

# AST Grammar

December 31, 2015

## 1 Common

$\langle comment \rangle ::= \text{NullComment} \mid (\langle comment \rangle)$

$\langle clock \rangle ::= () \mid \text{ident} \mid (\langle clock \rangle)$

$\langle clocks \rangle ::= (\langle clock \rangle [\langle clock \rangle]^*) \mid ()$

$\langle integer \rangle ::= [-+]?(0|[1-9][0-9]^*)$

$\langle float \rangle ::= [-+]?(0|[1-9][0-9]^*)(\.[0-9]^*[1-9])?$

$\langle ident \rangle ::= [\text{a-zA-Z\_}][\text{a-zA-Z0-9\_}]^*$

## 2 Program

$\langle topLevel \rangle ::= \text{TopLevel}(\langle mainBlk \rangle, \langle programBlk \rangle)$

$\langle mainBlk \rangle ::= \text{main}(\langle ident \rangle)$

$\langle programBlk \rangle ::= \text{program}(\langle stmtBlk \rangle [, \langle stmtBlk \rangle]^*)$

$\langle stmtBlk \rangle ::= \langle typeBlk \rangle \mid \langle constBlk \rangle \mid \langle nodeBlk \rangle$

## 3 Type Block

$\langle typeBlk \rangle ::= \text{type\_block}(\langle typeStmt \rangle [, \langle typeStmt \rangle]^*)$

$\langle typeStmt \rangle ::= \text{type}(\langle ident \rangle, \langle type \rangle, \langle comment \rangle)$

$\langle type \rangle ::= \text{bool} \mid \text{short} \mid \text{ushort} \mid \text{int} \mid \text{uint} \mid \text{float} \mid \text{real} \mid \text{char} \mid \langle construct \rangle \mid \langle constructEnum \rangle$   
 $\mid \langle array \rangle \mid \langle typename \rangle \mid (\langle type \rangle)$

$\langle types \rangle ::= (\langle type \rangle, [\langle type \rangle]^*) \mid ()$

$\langle construct \rangle ::= \text{construct}(\langle field \rangle [, \langle field \rangle]^*)$

$\langle field \rangle ::= \text{field}(\langle ident \rangle, \langle type \rangle)$

$\langle constructEnum \rangle ::= \text{construct\_enum}(\langle ident \rangle [, \langle ident \rangle]^*)$

$\langle array \rangle ::= \text{array}(\langle type \rangle, \langle intValue \rangle)$

$\langle intValue \rangle ::= \text{INT}(\langle integer \rangle)$

$\langle typename \rangle ::= \text{typename}(\langle ident \rangle)$

## 4 Const Block

$\langle constBlk \rangle ::= \text{const\_block}(\langle constStmt \rangle [, \langle constStmt \rangle]^*)$

$\langle constStmt \rangle ::= \text{const}(\langle ident \rangle, \langle type \rangle, \langle value \rangle, \langle comment \rangle)$

$\langle value \rangle ::= \text{ID}(\langle ident \rangle, \langle type \rangle) \mid \text{BOOL}(\langle bool \rangle) \mid \text{CHAR}(\langle integer \rangle) \mid \text{SHORT}(\langle integer \rangle) \mid \text{USHORT}(\langle integer \rangle) \mid \text{INT}(\langle integer \rangle) \mid \text{UINT}(\langle integer \rangle) \mid \text{FLOAT}(\langle float \rangle) \mid \text{REAL}(\langle float \rangle) \mid \langle constructValue \rangle \mid \langle constructArrValue \rangle$

$\langle bool \rangle ::= \text{true} \mid \text{false}$

$\langle constructValue \rangle ::= \text{construct}(\langle fieldValue \rangle [, \langle fieldValue \rangle]^*)$

$\langle fieldValBlk \rangle ::= \text{label\_const}(\langle ident \rangle, \langle value \rangle)$

$\langle constructArrValue \rangle ::= \text{construct\_array}(\langle value \rangle [, \langle value \rangle]^*)$

## 5 Node Block

$\langle nodeBlk \rangle ::= \text{node}(\langle kind \rangle, \langle guid \rangle, \langle ident \rangle, \langle comment \rangle, \langle paramBlk \rangle, \langle returnBlk \rangle, \langle bodyBlk \rangle)$

$\langle kind \rangle ::= \text{node} \mid \text{function}$

$\langle guid \rangle ::= [0-9a-z]^*$

$\langle paramBlk \rangle ::= \text{params}([ \langle varDeclsStmt \rangle [, \langle varDeclsStmt \rangle]^* ])$

$\langle returnBlk \rangle ::= \text{returns}([ \langle varDeclsStmt \rangle [, \langle varDeclsStmt \rangle]^* ])$

$\langle varDeclsStmt \rangle ::= \text{var\_decls}(\text{vars}(\langle ident \rangle [, \langle ident \rangle]^*), \langle type \rangle, \langle comment \rangle)$

$\langle bodyBlk \rangle ::= \text{body}([ \langle localVarsBlk \rangle, ] \langle assignBlk \rangle)$

$\langle localVarsBlk \rangle ::= \text{localvars}(\langle varDeclsStmt \rangle [, \langle varDeclsStmt \rangle]^*)$

$\langle assignBlk \rangle ::= [ \langle assignStmt \rangle [, \langle assignStmt \rangle]^* ]$

$\langle assignStmt \rangle ::= \text{=(lvalue}(\langle lhs \rangle \text{ [, } \langle lhs \rangle \text{)]}^*, \langle expr \rangle, \langle guidop \rangle, \langle guidVal \rangle, \langle imported \rangle, \langle importCode \rangle)$

$\langle lhs \rangle ::= \text{anonymous\_id} \mid \text{ID}(\langle ident \rangle, \langle type \rangle, \langle clock \rangle)$

$\langle guidop \rangle ::= \langle ident \rangle \mid \text{NOCALL}$

$\langle guidVal \rangle ::= \langle guid \rangle \mid \text{NOGUID}$

$\langle imported \rangle ::= \text{NOIMPORT} \mid \text{IMPORTED}$

$\langle importCode \rangle ::= \langle integer \rangle$

$\langle expr \rangle ::= \langle atomExpr \rangle \mid \langle binopExpr \rangle \mid \langle unopExpr \rangle \mid \langle ifExpr \rangle \mid \langle switchExpr \rangle \mid \langle tempoPreExpr \rangle$   
 $\mid \langle tempoArrowExpr \rangle \mid \langle tempoFbyExpr \rangle \mid \langle fieldAccessExpr \rangle \mid \langle constructExpr \rangle \mid$   
 $\langle constructArrExpr \rangle \mid \langle mixedConstructorExpr \rangle \mid \langle arrDimExpr \rangle \mid \langle arrIdxExpr \rangle \mid$   
 $\langle arrSliceExpr \rangle \mid \langle listExpr \rangle \mid \langle applyExpr \rangle \mid \langle dynamicProjExpr \rangle$

$\langle atomExpr \rangle ::= \text{ID}(\langle ident \rangle, \langle type \rangle, \langle clock \rangle) \mid \text{ID}(\langle ident \rangle) \mid \text{BOOL}(\langle bool \rangle) \mid \text{CHAR}(\langle integer \rangle)$   
 $\mid \text{SHORT}(\langle integer \rangle) \mid \text{USHORT}(\langle integer \rangle) \mid \text{INT}(\langle integer \rangle) \mid \text{UINT}(\langle integer \rangle) \mid$   
 $\text{FLOAT}(\langle float \rangle) \mid \text{REAL}(\langle float \rangle) \mid$

$\langle binopExpr \rangle ::= \langle binop \rangle(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle expr \rangle)$

$\langle binop \rangle ::= \text{binop\_add} \mid \text{binop\_subtract} \mid \text{binop\_multiply} \mid \text{binop\_divide} \mid \text{binop\_}$   
 $\text{div} \mid \text{binop\_mod} \mid \text{binop\_and} \mid \text{binop\_or} \mid \text{binop\_xor} \mid \text{binop\_gt} \mid \text{binop\_lt} \mid$   
 $\text{binop\_ge} \mid \text{binop\_le} \mid \text{binop\_eq} \mid \text{binop\_neq}$

$\langle unopExpr \rangle ::= \langle unop \rangle(\langle type \rangle, \langle clock \rangle, \langle expr \rangle)$

$\langle unop \rangle ::= \text{unop\_shortcast} \mid \text{unop\_intcast} \mid \text{unop\_floatcast} \mid \text{unop\_realcast} \mid \text{unop\_}$   
 $\text{not} \mid \text{unop\_pos} \mid \text{unop\_neg}$

$\langle ifExpr \rangle ::= \text{if\_expr}(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle expr \rangle, \langle expr \rangle)$

$\langle switchExpr \rangle ::= \text{switch\_expr}(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle caseStmt \rangle \text{ [, } \langle caseStmt \rangle]^*)$

$\langle caseStmt \rangle ::= \text{case}(\langle value \rangle, \langle expr \rangle) \mid \text{case}(\text{pattern\_any}, \langle expr \rangle)$

$\langle tempoPreExpr \rangle ::= \text{tempo\_pre}(\langle types \rangle, \langle clocks \rangle, \langle expr \rangle)$

$\langle tempoArrowExpr \rangle ::= \text{tempo\_arrow}(\langle types \rangle, \langle clocks \rangle, \langle expr \rangle, \langle expr \rangle)$

$\langle tempoFbyExpr \rangle ::= \text{tempo\_fby}(\langle types \rangle, \langle clocks \rangle, \langle listExpr \rangle, \langle expr \rangle, \langle listExpr \rangle)$

$\langle constructExpr \rangle ::= \text{construct}(\langle type \rangle, \langle clock \rangle, [\langle labelExpr \rangle \text{ ,}]^+)$

$\langle labelExpr \rangle ::= \text{label\_expr}(\langle ident \rangle, \langle expr \rangle)$

$\langle constructArrExpr \rangle ::= \text{construct\_array}(\langle type \rangle, \langle clock \rangle, \langle listExpr \rangle)$

$\langle mixedConstructorExpr \rangle ::= mixed\_constructor(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle labelIdxList \rangle, \langle expr \rangle)$   
 $\langle labelIdxList \rangle ::= (\langle labelIdx \rangle [, labelIdx]^*)$   
 $\langle labelIdx \rangle ::= struct\_item(\langle ident \rangle) \mid \langle expr \rangle$   
 $\langle fieldAccessExpr \rangle ::= field\_access(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle ident \rangle)$   
 $\langle arrDimExpr \rangle ::= array\_dim(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle intValue \rangle)$   
 $\langle arrIdxExpr \rangle ::= array\_index(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle intValue \rangle)$   
 $\langle arrSliceExpr \rangle ::= array\_slice(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, \langle expr \rangle, \langle expr \rangle)$   
 $\langle listExpr \rangle ::= list\_expr(\langle expr \rangle [, \langle expr \rangle]^*) \mid list\_expr()$   
 $\langle dynamicProjExpr \rangle ::= dynamic\_project(\langle type \rangle, \langle clock \rangle, \langle expr \rangle, (\langle expr \rangle [, \langle expr \rangle]^*), \langle expr \rangle)$   
 $\langle applyExpr \rangle ::= apply\_expr(\langle types \rangle, \langle clocks \rangle, \langle applyBlk \rangle, \langle listExpr \rangle)$   
 $\langle applyBlk \rangle ::= \langle makeStmt \rangle \mid \langle flattenStmt \rangle \mid \langle highOrderStmt \rangle \mid \langle prefixStmt \rangle \mid \langle mapwDefaultStmt \rangle \mid \langle mapwiDefaultStmt \rangle \mid \langle foldwIfStmt \rangle \mid \langle foldwiStmt \rangle$   
 $\langle makeStmt \rangle ::= make(\langle ident \rangle, \langle type \rangle)$   
 $\langle flattenStmt \rangle ::= flatten(\langle ident \rangle, \langle type \rangle)$   
 $\langle highOrderStmt \rangle ::= high\_order(\langle highOrderOp \rangle, \langle prefixStmt \rangle, \langle intValue \rangle)$   
 $\langle prefixStmt \rangle ::= prefix(\langle ident \rangle, param\_types \langle types \rangle, ret\_types \langle types \rangle) \mid prefix(\langle prefixBinOp \rangle) \mid prefix(\langle prefixUnOp \rangle)$   
 $\langle highOrderOp \rangle ::= highorder\_map \mid highorder\_fold \mid highorder\_mapfold \mid highorder\_mapi \mid highorder\_foldi$   
 $\langle prefixUnOp \rangle ::= short\$ \mid int\$ \mid float\$ \mid real\$ \mid not\$ \mid +\$ \mid -\$$   
 $\langle prefixBinOp \rangle ::= \$+\$ \mid \$-\$ \mid \$*\$ \mid \$/\$ \mid \$div\$ \mid \$mod\$ \mid \$and\$ \mid \$or\$ \mid \$xor\$ \mid \$=\$ \mid \$\>\$ \mid \$>=\$ \mid \$<\$ \mid \$<=\$$   
 $\langle mapwDefaultStmt \rangle ::= mapw\_default(\langle prefix \rangle, \langle intValue \rangle, \langle expr \rangle, \langle expr \rangle, )$   
 $\langle mapwiDefaultStmt \rangle ::= mapwi\_default(\langle prefix \rangle, \langle intValue \rangle, \langle expr \rangle, \langle expr \rangle, )$   
 $\langle foldwIfStmt \rangle ::= foldw\_if(\langle prefix \rangle, \langle intValue \rangle, \langle expr \rangle)$   
 $\langle foldwiStmt \rangle ::= foldwi(\langle prefix \rangle, \langle intValue \rangle, \langle expr \rangle)$