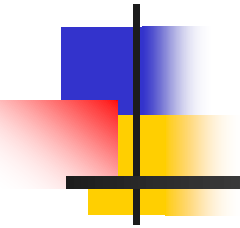
Лекция 9. Тестирование на основе формальных моделей. Пример технологии

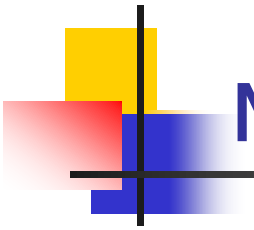


А.К.Петренко, А.В.Хорошилов,
Е.В.Корныхин
МГУ ВМК, ИСП РАН

http://sp.cmc.msu.ru/courses/fmsp

Осень, 2012

Тестирование API программного модуля/объекта



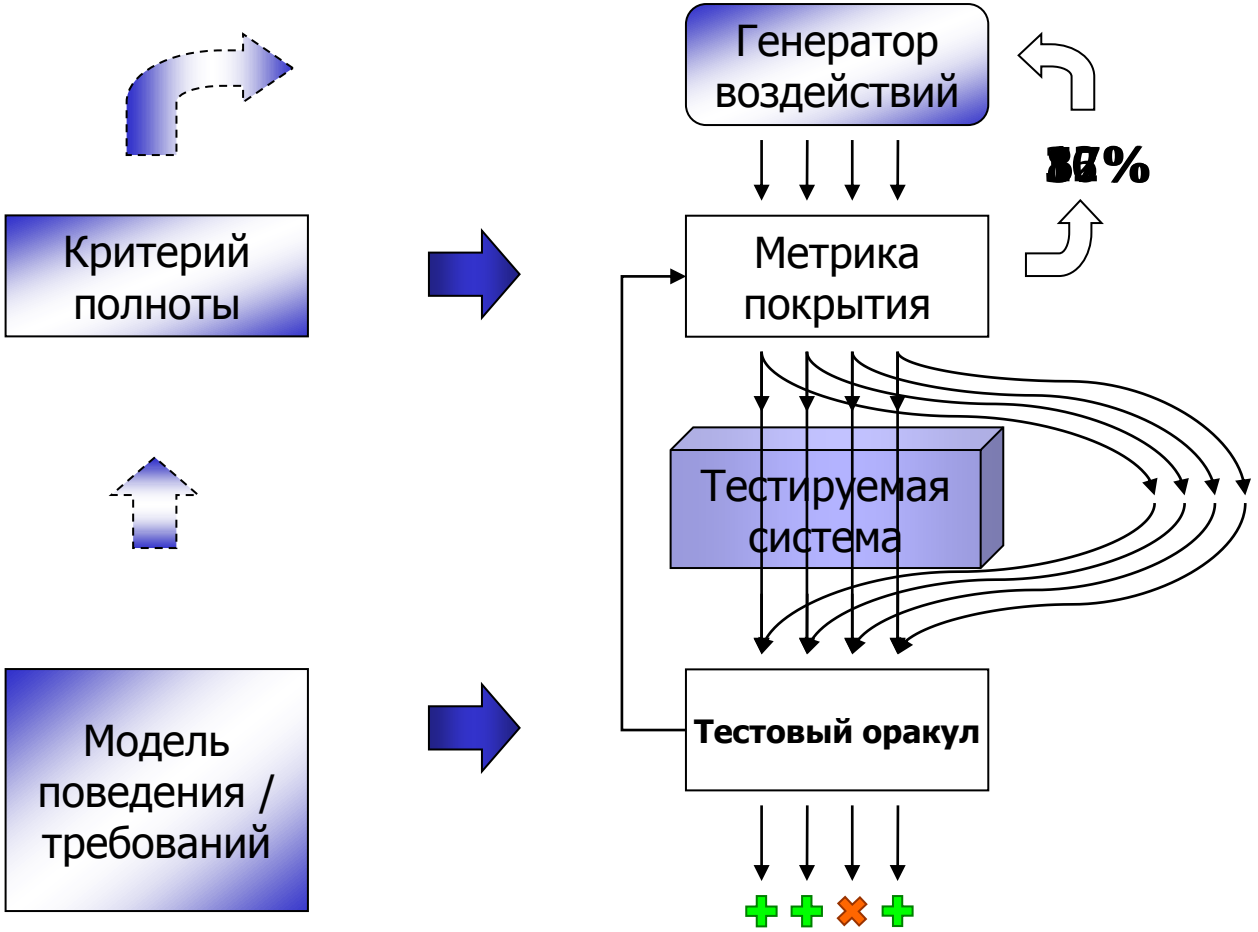
- Если за основу формальной модели брать FSM или LTS, их размер и сложность могут оказаться существенно больше, чем размер и сложность реализации модуля.
- Спецификация программного контракта, как правило, компактнее реализации
- Вопрос: как построить тест на основе программного контракта?

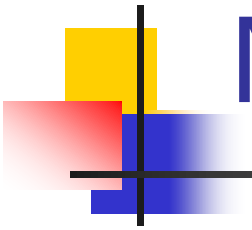
UniTESK. Пример Formal Testing технологии

План

- Архитектура тестовой системы для тестирования отдельных функций (методов)
- Описание программного контракта
- Сценарий тестирование подсистемы (объекта)
- Обобщенный сценарий тестирования
- Архитектура тестовой системы. Общий случай.
- Платформы.

Создание тестов и тестирование на основе программных контрактов





Модели в UniTESK

- Описание требований
  - Поведение – контрактные спецификации
  - Параллелизм – семантика чередования
- Критерии полноты
  -
- Построение тестов
  - Генераторы простых данных
  - Построение конечных автоматов, обеспечивающих полноту покрытия ситуаций, и их автоматический обход

Контрактные спецификации

```
specification void deposit(int sum)
{
        pre {return
                (0 < sum ) &&
                (balance <= Integer.MAX_VALUE - sum); }
        post {return
                balance == pre balance + sum;
        }
}
```

Контрактные спецификации и метрика покрытия

```
specification void deposit(int sum)
{
pre {return (0 < sum ) && (balance <= Integer.MAX_VALUE - sum);}
post
  {
      if (balance > 0)
            mark "Deposit on account with positive balance";
      else if (balance == 0)
            mark "Deposit on empty account";
      else
            mark "Deposit on account with negative balance";
      branch Single;
      return balance == pre balance + sum;
  }
}
```

Спецификация withdrow

```
specification int withdraw(int sum) {
pre { return sum > 0; }
post {
    if (balance > 0)
        mark "Withdrawal from account with positive balance";
    else if (balance == 0)
        mark "Withdrawal from empty account";
    else
        mark "Withdrawal from account with negative balance";
    if(balance < sum – maximumCredit) {
        branch TooLargeSum;
        return balance == pre balance && withdraw == 0 ;
    } else {
        branch Normal;
        return balance == pre balance – sum && withdraw == sum ;
    }
  }
}
```

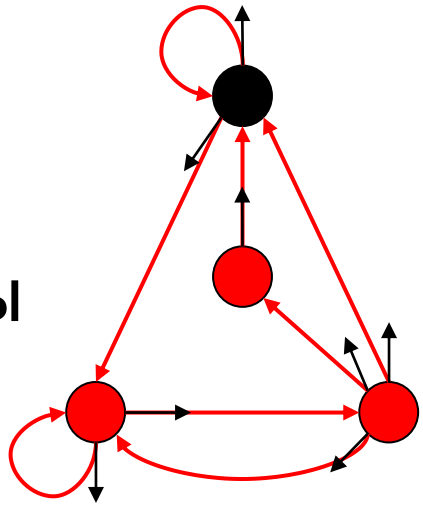
Простой тестовый сценарий

```
scenario deposit()
{
    if(objectUnderTest.balance < maxBalance)
    {
      objectUnderTest.deposit( 1 );
    }
    return true;
}
```

Задача автоматизированного сценария тестирования компонента с состоянием

State exploration:

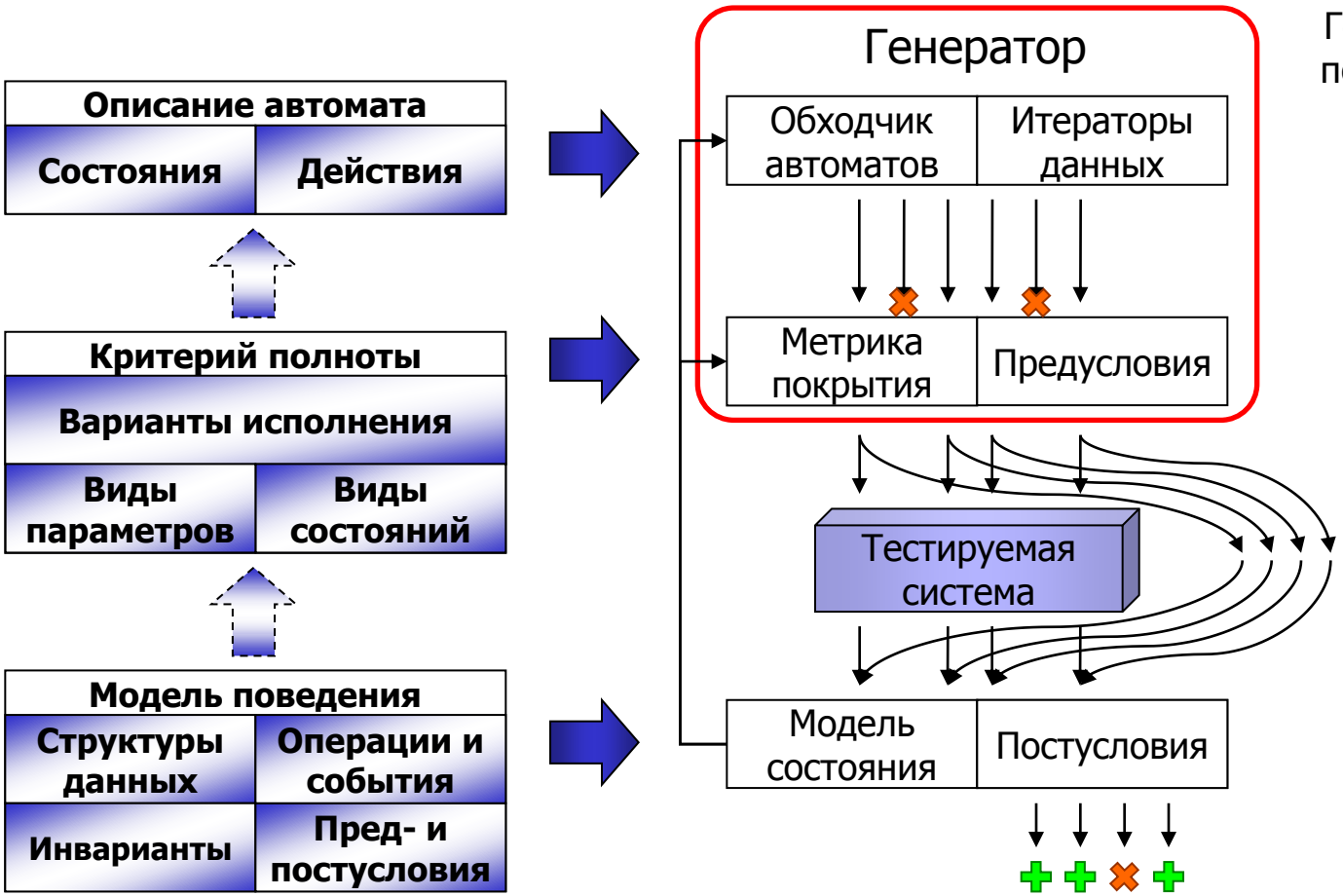
- для каждого метода
- в каждом состоянии
- перебрать все нужные параметры



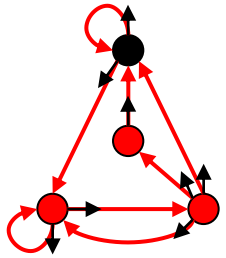
Структура переходов между состояниями –

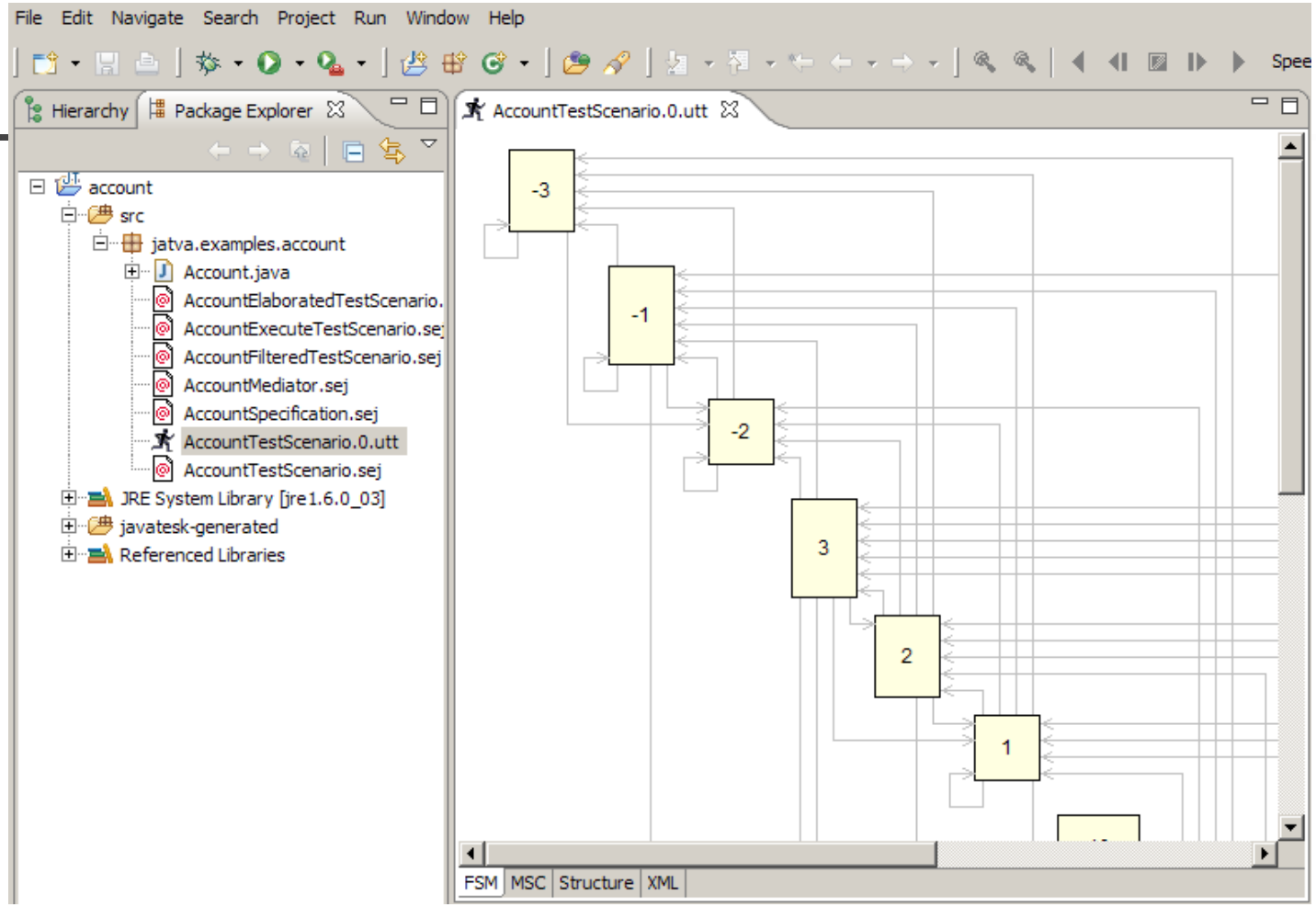
конечный автомат.

Тестирование компонентов (например, объектов)



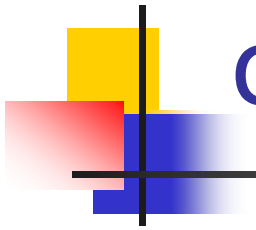
Генерация тестовой последовательности на лету





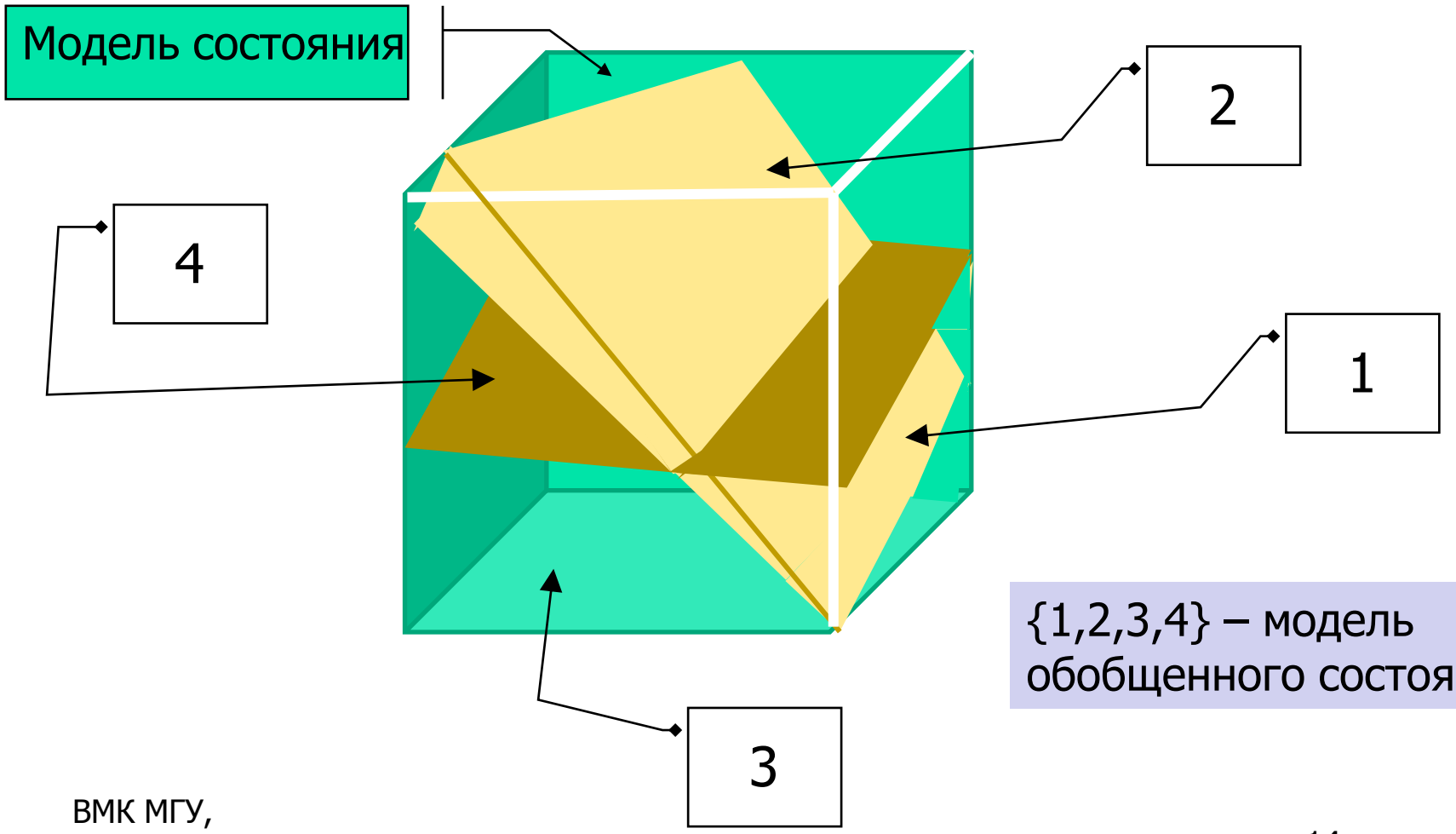
Что необходимо для того, чтобы построить такой автомат?

Состояния программы и состояния конечного автомата



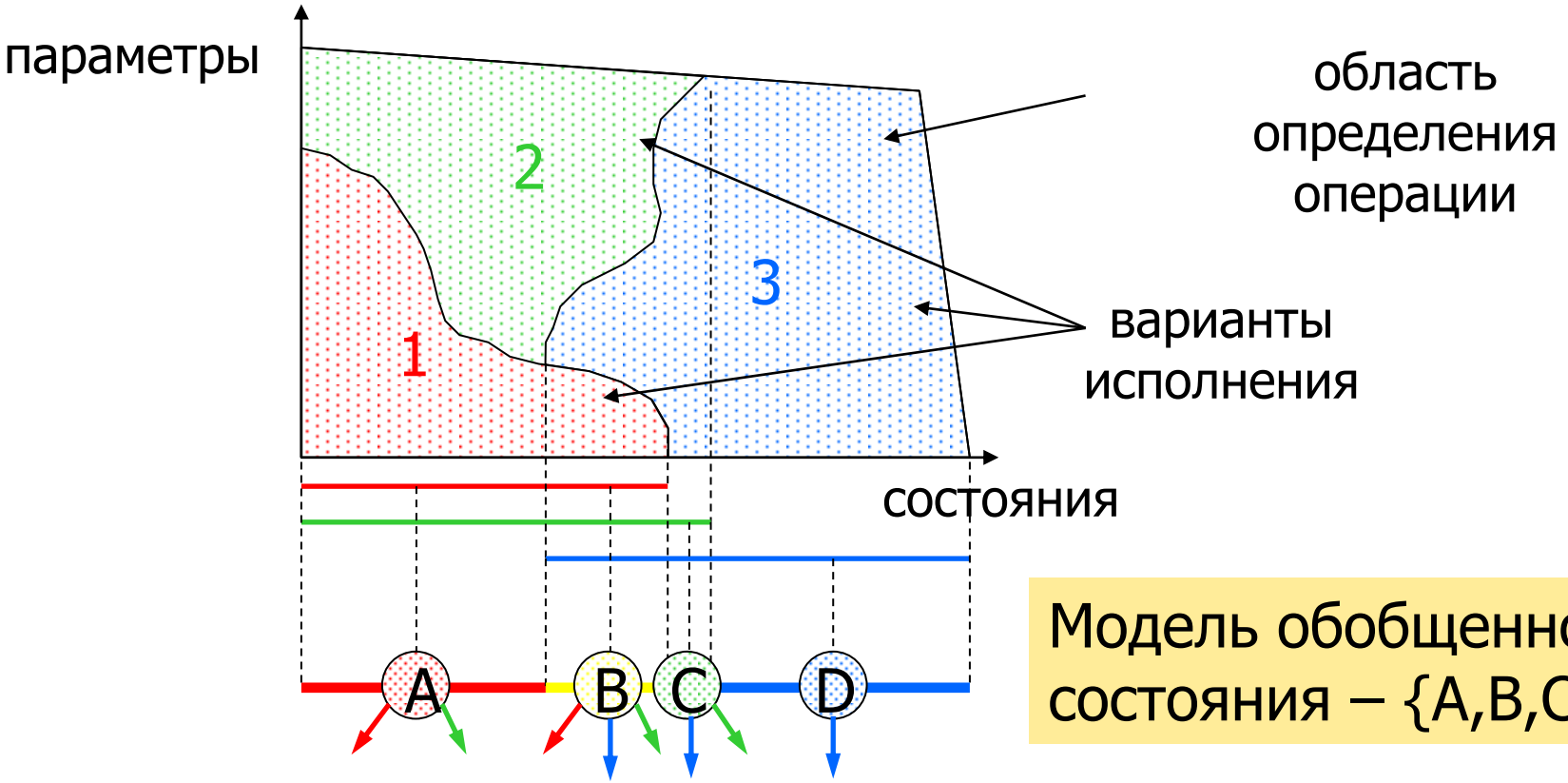
- Размер пространства значений данных, определяющих состояние программной системы (даже модели поведения) обычно крайне велико.
- Разумное ограничение на число состояний (и переходов между состояниями) конечного автомата при генерации тестов обычно до миллиона состояний.
- Как сократить число состояний? Как выбрать адекватное обобщение состояний модели?
- Что будет, если обобщение неадекватное?

Разбиение на классы эквивалентности. Выделение обобщенного автомата

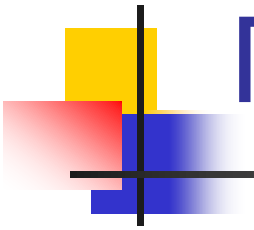


{1,2,3,4} – модель обобщенного состояния

Построение обобщенного автомата



Модель обобщенного состояния – {A,B,C,D}

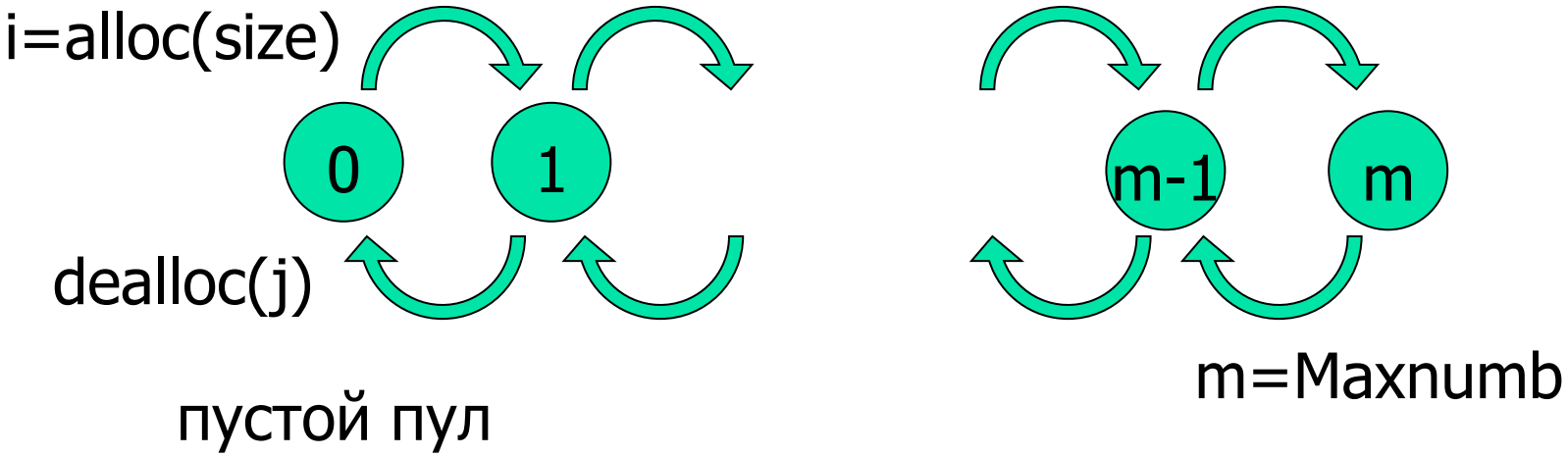


Пример задачи (1)

- Модель пула записей
- Операции:
  - int alloc(size), результат – номер записи в пуле, 0 означает исчерпание пула
  - bool dealloc(int itemnumb) – false означает отсутствие такой записи в пуле
  - int Maxnumber – максимальное число записей в пуле.
  - state - модель состояния (например, список items).

Пример задачи (2)

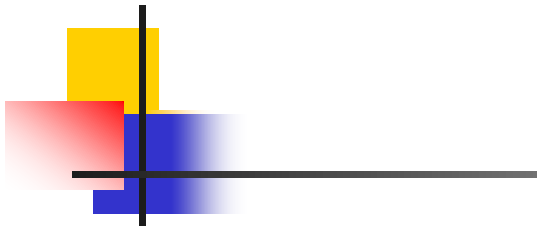
{0,1,…,m} – обобщенное состояние



- Вопросы:
  - данный автомат детерминирован?
  - а если Maxnumb в модели не известен?

Автоматизированный тестовый сценарий

```
package jatva.examples.account;
import jatva.engines.DFSMExplorer;
scenario class AccountTestScenario
{
public static AccountSpecification configureMediators()
{
AccountSpecification result = mediator AccountMediator( targetObject
= new Account() );
result.attachOracle();
return result;
}
int maxCredit = 3;
int maxBalance = 10;
public static void main( String[] args )
{
jatva.tracer.Tracer.getPrototype().setXmlFormat();
AccountTestScenario myScenario = new AccountTestScenario();
if(args.length > 0)
{
int n = Integer.parseInt(args[0]);
if(n < 1) n = 1;
myScenario.maxBalance = n;
if(args.length > 1)
{
n = Integer.parseInt(args[1]);
if(n < 0) n = 0;
myScenario.maxCredit = n;
Account.maximumCredit = n;
}
}
myScenario.run();
}
protected AccountSpecification objectUnderTest;
public AccountTestScenario()
{
objectUnderTest = configureMediators();
setTestEngine( new DFSMExplorer() );
}
state
{
return new Integer( objectUnderTest.balance );
}
scenario deposit()
{
if(objectUnderTest.balance < maxBalance)
{
objectUnderTest.deposit( 1 );
}
return true;
}
scenario withdraw()
{
iterate( int i = 1; i < maxCredit+3; i++; )
{
objectUnderTest.withdraw( i );
}
return true;
}
}
```

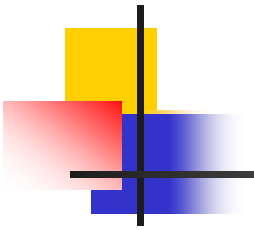


```
scenario class AccountTestScenario
{ public static AccountSpecification configureMediators()
{ AccountSpecification result = mediator AccountMediator( targetObject = new
Account() );
result.attachOracle();
return result;
}
int maxCredit = 3;
int maxBalance = 10;

...
public AccountTestScenario()
{ objectUnderTest = configureMediators();
setTestEngine( new DFSMExplorer() );
}
state
{  return new Integer( objectUnderTest.balance );
}
scenario deposit()
{ if(objectUnderTest.balance < maxBalance)
{ objectUnderTest.deposit( 1 );
}
return true;
}
scenario withdraw()
{ iterate( int i = 1; i < maxCredit+3; i++; )
{ objectUnderTest.withdraw( i );
}
return true;
} }
```

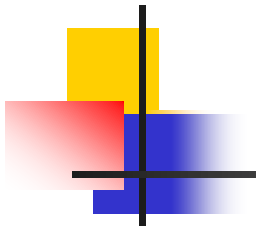
Платформы и области приложений UniTESK

- JavaTESK - open source
  - DOM API (www.unitesk.com)
- CTESK - open source
  - OS Linux (www.linuxtesting.org)
  - Тестирование микропроцессоров (http://hardware.ispras.ru/?q=CTESK)
- Summer (Java) - open source
  - Протоколы
- C++TESK (тестирование микропроцессоров) – free of charge http://hardware.ispras.ru/?q=CPPTESK



Аналоги UniTESK

- SpecExplorer (Microsoft Research) - free of charge
  - Microsoft Interoperability Initiative
- QTRONIC / Test Design (Conformiq) - commercial
  - Nokia projects, etc.
- NModel (Microsoft Research) - open source
  - many projects



Вопросы