Seminar in **automatic tools** for analyzing **programs** with **dynamic memory**

Noam Rinetzky

http://www.cs.tau.ac.il/~maon/teaching/2013-2014/seminar/Seminar1314b.html

Schreiber 326

Semester B. Sunday, 14:00-16:00. Dan David 201

Scope

- Automatic tools for analyzing / executing programs with dynamic memory

Programs with dynamic memory

- Programs manipulate resources
  - Files
  - Processes / threads
  - Connections
  - **Memory**

- malloc() / new()
- free() / delete() / -
  - GC

Programming with dynamic memory

```
typedef struct Data {int d; struct Data *n} Da;

main(){
    Da *p1 = (Da*) malloc(sizeof(Da));
    p1→d = SECRET_KEY;
    Da *p2 = (Da*) malloc(sizeof(Da));
    p2→d = 0;

    p1→n = p2;
    p2→n = null;

    free(p1);
    free(p2)
}
```

Common Mistakes

```
typedef struct Data {int d; struct Data *n} Da;

main(){
    Da *p1 = (Da*) malloc(sizeof(Da));
    p1→d = SECRET_KEY;
    int d1 = p1→d;

    Da *p2 = (Da*) malloc(sizeof(Da));
    int d2 = p2→d;

    p1→n = p2; p2→n = p1;

    free(p1);
    int d11 = p1->d;
    Da *p3 = (Da*) malloc(sizeof(Da));
    int d3 = p3→d;

    Da *p4 = null;
    int d4 = p4→d;

    free(p1);
}
```

Common Mistakes

- Accessing uninitialized fields
- Double free
- No free

- Null-dereference
- Breaking invariants
  - `p1→n = p2; p2→n = null;`
  - `p1→n = p2; p2→n = p1;`

Undesired Outcome

- Crashes
- Incorrect behavior
- Security vulnerabilities

- Loss of life
- Loss of money
- Loss of reputation
- Loss of Job

Programming Challenges

- Unbounded number of resources
  - Heap = global resource
  - No “names”
    - In direct access: pointers + pointer arithmetic's
- Complicated data structures
  - List, trees, graphs
    - Hard to maintain invariants
    - Sharing is challenging
- Multithreading makes things worse!

Deallocation

- Allocation is “easy”
- “Deletion” is hard

- Nasty bugs
- Hard to get right
  - Defensive programming

Programming Model

- Imperative programs
  - Java, C, C++
- Sequential + multithreaded
- free() / GC
  - Manual (free())
  - Automatic (GC)

Solutions

- Manual memory management
  - Runtime: Monitoring execution environment
    - Catches errors
    - Expensive
  - Compile-time: Verify memory safety
    - Static analysis
      - Fully automatic / User-provided annotations
    - Conservative
      - Problem is undecidable

Solutions

- Automatic memory management (aka Garbage Collection)
  - Runtime environment recycles memory that will not be used in the **future** of the execution
    - Unused = unreachable by the program
      - Object is garbage if it cannot be reached via a directed path from the roots.
  - Pros:
    - Safe
    - Simple
  - Cons
    - Runtime overhead
    - Imprecision (drag)
      - Compile time garbage collection can help

Topics – Manual Mem. Mang.

- Runtime environments
  - LINT and its derivatives
  - Specialized memory allocators
- Pointer Analysis
  - The “golden age”
  - Scalable and precise
- Escape Analysis
- Shape Analysis

Topics – Manual Mem. Mang.

- Sequential garbage collectors
  - Reference counting
  - Tracing
  - Copying
- Concurrent garbage collectors
  - Dijkstra, Steele, Yausa
  - Stop the world
  - Parallel
  - Concurrent

Topics - Last lesson

- 5 minutes summary
- Discussion

Admin

Grades

- 70% Presentation
  - 1 paper per person
  - Lecture ~ 1 hour
- 5% Original insight
- 10% Participation
  - Ask questions, clarifications, initiate discussions
- 15% Attendance
  - It is OK to miss 1 talk
    - Not yours!

Presentation

- Choose paper
- Read paper
- Make draft presentation
  - Use THIS theme
- Meet me
  - Discuss paper
  - Go over presentation
- Present paper
  - Describe work
  - Lead discussion
- Send me presentation + 2-3 paragraphs on paper, discussion, your own insight

Paper Title

Names of Authors

Your Name + date

Outline of talk

- Introduction
- Suggested Solution
- Evaluation
- Related work
- Conclusions
- Your *own* conclusions

Introduction

- Problem area
- Technical challenged addressed
- Why is it important
- What is the main insight
- How is main insight utilized (high level)

Solution

- Technical description
  - Algorithm
  - Correctness
  - Complexity

- Choose key subset of details
- Use examples + diagrams

Evaluation

- Experiments
- Benchmarks
- Conclusions

Related work

- What other solutions are out there
- How do they compare
  - Pros
  - Cons

Conclusions

- What was done
- Why is it important
- Novel idea
- What we learned

Your own conclusion

- Surprise me