CASL

Common Algebraic Specification Language

CASL/Slang Examples

Peter D. Mosses (BRICS, Aarhus, Denmark)

External Relations Coordinator of

CoFI: The Common Framework Initiative

for algebraic specification and development

spec NAT_BASIC = %% ~ Specware Language Manual, Fig. 1

sort *Nat*

op *zero : Nat*

pred $non_zero(x : Nat) \Leftrightarrow \neg(x = zero)$

sort $Pos = \{ x{:}Nat \bullet non_zero(x) \}$

op $succ : Nat \rightarrow Pos$

vars $x, y : Nat$

- $non_zero(succ(x))$
- $succ(x) = succ(y) \Rightarrow x = y$

generated { **sorts** $Pos < Nat$ **ops** $zero : Nat;\ succ : Nat \rightarrow Pos$ }

axiom $\forall x : Pos \bullet \exists y : Nat \bullet\ x = succ(y)$

%% **More concisely:**

spec NAT_BASIC **=**

free types *Nat* *::=* *zero* | *sort Pos;*

Pos *::=* *succ(Nat)*

pred *non_zero(x : Nat)* $\Leftrightarrow$ $\neg$*(x = zero)*

spec PARTIAL_ORDER = %% Specware User's Guide, Ch. 2

sort E

pred $leq : E \times E$

vars $x, y, z : E$

- %[transitivity]

  $leq(x,y) \wedge leq(y,z) \Rightarrow leq(x,z)$

- %[reflexivity]

  $leq(x,x)$

- %[antisymmetry]

  $leq(x,y) \wedge leq(y,x) \Rightarrow x = y$

end

```
%% Product sorts in (first-order) CASL:
spec  ACCOUNT_IMPL = %% Specware User's Guide, Ch. 2
    INTEGER and STRING
    then
      free type Account ::= make_account(owner:String; balance:Integer)

%% Sum sorts in CASL:
spec ERROR_VALUE = %% Specware User's Guide, Ch. 2
    sorts          Integer, Error
    free type      Return_Value ::= sort Integer | sort Error
```

%% **'Function sorts' in (first-order) CASL:**

spec MAP_SEQ [**sort** *E* **op** $f : E \to E$] = %% Specware User's Guide, Ch. 2

SEQ [**sort E**] **with** *Seq[E], empty_seq, prepend*

then

op *map[f]* : $Seq[E] \to Seq[E]$

vars $e : E; s : Seq[E]$

- $(\forall e{:}E \bullet f(e) = e) \Rightarrow map[f]\ (s) = s$
- $map[f]\ (empty_seq) = empty_seq$
- $map[f]\ (prepend(e,s)) = prepend(f(e), map[f]\ (s))$

%% **Subsorts in CASL:**

spec EVEN **=** %% Specware User's Guide, Ch. 2

NAT

then

sort *Even* = { *n:Nat* • *is_even(n)* }

op *double(n:Nat):Even* = *plus(n,n)* *as* *Even*

%% **Partial operations in CASL:**

spec DIVISION = %% Specware User's Guide, Ch. 2

sort *Real*

sort *Nonzero_Real = { r:Real • ¬(r=0) }*

free type *Real_with_Error ::= error | sort Real*

ops *div_with_subsort : Real × Nonzero_Real → Real ;*

div_with_sum : Real × Real → Real_with_Error ;

div_partial : Real × Real →? Real

vars *r, s, t : Real*

- *div(r, div(s, t)) = div(mult(r, t), s) if ¬(t=0)*

%% **Parameterized specifications in CASL:**

spec PAIR [**sort** *D1*] [**sort** *D2*] = %% Specware User's Guide, Ch. 3

free type *Pair[D1,D2]* ::= *make_pair(first:D1; second:D2)*

spec AD_PAIR =

ALPHA **and** DIGIT **then**

PAIR [ALPHA **fit** *D1* $\mapsto$ *Alpha*] [DIGIT **fit** *D2* $\mapsto$ *Digit*]

spec SET [**sort** *E*] = **sort** *Set[E]* **ops** …

spec SET_OF_PAIRS [**sort** *D1*] [**sort** *D2*] =

SET [PAIR [**sort** *D1*] [**sort** *D2*] **fit** *E* $\mapsto$ *Pair[D1,D2]*]