/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Definitions

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**// The start of the program**

**// Looks for events, variables, functions, and actors**

Program: DefEvent\* DefGlobalConst\* DefFunc\* DefActor+

ActorItem: DefHSM | DefActorOn | DefMember | DefMethod

**// Syntax for defining variable types, I think?**

**Type**: **'int'** | **'string'** | **'bool'** | **'actorname'** | **'statename'** | **'eventname'**

**// Syntax for designing what happens when used?**

**// Like a class?**

DefActor: **'actor'** ActorName **'{'** ActorItem\* **'}'**

DefActorOn: **'on'** EventMatch OnBlock

DefHSM:   **'statemachine'** '{' StateItem\* **'}'**

DefState: **'state'** StateName **'{'** StateItem\* **'}'**

DefEvent: **'event'** EventName **'{'** [**Type** (',' **Type** )\*] **'}'** **';'**

DefFunc: **'func'** FuncName FormalFuncArgs [**'->'** **Type**] Block

DefGlobalConst: **'const'** **Type** VarName **'='** ConstExpr **';'**

**// ???**

DefOn: 'on' EventMatch OnBody

OnBody: GoStmt | OnBlock

OnBlock: Block

EventMatch: EventName **'{'** [VarName (**','** VarName)\*] **'}'**

**// Set State options**

StateItem: DefOn | DefEntry | DefExit | DefMember | DefMethod | DefState | InitialState

**// I guess input of one actor and its output as the effect?**

**// Like how with a state machine, an input sends you towards**

**// a different path to a new node (if it sends you anywhere)**

DefEntry: **'entry'** **'{'** Block **'}'**

DefExit: **'exit'** **'{'** Block **'}**'

**// Defining a variable in a class?**

DefMember: **Type** VarName **'='** ConstExpr **';'**

**// Defining a method in a class?**

DefMethod: **'func'** FuncName FormalFuncArgs [**'->'** **Type**] Block

InitialState: **'initial'** StateName **';'**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Statements

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**// Syntax to have as many statements as needed**

**// This allows for nesting**

Block: **'{'** Stmt\* **'}'**

**// All statement types**

Stmt: IfStmt | WhileStmt | DecStmt | AssignStmt | ExitStmt | ApplyStmt | SendStmt | PrintStmt | PrintlnStmt

**// Exit a function, state, expression and other things maybe?**

ExitStmt: **'exit'** **'('** NUMBER **')' ';'**

**// Return value at the end of a function maybe**

ReturnStmt: **'return'** Expr **';'**

**// Using an expression to assign value to a variable**

DecStmt: **Type** VarName **'='** Expr **';'**

AssignStmt: VarName **'='** Expr **';'**

ApplyStmt: ApplyExpr **';'**

**// ???**

SendStmt : HSMName **'!'** EventName ExprListCurly **';'**

**// Console output**

PrintStmt: **'print'** ExprListParen **';'**

PrintlnStmt: **'println'** ExprListParen **';'**

**// Syntax for parameters?**

FormalFuncArgs: **'('** [**Type** VarName (**','** **Type** VarName)\*] **')'**

**// Skip-to statements and conditions?**

GoStmt: JustGoStmt | GoIfStmt

JustGoStmt: **'go'** StateName Block

GoIfStmt: **'goif'** ParenExpr StateName Block [**'else'** (GoIfStmt | ElseGoStmt)]

ElseGoStmt: **'go'** StateName Block

**// If and While**

IfStmt: **'if'** ParenExpr Block [**'else'** (IfStmt | Block)]

WhileStmt: **'while'** ParenExpr Block

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Expressions

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**// All return and variable types**

VarExpr: VarName

IntExpr: NUMBER

StrExpr: STRING

BoolExpr: BOOL

**// Which expression gets used**

ConstExpr: IntExpr | BoolExpr | StrExpr

ValExpr: VarExpr | IntExpr | StrExpr | BoolExpr | ActorExpr | StateExpr |

EventExpr | ParenExpr

ApplyExpr: FuncName ExprListParen

**// What initializes variables**

Expr: ValExpr | BinOpExpr | ApplyExpr

ParenExpr: **'('** Expr **')'**

ExprListParen:**'('** [Expr (**','** Expr)\*] **')'**

ExprListCurly:**'{'** [Expr (**','** Expr)\*] **'}'**

**// Syntax and use of binary expressions**

BinOpExpr: ValExpr BinOp Expr

BinOp: **'\*'** | **'/'** | **'%'** | **'+'** | **'-'** **| '<<'** | **'>>'** | **'<'** | **'>'** | **'<='** | **'>='** | **'=='** | **'!='** | **'^'** | **'&&'** | **'||'** | **'\*='** | **'/='** | **'%='** | **'+='** | **'-='** | **'<<='** | **'>>='** | **'^='**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Variables

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**// Declaring variable types with strings**

ActorExpr: **'actor'** ActorName

StateExpr: **'state'** StateName

EventExpr: **'event'** EventName

**// String name input State, Actor, Functions, Variables, Events**

StateName: **NAME**

ActorName: **NAME**

FuncName: **NAME**

VarName: **NAME**

EventName: **NAME**