Jake Mortlock (jm15g21)
Alberto Berni (ab3u21)
Augustas Mirinas (am26g21)

COMP 2212 Programming Language Concepts

Lexer and Parser Report

Submission date: 27 April 2023

Smol outline:

1. **Introduction**
    a. **Main features are listed (SStyping)**
    b. **Main ideas behind creation of syntax and why it was done that way**

2. **Main body**
    a. **Explain Syntax**
    b. **Scoping and lexing rules, type checking and error messages**
    c. **Overview of the execution model (cek machine with diagramsk that jake made)**
3. **Conclusion**
    a. **Running through one of the exercises showing how the language solves it this includes**
    b. **Syntax -> Interpreter -> Logical operations -> Actions in memory /CEK machine**

Introduction

The domain-specific programming language for specifying tiling patterns has been created with strong static typing and type safety in mind. The language allows for the specification of tiles, which are square discrete cells that can either be filled or empty. The main idea behind the creation of this language is to provide a convenient and safe way to specify tiling patterns, while also allowing for the expression of complex tile patterns in a concise and efficient manner.

The main features of this language include strong static typing, type safety, and a concise syntax that allows for the expression of complex tile patterns. The language also provides informative error messages and supports syntax sugar for programmer convenience. The language has been created with a formal specification to ensure its correctness and reliability.

Main Body

a. Syntax

The syntax of the language is designed to be concise and expressive. The basic structure of the language consists of a set of statements that describe the tiling pattern. The syntax supports the specification of variables, constants, functions, and operations on tiles.

The language provides a set of built-in functions for working with tiles, such as rotate, flip, and mirror. These functions can be used to transform tiles and create more complex patterns. The language also supports the definition of custom functions, which can be used to encapsulate complex logic and simplify the specification of tiles.

The syntax of the language is designed to be easy to read and understand, with a focus on readability and conciseness. This allows programmers to express complex tile patterns in a simple and efficient manner.

b. Scoping and Lexical Rules, Type Checking, and Error Messages

The language supports lexical scoping, which means that variables and functions declared in an outer scope can be accessed by inner scopes. The language also supports type checking, which ensures that the types of variables and expressions are consistent and compatible with the operations being performed on them.

The language provides informative error messages, which help programmers to identify and fix errors in their code. The error messages are designed to be easy to understand, with clear and concise explanations of the problem and suggested solutions.

c. Overview of the Execution Model (CEK Machine)

The execution model for the interpreter is based on the CEK machine. The CEK machine is a virtual machine that simulates the execution of the program. The CEK machine consists of three components: the control component, the environment component, and the continuation component.

The control component represents the current state of the interpreter, including the current statement being executed. The environment component represents the current state of the program, including the values of variables and functions. The continuation component represents the current state of the program execution, including the call stack and the current execution context.

During program execution, the interpreter performs a series of steps to evaluate the program. Each step involves transforming the current state of the interpreter into a new state, until the program is fully evaluated. The CEK machine provides a formal specification for the execution model, ensuring that the program is executed correctly and reliably.

Conclusion

In conclusion, the domain-specific programming language for specifying tiling patterns provides a convenient and safe way to specify complex tile patterns. The language features strong static typing, type safety, and a concise syntax that allows for the expression of complex tile patterns. The language also provides informative error messages and supports syntax sugar for programmer convenience. The execution model for the interpreter is based on the CEK machine, which provides a formal specification for the execution model.

To illustrate how the language solves problems, let's consider the following exercise: Given a set of tiles, generate a tiling pattern that covers a rectangular area of size M x N, where M and N are integers.

The solution would involve specifying the set of tiles, defining a function that generates the tiling pattern, and executing the function