Tiny Idris Program Synthesis

CS408 Project Proposal - Experimentation

Scott Mora - 201816428

1 Brief

Tiny Idris is a small dependently typed functional programming language developed for teaching the implementation of dependently typed programming languages. The language is a cut down version of Idris 2, removing some sophistication from the type system, along with many of the language constructs. The small size makes the language ideal for research by removing some of the complexities.

The project would take the Tiny Idris implementation and extend it with program synthesis functionality, in a similar style to that found in the full version of Idris, Agda, and systems designed specifically for synthesizing definitions such as Synquid and ReSyn.

2 Implementation

Currently Tiny Idris has no synthesis capability. The project would begin by implementing a brute force algorithm, attempting to use terms from the Context and Environment, with type checker reducing the search space. There are then many potential enhancements that could follow:

- The algorithm could continue for a certain amount of time (possibly) synthesizing multiple definitions.
- The algorithm could be implemented to return a "best effort" definition possibly containing meta variables that would still need instantiation.
- The algorithm could take in a file with multiple definitions to be synthesized and perform them, rather than working on an individual function.
- Additional language features could be included to provide the algorithm with extra information (without "cluttering" the type signatures) such as, predicates, example input-output tuples, or hints such as functions that may be of use similar to the hint functionality in Coq.
- The type system could be extended to contain quantities, and the proof search extended to take this into account.

A potential drawback to the system presented is the removal of certain features from the Idris 2 language. This could impact the performance of the proof search algorithm, or the quality of the results (from a human readability perspective). The implementation of these features is another direction the project could take.

3 Measuring Success

A set of example functions should be created and used to test the success rate of the algorithm at various stages of development. The examples would be a set of functions. The functions should be tested on the current systems mentioned above. The set should contain functions that can currently be synthesized, and some that cannot.

The benefits of synthesizing functions are of course limited if the process takes longer than it would to "do it by hand". For this reason the time taken must also be considered, a reasonable target time should be decided. This leaves room for potential optimisations to be made, along with the study of what these potential optimisations could be.