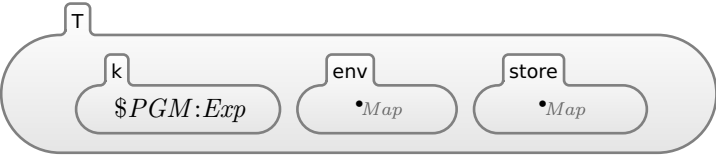


# LAMBDA

MODULE LAMBDA

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

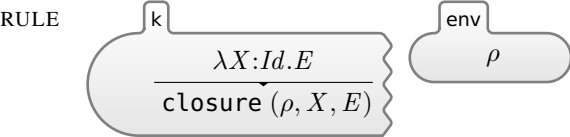
CONFIGURATION:



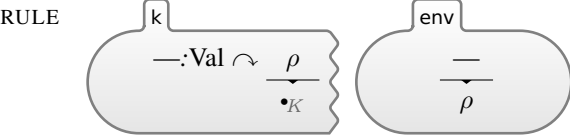
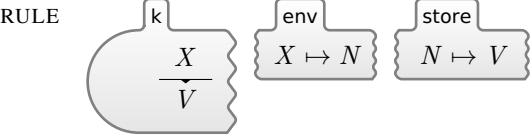
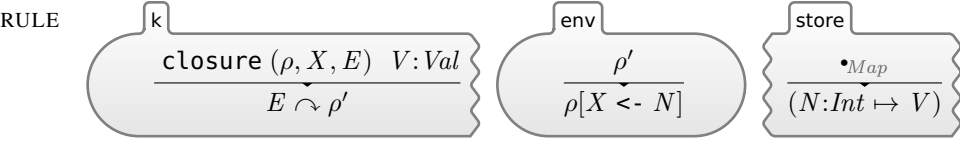
SYNTAX  $Val ::= \text{closure } (Map, Id, Exp)$  [klabel('closure)]

SYNTAX  $Exp ::= Val$

SYNTAX  $KResult ::= Val$



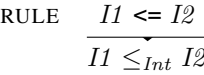
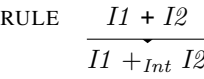
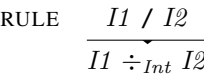
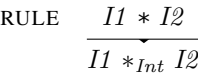
[structural]



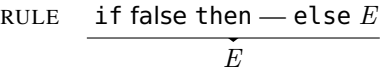
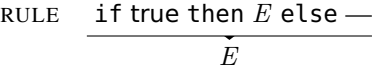
[structural]

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

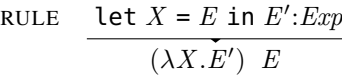
SYNTAX  $Exp ::= Exp * Exp$  [strict]  
|  $Exp / Exp$  [strict]  
|  $Exp + Exp$  [strict]  
|  $Exp <= Exp$  [strict]



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



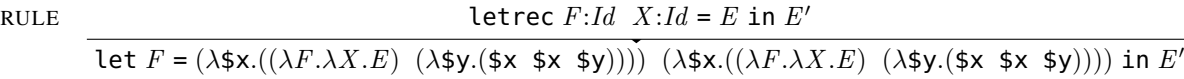
SYNTAX  $Exp ::= \text{let } Id = Exp \text{ in } Exp$



[macro]

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

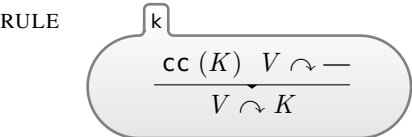
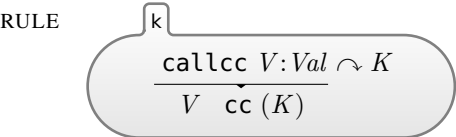
SYNTAX  $Id ::= \$x$   
|  $\$y$



[macro]

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

SYNTAX  $Val ::= \text{cc } (K)$  [klabel('cc)]



END MODULE