my own Implementation of I-Expressions

A lot of you hopefully already know their way around the family of Lisp-dialects. My personal favorite has got to be Scheme (R6RS).

Lately I've been getting into Haskell and Compilers a lot more and had the idea of using indentation instead of S-Expressions for Scheme. Obviously I wasn't the first one with that idea and found some articles on SRFI (Indentation-sensitive syntax) and Wikipedia (M-Expressions) regarding the subject of a new syntax for Lisp.

Next thing I wanted to do, was implement my own implementation of I-Expressions. So I hacked together some Haskell and after struggling a little bit I managed to get everything working.

The product was a small compiler that translates I-Expressions into S-Expressions. You can find my implementation on GitHub, Hackage and the AUR. I will specify the rules to the indentation in the `README.md`.

I still plan on refining the project to be useful for more usage, maybe to allow translation in the other direction and cover some more stuff like SheBangs and comments. It's still very barebones at the moment, but should work fine so you can give it a try!

In the following example I define a function `f` which calculates $f(x) = x^2 + 1$.

I-Expressions:

```
define f
  lambda
    x
    let
        y
          * x x
      + y 1
```

As you can see I-Expression use a lot of lines and readability is not very good since there is very little information per line.

S-Expressions:

```
(define f
  (lambda (x)
    (let ((y (* x x)))
      (+ y 1)))))
```

S-Expression are much more readable and most people will be kind of comfortable with them, but when you get to the end of the definition you cant distinguish which parantheses belong to each other.

mixed I- and S-Expressions:

```
define f
  lambda (x)
    let
        y (* x x)
      + y 1
```

Once you mix indented and symbolic expressions you get the best of both worlds.

Obviously the notation of S-Expressions is useful, as it doesn't differ from data to code, but readability will be increased with less parentheses.