ИТМО
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Генерация управляющих автоматов на основе генетического программирования и верификации

К.В. Егоров
специальность 05.13.11

Научный руководитель – д.т.н., проф. А.А. Шалыто

Актуальность темы

- Построенные вручную автоматы могут быть не оптимальны, а их построение очень трудоемко
  - Одно из решений – автоматизированная генерация управляющих конечных автоматов с помощью поисковой оптимизации, в частности, генетического программирования
- Тестирование позволяет находить ошибки, но не доказывает их отсутствие
  - Верификация позволяет проверить соответствие программы спецификации
- Автоматного программирования:
  - Возможность генерировать до 70 % исходного кода
  - Простота верификации на основе метода Model checking

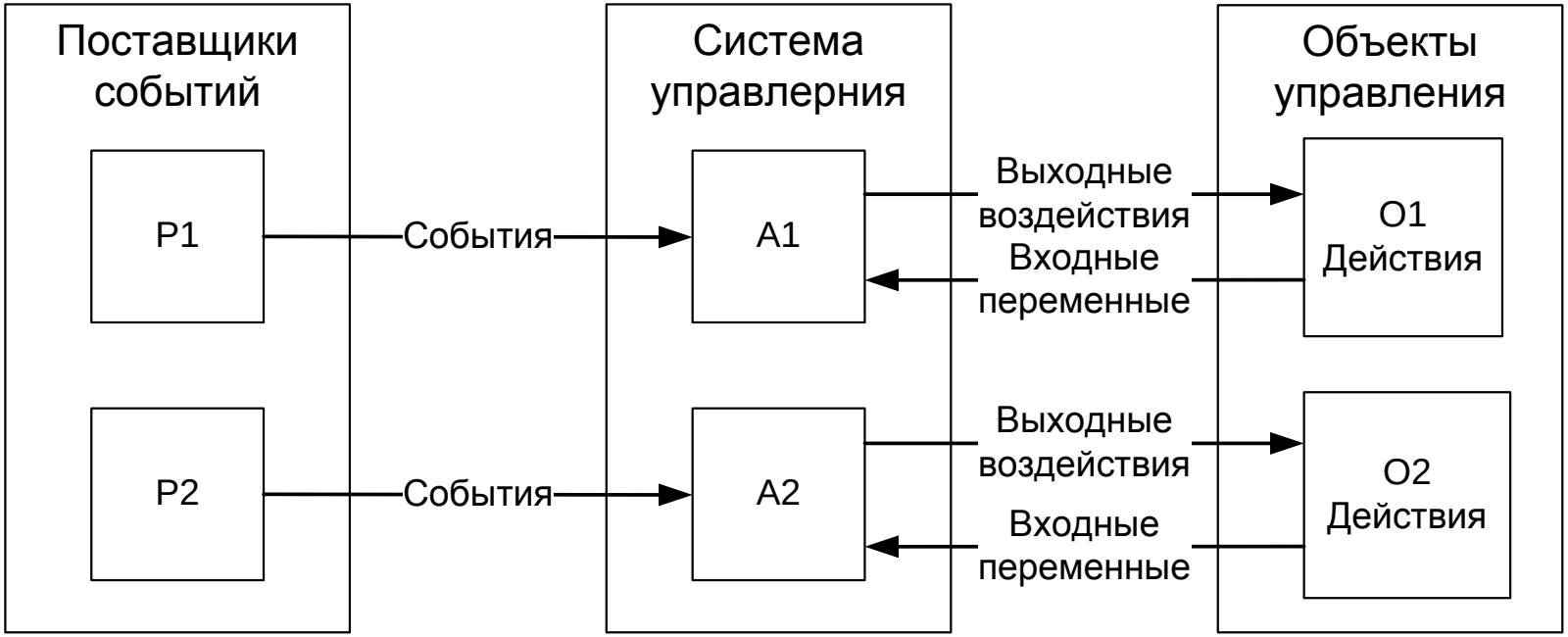
Задачи диссертационной работы

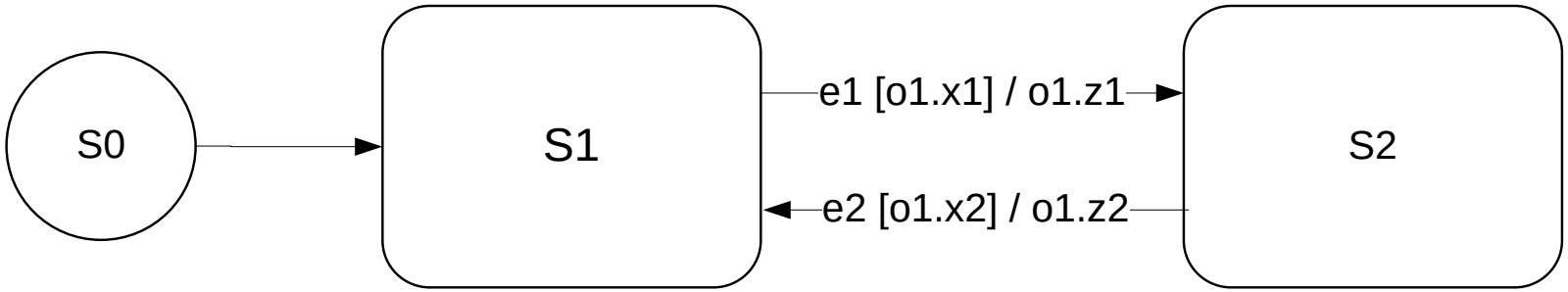
1. Предложить функцию приспособленности, учитывающую верификацию
2. Разработать операции скрещивания и мутации, учитывающие верификацию
3. Разработать алгоритм построения автоматов на основе генетического программирования с учетом контрактов, которые являются разновидностью темпоральных формул
4. Разработать алгоритм построения автоматов на основе генетического программирования по сценариям работы и верификации
5. Разработать верификатор, приспособленный для поддержки генерации автоматов на основе генетического программирования
6. Разработать технологию и инструментальное средство для генерации автоматов с помощью генетического программирования и верификации

Научная новизна

1. Предложена функция приспособленности, учитывающая выполнение формулы на части автомата. В известных работах учитывалось только выполнение или невыполнение каждой темпоральной формулы
2. Операции скрещивания и мутации отличаются от известных тем, что в ходе их выполнения используется верификация
3. Алгоритм генерации автоматов с учетом контрактов, который позволяет упростить запись некоторых темпоральных формул и ускорить построение автоматов по сравнению с применением темпоральных формул общего вида
4. Алгоритм генерации автоматов по темпоральным формулам и сценариям работы, который позволяет ускорить построение автоматов по сравнению с применением тестов

Автоматное программирования





Язык *Linear Temporal Logic (LTL)*

Темпоральные операторы:

- ***X*** *(ne**X**t)*, где «*Xp*» – в следующий момент выполнено *p*
- ***F*** *(in the **F**uture)*, где «*Fp*» – в будущем будет выполнено *p*
- ***G*** *(**G**lobally in the future)*, где «*Gp*» – всегда в будущем выполняется *p*
- ***U*** *(**U**ntil)*, где «*pUq*» – существует состояние, в котором выполнено *q* и во всех предыдущих выполняется *p*
- ***R*** *(**R**elease)*, где «*pRq*» – либо во всех состояниях выполняется *q*, либо существует состояние, в котором выполняется *p*, а во всех предыдущих выполнено *q*

LTL-формулы:

- *True, False*
- *Prop* – множество пропозициональных переменных
- Если *φ* и *ψ* – *LTL*-формулы, то:
  - *φ&ψ, φ|ψ* и *¬φ* – формулы
  - *Fφ, Xφ, Gφ, φRψ* и *φUψ* – формулы

Верификация автоматных программ

- Спецификация задается в виде набора формул на языке *LTL*
- Вместо пропозициональных переменных рассматриваются предикаты:
  - *wasEvent(e)* – переход совершен по событию *e*
  - *isInState(s)* – переход совершен в состояние *s*
  - *wasInState(s)* – переход совершен из состояния *s*
  - *wasAction(z)* – во время перехода было выполнено выходное воздействие *z*

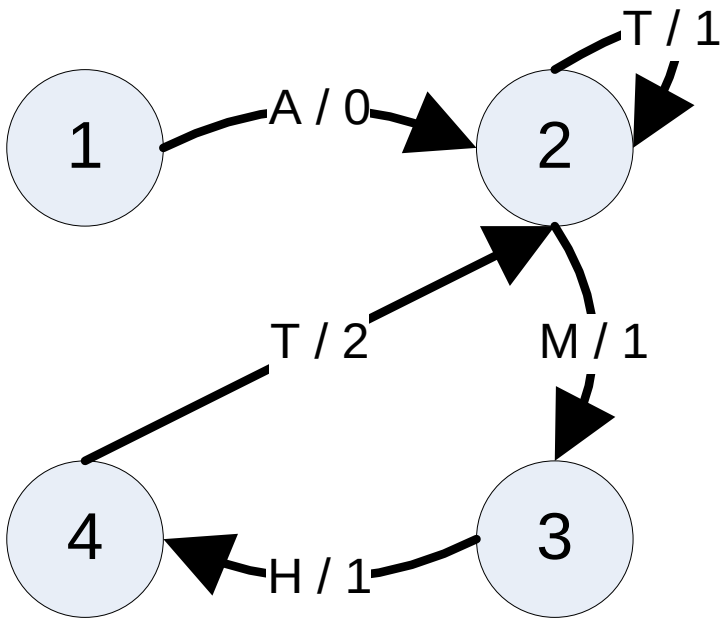
Схема работы верификатора

Тестовые примеры и сценарии работы

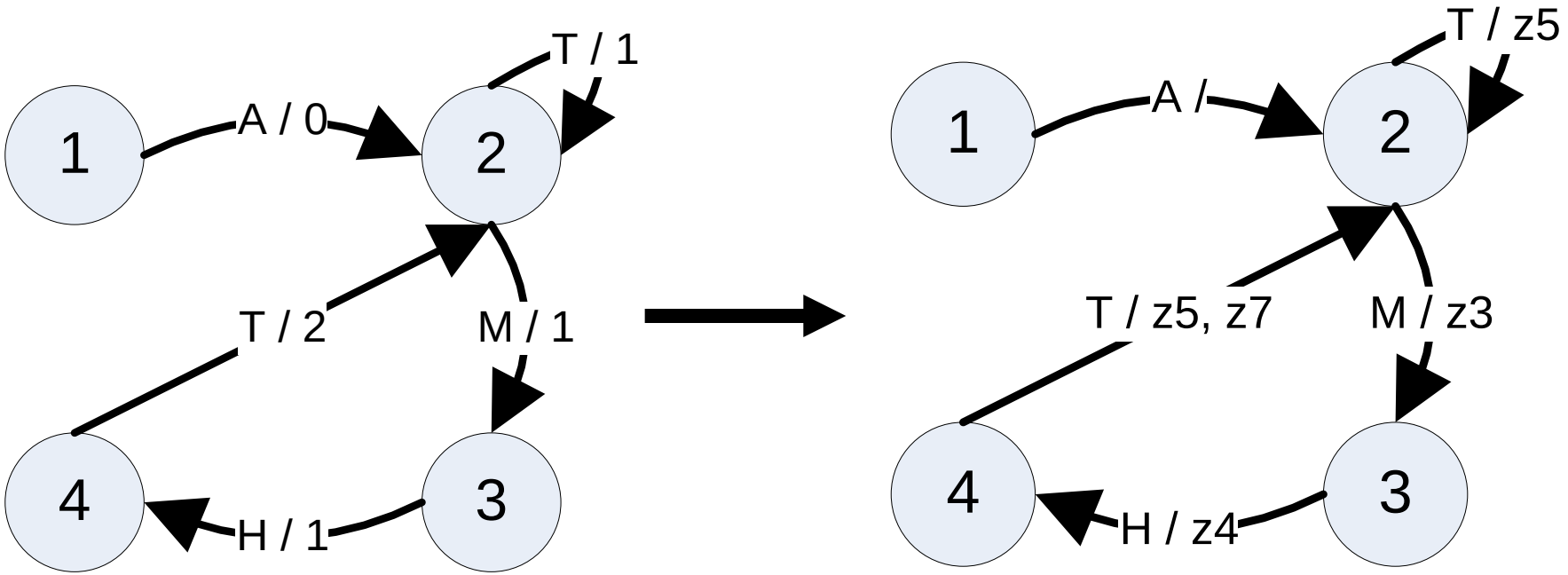
- Тестовый пример:
  - входная последовательность событий *Input*
  - соответствующая ей последовательность выходных воздействий *Output*
- Позитивный сценарий работы:
  - последовательность пар: событие – выходные воздействия $<e_i, <z_{i,1}, z_{i,2} \ldots z_{i,k}>>$
- Негативный сценарий работы:
  - последовательность событий *Input*, которая никогда не должна выполняться

Особь алгоритма генетического программирования

- Списки переходов для каждого состояния + номер начального состояния
- Для каждого перехода хранится событие, по которому он выполняется, и число выходных воздействий, но не хранятся выходные воздействия
- Для сценариев число выходных воздействия не хранится



Алгоритм расстановки пометок



Функция приспособленности

Учет поведения автомата на тестах →

$$FF_1 = \frac{\sum_{i=1}^{n} \left(1 - \frac{ED(\ Output\ [i],\ Answer\ [i])}{\max(|\ Output\ [i]\ |, |\ Answer\ [i]\ |)}\right)}{n}$$

$$FF = FF_1 + FF_1 \cdot \frac{n_1}{n_2} - FF_1 \cdot \frac{k_1}{k_2} + \frac{M - cnt}{10 \cdot M}$$

- ED – редакционное расстояние
- n_1 – число верных формул
- n_2 – общее число формул
- k_1 – число выполняющихся негативных сценариев
- k_2 – общее число негативных сценариев
- M – число, большее максимального числа переходов

Функция приспособленности

Учет поведения автомата на тестах

$$FF_1 = \frac{\sum_{i=1}^{n} \left(1 - \frac{ED(\ Output\ [i],\ Answer\ [i])}{\max(|\ Output\ [i]\ |, |\ Answer\ [i]\ |)}\right)}{n}$$

$$FF = FF_1 + FF_1 \cdot \frac{n_1}{n_2} - FF_1 \cdot \frac{k_1}{k_2} + \frac{M - cnt}{10 \cdot M}$$

- ED – редакционное расстояние
- n_1 – число верных формул
- n_2 – общее число формул
- k_1 – число выполняющихся негативных сценариев
- k_2 – общее число негативных сценариев
- M – число, большее максимального числа переходов

Учет числа переходов

Функция приспособленности

Учет поведения автомата на тестах

$$FF_1 = \frac{\sum_{i=1}^{n} \left(1 - \frac{ED(\ Output\ [i],\ Answer\ [i])}{\max(|\ Output\ [i]\ |, |\ Answer\ [i]\ |)}\right)}{n}$$

$$FF = FF_1 + FF_1 \cdot \frac{n_1}{n_2} - FF_1 \cdot \frac{k_1}{k_2} + \frac{M - cnt}{10 \cdot M}$$

- ED – редакционное расстояние
- n_1 – число верных формул
- n_2 – общее число формул
- k_1 – число выполняющихся негативных сценариев
- k_2 – общее число негативных сценариев
- M – число, большее максимального числа переходов

Учет LTL-формул

Функция приспособленности

Учет поведения автомата на тестах

$$FF_1 = \frac{\sum_{i=1}^{n} \left(1 - \frac{ED(Output[i], Answer[i])}{\max(|Output[i]|, |Answer[i]|)}\right)}{n}$$

$$FF = FF_1 + FF_1 \cdot \frac{n_1}{n_2} - FF_1 \cdot \frac{k_1}{k_2} + \frac{M - cnt}{10 \cdot M}$$

- ED – редакционное расстояние
- n_1 – число верных формул
- n_2 – общее число формул
- k_1 – число выполняющихся негативных сценариев
- k_2 – общее число негативных сценариев
- M – число, большее максимального числа переходов

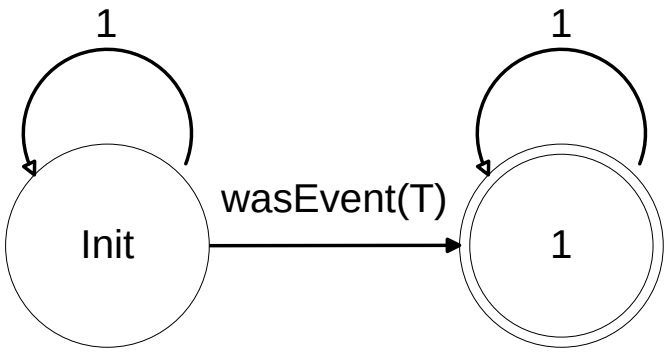
Учет негативных сценариев работы

Учет верификации в функции приспособленности

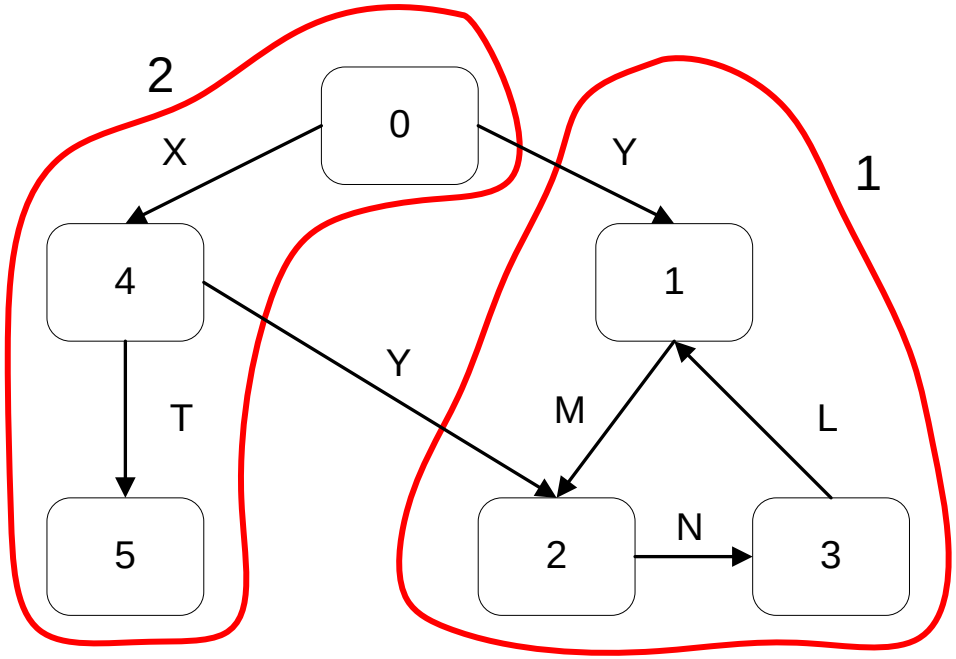
$$FF_{LTL} = \sum_{i=1}^{n} \frac{t_i}{T_i} \Big/ n$$

- n – число *LTL*-формул
- T_i – число достижимых переходов в автомате
- t_i – число «проверенных» переходов – переходы, на которых *LTL*-формула выполняется

Пример учета верификации в функции приспособленности



Автомат Бюхи, построенный по отрицанию формулы **G(!wasEvent(T))**



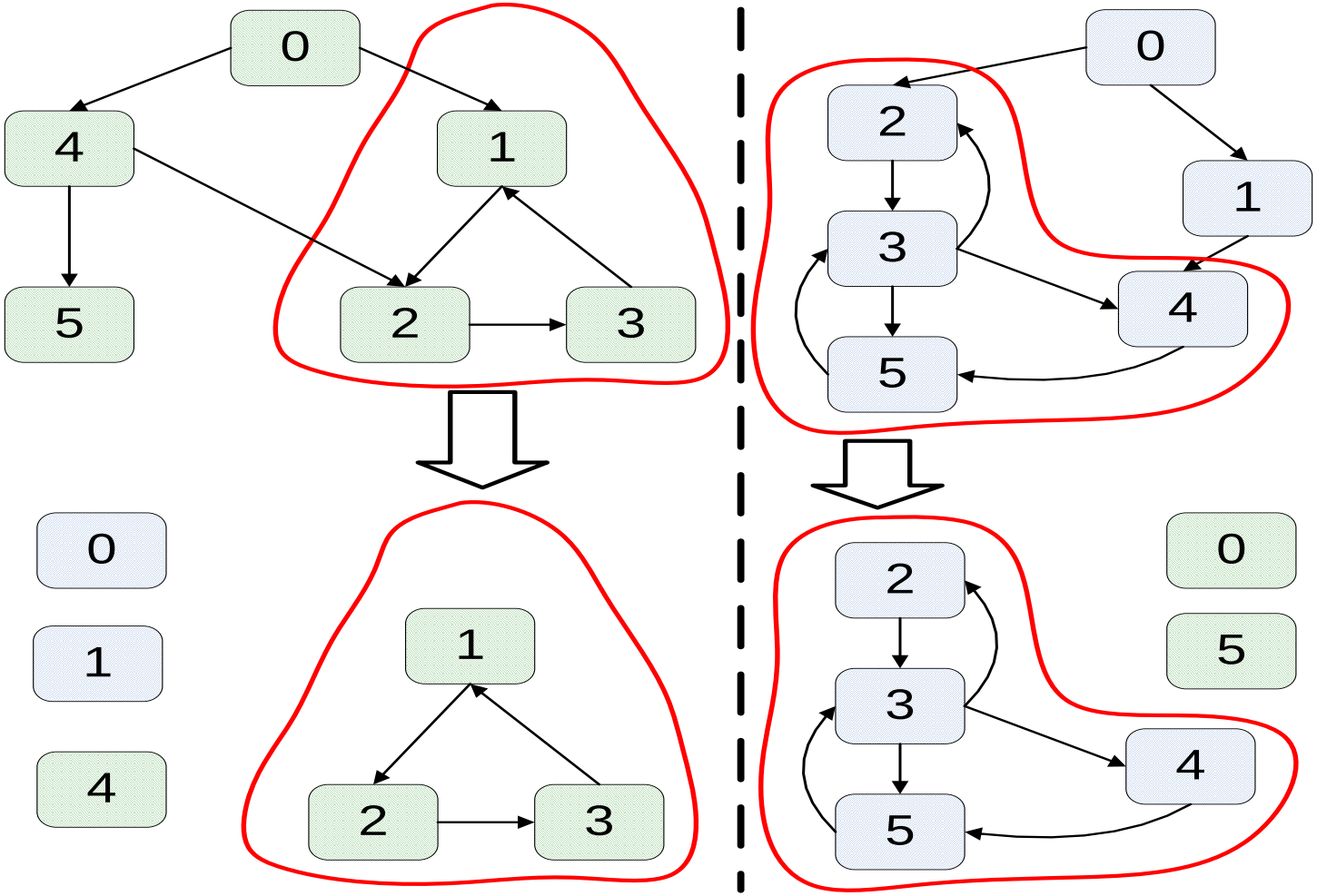
$$FF_i = \frac{3}{7}$$

- 3 – число «проверенных» переходов из первого подмножества
- 7 – число переходов в конечном автомате

Учет результата верификации при скрещивании и мутации

- Верификатор помечает путь в модели, опровергающий формулу
  - Удалить переход из контрпримера
  - Изменить входное воздействие, число выходных или состояние в которое перейдет автомат
- Когда верификатор покидает вершину при обходе в глубину, он помечает все исходящие переходы
  - Такие переходы соответствуют *LTL*-формуле
  - Подграф из таких переходом может перейти в новую особь без изменений при скрещивании

Пример скрещивания с учетом верификации



Программирование по контрактам

- *Предусловие* – ожидание метода объекта на входные параметры и состояние класса при его вызове
- *Постусловие* – обязательства метода при его завершении
- *Инвариант* – условие выполняющиеся на протяжении всего времени жизни экземпляра класса

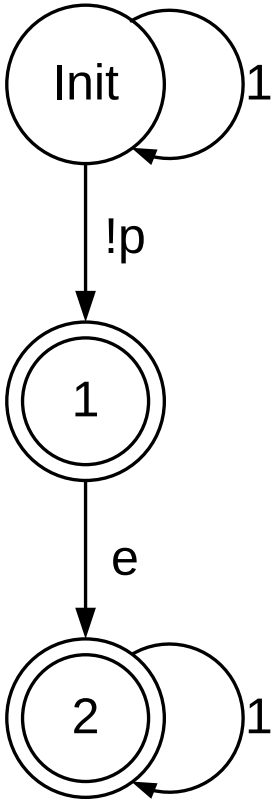
Автоматные контракты

- Контракты на состояния
- Контракты на переходы

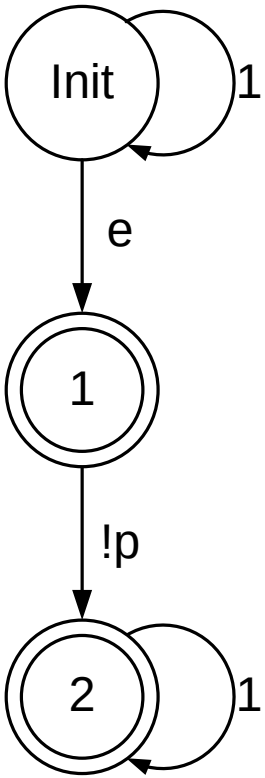
- Предусловие – $G(Xp_1 \rightarrow p_2)$
- Постусловие – $G(p_1 \rightarrow Xp_2)$
- Инвариант – $G(p_1 \rightarrow p_2)$

p_i – *утверждение о переходе или о состоянии.*

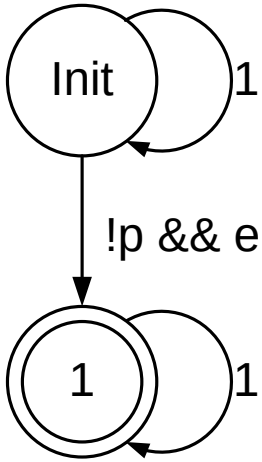
Автоматы Бюхи для отрицания контрактов



предусловие
G(Xe → p)



постусловие
G(e → Xp)



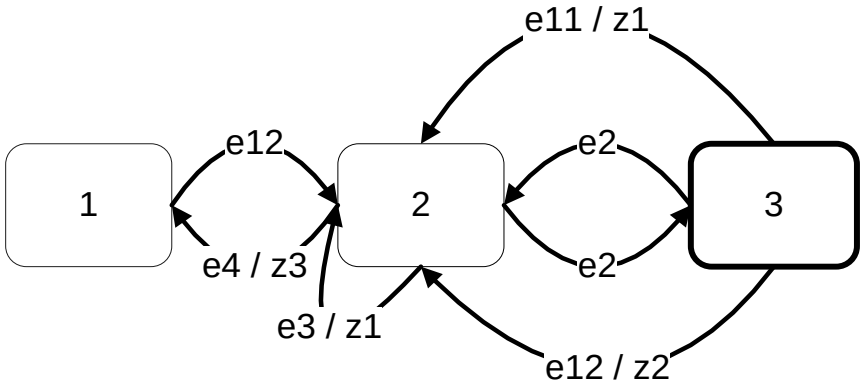
инвариант
G(e → p)

Пример – система управления дверьми лифта

- Пять событий, три выходных воздействия
- Девять тестов
- Одиннадцать *LTL*-формул:
  - **G**(wasEvent(e11) => wasAction(z1)) – начать открытие дверей после того, как нажата кнопка «Открыть двери»
  - **G**(wasEvent(e4) <=> wasAction(z3))
  - **G**(wasEvent(e3) => wasAction(z1))
  - **G**(wasEvent(e11) => **X**(wasEvent(e4) **or** wasEvent(e2))) – если нажали копку «Открыть двери», то следующим событием будет либо e2 (открытие дверей успешно завершено) или e4 (двери сломались)
  - **G**(wasAction(z1) => **X**(!wasAction(z1) **U**(wasAction(z2) **or** wasEvent(e4))))

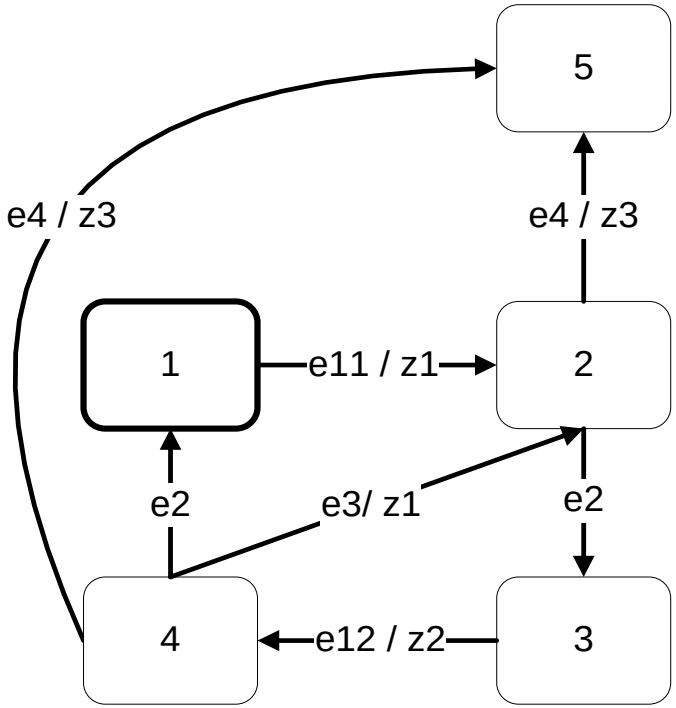
Построенные автоматы

Только тесты



После поломки дверей может быть отдана команда на их закрытие!

Тесты + верификация



Экспериментальные исследования

- Измерялось число вычислений функции приспособленности
- 1000 экспериментов

	Тесты и *LTL*-формулы	Тесты, *LTL*-формулы и контракты	Сценарии, *LTL*-формулы и контракты
Среднее значение	8.372×10^5	7.109×10^5	1.616×10^5
Среднеквадратичное отклонение	7.57×10^5	6.32×10^5	1.102×10^5
Минимальное значение	6.331×10^4	6.153×10^4	3.808×10^4
Максимальное значение	5.912×10^6	4.589×10^6	8.162×10^5

Технология генерации автоматов на основе генетического программирования и верификации

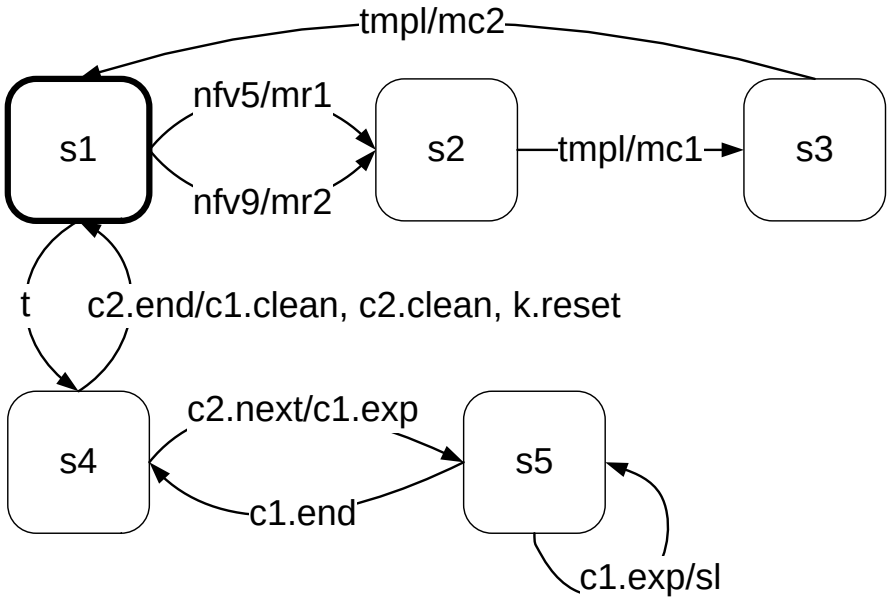
Инструментальное средство для поддержки технологии

- Исходный код размещен в открытом доступе в сети Интернет по адресу: http://code.google.com/p/gabp/
- Реализует разработанные методы и технологию построения конечных автоматов
- Входные данные – описание тестов, сценариев, *LTL*-формул и контрактов в *XML*-формате
- Выходные данные – описание конечного автомата в *XML*-формате инструментального средства *UniMod*

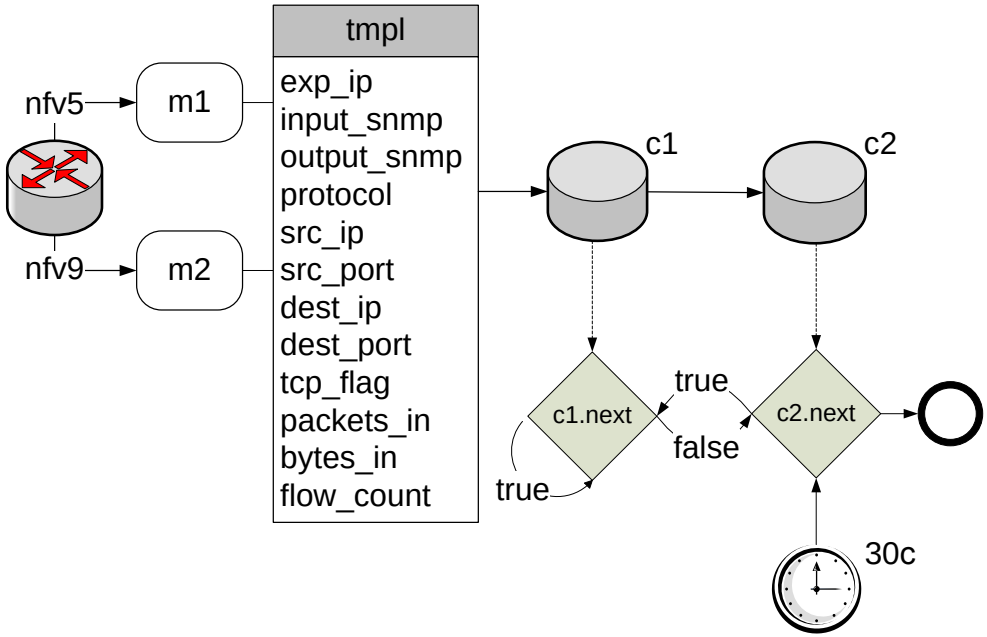
Построение модуля *Top Traffic Monitor* (1)

- *NetFlow Integrator* – программный продукт для мониторинга сети и поиска потенциальных сетевых атак или угроз
- *Top Traffic Monitor* – поиск узлов сети с максимальным трафиком
- Граф потока управления и данных (*Control and Data Flow Graph, CDFG*) – вершины выполняют операции над *NetFlow*-сообщениями, а ребра бывают двух типов:
  - *Control Flow* – последовательность выполнения операций
  - *Data Flow* – связь между источниками и приемниками данных

Построение модуля *Top Traffic Monitor* (2)



Управляющий автомат



Граф потока управления и данных

Результаты работы (1)

- Предложена функция приспособленности, учитывающая верификацию
- Разработаны операции мутации и скрещивания, учитывающие верификацию
- Разработан алгоритм генерации автоматов с учетом контрактов
- Разработан алгоритм генерации автоматов по сценариям работы и темпоральным формулам
- Разработан верификатор *AutomataVerificator*, приспособленный для поддержки генерации автоматов на основе генетического программирования
- Разработана технология генерации автоматов с помощью генетического программирования и верификации
- На основе разработанных верификатора и технологии создано инструментальное средство *GABP* для генерации автоматов с помощью генетического программирования и верификации

Результаты работы (2)

- Результаты, полученные в диссертации, были внедрены при построении модуля *Top Traffic Monitor* для определения узлов сети с максимальным трафиком в программном продукте *NetFlow Integrator,* выпускаемом компанией ООО *ЭВЕЛОПЕРС* (Санкт-Петербург)
- Результаты работы используются в учебном процессе на кафедре «Компьютерные технологии» НИУ ИТМО в курсе «Автоматное программирование»
- Опубликовано четыре статьи, из которых три в журналах из перечня ВАК
- Сделано 7 докладов на конференциях, в том числе, один доклад на международной конференции *GECCO*

Спасибо за внимание!

Спасибо за внимание!