

# Algebraische Spezifikation von Software und Hardware VI

---

Markus Roggenbach

Mai 2008

# 7. Ausdruckstärke von Logiken

Gegeben:

einen Klasse  $C$  von Algebren, die unter Isomorphie abgeschlossen ist.

Frage:

gibt es eine Spezifikation  $Sp$  mit  $Mod(Sp) = C$ ?

## Gleichungslogik

wir schränken unsere Logik  $FOL^=$  ein auf

- Signaturen  $\Sigma = (S, TF, PF, P)$  mit  $PF = \emptyset$ ,  $P = \emptyset$
- Formeln: ausschliesslich  $\forall X.t_1 = t_2$   
 $X$  ist ein Variablensystem,  $t_i \in T_\Sigma(X)$  für  $i = 1, 2$

# Gleichungslogik (EL) hat immer ein Modell

Sei  $Sp$  eine Spezifikation in EL mit Signatur  $\Sigma = (S, TF)$ .

Sei  $M(s) = \{*\}$  für alle  $s \in S$ ;

Sei  $M(f)(*, \dots, *) = *$  für alle  $f \in TF$ .

Dann gilt:  $M \in Mod(Sp)$ .

## Beweis

Sei  $\nu : X \rightarrow M$  eine Variablenbelegung.

Dann ist

$$\nu^\#(t_1) = * = \nu^\#(t_2)$$

# Konsequenz

Der Datentyp Boolean lässt sich nicht in EL mit looser Semantik spezifizieren.

## Def: Theorie einer Klasse von Algebren

Sei  $C$  eine Klasse von  $\Sigma$ -Algebren. Dann bezeichnet

$$Th(C) := \{\varphi \in FOL^=(\Sigma) \mid C \models \varphi\}$$

die  $(FOL^=)$  **Theorie** von  $C$ .

# Axiomatisierbarkeit einer Klasse von Algebren

- Eine Klasse  $C$  von  $\Sigma$ -Algebren heisst axiomatisierbar (in  $\text{FOL}^=$ ), wenn ihre ( $\text{FOL}^=$ ) Theorie rekursiv aufzählbar ist.
- Eine  $\Sigma$ -Algebra  $A$  heisst axiomatisierbar (in  $\text{FOL}^=$ ), wenn die von ihr erzeugte Klasse

$$\{A\} := \{B \mid A \sim B\}$$

aufzählbar ist.

## Grenzen von FOL

```
spec PeanoArithmetik =  
  sort Nat  
  ops __+__, __*__: Nat * Nat -> Nat  
  pred __<= __: Nat * Nat  
end
```

Sei  $A$  die klassische Algebra für die Peano Arithmetik  
(d.h.  $A(\text{Nat}) = \mathbf{Nat}$  etc)

Gödels 2. Unvollständigkeitssatz impliziert:  
 $A$  ist in  $\text{FOL}^=$  nicht axiomatisierbar, d.h.  $\text{Th}(A)$  ist nicht rekursiv aufzählbar.

Für CASL Spezifikationen  $Sp$  in  $\text{FOL}^=$  gilt:  
 $\text{Mod}(Sp)$  ist axiomatisierbar (in  $\text{FOL}^=$ ).

# 8. Freie Datentypen

## A simple example

```
spec Hugo =  
  sort Erna  
  op p,q: Erna  
end
```

$$A(\textit{Erna}) = \{*\}, \quad A(p) = A(q) = *.$$

$$B(\textit{Erna}) = \{1, 2\}, \quad B(p) = 1, \quad B(q) = 2.$$

$$C(\textit{Erna}) = \{1, 2, 3\}, \quad C(p) = 1, \quad C(q) = 2.$$

# Confusion – informal

Confusion: The interpretation of different terms is the same.

## Junk – informal

Junk: The carrier set  $A(s)$  for a sort  $s$  contains ‘non-reachable’ values.

A value is ‘reachable’ if there is a term that denotes it.

## Basic setting

Junk and confusion both speak about terms.

Which symbols do we allow in these terms???

~→ constructors for a sort.

## Sort generation constraint

Let  $\Sigma = (S, TF, PF, P)$  be a signature. A **sort generation constraint** is a pair

$$(s, Con)$$

with

- $s \in S$ 
  - the sort  $s$  we are speaking about – and
- $Con \subseteq TF \cup PF$ 
  - the set of function symbols we allow in the terms. These function symbols are often called **constructors** – such that

$f \in Con$  has  $s$  as its target sort.

## Confusion/Junk – a bit more formal

An algebra  $A$  has **junk with respect to a sort generation constraint**  $(s, Con)$ , if its carrier set  $A(s)$  includes values that have no denotation by a term formed from the set of constructors  $Con$ .

An algebra  $A$  has **confusion with respect to a sort generation constraint**  $(s, Con)$ , if there are syntactically different terms formed from the set of constructors  $Con$  that denote the same value in  $A(s)$ .

## Avoiding confusion

sort generation constraint: ( $Erna, \{p : s, q : s\}$ )

```
spec Hugo =  
  sort Erna  
  op p,q: Erna  
  forall x: Erna  
    . not (p=q)  
end
```

# Avoiding Junk

sort generation constraint: ( $Erna, \{p : s, q : s\}$ )

```
spec Hugo =  
  sort Erna  
  op p,q: Erna  
  forall x: Erna  
    . x=p \ / x=q  
end
```