LAMBDA

MODULE LAMBDA

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

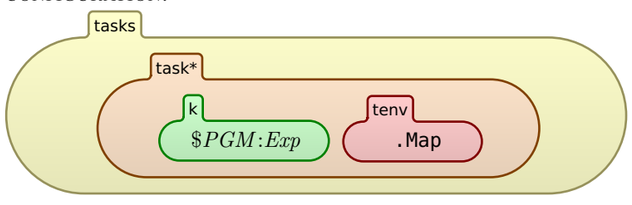
SYNTAX *Type* ::= `int`
| `bool`
| *Type* `->` *Type*
| (*Type*) [bracket(bracket())]

SYNTAX *Exp* ::= *Type*

SYNTAX *Variable* ::= *Id*

SYNTAX *KResult* ::= *Type*

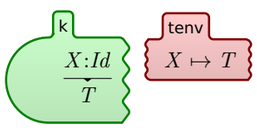
CONFIGURATION:



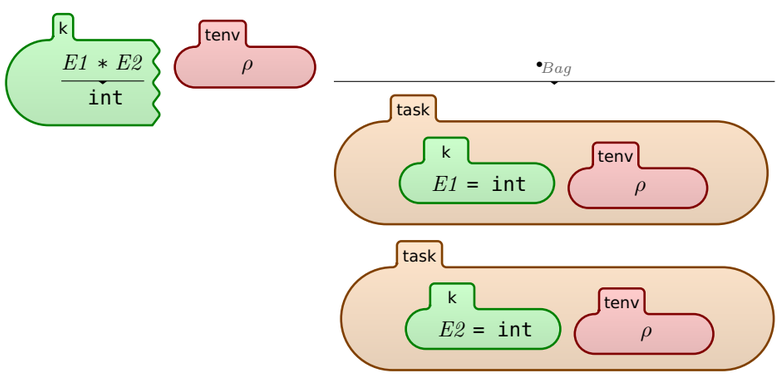
RULE $\frac{I:Int}{\texttt{int}}$

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

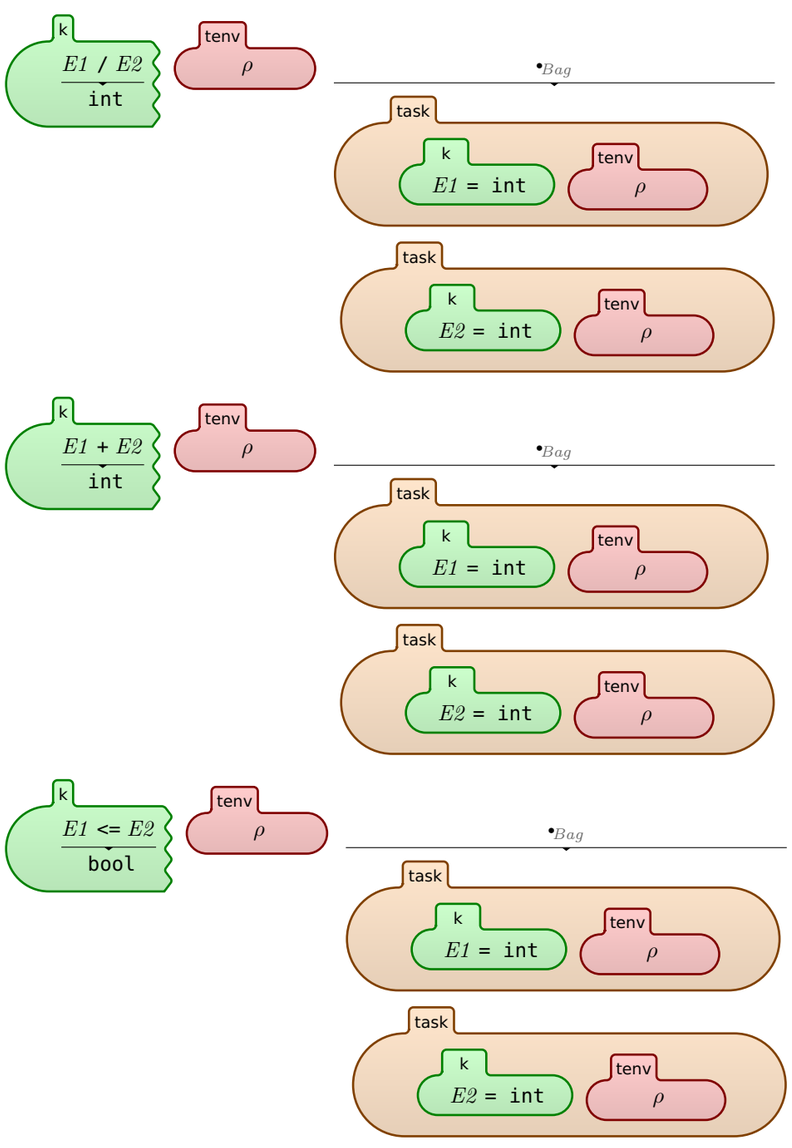
RULE



RULE



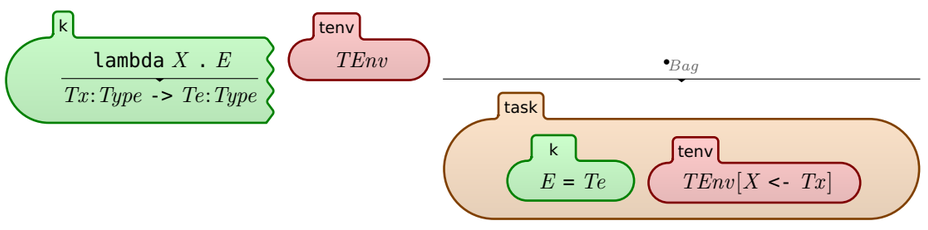
RULE



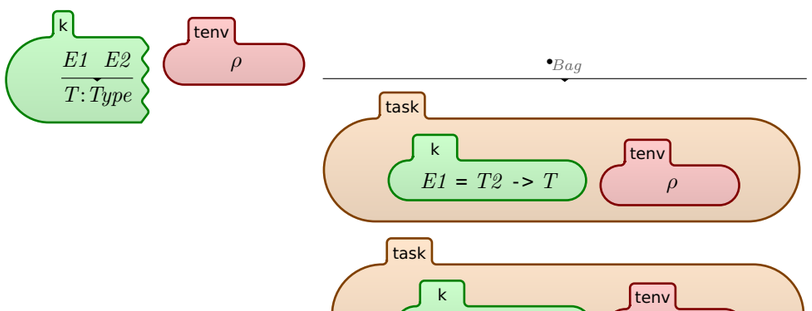
RULE

RULE

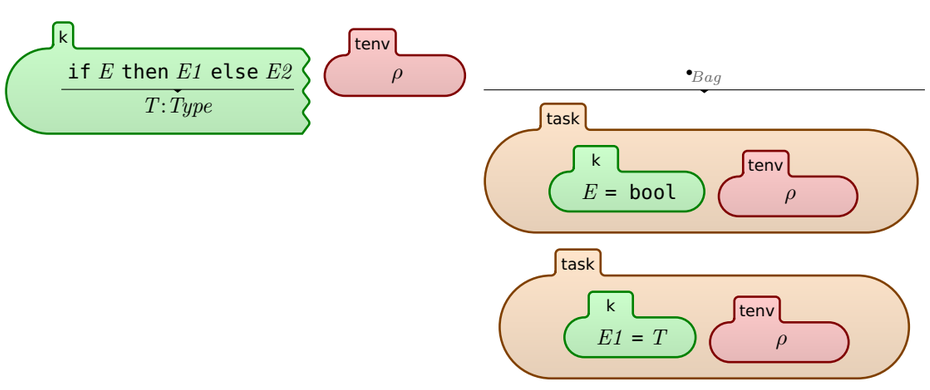
RULE



RULE



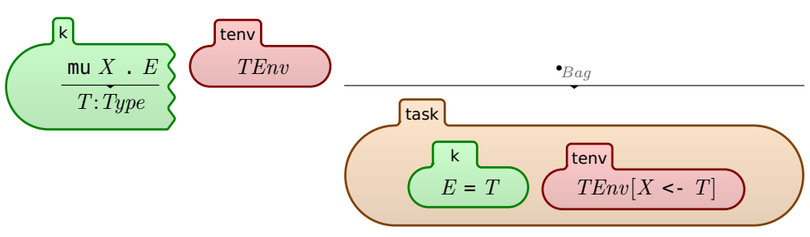
RULE



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

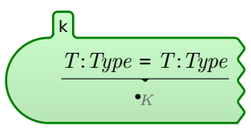
RULE $\frac{\texttt{letrec}\ F\ \ X\ \texttt{=}\ E\ \texttt{in}\ E'}{\texttt{let}\ F\ \texttt{=}\ \texttt{mu}\ F\ \texttt{.}\ \ \texttt{lambda}\ X\ \texttt{.}\ E\ \texttt{in}\ E'}$ [macro(macro())]

RULE

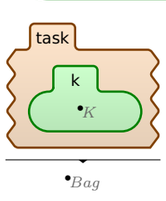


SYNTAX *KItem* ::= *Exp* = *Exp* [strict(strict())]

RULE



RULE



END MODULE