The Distributed Ontology, Modeling and Specification Language (DOL)

Till Mossakowski[1,2] Oliver Kutz[1] Mihai Codescu[3]

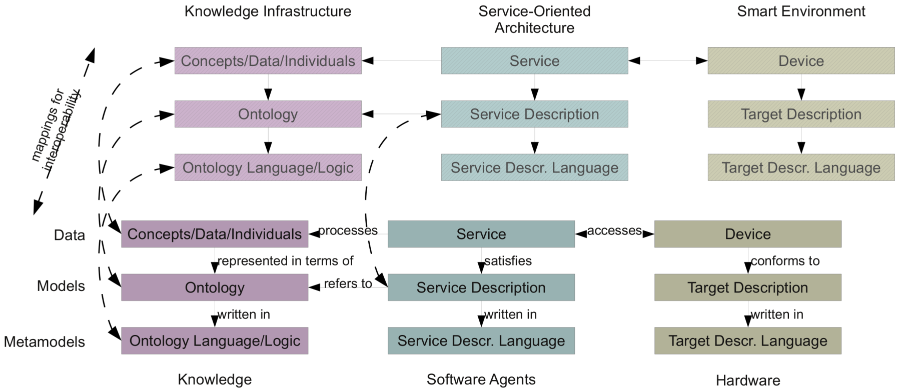
[1]Collaborative Research Centre on Spatial Cognition, University of Bremen

[2]DFKI GmbH Bremen

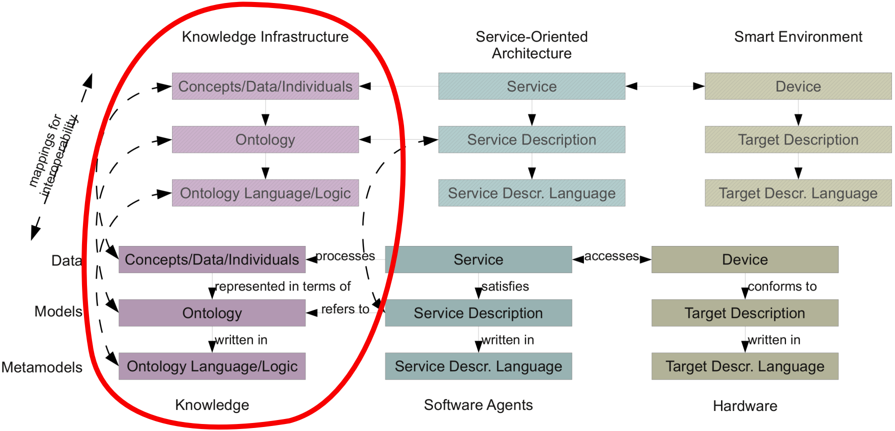
[3]University of Erlangen-Nürnberg

WoMo 2013, La Coruna

The Big Picture of Interoperability



The Big Picture of Interoperability



Motivation: Diversity of Ontology Languages

A great diversity of ontology languages is in use:

- OWL, RDF, OBO
- UML class diagrams
- RIF (Rule Interchange Format)
- EER (Enhanced Entity-Relationship Diagrams), Datalog, ORM (object role modeling)
- the meta model of schema.org
- SKOS (Simple Knowledge Organization System)
- FOL, F-logic, Common Logic

It is common practice to informally annotate OWL ontologies with FOL axioms (e.g. Keet's mereo-topological ontology, Dolce Lite, BFO-OWL)

Use Case: OMG's Date-Time Vocabulary

- date-time vocabulary is formulated in different languages: SBVR, Common Logic, IKL, UML+OCL, OWL
- different languages address different audiences
  - SBVR: business users
  - UML+OCL: software implementors
  - OWL: ontology developers and users
  - Common Logic, IKL: (foundational) ontology developers and users
- How can we
  - formally relate the different logical specifications?
  - specify the OWL version to be an approximation of the Common Logic version?
  - extract submodules covering specific aspects?

Use Case (cont'd): OWL/FOL ontologies

heterogeneous OWL/FOL ontologies

- e.g. an OWL ontology with some FOL axioms
- . . . for use with an OWL reasoner, a FOL theorem prover, and a FOL model finder
    1. OWL reasoner for reasoning about the OWL part
    2. FOL theorem prover for proving consequences of the whole ontology
    3. FOL model finder for finding a model of the whole ontology
    4. FOL model finder for disproving consequences of the whole ontology

More Use Cases

- The use of RDFS or OWL to specify a taxonomy of sorts for a more expressive logic with many-sorted semantics
- The use of Common Logic to express metadata concerning modelling assumptions for simulation (e.g. climate change) datasets (Datalog expressivity).
    - The Datalog theory describes the closed world of the observed dataset
    - The Common Logic theory is open-world and describes the physical laws of the object of observation

More Use Cases (Modeling and Specification)

- UML model involving different diagram types
  - check for semantic consistency
    (relative to a given formal semantics)
- temporal logic specification
  - check against some process model
  - then refine this into some finite automaton
- UML protocol state machine
  (possibly enriched with some UML sequence diagrams and OCL constraints)
  - refine to UML behaviour state machine

Motivation: Diversity of Operations on and Relations among Ontologies

Various operations and relations on ontologies are in use:

- matching and alignment
  - of many ontologies covering one domain
- module extraction
  - get relevant information out of large ontology
- approximation
  - model in an expressive language, reason fast in a lightweight one
- querying
- ontology-based database access/data management
- distributed description logics, $\mathcal{E}$-connections
  - bridges between different modellings

Need for a Unifying Meta Language

Not yet another ontology language, but a meta language covering

- diversity of ontology languages
- translations between these
- diversity of operations on and relations among ontologies

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

The
Ontology, Modeling and Specification
Integration and Interoperability (OntoIOp)
initiative addresses this

The OntoIOp initiative

- 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)
  - request for proposals (RFP) to be issued in fall 2013
  - proposals answering RFP due in December 2014
- 50 experts participate, $\sim$ 15 have contributed
- OntoIOp is open for your ideas, so join us!

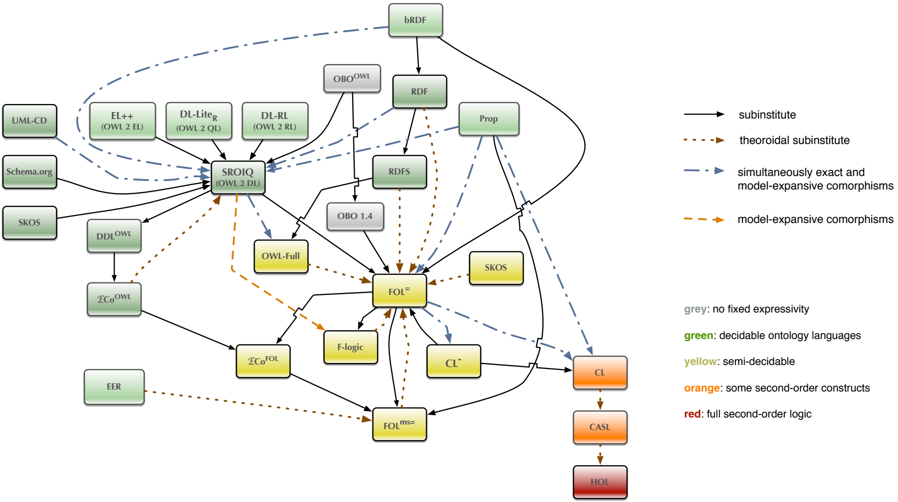
Requirements in the OMG RFP OntoIOp

- provide a meta-language for:
  - logically heterogeneous ontologies
  - modular ontologies
  - module extraction, approximation
  - links (interpretations, alignments) between ontologies/modules
  - combination of ontologies along links
- provide an abstract syntax as MOF or SMOF model
- provide a concrete syntax
- provide a formal semantics
  - criteria for logics to conform with OntoIOp
  - translations between these logics
- be logic-agnostic, e.g. ontologies consist of symbols and axioms

The Distributed Ontology, Modeling and Specification Language (DOL)

- has been prepared within ISO/TC 37/SC 3
- now continued as a proposal for the OMG RFP OntoIOp
  - DOL = one specific answer to the RFP requirements
  - there may be other answers to the RFP (but unlikely)
- DOL is based on some graph of logics and translations
- DOL has a model-theoretic semantics
  - semantics of an ontology:
    - logic
    - signature in that logic
    - class of models over that signature
- analysis and proof tools for DOL exist

An Initial Logic Graph for DOL



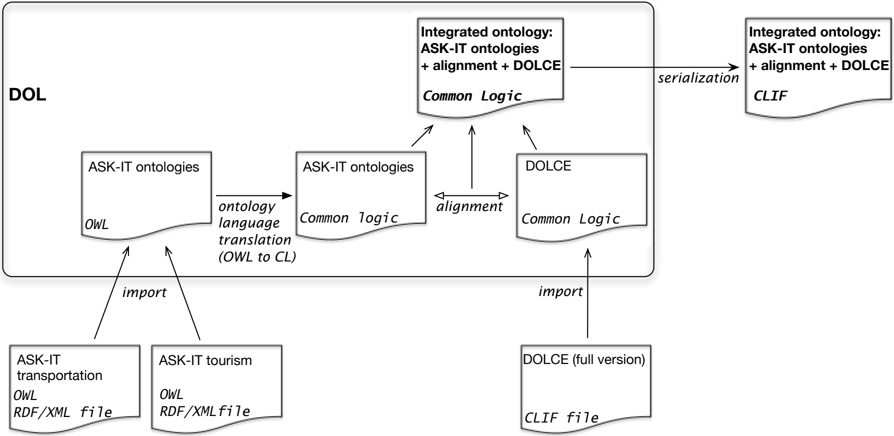
What is an Ontology Language / Logic?

We need the following ingredients:

- sentences, models, satisfaction
- signatures
- signature extensions, signature morphisms
- model reducts
- signature colimits

Details later . . .

Sample Use of DOL



Overview of DOL

1. modular and heterogeneous ontologies
    - basic ontologies
    - extensions, unions, translations, reductions
    - approximations, module extractions
    - minimization
    - combination
    - ontology bridges
2. distributed ontologies (based on 1)
    - ontology definitions (giving a name to an ontology)
    - interpretations (of theories)
    - equivalences
    - module relations
    - alignments

Basic Ontologies

- written in some ontology language from the logic graph
- semantics is inherited from the ontology language
- e.g. in OWL:

```
Class: Woman  EquivalentTo: Person and Female
ObjectProperty: hasParent
```

- e.g. in Common Logic:

```
(cl-module PreOrder
  (forall (x) (le x x))
  (forall (x y z)
          (if (and (le x y)
                   (le y z))
              (le x z))))
```

Extensions

- O_1 **then** O_2: extension of O_1 by new symbols and axioms O_2
- O_1 **then %mcons** O_2: model-conservative extension
  - each O_1-model has an expansion to O_1 **then** O_2
- O_1 **then %ccons** O_2: consequence-conservative extension
  - O_1 **then** $O_2 \models \varphi$ implies $O_1 \models \varphi$, for φ in the language of O_1
- O_1 **then %def** O_2: definitional extension
  - each O_1-model has a unique expansion to O_1 **then** O_2
- O_1 **then %implied** O_2: like %mcons, but O_2 must not extend the signature
- example in OWL:

```
  Class Person
  Class Female
then %def
  Class: Woman  EquivalentTo: Person and Female
```

References to Named Ontologies

- Reference to an ontology existing on the Web
- written directly as a URL (or IRI)
- Prefixing may be used for abbreviation

```
http://owl.cs.manchester.ac.uk/co-ode-files/
ontologies/pizza.owl
```

```
co-ode:pizza.owl
```

Unions

- O_1 **and** O_2: union of two stand-alone ontologies (for extensions O_2 needs to be basic)
- Signatures (and axioms) are united
- model classes are intersected

```
algebra:Monoid and algebra:Commutative
```

Translations

- **O with σ**, where σ is a signature morphism
  - semantics: all models whose σ-reduct is an O-model
- **O with translation ρ**, where ρ is a logic translation
  - semantics: all models whose ρ-reduct is an O-model

```
ObjectProperty: isProperPartOf
   Characteristics: Asymmetric
   SubPropertyOf: isPartOf
with translation trans:SROIQtoCL
then
 (if (and (isProperPartOf x y) (isProperPartOf y z))
     (isProperPartOf x z))
 %% transitivity; can't be expressed in OWL together
 %% with asymmetry
```

Reductions

- O **hide** σ, where σ is a signature morphism
  - semantics: the σ-reducts of all O-models
    i.e. some logical or non-logical symbols are hidden, but the semantic effect of sentences (also those involving these symbols) is kept
- O **hide along** μ, where μ is a logic projection
  - semantics: the μ-reducts of all O-models

```
sort Elem
ops 0:Elem; __+__:Elem*Elem->Elem; inv:Elem->Elem
forall x,y,z . 0+x=x
             . x+(y+z) = (X+y)+z
             . x+inv(x)=0
hide inv
```

Approximations

- *O* **approximate in** *l* **with** *m*
- approximation of *O* in a sublogic *l* using method *m*
- sentences not expressible in *l* are weakened or removed, as specified by *m*.
- Requirement: $O \models O$ **approximate in** *l* **with** *m*
- Not necessarily maximal with these properties (indeed, maximal such ontologies not always exist).

`DOLCE_Mereology` **`approximate in`** `OWL` **`with`** `luettich`

Module Extractions

- O **extract** c Σ **with** m
- Σ: restriction signature (subsignature of that of O)
- c: one of %mcons and %ccons
- m: module extraction method

O must be a conservative extension of the resulting extracted module.

```
co-ode:Pizza extract %mcons
  Class: VegetarianPizza
  Class: VegetableTopping
  ObjectProperty: hasTopping
  with locality
```

Minimizations

- **minimize { *O* }**
- forces mimimal interpretation of non-logical symbols in *O*

```
  Class: Block
  Individual: B1 Types: Block
  Individual: B2 Types: Block DifferentFrom: B1
then minimize {
        Class: Abnormal
        Individual: B1 Types: Abnormal }
then
  Class: Ontable
  Class: BlockNotAbnormal EquivalentTo:
    Block and not Abnormal SubClassOf: Ontable
then %implied
  Individual: B2 Types: Ontable
```

Ontology definitions

- **ontology** $Id = O$ **end**
- assigns name (resp. IRI) Id to ontology O, for later reference.

```
ontology co-code:Pizza =
  Class: VegetarianPizza
  Class: VegetableTopping
  ObjectProperty: hasTopping
  ...
end
```

Interpretations

- **interpretation** Id : O_1 **to** $O_2 = \sigma$
- σ is a signature morphism or a logic translation
- expresses that $O_2 \models \sigma(O_1)$

```
interpretation i : TotalOrder to Nat = Elem |-> Nat
interpretation geometry_of_time %mcons :
 %% Interpretation of linearly ordered time intervals..
  int:owltime_le
 %% ... that begin and end with an instant as lines
 %% that are incident with linearly ...
  to { ord:linear_ordering and bi:complete_graphical
%% ... ordered points in a special geometry, ...
      and  int:mappings/owltime_interval_reduction }
  = ProperInterval |-> Interval end
```

Equivalences

- **equivalence** $Id : O_1 \leftrightarrow O_2 = O_3$
- expresses that O_1 and O_2 have model classes that are in bijective correspondence
- (fragment) ontology O_3 is such that O_i **then** O_3 is a definitional extension of O_i for $i = 1, 2$.

```
equivalence e : algebra:BooleanAlgebra
                to algebra:BooleanRing =
  <define missing symbols>
end
```

Module Relations

- **module** *Id* c : O_1 **of** O_2 **for** Σ
- O_1 is a module of O_2 with restriction signature Σ and conservativity c

  c=%mcons every Σ-reduct of an O_1-model can be expanded to an O_2-model

  c=%ccons every Σ-sentence φ following from O_1 already follows from O_1

This relation shall hold for any module O_1 extracted from O_2 using the **extract** construct.

Alignments

- **alignment** *Id* $card_1$ $card_2$: O_1 **to** O_2 $= c_1, \ldots c_n$
- $card_i$ is (optionally) one of 1, ?, $+$, *
- the c_i are correspondences of form sym_1 *rel* *conf* sym_2
  - sym_i is a symbol from O_i
  - *rel* is one of $>$, $<$, $=$, %, $\ni$, $\in$, $\mapsto$, or an *Id*
  - *conf* is an (optional) confidence value between 0 and 1

Syntax of alignments follows the alignment API
`http://alignapi.gforge.inria.fr`

```
alignment Alignment1 : { Class: Woman } to { Class: Person } =
  Woman < Person
end
```

Alignment: Another Example

```
ontology Onto1 =
  Class: Person
  Class: Woman SubClassOf: Person
  Class: Bank
end

ontology Onto2 =
  Class: HumanBeing
  Class: Woman SubClassOf: HumanBeing
  Class: Bank
end

alignment VAlignment : Onto1 to Onto2 =
  Person = HumanBeing,
  Woman = Woman
end
```

Combinations

- **combine** $O_1, \ldots, O_n$ $L_1, \ldots, L_m$
- L_j are links (interpretations, alignments) between ontologies
- The individual ontologies can be prefixed with labels, like $n : O$
- semantics is a colimit

```
ontology AlignedOntology1 =
  combine Alignment1

ontology VAlignedOntology =
  combine 1 : Onto1, 2 : Onto2, VAlignment
  %% 1:Person is identified with 2:HumanBeing
  %% 1:Woman is identified with 2:Woman
  %% 1:Bank and 2:Bank are kept distinct

ontology VAlignedOntologyRenamed =
  VAlignedOntology with 1:Bank |-> RiverBank,
    2:Bank |-> FinancialBank, Person_HumanBeing |-> Person
```

Diagram for First Alignment

$$
\begin{tikzcd}
\{Woman\} & & Woman \sqsubseteq Person & & \{Person\} \\
& \{Woman\} \arrow[ul] \arrow[ur] & & \{Person\} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

Colimit for First Alignment

$$\begin{tikzcd}[column sep=small]
 & & & Woman \sqsubseteq Person & & & \\
\{Woman\} \arrow[urrr, dotted] & & & Woman \sqsubseteq Person \arrow[u, dotted] & & & \{Person\} \arrow[ulll, dotted] \\
 & \{Woman\} \arrow[ul] \arrow[urr] & & & & \{Person\} \arrow[ull] \arrow[ur] &
\end{tikzcd}$$

Diagram for Second Alignment

$$\begin{tikzcd}
Onto1 & & Onto2 \\
& \{Woman, Person_HumanBeing\} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Colimit for Second Alignment

$$
\begin{tikzcd}
 & \{Woman, Person\_HumanBeing, 1:Bank, 2:Bank\} & \\
Onto1 \arrow[ur, dotted] & & Onto2 \arrow[ul, dotted] \\
 & \{Woman, Person\_HumanBeing\} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

Construction of Diagrams

$$
\begin{tikzcd}
O_1 & & B & & O_2 \\
& O_1' \arrow[ul] \arrow[ur] & & O_2' \arrow[ul] \arrow[ur] & \\
& & O_1\_O_2 \arrow[uull, bend left, "\sigma_1"] \arrow[uurr, bend right, "\sigma_2"'] & &
\end{tikzcd}
$$

- $O_1\_O_2$ contains, for each $s_1 = s_2$ in A, a symbol $s_1\_s_2$
- O_1' and O_2' contain the symbols of O_1 and O_2 , respectively, which appear in A in a correspondence $s_1 \quad s_2$ such that is not equivalence and B is an ontology constructed
- the signature morphisms σ_1 and σ_2 map each symbol $s_1\_s_2$ to s_1 and respectively s_2.

Ontology Bridges

- O_1 **bridge with translation** t O_2
- t is a logic translation
- semantics: O_1 **with translation** t **then** O_2
- t will e.g. translate OWL to some DDL or $\mathcal{E}$-connections
- O_2: axioms involving the relations (introduced by t) between ontologies in O_1.

Ontology Bridge Example

```
ontology Publications1 =
  Class: Publication
  Class: Article SubClassOf: Publication
  Class: InBook SubClassOf: Publication
  Class: Thesis  SubClassOf: Publication
  ...

ontology Publications2 =
  Class: Thing
  Class: Article SubClassOf: Thing
  Class: BookArticle SubClassOf: Thing
  Class: Publication SubClassOf: Thing
  Class: Thesis  SubClassOf: Thing
```

Ontology Bridge Example, cont'd

```
ontology Publications_Combined =
combine
  1 : Publications1 with translation OWL2MS-OWL,
  2 : Publications2 with translation OWL2MS-OWL
  %% implicitly: Article ↦ 1:Article ...
  %%             Article ↦ 2:Article ...
bridge with translation MS-OWL2DDL
  %% implicitly added my translation MS-OWL2DDL: binary
```

1:Publication $\xrightarrow{\sqsubseteq}$ 2:Publication

1:PhdThesis $\xrightarrow{\sqsubseteq}$ 2:Thesis

1:InBook $\xrightarrow{\sqsubseteq}$ 2:BookArticle

1:Article $\xrightarrow{\sqsubseteq}$ 2:Article

1:Article $\xrightarrow{\sqsupseteq}$ 2:Article

Qualifications

Qualifications choose the logic, ontology language and/or serialization:

- **language** l
- **logic** l
- **serialization** s

This affects the subsequent definitions in the distributed ontology.

What is the Needed Abstract Infrastructure?

- models, sentences, satisfaction $\Rightarrow$ satisfaction systems
- reducts, (conservative) extensions $\Rightarrow$ institutes
- combinations via colimits $\Rightarrow$ institutions
- ...

Satisfaction Systems

A satisfaction system $S = (Sen, \mathcal{M}, \models)$ consists of

- a class *Sen* of sentences;
- a class *Mod* of models;
- a satisfaction relation $\models \subseteq Mod \times Sen$.

Institutes

An institute $\mathcal{I} = (Sig, \leq, Sen, Mod, \models)$ consists of

- a class Sen of sentences;
- a partially ordered class $(Sig, \leq)$ of signatures;
- a function $sig : Sen \rightarrow Sig$, giving the (minimal) signature of a sentence (then for each signature Σ, let $Sen(\Sigma) = \{\varphi \in Sen \mid sig(\varphi) \leq \Sigma\}$);
- for each signature Σ, a partially ordered class $Mod(\Sigma)$ of Σ-models;
- for each signature Σ, a satisfaction relation $\models_\Sigma \subseteq Mod(\Sigma) \times Sen(\Sigma)$;
- for any Σ_2-model M, a Σ_1-model $M|_{\Sigma_1}$ (called the reduct), provided that $\Sigma_1 \leq \Sigma_2$,

Institutes (cont'd)

…such that the following properties hold:

- given $\Sigma_1 \leq \Sigma_2$, for any Σ_2-model M and any Σ_1-sentence φ

$$M \models \varphi \text{ iff } M|_{\Sigma_1} \models \varphi$$

  (satisfaction is invariant under reduct),

- for any Σ-model, $M|_\Sigma = M$, and given $\Sigma_1 \leq \Sigma_2 \leq \Sigma$,

$$(M|_{\Sigma_2})|_{\Sigma_1} = M|_{\Sigma_1}$$

  (reducts are compositional), and

- for any signatures $\Sigma' \leq \Sigma$, and Σ-models $M_1 \leq M_2$, we have $M_1|_{\Sigma'} \leq M_2|_{\Sigma'}$ (reducts preserve the model ordering).

Institutions

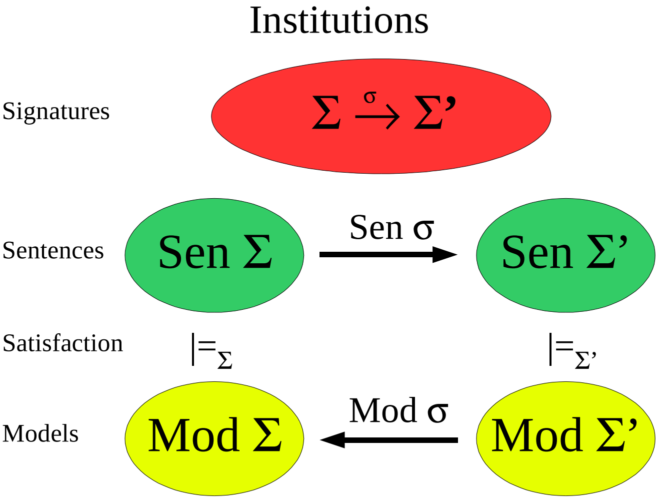
An institution $\mathcal{I}$ consists of:

- a category $\mathbf{Sign}_{\mathcal{I}}$ of signatures;
- a functor $\mathbf{Sen}_{\mathcal{I}} \colon \mathbf{Sign}_{\mathcal{I}} \to \mathbf{Set}$,giving a set $\mathbf{Sen}(\Sigma)$ of Σ-sentences for each signature $\Sigma \in |\mathbf{Sign}_{\mathcal{I}}|$, and a function $\mathbf{Sen}(\sigma) \colon \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ that yields σ-translation of Σ-sentences to Σ'-sentences for each $\sigma \colon \Sigma \to \Sigma'$;
- a functor $\mathbf{Mod}_{\mathcal{I}} \colon \mathbf{Sign}_{\mathcal{I}}^{op} \to \mathbf{Set}$,giving a set $\mathbf{Mod}(\Sigma)$ of Σ-models for each signature $\Sigma \in |\mathbf{Sign}_{\mathcal{I}}|$, and a functor $\_|_{\sigma} = \mathbf{Mod}(\sigma) \colon \mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$; for each $\sigma \colon \Sigma \to \Sigma'$;
- for each $\Sigma \in |\mathbf{Sign}_{\mathcal{I}}|$, a satisfaction relation $\models_{\mathcal{I},\Sigma} \ \subseteq \mathbf{Mod}_{\mathcal{I}}(\Sigma) \times \mathbf{Sen}_{\mathcal{I}}(\Sigma)$

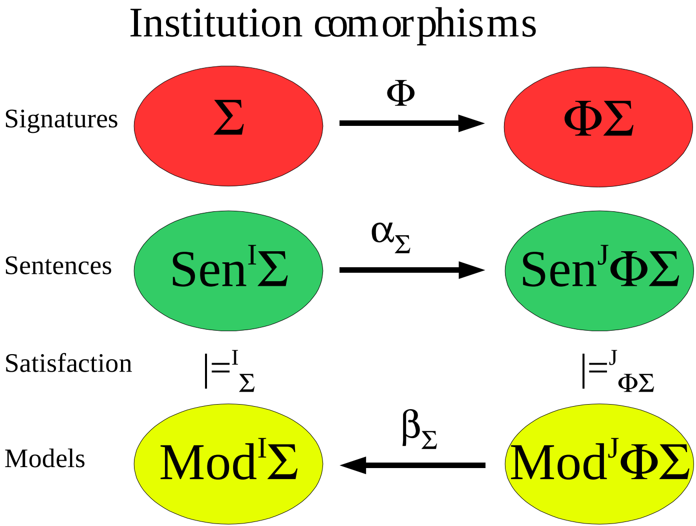
such that for any signature morphism $\sigma \colon \Sigma \to \Sigma'$, Σ-sentence $\varphi \in \mathbf{Sen}_{\mathcal{I}}(\Sigma)$ and Σ'-model $M' \in \mathbf{Mod}_{\mathcal{I}}(\Sigma')$:

$$M' \models_{\mathcal{I},\Sigma'} \sigma(\varphi) \iff M'|_{\sigma} \models_{\mathcal{I},\Sigma} \varphi \qquad \text{[Satisfaction condition]}$$

Institutions

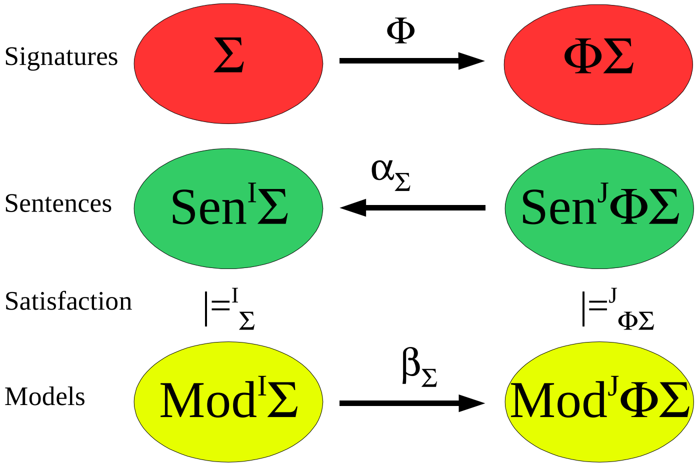


Institution Comorphisms (Translations)



Institution Morphisms (Projections)

Institution morphisms



Formal Semantics of DOL Ontologies

- $(\Sigma, \mathcal{M}, L)$
- L is a logic (institute, institution) in the logic graph
- Σ is a signature in L
- $\mathcal{M}$ is a class of Σ-models in L

Challenges

- What is a suitable abstract meta framework for non-monotonic logics and rule languages like RIF and RuleML? Are institutions suitable here? Are the modularity questions for these languages different from those for OWL?
- What is a useful abstract notion of query (language) and answer substitution?
- How to integrate TBox-like and ABox-like ontologies?
- Can the notions of class hierarchy and of satisfiability of a class be generalised from OWL to other languages?
- How to interpret alignment correspondences with confidence other that 1 in a combination?
- Can logical frameworks be used for the specification of ontology languages and translations?

Tool support: Heterogeneous Tool Set (Hets)

- available at `hets.dfki.de`
- speaks DOL, OWL, Common Logic, and other languages
- analysis
- management of proof obligations
- theorem proving, model finding

Tool support: Ontohub web portal and repository

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

- prototype available at `ontohub.org`
- speaks DOL, OWL, Common Logic, and other languages
- mid-term goal: follow the Open Ontology Repository Initiative (OOR) architecture and API
- API is discussed at `https://github.com/ontohub/OOR_Ontohub_API`
- annual Ontology summit as a venue for review, and discussion

Conclusion

- DOL covers many aspects of modularity that have been discussed in the different WoMos
- DOL ist a meta language, focussing on relations between ontologies ("ontology-in-the large")
- you can help with joining the OntoIOp discussion
    - see `ontoiop.org`