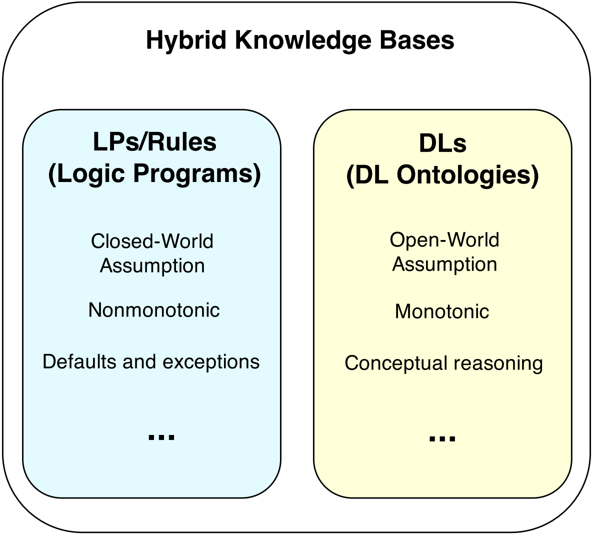
Inconsistencies in Hybrid Knowledge Bases
PhD Defense

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

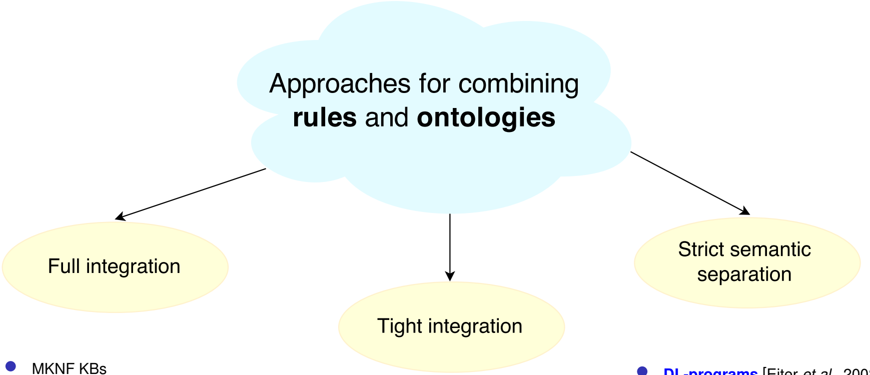
Doctoral School "Mathematical Logic in Computer Science",
Institute of Information Systems,
Vienna University of Technology

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
...

Hybrid Knowledge Bases



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

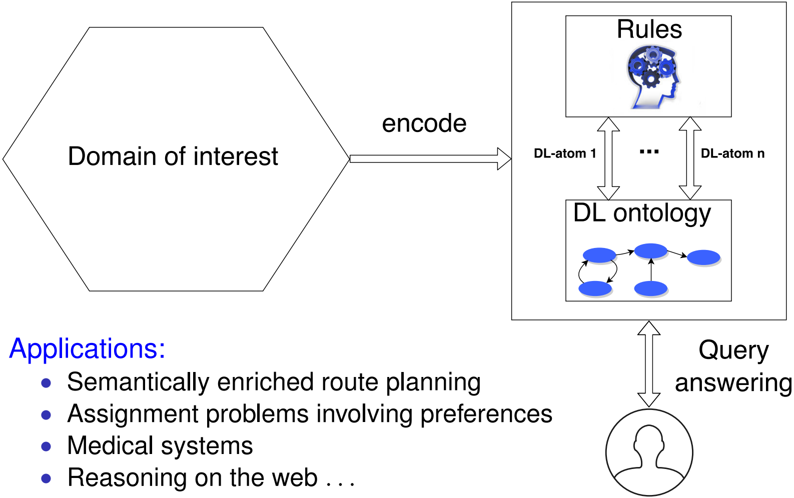
- Carin [Levy and Rousset, 1998]
- DL-safe rules [Motik *et al.*, 2005]
- R-hybrid KBs [Rosati, 2005]
- $\mathcal{DL}$+LOG [Rosati, 2006]

Strict semantic separation

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

DL-programs

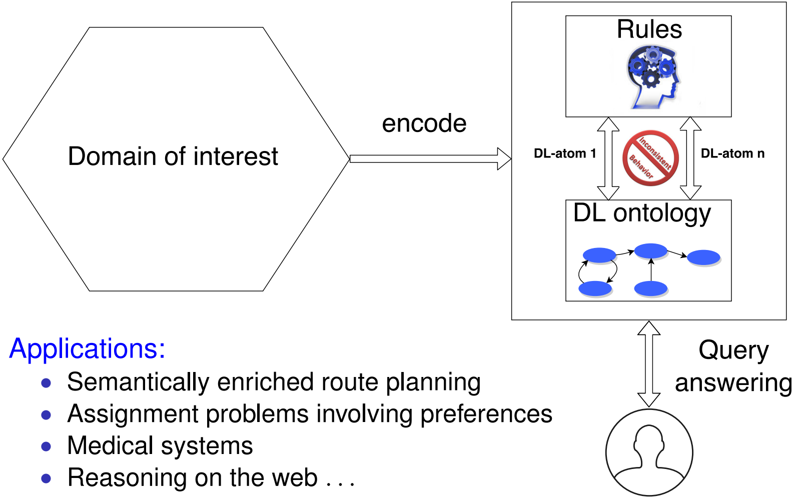
- DL-programs: Rules + Ontology (loose coupling combination)



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

DL-programs

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

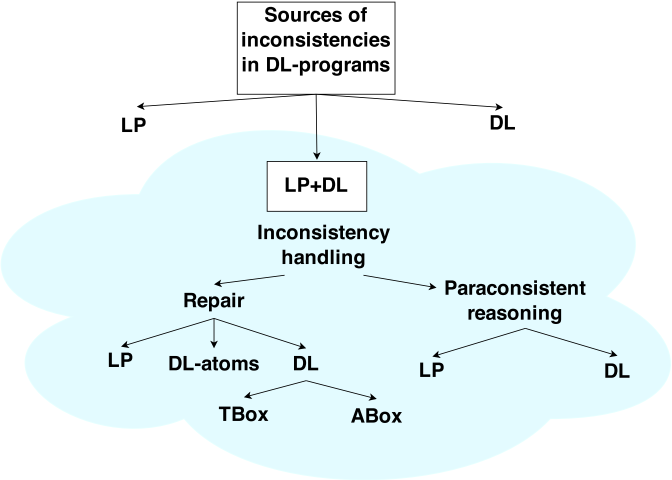


- Applications:
  - Semantically enriched route planning
  - Assignment problems involving preferences
  - Medical systems
  - Reasoning on the web . . .
- **Problem:** inconsistencies often arise as a result of combination

Inconsistency in DL-programs

Problem: inconsistency in a DL-program
Question: how to deal with it?

Many possibilities..



Overview

Description Logic Ontologies

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

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

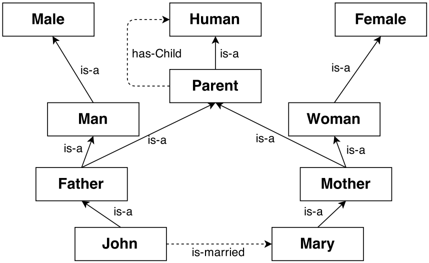
Description Logic Ontologies

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

$$\forall X(\mathit{Female}(X) \wedge \exists Y\,(\mathit{hasChild}(X,Y)) \rightarrow \mathit{Mother}(X))$$

Description Logic Ontologies

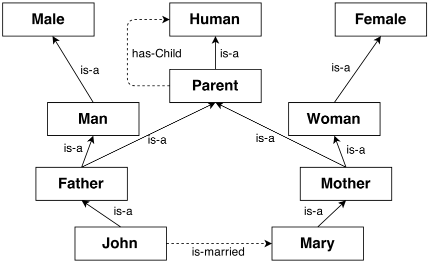
- 1950's-1960's: First Order Logic (FOL) for KR (undecidable) (e.g. [McCarthy, 1959])
- 1970's: Network-shaped structures for KR (e.g. semantic networks [Quillan, 1967], frames [Minsky, 1985])



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])

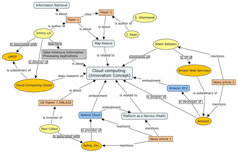


Description Logic Ontologies

- 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 [Quillan, 1967], frames [Minsky, 1985])
- 1979: Encoding of frames into FOL [Hayes, 1979]



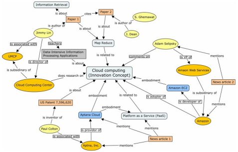
- 1980's: Description Logics (DL) for KR
  - Decidable fragments of FOL
  - Theories encoded in DLs are called ontologies
  - Many DLs with different expressiveness and computational features

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 ($\mathit{DL\text{-}Lite}_{\mathcal{A}}, \mathcal{EL}$)

Description Logic *DL-Lite*$_{\mathcal{A}}$

- Concepts model sets of objects and roles model binary relations
  - *Child*, *hasParent*

Description Logic $DL\text{-}Lite_{\mathcal{A}}$

- Concepts model sets of objects and roles model binary relations
- More complex concepts and roles can be constructed:

Construct	Syntax	Example
negated concept	$\neg C$	$\neg Male$
exist. on roles	$\exists R$	$\exists hasChild$
negated roles	$\neg R$	$\neg hasSibling$
role inverses	R^{-}	$hasParent^{-}$

Description Logic $DL\text{-}Lite_{\mathcal{A}}$

- Concepts model sets of objects and roles model binary relations
- More complex concepts and roles can be constructed:

$$C \to A \mid \exists R \qquad B \to C \mid \neg C$$
$$R \to U \mid U^{-} \qquad S \to R \mid \neg R$$

- A $DL\text{-}Lite_{\mathcal{A}}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of:
  - TBox $\mathcal{T}$ specifying constraints at the conceptual level.

    $$C \sqsubseteq B \qquad R \sqsubseteq S \qquad (\mathit{funct}\ R)$$

  - ABox $\mathcal{A}$ specifying facts that hold in the domain.

    $$A(b) \qquad P(a, b)$$

Description Logic $DL\text{-}Lite_{\mathcal{A}}$

- Concepts model sets of objects and roles model binary relations
- More complex concepts and roles can be constructed:

$$C \rightarrow A \mid \exists R \qquad B \rightarrow C \mid \neg C$$
$$R \rightarrow U \mid U^{-} \qquad S \rightarrow R \mid \neg R$$

- A $DL\text{-}Lite_{\mathcal{A}}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of:
  - TBox $\mathcal{T}$ specifying constraints at the conceptual level.

    $$C \sqsubseteq B \qquad R \sqsubseteq S \qquad (funct\ R)$$

  - ABox $\mathcal{A}$ specifying facts that hold in the domain.

    $$A(b) \qquad P(a, b)$$

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

$$\mathcal{T} = \{\ Child \sqsubseteq \exists hasParent \qquad Female \sqsubseteq \neg Male\}$$
$$\mathcal{A} = \{\ hasParent(john, pat) \qquad Male(john)\}$$

Description Logic $\mathcal{EL}$

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

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

$$\mathcal{A} = \left\{ \begin{array}{ll} Female(ann) & hasSibling(ann, pat) \\ Human(john) & hasChild(pat, john) \end{array} \right\}$$

- $\mathcal{EL}$-concepts:

Construct	Syntax	Example
Conjunction	$A \sqcap B$	$Female \sqcap Child$
Exist. restr.	$\exists R.A$	$\exists hasSibling.Male$

- TBox axioms[1]:

$$C \sqsubseteq D \qquad\qquad C \equiv D$$

[1] C and D are arbitrarily complex concepts constructed using $\exists$ and $\sqcap$

$DL\text{-}Lite_{\mathcal{A}}$ and $\mathcal{EL}$: FOL Formalization

$Child \sqsubseteq \exists hasParent$ is equiv. to $\forall x(Child(x) \rightarrow \exists y(hasParent(x, y)))$

Syntax	FOL formalization
$A_1 \sqsubseteq A_2$	$\forall x(A_1(x) \rightarrow A_2(x))$
$R_1 \sqsubseteq R_2$	$\forall x, y(R_1(x, y) \rightarrow R_2(x, y))$
$A_1 \sqsubseteq \neg A_2$	$\forall x(A_1(x) \rightarrow \neg A_2(X))$
$R_1 \sqsubseteq \neg R_2$	$\forall x, y(R_1(x, y) \rightarrow \neg R_2(x, y))$
$\exists R \sqsubseteq A$	$\forall x(\exists y(R(x, y)) \rightarrow A(x))$
$\exists R^- \sqsubseteq A$	$\forall x(\exists y(R(y, x)) \rightarrow A(x))$
$A \sqsubseteq \exists R$	$\forall x(A(x) \rightarrow \exists y(R(x, y)))$
$funct(R)$	$\forall x, y, y'(R(x, y) \wedge R(x, y') \rightarrow y = y')$
$A_1 \sqcap A_2 \sqsubseteq A_3$	$\forall x A_1(x) \wedge A_2(x) \rightarrow A_3(x)$
$\exists R.A_1 \sqsubseteq A_2$	$\forall x(\exists y(R(x, y) \wedge A_1(y)) \rightarrow A_2(x)$
$A_1 \sqsubseteq \exists R.A_2$	$\forall x(A(x) \rightarrow \exists y(R(x, y) \wedge A_2(y)))$
...	...

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

  $Human \sqsubseteq HeartOnLeft$
  $Human(john)$

Nonmonotonic Logic Programs

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

  $Human \sqsubseteq HeartOnLeft$
  $Human(john)$
  $\neg HeartOnLeft(john)$

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:

$$\underbrace{a_1 \vee \ldots \vee a_k}_{\text{Head (H)}} \leftarrow \underbrace{b_1, \ldots, b_m, \textit{not}\, b_{m+1}, \ldots, \textit{not}\, b_n.}_{\text{Body (B)}}$$

- a_i's and b_j's are first-order atoms and
- *not* is a negation as failure (default negation, weak negation)

Example

$$female(Y) \vee female(Z) \leftarrow \textit{not}\ adopted(X), hasparent(X, Y)$$
$$hasparent(X, Z), Y \neq Z$$

Answer Set Semantics

$$\mathcal{P} = \left\{ \begin{array}{l} \mathit{hasparent}(\mathit{john}, \mathit{pat}); \quad \mathit{hasparent}(\mathit{john}, \mathit{alex}); \\ \mathit{female}(\mathit{pat}) \vee \mathit{female}(\mathit{alex}) \leftarrow \mathit{not}\ \mathit{adopted}(\mathit{john}), \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathit{hasparent}(\mathit{john}, \mathit{pat}), \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathit{hasparent}(\mathit{john}, \mathit{alex}) \end{array} \right\}$$

- Semantics: given for ground programs (programs without variables)
- Interpretation: consistent set I of ground atoms over Herbrand Base of $\mathcal{P}$
  $I_1 = \{\mathit{hasparent}(\mathit{john}, \mathit{pat}), \mathit{hasparent}(\mathit{john}, \mathit{alex}), \mathit{female}(\mathit{alex})\}$
- Satisfaction relation: $I \models a$ iff $a \in I$
  $I_1 \models \mathit{hasparent}(\mathit{john}, \mathit{pat})$; $I_1 \not\models \mathit{adopted}(\mathit{john})$
- 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 $\mathcal{P}$ ($I \in AS(\mathcal{P})$) if it is a $\subseteq$-minimal model that allows founded model reconstruction using rules
  $I_1 \in AS(\mathcal{P})$

Answer Set Semantics

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

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

Answer Set Semantics

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

- $I_3 = \{hasparent(john, pat), hasparent(john, alex), adopted(john)\}$
  $I_3 \in AS(\mathcal{P})$
- $adopted(john)$ is added, $female(alex)/female(pat)$ are no longer derived
  Nonmonotonicity!

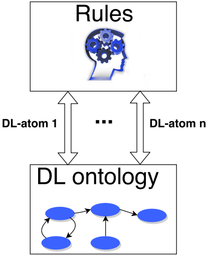
DL Ontologies vs Logic Programs

- $\neg$ in DLs is different from *not* in LP
  - $\neg$: classical negation, monotonicity, open world assumption
  - *not* : default negation, nonmonotonicity, closed world assumption

DL ontology $\mathcal{O}$	Logic Program $\mathcal{P}$
$Child \sqsubseteq Person$	$person(X) \leftarrow child(X)$
$\neg Child \sqsubseteq Adult$	$adult(X) \leftarrow not\ child(X)$
$Person(john)$	$person(john)$
$\mathcal{O} \not\models Adult(john)$	$\mathcal{P}$ infers $adult(john)$

- 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
- ...

DL-programs: syntax



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

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

  - 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.

$$\mathcal{O} = \left\{ \begin{array}{ll} (1)\ hasChild^{-} \sqsubseteq hasParent & (3)\ Male(pat) \\ (2)\ Female \sqsubseteq \neg Male & (4)\ hasChild(pat, john) \end{array} \right\}$$

$$\mathcal{P} = \left\{ \begin{array}{l} (5)\ boy(john); \\ (6)\ hasfather(john, pat) \leftarrow \mathrm{DL}[Male \uplus boy; Male](pat), \\ \qquad\qquad\qquad\qquad\qquad \mathrm{DL}[; hasParent](john, pat) \end{array} \right\}$$

DL-atoms

$\mathrm{DL}[\mathit{Male} \uplus \mathit{boy}; \mathit{Male}](\mathit{john})$

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

A DL-atom is of the form

$$\mathrm{DL}[S_1\ op_1\ p_1, \ldots, S_m\ op_m\ p_m; Q](\mathbf{t})$$

- S_i: ontology concept or role
- $op_i \in \{\uplus, \cupdot\}$: intuitively $\uplus$ (resp. $\cupdot$) increases S_i (resp. $\neg S_i$) by p_i
- p_i: unary or binary logic program predicate (input predicate)
- $Q(\mathbf{t})$ is a DL-query:
  - $C(t)$, $\neg C(t)$, $\mathbf{t} = t$, where C is an ontology concept
  - $R(t_1, t_2)$, $\neg R(t_1, t_2)$, $\mathbf{t} = t_1, t_2$, where R is an ontology role

DL-programs: semantics

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

$$\mathcal{O} = \left\{ \begin{array}{ll} (1)\ hasChild^{-} \sqsubseteq hasParent & (3)\ Male(pat) \\ (2)\ Female \sqsubseteq \neg Male & (4)\ hasChild(pat, john) \end{array} \right\}$$

$$\mathcal{P} = \left\{ \begin{array}{l} (5)\ boy(john); \\ (6)\ hasfather(john, pat) \leftarrow \underbrace{\mathrm{DL}[;\ hasParent](john, pat)}_{d_1}, \\ \qquad\qquad\qquad\qquad \underbrace{\mathrm{DL}[Male \uplus boy;\ Male](pat)}_{d_2} \end{array} \right\}$$

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

DL-programs: semantics

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

$$\mathcal{O} = \left\{ \begin{array}{ll} (1)\ hasChild^- \sqsubseteq hasParent & (3)\ Male(pat) \\ (2)\ Female \sqsubseteq \neg Male & (4)\ hasChild(pat, john) \end{array} \right\}$$

$$\mathcal{P} = \left\{ \begin{array}{l} (5)\ boy(john); \\ (6)\ hasfather(john, pat) \leftarrow \underbrace{\mathrm{DL}[; hasParent](john, pat)}_{d_1}, \\ \qquad \underbrace{\mathrm{DL}[Male \uplus boy; Male](pat)}_{d_2} \end{array} \right\}$$

- Interpretation: $I = \{boy(john), hasfather(john, pat)\}$
- Satisfaction relation: $I \models^{\mathcal{O}} boy(john)$ as $boy(john) \in I$
  $I \models^{\mathcal{O}} d_1$ as $\mathcal{O} \models hasParent(john, pat)$
  $I \models^{\mathcal{O}} d_2$ as $\mathcal{O} \cup Male(john) \models Male(pat)$
- 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

$\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$

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

$$\mathcal{P} = \left\{ \begin{array}{l} (7)\ \mathit{ischildof}(\mathit{john}, \mathit{alex});\quad (8)\ \mathit{boy}(\mathit{john}); \\ (9)\ \mathit{hasfather}(\mathit{john}, \mathit{pat}) \leftarrow \mathrm{DL}[\mathit{Male} \uplus \mathit{boy}; \mathit{Male}](\mathit{pat}), \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathrm{DL}[; \mathit{hasParent}](\mathit{john}, \mathit{pat}); \\ (10)\ \bot \leftarrow \mathit{not}\ \mathrm{DL}[; \mathit{Adopted}](\mathit{john}), \\ \qquad\qquad \mathit{hasfather}(\mathit{john}, \mathit{pat}), \mathit{ischildof}(\mathit{john}, \mathit{alex}), \\ \qquad\qquad \mathit{not}\ \mathrm{DL}[\mathit{Child} \uplus \mathit{boy}; \neg \mathit{Male}](\mathit{alex}) \end{array} \right\}$$

Example: Inconsistent DL-program

$\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$

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

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

Example: Inconsistent DL-program

$\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$

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

$$\mathcal{P} = \left\{ \begin{array}{l} (7)\ \mathit{ischildof}(\mathit{john}, \mathit{alex});\quad (8)\ \mathit{boy}(\mathit{john}); \\ (9)\ \mathit{hasfather}(\mathit{john}, \mathit{pat}) \leftarrow \mathrm{DL}[\mathit{Male} \uplus \mathit{boy}; \mathit{Male}](\mathit{pat}), \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathrm{DL}[; \mathit{hasParent}](\mathit{john}, \mathit{pat}); \\ (10)\ \bot \leftarrow \mathit{not}\ \mathrm{DL}[; \mathit{Adopted}](\mathit{john}), \\ \qquad\qquad \mathit{hasfather}(\mathit{john}, \mathit{pat}), \mathit{ischildof}(\mathit{john}, \mathit{alex}), \\ \qquad\qquad \mathit{not}\ \mathrm{DL}[\mathit{Child} \uplus \mathit{boy}; \neg \mathit{Male}](\mathit{alex}) \end{array} \right\}$$

Example: Inconsistent DL-program

$\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$ is inconsistent!

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

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

No answer sets

Related Work

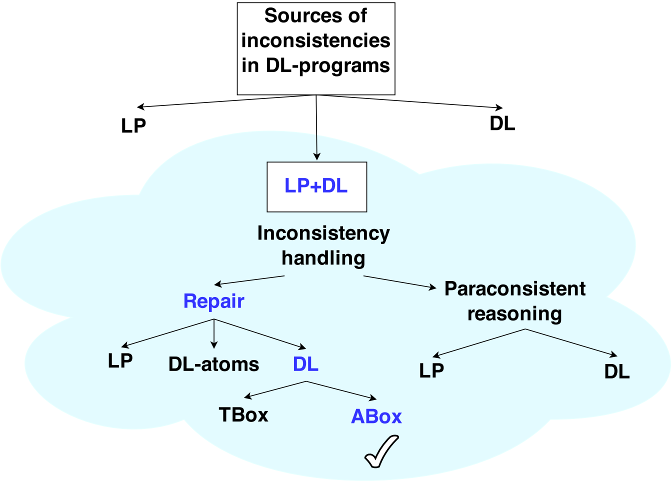
- Repairing ontologies
  - consistent query answering over *DL-Lite* ontologies based on repair technique [Bienvenu *et al.*, 2014], [Lembo *et al.*, 2010]
  - QA over $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ ontologies that miss expected tuples (abductive explanations corresponding to repairs) [Calvanese *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]

Research Goal

Our goal: develop techniques for handling inconsistencies in DL-programs
Our approach: repair ontology ABox to regain consistency



Research Questions

On the theoretical level:

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

On the practical level:

- ? Practical algorithms and optimizations?
- ? 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 $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ 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 $\Pi = \langle \mathcal{O}, P \rangle$ be a DL-program, where $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$

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

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

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

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

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

$RAS_x(\Pi)$ is the set of all repair AS of Π.

$rep_x^I(\Pi)$ is the set of all $\mathcal{A}'$ under which I is a repair answer set of Π.

Example: repair

$\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$ is inconsistent!

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

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

No answer sets

Example: repair

$\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$ is consistent!

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

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

$\mathcal{A}' = \{\mathit{Female}(\mathit{pat}), \mathit{Male}(\mathit{john}), \mathit{hasParent}(\mathit{john}, \mathit{pat})\}$ is a repair

$I' = \{\mathit{ischildof}(\mathit{john}, \mathit{alex}), \mathit{boy}(\mathit{john})\}$ is a repair answer set

$\mathcal{A}' \in \mathit{rep}^{I'}_{\mathit{flp}}(\Pi)$, $I' \in \mathit{RAS}_{\mathit{flp}}(\Pi)$

Example: repair

$\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$ is consistent!

$$\mathcal{O} = \left\{ \begin{array}{ll} (1)\ \mathit{Child} \sqsubseteq \exists \mathit{hasParent} & (4)\ \mathit{Male}(\mathit{pat}) \\ (2)\ \mathit{Adopted} \sqsubseteq \mathit{Child} & (5)\ \mathit{Male}(\mathit{john}) \\ (3)\ \mathit{Female} \sqsubseteq \neg \mathit{Male} & \end{array} \right\}$$

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

$\mathcal{A}'' = \{\mathit{Male}(\mathit{pat}), \mathit{Male}(\mathit{john})\}$ is a repair
$I' = \{\mathit{ischildof}(\mathit{john}, \mathit{alex}), \mathit{boy}(\mathit{john})\}$ is a repair answer set
$\mathcal{A}'' \in \mathit{rep}^{I'}_{\mathit{flp}}(\Pi)$, $I' \in \mathit{RAS}_{\mathit{flp}}(\Pi)$

Complexity of Repair Answer Sets

INSTANCE: A ground DL-program $\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$.
QUESTION: Does there exist a repair answer set for Π 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 $\Pi = \langle \mathcal{O}, \mathcal{P} \rangle$, where $\mathcal{O}$ is in DL-Lite$_{\mathcal{A}}$ or $\mathcal{EL}$.

Π	$RAS_{flp}(\Pi) \neq \emptyset$	$RAS_{weak}(\Pi) \neq \emptyset$
normal	Σ_2^P-complete	NP-complete
disjunctive	Σ_2^P-complete	Σ_2^P-complete

DL-program Evaluation

Algorithm *AnsSet*: Compute $AS_x(\Pi)$

Input: A DL-program Π, $x \in \{weak, flp\}$
Output: $AS_x(\Pi)$
for $\hat{I} \in AS(\hat{\Pi})$ **do**
  if $CMP(\hat{I}, \Pi) \wedge xFND(\hat{I}, \Pi)$ **then**
    output $\hat{I}|_{\Pi}$
  end
end

- $\hat{\Pi}$ is Π with all DL-atoms a substituted by ordinary atoms e_a plus additional guess rules $e_a \vee ne_a$ 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 Π

DL-program Evaluation

Algorithm $AnsSet$: Compute $AS_x(\Pi)$

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

Output: $AS_x(\Pi)$

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

(2a,b) **if** $CMP(\hat{I}, \Pi) \wedge xFND(\hat{I}, \Pi)$ **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

- $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ 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.

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

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

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

- $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ is consistent;
- $\mathcal{O}' \cup U_{j_1}^1 \models Q_j^1$ holds for $1 \leq j_1 \leq m_1$;
- $\mathcal{O}' \cup U_{j_2}^2 \not\models Q_k^2$ 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

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

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

- $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ is consistent;
- $\mathcal{O}' \cup U_{j_1}^1 \models Q_j^1$ holds for $1 \leq j_1 \leq m_1$;
- $\mathcal{O}' \cup U_{j_2}^2 \not\models Q_k^2$ 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 $DL\text{-}Lite_{\mathcal{A}}$

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 δ^{+}: 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:
$S = \{\mathcal{A}' \mid |\mathcal{A}'^{+} \backslash \mathcal{A}| \leq k$ or $|\mathcal{A}'^{-} \backslash \mathcal{A}| \leq k\}$, for some k

Tractable Cases of ORP for *DL-Lite*$_{\mathcal{A}}$

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 δ^{+}: 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:
$S = \{\mathcal{A}' \mid |\mathcal{A}'^{+} \backslash \mathcal{A}| \leq k \text{ or } |\mathcal{A}'^{-} \backslash \mathcal{A}| \leq k\}$, for some k

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

A selection $\sigma : 2^{\mathcal{AB}} x \mathcal{AB} \rightarrow 2^{\mathcal{AB}}$ is independent if $\sigma(S, \mathcal{A}) = \sigma(S', \mathcal{A}) \cup \sigma(S \backslash S', \mathcal{A})$, whenever $S' \subseteq S$.

Example

C1-C4 are independent, but $\subseteq$-minimal repairs are not.

Naive Repair Algorithm

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

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

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

```
for A' ∈ ORP(Î, Π, σ) do
    if xFND(Î, ⟨T, A', P⟩) then
        output A'
    end
end
```

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

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

Naive Repair Algorithm

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

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

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

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

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

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

RepAns and *RepAnsSet* are sound and complete for independent σ.

Ground Support Sets

For optimization purposes we introduce support sets:
Support set for $d = \mathrm{DL}[\lambda; Q](\mathbf{t})$ is a minimal set S, s.t. $S \cup \mathcal{T} \models Q(\mathbf{t})$

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

When is d true under interpretation I?

- $\mathit{Male}(\mathit{pat}) \in \mathcal{A}$
- $\mathit{boy}(\mathit{pat}) \in I$
- $\mathit{boy}(\mathit{alex}) \in I; \mathit{Female}(\mathit{alex}) \in \mathcal{A}$

Ground Support Sets

For optimization purposes we introduce support sets:
Support set for $d = \text{DL}[\lambda; Q](\mathbf{t})$ is a minimal set S, s.t. $S \cup \mathcal{T} \models Q(\mathbf{t})$

$$d = \text{DL}[\overbrace{Male \uplus boy}^{\lambda}; Male](pat);\ \mathcal{T}_d = \{Female \sqsubseteq \neg Male; Male_{boy} \sqsubseteq Male\}$$

When is d true under interpretation I?

- $Male(pat) \in \mathcal{A}$
- $Male_{boy}(pat) \in \mathcal{A}_d$, s.t. $boy(pat) \in I$
- $Male_{boy}(alex) \in \mathcal{A}_d$, s.t. $boy(alex) \in I$; $Female(alex) \in \mathcal{A}$

where $\mathcal{A}_d = \{P_p(\mathbf{t}) \mid P \uplus p \in \lambda\} \cup \{\neg P_p(\mathbf{t}) \mid P \cup\!\!\!\!- \, p \in \lambda\}$

Ground Support Sets ($DL\text{-}Lite_{\mathcal{A}}$)

Definition

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

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

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

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

$d = \mathrm{DL}[Male \uplus boy; Male](pat)$; $\mathcal{T}_d = \{Female \sqsubseteq \neg Male; Male_{boy} \sqsubseteq Male\}$

When is d true under interpretation I?

- $S_1 = \{Male(pat)\}$, 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 (*DL-Lite*$_{\mathcal{A}}$)

Definition

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

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

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

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

$I \models^{\mathcal{O}} d$ iff there exists $S \in Supp_{\mathcal{O}}(d)$, which is coherent with I.

Nonground Support Sets ($DL\text{-}Lite_{\mathcal{A}}$)

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

Nonground support sets:

- $S_1 = \{Male(X)\}$
- $S_2 = \{Male_{boy}(X)\}$
- $S_3 = \{Male_{boy}(Y); Female(Y)\}$

Nonground Support Sets ($DL\text{-}Lite_{\mathcal{A}}$)

Definition

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

Nonground support sets are compact representations of ground ones.

Nonground Support Sets (*DL-Lite*$_{\mathcal{A}}$)

Definition

$S = \{P(\mathbf{Y}), P'(\mathbf{Y}')\}$ ($S = \{P(\mathbf{Y})\}$) is a *DL-Lite*$_{\mathcal{A}}$ nonground support set for a DL-atom $d(\mathbf{X})$ w.r.t. $\mathcal{T}$ if for every $\theta : V \rightarrow \mathcal{C}$ it holds that $S\theta$ is a support set for $d(\mathbf{X}\theta)$ w.r.t. $\mathcal{O}_{\mathcal{C}} = \langle \mathcal{T}, \mathcal{A}_{\mathcal{C}} \rangle$, where $\mathcal{A}_{\mathcal{C}}$ is a set of all possible assertions over $\mathcal{C}$.

Nonground support sets are compact representations of ground ones.

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

Complete support families allow to avoid access to $\mathcal{O}$ during DL-atom evaluation.

Nonground Support Set Computation ($DL\text{-}Lite_{\mathcal{A}}$)

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

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

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

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

Nonground Support Set Computation ($DL\text{-}Lite_{\mathcal{A}}$)

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

- Construct $\mathcal{T}_d$ by compiling info about input predicates of d into $\mathcal{T}$:
  $\mathcal{T}_d = \mathcal{T} \cup \{Male_{boy} \sqsubseteq Male\}$
- Compute classification $Cl(\mathcal{T}_d)$ (e.g. using ASP techniques):
  $cl(\mathcal{T}_d) = \mathcal{T}_d \cup \{Male \sqsubseteq \neg Female;\ Male_{boy} \sqsubseteq \neg Female\} \cup \{P \sqsubseteq P \mid P \in \mathbf{P}\}$
- Extract support sets from $Cl(\mathcal{T}_d)$:
  - $S_1 = \{Male(X)\}$
  - $S_2 = \{Male_{boy}(X)\}$
  - $S_3 = \{Male_{boy}(Y), \neg Male(Y)\}$
  - $S_4 = \{Male_{boy}(Y), Female(Y)\}$
  - ~~$S_5 = \{Male(Y), \neg Male(Y)\}$~~
  - ~~$S_6 = \{Male(Y), Female(Y)\}$~~

  } $\mathcal{O}$ is consistent!

Nonground Support Set Computation ($DL\text{-}Lite_{\mathcal{A}}$)

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

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

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

- Extract support sets from $Cl(\mathcal{T}_d)$:
  - $S_1 = \{Male(X)\}$
  - $S_2 = \{Male_{boy}(X)\}$
  - $S_3 = \{Male_{boy}(Y), \neg Male(Y)\}$
  - $S_4 = \{Male_{boy}(Y), Female(Y)\}$

  $\{S_1, S_2, S_3, S_4\}$ is complete!

Optimized Deletion-RAS Computation ($DL\text{-}Lite_{\mathcal{A}}$)

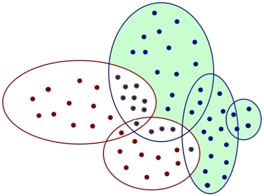
✓ Compute complete support families **S** for all DL-atoms of Π

- 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: $e_a \vee ne_a$
- 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 $\mathcal{A}$: $S^{\hat{I}}_{gr} \leftarrow Gr(\mathbf{S}, \hat{I}, \mathcal{A})$

✓ Find $\mathcal{A}'$, such that

- ✓ For all $a \in D_p$: there is $S \in S^{\hat{I}}_{gr}(a)$, s.t. $S \cap \mathcal{A}' \neq \emptyset$ or $S \subseteq \mathcal{A}_a$
- ✓ For all $a' \in D_n$: for all $S \in S^{\hat{I}}_{gr}(a')$: $S \cap \mathcal{A}' = \emptyset$ and $S \not\subseteq \mathcal{A}_{a'}$
- ✓ Minimality check of $\hat{I}|_{\Pi}$ wrt. $\Pi' = \langle \mathcal{O}', \mathcal{P} \rangle$, $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$



Optimized Deletion-RAS Computation ($DL\text{-}Lite_{\mathcal{A}}$)

✓ Compute complete support families **S** for all DL-atoms of Π

- 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: $e_a \vee ne_a$
- Fo

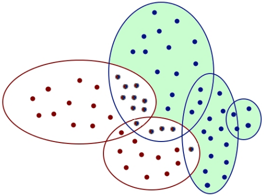
Sound and complete

wrt. deletion repair answer sets!

✓ Ground support sets in **S** wrt. ...

✓ Find $\mathcal{A}'$, such that

- ✓ For all $a \in D_p$: there is $S \in S^{\hat{I}}_{gr}(a)$, s.t. $S \cap \mathcal{A}' \neq \emptyset$ or $S \subseteq \mathcal{A}_a$
- ✓ For all $a' \in D_n$: for all $S \in S^{\hat{I}}_{gr}(a')$: $S \cap \mathcal{A}' = \emptyset$ and $S \not\subseteq \mathcal{A}_{a'}$
- ✓ Minimality check of $\hat{I}|_{\Pi}$ wrt. $\Pi' = \langle \mathcal{O}', \mathcal{P} \rangle$, $\mathcal{O}' = \langle \mathcal{T}, \mathcal{A}' \rangle$



Extending Approach to $\mathcal{EL}$

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

- Construct $\mathcal{T}_d$ by compiling info about input predicates of d into $\mathcal{T}$:
  $\mathcal{T}_d = \mathcal{T} \cup \{Proj_{projfile} \sqsubseteq Proj\}$
- Rewrite DL-query over normalized $\mathcal{T}_d$ into a datalog program:

$$\mathcal{T}_{d_{norm}} = \left\{ \begin{array}{ll} (1)\ StaffRequest \sqsubseteq \exists hasSubj.Staff & (2)\ Proj_{projfile} \sqsubseteq Proj \\ (3)\ StaffRequest \sqsubseteq hasTarg.Proj & (4)\ \exists hasSubj.Staff \sqsubseteq C_1 \\ (5)\ \exists hasTarg.Proj \sqsubseteq C_2 & (6)\ C_1 \sqcap C_2 \sqsubseteq StaffRequest \end{array} \right\}$$

Extending Approach to $\mathcal{EL}$

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

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

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

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

Extending Approach to $\mathcal{EL}$

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

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

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

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

- Unfold the DL-query and extract support sets:

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

Extending Approach to $\mathcal{EL}$

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

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

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

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

- Unfold the DL-query and extract support sets:

  $S_1 = \{hasSubj(X,Y), Staff(X), hasTarg(X,Z), Proj(Z)\}$

  $S_2 = \{hasSubj(X,Y), Staff(X), hasTarg(X,Z), Proj_{projfile}(Z)\}$

Extending Approach to $\mathcal{EL}$

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

- Construct $\mathcal{T}_d$ by compiling info about input predicates of d into $\mathcal{T}$:
  $\mathcal{T}_d = \mathcal{T} \cup \{Proj_{projfile} \sqsubseteq Proj\}$
- Rewrite DL-query over normalized $\mathcal{T}_d$ into a datalog program:

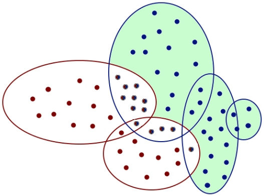
$$\mathcal{P}_{\mathcal{T}_{d_{norm}}} = \left\{ \begin{array}{l} (1^*)\ StaffRequest(X) \leftarrow C_1(X), C_2(X) \\ (2^*)\ C_1(X) \leftarrow hasSubj(X, Y), Staff(Y) \\ (3^*)\ C_2(X) \leftarrow hasTarg(X, Y), Proj(Y) \\ (4^*)\ Proj(X) \leftarrow Proj_{projfile}(X) \end{array} \right\}$$

- 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 ($\mathcal{EL}$)

- ✓ Compute partial support families **S** for all DL-atoms of Π
- 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: $e_a \vee ne_a$
- 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 $\mathcal{A}$: $S^{\hat{I}}_{gr} \leftarrow Gr(\mathbf{S}, \hat{I}, \mathcal{A})$
- ✓ 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 $S \in S^{\hat{I}}_{gr}$, s.t. $S \cap H = \emptyset$, then do eval. postcheck on D_n (evaluate atoms from D_n over I and $\mathcal{A} \backslash H$)
  - ✓ Else do eval. postcheck on D_n and D_p
- ✓ Check minimality of $\hat{I}|_\Pi$ wrt. $\Pi' = \langle \mathcal{T}, \mathcal{A} \backslash H, \mathcal{P} \rangle$



Optimized Deletion RAS Computation ($\mathcal{EL}$)

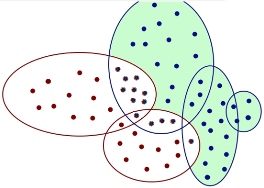
✓ Compute partial support families **S** for all DL-atoms of Π

- 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: $e_a \vee ne_a$
- 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!

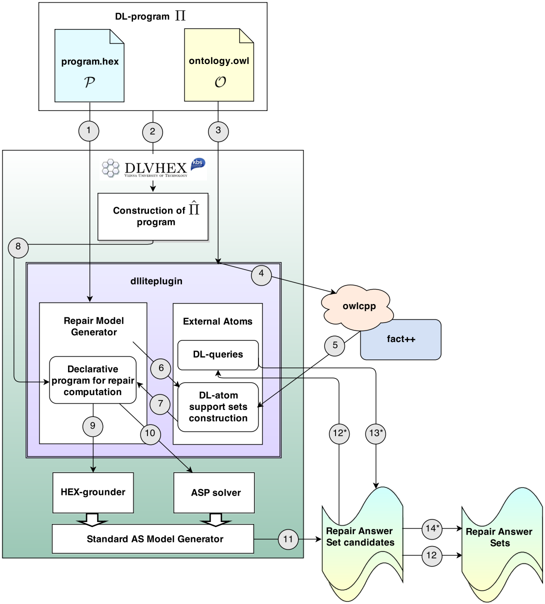
✓ If all $a \in D_p$ have at least one $S \in S^{I}_{gr}$, s.t. $S \cap H = \emptyset$, then do eval. postcheck on D_n (evaluate atoms from D_n over I and $\mathcal{A} \backslash H$)

✓ Else do eval. postcheck on D_n and D_p



✓ Check minimality of $\hat{I}|_{\Pi}$ wrt. $\Pi' = \langle \mathcal{T}, \mathcal{A} \backslash H, \mathcal{P} \rangle$

System Architecture



Experiments

Assessment of our algorithms concerns the following aspects:

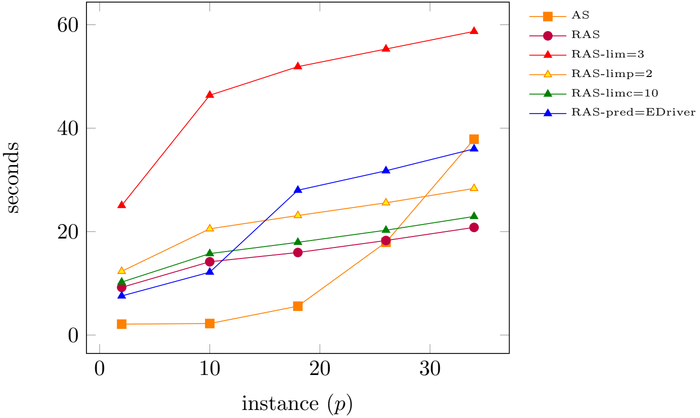
- *Scalability*
  - size of the DL-program data part
  - size of the ontology TBox
  - number of rules in the DL-program
- *Repair quality*
  - bounding number/type of assertions for deletion
- *Expressive features*
  - defaults
  - guesses
  - recursiveness
- *Real world data*
  - Taxi-driver assignment problem
  - Open Street Map
- *Effects of support family completeness*

Taxi-Driver Benchmark ($DL\text{-}Lite_{\mathcal{A}}$)

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

$$\mathcal{P} = \left\{ \begin{array}{l} (5)\ cust(X) \leftarrow isIn(X,Y), not\ \mathrm{DL}[;\neg Cust](X); \\ (6)\ driver(X) \leftarrow not\ cust(X), isIn(X,Y); \\ (7)\ drives(X,Y) \leftarrow driver(X), cust(Y), needsTo(Y,Z1), goTo(X,Z2), \\ \qquad\qquad \mathrm{DL}[;adjoint](Z1,Z2), not\ omit(X,Y); \\ (8)\ omit(X,Y) \leftarrow \mathrm{DL}[;EDriver](X), needsTo(Y,Z), \\ \qquad\qquad \mathrm{DL}[;notworksIn](X,Z); \\ (9)\ ok(Y) \leftarrow customer(Y), drives(X,Y); \\ (10)\ fail \leftarrow customer(Y), not\ ok(Y); \\ (11)\ \perp \leftarrow fail \end{array} \right\}$$

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 ($\mathcal{EL}$)

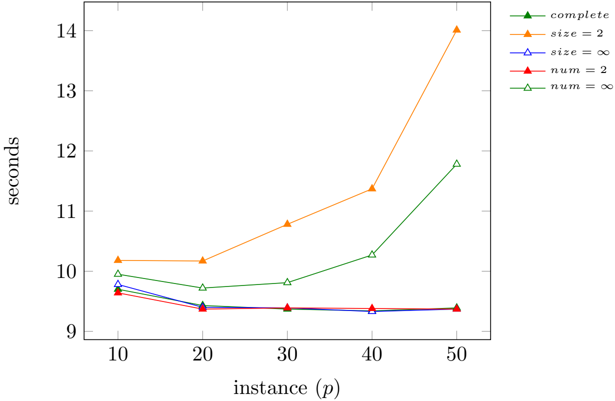
$$\mathcal{O} = \left\{ \begin{array}{l} (1)\ \mathit{BuildingFeature} \sqcap \exists \mathit{isLocatedInside}.\mathit{Private} \sqsubseteq \mathit{NoPublicAccess} \\ (2)\ \mathit{BusStop} \sqcap \mathit{Roofed} \sqsubseteq \mathit{CoveredBusStop} \end{array} \right\}$$

$$\mathcal{P} = \left\{ \begin{array}{l} (9)\ \mathit{publicstation}(X) \leftarrow \mathrm{DL}[\mathit{BusStop} \uplus \mathit{busstop}; \mathit{CoveredBusStop}](X); \\ \qquad\qquad\qquad\qquad \mathit{not}\ \mathrm{DL}[; \mathit{Private}](X); \\ (10)\ \bot \leftarrow \mathrm{DL}[\mathit{BuildingFeature} \uplus \mathit{publicstation}; \mathit{NoPublicAccess}](X), \\ \qquad\qquad \mathit{publicstation}(X). \end{array} \right\}$$

- Rules on top of the MyITS ontology:[4]
  - 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.

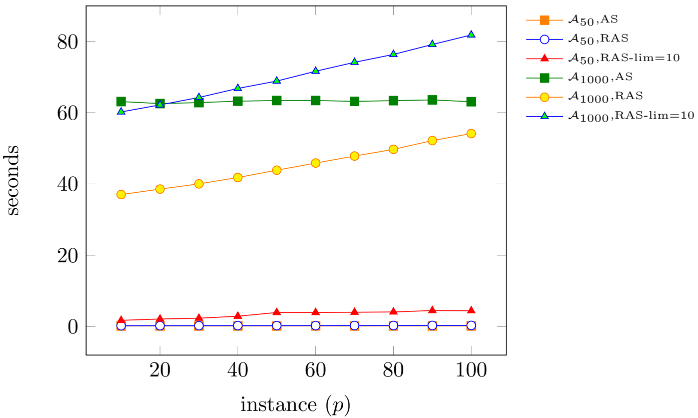
[4] http://www.kr.tuwien.ac.at/research/projects/myits/

Open Street Map Benchmark ($\mathcal{EL}$)



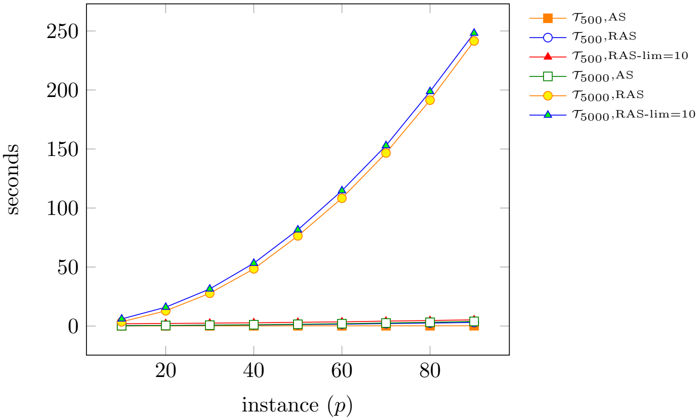
- $\mathcal{A}$: bus stops (285) and leisure areas (682) of Cork, plus role *isLocatedInside* on them (9)
- Randomly made 80% bus stops roofed, 60% leisure areas private
- For *isLocatedInside*(*bs*, *la*) make *bs* a bus stop with *p* chance (*x*-axis)
- DL-atoms have few support sets

Family Benchmark (*DL-Lite*$_\mathcal{A}$)



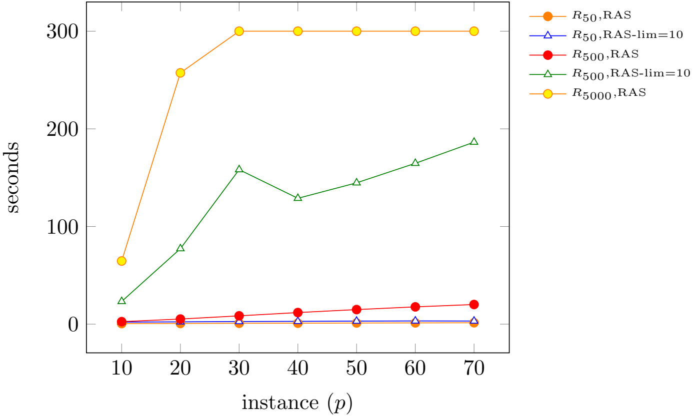
1. Data part variations:
   - $\mathcal{A}_{50}$ contains 50 children (7 adopted), 20 female, 32 male adults (20 times that many for $\mathcal{A}_{1000}$), $\mathcal{T}$ is fixed
   - Instance size p: facts $boy(c)$, $isChildOf(c, d)$ are in $\mathcal{P}$ with prob. $p/100$.

Family Benchmark (*DL-Lite*$_\mathcal{A}$)



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

Family Benchmark ($DL\text{-}Lite_{\mathcal{A}}$)



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

Benchmark		Ontology expressivity	TBox size	Concepts	Roles	ABox Size		Individuals
Family		DL-Lite$_{\mathcal{A}}$	3	5	1	$\mathcal{A}_{50}$	312	102
						$\mathcal{A}_{1000}$	6183	2021
Network		DL-Lite$_{\mathcal{A}}$	3	4	2	$\mathcal{A}_{67}$	204	67
			3	5	2	$\mathcal{A}_{161}$	672	161
Taxi	Basic	DL-Lite$_{\mathcal{A}}$	3	4	2	$\mathcal{A}_{50}$	259	75
						$\mathcal{A}_{500}$	4370	714
	Time		4	6	2		274	75
	Districts		389	339	41	$\mathcal{A}_{50}$	418	93
						$\mathcal{A}_{500}$	6744	723
LUBM	Basic	DL-Lite$_{\mathcal{A}}$	95	44	31		7293	1555
	Diamond							
	Extended		101	48	31		7412	1605
Policy		$\mathcal{EL}$	5	8	3	$\mathcal{A}_{40}$	199	64
						$\mathcal{A}_{100}$	475	148
						$\mathcal{A}_{1000}$	4615	1408
OSM		$\mathcal{EL}$	405	356	36		4195	1537
LUBM-basic		$\mathcal{EL}$	94	47	28		2285	832

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 $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ or $\mathcal{EL}$
- 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.
In *Proceedings of the 6th International Conference on Web Reasoning and Rule Systems (RR 2012), 58-74*, 2012.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Data repair of inconsistent DL-programs.
In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2013.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Inconsistency management for Description Logic Programs and beyond.
In *Proceedings of the 6th International Conference on Web Reasoning and Rule Systems (RR 2013), 1-3*, 2013.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Towards practical deletion repair of inconsistent DL-programs.
In *Proceedings of the 27th International Workshop on Description Logics (DL workshop 2014), 169-180*, 2014.

Thomas Eiter, Michael Fink, Christoph Redl, and Daria Stepanova.
Exploiting support sets for answer set programs with external computations.
In *Proceedings of the 28th Conference on Artificial Intelligence (AAAI 2014), 1041-1048*, 2014.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Computing repairs for inconsistent DL-programs over EL ontologies.
In *Proceedings of the 14th International Conference on Logics in Artificial Intelligence (JELIA 2014), 426-441*, 2014.

Thomas Eiter, Michael Fink, and Daria Stepanova.
Towards practical deletion repair of inconsistent DL-programs.
In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014), 285-290*, 2014.

Daria Stepanova.
Inconsistencies in hybrid knowledge bases.
In *Proceedings of 14th Doctoral Consortium on Knowledge Represenation (DC of KR 2014)*, 2014.

References I

Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué.
Querying inconsistent description logic knowledge bases under preferred repair semantics.
In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2014*, pages 996–1002, 2014.

Diego Calvanese, Magdalena Ortiz, Mantas Simkus, and Giorgio Stefanoni.
The complexity of explaining negative query answers in DL-Lite.
In *Proceedings of the 13th International Conference on the Principles of Knowledge Representation and Reasoning*, Rome, Italy, June 2012. American Association for Artificial Intelligence.

Jos de Bruijn, David Pearce, Axel Polleres, and Agustín Valverde.
Quantified equilibrium logic and hybrid rules.
In *Web Reasoning and Rule Systems, First International Conference, RR 2007, Innsbruck , Austria, June 7-8, 2007, Proceedings*, pages 58–72, 2007.

Jos de Bruijn, Thomas Eiter, Axel Polleres, and Hans Tompits.
Embedding nonground logic programs into autoepistemic logic for knowledge-base combination.
ACM Trans. Comput. Log., 12(3):20, 2011.

Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits.
Combining answer set programming with description logics for the semantic web.
Artificial Intelligence, 172(12-13):1495–1539, August 2008.

Michael Fink.
Paraconsistent hybrid theories.
In *Principles of Knowledge Representation and Reasoning: Proceedings of the 13th International Conference, KR*, pages 141–151, Rome, Italy, June 2012. American Association for Artificial Intelligence.

Michael Gelfond and Vladimir Lifschitz.
The stable model semantics for logic programming.
In *Proceedings of the 5th Intnl Conference and Symposium (ICLP'88)*, pages 1070–1080. The MIT Press, 1988.

P. J. Hayes.
The logic of frames.
In *Frame Conceptions and Text Understanding*, pages 46–61. 1979.

References II

Shasha Huang, Qingguo Li, and Pascal Hitzler.
Reasoning with inconsistencies in hybrid MKNF knowledge bases.
Logic Journal of the IGPL, 21(2):263–290, 2013.

Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo.
Inconsistency-tolerant semantics for description logic ontologies.
In *Proceedings of the 19th Italian Symposium on Advanced Database Systems*, pages 103–117, Bressanone/Brixen, Italy, September 2010. Springer.

Alon Y. Levy and Marie-Christine Rousset.
Combining horn rules and description logics in CARIN.
Artif. Intell., 104(1-2):165–209, 1998.

John McCarthy.
Programs with common sense.
In *TeddingtonConference*, pages 75–91, 1959.

Marvin Minsky.
A framework for representing knowledge.
In *Readings in Knowledge Representation*, pages 245–262. Kaufmann, 1985.

Boris Motik and Riccardo Rosati.
Reconciling Description Logics and Rules.
Journal of the ACM, 57(5):1–62, June 2010.

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

Johannes Oetsch, Jörg Pührer, and Hans Tompits.
Stepwise debugging of description-logic programs.
In *Correct Reasoning*, pages 492–508, 2012.

References III

Jörg Pührer, Stijn Heymans, and Thomas Eiter.
Dealing with inconsistency when combining ontologies and rules using DL-programs.
In *Proceedings of 7th Extended Semantic Web Conference, part I*, pages 183–197, Heraklion, Crete, Greece, May-June 2010. Springer.

Jörg Pührer.
Stepwise Debugging in Answer-Set Programming: Theoretical Foundations and Practical Realisation. Dissertation.
Phd thesis, Vienna University of Technology, Vienna, Austria, 2014.

M. Ross Quillan.
Word concepts: A theory and simulation of some basic capabilities.
Behavioral Science, pages 410–430, 1967.

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

Riccardo Rosati.
Dl+log: Tight integration of description logics and disjunctive datalog.
In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 68–78, 2006.

Chiaki Sakama and Katsumi Inoue.
An abductive framework for computing knowledge base updates.
Theory and Practice of Logic Programming, 3(6):671–713, May 2003.

Tommi Syrjänen.
Debugging Inconsistent Answer-Set Programs.
In *Proc. of the 11th Int. Workshop on Nonmonotonic Reasoning, (NMR'06), Lake District, England, UK*, pages 77–83. University of Clausthal, Department of Informatics, Technical Report, IfI-06-04, 2006.

References IV

Kewen Wang, David Billington, Jeff Blee, and Grigoris Antoniou.
Combining description logic and defeasible logic for the semantic web.
In *Rules and Rule Markup Languages for the Semantic Web: Third International Workshop, RuleML 2004, Hiroshima, Japan, November 8, 2004. Proceedings*, pages 170–181, 2004.

DL-program

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

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

- for ground literal ℓ: $I \models^{\mathcal{O}} \ell$ 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 $\mathcal{T} \cup \mathcal{A} \cup \lambda^I(a) \models Q(\mathbf{c})$, 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 $op_i = \uplus$;
  - $A_i(I) = \{\neg S_i(t) \mid p_i(t) \in I\}$, for $op_i = \cup\!\!\!-$;
  - $A_i(I) = \{\neg S_i(t) \mid p_i(t) \notin I\}$, for $\cap\!\!\!-$.

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

Weak-reduct $\mathcal{P}_{weak}^{I,\mathcal{O}}$ 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 $\mathcal{P}_{flp}^{I,\mathcal{O}}$.

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

Family: data

p	$\mathcal{A}_{50}$			$\mathcal{A}_{1000}$		
	AS	*RAS*		*AS*	*RAS*	
		no_restr	$lim = 10$		*no_restr*	$lim = 10$
10 (20)	0.14 (0)[0]	0.22 (0)[20]	1.73 (0)[20]	63.12 (0)[0]	37.03 (0)[20]	60.21 (0)[20]
20 (20)	0.14 (0)[0]	0.23 (0)[20]	2.10 (0)[19]	62.56 (0)[0]	38.56 (0)[20]	62.19 (0)[20]
30 (20)	0.14 (0)[0]	0.24 (0)[20]	2.33 (0)[18]	62.83 (0)[0]	40.03 (0)[20]	64.27 (0)[20]
40 (20)	0.14 (0)[0]	0.25 (0)[20]	2.88 (0)[11]	63.23 (0)[0]	41.81 (0)[20]	66.81 (0)[20]
50 (20)	0.14 (0)[0]	0.25 (0)[20]	3.93 (0) [1]	63.42 (0)[0]	43.86 (0)[20]	68.87 (0)[20]
60 (20)	0.15 (0)[0]	0.26 (0)[20]	3.93 (0) [2]	63.42 (0)[0]	45.87 (0)[20]	71.63 (0)[20]
70 (20)	0.14 (0)[0]	0.27 (0)[20]	4.00 (0) [0]	63.18 (0)[0]	47.83 (0)[20]	74.14 (0)[20]
80 (20)	0.15 (0)[0]	0.28 (0)[20]	4.08 (0) [0]	63.38 (0)[0]	49.71 (0)[20]	76.35 (0)[20]
90 (20)	0.15 (0)[0]	0.29 (0)[20]	4.48 (0) [0]	63.59 (0)[0]	52.18 (0)[20]	79.14 (0)[20]
100 (20)	0.14 (0)[0]	0.30 (0)[20]	4.42 (0) [0]	63.08 (0)[0]	54.14 (0)[20]	81.81 (0)[20]

Table : Family benchmark: data size variations, fixed $\mathcal{P}$ and $\mathcal{T}$

Family: TBox (*DL-Lite*$_{\mathcal{A}}$)

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

Table : Family benchmark: TBox size variations, fixed $\mathcal{P}$ and $\mathcal{A}_{50}$

Family: Rules ($DL\text{-}Lite_{\mathcal{A}}$)

p	$Rules_{max} = 50$		$Rules_{max} = 500$		$Rules_{max} = 5000$	
	RAS	$RAS_{lim=10}$	RAS	$RAS_{lim=10}$	RAS	$RAS_{lim=20}$
10 (20)	0.55 (0)[20]	2.09 (0)[20]	2.56 (0)[20]	23.23 (0)[0]	64.65 (0)[20]	110.92 (0)[20]
20 (20)	0.69 (0)[20]	2.35 (0)[20]	5.22 (0)[20]	77.30 (0)[0]	257.35 (11)[9]	300.00 (20)[0]
30 (20)	0.90 (0)[20]	2.67 (0)[20]	8.50 (0)[20]	158.23 (0)[0]	300.00 (20)[0]	300.00 (20)[0]
40 (20)	0.97 (0)[20]	2.86 (0)[20]	11.86 (0)[20]	128.87 (1)[0]	300.00 (20)[0]	300.00 (20)[0]
50 (20)	1.18 (0)[20]	3.11 (0)[20]	14.91 (0)[20]	144.71 (0)[0]	300.00 (20)[0]	300.00 (20)[0]
60 (20)	1.29 (0)[20]	3.28 (0)[20]	17.68 (0)[20]	164.70 (0)[0]	300.00 (20)[0]	300.00 (20)[0]
70 (20)	1.42 (0)[20]	3.19 (0)[20]	20.11 (0)[20]	186.38 (3)[0]	300.00 (20)[0]	300.00 (20)[0]

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

Taxi-Driver

p	*AS*	*RAS*					
		no_restr	*lim* = 3	*lim* = 10	*limp* = 2	*limc* = 10	*EDriver*
2 (20)	2.11 (0) [0]	9.22 (0) [7]	25.05 (0) [6]	24.91 (0) [7]	12.32 (0) [7]	10.24 (0) [6]	7.56 (0) [0]
10 (20)	2.23 (0) [0]	14.17 (0)[20]	46.37 (0)[20]	46.52 (0)[20]	20.54 (0)[20]	15.75 (0)[15]	12.16 (0) [4]
18 (20)	5.58 (0) [5]	15.96 (0)[20]	51.89 (0)[20]	52.44 (0)[20]	23.11 (0)[20]	17.93 (0)[20]	28.00 (0)[20]
26 (20)	17.95 (0)[12]	18.28 (0)[20]	55.30 (0)[20]	55.84 (0)[20]	25.57 (0)[20]	20.27 (0)[20]	31.76 (0)[20]
34 (20)	37.87 (0)[17]	20.81 (0)[20]	58.71 (0)[20]	58.51 (0)[20]	28.35 (0)[20]	22.93 (0)[20]	36.00 (0)[20]

Table : Taxi-driver benchmark results: $\mathcal{A}_{500}$

LUBM

p	AS	RAS				
		RAS	$lim = 20$	$limp = 2$	$limc = 20$	IS
2 (20)	3.97 (0)[0]	13.98 (0)[20]	38.90 (0)[20]	16.01 (0)[20]	15.24 (0)[20]	15.20 (0)[6]
6 (20)	4.25 (0)[0]	16.16 (0)[20]	115.62 (0)[19]	18.08 (0)[20]	18.63 (0)[19]	11.16 (0)[2]
10 (20)	4.64 (0)[0]	18.95 (0)[20]	245.40 (0)[7]	20.85 (0)[20]	20.79 (0)[4]	9.12 (0)[0]
14 (20)	4.86 (0)[0]	21.50 (0)[20]	236.40 (1)[3]	23.73 (0)[20]	23.50 (0)[1]	9.53 (0)[0]
18 (20)	5.33 (0)[0]	24.86 (0)[20]	230.21 (0)[1]	27.11 (0)[20]	26.86 (0)[0]	10.15 (0)[0]
22 (20)	5.54 (0)[0]	28.21 (0)[20]	228.12 (0)[0]	30.19 (0)[20]	29.93 (0)[0]	10.36 (0)[0]
26 (20)	5.71 (0)[0]	31.50 (0)[20]	222.78 (0)[0]	33.84 (0)[20]	33.26 (0)[0]	10.75 (0)[0]
30 (20)	6.07 (0)[0]	36.88 (0)[20]	225.18 (0)[0]	38.82 (0)[20]	38.47 (0)[0]	11.45 (0)[0]
34 (20)	6.36 (0)[0]	42.18 (0)[20]	241.30 (0)[0]	44.29 (0)[20]	44.01 (0)[0]	12.22 (0)[0]
38 (20)	6.55 (0)[0]	46.07 (0)[20]	245.77 (0)[0]	47.87 (0)[20]	47.64 (0)[0]	12.41 (0)[0]
42 (20)	6.93 (0)[0]	52.50 (0)[20]	255.74 (0)[0]	54.17 (0)[20]	56.91 (0)[0]	12.94 (0)[0]
46 (20)	7.15 (0)[0]	56.98 (0)[20]	276.52 (5)[0]	58.96 (0)[20]	58.47 (0)[0]	13.35 (0)[0]
50 (20)	7.53 (0)[0]	63.96 (0)[20]	276.07 (5)[0]	65.79 (0)[20]	65.50 (0)[0]	14.18 (0)[0]

Table : LUBM benchmark results

Network Guessing

p	*RAS*			
	no_restr	*lim* = 10	*limc* = 100	*Broken*
2 (20)	178.52 (3)[15]	187.65 (2)[16]	175.64 (2)[16]	179.57 (3)[15]
4 (20)	201.89 (6)[10]	211.10 (7) [9]	213.66 (9) [7]	178.55 (3)[13]
8 (20)	212.18 (10) [2]	215.44 (10) [2]	205.77 (9) [3]	191.97 (7) [5]
10 (20)	190.58 (9) [0]	184.80 (8) [1]	191.54 (9) [0]	191.06 (9) [0]

Table : Network-guessing benchmark results: $\mathcal{A}_{161}$

Network Connectivity

p	*RAS*				
	no_restr	$lim = 3$	$lim = 20$	$lim = 100$	*Broken, forbid*
2 (20)	179.49 (1)[19]	280.73 (16)[0]	288.64 (17)[3]	176.06 (1)[19]	125.47 (0)[0]
4 (20)	218.80 (8)[12]	291.80 (18)[0]	295.48 (19)[1]	226.25 (8)[12]	127.68 (0)[0]
8 (20)	230.79 (9)[11]	298.39 (19)[0]	300.00 (20)[0]	232.65 (9)[11]	126.97 (0)[0]
10 (20)	258.08 (14)[5]	300.00 (20)[0]	300.00 (17)[0]	259.69 (14)[6]	125.63 (0)[0]

Table : Network-connectivity benchmark results: $\mathcal{A}_{161}$

Optimizations: Independent DL-atoms

In our repair approach 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:

$DL[; C \not\sqsubseteq C]()$;
. . . **?**

Tautology:

$DL[; C \sqsubseteq C]()$;
. . . **?**

Contradictions

When is a DL-atom contradictory in general?

Proposition

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

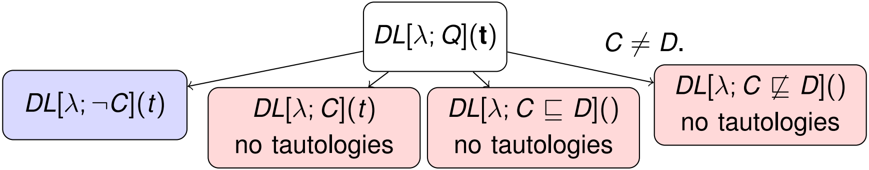
- $C \not\sqsubseteq C$;
- $C \not\sqsubseteq \top$;
- $\bot \not\sqsubseteq C$;
- $\bot \not\sqsubseteq \top$;
- $\top \sqsubseteq \bot$.

Tautologies

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

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

Concept query case distinction:

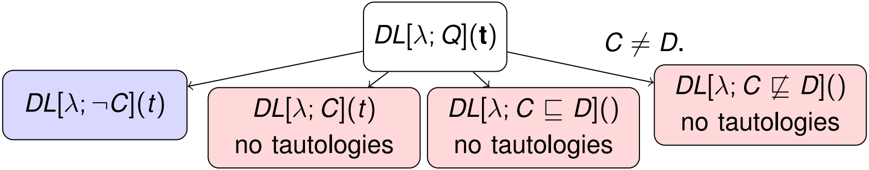


Tautologies

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

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

Concept query case distinction:



Example

$a = DL[\, C \mathbin{\cap\!\!\!\!\!-} p,\, C' \uplus p,\, C' \mathbin{\cap\!\!\!\!\!-} q,\, C \mathbin{\cup\!\!\!\!\!-} q; \neg C](c)$

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. $p(c) \notin I,\ q(c) \in I$

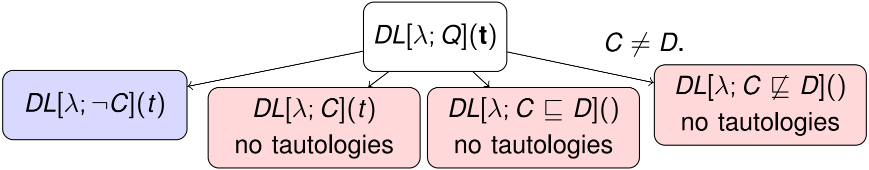
I is s.t. $p(c) \in I,\ q(c) \in I$

Tautologies

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

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

Concept query case distinction:



Example

$a = DL[\, C \mathbin{\cap\!\!\!\!\!-}\, p,\ C' \uplus p,\ C' \mathbin{\cap\!\!\!\!\!-}\, q,\ C \mathbin{\cup\!\!\!\!\!-}\, q; \neg C](c)$

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. $p(c) \notin I,\ q(c) \in I$

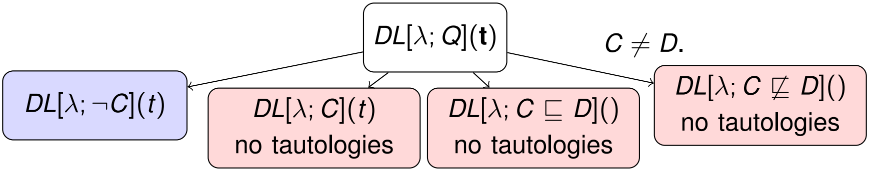
I is s.t. $p(c) \in I,\ q(c) \in I$

Tautologies

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

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

Concept query case distinction:



Example

$a = DL[\,C \mathbin{\cap\!\!\!\!-} p,\ C' \uplus p,\ C' \mathbin{\cap\!\!\!\!-} q,\ C \mathbin{\cup\!\!\!\!-} q; \neg C](c)$

I is s.t. $p(c) \notin I,\ q(c) \notin I$ $\qquad \tau^I(a) = \{\neg C(c)\}$

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

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

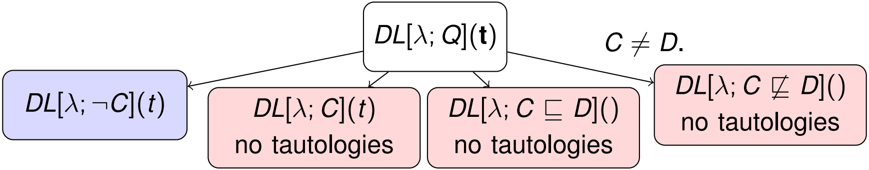
I is s.t. $p(c) \in I,\ q(c) \in I$

Tautologies

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

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

Concept query case distinction:



Example

$a = DL[\, C \mathbin{\cap\!\!\!\!-} p,\ C' \uplus p,\ C' \mathbin{\cap\!\!\!\!-} q,\ C \mathbin{\cup\!\!\!\!-} q; \neg C](c)$

I is s.t. $p(c) \notin I,\ q(c) \notin I$ $\qquad \tau^I(a) = \{\neg C(c)\}$

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

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

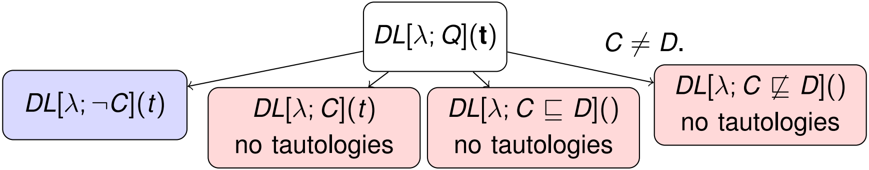
I is s.t. $p(c) \in I,\ q(c) \in I$

Tautologies

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

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

Concept query case distinction:



Example

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

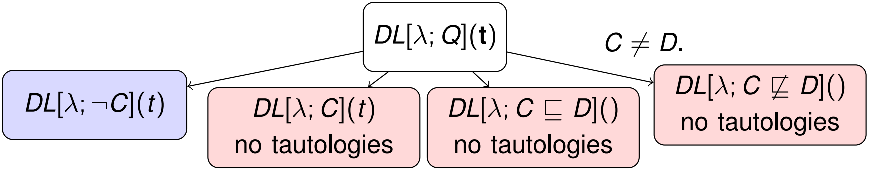
I is s.t. $p(c) \notin I,\ q(c) \notin I$	$\tau^I(a) = \{\neg C(c)\}$
I is s.t. $p(c) \in I,\ q(c) \notin I$	$\tau^I(a) = \{C'(c), \neg C'(c)\}$
I is s.t. $p(c) \notin I,\ q(c) \in I$	
I is s.t. $p(c) \in I,\ q(c) \in I$	

Tautologies

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

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

Concept query case distinction:



Example

$a = DL[\, C \mathbin{\cap\!\!\!\!-} p, C' \uplus p, C' \mathbin{\cap\!\!\!\!-} q, C \mathbin{\cup\!\!\!\!-} q; \neg C](c)$

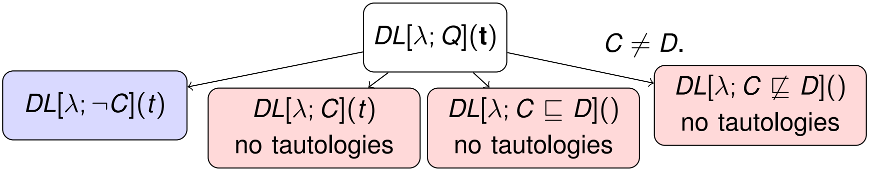
I is s.t. $p(c) \notin I,\ q(c) \notin I$	$\tau^I(a) = \{\neg C(c)\}$
I is s.t. $p(c) \in I,\ q(c) \notin I$	$\tau^I(a) = \{C'(c), \neg C'(c)\}$
I is s.t. $p(c) \notin I,\ q(c) \in I$	
I is s.t. $p(c) \in I,\ q(c) \in I$	

Tautologies

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

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

Concept query case distinction:



Example

$a = DL[C \cap\!\!\!\!- \, p, C' \uplus p, C' \cap\!\!\!\!- \, q, C \uminus q; \neg C](c)$

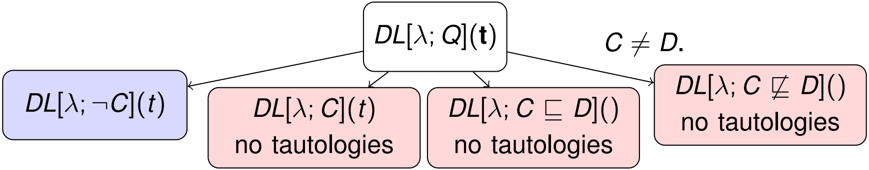
I is s.t. $p(c) \notin I$, $q(c) \notin I$	$\tau^I(a) = \{\neg C(c)\}$
I is s.t. $p(c) \in I$, $q(c) \notin I$	$\tau^I(a) = \{C'(c), \neg C'(c)\}$
I is s.t. $p(c) \notin I$, $q(c) \in I$	$\tau^I(a) = \{\neg C(c)\}$
I is s.t. $p(c) \in I$, $q(c) \in I$	

Tautologies

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

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

Concept query case distinction:



Example

$a = DL[\, C \mathbin{\cap\hspace{-0.6em}-}\, p,\ C' \uplus p,\ C' \mathbin{\cap\hspace{-0.6em}-}\, q,\ C \mathbin{\cup\hspace{-0.6em}-}\, q; \neg C](c)$

I is s.t. $p(c) \notin I,\ q(c) \notin I$ $\qquad \tau^I(a) = \{\neg C(c)\}$

I is s.t. $p(c) \in I,\ q(c) \notin I$ $\qquad \tau^I(a) = \{C'(c), \neg C'(c)\}$

I is s.t. $p(c) \notin I,\ q(c) \in I$ $\qquad \tau^I(a) = \{\neg C(c)\}$

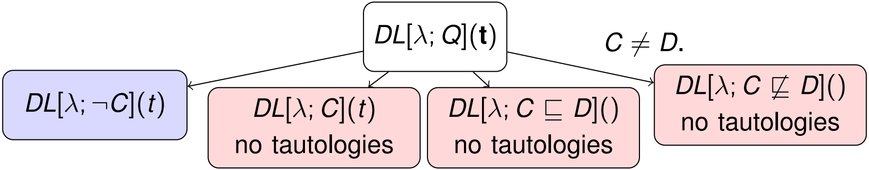
I is s.t. $p(c) \in I,\ q(c) \in I$

Tautologies

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

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

Concept query case distinction:



Example

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

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

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. $DL[\lambda, C \cap\!\!\!\!- \, p, C \uplus p; \neg C](t)$,

c2. $DL[\lambda, C \cap\!\!\!\!- \, p, D \uplus p, D \cup\!\!\!\!- \, p; \neg C](t)$,

Tautologies with Concept Query

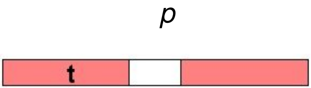
$$DL[\lambda; \neg C](t)$$

Proposition

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

c1. $DL[\lambda, C \cap\!\!\!\!\!- \; p, C \cup\!\!\!\!\!- \; p; \neg C](t)$,

c2. $DL[\lambda, C \cap\!\!\!\!\!- \; p, D \uplus p, D \cup\!\!\!\!\!- \; p; \neg C](t)$,



Tautologies with Concept Query

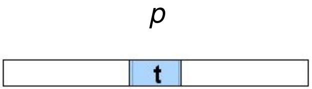
$DL[\lambda; \neg C](t)$

Proposition

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

c1. $DL[\lambda, C \cap\!\!\!\!- \, p, C \cup\!\!\!\!- \, p; \neg C](t)$,

c2. $DL[\lambda, C \cap\!\!\!\!- \, p, D \uplus p, D \cup\!\!\!\!- \, p; \neg C](t)$,



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

c1. $DL[\lambda, C \cap\!\!\!\!\cap\, p, C \cup\!\!\!\!\cup\, p; \neg C](t)$,

c2. $DL[\lambda, C \cap\!\!\!\!\cap\, p, D \uplus p, D \cup\!\!\!\!\cup\, p; \neg C](t)$,

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

c4. $DL[\lambda, C \cap\!\!\!\!\cap\, p_0, C^0 \uplus p_0, C^0 \cap\!\!\!\!\cap\, p'_0, C^1 \uplus p_1, C^1 \cap\!\!\!\!\cap\, p'_1, \dots,$
$C^n \uplus p_n, C^n \uplus p'_n, D \uplus p_{n+1}, D \cup\!\!\!\!\cup\, p'_{n+1}; \neg C](t)$,

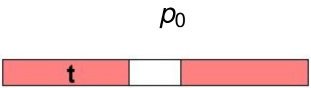
where for every $i = 0, \dots, n+1$, $p_i = p'_j$ for some $j < i$ or $p_i = p_0$, and $p'_{n+1} = p'_{i_j}$ for some $j \leq n$ or $p'_{n+1} = p_0$.

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



c1. $DL[\lambda, C \cap\!\!\!\!- \; p, C \cup\!\!\!\!- \; p; \neg C](t)$,

c2. $DL[\lambda, C \cap\!\!\!\!- \; p, D \uplus p, D \cup\!\!\!\!- \; p; \neg C](t)$,

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

c4. $DL[\lambda, C \cap\!\!\!\!- \; p_0, C^0 \uplus p_0, C^0 \cap\!\!\!\!- \; p'_0, C^1 \uplus p_1, C^1 \cap\!\!\!\!- \; p'_1, \ldots,$
$C^n \uplus p_n, C^n \uplus p'_n, D \uplus p_{n+1}, D \cup\!\!\!\!- \; p'_{n+1}; \neg C](t)$,

where for every $i = 0, \ldots, n+1$, $p_i = p'_j$ for some $j < i$ or $p_i = p_0$, and $p'_{n+1} = p'_{i_j}$ for some $j \leq n$ or $p'_{n+1} = p_0$.

Tautologies with Concept Query

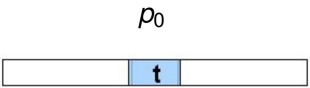
$$DL[\lambda; \neg C](t)$$

Proposition

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

c1. $DL[\lambda, C \Cap p, C \Cup p; \neg C](t)$,

c2. $DL[\lambda, C \Cap p, D \uplus p, D \Cup p; \neg C](t)$,



c3. $DL[\lambda, C \Cap p_0, C^0 \uplus p_0, C^0 \Cap p'_0, C^1 \uplus p_1, C^1 \Cap p'_1, \ldots,$
$C^n \uplus p_n, C^n \Cap p'_n, \boxed{C \Cup p_{n+1}}; \neg C](t)$,

c4. $DL[\lambda, C \Cap p_0, C^0 \uplus p_0, C^0 \Cap p'_0, C^1 \uplus p_1, C^1 \Cap p'_1, \ldots,$
$C^n \uplus p_n, C^n \uplus p'_n, \boxed{D \uplus p_{n+1}, D \Cup p'_{n+1}}; \neg C](t)$,

where for every $i = 0, \ldots, n+1$, $p_i = p'_j$ for some $j < i$ or $p_i = p_0$, and $p'_{n+1} = p'_{i_j}$ 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

c1. $DL[\lambda, C \Cap p, C \Cup p; \neg C](t)$,

c2. $DL[\lambda, C \Cap p, D \uplus p, D \Cup p; \neg C](t)$,

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

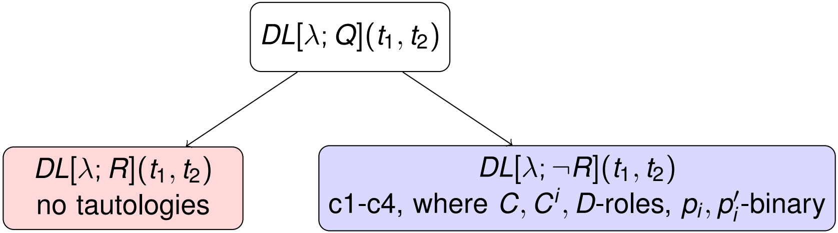
Example

$a = DL[C \Cap p, C' \uplus p, C' \Cap q, C \Cup q; \neg C](c)$ 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:



Example

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

Axiomatization for Tautologies ($\mathcal{K}_{taut}$)

Axioms:

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

a1. $DL[S \Cap p, S \Cup p; \neg S](\mathbf{t})$,

a2. $DL[S \Cap p, S' \uplus p, S' \Cup p; \neg S](\mathbf{t})$,

where $Q \in \{S \sqsubseteq S, S \sqsubseteq \top, \top \not\sqsubseteq \bot\}$, S, S' are distinct.

Rules of Inference:

Expansion

$$\frac{DL[\lambda; Q](\mathbf{t})}{DL[\lambda, \lambda'; Q](\mathbf{t})} \ (e)$$

Increase

$$\frac{DL[\lambda, S \uplus p; Q](\mathbf{t})}{DL[\lambda, S \uplus q, S' \uplus p, S' \Cap q; Q](\mathbf{t})} \ (in_{\uplus})$$

$$\frac{DL[\lambda, S \Cup p; Q](\mathbf{t})}{DL[\lambda, S \Cup q, S' \uplus p, S' \Cap q; Q](\mathbf{t})} \ (in_{\Cup})$$

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 $Y_i = X_{n-i+1}$.

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

$\mathcal{C}$ is *separable* for a if every $IC \in inp_a(CL(\mathcal{C}))$ 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 $Y_i = X_i$;
- $p \subseteq q^-$, if $n = m$ and $Y_i = X_{n-i+1}$.

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

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

Example

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

$\mathcal{C} = \{p_1 \subseteq p_2^-, p_1 \subseteq p_3\}$; $CL(\mathcal{C}) = \mathcal{C}$;
$inp_a(CL(\mathcal{C})) = \{p_1 \subseteq p_2^-\}$; $\mathcal{C}$ is separable for a.

Axiomatization for Tautologies under Inclusion $\mathcal{K}_{taut}^{\subseteq}$

Axioms:

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

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

a2. $DL[S \cap p, S' \uplus q, S' \cup q; \neg S](\mathbf{t})$,

where $q \in \{p, p^-\}$, $Q \in \{S \sqsubseteq S, S \sqsubseteq \top, \top \not\sqsubseteq \bot\}$, S, S' are distinct.

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

Inclusion

$$\frac{DL[\lambda, S \cup p; Q](\mathbf{t}) \quad p \subseteq q}{DL[\lambda, S \cup q; Q](\mathbf{t})} \ (i_1)$$

$$\frac{DL[\lambda, S \uplus p; Q](\mathbf{t}) \quad p \subseteq q}{DL[\lambda, S \uplus q; Q](\mathbf{t})} \ (i_2)$$

Increase

$$\frac{DL[\lambda, S \uplus p; Q](\mathbf{t})}{DL[\lambda, S \uplus q, S' \uplus p^-, S' \cap q^-; Q](\mathbf{t})} \ (in_{\uplus}^{-})$$

$$\frac{DL[\lambda, S \cup p; Q](\mathbf{t})}{DL[\lambda, S \cup q, S' \uplus p^-, S' \cap q^-; Q](\mathbf{t})} \ (in_{\cup}^{-})$$

Example

$$
\begin{aligned}
\Pi = \{&(1)\ so(ch, chile).\\
&(2)\ vi(X) \leftarrow ex(X).\\
&(3)\ sw(X) \leftarrow ex(X), not\ bi(X).\\
&(4)\ ex(X) \leftarrow so(X, Y).\\
&(5)\ no(X) \leftarrow DL[H \uplus vi, H \mathbin{\cup\!\!\!\!\!-} sw, A \mathbin{\cap\!\!\!\!\!-} ex; \neg A](X).
\end{aligned}
$$

(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.

Example

$$
\begin{array}{l}
\Pi = \{(1)\ \mathit{so}(\mathit{ch}, \mathit{chile}). \\
\quad (2)\ \mathit{vi}(X) \leftarrow \mathit{ex}(X). \\
\quad (3)\ \mathit{sw}(X) \leftarrow \mathit{ex}(X), \mathit{not}\ \mathit{bi}(X). \\
\quad (4)\ \mathit{ex}(X) \leftarrow \mathit{so}(X, Y). \\
\quad (5)\ \mathit{no}(X) \leftarrow \mathit{DL}[H \uplus \mathit{vi}, H \mathbin{\cup\!\!\!\!-} \mathit{sw}, A \mathbin{\cap\!\!\!\!-} \mathit{ex}; \neg A](X).
\end{array}
$$

(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

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

(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.

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

Example (cont.)

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

$$\dfrac{\dfrac{\dfrac{DL[H \uplus ex, H \cup ex, A \Cap ex; \neg A](ch)}{DL[H \uplus ex, H \cup ex, A \Cap ex; \neg A](ch)} \quad ex \subseteq vi}{DL[H \uplus vi, H \cup ex, A \Cap ex; \neg A](ch)}\,(i_2) \quad ex \subseteq sw}{DL[H \uplus vi, H \cup sw, A \Cap ex; \neg A](ch)}\,(i_1)$$

Example (cont.)

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

$$\cfrac{\cfrac{\cfrac{DL[H \uplus ex, H \uminus ex, A \cap^{-} ex; \neg A](ch)}{DL[H \uplus ex, H \uminus ex, A \cap^{-} ex; \neg A](ch)} \quad ex \subseteq vi}{DL[H \uplus vi, H \uminus ex, A \cap^{-} ex; \neg A](ch)}\;(i_2) \quad ex \subseteq sw}{DL[H \uplus vi, H \uminus sw, A \cap^{-} ex; \neg A](ch)}\;(i_1)$$

$DL[H \uplus ex, H \uminus ex, A \cap^{-} ex; \neg A](ch)$ is an axiom **a2** of $\mathcal{K}^{\subseteq}_{taut}$.