Повышение производительности итеративного динамического анализа за счёт применения статической инструментации исполняемого кода

Михаил Ермаков
МГУ им. М.В.Ломоносова, факультет ВМК

Динамический анализ программ

- Исследование аспектов времени выполнения
- Результаты ограничены путями выполнения

- Решение: автоматический обход возможных путей выполнения

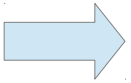
- Проверка наличия дефектов и уязвимостей
- Построение тестового покрытия для дальнейшего использования

Автоматический обход путей выполнения

- Отождествление инструкций пути булевым формулам
  - Операнды → символьные переменные
  - Внешние данные → свободные переменные
  - Эффект инструкций → ограничение на зависимости по данным
- Поиск новых путей с помощью модификации трасс

Модификация трасс (1)

```
int x = read(...)
int y = x;
if (y < 0)
   do1();
else
   do2();
```

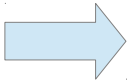


```
x : bitvector(32);
y : bitvector(32);
taint (x);
assert (y = x);
assert (y < 0 = true);
... // do1()
```

Модификация трасс (2)

```
x : bitvector(32);
y : bitvector(32);
taint (x);
assert (y = x);


assert (y < 0 = false);
```



```
int x = read(...)
int y = x;
if (y < 0)
   do1();
else
   do2();
```

Обход путей: подходы

- Online (KLEE, S2E, BitBlaze)
  - Выше скорость обхода; больше затраты ресурсов
- Offline (SAGE, DART, Avalanche)
  - Скорость обхода меньше; затраты ресурсов меньше
  - Гибкость системы; «классическое» тестирование
- Гибридный подход (Mayhem)

Обход в режиме offline: извлечение трассы

- Трансформация исходного кода (DART, CUTE)

- Динамическая инструментация (SAGE, Fuzzgrind, Catchconv, Avalanche)

Динамическая инструментация

- Модификация кода во время исполнения
- Полнота
  - Поддержка динамического кода
  - Поток данных во всём выполняемом коде
- Накладные расходы на каждом запуске

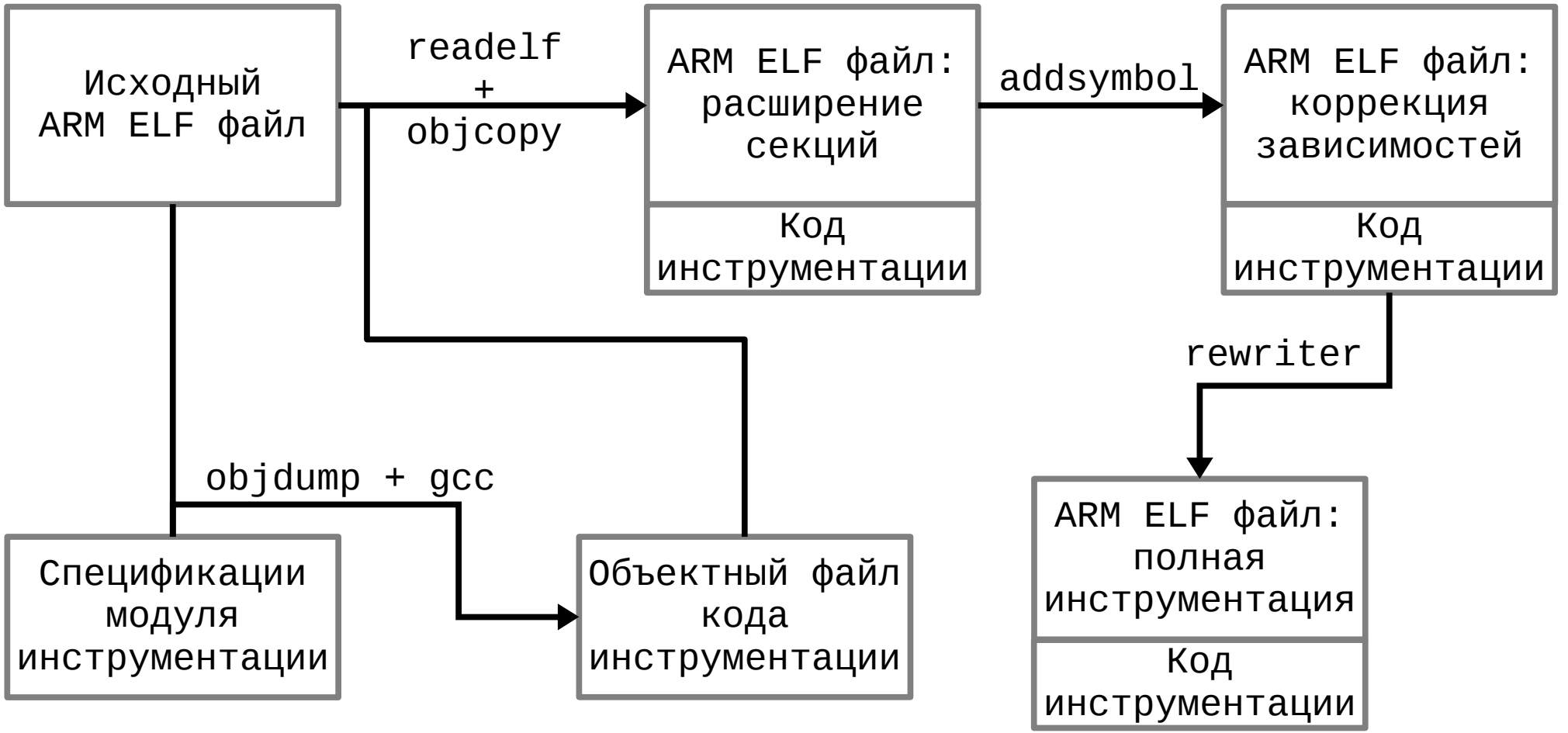
Обход в режиме offline: извлечение трассы

- Трансформация исходного кода (DART, CUTE)
- Динамическая инструментация (SAGE, Fuzzgrind, Catchconv, Avalanche)
- **Статическая инструментация исполняемого кода?**

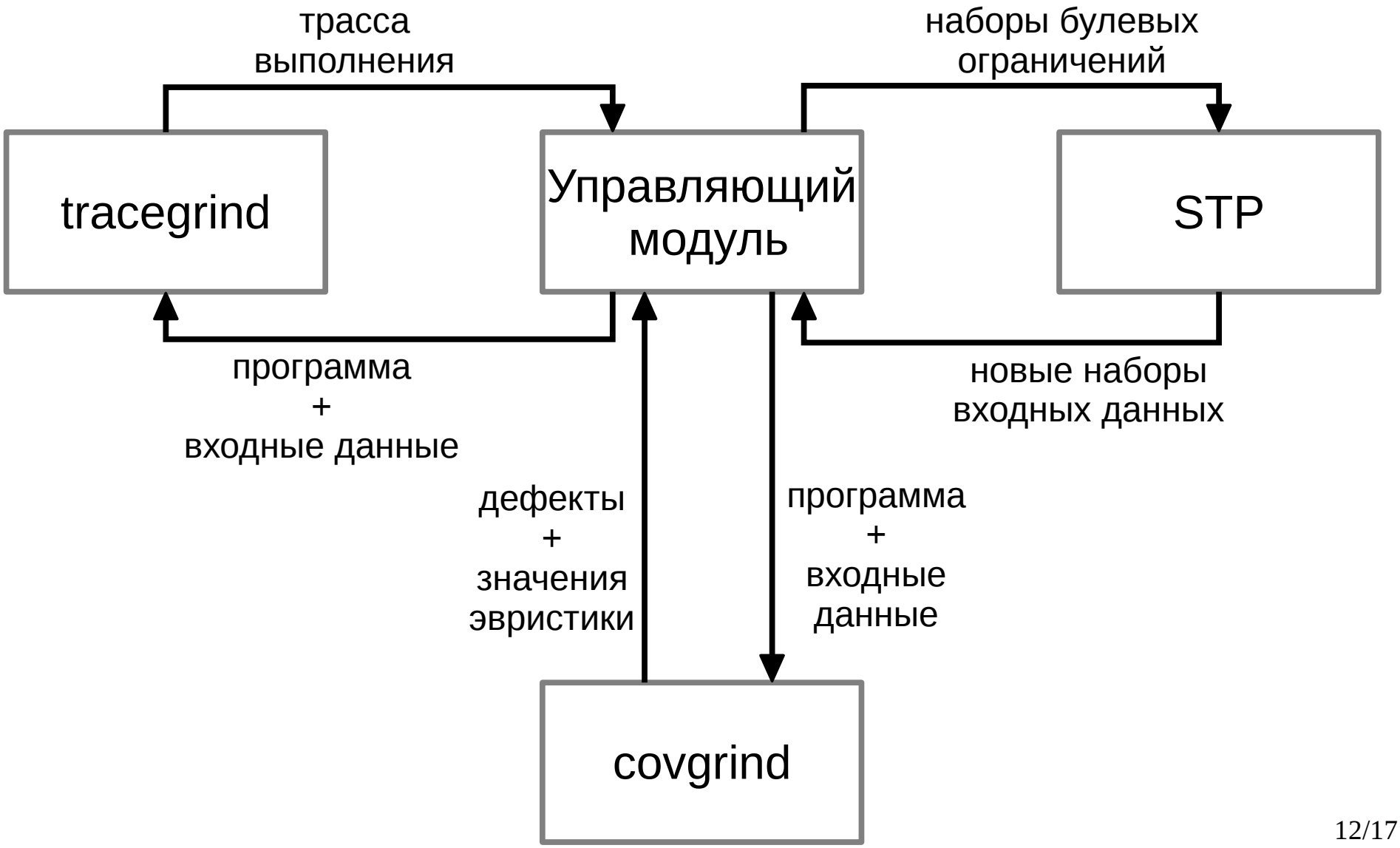
Статическая инструментация

- Модификация кода до выполнения
- Обработка ограниченного объёма кода
  - Точность разбора; нет поддержки динамического кода
  - Поток данных проходит сквозь «слепые» места
- Накладные расходы единократно
- Прямое сопряжение с анализом бинарного кода

Статическая инструментация: схема работы



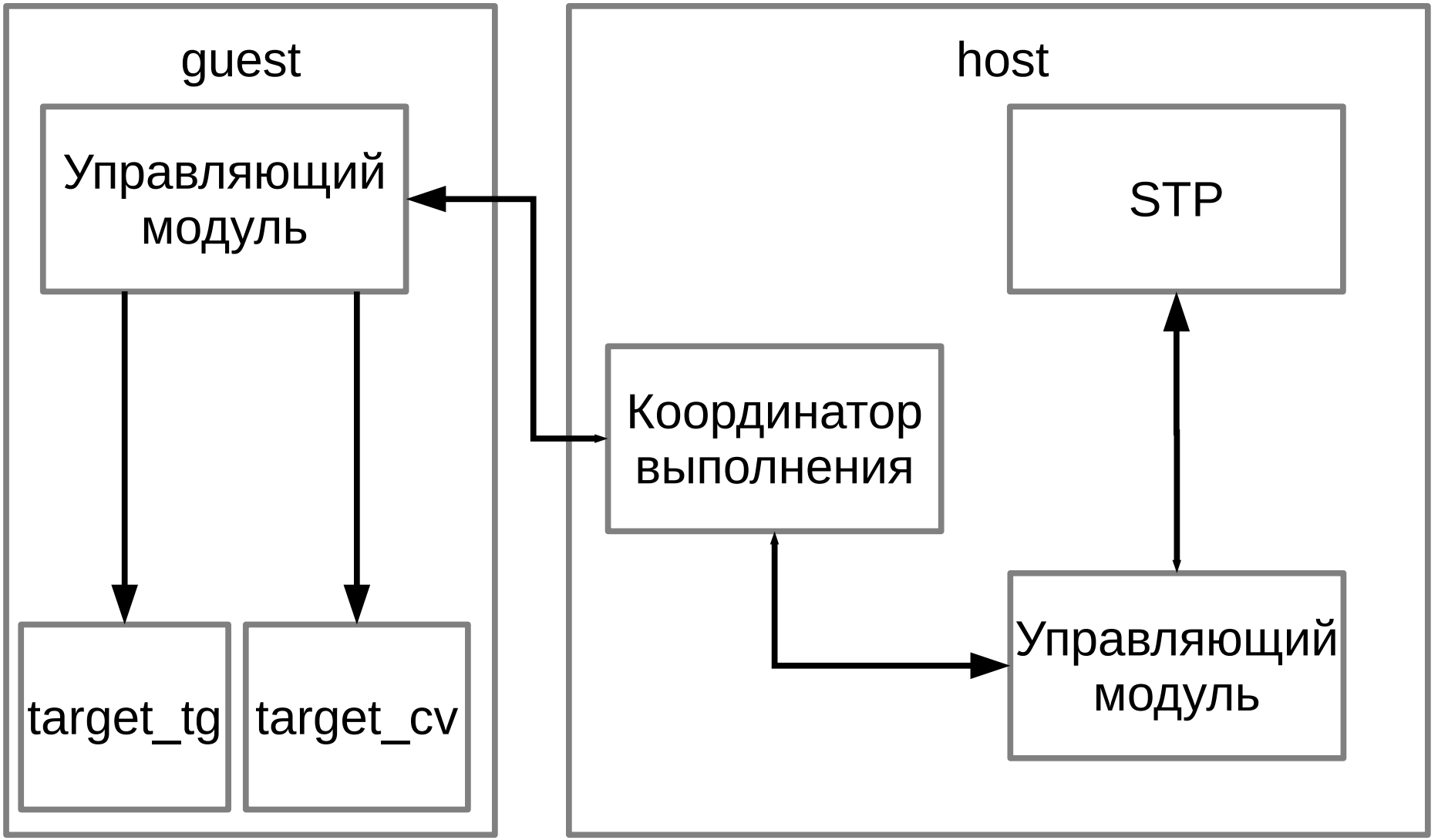
Итеративный динамический анализ: Avalanche



Статическая инструментация: Avalanche

- Tracegrind (Valgrind) → tracegrind (static)
  - Сокращение накладных расходов
  - Нет промежуточных переменных
- Covgrind (Valgrind) → covgrind (static)
  - Сокращение накладных расходов
  - Восстановление исходного бинарного кода во время выполнения

Avalanche: схема работы для анализа ARM



Avalanche: результаты анализа

	итерации		дефекты		(Tr + Cg)/STP	
	динамика	статика	динамика	статика	динамика	статика
swfdump	105	575	1	4	87/13	72/28
mpeg2dec	55	302	1	1	89/11	37/63
cjpeg	236	1331	1	1	88/12	10/90
qtdump	189	1027	1	1	80/20	21/79
mpeg3dump	54	125	1	2	76/24	41/59

Направления дальнейшей работы

- Применение статического анализа бинарного кода
  - Уменьшение числа точек инструментации
  - Оптимизация построения фрагментов трассы
- Оптимизация построения и внедрения инструментационного кода

Спасибо за внимание!