Co-design Finite State Machines

Many slides of this lecture are borrowed from Margarida Jacome

Summary of FSM

• Reactive
• synchronous operation
  – All states change state simultaneously
• syntactic determinism
**Synchrony**

- **Basics operation:** At each clock tick, each module reads input, computes and produces outputs simultaneously.
  ⇒ *zero delay* calculations, *infinite time* between ticks. (no cyclic dependencies among values of events with same tag)

- **Triggering and ordering:** All modules are triggered to compute at every clock tick. At each clock tick, there is no *ordering* of reading of inputs, computation or writing outputs. However, an ordering can be imposed with delta step (delay) concept.
  ⇒ *zero time* that passes between events at the same clock tick and that serves simply to order events.

**System Solution:** This is the output reaction to a set of inputs. Unique solution is desirable at each clock tick. This way, easy to analyze and verify.
- However, there are cyclic dependencies among values of events (due to selection of models and languages) that makes it difficult.

**Implementation cost:**
- For hardware, one must ensure the clock period is higher than the maximum possible computation time for a synchronous block, clock rate is much slower than that might otherwise achieved.
- For software, ensure that invoked process completes before process changes its input.
Asynchrony

- **Basic operation**: Events have non-zero time between them. Individual process runs whenever change in its input and can take arbitrary (bounded) time to complete its computation.
- **Triggering and ordering**: Triggered to run when input changes. There is no a priori ordering processes among triggered modules.
- **System solution**: Difficult to analyze due to solution depends on input signals and its timing.
- **Cost**: Less expensive.

---

**The Synchronous Hypothesis**

Operational Cycle of a FSM

1. Idle
2. Detect input events
3. Transition, according to which events are present and a transition relation
4. Emit output events

FSM phase 1: duration between zero and infinity
phase 2/3/4: duration of zero
The Synchronous Hypothesis

System instantaneously reacts to events...

- The chronometric notion of time is replaced by a notion of order among events
  - only relevant notions are simultaneity and precedence between events

FSMs and CFMSs

- Mixed hardware-software systems may contain components that proceed at very different speeds
  - synchronous hardware modules
    - execute concurrently
    - tolerate real-time states and output at each clock cycle
  - software modules
    - execute sequentially
    - motion in condition may not attempt to exit
    - need explicit external processor

FSMs and CFMSs

FSMs can be used to model such systems but their use would be excessively cumbersome...

CFMSs specialize model that incorporates the unbounded delay assumption

Operational Cycle of a CFMS

- Four Phases:
  1. Init
  2. Detect input events
  3. Transition, according to which events are present and match a transition relation defined
  4. Emit output events

- FSM phase 1: duration between zero and infinity
- FSM phase 2: duration between zero and infinity
- FSM phase 3: duration of set
CFSM Overview

- **A FSM part**: that contains set of inputs, outputs, and states, a transition relation and an output relation.
- **A data computation part**: references in transition relation to external, instantaneous (combinational) function.
- **A locally synchronous** behavior: Execute transition by producing a single output reaction based on a single, snap-shot input assignment in zero time. *(synchronous from its own perspective)*
- **A globally asynchronous** behavior: Each CFSM reads inputs, execute a transition, and produce outputs in an unbounded but finite time as seen by the rest of the system. This is asynchronous interaction from the *system perspective*.

Communication primitives

- Single input, single output communication process
- event emitted by sender (CFSM1) setting the event buffer to 1. Putting signal value in data buffer.
- Consumed by receiver (CFSM2) after detection of 1in event buffer. Then set “0” to event buffer.
CFSM Networks

- Net: set of connections on the same output signal.
- Network: a set of CFSMs and nets.
- Example:
  - set of CFSMs in software (e.g. C), a compiler, an operating system, and a microprocessor (software domain),
  - a set of CFSMs in hardware (e.g. gates mapped to an FPGA), a hardware initialization scheme and the interface between them (polling or interrupt).
**Events**

An event is a triple $e = (e_1, e_2, e_3)$:
- $e_1$ is the name of the event
- $e_2$ is the set of values where it occurs
- $e_3$ is the set of values the event takes

$i.e.$, the "communication port" where it occurs

A event with a name "temperature" could occur every time a certain sensor reports a new value, in the range between $0$ and $30^\circ C$.

Some events may have "interesting" value (e.g., clock)
In this case $e_2$ is the special symbol $\infty$.

*Example: Seat Belt*

Five seconds after the key is turned on, if the belt has not been fastened, an alarm will beep for ten seconds or until the key is turned off.

<table>
<thead>
<tr>
<th>Input events of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event name</td>
</tr>
<tr>
<td><strong>BELT</strong></td>
</tr>
<tr>
<td><strong>KEY</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output events of the system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event name</td>
</tr>
<tr>
<td><strong>ALARM</strong></td>
</tr>
</tbody>
</table>

*Example (cont.)*

Five seconds after the key is turned on, if the belt has not been fastened, an alarm will beep for ten seconds or until the key is turned off.

<table>
<thead>
<tr>
<th>Internal events of the system (i.e., events exchanged by the system components and not visible outside)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event name</td>
</tr>
<tr>
<td><strong>START</strong></td>
</tr>
<tr>
<td><strong>END</strong></td>
</tr>
</tbody>
</table>

*CSMs*

A CSM is basically constituted by:

- a set of input events
- a set of output events
- a transition relation

The transition relation describes how input events can cause output events.
### Transition Relation

Describe the input events and output events.

- It is a set of pairs of sets
  - First member of each pair: set of input names and values
  - Second member of each pair: set of output names and values

**Transition**

- Triggered by the input events with the appropriate values
- Acts on the output events with the appropriate values

The transition phase is nonsensical and non-useful.

---

### Input Events

1. Trigger events
   - Can be used only once to cause a transition of a given CFSM
   - Each occurrence is accompanied by the triggered transition
   - Can cause many transitions in different CFSMs
2. Pure value events
   - Cannot directly cause a transition
   - Can be used to choose among different possibilities involving the same set of trigger events (and their values).

---

### Input Events

For a given system, measure the temperature every minute, and act appropriately.

System can be modeled as a CFSM with two input events:
- Timing event and temperature change event.

- The action (CFSM transition) occurs only due to a time change.
- The action should take into account the value of the temperature event when the time change event occurs.

Modeling both actions allows some other system component to react to temperature changes rather than time changes.
States

1. the state of a CFM consists of a set of event types that are at the same time input and output for it.

   - the non-zero reaction time of this feedback loop provides the "storage" capability that is required to implement the concept of state.

\[
\text{CFMs reaction time is unbounded and non-zero}
\]

CFMs

A CFM is a quintuple \( C = (I, E, O, R, F) \):

1. \( I = \{(i_1, e_1), (i_2, e_2), \ldots\} \) is a finite set of input event names and of the corresponding finite set of allowed values
2. \( E \subseteq I \) is the set of "trigger" input event names
3. Events with names in \( I \setminus E \) are "pure data" events
4. \( O = \{(o_1', o_1), (o_2', o_2), \ldots\} \) is a finite set of output event names and of the corresponding finite set of allowed values, such that \( E \cap O = \emptyset \) (i.e., the same event cannot be both an input and an output)
5. \( R \subseteq \{(o_1', o_1), (o_2', o_2), \ldots\} \) is a set of possible initial values of (some) output events
6. \( F : (I \times R) \rightarrow (I 
\times R) \) is the transition relation
7. \( F : (I \times R) \rightarrow (I 
\times R) \) is the transition relation
8. \( F : (I \times R) \rightarrow (I 
\times R) \) is the transition relation
9. \( F : (I \times R) \rightarrow (I 
\times R) \) is the transition relation
10. \( F : (I \times R) \rightarrow (I 
\times R) \) is the transition relation

\[
\text{CFMs are sets of (some) output events}
\]
**Seat Belt Example Revisited**

When a belt is turned on, if the belt is not being fastened, an alarm will keep beeping until the key is turned off.

<table>
<thead>
<tr>
<th>Input events</th>
<th>Event name</th>
<th>Event values</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>BELT</em></td>
<td>ONOFF</td>
<td></td>
</tr>
<tr>
<td><em>KEY</em></td>
<td>ONOFF</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output events</th>
<th>Event name</th>
<th>Event values</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ALARM</em></td>
<td>ONOFF</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Internal events</th>
<th>Event name</th>
<th>Event values</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>START</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>END</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Seat Belt Example**

A CPSM describing the desired event/reaction pattern:

- *KEY = ON* -> *START*
- *KEY = OFF,* *BELT = ON* -> *WAIT*
- *END = 1* -> *ALARM = ON*
- *END = 2* + *BELT = ON* + *KEY = OFF* -> *ALARM = OFF*

+ denotes the logic or condition

Separate important related events of a given transition.
Example: Formal Description

The formal description of the state \( CFSM \)
\( C \in \{ O, E, R, F \} \) is:

- \( I = \{ [\text{KEY}, \text{ON}, \text{OFF}], [\text{BELT}, \text{ON}, \text{OFF}] \} \)
- \( E = \{ [\text{END}, [S, 0]], [\text{ON}, \text{OFF}, \text{WAIT}, \text{ALARM}] \} \)
- \( F = \{ [\text{KEY}, \text{ON}, \text{OFF}], [\text{BELT}, \text{ON}, \text{OFF}] \} \)

\( s \rightarrow \) particular event (convention: state not preceded by "\( s \)"

---

Example: Formal Description (cont.)

\( C \in \{ O, E, R, F \} \):

- \( F \in \{
  \{ [\text{KEY}, \text{ON}, \text{OFF}] \rightarrow \{ \text{ON}, \text{OFF} \}, \text{START} \},
  \{ [\text{KEY}, \text{ON}, \text{ON}, \text{OFF}] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ [\text{BELT}, \text{ON}, \text{ON}, \text{OFF}] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ [\text{END}, [S, 0]] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ [\text{KEY}, \text{ON}, \text{OFF}] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ [\text{BELT}, \text{ON}, \text{OFF}] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ \text{FINISH} \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ [\text{KEY}, \text{OFF}, \text{OFF}] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ [\text{BELT}, \text{OFF}, \text{OFF}] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \},
  \{ [\text{END}, [S, 0]] \rightarrow \{ \text{OFF}, \text{OFF} \}, \text{OFF} \}
\} \)
### Network of CFSMs

A Network of CFSMs is a set of CFSMs

\[ N = \{ C_1, C_2, C_3, \ldots \} \]

such that no two different CFSMs have the same output event name in common (implying that \( C_1 \cap C_2 = \emptyset \)).

- Output sets are disjoint in order to avoid the difficulties inherent in the implementation of the update of a single object by two concurrent agents (which requires the use of an explicit communication mechanism).

- Input sets need not be disjoint, thus implying a broadcast communication mechanism (as opposed to point-to-point).

### Seat Belt Example

The network of CFSMs would be composed by \( C \), plus a CFSM implementing the timer, \( C_1 \), defined as follows:

\[ C_1 = \{ 0_1, 0_2, 0_3, 0_4, \ldots \} \]

1. \( I = \{ \text{START}, \text{TICK} \} \) represents an input event in the environment occurring once a second.

2. \( E = \{ \text{END} \} \)

3. \( O_1 = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8 \} \)

4. \( R_1 = \{ 0, 1, 3 \} \)

### Seat Belt Example

\[ C = \{ 0, 1, 2, 3, 4, \ldots \} \]

- \( \text{TICK} \), (4, 0) \( \rightarrow \) (0, 0).
- \( \text{START} \), (6, 0) \( \rightarrow \) (0, 0).
- \( \text{TICK} \), (4, 0) \( \rightarrow \) (0, 0).
- \( \text{START} \), (6, 0) \( \rightarrow \) (0, 0).
- \( \text{TICK} \), (4, 0) \( \rightarrow \) (0, 0).
- \( \text{START} \), (6, 0) \( \rightarrow \) (0, 0).
- \( \text{TICK} \), (4, 0) \( \rightarrow \) (0, 0).
- \( \text{START} \), (6, 0) \( \rightarrow \) (0, 0).
- \( \text{TICK} \), (4, 0) \( \rightarrow \) (0, 0).
- \( \text{START} \), (6, 0) \( \rightarrow \) (0, 0).

- \( \text{TICK} \), (4, 0) \( \rightarrow \) (0, 0).
- \( \text{START} \), (6, 0) \( \rightarrow \) (0, 0).

The system continues to perform.
Exercise

- Consider the elevator problem. Create CFSM formal descriptions of various components (such as logic control, timer etc.) of your design. Illustrate the CFSM network for such system by associating these components.