

Quick Overview of VDM Operators

General

if *predicate* **then** Expression **else** Expression

exists p **in set** setOfP & p = value

cases expression:

(pattern list 1)→ Expression 1,
(pattern list 2),
(pattern list 3)→ Expression 2,
others → Expression 3
end;

for all value **in set** setOfValues
do Expression

dcl variable : *type* := Variable creation ;

let variable : *type* = Variable creation **in Expression**

The Boolean type

Operator	Name	Signature
not b	Negation	bool → bool
a and b	Conjunction	bool * bool → bool
a or b	Disjunction	bool * bool → bool
a => b	Implication	bool * bool → bool
a <=> b	Biimplication	bool * bool → bool
a = b	Equality	bool * bool → bool
a <> b	Inequality	bool * bool → bool

The numeric types

Operator	Name	Signature
-x	Unary minus	real → real
abs x	Absolute value	real → real
x + y	Sum	real * real → real
x - y	Difference	real * real → real
x * y	Product	real * real → real
x / y	Division	real * real → real
x div y	Integer division	int * int → int
x mod y	Modulus	int * int → int
x**y	Power	real * real → real
x < y	Less than	real * real → bool
x > y	Greater than	real * real → bool
x <= y	Less or equal	real * real → bool
x >= y	Greater or equal	real * real → bool
x = y	Equal	real * real → bool
x <> y	Not equal	real * real → bool

The character, quote and token types

Operator	Name	Signature
c1 = c2	Equal	char * char → bool
c1 <> c2	Not equal	char * char → bool

Tuple types

Operator	Name	Signature
t1 = t2	Equality	T * T → bool
t1 <> t2	Inequality	T * T → bool

Record types

Operator	Name	Signature
r.i	Field select	A * Id → Ai
r1 = r2	Equality	A * A → bool
r1 <> r2	Inequality	A * A → bool
is A (r1)	Is	Id * MasterA → bool

Union and optional types

Operator	Name	Signature
t1 = t2	Equality	A * A → bool
t1 <> t2	Inequality	A * A → bool

Set types

Operator	Name	Signature
e in set s1	Membership	A * set of A → bool
e not in set s1	Not membership	A * set of A → bool
s1 union s2	Union	set of A * set of A → set of A
s1 inter s2	Intersection	set of A * set of A → set of A
s1 \ s2	Difference	set of A * set of A → set of A
s1 subset s2	Subset	set of A * set of A → bool
s1 = s2	Equality	set of A * set of A → bool
s1 <> s2	Inequality	set of A * set of A → bool
card s1	Cardinality	set of A → nat
dunion ss	Distributed union	set of set of A → set of A
dinter ss	Distributed intersection	set of set of A → set of A

Sequence types

Operator	Name	Signature
hd l	Head	seq1 of A → A
tl l	Tail	seq1 of A → seq of A
len l	Length	seq of A → nat
elems l	Elements	seq of A → set of A
inds l	Indices	seq of A → set of nat1
l1 ^ l2	Concatenation	(seq of A) * (seq of A) → seq of A
conc ll	Distributed concatenation	seq of seq of A → seq of A
l ++ m	Sequence modification	seq of A * map nat to A → seq of A
l(i)	Sequence index	seq of A * nat1 → A
l1 = l2	Equality	(seq of A) * (seq of A) → bool
l1 <> l2	Inequality	(seq of A) * (seq of A) → bool

Mapping types

Operator	Name	Signature
dom m	Domain	(map A to B) → set of A
rng m	Range	(map A to B) → set of B
m1 munion m2	Map union	(map A to B) * (map A to B) → map A to B
m1 ++ m2	Override	(map A to B) * (map A to B) → map A to B
merge ms	Distributed merge	set of (map A to B) → map A to B
s <: m	Domain restrict to	(set of A) * (map A to B) → map A to B
s <-: m	Domain restrict by	(set of A) * (map A to B) → map A to B
m :> s	Range restrict to	(map A to B) * (set of B) → map A to B
m :-> s	Range restrict by	(map A to B) * (set of B) → map A to B
m(d)	Mapping apply	(map A to B) * A → B
m1 = m2	Equality	(map A to B) * (map A to B) → bool
m1 <> m2	Inequality	(map A to B) * (map A to B) → bool

Class Example

```
class Person

types
public String = seq of char;

values

protected Name : seq of char = "Peter";

instance variables

public nationality : seq of char := "Danish";
comment          : String;
yearOfBirth      : int;
sex              : Male | Female;
friends          : map String to Person;

operations

public GetAge : int ==> int
GetAge (year) == CalculateAge (year, yearOfBirth)
pre pre_CalculateYear (year, yearOfBirth);

functions

public CalculateAge : int * int -> int
CalculateAge (year, bornInYear) == year - bornInYear
pre year >= bornInYear;

thread
while true do
  skip;

end Person

class Male is subclass of Person
end Male
class Female is subclass of Person
end Female
```

Copyright © 2008 Overture

Revision: 1.00 , Date: 2008/10/26.

<http://www.vdmportal.org/twiki/bin/view/Main/VDMPPoverview>

Listing 1: Class Example