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

February 22 - March 1 2022, AAAI 2022

Bartosz "Bart" Bednarczyk, Sebastian Rudolph

TU Dresden & University of Wrocław

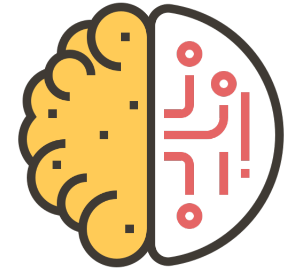
Powered by BeamerikZ

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

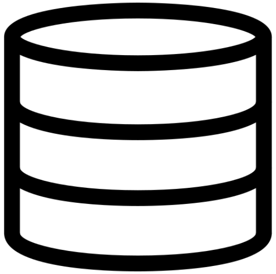
Database (ABox)

Knowledge (TBox)

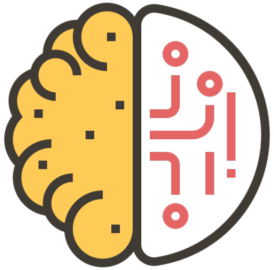


Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)



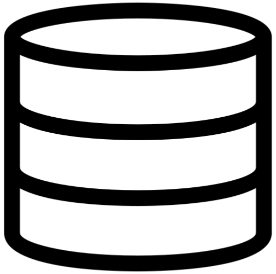
Knowledge (TBox)



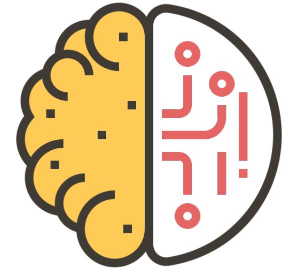
$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)



Knowledge (TBox)



$\mathit{hasParent}(\texttt{Heracles}, \texttt{Zeus})$

$\mathit{Diety}(\texttt{Zeus})$, $\mathit{Female}(\texttt{Rhea})$

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

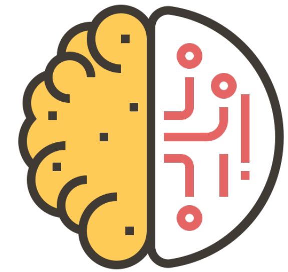
Database (ABox)

$\mathit{hasParent}(\texttt{Heracles}, \texttt{Zeus})$

$\mathit{Diety}(\texttt{Zeus})$, $\mathit{Female}(\texttt{Rhea})$

$\mathit{Narcissist}(\texttt{Narcissus})$

Knowledge (TBox)



Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

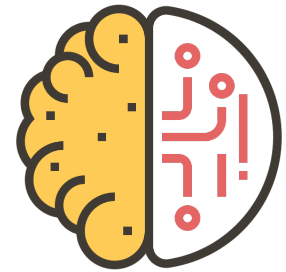
Database (ABox)

$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus})$, $\mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

Knowledge (TBox)



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

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

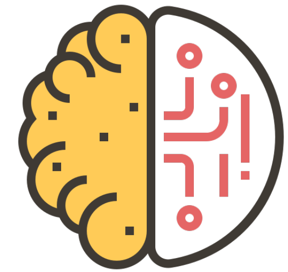
Database (ABox)

$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus}), \mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

Knowledge (TBox)

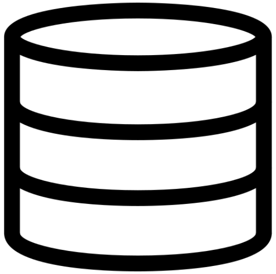


$\mathit{Mortal} \sqsubseteq \neg \mathit{Diety}$

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

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)

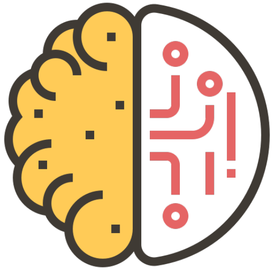


$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus})$, $\mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

Knowledge (TBox)

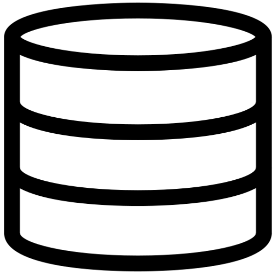


$$\mathit{Mortal} \sqsubseteq \neg \mathit{Diety}$$

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

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)

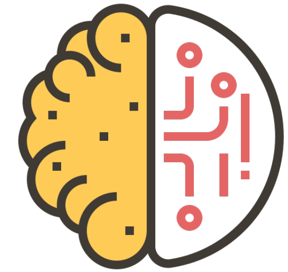


$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus})$, $\mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

Knowledge (TBox)



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

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

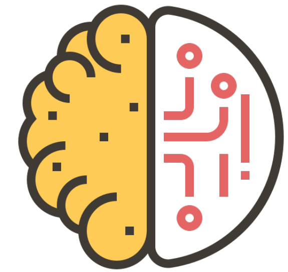
$$\mathit{Narcist} \sqsubseteq \exists \mathit{loves}.\mathsf{Self}$$

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)

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

Knowledge (TBox)



$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus})$, $\mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

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

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

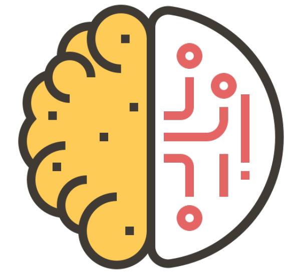
$$\mathit{Narcist} \sqsubseteq \exists \mathit{loves}.\mathsf{Self}$$

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)

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

Knowledge (TBox)



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

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

$Narcissist(\texttt{Narcissus})$

$$Mortal \sqsubseteq \neg Diety$$

$$\top \sqsubseteq \exists hasParent.Male \sqcap \exists hasParent.Female$$

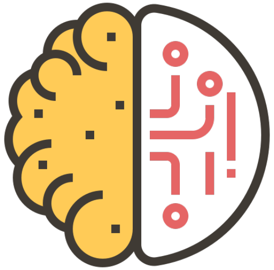
$$Narcist \sqsubseteq \exists loves.\mathsf{Self}$$

Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)

Knowledge (TBox)

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



$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus}), \mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

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

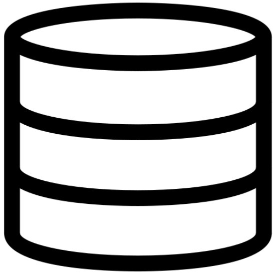
$$\top \sqsubseteq \exists \mathit{hasParent}.\mathit{Male} \sqcap \exists \mathit{hasParent}.\mathit{Female}$$

$$\mathit{Narcist} \sqsubseteq \exists \mathit{loves}.\mathsf{Self}$$

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

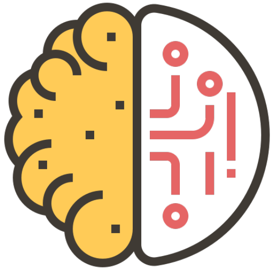
Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)



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

Knowledge (TBox)



$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus})$, $\mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

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

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

$$\mathit{Narcist} \sqsubseteq \exists \mathit{loves}.\mathsf{Self}$$

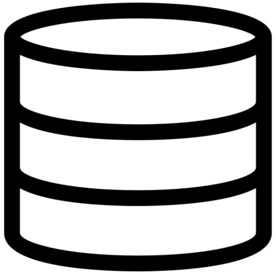
Conjunctive Queries: Give me IDs of all candidates who applied for "computer science".

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

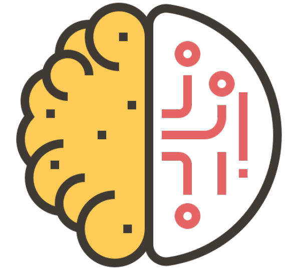
Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)

Knowledge (TBox)



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



$\mathit{hasParent}(\mathtt{Heracles}, \mathtt{Zeus})$

$\mathit{Diety}(\mathtt{Zeus}), \mathit{Female}(\mathtt{Rhea})$

$\mathit{Narcissist}(\mathtt{Narcissus})$

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

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

$$\mathit{Narcist} \sqsubseteq \exists \mathit{loves}.\mathsf{Self}$$

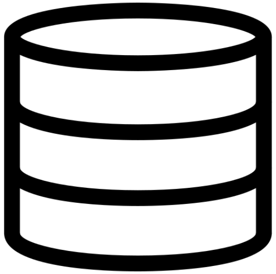
Conjunctive Queries: Give me IDs of all candidates who applied for "computer science".

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

$\rightsquigarrow \quad \varphi(i)$

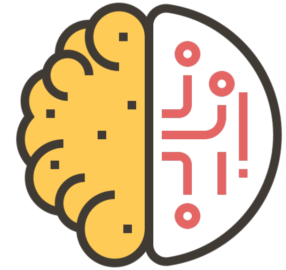
Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)



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

Knowledge (TBox)



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

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

$Narcissist(\mathtt{Narcissus})$

$$Mortal \sqsubseteq \neg Diety$$

$$\top \sqsubseteq \exists hasParent.Male \sqcap \exists hasParent.Female$$

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

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

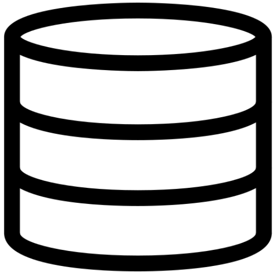
```
SELECT CandID
FROM Candidate
WHERE Major = "Computer Science"
```

$\rightsquigarrow \quad \varphi(i)$

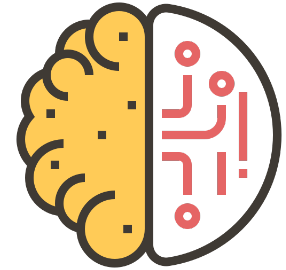
Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)

Knowledge (TBox)



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



$\mathit{hasParent}(\texttt{Heracles}, \texttt{Zeus})$

$\mathit{Diety}(\texttt{Zeus})$, $\mathit{Female}(\texttt{Rhea})$

$\mathit{Narcissist}(\texttt{Narcissus})$

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

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

$$\mathit{Narcist} \sqsubseteq \exists \mathit{loves}.\mathsf{Self}$$

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

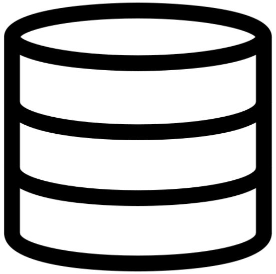
```
SELECT CandID
FROM Candidate
WHERE Major = "Computer Science"
```

$\rightsquigarrow \varphi(i)$

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

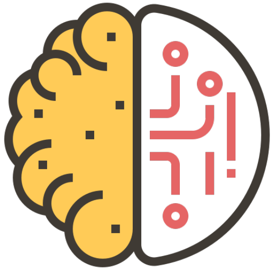
Running example: Greek mythology $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base

Database (ABox)



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

Knowledge (TBox)



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

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

$Narcissist(\texttt{Narcissus})$

$$Mortal \sqsubseteq \neg Diety$$

$$\top \sqsubseteq \exists hasParent.Male \sqcap \exists hasParent.Female$$

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

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

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

$\rightsquigarrow \quad \varphi(i)$

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

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

Our motivation: what features make CQ answering hard for $\mathcal{ALC}$?

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]

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$

$\mathit{Car} \sqsubseteq (= 4).\texttt{hasPart}\,\mathit{Wheel}$

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (=4).\texttt{hasPart}\,\mathit{Wheel}$

- 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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (=4).\texttt{hasPart}\,\mathit{Wheel}$

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

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (=4).\texttt{hasPart}\,\mathit{Wheel}$

- 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]

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (= 4).\texttt{hasPart}\,\mathit{Wheel}$

- 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:

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\quad$ $Car \sqsubseteq (= 4).\texttt{hasPart}\, Wheel$

- 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]

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]

$\mathtt{hasMother} \subseteq \mathtt{hasParent}$ $\qquad$ $Car \sqsubseteq (=4).\mathtt{hasPart}\, Wheel$

- 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 Self **operator? Is it harmless?**

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\quad$ $Car \sqsubseteq (=4).\texttt{hasPart}\,Wheel$

- 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 Self operator? Is it harmless?

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

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\quad$ $\mathit{Car} \sqsubseteq (=4).\texttt{hasPart}\,\mathit{Wheel}$

- 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 Self operator? Is it harmless?

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

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

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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $Car \sqsubseteq (=4).\texttt{hasPart}\, Wheel$

- 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 Self operator? Is it harmless?

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

- The complexity of satisfiability stays the same, even for very expressive $\mathcal{Z}$ family, a.k.a. $\mathcal{ALCH}b^{\mathsf{Self}}_{\mathsf{reg}}$
- 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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $Car \sqsubseteq (=4).\texttt{hasPart}\, Wheel$

- 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 Self operator? Is it harmless?

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

- The complexity of satisfiability stays the same, even for very expressive $\mathcal{Z}$ family, a.k.a. $\mathcal{ALCH}b^{\mathsf{Self}}_{\mathsf{reg}}$
- 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]

$\texttt{hasMother} \subseteq \texttt{hasParent}$ $\qquad$ $\mathit{Car} \sqsubseteq (= 4).\texttt{hasPart}\,\mathit{Wheel}$

- 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 Self operator? Is it harmless?

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

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

1. Some of them do n

hasMother ⊆ hasPar

- Also arithmetic and
- As well as regular e
- And even a tamed

]

2. Some of them incre

E.g. inverses [Lutz'07], et al.'16]

W ss?

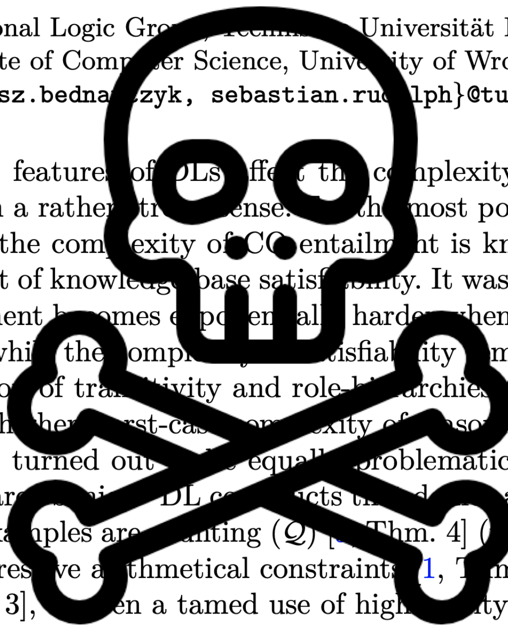
Self is suppo $(d, d) \in r^{\mathcal{I}}$ }

- The complexity of a.k.a. $\mathcal{ALCH}b^{\mathsf{Self}}_{\mathsf{reg}}$
- Easy to accommod
- Self is present in O VC'08]

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

Bartosz Bednarczyk[1,2] and Sebastian Rudolph[1]

[1] Computational Logic Gro Universität Dresden, Germany
[2] Institute of Comp er Science, Unive ity of Wrocław, Poland
{bartosz.bedna zyk, sebastian.ru lph}@tu-dresden.de



Various modelling feature of DLs affe t th complexity of conjunctive query (CQ) entailment in a rather tr ense. most popular basic description logic (DL), $\mathcal{ALC}$, the com exity of CQ entailm nt is known to be ExpTime-complete, as is that of knowledg base satisf ty. It was first shown in [9, Thm. 2] that CQ entailment b omes e po en al harder hen $\mathcal{ALC}$ is extended with inverse roles ($\mathcal{I}$), whi the omp isfiab ity mains the same. Shortly after, a combinatio of tra tivity and role- archie ($\mathcal{SH}$) was shown to be another culprit of h her rst-ca o lexity of easo ng [5, Thm. 1]. Finally, also nominals ($\mathcal{O}$) turned out e equall roblematic [10, Thm. 9]. On the other hand, there ar DL co cts t d affect the complexity of CQ entailment. Exa ples are nting ($\mathcal{Q}$) [Thm. 4] (e complexity stays the same even for expres ve a thmetical constraints 1, T m. 21]), role-hierarchies alone ($\mathcal{H}$) [6, Cor. 3], en a tamed use of hig ty relations [2, Thm. 20].

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

The skull icon by ©Freepik from flaticon.com.

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

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

Consequences?

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

Consequences?

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

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

Consequences?

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

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

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

Consequences?

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

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

Consequences?

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

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

Consequences?

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

† $\forall x_1\, (\mathrm{self_r}(x_1) \rightarrow \exists x_2 [\mathrm{R}(x_1, x_2) \wedge x_1{=}x_2]) \;\wedge\; \forall x_1 \forall x_2\, (\mathrm{R}(x_1, x_2) \rightarrow [x_1{=}x_2 \rightarrow \mathrm{self_r}(x_2)])$

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

Consequences?

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

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

Consequences?

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

Proof scheme?

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

Consequences?

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

Proof scheme?

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

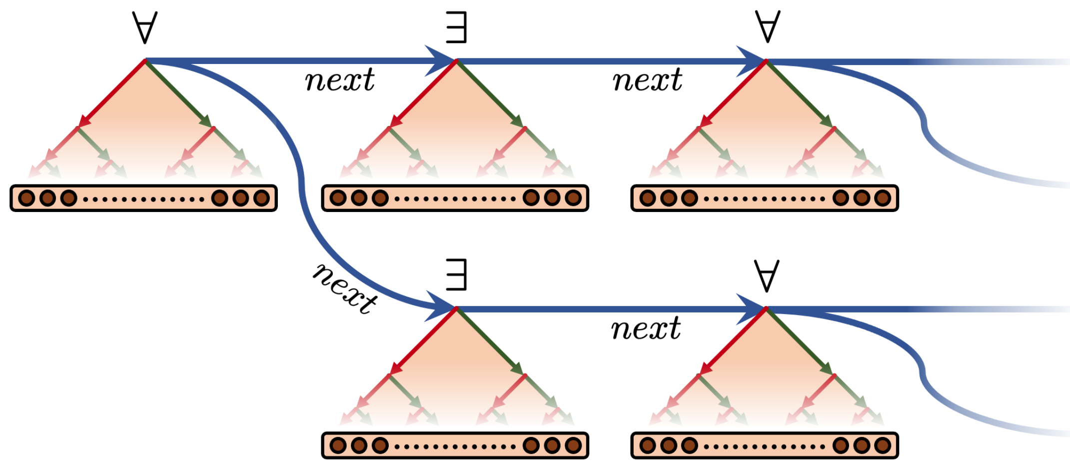
Conjunctive query entailment over $\mathcal{ALC}_{\mathsf{Self}}$ TBoxes is $2\textsc{ExpTime}$-hard.

Consequences?

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

Proof scheme?

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



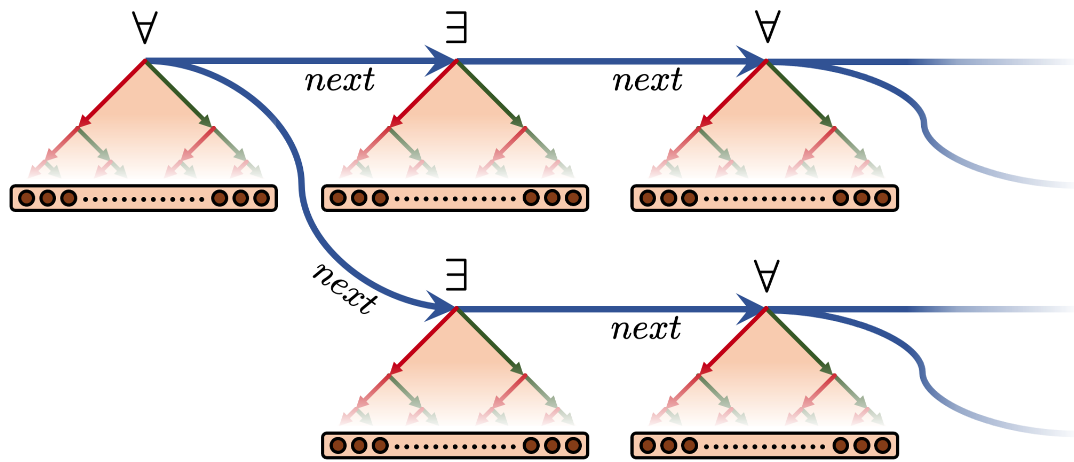
Conjunctive query entailment over $\mathcal{ALC}_{\mathsf{Self}}$ TBoxes is 2EXPTIME-hard.

Consequences?

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

Proof scheme?

- A reduction from the acceptance problem for the empty-tape AEXPSPACE TMs.



- The models of an $\mathcal{ALC}_{\mathsf{Self}}$-KB $\mathcal{K}_{\mathcal{M}}$ describe possibly faulty runs of a given ATM $\mathcal{M}$.

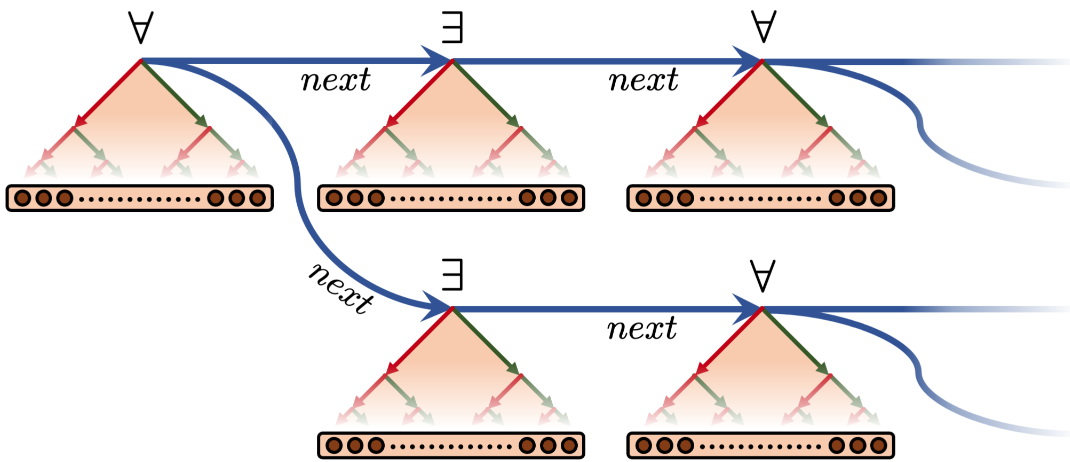
Conjunctive query entailment over $\mathcal{ALC}_{\mathsf{Self}}$ TBoxes is 2ExpTime-hard.

Consequences?

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

Proof scheme?

- A reduction from the acceptance problem for the empty-tape AExpSpace TMs.



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

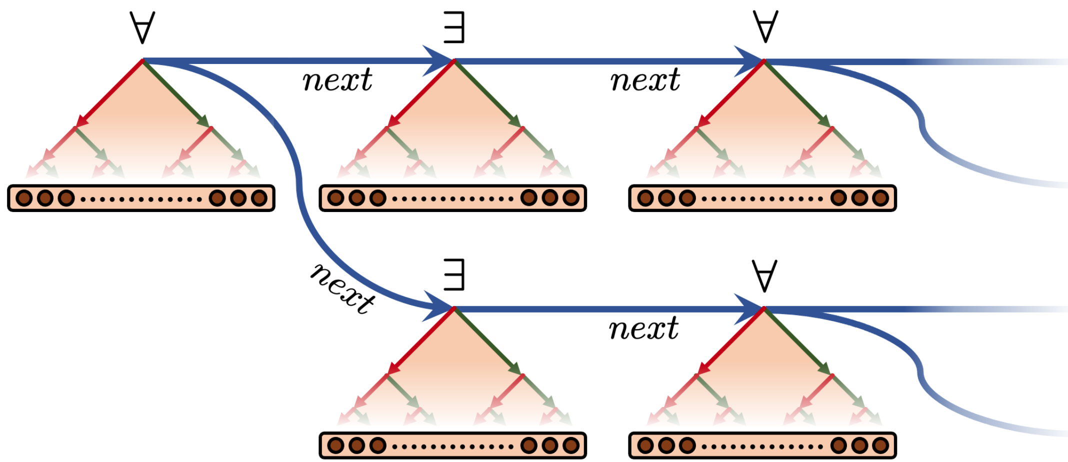
Conjunctive query entailment over $\mathcal{ALC}_{\mathsf{Self}}$ TBoxes is $2\textsc{ExpTime}$-hard.

Consequences?

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

Proof scheme?

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



- The models of an $\mathcal{ALC}_{\mathsf{Self}}$-KB $\mathcal{K}_{\mathcal{M}}$ describe possibly faulty runs of a given ATM $\mathcal{M}$.
- A CQ $q_{\mathcal{M}}$ detects mismatches in the consecutive transitions.
- $\mathcal{K}_{\mathcal{M}} \not\models q_{\mathcal{M}}$ iff there is a (non-faulty) accepting run of $\mathcal{M}$.

Proof ideas: Our encoding

Proof ideas: Our encoding

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

Proof ideas: Our encoding

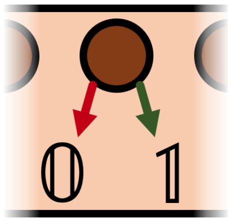
- We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.
- Novelty: nodes will be decorated with certain self-loops.

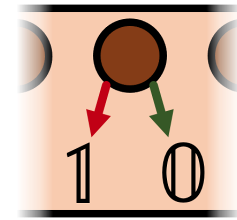
Proof ideas: Our encoding

- 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

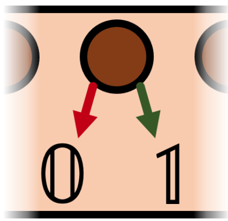
- We encode configurations as full-binary trees of depth $n+1$ with their roots connected with next-role.
- Novelty: nodes will be decorated with certain self-loops.
- To avoid a seemingly required disjunction in our CQs the tape content is stored implicitly with:

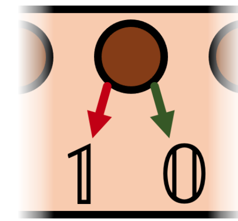




Proof ideas: Our encoding

- 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:

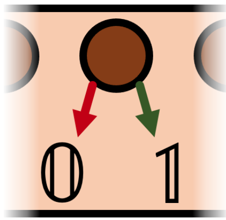


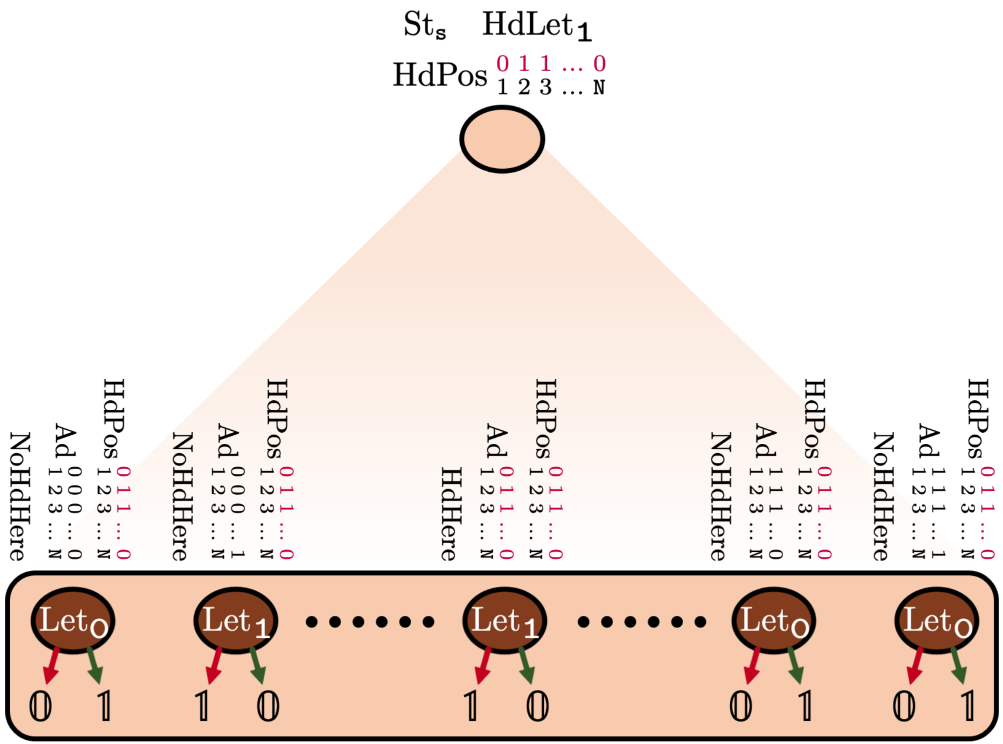


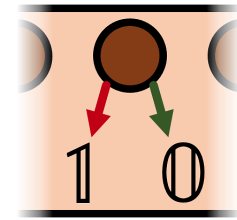
- All other details are as one may expect. See: https://arxiv.org/abs/2106.15150

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:



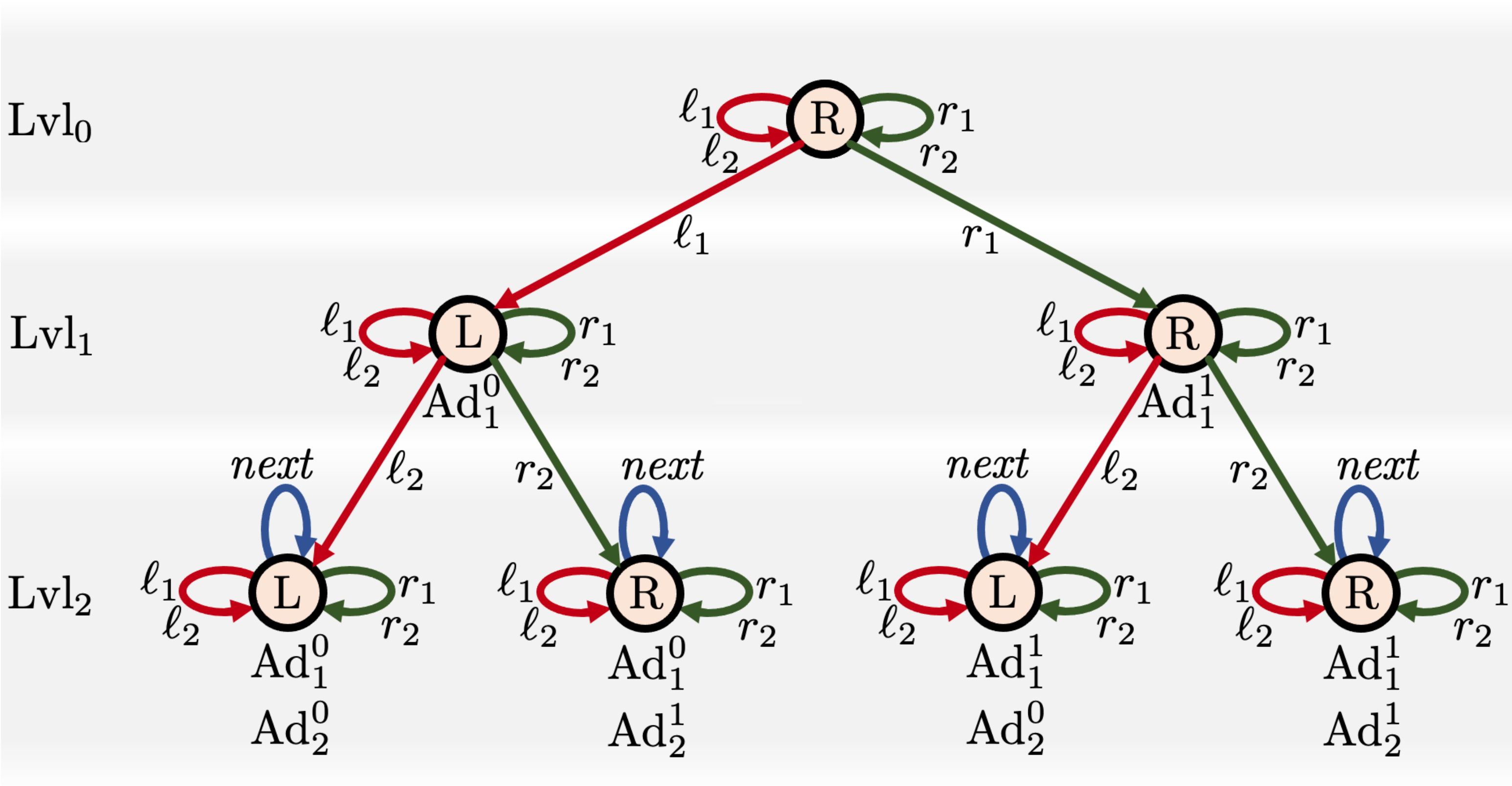




- All other details are as one may expect. See: https://arxiv.org/abs/2106.15150

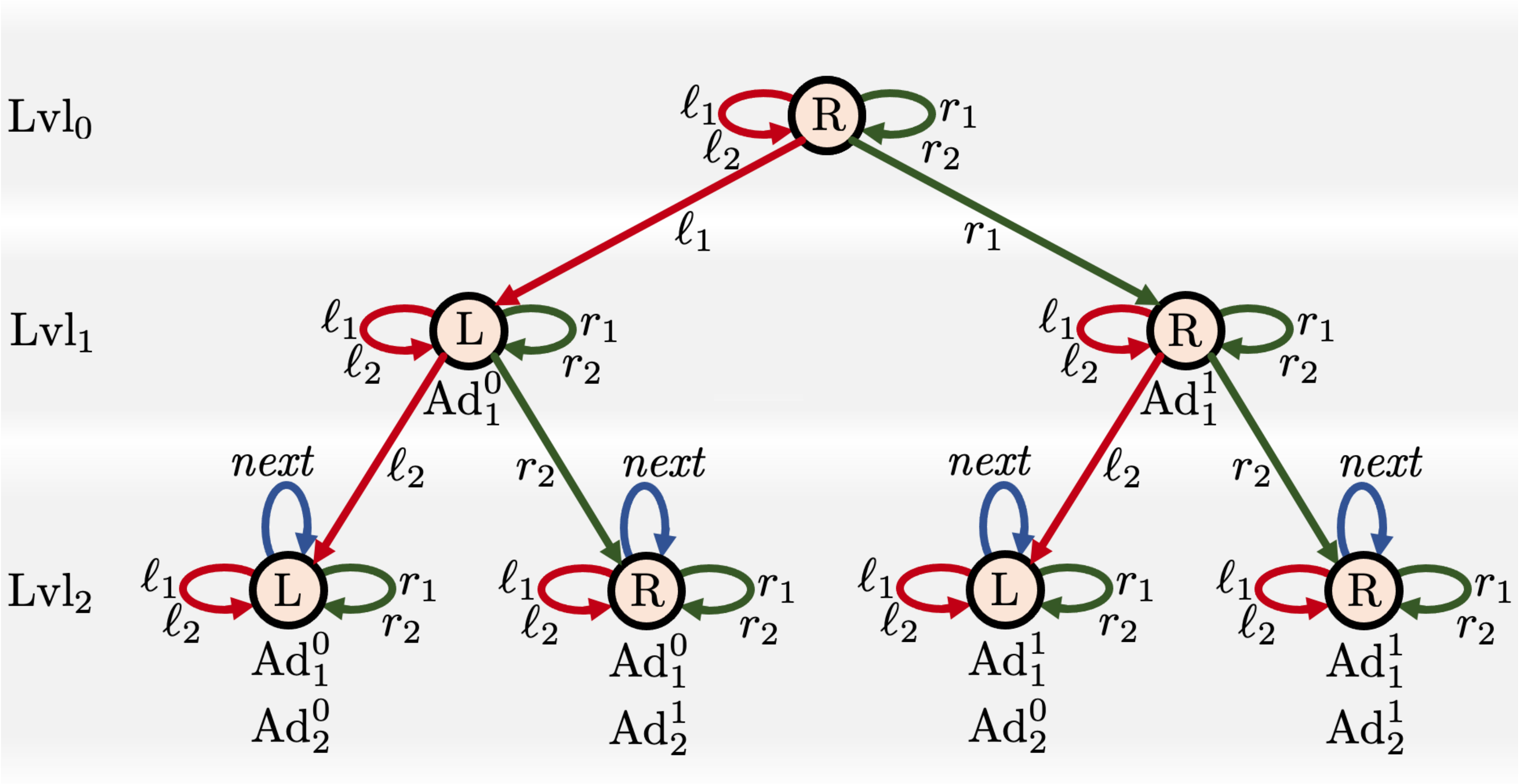
Trick no. 1: A single root-to-leaves conjunctive query

Trick no. 1: A single root-to-leaves conjunctive query



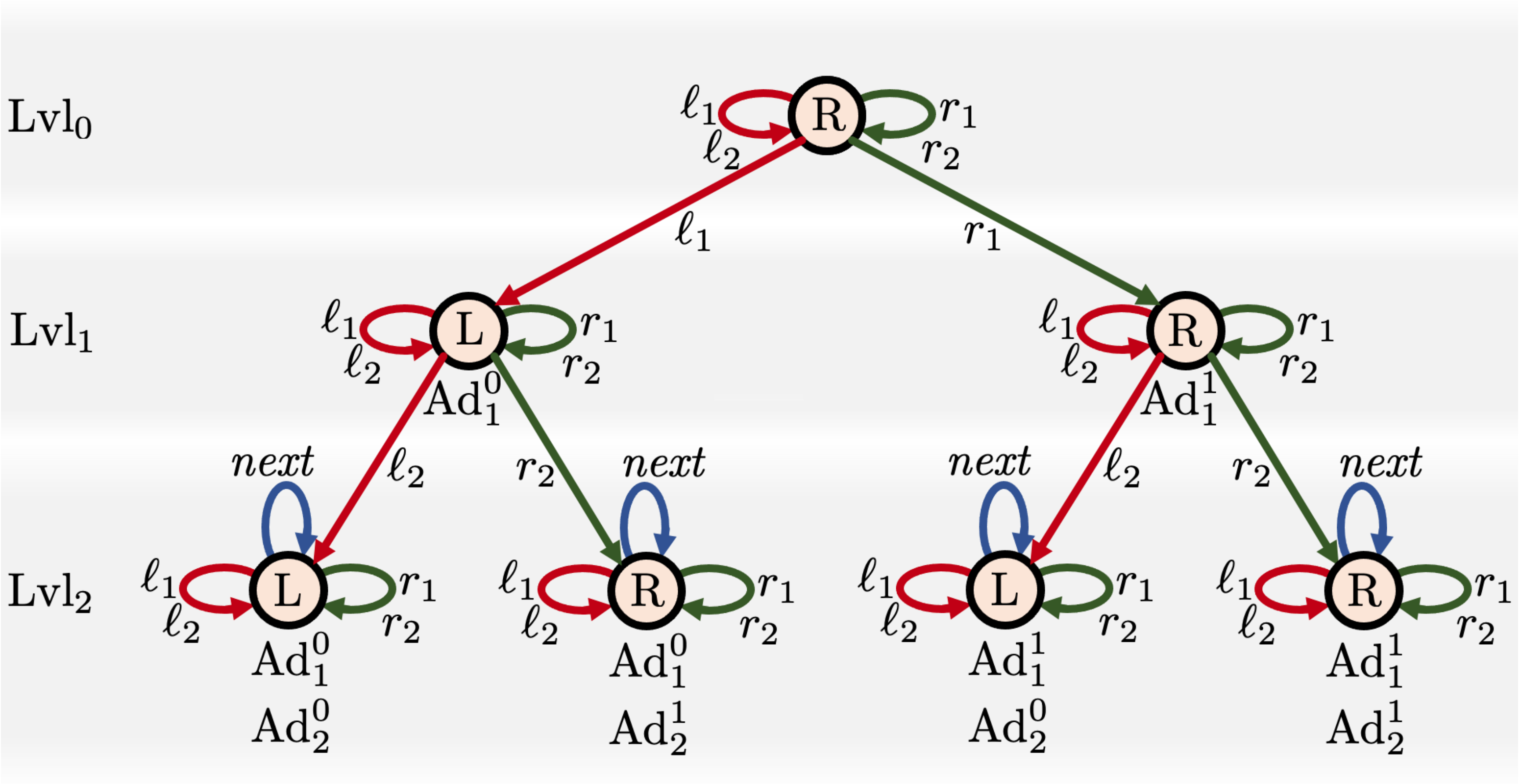
Trick no. 1: A single root-to-leaves conjunctive query

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



Trick no. 1: A single root-to-leaves conjunctive query

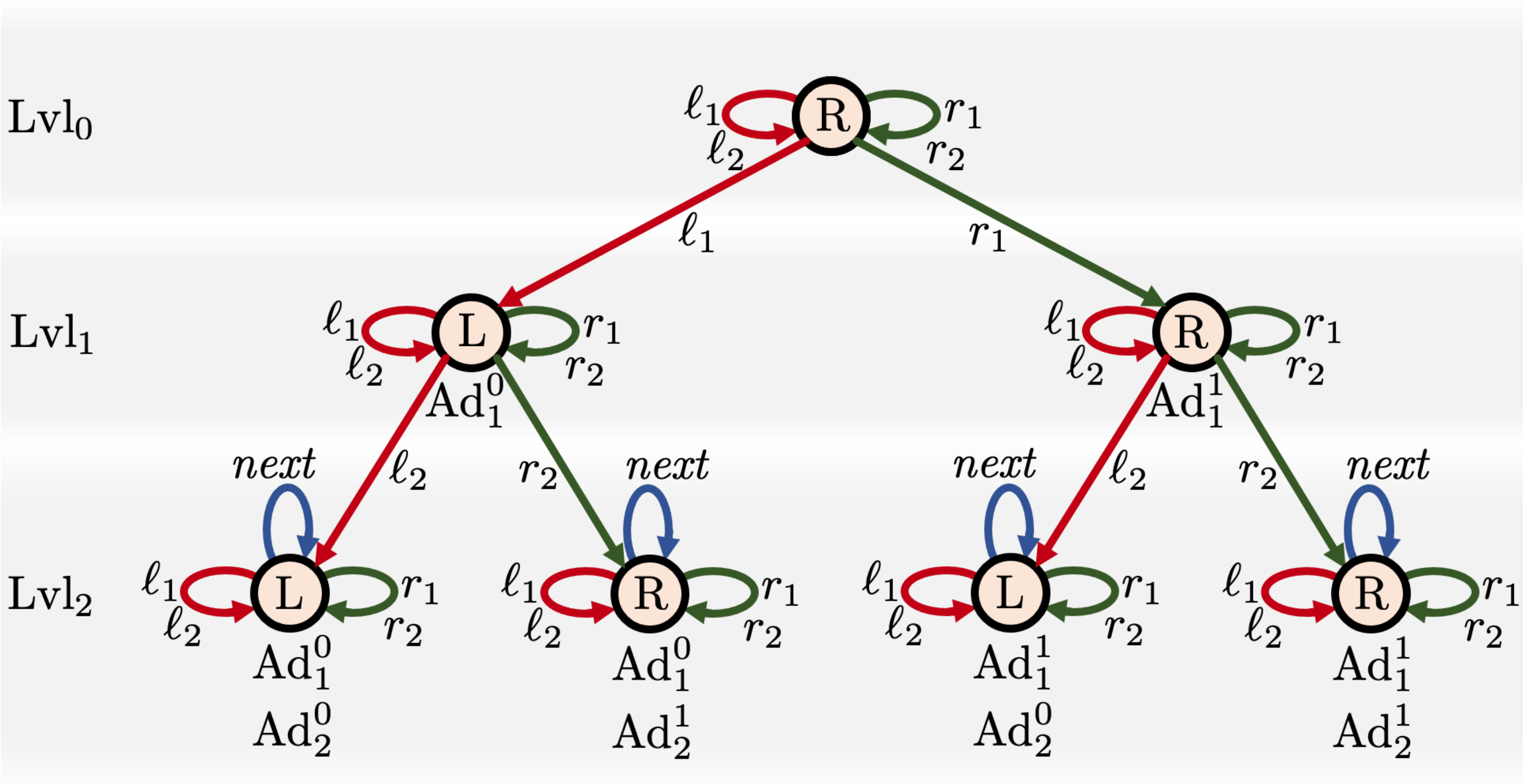
Goal: Design a CQ $q(x, y)$ such that x matches the root and y matches any of the leaves.



$$\exists x_1 \exists x_2 \exists x_3 \; Lvl_0(x) \wedge \ell_1(x, x_1) \wedge \mathrm{r}_1(x_1, x_2) \wedge \ell_2(x_2, x_3) \wedge \mathrm{r}_2(x_3, y) \wedge Lvl_2(y)$$

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.

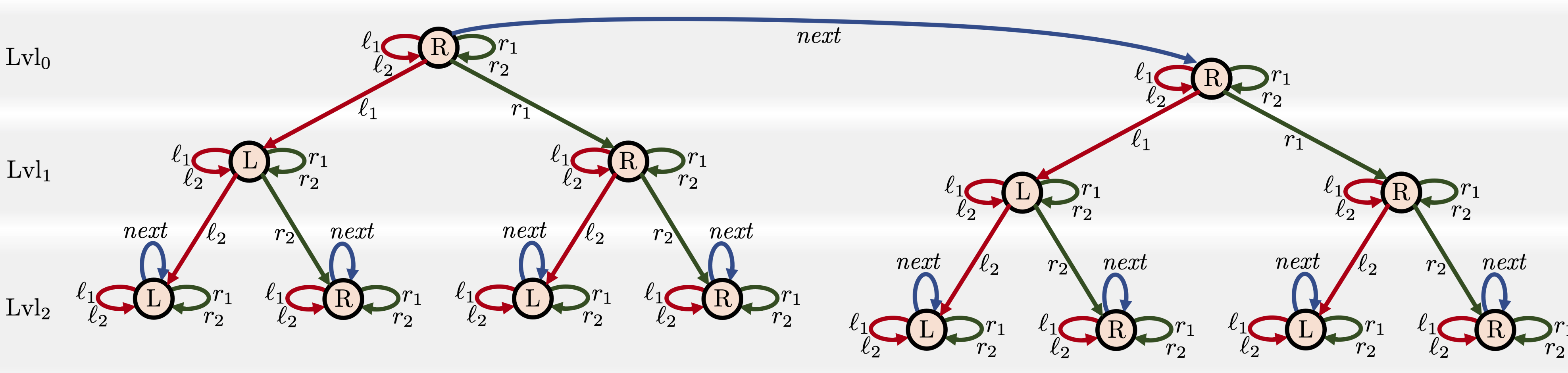


$$\exists x_1 \exists x_2 \exists x_3\ Lvl_0(x) \wedge \ell_1(x, x_1) \wedge \mathrm{r}_1(x_1, x_2) \wedge \ell_2(x_2, x_3) \wedge \mathrm{r}_2(x_3, y) \wedge Lvl_2(y)$$

For brevity we write: $(Lvl_0?; \ell_1; \mathrm{r}_1; \ell_2; \mathrm{r}_2; Lvl_2?)(x, y)$.

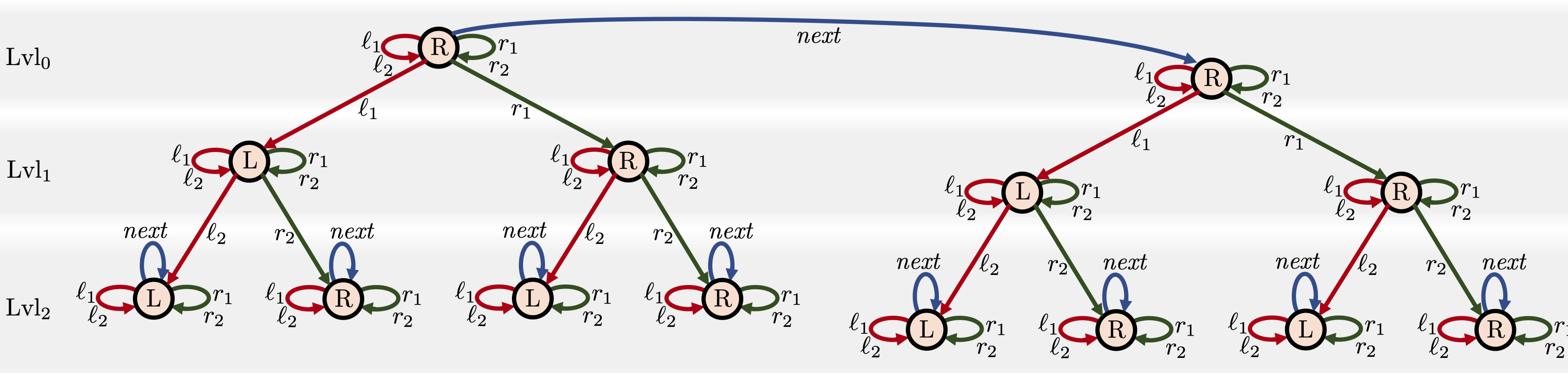
Trick no. 2: Synchronisation of leaves among two trees

Trick no. 2: Synchronisation of leaves among two trees



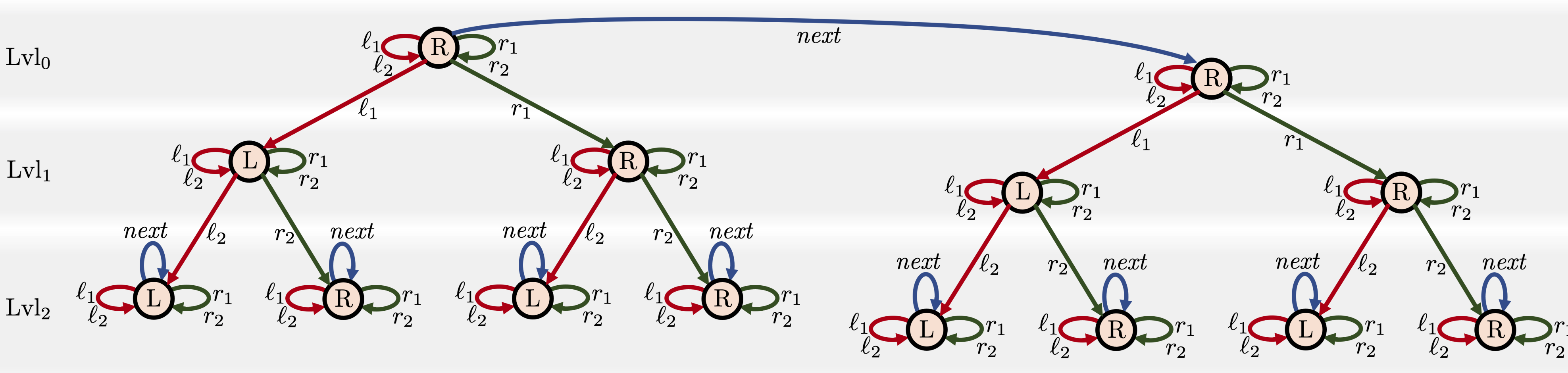
Trick no. 2: Synchronisation of leaves among two trees

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



Trick no. 2: Synchronisation of leaves among two trees

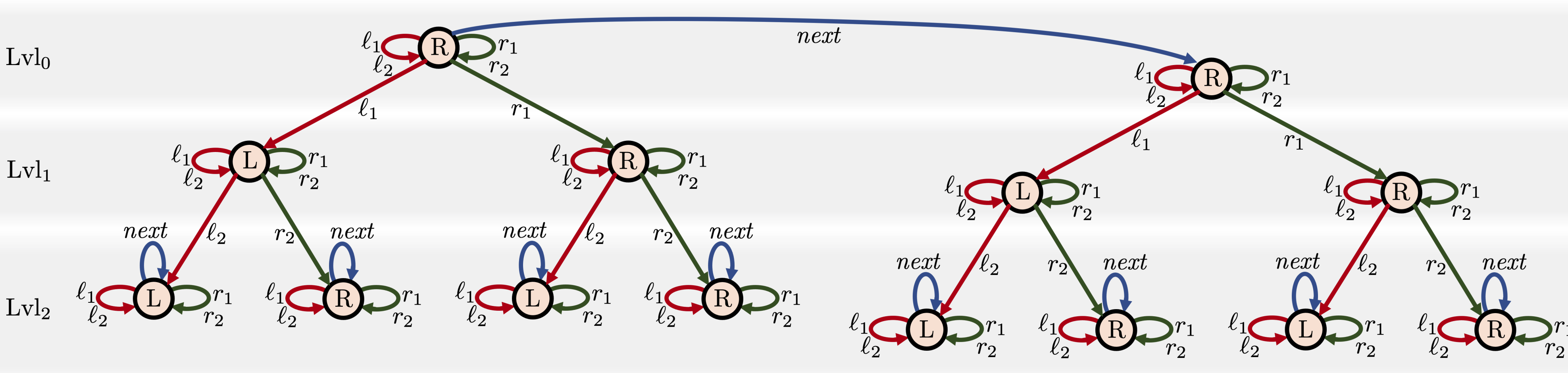
Goal: Design a CQ $q(x, y)$ that matches leaves x, y with equal addresses.



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.

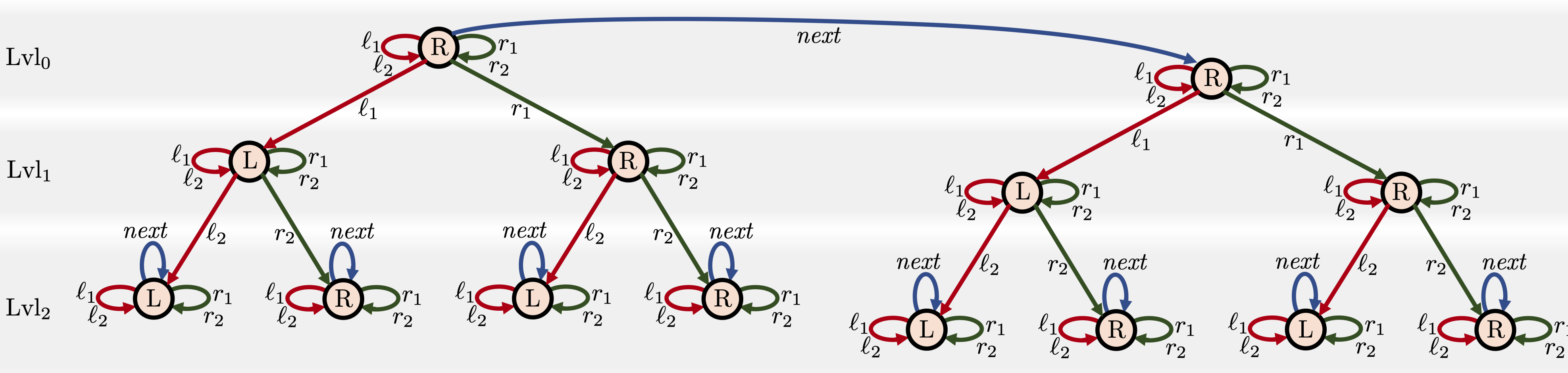


Select two leaves located in different trees:

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

Trick no. 2: Synchronisation of leaves among two trees

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



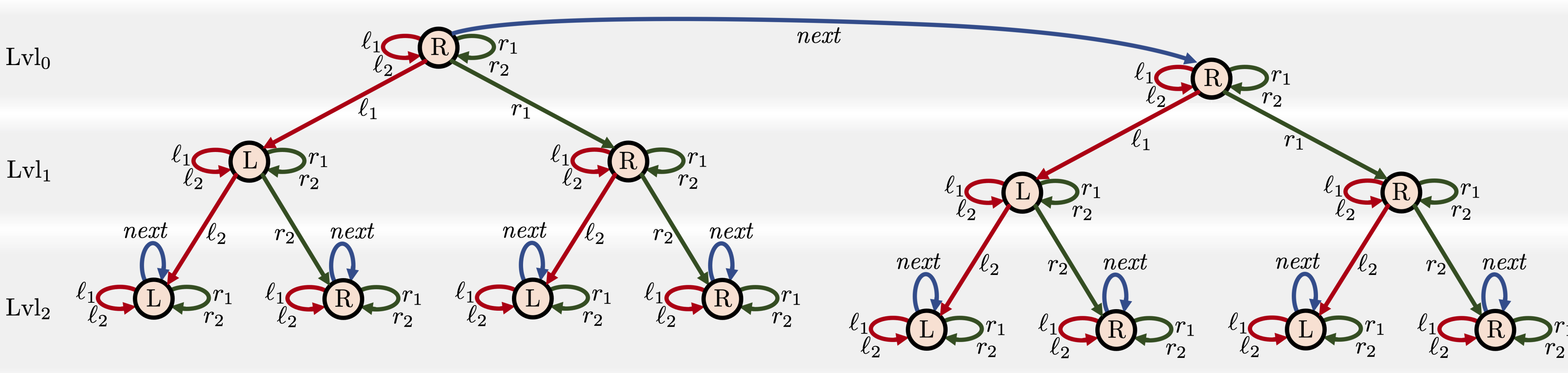
Select two leaves located in different trees:

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

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:

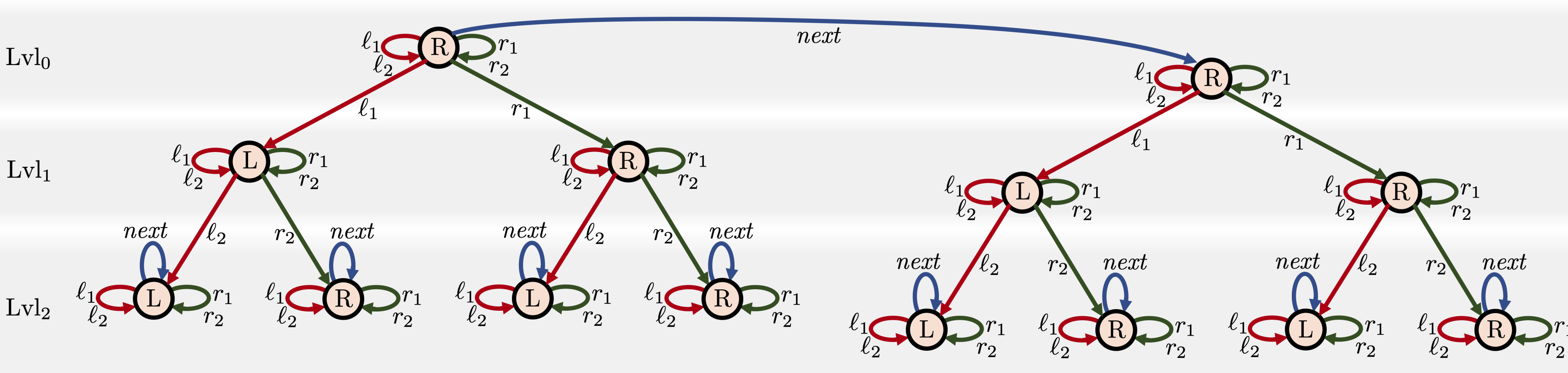
$$(\mathrm{Lvl}_2?; r_2^-; \ell_2^-; r_1^-; \ell_1^-; \mathrm{Lvl}_0?; \mathit{next}; \mathrm{Lvl}_0?; \ell_1; r_1; \ell_2; r_2; \mathrm{Lvl}_2?)(x, y)$$

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

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

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:

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

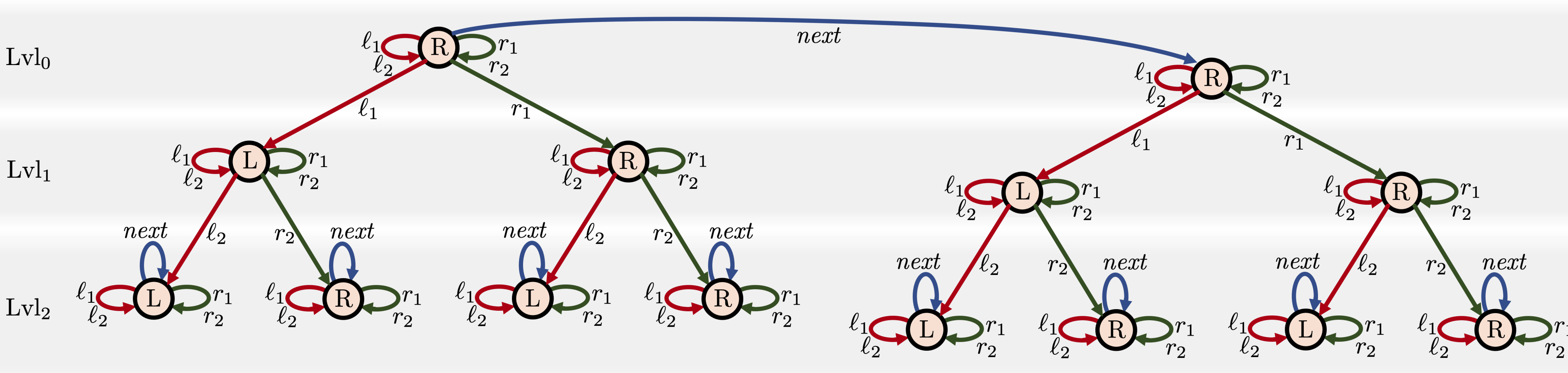
Impose that they have the same first bit of their address:

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

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:

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

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

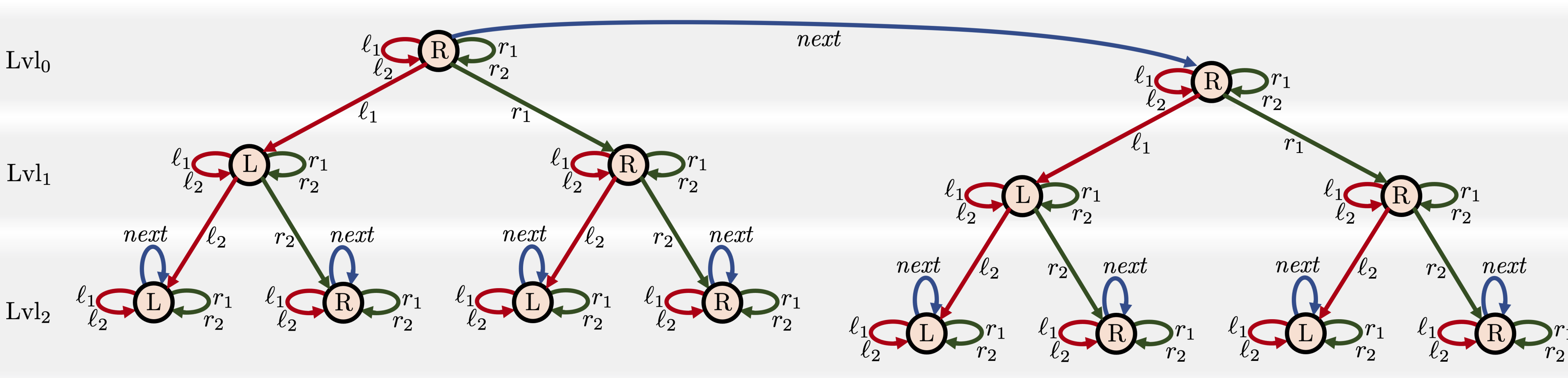
$$\wedge\ (r_2^-; \ell_2^-; \ell_1^-; next; \ell_1; \ell_2; r_2; \mathrm{Lvl}_2?; r_2^-; \ell_2^-; r_1^-; next; r_1; \ell_2; r_2)(x, y)$$

as well as the same second bit of their address:

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

The end: Thanks for your attention!

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



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

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

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

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