

Sort Generation Constraints

Introduction

Specification: “description by properties”

Main question on specifications:
“What happens if”

Specifications should be

- complete
- precise
- consistent (no contradictions)

Example 1 (What about $0 + x = x$?)

```
spec Nat =  
  sort Nat  
  ops 0: Nat;  
      suc: Nat -> Nat;  
      __+__: Nat * Nat -> Nat  
  forall x,y,z: Nat  
  . x + 0 = 0  
  . x + suc(y) = suc(x+y)  
  . (x+y)+z=x+(y+z)  
end
```

Background from Theory

Gödel's 2. Incompleteness Theorem:
Naturals are not axiomatizable in FOL

Sort Generation

Definition 1 (Generated Σ -Algebra)

$\Sigma = (S, \Omega)$ signature, $\Omega_c \subseteq \Omega$ set of constructors.

Let $\Sigma_c := (S, \Omega_c)$.

A Σ -algebra A is called

generated in Ω_c ,

if there exists at least one groundterm $t \in T_{\Sigma_c, s}$ with $a = A(t)$
for all $s \in S$ and for all $a \in A(s)$.

generated, if A is generated in Ω .

Definition 2 (Freely generated Σ -Algebra)

$\Sigma = (S, \Omega)$ signature, $\Omega_c \subseteq \Omega$ set of constructors.

Let $\Sigma_c := (S, \Omega_c)$.

A Σ -algebra A is called

freely generated in Ω_c ,

if there exists exactly one groundterm $t \in T_{\Sigma_c, s}$ with $a = A(t)$
for all $s \in S$ and for all $a \in A(s)$.

freely generated, if A is freely generated in Ω .

Definition 3 ((Freely) gen. Σ -Algebra (gen.))

$\Sigma = (S, \Omega)$ signature,

$S_c \subseteq S$ set of sorts,

$\Omega_c \subseteq \Omega$ set of constructors with target in S_c .

Let $\Sigma_c := (S, \Omega_c)$.

X set of variables with $X_s = \emptyset$ for $s \in S_c$.

Σ -Algebra A .

α assignment:

$\alpha_s : X_s \rightarrow A(s)$ bijective for $s \in S - S_c$

*The Σ -algebra A is called
generated in S_c by Ω_c , if:
for each sort $s \in S_c$:
for each element $a \in A(s)$:
there exists at least one term $t \in T_{\Sigma_c(X),s}$ with $a = A(\alpha)(t)$.*

*generated in S_c ,
if A is generated in S_c by Ω .*

The Σ -algebra A is called

freely generated in S_c by Ω_c , if:

for each sort $s \in S_c$:

for each element $a \in A(s)$:

there exists exactly one term $t \in T_{\Sigma_c(X),s}$ with $a = A(\alpha)(t)$.

freely generated in S_c ,

if A is freely generated in S_c by Ω .

Lists of Unspecified Elements

- $S = \{ \text{List}, \text{Elem} \}$
- $\Omega = \{ \text{nil}: \rightarrow \text{List},$
 $:: \quad : \text{Elem} \times \text{List} \rightarrow \text{List},$
 $++ \quad : \text{List} \times \text{List} \rightarrow \text{List} \}$
- $S_c = \{ \text{List} \}, \Omega_c = \{ \text{nil}, :: \}$

```
spec LIST =  
  sort Elem  
  free type List ::= nil | -- :: --(Elem; List)  
  op -- ++ -- : List × List → List  
  forall x : Elem; K, L : List  
    • nil ++ K = K  
    • (x :: K) ++ L = x :: (K ++ L)  
end
```

Specification with Constructors

Syntax

A *specification with constructors* is a tuple

$$sp = (\Sigma, \Phi, S_g, \Omega_g, S_f, \Omega_f),$$

where

- $\Sigma = (S, \Omega)$ is a signature,
- $\Phi \subseteq L(\Sigma)$ is a set of formulas,
- $S_g \subseteq S$, is the set of generated sorts,
- $S_f \subseteq S$, is the set of freely generated sorts, and
- $\Omega_g, \Omega_f \subseteq \Omega$ is are sets of constructors.

Semantics

$$M(sp) := \{A \in Alg(\Sigma) \mid \begin{array}{l} A \models \Phi, \\ A \text{ is generated in } S_g \text{ by } \Omega_g, \\ A \text{ is freely generated in } S_f \text{ by } \Omega_f \end{array}\}$$

```
spec Set =  
  sort Elem  
  generated type Set ::= empty | insert(Elem;Set)  
  pred __isIn__: Elem * Set  
  forall e,f: Elem; S,T: Set  
  . not (e isIn empty)  
  . e isIn insert(f,S) <=> e=f \/ e isIn(S)  
  . S=T <=> (e isIn S <=> f isIn T)  
end
```

spec PEANO_ARITHMETIC =

free type $Nat ::= 0 \mid suc(Nat);$

ops $++ + ---, ++ * --- : Nat \times Nat \rightarrow Nat;$

pred $++ \leq = --- : Nat \times Nat$

forall $m, n : Nat$

- $0 \leq = n$
- $\neg suc(n) \leq = 0$
- $suc(n) \leq = suc(m) \Leftrightarrow n \leq = m$
- $n + 0 = n$
- $n + suc(m) = suc(n + m)$
- $n * 0 = 0$
- $n * suc(m) = n + (n * m)$

spec INTEGER =

free type $Pos ::= 1 \mid suc(Pos);$

free type $Int ::= 0 \mid pos(Pos) \mid neg(Pos);$

pred $-- \leq -- : Int \times Int;$

forall $p, q : Pos$

- $0 \leq pos(p)$
- $\neg pos(p) \leq 0$
- $neg(p) \leq 0$
- $\neg 0 \leq neg(p)$
- $neg(p) \leq pos(p)$
- $\neg pos(p) \leq neg(p)$

- $0 \leq 0$
- $\text{pos}(1) \leq \text{pos}(p)$
- $\text{pos}(\text{suc}(p)) \leq \text{pos}(1)$
- $\text{pos}(\text{suc}(p)) \leq \text{pos}(\text{suc}(q)) = \text{pos}(p) \leq \text{pos}(q)$
- $\text{neg}(p) \leq \text{neg}(1)$
- $\neg \text{neg}(1) \leq \text{neg}(\text{suc}(p))$
- $\text{neg}(\text{suc}(p)) \leq \text{neg}(\text{suc}(q)) = \text{neg}(p) \leq \text{neg}(q)$

end