

# IMP

MODULE IMP-SYNTAX

```
SYNTAX  AExp ::= Int
          | String
          | Id
          | ++ Id
          | read ()
          | AExp / AExp [strict, division]
          | AExp + AExp [strict]
          | (AExp) [bracket]

SYNTAX  BExp ::= Bool
          | AExp ≤ AExp [seqstrict]
          | ! BExp [strict]
          | BExp && BExp [strict(1)]
          | (BExp) [bracket]

SYNTAX  Block ::= {}
          | { Stmt }

SYNTAX  Stmt ::= Block
          | Id = AExp ; [strict(2)]
          | if (BExp) Block else Block [strict(1)]
          | while (BExp) Block
          | int Ids ;
          | print (AExps) ; [strict]
          | halt ;
          | spawn Stmt
          | Stmt Stmt

SYNTAX  Ids ::= List{ Id, “,” } [strict]

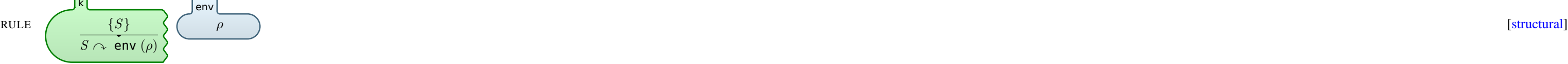
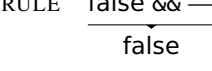
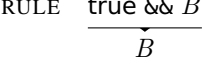
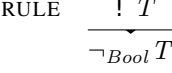
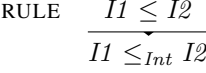
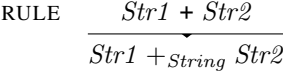
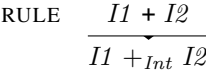
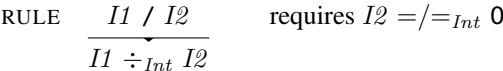
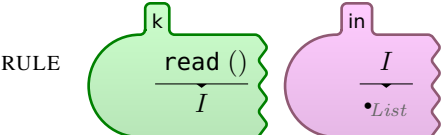
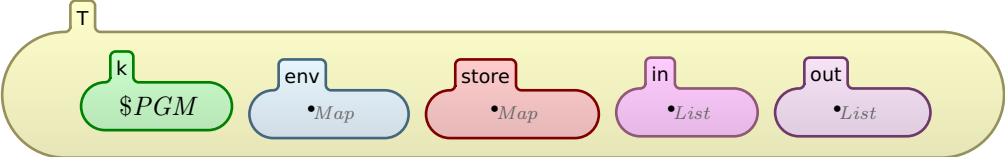
SYNTAX  AExps ::= List{ AExp, “,” } [strict]
```

END MODULE

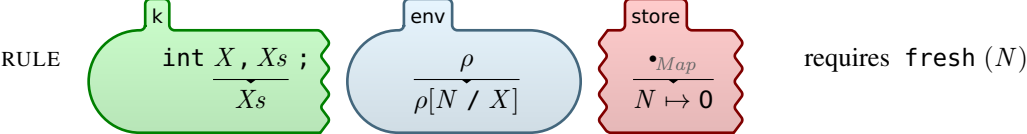
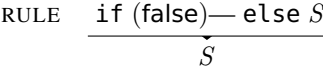
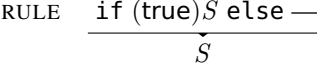
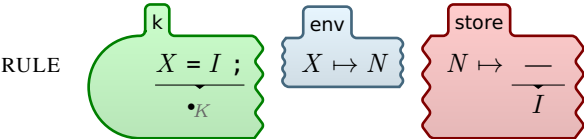
MODULE IMP

```
SYNTAX  KResult ::= Int
          | Bool
          | String
```

CONFIGURATION:

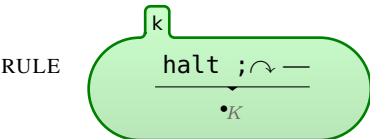
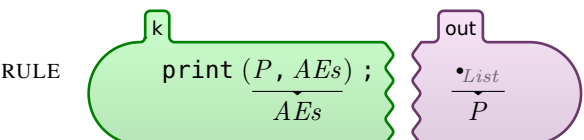


SYNTAX K ::= env (Map)



SYNTAX Printable ::= Int  
                  | String

SYNTAX AExp ::= Printable



END MODULE