The 5-Minute Review Session

1. Why does constructiveness analysis matter in circuit design?
2. How do cyclic and acyclic circuits differ?
3. How do we analyze constructiveness of a cyclic circuit?
4. How do we extend Boolean functions to three-valued logic?
5. Are TVFs monotonic? What does that mean?

Overview

SyncCharts (Safe State Machines)
Comparison with Harel’s Statecharts
Simple Automata, Hierarchy, Concurrency/Parallelism
A Tour of SyncCharts

From Esterel to SyncCharts
From SyncCharts to Esterel
Statecharts

Statecharts proposed by David Harel [1987]
In a nutshell: Statecharts = Mealy Machines
+ hierarchy (depth)
+ orthogonality
+ broadcast
+ data

Harel-Statecharts vs. SyncCharts—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

Harel-Statecharts vs. SyncCharts—Similarities

SyncCharts are made up of elements common to most Statecharts dialects:
- States
- Initial/terminal states
- Transitions
- Signals/Events
- Hierarchy
- Modularity
- Parallelism

Simple Sequential Automaton

SyncChart:
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
SyncCharts know four types of states:

1. **Simple States**: Carry just a label.
2. **Graphic Macrostates**: Encapsulates a hierarchy of other states, including further graphic states.
3. **Textual Macrostates**: Contain statements of the Esterel language. They are executed on entry of the state.
4. **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**

- 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**

A module like this with an interface:

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

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

```
signal S1 / I;
signal S2 / O;
```
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 ∨ 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 “,”.

**Extensions:**

- “#” is the flag for an immediate transition
- “factor” is the (natural) number of ticks a transition must be active before it is executed. These active ticks do not need to be consecutive, but S1 must be active all the time.
Transition Priorities

- 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:

1. Initial connector: Initial arc
   - Initial arcs connect the initial connectors of the chart with the other states.

2. No symbol: Weak abort
   - When the trigger/condition of the transition is enabled, then the actions of the originating state in the current tick 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 its last will”.

3. Red bullet: Strong abort
   - The action for the current tick of the old state is not executed. Only the transition action and the entry action of the new state is executed.

4. 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:

- 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 its priority:

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

Local Signals and Variables

Local signals

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

Local variables

- Not shared

Connectors

This (artificial) SyncChart demonstrates all four connector states:

- **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.

- **Suspension connector**
  - The suspend state is always active.
  - Only one departing transition 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:

- **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.

Overview

**SyncCharts (Safe State Machines)**

**From Esterel to SyncCharts**

- Step 1: Transform Esterel to SyncChart
- Step 2: Reduce to Fully Graphical SyncChart
- Step 3: Optimizations

**From SyncCharts to Esterel**

Equivalence of SyncCharts and Esterel

- Esterel and SyncCharts look different
- However, underlying model of computation/semantics are equivalent
- Both are based on synchrony hypothesis
- Can translate one into the other

Motivation

- Can transform Esterel projects into SyncCharts
- Better visualization of the behavior of Esterel projects
- Combine benefits of textual editing and graphical viewing (see KIELER project)
- Didactical purposes
From Esterel to SyncCharts

Step 1: Transform Esterel program into SyncChart with textual macro states
Step 2: Iteratively apply reduction rules to transform Esterel constructs into graphical components
Step 3: Optimize SyncChart

Step 2: Reduce to Fully Graphical SyncChart

Signal emission

Sequence

Signal awaiting

Weak Abortion

+ 19 additional rules
Translation of traps not trivial—see [Prochnow et al. 2006]
### Example: ABRO

#### Applying Rule (module)

```
module ABRO:
  input A, B, R;
  output O;
  loop
    [ await A ||
      await B ];
  emit O;
  each R
end module
```

#### Applying Rule (sequence)

```
ABRO
loopeach
I
R/
[ await A
  ||
  await B ]; emit O
```

#### Applying Rules (parallel) + (emit)

```
ABRO
loopeach
R/
emit
parallel
I
I/O
await B
await A
```

---

### Example: ABRO

#### Applying Rule (lnode)

```
module ABRO:
  input A, B, R;
  output O;
  loop
    [ await A
      ||
      await B ];
  emit O;
  each R
end module
```

#### Applying Rule (sequence)

```
ABRO
loopeach
I
R/
[ await A
  ||
  await B ]; emit O
```

#### Applying Rules (parallel) + (emit)

```
ABRO
loopeach
R/
emit
parallel
I
I/O
await B
await A
```
Example: ABRO
 Applying Rule (simple await)

Step 3: Optimizations

Motivation

- Automatic synthesis produces “verbose” modules
- However, also human modelers (esp. novices) may produce sub-optimal models

Notes:

- It may be a matter of style/opinion what “optimal” means
- However, consistency in style is desirable in any case—and standardized optimization rules help to achieve this

Preconditions:

- no abort originates from S
- $S$ has no local signals
  + further rules to remove conditionals, combine terminal states, remove normal terminations
Overview

SyncCharts (Safe State Machines)

From Esterel to SyncCharts

From SyncCharts to Esterel

Examples
Structural Translation
Ordering States

Basic Idea

- Structural Induction on SyncChart
- Simple-States become \texttt{await}
- Macro-States become \texttt{abort}
- Parallel stays parallel
- Problem: How to order states?

From SyncCharts to Esterel

Motivation

- Intermediate Step from SyncChart to C-Code (or VHDL, ...) 
- This is what Esterel Studio does
- This translation is one possibility to define the semantics of SyncCharts.
- The following description of the translation is based on [André 1996] and the synthesis actually done by Esterel Studio
  (excluding the optimizations done by Esterel Studio)
Simple Await

For an empty state, it is irrelevant whether an outgoing transition is strong or weak. This is directly reflected in the Esterel code.
SyncCharts (Safe State Machines)

Examples

Structural Translation

Ordering States

Simple Circle

module Circle:
  input I1, I2;
  output O1, O2;
  nothing;
  loop
    await case (tick) do
      nothing;
    await case (tick) do
      nothing
    end await
  end loop
end module

Not–so–simple Hierarchy

nothing;
% state Parent
trap sc_end_automaton in
nothing;
loop
% state C1
await case (I) do
nothing;
% state C2
exit sc_end_automaton
case (tick) do
emit 0
end await
end loop
end trap
nothing;
halt

Components

The translation is performed on a variant of the And-/Or-Tree

Reactive Cell
  State + Outgoing transitions
  State can contain macrostate

Region
  Multiple connected Reactive Cells

Macrostate
  Nonempty set of Regions

We have to introduce new signals and traps
  trap end_automaton for normal termination
  signal goto_s for each state s in a region
While this translation seems obvious, it is only possible due to strong restrictions on terminal states, e.g.,

- No outgoing transitions
- No on inside actions

Notes:

- In this example, the effects are emissions of some signal; in general, the effects can be more complex
- For simple states, run M is nothing.
- If no normal termination exists, we have to add a halt after the run. Hence for simple states, which never have normal terminations, the abort is equivalent to an await.
- The module interface may in general contain more signals than the ones shown here.
Note that the disjunction of the triggers must be a tautology—i.e., it must be ensured that one of the transitions will always be triggered.

- The default behavior is, in each loop iteration, to enter a cell, based on the goto signals, to spend some time in that cell, and then to re-enter the loop via the exception L.
- However, we also want to handle transient states. The signal α is used to prevent illegal instantaneous re-entries of the loop. If α is still present when it is tested, the corresponding cell has been entered and left again instantaneously, and we are not allowed to re-enter the loop; instead, we continue execution of the same iteration of the loop body.
- When a terminal state is reached, no further behavior is possible, thus the whole region can be stopped; this is done by throwing the exception end_automaton
- When a cell is run, we bind the transition signals according to the context of the cell. In this example, the transition signal trans_1 of cell S7 becomes goto_S4 of region Region2.
- The halt in the loop statement serves to assure the compiler that the loop is not instantaneous.
How to order the states of a Region?

- A state can be entered and left in the same tick
- Thus several states can be active in the same tick
- All immediate reachable states have to be later in the loop
- Could forbid immediate transitions
- This would make modeling more difficult
- All transitions form a dependence graph
- Remove all delayed abortions from this graph
- Note: normal terminations might be immediate
- This graph has to be acyclic!

If the graph contains a cycle, we could consider further details, to decide whether the circle can really occur, e.g., whether normal termination can take place in the first tick.

Summary

- SyncCharts can be translated to Esterel, and vice versa
- Translation of SyncCharts to Esterel:
  - Structural induction of And-/Or-Tree
  - Macrostates, regions and reactive cells are translated separately
  - Challenge: unstructured control flow
- Translation from Esterel to SyncCharts:
  - Structural induction on Esterel code
  - Resulting SyncChart has to be optimized to become readable
  - Challenge: traps

To Go Further