ThinKCharts User Guide

ThinKCharts User Guide

Thin Kieler SyncCharts Editor

Authors:

- Matthias Schmeling <schm at informatik.uni-kiel.de>
- Hauke Fuhrmann <haf at informatik.uni-kiel.de>

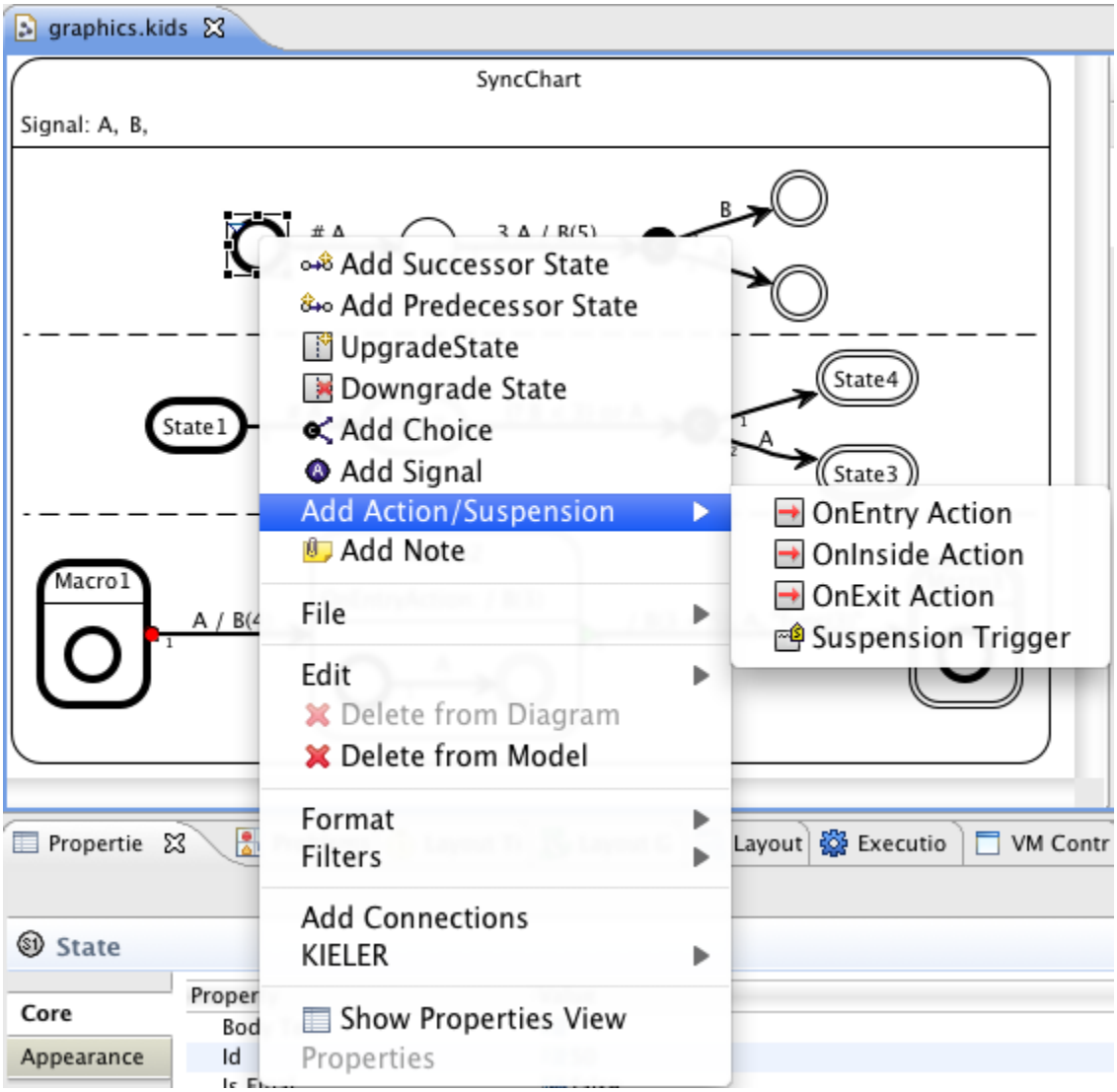
Create a new Diagram

- To create a new SyncCharts diagram first create a new project. (New->General->Project)
- Then create a new syncChart. (New->SyncChart Diagram)
- The wizard asks you for the name of the diagram file and its corresponding model file. These should be the same for both files.

- Use the KIELER main menu to add some default initial syncChart to your canvas to start from. Now you have a new syncChart with a top level state that contains a region. Note that there should only be one top level state in a syncChart.

Editing the diagram

Context Menu



- Use the *context menu*(right-click) to add most of the items
    - **Add Successor State**: Add a state with a transition following the currently selected state.
    - **Add Predecessor State**: Add a state with a transition with a transition to the currently selected state.
    - **Upgrade State**: Add a Region into a State and an initial State into that Region. Can be called either on Simple States or on Macro States.
    - **Downgrade State**: Remove a Region from a State.
    - **Add Choice**: Add a Conditional Pseudo States with two target States.
    - **Add Signal**: Add a Signal declaration to a State.
    - **Add Action/Suspension** : Add a State Action (OnEntry, OnInside (will be executed every tick), OnExit) or a Suspension Trigger. The first three may also contain a Trigger (hence a pure effect needs to start with a slash "/"), the latter is only a Trigger and may not contain an effect.
    - **Connect States**: Add a Transition between the two selected States.
    - **Flip Transition**: Change source and target of selected Transition.
    - **Set Transition Target/Set Transition Source**: Change either the target or the source of the currently selected transition to the currently selected State. Requires tp select exactly one Transition and one State.

Palette

The Palette is only there for old-style drag-and-drop editing. It is discouraged to use the palette to insert states and transitions. Better use the context menu as described above to edit the diagram with Structure-Based Editing operations. It will also automatically use auto-layout to get a nice graphical layout of the diagram.

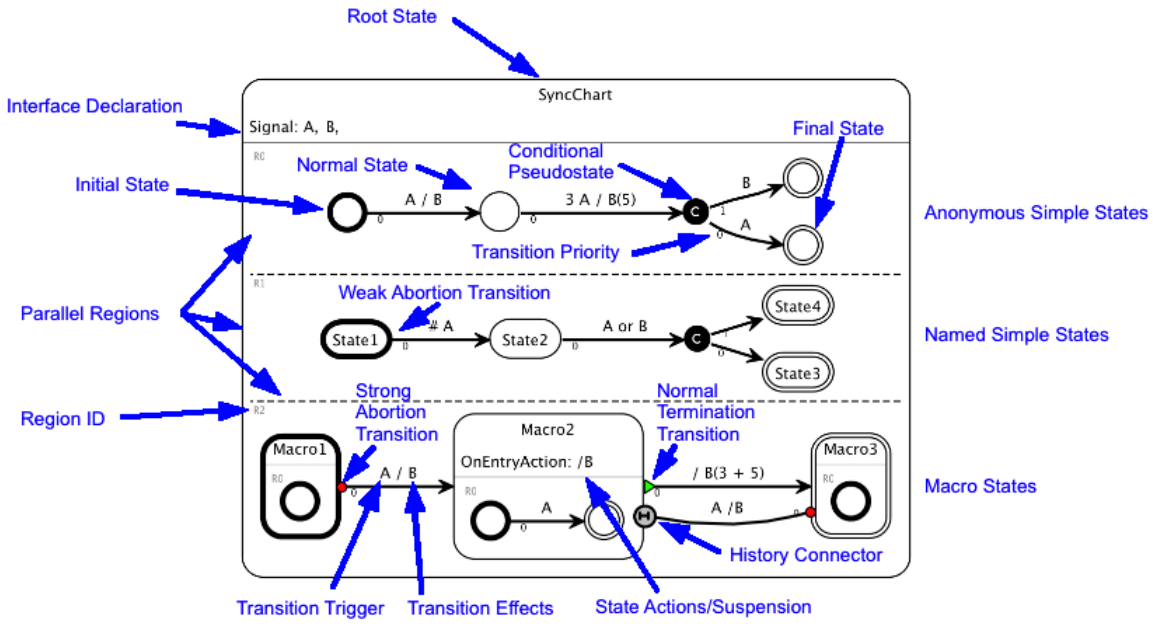
Properties

- Use the properties view to change internal attributes of graphical objects
- Open by
    - Window->Show view->Properties or
    - Right-click in the graphical editor -> Show Properties View
- Example attributes to change
    - Labels and IDs of States
    - Types and flags of States (Conditional, normal, reference, initial, final...)
    - Types and flags of Transitions (weak or strong abortion, normal termination, immediate, delay, history...)

Labels

- Main label properties like IDs or names can be changed by selecting the graphical object and just start typing on the keyboard. E.g. select a Region and start typing; this will edit the Region ID.
- Click a textual label multiple times for direct in-place editing of names, e.g.
    - State labels
    - Signal names
    - Transition labels
    - Action labels
    - Region IDs

Graphical Elements in a ThinKCharts SyncChart



Interface Declaration Parser

- Signals and Variables can be declared in a textual representation for a specific state and this state's subregions.
- If you are familiar with Xtext grammars or ENBF see following links for the complete syntax
  - Xtext: InterfaceDecl.xtext in de.cau.cs.kieler.syncharts.interfacedeclparser
  - ENBF: InterfaceDecl.html in de.cau.cs.kieler.syncharts.interfacedeclparser

Short Sample

- signals can be added using the *signal*keyword finished by a semicolon.
  - there can be an arbitrary number of comma separated signals
  - inputs are declared by using the keyword *input* (same for *output* and *input output*)
- variables are added in a similar way like signals by using the *var*keyword finished by a semicolon.
  - there can be an arbitrary number of comma separated variables just like for the signals
- an initial value is defined by :=following directly after the signal's name (care: use quotation marks for the value, as the underlying type is EString)
  - the value type can be declared directly afterwards with a preceding colon
- the combine operator and value type are declared by the keywords *combine ... with*
  - possible combine operators : NONE, +, *, max, min, or, and, host
  - possible value types : PURE, BOOL, UNSIGNED, INT, FLOAT,HOST
- signals and variables for a state's *subregions* are declared starting with region's identifier followed by a colon (*R0: signal...* for region R0)
- you can use single line comments *and multi line comments /* */ as you like*

```
signal A, B, C;                                    // three signals
input in1 := "5" : INT;                      // a signal with initial value
output out1 := "10" combine INT with +;
input output inout1 combine "uint8_t" with "combine_signals_fancy(x, y)"; // host combine operators can be fancy
output sig1, sig2 := "3" combine FLOAT with *;
R0: signal reg1, reg2, var var1, var2;
R0: var var3, var4;
R0: signal reg3, reg4;
```

Action Label Parser

- Action and Transition labels get automatically parsed and checked for correctness by a complex parser.
- The label language mainly follows Esterel and E-Studio label Syntax.
- For details, see
    - full label syntax grammar in Xtext notation: de.cau.cs.kieler.synccharts.labelparser/src/de/cau/cs/kieler/synccharts/labelparser/ActionLabel.xtext
    - full label syntax grammar in EBNF notation: de.cau.cs.kieler.xtext.docgenerator/src-gen/de.cau.cs.kieler.synccharts.labelparser.ActionLabel.html

General Form

```
Action:
        (isImmediate?='#')? (delay=INT)? (trigger=BooleanExpression)? ("/" (effects+=Effect (',')? )*)?;
```

- Example: # A or B / C, varD:=5 , or more complex:
  `A and B and C or D and not E or (43<?F) or (varA = 103) and not (B and (C or D)) / G, H(23) I, varB:=104`

- isImmediate: flag that indicates that the transition has to be tested immediately after the state has been entered. Especially in the same tick. By default, a transition is only tested beginning from the next tick.
- delay: counter that specifies the number of times that the trigger has to be evaluated to true before the transiton is taken. Example: 10 SECOND / ALARM.
    - Note: immediate and delay are mutual exclusive.
- Trigger: A boolean expression that is the guard of the transition (details below)
- Effects: A comma- or whitespace-separated list of effects that get executed when the transition is taken (details below).

Trigger

> *Trigger are boolean expressions over Signals and value comparisons that guard actions and transitions. They can comprise the following elements:*

- Simple Signal references:
  A

- Boolean expressions of Signals with usual precedence rules (not, and, or), e.g. these are equal
  `A and B or not C and D`

  `(A and B) or ((not C) and D)`

- Valued Signals and Variables can be used in conditions in these boolean expressions
    - Valid comparison operators: = , < , <= , > , >=
    - Value operator for valued signals: VAL="?", example: ?A
    - Examples:
      `variableX < 5`

      `?B = 3`

- Comparisons may contain arithmetic expressions with usual precedence rules with the operators +,-,*,/,mod
  `?B < (variableX + (2 * ?C))`

- Comparisons can be mixed into the boolean trigger expressions arbitrarily
  `A and (3 > ?B) or ((var5 + 2) = 6)`

- The **pre**operator will give the value of a signal of the instance of one tick behind, both in value or presence domain

  `A and pre(B)`

  `3 < pre(?A)`

Effects

- Effects get executed when a transition is taken (i.e. when its guard evaluates to true).
- They get separated from the trigger by a '/' symbol.
- Possible effects are Signal Emissions and Variable Assignments:
- Emission of a simple signal:
  `/ A`

- Emission of value of a valued Signal:
  `/ A(3)`

- Assignment of a variable:
  `/ varA := 42`

- Multiple effects get comma- or whitespace separated:
  `/ A, B, C(25), varA := 2`

- New values may use value expressions as explained above:
  `/ A(3 + pre(?B)), varC := (varD + 1)`

Host Code

> *If the action label syntax is not expressive enough, you can escape to any host-language, e.g. C, Java or whatever syntax.*

- Use quotes to indicate host code
  `'This is host code in the trigger' / 'this in an effect'`
- It can be mixed with other expressions
  `A and 'HostCode' / B, C(23), D('yet another Host Code'), 'EvenMore'`
- In parantheses the type of host code (e.g. the used language) may be given (optional)
  `B or 'MyCConstant'(C) / 'object.execute(xyz)'(Java)`

Naming

- Regions and States have IDs, that should be unique in their context (e.g. state IDs have to be unique within the same Region but not globally)
- States also have visible labels which in general is the name of the state. However, this label may be blank. In order to address all states correctly, IDs should not be blank.
- An automatic mechanism sets IDs automatically
  - the ID gets the label of the state where all whitespace is replaced by underscores '_'
  - an empty label results in an auto-generated ID like "S0", "S1", "S2", ...

Validation

- Finally, you should regularly test if your syncChart is correct. To do this, click on Diagram->Validate. Inconsistencies in your diagram are indicated by little red icons and also an entry in the problems view. Note that the validation may still be incomplete and not be able to point out all errors.
- Your diagram gets validated automatically whenever it is saved.
- For details on the checks performed see the Checkfile in Xtend Check language syntax: de.cau.cs.kieler.syncchartsmodel/SyncchartsChecks.chk

Differences to the original SyncCharts

- There is no initial connector, instead a state itself can be initial, by setting the isInitial property to true.
- There is no history connector, instead this is a transition property (isHistory).

Copy & Paste

Copy & Paste gives some extra functionality, as it allows to copy many graphical objects into many target objects. See the image as an example. In the image, nodes, edges or regions denoted with "S" are selected elements (sources) for the copy operation and "T" are selected elements for the paste operation (targets). Elements without label are just dummy elements.

Source/Target?	State	Region	Transition
State	replace, keep all transitions	add State to Region	insert, old Transition stick with source
Region	insert	replace, if root: create state and insert	insert, old Transition stick with source
Transition	add as selfloop	create dummy States and connect	replace
States	replace, discard transitions	add States to Region	insert, old Transition stick with source
Regions	insert	replace, if root: create state and insert	insert, old Transition stick with source
Transitions	add as selfloops	create dummy States and connect	replace with all

Source/Target?	States	Regions	Transitions
State	replace each	insert into each	insert into each
Region	insert into each	replace each, if root: do nothing	insert into each
Transition	Connect 2 states	add two dummy states with transition in each	replace each
States	replace each with all	insert all into each	insert into each
Regions	insert all into each	replace each with all	insert into each
Transitions	Connect 2 states with all	add two dummy states with all transition in each	replace each with all