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

COST Action IC0801 Meeting / Crete, Greece, June 2010

• Motivations
• Description Logics (DL) Ontologies
• Defeasible Logic Programming (DeLP)
• Expressing DL ontologies in DeLP
• DeLP-based ontologies: $\delta$-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.
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).

- (Grosof 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 $\delta$-ontologies.
- Reasoning in such ontologies will be carried out by means of a dialectical analysis.
- Given a DL $\Sigma$ ontology it will be translated as a DeLP program $\mathcal{P}$.
- Given a query $\phi$, a dialectical process will be performed to determine if $\phi$ 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 a 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 \((a,b):R\).

- Ontology \(\Sigma_1 = (T_1, A_1)\) about flying animals:

\[
T_1 = \begin{cases}
penguin \sqsubseteq bird \\
bird \sqcap \text{broken}_\text{wing} \sqsubseteq \neg \text{flies} \\
bird \sqsubseteq \text{flies} \\
\text{super}_\text{penguin} \sqsubseteq \text{flies} \\
\text{super}_\text{penguin} \sqsubseteq \text{penguin}
\end{cases}
\]

\[
A_1 = \begin{cases}
\text{opus} : \text{super}_\text{penguin}, \\
\text{opus} : \text{broken}_\text{wing}
\end{cases}
\]
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 \(\Sigma_1 = (T_1, A_1)\):

```lisp
(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)

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

\[
\begin{align*}
\Pi &= \{ \\
& \quad \text{bird}(X) \leftarrow \text{penguin}(X). \\
& \quad \text{penguin}(X) \leftarrow \text{super}_\text{penguin}(X). \\
& \quad \text{super}_\text{penguin}(\text{opus}). \\
& \quad \text{broken}_\text{wing}(\text{opus}). \\
& \quad \sim \text{flies}(X) \leftarrow \text{bird}(X), \text{broken}_\text{wing}(X). \\
& \quad \text{flies}(X) \leftarrow \text{bird}(X). \\
& \quad \text{flies}(X) \leftarrow \text{super}_\text{penguin}(X). \\
\} \\
\Delta &= \{ \\
& \quad \text{flies}(\text{opus}). \\
\} 
\end{align*}
\]

The set \( \Pi \) is assumed to be non-contradictory but \( \Pi \cup \Delta \) can derive contradictory literals.

\[ \text{Argument } \mathcal{A} \text{ supporting } \text{flies}(\text{opus}) \]
Defeasible Logic Programming (DeLP)

- From a given DeLP program several conflicting arguments may be derived.
- An argument $A$ defeats another argument $B$ if $A$ and $B$ are in conflict and $A$ is preferred or unrelated to $B$ according to a preference criterion.
- To determine if an argument $<A,q>$ 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

Reasoning with Inconsistent Ontologies Through Argumentation
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 $L_h$-classes).
  - Conjunction, disjunction and existential restriction can be mapped to rule bodies whenever they occur in the left-hand side of inclusion axioms (called $L_b$-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 $L_h$ and $L_b$ (called $L_{hb}$-classes).

\[
\text{penguin} \sqsubseteq \text{bird} \sqcap \text{swimmer} \\
\text{bird} \sqcap \text{broken\_wing} \sqsubseteq \neg \text{flies} \\
\]

\[
\text{bird}(X) \leftarrow \text{penguin}(X). \\
\text{swimmer}(X) \leftarrow \text{penguin}(X). \\
\neg \text{flies}(X) \leftarrow \text{bird}(X), \text{broken\_wing}(X).
\]
Expressing DL ontologies as DeLP strict rules

\[
\begin{align*}
T^*_n(\{C \sqsubseteq D\}) &= \text{df} \set{T_h(D, X) \leftarrow T_b(C, X)} \text{, if } C \text{ is an } \mathcal{L}_b\text{-class and } D \text{ an } \mathcal{L}_h\text{-class} \\
T^*_n(\{C \equiv D\}) &= \text{df} \set{T_h(D, Y) \leftarrow P(X, Y)} \text{, if } D \text{ is an } \mathcal{L}_h\text{-class} \\
T^*_n(\{\top \sqsubseteq \forall P.D\}) &= \text{df} \set{T_h(D, X) \leftarrow P(X, Y)} \text{, if } D \text{ is an } \mathcal{L}_h\text{-class} \\
T^*_n(\{\top \sqsubseteq \forall P^-.D\}) &= \text{df} \set{T_h(D, a)} \text{, if } D \text{ is an } \mathcal{L}_h\text{-class} \\
T^*_n(\{a : D\}) &= \text{df} \set{P(a, b)} \\
T^*_n(\{\langle a, b \rangle : P\}) &= \text{df} \set{Q(X, Y) \leftarrow P(X, Y)} \\
T^*_n(\{P \sqsubseteq Q\}) &= \text{df} \set{Q(X, Y) \leftarrow P(X, Y)} \\
T^*_n(\{P \equiv Q\}) &= \text{df} \set{Q(X, Y) \leftarrow P(X, Y)} \\
T^*_n(\{P \equiv Q^-\}) &= \text{df} \set{Q(X, Y) \leftarrow P(X, Y)} \\
T^*_n(\{P^+ \sqsubseteq P\}) &= \text{df} \set{P(X, Z) \leftarrow P(X, Y) \land P(Y, Z)} \\
T^*_n(\{s_1, \ldots, s_n\}) &= \text{df} \bigcup_{i=1}^n T^*_n(\{s_i\}), \text{ if } n > 1
\end{align*}
\]

where:

\[
\begin{align*}
T_h(A, X) &= \text{df} A(X) \\
T_h((C \sqcap D), X) &= \text{df} T_h(C, X) \land T_h(D, X) \\
T_h((\forall R.C), X) &= \text{df} T_h(C, Y) \land T_h(R(X, Y)) \\
T_b(A, X) &= \text{df} T_b(C, X) \land T_b(D, X) \\
T_b((C \sqcap D), X) &= \text{df} T_b(C, X) \land T_b(D, X) \\
T_b((\forall R.C), X) &= \text{df} T_b(C, X) \land T_b(D, X) \\
T_b((\exists R.C), X) &= \text{df} R(X, Y) \land T_b(C, Y)
\end{align*}
\]

A similar transformation \(T_\Delta\) 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 ⊆ D" (all C's are D's) is translated as "D(X) ← C(X)". DeLP is not able to derive "¬C(a)" from "¬D(a)".
- Given "C₁ ⊨ ... ⊨ Cₙ ⊆ D" we propose including all transposes of the strict rule "D(X) ← C₁(X), ..., Cₙ(X)" (Caminada&Amgoud, 2007)

Let \( r = H ← B₁, B₂, B₃, \ldots, Bₙ₋₁, Bₙ \) be a DeLP strict rule.

The set of transposes of rule \( r \) is defined as:

\[
\text{Trans}(r) = \left\{ \begin{array}{l}
H ← B₁, B₂, \ldots, Bₙ₋₁, Bₙ \\
\overline{B₁} ← \overline{H}, B₂, B₃, \ldots, Bₙ₋₁, Bₙ \\
\overline{B₂} ← \overline{H}, B₁, B₃, \ldots, Bₙ₋₁, Bₙ \\
\overline{B₃} ← \overline{H}, B₁, B₂, \ldots, Bₙ₋₁, Bₙ \\
\vdots \\
\overline{Bₙ₋₁} ← \overline{H}, B₁, B₂, B₃, \ldots, Bₙ \\
\overline{Bₙ} ← \overline{H}, B₁, B₂, \ldots, Bₙ₋₁
\end{array} \right\}
\]

\[
\mathcal{T}_\cap(\{t₁, \ldots, tₙ\}) = \bigcup_{i=1,...,n} \text{Trans}(\mathcal{T}_\cap^*(tᵢ))
\]
DeLP-based ontologies: $\delta$-ontologies

- Let $C$ be an $L_b$-class, $D$ an $L_h$-class, $A$, $B$ $L_{hb}$-classes, $P,Q$ properties, $a,b$ individuals.
- Let $T$ be a set of inclusion and equality sentences of the form $C \subseteq D$, $A \equiv B$, $\top \subseteq \forall P.D$, $\top \subseteq \forall P^- D$, $P \subseteq Q$, $P \equiv Q$, $P \equiv Q^-$, or $P^+ \subseteq P$ 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 $\delta$-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 $Abox$).

\[
T_S = \{ \text{penguin \(\subseteq\) bird}, \text{super\_penguin \(\subseteq\) flies} \}
\]
\[
T_D = \{ \text{super\_penguin \(\subseteq\) penguin}, \text{bird \(\cap\) broken\_wing \(\subseteq\) \neg flies} \}
\]
\[
A = \{ \text{opus : super\_penguin}, \text{opus : broken\_wing} \}
\]
DeLP-based ontologies: δ-ontologies

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

**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 $T_S$ iff it is not possible to derive two literals $C(a)$ and $\neg C(a)$ from $\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A)$. 
Membership in $\delta$-ontologies

- Potential, justified and strict membership of an individual to a class:
  - Let $\Sigma = (T_S, T_D, A)$ be a $\delta$-ontology, $C$ a class name, $a$ an individual, and $\mathcal{P} = T(\Sigma)$.
  - $a$ potentially belongs to $C$ iff there is an argument $\langle A, C(a) \rangle$ w.r.t. $\mathcal{P}$.
  - $a$ justifiedly belongs to $C$ iff there is a warranted argument $\langle A, C(a) \rangle$ 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 $\delta$-ontologies

- **Open retrieval:**
  - Given a $\delta$-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 $\langle A, C(a) \rangle$ w.r.t. $P = T(\Sigma)$.

- **Retrieval of all classes:**
  - Given a $\delta$-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 $\langle A, C(a) \rangle$ w.r.t. $P = T(\Sigma)$.

- **Property:**
  - The running time of the processes for “open retrieval” and “retrieval of all classes” is finite.
Some theoretical properties

- **Relation \( \models \approx \):**
  - Let \( \models \approx \) be relation “justified membership” of instances to concepts, \( \Sigma \models \approx \phi \) corresponds to a Yes answer to query \( \phi \) w.r.t. program \( T(\Sigma) \).

- **Epistemic status of an answer** (Huang et al. 2005):
  - Given an ontology \( \Sigma \) and a query \( \phi \), the answer to \( \phi \) will have one of the four epistemic states:
    1. **Over-determined:** \( \Sigma \models \approx \phi \) and \( \Sigma \models \approx \neg \phi \)
    2. **Accepted:** \( \Sigma \models \approx \phi \) and \( \neg (\Sigma \models \approx \neg \phi) \)
    3. **Rejected:** \( \neg (\Sigma \models \approx \phi) \) and \( \Sigma \models \approx \neg \phi \)
    4. **Undetermined:** \( \neg (\Sigma \models \approx \phi) \) and \( \neg (\Sigma \models \approx \neg \phi) \)

- **Property:**
  - Let \( \Sigma \) be a \( \delta \)-ontology.
  - Then the answer to a query \( \phi \) is never over-determined.
Some theoretical properties

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

- An inconsistency reasoner $\vdash \approx$ is **sound** if the formulas that follow from an inconsistency theory $\Sigma$ follow from a consistent subtheory of $\Sigma$ using classical reasoning.

- An inconsistency reasoner $\vdash \approx$ is **consistent** iff $\Sigma \vdash \approx \phi$ implies not($\Sigma \vdash \approx \lnot \phi$).

- 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:**

- $\vdash \approx$ 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 $\Sigma$ as a DeLP program $T(\Sigma)$.
- Given a query $\phi$ wrt an inconsistent ontology $\Sigma$, a dialectical analysis is performed on the DeLP program $\Pi(\Sigma)$ where all arguments in favor and against $\phi$’s acceptance are taken into account.
- Research on how to automate the partition between $T_S$ and $T_D$ is being pursued.