CS 3MI3: Fundamentals of Programming Languages

Due on Wednesday, November 23 at 11:59pm EST

Dr. Jacques Carette

Idea

The goals of this assignment are:

1. Combine your knowledge of semantics into a typechecker implementation

Logistics

This assignment uses the same language as that used in tutorial 07. It also assumes familiarity with typing.pl that was used in the lecture of November 3rd.

The Tasks

1 Typing [60 points]

Take the language of Tutorial 7 and its Call-By-Value semantics, and implement the same functions for it as was implemented in `typing.pl`. Specifically:

- `expr`
- `value`
- `type`
- `typed`
- `sstep`
- `mstep`
- `tsstep`
- `typederiv`

(the other predicates do **not** have to be implemented).

It is expected that you will have to add additional predicates, such as `context` and `variable`. You may use Prolog lists for contexts, or implement them yourself.

There are three ways for you to figure out the rules you need:

1. reverse engineer them from the `stepCBV` function of the tutorial,
2. find them on the various lecture slides
3. find them in the Types and Programming Language textbook.

All three sets of rules agree with each other, so you can use this fact to cross-check your answers.
Your task is to

1. Document the rules you will implement, and justify why you think those are the correct rules
2. Implement the rules in prolog.

2 Testing [40 points]

In a separate file `testing.pl` that uses your version of `typing.pl`, write queries (as tests) that achieve complete coverage of all the cases of all predicates. Make sure to include queries that return `false`.

You should document your queries to let the reader know what its intent is.

Provide a plain text file `testing.out` which is the results that you obtain when running your tests.

Your tests should run on *loading the file* `testing.pl` and should not be comments expected to be run manually. You may put some pseudo-tests in comments for things that do not work (but should) with appropriate comments. Those can be worth part marks as they indicate your partial understanding.

Submission Requirements

- Must be handed in as a `.zip`, `.gz`, or `.7z` file. Other archive formats will **not** be accepted, resulting in a score of 0. The archive should be called `A4_macemailid.zip` (with your email address, I am "carette", substituted in).
- The name of the file **does** matter.
- Code or tests which **produces errors** is worth **0** marks for the code (including testing) portion of the assignment. Let us know which platform you used (as a comment or in a README).
- Marks will be deducted if you have junk in your archive (such as object files, `.DS_Store` files, pointless subdirectories, etc.). Stack project files and cabal files are exempt from this.
- If you looked things up online (or in a book) to help, document it in your code. If you have asked a friend for help, document that too. "Looked things up online" includes all AI tools. Put this in a README file (as plain text, Markdown, or HTML). (*For this assignment, this is manditory, as it is built-in that you will have to look things up.*)