The DReW System for Nonmonotonic DL-Programs

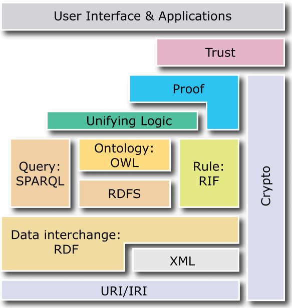
Guohui Xiao[1] Thomas Eiter[1] Stijn Heymans[2]

[1] Institute of Information Systems Vienna University of Technology, Austria

[2] Artificial Intelligence Center, SRI International, United States

CSWS 2012 & CWSC 2012, Shenzhen, 29 Nov 2012

Background: Semantic Web (W3C)

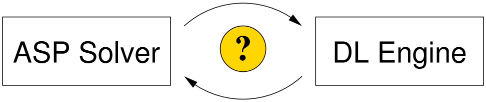


- *RDF (Resource Description Framework)* is the data model
- RDFS (Schema) enriches RDF by simple taxonomies and hierarchies
- More expressive: *OWL (Web Ontology Language)* (2004; 2009)
  - strongly builds on Description Logics
- Rule languages: *Rule Interchange Format (RIF)* (2010)

dl-Programs

- An extension of answer set programs with *queries to DL knowledge bases (KBs)* (through dl-*atoms*)
- dl-atoms allow to query a DL knowledge base differently

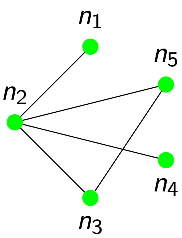
  bidirectional flow of information, with clean technical separation of DL engine and ASP solver ("loose coupling")



- Use DL-programs as "glue" for combining inferences on a DL KB.
- System Prototypes
  - NLP-DL `http://www.kr.tuwien.ac.at/research/systems/semweblp/`
  - dlvhex `http://www.kr.tuwien.ac.at/research/systems/dlvhex/`
  - #F-Logic programs (Ontoprise, extension to F-logic programs)

DL-Programs: Network Example

$KB = (L, P)$



Ontology L

$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$

$$wired = wired^{-};$$

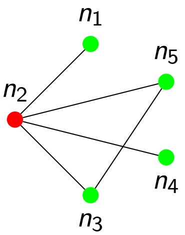
$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$

$$wired(n_1, n_2) \; wired(n_2, n_3) \; wired(n_2, n_4)$$

$$wired(n_2, n_5) \; wired(n_3, n_4) \; wired(n_3, n_5).$$

DL-Programs: Network Example

$KB = (L, P)$

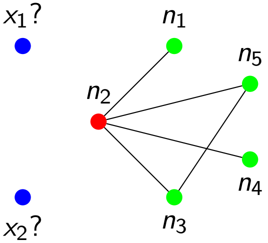


Ontology L

$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$
$$wired = wired^{-};$$
$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$
$$wired(n_1, n_2)\ wired(n_2, n_3)\ wired(n_2, n_4)$$
$$wired(n_2, n_5)\ wired(n_3, n_4)\ wired(n_3, n_5).$$
$$\geq 4.wired \sqsubseteq HighTrafficNode$$

DL-Programs: Network Example

$KB = (L, P)$



Ontology L

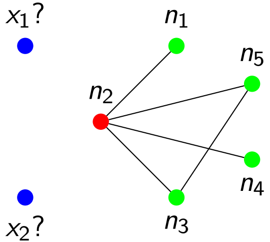
$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$
$$wired = wired^{-};$$
$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$
$$wired(n_1, n_2)\ wired(n_2, n_3)\ wired(n_2, n_4)$$
$$wired(n_2, n_5)\ wired(n_3, n_4)\ wired(n_3, n_5).$$
$$\geq 4.wired \sqsubseteq HighTrafficNode$$

Program P

$newnode(x_1).\quad newnode(x_2)$

DL-Programs: Network Example

$KB = (L, P)$



Ontology L

$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$

$$wired = wired^-;$$

$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$

$$wired(n_1, n_2)\ wired(n_2, n_3)\ wired(n_2, n_4)$$

$$wired(n_2, n_5)\ wired(n_3, n_4)\ wired(n_3, n_5).$$

$$\geq 4.wired \sqsubseteq HighTrafficNode$$

Program P

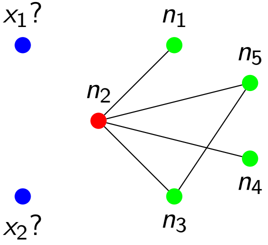
$$newnode(x_1). \quad newnode(x_2)$$

$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$

- DL atom: $\mathrm{DL}[wired \uplus connect; HighTrafficNode](X)$.
- Intuition: extend DL predicate *wired* by *connect*, then query *HighTrafficNode*
  - E.g. Suppose $\{connect(x_1, n_3), connect(x_2, n_3)\} \subseteq I$
  - Then $I \models \mathrm{DL}[wired \uplus connect; HighTrafficNode](n_3)$
  - Thus $I \models overloaded(n_3)$

DL-Programs: Network Example

$KB = (L, P)$



Ontology L

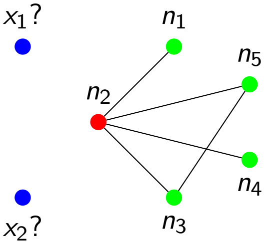
$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$
$$wired = wired^{-};$$
$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$
$$wired(n_1, n_2) \; wired(n_2, n_3) \; wired(n_2, n_4)$$
$$wired(n_2, n_5) \; wired(n_3, n_4) \; wired(n_3, n_5).$$
$$\geq 4.wired \sqsubseteq HighTrafficNode$$

Program P

$$newnode(x_1). \quad newnode(x_2)$$
$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$
$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y),$$
$$not\ overloaded(Y), not\ excl(X, Y).$$

DL-Programs: Network Example

$KB = (L, P)$



Ontology L

$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$

$$wired = wired^-;$$

$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$

$$wired(n_1, n_2)\ wired(n_2, n_3)\ wired(n_2, n_4)$$

$$wired(n_2, n_5)\ wired(n_3, n_4)\ wired(n_3, n_5).$$

$$\geq 4.wired \sqsubseteq HighTrafficNode$$

Program P

$$newnode(x_1). \quad newnode(x_2)$$

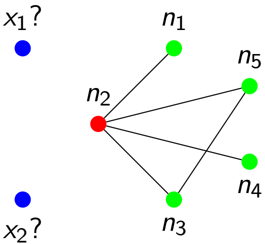
$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$

$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y),$$
$$not\ overloaded(Y), not\ excl(X, Y).$$

$$excl(X, Y) \leftarrow connect(X, Z), \mathrm{DL}[Node](Y), Y \neq Z.$$

DL-Programs: Network Example

$KB = (L, P)$



Ontology L

$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$

$$wired = wired^{-};$$

$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$

$$wired(n_1, n_2)\ wired(n_2, n_3)\ wired(n_2, n_4)$$

$$wired(n_2, n_5)\ wired(n_3, n_4)\ wired(n_3, n_5).$$

$$\geq 4.wired \sqsubseteq HighTrafficNode$$

Program P

$$newnode(x_1). \quad newnode(x_2)$$

$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$

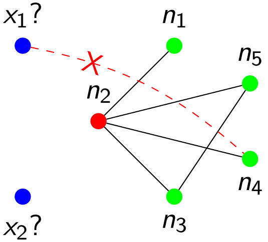
$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y),$$
$$not\ overloaded(Y), not\ excl(X, Y).$$

$$excl(X, Y) \leftarrow connect(X, Z), \mathrm{DL}[Node](Y), Y \neq Z.$$

$$excl(X, Y) \leftarrow connect(Z, Y), newnode(Z), newnode(X), Z \neq X.$$

DL-Programs: Network Example

$KB = (L, P)$



Ontology L

$$\geq 1.wired \sqsubseteq Node \quad \top \sqsubseteq \forall wired.Node$$

$$wired = wired^-;$$

$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$

$$wired(n_1, n_2)\ wired(n_2, n_3)\ wired(n_2, n_4)$$

$$wired(n_2, n_5)\ wired(n_3, n_4)\ wired(n_3, n_5).$$

$$\geq 4.wired \sqsubseteq HighTrafficNode$$

Program P

$$newnode(x_1). \quad newnode(x_2)$$

$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$

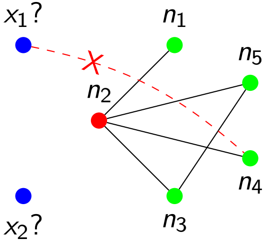
$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y),$$
$$not\ overloaded(Y), not\ excl(X, Y).$$

$$excl(X, Y) \leftarrow connect(X, Z), \mathrm{DL}[Node](Y), Y \neq Z.$$

$$excl(X, Y) \leftarrow connect(Z, Y), newnode(Z), newnode(X), Z \neq X.$$

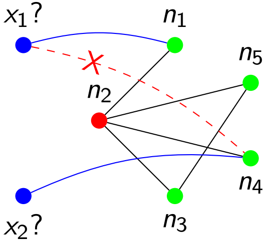
$$excl(x_1, n_4).$$

Semantics of DL-Programs



- Answer set semantics (Stable model semantics)
  - Extension of answer set semantics for normal logical programming
  - Multi models

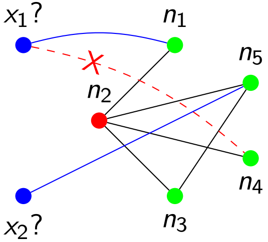
Semantics of DL-Programs



- Answer set semantics (Stable model semantics)
  - Extension of answer set semantics for normal logical programming
  - Multi models

- $M_1 = \{connect(x_1, n_1),\ connect(x_2, n_4), \ldots\}$,

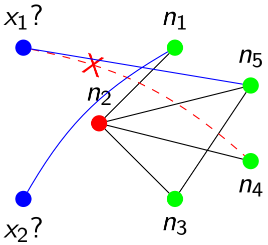
Semantics of DL-Programs



- Answer set semantics (Stable model semantics)
  - Extension of answer set semantics for normal logical programming
  - Multi models

- $M_1 = \{connect(x_1, n_1),\ connect(x_2, n_4), \ldots\}$,
- $M_2 = \{connect(x_1, n_1),\ connect(x_2, n_5),\ \ldots\}$,

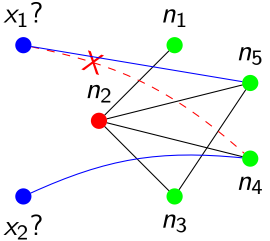
Semantics of DL-Programs



- Answer set semantics (Stable model semantics)
  - Extension of answer set semantics for normal logical programming
  - Multi models

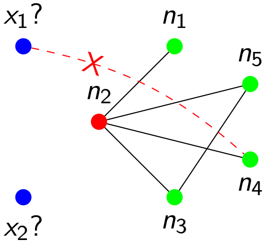
- $M_1 = \{connect(x_1, n_1),\ connect(x_2, n_4), \ldots\}$,
- $M_2 = \{connect(x_1, n_1),\ connect(x_2, n_5),\ \ldots\}$,
- $M_3 = \{connect(x_1, n_5),\ connect(x_2, n_1),\ \ldots\}$,

Semantics of DL-Programs



- Answer set semantics (Stable model semantics)
  - Extension of answer set semantics for normal logical programming
  - Multi models

- $M_1 = \{connect(x_1, n_1),\ connect(x_2, n_4), \ldots\}$,
- $M_2 = \{connect(x_1, n_1),\ connect(x_2, n_5),\ \ldots\}$,
- $M_3 = \{connect(x_1, n_5),\ connect(x_2, n_1),\ \ldots\}$,
- $M_4 = \{connect(x_1, n_5),\ connect(x_2, n_4), \ldots\}$.

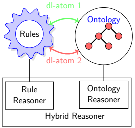


- Well-founded semantics
  - Extension of well-founded semantics for normal logical programming
  - Single model

- $M_0 = \{overloaded(n_2), \ldots\}$

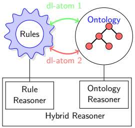
Loose Coupling - Features

- **Advantage:**
  - Clean semantics, can use legacy systems
  - Fairly easy to incorporate further knowledge formats (e.g. RDF)
  - Privacy, information hiding

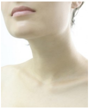


Loose Coupling - Features

- **Advantage:**
  - Clean semantics, can use legacy systems
  - Fairly easy to incorporate further knowledge formats (e.g. RDF)
  - Privacy, information hiding

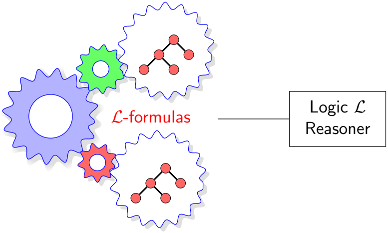


- **Drawback**: *impedance mismatch, performance*
  - Evaluation of DL-program needs multiple calls of a DL-reasoner
  - Calls are expensive
    - optimizations (caching, pruning ...)
  - In some case, exponentially many calls might be unavoidable
  - Even polynomially many calls might be too costly



Uniform Evaluation

Convert the evaluation problem into one for a single reasoning engine



- This means to transform a dl-program into an (equivalent) knowledge base in *one formalism* $\mathcal{L}$ for evaluation (*uniform evaluation*)
- In this talk, $\mathcal{L} = \text{Datalog}^{\neg}$

Reasoning with DL-Programs by Datalog$^\neg$ rewriting

1. Rewriting Ontology to Datalog
2. Duplicating rewritten ontologies according to the dl-inputs
3. Rewriting DL-rules to Datalog$^\neg$ rules
4. Rewriting DL-atoms to Datalog rules
5. Calling Datalog reasoner

DReW Reasoner

- DReW is a reasoner for DL-Programs over Datalog-rewritable Description Logics
- homepage: `http://www.kr.tuwien.ac.at/research/systems/drew/`
- open sourced: `https://github.com/ghxiao/drew`

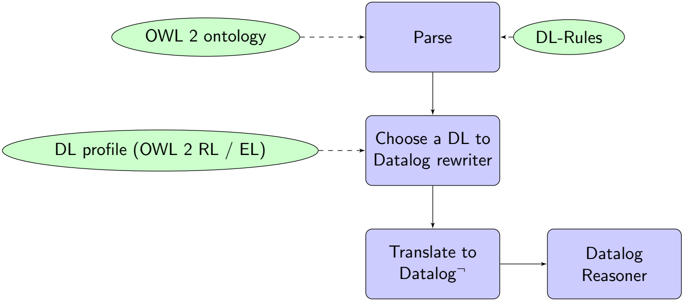


Figure : Control Flow of DReW with DL-programs

Features in DReW v0.3

- Ontology component
  - OWL 2 RL
  - OWL 2 EL
- Semantics
  - ASP semantics
  - Well-founded semantics
- Rule formalism
  - DL-Programs
  - Conjunctive Query under DL-safeness
  - Terminological default reasoning

Example Usage

Example with network DL-Programs under ASP semantics

```
$ ./drew -rl -ontology sample_data/network.owl \
  -dlp sample_data/network.dlp \
  -filter connect -dlv $HOME/bin/dlv

{ connect(x1, n1) connect(x2, n5) }

{ connect(x1, n5) connect(x2, n1) }

{ connect(x1, n5) connect(x2, n4) }

{ connect(x1, n1) connect(x2, n4) }
```

Example Usage

Example with network dl-Programs under well-founded semantics

```
$ ./drew -rl -ontology sample_data/network.owl \
-dlp sample_data/network.dlp \
-filter overloaded -wf -dlv ./dlv-wf

{ overloaded(n2) }
```

Summary and Outlook

- Summary
  - DL-programs is a strong formalism for combining Ontology and Rules
  - Traditional engine for dl-Programs suffers from the overhead of calling external DL Reasoner
  - By exploiting Datalog-rewritablity, reasoning over dl-programs can be reduced to Datalog$^{\neg}$
  - Try DReW Reasoner!
- Outlook
  - More evaluation
  - More expressive DL component, e.g. Horn-SHIQ
  - More reasoning paradigm support, e.g. Closed World Assumption