Métodos Formais em Engenharia de Software

Ana Paiva

apaiva@fe.up.pt

Universidade do Porto
Faculdade de Engenharia
FEUP

Operators on finite functions (maps)

Operador	Nome	Descrição	Tipo
dom m	Domain	Gives the domain (key set) of m	map *A* to *B* → set of *A*
rng m	Co-domain (range)	Gives the co-domain (set of values corresponding to keys) of m	map *A* to *B* → set of *B*
m1 munion m2	Merge	Makes a union of key-value pairs exist in m1 and m2, which must be compatible (they can not match different values to equal keys)	(map *A* to *B*) * (map *A* to *B*) → map *A* to *B*
m1 ++ m2	Override	Union with unrestricted compatibility. In case of dispute, m2 prevails.	
merge ms	Distributed union	Does the union of the mappings contained in ms that should be compatible.	set of (map *A* to *B*) → map *A* to *B*

Operators on finite functions (maps)

Operador	Nome	Descrição	Tipo
s <: m	Domínio restrito a	Dá o mapeamento constituído pelos elementos de `m` cuja chave está em `s` (que não tem de ser um subconjunto de `dom m`)	(set of A) * (map *A* to *B*) → map *A* to *B*
s <-: m	Domínio restrito por	Dá o mapeamento constituído pelos elementos de `m` cuja chave não está em `s` (que não tem de ser um subconjunto de `dom m`)	
m :> s	Contra-domínio restrito a	Dá o mapeamento constituído pelos elementos de `m` cujo valor de informação está em `s` (que não tem de ser um subconjunto de `rng m`)	(map *A* to *B*) * (set of *B*) → map *A* to *B*
m :-> s	Contra-domínio restrito por	Dá o mapeamento constituído pelos elementos de `m` cujo valor de informação não está em `s` (que não tem de ser um subconjunto de `rng m`)	

Operators on finite functions (maps)

Operador	Nome	Descrição	Tipo
m(d)	Aplicação de mapeamento	Dá o valor corresponde à chave `d` por `m`. A chave `d` deve existir no domínio de `m`.	(map *A* to *B*) * *A* → *B*
m1 comp m2	Composição de mapeamentos	Dá `m2` seguido de `m1`. O mapeamento resultante tem o mesmo domínio que `m2`. O valor correspondente a cada chave é obtido aplicando primeiro `m2` e depois `m1`. Restrição: `rng m2 subset dom m1`.	(map *B* to *C*) * (map *A* to *B*) → map *A* to *C*
m ** n	Iteração	Composição de `m` consigo próprio `n` vezes. Se `n=0`, dá a função identidade, em que cada elemento do domínio é mapeado para si próprio. Se `n=1`, dá `m`. Se `n>1`, `rng m` deve ser um subconjunto de `dom m`.	(map *A* to *A*) * nat → map *A* to *A*
inverse m	Mapeamento inverso	Dá o inverso de `m`, que deve ser injectivo.	inmap *A* to *B* → inmap *B* to *A*

Exercises (maps)

- dom {100 |-> <TIM>, 10 |-> <ROB>, 12 |-> <DAVE>}
- rng {100 |-> <TIM>, 10 |-> <ROB>, 12 |-> <DAVE>}
- {1000 |-> 3, 1005 |-> 4, 1002 |-> 1} ++ {1002 |-> 6}
- {1008 |-> 3, 1065 |-> 4, 1012 |-> 1} ++ {1011 |-> 6}
- {128} <: {100 |-> <TIM>, 10 |-> <ROB>, 12 |-> <DAVE>}
- {128} <-: {100 |-> <TIM>, 10 |-> <ROB>, 12 |-> <DAVE>}

Instructions/Expressions

- **Expressions:** return values
- **Instructions:** change system state, i.e., create, delete or change the state of objects (or the state of static variables) (e.g., assignment)
- For the model to be executable, you must write the body of transactions in the form of a statement or block of statements
- The body is also called "algorithmic body" because, while the postcondition specifies the "what" (effect), it is stated in the body "as" (algorithm)
- The VDM++ language allows to describe and test the algorithm to a high level of abstraction, refine it to the desired level, and generate an executable program in Java or C++ with the VDM Tools

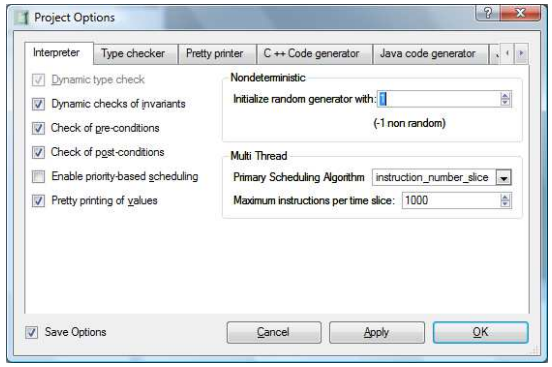
Expressions: examples

Expression	Example
Set enumeration	{a,3,3,**true**}
Set Comprehension	{a+2 | **mk_**(a,a) in set {**mk_**(**true**,1),**mk_**(1,1)}}
Set: type binding	{a | a: **nat** & a<10}
Set range	{3, ..., 10}
Seq enumeration	[7.7, **true**, "l",**true**]
Seq comprehension	[i*i | i **in set** {1,2,4,6}]
Subsequence	[4,**true**,"string",9,4](2,...,4)
Map enumeration	{1|->**true**, 7|->6}
Map comprehension	**Ex1:** {i|->**mk_**(i,**true**) | i: **bool**} **Ex2:** {a+b|-> b-a | a **in set** {1,2}, b **in set** {3,6}}
Tuple	mk_(2, 7, **true**, {|->})

Expressions: examples

Expression	Example
lambda	**Ex1:** **lambda** n: **nat** & n * n **Ex2:** **lambda** s: **nat**, b: **bool** & **if** b **then** a **else** 0
skip	**if** a <> [] **then** str := str ^ a **else skip** -- Para indicar que nenhuma acção foi executada.
error	**if** a = <OK> **then** DoSomething() **else error** -- O resultado é indefinido pelo que ocorreu um erro.
nondeterministic	|| (stmt1, stmt2, ..., stmtn)
iota	**iota** bind **&** expression -- it returns the unique value which satisfies the body expression e.g., iota x in set {sc1,sc2,sc3,sc4} & x.team = <France>

Nondeterministic



Exercises

dom {mk_(1,2).#1 |-> 3, mk_(2,3).#2 |-> 4}

[[5,6],[3,1,1],[5]] ++ {2 |-> [5,5],3 |-> [8]}

{mk_(x,y) | x in set elems ([1,2,2,1] ^ [2]), y in set inds [0,1] & x<=y}

{x|->y | x in set dom ({1|->2,2|->3} :> {3}), y in set rng {1|->4} & y = x*2}

conc ([[1,2],[2],[3,2]] ++ {1|->[3]})

{1|->2,2|->1,4|->4} munion ({1|->1,2|->2} ++ {1|->2,2|->1,3|->1})