IMP

MODULE IMP-SYNTAX

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

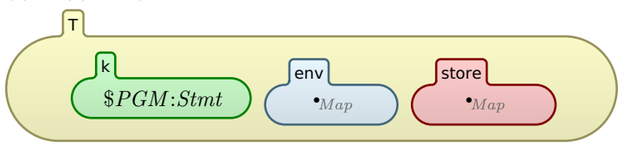
SYNTAX *AExps* ::= *List*{*AExp*,“,”} [strict]

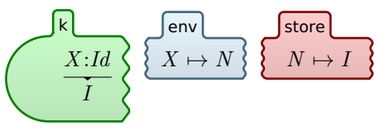
END MODULE

MODULE IMP

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

CONFIGURATION:



RULE 

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

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

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

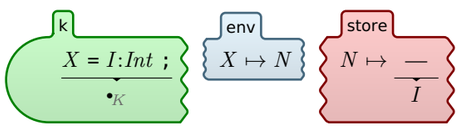
RULE $\frac{\texttt{!}\ T{:}Bool}{\neg_{Bool} T}$

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

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

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

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

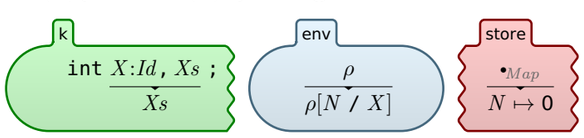
RULE 

RULE $\frac{S1 \quad S2}{S1 \curvearrowright S2}$ [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]

RULE 

requires `fresh` (*N*:*Nat*)

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

END MODULE