Introduction to Program Synthesis - Sheet 2

Lambda, expression and syntax trees

Deadline: 03-06-2025 - Anywhere on Earth (AoE, UTC-12)

Through the following exercises you will set up basic structures which will be used in future exercises. Do your best to keep your code clean and modular. In this sheet we are working with λ and S-expressions.

Exercise 1: Lambda tree

Consider the following λ-expression:

$$(\lambda z.\lambda y(z\,y\,w)((\lambda z.z)(\lambda x.x)))$$

Draw the above λ-expression as a lambda tree. Perform β-reductions and draw the immediate reduction steps also as lambda trees to visualise the reduction sequence. Also write down the β-reduction steps until in expression form by using the notations that have been presented in the lecture. **Note:** As reduction strategy apply applicative order whereby the rightmost, innermost redex is considered first.

Exercise 2: Expression tree

Consider the following mathematical expression:

$$(x - (\frac{y}{4})) + (z * (t^2))$$

Draw the above s-expression as an expression tree and write down a proper textual representation that can be used for parsing the expression..

Exercise 3: Syntax tree

Consider the following s-expression:

```
(defun foobar (
   (setq a 1)
   (setq b 3)
   (if
     (< a b)
     (setq c (- b a))
     (setq c (- a b))
   )
 ))
```

For the given example, draw the corresponding syntax tree.

Exercise 4: Tree implementation and parsing

Based on the expression parser seen during the lecture[a], write a parser in Python to read a textual representation of the above expression in Exercise 1 and store it in a syntax tree. Traverse and print the tree obtained and compare them to your previous answers to check for correctness.
For this exercise the following tree function should be implemented:

- `create_tree`
- `create_node`
- `remove_tree`
- `traverse`

Hint: In addition to the code example from the lecture, you can generally orientate on the the example of a binary search tree. It an also help to sketch the dependencies of the respective nodes and edges on a sheet of paper.

[a] https://github.com/RomanKalkreuth/program-synthesis-lecture/blob/main/lecture/examples/chap-02/nested_list.c