Querying OWL 2 QL and Non-monotonic Rules

Matthias Knorr and José Júlio Alferes

CENTRIA, Faculdade de Ciências e Tecnologia/Universidade Nova de Lisboa

October 26, 2011

Motivating Example

$$\texttt{Professor} \sqsubseteq \exists\texttt{TeachesTo} \qquad \texttt{Student} \sqsubseteq \exists\texttt{HasTutor}$$

$$\exists\texttt{TeachesTo}^- \sqsubseteq \texttt{Student} \qquad \exists\texttt{HasTutor}^- \sqsubseteq \texttt{Professor}$$

$$\texttt{Professor} \sqsubseteq \neg\texttt{Student} \qquad \texttt{HasTutor}^- \sqsubseteq \texttt{TeachesTo}$$

Motivating Example

$$\mathtt{Professor} \sqsubseteq \exists \mathtt{TeachesTo} \qquad \mathtt{Student} \sqsubseteq \exists \mathtt{HasTutor}$$

$$\exists \mathtt{TeachesTo}^{-} \sqsubseteq \mathtt{Student} \qquad \exists \mathtt{HasTutor}^{-} \sqsubseteq \mathtt{Professor}$$

$$\mathtt{Professor} \sqsubseteq \neg \mathtt{Student} \qquad \mathtt{HasTutor}^{-} \sqsubseteq \mathtt{TeachesTo}$$

$$\mathtt{Student}(\mathtt{Paul}) \quad \mathtt{HasTutor}(\mathtt{Jane}, \mathtt{Mary}) \quad \mathtt{TeachesTo}(\mathtt{Mary}, \mathtt{Bill})$$

Motivating Example

$$\mathtt{Professor} \sqsubseteq \exists \mathtt{TeachesTo} \qquad \mathtt{Student} \sqsubseteq \exists \mathtt{HasTutor}$$

$$\exists \mathtt{TeachesTo}^{-} \sqsubseteq \mathtt{Student} \qquad \exists \mathtt{HasTutor}^{-} \sqsubseteq \mathtt{Professor}$$

$$\mathtt{Professor} \sqsubseteq \neg \mathtt{Student} \qquad \mathtt{HasTutor}^{-} \sqsubseteq \mathtt{TeachesTo}$$

$$\mathtt{Student}(\mathtt{Paul}) \quad \mathtt{HasTutor}(\mathtt{Jane}, \mathtt{Mary}) \quad \mathtt{TeachesTo}(\mathtt{Mary}, \mathtt{Bill})$$

$$\mathtt{hasKnownTutor}(\mathrm{x}) \leftarrow \mathtt{Student}(\mathrm{x}), \mathtt{HasTutor}(\mathrm{x}, \mathrm{y})$$

$$\mathtt{hasUnknownTutor}(\mathrm{x}) \leftarrow \mathtt{Student}(\mathrm{x}), \mathbf{not}\, \mathtt{HasTutor}(\mathrm{x}, \mathrm{y})$$

Goals

1. Combine different KR formalisms (unknown individuals vs. default negation – akin to OWL vs. non-monotonic RIF)

Goals

1. Combine different KR formalisms (unknown individuals vs. default negation – akin to OWL vs. non-monotonic RIF)
2. Seamless integration, e.g., in the example, the facts might be rules or even derived from more complex rules that again use information from the ontology

Goals

1. Combine different KR formalisms (unknown individuals vs. default negation – akin to OWL vs. non-monotonic RIF)
2. Seamless integration, e.g., in the example, the facts might be rules or even derived from more complex rules that again use information from the ontology
3. Query efficiently for information in large knowledge bases without having to compute the entire model

Ingredients: 1. OWL 2 QL

- One of the tractable OWL 2 profiles
- Underlying Description Logic $DL\text{-}Lite_{\mathcal{R}}$
- Syntax: GCIs $C \sqsubseteq D$, RIs $R \sqsubseteq E$, and assertions $C(a)$ and $R(a,b)$ where

$$C \longrightarrow A \mid \exists R \quad R \longrightarrow P \mid P^{-} \quad D \longrightarrow C \mid \neg C \quad E \longrightarrow R \mid \neg R$$

- Standard reasoning in PTime (TBox) and LogSpace (ABox)
- Designed for answering queries efficiently

Ingredients: 2. Hybrid MKNF KBs

- DL KB $\mathcal{O}$ + a finite set of rules, $\mathcal{P}$, of the form

$$H \leftarrow A_1, \ldots, A_n, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_m$$

  where H, A_i, and B_j are first-order atoms
- Decidability ensured by DL-safety (application of rules restricted to known individuals)
- Well-founded MKNF semantics applied:
  - Data complexity PTIME with tractable DL
  - Admits top-down querying → **SLG($\mathcal{O}$)**

Ingredients: 3. **SLG($\mathcal{O}$)**

- Extension of SLG Resolution with Tabling (XSB) with an Oracle for $\mathcal{O}$

$$C \sqsubseteq D \qquad\qquad E \sqcap F \sqsubseteq D$$

Query for $D(a)$: $\mathcal{O}$ returns $C(a)$ and $E(a), F(a)$

- Limited to ground queries to the DL Oracle
- Computational complexity of well-founded MKNF maintained if the returned answers of $\mathcal{O}$ are bounded
- General procedure for arbitrary DL

Ingredients: 3. SLG($\mathcal{O}$)

- Extension of SLG Resolution with Tabling (XSB) with an Oracle for $\mathcal{O}$

$$C \sqsubseteq D \qquad\qquad E \sqcap F \sqsubseteq D$$

Query for $D(a)$: $\mathcal{O}$ returns $C(a)$ and $E(a), F(a)$

- Limited to ground queries to the DL Oracle
- Computational complexity of well-founded MKNF maintained if the returned answers of $\mathcal{O}$ are bounded
- General procedure for arbitrary DL

$\rightarrow$ Concrete procedure for $DL\text{-}Lite_{\mathcal{R}}$ is missing

Contribution

1. Concrete oracle for $DL\text{-}Lite_{\mathcal{R}}$
2. Paraconsistent approximation on **SLG(**$\mathcal{O}$**)** for efficiency
3. Oracle may return non-ground answers; improvement on **SLG(**$\mathcal{O}$**)** efficiency
4. Returned answers are bounded by a polynomial
5. Data complexity for (DL-safe) querying in PTIME

Intuitive Idea - Example

$$\mathtt{Professor} \sqsubseteq \exists \mathtt{TeachesTo} \qquad \mathtt{Student} \sqsubseteq \exists \mathtt{HasTutor}$$

$$\exists \mathtt{TeachesTo}^{-} \sqsubseteq \mathtt{Student} \qquad \exists \mathtt{HasTutor}^{-} \sqsubseteq \mathtt{Professor}$$

$$\mathtt{Professor} \sqsubseteq \neg \mathtt{Student} \qquad \mathtt{HasTutor}^{-} \sqsubseteq \mathtt{TeachesTo}$$

$$\mathtt{Student}(\mathtt{Paul}) \leftarrow \qquad \mathtt{HasTutor}(\mathtt{Jane}, \mathtt{Mary}) \leftarrow$$

$$\mathtt{TeachesTo}(\mathtt{Mary}, \mathtt{Bill}) \leftarrow$$

$$\mathtt{hasKnownTutor}(\mathtt{x}) \leftarrow \mathtt{o}(\mathtt{x}), \mathtt{o}(\mathtt{y}), \mathtt{Student}(\mathtt{x}), \mathtt{HasTutor}(\mathtt{x}, \mathtt{y})$$

$$\mathtt{hasUnknownTutor}(\mathtt{x}) \leftarrow \mathtt{o}(\mathtt{x}), \mathtt{o}(\mathtt{y}), \mathtt{Student}(\mathtt{x}), \mathbf{not}\, \mathtt{HasTutor}(\mathtt{x}, \mathtt{y})$$

DL-safety ensured by facts $\mathtt{o}(\mathtt{i}) \leftarrow$ for all $\mathtt{i}$ in the KB

Query $\texttt{hasKnownTutor}(\texttt{x})$

- Atoms $\texttt{o}(\texttt{i})$ ground the DL-atoms
- Query for, e.g., $\texttt{Student}(\texttt{Bill})$ to the $DL\text{-}Lite_{\mathcal{R}}$ Oracle

$\rightarrow$ Oracle computation starts

Query $\texttt{hasKnownTutor}(\texttt{x})$

- Atoms $\texttt{o}(\texttt{i})$ ground the DL-atoms
- Query for, e.g., $\texttt{Student}(\texttt{Bill})$ to the $DL\text{-}Lite_{\mathcal{R}}$ Oracle

$\rightarrow$ Oracle computation starts

1. Satisfiability check for DL part and stop with fail in case of failure

Query $\mathtt{hasKnownTutor}(\mathtt{x})$

- Atoms $\mathtt{o}(\mathtt{i})$ ground the DL-atoms
- Query for, e.g., $\mathtt{Student}(\mathtt{Bill})$ to the $DL\text{-}Lite_{\mathcal{R}}$ Oracle

$\rightarrow$ Oracle computation starts

1. Satisfiability check for DL part and stop with fail in case of failure
2. Otherwise: $\hat{\mathtt{A}}(\mathtt{Bill})$ and $\hat{\mathtt{A}} \sqsubseteq \neg\mathtt{Student}$ added for new predicate $\hat{\mathtt{A}}$

Query $\texttt{hasKnownTutor}(\texttt{x})$

- Atoms $\texttt{o}(\texttt{i})$ ground the DL-atoms
- Query for, e.g., $\texttt{Student}(\texttt{Bill})$ to the $DL\text{-}Lite_{\mathcal{R}}$ Oracle

→ Oracle computation starts

1. Satisfiability check for DL part and stop with fail in case of failure
2. Otherwise: $\hat{\texttt{A}}(\texttt{Bill})$ and $\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student}$ added for new predicate $\hat{\texttt{A}}$
    1. Satisfiability check for augmented KB and stop with success in case of failure (successful instance check for $\texttt{Student}(\texttt{Bill})$)

Query $\texttt{hasKnownTutor}(\texttt{x})$

- Atoms $\texttt{o}(\texttt{i})$ ground the DL-atoms
- Query for, e.g., $\texttt{Student}(\texttt{Bill})$ to the $DL\text{-}Lite_{\mathcal{R}}$ Oracle

$\rightarrow$ Oracle computation starts

1. Satisfiability check for DL part and stop with fail in case of failure
2. Otherwise: $\hat{\texttt{A}}(\texttt{Bill})$ and $\hat{\texttt{A}} \sqsubseteq \neg\texttt{Student}$ added for new predicate $\hat{\texttt{A}}$
    1. Satisfiability check for augmented KB and stop with success in case of failure (successful instance check for $\texttt{Student}(\texttt{Bill})$)
    2. Otherwise: Resolve and return answers

1.Compute negative closure

Closure of negative inclusions - all implicit and explicit inclusions with $\neg$ on the right hand side

$$\begin{aligned}
\mathtt{Professor} &\sqsubseteq \neg\mathtt{Student}\\
\exists\mathtt{HasTutor}^{-} &\sqsubseteq \neg\mathtt{Student}\\
\exists\mathtt{TeachesTo}^{-} &\sqsubseteq \neg\mathtt{Professor}\\
\exists\mathtt{TeachesTo} &\sqsubseteq \neg\mathtt{Student}\\
\exists\mathtt{HasTutor} &\sqsubseteq \neg\mathtt{Professor}\\
\hat{\mathtt{A}} &\sqsubseteq \neg\mathtt{Student}\\
\exists\mathtt{TeachesTo}^{-} &\sqsubseteq \neg\hat{\mathtt{A}}\\
\exists\mathtt{HasTutor} &\sqsubseteq \neg\hat{\mathtt{A}}
\end{aligned}$$

2. Translate negative closure

Disjunction of formulas derived used for (un)satisfiability check:

$$\delta(\texttt{Professor} \sqsubseteq \neg\texttt{Student}) = \exists \texttt{x}.(\texttt{Professor}(\texttt{x}) \wedge \texttt{Student}(\texttt{x}))$$

2. Translate negative closure

Disjunction of formulas derived used for (un)satisfiability check:

$$\delta(\texttt{Professor} \sqsubseteq \neg\texttt{Student}) = \exists \texttt{x}.(\texttt{Professor}(\texttt{x}) \wedge \texttt{Student}(\texttt{x}))$$

$$\exists \texttt{x}.((\exists \texttt{y}.\texttt{HasTutor}(\texttt{y},\texttt{x})) \wedge \texttt{Student}(\texttt{x}))$$
$$\exists \texttt{x}.((\exists \texttt{y}.\texttt{TeachesTo}(\texttt{y},\texttt{x})) \wedge \texttt{Professor}(\texttt{x}))$$
$$\exists \texttt{x}.((\exists \texttt{y}.\texttt{TeachesTo}(\texttt{x},\texttt{y})) \wedge \texttt{Student}(\texttt{x}))$$
$$\exists \texttt{x}.((\exists \texttt{y}.\texttt{HasTutor}(\texttt{x},\texttt{y})) \wedge \texttt{Professor}(\texttt{x}))$$
$$\exists \texttt{x}.(\hat{\texttt{A}}(\texttt{x}) \wedge \texttt{Student}(\texttt{x}))$$
$$\exists \texttt{x}.((\exists \texttt{y}.\texttt{TeachesTo}(\texttt{y},\texttt{x})) \wedge \hat{\texttt{A}}(\texttt{x}))$$
$$\exists \texttt{x}.((\exists \texttt{y}.\texttt{HasTutor}(\texttt{x},\texttt{y})) \wedge \hat{\texttt{A}}(\texttt{x}))$$

3. Derive Meaningful Answers

Resolve $\hat{\mathtt{A}}(\mathtt{Bill})$ with

$$\exists \mathtt{x}.(\hat{\mathtt{A}}(\mathtt{x}) \land \mathtt{Student}(\mathtt{x}))$$
$$\exists \mathtt{x}.((\exists \mathtt{y}.\mathtt{TeachesTo}(\mathtt{y},\mathtt{x})) \land \hat{\mathtt{A}}(\mathtt{x}))$$
$$\exists \mathtt{x}.((\exists \mathtt{y}.\mathtt{HasTutor}(\mathtt{x},\mathtt{y})) \land \hat{\mathtt{A}}(\mathtt{x}))$$

and obtain

$$\mathtt{Student}(\mathtt{Bill}) \quad \mathtt{TeachesTo}(\mathtt{y},\mathtt{Bill}) \quad \mathtt{HasTutor}(\mathtt{Bill},\mathtt{y})$$

3. Derive Meaningful Answers

Resolve $\hat{\mathtt{A}}(\mathtt{Bill})$ with

$$\exists \mathrm{x}.(\hat{\mathtt{A}}(\mathrm{x}) \wedge \mathtt{Student}(\mathrm{x}))$$
$$\exists \mathrm{x}.((\exists \mathrm{y}.\mathtt{TeachesTo}(\mathrm{y},\mathrm{x})) \wedge \hat{\mathtt{A}}(\mathrm{x}))$$
$$\exists \mathrm{x}.((\exists \mathrm{y}.\mathtt{HasTutor}(\mathrm{x},\mathrm{y})) \wedge \hat{\mathtt{A}}(\mathrm{x}))$$

and obtain

$$\mathtt{Student}(\mathtt{Bill}) \quad \mathtt{TeachesTo}(\mathrm{y},\mathtt{Bill}) \quad \mathtt{HasTutor}(\mathtt{Bill},\mathrm{y})$$

Match found with $\mathtt{TeachesTo}(\mathtt{Mary},\mathtt{Bill}) \leftarrow$.
Since we also can derive $\mathtt{HasTutor}(\mathtt{Bill},\mathtt{Mary})$, we obtain $\mathtt{hasKnownTutor}(\mathtt{Bill})$.

Properties

- Generalization of Oracles that now may return non-ground atoms
- Derivations using, e.g., $\exists \mathtt{x}.(\mathtt{Professor}(\mathtt{x}) \land \mathtt{Student}(\mathtt{x}))$ avoided for efficiency
- Well-founded MKNF semantics correspondence for consistent KBs, otherwise paraconsistent approximation
- Tractable data complexity

Conclusions

- Tractable querying for seamless integration of $DL\text{-}Lite_{\mathcal{R}}$ and non-monotonic rules

- Future work:
  - Implementation building on XSB and QuOnto/Mastro
  - Consider true conjunctive queries in our paraconsistent setting