Differences

SyncCharts differ from other implementations of Statecharts:

- Synchronous framework
- Determinism
- Compilation into backend language Esterel
- No interpretation for simulations
- No hidden behaviour
- Multiple events
- Negation of events
- No inter-level transitions

Simple Sequential Automaton

SyncChart:

main

Elements:

- States:
  - Regular state (circle)
  - Terminal state (doubled circle)
  - Hierarchic state (box with rounded edges)
- Transitions:
  - Arrows with labels
- Connectors:
  - Colored circles with single letters

Hierarchic States

main_hier

SyncCharts know four types of states:

- Simple States: Carry just a label.
- Graphic Macrostates: Encapsulates a hierarchy of other states, including further graphic states.
- Textual Macrostates: Contain statements of the Esterel language. They are executed on entry of the state.
- Run Modules: Include other modules.

Transitions are **not** allowed to cross the boundaries of graphic macrostates. This is in contrast to other modelling tools.

Parallel States

main_parallel

- Dashed lines (horizontal or vertical) separate parallel executed states inside a graphic macrostate.
- Each segment may be segmented into further parallel segments, but iterative segmentation does not introduce additional hierarchy. All parallel segments in a graphic macrostate are at the same level.

Parallel States

main_parallel

- A transition outside the graphic macrostate with normal termination is activated, when all parallel segments have reached their terminal state.
- If just one segment does not have one or if it is not reached, then the normal termination transition will never be activated.

Modules

m

A module like this with an interface:

```
input  I;
output O;
```

main_moduse

. . . can be used as a Run Module with these signal bindings:

```
  signal S1 / I;
  signal S2 / O
```

Syntax of Transition Labels

main_trans

Informal syntax of a transition label between states S1 and S2, all elements are optional:

```
# factor trigger {condition} / effect
```

Basic activation and action:

- `trigger` is an expression of signal presence like "`A or B`"
- Enclosed in braces is the `condition`. It is a data expression over signal values or variables like "`?A=42`"
- Behind a single "/" follows the "`effect`" as a list of emitted signals if the transition is executed. Multiple signal names are separated with ",".

Syntax of Transition Labels

main_trans

Informal syntax of a transition label between states S1 and S2, all elements are optional:

```
# factor trigger {condition} / effect
```

Extensions:

- "#" is the flag for an immediate transition
- "`factor`" is the (natural) number of instants a transition must be active before it is executed. These active instants does not need to be consecutive, but S1 must be active all the time.

Transition Labels: Examples

main_trans

The following label examples belong to the transition originating at S1 and leading to S2:

- `A/B`
  After entering S1 the signal A is tested from the next instant on. If A is present, then B is emitted in the same instant and state S2 is entered.
- `/B`
  After enabling S1, B is emitted in the next instant and S2 is entered.
- `3 A/`
  The transition is executed, if S1 is active consecutively and signal A is present for 3 times.

Transition Labels: Examples

- `#A/`
  If `S1` is entered, signal `A` is tested from the same instant on. If `A` is present in the instant `S1` is entered then state `S2` is entered in the same instant.
- `{?A=42}/`
  The transition is executed, if the (valued) signal `A` carries the value `42`. `A` does not need to be present for this test.
- `A {?A=42}/`
  This test succeeds if `A` is present and carries the value `42`.
- `A and (B or C)/`
  Logical combination of signal presence.
- `{?A=10 and (?B<3 or ?C=1)}/`
  Logical combination of value tests.
- `/A(2), B(4)`
  Emission of multiple valued signals

Transition Priorities

main_prio

- When more than one transition departs a state, an automatic (but editable) priority ordering is established.
- The transition labels are evaluated according to their priority.
- The first label that succeeds activates its transition.
- Low numbers mean high priority.

Transition Types

SyncCharts feature four different types of transitions: They are differentiated by a symbol at the arrow root:

main_transdiff

- Initial connector: Initial arc
  Initial arcs connect the initial connectors of the chart with the other states.
- No symbol: Weak abort
  When the trigger/condition of the transition is enabled, then the actions of the originating state in the current instant are executed for a last time, then the transition action, and the entry action of the new state.
  In other words:
  The old state can "express it's last will".

Transition Types

main_transdiff

- Red bullet: Strong abort
  The action for the current instant of the old state is not executed. Only the transition action and the entry action of the new state is executed.
- Green triangle: Normal termination
  This transition can be used to exit macro states. It is activated when the macro state terminates.

All these transition types must not be confused with "immediate" or "delayed" evaluation of the transition label (label prefix "#").

Transition Types and Labels

Some transition types have restrictions on their labels:

main_transdiff

- Initial arc:
  These are always "immediate," therefore the additional flag "#" is not needed.
- Weak abort: No restrictions.
- Strong abort: No restrictions.
- Normal termination:
  They support no triggers or conditions because they are activated by the termination of the originating state. The immediate flag is not used either.

Transition Types and Priorities

The type of a transition interacts with it's priority:

- Strong abort: Highest priority
- Weak abort: Middle priority
- Normal termination: Lowest priority

Esterel Studio enforces these rules by changing the numerical priorities of the transitions.

Local Signals and Variables

main_signal

Local signals:

- Defined in the body of a graphical macrostate
- Shared between parallel threads

Local variables:

- Attached to the initial connector
- Not shared

Connectors

This (artificial) SyncChart demonstrates all four connector states:

main_conn

Initial connector:

- Activated at activation of the macrostate
- Only departing transitions permitted
- All connected transitions are "immediate"

Conditional connector:

- All departing transitions are "immediate"
- One departing "default" transition without condition must be present.

Connectors

This (artificial) SyncChart demonstrates all four connector states:

main_conn

Suspend connector:

- The suspend state is always active.
- Only one departing transitions is permitted.
- The transition can only hold a trigger expression.
- The "immediate" flag can be enabled on demand.
- When the transition is activated, then the target state is (strongly) suspended.

Connectors

This (artificial) SyncChart demonstrates all four connector states:

main_conn

History connector:

- This connector is directly attached to macrostates
- Only incoming transitions can connect.
- The previous state of the macrostate is restored when it is entered through a history connector.

Esterel Studio

`http://www.esterel-technologies.com/`

Esterel Studio features:

- Graphical editor for Statechart dialect "Safe State Machines", a. k. a. SyncCharts
- Code generator for textual Esterel
- Esterel compiler for C code production
- Interface to backend in host language (C)
- Graphical simulation Validation/Testing environment

To Go Further

- David Harel, Statecharts: A Visual Formulation for Complex Systems, *Science of Computer Programming*, 8(3), June 1987, pp. 231–274
- D. Harel, M, Politi, *Modeling Reactive Systems with Statecharts: The STATEMATE Approach*, McGraw-Hill, ISBN 0-07-026205-5, 1998.
- Charles Andr, Semantics of S.S.M (Safe State Machine), Esterel Technologies Technical Report, April, 2003, `http://www.esterel-technologies.com/v3/?id=50399&dwnID=48`
- Home page of Esterel Technologies: `http://www.esterel-technologies.com/`
- Local information on Esterel-Studio and where to find further documentation: `http://www.informatik.uni-kiel.de/~esterel/`