The Distributed Ontology, Model and Specification Language (DOL) —Motivation and Introduction—

Oliver Kutz[1]
Till Mossakowski and Fabian Neuhaus[2]

[1]Free University of Bozen-Bolzano, Italy
[2]University of Magdeburg, Germany

Tutorial at LAC 2018, Melbourne, February 12–16

based on a course given with Till Mossakowski at ESSLLI 2016

Welcome to DOL!

Lectures:

- Day 1: Motivation and Introduction
- Day 2: Basic Structuring with DOL
- Day 3: Case Study 1: Mereotopology
- Day 3: Combinations and using Multiple Logical Systems
- Day 4: Case Study 2: Conceptual Blending

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 engeneering:

We begin with the question:

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

Note:

- The issues/problems disscussed 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 ontologists

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

Three challenges for aspiring ontologists

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

Reuse of ontologies I

First idea:
Reuse existing resources

Reuse of ontologies II

Reuse is hard

- Terminology is "wrong"
- Ontology is too wide
- Different pieces of ontologies don't fit with each other

Reuse of ontologies II

Reuse is hard

- Terminology is "wrong"
- Ontology is too wide
- Different pieces of ontologies don't fit with 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!

[1] C.M. Keet, F.C. Fernández-Reyes, and A. Morales-González. Representing mereotopological relations in OWL ontologies with ontoparts. In *Proc. of the ESWC'12*, vol. 7295 *LNCS*, 2012.

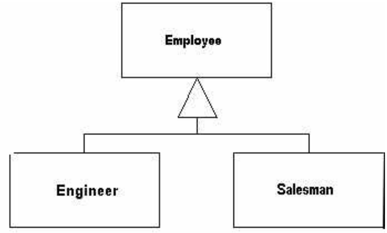
[2] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. Descriptve ontology for linguistic and cognitive engineering. http://www.loa.istc.cnr.it/DOLCE.html.

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.



$\forall x \sim ((\text{Contractor}\, x) \wedge (\text{Employee}\, x))$
(bob : Contractor), (bob : Engineer)

Diversity of Language: Conclusion

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

Competency Questions – Simplified Summary

- 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 Γ
  - Formalize Q, resulting in formula φ
  - 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

Competency Questions Challenge

- How do we keep track of scenarios and competency questions in a systematic way?

Competency Questions Challenge

- How do we keep track of scenarios and competency questions in a systematic way?

DOL provides a systematic solution to this: $\Rightarrow$ 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?

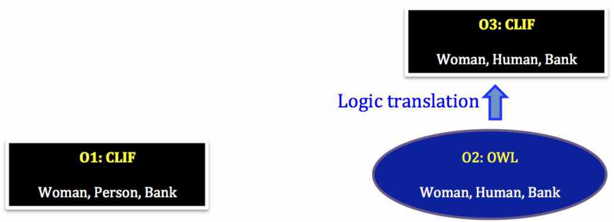
- Translations between ontology languages
- Renaming of symbols
- Unions of ontologies
- Removing of axioms
- Module extraction
- ...

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

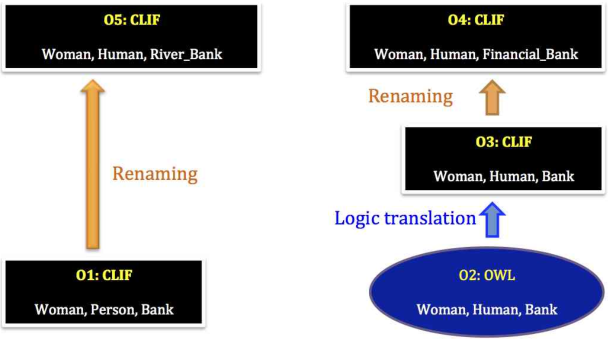
DOL covers all these operations: $\Rightarrow$ Lecture 2–4

Example Modifying / Reusing

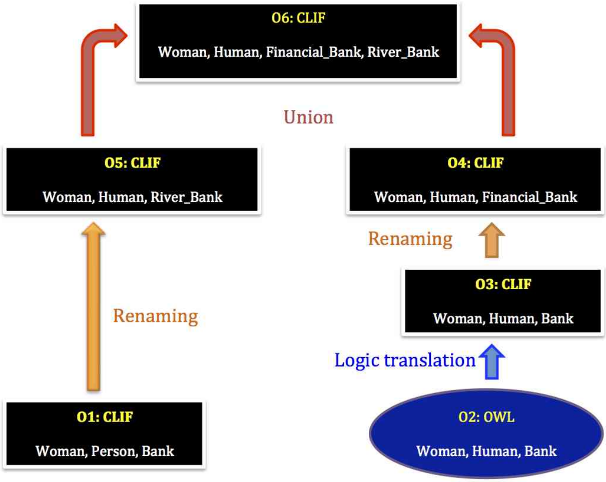
Example Modifying / Reusing



Example Modifying / Reusing



Example Modifying / Reusing



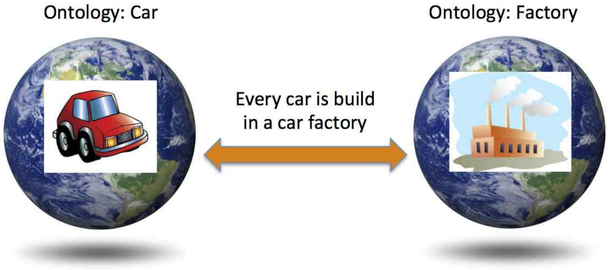
Declaration of Relations: Example Bridge Axiom

Declaration of Relations: Example Bridge Axiom

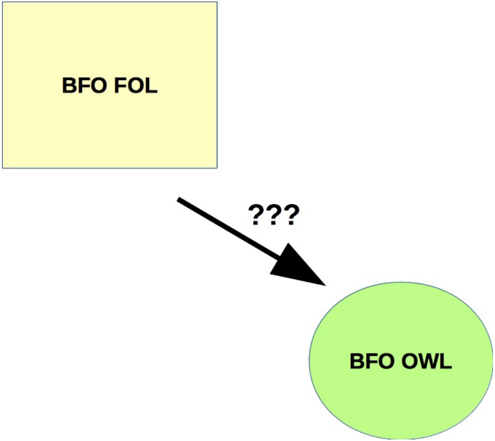
Ontology: Car

Ontology: Factory

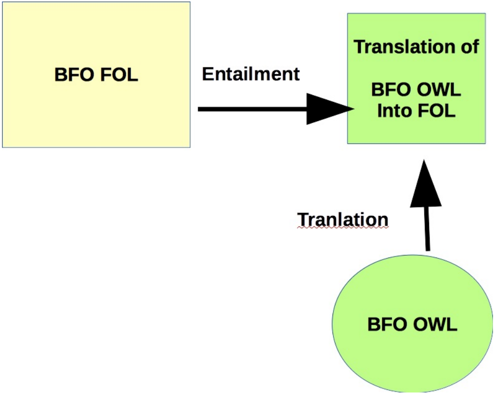
Declaration of Relations: Example Bridge Axiom



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)

projects 12 | classes 906,907

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)

classes 408,573

An ontology of drugs

Uploaded: 5/2/15

Systematized Nomenclature of Medicine - Clinical Terms (SNOMEDCT)

notes 2 | projects 18 | classes 316,031

SNOMED Clinical Terms

Uploaded: 6/10/15

Robert Hoehndorf Version of MeSH (RH-MESH)

projects 3 | classes 305,349

Medical Subjects Headings Thesaurus 2014, Modified version

Uploaded: 4/22/14

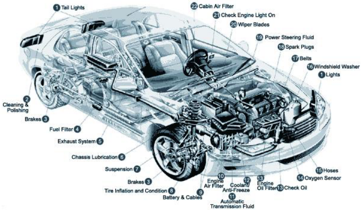
Cell Cycle Ontology (CCO)

projects 2 | classes 277,764

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
- Approved by OMG
- Finalised in late 2017

OBJECT MANAGEMENT GROUP®

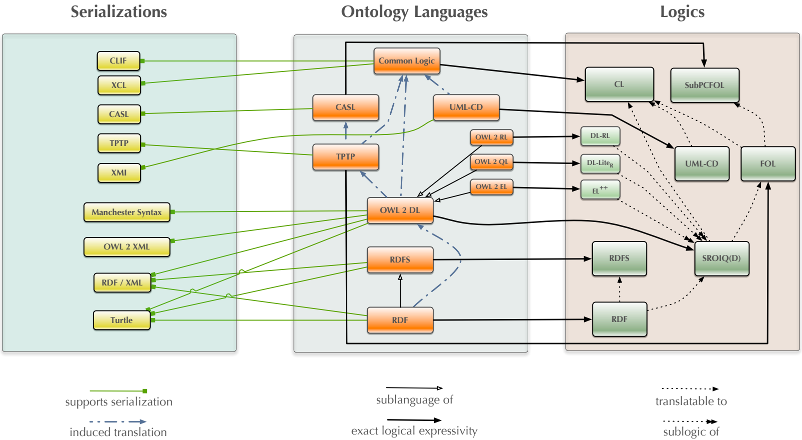
History of DOL

- 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
  - finalisation task force until end of 2017
- 50 experts participate, $\sim$ 15 have actively contributed
- DOL is open for your ideas, so join us!

The Big Picture of Interoperability

Modeling	Specification	Ontology engineering
Objects/data	Software	Concepts/data
Models	Specifications	Ontologies
Modeling Language	Spec. language	Ontology language

Diversity and the need for interoperability occur at all these levels!


Serializations
Ontology Languages
Logics
CLIF
XCL
CASL
TPTP
XMI
Manchester Syntax
OWL 2 XML
RDF / XML
Turtle
Common Logic
CASL
UML-CD
TPTP
OWL 2 RL
OWL 2 QL
OWL 2 EL
OWL 2 DL
RDFS
RDF
CL
SubPCFOL
DL-RL
DL-Lite$_R$
EL^{++}
UML-CD
FOL
RDFS
SROIQ(D)
RDF
supports serialization
induced translation
sublanguage of
exact logical expressivity
translatable to
sublogic of

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:

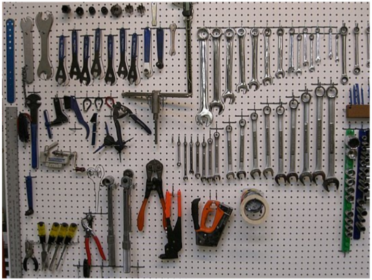
- Literally **reuse** existing OMS
- Operations for **modifying**/reusing OMS
- Declaration of **relations** between OMS
- Declaration of **intended relationships** between OMS
- Support for **heterogenous** OMS

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 aligment 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

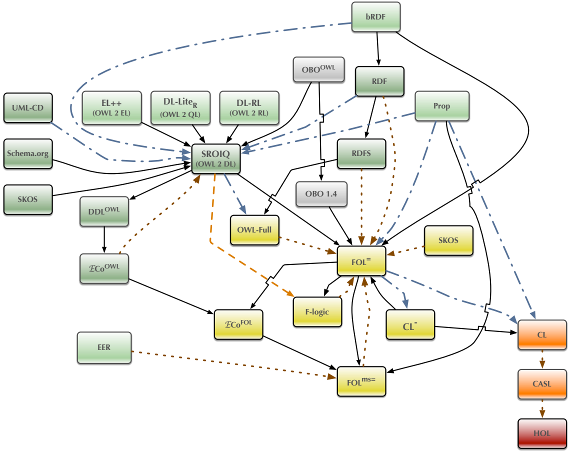
1. OMS
   - basic OMS (flattenable)
   - references to named OMS
   - extensions, unions, translations (flattenable)
   - reductions, minimization, maximization (elusive)
   - approximations, module extractions, filterings (flattenable)
   - combinations of networks (flattenable)

   (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, ...

DOL Semantic Foundations: Institutions

$$
\begin{array}{lccc}
\textit{Signatures}: & \Sigma & \xrightarrow{\sigma} & \Sigma' \\
\textit{Sentences}: & Sen(\Sigma) & \xrightarrow{Sen(\sigma)} & Sen(\Sigma') \\
\textit{Satisfaction}: & \models_{\Sigma} & & \models_{\Sigma'} \\
\textit{Models}: & Mod(\Sigma) & \xleftarrow{Mod(\sigma)} & Mod(\Sigma')
\end{array}
$$

DOL Semantic Foundations: Logic Translations



subinstitute

theoroidal subinstitute

simultaneously exact and model-expansive comorphisms

model-expansive comorphisms

grey: no fixed expressivity

green: decidable ontology languages

yellow: semi-decidable

orange: some second-order constructs

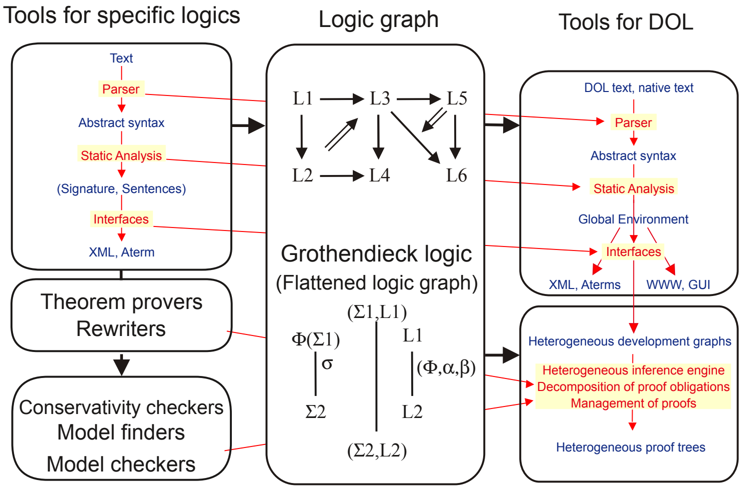
red: full second-order logic

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 ($\Rightarrow$ lecture 4)
- management of proof obligations
- interfaces to theorem provers, model checkers, model finders

Architecture of the heterogeneous tool set Hets



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, full ESSLLI 2016 course:

- `https://ontohub.org/esslli-2016`
  ESSLLI repository of DOL examples

Prop | FOL | OWL

Three Logics as Institutions

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

Reminder: Institutions

$$
\begin{array}{lccc}
\mathit{Signatures}: & \Sigma & \xrightarrow{\quad\sigma\quad} & \Sigma' \\[2em]
\mathit{Sentences}: & \mathit{Sen}(\Sigma) & \xrightarrow{\mathit{Sen}(\sigma)} & \mathit{Sen}(\Sigma') \\[2em]
\mathit{Satisfaction}: & \models_{\Sigma} & & \models_{\Sigma'} \\[2em]
\mathit{Models}: & \mathit{Mod}(\Sigma) & \xleftarrow{\mathit{Mod}(\sigma)} & \mathit{Mod}(\Sigma')
\end{array}
$$

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 Σ 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 Σ, a propositional sentence over Σ is one produced by the following grammar

$$\phi ::= p \mid \bot \mid \top \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\phi \leftrightarrow \phi)$$

with $p \in \Sigma$. $\mathrm{Sen}(\Sigma)$ is the set of all Σ-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 Σ, a Σ-model (or Σ-valuation) is a function $\Sigma \rightarrow \{T, F\}$. $\mathrm{Mod}(\Sigma)$ is the set of all Σ-models.

Propositional Logic in DOL: Models II

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

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

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

(a) base cases

$M^{\#}(\phi)$	$M^{\#}(\neg\phi)$
T	F
F	T

(b) not

$M^{\#}(\phi)$	$M^{\#}(\psi)$	$M^{\#}(\phi \wedge \psi)$	$M^{\#}(\phi \vee \psi)$	$M^{\#}(\phi \rightarrow \psi)$	$M^{\#}(\phi \leftrightarrow$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

(c) and, or, implication, biimplication

Figure: Truth tables

Propositional Logic in DOL: Satisfaction

We now can define what it means for a sentence to be satisfied in a model:

Definition

ϕ holds in M (or M satisfies ϕ), written $M \models_\Sigma \phi$ iff

$$M^{\#}(\phi) = T$$

Prop: Example

A common formalisation of some natural language constructs is as follows:

natural language	formalisation
A and B	$A \wedge B$
A but B	$A \wedge B$
A or B	$A \vee B$
either A or B	$(A \vee B) \wedge \neg(A \wedge B)$
if A then B	$A \rightarrow B$
A only if B	$A \rightarrow B$
A iff B	$A \leftrightarrow B$

Theories

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 Σ is a signature and $\Gamma \subseteq \mathrm{Sen}(\Sigma)$. A model of a theory $T = (\Sigma, \Gamma)$ is a Σ-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 Σ of propositional variables and a set Γ of propositional formulae expressed in Σ.

Prop: Example

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

```
logic Propositional
spec JohnMary =
  props sunny, weekend, john_tennis,
        mary_shopping, saturday
                               %% declaration of signature
. sunny /\ weekend => john_tennis %(when_tennis)%
. john_tennis => mary_shopping    %(when_shopping)%
. saturday                        %(it_is_saturday)%
. sunny                           %(it_is_sunny)%
end
```

Note: `%%` for comments and `%label%` for axiom labels.

First-order Logic in DOL: Signatures

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 Σ-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 Σ-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 Σ-term of sort s_i,
  $(1 \leq i \leq n, w = s_1, \ldots, s_n, p \in P_w)$
- $\phi_1 \wedge \phi_2$ for ϕ_1, ϕ_2 Σ-formulae
- $\phi_1 \vee \phi_2$ for ϕ_1, ϕ_2 Σ-formulae
- $\phi_1 \rightarrow \phi_2$ for ϕ_1, ϕ_2 Σ-formulae
- $\phi_1 \leftrightarrow \phi_2$ for ϕ_1, ϕ_2 Σ-formulae
- $\neg\phi_1$ for ϕ_1 Σ-formula
- $\top$, $\bot$

First-order Logic in DOL: Sentences II

Definition (continued)

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

- ...
- $\forall x : s \,.\, \phi$ if $s \in S$, ϕ is a $\Sigma \uplus \{x : s\}$-sentence where $\Sigma \uplus \{x : s\}$ is Σ 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 Σ-model M consists of

- a carrier set $M_s \neq \emptyset$ for each sort $s \in S$
- a function $f^m_{w,s} : M_{s_1} \times \ldots \times M_{s_n} \to M_s$ 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 $p \in P_w, w = s_1 \ldots s_n$

First-order Logic in DOL: Evaluating Terms

Definition

A Σ-term t is evaluated in a Σ-model M as follows:

$$M(f_{w,s}(t_1, \ldots t_n)) = f^M_{w,s}(M(t_1), \ldots M(t_n))$$

First-order Logic in DOL: Satisfaction

Definition

Let $\Sigma' = \Sigma \uplus \{x : s\}$. A Σ'-model M' is called a Σ'-expansion of a Σ-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 $M(t_1) = M(t_2)$

$M \models p_w(t_1 \dots t_n)$ iff $(M(t_1), \dots M(t_n)) \in p_w^M$

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

$M \models \forall x : s.\phi$ iff for all Σ'-expansions M' of M, $M' \models \phi$
where $\Sigma' = \Sigma \uplus \{x : s\}$

$M \models \exists x : s.\phi$ iff there is a Σ'-expansion M' of M such that $M' \models \phi$

FOL: Example

A specification of a total order in many-sorted first-order logic, using CASL syntax:

```
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   %(antisy
  . forall x,y,z : Elem . x leq y /\ y leq z => x leq z %(tran
  . forall x,y : Elem . x leq y \/ y leq x          %(dichotom
end
```

Full specification at
https://ontohub.org/esslli-2016/FOL/OrderTheory.dol

OWL: Description Logic in DOL

- DOL supports the logic $\mathcal{SROIQ}$ underlying OWL 2 DL
- We focus here on the basic DL $\mathcal{ALC}$

Description Logic in DOL: Signatures

Definition

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

- a set $\mathbf{C}$ of concept names,
- a set $\mathbf{R}$ of role names,
- a set $\mathbf{I}$ of individual names,

Description Logic in DOL: Concepts

Definition

For a signature $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ the set of $\mathcal{ALC}$-concepts[1] over Σ is defined by the following grammar:

		Manchester syntax
$C, D ::=$	A for $A \in \mathbf{C}$	**concept name**
	$\mid \top$	Thing
	$\mid \bot$	Nothing
	$\mid \neg C$	not C
	$\mid C \sqcap D$	C and D
	$\mid C \sqcup D$	C or D
	$\mid \exists R.C$ for $R \in \mathbf{R}$	R some C
	$\mid \forall R.C$ for $R \in \mathbf{R}$	R only C

[1] $\mathcal{ALC}$ stands for "attributive language with complement"

Description Logic in DOL: Sentences

Definition

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

- $C \sqsubseteq D$, where C and D are $\mathcal{ALC}$-concepts over Σ.

  Class: C SubclassOf: D

- $a : C$, where $a \in \mathbf{I}$ and C is a $\mathcal{ALC}$-concept over Σ.

  Individual : a Types: C

- $R(a_1, a_2)$, where $R \in \mathbf{R}$ and $a_1, a_2 \in \mathbf{I}$.

  Individual : a1 Facts: R a2

Description Logic in DOL: Models I

Definition

Given $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$, a Σ-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 $A \in \mathbf{C}$
- $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each $R \in \mathbf{R}$
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each $a \in \mathbf{I}$

Description Logic in DOL: Models II

Definition

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

$$\begin{array}{lcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} | \exists y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} | \forall y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}
\end{array}$$

Description Logic in DOL: Satisfaction

Definition (Satisfaction of sentences in a model)

$$
\begin{array}{lll}
\mathcal{I} \models C \sqsubseteq D & \text{iff} & C^{\mathcal{I}} \subseteq D^{\mathcal{I}}. \\
\mathcal{I} \models a : C & \text{iff} & a^{\mathcal{I}} \in C^{\mathcal{I}}. \\
\mathcal{I} \models R(a_1, a_2) & \text{iff} & (a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in R^{\mathcal{I}}.
\end{array}
$$

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`

Summary

- DOL is not a ‘Lingua Franca’

Summary

- 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

Exercise

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

Exercise

- 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

- 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

- 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

- 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