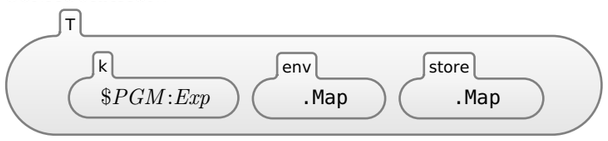
LAMBDA

MODULE LAMBDA

SYNTAX *Exp* ::= *Id*
| *λId.Exp*
| *Exp* *Exp* [strict(strict())]
| (*Exp*) [bracket(bracket())]

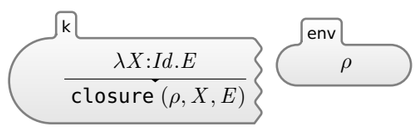
CONFIGURATION:



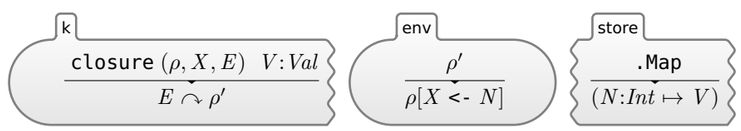
SYNTAX *Val* ::= `closure` (*Map*, *Id*, *Exp*) [klabel(klabel('closure))]

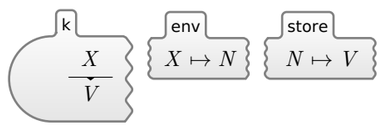
SYNTAX *Exp* ::= *Val*

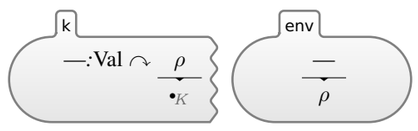
SYNTAX *KResult* ::= *Val*

RULE 

[structural(structural())]

RULE 

RULE 

RULE 

[structural(structural())]

SYNTAX *Val* ::= *Int*
| *Bool*

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

RULE $\frac{I1 * I2}{I1 *_{Int} I2}$

RULE $\frac{I1 \;/\; I2}{I1 \div_{Int} I2}$

RULE $\frac{I1 + I2}{I1 +_{Int} I2}$

RULE $\frac{I1 \;\texttt{<=}\; I2}{I1 \leq_{Int} I2}$

SYNTAX *Exp* ::= `if` *Exp* `then` *Exp* `else` *Exp* [strict(strict(1))]

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

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

SYNTAX *Exp* ::= `let` *Id* = *Exp* `in` *Exp*

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

SYNTAX *Exp* ::= `letrec` *Id* *Id* = *Exp* `in` *Exp*

SYNTAX *Id* ::= `$x`
| `$y`

RULE $\frac{\texttt{letrec } F{:}Id \;\; X{:}Id \texttt{ = } E \texttt{ in } E'}{\texttt{let } F \texttt{ = } (\lambda\texttt{\$x}.((\lambda F.\lambda X.E)\;\; (\lambda\texttt{\$y}.(\texttt{\$x \$x \$y}))))\;\; (\lambda\texttt{\$x}.((\lambda F.\lambda X.E)\;\; (\lambda\texttt{\$y}.(\texttt{\$x \$x \$y}))))\texttt{ in } E'}$ [macro(macro())]

SYNTAX *Exp* ::= `callcc` *Exp* [strict(strict())]

SYNTAX *Val* ::= `cc` (*K*) [klabel(klabel('cc))]

RULE

RULE

END MODULE