LAMBDA

MODULE LAMBDA

SYNTAX   *Val* ::= *Id*
  | λ*Id*.*Exp* [binder]

SYNTAX   *Exp* ::= *Val*
  | *Exp* *Exp* [strict]
  | (*Exp*) [bracket]

SYNTAX   *KResult* ::= *Val*

RULE $\dfrac{(\lambda X{:}Id.E{:}Exp)\ \ V{:}Val}{E[V \ / \ X]}$

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

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

RULE $\dfrac{I1{:}Int \ast I2{:}Int}{I1 \ast_{Int} I2}$

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

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

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

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

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

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

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

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

SYNTAX   *Exp* ::= letrec *Id* *Id* = *Exp* in *Exp*
  | μ*Id*.*Exp* [binder]

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

RULE $\dfrac{\mu X.E}{E[(\mu X.E) \ / \ X]}$

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

SYNTAX   *Val* ::= cc (*K*)

RULE
$$\dfrac{\text{callcc } V{:}Val \curvearrowright K}{V \ \ \text{cc } (K)} \quad \boxed{k}$$

RULE
$$\dfrac{\text{cc } (K) \ \ V \curvearrowright —}{V \curvearrowright K} \quad \boxed{k}$$

END MODULE