

TESTING

Dopo aver analizzato la specifica realizzo la seguente struttura dell'Ostello:

```
public class Ostello {

    private boolean lettiperStanza[][]= new boolean[10][5];
    private final static int numLetti = 50;

    public Ostello(){
        for(int i=0;i<10;i++)
            for(int j=0;j<5;j++)
                lettiperStanza[i][j]=true;
    }

    public boolean checkin(int stanza, int letto){
        //occupa letto specifico

        if(stanza<0 || stanza>9 || letto <0 || letto>4)
            return false;
        if(lettiperStanza[stanza][letto]){
            lettiperStanza[stanza][letto]=false;
            return true;
        }
        return false;
    }

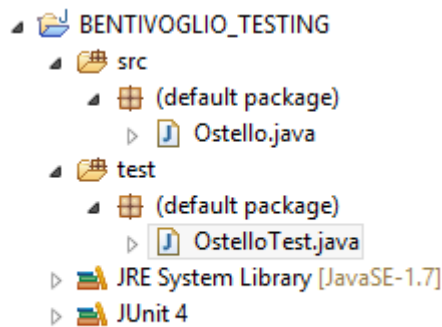
    public boolean checkin(int stanza){
        //occupa il primo letto libero della stanza
        if(stanza<0 || stanza>9)
            return false;
        for(int i=0;i<5;i++){
            if(lettiperStanza[stanza][i]){
                lettiperStanza[stanza][i]=false;
                return true;
            }
        }
        return false;
    }

    public boolean libera(int stanza){
        //una stanza è libera se ha tutti i letti liberi
        if(stanza<0 || stanza>9)
            return false;
        for(int i=0;i<5;i++){
            if(!lettiperStanza[stanza][i]){
                return false;
            }
        }

        return true;
    }
}
```

Scenari

Scrivo la classe di test OstelloTest con i relativi scenari in una cartella separata che chiamo test.



Eseguo gli scenari e controllo la copertura:

The screenshot shows the 'Coverage' view in the IDE. It displays a table with the following columns: Name, Statement, Branch, Loop, Term, ?-Operator, and Synchronized. The table shows the coverage for the 'BENTIVOGLIO_TESTING' project and its sub-project 'Ostello'.

Name	Statement	Branch	Loop	Term	?-Operator	Synchronized
BENTIVOGLIO_TESTING	46,2 %	41,7 %	?	50,0 %	?	?
Ostello	46,2 %	41,7 %	?	50,0 %	?	?

Come si vede la copertura non è del 100%.

Copertura istruzioni e branch

Aggiungo dei casi di test.

Scrivo dei casi di test in cui aggiungo un può di movimenti non coperti dal test precedente:

- provo a fare il checkin di una stanza che non esiste
- provo a fare il checkin di una stanza che non esiste
- provo ad occupare un letto in una stanza libera
- verifico stanza che non esiste libera
- verifico che una stanza sia libera
- verifico che una stanza sia libera ma in realtà non lo è

Ottingo la seguente copertura:

The screenshot shows the 'Coverage' view in the IDE. It displays a table with the following columns: Name, Statement, Branch, Loop, Term, ?-Operator, and Synchronized. The table shows the coverage for the 'BENTIVOGLIO_TESTING' project and its sub-project 'Ostello'.

Name	Statement	Branch	Loop	Term	?-Operator	Synchronized
BENTIVOGLIO_TESTING	100,0 %	100,0 %	?	83,3 %	?	?
Ostello	100,0 %	100,0 %	?	83,3 %	?	?

Copertura condizioni e MCDC

Ho una copertura degli statement e branch del 100%. Tuttavia se guardo la copertura delle condizioni nella decisione: `(stanza<0 || stanza>9 || letto <0 || letto>4)`

Test Sessions Coverage Boolean Analyzer Pick Test Cases Correlation Problems Console

Class: RushHour Condition: (((((row < 0 OR row > 5) OR col < 0) OR col > 5) OR dir < 1) OR dir > 4)

(((((row < 0	OR	row > 5)	OR	col < 0)	OR	col > 5)	OR	dir < 1)	OR	dir > 4)	Result	Test Case
				T	T		x		T	x		T	x		T	x		T	x		1	RushHour
				F	F		F		F	F		F	F		F	F		F	F		0	RushHour

Mi accorgo che molte condizioni non sono coperte.

Scrivo un caso di test in cui faccio variare solo una condizione vera tenendo tutte le altre a false (essendo un OR) per ottenere la copertura MCDC. I test sono:

- ```
- test1 -> provo ad occupare un letto con indice > 4 in una stanza che non esiste
- test2 -> provo ad occupare un letto con indice < 0 in una stanza che esiste
- test3 -> provo ad occupare un letto con indice > 4 in una stanza che esiste
```

Ottingo la copertura:

Problems
 Javadoc
 Declaration
 Test Sessions
 Coverage
 Boolean Analyzer

Class: Ostello
 Condition: (((stanza<0 OR stanza>9) OR letto <0) OR letto>4)

| ( | ( | ( | stanza<0 | OR | stanza>9 | ) | OR | letto <0 | ) | OR | letto>4 | ) | Result | Test Cases (Number of Execution)  |
|---|---|---|----------|----|----------|---|----|----------|---|----|---------|---|--------|-----------------------------------|
|   |   |   | F        | T  | T        |   | T  | x        |   | T  | x       |   | 1      | OstelloMCDC:test1 (1) Coverage    |
|   |   |   | F        | F  | F        |   | F  | F        |   | F  | F       |   | 0      | OstelloTest:testLettoOccEOccup    |
|   |   |   | F        | F  | F        |   | F  | F        |   | T  | T       |   | 1      | OstelloMCDC:test3 (1) Coverage    |
|   |   |   | T        | T  | x        |   | T  | x        |   | T  | x       |   | 1      | OstelloStatement:test1 (1), Ostel |
|   |   |   | F        | F  | F        |   | T  | T        |   | T  | x       |   | 1      | OstelloMCDC:test2 (1) Coverage    |

Che mi permette di coprire tutte le condizioni.

Mentre se guardo la copertura delle condizioni nella decisione: `(stanza<0 || stanza>9)` in **public** **boolean** checkin(**int** stanza)

Problems @ Javadoc Declaration Test Sessions Coverage Boolean Analyzer

Class: Ostello Condition: (stanza<0 OR stanza>9)

| ( | stanza<0 | OR | stanza>9 | ) | Result | Test Cases (Number of Executions)                                              |
|---|----------|----|----------|---|--------|--------------------------------------------------------------------------------|
|   | F        | F  | F        |   | 0      | OstelloStatement:test3 (1), OstelloTest:testStanzaOccEOccupo (1), OstelloStat  |
|   | T        | T  | x        |   | 1      | OstelloStatement:test2 (1), OstelloStatement:test2 (1), OstelloStatement:test2 |

Quindi aggiungo il seguente test:

- test4 -> provo ad occupare un letto in una stanza che non esiste (indice >9)

Ed ottengo la seguente copertura:

| Problems @ Javadoc Declaration Test Sessions Coverage Boolean Analyzer |          |                                   |          |   |        |                                                                                    |
|------------------------------------------------------------------------|----------|-----------------------------------|----------|---|--------|------------------------------------------------------------------------------------|
| Class: Ostello                                                         |          | Condition: (stanza<0 OR stanza>9) |          |   |        |                                                                                    |
| (                                                                      | stanza<0 | OR                                | stanza>9 | ) | Result | Test Cases (Number of Executions)                                                  |
|                                                                        | F        | T                                 | T        |   | 1      | OstelloMCDCTest4 (1) Coverage: 25,0                                                |
|                                                                        | F        | F                                 | F        |   | 0      | OstelloStatement:test3 (1), OstelloStatement:test3 (1), OstelloTest:testStanzaOc   |
|                                                                        | T        | T                                 | x        |   | 1      | OstelloStatement:test2 (1), OstelloStatement:test2 (1), OstelloStatement:test2 (1) |

Se guardo la decisione: (stanza<0 || stanza>9) in public boolean libera(int stanza) ho la seguente copertura:

| Problems @ Javadoc Declaration Test Sessions Coverage Boolean Analyzer |          |                                   |          |   |        |                                                                                |
|------------------------------------------------------------------------|----------|-----------------------------------|----------|---|--------|--------------------------------------------------------------------------------|
| Class: Ostello                                                         |          | Condition: (stanza<0 OR stanza>9) |          |   |        |                                                                                |
| (                                                                      | stanza<0 | OR                                | stanza>9 | ) | Result | Test Cases (Number of Executions)                                              |
|                                                                        | F        | F                                 | F        |   | 0      | OstelloStatement:test6 (1), OstelloStatement:test6 (1), OstelloStatement:test5 |
|                                                                        | T        | T                                 | x        |   | 1      | OstelloStatement:test4 (1), OstelloStatement:test4 (1), OstelloStatement:test4 |

Quindi aggiungo il seguente test:

- test5 -> provo a vedere se una stanza che non esiste (indice >9) sia libera

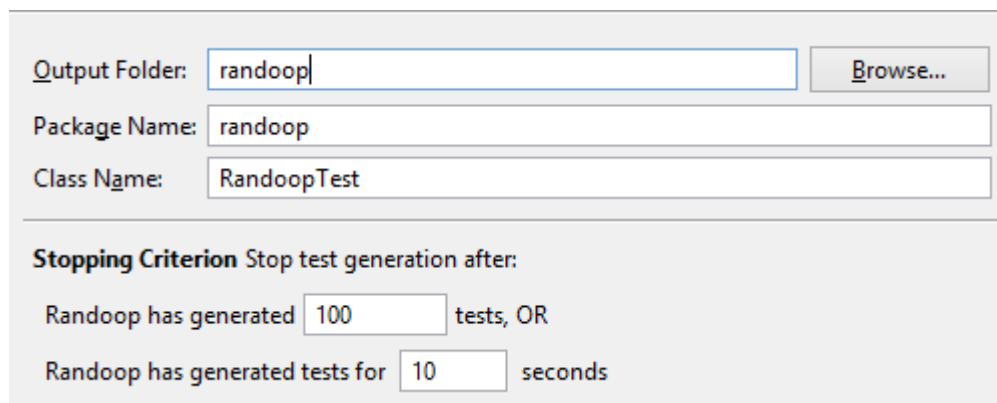
Ed ottengo la seguente copertura:

| Problems @ Javadoc Declaration Test Sessions Coverage Boolean Analyzer |          |                                   |          |   |        |                                                                             |
|------------------------------------------------------------------------|----------|-----------------------------------|----------|---|--------|-----------------------------------------------------------------------------|
| Class: Ostello                                                         |          | Condition: (stanza<0 OR stanza>9) |          |   |        |                                                                             |
| (                                                                      | stanza<0 | OR                                | stanza>9 | ) | Result | Test Cases (Number of Executions)                                           |
|                                                                        | F        | T                                 | T        |   | 1      | OstelloMCDCTest5 (1) Coverage: 25,0                                         |
|                                                                        | F        | F                                 | F        |   | 0      | OstelloStatement:test6 (1), OstelloStatement:test5 (1), OstelloStatement:te |
|                                                                        | T        | T                                 | x        |   | 1      | OstelloStatement:test4 (1), OstelloStatement:test4 (1), OstelloStatement:te |

Che mi permette di coprire tutte le condizioni e anche l'MCDC.

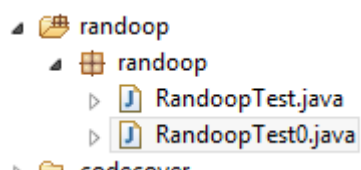
## Randoop

Genero casi di test con randoop e abbasso il numero di test e i secondi per fare prima (altrimenti mi blocca per troppo tempo):



The screenshot shows the Randoop configuration window. It has three input fields: 'Output Folder' with the value 'randoop' and a 'Browse...' button; 'Package Name' with the value 'randoop'; and 'Class Name' with the value 'RandoopTest'. Below these fields is a section titled 'Stopping Criterion' with the text 'Stop test generation after:'. It contains two lines of configuration: 'Randoop has generated 100 tests, OR' and 'Randoop has generated tests for 10 seconds'.

Ho generato i casi di test con randoop:



Che adesso lancio per valutare la copertura:

La copertura è molta bassa – come atteso.

## JML

Creo nuovo progetto e copio la classe. Modifica la visibilità del campo:

```
private /*@spec_public@/ boolean lettiperStanza[][]= new boolean[10][5];
```

Aggiungo i contratti:

Per il metodo `public boolean checkin(int stanza, int letto)` chiedo:

che gli indici siano giusti

```
//@requires !(stanza<0 || stanza>9 || letto <0 || letto>4);
```

Poi garantisco che se il letto è libero allora riesco ad occuparlo

```
//@ensures \result==true <==> lettiperStanza[stanza][letto];
```

Per il metodo **public boolean** checkin(**int** stanza) chiedo:

che gli indici siano giusti

```
//@requires !(stanza<0 || stanza>9);
```

Poi garantisco che se esiste un letto libero allora riesco ad occuparlo.

```
//@ensures \result==true <==> (\exists int i;i>=0 && i<5;lettiperStanza[stanza][i]);
```

Per il metodo **public boolean** libera(**int** stanza)chiedo:

che gli indici siano giusti

```
//@requires !(stanza<0 || stanza>9);
```

Poi garantisco che se tutti i letti sono liberi allora la stanza è libera.

```
//@ensures \result==true <==> (\forall int i;i>=0 && i<5;lettiperStanza[stanza][i]);
```

Poi aggiungo l'invariante:

```
//@public invariant lettiperStanza!=null && lettiperStanza.length==10 &&
lettiperStanza[0].length==5;
```

Quando testo la mia classe, posso semplicemente violare i vari contratti:

Per il metodo **public boolean** checkin(**int** stanza, **int** letto) violo la preconditione:

```
JML precondition is false
 o.checkin(-1,5);
 ^
```

Associated declaration:

```
C:\Users\l.bentivoglio\Desktop\ESAME_TEST\BENTIVOGLIO_JML\src\Main.java:6:
 //@requires !(stanza<0 || stanza>9 || letto <0 || letto>4);
```

E violo una postcondizione modificando il metodo :

```
//@requires !(stanza<0 || stanza>9 || letto <0 || letto>4);
//@ensures \result==true <==> !lettiperStanza[stanza][letto];
public boolean checkin(int stanza, int letto){
 //occupa letto specifico
 if(lettiperStanza[stanza][letto]){
 //lettiperStanza[stanza][letto]=false;
 return true;
 }
 return false;
}
```

```
JML postcondition is false
 public boolean checkin(int stanza, int letto){
 ^
```

Associated declaration:

```
 //@ensures \result==true <==> !lettiperStanza[stanza][letto];
 ^Ad esempio se modifico il metodo in questo modo:
```

Per il metodo **public boolean** checkin(**int** stanza) violo la preconditione:

```
//@requires !(stanza<0 || stanza>9);
```

JML precondition is false  
o.checkin(-1);  
          ^

Associated declaration:  
    //@requires !(stanza<0 || stanza>9);  
    ^

E modifico il metodo:

```
 //@requires !(stanza<0 || stanza>9);
 //@ensures \result==true <==> (\exists int i;i>=0 &&
i<5;lettiperStanza[stanza][i]);
 public boolean checkin(int stanza){
 //occupa il primo letto libero della stanza
 for(int i=0;i<5;i++){
qui--> if(!lettiperStanza[stanza][i]){
 lettiperStanza[stanza][i]=false;
 return true;
 }
 }
 return false;
}
```

E violo la postcondizione.

JML postcondition is false  
    **public boolean** checkin(**int** stanza){  
        ^

Associated declaration:  
    //@ensures \result==true <==> (\exists int i;i>=0 &&  
i<5;lettiperStanza[stanza][i]);  
    ^

Per il metodo **public boolean** libera(**int** stanza) violo la preconditione:

```
//@requires !(stanza<0 || stanza>9);
```

JML precondition is false  
o.libera(-3);  
          ^

Associated declaration:  
    //@requires !(stanza<0 || stanza>9);  
    ^

E modifico il codice in questo modo:

```
//@requires !(stanza<0 || stanza>9);
//@ensures \result==true <==> (\forallall int i;i>=0 &&
i<5;!lettiperStanza[stanza][i]);
public boolean libera(int stanza){
//una stanza è libera se ha tutti i letti liberi
for(int i=0;i<5;i++){
qui→ if(lettiperStanza[stanza][i]){
return false;
}
}
return true;
}
```

E violo la postcondizione:

JML postcondition is false

```
public boolean libera(int stanza){
^
```

Associated declaration::

```
//@ensures \result==true <==> (\forallall int i;i>=0 &&
i<5;!lettiperStanza[stanza][i]);
^
```

Provo a violare l'invariante modificando il costruttore in questo modo:

```
public Ostello(){
for(int i=0;i<10;i++)
for(int j=0;j<5;j++)
lettiperStanza[i][j]=true;
lettiperStanza=null;
}
```

JML assignment of null to a non\_null variable

```
lettiperStanza=null;
^
```

JML invariant is false on leaving method Ostello.Ostello()

```
public Ostello(){
^
```

Associated declaration:

```
//@public invariant lettiperStanza!=null && lettiperStanza.length==10 &&
lettiperStanza[0].length==5;
^
```



## KEY

Creo nuovo progetto come KeyProject. Copio la classe da JML.

Provo a dimostrare:

| Type    | Target            | Contract                 | Proof Reuse | Proof Result | Nodes | Branches |
|---------|-------------------|--------------------------|-------------|--------------|-------|----------|
| Ostello | libera(int)       | JML operation contract 0 | New Proof   | Open         | 105   | 2        |
| Ostello | checkin(int)      | JML operation contract 0 | New Proof   | Open         | 93    | 2        |
| Ostello | checkin(int, int) | JML operation contract 0 | New Proof   | Open         | 485   | 12       |

Provo a cambiare qualcosa ma rimane tutto open.

## NUMSV

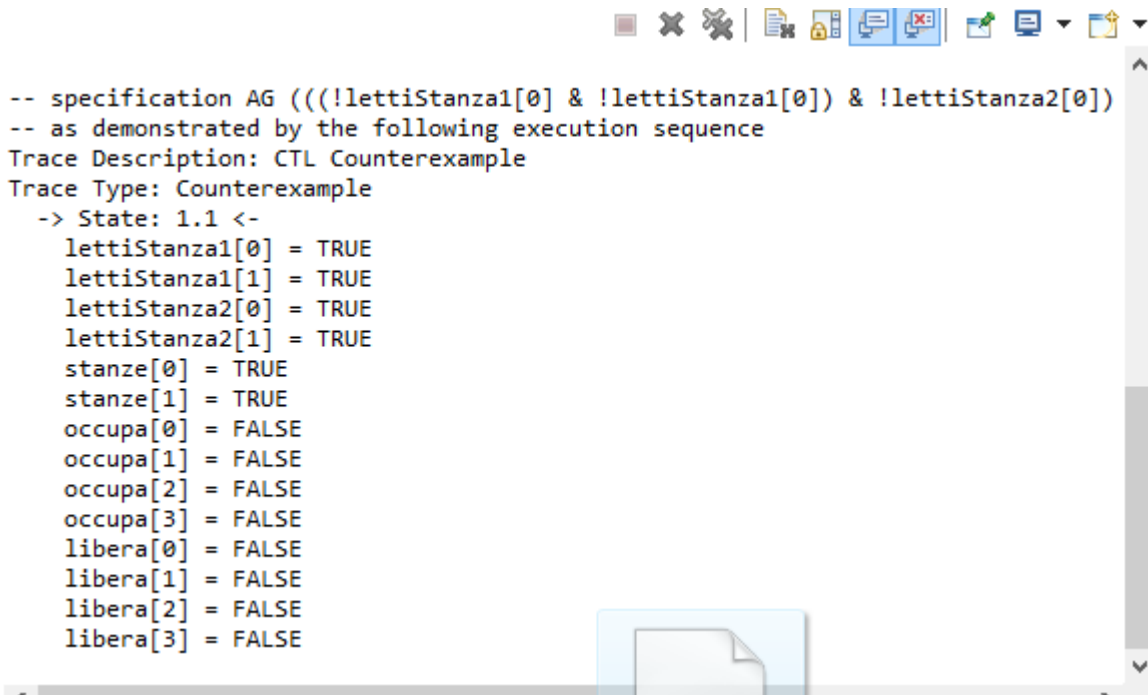
In questo caso ho solo 2 letti per stanze

```
MODULE main
VAR
 lettiStanza1: array 0..1 of boolean;
 lettiStanza2: array 0..1 of boolean;
 stanze: array 0..1 of boolean;
 occupa: array 0..3 of boolean;
 libera: array 0..3 of boolean;
```

Provo le seguenti proprietà:

```
SPEC
--una stanza non può mai essere occupata
 AG (!lettiStanza1[0] & !lettiStanza1[1] & !lettiStanza2[0] & !lettiStanza2[1])CTLSPEC
```

E trovo il seguente comportamento:



```
-- specification AG (((!lettiStanza1[0] & !lettiStanza1[0]) & !lettiStanza2[0])
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
 lettiStanza1[0] = TRUE
 lettiStanza1[1] = TRUE
 lettiStanza2[0] = TRUE
 lettiStanza2[1] = TRUE
 stanze[0] = TRUE
 stanze[1] = TRUE
 occupa[0] = FALSE
 occupa[1] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
 libera[0] = FALSE
 libera[1] = FALSE
 libera[2] = FALSE
 libera[3] = FALSE
```

Verifico

```
-- un letto una volta occupato non può diventare libero
CTLSPEC AG (!lettiStanza1[0] -> !EF(lettiStanza1[0]))
CTLSPEC AG (!lettiStanza1[1] -> !EF(lettiStanza1[1]))
CTLSPEC AG (!lettiStanza2[0] -> !EF(lettiStanza2[0]))
CTLSPEC AG (!lettiStanza2[1] -> !EF(lettiStanza2[1]))
```

Ed ottengo il seguente comportamento:

```
-- specification AG (!lettiStanza1[0] -> !(EF lettiStanza1[0])) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
 lettiStanza1[0] = TRUE
 lettiStanza1[1] = TRUE
 lettiStanza2[0] = TRUE
 lettiStanza2[1] = TRUE
 stanze[0] = TRUE
 stanze[1] = TRUE
 occupa[0] = FALSE
 occupa[1] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
 libera[0] = FALSE
 libera[1] = FALSE
 libera[2] = FALSE
 libera[3] = FALSE
-> State: 1.2 <-
 occupa[0] = TRUE
 occupa[2] = TRUE
 occupa[3] = TRUE
-> State: 1.3 <-
 lettiStanza1[0] = FALSE
 lettiStanza2[0] = FALSE
 lettiStanza2[1] = FALSE
 occupa[0] = FALSE
```

```

 occupa[2] = FALSE
 occupa[3] = FALSE
-> State: 1.4 <-
 occupa[2] = TRUE
 occupa[3] = TRUE
 libera[0] = TRUE
-> State: 1.5 <-
 lettiStanza1[0] = TRUE
 occupa[2] = FALSE
 occupa[3] = FALSE
 libera[0] = FALSE
-- specification AG (!lettiStanza1[1] -> !(EF lettiStanza1[1])) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 2.1 <-
 lettiStanza1[0] = TRUE
 lettiStanza1[1] = TRUE
 lettiStanza2[0] = TRUE
 lettiStanza2[1] = TRUE
 stanze[0] = TRUE
 stanze[1] = TRUE
 occupa[0] = FALSE
 occupa[1] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
 libera[0] = FALSE
 libera[1] = FALSE
 libera[2] = FALSE
 libera[3] = FALSE
-> State: 2.2 <-
 occupa[1] = TRUE
 occupa[3] = TRUE
-> State: 2.3 <-
 lettiStanza1[1] = FALSE
 lettiStanza2[1] = FALSE
 occupa[1] = FALSE
 occupa[3] = FALSE
-> State: 2.4 <-
 occupa[0] = TRUE
 occupa[2] = TRUE
 occupa[3] = TRUE
 libera[1] = TRUE
-> State: 2.5 <-
 lettiStanza1[0] = FALSE
 lettiStanza1[1] = TRUE
 lettiStanza2[0] = FALSE
 occupa[0] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
 libera[1] = FALSE
-- specification AG (!lettiStanza2[0] -> !(EF lettiStanza2[0])) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 3.1 <-
 lettiStanza1[0] = TRUE
 lettiStanza1[1] = TRUE
 lettiStanza2[0] = TRUE
 lettiStanza2[1] = TRUE
 stanze[0] = TRUE

```

```

stanze[1] = TRUE
occupa[0] = FALSE
occupa[1] = FALSE
occupa[2] = FALSE
occupa[3] = FALSE
libera[0] = FALSE
libera[1] = FALSE
libera[2] = FALSE
libera[3] = FALSE
-> State: 3.2 <-
 occupa[2] = TRUE
 occupa[3] = TRUE
-> State: 3.3 <-
 lettiStanza2[0] = FALSE
 lettiStanza2[1] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
-> State: 3.4 <-
 occupa[1] = TRUE
 occupa[3] = TRUE
 libera[2] = TRUE
-> State: 3.5 <-
 lettiStanza1[1] = FALSE
 lettiStanza2[0] = TRUE
 occupa[1] = FALSE
 occupa[3] = FALSE
 libera[2] = FALSE
-- specification AG (!lettiStanza2[1] -> !(EF lettiStanza2[1])) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 4.1 <-
 lettiStanza1[0] = TRUE
 lettiStanza1[1] = TRUE
 lettiStanza2[0] = TRUE
 lettiStanza2[1] = TRUE
 stanze[0] = TRUE
 stanze[1] = TRUE
 occupa[0] = FALSE
 occupa[1] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
 libera[0] = FALSE
 libera[1] = FALSE
 libera[2] = FALSE
 libera[3] = FALSE
-> State: 4.2 <-
 occupa[2] = TRUE
 occupa[3] = TRUE
-> State: 4.3 <-
 lettiStanza2[0] = FALSE
 lettiStanza2[1] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
-> State: 4.4 <-
 occupa[1] = TRUE
 occupa[2] = TRUE
 libera[3] = TRUE
-> State: 4.5 <-
 lettiStanza1[1] = FALSE
 lettiStanza2[1] = TRUE

```

```

 occupa[1] = FALSE
 occupa[2] = FALSE
 libera[3] = FALSE
-- specification AG (((!lettiStanza1[0] & !lettiStanza1[1]) & !lettiStanza2[0]) &
!lettiStanza2[1]) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 5.1 <-
 lettiStanza1[0] = TRUE
 lettiStanza1[1] = TRUE
 lettiStanza2[0] = TRUE
 lettiStanza2[1] = TRUE
 stanze[0] = TRUE
 stanze[1] = TRUE
 occupa[0] = FALSE
 occupa[1] = FALSE
 occupa[2] = FALSE
 occupa[3] = FALSE
 libera[0] = FALSE
 libera[1] = FALSE
 libera[2] = FALSE
 libera[3] = FALSE

```

Infine verifico:

```

--un letto non può mai venire occupato se è già occupato
CTLSPEC AG (!lettiStanza1[0] & occupa[0]-> !EF(!lettiStanza1[0] & occupa[0]))
CTLSPEC AG (!lettiStanza1[0] & occupa[1]-> !EF(!lettiStanza1[0] & occupa[1]))
CTLSPEC AG (!lettiStanza2[0] & occupa[2]-> !EF(!lettiStanza2[0] & occupa[2]))
CTLSPEC AG (!lettiStanza2[0] & occupa[3]-> !EF(!lettiStanza2[0] & occupa[3]))

```

## FSM

Scrivo un modello FSM per la versione semplificata.

Metto le azioni per ogni letto

```

 @Action
 public void liberaletto1Stanza1(){
 lettiperStanza[0][0]=true;
 }

 @Action
 public void occupaletto1Stanza1(){
 lettiperStanza[2][0]=false;
 }

```

Con le guardie

```

 public boolean liberaletto1Stanza1Guard(){
 if(lettiperStanza[0][0]) return false;
 return true;
 }

 public boolean occupaletto1Stanza1Guard(){
 if(!lettiperStanza[0][0]) return false;
 return true;
 }

```

Metto anche lo state che mi dice quali letti sono liberi.

@Override

```
public Object getState() {

 return " \nletto1Stanza1 occ: " + lettiperStanza[0][0] + " \nletto2Stanza1
occ: " + lettiperStanza[0][1] +
 " \nletto3Stanza1 occ: " + lettiperStanza[0][2] +
 " \nletto1Stanza2 occ: " + lettiperStanza[1][0] + "
\nletto2Stanza2 occ: " + lettiperStanza[1][1] +
 " \nletto3Stanza2 occ: " + lettiperStanza[1][2] +
 " \nletto1Stanza3 occ: " + lettiperStanza[2][0] + "
\nletto2Stanza3 occ: " + lettiperStanza[2][1] +
 " \nletto3Stanza3 occ: " + lettiperStanza[2][2];
}
```

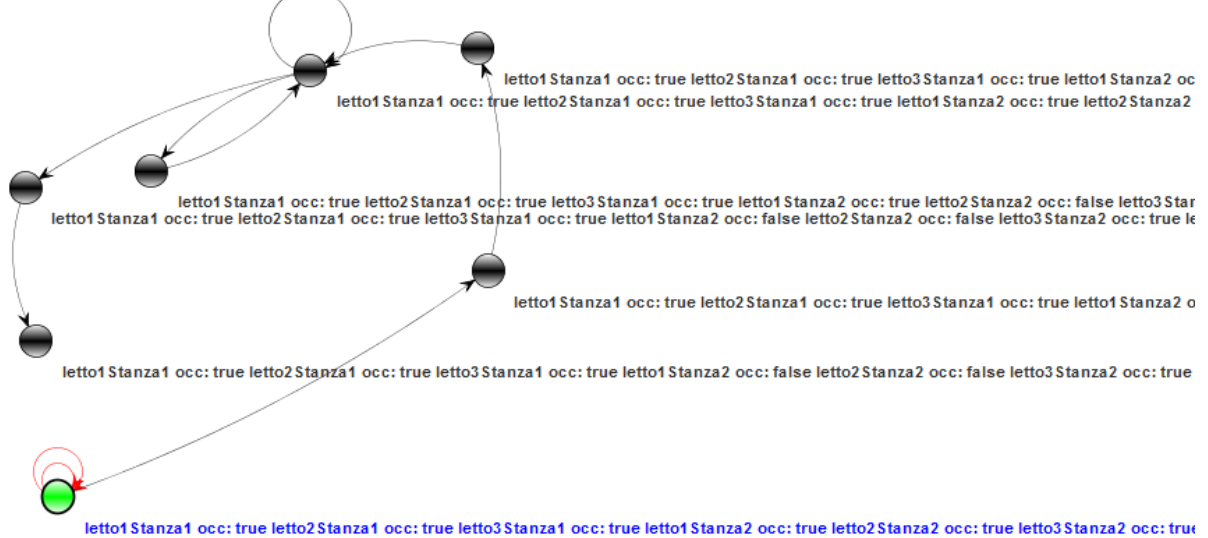
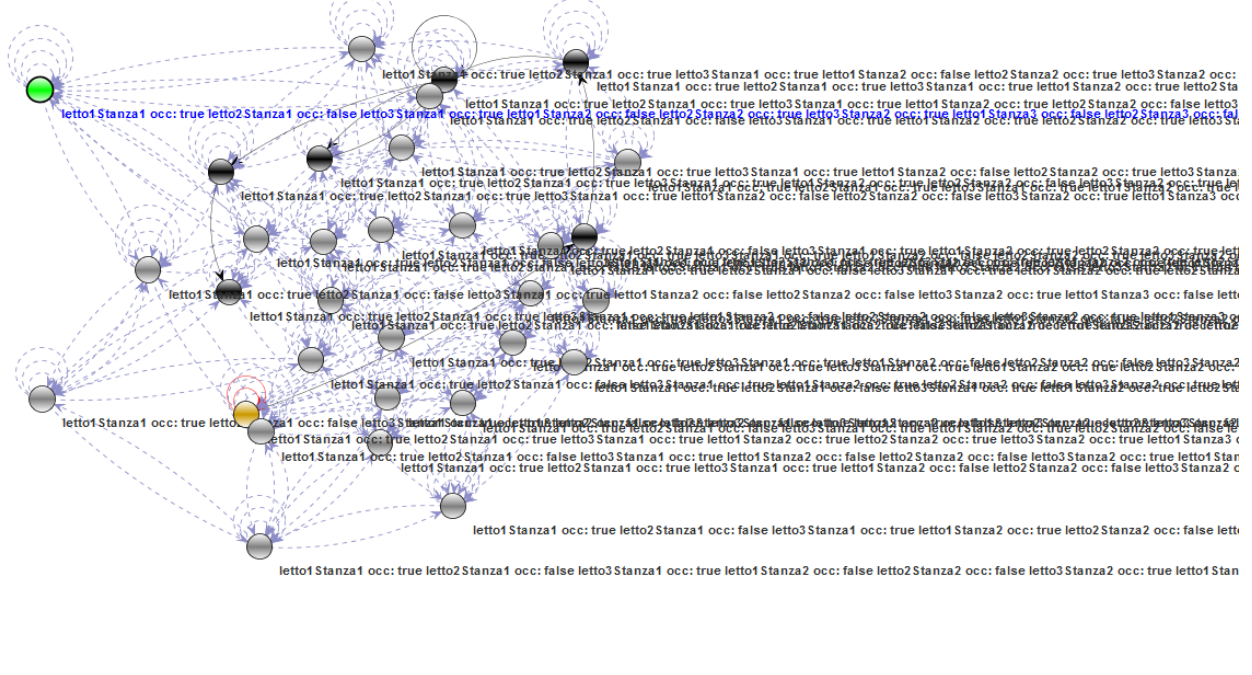
Creo online testing:

.....

```
done (
letto1Stanza1 occ: true
letto2Stanza1 occ: true
letto3Stanza1 occ: true
letto1Stanza2 occ: true
letto2Stanza2 occ: true
letto3Stanza2 occ: true
letto1Stanza3 occ: true
letto2Stanza3 occ: true
letto3Stanza3 occ: true, occupaletto2Stanza2,
letto1Stanza1 occ: true
letto2Stanza1 occ: true
letto3Stanza1 occ: true
letto1Stanza2 occ: true
letto2Stanza2 occ: false
letto3Stanza2 occ: true
letto1Stanza3 occ: true
letto2Stanza3 occ: true
letto3Stanza3 occ: true)
transition coverage was 27/288
```

mi accorgo che il numero di test è troppo piccolo quindi aumento il numero di test e dopo vari tentativi:

```
done (
letto1Stanza1 occ: true
letto2Stanza1 occ: true
letto3Stanza1 occ: true
letto1Stanza2 occ: true
letto2Stanza2 occ: true
letto3Stanza2 occ: true
letto1Stanza3 occ: true
letto2Stanza3 occ: true
letto3Stanza3 occ: true, occupaletto1Stanza3,
letto1Stanza1 occ: true
letto2Stanza1 occ: true
letto3Stanza1 occ: true
letto1Stanza2 occ: true
letto2Stanza2 occ: true
letto3Stanza2 occ: true
letto1Stanza3 occ: false
letto2Stanza3 occ: true
letto3Stanza3 occ: true)
```



Scrivo il modello:

## Model Ostello

## Types :

```
EnumerativeType STATO { OCCUPATA LIBERA };
```

end

### Parameters:

```
Boolean stanza1;
```

```
Boolean stanza2;
```

```
Boolean stanza3;
```

```
Boolean letto1S1;
```

```
Boolean letto2S1;
```

```
Boolean letto3S1;
```

```
Boolean letto1S2;
```

```
Boolean letto2S2;
```

```
Boolean letto3S2;
```

```
Boolean letto1S3;
```

```
Boolean letto2S3;
```

```
Boolean letto3S3;
```

end

### Constraints:

```
letto1S1==false and letto2S1==false and letto3S1==false <=> stanza1==false
```

```
letto1S2==false and letto2S2==false and letto3S2==false <=> stanza2==false
```

```
letto1S3==false and letto2S3==false and letto3S3==false <=> stanza3==false
```

end

Genero il pairwise e ottengo 10 casi di test:

| Test | stanza1 | stan... | sta... | lett... | lett... | let... | letto1S2 | letto2S2 | letto3S2 | letto1S3 | letto2S3 | letto3S3 |
|------|---------|---------|--------|---------|---------|--------|----------|----------|----------|----------|----------|----------|
| 1    | true    | true    | true   | true    | false   | true   | false    | false    | true     | false    | true     | false    |
| 2    | true    | false   | false  | true    | true    | false  | false    | false    | false    | false    | false    | false    |
| 3    | false   | true    | false  | false   | false   | false  | true     | true     | false    | false    | false    | false    |
| 4    | false   | false   | true   | false   | false   | false  | false    | false    | false    | false    | false    | true     |
| 5    | true    | true    | true   | false   | true    | true   | true     | true     | true     | true     | true     | true     |
| 6    | true    | false   | false  | true    | false   | true   | false    | false    | false    | false    | false    | false    |
| 7    | true    | true    | false  | true    | true    | false  | true     | false    | true     | false    | false    | false    |
| 8    | true    | true    | true   | true    | false   | false  | false    | true     | false    | true     | true     | true     |
| 9    | false   | true    | true   | false   | false   | false  | false    | false    | true     | true     | false    | true     |
| 10   | false   | false   | true   | false   | false   | false  | false    | false    | false    | true     | true     | false    |



