Lambda Calculus $\xrightarrow{\tau} \text{CL}$

That is not just of theoretical interest, but of eminently practical interest to implement a functional language on a machine. See, for example:


So, what is CL like? In Maude, CL has a very simple definition as a functional system module. Since CL is deterministic, it could also be specified as a functional module, replacing the keywords mod by fixed, and $\mathit{cl}$ by $\text{eq}$] of the form:
mod CL is

sort CL.

ops S K I : CL. *** basic combinators

op --- : CL CL → CL *** application

vars x y z : CL.

rl [K-red] : (K x) y ⇒ x.

rl [S-red] : ((S x) y) z ⇒ (x z) (y z).

rl [I-red] : I x ⇒ x.

endm

The meaning of the K combinator is that \( K x \) is the constant function that returns \( x \) when applied to any argument \( y \).

The meaning of \( S \) is that of a generalized application: \( x \) is applied to \( y \) and \( z \), and then \( (x z) \) is applied to \( (y z) \).

The meaning of \( I \) is the identity function.
Exercise. To get a feeling for the translation \( \tau \), consider the following instance of the translation:

\[
\tau (\lambda x. (\lambda y. (y x))) = \frac{(S(K(SI)))(S(KK))I)}{A = \uparrow},
\]

where, as you can see, names and \( \lambda \)-quantifiers have disappeared! [for \( \tau \) you can consult several sources such as the Combinatory logic Wikipedia page, the above two references for Turner and Peyton Jones, and Hindley and Selding book, "\( \lambda \)-Calculus and Combinator," Cambridge University Press].

Do the following:

1. Specify CL in Maude as suggested in the specification in page 8.

2. Apply by hand rules \( K \text{-} red \), \( S \text{-} red \) and \( I \text{-} red \) to the CL expression \( (A \times y) y \), where \( A \) abbreviates the righthand side of (88) above to check that \( \tau \) is correct, since it gives the same result as the \( \beta \)-reduction of \( (\lambda x. (\lambda y. (y x)) x) y \).

3. Compute \( (A \times y) y \) automatically by the rewrite command in Maude.