IMP

MODULE IMP-SYNTAX

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

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

SYNTAX *Block* ::= {}
| {*Stmt*}

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

SYNTAX *Ids* ::= *List*{*Id*, ",", "}

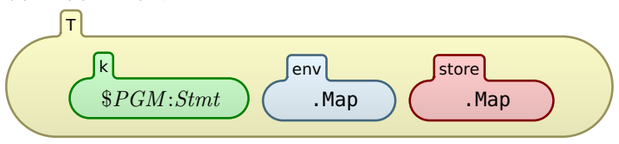
SYNTAX *AExps* ::= *List*{*AExp*, ",", "} [seqstrict(seqstrict())]

END MODULE

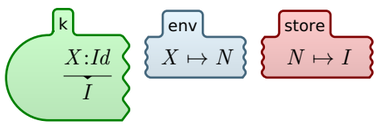
MODULE IMP

SYNTAX *KResult* ::= *Int*
| *Bool*

CONFIGURATION:



RULE



RULE $\frac{I1 \texttt{ / } I2}{I1 \div_{Int} I2}$ requires $I2 =/=_{Int} 0$

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

RULE $\frac{I1 \leq I2}{I1 \leq_{Int} I2}$

RULE $\frac{\texttt{!}\ T}{\neg_{Bool} T}$

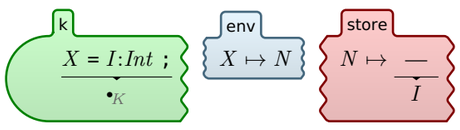
RULE $\frac{\texttt{true \&\& } B}{B}$

RULE $\frac{\texttt{false \&\& } __}{\texttt{false}}$

RULE $\frac{\{\}}{\bullet_K}$ [structural(structural())]

RULE $\frac{\{S\}}{S}$ [structural(structural())]

RULE



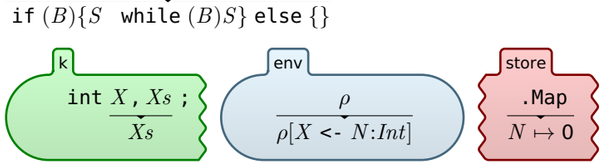
RULE $\frac{S1:Stmt \quad S2:Stmt}{S1 \curvearrowright S2}$ [structural(structural())]

RULE $\frac{\texttt{if (true)}S \texttt{ else } __}{S}$

RULE $\frac{\texttt{if (false)}__ \texttt{ else } S}{S}$

RULE $\frac{\texttt{while } (B)S}{\texttt{if } (B)\{S \quad \texttt{while } (B)S\} \texttt{ else } \{\}}$ [structural(structural())]

RULE



RULE $\frac{\texttt{int } \bullet_{Ids} \texttt{ ;}}{\bullet_K}$ [structural(structural())]

END MODULE