Inconsistencies in Hybrid Knowledge Bases
PhD Defense

Daria Stepanova
Supervisors: Prof. Thomas Eiter and Dr. Michael Fink

Doctoral School “Mathematical Logic in Computer Science”,
Institute of Information Systems,
Vienna University of Technology

April 20, 2015
Motivation

Hybrid Knowledge Bases

LPs/Rules (Logic Programs)
- Closed-World Assumption
- Nonmonotonic
- Defaults and exceptions
- ...

DLs (DL Ontologies)
- Open-World Assumption
- Monotonic
- Conceptual reasoning
- ...

Conclusion
Hybrid Knowledge Bases

Approaches for combining rules and ontologies

- Full integration
  - MKNF KBs [Motik and Rosati, 2010]
  - FO-Autoepistemic Logic [de Bruijn et al., 2011]
  - Quantified Equilibrium Logic [de Bruijn et al., 2007]

- Tight integration
  - Carin [Levy and Rousset, 1998]
  - DL-safe rules [Motik et al., 2005]
  - R-hybrid KBs [Rosati, 2005]
  - $\mathcal{DL}+$LOG [Rosati, 2006]

- Strict semantic separation
  - DL-programs [Eiter et al., 2008]
  - Defeasible Logic+DL [Wang et al., 2004]
DL-programs

- **DL-programs**: Rules + Ontology (loose coupling combination)

- **Applications**:
  - Semantically enriched route planning
  - Assignment problems involving preferences
  - Medical systems
  - Reasoning on the web . . .

- **Problem**: Inconsistencies often arise as a result of combination
**DL-programs**

- **DL-programs**: Rules + Ontology (loose coupling combination)

- **Applications**:
  - Semantically enriched route planning
  - Assignment problems involving preferences
  - Medical systems
  - Reasoning on the web ... 

- **Problem**: inconsistencies often arise as a result of combination
Inconsistency in DL-programs

**Problem:** inconsistency in a DL-program

**Question:** how to deal with it?

Many possibilities..

```
[Diagram]
Sources of inconsistencies in DL-programs

LP

DL

LP+DL

Inconsistency handling

Repair

LP

DL-atoms

DL

Paraconsistent reasoning

LP

DL
```
Overview

Hybrid Knowledge Bases

Problem Statement

Repair Semantics

Computation

Implementation and Evaluation

Conclusion
Description Logic Ontologies

• 1950’s-1960’s: First Order Logic (FOL) for KR (e.g. [McCarthy, 1959])

$$\forall X (Female(X) \land \exists Y (hasChild(X, Y)) \rightarrow Mother(X))$$
Description Logic Ontologies

- 1950’s-1960’s: First Order Logic (FOL) for KR (undecidable)
  (e.g. [McCarthy, 1959])

\[ \forall X (\text{Female}(X) \land \exists Y (\text{hasChild}(X, Y)) \rightarrow \text{Mother}(X)) \]
Description Logic Ontologies

- 1950’s-1960’s: First Order Logic (FOL) for KR (undecidable) (e.g. [McCarthy, 1959])

- 1970’s: Network-shaped structures for KR (e.g. semantic networks [Quillan, 1967], frames [Minsky, 1985])

![Diagram of Description Logic Ontologies]

- Man is-a Parent
- Father is-a Man
- John is-a Father
- Male is-a Parent
- Parent has-Child is-a Human
- Male is-a Human
- Male is-a Man
- John is-married with Mary
Description Logic Ontologies

- 1950’s-1960’s: First Order Logic (FOL) for KR (undecidable) (e.g. [McCarthy, 1959])

- 1970’s: Network-shaped structures for KR (no formal semantics) (e.g. semantic networks [Quillan, 1967], frames [Minsky, 1985])
Description Logic Ontologies

- 1950’s-1960’s: First Order Logic (FOL) for KR (undecidable) (e.g. [McCarthy, 1959])

- 1970’s: Network-shaped structures for KR (no formal semantics) (e.g. semantic networks [Quillan, 1967], frames [Minsky, 1985])

- 1979: Encoding of frames into FOL [Hayes, 1979]
Description Logic Ontologies

- **1950’s-1960’s**: First Order Logic (FOL) for KR (undecidable) (e.g. [McCarthy, 1959])

- **1970’s**: Network-shaped structures for KR (no formal semantics) (e.g. semantic networks [Quillian, 1967], frames [Minsky, 1985])

- **1979**: Encoding of frames into FOL [Hayes, 1979]

- **1980’s**: Description Logics (DL) for KR
  - Decidable fragments of FOL
  - Theories encoded in DLs are called ontologies
  - Many DLs with different expressiveness and computational features
Description Logic Ontologies

○ 1950’s-1960’s: First Order Logic (FOL) for KR (undecidable) (e.g. [McCarthy, 1959])

○ 1970’s: Network-shaped structures for KR (no formal semantics) (e.g. semantic networks [Quillan, 1967], frames [Minsky, 1985])

○ 1979: Encoding of frames into FOL [Hayes, 1979]

○ 1980’s: Description Logics (DL) for KR
  - Decidable fragments of FOL
  - Theories encoded in DLs are called ontologies
  - Many DLs with different expressiveness and computational features

○ In this work: lightweight DLs ($DL\textrm{-}Lite_A$, $\mathcal{EL}$)
Description Logic $DL\text{-}Lite_A$

- **Concepts** model sets of objects and **roles** model binary relations
  - $Child$, $hasParent$
Description Logic $DL\text{-}Lite_A$

- **Concepts** model sets of objects and **roles** model binary relations

- More complex concepts and roles can be constructed:

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>negated concept</td>
<td>$\neg C$</td>
<td>$\neg Male$</td>
</tr>
<tr>
<td>exist. on roles</td>
<td>$\exists R$</td>
<td>$\exists hasChild$</td>
</tr>
<tr>
<td>negated roles</td>
<td>$\neg R$</td>
<td>$\neg hasSibling$</td>
</tr>
<tr>
<td>role inverses</td>
<td>$R^-$</td>
<td>$hasParent^-$</td>
</tr>
</tbody>
</table>
Description Logic $DL\text{-}\text{Lite}_A$

- **Concepts** model sets of objects and **roles** model binary relations.

- More complex concepts and roles can be constructed:

  \[
  C \rightarrow A \mid \exists R \quad B \rightarrow C \mid \neg C \\
  R \rightarrow U \mid U^- \quad S \rightarrow R \mid \neg R
  \]

- A $DL\text{-}\text{Lite}_A$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of:

  - **TBox** $\mathcal{T}$ specifying constraints at the conceptual level.
    \[
    C \sqsubseteq B \quad R \sqsubseteq S \quad (\text{funct } R)
    \]
  - **ABox** $\mathcal{A}$ specifying facts that hold in the domain.
    \[
    A(b) \quad P(a, b)
    \]
Description Logic $DL$-$Lite_\mathcal{A}$

- **Concepts** model sets of objects and **roles** model binary relations

- More complex concepts and roles can be constructed:

  \[
  \begin{align*}
  C \rightarrow A \mid & \exists R \quad B \rightarrow C \mid \neg C \\
  R \rightarrow U \mid & U^- \quad S \rightarrow R \mid \neg R
  \end{align*}
  \]

- A $DL$-$Lite_\mathcal{A}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of:
  - **TBox** $\mathcal{T}$ specifying constraints at the conceptual level.
    \[
    C \sqsubseteq B \quad R \sqsubseteq S \quad (\text{funct } R)
    \]
  - **ABox** $\mathcal{A}$ specifying facts that hold in the domain.
    \[
    A(b) \quad P(a, b)
    \]

Ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ in $DL$-$Lite_\mathcal{A}$

\[
\begin{align*}
\mathcal{T} = \{ & \text{Child} \sqsubseteq \exists \text{hasParent} \quad \text{Female} \sqsubseteq \neg \text{Male} \} \\
\mathcal{A} = \{ & \text{hasParent}(john, pat) \quad \text{Male}(john) \}
\end{align*}
\]
Description Logic $\mathcal{EL}$

Ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ in $\mathcal{EL}$

$\mathcal{T} = \{ \text{Aunt} \equiv \text{Female} \sqcap \exists \text{hasSibling}(\exists \text{hasChild}.\text{Human}) \}$

$\mathcal{A} = \{ \text{Female(ann) hasSibling(ann, pat)} \text{, Human(john) hasChild(pat, john)} \}$

- $\mathcal{EL}$-concepts:

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conjunction</td>
<td>$A \sqcap B$</td>
<td>$\text{Female} \sqcap \text{Child}$</td>
</tr>
<tr>
<td>Exist. restr.</td>
<td>$\exists R.A$</td>
<td>$\exists \text{hasSibling}.\text{Male}$</td>
</tr>
</tbody>
</table>

- TBox axioms$^1$:

$C \sqsubseteq D \quad C \equiv D$

$^1$ $C$ and $D$ are arbitrarily complex concepts constructed using $\exists$ and $\sqcap$
**DL-Lite\(\mathcal{A}\)** and \(\mathcal{EL}\): FOL Formalization

*Child \(\sqsubseteq \exists \text{hasParent}\) is equiv. to \(\forall x (\text{Child}(x) \rightarrow \exists y (\text{hasParent}(x, y)))\)*

<table>
<thead>
<tr>
<th>Syntax</th>
<th>FOL formalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_1 \sqsubseteq A_2)</td>
<td>(\forall x (A_1(x) \rightarrow A_2(x)))</td>
</tr>
<tr>
<td>(R_1 \sqsubseteq R_2)</td>
<td>(\forall x, y (R_1(x, y) \rightarrow R_2(x, y)))</td>
</tr>
<tr>
<td>(A_1 \sqsubseteq \neg A_2)</td>
<td>(\forall x (A_1(x) \rightarrow \neg A_2(x)))</td>
</tr>
<tr>
<td>(R_1 \sqsubseteq \neg R_2)</td>
<td>(\forall x, y (R_1(x, y) \rightarrow \neg R_2(x, y)))</td>
</tr>
<tr>
<td>(\exists R \sqsubseteq A)</td>
<td>(\forall x (\exists y (R(x, y)) \rightarrow A(x)))</td>
</tr>
<tr>
<td>(\exists R^- \sqsubseteq A)</td>
<td>(\forall x (\exists y (R(y, x)) \rightarrow A(x)))</td>
</tr>
<tr>
<td>(A \sqsubseteq \exists R)</td>
<td>(\forall x (A(x) \rightarrow \exists y (R(x, y))))</td>
</tr>
<tr>
<td>(\text{funct}(R))</td>
<td>(\forall x, y, y' (R(x, y) \land R(x, y') \rightarrow y = y'))</td>
</tr>
<tr>
<td>(A_1 \sqcap A_2 \sqsubseteq A_3)</td>
<td>(\forall x A_1(x) \land A_2(x) \rightarrow A_3(x))</td>
</tr>
<tr>
<td>(\exists R.A_1 \sqsubseteq A_2)</td>
<td>(\forall x (\exists y (R(x, y) \land A_1(y)) \rightarrow A_2(x))</td>
</tr>
<tr>
<td>(A_1 \sqsubseteq \exists R.A_2)</td>
<td>(\forall x (A(x) \rightarrow \exists y (R(x, y) \land A_2(y))))</td>
</tr>
</tbody>
</table>

\(\ldots\)
Nonmonotonic Logic Programs

- **DLs** are powerful for KR but not well-suited for modelling *human-like* reasoning (e.g. exceptions) due to *monotonicity*
Nonmonotonic Logic Programs

- DLs are powerful for KR but not well-suited for modelling human-like reasoning (e.g. exceptions) due to monotonicity

\[ \text{Human} \sqsubseteq \text{HeartOnLeft} \]
\[ \text{Human}(\text{john}) \]
Nonmonotonic Logic Programs

- DLs are powerful for KR but not well-suited for modelling human-like reasoning (e.g. exceptions) due to monotonicity

\[
\text{Human} \sqsubseteq \text{HeartOnLeft} \\
\text{Human} (\text{john}) \\
\neg \text{HeartOnLeft} (\text{john})
\]
Nonmonotonic Logic Programs

- **DLs** are powerful for KR but not well-suited for modelling human-like reasoning (e.g. exceptions) due to **monotonicity**

- 1980’s: **Nonmonotonic logics** for KR (e.g. circumscription, default logic, auto-epistemic logic)

- 1970’s: **Logic programming** (e.g. Prolog)

- **Nonmonotonic logic programming** under **answer set semantics (ASP)** [Gelfond and Lifschitz, 1988]
Nonmonotonic Logic Programs

Definition

A nonmonotonic logic program $\mathcal{P}$ is a set of rules of the form:

$$a_1 \lor \ldots \lor a_k \leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n.$$  

- $a_i$’s and $b_j$’s are first-order atoms and
- $\text{not}$ is a negation as failure (default negation, weak negation)

Example

$$\text{female}(Y) \lor \text{female}(Z) \leftarrow \text{not adopted}(X), \text{hasparent}(X,Y) \,
\text{hasparent}(X,Z), Y \neq Z$$
Answer Set Semantics

\[ \mathcal{P} = \left\{ \begin{array}{c} \text{hasparent}(\text{john}, \text{pat}); \hspace{1cm} \text{hasparent}(\text{john}, \text{alex}); \\ \text{female}(\text{pat}) \lor \text{female}(\text{alex}) \leftarrow \text{not adopted}(\text{john}), \\ \text{hasparent}(\text{john}, \text{pat}), \\ \text{hasparent}(\text{john}, \text{alex}) \end{array} \right\} \]

- **Semantics**: given for ground programs (programs without variables)

- **Interpretation**: consistent set \( I \) of ground atoms over Herbrand Base of \( \mathcal{P} \)
  \[ I_1 = \{ \text{hasparent}(\text{john}, \text{pat}), \text{hasparent}(\text{john}, \text{alex}), \text{female}(\text{alex}) \} \]

- **Satisfaction relation**: \( I \models a \) iff \( a \in I \)
  \[ I_1 \models \text{hasparent}(\text{john}, \text{pat}); I_1 \not\models \text{adopted}(\text{john}) \]

- **Model**: \( I \) is a model of \( \mathcal{P} \) if, for every \( r \) in \( \mathcal{P} \), \( I \models H(r) \), whenever \( I \models B(r) \)
  \( I_1 \) is a model of \( \mathcal{P} \)

- **Answer set (stable model)**: \( I \) is an answer set of \( \mathcal{P} \) \((I \in \text{AS}(\mathcal{P}))\) if it is a \( \subseteq \)-minimal model that allows founded model reconstruction using rules
  \( I_1 \in \text{AS}(\mathcal{P}) \)
Answer Set Semantics

$$\mathcal{P} = \left\{ \begin{array}{c}
\text{hasparent}(john, pat); \quad \text{hasparent}(john, alex);
\text{female}(pat) \lor \text{female}(alex) \leftarrow \text{not adopted}(john),
\text{hasparent}(john, pat),
\text{hasparent}(john, alex) \end{array} \right\}$$

- $$I_1 = \{ \text{hasparent}(john, pat), \text{hasparent}(john, alex), \text{female}(alex) \}$$
- $$I_2 = \{ \text{hasparent}(john, pat), \text{hasparent}(john, alex), \text{female}(pat) \}$$
- $$I_1, I_2 \in AS(\mathcal{P})$$
Answer Set Semantics

$$\mathcal{P} = \{ \text{hasparent}(\text{john}, \text{pat}); \text{hasparent}(\text{john}, \text{alex}); \text{adopted}(\text{john}); \text{female}(\text{pat}) \lor \text{female}(\text{alex}) \leftarrow \text{not adopted}(\text{john}), \text{hasparent}(\text{john}, \text{pat}), \text{hasparent}(\text{john}, \text{alex}) \}$$

- $$l_3 = \{ \text{hasparent}(\text{john}, \text{pat}), \text{hasparent}(\text{john}, \text{alex}), \text{adopted}(\text{john}) \}$$
  $$l_3 \in \text{AS}(\mathcal{P})$$

- **adopted(john)** is added, **female(alex)/female(pat)** are no longer derived
  Nonmonotonicity!
DL Ontologies vs Logic Programs

- \( \neg \) in DLs is different from \textit{not} in LP
  - \( \neg \): classical negation, monotonicity, open world assumption
  - \textit{not}: default negation, nonmonotonicity, closed world assumption

<table>
<thead>
<tr>
<th>DL ontology ( \mathcal{O} )</th>
<th>Logic Program ( \mathcal{P} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{Child} ( \sqsubseteq ) \texttt{Person}</td>
<td>\texttt{person}(X) \leftarrow \texttt{child}(X)</td>
</tr>
<tr>
<td>( \neg \texttt{Child} \sqsubseteq \texttt{Adult}</td>
<td>\texttt{adult}(X) \leftarrow \texttt{not} \texttt{child}(X)</td>
</tr>
<tr>
<td>\texttt{Person}(\texttt{john})</td>
<td>\texttt{person}(\texttt{john})</td>
</tr>
<tr>
<td>( \mathcal{O} \not\models \texttt{Adult}(\texttt{john}) )</td>
<td>( \mathcal{P} ) infers \texttt{adult}(\texttt{john})</td>
</tr>
</tbody>
</table>

- DLs are strong in subsumption checking, LPs in expressing relations
- DLs allow complex expressions in heads (rhs of \( \sqsubseteq \)), while in LPs use of variables in rule bodies is more flexible

\ldots
**Problem Statement**

**Goal of the thesis**

Develop approaches for dealing with inconsistencies in DL-programs.

**• DL-programs:**

- Powerful formalism for solving advanced reasoning tasks on top of ontologies
- Possibility to add information from the rule part to ontology prior to querying it allows for bidirectional information flow

**• Issues:**

- Information exchange between rules and ontology can have unforeseen effects and cause inconsistency of the DL-program (absence of answer sets).

---

**DL-program** is a pair \( \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \), where

- \( \mathcal{O} \) is a DL ontology
- \( \mathcal{P} \) is a set of DL-rules of the form

\[
\begin{align*}
    a_1 \lor \ldots \lor a_k & \leftarrow b_1, \ldots, b_m, \text{not} \ b_{m+1}, \ldots, \text{not} \ b_n, \\
\end{align*}
\]

- \( a_i \)'s are first-order atoms and
- \( b_j \)'s are either first-order atoms or DL-atoms
DL-program: syntax

Example

\( \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \) is a DL-program.

\[ \mathcal{O} = \left\{ \begin{array}{ll}
(1) & \text{hasChild} \sqsubseteq \text{hasParent} \\
(2) & \text{Female} \sqsubseteq \neg \text{Male} \\
(3) & \text{Male}(\text{pat}) \\
(4) & \text{hasChild}(\text{pat}, \text{john}) \\
(5) & \text{boy}(\text{john}) \\
(6) & \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \uplus \text{boy}; \text{Male}](\text{pat}), \\
& \text{DL}[, \text{hasParent}](\text{john}, \text{pat})
\end{array} \right\} \]

\[ \mathcal{P} = \left\{ \begin{array}{l}
\end{array} \right\} \]
DL-atoms

\[ DL[Male \cup boy; Male](john) \]

**Intuition:** extend concept *Male* by *boy*, then query \( \mathcal{O} \) for *Male(john)*

A **DL-atom** is of the form

\[
DL[S_1 \text{ op}_1 p_1, \ldots, S_m \text{ op}_m p_m; Q](t)
\]

- \( S_i \): ontology concept or role
- \( \text{op}_i \in \{ \cup, \cup \} \): intuitively \( \cup \) (resp. \( \cup \)) increases \( S_i \) (resp. \( \neg S_i \)) by \( p_i \)
- \( p_i \): unary or binary logic program predicate (input predicate)
- \( Q(t) \) is a DL-query:
  - \( C(t), \neg C(t), t = t \), where \( C \) is an ontology concept
  - \( R(t_1, t_2), \neg R(t_1, t_2), t = t_1, t_2 \), where \( R \) is an ontology role
DL-programs: semantics

\[ \Pi = \langle O, P \rangle \] is a DL-program.

\[ O = \left\{ \begin{array}{ll}
(1) & \text{hasChild} \sqsubseteq \text{hasParent} \\
(2) & \text{Female} \sqsubseteq \neg \text{Male}
\end{array} \right\} \]

\[ P = \left\{ \begin{array}{l}
(5) & \text{boy}(john) \\
(6) & \text{hasfather}(john, pat) \leftarrow \text{DL}[\text{hasParent}](john, pat),
\end{array} \right\] \]

\[ \text{DL}[\text{Male} \uplus \text{boy}; \text{Male}](pat) \]

\[ d_1 \]

\[ d_2 \]

- Interpretation: \( I = \{ \text{boy}(john), \text{hasfather}(john, pat) \} \)
- Satisfaction relation: \( I \models^O \text{boy}(john) \) as \( \text{boy}(john) \in I \)
  \( I \models^O d_1 \) as \( O \models \text{hasParent}(john, pat) \)
DL-programs: semantics

\[ \Pi = \langle O, P \rangle \] is a DL-program.

\[ O = \begin{cases} 
(1) \text{hasChild} \sqsubseteq \text{hasParent} \\
(2) \text{Female} \sqsubseteq \neg \text{Male} \\
(3) \text{Male}(\text{pat}) \\
(4) \text{hasChild}(\text{pat}, \text{john}) 
\end{cases} \]

\[ P = \begin{cases} 
(5) \text{boy}(\text{john}); \\
(6) \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[; \text{hasParent}](\text{john}, \text{pat}), \\
\text{DL}[\text{Male} \cup \text{boy}; \text{Male}](\text{pat}) \\
\text{DL}\{d_1, d_2\} 
\end{cases} \]

- Interpretation: \( I = \{ \text{boy}(\text{john}), \text{hasfather}(\text{john}, \text{pat}) \} \)
- Satisfaction relation: \( I \models^O \text{boy}(\text{john}) \) as \( \text{boy}(\text{john}) \in I \)
  - \( I \models^O d_1 \) as \( O \models \text{hasParent}(\text{john}, \text{pat}) \)
  - \( I \models^O d_2 \) as \( O \cup \text{Male}(\text{john}) \models \text{Male}(\text{pat}) \)
- Answer sets: founded models (weak, flp semantics)
  - \( I \) is a weak and FLP answer set
- Inconsistent DL-program: no answer sets
Example: Inconsistent DL-program

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \]

\[ \mathcal{O} = \left\{ \begin{array}{l}
(1) \text{Child} \sqsubseteq \exists \text{hasParent} \\
(2) \text{Adopted} \sqsubseteq \text{Child} \\
(3) \text{Female} \sqsubseteq \neg \text{Male} \\
\end{array} \right\} \]

\[ \mathcal{P} = \left\{ \begin{array}{l}
(7) \text{ischildof}(\text{john}, \text{alex}); \\
(8) \text{boy}(\text{john}); \\
(9) \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \uplus \text{boy}; \text{Male}](\text{pat}), \\
\text{DL}[; \text{hasParent}](\text{john}, \text{pat}); \\
(10) \perp \leftarrow \neg \text{DL}[; \text{Adopted}](\text{john}), \\
\text{hasfather}(\text{john}, \text{pat}), \text{ischildof}(\text{john}, \text{alex}), \\
\neg \text{DL}[\text{Child} \uplus \text{boy}; \neg \text{Male}](\text{alex}) \\
\end{array} \right\} \]
Example: Inconsistent DL-program

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \]

\[ \mathcal{O} = \left\{ \begin{array}{ll}
    (1) & \text{Child} \sqsubseteq \exists \text{hasParent} \\
    (2) & \text{Adopted} \sqsubseteq \text{Child} \\
    (3) & \text{Female} \sqsubseteq \neg \text{Male} \\
  \end{array} \right\} \]

\[ \mathcal{P} = \left\{ \begin{array}{ll}
    (7) & \text{ischildof}(\text{john, alex}); \\
    (9) & \text{hasfather}(\text{john, pat}) \leftarrow \text{DL}[\text{Male} \sqcup \text{boy}; \text{Male}](\text{pat}), \\
        \text{DL}[; \text{hasParent}](\text{john, pat}); \\
    (10) & \bot \leftarrow \neg \text{DL}[; \text{Adopted}](\text{john}), \\
        \text{hasfather}(\text{john, pat}), \text{ischildof}(\text{john, alex}), \\
        \neg \text{DL}[\text{Child} \sqcup \text{boy}; \neg \text{Male}](\text{alex}) \\
  \end{array} \right\} \]
Example: Inconsistent DL-program

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \]

\[ \mathcal{O} = \{ \]
\( (1) \) Child \( \sqsubseteq \exists \text{hasParent} \)
\( (2) \) Adopted \( \sqsubseteq \) Child
\( (3) \) Female \( \sqsubseteq \neg \text{Male} \)
\[ \} \]

\[ \mathcal{P} = \{ \]
\( (7) \) ischildof(john, alex)
\( (8) \) boy(john)
\( (9) \) hasfather(john, pat) \( \leftarrow \) DL[Male \( \cup \) boy; Male](pat),
DL[; hasParent](john, pat);
\( (10) \) \( \perp \leftarrow \) not DL[; Adopted](john),
hasfather(john, pat), ischildof(john, alex),
not DL[Child \( \cup \) boy; \neg \text{Male}] (alex)
\[ \} \]
Example: Inconsistent DL-program

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \text{ is inconsistent!} \]

\[ \mathcal{O} = \begin{cases} (1) \text{\textit{Child}} \sqsubseteq \exists \text{\textit{hasParent}} & (4) \text{\textit{Male}(pat)} \\
(2) \text{\textit{Adopted}} \sqsubseteq \text{\textit{Child}} & (5) \text{\textit{Male}(john)} \\
(3) \text{\textit{Female}} \sqsubseteq \neg \text{\textit{Male}} & (6) \text{\textit{hasParent}(john, pat)} \end{cases} \]

\[ \mathcal{P} = \begin{cases} (7) \text{\textit{ischildof}(john, alex)}; & (8) \text{\textit{boy}(john)}; \\
(9) \text{\textit{hasfather}(john, pat)} \leftarrow \text{DL[\textit{Male} \uplus \textit{boy}; \textit{Male}](pat)}, \\
& \text{DL[; \textit{hasParent}(john, pat)}; \\
(10) \bot \leftarrow \text{not DL[; \textit{Adopted}(john)}, \\
& \text{\textit{hasfather}(john, pat), \textit{ischildof}(john, alex),} \\
& \text{\textit{not DL[\textit{Child} \uplus \textit{boy}; \neg \textit{Male}](alex)}.} \end{cases} \]

No answer sets
Related Work

- Repairing ontologies
  - consistent query answering over $DL$-$Lite$ ontologies based on repair technique [Bienvenu et al., 2014], [Lembo et al., 2010]
  - QA over $DL$-$Lite_A$ ontologies that miss expected tuples (abductive explanations corresponding to repairs) [Calvanese et al., 2012]

- Repairing nonmonotonic logic programs
  - extended abduction for deleting minimal sets of rules (in reality addition is also possible) [Sakama and Inoue, 2003]
  - debugging in ASP [Pührer, 2014], [Syrjänen, 2006]

- Handling inconsistencies in combination of rules and ontologies
  - paraconsistent semantics for MKNF KBs [Huang et al., 2013]
  - paraconsistent semantics, based on the HT logic [Fink, 2012]
  - stepwise debugging of inconsistent DL-programs [Oetsch et al., 2012]
  - inconsistency tolerance in DL-programs [Pührer et al., 2010]
Our goal: develop techniques for handling inconsistencies in DL-programs
Our approach: repair ontology ABox to regain consistency
Research Questions

On the theoretical level:

 questões

1. Repair problem formalization, complexity?
2. Under which DLs the repair computation is feasible?
3. Preferred repairs without complexity increase?
4. Can existing evaluation algorithms be extended to compute repairs?

On the practical level:

 questões

1. Practical algorithms and optimizations?
2. Can we reuse existing tools?
   - Benchmarks?
   - How to evaluate?
Contributions

On the theoretical level:

⊙ Repair semantics for DL-programs and its complexity
⊙ Algorithms for repair computation
⊙ Preference selection functions with benign properties

On the practical level:

⊙ Optimizations for $DL\text{-}Lite_A$ and $\mathcal{EL}$
⊙ Implementation as the dlliteplugin for the dlvhex$^2$ system
  implementation of repair semantics within drew$^3$ was not effective
  ◦ Set of novel benchmarks including real-world data
  ◦ Evaluation w.r.t. performance and quality of repairs

$^2$https://github.com/hexhex/core
$^3$https://github.com/ghxiao/drew
Repair Answer Sets

Definition
Let $\Pi = \langle O, P \rangle$ be a DL-program, where $O = \langle T, A \rangle$

- an ABox $A'$ is a repair of $\Pi$ if
  - $O' = \langle T, A' \rangle$ is consistent and
  - $\Pi' = \langle O', P \rangle$ has some answer set.

$rep_x(\Pi)$ is the set of all repairs of $\Pi$ ($x \in \{\text{weak}, \text{flp}\}$).
Repair Answer Sets

Definition

Let $\Pi = \langle \mathcal{O}, P \rangle$ be a DL-program, where $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$

- an ABox $\mathcal{A}'$ is a repair of $\Pi$ if
  - $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ is consistent and
  - $\Pi' = \langle \mathcal{O}', P \rangle$ has some answer set.

$\text{rep}_x(\Pi)$ is the set of all repairs of $\Pi$ ($x \in \{\text{weak, flp}\}$).

- $I$ is a repair answer set of $\Pi$, if $I \in \text{AS}_x(\Pi')$, where
  $\Pi' = \langle \mathcal{O}', P \rangle$, $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, and $\mathcal{A}' \in \text{rep}_x(\Pi)$.

$\text{RAS}_x(\Pi)$ is the set of all repair AS of $\Pi$.

$\text{rep}^I_x(\Pi)$ is the set of all $\mathcal{A}'$ under which $I$ is a repair answer set of $\Pi$. 
Example: repair

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \text{ is inconsistent!} \]

\[ \mathcal{O} = \ \begin{cases} 
(1) \text{Child} \sqsubseteq \exists \text{hasParent} \\
(2) \text{Adopted} \sqsubseteq \text{Child} \\
(3) \text{Female} \sqsubseteq \neg \text{Male} 
\end{cases} \]

\[ \mathcal{P} = \ \begin{cases} 
(7) \text{ischildof}(\text{john}, \text{alex}); \\
(8) \text{boy}(\text{john}); \\
(9) \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \uplus \text{boy}; \text{Male}](\text{pat}), \\
\quad \text{DL}[; \text{hasParent}](\text{john}, \text{pat}); \\
(10) \perp \leftarrow \neg \text{DL}[; \text{Adopted}](\text{john}), \\
\quad \text{hasfather}(\text{john}, \text{pat}), \text{ischildof}(\text{john}, \text{alex}), \\
\quad \neg \text{DL}[\text{Child} \uplus \text{boy}; \neg \text{Male}](\text{alex}).
\end{cases} \]

No answer sets
Example: repair

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \text{ is consistent!} \]

\[ \mathcal{O} = \begin{cases} 
(1) & \text{Child} \sqsubseteq \exists \text{hasParent} \\
(2) & \text{Adopted} \sqsubseteq \text{Child} \\
(3) & \text{Female} \sqsubseteq \neg \text{Male} 
\end{cases} \]

\[ \mathcal{P} = \begin{cases} 
(7) & \text{ischildof}(\text{john}, \text{alex}); \\
(8) & \text{boy}(\text{john}); \\
(9) & \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \cup \text{boy}; \text{Male}](\text{pat}), \\
& \text{DL}[; \text{hasParent}](\text{john}, \text{pat}); \\
(10) & \perp \leftarrow \text{not DL}[; \text{Adopted}](\text{john}), \\
& \text{hasfather}(\text{john}, \text{pat}), \text{ischildof}(\text{john}, \text{alex}), \\
& \text{not DL}[\text{Child} \cup \text{boy}; \neg \text{Male}](\text{alex}).
\end{cases} \]

\[ A' = \{ \text{Female}(\text{pat}), \text{Male}(\text{john}), \text{hasParent}(\text{john}, \text{pat}) \} \text{ is a repair} \]

\[ I' = \{ \text{ischildof}(\text{john}, \text{alex}), \text{boy}(\text{john}) \} \text{ is a repair answer set} \]

\[ A' \in \text{rep}_{flip}(\Pi), \ I' \in \text{RAS}_{flip}(\Pi) \]
Example: repair

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \text{ is consistent!} \]

\[ \mathcal{O} = \left\{ \begin{array}{ll}
(1) & \text{Child} \sqsubseteq \exists \text{hasParent} \\
(2) & \text{Adopted} \sqsubseteq \text{Child} \\
(3) & \text{Female} \sqsubseteq \neg \text{Male} \\
\end{array} \right. \]

\[ \mathcal{P} = \left\{ \begin{array}{ll}
(7) & \text{ischildof}(\text{john}, \text{alex}); \\
(8) & \text{boy}(\text{john}); \\
(9) & \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \uplus \text{boy}; \text{Male}](\text{pat}), \\
& \quad \text{DL}[: \text{hasParent}](\text{john}, \text{pat}); \\
(10) & \bot \leftarrow \text{not DL}[; \text{Adopted}](\text{john}), \\
& \quad \text{hasfather}(\text{john}, \text{pat}), \text{ischildof}(\text{john}, \text{alex}), \\
& \quad \text{not DL}[\text{Child} \uplus \text{boy}; \neg \text{Male}](\text{alex}). \\
\end{array} \right. \]

\[ \mathcal{A}'' = \{ \text{Male}(\text{pat}), \text{Male}(\text{john}) \} \text{ is a repair} \]

\[ \mathcal{I}' = \{ \text{ischildof}(\text{john}, \text{alex}), \text{boy}(\text{john}) \} \text{ is a repair answer set} \]

\[ \mathcal{A}'' \in \text{rep}_{flp}(\Pi), \mathcal{I}' \in \text{RAS}_{flp}(\Pi) \]
Complexity of Repair Answer Sets

**INSTANCE:** A ground DL-program $\Pi = \langle O, P \rangle$.

**QUESTION:** Does there exist a repair answer set for $\Pi$ under semantics $x$? (i.e. $RAS_x(\Pi) \neq \emptyset$?)

**Theorem**

Deciding $RAS_x(\Pi) \neq \emptyset$ and $AS_x(\Pi) \neq \emptyset$ have in all cases the same complexity for a ground $\Pi = \langle O, P \rangle$, where $O$ is in $DL$-$Lite_A$ or $EL$.

<table>
<thead>
<tr>
<th>$\Pi$</th>
<th>$RAS_{flp}(\Pi) \neq \emptyset$</th>
<th>$RAS_{weak}(\Pi) \neq \emptyset$</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>$\Sigma^P_2$-complete</td>
<td>$NP$-complete</td>
</tr>
<tr>
<td>disjunctive</td>
<td>$\Sigma^P_2$-complete</td>
<td>$\Sigma^P_2$-complete</td>
</tr>
</tbody>
</table>
Algorithm  \textit{AnsSet}: Compute $AS_x(\Pi)$

Input: A DL-program $\Pi$, $x \in \{\text{\textit{weak}, \textit{flp}}\}$
Output: $AS_x(\Pi)$

for $\hat{I} \in AS(\hat{\Pi})$ do
  \begin{align*}
  & \text{if } CMP(\hat{I}, \Pi) \land xFND(\hat{I}, \Pi) \text{ then} \\
  & \hspace{1em} \text{output } \hat{I}|_\Pi \\
  \end{align*}
end

- $\hat{\Pi}$ is $\Pi$ with all DL-atoms $a$ substituted by ordinary atoms $e_a$ plus additional guess rules $e_a \lor ne_a$ for values of $a$
- $CMP(\hat{I}, \Pi)$ is a compatibility check, i.e. check whether the values of DL-atoms coincide with the values of their replacement atoms in $\hat{I}$
- $xFND(\hat{I}, \Pi)$ is $x$-foundedness check
- $\hat{I}|_\Pi$ is a restriction of $\hat{I}$ to original language of $\Pi$
DL-program Evaluation

Algorithm AnsSet: Compute $AS_x(\Pi)$

Input: A DL-program $\Pi$, $x \in \{weak, flp\}$

Output: $AS_x(\Pi)$

(1) for $\hat{I} \in AS(\hat{\Pi})$ do
(2a,b) if $CMP(\hat{I}, \Pi) \land xFND(\hat{I}, \Pi)$ then
| output $\hat{I}|_\Pi$
| end
end

Reasons for inconsistencies:

1. $\hat{\Pi}$ does not have any answer sets;
2. for all $\hat{I} \in AS(\Pi)$:
   a. compatibility check failed or
   b. $x$-foundedness check failed.
Ontology Repair Problem

To address compatibility issue we introduce:

Definition

An ontology repair problem (ORP) is a triple $\mathcal{P} = \langle \mathcal{O}, D_1, D_2 \rangle$, where

- $\mathcal{O} = \langle T, A \rangle$ is an ontology and
- $D_i = \{ \langle U_j^i, Q_j^i \rangle \mid 1 \leq j \leq m_i \}$, $i = 1, 2$ are sets of pairs where
  - $U_j^i$ is any ABox (update) and
  - $Q_j^i$ is a DL-query.

ORP is $\text{NP}$-complete in general, even if $\mathcal{O} = \emptyset$. 
Ontology Repair Problem

To address compatibility issue we introduce:

Definition

An ontology repair problem (ORP) is a triple $P = \langle O, D_1, D_2 \rangle$, where

- $O = \langle T, A \rangle$ is an ontology and
- $D_i = \{ \langle U_{i_j}, Q_{i_j} \rangle | 1 \leq j \leq m_i \}, \ i = 1, 2$ are sets of pairs where
  - $U_{i_j}$ is any ABox (update) and
  - $Q_{i_j}$ is a DL-query.

A repair (solution) for $P$ is any ABox $A'$ s.t.

- $O' = \langle T, A' \rangle$ is consistent;
- $O' \cup U_{j_1}^1 \models Q_{j_1}^1$ holds for $1 \leq j_1 \leq m_1$;
- $O' \cup U_{j_2}^2 \not\models Q_{j_2}^2$ holds for $1 \leq j_2 \leq m_2$.
Ontology Repair Problem

To address compatibility issue we introduce:

Definition

An ontology repair problem (ORP) is a triple $\mathcal{P} = \langle \mathcal{O}, D_1, D_2 \rangle$, where

- $\mathcal{O} = \langle T, A \rangle$ is an ontology and
- $D_i = \{ \langle U^i_j, Q^i_j \rangle | 1 \leq j \leq m_i \}$, $i = 1, 2$ are sets of pairs where
  - $U^i_j$ is any ABox (update) and
  - $Q^i_j$ is a DL-query.

A repair (solution) for $\mathcal{P}$ is any ABox $A'$ s.t.

- $\mathcal{O}' = \langle T, A' \rangle$ is consistent;
- $\mathcal{O}' \cup U^1_{j_1} \models Q^1_j$ holds for $1 \leq j_1 \leq m_1$;
- $\mathcal{O}' \cup U^2_{j_2} \not\models Q^2_k$ holds for $1 \leq j_2 \leq m_2$.

ORP is $NP$-complete in general, even if $\mathcal{O} = \emptyset$. 
Tractable Cases of ORP for $DL$-$Lite_A$

C1. bounded $\delta^\pm$-change: $S = \{ \mathcal{A}' \mid |\mathcal{A}' \Delta \mathcal{A}| \leq k \}$, for some $k$

C2. deletion repair: $S = \{ \mathcal{A}' \mid \mathcal{A}' \subseteq \mathcal{A} \}$

C3. deletion $\delta^+$: first delete assertions, s.t. queries in $D_2$ are not satisfied, then add a bounded number of assertions to satisfy queries in $D_1$

C4. addition under bounded opposite polarity:
   $S = \{ \mathcal{A}' \mid |\mathcal{A}'^+ \setminus \mathcal{A}| \leq k \text{ or } |\mathcal{A}'^- \setminus \mathcal{A}| \leq k \}$, for some $k$
Tractable Cases of ORP for \( DL-Lite_A \)

C1. bounded \( \delta^\pm \)-change: \( S = \{ \mathcal{A}' \mid |\mathcal{A}' \Delta \mathcal{A}| \leq k \} \), for some \( k \)

C2. deletion repair: \( S = \{ \mathcal{A}' \mid \mathcal{A}' \subseteq \mathcal{A} \} \)

C3. deletion \( \delta^+ \): first delete assertions, s.t. queries in \( D_2 \) are not satisfied, then add a bounded number of assertions to satisfy queries in \( D_1 \)

C4. addition under bounded opposite polarity: 
\[
S = \{ \mathcal{A}' \mid |\mathcal{A}'^+ \setminus \mathcal{A}| \leq k \text{ or } |\mathcal{A}'^- \setminus \mathcal{A}| \leq k \}, \text{ for some } k
\]

Function \( \sigma : 2^{\mathcal{A}\mathcal{B}} \times \mathcal{A}\mathcal{B} \rightarrow 2^{\mathcal{A}\mathcal{B}} \) is a selection function, where \( \mathcal{A}\mathcal{B} \) is a set of all \( \mathcal{A}' \). \( \sigma(S, \mathcal{A}) \subseteq S \) is a set of preferred ABoxes.

A selection \( \sigma : 2^{\mathcal{A}\mathcal{B}} \times \mathcal{A}\mathcal{B} \rightarrow 2^{\mathcal{A}\mathcal{B}} \) is independent if
\[
\sigma(S, \mathcal{A}) = \sigma(S', \mathcal{A}) \cup \sigma(S \setminus S', \mathcal{A}), \text{ whenever } S' \subseteq S.
\]

Example

C1-C4 are independent, but \( \subseteq \)-minimal repairs are not.
Naive Repair Algorithm

**Algorithm** $RepAns$: Compute $\text{rep}_{(\sigma,x)}^{\hat{I} \mid \Pi}(\Pi)$

**Input:** $\Pi=\langle O, P \rangle$, $O=\langle T, A \rangle$, $\hat{I} \in AS(\hat{\Pi})$, $\sigma$, $x \in \{\text{weak, flp}\}$

**Output:** $\text{rep}_{(\sigma,x)}^{\hat{I} \mid \Pi}(\Pi)$

for $A' \in ORP(\hat{I}, \Pi, \sigma)$ do
  if $xFND(\hat{I}, \langle T, A', P \rangle)$ then
    output $A'$
  end
end

- $ORP(\hat{I}, \Pi, \sigma)$ computes $\sigma$ repairs for $\hat{I}, \Pi$
- $xFND(\hat{I}, \langle T, A', P \rangle)$ checks whether $\hat{I}$ is $x$-founded w.r.t. $\Pi'$

$RepAnsSet$ outputs $\hat{I} \mid \Pi$ if the result of $RepAns$ is nonempty.
Naive Repair Algorithm

**Algorithm** $\text{RepAns}$: Compute $\text{rep}_{(\sigma,x)}^{\hat{I}|\Pi}(\Pi)$

**Input:** $\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$, $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\hat{I} \in \text{AS}(\hat{\Pi})$, $\sigma$, $x \in \{\text{weak}, \text{flp}\}$

**Output:** $\text{rep}_{(\sigma,x)}^{\hat{I}|\Pi}(\Pi)$

for $\mathcal{A}' \in \text{ORP}(\hat{I}, \Pi, \sigma)$ do

\[\text{if } \text{xFND}(\hat{I}, \langle \mathcal{T}, \mathcal{A}', \mathcal{P} \rangle) \text{ then}\]

\[\text{output } \mathcal{A}'\]

end

---

- $\text{ORP}(\hat{I}, \Pi, \sigma)$ computes $\sigma$ repairs for $\hat{I}, \Pi$
- $\text{xFND}(\hat{I}, \langle \mathcal{T}, \mathcal{A}', \mathcal{P} \rangle)$ checks whether $\hat{I}$ is $x$-founded w.r.t. $\Pi'$

$\text{RepAnsSet}$ outputs $\hat{I}|\Pi$ if the result of $\text{RepAns}$ is nonempty.

$\text{RepAns}$ and $\text{RepAnsSet}$ are **sound** and **complete** for independent $\sigma$. 
Ground Support Sets

For optimization purposes we introduce support sets:

Support set for $d = \text{DL}[λ; Q](t)$ is a minimal set $S$, s.t. $S \cup T \models Q(t)$

$d = \text{DL}[\text{Male} \sqcup \text{boy}; \text{Male}](\text{pat}); \ T = \{\text{Female} \sqsubseteq \neg \text{Male}\}$

When is $d$ true under interpretation $I$?

- $\text{Male}(\text{pat}) \in A$
- $\text{boy}(\text{pat}) \in I$
- $\text{boy}(\text{alex}) \in I; \text{Female}(\text{alex}) \in A$
For optimization purposes we introduce support sets:

Support set for $d = DL[\lambda; Q](t)$ is a minimal set $S$, s.t. $S \cup T \models Q(t)$

$$d = DL[\text{Male } \cup \text{ boy}; \text{ Male}](\text{pat}); \ T_d = \{\text{Female } \sqsubseteq \neg\text{ Male}; \text{ Male}_{\text{boy}} \sqsubseteq \text{ Male}\}$$

When is $d$ true under interpretation $I$?

- $\text{Male}(\text{pat}) \in A$
- $\text{Male}_{\text{boy}}(\text{pat}) \in A_d$, s.t. $\text{boy}(\text{pat}) \in I$
- $\text{Male}_{\text{boy}}(\text{alex}) \in A_d$, s.t. $\text{boy}(\text{alex}) \in I; \text{ Female}(\text{alex}) \in A$

where $A_d = \{P_p(t) \mid P \cup p \in \lambda\} \cup \{\neg P_p(t) \mid P \cup p \in \lambda\}$
Ground Support Sets ($DL$-$Lite_A$)

**Definition**

$S \subseteq A \cup A_d$ is a support set for $d = DL[\lambda; Q](t)$ w.r.t. $O = \langle T, A \rangle$ in $DL$-$Lite_A$ if either

(i) $S = \{P(c)\}$ and $T_d \cup S \models Q(t)$ or

(ii) $S = \{P(c), P'(d)\}$, s.t. $T_d \cup S$ is inconsistent.

$\text{Supp}_O(d)$ is a set of all support sets for $d$.

$d = DL[\text{Male } \cup \text{ boy}; \text{Male}](\text{pat}); T_d = \{\text{Female } \subseteq \neg \text{Male}; \text{Male}_{\text{boy}} \subseteq \text{Male}\}$

When is $d$ true under interpretation $I$?

- $S_1 = \{\text{Male}(\text{pat})\}$, coherent with any $I$
- $S_2 = \{\text{Male}_{\text{boy}}(\text{pat})\}$, coherent with $I \supseteq \text{boy}(\text{pat})$
- $S_3 = \{\text{Male}_{\text{boy}}(\text{alex}); \text{Female}(\text{alex})\}$, coherent with $I \supseteq \text{boy}(\text{alex})$
Ground Support Sets ($\text{DL-Lite}_A$)

Definition

$S \subseteq A \cup A_d$ is a support set for $d = \text{DL}[\lambda; Q](t)$ w.r.t. $O = \langle T, A \rangle$ in $\text{DL-Lite}_A$ if either

(i) $S = \{P(c)\}$ and $T_d \cup S \models Q(t)$ or 

(ii) $S = \{P(c), P'(d)\}$, s.t. $T_d \cup S$ is inconsistent.

$\text{Supp}_O(d)$ is a set of all support sets for $d$.

$I \models^O d$ iff there exists $S \in \text{Supp}_O(d)$, which is coherent with $I$. 
Nonground Support Sets ($DL$-Lite$_A$)

\[ d = DL[Male \cup boy; Male](X), \quad T_d = \{Female \sqsubseteq \neg Male; Male_{boy} \sqsubseteq Male\} \]

Nonground support sets:

- \( S_1 = \{Male(X)\} \)
- \( S_2 = \{Male_{boy}(X)\} \)
- \( S_3 = \{Male_{boy}(Y); Female(Y)\} \)
Nonground Support Sets \((DL-Lite_A)\)

**Definition**

\[ S = \{ P(Y), P'(Y') \} \quad (S = \{ P(Y) \}) \] is a \( DL-Lite_A \) nonground support set for a DL-atom \( d(X) \) w.r.t. \( T \) if for every \( \theta : V \rightarrow C \) it holds that \( S\theta \) is a support set for \( d(X\theta) \) w.r.t. \( O_C = \langle T, A_C \rangle \), where \( A_C \) is a set of all possible assertions over \( C \).

Nonground support sets are **compact representations** of ground ones.
Nonground Support Sets ($DL\text{-}Lite_A$)

Definition

$S = \{P(Y), P'(Y')\} \quad (S = \{P(Y)\})$ is a $DL\text{-}Lite_A$ nonground support set for a DL-atom $d(X)$ w.r.t. $T$ if for every $\theta : V \rightarrow C$ it holds that $S\theta$ is a support set for $d(X\theta)$ w.r.t. $O_C = \langle T, A_C \rangle$, where $A_C$ is a set of all possible assertions over $C$.

Nonground support sets are compact representations of ground ones.

Completeness: family of nonground support sets $S$ for $d(X)$ is complete w.r.t. $O$ if for every $\theta : X \rightarrow C$ and $S \in Supp_O(d(X\theta))$ some $S' \in S$ exists, s.t. $S = S'\theta'$.

Complete support families allow to avoid access to $O$ during DL-atom evaluation.
Nonground Support Set Computation \((DL-Lite_A)\)

\[ d = DL[\text{Male} \cup \text{boy}; \text{Male}](X); \ T = \{\text{Female} \sqsubseteq \neg\text{Male}\} \]

- Construct \(T_d\) by compiling info about input predicates of \(d\) into \(T\):
  \[ T_d = T \cup \{\text{Male}_{\text{boy}} \sqsubseteq \text{Male}\} \]

- Compute classification \(Cl(T_d)\) (e.g. using ASP techniques):
  \[ cl(T_d) = T_d \cup \{\text{Male} \sqsubseteq \neg\text{Female}; \text{Male}_{\text{boy}} \sqsubseteq \neg\text{Female}\} \cup \{P \sqsubseteq P \mid P \in \mathcal{P}\} \]

- Extract support sets from \(Cl(T_d)\):
  - \(S_1 = \{\text{Male}(X)\}\)
  - \(S_2 = \{\text{Male}_{\text{boy}}(X)\}\)
  - \(S_3 = \{\text{Male}_{\text{boy}}(Y), \neg\text{Male}(Y)\}\)
  - \(S_4 = \{\text{Male}_{\text{boy}}(Y), \text{Female}(Y)\}\)
  - \(S_5 = \{\text{Male}(Y), \neg\text{Male}(Y)\}\)
  - \(S_6 = \{\text{Male}(Y), \text{Female}(Y)\}\)
Nonground Support Set Computation ($DL$-$Lite_A$)

$d = DL[\text{Male} \cup \text{boy}; \text{Male}](X); T = \{\text{Female} \sqsubseteq \neg \text{Male}\}$

- Construct $T_d$ by compiling info about input predicates of $d$ into $T$:
  
  $$T_d = T \cup \{\text{Male}_{\text{boy}} \sqsubseteq \text{Male}\}$$

- Compute classification $Cl(T_d)$ (e.g. using ASP techniques):
  
  $$cl(T_d) = T_d \cup \{\text{Male} \sqsubseteq \neg \text{Female}; \text{Male}_{\text{boy}} \sqsubseteq \neg \text{Female}\} \cup \{P \sqsubseteq P \mid P \in P\}$$

- Extract support sets from $Cl(T_d)$:
  
  - $S_1 = \{\text{Male}(X)\}$
  - $S_2 = \{\text{Male}_{\text{boy}}(X)\}$
  - $S_3 = \{\text{Male}_{\text{boy}}(Y), \neg \text{Male}(Y)\}$
  - $S_4 = \{\text{Male}_{\text{boy}}(Y), \text{Female}(Y)\}$
  - $S_5 = \{\text{Male}(Y), \neg \text{Male}(Y)\}$
  - $S_6 = \{\text{Male}(Y), \text{Female}(Y)\}$

  \[\bigcup\] is consistent!
Nonground Support Set Computation ($DL\text{-}Lite_A$)

\[ d = DL[Male \cup boy; Male](X); \quad T = \{Female \sqsubseteq \neg Male\} \]

- Construct $T_d$ by compiling info about input predicates of $d$ into $T$:
  \[ T_d = T \cup \{Male_{boy} \sqsubseteq Male\} \]

- Compute classification $Cl(T_d)$ (e.g. using ASP techniques):
  \[ cl(T_d) = T_d \cup \{Male \sqsubseteq \neg Female; Male_{boy} \sqsubseteq \neg Female\} \cup \{P \sqsubseteq P \mid P \in P\} \]

- Extract support sets from $Cl(T_d)$:
  \[
  \begin{align*}
    S_1 &= \{Male(X)\} \\
    S_2 &= \{Male_{boy}(X)\} \\
    S_3 &= \{Male_{boy}(Y), \neg Male(Y)\} \\
    S_4 &= \{Male_{boy}(Y), Female(Y)\}
  \end{align*}
  \]
  \{S_1, S_2, S_3, S_4\} \text{ is complete!}
Optimized Deletion-RAS Computation ($DL-\text{Lite}_A$)

✓ Compute complete support families $S$ for all DL-atoms of $\Pi$

- Construct $\hat{\Pi}$ from $\Pi = \langle O, P \rangle$:
  - Replace all DL-atoms $a$ with normal atoms $e_a$
  - Add guessing rules on values of $a$: $e_a \lor ne_a$

- For all $\hat{I} \in AS(\hat{\Pi})$: $D_p = \{ a \mid e_a \in \hat{I} \}$; $D_n = \{ a \mid ne_a \in \hat{I} \}$

✓ Ground support sets in $S$ wrt. $\hat{I}$ and $A$: $S^\hat{I}_{gr} \leftarrow Gr(S, \hat{I}, A)$

✓ Find $A'$, such that
  ✓ For all $a \in D_p$: there is $S \in S^\hat{I}_{gr}(a)$, s.t.
    $S \cap A' \neq \emptyset$ or $S \subseteq A_a$
  ✓ For all $a' \in D_n$: for all $S \in S^\hat{I}_{gr}(a')$:
    $S \cap A' = \emptyset$ and $S \not\subseteq A_{a'}$
  ✓ Minimality check of $\hat{I}|_\Pi$ wrt. $\Pi' = \langle O', P \rangle$, $O' = \langle T, A' \rangle$
Optimized Deletion-RAS Computation ($DL-Lite_A$)

✓ Compute complete support families $S$ for all DL-atoms of $\Pi$

  • Construct $\hat{\Pi}$ from $\Pi = \langle O, P \rangle$:
    • Replace all DL-atoms $a$ with normal atoms $e_a$
    • Add guessing rules on values of $a$: $e_a \lor \neg e_a$

  • For all $\hat{I} \in AS(\hat{\Pi})$
    - $D_p = \{ a | e_a \in \hat{I} \}$
    - $D_n = \{ a | \neg e_a \in \hat{I} \}$

✓ Ground support sets in $S$ wrt. $\hat{\Pi}$ and $\hat{\Pi}$: $S_{gr} \leftarrow Gr(S, \hat{\Pi}, \hat{\Pi})$

✓ Find $A'$, such that
  ✓ For all $a \in D_p$: there is $S \in S_{gr}(a)$, s.t.
    $S \cap A' \neq \emptyset$ or $S \subseteq A_a$
  ✓ For all $a' \in D_n$: for all $S \in S_{gr}(a')$:
    $S \cap A' = \emptyset$ and $S \not\subseteq A_{a'}$
  ✓ Minimality check of $\hat{\Pi}|_{\Pi}$ wrt. $\Pi' = \langle O', P \rangle$, $O' = \langle T, A' \rangle$
Extending Approach to \( \mathcal{EL} \)

\[
\mathcal{T} = \{ \text{StaffRequest} \equiv \exists \text{hasSubj}. \text{Staff} \sqcap \exists \text{hasTarg}. \text{Proj} \}
\]

\[
d = \text{DL}[\text{Proj} \uplus \text{projfile}; \text{StaffRequest}](X)
\]

- Construct \( \mathcal{T}_d \) by compiling info about input predicates of \( d \) into \( \mathcal{T} \):
  \[
  \mathcal{T}_d = \mathcal{T} \cup \{ \text{Proj}_{\text{projfile}} \sqsubseteq \text{Proj} \}
  \]

- Rewrite DL-query over normalized \( \mathcal{T}_d \) into a datalog program:
  \[
  \mathcal{T}_{d_{\text{norm}}} = \left\{ 
  \begin{array}{ll}
  (1) \text{StaffRequest} \sqsubseteq \exists \text{hasSubj}. \text{Staff} & (2) \text{Proj}_{\text{projfile}} \sqsubseteq \text{Proj} \\
  (3) \text{StaffRequest} \sqsubseteq \text{hasTarg}. \text{Proj} & (4) \exists \text{hasSubj}. \text{Staff} \sqsubseteq C_1 \\
  (5) \exists \text{hasTarg}. \text{Proj} \sqsubseteq C_2 & (6) C_1 \sqcap C_2 \sqsubseteq \text{StaffRequest}
  \end{array}
  \right\}
  \]
Extending Approach to \( \mathcal{EL} \)

\[
\mathcal{T} = \{ \text{StaffRequest} \equiv \exists \text{hasSubj}.\text{Staff} \land \exists \text{hasTarg}.\text{Proj} \}
\]

\[
d = \text{DL}[\text{Proj} \cup \text{projfile}; \text{StaffRequest}](X)
\]

- Construct \( \mathcal{T}_d \) by compiling info about input predicates of \( d \) into \( \mathcal{T} \):
  \[
  \mathcal{T}_d = \mathcal{T} \cup \{ \text{Proj}_{\text{projfile}} \sqsubseteq \text{Proj} \}
  \]

- Rewrite DL-query over normalized \( \mathcal{T}_d \) into a datalog program:
  \[
  \mathcal{P}_{\mathcal{T}_{dnorm}} = \begin{cases} 
  (1^*) \text{StaffRequest}(X) \leftarrow C_1(X), C_2(X) \\
  (2^*) C_1(X) \leftarrow \text{hasSubj}(X, Y), \text{Staff}(Y) \\
  (3^*) C_2(X) \leftarrow \text{hasTarg}(X, Y), \text{Proj}(Y) \\
  (4^*) \text{Proj}(X) \leftarrow \text{Proj}_{\text{projfile}}(X) 
  \end{cases}
  \]
Extending Approach to $\mathcal{EL}$

$\mathcal{T} = \{ StaffRequest \equiv \exists \text{hasSubj}.\ Staff \sqcap \exists \text{hasTarg}.\ Proj \}$

$d = \text{DL}[\text{Proj} \cup \text{projfile}; \ \text{StaffRequest}](X)$

- Construct $\mathcal{T_d}$ by compiling info about input predicates of $d$ into $\mathcal{T}$:
  $\mathcal{T_d} = \mathcal{T} \cup \{ \text{Proj}_{\text{projfile}} \sqsubseteq \text{Proj} \}$

- Rewrite DL-query over normalized $\mathcal{T_d}$ into a datalog program:

  $\mathcal{P}_{T_{d \text{norm}}} = \left\{ \begin{array}{l}
  (1^*) \ StaffRequest(X) \leftarrow C_1(X), C_2(X) \\
  (2^*) C_1(X) \leftarrow \text{hasSubj}(X, Y), \ Staff(Y) \\
  (3^*) C_2(X) \leftarrow \text{hasTarg}(X, Y), \ Proj(Y) \\
  (4^*) Proj(X) \leftarrow \text{Proj}_{\text{projfile}}(X) \end{array} \right\}$

- Unfold the DL-query and extract support sets:

  $\ StaffRequest(X) \leftarrow \text{hasSubj}(X, Y), \ Staff(Y), \text{hasTarg}(X, Z), \ Proj(Z)$

  $\ StaffRequest(X) \leftarrow \text{hasSubj}(X, Y), \ Staff(Y), \text{hasTarg}(X, Z), \ Proj_{\text{projfile}}(Z)$
Extending Approach to \( \mathcal{EL} \)

\[
\mathcal{T} = \{ \text{StaffRequest} \equiv \exists \text{hasSubj}.\text{Staff} \sqcap \exists \text{hasTarg}.\text{Proj} \}
\]
\[
d = \text{DL}[\text{Proj} \sqcup \text{projfile}; \text{StaffRequest}](X)
\]

- Construct \( \mathcal{T}_d \) by compiling info about input predicates of \( d \) into \( \mathcal{T} \):
  \[
  \mathcal{T}_d = \mathcal{T} \cup \{ \text{Proj}_{\text{projfile}} \subseteq \text{Proj} \}
  \]

- Rewrite DL-query over normalized \( \mathcal{T}_d \) into a datalog program:
  \[
  \mathcal{P}_{\mathcal{T}_{\text{norm}}} = 
  \begin{cases}
    (1^*) \text{StaffRequest}(X) \leftarrow C_1(X), C_2(X) \\
    (2^*) C_1(X) \leftarrow \text{hasSubj}(X, Y), \text{Staff}(Y) \\
    (3^*) C_2(X) \leftarrow \text{hasTarg}(X, Y), \text{Proj}(Y) \\
    (4^*) \text{Proj}(X) \leftarrow \text{Proj}_{\text{projfile}}(X)
  \end{cases}
  \]

- Unfold the DL-query and extract support sets:
  \[
  S_1 = \{ \text{hasSubj}(X, Y), \text{Staff}(X), \text{hasTarg}(X, Z), \text{Proj}(Z) \}
  \]
  \[
  S_2 = \{ \text{hasSubj}(X, Y), \text{Staff}(X), \text{hasTarg}(X, Z), \text{Proj}_{\text{projfile}}(Z) \}
  \]
Extending Approach to $\mathcal{EL}$

$\mathcal{T} = \{ \text{StaffRequest} \equiv \exists \text{hasSubj}.\text{Staff} \land \exists \text{hasTarg}.\text{Proj} \}$

$d = \text{DL}[\text{Proj} \uplus \text{projfile}; \text{StaffRequest}](X)$

- Construct $\mathcal{T}_d$ by compiling info about input predicates of $d$ into $\mathcal{T}$:
  $\mathcal{T}_d = \mathcal{T} \cup \{ \text{Proj}_{\text{projfile}} \sqsubseteq \text{Proj} \}$

- Rewrite DL-query over normalized $\mathcal{T}_d$ into a datalog program:

\[
\mathcal{P}_{\mathcal{T}_{d\text{norm}}} = \begin{cases} 
(1^*) \text{StaffRequest}(X) \leftarrow C_1(X), C_2(X) \\
(2^*) C_1(X) \leftarrow \text{hasSubj}(X, Y), \text{Staff}(Y) \\
(3^*) C_2(X) \leftarrow \text{hasTarg}(X, Y), \text{Proj}(Y) \\
(4^*) \text{Proj}(X) \leftarrow \text{Proj}_{\text{projfile}}(X)
\end{cases}
\]

- Unfold the DL-query and extract support sets:
  - infinitely many support sets (axioms $\exists R.A \sqsubseteq A$)
  - exponentially many for acyclic $\mathcal{T}$

- Completeness is costly!
- Compute partial support families: bound size/number
Optimized Deletion RAS Computation (EL)

✓ Compute partial support families $S$ for all DL-atoms of $\Pi$

- Construct $\hat{\Pi}$ from $\Pi = \langle O, P \rangle$:
  - Replace all DL-atoms $a$ with normal atoms $e_a$
  - Add guessing rules on values of $a$: $e_a \lor ne_a$
- For all $\hat{I} \in AS(\hat{\Pi})$: $D_p = \{ a \mid e_a \in \hat{I} \};$ $D_n = \{ a \mid ne_a \in \hat{I} \}$

✓ Ground support sets in $S$ wrt. $\hat{I}$ and $\mathcal{A}$: $S_{gr}^{\hat{I}} \leftarrow Gr(S, \hat{I}, \mathcal{A})$

✓ For all HS $H \subseteq \mathcal{A}$ of support families for all $a \in D_n$:

  ✓ If all $a \in D_p$ have at least one $S \in S_{gr}^{\hat{I}}$, s.t. $S \cap H = \emptyset$, then do eval. postcheck on $D_n$
  (evaluate atoms from $D_n$ over $I$ and $\mathcal{A} \setminus H$)

  ✓ Else do eval. postcheck on $D_n$ and $D_p$

✓ Check minimality of $\hat{I}|_{\Pi}$ wrt. $\Pi' = \langle T, \mathcal{A} \setminus H, P \rangle$
Optimized Deletion RAS Computation \((\mathcal{EL})\)

- Compute partial support families \(S\) for all DL-atoms of \(\Pi\)
  - Construct \(\hat{\Pi}\) from \(\Pi = \langle \mathcal{O}, \mathcal{P} \rangle\):
    - Replace all DL-atoms \(a\) with normal atoms \(e_a\)
    - Add guessing rules on values of \(a\): \(e_a \lor ne_a\)
  - For all \(\hat{I} \in AS(\hat{\Pi})\):
    \(D_p = \{a \mid e_a \in \hat{I}\}\)
    \(D_n = \{a \mid ne_a \in \hat{I}\}\)

**Sound wrt. deletion repair answer sets,**
complete if all support families are complete!

- If all \(a \in D_p\) have at least one \(S \in S_{gr}^I\), s.t.
  \(S \cap H = \emptyset\), then do eval. postcheck on \(D_n\)
  (evaluate atoms from \(D_n\) over \(I\) and \(\mathcal{A}\setminus H\))
- Else do eval. postcheck on \(D_n\) and \(D_p\)
- Check minimality of \(\hat{I}|_{\Pi}\) wrt. \(\Pi' = \langle \mathcal{T}, \mathcal{A}\setminus H, \mathcal{P} \rangle\)
System Architecture
Experiments

Assessment of our algorithms concerns the following aspects:

- **Scalability**
  - size of the DL-program data part
  - size of the ontology TBox
  - number of rules in the DL-program

- **Repair quality**
  - bounding number/type of assertions for deletion

- **Expressive features**
  - defaults
  - guesses
  - recursiveness

- **Real world data**
  - Taxi-driver assignment problem
  - Open Street Map

- **Effects of support family completeness**
Taxi-Driver Benchmark ($DL$-Lite$_A$)

$\mathcal{O} = \begin{cases} 
(1) & \text{Driver } \sqsubseteq \neg \text{Cust} \\
(2) & \exists \text{worksIn} \sqsubseteq \text{Driver} \\
(3) & \text{worksIn} \sqsubseteq \neg \text{notworksIn} \\
(4) & \text{adjoint} \sqsubseteq \neg \text{disjoint} \\
(5) & \text{EDriver} \sqsubseteq \text{Driver} 
\end{cases}$

$\mathcal{P} = \begin{cases} 
(5) & \text{cust}(X) \leftarrow \text{isIn}(X, Y), \neg \text{DL}[; \neg \text{Cust}](X); \\
(6) & \text{driver}(X) \leftarrow \neg \text{cust}(X), \text{isIn}(X, Y); \\
(7) & \text{drives}(X, Y) \leftarrow \text{driver}(X), \text{cust}(Y), \text{needsTo}(Y, Z1), \text{goTo}(X, Z2), \neg \text{omit}(X, Y); \\
& \text{DL}[; \text{adjoint}](Z1, Z2), \neg \text{omit}(X, Y); \\
(8) & \text{omit}(X, Y) \leftarrow \text{DL}[; \text{EDriver}](X), \text{needsTo}(Y, Z), \text{DL}[; \text{notworksIn}](X, Z); \\
(9) & \text{ok}(Y) \leftarrow \text{customer}(Y), \text{drives}(X, Y); \\
(10) & \text{fail} \leftarrow \text{customer}(Y), \neg \text{ok}(Y); \\
(11) & \bot \leftarrow \text{fail} 
\end{cases}$
Taxi-Driver Benchmark (DL-Lite_\mathcal{A})

- \mathcal{A}: 500 customers, 200 drivers (190 edrivers), 23 regions (Vienna districts), every driver works in 2-4 regions
- \mathcal{P}: randomly generated positions and intentions of customers and drivers
- Instance size reflects the size of the relevant data part
Open Street Map Benchmark ($\mathcal{EL}$)

$\mathcal{O} = \{\ (1) \  BuildingFeature \cap \exists isLocatedInside.\ Private \subseteq NoPublicAccess \\
              (2) \  BusStop \cap Roofed \subseteq CoveredBusStop \} \ $

$\mathcal{P} = \{\ (9) \  publicstation(X) \leftarrow \text{DL}[BusStop \cup busstop; CoveredBusStop](X); \not\text{DL}[; Private](X); \\
              (10) \  \bot \leftarrow \text{DL}[BuildingFeature \cup publicstation; NoPublicAccess](X), \ publicstation(X). \}$

- Rules on top of the MyITS ontology:
  - personalized route planning with semantic information
  - TBox with 406 axioms
- $\mathcal{O}$ (part): building features located inside private areas are not publicly accessible, covered bus stops are those with roof.
- $\mathcal{P}$ checks that public stations don’t lack public access, using CWA on private areas.

4http://www.kr.tuwien.ac.at/research/projects/myits/
Open Street Map Benchmark (EL)

- \(A\): bus stops (285) and leisure areas (682) of Cork, plus role isLocatedInside on them (9)
- Randomly made 80% bus stops roofed, 60% leisure areas private
- For isLocatedInside(\(bs, la\)) make \(bs\) a bus stop with \(p\) chance (x-axis)
- DL-atoms have few support sets
1. **Data part variations:**
   - \(A_{50}\) contains 50 children (7 adopted), 20 female, 32 male adults (20 times that many for \(A_{1000}\)), \(T\) is fixed
   - Instance size \(p\): facts \(\text{boy}(c)\), \(\text{isChildOf}(c, d)\) are in \(P\) with prob. \(p/100\).
2. **TBox part variations:**

- $\mathcal{T}_n$ additionally contains $P \sqsubseteq \text{Person}$ for all concepts $P$ of $\mathcal{O}$, for each concept $P$ and $1 \leq i \leq n$ the axiom $P_{\text{MemberOfSocGroup}} \sqsubseteq P$ is in $\mathcal{P}$ with prob. $p/100$, $\mathcal{A}_{50}$ is fixed
3. Rule part variations:

- $R_n$ additionally contains rules which identify contacts for children within a social group, contact information is propagated, $\mathcal{A}_{50}$ and $\mathcal{T}$ are fixed
## Benchmark Statistics

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Ontology expressivity</th>
<th>TBox size</th>
<th>Concepts</th>
<th>Roles</th>
<th>ABox Size</th>
<th>Individuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
<td>DL-Lite_A</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>A50</td>
<td>312</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A1000</td>
<td>6183</td>
</tr>
<tr>
<td>Network</td>
<td>DL-Lite_A</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>A67</td>
<td>204</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>A161</td>
<td>672</td>
</tr>
<tr>
<td>Taxi</td>
<td>Basic</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>A50</td>
<td>259</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A500</td>
<td>4370</td>
</tr>
<tr>
<td></td>
<td>Time</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>A50</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A500</td>
<td>418</td>
</tr>
<tr>
<td></td>
<td>Districts</td>
<td>389</td>
<td>339</td>
<td>41</td>
<td>A50</td>
<td>6744</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A500</td>
<td>50</td>
</tr>
<tr>
<td>LUBM</td>
<td>Basic</td>
<td>DL-Lite_A</td>
<td>95</td>
<td>44</td>
<td>31</td>
<td>7293</td>
</tr>
<tr>
<td></td>
<td>Diamond</td>
<td>DL-Lite_A</td>
<td>101</td>
<td>48</td>
<td>31</td>
<td>7412</td>
</tr>
<tr>
<td></td>
<td>Extended</td>
<td>DL-Lite_A</td>
<td>101</td>
<td>48</td>
<td>31</td>
<td>7412</td>
</tr>
<tr>
<td>Policy</td>
<td>EL</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>A40</td>
<td>199</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A100</td>
<td>475</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A1000</td>
<td>4615</td>
</tr>
<tr>
<td>OSM</td>
<td>EL</td>
<td>405</td>
<td>356</td>
<td>36</td>
<td>A40</td>
<td>4195</td>
</tr>
<tr>
<td>LUBM-basic</td>
<td>EL</td>
<td>94</td>
<td>47</td>
<td>28</td>
<td>A40</td>
<td>4195</td>
</tr>
</tbody>
</table>

**Network DL-Lite_A**: A = 3 4 2, A = 67 204 67, A = 161 672 161

**Taxi DL-Lite_A**: A = 3 4 2, A = 67 204 67, A = 161 672 161

**LUBM**: A = 95 44 31, A = 101 48 31
Conclusions

- **Hybrid Knowledge Bases**: rules + DL ontology
- **DL-programs**: loose coupling combination
- **Inconsistency** is a challenging issue
  - already for rules and ontology considered separately
- Many possibilities for repair
- **We focus on changing ontology data part to restore consistency**
Summary of Contributions

- **Repair semantics** for inconsistent DL-programs
- **Complexity** is the same as for ordinary AS computation if DL is in $DL-Lite_A$ or $EL$
- **Practical algorithms** for deletion repair answer set computation based on support sets
- **Implementation** as the dlliteplugin within the dlvhex system
- **Evaluation** on a set of novel benchmarks (promising results)
- **Further optimizations**: pruning out DL-atoms
Future Work

- Extend work to other DLs
- Practical algorithms for other independent selections
- Further optimizations
- Repairing rules and DL-atoms
- Paraconsistent reasoning ...
Relevant Publications

Thomas Eiter, Michael Fink, and Daria Stepanova.
Semantic independence in DL-programs.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Data repair of inconsistent DL-programs.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Inconsistency management for Description Logic Programs and beyond.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Towards practical deletion repair of inconsistent DL-programs.

Thomas Eiter, Michael Fink, Christoph Redl, and Daria Stepanova.
Exploiting support sets for answer set programs with external computations.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Computing repairs for inconsistent DL-programs over EL ontologies.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Towards practical deletion repair of inconsistent DL-programs.

Daria Stepanova.
Inconsistencies in hybrid knowledge bases.
Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué.
Querying inconsistent description logic knowledge bases under preferred repair semantics.

Diego Calvanese, Magdalena Ortiz, Mantas Simkus, and Giorgio Stefanoni.
The complexity of explaining negative query answers in DL-Lite.

Jos de Bruijn, David Pearce, Axel Polleres, and Agustín Valverde.
Quantified equilibrium logic and hybrid rules.

Jos de Bruijn, Thomas Eiter, Axel Polleres, and Hans Tompits.
Embedding nonground logic programs into autoepistemic logic for knowledge-base combination.

Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits.
Combining answer set programming with description logics for the semantic web.

Michael Fink.
Paraconsistent hybrid theories.

Michael Gelfond and Vladimir Lifschitz.
The stable model semantics for logic programming.

P. J. Hayes.
The logic of frames.
In Frame Conceptions and Text Understanding, pages 46–61. 1979.
Shasha Huang, Qingguo Li, and Pascal Hitzler.  
Reasoning with inconsistencies in hybrid MKNF knowledge bases.  

Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo.  
Inconsistency-tolerant semantics for description logic ontologies.  

Alon Y. Levy and Marie-Christine Rousset.  
Combining horn rules and description logics in CARIN.  

John McCarthy.  
Programs with common sense.  

Marvin Minsky.  
A framework for representing knowledge.  

Boris Motik and Riccardo Rosati.  
Reconciling Description Logics and Rules.  

Boris Motik, Ulrike Sattler, and Rudi Studer.  
Query answering for OWL-DL with rules.  

Johannes Oetsch, Jörg Pührer, and Hans Tompits.  
Stepwise debugging of description-logic programs.  
Jörg Pührer, Stijn Heymans, and Thomas Eiter.
Dealing with inconsistency when combining ontologies and rules using DL-programs.

Jörg Pührer.

M. Ross Quillan.
Word concepts: A theory and simulation of some basic capabilities.

Riccardo Rosati.
On the decidability and complexity of integrating ontologies and rules.

Riccardo Rosati.
Dl+log: Tight integration of description logics and disjunctive datalog.

Chiaki Sakama and Katsumi Inoue.
An abductive framework for computing knowledge base updates.

Tommi Syrjänen.
Debugging Inconsistent Answer-Set Programs.
**DL-program**

Consider grounding $\text{grd}(\Pi) = \langle \mathcal{O}, \text{grd}(\mathcal{P}) \rangle$ of $\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$ over $\mathcal{C}$ and $\mathcal{P}$.

**Interpretation** $I$ is a consistent set of ground literals over $\mathcal{C}$ and $\mathcal{P}$.

- for ground literal $\ell$: $I \models^\mathcal{O} \ell$ iff $\ell \in I$;
- for ground DL-atom $a = DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{c})$:

$$I \models^\mathcal{O} a$$

iff $\mathcal{T} \cup A \cup \lambda^I(a) \models Q(\mathbf{c})$, where $\lambda^I(a) = \bigcup_{i=1}^{m} A_i(I)$ is a DL-update of $\mathcal{O}$ under $I$ by $a$:

- $A_i(I) = \{S_i(t) \mid p_i(t) \in I\}$, for $op_i = \cup$;
- $A_i(I) = \{\neg S_i(t) \mid p_i(t) \in I\}$, for $op_i = \cup$;
- $A_i(I) = \{\neg S_i(t) \mid p_i(t) \not\in I\}$, for $\cap$.

**FLP-reduct** $\mathcal{P}_{\text{flip}}^{I,\mathcal{O}}$ of $\mathcal{P}$ is a set of ground DL-rules $r$ s.t. $I \models b^+(r)$, $I \not\models b^-(r)$.

**Weak-reduct** $\mathcal{P}_{\text{weak}}^{I,\mathcal{O}}$ of $\mathcal{P}$: removes all DL-atoms $b_i$, $1 \leq i \leq k$ and all not $b_j$, $k < j \leq m$ from the rules of $\mathcal{P}_{\text{flip}}^{I,\mathcal{O}}$.

$I$ is an $x$-answer set of $\mathcal{P}$ iff $I$ is a minimal model of its $x$-reduct.
Table: Family benchmark: data size variations, fixed $P$ and $T$
**Family: TBox** \( (DL\text{-}Lite}_A) \)

<table>
<thead>
<tr>
<th>p</th>
<th>AS</th>
<th>RAS ( T_{max} = 500 )</th>
<th>AS</th>
<th>RAS ( T_{max} = 5000 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no_restr</td>
<td>( lim = 10 )</td>
<td>no_restr</td>
<td>( lim = 10 )</td>
</tr>
<tr>
<td>10 (20)</td>
<td>0.15 (0)[0]</td>
<td>0.32 (0)[20]</td>
<td>1.95 (0)[20]</td>
<td>0.28 (0)[0]</td>
</tr>
<tr>
<td>20 (20)</td>
<td>0.16 (0)[0]</td>
<td>0.47 (0)[20]</td>
<td>2.17 (0)[20]</td>
<td>0.48 (0)[0]</td>
</tr>
<tr>
<td>30 (20)</td>
<td>0.17 (0)[0]</td>
<td>0.68 (0)[20]</td>
<td>2.47 (0)[20]</td>
<td>0.75 (0)[0]</td>
</tr>
<tr>
<td>40 (20)</td>
<td>0.19 (0)[0]</td>
<td>0.93 (0)[20]</td>
<td>2.78 (0)[20]</td>
<td>1.10 (0)[0]</td>
</tr>
<tr>
<td>50 (20)</td>
<td>0.20 (0)[0]</td>
<td>1.25 (0)[20]</td>
<td>3.19 (0)[20]</td>
<td>1.51 (0)[0]</td>
</tr>
<tr>
<td>60 (20)</td>
<td>0.21 (0)[0]</td>
<td>1.58 (0)[20]</td>
<td>3.56 (0)[20]</td>
<td>1.99 (0)[0]</td>
</tr>
<tr>
<td>70 (20)</td>
<td>0.23 (0)[0]</td>
<td>2.09 (0)[20]</td>
<td>4.18 (0)[20]</td>
<td>2.56 (0)[0]</td>
</tr>
<tr>
<td>80 (20)</td>
<td>0.24 (0)[0]</td>
<td>2.54 (0)[20]</td>
<td>4.68 (0)[20]</td>
<td>3.17 (0)[0]</td>
</tr>
<tr>
<td>90 (20)</td>
<td>0.26 (0)[0]</td>
<td>3.06 (0)[20]</td>
<td>5.28 (0)[20]</td>
<td>3.91 (0)[0]</td>
</tr>
<tr>
<td></td>
<td>3.58 (0)[20]</td>
<td>6.03 (0)[20]</td>
<td>12.89 (0)[20]</td>
<td>15.96 (0)[20]</td>
</tr>
<tr>
<td></td>
<td>27.76 (0)[20]</td>
<td>31.42 (0)[20]</td>
<td>48.46 (0)[20]</td>
<td>53.24 (0)[20]</td>
</tr>
<tr>
<td></td>
<td>76.39 (0)[20]</td>
<td>81.54 (0)[20]</td>
<td>108.33 (0)[20]</td>
<td>114.71 (0)[20]</td>
</tr>
<tr>
<td></td>
<td>146.62 (0)[20]</td>
<td>152.91 (0)[20]</td>
<td>191.37 (0)[20]</td>
<td>198.72 (0)[20]</td>
</tr>
<tr>
<td></td>
<td>241.51 (0)[20]</td>
<td>248.19 (0)[20]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Family benchmark: TBox size variations, fixed \( P \) and \( A_{50} \)
Family: Rules ($DL{-}Lite_{A}$)

<table>
<thead>
<tr>
<th>$\mathbf{p}$</th>
<th>$\text{Rules}_{\text{max}} = 50$</th>
<th>$\text{Rules}_{\text{max}} = 500$</th>
<th>$\text{Rules}_{\text{max}} = 5000$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{RAS}$</td>
<td>$\text{RAS}_{\text{lim}=10}$</td>
<td>$\text{RAS}$</td>
</tr>
<tr>
<td>10 (20)</td>
<td>0.55 (0)[20]</td>
<td>2.09 (0)[20]</td>
<td>2.56 (0)[20]</td>
</tr>
<tr>
<td>20 (20)</td>
<td>0.69 (0)[20]</td>
<td>2.35 (0)[20]</td>
<td>5.22 (0)[20]</td>
</tr>
<tr>
<td>30 (20)</td>
<td>0.90 (0)[20]</td>
<td>2.67 (0)[20]</td>
<td>8.50 (0)[20]</td>
</tr>
<tr>
<td>40 (20)</td>
<td>0.97 (0)[20]</td>
<td>2.86 (0)[20]</td>
<td>11.86 (0)[20]</td>
</tr>
<tr>
<td>50 (20)</td>
<td>1.18 (0)[20]</td>
<td>3.11 (0)[20]</td>
<td>14.91 (0)[20]</td>
</tr>
<tr>
<td>60 (20)</td>
<td>1.29 (0)[20]</td>
<td>3.28 (0)[20]</td>
<td>17.68 (0)[20]</td>
</tr>
<tr>
<td>70 (20)</td>
<td>1.42 (0)[20]</td>
<td>3.19 (0)[20]</td>
<td>20.11 (0)[20]</td>
</tr>
</tbody>
</table>

Table: Family benchmark: rule size variations, fixed $\mathcal{T}$ and $\mathcal{A}_{50}$
### Taxi-Driver

**Table:** Taxi-driver benchmark results: $A_{500}$

<table>
<thead>
<tr>
<th>$p$</th>
<th>AS</th>
<th>$RAS_{no_restr}$</th>
<th>$lim = 3$</th>
<th>$lim = 10$</th>
<th>$limp = 2$</th>
<th>$limc = 10$</th>
<th>$EDriver$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (20)</td>
<td>2.11 (0) [0]</td>
<td>9.22 (0) [7]</td>
<td>25.05 (0) [6]</td>
<td>24.91 (0) [7]</td>
<td>12.32 (0) [7]</td>
<td>10.24 (0) [6]</td>
<td>7.56 (0) [0]</td>
</tr>
<tr>
<td>10 (20)</td>
<td>2.23 (0) [0]</td>
<td>14.17 (0)[20]</td>
<td>46.37 (0)[20]</td>
<td>46.52 (0)[20]</td>
<td>20.54 (0)[20]</td>
<td>15.75 (0)[15]</td>
<td>12.16 (0) [4]</td>
</tr>
<tr>
<td>18 (20)</td>
<td>5.58 (0) [5]</td>
<td>15.96 (0)[20]</td>
<td>51.89 (0)[20]</td>
<td>52.44 (0)[20]</td>
<td>23.11 (0)[20]</td>
<td>17.93 (0)[20]</td>
<td>28.00 (0)[20]</td>
</tr>
<tr>
<td>26 (20)</td>
<td>17.95 (0)[12]</td>
<td>18.28 (0)[20]</td>
<td>55.30 (0)[20]</td>
<td>55.84 (0)[20]</td>
<td>25.57 (0)[20]</td>
<td>20.27 (0)[20]</td>
<td>31.76 (0)[20]</td>
</tr>
<tr>
<td>34 (20)</td>
<td>37.87 (0)[17]</td>
<td>20.81 (0)[20]</td>
<td>58.71 (0)[20]</td>
<td>58.51 (0)[20]</td>
<td>28.35 (0)[20]</td>
<td>22.93 (0)[20]</td>
<td>36.00 (0)[20]</td>
</tr>
<tr>
<td>$p$</td>
<td>AS</td>
<td>( RAS )</td>
<td>( lim = 20 )</td>
<td>( limp = 2 )</td>
<td>( limc = 20 )</td>
<td>( IS )</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>-----------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>2 (20)</td>
<td>3.97 (0)[0]</td>
<td>13.98 (0)[20]</td>
<td>38.90 (0)[20]</td>
<td>16.01 (0)[20]</td>
<td>15.24 (0)[20]</td>
<td>15.20 (0)[6]</td>
<td></td>
</tr>
<tr>
<td>6 (20)</td>
<td>4.25 (0)[0]</td>
<td>16.16 (0)[20]</td>
<td>115.62 (0)[19]</td>
<td>18.08 (0)[20]</td>
<td>18.63 (0)[19]</td>
<td>11.16 (0)[2]</td>
<td></td>
</tr>
<tr>
<td>10 (20)</td>
<td>4.64 (0)[0]</td>
<td>18.95 (0)[20]</td>
<td>245.40 (0)[7]</td>
<td>20.85 (0)[20]</td>
<td>20.79 (0)[4]</td>
<td>9.12 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>14 (20)</td>
<td>4.86 (0)[0]</td>
<td>21.50 (0)[20]</td>
<td>236.40 (1)[3]</td>
<td>23.73 (0)[20]</td>
<td>23.50 (0)[1]</td>
<td>9.53 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>18 (20)</td>
<td>5.33 (0)[0]</td>
<td>24.86 (0)[20]</td>
<td>230.21 (0)[1]</td>
<td>27.11 (0)[20]</td>
<td>26.86 (0)[0]</td>
<td>10.15 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>22 (20)</td>
<td>5.54 (0)[0]</td>
<td>28.21 (0)[20]</td>
<td>228.12 (0)[0]</td>
<td>30.19 (0)[20]</td>
<td>29.93 (0)[0]</td>
<td>10.36 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>26 (20)</td>
<td>5.71 (0)[0]</td>
<td>31.50 (0)[20]</td>
<td>222.78 (0)[0]</td>
<td>33.84 (0)[20]</td>
<td>33.26 (0)[0]</td>
<td>10.75 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>30 (20)</td>
<td>6.07 (0)[0]</td>
<td>36.88 (0)[20]</td>
<td>225.18 (0)[0]</td>
<td>38.82 (0)[20]</td>
<td>38.47 (0)[0]</td>
<td>11.45 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>34 (20)</td>
<td>6.36 (0)[0]</td>
<td>42.18 (0)[20]</td>
<td>241.30 (0)[0]</td>
<td>44.29 (0)[20]</td>
<td>44.01 (0)[0]</td>
<td>12.22 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>38 (20)</td>
<td>6.55 (0)[0]</td>
<td>46.07 (0)[20]</td>
<td>245.77 (0)[0]</td>
<td>47.87 (0)[20]</td>
<td>47.64 (0)[0]</td>
<td>12.41 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>42 (20)</td>
<td>6.93 (0)[0]</td>
<td>52.50 (0)[20]</td>
<td>255.74 (0)[0]</td>
<td>54.17 (0)[20]</td>
<td>56.91 (0)[0]</td>
<td>12.94 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>46 (20)</td>
<td>7.15 (0)[0]</td>
<td>56.98 (0)[20]</td>
<td>276.52 (5)[0]</td>
<td>58.96 (0)[20]</td>
<td>58.47 (0)[0]</td>
<td>13.35 (0)[0]</td>
<td></td>
</tr>
<tr>
<td>50 (20)</td>
<td>7.53 (0)[0]</td>
<td>63.96 (0)[20]</td>
<td>276.07 (5)[0]</td>
<td>65.79 (0)[20]</td>
<td>65.50 (0)[0]</td>
<td>14.18 (0)[0]</td>
<td></td>
</tr>
</tbody>
</table>

**Table**: LUBM benchmark results
# Network Guessing

<table>
<thead>
<tr>
<th>$p$</th>
<th>$no_{\text{restr}}$</th>
<th>$lim = 10$</th>
<th>$limc = 100$</th>
<th>$Broken$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (20)</td>
<td>178.52 (3)[15]</td>
<td>187.65 (2)[16]</td>
<td>175.64 (2)[16]</td>
<td>179.57 (3)[15]</td>
</tr>
<tr>
<td>4 (20)</td>
<td>201.89 (6)[10]</td>
<td>211.10 (7) [9]</td>
<td>213.66 (9) [7]</td>
<td>178.55 (3)[13]</td>
</tr>
<tr>
<td>10 (20)</td>
<td>190.58 (9) [0]</td>
<td>184.80 (8) [1]</td>
<td>191.54 (9) [0]</td>
<td>191.06 (9) [0]</td>
</tr>
</tbody>
</table>

**Table**: Network-guessing benchmark results: $A_{161}$
# Network Connectivity

Table: Network-connectivity benchmark results: $A_{161}$

<table>
<thead>
<tr>
<th>$p$</th>
<th>no_restr</th>
<th>$lim = 3$</th>
<th>$lim = 20$</th>
<th>$lim = 100$</th>
<th>$Broken, forbid$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (20)</td>
<td>179.49 (1)[19]</td>
<td>280.73 (16)[0]</td>
<td>288.64 (17)[3]</td>
<td>176.06 (1)[19]</td>
<td>125.47 (0)[0]</td>
</tr>
<tr>
<td>4 (20)</td>
<td>218.80 (8)[12]</td>
<td>291.80 (18)[0]</td>
<td>295.48 (19)[1]</td>
<td>226.25 (8)[12]</td>
<td>127.68 (0)[0]</td>
</tr>
<tr>
<td>8 (20)</td>
<td>230.79 (9)[11]</td>
<td>298.39 (19)[0]</td>
<td>300.00 (20)[0]</td>
<td>232.65 (9)[11]</td>
<td>126.97 (0)[0]</td>
</tr>
<tr>
<td>10 (20)</td>
<td>258.08 (14)[5]</td>
<td>300.00 (20)[0]</td>
<td>300.00 (17)[0]</td>
<td>259.69 (14)[6]</td>
<td>125.63 (0)[0]</td>
</tr>
</tbody>
</table>
Optimizations: Independent DL-atoms

In our repair approach, the number of DL-atoms impacts performance.

**Optimizations:** identify DL-atoms that always have the same value!

**Definition**

A ground DL-atom $a$ is **independent** if for all satisfiable ontologies $\mathcal{O}, \mathcal{O}'$ and all interpretations $I, I'$ it holds that $I \models^\mathcal{O} a$ iff $I' \models^\mathcal{O}' a$.

A ground DL-atom $a$ is a **contradiction** (resp. **tautology**), if for all satisfiable ontologies $\mathcal{O}$ and all interpretations $I$, it holds that $I \not\models^\mathcal{O} a$ (resp. $I \models^\mathcal{O} a$).

**Contradiction:**

$DL[; C \not\sqsubseteq C]()$;

$\ldots$ ?

**Tautology:**

$DL[; C \sqsubseteq C]()$;

$\ldots$ ?
When is a DL-atom contradictory in general?

**Proposition**

A ground DL-atom $a = DL[\lambda; Q](t)$ is contradictory iff $\lambda = \epsilon$ and $Q(t)$ is unsatisfiable, i.e. has one of the forms:

- $C \nsubseteq C$;
- $C \nsubseteq \top$;
- $\bot \nsubseteq C$;
- $\bot \nsubseteq \top$;
- $\top \subseteq \bot$. 
When is a DL-atom $a = DL[\lambda; Q](t)$ tautologic in general?

- $Q$ is tautologic: $Q \in \{ C \sqsubseteq \top, \bot \sqsubseteq C, C \sqsubseteq C \}$;
- $\lambda$ is s.t. $a$ is tautologic.

**Concept query case distinction:**

- $DL[\lambda; \neg C](t)$: no tautologies
- $DL[\lambda; C](t)$: no tautologies
- $DL[\lambda; C \sqsubseteq D]()$: no tautologies
- $DL[\lambda; C \not\sqsubseteq D]()$: $C \neq D$. 

**Tautologies**
Tautologies

When is a DL-atom \( a = DL[\lambda; Q](t) \) tautologic in general?

- \( Q \) is tautologic: \( Q \in \{ C \sqsubseteq \top, \bot \sqsubseteq C, C \sqsubseteq C \} \);
- \( \lambda \) is s.t. \( a \) is tautologic.

Concept query case distinction:

\[
DL[\lambda; Q](t) \\
DL[\lambda; C](t) \\
DL[\lambda; C \sqsubseteq D]() \\
DL[\lambda; C \not\sqsubseteq D]()
\]

\( C \neq D. \)

Example

\[
a = DL[ C \cap p, C' \cup p, C' \cap q, C \cup q; \neg C](c)
\]

\( l \) is s.t. \( p(c) \not\in l, q(c) \not\in l \)
\( l \) is s.t. \( p(c) \in l, q(c) \not\in l \)
\( l \) is s.t. \( p(c) \not\in l, q(c) \in l \)
\( l \) is s.t. \( p(c) \in l, q(c) \in l \)
Tautologies

When is a DL-atom $a = DL[\lambda; Q](t)$ tautologic in general?

- $Q$ is tautologic: $Q \in \{C \sqsubseteq \top, \bot \sqsubseteq C, C \sqsubseteq C\}$;
- $\lambda$ is s.t. $a$ is tautologic.

**Concept query case distinction:**

- $DL[\lambda; -C](t)$
- $DL[\lambda; C](t)$
- $DL[\lambda; C \sqsubseteq D]()$

**Example**

$a = DL[ C \land p, C' \lor p, C' \land q, C \lor q; -C](c)$

- $I$ is s.t. $p(c) \not\in I$, $q(c) \not\in I$
- $I$ is s.t. $p(c) \in I$, $q(c) \not\in I$
- $I$ is s.t. $p(c) \not\in I$, $q(c) \in I$
- $I$ is s.t. $p(c) \in I$, $q(c) \in I$
Tautologies

When is a DL-atom \( a = DL[\lambda; Q](t) \) tautologic in general?

- \( Q \) is tautologic: \( Q \in \{ C \sqsubseteq T, \bot \sqsubseteq C, C \sqsubseteq C \} \);
- \( \lambda \) is s.t. \( a \) is tautologic.

Concept query case distinction:

Example

\[ a = DL[ C \cap p, C' \cup p, C' \cap q, C \cup q; \neg C](c) \]

\( l \) is s.t. \( p(c) \not\in l, \; q(c) \not\in l \)
\( l \) is s.t. \( p(c) \in l, \; q(c) \not\in l \)
\( l \) is s.t. \( p(c) \not\in l, \; q(c) \in l \)
\( l \) is s.t. \( p(c) \in l, \; q(c) \in l \)

\( \tau^l(a) = \{ \neg C(c) \} \)
**Tautologies**

When is a DL-atom $a = DL[\lambda; Q](t)$ tautologic in general?

- $Q$ is tautologic: $Q \in \{C \sqsubseteq \top, \bot \sqsubseteq C, C \sqsubseteq C\}$;
- $\lambda$ is s.t. $a$ is tautologic.

**Concept query case distinction:**

- $DL[\lambda; \neg C](t)$
  - no tautologies
- $DL[\lambda; C](t)$
  - no tautologies
- $DL[\lambda; C \sqsubseteq D]()$
  - no tautologies
- $DL[\lambda; C \not\sqsubseteq D]()$

**Example**

$a = DL[ C \cap p, C' \cup p, C' \cap q, C \cup q; \neg C](c)$

- $I$ is s.t. $p(c) \not\in I$, $q(c) \not\in I$  
  \[ \tau^I(a) = \{\neg C(c)\} \]
- $I$ is s.t. $p(c) \in I$, $q(c) \not\in I$  
- $I$ is s.t. $p(c) \not\in I$, $q(c) \in I$  
- $I$ is s.t. $p(c) \in I$, $q(c) \in I$
Tautologies

When is a DL-atom \( a = DL[\lambda; Q](t) \) tautologic in general?

- \( Q \) is tautologic: \( Q \in \{ C \sqsubseteq \top, \bot \sqsubseteq C, C \sqsubseteq C \} \);
- \( \lambda \) is s.t. \( a \) is tautologic.

Concept query case distinction:

- \( DL[\lambda; \neg C](t) \)
- \( DL[\lambda; C](t) \) no tautologies
- \( DL[\lambda; C \sqsubseteq D]() \) no tautologies
- \( C \neq D \).

Example

\[ a = DL[ C \land p, C' \lor p, C' \land q, C \lor q; \neg C](c) \]

- \( l \) is s.t. \( p(c) \notin l, q(c) \notin l \)
- \( l \) is s.t. \( p(c) \in l, q(c) \notin l \)
- \( l \) is s.t. \( p(c) \notin l, q(c) \in l \)
- \( l \) is s.t. \( p(c) \in l, q(c) \in l \)

\[ \tau^l(a) = \{ \neg C(c) \} \]

\[ \tau^l(a) = \{ C'(c), \neg C'(c) \} \]
Tautologies

When is a DL-atom \( a = DL[\lambda; Q](t) \) tautologic in general?

- \( Q \) is tautologic: \( Q \in \{ C \sqsubseteq \top, \bot \sqsubseteq C, C \sqsubseteq C \} \);
- \( \lambda \) is s.t. \( a \) is tautologic.

Concept query case distinction:

\[
\begin{align*}
DL[\lambda; \neg C](t) & \quad \text{no tautologies} \\
DL[\lambda; C](t) & \quad \text{no tautologies} \\
DL[\lambda; C \sqsubseteq D]() & \quad \text{no tautologies} \\
C \neq D.
\end{align*}
\]

Example

\[a = DL[ C \sqcap p, C' \sqcup p, C' \sqcap q, C \sqcup q; \neg C](c)\]

\( I \) is s.t. \( p(c) \notin I, q(c) \notin I \)
\( I \) is s.t. \( p(c) \in I, q(c) \notin I \)
\( I \) is s.t. \( p(c) \notin I, q(c) \in I \)
\( I \) is s.t. \( p(c) \in I, q(c) \in I \)

\[\tau^I(a) = \{\neg C(c)\}\]
\[\tau^I(a) = \{C'(c), \neg C'(c)\}\]
**Tautologies**

When is a DL-atom \( a = DL[\lambda; Q](t) \) tautologic in general?

- \( Q \) is tautologic: \( Q \in \{ C \sqsubseteq T, \bot \sqsubseteq C, C \sqsubseteq C \} \);
- \( \lambda \) is s.t. \( a \) is tautologic.

**Concept query case distinction:**

- \( DL[\lambda; \neg C](t) \)
  - no tautologies
- \( DL[\lambda; C](t) \)
  - no tautologies
- \( DL[\lambda; C \sqsubseteq D]() \)
  - no tautologies
- \( DL[\lambda; C \not\sqsubseteq D]() \)
  - no tautologies

**Example**

\[ a = DL[ C \sqcup p, C' \sqcup p, C' \sqcap q, C \sqcap q; \neg C](c) \]

\( \tau^I(a) = \{ \neg C(c) \} \)

\( \tau^I(a) = \{ C'(c), \neg C'(c) \} \)

\( \tau^I(a) = \{ \neg C(c) \} \)

\( l \) is s.t. \( p(c) \not\in l, q(c) \not\in l \)

\( l \) is s.t. \( p(c) \in l, q(c) \not\in l \)

\( l \) is s.t. \( p(c) \not\in l, q(c) \in l \)

\( l \) is s.t. \( p(c) \in l, q(c) \in l \)
Tautologies

When is a DL-atom \( a = \text{DL}[\lambda; Q](t) \) tautologic in general?

- \( Q \) is tautologic: \( Q \in \{ C \subseteq \top, \bot \subseteq C, C \subseteq C \} \);
- \( \lambda \) is s.t. \( a \) is tautologic.

**Concept query case distinction:**

\[
\begin{align*}
\text{DL}[\lambda; \neg C](t) & \quad \text{no tautologies} \\
\text{DL}[\lambda; C](t) & \quad \text{no tautologies} \\
\text{DL}[\lambda; C \sqsubseteq D]() & \quad \text{no tautologies}
\end{align*}
\]

Example

\( a = \text{DL}[ C \cap p, C' \cup p, C' \cap q, C \cup q; \neg C](c) \)

\( l \) is s.t. \( p(c) \not\in l, q(c) \not\in l \)
\( \tau^l(a) = \{\neg C(c)\} \)

\( l \) is s.t. \( p(c) \in l, q(c) \not\in l \)
\( \tau^l(a) = \{C'(c), \neg C'(c)\} \)

\( l \) is s.t. \( p(c) \not\in l, q(c) \in l \)
\( \tau^l(a) = \{\neg C(c)\} \)

\( l \) is s.t. \( p(c) \in l, q(c) \in l \)
When is a DL-atom \( a = DL[\lambda; Q](t) \) tautologic in general?

- \( Q \) is tautologic: \( Q \in \{ C \sqsubseteq \top, \bot \sqsubseteq C, C \sqsubseteq C \} \);
- \( \lambda \) is s.t. \( a \) is tautologic.

**Concept query case distinction:**

- \( DL[\lambda; -C](t) \)
  - no tautologies
- \( DL[\lambda; C](t) \)
  - no tautologies
- \( DL[\lambda; C \sqsubseteq D]() \)
  - no tautologies

**Example**

\[
a = DL[ C \sqcap p, C' \sqcup p, C' \sqcap q, C \sqcup q; -C](c)
\]

- \( I \) is s.t. \( p(c) \notin I, q(c) \notin I \)
  - \( \tau^I(a) = \{ -C(c) \} \)
- \( I \) is s.t. \( p(c) \in I, q(c) \notin I \)
  - \( \tau^I(a) = \{ C'(c), -C'(c) \} \)
- \( I \) is s.t. \( p(c) \notin I, q(c) \in I \)
  - \( \tau^I(a) = \{ -C(c) \} \)
- \( I \) is s.t. \( p(c) \in I, q(c) \in I \)
  - \( \tau^I(a) = \{ -C(c) \} \)
Tautologies with Concept Query

\[ DL[λ; ¬C](t) \]

Proposition

A ground DL-atom \( a \) with the query \( ¬C(t) \) is tautologic iff it has one of the following forms

1. \( DL[λ, C \cap p, C \cup p; ¬C](t) \),
2. \( DL[λ, C \cap p, D \cup p, D \cup p; ¬C](t) \),
Tautologies with Concept Query

\[ DL[\lambda; \neg C](t) \]

Proposition

A ground DL-atom \( a \) with the query \( \neg C(t) \) is tautologic iff it has one of the following forms

1. \( DL[\lambda, C \cap p, C \cup p; \neg C](t) \),
2. \( DL[\lambda, C \cap p, D \cup p, D \cup p; \neg C](t) \),
Tautologies with Concept Query

Proposition

A ground DL-atom \( a \) with the query \( \neg C(t) \) is tautologic iff it has one of the following forms

\[
\begin{align*}
\text{c1. } & DL[\lambda, C \cap p, C \cup p; \neg C](t), \\
\text{c2. } & DL[\lambda, C \cap p, D \cup p, D \cup p; \neg C](t),
\end{align*}
\]

where for every \( i = 0, \ldots, n+1 \), \( p_i = p_{i-1} \) for some \( j < i \) or \( p_i = p_0 \), and \( p_{n+1} = p_j \) for some \( j \leq n \) or \( p_{n+1} = p_0 \).
Tautologies with Concept Query

\[ DL[\lambda; \neg C](t) \]

Proposition

A ground DL-atom \( a \) with the query \( \neg C(t) \) is tautologic iff it has one of the following forms

1. \( DL[\lambda, C \sqcap p, C \sqcup p; \neg C](t) \),
2. \( DL[\lambda, C \sqcap p, D \sqcup p, D \sqcup p; \neg C](t) \),
3. \( DL[\lambda, C \sqcap p_0, C_0 \sqcup p_0, C_0 \sqcup p_0', C_1 \sqcup p_1, C_1 \sqcup p_1', \ldots, C^n \sqcup p_n, C^n \sqcup p_n', C \sqcup p_{n+1}; \neg C](t) \),
4. \( DL[\lambda, C \sqcap p_0, C_0 \sqcup p_0, C_0 \sqcup p_0', C_1 \sqcup p_1, C_1 \sqcup p_1', \ldots, C^n \sqcup p_n, C^n \sqcup p_n', D \sqcup p_{n+1}, D \sqcup p_{n+1}'; \neg C](t) \),

where for every \( i = 0, \ldots, n+1 \), \( p_i = p_j' \) for some \( j < i \) or \( p_i = p_0 \), and \( p_{n+1}' = p_{i_j} \) for some \( j \leq n \) or \( p_{n+1}' = p_0 \).
Tautologies with Concept Query

\[ DL[\lambda; \neg C](t) \]

**Proposition**

A ground DL-atom \( a \) with the query \( \neg C(t) \) is tautologic iff it has one of the following forms

\begin{align*}
\text{c1. } & DL[\lambda, C \land p, C \lor p; \neg C](t), \\
\text{c2. } & DL[\lambda, C \land p, D \lor p, D \lor p; \neg C](t), \\
\text{c3. } & DL[\lambda, C \land p_0, C^0 \lor p_0, C^0 \land p'_0, C^1 \lor p_1, C^1 \land p'_1, \ldots, \\
& C^n \lor p_n, C^n \land p'_n, C \lor p_{n+1}; \neg C](t), \\
\text{c4. } & DL[\lambda, C \land p_0, C^0 \lor p_0, C^0 \land p'_0, C^1 \lor p_1, C^1 \land p'_1, \ldots, \\
& C^n \lor p_n, C^n \lor p'_n, D \lor p_{n+1}, D \lor p'_{n+1}; \neg C](t),
\end{align*}

where for every \( i = 0, \ldots, n + 1, p_i = p'_j \) for some \( j < i \) or \( p_i = p_0 \), and \( p'_{n+1} = p'_{ij} \) for some \( j \leq n \) or \( p'_{n+1} = p_0 \).
Tautologies with Concept Query

\[ DL[\lambda; \neg C](t) \]

**Proposition**

A ground DL-atom \( a \) with the query \( \neg C(t) \) is tautologic iff it has one of the following forms

1. \( DL[\lambda, C \cap p, C \cup p; \neg C](t) \),
   \[ p_0 \]
2. \( DL[\lambda, C \cap p, D \cup p, D \cup p; \neg C](t) \),
   \[ t \]
3. \( DL[\lambda, C \cap p_0, C^0 \cup p_0, C^0 \cap p'_0, C^1 \cup p_1, C^1 \cap p'_1, \ldots, C^n \cup p_n, C^n \cap p'_n, C \cup p_{n+1}; \neg C](t) \),
4. \( DL[\lambda, C \cap p_0, C^0 \cup p_0, C^0 \cap p'_0, C^1 \cup p_1, C^1 \cap p'_1, \ldots, C^n \cup p_n, C^n \cup p'_n, D \cup p_{n+1}, D \cup p'_{n+1}; \neg C](t) \),

where for every \( i = 0, \ldots, n + 1, p_i = p'_j \) for some \( j < i \) or \( p_i = p_0 \), and \( p'_{n+1} = p'_{ij} \) for some \( j \leq n \) or \( p'_{n+1} = p_0 \).
Tautologies with Concept Query

$$DL[\lambda; \neg C](t)$$

Proposition

A ground DL-atom $a$ with the query $\neg C(t)$ is tautologic iff it has one of the following forms

c1. $DL[\lambda, C \cap p, C \cup p; \neg C](t)$,
c2. $DL[\lambda, C \cap p, D \cup p, D \cup p; \neg C](t)$,
c3. $DL[\lambda, C \cap p_0, C^0 \cup p_0, C^0 \cap p'_0, C^1 \cup p_1, C^1 \cap p'_1, \ldots, C^n \cup p_n, C^n \cap p'_n, C \cup p_{n+1}; \neg C](t)$,

Example

$a = DL[C \cap p, C' \cup p, C' \cap q, C \cup q; \neg C](c)$ is the special case of c3.
Tautologies with Role Query

What if the query is a role $R(t_1, t_2)$ or negated role $\neg R(t_1, t_2)$?

**Role query case distinction:**

- $DL[\lambda; Q](t_1, t_2)$
- $DL[\lambda; R](t_1, t_2)$, no tautologies
- $DL[\lambda; \neg R](t_1, t_2)$, c1-c4, where $C, C^i, D$-roles, $p_i, p'_i$-binary

**Example**

$(c_2)$ for roles is of the form $DL[\lambda, R_1 \sqcap p, R_2 \cup p; \neg R_1](t_1, t_2)$. 

(caption for the diagram: 17 / 22)
Axiomatization for Tautologies ($\mathcal{K}_{taut}$)

Axioms:

a0. $DL[; Q]()$,

a1. $DL[S \cap p, S \cup p; \neg S](t)$,

a2. $DL[S \cap p, S' \cup p, S' \cup p; \neg S](t)$,

where $Q \in \{S \subseteq S, S \subseteq T, T \not\subseteq \bot\}$, $S, S'$ are distinct.

Rules of Inference:

**Expansion**

$$DL[\lambda; Q](t) \quad DL[\lambda, \lambda'; Q](t) \quad (e)$$

**Increase**

$$DL[\lambda, S \cup p; Q](t)$$

$$DL[\lambda, S \cup q, S' \cup p, S' \cap q; Q](t) \quad (in_{\cup})$$

$$DL[\lambda, S \cup p; Q](t)$$

$$DL[\lambda, S \cup q, S' \cup p, S' \cap q; Q](t) \quad (in_{\cup})$$
Inclusion Constraints

Inclusion constraint (IC): $q(Y_1, \ldots, Y_n) \leftarrow p(X_1, \ldots, X_m)$, where $n \leq m$, $Y_i$ are pairwise distinct from $X_i$;

- $p \subseteq q$, if $n = m$ and $Y_i = X_i$;
- $p \subseteq q^-$, if $n = m$ and $Y_i = X_{n-i+1}$.

$C$ is a set of inclusion constraints of $\Pi$; $CL(C)$ is the logical closure of $C$;

$inp_a(C)$ is a set of all $q(Y) \leftarrow p(X)$ in $C$ s.t. $p, q$ are in $\lambda$, $a = DL[\lambda; Q](t)$;

$C$ is separable for $a$ if every $IC \in inp_a(CL(C))$ involves predicates of same arity.
Inclusion Constraints

**Inclusion constraint (IC):** \( q(Y_1, \ldots, Y_n) \leftarrow p(X_1, \ldots, X_m) \),
where \( n \leq m, \ Y_i \) are pairwise distinct from \( X_i \);

- \( p \subseteq q \), if \( n = m \) and \( Y_i = X_i \);  
- \( p \subseteq q^\perp \), if \( n = m \) and \( Y_i = X_{n-i+1} \).

\( C \) is a set of inclusion constraints of \( \Pi \); \( CL(C) \) is the logical closure of \( C \);
\( inp_a(C) \) is a set of all \( q(Y) \leftarrow p(X) \) in \( C \) s.t. \( p, q \) are in \( \lambda \), \( a = DL[\lambda; Q](t) \);

\( C \) is **separable** for \( a \) if every \( IC \in inp_a(CL(C)) \) involves predicates of same arity.

**Example**

\( \Pi = \{(1)\ p_2(Y, X) \leftarrow p_1(X, Y).\)
\( (2)\ p_3(Z) \leftarrow p_1(X, Y).\)
\( (3)\ r_1(X, Y) \leftarrow DL[S_1 \cup p_1, S_2 \cup p_2; S_3](X, Y).\}\)

\( C = \{p_1 \subseteq p_2^\perp, p_1 \subseteq p_3\}; \  CL(C) = C; \)
\( inp_a(CL(C)) = \{p_1 \subseteq p_2^\perp\}; \ C \) is separable for \( a \).
Axiomatization for Tautologies under Inclusion $\mathcal{K}_{taut}^\subseteq$

**Axioms:**

a0. $DL[\cdot; Q]()$,

a1. $DL[S \sqcap p, S \sqcup p; \neg S](t)$,

a2. $DL[S \sqcap p, S' \sqcup q, S' \sqcup q; \neg S](t)$,

where $q \in \{p, p^\perp\}$, $Q \in \{S \sqsubseteq S, S \sqsubseteq T, T \nsubseteq \bot\}$, $S, S'$ are distinct.

**Rules of Inference:** rules of $\mathcal{K}_{taut}$ plus additional:

**Inclusion**

\[
\frac{DL[\lambda, S \sqcup p; Q](t) \quad p \sqsubseteq q}{DL[\lambda, S \sqcup q; Q](t)} (i_1)
\]

\[
\frac{DL[\lambda, S \sqcup p; Q](t) \quad p \sqsubseteq q}{DL[\lambda, S \sqcup q; Q](t)} (i_2)
\]

**Increase**

\[
\frac{DL[\lambda, S \sqcup p; Q](t) \quad p \subseteq q}{DL[\lambda, S \sqcup q, S' \sqcup p^\perp, S' \sqcup q^-; Q](t)} (in^-)
\]

\[
\frac{DL[\lambda, S \sqcup p; Q](t) \quad p \subseteq q}{DL[\lambda, S \sqcup q, S' \sqcup p^\perp, S' \sqcup q^-; Q](t)} (in^-)
\]
\[ \Pi = \{ (1) \text{so}(ch, \text{chile}). \]

(2) \text{vi}(X) \leftarrow \text{ex}(X).

(3) \text{sw}(X) \leftarrow \text{ex}(X), \text{not } bi(X).

(4) \text{ex}(X) \leftarrow \text{so}(X, Y).

(5) no(X) \leftarrow DL[H \cup vi, H \cup sw, A \cap ex; \neg A](X).

\begin{enumerate}
    \item Cherimoya (ch) is a Southern fruit (so) from Chile;
    \item All exotic fruits (ex) are vitaminized (vi);
    \item Any exotic fruit is sweet (sw) unless it is known to be bitter (bi);
    \item All Southern fruits are exotic;
    \item \( H \) is healthy, \( A \) is African, \( no \) is nonafrican.
\end{enumerate}
Example

\[ \Pi = \{ (1) \; so(ch, \text{chile}). \]

(2) \( vi(X) \leftarrow ex(X). \)

(3) \( sw(X) \leftarrow ex(X), \text{not bi}(X). \)

(4) \( ex(X) \leftarrow so(X, Y). \)

(5) \( no(X) \leftarrow DL[H \cup vi, H \cup sw, A \cap ex; \neg A](X). \)

(1) \( ch \) is a Southern fruit \( so \) from Chile;

(2) All exotic fruits \( ex \) are vitaminized \( vi \);

(3) Any exotic fruit is sweet \( sw \) unless it is known to be bitter \( bi \);

(4) All Southern fruits are exotic;

(5) \( H \) is healthy, \( A \) is African, \( no \) is nonafrican.
Example

\[ \Pi = \{(1) \; so(ch, \text{chile}). \]
\[ (2) \; vi(X) \leftarrow ex(X). \]
\[ (3) \; sw(X) \leftarrow ex(X), \text{not bi}(X). \]
\[ (4) \; ex(X) \leftarrow so(X, Y). \]
\[ (5) \; no(X) \leftarrow DL[H \cup vi, H \cup sw, A \cap ex; \neg A](X). \]

(1) Cherimoya (ch) is a Southern fruit (so) from Chile;

(2) All exotic fruits (ex) are vitaminized (vi);

(3) Any exotic fruit is sweet (sw) unless it is known to be bitter (bi);

(4) All Southern fruits are exotic;

(5) \( H \) is healthy, \( A \) is African, \( no \) is nonafrican.

Is \( a = DL[H \cup vi, H \cup sw, A \cap ex; \neg A](ch) \) tautologic?
Example (cont.)

Is \( a = DL[H \cup vi, H \cup sw, A \cup ex; \neg A](ch) \) tautologic?

\[
\begin{align*}
&DL[H \cup ex, H \cup ex, A \cup ex; \neg A](ch) \\
&\quad \quad \text{ex} \subseteq vi \\
&\quad \quad (i_2) \\
&DL[H \cup vi, H \cup ex, A \cup ex; \neg A](ch) \\
&\quad \quad \text{ex} \subseteq sw \\
&\quad \quad (i_1) \\
&DL[H \cup vi, H \cup sw, A \cup ex; \neg A](ch)
\end{align*}
\]
Example (cont.)

Is $a = DL[H \uplus vi, H \uplus sw, A \cap ex; \neg A](ch)$ tautologic?  Yes, it is!

\[
\begin{align*}
&DL[H \uplus ex, H \uplus ex, A \cap ex; \neg A](ch) \\
\quad &\quad DL[H \uplus ex, H \uplus ex, A \cap ex; \neg A](ch) \quad ex \subseteq vi \\
\quad &\quad \quad DL[H \uplus vi, H \uplus ex, A \cap ex; \neg A](ch) \quad ex \subseteq sw \\
\quad &\quad \quad \quad DL[H \uplus vi, H \uplus sw, A \cap ex; \neg A](ch)
\end{align*}
\]

$DL[H \uplus ex, H \uplus ex, A \cap ex; \neg A](ch)$ is an axiom $a_2$ of $\mathcal{K}_{\text{taut}}$. 