Co-design Finite State Machines

Many slides of this lecture are borrowed from Margarida Jacome

Summary of Dataflow Network Model

- Partially ordered tags
- Explicit concurrency
- Blocking read (non-reactive)
- Fundamentally deterministic
- No input or output choice
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.
  \[ \Rightarrow \text{zero delay calculations, infinite time between ticks. (no cyclic dependencies among values of events with same tag)} \]
- **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.
  \[ \Rightarrow \text{zero time that passes between events at the same clock tick and that serves simply to order events.} \]
Synchrony

- **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. Init
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
- Phases 2-3: 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 CFSMs

- Mixed hardware-software systems may contain components that proceed at very different speeds
- Synchronous hardware modules
  - Execute concurrently
  - Compute and state and output events cyclically
- Software modules
  - Execute sequentially
  - Require to conditionally synchronize hundreds or even thousands of modules

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 CFSMs

- Mixed hardware-software systems may contain components that proceed at very different speeds
- Synchronous hardware modules
  - Execute concurrently
  - Compute and state and output events cyclically
- Software modules
  - Execute sequentially
  - Require to conditionally synchronize hundreds or even thousands of modules

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 CFSMs

- Mixed hardware-software systems may contain components that proceed at very different speeds
- Synchronous hardware modules
  - Execute concurrently
  - Compute and state and output events cyclically
- Software modules
  - Execute sequentially
  - Require to conditionally synchronize hundreds or even thousands of modules

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 CFSMs

- Mixed hardware-software systems may contain components that proceed at very different speeds
- Synchronous hardware modules
  - Execute concurrently
  - Compute and state and output events cyclically
- Software modules
  - Execute sequentially
  - Require to conditionally synchronize hundreds or even thousands of modules
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

1-place buffer

- 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 1 in event buffer. Then set “0” to event buffer.

Basics

1. System modeled as a network of interacting CFSMs communicating through events
   - Each CFSM takes a non-zero unbounded time to perform its task
   - All need buffer as implementation is infinite
2. Protocol between communicating CFSMs
   - Receiver waits for the sender to emit the event
   - Sender can proceed after emitting the event without the need to wait

Network of CFSMs: Depth-1 Buffers
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).
### Transition Relation

Describes how input events can cause 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
- Exits the output events with the appropriate values

The react time is unbounded and non-zero.

### Input Events

- Trigger events
  - Can be used only once to cause a transition of a given CFSM
  - Each occurrence is committed by the triggered transition
  - Can cause many transitions in different CFSMs
  - 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).
**Example: Formal Description**

The formal description of the same CSFM C = (S, E, O, R, F) is:

1. \( s = (\text{KEY ON, OFF), (BELT ON, OFF), (END 0, 10), (OFF, WAIT, ALARM)} \)
2. \( E = (\text{KEY ON, OFF), (BELT ON, OFF), (END 0, 10)} \)

\( x^2 \) appears in input and output event => state event

\( x^2 \) := pure data event (convention; names not preceded by $^*$)
**Network of CFSMs**

A Network of CFSMs is a set of CFSMs:

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

such that no two different CFSMs have an output event name in common (i.e., + implies that \( \cap C = \emptyset \))

- 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 (potentially require some mutual exclusion mechanism or some notification function)

- Input sets need not be disjoint, thus allowing 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, \( C_0 \), defined as follows:

\[ C_0 = \{ I_0, I_1, I_2, I_3, F_0, F_1, F_2 \} \]

1. \( I = \{ \langle \text{START}, \emptyset \rangle, \langle \text{TICK}, \{ \} \rangle, \langle \emptyset, \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \rangle \} \)
   - Represents an input event from the environment occurring once a second

2. \( O = \{ \langle \text{END}, \emptyset, \{1, 10\} \rangle, \langle \emptyset, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \rangle \} \)

3. \( R = \{ \langle \emptyset, \emptyset \rangle \} \)

**Seat Belt Example**

\[ C = \{ I_0, I_1, I_2, I_3, F_0, F_1, F_2 \} \]

- \( T = \{ \langle \text{TICK}, 0, \{0, 9\} \rangle \}
  - \langle \text{START}, 0, \{0, 9\} \rangle \rightarrow \{0, 9\}, \{10\}
  - \langle \text{TICK}, 0, \{0, 1\} \rangle \rightarrow \{0, 1\}, \{0, 10\}
  - \langle \text{START}, 0, \{0, 2\} \rangle \rightarrow \{0, 2\}, \{0, 10\}
  - \langle \text{TICK}, 0, \{0, 3\} \rangle \rightarrow \{0, 3\}, \{0, 10\}
  - \langle \text{START}, 0, \{0, 4\} \rangle \rightarrow \{0, 4\}, \{0, 10\}
  - \langle \text{TICK}, 0, \{0, 5\} \rangle \rightarrow \{0, 5\}, \{0, 10\}
  - \langle \text{START}, 0, \{0, 6\} \rangle \rightarrow \{0, 6\}, \{0, 10\}
  - \langle \text{TICK}, 0, \{0, 7\} \rangle \rightarrow \{0, 7\}, \{0, 10\}
  - \langle \text{START}, 0, \{0, 8\} \rangle \rightarrow \{0, 8\}, \{0, 10\}
  - \langle \text{TICK}, 0, \{0, 9\} \rangle \rightarrow \{0, 9\}, \{0, 10\} \} \]