LAMBDA

MODULE LAMBDA

SYNTAX   *Exp* ::= *Int*
         | *Bool*
         | *Id*
         | *(Exp)* [bracket]
         | *Exp Exp* [strict]
         | *Exp * Exp* [strict]
         | *Exp / Exp* [strict]
         | *Exp + Exp* [strict]
         | *Exp <= Exp* [strict]
         | `lambda` *Id . Exp* [binder]
         | `if` *Exp* `then` *Exp* `else` *Exp* [strict]
         | `let` *Id = Exp* `in` *Exp* [binder]
         | `letrec` *Id Id = Exp* `in` *Exp* [binder]
         | `mu` *Id . Exp* [binder]

SYNTAX   *Type* ::= `int`
         | `bool`
         | *Type* `->` *Type*
         | *(Type)* [bracket]

SYNTAX   *Exp* ::= *Type*

SYNTAX   *KResult* ::= *Type*

CONFIGURATION:

RULE $\dfrac{I{:}Int}{\texttt{int}}$

RULE $\dfrac{B{:}Bool}{\texttt{bool}}$

RULE $\dfrac{T1{:}Type * T2{:}Type}{T1 = \texttt{int} \curvearrowright T2 = \texttt{int} \curvearrowright \texttt{int}}$

RULE $\dfrac{T1{:}Type / T2{:}Type}{T1 = \texttt{int} \curvearrowright T2 = \texttt{int} \curvearrowright \texttt{int}}$

RULE $\dfrac{T1{:}Type + T2{:}Type}{T1 = \texttt{int} \curvearrowright T2 = \texttt{int} \curvearrowright \texttt{int}}$

RULE $\dfrac{T1{:}Type <= T2{:}Type}{T1 = \texttt{int} \curvearrowright T2 = \texttt{int} \curvearrowright \texttt{bool}}$

SYNTAX   *Exp* ::= *Exp* `->` *Exp* [strict]

RULE $\dfrac{\texttt{lambda}\ X\ .\ E{:}Exp}{T \texttt{ -> } E[T\ /\ X]}$ requires `fresh` $(T{:}Type)$

RULE $\dfrac{T1{:}Type\ \ T2{:}Type}{T1 = (T2\texttt{ -> }T) \curvearrowright T}$ requires `fresh` $(T{:}Type)$
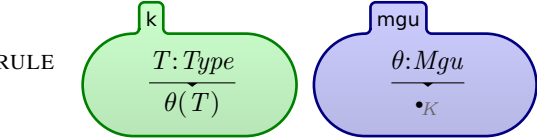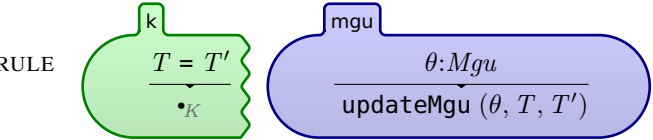
RULE $\dfrac{\texttt{if}\ T{:}Type\ \texttt{then}\ T1{:}Type\ \texttt{else}\ T2{:}Type}{T = \texttt{bool} \curvearrowright T1 = T2 \curvearrowright T1}$

RULE $\dfrac{\texttt{let}\ X = E\ \texttt{in}\ E'}{(\texttt{lambda}\ X\ .\ E')\ E}$ [macro]

RULE $\dfrac{\texttt{letrec}\ F\ X = E\ \texttt{in}\ E'}{\texttt{let}\ F = \texttt{mu}\ F\ .\ \texttt{lambda}\ X\ .\ E\ \texttt{in}\ E'}$ [macro]

RULE $\dfrac{\texttt{mu}\ X\ .\ E}{(T\texttt{ -> }T)\ E[T\ /\ X]}$ requires `fresh` $(T{:}Type)$

SYNTAX   *K* ::= *Type = Type*

RULE

RULE

END MODULE