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*
| `letrec` *Id* *Id* `=` *Exp* `in` *Exp*
| `mu` *Id* `.` *Exp* [binder]

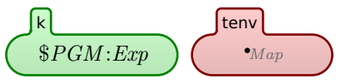
SYNTAX *Type* ::= `int`
| `bool`
| *Type* `->` *Type*
| (*Type*) [bracket]

SYNTAX *Exp* ::= *Type*

SYNTAX *Variable* ::= *Id*

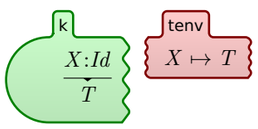
SYNTAX *KResult* ::= *Type*

CONFIGURATION:



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

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

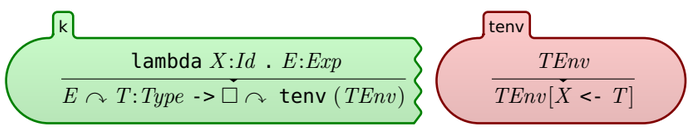
RULE 

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

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

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

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

RULE 

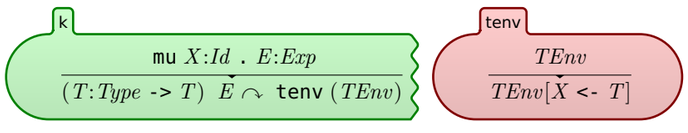
RULE $\frac{T2:Type \curvearrowright T1:Type \texttt{ -> } \square}{T1 \texttt{ -> } T2}$

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

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

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

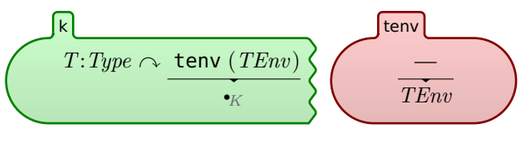
RULE $\frac{\texttt{letrec } F \quad X \texttt{ = } E \texttt{ in } E'}{\texttt{let } F \texttt{ = mu } F \texttt{ . lambda } X \texttt{ . } E \texttt{ in } E'}$ [macro]

RULE 

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

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

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

RULE 

END MODULE