

Tools for teaching formal methods

fm4fun

fm4fun

Formal Methods - A Learning Environment

Input Program

```
y:=1;
do x>0 -> y:=x*y;
  x:=x-1
od
```

Extended Guarded Commands Code

Step-wise Execution

Explore how the memory changes, upon execution of the actions. Recall that if more guards are satisfied then it is non-deterministic which action to take.

When using the **Show Trace** feature, non-deterministic choices will be shown in the first column as **q2** and if clicked the trace will be "reset" from that configuration.

When using the **Show Resulting Configurations** feature, all non-deterministic choices will be explored, until it reaches the number of steps, gets stuck or terminates. A short summary of the final configurations is found at the end. **Warning** If the input code is highly non-deterministic, and the number of steps is high, the tree of possible end-configurations will grow big (and therefore cause long execution).

For both outputs, all non-terminated and non-stuck configurations can be extended with 1-1000 steps.

Initialization of Variables and Arrays

Number of Steps

Show Trace Show Resulting Configurations

Action	Node	x	y
	q0	3	0
y:=1	q1	3	1
x>0	q2	3	1
y:=x*y	q3	3	3
x:=x-1	q1	2	3
x>0	q2	2	3
y:=x*y	q3	2	6
x:=x-1	q1	1	6
x>0	q2	1	6
y:=x*y	q3	1	6
x:=x-1	q1	0	6
!(x>0)	q4	0	6

Successfully terminated in 11 steps.

Program Graph

Non-det. Det.

moka

Moka

```
1 > n = 0
2 > turn = 0
3 > wants_to_enter_0 = 0
4 > wants_to_enter_1 = 0
5 > in_0 = 0
6 > in_1 = 0
7 par
8   wants_to_enter_0:=1;
9   do wants_to_enter_1=1 -> if !(turn=0) -> wants_to_enter_0:=0;
10      do !(turn=0) -> skip
11      od;
12      wants_to_enter_0:=1
13 fi
14 od;
15 n:=n+1;
16 in_0:=1;
17 in_1:=0;
18 n:=n-1;
19 turn:=1;
20 wants_to_enter_0:=0
21 []
22 wants_to_enter_1:=1;
23 do wants_to_enter_0=1 -> if !(turn=1) -> wants_to_enter_1:=0;
24      do !(turn=1) -> skip
25      od;
26      wants_to_enter_1:=1
27 fi
28 od;
29 LTL property does not hold
30 n:= Step in_0 in_1 n turn wants_to_enter_0 wants_to
31 in_1 0 0 0 0 0
32 in_2 0 0 0 0 0
33 n:= 3 0 0 0 0 1
34 tur 4 0 0 0 0 1
35 wan 5 0 0 0 0 1
36 rap 6 0 0 0 0 1
37 rap 7 0 0 0 0 1
38 check G ((wants_to_enter_0 = 1) ==> F in_0 = 1) // fairness
39
40
41 Error
```

chip

Chip

```
1 { x = n & n >= 0 }
2 y := 1;
3 do [ x >= 0 & y * fac(x) = fac(n) ]
4   x > 0 ->
5     y := y * x;
6     x := x - 1
7 od
8 { y = fac(n) }
9
```

Verified

The program is **not** fully annotated

inspectify

Inspectify

Calculator Parser Compiler Interpreter Security Sign

Generate

```
y:=1;
do x>0 -> y:=x*y;
  x:=x-1
od
```

Initialization of variables and arrays

x 3

y 0

Options

Number of steps 14

Determinism Deterministic NonDeterministic

Jobs:

Action	Node	x	y
y := 1	q0	3	0
(x > 0)	q1	3	1
y := (x * y)	q2	3	1
x := (x - 1)	q3	3	3
(x > 0)	q1	2	3
y := (x * y)	q2	2	3
x := (x - 1)	q3	2	6
(x > 0)	q1	1	6
y := (x * y)	q2	1	6
x := (x - 1)	q3	1	6
!(x > 0)	q4	0	6

Terminated

BSc projects:

- > New tools
- > Gamification of existing tools
- > Improvements of existing tools:
 - > Smarter test case generators
 - > Make them run blazingly fast
 - > More informative feedbacks
 - > ...

Why?

- > Learn more about CS theory, algorithms, and FM
- > Improve learning experience of future students
- > Work with Alberto, Andrey, Oliver, ... :)

See <https://gitlab.gbar.dtu.dk/O2141/student-projects/>