LAMBDA

MODULE LAMBDA

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

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

SYNTAX *Exp* ::= *Type*

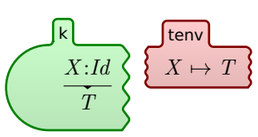
SYNTAX *Variable* ::= *Id*

SYNTAX *KResult* ::= *Type*

CONFIGURATION:

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

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

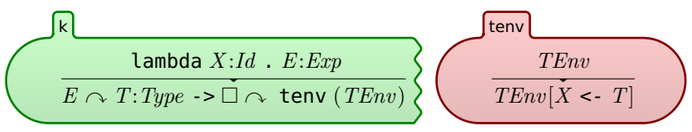
RULE 

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

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

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

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

RULE 

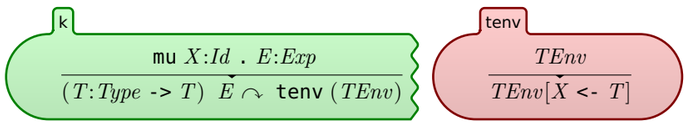
RULE $\frac{T2:Type \curvearrowright T1:Type \texttt{ -> } \square}{T1 \texttt{ -> } T2}$

RULE $\frac{T1:Type \quad T2:Type}{T1 = (T2 \texttt{ -> } T:Type) \curvearrowright T}$

RULE $\frac{\texttt{if } T:Type \texttt{ then } T1:Type \texttt{ else } T2:Type}{T = \texttt{bool} \curvearrowright T1 = T2 \curvearrowright T1}$

RULE $\frac{\texttt{let } X = E \texttt{ in } E'}{E'[E / X]}$ [macro(macro())]

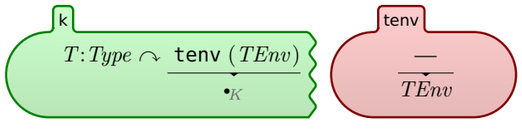
RULE $\frac{\texttt{letrec } F \quad X = E \texttt{ in } E'}{\texttt{let } F = \texttt{mu } F \texttt{ . lambda } X \texttt{ . } E \texttt{ in } E'}$ [macro(macro())]

RULE 

SYNTAX *KItem* ::= *Type* = *Type*

RULE $\frac{T = T}{\bullet_K}$

SYNTAX *KItem* ::= tenv (*Map*) [klabel(klabel('tenv))]

RULE 

END MODULE