The Distributed Ontology, Modeling and Specification Language (DOL)
Language overview

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

[1]University of Magdeburg
[2]University of Bonn

OntoIOp telecon, 2014-09-09

Motivation

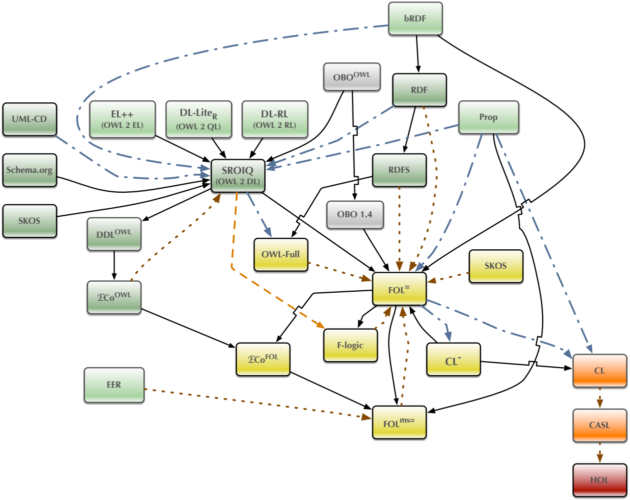
The Big Picture of Interoperability

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) models and (formal) specifications will henceforth be abbreviated as OMS.

Ontologies: An Initial Logic Graph



→ subinstitute

⇢ theoroidal subinstitute

⇢ simultaneously exact and model-expansive comorphisms

⇢ model-expansive comorphisms

grey: no fixed expressivity

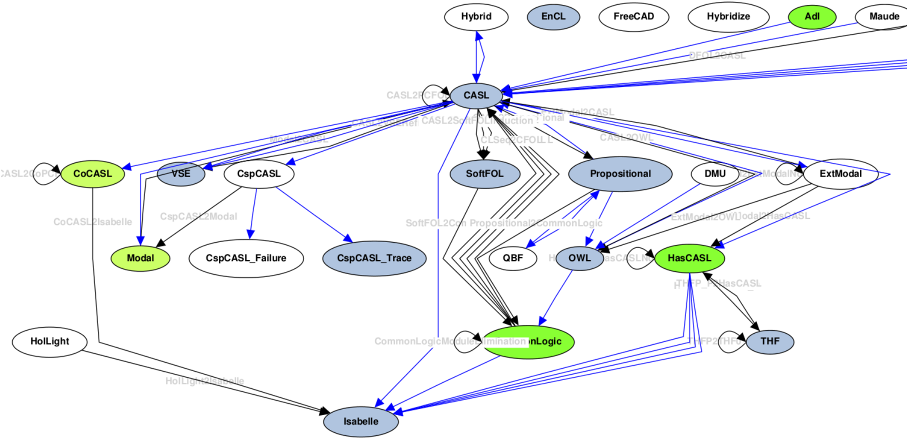
green: decidable ontology languages

yellow: semi-decidable

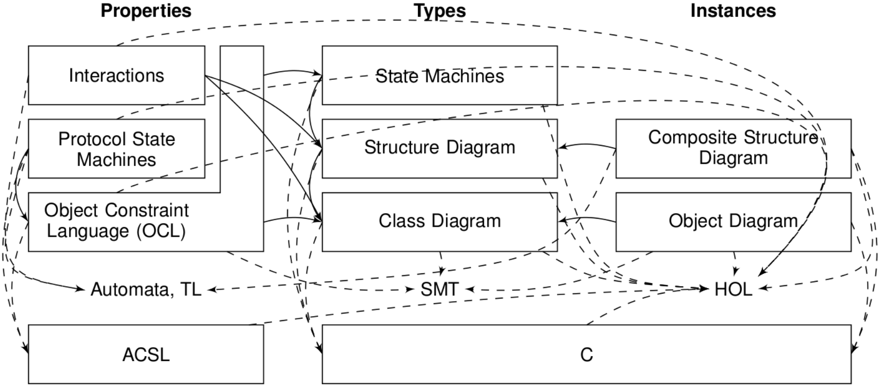
orange: some second-order constructs

red: full second-order logic

Specifications: An Initial Logic Graph



UML models: An Initial Logic Graph



Motivation: Diversity of Operations on and Relations among OMS

Various operations and relations on OMS are in use:

- structuring: union, translation, hiding, . . .
- refinement
- matching and 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
- ontology-based database access/data management
- distributed OMS
    - bridges between different modellings

OntoIOp

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

The OntoIOp initiative (ontoiop.org)

- 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
- OntoIOp is open for your ideas, so join us!
- Distributed Ontology, Modeling and Specification Language
    - DOL = one specific answer to the RFP requirements
    - there may be other answers to the RFP
    - DOL is based on some graph of institutions and (co)morphisms
    - DOL has a model-level and a theory-level semantics

DOL

Overview of DOL

1. Focused OMS
    - basic OMS (flattenable)
    - references to named OMS
    - extensions, unions, translations (flattenable)
    - reductions (elusive)
    - approximations, module extractions (flattenable)
    - minimization, maximization (elusive)
    - combination, OMS bridges (flattenable)

   only OMS with flattenable components are flattenable
   flattenable = can be flattened to a basic OMS
2. Distributed OMS (based on focused OMS)
    - OMS definitions (giving a name to an OMS)
    - interpretations (of theories), equivalences
    - module relations
    - alignments

Focused OMS

```
BasicOMS       ::= OMSInConformingLanguage
MinimizableOMS ::= BasicOMS | OMSRef [ImportName]
ExtendingOMS   ::= MinimizableOMS
                     | MinimizeKeyword '{' MinimizableOMS '}'
                     | OMS Extraction
OMS            ::= ExtendingOMS
                     | OMS Minimization
                     | OMS Translation
                     | OMS Reduction
                     | OMS Approximation
                     | OMS Filtering
                     | OMS 'and' [ConsStrength] OMS
                     | OMS 'then' ExtensionOMS
                     | Qualification∗ ':' GroupOMS
                     | OMS 'bridge' Translation∗ OMS
                     | 'combine' GraphElements [ExcludeExtensions]
                     | 'apply' SubstName Sentence
GroupOMS           ::= '{' OMS '}' | OMSRef
ImportName         ::= '%(' IRI ')%'
```

Basic OMS

- written in some OMS language from the logic graph
- semantics is inherited from the OMS 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))))
```

```
ExtensionOMS       ::= [ExtConsStrength] [ExtensionName] ExtendingOMS
ExtensionName      ::= ’%(’ IRI ’)%’
```

Extensions

- O_1 **then** O_2: extension of O_1 by new symbols and axioms O_2
- example in OWL:

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

```
ExtensionOMS       ::= [ExtConsStrength] [ExtensionName] ExtendingOMS
ConsStrength       ::= Conservative | '%mono' | '%wdef' | '%def'
ExtConsStrength    ::= ConsStrength | '%implied'
Conservative       ::= '%ccons' | '%mcons'
ExtensionName      ::= '%(' IRI ')%'
```

Extensions with annotations

- 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 OMS

- Reference to an OMS 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`

Semantics Reference to Named OMS: $[\![iri]\!]_\Gamma = \Gamma(iri)$

Unions

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

`algebra:Monoid and algebra:Commutative`

```
Translation        ::= 'with' LogicTranslation∗ [SymbolMapItems]
SymbolMapItems     ::= SymbolOrMap ( ',' SymbolOrMap )∗
LogicTranslation   ::= 'translation' OMSLangTrans
SymbolMap          ::= Symbol '$\mapsto$' Symbol
SymbolOrMap        ::= Symbol | SymbolMap
LoLaRef            ::= LanguageRef | LogicRef
OMSLangTrans       ::= OMSLangTransRef | '<$\to$>' LoLaRef
OMSLangTransRef    ::= IRI
```

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
```

Hide – Extract – Forget – Filter

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

```
Reduction          ::= 'hide' LogicReduction∗ [SymbolItems]
                     | 'reveal' [SymbolMapItems]
SymbolItems        ::= Symbol ( ',' Symbol )∗
LogicReduction     ::= 'along' OMSLangTrans
```

Reduction: Hide/reveal

- 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

Reduction: example

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

Semantics: class of all monoids that can be extended with an inverse, i.e. class of all groups. The effect is second-order quantification:

```
sort Elem
ops 0:Elem; __+__:Elem*Elem->Elem;
exists inv:Elem->Elem .
  forall x,y,z:elem . 0+x=x
                    /\ x+(y+z) = (x+y)+z
                    /\ x+inv(x)=0
```

```
Extraction         ::= 'extract' ModuleProperties InterfaceSignature
                     | 'remove' ModuleProperties InterfaceSignature
ModuleProperties   ::= Conservative | '%min' | '%depliting' | '%safe'
InterfaceSignature ::= SymbolItems
SymbolItems        ::= Symbol ( ',' Symbol )*
```

Module Extraction: remove/extract

O extract Σ

- Σ: restriction signature (subsignature of that of O)
- O must be a conservative extension of the resulting extracted module. (If not, the module is suitably enlarged.)
- Dually: **O remove Σ**
- Note: The extraction methods from the literature all guarantee model-theoretic conservativity.

Module Extraction: example

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

The semantics is the following theory:

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

The module needs to be enlarged to the whole OMS.

Module Extraction: 2nd example

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

The semantics is the following theory:

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

Here, adding `inv` is conservative.

```
Approximation       ::= 'forget' InterfaceSignature ['with' LogicRef]
                      | 'keep' InterfaceSignature ['with' LogicRef]
InterfaceSignature ::= SymbolItems
SymbolItems         ::= Symbol ( ',' Symbol )*
```

Interpolation: forget/keep

- 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

Interpolation: example

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

The semantics is the following theory:

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

Computing interpolants can be hard, even undecidable.

```
Filtering          ::= 'filter' BasicOMS
```

Filtering

- O **filter** T, where T is a subtheory (fragment) of that of O
  - intuition: all axioms involving symbols in $Sig(T)$ are deleted
  - moreover, all axioms contained in T are deleted as well
- A dual notion does not make much sense (indeed, just T would be delivered).

Filtering: example

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

The semantics is the following theory:

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

Hide – Extract – Forget – Filter

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

Relations among the different notions

$$\begin{aligned} & Mod(O\ \textbf{hide}\ \Sigma) \\ = \quad & Mod(O\ \textbf{extract}\ \Sigma)|_{Sig(O)\setminus\Sigma} \\ \subseteq \quad & Mod(O\ \textbf{forget}\ \Sigma) \\ \subseteq \quad & Mod(O\ \textbf{filter}\ \Sigma) \end{aligned}$$

Pros and Cons

	hide/reveal	remove/extract	forget/keep	filter
information loss	none	none	minimal	large
computability	bad	good/depends	depends	easy
signature of result	$= \Sigma$	$\geq \Sigma$	$= \Sigma$	$= \Sigma$
change of logic	possible	not possible	possible	not possible
conceptual simplicity	simple (but unintuitive)	complex	farily simple	simple

```
Minimization       ::= MinimizeKeyword CircMin [CircVars]
MinimizeKeyword    ::= ’minimize’
                     | ’closed-world’
                     | ’maximize’
                     | ’free’
                     | ’cofree’
CircMin            ::= Symbol Symbol∗
CircVars           ::= ’vars’ (Symbol Symbol∗)
```

Minimizations (circumscription)

- O_1 **then minimize {** O_2 **}**
- forces minimal 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 initial 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
         }
```

Distributed OMS

```
DistOMS                 ::= [PrefixMap] DistOMSDefn
                          | OMSInConformingLanguage
DistOMSDefn             ::= ’distributed OMS’ DistOMSName DistOMSItem∗
OMSInConformingLanguage ::= ($<$) language and serialization specific ($>$)
DistOMSItem             ::= OMSDefn | MappingDefn | Qualification
Qualification           ::= LanguageQual | LogicQual | SyntaxQual
LanguageQual            ::= ’language’ LanguageRef
LogicQual               ::= ’logic’ LogicRef
SyntaxQual              ::= ’serialization’ SyntaxRef
DistOMSName             ::= IRI
PrefixMap               ::= ’%prefix(’ PrefixBinding∗ ’)%’
PrefixBinding           ::= BoundPrefix IRIBoundToPrefix
BoundPrefix             ::= ’:’ | Prefix
OMSkeyword              ::= ’ontology’
                          | ’onto’
                          | ’specification’
                          | ’spec’
                          | ’model’
OMSDefn                 ::= OMSkeyword OMSName ’=’ [ConsStrength] OMS [’end’]
```

OMS definitions

- **OMS** *IRI* $=$ *O* **end**
- assigns name *IRI* to OMS *O*, for later reference$\Gamma(IRI) := [\![O]\!]_\Gamma$

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

```
MappingDefn          ::= IntprDefn
                       | EquivDefn
                       | ModuleRelDefn
                       | AlignDefn
IntprDefn            ::= IntprKeyword IntprName [Conservative] ’:’
                         [’end’]
                       | IntprKeyword IntprName [Conservative] ’:’
                         ’=’ LogicTranslation∗ [SymbolMapItems]
IntprKeyword         ::= ’interpretation’ | ’view’
IntprName            ::= IRI
IntprType            ::= GroupOMS ’to’ GroupOMS
```

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
```

```
OMSOrMappingorGraphRef ::= IRI
GraphElements       ::= GraphElement ( ',' GraphElement )∗
GraphElement        ::= [Id ':'] OMSOrMappingorGraphRef
ExcludeExtensions   ::= 'excluding' ExtensionRef ( ',' ExtensionRef )∗
```

Graphs (diagrams)

graph G =
$G_1, \ldots, G_m, O_1, \ldots, O_n, M_1, \ldots, M_p$
excluding $G'_1, \ldots, G'_i, O'_1, \ldots, O'_j, M'_1, \ldots, M'_k$

- G_i are other graphs
- O_i are OMS (possibly prefixed with labels, like $n : O$)
- M_i are mappings (views, interpretations)

Combinations

- combine G
- G is a graph
- semantics is the (a) colimit of the diagram G

```
ontology AlignedOntology1 =
  combine G
```

There is a natural semantics of diagrams: compatible families of models.
Then in exact institutions, models of diagrams are in bijective correspondence to models of the colimit.

Sample combination

```
ontology Source =
  Class: Person
  Class: Woman SubClassOf: Person
ontology Onto1 =
  Class: Person           Class: Bank
  Class: Woman SubClassOf: Person
interpretation I1 : Source to Onto1 =
   Person |-> Person, Woman |-> Woman
ontology Onto2 =
  Class: HumanBeing       Class: Bank
  Class: Woman SubClassOf: HumanBeing
interpretation I2 : Source to Onto2 =
   Person |-> HumanBeing, Woman |-> Woman
ontology CombinedOntology =
  combine Source, Onto1, Onto2, I1, I2
```

Resulting colimit

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

```
AlignDefn          ::= 'alignment' AlignName [AlignCards] ':'
                       ['end']
                     | 'alignment' AlignName [AlignCards] ':'
                       '=' Correspondence ( ',' Correspondence )*
AlignName          ::= IRI
AlignCards         ::= AlignCardForward AlignCardBackward
AlignCardForward   ::= AlignCard
AlignCardBackward  ::= AlignCard
AlignCard          ::= '1' | '?' | '+' | '*'
AlignType          ::= GroupOMS 'to' GroupOMS<\CLnote[type=q-aut]{would it make se
Correspondence     ::= CorrespondenceBlock | SingleCorrespondence | '*'
CorrespondenceBlock ::= 'relation' [RelationRef] [Confidence] '{'
                        ( ',' Correspondence )* '}'
SingleCorrespondence ::= SymbolRef [RelationRef] [Confidence]
                         [CorrespondenceId]
CorrespondenceId   ::= '%(' IRI ')%'
SymbolRef          ::= IRI
TermOrSymbolRef    ::= Term | SymbolRef
RelationRef        ::= '<\greaterthan>' | '<\lessthan>'
                     | '=' | '%'
                     | '$\ni$' | '$\in$'
                     | '$\mapsto$' | IRI
Confidence         ::= Double
Double ::= ($<$ a number $\in [0,1]$ $>$)
```

Alignments

- **alignment** Id $card_1$ $card_2$: O_1 **to** O_2 $= c_1, \ldots c_n$
  assuming SingleDomain | GlobalDomain | ContextualizedDomain
- $card_i$ 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: Example

```
ontology S =  Class: Person
  Individual: alex Types: Person
  Class: Child

ontology T =  Class: HumanBeing
  Class: Male SubClassOf: HumanBeing
  Class: Employee

alignment A : S to T =
  Person = HumanBeing
  alex in Male
  Child < not Employee
  assuming GlobalDomain
```

Graphs (diagrams), revisited

`graph G =`
$G_1, \dots, G_m, O_1, \dots, O_n, M_1, \dots, M_p, A_1, \dots, A_r$
`excluding` $G'_1, \dots, G'_i, O'_1, \dots, O'_j, M'_1, \dots, M'_k$

- G_i are other graphs
- O_i are OMS (possibly prefixed with labels, like $n : O$)
- M_i are mappings (views, equivalences)
- A_i are alignments

The resulting diagram G includes (institution-specific) W-alignment diagrams for each alignment A_i. Using **assuming**, assumptions about the domains of all OMS can be specified:

SingleDomain aligned symbols are mapped to each other

GlobalDomain aligned OMS a relativized

ContextualizedDomain alignments are reified as binary relations

Diagram of a SingleDomain alignment

$$\begin{tikzcd}
S & & B & & T \\
& \{Person, alex, Child\} \arrow[ul, "\iota_1"'] \arrow[ur, "\sigma_1"] & & \{Male, HumanBeing, Employee\} \arrow[ul, "\sigma_2"'] \arrow[ur, "\iota_2"] &
\end{tikzcd}$$

where

ontology B =
Class: *Person_HumanBeing*
Class: *Employee*
Class: *Child*
SubClassOf: $\neg$ *Employee*
Individual: *alex*
Types: *Male*

Resulting colimit

The colimit ontology of the diagram of the alignment above is:

ontology B = **Class:** *Person_HumanBeing*
Class: *Employee*
Class: *Male* **SubClassOf:** *Person_HumanBeing*
Class: *Child* **SubClassOf:** ¬ *Employee*
Individual: *alex* **Types:** *Male*, *Person_HumanBeing*

Background Simple semantics of diagrams

Framework: institutions like OWL, FOL, . . .
Ontologies are interpreted over the same domain

$$\begin{tikzcd}
O_1 \arrow[drr, "m_1"'] & O_2 \arrow[dr, "m_2"] & \dots & O_n \arrow[dl, "m_n"] \\
 & & D &
\end{tikzcd}$$

- model for A: (m_1, m_2) such that $m_1(s)\ R\ m_2(t)$ for each $s\ R\ t$ in A
- model for a diagram: family (m_i) of models such that (m_i, m_j) is a model for A_{ij}
- local models of O_j modulo a diagram: jth-projection on models of the diagram

Integrated semantics of diagrams

Framework: different domains reconciled in a global domain

$$\begin{tikzcd}
O_1 \arrow[d, "m"'] & O_2 \arrow[d, "m_2"] & \ldots & O_n \arrow[d, "m_n"] \\
D_1 \arrow[drr, "\gamma_1"'] & D_2 \arrow[dr, "\gamma_2"] & \ldots & D_n \arrow[dl, "\gamma_n"] \\
& & U &
\end{tikzcd}$$

- model for a diagram: family (m_i) of models with equalizing function γ such that $(\gamma_i m_i, \gamma_j m_j)$ is a model for A_{ij}

Relativization of an OWL ontology

Let O be an ontology, define its relativization $\tilde{O}$:

- concepts are concepts of O with a new concept $\top_O$;
- roles and individuals are the same
- axioms:
  - each concept C is subsumed by $\top_O$,
  - each individual i is an instance of $\top_O$,
  - each role r has domain and range $\top_O$.

  and the axioms of O where the following replacement of concept is made:
  - each occurence of $\top$ is replaced by $\top_O$,
  - each concept $\neg C$ is replaced by $\top_O \setminus C$, and
  - each concept $\forall R.C$ is replaced by $\top_O \sqcap \forall R.C$.

Example: integrated semantics

$$\begin{tikzcd}
 & & C & & \\
\tilde{S} \arrow[urr, dotted] & & B \arrow[u, dotted] & & \tilde{T} \arrow[ull, dotted] \\
 & \{\mathit{Person}, \mathit{alex}, \mathit{Child}\} \arrow[ul, "\iota_1"] \arrow[ur, "\sigma_1"'] \arrow[uur, dotted] & & \{\mathit{Male}, \mathit{HumanBeing}, \mathit{Employee}\} \arrow[ul, "\sigma_2"'] \arrow[ur, "\iota_2"] \arrow[uul, dotted] &
\end{tikzcd}$$

where

ontology B =
Class: Thing_S **Class:** Thing_T
Class: *Person_HumanBeing* **SubClassOf:** Thing_S, Thing_T
Class: *Male* **Class:** *Employee*
Class: *Child* **SubClassOf:** Thing_T **and** $\neg$ *Employee*
Individual: *alex* **Types:** *Male*

Example: integrated semantics (cont'd)

ontology C =
Class: *ThingS*
Class: *ThingT*
Class: *Person_HumanBeing* **SubClassOf:** *ThingS*, *ThingC*
Class: *Male* **SubClassOf:** *Person_HumanBeing*
Class: *Employee* **SubClassOf:** *ThingT*
Class: *Child* **SubClassOf:** *ThingS*
Class: *Child* **SubClassOf:** *ThingT* **and** ¬ *Employee*
Individual: *alex* **Types:** *Male*, *Person_HumanBeing*

Contextualized semantics of diagrams

Framework: different domains related by coherent relations

$$\begin{tikzcd}
O_1 \arrow[d, "m"'] & O_2 \arrow[d, "m_2"] & \dots & O_n \arrow[d, "m_n"] \\
D_1 \arrow[r, "r_{1,2}"] \arrow[rr, bend right, "r_{1,3}"] & D_2 \arrow[r, "r_{2,3}"] & \dots & D_n \arrow[lll, bend left, "r_{n,1}"]
\end{tikzcd}$$

such that

- r_{ij} is functional and injective,
- r_{ii} is the identity (diagonal) relation,
- r_{ji} is the converse of r_{ij}, and
- r_{ik} is the relational composition of r_{ij} and r_{jk}
- model for a diagram: family (m_i) of models with coherent relations (r_{ij}) such that $(m_i, r_{ji}m_j)$ is a model for A_{ij}

Contextualized semantics of diagrams, revisited

$$\begin{tikzcd}
\tilde{S} & & \overline{B} & & \tilde{T} \\
& O_1 \arrow[ul, "\iota_1"'] \arrow[ur, "\sigma_1"] & & O_2 \arrow[ul, "\sigma_2"'] \arrow[ur, "\iota_2"] &
\end{tikzcd}$$

where $\overline{B}$ modifies B as follows:

- r_{ij} are added to $\overline{B}$ as roles with domain $\top_S$ and range $\top_T$
- the correspondences are translated to axioms involving these roles:
  - $s_i = t_j$ becomes $s_i \ r_{ij} \ t_j$
  - $a_i \in c_j$ becomes $a_i \in \exists r_{ij}.c_j$
  - $\ldots$
- the properties of the roles are added as axioms in $\overline{B}$

Adding domain relations to the bridge

ontology $\overline{B}$ =
Class: *ThingS*
Class: *ThingT*
ObjectPropery: r_{ST} **Domain:** *ThingS* **Range:** *ThingT*
Class: *Person* **EquivalentTo:** r_{ST} **some** *HumanBeing*
Class: *Employee*
Class: *Child* **SubClassOf:** r_{ST} **some** $\neg$ *Employee*
Individual: *alex* **Types:** r_{ST} **some** *Male*

Example: contextualized semantics

$$\begin{tikzcd}
 & & C & & \\
S \arrow[rru, dotted] & & \overline{B} \arrow[u, dotted] & & T \arrow[llu, dotted] \\
 & \{Person, alex, Child, \top_s\} \arrow[lu, "\iota_1"] \arrow[ru, "\sigma_1"'] \arrow[ruu, dotted] & & \{Male, HumanBeing, Employee, \top_T\} \arrow[lu, "\sigma_2"] \arrow[ru, "\iota_2"'] \arrow[luu, dotted] &
\end{tikzcd}$$

where

ontology C =

Class: *ThingS*
Class: *ThingT*
ObjectPropery: r_{ST} **Domain:** *ThingS* **Range:** *ThingT*
Class: *Person* **EquivalentTo:** r_{ST} **some** *HumanBeing*
Class: *Employee*
Class: *Child* **SubClassOf:** r_{ST} **some** ¬ *Employee*
Individual: *alex* **Types:** r_{ST} **some** *Male*, *Person*

```
QueryRelatedDefn ::= QueryDefn | SubstDefn | ResultDefn
QueryDefn       ::= 'query' QueryName '=' 'select' Vars 'where' Sentence
                    OMS ['along' Translation]
SubstDefn       ::= 'substitution' SubstName ':' OMS 'to' OMS '=' SymbolMap
ResultDefn      ::= 'result' ResultName SubstName ( ',' SubstName )*
                    QueryName ['%complete']
QueryName       ::= IRI
SubstName       ::= IRI
ResultName      ::= IRI
Vars            ::= Symbol ( ',' Symbol )*
```

Queries

DOL is a logical (meta) language

- focus on ontologies, models, specifications,
- and their logical relations: logical consequence, interpretations, . . .

Queries are different:

- answer is not “yes” or “no”, but an answer substitution
- query language may differ from language of OMS that is queried

Sample query languages

- conjunctive queries in OWL
- Prolog/Logic Programming
- SPARQL

Syntax of queries in DOL

New OMS declarations and relations:

```
query qname = select vars where sentence in OMS
               [along language-translation]
substitution sname : OMS1 to OMS2 = derived-symbol-map
result rname = sname_1, ..., sname_n for qname
             %% result is a substitution
```

New sentences (however, as structured OMS!):

```
apply(sname,sentence)     %% apply substition
```

Open question: how to deal with “construct” queries?

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 OMS?
- 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 OMS 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) OMS

- 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

```
EquivKeyword       ::= 'equivalence'
EquivName          ::= IRI
EquivType          ::= GroupOMS '<\lessthan>-<\greaterthan>' GroupOMS
```

Equivalences

- **equivalence** $Id : O_1 \leftrightarrow O_2 = O_3$
- (fragment) OMS 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
```

```
ModuleRelDefn      ::= 'module' ModuleName [Conservative] ':'
                          'for' InterfaceSignature
ModuleName         ::= IRI
ModuleType         ::= OMS 'of' OMS
```

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.

Conclusion

- DOL is a meta language for (formal) ontologies, specifications and models (OMS)
- DOL covers many aspects of modularity of and relations among OMS ("OMS-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`

Related work

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