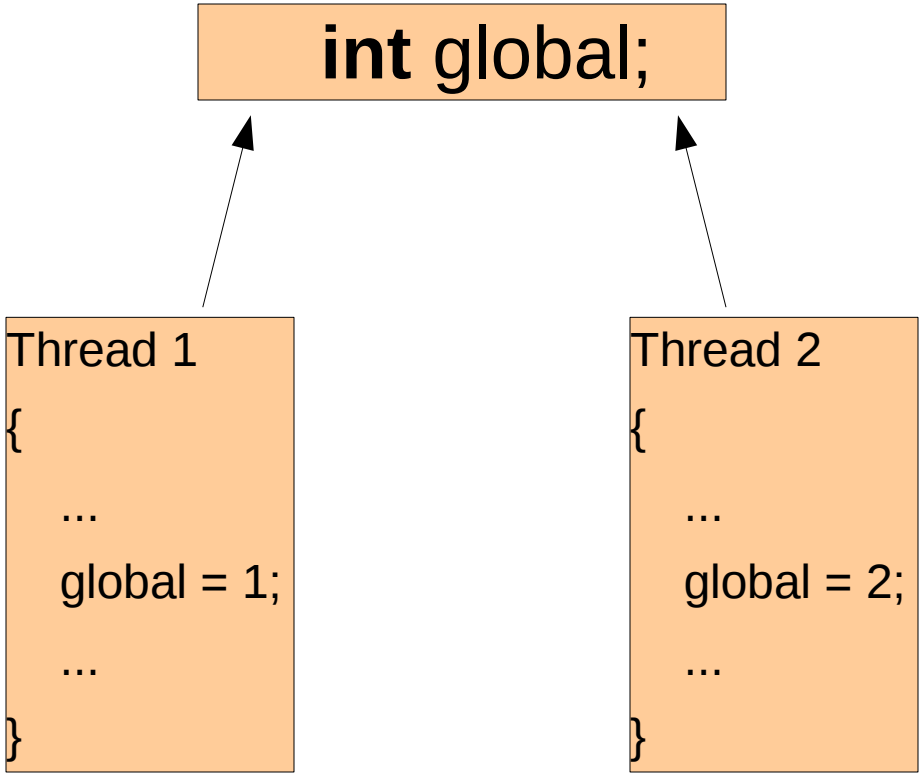
Static analysis for data race detection

Pavel Andrianov
andrianov@ispras.ru
http://linuxtesting.org/project/ldv/

Institute for System Programming of the Russian Academy of Sciences

Data Race Condition



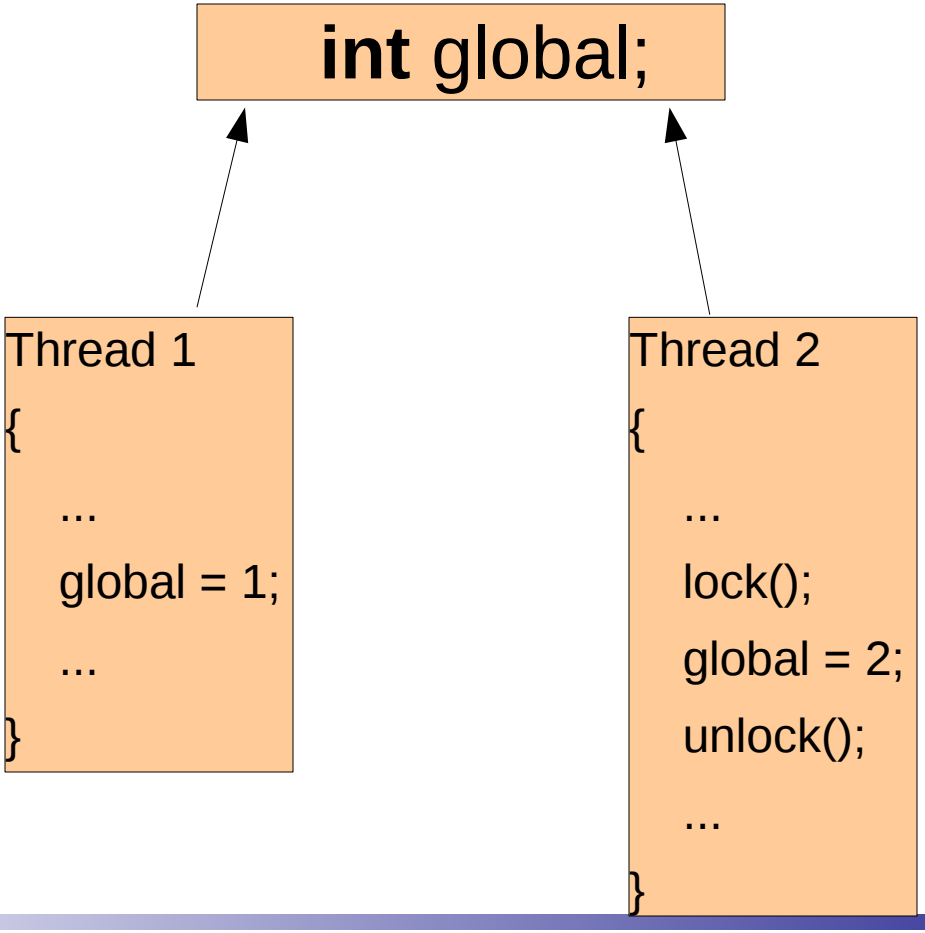
Notions

- ***Variable*** – local, global, structure field
- ***Lock*** – synchronization primitive
- ***Usage of variable*** – write or read data access
- ***Shared data*** – available from several threads data
- ***Unsafe*** – potential error (race)

Data Race Condition

- *Potential data race condition* is a situation, when usage of the same variable occurs with **different** disjoint sets of locks, one usage is write access.

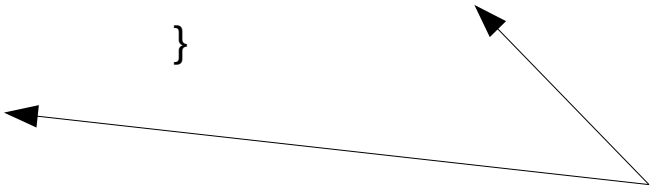
Data Race Condition



Example

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

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



Potential race condition

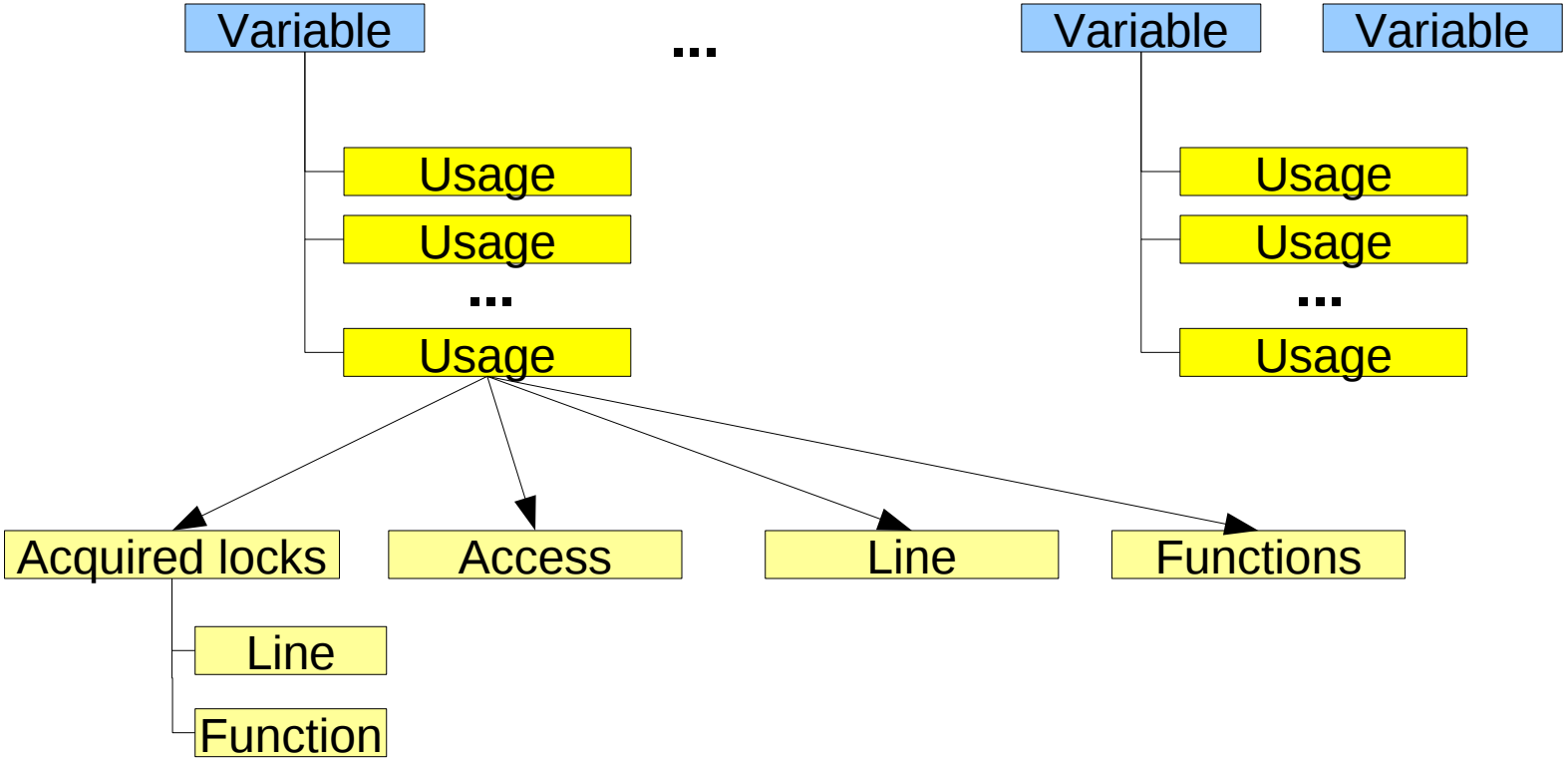
Acquiring a lock

Releasing a lock

Write access to a variable

Read access to a variable

Necessary information



Analysis of shared data

```
struct my_struct {
        int* b;
} A;
int func() {
        int *a;
        a = malloc();
        *a = 1;
        A->b = a;
        *a = 1;
        a = g();
}
```

a -> local data

Write access to local data – not a race

a -> shared data

Write access to shared data – subject for data race

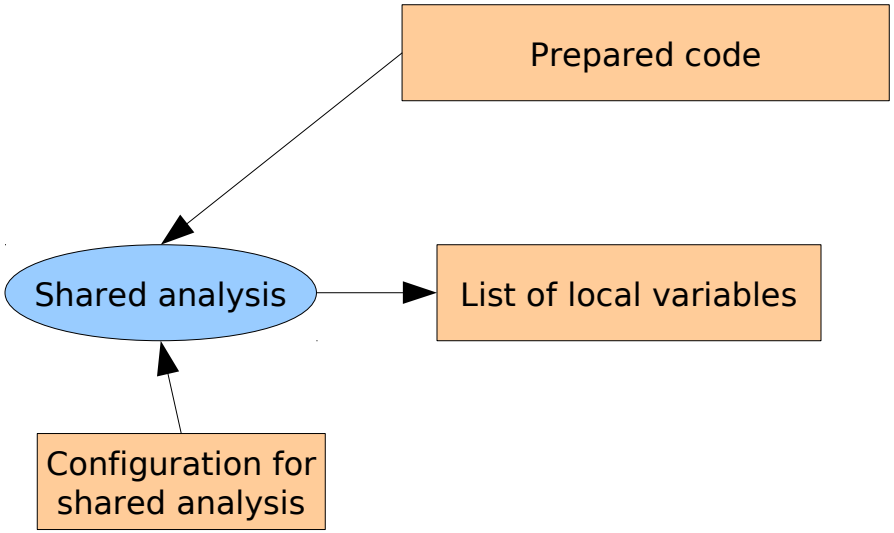
a -> unknown data

Stage of analysis:
Instrumentation

- Replacement macroexpantions by calls of model functions
- Insert entry point (main function)

Stage of analysis:
Analysis of shared data

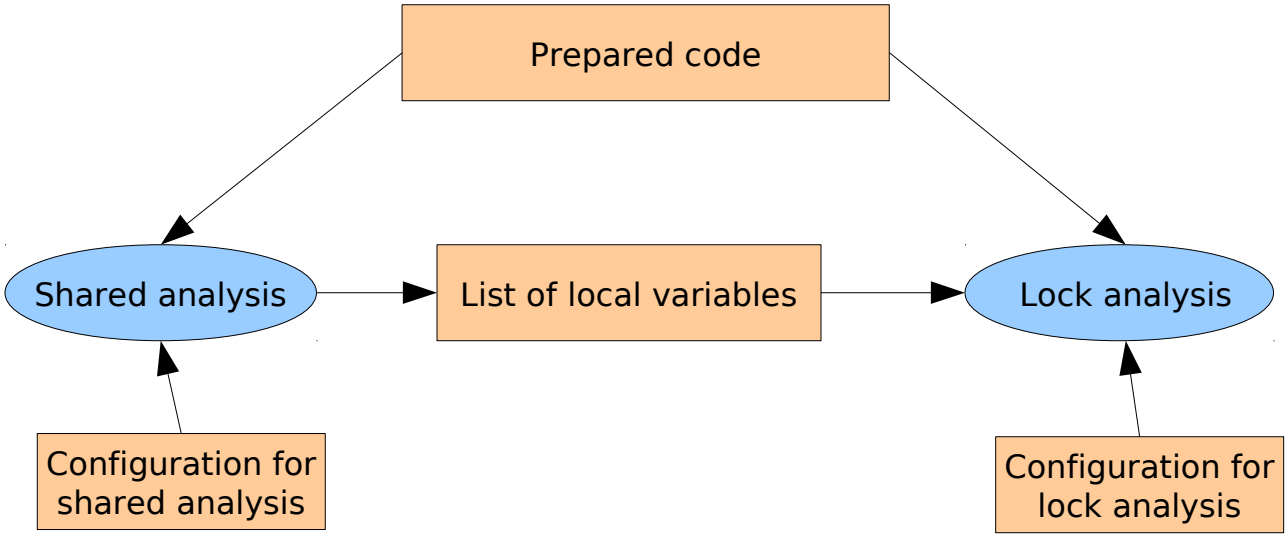
- Obtain information about local variables for every point in program



Stage of analysis:

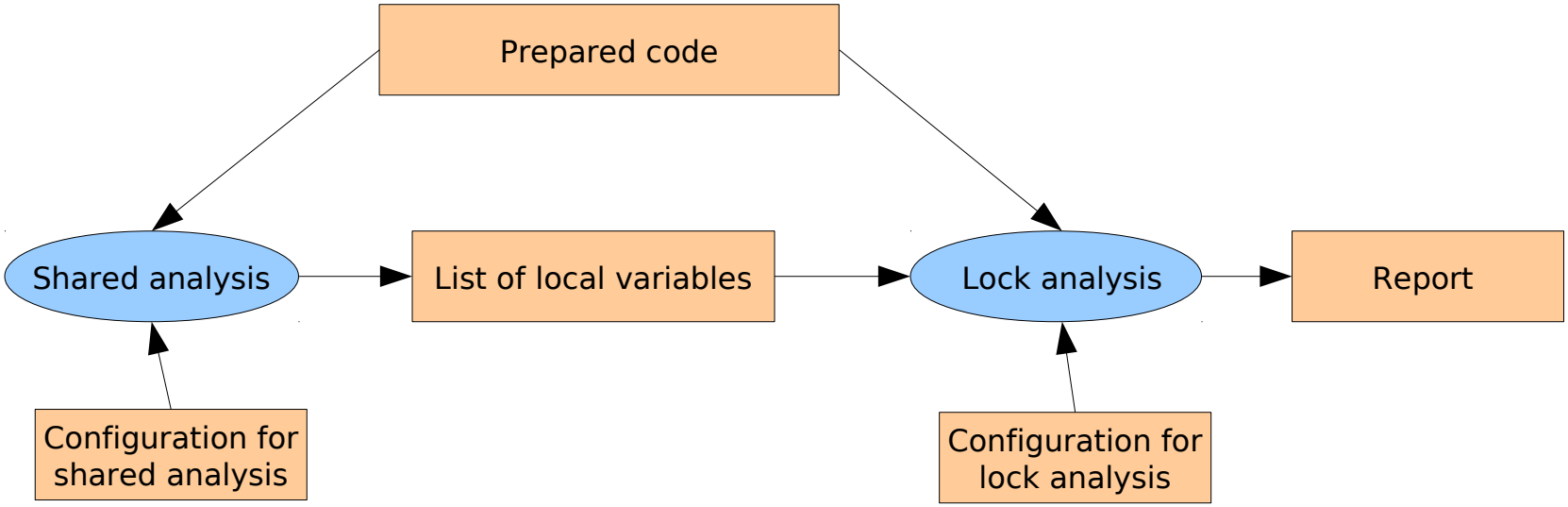
Analysis of synchronization primitives

Save information about acquired locks for every access to memory



Stage of analysis:
Result

- List of unsafe cases
- Detail information about every unsafe



Report

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

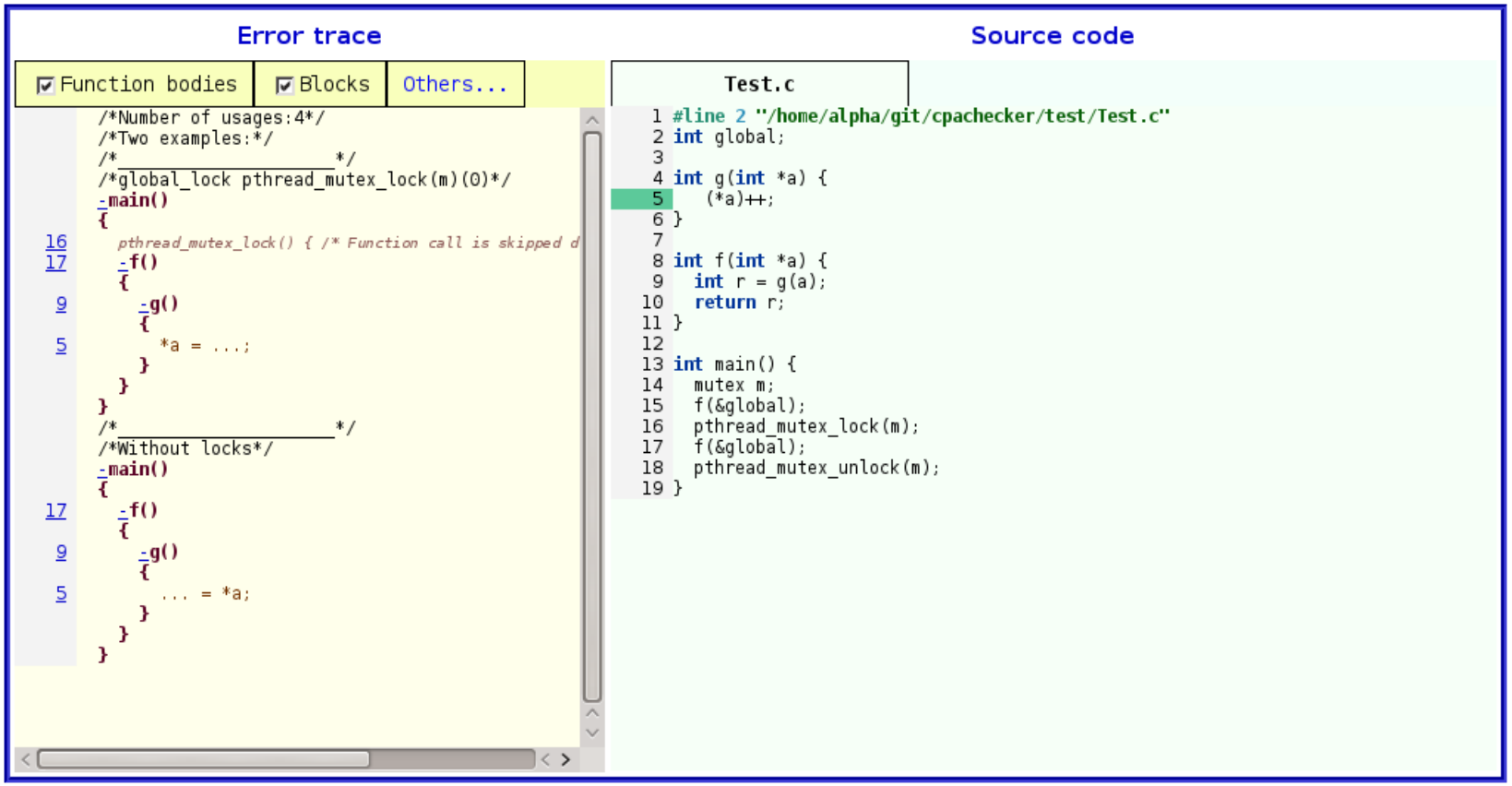
Finded locks:

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

List of unsafes:

Visualization of Unsafes

int *a



Main idea of static analysis

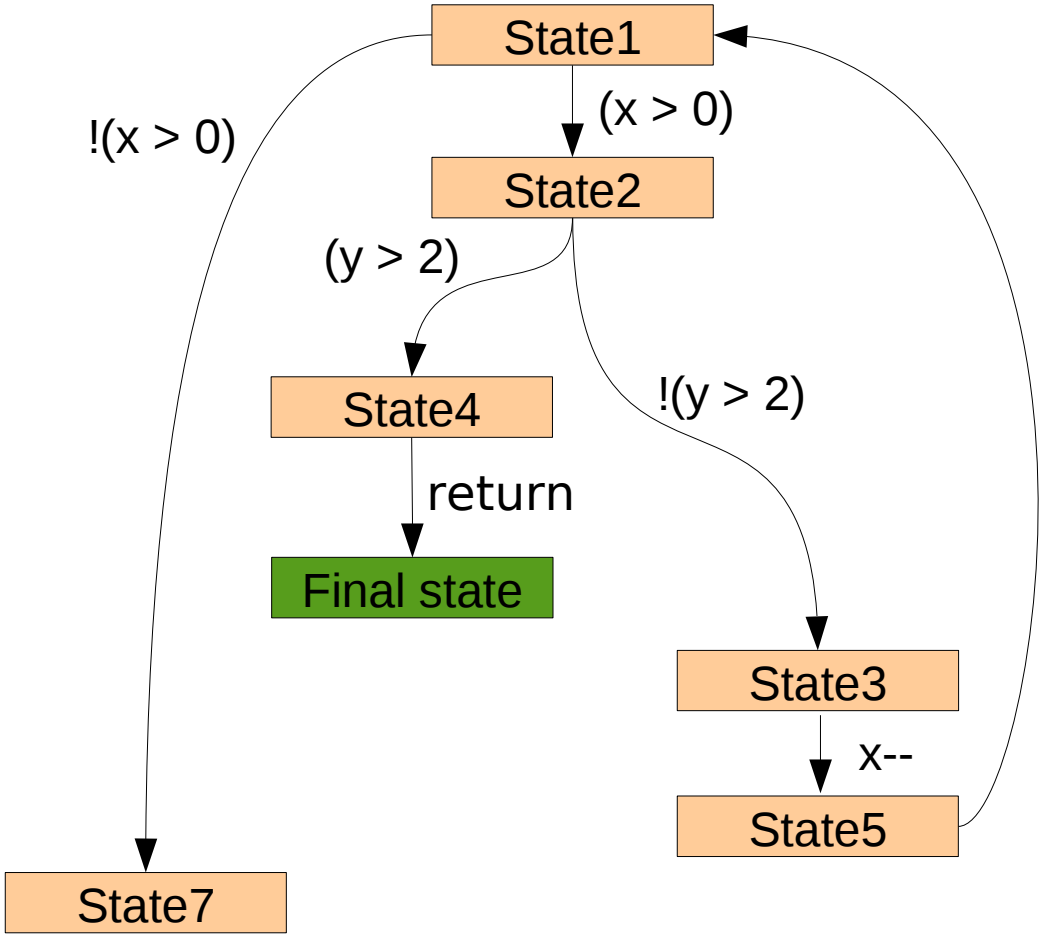
```
while (x > 0)
{

    if (y > 2) {

        return;

    }

    x--;
}
```



Configurable Program Analysis (CPA)

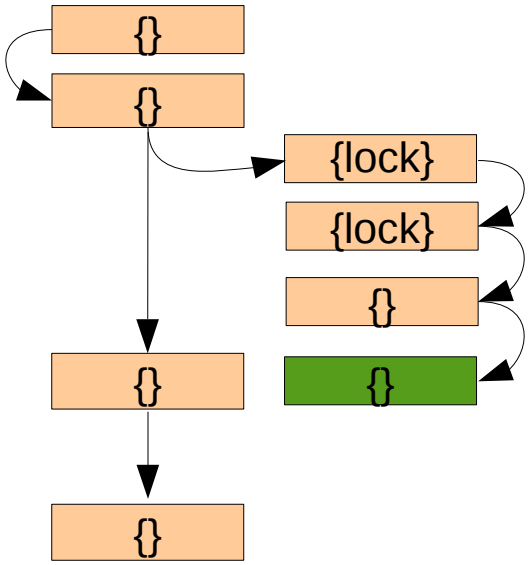
- Abstract domain – set of abstract states
- Transfer relation – for every state ***e*** it defines next state ***e′***
- Merge operator – combines information about two abstract states
- Stop operator – checks, if current state ***e*** is covered by set of others

Example: Lock statistics CPA

- Abstract domain – set of acquired locks
- Transfer relation – changes state, when function of acquiring or releasing lock is called
- Merge operator – states never combines
- Stop operator – state is covered, only if given set contains equal state

Example: LockStatisticsCPA

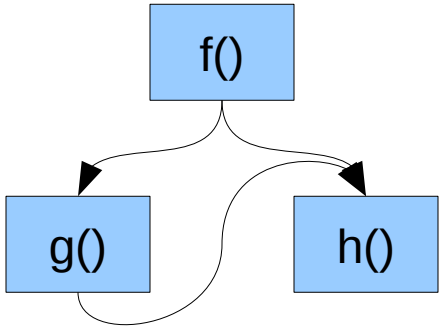
```
int global;
int func() {
        if (global) {
            lock();
            global++;
            unlock();
        }
}
```



Main problems

- Resources: time and memory
- Analysis of infeasible paths

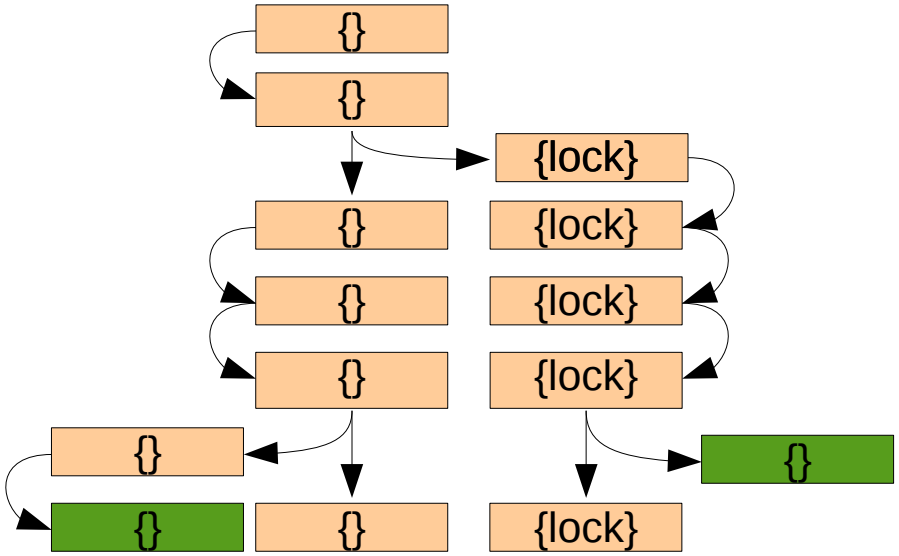
Abstract block memorization (ABM)



- Store the results of previous analysis
- Configurable blocks
- Greatly accelerate the analysis

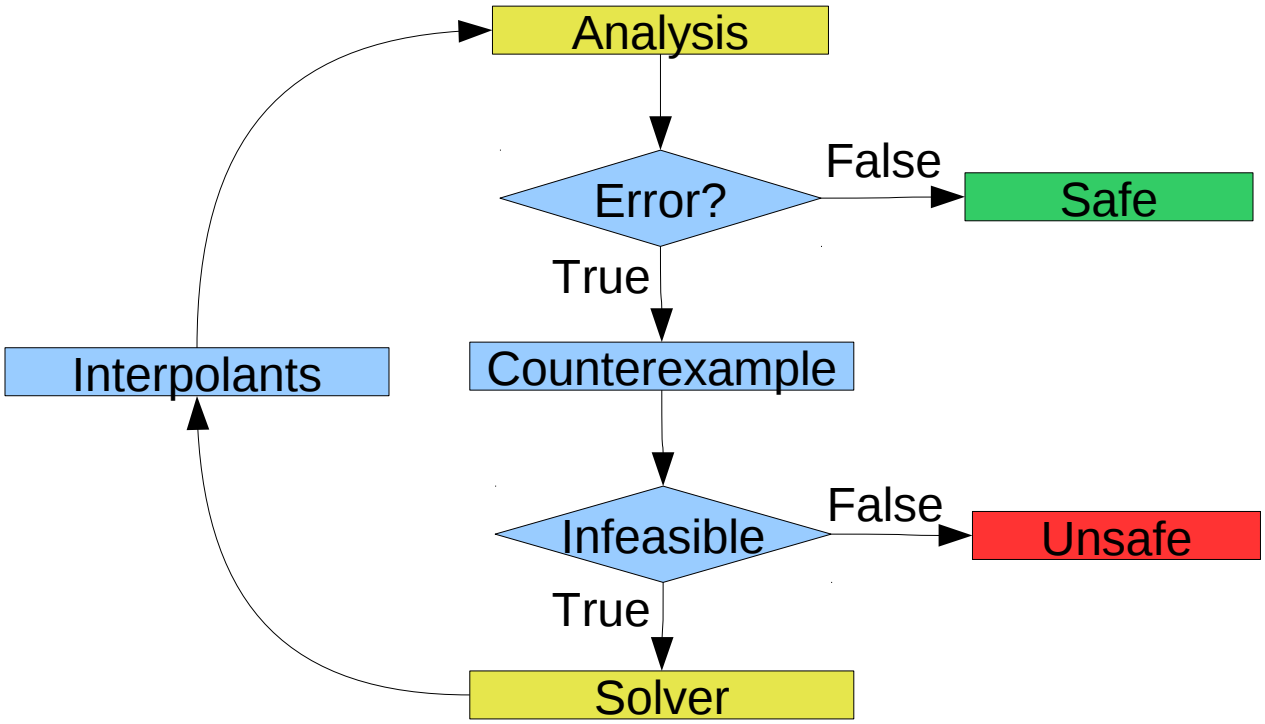
Example of infeasible path

```
int global;
int func() {
        int undef_var;
        ...
        if (undef_var) {
            lock();
        }
        global++;
        if (undef_var) {
            unlock();
        }
}
```



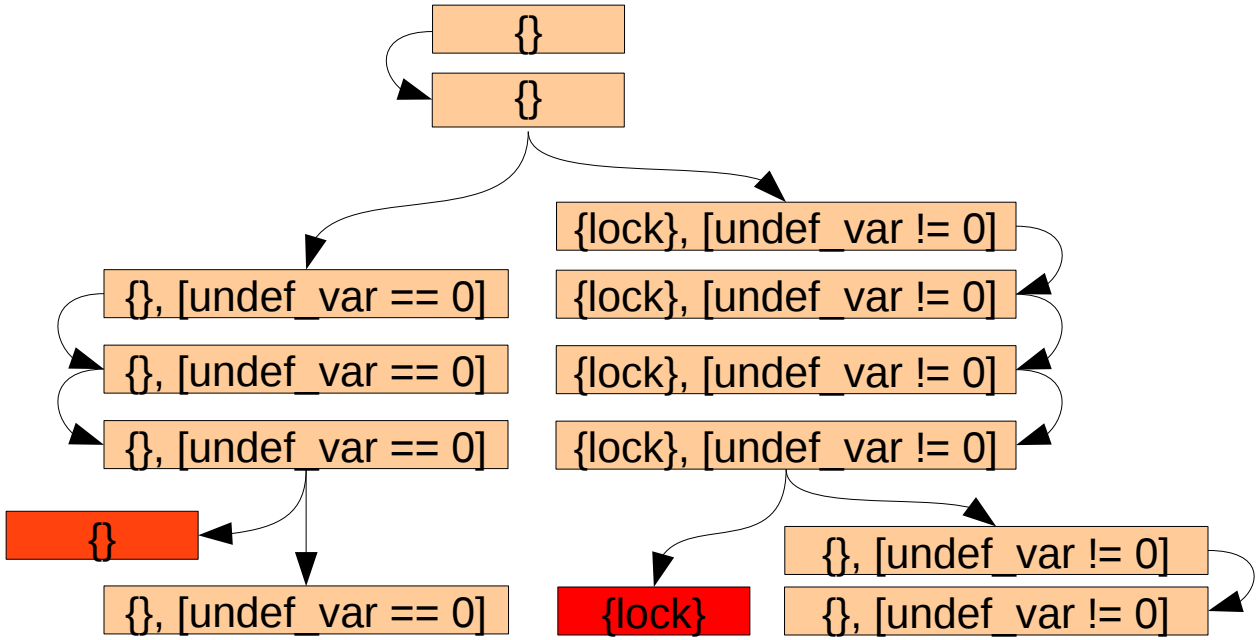
CEGAR

Counter Example Guided Abstraction Refinement



Example of infeasible path

```
int global;
int func() {
        int undef_var;
        ...
        if (undef_var) {
            lock();
        }
        global++;
        if (undef_var) {
            unlock();
        }
}
```



Based on open-source projects

- Linux Driver Verification (LDV)

  http://linuxtesting.org/project/ldv/
- CPAchecker

  http://cpachecker.sosy-lab.org

Thank you

Questions?