# LAMBDA

MODULE LAMBDA

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

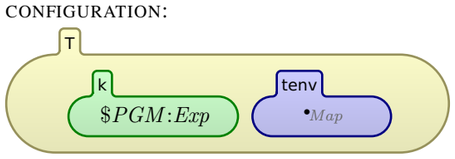
SYNTAX *Exp* ::= *Id*
| `lambda` *Id* `:` *Type* `.` *Exp*
| *Exp* *Exp* [strict]
| (*Exp*) [bracket]

SYNTAX *Exp* ::= *Type*

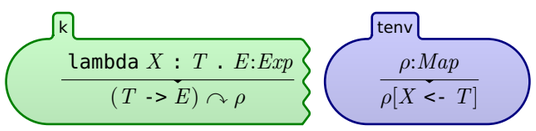
SYNTAX *Variable* ::= *Id*

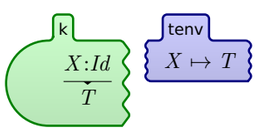
SYNTAX *KResult* ::= *Type*

CONFIGURATION:




SYNTAX *Exp* ::= *Exp* `->` *Exp* [strict]

RULE 


RULE 


RULE $\dfrac{(T1 \texttt{ -> } T2)\ \ T1}{T2}$

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

RULE $\dfrac{\text{—:Int}}{\texttt{int}}$

RULE $\dfrac{\text{—:Bool}}{\texttt{bool}}$

RULE $\dfrac{\texttt{int * int}}{\texttt{int}}$

RULE $\dfrac{\texttt{int / int}}{\texttt{int}}$

RULE $\dfrac{\texttt{int + int}}{\texttt{int}}$

RULE $\dfrac{\texttt{int <= int}}{\texttt{bool}}$

SYNTAX *Exp* ::= `if` *Exp* `then` *Exp* `else` *Exp* [strict]

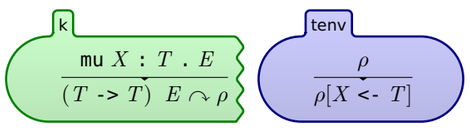
RULE $\dfrac{\texttt{if bool then } T:Type \texttt{ else } T}{T}$

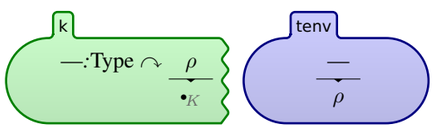
SYNTAX *Exp* ::= `let` *Id* `:` *Type* `=` *Exp* `in` *Exp*

RULE $\dfrac{\texttt{let } X \texttt{ : } T \texttt{ = } E \texttt{ in } E'}{(\texttt{ lambda } X \texttt{ : } T \texttt{ . } E')\ \ E}$ [macro]

SYNTAX *Exp* ::= `letrec` *Id* `:` *Type* *Id* `:` *Type* `=` *Exp* `in` *Exp*
| `mu` *Id* `:` *Type* `.` *Exp*

RULE $\dfrac{\texttt{letrec } F \texttt{ : } T1 \ \ X \texttt{ : } T2 \texttt{ = } E \texttt{ in } E'}{\texttt{let } F \texttt{ : } T1 \texttt{ = mu } F \texttt{ : } T1 \texttt{ . lambda } X \texttt{ : } T2 \texttt{ . } E \texttt{ in } E'}$ [macro]

RULE 


RULE 


END MODULE