В. В. Кулямин
Институт системного программирования РАН

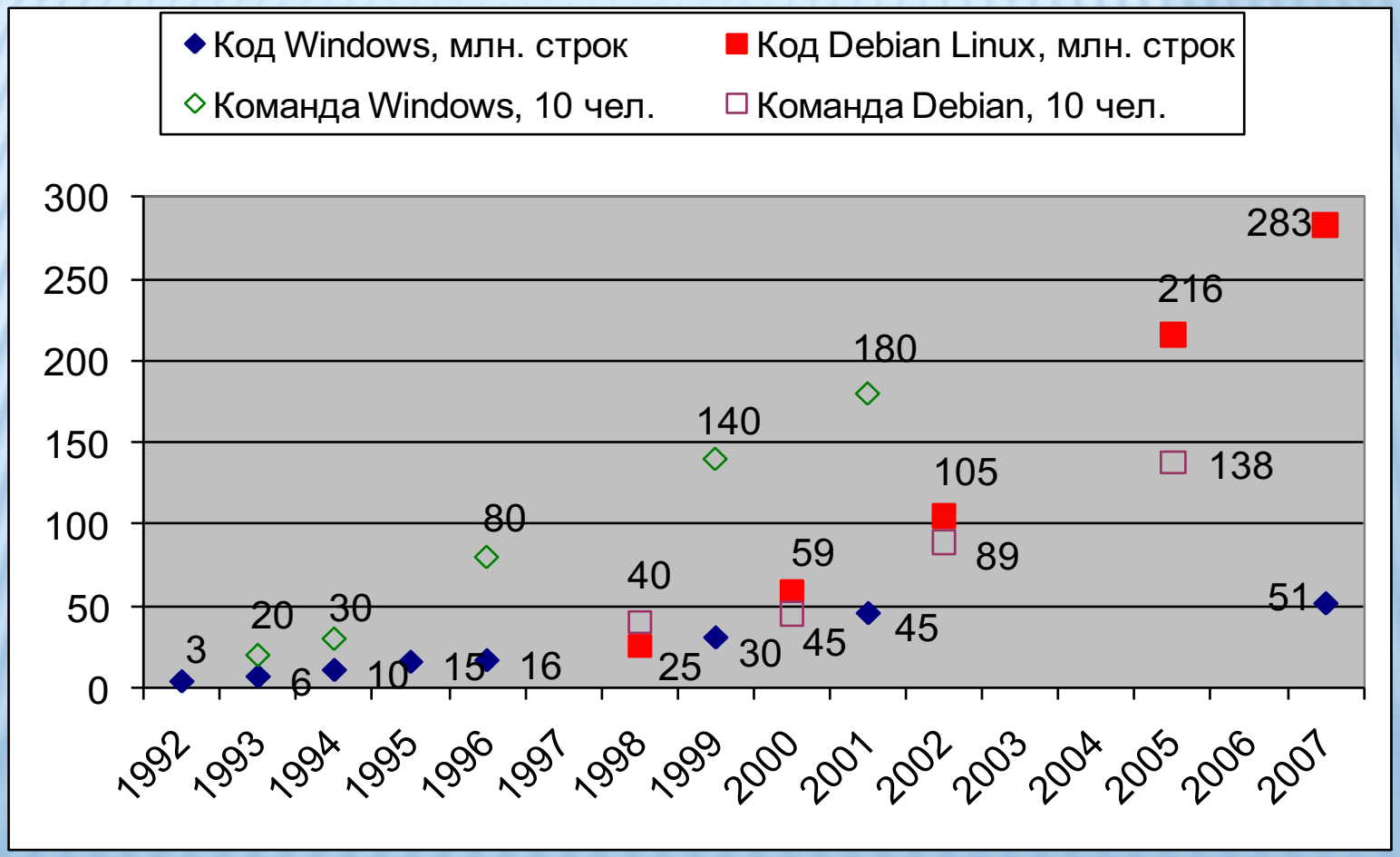
ТЕСТИРОВАНИЕ НА ОСНОВЕ МОДЕЛЕЙ

План

- Введение
  - Сложность современных систем
  - Тестирование и связанные понятия
- Тестирование на основе моделей
  - Модели поведения
  - Модели ситуаций
  - Методы построения тестов
- Работы ИСП РАН

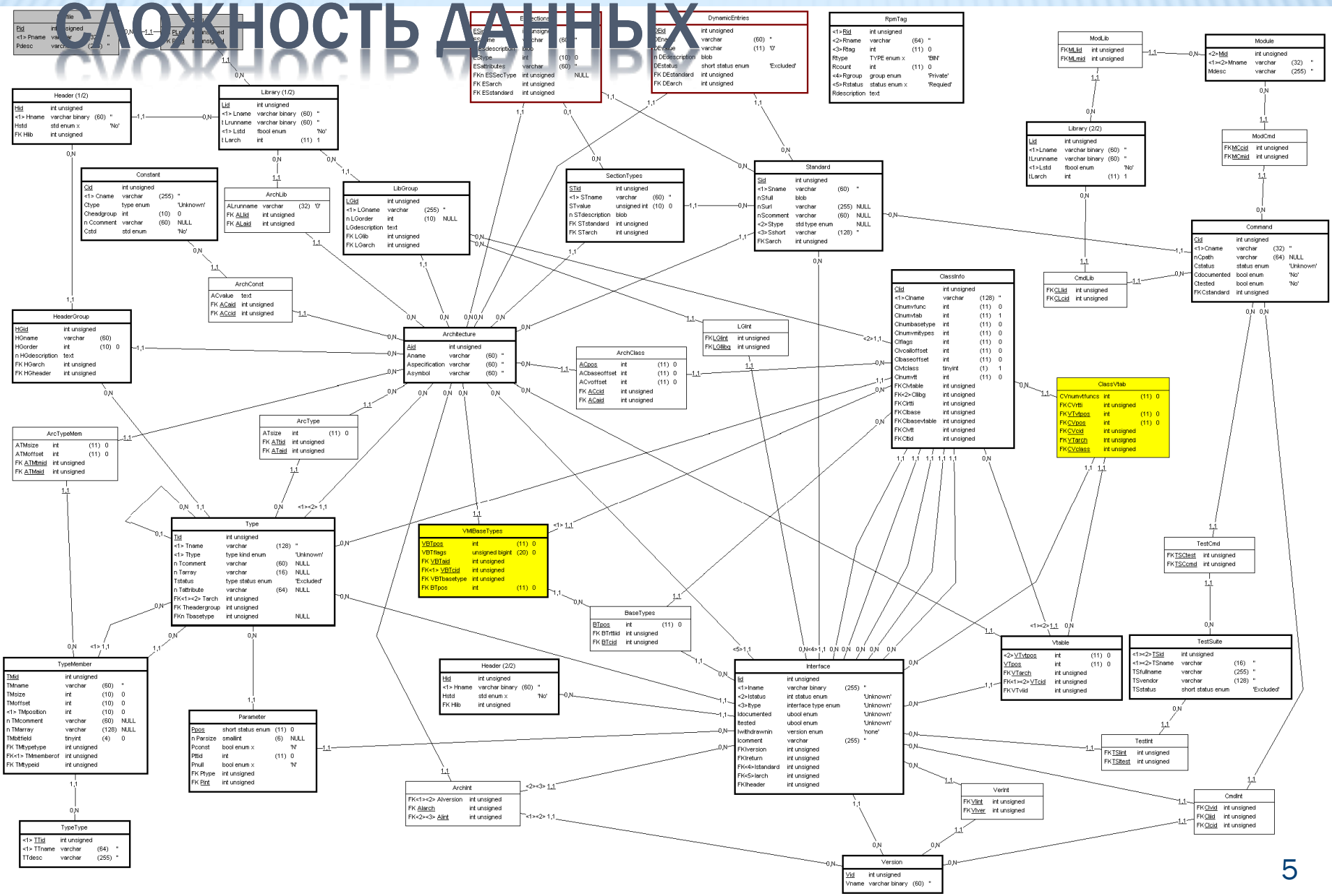
СЛОЖНОСТЬ СОВРЕМЕННЫХ СИСТЕМ

Software сложнее hardware (1968)



СЛОЖНОСТЬ ИНТЕРФЕЙСА

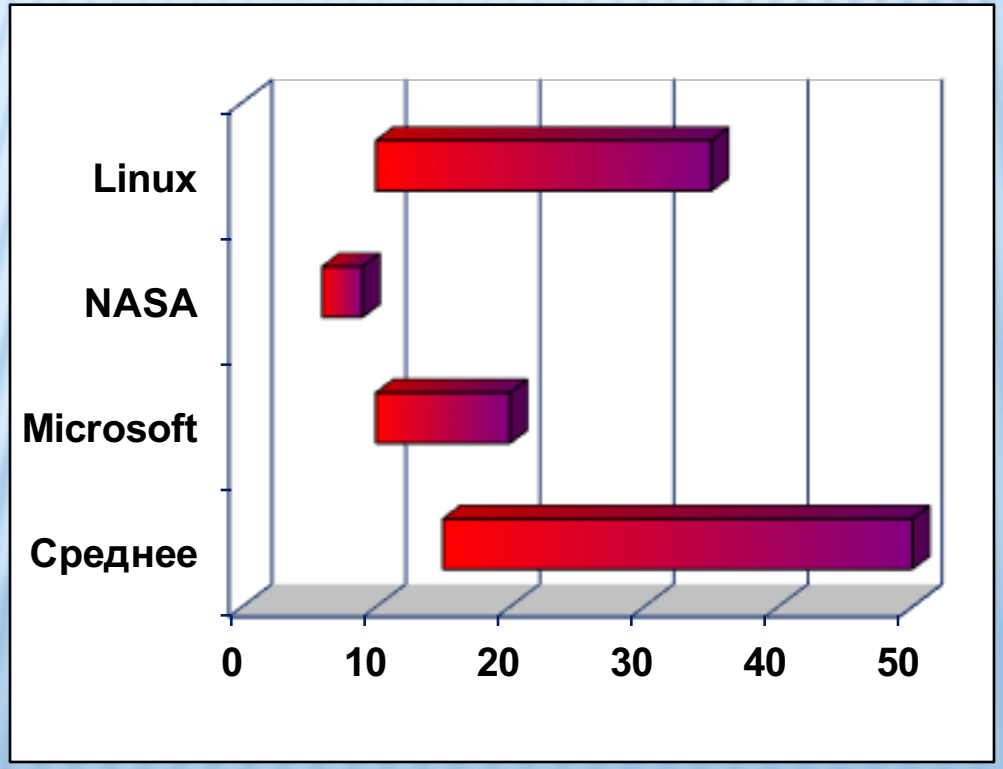
СЛОЖНОСТЬ ДАННЫХ



СЛОЖНОСТЬ ПОВЕДЕНИЯ

СТАТИСТИКА ОШИБОК

- Основная причина – сложность
- Среднее количество ошибок на 1000 строк кода – постоянно (???)



РИСКИ, СВЯЗАННЫЕ С ОШИБКАМИ

- Космические аппараты
  - Mariner I (1962)
  - Фобос-1 (1988)
  - Mars Climate Orbiter, Mars Polar Lander (1999)
- Инфраструктура
  - AT&T long distance network crash (1990)
  - Northeast Blackout (2003)
  - OpenSSL rnd in Debian (2006-8)
  - Heathrow Airport Terminal 5 baggage system (2008)
- Автомобили
  - Toyota Prius (2005, 2010)
- Медицинское оборудование
  - Therac-25 (1985-7)
- Авионика и военное оборудование
  - Lockheed F-117 (1982)
  - MIM-104 Patriot (1991)
  - Ariane 5 (1996)
  - USS Yorktown (1997)
  - F-22 Raptor (2007)

Потери индустрии США в 2001 году – 60 G$

Что делать?

- Не делать ошибок
  - Невозможно из-за сложности
- Предотвращение ошибок
  - Повышение уровня абстракции языков
  - Устранение error-prone конструкций
  - Стандартизация и документирование языков, интерфейсов и библиотек
- Выявление ошибок
  - Верификация и валидация

  Тестирование на основе моделей – частный случай
- Исправление ошибок

COMPUTER SCIENCE И SOFTWARE ENGINEERING

- Computer Science
  - Алгоритм
  - Одна четкая задача
    - ее м.б. невозможно решить
    - или можно решить идеально
  - Можно устранить ошибки
  - Основа – математика
- Software Engineering
  - Система
  - Много неясных задач
    - нужно решать неразрешимое
    - и нельзя решить окончательно
  - Ошибки есть всегда
  - Основа – mix (???)
    - Ad hoc решения
    - Микроэкономика
    - Групповая и когнитивная психология
    - Социология
- Фундаментальная дисциплина
  - Электродинамика
- Инженерная дисциплина (???)
  - Электротехника

ТЕСТИРОВАНИЕ

[SWEBOK 2004]

Проверка соответствия тестируемой системы требованиям

- Верификация
- Альтернативы – валидация
- Другие мнения (Cem Caner) – любой анализ

В ходе реальной работы этой системы

- Альтернативы

  Экспертиза, статический анализ, проверка моделей (model checking) дедуктивный анализ (theorem proving)

В конечном наборе специально созданных ситуаций

- Другие мнения (IEEE 610.12-1990)
- Альтернативы

  Мониторинг (runtime verification, passive testing)

ЦЕЛИ ТЕСТИРОВАНИЯ

- Поиск ошибок
  - Четкие результаты
  - Гарантировать отсутствие ошибок нельзя
  - Что делать, если ошибки не находятся?
  - Когда можно прекратить тестирование?
- Оценка качества тестируемой системы (SUT)
  - Объективные условия завершения
  - Критерии полноты тестирования – как выбирать тесты
- Контроль развития системы
  - Тесты – инструмент отслеживания изменений
  - Организация тестов – как облегчить модификации

ВИДЫ ТЕСТИРОВАНИЯ

- По проверяемым свойствам
  - ISO 9126
    - Функциональность
    - Надежность
    - Производительность
      - Нагрузочное
    - Переносимость
    - Удобство использования
    - Удобство сопровождения
  - Регрессионное
  - Аттестационное (соответствия)
- По исполнителю
  - При разработке
  - Альфа
  - Бета
  - Приемочное
- По уровню
  - Модульное
  - Компонентное
  - Интеграционное
  - Функций (features)
  - Системное
- По источнику тестов
  - Структурное
  - «Черного ящика»
  - «Серого ящика»
  - На отказ
    - «Дымовое»
    - Стрессовое
- По интерфейсу
  - UI
  - API
  - Сообщения

СТРУКТУРА ТЕСТА

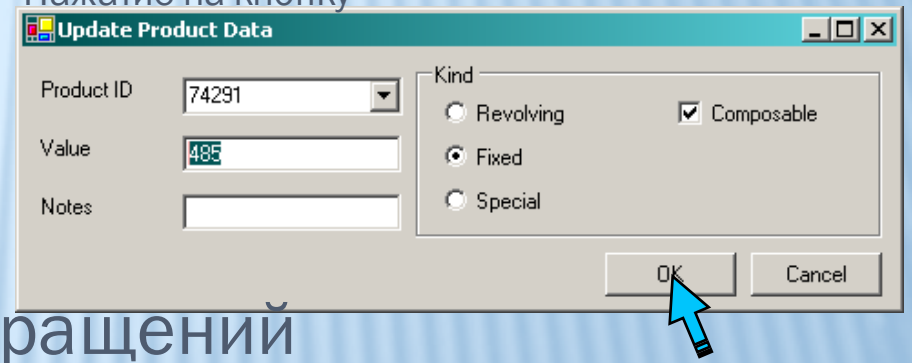
Тест – тестовая ситуация + проверка

- Тестовый вариант
  - Инициализация
  - Выполнение
  - Проверка
  - Финализация

```
public void testGetClients()
{
  Session s = SessionFactory.getSession();
  s.beginTransaction();
  List<Client> clients = s.createQuery("from Client").list();
  Client jones = new Client("Mike", "Jones");

  Assert.assertContains(clients, jones "Jones should be in clients");
  s.rollbackTransaction();
}
```

ТЕСТОВЫЕ СИТУАЦИИ

- Обращение к тестируемой системе
  - Воздействие
    - Вызов функции или метода
      Product.get(74291).setValue(485);
    - Отправка сообщения
      Send(new ProductValueChangeMsg(74291, 485));
    - Ввод команды
      produpd -t -id 74291 -val 485
    - Нажатие на кнопку
  - Данные обращения



- Набор параллельных обращений
- Состояние системы
- Внешние условия

МОДЕЛИ В ТЕСТИРОВАНИИ

- Требования
  - ***Модель поведения*** тестируемой системы (SUT)
  - Определяет корректное и некорректное поведение
- Структура тестовых ситуаций
  - ***Модель ситуаций***, возникающих при работе SUT
  - Определяет виды и элементы ситуаций, их важность и связанные риски
  - Критерий полноты тестирования, критерий покрытия, метрика покрытия

Тестирование на основе моделей

— Используемые модели заданы ***явно***

МОДЕЛИ ПОВЕДЕНИЯ

- Логико-алгебраические
  - Различные логики

    Первого порядка, теории типов, временные, модальные, …
  - Различные алгебраические структуры
- Исполнимые
  - Различные автоматы

    FSM, LTS, расширенные, взаимодействующие, иерархические, временные, сети Петри, Statecharts, …
- Гибридные
  - LTS ~ Алгебры процессов, модели стандартных термов
  - Abstract State Machines
  - Программные контракты с состоянием

МОДЕЛИ ПОВЕДЕНИЯ – АРИФМЕТИКА

x – 32-битное целое число

- abs(x) – (x ≥ 0) ? (x) : (-x)
- abs(x) ≥ 0
- abs(–2147483648) = ?2147483648

Машинная целочисленная арифметика

- $Z/2^{32}Z = \{[-2^{31}], [-2^{31}+1], \ldots, [-2], [-1], [0], [1], [2], \ldots, [2^{31}-1]\}$
- +, –, *
- (34)! = ?

МОДЕЛИ ПОВЕДЕНИЯ – SQRT

x – число с плавающей точкой (double)

sqrt(x) – квадратный корень

- pre : $(x \geq 0)$
- post: (sqrt*sqrt = x)

  $r = [sqrt(2)]_{double} = 6369051672525773/2^{52}$

  $r*r = 2.0000000000000002734323463...$

- post: $abs(sqrt*sqrt - x) < \varepsilon = 3*10^{-16}$

  $r = [sqrt(2^{105})]_{double} = 6369051672525773$

  $abs(r*r - 2^{105}) = 5545866846675497$

- post: $(x = 0) \Rightarrow (sqrt = 0)$

  $\& (x \neq 0) \Rightarrow abs((sqrt*sqrt - x)/x) < \varepsilon$

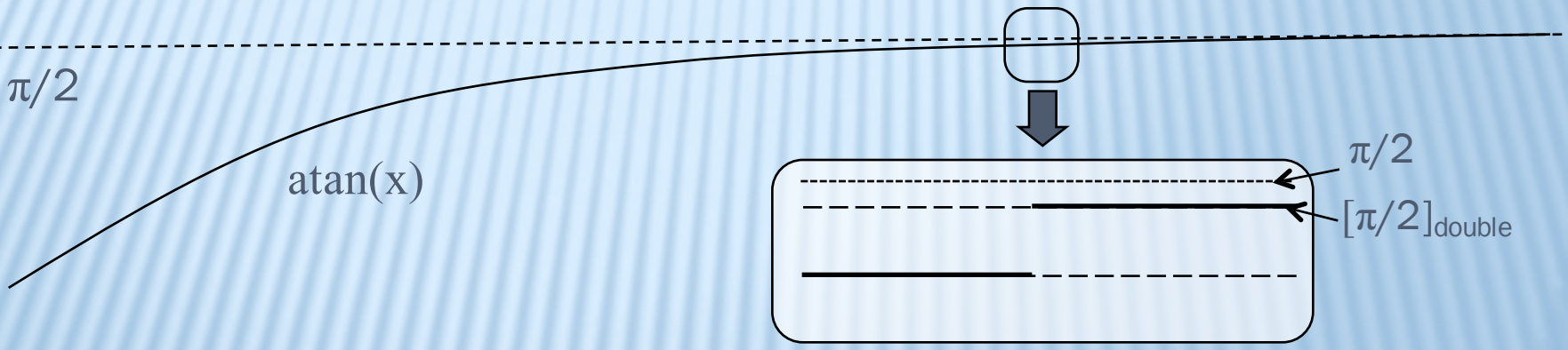
МОДЕЛИ ПОВЕДЕНИЯ – АРКТАНГЕНС

x – число с плавающей точкой (double)

atan(x) – арктангенс

- post: $(x = 0) \Rightarrow (atan = 0)$
  $\& (x \neq 0) \Rightarrow abs((tan(atan) - x)/x) < \varepsilon$

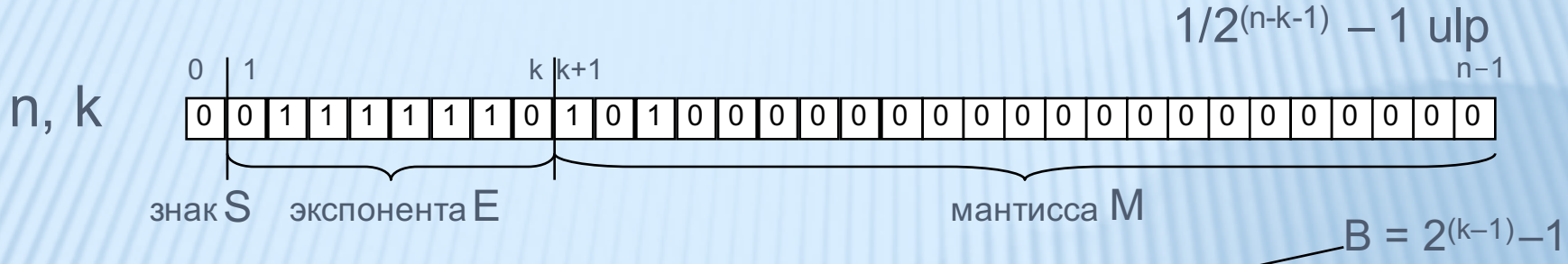
$(tan(atan(10^{50})) = 1.633123935319537... \cdot 10^{16}$



$[\pi/2]_{double} = 884279719003555/2^{49}$

$(\pi/2 - [\pi/2]_{double}) = 6.123233995736765... \cdot 10^{-17}$

ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ (IEEE 745)



- Нормализованные: $E > 0$ & $E < 2^k - 1$ $X = (-1)^S \cdot 2^{(E-B)} \cdot (1 + M/2^{(n-k-1)})$

$$2^{(-1)} \cdot 1.101_2 = 13/16 = 0{,}8125$$

- Денормализованные: $E = 0$ $X = (-1)^S \cdot 2^{(-B+1)} \cdot (M/2^{(n-k-1)})$

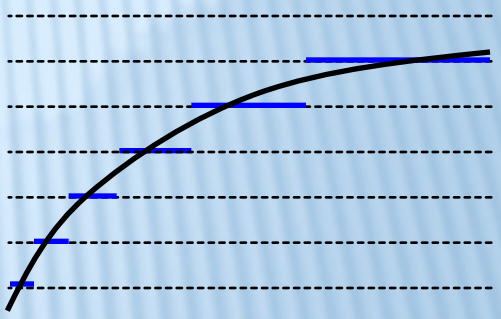
0, -0

- Exceptional : $E = 2^k - 1$
  - $M = 0$: $+\infty, -\infty$ $1/0 = +\infty$, $(2,1)/0 = -\infty$
  - $M \neq 0$: NaN $0/0 = \text{NaN}$

n = 32, k = 8 – float (single precision)
n = 64, k = 11 – double
n = 79, k = 15 – extended double
n = 128, k = 15 – quadruple

ТОЧНАЯ МОДЕЛЬ SQRT (IEEE 745)

- дает результат округления точного значения в соответствии с текущим режимом
  - к ближайшему
  - к +∞, к –∞, к 0



- для +∞ дает +∞ и выставляет флаг DIVISION-BY-ZERO
- для -0 дает -0 (???!!!)
- для x < 0 дает NaN и выставляет флаг INVALID
- в случае неточного результата выставляет флаг INEXACT
- нигде не выставляет флаги OVERFLOW, UNDERFLOW

МОДЕЛИ ПОВЕДЕНИЯ – СЕМАФОР

Расширенный конечный автомат

create() /sid; owner := null
waiting := new Queue
timer:= new TimerList

lock(tid)/ok_{tid}; owner:=tid
trylock(tid, wt)/ ok_{tid}; owner:=tid

[tid=owner]lock(tid)/$locked_{tid}$
[tid≠owner]lock(tid)/$fail_{tid}$
[tid=owner]trylock(tid, wt)/$locked_{tid}$
[tid≠owner]trylock(tid, wt)/timer[tid]:=wt,
waiting.push(tid)

destroy()

[tid=owner]unlock(tid)/ok_{tid}; owner:=null

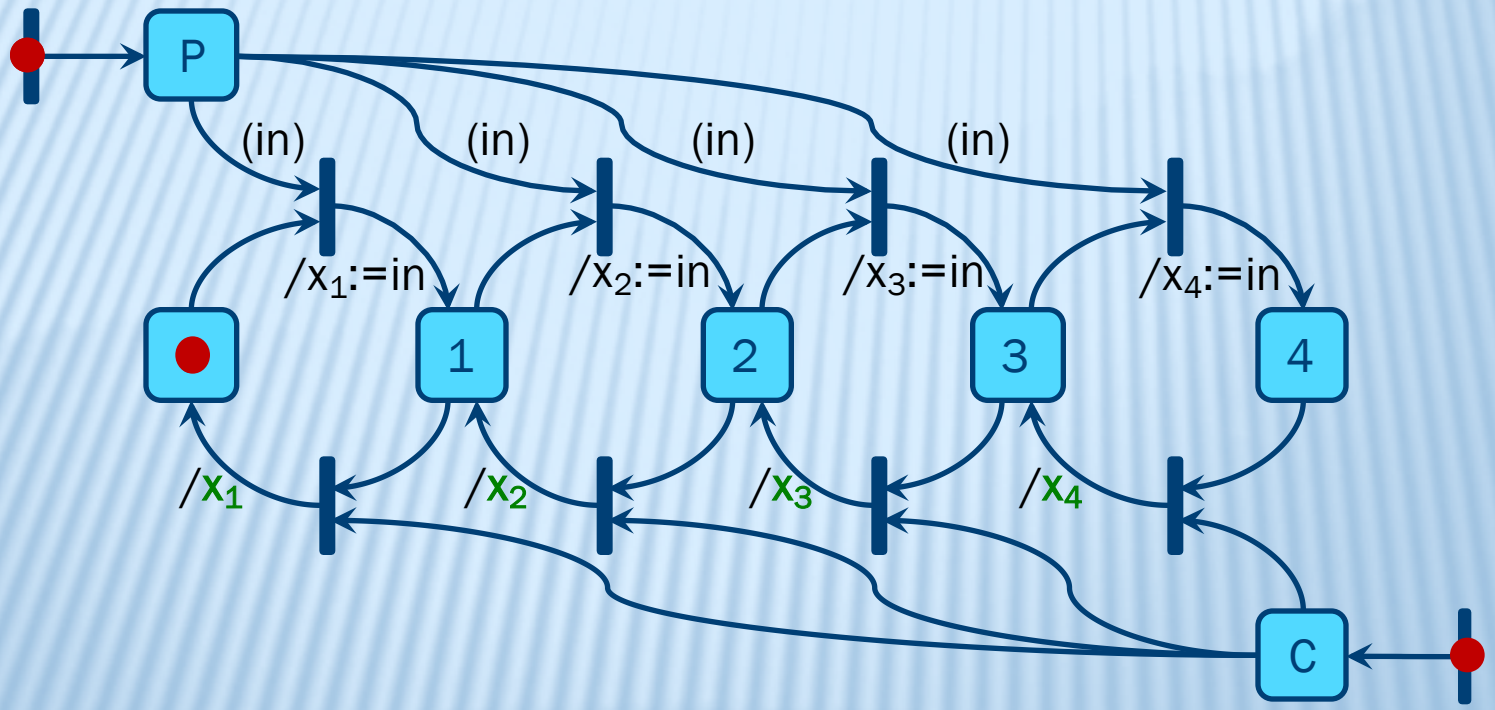
timeout[tid]/$fail_{tid}$; waiting.remove(tid)

[tid=owner & !waiting.empty()]unlock(tid)/ok;
owner:=waiting.pop(), timer[owner]:=0

[tid=owner & waiting.empty()]unlock(tid)/ok_{tid}; owner:=null

МОДЕЛИ ПОВЕДЕНИЯ – БУФЕР-СТЕК

Расширенная сеть Петри



МОДЕЛИ ПОВЕДЕНИЯ – СПИСОК

Абстрактный тип данных L – список объектов типа E

- Операции

  empty : L insert : L×N×E → L remove : L×N → L

  size : L → N get : L → E

- Аксиомы

	empty.size()	= 0
[0≤i≤X.size()]	X.insert(i, e).size()	= X.size() + 1
[0≤i<X.size()]	X.remove(i).size()	= X.size() – 1
[0≤i≤X.size()]	X.insert(i, e).get(i)	= e
[0<i≤X.size() & 0≤j<i]	X.insert(i, e).get(j)	= X.get(j)
[0≤i<X.size() & i<j≤X.size()]	X.insert(i, e).get(j)	= X.get(j-1)
[0≤i≤X.size()]	X.insert(i, e).remove(i)	≡ X
[0<i≤X.size() & 0≤j<i]	X.insert(i, e).remove(j)	≡ X.remove(j). insert(i-1, e)
[0≤i≤X.size() & i<j≤X.size()]	X.insert(i, e).remove(j)	≡ X.remove(j-1). insert(i, e)
[0≤i≤X.size() & 0≤j≤i]	X.insert(i, e_1).insert(j, e_2)	≡ X.insert(j, e_2).insert(i+1, e_1)
[0<i<X.size() & 0≤j<i]	X.remove(i).remove(j)	≡ X.remove(j). remove(i-1)

МОДЕЛИ СИТУАЦИЙ

- Составляющие
  - Элементы тестовых ситуаций
    - Действия и их наборы
    - Данные и состояния
  - Классификация ситуаций и их элементов, выделение однородных областей
  - Важность и риски
- Виды моделей ситуаций
  - **Доменные** (на основе входных и выходных данных)
  - **Структурные** (на основе затрагиваемых элементов SUT)
  - **Функциональные** (на основе элементов требований)
  - **Проблемные** (на основе возможных проблем и рисков, гипотез об ошибках)

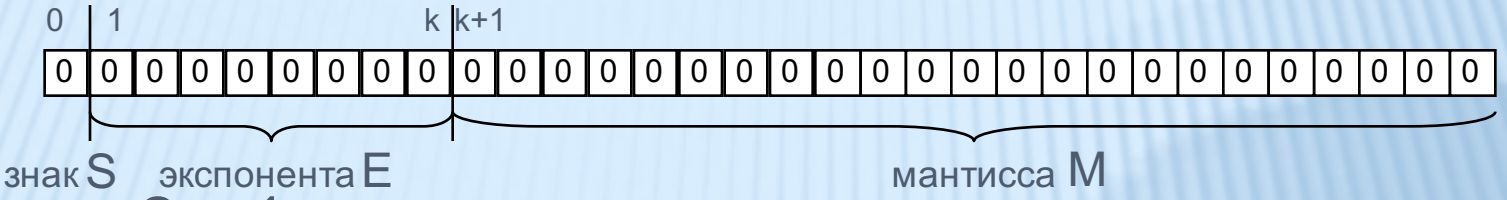
ДОМЕНЫ – КОМБИНАЦИИ ПОДТИПОВ

Вариант	Число A	Число B	A+B
1	0	0	0
2	0	+	+
3	0	MAX	MAX
4	0	–	–
5	0	MIN	MIN
6	+	0	+
7	+	+	+
8	+	+	OVR
9	+	MAX	OVR
10	+	–	+
11	+	–	0
12	+	–	–
13	+	MIN	–
14	–	0	–
15	–	+	+
16	–	+	0

Вариант	Чи
17	
18	
19	
20	
21	
22	
23	
24	
25	MAX MAX OVR
26	MAX MIN –
27	MIN 0 MIN
28	MIN + –
29	MIN - OVR
30	MIN MAX –
31	MIN MIN OVR

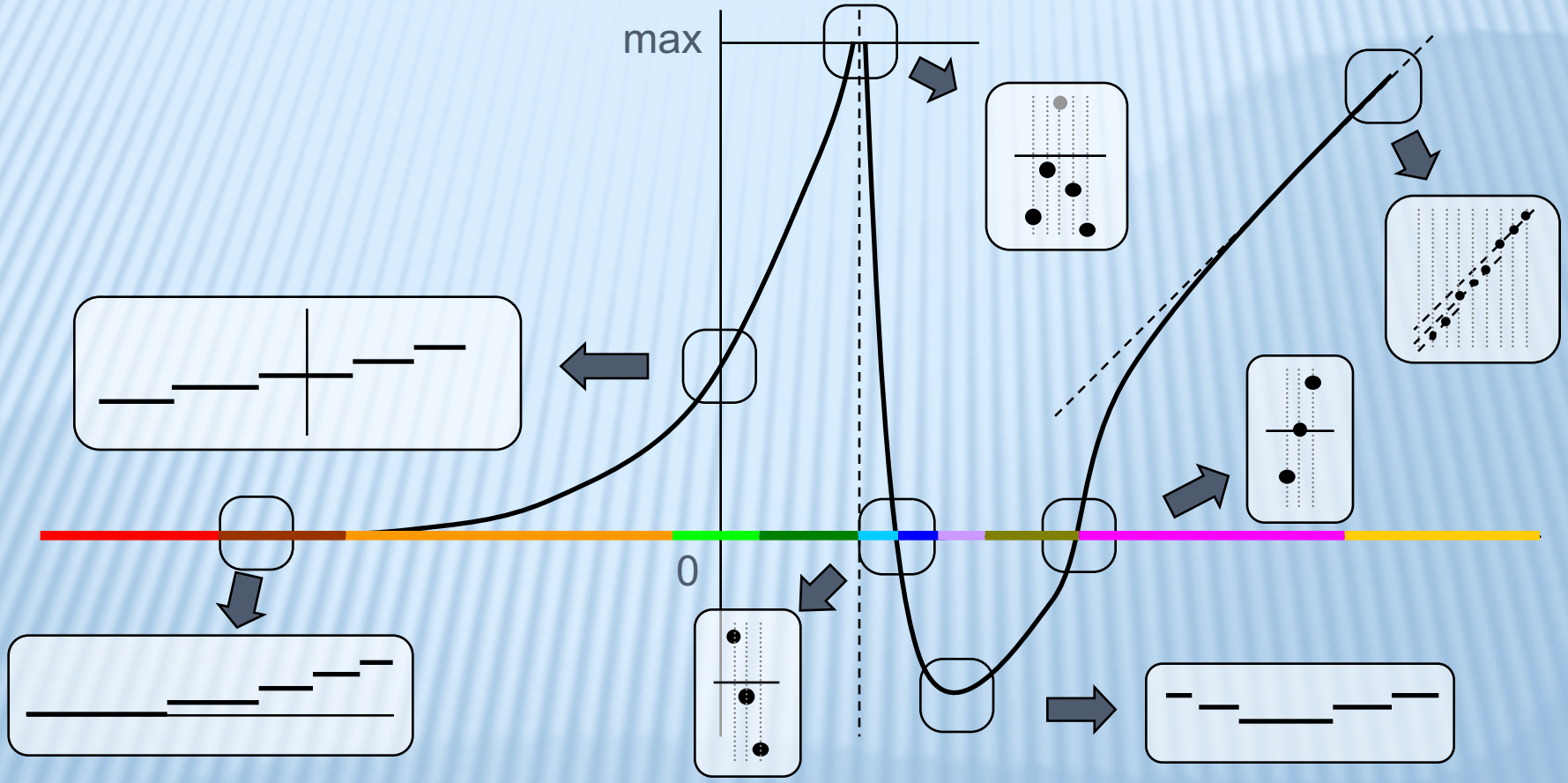
Вариант	Множество A	Множество B	A∩B
1	∅	∅	∅
2	∅	непусто	∅
3	непусто	∅	∅
4	непусто	непусто	∅
5	непусто	непусто	непусто = A = B
6	непусто	непусто	непусто = A ≠ B
7	непусто	непусто	непусто = B ≠ A
8	непусто	непусто	непусто ≠ A, ≠ B

ДОМЕНЫ – СТРУКТУРА ДАННЫХ



- S = 0 или S = 1
- Денормализованные числа E = 0
  - M = 0 +0, –0
  - M ≠ 0 000001, 000101, 010100, 100000, 111111
- Нормализованные числа $0 < E < 2^{k-1}$
  - E 000001, 000101, 011111, 100010, 111110
  - M 000000, 000001, 000110, 011001, 100110, 111111
- Исключительные числа $E = 2^{k-1}$
  - M = 0 $+\infty$, $-\infty$
  - M ≠ 0 NaN 000001, 000101, 010100, 101000, 111111

ДОМЕНЫ – ИНТЕРВАЛЫ ОДНОРОДНОСТИ

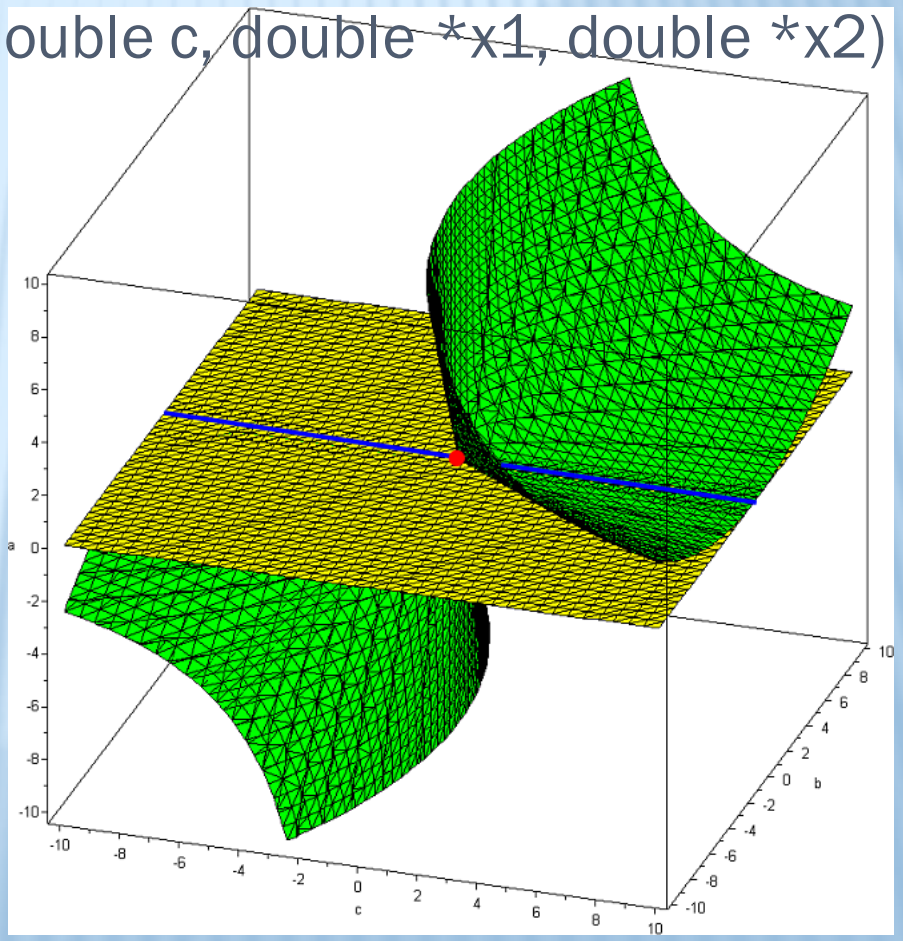


ДОМЕНЫ – ПОДОБЛАСТИ

$ax^2 + bx + c = 0$

int solve(double a, double, b, double c, double *x1, double *x2)

Вариант	Условие	Число решений	Результат
1	a = b = c = 0	∞	–1
2	a = b = 0, c > 0	0	0
3	a = b = 0, c < 0	0	0
4	a = 0, b > 0	1	1
5	a = 0, b < 0	1	1
6	a > 0, b^2-4ac < 0	0	0
7	a < 0, b^2-4ac < 0	0	0
8	a > 0, b^2-4ac = 0	1	1
9	a < 0, b^2-4ac = 0	1	1
10	a > 0, b^2-4ac > 0	2	2
11	a < 0, b^2-4ac > 0	2	2



Структура потока управления

- Компоненты, классы, методы
- Инструкции
- Ветвления
- Цепочки блоков
- Цепочки вызовов
- Элементарные условия и их комбинации

ПРИМЕРЫ КРИТЕРИЕВ ПОКРЫТИЯ

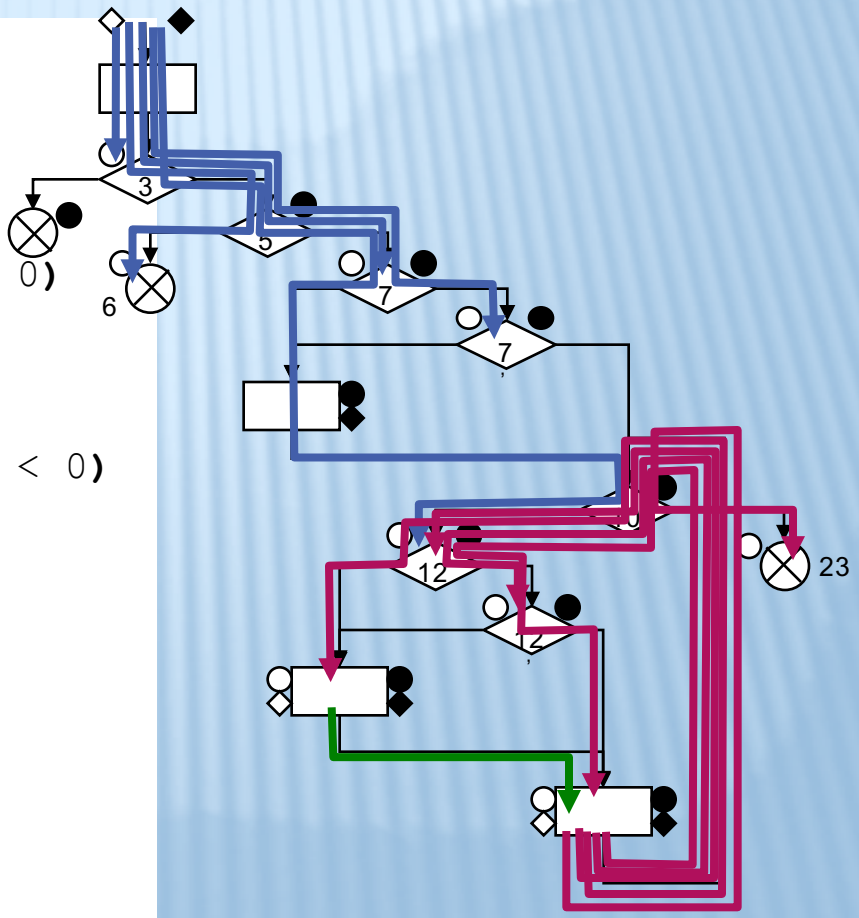
```
public Node ReplaceChild(Node nc, Node oc)
{
  if(nc == null) throw new NullI
  if(oc == null || !Children.Cor
  if( !(this is Document) && nc
    throw new DomException("New
  if(nc == this || Ancestors.Cor
  if(!(nc is ISPImplDocumentFrac
  if(this is Document && nc is I
    throw new DomException("The
  if(   this is Document && nc i
     && HasDifferentNoncommentCh
    throw new DomException("Docu
  if(   this is Document && !(nc
     && HasDifferentDocumentType
    throw new DomException("Non-
  if(this is Document && nc is I
    throw new DomException("The
  if(this.IsReadOnly) throw new
  if(nc.ParentNode != null && nc
  if(nc is DocumentFragment) {
    foreach(Node ni in nc.Childr
      if(!this.IsAllowedChild(ni
      if(this is Document && ni
        throw new DomException("
      if(this is Document && ni
        throw new DomException("
      if(this is Document && !(n
        throw new DomException("
    }
  }

  int i = this.Children.IndexOf
  if(!(nc is DocumentFragment) {
    if(nc.ParentNode != null) nc
    nc.parent = this;
    if(this.Children.Contains(nc
    this.Children.Insert(i, nc);
  }
  else if(nc is DocumentFragment
    foreach(Node ni in nc.Childr
    nc.Children.Clear();
  }
  if(nc != oc) { this.Children.Remove(oc); oc.parent = null; }
  return oc;
}
```

Условия				Ветви		Критерии покрытия						
a	b	c	d	a || !b && c	c && !a || b && d	DC	CC	D/CC	MDC	MC/DC	SMCC	MCC
0	0	0	0	0	0	X	X	X	X	X	X	X
0	0	0	1	0	0							X
0	0	1	0	1	1	X			X	X	X	X
0	0	1	1	1	1							X
0	1	0	0	0	0						X	X
0	1	0	1	0	1				X		X	X
0	1	1	0	0	1					X		X
0	1	1	1	0	1							X
1	0	0	0	1	0				X		X	X
1	0	0	1	1	0							X
1	0	1	0	1	0						X	X
1	0	1	1	1	0					X		X
1	1	0	0	1	0							X
1	1	0	1	1	1							X
1	1	1	0	1	0					X	X	X
1	1	1	1	1	1		X	X		X	X	X

СТРУКТУРА ПОТОКОВ ДАННЫХ

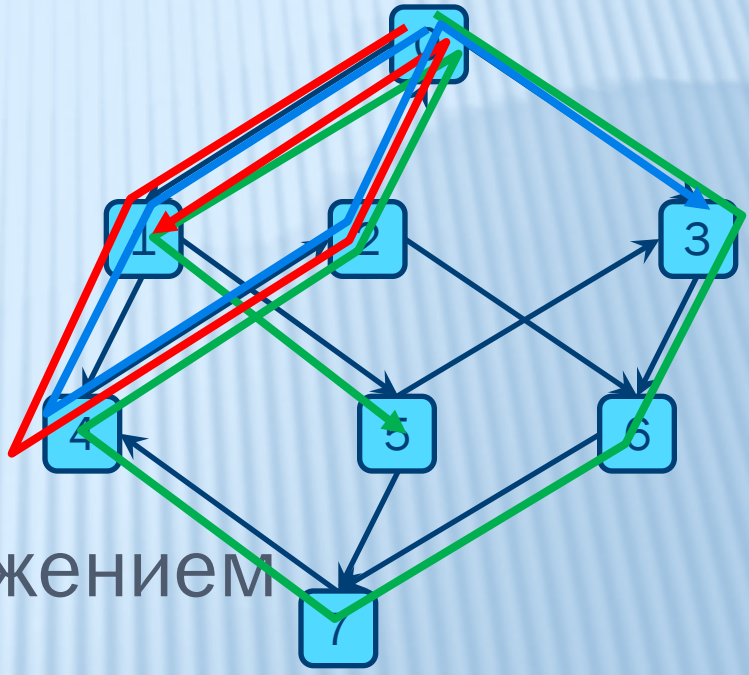
DU-пары и DU-пути

```
 1  int gcd(int a, int b)
 2  {
 3    if(a == 0)
 4      return b;
 5    if(b == 0)
 6      return a;
 7    if(a > 0 && b < 0 || a < 0 && b > 0)
 8      b = -b;
 9
10    while(b != 0)
11    {
12      if(b > a && a > 0 || b < a && a < 0)
13      {
14        a = b-a;
15        b = b-a;
16        a = a+b;
17      }
18
19      b = a-b;
20      a = a-b;
21    }
22
23    return a;
24  }
```



Структура автоматов

- Состояния
- Переходы
- Цепочки переходов
- Простые пути
- Простые циклы с продолжением



ПРОБЛЕМНАЯ МОДЕЛЬ – МУТАНТЫ

- Операторы мутации
  - if(A) B else C
    - ⇒ if(A) B
    - ⇒ if(A) C
    - ⇒ if(A) C else B
    - ⇒ B
    - ⇒ C
  - X := Y + Z
    - ⇒ X := Y
    - ⇒ X := Z
    - ⇒ X := Y – Z
    - ⇒ Y := Y + Z

- Программа P
  - ⇒ мутанты $\{P_1, P_2, ..., P_n\}$
  - устраняем эквивалентные
  - выполняем тесты
  - определяем «неубитых» мутантов

Методы построения тестов

- Вероятностные
- Комбинаторные
  - Автоматные
- Нацеленные

ВЕРОЯТНОСТНЫЕ МЕТОДЫ

Тестовые ситуации строятся на основе вероятностных распределений

- Модель ситуаций представляет собой распределения для различных элементов ситуаций
  - По частоте использования
  - По связанным рискам
  - По отлаженности

ДОСТОИНСТВА И НЕДОСТАТКИ

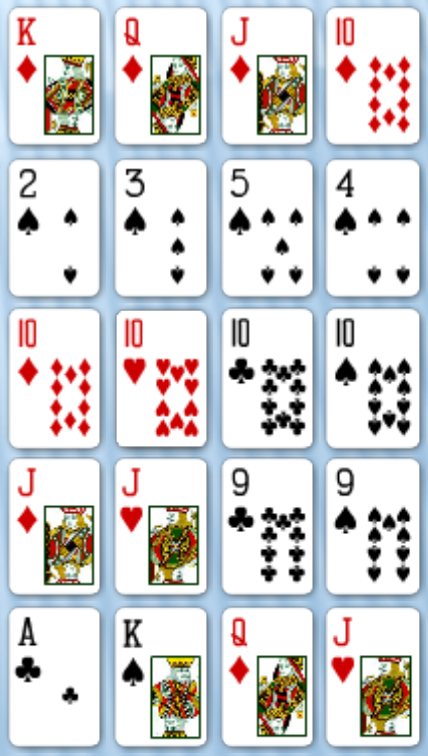
Вероятностное тестирование

- ✚ Позволяет получить много тестов с минимальными затратами
- ✚ Хорошо автоматизируется
- ✚ Хорошо находятся случайные ошибки (опечатки)

- − Полнота полученного набора тестов непредсказуема
- − Плохо находятся серьезные ошибки
- − Исходные распределения часто неизвестны

КОМБИНАТОРНЫЕ МЕТОДЫ

Тестовые воздействия разбиваются на некоторые элементы
Тестовые ситуации строятся как всевозможные комбинации этих элементов по определенным правилам



- Дерево классификации
- Тестирование на основе грамматик
- Покрывающие наборы
- Пути в автоматах

ДОСТОИНСТВА И НЕДОСТАТКИ

Комбинаторное тестирование

- ✚ Во многих случаях достаточно дешево
- ✚ Хорошо автоматизируется
- ✚ Более систематично, чем вероятностное

- − Некоторые виды ошибок находятся плохо
- − При учете многих факторов построение набора тестов требует гораздо больше ресурсов

ПРИМЕР: ПЕЧАТЬ WEB-СТРАНИЦЫ

Размер	Цвета	Формат	Принтер	Браузер	ОС
1 страница	Нет цветных рисунков	A4	HP	Internet Explorer	Windows XP
2 страницы	Есть цветные рисунки	A3	Epson	Mozilla Firefox	Windows Vista
7 страниц		A5	Canon	Opera	Linux Ubuntu
		B5	Xerox		Linux SUSE
		Letter			Linux RHEL

3·2·5·4·3·5 = 1800 вариантов

РЕШЕНИЕ

1	1 страница	Нет цветных рисунков	A4	HP	Internet Explorer	Linux RHEL
2	1 страница	Нет цветных рисунков	A4	HP	Opera	Windows XP
3	1 страница	Нет цветных рисунков	A3	Epson	Internet Explorer	Windows Vista
4	1 страница	Нет цветных рисунков	Letter	Canon	Internet Explorer	Linux RHEL
5	1 страница	Есть цветные рисунки	A5	Epson	Opera	Linux Ubuntu
6	1 страница	Есть цветные рисунки	B5	HP	Mozilla Firefox	Windows Vista
7	1 страница	Есть цветные рисунки	Letter	Xerox	Opera	Linux SUSE
8	2 страницы	Нет цветных рисунков	A3	Xerox	Mozilla Firefox	Linux RHEL
9	2 страницы	Нет цветных рисунков	B5	Canon	Opera	Linux SUSE
10	2 страницы	Нет цветных рисунков	Letter	HP	Mozilla Firefox	Linux Ubuntu
11	2 страницы	Есть цветные рисунки	A4	Xerox	Opera	Linux Ubuntu
12	2 страницы	Есть цветные рисунки	A3	HP	Opera	Linux SUSE
13	2 страницы	Есть цветные рисунки	A5	Xerox	Opera	Windows Vista
14	2 страницы	Есть цветные рисунки	Letter	Epson	Internet Explorer	Windows XP
15	7 страниц	Нет цветных рисунков	A4	Canon	Internet Explorer	Windows Vista
16	7 страниц	Нет цветных рисунков	A3	Xerox	Opera	Windows XP
17	7 страниц	Нет цветных рисунков	A5	HP	Internet Explorer	Linux SUSE
18	7 страниц	Нет цветных рисунков	A5	Canon	Mozilla Firefox	Windows XP
19	7 страниц	Нет цветных рисунков	B5	HP	Internet Explorer	Linux RHEL
20	7 страниц	Нет цветных рисунков	B5	Xerox	Internet Explorer	Linux Ubuntu
21	7 страниц	Есть цветные рисунки	A4	Epson	Mozilla Firefox	Linux SUSE
22	7 страниц	Есть цветные рисунки	A3	Canon	Opera	Linux Ubuntu
23	7 страниц	Есть цветные рисунки	A5	Epson	Opera	Linux RHEL
24	7 страниц	Есть цветные рисунки	B5	Epson	Opera	Windows XP
25	7 страниц	Есть цветные рисунки	Letter	Epson	Internet Explorer	Windows Vista

25 вариантов
Покрытие пар – **покрывающий набор силы 2**

Можно построить комбинации всех троек – **покрывающий набор силы 3** – 100 вариантов

ПОКРЫВАЮЩИЕ НАБОРЫ

- Покрывающий набор силы t для k параметров, имеющих значения $n_1, \ldots, n_k$
  – матрица $\{C_{ij}\}$ $i \in [1..N]$, $j \in [1..k]$, $C_{ij} \in [1..n_j]$
  $\forall j1, \ldots, jt$ $jl \in [1..k]$ $\forall v_{j1}, \ldots, v_{jt}$ $v_{jl} \in [1..n_{jl}]$
  $\exists i \in [1..N]$ $\forall l \in [1..t]$ $C_{ijl} = v_{jl}$
  множество таких наборов – $CA(t; n_1, \ldots, n_k)$
- Однородные наборы
  – $n_1 = \ldots = n_k = n$, $CA(t; k, n)$
- Минимальное число строк в однородном наборе
  – $\sim t n^t \log^{1+\varepsilon}(k)$
- Построение минимальных однородных наборов из $CA(t; k, 2)$ или $CA(2; k, n > 2)$ – NP-полная задача

БИНАРНЫЕ ПОКРЫВАЮЩИЕ НАБОРЫ

Алгоритм построения CA(2; k, 2)

- Выбрать N : $k \leq C_{N-1}^{\lceil N/2 \rceil}$
- Можно построить набор из N строк

k	N
2-3	4
4	5
5-10	6
11-15	7
16-35	8
36-56	9
57-126	10
127-210	11

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
3	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1
4	1	0	1	1	1	0	1	1	1	0	0	0	1	1	1
5	1	1	0	1	1	1	0	1	1	0	1	1	0	0	1
6	1	1	1	0	1	1	1	0	1	1	0	1	0	1	0
7	1	1	1	1	0	1	1	1	0	1	1	0	1	0	0

ГЕОМЕТРИЧЕСКИЕ КОНСТРУКЦИИ

Алгоритм построения CA(2; p^k+1, p^k)

- $n=p^k$, p – простое
- Первый столбец (∞)– блоки по n одинаковых значений i
- Второй столбец (0) – все n значений j в каждом блоке
- В столбце с номером $m \in 1..(n-1)$ (i*m+j) в поле GF(p^k)

	∞	0	1	2	3	4
1	0	0	0	0	0	0
2	0	1	1	1	1	1
3	0	2	2	2	2	2
4	0	3	3	3	3	3
5	0	4	4	4	4	4
6	1	0	1	2	3	4
7	1	1	2	3	4	0
8	1	2	3	4	0	1
9	1	3	4	0	1	2
10	1	4	0	1	2	3
11	2	0	2	4	1	3
12	2	1	3	0	2	4
13	2	2	4	1	3	0
14	2	3	0	2	4	1
15	2	4	1	3	0	2
16	3	0	3	1	4	2
17	3	1	4	2	0	3
18	3	2	0	3	1	4
19	3	3	1	4	2	0
20	3	4	2	0	3	1
21	4	0	4	3	2	1
22	4	1	0	4	3	2
23	4	2	1	0	4	3
24	4	3	2	1	0	4
25	4	4	3	2	1	0

РЕКУРСИВНЫЕ КОНСТРУКЦИИ

Алгоритм построения CA(2; mn+1, n=p^k) из CA(2; m, n)

- Исходный набор $\{A_{ij}\}$ i∈[1..N], j∈[1..
- Вспомогательная матрица $\{B_{ij}\}$ j∈[1..(n+1)]

		n				n				...	
N	0	A_{11}	A_{11}	...	A_{11}	A_{12}	A_{12}	...	A_{12}	...	A_{1m}
	0	A_{21}	A_{21}	...	A_{21}	A_{22}	A_{22}	...	A_{22}	...	A_{2m}
	...	...	...	...	...	...	...	...	...	...	...
	0	A_{N1}	A_{N1}	...	A_{N1}	A_{N2}	A_{N2}	...	A_{N2}	...	A_{Nm}
z	B_{11}	B_{12}	...	B_{12}	B_{13}	...	B_{13}	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...
	B_{z1}	B_{z2}	...	B_{z2}	B_{z3}	...	B_{z3}	...	...	...	...
		m			m			...	...	...	...

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	1	1	1	1	1	1
3	0	0	0	0	2	2	2	2	2	2	2	2	2	2
4	0	1	1	1	0	0	0	1	1	1	2	2	2	2
5	0	1	1	1	1	1	1	2	2	2	0	0	0	0
6	0	1	1	1	2	2	2	0	0	0	1	1	1	1
7	0	2	2	2	0	0	0	2	2	2	1	1	1	1
8	0	2	2	2	1	1	1	0	0	0	2	2	2	2
9	0	2	2	2	2	2	2	1	1	1	0	0	0	0
10	1	0	0	0	0	1	1	1	1	2	2	2	2	
11	1	1	1	1	1	2	2	2	2	0	0	0	0	
12	1	2	2	2	2	0	0	0	0	1	1	1	1	
13	2	0	0	0	0	2	2	2	2	1	1	1	1	
14	2	1	1	1	1	0	0	0	0	2	2	2	2	
15	2	2	2	2	2	1	1	1	1	0	0	0	0	

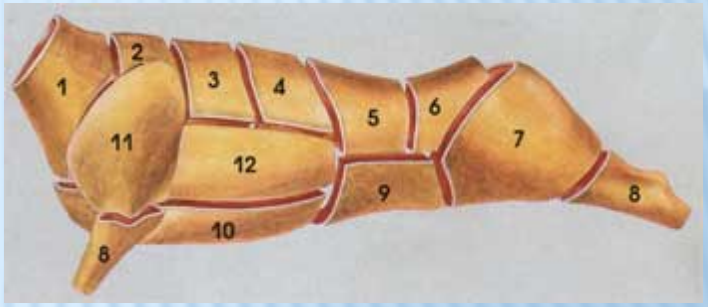
ОБЩИЕ ПОКРЫВАЮЩИЕ НАБОРЫ

- Неоднородные
- Переменной силы
- С ограничениями на сочетания значений

Переборно-оптимизационные алгоритмы

НАЦЕЛЕННЫЕ МЕТОДЫ

Тестовые ситуации, определяемые критерием полноты, строятся прямо



Источники выделяемых ситуаций

- Разбиения на классы эквивалентности
- Ограничения на элементы ситуаций
- Граничные случаи
- Риски
- Функции системы
- Длинные сценарии использования

ДОСТОИНСТВА И НЕДОСТАТКИ

Нацеленное тестирование

- ✚ Позволяет находить практически любые виды ошибок
- ✚ Позволяет концентрировать усилия на различных целях

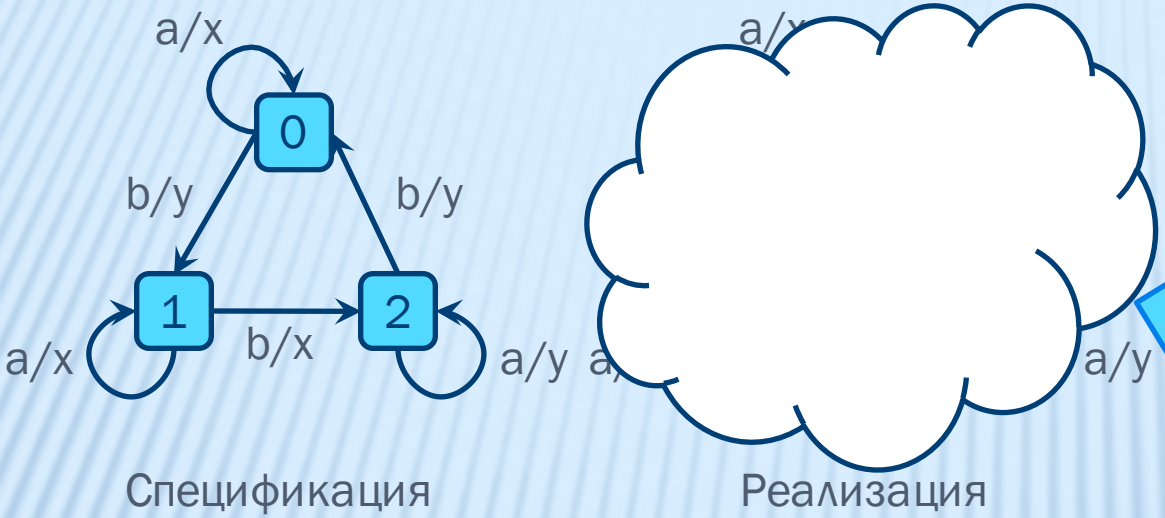
- ━ Плохо автоматизируется
- ━ Требует трудоемкого построения разбиений
- ━ Требует анализа осуществимости вариантов и подбора данных

Автоматизация создания тестов

Разрешение ограничений

- Перебор данных с проверкой попадания в нужную ситуацию
- Использование решателей (SMT-solvers)
- Оптимизационные алгоритмы

АВТОМАТНЫЕ МЕТОДЫ



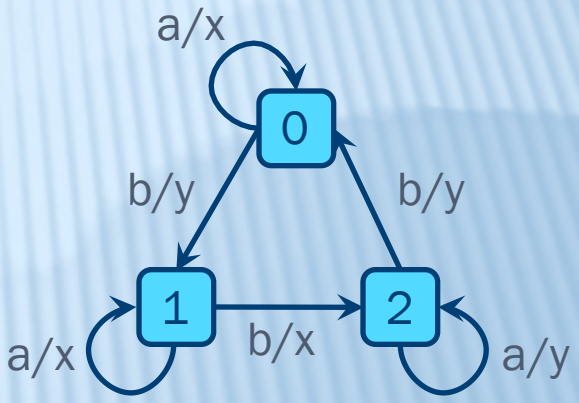
Всегда ли реализация ведет себя как спецификация?

ОГРАНИЧЕНИЯ

- Требования к спецификации
  - Детерминизм
  - Полная определенность
  - Минимальность
  - Сильная связность или наличие reset (R)
- Гипотезы о реализации
  - Реализация – конечный детерминированный автомат с теми же стимулами I и реакциями O
  - Полная определенность
  - Сильная связность или наличие reset (R)
  - В начале находимся в начальном состоянии
  - Число состояний ограничено

W-МЕТОД И D-МЕТОД

- Покрывающее множество C
  - Пример: C = {ε, b, bb}
- Различающая последовательность d
  - Есть не всегда, м.б. экспоненциальной длины
  - Пример: d = ab (0→xy, 1→xx, 2→yy)
- D-метод (не всегда применим, экспоненциален)
  - $\{R\}CI^{N-n+1}\{d\}$
  - Пример: RaabRbabRbaabRbbabRbbaabRbbbab
- Характеристическое множество W
  - Есть всегда, строится за $O(pn^2)$
  - Пример: W = {a, b} или {ab}
- W-метод (применим всегда, сложность $O(pn^3)$)
  - $\{R\}CI^{N-n+1}W$
  - Пример: RaaRabRbaRbbRbaaRbabRbbaRbbbRbbaaRbbabRbbbaRbbbb



ОБЩИЕ АВТОМАТНЫЕ МЕТОДЫ

- Недетерминизм
- Неполная определенность
- Различные тестовые возможности

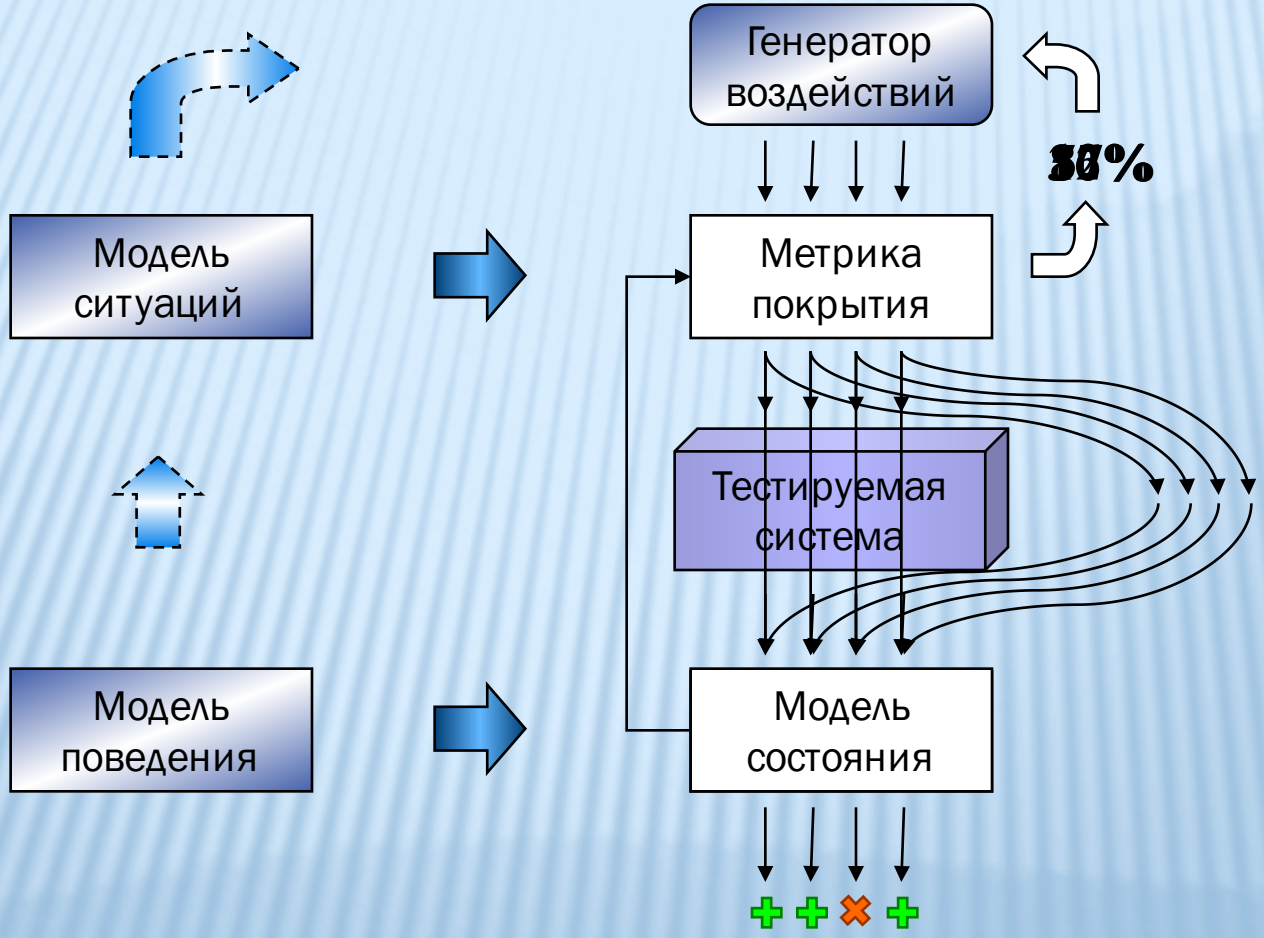
- Модели – Labeled Transition Systems (LTS)
- Сложные отношения согласованности

Теория разработана плохо
– нет финитных моделей ситуаций

РАБОТЫ ИСП РАН

- **Разработка тестов и тестирование**
  - Операционные системы реального времени
  - Базовые библиотеки Linux (Linux Standard Base)
  - Протоколы IPv6, Mobile IPv6, IPsec
  - Отдельные модули компиляторов Intel
  - Микропроцессоры архитектуры MIPS
- **Создание технологий и инструментов**
  - Тестирование на основе моделей (UniTESK)
  - Проверка соответствия стандарту LSB

ТЕХНОЛОГИЧЕСКАЯ СХЕМА



РАСШИРЕНИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

```
specification class SqrtSpecification
{
  specification static double sqrt(double x)
    reads x
  {
    pre { return x >= 0; }
    post
    {
      if(x == 0)
      {
        branch "Zero argument";
        return sqrt == 0;
      }
      else
      {
        branch "Positive argument";
        return     sqrt >= 0
                && Math.abs((sqrt*sqrt-x)/x
                   < epsilon;
      }
    }
  }
}
```

сигнатура операции

ограничения доступа

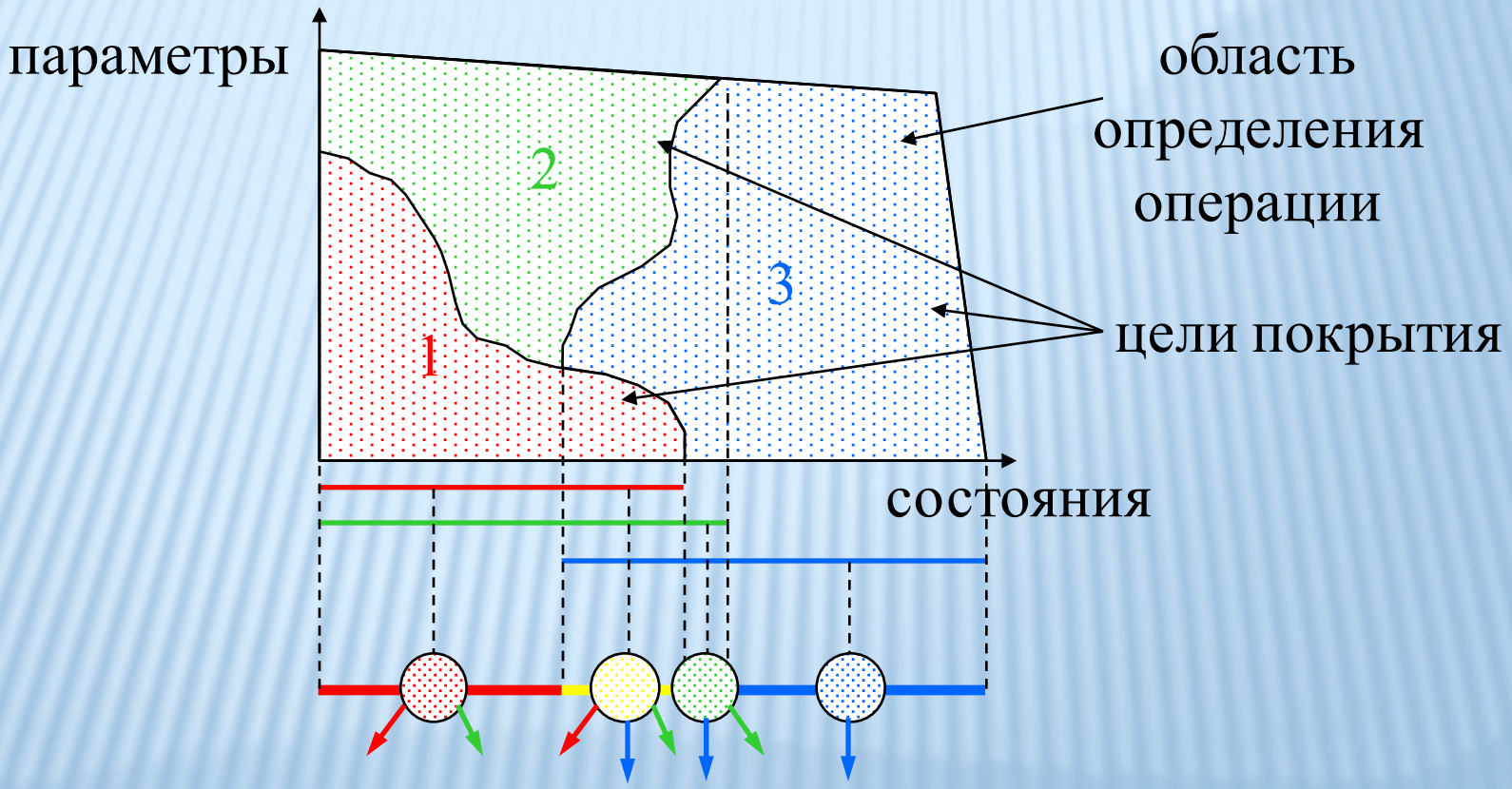
предусловие

```
specification double SQRT(double x)
  reads (double)x
{
  pre { return x >= 0.; }
  coverage ZP
  {
    if(x == 0.)
      return (ZERO, "Zero argument");
    else
      return (POS, "Positive argument");
  }
  post
  {
    if(coverage(ZP, ZERO))
      return SQRT == 0.;
    else
      return     SQRT >= 0.
              && abs((SQRT*SQRT - x)/x)
                 < epsilon;
    }
  }
}
```

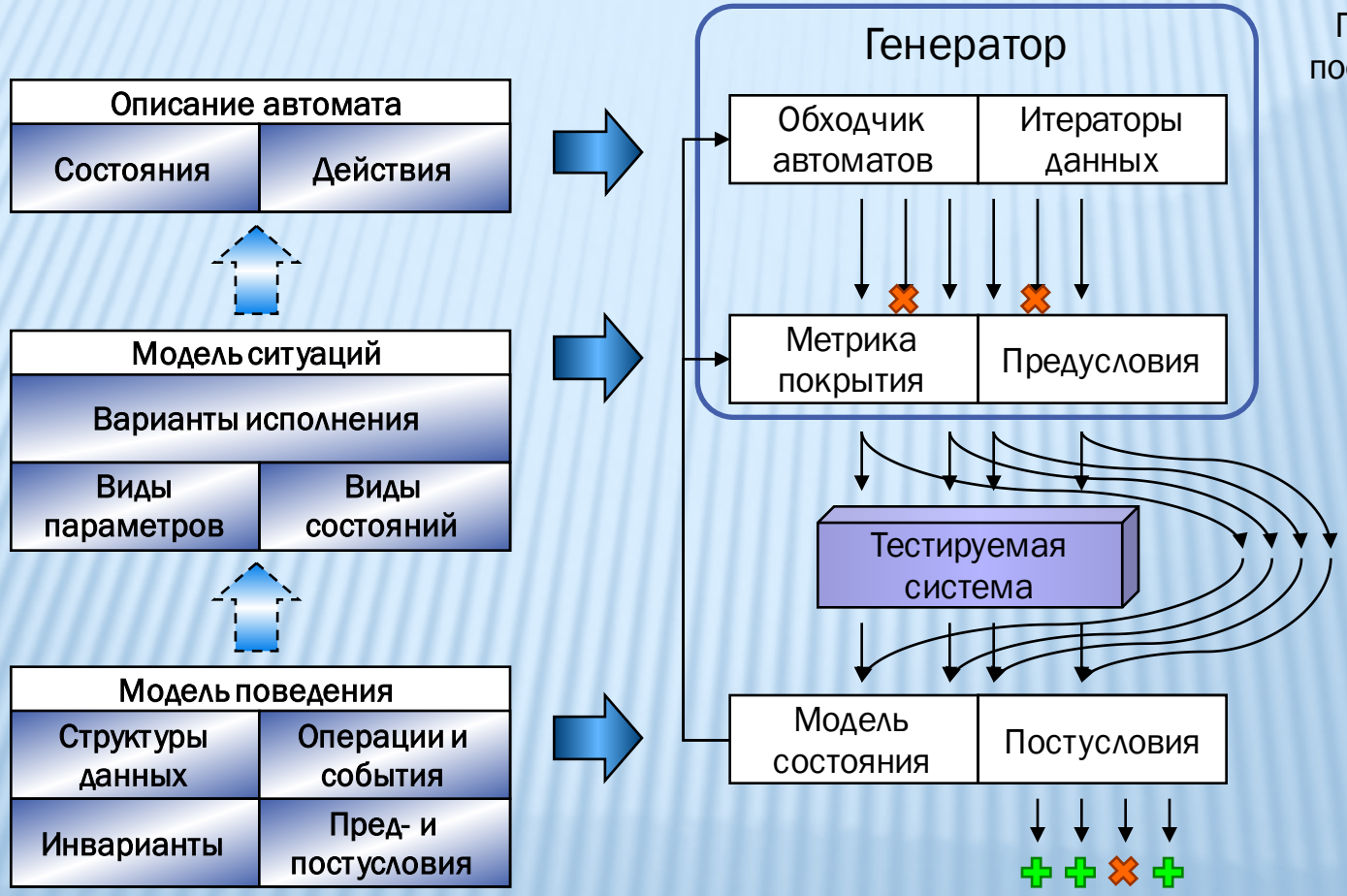
ветви функциональности

постусловие

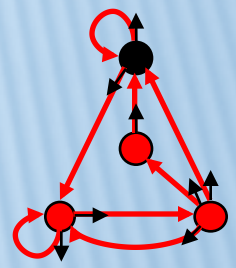
ПОСТРОЕНИЕ КОНЕЧНОГО АВТОМАТА



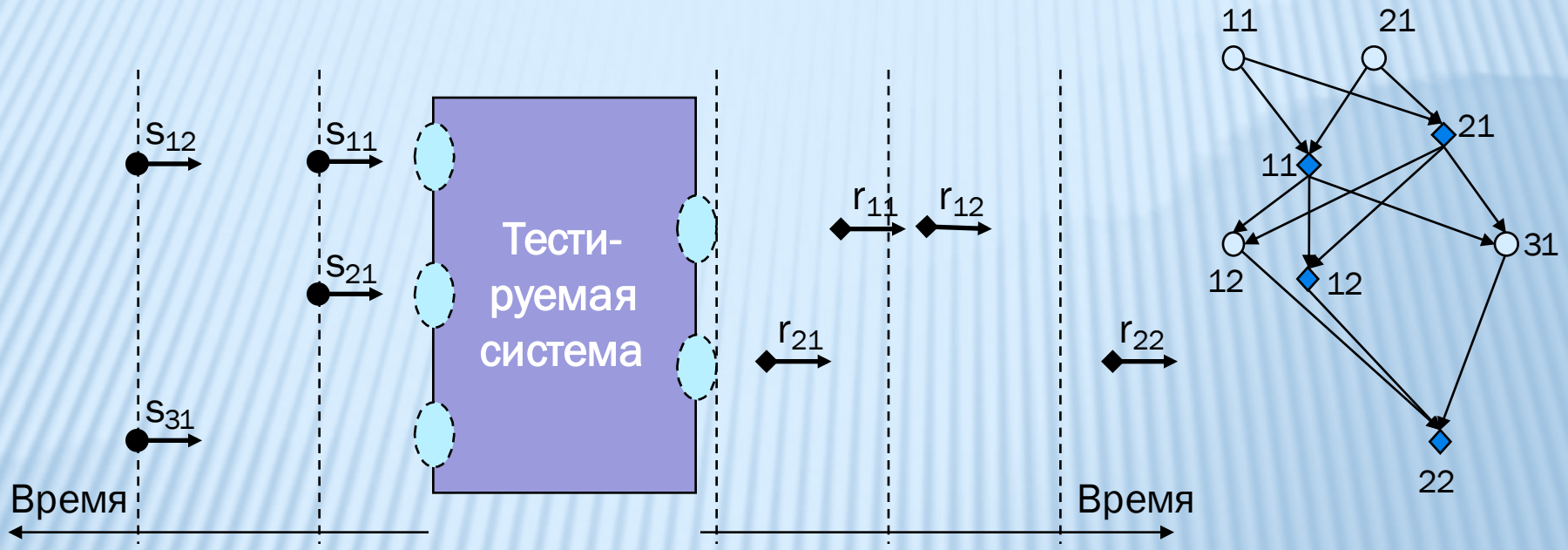
ТЕСТИРОВАНИЕ КОМПОНЕНТОВ



Генерация тестовой последовательности на лету

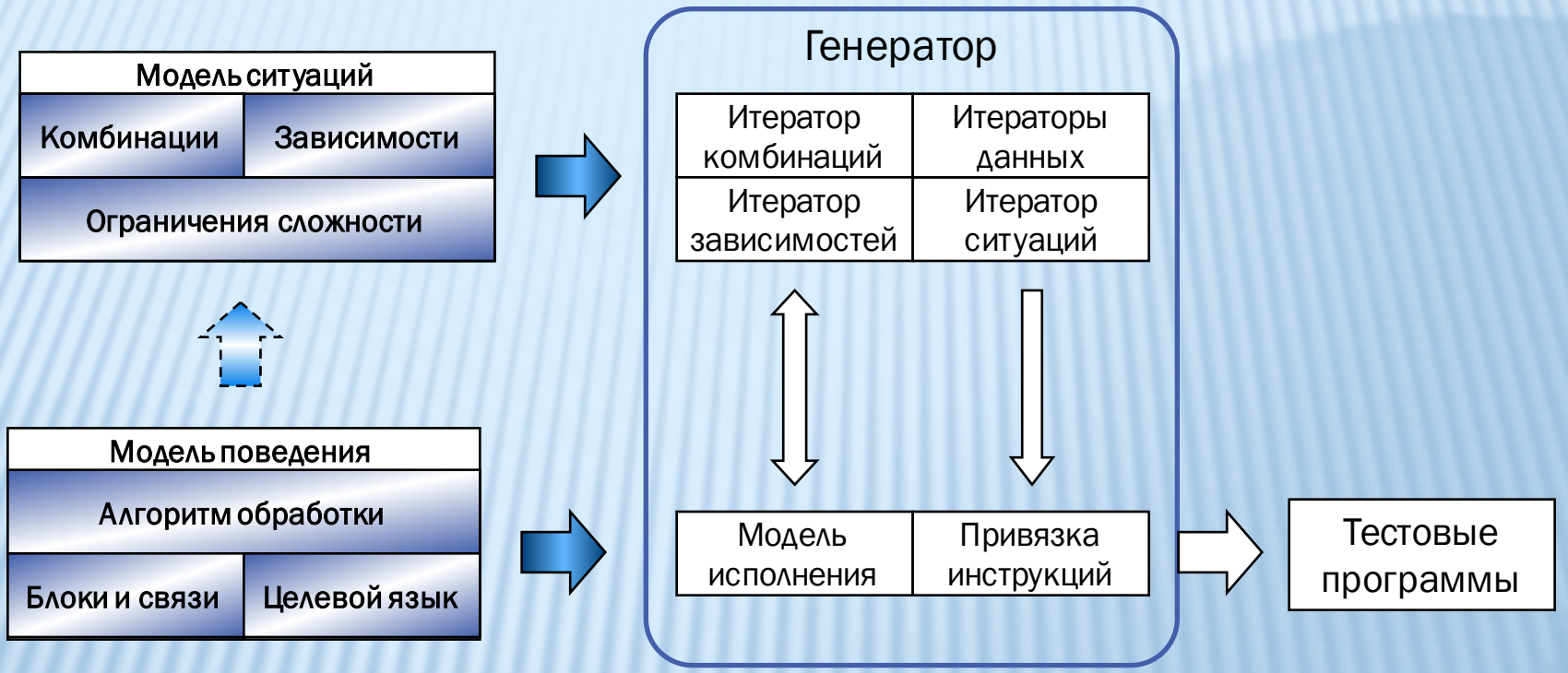


ТЕСТИРОВАНИЕ АСИНХРОННОСТИ



- Несколько последовательностей воздействий
- Частично упорядоченное множество событий
- Семантика чередования
  - это множество должно вытягиваться в последовательность

ГЕНЕРАЦИЯ ТЕСТОВЫХ ПРОГРАММ



СПАСИБО ЗА ВНИМАНИЕ!