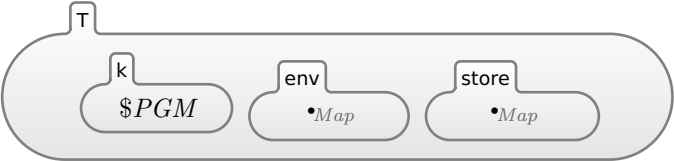


# LAMBDA

MODULE LAMBDA

SYNTAX  $Exp ::= Id$   
           $| \lambda Id. Exp$   
           $| Exp \ Exp \text{ [strict]}$   
           $| (Exp) \text{ [bracket]}$

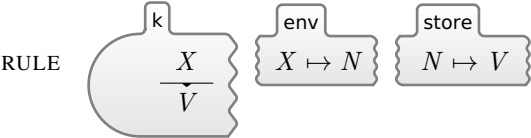
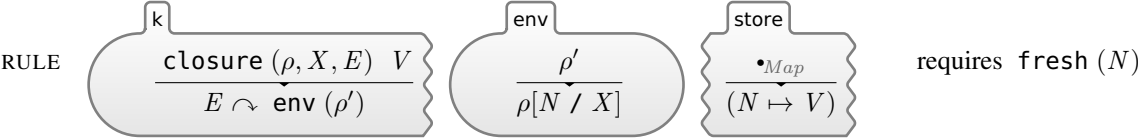
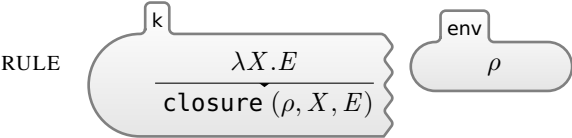
CONFIGURATION:



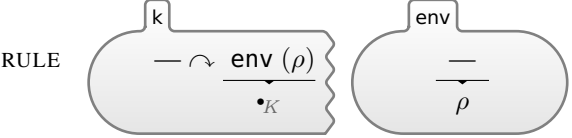
SYNTAX  $Val ::= \text{closure } (Map, Id, Exp)$

SYNTAX  $Exp ::= Val$

SYNTAX  $KResult ::= Val$

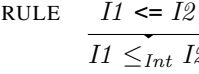
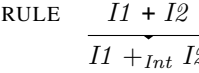
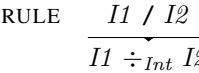
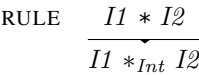


SYNTAX  $K ::= \text{env } (Map)$

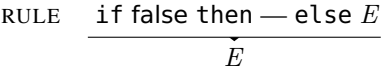
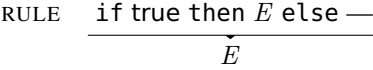


SYNTAX  $Val ::= Int$   
           $| Bool$

SYNTAX  $Exp ::= Exp * Exp \text{ [strict]}$   
           $| Exp / Exp \text{ [strict]}$   
           $| Exp + Exp \text{ [strict]}$   
           $| Exp <= Exp \text{ [strict]}$



SYNTAX  $Exp ::= \text{if } Exp \text{ then } Exp \text{ else } Exp \text{ [strict(1)]}$



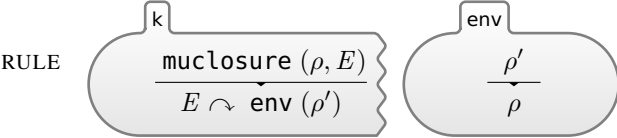
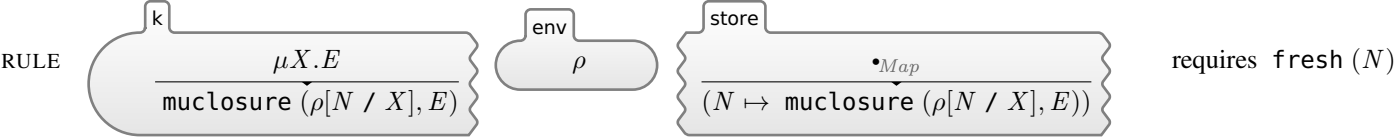
SYNTAX  $Exp ::= \text{let } Id = Exp \text{ in } Exp$



SYNTAX  $Exp ::= \text{letrec } Id \ Id = Exp \text{ in } Exp$   
           $| \mu Id. Exp$

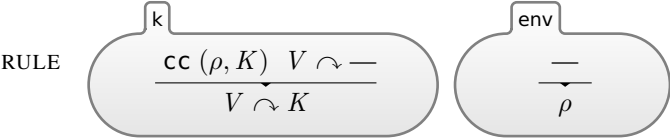
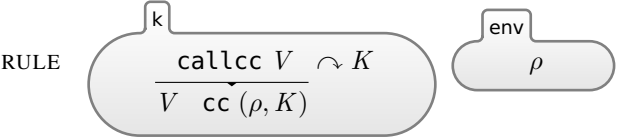


SYNTAX  $Exp ::= \text{muclosure } (Map, Exp)$



SYNTAX  $Exp ::= \text{callcc } Exp \text{ [strict]}$

SYNTAX  $Val ::= \text{cc } (Map, K)$



END MODULE