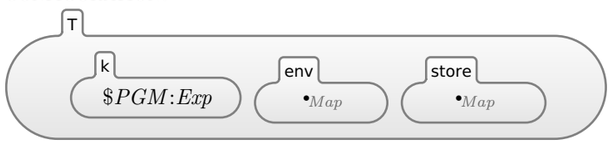
LAMBDA

MODULE LAMBDA

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

CONFIGURATION:

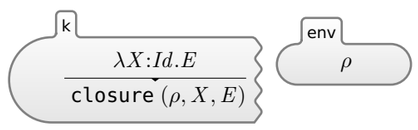


SYNTAX $Val ::= \texttt{closure}\ (Map, Id, Exp)$

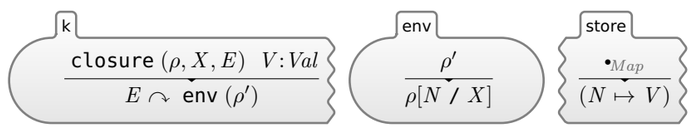
SYNTAX $Exp ::= Val$

SYNTAX $KResult ::= Val$

RULE

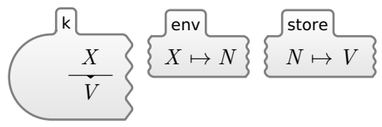


RULE



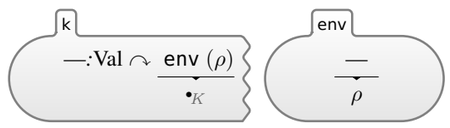
requires $\texttt{fresh}\ (N:Nat)$

RULE



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

RULE



[structural]

SYNTAX $Val ::= Int$
$\mid Bool$

SYNTAX $Exp ::= Exp * Exp$ [strict]
$\mid Exp\ \texttt{/}\ Exp$ [strict]
$\mid Exp + Exp$ [strict]
$\mid Exp\ \texttt{<=}\ Exp$ [strict]

RULE $\dfrac{I1:Int * I2:Int}{I1 *_{Int} I2}$

RULE $\dfrac{I1:Int\ \texttt{/}\ I2:Int}{I1 \div_{Int} I2}$

RULE $\dfrac{I1:Int + I2:Int}{I1 +_{Int} I2}$

RULE $\dfrac{I1:Int\ \texttt{<=}\ I2:Int}{I1 \leq_{Int} I2}$

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

RULE $\dfrac{\texttt{if true then}\ E\ \texttt{else}\ —}{E}$

RULE $\dfrac{\texttt{if false then}\ —\ \texttt{else}\ E}{E}$

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

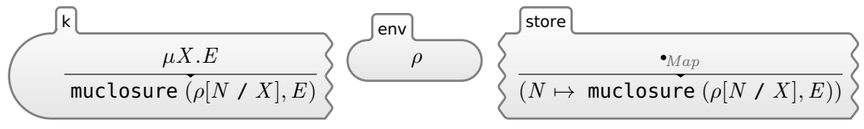
RULE $\dfrac{\texttt{let}\ X\ \texttt{=}\ E\ \texttt{in}\ E':Exp}{(\lambda X.E')\ \ E}$ [macro]

SYNTAX $Exp ::= \texttt{letrec}\ Id\ \ Id\ \texttt{=}\ Exp\ \texttt{in}\ Exp$
$\mid \mu Id.Exp$

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

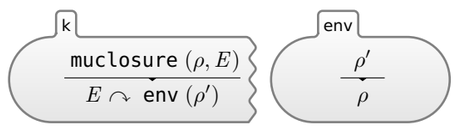
SYNTAX $Exp ::= \texttt{muclosure}\ (Map, Exp)$

RULE



requires $\texttt{fresh}\ (N:Nat)$ [structural]

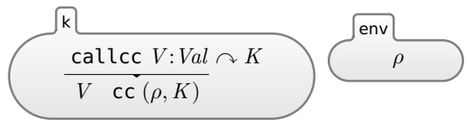
RULE



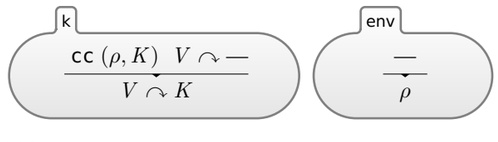
SYNTAX $Exp ::= \texttt{callcc}\ Exp$ [strict]

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

RULE



RULE



END MODULE