Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Верификация и анализ программ

Отчет по лабораторной работе №2

Верификация модели с помощью Spin

Выполнил студент гр. 23541/3 Смирнов С.В.

(подпись)

Преподаватель Ицыксон В.М.

(подпись)

“ ” 2019 г.

Санкт – Петербург

2018

## Цель работы

Необходимо построить автоматную модель для следующей задачи:

Парк развлечений, на котором есть две кассы для покупки билетов на аттракционы и две очереди к аттракционам. Для того, чтобы прокатиться на любом аттракционе необходимо отстоять очередь в кассу, приобрести билет и затем отстоять очередь к аттракциону. Человек из очереди подходит к кассе, оплачивает билет, после оплаты клиенты встают в очередь к интересующему аттракциону. На каждой кассе действует лотерея- каждый клиент тянет билет из корзинки и если его номер равен 5 или 7 (в зависимости от кассы), то клиент пропускается без очереди.

В том случае, когда очередь в кассу занимает больше положенного времени, клиент может уйти. Считается, что если клиент приобрел билет, то он его израсходует.

При описании модели стремиться сделать ее масштабируемой по отношению к клиентам и кассам. За стандартную модель были приняты следующие значения: длина очереди - 2, количество касс – 2.

Проверить следующие утверждения:

* Все кассы, в конечном итоге, освободятся.
* Все очереди, в конечном итоге, освободятся.
* Все клиенты прокатятся на аттракционе или покинут парк
* Если клиент был в очереди, то, в конечном итоге, он прокатится.

## Построение модели

Для реализации описанной системы были построены два конечных автомата. Первый отвечает за поведение кассира, второй за движение клиентов

Кассир имеет следующие состояния:

* FREE
* BUSY

BUSY

FREE

Рис.1. Конечный автомат кассира

Состояния клиента:

* WAIT\_PAY – ожидание в очереди кассы
* WAIT\_ACTION – ожидание в очереди аттракциона
* ACTION – катание на аттракционе- позитивное завершение
* GO\_AWAY - негативное завершение- у клиента кончилось терпение

Клиент встает в очередь. Если клиент не выигрывает в лотерее, то он встает в очередь на аттракцион. Если клиент получает счастливый билет- он пропускается без очереди. Если время ожидания обслуживания больше 10 секунд, то клиент уходит.

ACTION

GO\_AWAY

WAIT\_ACTION

WAIT\_PAY

10

Рис.2. Конечный автомат движения клиента

Ниже представлена реализация описанной модели на языке PROMELA:

#define num\_visitors 5

#define waitingTime 10

mtype = { WAIT\_PAY, WAIT\_ACTION, ACTION, GO\_AWAY };

mtype = { FREE, BUSY };

mtype visitor[num\_visitors];

mtype pay\_queue[2];

mtype wait\_queue[2];

proctype queueAction(int id)

{

atomic{

int lottery;

int timer = 0;

do

::

if

:: visitor[id] == WAIT\_PAY ->

if

:: pay\_queue[0] == FREE ->

pay\_queue[0] = BUSY;

select( lottery : 1..25);

if

:: lottery == 5 ->

visitor[id] = ACTION;

:: lottery != 5

visitor[id] = WAIT\_ACTION;

fi

:: pay\_queue[1] == FREE ->

pay\_queue[1] = BUSY;

if

:: lottery == 7 ->

visitor[id] = ACTION;

:: lottery != 7

visitor[id] = WAIT\_ACTION;

fi

fi

:: visitor[id] == WAIT\_ACTION;

if

:: wait\_queue[0] == FREE ->

visitor[id] = ACTION;

wait\_queue[0] = BUSY;

:: wait\_queue[1] == FREE ->

visitor[id] = ACTION;

wait\_queue[1] = BUSY;

:: timer < waitingTime ->

timer = timer + 1;

:: timer == waitingTime ->

visitor[id] = GO\_AWAY;

fi

:: visitor[id] == ACTION || visitor[id] == GO\_AWAY->

if

:: pay\_queue[0] == BUSY ->

pay\_queue[0] = FREE;

:: pay\_queue[1] == BUSY ->

pay\_queue[1] = FREE;

:: wait\_queue[0] == BUSY ->

wait\_queue[0] = FREE;

:: wait\_queue[1] == BUSY ->

wait\_queue[1] = FREE;

fi

fi

od

}

}

init {

int i;

for(i: 0 .. (num\_visitors - 1)){

visitor[i] = WAIT\_PAY;

}

pay\_queue[0] = FREE;

pay\_queue[1] = FREE;

wait\_queue[0] = FREE;

wait\_queue[1] = FREE;

for(i: 0 .. (num\_visitors -1)){

run queueAction(i)

}

}

ltl rool1{

<>[] (pay\_queue[0] == FREE && pay\_queue[1] == FREE)

}

ltl rool2{

<>[] ( wait\_queue[0] == FREE && wait\_queue[1] == FREE)

}

ltl rool3{

<>[] ((visitor[0] == ACTION || visitor[0] == GO\_AWAY) && (visitor[1] ==

ACTION || visitor[1] == GO\_AWAY) && (visitor[2] == ACTION

|| visitor[2] == GO\_AWAY) && (visitor[3] == ACTION || visitor[3] ==

GO\_AWAY) )

}

ltl rool4{

[]((visitor[0] == WAIT\_PAY && (pay\_queue[0] == FREE || pay\_queue[1] ==

FREE || wait\_queue[0] == FREE))->(<>(visitor[0] ==

WAIT\_ACTION))->(<>(visitor[0] == ACTION || visitor[0] == GO\_AWAY))) &&

[]((visitor[1] == WAIT\_PAY && (pay\_queue[0] == FREE || pay\_queue[1] ==

FREE || wait\_queue[0] == FREE))->(<>(visitor[1] ==

WAIT\_ACTION))->(<>(visitor[1] == ACTION || visitor[1] == GO\_AWAY))) &&

[]((visitor[2] == WAIT\_PAY && (pay\_queue[0] == FREE || pay\_queue[1] ==

FREE || wait\_queue[0] == FREE))->(<>(visitor[2] ==

WAIT\_ACTION))->(<>(visitor[2] == ACTION || visitor[2] == GO\_AWAY))) &&

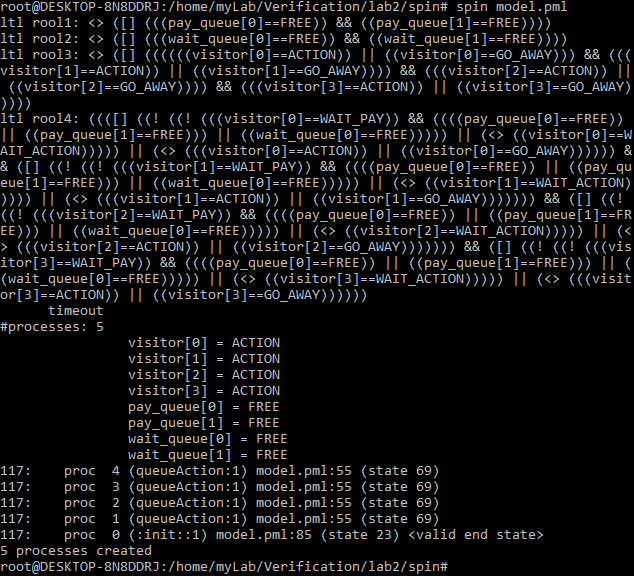
[]((visitor[3] == WAIT\_PAY && (pay\_queue[0] == FREE || pay\_queue[1] ==

FREE || wait\_queue[0] == FREE))->(<>(visitor[3] ==

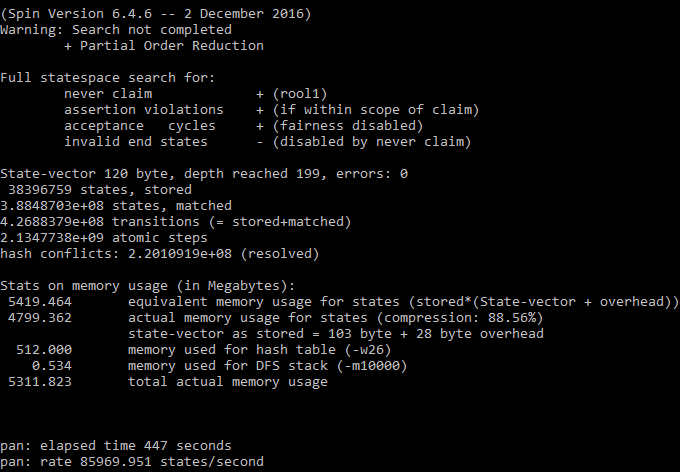
WAIT\_ACTION))->(<>(visitor[3] == ACTION || visitor[3] == GO\_AWAY)))

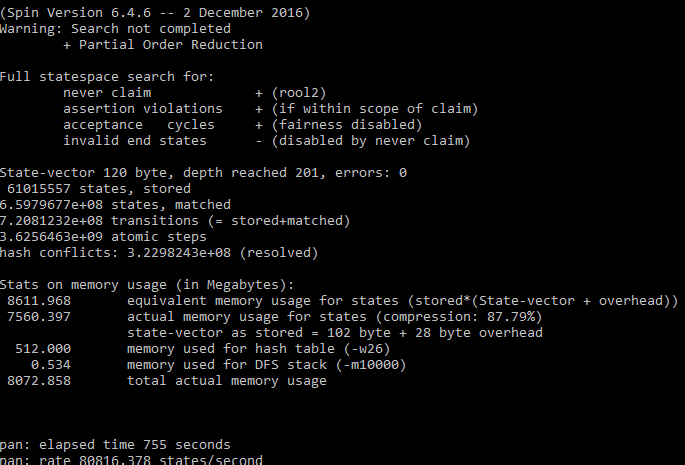
}

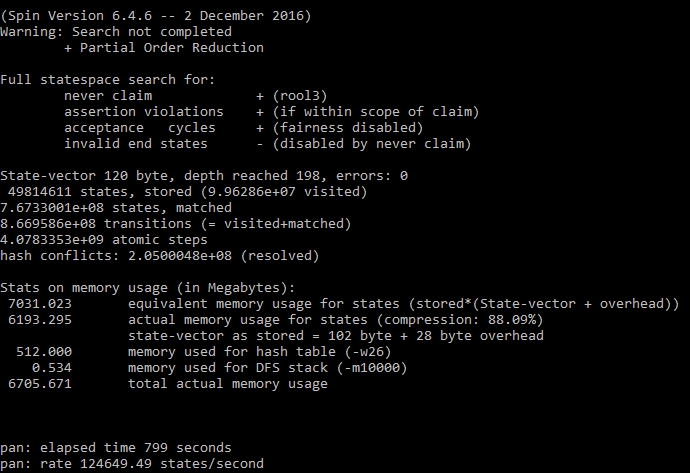
Результат запуска:

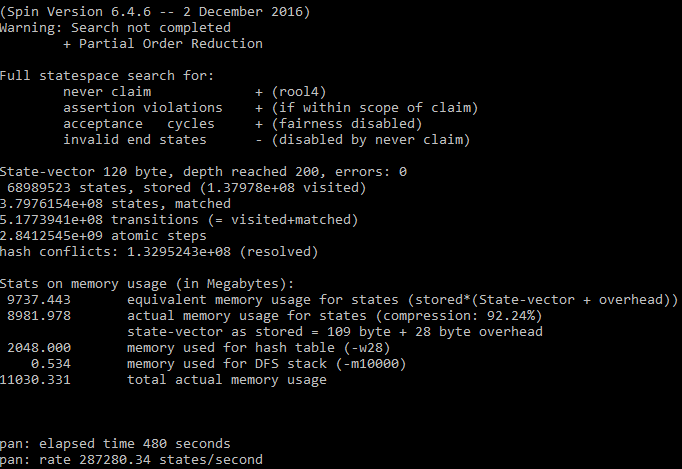


Результат проверки составленных требований:









## Выводы

В ходе работы была описана и верифицирована модель системы работы парка аттракционов. Для описания модели был использован язык PROMELA. А для описания спецификаций --- LTL формулы. Верификация была проведена с помощью утилиты Spin.

Исходя из опыта работы с утилитой Spin, полученного в ходе выполнения данной работы, можно сделать вывод о том, что верификация требует не мало технических ресурсов и занимает значительное количество времени. Однако, она позволяет решить множество задач, которые оказываются трудоемкими для решения человеком.