Design and implementation of `ps11` – Lisp-like programming language which compiles to Pyramid Scheme

Marcin Konowalczyk[1, a)]

Department of Chemistry, University of Oxford, Inorganic Chemistry Laboratory, Oxford OX1 3QR, U.K.

(Dated: September 22, 2020)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords: a; b; c

I. INTRODUCTION

Pyramid Scheme (PS) is a variant of the Scheme dialect of Lisp, designed by Conor O'Brien [info][???].

Golf TiO

A. Implementation of Pyramid Scheme

Ruby Dictionary Memory (RDM)

[1]

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget

[a)] Electronic mail: marcin.konow@lczyk.xyz

nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

II. LANGUAGE DESCRIPTION

A. Bracket structure

B. Syntactic sugar

The above specification is, in principle, enough to create fully fully functional PS programs. Certain tasks are, however, still rather cumbersome. This section outlines these cases, as well as syntactic sugar constructs introduced to `psll` to alleviate them. All of these are implemented as local (or semi-local) expansion macros, as described in Section III B. Despite authors best efforts, this introduces some sharp edges into the language (see Section II C).

Implicit bracket expansion Each bracket must have exactly three elements. For small expressions this is almost always the case, but becomes problematic for larger, flow-control and loop structures where each such expression can contain an arbitrarily large number of sub-expressions (see **[info]** for an example of such expression). Hence a bracket containing > 2 other brackets gets expanded as follows:

```
( (out 1) (out 2) (out 3) (out 4) (out 5) )
```

is interpreted as:

```
( (((out 1) (out 2)) ((out 3) (out 4))) (out 5) )
```

Each neighbouring pair or elements of the parent gets put together into a bracket, until the length of the parent is less than 2. This results in a (literal) balanced binary tree in the final PS code, and so for a parent bracket of N sub-expressions will result in a tree of height $\mathcal{O}\left(\log_2(N)\right)$.

String literals Single characters can be created in RDM with the `chr` keyword (Ruby `.to_i.chr`). It is also possible to construct longer strings in RDM since Ruby's "+" sign overloads string concatenation. The string *"hello"* is therefore:

```
(+ (+ (+ (+ (chr 72) (chr 101)) (chr 108))
   (chr 108)) (chr 111))
```

`psll` introduces string literals, such that `(set s "hello")` expands into the above code. Note that this is a very left-child heavy tree. To balance it, the above string could also be made by recursively concatenating its binary split:

```
(+ (+ (chr 72) (chr 101))
   (+ (chr 108) (+ (chr 108)) (chr 111)))
```

such that *"hello"* = *"he"* + *"llo"* = (*"h"* + *"e"*) + (*"l"* + *"lo"*) = ...

Array literals

Rolling sum and product

`def` keyword

Semi-local

C. Sharp edges

As mentioned at the beginning of Section II B, the introduction of syntactic sugar into `psll` introduces some edge cases which one ought to watch out for.

Underscore keyword _

Escape characters Because " is used for strings, and [ ] for arrays...

III. COMPILER

A. Abstract syntax tree

B. Local macro expansion

C. Optimisation

IV. EXAMPLE PROGRAMS

A. Pseudorandom number generation

B. Bubble sort

C. Chess engine

```
1 (set a 0) // Flip-flop
2 (set N 10) (set j 0) // N of iteration and loop counter
3 (loop (! (= j N)) (
4    // Do some work...
5    (out j (chr 32)) // Print j and space
6    (out a (chr 10)) // Print a and newline
7    (set a (! a)) // Flip a
8 (set j (+ j 1))
9 )) (set test "hi")
```

V. CONCLUSIONS AND OUTLOOK

Program in Pyramid Scheme! Teach your friends! Have them teach *their* friends! Then have those friends teach *their* friends!

- Joined pyramids

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien

facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo, nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue. `(chr 10)`

LINKS

- psll git
- pyramid scheme
- TiO hello world in ps
- Esolang page?
- Code Golf page

ACKNOWLEDGEMENTS

REFERENCES

[1] David Beazley. Reinventing the parser generator. In *PyCon2018*, 2018.
[2] Conor O'Brien. Pyramid scheme. `https://github.com/ConorOBrien-Foxx/Pyramid-Scheme`, 2017.