

LAMBDA

MODULE LAMBDA

SYNTAX *Val* ::= *Id*
 | $\lambda Id.Exp$ [binder]

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

SYNTAX *Variable* ::= *Id*

SYNTAX *KResult* ::= *Val*

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

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

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

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

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

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

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

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

RULE
$$\frac{\text{if true then } E \text{ else } \text{---}}{E}$$

RULE
$$\frac{\text{if false then --- else } E}{E}$$

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

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

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

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

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

END MODULE