Damian Koper **241292**

Modelowanie i analiza systemów informatycznych

Logika Temporalna i Automaty Czasowe - konstrukcja i weryfikacja automatów NuSMV do analizy programu.

Zadanie 1.

Liczba liczb pierwszych

```
1  MODULE main
2  FROZENVAR
3      number : 0..8;
4  VAR
5      e : {e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e13};
6      j : 3..9;
7      i : 1..7;
8      jm : 1..7;
9      x : 0..6;
10     factorial : {1,2,6,24,120,720};
11     sigma : 0..5;
12     primes : 0..4;
13     c : 0..6;
14 INIT
15     e = e1 &
16     j = 3 &
17     i = 1 &
18     jm = 1 &
19     x = 0 &
```

```
20      primes = 0 &
21      factorial = 1 &
22      sigma = 0 &
23      c = 0;
24  TRANS next(jm) = j − 2;
25  TRANS
26      next(e) in case
27          e=e1 & number > 2 : e2;
28          e=e1 & number <= 2 : e13;
29          e=e2 & j <= number : e3;
30          e=e2 & j > number : e9;
31          e=e3 : e4;
32          e=e4 : e5;
33          e=e5 & x < jm : e6;
34          e=e5 & x >= jm : e8;
35          e=e6 : e7;
36          e=e7 : e5;
37          e=e8 : e2;
38          e=e9 : e10;
39          e=e10 & number = 3 : e11;
40          e=e10 & number != 3 : e12;
41          TRUE : e;
42      esac;
43  TRANS
44      next(j) in case
45          e=e1 & next(e)=e2 : 3;
46          e=e8 & next(e)=e2 : j + 1;
47          TRUE : j;
48      esac;
```

```
49 TRANS
50     next(i) in case
51         e=e3 & next(e)=e4 : 1;
52         e=e6 & next(e)=e7 : i + 1;
53         TRUE : i;
54     esac;
55 TRANS
56     next(x) in case
57         e=e4 & next(e)=e5 : 0;
58         e=e7 & next(e)=e5 : x + 1;
59         TRUE : x;
60     esac;
61 TRANS
62     next(factorial) in case
63         e=e2 & next(e)=e3 : 1;
64         e=e5 & next(e)=e6 : factorial * i;
65         TRUE : factorial;
66     esac;
67 TRANS
68     next(x) in case
69         e=e4 & next(e)=e5 : 0;
70         e=e7 & next(e)=e5 : x + 1;
71         TRUE : x;
72     esac;
73 TRANS
74     next(sigma) in case
75         next(e)=e8 : sigma + (factorial - (j * (factorial / j)));
76         TRUE : sigma;
77     esac;
```

```
78  TRANS
79      next(primes) in case
80          next(e)=e9 : -1 + sigma;
81          TRUE : primes;
82      esac;
83  TRANS
84      next(c) in case
85          next(e)=e3 : c + 1;
86          TRUE : c;
87      esac;
88  DEFINE
89      begin := e=e1;
90      end := e=e11 | e=e12 | e=e13;
91  CTLSPEC number in 0..2 -> AF(e = e13)
92  -- jesli number in 0..2 to zawsze kiedyś e=e13
93  -- true
94  COMPUTE MAX [number in 3..8 & begin, number in 3..8 & end] -- 97
95  COMPUTE MIN [number in 3..8 & begin, number in 3..8 & end] -- 12
96  -- minimalna i maksymalna liczba stanów dla number in 3..8 od początku do końca
97  CTLSPEC number = 3 -> !EBF 12..12 (end) -- false (wypisanie ścieżki)
98  -- -> State: 1.13 <-
99  -- e = e11
100 -- end = TRUE
101 --
102 ---- e11 -> cout << 2;
103 CTLSPEC number = 8 -> !EBF 97..97 (end) -- false (wypisanie ścieżki)
104 -- -> State: 2.96 <-
105 -- e = e9
106 -- jm = 7
```

```
-- primes = 4
-- -> State: 2.97 <-
-- e = e10
-- -> State: 2.98 <-
-- e = e12
-- end = TRUE
--
---- e12 -> cout << primes // 4;
COMPUTE MIN [number = 6 & begin, number = 6 & end] -- 54
-- minimalna liczba stanów dla number=6 od początku do końca
CTLSPEC number = 6 -> !EBF 54..54 (end) -- false (wypisanie ścieżki)
-- -> State: 3.36 <-
-- c = 4
CTLSPEC number = 6 -> AF(c=4)
-- jeśli number=6 to zawsze kiedyś c=4
-- true
CTLSPEC number = 6 -> AG(c<=4)
-- jeśli number=6 to zawsze c<=4
-- true
```
