

# Step2Core Project

## Testing

Anna Kukuy & Valeriya Bodnya

# Augmentation

```
Module System SimpleTest:
```

```
Module simple:
```

```
out x: int where x = 0
```

```
local z: int where z = 150
```

```
Transition t1:
```

```
modvar x,z
```

```
enable x = 0  $\wedge$  z = 0
```

```
assign x := z+1, z := x+2
```

```
modrel x' > 3
```

```
EndModule
```

```
Module sim = a:simple
```

```
Module sim1 = Augment( b:sim  
  where out grants: int grants = x)
```

```
--Module System SimpleTest
```

```
VAR x: int
```

```
grants: int
```

```
Module simple () {
```

```
  VAR z: int
```

```
Transition t1:
```

```
  enable: ((x = 0)  $\wedge$  (z = 0));
```

```
  assign: x := (z + 1);
```

```
          z := (x + 2);
```

```
  relation: (x' > 3);}
```

```
Module module_st_0() {
```

```
Transition t:
```

```
  enable: true;
```

```
  assign: x' := x;
```

```
  relation: (grants = x);}
```

```
Module sim1() {
```

```
(simple () || (module_st_0 ()))}
```

# Hiding variables

```
Module System SimpleTest:
```

```
Module simple:
```

```
out x: int where x = 0
```

```
local z: int where z = 150
```

```
Transition t1:
```

```
modvar x,z
```

```
enable x = 0  $\wedge$  z = 0
```

```
assign x := z+1, z := x+2
```

```
modrel x' > 3
```

```
EndModule
```

```
Module sim1 = a:simple
```

```
Module sim2 = Hide(b:sim1 x)
```

```
--Module System SimpleTest
```

```
Module simple_step_hiding_1()
```

```
VAR
```

```
z: int
```

```
x: int
```

```
Transition t1:
```

```
enable: ((x = 0)  $\wedge$  (z = 0));
```

```
assign: x := (z + 1);
```

```
z := (x + 2);
```

```
relation: (x' > 3);
```

```
}
```

```
Module sim1_step_hiding_0() {
```

```
simple_step_hiding_1 ()
```

```
}
```

```
Module sim2() {
```

```
sim1_step_hiding_0 ()}
```

# Composition of modules

```
Module System SimpleTest:  
type a = [ 1..2 ]
```

```
Module simple:  
out x: int where x=4  
local z: int where z = 150  
out w: int where w =200  
export t1
```

```
Transition t1:  
  modvar x,z  
  enable x = 0  $\wedge$  z = 0  
  assign x := z+1, z := x+2  
  modrel x' > 3  $\wedge$  w' >= 5  
EndModule
```

```
Module sim1 = a1:simplella2:simple
```

```
--Module System SimpleTest  
  TYPE
```

```
a : [1..2]  
  VAR
```

```
x,w: int
```

```
Module simple
```

```
VAR
```

```
z: int
```

```
Transition t1:
```

```
  enable: ((x = 0)  $\wedge$  (z = 0));
```

```
  assign: x := (z + 1);
```

```
          z := (x + 2);
```

```
  relation: ((x' > 3)  $\wedge$  (w' >= 5));
```

```
}
```

```
Module sim1() {
```

```
(simple ()|(t1,t1)|(simple ())}
```

# Renaming of transitions & vars

```
Module System SimpleTest:
```

```
Module simple:
```

```
out x: int where x = 0
```

```
local z: int where z = 150
```

```
out w: int where w = 200
```

```
export t1,t2
```

```
Transition t1:
```

```
modvar x,z
```

```
enable x = 0  $\wedge$  z = 0
```

```
assign x := z+1, z := x+2
```

```
modrel x' > 3  $\wedge$  w' >= 5
```

```
Transition t2:
```

```
modvar x,z
```

```
enable x = 0  $\wedge$  z = 0
```

```
assign x := z+1, z := x+2
```

```
modrel x' > 3  $\wedge$  w' >= 5
```

```
EndModule
```

```
Module sim1 = s:simple
```

```
Module sim2 = Rename(Rename( b:sim1  
  where Transition t2 = t) where out  
  x1,w1: int x=x1, w=w1, Transition t1  
  = tt)
```

```
--Module System SimpleTest VAR
```

```
w1,x1: int
```

```
Module
```

```
simple_step_renaming_1_step_renaming_3() {
```

```
VAR
```

```
z: int
```

```
Transition tt:
```

```
enable: ((x1 = 0)  $\wedge$  (z = 0));
```

```
assign: x1 := (z + 1);
```

```
z := (x1 + 2);
```

```
relation: ((x1' > 3)  $\wedge$  (w1' >= 5));
```

```
Transition t:
```

```
enable: ((x1 = 0)  $\wedge$  (z = 0));;
```

```
assign: x1 := (z + 1); z := (x1 + 2);
```

```
relation: ((x1' > 3)  $\wedge$  (w1' >= 5));}
```

```
Module
```

```
sim1_step_renaming_0_step_renaming_2() {
```

```
simple_step_renaming_1_step_renaming_3 ()}
```

```
Module sim2() {
```

```
sim1_step_renaming_0_step_renaming_2 ()}
```

# Parameters

```
Module System SimpleTest:
```

```
Module simple(h:int) :
```

```
out x: int where x =0
```

```
local z: int where z = 150
```

```
out w: int where w =200
```

```
external in m:int where m = 300
```

```
export t1
```

```
Transition t1:
```

```
modvar x,z
```

```
enable x = 0  $\wedge$  z = 0
```

```
assign x := z+1, z := x+2
```

```
modrel x' > 3  $\wedge$  w' >= 5
```

```
EndModule
```

```
Module sim1(k:int) =
```

```
  b:simple(k)||bb:simple(k)
```

```
Module sim2 = a:sim1(1)
```

```
--Module System SimpleTest
```

```
VAR
```

```
x,m,w: int
```

```
Module simple(h) VAR
```

```
z: int
```

```
Transition t1:
```

```
enable: ((x = 0)  $\wedge$  (z = 0));
```

```
assign: x := (z + 1);
```

```
z := (x + 2);
```

```
m' := m;
```

```
relation: ((x' > 3)  $\wedge$  (w' >= 5));}
```

```
Module sim1(k) {
```

```
(simple (k)|(t1,t1)|(simple (k)))}
```

```
Module sim2() {
```

```
sim1 (1)[ (t1,t1) -> t1_simple_t1_simple]
```

```
}
```

# Translation of variables of modes external in, external out, out

Module System SimpleTest:

Module simple1 :

out x: int where x = 0

local z: int where z = 150

out w: int where w = 200

external in m: int where m = 300

external out o: int where o = 400

export t1

Transition t1:

modvar x, z

enable  $x = 0 \wedge z = 0$

assign  $x := z + 1, z := x + 2$

modrel  $x' > 3 \wedge w' \geq 5$

EndModule

--Module System SimpleTest VAR

x, m, w, o: int

Module simple2( ) {

VAR

z, x: int

Transition t1:

enable:  $((x = 0) \wedge (z = 0));$

assign:  $x := (z + 1);$

$z := (x + 2);$

$w' := w;$

relation:  $(x' > 3);$

}

Module simple1( )

VAR

z: int

# Continuing of the previous example + long composition

```
Module simple2 :
```

```
local x: int where x = 0  
local z: int where z = 150  
export t1
```

```
Transition t1:
```

```
modvar x,z  
enable x = 0  $\wedge$  z = 0  
assign x := z+1, z := x+2  
modrel x' > 3
```

```
EndModule
```

```
Module sim1 =
```

```
  b:simple1 || bb:simple2 || bbb:simple1
```

```
Transition t1:
```

```
enable: ((x = 0)  $\wedge$  (z = 0));  
assign: x := (z + 1);  
        z := (x + 2);  
        m' := m;  
relation: ((x' > 3)  $\wedge$  (w' >= 5));  
}
```

```
Module module_comp_0() {  
  (simple1 ()|(t1,t1)|(simple1 ()))
```

```
Module sim1() {  
  (simple2 ()| (t1,t1_simple1_t1_simple1)|  
  (module_comp_0 ()[(t1,t1) ->  
  t1_simple1_t1_simple1])  
}
```