Lab 4

Due Wednesday, March 5 by midnight

Handout 8
CSCI 334: Spring 2025

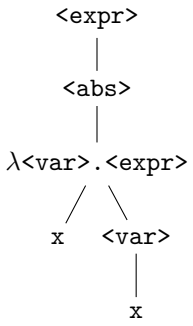
Encoding Trees

Each question in this assignment must be answered by writing a function that returns a tree. To facilitate working with trees, I provide some starter code, `ParseViz.fs`, in your repository. For each question, you will need to create a new F# console program that opens this library's module, `ParseViz`, just as your previous assignments asked you to open `CS334`.

`ParseViz` comes with the following record datatype for encoding trees.

```
type Node = { label: string; children: Node list }
```

Suppose you have the following tree:



To use `ParseViz`, we encode every node in the tree as a `Node`, with the node's text encoded as a `label`. Child nodes are added to a `Node`'s list of `children`. The above tree is encoded as follows. I use lots of whitespace to make things clear but you do not have to.

```
let tree =
  { label = "<expr>";
    children = [
      { label = "<abs>";
        children = [
          { label = "L<var>.<expr>";
            children = [
              { label = "x";
                children = []
              };
              { label = "<var>";
                children = [
                  { label = "x";
                    children = []
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
```

The `prettyprint` function in `ParseViz` can be used to visualize the tree, which is helpful to check your work. Use the following in your `main` function to print the tree to a `string` and to save that `string` to a file.

```
let tree1_latex = prettyprint tree1
System.IO.File.WriteAllText("tree1.tex", tree1_latex)
```

The easiest way to view this output, which is a LaTeX document, is to paste it into Overleaf. Alternatively, if you have `texlive` installed, you can run the following on the command line,

```
$ pdflatex tree1.tex
```

and then view the resulting `tree1.pdf` file. Every CS lab computer has `texlive` pre-installed for your use.

For derivations, a correct tree will show non-terminals surrounded by `<angle brackets>` while terminals will be `bare`. Whitespace does not matter, so feel free to add it if it helps you.

For lambda calculus derivations, be sure to encode expression as `<expr>`, application as `<app>`, abstractions as `<abs>`, and variable as `<var>`. Non-terminals, like `x`, should be encoded without brackets. You should also include the period, `.`, and lambda non-terminals in your derivation trees. You may either write `L` for the lambda symbol, or if you're feeling adventurous, the LaTeX macro `$\\lambda$`, which will print a λ character.

Coding Guidelines

Each question in this assignment should go into the appropriate project directory. For example, the solution to question 1 should be in a folder called "`q1`". When a solution is a program, one should be able to `cd` into the question directory and then run your program by typing the command "`dotnet run`", with additional arguments depending on the question.

Every program should be split into two pieces: a "`Program.fs`" file that contains the `main` method and associated program-startup helpers (if needed), and another "`Library.fs`" file that contains the function(s) of interest in the question. Library code should be contained within a module named "`CS334`". Be sure to provide usage output (defined in `main`) for all programs that require arguments. For full credit, your program should both build and run correctly.

Turn-In Instructions

Turn in your work using your assigned `git` repository. The name of your repository will have the form `https://aslan.barowy.net/cs334-s25/cs334-lab04-<USERNAME>.git`. For example, if your CS username is `abc1`, the repository would be `https://aslan.barowy.net/cs334-s25/cs334-lab04-abc1.git`.

You should have received an invite to commit to the repository via email. If you did not receive an email, please contact me right away!

Single-Author Programming Assignment

This is a solo lab. You may work with another classmate to understand what the problems ask, but you are not permitted to develop solutions together. Submitted solutions must be exclusively your own. Please refer to the section "single author programming assignments" in the honor code handout for additional information. You do not need to submit a `collaborators.txt` file for this assignment. You are always welcome to ask me for clarification if the above is unclear in some circumstance.

This assignment is due on Wednesday, March 5 by midnight.

Reading

1. **(Required)** "Syntax"
2. **(Required)** "Introduction to the Lambda Calculus, Part 1"

Problems

Q1. (34 points) ..... Parse Tree

Encode the two possible derivation trees for the expression "`1 - 5 + 24`". Each tree should be encoded as one of the following functions.

```
let q1a() : Node = ...

let q1b() : Node = ...
```

`q1a()` should encode the tree where the first non-terminal after the start symbol is an addition expression. `q1b()` should encode the tree where the first non-terminal after the start symbol is a subtraction expression. To solve this problem, refer to the grammar at the top of the section titled Ambiguity in the reading, "Syntax".

The project directory for this question should be called "`q1`". Your functions should be in a module called `CS334` in a file called `Library.fs` and your `main` function should be in a file called `Program.fs`.

Q2. (33 points) ..... Handling Ambiguity

Encode derivation trees for the following expressions, assuming the precedence and associativity rules shown in Table 2 of the reading, "Syntax."

(a) `1 + 1 * 1` in the function `q2a()`,

(b) `1 + 1 - 1` in the function `q2b()`,

(c) `1 - 1 + 1 - 1 * 1`, but give `+` higher precedence than `-`, in the function `q2c()`

The project directory for this question should be called "`q2`". Your functions should be in a module called `CS334` in a file called `Library.fs` and your `main` function should be in a file called `Program.fs`.

Q3. (33 points) ..... Parsing Lambda Expressions

Given the following grammar for the lambda calculus,

```
<expr>   ::= <var>
           | <abs>
           | <app>
           | <parens>
<var>    ::= x | y
<abs>    ::= λ<var>.<expr>
<app>    ::= <expr><expr>
<parens> ::= (<expr>)
```

encode derivations for the following expressions. Refer to the reading for precedence and associativity rules.

(a) $\lambda x.xy$ in the function `q3a()`,

(b) $\lambda x.x\lambda y.xx$ in the function `q3b()`, and

(c) $(\lambda x.\lambda y.xy)(\lambda x.xy)$ in the function `q3c()`.

The project directory for this question should be called "`q3`". Your functions should be in a module called `CS334` in a file called `Library.fs` and your `main` function should be in a file called `Program.fs`.

Q4. ($\frac{1}{10}^{\text{th}}$ bonus point) ..... Optional: Feedback

I always appreciate hearing back about how easy or difficult an assignment is.

For $\frac{1}{10}^{\text{th}}$ of a bonus to your final grade, please fill out the following Google Form.