FROM Candidate
WHERE Major = "Computer Science"
```

$\varphi(i) \rightsquigarrow \varphi(i)$

Icons downloaded from icon-icons.com by ©Rena Xiao and ©Eucalyp Studio (both under CC BY 4.0). No changes have been made.
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

**Database (ABox)**

- `hasParent(Heracles, Zeus)`
- `Diety(Zeus), Female(Rhea)`
- `Narcissist(Narcissus)`

**Knowledge (TBox)**

- `Mortal \subseteq \neg Diety`
- `\top \subseteq \exists hasParent.\text{Male} \sqcap \exists hasParent.\text{Female}`
- `Narcist \subseteq \exists \text{loves}.\text{Self}`

**Conjunctive Queries:** Give me IDs of all candidates who applied for “computer science”.

```
SELECT CandID
FROM Candidate
WHERE Major = "Computer Science"
```

The DL encompasses all these features is called $\mathcal{ALC}_{\text{Self}}$.

Bartosz “Bart” Bednarczyk

The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{EXP\-TIME}$-hard
Running example: Greek mythology \( \mathcal{ALC}_{Self} \) knowledge base

Database (ABox)

- hasParent(Heracles, Zeus)
- Diety(Zeus), Female(Rhea)
- Narcissist(Narcissus)

Knowledge (TBox)

- Mortal \( \sqsubseteq \neg \text{Diety} \)
- \( \top \sqsubseteq \exists \text{hasParent}.\text{Male} \sqcap \exists \text{hasParent}.\text{Female} \)
- Narcist \( \sqsubseteq \exists \text{loves}.\text{Self} \)

Conjunctive Queries: Give me IDs of all candidates who applied for “computer science”.

```
SELECT CandID
FROM Candidate
WHERE Major = "Computer Science"
```

\[ \varphi(i) = \exists n \exists s \text{CANDIDATE}(i, n, s) \land \text{APPL}(\text{"Computer Science"}, i) \]
Running example: Greek mythology $\mathcal{ALC}_{\text{Self}}$ knowledge base

**Database (ABox)**

- hasParent(Heracles, Zeus)
- Diety(Zeus), Female(Rhea)
- Narcissist(Narcissus)

**Knowledge (TBox)**

- Mortal $\subseteq \neg$Diety
- $\top \subseteq \exists$hasParent.Male $\land \exists$hasParent.Female
- Narcist $\subseteq \exists$loves.Self

**Conjunctive Queries:** Give me IDs of all candidates who applied for "computer science".

```
SELECT CandID
FROM Candidate
WHERE Major = "Computer Science"
```

$\varphi(i) = \exists n \exists s \text{CANDIDATE}(i, n, s) \land \text{APPL}("Computer Science", i)$

A knowledge base $\mathcal{K}$ entails a conjunctive query $q$ (written: $\mathcal{K} \models q$) if $q$ matches all models of $\mathcal{K}$. 

Icons downloaded from icon-icons.com by ©Rena Xiao and ©Eucalyp Studio (both under CC BY 4.0). No changes have been made.
Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them do not increase the complexity, e.g. $\mathcal{ALC} + H$, $\mathcal{ALC} + Q$ [Lutz'08]
   
   $\text{car} \sqsubseteq \text{hasParent}$
   
   $\text{hasPart} \sqsubseteq \text{Wheel}$

   Also arithmetic and statistical properties [Baader, B., Rudolph'20]

   As well as regular expressions, fixed points, (safe) role combination [B.'21, ArXiV]

   And even a tamed use of higher-arity relations [B.'21, JELIA]

2. Some of them increase the complexity exponentially:
   
   E.g. inverses [Lutz'07], transitivity [Eiter et al.'09], nominals (a.k.a. constants) [Ngo et al.'16]

   What about the eponymous $\text{Self}$ operator? Is it harmless?

   $\text{Self}$ is supported by OWL 2 Web Ontology Language,

   $\exists r. \text{Self} I := \{ d \mid (d, d) \in r \}$

   The complexity of satisfiability stays the same, even for very expressive $\mathcal{ALCH}$ family, a.k.a. $\mathcal{ALC}_{\text{Self} \text{reg}}$

   Easy to accommodate in the automata-based approach

   $\text{Self}$ is present in OWL2 EL/RL, without harming tractability [Krötzsch et al., ISWC'08]

   Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}} T$Boxes is $2\text{ExpTime}$-hard.
Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them do not increase the complexity, e.g. $\mathcal{ALC} + \mathcal{H}, \mathcal{ALC} + \mathcal{Q}$ [Lutz'08]

2. Some of them increase the complexity exponentially: E.g. inverses [Lutz'07], transitivity [Eiter et al.'09], nominals (a.k.a. constants) [Ngo et al.'16]

What about the eponymous $\mathcal{Self}$ operator? Is it harmless?

$\mathcal{Self}$ is supported by OWL 2 Web Ontology Language, $(\exists r. \mathcal{Self})$:

$I := \{ d | (d, d) \in r \}$

The complexity of satisfiability stays the same, even for very expressive $\mathcal{Z}$ family, a.k.a. $\mathcal{ALC}_{b\mathcal{Self}}$

It's easy to accommodate in the automata-based approach.

$\mathcal{Self}$ is present in OWL2 EL/RL, without harming tractability [Krötzsch et al., ISWC'08]

Conjunctive query entailment over $\mathcal{ALC} \mathcal{Self}^T$Boxes is $2\text{ExpTime}$-hard.
Our motivation: what features make CQ answering hard for $ALC$?

1. Some of them do not increase the complexity, e.g. $ALC+H$, $ALC+Q$ [Lutz’08]

$\text{hasMother} \subseteq \text{hasParent}$

2. Some of them increase the complexity exponentially: E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

What about the eponymous $Self$ operator? Is it harmless?

$Self$ is supported by OWL 2 Web Ontology Language, $(\exists r. Self) := \{ d | (d, d) \in r \}$

- The complexity of satisfiability stays the same, even for very expressive $Z$ family, a.k.a. $ALC_{Self}^{\text{reg}}$
- Easy to accommodate in the automata-based approach
- $Self$ is present in OWL2 EL/RL, without harming tractability [Krötzsch et al., ISWC’08]

Conjunctive query entailment over $ALC_{Self}$ $T$Boxes is $2\text{ExpTime}$-hard.
Our motivation: what features make CQ answering hard for \( \mathcal{ALC} \)?

1. Some of them do not increase the complexity, e.g. \( \mathcal{ALC} + H, \mathcal{ALC} + Q \) [Lutz'08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \sqsubseteq (\,=4\,.)\text{hasPartWheel}
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph'20]
- As well as regular expressions, fixed points, (safe) role combination [B.'21, ArXiV]
- And even a tamed use of higher-arity relations [B.'21, JELIA]

2. Some of them increase the complexity exponentially:
- E.g. inverses [Lutz'07], transitivity [Eiter et al.'09], nominals (a.k.a. constants) [Ngo et al.'16]

What about the eponymous \( \text{Self} \) operator? Is it harmless?

\[
\text{Self} = \{ d \mid (d, d) \in r \}
\]

- The complexity of satisfiability stays the same, even for very expressive \( \mathcal{Z} \) family, a.k.a. \( \mathcal{ALC}_{\text{Self}} \)
- Easy to accommodate in the automata-based approach
- \( \text{Self} \) is present in OWL2 EL/RL, without harming tractability [Krötzsch et al, ISWC'08]

Conjunctive query entailment over \( \mathcal{ALC} \) \( \text{Self} \) TBoxes is 2ExpTime-hard.
Our motivation: what features make CQ answering hard for $ALC$?

1. Some of them do not increase the complexity, e.g. $ALC+\mathcal{H}$, $ALC+\mathcal{Q}$ [Lutz’08]

- $\text{hasMother} \subseteq \text{hasParent}$
- $\text{Car} \sqsubseteq (\mathbf{= 4})\text{hasPartWheel}$

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
Our motivation: what features make CQ answering hard for \( \mathcal{ALC} \)?

1. Some of them do not increase the complexity, e.g. \( \mathcal{ALC} + \mathcal{H}, \mathcal{ALC} + \mathcal{Q} \) [Lutz’08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \sqsubseteq (= 4).\text{hasPartWheel} \\
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]

\[
\begin{align*}
\text{hasMother} & \subseteq \text{hasParent} \\
\text{Car} & \sqsubseteq (= 4).\text{hasPartWheel} \\
\end{align*}
\]
Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them do not increase the complexity, e.g. $\mathcal{ALC}+\mathcal{H}$, $\mathcal{ALC}+\mathcal{Q}$ [Lutz’08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \sqsubseteq (= 4).\text{hasPartWheel}
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
- And even a tamed use of higher-arity relations [B.’21, JELIA]
Our motivation: what features make CQ answering hard for \( \mathcal{ALC} \)?

1. Some of them do not increase the complexity, e.g. \( \mathcal{ALC} + H \), \( \mathcal{ALC} + Q \) [Lutz’08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \sqsubseteq (= 4).\text{hasPartWheel}
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
- And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:

- Inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]
- What about the eponymous \texttt{Self} operator? Is it harmless?

\[ (\exists r. \text{Self}) \]

\[
\{ d \mid (d, d) \in r \}
\]

- The complexity of satisfiability stays the same, even for very expressive \( \mathcal{Z} \) family, a.k.a. \( \mathcal{ALCH} \)
- \texttt{Self} is present in OWL2 EL/RL, without harming tractability [Krötzsch et al, ISWC’08]

Conjunctive query entailment over \( \mathcal{ALC} \texttt{Self} \) TBoxes is 2ExpTime-hard.
Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them do not increase the complexity, e.g. $\mathcal{ALC} + \mathcal{H}$, $\mathcal{ALC} + \mathcal{Q}$ [Lutz‘08]

   
   \[
   \text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \sqsubseteq (= 4).\text{hasPartWheel}
   \]

   - Also arithmetic and statistical properties [Baader, B., Rudolph‘20]
   - As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
   - And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:

E.g. inverses [Lutz‘07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]
Our motivation: what features make CQ answering hard for \(\text{ALC}\)?

1. Some of them do not increase the complexity, e.g. \(\text{ALC} + H, \text{ALC} + Q\) [Lutz’08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \subseteq (= 4).\text{hasPartWheel}
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
- And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:

E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

What about the eponymous Self operator? Is it harmless?
Our motivation: what features make CQ answering hard for \( \text{ALC} \)?

1. Some of them do not increase the complexity, e.g. \( \text{ALC}+H \), \( \text{ALC}+Q \) [Lutz’08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \subseteq (\text{= 4}).\text{hasPartWheel}
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
- And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:

E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

What about the eponymous \text{Self} operator? Is it harmless?

Self is supported by OWL 2 Web Ontology Language, \( (\exists r.\text{Self})^I := \{d \mid (d, d) \in r^I \} \)
Our motivation: what features make CQ answering hard for \( \mathcal{ALC} \)?

1. Some of them do not increase the complexity, e.g. \( \mathcal{ALC} + \mathcal{H}, \mathcal{ALC} + \mathcal{Q} \) [Lutz’08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \sqsubseteq (\equiv 4).\text{hasPartWheel}
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
- And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:

E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

---

What about the eponymous Self operator? Is it harmless?

Self is supported by OWL 2 Web Ontology Language, \((\exists r.\text{Self})^I := \{ d \mid (d, d) \in r^I \}\)

- The complexity of satisfiability stays the same, even for very expressive \( \mathcal{Z} \) family, a.k.a. \( \mathcal{ALCHb}^{\text{Self}} \)
Our motivation: what features make CQ answering hard for \( \mathcal{ALC} \)?

1. Some of them do not increase the complexity, e.g. \( \mathcal{ALC} + \mathcal{H}, \mathcal{ALC} + \mathcal{Q} \) [Lutz’08]
   
   \[
   \text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \subseteq (= 4).\text{hasPartWheel}
   \]
   
   - Also arithmetic and statistical properties [Baader, B., Rudolph’20]
   - As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
   - And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:
   
   E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

What about the eponymous Self operator? Is it harmless?

Self is supported by OWL 2 Web Ontology Language, \( (\exists r.\text{Self})^I := \{ d \mid (d, d) \in r^I \} \)

- The complexity of satisfiability stays the same, even for very expressive \( \mathcal{Z} \) family, a.k.a. \( \mathcal{ALCHb}_{\text{reg}}^{\text{Self}} \)
- Easy to accommodate in the automata-based approach
**Our motivation: what features make CQ answering hard for \( \mathcal{ALC} \)?**

1. Some of them do not increase the complexity, e.g. \( \mathcal{ALC} + \mathcal{H}, \mathcal{ALC} + \mathcal{Q} \) [Lutz’08]

\[
\text{hasMother} \subseteq \text{hasParent} \quad \text{Car} \sqsubseteq (= 4).\text{hasPartWheel}
\]

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
- And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:

E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

---

**What about the eponymous Self operator? Is it harmless?**

Self is supported by OWL 2 Web Ontology Language, \((\exists r.\text{Self})^I := \{d \mid (d, d) \in r^I\}\)

- The complexity of satisfiability stays the same, even for very expressive \( \mathcal{Z} \) family, a.k.a. \( \mathcal{ALCH} b^\text{Self}_{\text{reg}} \)
- Easy to accommodate in the automata-based approach
- Self is present in OWL2 EL/RL, without harming tractability [Krötzsch et al, ISWC’08]
Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them do not increase the complexity, e.g. $\mathcal{ALC} + \mathcal{H}$, $\mathcal{ALC} + \mathcal{Q}$ [Lutz’08]

- $\text{hasMother} \subseteq \text{hasParent}$
- $\text{Car} \sqsubseteq (\mathcal{=} 4).\text{hasPartWheel}$

- Also arithmetic and statistical properties [Baader, B., Rudolph’20]
- As well as regular expressions, fixed points, (safe) role combination [B.’21, ArXiV]
- And even a tamed use of higher-arity relations [B.’21, JELIA]

2. Some of them increase the complexity exponentially:

- E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

What about the eponymous Self operator? Is it harmless?

Self is supported by OWL 2 Web Ontology Language, $(\exists r. \text{Self})^I := \{ d \mid (d, d) \in r^I \}$

- The complexity of satisfiability stays the same, even for very expressive $\mathcal{Z}$ family, a.k.a. $\mathcal{ALCHb}_{\text{reg}}$
- Easy to accommodate in the automata-based approach
- Self is present in OWL2 EL/RL, without harming tractability [Krötzsch et al, ISWC’08]
Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

1. Some of them do not increase the complexity, e.g. $\mathcal{ALC} + H$

   • Also arithmetic and statistical properties
   • As well as regular expressions
   • And even a tamed use of higher-arity relations

2. Some of them increase the complexity exponentially:

   E.g. inverses [Lutz’07], transitivity [Eiter et al.’09], nominals (a.k.a. constants) [Ngo et al.’16]

   What about the eponymous $\text{Self}$ operator? Is it harmless?

   $\text{Self}$ is supported by OWL 2 Web Ontology Language,

   $\exists r . \text{Self} = \{ d | (d,d) \in r \}$

   • The complexity of satisfiability stays the same, even for very expressive $\mathcal{Z}$ family, a.k.a. $\mathcal{ALCH}$
   • Easy to accommodate in the automata-based approach
   • $\text{Self}$ is present in OWL2 EL/RL, without harming tractability [Krötzsch et al, ISWC’08]

Conjunctive query entailment over $\mathcal{ALC}_\text{Self}$ TBoxes is $2\text{ExpTime}$-hard.

The Price of Selfishness: Conjunctive Query Entailment for $\mathcal{ALC}_\text{Self}$ is $2\text{ExpTime}$-hard

(Extended Abstract)*

Bartosz Bednarczyk$^{1,2}$ and Sebastian Rudolph$^1$

1 Computational Logic Group, University of Leipzig, Germany
2 Institute of Computer Science, University of Wrocław, Poland

{bartosz.bednarczyk, sebastian.rudolph}@tu-dresden.de

Various modelling features of DLs affect the complexity of conjunctive query (CQ) entailment in a rather intricate manner. For the most popular basic description logic (DL), $\mathcal{ALC}$, the complexity of CQ entailment is known to be $\text{ExpTime}$-complete, as is that of knowledge base satisfiability. It was first shown in [9, Thm. 2] that CQ entailment becomes exponentially harder when $\mathcal{ALC}$ is extended with inverse roles ($\mathcal{I}$), while the complexity of satisfiability remains the same. Shortly after, a combination of transitivity and role hierarchies ($\mathcal{SH}$) was shown to be another culprit of the $\text{ExpTime}$-complexity explosion [5, Thm. 1]. Finally, also nominals ($\mathcal{O}$) turned out to be equally problematic [10, Thm. 9]. On the other hand, there are also DL constructs that do not affect the complexity of CQ entailment. Examples are counting ($\mathcal{Q}$) [9, Thm. 4] (the complexity stays the same even for expressive arithmetical constraints [1, Thm. 21]), role-hierarchies alone ($\mathcal{H}$) [6, Cor. 3], and even a tamed use of highly-arity relations [2, Thm. 20].
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.

**Consequences?**

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{reg}}$) family is $2\text{ExpTime}$-hard.
- Hardness does not follow from $\mathcal{SH}$ (no transitivity in CQs!).
- Fluted Guarded Fragment with $\exists$ has $2\text{ExpTime}$-hard CQ querying (contrasts [B’21, JELIA]).

**Proof scheme?**

- A reduction from the acceptance problem for the empty-tape $\mathcal{AExpSpace}$ TMs.
- The models of an $\mathcal{ALC}_{\text{Self}}$-KB $\mathcal{K}_M$ describe possibly faulty runs of a given ATM $M$.
- A CQ $q_M$ detects mismatches in the consecutive transitions.
- $\mathcal{K}_M \not= q_M$ iff there is a (non-faulty) accepting run of $M$.
Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}^{\text{Self}}_{\text{reg}}$) family is $2\text{ExpTime}$-hard.*
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.

Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALC}_{\text{Self}}^{\text{Self}}^{\text{reg}}$) family is $2\text{ExpTime}$-hard.*

---

* Hardness does not follow from $\mathcal{SH}$ (no transitivity in CQs!).
Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{Self}}^{\text{reg}}$) family is $2\text{ExpTime}$-hard.*
Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{Self}}^{\text{reg}}$) family is $\mathsf{2ExpTime}$-hard.*
- **Fluted Guarded Fragment with $=$** has $\mathsf{2ExpTime}$-hard CQ querying (contrasts [B’21, JELIA]).†
Conjunctive query entailment over $\mathcal{ALC}_{Self}$ TBoxes is $2\text{ExpTime}$-hard.

Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{reg}}^{\text{Self}}$) family is $2\text{ExpTime}$-hard.*
- Fluted Guarded Fragment with $=\text{has}$ $2\text{ExpTime}$-hard CQ querying (contrasts [B’21, JELIA])†

$\forall x_1 (\text{self}_r(x_1) \rightarrow \exists x_2 [R(x_1, x_2) \land x_1=x_2]) \land \forall x_1 \forall x_2 (R(x_1, x_2) \rightarrow [x_1=x_2 \rightarrow \text{self}_r(x_2)])$
Consequences?

- Querying the \( \mathcal{Z} \) (a.k.a. \( \mathcal{ALCH_{\text{Self}}}^{\text{Self}} \)) family is \( 2\text{ExpTime} \)-hard.*
- Fluted Guarded Fragment with \( = \) has \( 2\text{ExpTime} \)-hard CQ querying (contrasts \([B'21, JELIA])\)†

*Hardness does not follow from \( \mathcal{SH} \) (no transitivity in CQs!).
†∀ \( x_1 (\text{self} r (x_1) \rightarrow \exists x_2 [R(x_1, x_2) \land x_1 = x_2]) \land \forall x_1 \forall x_2 (R(x_1, x_2) \rightarrow [x_1 = x_2 \rightarrow \text{self} r(x_2)]) \)

Proof scheme?

- A reduction from the acceptance problem for the empty-tape \( \mathcal{AExpSpace} \) TMs.
- The models of an \( \mathcal{ALC_{Self}} \)-KB \( K_M \) describe possibly faulty runs of a given ATM \( M \).
- A CQ \( q_M \) detects mismatches in the consecutive transitions.
- \( K_M \not\models q_M \) iff there is a (non-faulty) accepting run of \( M \).
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{EXP}T\text{IME}$-hard.

Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{reg}}^{\text{Self}}$) family is $2\text{EXP}T\text{IME}$-hard.*
- Fluted Guarded Fragment with $=$ has $2\text{EXP}T\text{IME}$-hard CQ querying (contrasts [B’21, JELIA])†

Proof scheme?

---

*Bartosz “Bart” Bednarczyk
The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{EXP}T\text{IME}$-hard 3 / 7
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.

Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{reg}}^{\text{Self}}$) family is $2\text{ExpTime}$-hard.*
- Fluted Guarded Fragment with $=$ has $2\text{ExpTime}$-hard CQ querying (contrasts $[B'21, \text{JELIA}]$)†

Proof scheme?

- A reduction from the acceptance problem for the empty-tape $\text{AExpSpace}$ TMs.
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.

Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}^\text{Self}_{\text{reg}}$) family is $2\text{ExpTime}$-hard.∗

- Fluted Guarded Fragment with $=$ has $2\text{ExpTime}$-hard CQ querying (contrasts [B’21, JELIA])†

Proof scheme?

- A reduction from the acceptance problem for the empty-tape $\text{AExpSpace}$ TMs.
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.

Consequences?
- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALC}h_{\text{Self}}^{\text{reg}}$) family is $2\text{ExpTime}$-hard.∗
- Fluted Guarded Fragment with $=$ has $2\text{ExpTime}$-hard CQ querying (contrasts [B’21, JELIA])†

Proof scheme?
- A reduction from the acceptance problem for the empty-tape $\text{AExpSpace}$ TMs.
- The models of an $\mathcal{ALC}_{\text{Self}}$-KB $\mathcal{K}_M$ describe possibly faulty runs of a given ATM $M$. 

Bartosz “Bart” Bednarczyk
The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard

∗ Hardness does not follow from $\mathcal{SH}$ (no transitivity in CQs!).
† ∀ $x_1$ ($\text{self}_r(x_1) \rightarrow \exists x_2$ [$R(x_1, x_2) \land x_1 = x_2$] \land ∀ $x_1$ ∀ $x_2$ ($R(x_1, x_2) \rightarrow [x_1 = x_2 \rightarrow \text{self}_r(x_2)$]).
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.

Consequences?

- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{reg}}^{\text{Self}}$) family is $2\text{ExpTime}$-hard.*
- Fluted Guarded Fragment with $=$ has $2\text{ExpTime}$-hard CQ querying (contrasts [B’21, JELIA])†

Proof scheme?

- A reduction from the acceptance problem for the empty-tape $\text{AExpSpace}$ TMs.

- The models of an $\mathcal{ALC}_{\text{Self}}$-KB $\mathcal{K}_M$ describe possibly faulty runs of a given ATM $\mathcal{M}$.
- A CQ $q_M$ detects mismatches in the consecutive transitions.

Bartosz “Bart” Bednarczyk
The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard
Conjunctive query entailment over $\mathcal{ALC}_{\text{Self}}$ TBoxes is $2\text{ExpTime}$-hard.

Consequences?
- Querying the $\mathcal{Z}$ (a.k.a. $\mathcal{ALCH}_{\text{reg}}^{\text{Self}}$) family is $2\text{ExpTime}$-hard.*
- Fluted Guarded Fragment with $=$ has $2\text{ExpTime}$-hard CQ querying (contrasts [B’21, JELIA])†

Proof scheme?
- A reduction from the acceptance problem for the empty-tape $\text{AExpSpace}$ TMs.
- The models of an $\mathcal{ALC}_{\text{Self}}$-KB $\mathcal{K}_M$ describe possibly faulty runs of a given ATM $M$.
- A CQ $q_M$ detects mismatches in the consecutive transitions.
- $\mathcal{K}_M \not\models q_M$ iff there is a (non-faulty) accepting run of $M$. 

Bartosz “Bart” Bednarczyk  The Price of Selfishness: CQ Entailment for $\mathcal{ALC}_{\text{Self}}$ is $2\text{ExpTime}$-hard 3 / 7
Proof ideas: Our encoding

• We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.

• Novelty: nodes will be decorated with certain self-loops.

• To avoid a seemingly required disjunction in our CQs the tape content is stored implicitly with:

• All other details are as one may expect. See: https://arxiv.org/abs/2106.15150
Proof ideas: Our encoding

- We encode configurations as full-binary trees of depth $n+1$ with their roots connected with $\text{next-role}$.

- Novelty: nodes will be decorated with certain self-loops.

- To avoid a seemingly required disjunction in our CQs the tape content is stored implicitly with $\text{next-role}$.

- All other details are as one may expect. See: https://arxiv.org/abs/2106.15150
Proof ideas: Our encoding

- We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.
- Novelty: nodes will be decorated with certain self-loops.
Proof ideas: Our encoding

- We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.
- Novelty: nodes will be decorated with certain self-loops.
- To avoid a seemingly required disjunction in our CQs the tape content is stored implicitly with:
Proof ideas: Our encoding

• We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.

• Novelty: nodes will be decorated with certain self-loops.

• To avoid a seemingly required disjunction in our CQs the tape content is stored implicitly with:
Proof ideas: Our encoding

- We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.
- Novelty: nodes will be decorated with certain self-loops.
- To avoid a seemingly required disjunction in our CQs the tape content is stored implicitly with:

![Diagram of tree with self-loops](image)

- All other details are as one may expect. See: https://arxiv.org/abs/2106.15150
Proof ideas: Our encoding

- We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.
- Novelty: nodes will be decorated with certain self-loops.
- To avoid a seemingly required disjunction in our CQs the tape content is stored implicitly with:

All other details are as one may expect. See: https://arxiv.org/abs/2106.15150
Trick no. 1: A single root-to-leaves conjunctive query

∃x₁ ∃x₂ ∃x₃ \text{Lvl}_0(x) \land ℓ_1(x, x₁) \land r_1(x₁, x₂) \land ℓ_2(x₂, x₃) \land r_2(x₃, y) \land \text{Lvl}_2(y).

For brevity we write: \((\text{Lvl}_0; ℓ_1; r_1; ℓ_2; r_2; \text{Lvl}_2)(x, y)\).
Trick no. 1: A single root-to-leaves conjunctive query

Goal: Design a CQ $q(x, y)$ such that $x$ matches the root and $y$ matches any of the leaves.

For brevity we write: $(Lvl_0; \ell_1; r_1; \ell_2; r_2; Lvl_2)(x, y)$.
Trick no. 1: A single root-to-leaves conjunctive query

Goal: Design a CQ $q(x, y)$ such that $x$ matches the root and $y$ matches any of the leaves.

For brevity we write: $(Lvl_0; \ell_1; r_1; \ell_2; r_2; Lvl_2)(x, y)$.
Trick no. 1: A single root-to-leaves conjunctive query

Goal: Design a CQ $q(x, y)$ such that $x$ matches the root and $y$ matches any of the leaves.

$\exists x_1 \exists x_2 \exists x_3 \ Lvl_0(x) \land \ell_1(x, x_1) \land r_1(x_1, x_2) \land \ell_2(x_2, x_3) \land r_2(x_3, y) \land Lvl_2(y)$
Trick no. 1: A single root-to-leaves conjunctive query

Goal: Design a CQ \( q(x, y) \) such that \( x \) matches the root and \( y \) matches any of the leaves.

\[
\exists x_1 \exists x_2 \exists x_3 \ Lvl_0(x) \land \ell_1(x, x_1) \land r_1(x_1, x_2) \land \ell_2(x_2, x_3) \land r_2(x_3, y) \land Lvl_2(y)
\]

For brevity we write: \((Lvl_0?; \ell_1; r_1; \ell_2; r_2; Lvl_2?)(x, y)\).
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a $CQ(q(x,y))$ that matches leaves $x, y$ with equal addresses.

Select two leaves located in different trees:

\[
\begin{align*}
&Lvl_2 ; r^{2-2}; \ell^{2-2}; r^{2-1}; \ell^{1-2}; Lvl_0 ; next; Lvl_0 ; \ell^{1-1}; r^{1-1}; \ell^{2-2}; r^{2-1}; Lvl_2 ; \\
\end{align*}
\]

Impose that they have the same first bit of their address:

\[
\begin{align*}
&\wedge(r^{2-2}; \ell^{2-2}; \ell^{1-2}; next; \ell^{1-1}; r^{1-1}; \ell^{2-2}; r^{2-1}; Lvl_2 ; r^{2-2}); \\
\end{align*}
\]

as well as the same second bit of their address:

\[
\begin{align*}
&\wedge(\ell^{2-2}; r^{2-1}; \ell^{1-2}; next; \ell^{1-1}; r^{1-1}; \ell^{2-2}; r^{2-1}; Lvl_2 ; r^{2-2}; r^{2-1}; \ell^{1-2}; next; \ell^{1-1}; r^{1-1}; \ell^{2-2}; r^{2-1}; Lvl_2 ) \\
\end{align*}
\]
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ

\[ q(x, y) \]

that matches leaves \( x, y \) with equal addresses.

Select two leaves located in different trees:

\[ (\text{Lvl } 2; r - 2; \ell - 2; r - 1; \ell - 1; \text{Lvl } 0; \text{next}; \text{Lvl } 0; \ell 1; r 1; \ell 2; r 2; \text{Lvl } 2; r - 2; \ell - 2; r - 1; \text{next}; \ell 1; r 1; \ell 2; r 2) \]

Impose that they have the same first bit of their address:

\[ \land (r - 2; \ell - 2; \ell - 1; \text{next}; \ell 1; \ell 2; r 2; \text{Lvl } 2; r - 2; \ell - 2; r - 1; \text{next}; \ell 1; r 1; \ell 2; r 2) \]

as well as the same second bit of their address:

\[ \land (\ell - 2; r - 1; \ell - 1; \text{next}; \ell 1; r 1; \ell 2; \text{Lvl } 2; r - 2; r - 1; \ell - 1; \text{next}; \ell 1; r 1; \ell 2; r 2) \]
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ $q(x, y)$ that matches leaves $x$, $y$ with equal addresses.

Select two leaves located in different trees:

- Lvl 2: $\ell - 2; r - 2; \ell - 1; r - 1; Lvl 0; next; Lvl 0; \ell 1; r 1; \ell 2; r 2; Lvl 2$)
- Lvl 0: $\ell; r 1; \ell; r 2$

Impose that they have the same first bit of their address:

$\land (\ell - 2; r - 2; \ell - 1; next; r 1; \ell 2; r 2)$

as well as the same second bit of their address:

$\land (\ell - 2; r - 1; \ell - 1; next; r 1; \ell 2; r 2)$
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ \( q(x, y) \) that matches leaves \( x, y \) with equal addresses.

Select two leaves located in different trees:
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ $q(x, y)$ that matches leaves $x, y$ with equal addresses.

Select two leaves located in different trees:

$$(Lvl_2?; r_2^-; \ell_2^-; r_1^-; Lvl_0?; \text{next}; Lvl_0?; \ell_1; \ell_2; r_2; Lvl_2?)(x, y)$$
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ \( q(x, y) \) that matches leaves \( x, y \) with equal addresses.

Select two leaves located in different trees:

\[
(Lvl_2?; r_2^-; l_2^-; l_1^-; Lvl_0?; next; Lvl_0?; l_1; r_1; Lvl_2?)(x, y)
\]

Impose that they have the same first bit of their address:
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ $q(x, y)$ that matches leaves $x, y$ with equal addresses.

Select two leaves located in different trees:

$$(\text{Lvl}_2?; r_2^-; \ell_2^-; r_1^-; \ell_1^-; \text{Lvl}_0?; \text{next}; \text{Lvl}_0?; \ell_1; \ell_2; r_2; \text{Lvl}_2?)(x, y)$$

Impose that they have the same first bit of their address:

$$\land (r_2^-; \ell_2^-; \ell_1^-; \text{next}; \ell_1; \ell_2; r_2; \text{Lvl}_2?; r_2^-; \ell_2^-; r_1^-; \text{next}; r_1; \ell_2; r_2)(x, y)$$
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ $q(x, y)$ that matches leaves $x, y$ with equal addresses.

Select two leaves located in different trees:

$$(Lvl_2^-; r_2^-; ℓ_2^-; r_1^-; ℓ_1^-; Lvl_0^-; next; Lvl_0^-; ℓ_1; r_1; ℓ_2; r_2; Lvl_2^-)(x, y)$$

Impose that they have the same first bit of their address:

$$\land (r_2^-; ℓ_2^-; next; ℓ_1; ℓ_2; r_2; Lvl_2^-; r_2^-; ℓ_2^-; r_1^-; next; r_1; ℓ_2; r_2)(x, y)$$

as well as the same second bit of their address:
Trick no. 2: Synchronisation of leaves among two trees

Goal: Design a CQ \( q(x, y) \) that matches leaves \( x, y \) with equal addresses.

Select two leaves located in different trees:

\[
(Lvl_2?; r_2^-; l_2^-; r_1^-; l_1^-; Lvl_0?; next; Lvl_0?; l_1; l_2; r_2; Lvl_2?)(x, y)
\]

Impose that they have the same first bit of their address:

\[
\land (r_2^-; l_2^-; l_1^-; next; l_1; l_2; r_2; Lvl_2?; r_2^-; l_2^-; r_1^-; next; r_1; l_2; r_2)(x, y)
\]

as well as the same second bit of their address:

\[
\land (l_2^-; r_1^-; l_1^-; next; l_1; r_1; l_2; Lvl_2?; r_2^-; r_1^-; l_1^-; next; l_1; r_1; r_2)(x, y)
\]
The end: Thanks for your attention!

Biggest challenge: Design a CQ $q(x, y)$ that matches leaves $x, y$ with equal addresses.

Conjunctive query entailment over $\mathcal{ALC}_{Self}$ TBoxes is $2\text{ExpTime}$-hard.