Towards Practical Deletion Repair of Inconsistent DL-programs

Thomas Eiter    Michael Fink    Daria Stepanova

Knowledge-Based Systems Group,
Institute of Information Systems,
Vienna University of Technology
http://www.kr.tuwien.ac.at/

DL workshop 2014 –July 18, 2014
**Motivation**

- **DL-program**: consistent ontology $\mathcal{O}$ + rules $\mathcal{P}$ (loose coupling combination approach)
- DL-atoms serve as query interfaces to $\mathcal{O}$
- Possibility to add information from $\mathcal{P}$ to $\mathcal{O}$ prior to querying it allows for bidirectional information flow
Motivation

- **DL-program**: consistent ontology $\mathcal{O} + \text{rules } \mathcal{P}$ (loose coupling combination approach)
- **DL-atoms** serve as query interfaces to $\mathcal{O}$
- Possibility to add information from $\mathcal{P}$ to $\mathcal{O}$ prior to querying it allows for bidirectional information flow

However, information exchange between $\mathcal{P}$ and $\mathcal{O}$ can cause **inconsistency** of the DL-program (absence of answer sets).

![Diagram of DL-program](image)

[Eiter et al, *IJCAI’2013*] Repair answer sets and algorithm for repairing ontology data part, but the latter **lacks practicality**.
Motivation

- **DL-program**: consistent ontology $\mathcal{O} +$ rules $\mathcal{P}$ (loose coupling combination approach)
- DL-atoms serve as query interfaces to $\mathcal{O}$
- Possibility to add information from $\mathcal{P}$ to $\mathcal{O}$ prior to querying it allows for bidirectional information flow

However, information exchange between $\mathcal{P}$ and $\mathcal{O}$ can cause **inconsistency** of the DL-program (absence of answer sets).

![DL-program diagram]

[Etier et al., *IJCAI*’2013] Repair answer sets and algorithm for repairing ontology data part, but the latter **lacks practicality**.

**In this work**: Algorithm for DL-program repair based on **support sets** for DL-atoms. Effective for ontologies in $DL$-$Lite_{\lambda}$. 
Overview

Motivation

DL-programs

Support Sets for DL-atoms

Repair Answer Set Computation

Experiments

Conclusion
**DL-Lite\(_A\)**

- Lightweight Description Logic for accessing large data sources
- Concepts and roles model sets of objects and their relationships

\[
C \rightarrow A \mid \exists R \quad R \rightarrow P \mid P^-
\]

- A \(DL-Lite\(_A\)\) ontology \(\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle\) consists of:
  - **TBox** \(\mathcal{T}\) specifying constraints at the conceptual level
    \[
    C_1 \sqsubseteq C_2, \quad C_1 \sqsubseteq \neg C_2, \\
    R_1 \sqsubseteq R_2, \quad R_1 \sqsubseteq \neg R_2, \quad (\text{funct } R)
    \]
  - **ABox** \(\mathcal{A}\) specifying the facts that hold in the domain
    \[
    A(b) \quad P(a, b)
    \]
**DL-Lite**

- Lightweight Description Logic for accessing large data sources
- Concepts and roles model sets of objects and their relationships

\[
C \rightarrow A \mid \exists R \quad R \rightarrow P \mid P^-
\]

- A **DL-Lite** ontology \( \mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle \) consists of:
  - **TBox** \( \mathcal{T} \) specifying constraints at the conceptual level
    \[
    C_1 \sqsubseteq C_2, \quad C_1 \sqsubseteq \neg C_2, \\
    R_1 \sqsubseteq R_2, \quad R_1 \sqsubseteq \neg R_2, \quad (\text{funct } R)
    \]
  - **ABox** \( \mathcal{A} \) specifying the facts that hold in the domain
    \[
    A(b) \quad P(a, b)
    \]

**Example**

\[
\mathcal{T} = \left\{ \begin{array}{l}
\text{Child} \sqsubseteq \exists \text{hasParent} \\
\text{Female} \sqsubseteq \neg \text{Male}
\end{array} \right\} \\
\mathcal{A} = \left\{ \begin{array}{l}
\text{hasParent}(\text{john, pat}) \\
\text{Male}(\text{john})
\end{array} \right\}
\]
**DL-Lite\(\mathcal{A}\)**

- Lightweight Description Logic for accessing large data sources
- Concepts and roles model sets of objects and their relationships

\[
C \rightarrow A \mid \exists R \quad R \rightarrow P \mid P^{-}
\]

- 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
    \[
    C_1 \sqsubseteq C_2, \quad C_1 \sqsubseteq \neg C_2, \\
    R_1 \sqsubseteq R_2, \quad R_1 \sqsubseteq \neg R_2, \quad (\text{funct } R)
    \]
  - **ABox** \(\mathcal{A}\) specifying the facts that hold in the domain
    \[
    A(b) \quad P(a, b)
    \]

- For query derivation: **single** ABox assertion
- For inconsistency: at most **two** ABox assertions
- Classification is **tractable**

[Calvanese *et al.*, 2007]
Example: DL-program

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \text{ is a DL-program} \]

\[ \mathcal{O} = \{ (1) \text{Child} \sqsubseteq \exists \text{hasParent} \quad (4) \text{Male}(\text{pat}) \\
(2) \text{Adopted} \sqsubseteq \text{Child} \quad (5) \text{Male}(\text{john}) \\
(3) \text{Female} \sqsubseteq \neg \text{Male} \quad (6) \text{hasParent}(\text{john}, \text{pat}) \} \]
Example: DL-program

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \text{ is a DL-program} \]

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

\[ \mathcal{P} = \{ \begin{array}{ll}
(7) \text{ischildof}(john, alex); & (8) \text{boy}(john); \\
(9) \text{hasfather}(john, pat) \leftarrow \text{DL}[\text{Male} \cup \text{boy}; \text{Male}](pat), \\
& \text{DL}[; \text{hasParent}](john, pat) \\
\end{array} \] \]
Example: DL-program

\[ \Pi = \langle \mathcal{O}, \mathcal{P} \rangle \text{ is a DL-program} \]

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

\[ (4) \text{Male}(\text{pat}) \]

\[ (5) \text{Male}(\text{john}) \]

\[ (6) \text{hasParent}(\text{john}, \text{pat}) \]

\[ \mathcal{P} = \left\{ \begin{array}{ll}
(7) & \text{ischildof}(\text{john}, \text{alex}); \\
(8) & \text{boy}(\text{john}); \\
(9) & \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \cup \text{boy}; \text{Male}](\text{pat}), \\
& \text{DL}[; \text{hasParent}](\text{john}, \text{pat}) \\
\end{array} \right\} \]

- **Interpretation:** \( I = \{ \text{ischildof}(\text{john}, \text{alex}), \text{boy}(\text{john}), \text{hasfather}(\text{john}, \text{pat}) \} \)

- **Satisfaction relation:** \( I \models^\mathcal{O} \text{boy}(\text{john}); I \models^\mathcal{O} \text{DL}[; \text{hasParent}](\text{john}, \text{pat}) \)

- **Semantics:** in terms of answer sets, i.e. founded models (weak, flp, . . .)

- \( I \) is a weak and flp answer set
Example: Inconsistent DL-program

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

\[ \mathcal{O} = \{ \]

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

\[ \mathcal{P} = \{ \]

\( (4) \) \text{ Male}(\text{pat}) \]
\( (5) \) \text{ Male}(\text{john}) \]
\( (6) \) \text{ hasParent}(\text{john, pat}) \]
\( (7) \) \text{ ischildof}(\text{john, alex}); \]
\( (8) \) \text{ boy}(\text{john}); \]
\( (9) \) \text{ hasfather}(\text{john, pat}) \leftarrow \text{DL}[\text{Male} \cup \text{ boy}; \text{ Male}](\text{pat}), \]
\( \text{DL}[; \text{ hasParent}](\text{john, pat}); \]
\( (10) \) \text{ not DL}[; \text{ Adopted}](\text{john}), \text{ pat} \neq \text{ alex}, \]
\( \text{ hasfather}(\text{john, pat}), \text{ ischildof}(\text{john, alex}), \]
\( \text{ not DL}[\text{ Child} \cup \text{ boy}; \neg \text{ Male}](\text{alex}) \]
Example: Inconsistent DL-program

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

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

\[ \mathcal{P} = \left\{ \begin{array}{ll}
(4) & \text{Male}(\text{pat}) \\
(5) & \text{Male}(\text{john}) \\
(6) & \text{hasParent}(\text{john}, \text{pat}) \\
(7) & \text{ischildof}(\text{john}, \text{alex}) ; \\
(8) & \text{boy}(\text{john}) ; \\
(9) & \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \sqcup \text{boy}; \text{Male}](\text{pat}), \\
& \text{DL}[; \text{hasParent}](\text{john}, \text{pat}) ; \\
(10) & \bot \leftarrow \text{not DL}[; \text{Adopted}](\text{john}), \text{pat} \neq \text{alex}, \\
& \text{hasfather}(\text{john}, \text{pat}), \text{ischildof}(\text{john}, \text{alex}), \\
& \text{not DL}[\text{Child} \sqcup \text{boy}; \neg \text{Male}](\text{alex})
\end{array} \right\} \]
Example: Inconsistent DL-program

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

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

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

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

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

\[ \mathcal{P} = \left\{ \begin{array}{l}
(4) \text{Male}(\text{pat}) \\
(5) \text{Male}(\text{john}) \\
(6) \text{hasParent}(\text{john}, \text{pat}) \\
(7) \text{ischildof}(\text{john}, \text{alex}); \\
(8) \text{boy}(\text{john}); \\
(9) \text{hasfather}(\text{john}, \text{pat}) \leftarrow \text{DL}[\text{Male} \uplus \text{boy}; \text{Male}](\text{pat}), \\
\hspace{1cm} \text{DL}[; \text{hasParent}](\text{john}, \text{pat}); \\
(10) \bot \leftarrow \neg \text{DL}[; \text{Adopted}](\text{john}), \text{pat} \neq \text{alex}, \\
\hspace{1cm} \text{hasfather}(\text{john}, \text{pat}), \text{ischildof}(\text{john}, \text{alex}), \\
\hspace{1cm} \neg \text{DL}[\text{Child} \uplus \text{boy}; \neg \text{Male}](\text{alex}).
\end{array} \right. \]

No answer sets
Example: Inconsistent DL-program

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

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

\[ \mathcal{P} = \{ \]
\( (7) \) ischildof(john, alex); (8) boy(john); (9) hasfather(john, pat) \( \leftarrow \) DL[Male \( \sqcup \) boy; Male](pat), DL[; hasParent](john, pat); (10) \( \perp \leftarrow \) not DL[; Adopted](john), pat \( \neq \) alex, hasfather(john, pat), ischildof(john, alex), not DL[Child \( \sqcup \) boy; \( \neg \) Male](alex) \)
\[ \} \]

\[ I_1 = \{ \text{ischildof}(john, alex), \text{boy}(john) \} \]
Example: Inconsistent DL-program

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

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

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

\[ I_1 = \{ \text{ischildof}(\text{john}, \text{alex}), \text{boy}(\text{john}) \} \]
Ground Support Sets

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

When is \( d \) true under interpretation \( I \)?
Ground Support Sets

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

When is \( d \) true under interpretation \( I \)?

- \( Male(pat) \in A \)
- \( boy(pat) \in I \)
- \( boy(alex) \in I; Female(alex) \in A \)
Ground Support Sets

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

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 \( A_d = \{P_p(t) \mid P \uplus p \in \lambda\} \cup \{\neg P_p(t) \mid P \uplus p \in \lambda\} \)
Ground Support Sets

Definition

$S \subseteq \mathcal{A} \cup \mathcal{A}_d$ is a support set for $d = DL[\lambda; Q](t)$ w.r.t. $\mathcal{O} = \langle T, \mathcal{A} \rangle$ 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.

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

d = DL[Male \uplus boy; Male](pat); T_d = \{Female \sqsubseteq \neg Male; Male_{boy} \sqsubseteq Male\}

Support sets:

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

Definition

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

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

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

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

$I \models^\mathcal{O} d$ iff there exists $S \in \text{Supp}_\mathcal{O}(d)$, which is coherent with $I$. 
Nonground Support Sets

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

Support sets:

- \( S_1 = \{\text{Male}(\text{pat})\} \)
- \( S_2 = \{\text{Male}_{\text{boy}}(\text{pat})\} \)
- \( S_3 = \{\text{Male}_{\text{boy}}(c); \text{Female}(c)\} \quad c \in \mathcal{C} \)
Nonground Support Sets

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

Nonground support sets:

- \( S_1 = \{\text{Male}(X)\} \)
- \( S_2 = \{\text{Male}_{\text{boy}}(X)\} \)
- \( S_3 = \{\text{Male}_{\text{boy}}(Y); \text{Female}(Y)\} \)
Nonground Support Sets

Definition

\( S = \{P(Y), P'(Y')\} \) \((S = \{P(Y)\})\) is 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 \(C\).

\[\]

\(d = \text{DL}[\text{Male} \sqcup \text{boy}; \text{Male}](X), \quad T_d = \{\text{Female} \sqsubseteq \lnot\text{Male}; \ \text{Male}_{\text{boy}} \sqsubseteq \text{Male}\}\]

Nonground support sets:

- \(S_1 = \{\text{Male}(X)\}\)
- \(S_2 = \{\text{Male}_{\text{boy}}(X)\}\)
- \(S_3 = \{\text{Male}_{\text{boy}}(Y); \ \text{Female}(Y)\}\)
Nonground Support Sets

Definition

\( S = \{P(Y), P'(Y')\} \) (\( S = \{P(Y)\} \)) is 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

Definition

\[ S = \{ P(Y), P'(Y') \} \] \((S = \{ P(Y) \})\) is 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 \(S \in \text{Supp}_O(d(X\theta))\) some \(S' \in S\) exists, s.t. \(S = S'\theta'\).

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

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

- Construct \( \mathcal{T}_d \):

- Compute classification \( Cl(\mathcal{T}_d) \) (e.g. using ASP techniques):

- Extract support sets from \( Cl(\mathcal{T}_d) \):
Nonround Support Set Computation

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

- **Construct** \( \mathcal{T}_d \):
  \[ \mathcal{T}_d = \mathcal{T} \cup \{\text{Male}_{\text{boy}} \sqsubseteq \text{Male}\} \]

- **Compute classification** \( Cl(\mathcal{T}_d) \) (e.g. using ASP techniques):

- **Extract support sets from** \( Cl(\mathcal{T}_d) \):
  \[
  \begin{align*}
  S_1 &= \{\text{Male}(X)\} \\
  S_2 &= \{\text{Male}_{\text{boy}}(X)\} \\
  S_3 &= \{\text{Male}_{\text{boy}}(Y), \neg \text{Male}(Y)\} \\
  S_4 &= \{\text{Male}_{\text{boy}}(Y), \text{Female}(Y)\} \\
  S_5 &= \{\text{Male}(Y), \neg \text{Male}(Y)\} \\
  S_6 &= \{\text{Male}(Y), \text{Female}(Y)\}
  \end{align*}
  \]
  \[ \{S_1, S_2, S_3, S_4\} \text{ is complete!} \]
  \[ \{S_5, S_6\} \text{ is consistent!} \]
**Nonround Support Set Computation**

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

- **Construct** \( T_d \):
  \[ T_d = T \cup \{\text{Male}_{\text{boy}} \sqsubseteq \text{Male}\} \]

- **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** \( Cl(T_d) \):
Nonround Support Set Computation

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

- **Construct** \( \mathcal{T}_d \):

  \[
  \mathcal{T}_d = \mathcal{T} \cup \{Male_{boy} \sqsubseteq Male\}
  \]

- **Compute classification** \( Cl(\mathcal{T}_d) \) (e.g. using ASP techniques):

  \[
  cl(\mathcal{T}_d) = \mathcal{T}_d \cup \{Male \sqsubseteq \neg Female; Male_{boy} \sqsubseteq \neg Female\} \cup \{P \sqsubseteq P | P \in \mathcal{P}\}
  \]

- **Extract support sets from** \( Cl(\mathcal{T}_d) \):

  - \( S_1 = \{Male(X)\} \)
  - \( S_2 = \{Male_{boy}(X)\} \)
  - \( S_3 = \{Male_{boy}(Y), \neg Male(Y)\} \)
  - \( S_4 = \{Male_{boy}(Y), Female(Y)\} \)
  - \( S_5 = \{Male(Y), \neg Male(Y)\} \)
  - \( S_6 = \{Male(Y), Female(Y)\} \)
Nonround Support Set Computation

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

- **Construct** \( \mathcal{T}_d \):
  \[ \mathcal{T}_d = \mathcal{T} \cup \{Male_{boy} \sqsubseteq Male\} \]

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

- **Extract support sets from** \( Cl(\mathcal{T}_d) \):
  - \( S_1 = \{Male(X)\} \)
  - \( S_2 = \{Male_{boy}(X)\} \)
  - \( S_3 = \{Male_{boy}(Y), \neg Male(Y)\} \)
  - \( S_4 = \{Male_{boy}(Y), Female(Y)\} \)
  - \( S_5 = \{Male(Y), \neg Male(Y)\} \)
  - \( S_6 = \{Male(Y), Female(Y)\} \)
Nonround Support Set Computation

d = DL[Male ∪ boy; Male](X); T = \{Female ⊑ ¬Male\}

• Construct \( T_d \):
  \[ T_d = T \cup \{Male_{boy} \sqcap Male\} \]

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

• Extract support sets from \( Cl(T_d) \):
  \[
  \begin{align*}
  S_1 &= \{Male(X)\} \\
  S_2 &= \{Male_{boy}(X)\} \\
  S_3 &= \{Male_{boy}(Y), ¬Male(Y)\} \\
  S_4 &= \{Male_{boy}(Y), Female(Y)\} \\
  S_5 &= \{Male(Y), ¬Male(Y)\} \\
  S_6 &= \{Male(Y), Female(Y)\}
  \end{align*}
  \]

\( \mathcal{O} \) is consistent!
Nonround Support Set Computation

d = DL[Male ∪ boy; Male](X); T = {Female ⊑ ¬Male}

• Construct $T_d$:
  \[ T_d = T \cup \{ Male_{\text{boy}} \sqsubseteq Male \} \]

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

• Extract support sets from $Cl(T_d)$:
  \[
  \begin{align*}
    S_1 &= \{ Male(X) \} \\
    S_2 &= \{ Male_{\text{boy}}(X) \} \\
    S_3 &= \{ Male_{\text{boy}}(Y), \neg Male(Y) \} \\
    S_4 &= \{ Male_{\text{boy}}(Y), Female(Y) \}
  \end{align*}
  \]
  \{ S_1, S_2, S_3, S_4 \} is complete!
Repair Answer Set Computation

✓ 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. $\Pi' = \langle O', P \rangle$, $O' = \langle T, A' \rangle$
Repair Answer Set Computation

**Algorithm 1: SupRAAnsSet:** all deletion repair answer sets

**Input:** $\Pi=\langle T \cup A, P \rangle$

**Output:** $flipRAS(\Pi)$

(a) compute a complete set $S$ of nongr. supp. sets for the DL-atoms in $\Pi$

(b) for $\hat{I} \in AS(\hat{\Pi})$ do

(c) if $S_{gr}^{\hat{I}}(a) \neq \emptyset$ for $a \in D_p$ and every $S \in S_{gr}^{\hat{I}}(a)$ for $a \in D_n$ fulfills $S \cap A \neq \emptyset$ then

(d) for all $a \in D_p$ do

(e) if some $S \in S_{gr}^{\hat{I}}(a)$ exists s.t. $S \cap A = \emptyset$ then pick next $a$

(e) else remove each $S$ from $S_{gr}^{\hat{I}}(a)$ s.t. $S \cap A \cap \bigcup_{a' \in D_n} S_{gr}^{\hat{I}}(a') \neq \emptyset$

(f) if $S_{gr}^{\hat{I}}(a) = \emptyset$ then pick next $\hat{I}$

end

(g) $A' \leftarrow A \setminus \bigcup_{a' \in D_n} S_{gr}^{\hat{I}}(a')$

(h) if $flipFND(\hat{I}, \langle T \cup A', P \rangle)$ then output $\hat{I}|_{\Pi}$

end
Algorithm 1: $SupRA\text{nsSet}$: all deletion repair answer sets

Input: $\Pi = \langle T \cup A, P \rangle$

Output: $flipRAS(\Pi)$

(a) compute a complete set $S$ of nongr. supp. sets for the DL-atoms in $\Pi$

(b) for $\hat{I} \in AS(\hat{\Pi})$ do

SupRA\text{nsSet} is sound and complete wrt. deletion repair answer sets.

(e) if some $S \in S^i_{gr}(a)$ exists s.t. $S \cap A = \emptyset$ then pick next $a$

else remove each $S$ from $S^i_{gr}(a)$ s.t. $S \cap A \cap \bigcup_{a' \in D_n} S^i_{gr}(a') \neq \emptyset$

(f) if $S^i_{gr}(a) = \emptyset$ then pick next $\hat{I}$

end

(g) $A' \leftarrow A \setminus \bigcup_{a' \in D_n} S^i_{gr}(a')$

(h) if $\text{flipFND}(\hat{I}, \langle T \cup A', P \rangle)$ then output $\hat{I}|_\Pi$

end
Experiments

Motivation
Overview
DL-programs
Support Sets for DL-atoms
Repair Answer Set Computation
Experiments
Conclusion

Experiments

- $A_{50}$ AS
- $A_{50}$ rep
- $A_{1000}$ AS
- $A_{1000}$ rep

- $P_{con}$ AS
- $P_{con}$ rep
- $P_{guess}$ AS
- $P_{guess}$ rep
Related Work

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

- Support sets in other works
  - Support sets for $\text{HEX}$-programs [Eiter et al, AAAI’2014] as more abstract structures
Conclusion and Future Work

Conclusions:

- Ground and nonground support sets for DL-atoms
  - Allow evaluation of DL-atoms avoiding ontology access
- Support sets for $DL-Lite_A$ are small and efficiently computable
- Effective sound and complete algorithm $SupRAnsSet$ for deletion repair computation based on support sets
- Implementation in DLVHEX and evaluation on a set of benchmarks

Further and future work:

- Extensions to other DLs (e.g. $\mathcal{EL}$)
- Computing preferred repairs (e.g. $\sigma$-selection [Eiter et al, IJCAI’2013])
Meghyn Bienvenu. 
On the complexity of consistent query answering in the presence of simple ontologies. 

Diego Calvanese, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 
Tractable reasoning and efficient query answering in description logics: The DL-Lite family. 
Diego Calvanese, Magdalena Ortiz, Mantas Simkus, and Giorgio Stefanoni.

Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo.
DL-program: syntax

Signature: $\Sigma = \langle C, I, P, C, R \rangle$, where
- $\Sigma_0 = \langle I, C, R \rangle$ is a DL signature;
- $C \supseteq I$ is a set of constant symbols;
- $P$ is a finite set of predicate symbols of arity $\geq 0$, s.t. $P \cap \{C \cup R\} = \emptyset$.

DL-atom is of the form $DL[S_1 o_1 p_1, \ldots, S_m o_m p_m; Q](t)$, $m \geq 0$, where
- $S_i \in C \cup R$;
- $o_i \in \{\lor, \cup, \land\}$;
- $p_i \in P$ (unary or binary);
- $Q(t)$ is a DL-query:
  - $C(t_1), \neg C(t_1), t = t_1$, where $C \in C$;
  - $R(t_1, t_2), \neg R(t_1, t_2), t = t_1, t_2$, where $R \in R$.
- $C \sqsubseteq D, C \not\sqsubseteq D, t = \epsilon$, where $C, D \in C \cup \{T, \bot\}$;

DL-program: $\Pi = \langle \mathcal{O}, P \rangle$, $\mathcal{O}$ is a DL ontology, $P$ is a set of DL-rules:

$$a_1 \lor \ldots \lor a_n \leftarrow b_1, \ldots b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m,$$

$m \geq k \geq 0$, $a_i$ is a classical literal; $b_i$ is a classical literal or a DL-atom.
Consider grounding $\text{grd}(\Pi) = \langle O, \text{grd}(P) \rangle$ of $\Pi = \langle O, P \rangle$ over $C$ and $P$.

Interpretation $I$ is a consistent set of ground literals over $C$ and $P$.

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

$$I \models O a$$

iff $\tau(\langle T, A \cup \lambda^I(a) \rangle) \models Q(c)$, where $\tau(O)$ is a modular translation of $O$ to FOL, $\lambda^I(a) = \bigcup_{i=1}^m A_i(I)$ is a DL-update of $O$ 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 $op_i = \cup$;
- $A_i(I) = \{ \neg S_i(t) \mid p_i(t) \notin I \}$, for $\cap$.

FLP-reduct $\rho_{flp} P^I$ of $P$ is a set of ground DL-rules $r$ s.t. $I \models b^+(r)$, $I \nvdash b^-(r)$.

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

$I$ is an $x$-answer set of $P$ iff $I$ is a minimal model of its $x$-reduct.
Network Benchmark

\[ O = \begin{cases} 
(1) \exists \text{forbid} \sqsubseteq \text{Block} & (4) \text{edge}(n_i, n_j) \\
(2) \text{Broken} \sqsubseteq \text{Block} & (5) \ldots \\
(3) \text{Block} \sqsubseteq \neg \text{Avail} & (6) \ldots 
\end{cases} \]

\[ P_{\text{guess}} = \begin{cases} 
(1) \text{go}(X, Y) \leftarrow \text{open}(X), \text{open}(Y), \text{DL}[; \text{edge}](X, Y). \\
(2) \text{route}(X, Z) \leftarrow \text{route}(X, Y), \text{route}(Y, Z). \\
(3) \text{route}(X, Y) \leftarrow \neg \text{DL}[\text{Block} \sqcup \text{block}; \text{forbid}](X, Y), \text{go}(X, Y). \\
(4) \text{open}(X) \lor \text{block}(X) \leftarrow \neg \text{DL}[; \neg \text{Avail}](X), \text{node}(X). \\
(5) \text{negls}(X) \leftarrow \text{node}(X), \text{route}(X, Y), X \neq Y. \\
(6) \bot \leftarrow \text{node}(X), \neg \text{negls}(X). 
\end{cases} \]
Network Benchmark

$\mathcal{O} = \{ (1) \exists forbid \sqsubseteq Block \quad (4) \text{edge}(n_i, n_j) \\
(2) \text{Broken} \sqsubseteq Block \quad (5) \ldots \\
(3) \text{Block} \sqsubseteq \neg \text{Avail} \quad (6) \ldots \}$

$\mathcal{P}_{\text{con}} = \left\{ \begin{array}{l}
(1) \text{go}(X, Y) \leftarrow \text{open}(X), \text{open}(Y), \text{DL}[; \text{edge}](X, Y).
(2) \text{route}(X, Z) \leftarrow \text{route}(X, Y), \text{route}(Y, Z).
(3') \text{route}(X, Y) \leftarrow \text{go}(X, Y), \neg \text{DL}[; \neg \text{forbid}](X, Y).
(4') \text{open}(X) \leftarrow \text{node}(X), \neg \text{DL}[; \neg \text{Avail}](X).
(5) \text{negls}(X) \leftarrow \text{node}(X), \text{route}(X, Y), X \neq Y.
(6') \bot \leftarrow \text{in}(X), \text{out}(Y), \neg \text{route}(X, Y). \end{array} \right\}$