Discussion 10: November 11th

Operational Semantics

Formal semantics of a PL: Mathematical description of meaning of programs in a PL.

Terminologies in Operational Semantics

```
    A; e1 => v1   A;e2 => v2   v3 is v1 + v2
    ---------------------------------------
              A; e1 + e2 => v3
```

- **Expression**: A program that evaluates to a value
- **Value**: A result of an expression
- **Environment**: A mapping from variables to values
- **Hypothesis**: A set of rules that describe the meaning of expressions
- **Judgement**: A statement with expressions and values (`e => v`). The expression `e` evaluates to the value `v`.

Operational Semantics Exercise

Solve the following problems using operational semantics:

```
Evaluate 1 + (2 + 3) = 6 using the following hypotheses:

                                      e1 => v1   e2 => v2   v3 is v1 + v2
       ------------                  ---------------------------------------
         n  => n                                 e1 + e2 => v3
```

Evaluate the expression using the given hypotheses: A; let y = 1 in let x = 2 in x => 2

$$\frac{}{n \Rightarrow n} \qquad \frac{A(x) = v}{A; x \Rightarrow v}$$

$$\frac{A; e1 \Rightarrow v1 \quad A, x: v1; e2 \Rightarrow v2}{A; \text{let } x = e1 \text{ in } e2 \Rightarrow v2} \qquad \frac{A; e1 \Rightarrow v1 \quad A; e2 \Rightarrow v2 \quad v3 \text{ is } v1 + v2}{A; e1 + e2 \Rightarrow v3}$$

Lambda Calculus

Lambda Calculus Terminologies

```
\x. \y. x + y + a
```

- **Lambda Expression**: A lambda expression is a function that takes an argument and returns a value.
- **Free Variable**: A variable that is not bound by a lambda expression.
- **Lambda Abstraction**: A lambda abstraction is a function that takes an argument and returns a function.
- **Alpha Conversion**: A process of renaming a variable in a lambda expression to avoid name conflict. Does not change the meaning of the expression. **Do not rename free variables**
- **Beta Reduction**: A process of substituting a lambda expression for a variable in a lambda abstraction.

Lambda Calculus Exercise

Things to keep in mind: 1. Alpha conversion: Do not rename free variables 2. Explicit Parentheses: Scope of a variable extends to far right or the first `)` seen. 3. Lambda Calculus is left-associative. 4. Beta reduction: Keep applying the functions until you can't anymore.

Solve the following problems using lambda calculus:

1. Make parentheses explicit in the following lambda expressions:

```
1. a b c



2. \a. \b. a b



3. \a. a b \a. a b
```

2. Identify the free variables in the following lambda expressions:

```
1. \a. a b a
```

```
2. \a. \b. a b
```

```
3. \a. (\b. a b) a b
```

3. Perform alpha conversion on the following lambda expressions:

```
1. \a. \a. a
```

```
2. (\a. a) a b
```

```
3. (\a. (\a. (\a. a) a) a) a
```

4. Perform beta reduction on the following lambda expressions:

1. (\a. a b) x b

2. (\a. b) (\a. \b. \c. a b c)

3. (\a. a a) (\a. a a)