```
/* TemaEsame_Schema_Inventato2: Esercizio 5 */
/* Logica Asmeta */

asm cassaforte

import StandardLibrary
import CTLlibrary
import LTLlibrary

signature:
    domain Sensore subsetof Integer

    // variabile che indica di quale è il
    // sensore di cui cambiare lo stato
    // esempio: Sensore = 1 -> provo a cambiare
    // lo stato del sensore numero 1
    monitored switchSensore : Sensore

    // variabile che restituisce se un sensore
    // è acceso o spento
    controlled statoSensore : Sensore -> Boolean

    // funzione derivata che dice se i sensori
    // adiacenti al sensore passato come parametro
    // sono entrambi accesi
    derived adiacenti : Sensore -> Boolean

    // funzione che restituisce true per un qualsiasi
    // sensore
    derived pericolo :  Boolean

definitions:
    domain Sensore = {0:3}

    // restiuisce true solo se i due sensori adiacenti a $s sono accesi
    function adiacenti($s in Sensore) =
        if($s = 0) then
            statoSensore(3) and statoSensore(1)
        else
            if($s=3) then
                statoSensore(0) and statoSensore(2)
            else
                statoSensore($s+1) and statoSensore($s-1)
            endif
        endif

    // ho pericolo se il sensore $s è spento e i sensori a lui adiacenti sono spenti
    function pericolo =
        ( forall $s in Sensore with not adiacenti($s) and not statoSensore($s) )

    // ---------------------------------------
    // PROPRIETA':
    // 1) Non posso mai avere una situazione di pericolo
    CTLSPEC ag(not pericolo)
    LTLSPEC g(not pericolo)

    // 2) Non posso mai avere i sensori 0 e 2 contemporaneamente spenti
    // Mi aspetto che questa proprietà sia falsa perché 0
    // e 2 non sono sensori adiacenti
    CTLSPEC ag(not (statoSensore(0)=false and statoSensore(2)=false) )
    // viene mostrato controesempio con sensore 0 e sensore 2 spenti insieme

```

```
63      // 3) Una volta spento un sensore può essere sempre riacceso
64      CTLSPEC (
65          forall $s in Sensore with
66              ag(statoSensore($s)=false implies ef(statoSensore($s)=true))
67      )
68
69      // 4) Un sensore acceso può essere sempre spento
70      // Mi aspetto che questa proprietà sia FALSA perché non è
71      // detto che possa essere sempre spento, dipende dagli adiacenti
72      CTLSPEC (
73          forall $s in Sensore with
74              ag(statoSensore($s)=true implies af(statoSensore($s)=false))
75      )
76      // viene mostrato un controesempio adatto
77
78      // 5) se il sensore 0 è acceso e quello 1 è spento,
79      // allora 0 rimarrà acceso almeno fino a quando 1
80      // non si accenderà (se si accenderà)
81      CTLSPEC ag( (statoSensore(0)=true and statoSensore(1)=false) implies aw(statoSensore
    (0)=true, statoSensore(1)=true))
82      // ----------------------------------------
83
84      main rule r_Main =
85          // se un sensore è spento posso accenderlo
86          // a prescindere dalla sua posizione
87          if(statoSensore(switchSensore)=false) then
88              statoSensore(switchSensore) := not statoSensore(switchSensore)
89          else
90              // se adiacenti = true, i sensori adiacenti al suo parametro
91              // sono entrambi accesi e quindi posso spegnere il parametro
92              if(adiacenti(switchSensore)) then
93                  statoSensore(switchSensore) := not statoSensore(switchSensore)
94              endif
95          endif
96
97  default init s0:
98      function statoSensore($s in Sensore) = true
```

```
1 /* TemaEsame_Schema_Inventato2: Esercizio 5 */
2 /* Scenario avalla */
3
4 scenario scenarioCassaforte
5
6 load cassaforte.asm
7
8 // scenario in cui:
9 // 0) controllo che tutti i sensori siano accesi
10 // 1) cerco di spegnere sensore 0 -> si spegne
11 // 2) cerco di spegnere sensore 1 -> NON si spegne
12 // 3) cerco di spegnere sensore 2 -> si spegne
13 // 4) riaccendo il sensore 0 -> si accende
14 // 5) riaccendo il sensore 2 -> si accende
15
16 // 0)
17 check statoSensore(0);
18 check statoSensore(1);
19 check statoSensore(2);
20 check statoSensore(3);
21
22 // 1)
23 set switchSensore := 0;
24 step
25 check statoSensore(0) = false;
26
27 // 2)
28 set switchSensore := 1;
29 step
30 check statoSensore(1);
31
32 // 3)
33 set switchSensore := 2;
34 step
35 check statoSensore(2) = false;
36
37 // 4)
38 set switchSensore := 0;
39 step
40 check statoSensore(0);
41
42 // 5)
43 set switchSensore := 2;
44 step
45 check statoSensore(2);
```