LAMBDA

MODULE LAMBDA

SYNTAX *Exp* ::= *Int*
    | *Bool*
    | *Id*
    | (*Exp*) [bracket]
    | *Exp Exp*
    | *Exp* * *Exp*
    | *Exp* / *Exp*
    | *Exp* + *Exp*
    | *Exp* <= *Exp*
    | lambda *Id* . *Exp*
    | if *Exp* then *Exp* else *Exp*
    | let *Id* = *Exp* in *Exp*
    | letrec *Id* *Id* = *Exp* in *Exp*
    | mu *Id* . *Exp*

SYNTAX *Type* ::= int
    | bool
    | *Type* -> *Type*
    | (*Type*) [bracket]

SYNTAX *Exp* ::= *Type*

SYNTAX *KResult* ::= *Type*

CONFIGURATION:
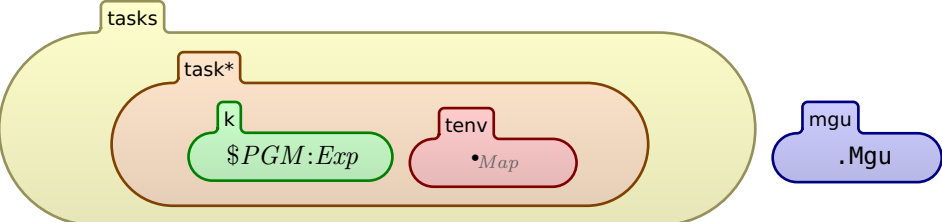
RULE $\dfrac{I:Int}{\texttt{int}}$

RULE $\dfrac{B:Bool}{\texttt{bool}}$

RULE

RULE

RULE

RULE

RULE

RULE

RULE ...requires fresh (*Tx:Type*) $\wedge_{Bool}$ fresh (*Te:Type*)

RULE ...requires fresh (*T2:Type*) $\wedge_{Bool}$ fresh (*T:Type*)

RULE ...requires fresh (*T:Type*)

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

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

RULE ...requires fresh (*T:Type*)

SYNTAX *K* ::= *Exp* = *Exp* [strict]

RULE

RULE

RULE

END MODULE