Unbounded… ≡ λ

A simple implementation of the Lambda Calculus.

# Why do this?

* There are plenty of implementations available in many languages however
  + They typically require some syntactic modification to fit, eg Python “I = lambda x : x”. JavaScript syntax is worse.   
    I would like to be able to cut/paste from a document, or to directly enter something approaching standard Lambda Calculus, eg “I = fn x.x”, or “I≡λx.x”
  + They inherit the languages internal representations. This makes error logging, tracing or displaying expressions difficult, eg the following example from Python is not very helpful:  
    “>>> I(I)  
    <function <lambda> at 0x7f16a46086e0>”  
    Would be nicer if I got back either “I” or “fn x.x”
* The Lambda Calculus is very simple, this should be the simplest language interpreter to build.
* A good excuse to learn more about functional programming, and have a little fun.

# Notes

* Implementation in Ada because
  + Ultra strict typing takes care of memory and other related issues. I just need to worry about the algorithm.   
    (Not exactly true – I spend most of my time worrying about how to correctly define the data types)
  + I have not used Ada for some time and was looking for an excuse to brush off my skills.
* Not to say this could not be done more cleanly in a functional language such as Lisp, or that this has not already been done in a more popular procedural language.

### Specification

# Syntax

Minimum viable

* Expression := **Variable** | **Function** | **Application**
* Function := **λ** **Variable**(s) **.** **Expression**
* Application := **Expression**(s)

Extensions to make it easier to use

* Synonym := **Expression**

Reserved. How many of these are actually needed?

* “**λ**” - Function
* “**fn** ” - Function
* “**F**” - Function (MVP)
* “**.**” - Function separator (MVP)
* “ ” - Space as expression separator
* “**(**“ - Start expression
* “**)**” - End expression
* “**=**” - Synonym assignment
* “:**=**” - Synonym assignment
* “**≡**” - Synonym assignment

# Data structures

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Attribute** | **Values** | **Example** | **Notes** |
| Variable | Name | [a-z] | x |  |
|  | ID |  | x1 | Internally generated |
|  | Value | int, char, ... | TRUE, FALSE  0,1,2,3  ‘stuff’ | Extension  Mapping to a typed language representation. Used to pretty print a value. |
|  | IsBound | TRUE|FALSE |  | May not need this. |
| Expression | Value |  | λx.x  I(I) | The expression itself. Can be printed. |
|  | Instructions | ... | ... | Decomposes an Expression into Functions and Variables |
| Synonym | Name | [A-Z0-9]... | I  MI  N5 | Extension.  Could be just an attribute – what of: an Expression, Variable, Application, some or all of the above? |
|  | KnownAs | string | Identity | Description. Can be printed. |
|  | Expression |  |  |  |
| Environment |  |  |  | Storage.  Could be bounded by the context. |
| Application | Value |  |  | Container of Expressions |
|  | Expression(s) |  |  | Do we need this? |
| Function | Value | **λ**Variable(s)**.** Expression | λx.x  λxy.yx |  |
|  | Instructions |  |  |  |

# Interpreter

|  |  |
| --- | --- |
| load() → Application | Read in an Application from a file or the console |
| parse(Application) → Application | Parse into executable format |
| compute(Application) → Application | Compute an expression |
| format(Application) → Application | Generates display values from executable code |
| display(Application) | Writes one or more Expressions |

Is polymorphism required for parse, compute and format?

### References

* <https://www.inf.fu-berlin.de/lehre/WS03/alpi/lambda.pdf>

### Published under the MIT license.

In the words of Tom Lehrer

*[Plagiarize!](https://genius.com/Tom-lehrer-lobachevsky-lyrics" \l "note-12999201)  
Let no one else's work evade your eyes  
Remember why the good Lord made your eyes  
So don't shade your eyes  
[But plagiarize, plagiarize, plagiarize](https://genius.com/Tom-lehrer-lobachevsky-lyrics" \l "note-12999201)  
[Only be sure always to call it please "research"](https://genius.com/Tom-lehrer-lobachevsky-lyrics" \l "note-8017028)*