2.5 Rechenstrukturen

10.12.02 (1)

Rückblick: Die bisher betrachteten Signaturen und Terme sind rein syntaktische Strukturen, deren Semantik noch undefiniert ist.

Rechenstruktur: Gegeben sei eine Signatur $\Sigma = (S, F)$. Eine Rechenstruktur A ordnet den Sorten Trägermengen $S_A = \{ s_A : s \in S \}$ und den Funktionssymbolen Funktionen $F_A = \{ f^A : f \in F \}$ zu.

Hierbei sind f^A

$$f^A : S_A^A \times S_2^A \times \ldots \times S_n^A$$

n-stellige Funktionen.

Allgemeines zu Trägermengen: Es gibt Funktionen, die für gewisse Argumente kein definiertes Resultat liefern (DSP: Maximum einer leeren Liste); solche Funktionen nennt man partiell. Um partielle Funktionen zu vermeiden, verwendet man das Symbol $\perp$ (bottom) und fügt dieses Symbol der Trägermenge hinzu: Sei M eine Trägermenge

$$M^\perp := M \cup \{\perp\}$$

Eine Abbildung

$$f_A : M_1^\perp \times \ldots \times M_n^\perp$$

② heißt strikt, wenn sie das Resultat $\perp$ liefert, sobald eines der Argumente $\perp$ ist.

Beispiel: Die Verzweigung „if ... then ... else ...“ ist in der Regel nicht strikt

Beispiele für Rechenstrukturen:

① Die Rechenstruktur der Wahrheitswerte BOOL:

Signatur $S = \{bool\}$

$F = \{true, false, \neg, \wedge, \vee\}$

Zuordnung der Sorten zu Trägermengen

$bool^{BOOL} : \to \mathbb{B}^\perp$

Zuordnung der Funktionssymbole zu Funktionen:

$true^{BOOL} : \to \mathbb{B}^\perp$

$false^{BOOL} : \to \mathbb{B}^\perp$

$\neg^{BOOL} : \mathbb{B}^\perp \to \mathbb{B}^\perp$

$\wedge^{BOOL} : \ldots$

$\vee^{BOOL} : \ldots$

hierbei ist die Semantik der Funktion folgendermaßen gegeben: Sei $a, b \in \mathbb{B}^\perp$

$true^{BOOL} = L$

$false^{BOOL} = 0$

$\neg^{BOOL} b = not(b)$

$a \wedge^{BOOL} b = and(a, b)$

$a \vee^{BOOL} b = or(a, b)$

b	0	L
$not(b)$	L	0

} definiert durch Wertetabellen

(2) Die Rechenstruktur SET der Mengen über der Signatur (3)

$S = \{bool\}$; $F = \{true, false, \neg, \wedge, \vee\}$

Trägermenge:

$bool^{SET} = \mathcal{P}(U)$ (Potenzmenge einer Menge U)

Funktionalität:

$true^{SET} : \rightarrow \mathcal{P}(U)$

$false^{SET} : \rightarrow \mathcal{P}(U)$

$\neg^{SET} a : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$

⋮

Semantik der Funktionen: Seien $U', U'' \in \mathcal{P}(U)$

$true^{SET} = U$

$false^{SET} = \emptyset$

$\neg^{SET} U' = U \setminus U'$;

$U' \wedge^{SET} U'' = \{m \in U : and((m \in U'), (m \in U''))\}$

$U' \vee^{SET} U'' = \{$ " : or (" , " $)\}$

Bemerkung: Die Rechenstruktur SET und die Rechenstruktur BOOL genügen denselben syntaktischen Gesetzen. (z. B.: $x \wedge (y \wedge z) = (x \wedge y) \wedge z$) AL) (etwa Broy (I) S. 17)

Abstrakte Algebra: Eine Signatur zusammen mit den für die Terme der Signatur geltenden Gesetze nennt man abstrakte Algebra. (4)
(weitere Beispiele abstr. Algebren: Gruppen, Vektorräume)

(3) Die Rechenstruktur NAT der natürlichen Zahlen

Signatur: $S = \{bool, nat\}$

$F = \{true, false, \ldots, zero, succ, pred, add, mult, sub, div, \leq, =\}$

Trägermengen:

$bool^{NAT} = \mathbb{B}^\perp$, $nat^{NAT} = \mathbb{N}^\perp$

Funktionalitäten:

/* boolesch → siehe oben */

$zero^{NAT} : \to \mathbb{N}^\perp$

$succ^{NAT} : \mathbb{N}^\perp \to \mathbb{N}^\perp$

$pred^{NAT} :$ "

$add^{NAT} : \mathbb{N}^\perp \times \mathbb{N}^\perp \to \mathbb{N}^\perp$

⋮

Semantik der Funktionen:

$zero^{NAT} = 0$

$succ^{NAT}\ x = x + 1$

$pred^{NAT}\ x = x - 1$

$pred^{NAT}\ 0 = \perp$

$add^{NAT}(x,y) = x + y$ (5)

⋮

Weitere Rechenstrukturen: Broy (I) S. 49 ff.

2.6 Termersetzungssysteme:

Beispiel: Betrachte den Term

(3) $(\neg(x \vee false)) \vee (y \wedge false)$ über den Rechenstrukturen BOOL und SET. Da SET und BOOL Exemplare derselben abstrakten Algebra sind, genügen sie denselben Gesetzen:

$$x \vee false = x \quad (1)$$

$$y \wedge false = false \quad (2)$$

(1) + (2) in (3)

$$\Rightarrow (\neg x \vee false) \overset{(1)}{\Rightarrow} \neg x$$

⇒ Terme unterschiedlicher Rechenstr. lassen durch rein syntaktische Umwandlungen ohne Beachtung der Semantik vereinfachen.

Termersetzungsregel:

Sei Σ eine Signatur, X sei eine Menge von Identifikatoren. Seien $t, r \in W_\Sigma$ (mit Identif. aus X)

Dann ist

$$t \to r$$

eine Termersetzungsregel. Dann ⑥
heißt

$$t'[t/x] \to r[t/x]$$

eine Instanz der Regel

Beispiele

$succ(pred(x)) \to x; \quad x \neq zero.$

$pred(succ(x)) \to x$

$add(x, zero) \to x$

Termersetzungssystem: Eine endliche Menge von Termer-Regeln über einer Signatur Σ heißt Termersetzungssystem.

Termersetzungsalgorithmus: Gegeben sei ein Termersetzungssystem über Σ

- Läßt sich eine Regel $r \in R$ auf einen Term t anwenden, so dass $t \overset{*}{\Rightarrow} t'$, dann setze man den Alg. mit t' fort. (nichtdeterministische Termersetzung)
- Kann keine Regel angewendet werden, so endet der Alg.; man erhält terminale Terme.

(term. Terme: $succ(succ, \ldots(zero)\ldots)$
$pred(zero)$

Bemerkung: Termersetzungssysteme sind abstrakte Algebren, deren Gesetze nur Äquivalenzrelationen zwischen Termen enthalten

Beispiele für Termersetzungssysteme (7)

① Die bekannten Gesetze der (abstrakten) boolesschen Algebra induzieren ein Termersetzungssystem.

② Termersetzungssystem über der Signatur der Rechenstruktur NAT.

Für die verschiedenen Funktionen von NAT lassen sich folgende Regeln definieren:

pred: pred(succ(x)) → x

add: add(succ(x), y) → succ(add(x, y))
add(zero, y) → y

mult:
mult(succ(x), y) → add(mult(x, y), y)
mult(x, succ(y)) → add(mult(x, y), x)
mult(x, zero) → zero

Transformation in ein funktionales Programm:

```
type nat = Zero | Succ of nat;;

let pred n = match n with
  | Zero -> fail with ("undefined")
  | Succ x -> x ;;
```

```
Let rec add (n, m) = match (n, m) with
    | (Zero, m) -> m
    | (Succ x, m) -> Succ (add (x, m));;
```