Korrektheit

Definition 7.2 *Eine Spezifikation* $spec = (sig, E)$ *ist sig-korrekt bzgl. einer sig-Algebra* $\mathfrak{A}$ *gdw* $T_{spec} \cong \mathfrak{A}$ *(d. h. der eindeutige Hom. ist Bijektion).*

Beispiel 7.3 *Anwendung:*
INT korrekt für $\mathbb{Z}$, *BOOL korrekt für* $\mathbb{B}$

Beachte: Begriff ist hier auf initiale Semantik beschränkt!

Einschränkungen/Vergißbilder

Definition 7.3 *Einschränkungen/Vergißbilder*

a) $sig = (S, F, \tau)$, $sig' = (S', F', \tau')$ *Signaturen mit* $sig \subseteq sig'$, *d. h.* $(S \subseteq S', F \subseteq F', \tau \subseteq \tau')$.

Für jede sig'*-Algebra* $\mathfrak{A}$ *sei sig-Anteil* $\mathfrak{A}|_{sig}$ *von* $\mathfrak{A}$ *die sig-Algebra mit*

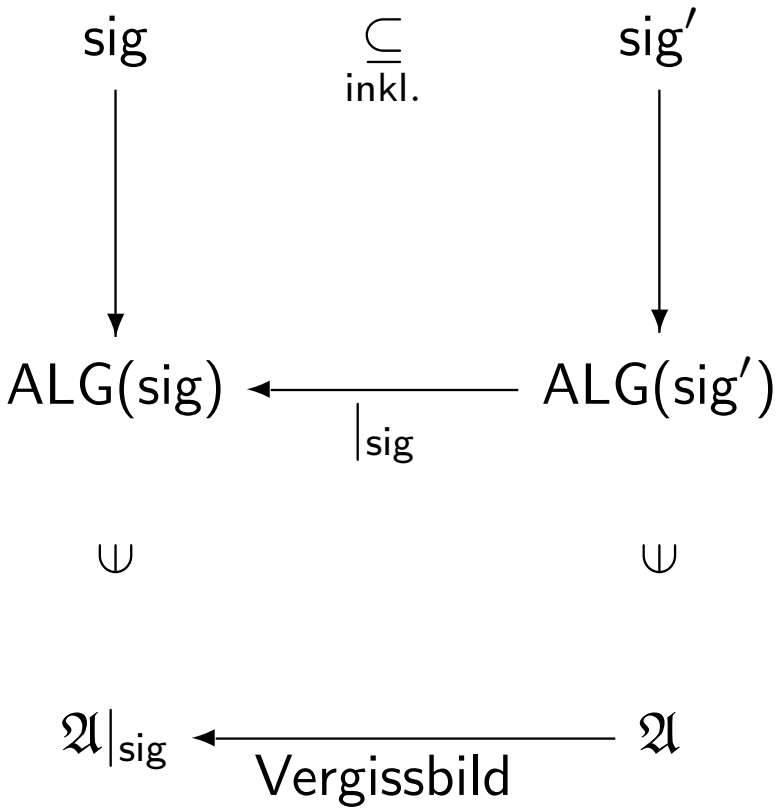
i) $(\mathfrak{A}|_{sig})_s = A_s$ *für* $s \in S$
ii) $f_{\mathfrak{A}|_{sig}} = f_{\mathfrak{A}}$ *für* $f \in F$

Beachte: $\mathfrak{A}|_{\text{sig}}$ ist sig - Algebra.

$\mathfrak{A}|_{\text{sig}}$ heißt auch Vergißbild von $\mathfrak{A}$ (bzgl. sig).

Einschränkungen/Vergißbilder

$\mathfrak{A}|_{\text{sig}}$ Vergißbild von $\mathfrak{A}$ (bzgl. sig). Vergißbild induziert somit eine Abbildung zwischen Algebrenklassen wie folgt:



Einschränkungen/Vergißbilder

b) Eine Spezifikation $\text{spec} = (\text{sig}', E)$ mit $\text{sig} \subseteq \text{sig}'$ ist korrekt bzgl. sig-Algebra $\mathfrak{A}$ gdw

$$(T_{\text{spec}})|_{\text{sig}} \cong \mathfrak{A}$$

(Der Hom. von $(T_{\text{spec}})|_{\text{sig}}$ nach $\mathfrak{A}$ ist Bijektion).

c) Eine Spezifikation $\text{spec}' = (\text{sig}', E')$ implementiert eine Spezifikation $\text{spec} = (\text{sig}, E)$ gdw

$$\text{sig} \subseteq \text{sig}' \text{ und } (T_{\text{spec}'})|_{\text{sig}} \cong T_{\text{spec}}$$

Beachte:

- Konsistenz-Begriff für $=$-Spezifikation nicht notwendig. (Modelle existieren immer!).
- Allgemeiner Implementierungsbegriff ($Cl(\text{spec}) \subseteq Cl(\text{spec}')$) reduziert sich immer auf $=$.
  "vollständige„ Theorien.

Probleme

Nachweis $s = t \in Th(E)$ oder $\in ITH(E)$.

Für $Th(E)$ finde $=_E$ äquivalentes konvergentes Termersetzungssystem (Siehe Gruppenbeispiel).

Für $ITH(E)$ Induktionsmethoden:

s, t induzieren Funktionen auf T_{spec}. Sind $x_1, \ldots, x_n$ die Variablen in s und t, Sorten $s_1, \ldots, s_n$.
$s : (T_{\text{spec}})_{s_1} \times \cdots \times (T_{\text{spec}})_{s_n} \to (T_{\text{spec}})_s$
$s = t \in ITh(E)$ gdw s und t induzieren gleiche Funktionen $\rightsquigarrow$ Beweise durch Induktion über Aufbau der Terme.
NAT $0, \text{suc}, +$ $x + y = y + x \in ITH$
$0 + x = x$

Probleme

- $0 + 0 = 0 \qquad \text{Ann} : 0 + a = a$
  $0 + Sa =_E S(0 + a) =_I S(a)$
- $x + 0 = 0 + x \qquad \text{Ann} : x + a = a + x$
  $x + Sa =_E S(x + a) =_I S(a + x) =_E a + Sx \stackrel{?}{=} Sa + x$
- $x + Sy = Sx + y$
  $x + So =_E S(x + o) =_E Sx =_E Sx + o$
  $x + SSa =_E S(x + Sa) =_I S(Sx + a) = Sx + Sa$

$\text{spec}(\text{sig}, E)$	$P_{\text{spec}}(\text{sig}, E, Prop)$
Gleichungen reichen	Eigenschaften die gelten sollen!
oft nicht aus	Verifikationsaufgaben

Strukturierungsmechanismen

Horizontal:	- Zerlegung, - Kombination,
	- Erweiterung, - Instantiierung
Vertikal:	- Realisierung, - Information hiding,
	- Vertikale Komposition

Hier:
Kombination, Anreicherung, Erweiterung,
Modularisierung, Parametrisierung
$\rightsquigarrow$ **Wiederverwendbarkeit.**

Strukturierungsmechanismen

BIN-TREE

1)
spec	NAT	
sorts	nat	
ops	$0 :\longrightarrow \text{nat}$	
	$\text{suc} : \text{nat} \longrightarrow \text{nat}$	

2)
spec	NAT1
use	NAT
ops	$\text{max} : \text{nat}, \text{nat} \longrightarrow \text{nat}$
eqns	$\text{max}(0, n) = n$
	$\text{max}(n, 0) = n$
	$\text{max}(s(m), s(n)) =$
	$s(\text{max}(m, n))$

Strukturierungsmechanismen

BIN-TREE (Forts.)

3) spec BINTREE1
sorts bintree
ops leaf :⟶ bintree

left, right : bintree
⟶ bintree
both : bintree, bintree
⟶ bintree

4) spec BINTREE2
use NAT1, BINTREE1
ops height : bintree ⟶ nat
eqns ⋮

Kombination

Definition 7.4 *Sei* $spec_1 = (sig_1, E_1)$, *mit* $sig_1 = (S_1, F_1, \tau_1)$ *Signatur und* $sig_2 = [S_2, F_2, \tau_2]$ *Tripel,* E_2 *Gleichungsmenge.*

$\mathsf{comb} = spec_1 + (sig_2, E_2)$ *heißt Kombination*
gdw
$spec = ((S_1 \cup S_2), (F_1 \cup F_2), (\tau_1 \cup \tau_2)), E_1 \cup E_2)$ *eine Spezifikation ist.*

Insbesondere ist $((S_1 \cup S_2), (F_1 \cup F_2), (\tau_1 \cup \tau_2))$ *eine Signatur und* E_2 *enthält „syntaktisch korrekte“ Gleichungen.*

Die Semantik von comb: $T_{\mathsf{comb}} := T_{spec}$

Die Semantik von comb

$T_{\text{comb}} := T_{\text{spec}}$

Typische Fälle:

$S_2 = \varnothing$, F_2 neue Funktionssymbole mit Stelligkeiten τ_2 (alte).

S_2 neue Sorten, F_2 neue Funktionssymbole.
τ_2 Stelligkeiten neue + alte.

E_2 nur „neue“ Gleichungen.

Notationen: use, include (protected)

Beispiel

Beispiel 7.4 *a) Schrittweiser Entwurf der ganzen Zahlen*

		Semantik
spec	INT1	
sorts	int	$T_{\text{INT1}} \cong (\mathbb{N}, 0, \text{suc}_{\mathbb{N}})$
ops	$0 :\longrightarrow \text{int}$	
	$\text{suc} : \text{int} \longrightarrow \text{int}$	
	$\cap$	$\cap$
spec	INT2	
use	INT1	$T_{\text{INT2}} \cong (\mathbb{Z}, 0, \text{suc}_{\mathbb{Z}}, \text{pred}_{\mathbb{Z}})$
ops	$\text{pred} : \text{int} \longrightarrow \text{int}$	
eqns	$\text{pred}(\text{suc}(x)) = x$	
	$\text{suc}(\text{pred}(x)) = x$	

Beispiel (Forts.)

Frage: Stimmt der INT1-Anteil von T_{INT2} mit T_{INT1} überein??
Implementiert INT2 INT1?

$$(T_{\mathsf{INT2}})|_{\mathsf{INT1}} \cong T_{\mathsf{INT1}}$$

$$(\mathbb{Z}, 0, \mathsf{suc}_{\mathbb{Z}}, \mathsf{pred}_{\mathbb{Z}})|_{\mathsf{INT1}}$$
$$\shortparallel$$
$$(\mathbb{Z}, 0, \mathsf{suc}_{\mathbb{Z}}) \qquad \not\cong (\mathbb{N}, 0, \mathsf{suc}_{\mathbb{N}})$$

Vorsicht:
Nicht immer erfasst man das richtige! Hier wurden neue Daten von der Sorte int

Beispiel (Forts.)

b) spec NAT2
use NAT
eqns $\mathsf{suc}(\mathsf{suc}(x)) = x$

$$(T_{\mathsf{NAT2}})|_{\mathsf{NAT}} = (\mathbb{N} \bmod 2)|_{\mathsf{NAT}} = \mathbb{N} \bmod 2 \supsetneq \mathbb{N} = T_{\mathsf{NAT}}$$

Problem: Hinzufügen neuer bzw. Identifizieren alter Elemente.

Probleme mit der Kombination

Sei

$$\text{comb} = \text{spec}_1 + (\text{sig}, E)$$

$$\left.\begin{array}{l} (T_{\text{comb}})|_{\text{spec}_1} \text{ ist spec}_1 \text{ Algebra} \\ T_{\text{spec}_1} \text{ ist initiale spec}_1 \text{ Algebra} \end{array}\right\} \rightsquigarrow$$

$\exists!$ Homomorphismus $h : T_{\text{spec}_1} \rightarrow (T_{\text{comb}})|_{\text{spec}_1}$

Eigenschaften von
h: nicht injektiv / nicht surjektiv / bijektiv.

z. B. $(T_{\text{BINTREE2}})|_{\text{NAT}} \cong T_{\text{NAT}}$.

Erweiterung und Anreicherung

Definition 7.5 *a) Eine Kombination* $\mathrm{comb} = spec_1 + (sig, E)$ *ist eine Erweiterung gdw*

$$(T_{\mathrm{comb}})|_{spec_1} \cong T_{spec_1}$$

b) Eine Erweiterung heißt Anreicherung, wenn sig keine neuen Sorten enthält, d. h. $sig = [\varnothing, F_2, \tau_2]$

- *hinreichende Bedingungen?*

Parametrisierung

Definition 7.6 *Eine parametrisierte Spezifikation Parameter=(Formal, Body) besteht aus zwei Spezifikationen: Formal und Body mit Formal* $\subseteq$ *Body.*

D. h. Formal=(sig_F, E_F)*, Body=*(sig_B, E_B)*, wobei*
$sig_F \subseteq sig_B \qquad E_F \subseteq E_B$.

Notation: Body[Formal]

Syntaktisch: Body = Formal $+(sig', E')$ *Kombination*

Beispiel

Beispiel 7.5

spec ELEM
sorts elem
ops next : elem $\longrightarrow$ elem

$(T_{spec})_{\text{elem}} = \varnothing$

spec STRING[ELEM]
use ELEM
sorts string
ops empty :$\longrightarrow$ string
unit : elem $\longrightarrow$ string
concat : string, string $\longrightarrow$ string
ladd : elem, string $\longrightarrow$ string
radd : string, elem $\longrightarrow$ string

$(T_{\text{spec}})_{\text{string}} = \{[\text{empty}]\}$

Beispiel (Forts.)

$$
\begin{aligned}
\text{eqns}\quad & \text{concat}(s, \text{empty}) = s \\
& \text{concat}(\text{empty}, s) = s \\
& \text{concat}(\text{concat}(s_1, s_2), s_3) = \text{concat}(s_1, \text{concat}(s_2, s_3)) \\
& \text{ladd}(e, s) = \text{concat}(\text{unit}(e), s) \\
& \text{radd}(s, e) = \text{concat}(s, \text{unit}(e))
\end{aligned}
$$

Parameterübergabe: ELEM $\rightarrow$ NAT

$$\text{STRING[ELEM]} \rightarrow \text{STRING[NAT]}$$

Zuordnung: formale Parameter $\rightarrow$ aktuelle Parameter

$$
\begin{aligned}
S_F &\rightarrow S_A \\
Op &\rightarrow Op_A
\end{aligned}
$$

Semantik?

Signaturmorphismen - Parameterübergabe

Definition 7.7 *a) Seien* $sig_i = (S_i, F_i, \tau_i)$ $i = 1, 2$ *Signaturen. Ein Paar* $\sigma = (g, h)$ *mit* $g : S_1 \to S_2, h : F_1 \to F_2$ *von Funktionen ist Signaturmorphismus, falls für alle* $f \in F_1$

$$\tau_2(hf) = g(\tau_1 f)$$

(g *fortgesetzt auf* $g : S_1^* \to S_2^*$).

im Bsp. g : elem $\to$ nat h :: next $\to$ suc

oder $\sigma : sig_{\text{BOOL}} \to sig_{\text{NAT}}$ *mit*

g ::	*bool* $\to$ nat	h ::	not $\to$ suc
h ::	*true* $\to$ 0		*and* $\to$ plus
	false $\to$ 0		*or* $\to$ times

Signaturmorphismen - Parameterübergabe

b) spec = Body[Formal] Parameterspezifikation
Eine Parameterübergabe ist ein Signaturmorphismus σ : sig(Formal) $\longrightarrow$ sig(Actual) wobei Actual eine Spezifikation ist: Die aktuelle Parameterspezifikation.

(Actual, σ) definiert eine Spezifikation VALUE durch folgende Änderungen von Body:

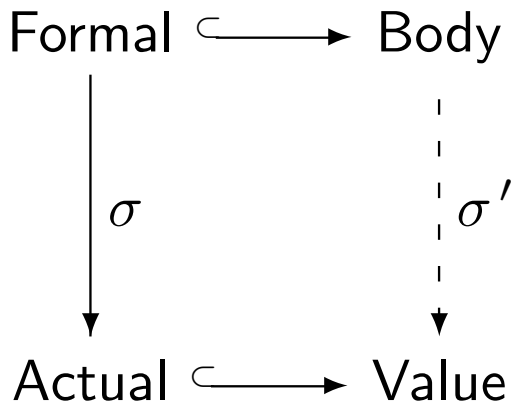
1) Ersetze Formal durch Actual: Body[Actual].
2) Ersetze in $op : s_1 \ldots s_n \longrightarrow s_0 \in$ Body, das nicht in Formal auftritt jedes s_i, $s_i \in$ Formal durch $\sigma(s_i)$.
3) Ersetze in jeder Gleichung $L = R$ aus Body, die nicht in Formal ist, jedes $o_P \in$ Formal durch $\sigma(o_P)$.
4) Fasse jede Variable einer Sorte s mit $s \in$ Formal als Variable der Sorte $\sigma(s)$ auf.
5) Vermeide Namenskonflikte zwischen Actual und Body/Formal.

Parameterübergabe

Bezeichnung:

$$\text{Value} = \text{Body}[\text{Actual}, \sigma]$$

Man erhält somit für σ : sig(Formal) $\longrightarrow$ sig(Actual) einen Signaturmorphismus
σ' : sig(Body[Formal]) $\longrightarrow$ sig(Body[Actual, σ]



Signaturmorphismen (Forts.)

$$\sigma'(x) = \begin{cases} \sigma(x) & x \in \text{Formal} \\ x' & x \notin \text{Formal} \end{cases}$$

x' ist dabei eine Umbenennung, wenn Konflikte.
Beachte: $\sigma : \text{sig}' \to \text{sig}$ Signaturmorphismus, dann ist für jede sig-Algebra $\mathfrak{A}$, $\mathfrak{A}|_\sigma$ eine sig'-Algebra, wobei für $\text{sig}' = (S', F', \tau')$ $(A|_\sigma)_s = A_{\sigma(s)}$ $s \in S'$ und $f_{\mathfrak{A}|_\sigma} = \sigma(f)_{\mathfrak{A}}$ $f \in F'$.
$\mathfrak{A}|_\sigma$ heißt Vergißbild von $\mathfrak{A}$ entlang σ (Spezialfall: $\text{sig}' \subseteq \text{sig}$ $:\hookrightarrow$) $|_{\text{sig}'}$

Beispiel

Beispiel 7.6 $\mathfrak{A} = T_{\mathsf{NAT}}$ *(mit* $0, \mathsf{suc}, \mathsf{plus}, \mathsf{times}$*)*
$sig' = sig(\mathsf{BOOL})$ $sig = sig(\mathsf{NAT})$
$\sigma : sig' \to sig$ *wie oben definiert.*

$$\begin{aligned}((T_{\mathsf{NAT}})|_{sig})_{\mathsf{bool}} &= (T_{\mathsf{NAT}})_{\sigma(\mathsf{bool})} = (T_{\mathsf{NAT}})_{\mathsf{nat}} \\ &= \{[0], [\mathsf{suc}(0)], \ldots\}\end{aligned}$$

Beispiel

$$
\begin{array}{lcl}
\mathsf{true}_{(T_{\mathsf{NAT}})|_\sigma} & = & \sigma(\mathsf{true})_{T_{\mathsf{NAT}}} = [0] \\
\mathsf{false}_{(T_{\mathsf{NAT}})|_\sigma} & = & \sigma(\mathsf{false})_{T_{\mathsf{NAT}}} = [0] \\
\mathsf{not}_{(T_{\mathsf{NAT}})|_\sigma} & = & \sigma(\mathsf{not})_{T_{\mathsf{NAT}}} = \mathsf{suc}_{T_{\mathsf{NAT}}} \\
\mathsf{and}_{(T_{\mathsf{NAT}})|_\sigma} & = & \sigma(\mathsf{and})_{T_{\mathsf{NAT}}} = \mathsf{plus}_{T_{\mathsf{NAT}}}
\end{array}
$$

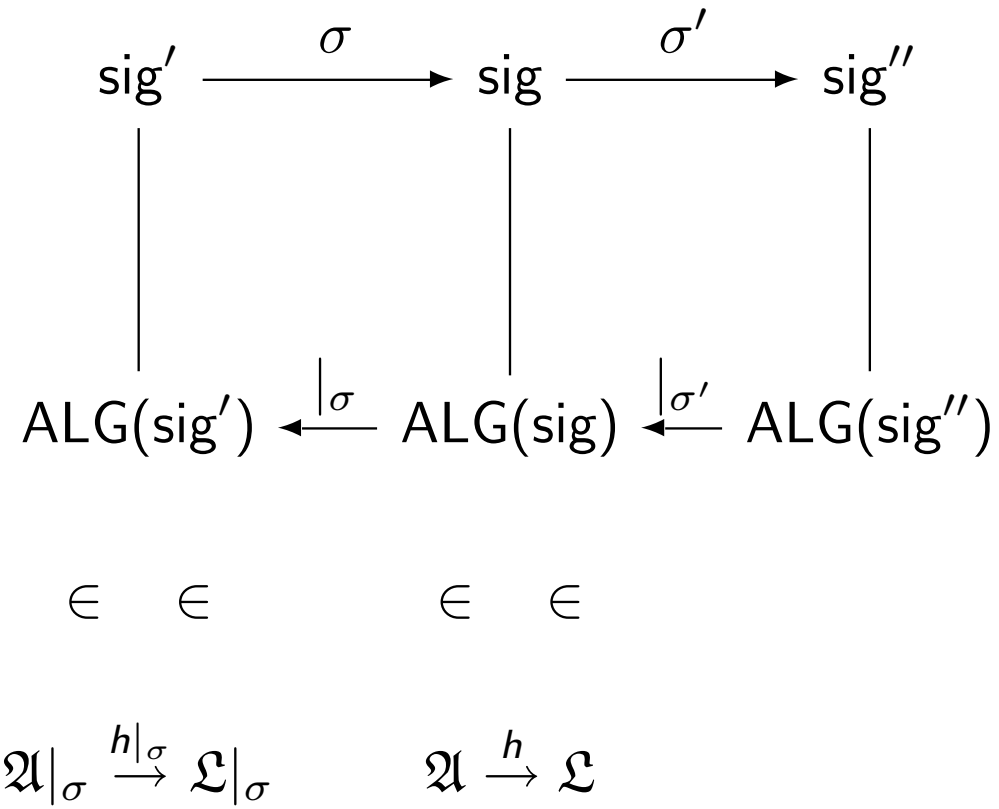
Beachte: Ist $\sigma : \mathsf{sig}' \longrightarrow \mathsf{sig}$ Signaturmorphismus, $\mathfrak{A}, \mathfrak{L}$ sig-Algebren und $h : \mathfrak{A} \longrightarrow \mathfrak{L}$ sig-Homomorphismus, dann ist

$h|_\sigma := \{h_{\sigma(s)} \mid s \in S'\}$, wobei $\mathsf{sig}' = (S', F', \tau')$ ein sig'-Homomorphismus: $(h|_\sigma)_s = h_{\sigma(s)}$:

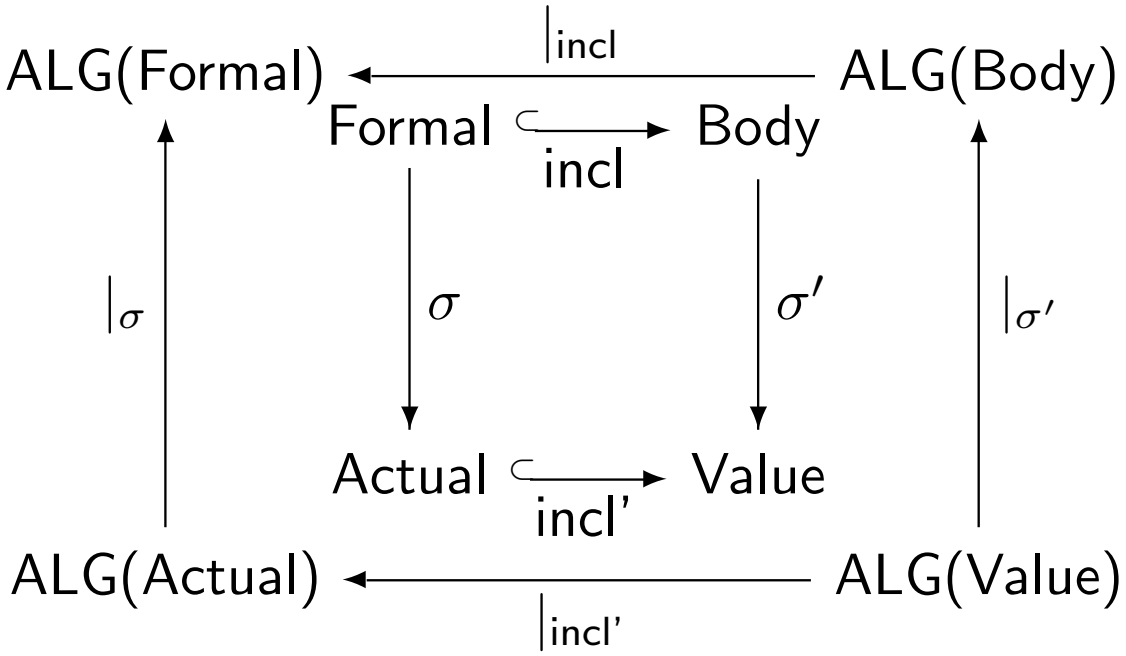
$$
\begin{array}{ccc}
A_{\sigma(s)} & \longrightarrow & B_{\sigma(s)} \\
\| & & \| \\
(A|_\sigma)_s & \longrightarrow & (B|_\sigma)_s
\end{array}
$$

Beispiel (Forts.)

$h|_\sigma$ heißt Vergißbild von h entlang σ



Parameterspezifikation



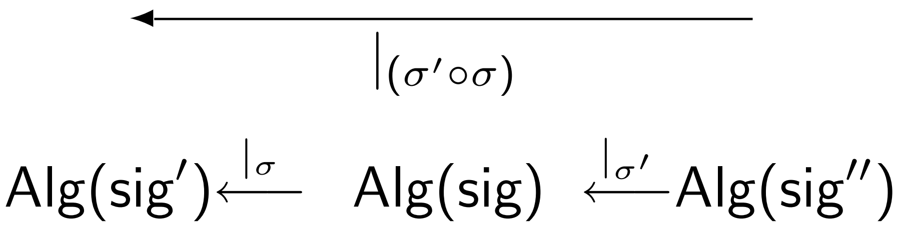
Parameterspezifikation

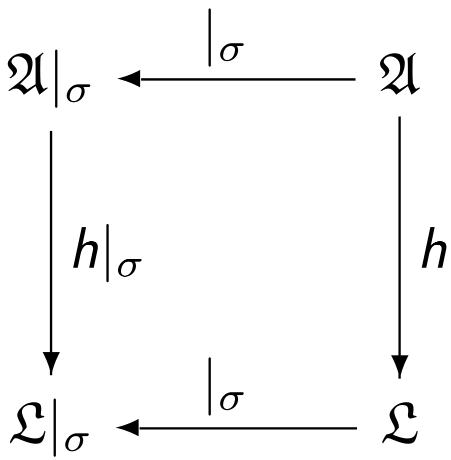
σ : sig$'$ $\rightarrow$ sig, $\mathfrak{A}, \mathfrak{L}$, sig-Algebren.
$h : \mathfrak{A} \rightarrow \mathfrak{L}$, sig-Homomorphismus.
$h\sigma_\sigma = \{h_{\sigma(s)} \mid s \in S'\}$, sig$' = (S', F', \tau') =$ mit
$h\sigma_\sigma : A|_\sigma \rightarrow B|_\sigma$ Vergißbild von h entlang σ.

$$\text{sig}' \xrightarrow{\sigma} \text{sig} \xrightarrow{\sigma'} \text{sig}''$$

$$\text{sig}' \xrightarrow{\sigma' \circ \sigma} \text{sig}''$$

Parameterspezifikation





Semantik der Parameterübergabe (nur Signatur)

Definition

Sei Body[Formal] Parameterspezifikation. σ : Formal $\rightarrow$ Actual Signaturmorphismus. Semantik der Parameterübergabe.

Zuordnung: σ : Formal $\rightarrow$ Actual

$\downarrow$

initiale Semantik von Value. D. h.

$T_{\text{Body[Actual},\sigma]}$

$S :: (T_{\text{Actual}}, \sigma) \mapsto T_{\text{Body[Actual},\sigma]}$

Abbildung zwischen init Algebren

Kann aufgefasst werden als Zuordnung zwischen Formal Algebren $\longrightarrow$ Body-Algebren.
$(T_{\text{Actual}}|_{\sigma} \mapsto (T_{\text{Body[Actual},\sigma]})|_{\sigma'}$

Abbildung zwischen init Algebren

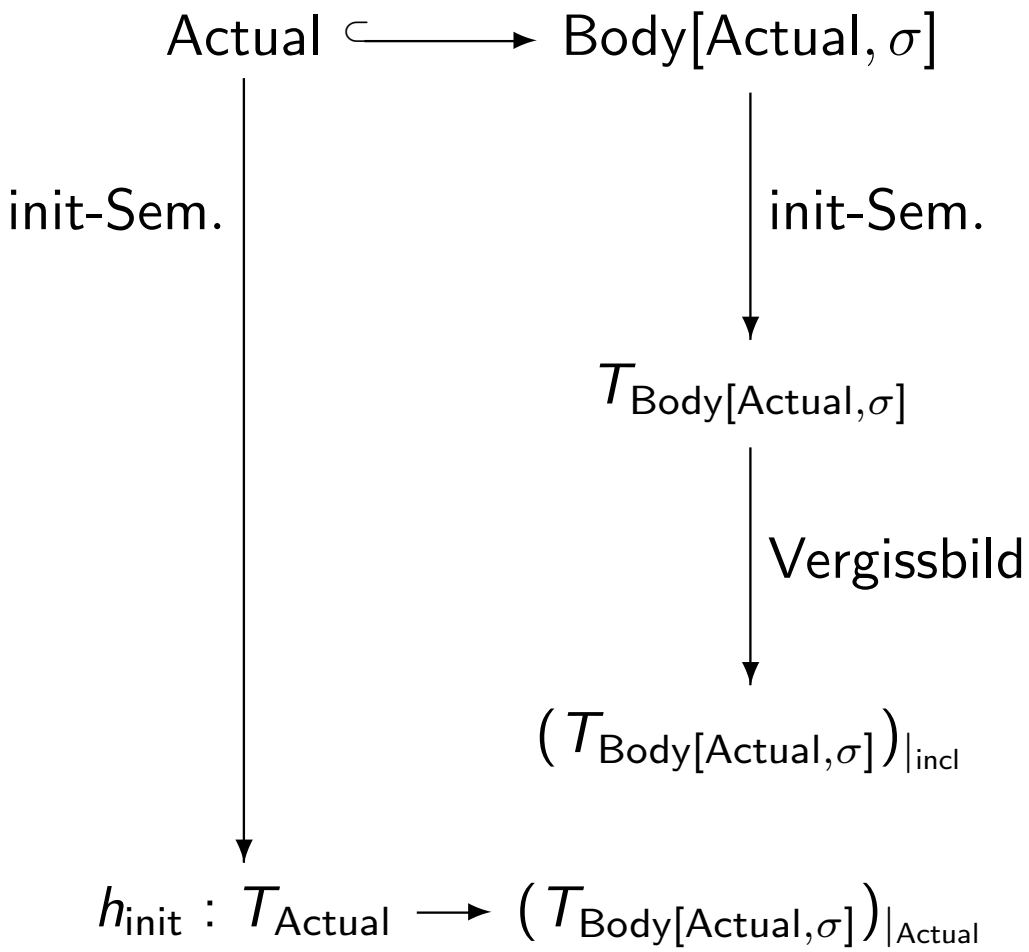


Abbildung zwischen init Algebren

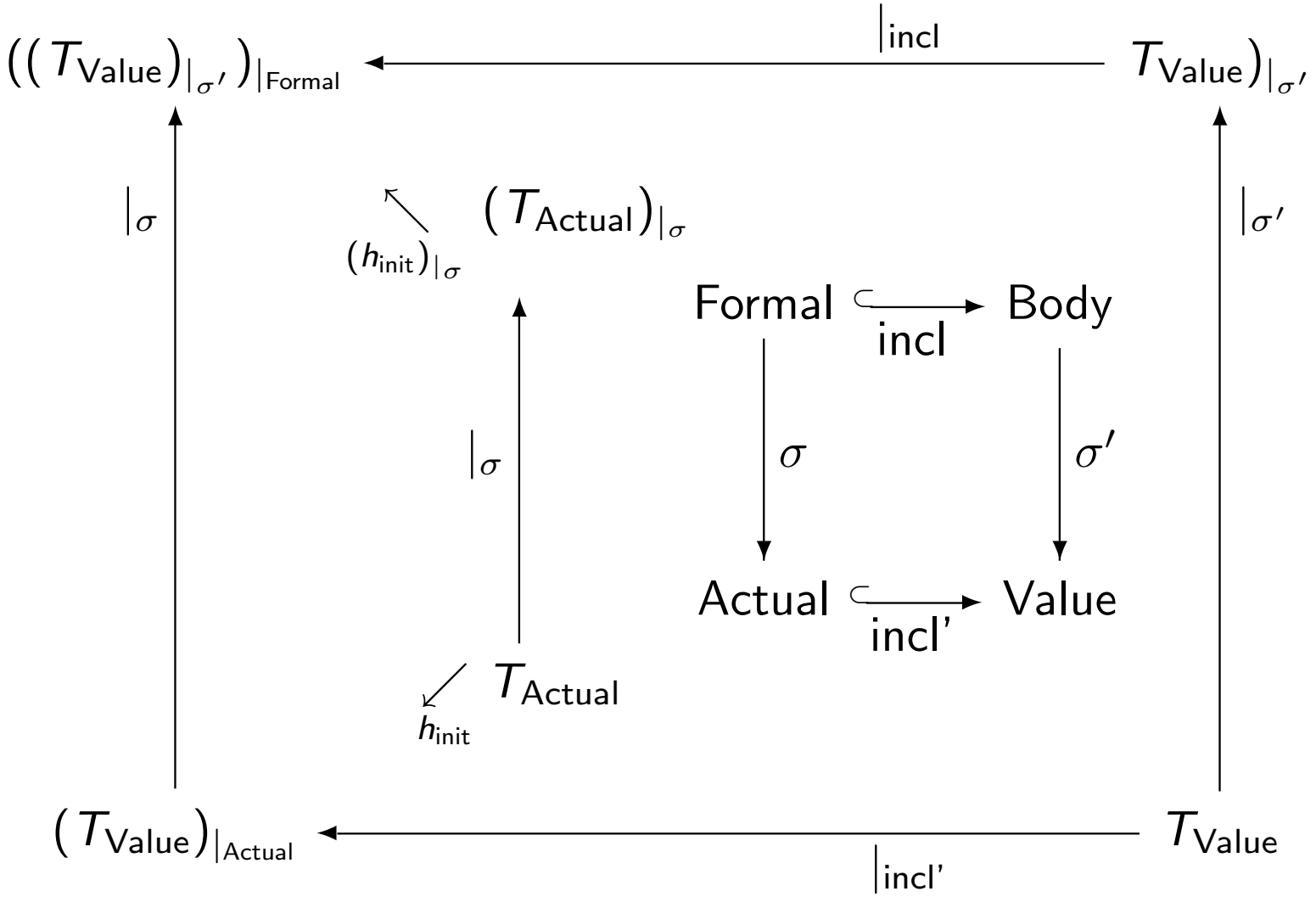


Abbildung zwischen init Algebren

Formal		elem $\longrightarrow$ nat	Actual	
sorts	elem	$\xrightarrow{\sigma}$	sorts	nat
ops	$a, b :\longrightarrow$ elem	$a \rightarrow 0$	ops	$0, 1 :\longrightarrow$ nat
eqns	$a = b$	$b \rightarrow 1$		

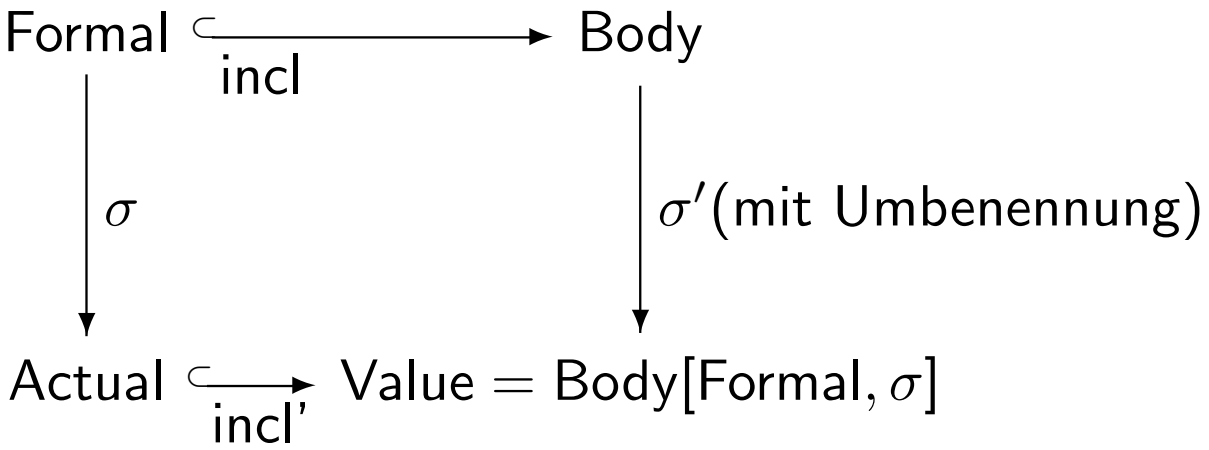
$$\mathfrak{A} = T_{\text{Actual}} \quad A_{\text{nat}} = \{0, 1\}$$

$\mathfrak{A}|_\sigma \in \text{Alg}(\text{sig Formal})$ $(A|_\sigma)_{\text{elem}} = \{0, 1\}$
$a|_{\mathfrak{A}_{|\sigma}} = 0 \neq 1 = b|_{\mathfrak{A}_{|\sigma}}$
Gleichung von Formal nicht erfüllt! D. h. $\mathfrak{A}|_\sigma \notin \text{Alg(Formal)}$.

Parameterübergabe (Actual, σ)

Body[Formal]

$$\sigma : \text{sig(Formal)} \rightarrow \text{sig(Actual)}$$
Signatur Morphismus



Vor: sig(Actual) und sig(Value) strikt.

Parameterübergabe (Actual, σ)

Vergißbilder: $|_\sigma : \mathrm{Alg}(\mathrm{sig}) \to \mathrm{Alg}(\mathrm{sig}')$

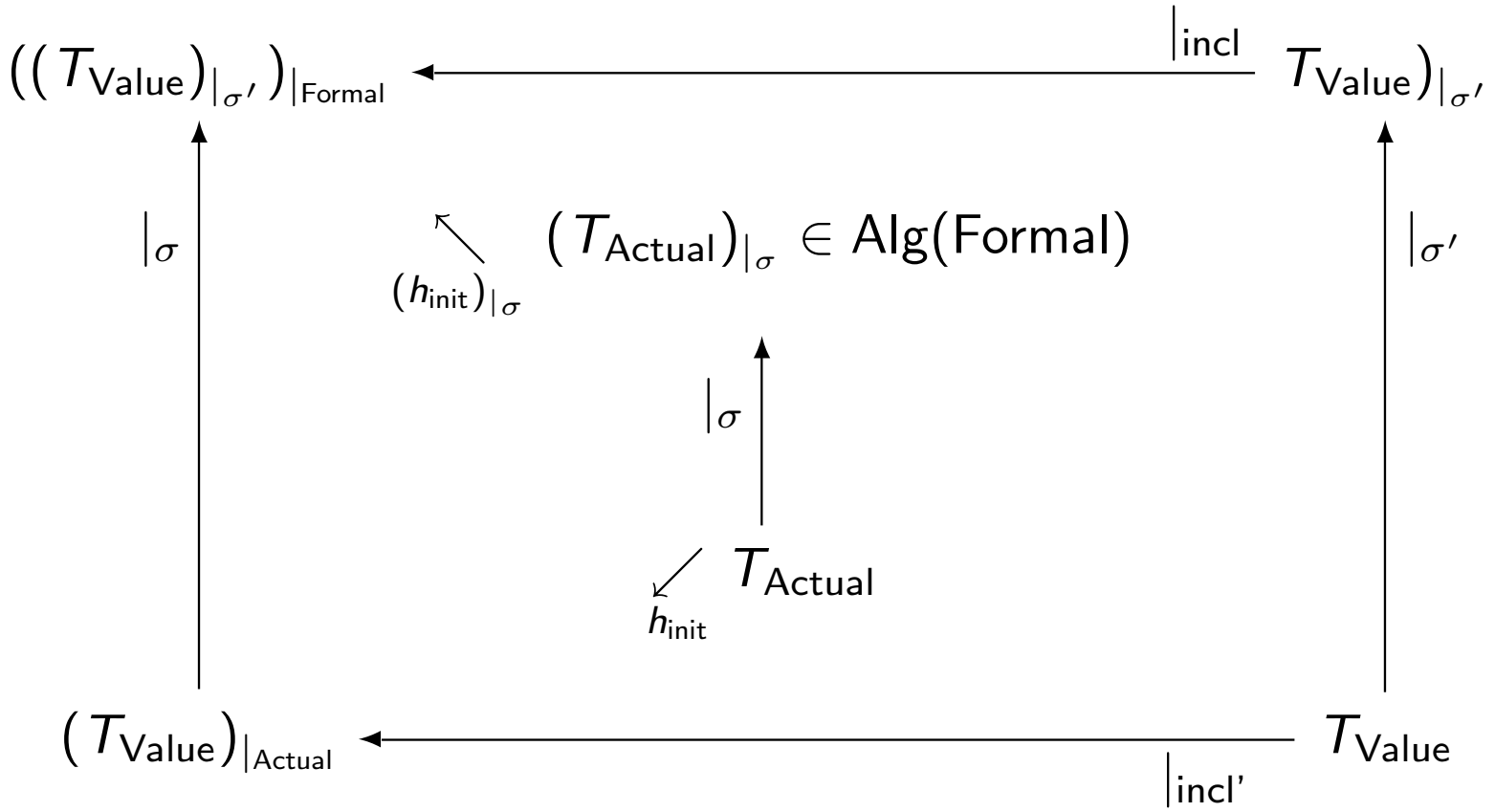
$$\mathfrak{A}|_\sigma \text{ für } \sigma : \mathrm{sig}' \to \mathrm{sig}$$

$h : \mathfrak{A} \to \mathfrak{L}$ sig-Homomorphismus

$$h|_\sigma : \mathfrak{A}|_\sigma \to \mathfrak{L}|_\sigma$$

sig′-Homomorphismus

Parameterübergabe (Actual, σ)



Probleme: 1) $(T_{\text{Actual}})|_{\sigma} \notin \text{Alg}(\text{Formal})$, 2) h_{init} keine Bijektion.

Spezifikationsmorphismus

Definition

Seien $\text{spec}' = (\text{sig}', E'), \text{spec} = (\text{sig}, E)$ (allg.) Spezifikationen. Ein Signaturmorphismus $\sigma : \text{sig}' \longrightarrow \text{sig}$ heißt Spezifikationsmorphismus, falls für alle $s = t \in E'$ gilt $\sigma(s) = \sigma(t) \in Th(E)$.

Schreibe: $\sigma : \text{spec}' \longrightarrow \text{spec}$

Fakt: Für $\mathfrak{A} \in \text{Alg}(\text{spec})$ gilt $\mathfrak{A}|_\sigma \in \text{Alg}(\text{spec}')$

D. h. $|_\sigma : \text{Alg}(\text{spec}) \longrightarrow \text{Alg}(\text{spec}')$!

Oft verlangt man „nur“ $\sigma(s) = \sigma(t) \in ITh(E)$.!

Spezifikationsmorphismus

Eine Parameterübergabe für Body[Formal] ist ein Paar (Actual, σ):
Actual Spezifikation und σ : Formal $\rightarrow$ Actual Spezifikationsmorphismus.
$(T_{\text{Actual}})|_{\sigma} \in \text{Alg}(\text{Formal})$
- verlange auch h_{init} Bijektion, syntaktische Einschränkungen.
Spezifikationssprachen
CLEAR, Act-one, -Cip-C, Affirm, ASL, Aspik, OBJ, ASF, $\overset{\rightsquigarrow}{+}$ neuere Sprachen: - Spectrum, - Troll.

Beispiel

Beispiel 7.7

$$\text{Formal} :: \begin{cases} \textit{spec} & \text{ELEMENT} \\ \text{use} & \text{BOOL} \\ \text{sorts} & \text{elem} \\ \text{ops} & . \leq . : \text{elem}, \text{elem} \rightarrow \text{bool} \\ \text{eqns} & x \leq x = \textit{true} \\ & \text{imp}(x \leq x \ \textit{and}\ y \leq z, x \leq z) = \textit{true} \\ & x \leq y \ \textit{or}\ y \leq x = \textit{true} \end{cases}$$

Beispiel (Forts.)

spec	LIST[ELEMENT]
use	ELEMENT
sorts	list
ops	$nil :\longrightarrow list$
	$\cdot : elem, list \longrightarrow list$
	$insert : elem, list \longrightarrow list$
	$case : bool, list, list \longrightarrow list$
	$sorted : list \longrightarrow bool$

Beispiel (Forts.)

$$
\begin{aligned}
\text{eqns}\quad & \mathsf{insert}(x, \mathsf{nil}) = x.\mathsf{nil} \\
& \mathsf{insert}(x, y.l) = \mathsf{case}(x \leq y, x.\,\mathsf{insert}(y, l), y.\,\mathsf{insert}(x, l)) \\
& \mathsf{case}(\mathsf{true}, l_1, l_2) = l_1 \\
& \mathsf{case}(\mathsf{false}, l_1, l_2) = l_2 \\
& \mathsf{sorted}(\mathsf{nil}) = \mathsf{true} \\
& \mathsf{sorted}(x, \mathsf{nil}) = \mathsf{true} \\
& \mathsf{sorted}(x, y, l) = \mathsf{if}\ x \leq y\ \mathsf{then}\ \mathsf{sorted}(y, l)\ \mathsf{else}\ \mathsf{false}
\end{aligned}
$$

Eigenschaft: $\mathsf{sorted}(\mathsf{insert}(x, l)) = \mathsf{true}$

Beispiel (Forts)

ACTUAL $\equiv$ BOOL

σ : elem $\longrightarrow$ bool, bool $\longrightarrow$ bool

. $\leq$. $\longrightarrow$ impl

Die Gleichungen von ELEMENT sind in Th(BOOL)

$\rightsquigarrow$ Spezifikationsmorphismus

Beispiel (Forts.)

ACTUAL $\equiv$ NAT

σ :

bool $\longrightarrow$ nat	elem $\longrightarrow$ nat
true $\longrightarrow$ suc(0)	nicht erlaubt
false $\longrightarrow$ 0	
not $\longrightarrow$ suc	
or $\longrightarrow$ plus	
and $\longrightarrow$ times	
$\vdots$	
$. \leq . \longrightarrow \cdots$	

kein Spezifikationsmorphismus
not(false) = true
not(true) = false gilt nicht!.