

Co-design Finite State Machines

Many slides of this lecture are borrowed from Margarida Jacome

Mahapatra-A&M-Fall'00

1

Summary of Dataflow Network Model

- Partially ordered tags
- Explicit concurrency
- Blocking read (non-reactive)
- Fundamentally deterministic
- No input or output choice

Mahapatra-A&M-Fall'00

2

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.

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.

Mahapatra-A&M-Fall'00

5

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.

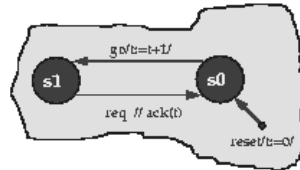
Mahapatra-A&M-Fall'00

6

■■■■ 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
 phases 2/3/4: duration of zero

Magalida Jacome - UT Austin

7

■■■■ The Synchronous Hypothesis

System *instantaneously* reacts to events...

1. The chronometric notion of time is replaced by a notion of *order among events*

only relevant notions are *simultaneity* and *precedence* between events

Magalida Jacome - UT Austin - 1997

■■■■ FSMs and CFSMs

1. Mixed hardware-software systems may contain components that proceed at very different speeds
 - **synchronous hardware modules**
 - execute concurrently
 - compute next state and outputs at each clock cycle
 - **software modules**
 - execute sequentially
 - reaction to conditions may take hundreds of clock cycles to compute and propagate

Magalida Jacome - UT Austin - 1997

8

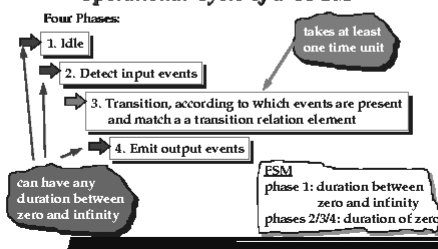
FSMs and CFSMs

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

→ CFSMs: specialized model that incorporates the unbounded delay assumption

Magañda Jacome - UT Austin - 1997

Operational Cycle of a CFSM



Magañda Jacome - UT Austin - 1997

9

CFSM Overview

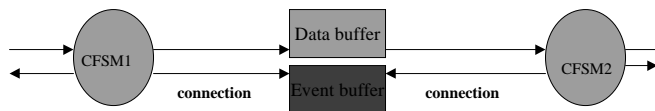
- A **FSM part**: that contains set of inputs, outputs, and states, a transition relation and an output relation.
- A **data computation part**: references in transition relation to external, instantaneous (combinational) function.
- A **locally synchronous** behavior: Execute transition by producing 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*.

Mahapatra-A&M-Fall00

10

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.

Mahapatra-A&M-Fall'00

11

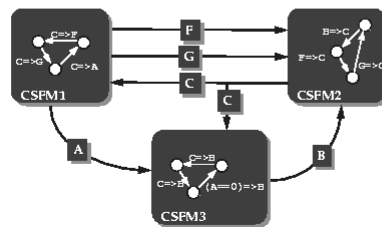
Basics

1. System modeled as a network of interacting CFSMs communicating through events
 - Each CFSM takes a non-zero unbounded time to perform its task
 - at least before an implementation is chosen
1. 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

implicit *one place buffer* between the sender and each receiver saves the event until it is detected (or overwritten)

Mazurka, Jacques - UT Austin - 1997

Network of CFSMs: Depth-1 Buffers



Mazurka, Jacques - UT Austin - 1997

12

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 $e = (e_n, e_v, e_t)$:

- 1. e_n is the name of the event
 - i.e., the "communication port" where it occurs
- 1. $e_v \in e_v$ is the value of the event; (e_v is the set of values the event can take)
- 1. e_t , a non-negative integer, is the time of occurrence of a particular instance of an event

Ex: event with a name "temperature" could occur every time a certain sensor reports a new value, in the range between 0 and 100°C.

⇒ Some events may not have "interesting" values (e.g., reset) – in this case e_v is the special symbol ϵ .

Magedda Jacome - UT Austin - 1997

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.

⇒ input events of the system:

event name	event values
"BELT	ON/OFF
"KEY	ON/OFF

⇒ output events of the system:

event name	event values
"ALARM	ON/OFF

Magedda Jacome - UT Austin - 1997

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.

⇒ internal events of the system (i.e., events exchanged by the system components and not visible outside):

event name	event values
*START	ε ← starting of the timer
*END	5/10 ← elapsed time

Maugesda Jacome - UT Austin - 1997

CFSMs

1. A CFSM is basically constituted by

- a set of **input events**
 - each with its associated set of values
- a set of **output events**
 - each with its associated set of values and possibly with an initial value
- a **transition relation**

The transition relation describes how input events can cause output events.

Maugesda Jacome - UT Austin - 1997

15

Transition Relation

Describes how input events can cause output events.

1. 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
- emits the output events with the appropriate values

The reaction time is **unbounded** and **non-zero**

Maugesda Jacome - UT Austin - 1997

Input Events

1. **Trigger events**
 - can be used *only once* to cause a transition of a given CFSM
 - each occurrence is *consumed* by the triggered transition
 - can cause many transitions in *different* CFSMs
1. **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).

Maugesda Jacome - UT Austin - 1997

16

Input Events

Ex: a given system must *sample the temperature every minute*, and react appropriately.

System can be modeled as a CFSM with two input events: *time (trigger)* and *temperature (pure value)*.

- ⇒ the reaction (CFSM transition) can occur only due to a *time change*
- ⇒ the reaction must take into account the *value of the temperature* event when the time change event occurs.

Modeling both as events allows *some other system component* to react to *temperature changes* rather than time changes

Margaida jacome - UT Austin - 1997

States

1. the *state* of a CFSM 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*.



CFSMs: *reaction time* is unbounded and *non-zero*

Margaida jacome - UT Austin - 1997

17

CFSMs

A CFSM is a quintuple $C = (I, E, O, R, F)$:

1. $I = \{(I_n, I_v), (I_n, I_v), \dots\}$ is a finite set of *input event names* and of the corresponding *finite set of allowed values*
1. $E, I \supseteq E$, is the set of "trigger" *input event names*
Events with names in $(I - E)$ are "pure data" events
1. $O = \{(O_n, O_v), (O_n, O_v), \dots\}$ 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 a trigger input and an output)
1. $R, \{(e_n, e_v) \mid (e_n, e_v) \in O, e_n \in E\} \supseteq R$, is a set of possible *initial values* of (some) output events
1. $F, \{(f, f_v) \mid f = \{(e_n, e_v) \mid (e_n, e_v) \in I, e_n \in E\}, f_v = \{(e_n, e_v) \mid (e_n, e_v) \in O, e_n \in E\}\} \supseteq F$, is the *transition relation*
• for all $(f, F) \in F$ there must exist at least one $(I_n, I_v) \in E, I_n \in I_v$ such that $(I_n, I_v) \in F$ (i.e., at least one trigger event)

Margaida jacome - UT Austin - 1997

Seat Belt Example Revisited

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

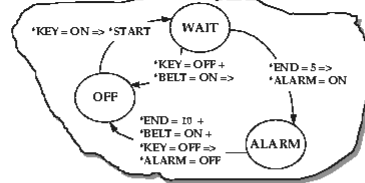
⇒ input events:	event name	event values
	*BELT	ON/OFF
	*KEY	ON/OFF
⇒ output events:	event name	event values
	*ALARM	ON/OFF
⇒ internal events:	event name	event values
	*START	s
	*END	S/10

Margaida jacome - UT Austin - 1997

18

Seat Belt Example

A CFSM describing the desired event/reaction pattern:



+ denotes the logic or condition
=> separates input and output events of a given transition

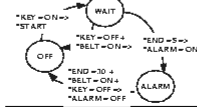
Margaida jacome - UT Amstn - 1997

Example: Formal Description

The formal description of the same CFSM

$C_1 = (I_1, E_1, O_1, R_1, F_1)$ is

- 1 $I_1 = \{(*KEY, (ON, OFF)), (*BELT, (ON, OFF)), (*END, (5, 10)), (s_1, (OFF, WAIT, ALARM))\}$
- 1 $E_1 = \{(*KEY, (ON, OFF)), (*BELT, (ON, OFF)), (*END, (5, 10))\}$



s_1 => pure data event
(convention: name not preceded by **)

Margaida jacome - UT Amstn - 1997

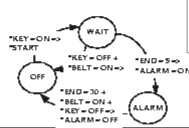
19

Example: Formal Description (cont.)

$C_1 = (I_1, E_1, O_1, R_1, F_1)$:

- 1 $O_1 = \{(*START, \{s_1\}), (*ALARM, \{ON, OFF\}), (s_1, \{OFF, WAIT, ALARM\})\}$

initial values \Rightarrow 1 $R_1 = \{(s_1, OFF)\}$



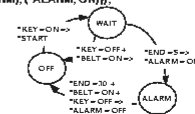
s_1 : appears as input and output event => state event

Margaida jacome - UT Amstn - 1997

Example: Formal Description (cont.)

$C_1 = (I_1, E_1, O_1, R_1, F_1)$:

- 1 $F_1 = \{$
- $\{(*KEY, ON), (s_1, OFF)\} \Rightarrow \{(s_1, WAIT), (*START, s_1)\}$,
- $\{(*KEY, ON), (*BELT, ON), (s_1, OFF)\} \Rightarrow \{(s_1, OFF)\}$,
- $\{(*KEY, OFF), (s_1, WAIT)\} \Rightarrow \{(s_1, OFF)\}$,
- $\{(*BELT, ON), (s_1, WAIT)\} \Rightarrow \{(s_1, OFF)\}$,
- $\{(*END, 5), (s_1, WAIT)\} \Rightarrow \{(s_1, ALARM), (*ALARM, ON)\}$,
- $\{(*END, 10), (s_1, ALARM)\} \Rightarrow$
- $\{(s_1, OFF), (*ALARM, OFF)\}$,
- $\{(*BELT, ON), (s_1, ALARM)\} \Rightarrow$
- $\{(s_1, OFF), (*ALARM, OFF)\}$,
- $\{(*KEY, OFF), (s_1, ALARM)\} \Rightarrow$
- $\{(s_1, OFF), (*ALARM, OFF)\}$
- $\}$



Margaida jacome - UT Amstn - 1997

20

Network of CFSMs

A Network of CFSMs is a set of CFSMs

$N = \{C_1 = (I_1, E_1, O_1, R_1, F_1), C_2 = (I_2, E_2, O_2, R_2, F_2), \dots\}$

such that no two different CFSMs have an output event name in common (i.e., $i \neq j$ implies that $O_i \cap O_j = \emptyset$)

⇒ 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* (would require some mutual exclusion mechanism or some resolution function)

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

Margaida jacome - UT AmstU - 1997

Seat Belt Example

The network of CFSMs would be composed by C_1 plus a CFSM implementing the timer, C_2 , defined as follows:

$C_2 = (I_2, E_2, O_2, R_2, F_2)$:

1 $I_2 = \{(*START, \{e\}), (*TICK, \{e\}), (s_2, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\})\}$

represents an input event from the environment occurring once a second

1 $E_2 = \{(*START, \{e\}), (*TICK, \{e\})\}$

1 $O_2 = \{(*END, \{5, 10\}), (s_2, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\})\}$

1 $R_2 = \{(s_2, 0)\}$

Margaida jacome - UT AmstU - 1997

21

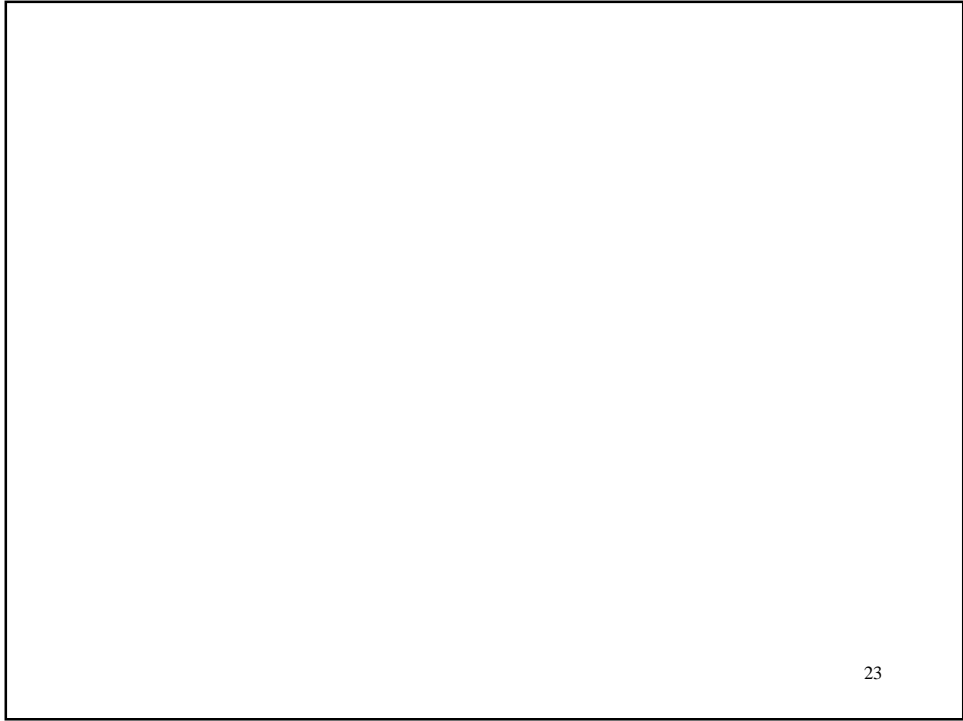
Seat Belt Example

$C_2 = (I_2, E_2, O_2, R_2, F_2)$:

1 $F_2 = \{$
 $\{(*TICK, e), (s_2, 0)\} \Rightarrow \{(s_2, 1)\}$,
 $\{(*START, e), (*TICK, e), (s_2, 0)\} \Rightarrow \{(s_2, 0)\}$,
 $\{(*TICK, e), (s_2, 1)\} \Rightarrow \{(s_2, 2)\}$,
 $\{(*START, e), (*TICK, e), (s_2, 1)\} \Rightarrow \{(s_2, 0)\}$,
 $\{(*TICK, e), (s_2, 2)\} \Rightarrow \{(s_2, 3)\}$,
 $\{(*START, e), (*TICK, e), (s_2, 2)\} \Rightarrow \{(s_2, 0)\}$,
 \dots
 $\{(*TICK, e), (s_2, 4)\} \Rightarrow \{(s_2, 5), (*END, 5)\}$,
 $\{(*START, e), (*TICK, e), (s_2, 4)\} \Rightarrow \{(s_2, 0), (*END, 5)\}$,
 \dots
 $\{(*TICK, e), (s_2, 9)\} \Rightarrow \{(s_2, 0), (*END, 10)\}$,
 $\}$

Margaida jacome - UT AmstU - 1997

22



23