Description Logics and OWL

Antoine Zimmermann

École des mines de Saint-Étienne

20th January 2014, EMSE
Outline

History and Applications

Basic DL
  Syntax
  Semantics

Reasoning and Reasoners

Advanced DL (OWL and more)

Tools
Knowledge Representation

General goal of knowledge representation: “develop formalisms for providing high-level descriptions of the world that can be effectively used to build intelligent applications.”

- “formalisms”: syntax + well-defined semantics + reasoning services
- “high-level descriptions”: which aspects should be represented, which left out?
- “intelligent applications”: are able to infer new knowledge from given knowledge
- “effectively use”: reasoning techniques should allow “usable” implementation
Early formalisms

DLs inherit from previous KR languages:
- Semantic networks (1966)
- Frame-based languages (1970s)
- KL-One (1st “DL-like” system 1985)

Why not FOL? With unrestricted FOL...
- ...structure of knowledge is destroyed ⇒ cannot be exploited for driving the inference
- ...expressive power too high for obtaining decidable and efficient inference problems
- ...inference power may be too low for expressing interesting, but still decidable theories
What are Description Logics?

A family of logic based Knowledge Representation formalisms

- Descendants of semantic networks and KL-ONE
- Describe domain in terms of concepts (classes), roles (properties, relationships) and individuals (instances)

Distinguished by:

- **Formal semantics** (typically model theoretic)
  - Decidable fragments of FOL

- **Provision of inference services**
  - Decision procedures for key problems (satisfiability, subsumption, etc)
  - Implemented systems (highly optimised)
Applications

The fields where DL were applied / had an impact are

- Medical informatics e.g., SNOMED, 450000 concepts about anatomy, diseases, etc.
- Bioinformatics e.g., the Gene Ontology, 17000 concepts.
- Semantic Web (thousands of DL ontologies available)
- Code management (Lassie, 1990s)
- Configuration
- Digital Libraries
- Planning
- Data mining
- DB integration
- Schema mapping
Applications

The fields where DL were applied / had an impact are

- Medical informatics e.g., SNOMED, 450000 concepts about anatomy, diseases, etc.
- Bioinformatics e.g., the Gene Ontology, 17000 concepts.
- Semantic Web (thousands of DL ontologies available)
- Code management (Lassie, 1990s)
- Configuration
- Digital Libraries
- Planning
- Data mining
- DB integration
- Schema mapping
Vocabulary

The DL vocabulary consists of:
- Concepts (classes / unary predicates)
  - E.g., Person, Doctor, HappyParent
- Roles (relations / properties / binary predicates)
  - E.g., hasChild, loves, etc...
- Individuals (instances / constants)
  - E.g., Antoine, DERI, Galway
There are constructs for building complex concepts

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>$A$</td>
<td>Human</td>
</tr>
<tr>
<td>bottom concept</td>
<td>$\bot$</td>
<td>—</td>
</tr>
<tr>
<td>top concept</td>
<td>$\top$</td>
<td>—</td>
</tr>
<tr>
<td>atomic role</td>
<td>$R$</td>
<td>hasFriend</td>
</tr>
<tr>
<td>conjunction</td>
<td>$C \sqcap D$</td>
<td>Human $\sqcap$ Male</td>
</tr>
<tr>
<td>disjunction</td>
<td>$C \sqcup D$</td>
<td>Nice $\sqcup$ Rich</td>
</tr>
<tr>
<td>negation</td>
<td>$\neg C$</td>
<td>$\neg$ Meat</td>
</tr>
<tr>
<td>exists restrict.</td>
<td>$\exists R.C$</td>
<td>$\exists$ hasChildPerson</td>
</tr>
<tr>
<td>value restrict.</td>
<td>$\forall R.C$</td>
<td>$\forall$ hasChildBlond</td>
</tr>
</tbody>
</table>

(for $C$ and $D$ any concept, $R$ atomic role)
DLs Knowledge Base usually the terminological knowledge (universal knowledge) from the assertional (factual) knowledge:

\[
\text{Father} = \text{Man} \sqcap \exists \text{has\_child}.
\]

\[
\text{Human} = \text{Mammal} \sqcap \text{Biped}
\]

\[
\text{Concrete Situation}
\]

\[
\text{John:Human} \sqcap \text{Father}
\]

\[
\text{John has\_child Bill}
\]
TBox / Abox

TBox: concept definition and subsumption axioms:

- Concept definition

  \[ A \equiv C \]

  e.g., Father \equiv Man \cap \exists \text{hasChild}.\text{Human}

- Subsumption axioms

  \[ C_1 \subseteq C_2 \]

  e.g., \exists \text{hasFavourite}.\text{Brewery} \subseteq \exists \text{drinks}.\text{Beer}

ABox: facts about individuals:

- Concept assertions

  \[ a : C \]

  e.g., Antoine : Human \cap Male

- Role assertions

  \[ \langle a_1, a_2 \rangle : R \]

  e.g., \langle Antoine, DERI \rangle : \text{worksIn}
Interpretation

Given a vocabulary (set of atomic terms), a (model-theoretic) interpretation is defined by:

Interpretation domain $\Delta^x$
Given a vocabulary (set of atomic terms), a (model-theoretic) interpretation is defined by:

**Interpretation function** $\mathcal{I}$  
**Interpretation domain** $\Delta^\mathcal{I}$

**Individuals** $i^\mathcal{I} \in \Delta^\mathcal{I}$

- John
- Mary
Interpretation

Given a vocabulary (set of atomic terms), a (model-theoretic) interpretation is defined by:

- **Interpretation function** $\mathcal{I}$
- **Interpretation domain** $\Delta^\mathcal{I}$

**Individuals** $i^\mathcal{I} \in \Delta^\mathcal{I}$
- John
- Mary

**Concepts** $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$
- Lawyer
- Doctor
- Vehicle
Interpretation

Given a vocabulary (set of atomic terms), a (model-theoretic) interpretation is defined by:

**Interpretation function** $\mathcal{I}$

**Interpretation domain** $\Delta^I$

**Individuals** $i^I \in \Delta^I$
- John
- Mary

**Concepts** $C^I \subseteq \Delta^I$
- Lawyer
- Doctor
- Vehicle

**Roles** $r^I \subseteq \Delta^I \times \Delta^I$
- hasChild
- owns
Given a vocabulary (set of atomic terms), a (model-theoretic) interpretation is defined by:

- **Individuals** $i^I \in \Delta^I$
  - John
  - Mary
- **Concepts** $C^I \subseteq \Delta^I$
  - Lawyer
  - Doctor
  - Vehicle
- **Roles** $r^I \subseteq \Delta^I \times \Delta^I$
  - hasChild
  - owns
  - (Lawyer $\cap$ Doctor)
Interpreting complex concepts

Given an interpretation $\langle \Delta^\mathcal{I}, \mathcal{I} \rangle$ of terms only, interpretations of complex concepts are built recursively:

<table>
<thead>
<tr>
<th>Construct</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a^\mathcal{I} \in \Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>$A$</td>
<td>$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\bot^\mathcal{I} = \emptyset$</td>
</tr>
<tr>
<td>$\top$</td>
<td>$\top^\mathcal{I} = \Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>$R$</td>
<td>$R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$</td>
</tr>
<tr>
<td>$C \sqcap D$</td>
<td>$C^\mathcal{I} \cap D^\mathcal{I}$</td>
</tr>
<tr>
<td>$C \sqcup D$</td>
<td>$C^\mathcal{I} \cup D^\mathcal{I}$</td>
</tr>
<tr>
<td>$\neg C$</td>
<td>$\Delta^\mathcal{I} \setminus C^\mathcal{I}$</td>
</tr>
<tr>
<td>$\exists R.C$</td>
<td>${ x \in \Delta^\mathcal{I} \mid \exists y, \langle x, y \rangle \in R^\mathcal{I} \land y \in C^\mathcal{I} }$</td>
</tr>
<tr>
<td>$\forall R.C$</td>
<td>${ x \in \Delta^\mathcal{I} \mid \forall y, \langle x, y \rangle \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I} }$</td>
</tr>
</tbody>
</table>
Satisfaction relation

The satisfaction relation $\models$ is a relation between interpretations and axioms and assertions:

- $\mathcal{I} \models C \subseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$
  (read $\mathcal{I}$ satisfies $C \subseteq D$)
- $\mathcal{I} \models a : C$ iff $a^\mathcal{I} \in C^\mathcal{I}$
- $\mathcal{I} \models \langle a_1, a_2 \rangle : R$ iff $\langle a_1^\mathcal{I}, a_2^\mathcal{I} \rangle \in R^\mathcal{I}$

If $\mathcal{I}$ satisfies all axioms and assertions, it is a model of the Knowledge Base.
DL axioms can be expressed in FOL:

\[ \forall x. \text{Father}(x) \Rightarrow \text{Human}(x) \]  
\[ \land \text{Male}(x) \]  
\[ \land \exists y. \text{hasChild}(x, y) \]  
\[ \land \text{Human}(y) \]  

\[ \text{Father} \sqsubseteq \text{Human} \sqcap \text{Male} \sqcap \exists \text{hasChild}. \text{Human} \]
Inference problems

- **Consistency** (check if ontology meaningful)
  - $O$ consistent $\iff$ there exists a model of $O$
  - $C$ consistent $\iff$ there is a model $\mathcal{I}$ for which $C^\mathcal{I} \neq \emptyset$

- **Subsumption** (structure knowledge, compute taxonomy)
  - $O$ implies $C \sqsubseteq D$? $\iff C^\mathcal{I} \subseteq D^\mathcal{I}$ for all models of $O$

- **Instantiation** (check if a instance of $C$)
  - $O$ implies $a : C$? $\iff a^\mathcal{I} \in C^\mathcal{I}$ for all models of $O$

- **Retrieval** (retrieve individuals $a$ that instantiate $C$)
  - set of $a$ s.t. $a^\mathcal{I} \in C^\mathcal{I}$ for all models of $O$

All problems reducible to consistency.
Properties of reasoning and reasoners

- **Soundness:** when the system gives an answer, it is correct wrt the KB (but it may not give all answers)
- **Completeness:** the system can provide all the correct answers (but it may also provide additional wrong answers)
- **Complexity:** Complexity of the problem and/or of the reasoner in function of the size of the input (usually, a class of complexity)
The general architecture

Knowledge Base

Terminology
Father = Man \( \sqcap \exists \) has_child. T...
Human = Mammal \( \sqcap \) Biped

Concrete Situation
John : Human \( \sqcap \) Father
John has_child Bill
New constructors

How to express that a Person has exactly 2 parents, 1 who is Male, 1 who is Female?

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>nominals ($O$)</td>
<td>${a_1, \ldots, a_n}$</td>
<td>week days ${\text{Monday, Tuesday, ...}}$</td>
</tr>
<tr>
<td>max cardinality ($\mathcal{N}$)</td>
<td>$\leq n.R$</td>
<td>$\leq 1$\text{hasChild}</td>
</tr>
<tr>
<td>min cardinality ($\mathcal{N}$)</td>
<td>$\geq n.R$</td>
<td>$\geq 2$\text{hasParent}</td>
</tr>
<tr>
<td>qualified max card. ($Q$)</td>
<td>$\leq n.RC$</td>
<td>$\leq 1$\text{hasParentMale}</td>
</tr>
<tr>
<td>qualified min card. ($Q$)</td>
<td>$\geq n.RC$</td>
<td>$\geq 1$\text{hasParentFemale}</td>
</tr>
<tr>
<td>Self concept (?)</td>
<td>$\exists R.\text{Self}$</td>
<td>$\exists \text{regulateSelf}$</td>
</tr>
<tr>
<td>inverse role ($\mathcal{H}$)</td>
<td>$R^-$</td>
<td>\text{hasParent}^- is \text{hasChild}</td>
</tr>
<tr>
<td>role chain ($\cdot^\circ$)</td>
<td>$R \circ S$</td>
<td>\text{hasParent} \circ \text{Brother}</td>
</tr>
<tr>
<td>cartesian product (?)</td>
<td>$C \times D$</td>
<td>\text{Elephant} \circ \text{Mice}</td>
</tr>
<tr>
<td>other role const. ($\cdot$, $\sqcap$, $\sqcup$, $\neg$)</td>
<td>$R \sqcap C$</td>
<td>\text{and} $\sqcup$, $\neg$, identity, etc.</td>
</tr>
<tr>
<td>subrole axiom ($\mathcal{H}$)</td>
<td>$R \sqsubseteq S$</td>
<td>\text{hasFriend} $\sqsubseteq$ knows</td>
</tr>
<tr>
<td>transitive role ($\cdot_{R^+}$)</td>
<td>$\text{Trans}(R)$</td>
<td>$\text{Trans}(\text{hasAncestor})$</td>
</tr>
</tbody>
</table>
New constructors: semantics

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a_1, \ldots , a_n}</td>
<td>{a^I_1, \ldots , a^I_n}</td>
</tr>
<tr>
<td>\leq n.R</td>
<td>{x \in \Delta^I \mid #{y \in \Delta^I \mid \langle x, y \rangle \in R^I } \leq n}</td>
</tr>
<tr>
<td>\geq n.R</td>
<td>{x \in \Delta^I \mid #{y \in \Delta^I \mid \langle x, y \rangle \in R^I } \geq n}</td>
</tr>
<tr>
<td>\leq n.RC</td>
<td>{x \in \Delta^I \mid #{y \in \Delta^I \mid \langle x, y \rangle \in R^I \land y \in C^I } \leq n}</td>
</tr>
<tr>
<td>\geq n.RC</td>
<td>{x \in \Delta^I \mid #{y \in \Delta^I \mid \langle x, y \rangle \in R^I \land y \in C^I } \geq n}</td>
</tr>
<tr>
<td>\exists R.Self</td>
<td>{x \in \Delta^I \mid \langle x, x \rangle \in R^I }</td>
</tr>
<tr>
<td>R^-</td>
<td>{\langle x, y \rangle \mid \langle y, x \rangle \in R^I }</td>
</tr>
<tr>
<td>R \circ S</td>
<td>{\langle x, y \rangle \in \Delta^I \times \Delta^I \mid \exists t \in \Delta^I, \langle x, t \rangle \in R^I \land \langle t, y \rangle \in S^I }</td>
</tr>
<tr>
<td>C \times D</td>
<td>{\langle x, y \rangle \in \Delta^I \mid \exists t \in \Delta^I, \langle x, t \rangle \in R^I \land \langle t, y \rangle \in S^I }</td>
</tr>
<tr>
<td>R \cap C</td>
<td>{\langle x, y \rangle \in \Delta^I \mid \exists t \in \Delta^I, \langle x, t \rangle \in R^I \land \langle t, y \rangle \in S^I }</td>
</tr>
<tr>
<td>R \subseteq S</td>
<td>{\langle x, y \rangle \in \Delta^I \mid \exists t \in \Delta^I, \langle x, t \rangle \in R^I \land \langle t, y \rangle \in S^I }</td>
</tr>
<tr>
<td>\text{Trans}(R)</td>
<td>R^I \text{ is transitively closed}</td>
</tr>
</tbody>
</table>
## Complexity

<table>
<thead>
<tr>
<th>P</th>
<th>(co-)NP</th>
<th>PSpace</th>
<th>ExpTime</th>
<th>NExpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{ALN}$</td>
<td>(NP)</td>
<td>$\mathcal{ALCN}$</td>
<td>$\mathcal{ALC}_{\text{reg}}$ add regular roles</td>
<td>$\mathcal{QI}$ still in ExpTime</td>
</tr>
<tr>
<td>without $\sqcup$</td>
<td>without $\exists$, only $\neg A$</td>
<td>(wrt acyc. TBoxes)</td>
<td>$\mathcal{ALC}_u$ add universal role $\mathcal{ALC}$ wrt general TBoxes</td>
<td></td>
</tr>
<tr>
<td>subsumption of $FL_0$ $\boxtimes$ and $\forall$ only</td>
<td>subsumption of $FL_0$ (co-NP) wrt acyc. TBoxes</td>
<td>$\mathcal{ALCNO}$ $\mathcal{ALCO}$ $\mathcal{ALCHIQR}^+$ add role hierarchies</td>
<td>$\mathcal{ALCIO}$ $\mathcal{ALCIOQ}$ $\mathcal{ALC}^-$ $\mathcal{ALC}^{-,n,u}$ $\mathcal{ALCF}$ wrt acyc. TBoxes</td>
<td></td>
</tr>
</tbody>
</table>

- $I$ inverse roles: h-child
- $N$ NRs: $(\geq n$ h-child$)$
- $O$ Qual. NRs: $(\geq n$ h-child Blond$)$
- $O$ nominals: ”John” is a concept
- $F$ feature chain (dis)agreement
- $\cdot_R^+$ declare roles as transitive
- $\neg, \land, \lor$ Boolean ops on roles
More DL

High complexity of DL is worrying... New forms of DL
- $\mathcal{ECL}$ and extensions;
- DL-Lite and extensions;
- DLP and variants.

that restrict
- constructs;
- axioms;
- left/right side of subclass/subproperty axioms;
- datatypes.
**£L and extensions**

£L restricts constructs to:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>( A )</td>
</tr>
<tr>
<td>top concept</td>
<td>( \top )</td>
</tr>
<tr>
<td>atomic role</td>
<td>( R )</td>
</tr>
<tr>
<td>conjunction</td>
<td>( C \sqcap D )</td>
</tr>
<tr>
<td>exists restrict.</td>
<td>( \exists R.C )</td>
</tr>
<tr>
<td>property chains</td>
<td>( R_1 \circ R_2 \sqsubseteq R )</td>
</tr>
</tbody>
</table>

£L++ adds the following constructs:

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottom concept</td>
<td>( \bot )</td>
</tr>
<tr>
<td>nominal</td>
<td>{a}</td>
</tr>
<tr>
<td>concrete domain</td>
<td>( p(f_1, \ldots, f_n) )</td>
</tr>
</tbody>
</table>

Polynomial complexity. Any other typical construct added to £L leads to intractability.
DL-Lite\textsubscript{core} is based on the following grammar:

\[ B ::= A \mid \exists R \]
\[ C ::= B \mid \neg B \]
\[ R ::= P \mid P^- \]
\[ E ::= R \mid \neg R \]

TBox is restricted to \( B \sqsubseteq C \).
DL-Lite\textsubscript{R} extends it with \( R \sqsubseteq E \).
Reasoning is polynomial in size of TBox. Conjunctive query answering in LogSpace in the size of ABox (Polynomial in size of TBox).
**DLP and variants**

OWL 2 RL is roughly based on the following grammar:

\[
B ::= A \mid B \cap B \mid B \cup B \mid \{ a \} \mid \exists R.B \\
C ::= A \mid C \cap C \mid \neg B \mid \forall R.C \mid \exists R.\{ a \} \mid \leq 1.B \mid \leq 1 \\
R ::= P \mid P^{-}
\]

TBox is restricted to \( B \sqsubseteq C \).
Reasoning is polynomial in size of TBox.
OWL 1

OWL Lite corresponds to DL $SHIQ(D)$, OWL DL to $SHOIN(D)$

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \cdots \sqcap C_n$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \cdots \sqcup C_n$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${a_1, \ldots, a_n}$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall R.C$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists R.C$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nR$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nR$</td>
</tr>
</tbody>
</table>

+ XMLS datatypes as well as classes in $\forall R.C$ and $\exists R.C$ + nesting of constructors
Person ⊑ ∀ hasParent. Person needs 5 triples:
: Person a owl:Class;
    rdfs:subClassOf [
        a owl:Restriction;
        owl:onProperty :hasParent;
        owl:allValuesFrom :Person
    ] .

Person ⊑ Man ⊔ Woman needs 7 triples:
: Person owl:equivalentClass [
    a owl:Class;
    owl:unionOf (:Man :Woman)
] .
OWL 2 corresponds to DL $SROIQ(D)$

- max/maxCardinality (qualified and datatypes)
- hasSelf
- ReflexiveProperty, IrreflexiveProperty, AsymmetricProperty
- propertyDisjointWith
- propertyChainAxioms with restrictions
- hasKey (multiple keys, including datatype keys... allows to simulate datatype IFPs)
- top/bottom properties
- more datatype capabilities
- metamodelling & “punning”, more annotations, versioning, etc.
OWL 2 Profiles

OWL 2 DL has 3 sublanguages:

- OWL 2 EL
- OWL 2 QL
- OWL 2 RL
Reasoners and API

DL Reasoners are usually made for (a fragment of) OWL

- **Pellet**: in Java, open source, for SROIQ, i.e., OWL2
- **FaCT++**: in C++, open source, for SROIQ
- **RacerPro**: commercial, for SHIQ, i.e., OWL-Lite
- **KAON2**: in Java, free for non-commercial use, for SHIQ
- **HermiT**: in Java, LGPL, for SHIQ
- **CEL**: in LISP, free for non-commercial use, for EL (light-weight ontologies)

These reasoners implement the OWL Api. Quite easy to use when one knows Jena, Sesame, etc.

See an up to date list at http://www.cs.man.ac.uk/~sattler/reasoners.html
Ontology editors

- **Protégé**: in Java, open source, very popular (v.4 is not an extension of v.3, it is a new branch)
- **SWOOP**: Java, open source, lighter than Protégé
- **NeOn toolkit**: some parts owned by OntoPrise, lots of stuff
Thank you

Questions and comments?