TLC Project Proposal: Jambda

Eric Feng, Emily Sillars

ef2648, ems2331

We intend to design and implement an interpreter for our own toy language, *JambaJuice*, and use it as a vehicle to demonstrate our modular type inference plugin. Our plugin would consist of three parts: a library written in the host language to generate Hindley-Milner typing constraints, the constraints generated by the library, and the constraint solver. Our plugin would take in generated constraints and solve them on behalf of the source language's compiler. The idea is to delegate type inference functionalities to a generic library given some information about the language (such as built-in types, functions, operators, etc) instead of having to implement a separate type inference interface on a "per-compiler" basis. Instead, the programmer would only need to specify information about their language through our library.

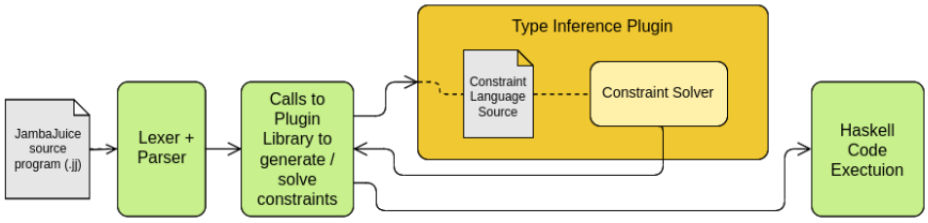

Figure 1: JambaJuice Interpreter Architecture

For simplicity, JambdaJuice will have a limited set of supported types (e.g bool, int) and we will interpret instead of compile it; these simplifications will allows us to save on toy-language development time and focus more on the constraint language/type inference part of our project.

We have two ideas for our plugin's constraint solver. The first idea is to translate our constraints to prolog and let prolog's constraint solver handle them. The second idea is to implement our own constraint solver from scratch using a Hindley-Milner constraint solving algorithm like union-find. We are leaning towards the first idea but may revert to the latter depending on time/difficulty.

Deliverables: Final Report, example JamdaJuice programs, JJ Interpreter.