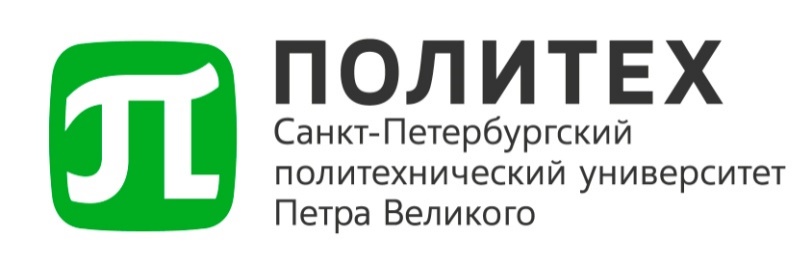
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

ВШПИ



**Отчёт по выполнению самостоятельной рабоы по дисциплине** «**Модальные логики и многоагентные системы**»

**Верификация аппаратуры.**

Студенты группы 3530202/80201 Е.А. Козлова

Руководитель Ю.Г. Карпов

Санкт-Петербург

2022 г.

В данной работе предстоит построить свою логическую схему с памятью, которая будет подчиняться выбранным логическим формулам f1 и f2.

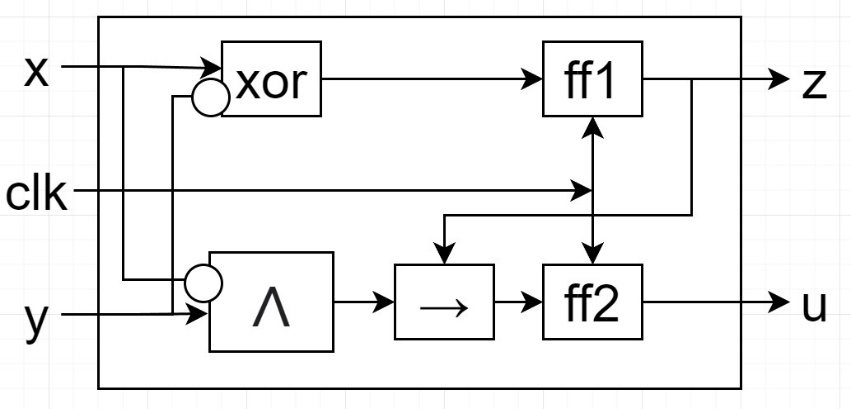
Далее, для воссозданной схемы строится структура Крипке и автомат Бюхи. После построения автоматов, для схемы задается некое требование к поведению в словесном и формальном видах. Требование два раза тестируется и дальше начинается самое интересное – мы будем использовать широко известный алгоритм проверки моделей **– model checking**.

После проверки модели мы с уверенностью сможем сказать, выполняется ли введенное требование на всех входных цепочках нашей схемы, или же нет.

1. Изображение выглядит как текст

   Автоматически созданное описание**Выбранный вариант функций f1 и f2** Пусть:
2. **Построенная схема M:**

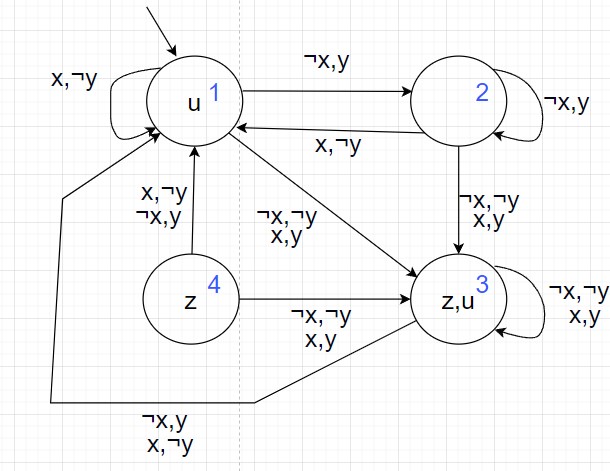
Изображение выглядит как текст

Автоматически созданное описание

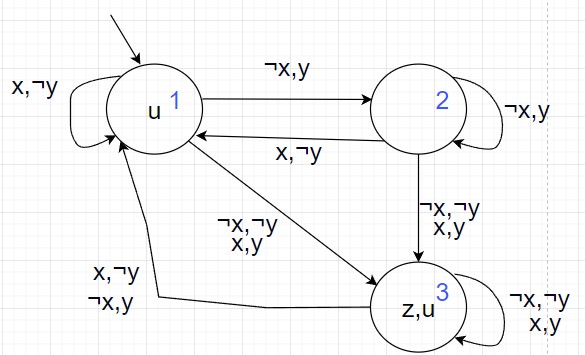
**Построим таблицу истинности по заданным формулам:**

|  |  |  |
| --- | --- | --- |
| **x y z u** | **z'=f1** | **u'=f2** |
| 0 0 0 0 | 1 | 1 |
| 0 0 0 1 | 1 | 1 |
| 0 0 1 0 | 1 | 1 |
| 0 0 1 1 | 1 | 1 |
| 0 1 0 0 | 0 | 0 |
| 0 1 0 1 | 0 | 0 |
| 0 1 1 0 | 0 | 1 |
| 0 1 1 1 | 0 | 1 |
| 1 0 0 0 | 0 | 1 |
| 1 0 0 1 | 0 | 1 |
| 1 0 1 0 | 0 | 1 |
| 1 0 1 1 | 0 | 1 |
| 1 1 0 0 | 1 | 1 |
| 1 1 0 1 | 1 | 1 |
| 1 1 1 0 | 1 | 1 |
| 1 1 1 1 | 1 | 1 |

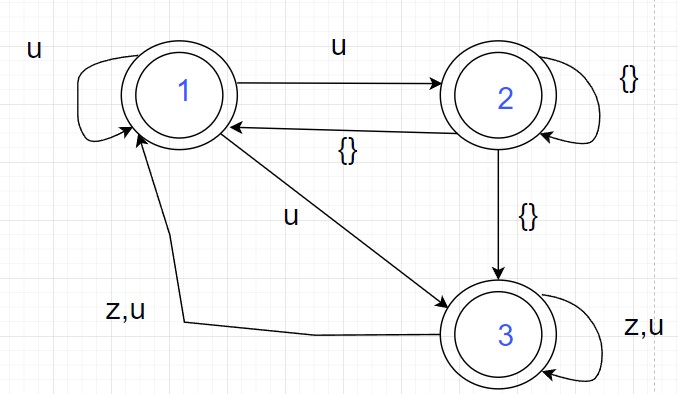
**Теперь можем построить структуру Крипке:**



**Заметим, что первое состояние не достижимо – тогда:**



**Построим автомат Бюхи для этой схемы:**



1. **Выбранный вариант простого требования к поведению**

Для построенной схемы введем следующее

ограничение:

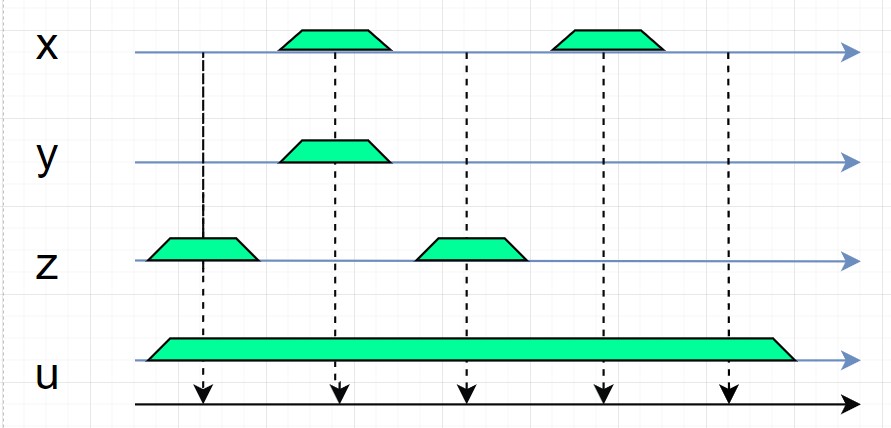
**1) Словесно**

Всегда либо z имеет значение 1, либо u имеет значение 1, либо они вместе имеют значение 1.

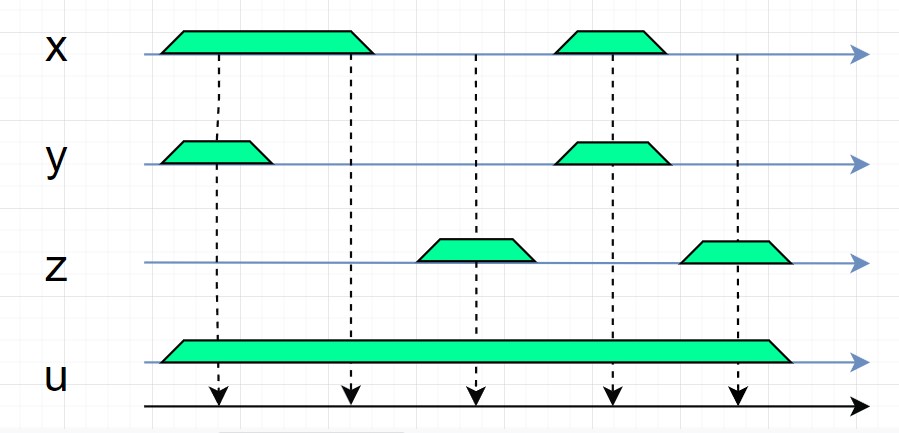
**2) Формулой LTL:**

Ф = [](u V z)

1. **Ручная тестовая проверка требования по двум тестовым траекториям Первая проверка:**



**Вторая проверка**



Как видно из проверок, контрпример введенному требованию не был найден. Для того, чтобы убедится в его существовании или не существовании, воспользуемся алгоритмом **model checking**

1. **Использование пакета для построения автомата Бюхи по отрицанию формулы LTL для выбранного требования к поведению схемы**

Ф = G[u V z], тогда

¬Ф = F[¬u ∧¬z]

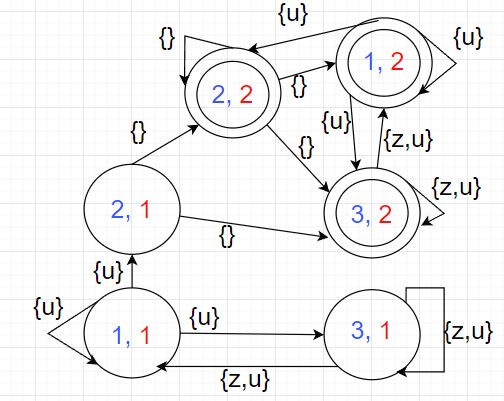
Изображение выглядит как спорт, спортивная игра, стена

Автоматически созданное описаниеПостроим автомат Бюхи 𝐵¬Ф с помощью предложенного пакета, получим:

1. **Синхронная композиция автомата Бюхи для схемы (**𝑩𝑴**) и автомата Бюхи для отрицания требования (**𝑩¬Ф)

Необходимо проверить, существует ли в автомате 𝐵𝑀 такая циклическая история повеления, на котором есть состояние, в котором нет ни g1, ни g2

Для этого, построим синхронную композицию двух автоматов Бюхи: 𝐵𝑀 и 𝐵¬Ф:

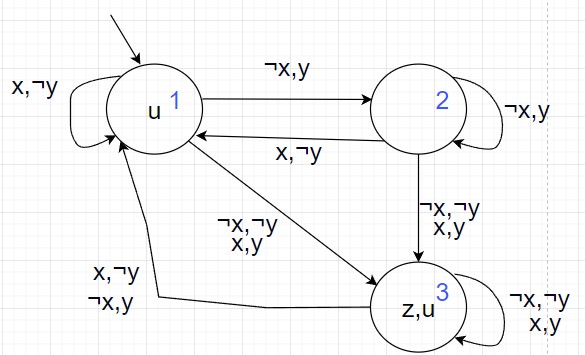


1. **Вариант найденной траектории (контрпримера)**

Найдем одну из траекторий, на которой состояние 2 автомата 𝐵¬Ф проходится в цикле.

Из этой траектории вычисляем контрпример – траекторию автомата 𝐵𝑀 **Траектория:** 1 2 3 1 2

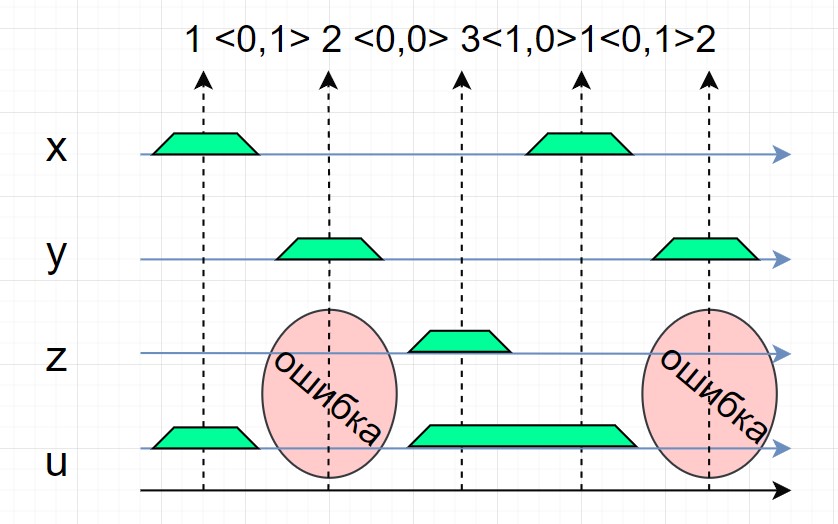
1. **Проверка того, что траектория действительно является контрпримером и действительно нарушает требование Структура Крипке:**



**Траектория: 1 2 3 1 2**

**Контрпример:**

Цепочка входов <x, y> схемы, приводящая к ошибке 1 2 3 1 2 на цепочке выходов схемы



1. **Заключение**

В заключении хочется сказать, что данная работа развила во мне много новых навыков – разобран процесс построения логической схемы, изучен механизм построения автомата Бюхи по структуре Крипке и по логической формуле. А самое главное – в результате работы, с помощью использования алгоритма modal checking, получилось найти контрпример, показывающий, что введенное мною требование к схеме - не выполняется.

Также алгоритм проверки моделей удивил меня тем, что он позволяет не просто доказать выполнение или невыполнение требования на конкретной схеме, а также позволяет найти контрпример – цепочку входных данных, приводящей к ошибке на цепочке выходных данных. Эта особенность отличает этот алгоритм от многих других, с которыми приходилось сталкиваться в решении задач различного рода. Рассмотренный на семинаре пример решения похожей задачи в полной мере дал понимание, как поэтапно проверять такие схемы на истинность утверждений.