Damian Koper **241292**

Modelowanie i analiza systemów informatycznych

Logika Temporalna i Automaty Czasowe - konstrukcja i weryfikacja zsynchronizowanych automatów NuSMV.

Zadanie 1.

Zamek

```
1  MODULE main
2  VAR locked : boolean;
3      state : 0..4;
4      n : 0..9;
5  INIT
6      locked = TRUE &
7      state = 0;
8  TRANS next(locked)
9      in case
10         next(state) = 4 : FALSE;
11         TRUE : TRUE;
12     esac;
13 TRANS next(state)
14     in case
15         next(n) = 1 & state < 4 : 1;
16         next(n) = 2 & state = 1 : 2;
17         next(n) = 3 & state = 2 : 3;
18         next(n) = 4 & state = 3 : 4;
19         state = 4 : 4;
```

```
20          TRUE : 0;
21      esac;
22  CTLSPEC AG(state = 4 -> locked = FALSE)
23  -- zawsze odblokoway w state=4
24  -- true
25  CTLSPEC AG(state < 4 -> locked = TRUE)
26  -- zawsze zablokowany w state<4
27  -- true
28  CTLSPEC EF((n=1 & locked & EX(n=2 & locked & EX(n=3 & locked& EX(n=4 & !
        locked)))))
29  -- możliwe, że kiedyś n=1 i zablokowany, w następym stanie n=2 i zablokowany,
30  -- w następym stanie n=3 i zablokowany, w następym stanie n=4 i odblokowany
31  -- true
32  CTLSPEC AG(locked -> AX(n=1 -> state=1))
33  -- jeśli zablokowany to n=1 resetuje do state=1
34  -- true
35  CTLSPEC AG(locked & state=1 -> AX(n=2 -> state=2))
36  -- jeśli state=1 to n=2 przechodzi do state=2
37  -- true
38  CTLSPEC AG(locked & state=1 -> AX(n>=3 -> state=0))
39  -- jeśli state=1 to n>=3 przechodzi do state=0
40  -- true
41  CTLSPEC AG(locked & state=2 -> AX(n=3 -> state=3))
42  -- jeśli state=2 to n=3 przechodzi do state=3
43  -- true
44  CTLSPEC AG(locked & state=2 -> AX(n=2 | n>=4 -> state=0))
45  -- jeśli state=2 to n=2|n>=4 resetuje do state=0
46  -- true
47  CTLSPEC AG(locked & state=3 -> AX(n=4 -> state=4))
```

```
48 -- jeśli state=3 to n=4 przechodzi do state=4
49 -- true
50 CTLSPEC AG(locked & state=3 -> AX(n=2 | n=3 | n>=5 -> state=0))
51 -- jeśli state=3 to n=2|n=3|n>=5 resetuje do state=0
52 -- true
53 CTLSPEC AG(state=4 -> AX(state=4))
54 -- jeśli state=4 to n=? przechodzi do state=4
55 -- true
```

Zadanie 2.

Światła

```
1  --########################################## LIGHTS2
2  MODULE lights2(lights3)
3  VAR
4      state : {green, red};
5      awaitingLights3 : boolean;
6      t : 0..99;
7  INIT
8      state = red &
9      awaitingLights3 = FALSE &
10     t = 0;
11 TRANS
12     next(t) in case
13         next(state) != state : 0;
14         lights3.state != next(lights3.state): 0;
15         TRUE : (t+1) mod 99;
16     esac;
17 TRANS next(awaitingLights3) in case
18         state = green & next(state) = red : TRUE;
19         lights3.state = red & next(lights3.state) = red_yellow : FALSE;
20         TRUE : awaitingLights3;
21     esac;
22 TRANS
23     next(state) in case
24         state = red & !awaitingLights3 & t = 1 & lights3.state = red : {green, red};
25         state = red & !awaitingLights3 & t = 2 & lights3.state = red : green;
26         state = green & t = 15 : red;
27         TRUE : state;
```

```
28      esac;
29  CTLSPEC AG((state = green & AX(state = red)) -> t = 15)
30  -- zawsze state=green trwa t = 15
31  -- true
32  CTLSPEC AG((state = red & AX(state = green)) -> lights3.state = red & t in 1..2)
33  -- zmiana z state=red na state=green jeśli lights3.state=red w czasie t in 1..2
34  -- true
35  CTLSPEC EF(state = red & lights3.state = yellow & AX(lights3.state = red))
36  CTLSPEC AG(state = red & lights3.state = yellow & AX(lights3.state = red) -> t = 3)
37  -- jest możliwe, że kiedyś state = red i lights3.state = yellow, a potem lights3.state =
        red
38  -- zawsze jeżeli state = red i lights3.state = yellow, a potem lights3.state = red to t = 3
39  -- lights3.state=yellow trwa t = 3 i t jest zsynchronizowane pomiędzy lights2 i lights3
40  -- true
41  CTLSPEC AG(state = green -> AX(state = green | state = red))
42  CTLSPEC AG(state = red -> AX(state = red | state = green))
43  -- zawsze jeżeli state to zawsze po nim state ten sam, albo poprawny
44  -- true
45  COMPUTE MIN[state = red, state = green] -- 1
46  -- minimalny czas zmiany z state=red na state=green
47  --############################################ LIGHTS3
48  MODULE lights3(button, lights2)
49  VAR
50      state : {green, yellow, red, red_yellow};
51      awaitingLights2 : boolean;
52      t : 0..99;
53  INIT
54      state = green &
55      awaitingLights2 = FALSE &
```

```
56      t = 0;
57  TRANS
58      next(t) in case
59          next(state) != state : 0;
60          lights2.state != next(lights2.state): 0;
61          TRUE : (t+1) mod 99;
62      esac;
63  TRANS next(awaitingLights2) in case
64          state = yellow & next(state) = red : TRUE;
65          lights2.state = red & next(lights2.state) = green : FALSE;
66          TRUE : awaitingLights2;
67      esac;
68  TRANS
69      next(state) in case
70          state = green & t >= 60 & button.pressed : yellow;
71          state = yellow & t = 3 : red;
72          state = red & !awaitingLights2 & lights2.state = red & t = 1 : {red, red_yellow};
73          state = red & !awaitingLights2 & lights2.state = red & t = 2 : {red_yellow};
74          state = red_yellow & t = 3 : green;
75          TRUE : state;
76      esac;
77  CTLSPEC AG((state = green & AX(state = yellow)) -> t >= 60)
78  CTLSPEC EF(state = green & AX(state = yellow) & t = 60)
79  CTLSPEC !EF(state = green & AX(state = yellow) & t < 60)
80  -- zawsze state=green trwa t >= 60
81  -- możliwe jest, żeby state=green trwał t = 60
82  -- niemożliwe jest, żeby state=green trwał t < 60
83  -- true
84  CTLSPEC AG((state = yellow & AX(state = red)) -> t = 3)
```

```
85  CTLSPEC !EF(state = yellow & AX(state = red) & (t < 3 | t > 3))
86  -- zawsze state=yellow trwa t >= 60
87  -- niemożliwe jest, żeby state=yellow trwał t != 3
88  -- true
89  CTLSPEC AG((state = red_yellow & AX(state = green)) -> t = 3)
90  CTLSPEC !EF(state = red_yellow & AX(state = green) & (t < 3 | t > 3))
91  -- zawsze state=red_yellow trwa t >= 60
92  -- niemożliwe jest, żeby state=red_yellow trwał t != 3
93  -- true
94  CTLSPEC EF((state = red & AX(state = red_yellow)))
95  CTLSPEC AG((state = red & AX(state = red_yellow)) -> lights2.state = red & t in
        1..2)
96  -- zmiana z state=red na state=red_yellow jeśli lights2.state=red w czasie 1..2
97  -- true
98  COMPUTE MIN[state = red, state = red_yellow] -- 1
99  -- minimalny czas zmiany z state=red na state=red_yellow
100 CTLSPEC EF(state = red & lights2.state = green & AX(lights2.state = red))
101 CTLSPEC AG(state = red & lights2.state = green & AX(lights2.state = red) -> t = 15)
102 -- zawsze jeżeli state = red i lights2.state = green, a potem lights2.state = red to t=15
103 -- światło zielone dla pieszych trwa zawsze t = 15
104 -- true
105 CTLSPEC AG(state = green -> AX(state = green | state = yellow))
106 CTLSPEC AG(state = yellow -> AX(state = yellow | state = red))
107 CTLSPEC AG(state = red -> AX(state = red | state = red_yellow))
108 CTLSPEC AG(state = red_yellow -> AX(state = red_yellow | state = green))
109 -- zawsze jeżeli state to zawsze po nim state ten sam, albo poprawny
110 -- true
111 --################################### BUTTON
112 MODULE button(lights2)
```

```
113 VAR
114     pressed: boolean;
115     pressed_signal: boolean;
116     t : 0..5;
117 INIT
118     pressed = FALSE &
119     pressed_signal = FALSE &
120     t = 0;
121 TRANS
122     next(t) in case
123         !pressed_signal & next(pressed_signal) : 0;
124         TRUE : (t+1) mod 6;
125     esac;
126 TRANS
127     next(pressed_signal) in case
128         next(pressed) : FALSE;
129         pressed_signal : TRUE;
130         TRUE : {TRUE, FALSE};
131     esac;
132 TRANS
133     next(pressed) in case
134         pressed & !(lights2.state = green & next(lights2.state) = red) : TRUE;
135         pressed_signal & next(t) < 1 : FALSE;
136         pressed_signal & next(t) < 5 : {TRUE, FALSE};
137         pressed_signal & next(t) = 5 : TRUE;
138         TRUE: FALSE;
139     esac;
140 COMPUTE MIN[pressed_signal, pressed] -- 1
141 COMPUTE MAX[pressed_signal, pressed] -- 5
```

```
142 -- min i max czas między naciśnięciem a obsługą
143 CTLSPEC EF(pressed)
144 CTLSPEC EF(pressed_signal)
145 -- możliwe że kiedyś pressed i pressed_signal
146 -- true
147 CTLSPEC AG(pressed_signal -> ABF 1..5 (pressed))
148 -- zawsze jeśli pressed_signal to zawsze w przeciągu 1..5 stanów pressed
149 -- true
150 CTLSPEC AG(!pressed -> AX(pressed -> t in 1..5))
151 -- zawsze jesli !pressed to zawsze w następnym stanie, jeśli pressed, to czas jest pomiedzy
        1..5
152 -- true
153 CTLSPEC AG(pressed_signal -> AX(pressed -> t in 1..5))
154 -- zawsze jesli pressed_signal to zawsze w następnym stanie, jeśli pressed, to czas jest
        pomiedzy 1..5
155 -- true
156 CTLSPEC AG(pressed -> AX(AF !pressed))
157 -- zawsze jesli pressed to kiedyś zawsze !pressed
158 -- true
159 --################################### MAIN
160 MODULE main
161 VAR
162     button: button(lights2);
163     lights2: lights2(lights3);
164     lights3: lights3(button, lights2);
165 CTLSPEC AG(lights3.state = green -> lights2.state = red)
166 INVARSPEC lights3.state = green -> lights2.state = red
167 -- zawsze jeśli lights3.state=green to lights2.state=red
168 -- true
```

```
169 CTLSPEC AG(lights2.state = green -> lights3.state = red)
170 INVARSPEC lights2.state = green -> lights3.state = red
171 -- zawsze jeśli lights2.state=green to lights3.state=red
172 -- true
173 LTLSPEC button.pressed U lights2.state = red
174 -- button.pressed dopóki lights2.state = red
175 -- true
```
