## **LAMBDA**

```
MODULE LAMBDA
   SYNTAX Exp ::= Int
                          (Exp) [bracket]
                          Exp Exp [strict]
                          Exp * Exp [strict]
                          Exp / Exp [strict]
                          Exp + Exp [strict]
                          Exp <= Exp [strict]</pre>
                          lambda Id . Exp
                          if Exp then Exp else Exp [strict]
                          let Id = Exp in Exp
                          letrec Id Id = Exp in Exp
                          mu Id . Exp
    SYNTAX Type ::= int
                           bool
                          Type \rightarrow Type
                         (Type) [bracket]
    SYNTAX Exp ::= Type
   SYNTAX Variable ::= Id
   SYNTAX KResult ::= Type
   CONFIGURATION:
      PGM:Exp
                                  {}^{ullet}Map
  {\tt RULE} \quad I{:}Int
             int
   RULE B:Bool
              bool
   RULE
                                 X\mapsto T
                    T1\!:\!Type\,*\,T2\!:\!Type
  RULE
            T1 = \text{int} \curvearrowright T2 = \text{int} \curvearrowright \text{int}
                    T1:Type \ / \ T2:Type
   RULE
            T1 = \text{int} \curvearrowright T2 = \text{int} \curvearrowright \text{int}
                   T1:Type + T2:Type
  RULE
            T1 = \text{int} \curvearrowright T2 = \text{int} \curvearrowright \text{int}
                    T1:Type \iff T2:Type
            T1 = \text{int} \curvearrowright T2 = \text{int} \curvearrowright \text{bool}
  RULE
                            \verb|lambda| X: Id . E: Exp|
                                                                                   TEnv
                    E \curvearrowright T: Type \rightarrow \Box \curvearrowright tenv (TEnv)
                                                                             TEnv[X \leftarrow T]
  RULE T2:Type 
ightharpoonup T1:Type 	ext{->} \square
                       T1 -> T2
                  T1:Type T2:Type
  RULE
            T1 = (T2 \rightarrow T: Type) \curvearrowright T
   {\tt RULE} \quad \hbox{if $T$:$ $Type$ then $T1$:$ $Type$ else $T2$:$ $Type$}
                    T = bool \curvearrowright T1 = T2 \curvearrowright T1
  RULE let X = E in E'
            ( \operatorname{lambda} X . E') E
                      letrec F \ X = E \text{ in } E'
  RULE
             let F = mu F . lambda X . E in E'
  RULE
                               \mathsf{mu}\ X{:}Id\ .\ E{:}Exp
                                                                                  TEnv
                   \overline{(T:Type \rightarrow T) \ E \curvearrowright \text{tenv}(TEnv)}
                                                                             \overline{TEnv[X \leftarrow T]}
   SYNTAX KItem ::= Type = Type
  RULE T = T
   SYNTAX KItem ::= tenv (Map) [klabel('tenv)]
  RULE
                    T:Type \curvearrowright \mathsf{tenv}(\mathit{TEnv})
                                                                TEnv
```

[macro]

[macro]

END MODULE