

# Математические методы верификации схем и программ

Лекторы:

Захаров Владимир Анатольевич

Подымов Владислав Васильевич

e-mail рассказчика:

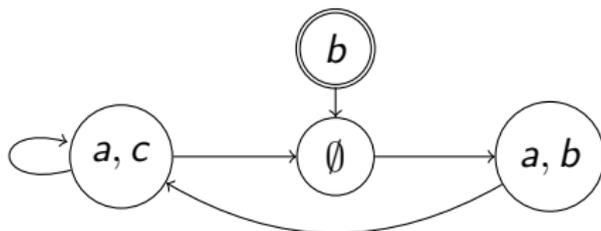
[valdus@yandex.ru](mailto:valdus@yandex.ru)

Осень 2018

## Spin (практика)

# Упражнение 1: реализация модели Крипке

Проверить выполнимость свойств в модели Крипке



- ▶  $\mathbf{G}(a \rightarrow b \vee c)$
- ▶  $\mathbf{GF}a$
- ▶  $\mathbf{FG}a$
- ▶  $\mathbf{GF}\neg c \rightarrow \mathbf{F}(a \wedge b)$
- ▶  $\mathbf{GF}\neg c \rightarrow \mathbf{G}(\neg b \mathbf{U} a \wedge b)$

## Упражнение 2: доступ в критическую секцию

Два процесса с разделяемой булевой переменной

```
bool free = true;
```

исполняют одну и ту же программу:

```
while(true) {  
    NONCRITICAL  
    block(free);  
    free = false;  
    CRITICAL  
    free = true;  
}
```

Процесс может сколь угодно долго находиться в критической и некритической секциях (CRITICAL и NONCRITICAL соответственно)

Переменная `free` не изменяется процессом, находящимся в этих секциях

## Упражнение 2: доступ в критическую секцию

Два процесса с разделяемой булевой переменной

```
bool free = true;
```

исполняют одну и ту же программу:

```
while(true) {  
    NONCRITICAL  
    block(free);  
    free = false;  
    CRITICAL  
    free = true;  
}
```

Инструкция в каждой отдельной строке выполняется атомарно

Инструкция `block(free)` блокирует процесс, пока не станет истинным условие `free`

## Упражнение 2: доступ в критическую секцию

Два процесса с разделяемой булевой переменной

```
bool free = true;
```

исполняют одну и ту же программу:

```
while(true) {  
    NONCRITICAL  
    block(free);  
    free = false;  
    CRITICAL  
    free = true;  
}
```

Выяснить,

- ▶ могут ли процессы одновременно находиться в своих критических секциях
- ▶ возможна ли блокировка обоих процессов сразу,
- ▶ верно ли, что, выйдя из некритической секции, процесс рано или поздно достигнет критической

## Упражнение 2: доступ в критическую секцию

Два процесса с разделяемой булевой переменной

```
bool free = true;
```

исполняют одну и ту же программу:

```
while(true) {  
    NONCRITICAL  
    block(free);  
    free = false;  
    CRITICAL  
    free = true;  
}
```

Добавить в модель справедливость: процесс не может бесконечно долго находиться в критической секции

Изменить модель так, чтобы пара инструкций `block(free); free = false;` выполнялась атомарно: если `block(free)` не блокирует процесс, то немедленно выполнить `free = false`

## Упражнение 2: доступ в критическую секцию

Два процесса с разделяемой булевой переменной

```
bool free = true;
```

исполняют одну и ту же программу:

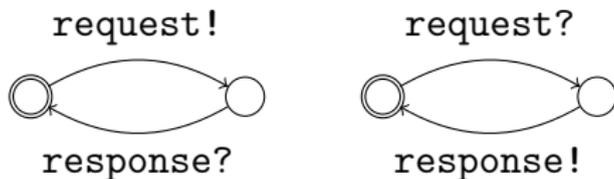
```
while(true) {  
    NONCRITICAL  
    block(free);  
    free = false;  
    CRITICAL  
    free = true;  
}
```

Внимательно посмотреть на флаг “**слабая справедливость**” в настройках верификации

## Упражнение 3: передача сообщений

Система состоит из одного клиента, одного сервера и  
двунаправленного канала связи между ними

Клиент и сервер описываются левым и правым автоматами  
соответственно:

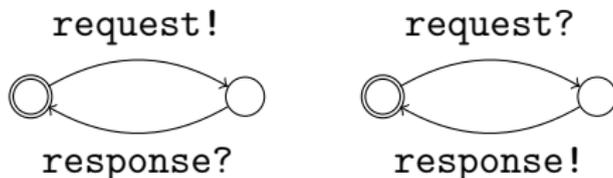


Запись  $m!$  означает, что при выполнении перехода сообщение  $m$   
посылается в канал, запись  $m?$  — что сообщение  $m$  принимается  
из канала

## Упражнение 3: передача сообщений

Система состоит из одного клиента, одного сервера и двунаправленного канала связи между ними

Клиент и сервер описываются левым и правым автоматами соответственно:



Выяснить,

- ▶ верно ли, что если клиент посылает сообщение `request`, то он обязательно примет сообщение `response`
- ▶ возможна ли ситуация, в которой и клиент, и сервер ожидают приёма сообщений

Рассмотреть два варианта устройства канала:

- ▶ синхронный
- ▶ асинхронный ёмкости 1

## Упражнение 4: синхронные системы

По комнате летает два комара

В начальный момент времени оба комара не жужжат

Если комар не жужжит, в следующий момент времени он начинает жужжать

Если комар жужжит, в следующий момент времени он перестаёт жужжать

Жужжание комара описывается булевой переменной

Описать систему жужжания двух комаров с таким требованием: в каждый момент времени либо оба комара жужжат, либо оба комара не жужжат

## Упражнение 5: переправа

Волк, коза, капуста и лодочник с лодкой стоят на левом берегу реки и хотят переправиться на правый

Только лодочник может грести

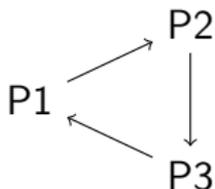
Лодка вмещает только двоих, включая лодочника

Нельзя оставлять волка с козой, а также козу с капустой на одном берегу без присмотра лодочника

Могут ли все переправиться на правый берег?

## Упражнение 6: распределённые алгоритмы

Три процесса соединены друг с другом в кольцо  
однонаправленными асинхронными каналами связи ёмкости 1:

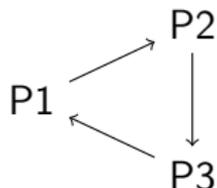


Каждый процесс имеет булеву переменную  $b$ , имеющую произвольное значение в начале работы системы, и **ровно два раза** делает следующее:

- ▶ посылает в канал значение  $b$
- ▶ принимает из канала значение, и если принято `true`, то записывает `true` в переменную  $b$

## Упражнение 6: распределённые алгоритмы

Три процесса соединены друг с другом в кольцо  
одна направленными асинхронными каналами связи ёмкости 1:



Убедиться, что

- ▶ каждый процесс рано или поздно выполнит всю свою последовательность действий
- ▶ (каждый процесс в конце работы хранит значение true)  $\Leftrightarrow$  (хотя бы один процесс в начале работы хранил значение true)
- ▶ (каждый процесс в конце работы хранит значение false)  $\Leftrightarrow$  (ни один процесс в начале работы не хранил значение true)