COMPETITION

MODULE COMPETITION-SYN

SYNTAX *Decl* ::= `var` *Ids* `;`
| `mutex` *Ints* `;`

SYNTAX *Exp* ::= *Int*
| *Id*
| *Bool*
| `(` *Exp* `)` [bracket]
| `-` *Exp* [strict]
| `read` `(` `)`
| *Exp* `*` *Exp* [strict]
| *Exp* `/` *Exp* [strict]
| *Exp* `%` *Exp* [strict]
| *Exp* `+` *Exp* [strict]
| *Exp* `-` *Exp* [strict]
| *Exp* `<` *Exp* [strict]
| *Exp* `<=` *Exp* [strict]
| *Exp* `>` *Exp* [strict]
| *Exp* `>=` *Exp* [strict]
| *Exp* `==` *Exp* [strict]
| *Exp* `!=` *Exp* [strict]
| `not` *Exp* [strict]
| *Exp* `&&` *Exp* [strict(1)]
| *Exp* `||` *Exp* [strict(1)]
| `execute` *Block*

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

SYNTAX *Exps* ::= $List\{Exp,","\}$ [strict]

SYNTAX *Ints* ::= $List\{Int,","\}$

SYNTAX *Block* ::= `{` `}`
| `{` *Stmts* `}`

SYNTAX *Stmt* ::= *Decl*
| *Block*
| *Exp* `;` [strict]
| *Id* `=` *Exp* `;` [strict(2)]
| `if` `(` *Exp* `)` *Block* `else` *Block* [avoid, strict(1)]
| `if` `(` *Exp* `)` *Block*
| `while` `(` *Exp* `)` *Block*
| `return` *Exp* `;` [strict]
| `print` `(` *Exp* `)` `;` [strict]
| `lock` *Int* `;` [strict]
| `unlock` *Int* `;` [strict]

SYNTAX *Stmts* ::= *Stmt*
| *Stmts* *Stmts*

RULE $\dfrac{\texttt{if}\ (E)\,S}{\texttt{if}\ (E)\,S\ \texttt{else}\ \{\}}$ [macro]

END MODULE

MODULE COMPETITION

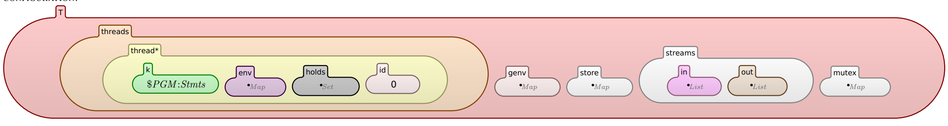
SYNTAX *Val* ::= *Int*
| *Bool*

SYNTAX *Vals* ::= $List\{Val,","\}$

SYNTAX *Exp* ::= *Val*

SYNTAX *KResult* ::= *Val*

CONFIGURATION:



RULE $\dfrac{\texttt{not}\ T{:}Bool}{\neg_{Bool}(T)}$

RULE $\dfrac{\texttt{true}\ \&\&\ E}{E}$

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

RULE $\dfrac{\texttt{true}\ ||\ \_\_}{\texttt{true}}$

RULE $\dfrac{\texttt{false}\ ||\ E}{E}$

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

RULE $\dfrac{I1\ \texttt{-}\ I2}{I1\ -_{Int}\ I2}$

RULE $\dfrac{I1\ \texttt{*}\ I2}{I1\ *_{Int}\ I2}$

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

RULE $\dfrac{I1\ \texttt{\%}\ I2}{I1\ \%_{Int}\ I2}$ requires $I2 =/=_{Int} 0$

RULE $\dfrac{\texttt{-}\ I}{0\ -_{Int}\ I}$

RULE $\dfrac{I1\ \texttt{<}\ I2}{I1\ <_{Int}\ I2}$

RULE $\dfrac{I1\ \texttt{<=}\ I2}{I1\ \leq_{Int}\ I2}$

RULE $\dfrac{I1\ \texttt{>}\ I2}{I1\ >_{Int}\ I2}$

RULE $\dfrac{I1\ \texttt{>=}\ I2}{I1\ \geq_{Int}\ I2}$

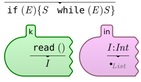
RULE $\dfrac{V1{:}Val\ \texttt{==}\ V2{:}Val}{V1\ =_K\ V2}$

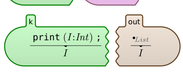
RULE $\dfrac{V1{:}Val\ \texttt{!=}\ V2{:}Val}{V1\ \neq_K\ V2}$

RULE $\dfrac{\texttt{if}\ (\texttt{true})S\ \texttt{else}\ \_\_}{S}$

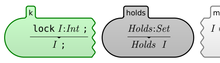
RULE $\dfrac{\texttt{if}\ (\texttt{false})\_\_\ \texttt{else}\ S}{S}$

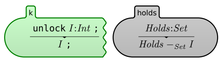
RULE $\dfrac{\texttt{while}\ (E)S}{\texttt{if}\ (E)\{S\ \ \texttt{while}\ (E)S\}}$ [structural]

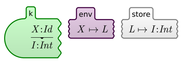
RULE  [read]

RULE  [print]

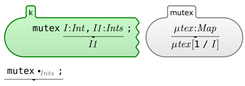
RULE $\dfrac{\_\_{:}\mathrm{Val}\ ;}{\bullet_K}$

RULE  requires $\neg_{Bool}(I\ \texttt{in}\ Holds)$

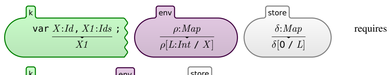
RULE  requires $(I\ \texttt{in}\ Holds)$

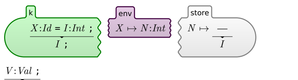
RULE 

RULE  requires $\texttt{fresh}\ (T{:}Int)$

RULE 

RULE $\dfrac{\texttt{mutex}\ \bullet_{Ints}\ \texttt{;}}{\bullet_K}$

RULE  requires $\texttt{fresh}\ (L{:}Int)$

RULE 

RULE $\dfrac{V{:}Val\ \texttt{;}}{\bullet_K}$ [structural]

RULE $\dfrac{\texttt{var}\ \bullet_{Ids}\ \texttt{;}}{\bullet_K}$ [structural]

RULE $\dfrac{S1{:}Stmts\ \ S2{:}Stmts}{S1\curvearrowright S2}$ [structural]

RULE $\dfrac{\{S{:}Stmts\}}{S}$

END MODULE