Distributed Termination Detection

Gilt die Invariante auch in allen Zuständen aller Linearisierungen von Läufen der DASM ? **Nein**

- **Problem 1** Rot Färben einer aktiven Maschine die eine Nachricht versendet geschieht in späteren Zustand. Es müsste im selben Zustand passieren in dem die nachrichterhaltende Maschine aktiv wird.
  Lösung *color* ist shared Funktion. Anstatt $SendMessageEvent(m)$ zu setzen wird $color(m) = red$ von der Umgebung gesetzt.
- **Problem 2** Es gibt Zustände für die keine Maschine das Token hat. Die tokenhabende Maschine Initialisiert sich und setzt ein Event, im resultierenden Zustand hat keine Maschine das Token.
  Lösung Statt ein *FarbTokenEvent* zu setzen, wird direkt $token(next(m))$ richtig gesetzt.
- **Resultat** Abstraktere Maschine. Die Umgebung regelt die Aktivität der Maschinen, das Nachrichtenübermitteln und die Färbung.

Verfeinerungsbegriffe für ASM's

Frage: Ist im Terminierungsbeispiel die angegebene DASM eine Verfeinerung der abstrakteren DASM? $\rightsquigarrow$

Allgemeine Verfeinerungsbegriffe für ASM's

- Verfeinerungen werden in der Regel für BASM definiert, d.h. Läufe sind stets linear, was die Betrachtung vereinfacht.
- Verfeinerungen erlauben Abstraktionen, Realisierungen von Daten und Prozeduren.
- ASM Verfeinerungen sind meistens Problemorientiert: Abhängigkeit von der Anwendung und somit sollten sie flexibel sein.
- Beweisaufgaben werden mit Hilfe von korrekten und vollständigen Verfeinerungen Strukturiert und Vereinfacht.

Siehe ASM-Buch.

Algebraische Spezifikation - Gleichheitskalküle

Algebraische Spezifikation - Gleichheitslogik

Anforderungen an Spezifikationstechniken:

- Abstraktion (Verfeinerung)
- Strukturierungsmechanismen
  Zerlegung, Kombination, Erweiterung-Instantierung
- klare (eindeutige und plausible) Semantik
- Unterstützung des „verify while develop“-Prinzips
- Ausdruckskraft (alle partial rekursiven Funktionen darstellbar)
- Lesbarkeit (Adequatheit) (Angemessenheit)

  ⋮

Algebraische Spezifikation - Gleichheitskalküle

Algebraische Spezifikation - Algebren

Spezifikation von Datentypen

Syntax	Gleichungen	Programme
Signatur	$t_1 = t_2$	Datenoperation
Axiome	if φ then $t_1 = t_2$	Gerichtete Anwendung

Algebren

heterogene	ordnungssortierte	homogene
(mehrsortig)	(mehrsortig)	(einsortig)

Einsortige Algebren

Beispiel 6.1 *a) Gruppen*

SORT:: g

SIG:: $\cdot : g, g \to g \qquad 1 :\to g \qquad {}^{-1} : g \to g$

EQN:: $x \cdot 1 = x \qquad x \cdot x^{-1} = 1 \qquad (x \cdot y) \cdot z = x \cdot (y \cdot z)$

All-quantifizierte Gleichungen

Modelle sind Gruppen

Frage: Welche Gleichungen gelten in allen Gruppen,
d. h. $EQN \models t_1 = t_2$

$1 \cdot x = x \qquad x^{-1} \cdot x = 1 \qquad (x^{-1})^{-1} = x$

Einsortige Algebren

Gleicheitslogik: Ersetze „Gleiches“ durch „Gleiches“
Problem: Zykel, Nichtterminierung
Lösung: Gerichtete Gleichungen, $\rightsquigarrow$ Termersetzungssysteme

Finde R konvergent mit $\underset{EQN}{=} = \overset{*}{\underset{R}{\Longleftrightarrow}}$

$$
\begin{array}{ll}
x \cdot 1 \to x & 1 \cdot x \to x \\
x \cdot x^{-1} \to 1 & x^{-1} \cdot x \to 1 \\
1^{-1} \to 1 & (x^{-1})^{-1} \to x \\
(x \cdot y)^{-1} \to y^{-1} \cdot x^{-1} & (x \cdot y) \cdot z \to x \cdot (y \cdot z) \\
x^{-1} \cdot (x \cdot y) \to y & x \cdot (x^{-1} \cdot y) \to y
\end{array}
$$

Mehrsortige Algebren

b) Listen über nat-Zahlen

SIG: BOOL, NAT, LIST — Sorten

true, false: $\rightarrow$ BOOL

0 $\rightarrow$ NAT suc: NAT $\rightarrow$ NAT

+: NAT, NAT $\rightarrow$ NAT

eq: NAT, NAT $\rightarrow$ BOOL

nil: $\rightarrow$ LIST . : NAT, LIST $\rightarrow$ LIST

app: LIST, LIST $\rightarrow$ LIST

rev: LIST $\rightarrow$ LIST

Mehrsortige Algebren

Axiome sind allquantifizierte Gleichungen, d.h.
$\forall x_1, ..., x_n : \quad t_1(x_1, ..., x_n) = t_2(x_1, ..., x_n)$ wobei

$t_1(x_1, ..., x_n), t_2(x_1, ..., x_n)$ Terme gleicher Sorte in der Signatur sind.

EQN : $n + 0 = n \quad n_1 + \text{suc}(n_2) = \text{suc}(n_1 + n_2)$
$\text{eq}(0, 0) = \text{true} \quad \text{eq}(0, \text{suc}(n)) = \text{false}$
$\text{eq}(\text{suc}(n), 0) = \text{false}$
$\text{eq}(\text{suc}(n_1), \text{suc}(n_2)) = \text{eq}(n_1, n_2)$
$\text{app}(\text{nil}, l) = l \quad \text{app}(n.l_1, l_2) = n.\,\text{app}(l_1, l_2)$
$\text{rev}(\text{nil}) = \text{nil} \quad \text{rev}(n.l) = \text{app}(\text{rev}(l), n.\text{nil})$

Mehrsortige Algebren

Terme der Sorten BOOL, NAT, LIST als Bezeichner für Elemente (Standarddefinition!)
Welche Algebra wird spezifiziert? Wie rechnet man in dieser Algebra?
Gleichungen richten $\rightsquigarrow$ Termersetzungssystem R. Offenbar gilt:

$$s^i(0) + s^j(0) \xrightarrow[R]{*} s^{i+j}(0)$$

$$\mathrm{app}(3.1.\mathrm{nil}, \mathrm{app}(5.\mathrm{nil}, 1.2.3.\mathrm{nil})) \xrightarrow[R]{*} 3.1.5.1.2.3.\mathrm{nil}$$

$$\begin{aligned} \mathrm{rev}(3.1.\mathrm{nil}) &\longrightarrow \mathrm{app}(\mathrm{rev}(1.\mathrm{nil}), 3.\mathrm{nil}) \\ &\longrightarrow \mathrm{app}(\mathrm{app}(\mathrm{rev}(\mathrm{nil}), 1.\mathrm{nil}), 3.\mathrm{nil}) \\ &\longrightarrow \mathrm{app}(\mathrm{app}(\mathrm{nil}, 1.\mathrm{nil}), 3.\mathrm{nil}) \\ &\longrightarrow \mathrm{app}(1.\mathrm{nil}, 3.\mathrm{nil}) \xrightarrow{*} 1.3.\mathrm{nil} \end{aligned}$$

Frage: Gilt $\mathrm{app}(x.y.\mathrm{nil}, z.\mathrm{nil}) =_E \mathrm{app}(x.\mathrm{nil}, y.z.\mathrm{nil})$?

Mehrsortige Algebren

Manche Gleichungen gelten nicht in allen Modellen von EQN$=E$.
z. B.

$$x + y \neq_E y + x$$
$$\text{app}(x, \text{app}(y, z)) \neq_E \text{app}(\text{app}(x, y), z)$$
$$\text{rev}(\text{rev}(x)) \neq_E x$$

die Termpaare sind nicht zusammenführbar.

Unterscheidung:

- Gleichungen die in allen Modellen von E gelten.
- Gleichungen die im Datenmodell von E gelten.

$x + y = y + x :: s^i 0 + s^j 0 = s^j 0 + s^i 0$ alle i, j
$\text{rev}(\text{rev}(x)) = x$ für $x \equiv s^{i_1} 0.s^{i_2} 0. \ldots . s^{i_n} 0.\text{nil}$

These: Datentypen sind Algebren

ADT: Abstrakte Datentypen. Unabhängig von der Repräsentatioon der Daten.

Spezifikation abstrakter Datentypen:

Konzepte aus Logik/universelle Algebra
Ziel: gemeinsame Sprachebene für Spezifikation und Implementierung.

Mittel für Korrektheitsbeweise:

Syntax, L Formeln (P-Logik,Hoare,...)

Cl: Folgerungsabschluß (Z.B. $\models$, $Th(A)$,...)

Folgerungssbschluß

$Cl : \mathbb{P}(L) \to \mathbb{P}(L)$ (Teilmengen von L) mit

a) $A \subset L \rightsquigarrow A \subset Cl(A)$
b) $A, B \subset L, A \subseteq B \rightsquigarrow Cl(A) \subseteq Cl(B)$ (Monotonie)
c) $Cl(A) = Cl(Cl(A))$ (Maximalität)

Wichtige Begriffe:

Konsistenz: $A \subsetneq L$ A ist konsistent falls $Cl(A) \subsetneq L$
Implementierung: A implementiert B

(Verfeinerung) $L \subset L', Cl(B) \subseteq Cl(A)$

Signatur - Terme

Definition 6.1 *a) Signatur ist Tripel* $\mathrm{sig} = (S, F, \tau)$ *(Abgekzt.:* Σ*)*

- S *endliche Menge von Sorten*
- F *Menge von Operatoren (Funktionssymbolen)*
- $\tau : F \to S^+$ *Funktion Stelligkeit, d. h.*
  $\tau(f) = s_1 \cdots s_n s$, $n \geq 0$, s_i *Argumentsorten*, s *Zielsorte.*

Schreibe: $f : s_1, \ldots, s_n \to s$

(Beachte $n = 0$*) möglich, Konstante der Sorte* S*.*

Signatur - Terme

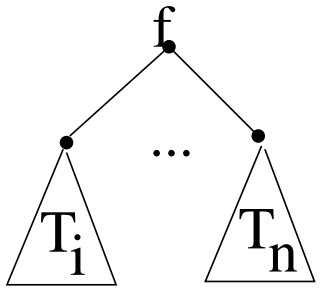
b) Term(F): Menge der Grundterme über sig und ihre Baumpräsentation.

$$\mathrm{Term}(F) := \dot{\bigcup_{s \in S}} \mathrm{Term}_s(F)$$

rekursiv def.

- $f :\longrightarrow s$, so $f \in \mathrm{Term}_s(F)$ Repräsentation: $\cdot f$
- $f : s_1, \ldots, s_n \longrightarrow s$, $t_i \in \mathrm{Term}_{s_i}(F)$ mit Rep. T_i so $f(t_1, \ldots, t_n) \in \mathrm{Term}_s(F)$ mit Rep.

Beachte Repräsentation durch geordnete Bäume



Signatur - Terme

c) $V = \dot{\bigcup_{s \in S}} V_s$ Variablensystem $V \cap F = \varnothing$.

Jedes $x \in V_s$ hat Funktionalität $x :\longrightarrow s$

Setze: $\mathrm{Term}(F, V) := \mathrm{Term}(F \cup V)$.

Bezeichnung: Terme über sig in Var. V.
(F und τ geeignet erweitert).

Intention: Für Variablen dürfen beliebige Objekte, d.h. Terme, eingesetzt werden.

Striktheit - Positionen-Teilterme

Definition 6.2 *a)* $s \in S$ *strikt, falls* $\mathrm{Term}_s(F) \neq \varnothing$
Gibt es zu jeder Sorte $s \in S$ *entweder eine Konstante der Sorte* S *oder eine Funktion* $f : s_1, \ldots, s_n \longrightarrow s$, *so dass die* s_i *strikt sind, dann sind alle Sorten der Signatur strikt.*$\rightsquigarrow$ *strikte Signaturen (Generalvoraussetzung)*

b) Teilterme $(t) = \{t_p \mid p$ *Stelle (Position) in* p, t_p *Teilterm in* $p\}$
Stellen werden durch Folgen über $\mathbb{N}$ *dargestellt*
(Elemente von $\mathbb{N}^*$, e *leere Folge).*
$O(t)$ *Menge der Stellen in* t,
Für $p \in O(t)$ t_p *(oder* $t|_p$*) Teilterm von* t *an Stelle* p

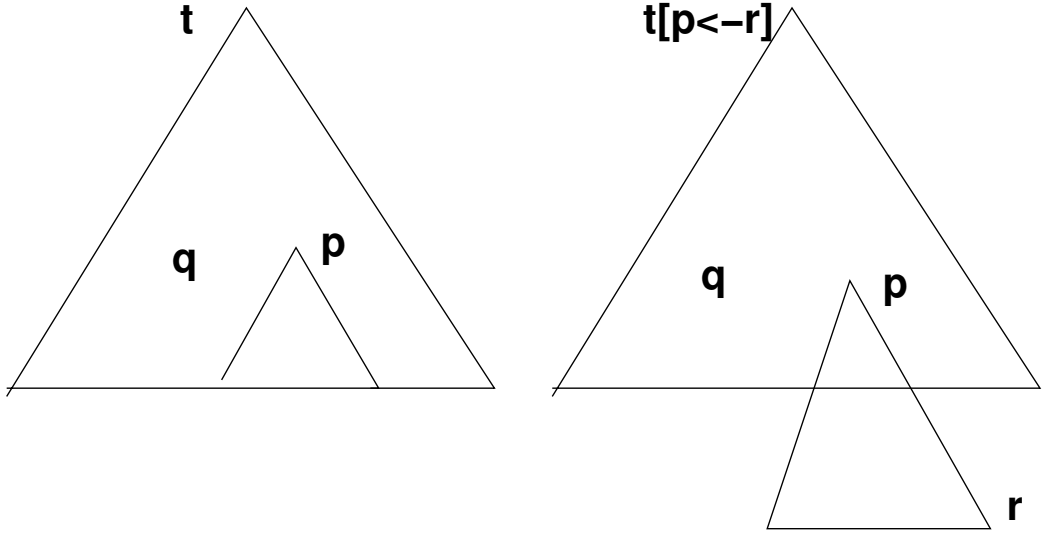
- t *Konstante oder Variable:* $O(t) = \{e\} \quad t_e \equiv t$
- $t \equiv f(t_1, \ldots, t_n)$ *so*
  $O(t) = \{ip \mid 1 \leq i \leq n, p \in O(t_i)\} \cup \{e\}$
  $t_{ip} \equiv t_i|_p$ *und* $t_e \equiv t$.

Termersetzung

c) Termersetzung: $t, r \in \mathrm{Term}(F, V)$
$p \in O(t)$: mit $r, t_p \in \mathrm{Term}_s(F, V)$ für eine Sorte s.

Dann ist
$t[r]_p$, bzw. $t[p \leftarrow r]$ bzw. t_p^r der Term der aus t entsteht durch Ersetzen vom Teilterm t_p durch r.

Also $t[p \leftarrow r]_q = t_q$ für $q \mid p$ und



$t[p \leftarrow r]_p = r$

Signaturen - Terme

Beispiel 6.2 $S = (\mathrm{BOOL}, \mathrm{NAT}, \mathrm{LIST})$, $F = \{true, false, \ldots\}$,
$\tau : F \to S^* :: true :\to \mathrm{BOOL}$, $eq : \mathrm{NAT}, \mathrm{NAT} \to \mathrm{BOOL} \ldots$

$$V = \underbrace{V_{\mathrm{BOOL}}}_{\{b_i : i \in \mathbb{N}\}} \cup \underbrace{V_{\mathrm{NAT}}}_{\{x_i : i \in \mathbb{N}\}} \cup \underbrace{V_{\mathrm{LIST}}}_{\{l_i : i \in \mathbb{N}\}}$$

Grundterme:

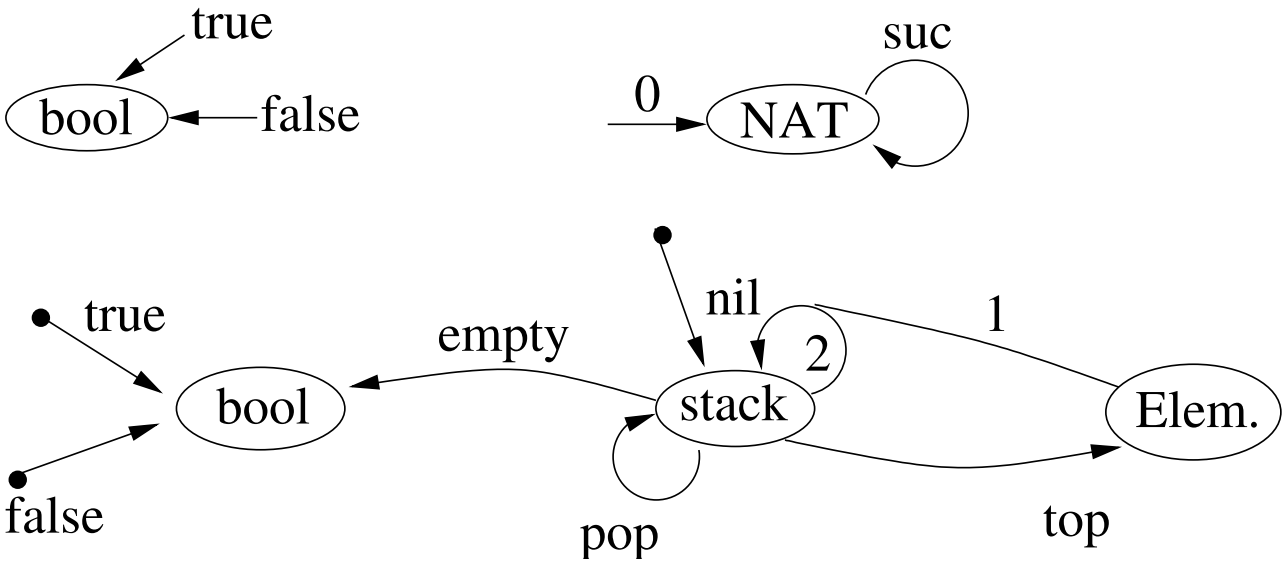
$true, false, eq(0, \mathrm{suc}(0)) \in \mathrm{Term}_{\mathrm{BOOL}}(S)$
$0, \mathrm{suc}(0), \mathrm{suc}(0) + (\mathrm{suc}(\mathrm{suc}(0)) + 0) \in \mathrm{Term}_{\mathrm{NAT}}(S)$
$\mathrm{app}(nil, \mathrm{suc}(0), (\mathrm{suc}(\mathrm{suc}(0)).nil) \in \mathrm{Term}_{\mathrm{LIST}}(S)$
$0.\,\mathrm{suc}(0), eq(true, false), \mathrm{rev}(0)$ *keine Terme.*

Allgemeine Terme:

$eq(x_1, x_2) \in \mathrm{Term}_{\mathrm{BOOLE}}(F, V), suc(x_1) + (x_2 + \mathrm{suc}(0)) \in \mathrm{Term}_{\mathrm{NAT}}(F, V)$
$\mathrm{app}(l_1, x_1.l_0) \in \mathrm{Term}_{\mathrm{LIST}}(F, V)$
$\mathrm{rev}(x_1.l) \in \mathrm{Term}_{\mathrm{LIST}}(F, V)$
$\mathrm{app}(x_1, l_2)$ *kein Term.*

Signaturen

Darstellung von Signaturen (graphisch oder normiert)



Notationen:

sig ...
sorts ...
ops ...
op: $W \rightarrow S$
$op_1, \ldots, op_i : W \rightarrow S$

Interpretationen: sig-Algebren

Definition 6.3 $\mathrm{sig} = (S, F, \tau)$ *Signatur. Eine sig-Algebra* $\mathfrak{A}$ *besteht aus*

1) *Trägermenge* $A = \bigcup_{s \in S} A_s, A_s \neq \varnothing$ *Trägermenge der Sorte* s.
2) *Funktionensystem* $F_{\mathfrak{A}} = \{f_{\mathfrak{A}} : f \in F\}$ *mit*
   $f_{\mathfrak{A}} : A_{s_1} \times \cdots \times A_{s_n} \to A_s$ *Funktion und* $\tau(f) = s_1 \cdots s_n s$.

Beachte: Die $f_{\mathfrak{A}}$ sind totale Funktionen.

Die Voraussetzung $A_s \neq \varnothing$ ist nicht zwingend.

Interpretationen: sig-Algebren

Beispiel 6.3 *a)* sig $\equiv$ BOOL-*Algebren*, *true*, *false* $:\longrightarrow$ BOOL

$$\left.\begin{array}{llll} \mathfrak{A}_1 & \{0,1\} & true_{\mathfrak{A}_1} = 0 & false_{\mathfrak{A}_1} = 1 \\ \mathfrak{A}_2 & \{0,1\} & true_{\mathfrak{A}_2} = 0 & false_{\mathfrak{A}_2} = 0 \\ \mathfrak{A}_3 & \mathbb{N} & true_{\mathfrak{A}_3} = 4 & false_{\mathfrak{A}_3} = 5 \\ \mathfrak{A}_4 & \{true, false\} & true_{\mathfrak{A}_4} = true & false_{\mathfrak{A}_4} = false \end{array}\right\} \textit{bool-Alg.}$$

b) sig $\equiv$ NAT, 0, suc

$$\left\{\begin{array}{lllll} A_{i_{\mathrm{NAT}}} & \mathbb{N} & \mathbb{Z} & \mathbb{N} & \{true, false\} & \{0, \mathrm{suc}^i(0)\} \\ 0_{\mathfrak{A}_i} & 0 & 0 & 1 & true & 0 \\ \mathrm{suc}_{\mathfrak{A}_i} & \mathrm{suc}_{\mathbb{N}} & \mathrm{pred}_{\mathbb{Z}} & \mathrm{id}_{\mathbb{N}} & \mathrm{suc}(true) = false & \mathrm{suc}(0) = suc(0) \\ & & & & \mathrm{suc}(false) = true & \mathrm{suc}(\mathrm{suc}^i(0)) = \mathrm{suc}^{i+1}(0) \end{array}\right.$$

Die von V erzeugte freie sig-Algebra

Definition 6.4

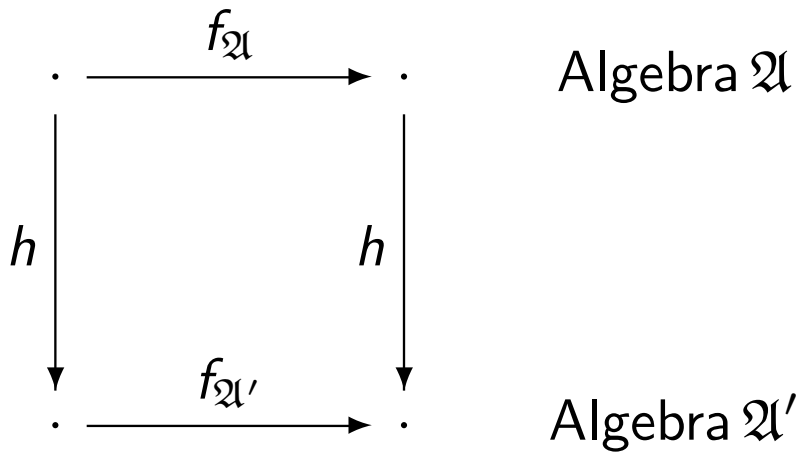
- $\mathfrak{A} = (A, F_{\mathfrak{A}})$ *mit:* $A = \bigcup_{s \in S} A_s$
  $A_s = \mathrm{Term}_s(F, V)$, *d. h.* $A = \mathrm{Term}(F, V)$
  $F \ni f : s_1, \ldots, s_n \to s$, $f_{\mathfrak{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
  $\mathfrak{A}$ *ist sig-Algebra::* $T_{\mathrm{sig}}(V)$ *die von* V *erzeugte freie Termalgebra mit Variablenmenge* V.
- $V = \varnothing$: $A_s = \mathrm{Term}_s(F)$ *Menge der Grundterme* ($A_s \neq \varnothing$, *da sig strikt*).
  $\mathfrak{A}$ *Grundtermalgebra::* T_{sig}

Homomorphismen

Definition 6.5 *sig-Homomorphismus:* $\mathfrak{A}, \mathfrak{A}'$ *sig-Algebren*
$h : \mathfrak{A} \to \mathfrak{A}'$ *Familie von Abbildungen.*
$h = \{h_s : A_s \to A'_s : s \in S\}$ *ist sig-Homomorphismus.*
Wenn

$$h_s(f_{\mathfrak{A}}(a_1, \ldots, a_n)) = f_{\mathfrak{A}'}(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$$

Wie üblich: injektiv, surjektiv, bijektiv, Isomorphismus



Kanonische Homomorphismen

Lemma 6.1 $\mathfrak{A}$ *sig-Algebra,* T_{sig}

a) Die Familie der Interpretationsfunktionen $h_s : \text{Term}_s(F) \to A_s$ *ist definiert durch*

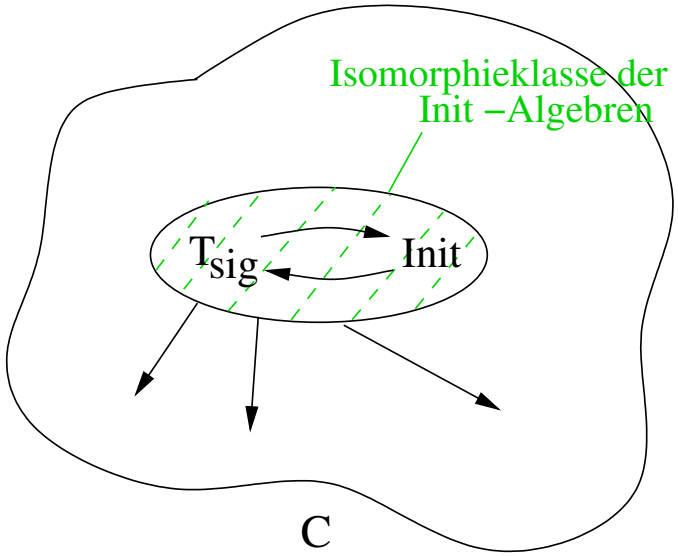
$$h_s(f(t_1, \ldots, t_n)) = f_{\mathfrak{A}}(h_{s_1}(t_1), \ldots, h_{s_n}(t_n))$$

mit $h_s(c) = c_{\mathfrak{A}}$ *ist ein sig-Homomorphismus.*

b) Es gibt keinen anderen sig-Homomorphismus von T_{sig} *nach* $\mathfrak{A}$*. Eindeutigkeit!*

Initiale Algebren

Definition 6.6 *Initiale Algebren:*
Eine sig-Algebra $\mathfrak{A}$ heißt Initial in einer Klasse C von sig-Algebren, wenn für jede sig-Algebra $\mathfrak{A}' \in C$ genau ein sig-Homomorphismus $h : \mathfrak{A} \to \mathfrak{A}'$ existiert.
Insbesondere: T_{sig} ist initial in der Klasse aller sig-Algebren.
Fakt: Initiale Algebren sind isomorph.



Analog lassen sich Finale Algebren definieren.

Kanonische Homomorphismen

$\mathfrak{A}$ sig-Algebra, $h : T_{\text{sig}} \to \mathfrak{A}$ Interpretationshomomorphismus.
$\mathfrak{A}$ sig-erzeugt (Term-erzeugt) gdw
$\forall s \in S \quad h_s : \text{Term}_s(F) \to A_s$ surjektiv
Die (freie) Termalgebra ist sig-erzeugt.
ADT Forderungen:

- Repräsentationsunabhängig (Isomorphieklasse)
- Operationserzeugt (sig-erzeugt)

These: Ein ADT ist die Isomorphieklasse einer initialen Algebra.
Termalgebren als initiale Algebren sind ADT.
Beachte Eigenschaften der freien Termalgebra: Fortsetzung von Abbildungen von V in $\mathfrak{A}$ lassen sich eindeutig zu Homomorphismen von $T_{sig}(V)$ in$\mathfrak{A}$ fortsetzen.

Kanonische Homomorphismen

Für Spezifikationsformalismen:
Klassen von Algebren die initialen Algebren zulassen.

$\rightsquigarrow$ Horn (Siehe Literatur)

sig INT sorts int
ops $0 :\longrightarrow$ int
$\quad$ suc : int $\longrightarrow$ int
$\quad$ pred : int $\longrightarrow$ int

Gleichungsspezifikation

Definition 6.7 $\mathrm{sig} = (S, F, \tau)$ *Signatur,* V *Variablensystem*

a) *Gleichung:* $(u, v) \in \mathrm{Term}_s(F, V) \times \mathrm{Term}_s(F, V)$
Schreibe: $u = v$
Gleichungssystem E*: Menge von Gleichungen*

b) *(Gleichungs)-Spezifikation:* $\mathrm{spec} = (\mathrm{sig}, E)$
E *Gleichungssystem über* $F \cup V$*.*

Notation

Schlüsselwort eqns

spec INT		
sorts int		implizite
ops	$0 :\longrightarrow$ int	All-Quantifikation
	suc, pred: int $\longrightarrow$ int	oft Deklaration
eqns	$\mathrm{suc}(\mathrm{pred}(x)) = x$	der Sorten
	$\mathrm{pred}(\mathrm{suc}(x)) = x$	der Variablen

Semantik::

- loose alle Modelle (PL1)
- enge (spezielles Modell initial, final)
- operational (Gleichungskalkül+Induktionsprinzip)

Modelle von spec = (sig, E)

Definition 6.8 $\mathfrak{A}$ *sig-Algebra,* $V(S)$*-Variablensystem*

a) *Belegungsfunktion* φ *für* $\mathfrak{A}$*:* $\varphi_s : V_s \to A_s$
Bewertung $\varphi : \mathrm{Term}(F, V) \to \mathfrak{A}$
$\varphi(f) = f_{\mathfrak{A}}$, f *konstant,* $\varphi(x) := \varphi_s(x)$, $x \in V_s$
$\varphi(f(t_1, \ldots, t_n)) = f_{\mathfrak{A}}(\varphi(t_1), \ldots, \varphi(t_n))$

$$\begin{array}{rcll} V_s & \xrightarrow{\varphi_s} & A_s & \\ \mathrm{Term}_s(F, V) & \xrightarrow{\varphi_s} & A_s & \\ \mathrm{Term}(F, V) & \xrightarrow{\varphi} & \mathfrak{A} & \textit{Homomorphismus} \end{array}$$

Modelle von spec $=(\text{sig}, E)$

b) $s = t$ Gleichung über sig, V
$\mathfrak{A} \models_{\varphi} s = t$: $\mathfrak{A}$ erfüllt $s = t$ mit Belegung φ gdw $\varphi(s) = \varphi(t)$,
Gleichheit in A.

c) $\mathfrak{A}$ erfüllt $s = t$ bzw. $s = t$ gilt in $\mathfrak{A}$
$\mathfrak{A} \models_{\varphi} s = t$: Für jede Belegung φ
$\mathfrak{A} \models_{\varphi} s = t$

d) $\mathfrak{A}$ ist Modell von spec $=(\text{sig}, E)$
gdw $\mathfrak{A}$ erfüllt jede Gleichung von E
$\mathfrak{A} \models E$ ALG(spec).

Beispiele

Beispiel 6.4 *1)*

spec	NAT
sorts	nat
ops	$0 :\longrightarrow \text{nat}$
	$s : \text{nat} \longrightarrow \text{nat}$
	$\_ + \_ : \text{nat}, \text{nat} \longrightarrow \text{nat}$
eqns	$x + 0 = x$
	$x + s(y) = s(x + y)$

Beispiele

sig-Algebren

a) $\mathfrak{A} = (\mathbb{N}, \hat{0}, \hat{+}, \hat{s})$
$\hat{0} = 0 \quad \hat{s}(n) = n + 1 \quad n \hat{+} m = n + m$

b) $\mathfrak{L} = (\mathbb{Z}, \hat{0}, \hat{+}, \hat{s})$
$\hat{0} = 1 \quad \hat{s}(i) = i \cdot 5 \quad i \hat{+} j = i \cdot j$

c) $\mathfrak{C} = (\{\mathsf{true}, \mathsf{false}\}, \hat{0}, \hat{+}, \hat{s})$
$\hat{0} = \mathsf{false} \quad \hat{s}(\mathsf{true}) = \mathsf{false} \quad \hat{s}(\mathsf{false}) = \mathsf{true}$
$i \hat{+} j = i \cup j$

Beispiele

$\mathfrak{A}, \mathfrak{L}, \mathfrak{C}$ sind Modelle von spec NAT

z.B. $\mathfrak{L}$: $\varphi(x) = a \quad \varphi(y) = b \quad a, b \in \mathbb{Z}$

$$\varphi(x + 0) = a \hat{+} \hat{0} = a \cdot 1 = a = \varphi(x)$$

$$\begin{aligned} \varphi(x + s(y)) &= a \hat{+} \hat{s}(b) = a \cdot (b \cdot 5) \\ &= (a \cdot b) \cdot 5 = \hat{s}(a \hat{+} b) \\ &= \varphi(s(x + y)) \end{aligned}$$

Beispiele

2)

spec	LIST(NAT)
use	NAT
sorts	nat, list
ops	$\mathsf{nil} :\longrightarrow \mathsf{list}$
	$\_.\_ : \mathsf{nat}, \mathsf{list} \longrightarrow \mathsf{list}$
	$\mathsf{app} : \mathsf{list}, \mathsf{list} \longrightarrow \mathsf{list}$
eqns	$\mathsf{app}(\mathsf{nil}, q_2) = q_2$
	$\mathsf{app}(x.q_1, q_2) = x.\,\mathsf{app}(q_1, q_2)$

Beispiele

spec-Algebra

$$\begin{array}{ll} \mathfrak{A} & \mathbb{N}, \mathbb{N}^* \\ \hat{0} = 0 \quad \hat{+} = + & \hat{s} = +1 \\ \hat{\mathsf{nil}} = e & (\mathsf{leeresWort}) \\ \hat{:}\ (i, z) = i\ z & \\ \widehat{\mathsf{app}}(z_1, z_2) = z_1 z_2 & (\mathsf{Konkatuation}) \end{array}$$

Beispiele

3) spec INT $\quad \mathrm{suc}(\mathrm{pred}(x)) = x \quad \mathrm{pred}(\mathrm{suc}(x)) = x$

	1	2	3
A_{int}	$\mathbb{Z}$	$\mathbb{N}$	$\{\mathrm{true}, \mathrm{false}\}$
$0_{\mathfrak{A}_i}$	0	0	true
$\mathrm{suc}_{\mathfrak{A}_i}$	$\mathrm{suc}_{\mathbb{Z}}$	$\mathrm{suc}_{\mathbb{N}}$	$\left\{ \begin{array}{c} \mathrm{true} \longrightarrow \mathrm{false} \\ \mathrm{false} \longrightarrow \mathrm{true} \end{array} \right\}$
$\mathrm{pred}_{\mathfrak{A}_i}$	$\mathrm{pred}_{\mathbb{Z}}$	$\left\{ \begin{array}{c} n+1 \longrightarrow n \\ 0 \longrightarrow 0 \end{array} \right\}$	$\left\{ \begin{array}{c} \mathrm{true} \longrightarrow \mathrm{false} \\ \mathrm{false} \longrightarrow \mathrm{true} \end{array} \right\}$
	$+$	$-$	$+$

Beispiele

	4	5
A_{int}	$\{a, b\}^* \cup \mathbb{Z}$	$\{1\}^+ \cup \{0\}^+ \cup \{z\}$
$0_{\mathfrak{A}_i}$	0	z
$\text{suc}_{\mathfrak{A}_i}$	$\text{suc}_{\mathbb{Z}}$	$\left\{ \begin{array}{c} 1^n \longrightarrow 1^{n+1} \\ z \longrightarrow 1 \\ 0^{n+1} \longrightarrow 0^n \\ 0 \longrightarrow z \end{array} \right\}$
$\text{pred}_{\mathfrak{A}_i}$	$\text{pred}_{\mathbb{Z}}$	$\left\{ \begin{array}{c} 1^{n+1} \longrightarrow 1^n \\ 1 \longrightarrow z \\ z \longrightarrow 0 \\ 0^n \longrightarrow 0^{n+1} \end{array} \right\}$
	$-$	$+$

Substitution

Definition 6.9 *sig*, $Term(F, V)$
σ: $\sigma_s : V_s \to \mathrm{Term}_s(F, V)$, $\sigma(x) \in \mathrm{Term}_s(F, V)$, $x \in V_s$
$\sigma(x) = x$ *für fast alle* $x \in V$
$D(\sigma) = \{x \mid \sigma(x) \neq x\}$ *endlich Definitionsbereich*
Schreibe $\sigma = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$
Fortsetzung zu Homomorphismus $\sigma : \mathrm{Term}(F, V) \to \mathrm{Term}(F, V)$

$$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$$

Grundsubstitution: $t_i \in \mathrm{Term}_S(F) \quad x_i \in D(\sigma)_S$

Lose Semantik

Definition 6.10 spec $= (\text{sig}, E)$
ALG(spec)$= \{\mathfrak{A} \mid$ *sig-Algebra,* $\mathfrak{A} \models E\}$
Charakterisierungen der Gleichungen die in ALG(spec) bzw. $\text{ALG}_{\text{TE}}(\text{spec})$ *(Term erzeugte).*

a) *Semantische Gleichheit:* $E \models s = t$

b) *Operationale Gleichheit:* $t_1 \underset{E}{\vdash\!\dashv} t_2$ *gdw*

Es gibt $p \in 0(t_1), s = t \in E$, *Substitution* σ *mit*
$t_1|_p \equiv \sigma(s),\ t_2 \equiv t_1[\sigma(t)]_p(t_1[p \leftarrow \sigma(t)])$
oder $t_1|_p \equiv \sigma(t),\ t_2 \equiv t_1[\sigma(s)]_p$

$t_1 =_E t_2$ gdw $t_1 \overset{*}{\underset{E}{\vdash\!\dashv}} t_2$ *Gleiches* $\leftrightarrow$ *Gleich*

Gleicheitskalkül

c) Gleichheitskalkül: Inferenzregeln (deduktiv)

Reflexivität $\dfrac{}{t = t}$

Symmetrie $\dfrac{t = t'}{t' = t}$

Transitivität $\dfrac{t = t', t' = t''}{t = t''}$

Ersetzung $\dfrac{t' = t''}{s[t']_p = s[t'']_p} \qquad p \in 0(s)$

(oft auch mit Substitution σ)

Gleicheitskalkül

$E \vdash s = t$ gdw es gibt einen Beweis für $s = t$ aus E, d. h.
$P =$ Folge von Gleichungen die mit $s = t$ endet, wobei für $t_1 = t_2 \in P$ gilt.

i) $t_1 = t_2 \in \sigma(E)$ für ein σ: Substitution

ii) $t_1 = t_2 \ldots$ aus vorangehenden Gleichungen durch Anwendung einer der Inferenzregeln.

Eigenschaften und Beispiele

a) Gilt $E \models s = t$ oder $s =_E t$ oder $E \vdash s = t$

i) Ist σ eine Substitution, so auch
$E \models \sigma(s) = \sigma(t)$ / $\sigma(s) =_E \sigma(t)$ / $E \models \sigma(s) = \sigma(t)$
d. h. die induzierten Äquivalenzrelationen auf $\mathrm{Term}(F, V)$ sind stabil bzgl. Substitutionen

ii) $r \in \mathrm{Term}(F, V)$, $p \in 0(r)$, $r|_p$, $s, t \in \mathrm{Term}_{s'}(F, V)$ so
$E \models r[s]_p = r[t]_p$ / $r[s]_p =_E r[t]_p$ / $E \vdash r[s]_p = r[t]_p$
Monotonie Eigenschaft

Kongruenz auf $\mathrm{Term}(F, V)$ die stabil ist.

Kongruenzen / Quotientenalgebren

b) $\mathfrak{A} = (A, f_{\mathfrak{A}})$ sig-Algebra. $\sim$ bin. Relation auf A ist Kongruenzrelation auf $\mathfrak{A}$, falls

i) $a \sim b \rightsquigarrow \exists s \in S : a, b \in A_s$ (Sortentreu)
ii) $\sim$ ist Äquivalenzrelation
iii) $a_i \sim b_i$ $(i = 1, \ldots, n)$, $f_{\mathfrak{A}}(a_1, \ldots, a_n)$ definiert
$\rightsquigarrow f_{\mathfrak{A}}(a_1, \ldots, a_n) \sim f_{\mathfrak{A}}(b_1, \ldots, b_n)$ (monoton)

$\mathfrak{A}/\sim$ Quotientenalgebra:
$A/\sim = \bigcup_{s \in S}(A_s/\sim)_s$ mit $(A_s/\sim)_s = \{[a]_\sim : a \in A_s\}$ und $f_{\mathfrak{A}/\sim}$ mit $f_{\mathfrak{A}/\sim}([a_1], \ldots, [a_n]) = [f_{\mathfrak{A}}(a_1, \ldots, a_n)]$

wohldefiniert, d. h. $\mathfrak{A}/\sim$ ist sig-Algebra.

Zusammenhänge zwischen $\models, =_E, \vdash_E$

$\varphi : \mathfrak{A} \to \mathfrak{A}_\sim$ mit $\varphi_s(a) = [a]_\sim$ ist surjektiver Homomorphismus, kanonischer Homomorphismus.

c) $\mathfrak{A}, \mathfrak{A}'$ sig-Algebren $\varphi : \mathfrak{A} \to \mathfrak{A}'$ surjektiver Homomorphismus.
Dann $\mathfrak{A} \models s = t \rightsquigarrow \mathfrak{A}' \models s = t$

d) spec $= (\text{sig}, E)$:

$$s =_E t \text{ gdw } E \vdash s = t$$

e) $\mathfrak{A}$ sig-Algebra, R eine sortentreue bin. Relation auf $\mathfrak{A}$. Dann gibt es eine kleinste Kongruenz $\equiv_R$ auf $\mathfrak{A}$ die R enthält, d. h. $R \subseteq \equiv_R$
$\equiv_R$ die von R erzeugte Kongruenz

Zusammenhänge zwischen $\models, =_E, \vdash_E$

f) $\mathfrak{A}$ sig-Algebra, E Gleichungssystem über (sig, V). E induziert eine Relation $\underset{E,\mathfrak{A}}{\sim}$ auf $\mathfrak{A}$ wobei

$a \underset{E,\mathfrak{A},s}{\sim} a' \quad (a, a' \in A_s)$ gdw es gibt $t = t' \in E$ und Belegung $\varphi : V \to \mathfrak{A}$ mit $\varphi(t) = a$, $\varphi(t') = a'$

Diese Relation ist sortentreu.

Fakt: Sei $\equiv$ eine Kongruenz auf $\mathfrak{A}$ die $\underset{E,\mathfrak{A}}{\sim}$ enthält, dann ist $\mathfrak{A}/\equiv$ eine spec $= (\text{sig}, E)$-Algebra, d. h. Modell von E.

g) Existenz: $\mathfrak{A} = T_{\text{sig}}$ die Termalgebra, dann ist $=_E$ auf T_{sig} die kleinste Kongruenz die $\underset{E,\mathfrak{A}}{\sim}$ umfasst.

Insbesondere ist $T_{\text{sig}}/=_E$ ein Modell von E.

Beispiel

spec :: INT mit $\text{pred}(\text{suc}(x)) = x$, $\text{suc}(\text{pred}(x)) = x$

$$
\begin{aligned}
(T_{\text{INT}}/=_E)_{\text{int}} = \quad & \{[0] = \{0, \text{pred}(\text{suc}(0)), \text{suc}(\text{pred}(0)), \ldots \\
& [\text{suc}(0)] = \{\text{suc}(0), \text{pred}(\text{suc}(\text{suc}(0))), \ldots \\
& [\text{suc}(\text{suc}(0))] = \{\cdots \\
& [\text{pred}(0)] = \{\text{pred}(0), \text{suc}(\text{pred}(\text{pred}(0))) \ldots
\end{aligned}
$$

$$
\begin{aligned}
\text{suc}_{T_{\text{INT}}/=_E} \quad & ([\text{pred}(\text{suc}(0))]) = [\text{suc}(\text{pred}(\text{suc}(0)))] \\
& = [\text{suc}] \\
& = \text{suc}_{T_{\text{INT}}/=_E}([0])
\end{aligned}
$$

Satz von Birkhoff

Satz 6.1 *Für jede Spezifikation* $spec = (sig, E)$ *gilt*

$$E \models s = t \text{ gdw } E \vdash s = t \quad (\text{d. h. } s =_E t)$$

Definition

Initiale Semantik

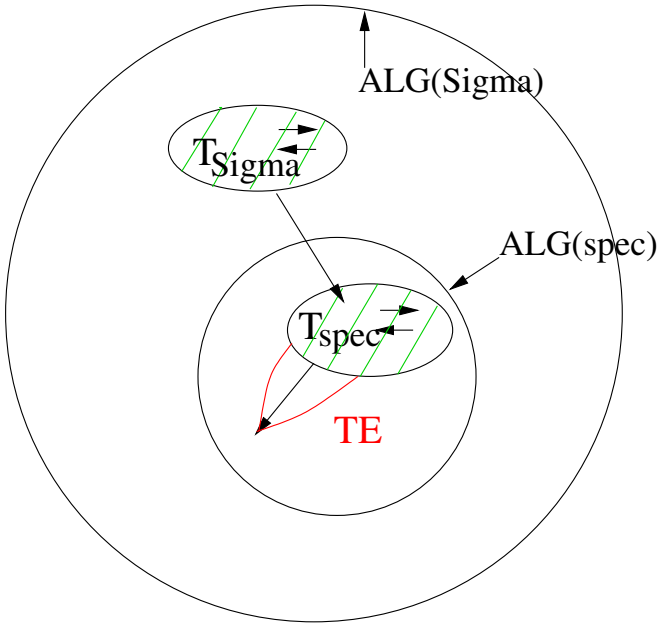
Sei spec = (sig, E) die Algebra $T_{sig}/ =_E$ ($=_E$ die von E erzeugte kleinste Kongruenzrelation auf T_{sig})

Quotiententermalgebra

Sie ist operationserzeugt und initial in ALG(spec)!

Initiale Algebra Semantik

Initiale Algebra Semantik ordnet jeder Gleichungsspezifikation spec die Isomorphieklasse der (initialen) Quotiententermalgebra $T_{sig}/=_E$ zu.
Schreibe: T_{spec} oder $I(E)$
Sigma=Σ



$$sig = \Sigma,\ spec = (\Sigma, E)$$

Quotiententermalgebren

Quotiententermalgebren sind ADT.

Beispiel 6.5 *(Fortsetzung) INT*

A^i_{int}	$\mathbb{Z}$	$\{true, false\}$	$\{1\}^+ \cup \{0\}^+ \cup \{z\}$
0_{A^i}	0	*true*	*z*
suc_{A^i}	$\text{suc}_{\mathbb{Z}}$	not	...
pred_{A^i}	$\text{pred}_{\mathbb{Z}}$	not	...

$T_{\text{INT}}/=_E$ $[0] \mapsto true$ $[\text{suc}^{2n}(0)] \mapsto true$
$[\text{suc}^{2n+1}(0)] \mapsto false$ $[\text{pred}^{2n+1}(0)] \mapsto false$
$[\text{pred}^{2n}(0)] \mapsto true$

Initiale Algebra

spec $= (\text{sig}, E)$ Initiale Algebra $T_{\text{spec}}(I(E))$
Probleme:

- Ist T_{spec} berechenbar?
- Wortproblem $(T_{\text{sig}}, =_E)$ lösbar?
- Operationalisierung von T_{spec}?
- Welche Eigenschaften (PL1) gelten?

Gleichheitstheorie/Induktive Theorie

a) $TH(E) = \{s = t : E \models s = t\}$ Gleichheitstheorie
Gleichungen, die in allen spec-Algebren gelten.

b) $ITH(E) = \{s = t : T_{\text{spec}} \models s = t\}$ induktive $(=)$-Theorie
Gleichungen, die in allen termerzeugten spec-Algebren gelten.

Gleichheitstheorie/Induktive Theorie

Bemerkung:

a) $TH(E) \subseteq ITH(E)$, da T_{spec} Modell von E.

b) i. Allg. $TH(E) \subsetneq ITH(E)$
$=$ so E *w*-vollständig
E r.a., so $TH(E)$ r.a., aber $ITH(E)$ i. Allg. nicht r.a.

c) $T_{\text{spec}} \models s = t$ gdw $\sigma(s) =_E \sigma(t)$ für jede Grundsubstitution der Var. in s, t.

d) $E : x + 0 = x \qquad x + s(y) = s(x + y)$
$\rightsquigarrow x + y = y + x \in ITH(E) - TH(E)$
$(x + y) + z = x + (y + z)$

Beispiele

Beispiel 6.6 *a)*

spec BOOL
sorts bool
ops *true*, *false* :⟶ bool
not : bool ⟶ bool
and, *or*, impl, eqv : bool, bool ⟶ bool
if _ then _ else _ : bool, bool, bool ⟶ bool

Beispiel (Forts.)

$$
\begin{array}{ll}
\underline{\text{eqns}} & \text{not}(\text{true}) = \text{false} \\
& \text{not}(\text{false}) = \text{true} \\
& \text{and}(\text{true}, b) = b \\
& \text{and}(\text{false}, b) = \text{false} \\
& \text{or}(b, b') = \text{not}(\text{and}(\text{not}(b), \text{not}(b'))) \\
& \text{impl}(b, b') = \text{or}(\text{not}(b), b') \\
& \text{eqv}(b, b') = \text{and}(\text{impl}(b, b'), \text{impl}(b, b')) \\
& \text{if true } b' \text{ else } b'' = b' \\
& \text{if false } b' \text{ else } b'' = b''
\end{array}
$$

$(T_{\text{BOOL}})_{\text{bool}} = \{[\text{true}], [\text{false}]\}$ (Beweis!)

Beispiel (Forts.)

b)

spec	SET-OF-CHARACTERS
sorts	char, set
ops	$a, b, c, \cdots :\longrightarrow$ char
	$\varnothing :\longrightarrow$ set
	insert : char, set $\longrightarrow$ set
eqns	$\text{insert}(x, \text{insert}(x, s)) = \text{insert}(x, s)$
	$\text{insert}(x, \text{insert}(y, s)) = \text{insert}(y, \text{insert}(x, s))$

$(T_{\text{soc}})_{\text{char}} = \{a, b, c, \ldots\}$

$(T_{\text{soc}})_{\text{set}} = \{[\varnothing], [\text{insert}(a, \varnothing), \ldots$

$\{\varnothing\}\{\text{insert}(a, \varnothing)\}$

Beispiel (Forts.)

c)

spec NAT

sorts nat

ops $0 :\longrightarrow \text{nat}$

$\text{suc} : \text{nat} \longrightarrow \text{nat}$

$\_ + \_, \_ * \_ : \text{nat}, \text{nat} \longrightarrow \text{nat}$

eqns $x + 0 = x$

$x + \text{suc}\, y = \text{suc}(x + y)$

$x * 0 = 0$

$x * \text{suc}(y) = (x * y) + x$

$(T_{\text{NAT}})_{\text{nat}} = \{ [0, 0 + 0, 0 * 0, \ldots$

$[\text{suc}\, 0, 0 + \text{suc}\, 0, \ldots$

$[\text{suc}(\text{suc}(0)), \ldots$

Beispiel (Forts.)

d) Binärbäume

spec BINTREE

sorts nat, tree

ops $0 :\longrightarrow \text{nat}$

$\text{suc} : \text{nat} \longrightarrow \text{nat}$

$\text{max} : \text{nat}, \text{nat} \longrightarrow \text{nat}$

$\text{leaf} :\longrightarrow \text{tree}$

$\text{right} : \text{tree} \longrightarrow \text{tree}$

$\text{both} : \text{tree}, \text{tree} \longrightarrow \text{tree}$

$\text{height} : \text{tree} \longrightarrow \text{nat}$

$\text{dleft} : \text{tree} \longrightarrow \text{tree}$

$\text{dright} : \text{tree} \longrightarrow \text{tree}$

Beispiel (Forts.)

zu d) Binärbäume

eqns
$$\max(0, n) = n$$
$$\max(n, 0) = n$$
$$\max(\mathrm{suc}(m), \mathrm{suc}(n)) = \mathrm{suc}(\max(m, n))$$
$$\mathrm{height}(\mathrm{leaf}) = 0$$
$$\mathrm{height}(\mathrm{both}(t, t')) = \mathrm{suc}(\max(\mathrm{height}(t), \mathrm{height}(t')))$$
$$\mathrm{height}(\mathrm{left}(t)) = \mathrm{suc}(\mathrm{height}(t))$$
$$\mathrm{height}(\mathrm{right}(t)) = \mathrm{suc}(\mathrm{height}(t))$$