CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: MidGARD (Model-initiated di-Graph Analysis for Regression Deployment)

SANDISK™

Chris Ortiz, Senior Technologist
Tools & Infrastructure
SanDisk Corp., Engineering & Product Management
chris.ortiz@sandisk.com

Rex Jackson, Vice President
Tools & Infrastructure
SanDisk Corp., Engineering & Product Management
rex.jackson@sandisk.com

Project Overview:

In SanDisk, we are serious about the Quality of our popular storage products, and we always strive to be best-in-class in upholding our global SanDisk brand. SanDisk has launched an initiative to write our designs using TLA+ (Temporal Logic of Actions) to enable formal verification through a model checker. TLA+ is a mathematical language model based on set theory and logic use in writing formal specifications. We plan to convert a few of NVMe and PCIe's specification features into TLA+ as our **model to initiate** a **directed graph analysis** that will be intended for test **regression deployment**.

The model-checker that comes with the TLA+ tool ecosystem is called TLC, which has an option to generate a Graphviz DOT file compatible directed graphs (di-graphs) of the state predicates (as nodes) and actions (as edges). We intend to only generate such directed-graph once the TLA+ spec passes the TLC model-checker. Though the TLA+ spec may pass the model-checker, it is possible the DOT file may not be able to be generated due to huge state-space size that the TLC DOT dump option may not be able to handle such complexity. Since the pure formal verification of the TLA+ spec has already passed, further TLC options may be used such as VIEW and ALIAS to abstract further the DOT file so it can be generated (i.e. lessen the details which tends to decrease the number of nodes, therefore smaller state-space size).

Understanding TLA+ concept and Graphviz DOT file structure is important in this project. All TLA+ specifications are stuttering invariant. This means the directed-graph DOT file generated may have self-referenced edges going back to the same state predicate node. Knowing this we may have an external script written in Rust to remove such edges for those edges that we may not need in our graph analysis.

The directed-graph we able to generate and cleaned up may be convoluted with lots of paths forming loops that we have to ask ourselves if we are going to have a set of list of sequences of steps/paths what would it be so that we have a best candidate set that we can cover a good representation of all the nodes in that directed-graph. Similarly, like Traveling Salesman problem. This will be the basis of our graph analysis.

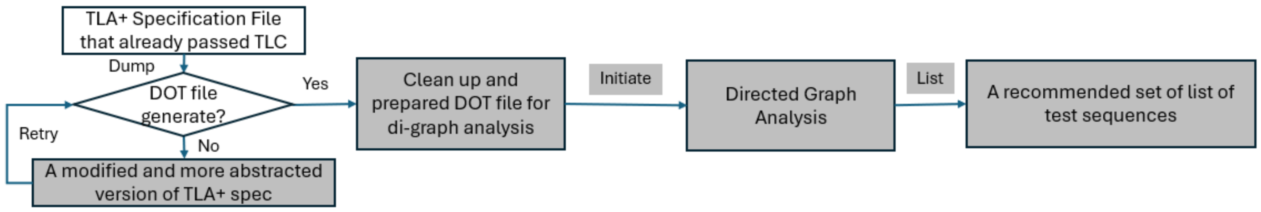
Directed graph analysis such as isomorphism, tournament, Hamiltonian, Tarjan, etc. may be utilized. **The goal** is to have a robust Rust program that allows TLA+ specification model to initiate the TLC model-checking and if it passes it may generate the directed-graph DOT file, clean and prepare it up, then initiate directed-graph analysis which provides us a set of lists of sequences representing very good coverage. Each of these sequences of actions (edges) will be our test sequences which will guide SanDisk to plan out regression coverage which we should be deploying in our validation process. The core skills required for this project are the creative application of solutions to analyze directed graph with an objective in meeting the goal stated above.

The raw DOT directed graph that comes up from TLA+ specification provides infinite test sequences possibilities, but we cannot forecast and even deploy such scheme in our validation process. MidGARD will give us finite set of lists that are scientifically best calculated coverage that we will have greater confidence with, than if the list comes from the limited intuition of a test engineer which can miss corner cases or cross features.

This breakthrough will benefit us in three important ways:

1. **Provide us with a finite number of test sequences we can forecast for our regression deployment** which is part of our optimized product life cycle and resource planning.
2. **It will encourage test engineers** to adopt TLA+ in scientific analysis of requirements to fully understand specifications in precise logically way and for which the task of brainstorming and administratively composing test sequences becomes free and automatic.
3. **More confidence in the quality of the set of test sequences** as they are mathematically analyzed from all possibilities generated from TLA+ specification that is very complex to be analyzed by test engineers to uncover corner cases, find cross features or interesting test scenarios.

The figure below is our proposed solution where the grayed boxes and arrow's label are the ones we need NAU to help. This project will expose the NAU students to be familiar with TLA+ which Leslie Lamport invented. He is an ACM Turing Awardee where his contribution to Computer Science is used in universities, and by big companies like AWS, Microsoft, Oracle, Google, LinkedIn, MongoDB, Datadog, Intel, ARM, Nvidia, etc. Also, there is also potential that projects such as this can get granted from TLA+ Foundation if submitted to and get selected by TLA+ committee.


TLA+ Specification File that already passed TLC
Dump
DOT file generate?
Yes
No
Retry
A modified and more abstracted version of TLA+ spec
Clean up and prepared DOT file for di-graph analysis
Initiate
Directed Graph Analysis
List
A recommended set of list of test sequences

Knowledge, skills, and expertise required for this project:

- Knowledge of TLA+ using TLA+ VSCode extension.
  - Home Page: https://lamport.azurewebsites.net/tla/tla.html
  - Github: https://github.com/tlaplus
  - Conferences: https://conf.tlapl.us/home/
  - Foundation: https://foundation.tlapl.us/
- Coding skill in Rust (preferred) or Python: https://www.rust-lang.org/

Equipment Requirements:

Computer system with VSCode installed with the following extensions and internet connectivity.

- Rust-analyzer (The Rust Programming Language)
- TLA+ (Temporal Logic of Actions by TLA+ Foundation)
- Graphviz Interactive Preview (tintinweb)
- Live Share (Microsoft)

Other software:
- OpenJDK >= 11.0.6 (Please do not use Oracle's)
- Any Rust cargo extensions needed especially for directed-graph analysis

Software and other Deliverables:

Detailed, clear, and professionally composed documentation and literature which SanDisk product teams will use as reference manual.

Complete documentation and well-commented codebase in **Private** Github account, for which SanDisk can solely decide freely if it is beneficial for SanDisk to open-source this repository (or fork of it) or not in the future.