Lab 1: Substitution Evaluator, normal order

January 10, 2024

The goal of this lab is to implement our first evaluator for the *kl* interpreter, using the substition model, in normal order. The goal is not to produce a fully functional interpreter, you are not expected to cover all the possible use cases for the language, but a restricted set which is delimited by the unit test.

1 Get Started

The base code used in the labs can be found in the directory `/courses/TDDE55/Labs/Lab1/`. In your repository for lab submission:

```
mkdir Lab1
cd Lab1
cp /courses/TDDE55/Labs/Lab1/applicative_order.py .
```

2 Instructions

In `normal_order.py`, you will find the following classes:

- `Value`: holds the name of the value and the value itself
- `Environment`: represents an execution environment, with a list of values and an optional reference to a parent environment
- `Function`: holds information for defining a function: the environment where the function was defined, the name of the arguments, the name of the return variable and the body of the function
- `ASTree`: represents the source code as an AST. It is generated from the lark parse tree
- `Evaluator`: the class that you need to complement to implement your evaluator

In the `Evaluator` class:

- In the constructor (`__init__`), you will find the initialisation of the parser, you should use the same parser as in *Exercise 1*.
- The `eval` function is the entry point for evaluating a *kl* expression. It is devided in three steps:
  1. *parsing* using the Lark parser, transform into an ASTree. In case the conversion to ASTree fails, contact your lab assistant for help.

2. *expansion* which replace identifier by their values, function call by the function body, into a tree with only elementary operations
3. *reduction* which apply the elementary operations

It is recommended you set the `debug` argument to `True` during testing to visualise the AST.

- The `reduce` computes the *reduction*. This function is fully implemented and should not be modified.
- The `expand` computes the *expansion*. This is the function that you need to implement in this lab

3 Test

You can test your evaluator with:

```
tdde55_lab1_tests dir_to_lab1
```

The test suite can be found in `/courses/TDDE55/Labs/Lab1/Tests/evaluator.py`. The following test cases have been defined:

- `test_00_literal` test an expression with just a number
- `test_01_arithmetic` test arithmetic
- `test_02_cond` test conditional expression
- `test_03_assignment` test assigning a literal to a named value and computing expression
- `test_04_custom_function` test defining and calling a custom function
- `test_05_custom_function_div_0` test passing an invalid expression as an argument
- `test_06_custom_high_order_function` test defining a high order function

The first three tests should pass without changes to the evaluator, as they require no expansion.

You can run a single test case for your evaluator with:

```
tdde55_lab1_tests dir_to_lab1 Evaluator.nameoftest
```

For instance:

```
tdde55_lab1_tests dir_to_lab1 Evaluator.test_00_literal
```

4 Demonstration

- Make sure the unit test works on the thinlinc or lab computer
- Make sure your `expand` is easy to understand and well commented