FAKULTÄT FÜR INFORMATIK

Faculty of Informatics

DReW: a Reasoner for Datalog-rewritable Description Logics and DL-Programs

Guohui Xiao Stijn Heymans Thomas Eiter

Knowledge-based Systems Group, Vienna University of Technology

21 September BuRO 2010

FACULTY OF !NFORMATICS

Motivation

- Ontologies + Rules $\mathcal{KB} = (\Sigma, P)$
- Ontology Σ

$$Father \equiv Man \sqcap \exists hasChild.Human$$

- Rule P

$$rich(X) \leftarrow famous(X), not\ scientist(X)$$

- Proposals:
  - *Description Logic Programs* [Grosof et al., 2003]
  - *DL-safe rules* [Motik et al., 2005]
  - *r-hybrid KBs* [Rosati, 2005]
  - *MKNF KBs* [Motik and Rosati, 2007]
  - *Description Logic Rules* [Krötzsch et al., 2008]
  - *ELP* [Krötzsch et al., 2008]
  - $\mathcal{DL}+log$ [Rosati, 2006]
  - *SWRL* [Horrocks et al., 2004]
  - *dl-programs* [Eiter et al., 2008]

Motivation (2)

- dl-programs are a *loosely-coupled* approach — *treat DL KB as a black box*
- Search for *scalable* approaches:
  - answer set semantics vs. well-founded semantics
    - co-NP-complete vs. PTIME-complete
  - tractable Description Logics (OWL 2 Profiles)
    - OWL 2 EL, OWL 2 QL, OWL 2 RL
    - PTIME-complete for data complexity

Motivation (3)

- However
- despite a well-founded semantics and tractable DLs, dl-programs still
- need a dedicated algorithm using native DL reasoners to perform external queries, causing
- *overhead*.
- eg. dlvhex = dlv + racerpro + ...
- How to avoid this overhead?

Motivation (4)

- identify a class of DLs, Datalog-*rewritable DLs*

Motivation (4)

- identify a class of DLs, Datalog-*rewritable DLs*
- reduce reasoning with dl-programs over Datalog-rewritable DLs to Datalog$^{\neg}$

Motivation (4)

- identify a class of DLs, Datalog-*rewritable DLs*
- reduce reasoning with dl-programs over Datalog-rewritable DLs to Datalog$^{\neg}$
- use of mature LP technology for efficient reasoning with dl-programs

Motivation

Preliminaries: DL-Programs

Reducing DL-Programs to Datalog$^\neg$

Implementation & Evaluation

DL-Programs

DL-Program: $\mathcal{KB} = (\Sigma, P)$.

Atoms in P:

- $student(X)$
  - normal atom
- $DL[; Person](X)$
  - query $Person$ from DL part
- $DL[Student \uplus student; Person](X)$
  - extend DL predicate $Student$ with LP predicate $student$
  - then query $Person$

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{ C \sqsubseteq D \}$ and

$$P = \{ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \}.$$

$KB \models^{wf} q$?

Example of DL-program

$$\text{Let } \mathcal{KB} = (\Sigma, P) \text{ where } \Sigma = \{\ C \sqsubseteq D\ \} \text{ and}$$

$$P = \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

$KB \models^{wf} q$?

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$

Example of DL-program

$$\text{Let } \mathcal{KB} = (\Sigma, P) \text{ where } \Sigma = \{\ C \sqsubseteq D\ \} \text{ and}$$

$$P = \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

$KB \models^{wf} q$?

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$?

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{ C \sqsubseteq D \}$ and

$$P = \{ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$?
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{ \, C \sqsubseteq D \, \}$ and

$$P = \{ \, p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow; \\ q \leftarrow DL[C \uplus s; D](a), not \; DL[C \uplus p; D](b) \, \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$?
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D

Example of DL-program

$$\text{Let } \mathcal{KB} = (\Sigma, P) \text{ where } \Sigma = \{\ C \sqsubseteq D\ \} \text{ and}$$

$$P = \{\ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$

$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$?
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{\ C \sqsubseteq D\ \}$ and

$$P = \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

$KB \models^{wf} q$?

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{\ C \sqsubseteq D\ \}$ and

$$P = \{\ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$
- $I \models DL[C \uplus p; D](b)?$

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{ C \sqsubseteq D \}$ and

$$P = \{ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$
- $I \models DL[C \uplus p; D](b)?$
  - input $C \uplus p$: $\{p(a)\} \Rightarrow \{C(a)\}$

Example of DL-program

$$\text{Let } \mathcal{KB} = (\Sigma, P) \text{ where } \Sigma = \{\ C \sqsubseteq D\ \} \text{ and}$$

$$P = \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$
- $I \models DL[C \uplus p; D](b)?$
  - input $C \uplus p$: $\{p(a)\} \Rightarrow \{C(a)\}$
  - query D

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{ C \sqsubseteq D \}$ and

$$P = \{ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$
- $I \models DL[C \uplus p; D](b)?$
  - input $C \uplus p$: $\{p(a)\} \Rightarrow \{C(a)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a)\} \not\models D(b)$

Example of DL-program

$$\text{Let } \mathcal{KB} = (\Sigma, P) \text{ where } \Sigma = \{\ C \sqsubseteq D\ \} \text{ and}$$

$$P = \{\ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

$KB \models^{wf} q$?

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$
- $I \not\models DL[C \uplus p; D](b)$
  - input $C \uplus p$: $\{p(a)\} \Rightarrow \{C(a)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a)\} \not\models D(b)$

Example of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{ C \sqsubseteq D \}$ and

$$P = \{ p(a) \leftarrow ; \quad s(a) \leftarrow ; \quad s(b) \leftarrow ;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \}.$$

$KB \models^{wf} q?$

- Take a model I of KB, $\{p(a), s(a), s(b)\} \subseteq I$
- $I \models DL[C \uplus s; D](a)$
  - input $C \uplus s$: $\{s(a), s(b)\} \Rightarrow \{C(a), C(b)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a), C(b)\} \models D(a)$
- $I \not\models DL[C \uplus p; D](b)$
  - input $C \uplus p$: $\{p(a)\} \Rightarrow \{C(a)\}$
  - query D
  - $\{C \subseteq D\} \cup \{C(a)\} \not\models D(b)$
- $I \models q$

Datalog-**rewriteable DLs**

Definition

A DL $\mathcal{DL}$ is Datalog-*rewritable* if there exists a transformation $\Phi_{\mathcal{DL}}$ from $\mathcal{DL}$ KBs to Datalog programs such that, for any $\mathcal{DL}$ KB Σ,

(i) $\Sigma \models Q(\mathbf{o})$ iff $\Phi_{\mathcal{DL}}(\Sigma) \models Q(\mathbf{o})$ for any concept or role name Q from Σ, and individuals $\mathbf{o}$ from Σ;

(ii) $\Phi_{\mathcal{DL}}$ is *modular*, i.e., for $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox, $\Phi_{\mathcal{DL}}(\Sigma) = \Phi_{\mathcal{DL}}(\mathcal{T}) \cup \mathcal{A}$;

We refer to a *polynomial* Datalog-rewritable DL $\mathcal{DL}$, if $\Phi_{\mathcal{DL}}(\Sigma)$ for a $\mathcal{DL}$ KB Σ is computable in polynomial time.

Reducing DL-Programs to Datalog$^\neg$

- $\mathcal{KB} = (\Sigma, P)$ a dl-program and let a be a ground atom from $\mathcal{B}_{\mathcal{KB}}$

Reducing DL-Programs to Datalog$^\neg$

- $\mathcal{KB} = (\Sigma, P)$ a dl-program and let a be a ground atom from $\mathcal{B}_{\mathcal{KB}}$
- for Datalog-*rewritable* DLs, reasoning w.r.t. dl-programs over such DLs becomes reducible to Datalog$^\neg$:

Reducing DL-Programs to Datalog$^\neg$

- $\mathcal{KB} = (\Sigma, P)$ a dl-program and let a be a ground atom from $\mathcal{B}_{\mathcal{KB}}$
- for Datalog-*rewritable* DLs, reasoning w.r.t. dl-programs over such DLs becomes reducible to Datalog$^\neg$:
- we can reduce a dl-program $\mathcal{KB} = (\Sigma, P)$ to a Datalog$^\neg$ program $\Psi(\mathcal{KB})$ such that $\mathcal{KB} \models^{wf} a$ iff $\Psi(\mathcal{KB}) \models^{wf} a$

Example of Reduction of DL-program

Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{ C \sqsubseteq D \}$ and

$$P = \{ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \}.$$

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$
- each different input from a dl-atom causes a different DL knowledge base to be queried (i.e., the original DL + the input from the program)

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$
- each different input from a dl-atom causes a different DL knowledge base to be queried (i.e., the original DL + the input from the program)
- Make different disjoint copies of Σ to cope for those different inputs: Σ_{λ_1} and Σ_{λ_2}:

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$
- each different input from a dl-atom causes a different DL knowledge base to be queried (i.e., the original DL + the input from the program)
- Make different disjoint copies of Σ to cope for those different inputs: Σ_{λ_1} and Σ_{λ_2}:
  - $\Sigma_{\lambda_1} = \{ C_{\lambda_1} \sqsubseteq D_{\lambda_1} \}$

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$
- each different input from a dl-atom causes a different DL knowledge base to be queried (i.e., the original DL + the input from the program)
- Make different disjoint copies of Σ to cope for those different inputs: Σ_{λ_1} and Σ_{λ_2}:
  - $\Sigma_{\lambda_1} = \{ C_{\lambda_1} \sqsubseteq D_{\lambda_1} \}$
  - $\Sigma_{\lambda_2} = \{ C_{\lambda_2} \sqsubseteq D_{\lambda_2} \}$

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$
- each different input from a dl-atom causes a different DL knowledge base to be queried (i.e., the original DL + the input from the program)
- Make different disjoint copies of Σ to cope for those different inputs: Σ_{λ_1} and Σ_{λ_2}:
  - $\Sigma_{\lambda_1} = \{ C_{\lambda_1} \sqsubseteq D_{\lambda_1} \}$
  - $\Sigma_{\lambda_2} = \{ C_{\lambda_2} \sqsubseteq D_{\lambda_2} \}$
- Use the Datalog-rewritability: $\Phi_{\mathcal{DL}}(\Sigma_{\lambda_1}) \cup \Phi_{\mathcal{DL}}(\Sigma_{\lambda_2})$.

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$
- each different input from a dl-atom causes a different DL knowledge base to be queried (i.e., the original DL + the input from the program)
- Make different disjoint copies of Σ to cope for those different inputs: Σ_{λ_1} and Σ_{λ_2}:
  - $\Sigma_{\lambda_1} = \{ C_{\lambda_1} \sqsubseteq D_{\lambda_1} \}$
  - $\Sigma_{\lambda_2} = \{ C_{\lambda_2} \sqsubseteq D_{\lambda_2} \}$
- Use the Datalog-rewritability: $\Phi_{\mathcal{DL}}(\Sigma_{\lambda_1}) \cup \Phi_{\mathcal{DL}}(\Sigma_{\lambda_2})$.
  - $\Sigma_{\lambda_1}: \quad D_{\lambda_1}(X) \leftarrow C_{\lambda_1}(X)$

Example (2)

- collect the inputs: $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$
- each different input from a dl-atom causes a different DL knowledge base to be queried (i.e., the original DL + the input from the program)
- Make different disjoint copies of Σ to cope for those different inputs: Σ_{λ_1} and Σ_{λ_2}:
  - $\Sigma_{\lambda_1} = \{ C_{\lambda_1} \sqsubseteq D_{\lambda_1} \}$
  - $\Sigma_{\lambda_2} = \{ C_{\lambda_2} \sqsubseteq D_{\lambda_2} \}$
- Use the Datalog-rewritability: $\Phi_{\mathcal{DL}}(\Sigma_{\lambda_1}) \cup \Phi_{\mathcal{DL}}(\Sigma_{\lambda_2})$.
  - $\Sigma_{\lambda_1}: \quad D_{\lambda_1}(X) \leftarrow C_{\lambda_1}(X)$
  - $\Sigma_{\lambda_2}: \quad D_{\lambda_2}(X) \leftarrow C_{\lambda_2}(X)$

Example (3)

- Make sure the input from the dl-atoms is transferred:

$$
\begin{aligned}
\lambda_1 \mathrel{\acute{=}} C \uplus s : &\qquad C_{\lambda_1}(X) \leftarrow s(X) \\
\lambda_2 \mathrel{\acute{=}} C \uplus p : &\qquad C_{\lambda_2}(X) \leftarrow p(X)
\end{aligned}
$$

Example (3)

- Make sure the input from the dl-atoms is transferred:

$$\lambda_1 \acute{=} C \uplus s : \qquad C_{\lambda_1}(X) \leftarrow s(X)$$
$$\lambda_2 \acute{=} C \uplus p : \qquad C_{\lambda_2}(X) \leftarrow p(X)$$

- Rewrite the original dl-rules to remove the dl-atoms:

$$q \leftarrow DL[\lambda_1; D](a), not\ DL[\lambda_2; D](b) : \qquad q \leftarrow D_{\lambda_1}(a), not\ D_{\lambda_2}(b)$$
$$p(a) \leftarrow \ : \qquad p(a) \leftarrow$$
$$s(a) \leftarrow \ : \qquad s(a) \leftarrow$$
$$s(b) \leftarrow \ : \qquad s(b) \leftarrow$$

Example (4)

$\Sigma = \{ C \sqsubseteq D \}$ and

$$P = \{ p(a) \leftarrow; \quad s(a) \leftarrow; \quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b) \}.$$

becomes

$$\begin{aligned} D_{\lambda_1}(X) &\leftarrow C_{\lambda_1}(X) \\ D_{\lambda_2}(X) &\leftarrow C_{\lambda_2}(X) \end{aligned}$$

$$\begin{aligned} C_{\lambda_1}(X) &\leftarrow s(X) \\ C_{\lambda_2}(X) &\leftarrow p(X) \end{aligned}$$

$$\begin{aligned} q &\leftarrow D_{\lambda_1}(a), not\ D_{\lambda_2}(b) \\ p(a) &\leftarrow \\ s(a) &\leftarrow \\ s(b) &\leftarrow \end{aligned}$$

Results Reduction

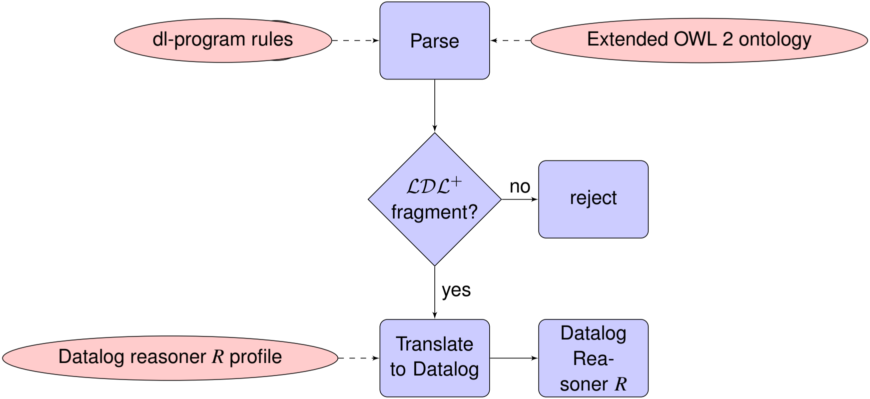
Theorem

Let $\mathcal{KB}$ *be a dl-program over a* Datalog*-rewritable DL and* a *from* $\mathcal{B}_{\mathcal{KB}}$*. Then,* $\mathcal{KB} \models^{wf} a$ *iff* $\Psi(\mathcal{KB}) \models^{wf} a$*.*

Corollary

For any dl-program $\mathcal{KB}$ *over a DL* $\mathcal{DL}$ *and ground atom* a *from* $\mathcal{B}_{\mathcal{KB}}$*, deciding* $\mathcal{KB} \models^{wf} a$ *is*

- *data complete for* PTime*, if* $\mathcal{DL}$ *is* Datalog*-rewritable*
- *combined complete for* ExpTime*, if* $\mathcal{DL}$ *is polynomial* Datalog*-rewritable*


dl-program rules
Parse
Extended OWL 2 ontology
$\mathcal{LDL}^+$ fragment?
no
reject
yes
Datalog reasoner R profile
Translate to Datalog
Datalog Reasoner R

DReW

- Written in Java ($\sim$ 14k lines)
- Datalog Reasoner: DLV
- Ontology Parser: owl-api

DReW

- Written in Java ($\sim$ 14k lines)
- Datalog Reasoner: DLV
- Ontology Parser: owl-api
- DL Reasoner: Conjuctive query for Datalog-rewritable ontologies
- DL-program Reaseaoner: Compute Well-founded models for dl-programs over Datalog-rewritable ontologies

DReW

- Written in Java ($\sim$ 14k lines)
- Datalog Reasoner: DLV
- Ontology Parser: owl-api
- DL Reasoner: Conjuctive query for Datalog-rewritable ontologies
- DL-program Reaseaoner: Compute Well-founded models for dl-programs over Datalog-rewritable ontologies
- http://www.kr.tuwien.ac.at/research/systems/drew/

Evaluation

- Test on LUBM benchmark
- Compare with dlvhex+dlplugin

Query	DReW (ms)	dlvhex+dlplugin (ms)	DL Inputs	factor
0	2,812	4,307	1	1.53
1	2,631	3,043	1	1.16
2	2,601	3,877	1	1.49
3	2,588	4,043	1	1.56
4	2,754	3,508	1	1.27
5	2,995	5,097	1	1.7
6	4,693	19,593	6	4.17
7	3,204	8,382	2	2.62

Summary

- Conclusion
  - Datalog-rewritable DLs
  - Reducing DL-Program to Datalog$^{\neg}$, avoid the overhead
  - DReW Reasoner

References I

- Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., and Tompits, H. (2008).
  Combining answer set programming with description logics for the Semantic Web.
  Artificial Intelligence, 172(12-13):1495–1539.
- Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003).
  Description logic programs: Combining logic programs with description logic.
  In *Proc. WWW 2003*, pages 48–57. ACM.
- Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004).
  Swrl: A semantic web rule language combining owl and ruleml.
  W3c member submission, World Wide Web Consortium.

References II

- Krötzsch, M., Rudolph, S., and Hitzler, P. (2008).
  Description logic rules.
  In *Proc. ECAI*, pages 80–84. IOS Press.
- Krötzsch, M., Rudolph, S., and Hitzler, P. (2008).
  ELP: Tractable rules for OWL 2.
  In *Proc. ISWC 2008*, pages 649–664.
- Motik, B. and Rosati, R. (2007).
  A faithful integration of description logics with logic programming.
  In *Proc. IJCAI*, pages 477–482.
- Motik, B., Sattler, U., and Studer, R. (2005).
  Query answering for OWL-DL with rules.
  Journal of Web Semantics, 3(1):41–60.

References III

Rosati, R. (2005).
On the decidability and complexity of integrating ontologies and rules.
Journal of Web Semantics, 3(1):41–60.

Rosati, R. (2006).
Dl+log: Tight integration of description logics and disjunctive datalog.

In Doherty, P., Mylopoulos, J., and Welty, C. A., editors, *KR*, pages 68–78. AAAI Press.