REWRITING $\mathcal{ALCHIQ}$ TO DISJUNCTIVE EXISTENTIAL RULES

David Carral Markus Krötzsch

Knowledge-Based Systems

TU Dresden

Full paper and video at
https://tud.link/h5l5

IJCAI 2020

Rewriting DLs to Rules

Given a theory $\mathcal{T}_1$ in a logic $\mathcal{L}_1$
and a theory $\mathcal{T}_2$ in a logic $\mathcal{L}_2$,
$\mathcal{T}_2$ is a **rewriting** of $\mathcal{T}_1$ if,

$$\mathcal{T}_1, \mathcal{F} \models \varphi \text{ iff } \mathcal{T}_2, \mathcal{F} \models \varphi$$

for every set $\mathcal{F}$ of ground facts and every ground fact φ over the signature of $\mathcal{T}_1$.

Rules and DLs

Rule languages we encounter:

- Datalog: the “simplest rules conceivable”, e.g.,
  $A(x) \wedge R(x, y) \rightarrow B(y)$
- Datalog$^{\vee}$: Datalog + $\vee$ in heads
- Datalog$^{\exists}$: Datalog + $\exists$ in heads, a.k.a. existential rules
- Datalog$^{\vee\exists}$: Datalog + $\vee$ and $\exists$ in heads

Rules and DLs

Rule languages we encounter:

- Datalog: the "simplest rules conceivable", e.g.,
  $A(x) \wedge R(x,y) \to B(y)$
- Datalog$^{\vee}$: Datalog + $\vee$ in heads
- Datalog$^{\exists}$: Datalog + $\exists$ in heads, a.k.a. existential rules
- Datalog$^{\vee\exists}$: Datalog + $\vee$ and $\exists$ in heads

The DL $\mathcal{ALCHIQ}$ can be normalised* to rules of nine forms:

$A(x) \wedge B(x) \to C(x)$	$A \sqcap B \sqsubseteq C$	$A(x) \to B(x) \vee C(x)$	$A \sqsubseteq B \sqcup C$
$A(x) \wedge R(x,y) \to B(y)$	$A \sqsubseteq \forall R.B$	$A(x) \to \exists y.R(x,y) \wedge B(y)$	$A \sqsubseteq \exists R.B$
$R(x,y) \wedge R(x,z) \to y \approx z$	$\top \sqsubseteq \leqslant 1\, R.\top$	$R(x,y) \to S(x,y) \vee V(x,y)$	$R \sqsubseteq S \sqcup V$
$R(x,y) \wedge S(x,y) \to V(x,y)$	$R \sqcap S \sqsubseteq V$	$R(y,x) \to S(x,y)$	$R^{-} \sqsubseteq S$

$$A(x) \wedge R(x,y) \wedge B(y) \to S(x,y) \qquad A \circ R \circ B \sqsubseteq S$$

*) this is polynomial under unary encoding of numbers

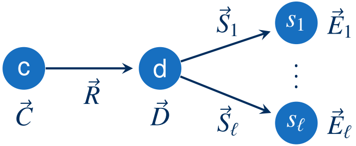
Work	Source	Target	Size	Rules
Hustadt et al. [2007]	$\mathcal{ALCHIQ}$	$\text{Datalog}^{\vee}$	exp.	bounded
Eiter et al. [2012]	Horn-$\mathcal{SHIQ}$	Datalog	exp.	bounded
Rudolph et al. [2012]	$\mathcal{SHIQ}\text{b}_s$	$\text{Datalog}^{\vee}$	exp.	bounded
Bienvenu et al. [2014]	$\mathcal{SHI}$	$\text{Datalog}^{\vee}$	exp.	bounded
Carral et al. [2018]	Horn-$\mathcal{ALCHOIQ}$	Datalog	exp.	bounded
Carral et al. [2019b]	Horn-$\mathcal{SHIQ}$	Datalog	exp.	bounded
	Horn-$\mathcal{SRIQ}$	Datalog	2exp.	bounded
Ortiz et al. [2010]	Horn-$\mathcal{ALCHOIQ}$	Datalog	poly.	unbounded
Ahmetaj et al. [2016]	$\mathcal{ALCHIO}$	$\text{Datalog}^{\vee}$	poly.	unbounded
Krötzsch [2011]	$\mathcal{EL}^{++}$	Datalog	poly.	bounded
Carral et al. [2019a]	Horn-$\mathcal{ALC}$	$\text{Datalog}^{\exists}$	poly.	bounded

Work	Source	Target	Size	Rules
Hustadt et al. [2007]	$\mathcal{ALCHIQ}$	Datalog$^{\vee}$	exp.	bounded
Eiter et al. [2012]	Horn-$\mathcal{SHIQ}$	Datalog	exp.	bounded
Rudolph et al. [2012]	$\mathcal{SHIQ}b_s$	Datalog$^{\vee}$	exp.	bounded
Bienvenu et al. [2014]	$\mathcal{SHI}$	Datalog$^{\vee}$	exp.	bounded
Carral et al. [2018]	Horn-$\mathcal{ALCHOIQ}$	Datalog	exp.	bounded
Carral et al. [2019b]	Horn-$\mathcal{SHIQ}$	Datalog	exp.	bounded
	Horn-$\mathcal{SRIQ}$	Datalog	2exp.	bounded
Ortiz et al. [2010]	Horn-$\mathcal{ALCHOIQ}$	Datalog	poly.	unbounded
Ahmetaj et al. [2016]	$\mathcal{ALCHIO}$	Datalog$^{\vee}$	poly.	unbounded
Krötzsch [2011]	$\mathcal{EL}^{++}$	Datalog	poly.	bounded
Carral et al. [2019a]	Horn-$\mathcal{ALC}$	Datalog$^{\exists}$	poly.	bounded
NEW!	$\mathcal{ALCHIQ}$	Datalog$^{\vee}$	poly.	unbounded
	$\mathcal{ALCHIQ}$	Datalog$^{\vee\exists}$	poly.	bounded
	Horn-$\mathcal{ALCHIQ}$	Datalog$^{\exists}$	poly.	bounded

From $\mathcal{ALCHIQ}$ to Datalog$^{\vee}$ using types

From $\mathcal{ALCHIQ}$ to Datalog$^\vee$ using types

We decompose $\mathcal{ALCHIQ}$ models into structures of bounded size, i.e. "types":

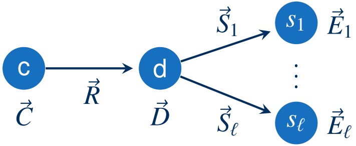


A type is given by a fixed number of:

- sets of concepts $\vec{C}$, $\vec{D}$, $\vec{E}_1, \ldots, \vec{E}_\ell$
- sets of (inverse) relations $\vec{R}$, $\vec{S}_1, \ldots, \vec{S}_\ell$
- where ℓ is the number of $\mathcal{ALCHIQ}$ axioms of form $A(x) \to \exists y.R(x, y) \wedge B(y)$

From $\mathcal{ALCHIQ}$ to Datalog$^{\vee}$ using types

We decompose $\mathcal{ALCHIQ}$ models into structures of bounded size, i.e. "types":



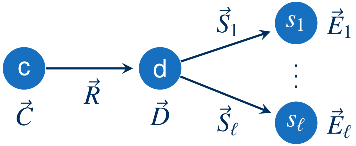
A type is given by a fixed number of:

- sets of concepts $\vec{C}$, $\vec{D}$, $\vec{E}_1,\ldots,\vec{E}_\ell$
- sets of (inverse) relations $\vec{R}$, $\vec{S}_1,\ldots,\vec{S}_\ell$
- where ℓ is the number of $\mathcal{ALCHIQ}$ axioms of form $A(x) \to \exists y.R(x,y) \wedge B(y)$

$\Rightarrow$ We can represent sets as bit vectors and store types as facts Type$(\underbrace{1,0,1,0,1,0,\ldots}_{\text{suitably long bit vector}})$

From $\mathcal{ALCHIQ}$ to Datalog$^\vee$ using types

We decompose $\mathcal{ALCHIQ}$ models into structures of bounded size, i.e. "types":



A type is given by a fixed number of:

- sets of concepts $\vec{C}$, $\vec{D}$, $\vec{E}_1, \ldots, \vec{E}_\ell$
- sets of (inverse) relations $\vec{R}$, $\vec{S}_1, \ldots, \vec{S}_\ell$
- where ℓ is the number of $\mathcal{ALCHIQ}$ axioms of form $A(x) \to \exists y.R(x, y) \wedge B(y)$

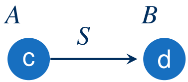
⇒ We can represent sets as bit vectors and store types as facts Type$(\underbrace{1, 0, 1, 0, 1, 0, \ldots}_{\text{suitably long bit vector}})$

An $\mathcal{ALCHIQ}$ ontology is satisfiable iff it admits a consistent set of types.

⇒ Datalog$^\vee$ encoding: axiomatise required types and consistency conditions

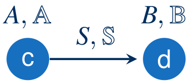
From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

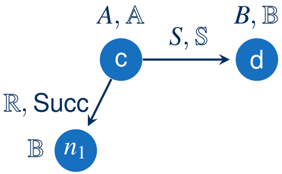
We construct a tableau-like structure:



$$A(x) \to \mathbb{A}(x)$$
$$B(x) \to \mathbb{B}(x)$$
$$S(x, y) \to \mathbb{S}(x, y)$$

From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

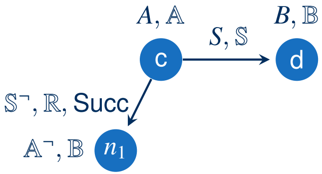
We construct a tableau-like structure:



$\mathbb{A}(x) \to \exists y.\mathbb{R}(x,y) \wedge \mathbb{B}(y) \wedge \mathsf{Succ}(x,y)$ $\qquad$ $A \sqsubseteq \exists R.B$

From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

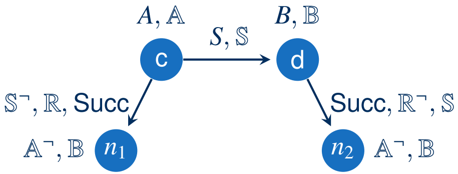
We construct a tableau-like structure:



$$\text{Succ}(x, y) \to \mathbb{S}(x, y) \vee \mathbb{S}^{\neg}(x, y)$$
$$\text{Unnamed}(x) \to \mathbb{A}(x) \vee \mathbb{A}^{\neg}(x)$$

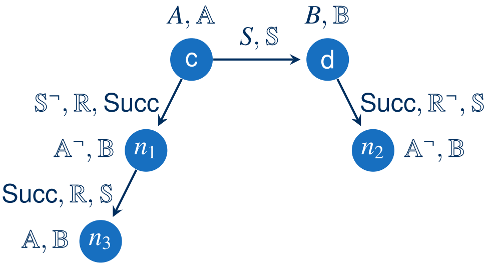
From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



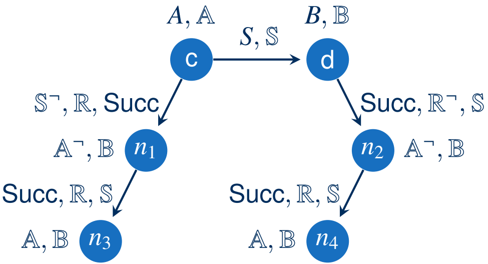
From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



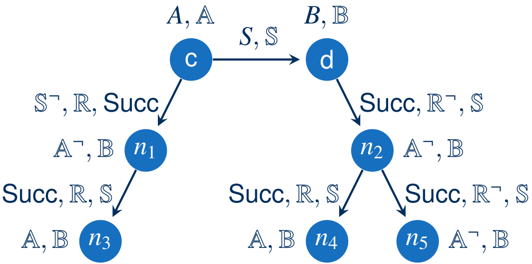
From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



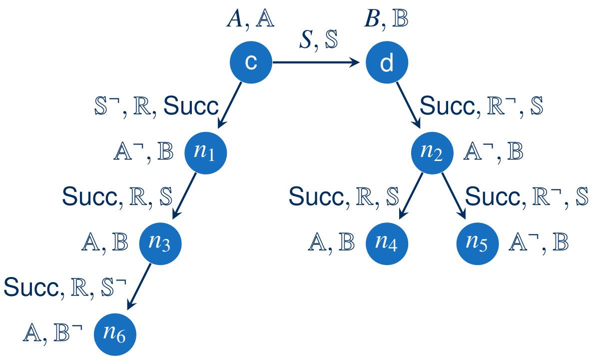
From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



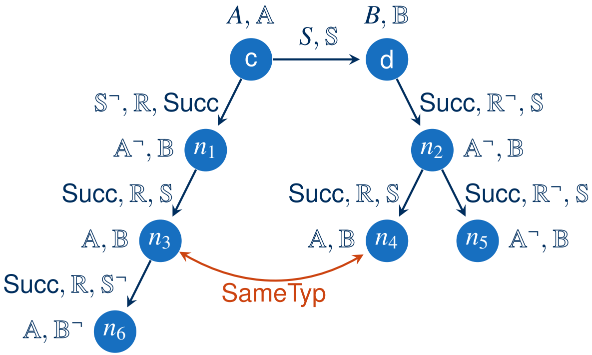
From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



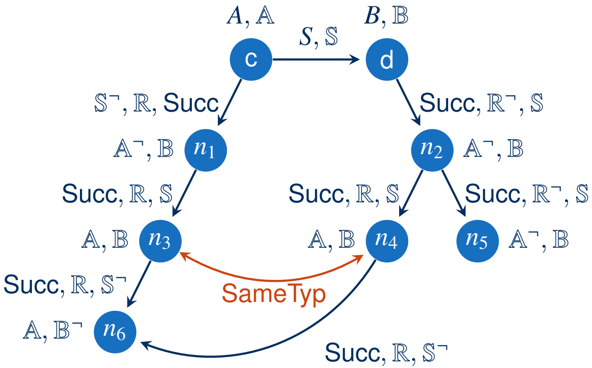
From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



From $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$ by simulating tableau

We construct a tableau-like structure:



Further Results and Outlook

Result Summary: There are polynomial time, fact-preserving rewritings from

- $\mathcal{ALCHIQ}$ to Datalog$^{\vee}$
- $\mathcal{ALCHIQ}$ to Datalog$^{\vee\exists}$
- Horn-$\mathcal{ALCHIQ}$ to Datalog$^{\exists}$ (not shown here)

where all translations with $\exists$ use rules of bounded size on which the (disjunctive) restricted chase will terminate when prioritising rules without $\exists$

Open Challenges

- Can a chase-based system be worst-case optimal for non-Horn logics?
- Rewritings for more DLs ($\mathcal{ALCHOIQ}$ anyone?)
- Further exploitation in implementations