Design Document

Seika Mahmud, Swetha Jayapathy, Sriram Rakshith Kolar

Introduction

1. Name of language: CAT
2. Paradigm: Imperative
3. Something unique: Factorial implementation

Design

Features

1. Basic data types and operations

    Data Types :
        Integer
        Boolean

    Operations :
        Addition
        Multiplication
        Negation

2. Conditionals: Our boolean data type includes a “less than or equal to” operation.

3. Loops: Our statement data type includes a while loop that takes a boolean and a statement or list of statements.

4. Variables/local names: Each function accepts a list of variable names, the values of which are set within the function and can be changed. We also used a variable to hold a while loop.

5. Functions:
    - We have implemented the Factorial as a function, which calculates the factorial of a number.
    - We have implemented the Concatenation function which conctenates the given Strings

- We have implemented the euclid function which calculates the Greatest common divisor.

Additionally, we have used the below features:

1. Strings - We created a string data type that we can perform concatenation operations on.
2. List/array data type and operations - We implemented a list type that can hold expressions, such as integers, strings, and booleans.

B) What level does each feature fall under (core, sugar, or library), and how did you determine this?

- Basic data types and operations are a core feature because without having at least integers and booleans, it would not be possible to perform any type of arithmetic or implement conditional statements.
- Conditionals and loops are also core features because they are necessary for branching and iteration.
- Variables are a core feature because in imperative languages, they are used to represent the current state of the program.
- Functions are syntactic sugar because they make it easier to reuse and organize code, but anything a function can do could also be done without it, albeit longer and messier.
- String operations are a core feature because they allow for string concatenation and other operations, which wouldn't be possible otherwise.
- Arrays are syntactic sugar because the expressions they hold could be otherwise assigned to variables or operated on.

C) What are the safety properties of your language? If you implemented a static type system, describe it here. Otherwise, describe what kinds of errors can occur in your language and how you handle them.

We have implemented the safety property for the "get" function, where we first verify whether the variable is present in our environment which is of Maybe type. So when it is not present, we map the Nothing case to 0.

Implementation

1. What semantic domains did you choose for your language? How did you decide on these?
   1. Boolean is the semantic domain for Test, as every option evaluates to Bool in that function.
   2. Integer is a semantic domain as it can be produced from the Expr function
   3. Strings is a semantic domain as it can be produced from the Expr function

4. Store, where Store is of type (Var, Int) as for all the program, we store the result in a variable.

2. Are there any unique/interesting aspects of your implementation you'd like to describe?

We implemented the semantics for factorial to be unique and also developed a program which gives us the factorial of a number.