The Distributed Ontology, Modeling and Specification Language (DOL)

Till Mossakowski[1] Oliver Kutz[1]
Christoph Lange[2] Mihai Codescu[1]

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

[1]University of Magdeburg

[2]University of Bonn

OntoIOp teleconference, 2014-01-29

Motivation

The Big Picture of Interoperability

Modeling	**Specification**	**Knowledge engineering**
Objects/data	Software	Concepts/data
Models	Specifications	Ontologies
Metamodels	Specification languages	Ontology languages

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

(Formal) ontologies, (formal) specifications and (formal) models will henceforth be abbreviated as OSMs.

Ontology 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?

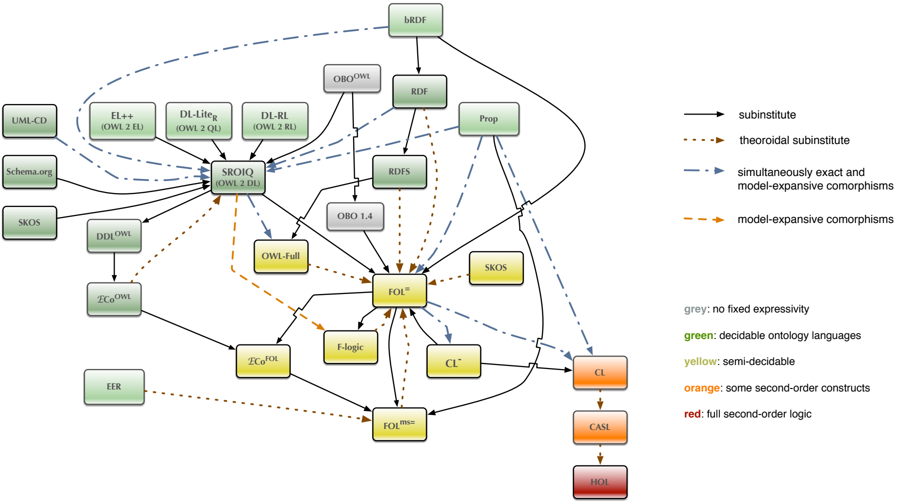
Use Case: Refinement of specifications

- refinement from requirements to design to code
- many different formalisms
- formalism may change during formal development
- yet, some general mechanism of refinements are always the same

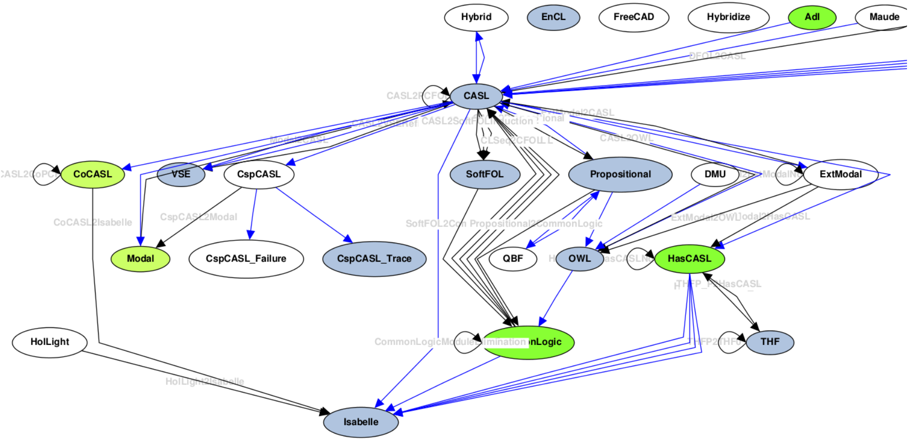
Use Case: Consistency and satisfiability among UML models

- does an object diagram satisfy a class diagram?
- Does a state machine satisfy an OCL specification?
- Do the protocal state machines at the ends of a connector fit together?
- Does a state machine refine the protocol state machines in a structure diagram?

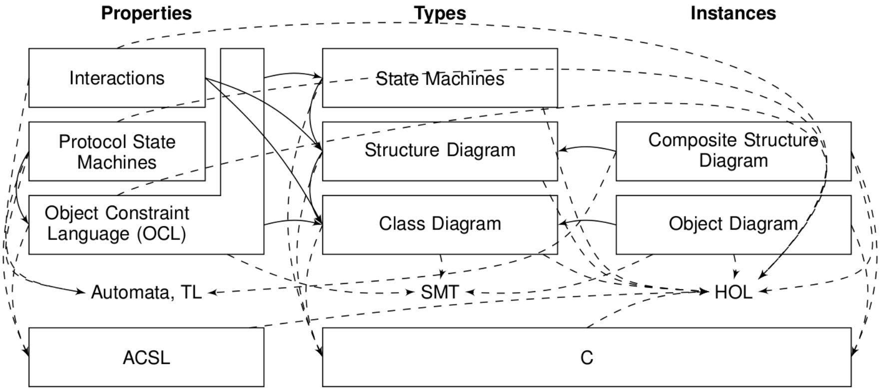
Ontologies: An Initial Logic Graph



Specifications: An Initial Logic Graph



UML models: An Initial Logic Graph



Motivation: Diversity of Operations on and Relations among OSMs

Various operations and relations on OSMs are in use:

- structuring: union, translation, hiding, ...
- refinement
- matching and alignment
  - of many OSMs covering one domain
- module extraction
  - get relevant information out of large OSM
- approximation
  - model in an expressive language, reason fast in a lightweight one
- ontology-based database access/data management
- distributed OSMs
  - bridges between different modellings

OntoIOp

Need for a Unifying Meta Language

Not yet another OSM language, but a meta language covering

- diversity of OSM languages
- translations between these
- diversity of operations on and relations among OSMs

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 OntolOp 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) has been issued in December 2013
    - proposals answering RFP due in December 2014
- 50 experts participate, $\sim$ 15 have contributed
- OntolOp is open for your ideas, so join us!

Requirements in the OMG RFP OntoIOp

- provide a meta-language for:
  - logically heterogeneous OSMs
  - modular OSMs
  - module extraction, approximation
  - links (interpretations, alignments) between OSMs/modules
  - combination of OSMs 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. OSMs consist of symbols and axioms

DOL

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 institutions and (co)morphisms
- DOL has a model-level and a theory-level semantics

Related work

- Structured specifications and their semantics (Clear, ASL, CASL, . . .)
- Heterogeneous specification (HetCASL)
- modular ontologies (WoMo workshop series)

Overview of DOL

1. modular and heterogeneous OSMs
   - basic OSMs (flattenable)
   - references to named OSMs
   - extensions, unions, translations (flattenable)
   - reductions (elusive)
   - approximations, module extractions (flattenable)
   - minimization, maximization (elusive)
   - combination, OSM bridges (flattenable)

   only OSMs with flattenable components are flattenable
2. OSM declarations and relations (based on 1)
   - OSM definitions (giving a name to an OSM)
   - interpretations (of theories)
   - equivalences
   - module relations
   - alignments

Modular and Heterogeneous OSMs

Basic OSMs

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

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

- e.g. in Common Logic:

```
(cl-text 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 %implies** 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 OSMs

- Reference to an OSM 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 OSMs (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
- *O* **with translation** ρ, where ρ is an institution comorphism

```
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

- intuition: some logical or non-logical symbols are hidden, but the semantic effect of sentences (also those involving these symbols) is kept
- O **reveal** Σ, where Σ is a subsignature of that of O
- O **hide** Σ, where Σ is a subsignature of that of O
- O **hide along** μ, where μ is an institution morphism

```
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
```

Interpolation

- O **keep in** Σ, where Σ is a subsignature of that of O
- O **keep in** Σ **with** I, where Σ is a subsignature of that of O, and I is a subinstitution of that of O
    - intuition: theory of O is interpolated in smaller signature/logic
- dually
    - O **forget** Σ
    - O **forget** Σ **with** I

```
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
forget inv
```

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
```

- Dually: *O* **remove** *c* Σ **with** *m*

Extract – Forget – Hide

	remove/extract	forget/keep	hide/reveal
semantic background	conservative extension	uniform interpolation	model reduct
relation to original	subtheory	interpretable	interpretable
approach	theory level	theory level	model level
type of ontology	flattenable	flattenable	elusive
signature of result	$\geq \Sigma$	$= \Sigma$	$= \Sigma$
change of logic	not possible	possible	possible

Minimizations (circumscription)

- O_1 **then minimize** { O_2 }
- forces mimimal interpretation of non-logical symbols in O_2

```
  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
```

Freeness

- O_1 **then free {** O_2 **}**
- forces initital interpretation of non-logical symbols in O_2

```
sort Elem
then free {
   sort Bag
   ops mt:Bag;
       __union__:Bag∗Bag->Bag, assoc, comm, unit mt
         }
```

Cofreeness

- O_1 **then cofree** { O_2 }
- forces final interpretation of non-logical symbols in O_2

```
sort Elem
then cofree {
   sort Stream
   ops head:Stream->Elem;
       tail:Stream->Stream
         }
```

OSM declarations and relations

OSM definitions

- **OSM** *IRI* = *O* **end**
- assigns name *IRI* to OSM *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 logically implies $\sigma(O_1)$

```
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** $Id : O_1 \leftrightarrow O_2 = O_3$
- (fragment) OSM O_3 is such that O_i **then %def** O_3 is a definitional extension of O_i for $i = 1, 2$;
- this implies that O_1 and O_2 have model classes that are in bijective correspondence

```
equivalence e : algebra:BooleanAlgebra
                <-> algebra:BooleanRing =
    x∧y = x·y
    x∨y = x+y+x·y
    ¬x = 1+x
    x·y = x∧y
    x+y = (x∨y) ∧ ¬(x∧y)
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

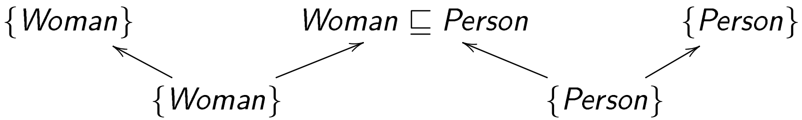
- combine $O_1, \ldots, O_n$ $L_1, \ldots, L_m$
- L_j are links (interpretations, alignments) between OSMs
- The individual OSMs can be prefixed with labels, like $n : O$
- semantics is a colimit

```
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



Colimit for First Alignment

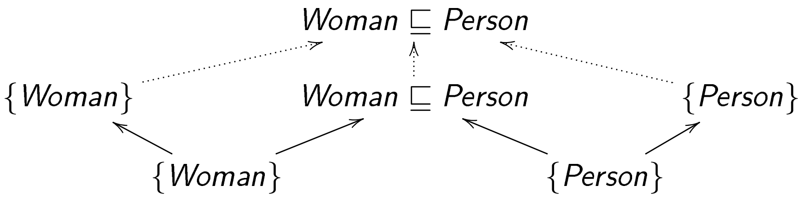
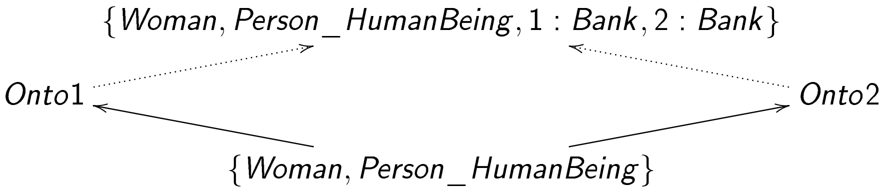
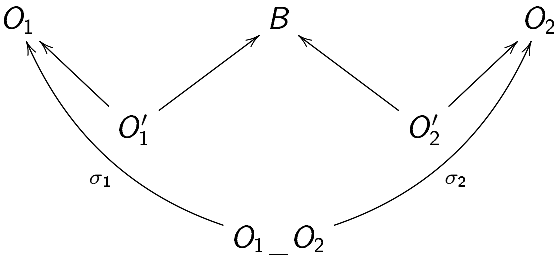


Diagram for Second Alignment

Colimit for Second Alignment



Construction of Diagrams



- $O_1\_O_2$ 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 $s_1\ R\ s_2$ such that R is not equivalence and B is an OSM constructed
- the signature morphisms σ_1 and σ_2 map each symbol $s_1\_s_2$ to s_1 and respectively s_2.

OSM 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 OSMs in O_1.

OSM 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
```

OSM 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: binar
```

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, OSM language and/or serialization:

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

This affects the subsequent declarations and relations in the distributed OSM.

Conclusion

Challenges

- What is a suitable abstract meta framework for non-monotonic logics and rule languages like RIF and RuleML? Are institutions suitable here? 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 OSMs?
- 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 OSM languages and translations?
- Proof support

Tool support: Heterogeneous Tool Set (Hets)

- available at `hets.dfki.de`
- speaks DOL, HetCASL, CoCASL, CspCASL, MOF, QVT, OWL, Common Logic, and other languages
- analysis
- computation of colimits
- 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) OSMs

- 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 is a meta language for (formal) ontologies, specifications and models (OSMs)
- DOL covers many aspects of modularity of and relations among OSMs ("OSM-in-the large")
- DOL will be submitted to the OMG as an answer to the OntoIOp RFP
- you can help with joining the OntoIOp discussion
  - see `ontoiop.org`