DCC831 Formal Methods
2025.2

Final Project

The final project is your opportunity to apply what you've learned in this course. You will pick a topic and develop a formal theory about it in the **Alloy** language: you'll define the objects/programs, specify their behavior and requirements, and prove that the specification is sound, i.e., that the behavior respects the requirements.

This topic can be an algorithm, some math, or some software application. The final result won't necessarily be a lot of code — sometimes the hardest part is the design rather than the verification. A good project will go beyond the mini projects we did in this course, like the MailApp system of the Alloy mini project. Below there will also be some examples of topics, but you should check the example sample models in the Alloy Analyzer, which can offer some inspiration as well. You must consult with Haniel before starting your project, to make sure that the topic is sufficient, manageable, and in scope. Be mindful of the deadlines below. Failing to come up with a topic will result in failing the project.

To do this project, the 'utils' that ship with the Alloy Analyzer may be useful. Make sure to check them.

Points will be removed from models where the transitions are not labeled and it is not straightforward to determine which operation was applied between state transitions.

The project will be graded in the following way:

- 5 points for the initial proposal
- 10 points for the presentation
- 15 points for the report
- 20 points for the Alloy model

Dates and deliverables

The project has four deadlines. All deadlines are end-of-day, i.e., 11:59pm.

- **Sunday, Nov 02**: Definition of the topic, together with a high-level description of the topic and of what is to be specified and verified. This should all be in one PDF file of up to 2 pages.

  This should be communicated to Haniel via e-mail, and some discussion may ensue to adjust expectations. The topic must *necessarily* be consolidated by Nov 10.

  The finalized proposal must contain:

  - Which properties the model must have and that will be tested, be it via validating assertions or by exhibiting instances illustrating a given functionality.

- **Nov 24 to Dec 01**: Presentations in class of the work-in-progress project. Each presentation will be 25min total (reserve time for questions). At this point all the main elements of the project must be defined and in a substitution state of specification and validation.

- **Sunday, Dec 07**: Detailed description of the topic, behavior, requirements, and the Alloy specification (both the model and the assertions).

  This description should be in a PDF file of up to 8 pages. The Alloy specification must be in a single file, and be such that it conforms to the described behavior, and the stated assertions must hold.

  The report must contain a running example to illustrate how an instance of the problem being modeled would be represented in the Alloy model and how the Alloy model could be run to visualize the expected behavior.

  The running example must be included in the Alloy specification in a way that allows easily running it.

  The report must specify whether the proposed defined assertions or functionality-exhibitions are achieved, and clarify why yes or why not.

  When defining assertions, special care must be taken to not have them hold vacuously, i.e., they hold because the behavior they are guarding against is not feasible. Justifications (via run commands for the expected behavior) must be provided to show that they do not hold vacuously.

  The introduction of the report must contextualize the *challenges* for defining an Alloy specification to the proposed topic. And which design decisions you took to address these challenges.

  The report should precisely describe the initial state and the transitions that the system can take in Alloy.

Project suggestions

Below are some general suggestions. You are free to follow other directions, however.

Data structures and algorithms

Pick a familiar data type and associated operations, model it in Alloy, and verify some properties of these operations.

There are some good ideas in chapters 8 and 9 of the textbook *Certified Programming with Dependent Types*[1] by Adam Chlipala: red-black trees, heterogeneous lists, and others. Non-dependent data types are okay too: you could even implement a few sorting algorithms and compare them. Graphs and graph algorithms can be good projects, e.g., Tarjan's algorithm.

Or: pick some "competitive programming" problems, solve them, and prove that your solutions are right. This statement of correctness could have many forms — it depends heavily on what the problem is. See the USACO problems for one source of inspiration.

[1] `http://adam.chlipala.net/cpdt/cpdt.pdf`

Program semantics

Extend a simple language such as the **WHILE** language with extra features. For example, give it static arrays, or function definitions, or multiple datatypes, or pointers/references. Sections 2.4 and 2.5 of Nielsen et al.[2] have good ideas. Then adjust the semantics to one of your choice.

[2] `https://www.cs.ru.nl/~herman/onderwijs/semantics2019/wiley.pdf`