5-2023

Markov Chain Analysis Comparison

Thomas Robert Prouty
Utah State University

MARKOV CHAIN ANALYSIS COMPARISON

by

Thomas Robert Prouty

Capstone submitted in partial fulfillment of the requirements for graduation with

University Honors

with a major in
Computer Engineering

in the Department of Electrical and Computer Engineering

Approved:

Capstone Mentor	**Departmental Honors Advisor**
Dr. Chris Winstead	Dr. Todd Moon

University Honors Program Executive Director
Dr. Kristine Miller

UTAH STATE UNIVERSITY
Logan, UT

Spring of 2023

ABSTRACT

This Honors Capstone was proposed as a method to develop a greater understanding of modern model-checking tools. To do this the student chose to analyze and compare the results of the tools Prism, Storm, and Stamina. To evaluate the effectiveness of each tool, comparisons were made of the results for running each tool on a simplified communication network.

The simplified communication network model used was a CTMC (Continuous-Time Markov Chain) model that employed correct signal transitions and erroneous bit flipping transitions. This was done in an effort to simulate possible errors and faults that can occur between a provider and a receiver. Tool effectiveness was measured by examining the probability and time that the model-checking tool took to verify the communication model's probability of entering a specific state. The communication model initially starts with all signals at zero, and each signal changes to facilitate a handshaking protocol.

In addition to the standard tools that were being tested, it was desired to additionally make a comparison against these tools using an altered form of SSA (Stochastic Simulation Algorithm) being designed in a proposed senior project. This other simultaneous project did in fact achieve and complete the implementation of this altered SSA. The senior project experienced issues in attempting to gain results that could be fairly compared to the other model-checking tools used in this capstone design. Therefore, the results of the altered SSA are not addressed in this report due to the lack of fair comparison.

There were multiple initial assumptions made in the capstone design regarding the communication model. The results of using the model-checking tools either confirmed or disproved these assumptions. Educated guesses were used to predict the most probable error states of the communication model as well as for the states with the lowest probability of entry. These error state predictions were disproved when other states were shown to have far lower entry probability than anticipated.

ACKNOWLEDGMENTS

I thank my undergraduate research mentor Professor, Dr. Chris Winstead, for the hours given to develop my understanding of stochastic computing concepts and connecting me to the FLUENT network within a nationwide university research community. I am indebted to these others for encouragement, ideas and comments, discussions, organizational directions and enthusiasm: Dr. Todd Moon who made time to fit into my hectic schedule, and Dave and Marty Prouty for supporting my goals and dreams of completing this and many other projects as well as a great education.

CONTENTS

LIST OF TABLES AND FIGURES

EXECUTIVE SUMMARY

Standard SSA (Stochastic Simulation Algorithm) works by taking a Markov Chain model and simulating sample paths. To generate a sample path, the application performing SSA will choose a state transition based on its rate after a random amount of time. Transitions with higher rates are more likely to be chosen over transitions with lower rates. In Markov Chain models, a transition rate is analogous to the probability of a transition being chosen. A set of chosen transitions is known as a sample path. SSA involves generating hundreds of sample paths to approximate the probability of a property being true or false for a given model.

The model checkers being explored used Monte-Carlo simulation to explore the state space of a model. SSA was then used to verify a property of the examined state space.

The alternate development, a Prism model-checking tool, is still underway and may be available for future comparison. For the purpose of completing the Capstone, comparing the three initial existing models shows interesting differences. One big difference to investigate was the difference in languages between Storm and Prism. Each uses a very different modeling language. Prism uses its own unique modeling language. Storm implements a language called JONI which is based on JSON.

INTRODUCTION

The goal of this Capstone project was to evaluate the effectiveness of multiple probabilistic model checkers and determine if there is a notable difference in the time that it takes each to perform SSA simulation. The base line model used for testing was a simplified communication CTMC model developed during the initial stages of this process. A major step in the project was converting that initial Prism model and its properties into the JONI model language, so that incoming model checkers (Storm and Stamina) could interpret the original model. After performing this conversion, each of the probabilistic model-checkers was run on the simplified communication model to determine how fast each could verify the model properties using SSA (Stochastic Simulation Algorithm).

Definitions

CTMC stands for Continuous-Time Markov Chain. It is a mathematical model that describes stochastic processes that evolve during iteration, with transitions between states occurring randomly in continuous time. CTMCs are used in various applications, including modeling biological systems, communication networks, and reliability analysis.

DTMC stands for Discrete-Time Markov Chain, a mathematical model used to describe a system that changes over time in a probabilistic manner. In the context of stochastic error prediction, DTMC can be used to model the behavior of a system that produces errors or failures with a certain probability at each time step.

SSA stands for Stochastic Simulation Algorithm. This is a method used to study probabilistic models by way of generating transition paths and then computing the final reachability of a model property based on the state space covered by the sample paths.

Motivation

Gaining experience in using multiple model-checking tools that are in development was the primary goal of this Honors Capstone. After several years spent being introduced to the concepts of investigating CTMC and DTMC models, I was interested in gaining an understanding of how properties in these models are verified.

My prior experience using the model-checker Prism left a gap to be filled and it was time to extend into using other tools. This Capstone was designed to investigate Storm and Stamina-Storm alongside the existing Prism and a possible new build of an alternate extension of Prism. If an alternative could be built as part of the Senior Design Project, a comparison of all four was desired.

BACKGROUND

This Honors Capstone Project is a complementary outcome of my Senior Project (Prism Importance Sampling Extension) which was to develop a new extension to Prism model-checker. The code would act as subsection of a Dynamically Weight Steady State Analysis function. The larger function uses commands from the Prism model-checker code as well as its own java code. The code developed in the senior project would augment the original DWSSA to allow for more effectiveness in model verification, in comparison to the use of standard Steady State Analysis.

Frequently, finding the probability of a property being confirmed for a model can be very difficult when the probability of reaching states that satisfy the property is extremely small. Therefore, the altered method of SSA seeks to allow SSA to generate improbable sample paths without permanently distorting the overall probability of events within the model. The altered SSA uses methods of importance sampling to artificially increase the transition rates of unlikely events in ways that can be reversed when the simulation has been completed. The use of importance sampling by the altered SSA should allow users to obtain the probability of remote events in the model faster than if standard SSA was implemented. The following diagram shows the intended flowchart of the altered SSA.

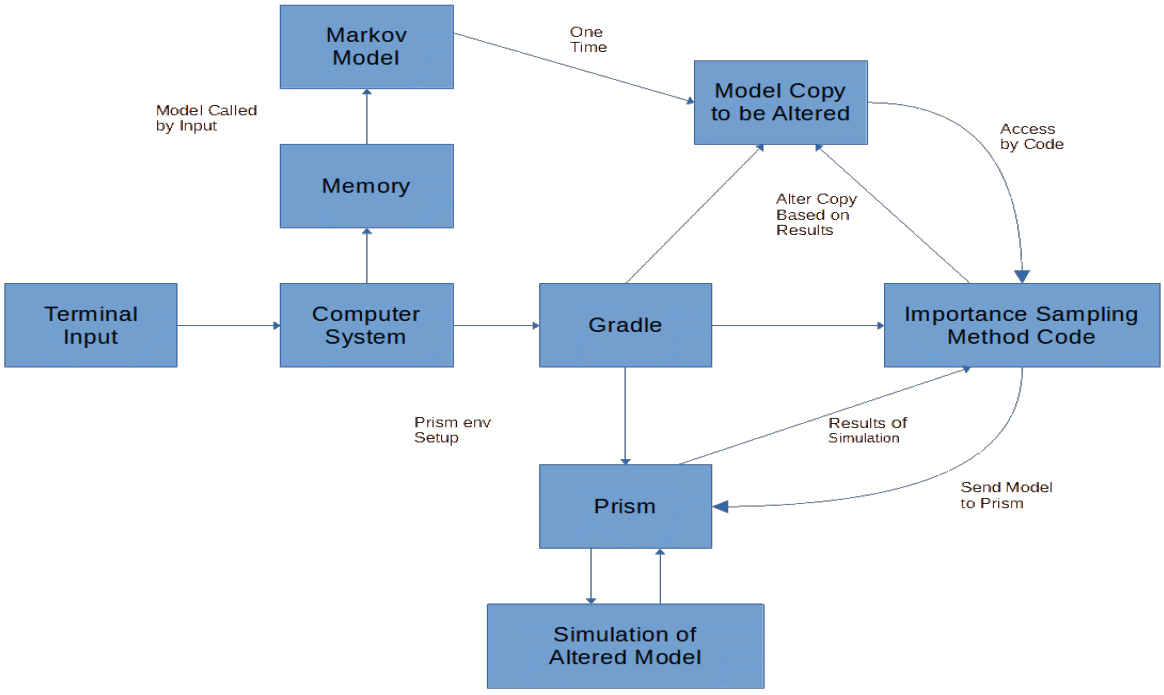

Figure 1, Flow Chart of Altered SSA

METHODS

For this honors capstone several different model-checking applications were used to discover the probability of reaching error states in a simplified communication system. The simplified communication system I examined was a handshaking algorithm operating between a device that produces data and a device that takes in and processes it. In addition to finding the probability of reaching predicted error states, work was undertaken to discover the probability of reaching other states. This would prove if there were misconceptions regarding the predicted error states of the system.

The model analyzed contained four modules, each with an associated signal that the module could alter depending on the states of other signals. The following image in Figure 2 contains a visualization of the communication model's state space.

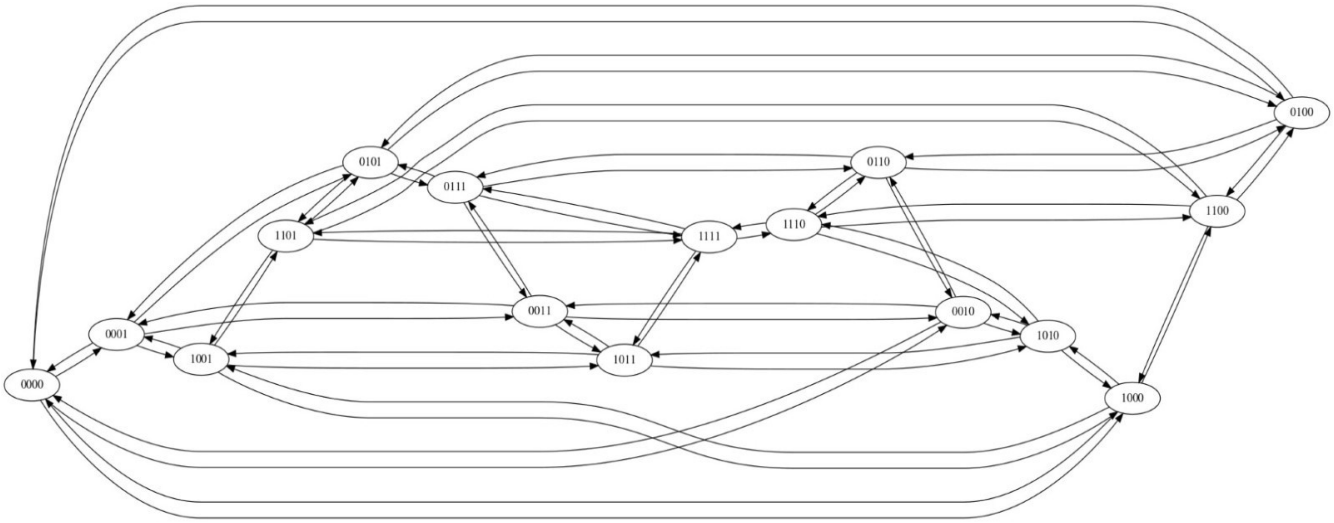

Figure 2, State Space visualization Using Dot

In Figure 2 Each state is assigned a binary value dependent on the conditions of the signals in the system. The signals are Booleans that are named as follows: done, ack, dry, and valid. Therefore, state "0101" is a condition where "valid" and "ack" are true while "done" and "ready" are false. Each of the four signals is associated with one of the four modules.

The names of the modules are producer, consumer, sender, and receiver. Each module is a simplified component of a communication network. All error prevention methods have been excluded from the model to simulate the probability of an error occurring without those methods. The producer controls the valid signal, representing whether the producer has created valid data to be sent to other devices. The sender module starts the handshaking procedure by informing the receiver module that valid data is ready to be transmitted.

It should be noted that since the model contains the possibility of bit-flipping errors, the model cannot

become deadlocked as there is always a possibility of a bit-flipping error pushing the system to a new state.

To examine this model, the model-checkers Prism, Storm, and Stamina were used. Each model checker was given the model file and then a set of properties that informed the model-checker about what it should find and report the total probability of entering each state in the simplified communication model. It should be noted that the model language that Prism uses is different from the model language used in Stamina and Storm. Therefore, as the original model and property files were written in Prism, both model files had to be altered and converted for Storm and Stamina.

RESULTS

To perform the comparison between the model-checkers a simplified communication model was analyzed, as discussed in the methods section. For verification, each property had a different set of true and false signals to test each state's reachability probability, as all state variables/signals for the model are defined as Boolean values. To find the probability of reaching a state from the initial state of "0000," the verification model used properties with the following format:

```
P=? [ true U [0,60000] ((!valid) & (!ack) & (!done) & (!rdy)) ];
```

Each model checker verifies a given property by performing SSA simulation to explore a model's state space. Once the model's state space is explored, the model checker states the result for the property it was checking based on the amount of the model's state space that was discovered. Tables 1 and 2 showcase the results of the property analysis and the verification time for each model-checker and property.

Table 1: Reachability Probability per State

States	Stamina Min	Stamina Max	Storm	Prism
0000	1	1	1	1
0001	1	1	1	0.999
0010	3.061E-08	3.061E-08	3.06E-08	3.06E-08
0011	1	1	1	0.999
0100	1	1	1	0.999
0101	1	1	1	0.999
0110	4.0183E-08	4.0183E-08	4.02E-08	4.02E-08
0111	1	1	1	0.999
1000	1	1	1	0.999
1001	1	1	1	0.999
1010	2.2541E-08	2.2541E-08	2.25E-08	2.25E-08
1011	1	1	1	0.999
1100	1	1	1	0.999
1101	1	1	1	0.999
1110	4.69E-09	4.69E-09	4.69E-09	4.69E-09
1111	1	1	1	0.999

Table 2: Table of Completion Times (in seconds) for each State and each Model Checker

States	Prism	Storm	Stamina
0000	0.005	0.004	0.017
1000	0.104	0.004	0.018
0010	2.00E-03	4.00E-03	1.80E-02
1001	0.097	0.004	0.017
0100	0.001	0.003	0.018
1100	0.097	0.004	0.017
0101	1.00E-03	4.00E-03	1.70E-02
1110	0.097	0.004	0.017
0010	0.002	0.004	0.017
1010	0.001	0.003	0.016
0011	1.00E-07	4.00E-03	1.70E-02
1011	0.001	0.004	0.017
0110	0.001	0.004	0.017
1110	0.001	0.004	0.017
0111	1.00E-03	4.00E-03	1.80E-02
1111	0.005	0.004	0.018

DISCUSSION

There are several reasons behind why the evaluation was done as a part of this project and the topics it covers are important. DTMCs and CTMCs are methods of system modeling that have been used for a large range of applications. Their most common use is in determining the failure rates or reachability rates of a system. In many cases attempting to find the probability of the properties, failure rates and others can be very time-consuming. So, even minor improvements regarding speed and efficiency can be very important for all types of models. In the case of this project the base line model used for testing contained very few states but was very transition and cycle dense in comparison to the number of states. Therefore, differences in the results of the Prism, Stamina, and Storm seen in the simplified model may be exacerbated in larger DTMC and CTMC models.

DTMCs can be used to predict the likelihood of errors or failures occurring in a system over time and can also be used to evaluate the effectiveness of different error correction or prevention strategies. For example, consider a system that has two states: "working" and "failed". At each time step, the system can either remain in the "working" state with a probability of 0.99, or transition to the "failed" state with a probability of 0.01. This behavior can be modeled using a DTMC, where the states represent the different states of the system, and the transition probabilities represent the likelihood of moving from one state to different state.

One of the applications of CTMCs is in stochastic error predictions. CTMC models can be used to analyze the behavior of error-prone systems over time, and to predict the likelihood of different types of errors occurring. For example, CTMC models have been used to analyze the behavior of communication networks and to predict the probability of packet loss, delay, or corruption.

As can be seen in Table 1, the states with the lowest reachability probabilities were the states: 0010, 0110, 1010, and 1110. These states having low reachability probabilities did not match my initial assumptions about the model. My initial assumptions were that the only states with low probability would be 1111 and 1010. In hindsight, this makes sense since if a system can process quickly then it is possible for all signals to be on at the same time. The state 1010 being an error state was anticipated as the state is not reachable by non-erroneous transitions.

It was very interesting to discover that 0010, 0110, 1110 were considered error states. This was because their low reachability probability means that they were not accessed by the normal processes of the system. Upon review 0010, 0110, and 1110 being error states makes some sense as they are all states in which the handshaking is being performed without valid data being in the system. Another interesting outcome of my analysis with the model was the difference between Stamina's and Storm's completion time. I believe that the difference is primarily in using the default values for Stamina, therefore it may be possible that if those default values were altered, that Stamina would outperform Storm.

It was an interesting result that each of the error states had a reachability probability of around 1E-08. It was initially thought that there would be either greater variance in the probabilities or that the properties would be approximately around the value of 1E-12. The magnitude of difference between the rate of error transitions and error state was unexpected and the method for implementing the model should be investigated further to see if it correctly replicated the desired behavior.

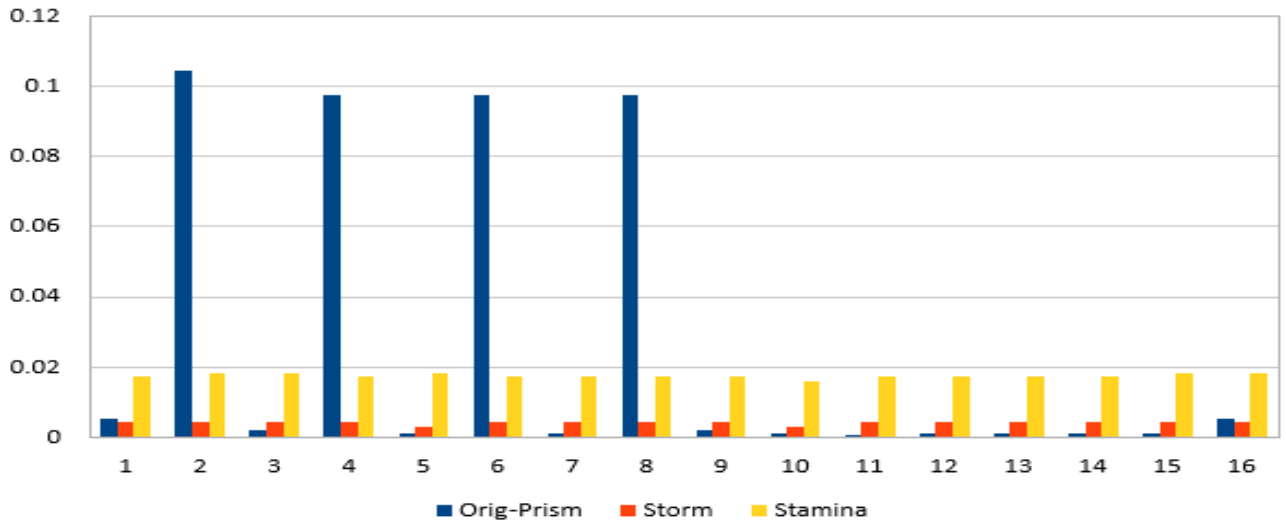

Figure 3, Comparison of models.

The results shown in Table 2 were very surprising. It was initially anticipated that Prism would perform equally as well as Storm and Stamina in the time spent verifying properties. However, it seems that the output times for Prism were far more variable than anticipated. Assessing "time to verify"; Storm was consistently the best with a somewhat stable verification time. The verification time results for Stamina were slightly longer but still within a comparable scope. These times were anticipated since Stamina is built on the Storm framework.

Finally, the verification times for Prism seemed to vary significantly, depending on the state being examined. It should be noted that the results from Prism seemed to show that the model-checker sped up in its verification speed as more properties were checked. This may be because as the model was explored more, Prism improved in its ability to determine the final probability of designated properties.

Issues

Throughout this project there were a few issues to overcome. For the most part this was a case of "not knowing what you don't know". Problems were expected, there just was no way to know what they would be until they occurred. The most significant and time-consuming matter was the difficulty with the model being read in Storm. Originally, the system was modeled in a Prism file. This method of modeling caused problems when Storm was trying to parse the model and its properties. Therefore, to fix the errors caused by parsing, it was necessary to alter the format of the model and properties to match what Storm could parse.

While attempting to use Stamina, another situation became evident. It seemed that Stamina had issues with striving to read more than one property in a file at a time. To fix this issue, the file for "storm_properties.csl" was split into 16 different "stamina_properties.csl" files. This problem of separation and split files resulted in the determination that performing verification on several models

would take substantially more time than originally designed. Therefore, the project migrated to a process of examining the singular test model in detail.

A new issue was encountered while attempting to run altered SSA that was built as part of the senior design project. The code for the altered SSA was added to an already functional Dynamic Binary Weight version of SSA (dbwSSA). After implementation it became apparent that attempting to run the dbwSSA was substantially more difficult and time-intensive than forecasted. The command line to run the altered SSA required a multitude of extra arguments that caused errors. These problems were solved when it was determined that the language used for the command line options had to be very specifically formatted for correct execution. In addition, it was discovered that determining the correct time-bound for a given model is extremely important with the altered SSA, as if the bound is too small that important samples are rejected by the algorithm.

CONCLUSION

Comparison of the three model-checkers used in the project showed that Prism execution time was far more variable than originally anticipated. The execution times for Storm and Stamina were significantly comparable to each other and very consistent. This lends credence to the idea that the JONI modeling language may result in a more consistent time for execution in comparison to the Prism modeling language. All three model-checker tools functioned efficiently to determine a correct entry probability for each state in the simplified communication model.

For the nodes examined, Prism performed 62 percent better than Storm or Stamina. However, when attempting to determine the entry probability of error states, Prism significantly struggled. The most likely cause of this difference is that Storm and Stamina have built-in methods to reduce the effect of low probability transitions resulting in bit-flipping errors. The extra built-in methods of Storm and Stamina most likely allow the two JONI languages to infer or discover error states far more consistently than Prism. It should be noted that Prism may perform better than Storm or Stamina when exploring larger non-cyclic state spaces, but such a comparison will need to be performed.

The alternative SSA was successfully implemented in late-stage testing. More time will allow further comparison and results can be investigated in relation to the three model-checkers used for the Capstone investigation.

Word Count

Report – 2,691

HONORS REFLECTION

During my honors capstone experience, I learned a lot about project management and technical development. One of the most important lessons I gained was the value of effective scheduling. In hindsight, I realized that I could have improved my scheduling by breaking down larger tasks into

smaller, more manageable steps. By doing so, I could have given myself more flexibility to adjust my schedule when other academic commitments arose. I also learned that it is important to prioritize tasks based on their level of importance and urgency. While some tasks for my senior project and honors capstone may have been seen as lower priority in comparison to immediate academic assignments, I could have benefited from reevaluating their relative importance and setting aside dedicated time for them in my schedule.

Since my Senior Project and thus the Capstone were abruptly changed during my second-to-last semester (when I was advised that the topic was not close enough to my subject area), two semesters of work were lost. Having to rebuild both projects, their proposals, outlines, timelines, and approvals put my seriously behind on all levels. At the same time, this took me from working with a team to working on my own. This had benefits and difficulties as well. Without a team, the work was all down to just me and learning to find what resources were available. There was no way to spread out any of the task development or scheduling. But this also meant that for many aspects, I was not dependent on others for timing. Timing was my biggest challenge and my greatest freedom.

In addition to these project management lessons, I also gained valuable technical development experience during my honors capstone. One area of particular interest to me was the evaluation and development of modern computing techniques and error checking tools. As part of my project, I worked on developing a tool that could be used to analyze large data sets. Through this work, I gained a deeper understanding of the complexities involved in tool and model development.

One of the biggest challenges I faced during my tool development work was testing. I quickly learned that testing was a time-consuming and often frustrating process. There were times when I spent days trying to verify the results produced by my program, only to find out that there was an error or that I had misinterpreted the mathematics. This experience taught me the importance of being patient and thorough in testing, as well as the value of seeking feedback from others to ensure accuracy.

Another challenge I worked to overcome during my tool development work was the difficulty of investigating tools that were still in development by a global community. This required me to be adaptable and open to making changes on the fly as I learned more about the tools' capabilities and limitations. While this could be frustrating at times, it was also rewarding to see my work progress and improve as I gained more experience.

Overall, my greatest personal development was in understanding how I need to develop skills and knowledge around scheduling projects and managing multiple demands in the same timeframe. Trying to do upper-level courses while at the same time developing a Senior Project, Capstone Project and Undergraduate Research required a complexity that I just did not have the experience to manage well.

I saw that the development of my timeline and project management for the Senior Project had quite a bit of detail as it was initially built, but with hindsight, I can see that it really needed several sub-task areas with timing that gave lot more attention to the amount of time I would have to wait for others to respond to my questions, or how much more time it would take to learn or investigate many of the development steps. As my Senior project changed, the Capstone got further and further behind since it was fully dependent on the development results of the model checkers being built in the Senior Project.

As a result, I often found myself having to delay some tasks to attend to other priorities. Compressed timelines and became difficult to manage and adjust.

Here is an example of the Senior Project timeline that shows a fair bit of planning detail. Comparing it with the simplistic timeline for the Capstone shows the need to develop greater sub-tasking.

TASK	ASSIGNED TO	PROGRESS	START	END
Determine & Setup Tools				
Research and select programming tool	Tom	100%	1/8/23	1/15/23
Obtain Tools: Program, Drive Space,		100%	1/13/23	1/15/23
Setup workspace on required computers		100%	1/15/23	1/19/23
Identify access programs and how to connect		100%	1/19/23	1/30/23
Outline Programming Steps				
Identify progression and project needs		100%	1/23/23	1/27/23
Identify programming-code issues		100%	1/25/23	1/30/23
Build and Evaluate				
Build Extension of Prism to simulate alternate methods of Markov Chain Model assessments		100%	2/1/23	3/1/23
Run simulation to identify issues		100%	3/2/23	3/15/23
Document issues and identify		100%	3/2/23	3/30/23
Re-calibrate Extension based on results		95%	3/31/23	5/1/23
Final Extension build complete		90%	4/4/23	5/1/23

Final Reporting and Demonstration			
Assemble documentation	100%	4/4/23	4/30/23
Draft Report	100%	4/15/23	4/30/23
Final Report compiled and submitted	70%	5/1/23	5/3/23
Presentation of Project	20%	5/3/23	5/3/23

Figure 4, Senior Project Timeline

TASK	WITH	PROGRESS	START	END
Determine & Setup Tools				
Research and select assessment tools: Find version of PRISM that will be used for project. Also decide on Java version to use for senior design project. Finally, find and select stable versions of Storm and Stamina that can be used for comparison to senior design project's methods.	Tom	100%	1/8/23	1/15/23
Obtain Tools: Determine the computer that project will be completed on. Decide on maximum memory usage of the senior design project's final code.	Tom	100%	1/13/23	1/18/23
Setup workspace on required computers: Download selected versions of PRISM, Storm, and Stamina. Update or download the version of java that will be used with senior design project.	Tom	70%	1/18/23	1/20/23
Identify Markov Chain Models: Find and document Markov Chain models and properties. Models and properties should allow for clear comparisons in the results of the Model Checkers.	Tom	0%	1/20/23	1/30/23
Capstone and Senior Design Project Steps & Tasks				
Senior Project Development: Develop and Create software that will be utilizing PRISM to implement	Tom	10%	1/30/23	3/31/23

Scaffolding and Modulo importance sampling methods.				
Identify Analysis issues: Document possible issues that could arise while trying to make comparisons between the different analysis methods, and model checkers. For example, do all model checkers track time in similar ways?	Tom	70%	1/29/23	2/7/23
Comparison Outline: After identifying potential issues, make a list of possible ways to compare the model checkers. Then, make a list of ways to compare the methods the model checkers implement to the methods implemented in the senior design project.	Tom	0%	2/8/23	2/13/23
Begin Draft of Final Report: Create draft of final report so that the results of the comparisons and analyses can be inserted as soon as they are completed.	Tom	0%	2/16/23	2/21/23
Assess and Evaluate				
Begin Documentation of Markov Chain Model Assessments: Document results of simulating test Markov Chain model with base PRISM, Storm, and Stamina. These three model checkers primarily use standard SSA for find answers to posed properties. Therefore, a comparison between the three makes for a good benchmark to compare the methods from the Senior Design Project to.	Tom	0%	2/22/23	3/1/23
Document Simulation Issues: Document issues that are found while attempting to simulate the model with PRISM, Storm, and Stamina.	Tom	0%	2/23/23	3/2/23
Analyze Progress and Viability: Take a step back to readjust and evaluate current progress. Alter and update Work Plan based on current progress and current goals.	Tom	0%	3/2/23	3/4/23
Perform and Document results of Method Comparison: Simulate test Markov Chain Model with Senior Design Project methods. Document results from simulation and any issues that were encountered.	Tom	0%	3/2/23	3/27/23

Begin Report of results: Compare and document the differences between the Model Checkers. Also, compare and document the differences between the analyzed methods.	Tom	0%	3/28/23	4/1/23
Draft of final report/presentation: Create Draft of Final Report. Design and make presentation regarding capstone and Senior Design Project Findings.	Tom	0%	4/4/23	4/8/23
Final report/presentation - run thru: Perform trial run thru of presentation and have peers evaluate current report draft.	Tom	0%	4/4/23	4/10/23
Final Reporting and Demonstration				
Assemble documentation: Compile all documents created by or used in the completion of the Capstone Project.	Tom	0%	4/4/23	4/15/23
Final Edit Report: Meet with Advisor and Project Mentor to get a final review of submission report.	Tom	0%	4/15/23	4/20/23
Final Report complied and Submitted: Submit final report for project.	Tom	0%	4/25/23	4/25/23
Presentation of Project: Present Project to interested parties.	Tom	0%	4/16/23	4/22/23
Project Reviews				
Work Plan Review: Meet with DHA to review work plan.	Dr Moon	0%	1/20/23	1/23/23
Final Proposal Review: Meet with DHA to review final draft of Capstone Proposal.	Dr Moon	0%	1/23/23	1/25/23
Final Design Review: Meet with DHA to discuss goal prioritization as well as any aspects of the Capstone project that should be modified.	Dr Moon	0%	2/2/23	2/3/23
Progress Report #1 - Advisor: Provide update on current progress. Discuss any issues with current progress I have had. Explain any ideas I have developed regarding how to make fair comparisons and get feed back on ideas.	Dr Moon	0%	2/17/23	2/18/23

Progress Report #2 - Advisor: Discuss current Progress regarding comparisons between Model Checkers. Explain any issues I have come across. Get advice regarding how to compare the Markov Chain analysis methods fairly.	Dr Moon	0%	3/3/23	3/5/23
Progress Report #3 - Advisor: Report current progress regarding the final report. Explain any issues with comparing methods that I have come across.	Dr Moon	0%	3/17/23	3/22/23
Progress Report #4 - Advisor: Discuss current progress regarding Final Report as well as presentation.	Dr Moon	0%	4/7/23	4/10/23
Draft Capstone Review: Submit final report to Dr Moon for final review and make any edits that are suggested.	Dr Moon	0%	4/27/2023	4/28/23
Final Capstone Review & Approval: Meet with all interested parties and provide report as well as perform presentation, as required.	Dr Moon, et al	0%	4/25/23	5/1/23
Progress Report #1 - Project Prof: Inform Project Menor about current progress in Senior Design project. Discuss possible ways to fairly compare methods being analyzed.	Dr Winstead	0%	2/17/23	2/18/23
Progress Report #2 - Project Prof: Inform Project Menor about current progress. Discuss with Project Mentor regarding results of Model Checker comparisons.	Dr Winstead	0%	3/3/23	3/5/23
Progress Report #3 - Project Prof: Inform Project Menor about current progress. Discuss current results of methods implemented in Senior Design Project.	Dr Winstead	0%	3/17/23	3/22/23

Figure 5, Capstone Timeline

When I compare the Senior Project Design to the Capstone timeline seen here, it is apparent that I was not managing the tasks as deeply as needed. While I had plenty of detail it wasn't broken down well into subtasks. But, more importantly the sliding delays of the Senior Project had the most impact on my management ability.

Training in time management as part of the process, way back in the Junior year would have helped. But I acknowledge that the change of project took the most tole in getting to the finish line on time.

Upon this reflection, I was led to the important realization that I could have planned my academic classes with better regard to my senior and capstone project commitments. Although I had a general idea of what tasks I needed to complete for my project at the start of the semester, as the semester progressed, I found that my long-term planning for my project was not as robust as it should have been. As a result, I struggled to effectively prioritize and complete the tasks necessary for effective progress on my projects. If I had aligned my academic coursework with my project timeline, I would have been able to better allocate my time and stay on track with my project milestones. Additionally, I realized late in the process that having lost the prior two semesters of work to the former team meant it would be a significant struggle to catch up.

My greatest suggestion to future Honors Capstone students is to get really great feedback very early in the process and make sure that the proposed project has a team that can stay together for two years. Many students dropped out of school during the pandemic or went in different directions with their studies. This meant that in the end, I was on my own to do both the Senior Project and Capstone by myself with no support for time management. Thankfully, the Engineering Department as a No-fail guarantee for Senior Design Projects. This meant that I could move to a project that was not likely to get a great result in a short timeframe but would be a topic that I found very interesting and led me to a Capstone Project with very interesting results and great information to build on for future Stochastic investigation.

Word Count

Reflection - 1,120

BIBLIOGRAPHY

Parts of this report are pending publication in Digital Commons; the Methods section contains content excerpts and data from the paper listed below. This source document includes nothing about the outcome of work done in collaboration.

Publications

Non-refereed papers
Thomas R Prouty. Model Checking Software: Project Report. Cyber-Physical Systems. Department of Electrical and Computer Engineering, Utah State University.

1. Baldi, P., Piccioni, M. Importance Sampling for Continuous Time Markov Chains and Applications to Fluid Models*. *Methodology and Computing in Applied Probability* **1**, 375–390 (1999). https://doi.org/10.1023/A:1010050800089.
2. Elvira, Víctor, et al. "Improving Population Monte Carlo: Alternative Weighting and Resampling Schemes." *ArXiv:1607.02758 [Stat]*, July 2016, http://arxiv.org/abs/1607.02758.
    2.1. Description: This paper discusses the problems with current PMC schemes, Population Monte Carlo. PMC methods are ways of approximating unknown probabilities given a probability distribution and an importance sampling scheme. The article also includes three new importance sampling schemes that could potentially reduce the problems found in current PMC methods.
3. Colin S. Gillespie and Andrew Golightly, "Guided proposals for efficient weighted stochastic simulation", J. Chem. Phys. 150, 224103 (2019) https://doi.org/10.1063/1.5090979.
    3.1. Description: This journal discusses the possible use of importance sampling in speeding up biochemical reaction simulation. Biochemical reaction networks often contain very rare events that are extremely difficult to simulate, and the authors of the article believe that importance sampling can reduce the time needed to simulate these rare events. The document contains comparisons between the method the authors devised and other current importance sampling methods.
4. Martino, Luca, et al. "An Adaptive Population Importance Sampler: Learning From Uncertainty." *IEEE Transactions on Signal Processing*, vol. 63, no. 16, Aug. 2015, pp. 4422–37, https://doi.org/10.1109/TSP.2015.2440215.
    4.1. Description: The document discusses the possible benefits of an importance sampling method known as APIS, adaptive population importance sampling. One of the important topics surrounding APIS is that is parallelizable and can be used to more quickly simulate appropriate models. Finally, the article discusses the results of APIS when it was applied to five sample problems.
5. Shahabuddin, Perwez. "Importance Sampling for the Simulation of Highly Reliable Markovian Systems." *Management Science*, vol. 40, no. 3, Mar. 1994, pp. 333–52, https://doi.org/10.1287/mnsc.40.3.333.
6. K. Chatterjee, L. de Alfaro, and T. A. Henzinger, "Stochastic real-time systems," in Proceedings of the IEEE, vol. 91, no. 1, pp. 34-55, Jan. 2003. doi: 10.1109/JPROC.2002.805826
7. R. L. Peterson, "Stochastic Petri nets: an elementary introduction," in Computer, vol. 21, no. 3, pp. 25-33, Mar. 1988. doi: 10.1109/2.15

8. E. P. K. Tsang, "A survey of Markov chain models for reliability analysis," in Reliability Engineering & System Safety, vol. 91, no. 10-11, pp. 1295-1310, Oct-Nov. 2006. doi: 10.1016/j.ress.2005.11.020

AUTHOR'S BIOGRAPHY

Thomas R. Prouty is a student and aspiring computer engineer with a passion for research in the field of AI and stochastic concepts. He has completed his BS degree in Electrical-Computer Engineering from Utah State University in Logan, UT. His research interests include nanotechnology, synthetic biology, and CTMC Modeling.

Thomas has been actively involved in research during his four undergraduate years and has served as an Undergraduate Student/Research Assistant in the Department of Engineering at USU since 2021. In this role, he has worked with the nationwide university FLUENT Team. He has also used GitHub and MATLAB to engage directly with researchers, student teams, and professors to develop code skills in C++, Java, and PRISM.

In addition to his research work, Thomas has also volunteered at St. Alphonsus Hospital and the Ronald McDonald House, where he has demonstrated strong organizational skills and a passion for helping others. He is a member of the USU Honors Program and the USU Undergraduate Research Fellowship.

Outside of his academic and research work, Thomas enjoys cross-country hiking, reading, scientific investigation, and music. He is also an avid traveler, having visited many US states, Canada, Eastern Europe, and Australia.