

Software Specification and Development in Heterogeneous Environments

Andrzej Tarlecki

Institute of Informatics, Warsaw University

and

Institute of Computer Science, Polish Academy of Sciences

Warsaw, Poland

Two topics

- **Combining logical systems:
An institutional view**

joint work with **Till Mossakowski** and **Wiesiek Pawłowski**

(a while ago, hoped to be continued by Wiesiek...)

see also more recent work by **Carlos Caleiro**

- **Heterogeneous specification
and software development**

... continued by **Till Mossakowski** et al. (**HETS** for **CASL**)

for **CASL** see: LNCS 2900 & 2960

to be continued: **HiFi** initiative

Institutions

*Abstract model theory
for specification and programming*

Goguen & Burstall: 1980 → 1992

- a standard formalization of the concept of the underlying logical system for specification formalisms and most work on foundations of software specification and development from algebraic perspective;
- a formalization of the concept of a logical system for foundational studies:
 - truly abstract model theory
 - proof-theoretic considerations
 - building complex logical systems

Institution

- a category **Sign** of *signatures*
- a functor **Sen**: **Sign** \rightarrow **Set**
 - **Sen**(Σ) is the set of Σ -*sentences*, for $\Sigma \in |\mathbf{Sign}|$
- a functor **Mod**: **Sign**^{op} \rightarrow **Cat**
 - **Mod**(Σ) is the category of Σ -*models*, for $\Sigma \in |\mathbf{Sign}|$
- for each $\Sigma \in |\mathbf{Sign}|$, Σ -*satisfaction relation* $\models_{\Sigma} \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$

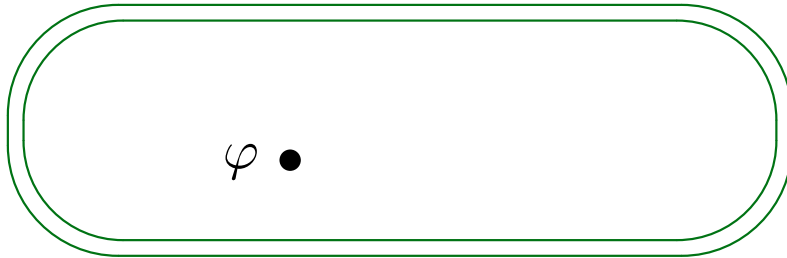
subject to the *satisfaction condition*:

$$M' |_{\sigma} \models_{\Sigma} \varphi \iff M' \models_{\Sigma'} \sigma(\varphi)$$

where $\sigma: \Sigma \rightarrow \Sigma'$ in **Sign**, $M' \in |\mathbf{Mod}(\Sigma')|$, $\varphi \in \mathbf{Sen}(\Sigma)$,
 $M' |_{\sigma}$ stands for $\mathbf{Mod}(\sigma)(M')$, and $\sigma(\varphi)$ for $\mathbf{Sen}(\sigma)(\varphi)$.

Institution: abstraction

Sen



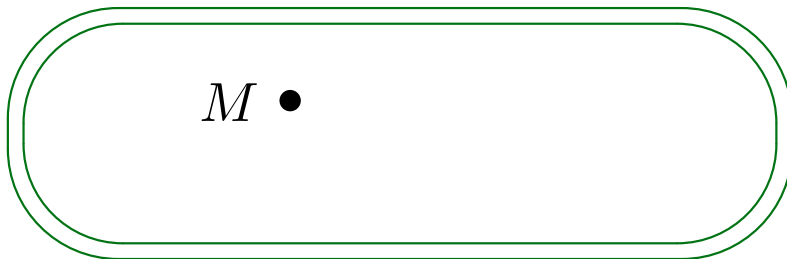
plus *satisfaction relation*:

$$M \models \varphi$$

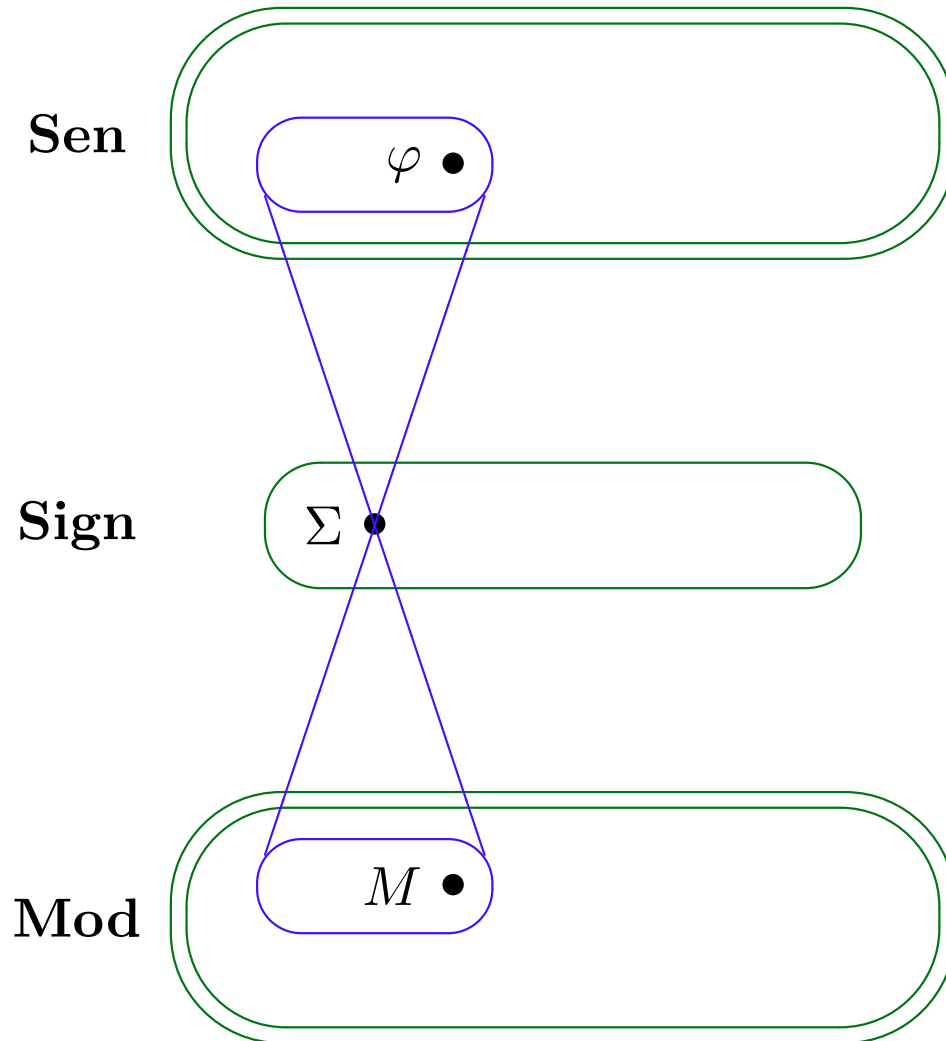
and so the usual Galois connection between classes of models and sets of sentences, with the standard notions induced ($Mod[\Phi]$, $Th[\mathcal{M}]$, $Th[\Phi]$, $\Phi \models \varphi$, etc).

- Also, possibly adding (sound) consequence: $\Phi \vdash \varphi$ (implying $\Phi \models \varphi$) to deal with proof-theoretic aspects.

Mod



Institution: first insight



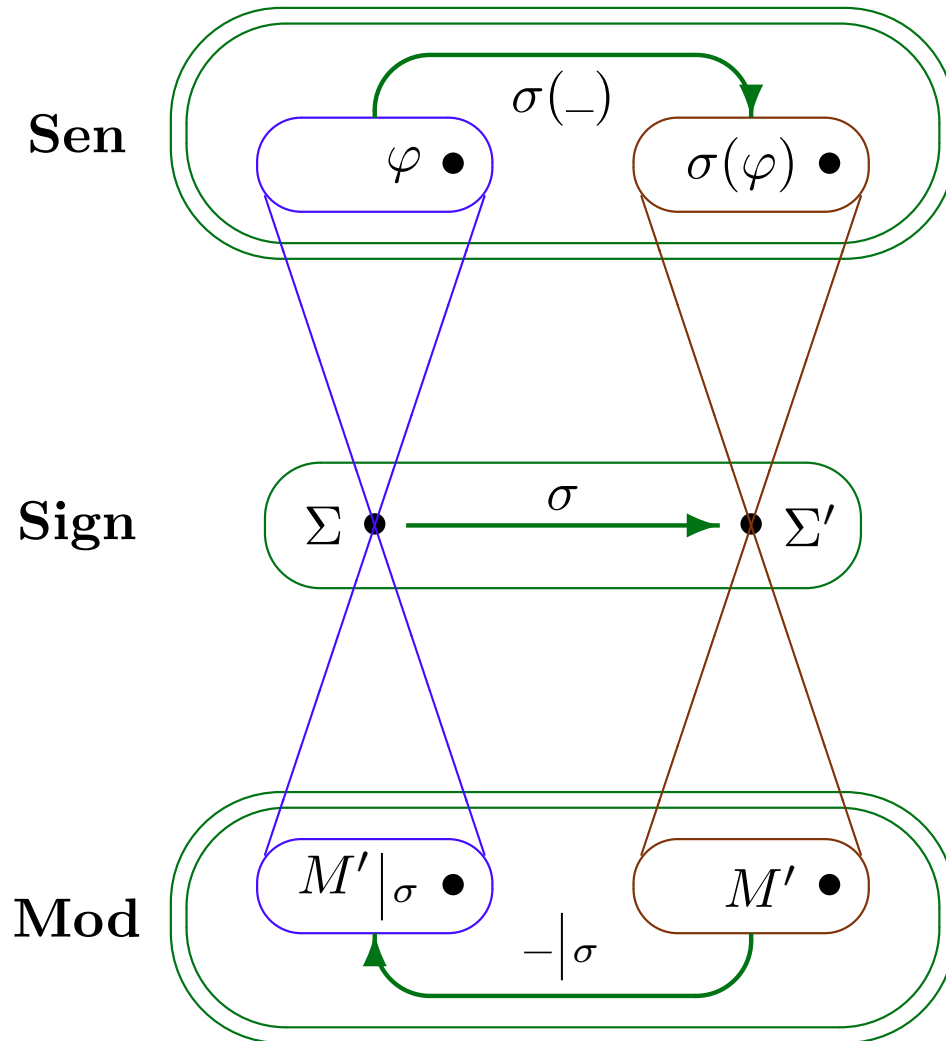
plus *satisfaction relation*:

$$M \models_{\Sigma} \varphi$$

and so, for each signature, the usual Galois connection between classes of models and sets of sentences, with the standard notions induced ($Mod_{\Sigma}[\Phi]$, $Th_{\Sigma}[\mathcal{M}]$, $Th_{\Sigma}[\Phi]$, $\Phi \models_{\Sigma} \varphi$, etc).

- Also, possibly adding (sound) consequence: $\Phi \vdash_{\Sigma} \varphi$ (implying $\Phi \models_{\Sigma} \varphi$) to deal with proof-theoretic aspects.

Institution: key insight



imposing the *satisfaction condition*:

$$M' \models_{\Sigma'} \sigma(\varphi) \text{ iff } M' |_{\sigma} \models_{\Sigma} \varphi$$

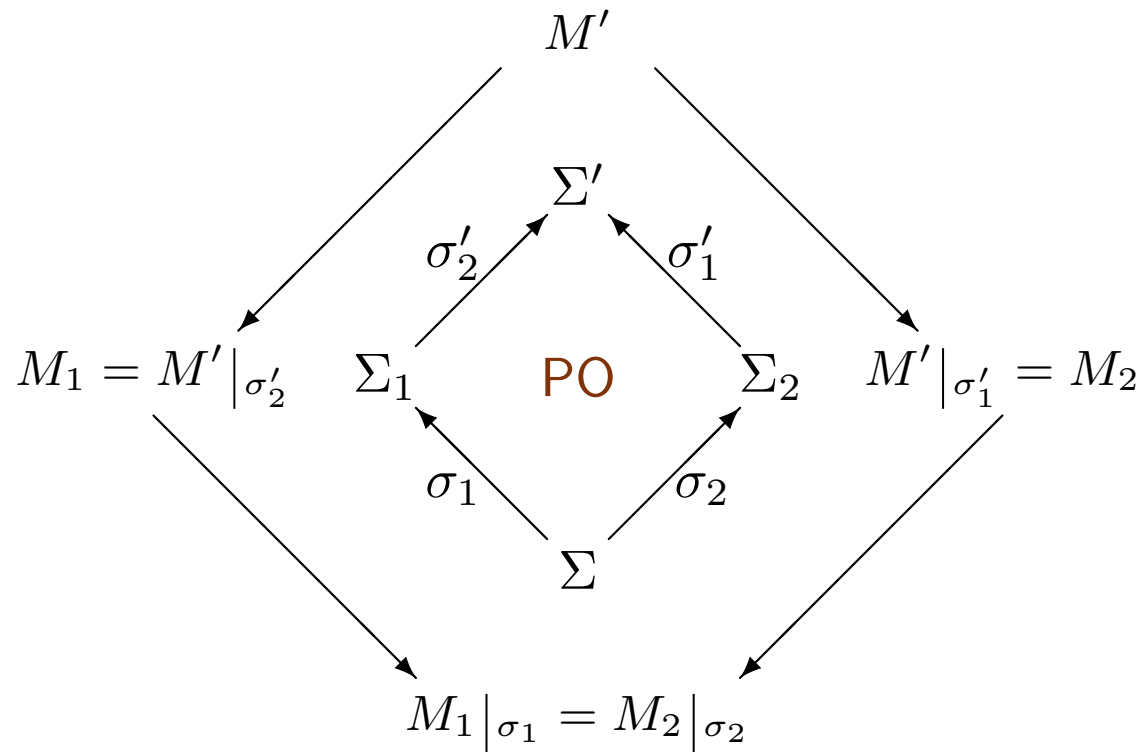
*Truth is invariant
under change of notation
and independent of
any additional symbols around*

Examples

- typical, not so typical and perhaps unexpected examples abound.

WORK WITH INSTITUTIONS

Amalgamation

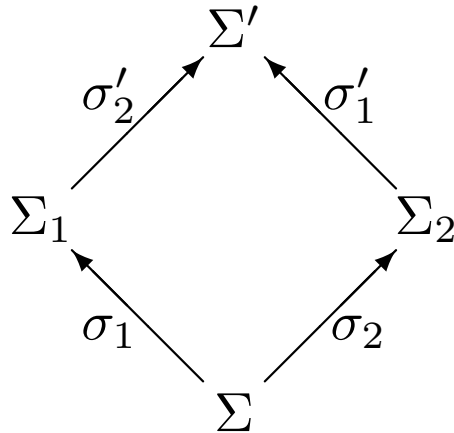


Given $M_1 \in |\mathbf{Mod}(\Sigma_1)|$ and $M_2 \in |\mathbf{Mod}(\Sigma_2)|$ with $M_1|_{\sigma_1} = M_2|_{\sigma_2}$, there is a unique $M' \in |\mathbf{Mod}(\Sigma')|$ with $M'|_{\sigma'_1} = M_2$ and $M'|_{\sigma'_2} = M_1$.

Fact: *The institution of many-sorted first-order logic admits amalgamation.*

Interpolation

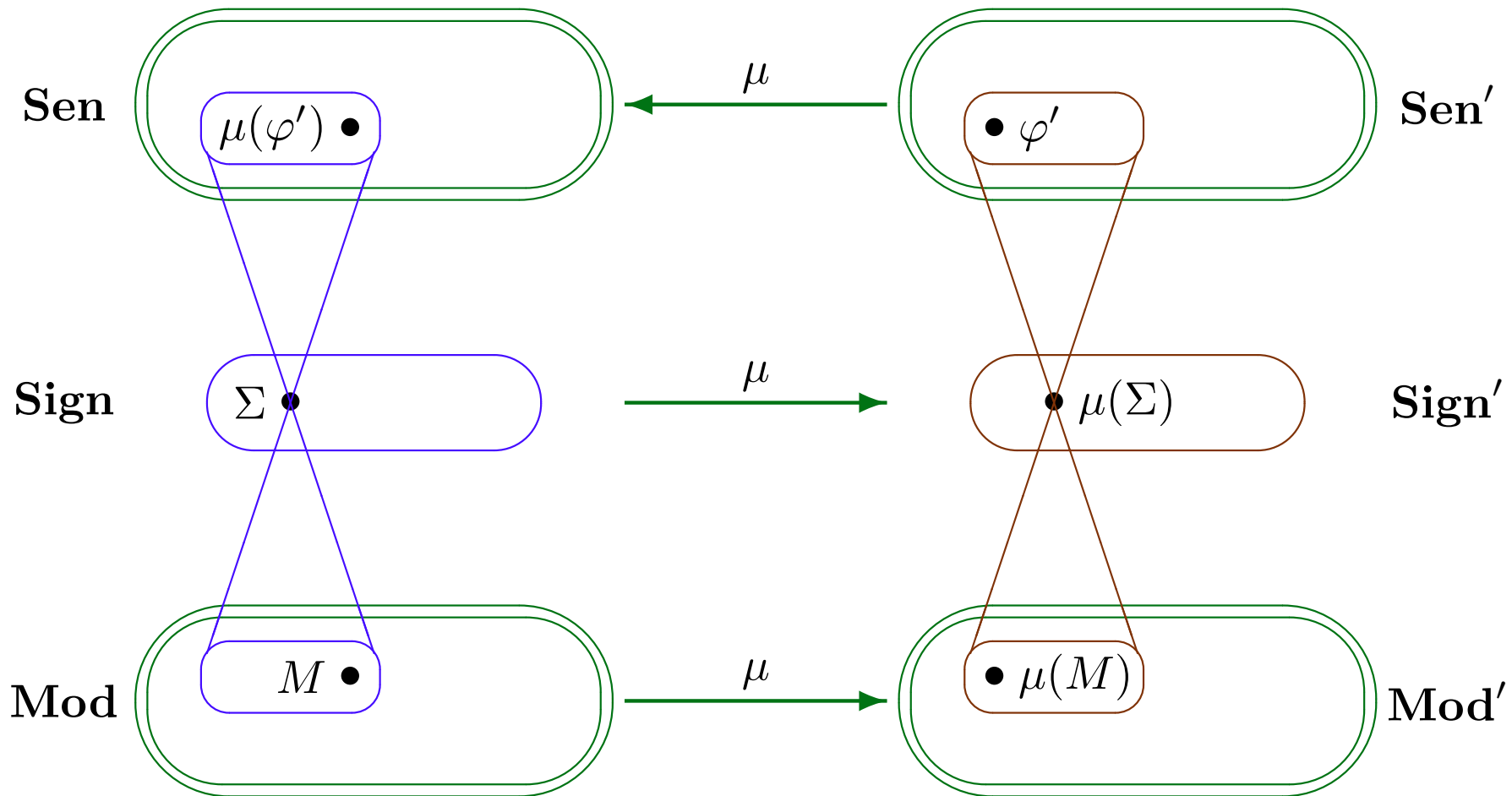
I has the *interpolation property* for a pushout in **Sign**



if for all $\varphi_1 \in \mathbf{Sen}(\Sigma_1)$ and $\varphi_2 \in \mathbf{Sen}(\Sigma_2)$ such that $\sigma'_2(\varphi_1) \models_{\Sigma'} \sigma'_1(\varphi_2)$ there is $\theta \in \mathbf{Sen}(\Sigma)$ such that $\varphi_1 \models_{\Sigma_1} \sigma_1(\theta)$ and $\sigma_2(\theta) \models_{\Sigma_2} \varphi_2$.

Fact: *The institution of many-sorted first-order logic has the interpolation property for the pushout as above provided that at least one of the two morphisms σ_1, σ_2 is injective on sorts.*

Institution morphism: $\mu: \mathbf{I} \longrightarrow \mathbf{I}'$



with the *satisfaction condition* lurking again:

$$M \models \mu(\varphi) \text{ iff } \mu(M) \models' \varphi'$$

Institution morphism

An *institution morphism* $\mu: \mathbf{I} \longrightarrow \mathbf{I}'$ consists of:

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

subject to the following *satisfaction condition*:

$$M \models_{\Sigma} \mu(\varphi) \text{ iff } \mu(M) \models'_{\mu(\Sigma)} \varphi'$$

where $\Sigma \in |\mathbf{Sign}|$, $M \in |\mathbf{Mod}(\Sigma)|$ and $\varphi' \in \mathbf{Sen}'(\mu(\Sigma))$.

Moving between institutions: a taxonomy of maps

<i>morphisms</i> μ	$\text{Sen} \longleftarrow \text{Sen}'$ $\text{Sign} \longrightarrow \text{Sign}'$ $\text{Mod} \longrightarrow \text{Mod}'$	<i>semi-morphisms</i> μ	$\text{Sen} \longrightarrow \text{Sen}'$ $\text{Sign} \longrightarrow \text{Sign}'$ $\text{Mod} \longrightarrow \text{Mod}'$
<i>comorphisms</i> ρ	$\text{Sen} \longrightarrow \text{Sen}'$ $\text{Sign} \longrightarrow \text{Sign}'$ $\text{Mod} \longleftarrow \text{Mod}'$	<i>semi-comorphisms</i> ρ	$\text{Sen} \longrightarrow \text{Sen}'$ $\text{Sign} \longrightarrow \text{Sign}'$ $\text{Mod} \longleftarrow \text{Mod}'$
<i>forward morphisms</i>	$\text{Sen} \longrightarrow \text{Sen}'$ $\text{Sign} \longrightarrow \text{Sign}'$ $\text{Mod} \longrightarrow \text{Mod}'$		
<i>forward comorphisms</i>	$\text{Sen} \longleftarrow \text{Sen}'$ $\text{Sign} \longrightarrow \text{Sign}'$ $\text{Mod} \longleftarrow \text{Mod}'$		

plus *theoroidal* versions,
 plus *weak* versions, plus ...

Putting institutions together

Fact: *The category \mathcal{INS} of institutions and institution morphisms is complete.*

- Limits in \mathcal{INS} : a rudimentary way of combining institutions linked by institution morphisms to capture how one institution is built over another.
- This is in contrast with the *Grothendieck institution* built over the same diagram, which just puts the institutions involved next to each other, with additional signature morphisms induced by institution morphisms.

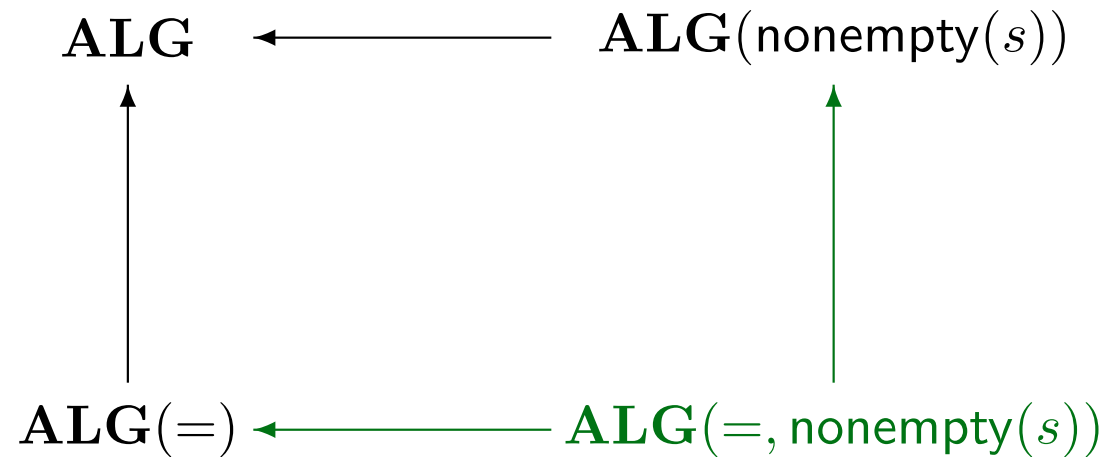
Limits of limits

- In general, limits in \mathcal{INS} do not preserve cocompleteness of the category of signatures, amalgamability, interpolation, etc.

Fact: *Given a diagram in \mathcal{INS} of institutions that have cocomplete categories of signatures and admit amalgamation (are **exact**), if its limit has a cocomplete category of signatures and the limit projections have cocontinuous signature functors then the limit admits amalgamation (is **exact**).*

Fact: *Given a diagram in \mathcal{INS} of institutions that have cocomplete skeletal categories of signatures and admit amalgamation (are **exact**) with institution morphisms that have cocontinuous signature functors, its limit has a (skeletal) cocomplete category of signatures and admits amalgamation (is **exact**).*

Feature interference



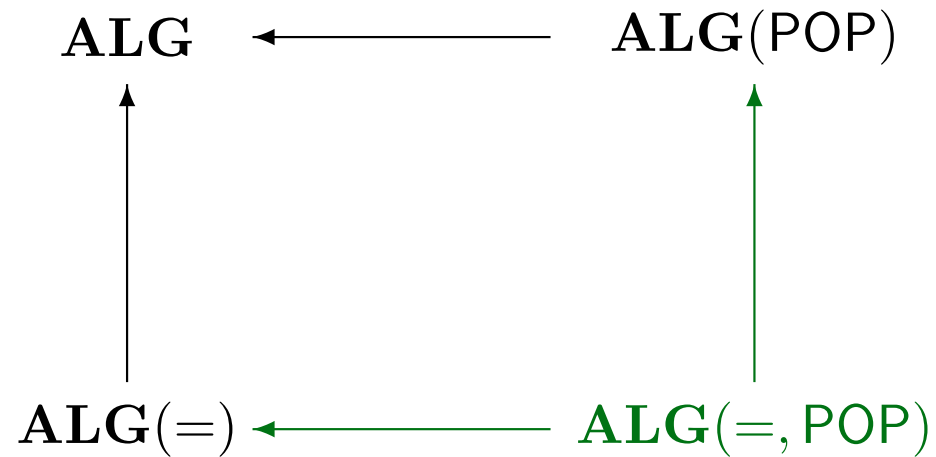
In **ALG(=)**:

$$\forall x:s \cdot a = b:s' \not\models a = b:s'$$

In **ALG(=, nonempty(s))**:

$$\{\forall x:s \cdot a = b:s', \text{nonempty}(s)\} \models a = b:s'$$

No feature interleaving



In $ALG(=, POP)$:

no equations between terms with POP's.

Parchments

Formal structures to present institutions

Goguen & Burstall: 1985

- We will work with a “model-theoretic” version (rather than relying on super-large “universal” signatures and semantic structures).

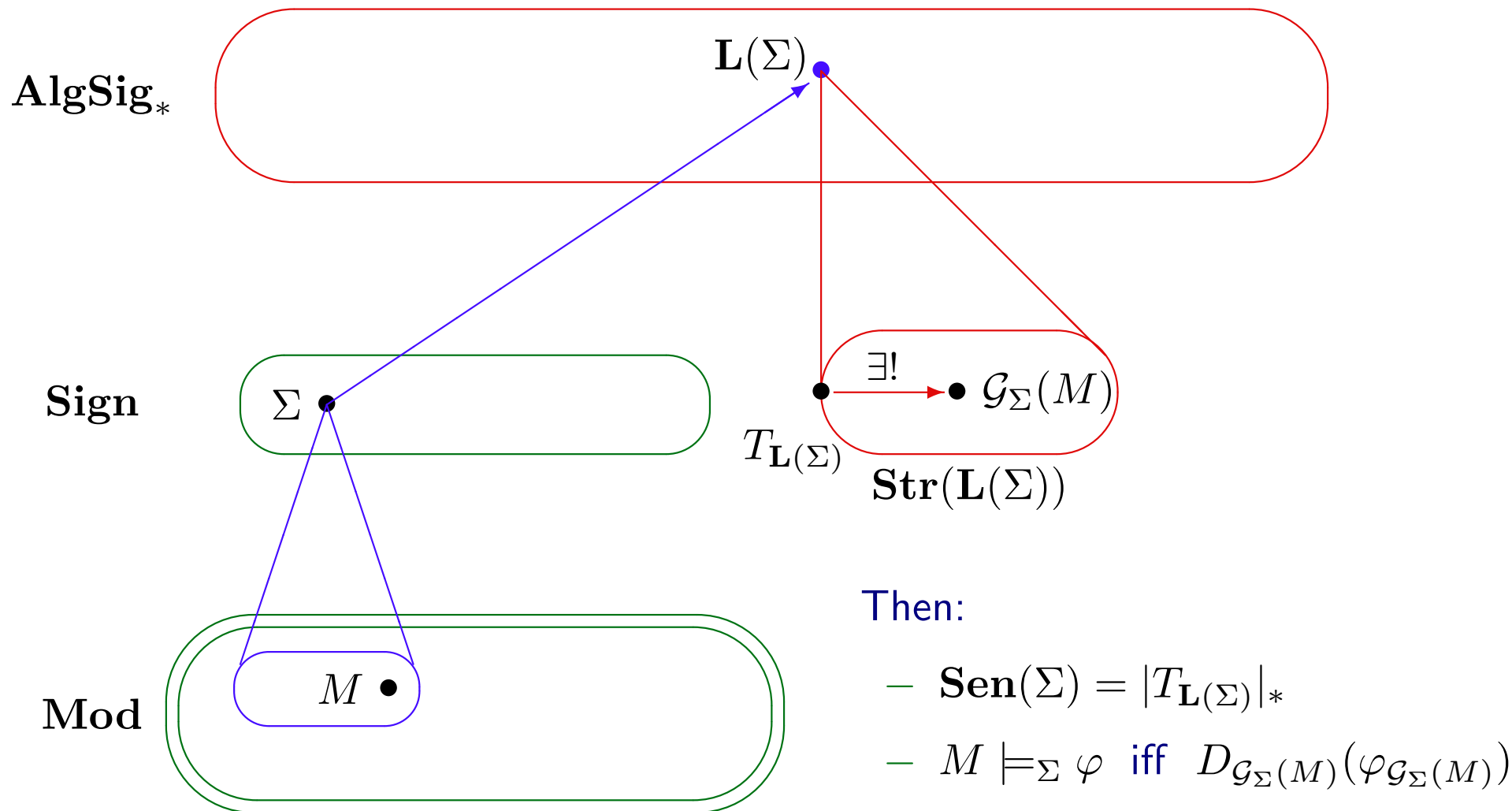
M-T Parchment

A *model-theoretic parchment* $\mathbf{P} = (\mathbf{Sign}, \mathbf{Mod}, \mathbf{L}, \mathcal{G})$ consists of

- a category \mathbf{Sign} of signatures,
- a functor $\mathbf{Mod}: \mathbf{Sign}^{op} \rightarrow \mathbf{Cat}$ (as for institutions),
- a functor $\mathbf{L}: \mathbf{Sign} \rightarrow \mathbf{AlgSig}_*$ giving the abstract syntax of sentences,
- for $\Sigma \in |\mathbf{Sign}|$ and $M \in |\mathbf{Mod}(\Sigma)|$, an $\mathbf{L}(\Sigma)$ -structure $\mathcal{G}_\Sigma(M) \in |\mathbf{Str}(\mathbf{L}(\Sigma))|$ which determines semantic evaluation of Σ -sentences in M ,
- for $\sigma: \Sigma \rightarrow \Sigma'$ and $M' \in |\mathbf{Mod}(\Sigma')|$, an $\mathbf{L}(\Sigma)$ -homomorphism $\mathcal{G}_\sigma(M'): \mathcal{G}_\Sigma(M'|_\sigma) \rightarrow \mathcal{G}_{\Sigma'}(M')|_{\mathbf{L}(\sigma)}$, compositional in σ ,

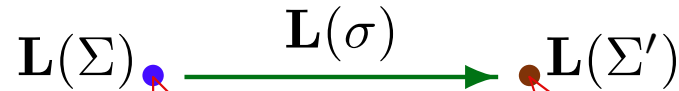
where: \mathbf{AlgSig}_* is the category of many-sorted signatures with a distinguished “logical” sort $*$ and a predicate D on $*$.

M-T Parchment

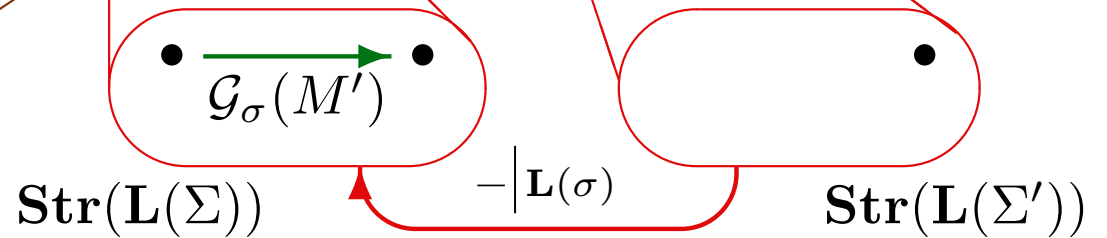
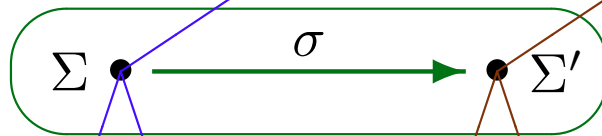


M-T Parchment

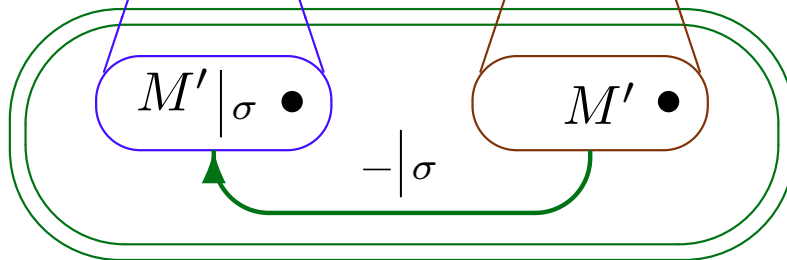
AlgSig_{*}



Sign



Mod



To ensure the satisfaction condition, require parchments to be *logical*:

$$\mathcal{G}_\sigma(M') : \mathcal{G}_\Sigma(M'|_\sigma) \rightarrow \mathcal{G}_{\Sigma'}(M')|_{L(\sigma)}$$

be a closed injective homomorphism.

Parchment morphism

A (model-theoretic) *parchment morphism* $\pi: \mathbf{P} \longrightarrow \mathbf{P}'$ consists of:

- a functor $\pi: \mathbf{Sign} \rightarrow \mathbf{Sign}'$,
- a natural transformation $\pi: \mathbf{Mod} \rightarrow (\pi)^{op};\mathbf{Mod}'$,
- a natural transformation $\pi: \pi;\mathbf{L}' \rightarrow \mathbf{L}$, and
- for $\Sigma \in |\mathbf{Sign}|$ and $M \in |\mathbf{Mod}(\Sigma)|$, an $\mathbf{L}'(\pi(\Sigma))$ -homomorphism

$$g_{\Sigma, M}: \mathcal{G}'_{\pi(\Sigma)}(\pi(M)) \rightarrow \mathcal{G}_{\Sigma}(M) \Big|_{\pi_{\Sigma}}$$

subject to the routine (though not easy) naturality condition: for $\sigma: \Sigma_1 \rightarrow \Sigma_2$ and $M_2 \in |\mathbf{Mod}(\Sigma_2)|$,

$$g_{\Sigma_1, M_2} \Big|_{\sigma}; \mathcal{G}_{\sigma}(M_2) \Big|_{\pi_{\Sigma_2}} = \mathcal{G}'_{\pi(\sigma)}(\pi(M_2)); g_{\Sigma_2, M_2} \Big|_{\mathbf{L}(\pi(\sigma))}$$

(which makes g a 2-natural transformation...).

From parchments to institutions

Fact: *There is a functor*

$$\mathcal{I}: \mathcal{LPAR} \rightarrow \mathcal{INS}$$

that maps logical (model-theoretic) parchments to institutions and logical (model-theoretic) parchment morphisms to institution morphisms.

Indeed, logical parchments smoothly present institutions!

BUT: the construction only “nearly works” for arbitrary (non-logical) parchments and parchment morphisms — the satisfaction condition may fail.

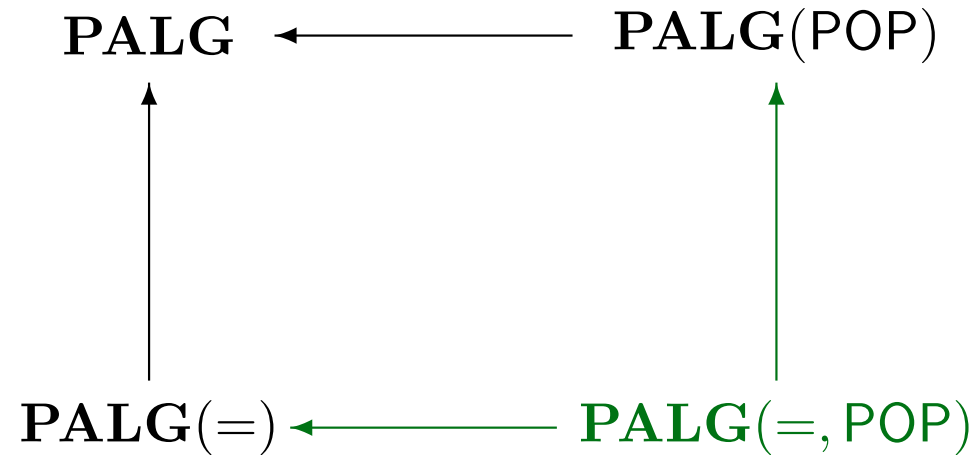
Putting parchments together

Fact: *The category \mathcal{PAR} of parchments and their morphisms is complete.*

Fact: *The category \mathcal{LPAR} of logical parchments and their logical morphisms is **not** complete.*

NO MIRACLES!

Feature interleaving



In **PALG(=, POP)**:

*equations between terms with POP's arise as expected
but generate new free "logical" values*

Adjusting limit parchments

Typically: new feature interleavings *freely generate new semantic values*, including those in the sort $*$, thus indicating where essential semantic design decisions are necessary for the logic combination.

Technique: choose (minimal) *congruences* on the semantic structures $\mathcal{G}_\Sigma(M)$ to glue the new “logical” values with the standard (old) ones, so that the limit parchment and the limit projection morphisms are transformed to become logical.

Cheating: work with logical encodings (comorphisms) of logical systems into a universal, rich logical framework; *parchment comorphisms lift to limits* and then determine the congruences to divide the semantic structures by.

Further work

Done:

- context institutions: introducing free variables and quantification; their morphisms;
- parchments for context institutions; their morphisms;
- results and techniques (non-trivially) carry over from the no-free-variables case.

This opens a way for introducing and studying proof systems:

- adding schematic proof rules;
- putting proof systems together, hand-in-hand with institution and parchment combination;
- results to come?

Now, at last, to:

Heterogeneous specification and software development

Specifications in an arbitrary institution

Adopt the model-theoretic view of specifications

The meaning of any specification SP built over \mathbf{I} is given by:

- its *signature* $Sig[SP] \in |\mathbf{Sign}|$, and
- a class of its *models* $Mod[SP] \subseteq |\mathbf{Mod}(Sig[SP])|$.

This yields the usual notions

- semantic equivalence: $SP_1 \equiv SP_2$,
- semantic consequence: $SP \models \varphi$,
- theory of a specification: $Th[SP]$, etc.

Structured specifications

Basic specification:

$\langle \Sigma, \Phi \rangle$

— for $\Sigma \in |\mathbf{Sign}|$ and $\Phi \subseteq \mathbf{Sen}(\Sigma)$:

$$\mathit{Sig}[\langle \Sigma, \Phi \rangle] = \Sigma$$

captures basic properties

$$\mathit{Mod}[\langle \Sigma, \Phi \rangle] = \mathit{Mod}[\Phi]$$

Union:

$SP_1 \cup SP_2$

— for SP_1 and SP_2 with $\mathit{Sig}[SP_1] = \mathit{Sig}[SP_2]$:

$$\mathit{Sig}[SP_1 \cup SP_2] = \mathit{Sig}[SP_1]$$

combines the constraints imposed

$$\mathit{Mod}[SP_1 \cup SP_2] = \mathit{Mod}[SP_1] \cap \mathit{Mod}[SP_2]$$

Translation:

$\sigma(SP)$

— for any SP and $\sigma: \mathit{Sig}[SP] \rightarrow \Sigma'$:

$$\mathit{Sig}[\sigma(SP)] = \Sigma'$$

renames and introduces new components

$$\mathit{Mod}[\sigma(SP)] = \{M' \in |\mathbf{Mod}(\Sigma')| \mid M'|_{\sigma} \in \mathit{Mod}[SP]\}$$

Hiding:

$SP'|_{\sigma}$

— for any SP' and $\sigma: \Sigma \rightarrow \mathit{Sig}[SP']$:

$$\mathit{Sig}[SP'|_{\sigma}] = \Sigma$$

hides auxiliary components

$$\mathit{Mod}[SP'|_{\sigma}] = \{M'|_{\sigma} \mid M' \in \mathit{Mod}[SP']\}$$

Proving semantic consequence

$$\frac{\mathbf{normal_form}(SP) = \langle \Sigma_{all}, \Phi_{all} \rangle \Big|_{\sigma_{res}} \quad \Phi_{all} \vdash_{\Sigma_{all}} \sigma_{res}(\varphi)}{SP \vdash \varphi}$$

This is sound and complete when the category of signatures has pushouts, the institution admits amalgamation (then the normal forms as above can be constructed), and $\vdash_{\Sigma_{all}}$ is sound and complete, but:

This is a bad way!

- lack of compositionality
- no use of the specification structure
- typically, Φ_{all} is HUGE
- no help in proof search

Compositional proof system

$$\begin{array}{c}
 \frac{\varphi \in \Phi}{\langle \Sigma, \Phi \rangle \vdash \varphi} \quad \frac{SP_1 \vdash \varphi}{SP_1 \cup SP_2 \vdash \varphi} \quad \frac{SP_2 \vdash \varphi}{SP_1 \cup SP_2 \vdash \varphi} \\
 \\
 \frac{SP \vdash \varphi}{\sigma(SP) \vdash \sigma(\varphi)} \quad \frac{SP' \vdash \sigma(\varphi)}{SP' \upharpoonright_{\sigma} \vdash \varphi}
 \end{array}$$

Plus a *structural rule* (relying on a proof system for the underlying institution to derive $\Phi \vdash \varphi$):

$$\frac{\text{for } i \in J, SP \vdash \varphi_i \quad \{\varphi_i\}_{i \in J} \vdash \varphi}{SP \vdash \varphi}$$

Soundness & completeness

$$SP \vdash \varphi \implies SP \models \varphi$$

Fact: *If the category of signatures has pushouts, the institution admits amalgamation and interpolation, and has a sound and complete proof system to derive $\Phi \vdash \varphi$, then*

$$SP \vdash \varphi \iff SP \models \varphi$$

In general: there is *no* sound and complete *compositional* proof system for semantic consequence for structured specifications.

- Perhaps a right balance between compositionality and flexibility:

development graphs

Heterogeneous specifications

Translation: introduces new structure to specification models, following an institution semi-comorphism $\rho: \mathbf{I} \rightarrow \mathbf{I}'$; for any \mathbf{I} -specification SP ,

$$\rho(SP)$$

is an \mathbf{I}' -specification with $Sig[\rho(SP)] = \rho(Sig[SP])$ and $Mod[\rho(SP)] = \{M' \in |\mathbf{Mod}'(\rho(Sig[SP]))| \mid \rho(M') \in Mod[SP]\}$.

Hiding: hides extra structure of specification models, following an institution semi-morphism $\mu: \mathbf{I}' \rightarrow \mathbf{I}$; for any \mathbf{I}' -specification SP' ,

$$SP' \mid_{\mu}$$

is an \mathbf{I} -specification with $Sig[SP' \mid_{\mu}] = \mu(Sig[SP'])$ and $Mod[SP' \mid_{\mu}] = \{\mu(M') \mid M' \in Mod[SP']\}$.

Proof system for heterogeneous specifications

Given institution comorphism $\rho: \mathbf{I} \rightarrow \mathbf{I}'$ and institution morphism $\mu: \mathbf{I}' \rightarrow \mathbf{I}$:

$$\frac{SP \vdash^{\mathbf{I}} \varphi}{\rho(SP) \vdash \rho(\varphi)} \quad \frac{SP' \vdash^{\mathbf{I}'} \mu(\varphi)}{SP' \upharpoonright_{\mu} \vdash \varphi}$$

semi-(co)morphisms are not enough

Soundness: No problem.

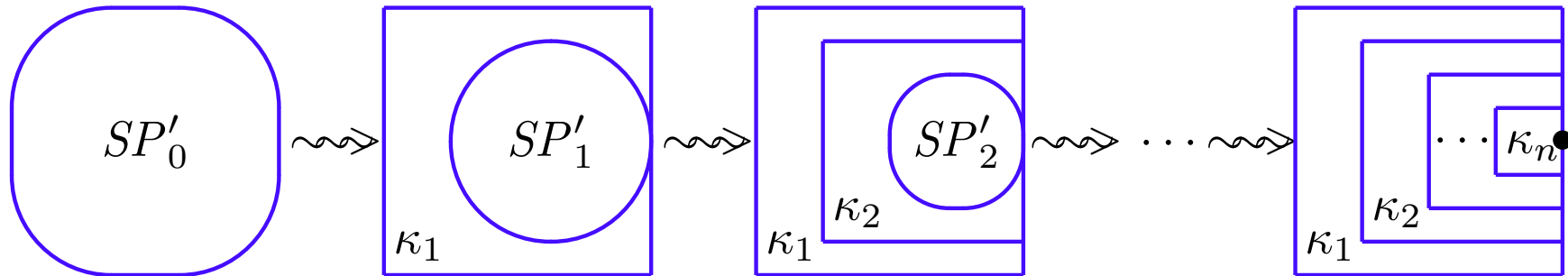
Completeness: Heterogeneous specifications over an institution diagram are usual specifications in the Grothendieck institution; completeness conditions can be derived from this...

Details can be provided on demand

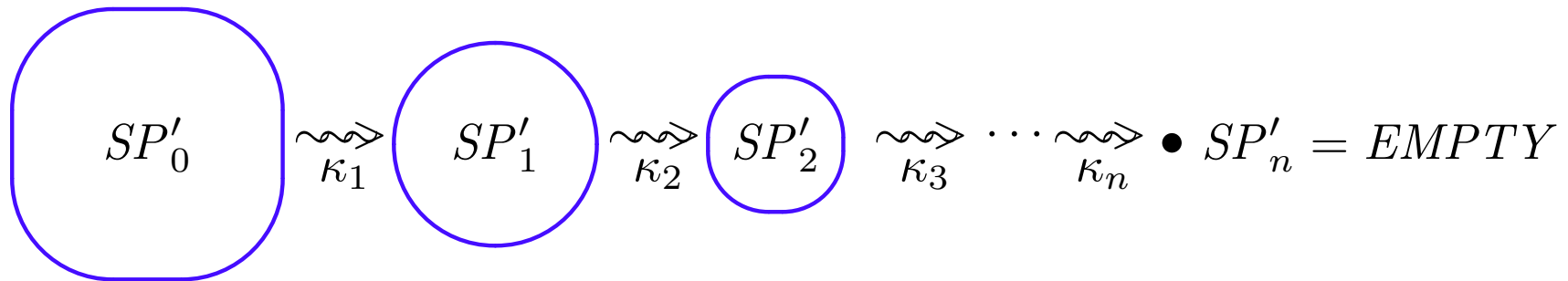
Stepwise software development

$$SP_0 \rightsquigarrow SP_1 \rightsquigarrow \dots \rightsquigarrow SP_n$$

In practice, some parts will get fixed on the way:



Keep them apart from whatever is really left for implementation:



Constructor refinements

$$SP' \overset{\kappa}{\rightsquigarrow} SP$$

Means:

$$\kappa(\text{Mod}[SP]) \subseteq \text{Mod}[SP']$$

where

$$\kappa: |\mathbf{Mod}(\text{Sig}[SP])| \rightarrow |\mathbf{Mod}(\text{Sig}[SP'])|$$

is a *constructor*:

Intuitively: *generic module* (SML *functor*)

Semantically: function between model classes

Heterogeneous software development

Inter-institutional constructors needed!

IDEA: Use the model component of institution semi-(co)morphisms

For institution semi-morphism $\mu: \mathbf{I}' \rightarrow \mathbf{I}$, \mathbf{I}' -specification SP' and \mathbf{I} -specification SP with $Sig[SP] = \mu(Sig[SP'])$, SP' is a constructor refinement of SP via μ

$$SP \overset{\mu}{\rightsquigarrow} SP'$$

provided that $\mu(\text{Mod}[SP']) \subseteq \text{Mod}[SP]$.

For instance:

Use $\mu: \mathbf{SML} \rightarrow \mathbf{CASL}$ to implement CASL specifications by SML programs.

Branching refinement steps

$$SP \rightsquigarrow BR \left\{ \begin{array}{l} SP_1 \\ \vdots \\ SP_n \end{array} \right.$$

Branching step BR involves a “linking procedure” (n -argument constructor)

$$\kappa_{BR}: |\mathbf{Mod}(Sig[SP_1])| \times \cdots \times |\mathbf{Mod}(Sig[SP_n])| \rightarrow |\mathbf{Mod}(Sig[SP])|$$

and we require

$$\frac{M_1 \in Mod[SP_1] \quad \cdots \quad M_n \in Mod[SP_n]}{\kappa_{BR}(M_1, \dots, M_n) \in Mod[SP]}$$

CASL architectural specifications

CASL provides an explicit way to write down the organizational specification such a branching amounts to:

$$\begin{array}{l} \text{arch spec } BR = \\ \text{units } U_1 : SP_1 \\ \quad \dots \\ \quad U_n : SP_n \\ \text{result } \kappa_{BR}(U_1, \dots, U_n) \end{array}$$

Moreover:

- units may be generic (parametrized modules, SML functors), but always are declared with their specifications
- CASL provides a rich collection of combinators to define κ_{BR} and various additional ways to *define* units

Heterogeneous architectural specifications

... given the above, not much to write ...

Just: *extend the repertoire of basic constructors by inter-institutional constructors:*
model components of institution semi-morphisms.

Final remarks

*A semantic view of heterogeneous logical environment
for software specification and programming emerges:
a diagram of institutions*

Sample further work:

- keep building up the environment of relevant institutions and (semi-)(co)morphisms between them;
- develop proof theoretic links between institutions linked semantically;
- detach specifications from a particular target institution (inter-institutional specification diagrams?);
- develop programming links between “programming” institutions linked semantically (*PĄCZKI* and *KODKI*).