Дедуктивная верификация системы защиты информации ОС Astra Linux

д.т.н., доцент Девянин П.Н. *УМО ИБ*

Мандрыкин М.У. *ИСП РАН*

д.ф.-м.н., профессор Петренко А.К. *ИСП РАН*

к.ф.-м.н. Хорошилов А.В. *ИСП РАН*

Постановка задачи исследования

Семейство ДП-моделей

Отечественная защищенная ОССН Astra Linux Special Edition

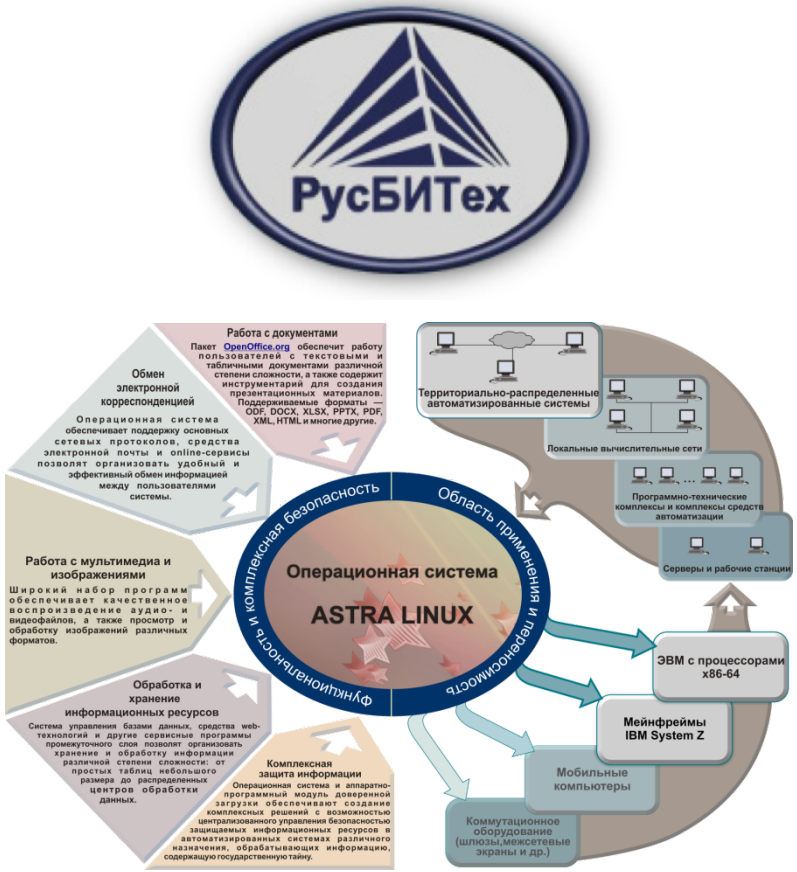
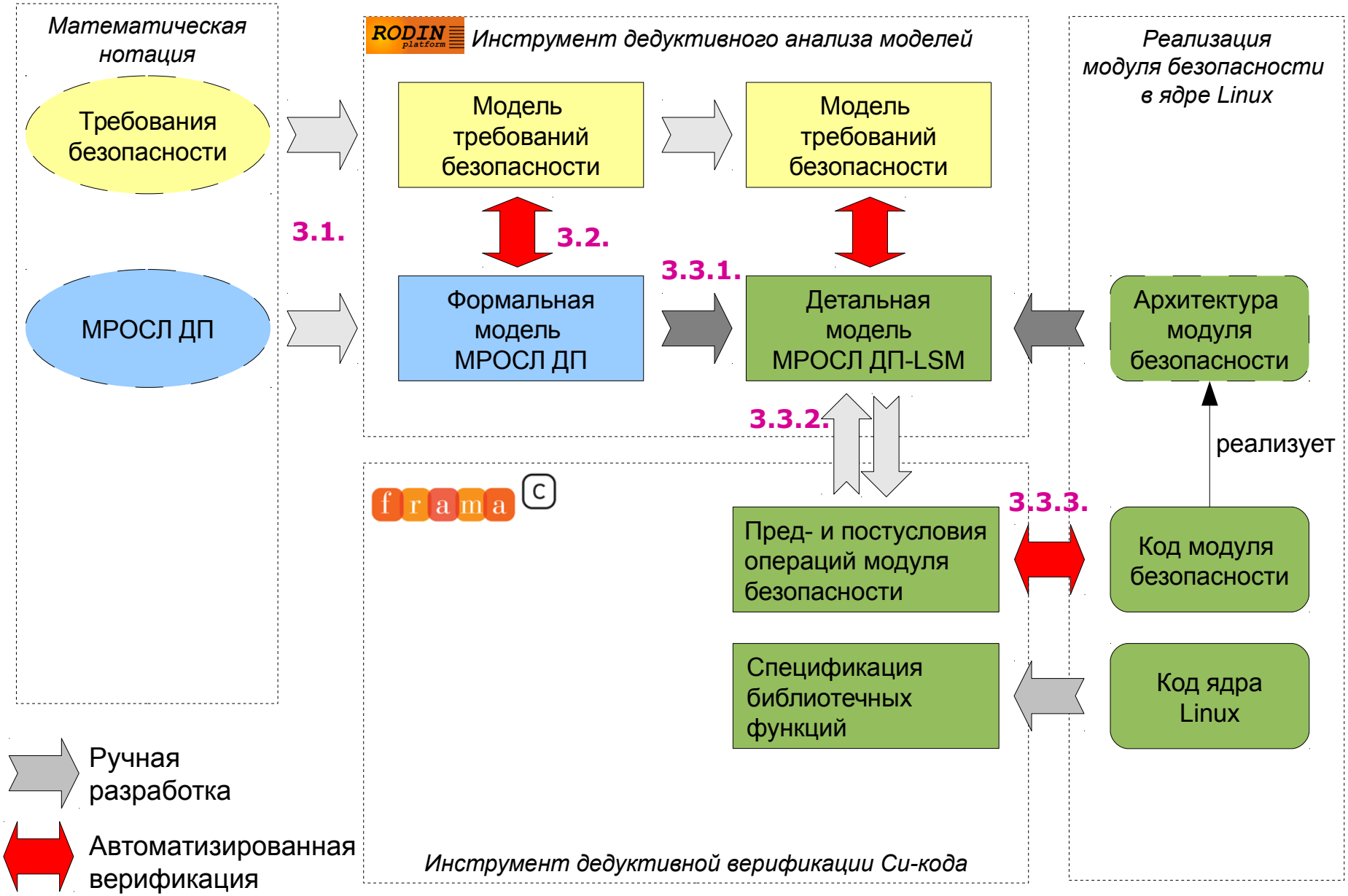


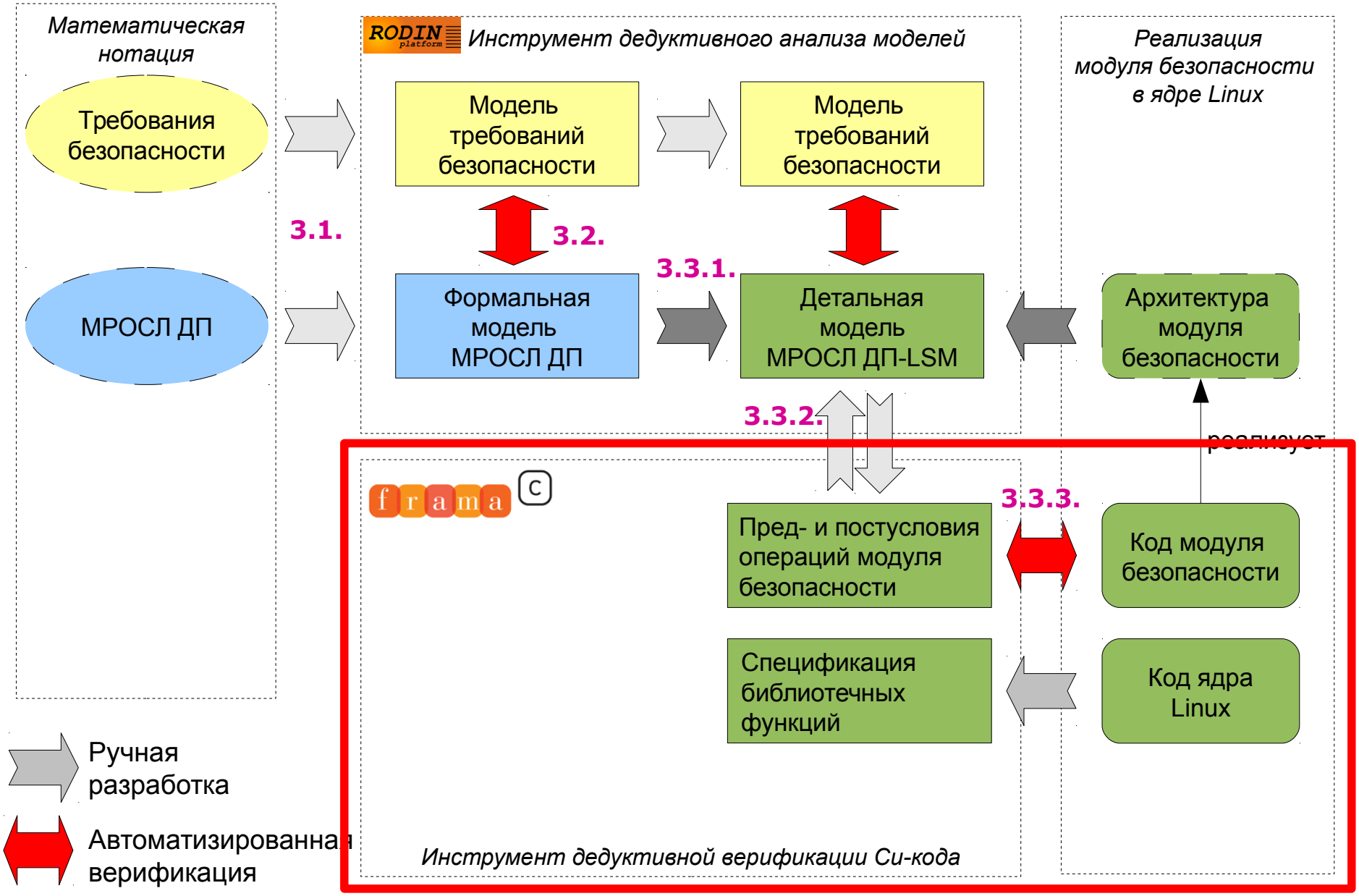
Схема решения задач исследования

Схема решения задач исследования

Обоснование адекватности реализации модели в коде ОССН



Обоснование адекватности реализации модели в коде ОССН



Дедуктивная верификация программ

Позволяет доказать корректность программы,

т. е.

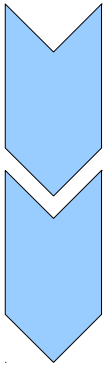
- *для любых входных данных*
- *для любого начального состояния*
- *для всех возможных поведений окружения*

программа завершается и выходные данные соответствуют постусловию

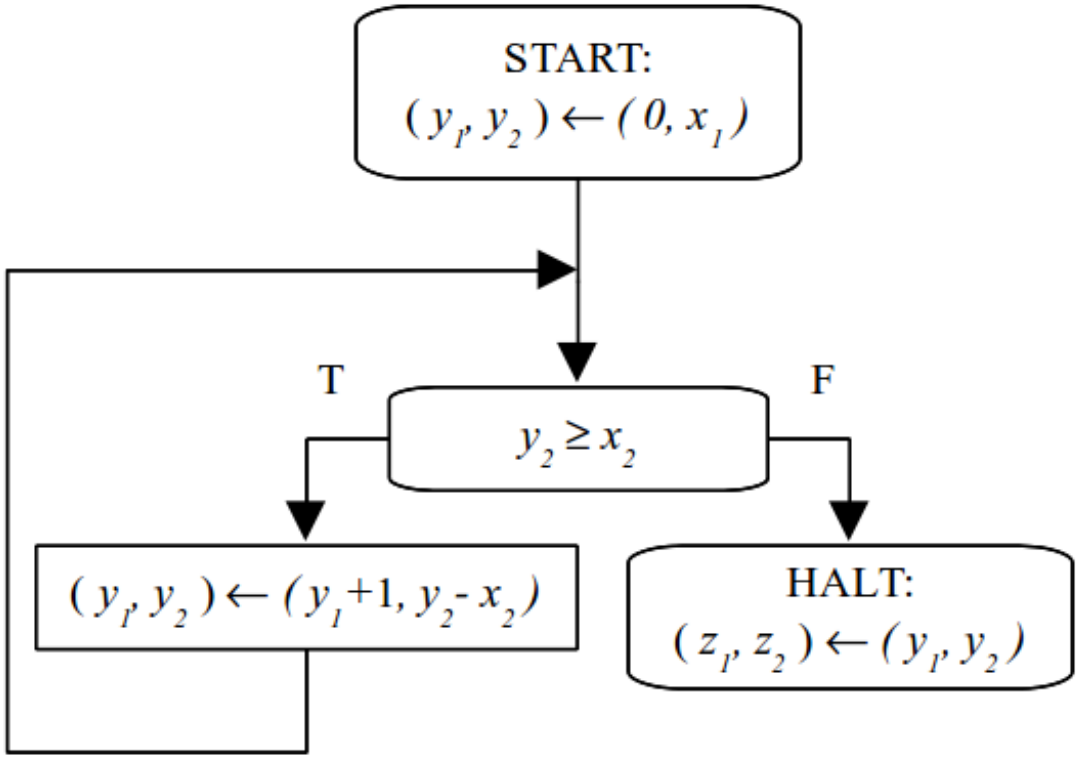
Если выполнены предположения

- *о входных данных программы*
- *о начальном состоянии*
- *о поведении окружения программы*
- *о работе компилятора*

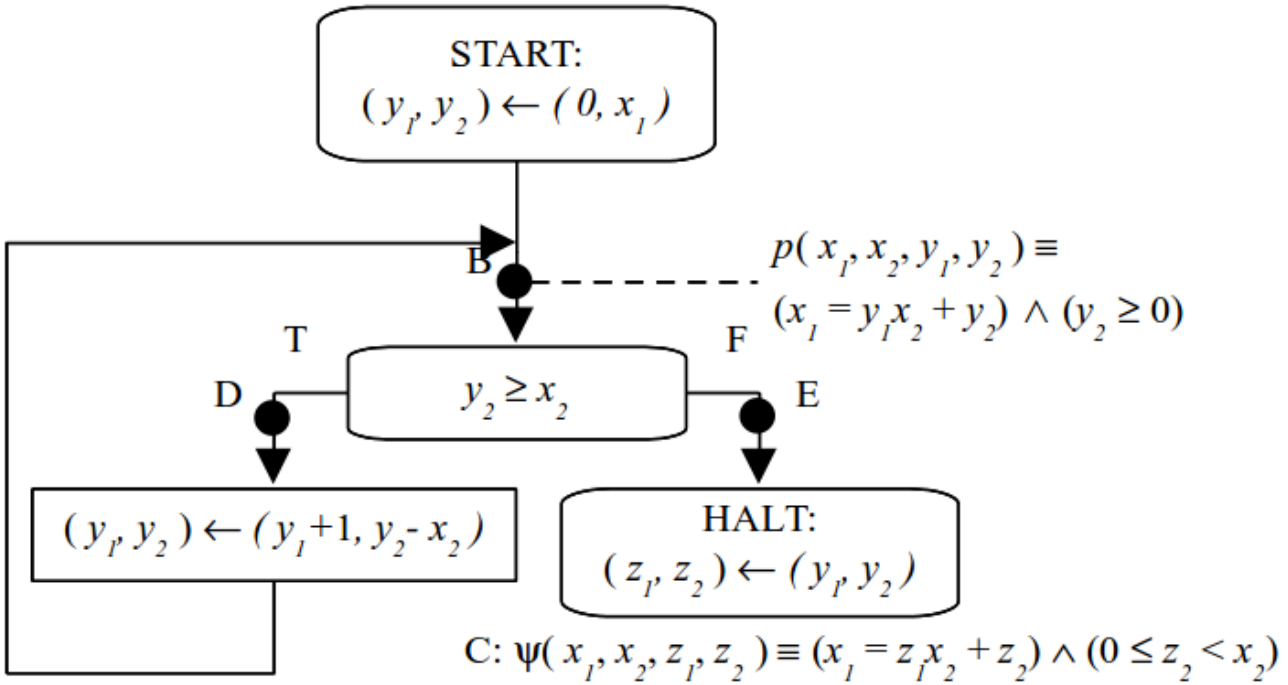
Дедуктивная верификация программ
историческая перспектива



- Лекция Алана Тьюринга Лондонскому математическому обществу
- Методы Флойда/Хоара


START:
$(y_1, y_2) \leftarrow (0, x_1)$
T
F
$y_2 \geq x_2$
$(y_1, y_2) \leftarrow (y_1+1, y_2-x_2)$
HALT:
$(z_1, z_2) \leftarrow (y_1, y_2)$

A: $\varphi_0(x_1, x_2) \equiv (x_1 \geq 0) \wedge (x_2 \geq 0)$



C: $\psi(x_1, x_2, z_1, z_2) \equiv (x_1 = z_1 x_2 + z_2) \wedge (0 \leq z_2 < x_2)$

VC1. $\forall\ x_1\ x_2\ \ (x_1 > 0) \wedge (x_2 > 0) \wedge \neg(x_1 > x_2) \wedge (x_1 = 0) \Rightarrow (\ x_2 = НОД(x_1, x_2)\ )$

VC2. $\forall\ x_1\ x_2\ \ (x_1 > 0) \wedge (x_2 > 0) \wedge (x_1 > x_2) \wedge (x_2 = 0) \Rightarrow (\ x_1 = НОД(x_1, x_2)\ )$

VC3. $\forall\ x_1\ x_2$

$(x_1 > 0) \wedge (x_2 > 0) \wedge \neg(x_1 > x_2) \wedge \neg(x_1 = 0) \Rightarrow (0 < x_1 \le x_2) \wedge (НОД(x_1, x_2) = НОД(x_1, x_2))$

VC4. $\forall\ x_1\ x_2$

$(x_1 > 0) \wedge (x_2 > 0) \wedge (x_1 > x_2) \wedge \neg(x_1 = 0) \Rightarrow (0 < x_2 \le x_1) \wedge (НОД(x_1, x_2) = НОД(x_2, x_1))$

VC5. $\forall\ x_1\ x_2\ y_1\ y_2\ (0 < y_1 \le y_2) \wedge (НОД(x_1, x_2) = НОД(y_1, y_2)) \wedge \neg(\mathrm{rem}(y_2, y_1) = 0)$

$\Rightarrow (0 < \mathrm{rem}(y_2, y_1) \le y_1) \wedge (НОД(x_1, x_2) = НОД(\mathrm{rem}(y_2, y_1), y_1))$

VC6. $\forall\ x_1\ x_2\ y_1\ y_2\ (0 < y_1 \le y_2) \wedge (НОД(x_1, x_2) = НОД(y_1, y_2)) \wedge (\mathrm{rem}(y_2, y_1) = 0)$

$\Rightarrow (\ y_1 = НОД(x_1, x_2)\ )$

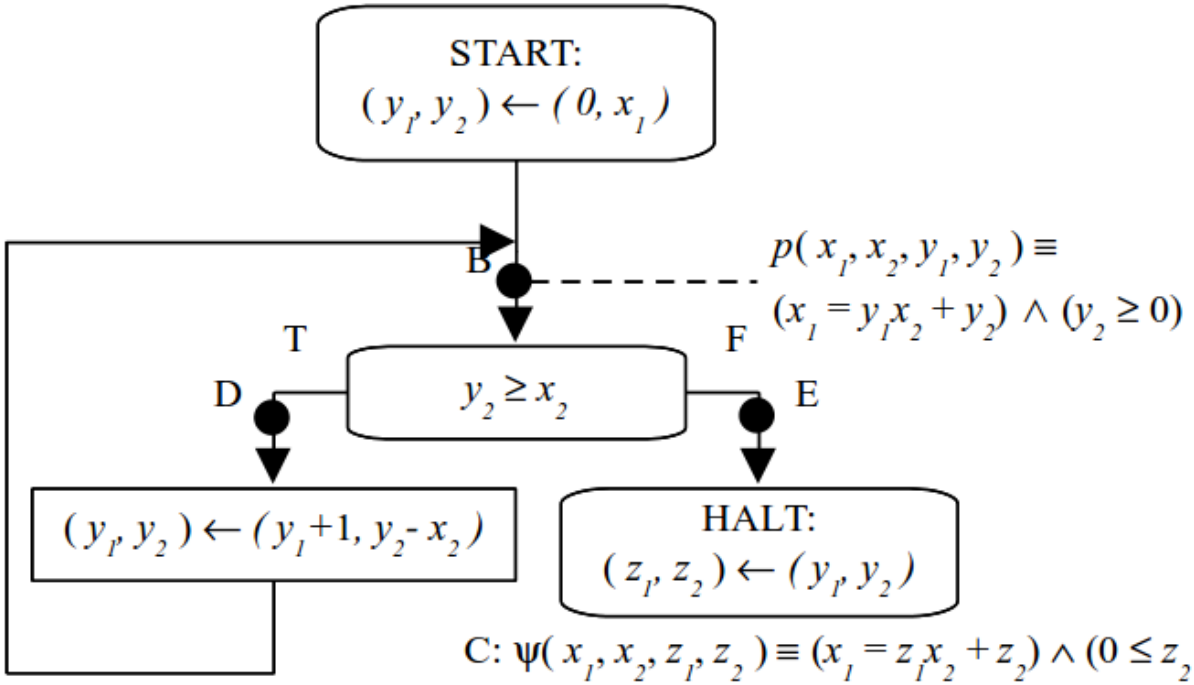
VC7. $\forall\ x_1\ x_2\ y_1\ y_2\ \ (0 < y_1 \le y_2) \wedge (НОД(x_1, x_2) = НОД(y_1, y_2)) \wedge \neg\ (\mathrm{rem}(y_2, y_1) = 0)$

$\Rightarrow (y_1 > 0)$

VC8. $\forall\ x_1\ x_2\ y_1\ y_2\ \ (0 < y_1 \le y_2) \wedge (НОД(x_1, x_2) = НОД(y_1, y_2)) \wedge \neg\ (\mathrm{rem}(y_2, y_1) = 0)$

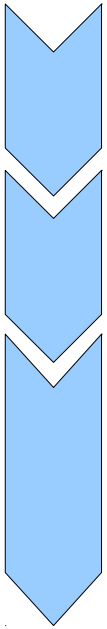
$\Rightarrow (y_1 > \mathrm{rem}(y_2, y_1))$

A: $\varphi_0($



C: $\psi(x_1, x_2, z_1, z_2) \equiv (x_1 = z_1 x_2 + z_2) \wedge (0 \le z_2 < x_2)$

Дедуктивная верификация программ
историческая перспектива

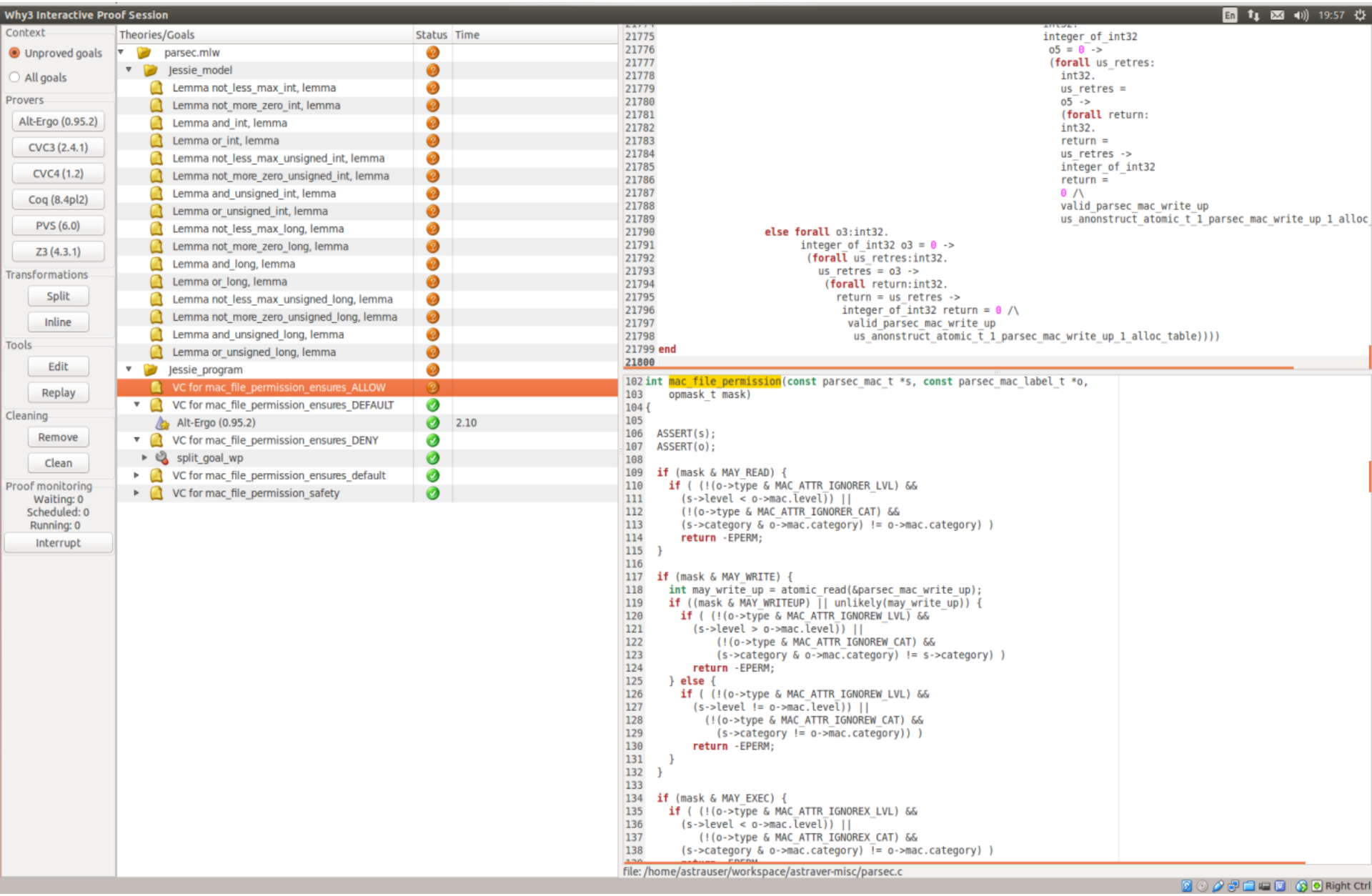


1947
- Лекция Алана Тьюринга Лондонскому математическому обществу

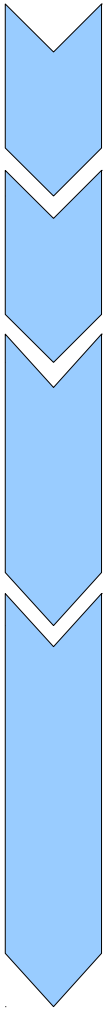
1970
- Методы Флойда/Хоара

2000
- Инструменты дедуктивной верификации для Си, Java, C#
  - SunRise, ESC/Java, Frama-C, LOOP, Boogie/VCC

```
/*@ predicate is_divisor(int m, int n) { if (m) then (n % m == 0) else \false } */

/*@ predicate is_gcd(int z, int x1, int x2) {
  @         is_divisor(z,x1)
  @      && is_divisor(z,x2)
  @      && \forall int i; is_divisor(i,x1) && is_divisor(i,x2) => (i <= z) }
  @*/

/*@ logic int gcd(int m, int n) */
/*@ axiom gcd_def: \forall int m; \forall int n; is_gcd(gcd(m,n),m,n) */


/* The function returns the greatest common divisor of x1 and x2 */
/*@ requires (x1 > 0) && (x2 > 0)
  @ ensures is_gcd(\result,x1,x2)
  @*/
int nod(int x1, int x2)
{
int y1 = x1;
int y2 = x2;
int tmp = 0;

  if (y1 > y2) {
    y1 = x2;
    y2 = x1;
  }
  /*@
    @ invariant ((0 < y1 <= y2) && gcd(x1,x2) == gcd(y1,y2))
    @         || ((y1 == 0) && gcd(x1,x2) == y2)
    @ variant y1
    @*/
  while (y1 != 0) {
    tmp = y1;
    y1 = y2%y1;
    y2 = tmp;
  }
```


Why3 Interactive Proof Session
Context
Unproved goals
All goals
Provers
Alt-Ergo (0.95.2)
CVC3 (2.4.1)
CVC4 (1.2)
Coq (8.4pl2)
PVS (6.0)
Z3 (4.3.1)
Transformations
Split
Inline
Tools
Edit
Replay
Cleaning
Remove
Clean
Proof monitoring
Waiting: 0
Scheduled: 0
Running: 0
Interrupt
Theories/Goals
Status
Time
parsec.mlw
Jessie_model
Lemma not_less_max_int, lemma
Lemma not_more_zero_int, lemma
Lemma and_int, lemma
Lemma or_int, lemma
Lemma not_less_max_unsigned_int, lemma
Lemma not_more_zero_unsigned_int, lemma
Lemma and_unsigned_int, lemma
Lemma or_unsigned_int, lemma
Lemma not_less_max_long, lemma
Lemma not_more_zero_long, lemma
Lemma and_long, lemma
Lemma or_long, lemma
Lemma not_less_max_unsigned_long, lemma
Lemma not_more_zero_unsigned_long, lemma
Lemma and_unsigned_long, lemma
Lemma or_unsigned_long, lemma
Jessie_program
VC for mac_file_permission_ensures_ALLOW
VC for mac_file_permission_ensures_DEFAULT
Alt-Ergo (0.95.2)
2.10
VC for mac_file_permission_ensures_DENY
split_goal_wp
VC for mac_file_permission_ensures_default
VC for mac_file_permission_safety
integer_of_int32
o5 = 0 ->
(forall us_retres:
int32.
us_retres =
o5 ->
(forall return:
int32.
return =
us_retres ->
integer_of_int32
return =
0 /\
valid_parsec_mac_write_up
us_anonstruct_atomic_t_1_parsec_mac_write_up_1_alloc
else forall o3:int32.
integer_of_int32 o3 = 0 ->
(forall us_retres:int32.
us_retres = o3 ->
(forall return:int32.
return = us_retres ->
integer_of_int32 return = 0 /\
valid_parsec_mac_write_up
us_anonstruct_atomic_t_1_parsec_mac_write_up_1_alloc_table))))
end
int mac_file_permission(const parsec_mac_t *s, const parsec_mac_label_t *o,
opmask_t mask)
{
ASSERT(s);
ASSERT(o);
if (mask & MAY_READ) {
if ((!(o->type & MAC_ATTR_IGNORER_LVL) &&
(s->level < o->mac.level)) ||
(!(o->type & MAC_ATTR_IGNORER_CAT) &&
(s->category & o->mac.category) != o->mac.category))
return -EPERM;
}
if (mask & MAY_WRITE) {
int may_write_up = atomic_read(&parsec_mac_write_up);
if ((mask & MAY_WRITEUP) || unlikely(may_write_up)) {
if ((!(o->type & MAC_ATTR_IGNOREW_LVL) &&
(s->level > o->mac.level)) ||
(!(o->type & MAC_ATTR_IGNOREW_CAT) &&
(s->category & o->mac.category) != s->category))
return -EPERM;
} else {
if ((!(o->type & MAC_ATTR_IGNOREW_LVL) &&
(s->level != o->mac.level)) ||
(!(o->type & MAC_ATTR_IGNOREW_CAT) &&
(s->category != o->mac.category)))
return -EPERM;
}
}
if (mask & MAY_EXEC) {
if ((!(o->type & MAC_ATTR_IGNOREX_LVL) &&
(s->level < o->mac.level)) ||
(!(o->type & MAC_ATTR_IGNOREX_CAT) &&
(s->category & o->mac.category) != o->mac.category))
file: /home/astrauser/workspace/astraver-misc/parsec.c
Right Ctrl

Дедуктивная верификация программ
историческая перспектива



1947
- Лекция Алана Тьюринга Лондонскому математическому обществу

1970
- Методы Флойда/Хоара

2000
- Инструменты дедуктивной верификации для Си, Java, C#
  - SunRise, ESC/Java, Frama-C, LOOP, Boogie/VCC

2010
- Применение к реальным проектам небольшого размера
  - Атомная энергетика (Англия, Франция)
  - Авионика (Airbus, NASA)
  - Компоненты специализированных ОС (seL4, Hyper-V)

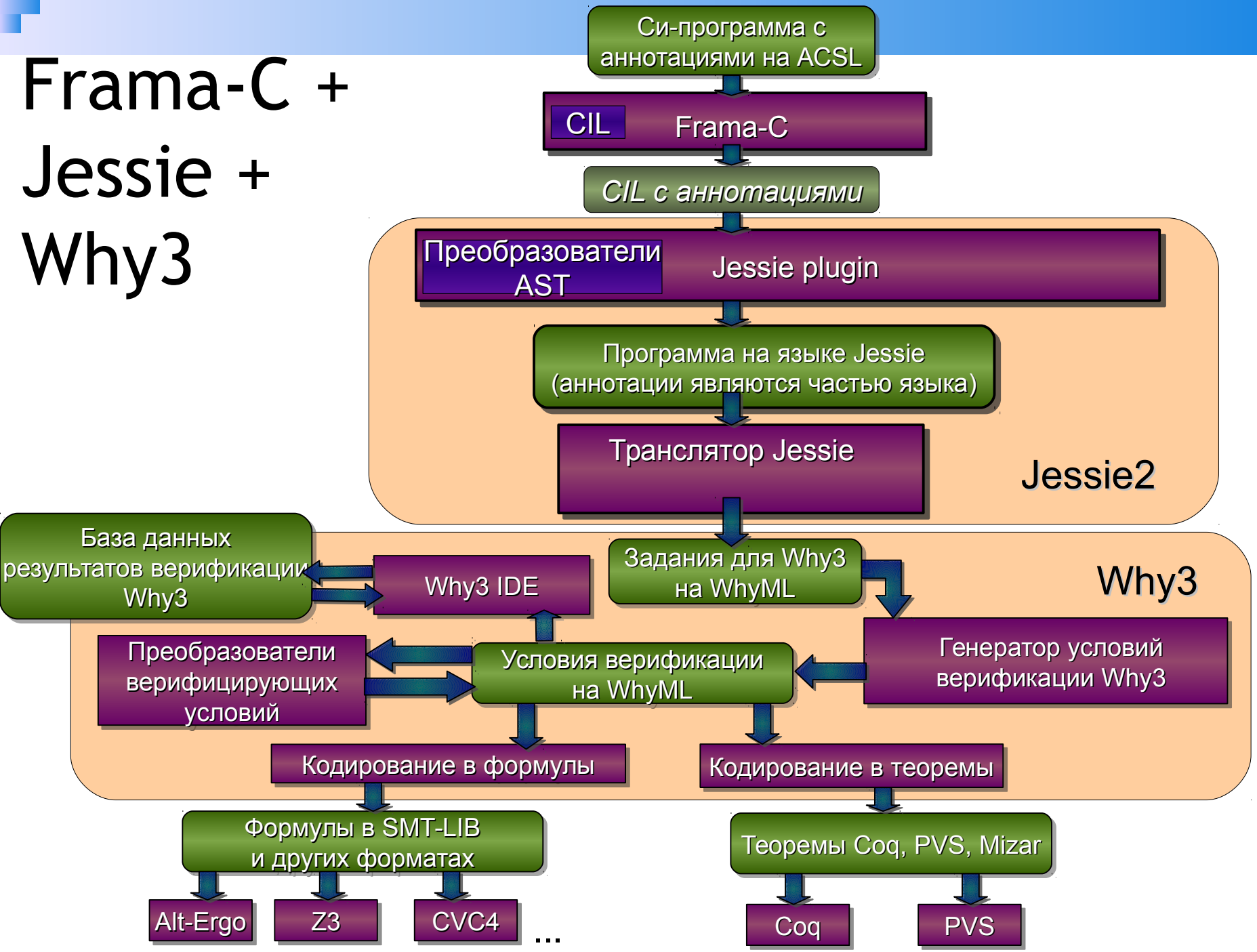
Ограничения дедуктивной верификации программ

- Трудоёмкость
- Инструменты поддерживают не все конструкции языков программирования
- Применяется для программ небольшого размера
  - обычно не более 10 тыс. строк
  - iFACTS Project (air traffic, UK) 250 KLOC

Инструменты дедуктивной верификации Си программ

	Open source	Модель памяти	Разделяемые данные	Опыт применения к коду ядра	Удобство использования
VCC	–	+	±	+	–
Why3	+	+	–	–	+
Frama-C WP	+	∓	–	–	+
VeriFast	–	+	∓	∓	–
C-to-Isabelle	+	+	–	+	±

Frama-C + Jessie + Why3



Проблемы

- использование указателей на внутренние поля структур и адресная арифметика с такими указателями
- наличие неявных преобразований типов посредством обращения к содержимому одной области памяти как к объектам разных типов
- использование функциональных указателей

Предложенные решения

Высокоуровневая модель с регионами

```
f(int *a, int *d,
  char *c)
{
  int *b;
  int n, m;

  ……

  b = a;
  b[n] = 1;
  d[m] = 2;

  c[1] = 'a';
}
```

предусловие

$(a_0 \le d_0 - 16 \vee a_0 \ge d_0 + 16)$

……

$b_1 = a_0 \wedge$

$M^{int}_{a0,1} = M^{int}_{a0,0}[b_1 +_{32} n \leftarrow 0_8] \wedge$

$M^{int}_{a0,2} = M^{int}_{a0,1}[b_1 +_{32} n +_{32} 1_{32} \leftarrow 0_8] \wedge$

$M^{int}_{a0,3} = M^{int}_{a0,2}[b_1 +_{32} n +_{32} 2_{32} \leftarrow 0_8] \wedge$

$M^{int}_{a0,4} = M^{int}_{a0,3}[b_1 +_{32} n +_{32} 3_{32} \leftarrow 1_8] \wedge$

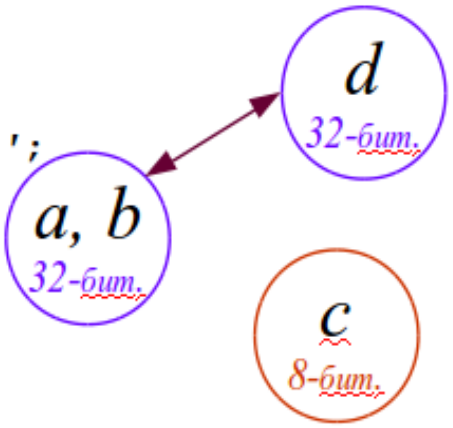
$M^{int}_{d0,1} = M^{int}_{d0,0}[d_0 +_{32} m \leftarrow 0_8] \wedge$

$M^{int}_{d0,2} = M^{int}_{d0,1}[d_0 +_{32} m +_{32} 1_{32} \leftarrow 0_8] \wedge$

$M^{int}_{d0,3} = M^{int}_{d0,2}[d_0 +_{32} m +_{32} 2_{32} \leftarrow 0_8] \wedge$

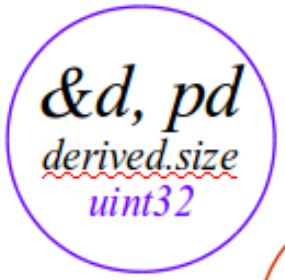
$M^{int}_{d0,4} = M^{int}_{d0,3}[d_0 +_{32} m +_{32} 3_{32} \leftarrow 2_8] \wedge$

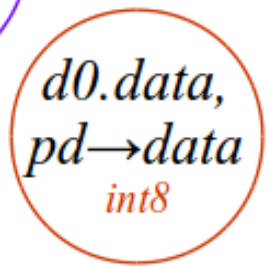
$M^{char}_{c0,1} = M^{char}_{c0,0}[c_0 +_{32} 1_{32} \leftarrow 97_8]$



Поддержка структур

```
struct derived {
        size_t size;
        char data[8];
};

struct derived *d;
struct derived *pd = &d;
.....

  pd->data[1] = 0;
  pd->size = 2;
```





.....

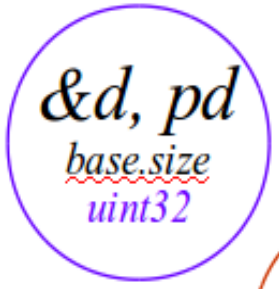
$pd_1 = d_0 /\!\!/$

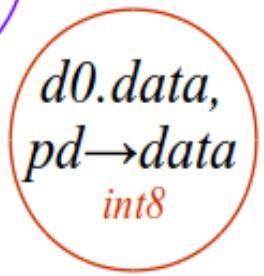
$M^{int8}_{d0.data,\ 1} = M^{int8}_{d0.data,\ 0}[M^{int8*}_{(d0,\ derived.data),\ 0}[pd_0] + 1 \leftarrow 0] /\!\!/$

$M^{uint32}_{(d0,\ derived.size),\ 1}[pd_0] = M^{uint32}_{(d0,\ derived.size),\ 0}[pd_0 \leftarrow 2]$

Поддержка префиксного кастирования

```
struct base
        size_t size;
};

struct derived {
        struct base base;
        char data[8];
};

struct derived *d;
struct derived *pd = &d;
```





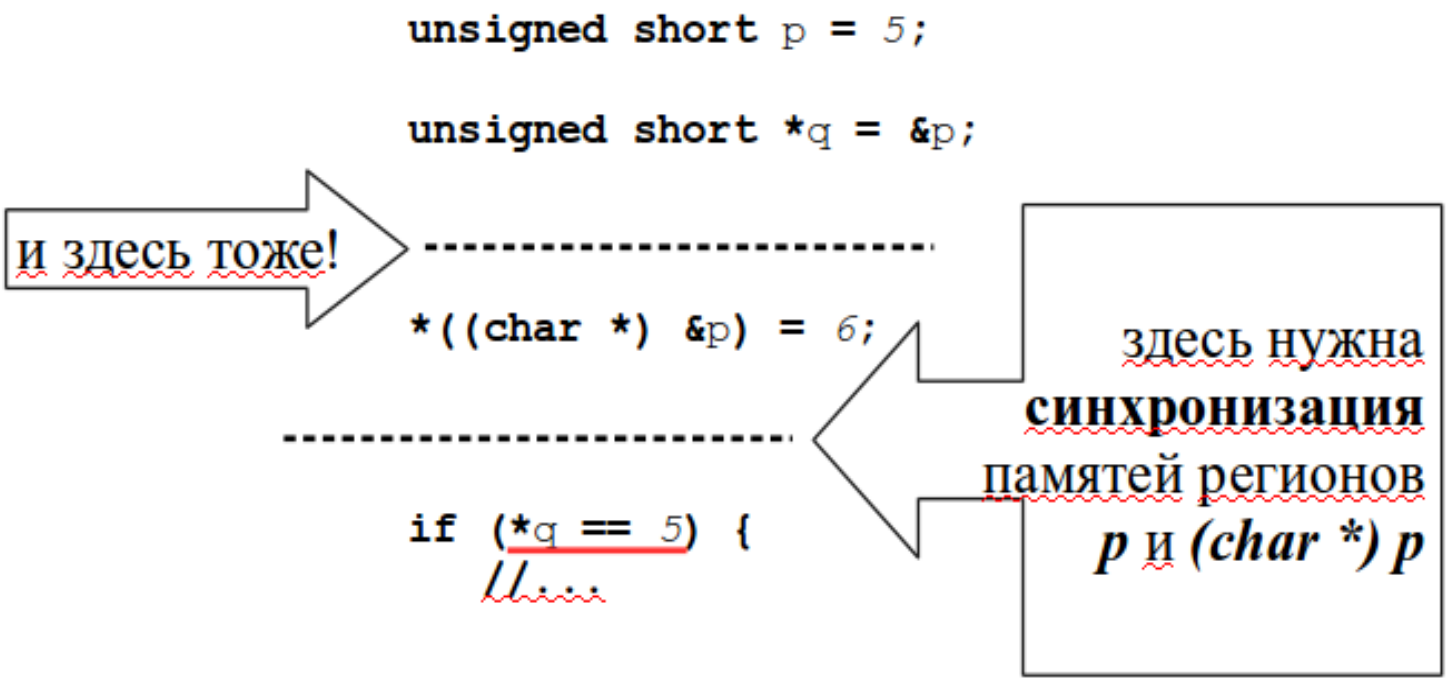

```
((struct base *) pd)->size = 2;
```

$$M^{uint32}_{(d0,\ base.size),\ 1}[pd_0] = M^{uint32}_{(d0,\ base.size),\ 0}[pd_0 \leftarrow 2]$$

Проблема когерентности обновлений

```
unsigned short p = 5;

unsigned short *q = &p;


*((char *) &p) = 6;


if (*q == 5) {
    //...
```

Проблема когерентности обновлений



Проблема когерентности обновлений

```
unsigned short p = 5;

unsigned short *q = &p;

//@ jessie pragma p :> char *;

*((char *) &p) = 6;

//@ jessie pragma ((char * )&p) :> unsigned short *;

if (*q == 5) {
   //...
```

Подробнее:
М.У. Мандрыкин, А.В. Хорошилов «Высокоуровневая модель памяти промежуточного языка Jessie с поддержкой произвольного приведения типов указателей» // Программирование, 2015, №4

Поддержка функциональных указателей

- Перебор всех подходящих функций
  - в предположение о том, что значениями функциональных указателей могут быть только адреса каких-либо функций из анализируемого кода

Поддержка строковых литералов

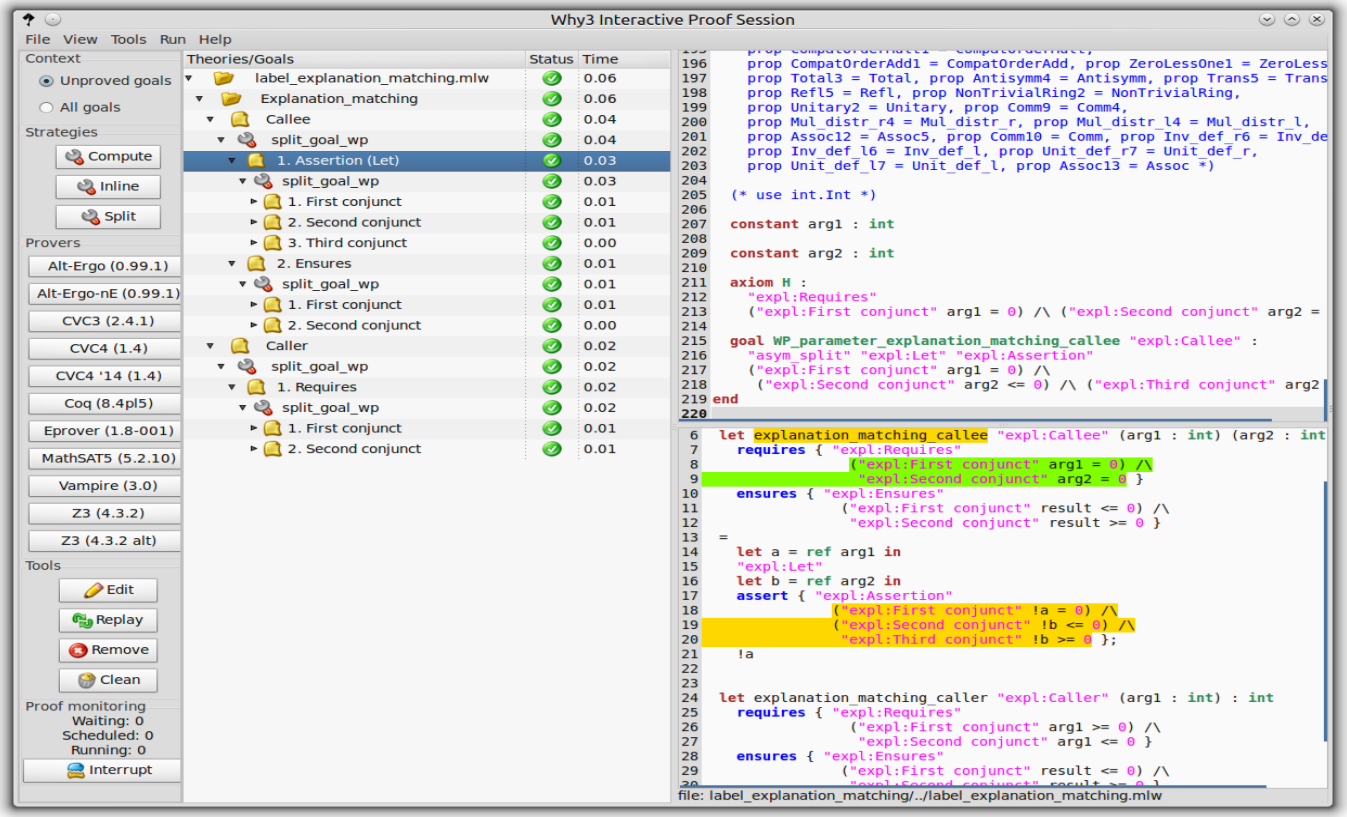
- Прокси-переменные для строковых литералов
  - с генерацией соответствующих инвариантов на содержимое

Масштабируемость

- Поддержка больших программ (> 10 000 строк кода), где лишь небольшая часть кода существенно необходима для анализа
  - масштабируемость достигается за счет удаления несущественных глобальных объявлений и определений перед началом основных существенных преобразований кода

Удобство использования

- Подсветка условий пути



О НАС
- О центре
- Наша команда
- Новости
- Партнеры
- Контакты

Проекты
- Верификация модулей ядра
- Инфраструктура LSB
- Технологии тестирования
- Тесты и средства их запуска
- Инструменты обеспечения переносимости

Результаты
- Обнаруженные проблемы
- Публикации
- Мероприятия

Ссылки
- Стандарты Linux
- Похожие проекты
- Это интересно

khoroshilov
- Мой профиль
- Список пользователей
- Создать материал
- Feed aggregator
- Администрирование
- Выйти

18.02.2015: Первый публичный выпуск Astraver Toolset

Просмотр | Редактировать | Следить | Переводы

Опубликовано Mikhail Mandrykin в Втр, 10/03/2015 - 13:00

Состоялся первый публичный выпуск системы верификации **Astraver Toolset 1.0**, построенной на базе набора инструментов дедуктивной верификации '**Frama-C** + **Jessie** + **Why3 IDE**'. Инструменты были адаптированы для спецификации и доказательства свойств в коде ядра ОС Linux. Большинство наших изменений затронули встраиваемый модуль Jessie, в то время как во внешний интерфейс Frama-C и платформу Why3 были внесены лишь небольшие исправления и улучшения. Некоторые из наших изменений уже интегрированы в официальную версию инструментов, в то время как остальные доступны в наших **публичных репозиториях**.

Основные изменения описаны ниже.

Поддержка языка Си

- Поддержка низкоуровневых приведений типов указателей на целочисленные типы данных. Добавление этой возможности потребовало модификации модели памяти Jessie в соответствии с описанием в статье "Расширенная высокоуровневая Си-совместимая модель памяти для промежуточного языка Jessie с ограниченной поддержкой низкоуровневого приведения типов указателей". Основная идея заключается в предоставлении пользователю возможности производить некоторые теневые перевыделения блоков памяти в явно указанных точках с целью преобразования выделенных массивов одного типа в другой.
  ВНИМАНИЕ. Поддержка различаемых объединений (union) пока еще не была полностью адаптирована для новой модели памяти.
- Префиксное приведение типов между объемлющими структурами и структурами, вложенными в них в качестве первых полей. Реализовано при помощи встраивания полей и установление отношения наследования между структурами.
- Поддержка функций выделения/освобождения памяти в адресном пространстве ядра (kmalloc()/kzalloc(), kfree()).
- Встроенный в C99 тип __Bool.
- Функции memcpy(), memmove(), memcmp() and memset() из стандартной библиотеки. Поддержка основана на использовании нескольких предопределенных шаблонных спецификаций, в которые подставляются различные конкретные типы данных.(*)
- Указатели на функции (через полный перебор возможных алиасов). (*)
- Функции с переменным числом параметров (через дополнительный параметр-массив). (*)
- Ассемблерные вставки (через вызовы неопределенных функций). (*)

> *(*)Основной целью реализации поддержки этих возможностей был запуск инструмента на целевом коде без необходимости его значительной предварительной модификации. Как следствие, поддержка перечисленных возможностей не достаточно полна для верификации кода, в котором они используются существенным*

AstraVer Toolset

- Система верификации на основе Frama-C+Jessie+Why3 опубликована под свободной лицензией

  http://linuxtesting.ru/astraver

Заключение

- Современные инструменты дедуктивной верификации приближаются к возможности доказывать корректность функций из ядра
- По-прежнему высокая трудоёмкость
- Имеет смысл для наиболее критичных компонентов
- Открытые проблемы
  - Поддержка верификации функций, работающих с разделяемыми данными
    - т. е. с данными, которые могут быть изменены параллельно выполняющимися функциями

LinuxTesting.org

Спасибо!

ИСП РАН

Институт системного программирования РАН

http://linuxtesting.ru/astraver