Knowledge Representation and Description Logic
Part 3

Renata Wassermann
renata@ime.usp.br
Computer Science Department
University of São Paulo

September 2014 — IAOA School — Vitória
\textbf{ALC}

\[ C, D \quad \rightarrow \quad A \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C \]
The family of DLs

**EL**

\[ C, D \rightarrow A \mid T \mid C \sqcap D \mid \exists R.C \]

Looks rather inexpressive, but: SNOMED, Galen, etc.
The family of DLs

Below $ALC$
The family of DLs

Below $ALC$

**DL-Lite**

\[ B \rightarrow A | \top | \exists R | \exists R^{-1} \text{ inverse of roles (isPetOf = hasPet}^{-1}) \]
**DL-Lite**

\[ B \rightarrow A \mid T \mid \exists R \mid \exists R^{-1} \text{ inverse of roles (isPetOf = hasPet}^{-1}) \]

\[ C, D \rightarrow B \mid \neg B \mid C \sqcap D \]
**SHIF**

\[ S = \mathcal{ALC} + \text{transitive roles} \]
\[ H = \text{role hierarchies} \]
\[ I = \text{inverse of roles} \]
\[ F = \text{functional roles} \]
Above $\mathcal{ALC}$

$SHIF$

$S = \mathcal{ALC} +$ transitive roles (ancestor)

$H = \text{role hierarchies}$

$I = \text{inverse of roles}$

$F = \text{functional roles}$
\[ S = ALC + \text{ transitive roles (ancestor)} \]
\[ H = \text{role hierarchies (hasSon} \sqsubseteq \text{hasChild)} \]
\[ I = \text{inverse of roles} \]
\[ F = \text{functional roles} \]
The family of DLs

Above $\mathcal{ALC}$

$SHIF$

$S = \mathcal{ALC} +$ transitive roles (ancestor)

$H = \text{role hierarchies (hasSon} \sqsubseteq \text{hasChild})$

$I = \text{inverse of roles (isPetOf} = \text{hasPet}^{-1})$

$F = \text{functional roles}$
The family of DLs

Above \textit{ALC}

\textbf{SHIF}

\begin{align*}
S &= \text{\textit{ALC}+ transitive roles (ancestor)} \\
H &= \text{role hierarchies (hasSon} \sqsubseteq \text{hasChild)} \\
I &= \text{inverse of roles (isPetOf} = \text{hasPet}^{-1}) \\
F &= \text{functional roles (isFatherOf)}
\end{align*}
The family of DLs

Above $\mathcal{ALC}$

$\mathcal{SHOIN}$

$S = \mathcal{ALC} +$ transitive roles

$H =$ role hierarchies

$O =$ nominals

$I =$ inverse of roles

$N =$ number restriction
**SHOIN**

\[
S = ALC + \text{transitive roles} \\
H = \text{role hierarchies} \\
O = \text{nominals } \{\text{hogwarts}\} \\
I = \text{inverse of roles} \\
N = \text{number restriction}
\]
SHOIN

$S = \text{ALC} + \text{transitive roles}$

$H = \text{role hierarchies}$

$O = \text{nominals (\{hogwarts\})}$

$I = \text{inverse of roles}$

$N = \text{number restriction (}> 3\text{hasChild})$
S = \textit{ALC} + \text{transitive roles}
R = \text{role chains} + \text{hierarchies}
O = \text{nominals}
I = \text{inverse of roles}
Q = \text{qualified number restriction}
\textbf{SROIQ}

\[ S = \mathcal{ALC} + \text{transitive roles} \]
\[ R = \text{role chains} + \text{hierarchies} \text{ (hasParent} \circ \text{hasBrother} \sqsubseteq \text{hasUncle)} \]
\[ O = \text{nominals} \]
\[ I = \text{inverse of roles} \]
\[ Q = \text{qualified number restriction} \]
**SROIQ**

\[ S = ALC + \text{transitive roles} \]
\[ R = \text{role chains} + \text{hierarchies} (\text{hasParent} \circ \text{hasBrother} \sqsubseteq \text{hasUncle}) \]
\[ O = \text{nominals} \]
\[ I = \text{inverse of roles} \]
\[ Q = \text{qualified number restriction} (\geq 3\text{hasChild.Male}) \]
OWL (1.0)

- Web Ontology Language
- W3C standard for ontologies since 2004
- Based on SHIF and SHOIN
- XML syntax
OWL (1.0)

- Three flavours:
  - OWL Full - Allows for any combination of OWL operators, extends RDF (undecidable)
OWL (1.0)

- Three flavours:
  - **OWL Full** - Allows for any combination of OWL operators, extends RDF (undecidable)
  - **OWL DL** - Restricts constructors, decidable, corresponds to \( SHOIN \).
OWL (1.0)

- Three flavours:
  - OWL Full - Allows for any combination of OWL operators, extends RDF (undecidable)
  - OWL DL - Restricts constructors, decidable, corresponds to $SHOIN$.
  - OWL Lite - Excludes disjoint classes, nominals, cardinality only 0/1. Decidable, corresponds to $SHIF$. 
Classes

- owl:disjointWith
- owl:equivalentClass
- owl:Thing
- owl:Nothing
Boolean combinations

- owl:complementOf
- owl:unionOf
- owl:intersectionOf
Properties

Two kinds of properties:

- **Object** - teaches, advises
- **Datatype** - name, age
Properties

Two kinds of properties:

- **Object** - teaches, advises
- **Datatype** - name, age

```xml
<owl:DatatypeProperty rdf:ID="age">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:ID="taughtBy">
    <owl:domain rdf:resource="Course"/>
    <owl:range rdf:resource="AcademicStaff"/>
    <rdfs:subPropertyOf rdf:resource="involves"/>
</owl:ObjectProperty>
```
Properties

Restrictions:

- `owl:allValuesFrom` universal quantification
- `owl:hasValue` specific value
- `owl:someValuesFrom` existential quantification
- Cardinality `owl:minCardinality` and `owl:maxCardinality`
Special Properties

- `owl:TransitiveProperty` ex.: ancestral
- `owl:SymmetricProperty` ex.: hasSameGradeAs
- `owl:FunctionalProperty` ex.: age, father
- `owl:InverseFunctionalProperty` (two different objects cannot have the same value) ex.: marriedTo
Example OWL-RDF

```xml
<owl:Class rdf:ID="associateProfessor">
  <rdfs:subClassOf rdf:resource="#academicStaffMember"/>
</owl:Class>

<owl:Class rdf:about="associateProfessor">
  <owl:disjointWith rdf:resource="#professor"/>
  <owl:disjointWith rdf:resource="#assistantProfessor"/>
</owl:Class>

<owl:Class rdf:ID="faculty">
  <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```
Example OWL-RDF

```xml
<owl:ObjectProperty rdf:ID="isTaughtBy">
  <rdfs:domain rdf:resource="#course"/>
  <rdfs:range rdf:resource="#academicStaffMember"/>
  <rdfs:subPropertyOf rdf:resource="#involves"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="teaches">
  <rdfs:range rdf:resource="#course"/>
  <rdfs:domain rdf:resource="#academicStaffMember"/>
  <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```
Example OWL-RDF

```xml
<owl:Class rdf:about="#firstYearCourse">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
      <owl:allValuesFrom rdf:resource="#Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
Example OWL-RDF

```xml
<owl:Class rdf:ID="peopleAtUni">
   <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#staffMember"/>
      <owl:Class rdf:about="#student"/>
   </owl:unionOf>
</owl:Class>

<academicStaffMember rdf:ID="949352">
   <uni:age rdf:datatype="&xsd;integer">39</uni:age>
</academicStaffMember>

<course rdf:about="CIT1111">
   <isTaughtBy rdf:resource="949318"/>
   <isTaughtBy rdf:resource="949352"/>
</course>
```
Example OWL-Turtle

:academicStaffMember rdf:type owl:Class ;
    owl:equivalentClass :faculty .

:assistantProfessor rdf:type owl:Class ;
    rdfs:subClassOf :academicStaffMember ;
    owl:disjointWith :associateProfessor .

:associateProfessor rdf:type owl:Class ;
    rdfs:subClassOf :academicStaffMember ;
    owl:disjointWith :professor .
Example OWL-Turtle

:course rdf:type owl:Class ;
    rdfs:subClassOf [ rdf:type owl:Restriction ;
        owl:onProperty :isTaughtBy ;
        owl:minCardinality "1"^^xsd:nonNegativeInteger
    ] .

:faculty rdf:type owl:Class .

:firstYearCourse rdf:type owl:Class ;
    rdfs:subClassOf :course ,
        [ rdf:type owl:Restriction ;
            owl:onProperty :isTaughtBy ;
            owl:allValuesFrom :professor
        ] .
OWL (2.0)

- W3C standard for ontologies since 2009
- Even OWL-Lite is intractable
- OWL-DL 2 is based on $SROIQ$
- Three new profiles for “Lite fragment”
OWL 2.0

- **EL**: polynomial time reasoning for schema and data
  Useful for ontologies with large conceptual part
OWL 2.0

- **EL**: polynomial time reasoning for schema and data
  Useful for ontologies with large conceptual part

- **QL**: fast (logspace) query answering using RDBMs via SQL
  Useful for large datasets already stored in RDBs
OWL 2.0

- **EL**: polynomial time reasoning for schema and data
  - Useful for ontologies with large conceptual part

- **QL**: fast (logspace) query answering using RDBMs via SQL
  - Useful for large datasets already stored in RDBs

- **RL**: fast (polynomial) query answering using rule-extended DBs
  - Useful for large datasets stored as RDF triples
Motivation

- Study the dynamics of ontologies, specially “OWL-like” DL ontologies.
- AGM Belief Revision deals with the problem of adding/removing information in a consistent way.
- AGM is most commonly applied to propositional classical logic and cannot be directly used with DLs.
- How can we adapt AGM so that it can deal with interesting DLs?
In this work

- Show reasons why AGM fails to apply to DLs.
- Adapt Contraction (easy).
- Adapt Revision (less easy).
AGM Belief Revision

Three operations defined to deal with knowledge base dynamics:

- **Expansion** - adding knowledge (possibly inconsistent)
AGM Belief Revision

Three operations defined to deal with knowledge base dynamics:

- **Expansion** - adding knowledge (possibly inconsistent)
- **Contraction** - removing knowledge
AGM Belief Revision

Three operations defined to deal with knowledge base dynamics:

- **Expansion** - adding knowledge (possibly inconsistent)
- **Contraction** - removing knowledge
- **Revision** - adding knowledge consistently
AGM Belief Revision

Three operations defined to deal with knowledge base dynamics:

- **Expansion** - adding knowledge (possibly inconsistent)
- **Contraction** - removing knowledge
- **Revision** - adding knowledge consistently

Revision usually defined in terms of contraction:

\[ K \ast \alpha = (K - \neg\alpha) + \alpha \]
AGM Theory

For contraction and revision:

- **Rationality Postulates**
AGM Theory

For contraction and revision:

- Rationality Postulates
- Construction
AGM Theory

For contraction and revision:
- **Rationality Postulates**
- **Construction**
- **Representation Theorem** (postulates $\Leftrightarrow$ construction)
AGM Theory

For contraction and revision:

- **Rationality Postulates**
- **Construction**
- **Representation Theorem** (postulates ⇔ construction)

**AGM Assumptions:** Tarskian, Compact, Deduction Theorem, Supraclassical.
AGM contraction

(closure) \( K - \alpha = Cn(K - \alpha) \)

(success) If \( \alpha \notin Cn(\emptyset) \) then \( \alpha \notin K - \alpha \)

(inclusion) \( K - \alpha \subseteq K \)

(vacuity) If \( \alpha \notin K \) then \( K - \alpha = K \)

(recovery) \( K \subseteq K - \alpha + \alpha \)

(extensionality) If \( Cn(\alpha) = Cn(\beta) \) then \( K - \alpha = K - \beta \)
Partial-meet contraction - remainder sets

Definition

The remainder set $B \perp A$ is defined as follows. $X \in B \perp A$ if and only if:

1. $X \subseteq B$
2. $X \not\models A$
3. If $X \subset X' \subseteq B$ then $X' \models A$

The remainder set $B \perp A$ is the set of maximal subsets of $B$ that do not imply $A$. 
Partial-meet contraction - selection function

A selection function then chooses some elements of $B \bot A$.

**Definition**

$\gamma$ is a *selection function for B* iff for every set $A$:

1. If $B \bot A \neq \emptyset$ then $\emptyset \neq \gamma(B \bot A) \subseteq B \bot A$.
2. Otherwise $\gamma(B \bot A) = \{B\}$.

Multiple partial meet contraction is then defined as:

$$B - \gamma A = \bigcap \gamma(B \bot A)$$
Kernel contraction - kernels

Definition (kernel)

The *kernel* $B \models A$ is defined as follows. $X \in B \models A$ if and only if:

1. $X \subseteq B$
2. $X \vdash A$
3. If $X' \subset X$ then $X' \not\models A$
Kernel contraction - incision function

Definition

An incision function for $B$ is a function $\sigma$ such that for every $A$:

1. $\sigma(B \perp A) \subseteq \bigcup (B \perp A)$
2. If $\emptyset \neq X \in B \perp A$, then $X \cap \sigma(B \perp A) \neq \emptyset$.

Multiple kernel contraction is defined as:

$$B - \sigma A = B \setminus \sigma(B \perp A)$$
Applying to DL

- AGM cannot be applied to every logic. In particular it can not be applied to SHIF and SHOIN. [Flouris 2006]
- Solution: substitute recovery by relevance

  \[(\text{relevance}) \text{ If } \beta \in K \setminus K - \alpha, \text{ then there is } K' \text{ s. t. } K - \alpha \subseteq K' \subseteq K \text{ and } \alpha \notin Cn(K'), \text{ but } \alpha \in Cn(K' \cup \{\beta\}).\]

- Good property: AGM assumptions + 5 postulates $\Rightarrow$ recovery and relevance are equivalent.
Results - contraction

Representation Theorem [RW06]

If the underlying logic is tarskian and compact, partial meet contraction is equivalent to the AGM postulates with relevance instead of recovery.
Results - contraction

Representation Theorem [RW06]

If the underlying logic is tarskian and compact, partial meet contraction is equivalent to the AGM postulates with relevance instead of recovery.
Results - contraction

Representation Theorem [RW06]

If the underlying logic is tarskian and compact, partial meet contraction is equivalent to the AGM postulates with relevance instead of recovery.

Can we do the same for revision???
AGM Revision

(closure) \( K \ast \alpha = \text{Cn}(K \ast \alpha) \)

(success) \( \alpha \in K \ast a \)

(inclusion) \( K \ast \alpha \subseteq K + \alpha \)

(vacuity) If \( K + \alpha \) is consistent then \( K \ast \alpha = K + \alpha \)

(consistency) If \( \alpha \) is consistent then \( K \ast \alpha \) is consistent.

(extensionality) If \( \text{Cn}(\alpha) = \text{Cn}(\beta) \) then \( K \ast \alpha = K \ast \beta \)
Applying to DL

- Problem: no negation $\Rightarrow$ no Levi identity.
- Solution: Direct constructions.
Applying to DL

- Problem: no negation $\Rightarrow$ no Levi identity.
- Solution: Direct constructions.

**Definition**

$X \in K \downarrow \alpha$ iff $X$ maximal subset of $K$ such that $X \cup \{\alpha\}$ is consistent.
Applying to DL

- Problem: no negation ⇒ no Levi identity.
- Solution: Direct constructions.

**Definition**

\[ X \in K \downarrow \alpha \text{ iff } X \ \text{maximal subset of } K \text{ such that } X \cup \{\alpha\} \text{ is consistent.} \]

**Definition (Revision without negation)**

\[ K \ast_\gamma \alpha = \bigcap \gamma(K \downarrow \alpha) + \alpha \]

where \( \gamma \) selects at least one element of \( K \downarrow \alpha \).
Properties

1. **Inconsistent explosion:** Whenever $K$ is inconsistent, then for all formulas $\alpha$, $\alpha \in Cn(K)$

2. **Distributivity:** For all sets of formulas $X$, $Y$ and $W$, $Cn(X \cup (Cn(Y) \cap Cn(W))) = Cn(X \cup Y) \cap Cn(X \cup W)$
Properties

1. Inconsistent explosion: Whenever $K$ is inconsistent, then for all formulas $\alpha, \alpha \in Cn(K)$

2. Distributivity: For all sets of formulas $X, Y$ and $W$, $Cn(X \cup (Cn(Y) \cap Cn(W))) = Cn(X \cup Y) \cap Cn(X \cup W)$

Representation Theorem [RW09]

If the logic is monotonic and compact and satisfies Inconsistent explosion and Distributivity, then $*$ is a revision without negation iff it satisfies closure, success, inclusion, consistency, relevance and uniformity.

(uniformity) If for all $K' \subseteq K$, $K' \cup \{\alpha\}$ is inconsistent iff $K' \cup \{\beta\}$ is inconsistent then $K \cap K * \alpha = K \cap K * \beta$
Which Logics Satisfy Distributivity?

- Classical logic does.
- But what about DLs?
  - $\mathcal{ALC}$ does not.
  - $\mathcal{ALC}$ with empty $ABox$ does.
  - not many more...
New characterisation

**Representation Theorem [RW14]**

If the logic is monotonic and compact and satisfies Inconsistent explosion and Distributivity, then * is a revision without negation iff it satisfies closure, success, *strong* inclusion, consistency, relevance and uniformity.

\[(\text{strong inclusion}) \quad K \ast \alpha \subseteq (K \cap K \ast \alpha) + \alpha\]

In classical logics this postulate is equivalent to inclusion.
Implementing

- Using Protégé, OWL API, Pellet/HermiT
- Flexibility of choosing operation and properties (postulates)

Protégé

Figure: Pizza ontology modeled using the Protégé tutorial
The Protégé Revision Plugin

Figure: Initial screen of the plug-in with functionality for multiple contraction

Renata Wassermann
Knowledge Representation and Description Logic Part 3
The Protégé Revision Plugin

Figure: Initial screen of the plug-in and example of how to add sentences and call the contraction operation.
The Protégé Revision Plugin

Figure: Ontology after applying package Kernel contraction

Renata Wassermann

Knowledge Representation and Description Logic Part 3