



Administrative Matters

- Lecture notes posted online
- Presentation sign-up
 - 1/17: Ryan
 - 1/19: Brett
- Every one else: Send me e-mail on time slot and article of choice by Tuesday, 1/17, 10AM




CS122B: Adv. Embedded and Real-Time Systems, Winter 06
1

Embedded System Specification

2


Requirements for specification techniques

- **Hierarchy**
Humans not capable to understand systems containing more than ~5 objects.


Most actual systems require more objects


☞ Hierarchy

- Behavioral hierarchy
Examples: states, processes, procedures.
- Structural hierarchy
Examples: processors, racks, printed circuit boards






proc
proc
proc





CS122B: Adv. Embedded and Real-Time Systems, Winter 06
3

Requirements for specification techniques

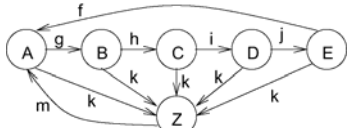
- **Timing behavior.**
- **State-oriented behavior**
Required for reactive systems; classical automata insufficient.
- **Event-handling**
(external or internal events)
- **No obstacles for efficient implementation**







CS122B: Adv. Embedded and Real-Time Systems, Winter 06
4

Requirements for specification techniques


- **Support for the design of dependable systems**
Unambiguous semantics, ...
- **Exception-oriented behavior**
Not acceptable to describe exceptions for every state.





CS122B: Adv. Embedded and Real-Time Systems, Winter 06
5




Requirements for specification techniques

- **Concurrency**
Real-life systems are concurrent
- **Synchronization and communication**
Components have to communicate!
- **Presence of programming elements**
For example, arithmetic operations, loops, and function calls should be available
- **Executability**
- **Support for the design of large systems** (☞ OO)
- **Domain-specific support**







CS122B: Adv. Embedded and Real-Time Systems, Winter 06
6

Requirements for specification techniques

- **Readability** 
- **Portability and flexibility** 
- **Termination**
It should be clear, at which time all computations are completed 

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Requirements for specification techniques

- **Support for non-standard I/O devices**
Direct access to switches, displays, ... 
- **Non-functional properties**
fault-tolerance, disposability, EMC-properties, weight, size, user friendliness, extendibility, expected life time, power consumption... 
- **Adequate model of computation** 


CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Models of computation - Definition -

- **Models of computation define:**
 - What it means to be a **component**:
Subroutine? Process? Thread?
 - The mechanisms by which components **interact**:
Message passing? *Rendez-vous*?
 - Possibly also:
What components know about each other
(global variables? Implicit behavior of other components)

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Models of computation - Examples (1) -


- **Communicating finite state machines (CFSMs):** 
- **Discrete event model**

a	6
b	7
c	8

←

queue				
5	10	13	15	19
a:=5	b:=7	c:=8	a:=6	a:=9


time
action



CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Models of computation - Examples (2) -

- **Differential equations**

$$\frac{\partial^2 x}{\partial t^2} = b$$


CS122B: Adv. Embedded and Real-Time Systems, Winter 06 ≠ Ptolemy simulations?

An prototypical example

- **Given a differential equation: $y'' + 3xy' + 3y = 0$**
 - Solve it for $x = [0, a]$ with step size dx
 - Initial value $x(0)=x, y(0)=y, y'(0)=u$
 - Using Forward Euler Method

```

diffeq {
  read ( x, y, u, dx, a );
  repeat {
    x1 = x + dx;
    u1 = u - ( 3 * x * u * dx ) - ( 3 * y * dx );
    y1 = y + u * dx;
    c = x1 < a;
    x = x1 ; u = u1 ; y = y1;
  }
  until ( c );
  write ( y );
}
    
```

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

A possible Architecture

```

diffEq {
  read ( x, y, u, dx, a );
  repeat {
    x1 = x + dx;
    u1 = u - ( 3 * x * u * dx ) - ( 3 * y * dx );
    y1 = y + u * dx;
    c = x1 < a;
    x = x1 ; u = u1 ; y = y1;
  }
  until ( c );
  write ( y );
}
    
```

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Dataflow graph

- Vertices = operations
- Edges = data dependencies

$x1 = x + dx;$
 $u1 = u - (3 * x * u * dx) - (3 * y * dx);$
 $y1 = y + u * dx;$
 $c = x1 < a;$

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Different Method of Communication

- Asynchronous message passing
- Synchronous message passing

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Recap of classical FSM

- Classical state machine:

input X → Internal state Z → output Y
 clock →

Next state Z^+ computed by function δ
 Output computed by function λ

- Moore-machine:
 $Y = \lambda (Z); Z^+ = \delta (X, Z)$
- Mealy-machine
 $Y = \lambda (X, Z); Z^+ = \delta (X, Z)$

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

StateCharts

- Classical automata not useful for complex systems (complex graphs cannot be understood by humans).
- Introduction of hierarchy StateCharts [Harel, 1987]

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Introducing hierarchy

FSM will be in exactly one of the substates of S if S is active (either in A or in B or ..)

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Definitions

- Current states = active states.
- Basic States
 - States which are not composed of other states
- Super-states
 - States containing other states
- Ancestor States
 - the super-states containing the given state
- OR-super-states (a.k.a. XOR-super-states)
 - if exactly one of the sub-states of S is active whenever S is active.

19

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Default state mechanism

- hide internal structure from outside world!
- Default state
 - sub-state entered whenever super-state is entered.

20

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

History mechanism

- For input m, S enters state it was in before S was left
 - can be A, B, C, D, or E
 - If S is entered for the first time, the default applies.

21

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Combining history and default state mechanism

22

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Concurrency

- **AND-super-states:** FSM is in **all** (immediate) sub-states of a super-state; Example:

23

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Entering and leaving AND-super-states

- Line-monitoring and key-monitoring are entered and left, when service switch is operated.

24

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Types of states

- In StateCharts, states are either
- **basic states, or**
- **AND-super-states, or**
- **OR-super-states.**

25

Timers

- Special edges are used for timeouts.

If event a does not happen while the system is in the left state for 20 ms, a timeout will take place.

26

Using timers in answering machine

27

General form of edge labels

● event [condition] / reaction ●

- **Events:**
 - Exist only until the next evaluation of the model
 - Can be either internally or externally generated
- **Conditions:**
 - Refer to values of variables that keep their value until they are reassigned
- **Reactions:**
 - Can either be assignments for variables or creation of events
- **Example:**
 - service-off [not in Lproc] / service:=0

28

The StateCharts simulation phases (StateMate Semantics)

● event [condition] / reaction ●

- How are edge labels evaluated?
 - Three phases:
 1. Effect of external changes on events and conditions is evaluated,
 2. The set of transitions to be made in the current step and right hand sides of assignments are computed,
 3. Transitions become effective, variables obtain new values.
- Separation into phases 2 and 3 guarantees deterministic and reproducible behavior.

29

Example

- In phase 2,
 - variables a and b are assigned to temporary variables.
- In phase 3,
 - these are assigned to a and b.
- As a result, variables a and b are swapped.

30

Example

- In a single phase environment,
 - Executing the left state first would assign the old value of b (=0) to a and b.
 - Executing the right state first would assign the old value of a (=1) to a and b.
 - The execution would be non-deterministic.

31

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Steps

- Execution of a StateChart model consists of a sequence of (status, step) pairs

Status = values of all variables + set of events + current time
 Step = execution of the three phases (StateMate semantics)

32

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Broadcast mechanism

- Values of variables are visible to all parts of the StateChart model.
- New values
 - become effective in phase 3 of the current step
 - are obtained by all parts of the model in the following step.

☞ StateCharts implicitly assumes a **broadcast** mechanism for variables.

☞ StateCharts is appropriate for local control systems (☺),

☞ but not for distributed applications for which updating variables might take some time (☹).

33

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Evaluation of StateCharts (1)

- **Pros:**
- Hierarchy allows arbitrary nesting of AND- and OR-super states.
- (StateMate-) Semantics defined in a follow-up paper to original paper.
- Large number of commercial simulation tools available
 - StateMate (ilogic), StateFlow (mathworks), BetterState (windriver)
- Available back-ends translation into C or VHDL

34

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Evaluation of StateCharts (2)

- **Cons:**
- Generated C programs frequently inefficient,
- Not useful for distributed applications,
- No program constructs,
- No description of non-functional behavior,
- No object-orientation,
- No description of structural hierarchy.

35

CS122B: Adv. Embedded and Real-Time Systems, Winter 06

Summary

- **Requirements for specification languages**
 - Hierarchy
 - Timing behavior
 - State-oriented behavior
 - Concurrency
 - Synchronization & communication, ...
- **Models of computation**
- **StateCharts**
 - AND-states
 - OR-states
 - Timer
 - Broadcast
 - Semantics
 - ...

36

CS122B: Adv. Embedded and Real-Time Systems, Winter 06