# Modular and heterogeneous logical theories in DOL

Till Mossakowski
joint work with Oliver Kutz, Mihai Codescu, Fabian Neuhaus et al.

Otto-von-Guericke-Universität Magdeburg

Augsburg, Jan 25, 2018

# Overview

# Overview

# Computer science uses logic in many ways

- programming languages
- formal specification and verification
- databases, WWW, artificial intelligence
- ontologies
- algorithms & complexity
- (semi-)automated theorem proving
- metatheory
- ...

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p, \neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \top, \bot$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p, \neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \top, \bot$ |
| modal logics | $\ldots, \Box\varphi, \Diamond\varphi$ |
| | |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\top$, $\bot$ |
| modal logics | $\ldots$, $\Box\varphi$, $\Diamond\varphi$ |
| hybrid logics | $\ldots$, $i$, $@i.\varphi$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\top$, $\bot$ |
| modal logics | $\ldots$, $\Box\varphi$, $\Diamond\varphi$ |
| hybrid logics | $\ldots$, $i$, $@i.\varphi$ |
| temporal logics | $\ldots$, $F\varphi$, $G\varphi$, $\varphi U \psi$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \to \psi$, $\top$, $\bot$ |
| modal logics | $\dots$, $\Box\varphi$, $\Diamond\varphi$ |
| hybrid logics | $\dots$, $i$, $@i.\varphi$ |
| temporal logics | $\dots$, $F\varphi$, $G\varphi$, $\varphi U \psi$ |
| QBF | $\dots$, $\exists p.\varphi$, $\forall p.\varphi$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\top$, $\bot$ |
| modal logics | $\ldots$, $\Box\varphi$, $\Diamond\varphi$ |
| hybrid logics | $\ldots$, $i$, $@i.\varphi$ |
| temporal logics | $\ldots$, $F\varphi$, $G\varphi$, $\varphi U \psi$ |
| QBF | $\ldots$, $\exists p.\varphi$, $\forall p.\varphi$ |
| first-order logics | $\ldots$, $t = u$, $P(t_1, \ldots, t_n)$, $\exists x.\varphi$, $\forall x.\varphi$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\top$, $\bot$ |
| modal logics | $\ldots$, $\Box\varphi$, $\Diamond\varphi$ |
| hybrid logics | $\ldots$, $i$, $@i.\varphi$ |
| temporal logics | $\ldots$, $F\varphi$, $G\varphi$, $\varphi U \psi$ |
| QBF | $\ldots$, $\exists p.\varphi$, $\forall p.\varphi$ |
| first-order logics | $\ldots$, $t = u$, $P(t_1, \ldots, t_n)$, $\exists x.\varphi$, $\forall x.\varphi$ |
| description logics | $C \sqsubseteq D \qquad C ::= A \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \ldots$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\top$, $\bot$ |
| modal logics | $\ldots$, $\Box\varphi$, $\Diamond\varphi$ |
| hybrid logics | $\ldots$, $i$, $@i.\varphi$ |
| temporal logics | $\ldots$, $F\varphi$, $G\varphi$, $\varphi U \psi$ |
| QBF | $\ldots$, $\exists p.\varphi$, $\forall p.\varphi$ |
| first-order logics | $\ldots$, $t = u$, $P(t_1, \ldots, t_n)$, $\exists x.\varphi$, $\forall x.\varphi$ |
| description logics | $C \sqsubseteq D \qquad C ::= A \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \ldots$ |
| higher-order logics | $\ldots, P\ t, \exists P.\varphi, \forall P.\varphi, \iota x.\varphi \quad t ::= f\ t \mid \lambda x.t \mid \ldots$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p, \neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \top, \bot$ |
| modal logics | $\ldots, \Box\varphi, \Diamond\varphi$ |
| hybrid logics | $\ldots, i, @i.\varphi$ |
| temporal logics | $\ldots, F\varphi, G\varphi, \varphi U \psi$ |
| QBF | $\ldots, \exists p.\varphi, \forall p.\varphi$ |
| first-order logics | $\ldots, t = u, P(t_1, \ldots, t_n), \exists x.\varphi, \forall x.\varphi$ |
| description logics | $C \sqsubseteq D \quad C ::= A \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \ldots$ |
| higher-order logics | $\ldots, P\ t, \exists P.\varphi, \forall P.\varphi, \iota x.\varphi \quad t ::= f\ t \mid \lambda x.t \mid \ldots$ |
| type theories | $\ldots, \forall \alpha.\forall f : \alpha \rightarrow \alpha.\varphi$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p, \neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \top, \bot$ |
| modal logics | $\ldots, \Box\varphi, \Diamond\varphi$ |
| hybrid logics | $\ldots, i, @i.\varphi$ |
| temporal logics | $\ldots, F\varphi, G\varphi, \varphi U \psi$ |
| QBF | $\ldots, \exists p.\varphi, \forall p.\varphi$ |
| first-order logics | $\ldots, t = u, P(t_1, \ldots, t_n), \exists x.\varphi, \forall x.\varphi$ |
| description logics | $C \sqsubseteq D \qquad C ::= A \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \ldots$ |
| higher-order logics | $\ldots, P\ t, \exists P.\varphi, \forall P.\varphi, \iota x.\varphi \quad t ::= f\ t \mid \lambda x.t \mid \ldots$ |
| type theories | $\ldots, \forall \alpha.\forall f : \alpha \rightarrow \alpha.\varphi$ |
| Hoare logics | $\{\varphi\}P\{\psi\} \quad P ::= x := t \mid \text{while } \varphi \text{ do } P \mid \ldots$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \to \psi$, $\top$, $\bot$ |
| modal logics | $\ldots$, $\Box\varphi$, $\Diamond\varphi$ |
| hybrid logics | $\ldots$, $i$, $@i.\varphi$ |
| temporal logics | $\ldots$, $F\varphi$, $G\varphi$, $\varphi U \psi$ |
| QBF | $\ldots$, $\exists p.\varphi$, $\forall p.\varphi$ |
| first-order logics | $\ldots$, $t = u$, $P(t_1, \ldots, t_n)$, $\exists x.\varphi$, $\forall x.\varphi$ |
| description logics | $C \sqsubseteq D \quad C ::= A \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \ldots$ |
| higher-order logics | $\ldots, P\ t, \exists P.\varphi, \forall P.\varphi, \iota x.\varphi \quad t ::= f\ t \mid \lambda x.t \mid \ldots$ |
| type theories | $\ldots$, $\forall\alpha.\forall f : \alpha \to \alpha.\varphi$ |
| Hoare logics | $\{\varphi\}P\{\psi\} \quad P ::= x := t \mid \text{while } \varphi \text{ do } P \mid \ldots$ |
| dynamic logics | $[P]\varphi$, $\langle P\rangle\varphi \quad P ::= a \mid P? \mid PQ \mid P \cup Q \mid P^*$ |

# Plenty of logics are used in computer science

| | |
|---|---|
| propositional logics | $p, \neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \top, \bot$ |
| modal logics | $\ldots, \Box\varphi, \Diamond\varphi$ |
| hybrid logics | $\ldots, i, @i.\varphi$ |
| temporal logics | $\ldots, F\varphi, G\varphi, \varphi U \psi$ |
| QBF | $\ldots, \exists p.\varphi, \forall p.\varphi$ |
| first-order logics | $\ldots, t = u, P(t_1, \ldots, t_n), \exists x.\varphi, \forall x.\varphi$ |
| description logics | $C \sqsubseteq D \qquad C ::= A \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \ldots$ |
| higher-order logics | $\ldots, P\ t, \exists P.\varphi, \forall P.\varphi, \iota x.\varphi \quad t ::= f\ t \mid \lambda x.t \mid \ldots$ |
| type theories | $\ldots, \forall \alpha.\forall f : \alpha \rightarrow \alpha.\varphi$ |
| Hoare logics | $\{\varphi\}P\{\psi\} \quad P ::= x := t \mid \text{while } \varphi \text{ do } P \mid \ldots$ |
| dynamic logics | $[P]\varphi, \langle P\rangle\varphi \qquad P ::= a \mid P? \mid PQ \mid P \cup Q \mid P^*$ |
| linear logic | $!\varphi, ?\varphi, \varphi \otimes \psi, \varphi \oplus \psi, \varphi \& \psi, \varphi \parr \psi$ |
| ... | |

# What do these logics have in common?

- formulas / sentences
- entailment, logical consequence
- models
- soundness, completeness
- conservative extensions
- ...

Are there definitions and theorems that we can

- introduce once and for all
- and then apply them to many logics?

# What is a logic, after all?

## Definition (Gentzen, Tarski, Scott)

An entailment relation (ER) $(S, \vdash)$ is a binary relation $\vdash \subseteq \mathscr{P}(S) \times S$ on a set $S$ of sentences.

$(S, \vdash)$ is Tarskian, if

1. reflexivity: for any $\varphi \in S$, $\{\varphi\} \vdash \varphi$,
2. monotonicity: if $\Gamma \vdash \varphi$ and $\Gamma' \supseteq \Gamma$ then $\Gamma' \vdash \varphi$,
3. transitivity: if $\Gamma \vdash \varphi_i$, for $i \in I$, and $\Gamma \cup \{\varphi_i \mid i \in I\} \vdash \psi$, then $\Gamma \vdash \psi$.

# What is a logic, after all?

## Definition (Gentzen, Tarski, Scott)

An entailment relation (ER) $(S, \vdash)$ is a binary relation $\vdash \subseteq \mathscr{P}(S) \times S$ on a set $S$ of sentences.

$(S, \vdash)$ is Tarskian, if

1. reflexivity: for any $\varphi \in S$, $\{\varphi\} \vdash \varphi$,
2. monotonicity: if $\Gamma \vdash \varphi$ and $\Gamma' \supseteq \Gamma$ then $\Gamma' \vdash \varphi$,
3. transitivity: if $\Gamma \vdash \varphi_i$, for $i \in I$, and $\Gamma \cup \{\varphi_i \mid i \in I\} \vdash \psi$, then $\Gamma \vdash \psi$.

## Definition

A theory $\Gamma \subseteq S$ is consistent if $\Gamma \nvdash \varphi$ for some $\varphi$.

# ER for propositional logic

## Example (Propositional logic)

Propositional logic (**PL**) has sentences given by the following grammar

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \top \mid \bot$$

where $p$ denotes propositional variables.

$\vdash$ is the minimal Tarskian entailment relation satisfying:

$$\frac{\Gamma \vdash \varphi, \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi} \qquad \frac{\Gamma, \varphi \vdash \chi, \quad \Gamma, \psi \vdash \chi}{\Gamma, \varphi \vee \psi \vdash \chi} \qquad \frac{\Gamma \vdash \varphi \rightarrow \psi}{\Gamma, \varphi \vdash \psi}$$

$$\frac{}{\Gamma \vdash \top} \qquad \frac{}{\bot \vdash \varphi} \qquad \frac{\Gamma \vdash \neg\varphi}{\Gamma, \varphi \vdash \bot} \qquad \frac{\Gamma \vdash \neg\neg\varphi}{\Gamma \vdash \varphi}$$

# ER for modal logic K

## Example (modal logic K)

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \top \mid \bot \mid \Box\varphi \mid \diamond\varphi$$

$\vdash$ is the minimal Tarskian entailment relation satisfying the rules for propositional logic plus:

$$\frac{}{\Gamma \vdash \Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)} \qquad \frac{\vdash \varphi}{\vdash \Box\varphi} \qquad \frac{\Gamma \vdash \diamond\varphi}{\Gamma \vdash \neg\Box\neg\varphi}$$

Note: with multiple modalities, this is equivalent to the description logic $\mathscr{ALC}$.

# ER for first-order logic (without function symbols)

## Example (First-order logic)

$$
\begin{aligned}
t ::= \;& x \mid c \\
\varphi ::= \;& P(t_1, \ldots, t_n) \mid \exists x.\varphi \mid \forall x.\varphi \mid \\
& \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \top \mid \bot
\end{aligned}
$$

$\vdash$ is the minimal Tarskian entailment relation satisfying the rules for propositional logic plus:

$$
\frac{\Gamma \vdash \varphi(t)}{\Gamma \vdash \exists x.\varphi(x)} \qquad \frac{\Gamma, \varphi(c) \vdash \psi}{\Gamma, \exists x.\varphi(x) \vdash \psi} \; (c \text{ does not occur in } \Gamma,\ \varphi,\ \psi)
$$

$$
\frac{\Gamma \vdash \forall x.\varphi(x)}{\Gamma \vdash \varphi(t)} \qquad \frac{\Gamma \vdash \varphi(c)}{\Gamma \vdash \forall x.\varphi(x)} \; (c \text{ does not occur in } \Gamma,\ \varphi)
$$

# Morphisms of entailment relations

## Definition

An entailment relation morphism $\alpha\colon (S_1, \vdash^1) \longrightarrow (S_2, \vdash^2)$ is a function $\alpha\colon S_1 \longrightarrow S_2$ such that

$$\Gamma \vdash^1 \varphi \quad \text{implies} \quad \alpha(\Gamma) \vdash^2 \alpha(\varphi)$$

If the converse holde, $\alpha$ is conservative.
ERs and ER morphisms form a category $\mathbb{ER}$.

Observation:

> *If we have a conservative ER morphism and a theorem prover for $\vdash^2$, we can borrow it for $\vdash^1$.*

For propositional and first-order logic, there are many automated theorem provers, but not for modal logic.

# Translating modal logic K into first-order logic

## Example

A conservative ER morphism Modal $\to$ FOL is defined by

$$
\begin{aligned}
&\alpha_x(p) = p(x)\\
&\alpha_x(\Box\varphi) = \forall y . R(x,y) \to \alpha_y(\varphi)\\
&\alpha_x(\diamond\varphi) = \exists y . R(x,y) \wedge \alpha_y(\varphi)\\
&\alpha_x(\neg\varphi) = \neg\alpha_x(\varphi)\\
&\ldots\\
&\alpha(\varphi) = \forall x . \alpha_x(\varphi)
\end{aligned}
$$

Proof of ER property: induction over proofs.
Proof of conservativity property is more complicated
$\Rightarrow$ use model theory.

# Overview

# Adding model theory

## Definition (Goguen, Burstall)

A satisfaction system $(S, \mathscr{M}, \models)$ consists of

- a set of $S$ of sentences,
- a category $\mathscr{M}$ of models and model homomorphisms, and
- a binary relation $\models \subseteq |\mathscr{M}| \times S$, the satisfaction relation.

## Definition (Logical consequence)

$\Gamma \models \varphi$ iff for all $M \in \mathscr{M}$, $M \models \Gamma$ implies $M \models \varphi$.

# Logics

## Definition (Logic)

A logic $(S, \vdash, \mathscr{M}, \models)$ consists of

- an entailment relation $(S, \vdash)$, and
- a satisfaction system $(S, \mathscr{M}, \models)$,

such that soundness holds:

$$\Gamma \vdash \varphi \text{ implies } \Gamma \models \varphi$$

A logic is complete, if

$$\Gamma \models \varphi \text{ implies } \Gamma \vdash \varphi$$

# Satisfaction system for propositional logic

## Example (Propositional logic)

**sentences** as above

**models** maps from propositional variables to $\{true, false\}$

**model homomorphisms** $M_1 \to M_2$ iff
$(M_1(p) = true$ implies $M_2(p) = true)$

**satisfaction** $M \models \varphi$ iff
$M(\varphi) = true$ according to standard truth tables

## Proposition

*Propositional logic is sound and complete.*

# Satisfaction system for modal logic K

## Example (modal logic K)

- sentences as above
- a model $M$ consists of
  - a non-empty set $W$ of worlds,
  - a binary accessibility relation $R \subseteq W \times W$,
  - a map maps from propositional variables and worlds to $\{\mathit{true}, \mathit{false}\}$
- satisfaction
  - $M, w \models p$ iff $M(p, w) = \mathit{true}$
  - $M, w \models \Box\varphi$ iff for all $v \in W$ with $R(w, v)$, $M, v \models \varphi$
  - $M, w \models \diamond\varphi$ iff for somd $v \in W$ with $R(w, v)$, $M, v \models \varphi$
  - $M, w \models \neg\varphi$ iff $M, w \not\models \varphi$ etc.
  - $M \models \varphi$ iff for all $w \in W$, $\quad M, w \models \varphi$

# Satisfaction system for modal logic K (cont'd)

**Proposition**

*modal logic K is sound and complete.*

# Satisfaction system for first-order logic

## Example (First-order logic)

- sentences as above
- models: a first-order $M$ model consist of
  - a non-empty set $|M|$ called universe,
  - an element $M_c \in |M|$ for each constant $c$,
  - an $n$-ary relation $M_P$ on $|M|$ for each $n$-ary predicate symbol $P$
- satisfaction
  - $M, v \models P(t_1, \dots, t_n)$ iff $(v^{\#}(t_1), \dots, v^{\#}(t_n)) \in M_P$
  - $M, v \models \forall x.\varphi$ iff for all $\xi$ differing from $v$ at most for $x$, $\quad M, \xi \models \varphi$
  - $M, v \models \exists x.\varphi$ iff forsome $\xi$ differing from $v$ at most for $x$, $M, \xi \models \varphi$
  - $M, v \models \neg\varphi$ iff $M, v \not\models \varphi$ etc.
  - $M \models \varphi$ iff for all $v$, $\quad M, v \models \varphi$

# Satisfaction system for first-order logic (cont'd)

## Proposition

*First-order logic is sound and complete.*

# Morphisms of satisfaction systems

## Definition (Goguen, Burstall)

A satisfaction system morphism
$(\alpha,\beta)\colon (S_1,\mathscr{M}_1,\models_1)\longrightarrow(S_2,\mathscr{M}_2,\models_2)$ consists of

- a sentence translation function $\alpha\colon S_1\longrightarrow S_2$, and
- a model reduction functor $\beta\colon \mathscr{M}_2\longrightarrow\mathscr{M}_1$, such that

$$M_2 \models_2 \alpha(\varphi_1) \text{ iff } \beta(M_2) \models_1 \varphi_1$$

(satisfaction condition).
This gives us a category $\mathbb{S}at$ of satisfaction systems and satisfaction system morphisms.

# Translating modal logic K into first-order logic

## Example

A satisfaction system morphism Modal $\rightarrow$ FOL is defined by

- sentence translation as above
- a first-order model is reduced to a modal model by
  - taking the universe as set of worlds
  - taking the interpretation of the binary predicate $R$ as accessibility relation
  - taking the interpretation of the unary predicate $p$ as interpretation of the propositional variable $p$

## Proposition

*The satisfaction condition holds.*
*Proof: induction over formulas.*

# Semantic proof of conservative ER morphism property

## Theorem (Cerioli, Meseguer)

*Let* $(S_1, \vdash_1, \mathscr{M}_1, \models_1)$ *and* $(S_2, \vdash_2, \mathscr{M}_2, \models_2)$ *be two sound and complete logics and a satisfaction system morphism*

$$(\alpha, \beta) : (S_1, \mathscr{M}_1, \models_1) \longrightarrow (S_2, \mathscr{M}_2, \models_2)$$

*be given.*
*If* $\beta$ *is surjectivive, then* $\alpha$ *is a conservative ER morphism*

$$\alpha : (S_1, \vdash_1) \longrightarrow (S_2, \vdash_2).$$

# Overview

# Indexing over signatures

We have been imprecise at various places.

# Indexing over signatures

We have been imprecise at various places.
Strictly speaking, we need to index over signatures. Signatures are vocabularies of non-logical (=user-defined) symbols.

# Indexing over signatures

We have been imprecise at various places.
Strictly speaking, we need to index over signatures. Signatures are vocabularies of non-logical (=user-defined) symbols.
Entailment:

- set $\mathrm{Sig}$ of signatures, and
- family of ERs $(\mathrm{Sen}(\Sigma), \vdash_\Sigma)_{\Sigma \in \mathrm{Sig}}$

# Indexing over signatures

We have been imprecise at various places.
Strictly speaking, we need to index over signatures. Signatures are vocabularies of non-logical (=user-defined) symbols.

Entailment:

- set $\mathbb{S}\mathrm{ig}$ of signatures, and
- family of ERs $(\mathbf{Sen}(\Sigma), \vdash_\Sigma)_{\Sigma \in \mathbb{S}\mathrm{ig}}$

Satisfaction:

- set $\mathbb{S}\mathrm{ig}$ of signatures, and
- family of satisfaction systems $(\mathbf{Sen}(\Sigma), \mathbf{Mod}(\Sigma), \models_\Sigma)_{\Sigma \in \mathbb{S}\mathrm{ig}}$

# However, this is not the whole story!

Within this framework, we can study

- abstract logical connectives
- logic translations
- logic combination
- consistency strength, expressiveness
- ...

# However, this is not the whole story!

Within this framework, we can study

- abstract logical connectives
- logic translations
- logic combination
- consistency strength, expressiveness
- ...

However, we cannot study

- refinements, conservative extensions
- modular logical theories
- abstract quantifiers
- Craig interpolation, Robinson consistency, Beth definability
- elementary diagrams
- ...

# Indexing over signature morphisms

## Definition (Fiadeiro, Meseguer)

An entailment system is a functor $I\colon \mathbb{Sig} \longrightarrow \mathbb{ER}$, where $\mathbb{Sig}$ is the category of signatures.

This gives us

- a graph $\mathbb{Sig}$ of signatures and signature morphisms,
- for each signature $\Sigma$, an identity morphism $id_\Sigma : \Sigma \longrightarrow \Sigma$,
- a composition operation $\circ$ on signature morphisms,
- for each $\Sigma \in \mathbb{Sig}$, an ER $(\mathbf{Sen}(\Sigma), \vdash_\Sigma)$,
- for each signature morphism $\sigma_1 : \Sigma_1 \longrightarrow \Sigma_2 \in \mathbb{Sig}$, an ER morphism $I(\sigma)\colon (\mathbf{Sen}(\Sigma_1), \vdash_{\Sigma_1}) \longrightarrow (\mathbf{Sen}(\Sigma_2), \vdash_{\Sigma_2})$, by abuse of notation also denoted by $\sigma$.

# Sample entailment systems

## Example (Entailment system for propositional logic)

**signatures** sets of propositional variables

**ERs** $(\mathrm{Sen}(\Sigma), \vdash_\Sigma)$ as before, but built over $\Sigma$

**ER morphisms** $\sigma(\varphi)$ replaces symbols in $\varphi$ along $\sigma$.
We have

$$\Gamma \vdash_{\Sigma_1} \varphi \text{ implies } \sigma(\Gamma) \vdash_{\Sigma_2} \sigma(\varphi)$$

Further examples:
modal logic, first-order logic, and many more.

# Indexing over signature morphisms (cont'd)

## Definition (Goguen, Burstall)

An institution is a functor $I \colon \mathrm{Sig} \longrightarrow \mathbb{S}at$.

This gives us

- a graph $\mathrm{Sig}$ of signatures and signature morphisms, $(\ldots)$
- for each $\Sigma \in \mathrm{Sig}$, a satisfaction system $(\mathrm{Sen}(\Sigma), \mathrm{Mod}(\Sigma), \models_\Sigma)$,
- for each signature morphism $\sigma_1 \colon \Sigma_1 \longrightarrow \Sigma_2 \in \mathrm{Sig}$, a satisfaction system morphism $I(\sigma) \colon (\mathrm{Sen}(\Sigma_1), \mathrm{Mod}(\Sigma_1), \models_{\Sigma_1}) \longrightarrow (\mathrm{Sen}(\Sigma_2), \mathrm{Mod}(\Sigma_2), \models_{\Sigma_2})$, by abuse of notation also denoted by $(\sigma, \_|_\sigma)$.

# Sample institutions

**Example (Institutions for propositional logic)**

signatures sets of propositional variables

Sat. systems $(\mathrm{Sen}(\Sigma), \mathrm{Mod}(\Sigma), \models_\Sigma)$ as before, but built over $\Sigma$

Sat. syst. morphisms
- $\sigma(\varphi)$ replaces symbols in $\varphi$ along $\sigma$
- $M|_\sigma$ interprets $p$ as $M|_\sigma(p) := M_{\sigma(p)}$
  We have
  $$M_2|_\sigma \models_{\Sigma_1} \varphi_1 \text{ iff } M_2 \models_{\Sigma_2} \sigma(\varphi_1)$$

Further examples:
modal logic, first-order logic, and many more.

# Indexing over signature morphisms (cont'd)

**Definition (Meseguer)**

A logic is an institution equipped with an entailment system, agreeing on signatures and sentences.

# Abstraction via institutions

Institution independent notions and theorems, languages, calculi, and software tools

Semantics, calculi and proof tools of particular institutions

# Overview

# Distributed Ontology, Model and Specification Language

## DOL

- DOL has been adopted as an OMG-Standard (under my leadership)
- combines modularity, interoperability and language heterogeneity
- continuous formal semantics, based on institutions
  - Ontologies, models and specifications are logical theories
- cooperation of different communities:
  - Ontologies, UML, specification

T. Mossakowski, C. Lange, O. Kutz (2012). Three Semantics for the Core of the Distributed Ontology Language, FOIS 2012. Best paper award

# Structured ontologies, models, specifications (OMS) over an arbitrary institution

| mathematical notation | | | type of OMS | DOL notation |
|---|---|---|---|---|
| $O$ | $::=$ | $\langle\Sigma,\Gamma\rangle$ | basic OMS | logic specific |
| | $\mid$ | $O_1 \cup O_2$ | union | $O_1$ **and** $O_2$ |
| | $\mid$ | $\sigma(O)$ | translation | $O$ **with** $\sigma$ |
| | $\mid$ | $O\vert_\sigma$ | hiding | $O$ **hide** $\sigma$ |

Note: $O_1$ **then** $\langle\Sigma,\Gamma\rangle$ is similar to $O_1$ **and** $\langle\Sigma,\Gamma\rangle$
(but $\Sigma$ can be signature fragment)

# . . . and their semantics

## Definition (Signature and model class of an OMS)

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

$$Sig(O_1 \cup O_2) = Sig(O_1) = Sig(O_2)$$
$$\mathrm{Mod}(O_1 \cup O_2) = \mathrm{Mod}(O_1) \cap \mathrm{Mod}(O_2)$$

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

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

# Unions

$O_1$ **and** $O_2$: union of two stand-alone OMS

- Signatures (and axioms) are united
- model classes are intersected
- difference to extensions: there, $O_2$ needs to be basic

```
logic CASL.FOL=
spec Magma =
  sort Elem;  ops 0:Elem; __+__:Elem*Elem->Elem  end
spec CommutativeMagma = Magma then
  forall x,y:Elem . x+y=y+x                     end
spec Monoid = Magma then
  forall x,y,z:Elem . x+0=x
                    . x+(y+z) = (x+y)+z         end
spec CommutativeMonoid =
  CommutativeMagma and Monoid                   end
```

# Translations

A translation $O$ **with** $\sigma$ renames $O$ along $\sigma$

- $\sigma$ is a signature morphism
- in practice, $\sigma$ is a symbol map, from which one can compute a signature morphism

```
ontology BankOntology =
  Class: Bank  Class: Account ...     end
ontology RiverOntology =
  Class: River  Class: Bank ...       end
ontology Combined =
  BankOntology with Bank |-> FinancialBank
 and
  RiverOntology with Bank |-> RiverBank
    %% necessary disambiguation when uniting OMS
end
```

# Hiding (preparation)

```
logic CASL.FOL=     %right_assoc( __::__ )%
spec PartialOrder =
  sort Elem;      pred __leq__ : Elem * Elem
  forall x,y,z:Elem
  . x leq x %(refl)%
  . x leq y /\ y leq x => x = y %(antisym)%
  . x leq y /\ y leq z => x leq z %(trans)%
spec TotalOrder =
  PartialOrder then forall x,y:Elem
  . x leq y \/ y leq x \/ x=y %(trichotomy)%
spec List = sort Elem
  free type List ::= [] | __::__(Elem; List)
  pred __elem__ : Elem * List
  forall x,y:Elem; L,L1,L2:List
  . not x elem []
  . x elem (y :: L) <=>  x=y \/ x elem L
```

# Hiding (cont'd)

```
spec AbstractSort =
  TotalOrder and List
then %def
  preds is_ordered : List;
        permutation : List * List
  op sorter : List->List
  forall x,y:Elem; L,L1,L2:List
  . is_ordered([])
  . is_ordered(x::[])
  . is_ordered(x::y::L) <=> x leq 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 is_ordered, permutation
end
```

# Logical notions for OMS

## Definition (Logical consequence for OMS)

$$O \models \varphi \qquad \text{iff} \qquad M \models \varphi \text{ for all } M \in \mathrm{Mod}(O)$$

## Definition (OMS refinement)

$$O \rightsquigarrow O' \qquad \text{iff} \qquad \mathrm{Mod}(O') \subseteq \mathrm{Mod}(O)$$

# Intended Consequences in Propositional Logic

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

Full specification at
`https://ontohub.org/esslli-2016/Propositional/leisure_structured.dol`

# A Countermodel

```
logic Propositional
spec Countermodel =
 props sunny, weekend, john_tennis, mary_shopping,
       saturday %% declaration of signature
  . sunny
  . not weekend
  . not john_tennis
  . not mary_shopping
  . saturday
end
```

This OMS has exactly one model, and hence can be seen as a syntactic description of this model.

# Repaired OMS

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

# Intended Consequences in FOL

```
logic CASL.FOL=
spec BooleanAlgebra =
  sort Elem
  ops 0,1 : Elem;
      __ cap __ : Elem * Elem -> Elem, assoc, comm, unit 1;
      __ cup __ : Elem * Elem -> Elem, assoc, comm, unit 0;
  forall x,y,z:Elem
  . x cap (x cup y) = x      %(absorption_def1)%
  . x cup (x cap y) = x      %(absorption_def2)%
  . x cap 0 = 0              %(zeroAndCap)%
  . x cup 1 = 1              %(oneAndCup)%
  . x cap (y cup z) = (x cap y) cup (x cap z)
                             %(distr1_BooleanAlgebra)%
  . x cup (y cap z) = (x cup y) cap (x cup z)
                             %(distr2_BooleanAlgebra)%
  . exists x' : Elem . x cup x' = 1 /\ x cap x' = 0
                             %(inverse_BooleanAlgebra)%
  . x cup x = x              %(idem_cup)% %implied
  . x cap x = x              %(idem_cap)% %implied
end
```

https://ontohub.org/esslli-2016/FOL/OrderTheory_structured.dol

# Structuring Using Extensions

```
logic Propositional
spec JohnMary_TBox = %% general rules
  props sunny, weekend, john_tennis, mary_shopping,
        saturday   %% declaration of signature
  . sunny /\ weekend => john_tennis %(when_tennis)%
  . john_tennis => mary_shopping   %(when_shopping)%
  . saturday => weekend  %(sat_weekend)%
end
spec JohnMary_ABox = %% specific facts
  JohnMary_TBox then
  . saturday                %(it_is_saturday)%
  . sunny                   %(it_is_sunny)%
  . mary_shopping    %(mary_goes_shopping)% %implied
end
```

# Implied Extensions in Prop

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

# Theory Morphisms

## Definition

A theory morphism $\sigma : (\Sigma_1, \Gamma_1) \to (\Sigma_2, \Gamma_2)$ is a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ such that

$$\text{for } M \in \mathrm{Mod}(\Sigma_2, \Gamma_2), \text{ we have } M|_\sigma \in \mathrm{Mod}(\Sigma_1, \Gamma_1)$$

Extensions are theory morphisms:

$$(\Sigma, \Gamma) \textbf{ then } (\Delta_\Sigma, \Delta_\Gamma)$$

leads to the theory morphism

$$(\Sigma, \Gamma) \xrightarrow{\iota} (\Sigma \cup \Delta_\Sigma, \iota(\Gamma) \cup \Delta_\Gamma)$$

Proof: $M \models \iota(\Gamma) \cup \Delta_\Gamma$ implies $M|_\iota \models \Gamma$ by the satisfaction condition.

# Interpretations (views, refinements)

- **interpretation** *name* : $O_1$ **to** $O_2$ = $\sigma$
- $\sigma$ is a signature morphism (if omitted, assumed to be identity)
- expresses that $\sigma$ is a theory morphism $O_1 \to O_2$

```
logic CASL.FOL=
spec RichBooleanAlgebra =
  BooleanAlgebra
then %def
  pred __ <= __ : Elem * Elem;
  forall x,y:Elem
  . x <= y <=> x cap y = x %(leq_def)%
end
interpretation order_in_BA :
  PartialOrder to RichBooleanAlgebra
end
```

# Sorting (cont'd)

Formal design specification for sorting:

```
spec InsertSort =  List then
  ops insert : Elem*List -> List;
      insert_sort : List->List
  vars x,y:Elem; L:List
  . insert(x,[]) = x::[]
  . x leq y => insert(x,y::L) = x::insert(y,L)
  . not x leq y => insert(x,y::L) = y::insert(x,L)
  . insert_sort([]) = []
  . insert_sort(x::L) = insert(x,insert_sort(L))
hide insert
end
```

# Correctness

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

```
interpretation correctness_int :
     AbstractSort to InsertSort
end

refinement correctness_ref =
     AbstractSort refined to InsertSort
end
```

Two notational variants with the same semantics.

# Criterion for Theory Morphisms

## Theorem

*A signature morphism* $\sigma : \Sigma_1 \to \Sigma_2$ *is a theory morphism* $\sigma : (\Sigma_1, \Gamma_1) \to (\Sigma_2, \Gamma_2)$ *iff*

$$\Gamma_2 \models_{\Sigma_2} \sigma(\Gamma_1)$$

## Proof.

By the satisfaction condition. $\square$

# Overview

# Proof calculus for entailment (Borzyszkowski)

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

$$(sum1)\ \frac{O_1 \vdash \varphi}{O_1 \cup O_2 \vdash \varphi} \qquad (sum2)\ \frac{O_1 \vdash \varphi}{O_1 \cup O_2 \vdash \varphi}$$

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

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

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

# Proof calculus for refinement (Borzyszkowski)

$$(Basic)\ \frac{O \vdash \Gamma}{\langle \Sigma, \Gamma \rangle \rightsquigarrow O} \qquad (Sum)\ \frac{O_1 \rightsquigarrow O \quad O_2 \rightsquigarrow O}{O_1 \cup O_2 \rightsquigarrow O}$$

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

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

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

Soundness means: $O_1 \rightsquigarrow O_2$ implies $O_1 \leadsto\!\!\!\!> O_2$
Completeness means: $O_1 \leadsto\!\!\!\!> O_2$ implies $O_1 \rightsquigarrow O_2$

# Craig-Robinson interpolation

## Definition

$\Psi$

$$\begin{tikzcd} \Sigma \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Sigma_1 \arrow[d, "\theta_1"] \\ \Sigma_2 \arrow[r, "\theta_2"'] & \Sigma' \end{tikzcd}$$

(1) $\Psi_1 \models \varphi_1(\Psi)$

(2) $\varphi_2(\Psi) \cup \Gamma_2 \models \Psi_2$

(0) $\theta_1(\Psi_1) \cup \theta_2(\Gamma_2) \models \theta_2(\Psi_2)$

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

$\mathscr{I}$ 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 = $\mathrm{Mod}$ maps pushouts to weak pullbacks

# Overview

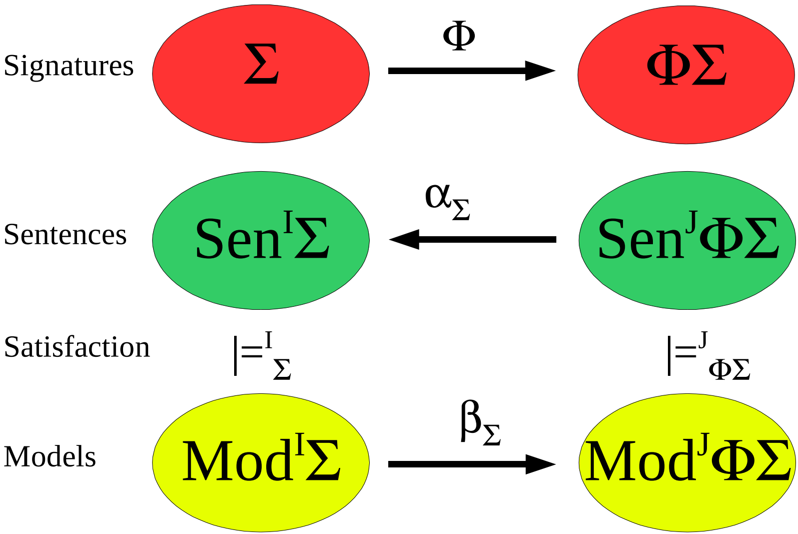
# Heterogeneous OMS

## Definition

A heterogeneous logical environment ($\mathscr{HLE}$) is diagram of institutions, morphisms and comorphisms.

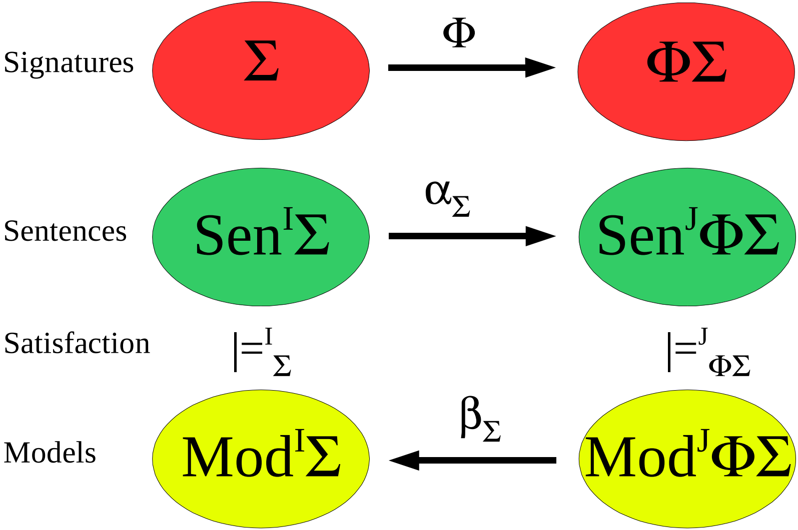
# Institution morphisms (projections)

## Institution morphisms

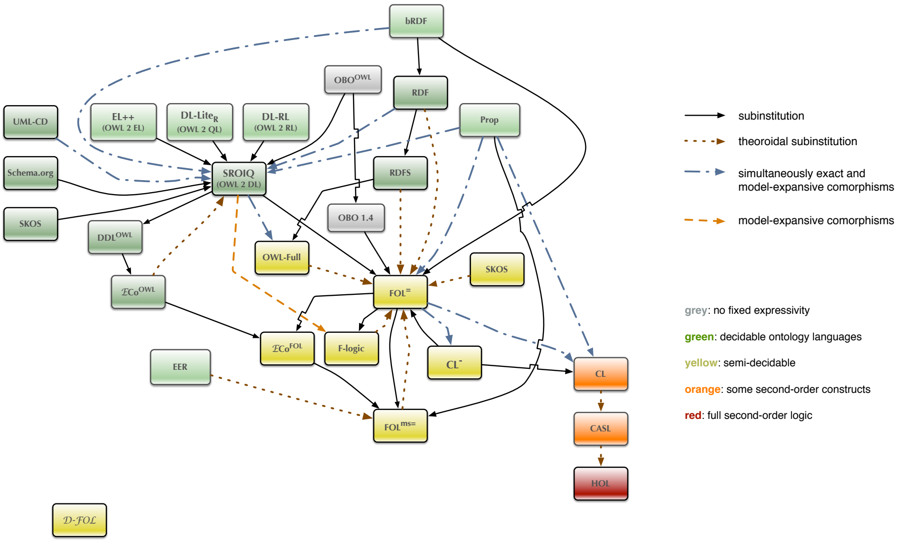


# Institution comorphisms (encodings)

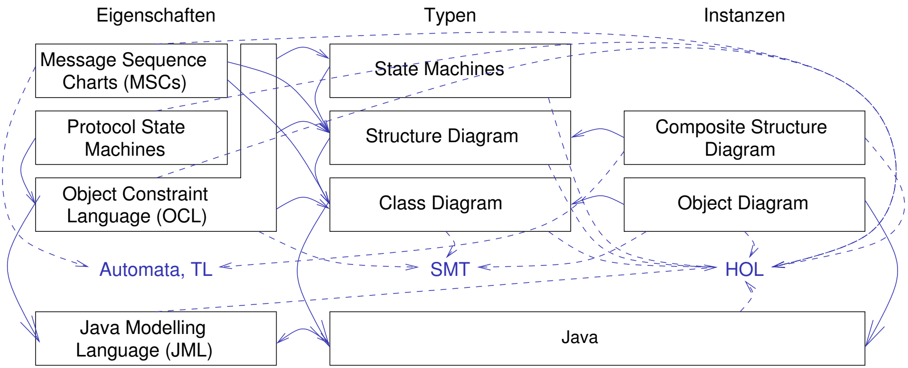
## Institution comorphisms



# Some $\mathcal{HLE}$ for ontologies



# Some $\mathcal{HLE}$ for UML and Java



# Heterogeneous structuring operations

*heterogeneous translation*: For any $\mathscr{I}$-OMS $O$, $\rho(O)$ is a OMS with:

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

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

*heterogeneous hiding*: For any $\mathscr{I}'$-OMS $O'$ and signature $\Sigma$ with $Sig[O'] = \Phi(\Sigma)$, $O'|^{\Sigma}_{\rho}$ is a OMS with:

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

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

# A heterogeneous proof calculus

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

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

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

# A heterogeneous proof calculus for refinement

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

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

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

# Heterogeneous completeness

## Theorem

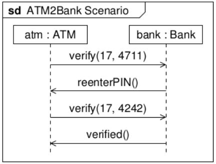
*For a heterogeneous logical environment $\mathscr{HLE} : \mathscr{G} \longrightarrow co\mathscr{INS}$ (with some of the institutions having entailment systems), the proof calculi for heterogeneous OMSs are sound for $\mathscr{I}^{\mathscr{HLE}}/\equiv$. If*

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

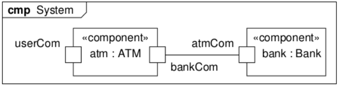
*the proof calculus for entailments between heterogeneous OMSs and sentences is complete over $\mathscr{I}^{\mathscr{HLE}}$.*

# Overview

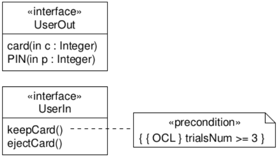
# Is a family of UML models consistent?



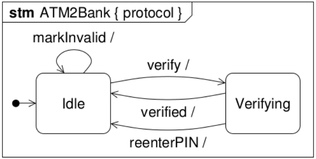

(a) Interaction



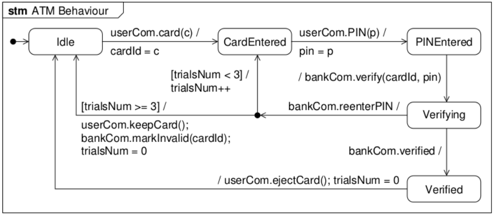

(b) Composite structure




(c) Interfaces

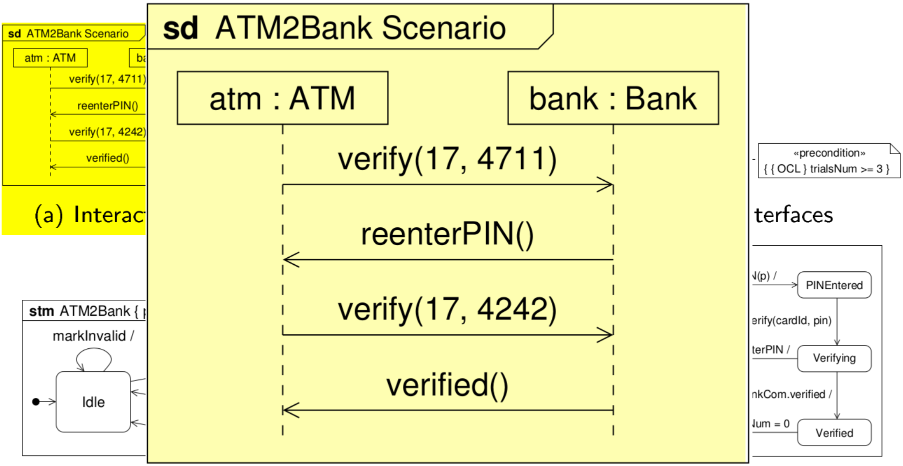



(d) Protocol state machine




(e) State machine

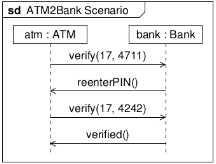
# Is a family of UML models consistent?



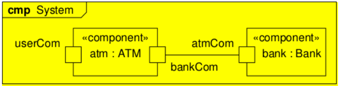

(d) Protocol state machine

(e) State machine

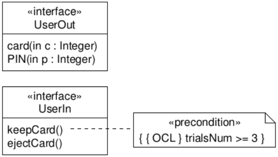
# Is a family of UML models consistent?



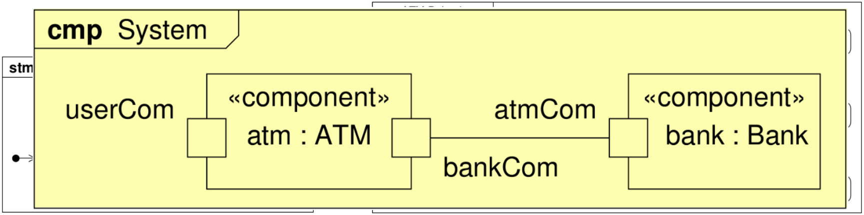

(a) Interaction




(b) Composite structure



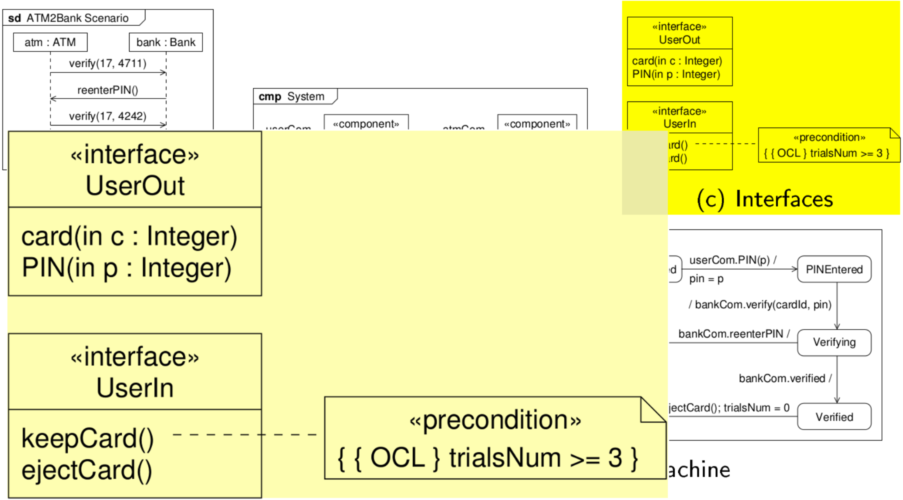

(c) Interfaces



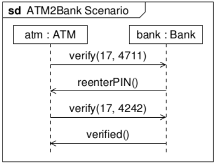

(d) Protocol state machine

(e) State machine

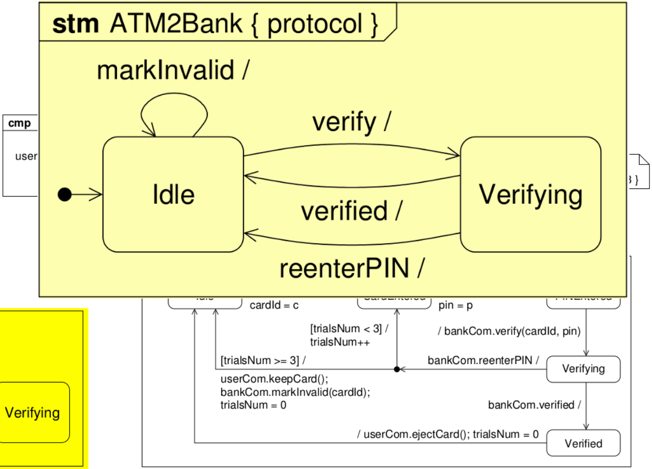
# Is a family of UML models consistent?

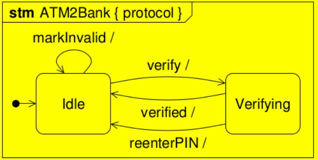


# Is a family of UML models consistent?




(a) Interaction

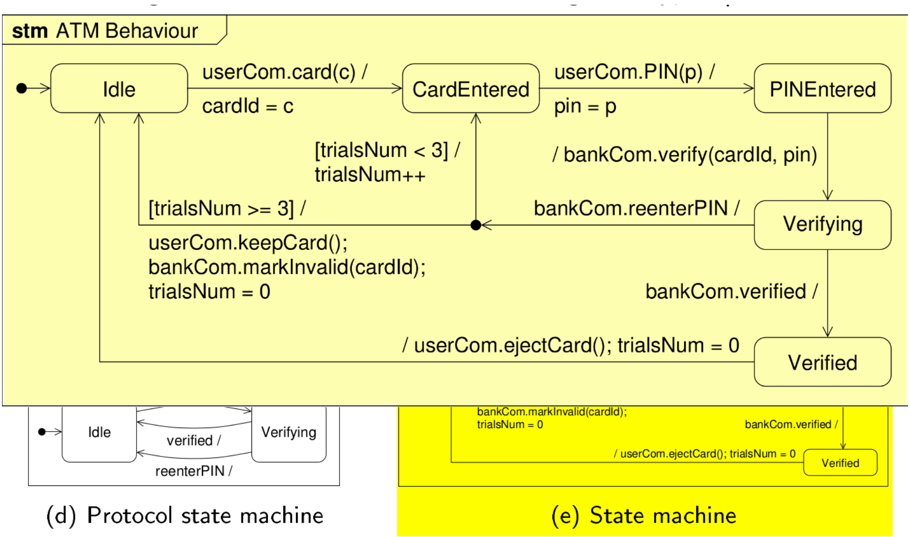



(d) Protocol state machine

(e) State machine

# Is a family of UML models consistent?




(d) Protocol state machine

(e) State machine

# UML multi-view consistency through DOL networks: sequence diagrams and class diagrams

```
model ATM2Bank_Scenario_cd =
 ATM2Bank_Scenario hide along sd2cd %% institution morphism
end

refinement r0 =
 sig { ATM2Bank_Scenario_cd } refined to User_Interface
end

refinement r1 =
 { User_Interface reveal sig { ATM2Bank_Scenario_cd } }
   refined to ATM2Bank_Scenario_cd
end
```

Semantics of **refined to**: Theory morphism

# State machines

```
model ATM_stm =
  User_Interface with translation cd2stm
then
  ATM_Behaviour
end

model Bank_stm =
  User_Interface with translation cd2stm
then
  Bank_Behaviour
end
```

Semantics of `with translation cd2stm`:
Translation along institution comorphism

# Composite Structure Diagram

```
model System =
  ATM_stm with translation stm2cmp with cid |-> atm
and
  Bank_stm with translation stm2cmp with cid |-> bank
then
  cmp
end
```

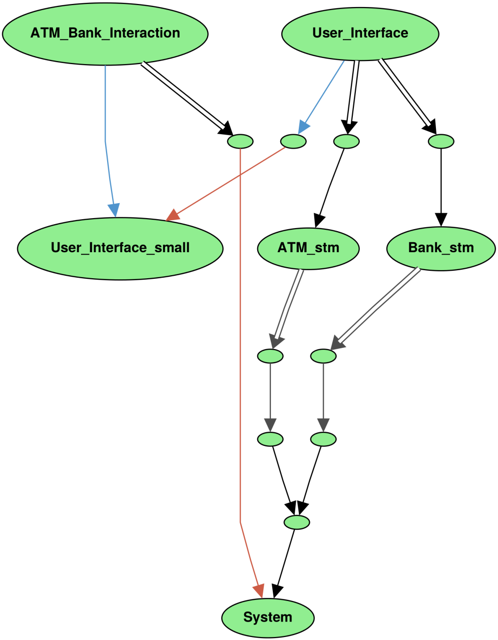
# State machines vs. sequence diagram

```
%% the sequence diagram can be realised by
%% the two state machines
%% as combined by the composite structure diagram
refinement r2 =
   ATM_Bank_Interaction refined to
     { System hide along cmp2sd }
end
```

# A network of OMS and mappings

```
%% multi-view consistency
network N = %consistent
  User_Interface, ATM_stm, Bank_stm, System,
  ATM_Bank_Interaction, r0, r1, r2
end
```

Realisation of a network = family of realisations, one for each node, that is compatible along the edges

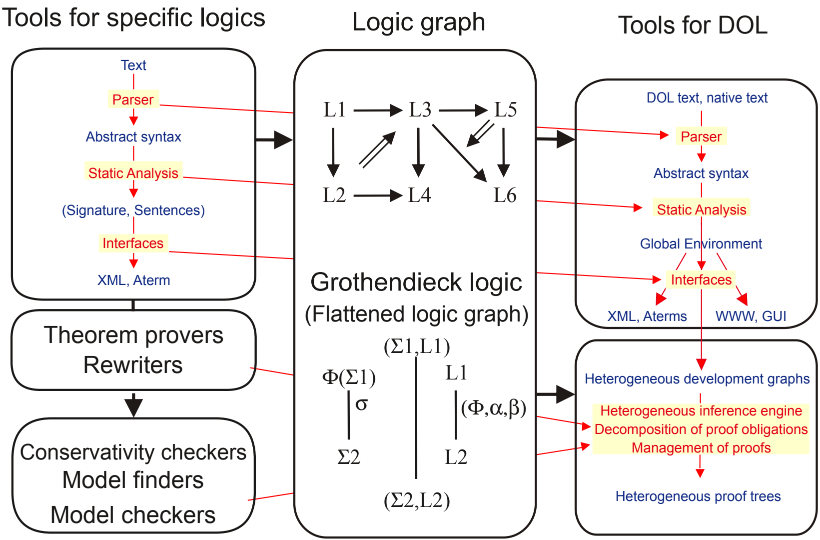

ATM_Bank_Interaction
User_Interface
User_Interface_small
ATM_stm
Bank_stm
System

# Overview

# Tool support: Heterogeneous Tool Set (Hets)

- available at `http://hets.eu`
- speaks DOL, propositional logic, OWL, CASL, Common Logic, QBF, modal logic, UML, MOF, QVT, and other languages
- analysis of native documents and DOL documents
- computation of colimits (combinations of networks)
- management of proof obligations
- interfaces to theorem provers, model checkers, model finders

# Architecture of the heterogeneous tool set Hets



# Tool support: Ontohub web portal and repository

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

web-based
: prototype available at `ontohub.org`

multi-logic
: speaks the same languages as Hets

multiple repositories
: ontologies can be organized in multiple repositories, each with its own management of editing and ownership rights,

Git interface
: version control of ontologies is supported via interfacing the Git version control system,

linked-data compliant
: one and the same URL is used for referencing an ontology, downloading it (for use with tools), and for user-friendly presentation in the browser.

# Overview

# Conclusions

- 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 is standardized at OMG
- institutions form the semantic basis of DOL
- you can help with joining the DOL discussion
  - see `dol-omg.org`
- open research problem:
  proof calculus and tool support for all of DOL

# Overview of DOL: Toolkit in Summary

1. OMS (ontologies, models, specifications)
   - basic OMS, written as-is (flattenable)
   - references to named OMS (by URL)
   - extensions, unions, translations (flattenable)
   - reductions, minimization, maximization (elusive)
   - approximations, module extractions, filterings (flattenable)
   - combinations of networks (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 networks, entailments, equivalences, . . .

# DOL Resources

- `http://omg.org/spec/DOL` Official OMG page for DOL
- `http://dol-omg.org` Central page for DOL
- `http://hets.eu` Analysis and Proof Tool Hets, speaking DOL
- `http://ontohub.org` Ontohub web platform, speaking DOL
- `http://ontohub.org/dol-examples` DOL examples
- `http://ontoiop.org` Initial standardization initiative

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

# OMS in DOL

```
OMS ::= ⟨I,Σ,Γ⟩ %% basic OMS in institution I
      | OMS then ⟨I,Σ,Γ⟩ %% extension of OMS
      | OMS and OMS  %% intersection of realisation classes
      | OMS with σ   %% σ: signature morphism
      | OMS with translation ρ %% ρ: institution comorphism
      | OMS hide Σ      | OMS reveal Σ
      | OMS hide along μ %% μ: institution morphism
      | OMS remove Σ    | OMS extract Σ
      | OMS forget Σ    | OMS keep Σ
      | OMS keep I
      | OMS reject Σ    | OMS select Σ
      | free { OMS }          %% initial semantics
      | minimize { OMS }      %% McCarthy's circumscription
      | logic l : { OMS }
      | combine Network       %% colimit of diagram
```

# Institutional model theory

In an arbitrary institution (possibly with some extra infrastructure), one can study:

- abstract quantifiers
- elementary diagrams
- elementary embeddings
- ultraproducts, Łos' theorem
- saturated models
- varieties, Birkhoff axiomatizability
- Craig interpolation, Robinson consistency, Beth definability
- Gödel's completeness theorem

R. Diaconescu. *Institution-independent Model Theory*. Birkhäuser Basel, 2008.

# Main references


- J. Goguen and R. Burstall. Institutions: abstract model theory for specification and programming. J.ACM 39, 1992.
- Till Mossakowski, Mihai Codescu, Fabian Neuhaus and Oliver Kutz. The Distributed Ontology, Modeling and Specification Language – DOL. In: A. Koslow and A. Buchsbaum, editors, The Road to Universal Logic, pages 489-520. Birkhäuser, 2015.
- Alexander Knapp and Till Mossakowski. Multi-view Consistency in UML. A Survey. In: R. Heckel and G. Taentzer, editors, Formal Methods for Software Systems, volume 10800, series Lecture Notes in Computer Science, pages 1-25. Springer Verlag, 2018.