Assignment 3 : A Swift Backend for BNFC

By Filbert Phang and Armaan Rashid

Introduction:

For our project, we have chosen the Haskell package BNFC (Backus-Naur Form Converter) as the foundation for our work. BNFC is a well-known tool for generating parsers in Haskell, C, Java and other languages from files written in pure EBNF (extended Backus-Naur form). Our goal is to add a backend compiling EBNF files for use in Swift's extremely fast Lotsawa parser, which accepts all LR and LL grammars and is based on the Earley parsing algorithm.

Learning Goals

BNFC operates on EBNF files, which is a DSL for specifying language grammars. It has been actively maintained for almost 20 years and has over that time been extended with backends into many functional and imperative languages like C, C++, Java, OCaml and respective parsers in those languages. We propose adding to this list by creating a Swift backend using the Lotsawa parser. Our implementation uses BNFC to compile the raw EBNF files into a format compatible with Swift and Lotsawa.

Lotsawa is not just another backend for BNFC, however: all the current backends, including the Haskell backend with Happy, rely on recursive descent parsing. Lotsawa, however, is an Earley parser boosted with smaller iterative improvements that, for any totally deterministic grammar, operates in linear time, and works for even ambiguous grammars by returning multiple parses.

In addition to leveraging BNFC, we plan to extend its functionality by implementing a new backend using the Lotsawa parser in Swift. The Lotsawa parser is a powerful parsing library for Swift, which provides efficient and flexible parsing capabilities.

Because of time constraints, we only intend to implement the backend for a subset of EBNF which is normally accepted by BNFC. In particular we are only aiming to support EBNF files which consist of standard '::=' rules and the separator and terminator pragmas.

We also plan to link to the specification learning goal for this course by implementing QuickCheck tests to ensure this cross-compilation works as intended.