LAMBDA

MODULE LAMBDA

SYNTAX *Exp* ::= *Int*
| *Bool*
| *Id*
| (*Exp*) [bracket(bracket())]
| *Exp* *Exp* [strict(strict())]
| *Exp* `*` *Exp* [strict(strict())]
| *Exp* `/` *Exp* [strict(strict())]
| *Exp* `+` *Exp* [strict(strict())]
| *Exp* `<=` *Exp* [strict(strict())]
| `lambda` *Id* `.` *Exp*
| `if` *Exp* `then` *Exp* `else` *Exp* [strict(strict())]
| `let` *Id* `=` *Exp* `in` *Exp* [strict(strict(2))]
| `letrec` *Id* *Id* `=` *Exp* `in` *Exp*
| `mu` *Id* `.` *Exp*

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

SYNTAX *Exp* ::= *Type*

SYNTAX *Variable* ::= *Id*

SYNTAX *KResult* ::= *Type*

CONFIGURATION:

k $\$PGM:Exp$ tenv `.Map`

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

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

RULE k $\dfrac{X:Id}{T}$ tenv $X \mapsto T:Type$

RULE $\dfrac{T1:Type \texttt{ * } T2:Type}{T1 \texttt{ = int} \curvearrowright T2 \texttt{ = int} \curvearrowright \texttt{int}}$

RULE $\dfrac{T1:Type \texttt{ / } T2:Type}{T1 \texttt{ = int} \curvearrowright T2 \texttt{ = int} \curvearrowright \texttt{int}}$

RULE $\dfrac{T1:Type \texttt{ + } T2:Type}{T1 \texttt{ = int} \curvearrowright T2 \texttt{ = int} \curvearrowright \texttt{int}}$

RULE $\dfrac{T1:Type \texttt{ <= } T2:Type}{T1 \texttt{ = int} \curvearrowright T2 \texttt{ = int} \curvearrowright \texttt{bool}}$

RULE k $\dfrac{\texttt{lambda } X:Id \texttt{ . } E:Exp}{E \curvearrowright T:Type \texttt{ -> } \square \curvearrowright \texttt{tenv } (TEnv)}$ tenv $\dfrac{TEnv}{TEnv[X \texttt{ <- } T]}$

RULE $\dfrac{T2:Type \curvearrowright T1:Type \texttt{ -> } \square}{T1 \texttt{ -> } T2}$

RULE $\dfrac{T1:Type \;\; T2:Type}{T1 \texttt{ = } (T2 \texttt{ -> } T:Type) \curvearrowright T}$

RULE $\dfrac{\texttt{if } T:Type \texttt{ then } T1:Type \texttt{ else } T2:Type}{T \texttt{ = bool} \curvearrowright T1 \texttt{ = } T2 \curvearrowright T1}$

SYNTAX *TypeSchema* ::= (`forall` *Set*)*Type*

SYNTAX *Type* ::= *MetaVariable*

RULE k $\dfrac{\texttt{let } X \texttt{ = } T:Type \texttt{ in } E}{E \curvearrowright \texttt{tenv } (TEnv)}$ tenv $\dfrac{TEnv}{TEnv[X \texttt{ <- } (\texttt{ forall \#metaVariables } (T) -_{Set} \texttt{ \#metaVariables } (\texttt{ tenv } (TEnv))) \texttt{ \#freezeVariables } (T, \texttt{ tenv } (TEnv))]}$

RULE k $\dfrac{X:Id}{\texttt{\#rename } (T, Tvs)}$ tenv $X \mapsto (\texttt{ forall } Tvs) T$

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

RULE k $\dfrac{\texttt{mu } X:Id \texttt{ . } E:Exp}{(T:Type \texttt{ -> } T) \;\; E \curvearrowright \texttt{tenv } (TEnv)}$ tenv $\dfrac{TEnv}{TEnv[X \texttt{ <- } T]}$

SYNTAX *KItem* ::= *Type* `=` *Type*

RULE $\dfrac{T \texttt{ = } T}{\bullet_K}$

SYNTAX *KItem* ::= `tenv` (*Map*) [klabel(klabel('tenv))]

RULE k $\dfrac{T:Type \curvearrowright \texttt{tenv } (TEnv)}{\bullet_K}$ tenv $\dfrac{\text{—}}{TEnv}$

END MODULE