Санкт-Петербургский Государственный Политехнический Университет
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Самостоятельная работа №1

по дисциплине «Распределенные алгоритмы»

Выполнила
студент гр. 3530202/90202 Alina Potapova Потапова А.М.

Преподаватель Шошмина И. В.

2022 г

Содержание

ПОСТАНОВКА ЗАДАЧИ 3

ВВЕДЕНИЕ 4

ОПИСАНИЕ АЛГОРИТМА 5

МОДЕЛЬ АЛГОРИТМА НА ЯЗЫКЕ PROMELA 6

СИМУЛЯЦИЯ 8

ВЕРИФИКАЦИЯ И КОНТРПРИМЕР 11

ЗАКЛЮЧЕНИЕ 15

СПИСОК ЛИТЕРАТУРЫ 16

Постановка задачи

1. Установить SPIN
2. Смоделировать один из алгоритмов (в качестве рассматриваемого алгоритма я выбрала алгоритм 2.14 из Ben-Ari.)
3. Провести его симуляцию

Введение

SPIN — утилита для верификации корректности распределенных программных моделей. Служит для автоматизированной проверки моделей.

Promela — абстрактный язык спецификации алгоритмов. Абстрагирование направлено на то, чтобы с помощью минимальных выразительных средств строить такие абстрактные модели реальных параллельных и распределенных систем, которые легко представляются формальной моделью с конечным числом состояний.

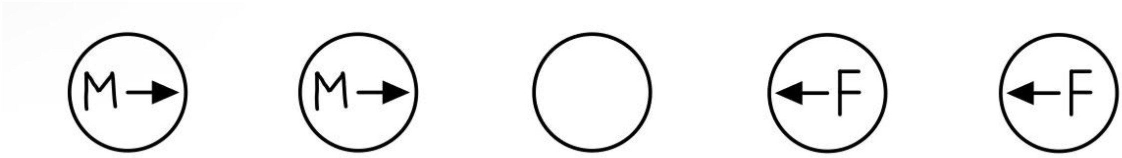
В данной самостоятельной работе основной задачей стоит реализация алгоритма на языке Promela и знакомство с режимом симуляции и верификации утилиты SPIN. Установка SPIN производилась при помощи homebrew (менеджер пакетов с открытым исходным кодом): brew install spin.

Описание алгоритма

Алгоритм 2.14: "Головоломка с лягушками"

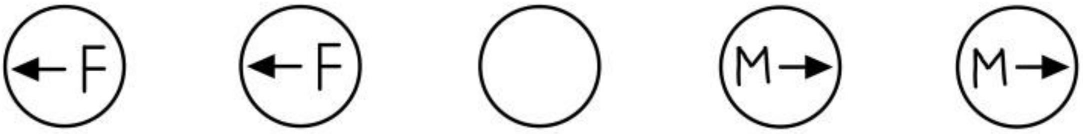
<u>Дано:</u>
Пять камней
2 лягушки-самца справа смотрят влево
2 лягушки-самки слева смотрят направо

Для наглядности можем изобразить следующим образом:



Лягушка может двигаться только в направлении своего взгляда, при условии, что перед ней пустой камень, если же нет, то она перепрыгивает препятствие и становится на следующий камень, если он существует и он пуст.

<u>Вопрос:</u>
Существует ли последовательность ходов, которая поменяет местами самцов и самок лягушек?

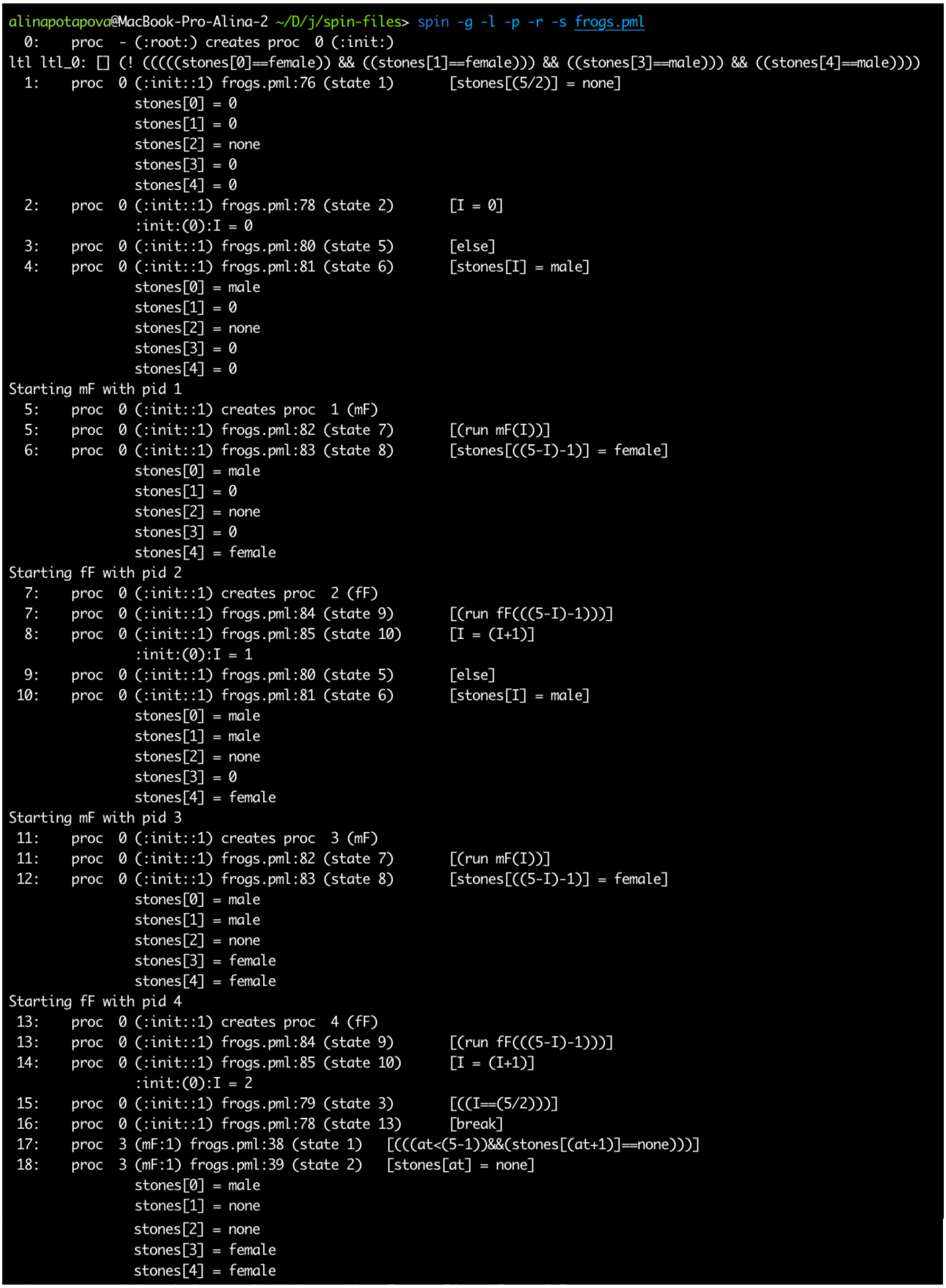


Модель алгоритма на языке Promela

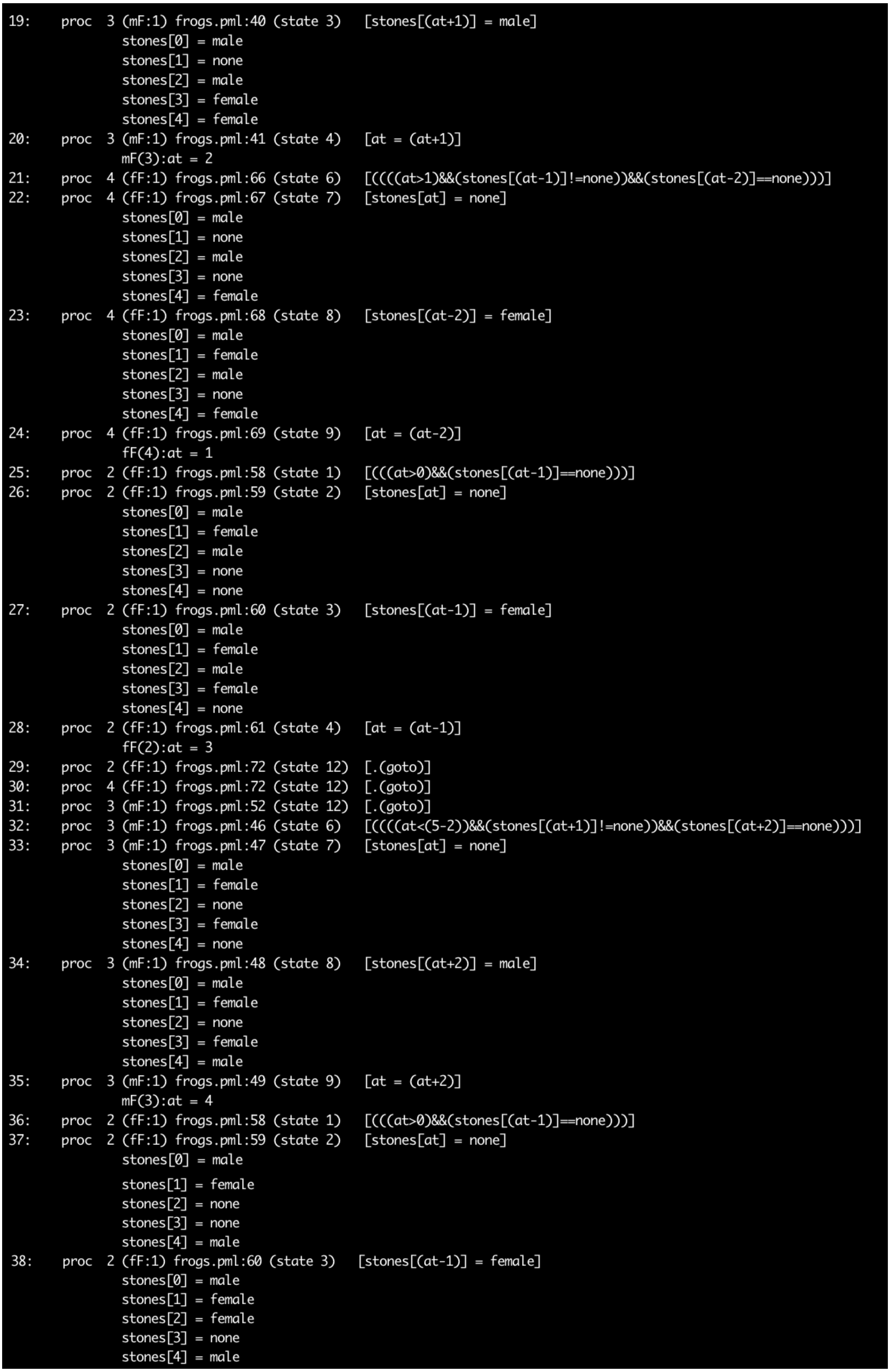
```
#define STONES 5

//для верификации
#define success (\(stones[0]==female) && \(stones[1]==female) && \(stones[3]==male) && \(stones[4]==male)\)

//для верификации
ltl { []!success }

mtype = { none, male, female }
mtype stones[STONES];


proctype mF(byte at) {
end:do
        :: atomic {
                        (at < STONES-1) &&
                        (stones[at+1] == none) ->
                        stones[at] = none;
                        stones[at+1] = male;
                        at = at + 1;
                }
        :: atomic {
                        (at < STONES-2) &&
                        (stones[at+1] != none) &&
                        (stones[at+2] == none) ->
                        stones[at] = none;
                        stones[at+2] = male;
                        at = at + 2;
                }
        od
}

proctype fF(byte at) {
end:do
        :: atomic {
                        (at > 0) &&
                        (stones[at-1] == none) ->
                        stones[at] = none;
                        stones[at-1] = female;
                        at = at - 1;
                }
        :: atomic {
                        (at > 1) &&
                        (stones[at-1] != none) &&
            (stones[at-2] == none) ->
            stones[at] = none;
            stones[at-2] = female;
            at = at - 2;
        }
    od
}

init {
    atomic {
        stones[STONES/2] = none;
        byte I = 0;
        do
        :: I == STONES/2 -> break;
        :: else ->
            stones[I] = male;
            run mF(I);
            stones[STONES-I-1] = female;
            run fF(STONES-I-1);
            I++
        od
    }
}
```

Симуляция

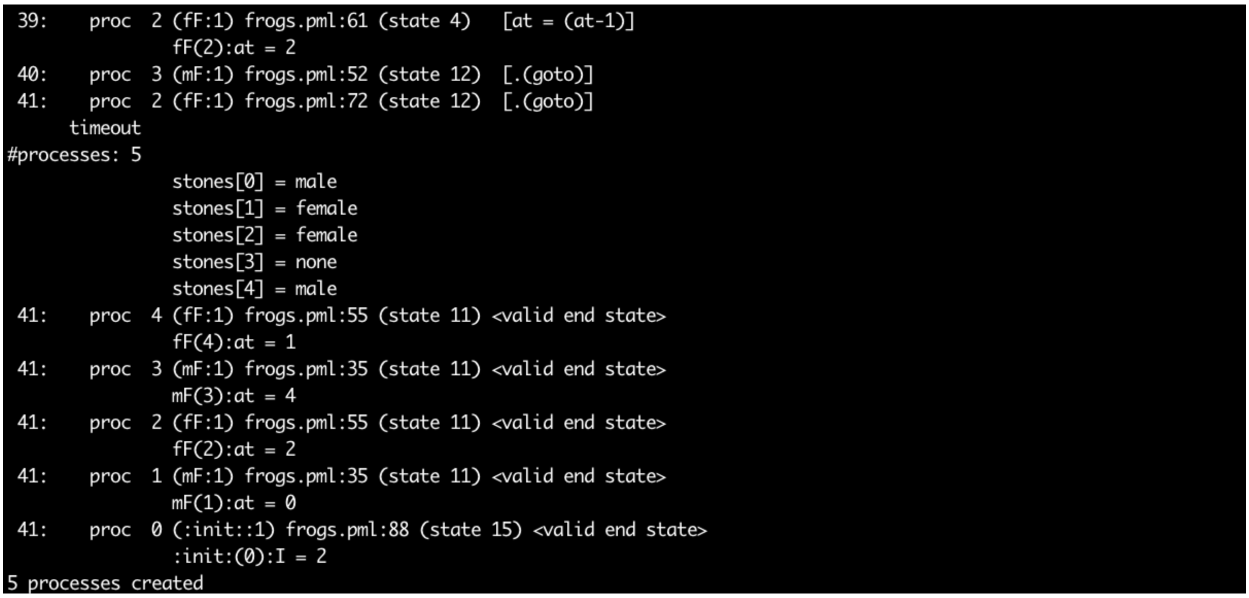
Произведем простой запуск модели:



```
alinapotapova@MacBook-Pro-Alina-2 ~/D/j/spin-files> spin -g -l -p -r -s frogs.pml
  0:    proc  - (:root:) creates proc  0 (:init:)
ltl ltl_0: □ (! (((((stones[0]==female)) && ((stones[1]==female))) && ((stones[3]==male))) && ((stones[4]==male))))
  1:    proc  0 (:init::1) frogs.pml:76 (state 1)       [stones[(5/2)] = none]
                stones[0] = 0
                stones[1] = 0
                stones[2] = none
                stones[3] = 0
                stones[4] = 0
  2:    proc  0 (:init::1) frogs.pml:78 (state 2)       [I = 0]
                :init:(0):I = 0
  3:    proc  0 (:init::1) frogs.pml:80 (state 5)       [else]
  4:    proc  0 (:init::1) frogs.pml:81 (state 6)       [stones[I] = male]
                stones[0] = male
                stones[1] = 0
                stones[2] = none
                stones[3] = 0
                stones[4] = 0
Starting mF with pid 1
  5:    proc  0 (:init::1) creates proc  1 (mF)
  5:    proc  0 (:init::1) frogs.pml:82 (state 7)       [(run mF(I))]
  6:    proc  0 (:init::1) frogs.pml:83 (state 8)       [stones[((5-I)-1)] = female]
                stones[0] = male
                stones[1] = 0
                stones[2] = none
                stones[3] = 0
                stones[4] = female
Starting fF with pid 2
  7:    proc  0 (:init::1) creates proc  2 (fF)
  7:    proc  0 (:init::1) frogs.pml:84 (state 9)       [(run fF(((5-I)-1)))]
  8:    proc  0 (:init::1) frogs.pml:85 (state 10)      [I = (I+1)]
                :init:(0):I = 1
  9:    proc  0 (:init::1) frogs.pml:80 (state 5)       [else]
 10:    proc  0 (:init::1) frogs.pml:81 (state 6)       [stones[I] = male]
                stones[0] = male
                stones[1] = male
                stones[2] = none
                stones[3] = 0
                stones[4] = female
Starting mF with pid 3
 11:    proc  0 (:init::1) creates proc  3 (mF)
 11:    proc  0 (:init::1) frogs.pml:82 (state 7)       [(run mF(I))]
 12:    proc  0 (:init::1) frogs.pml:83 (state 8)       [stones[((5-I)-1)] = female]
                stones[0] = male
                stones[1] = male
                stones[2] = none
                stones[3] = female
                stones[4] = female
Starting fF with pid 4
 13:    proc  0 (:init::1) creates proc  4 (fF)
 13:    proc  0 (:init::1) frogs.pml:84 (state 9)       [(run fF(((5-I)-1)))]
 14:    proc  0 (:init::1) frogs.pml:85 (state 10)      [I = (I+1)]
                :init:(0):I = 2
 15:    proc  0 (:init::1) frogs.pml:79 (state 3)       [((I==(5/2)))]
 16:    proc  0 (:init::1) frogs.pml:78 (state 13)      [break]
 17:    proc  3 (mF:1) frogs.pml:38 (state 1)   [(((at<(5-1))&&(stones[(at+1)]==none)))]
 18:    proc  3 (mF:1) frogs.pml:39 (state 2)   [stones[at] = none]
                stones[0] = male
                stones[1] = none
                stones[2] = none
                stones[3] = female
                stones[4] = female
```

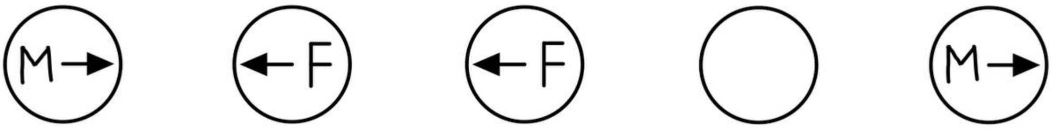


```
19:     proc  3 (mF:1) frogs.pml:40 (state 3)   [stones[(at+1)] = male]
                stones[0] = male
                stones[1] = none
                stones[2] = male
                stones[3] = female
                stones[4] = female
20:     proc  3 (mF:1) frogs.pml:41 (state 4)   [at = (at+1)]
                mF(3):at = 2
21:     proc  4 (fF:1) frogs.pml:66 (state 6)   [((((at>1)&&(stones[(at-1)]!=none))&&(stones[(at-2)]==none)))]
22:     proc  4 (fF:1) frogs.pml:67 (state 7)   [stones[at] = none]
                stones[0] = male
                stones[1] = none
                stones[2] = male
                stones[3] = none
                stones[4] = female
23:     proc  4 (fF:1) frogs.pml:68 (state 8)   [stones[(at-2)] = female]
                stones[0] = male
                stones[1] = female
                stones[2] = male
                stones[3] = none
                stones[4] = female
24:     proc  4 (fF:1) frogs.pml:69 (state 9)   [at = (at-2)]
                fF(4):at = 1
25:     proc  2 (fF:1) frogs.pml:58 (state 1)   [(((at>0)&&(stones[(at-1)]==none)))]
26:     proc  2 (fF:1) frogs.pml:59 (state 2)   [stones[at] = none]
                stones[0] = male
                stones[1] = female
                stones[2] = male
                stones[3] = none
                stones[4] = none
27:     proc  2 (fF:1) frogs.pml:60 (state 3)   [stones[(at-1)] = female]
                stones[0] = male
                stones[1] = female
                stones[2] = male
                stones[3] = female
                stones[4] = none
28:     proc  2 (fF:1) frogs.pml:61 (state 4)   [at = (at-1)]
                fF(2):at = 3
29:     proc  2 (fF:1) frogs.pml:72 (state 12)  [.(goto)]
30:     proc  4 (fF:1) frogs.pml:72 (state 12)  [.(goto)]
31:     proc  3 (mF:1) frogs.pml:52 (state 12)  [.(goto)]
32:     proc  3 (mF:1) frogs.pml:46 (state 6)   [((((at<(5-2))&&(stones[(at+1)]!=none))&&(stones[(at+2)]==none)))]
33:     proc  3 (mF:1) frogs.pml:47 (state 7)   [stones[at] = none]
                stones[0] = male
                stones[1] = female
                stones[2] = none
                stones[3] = female
                stones[4] = none
34:     proc  3 (mF:1) frogs.pml:48 (state 8)   [stones[(at+2)] = male]
                stones[0] = male
                stones[1] = female
                stones[2] = none
                stones[3] = female
                stones[4] = male
35:     proc  3 (mF:1) frogs.pml:49 (state 9)   [at = (at+2)]
                mF(3):at = 4
36:     proc  2 (fF:1) frogs.pml:58 (state 1)   [(((at>0)&&(stones[(at-1)]==none)))]
37:     proc  2 (fF:1) frogs.pml:59 (state 2)   [stones[at] = none]
                stones[0] = male
                stones[1] = female
                stones[2] = none
                stones[3] = none
                stones[4] = male
38:     proc  2 (fF:1) frogs.pml:60 (state 3)   [stones[(at-1)] = female]
                stones[0] = male
                stones[1] = female
                stones[2] = female
                stones[3] = none
                stones[4] = male
```



```
 39:    proc  2 (fF:1) frogs.pml:61 (state 4)   [at = (at-1)]
                fF(2):at = 2
 40:    proc  3 (mF:1) frogs.pml:52 (state 12)  [.(goto)]
 41:    proc  2 (fF:1) frogs.pml:72 (state 12)  [.(goto)]
      timeout
#processes: 5
                stones[0] = male
                stones[1] = female
                stones[2] = female
                stones[3] = none
                stones[4] = male
 41:    proc  4 (fF:1) frogs.pml:55 (state 11) <valid end state>
                fF(4):at = 1
 41:    proc  3 (mF:1) frogs.pml:35 (state 11) <valid end state>
                mF(3):at = 4
 41:    proc  2 (fF:1) frogs.pml:55 (state 11) <valid end state>
                fF(2):at = 2
 41:    proc  1 (mF:1) frogs.pml:35 (state 11) <valid end state>
                mF(1):at = 0
 41:    proc  0 (:init::1) frogs.pml:88 (state 15) <valid end state>
                :init:(0):I = 2
5 processes created
```

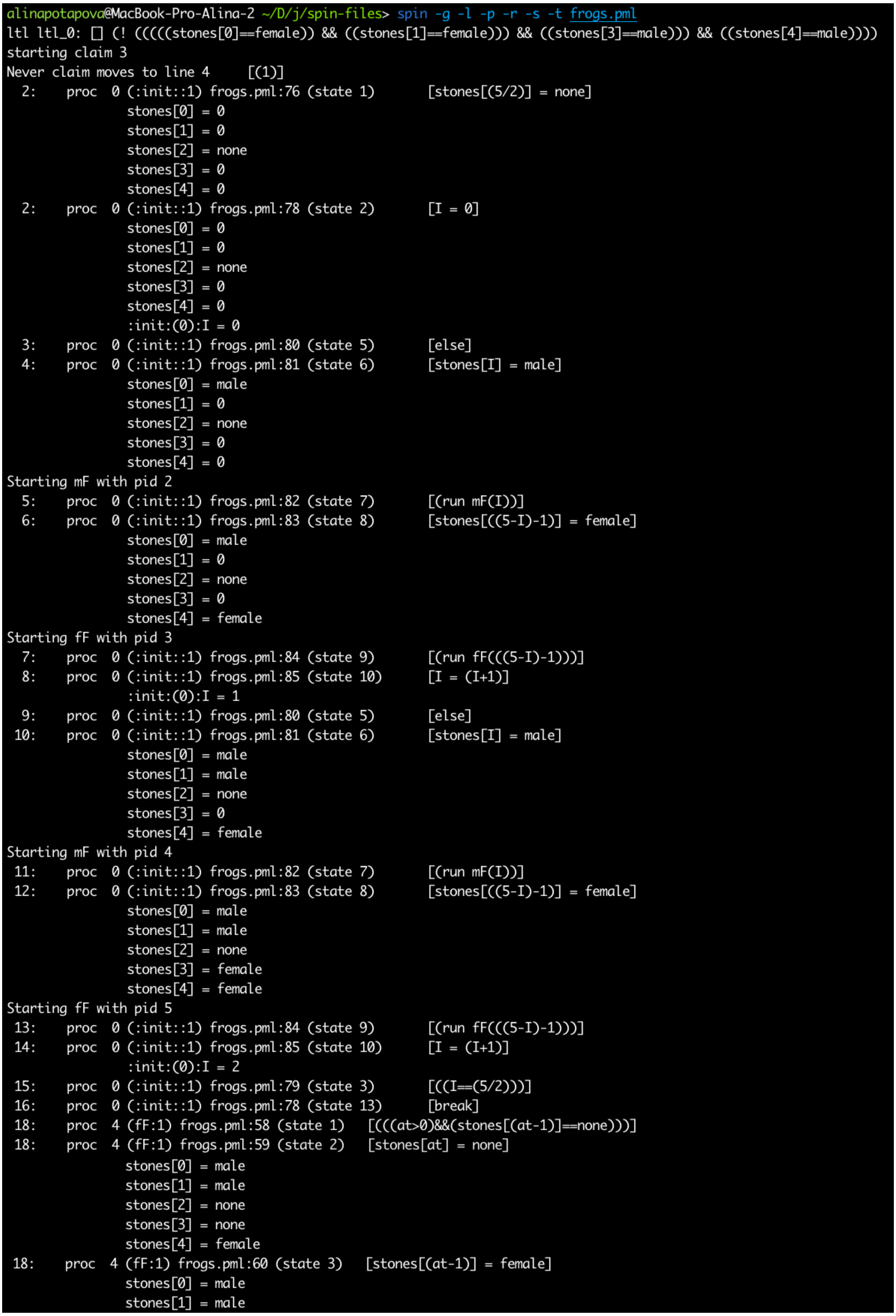
В результате получили следующее расположение (не совпало с желаемым, для ответа на вопрос):



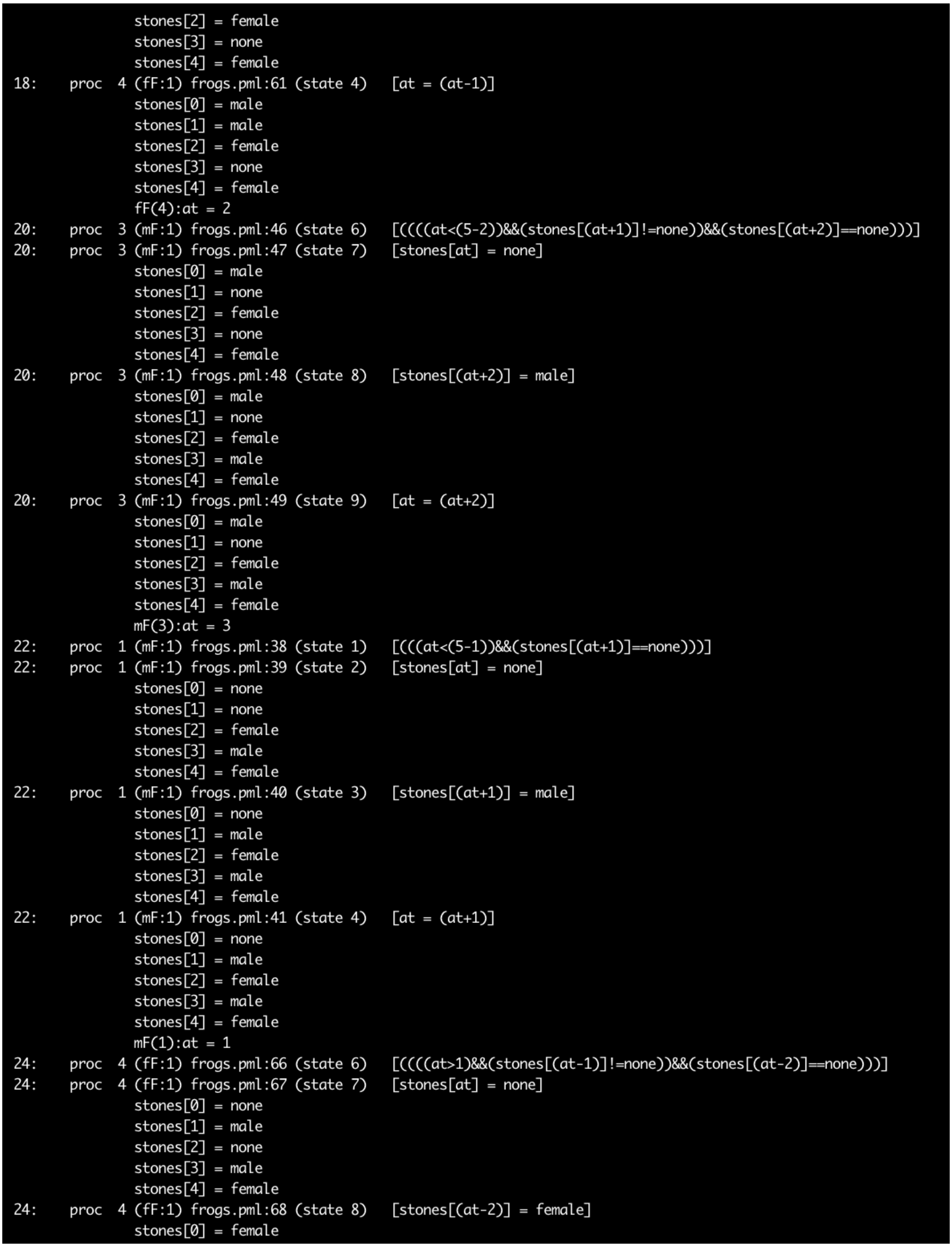
Теперь попробуем ответить на поставленный вопрос. Для этого воспользуемся режимом верификации, добавив в программу желаемое расположение лягушек в переменную success и LTL-формулу, которая будет отрицать это утверждение.

```
#define success (\(stones[0]==female) && \(stones[1]==female) && \(stones[3]==male)
&& \(stones[4]==male)\)

ltl { []!success }
```

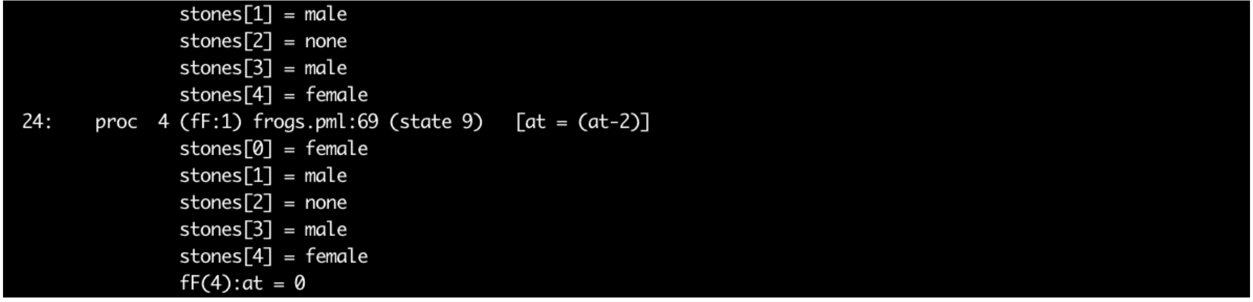
Верификация и контрпример



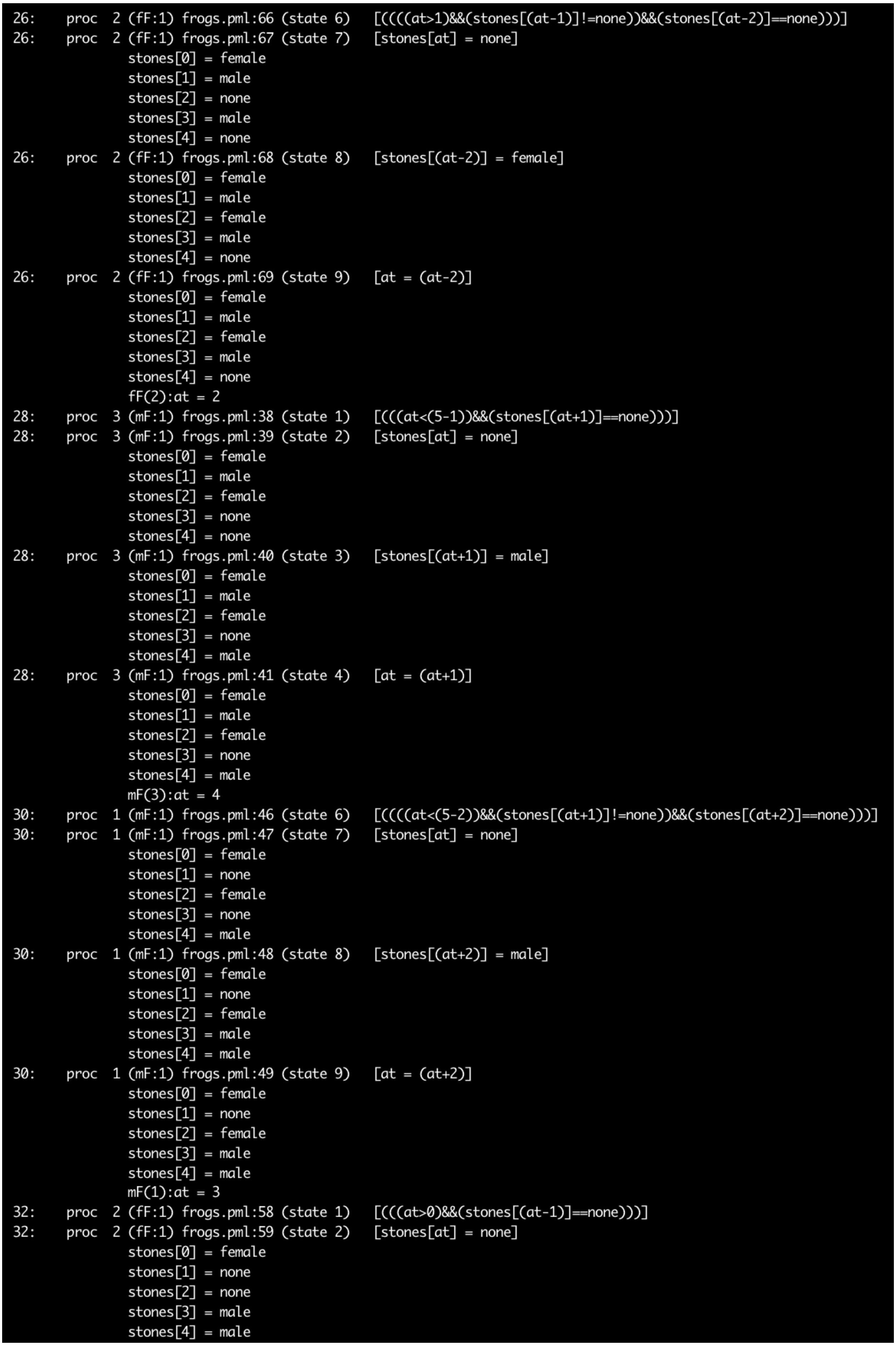
```
alinapotapova@MacBook-Pro-Alina-2 ~/D/j/spin-files> spin -g -l -p -r -s -t frogs.pml
ltl ltl_0: [] (! (((((stones[0]==female)) && ((stones[1]==female))) && ((stones[3]==male))) && ((stones[4]==male))))
starting claim 3
Never claim moves to line 4     [(1)]
  2:    proc  0 (:init::1) frogs.pml:76 (state 1)       [stones[(5/2)] = none]
                stones[0] = 0
                stones[1] = 0
                stones[2] = none
                stones[3] = 0
                stones[4] = 0
  2:    proc  0 (:init::1) frogs.pml:78 (state 2)       [I = 0]
                stones[0] = 0
                stones[1] = 0
                stones[2] = none
                stones[3] = 0
                stones[4] = 0
                :init:(0):I = 0
  3:    proc  0 (:init::1) frogs.pml:80 (state 5)       [else]
  4:    proc  0 (:init::1) frogs.pml:81 (state 6)       [stones[I] = male]
                stones[0] = male
                stones[1] = 0
                stones[2] = none
                stones[3] = 0
                stones[4] = 0
Starting mF with pid 2
  5:    proc  0 (:init::1) frogs.pml:82 (state 7)       [(run mF(I))]
  6:    proc  0 (:init::1) frogs.pml:83 (state 8)       [stones[((5-I)-1)] = female]
                stones[0] = male
                stones[1] = 0
                stones[2] = none
                stones[3] = 0
                stones[4] = female
Starting fF with pid 3
  7:    proc  0 (:init::1) frogs.pml:84 (state 9)       [(run fF(((5-I)-1)))]
  8:    proc  0 (:init::1) frogs.pml:85 (state 10)      [I = (I+1)]
                :init:(0):I = 1
  9:    proc  0 (:init::1) frogs.pml:80 (state 5)       [else]
 10:    proc  0 (:init::1) frogs.pml:81 (state 6)       [stones[I] = male]
                stones[0] = male
                stones[1] = male
                stones[2] = none
                stones[3] = 0
                stones[4] = female
Starting mF with pid 4
 11:    proc  0 (:init::1) frogs.pml:82 (state 7)       [(run mF(I))]
 12:    proc  0 (:init::1) frogs.pml:83 (state 8)       [stones[((5-I)-1)] = female]
                stones[0] = male
                stones[1] = male
                stones[2] = none
                stones[3] = female
                stones[4] = female
Starting fF with pid 5
 13:    proc  0 (:init::1) frogs.pml:84 (state 9)       [(run fF(((5-I)-1)))]
 14:    proc  0 (:init::1) frogs.pml:85 (state 10)      [I = (I+1)]
                :init:(0):I = 2
 15:    proc  0 (:init::1) frogs.pml:79 (state 3)       [((I==(5/2)))]
 16:    proc  0 (:init::1) frogs.pml:78 (state 13)      [break]
 18:    proc  4 (fF:1) frogs.pml:58 (state 1)   [(((at>0)&&(stones[(at-1)]==none)))]
 18:    proc  4 (fF:1) frogs.pml:59 (state 2)   [stones[at] = none]
                stones[0] = male
                stones[1] = male
                stones[2] = none
                stones[3] = none
                stones[4] = female
 18:    proc  4 (fF:1) frogs.pml:60 (state 3)   [stones[(at-1)] = female]
                stones[0] = male
                stones[1] = male
```



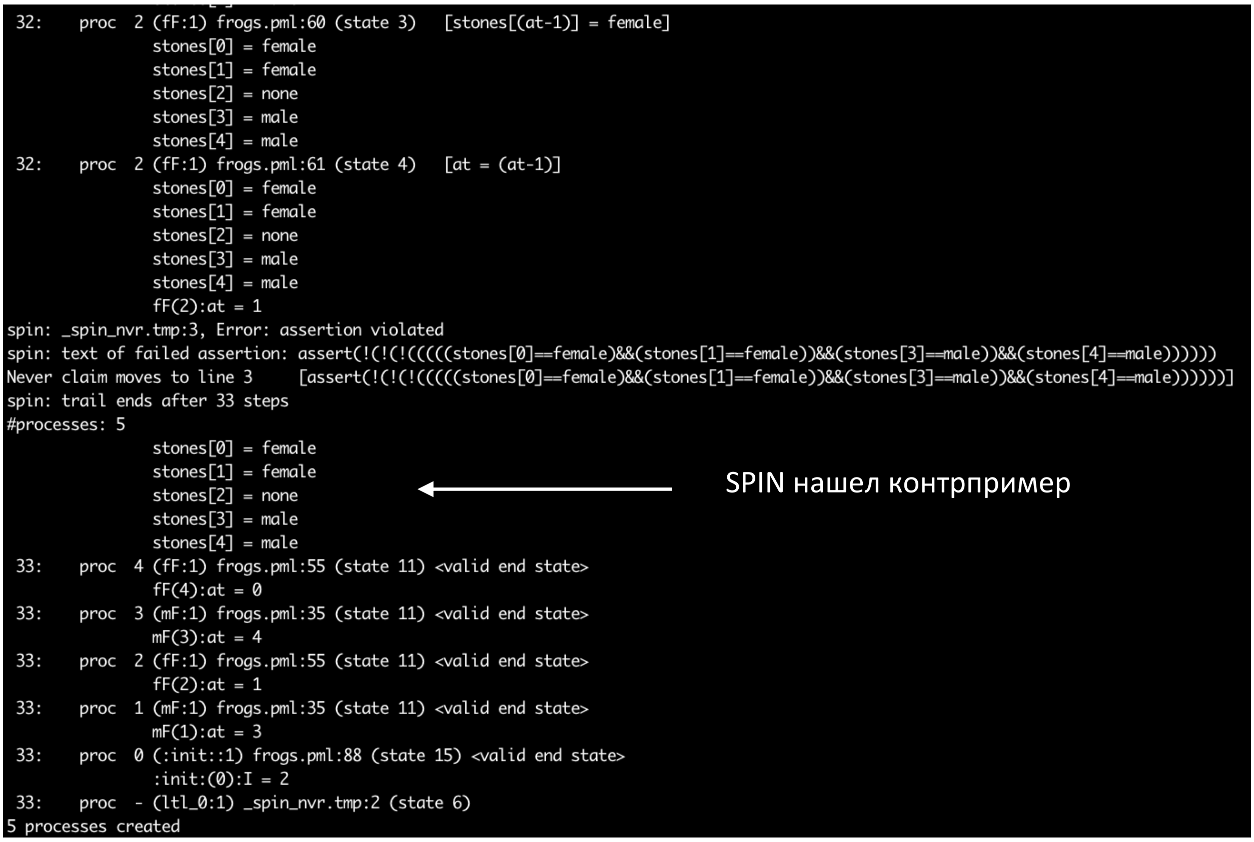
```
                stones[2] = female
                stones[3] = none
                stones[4] = female
18:     proc  4 (fF:1) frogs.pml:61 (state 4)   [at = (at-1)]
                stones[0] = male
                stones[1] = male
                stones[2] = female
                stones[3] = none
                stones[4] = female
                fF(4):at = 2
20:     proc  3 (mF:1) frogs.pml:46 (state 6)   [((((at<(5-2))&&(stones[(at+1)]!=none))&&(stones[(at+2)]==none)))]
20:     proc  3 (mF:1) frogs.pml:47 (state 7)   [stones[at] = none]
                stones[0] = male
                stones[1] = none
                stones[2] = female
                stones[3] = none
                stones[4] = female
20:     proc  3 (mF:1) frogs.pml:48 (state 8)   [stones[(at+2)] = male]
                stones[0] = male
                stones[1] = none
                stones[2] = female
                stones[3] = male
                stones[4] = female
20:     proc  3 (mF:1) frogs.pml:49 (state 9)   [at = (at+2)]
                stones[0] = male
                stones[1] = none
                stones[2] = female
                stones[3] = male
                stones[4] = female
                mF(3):at = 3
22:     proc  1 (mF:1) frogs.pml:38 (state 1)   [(((at<(5-1))&&(stones[(at+1)]==none)))]
22:     proc  1 (mF:1) frogs.pml:39 (state 2)   [stones[at] = none]
                stones[0] = none
                stones[1] = none
                stones[2] = female
                stones[3] = male
                stones[4] = female
22:     proc  1 (mF:1) frogs.pml:40 (state 3)   [stones[(at+1)] = male]
                stones[0] = none
                stones[1] = male
                stones[2] = female
                stones[3] = male
                stones[4] = female
22:     proc  1 (mF:1) frogs.pml:41 (state 4)   [at = (at+1)]
                stones[0] = none
                stones[1] = male
                stones[2] = female
                stones[3] = male
                stones[4] = female
                mF(1):at = 1
24:     proc  4 (fF:1) frogs.pml:66 (state 6)   [((((at>1)&&(stones[(at-1)]!=none))&&(stones[(at-2)]==none)))]
24:     proc  4 (fF:1) frogs.pml:67 (state 7)   [stones[at] = none]
                stones[0] = none
                stones[1] = male
                stones[2] = none
                stones[3] = male
                stones[4] = female
24:     proc  4 (fF:1) frogs.pml:68 (state 8)   [stones[(at-2)] = female]
                stones[0] = female
```



```
                stones[1] = male
                stones[2] = none
                stones[3] = male
                stones[4] = female
24:     proc  4 (fF:1) frogs.pml:69 (state 9)   [at = (at-2)]
                stones[0] = female
                stones[1] = male
                stones[2] = none
                stones[3] = male
                stones[4] = female
                fF(4):at = 0
```



```
26:     proc  2 (fF:1) frogs.pml:66 (state 6)   [((((at>1)&&(stones[(at-1)]!=none))&&(stones[(at-2)]==none)))]
26:     proc  2 (fF:1) frogs.pml:67 (state 7)   [stones[at] = none]
                stones[0] = female
                stones[1] = male
                stones[2] = none
                stones[3] = male
                stones[4] = none
26:     proc  2 (fF:1) frogs.pml:68 (state 8)   [stones[(at-2)] = female]
                stones[0] = female
                stones[1] = male
                stones[2] = female
                stones[3] = male
                stones[4] = none
26:     proc  2 (fF:1) frogs.pml:69 (state 9)   [at = (at-2)]
                stones[0] = female
                stones[1] = male
                stones[2] = female
                stones[3] = male
                stones[4] = none
                fF(2):at = 2
28:     proc  3 (mF:1) frogs.pml:38 (state 1)   [(((at<(5-1))&&(stones[(at+1)]==none)))]
28:     proc  3 (mF:1) frogs.pml:39 (state 2)   [stones[at] = none]
                stones[0] = female
                stones[1] = male
                stones[2] = female
                stones[3] = none
                stones[4] = none
28:     proc  3 (mF:1) frogs.pml:40 (state 3)   [stones[(at+1)] = male]
                stones[0] = female
                stones[1] = male
                stones[2] = female
                stones[3] = none
                stones[4] = male
28:     proc  3 (mF:1) frogs.pml:41 (state 4)   [at = (at+1)]
                stones[0] = female
                stones[1] = male
                stones[2] = female
                stones[3] = none
                stones[4] = male
                mF(3):at = 4
30:     proc  1 (mF:1) frogs.pml:46 (state 6)   [((((at<(5-2))&&(stones[(at+1)]!=none))&&(stones[(at+2)]==none)))]
30:     proc  1 (mF:1) frogs.pml:47 (state 7)   [stones[at] = none]
                stones[0] = female
                stones[1] = none
                stones[2] = female
                stones[3] = none
                stones[4] = male
30:     proc  1 (mF:1) frogs.pml:48 (state 8)   [stones[(at+2)] = male]
                stones[0] = female
                stones[1] = none
                stones[2] = female
                stones[3] = male
                stones[4] = male
30:     proc  1 (mF:1) frogs.pml:49 (state 9)   [at = (at+2)]
                stones[0] = female
                stones[1] = none
                stones[2] = female
                stones[3] = male
                stones[4] = male
                mF(1):at = 3
32:     proc  2 (fF:1) frogs.pml:58 (state 1)   [(((at>0)&&(stones[(at-1)]==none)))]
32:     proc  2 (fF:1) frogs.pml:59 (state 2)   [stones[at] = none]
                stones[0] = female
                stones[1] = none
                stones[2] = none
                stones[3] = male
                stones[4] = male
```



```
 32:    proc  2 (fF:1) frogs.pml:60 (state 3)   [stones[(at-1)] = female]
                stones[0] = female
                stones[1] = female
                stones[2] = none
                stones[3] = male
                stones[4] = male
 32:    proc  2 (fF:1) frogs.pml:61 (state 4)   [at = (at-1)]
                stones[0] = female
                stones[1] = female
                stones[2] = none
                stones[3] = male
                stones[4] = male
                fF(2):at = 1
spin: _spin_nvr.tmp:3, Error: assertion violated
spin: text of failed assertion: assert(!(!(!(((((stones[0]==female)&&(stones[1]==female))&&(stones[3]==male))&&(stones[4]==male))))))
Never claim moves to line 3     [assert(!(!(!(((((stones[0]==female)&&(stones[1]==female))&&(stones[3]==male))&&(stones[4]==male))))))]
spin: trail ends after 33 steps
#processes: 5
                stones[0] = female
                stones[1] = female
                stones[2] = none
                stones[3] = male
                stones[4] = male
 33:    proc  4 (fF:1) frogs.pml:55 (state 11) <valid end state>
                fF(4):at = 0
 33:    proc  3 (mF:1) frogs.pml:35 (state 11) <valid end state>
                mF(3):at = 4
 33:    proc  2 (fF:1) frogs.pml:55 (state 11) <valid end state>
                fF(2):at = 1
 33:    proc  1 (mF:1) frogs.pml:35 (state 11) <valid end state>
                mF(1):at = 3
 33:    proc  0 (:init::1) frogs.pml:88 (state 15) <valid end state>
                :init:(0):I = 2
 33:    proc  - (ltl_0:1) _spin_nvr.tmp:2 (state 6)
5 processes created
```

SPIN нашел контрпример

Заключение

В ходе выполнения первой самостоятельной работы произошло знакомство со средой SPIN. А именно, был смоделирован алгоритм на языке Promela, была произведена его симуляция, а также верификация.

Список литературы

1. Ben-Ari.M. Principles of concurrent and distributed programming. – 2006
2. Ю. Г. Карпов. И. В. Шошмина. Верификация распределенных систем. – 2011
3. Ю. Г. Карпов. И. В. Шошмина. Введение в язык Promela и систему комплексной верификации SPIN. – 2009
4. SPIN Run-Time Options [Электронный ресурс], URL: http://spinroot.com/spin/Man/Spin.html