

LAMBDA

MODULE LAMBDA

SYNTAX $Exp ::= Int$
| $Bool$
| Id
| (Exp) [bracket]
| $Exp\ Exp$ [strict]
| $Exp * Exp$ [strict]
| Exp / Exp [strict]
| $Exp + Exp$ [strict]
| $Exp \leq Exp$ [strict]
| $\text{lambda } Id . Exp$
| $\text{if } Exp \text{ then } Exp \text{ else } Exp$ [strict]
| $\text{let } Id = Exp \text{ in } Exp$ [strict(2)]
| $\text{letrec } Id\ Id = Exp \text{ in } Exp$
| $\text{mu } Id . Exp$

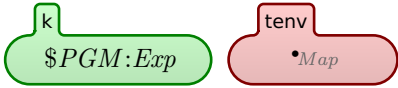
SYNTAX $Type ::= \text{int}$
| bool
| $Type \rightarrow Type$
| $(Type)$ [bracket]

SYNTAX $Exp ::= Type$

SYNTAX $Variable ::= Id$

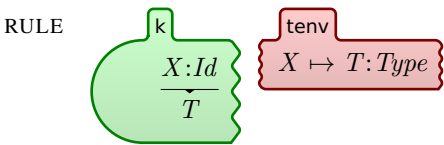
SYNTAX $KResult ::= Type$

CONFIGURATION:



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

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

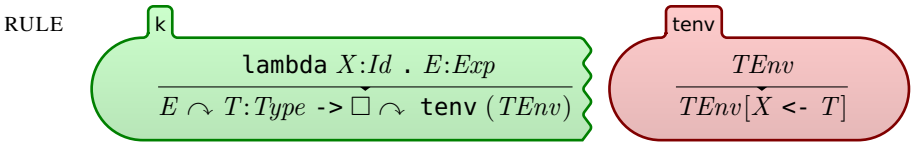


RULE $\frac{T1: Type * T2: Type}{T1 = \text{int} \curvearrowright T2 = \text{int} \curvearrowright \text{int}}$

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

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

RULE $\frac{T1: Type \leq T2: Type}{T1 = \text{int} \curvearrowright T2 = \text{int} \curvearrowright \text{bool}}$



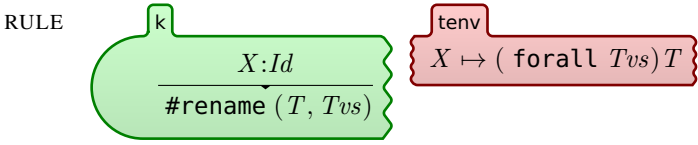
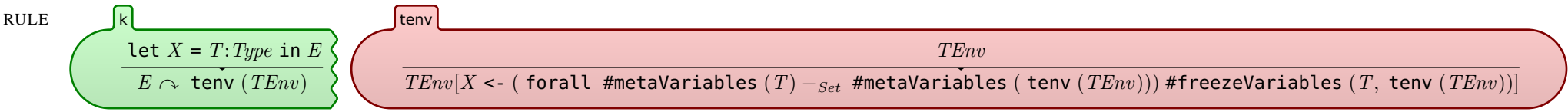
RULE $\frac{T2: Type \curvearrowright T1: Type \rightarrow \square}{T1 \rightarrow T2}$

RULE $\frac{T1: Type\ T2: Type}{T1 = (T2 \rightarrow T: Type) \curvearrowright T}$

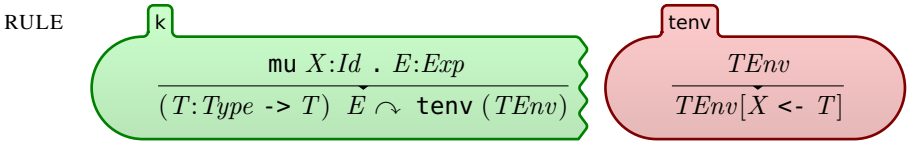
RULE $\frac{\text{if } T: Type \text{ then } T1: Type \text{ else } T2: Type}{T = \text{bool} \curvearrowright T1 = T2 \curvearrowright T1}$

SYNTAX $TypeSchema ::= (\text{forall } Set) Type$

SYNTAX $Type ::= MetaVariable$



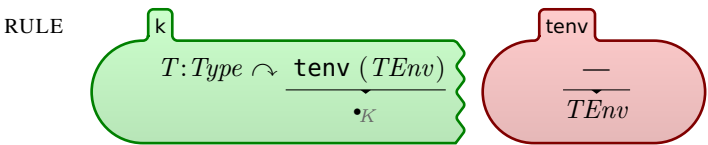
RULE $\frac{\text{letrec } F\ X = E \text{ in } E'}{\text{let } F = \text{mu } F . \text{lambda } X . E \text{ in } E'}$



SYNTAX $KItem ::= Type = Type$

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

SYNTAX $KItem ::= \text{tenv } (Map)$ [klabel('tenv)]



END MODULE

[macro]