Lustre Grammar

January 1, 2016

1 Common

⟨*char*⟩ ::= all printable character

⟨*bool*⟩ ::= true | false

⟨*integer*⟩ ::= [-+]?(0|[1-9][0-9]*)

⟨*float*⟩ ::= [-+]?(0|[1-9][0-9]*)(.[0-9]*[1-9])?

⟨*ident*⟩ ::= [a-zA-Z_][a-zA-Z0-9_]*

⟨*clock*⟩ ::= (⟨*ident*⟩ [, ident]*)

2 Program

⟨*program*⟩ ::= ⟨*nodeBlk*⟩*

⟨*nodeBlk*⟩ ::= ⟨*typeBlk*⟩ | ⟨*constBlk*⟩ | ⟨*funcBlk*⟩

3 Type

⟨*typeBlk*⟩ ::= type ⟨*typeStmt*⟩*

⟨*typeStmt*⟩ ::= [⟨*modifier*⟩] ⟨*ident*⟩ = ⟨*type*⟩;

⟨*modifier*⟩ ::= private | public | protected

⟨*type*⟩ ::= ⟨*atomType*⟩ | ⟨*struct*⟩ | ⟨*type*⟩ ^ ⟨*expr*⟩la

⟨*atomType*⟩ ::= char | bool | short | ushort | int | uint | float | real

⟨*struct*⟩ ::= ⟨*field*⟩ [, ⟨*field*⟩]*

⟨*field*⟩ ::= ⟨*ident*⟩ : ⟨*type*⟩

4 Const

$\langle\mathit{constBlk}\rangle$ ::= const $\langle\mathit{constStmt}\rangle$*

$\langle\mathit{constStmt}\rangle$ ::= $\langle\mathit{ident}\rangle$: $\langle\mathit{type}\rangle$ = $\langle\mathit{expr}\rangle$;

5 Function

$\langle\mathit{funcBlk}\rangle$ ::= function $\langle\mathit{ident}\rangle$ $\langle\mathit{paramBlk}\rangle$ $\langle\mathit{returnBlk}\rangle$ $\langle\mathit{funcBody}\rangle$

$\langle\mathit{paramBlk}\rangle$ ::= ($\langle\mathit{field}\rangle$ [, $\langle\mathit{field}\rangle$]*)

$\langle\mathit{returnBlk}\rangle$ ::= returns($\langle\mathit{field}\rangle$ [, $\langle\mathit{field}\rangle$]*)

$\langle\mathit{funcBody}\rangle$ ::= [$\langle\mathit{varBlk}\rangle$] let $\langle\mathit{eqStmt}\rangle$ tel

$\langle\mathit{varBlk}\rangle$ ::= var $\langle\mathit{field}\rangle$*

$\langle\mathit{eqStmt}\rangle$ ::= $\langle\mathit{lhs}\rangle$ = $\langle\mathit{expr}\rangle$

$\langle\mathit{lhs}\rangle$:= $\langle\mathit{ident}\rangle$ [, $\langle\mathit{ident}\rangle$]*

6 Expr

$\langle\mathit{expr}\rangle$ ::= $\langle\mathit{atomExpr}\rangle$ | $\langle\mathit{UnopExpr}\rangle$ | $\langle\mathit{BinopExpr}\rangle$ | $\langle\mathit{fieldExpr}\rangle$ | $\langle\mathit{structExpr}\rangle$
| $\langle\mathit{arrAccessExpr}\rangle$ | $\langle\mathit{arrInitExpr}\rangle$ | $\langle\mathit{preExpr}\rangle$ | $\langle\mathit{fbyExpr}\rangle$ | $\langle\mathit{arrowExpr}\rangle$ |
$\langle\mathit{whenExpr}\rangle$ | $\langle\mathit{ifExpr}\rangle$ | $\langle\mathit{caseExpr}\rangle$ | $\langle\mathit{exprList}\rangle$ | $\langle\mathit{applyExpr}\rangle$ | ($\langle\mathit{expr}\rangle$)

$\langle\mathit{atomExpr}\rangle$ ::= $\langle\mathit{bool}\rangle$ | $\langle\mathit{integer}\rangle$ | $\langle\mathit{float}\rangle$ | $\langle\mathit{char}\rangle$ | $\langle\mathit{ident}\rangle$

$\langle\mathit{UnopExpr}\rangle$ ::= $\langle\mathit{unop}\rangle$ $\langle\mathit{expr}\rangle$

$\langle\mathit{unop}\rangle$ ::= $\langle\mathit{atomType}\rangle$ | not | + | -

$\langle\mathit{BinopExpr}\rangle$ ::= $\langle\mathit{expr}\rangle$ $\langle\mathit{binop}\rangle$ $\langle\mathit{expr}\rangle$

$\langle\mathit{binop}\rangle$ ::= + | - | * | / | div | mod | and | or | xor | = | != | <| >| <= | >=

$\langle\mathit{fieldExpr}\rangle$ ::= $\langle\mathit{expr}\rangle$.$\langle\mathit{ident}\rangle$

$\langle\mathit{structExpr}\rangle$ ::= $\langle\mathit{expr}\rangle$ [, $\langle\mathit{expr}\rangle$]*

$\langle\mathit{arrAccessExpr}\rangle$ ::= $\langle\mathit{expr}\rangle$[$\langle\mathit{expr}\rangle$]

$\langle\mathit{arrInitExpr}\rangle$ ::= $\langle\mathit{expr}\rangle$ ^ $\langle\mathit{expr}\rangle$

⟨*preExpr*⟩ ::= pre ⟨*expr*⟩

⟨*fbyExpr*⟩ ::= fby(⟨*expr*⟩; ⟨*integer*⟩; ⟨*expr*⟩)

⟨*arrowExpr*⟩ ::= ⟨*expr*⟩ ->⟨*expr*⟩

⟨*whenExpr*⟩ ::= ⟨*expr*⟩ when ⟨*ident*⟩

⟨*ifExpr*⟩ ::= if ⟨*expr*⟩ then ⟨*expr*⟩ else ⟨*expr*⟩

⟨*caseExpr*⟩ ::= case ⟨*expr*⟩ of (| ⟨*pattern*⟩ : ⟨*expr*⟩)*

⟨*pattern*⟩ ::= ⟨*ident*⟩ | ⟨*integer*⟩ | ⟨*char*⟩ | ⟨*bool*⟩ | _

⟨*exprList*⟩ ::= ⟨*expr*⟩ [, ⟨*expr*⟩]*

⟨*applyExpr*⟩ ::= ⟨*prefixExpr*⟩ | ⟨*highorderExpr*⟩

⟨*prefixExpr*⟩ ::= ⟨*prefixOp*⟩(⟨*exprList*⟩)

⟨*prefixOp*⟩ ::= ⟨*ident*⟩ | ⟨*prefixUnop*⟩ | ⟨*prefixBinOp*⟩

⟨*prefixUnOp*⟩ ::= short\$ | int\$ | float\$ | real\$ | not\$ | +\$ | -\$

⟨*prefixBinOp*⟩ ::= \$+\$ | \$-\$ | \$*\$ | \$/\$ | \$div\$ | \$mod\$ | \$and\$ | \$or\$ | \$xor\$ |
\$=\$ | \$⟨⟩\$ | \$>\$ | \$>=\$ | \$<\$ | \$<=\$

⟨*highorderExpr*⟩ ::= (⟨*highorderOp*⟩ ⟨*prefixOp*⟩<<⟨*integer*⟩ >>)(⟨*exprList*⟩)

⟨*highorderOp*⟩ ::= fold | foldi | map | mapfold | mapi