Федеральное государственное бюджетное учреждение науки
Институт системного программирования Российской академии наук

ИСП РАН

Методы верификации программ на основе композиции задач достижимости

Мордань Виталий Олегович

Научный руководитель:
д.ф.-м.н., проф. Петренко Александр Константинович

Москва
25 мая 2017 года

Статическая верификация программного обеспечения

- Проверка кода без его выполнения
- Рассмотрение всех путей программы
- Доказательство отсутствия ошибок
- Недостатки
  - Высокие требования на ресурсы
  - Неразрешимость задачи в общем случае

Актуальность

- Верификация модулей ядра ОС Linux
  - Более 20 млн. строк кода
  - Новая версия выходит каждые 2-3 месяца
  - Около 5 000 модулей
- Сотни требований на корректное использование интерфейсов

- Задача эффективного использования ресурсов при верификации

Задача достижимости

- Требования к программе (свойства безопасности)
  - *Смещение не должно превышать размер массива в параметрах функций вида find_next_bit()*
- Сведение к задаче достижимости
  - Добавление дополнительных проверок в программу
  - Достижение метки ошибки → нарушение требования

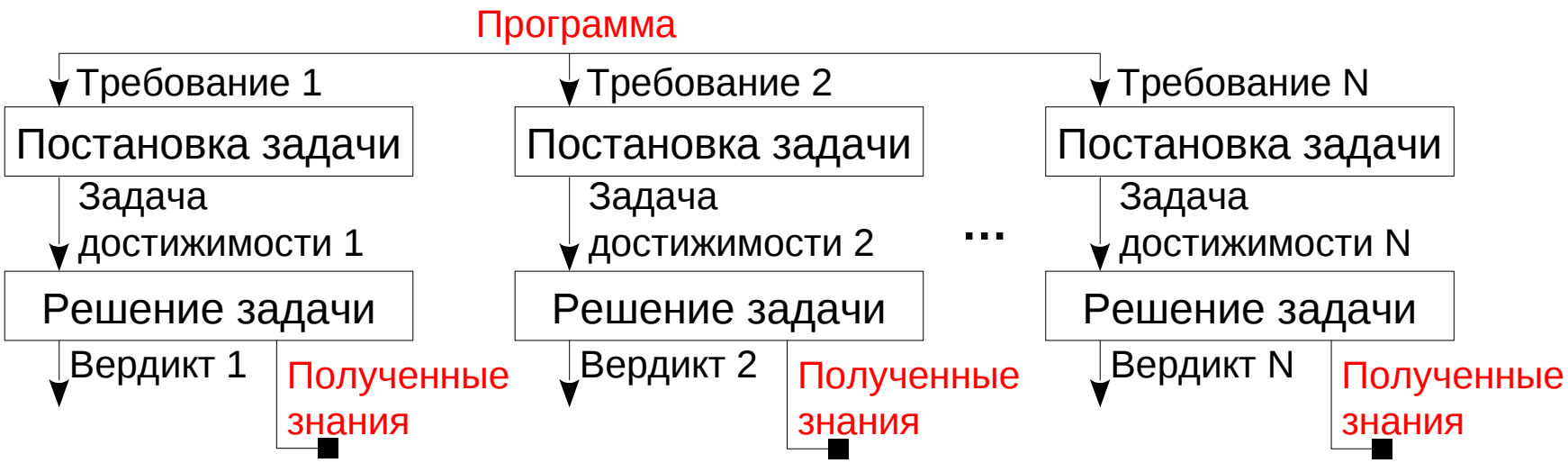
Исходная программа

```
...
assert(offset < size);
...
```

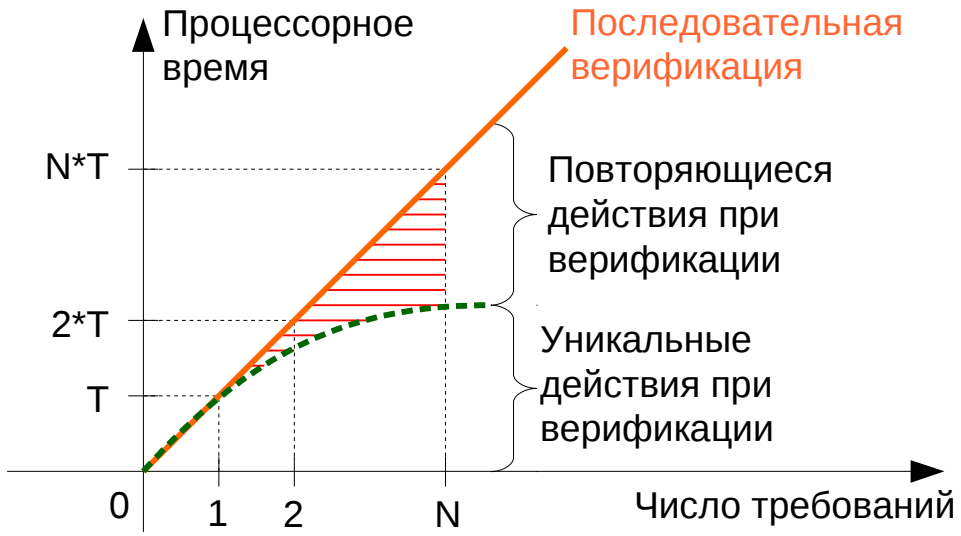
Задача достижимости

```
...
if(offset >= size) {
  ERROR: error();
  // метка ошибки
}
...
```

Последовательная верификация



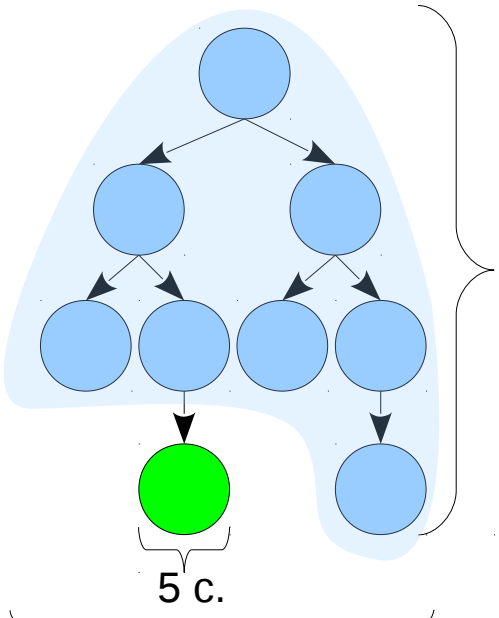
Теряются полученные «знания» о верификации программы



«Знания» о верификации

Последовательная верификация

Верификация требования **1**

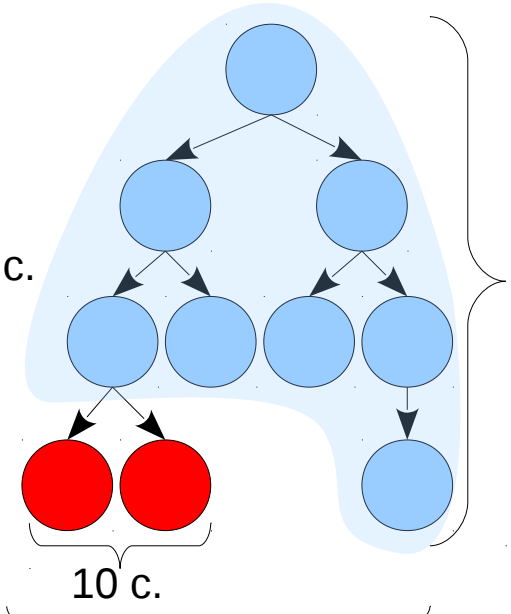


105 с.

4 пути выполнения

\+

Верификация требования **2**



110 с.

5 путей выполнения

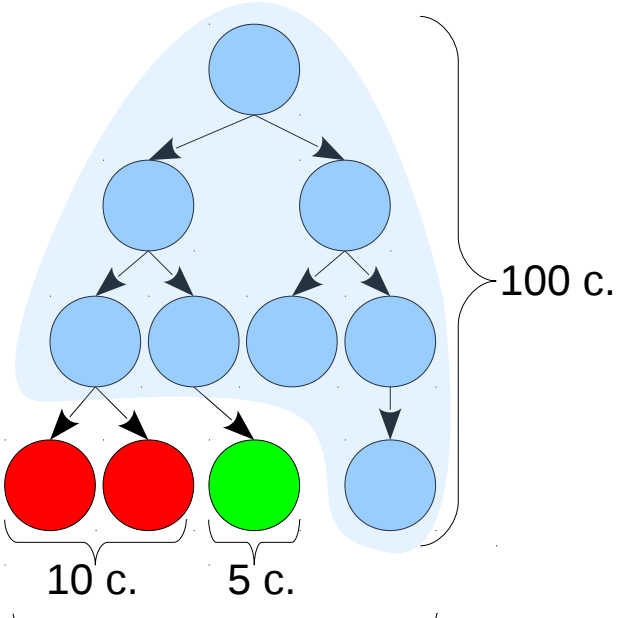
\+

= **215 с.**

= **9** путей выполнения

Одновременная

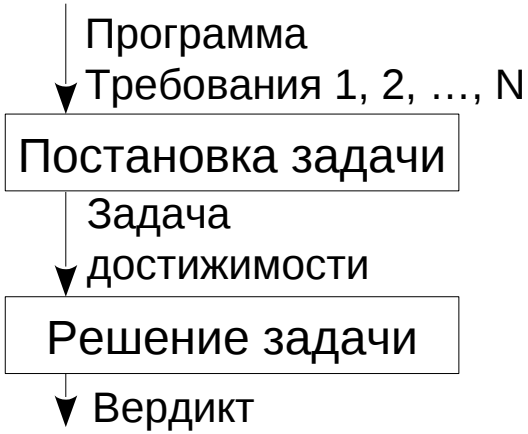
верификация требований **1** и **2**



115 с.

5 путей выполнения

Верификация композиции требований



- ✓ Переиспользование полученных знаний
- − Чрезмерный рост числа состояний
  - Потеря результата
- − Остановка проверки после нахождения нарушения требования
  - Пропуск других нарушений

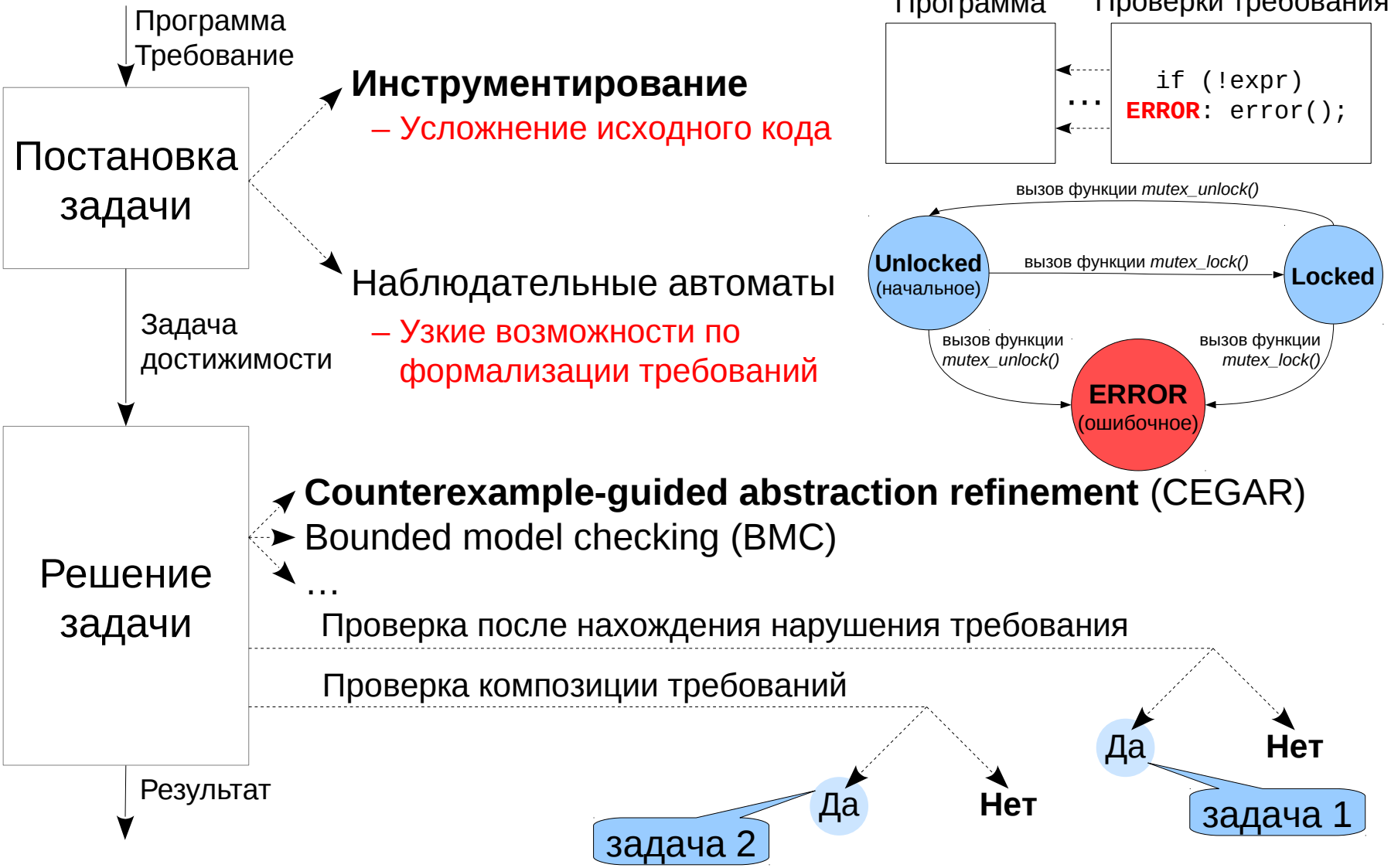
Цель работы

Разработка методов статической верификации программного обеспечения для проверки соответствия программ композиции требований

Задачи работы

- Провести анализ существующих методов статической верификации для определения того, насколько они подходят для достижения поставленной цели работы
- Разработать новые методы верификации программного обеспечения, предназначенные для проверки композиции требований с учетом того, что каждое требование может нарушаться более одного раза
- Реализовать предложенные методы
- Дать оценку области применимости предложенных методов и составить рекомендации по их использованию

Классические методы верификации

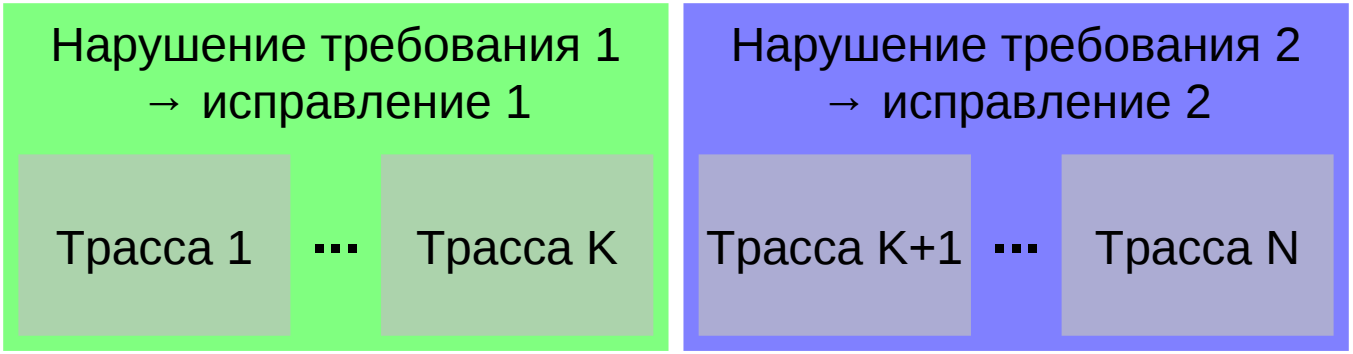


Предложенные методы

Способ постановки задачи достижимости	Проверка композиции требований	Проверка после нахождения нарушения	Подход решения задач	Метод
Инструментирование	Нет	Нет	Любой	Базовый
Инструментирование	Нет	**Да**	Любой	**ОВН**
Инструментирование	**Да**	Нет	CEGAR	**УМАВ**
Инструментирование	Да	Да	CEGAR	УМАВ с ОВН
Наблюдательные автоматы	Нет	Нет	Любой	**АС**
Наблюдательные автоматы	Нет	Да	Любой	АС с ОВН
Наблюдательные автоматы	**Да**	Нет	Любой	**ДАС**
Наблюдательные автоматы	Да	Да	Любой	ДАС с ОВН

Метод обнаружения всех однотипных нарушений (ОВН*)

- Продолжение верификации после нахождения нарушения требования
  - Увеличение времени верификации
- Анализ результата (трасс ошибок)
  - Автоматическая фильтрация
  - Ручная фильтрация



* Mordan V., Novikov E. *Minimizing the number of static verifier traces to reduce time for finding bugs in Linux kernel modules* // Proceedings of the SYRCoSE'2014.

Метод условной многоаспектной верификации (УМАВ*)

- Чрезмерный рост числа состояний
  - Ограничить ресурсы на проверку каждого требования
  - Переиспользовать знания о верификации
- Остановка проверки после нахождения нарушения требования
  - Продолжать верификацию без нарушенного требования
  - Получить результат для остальных требований

* Mordan V., Mutilin V. *Checking Several Requirements at once by CEGAR* // Proceedings of the PSI'2015.

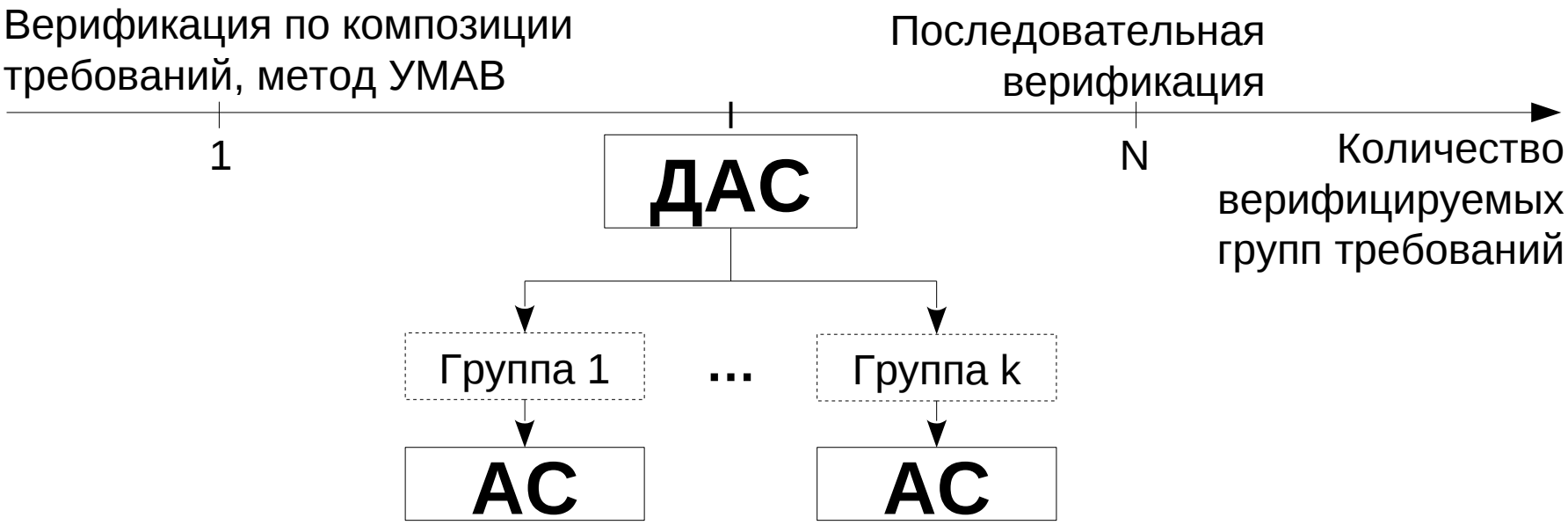
Метод автоматных спецификаций (АС*)

- Проблема
  - Инструментирование усложняет задачи
  - Нет возможности удалить проверки требования из кода во время верификации
- Решение (метод автоматных спецификаций)
  - Передача моделей требований независимо от исходного кода
  - Расширение возможностей наблюдательных автоматов
    - Добавление произвольных операторов C

* S. Apel, D. Beyer, V. Mordan, V. Mutilin, A. Stahlbauer. *On-The-Fly Decomposition of Specifications in Software Model Checking* // FSE'2016.

Метод декомпозиции автоматной спецификации (ДАС*)

- Цель
  - Найти разбиение на группы требований для совместной верификации



* S. Apel, D. Beyer, V. Mordan, V. Mutilin, A. Stahlbauer. *On-The-Fly Decomposition of Specifications in Software Model Checking* // FSE'2016.

Теоремы о полноте и корректности

Для требований, удовлетворяющих ограничениям на инструментирование исходного кода, сохраняются полнота и корректность предложенных методов относительно базового метода статической верификации

Ограничения на инструментирование

- **Запрещается** изменять исходные пути выполнения программы

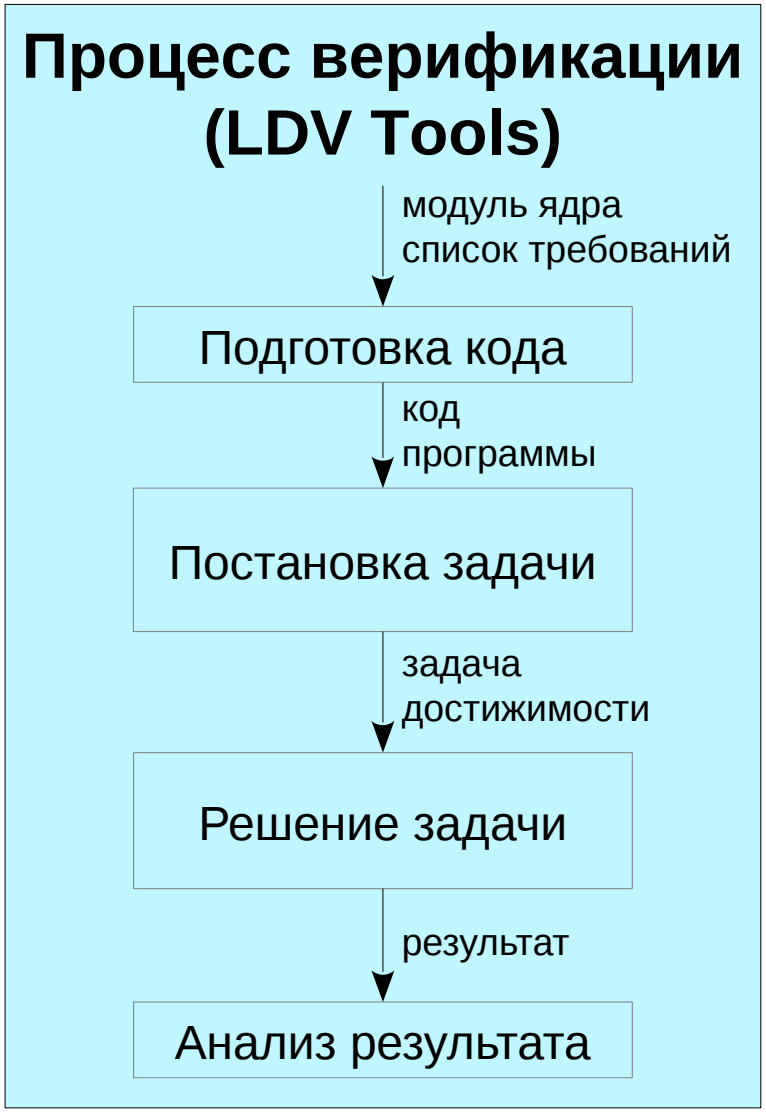
Реализация предложенных методов

Модификация инструментов

Система **Linux Driver Verification Tools**
- – Формализация требований (АС)
- – Подготовка задач достижимости с помощью автоматов (АС)

Статический верификатор **CPAchecker**
- – Расширение алгоритма CEGAR (УМАВ)
- – Расширение наблюдательных автоматов (АС)
- – Новый вид адаптивного анализа (ДАС)

Пользовательский web-интерфейс LDV
- – Фильтрация трасс ошибок (ОВН)



Оценка базовых методов

- 4 041 модуль ядра ОС Linux
  - 85% модулей ядра linux-4.0-rc1
  - 9 млн. строк кода
- 30 требований системы LDV Tools
- 121 230 задач достижимости

Метод	Safe	Unsafe	Потери / новые (%)	Время (с) / ускорение	
				Решение задач	Всего
Последовательная верификация	118 703	667	–0.00 +0.00	3 889 000 1.00	6 742 000 1.00
Верификация композиции требований	98 580	527	**–16.80** +0.09	3 527 000 **1.10**	3 780 000 1.78

➡ **17%** потерь, ускорение 10%

Оценка предложенных методов

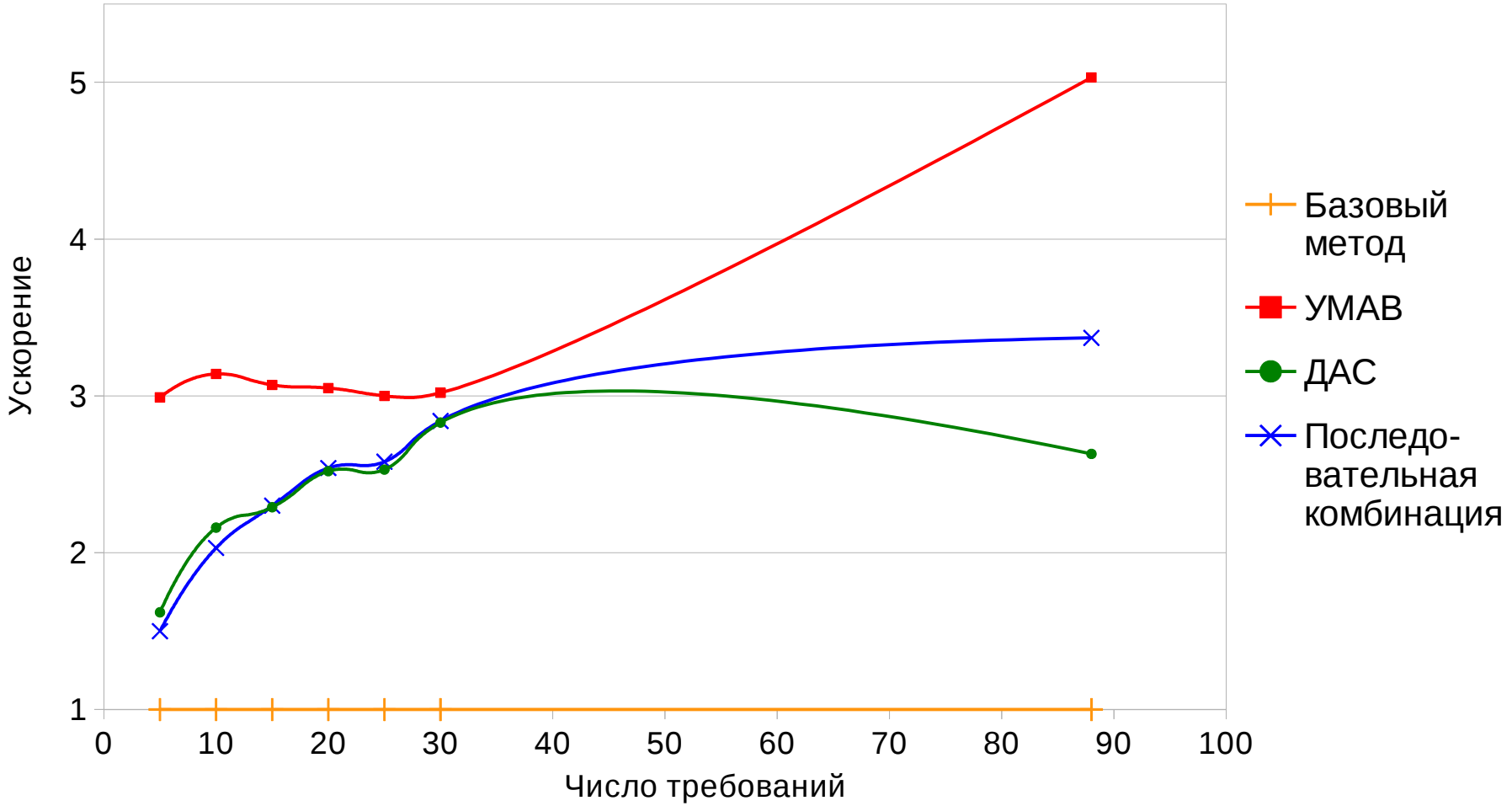
Метод	Safe	Unsafe	Потери / новые (%)	Реальные ошибки		Время (с) / ускорение	
						Решение задач	Всего
Базовый метод	118 703	667	–0.00 +0.00	121	–0 +0	3 889 000 1.00	6 742 000 1.00
Метод УМАВ	117 162	634	**–1.36** +0.06	114	–9 +2	1 289 000 **3.02**	1 514 000 **4.45**
Метод УМАВ (с разбиением*)	117 628	660	**–0.93** +0.04	121	–2 +2	1 041 000 **3.74**	1 382 000 **4.88**
Метод ДАС	118 386	673	**–0.45** +0.20	120	–2 +1	1 373 000 **2.83**	1 550 000 **4.35**
Последовательная комбинация**	118 679	695	**–0.26** +0.26	125	–0 +4	1 367 000 **2.84**	1 592 000 **4.23**

➡ Ускорение в **4-5 раз** при потерях порядка **1%**

* Для упрощения задач достижимости 30 требований были разбиты на 2 группы требований.

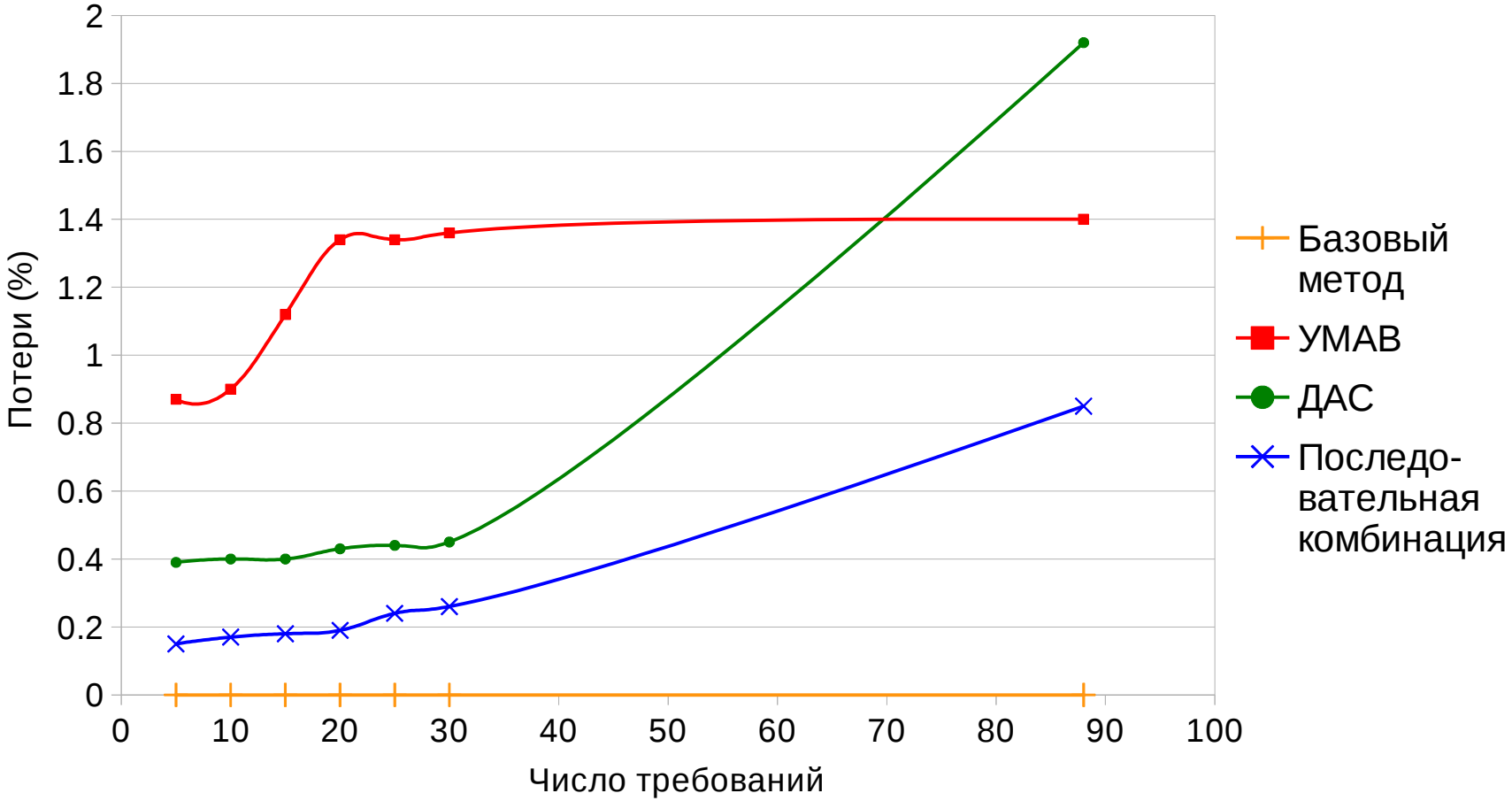
** Сначала задача решается методом УМАВ (более быстрый метод), если ее решить не удается, то применяется метод ДАС (более ресурсоемкий, но более точный).

Зависимость ускорения от числа требований*



* 30 требований были разбиты по различным типам нарушений (например, утечка и повторное освобождение ресурса), что позволило увеличить их число до 88. На практике проверять каждый тип нарушения требования избыточно, поскольку они взаимосвязаны.

Зависимость потерь от числа требований



Научная новизна

- Метод статической верификации программного обеспечения для обнаружения всех однотипных нарушений проверяемого требования
- Метод статической верификации программного обеспечения для проверки выполнения композиции требований (условная многоаспектная верификация)
- Метод статической верификации программного обеспечения, расширяющий возможности представления требований в виде их автоматных спецификаций
- Метод статической верификации программного обеспечения на основе декомпозиции автоматной спецификации требований на группы требований для совместной верификации внутри группы
- Сформулированы и доказаны утверждения и теоремы, являющиеся обоснованием корректности предложенных методов

Результаты, выносимые на защиту

- Методы статической верификации программного обеспечения, основанные на инструментировании исходного кода и предназначенные для обнаружения всех однотипных нарушений (ОВН) и проверки выполнения композиции требований с помощью условной многоаспектной верификации (УМАВ)
- Методы статической верификации программного обеспечения, с использованием формализации требований в виде автоматных спецификаций (АС) и декомпозиции автоматной спецификации на группы требований для совместной верификации (ДАС)
- Теорема о полноте и корректности предложенных методов для требований, удовлетворяющих ограничениям инструментирования исходного кода программы

Публикации по теме диссертации

1. Мордань В. О. Многоаспектная верификация модулей ядра операционной системы Linux // Материалы XXI Международной молодежной научной конференции студентов, аспирантов и молодых ученых "Ломоносов", с. 122-123, 2014.
2. Mordan V., Novikov E. Minimizing the number of static verifier traces to reduce time for finding bugs in Linux kernel modules // Proceedings of 8th Spring/Summer Young Researchers Colloquium on Software Engineering, vol. 1, 2014.
3. Mordan V., Mutilin V. Checking several requirements at once with CEGAR // Perspectives of Systems Informatics. LNCS, vol. 9609, pp. 218-232, 2016.
4. Мордань В. О., Мутилин В. С. Проверка нескольких требований за один запуск инструмента статической верификации с помощью CEGAR // Программирование, т. 4. с. 225-238, 2016.
5. Apel S., Beyer D., Mordan V., Mutilin V., Stahlbauer A. On-The-Fly Decomposition of Specifications in Software Model Checking // Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 349-361, 2016.

Свидетельства о государственной регистрации программы для ЭВМ

1. Мордань В.О. «Программный компонент для выявления нескольких ошибок в программном обеспечении». Свидетельство о государственной регистрации программы для ЭВМ № 2016616600 от 15.06.2016.
2. Мордань В.О. «Программный компонент для проверки нескольких правил корректности за один запуск инструмента статической верификации». Свидетельство о государственной регистрации программы для ЭВМ № 2016616661 от 16.06.2016.

Федеральное государственное бюджетное учреждение науки
Институт системного программирования Российской академии наук

ИСП РАН

Спасибо

Мордань Виталий Олегович
mordan@ispras.ru