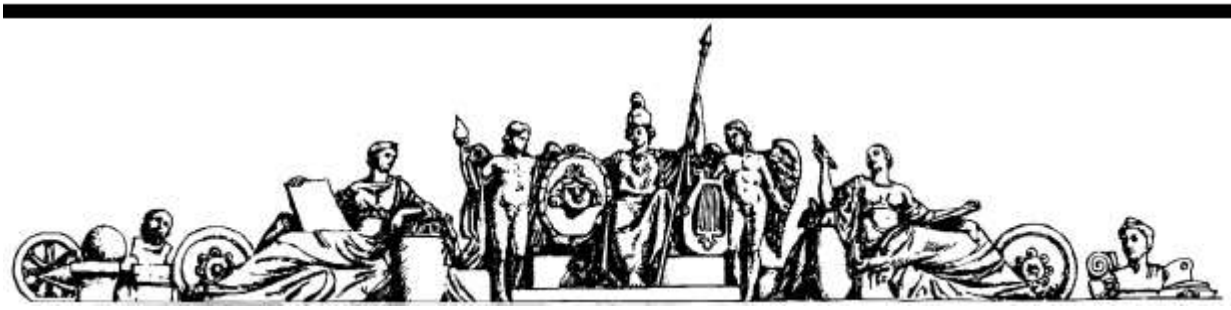
Государственное образовательное учреждение высшего профессионального образования
«Московский Государственный Технический Университет имени Н.Э. Баумана»



ОТЧЁТ

По математическим основам верификации ПО

Лабораторная работа № 3

Тема: « ***Моделирование сетевого протокола*** »

Студент: Нгуен Дык Бинь
Группа: ИУ7-41М
Преподаватель: Кузнецова О.В.

Москва, 2020

1. Задание

На языке promela необходимо описать прототип сетевого протокола, реализующего отправление и получение данных.

В лабораторной работе производим запись и чтение в/из канал/а.

2. Текст программы

```
mtype = {msg, ack};
chan s_r = [2] of {mtype, bit};
chan r_s = [2] of {mtype, bit};

active proctype sender() {
    bit seqno;
    do
    :: s_r!msg, seqno ->
        if
        :: r_s?ack, eval(seqno) ->            // Считываем новое сообщение
            seqno = 1 - seqno;
        :: r_s?ack, eval(1 - seqno)
        fi
    od
}

active proctype receiver() {
    bit expect, seqno;
    do
    :: s_r?msg, seqno ->
        r_s!ack, seqno;
        if
        :: seqno == expect;                   // Сохраняем сообщение
            printf("Expected!\n");
            expect = 1 - expect
        :: else                               // Игнорируем сообщение
        fi
    od
}
```

3. Подробное описание

- В программе 2 процесса: получатель и отправитель.
  - Тип сообщений: $mtype = \{msg, ack\}$;
  - Отправка сообщения через команду: $\boldsymbol{a?b,c}$
  - Приём сообщения через команду: $\boldsymbol{a!b,c}$
- К каждому сообщению добавляется один бит.
- Получатель сообщает о доставке сообщения, возвращая бит отправителя.
- Если отправитель убедился в доставке сообщения, он отправляет новое, изменяя значения бита:

$$seqno = 1 - seqno;$$

- Если значение бита не изменилось, получатель считает, что идёт повтор сообщения.
- В данной программе используется функция ***eval(x)***, она отображает текущее значение ***x*** на константу (0 или 1), которая служит ограничением для принимаемых сообщений.
- Процесс передачи сообщений:

4. Пример работы программы

- Приведен результат моделирования 20х первых шагов:

```
~# spin -u20 -c channel.pml
proc 0 = sender
proc 1 = receiver
q\p   0   1
  1   s_r!msg,0
  1   .   s_r?msg,0
  2   .   r_s!ack,0
  2   r_s?ack,0
        Expected!
  1   s_r!msg,1
  1   .   s_r?msg,1
  2   .   r_s!ack,1
  2   r_s?ack,1
        Expected!
-------------
depth-limit (-u20 steps) reached
-------------
final state:
-------------
#processes: 2
              queue 1 (s_r):
              queue 2 (r_s):
 20:    proc  1 (receiver:1) channel.pml:25 (state 5)
 20:    proc  0 (sender:1) channel.pml:15 (state 8)
2 processes created
```

- Граф состояний (-u20)