



On the Semantic Relationship between Datalog and Description Logics



Markus Krötzsch*

Sebastian Rudolph^

Peter H. Schmitt^

* Oxford University Computing Laboratory

^ Karlsruhe Institute of Technology (KIT)

Paper published at RR 2010 → [further details & download](#)

September 23, 2010

Description Logic Programs



- Description Logics are fragments of first-order logic
- Datalog can be considered as fragment of first-order logic
- **Description Logic Programming (DLP):**
DLs in the “expressive intersection” of description logic and datalog
- Research questions:
 - What is an “expressive intersection” of two logics?
 - How does the expressive intersection of datalog and DLs look like?

From DL to Datalog

- Building blocks:
individuals I, **concepts A** (unary pred.), **roles R** (binary pred.)
- \mathcal{ALC} Concept expressions:

$$C ::= A \mid C \sqcap C \mid C \sqcup C \mid \neg C \mid \exists R.C \mid \forall R.C$$

- Axioms of \mathcal{ALC} :

$$\begin{array}{ll} C1 \sqsubseteq C2 & \text{(concept inclusion)} \\ R1 \sqsubseteq R2 & \text{(role inclusion)} \end{array} \quad \begin{array}{ll} C(a) & \text{(concept assertion)} \\ R(a,b) & \text{(role assertion)} \end{array}$$

- Examples for DLP axioms:

$$\begin{array}{ll} \square C \sqsubseteq D & C(x) \rightarrow D(x) \\ \square C \sqcap D \sqsubseteq E & C(x) \wedge D(x) \rightarrow E(x) \\ \square C \sqsubseteq \forall R.D & C(x) \wedge R(x,y) \rightarrow D(y) \\ \square \exists R.D \sqsubseteq E & R(x,y) \wedge D(y) \rightarrow E(x) \end{array}$$

Expressing Description Logic in Datalog

- Many translations from DL to datalog have been proposed:
 - DLP [Grosz et al. WWW'03]
 - \mathcal{EL} and extensions [Kazakov PhD'06; K et al. ISWC'08; K Jelia'10]
 - Horn-*SHIQ* [Motik et al. IJCAI'05]
 - Horn-*SROIQ* [Ortiz et al. KR'10]
 - Further approaches for disjunctive datalog
- What distinguishes DLP from these approaches?

Defining Expressibility in Datalog

- What does it mean that a knowledge base KB is “expressible in datalog”?

Some proposals:

- KB is **semantically equivalent** to some datalog program

Defining Expressibility in Datalog

- What does it mean that a knowledge base KB is “expressible in datalog”?

Some proposals:

- KB is **semantically equivalent** to some datalog program
 - Very strong: no Skolemization allowed
- KB is **equisatisfiable** to some datalog program

Defining Expressibility in Datalog

- What does it mean that a knowledge base KB is “expressible in datalog”?

Some proposals:

- KB is **semantically equivalent** to some datalog program
 - Very strong: no Skolemization allowed
- KB is **equisatisfiable** to some datalog program
 - Very weak: true for every knowledge base
- KB **entails the same facts as** some datalog program
 - Proposed notion of “datalog expressible” in [Xiao et al. BuRo'10]

Semantic Emulation



Given theories T and T' with signatures $S \subseteq S'$, then **T' semantically emulates T** if

- every model of T' becomes a model of T when restricted to S ,
- every model of T can be extended to a model of T' by defining interpretations on the additional symbols in S'

→ Like conservative extensions, but no syntactic inclusion

Semantic Emulation



Given theories T and T' with signatures $S \subseteq S'$, then

T' semantically emulates T if

- every model of T' becomes a model of T when restricted to S ,
- every model of T can be extended to a model of T' by defining interpretations on the additional symbols in S'

→ Like conservative extensions, but no syntactic inclusion

- “Expressible in datalog” = can be emulated in datalog
- Consequence: same entailments over signature S
(stronger than “datalog expressible” in [Xiao et al. BuRo'10])

Maximising Datalog-expressible DLs



- **Q:** What is the maximal \mathcal{ALC} fragment that can be emulated in datalog?

Maximising Datalog-expressible DLs



- **Q:** What is the maximal \mathcal{ALC} fragment that can be emulated in datalog?
- **A:** Not very useful!

Reason:

Every inconsistent knowledge base can be emulated in datalog

→ Recognizing a thusly defined “DLP” would require reasoning

Restricting to PTime Transformations



- **Q:** What is the maximal \mathcal{ALC} fragment for which a datalog emulation can be found in polynomial time?

Restricting to PTime Transformations



- **Q:** What is the maximal \mathcal{ALC} fragment for which a datalog emulation can be found in polynomial time?
- **A:** Non-existent!

Reason:

Every such fragment can be extended with finitely many further cases while still allowing polynomial time transformations.

→ Limit of this extension cannot be reached in polynomial time.

Closure under Variants

- **Syntactic variant:** a version of a formula obtained by non-uniform replacement of symbols (within a signature).
- **Closed under variants:** A language containing all possible variants of its member formulae.
 - Every BNF-definable language is closed under variants (but the converse does not hold)
- Strong restriction for preventing many semantic interactions

Maximising DLP

- A description logic D is a **DLP fragment** of a description logic L if:
 - all axioms of D are in L ,
 - there is a datalog transformation function d such that $d(F)$ emulates F in datalog (for all axioms F in D),
 - D is closed under variants

Maximising DLP

- A description logic D is a **DLP fragment** of a description logic L if:
 - all axioms of D are in L ,
 - there is a datalog transformation function d such that $d(F)$ emulates F in datalog (for all axioms F in D),
 - D is closed under variants
- **Theorem:** The largest DLP fragment of \mathcal{ALC} exists and can be described by a BNF.

Body ::= Top | Bot | $\neg A$ | $\forall R.Body$ | **Body** \sqcap **Body** | **Body** \sqcup **Body**

Head ::= **Body** | **A** | $\forall R.Head$ | **Head** \sqcap **Head** | **Head** \sqcup **Body**

Abox ::= **H** | $\exists R.Abox$ | **Abox** \sqcap **Abox** | **Abox** \sqcup **Body**

The *SR0IQ* Case

- Things are not as simple here ...
- Some unexpected emulations:

$$A \sqsubseteq \geq 2R.(\{c\} \sqcup \{d\}) \quad A(x) \rightarrow R(x,c), A(x) \rightarrow R(x,d), A(x) \wedge c \approx d \rightarrow \perp$$

The *SR0IQ* Case

- Things are not as simple here ...
- Some unexpected emulations:

$$A \sqsubseteq \geq 2R.(\{c\} \sqcup \{d\})$$
$$\{c\} \sqsubseteq \geq 2R.A$$

$$A(x) \rightarrow R(x,c), A(x) \rightarrow R(x,d), A(x) \wedge c \approx d \rightarrow \perp$$
$$R(c,s1), R(c,s2), A(s1), A(s2), s1 \approx s2 \rightarrow \perp$$

The *SR0IQ* Case

- Things are not as simple here ...
- Some unexpected emulations:

$$\begin{array}{ll} A \sqsubseteq \geq 2R.(\{c\} \sqcup \{d\}) & A(x) \rightarrow R(x,c), A(x) \rightarrow R(x,d), A(x) \wedge c \approx d \rightarrow \perp \\ \{c\} \sqsubseteq \geq 2R.A & R(c,s1), R(c,s2), A(s1), A(s2), s1 \approx s2 \rightarrow \perp \\ \{c\} \sqsubseteq \geq 2R.(\neg\{a\} \sqcup A \sqcup B) & R(c,s1), R(c,s2), a \approx s1 \rightarrow A(s1), a \approx s2 \rightarrow B(s2), \\ & s1 \approx s2 \rightarrow \perp \end{array}$$

The *SR0IQ* Case

- Things are not as simple here ...
- Some unexpected emulations:

$$\begin{array}{ll}
 A \sqsubseteq \geq 2R.(\{c\} \sqcup \{d\}) & A(x) \rightarrow R(x,c), A(x) \rightarrow R(x,d), A(x) \wedge c \approx d \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 2R.A & R(c,s1), R(c,s2), A(s1), A(s2), s1 \approx s2 \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 2R.(\neg\{a\} \sqcup A \sqcup B) & R(c,s1), R(c,s2), a \approx s1 \rightarrow A(s1), a \approx s2 \rightarrow B(s2), \\
 & s1 \approx s2 \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 4R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leq 1S.(\{c\} \sqcup \{d\}))) & \text{<a program of ca. 30 rules>}
 \end{array}$$

- Similar axioms that cannot be emulated:

$$\begin{array}{l}
 \{c\} \sqsubseteq \geq 2R.(\neg\{a\} \sqcup A \sqcup B \sqcup C) \\
 \{c\} \sqsubseteq \geq 3R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leq 1S.(\{c\} \sqcup \{d\})))
 \end{array}$$

The *SR0IQ* Case

- Things are not as simple here ...
- Some unexpected emulations:

$$\begin{array}{ll}
 A \sqsubseteq \geq 2R.(\{c\} \sqcup \{d\}) & A(x) \rightarrow R(x,c), A(x) \rightarrow R(x,d), A(x) \wedge c \approx d \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 2R.A & R(c,s1), R(c,s2), A(s1), A(s2), s1 \approx s2 \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 2R.(\neg\{a\} \sqcup A \sqcup B) & R(c,s1), R(c,s2), a \approx s1 \rightarrow A(s1), a \approx s2 \rightarrow B(s2), \\
 & s1 \approx s2 \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 4R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leq 1S.(\{c\} \sqcup \{d\}))) & \text{<a program of ca. 30 rules>}
 \end{array}$$

- Similar axioms that cannot be emulated:

$$\begin{array}{l}
 \{c\} \sqsubseteq \geq 2R.(\neg\{a\} \sqcup A \sqcup B \sqcup C) \\
 \{c\} \sqsubseteq \geq 3R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leq 1S.(\{c\} \sqcup \{d\})))
 \end{array}$$

- Maximal DLP fragment for *SR0IQ* exists when restricting to disjunctive normal form axioms.

The *SR0IQ* Case

- Things are not as simple here ...
- Some unexpected emulations:

$$\begin{array}{ll}
 A \sqsubseteq \geq 2R.(\{c\} \sqcup \{d\}) & A(x) \rightarrow R(x,c), A(x) \rightarrow R(x,d), A(x) \wedge c \approx d \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 2R.A & R(c,s1), R(c,s2), A(s1), A(s2), s1 \approx s2 \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 2R.(\neg\{a\} \sqcup A \sqcup B) & R(c,s1), R(c,s2), a \approx s1 \rightarrow A(s1), a \approx s2 \rightarrow B(s2), \\
 & s1 \approx s2 \rightarrow \perp \\
 \{c\} \sqsubseteq \geq 4R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leq 1S.(\{c\} \sqcup \{d\}))) & \text{<a program of ca. 30 rules>}
 \end{array}$$

- Similar axioms that cannot be emulated:

$$\begin{array}{l}
 \{c\} \sqsubseteq \geq 2R.(\neg\{a\} \sqcup A \sqcup B \sqcup C) \\
 \{c\} \sqsubseteq \geq 3R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leq 1S.(\{c\} \sqcup \{d\})))
 \end{array}$$

- Maximal DLP fragment for *SR0IQ* exists when restricting to disjunctive normal form axioms, **but it's not pretty.**

Lessons Learnt



- Negative experiences:
 - DL syntax is very hard to align with rule language syntax
 - Relationships of syntax and semantics are not trivial to study
 - Study did not lead to relevant new DLP-type logics
 - Positive experiences:
 - Semantic emulation is powerful yet workable
 - Closure under variants is a useful abstraction to BNF
 - Principle difference between DLP and EL, Horn-*SHIQ*, Horn-*SROIQ*
- Techniques applicable elsewhere,
but working with DL syntax is hard!

