TPL Chapter 5 Exercise
Your Name

Questions

5.2.1 Define logical `or` and `not` functions

5.2.4 Define a term for raising one number to the power of another.

5.2.7 Write a function `equal` that tests two numbers for equality and returns a Church boolean. For example,

$$\begin{aligned} & \text{equal } c_3\ c_3; \\ > & (\lambda t.\ \lambda f.\ t) \\ & \text{equal } c_3\ c_2; \\ > & (\lambda t.\ \lambda f.\ f) \end{aligned}$$

5.2.8 A list can be represented in the lambda calculus by its `fold` function. (OCaml's name for this function is `fold_left`; it is also sometimes called `reduce`.) For example, the list `[x,y,z]` becomes a function that takes two arguments `c` and `n` and returns `c x (c y (c z n))`. What would the representation of `nil` be? Write a function `cons` that takes an element `h` and a list (that is, a `fold` function) `t` and returns a similar representation of the list formed by prepending `h` to `t`. Write `isnil` and `head` functions, each taking a list parameter. Finally, write a `tail` function for this representation of lists (this is quite a bit harder and requires a trick analogous to the one used to define `prd` for numbers).

5.2.11 Use `fix` and the encoding of lists from Exercise 5.2.8 to write a function that sums lists of Church numerals

5.3.6 Adapt these rules to describe the other three strategies for evaluation—full beta-reduction, normal-order, and lazy evaluation.

5.3.8 Exercise 4.2.2 introduced a "big-step" style of evaluation for arithmetic expressions, where the basic evaluation relation is "term `t` evaluates to final result `v`." Show how to formulate the evaluation rules for lambda-terms in the big-step style.

Appendix

Here is an example latex typesetting of an inductive definition. You can copy/paste and modify this code. After finishing all the exercises, feel free to delete this section.

$$\frac{}{\text{zero nat}} \qquad \frac{n \text{ nat}}{\text{succ}(n) \text{ nat}}$$

You can nest \inferrule* for derivation tree. For example:

$$\frac{\dfrac{\text{empty tree}\quad\text{empty tree}}{\text{node(empty;empty) tree}} \qquad \text{empty tree}}{\text{node(node(empty;empty); empty) tree}}$$