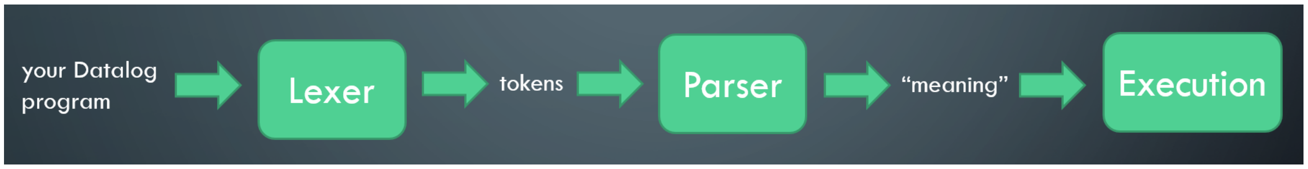
Project 4 Guide

CS236 - Discrete Structures
Instructor: Brett Decker
Fall 2020

CS236 Projects Overview

There are five projects for CS236 that are each a piece of creating an interpreter for Datalog programs (see `https://www.geeksforgeeks.org/compiler-vs-interpreter-2/` for details about interpreters). An interpreter is comprised of three components: the Lexer, the Parser, and the Execution engine. In Project 1 you will build the Lexer. In Project 2 you will create the Parser. In Projects 3, 4, and 5 you will develop the Execution engine.
Here is a graphical representation:



An execution engine is a program that takes as input the "meaning" of source code (from a parser). It executes the meaning of the program defined by the source code. For this project, you will implement the generation of new Datalog facts using the rules to complete the evaluation of queries for the Datalog execution engine.

Project 4: Execution Engine, Part II

Building this next part of the execution engine for Datalog will help you apply your study of relational data models. First, go read the entire specification for Project 4. Good software design is about breaking out the functionality into smaller pieces, which we call decomposition. If you have a good design from your previous projects you should be able to add in the new functionality with relative easy. If your design is a mess, adding the new functionality will be a nightmare.

Four Step Process

It is strongly encouraged to approach this project in four steps:
(1) create the natural join function,
(2) create the code to evaluate all rules,
(3) create the fixed-point algorithm code to converge for rule evaluation,
(4) add in the new code to finalize the database interpreter.

During and after each step, test your code for correctness and perform clean-up before moving to the next step (do this after step 4 before trying to pass off). You *could* write all the code and then just try to pass off. But then you will have now idea where the error in your code is. Testing at each step helps you localize your error detection and correction. See the project specification, lecture note (on Relational Data Model), and the notes from Dr. Goodrich for details on how to implement the first three steps (the fourth step is dependent on your code).

Conclusion

Start this project as early as possible. You will code better when not rushed, and you will be more inclined to test as you go (which will reduce overall coding time). See Project 4 on the course website for requirements and specifications.