HCI: Dialog Design (State-Charts)

Learning Objective

- In the previous lecture, we introduced the need for dialog design
- We also learned about the advantages about formal modeling of dialogs
- We discussed how to use STNs for the purpose

Learning Objective

- As we mentioned, STNs are good for modeling simple systems; for complex systems as well as systems having concurrency, STNs fail
- In this lecture, we shall learn about the State-Chart formalism that can overcome the problems with STNs

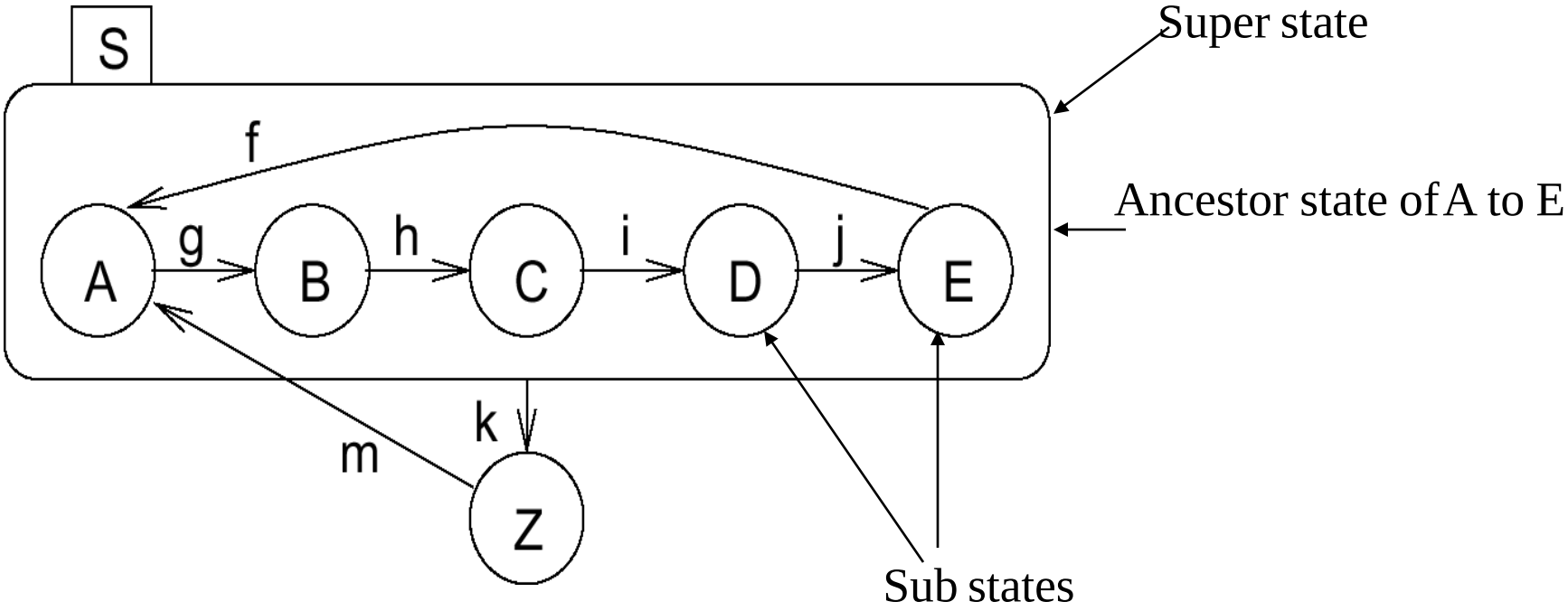
State-Charts

- Proposed by David Harel (1987) to represent complex *reactive* systems
- Extends finite state machines (FSM)
  - Better handle concurrency
  - Adds memory, conditional statements to FSM
- Simplifies complex system representation (states and arcs) to a great extent

Definitions

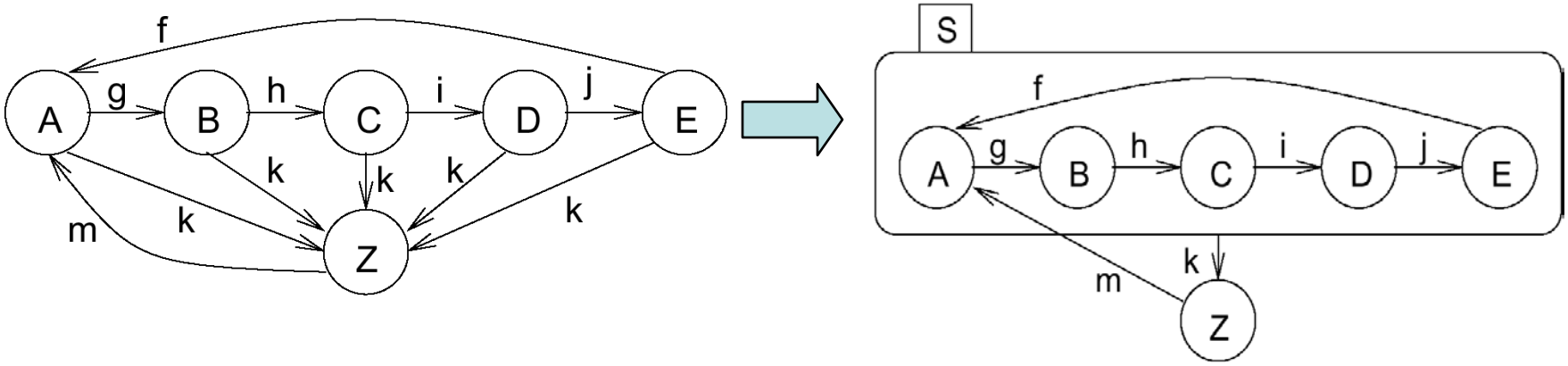
- **Active state**: the current state of the underlying FSM
- **Basic states**: states that are not composed of other states
- **Super states**: states that are composed of other states
  - For each basic state *b*, the super state containing *b* is called the *ancestor* state
  - A super state is called OR super state if exactly one of its sub states is active, whenever it is active

Definitions



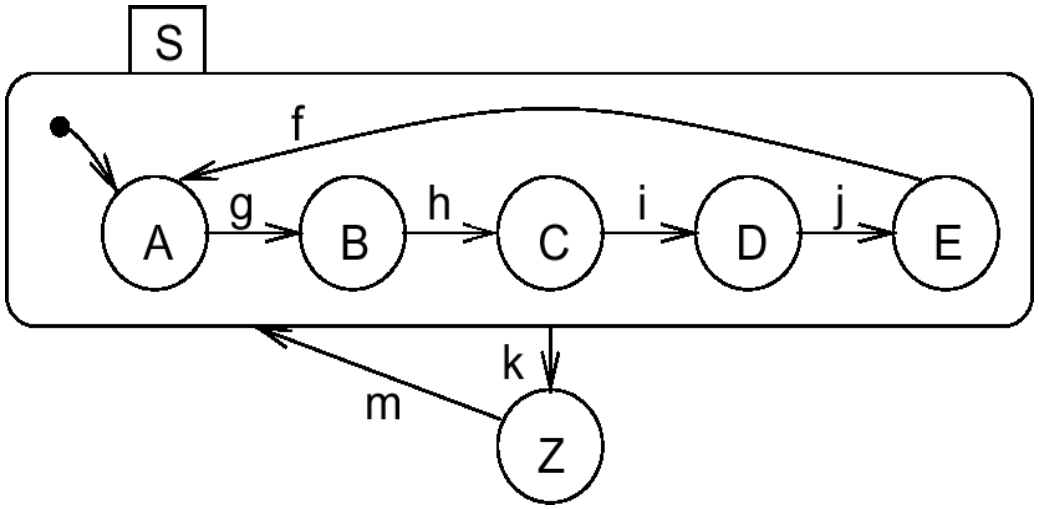
Super State Advantage

- It allows us to represent complex FSM in a nice way, by clustering states



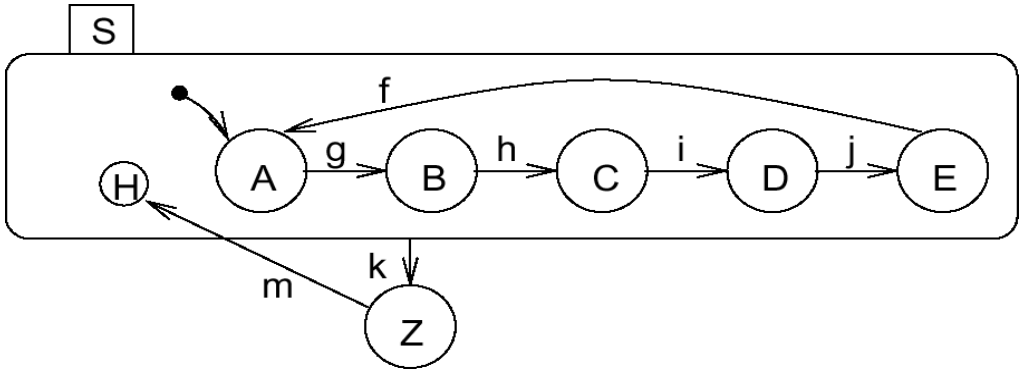
Default State Mechanism

- Indicates the sub state entered whenever super state is entered – represented using a filled circle
  - Not a state by itself

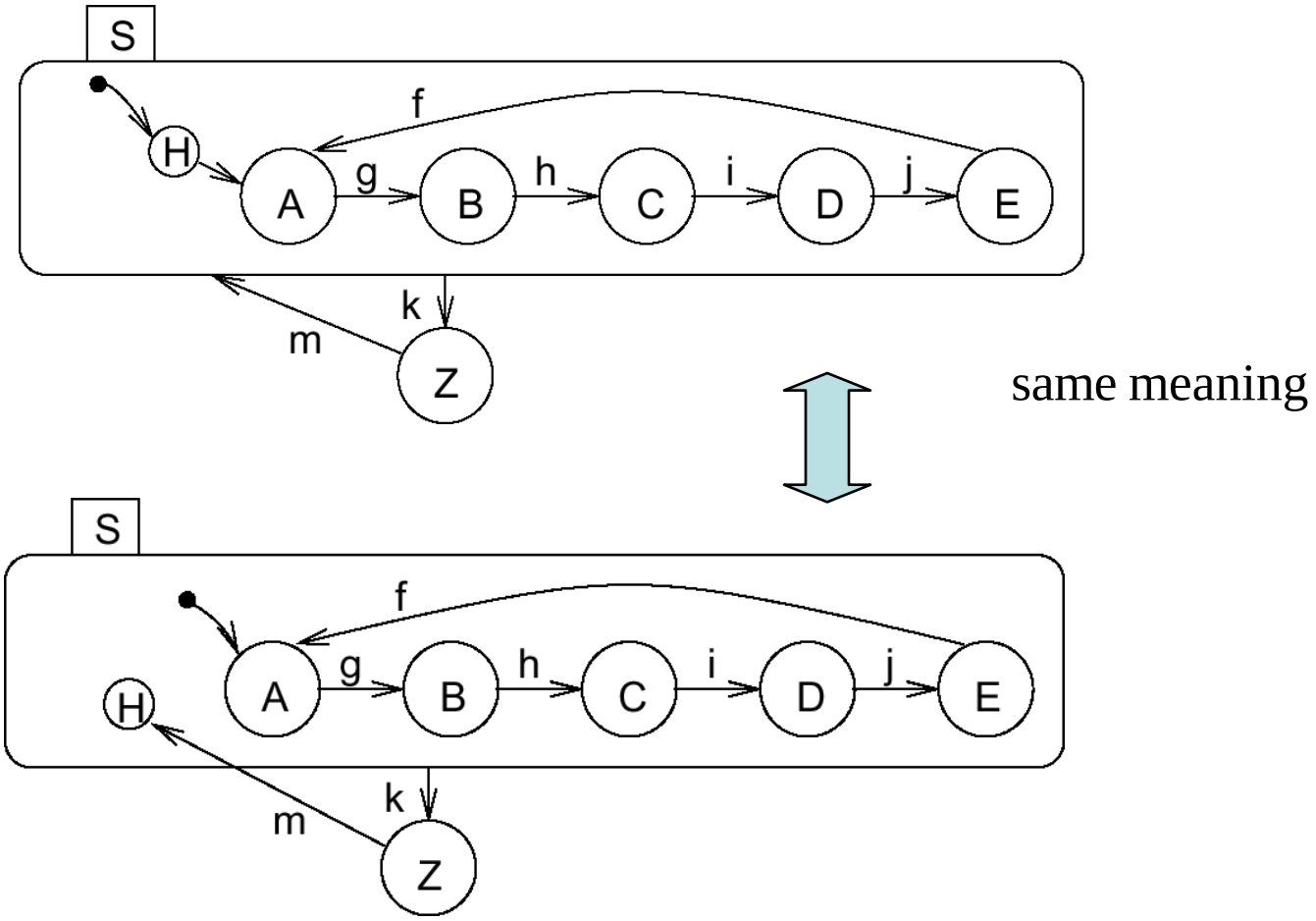


History Mechanism

- For input m, S enters the state it was in before S was left
  - If S is entered for the very first time, the default mechanism applies
  - History and default mechanisms can be used hierarchically

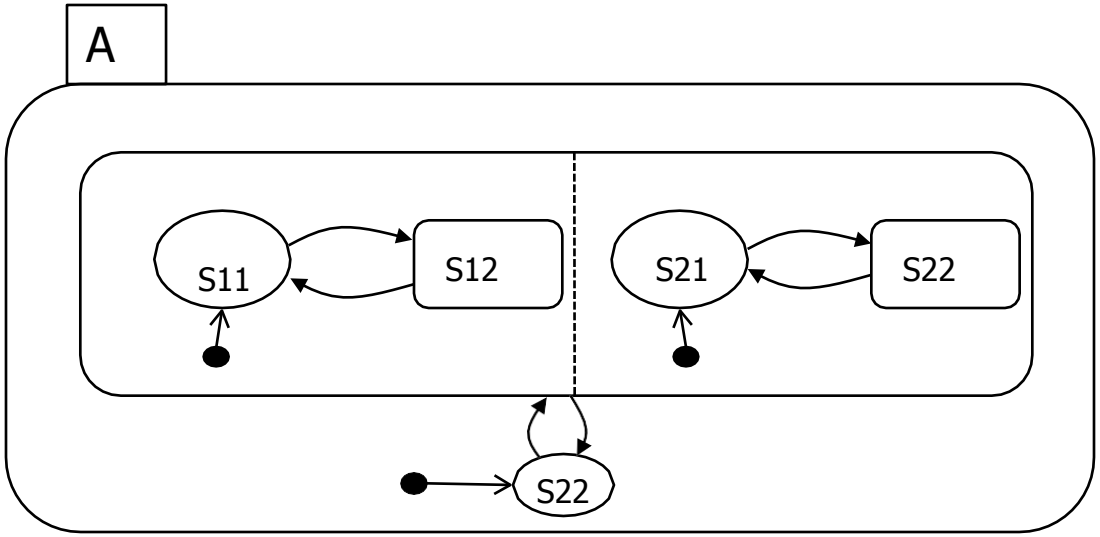


Combining History and Default State Mechanism



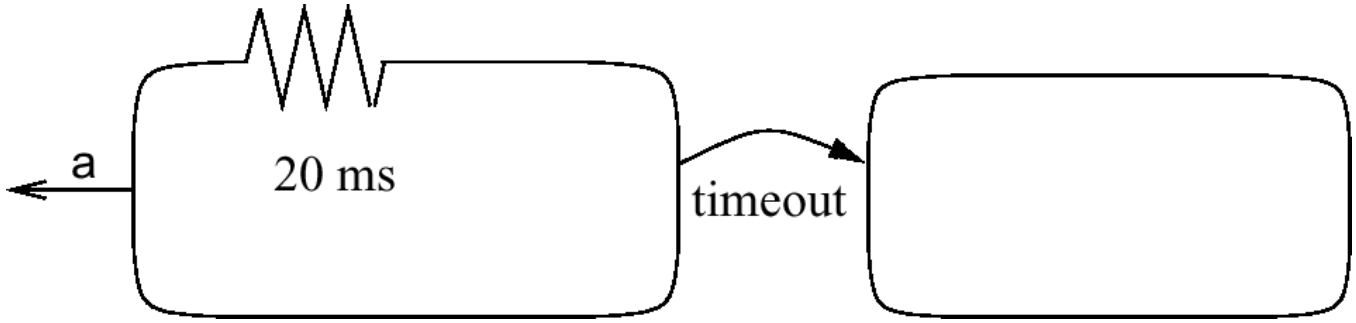
Concurrency

- State-Charts supports concurrency using the notion of the AND super states
  - In AND super states, the FSM is active in all (immediate) sub states simultaneously



Timing Constraints

- State-Chart supports delay/timeout modeling
  - using special edges
    - Do we need it??



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

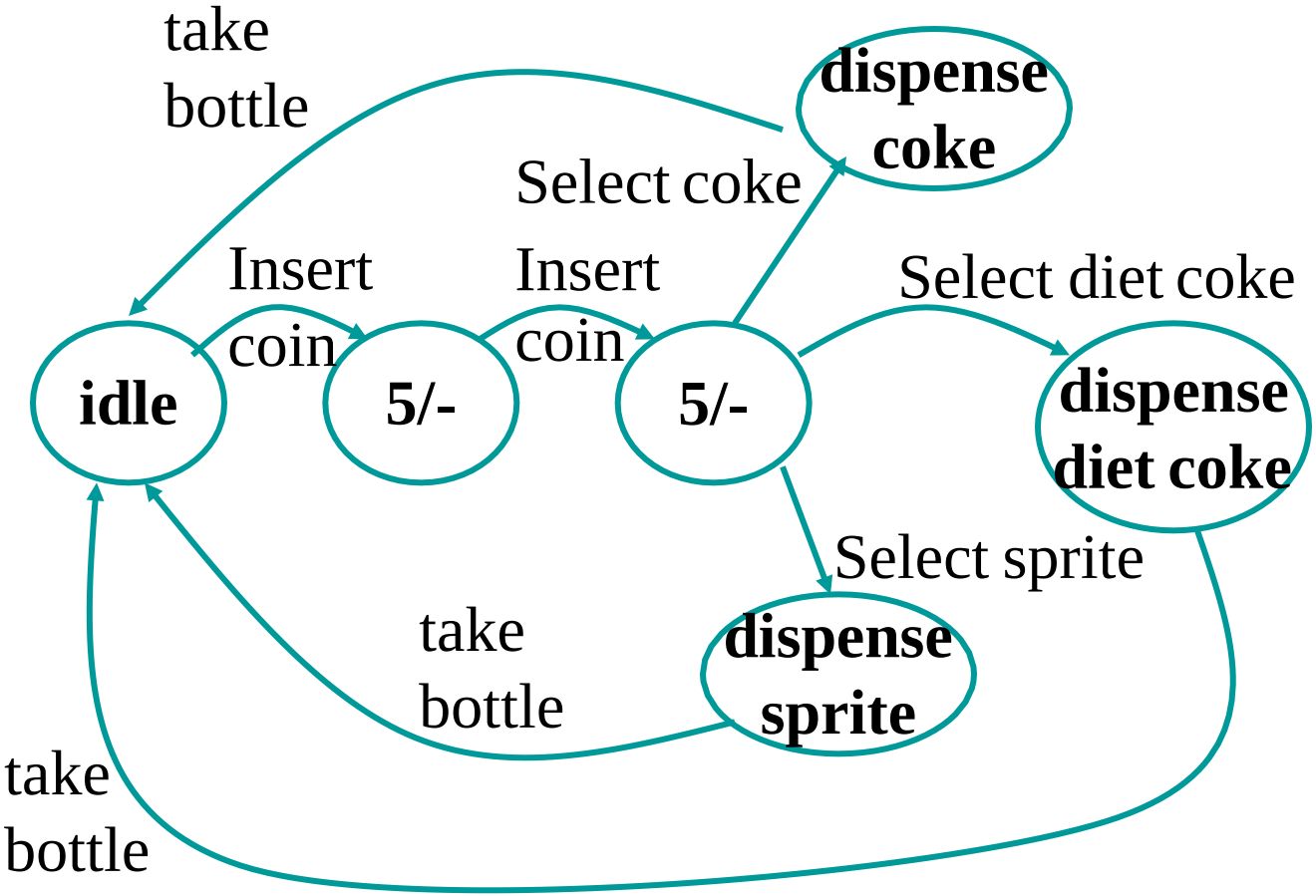
Example: Vending Machine v1.0

- Suppose we have a juice/beverages vending machine:
  - When turned on, this vending machine waits for money
  - When Rs. 5/- coin is deposited, the machine waits for another Rs. 5/- coin
  - When the second coin is deposited, the machine waits for a selection
  - When the user presses “COKE,” a coke is dispensed

Example: Vending Machine v1.0

- Suppose we have a juice/beverages vending machine:
  - When the user takes the bottle, the machine waits again
  - When the user presses either “SPRITE” or “DIET COKE,” a Sprite or a Diet Coke is dispensed
  - When the user takes the bottle, the machine waits again
- Let us represent this behavior using FSM

Vending Machine v1.0



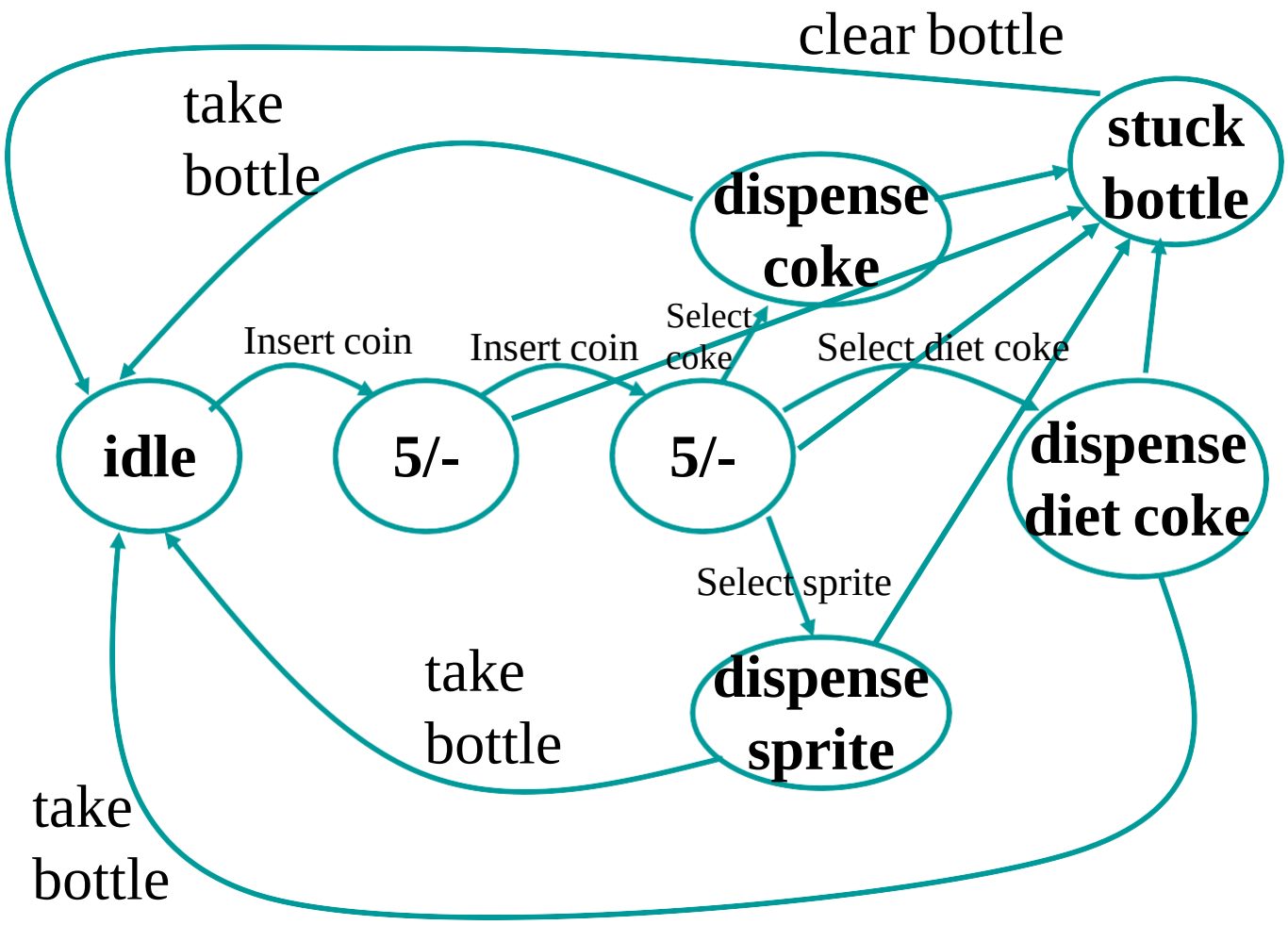
Vending Machine v2.0

- Let us include some more features in the vending machine
- The Bottles can get stuck in the machine
  - An automatic indicator will notify the system when a bottle is stuck
  - When this occurs, the machine will not accept any money or issue any bottles until the bottle is cleared
  - When the bottle is cleared, the machine will wait for money again

Vending Machine v2.0

- State machine changes
  - How many new states are required?
  - How many new transitions?

Vending Machine v2.0



Vending Machine v3.0

- Let us add some more features in the vending machine
- Bottles sometimes shake loose
  - An additional, automatic indicator will indicate that the bottle is cleared
  - When the bottles are cleared, the machine will return to the same state it was in before the bottle got stuck

Vending Machine v3.0

- State machine changes
  - How many new states are required?
  - How many new transitions?

Vending Machine v4.0

- We can add even more features
- Automatic bottle filler
  - If a button is pressed, the machine will toggle between bottle filling and dispensing modes
  - When in bottle filling mode
    - Bottles may be inserted if the Coke/Diet Coke/Sprite machine is ready
    - When a bottle is inserted, the machine will NOT be ready to accept another bottle and will check the bottle

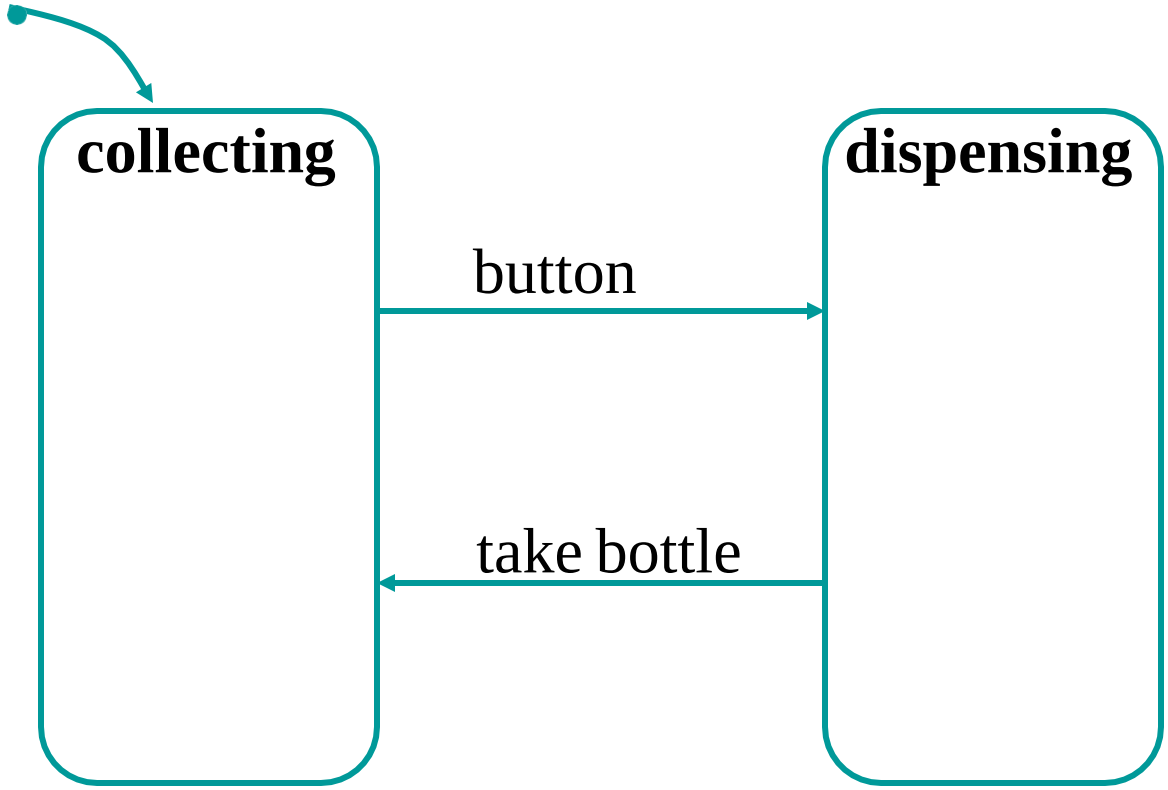
Vending Machine v4.0

- We can add even more features
- Automatic bottle filler
  - If a button is pressed, the machine will toggle between bottle filling and dispensing modes
  - When in bottle filling mode
    - If the bottle check finds a Coke was inserted, it will signal Coke_OK and return to ready
    - If the bottle check finds a Diet Coke was inserted, the coke machine will signal Diet_OK and return to ready
    - Otherwise, the bottle will be immediately dispensed

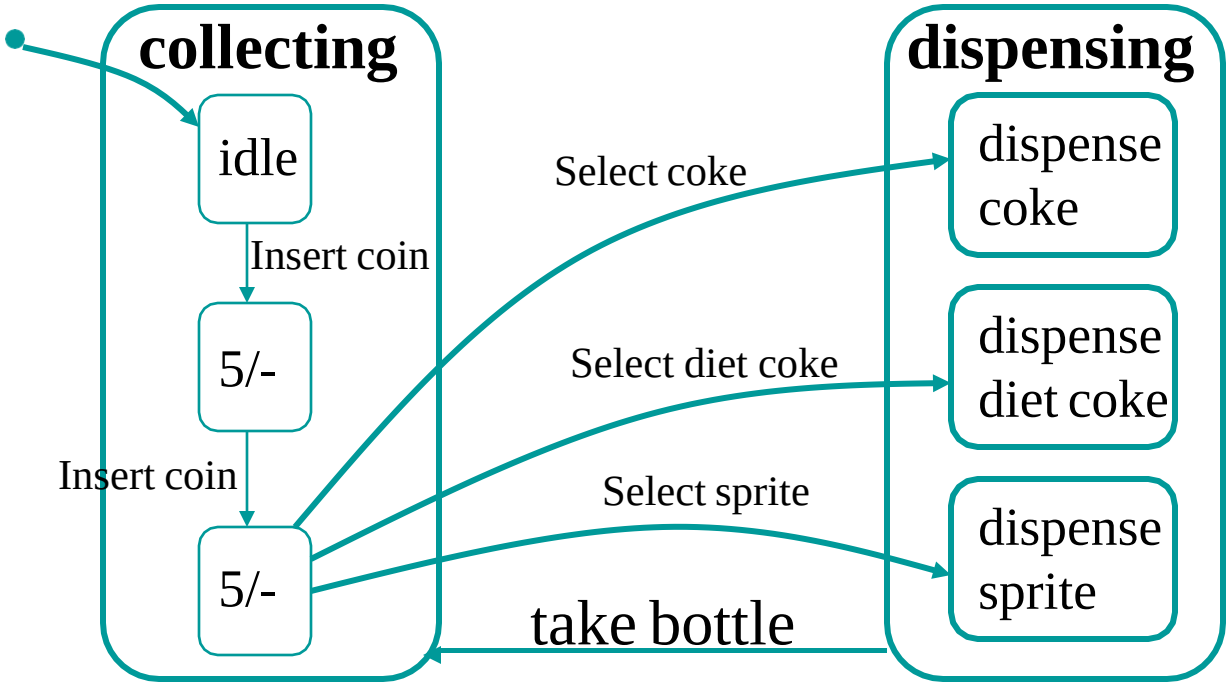
Vending Machine v4.0

- State machine changes
  - How many new states are required?
  - How many new transitions?

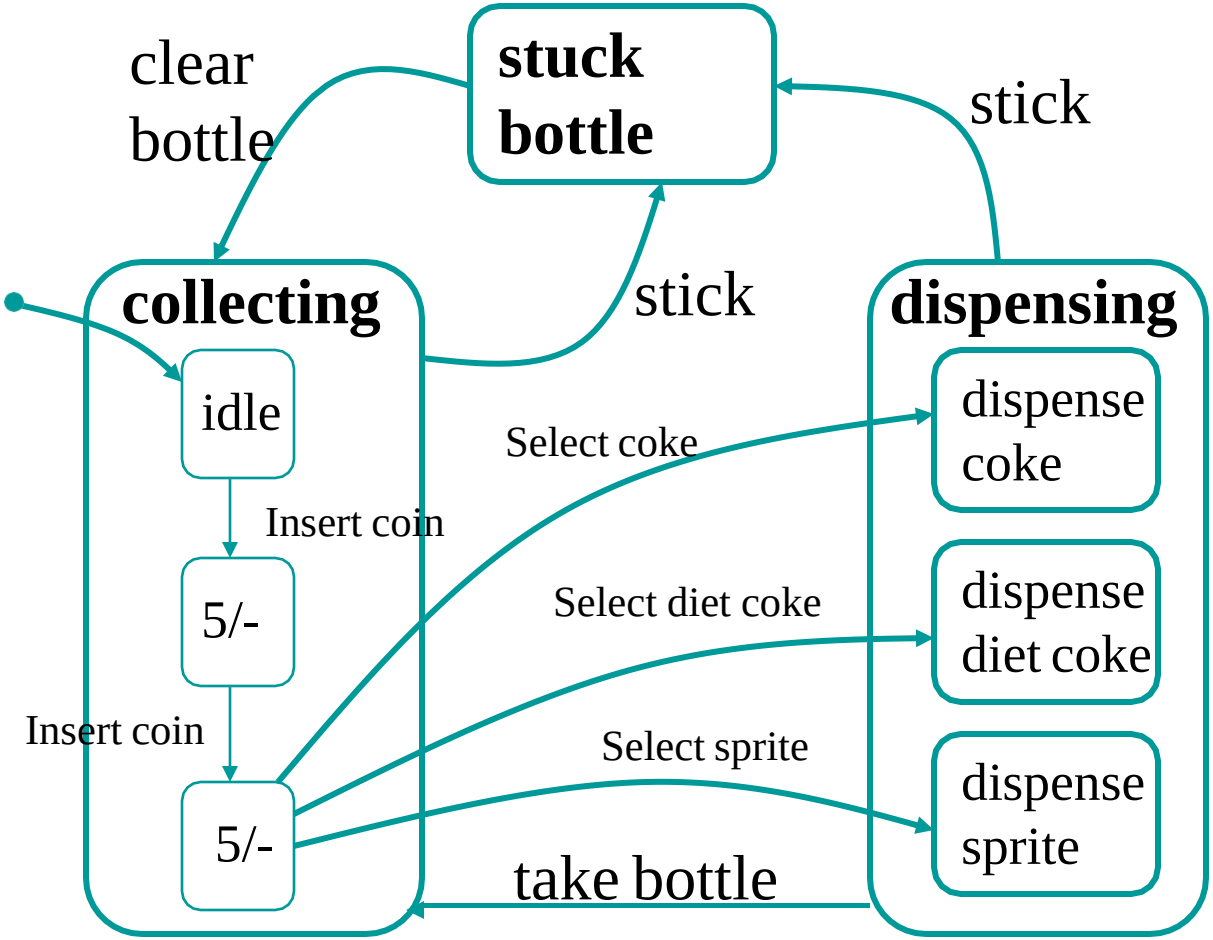
State-Chart Construction: Bottle Dispenser



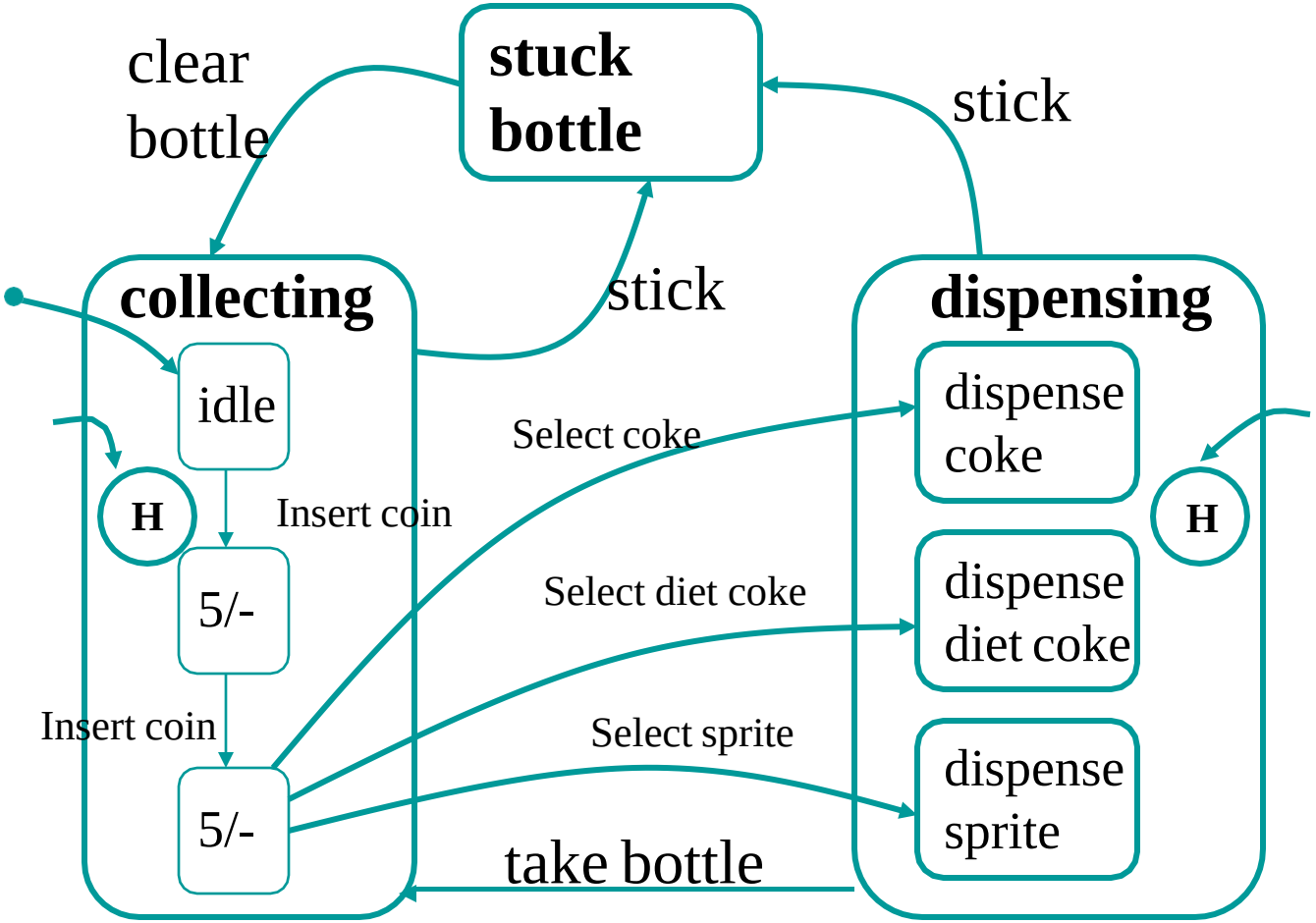
State-Chart Construction: Bottle Dispenser



State-Chart Construction: Bottle Dispenser



State-Chart Construction: Adding History



State-Chart Pros

- Large number of commercial simulation tools available (StateMate, StateFlow, BetterState, ...)
- Available “back-ends” translate State-Charts into C or VerilogHDL or VHDL, thus enabling software or hardware implementations