The Distributed Ontology, Model and Specification Language (DOL)

Day 1: Motivation and Introduction

Oliver Kutz\(^1\)
Till Mossakowski\(^2\)

\(^1\)Free University of Bozen-Bolzano, Italy
\(^2\)University of Magdeburg, Germany

Tutorial at ESSLLI 2016, Bozen-Bolzano, August 15 – 19
Welcome to DOL!

Lectures:

- **Day 1**: Motivation and Introduction
- **Day 2**: Basic Structuring with DOL
- **Day 3**: Semantics of DOL
- **Day 4**: Using Multiple Logical Systems
- **Day 5**: Advanced Concepts and Applications
Welcome to DOL!

Daily practical sessions:

- We will learn the basics of how to use DOL in practice employing the Ontohub.org platform and the HETS.eu proof management and reasoning system.
Background:

**DOL is for:**

1. Ontology engineering (e.g. working with OWL or FOL)
2. Model-driven engineering (e.g. working with UML, ORM)
3. Formal (algebraic) specification (e.g. working with FOL, CASL, VDM, Z)

DOL is a **metalanguage** providing formal syntax & semantics for all of them!
Motivation from ontology engineering:

We begin with the question:

- What kind of **ontology** engineering problems does DOL address?

**Note:**

- The issues/problems discussed in the following apply equally to **model-driven engineering** and **formal specification**, and to other uses of logical theories.

**Examples throughout the course** will be taken from the ontology world (understood as logical theories), using propositional, description, and first-order logic, but also from algebra, mereotopology, and software specification.
Where we are in the ontology landscape

- Formal ontology
- Ontology based on linguistic observations
- Ontology based on scientific evidence
- Ontology as information system

- Ontology languages
A basic problem in ontology engineering:

How can we make it easier to build better ontologies?
A basic problem in ontology engineering:

How can we make it easier to build better ontologies?

Claim:
Distributed Ontology, Model and Specification Language (DOL) solves many basic (and advanced) ontology engineering problems
Assume you need to build an ontology
Three challenges for aspiring ontologist

1. Reuse of ontologies
2. Diversity of languages
3. Evaluate against requirements
Three challenges for aspiring ontologist

1. Reuse of ontologies
2. Diversity of languages
3. Evaluate against requirements
Reuse of ontologies I

First idea: Reuse existing resources
Reuse is hard

- Terminology is “wrong”
- Ontology is too wide
- Different ontologies pieces don’t fit to each other
Reuse of ontologies II

Reuse is hard

- Terminology is “wrong”
- Ontology is too wide
- Different ontologies pieces don’t fit to each other

Modifying local copies of ontologies leads to maintenance issues
Three challenges for aspiring ontologist

1. Reuse of ontologies
2. Diversity of languages
3. Evaluate against requirements
Diversity of OMS Languages

Languages that have been used for ontological modelling:

- First-order logic
- Higher-order logic
- OWL (Lite, EL, QL, RL, DL, Full), other DLs
- UML (e.g. class diagrams)
- Entity Relationship Diagrams
- Other languages: SWRL, RIF, ORM, BPMN, …
Which language should I use?
Example 1: DTV: Can you use these tools together?

The OMG Date-Time Vocabulary (DTV) is a heterogenous* ontology:

- SBVR: very expressive, readable for business users
- UML: graphical representation
- OWL DL: formal semantics, decidable
- Common Logic: formal semantics, very expressive

Benefit: DTV utilizes advantages of different languages

* heterogenous = components are written in different languages
Example 2: Relation between OWL and FOL ontologies

Common practice: annotate OWL ontologies with informal FOL:

- Keet’s mereotopological ontology [1],
- Dolce Lite and its relation to full Dolce [2],
- BFO-OWL and its relation to full BFO.

OWL gives better tool support, FOL greater expressiveness.

But: informal FOL axioms are not available for machine processing!


Challenge for combined ontologies I:
Where is the glue?

- The different modules need to be fitted together.
- Challenge: Languages may differ widely with respect to syntactic categories!
Challenge for combined ontologies II: Consistency

- Different people work independently on different parts.
- How do we ensure consistency across the whole ontology?
- Automatic theorem provers are specialized in one language.

\[ \forall x \sim ((\text{Contractor } x) \land (\text{Employee } x)) \]

(bob : Contractor), (bob : Engineer)
Use of different languages

- theoretically good idea
- leads to interoperability problems
- obstacle to reuse of ontologies
Three challenges for aspiring ontologist

1. Reuse of ontologies
2. Diversity of languages
3. Evaluate against requirements
Frequently asked question by students

ARE WE THERE YET?
Let $O$ be an ontology

Capture requirements for $O$ as pairs of *scenarios* and *competency questions*

For each scenario competency question pair $S, Q$:

- Formalize $S$, resulting in theory $\Gamma$
- Formalize $Q$, resulting in formula $\varphi$
- Check with theorem prover whether $O \cup \Gamma \vdash \varphi$

When all proofs are successful, your ontology meets the requirements.
Competency Questions Revisited

- CQ most successful idea for ontology evaluation
- Technically, CQ = proof obligations
- Language for expressing proof obligations?
- Ad hoc handling of CQs
How do we keep track of scenarios and competency questions in a systematic way?
How do we keep track of scenarios and competency questions in a systematic way?

DOL provides a systematic solution to this: ⇒ Lecture 2
What does “Modifying / Reusing” mean?

- Translations between ontology languages
- Renaming of symbols
- Unions of ontologies
- Removing of axioms
- Module extraction
- ...

None of these features are directly supported by widely used languages such as OWL or FOL.
What does “Modifying / Reusing” mean?

- Translations between ontology languages
- Renaming of symbols
- Unions of ontologies
- Removing of axioms
- Module extraction
- ...

None of these features are directly supported by widely used languages such as OWL or FOL.

**DOL covers all these operations:** ⇒ Lecture 2–4
Example Modifying / Reusing

01: CLIF
Woman, Person, Bank

02: OWL
Woman, Human, Bank
Example Modifying / Reusing

01: CLIF
Woman, Person, Bank

02: OWL
Woman, Human, Bank

03: CLIF
Woman, Human, Bank

Logic translation
Example Modifying / Reusing

01: CLIF
Woman, Person, Bank

02: OWL
Woman, Human, Bank

03: CLIF
Woman, Human, Bank

04: CLIF
Woman, Human, Financial_Bank

05: CLIF
Woman, Human, River_Bank

Renaming

Logic translation

Kutz, Mossakowski
Distributed Ontology, Model and Specification Language (DOL)
Example Modifying / Reusing

01: CLIF
Woman, Person, Bank

02: OWL
Woman, Human, Bank

03: CLIF
Woman, Human, Bank

04: CLIF
Woman, Human, Financial_Bank

05: CLIF
Woman, Human, River_Bank

06: CLIF
Woman, Human, Financial_Bank, River_Bank

Union

Renaming

Logic translation
Declaration of Relations: Example Bridge Axiom
Declaration of Relations: Example Bridge Axiom

Ontology: Car

Ontology: Factory
Declaration of Relations: Example Bridge Axiom

Ontology: Car

Every car is build in a car factory

Ontology: Factory
Specification of Intended Relations: Example BFO (Basic Formal Ontology)
Specification of Intended Relations: Example BFO (Basic Formal Ontology)
DOL: change in perspective

- Modular design vs ontology blobs
Ontologies are often big monolithic blobs

National Center for Biotechnology Information (NCBI) Organismal Classification (NCBITAXON)
The NCBI Taxonomy Database is a curated classification and nomenclature for all of the organisms in the public sequence databases.

Uploaded: 6/10/15

The Drug Ontology (DRON)
An ontology of drugs

Uploaded: 5/2/15

Systematized Nomenclature of Medicine - Clinical Terms (SNOMEDCT)
SNOMED Clinical Terms

Uploaded: 6/10/15

Robert Hoehndorf Version of MeSH (RH-MESH)
Medical Subjects Headings Thesaurus 2014, Modified version

Uploaded: 4/22/14

Cell Cycle Ontology (CCO)
An application ontology integrating knowledge about the eukaryotic cell cycle.

Uploaded: 3/7/15
Engineers like it modular
Obvious benefits of modular design

Modularity allows for better

- Maintainability
- Reusability
- Quality control
- Adaptability
Obvious benefits of modular design

Modularity allows for better
- Maintainability
- Reusability
- Quality control
- Adaptability

Why not in ontology engineering?
The OMG standard DOL: Basic Ideas
DOL – An OMG standard

- DOL = Distributed Ontology, Model, and Specification Language
- OMG Specification, Beta 1 released
- Has been approved by OMG
- Now in finalization process
History of DOL

- **First Initiative:** Ontology Integration and Interoperability (OntoIOp)
- started in 2011 as ISO 17347 within ISO/TC 37/SC 3
- now continued as **OMG standard**
  - OMG has more experience with formal semantics
  - OMG documents will be freely available
  - focus extended from ontologies only to formal models and specifications (i.e. logical theories)
  - vote for DOL becoming a standard taken in Spring 2016
  - now finalization task force until end of 2016
- 50 experts participate, ~ 15 have actively contributed
- DOL is open for your ideas, so join us!
**The Big Picture of Interoperability**

<table>
<thead>
<tr>
<th>Modeling</th>
<th>Specification</th>
<th>Ontology engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects/data</td>
<td>Software</td>
<td>Concepts/data</td>
</tr>
<tr>
<td>Models</td>
<td>Specifications</td>
<td>Ontologies</td>
</tr>
<tr>
<td>Modeling Language</td>
<td>Specification language</td>
<td>Ontology language</td>
</tr>
</tbody>
</table>

Diversity and the need for interoperability occur at all these levels!
What have ontologies, models and specifications in common?

OMS …

- are formalised in some logical system
- have a signature with non-logical symbols (domain vocabulary)
- have axioms expressing the domain-specific facts
- semantics: class of structures (models) interpreting signature symbols in some semantic domain
- we are interested in those structures (models) satisfying the axioms
- rich set of annotations and comments

In DOL, ontologies, models and specifications are called “OMS”!
DOL metalanguage capabilities

DOL enables reusability and interoperability.

DOL is a meta-language:

- Literally **reuse** existing OMS
- Operations for **modifying**/reusing OMS
- Declaration of **relations** between OMS
- Declaration of **intended relationships** between OMS
- Support for **heterogenous** OMS
Diversity of Operations on and Relations among OMS

Various operations and relations on OMS are in use:

- **structuring**: import, union, translation, hiding, ... 
- alignment 
  - of many OMS covering one domain 
- module extraction 
  - get **relevant information** out of large OMS 
- approximation 
  - model in an **expressive** language, **reason fast** in a lightweight one 
- distributed OMS 
  - **bridges** between different modellings 
- refinement / interpretation
From Babylonian Confusion to Toolkit
There is a Need for a Unifying Meta Language

Not yet another OMS language, but a meta language covering
- diversity of OMS languages
- translations between these
- diversity of operations on and relations among OMS

Current standards like the OWL API or the alignment API only cover parts of this

The DOL standard addresses this

The DOL language requires abstract semantics covering a diversity of OMSs.
Overview of DOL: Toolkit in Summary

1. **OMS**
   - basic OMS (flattenable)
   - references to named OMS
   - extensions, unions, translations (flattenable)
   - reductions, minimization, maximization (elusive)
   - approximations, module extractions, filterings (flattenable)
   - combinations of networks (flattenable)

   (flattenable = can be flattened to a basic OMS)

2. **OMS mappings** (between OMS)
   - interpretations, refinements, alignments, ...

3. **OMS networks** (based on OMS and mappings)

4. **OMS libraries** (based on OMS, mappings, networks)
   - OMS definitions (giving a name to an OMS)
   - definitions of interpretations, refinements, alignments
   - definitions of networks, entailments, equivalences, ...

Kutz, Mossakowski

Distributed Ontology, Model and Specification Language (DOL)
DOL Semantic Foundations: Institutions

**Signatures**: \( \Sigma \xrightarrow{\sigma} \Sigma' \)

**Sentences**: \( \text{Sen}(\Sigma) \xrightarrow{\text{Sen}(\sigma)} \text{Sen}(\Sigma') \)

**Satisfaction**: \( \models \Sigma \quad \models \Sigma' \)

**Models**: \( \text{Mod}(\Sigma) \leftarrow \text{Mod}(\sigma) \xrightarrow{\text{Mod}(\sigma)} \text{Mod}(\Sigma') \)
DOL Semantic Foundations: Logic Translations

- **EL++** (OWL 2 EL)
- **DL-LiteR** (OWL 2 QL)
- **DL-RL** (OWL 2 RL)
- **SROIQ** (OWL 2 DL)
- **OBO Owl**
- **RDF**
- **Prop**
- **RDFS**
- **OBO 1.4**
- **OWL-Full**
- **OWL-Fox**
- **F-logic**
- **CL^+**
- **CL**
- **CASL**
- **HOL**
- **bRDF**

- **UML-CD**
- **Schema.org**
- **SKOS**
- **DDL Owl**
- **TCoOwl**
- **TCoFox**

The diagram illustrates the relationships and translations between various ontology languages and logic frameworks, highlighting their expressivity levels:

- **grey**: no fixed expressivity
- **green**: decidable ontology languages
- **yellow**: semi-decidable
- **orange**: some second-order constructs
- **red**: full second-order logic

Legend:
- **subinstitute**
- **theoroidal subinstitute**
- **simultaneously exact and model-expansive comorphisms**
- **model-expansive comorphisms**

Kutz, Mossakowski  
Distributed Ontology, Model and Specification Language (DOL)  
2016-08-15 44
Tools & Ressources
Tool support: Heterogeneous Tool Set (Hets)

- available at http://hets.eu
- speaks DOL, propositional logic, OWL, CASL, Common Logic, QBF, modal logic, MOF, QVT, and other languages
- analysis
- computation of colimits (⇒ lecture 5)
- management of proof obligations
- interfaces to theorem provers, model checkers, model finders
Tool support: Ontohub web portal and repository

Ontohub is a web-based repository engine for distributed heterogeneous (multi-language) OMS

- **web-based** prototype available at ontohub.org
- **multi-logic** speaks the same languages as Hets

**multiple repositories** ontologies can be organized in multiple repositories, each with its own management of editing and ownership rights,

**Git interface** version control of ontologies is supported via interfacing the Git version control system,

**linked-data compliant** one and the same URL is used for referencing an ontology, downloading it (for use with tools), and for user-friendly presentation in the browser.
DOL Resources

- http://dol-omg.org Central page for DOL
- http://hets.eu Analysis and Proof Tool Hets, speaking DOL
- http://ontohub.org Ontohub web platform, speaking DOL
- http://ontohub.org/dol-examples DOL examples
- http://ontoiop.org Initial standardization initiative

In particular for this course:

- https://ontohub.org/esslli-2016
  ESSLLI repository of DOL examples
Prop | FOL | OWL
Following the framework of institution theory, we introduce the three logics, propositional, DL, and first-order, by outlining their

1. signatures
2. sentences
3. models
4. satisfaction relation
Propositional Logic in DOL: Signatures

The non-logical symbols are collected in a signature. In propositional logic, these are just propositional letters:

**Definition (Propositional Signatures)**

A propositional signature $\Sigma$ is a set (of propositional letters, or propositional symbols, or propositional variables).
Propositional Logic in DOL: Sentences

A signature provides us with the basic material to form logical expressions, called formulas or sentences.

Definition (Propositional Sentences)

Given a propositional signature $\Sigma$, a propositional sentence over $\Sigma$ is one produced by the following grammar

$$\phi ::= p \mid \bot \mid \top \mid (\neg \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid (\phi \rightarrow \phi) \mid (\phi \leftrightarrow \phi)$$

with $p \in \Sigma$. $\text{Sen}(\Sigma)$ is the set of all $\Sigma$-sentences. We can omit the outermost brackets of a sentence.
Propositional Logic in DOL: Models I

Models (or Truth valuations) provide an interpretation of propositional sentences. Each propositional letter is interpreted as a truth value:

**Definition (Model)**

Given a propositional signature $\Sigma$, a $\Sigma$-model (or $\Sigma$-valuation) is a function $\Sigma \rightarrow \{ T, F \}$. Mod(\Sigma) is the set of all $\Sigma$-models.
Propositional Logic in DOL: Models II

Models interpret not only the propositional letters, but all sentences. A $\Sigma$-model $M$ can be extended using truth tables to

$$M^\# : \text{Sen}(\Sigma) \rightarrow \{T, F\}$$

- $M^\#(p) = M(p)$
- $M^\#(\top) = T$
- $M^\#(\bot) = F$

(a) base cases

<table>
<thead>
<tr>
<th>$M^#(\phi)$</th>
<th>$M^#(\psi)$</th>
<th>$M^#(\phi \land \psi)$</th>
<th>$M^#(\phi \lor \psi)$</th>
<th>$M^#(\phi \rightarrow \psi)$</th>
<th>$M^#(\phi \leftrightarrow \psi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

(b) not

(c) and, or, implication, biimplication
We now can define what it means for a sentence to be satisfied in a model:

**Definition**

\( \phi \) holds in \( M \) (or \( M \) satisfies \( \phi \)), written \( M \models_{\Sigma} \phi \) iff

\[ M^\#(\phi) = T \]
A common formalisation of some natural language constructs is as follows:

<table>
<thead>
<tr>
<th>natural language</th>
<th>formalisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ and $B$</td>
<td>$A \land B$</td>
</tr>
<tr>
<td>$A$ but $B$</td>
<td>$A \land B$</td>
</tr>
<tr>
<td>$A$ or $B$</td>
<td>$A \lor B$</td>
</tr>
<tr>
<td>either $A$ or $B$</td>
<td>$(A \lor B) \land \neg (A \land B)$</td>
</tr>
<tr>
<td>if $A$ then $B$</td>
<td>$A \rightarrow B$</td>
</tr>
<tr>
<td>$A$ only if $B$</td>
<td>$A \rightarrow B$</td>
</tr>
<tr>
<td>$A$ iff $B$</td>
<td>$A \leftrightarrow B$</td>
</tr>
</tbody>
</table>
Common to all logics is the notion of a **theory** commonly introduced as follows. In a given logic with fixed notions of signatures, sentences, models, and satisfaction:

**Definition (Theories)**

A **theory** is a pair $T = (\Sigma, \Gamma)$ where $\Sigma$ is a signature and $\Gamma \subseteq \text{Sen}(\Sigma)$. A **model** of a theory $T = (\Sigma, \Gamma)$ is a $\Sigma$-model $M$ with $M \models \Gamma$. In this case $T$ is called **satisfiable**.

Therefore, a propositional theory is a pair $T = (\Sigma, \Gamma)$ consisting of a set $\Sigma$ of propositional variables and a set $\Gamma$ of propositional formulae expressed in $\Sigma$. 
Prop: Example

A scenario involving John and Maria’s weekend entertainment may be written as follows in DOL (to be continued in Lecture 2):

```dol
logic Propositional
spec JohnMary =
  props sunny, weekend, john_ten, mary_shopping, saturday
  \% declaration of signature
  . sunny \and weekend => john_ten \%(when_ten)%
  . john_ten => mary_shopping \%(when_shopping)%
  . saturday \%(it_is_saturday)%
  . sunny \%(it_is_sunny)%
end
```

Note: \% for comments and \%label\% for axiom labels.
We describe a many-sorted variant of first-order logic:

**Definition**

A **Signature** $\Sigma = (S, F, P)$ of many-sorted-FOL consists of:

- a set $S$ of sorts, where $S^*$ is the set of words over $S$
- for each $w \in S^*$, and each $s \in S$ a set $F_{w,s}$ of function symbols (here $w$ are the argument sorts and $s$ are the result sorts)
- for each $w \in S^*$ a set $P_w$ of predicate symbols
First-order Logic in DOL: Terms

Definition

Given a Signature $\Sigma = (S, F, P)$ the set of ground $\Sigma$-terms is inductively defined by:

- $f_{w,s}(t_1, \ldots, t_n)$ is a term of sort $s$, if each $t_i$ is a term of sort $s_i$ ($i = 1 \ldots n$, $w = S_1 \ldots S_n$) and $f \in F_{w,s}$.

In particular (for $n = 0$) this means that $w = \lambda$ (the empty word), and for $c \in F_{\lambda,s}$, $c_s$ is a constant term of sort $s$.

Note: In this version of FOL, variables are not needed as terms.
First-order Logic in DOL: Sentences I

Definition

Given a signature $\Sigma = (S, F, P)$ the set of $\Sigma$-sentences is inductively defined by:

- $t_1 = t_2$ for $t_1, t_2$ of the same sort
- $p_w(t_1, \ldots, t_n)$ for $t_i \Sigma$-term of sort $s_i$,
  $(1 \leq i \leq n, w = s_1, \ldots, s_n, p \in P_w)$
- $\phi_1 \land \phi_2$ for $\phi_1, \phi_2 \Sigma$-formulae
- $\phi_1 \lor \phi_2$ for $\phi_1, \phi_2 \Sigma$-formulae
- $\phi_1 \rightarrow \phi_2$ for $\phi_1, \phi_2 \Sigma$-formulae
- $\phi_1 \leftrightarrow \phi_2$ for $\phi_1, \phi_2 \Sigma$-formulae
- $\neg \phi_1$ for $\phi_1 \Sigma$-formula
- $\top, \bot$
First-order Logic in DOL: Sentences II

Definition (continued)

Given a signature $\Sigma = (S, F, P)$ the set of $\Sigma$-sentences is inductively defined by:

- ...  
- $\forall x : s . \phi$ if $s \in S$, $\phi$ is a $\Sigma \cup \{x : s\}$-sentence where $\Sigma \cup \{x : s\}$ is $\Sigma$ enriched with a new constant $x$ of sort $s$
- $\exists x : s . \phi$ likewise

Note: We have no ‘open formulae’ in this version of FOL.
First-order Logic in DOL: Models

Definition

Given a signature $\Sigma = (S, F, P)$ a $\Sigma$-model $M$ consists of

- a carrier set $M_s \neq \emptyset$ for each sort $s \in S$
- a function $f^m_{w,s} : M_{s_1} \times \ldots \times M_{s_n} \rightarrow M_s$ for each $f \in F_{w,s}$, $w = s_1, \ldots, s_n$.
  
  In particular, for a constant, this is just an element of $M_s$
- a relation $p^M_w \subseteq M_{s_1} \times \ldots \times M_{s_n}$ for each $p \in P_w$, $w = s_1 \ldots s_n$
Definition

A $\Sigma$-term $t$ is evaluated in a $\Sigma$-model $M$ as follows:

$$M(f_{w,s}(t_1, \ldots t_n)) = f_{w,s}^M(M(t_1), \ldots M(t_n))$$
First-order Logic in DOL: Satisfaction

**Definition**

Let $\Sigma' = \Sigma \cup \{x : s\}$. A $\Sigma'$-model $M'$ is called a $\Sigma'$-expansion of a $\Sigma$-model $M$ if $M'$ and $M$ interpret every symbol except $x$ in the same way.

**Definition (Satisfaction of sentences)**

$M \models t_1 = t_2$ iff $M(t_1) = M(t_2)$

$M \models p_w(t_1 \ldots t_n)$ iff $(M(t_1), \ldots, M(t_n)) \in p^M_w$

$M \models \phi_1 \land \phi_2$ iff $M \models \phi_1$ and $M \models \phi_2$ etc.

$M \models \forall x : s. \phi$ iff for all $\Sigma'$-expansions $M'$ of $M$, $M' \models \phi$

where $\Sigma' = \Sigma \cup \{x : s\}$

$M \models \exists x : s. \phi$ iff there is a $\Sigma'$-expansion $M'$ of $M$ such that $M' \models \phi$
A specification of a total order in many-sorted first-order logic, using CASL syntax:

```casl
logic CASL.FOL=

spec TotalOrder =
    sort Elem
    pred __leq__ : Elem * Elem
    . forall x : Elem . x leq x %(refl)%
    . forall x,y : Elem . x leq y /\ y leq x => x = y  %(antisym)%
    . forall x,y,z : Elem . x leq y /\ y leq z => x leq z %(trans)%
    . forall x,y : Elem . x leq y \/ y leq x  %(dichotomy)%

end
```

Full specification at https://ontohub.org/esslli-2016/FOL/OrderTheory.dol
DOL supports the logic $\mathcal{SROIQ}$ underlying OWL 2 DL

We focus here on the basic DL $\mathcal{ALC}$
Definition

A DL-signature $\Sigma = (C, R, I)$ consists of

- a set $C$ of concept names,
- a set $R$ of role names,
- a set $I$ of individual names,
Description Logic in DOL: Concepts

Definition

For a signature $\Sigma = (C, R, I)$ the set of $ALC$-concepts over $\Sigma$ is defined by the following grammar:

$$C, D ::= \begin{align*}
A & \text{ for } A \in C \\
\top & \\
\bot & \\
\lnot C & \\
C \land D & \\
C \lor D & \\
\exists R.C & \text{ for } R \in R \\
\forall R.C & \text{ for } R \in R
\end{align*}$$

Manchester syntax

- concept name
- Thing
- Nothing
- not C
- C and D
- C or D
- R some C
- R only C

$^aALC$ stands for “attributive language with complement”
Description Logic in DOL: Sentences

**Definition**

The set of $\mathcal{ALC}$-Sentences over $\Sigma$ ($\text{Sen}(\Sigma)$) is defined as:

- $C \sqsubseteq D$, where $C$ and $D$ are $\mathcal{ALC}$-concepts over $\Sigma$.
  
  **Class:** $C$  **SubclassOf:** $D$

- $a : C$, where $a \in I$ and $C$ is a $\mathcal{ALC}$-concept over $\Sigma$.
  
  **Individual:** $a$  **Types:** $C$

- $R(a_1, a_2)$, where $R \in R$ and $a_1, a_2 \in I$.
  
  **Individual:** $a_1$  **Facts:** $R$ $a_2$
Description Logic in DOL: Models I

**Definition**

Given $\Sigma = (C, R, I)$, a $\Sigma$-model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

- $\Delta^{\mathcal{I}}$ is a non-empty set
- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each $A \in C$
- $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each $R \in R$
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each $a \in I$
Description Logic in DOL: Models II

Definition

We can extend $\cdot^I$ to all concepts as follows:

- $\top^I = \Delta^I$
- $\bot^I = \emptyset$
- $(\neg C)^I = \Delta^I \setminus C^I$
- $(C \cap D)^I = C^I \cap D^I$
- $(C \cup D)^I = C^I \cup D^I$
- $(\exists R.C)^I = \{x \in \Delta^I | \exists y \in \Delta^I .(x, y) \in R^I, y \in C^I\}$
- $(\forall R.C)^I = \{x \in \Delta^I | \forall y \in \Delta^I .(x, y) \in R^I \implies y \in C^I\}$
Description Logic in DOL: Satisfaction

Definition (Satisfaction of sentences in a model)

\[ \mathcal{I} \models C \subseteq D \quad \text{iff} \quad C^\mathcal{I} \subseteq D^\mathcal{I}. \]

\[ \mathcal{I} \models a : C \quad \text{iff} \quad a^\mathcal{I} \in C^\mathcal{I}. \]

\[ \mathcal{I} \models R(a_1, a_2) \quad \text{iff} \quad (a_1^\mathcal{I}, a_2^\mathcal{I}) \in R^\mathcal{I}. \]
**OWL: Example**

**logic** OWL

**ontology** FamilyBase =

Class: Person
Class: Female
Class: Woman EquivalentTo: Person and Female
Class: Man EquivalentTo: Person and not Woman

ObjectProperty: hasParent
ObjectProperty: hasChild InverseOf: hasParent
ObjectProperty: hasHusband

Class: Mother EquivalentTo: Woman and hasChild some Person
Class: Parent EquivalentTo: Father or Mother
Class: Wife EquivalentTo: Woman and hasHusband some Man

...
OWL: Example (continued)

...  
Class: Married  
Class: MarriedMother EquivalentTo: Mother and Married  
SubClassOf: Female and Person  
Individual: john Types: Father  
Individual: mary Types: Mother  
Facts: hasChild john  
end

Full specification at  
https://ontohub.org/esslli-2016/OWL/Family.dol
DOL is not a ‘Lingua Franca’
DOL is not a ‘Lingua Franca’
DOL is a metalanguage reusing, modifying, connecting ontologies, models, and specifications (called OMS)
Summary

- DOL is not a ‘Lingua Franca’
- DOL is a metalanguage reusing, modifying, connecting ontologies, models, and specifications (called OMS)
- DOL enables a modular/structured approach to knowledge engineering
Detailed Course Overview

Day 1: We just learned what DOL is about and what kind of problems it can help solve.

Remainder of today: Get started with Ontohub.org and HETS.
Detailed Course Overview

- **Day 1:** We just learned what DOL is about and what kind of problems it can help solve.  
  **Remainder of today:** Get started with Ontohub.org and HETS.

- **Day 2:** Basic structuring concepts for logical theories and examples in DOL
Detailed Course Overview

- **Day 1:** We just learned what DOL is about and what kind of problems it can help solve.
  
  **Remainder of today:** Get started with Ontohub.org and HETS.

- **Day 2:** Basic structuring concepts for logical theories and examples in DOL

- **Day 3:** General institution-theoretic semantics for DOL: logic-independence
Detailed Course Overview

- **Day 1**: We just learned what DOL is about and what kind of problems it can help solve.
  
  **Remainder of today**: Get started with Ontohub.org and HETS.

- **Day 2**: Basic structuring concepts for logical theories and examples in DOL

- **Day 3**: General institution-theoretic semantics for DOL: logic-independence

- **Day 4**: Working with multiple logics: heterogeneity
Day 1: We just learned what DOL is about and what kind of problems it can help solve.

Remainder of today: Get started with Ontohub.org and HETS.

Day 2: Basic structuring concepts for logical theories and examples in DOL

Day 3: General institution-theoretic semantics for DOL: logic-independence

Day 4: Working with multiple logics: heterogeneity

Day 5: Advanced applications: alignments, networks, blending
Exercise for tomorrow

- clone the ESSLLI repository on ontohub.org:
  git clone git://ontohub.org/esslli-2016.git
Exercise for tomorrow

- clone the ESSLLI repository on ontohub.org:
  git clone git://ontohub.org/esslli-2016.git
- Look at the following theories expressed in Prop / OWL
  1) Propositional/Penguin.dol
  2) OWL/Family.dol
Exercise for tomorrow

- clone the ESSLLI repository on ontohub.org:
  
git clone git://ontohub.org/esslli-2016.git

- Look at the following theories expressed in Prop / OWL
  (1) Propositional/Penguin.dol
  (2) OWL/Family.dol

- Determine whether they are satisfiable or not
Exercise for tomorrow

- clone the ESSLLI repository on ontohub.org:
  git clone git://ontohub.org/esslli-2016.git
- Look at the following theories expressed in Prop / OWL
  (1) Propositional/Penguin.dol
  (2) OWL/Family.dol
- Determine whether they are satisfiable or not
- If they are, make them unsatisfiable, if they are not, make them satisfiable
Exercise for tomorrow

- clone the ESSLLI repository on ontohub.org:
  
  ```
  git clone git://ontohub.org/esslli-2016.git
  ```

- Look at the following theories expressed in Prop / OWL
  
  1. Propositional/Penguin.dol
  2. OWL/Family.dol

- Determine whether they are satisfiable or not

- If they are, make them unsatisfiable, if they are not, make them satisfiable

- Upload your results in your private OntoHub.org repository