Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «**Информатика и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

ОТЧЕТ ПО по Лабораторной работе №1
по курсу
«Математические основы верификации ПО»
Тема
«Знакомство с языком Promela»

Студент группы **ИУ7И-41М** **Баматраф Сохайб С.А.**

(И.О. Фамилия)

Преподаватель **Кузнецова О.В.**

(И.О. Фамилия)

***2024** г.*

Содержание

1. Введение

В этом докладе мы исследуем использование языка Promela и программы проверки моделей SPIN для моделирования и анализа протокола связи клиент-сервер. Цель этой задачи - продемонстрировать, как инструменты формальной верификации могут применяться для обеспечения корректности и надежности коммуникационных протоколов в программных системах.

1.1 Цель и задачи

Цель — Основная цель этого лабораторного занятия - смоделировать простое взаимодействие клиента и сервера, при котором клиент отправляет запросы серверу, сервер обрабатывает их, а затем отправляет ответы обратно клиенту. Применяя методы формальной верификации, мы хотим проверить корректность протокола и выявить любые потенциальные проблемы синхронизации или связи.

Задачи:

Для небольшого фрагмента программы необходимо описать модель этой программы на Promela и изучить её (SPIN).

2. Фрагмент кода

Ниже приведена упрощенная версия кода Promela, используемого в данной модели:

Листинг 1 — Код модели.

```
mtype = {REQ, RES};
inline rand(num) {
   num = (_pid % 5) + 1;
}
```

```
proctype Client(chan ch; byte reqCount) {
    byte reqbit, resbit;
    printf("CLIENT: running, pid=%d\n", _pid);
    do
    :: reqCount > 0 ->
        rand(reqbit);
        printf("CLIENT: sending REQ with value %d\n", reqbit);
        ch ! REQ, reqbit;
        printf("CLIENT: waiting for RES\n");
        ch ? RES, resbit;
        printf("CLIENT: RES received with value %d\n", resbit);
        reqCount = reqCount - 1;
        printf("CLIENT: REQ and RES process complete for value %d\n", reqbit);
    :: reqCount == 0 ->
        printf("CLIENT: All requests processed. Client stops.\n");
        break;
    od
}
proctype Server(chan ch) {
    byte reqbyte, resbyte;
    printf("SERVER: running, pid=%d\n", _pid);

    do
    :: ch ? REQ, reqbyte ->
        printf("SERVER: REQ received with code %d\n", reqbyte);
        rand(resbyte);
```

```
        printf("SERVER: processing REQ, sending RES with value %d\n", resbyte);
        ch ! RES, resbyte;
        printf("SERVER: RES sent for request %d\n", reqbyte);
    od
}
init {
    chan ch = [2] of {mtype, byte};  // Channel with 2 slots for mtype and byte
    run Client(ch, 5);
    run Server(ch);
}
```

3. Описание модели

Модель состоит из двух процессов: клиента и сервера. Клиент генерирует серию запросов, представленных случайными числами от 1 до 5, и отправляет их на сервер по каналу. Сервер получает каждый запрос, обрабатывает его, генерируя соответствующий ответ (также случайное число), и отправляет его обратно клиенту.

Канал связи между клиентом и сервером рассчитан на обработку двух типов сообщений: запросов (REQ) и ответов (RES). Оба процесса используют встроенную функцию rand(num) для генерации этих значений, что имитирует динамический аспект обработки данных в реальном мире.

4. Перечисление множества состояний и текстовое пояснение

Состояние клиента:

1. Старт: Клиент инициирует и готовится к отправке запросов.
2. Отправить REQ: Клиент отправляет запрос на сервер.
3. Ждать RES: Клиент ожидает получения ответа от сервера.
4. Обработать RES: Клиент обрабатывает полученный ответ.
5. Проверить наличие новых запросов: Решает, отправлять ли еще один запрос или завершить работу.

Состояния сервера:

1. Старт: Сервер инициирует и ожидает запрос.
2. Получение REQ: сервер получает запрос.
3. Обработать и отправить RES: Сервер обрабатывает запрос и отправляет ответ.

5. Граф переходов между состояниями модели

Приведенные ниже графики наглядно изображают переходы между этими состояниями как для клиента, так и для сервера. Эти диаграммы упрощают понимание того, как клиент и сервер взаимодействуют в ходе коммуникационного процесса.

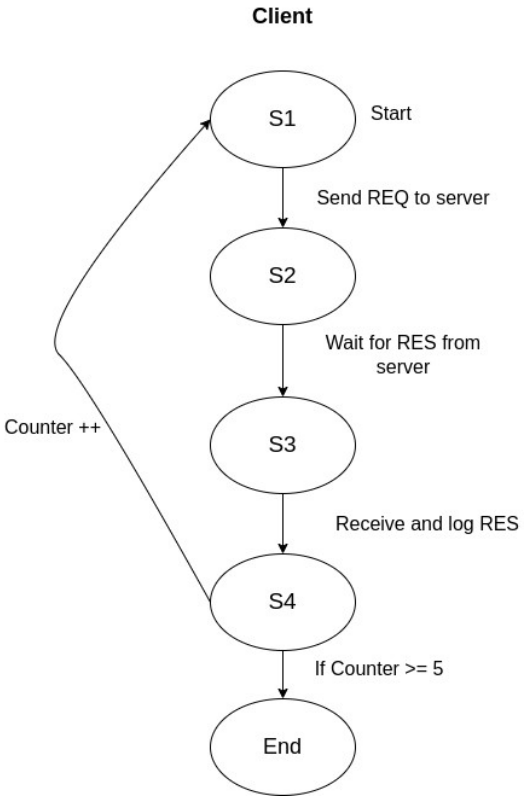

Рисунок 1.- Блок-схема процесса работы с клиентом.

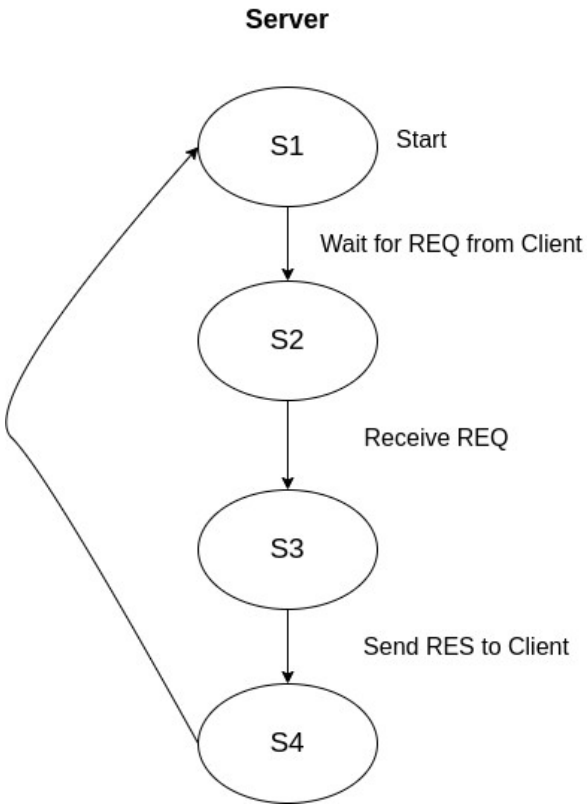

Рисунок 2.- Блок-схема для процесса сервера

Заключение

Это упражнение эффективно демонстрирует, как Promela и SPIN могут быть использованы для моделирования и анализа базового протокола связи клиент-сервер. Модель помогла определить критические взаимодействия и возможные состояния в процессе коммуникации. Формально проверив модель с помощью SPIN, мы убедились, что протокол работает корректно при заданных условиях, тем самым повысив уверенность в надежности и корректности протокола.

В целом, применение формальных методов в разработке программного обеспечения, как показано в данной работе, имеет неоценимое значение для создания надежных и безошибочных систем, особенно в сценариях, где надежность имеет решающее значение.

Литература

1. https://github.com/sohaibssb/Mathematical-basics-of-verification/tree/main/PromelaLanguage-Lab1.