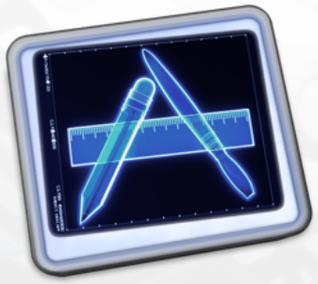
iPhone Application Programming
Lecture 8: Instruments

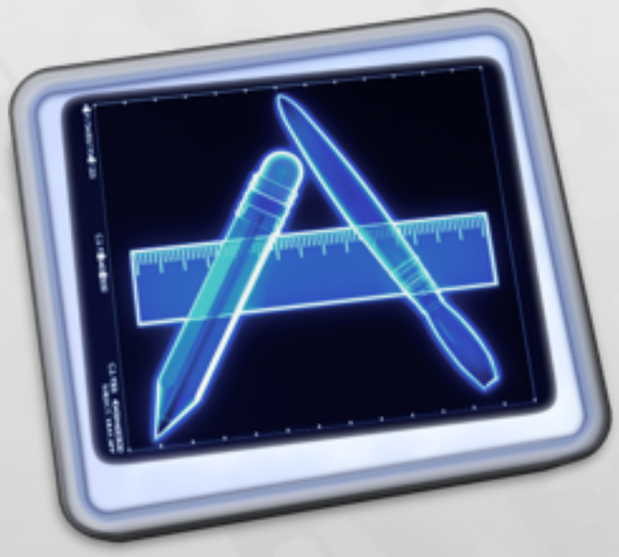
Moritz Wittenhagen
Media Computing Group
RWTH Aachen University

Winter Semester 2013/2014

http://hci.rwth-aachen.de/iphone

RWTH AACHEN UNIVERSITY





Instruments

Where is the problem?

```
int main(int argc, const char * argv[])
{
    @autoreleasepool
    {
        for(int i = 0; i < HUGE_VAL; i++)
        {
            NSArray *array = @[];
            [[array retain] autorelease];
        }
    }
    return 0;
}
```

Analyzing Runtime Behavior

- Memory
  - How much is used?
  - When is it allocated / freed?
- CPU Time
  - Where is it spent?
  - How is it distributed between multiple CPUs?

How is my algorithm doing things?
Not: Is it doing them correctly?

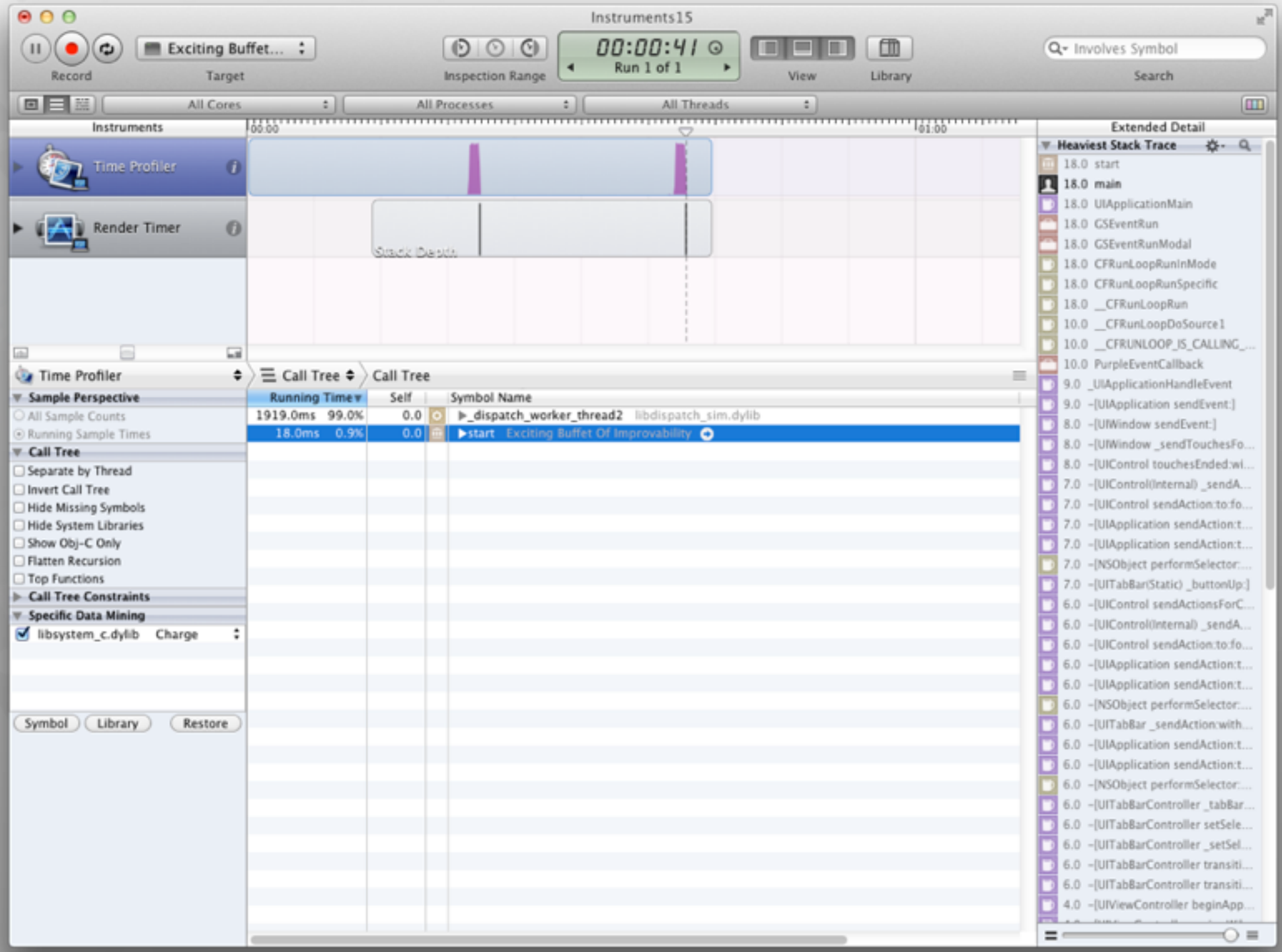
Instruments UI

Track view

Data view

Extended details

Strategies



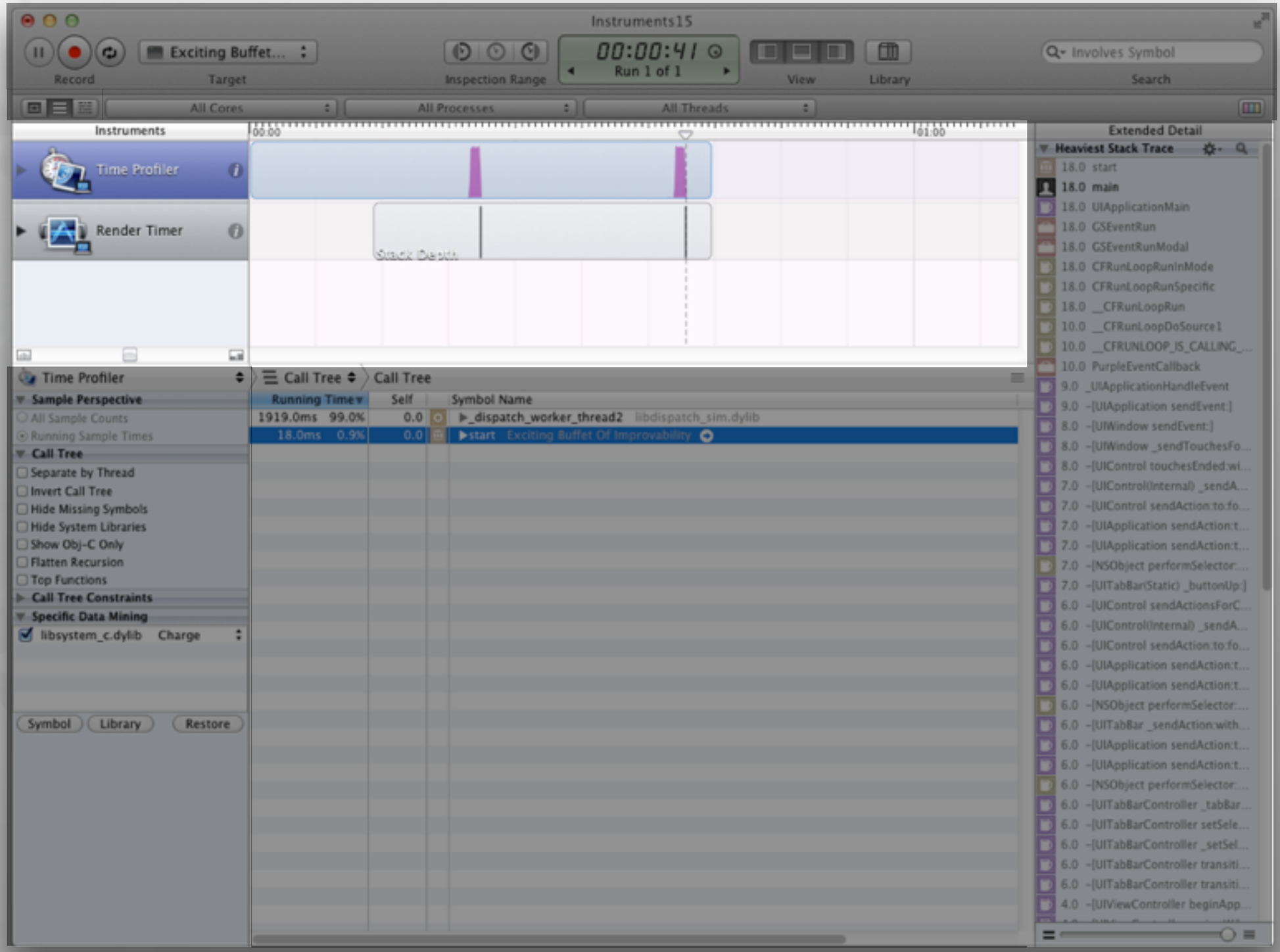
Instruments UI

Track view

Data view

Extended details

Strategies



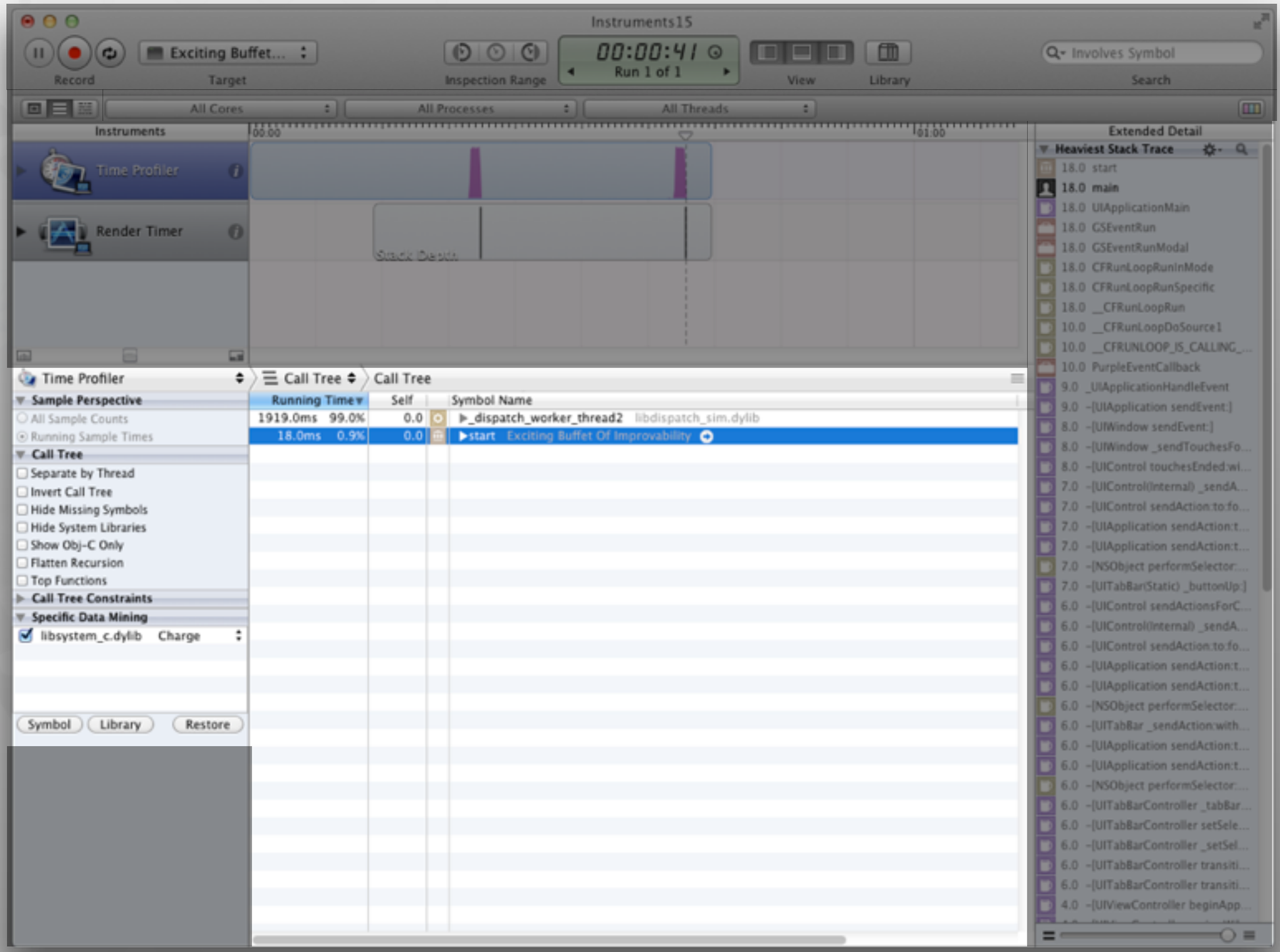
Instruments UI

Track view

Data view

Extended details

Strategies



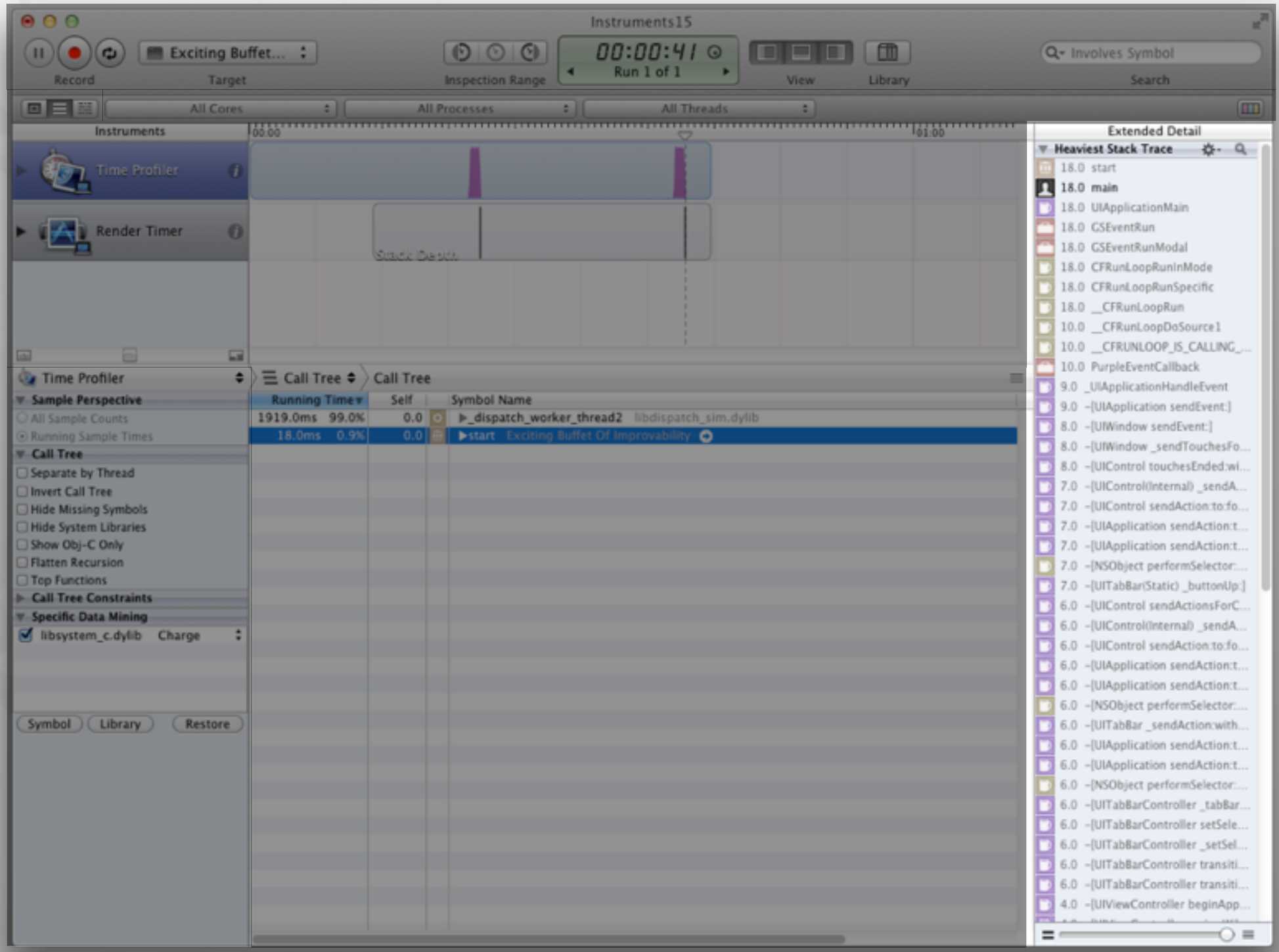
Instruments UI

Track view

Data view

Extended details

Strategies



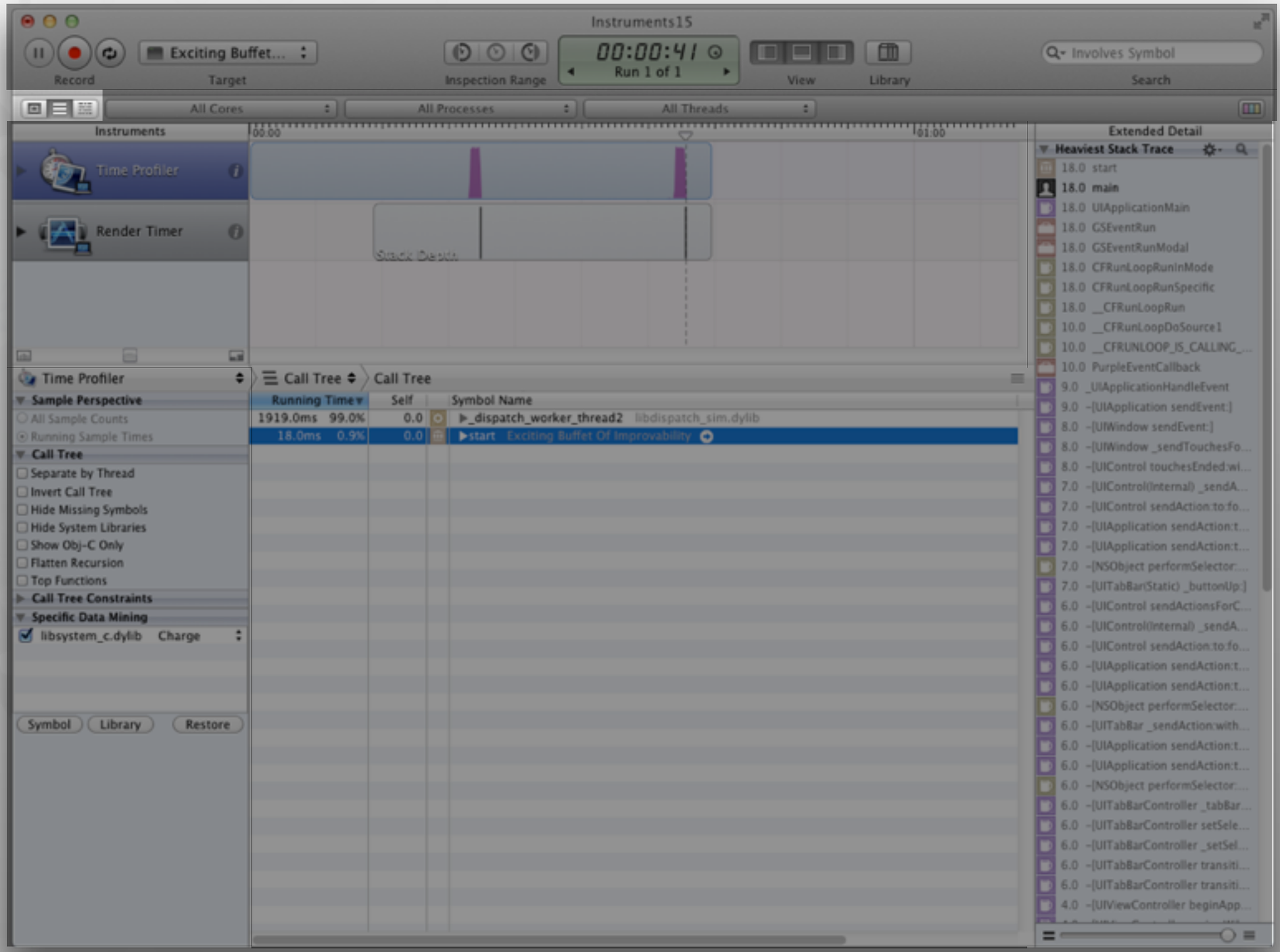
Instruments UI

Track view

Data view

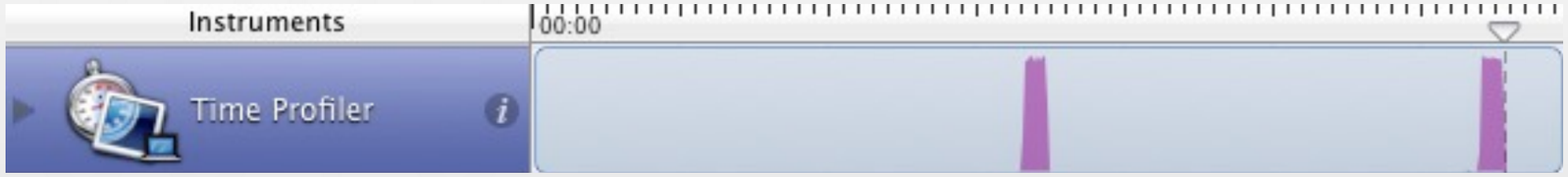
Extended details

Strategies



Strategies (Example)

- Instruments

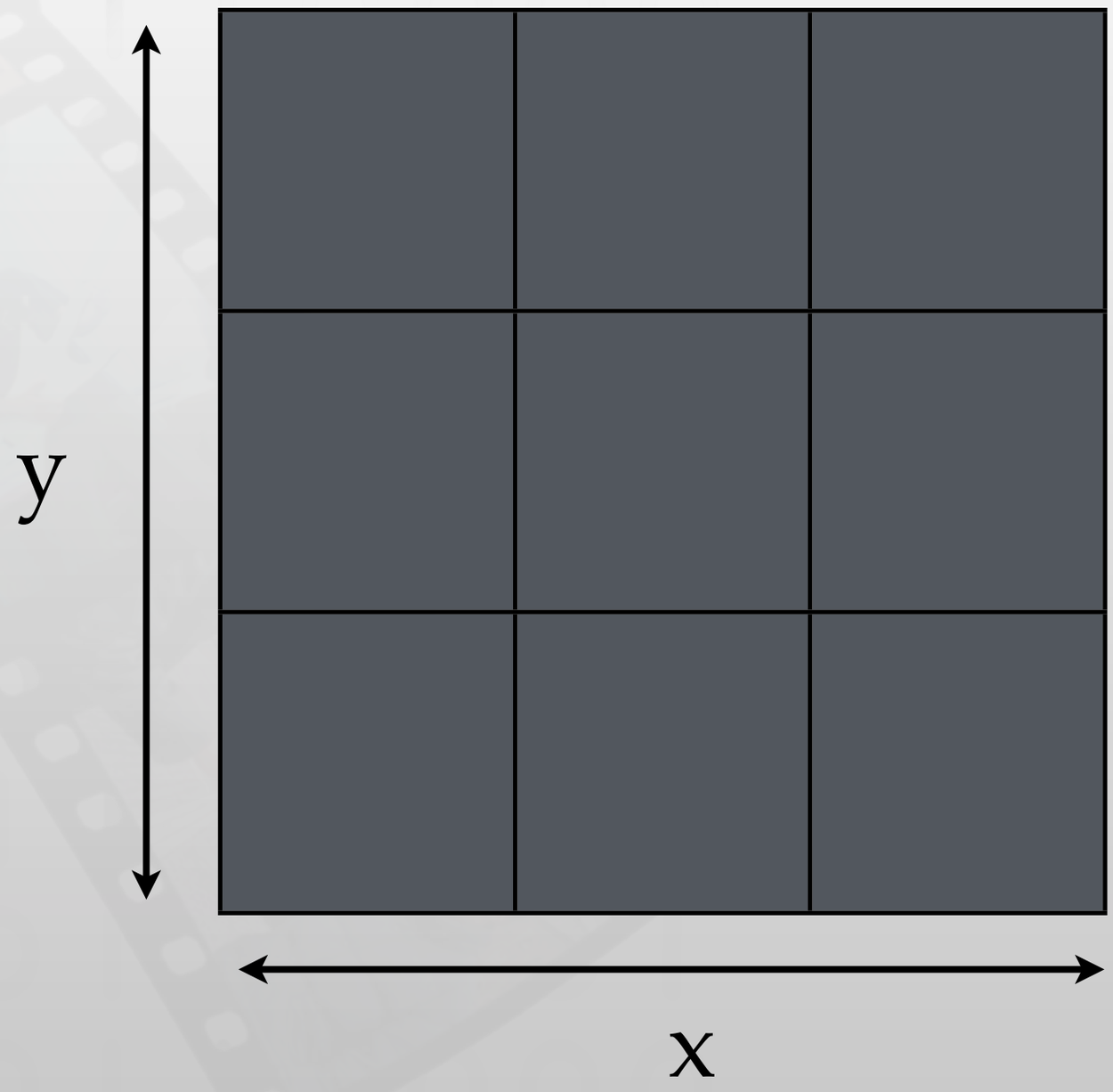


- Threads

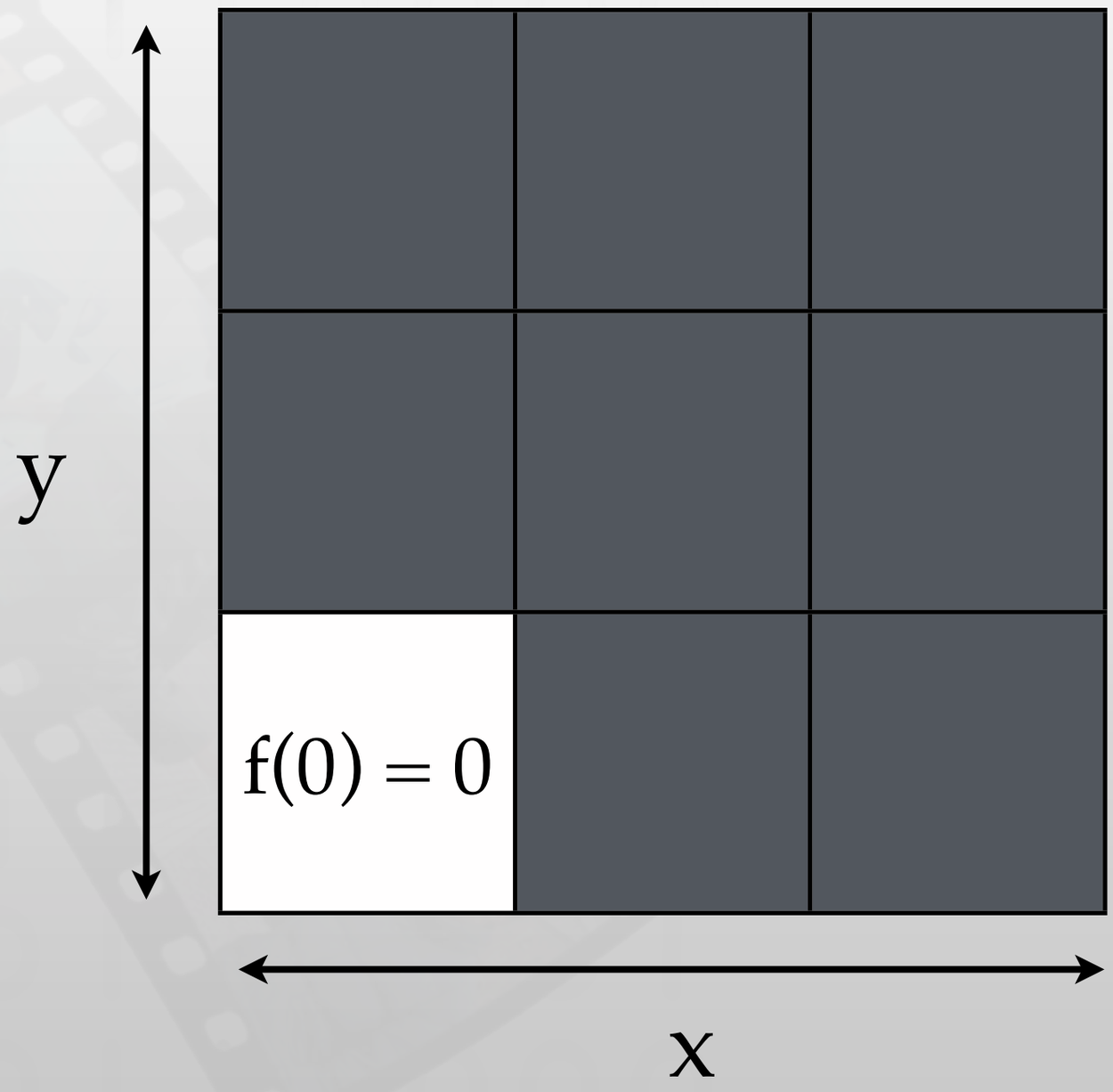
- CPUs

Demo Project

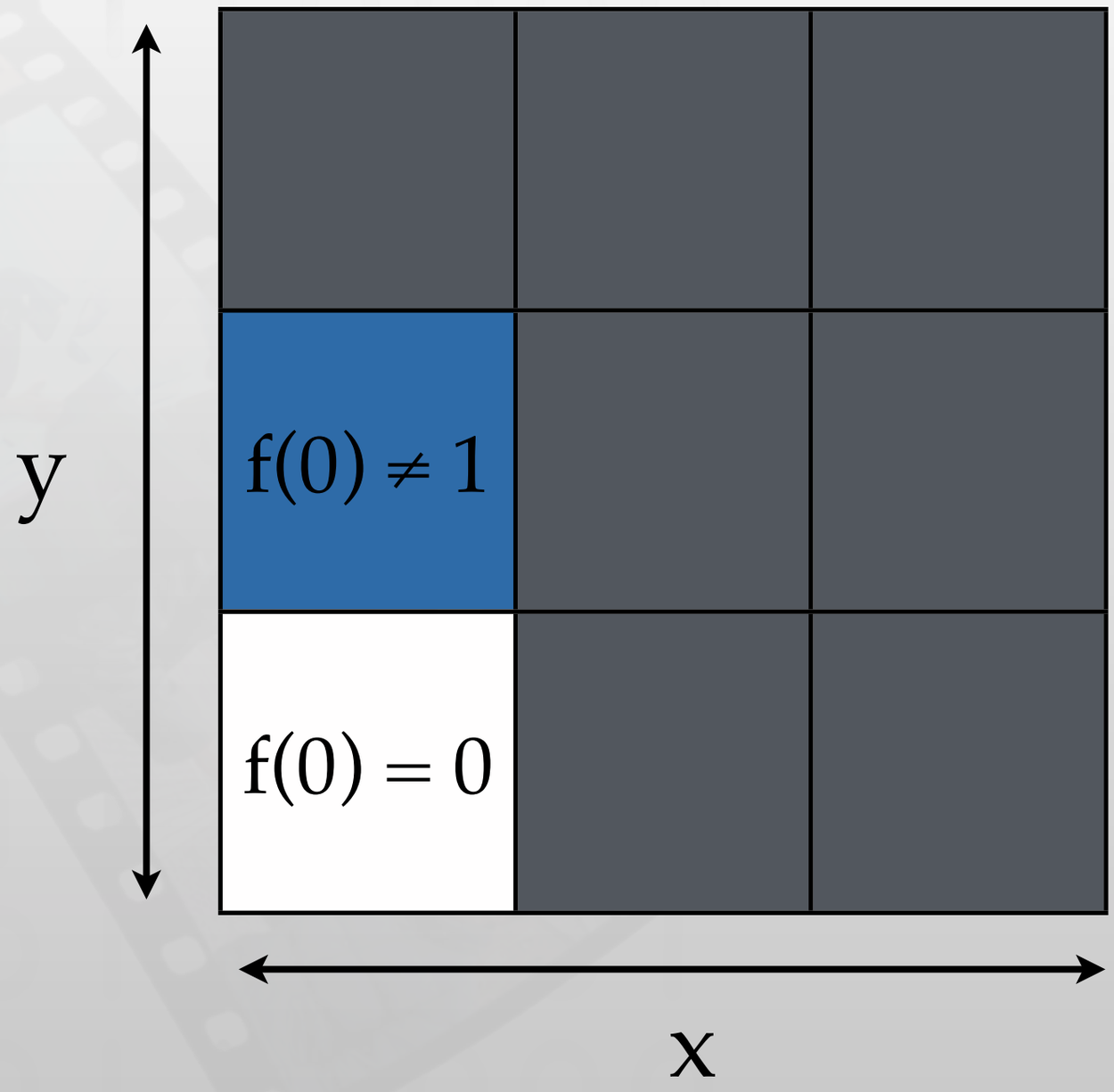
$$f(x) = x$$



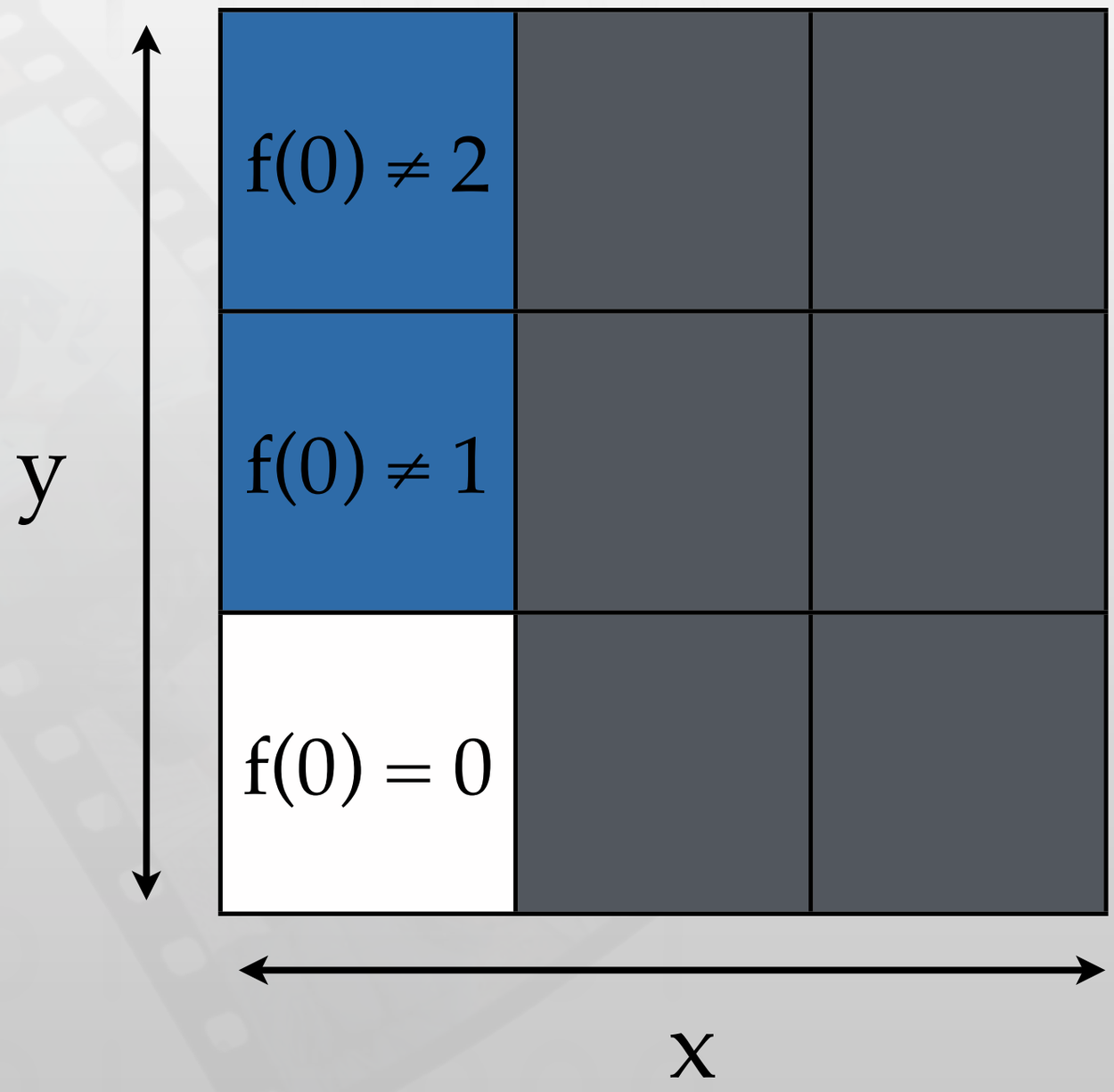
$$f(x) = x$$



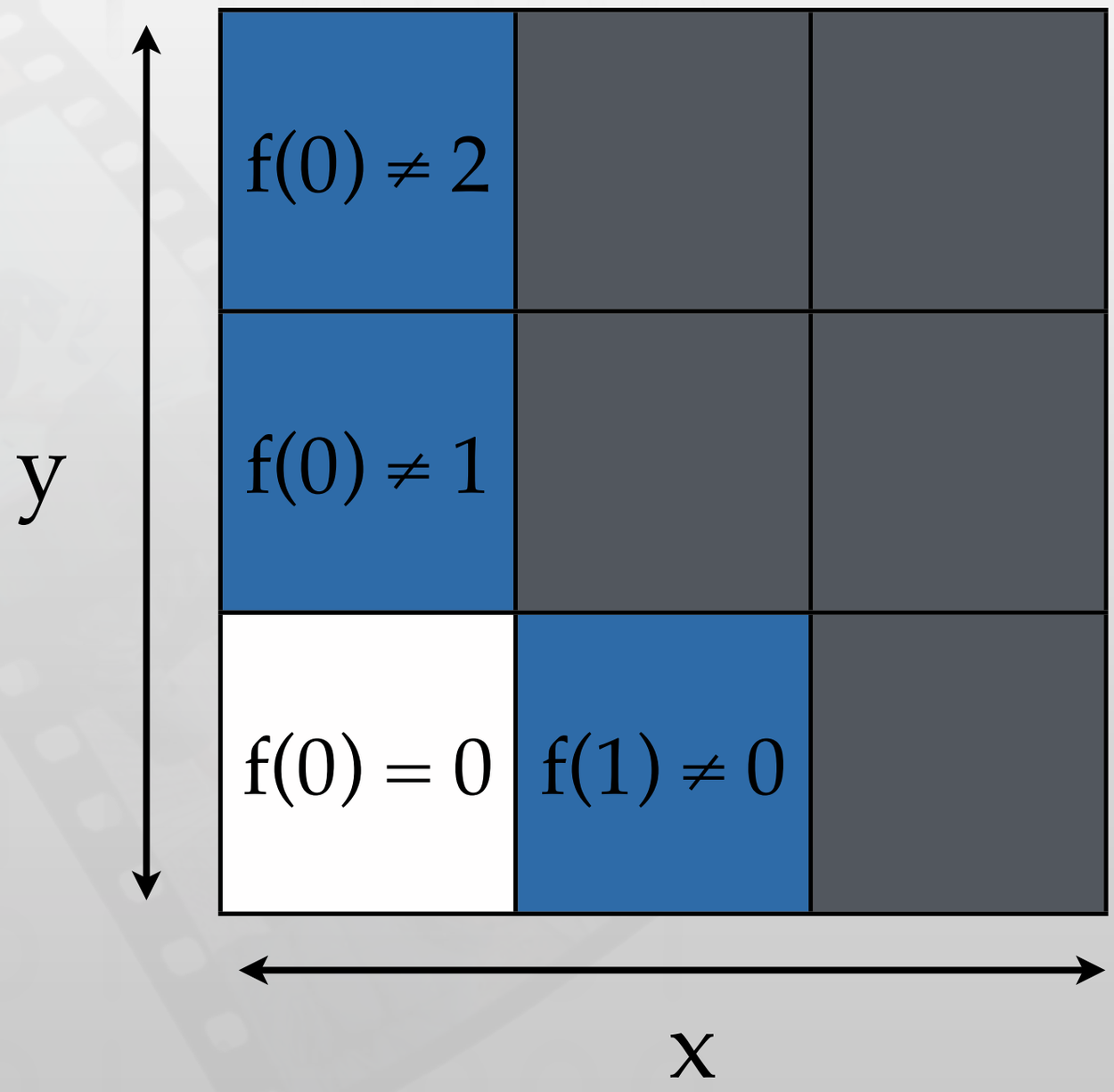
$$f(x) = x$$



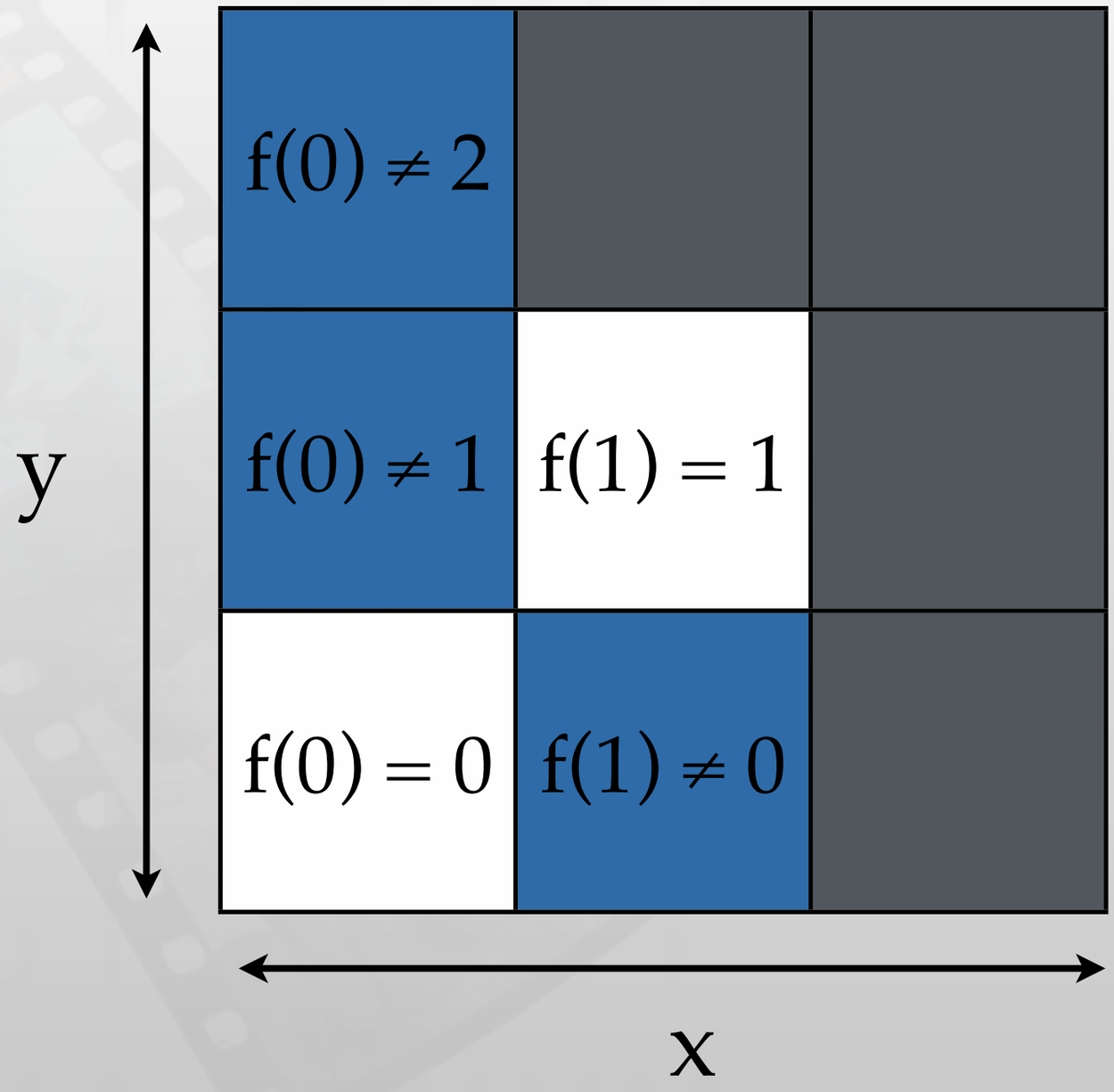
$$f(x) = x$$



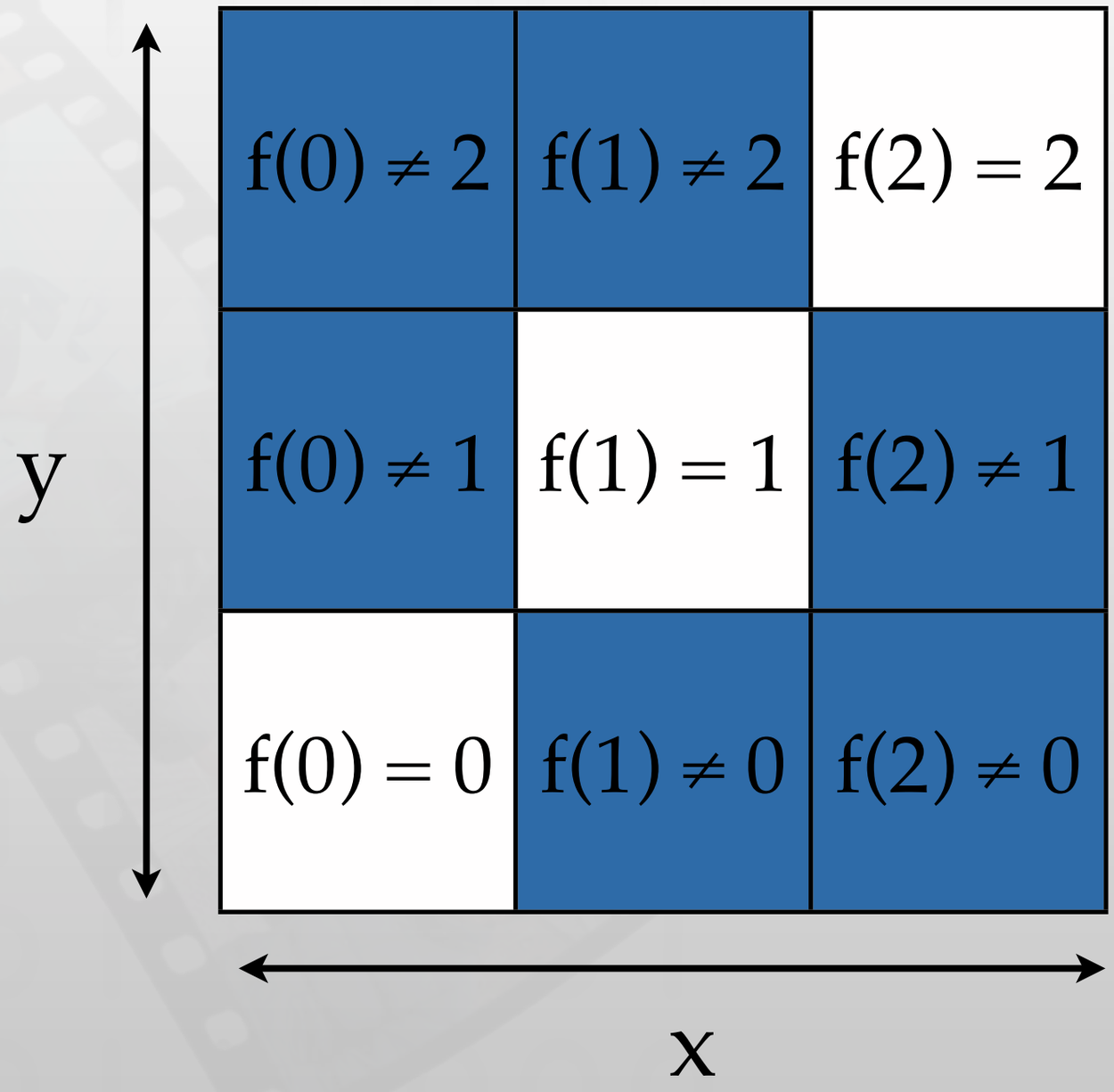
$$f(x) = x$$



$$f(x) = x$$



$$f(x) = x$$



Simple Optimization

Demo

Memory Analysis

- Allocation
  - Monitors memory allocation and reference counting

- Leaks
  - Checks for inaccessible memory
  - Finds retain cycles

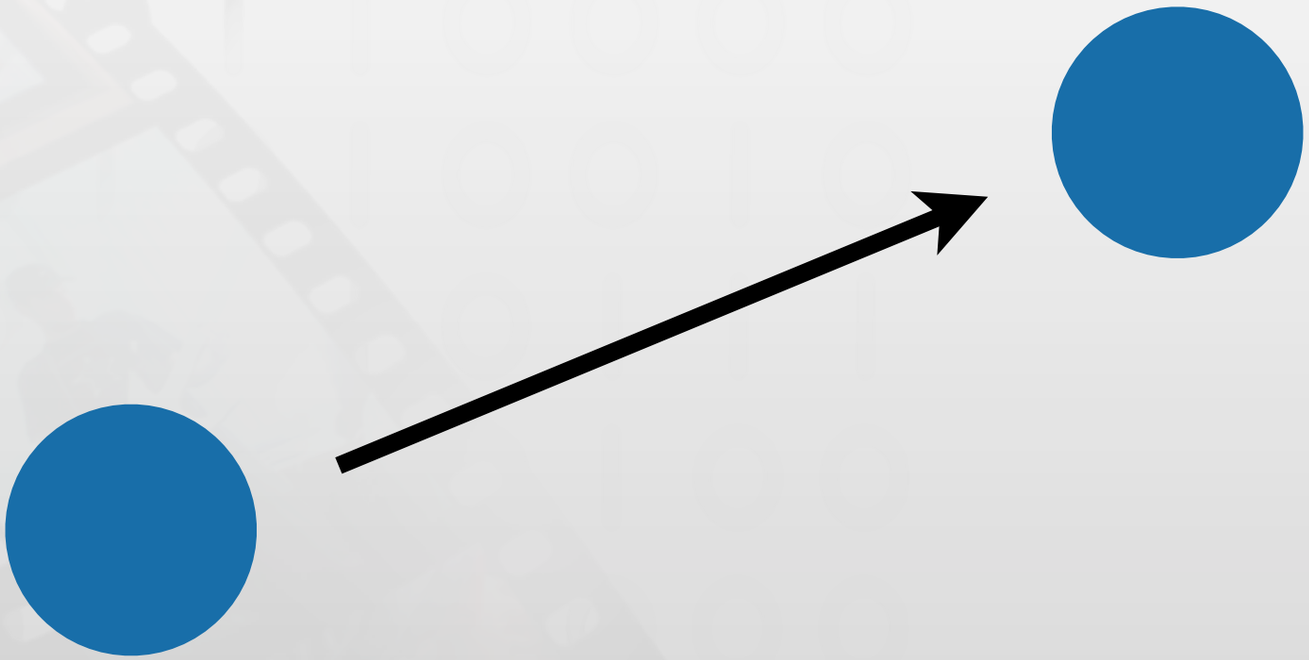
- Zombies
  - Checks for freed memory being accessed

Leaks

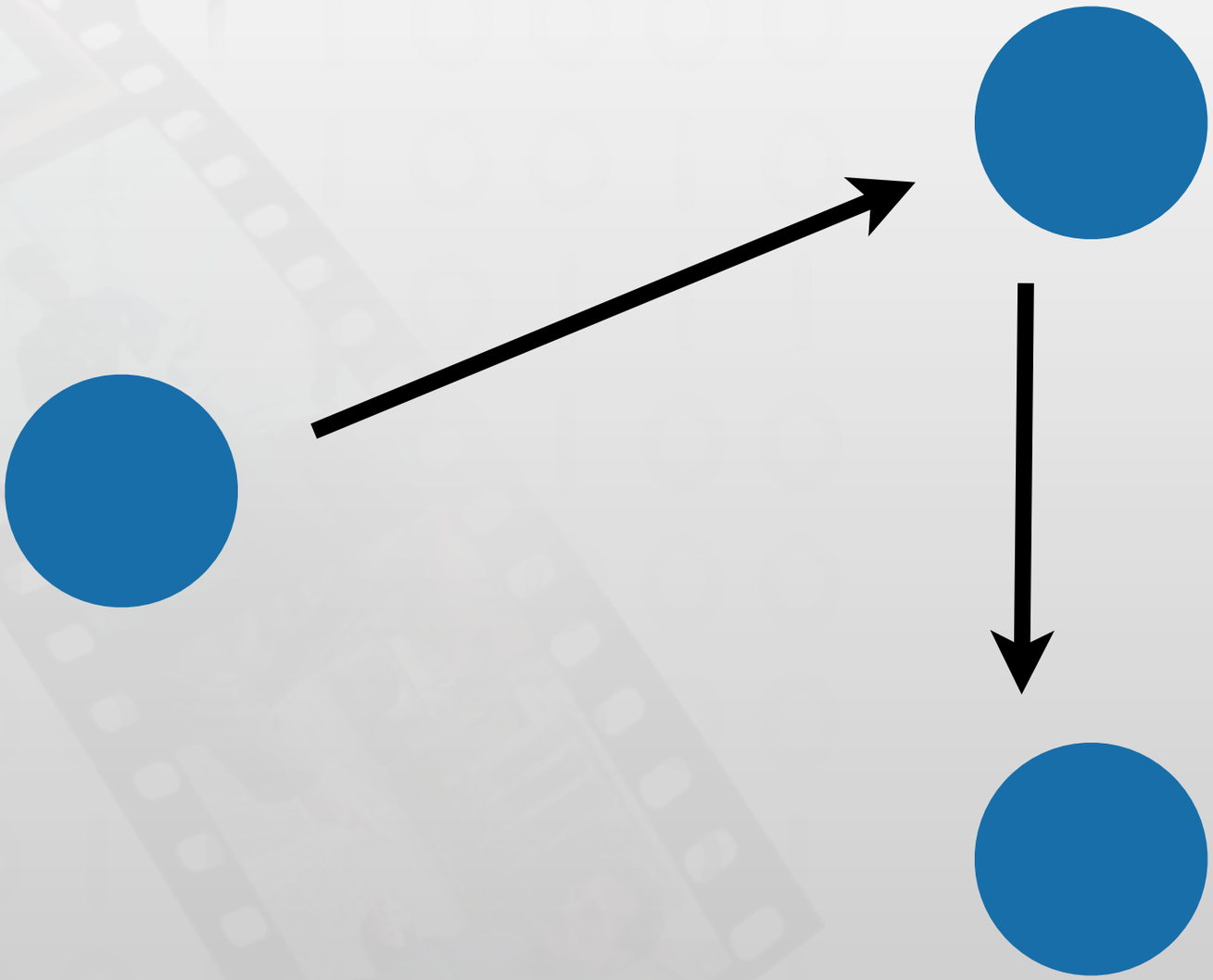
Leaks

Retain Cycles

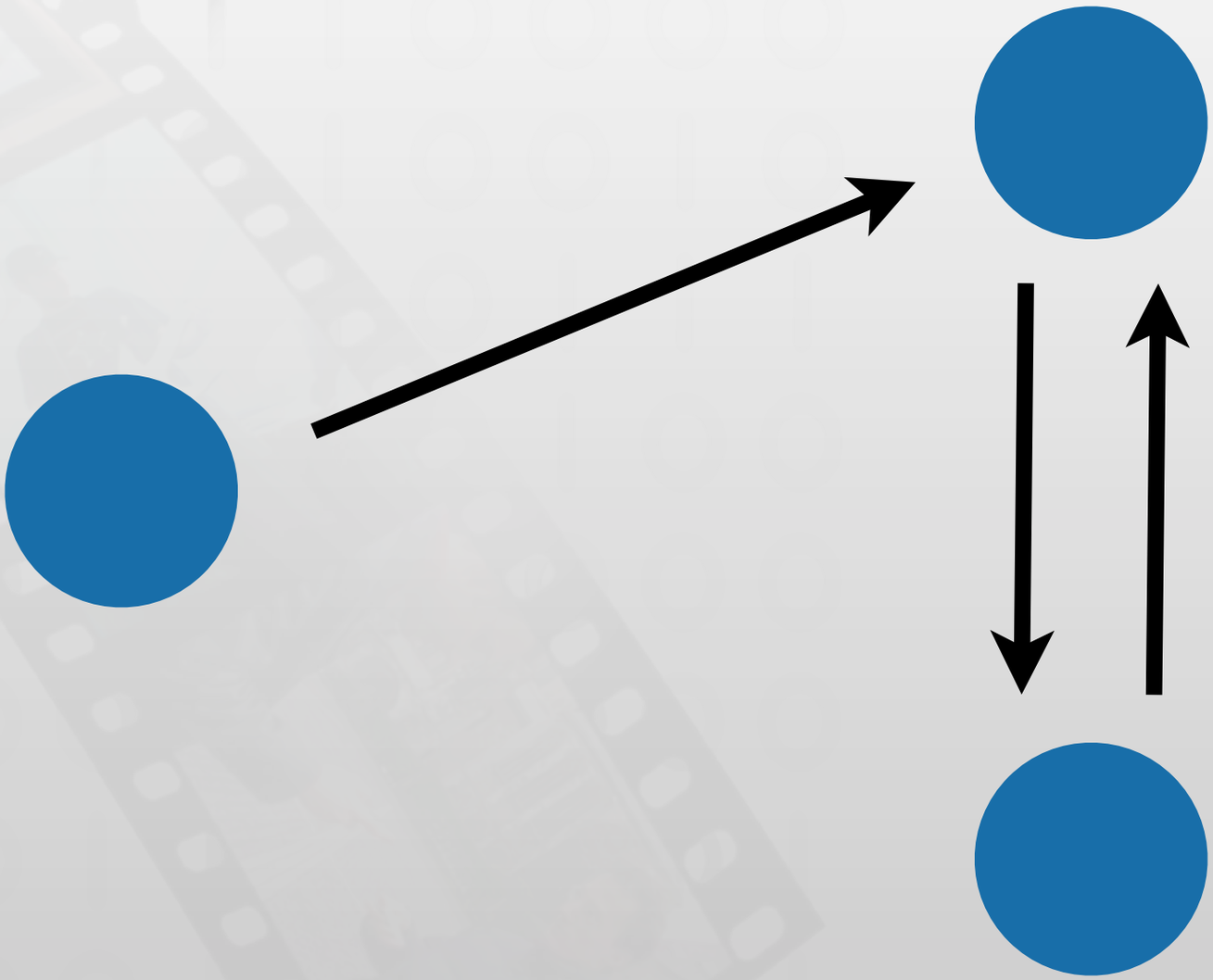
Retain Cycles



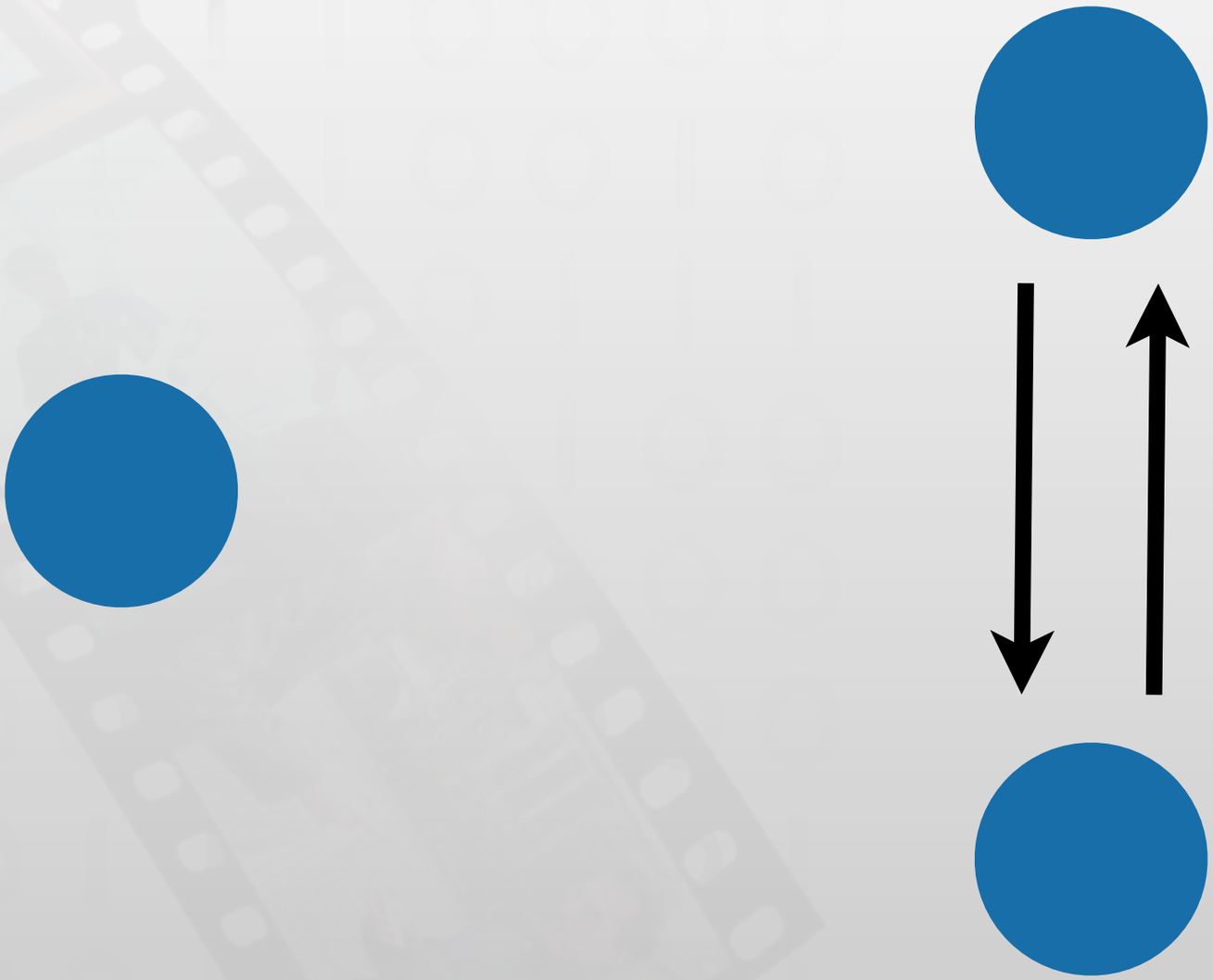
Retain Cycles



Retain Cycles



Retain Cycles



Allocations & Leaks

Demo

Zombies

Freed memory being accessed

- "Good" Zombies
  - Obvious crashes
  - You release, system reuses, you try to access
  - Crash (usually EXC_BAD_ACCESS)
- Bad Zombies
  - No crash, or crash at strange location
  - You release, you allocate something, you try to access
  - Weird side-effects

Zombies

Demo

Using Memory Instruments

- When you are done with a task: Leaks
- Whenever you get strange crashes or inexplicable values: Zombies
- You can use the simulator

Profiling

- Check in regular intervals what the CPU is doing
- Time Profiling
  - Where does the CPU spend time?
  - Distribution of work between threads / CPUs
- System Trace
  - What is the system doing?

    Thread scheduling

    Paging

    System calls

Time Profiling

Demo

Using Time Profiling

- When your app seems too slow
  - Identify hotspots
  - Identify opportunities for parallelization
  - Identify parallelization issues (e.g. forced serial execution)
- Use on iOS Device

System Trace

Demo

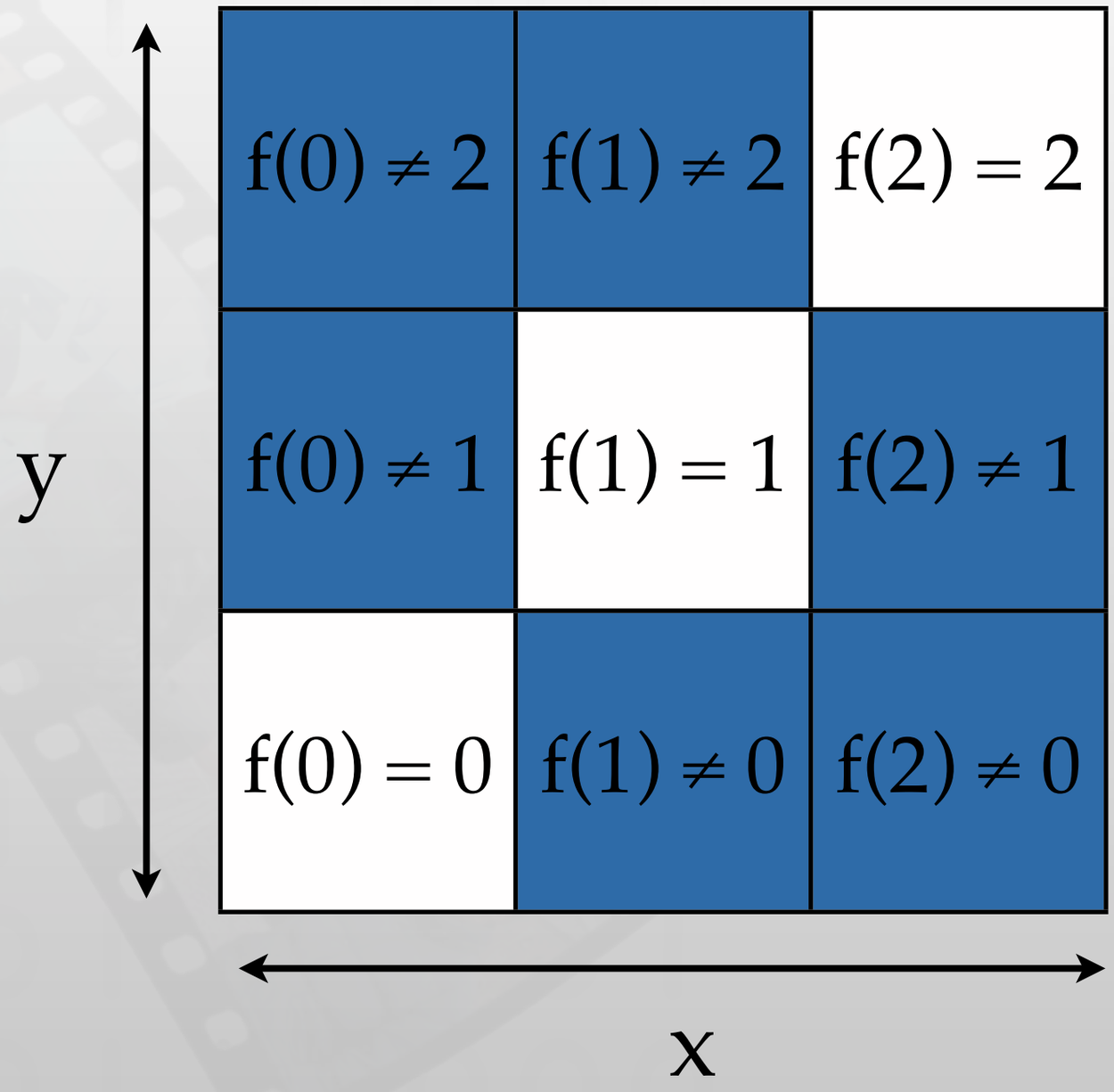
Using System Trace

- When results of Time Profiling are insufficient
  - Excessive context switching
  - Paging issues
  - Find opportunities to group system calls
- Use on iOS Device

$f(x) = x$

Layout in Memory

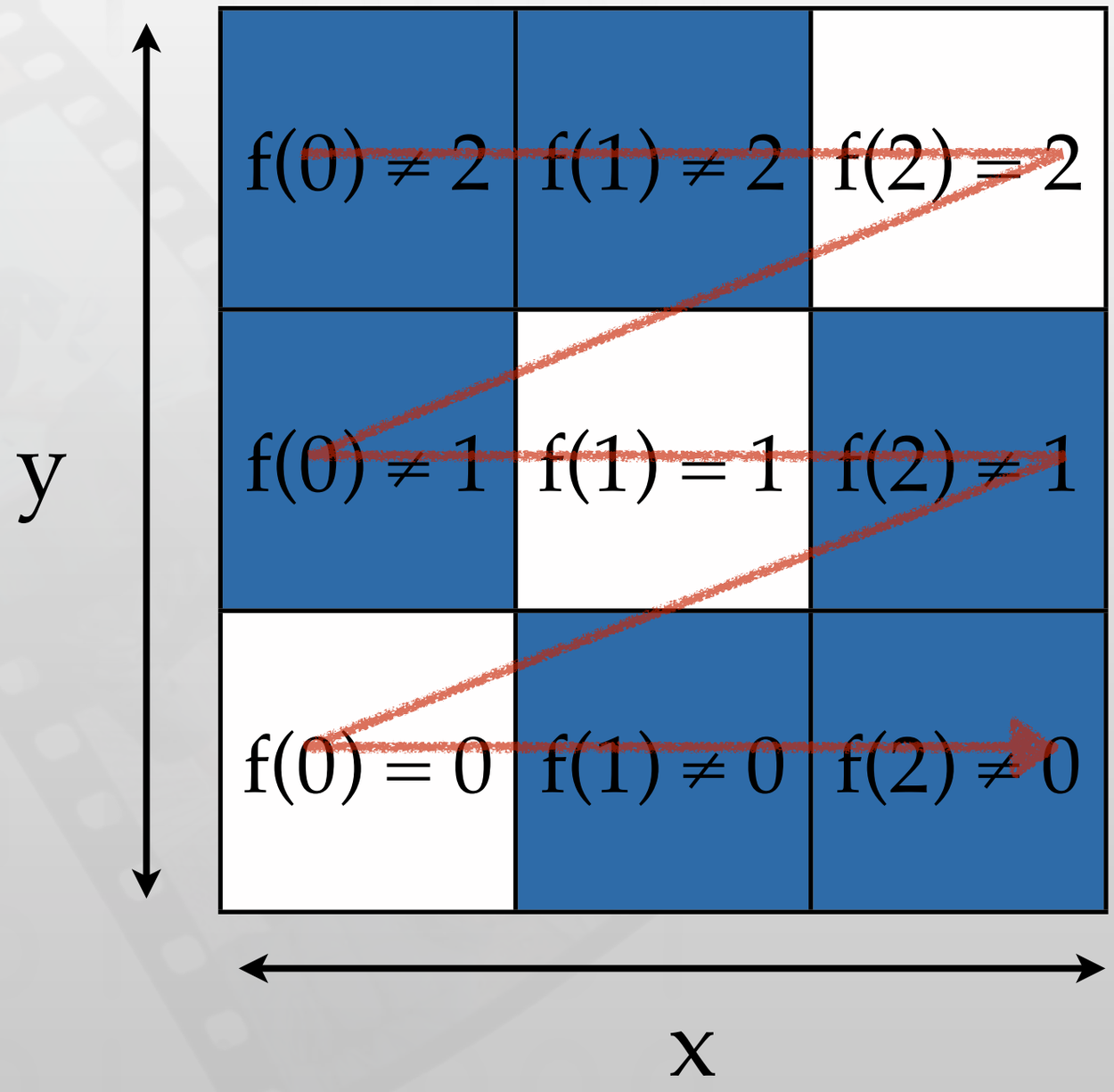
Order of Drawing



$f(x) = x$

Layout in Memory

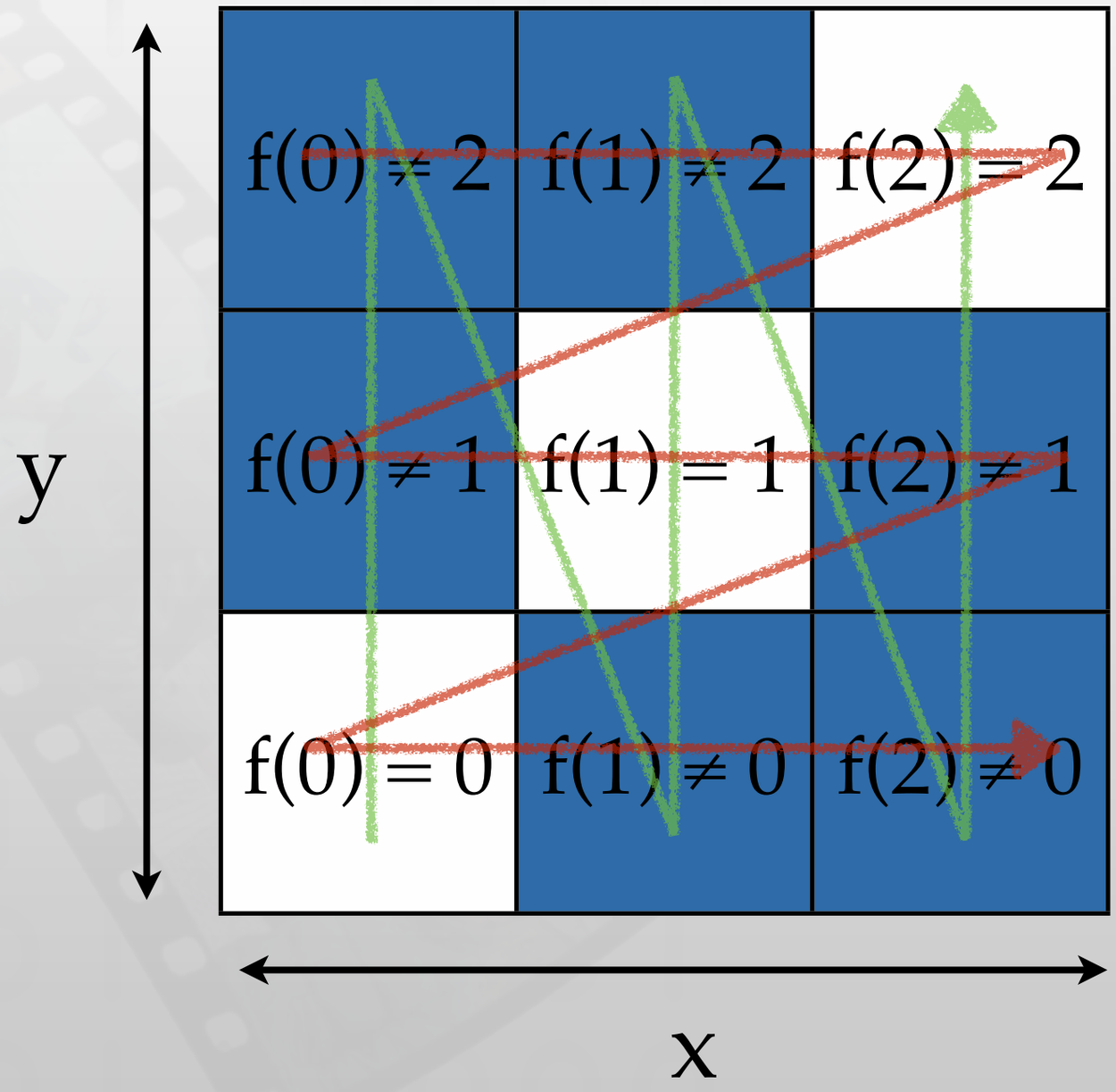
Order of Drawing



$f(x) = x$

Layout in Memory

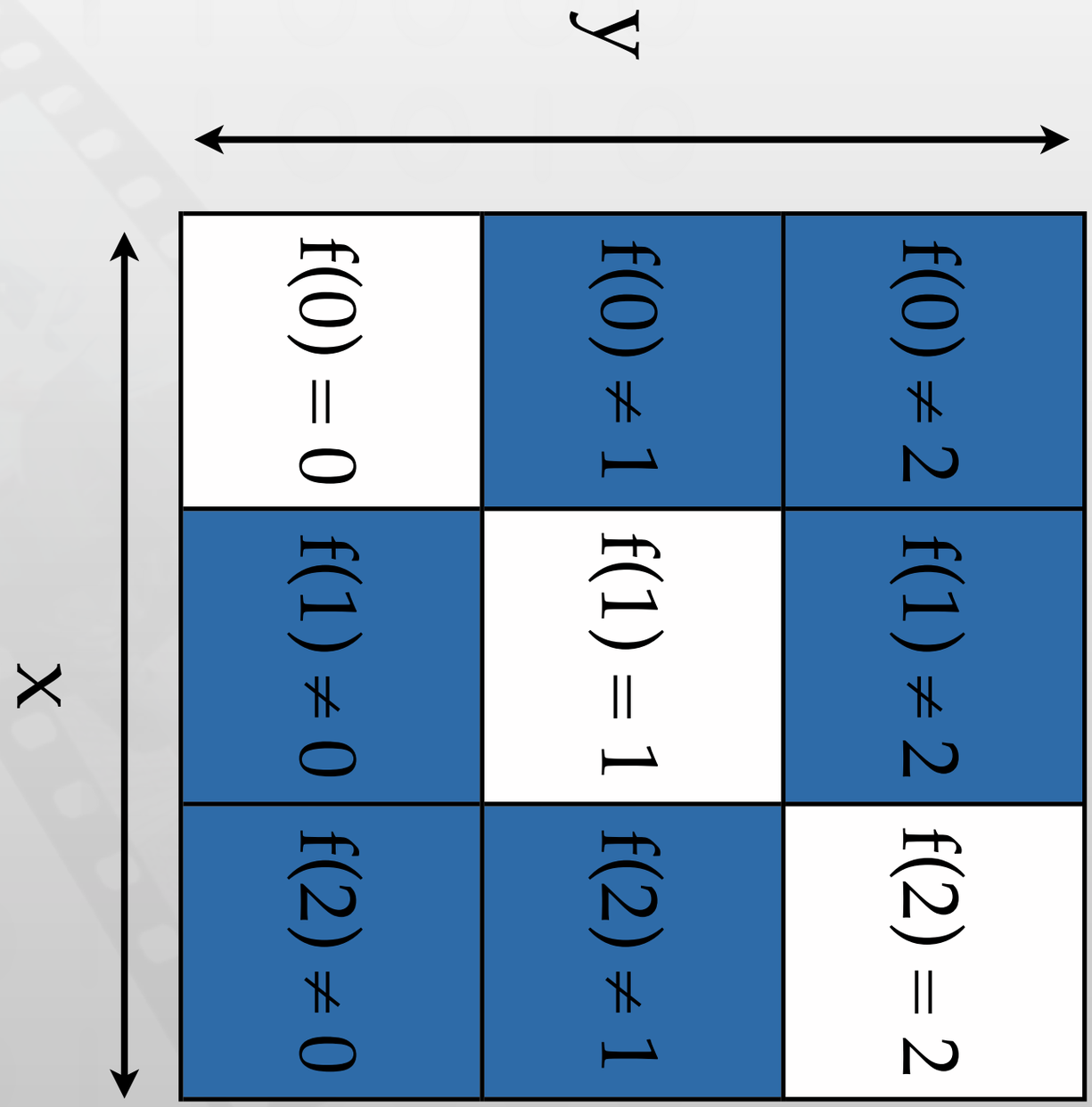
Order of Drawing



$$f(x) = x$$

Layout in Memory

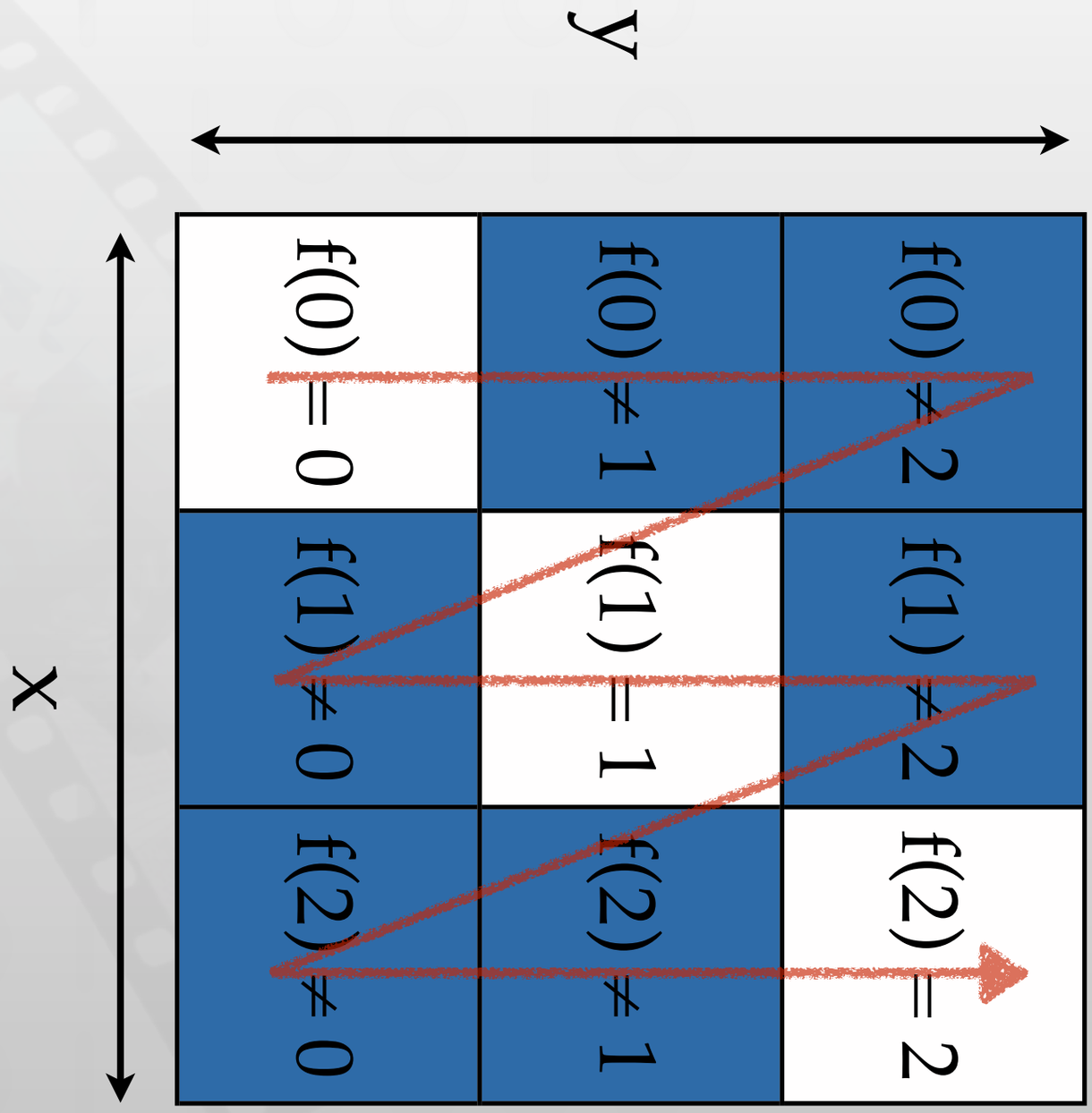
Order of Drawing



$$f(x) = x$$

Layout in Memory

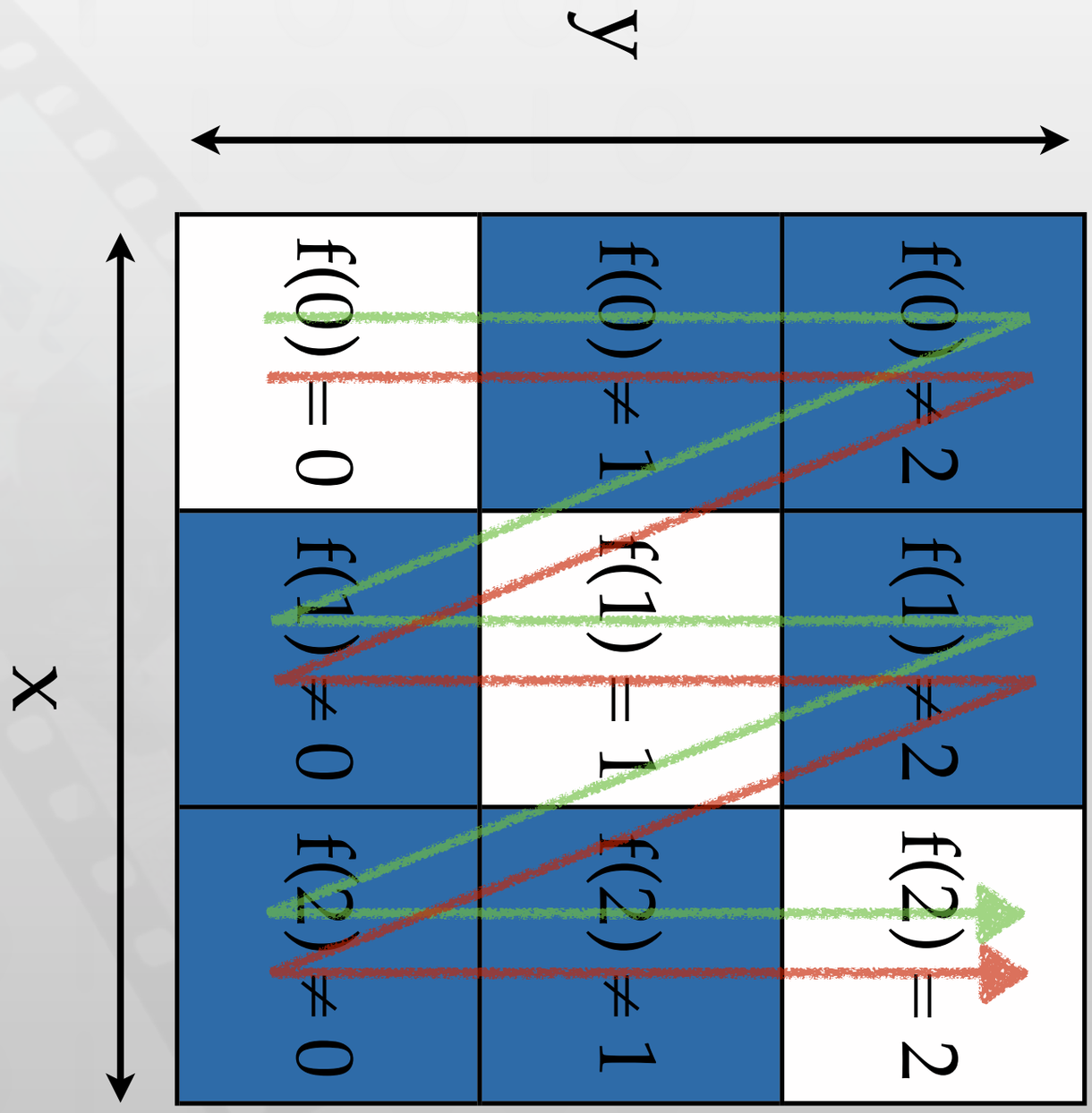
Order of Drawing



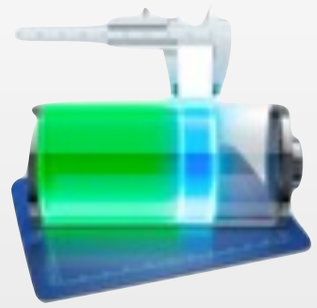
$$f(x) = x$$

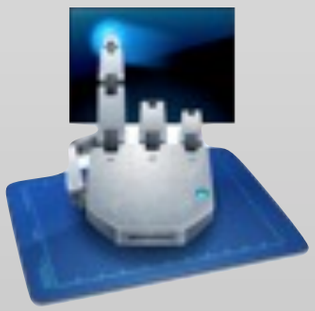
Layout in Memory

Order of Drawing



Other Instruments





- Energy Diagnostic
- Core Animation
- OpenGL ES
- System Usage
- UI Automation

Summary

- General
  - Find bugs at runtime
  - Increase algorithmic efficiency
- Profiling
  - Identify bottlenecks
  - Parallelization
- Memory Instruments
  - Sanity checks to find leaks and zombies
  - Increase memory efficiency