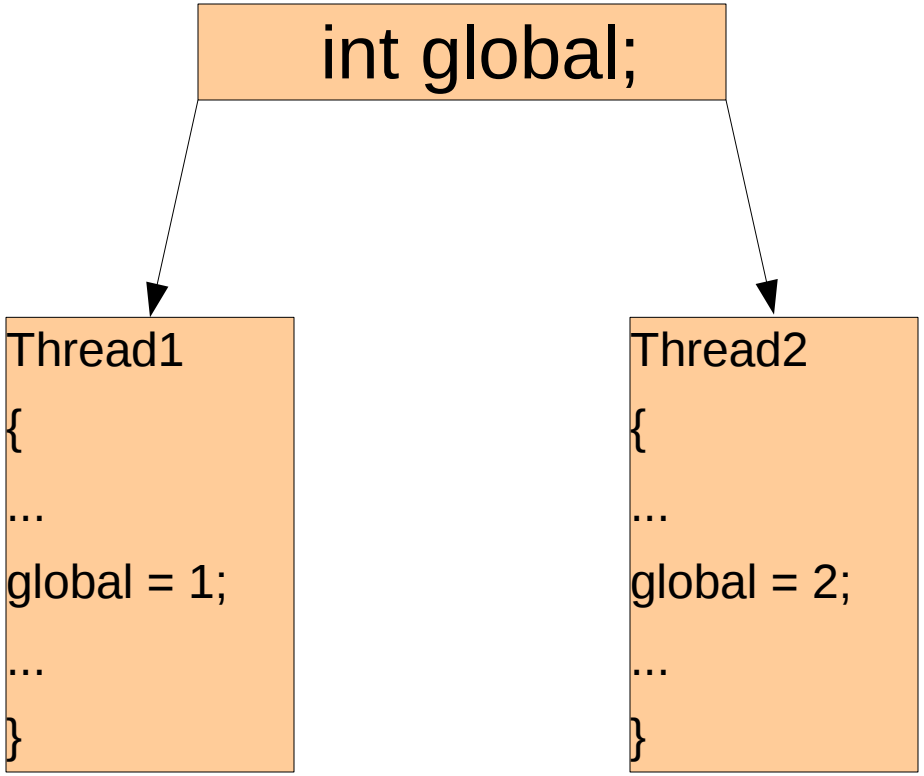
Поиск состояний гонок методами статического анализа

Андрианов Павел
andrianov@ispras.ru

Institute for System Programming of the Russian Academy of Sciences

Состояние гонки



Существующие инструменты

- Динамические (Eraser, Intel Thread Checker, Sun Thread Analyzer)
- Статические (Coverity Prevent, KlocWork K7, Locksmith, Prefast)
- Проверка на основе моделей (SLAM)

Особенности задачи

- Специфика ядра операционной системы
- Масштаб анализируемого кода
- Критичность ошибки

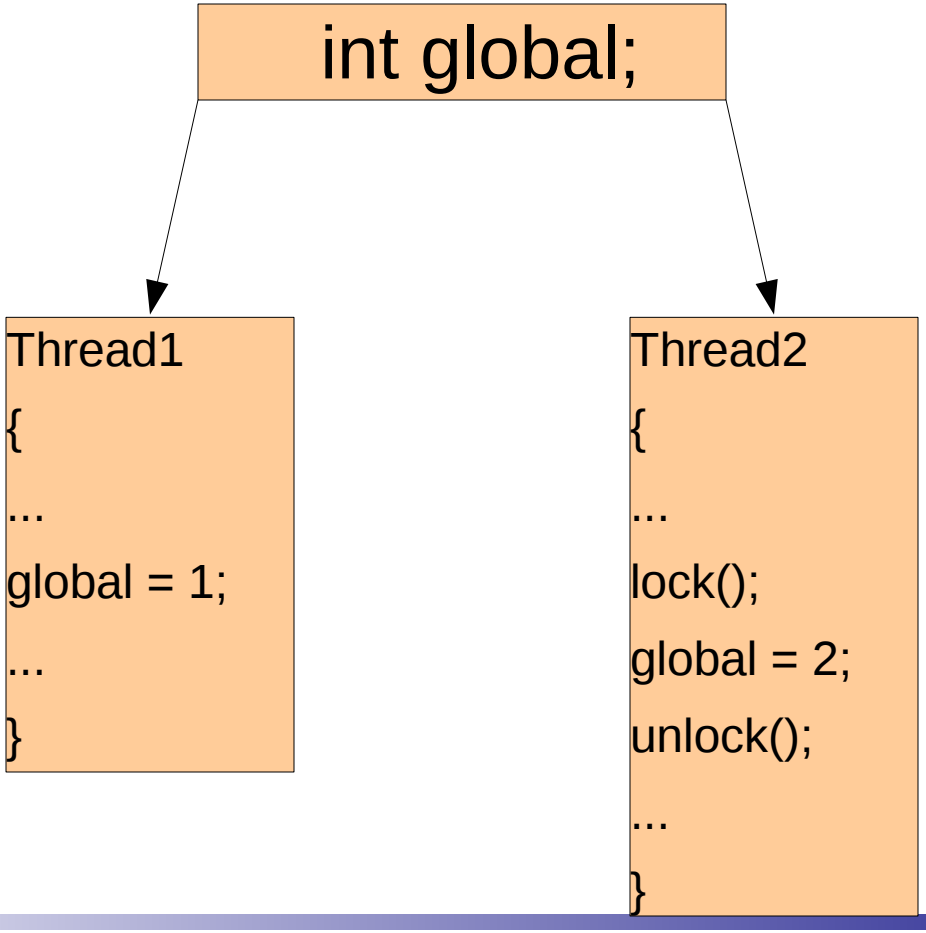
Некоторые понятия

- Переменная – локальные, глобальные и поля структур
- Механизм синхронизации (блокировка)
- Использование переменной – доступ к данным на чтение или запись

Состояние гонки

- *Потенциальным состоянием гонки* будем считать ситуацию, в которой доступ к одной и той же переменной происходит **с разным** непересекающимся набором блокировок, при этом одно из обращений является записью.

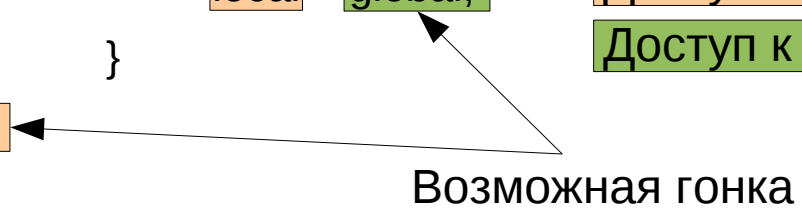
Состояние гонки



Пример

```
int func1() {
        struct A *s;
        int local;
        s->f = ...
        lock();
        global = ...
        local = ...
        unlock();
}
```

```
int func2() {
        int local;
        ...
        local = global;
}
```



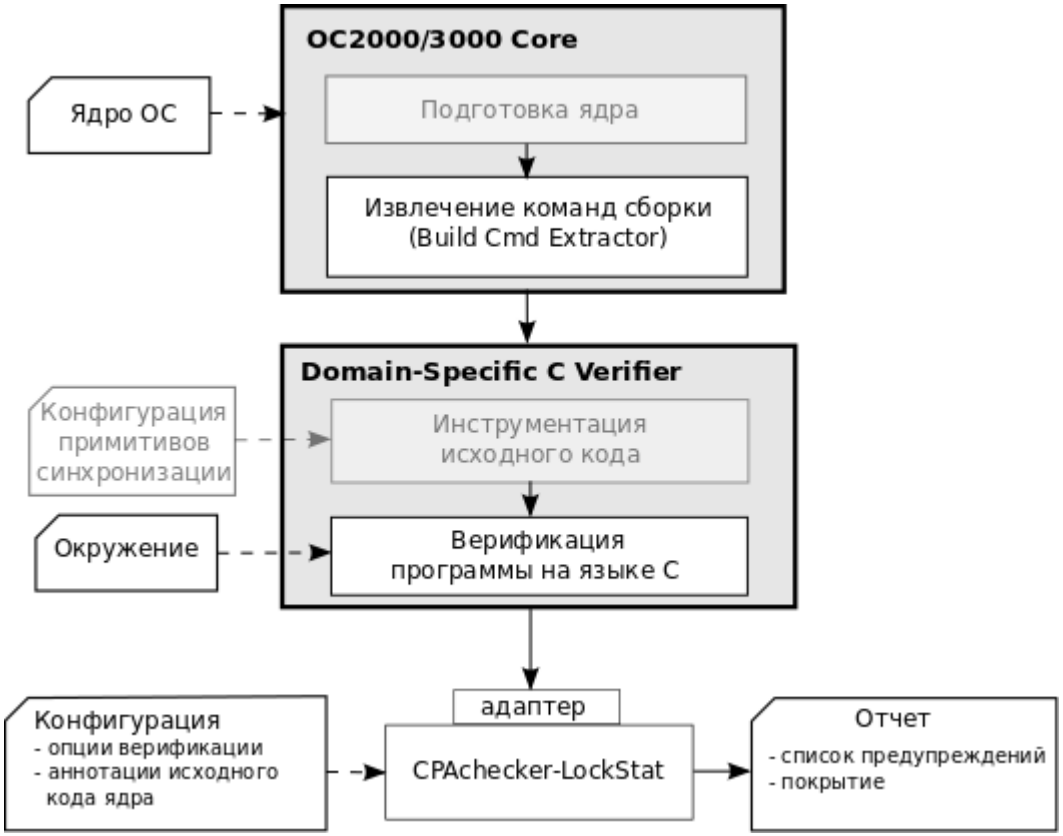
Захват блокировки

Освобождение блокировки

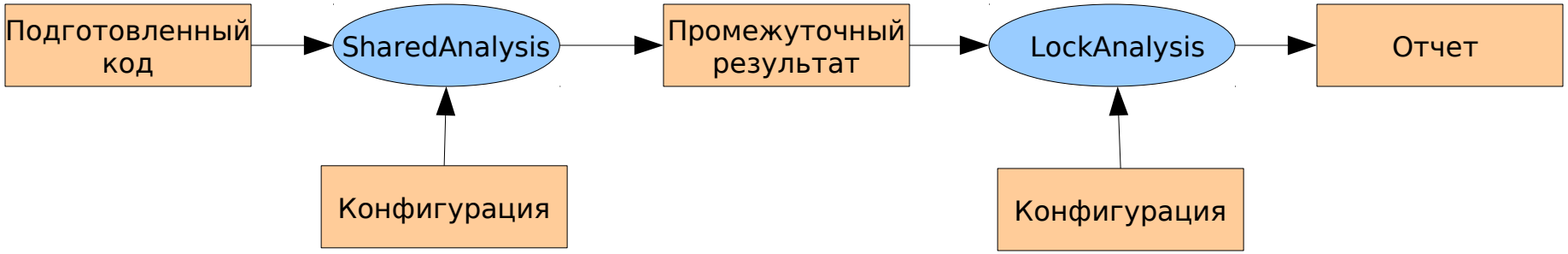
Доступ к переменной на запись

Доступ к переменной на чтение

Архитектура решения



Структура инструмента



Общая идея

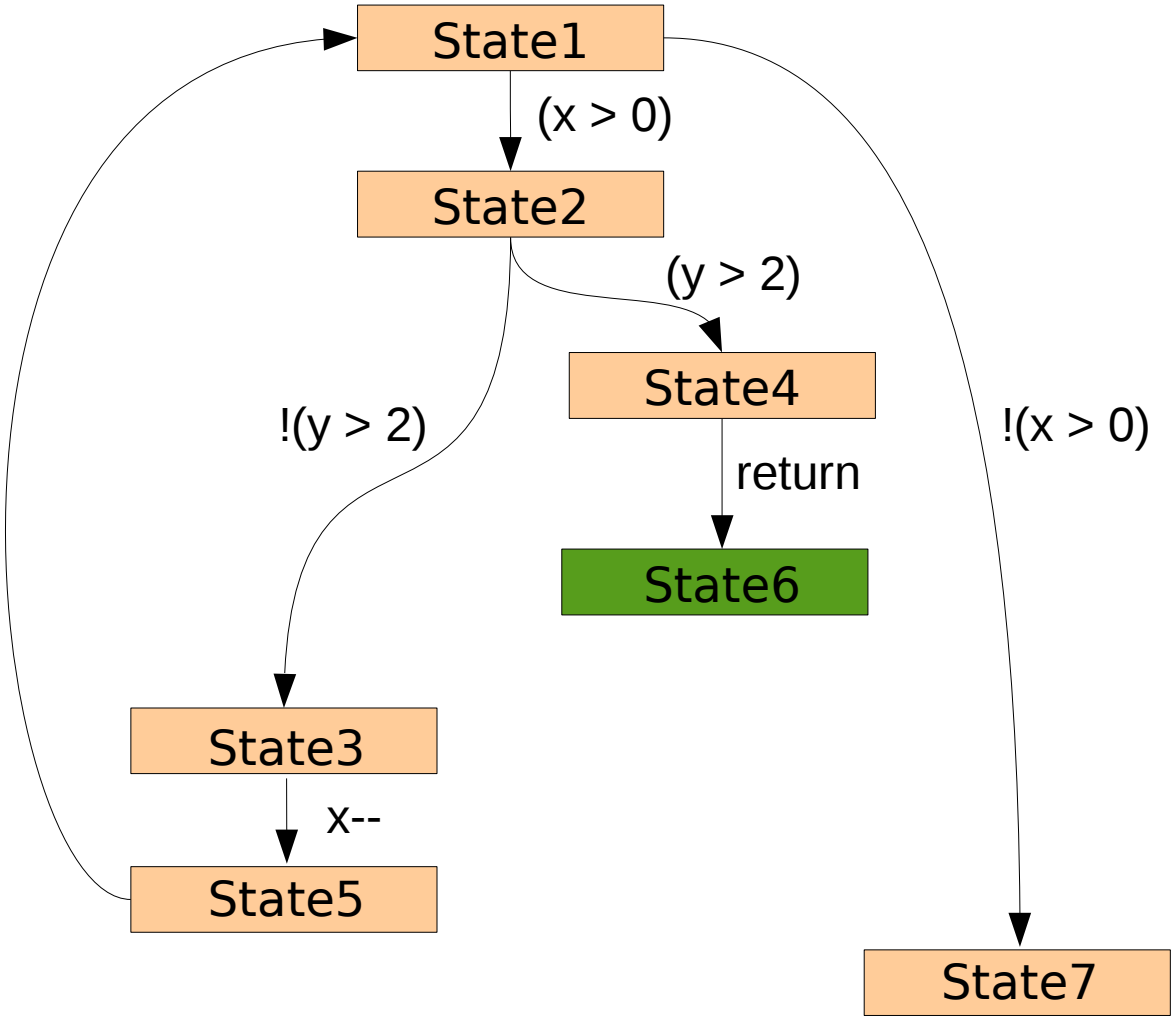
```
while (x > 0) {

    if (y > 2) {

        return;

    }

    x--;
}
```



Адаптивный статический анализ (Configurable Program Analysis)

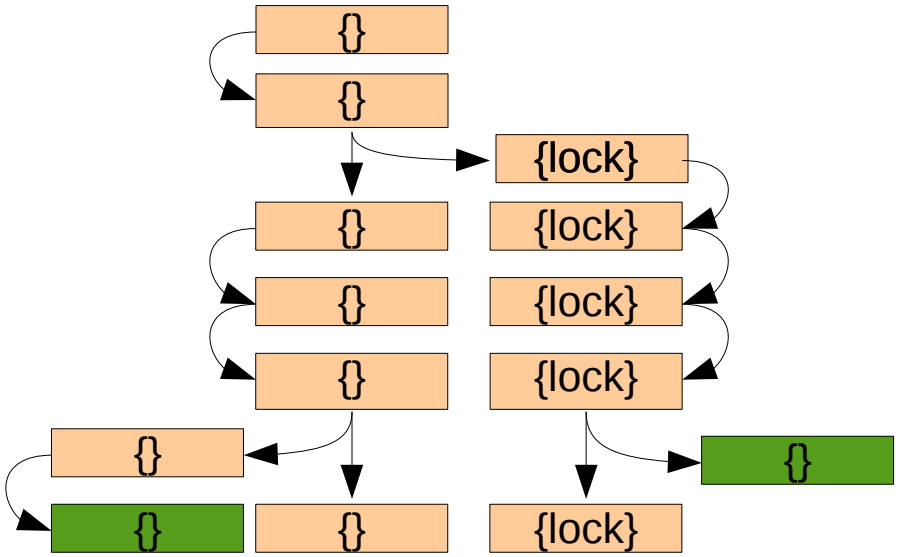
- Абстрактный домен – задает множество абстрактных состояний
- Отношение переходов – для каждого абстрактного состояния ***e*** определяет следующее состояние ***e′***.
- Оператор слияния – объединяет информацию от двух абстрактных состояний
- Оператор останова – проверяет, покрывается ли состояние множеством других

Пример: LockStatisticsCPA

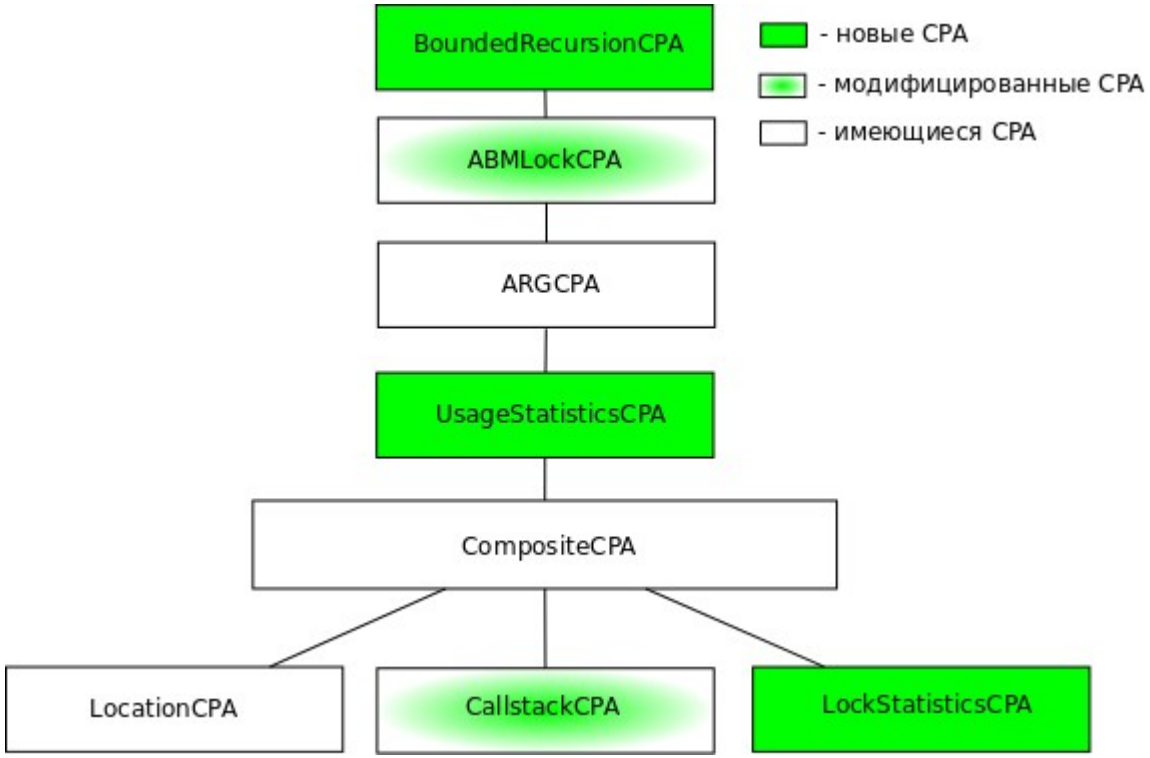
- Абстрактное состояние – множество захваченных блокировок
- Отношение переходов – изменяет состояние, если вызывается функция захвата или освобождения блокировки
- Оператор слияния – SEP, т.е. состояния никогда не объединяются
- Оператор останова – SEP, т.е. состояние покрыто тогда, когда в переданном множестве имеется равное ему состояние

Пример: LockStatisticsCPA

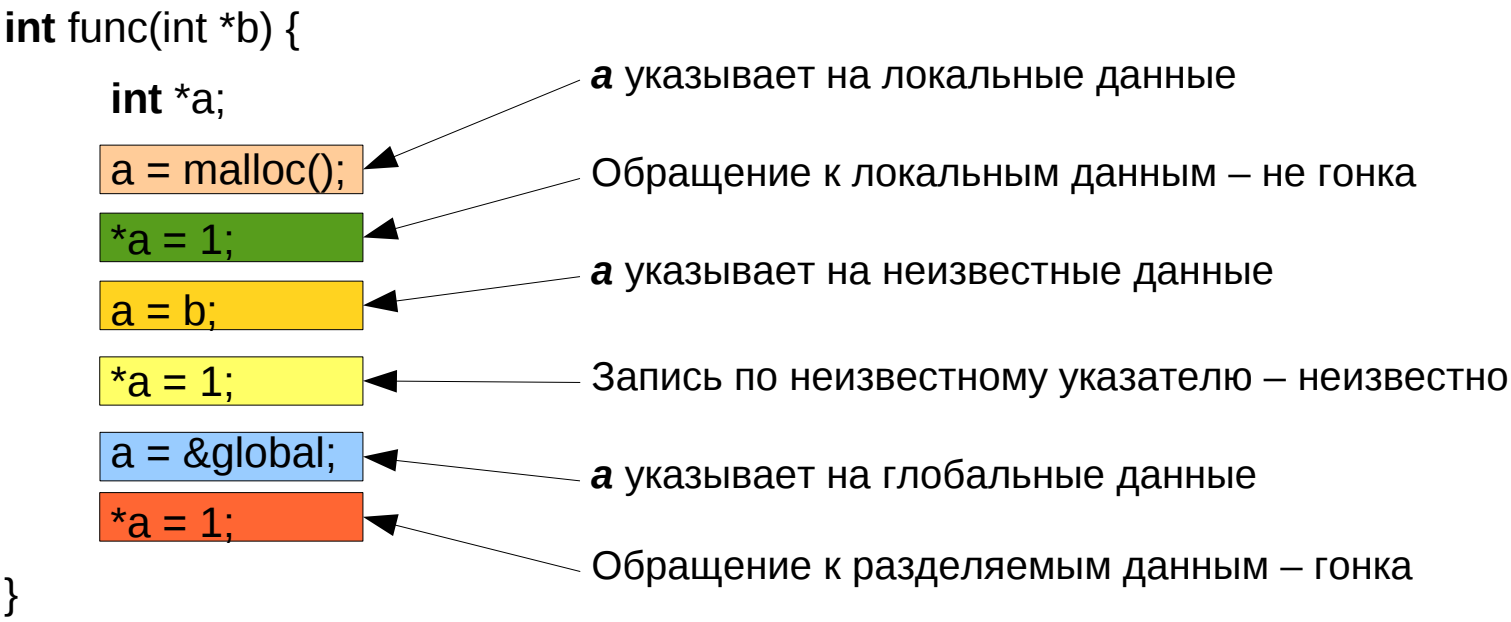
```
int global1, global2;
int func() {
        int a = 0;
        if (global1) {
            lock();
        }
        global2++;
        if (global1) {
            unlock();
        }
}
```



Конфигурация алгоритмов CPA в LockAnalysis



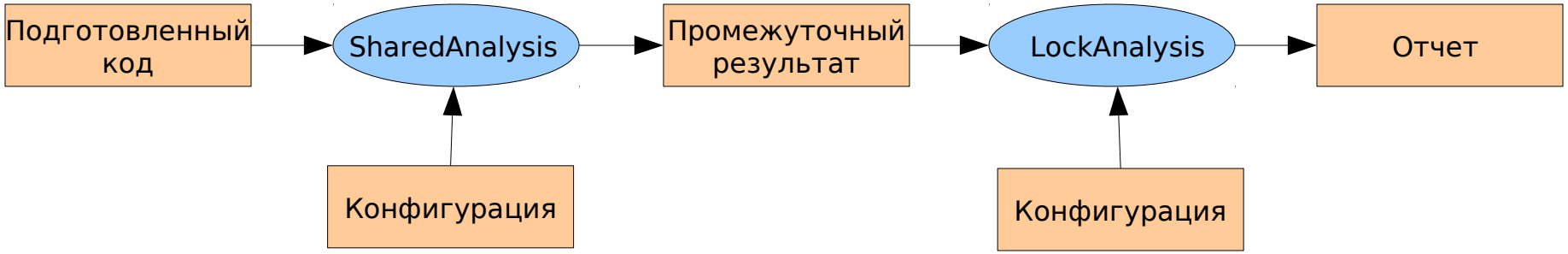
Локальность данных



Структура SharedCPA

- Абстрактное состояние – содержит информацию о локальности переменных, доступных в данной точке программы
- Отношение переходов – консервативно изменяет состояние
- Оператор слияния – JOIN, т.е. информация из двух состояний объединяется с учетом иерархии *global* > *null* > *local*
- Оператор останова - SEP

Структура инструмента



Отчет

Statistics	General	Unsafe
Global variables:	**591**	**24**
Simple:	411	20
Pointer:	180	4
Local variables:	**1296**	**50**
Simple:	0	0
Pointer:	1296	50
Structure fields:	**1324**	**300**
Simple:	1038	287
Pointer:	286	13
Total variables:	3211	374

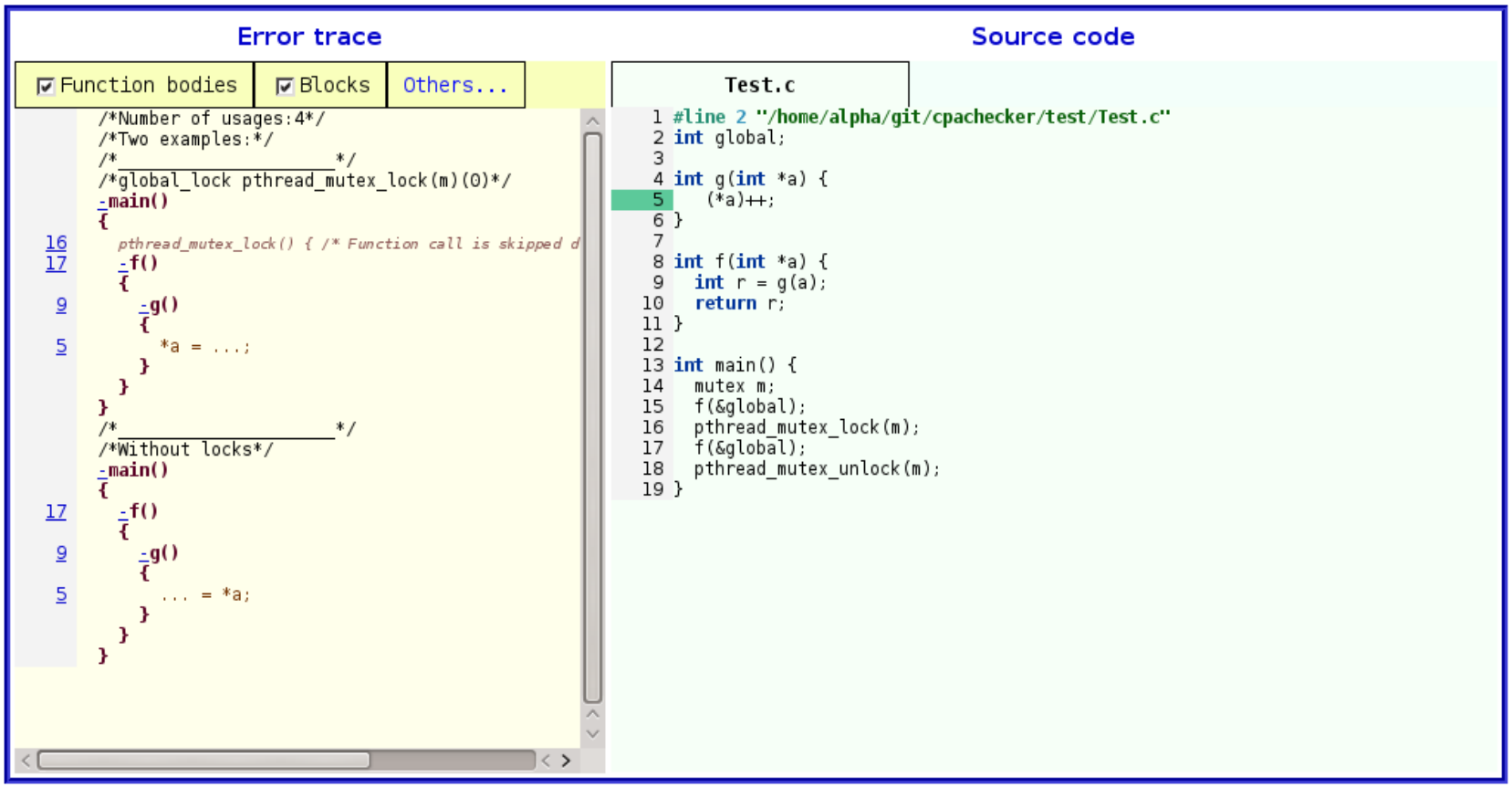
Finded locks:

1. global_lock lock()
2. global_lock pthread_mutex_lock(m->_mutex)(0)
3. global_lock pthread_mutex_lock(mutex)(0)

List of unsafes:

Визуализация ошибочной трассы

int *a



Полученные результаты

- Покрытие 26% кода по функциям и 20% по строкам кода
- Ошибки, найденные при компиляции ядра и разборе исходного кода
- Более 10 ошибок, связанных с состоянием гонки

Спасибо за внимание

Вопросы?