CMU REUSE Notes

Eduardo Lozano

Summer 2024

Contents

Introduction

This document contains detailed notes for the 10-week research internship focused on coding and potential paper publication.

1 Week 1: Introduction to Project

This

2 Week 2: Literature Review

2.1 Day 1 - Learning decomp basics

Tasks:

- Make another branch on the recomp-verify repo.
- Make a new benchmark folder: toy_mutex.
- Make a new constants folder 2-1.
- Create the .tla and .cfg files (see other benchmarks for examples).
- Decompose the files by hand.
- Check your work using recomp-verify.
- Run: `python3 ......../..../recomp-verify.py ToyMutex.tla ToyMutex.cfg -naive`.

Summary:

Had trouble compiling because constants aren't in config file. Java compiles faster.

Potential Concerns:

Where to read papers / how to read effectively.

2.2 Day 2 - Algorithm and Logic for State Graph Traversal

Tasks:

- Algorithm for checking a model (use BFS).
- Read some articles and watch video given.

Summary:

Article [How Amazon Web Services Uses Formal Methods] takeaways:

- Formal Methods scare new methods.
- Formal Methods - mathematically rigorous techniques for the specification, analysis, and verification of software and hardware systems.
- Safety - what the system is allowed to do.
- Liveness - what the system is must eventually do.
- Helps with communication and all engineers understanding a description of a design.

- Formal Specification is not good for bugs and operator errors that cause a departure from the system's logical intent.
- Surprising "sustained emergent performance degradation."
- TLA+ can be used to specify an upper bound on response time, a real-world property.
- However, real-world infrastructure does not support hard real-time scheduling guarantees.
- TLA is more expressive than Alloy.

2.3 Day 3 - Do comp-verify by hand for ToyMutex

Tasks:

-

Summary:

Article [How Amazon Web Services Uses Formal Methods] takeaways:

- Formal Methods scare new methods.
- Formal Methods - mathematically rigorous techniques for the specification, analysis, and verification of software and hardware systems.
- Safety - what the system is allowed to do.
- Liveness - what the system is must eventually do.
- Helps with communication and all engineers understanding a description of a design.
- Formal Specification is not good for bugs and operator errors that cause a departure from the system's logical intent.
- Surprising "sustained emergent performance degradation."
- TLA+ can be used to specify an upper bound on response time, a real-world property.
- However, real-world infrastructure does not support hard real-time scheduling guarantees.
- TLA is more expressive than Alloy.

2.4 Day 4 - CRA experimentation

Tasks:

- CRA by hand with new order: C1, C2, T2. Is this more efficient?
- Check your work with recomp-verify, use —verbose flag to check number of states
-

Algorithm 1 An algorithm with caption

Require: $n \geq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  $X \leftarrow x$
  $N \leftarrow n$
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow \frac{N}{2}$ ▷ This is a comment
    else if N is odd **then**
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

Summary:

Trying to run different bits of code. What I got from doing CRA by hand and checking with the machine:

- *By hand:*
  - From simply the $C_1||C_2$, there ended up being no π state meaning there could be short circuiting.
  - Since the parallel composition operator ($||$) is commutative, $C_1||C_2||T_2 \Leftrightarrow C_1||T_2||C_2$
- recom-verify checking:
  - There were 6 states modeled with $C_1||C_2||T_2$ and 8 states modeled with

Summary:

I was able to get LaTeXworking! I was able to set up the algorithm library as well which a little too long.

2.5 Day 5 - Coding

Tasks:

- CRA by hand with new order: C1, C2, T2. Is this more efficient?
- Check your work with recomp-verify, use —verbose flag to check number of states
-

Summary:

Trying to run different bits of code. What I got from doing CRA by hand and checking with the machine:

- *By hand:*
  - From simply the $C_1||C_2$, there ended up being no π state meaning there could be short circuiting.

  - Since the parallel composition operator (||) is commutative, $C_1||C_2||T_2 \Leftrightarrow C_1||T_2||C_2$

- recom-verify checking:
  - There were 6 states modeled with $C_1||C_2||T_2$ and 8 states modeled with

Summary:

I was able to get LaTeXworking! I was able to set up the algorithm library as well which a little too long.

3 Week 3: Initial Coding

3.1 Day 1 - Eduardo House Keeping

Solutions to LaTeX dependencies

```
sudo tlmgr install [package name]
```

3.2 Day 2 - Learning decomp basics

Concerns

- Shared import problems of both Will and Eduardo
  - import util.ToolIO;
  - import tlc2.tool.distributed.fp.DistributedFPSet;
  - import tlc2.tool.distributed.TLCWorker;
  - import org.osgi.framework.Bundle;
  - import org.osgi.framework.BundleContext;
  - import org.osgi.framework.FrameworkUtil;
  - import org.osgi.service.packageadmin.PackageAdmin;
- Eduardo specific issues
  - import tlc2.IDistributedFPSet;
  - import tlc2.ITLCWorker; (distributed.consumer)
  - import tlc2.IDistributedFPSet; (distributed.consumer)Z

Summary

Task for today was to try editing recomp-verify with custom. I am using the following constants:

```
Processes == {"p1","p2","p3","p4"}
Max == 3
```

1. *"control variable"* C1;C2;T2 → 13 states and ∼1.74 seconds
2. C1,C2,T2 → 20 states and 1.416/1.44 seconds
3. if I have C1, C2; T2 → 5 states and 1.45/1.5 seconds
4. if I have C1, T2; C2 → 46 states and 1.6 seconds
5. if I have C1; C2; T2 → 13 states and 1.58 seconds
6. if I have C1; T2; C2 → 20 states and 1.8 seconds

3.3 Day 3 - Meeting Eunsuk!

Stand up Meeting

What I've worked on/working on

1. Learned TLA+ basics
2. Decomposed a few examples (TwoPhaseCommit, ToyMutex)
3. Learned how to use recomp-verify
4. Read up on Ian's paper (understand relatively well)

What I'll be working on

1. Modifying the recomp-verify code to specifically the number of states each component works on
2. Actually being able to modify the code
3. Reviewing java stuffs

What's Blocking me

1. Java, Eclipse, and setting up jar files :(
2. Actually being able to modify the code

4 Week 4: Advanced Coding Techniques

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

5 Week 5: Data Analysis

6 Week 6: Mid-Internship Review

7 Week 7: Continued Research

8 Week 8: Preparing Results

9 Week 9: Discussion

10 Week 10:

11 REUSE seminars with Dr. Sunshine!

11.1 Week 1

11.2 Week 2 - Reading Research papers

Why read papers?

1. Get a breadth of the paper.
   (a) See solutions and problems of the field
   (b) Get an idea at high level from the Abstract and Introduction
2. The work is going to be useful in pointing you in the right direction
   (a) Author works a lot in a particular idea → they have a good sense of prior work and identify resources
3. Get ideas for reading papers
4. Good models for writing
5. Burrow ideas / methodologies

How to read papers

1. Make sure to read the abstract! You can get everything you need from the abstract (most of the time)
   (a) Sometimes papers are wrongly cited because their title was relevant even though they had nothing to do with their citation.

Know the difference between a defect, a fault, and a failure. I guess in general, SWE terms.

Conclusion

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.