LAMBDA

```
MODULE LAMBDA
   SYNTAX Exp ::= Int
                    Bool
                   (Exp) [bracket( bracket())]

Exp Exp

Exp * Exp
                    Exp / Exp
                    Exp + Exp
                    Exp \le 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 \rightarrow Type
                    (Type) [bracket( bracket())]
   SYNTAX Exp ::= Type
   SYNTAX Variable ::= Id
   SYNTAX KResult ::= Type
  CONFIGURATION:
           tasks
                 stask*
                    PGM:Exp
                                          .Map
  RULE I:Int
          int
  RULE B:Bool
          bool
  RULE
               X:Id
                         X \mapsto T
               E1 * E2
  RULE
                 int
                                                task
                                                    E1 = int
                                                 task
                                                     E2 = int
               E1 / E2
                                                task
                                                    E1 = int
                                                 task
                                                     E2 = int
               E1 + E2
  RULE
                                                    E1 = int
                                                    E2 = int
               E1 <= E2
  RULE
                                                     E1 = int
                                                      E2 = int
                                          TEnv
                  \mathsf{lambda}\,X\, \mathrel{\ldotp\ldotp} E
  RULE
               Tx: Type -> Te: Type
                                                               E = Te
                                                                              TEnv[X \leftarrow Tx]
               E1 E2
                                               task
                                                   E1 = T2 \rightarrow T
                                                task
                                                   E2 = T2:Type
               if E then E1 else E2
  RULE
                       T: Type
                                                               task
                                                                   E = bool
                                                                 task
                                                                     E1 = T
                                                                 task
                                                                     E2 = T
         \mathsf{let}\; X = E \; \mathsf{in}\; E'
  RULE
         \overbrace{(\; \mathsf{lambda}\; X \; . \; E') \;\; E}
                \mathsf{letrec}\; F\;\; X = E\;\mathsf{in}\; E'
  RULE
         TEnv
  RULE
               \operatorname{mu} X \mathrel{.} E
                T: Type
                                                 task
                                                                   tenv
                                                                   TEnv[X \leftarrow T]
                                                     E = T
   SYNTAX KItem ::= Exp = Exp [strict(strict())]
  RULE
               T:Type = T:Type
END MODULE
```

[macro(macro())]

[macro(macro())]