The Problem

Discrete math is a foundational topic for Computer Science. Undergraduate computer science students enroll in a course at some point in their college career focusing on topics in discrete math including set theory, relational algebra, and predicate logic. These courses are extremely important for students as they form the basis of analysis in the field. Analysis of so-called *cyber-physical* systems, where computer and physical systems interact, is especially important, as shown by public issues like the recent Boeing 737 Max; if Boeing engineers had employed more discrete analytical thinking, they may have avoided the issues Boeing is facing today.

CS students typically don't get the kind of interactive practice with math that they do with programming in languages like Java. As a result, misconceptions can persist, and students can develop an attitude that discrete math has no applicability to their field [2]. Progressive CS departments have included interactive tools in their curriculum to assist students with the learning of such topics. Here at Michigan Tech, instructors in the College of Computing have created a curriculum involving the use of the *Alloy* modeling tool. Alloy consists of two parts: (1) a programming language based on the mathematical languages of relational algebra and predicate logic, and (2) an application for finding *instances* (situations) that follow (or break) the requirements set by an Alloy program. Alloy is a powerful tool for modeling real world systems. Giving students an opportunity to use Alloy provides them an authentic sense of math in their field.

Alloy provides more feedback to students than traditional pencil-and-paper exercises; however the experience could be better for first-time students. When students work with Alloy, it can be daunting because while they are trying to develop models they may become burdened with small syntactical issues they may not be familiar with as a beginner. Adding to this, the base error messages provided by Alloy can be confusing because they are tailored towards experts and not first time users.

The Direction

Alloy is an industrial-strength tool with a growing community of users. At the center of Alloy is a tool that provides a highly expressive language that can be used to model real-world scenarios via quantifiers of predicate logic and operators from relational calculus [2]. It is based on the concept of "lightweight modeling" - developing a mathematical model that can quickly and easily uncover hidden assumptions or other design flaws early in the process. Its language is based on the mathematical languages that introductory discrete math students are learning. Alloy allows students to learn these languages in the context of an interactive tool with real applications.

Scaffolding can be thought of as creating a base structure in which students can focus their knowledge on the task at hand while not overloading them with unknown or distracting content. We will implement automated scaffolding for the student newcomer to Alloy. There has been some initial effort at developing and using scaffolded exercises that introduce students gently to Alloy, and these initial applications have been fairly effective [2]. This project will extend the scaffolding with automated support, in two ways. (1) Providing a graphical block-based interface for the language, similar to introductory programming languages like *Scratch* and *Snap!* This has the potential to keep students focused on learning fundamental concepts of logic and relations, without distracting syntax problems. (2) Developing automated detection and response for common problems in newcomer code. This follows from successful work in automated critique of Java programs using the *WebTA* tool, developed at Michigan Tech [3].

The Plan

In *Phase I*, we will (1) implement a block-based interface for Alloy's syntax, using *Blockly*. Blockly is a programming library that assists in creating a visual interface where users can drag blocks like puzzle pieces together to develop a codebase. Blockly is often used in educational programs to teach scripting languages like JavaScript, Python, and many more. We will create custom Blockly blocks to capture the syntax of Alloy. (2) We will also develop a categorization of common newcomer mistakes, based on anonymized student submissions.

In *Phase II*, we will (3) integrate the block-based interface into the Alloy Analyzer engine to give a seamless user experience; (4) develop automatic detection of student errors, using the WebTA architecture. By the end of the summer 2020, (1) and (2) will be complete, and progress will be made on (3) and (4). In the academic year 2020-2021, we will pilot the use of the prototype tools with student volunteers, and submit materials for publication at SIGCSE and/or ITiCSE conferences. Feedback from the pilot studies will inform further development in summer 2021 and use in the Discrete Structures course at Michigan Tech in fall 2021.

My Motivation

My personal experiences have provided me with the expertise and motivation for a project like this. After graduating high school I worked for an up-start called Ampel Feedback. It was then that I learned Electron, ReactJS, Typescript and many other technologies I aim to use for this research program. For my research, I would like to develop a simple, and easy to use graphical application for undergraduate students and users of Alloy globally to develop simulations without the need to fully understand the underlying programming language involved.

When I came to Michigan Tech and took discrete structures as a second-year student, the interactive element of the course with Alloy greatly benefited me in learning the course material, but it was also the most difficult part of the course for me. I knew what I wanted to do, but I could not replicate the thoughts in my head into the code required to run simulations. I want to be able to help other students with their learning of discrete math and aim to remove some of the unnecessary confusion that may develop.

References

[1] Jackson, Daniel. *Software Abstractions: Logic, Language, and Analysis.* MIT Press, 2016.

[2] Brown, Laura; Feltz, Adam; Wallace, Charles. Lab Exercises for a Discrete Structures Course: Exploring Logic and Relational Algebra with Alloy. ITiCSE Conference, 2018.

[3] Ureel, Leo; Wallace, Charles. Automated Critique of Early Programming Antipatterns. SIGCSE Conference, 2019.