Software specification in CASL - The Common Algebraic Specification Language

Till Mossakowski, Lutz Schröder

January 2007

SFB/TR 8 SPATIAL COGNITION

Semantics of CASL basic specifications (recalled)

The CASL logic (institution)

- Signatures: a signature provides the vocabulary

The CASL logic (institution)

- Signatures: a signature provides the vocabulary
- Signature morphisms: for extending and renaming signatures

The CASL logic (institution)

- Signatures: a signature provides the vocabulary
- Signature morphisms: for extending and renaming signatures
- Models: interpret the vocabulary of a signature with mathematical objects (sets, functions, relations)

The CASL logic (institution)

- Signatures: a signature provides the vocabulary
- Signature morphisms: for extending and renaming signatures
- Models: interpret the vocabulary of a signature with mathematical objects (sets, functions, relations)
- Sentences (formulae): for axiomatizing models denote true or false in a given model

The CASL logic (institution)

- Signatures: a signature provides the vocabulary
- Signature morphisms: for extending and renaming signatures
- Models: interpret the vocabulary of a signature with mathematical objects (sets, functions, relations)
- Sentences (formulae): for axiomatizing models denote true or false in a given model
- Terms: parts of sentences, denote data values

The CASL logic (institution)

- Signatures: a signature provides the vocabulary
- Signature morphisms: for extending and renaming signatures
- Models: interpret the vocabulary of a signature with mathematical objects (sets, functions, relations)
- Sentences (formulae): for axiomatizing models denote true or false in a given model
- Terms: parts of sentences, denote data values
- Satisfaction of sentences in models

CASL many-sorted signatures

- a set S of sorts,

CASL many-sorted signatures

- a set S of sorts,
- an $S^* \times S$-indexed set $(TF_{w,s})_{w,s \in S^* \times S}$ of total operation symbols,

CASL many-sorted signatures

- a set S of sorts,
- an $S^* \times S$-indexed set $(TF_{w,s})_{w,s \in S^* \times S}$ of total operation symbols,
- an $S^* \times S$-indexed set $(PF_{w,s})_{w,s \in S^* \times S}$ of partial operation symbols, such that $TF_{w,s} \cap PF_{w,s} = \emptyset$,

CASL many-sorted signatures

- a set S of sorts,
- an $S^* \times S$-indexed set $(TF_{w,s})_{w,s \in S^* \times S}$ of total operation symbols,
- an $S^* \times S$-indexed set $(PF_{w,s})_{w,s \in S^* \times S}$ of partial operation symbols, such that $TF_{w,s} \cap PF_{w,s} = \emptyset$,
- an S^*-indexed set $(P_w)_{w \in S^*}$ of predicate symbols

Signature morphisms map these components in a compatible way

Example signatures

- $\Sigma^{Nat} = (\{Nat\}, \{0 : Nat, succ\colon Nat \longrightarrow Nat\},$
  $\{pre\colon Nat \longrightarrow ?Nat\}, \emptyset)$

- $(\{Elem\}, \emptyset, \emptyset, \{\_\_ < \_\_ : Elem * Elem\})$

- $(\{Elem, List\},$
  $\{Nil : Elem, Cons\colon Elem * List \longrightarrow List\}, \emptyset, \emptyset)$

CASL many-sorted models

For a many-sorted signature $\Sigma = (S, TF, PF, P)$ a many-sorted model $M \in \mathbf{Mod}(\Sigma)$ consists of

- a non-empty carrier set s^M for each sort $s \in S$ (let w^M denote the Cartesian product $s_1^M \times \cdots \times s_n^M$ when $w = s_1 \ldots s_n$),

CASL many-sorted models

For a many-sorted signature $\Sigma = (S, \mathit{TF}, \mathit{PF}, P)$ a many-sorted model $M \in \mathbf{Mod}(\Sigma)$ consists of

- a non-empty carrier set s^M for each sort $s \in S$ (let w^M denote the Cartesian product $s_1^M \times \cdots \times s_n^M$ when $w = s_1 \ldots s_n$),
- a partial function f^M from w^M to s^M for each function symbol $f \in \mathit{TF}_{w,s}$ or $f \in \mathit{PF}_{w,s}$, the function being required to be total in the former case,

CASL many-sorted models

For a many-sorted signature $\Sigma = (S, TF, PF, P)$ a many-sorted model $M \in \mathbf{Mod}(\Sigma)$ consists of

- a non-empty carrier set s^M for each sort $s \in S$ (let w^M denote the Cartesian product $s_1^M \times \cdots \times s_n^M$ when $w = s_1 \ldots s_n$),
- a partial function f^M from w^M to s^M for each function symbol $f \in TF_{w,s}$ or $f \in PF_{w,s}$, the function being required to be total in the former case,
- a predicate $p^M \subseteq w^M$ for each predicate symbol $p \in P_w$.

Example Σ^{Nat}-models

- $Nat^M = I\!N$, 0^M=0, $suc^M(x) = x + 1$,
  $pre^M(x) = \begin{cases} x - 1, x > 0 \\ \text{undefined, otherwise} \end{cases}$

Example Σ^{Nat}-models

- $Nat^M = \mathbb{N}$, 0^M=0, $suc^M(x) = x + 1$,
  $$pre^M(x) = \begin{cases} x-1, x>0 \\ \text{undefined, otherwise} \end{cases}$$

- $Nat^N = \mathbb{N} \cup \{\infty\}$, 0^N=0,
  $$suc^N(x) = \begin{cases} \infty, & \text{if } x = \infty \\ x+1, & \text{otherwise} \end{cases},$$
  $$pre^N(x) = \begin{cases} x-1, & \text{if } 0 < x \neq \infty \\ \text{undefined}, & \text{otherwise} \end{cases}$$

Example Σ^{Nat}-models

- $Nat^M = \mathbb{N}$, 0^M=0, $suc^M(x) = x + 1$,
  $pre^M(x) = \begin{cases} x - 1, x > 0 \\ \text{undefined, otherwise} \end{cases}$

- $Nat^N = \mathbb{N} \cup \{\infty\}$, 0^N=0,
  $suc^N(x) = \begin{cases} \infty, & \text{if } x = \infty \\ x + 1, & \text{otherwise} \end{cases}$,
  $pre^N(x) = \begin{cases} x - 1, & \text{if } 0 < x \neq \infty \\ \text{undefined}, & \text{otherwise} \end{cases}$

- $Nat^T = \{*\}$, $0^T = *$, $suc^T(*) = *$, $pre^T(*) = *$

Example Σ^{Nat}-models

- $Nat^M = \mathbb{N}$, 0^M=0, $suc^M(x) = x + 1$,
  $$pre^M(x) = \begin{cases} x - 1, x > 0 \\ \text{undefined, otherwise} \end{cases}$$

- $Nat^N = \mathbb{N} \cup \{\infty\}$, 0^N=0,
  $$suc^N(x) = \begin{cases} \infty, & \text{if } x = \infty \\ x + 1, & \text{otherwise} \end{cases},$$
  $$pre^N(x) = \begin{cases} x - 1, & \text{if } 0 < x \neq \infty \\ \text{undefined}, & \text{otherwise} \end{cases}$$

- $Nat^T = \{*\}$, $0^T = *$, $suc^T(*) = *$, $pre^T(*) = *$

- $Nat^K = \mathbb{N}$, $0^N = K$, $suc^K(x) = x$,
  $$pre^K(x) = \begin{cases} y, & \text{if TM } x \text{ outputs } y \text{ on input } x \\ \text{undefined}, & \text{otherwise} \end{cases}$$

- $Nat^K = \mathbb{N}$, $0^N = K$, $suc^K(x) = x$,
  $pre^K(x) = \begin{cases} y, & \text{if TM } x \text{ outputs } y \text{ on input } x \\ \text{undefined}, & \text{otherwise} \end{cases}$

- $Nat^F = \mathbb{N} \to \mathbb{N}$, $0^F(x) = 0$, $suc^F(f)(x) = f(x) + 1$,
  $pre^F(f)$ undefined for each f

CASL many-sorted terms

Given a signature Σ and a variable system $(X_s)_{s \in S}$, the set of terms is defined inductively as follows:

- variables $x \in X_s$ are terms of sort s

CASL many-sorted terms

Given a signature Σ and a variable system $(X_s)_{s \in S}$, the set of terms is defined inductively as follows:

- variables $x \in X_s$ are terms of sort s
- applications $f_{w,s}(t_1, \ldots, t_n)$ is a term of sort s, if $f \in TF_{w,s} \cup PF_{w,s}$ and t_i is a term of sort s_i, $w = s_1 \ldots s_n$.

Semantics of terms

Given a Σ-model and a variable valuation $\nu\colon X \longrightarrow M$, the semantics $\nu^{\#}$ of terms is defined as follows:

- variables $\nu^{\#}(x) = \nu(x)$

Semantics of terms

Given a Σ-model and a variable valuation $\nu\colon X \longrightarrow M$, the semantics $\nu^{\#}$ of terms is defined as follows:

- variables $\nu^{\#}(x) = \nu(x)$
- applications $\nu^{\#}(f_{w,s}(t_1, \ldots, t_n)) = f_{w,s}^{M}(\nu^{\#}(t_1), \ldots, \nu^{\#}(t_n))$ if all components are defined (undefined otherwise)

CASL formulae

The set of (Σ, X)-formulae is defined inductively as follows:

- strong equations $t_1 = t_2$

CASL formulae

The set of (Σ, X)-formulae is defined inductively as follows:

- strong equations $t_1 = t_2$
- existential equations $t_1 \stackrel{e}{=} t_2$

CASL formulae

The set of (Σ, X)-formulae is defined inductively as follows:

- strong equations $t_1 = t_2$
- existential equations $t_1 \stackrel{e}{=} t_2$
- predications $p_w(t_1, \ldots, t_n)$

CASL formulae

The set of (Σ, X)-formulae is defined inductively as follows:

- strong equations $t_1 = t_2$
- existential equations $t_1 \stackrel{e}{=} t_2$
- predications $p_w(t_1, \ldots, t_n)$
- definedness assertions $def(t)$

CASL formulae

The set of (Σ, X)-formulae is defined inductively as follows:

- strong equations $t_1 = t_2$
- existential equations $t_1 \stackrel{e}{=} t_2$
- predications $p_w(t_1, \ldots, t_n)$
- definedness assertions $def(t)$
- conjunctions, disjunctions, implications, equivalences of formulae

CASL formulae

The set of (Σ, X)-formulae is defined inductively as follows:

- strong equations $t_1 = t_2$
- existential equations $t_1 \stackrel{e}{=} t_2$
- predications $p_w(t_1, \ldots, t_n)$
- definedness assertions $def(t)$
- conjunctions, disjunctions, implications, equivalences of formulae
- universal, existential, unique-existential quantifications

Satisfaction of atomic formulae

A formula φ is satisfied in a model M w.r.t. a valuation $\nu\colon X \longrightarrow M$ (short notation: $M, \nu \models \varphi$), if

- $M, \nu \models t_1 = t_2$ if $\nu^{\#}(t_1) = \nu^{\#}(t_2)$ or both sides are undefined,

Satisfaction of atomic formulae

A formula φ is satisfied in a model M w.r.t. a valuation $\nu: X \longrightarrow M$ (short notation: $M, \nu \models \varphi$), if

- $M, \nu \models t_1 = t_2$ if $\nu^{\#}(t_1) = \nu^{\#}(t_2)$ or both sides are undefined,
- $M, \nu \models t_1 \stackrel{e}{=} t_2$ if $\nu^{\#}(t_1) = \nu^{\#}(t_2)$ and both sides defined,

Satisfaction of atomic formulae

A formula φ is satisfied in a model M w.r.t. a valuation $\nu\colon X \longrightarrow M$ (short notation: $M, \nu \models \varphi$), if

- $M, \nu \models t_1 = t_2$ if $\nu^{\#}(t_1) = \nu^{\#}(t_2)$ or both sides are undefined,
- $M, \nu \models t_1 \stackrel{e}{=} t_2$ if $\nu^{\#}(t_1) = \nu^{\#}(t_2)$ and both sides defined,
- $M, \nu \models p_w(t_1, \dots, t_n)$
  if $(\nu^{\#}(t_1), \dots, \nu^{\#}(t_n))$ is defined and $\in p_w^M$,

Satisfaction of atomic formulae

A formula φ is satisfied in a model M w.r.t. a valuation $\nu: X \longrightarrow M$ (short notation: $M, \nu \models \varphi$), if

- $M, \nu \models t_1 = t_2$ if $\nu^{\#}(t_1) = \nu^{\#}(t_2)$ or both sides are undefined,
- $M, \nu \models t_1 \stackrel{e}{=} t_2$ if $\nu^{\#}(t_1) = \nu^{\#}(t_2)$ and both sides defined,
- $M, \nu \models p_w(t_1, \ldots, t_n)$
  if $(\nu^{\#}(t_1), \ldots, \nu^{\#}(t_n))$ is defined and $\in p_w^M$,
- $M, \nu \models def(t)$ if $\nu^{\#}(t)$ is defined

Satisfaction of compound formulae

A standard in first-order logic, i.e.

- a conjuction is satisfied iff all the conjuncts are satisfied

Satisfaction of compound formulae

A standard in first-order logic, i.e.

- a conjuction is satisfied iff all the conjuncts are satisfied
- similar for disjunction etc.

Satisfaction of compound formulae

A standard in first-order logic, i.e.

- a conjuction is satisfied iff all the conjuncts are satisfied
- similar for disjunction etc.
- a universal (existential) quantification is satisfied when all (some) of the changes of the valuation for the quantified variable lead to satisfcation in the model: $M, \nu \models \forall x : s \,.\, \phi$ iff $M, \xi \models \phi$ for all valuation ξ that differ from ν only on $x : s$

Satisfaction of closed formulae

A closed formula (sentences) is satisfied in a model iff it is satisfied w.r.t. the empty valuation:

$$M \models \varphi \text{ iff } M, \emptyset \models \varphi$$

Sort generation constraints

A Σ-sort-generation constraint (S', F') consists of

Sort generation constraints

A Σ-sort-generation constraint (S', F') consists of

- a set of sorts $S' \subseteq S$

Sort generation constraints

A Σ-sort-generation constraint (S', F') consists of

- a set of sorts $S' \subseteq S$
- a set of (qualified) operation symbols $F' \subseteq TF \cup PF$

Sort generation constraints

A Σ-sort-generation constraint (S', F') consists of

- a set of sorts $S' \subseteq S$
- a set of (qualified) operation symbols $F' \subseteq TF \cup PF$

$M \models (S', F')$ iff the carriers of sorts in S' are generated by terms in F' (with variables of sorts outside S')

Sort generation constraints

A Σ-sort-generation constraint (S', F') consists of

- a set of sorts $S' \subseteq S$
- a set of (qualified) operation symbols $F' \subseteq TF \cup PF$

$M \models (S', F')$ iff the carriers of sorts in S' are generated by terms in F' (with variables of sorts outside S')
i.e. for each $s \in S'$, $a \in s^M$, there is some term t (with variables of sorts outside S') and some valuation ν with $\nu^{\#}(t) = a$.

Example Σ^{Nat}-models

- $Nat^M = \mathbb{N}$, 0^M=0, $suc^M(x) = x + 1$,
  $$pre^M(x) = \begin{cases} x - 1, x > 0 \\ \text{undefined, otherwise} \end{cases}$$

- $Nat^N = \mathbb{N} \cup \{\infty\}$, 0^N=0,
  $$suc^N(x) = \begin{cases} \infty, & \text{if } x = \infty \\ x + 1, & \text{otherwise} \end{cases},$$
  $$pre^N(x) = \begin{cases} x - 1, & \text{if } 0 < x \neq \infty \\ \text{undefined}, & \text{otherwise} \end{cases}$$

- $Nat^T = \{*\}$, $0^T = *$, $suc^T(*) = *$, $pre^T(*) = *$

- $Nat^K = \mathbb{N}$, $0^N = K$, $suc^K(x) = x$,
  $pre^K(x) = \begin{cases} y, & \text{if TM } x \text{ outputs } y \text{ on input } x \\ \text{undefined}, & \text{otherwise} \end{cases}$

- $Nat^F = \mathbb{N} \to \mathbb{N}$, $0^F(x) = 0$, $suc^F(f)(x) = f(x) + 1$,
  $pre^F(f)$ undefined for each f

Semantics of CASL Structured Specifications

Institutions

- Basic idea: abstract away from the details of signature, model, sentence, satisfaction.
- The semantics of CASL structured specifications is defined for an arbitrary institution.
- first-order, higher-order, polymorphic, modal, temporal, process, behavioural, ASM- und Z-like and object-oriented logics have been shown to be institutions.
- Hence, you may replace the CASL institution with your favourite institution.

The CASL institution revisited

Given a signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, $\Sigma = (S, TF, PF, P)$ and a Σ'-model M', the reduct $M'|_\sigma$ is defined as follows

- $s^M := \sigma(s)^{M'}$ for $s \in S$,
- $f_{w,s}^M := \sigma(f_{w,s})^{M'}$ for $f \in TF_{w,s} \cup PF_{w,s}$,
- $p_w^M := \sigma(p_w)^{M'}$ for $p \in P_w$.

A Σ-formula φ is translated along σ by just replacing the symbols in φ according to σ.

The Satisfaction Condition

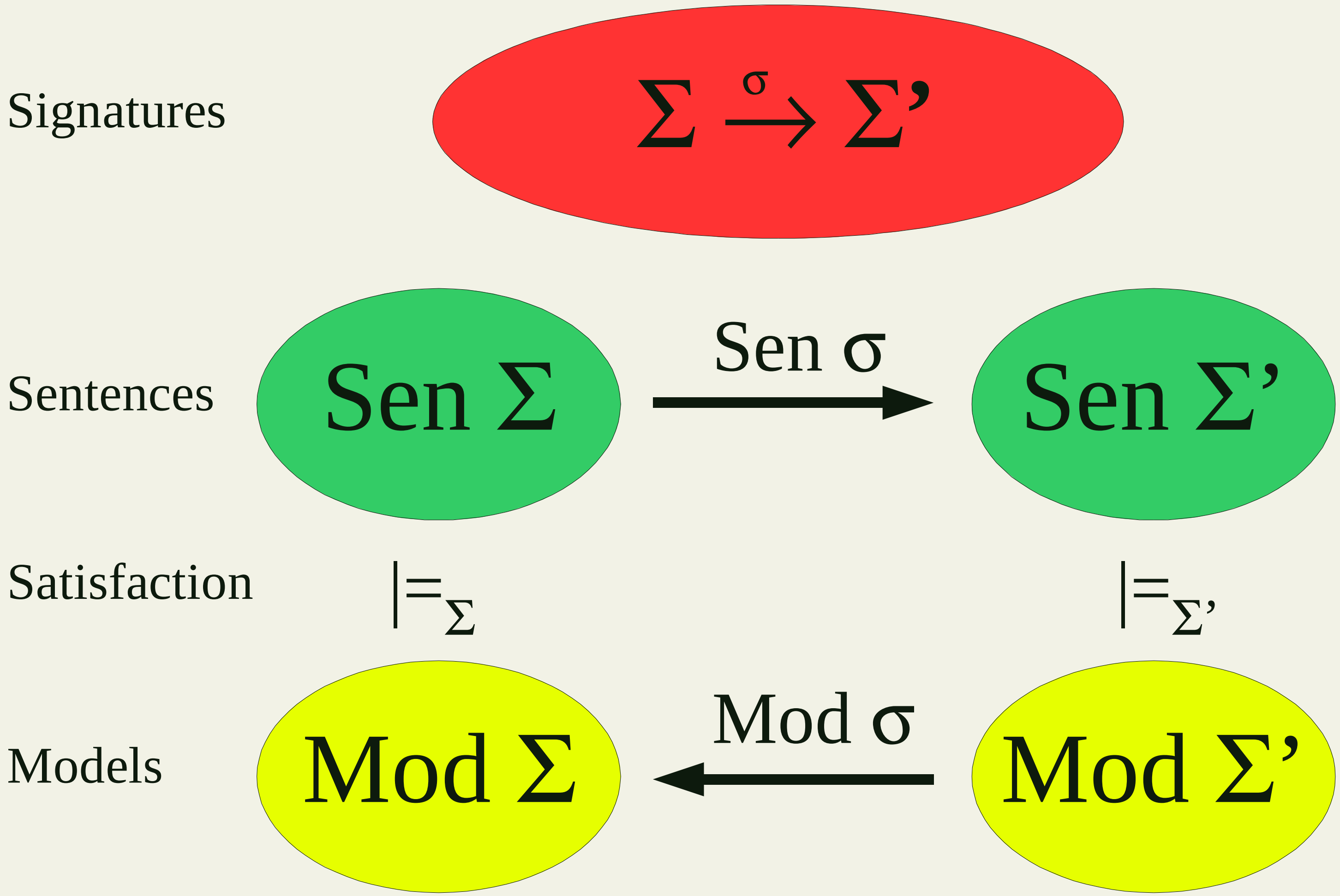
Theorem

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

That is:

Truth is invariant under change of notation and enlargement of context.

Institutions



Institutions, formally

- category $\mathbf{Sign}$ of signatures,
- a sentence functor $\mathbf{Sen}\colon \mathbf{Sign} \longrightarrow \mathbf{Set}$,
- a model functor $\mathbf{Mod}\colon \mathbf{Sign}^{op} \longrightarrow \mathcal{CAT}$,
- a satisfaction relation $\models_\Sigma \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$,

such that the following satisfaction condition holds:

$$M' \models_{\Sigma'} \mathbf{Sen}(\sigma)(\varphi) \Leftrightarrow \mathbf{Mod}(\sigma)(M)' \models_\Sigma \varphi$$

or shortly

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

Benefits of institutions

- Institution independent semantics (and proof system) of structured specifications, architectural specifications, refinement, behavioural abstraction etc.
- ASMs over arbitrary institutions (Zucca 1999, TCS 216)
- Borrowing of parts of a logic from other logics
- Combination of logics
- Heterogeneous specification and tools
- Abstract model theory with deep results (Diaconescu)

Semantics of basic specifications

$$\frac{\Sigma \vdash \texttt{BASIC-SPEC} \rhd (\Sigma', \Psi)}{\Sigma \vdash \texttt{BASIC-SPEC}\ qua\ \texttt{SPEC} \rhd \Sigma'}$$

$$\frac{\Sigma \vdash \texttt{BASIC-SPEC} \rhd (\Sigma', \Psi) \qquad \mathcal{M}' = \{M \in \mathbf{Mod}(\Sigma') \mid M|_{\Sigma} \in \mathcal{M}, M \models \Psi\}}{\Sigma, \mathcal{M} \vdash \texttt{BASIC-SPEC}\ qua\ \texttt{SPEC} \Rightarrow \Sigma', \mathcal{M}'}$$

Semantics of translations

$$\frac{\Sigma \vdash \texttt{SPEC} \rhd \Sigma'}{\Sigma \vdash \texttt{SPEC}\ \textbf{with}\ \sigma\colon \Sigma' \longrightarrow \Sigma'' \rhd \Sigma''}$$

$$\frac{\begin{array}{c}\Sigma, \mathcal{M} \vdash \texttt{SPEC} \Rightarrow \Sigma', \mathcal{M}' \\ \mathcal{M}'' = \{M \in \mathbf{Mod}(\Sigma'') \mid M|_\sigma \in \mathcal{M}'\}\end{array}}{\Sigma, \mathcal{M} \vdash \texttt{SPEC}\ \textbf{with}\ \sigma\colon \Sigma' \longrightarrow \Sigma'' \Rightarrow \Sigma'', \mathcal{M}''}$$

Semantics of reductions

$$\frac{\Sigma \vdash \texttt{SPEC} \rhd \Sigma'}{\Sigma \vdash \texttt{SPEC}\ \textbf{hide}\ \sigma\colon \Sigma'' \longrightarrow \Sigma' \rhd \Sigma''}$$

$$\frac{\begin{array}{c}\Sigma, \mathcal{M} \vdash \texttt{SPEC} \Rightarrow \Sigma', \mathcal{M}' \\ \mathcal{M}'' = \{M|_\sigma \mid M \in \mathcal{M}'\}\end{array}}{\Sigma, \mathcal{M} \vdash \texttt{SPEC}\ \textbf{hide}\ \sigma\colon \Sigma'' \longrightarrow \Sigma' \Rightarrow \Sigma'', \mathcal{M}''}$$

Semantics of extensions

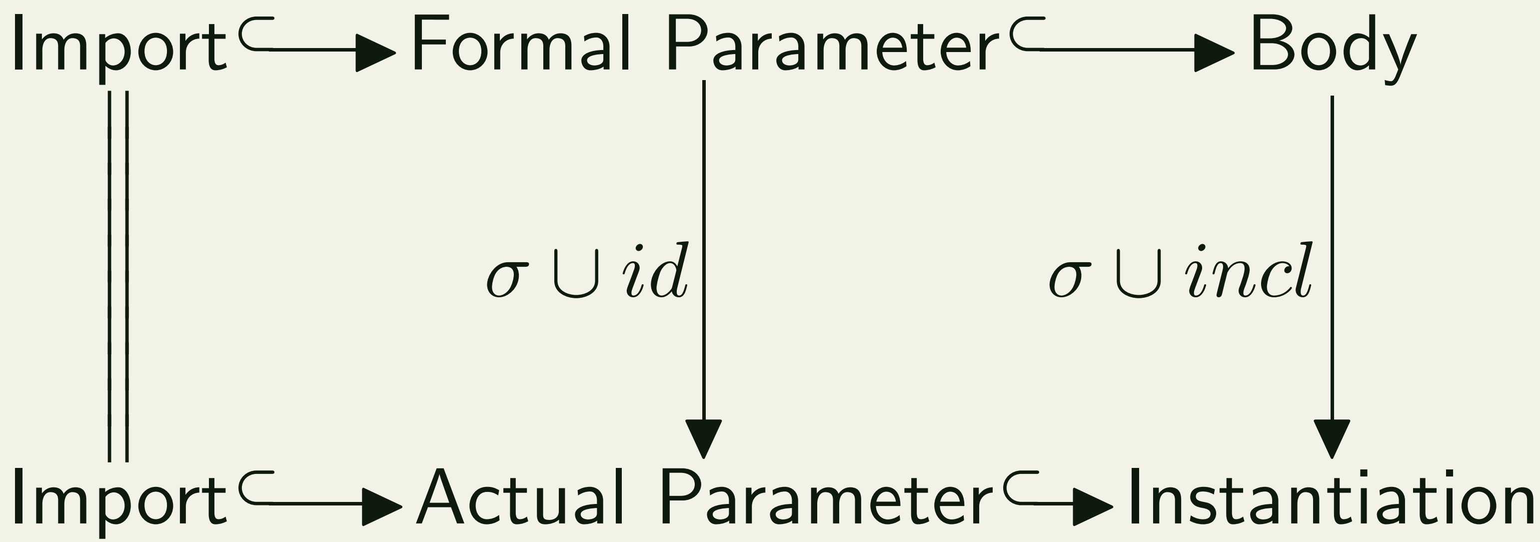
$$\frac{\Sigma \vdash \mathrm{SPEC}_1 \rhd \Sigma' \qquad \Sigma' \vdash \mathrm{SPEC}_2 \rhd \Sigma''}{\Sigma \vdash \mathrm{SPEC}_1 \textbf{ then } \mathrm{SPEC}_2 \rhd \Sigma''}$$

$$\frac{\Sigma, \mathcal{M} \vdash \mathrm{SPEC}_1 \Rightarrow \Sigma', \mathcal{M}' \qquad \Sigma', \mathcal{M}' \vdash \mathrm{SPEC}_2 \Rightarrow \Sigma'', \mathcal{M}''}{\Sigma, \mathcal{M} \vdash \mathrm{SPEC}_1 \textbf{ then } \mathrm{SPEC}_2 \Rightarrow \Sigma'', \mathcal{M}''}$$

Semantics of views

$$\frac{\begin{array}{c}\emptyset, \mathcal{M}_\perp \vdash \mathtt{SPEC}_1 \Rightarrow \Sigma_1, \mathcal{M}_1 \\ \emptyset, \mathcal{M}_\perp \vdash \mathtt{SPEC}_2 \Rightarrow \Sigma_2, \mathcal{M}_2 \\ \text{for each } M \in \mathcal{M}_2,\ M|_\sigma \in \mathcal{M}_1\end{array}}{\vdash \textbf{view } \mathtt{SPEC}_1 \textbf{ to } \mathtt{SPEC}_2 = \sigma \Rightarrow \sigma, \mathcal{M}_1, \mathcal{M}_2}$$

Semantics of parameterization (simplified)



The right square is required to be a pushout, that is, all symbols shared between the body and the actual parameter must occur also in the formal parameter.

Models: those models of the instantiation whose reducts are models of the body and of the actual parameter.

Development graphs $\mathcal{S} = \langle \mathcal{N}, \mathcal{L} \rangle$

Nodes in $\mathcal{N}$: (Σ^N, Γ^N) with

- Σ^N signature,
- $\Gamma^N \subseteq \mathbf{Sen}(\Sigma^N)$ set of local axioms.

Links in $\mathcal{L}$:

- global $M \xrightarrow{\sigma} N$, where $\sigma : \Sigma^M \rightarrow \Sigma^N$,
- local $M \overset{\sigma}{\cdots\cdots\blacktriangleright} N$ where $\sigma : \Sigma^M \rightarrow \Sigma^N$, or
- hiding $M \xrightarrow[h]{\sigma} N$ where $\sigma : \Sigma^N \rightarrow \Sigma^M$ going against the direction of the link.

Semantics of development graphs

$\mathbf{Mod}_S(N)$ consists of those Σ^N-models n for which

1. n satisfies the local axioms Γ^N,
2. for each $K \xrightarrow{\sigma} N \in \mathcal{S}$, $n|_\sigma$ is a K-model,
3. for each $K \overset{\sigma}{\dashrightarrow} N \in \mathcal{S}$,
   $n|_\sigma$ satisfies the local axioms Γ^K,
4. for each $K \xrightarrow[h]{\sigma} N \in \mathcal{S}$,
   n has a σ-expansion k (i.e. $k|_\sigma = n$) that is a K-model.

Theorem links

Theorem links come in two versions:

- global theorem links $M \xrightarrow{\sigma} N$, where $\sigma\colon \Sigma^M \longrightarrow \Sigma^N$,
  - $\mathcal{S} \models M \xrightarrow{\sigma} N$ iff for all $n \in \mathbf{Mod}_{\mathcal{S}}(N)$, $n|_\sigma \in \mathbf{Mod}_{\mathcal{S}}(M)$.
- local theorem links $M \overset{\sigma}{\dashrightarrow} N$, where $\sigma\colon \Sigma^M \longrightarrow \Sigma^N$,
  - $\mathcal{S} \models M \overset{\sigma}{\dashrightarrow} N$ iff for all $n \in \mathbf{Mod}_{\mathcal{S}}(N)$, $n|_\sigma \models \Gamma^M$.
- the calculus reduces these to local proof obligations.