

Reasoning with Inconsistent Ontologies Through Argumentation

Carlos Iván Chesñevar

`cic@cs.uns.edu.ar`

`http://cs.uns.edu.ar/~cic`



CONICET and Laboratory of R&D in A.I.
Department of Computer Science and Engineering
Universidad Nacional del Sur
Bahía Blanca, Argentina



Publications

- ➔ S.A. Gómez, C.I. Chesñevar & G.R. Simari. **An Argumentative Approach to Reasoning with Inconsistent Ontologies**. In *Procs. of Knowledge Representation Ontology Workshop (KROW 2008)*, Sydney, Australia. *Conferences in Research and Practice in Information Technology*, Vol. 90, Pag. 11-20. Thomas Meyer and Mehmet A. Orgun, Eds.
- ➔ S.A. Gómez, C.I. Chesñevar & G.R. Simari. **Reasoning with Inconsistent Ontologies through Argumentation**. *Applied Artificial Intelligence*, 24: 1, 102-148, 2010.

Outline

- Motivations
- Description Logics (DL) Ontologies
- Defeasible Logic Programming (DeLP)
- Expressing DL ontologies in DeLP
- DeLP-based ontologies: δ -ontologies
- Some theoretical properties
- Conclusions

Motivations

- ➔ The Semantic Web is a future vision of the web where stored information has exact meaning enabling computers to understand and reason on the basis of such information.
- ➔ The assignment of semantics to web resources is addressed by means of **ontology definitions**.
- ➔ Traditional reasoners cannot deal with **inconsistent ontology definitions**
- ➔ A particular source of inconsistency is related to the use of inconsistent imported ontologies where the knowledge engineer has no authority to correct them.

Motivations

- ➔ There are two main ways of dealing with inconsistency in ontologies:
 - One is to diagnose and repair it when it is encountered
 - Another is to avoid the inconsistency by applying a non-standard inference relation to obtain meaningful answers
- ➔ We propose using defeasible argumentation to focus on the latter.

Motivations

- ➔ Description Logics (DL) are a well-known family of knowledge representation formalisms (Baader et al, 2003)
- ➔ (Grosz et al., 2003) have determined that a subset of DL can be translated into an equivalent subset of Horn-logic.
- ➔ DeLP (García&Simari, 2004; Viglizzo et al 2010) is an argumentative framework based on logic programming capable of dealing with possibly inconsistent knowledge bases codified as a set of Horn-like clauses called **DeLP programs**.

Motivations

- ➔ We propose a framework based on DeLP for representing possibly inconsistent DL ontologies.
- ➔ Our proposal involves mapping DL ontologies into DeLP programs, resulting in δ -ontologies.
- ➔ Reasoning in such ontologies will be carried out by means of a dialectical analysis.
- ➔ Given a DL Σ ontology it will be translated as a DeLP program \mathcal{P} .
- ➔ Given a query ϕ , a dialectical process will be performed to determine if ϕ is warranted w.r.t. \mathcal{P} .

Description Logics (DL)

- ➔ The basic language for representing concepts and roles includes conjunction ($C \sqcap D$), disjunction ($C \sqcup D$), complement ($\neg C$), existential restriction ($\exists R.C$), value restriction ($\forall R.C$), inverse (R^-) and transitive roles (R^+).
- ➔ A **DL ontology** is pair $\Sigma = (T, A)$:
 - The **Tbox** T is a finite set of inclusion ($C \sqsubseteq D$) or equality axioms ($C \equiv D$).
 - The **Abox** A is a finite set of facts of the form $(a:C)$ and $(\langle a,b \rangle:R)$.
- ➔ Ontology $\Sigma_1 = (T_1, A_1)$ about flying animals:

$$T_1 = \left\{ \begin{array}{l} penguin \sqsubseteq bird \\ bird \sqcap broken_wing \sqsubseteq \neg flies \\ bird \sqsubseteq flies \\ super_penguin \sqsubseteq flies \\ super_penguin \sqsubseteq penguin \end{array} \right\} \quad A_1 = \left\{ \begin{array}{l} opus : super_penguin, \\ opus : broken_wing \end{array} \right\}$$

Description Logics (DL)

➔ Inference tasks for Aboxes:

- ➔ Given an ontology (T,A) , **instance checking** refers to determining whether the assertions in the Abox (along with the Tbox T) entail that a particular individual a is an instance of a given concept description C
- ➔ Given an ontology (T,A) , **retrieval** refers to know all individuals a that are instances of a certain concept C .

➔ Racer code for inconsistent ontology $\Sigma_1=(T_1,A_1)$:

```
(signature
  :atomic-concepts (bird penguin flies super_penguin broken_wing)
  :individuals (opus)
)
(implies penguin bird)
(implies (and bird broken_wing) (not flies))
(implies bird flies)
(implies super_penguin flies)
(implies super_penguin penguin)
(instance opus super_penguin)
(instance opus broken_wing)
```

Query:

(individual-instance? opus flies)

Output:

Error: ABox DEFAULT is incoherent.

Defeasible Logic Programming (DeLP)

DL ontology $\Sigma_1 = (T_1, A_1)$

Tbox T_1 :

$penguin \sqsubseteq bird$
 $bird \sqcap broken_wing \sqsubseteq \neg flies$
 $bird \sqsubseteq flies$
 $super_penguin \sqsubseteq flies$
 $super_penguin \sqsubseteq penguin$

Abox A_1 :

$opus : super_penguin$
 $opus : broken_wing$

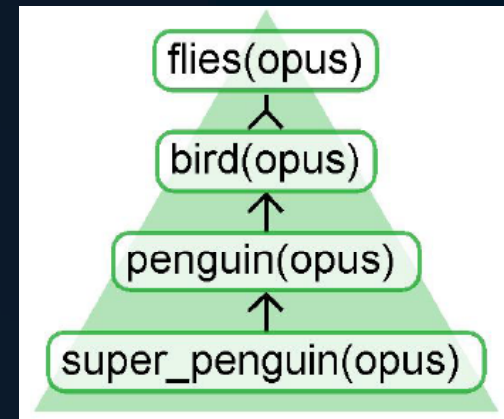


DeLP program $\mathcal{P} = (\Pi, \Delta)$

$$\Pi = \left\{ \begin{array}{l} bird(X) \leftarrow penguin(X). \\ penguin(X) \leftarrow super_penguin(X). \\ super_penguin(opus). \\ broken_wing(opus). \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} \sim flies(X) \prec bird(X), broken_wing(X). \\ flies(X) \prec bird(X). \\ flies(X) \prec super_penguin(X). \end{array} \right\}$$

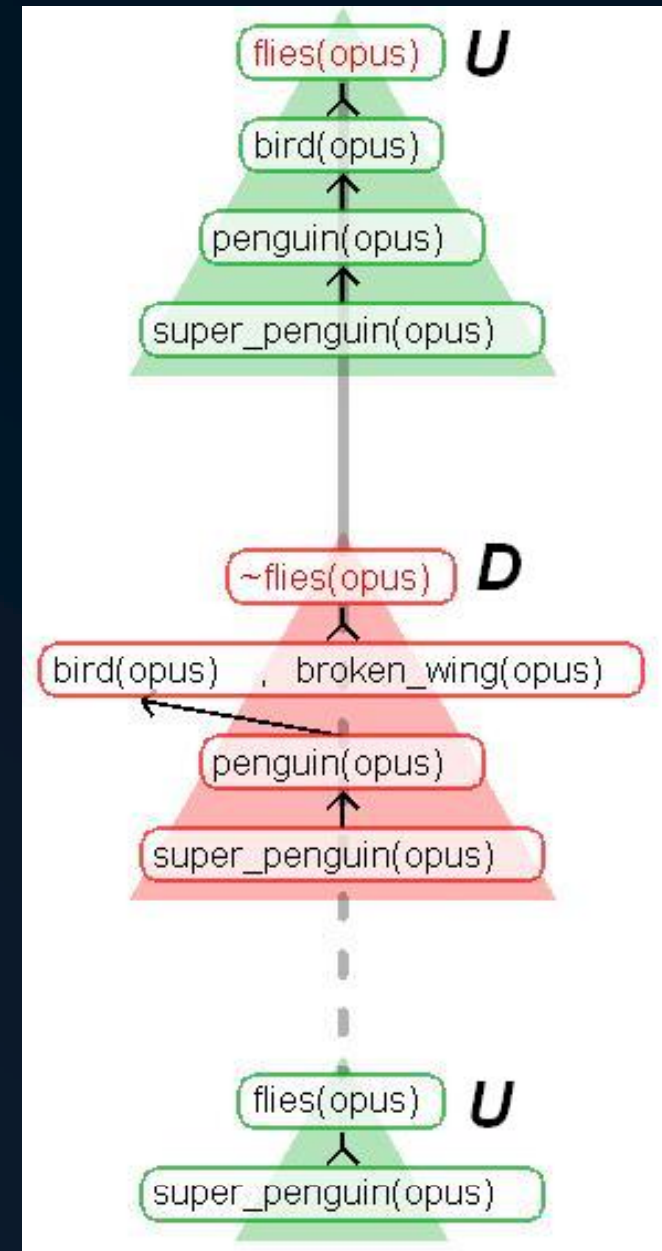
*The set Π is assumed to be non-contradictory
 but $\Pi \cup \Delta$ can derive contradictory literals.*



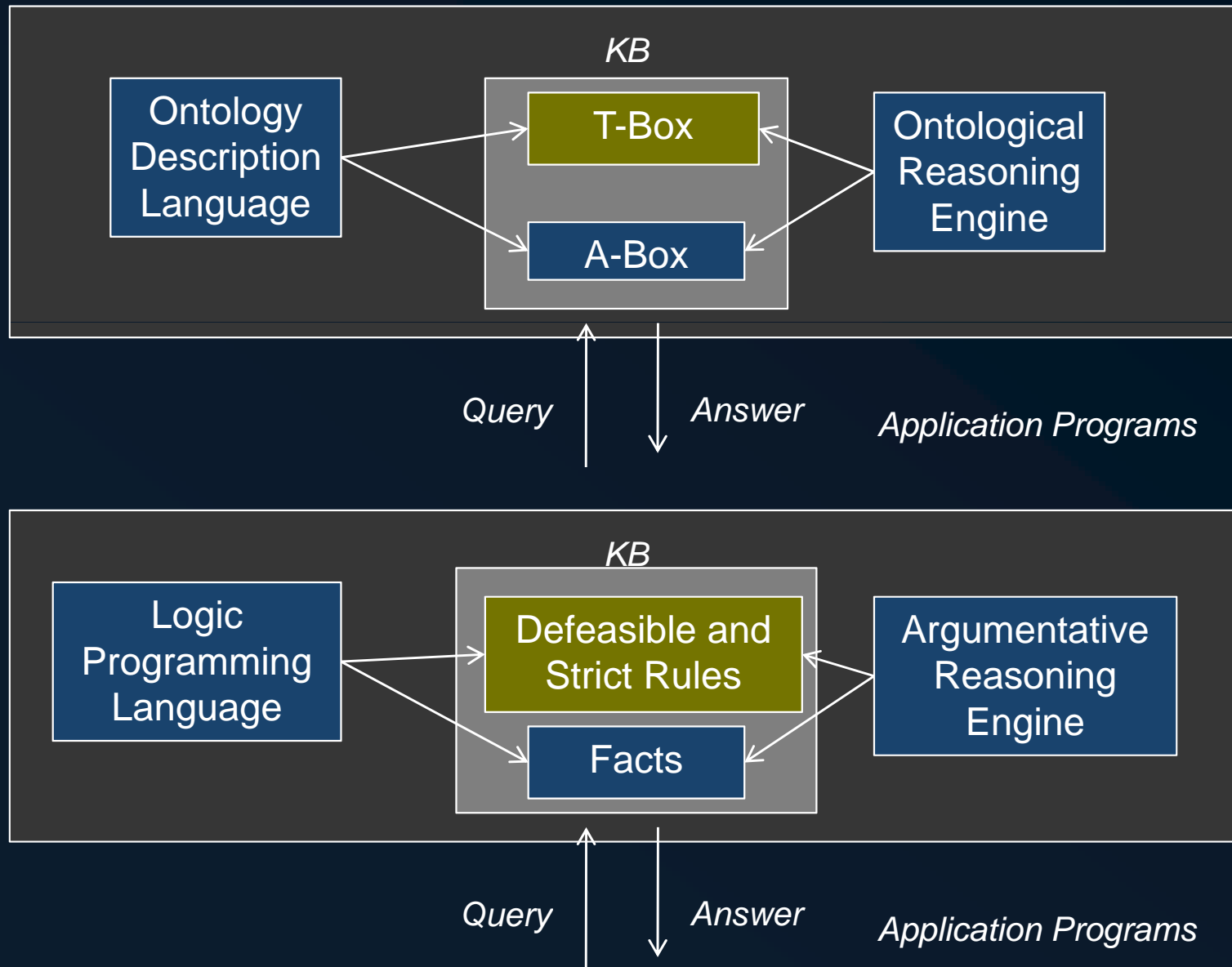
Argument \mathcal{A} supporting “flies(opus)”

Defeasible Logic Programming (DeLP)

- ➔ From a given DeLP program several conflicting arguments may be derived
- ➔ An argument \mathcal{A} defeats another argument \mathcal{B} if \mathcal{A} and \mathcal{B} are in conflict and \mathcal{A} is preferred or unrelated to \mathcal{B} according a preference criterion.
- ➔ To determine if an argument $\langle \mathcal{A}, q \rangle$ is ultimately accepted or **warranted**, a **dialectical tree** is automatically built by the DeLP inference engine.
- ➔ Four possible answers:
yes, no, undefined, unknown



A generic DL-based architecture



Expressing DL ontologies in DeLP

- ➔ Constraints for translating DL to logic programming:
 - Conjunction and universal restrictions appearing in the right-hand side of inclusion axioms can be mapped to head of rules (called \mathcal{L}_h -classes).
 - Conjunction, disjunction and existential restriction can be mapped to rule bodies whenever they occur in the left-hand side of inclusion axioms (called \mathcal{L}_b -classes).
 - As equality axioms ($C \equiv D$) are interpreted as two inclusion axioms ($C \sqsubseteq D$) and ($D \sqsubseteq C$), they must belong to the intersection of \mathcal{L}_h and \mathcal{L}_b (called \mathcal{L}_{hb} -classes).

$penguin \sqsubseteq bird \sqcap swimmer$
 $bird \sqcap broken_wing \sqsubseteq \neg flies$



$bird(X) \leftarrow penguin(X).$
 $swimmer(X) \leftarrow penguin(X).$
 $\sim flies(X) \leftarrow bird(X), broken_wing(X).$

Expressing DL ontologies as DeLP strict rules

$\mathcal{T}_{\sqcap}^*(\{C \sqsubseteq D\})$	$=df$	$\{ T_h(D, X) \leftarrow T_b(C, X) \}$, if C is an \mathcal{L}_b -class and D an \mathcal{L}_h -class
$\mathcal{T}_{\sqcap}^*(\{C \equiv D\})$	$=df$	$\mathcal{T}_{\sqcap}^*(\{C \sqsubseteq D\}) \cup \mathcal{T}_{\sqcap}^*(\{D \sqsubseteq C\})$, if C and D are \mathcal{L}_{hb} -classes
$\mathcal{T}_{\sqcap}^*(\{\top \sqsubseteq \forall P.D\})$	$=df$	$\{ T_h(D, Y) \leftarrow P(X, Y) \}$, if D is an \mathcal{L}_h -class
$\mathcal{T}_{\sqcap}^*(\{\top \sqsubseteq \forall P^-.D\})$	$=df$	$\{ T_h(D, X) \leftarrow P(X, Y) \}$, if D is an \mathcal{L}_h -class
$\mathcal{T}_{\sqcap}^*(\{a : D\})$	$=df$	$\{ T_h(D, a) \}$, if D is an \mathcal{L}_h -class
$\mathcal{T}_{\sqcap}^*(\{\langle a, b \rangle : P\})$	$=df$	$\{ P(a, b) \}$
$\mathcal{T}_{\sqcap}^*(\{P \sqsubseteq Q\})$	$=df$	$\{ Q(X, Y) \leftarrow P(X, Y) \}$
$\mathcal{T}_{\sqcap}^*(\{P \equiv Q\})$	$=df$	$\left\{ \begin{array}{l} Q(X, Y) \leftarrow P(X, Y) \\ P(X, Y) \leftarrow Q(X, Y) \end{array} \right\}$
$\mathcal{T}_{\sqcap}^*(\{P \equiv Q^-\})$	$=df$	$\left\{ \begin{array}{l} Q(X, Y) \leftarrow P(Y, X) \\ P(Y, X) \leftarrow Q(X, Y) \end{array} \right\}$
$\mathcal{T}_{\sqcap}^*(\{P^+ \sqsubseteq P\})$	$=df$	$\{ P(X, Z) \leftarrow P(X, Y) \wedge P(Y, Z) \}$
$\mathcal{T}_{\sqcap}^*(\{s_1, \dots, s_n\})$	$=df$	$\bigcup_{i=1}^n \mathcal{T}_{\sqcap}^*(\{s_i\})$, if $n > 1$
where:		
$T_h(A, X)$	$=df$	$A(X)$
$T_h((C \sqcap D), X)$	$=df$	$T_h(C, X) \wedge T_h(D, X)$
$T_h((\forall R.C), X)$	$=df$	$T_h(C, Y) \leftarrow R(X, Y)$
$T_b(A, X)$	$=df$	$A(X)$
$T_b((C \sqcap D), X)$	$=df$	$T_b(C, X) \wedge T_b(D, X)$
$T_b((C \sqcup D), X)$	$=df$	$T_b(C, X) \vee T_b(D, X)$
$T_b((\exists R.C), X)$	$=df$	$R(X, Y) \wedge T_b(C, Y)$

A similar transformation T_{Δ} is defined for defeasible rules

Expressing DL ontologies as DeLP strict rules

- ➔ As DeLP is based on SLD-derivation of literals, simple translation of DL sentences to DeLP strict rules does not allow to infer negative information by modus tollens.
- ➔ E.g. “ $C \sqsubseteq D$ ” (all C’s are D’s) is translated as “ $D(X) \leftarrow C(X)$ ”, DeLP is not able to derive “ $\sim C(a)$ ” from “ $\sim D(a)$ ”.
- ➔ Given “ $C_1 \sqcap \dots \sqcap C_n \sqsubseteq D$ ” we propose including all **transposes** of the strict rule “ $D(X) \leftarrow C_1(X), \dots, C_n(X)$ ” (Caminada&Amgoud,2007)

Let $r = H \leftarrow B_1, B_2, B_3, \dots, B_{n-1}, B_n$ be a DeLP strict rule.

The **set of transposes of rule r** is defined as:

$$Trans(r) = \left\{ \begin{array}{l} H \leftarrow B_1, B_2, \dots, B_{n-1}, B_n \\ \overline{B_1} \leftarrow \overline{H}, B_2, B_3, \dots, B_{n-1}, B_n \\ \overline{B_2} \leftarrow \overline{H}, B_1, B_3, \dots, B_{n-1}, B_n \\ \overline{B_3} \leftarrow \overline{H}, B_1, B_2, \dots, B_{n-1}, B_n \\ \dots \\ \overline{B_{n-1}} \leftarrow \overline{H}, B_1, B_2, B_3, \dots, B_n \\ \overline{B_n} \leftarrow \overline{H}, B_1, B_2, \dots, B_{n-1} \end{array} \right\}$$

$$\mathcal{T}_{\sqcap}(\{t_1, \dots, t_n\}) = \bigcup_{i=1, \dots, n} Trans(\mathcal{T}_{\sqcap}^*(t_i))$$

DeLP-based ontologies: δ -ontologies

- ➔ Let C be an \mathcal{L}_b -class, D an \mathcal{L}_h -class, A, B \mathcal{L}_{hb} -classes, P, Q properties, a, b individuals
- ➔ Let T be a set of inclusion and equality sentences of the form $C \sqsubseteq D$, $A \equiv B$, $T \sqsubseteq \forall P.D$, $T \sqsubseteq \forall P^-.D$, $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^-$, or $P^+ \sqsubseteq P$ **such that T can be partitioned into two disjoint sets T_S and T_D .**
- ➔ Let A be a set of assertions disjoint with T of the form $(a:C)$ or $(\langle a, b \rangle:P)$.
- ➔ A **δ -ontology** is a tuple (T_S, T_D, A) .
- ➔ The set T_S is called the **strict terminology** (or *Sbox*), T_D the **defeasible terminology** (or *Dbox*) and A the **assertional box** (or *Abox*).

$$\begin{array}{l}
 T_S = \left\{ \begin{array}{l} \text{penguin} \sqsubseteq \text{bird} \\ \text{super_penguin} \sqsubseteq \text{flies} \\ \text{super_penguin} \sqsubseteq \text{penguin} \end{array} \right\} \\
 T_D = \left\{ \begin{array}{l} \text{bird} \sqcap \text{broken_wing} \sqsubseteq \neg \text{flies} \\ \text{bird} \sqsubseteq \text{flies} \end{array} \right\} \\
 A = \left\{ \begin{array}{l} \text{opus} : \text{super_penguin}, \\ \text{opus} : \text{broken_wing} \end{array} \right\}
 \end{array}$$

DeLP-based ontologies: δ -ontologies

➔ Interpretation of a δ -ontology:

- Let $\Sigma = (\mathcal{T}_S, \mathcal{T}_D, A)$ be a δ -ontology.
- The **interpretation** of Σ is a DeLP program
- $\mathcal{P} = \mathcal{T}(\Sigma) = (\mathcal{T}_{\Pi}(\mathcal{T}_S) \cup \mathcal{T}_{\Pi}(A), \mathcal{T}_{\Delta}(\mathcal{T}_D))$

➔ Internal coherence in Aboxes. Consistency of Aboxes w.r.t. Tboxes:

- The Abox A is **internally coherent** iff there are no pair of assertions $a:C$ and $a:\neg C$.
- The Abox A is **consistent** w.r.t. the terminology \mathcal{T}_S iff it is not possible to derive two literals $C(a)$ and $\sim C(a)$ from $\mathcal{T}_{\Pi}(\mathcal{T}_S) \cup \mathcal{T}_{\Pi}(A)$.

Membership in δ -ontologies

➔ Potential, justified and strict membership of an individual to a class:

- Let $\Sigma = (T_S, T_D, A)$ be a δ -ontology, C a class name, a an individual, and $\mathcal{P} = \mathcal{T}(\Sigma)$.
- a **potentially belongs** to C iff there is an argument $\langle \mathcal{A}, C(a) \rangle$ w.r.t. \mathcal{P}
- a **justifiedly belongs** to C iff there is a warranted argument $\langle \mathcal{A}, C(a) \rangle$ w.r.t. \mathcal{P}
- a **strictly belongs** to class C iff there is an argument $\langle \emptyset, C(a) \rangle$ w.r.t. \mathcal{P}

➔ Properties:

- Strict membership implies justified membership, which implies potential membership of individuals to concepts.
- It cannot be the case that an individual a strictly or justifiedly belongs to concept C and $\neg C$ simultaneously.

DL retrieval problems in δ -ontologies

➔ Open retrieval:

➤ Given a δ -ontology S and a class C , find all individuals which are instances of C

➤ Solution: Find all individuals a s.t. there exists a warranted argument $\langle \mathcal{A}, C(a) \rangle$ w.r.t. $\mathcal{P} = \mathcal{T}(\Sigma)$.

➔ Retrieval of all classes:

➤ Given a δ -ontology S and an individual a , find all named classes C s.t. a is an instance of C .

➤ Solution: Find all classes C s.t. there exists a warranted argument $\langle \mathcal{A}, C(a) \rangle$ w.r.t. $\mathcal{P} = \mathcal{T}(\Sigma)$.

➔ Property:

➤ The running time of the processes for “open retrieval” and “retrieval of all classes” is finite.

Some theoretical properties

➔ Relation $|\approx$:

- ➔ Let $|\approx$ be relation “justified membership” of instances to concepts, $\Sigma |\approx \phi$ corresponds to a Yes answer to query ϕ w.r.t. program $\mathcal{T}(\Sigma)$.

➔ Epistemic status of an answer (Huang et al. 2005):

- ➔ Given an ontology Σ and a query ϕ , the answer to ϕ will have one of the four epistemic states:

1. **Over-determined:** $\Sigma |\approx \phi$ and $\Sigma |\approx \neg\phi$
2. **Accepted:** $\Sigma |\approx \phi$ and $\text{not}(\Sigma |\approx \neg\phi)$
3. **Rejected:** $\text{not}(\Sigma |\approx \phi)$ and $\Sigma |\approx \neg\phi$
4. **Undetermined:** $\text{not}(\Sigma |\approx \phi)$ and $\text{not}(\Sigma |\approx \neg\phi)$

➔ Property:

- ➔ Let Σ be a δ -ontology.
Then the answer to a query ϕ is never over-determined.

Some theoretical properties

➔ Soundness, Consistency, Meaningfulness (Huang et al. 2005):

- An inconsistency reasoner $|\approx$ is **sound** if the formulas that follow from an inconsistency theory Σ follow from a consistent subtheory of Σ using classical reasoning.
- An inconsistency reasoner $|\approx$ is **consistent** iff $\Sigma |\approx \phi$ implies $\text{not}(\Sigma |\approx \neg\phi)$.
- An answer given by an inconsistency reasoner is **meaningful** iff it is consistent and sound.
- An inconsistency reasoner is said to be **meaningful** iff all of its answers are meaningful.

➔ Property:

- $|\approx$ is a sound, consistent and meaningful inconsistency reasoner.

Conclusions

- ➔ We have presented a framework for reasoning with inconsistent DL ontologies
- ➔ Our proposal involves expressing a DL ontology Σ as a DeLP program $\mathcal{T}(\Sigma)$.
- ➔ Given a query ϕ wrt an inconsistent ontology Σ , a dialectical analysis is performed on the DeLP program $\mathcal{T}(\Sigma)$ where all arguments in favor and against ϕ 's acceptance are taken into account.
- ➔ Research on how to automate the partition between T_S and T_D is being pursued.

References

- ➔ (Grosz et al. 2003) B. Grosz, I. Horrocks, R. Volz & S. Decker. **Description Logic Programs: Combining Logic Programs with Description Logic Programs.** *WWW2003*, Budapest, Hungary, May 20-24, 2003.
- ➔ (Huang et al. 2005) Z. Huang, F. van Harmelen & A. ten Teije. **Reasoning with Inconsistent Ontologies.** *19th Int. Joint Conf. on Artificial Intelligence*. 454-459, Edinburgh, Scotland, 2005.