# Inline Evaluation of Hybrid Knowledge Bases

Guohui Xiao

Vienna PhD School of Informatics
Institute of Information Systems
Vienna University of Technology

January 23, 2014

# Hybrid Knowledge Bases

- Hybrid Knowledge Bases: combining KBs formulated in different logics
- In the context of Semantic Web: OWL Ontologies + Rules
- In this thesis: $\mathrm{dl}$-Programs – loose coupling ontologies and rules

# Inline Evaluation

```
// max of integers x and y
inline int max(int x, int y){
  return x > y ? x : y; }

// max of an integer array of size n
int max_array(int array[], int n) {
  int result = INT_MIN;
  for (int i = 0; i < n; i++) {
    result = max(result, array[i]);


  }
  return result;
}
```

# Inline Evaluation

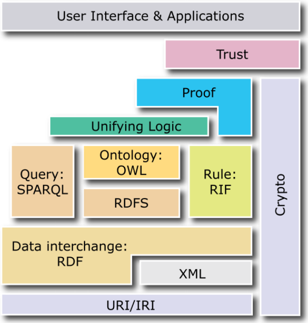
```
// max of integers x and y
inline int max(int x, int y){
  return x > y ? x : y; }

// max of an integer array of size n
int max_array(int array[], int n) {
  int result = INT_MIN;
  for (int i = 0; i < n; i++) {
    //result = max(result, array[i]);
    result = result > array[i] ? result :
       array[i];
  }
  return result;
}
```

# Outline

# Rules and the Semantic Web

http://www.w3.org/2007/03/layerCake.png




- **Issue:** Combining rules and ontologies (logic framework)
- Rules and ontology formalisms like RDF/s, OWL resp. Description Logics have related yet different underlying settings
- Combination is nontrivial (at the heart, the difference is between LP and classical logic)

# OWL Ontologies and Description Logics

- Knowledge about concepts, individuals, their properties and relationships
- W3C Recommendation (2004): *Web Ontology Language (OWL)*
- OWL2 (2009): tractable profiles OWL2 EL, OWL2 QL, OWL2 RL
- OWL syntax is based on RDF
- OWL semantics is based on Description logics

# Description Logics (DLs)

Description Logics are fragments of First-order Logics

- The vocabulary of basic DLs comprises:
  - Concepts (e.g., *Wine*, *WhiteWine*)
  - Roles (e.g., *hasMaker*, *madeFromGrape*)
  - Individuals (e.g., *SelaksIceWine*, *TaylorPort*)
- Statements relate individuals and their properties using
  - logical connectives ($\sqcap$, $\sqcup$, $\neg$, $\sqsubseteq$, etc), and
  - quantifiers ($\exists$, $\forall$, $\leq k$, $\geq k$, etc)
- A DL knowledge base $L = (\mathcal{T}, \mathcal{A})$ (ontology) usually comprises
  - a TBox $\mathcal{T}$ (terminology, conceptualization), and
  - an ABox $\mathcal{A}$ (assertions, extensional knowledge)
- DLs are tailored for decidable reasoning (key task: satisfiability)

# Example: Wine Ontology

- Available at `http://www.w3.org/TR/owl-guide/wine.rdf`
- Some axioms from the TBox

$$\begin{aligned} Wine &\sqsubseteq PotableLiquid \sqcap =1 hasMaker \sqcap \forall hasMaker.Winery; \\ \exists hasColor^{-}.Wine &\sqsubseteq \{"White", "Rose", "Red"\}; \\ WhiteWine &\equiv Wine \sqcap \forall hasColor.\{"White"\}. \end{aligned}$$

  - A wine is a potable liquid, having exactly one maker, who is a member of the class "*Winery*".
  - Wines have colors "*White*", "*Rose*", or "*Red*".
  - A *WhiteWine* is a wine with exclusive color "*White*".

- The ABox contains, e.g.,

$$WhiteWine("StGenevieveTexasWhite"),\ hasMaker("TaylorPort", "Taylor")$$

# Formal OWL / DL Semantics

- The semantics of core DLs is given by a mapping to first-order logic

  *In essence, DLs are "FO logic in disguise"*

# Formal OWL / DL Semantics

- The semantics of core DLs is given by a mapping to first-order logic

  *In essence, DLs are "FO logic in disguise"*

| OWL property axioms as RDF Triples | DL syntax | FOL short representation |
|---|---|---|
| $\langle P$ rdfs:domain $C\rangle$ | $\top \sqsubseteq \forall P^{-}.C$ | $\forall x, y.P(x, y) \supset C(x)$ |
| $\langle P$ rdfs:range $C\rangle$ | $\top \sqsubseteq \forall P.C$ | $\forall x, y.P(x, y) \supset C(y)$ |
| $\langle P$ owl:inverseOf $P_0\rangle$ | $P \equiv P_0^{-}$ | $\forall x, y.P(x, y) \equiv P_0(y, x)$ |
| $\langle P$ rdf:type owl:SymmetricProperty $\rangle$ | $P \equiv P^{-}$ | $\forall x, y.P(x, y) \equiv P(y, x)$ |
| $\langle P$ rdf:type owl:FunctionalProperty $\rangle$ | $\top \sqsubseteq \leqslant 1P$ | $\forall x, y_1, y_2.P(x, y_1) \wedge P(x, y_2) \supset y_1 = y_2$ |
| $\langle P$ rdf:type owl:TransitiveProperty $\rangle$ | $P^{+} \sqsubseteq P$ | $\forall x, y, z.P(x, y) \wedge P(y, z) \supset P(x, z)$ |

| OWL complex class descriptions | DL syntax | FOL short representation |
|---|---|---|
| owl:Thing | $\top$ | $x = x$ |
| owl:Nothing | $\bot$ | $\neg x = x$ |
| owl:intersectionOf $(C_1 \dots C_n)$ | $C_1 \sqcap \dots \sqcap C_n$ | $\bigwedge C_i(x)$ |
| owl:unionOf $(C_1 \dots C_n)$ | $C_1 \sqcup \dots \sqcup C_n$ | $\bigvee C_i(x)$ |
| owl:complementOf $(C)$ | $\neg C$ | $\neg C(x)$ |
| owl:oneOf $(o_1 \dots o_n)$ | $\{o_1 \dots o_n\}$ | $\bigvee x = o_i$ |
| owl:restriction ($P$ owl:someValuesFrom $(C)$) | $\exists P.C$ | $\exists y.P(x, y) \wedge C(y)$ |
| owl:restriction ($P$ owl:allValuesFrom $(C)$) | $\forall P.C$ | $\forall y.P(x, y) \supset C(y)$ |
| owl:restriction ($P$ owl:value $(o)$) | $\exists P.\{o\}$ | $P(x, o)$ |
| owl:restriction ($P$ owl:minCardinality $(n)$) | $\geqslant n\,P$ | $\exists_{i=1}^{n} y_i . \bigwedge_{j=1}^{n} P(x, y_j) \wedge \bigwedge_{i \neq j} y_i \neq y_j$ |

# Systems

- Java API: OWL-API
- Ontology Editor: Protege
- Reasoners
  - OWL(2): Pellet, RacerPro, KAON2, HermiT
  - OWL2 RL: Jena, Base VISor
  - OWL2 EL: CEL, ELK
  - OWL2 QL: QuOnto, OWLGres, Requiem, Ontop

# Normal Logic Programs

## Normal Logic Program

A *normal logic program* is a set of rules of the form

$$a \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n \qquad (n, m \geq 0) \tag{1}$$

where $a$ and all $b_i$, $c_j$ are atoms in a first-order language $L$.

$not$ is called "negation as failure", "default negation", or "weak negation"

## Example

$$\begin{aligned} man(dilbert). \\ single(X) \leftarrow man(X), not\ husband(X). \\ husband(X) \leftarrow man(X), not\ single(X). \end{aligned}$$

# Semantics of Logic Programs

- "War of Semantics" in Logic Programming (1980/90ies):
  Meaning of programs like the Dilbert example above

# Semantics of Logic Programs

- “War of Semantics” in Logic Programming (1980/90ies):
  Meaning of programs like the Dilbert example above
- Great Schism: Single model vs. multiple model semantics

# Semantics of Logic Programs

- "War of Semantics" in Logic Programming (1980/90ies):

  Meaning of programs like the Dilbert example above
- Great Schism: Single model vs. multiple model semantics
- To date:

# Semantics of Logic Programs

- "War of Semantics" in Logic Programming (1980/90ies):

  Meaning of programs like the Dilbert example above

- Great Schism: Single model vs. multiple model semantics
- To date:
  - *Answer Set (alias Stable Model) Semantics* by Gelfond and Lifschitz [1988,1991].

    Alternative models: $M_1 = \{man(dilbert), single(dilbert)\}$,
    $M_2 = \{man(dilbert), husband(dilbert)\}$.

  - *Well-Founded Semantics* [van Gelder *et al.*, 1991]

    Partial model: $man(dilbert)$ is true,
    $single(dilbert)$, $husband(dilbert)$ are *unknown*

# Semantics of Logic Programs

- "War of Semantics" in Logic Programming (1980/90ies):

  Meaning of programs like the Dilbert example above

- Great Schism: Single model vs. multiple model semantics
- To date:
  - *Answer Set (alias Stable Model) Semantics* by Gelfond and Lifschitz [1988,1991].

    Alternative models: $M_1 = \{man(dilbert), single(dilbert)\}$,
    $M_2 = \{man(dilbert), husband(dilbert)\}$.

  - *Well-Founded Semantics* [van Gelder *et al.*, 1991]

    Partial model: $man(dilbert)$ is true,
    $single(dilbert)$, $husband(dilbert)$ are *unknown*

- Agreement for so-called "stratified programs" (acyclic negation)

  Different selection principles for non-stratified programs

# Practical Considerations

- Standards
  - W3C recommedation RIF: RID-Core, RIF-BLD, RIF-PRD
  - ASP-Core-2 (2013)
    - ASP Standardization Working Group
    - Partially for ASP Competition
- Systems
  - LParse
  - Smodels
  - ASSAT
  - Patassco (Gringo, Clasp, Clingo)
  - DLV

# Main Differences OWL vs. Rules?

- `not` in rule paradigms is different from negation (e.g., ComplementOf) in OWL:
  - $\neg$: Classical negation! Open world assumption! Monotonicity!
  - `not`: Different purpose! Closed world assumption! Non-monotonicity!

| | |
|---|---|
| $Publication \sqsubseteq Paper$ | $Paper(X) \leftarrow Publication(X)$ |
| $\neg Publication \sqsubseteq Unpublished$ | $Unpublished(X) \leftarrow not\ Publication(X)$ |
| $paper_1 \in Paper.$ | $Paper(paper_1) \leftarrow$ |
| in DL: $\not\models paper_1 \in Unpublished$ | Does infer in LP: $Unpublished(paper_1)$. |

# Main Differences OWL vs. Rules?

- `not` in rule paradigms is different from negation (e.g., ComplementOf) in OWL:
  - $\neg$: Classical negation! Open world assumption! Monotonicity!
  - `not`: Different purpose! Closed world assumption! Non-monotonicity!

| | |
|---|---|
| $Publication \sqsubseteq Paper$<br>$\neg Publication \sqsubseteq Unpublished$<br>$paper_1 \in Paper.$ | $Paper(X) \leftarrow Publication(X)$<br>$Unpublished(X) \leftarrow not\ Publication(X)$<br>$Paper(paper_1) \leftarrow$ |
| in DL: $\not\models paper_1 \in Unpublished$ | Does infer in LP: $Unpublished(paper_1)$. |

- Also strong negation in LP ("$-$", sometimes "$\neg$") is not completely the same as classical negation in DLs, e.g.

| | |
|---|---|
| $Publication \sqsubseteq Paper$<br>$a \in \neg Paper.$ | $Paper(X) \leftarrow Publication(X)$<br>$\neg Paper(a)$ |
| in DL: $\models a \in \neg Publication$ | Does not automatically infer in LP: $\neg Publication(a)$. |

# Main Differences OWL vs. Rules?

- LPs are strong in query answering, but subsumption checking as in DLs is infeasible (undecidable even for positive function-free programs).

# Main Differences OWL vs. Rules?

- LPs are strong in query answering, but subsumption checking as in DLs is infeasible (undecidable even for positive function-free programs).
- OWL DL allows complex statements in the “head” (rhs of $\sqsubseteq$), while use of variables in LP rule bodies is more flexible

# Main Differences OWL vs. Rules?

- LPs are strong in query answering, but subsumption checking as in DLs is infeasible (undecidable even for positive function-free programs).
- OWL DL allows complex statements in the “head” (rhs of $\sqsubseteq$), while use of variables in LP rule bodies is more flexible
- DLs are stronger in type *inference*, while LPs are stronger in *type checking*:

| | |
|---|---|
| $Person \sqsubseteq \exists hasName.xs{:}string$<br>$john \in Person$ | $\leftarrow Person(X), not\ hasName(X, Y)$<br>$Person(john)$ |
| is consistent in DL and infers<br>$john \in \exists hasName$ | is inconsistent, since there is no<br>known name for *john* |

# Marrying Rules and Ontologies

- Hybrid knowledge base: $KB = (O, P)$
  - $O$ is an ontology

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

  - $P$ is the rules part (program)

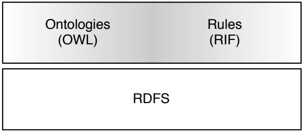
$$rich(X) \leftarrow famous(X), not\ scientist(X)$$

  - *Description Logic Programs* [Grosof *et al.*, 2003]
  - *DL-safe rules* [Motik *et al.*, 2005]
  - *r-hybrid KBs* [Rosati, 2005]
  - *hybrid MKNF KBs* [Motik and Rosati, 2010]
  - *Description Logic Rules* [Krötzsch *et al.*, 2008a]
  - *ELP* [Krötzsch *et al.*, 2008b]
  - $\mathcal{DL}{+}log$ [Rosati, 2006]
  - *SWRL* [Horrocks *et al.*, 2004]
  - dl-programs [E_ *et al.*, 2008]
  - ...

# Semantics

- Different ways to give semantics to $\mathcal{K} = (\mathcal{O}, P)$
  overviews e.g. [Motik and Rosati, 2010], [de Bruijn *et al.*, 2009]
  - **Tight semantic integration**
  - **Full integration**
  - **Strict semantic separation (loose coupling)**
- **Nonmonotonic semantics:**
  - answer sets
  - well-founded semantics
  - ...

# Tight Semantic Integration

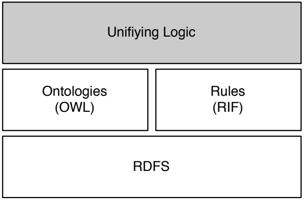



- Integrate FOL statements and the logic program to a large extent, but keep predicates of $\Sigma_{\mathcal{O}}$ and $\Sigma_P$ separate.
- Build an *integrated model* $M$ as the "union" of a model $M_{\mathcal{O}}$ of the FO theory $\mathcal{O}$ and a model $M_P$ of $P$ with the same domain.
- Ensure *"safe interaction"* between $M_{\mathcal{O}}$ and $M_P$.

## Examples

**CARIN** [Levy and Rousset, 1998], **DLP (≈ OWL 2 RL)** [Grosof *et al.*, 2003], dl**-safe rules** [Motik *et al.*, 2005], **R-hybrid KBs** [Rosati, 2005]
$\mathcal{DL}$+LOG [Rosati, 2006]

# Full Integration



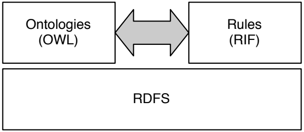

- No fundamental separation between $\Sigma_{\mathcal{O}}$, $\Sigma_P$ (but special axioms)

## Examples

- **Hybrid MKNF knowledge bases** [Motik and Rosati, 2010; Knorr *et al.*, 2008]
- **FO-Autoepistemic Logic** [de Bruijn *et al.*, 2007a]
- **Quantified Equilibrium Logic** [de Bruijn *et al.*, 2007b] (use special axioms)

# Loose Coupling

- Strict semantic separation between rules / ontology




  - View rule base $P$ and FO theory $\mathcal{O}$ as separate, independent components. $\Sigma_{\mathcal{O}}$ and $\Sigma_P$ do (a priori) not share meaning.
  - They are connected through a minimal "safe interface" for exchanging knowledge (formulas, usually ground atoms).

- Well-suited for implementation on top of LP & DL reasoners.

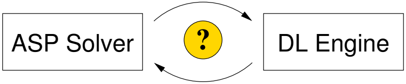
## Examples

**nonmonotonic $\mathrm{dl}$-programs** [E_ *et al.*, 2008], [E_ *et al.*, 2011]
**defeasible logic+DLs** [Wang *et al.*, 2004]

# dl-Programs

- An extension of answer set programs with *queries to DL knowledge bases (DL KBs)*
- Queries can *temporarily update* the DL KB

  *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.

# dl-Programs

dl-programs are hybrid KBs with dl-atoms in rules

### dl-Program

A dl-*program* is a pair $\Pi = (\mathcal{O}, P)$ where

- $\mathcal{O}$ is a DL knowledge base ("*ontology*")
- $P$ consists of dl-*rules*

$$a \leftarrow b_1, \dots, b_k, not\, b_{k+1}, \dots, not\, b_m, \qquad (1)$$

where

- $not$ is default negation ("unless derivable"),
- $a_1, \dots, a_n$ are atoms,
- $b_1, \dots, b_m$, $m \geq 0$, are atoms or dl-atoms (no function symbols).

# dl-Atoms

**Basic Idea:**

- Query the DL KB $\mathcal{O}$ using the *query interface* of the DL engine

  Query $Q$ may be concept/role instance $C(X)$ / $R(X, Y)$; subsumption test $C \sqsubseteq D$; etc (recent extension: conjunctive queries)

- **Important**: Possible to modify the extensional part (ABox) of $\mathcal{O}$, by adding positive ($\uplus$) or negative ($\cup\!\!\!-$, $\cap\!\!\!-$) assertions, before querying

- $Q$ evaluates to true iff the modified $\mathcal{O}$ proves $Q$.

# dl-Atoms: Syntax

## dl-atom

A dl-*atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}), \qquad m \geq 0,$$

# dl-Atoms: Syntax

## dl-atom

A dl-*atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}), \qquad m \geq 0,$$

where

- each $S_i$ is either a concept or a role

# dl-Atoms: Syntax

## dl-atom

A dl-*atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}), \qquad m \geq 0,$$

where

- each $S_i$ is either a concept or a role
- Intuitively $\uplus$ increases $S_i$ by $p_i$

# dl-Atoms: Syntax

## dl-atom

A dl-*atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}), \qquad m \geq 0,$$

where

- each $S_i$ is either a concept or a role
- Intuitively $\uplus$ increases $S_i$ by $p_i$
- $p_i$ is a unary resp. binary predicate (*input predicate*),

# dl-Atoms: Syntax

## dl-atom

A dl-*atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}), \qquad m \geq 0,$$

where

- each $S_i$ is either a concept or a role
- Intuitively $\uplus$ increases $S_i$ by $p_i$
- $p_i$ is a unary resp. binary predicate (*input predicate*),
- $Q(\mathbf{t})$ is a dl-*query* ($\mathbf{t}$ contains variables and/or constants), which is one of
  - (a) $C(t)$, for a concept $C$ and term $t$, or
  - (b) $R(t_1, t_2)$, for a role $R$ and terms $t_1$, $t_2$.

# dl-Atoms: Syntax

## dl-atom

A dl-*atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}), \qquad m \geq 0,$$

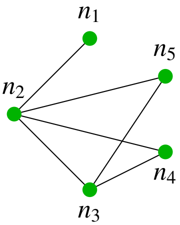
where

- each $S_i$ is either a concept or a role
- Intuitively $\uplus$ increases $S_i$ by $p_i$
- $p_i$ is a unary resp. binary predicate (*input predicate*),
- $Q(\mathbf{t})$ is a dl-*query* ($\mathbf{t}$ contains variables and/or constants), which is one of
  - (a) $C(t)$, for a concept $C$ and term $t$, or
  - (b) $R(t_1, t_2)$, for a role $R$ and terms $t_1$, $t_2$.

Shorthand: $\lambda = S_1 op_1 p_1, \ldots, S_m op_m p_m$

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

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

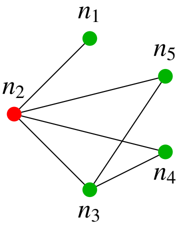
$$n_1 \neq n_2 \neq n_3 \neq n_4 \neq n_5$$

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

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

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

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

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

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

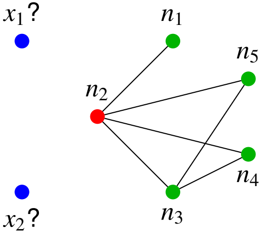
$$wired(n_2, n_5)\ wired(n_3, n_4)\ wired(n_3, n_5).$$

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

---

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

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

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

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

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

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

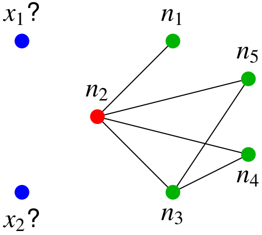
---

Rules $P$

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

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

---

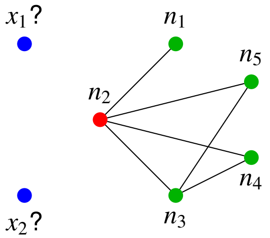
Rules $P$

$$newnode(x_1). \quad newnode(x_2).$$
$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$

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

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

$$\geq 1.\mathit{wired} \sqsubseteq \mathit{Node} \quad \top \sqsubseteq \forall \mathit{wired}.\mathit{Node}$$

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

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

$$\mathit{wired}(n_1, n_2)\ \mathit{wired}(n_2, n_3)\ \mathit{wired}(n_2, n_4)$$

$$\mathit{wired}(n_2, n_5)\ \mathit{wired}(n_3, n_4)\ \mathit{wired}(n_3, n_5).$$

$$\geq 4.\mathit{wired} \sqsubseteq \mathit{HighTrafficNode}$$

---

Rules $P$

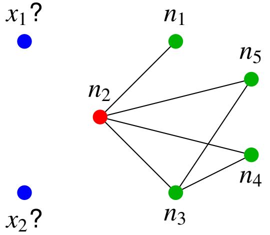
$$\mathit{newnode}(x_1). \quad \mathit{newnode}(x_2).$$

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

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

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

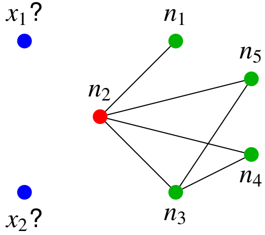
---

Rules $P$

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

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

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

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

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

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

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

---

Rules $P$

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

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

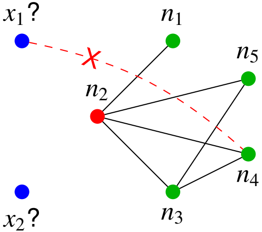
$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y), not\ overloaded(Y), not\ excl(X, Y).$$

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

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

# Example: Network Connections

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

---

Rules $P$

$$newnode(x_1). \quad newnode(x_2).$$
$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$
$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y), not\ overloaded(Y), not\ excl(X, Y).$$
$$excl(X, Y) \leftarrow connect(X, Z), \mathrm{DL}[Node](Y), Y \neq Z.$$
$$excl(X, Y) \leftarrow connect(Z, Y), newnode(Z), newnode(X), Z \neq X.$$
$$excl(x_1, n_4).$$

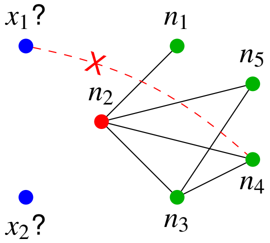
# Semantics

## Satisfaction ($I \models_{\mathcal{O}} a$)

- $I$ satisfies a classical ground atom $a$ iff $a \in I$;
- $I$ satisfies a ground dl-atom $a = DL[\lambda; Q](\mathbf{c})$ iff $\mathcal{O} \cup \bigcup_{i=1}^{m} A_i(I) \models Q(\mathbf{c})$, where $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$,

- The semantics of Logic Programmings can be extended to dl-Programs
- Answer set semantics
- Well-founded semantics

# Network Example: Answer Sets




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

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

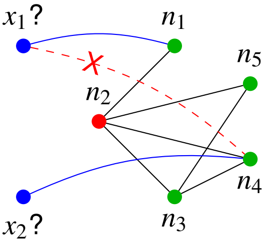
$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y), notoverloaded(Y), notexcl(X, Y).$$

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

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

$$excl(x_1, n_4).$$

# Network Example: Answer Sets




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

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

$$connect(X,Y) \leftarrow newnode(X), \mathsf{DL}[Node](Y), notoverloaded(Y), notexcl(X,Y).$$

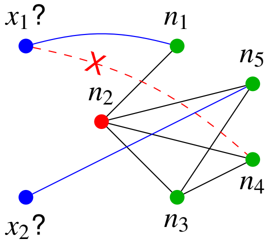
$$excl(X,Y) \leftarrow connect(X,Z), \mathsf{DL}[Node](Y), Y \neq Z.$$

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

$$excl(x_1, n_4).$$

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

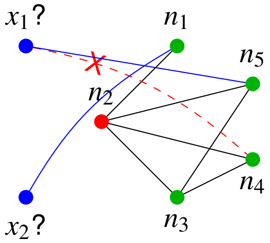
# Network Example: Answer Sets




$$
\begin{aligned}
& newnode(x_1). \quad newnode(x_2). \\
overloaded(X) &\leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X). \\
connect(X,Y) &\leftarrow newnode(X), \mathrm{DL}[Node](Y), \\
& \quad notoverloaded(Y), notexcl(X,Y). \\
excl(X,Y) &\leftarrow connect(X,Z), \mathrm{DL}[Node](Y), Y \neq Z. \\
excl(X,Y) &\leftarrow connect(Z,Y), newnode(Z), newnode(X), Z \neq X. \\
& excl(x_1, n_4).
\end{aligned}
$$

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

# Network Example: Answer Sets




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

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

$$connect(X,Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y), notoverloaded(Y), notexcl(X,Y).$$

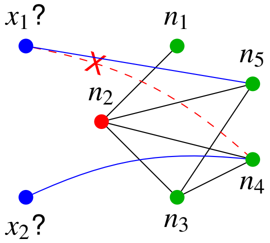
$$excl(X,Y) \leftarrow connect(X,Z), \mathrm{DL}[Node](Y), Y \neq Z.$$

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

$$excl(x_1, n_4).$$

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

# Network Example: Answer Sets




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

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

$$connect(X,Y) \leftarrow newnode(X), \mathsf{DL}[Node](Y), notoverloaded(Y), notexcl(X,Y).$$

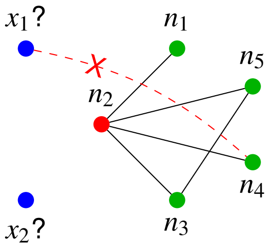
$$excl(X,Y) \leftarrow connect(X,Z), \mathsf{DL}[Node](Y), Y \neq Z.$$

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

$$excl(x_1, n_4).$$

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

# Network Example: Well-founded Semantics




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

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

$$connect(X,Y) \leftarrow newnode(X), \mathsf{DL}[Node](Y), notoverloaded(Y), notexcl(X,Y).$$

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

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

$$excl(x_1, n_4).$$

- $WFS(\Pi) = \{overloaded(n_2), \ldots\}$
- $\Pi \models_{wf} \neg connect(x_1, n_4)$, ...
- $WFM(\Pi) = \{overloaded(n_2), \neg connect(x_1, n_4), \ldots\}$

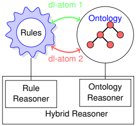
# System for dl-Programs

- NLP-DL
  - `https://www.mat.unical.it/ianni/swlp/`
  - First Experimental prototype
  - DL Engine: RacerPro
  - ASP Solver: DLV
  - PHP
- dlvhex DL Plugin
  - `www.kr.tuwien.ac.at/research/systems/dlvhex/dlplugin.html`
  - DL Engine: RacerPro
  - ASP Solver: DLV or Clingo
  - C++

# Problem Statement

Loose Coupling - revisited

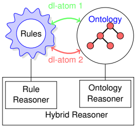
- **Advantage:**
  - clean semantics, can use legacy systems
  - fairly easy to incorporate further knowledge formats (e.g. RDF)
  - supportive to privacy, information hiding



# Problem Statement

Loose Coupling - revisited

- **Advantage:**
  - clean semantics, can use legacy systems
  - fairly easy to incorporate further knowledge formats (e.g. RDF)
  - supportive to privacy, information hiding




- **Drawback**: *impedance mismatch, performance*
  - dl-program evaluation needs multiple calls of a dl-reasoner
  - Calls are expensive
    - ∗ optimizations (caching, pruning ...)
  - exponentially many calls may be unavoidable
  - Even polynomially many calls might be too costly

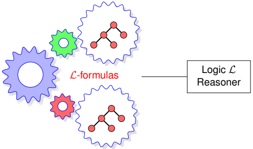
# Motivation

## Goal

Improving the efficiency of reasoning over $\mathrm{dl}$-Programs

## Approach

Converting the evaluation problem into one for a single reasoning engine




- Transform dl-program $\Pi$ into an (equivalent) knowledge base in formalism $\mathcal{L}$ for evaluation (*uniform evaluation*)
  - $\mathcal{L}$ = FO Logic (SQL): **MOR**; acyclic $\Pi$ over $\mathcal{DL}$-*Lite*, using an RDBMS
  - $\mathcal{L} =$ Datalog$^{\neg}$ (ASP)

# Questions arising from Datalog$^{\neg}$ rewritings of $\mathrm{dl}$-Progmas

- Possibility of transformation?
  - Is there a general framework?
  - Which DLs can be transformed?
- Suitable for implementation?
  - Can we reuse existing tools?
- Performance?
  - Benchmarks?
  - How to evaluate?

# Inline Evaluation of $\mathrm{dl}$-Programs by Datalog rewriting

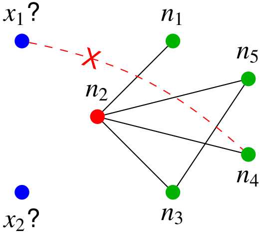
**Idea:**

- for Datalog-rewritable ontologies, we may replace $\mathrm{dl}$-atoms $DL[\lambda; Q](\vec{c})$ with Datalog programs evaluating the atoms
- the result is computed in an atom $Q_\lambda(\vec{c})$
- rewrite the $\mathrm{dl}$-rules to ordinary rules, by replacing $\mathrm{dl}$-atoms
- evaluate the resulting logic program using a Datalog engine / ASP solver

Demonstrate the method on the Network example

# Network Example

$\Pi = (\mathcal{O}, P)$




Ontology $\mathcal{O}$ :

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

---

Rules $P$

$$newnode(x_1). \quad newnode(x_2).$$
$$overloaded(X) \leftarrow \mathrm{DL}[wired \uplus connect; HighTrafficNode](X).$$
$$connect(X, Y) \leftarrow newnode(X), \mathrm{DL}[Node](Y), not\ overloaded(Y), not\ excl(X, Y).$$
$$excl(X, Y) \leftarrow connect(X, Z), \mathrm{DL}[Node](Y), Y \neq Z.$$
$$excl(X, Y) \leftarrow connect(Z, Y), newnode(Z), newnode(X), Z \neq X.$$
$$excl(x_1, n_4).$$

# Network Example, cont'd

1. Rewriting the ontology

- The DL component $\mathcal{O}$ is in OWL 2 RL resp. $\mathcal{LDL}^+$, which is Datalog-rewritable ($\mathcal{LDL}^+$ will be introduced later).
- We transform $\mathcal{O}$ to the Datalog program $\Phi_{\mathcal{LDL}^+}(\mathcal{O})$:

$$
\begin{aligned}
wired^-(Y,X) &\leftarrow wired(X,Y) \quad wired(Y,X) \leftarrow wired^-(X,Y)\\
\top(X) &\leftarrow wired(X,Y) \quad \top(Y) \leftarrow wired(X,Y)\\
\top(X) &\leftarrow wired^-(X,Y) \quad \top(Y) \leftarrow wired^-(X,Y)\\
&\%\text{axiom } \geq 1.wired \sqsubseteq Node\\
Node(Y) &\leftarrow wired(X,Y)\\
&\%\text{axiom } \top \sqsubseteq \forall wired.Node\\
Node(Y) &\leftarrow wired(X,Y), \top(X)\\
&\%\text{axiom } \geq 4.wired \sqsubseteq HighTrafficNode\\
HighTrafficNode(X) &\leftarrow wired(X,Y_1), wired(X,Y_2), wired(X,Y_3), wired(X,Y_4),\\
&\quad Y_1 \neq Y_2, Y_1 \neq Y_3, \ldots, Y_3 \neq Y_4.
\end{aligned}
$$

$$wired(n_1,n_2)\ wired(n_2,n_3)\ wired(n_2,n_4), wired(n_2,n_5).\ wired(n_3,n_4).\ wired(n_3,n_5).$$

# Network Example, cont'd

## 2. Duplicating for dl-inputs

dl-atoms in $\Pi$:

$$\text{DL}[Node](Y), \quad \text{DL}[wired \uplus connect; HighTrafficNode](X)$$

- the dl-queries in are just instance queries, so given by $Node(Y)$ resp. $HighTrafficNode(X)$
- Each DL-atom sends up a different input $\lambda$ to $\mathcal{O}$ and so entailments for the $\lambda$'s might be different.
- To this purpose, we copy $\Phi_{\mathcal{LDL}^+}(\mathcal{O})$ to new disjoint equivalent versions for each DL-input $\lambda$
- For the set $\Lambda_P = \{\lambda_1 = \epsilon, \lambda_2 = wired \uplus connect\}$, we have

  - $\Phi_{\mathcal{LDL}^+,\lambda_1}(\mathcal{O}) = \{Node_{\lambda_1}(X) \leftarrow wired_{\lambda_1}(X,Y), \ldots\}$ and
  - $\Phi_{\mathcal{LDL}^+,\lambda_2}(\mathcal{O}) = \{Node_{\lambda_2}(X) \leftarrow wired_{\lambda_2}(X,Y), \ldots\}$

# Network Example, cont'd

3. Rewriting dl-rules to ordinary rules

- To rewrite DL-rules $P$ into ordinary rules $P^{ord}$, we simply replace each DL-atom $DL[\lambda; Q](\vec{t})$ by a new atom $Q_{\lambda}(\vec{t})$.

# Network Example, cont'd

## 3. Rewriting dl-rules to ordinary rules

- To rewrite DL-rules $P$ into ordinary rules $P^{ord}$, we simply replace each DL-atom $DL[\lambda; Q](\vec{t})$ by a new atom $Q_\lambda(\vec{t})$.

$P^{ord}$

$$
\begin{aligned}
newnode(x_1).\quad newnode(x_2).& \\
overloaded(X) &\leftarrow HighTrafficNode_{\lambda_2}(X). \\
connect(X,Y) &\leftarrow newnode(X), Node_{\lambda_1}(Y), \\
&\quad\; not\ overloaded(Y), not\ excl(X,Y). \\
excl(X,Y) &\leftarrow connect(X,Z), Node_{\lambda_1}(Y), Y \neq Z. \\
excl(X,Y) &\leftarrow connect(Z,Y), newnode(Z), newnode(X), Z \neq X. \\
excl(x_1, n_4).&
\end{aligned}
$$

# Network Example, cont'd

## 4. Rewriting $\mathrm{dl}$-atom Input to Datalog rules

- The inputs $\lambda$ for the copies $\Phi_{\mathcal{LDL}^+,\lambda}$ can be transferred by rules:
  - $\lambda_1 = \epsilon$ (no input); no rule needed
  - $\lambda_2 = wired \uplus connect$:

$$wired_{\lambda_2}(X, Y) \leftarrow connect(X, Y).$$

# Network Example, cont'd

## 5. Calling the Datalog reasoner

- Now we have transformed all the components into a Datalog$^{\neg}$ program

$$\Psi_{\mathcal{LDL}^+}(\Pi) = \Phi_{\mathcal{LDL}^+,\lambda_1}(\Sigma) \cup \Phi_{\mathcal{LDL}^+,\lambda_2}(\Sigma) \cup P^{ord} \cup P(\Lambda_P).$$

- We can send it to a datalog engine, e.g. DLV, and compute its answer set or the well-founded model
- The answer sets of $\Psi_{\mathcal{LDL}^+}(\Pi)$, filtered to $connect$, $overloaded$, $newnode$, $excl$, are the (strong) answer sets of $\Pi$
- $\Psi_{\mathcal{LDL}^+}(\Pi) \models_{wf} p(a)$ iff $\Pi \models_{wf} p(a)$ for ground atom

  Example: $\Psi_{\mathcal{LDL}^+}(\Pi) \models_{wf} overloaded(n_2)$

# dl-program Transformation (General Case)

$\mathcal{DL}$: Datalog-rewritable Description Logic

$\Pi = (\mathcal{O}, P)$: a dl-program with dl-atoms $DL[\lambda_i; Q_i](\vec{t_i})$, $1 \leq i \leq n$, where

- $\lambda_i = S_{i,1} \uplus p_{i,1}, \ldots, S_{i,m_i} \uplus p_{i,m_i}$, and
- $Q_i$ is an instance query.

Let $\Lambda_P = \{\lambda_1, \ldots, \lambda_n\}$ and define

$$\Psi_{\mathcal{DL}}(\Pi) := \bigcup_{\lambda_i \in \Lambda_P} \Phi_{\mathcal{DL},\lambda_i}(\mathcal{O}) \cup P^{ord} \cup \rho(\Lambda_P) \cup T_P$$

where

- $\Phi_{\mathcal{DL},\lambda_i}(\mathcal{O})$ is a copy of $\Phi_{\mathcal{DL}}(\mathcal{O})$ with all predicates subscripted with $\lambda_i$
- $\rho(\Lambda_P)$ consists of rules $S_{i,j,\lambda}(\vec{X}_{i,j}) \leftarrow p_{i,j}(\vec{X}_{i,j})$, for all $\lambda_i \in \Lambda_P$
- $P^{ord}$ is $P$ with each $DL[\lambda_i; Q_i](\vec{t_i})$ replaced by a new atom $Q_{\lambda_i}(\vec{t_i})$
- $T_P = \{\top(a), \top^2(a, b) \mid a, b \text{ occur in } P\}$

# dl-program Transformation (General Case)

## Theorem

*Let $\Pi = (\mathcal{O}, P)$ be a dl-program over Datalog-rewritable $\mathcal{DL}$. Then*

(1) *for every $a \in HB_P$, $\Pi \models_{wf} a$ iff $\Psi_{\mathcal{DL}}(\Pi) \models_{wf} a$;*

(2) *the answer sets of $\Pi$ correspond 1-1 to the answer sets of $\Psi(\Pi)$, s.t.*

  (i) *every answer set of $\Pi$ is expendable to an answer set of $\Psi(\Pi)$; and*

  (ii) *for every answer set $J$ of $\Psi(\Pi)$, its restriction $I = J\mid_{HB_P}$ to $HB_P$ is an answer set of $\Pi$.*

# Datalog-Rewritable DLs

## Definition (Datalog-rewritable)

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 $\mathcal{O}$,

1. $\mathcal{O} \models Q(\mathbf{o})$ iff $\Phi_{\mathcal{DL}}(\mathcal{O}) \models Q(\mathbf{o})$ for any concept or role name $Q$ from $\mathcal{O}$, and individuals $\mathbf{o}$ from $\mathcal{O}$;
2. $\Phi_{\mathcal{DL}}$ is *modular*, i.e., for $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox, $\Phi_{\mathcal{DL}}(\mathcal{O}) = \Phi_{\mathcal{DL}}(\mathcal{T}) \cup \mathcal{A}$;

Further properties: A DL $\mathcal{DL}$ is

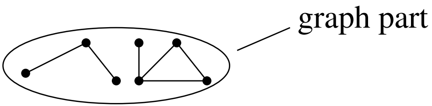
- *polynomial Datalog-rewritable*, if $\mathcal{DL}$ is Datalog-rewritable and $\Phi_{\mathcal{DL}}(\mathcal{O})$ is computable in polynomial time;
- *non-uniform Datalog-rewritable*, if only condition (1) of Datalog-rewritability holds for $\mathcal{DL}$.

# Example Datalog-Rewritable DLs

- $\mathcal{LDL}^+$ [Heymans *et al.*, 2010]:
  lightweight ontology language, extending in essence core OWL 2 RL with singleton nominals, role conjunctions, and transitive closure
- $\mathcal{SROEL}(\sqcap, \times)$ [Krötzsch, 2010]:
  superset of OWL 2 EL [Motik *et al.*, 2008] resp. $\mathcal{EL}$++
  - disregarding datatypes
  - adding (restricted) conjunction of roles ($R \sqcap S$), local reflexivity ($Self$), concept production ($C \times D \sqsubseteq T$, $R \sqsubseteq C \times D$)
- $\mathcal{SROEL}(\times)$ [Krötzsch, 2011]
- Horn-$\mathcal{SHIQ}$ [Ortiz *et al.*, 2010]:
  Horn fragment of $\mathcal{SHIQ}$
- $\mathcal{SROIQ}$-RL [Bozzato and Serafini, 2013]:
  restriction of $\mathcal{SROIQ}$ for OWL 2 RL

# $\mathcal{LDL}^+$

- $\mathcal{LDL}^+$ forbids in axioms $X \sqsubseteq Y$
  - disjunction $C \sqcup D$ in $Y$
  - existentials $\exists R$ in $Y$
- Viewing $X \sqsubseteq Y$ as rule $Y \leftarrow X$, it distinguishes head (h) and body (b) concepts/roles, for occurrence in $Y$ resp. $X$
- $\mathcal{LDL}^+$ shares properties with datalog programs:
  - It can express transitive closure (via an operator $^+$)
  - An $\mathcal{LDL}^+$ ontology $\mathcal{O}$ has a least model in each domain
  - For query answering, we can exclude *unnamed individuals* (i.e., use the *active* domain of individuals occurring in $\mathcal{O}$



# Syntax of $\mathcal{LDL}^+$– Roles

head (h-) and body (b-) restrictions on roles in $\mathcal{LDL}^+$ axioms

- *h-roles* ($h$ for *head*) $S, T$ are
  - (i) *role names* $R$,
  - (ii) *role inverses* $S^-$,
  - (iii) *role conjunctions* $S \sqcap T$, and
  - (iv) *role top* $\top^2$;
- *b-roles* ($b$ for *body*) $S, T$ are the same as h-roles, plus
  - (v) *role disjunctions* $S \sqcup T$,
  - (vi) *role sequences* $S \circ T$,
  - (vii) *transitive closures* $S^+$, and
  - (viii) *role nominals* $\{(o_1, o_2)\}$, where $o_1, o_2$ are individuals.

# Syntax of $\mathcal{LDL}^+$– Concepts

head (h-) and body (b-) restrictions on concepts in $\mathcal{LDL}^+$ axioms

- *basic concepts* $C, D$ are concept names $A$, $\top$, and conjunctions $C \sqcap D$;
- *h-concepts* are
  - (i) *basic concepts* $B$, and
  - (ii) *value restrictions* $\forall S.B$ where $S$ is a *b-role*;
- *b-concepts* $C, D$ are
  - (i) *basic concepts* $B$,
  - (ii) *disjunctions* $C \sqcup D$,
  - (iii) *exists restrictions* $\exists S.C$,
  - (iv) *atleast restrictions* $\geq nS.C$, and
  - (v) *nominals* $\{o\}$, where $S$ is a b-role, and $o$ is an individual.

# Transformation of $\mathcal{LDL}^+$ to Datalog

The transformation $\Phi_{\mathcal{LDL}^+}(\mathcal{O})$ of an $\mathcal{LDL}^+$ ontology $\mathcal{O}$ to Datalog contains the following elements:

- transformation of the $\mathcal{LDL}^+$ axioms in $\mathcal{O}$;
- transformation of the *closure* of $\mathcal{O}$.

## Definition (closure)

The *closure* of an $\mathcal{LDL}^+$ knowledge base $\mathcal{O}$, denoted $clos(\mathcal{O})$, as the smallest set containing

- all subexpressions that occur in $\mathcal{O}$ (both roles and concepts) except value restrictions, and
- for each role name occurring in $\mathcal{O}$, its inverse.

# Transformation Rules

- Axiom translation:

| | |
|---|---|
| $B \sqsubseteq H$ | $H(X) \leftarrow B(X)$ |
| $B \sqsubseteq \forall E.A$ | $A(Y) \leftarrow B(X), E(X,Y).$ |
| $S \sqsubseteq T$ | $T(X,Y) \leftarrow S(X,Y)$ |

- closure translation:

| | |
|---|---|
| role name $P$ | $P(X,Y) \leftarrow P^-(Y,X)$ |
| concept name $A$ | $\top(X) \leftarrow A(X)$ |
| role name ($R$) | $\top(X) \leftarrow R(X,Y) \quad \top(Y) \leftarrow R(X,Y)$ |
| $\top$ | $\top^2(X,Y) \leftarrow \top(X), \top(Y).$ |
| $D = \{o\}$ | $D(o) \leftarrow$ |
| $D = D_1 \sqcap D_2$ | $D(X) \leftarrow D_1(X), D_2(X)$ |
| $D = D_1 \sqcup D_2$ | $D(X) \leftarrow D_1(X) \quad D(X) \leftarrow D_2(X)$ |
| $D = \exists E.D_1$ | $D(X) \leftarrow E(X,Y), D_1(Y)$ |
| $D = \ \geq n\, E.D_1$ | $D(X) \leftarrow E(X,Y_1), D(Y_1), \ldots, E(X,Y_n), D(Y_n),$ $Y_1 \neq Y_2, \ldots, Y_i \neq Y_j, \ldots, Y_{n-1} \neq Y_n$ |
| $E = \{(o_1,o_2)\}$ | $E(o_1,o_2) \leftarrow$ |
| $E = F^-$ | $E(X,Y) \leftarrow F(Y,X)$ |
| $E = E_1 \sqcap E_2$ | $E(X,Y) \leftarrow E_1(X,Y), E_2(X,Y)$ |
| $E = E_1 \sqcup E_2$ | $E(X,Y) \leftarrow E_1(X,Y) \quad E(X,Y) \leftarrow E_2(X,Y)$ |
| $E = E_1 \circ E_2$ | $E(X,Y) \leftarrow E_1(X,Z), E_2(Z,Y)$ |
| $E = F^+$ | $E(X,Y) \leftarrow F(X,Y) \qquad E(X,Y) \leftarrow F(X,Z), E(Z,Y)$ |

# Formal Properties

**Theorem**

*For every $\mathcal{LDL}^+$ ontology $\mathcal{O}$,*

(i) $\mathcal{O} \models C(a)$ *iff* $\Phi_{\mathcal{LDL}^+}(\mathcal{O}) \models C(a)$

(ii) $\mathcal{O} \models R(a,b)$ *iff* $\Phi_{\mathcal{LDL}^+}(\mathcal{O}) \models R(a,b)$.

Notes:

- $\Phi_{\mathcal{LDL}^+}(\mathcal{O})$ can be constructed in polynomial time from $\mathcal{O}$ (unary encoding of counting $\geq n\,R$)
- can be evaluted in polynomial time (rule matching is polynomial)
- the above result extends to CQs and UCQs $Q(\vec{X})$:

  $\vec{c} \in ans(Q, \mathcal{O})$ iff $\Phi_{\mathcal{LDL}^+}(\mathcal{O}) \cup Q(\vec{X}) \models q(\vec{c})$

# $\mathcal{SROEL}(\sqcap, \times)$

- $\mathcal{SROEL}(\sqcap, \times)$ is in essence a superset of OWL 2 EL
  Differences:
  - disregards datatypes
  - adding conjunction of roles ($R \sqcap S$), local reflexivity ($Self$), concept production ($C \times D \sqsubseteq T$, $R \sqsubseteq C \times D$)
  - restrictions on role occurrences in a KB (simplicity, range restrictions), but not role regularities
- $\mathcal{SROEL}(\sqcap, \times)$ has polynomial complexity (sat, instance checking)
- [Krötzsch, 2010] describes a proof system for instance checking over a $\mathcal{SROEL}(\sqcap, \times)$ ontology
- This proof system can be naturally encoded in a logic program, viewing axioms $\alpha$ as facts and inference rules $\frac{\alpha_1,\ldots,\alpha_n}{\alpha}$ as rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$
- A universal (schematic) encoding in Datalog is possible

# Transformation of $\mathcal{SROEL}(\sqcap, \times)$ to Datalog

- $\mathcal{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)

# Transformation of $\mathcal{SROEL}(\sqcap, \times)$ to Datalog

- $\mathcal{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)
  - view the inference rules $\frac{\alpha}{\alpha_1, \ldots, \alpha_n}$ as LP rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$
    E.g., $\frac{C \sqsubseteq D,\ C(a)}{D(a)}$ can be viewed as rule $D(a) \leftarrow \sqsubseteq(C, D), C(a)$

# Transformation of $\mathcal{SROEL}(\sqcap, \times)$ to Datalog

- $\mathcal{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)
  - view the inference rules $\frac{\alpha}{\alpha_1, \ldots, \alpha_n}$ as LP rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$
    E.g., $\frac{C \sqsubseteq D,\ C(a)}{D(a)}$ can be viewed as rule $D(a) \leftarrow \sqsubseteq(C, D), C(a)$
- Use *reification* to obtain a Datalog representation

$$\Phi_{\mathcal{EL}}(\mathcal{O}) = I_{inst}(\mathcal{O}) \cup P_{inst}$$

where $I_{inst}(\mathcal{O})$ encodes $\mathcal{O}$ and $P_{inst}$ is a fixed set of rules (schemata)

- names: $C \rightsquigarrow cls(C)$; $R \rightsquigarrow rol(R)$; $a \rightsquigarrow nom(a)$
- assertions: e.g $C(a) \rightsquigarrow isa(a, C)$; $R(a, b) \rightsquigarrow triple(a, R, b)$
- axioms: e.g. $A \sqsubseteq C \rightsquigarrow subClass(A, C)$,

# Transformation of $\mathcal{SROEL}(\sqcap, \times)$ to Datalog

- $\mathcal{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts

    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)
  - view the inference rules $\frac{\alpha}{\alpha_1, \ldots, \alpha_n}$ as LP rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$

    E.g., $\frac{C \sqsubseteq D,\ C(a)}{D(a)}$ can be viewed as rule $D(a) \leftarrow \sqsubseteq(C, D), C(a)$
- Use *reification* to obtain a Datalog representation

$$\Phi_{\mathcal{EL}}(\mathcal{O}) = I_{inst}(\mathcal{O}) \cup P_{inst}$$

  where $I_{inst}(\mathcal{O})$ encodes $\mathcal{O}$ and $P_{inst}$ is a fixed set of rules (schemata)
  - names: $C \rightsquigarrow cls(C)$; $R \rightsquigarrow rol(R)$; $a \rightsquigarrow nom(a)$
  - assertions: e.g $C(a) \rightsquigarrow isa(a, C)$; $R(a, b) \rightsquigarrow triple(a, R, b)$
  - axioms: e.g. $A \sqsubseteq C \rightsquigarrow subClass(A, C)$,
- Make reified rules generic using variables

  E.g. $isa(a, D) \leftarrow subClass(C, D), isa(a, C)$ gets
  $isa(X, Z) \leftarrow subClass(Y, Z), isa(X, Y)$

# Rewrtings of $\mathcal{LDL}^+$ vs $\mathcal{SROEL}(\sqcap, \times)$

- $\mathcal{LDL}^+$
  - TBox assertions $\rightsquigarrow$ Rules
  - Direct rewrting
- $\mathcal{SROEL}(\sqcap, \times)$
  - TBox assertions $\rightsquigarrow$ Facts
  - Fixed set of rules
  - Reification based rewriting
  - The resulting program is always recursive

# DReW Reasoner

**DReW** prototype: uniform $\mathrm{dl}$-program evaluation in Datalog$^{\neg}$

`http://www.kr.tuwien.ac.at/research/systems/drew/`

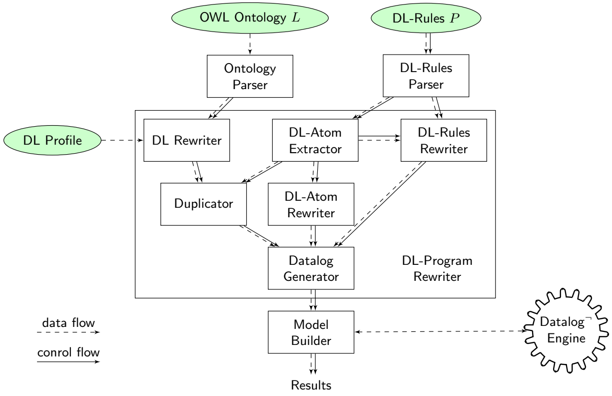
at GitHub: `https://github.com/ghxiao/drew`

- written in Java
- ontology parser: OWL-API
- Datalog reasoner: DLV (inside DReW); Clingo may be used as well (compute rewriting, via command line)

**Features in DReW v0.3**

- ontology component
  - OWL 2 RL ($\mathcal{LDL}^+$)
  - OWL 2 EL ($\mathcal{SROEL}(\sqcap, \times)$)
- rule formalism
  - $\mathrm{dl}$-Programs (answer sets, well founded semantics)
  - CQs under DL-safeness
  - Terminological Default Reasoning (frontend)

# System Architecture (Core)



# Example Usage

Example with Network dl-Program 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, cont'd

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) }
```

# Benchmark Scenarios

- Graph
  - Ontologies derived from Random Graph Generator
  - Programs for computing the transitive closure
- University
  - Ontologies from LUBM and ModLUBM
  - DL-Programs for computing e.g. co-author relations
- GeoData
  - TBox from MyITS Project; ABox from Open Street Map
  - semantically enriched spatial queries
- EDI (Electronic data interchange)
  - TBox from EDIMine project; ABox from EDI messages
  - Rule-based reasoning over Business ontologies
- Policy
  - EL ontoloigy
  - Default Rules modeling Role Based Access Control

# Platform

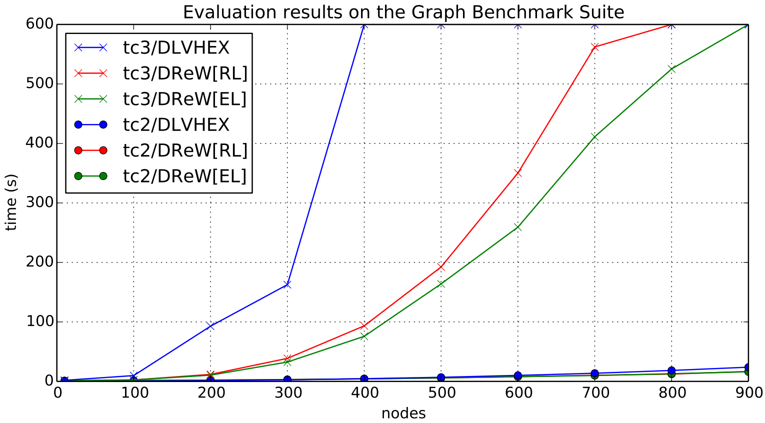
- Ubuntu 12.04 Linux Server
- DReW 0.3
  - Java: Oracle JDK 1.7.0_21, JVM memory 6G
  - DLV 2012-12-17
- dlvhex 1.7.2
  - RacerPro 1.9.2 beta (released on 2007-10-25)
  - DLV 2012-12-17
- HTCondor for scheduling the runs

# Graph Benchmark Suite

- TBox: Empty
- ABox: Generated by a random graph generator
- DL-Programs for Computing transitive closure

| | |
|---|---|
| $tc_2$ | *extracts the arc relations from the ontology and computes the closure by linear recursion*<br>edge(X, Y) :- DL[arc](X, Y).<br>tc(X, Y) :- edge(X, Y).<br>tc(X, Y) :- edge(X, Z), tc(Z, Y). |
| $tc_3$ | *extracts the arc relations from the ontology and computes the closure by recursion while feeding back the arc relations*<br>tc(X, Y) :- DL[arc](X, Y).<br>tc(X, Y) :- DL[arc$\uplus$ tc; arc](X, Z), tc(Z, Y). |

# Graph Benchmark Suite Evaluation



# GeoData Benchmark Suite

- TBox
  - Ontology developed in the MyITS Project
  - GeoConceptsMyITS-v0.9-Lite[1]
- ABox
  - Features derived from Open Street Map
  - Geo Relations (next, within) computed by our scripts
  - Four Areas: Vienna, Salzburg, Austria, Upper Bavaria
- Programs
  - Geo Relation enriched Queries

| | #IND | #CA | #OPA | #DPA | #next | #within | File Size |
|---|---|---|---|---|---|---|---|
| Salzburg | 12971 | 13037 | 539 | 19513 | 79615 | 455 | 11M |
| Vienna | 33405 | 33531 | 1303 | 50520 | 292985 | 2610 | 36M |
| Austria | 150911 | 151616 | 5326 | 222189 | 893438 | 6712 | 133M |
| Upper Bavaria | 70837 | 71201 | 2182 | 106140 | 414512 | 3772 | 55M |

Table: ABox Sizes of the GeoData benchmark suite

[1] http://www.kr.tuwien.ac.at/staff/patrik/GeoConceptsMyITS-v0.9-Lite.owl

# GeoData Benchmark Suite – Example Program

## P5: List all the Italian restaurants next to a subway station which can be reached from “Karlsplatz” by one change.

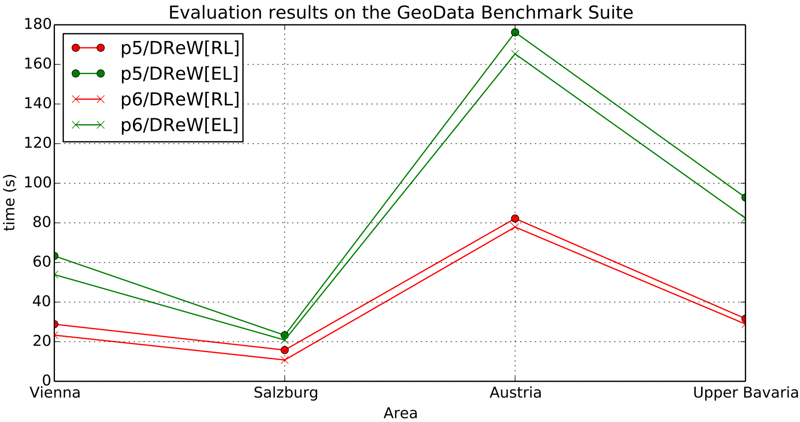
```
                   q(YN, ZN, L1, L2) :- metro_connect_1(L1,L2,“Karlsplatz”, YN),
                                        DL[SubwayStation](Y),
                                        DL[featurename](Y, YN), DL[Restaurant](Z),
                                        DL[next](Y, Z), DL[featurename](Z, ZN),
                                        DL[hasCuisine](Z, “ItalianCuisine”).
        metro_next(Line, Stop1, Stop2) :- metro_next(Line, Stop2, Stop1).
       metro_connect_0(L, Stop1, Stop2) :- metro_next(L, Stop1, Stop2).
       metro_connect_0(L, Stop1, Stop2) :- metro_connect_0(L, Stop1, Stop3),
                                        metro_connect_0(L, Stop3, Stop2).
metro_connect_1(L1, L2, Stop1, Stop2) :- metro_connect_0(L1, Stop1, Stop3),
                                        metro_connect_0(L2, Stop3, Stop2), L1 != L2.
% and the facts of the subway lines
metro_next(“U1”,“Reumannplatz” , “Keplerplatz”).
metro_next(“U1” , “Keplerplatz” , “Suedtiroler Platz”).
. . .
metro_next(“U6”, “Handelskai”, “Neue Donau”).
metro_next(“U6”, “Neue Donau”, “Floridsdorf”).
```

# GeoData Benchmark Suite – Example Program

## P6: Select restaurants next to “Karlsplatz” with preference: ChineseCuisine > AsianCuisine > Other.

```
               restaurant(X) :- DL[Restaurant](X), DL[next](X,Y),
                                DL[SubwayStation](Y), DL[featurename](Y, “Karlsplatz”).
       chinese_restaurant(X) :- restaurant(X), DL[hasCuisine](X, “ChineseCuisine”).
         asian_restaurant(X) :- restaurant(X), DL[hasCuisine](X, “AsianCuisine”).
 exists_chinese_restaurant :- chinese_restaurant(X), restaurant(X).
   exists_asian_restaurant :- asian_restaurant(X), restaurant(X).
                      sel(X) :- chinese_restaurant(X), exists_chinese_restaurant.
                      sel(X) :- asian_restaurant(X), not exists_chinese_restaurant,
                                exists_asian_restaurant.
                      sel(X) :- restaurant(X), not exists_asian_restaurant,
                                not exists_chinese_asian_restaurant.
                       q(XN) :- sel(X), DL[featurename](X, XN).
```

# Graph Benchmark Suite Evaluation




Note: dlvhex [DL, RacerPro] does not terminate in 20mins

# Policy Benchmark

Terminological default KB $\Delta = \langle L, D \rangle$, where the the TBox of $L$ and the defaults $D$ are shown bellow:

$$\mathcal{T} = \left\{ \begin{array}{l} \mathit{Staff} \sqsubseteq \mathit{User}, \quad \mathit{Blacklisted} \sqsubseteq \mathit{Staff}, \quad \mathit{Deny} \sqcap \mathit{Grant} \sqsubseteq \bot, \\ \mathit{UserRequest} \equiv \exists \mathit{hasAction}.\mathit{Action} \sqcap \exists \mathit{hasSubject}.\mathit{User} \sqcap \exists \mathit{hasTarget}.\mathit{Project}, \\ \mathit{StaffRequest} \equiv \exists \mathit{hasAction}.\mathit{Action} \sqcap \exists \mathit{hasSubject}.\mathit{Staff} \sqcap \exists \mathit{hasTarget}.\mathit{Project}, \\ \mathit{BlacklistedStaffRequest} \equiv \mathit{StaffRequest} \sqcap \exists \mathit{hasSubject}.\mathit{Blacklisted} \end{array} \right\}$$

$$D = \left\{ \begin{array}{l} \mathit{UserRequest}(X) : \mathit{Deny}(X) / \mathit{Deny}(X), \\ \mathit{StaffRequest}(X) : \neg \mathit{BlacklistedStaffRequest}(X) / \mathit{Grant}(X), \\ \mathit{BlacklistedStaffRequest}(X) : \top / \mathit{Deny}(X) \end{array} \right\}$$

Informally, $D$ expresses that

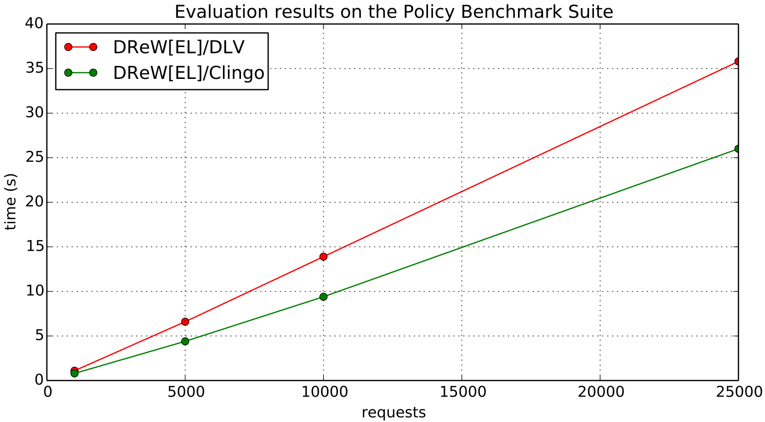
- users normally are denied access to files,
- staff is normally granted access to files,
- while to blacklisted staff any access is denied.

# Policy Benchmark Suite – dl-Programs

The default theory $D$ is equivalent to the following highly recursive dl-Programs

$$
\begin{aligned}
Deny^+(X) &\leftarrow DL[\lambda; UserRequest](X), not\ DL[\lambda'; \neg Deny](X)\\
Grant^+(X) &\leftarrow DL[\lambda; StaffRequest](X), not\ DL[\lambda'; BlacklistedStaffRequest](X)\\
Deny^+(X) &\leftarrow DL[\lambda; BlacklistedStaffRequest](X).\\
in\_Deny(X) &\leftarrow not\ out\_Deny(X)\\
out\_Grant(X) &\leftarrow not\ in\_Grant(X)\\
fail &\leftarrow DL[\lambda'; Deny](X), out\_Deny(X), not\ fail\\
fail &\leftarrow DL[\lambda; Deny](X), in\_Deny(X), not\ fail\\
fail &\leftarrow DL[\lambda; Deny](X), out\_Deny(X), not\ fail\\
fail &\leftarrow DL[\lambda'; Grant](X), out\_Grant(X), not\ fail\\
fail &\leftarrow DL[\lambda; Grant](X), in\_Grant(X), not\ fail\\
fail &\leftarrow DL[\lambda; Grant](X), out\_Grant(X), not\ fail
\end{aligned}
$$

where $\lambda' = \{Deny \uplus in\_Deny, Grant \uplus in\_Grant\}$, and
$\lambda = \{Deny \uplus Deny^+, Grant \uplus Grant^+\}$.




dlvhex with DF-front end can only handle up to 5 requests in almost 3 mins.

# Observations from the Evaluation

- dl-Programs are exprssive and useful as a query language
- the DReW system outperforms dlvhex [DL, RacerPro] in general, especially for dl-Programs of complex structure or dl-programs with large instances
- DReW scales polynomially on large ABoxes in general
- In most of the evaluations, the direct rewriting approach (RL) is faster than the reification-based rewriting (EL)

# Summary

- dl-Programs: Loose coupling ontologies and rule
- current systems are not very efficient due to the overhead of calling external DL reasoners

## Contributions

- Theoretical Contributions
  - A framework of inline evaluation of dl-Programs by Datalog$^{\neg}$ rewriting
  - Identifying a class of Datalog-rewritable DLs
- Practical Contributions
  - DReW reasoner for Datalog-rewritable dl-Programs
  - Extensive evaluations on novel benchmark suites with promising results

# Ongoing / Future Work

- Optimization of the DReW system
- Experiments with other Backend Engines (e.g., RDBMS and $DLV^{\exists}$)
- More reasoning paradigm support, e.g. Closed World Assumption
- Supporting W3C standard OWL-RIF
- Further update operators ($\Cap$) and semantics

# Relevant Publications


- G. Xiao, T. Eiter, and S. Heymans, "The DReW system for nonmonotonic DL-Programs", in *SWWS 2012*, Shenzhen City, China, November 2012.
- T. Eiter, T. Krennwallner, P. Schneider, and G. Xiao, "Uniform evaluation of nonmonotonic DL-Programs", in *FoIKS 2012*, Springer, March 2012.
- T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao, "Query rewriting for Horn-SHIQ plus rules", in *AAAI 2012* , AAAI Press, 2012.
- G. Xiao and T. Eiter, "Inline evaluation of hybrid knowledge bases – PhD description", in *RR 2011*, Springer, 2011.
- S. Heymans, T. Eiter, and G. Xiao, "Tractable reasoning with DL-Programs over datalog-rewritable description logics", in *ECAI 2010*, IOS Press, 2010.
- T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao, "Towards practical query answering for Horn-SHIQ", in *DL-2012*, CEUR-WS.org, 2012.
- G. Xiao, S. Heymans, and T. Eiter, "DReW: a reasoner for datalog-rewritable description logics and DL-programs", in *Informal Proc. 1st Int'l Workshop on Business Models, Business Rules and Ontologies (BuRO 2010)*, 2010.

# References I


- Jos de Bruijn, Thomas Eiter, Axel Florian Polleres, and Hans Tompits.
  Embedding non-ground logic programs into autoepistemic logic for knowledge base combination.
  In Manuela Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 304–309. AAAI Press/IJCAI, 2007.
  Extended paper to appear in *ACM Trans. Computational Logic*.
- Jos de Bruijn, David Pearce, Axel Polleres, and Agustín Valverde.
  Quantified equilibrium logic and hybrid rules.
  In *RR*, pages 58–72, 2007.
- Jos de Bruijn, Philippe Bonnard, Hugues Citeau, Sylvain Dehors, Stijn Heymans, Jörg Pührer, and Thomas Eiter.
  Combinations of rules and ontologies: State-of-the-art survey of issues.
  Technical Report Ontorule D3.1, Ontorule Project Consortium, June 2009.
  `http://ontorule-project.eu/`.
- T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits.
  Combining answer set programming with description logics for the Semantic Web.
  *Artificial Intelligence*, 172(12-13):1495–1539, 2008.
- T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer.
  Well-founded semantics for description logic programs in the Semantic Web.
  *ACM Trans. Comput. Log.*, 12(2):11, 2011.

# References II


M. Gelfond and V. Lifschitz.
The Stable Model Semantics for Logic Programming.
In *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.

M. Gelfond and V. Lifschitz.
Classical Negation in Logic Programs and Disjunctive Databases.
*New Generation Computing*, 9:365–385, 1991.

B. N. Grosof, I. Horrocks, R. Volz, and S. Decker.
Description logic programs: Combining logic programs with description logics.
In *Proceedings of the 12th International World Wide Web Conference (WWW'03)*, pages 48–57. ACM Press, 2003.

I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean.
SWRL: A semantic web rule language combining OWL and RuleML.
W3C Member Submission, World Wide Web Consortium, 2004.

M. Kifer, G. Lausen, and J. Wu.
Logical foundations of object-oriented and frame-based languages.
*Journal of the ACM*, 42(4):740–843, 1995.

# References III


M. Knorr, J.J. Alferes, and P. Hitzler.
A coherent well-founded model for hybrid MKNF knowledge bases.
In *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 99–103. IOS Press, 2008.

M. Krötzsch, S. Rudolph, and P. Hitzler.
Description logic rules.
In *Proc. ECAI*, pages 80–84. IOS Press, 2008.

M. Krötzsch, S. Rudolph, and P. Hitzler.
ELP: Tractable rules for OWL 2.
In *Proc. ISWC 2008*, pages 649–664, 2008.

Alon Y. Levy and Marie-Christine Rousset.
Combining horn rules and description logics in CARIN.
*Artificial Intelligence*, 104:165 – 209, 1998.

Boris Motik and Riccardo Rosati.
Reconciling description logics and rules.
*Journal of the ACM*, 2010.
To appear.

Boris Motik, Ulrike Sattler, and Rudi Studer.
Query answering for OWL-DL with rules.
*J. Web Sem.*, 3(1):41–60, 2005.

# References IV


Riccardo Rosati.
On the decidability and complexity of integrating ontologies and rules.
*Journal of Web Semantics*, 3(1):61–73, 2005.

Riccardo Rosati.
$\mathcal{DL}$+$log$: Tight Integration of Description Logics and Disjunctive Datalog.
In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning* (*KR 2006*), pages 68–78. AAAI Press, 2006.

A. van Gelder, K.A. Ross, and J.S. Schlipf.
The Well-Founded Semantics for General Logic Programs.
*Journal of the ACM*, 38(3):620–650, 1991.

Kewen Wang, David Billington, Jeff Blee, and Grigoris Antoniou.
Combining description logic and defeasible logic for the semantic web.
In Grigoris Antoniou and Harold Boley, editors, *RuleML*, volume 3323 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2004.

Meghyn Bienvenu, Magdalena Ortiz, Mantas Simkus, and Guohui Xiao.
Tractable queries for lightweight description logics.
In Francesca Rossi, editor, *IJCAI*. IJCAI/AAAI, 2013.

# References V


- Loris Bozzato and Luciano Serafini.
  Materialization calculus for contexts in the semantic web.
  In Thomas Eiter, Birte Glimm, Yevgeny Kazakov, and Markus Krötzsch, editors, *Description Logics*, volume 1014 of *CEUR Workshop Proceedings*, pages 552–572. CEUR-WS.org, 2013.
- Diego Calvanese, Thomas Eiter, and Magdalena Ortiz.
  Answering regular path queries in expressive description logics: An automata-theoretic approach.
  In *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007)*, pages 391–396, 2007.
- Diego Calvanese, Thomas Eiter, and Magdalena Ortiz.
  Regular path queries in expressive description logics with nominals.
  In C. Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 714–720. AAAI Press/IJCAI, 2009.
- Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Šimkus.
  Query answering in the description logic Horn-SHIQ.
  In S. Hölldobler, C. Lutz, and H. Wansing, editors, *Proceedings 11th European Conference on Logics in Artificial Intelligence (JELIA 2008)*, number 5293 in LNCS, pages 166–179. Springer, 2008.
  doi:10.1007/978-3-540-87803-2_15.

# References VI


Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Šimkus.
Query answering in description logics with transitive roles.
In C. Boutilier, editor, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 759–764. AAAI Press/IJCAI, 2009.

Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Kien Trung-Tran, and Guohui Xiao.
Query rewriting for Horn-$\mathcal{SHIQ}$ plus rules.
In *Proceedings 26th Conference on Artificial Intelligence (AAAI '12), July 22-26, 2012, Toronto*. AAAI Press, 2012.

Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Kien Trung-Tran, and Guohui Xiao.
Towards practical query answering for Horn-$\mathcal{SHIQ}$.
In Yevgeny Kazakov, Domenico Lembo, and Frank Wolter, editors, *Proceedings of the 25th International Workshop on Description Logics (DL2012), June -9, Rome, Italy*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
11 pp. `http://ceur-ws.org/Vol-846`.

Georg Gottlob and Thomas Schwentick.
Rewriting ontological queries into small nonrecursive datalog programs.
In Rosati et al. [2011].

# References VII


Stijn Heymans, Thomas Eiter, and Guohui Xiao.
Tractable reasoning with DL-programs over Datalog-rewritable description logics.
In M. Wooldridge et al., editor, *Proceedings of the 19th Eureopean Conference on Artificial Intelligence, ECAI'2010, Lisbon, Portugal, August 16-20, 2010*, pages 35–40. IOS Press, 2010.

Ullrich Hustadt, Boris Motik, and Ulrike Sattler.
A decomposition rule for decision procedures by resolution-based calculi.
In F. Baader and A. Voronkov, editors, *Proceedings 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004)*, number 3452 in LNCS, pages 21–35. Springer, 2005.

Y. Kazakov.
Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies.
In Craig Boutilier, editor, *IJCAI*, pages 2040–2045, 2009.

Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev.
On (in)tractability of OBDA with OWL 2 QL.
In Rosati et al. [2011].

Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev.
The combined approach to query answering in dl-lite.
In *KR*, 2010.

# References VIII


Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler.
Complexity boundaries for Horn description logics.
In *AAAI'07*, pages 452–457. AAAI Press, 2007.

Markus Krötzsch.
Efficient inferencing for OWL EL.
In Tomi Janhunen and Ilkka Niemelä, editors, *JELIA*, volume 6341 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2010.

Markus Krötzsch.
Efficient rule-based inferencing for OWL EL.
In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2668–2673. IJCAI/AAAI, 2011.

Alon Y. Levy and Marie-Christine Rousset.
Combining Horn rules and description logics in CARIN.
*Artificial Intelligence*, 104(1–2):165–209, 1998.

Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz.
OWL 2 Web Ontology Language: Profiles.
W3C Working Draft, World Wide Web Consortium,
http://www.w3.org/TR/2008/WD-owl2-profiles-20081008/, 2008.

# References IX


Magdalena Ortiz, Diego Calvanese, and Thomas Eiter.
Data complexity of query answering in expressive description logics via tableaux.
*Journal of Automated Reasoning*, 41(1):61–98, 2008.

Magdalena Ortiz, Mantas Šimkus, and Thomas Eiter.
Worst-case optimal conjunctive query answering for an expressive description logic without inverses.
In *Proceedings 23rd Conference on Artificial Intelligence (AAAI '08), July 13-17, 2008, Chicago*, pages 504–510. AAAI Press, 2008.

Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus.
Worst-case optimal reasoning for the horn-DL fragments of OWL 1 and 2.
In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczyński, editors, *KR*. AAAI Press, 2010.

Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev, editors.
*Proceedings of the 24th International Workshop on Description Logics (DL 2011), Barcelona, Spain, July 13-16, 2011*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

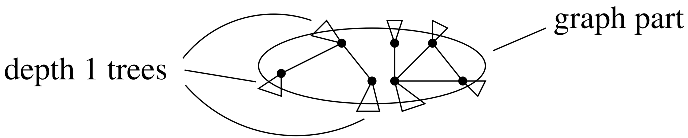
# $\mathcal{SROEL}(\sqcap, \times)$

- $\mathcal{SROEL}(\sqcap, \times)$ is in essence a superset of OWL 2 EL
  Differences:
  - disregards datatypes
  - adding conjunction of roles ($R \sqcap S$), local reflexivity ($Self$), concept production ($C \times D \sqsubseteq T$, $R \sqsubseteq C \times D$)
  - restrictions on role occurrences in a KB (simplicity, range restrictions), but not role regularities
- $\mathcal{SROEL}(\sqcap, \times)$ has polynomial complexity (sat, instance checking)
- [Krötzsch, 2010] describes a proof system for instance checking over a $\mathcal{SROEL}(\sqcap, \times)$ ontology
- This proof system can be naturally encoded in a logic program, viewing axioms $\alpha$ as facts and inference rules $\frac{\alpha_1,\ldots,\alpha_n}{\alpha}$ as rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$
- A universal (schematic) encoding in Datalog is possible

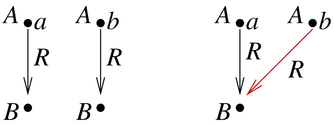
# $\mathcal{SROEL}(\sqcap, \times)$, cont'd

**Key aspects:**

- It is suffcient to generate a small part of a canonical forest-shaped model




- More precisely, only new elements directly connected to some individual, due to existenial axioms $A \sqsubseteq \exists R.B$
- For uniform (ABox independent) encoding, *share new elements*



# Transformation of $\mathcal{SROEL}(\sqcap, \times)$ to Datalog

- $\mathcal{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)

# Transformation of $\mathcal{SROEL}(\sqcap, \times)$ to Datalog

- $\mathcal{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)
  - view the inference rules $\frac{\alpha}{\alpha_1, \ldots, \alpha_n}$ as LP rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$
    E.g., $\frac{C \sqsubseteq D,\ C(a)}{D(a)}$ can be viewed as rule $D(a) \leftarrow \sqsubseteq(C, D), C(a)$

# Transformation of $\mathcal{SROEL}(\sqcap, \times)$ to Datalog

- $\mathcal{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)
  - view the inference rules $\frac{\alpha}{\alpha_1,\ldots,\alpha_n}$ as LP rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$
    E.g., $\frac{C \sqsubseteq D,\ C(a)}{D(a)}$ can be viewed as rule $D(a) \leftarrow \sqsubseteq(C, D), C(a)$
- Use *reification* to obtain a Datalog representation

$$\Phi_{\mathcal{EL}}(\mathcal{O}) = I_{inst}(\mathcal{O}) \cup P_{inst}$$

where $\mathcal{O}_{inst}$ encodes $\mathcal{O}$ and $P_{inst}$ is a fixed set of rules (schemata)

  - names: $C \rightsquigarrow cls(C)$; $R \rightsquigarrow rol(R)$; $a \rightsquigarrow nom(a)$
  - assertions: e.g $C(a) \rightsquigarrow isa(a, C)$; $R(a, b) \rightsquigarrow triple(a, R, b)$
  - axioms: e.g. $A \sqsubseteq C \rightsquigarrow subClass(A, C)$,

# Transformation of $\mathcal{SROEL}(\sqcap,\times)$ to Datalog

- $\mathcal{SROEL}(\sqcap,\times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C,D)$ (infix)
  - view the inference rules $\frac{\alpha}{\alpha_1,\ldots,\alpha_n}$ as LP rules $\alpha \leftarrow \alpha_1,\ldots,\alpha_n$
    E.g., $\frac{C \sqsubseteq D,\ C(a)}{D(a)}$ can be viewed as rule $D(a) \leftarrow \sqsubseteq(C,D), C(a)$
- Use *reification* to obtain a Datalog representation

$$\Phi_{\mathcal{EL}}(\mathcal{O}) = I_{inst}(\mathcal{O}) \cup P_{inst}$$

where $\mathcal{O}_{inst}$ encodes $\mathcal{O}$ and $P_{inst}$ is a fixed set of rules (schemata)

  - names: $C \rightsquigarrow cls(C)$; $R \rightsquigarrow rol(R)$; $a \rightsquigarrow nom(a)$
  - assertions: e.g $C(a) \rightsquigarrow isa(a,C)$; $R(a,b) \rightsquigarrow triple(a,R,b)$
  - axioms: e.g. $A \sqsubseteq C \rightsquigarrow subClass(A,C)$,
- Make reified rules generic using variables

E.g. $isa(a,D) \leftarrow subClass(C,D), isa(a,C)$ gets
$isa(X,Z) \leftarrow subClass(Y,Z), isa(X,Y)$

# Encoding $I_{inst}(\mathcal{O})$

| | | |
|---|---|---|
| $C(a) \leadsto isa(a, C)$ | $R(a, b) \leadsto triple(a, R, b)$ | $a \in N_I \leadsto nom(a)$ |
| $\top \sqsubseteq C \leadsto top(C)$ | $A \sqsubseteq \bot \leadsto bot(A)$ | $A \in N_C \leadsto cls(A)$ |
| $\{a\} \sqsubseteq C \leadsto subClass(a, C)$ | $A \sqsubseteq \{c\} \leadsto subClass(A, c)$ | $R \in N_R \leadsto rol(R)$ |
| $A \sqsubseteq C \leadsto subClass(A, C)$ | $A \sqcap B \sqsubseteq C \leadsto subConj(A, B, C)$ | |
| $\exists R.Self \sqsubseteq C \leadsto subSelf(R, C)$ | $A \sqsubseteq \exists R.Self \leadsto supSelf(A, R)$ | |
| $\exists R.A \sqsubseteq C \leadsto subEx(R, A, C)$ | $A \sqsubseteq \exists R.B \leadsto supEx(A, R, B, e^{A \sqsubseteq \exists R.B})$ | |
| $R \sqsubseteq T \leadsto subRole(R, T)$ | $R \circ S \sqsubseteq T \leadsto subRChain(R, S, T)$ | |
| $R \sqsubseteq C \times D \leadsto supProd(R, C, D)$ | $A \times B \sqsubseteq R \leadsto subProd(A, B, R)$ | |
| $R \sqcap S \sqsubseteq T \leadsto subRConj(R, S, T)$ | | |

- Encode axiom $\alpha \leadsto I_{inst}(\alpha)$
- Encode individual $s \leadsto I_{inst}(s)$
- $I_{inst}(\mathcal{O}) = \{I_{inst}(\alpha) \mid \alpha \in L\} \cup \{I_{inst}(s) \mid s \in N_I \cup N_C \cup N_R\}$

# Encoding $I_{inst}(\mathcal{O})$

$$
\begin{array}{rcl}
C(a) \leadsto isa(a, C) & R(a,b) \leadsto triple(a, R, b) & a \in N_I \leadsto nom(a) \\
\top \sqsubseteq C \leadsto top(C) & A \sqsubseteq \bot \leadsto bot(A) & A \in N_C \leadsto cls(A) \\
\{a\} \sqsubseteq C \leadsto subClass(a, C) & A \sqsubseteq \{c\} \leadsto subClass(A, c) & R \in N_R \leadsto rol(R) \\
A \sqsubseteq C \leadsto subClass(A, C) & A \sqcap B \sqsubseteq C \leadsto subConj(A, B, C) & \\
\exists R.Self \sqsubseteq C \leadsto subSelf(R, C) & A \sqsubseteq \exists R.Self \leadsto supSelf(A, R) & \\
\exists R.A \sqsubseteq C \leadsto subEx(R, A, C) & A \sqsubseteq \exists R.B \leadsto supEx(A, R, B, e^{A \sqsubseteq \exists R.B}) & \\
R \sqsubseteq T \leadsto subRole(R, T) & R \circ S \sqsubseteq T \leadsto subRChain(R, S, T) & \\
R \sqsubseteq C \times D \leadsto supProd(R, C, D) & A \times B \sqsubseteq R \leadsto subProd(A, B, R) & \\
R \sqcap S \sqsubseteq T \leadsto subRConj(R, S, T) & & 
\end{array}
$$

- Encode axiom $\alpha \leadsto I_{inst}(\alpha)$
- Encode individual $s \leadsto I_{inst}(s)$
- $I_{inst}(\mathcal{O}) = \{I_{inst}(\alpha) \mid \alpha \in L\} \cup \{I_{inst}(s) \mid s \in N_I \cup N_C \cup N_R\}$
  - use constants $e^{A \sqsubseteq \exists R.B}$ for elements enforced by existential axioms $A \sqsubseteq \exists R.B$
  - encode in $supEx(A, R, B, e^{A \sqsubseteq \exists R.B})$ the pattern $\overset{A}{\circ} \xrightarrow{R} \overset{B}{\circ}$
  - "share" $e^{A \sqsubseteq \exists R.B}$ for individuals $a, b$ belonging to $A$

# Example

Consider

$$\mathcal{O} = \{A(a),\ A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D\}$$

$\mathcal{O}$ is translated to

$$I_{inst}(\mathcal{O}) = \left\{ \begin{array}{l} isa(a, A),\ supEx(A, R, B, e^{A \sqsubseteq \exists R.B}),\ subClass(B, C), \\ subEx(R, C, D),\ nom(a),\ cls(A),\ cls(B),\ cls(C),\ cls(D),\ rol(R) \end{array} \right\}.$$

# Inference Rules (Datalog Encoding)

Datalog program $P_{inst}$: instance inference

$$
\begin{aligned}
isa(X,Z) &\leftarrow top(Z), isa(X,Z') \\
isa(X,Y) &\leftarrow bot(Z), isa(U,Z), isa(X,Z'), cls(Y) \\
isa(X,Z) &\leftarrow subClass(Y,Z), isa(X,Y) \\
isa(X,Z) &\leftarrow subConj(Y_1,Y_2,Z), isa(X,Y_1), isa(X,Y_2) \\
isa(X,Z) &\leftarrow subEx(V,Y,Z), triple(X,V,X'), isa(X',Y) \\
isa(X,Z) &\leftarrow subEx(V,Y,Z), self(X,V), isa(X,Y) \\
isa(X',Z) &\leftarrow supEx(Y,V,Z,X'), isa(X,Y) \\
isa(X,Z) &\leftarrow subSelf(V,Z), self(X,V) \\
isa(X,Z_1) &\leftarrow supProd(V,Z_1,Z_2), triple(X,V,X') \\
isa(X,Z_1) &\leftarrow supProd(V,Z_1,Z_2), self(X,V) \\
isa(X',Z_2) &\leftarrow supProd(V,Z_1,Z_2), triple(X,V,X') \\
isa(X,Z_2) &\leftarrow supProd(V,Z_1,Z_2), self(X,V) \\
isa(X,X) &\leftarrow nom(X) \\
isa(Y,Z) &\leftarrow isa(X,Y), nom(Y), isa(X,Z) \\
isa(X,Z) &\leftarrow isa(X,Y), nom(Y), isa(Y,Z)
\end{aligned}
$$

# Inference Rules (Datalog Encoding), cont'd

## Datalog program $P_{inst}$: role and *Self* inference

$$triple(X, W, X') \leftarrow subRole(V, W), triple(X, V, X')$$
$$triple(X, W, X'') \leftarrow subRChain(U, V, W), triple(X, U, X'), triple(X', V, X'')$$
$$triple(X, W, X') \leftarrow subRChain(U, V, W), self(X, U), triple(X, V, X')$$
$$triple(X, W, X') \leftarrow subRChain(U, V, W), triple(X, U, X'), self(X', V)$$
$$triple(X, W, X) \leftarrow subRChain(U, V, W), self(X, U), self(X, V)$$
$$triple(X, W, X') \leftarrow subRConj(V_1, V_2, W), triple(X, V_1, X'), triple(X, V_2, X')$$
$$triple(Z, U, Y) \leftarrow isa(X, Y), nom(Y), triple(Z, U, X)$$
$$triple(X, V, X') \leftarrow supEx(Y, V, Z, X'), isa(X, Y)$$
$$triple(X, W, X') \leftarrow subProd(Y_1, Y_2, W), isa(X, Y_1), isa(X', Y_2)$$

$$self(X, V) \leftarrow nom(X), triple(X, V, X)$$
$$self(X, W) \leftarrow subRole(V, W), self(X, V)$$
$$self(X, W) \leftarrow subRConj(V_1, V_2, W), self(X, V_1), self(X, V_2)$$
$$self(X, W) \leftarrow subProd(Y_1, Y_2, W), isa(X, Y_1), isa(X, Y_2)$$
$$self(X, V) \leftarrow supSelf(Y, V), isa(X, Y)$$

# Instance Queries

- $\Phi_{\mathcal{EL}}(\mathcal{O}) = P_{inst} \cup I_{inst}(\mathcal{O})$ can be used to decide satisfiability
- $\Phi_{\mathcal{EL}}(\mathcal{O})$ can be used to answer instance queries

## Theorem

*For every $\mathcal{SROEL}(\sqcap, \times)$ ontology $\mathcal{O}$ and $a, b \in N_I$*

(i) $\mathcal{O} \models C(a)$ *iff* $\Phi_{\mathcal{EL}}(\mathcal{O}) \models isa(a, C)$

(ii) $\mathcal{O} \models R(a, b)$ *iff* $\Phi_{\mathcal{EL}}(\mathcal{O}) \models triple(a, R, b)$.

# Example, cont'd

Consider $\mathcal{O} = \{\ A(a),\ A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D\ \}$

$$I_{inst}(\mathcal{O}) = \left\{ \begin{array}{l} isa(a,A),\ supEx(A,R,B,e^{A \sqsubseteq \exists R.B}),\ subClass(B,C), \\ subEx(R,C,D),\ nom(a),\ cls(A),\ cls(B),\ cls(C),\ cls(D),\ rol(R) \end{array} \right\} .$$

- We have $\mathcal{O} \models D(a)$
- From $\Phi_{\mathcal{EL}}(\mathcal{O})$ we can derive $I_{inst}(D(a)) = isa(a,D)$:

# Example, cont'd

Consider $\mathcal{O} = \{ A(a),\ A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D \}$

$$I_{inst}(\mathcal{O}) = \left\{ \begin{array}{l} isa(a, A),\ supEx(A, R, B, e^{A \sqsubseteq \exists R.B}),\ subClass(B, C), \\ subEx(R, C, D),\ nom(a),\ cls(A),\ cls(B),\ cls(C),\ cls(D),\ rol(R) \end{array} \right\} .$$

- We have $\mathcal{O} \models D(a)$
- From $\Phi_{\mathcal{EL}}(\mathcal{O})$ we can derive $I_{inst}(D(a)) = isa(a, D)$:
  - apply $isa(X', Z) \leftarrow supEx(Y, V, Z, X'), isa(X, Y)$:
    $isa(e^{A \sqsubseteq \exists R.B}, B)$

# Example, cont'd

Consider $\mathcal{O} = \{\ A(a),\ A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D\ \}$

$$I_{inst}(\mathcal{O}) = \left\{ \begin{array}{l} isa(a, A),\ supEx(A, R, B, e^{A \sqsubseteq \exists R.B}),\ subClass(B, C), \\ subEx(R, C, D),\ nom(a),\ cls(A),\ cls(B),\ cls(C),\ cls(D),\ rol(R) \end{array} \right\} .$$

- We have $\mathcal{O} \models D(a)$
- From $\Phi_{\mathcal{EL}}(\mathcal{O})$ we can derive $I_{inst}(D(a)) = isa(a, D)$:
  - apply $isa(X', Z) \leftarrow supEx(Y, V, Z, X'), isa(X, Y)$:
    $isa(e^{A \sqsubseteq \exists R.B}, B)$
  - apply $isa(X, Z) \leftarrow subClass(Y, Z), isa(X, Y)$:
    $isa(e^{A \sqsubseteq \exists R.B}, C)$

# Example, cont'd

Consider $\mathcal{O} = \{\ A(a),\ A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D\ \}$

$$I_{inst}(\mathcal{O}) = \left\{ \begin{array}{l} isa(a, A),\ supEx(A, R, B, e^{A \sqsubseteq \exists R.B}),\ subClass(B, C), \\ subEx(R, C, D),\ nom(a),\ cls(A),\ cls(B),\ cls(C),\ cls(D),\ rol(R) \end{array} \right\}.$$

- We have $\mathcal{O} \models D(a)$
- From $\Phi_{\mathcal{EL}}(\mathcal{O})$ we can derive $I_{inst}(D(a)) = isa(a, D)$:
  - apply $isa(X', Z) \leftarrow supEx(Y, V, Z, X'), isa(X, Y)$:
    $isa(e^{A \sqsubseteq \exists R.B}, B)$
  - apply $isa(X, Z) \leftarrow subClass(Y, Z), isa(X, Y)$:
    $isa(e^{A \sqsubseteq \exists R.B}, C)$
  - apply $triple(X, V, X') \leftarrow supEx(Y, V, Z, X'), isa(X, Y)$
    $triple(a, R, e^{A \sqsubseteq \exists R.B})$

# Example, cont'd

Consider $\mathcal{O} = \{\ A(a),\ A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D\ \}$

$$I_{inst}(\mathcal{O}) = \left\{ \begin{array}{l} isa(a,A),\ supEx(A,R,B,e^{A\sqsubseteq\exists R.B}),\ subClass(B,C), \\ subEx(R,C,D),\ nom(a),\ cls(A),\ cls(B),\ cls(C),\ cls(D),\ rol(R) \end{array} \right\} .$$

- We have $\mathcal{O} \models D(a)$
- From $\Phi_{\mathcal{EL}}(\mathcal{O})$ we can derive $I_{inst}(D(a)) = isa(a,D)$:
  - apply $isa(X',Z) \leftarrow supEx(Y,V,Z,X'), isa(X,Y)$:
    $isa(e^{A\sqsubseteq\exists R.B}, B)$
  - apply $isa(X,Z) \leftarrow subClass(Y,Z), isa(X,Y)$:
    $isa(e^{A\sqsubseteq\exists R.B}, C)$
  - apply $triple(X,V,X') \leftarrow supEx(Y,V,Z,X'), isa(X,Y)$
    $triple(a,R,e^{A\sqsubseteq\exists R.B})$
  - apply $isa(X,Z) \leftarrow subEx(V,Y,Z), triple(X,V,X'), isa(X',Y)$
    $isa(a,D)$

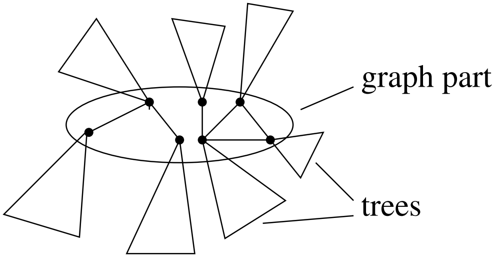
# Query Answering in Horn-$\mathcal{SHIQ}$

- $\mathcal{SHIQ}$ is an expressive DL (cf. OWL Lite)
  - transitive roles ($\mathcal{S}$), role hierarchies ($\mathcal{H}$), inverses ($\mathcal{I}$)
  - qualified number restrictions ($\mathcal{Q}$)
- Horn fragment (Horn-$\mathcal{SHIQ}$): eliminate positive disjunction $\sqcup$ on right hand side
- Horn-$\mathcal{SHIQ}$ has useful features missing in $\mathcal{EL}$ and $\mathcal{DL}$-*Lite*

$$\mathit{trans}(\text{isLocatedIn}) \quad \text{country} \sqsubseteq \forall \text{hasCapital.city} \quad \text{country} \sqsubseteq \leqslant 1\ \text{isLocatedIn}^{-}.\text{capital}$$

- CQ Answering for Horn-$\mathcal{SHIQ}$ is tractable in data complexity (PTIME-complete)
- The combined complexity of CQs is not higher than for satisfiability testing (EXPTIME-complete)
- Its features make CQ answering for Horn-$\mathcal{SHIQ}$ significantly more complex than for $\mathcal{EL}$

# Issues




- Match the query $Q$ *partially between graph part and trees* (⇒ tree-shaped query parts)
- Inverse roles allow to move up and down the tree (⇒ connect different trees)
- Transitive roles: how far to go for a match in a tree?

# Datalog Query Answering for Horn-$\mathcal{SHIQ}$

Ortiz *et al.* [2010]: CQ rewriting to Datalog (big predicate arities; impractical)

E_ *et al.* [2012a,2012b]: better rewriting

Three components:

- **UOC rewriting:** CQ $Q \rightsquigarrow$ UCQ rew$_{\mathcal{T}}(Q)$ (depends on the TBox $\mathcal{T}$)
- **TBox saturation:** enrich $\mathcal{T}$ with relevant axioms for rewriting ($\Xi(\mathcal{T})$)
- **ABox completion:** $\mathcal{T}$ is rewritten into a set of Datalog rules cr$(\mathcal{T})$ to "complete" the graph part

Answering $Q$ over $(\mathcal{T}, \mathcal{A})$ amounts to evaluating the Datalog program

$$\mathcal{A} \cup \text{cr}(\mathcal{T}) \cup \text{rew}_{\mathcal{T}}(q)$$

- One can evaluate rew$_{\mathcal{T}}(Q)$ over the completion of $\mathcal{A}$ (with no additional unnamed objects)
- rew$_{\mathcal{T}}(q)$ can be exponential, but has manageable size for real queries and ontologies

# The rewriting algorithm

## Main idea:

- Eliminate query variables that can be matched at unnamed objects
  - Query matches have tree-shaped parts
  - We clip off the variables $x$ that can be leaves
  - Replace them by constraints $D(y)$ on their parent variables $y$
  - The added atoms $D(y)$ ensure the existence of a match for $x$
- In the resulting queries all variables are matched to named objects

# The rewriting algorithm

## Main idea:

- Eliminate query variables that can be matched at unnamed objects
  - Query matches have tree-shaped parts
  - We clip off the variables $x$ that can be leaves
  - Replace them by constraints $D(y)$ on their parent variables $y$
  - The added atoms $D(y)$ ensure the existence of a match for $x$
- In the resulting queries all variables are matched to named objects

A Horn-$\mathcal{SHIQ}$ TBox $\mathcal{T}$ is in *normal form*, if GCIs in $\mathcal{T}$ have the forms:

(F1) $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$, (F3) $A_1 \sqsubseteq \forall r.B$,

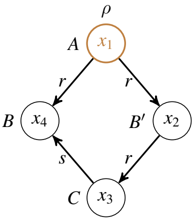
(F2) $A_1 \sqsubseteq \exists r.B$, (F4) $A_1 \sqsubseteq \leqslant 1\, r.B$,

where $A_1, \ldots, A_n, B$ are concept names and $r$ is a role.

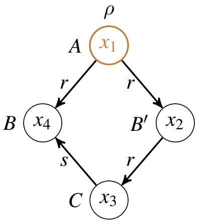
Normalize $\mathcal{T}$ (efficiently doable, [Kazakov, 2009], [Krötzsch *et al.*, 2007])

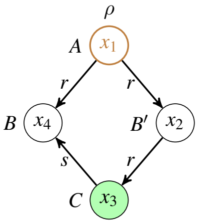
# One Step of Query Rewriting

$$q(x_1) \leftarrow r(x_1,x_2), r(x_1,x_4), r(x_2,x_3), s(x_3,x_4), A(x_1), B(x_4), B'(x_2), C(x_3)$$



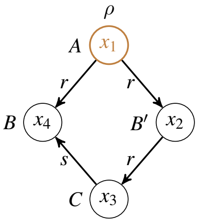
# One Step of Query Rewriting

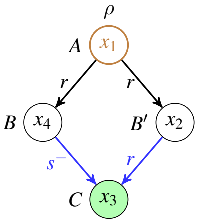



1. Select the non-distinguished variable $x_3$

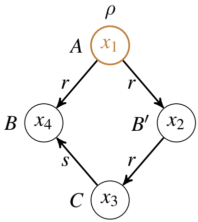
# One Step of Query Rewriting

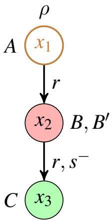



1. Select the non-distinguished variable $x_3$
2. Ensure that $x_3$ has only incoming edges
   ➤replace $r(x, y)$ by $r^{-}(y, x)$ as needed

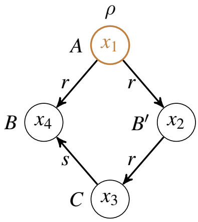
# One Step of Query Rewriting



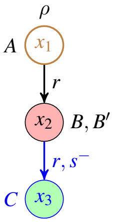




1. Select the non-distinguished variable $x_3$
2. Ensure that $x_3$ has only incoming edges
   ➤replace $r(x, y)$ by $r^{-}(y, x)$ as needed
3. Merge the predecessors
   ➤if $x_3$ is a leaf of a tree, they must be mapped together

# One Step of Query Rewriting

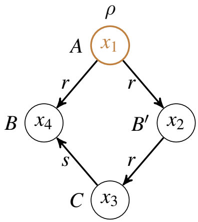



$\mathcal{T} \models D \sqsubseteq \exists(r \sqcap s^{-}).C$

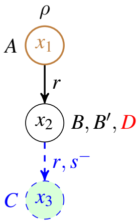



1. Select the non-distinguished variable $x_3$
2. Ensure that $x_3$ has only incoming edges
   ➤replace $r(x, y)$ by $r^{-}(y, x)$ as needed
3. Merge the predecessors
   ➤if $x_3$ is a leaf of a tree, they must be mapped together
4. Find an axiom that enforces an $(r \sqcap s^{-})$-child that is $C$
   ➤fail if $\mathcal{T}$ does not imply such an axiom

# One Step of Query Rewriting

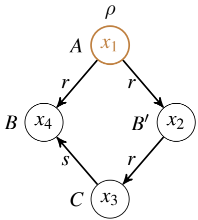



$\mathcal{T} \models D \sqsubseteq \exists(r \sqcap s^-).C$

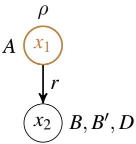



1. Select the non-distinguished variable $x_3$
2. Ensure that $x_3$ has only incoming edges
   ➤replace $r(x,y)$ by $r^-(y,x)$ as needed
3. Merge the predecessors
   ➤if $x_3$ is a leaf of a tree, they must be mapped together
4. Find an axiom that enforces an $(r \sqcap s^-)$-child that is $C$
   ➤fail if $\mathcal{T}$ does not imply such an axiom
5. Drop $x_3$ and add $D(x_2)$

# One Step of Query Rewriting



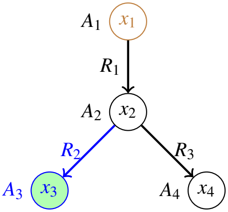

$$\mathcal{T} \models D \sqsubseteq \exists(r \sqcap s^-).C$$




1. Select the non-distinguished variable $x_3$
2. Ensure that $x_3$ has only incoming edges
   - replace $r(x, y)$ by $r^-(y, x)$ as needed
3. Merge the predecessors
   - if $x_3$ is a leaf of a tree, they must be mapped together
4. Find an axiom that enforces an $(r \sqcap s^-)$-child that is $C$
   - fail if $\mathcal{T}$ does not imply such an axiom
5. Drop $x_3$ and add $D(x_2)$

# Another Step of Query Rewriting

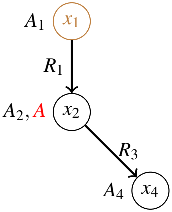
The query




using the axiom

$A \sqsubseteq \exists R_2.A_3$

is rewritten to



# Transitive Roles

To handle transitive roles in the query $Q$:

- introduce a new variable between eliminated variable and some of its predecessors
- eliminate sets of variables
  variables connected in the query may be mapped to same element (reach the element on paths of different length)

Note:

- the number of variables in $Q$ does not increase (reuse of variables possible)
- only an exponential number of queries are possible
- the labels on edges of the query graph increase

Thus, rewriting terminates

# TBox Saturation

- A set $\Xi(\mathcal{T})$ of relevant axioms is computed in advance
  - Tailored resolution calculus for Horn-$\mathcal{ALCHIQ}^{\sqcap}$
  - Adaptation of existing *consequence driven* procedures for satisfiability [Kazakov, 2009], [Ortiz *et al.*, 2010]

## Example Rules (all: Appendix)

$$\frac{M \sqsubseteq \exists S.(N \sqcap N') \quad N \sqsubseteq A}{M \sqsubseteq \exists S.(N \sqcap N' \sqcap A)} \; \mathbf{R}^{c}_{\sqsubseteq}$$

$$\frac{M \sqsubseteq \exists (S \sqcap inv(r)).(N \sqcap A) \quad A \sqsubseteq \forall r.B}{M \sqsubseteq B} \; \mathbf{R}^{-}_{\forall}$$

- The rewriting step simply searches for an axiom in $\Xi(\mathcal{T})$

# ABox Completion Rules

The completion rules cr($\mathcal{T}$) are straightforward:

$B(y) \leftarrow A(x), r(x,y)$ for each $A \sqsubseteq \forall r.B \in \mathcal{T}$

$B(x) \leftarrow A_1(x), \ldots, A_n(x)$ for all $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \Xi(\mathcal{T})$

$r(x,y) \leftarrow r_1(x,y), \ldots, r_n(x,y)$ for all $r_1 \sqcap \ldots \sqcap r_n \sqsubseteq r \in \mathcal{T}$

$\bot(x) \leftarrow A(x), r(x,y_1), r(x,y_2), B(y_1), B(y_2), y_1 \neq y_2$
for each $A \sqsubseteq\ \leqslant 1\, r.B \in \mathcal{T}$

$\Gamma \leftarrow A(x), A_1(x), \ldots, A_n(x), r(x,y), B(y)$
for all $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \exists(r_1 \sqcap \ldots \sqcap r_m).B_1 \sqcap \ldots \sqcap B_k$ and $A \sqsubseteq\ \leqslant 1\, r.B$ of $\Xi(\mathcal{T})$ such that $r = r_i$ and $B = B_j$ for some $i, j$ with $\Gamma \in \{B_1(y), \ldots, B_k(y), r_1(x,y), \ldots, r_k(x,y)\}$

# Query Answering Algorithm

**Algorithm Horn-$\mathcal{SHIQ}$-CQ:**

---

**Input**: normal Horn-$\mathcal{SHIQ}$ KB $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, conjunctive query $Q$
**Output**: query answers
$\Xi(\mathcal{T}) \leftarrow \texttt{Saturate}(\mathcal{T})$;
$\text{rew}_{\mathcal{T}}(Q) \leftarrow \texttt{Rewrite}(Q, \Xi(\mathcal{T}))$;
$\text{cr}(\mathcal{T}) \leftarrow \texttt{CompletionRules}(\mathcal{T})$;
$P \leftarrow \mathcal{A} \cup \text{cr}(\mathcal{T}) \cup \text{rew}_{\mathcal{T}}(Q)$;
$ans \leftarrow \{\vec{u} \mid q(\vec{u}) \in \texttt{Datalog-eval}(P)\}$; $\triangleright$ `call Datalog reasoner`

---

## Theorem

*For satisfiable Horn-$\mathcal{SHIQ}$ $\mathcal{O}$ in normal form and CQ $Q$, the algorithm Horn-$\mathcal{SHIQ}$-CQoutputs $ans(Q, \mathcal{O})$. It runs (properly implemented) polynomial in data complexity and exponential in combined complexity.*

# Closed-world Assumption

- Reiter's well-known closed-world assumption (CWA) is acknowledged as an important reasoning principle for inferring negative information from a first-order theory $T$.
- For a ground atom $p(c)$, conclude $\neg p(c)$ if $T \not\models p(c)$. Any such atom $p(c)$ is also called free for negation.
- The CWA of $T$ , denoted $CWA(T)$, is then the extension of $T$ with all literals $\neg p(c)$ where $p(c)$ is free for negation.
- Using $\mathrm{dl}$-Programs, the CWA may be intuitively expressed on top of an external DL knowledge base, which can be queried through suitable dl-atoms.