Name SOLUTIONS

Wednesday, September 23

Email 6.818-www@mit.edu

6.818 Fall 2020 Miniquiz #10 5 Minutes

1. Suppose we extended IMP to support floating-point values (with the constructor `Float(f)`) in addition to integers. Now suppose we would like to define a new type of expression `[e]`, which converts the floating-point expression `e` to an integer value by truncating the decimal part. For instance, the expression `[2.3]` evaluates to 2 while the expression `[-2.3]` evaluates to -2.

   Write inference rules that define the expression `[e]` (assume you can use the standard floor and ceil functions to write your inference rules):

$$\frac{(\sigma, h, e) \rightarrow \text{Float}(f) \qquad f \geq 0 \qquad \text{floor}(f) = n \qquad \text{Integer}(n) = v}{(\sigma, h, [e]) \rightarrow v}$$

$$\frac{(\sigma, h, e) \rightarrow \text{Float}(f) \qquad f < 0 \qquad \text{ceil}(f) = n \qquad \text{Integer}(n) = v}{(\sigma, h, [e]) \rightarrow v}$$

2. Consider the extension to IMP that we saw last lecture, which added block scopes to the language so that braces introduce new local frames. How do scoping rules in this extended version of IMP differ from that of MITScript?

   In MITScript, braces that are used to delimit the bodies of while loops and if statements do not introduce new local frames/scopes.