Yakindu-SCCharts textual desciption language

- State Scope
  - Variable Declarations
  - Signal Declarations
  - Local Reactions
    - LocalReactionTrigger
    - LocalReaction Effect
- Transition
  - Transition Trigger
  - Transition Effect
- Extending the Validator

State Scope

State Scopes allow to define **Variable Declarations**, **Signal Declarations**, and **Local Reactions:**

StateScope:

{SimpleScope} declarations+=(VariableDeclaration | SignalDeclaration | LocalReaction)*;

Variable Declarations

Variable Declarations in a State Scope are visible in this state and its descendant states.

A Variable:

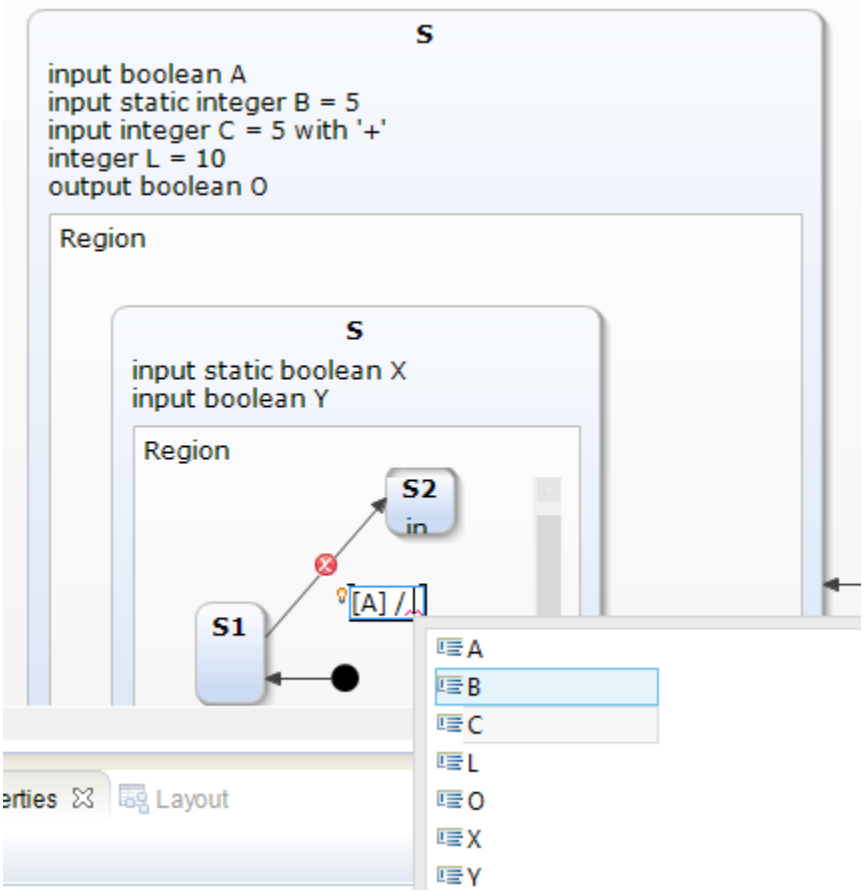
- has a Direction. It can be an input, an output, or both,
- can be Static (optional),
- has a Datatype (integer, boolean, real, string, void),
- has an Name,
- can be initialised (optional),
- can get a CombineOperator(optional) (-, +, *, max, min, or, and, host).

VariableDefinition:

{VariableDefinition} (isInput?='input')? (isOutput?='output')? (isStatic?='static')? type=[types::Type|FQN] name=ID ('=' varInitialValue=Expression)? ('with' varCombineOperator=CombineOperator)? ';';

Examples:

- input boolean A;
- output boolean A;
- input output boolean A; // double direction (input and output))
- input static boolean A; //static variable
- boolean A; //local, without direction
- input integer I=10; //the variable I is initialised
- input integer I with '+' //the combine operator is '+'



Signal Declarations

Yakindu events are interpreted as signals. A signal has a Direction and a Name.

SignalDefinition:

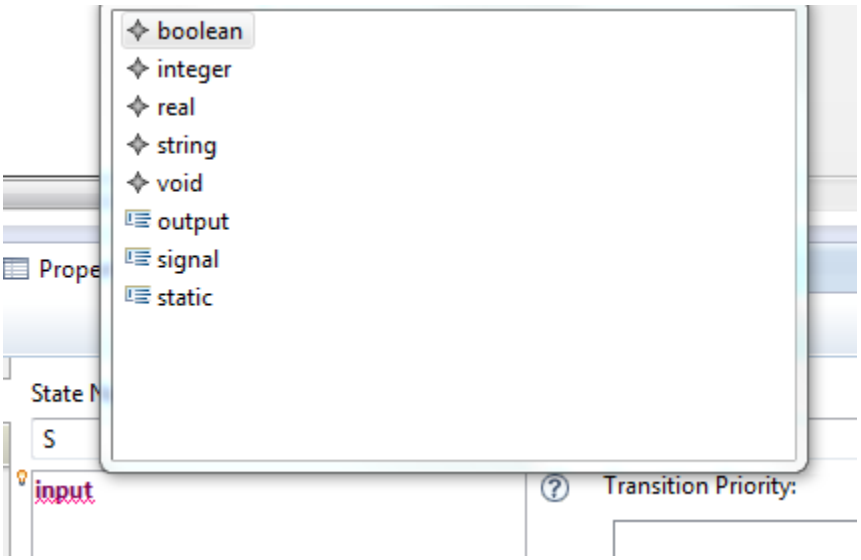
{EventDefinition} (isInput?='input')? (isOutput?='output')? 'signal' name=ID ';';

Examples:

- input signal I;
- output signal I;
- input output signal I;
- signal I;

	advantage	disadvantage	
Don't use signals	• no need to be implemented	• no signals	No
Boolean variables as signals	• implementation in a short time • a short form declaration: ***in I*** instead of ***in I:boolean*** • a short form of use: ***I / O*** instead of ***[I==true] / O = false***	• no more possible to use of Boolean variables	No

Yakindu events as signals	• use of already implemented features • differ between Signals and Boolean Variables Declaration: ***in signal I;***	• signals are implemented as events • **Question**: How to mix Signals and variables?	Yes
New declaration type	• differ between Signals and Boolean Variables Declaration: ***in* signal I;**	• expensive	no



Local Reactions

A Local Reaction has a Trigger and an Effect.

LocalReaction:

(trigger=(LocalReactionTrigger | ReactionTrigger))? ('/' effect=(ReactionEffect | SuspendEffect)) ';';

LocalReactionTrigger

LocalReactionTrigger returns sgraph::Trigger:

{ReactionTrigger} triggers+=LocalReactionType ('&&' (isImmediate?='#')? (delay=INT)? ((triggers+=RegularEventSpec) | ('[' guardExpression=Expression ']'))?)?;

Examples:

- Entry && S
- During && S
- Exit && S
- Exit && [S1 &&S2]
- Entry && # S
- Entry && # 3 S

LocalReaction Effect

ReactionEffect returns sgraph::Effect:

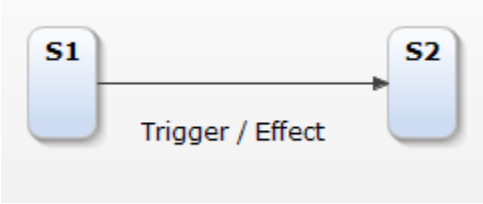
{ReactionEffect} actions+=Expression (=> ',' actions+=Expression)*;

Example:

- I / Suspend;
- / O=false;
- / I1=true, I2=false;
- / I1=true, I2=false;
- / O=true;

Transition

A Transition is defined by a Reaction, consisting of a Trigger and an Effect.



Transition Trigger

A trigger:

- can be a Signal,
- a boolean Variable
- an Expression
- can be immediate (optional)
- delay (optional)

ReactionTrigger returns sgraph::Trigger:

{ReactionTrigger} (isImmediate?='#')? (delay=INT)? ((triggers+=RegularEventSpec) | ('[' guardExpression=Expression ']'));

Examples:

- S
- [S1 && S2]
- #_S
- #_3 S

Transition Effect

ReactionEffect returns sgraph::Effect:

{ReactionEffect} actions+=Expression (=> ',' actions+=Expression)*;

Examples:

- / S
- / I = false
- / I1 = false, I2 = I3 + 10

Extending the Validator