CAREER: Principled Debugging for Formal Specifications

Goal: Formal specifications can be invaluable because they enable automatic checking and synthesis, but they are only useful if the specification is the correct one. Current methods to design and debug specifications are ad-hoc. This proposal aims to discover the scientific principles for encoding informal constraints as formal specifications by investigating a variety of domains.

IM: Promote the use of formal specifications for correct software, identify and fix misconceptions in specification, deepen our understanding of programming as a science that applies to specifications as well as code.

BI: Teach specification at the K-12 level, build interactive modules for students and for professional developers, release open-source debugging tools.

Research Questions

- How to translate informal descriptions into formal specifications, whether types or logical formulae or equations?
- How to test specifications to ensure they match the intended behavior?

Methods

- Finite model-finding (e.g. Alloy)
- Domain-specific visualization
- Language, type system, logic design
- Rational-programmer studies, user studies
- Random testing, mutation testing
- LLM feedback-driven translation

Examples / Application Areas

- Internet protocols: efficiently translate design documents to logical models that employ domain-specific visualization.

  Inspiration: `https://datatracker.ietf.org/rg/ufmrg/about/`

  Related: Jest, compiling ECMAscript specification to an interpreter.

- Gradual types: generate type specifications for untyped components, test specifications for additional or undesirable behaviors.
- UAV routing (local mentor Henderson): synthesize informal requirements from a variety of stakeholders (e.g., UAV manufacturers, city officials) into temporal logic specifications. Map solver errors to stakeholder constraints.
- Rhombus programming language (local mentor Flatt): when a new feature is implemented for the language prototype, generate tests that explore its interactions with existing features. Look for "surprising" combinations.

  Related: Java unsoundness due to null and generics.

  Modula 2 confusion with implicit coercions.

Abstract

Formal specifications are a prerequisite for correct software, yet programmers ofter choose not to write them because tool support for designing specification lags far behind tools for designing code. Specifications can be wrong; tools must embrace this fact and encourage exploration and debugging. This proposal aims to uncover scientific principles to support the nimble design of formal specifications. It focuses on three target areas: type specifications for untyped programs, logic specifications for uncrewed aircraft routes, and models for reactive systems. The third domain will serve as the focus for outreach efforts. The PI will develop on online environment for system modeling and deploy it in an undergraduate course that gives students hands-on experience writing specifications and in K-12 activities through the University of Utah GREAT summer camp program. To engage students, the online environment will frame specification as a game between the specifier (student) and an adversary who invents a wrong system while obeying all rules from the specification.