Programming Languages

Project 1

Big-Step Evaluation Rules for L1++

$$\frac{}{\mathcal{E};\ nil\ \Downarrow\ nil}$$

$$\frac{}{\mathcal{E};\ Icons(M,N)\ \Downarrow\ Icons(M,N,\mathcal{E});\ \mathcal{S}}$$

$$\frac{\mathcal{E};\ \mathcal{S};\ M\ \Downarrow\ nil;\ \mathcal{S}' \qquad \mathcal{E};\ \mathcal{S}'N\ \Downarrow\ U;\ \mathcal{S}''}{\mathcal{E};\ \mathcal{S};\ match\ M\ \{nil\ \rightarrow\ N\mid cons(x,l)\ \rightarrow\ R\}\ \Downarrow\ U;\ \mathcal{S}''}$$

$$\frac{\mathcal{E};\ \mathcal{S};\ M\ \Downarrow\ Icons(V,L,\mathcal{F});\ \mathcal{S}' \qquad \mathcal{F};\ \mathcal{S}';\ V\ \Downarrow\ J;\ \mathcal{S}'' \qquad \mathcal{F};\ \mathcal{S}'';\ L\ \Downarrow\ K;\ \mathcal{S}''' \qquad \mathcal{E}[x\ \rightarrow\ J][l\ \rightarrow\ K];\ \mathcal{S}''';\ R\ \Downarrow\ U;\ \mathcal{S}''''}{\mathcal{E};\ \mathcal{S};\ match\ M\ \{nil\ \rightarrow\ N\mid cons(x,l)\ \rightarrow\ R\}\ \Downarrow\ U;\ \mathcal{S}''''}$$

Unlike Eager Lists, Lazy Lists defer evaluation of their elements until explicitly needed. This is reflected in the second creation rule, where expressions are bound to the current environment but not immediately evaluated. Evaluation occurs later through the match construct.

In match, if M evaluates to nil, the behaviour mirrors that of Eager Lists. If M is a Lazy List, its Node contents are first evaluated in the Node's environment, after which processing continues as with an Eager List.

Implementation

Implementation wise, I created 3 new IValue's:

- VNil – To represent an empty list
- VCons – To represent a list node
- VICons - To represent a lazy list node

And 4 new ASTNode's:

- ASTNil – To create VNil values
- ASTList – To create VCons values
- ASTLazyList – To create VICons values
- ASTMatch – To create nodes for the *match* construct