Embedded System Design and Modeling

VII. Hierarchical State Machines

Hierarchical state machines

Hierarchical state machines:

- A state in a top-level FSM can be implemented (“refined”) as an internal/embedded state machine
  - The top level state = “super-state”
  - An internal state inside it = “sub-state”

Problems:

- Which sub-state is entered?
- What transitions are executed and in what order?

Example

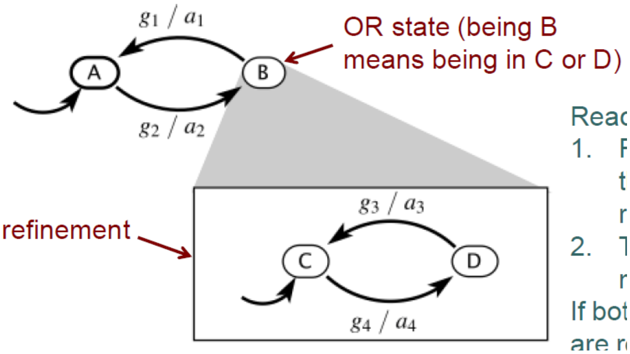


Figure 1: State refinement

- If g1 and g3 are both true, which reacts first? The inner FSM or the outer FSM?

Reaction order

Two solutions:

- 1. [Statecharts language] Inner FSM reacts first, outer FSM reacts later
  - The two reactions are considered simultaneous
  - The output actions are required to not conflict
- 2. [Stateflow, Matlab] Outer FSM reacts first, inner FSM reacts later (if at all)
  - If state is left, the inner FSM will not react at all

Reaction order

Specify here the order of checks/operations in both cases

Reaction order

Specify here the order of checks/operations in both cases

History transitions

When entering a super-state, which sub-state is entered?

Two solutions:

- 1. Enter the last sub-state you were in, when you last left the super-state
    - Represented as a **history transition** (marked with a full black arrow on these schematics / a H sign in Matlab)
- 2. Enter the default sub-state every time
    - Known as a **reset transition** (marked with a white arrow on these schematics / default behavior in Matlab)

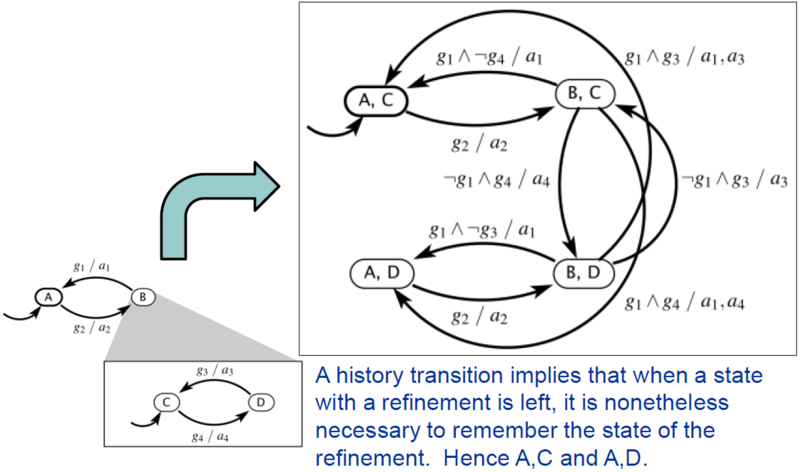
Example

Equivalent flattened FSM

- Any hierarchical FSM can be “flattened”, e.g. converted into an equivalent model with no super-states
  - e.g. Super-state A with two substates B and C is split into to substates AB and AC, transitions from A now leaving from both AB and AC
- Hierarchy in models brings representation efficiency

Example

Flattening the state machine
(assuming history transitions):


g_1 / a_1
A
B
g_2 / a_2
g_3 / a_3
C
D
g_4 / a_4
$g_1 \wedge \neg g_4$ / a_1
$g_1 \wedge g_3$ / a_1, a_3
A, C
B, C
g_2 / a_2
$\neg g_1 \wedge g_4$ / a_4
$\neg g_1 \wedge g_3$ / a_3
$g_1 \wedge \neg g_3$ / a_1
A, D
B, D
g_2 / a_2
$g_1 \wedge g_4$ / a_1, a_4

A history transition implies that when a state with a refinement is left, it is nonetheless necessary to remember the state of the refinement. Hence A,C and A,D.

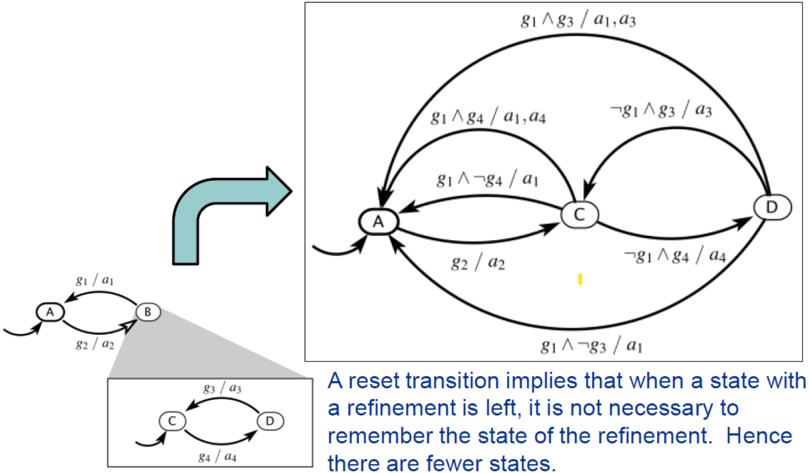
Figure 2: Flattenning example

Example

Redraw here

Example

Flattening the state machine
(assuming reset transitions):


g_1 / a_1
A
B
g_2 / a_2
g_3 / a_3
C
D
g_4 / a_4
$g_1 \wedge g_3 / a_1, a_3$
$g_1 \wedge g_4 / a_1, a_4$
$\neg g_1 \wedge g_3 / a_3$
$g_1 \wedge \neg g_4 / a_1$
A
C
D
g_2 / a_2
$\neg g_1 \wedge g_4 / a_4$
$g_1 \wedge \neg g_3 / a_1$

A reset transition implies that when a state with a refinement is left, it is not necessary to remember the state of the refinement. Hence there are fewer states.

Figure 3: Flattenning example

Example

Redraw here