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

Mihai Codescu[2] Oliver Kutz[2]
Christoph Lange[3] Till Mossakowski[1] Fabian Neuhaus[1]

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG
INF FAKULTÄT FÜR INFORMATIK

[1]University of Magdeburg, Germany
[2]Free University of Bolzano, Italy
[3]University of Bonn, Germany

FroCoS/TABLEAUX tutorial, Wrocław, 2015-09-21

Resources

Resources

- http://ontoiop.org Initial standardization initiative
- https://github.com/tillmo/DOL repository for development of the DOL standard
- http://www.omg.org/spec/DOL future place for DOL standard
- http://www.omg.org/techprocess/meetings/schedule/OntoIOp_RFP.html process at OMG (for members only)
- http://hets.eu Tool Hets
- http://ontohub.org Ontohub platform

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!

Ontologies

```
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: Father
    EquivalentTo: Man and hasChild some Person
Class: Parent
    EquivalentTo: Father or Mother
```

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.

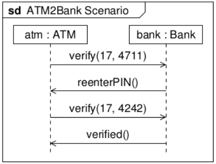
OWL gives better tool support, FOL greater expressiveness.

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

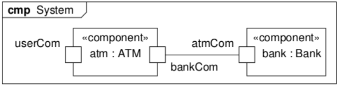
[1] C.M. Keet, F.C. Fernández-Reyes, and A. Morales-González. Representing mereotopological relations in owl ontologies with ontoparts. In *Proceedings of the 9th Extended Semantic Web Conference (ESWC'12), 29-31 May 2012, Heraklion, Crete, Greece*, volume 7295 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 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.

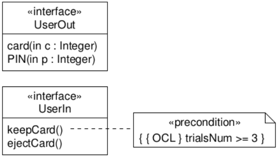
Models: Is a family of UML models consistent?



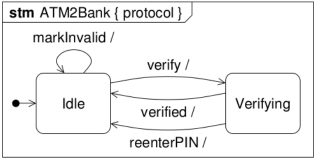
(a) Interaction



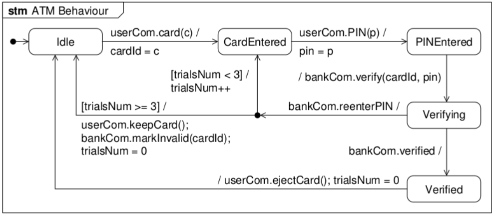
(b) Composite structure



(c) Interfaces

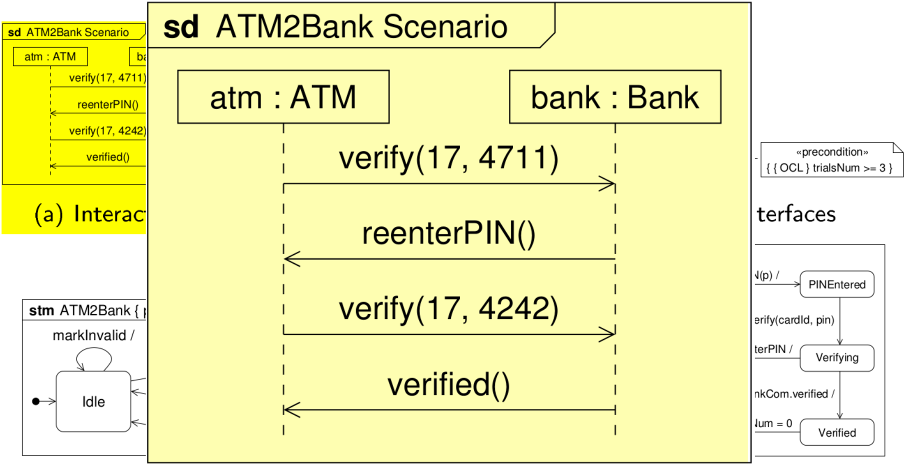


(d) Protocol state machine



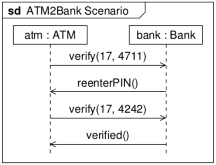
(e) State machine

Models: Is a family of UML models consistent?

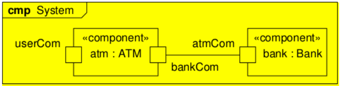


(d) Protocol state machine (e) State machine

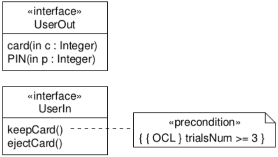
Models: Is a family of UML models consistent?



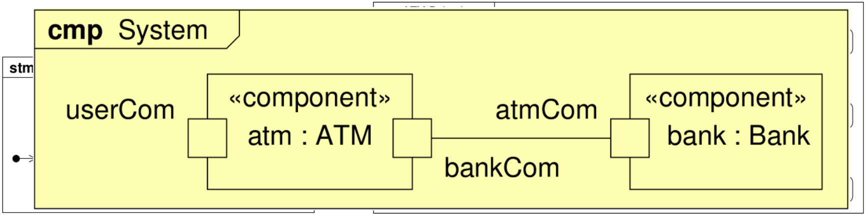
(a) Interaction



(b) Composite structure



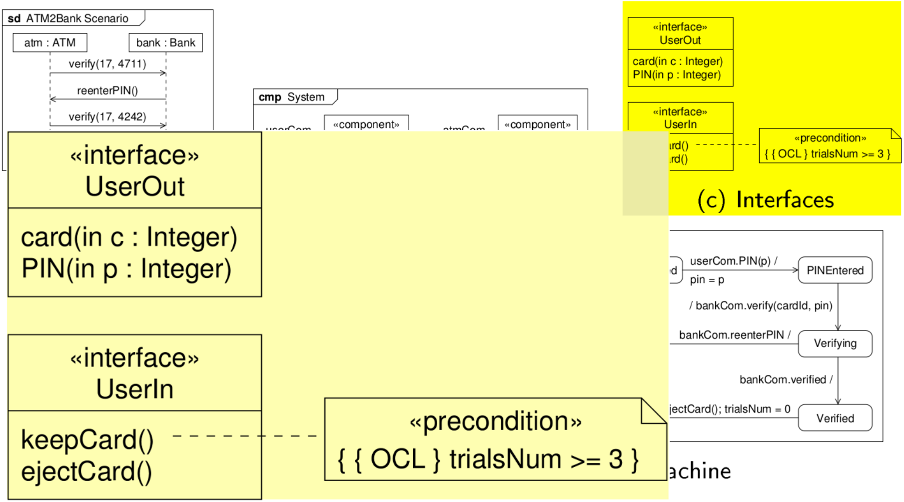
(c) Interfaces



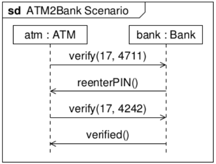
(d) Protocol state machine

(e) State machine

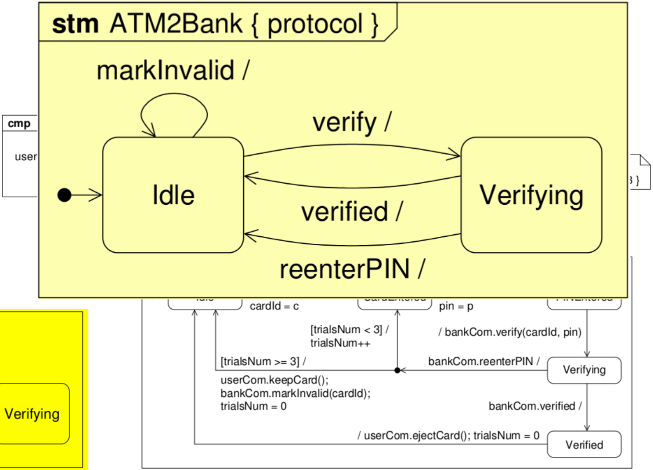
Models: Is a family of UML models consistent?

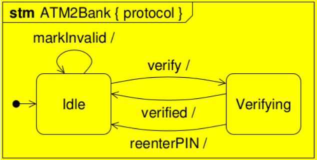


Models: Is a family of UML models consistent?



(a) Interaction

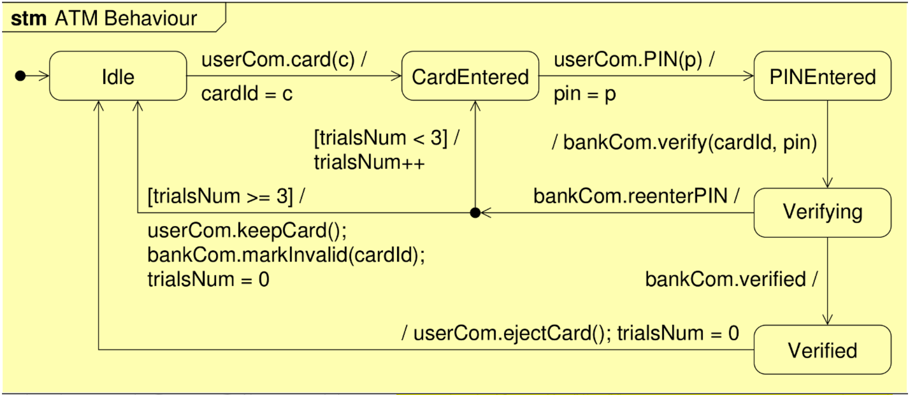


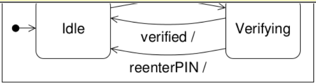


(d) Protocol state machine

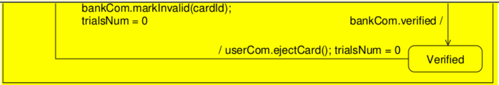
(e) State machine

Models: Is a family of UML models consistent?





(d) Protocol state machine



(e) State machine

Specification of sorting in CASL/FOL

```
sort Elem
free type List[Elem] ::= [] | __::__(Elem; List[Elem])
pred __<=__ : Elem * Elem
pred __elem__ : Elem * List[Elem}
preds is_ordered : List[Elem];
      permutation : List[Elem] * List[Elem]
op sorter : List[Elem]->List[Elem]
forall x,y:Elem; L,L1,L2:List[Elem]
. not x elem []
. x elem (y :: L) <=>  x=y \/ x elem L
. is_ordered([])        . is_ordered(x::[])
. is_ordered(x::y::L) <=> x<=y /\ is_ordered(y::L)
. permutation(L1,L2) <=>
        (forall x:Elem . x elem L1 <=> x elem L2)
. is_ordered(sorter(L))
. permutation(L,sorter(L))
```

Specification of insert sort in CASL/FOL

```
sort Elem
free type List[Elem] ::= [] | __::__(Elem; List[Elem])
ops insert : Elem*List[Elem] -> List[Elem];
    insert_sort : List[Elem]->List[Elem]
forall x,y:Elem; L:List[Elem]
. insert(x,[]) = x::[]
. x<=y => insert(x,y::L) = x::insert(y,L)
. not x<=y => insert(x,y::L) = y::insert(x,L)
. insert_sort([]) = []
. insert_sort(x::L) = insert(x,insert_sort(L))
```

Is insert sort correct w.r.t. the sorting specification?

What have ontologies, models and specifications in common?

- formalised in some logical system
- signature with non-logical symbols (domain vocabulary)
- 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

We henceforth call them "OMS"!

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

OntolOp

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

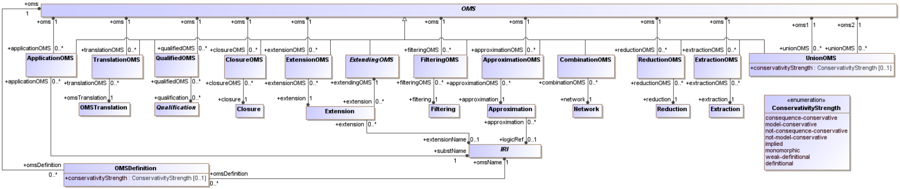
Overview of DOL

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

   only OMS with flattenable components are flattenable
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 module relations

OMS

Abstract syntax of OMS



Concrete syntax of OMS

```
BasicOMS             ::= <language and serialization specific>
ClosableOMS          ::= BasicOMS | OMSRef [ImportName]
ImportName           ::= '%(' IRI ')%'
OMSRef               ::= IRI
ExtendingOMS         ::= ClosableOMS | RelativeClosureOMS
RelativeClosureOMS   ::= ClosureType '{' ClosableOMS '}'
OMS                  ::= ExtendingOMS
                       | OMS Closure
                       | OMS OMSTranslation
                       | OMS Reduction
                       | OMS Approximation
                       | OMS Filtering
                       | OMS 'and' [ConservativityStrength] OMS
                       | OMS 'then' ExtensionOMS
                       | Qualification* ':' GroupOMS
                       | 'combine' NetworkElements [ExcludeExtensions]
                       | GroupOMS
GroupOMS             ::= '{' OMS '}' | OMSRef
```

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))))
```

Syntax of extensions

```
BasicOMS          ::= <language and serialization specific>
OMS               ::= ...
                    | OMS 'then' BasicOMS
                    | ...
```

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
```

Full syntax of extensions

```
BasicOMS              ::= <language and serialization specific>
ClosableOMS           ::= BasicOMS | OMSRef [ImportName]
ExtendingOMS          ::= ClosableOMS | RelativeClosureOMS
OMS                   ::= ...
                        | OMS 'then' ExtensionOMS
                        | ...
ExtensionOMS          ::= [ExtConservativityStrength] [ExtensionName] ExtendingOMS
ExtensionName         ::= '%(' IRI ')%'
ExtConservativityStrength ::= '%ccons' | '%mcons'
                        | '%notccons' | '%notmcons'
                        | '%mono' | '%wdef' | '%def'
                        | '%implied'
```

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
```

Syntax of unions

```
OMS ::= ...
      | OMS 'and' [ConservativityStrength] OMS
      | ...
```

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
```

Syntax of translations

```
OMS               ::= ...
                    | OMS OMSTranslation
                    | ...
OMSTranslation    ::= 'with' SymbolMap
SymbolMap         ::= SymbolMapItem ( ',' SymbolMapItem )*
SymbolMapItem     ::= Symbol '|->' Symbol
Symbol            ::= IRI
```

Translations

- *O* **with** σ, where σ is a symbol map (signature morphism)

```
BankOntology with Bank |-> FinancialBank
and
RiverOntology with Bank |-> RiverBank
 %% necessary disambiguation when uniting ontologies
```

Full syntax of translations

```
OMS                   ::= ...
                        | OMS OMSTranslation
                        | ...
OMSTranslation        ::= 'with' LanguageTranslation* SymbolMap
LanguageTranslation ::= 'translation' OMSLanguageTranslation
SymbolMap             ::= GeneralSymbolMapItem ( ',' GeneralSymbolMapItem )*
GeneralSymbolMapItem ::= Symbol | SymbolMapItem
SymbolMapItem         ::= Symbol '|->' Symbol
Symbol                ::= IRI
LanguageTranslation ::= 'translation' OMSLanguageTranslation
OMSLanguageTranslation ::= OMSLanguageTranslationRef | '->' LoLaRef
OMSLanguageTranslationRef ::= IRI
LoLaRef               ::= LanguageRef | LogicRef
LanguageRef           ::= IRI
LogicRef              ::= IRI
```

Translations

- *O* **with** σ, where σ is a signature morphism
- *O* **with translation** ρ, where ρ is a logic translation

```
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 – Select

	hide/reveal	remove/extract	forget/keep	select/reject
semantic background	model reduct	conservative extension	uniform interpolation	theory filtering
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	$= \Sigma$	$\geq \Sigma$	$= \Sigma$	$\geq \Sigma$
change of logic	possible	not possible	possible	not possible
application	specification	ontologies	ontologies	blending

Syntax of reduction

```
OMS                 ::= ...
                      | OMS Reduction
                      | ...
Reduction           ::= 'hide' LogicReduction* SymbolList
                      | 'reveal' SymbolList
LogicReduction      ::= 'along' OMSLanguageTranslation
SymbolList          ::= Symbol ( ',' Symbol )*
```

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 a logic projection

Reduction: example

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

Syntax of module extraction

```
OMS                   ::= ...
                        | OMS Extraction
                        | ...
Extraction            ::= 'extract' InterfaceSignature
                        | 'remove' InterfaceSignature
InterfaceSignature ::= SymbolList
SymbolList            ::= Symbol ( ',' Symbol )*
```

Module Extraction: remove/extract

O **extract** Σ

- Σ: interface signature (subsignature of that of O)
- O **extract** Σ is the minimal depleting Σ-module of O
- Note: O is a Σ-conservative extension of O **extract** Σ.
- Dually: O **remove** Σ (here, Σ specifies the symbols that are not in the interface signature)

Module Extraction: example

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

Here, adding `inv` is conservative.

Syntax of approximation

```
OMS                   ::= ...
                        | OMS Approximation
                        | ...
Approximation         ::= 'forget' InterfaceSignature ['keep' LogicRef]
                        | 'keep' InterfaceSignature ['keep' LogicRef]
                        | 'keep' LogicRef
InterfaceSignature ::= SymbolItems
LogicRef              ::= IRI
```

Approximaation: forget/keep

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

Interpolation: example

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

Computing interpolants can be hard, even undecidable.

Syntax of filtering

```
OMS               ::= ...
                    | OMS Filtering
                    | ...
Filtering         ::= 'select' SymbolList
                    | 'select' BasicOMS
                    | 'reject' SymbolList
                    | 'reject' BasicOMS
```

Filtering

- O **select** T, where T is a subtheory (fragment) of that of O
  - intuition: axioms involving only symbols in $Sig(T)$ are kept
  - moreover, all axioms contained in T are kept as well
- O **reject** 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

Filtering: example

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

The semantics is the following theory:

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

Hide – Extract – Forget – Select

	hide/reveal	remove/extract	forget/keep	select/reject
semantic background	model reduct	conservative extension	uniform interpolation	theory filtering
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	$= \Sigma$	$\geq \Sigma$	$= \Sigma$	$\geq \Sigma$
change of logic	possible	not possible	possible	not possible
application	specification	ontologies	ontologies	blending

Reduction: specification example

```
spec List =  sort Elem
  free type List[Elem] ::= [] | __::__(Elem; List[Elem])
  pred __elem__ : Elem * List[Elem}
  forall x,y:Elem; L,L1,L2:List[Elem]
  . not x elem []  . x elem (y :: L) <=>  x=y \/ x elem L
spec Sorting = List then
  preds is_ordered : List[Elem];
        permutation : List[Elem] * List[Elem]
  op sorter : List[Elem]->List[Elem]
  forall x,y:Elem; L,L1,L2:List[Elem]
  . is_ordered([])        . is_ordered(x::[])
  . is_ordered(x::y::L) <=> x<=y /\ is_ordered(y::L)
  . permutation(L1,L2) <=>
            (forall x:Elem . x elem L1 <=> x elem L2)
  . is_ordered(sorter(L)) . permutation(L,sorter(L))
hide permutation, is_ordered
```

Relations among the different notions

$$
\begin{array}{ll}
 & Mod(O\ \textbf{hide}\ \Sigma) \\
= & Mod(O\ \textbf{remove}\ \Sigma)|_{Sig(O)\setminus\Sigma} \\
\subseteq & Mod(O\ \textbf{forget}\ \Sigma) \\
\subseteq & Mod(O\ \textbf{reject}\ \Sigma)
\end{array}
$$

Pros and Cons

	hide/reveal	remove/extract	forget/keep	select/reject
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

Syntax of closure

```
ClosableOMS         ::= BasicOMS | OMSRef [ImportName]
ExtendingOMS        ::= ClosableOMS | RelativeClosureOMS
RelativeClosureOMS ::= ClosureType '{' ClosableOMS '}'
OMS                 ::= ...
                      | OMS Closure
                      | ...
Closure             ::= ClosureType CircClosure [CircVars]
ClosureType         ::= 'minimize'
                      | 'closed-world'
                      | 'maximize'
                      | 'free'
                      | 'cofree'
CircClosure         ::= Symbol+
CircVars            ::= 'vars' 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
```

Maximizations

- O_1 **then maximize {** O_2 **}**
- forces maximal interpretation of non-logical symbols in O_2

```
  Class: Block
  Individual: B1 Types: Block
  Individual: B2 Types: Block DifferentFrom: B1
then maximize {
        Class: Normal
        Individual: B2 Types: Normal }
then
  Class: Ontable SubClassOf: Block and Normal
then %implied
  Individual: B1 Types: not 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
         }
```

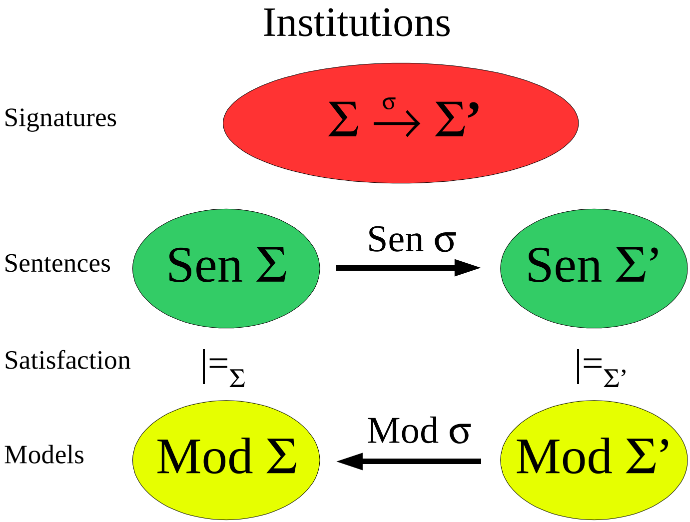
Cofreeness

- 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
         }
```

Semantics of OMS

Institutions (intuition)



Institutions (formal definition)

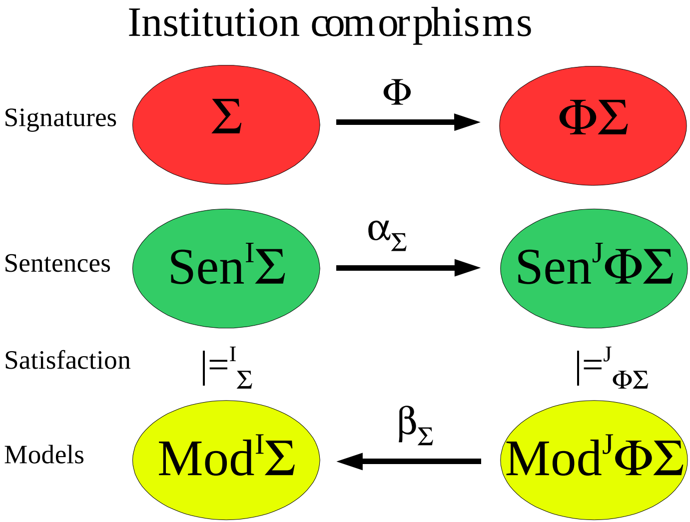
An institution $\mathcal{I} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ consists of:

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

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

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

Institution comorphisms (embeddings, encodings)



Institution comorphisms (embeddings, encodings)

Definition

Let $\mathcal{I} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ and $\mathcal{I}' = \langle \mathbf{Sign}', \mathbf{Sen}', \mathbf{Mod}', \langle \models'_{\Sigma'} \rangle_{\Sigma' \in |\mathbf{Sign}'|} \rangle$ be institutions. An *institution comorphism* $\rho \colon \mathcal{I} \to \mathcal{I}'$ consists of:

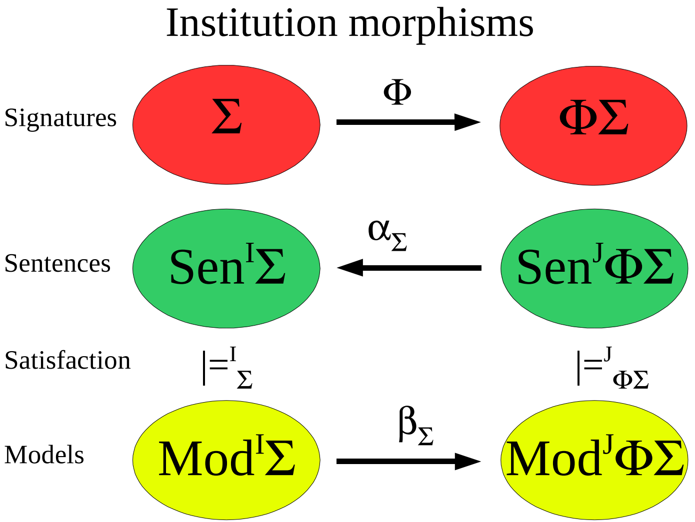
- a functor $\Phi \colon \mathbf{Sign} \to \mathbf{Sign}'$;
- a natural transformation $\alpha \colon \mathbf{Sen} \to \Phi \,; \mathbf{Sen}'$, and
- a natural transformation $\beta \colon (\Phi)^{op} \,; \mathbf{Mod}' \to \mathbf{Mod}$,

such that for any $\Sigma \in |\mathbf{Sign}|$, any $\varphi \in \mathbf{Sen}(\Sigma)$ and any $M' \in \mathbf{Mod}'(\Phi(\Sigma))$:

$$M' \models'_{\Phi(\Sigma)} \alpha_\Sigma(\varphi) \iff \beta_\Sigma(M') \models_\Sigma \varphi$$

$[Satisfaction\ condition]$

Institution morphisms (projections)



Institution morphisms (projections)

Definition

Let $\mathcal{I} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ and $\mathcal{I}' = \langle \mathbf{Sign}', \mathbf{Sen}', \mathbf{Mod}', \langle \models'_{\Sigma'} \rangle_{\Sigma' \in |\mathbf{Sign}'|} \rangle$ be institutions. An *institution morphism* $\mu \colon \mathcal{I} \to \mathcal{I}'$ consists of:

- a functor $\mu^{Sign} \colon \mathbf{Sign} \to \mathbf{Sign}'$;
- a natural transformation $\mu^{Sen} \colon \mu^{Sign} ; \mathbf{Sen}' \to \mathbf{Sen}$, and
- a natural transformation $\mu^{Mod} \colon \mathbf{Mod} \to (\mu^{Sign})^{op} ; \mathbf{Mod}'$,

such that for any signature $\Sigma \in |\mathbf{Sign}|$, any $\varphi' \in \mathbf{Sen}'(\mu^{Sign}(\Sigma))$ and any $M \in \mathbf{Mod}(\Sigma)$:

$$M \models_\Sigma \mu^{Sen}_\Sigma(\varphi') \iff \mu^{Mod}_\Sigma(M) \models'_{\mu^{Sign}(\Sigma)} \varphi'$$

$$[Satisfaction\ condition]$$

Unions, differences and inclusive institutions

We assume that for each institution, there exists (possibly partial) union and difference operations on signatures. E.g. an inclusion system on signatures would be a good framework where this can be required.

Definition (adopted from Goguen, Roșu)

An *weakly inclusive category* is a category having a broad subcategory which is a partially ordered class.
An *weakly inclusive institution* is one with an inclusive signature category such that the sentence functor preserves inclusions.

We also assume that model categories are weakly inclusive.

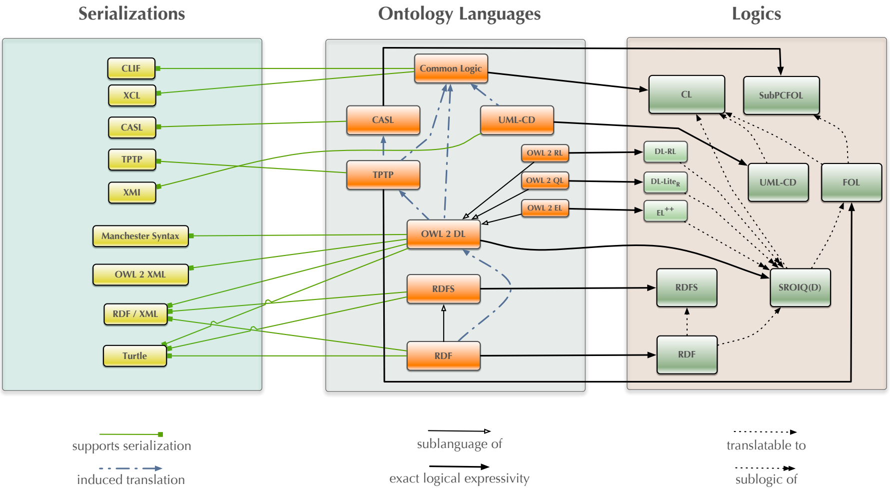
$M|_{\Sigma}$ means $M|_{\iota}$ where $\iota : \Sigma \to Sig(M)$ is the inclusion.

Heterogeneous logical environments

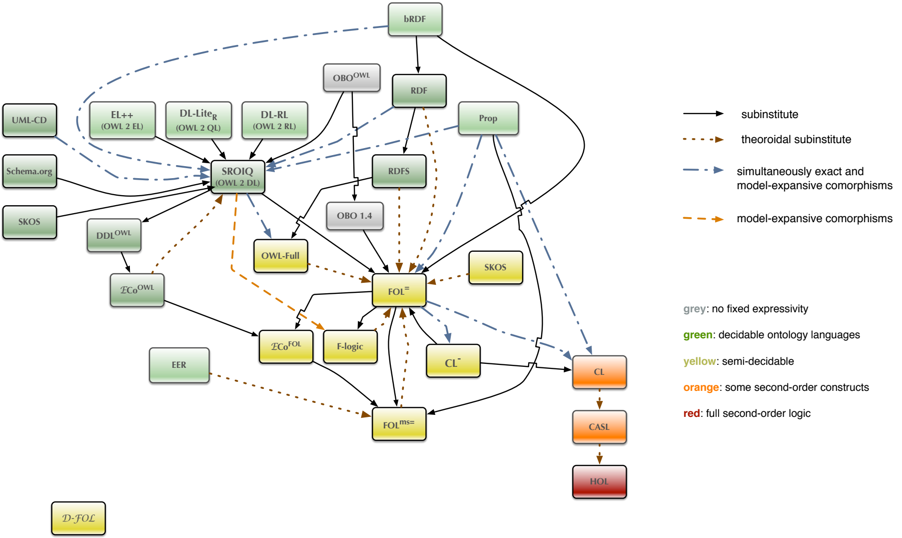
A heterogeneous logical environment consists of

- a logic graph, consisting of institutions, institution comporphisms (translations) and institution morphisms (projections),
- an OMS language graph, and
- supports relations.

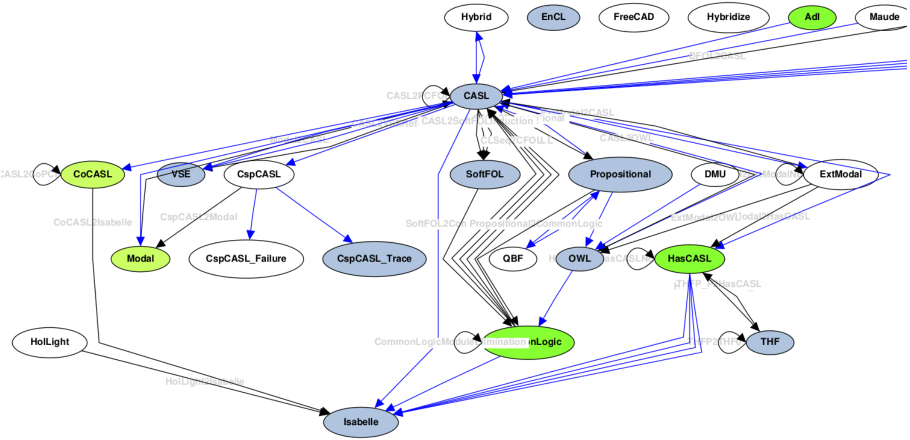
The support relations specify which language supports which logics and which serializations, and which language translation supports which logic translation or reduction.
Moreover, each language has a default logic and a default serialization. There are also default translations.


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

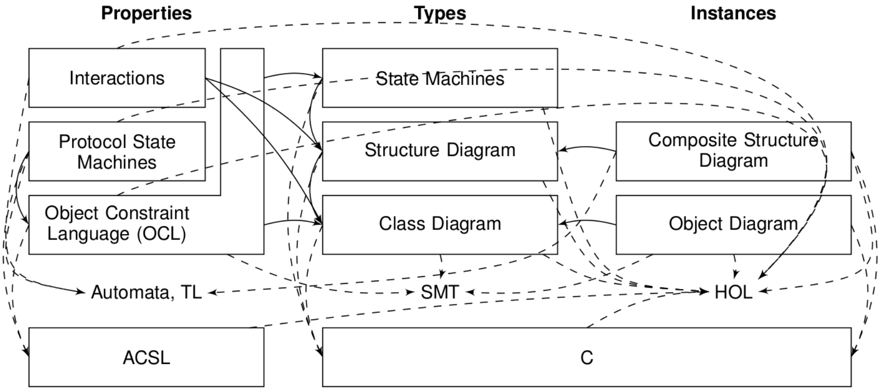
Ontologies: An Initial Logic Graph



Specifications: An Initial Logic Graph



UML models: An Initial Logic Graph



Semantic domains of DOL

- semantics of a flattenable OMS has form (I, Σ, Ψ) (theory-level)
- semantics of an elusive OMS has form $(I, \Sigma, \mathcal{M})$ (model-level)
  - institution I
  - signature Σ in I
  - set Ψ of Σ-sentences
  - class $\mathcal{M}$ of Σ-models

  We can obtain the model-level semantics from the theory-level semantics by taking $\mathcal{M} = \{M \in \mathbf{Mod}(\Sigma) \mid M \models \Psi\}$.
- semantics of an OMS declaration/relation has form $\Gamma\colon \mathit{IRI} \longrightarrow (\mathit{OMS} \uplus \mathit{OMS} \times \mathit{OMS} \times \mathit{SigMor})$
  - OMS is the class of all triples (I, Σ, Ψ), $(I, \Sigma, \mathcal{M})$
  - for interpretations etc., domain, codomain and signature morphism is recorded: $\mathit{OMS} \times \mathit{OMS} \times \mathit{SigMor}$

Semantics of basic OMS

We assume that $[\![O]\!]_{basic} = (I, \Sigma, \Psi)$ for some OMS language based on I. The semantics consists of

- the institution I
- a signature Σ in I
- a set Ψ of Σ-sentences

This direct leads to a theory-level semantics for the OMS:

$$[\![O]\!]^T_\Gamma = [\![O]\!]_{basic}$$

Generally, if a theory-level semantics is given: $[\![O]\!]^T_\Gamma = (I, \Sigma, \Psi)$, this leads to a model-level semantics as well:

$$[\![O]\!]^M_\Gamma = (I, \Sigma, \{M \in Mod(\Sigma) \mid M \models \Psi\})$$

Semantics of extensions

O_1 flattenable $[\![O_1 \textbf{ then } O_2]\!]^T_\Gamma = (I, \Sigma_1 \cup \Sigma_2, \Psi_1 \cup \Psi_2)$

where

- $[\![O_1]\!]^T_\Gamma = (I, \Sigma_1, \Psi_1)$
- $[\![O_2]\!]_{basic} = (I, \Sigma_2, \Psi_2)$

O_1 elusive $[\![O_1 \textbf{ then } O_2]\!]^M_\Gamma = (I, \Sigma_1 \cup \Sigma_2, \mathcal{M}')$

where

- $[\![O_1]\!]^M_\Gamma = (I, \Sigma_1, \mathcal{M}_1)$
- $[\![O_2]\!]_{basic} = (I, \Sigma_2, \Psi_2)$
- $\mathcal{M}' = \{M \in \textbf{Mod}(\Sigma_1 \cup \Sigma_2) \mid M \models \Psi_2, M|_{\Sigma_1} \in \mathcal{M}_1\}$

Semantics of extensions (cont'd)

%mcons (%def, %mono) leads to the additional requirement that

each model in $\mathcal{M}_1$ has a (unique, unique up to isomorphism) $\Sigma_1 \cup \Sigma_2$-expansion to a model in $\mathcal{M}'$.

%implies leads to the additional requirements that

$\Sigma_2 \subseteq \Sigma_1$ *and* $\mathcal{M}' = \mathcal{M}_1$.

%ccons leads to the additional requirement that

$\mathcal{M}' \models \varphi$ *implies* $\mathcal{M}_1 \models \varphi$ *for any* Σ_1*-sentence* φ.

Theorem

%mcons implies %ccons, but not vice versa.

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: $[\![iri]\!]_\Gamma = \Gamma(iri)$

Semantics of unions

O_1, O_2 flattenable $[\![O_1 \textbf{ and } O_2]\!]^T_\Gamma = (I, \Sigma_1 \cup \Sigma_2, \Psi_1 \cup \Psi_2)$, where

- $[\![O_i]\!]^T_\Gamma = (I, \Sigma_i, \Psi_i)$ $(i = 1, 2)$

one of O_1, O_2 elusive $[\![O_1 \textbf{ and } O_2]\!]^M_\Gamma = (I, \Sigma_1 \cup \Sigma_2, \mathcal{M})$, where

- $[\![O_i]\!]^M_\Gamma = (I, \Sigma_i, \mathcal{M}_i)$ $(i = 1, 2)$
- $\mathcal{M} = \{M \in \textbf{Mod}(\Sigma_1 \cup \Sigma_2) \mid M|_{\Sigma_i} \in \mathcal{M}_i, i = 1, 2\}$

Semantics of translations

O flattenable Let $[\![O]\!]^T_\Gamma = (I, \Sigma, \Psi)$

- homogeneous translation
  $[\![O \textbf{ with } \sigma : \Sigma \to \Sigma']\!]^T_\Gamma = (I, \Sigma', \sigma(\Psi))$
- heterogeneous translation
  $[\![O \textbf{ with translation } \rho : I \to I']\!]^T_\Gamma = (I', \rho^{Sig}(\Sigma), \rho^{Sen}(\Psi))$

O elusive Let $[\![O]\!]^M_\Gamma = (I, \Sigma, \mathcal{M})$

- homogeneous translation
  $[\![O \textbf{ with } \sigma : \Sigma \to \Sigma']\!]^M_\Gamma = (I, \Sigma', \mathcal{M}')$
  where $\mathcal{M}' = \{M \in \textbf{Mod}(\Sigma') \mid M|_\sigma \in \mathcal{M}\}$
- heterogeneous translation
  $[\![O \textbf{ with translation } \rho : I \to I']\!]^M_\Gamma = (I', \rho^{Sig}(\Sigma), \mathcal{M}')$ where
  $\mathcal{M}' = \{M \in \textbf{Mod}^{I'}(\rho^{Sig}(\Sigma)) \mid \rho^{Mod}(M) \in \mathcal{M}\}$

Hide – Extract – Forget – Select

	hide/reveal	remove/extract	forget/keep	select/reject
semantic background	model reduct	conservative extension	uniform interpolation	theory filtering
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	$= \Sigma$	$\geq \Sigma$	$= \Sigma$	$\geq \Sigma$
change of logic	possible	not possible	possible	not possible
application	specification	ontologies	ontologies	blending

Semantics of reductions

Let $[\![O]\!]_\Gamma^M = (I, \Sigma, \mathcal{M})$

- homogeneous reduction
  $[\![O \textbf{ reveal } \Sigma'\,]\!]_\Gamma^M = (I, \Sigma', \mathcal{M}|_{\Sigma'})$
  $[\![O \textbf{ hide } \Sigma'\,]\!]_\Gamma^M = [\![O \textbf{ reveal } \Sigma \setminus \Sigma']\!]_\Gamma^M$
- heterogeneous reduction
  $[\![O \textbf{ hide along } \rho : I \to I']\!]_\Gamma^M = (I', \rho^{Sig}(\Sigma), \rho^{Mod}(\mathcal{M}))$

$\mathcal{M}|_{\Sigma'}$ may be impossible to capture by a theory (even if $\mathcal{M}$ is).
The proof calculus for refinements involving reduction needs invention of some OMS O'':

$$\frac{O \leadsto O''}{O \textbf{ hide } \Sigma \leadsto O'} \quad \text{if } \iota\colon O' \longrightarrow O'' \text{ is a conservative extension}$$

where $\iota : \Sigma \to Sig(O)$ is the inclusion

Modules

Definition

$O' \subseteq O$ is a Σ-module of (flat) O iff O is a model-theoretic Σ-conservative extension of O', i.e. for every model M of O', $M|_\Sigma$ can be expanded to an O-model.

Depleting modules

Definition

Let O_1 and O_2 be two OMS and $\Sigma \subseteq Sig(O_i)$.
Then O_1 and O_2 are Σ-inseparable ($O_1 \equiv_\Sigma O_2$) iff

$$Mod(O_1)|_\Sigma = Mod(O_2)|_\Sigma$$

Definition

$O' \subseteq O$ is a depleting Σ-module of (flat) O iff $O \setminus O' \equiv_{\Sigma \cup Sig(O')} \emptyset$.

Theorem

1. *Depleting Σ-modules are Σ-conservative.*
2. *The minimum depleting Σ-module always exists.*

Semantics of module extraction (remove/extract)

Note: O must be flattenable!

Let $[\![O]\!]^T_\Gamma = (I, \Sigma, \Psi)$.
$[\![O \textbf{ extract } \Sigma_1]\!]^T_\Gamma = (I, \Sigma_2, \Psi_2)$
where $(\Sigma_2, \Psi_2) \subseteq (\Sigma, \Psi)$ is the minimum depleting Σ_1-module of (Σ, Ψ)

$[\![O \textbf{ remove } \Sigma_1]\!]^T_\Gamma = [\![O \textbf{ extract } \Sigma \setminus \Sigma_1]\!]^T_\Gamma$

Tools can extract any module (i.e. using locality). Any two modules will have the same Σ-consequences.

Semantics of interpolation (forget/keep)

Note: O must be flattenable!
Let $[\![O]\!]_\Gamma^T = (I, \Sigma, \Psi)$.

- homogeneous interpolation
  $[\![O \textbf{ keep in } \Sigma']\!]_\Gamma^T = (I, \Sigma', \{\varphi \in \textbf{Sen}(\Sigma') \mid \Psi \models \varphi\})$
  (note: any logically equivalent theory will also do)
  $[\![O \textbf{ forget } \Sigma']\!]_\Gamma^T = [\![O \textbf{ keep in } \Sigma \setminus \Sigma']\!]_\Gamma^T$
- heterogeneous interpolation
  $[\![O \textbf{ keep in } \Sigma' \textbf{ with } I']\!]_\Gamma^T =$
  $(I', \Sigma', \{\varphi \in \textbf{Sen}^{I'}(\Sigma') \mid \Psi \models \rho^{Sen}(\varphi)\})$
  where $\rho : I' \to I$ is the inclusion
  and Σ' is such that $\rho^{Sig}(\Sigma') \subseteq \Sigma$
  $[\![O \textbf{ forget } \Sigma' \textbf{ with } I']\!]_\Gamma^T = [\![O \textbf{ keep in } \Sigma \setminus \Sigma' \textbf{ with } I']\!]_\Gamma^T$

Semantics of select/reject

Note: O must be flattenable!

Let $[\![O]\!]^T_\Gamma = (I, \Sigma, \Psi)$.
$[\![O \textbf{ select } (\Sigma', \Phi)]\!]^T_\Gamma = (I, \Sigma, Sen(\iota)^{-1}(\Psi) \cup \Phi)$
where $\iota : \Sigma' \to \Sigma$ is the inclusion
$[\![O \textbf{ reject } (\Sigma', \Phi)]\!]^T_\Gamma = (I, \Sigma \setminus \Sigma', Sen(\iota)^{-1}(\Psi) \setminus \Phi)$
where $\iota : \Sigma \setminus \Sigma' \to \Sigma$ is the inclusion

Hide – Extract – Forget – Select

	hide/reveal	remove/extract	forget/keep	select/reject
semantic background	model reduct	conservative extension	uniform interpolation	theory filtering
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	$= \Sigma$	$\geq \Sigma$	$= \Sigma$	$\geq \Sigma$
change of logic	possible	not possible	possible	not possible
application	specification	ontologies	ontologies	blending

Semantics of minimizations

Let $[\![O_1]\!]^M_\Gamma = (I, \Sigma_1, \mathcal{M}_1)$
Let $[\![O_1 \textbf{ then } O_2]\!]^M_\Gamma = (I, \Sigma_2, \mathcal{M}_2)$
Then

$$[\![O_1 \textbf{ then minimize } O_2]\!]^M_\Gamma = (I, \Sigma_2, \mathcal{M})$$

where
$\mathcal{M} = \{M \in \mathcal{M}_2 \mid M \text{ is minimal in } \{M' \in \mathcal{M}_2 \mid M'|_{\Sigma_1} = M|_{\Sigma_1}\}\}$

Dually: maximization.

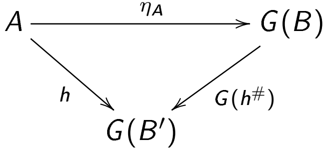
Semantics of freeness

Let $[\![O_1]\!]^M_\Gamma = (I, \Sigma_1, \mathcal{M}_1)$
Let $[\![O_1 \textbf{ then } O_2]\!]^M_\Gamma = (I, \Sigma_2, \mathcal{M}_2)$
Let $\iota : \Sigma_1 \to \Sigma_2$ be the inclusion
Then

$$[\![O_1 \textbf{ then free } O_2]\!]^M_\Gamma = (I, \Sigma_2, \mathcal{M})$$

where $\mathcal{M} = \{M \in \mathcal{M}_2 \mid M \text{ is } Mod(\iota)\text{-free over } M|_\iota \text{ with unit } id\}$

Given a functor $G\colon \mathbf{B} \longrightarrow \mathbf{A}$, an object $B \in \mathbf{B}$ is called *G-free (with unit* $\eta_A\colon A \longrightarrow G(B)$*) over* $A \in \mathbf{A}$, if for any object $B' \in \mathbf{B}$ and any morphism $h\colon A \longrightarrow G(B')$, there is a unique morphism $h^{\#}\colon B \longrightarrow B'$ such that $\eta_A; G(h^{\#}) = h$.



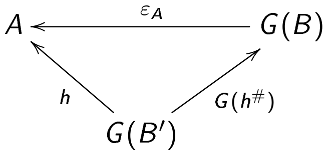
Semantics of cofreeness

Let $[\![O_1]\!]^M_\Gamma = (I, \Sigma_1, \mathcal{M}_1)$
Let $[\![O_1 \textbf{ then } O_2]\!]^M_\Gamma = (I, \Sigma_2, \mathcal{M}_2)$
Let $\iota : \Sigma_1 \to \Sigma_2$ be the inclusion
Then

$$[\![O_1 \textbf{ then cofree } O_2]\!]^M_\Gamma = (I, \Sigma_2, \mathcal{M})$$

$\mathcal{M} = \{M \in \mathcal{M}_2 \mid M \text{ is } Mod(\iota)\text{-cofree over } M|_\iota \text{ with counit } id\}$

Given a functor $G\colon \mathbf{B} \longrightarrow \mathbf{A}$, an object $B \in \mathbf{B}$ is called *G-cofree (with counit $\varepsilon_A\colon G(B) \longrightarrow A$) over $A \in \mathbf{A}$*, if for any object $B' \in \mathbf{B}$ and any morphism $h\colon G(B') \longrightarrow A$, there is a unique morphism $h^{\#}\colon B' \longrightarrow B$ such that $G(h^{\#}); \varepsilon_A = h$.



OMS Libraries

Syntax of DOL libraries

```
Document              ::= DOLLibrary | NativeDocument
NativeDocument        ::= <language and serialization specific>
DOLLibrary            ::= [PrefixMap] 'library' LibraryName
                              Qualification LibraryItem*
LibraryItem           ::= LibraryImport | Definition | Qualification
Definition            ::= OMSDefinition | NetworkDefinition | MappingDefinition
LibraryImport         ::= 'import' LibraryName
Qualification         ::= LanguageQualification
                        | LogicQualification
                        | SyntaxQualification
LanguageQualification ::= 'language' LanguageRef
LogicQualification ::= 'logic' LogicRef
SyntaxQualification ::= 'serialization' SyntaxRef
OMSDefinition         ::= OMSkeyword OMSName '='
                          [ConservativityStrength] OMS 'end'
OMSkeyword            ::= 'ontology'
                        | 'onto'
                        | 'specification'
                        | 'spec'
                        | 'model'
                        | 'oms'
```

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
```

Syntax of mappings

```
MappingDefinition  ::= InterpretationDefinition
                     | EntailmentDefinition
                     | EquivalenceDefinition
                     | ModuleRelDefinition
                     | AlignmentDefinition
InterpretationDefinition ::= InterpretationKeyword
                             InterpretationName
                             [Conservative] ':'
                             InterpretationType '='
                             LanguageTranslation*
                             [SymbolMap] 'end'
InterpretationKeyword ::= 'interpretation' | 'view' | 'refinement'
InterpretationName ::= IRI
InterpretationType ::= GroupOMS 'to' GroupOMS
```

Interpretations (refinements)

- **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
```

An interpretation in UML

```
%prefix( :      <http://www.example.org/uml#>
         uml:   <http://www.uml.org/spec/UML/>
%% descriptions of logics ...
         log:   <http://www.omg.org/spec/DOL/logics/>

logic log:uml

interpretation abstract_to_concrete_atm :
  psm to
  {  atm with Idle |-> Idle, CardEntered |-> Idle,
              PINEntered |-> Idle, Verified |-> Idle,
              Verifying |-> Verifying
     hide card, PIN   } = translation psm2atm
end
```

An interpretation in CASL

```
spec InsertSort =
  List
then
  ops insert : Elem*List[Elem] -> List[Elem];
      insert_sort : List[Elem]->List[Elem]
  vars x,y:Elem; L:List[Elem]
  . insert(x,[]) = x::[]
  . insert(x,y::L) = x::insert(y,L) when x<=y else y::
  . insert_sort([]) = []
  . insert_sort(x::L) = insert(x,insert_sort(L))
 hide insert
interpretation InsertSortCorrectness :
     Sorting to InsertSort =
    sorter |-> insert_sort
```

Semantics of interpretations

Let $[\![O_i]\!]_\Gamma^M = (I, \Sigma_i, \mathcal{M}_i)$ $(i = 1, 2)$

$$[\![\textbf{interpretation } IRI : O_1 \textbf{ to } O_2 = \sigma]\!]_\Gamma^M$$

is defined iff

$$Mod(\sigma)(\mathcal{M}_2) \subseteq \mathcal{M}_1$$

In this case, $\Gamma(IRI) := ((I, \Sigma_1, \mathcal{M}_1), (I, \Sigma_2, \mathcal{M}_2), \sigma)$.

Syntax of OMS networks (diagrams)

```
NetworkDefinition ::= 'network' NetworkName '='
                      [ConservativityStrength] Network
NetworkName       ::= IRI
Network           ::= NetworkElements [ExcludedElements]
NetworkElements ::= NetworkElement ( ',' NetworkElement )*
NetworkElement    ::= [Id ':'] ElementRef
ExcludedElements ::= 'excluding' ExcludedElement ( ',' ExcludedElement )*
ExcludedElement ::= PathReference | ElementRef
PathReference     ::= IRI '->' IRI
ElementRef        ::= IRI
Id                ::= Letter LetterOrDigit*
```

OMS networks (diagrams)

network N =
$N_1, \ldots, N_m, O_1, \ldots, O_n, M_1, \ldots, M_p$
excluding $N'_1, \ldots, N'_i, O'_1, \ldots, O'_j, M'_1, \ldots, M'_k$

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

Combinations

- combine N
- N is a network
- semantics is the (a) colimit of the diagram N

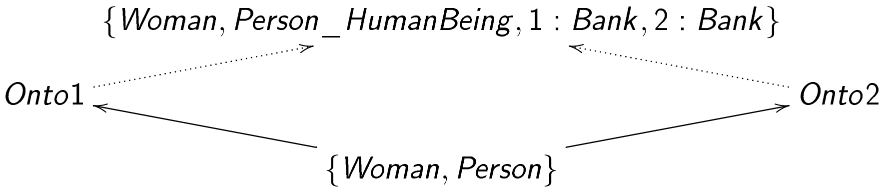
```
ontology AlignedOntology1 =
  combine N
```

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



Syntax of alignments

```
AlignmentDefinition ::= 'alignment' AlignmentName
                        [AlignmentCardinalityPair] ':'
                        AlignmentType
                        ['=' Correspondence ( ',' Correspondence )*]
                        ['assuming' AlignmentSemantics] 'end'
AlignmentName      ::= IRI
AlignmentCardinalityPair ::= AlignmentCardinalityForward
                             AlignmentCardinalityBackward
AlignmentCardinalityForward ::= AlignmentCardinality
AlignmentCardinalityBackward ::= AlignmentCardinality
AlignmentCardinality ::= '1' | '?' | '+' | '*'
AlignmentType      ::= GroupOMS 'to' GroupOMS
AlignmentSemantics ::= 'SingleDomain'
                     | 'GlobalDomain'
                     | 'ContextualizedDomain'
Correspondence     ::= CorrespondenceBlock | SingleCorrespondence | '*'
CorrespondenceBlock ::= 'relation' [Relation] [Confidence] '{'
                        Correspondence ( ',' Correspondence )* '}'
SingleCorrespondence ::= SymbolRef [Relation] [Confidence] SymbolRef
GeneralizedTerm    ::= SymbolRef
Relation           ::= '>' | '<' | '=' | '%' | 'ni' | 'in' | IRI
Confidence         ::= Double
```

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
```

Networks, revisited

network N =
$N_1, \ldots, N_m, O_1, \ldots, O_n, M_1, \ldots, M_p, A_1, \ldots, A_r$
excluding $N'_1, \ldots, N'_i, O'_1, \ldots, O'_j, M'_1, \ldots, M'_k$

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

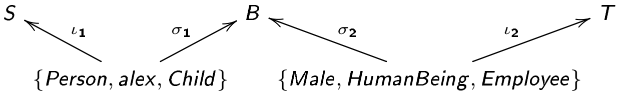
The resulting diagram N includes (institution-specific) W-alignment diagrams for each alignment A_i. Using **assuming**, assumptions about the domains of all OMS can be specified:

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



where

ontology B =

Class: *Person_HumanBeing*
Class: *Employee*
Class: *Child*
SubClassOf: ¬ *Employee*
Individual: *alex*
Types: *Male*

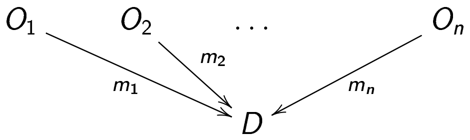
Resulting colimit

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

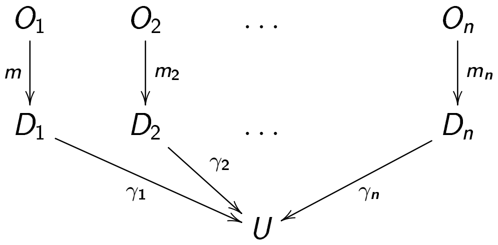
Framework: institutions like OWL, FOL, ...
Ontologies are interpreted over the same domain



- 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



- 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}

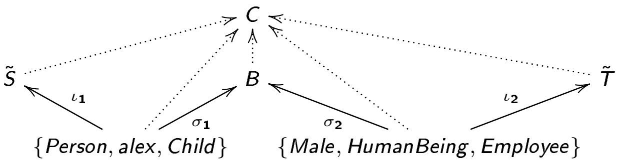
Relativization of an OWL ontology

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



where

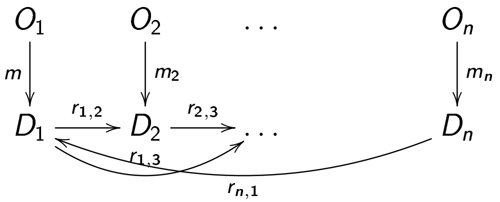
ontology B =
  Class: *Thing*$_S$ **Class:** *Thing*$_T$
  Class: *Person_HumanBeing* **SubClassOf:** *Thing*$_S$, *Thing*$_T$
  Class: *Male* **Class:** *Employee*
  Class: *Child* **SubClassOf:** *Thing*$_T$ **and** ¬ *Employee*
  Individual: *alex* **Types:** *Male*

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

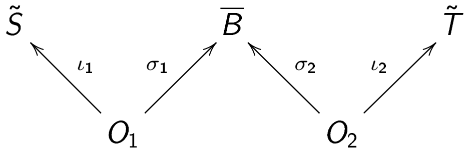
Framework: different domains related by coherent relations



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



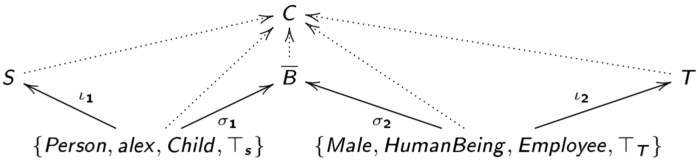
where $\overline{B}$ modifies B as follows:

- 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 $a_i \in \exists r_{ij}.c_j$
  - ...
- 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



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** $\neg$ *Employee*

Individual: *alex* **Types:** r_{ST} **some** *Male*, *Person*

Syntax of equivalences

```
EquivalenceDefinition ::= 'equivalence' EquivalenceName ':'
                          EquivalenceType 'end'
EquivalenceName    ::= IRI
EquivalenceType    ::= GroupOMS '<->' GroupOMS '=' OMS
                     | Network '<->' Network '=' Network
```

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
```

Syntax of module relations

```
ModuleRelDefinition ::= 'module' ModuleName [Conservative] ':'
                        ModuleType 'for' InterfaceSignature
ModuleName         ::= IRI
ModuleType         ::= GroupOMS 'of' GroupOMS
```

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.

Syntax of queries (only informative annex!)

```
Term                     ::= ($<$)an expression specific to an OMS language($>$)
GeneralizedTerm          ::= Term | SymbolRef
QueryRelatedDefinition   ::= QueryDefinition
                           | SubstitutionDefinition
                           | ResultDefinition
QueryDefinition          ::= 'query' QueryName '=' 'select' Vars 'where'
                             Sentence 'in' GroupOMS
                             ['along' OMSLanguageTranslation] 'end'
SubstitutionDefinition   ::= 'substitution' SubstitutionName ':'
                                 GroupOMS 'to' GroupOMS '=' SymbolMap
                                 'end'
ResultDefinition         ::= 'result' ResultName '=' SubstitutionName
                             ( ',' SubstitutionName )* 'for' QueryName
                             ['%complete'] 'end'
OMS                      ::= ($...$) | OMS 'with' SubstitutionName
QueryName                ::= IRI
SubstitutionName         ::= 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?

Proof calculus

Structured specifications over an arbitrary institution (covers part of DOL OMS)

SP	$::=$	$\langle \Sigma, \Gamma \rangle$	basic specification
	$\mid$	$SP \cup SP$	union
	$\mid$	$\sigma(SP)$	translation
	$\mid$	$SP\vert_{\sigma}$	hiding

. . . and their semantics

Definition (Signature and model class of a specification)

$Sig(\langle\Sigma,\Gamma\rangle) = \Sigma$
$Mod(\langle\Sigma,\Gamma\rangle) = \{M \in Mod(\Sigma) | M \models \Gamma\}$

$Sig(SP_1 \cup SP_2) = Sig(SP_1) = Sig(SP_2)$
$Mod(SP_1 \cup SP_2) = Mod(SP_1) \cap Mod(SP_2)$

$Sig(\sigma : \Sigma_1 \longrightarrow \Sigma_2(SP)) = \Sigma_2$
$Mod(\sigma(SP)) = \{M \in Mod(\Sigma_2) \mid M|_\sigma \in Mod(SP)\}$

$Sig(SP|_{\sigma : \Sigma_1 \longrightarrow \Sigma_2}) = \Sigma_1$
$Mod(SP|_{\sigma : \Sigma_1 \longrightarrow \Sigma_2}) = \{M|_\sigma \mid M \in Mod(SP)\}$

Definition (Logical consequence, specification refinement)

$$SP \models \varphi \quad \text{iff} \quad M \models \varphi \text{ for all } M \in Mod(SP)$$

Entailment systems

Definition

Given an institution $\mathcal{I} = (\mathbf{Sign}, \mathbf{Sen}, Mod, \models)$, an *entailment system* $\vdash$ for $\mathcal{I}$ consists of relations $\vdash_\Sigma \subseteq \mathcal{P}(\mathbf{Sen}(\Sigma)) \times \mathbf{Sen}(\Sigma)$ such that

1. *reflexivity:* for any $\varphi \in \mathbf{Sen}(\Sigma)$, $\{\varphi\} \vdash_\Sigma \varphi$,
2. *monotonicity:* if $\Gamma \vdash_\Sigma \varphi$ and $\Gamma' \supseteq \Gamma$ then $\Gamma' \vdash_\Sigma \varphi$,
3. *transitivity:* if $\Gamma \vdash_\Sigma \varphi_i$ for $i \in I$ and $\Gamma \cup \{\varphi_i \mid i \in I\} \vdash_\Sigma \psi$, then $\Gamma \vdash_\Sigma \psi$,
4. $\vdash$*-translation:* if $\Gamma \vdash_\Sigma \varphi$, then for any $\sigma : \Sigma \longrightarrow \Sigma'$ in $\mathbf{Sign}$, $\sigma(\Gamma) \vdash_{\Sigma'} \sigma(\varphi)$,
5. *soundness:* if $\Gamma \vdash_\Sigma \varphi$ then $\Gamma \models_\Sigma \varphi$.

The entailment system is *complete* if, in addition, $\Gamma \models_\Sigma \varphi$ implies $\Gamma \vdash_\Sigma \varphi$.

Proof calculus for entailment (Borzyszkowski)

$$(CR)\ \frac{\{SP \vdash \varphi_i\}_{i \in I} \quad \{\varphi_i\}_{i \in I} \vdash \varphi}{SP \vdash \varphi} \qquad (basic)\ \frac{\varphi \in \Gamma}{\langle \Sigma, \Gamma \rangle \vdash \varphi}$$

$$(sum1)\ \frac{SP_1 \vdash \varphi}{SP_1 \cup SP_2 \vdash \varphi} \qquad (sum2)\ \frac{SP_1 \vdash \varphi}{SP_1 \cup SP_2 \vdash \varphi}$$

$$(trans)\ \frac{SP \vdash \varphi}{\sigma(SP) \vdash \sigma(\varphi)} \qquad (derive)\ \frac{SP \vdash \sigma(\varphi)}{SP|_\sigma \vdash \varphi}$$

Soundness means: $SP \vdash \varphi$ implies $SP \models \varphi$

Completeness means: $SP \models \varphi$ implies $SP \vdash \varphi$

Proof calculus for refinement (Borzyszkowski)

$$(Basic)\ \frac{SP \vdash \Gamma}{\langle \Sigma, \Gamma \rangle \rightsquigarrow SP} \qquad (Sum)\ \frac{SP_1 \rightsquigarrow SP \quad SP_2 \rightsquigarrow SP}{SP_1 \cup SP_2 \rightsquigarrow SP}$$

$$(Trans)\ \frac{SP \rightsquigarrow SP'|_\sigma}{\sigma(SP) \rightsquigarrow SP'}$$

$$(Derive)\ \frac{SP \rightsquigarrow SP''}{SP|_\sigma \rightsquigarrow SP'}$$

if $\sigma : SP' \longrightarrow SP''$
is a conservative extension

Soundness means: $SP_1 \rightsquigarrow SP_2$ implies $SP_1 \leadsto SP_2$
Completeness means: $SP_1 \leadsto SP_2$ implies $SP_1 \rightsquigarrow SP_2$

Craig-Robinson interpolation

Definition

$$\begin{array}{ccccc} \Psi & & \Sigma \xrightarrow{\varphi_1} \Sigma_1 & & (1)\ \Psi_1 \models \varphi_1(\Psi) \\ & & {\scriptstyle\varphi_2}\downarrow \qquad \downarrow{\scriptstyle\theta_1} & & \\ (2)\ \varphi_2(\Psi) \cup \Gamma_2 \models \Psi_2 & & \Sigma_2 \xrightarrow[\theta_2]{} \Sigma' & & (0)\ \theta_1(\Psi_1) \cup \theta_2(\Gamma_2) \models \theta_2(\Psi_2) \end{array}$$

A commutative square admits *Craig-Robinson interpolation*, if for all finite $\Psi_1 \subseteq Sen(\Sigma_1)$, $\Psi_2, \Gamma_2 \subseteq Sen(\Sigma_2)$, if (0), then there exists a finite $\Psi \subseteq Sen(\Sigma)$ with (1) and (2).

$\mathcal{I}$ has *Craig-Robinson interpolation* if all signature pushouts admit Craig-Robinson interpolation.

Soundness and Completeness

Theorem (Borzyszkowski, Tarlecki, Diaconescu)

Under the assumptions that

- *the institution admits Craig-Robinson interpolation,*
- *the institution is weakly semi-exact, and*
- *the entailment system is complete,*

the calculus for structured entailment and refinement is sound and complete.

For refinement, we need an oracle for conservative extensions.
Weak semi-exactness = *Mod* maps pushouts to weak pullbacks

Problem: Craig interpolation often fails:

- many-sorted FOL (with non-injective signature morphisms)
- many-sorted equational logic
- quantified modal logic S5

Structured normal form

$$\frac{}{snf(\langle \Sigma, \Gamma \rangle) = \langle \Sigma, \Gamma \rangle|_{id}}$$

$$\frac{snf(SP_1) = SP_1'|_{\sigma_1} \quad snf(SP_2) = SP_2'|_{\sigma_2}}{snf(SP_1 \cup SP_2) = (\theta_1(SP_1') \cup \theta_2(SP_2'))|_{\sigma_1;\theta_1}} \quad \text{if} \quad \begin{array}{ccc} Sig[SP_1] & \xrightarrow{\sigma_1} & Sig[SP_1'] \\ \downarrow{\scriptstyle \sigma_2} & & \downarrow{\scriptstyle \theta_1} \\ Sig[SP_2'] & \xrightarrow{\theta_2} & \Sigma' \end{array}$$

$$\frac{snf(SP) = SP'|_{\sigma_1}}{snf(\sigma_2(SP)) = (\theta_1(SP'))|_{\theta_2}} \quad \text{if} \quad \begin{array}{ccc} Sig[SP] & \xrightarrow{\sigma_1} & Sig[SP'] \\ \downarrow{\scriptstyle \sigma_2} & & \downarrow{\scriptstyle \theta_1} \\ \Sigma_2 & \xrightarrow{\theta_2} & \Sigma' \end{array} \quad \text{is a pushout}$$

$$\frac{snf(SP) = SP'|_{\sigma}}{snf(SP|_{\theta}) = SP|_{\theta\,;\,\sigma}}$$

Properties of the structured normal form

Proposition

In any weakly semi-exact institution, SP and snf(SP) are equivalent.

Moreover, we can obtain a stronger completeness result:

Theorem

Under the assumptions that the institution is weakly semi-exact and the entailment system is complete, the calculi for specification entailments and refinement between structured specifications extended by the following structured normal form rule:

$$(snf)\ \frac{SP' \vdash \sigma(\varphi)}{SP \vdash \varphi} \qquad \text{if } snf(SP) = SP'|_\sigma$$

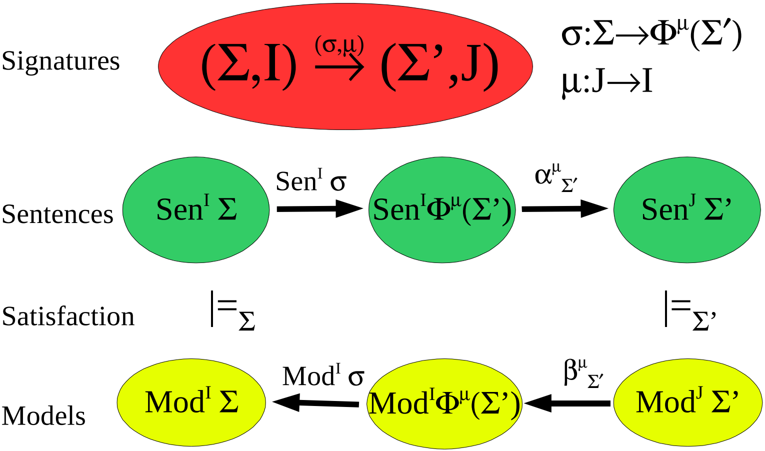
Heterogeneous specification

Definition

A *heterogeneous logical environment* ($\mathcal{HLE}$) (or indexed coinstitution) is diagram of institutions and comorphisms.

Grothendieck institution over an $\mathcal{HLE}$

The Grothendieck Institution



Heterogeneous structuring operations

heterogeneous translation: For any $\mathcal{I}$-specification SP, $\rho(SP)$ is a specification with:

$$Sig[\rho(SP)] := \Phi(Sig[SP])$$

$$Mod[\rho(SP)] := \beta^{-1}_{Sig[SP]}(Mod[SP])$$

heterogeneous hiding: For any $\mathcal{I}'$-specification SP' and signature Σ with $Sig[SP'] = \Phi(\Sigma)$, $SP'|^{\Sigma}_{\rho}$ is a specification with:

$$Sig[SP'|^{\Sigma}_{\rho}] := \Sigma$$

$$Mod[SP'|^{\Sigma}_{\rho}] := \beta_{\Sigma}(Mod[SP'])$$

This can be interpreted as structuring in the Grothendieck institution.

A heterogeneous proof calculus

$$(\textit{het-trans})\ \frac{SP \vdash \varphi}{\rho(SP) \vdash \alpha(\varphi)} \qquad (\textit{het-derive})\ \frac{SP \vdash \alpha(\varphi)}{SP|^{\Sigma}_{\rho} \vdash \varphi}$$

$$(\textit{borrowing})\ \frac{\rho(SP) \vdash \alpha(\varphi)}{SP \vdash \varphi} \qquad \text{if } \rho \text{ is model-expansive}$$

$$(\textit{Het-snf})\ \frac{SP' \vdash \sigma(\alpha(\varphi))}{SP \vdash \varphi} \qquad \text{if } \textit{hsnf}(SP) = (SP'|_{\sigma})|^{\Sigma}_{\rho}$$

A heterogeneous proof calculus for refinement

$$(\mathit{Het\text{-}Trans})\ \frac{SP \rightsquigarrow SP'|^{\Sigma}_{\rho}}{\rho(SP) \rightsquigarrow SP'}$$

$$(\mathit{Het\text{-}Derive})\ \frac{SP \rightsquigarrow SP''}{SP|^{\Sigma}_{\rho} \rightsquigarrow SP'}$$ if $\rho\colon SP' \longrightarrow SP''$ is a conservative extension

Conservativity of $\rho = (\Phi, \alpha, \beta)\colon SP' \longrightarrow SP''$ means that for each model $M' \in Mod(SP')$, there is a model $M'' \in Mod(SP'')$ with $\beta(M'') = M'$.

Heterogeneous completeness

Theorem

For a lax heterogeneous logical environment $\mathcal{HLE} : \mathcal{G} \longrightarrow co\mathcal{INS}$ (with some of the institutions having entailment systems), the proof calculi for heterogeneous specifications are sound for $\mathcal{I}^{\mathcal{HLE}}/\equiv$. If

1. *$\mathcal{HLE}$ is lax-quasi-exact,*
2. *all institution comorphisms in $\mathcal{HLE}$ are weakly exact,*
3. *there is a set $\mathcal{L}$ of institutions in $\mathcal{HLE}$ that come with complete entailment systems,*
4. *all institutions in $\mathcal{L}$ are quasi-semi-exact,*
5. *from each institution in $\mathcal{HLE}$, there is some model-expansive comorphism in $\mathcal{HLE}$ going into some institution in $\mathcal{L}$,*

the proof calculus for entailments between heterogeneous specifications and sentences is complete over $\mathcal{I}^{\mathcal{HLE}}/\equiv$. If, moreover,

Tool support

Tool support: Heterogeneous Tool Set (Hets)

- available at `http://hets.eu`
- 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

Conclusion

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`

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 for whole of DOL

Related work

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