

Автоматное программирование

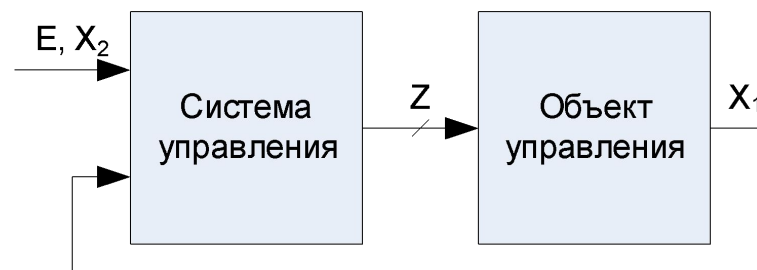
А.А. Шалыто

Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

2007 г.

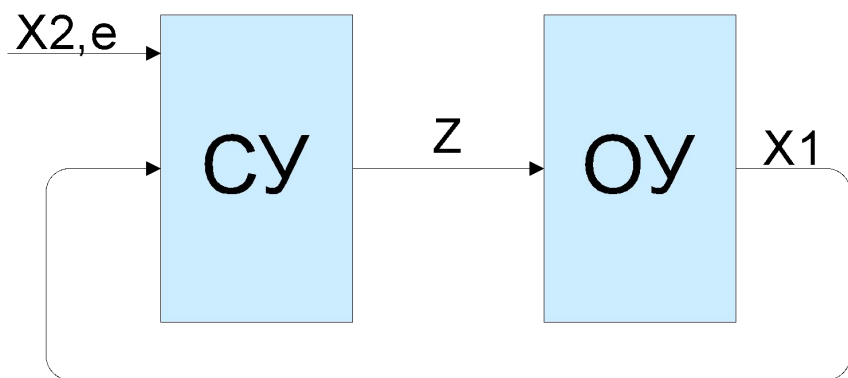
Автоматное программирование

- Предложено мною в 1991 году
- Программные системы предлагается разрабатывать так же, как выполняется автоматизация технологических (и не только) процессов
- Система управления является системой взаимодействующих конечных автоматов

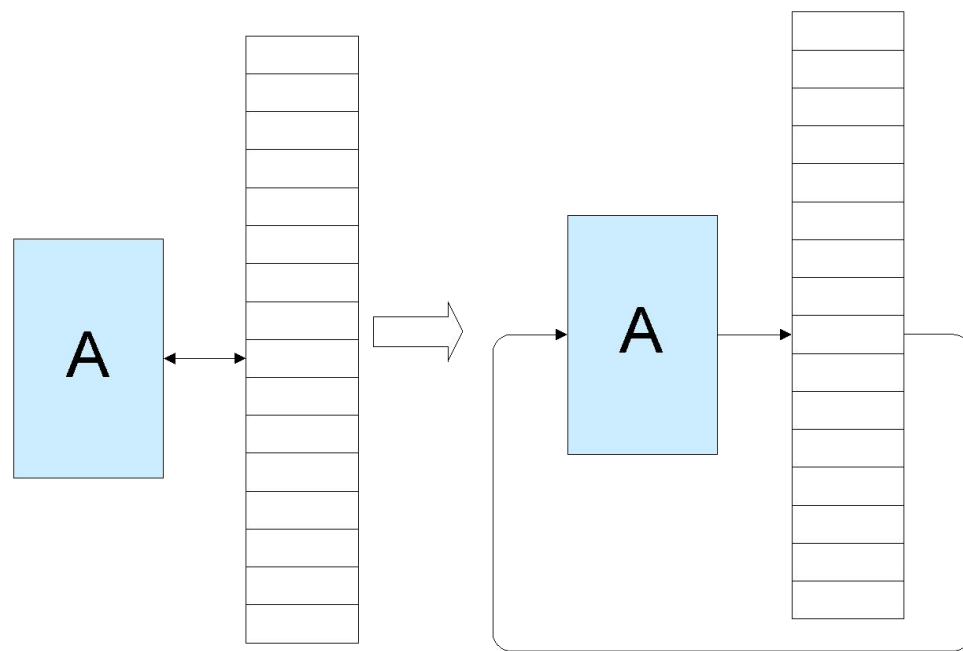


- Состояния
- События и входные переменные
- Выходные воздействия
- Конечный автомат
- Система конечных автоматов

Автоматное программирование

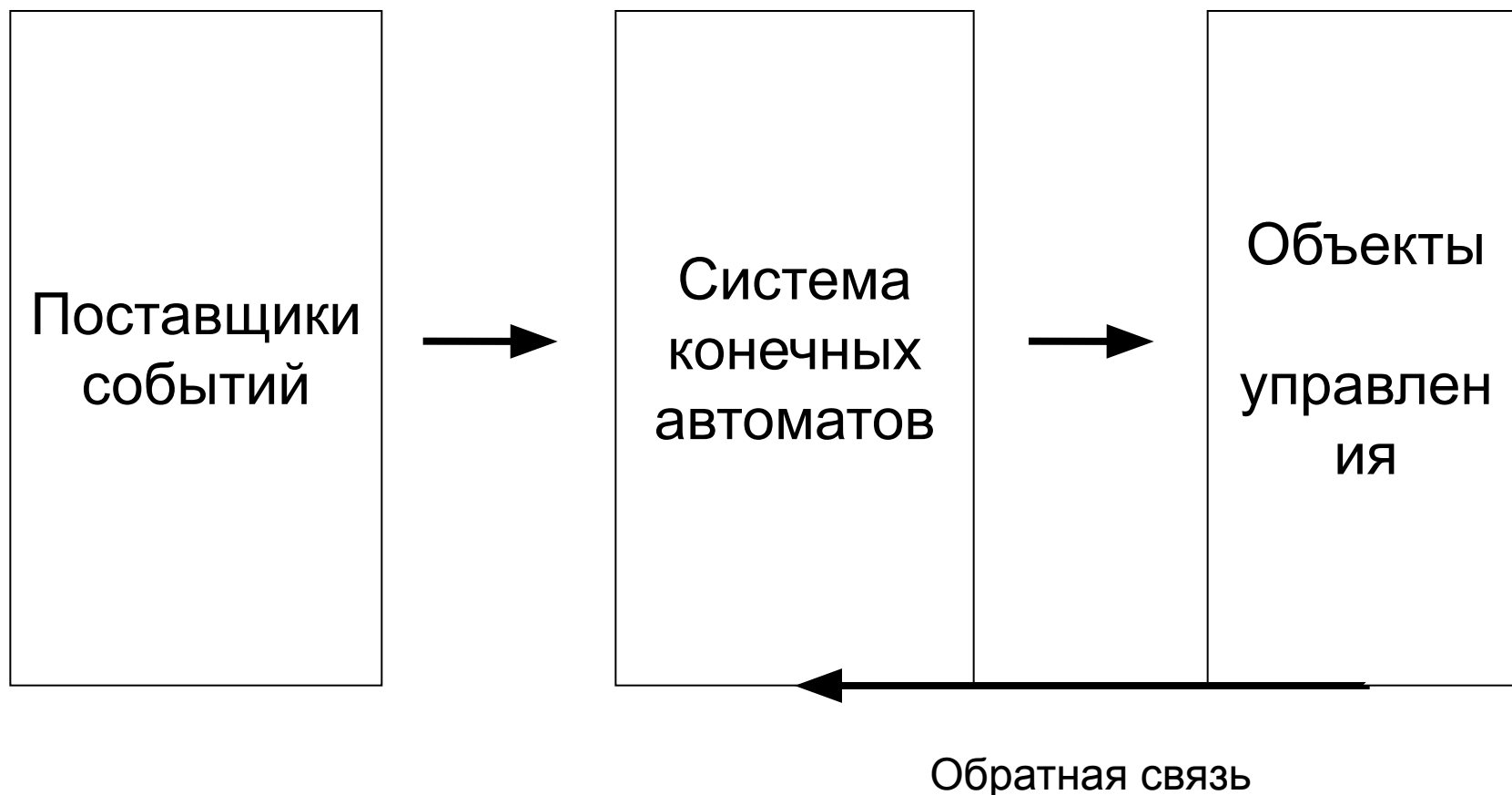


Система управления



Машина Тьюринга

Как строить программы?



- Обладает наибольшей эффективностью для систем со сложным поведением
- Формальное и понятное описание поведения
- Автоматическая генерация кода по диаграммам переходов
- Возможность верификации программ
- Проектная документация

- Программное обеспечение
 - Системы высокой надежности
 - Военные приложения
 - Аэрокосмическая индустрия
 - Встроенные системы
 - Мобильные системы
 - Визуализаторы
 - Web-приложения
 - Клиент-сервер приложения
- «Железо»
 - Микропроцессоры
 - Микроконтроллеры
 - Программируемые логические контроллеры
- Парадигмы
 - Процедурная
 - Объектно-ориентированная
 - Языки контроллеров
 - Лестничные схемы
 - Функциональные схемы

Инструментальное средство *UniMod (1)*

- Инструментальное средство для поддержки автоматного программирования
- Создано в рамках ФЦНТП «Исследования и разработки по приоритетным направлениям развития науки и техники» на 2002-2006 годы по приоритетному направлению «Информационно-телекоммуникационные системы»
- Критическая технология – «Технология производства программного обеспечения»
- Вошел в число 15 наиболее инновационно перспективных и социально значимых проектов Федерального агентства по науке и инновациям

- Локальная и удаленная отладка диаграмм в терминах состояний
- Проверка формальных свойств диаграмм
- Интерпретируемый и компилируемый подходы
- Запись автоматов в нотации *UML*-диаграмм классов и состояний
- Встраиваемый редактор *UML*-диаграмм для платформы *Eclipse*
- Запуск диаграмм в «одно нажатие»

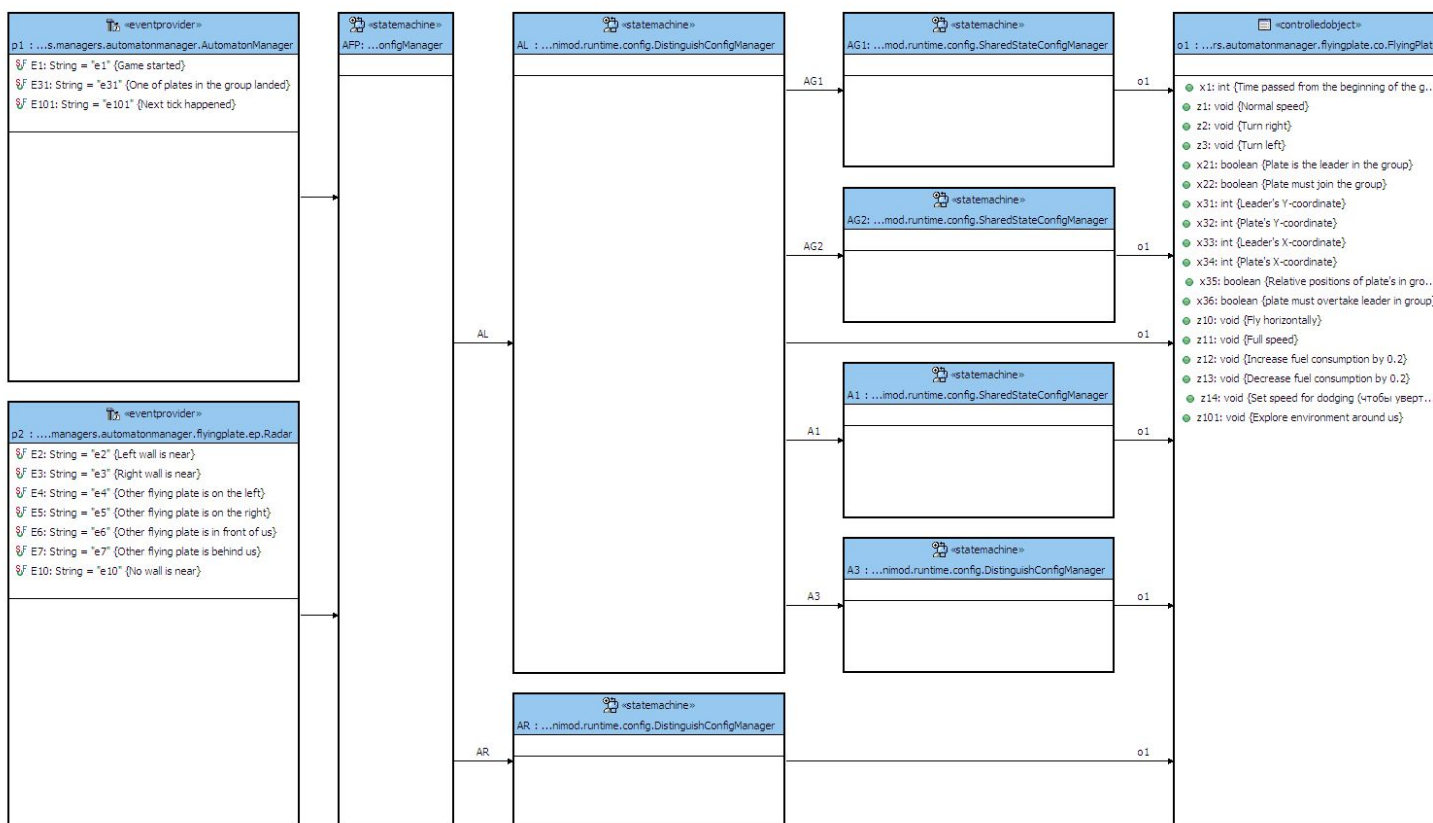
Инструментальное средство *UniMod (3)*

Семь автоматов

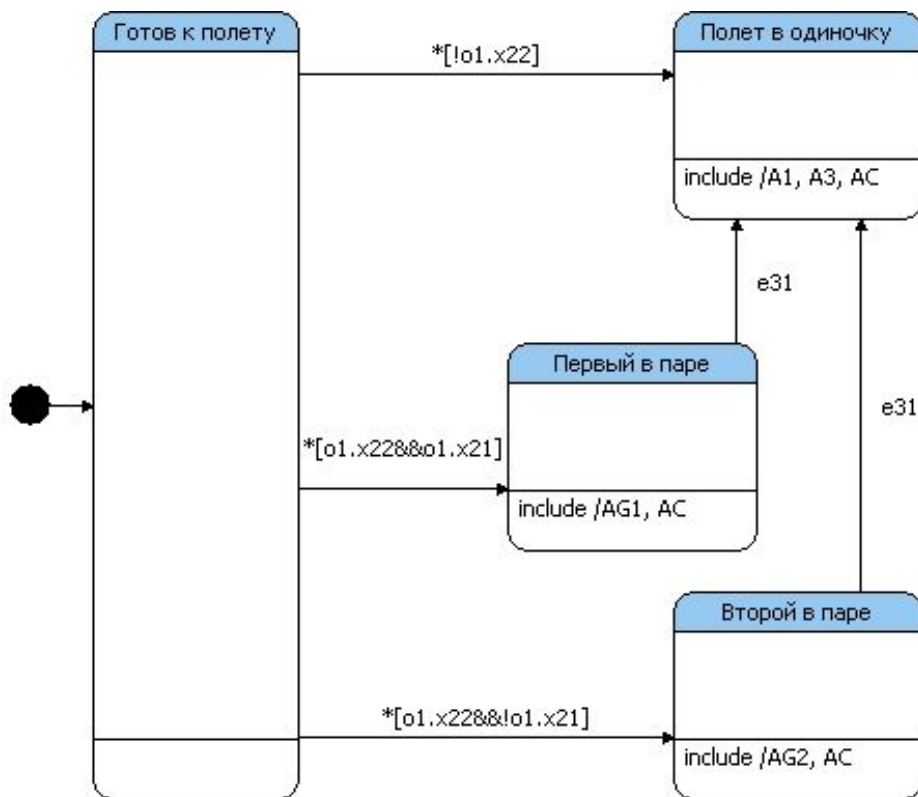
Вручную

Автоматическая генерация

Вручную



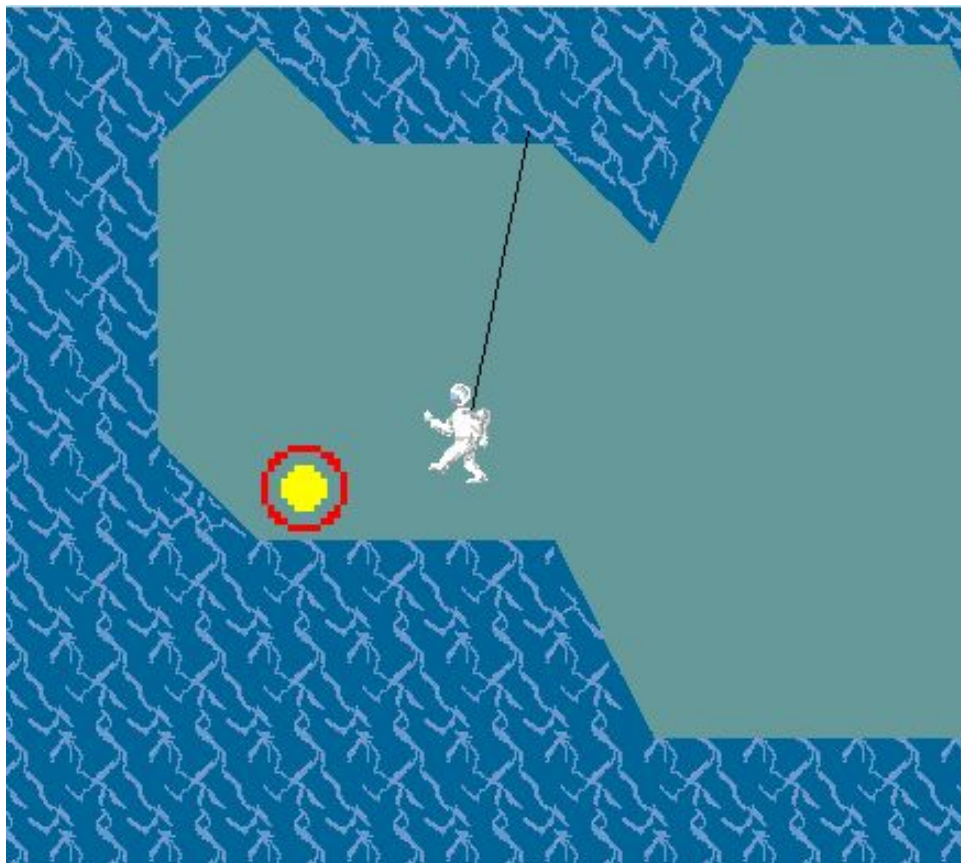
Один из автоматов – AL



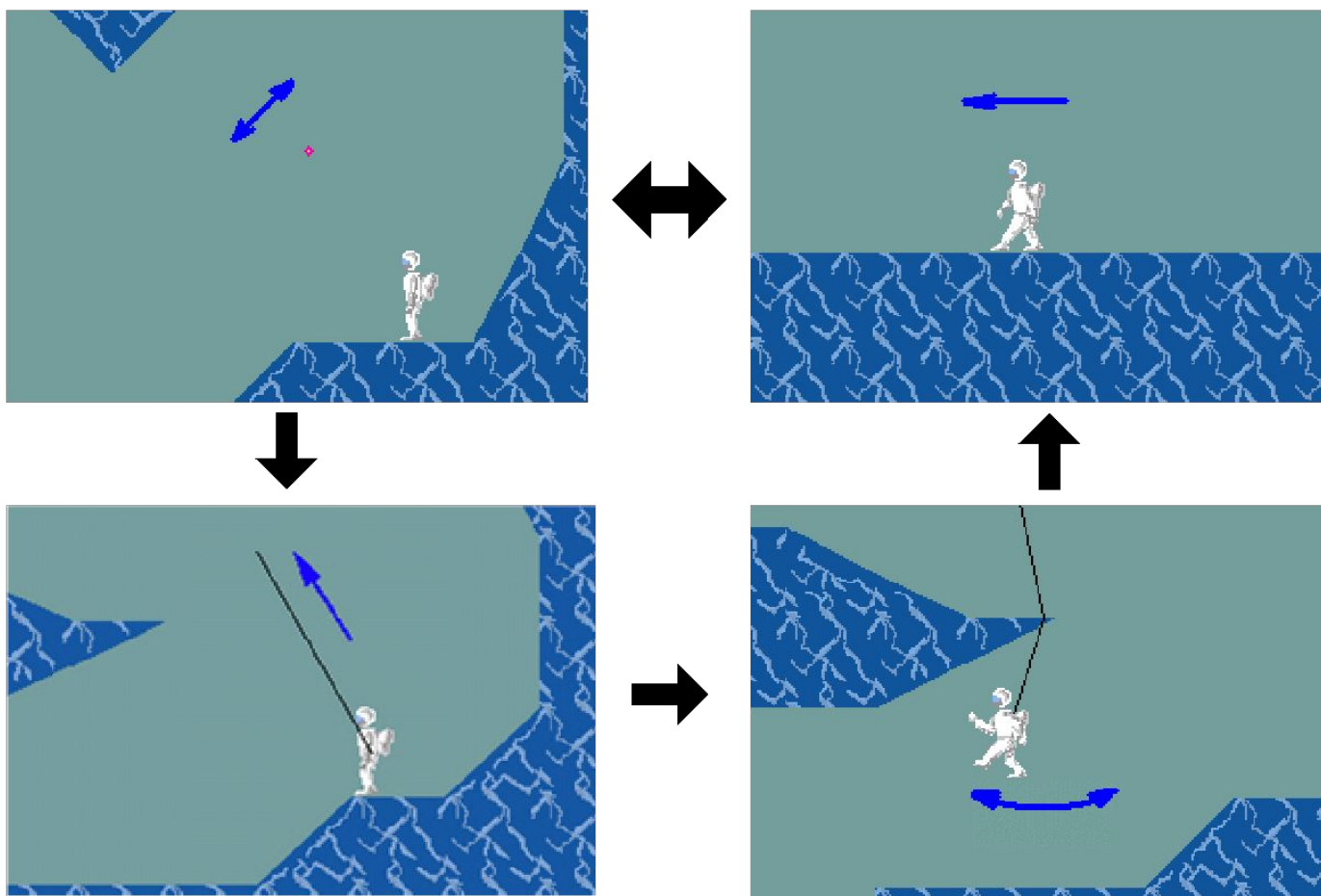
Движение за открытую проектную документацию

- Три задачи:
 - Повышается качество обучения – обучение на проектах
 - Создаются предпосылки для научной работы и отбор «ученых»
 - Совершенствуется технология автоматного программирования
- Создано более 100 студенческих проектов, содержащих не только программную часть, но и открытую проектную документацию
- Из них – 15 UniMod-проектов
- Проекты опубликованы на сайте <http://is.ifmo.ru>

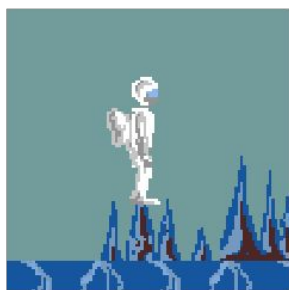
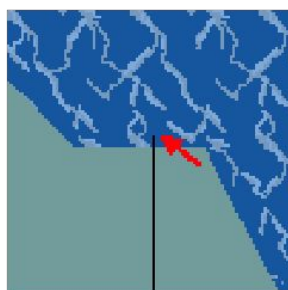
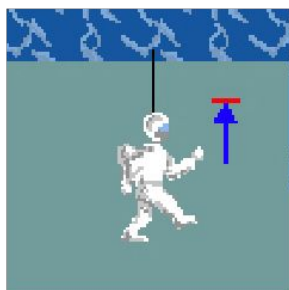
Примеры. Игра «Космонавт» (1)



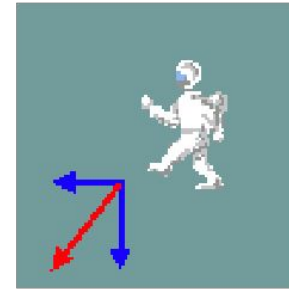
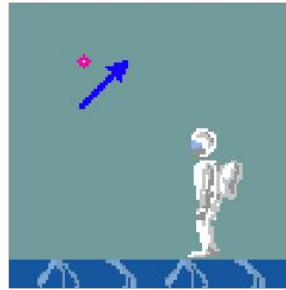
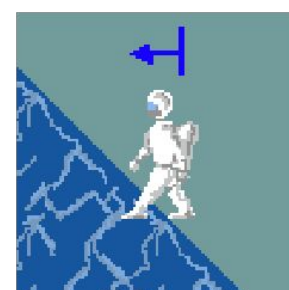
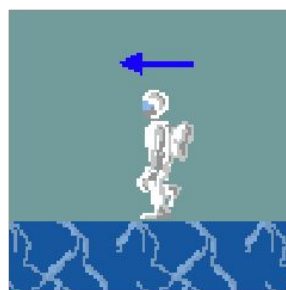
Примеры. Игра «Космонавт» (2)



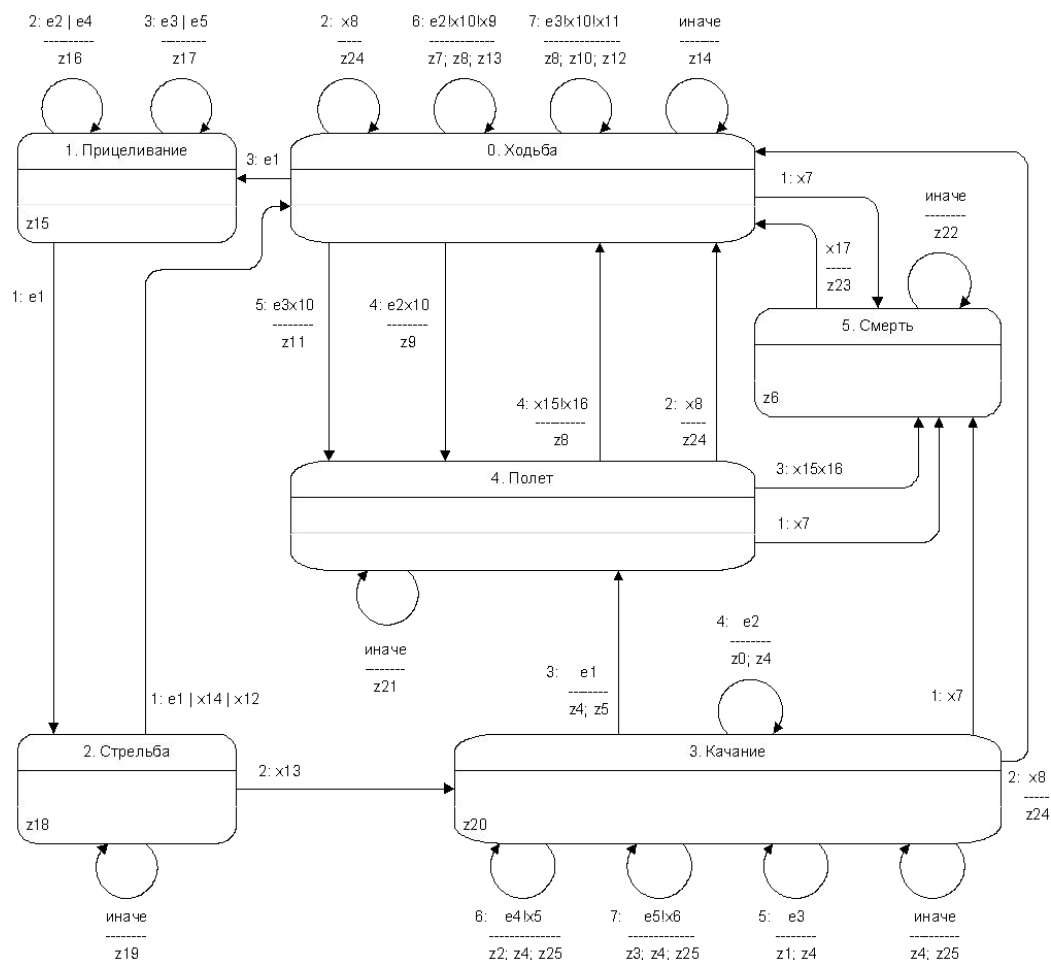
Входные воздействия



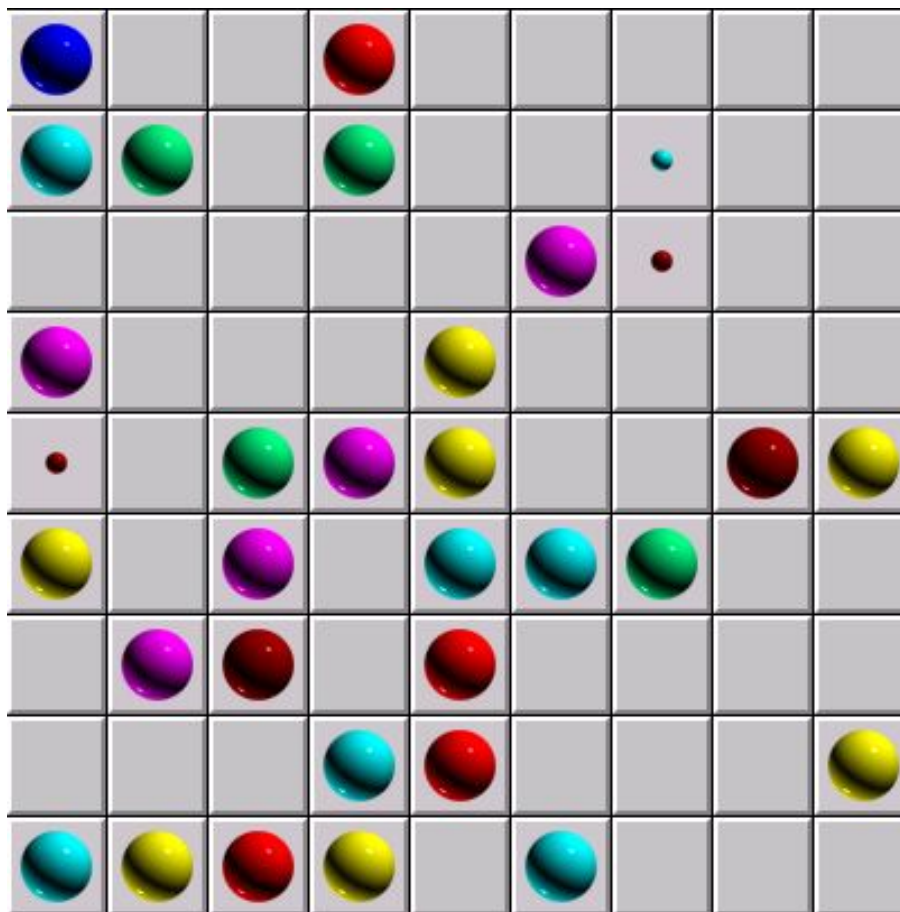
Выходные воздействия



Примеры. Игра «Космонавт» (4)

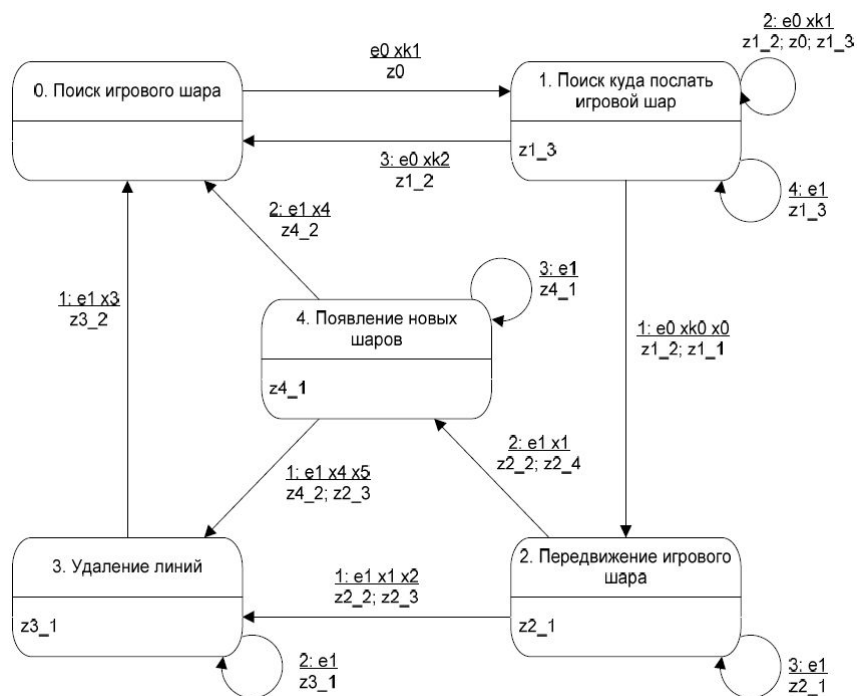


Примеры. Игра «Lines» (1)

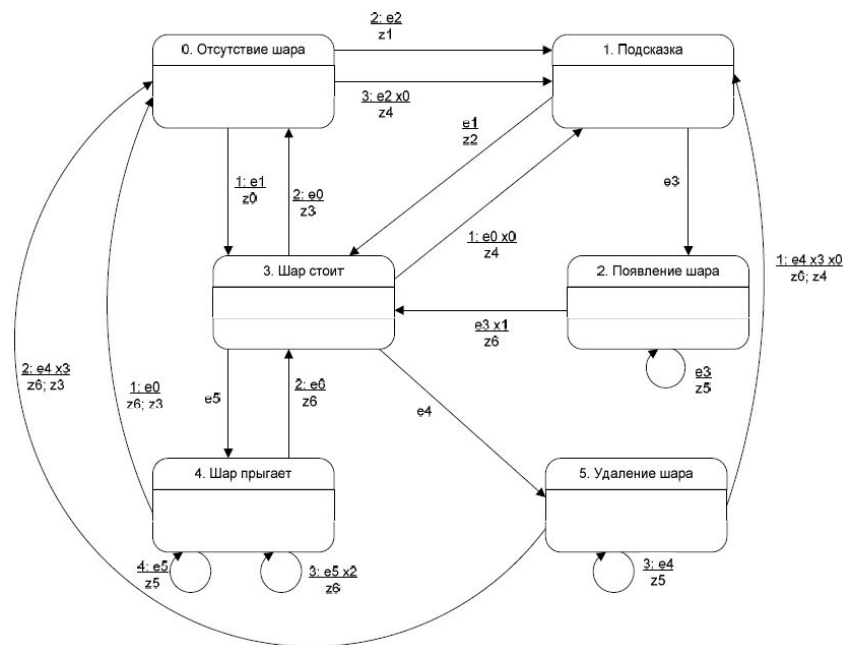


Примеры. Игра «Lines» (2)

Управление игрой



Управление клеткой



Новые направления в автоматном программировании

- В рамках Федеральной целевой программы *«Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2012 годы»*
- Технология генетического программирования для генерации автоматов управления системами со сложным поведением (шифр «2007-4-1.4-18-01-033»)
- Разработка технологии верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода (шифр «2007-4-1.4-18-02-041»)

Генерация автоматов и генетическое программирование

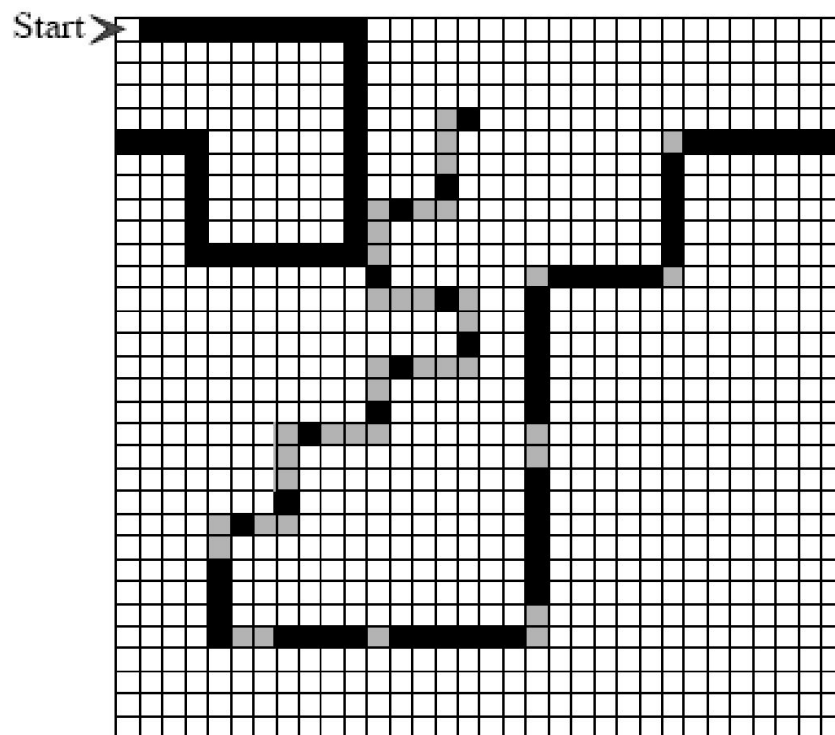
- Основная сложность в автоматном программировании – построение автоматов
- В большинстве случаев автоматы проектируются вручную
- Однако эвристическое построение автоматов часто затруднено или невозможно
- Решение – автоматическое построение конечных автоматов с помощью генетического программирования
- Это позволит повысить уровень автоматизации построения программ рассматриваемого класса
- Материалы – на сайте <http://is.ifmo.ru> (раздел «Генетические алгоритмы»)

Три рассматриваемые задачи

- «Простая» задача – задача об «Умном муравье»
- «Сложная» задача – задача «Беспилотные летательные объекты»
- «Народная» задача – «Разливочная линия»

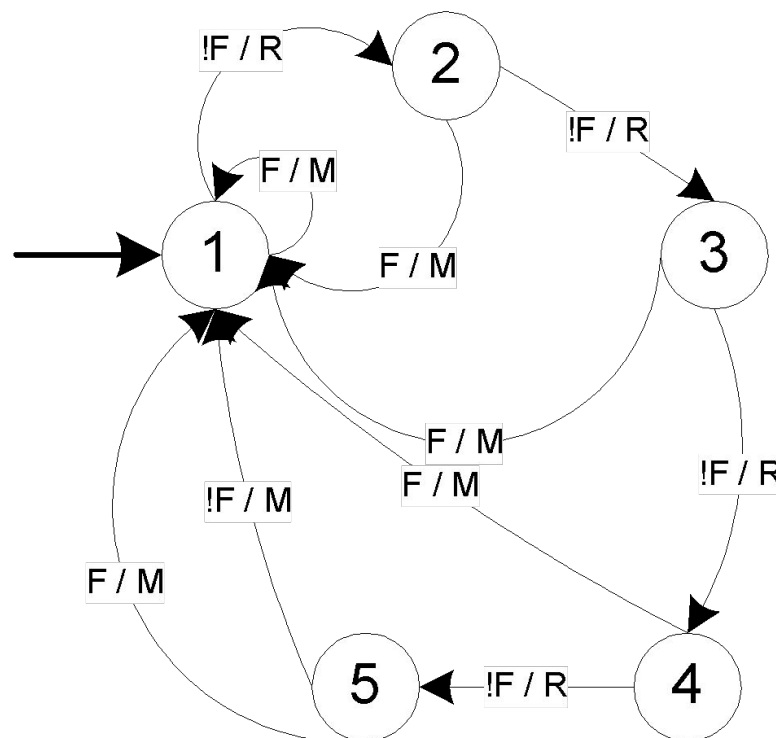
«Простая» задача – задача об «Умном муравье»

- Тор – 32x32
- 89 клеток с едой
- 200 ходов
- Расположение еды и начальная позиция муравья фиксированы
- Цель – создать муравья, который съест всю еду
- Муравей = конечный автомат



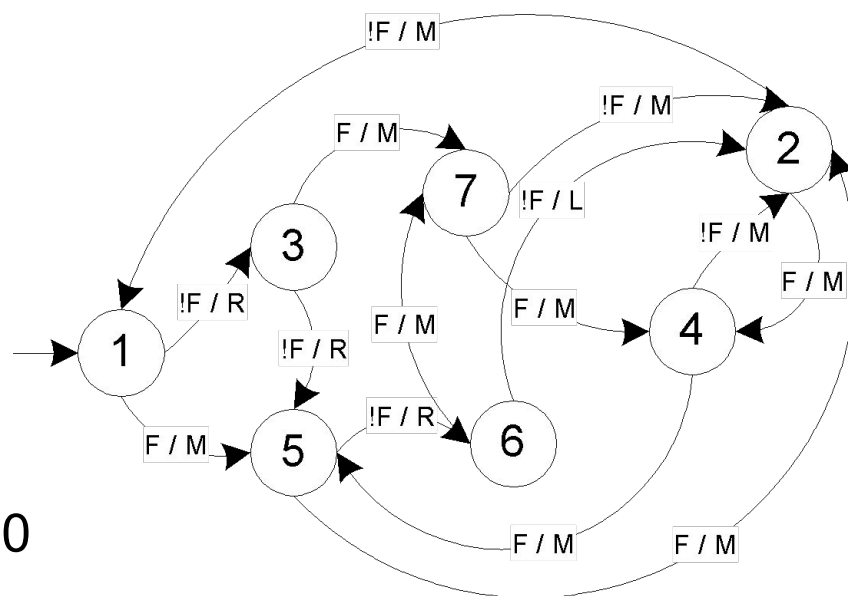
Эвристическое построение задачу не решает

- Пять состояний, за 200 ходов съедается 81 единица еды или все 89 единиц еды за 314 ходов



Решение «простой» задачи

- Известные подходы – кодирование битовыми строками + генетический алгоритм
- Известные решения:
 - 13 состояний (1992)
 - 11 состояний (1993)
 - 8 состояний (1999)
- Предлагаемый подход – генетическое программирование
- Построены два автомата с **7 состояниями** после генерации 160 и 250 млн. автоматов
- Полный перебор $\sim 3 \cdot 10^{18}$ автоматов



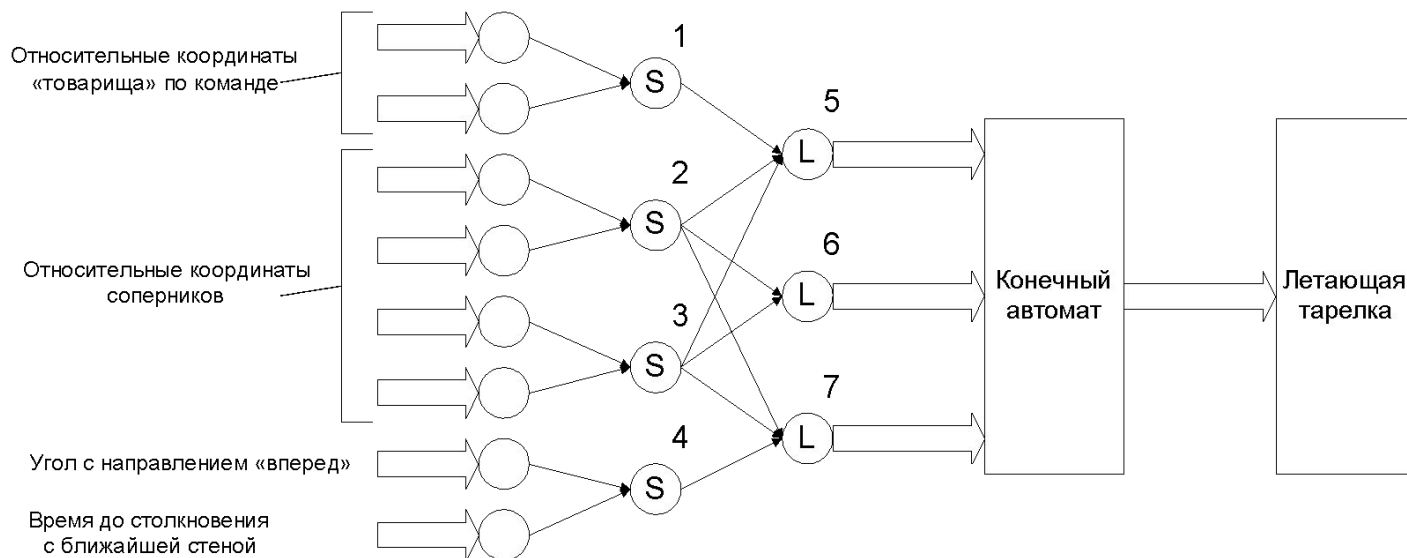
«Сложная» задача – задача «Беспилотные летательные объекты»

- Соревнование на дальность полета
- Две команды по восемь объектов
- Ограничения: запас топлива, столкновения, аэродинамическое взаимодействие
- Цель – разработка управляющей программы
- Задача заочного тура VI Открытой Всесибирской олимпиады по программированию (2005 год)
- Была решена при участии автора путем эвристического построения автоматов

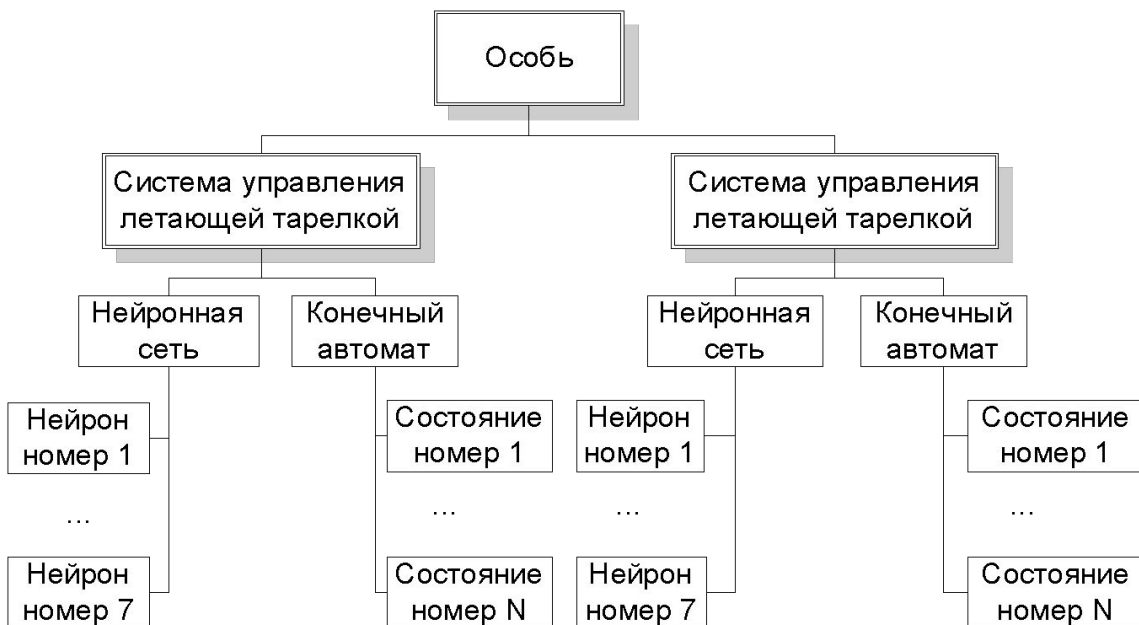
<http://is.ifmo.ru/unimod-projects/plates/>

Решение (1)

- Система управления = нейронная сеть + конечный автомат
- Нейронная сеть преобразует вещественные входные переменные в логические



- Особь = две системы управления беспилотным объектом
- Особь из двух систем – для учета взаимодействия объектов



- Мутация особи
 - Мутация системы управления летательным объектом
 - Мутация нейронной сети
 - Мутация элемента сети
 - Мутация конечного автомата
 - Изменение начального состояния
 - Мутация перехода
- Скрещивание особей
 - Скрещивание систем управления летающей тарелкой
 - Скрещивание автоматов
 - Скрещивание нейронных сетей

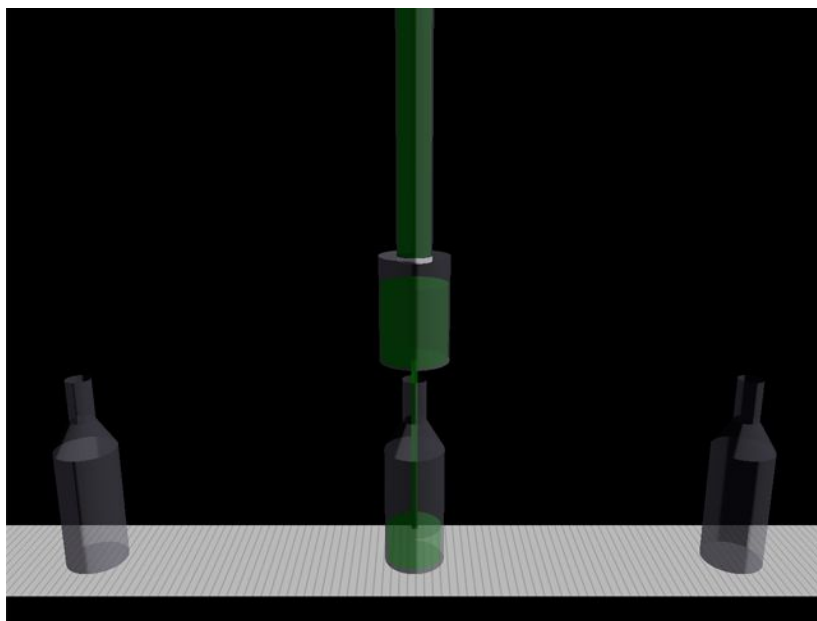
Результаты применения генетического программирования

- За сутки была построена управляющая система, содержащая нейронную сеть и один автомат с шестью состояниями (вместо семи автоматов с 21 состоянием)

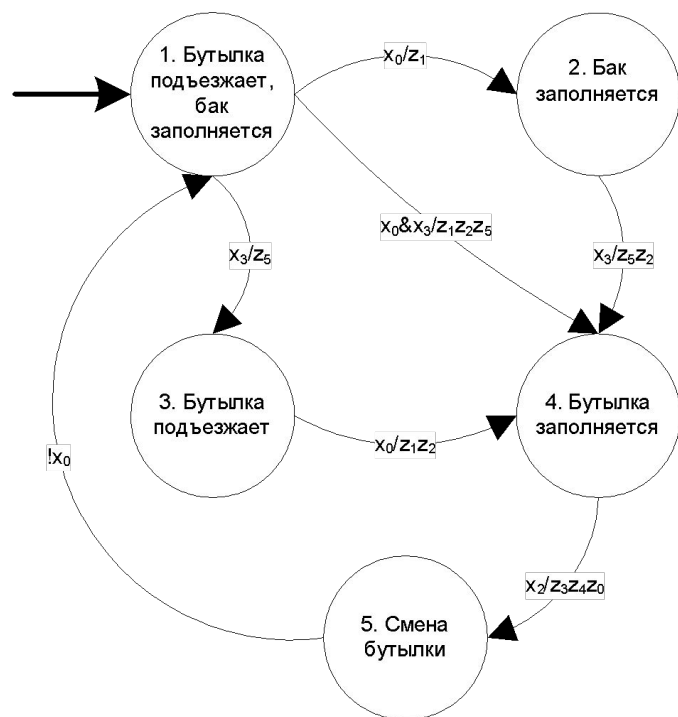
	Построенная с помощью ГП система	Построенная вручную система
Среднее	216,55	212,59
Минимум	203,05	203,44
Максимум	241,11	225,09

«Народная» задача – «Разливочная линия»

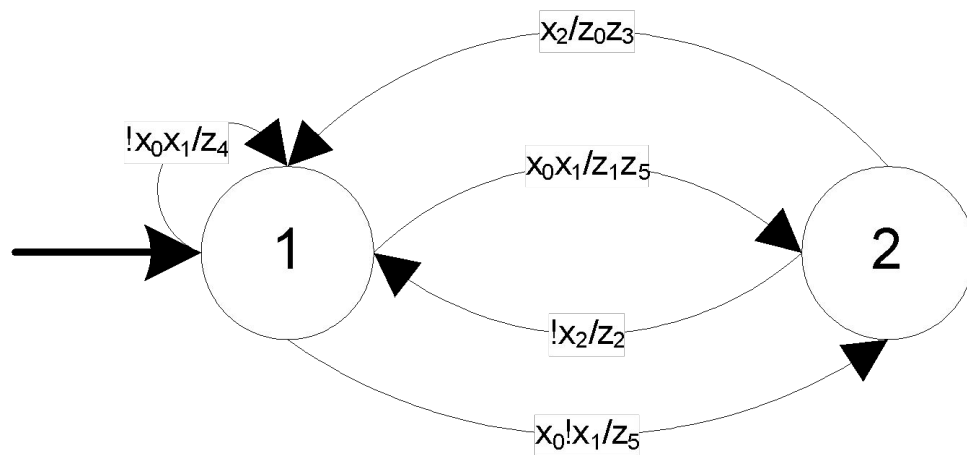
Задача: налить как можно больше бутылок
за заданный промежуток времени



Решения «народной» задачи



Построен вручную



Построен автоматически

Три традиционных подхода к верификации программ

- Тестирование – ничего не доказывает
- Доказательства теорем – все доказывает, но практически нигде не используется
- Верификация на моделях – *Model Checking* (модель программы с конечным числом состояний и свойства программы в темпоральной логике)

Недостатки *Model Checking* для программ общего вида

- Не ясно, как построить модель
- Не ясно, как записать темпоральные свойства
- Не ясно, как перейти от модели к реальной программе в случае обнаружения ошибки

Верификация автоматных программ

- Так как поведение автоматных программ задается графами переходов конечных автоматов, то все три указанные проблемы для этого класса программ эффективно решаются
- Работы проводятся совместно с кафедрой «Теоретическая информатика» Ярославского государственного университета им. П.Г. Демидова
- Материалы – на сайте <http://is.ifmo.ru> (раздел «Верификация»)

Повышение качества программного обеспечения

- NASA (Открытые системы #03/2004)
 - Использование состояний
 - Генерация программ с помощью инструментального средства *Stateflow*
 - Верификатор *SPIN*
- СПбГУ ИТМО <http://is.ifmo.ru>
 - Использование состояний
 - Генерация программ с помощью инструментального средства *UniMod*
 - Верификаторы *SPIN* и *Bogor*

- Есть области, в которых верификация необходима: авиация, управление ядерными реакторами
- Автор надеется, что в этих областях заказчики могут заставить программировать автоматически, если они поймут, что только такие программы можно формально верифицировать

Спасибо за внимание!

Спасибо за внимание!