Hierarchical State Machines; Synchronous Dataflow

CS 447– Wireless Embedded Systems
Outline

• Overview
• Depth-First reactions
• Reset transitions
• Preemptive transitions
• History transitions
• Synchronous dataflow
Overview

Key idea: **state refinement**
- E.g., state B is another FSM with two states, C and D
Overview

- When machine in state B, it’s actually in state C or D
Overview

- Starts in state A
- $g_2$ evaluates to true
- outputs $a_2$
- moves state B
- (actually goes to state C)
Overview

• In state C
• $g_4$ evaluates to true
• outputs $a_4$
• moves to D
Overview

- In state D
- $g_3$ evaluates to $true$
- outputs $a_3$
- moves to C
Overview

• In state C or D
• $g_1$ evaluates to true
• outputs $a_1$
• moves to state A
Outline

• Overview
• **Depth-First reactions**
• Reset transitions
• Preemptive transitions
• History transitions
• Synchronous dataflow
Depth-First Reactions

• Variant of hierarchical finite state machine
• *Deepest refinement reacts first*
• Then container state machine
• Then its container
• ...
**Depth-First Reactions**

**Depth-First** reaction: refinement reacts first, e.g.,
- machine in state C (within B)
- $g_4$ true
- outputs $a_4$
- moves to D
Depth-First Reactions

**Depth-First** reaction: refinement reacts first, e.g.,

- machine in state C (within B)
- $g_4$ and $g_1$ true
- outputs $a_4$; $a_1$
- moves to D A

- Logically, both transitions are *simultaneous* and *instantaneous*
Depth-First Reactions

What if actions conflict? E.g.,
• Two actions write to same output port
• Set same variable to different values

One choice: perform actions in **depth-first** sequence
Overview

• In state C
• Both $g_4$ and $g_1$ true, what happens?
• Based on priority
• output $a_4$
• move to D
• output $a_1$
• move to A
Outline

• Overview
• Depth-First reactions
• **Reset transitions**
• Preemptive transitions
• History transitions
• Synchronous dataflow
Overview

The “hollow arrow” indicates reset transition
• Whenever machine enters B
• It always “resets” to C
• Never D
Overview

What happens when:

• Start in state A
• Input: $g_2 \ g_4 \ g_1 \ g_2$

• What’s the output?
• What’s the current state?
  • C or D?
Overview

• Start in state A
• Input: \( g_2 \ g_4 \ g_1 \ g_2 \)
• Output: \( a_2 \ a_4 \ a_1 \ a_2 \)
• States: C, D, A, C
• Current state: C

• **NOTE: transition from A \( \rightarrow \) B “resets” the inner state machine**
  • Always restarts in state C
  • This machine cannot go from A to D
Overview

• Semantics:
Outline

• Overview
• Depth-First reactions
• Reset transitions
• Preemptive transitions
• History transitions
• Synchronous dataflow
Preemptive Transitions

Conflicts can be avoided with preemption

- Guards of preemptive transitions evaluated before refinement

- If guard is true for preemptive transition, refinement does NOT react
Preemptive Transitions

Preemptive transitions indicated with red circle

- E.g., *preemptive transition* from **B** to **A**
Preemptive Transitions

• In state C (or D)
• \( g_1 \) true
• output \( a_1 \)
• move to A

• Regardless of whether \( g_4 \) (or \( g_3 \)) is true

• Inner FSM gets preempted
Outline

• Overview
• Depth-First reactions
• Reset transitions
• Preemptive transitions
• History transitions
• Synchronous dataflow
History Transitions

• When previous state of refinement is “remembered”
• Indicated by solid arrow or arrow with an ‘H'
History Transitions

- Now FSM has **history transition** from A to B
- Transition A to B, must remember refinement state (C or D)
History Transitions

• Semantics more complicated:
Brief Review

• Depth-first reaction by default (inner refinement reacts first)

• **Reset transition** – hollow arrow

• **Preemptive transition** – red circle

• **History transition** – solid arrow or $H$
Outline

- Overview
- Depth-First reactions
- Reset transitions
- Preemptive transitions
- History transitions
- Synchronous dataflow
Synchronous Dataflow

Constrained form of data flow for each actor

Every firing, each actor:
• Consumes fixed number of input tokens on input port
• Produces fixed number of output tokens on output port
Synchronous Dataflow

E.g., single connection between actors A and B

- A fires: produces M tokens on output port
- B fires: consumes N tokens on input port
- (M, N are positive integers)
Synchronous Dataflow

All tokens produced by A are consumed by B iff balance equation is satisfied: \( q_A^M = q_B^N \)

- \( q_A^M := \) number of times A fires
- \( q_B^M := \) number of times B fires
Synchronous Dataflow

If equation balanced for some positive integers $q_A$, $q_B$

• Then we can repeat *forever* with **bounded** buffers

![Diagram of synchronous dataflow](image)
Synchronous Dataflow

E.g., $M = 2$, $N = 3$

- $q_A = 3$, $q_B = 2$ is solution

- Possible firing sequences: AAABB, AABAB
Synchronous Dataflow

Possible firing sequences: AAABB, AABAB

• Which sequence requires more memory?
Synchronous Dataflow

Possible firing sequences: **AAABB, AABAB**
- Which sequence requires more memory? **AAABB**
- AAABB requires 6 token buffer
- AABAB only requires 4 token buffer
Synchronous Dataflow

More complicated synchronous dataflow models are common
• Every connection between actors results in balance equation
• Model defines **system of equations**

E.g.,
• 3 actors A, B, C
• 3 connections x, y, z
Synchronous Dataflow

Results in system of balance equations

- $q_A = q_B$
- $2q_B = q_C$
- $2q_A = q_C$

January 29, 2018
Synchronous Dataflow

Results in system of balance equations

- \( q_A = q_B \)
- \( 2q_B = q_C \)
- \( 2q_A = q_C \)
Synchronous Dataflow

Results in system of balance equations

- $q_A = q_B$
- $2q_B = q_C$
- $2q_A = q_C$
Synchronous Dataflow

Results in system of balance equations

- $q_A = q_B$
- $2q_B = q_C$
- $2q_A = q_C$
Synchronous Dataflow

System of balance equations:

- $q_A = q_B$
- $2q_B = q_C$
- $2q_A = q_C$

Least positive integer solution?

- $q_A = q_B = 1$
- $q_C = 2$
Synchronous Dataflow

Least positive integer solution?
• $q_A = q_B = 1$
• $q_C = 2$

How many times must each actor fire to reach steady state?
• Possible sequence: $ABCC$
• Can repeat forever w/ **bounded** buffers
Synchronous Dataflow

Balance equations DO NOT always have non-trivial solution

• What’s the system of balance equations for this model?
What’s the system of balance equations for this model?

- \( q_A = q_B \)
- \( q_B = q_C \)
- \( 2q_A = q_C \)
Synchronous Dataflow

Positive integer solution for this system of equations?
• $q_A = q_B$
• $q_B = q_C$
• $2q_A = q_C$

**NOPE!** Only $q_A = q_B = q_C = 0$
• *With bounded buffers, no unbounded execution!*
• Cannot be kept in balance
• Will eventually overflow
Synchronous Dataflow (SDF)

**Consistent** – SDF model w/ positive integer (non-zero) solution to balance equation(s)

**Inconsistent** – only solution is zero
- No unbounded execution with bounded buffers
Synchronous Dataflow (SDF)

Important: **consistent** SDF model can ensure **bounded buffers**
- **BUT** – it *cannot guarantee unbounded execution*
- Deadlock may still occur..