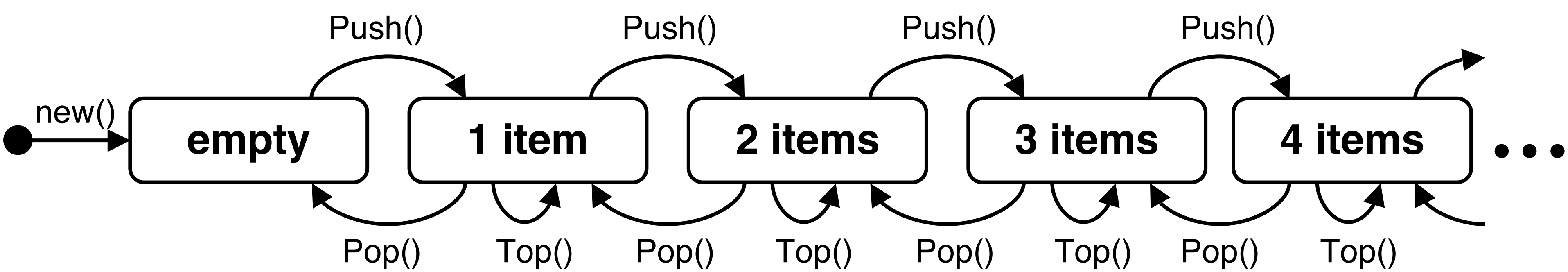
Lecture 13: Modelling "State"

- What is State?
  - statespace for an object
  - concrete vs. abstract states
- Finite State Machines
  - states and transitions
  - events and actions
- Modularized State machine models: Statecharts
  - superstates and substates
  - Guidelines for drawing statecharts

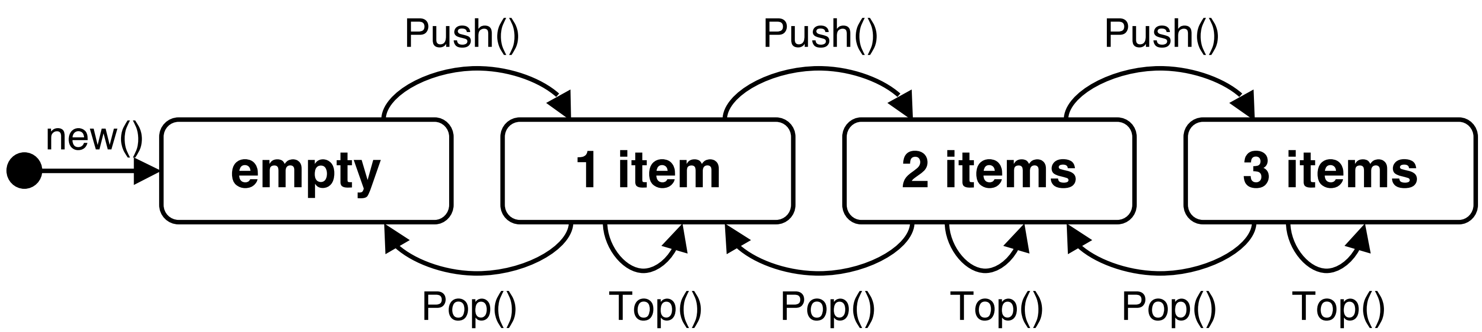
Getting objects to behave

- All objects have "state"
  - The object either exists or it doesn't
  - If it exists, then it has a value for each of its attributes
  - Each possible assignment of values to attributes is a "state"
    - (and non-existence of the object is also a state)
- E.g. For a stack object



What does the model mean?

- Finite State Machines
  - There are a finite number of states (all attributes have finite ranges)
    - E.g. imagine a stack with max length = 3

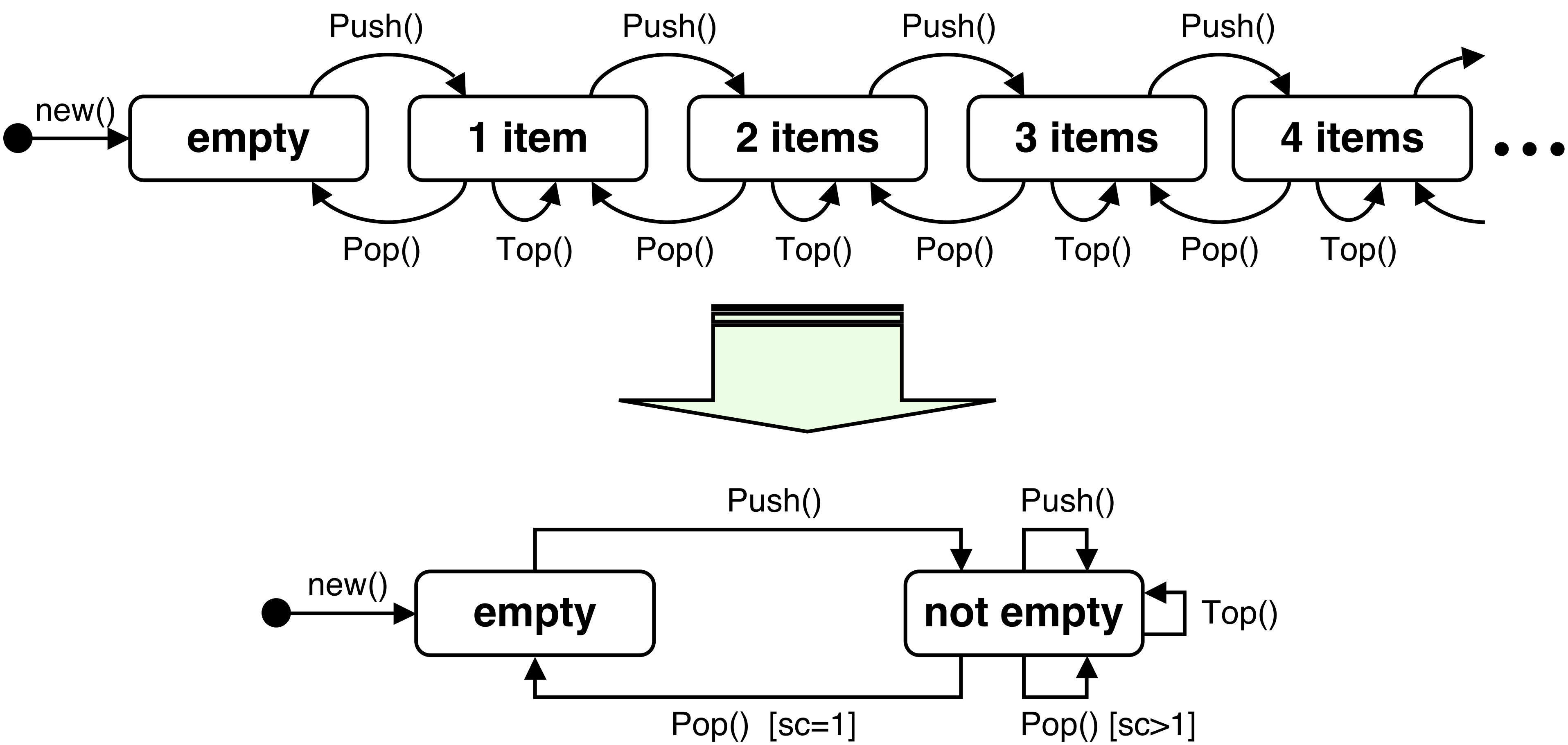


  - The model specifies a set of traces
    - E.g. new();Push();Push();Top();Pop();Push()...
    - E.g. new();Push();Pop();Push();Pop()...
    - There may be an infinite number of traces (and traces may be of infinite length)
  - The model excludes some behaviours
    - E.g. no trace can start with a Pop()
    - E.g. no trace may have more Pops than Pushes
    - E.g. no trace may have more than 3 Pushes without a Pop in between

Abstraction

- The state space of most objects is enormous
  - State space size is the product of the range of each attribute
    - E.g. object with five boolean attributes: 2^5+1 states
    - E.g. object with five integer attributes: $(maxint)^5+1$ states
    - E.g. object with five real-valued attributes: …?
  - If we ignore computer representation limits, the state space is infinite
- Only part of that state space is "interesting"
  - Some states are not reachable
  - Integer and real values usually only vary within some relevant range
  - Often, we're only interested in certain thresholds:
    - E.g. for Age, we may be interested in age<18; 18≤age≤65; and age>65
    - E.g. for Cost, we may want to distinguish cost≤budget, cost=0, cost>budget, and cost>(budget+10%)

Collapsing the state space



- The abstraction usually permits more traces
  - E.g. this model does not prevent traces with more pops than pushes
  - But it still says something useful

What are we modelling?

(D) Observed states of an application domain entity?

- E.g. a phone can be idle, ringing, connected, …

- Model shows the states an entity can be in, and how events can change its state
- This is an **indicative** model

(R) Required behaviour of an application domain entity?

- E.g. a telephone switch shall connect the phones only when the callee accepts the call

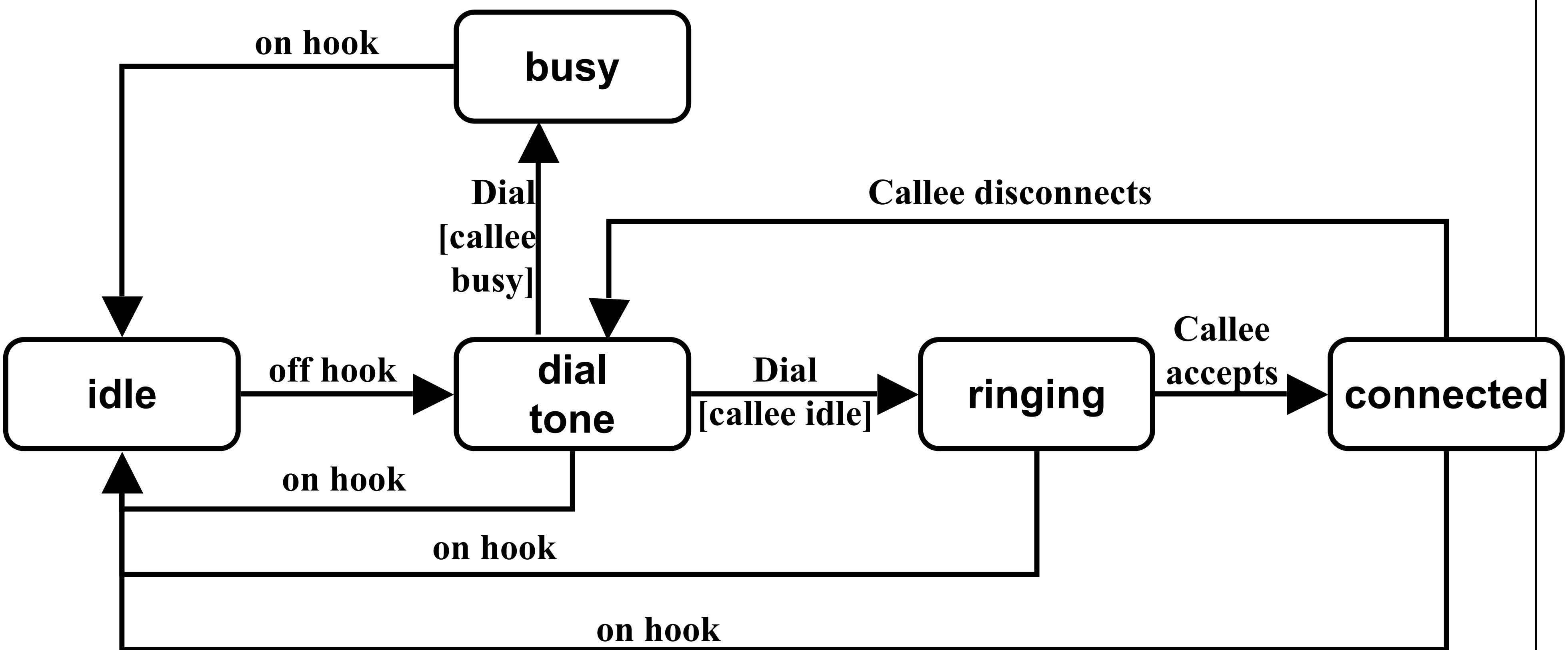
- Model distinguishes between traces that are desired and those that are not
- This is an **optative** model

(S) Specified behaviour of a machine domain entity?

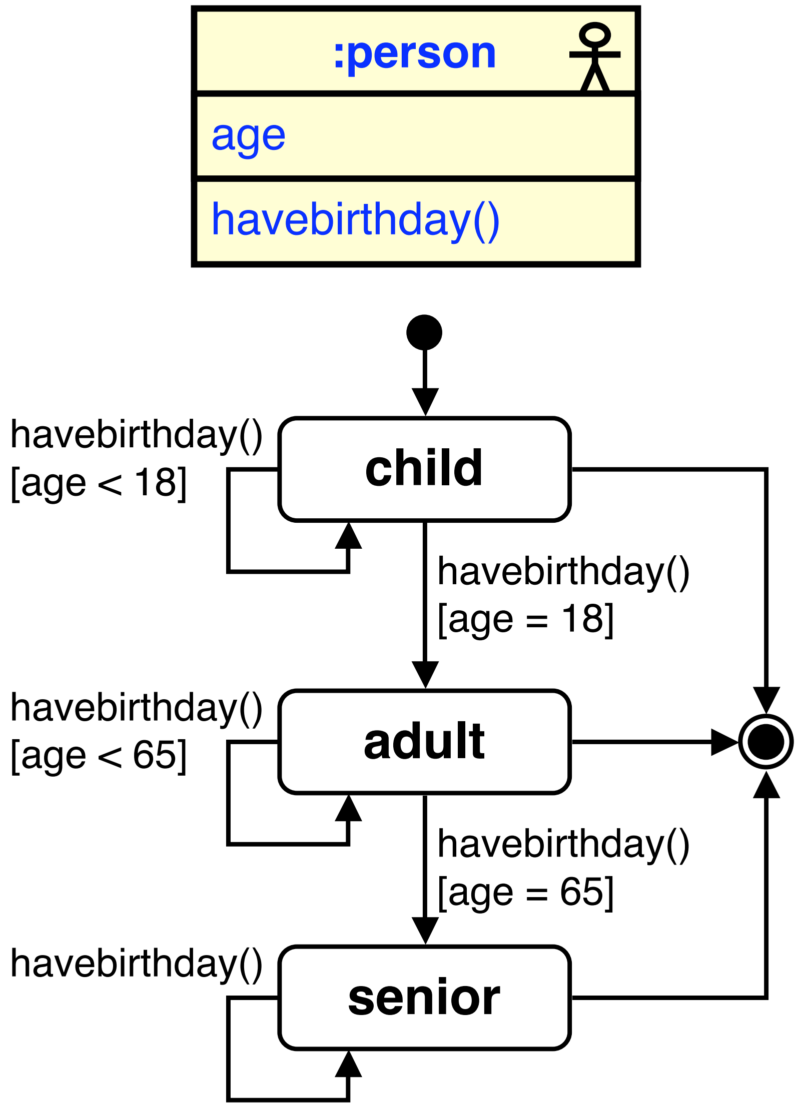
- E.g. when the user presses the 'connect' button the incoming call shall be connected

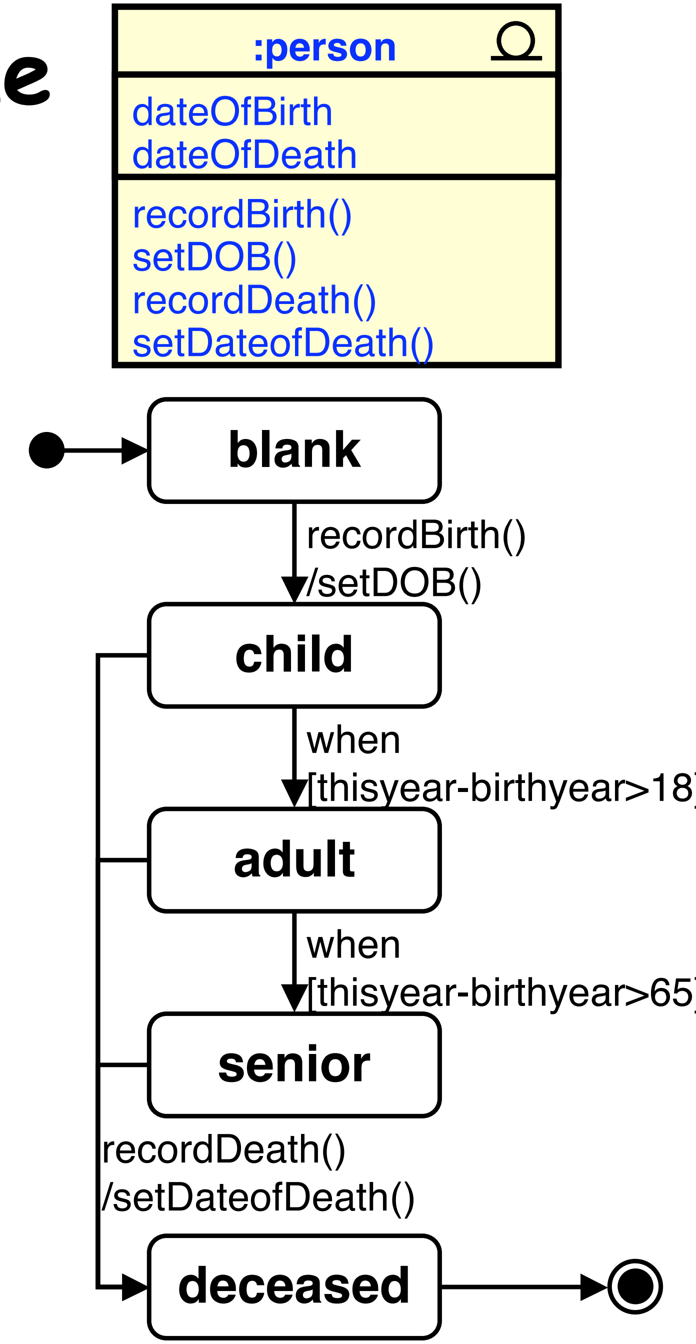
- Model specifies how the machine should respond to input events
- This is an **optative** model, in which all events are shared phenomena

Is this model indicative or optative?



the world vs. the machine





StateCharts

→ Notation:

States

- "interesting" configurations of the values of an object's attributes
- may include a specification of action to be taken on entry or exit
- States may be nested
- States may be "on" or "off" at any given moment

Transitions

- Are enabled when the state is "on"; disabled otherwise
- Every transition has an event that acts as a trigger
- A transition may also have a condition (or guard)
- A transitions may also cause some action to be taken
- When a transition is enabled, it can fire if the trigger event occurs and it guard is true
- Syntax: event [guard] / action

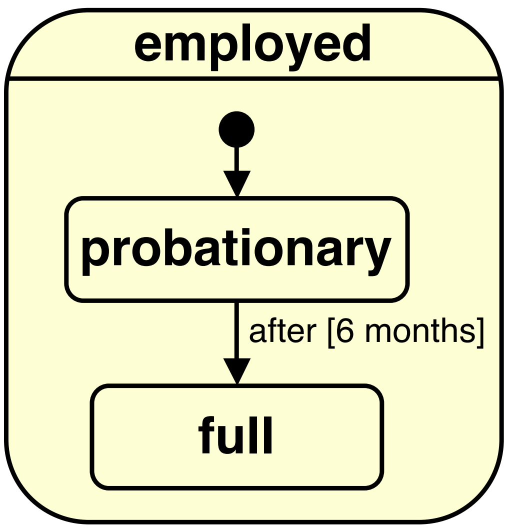
Events

- occurrence of stimuli that can trigger an object to change its state
- determine when transitions can fire

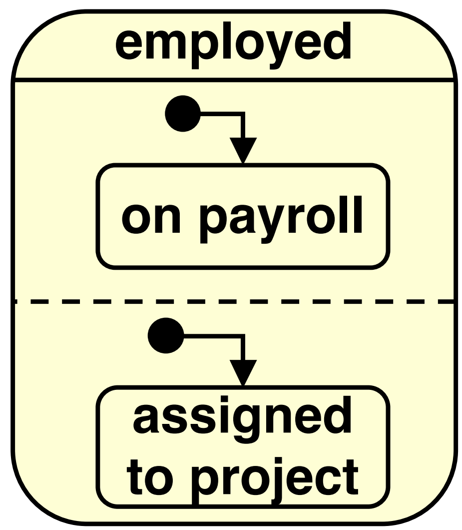
Superstates

→States can be nested, to make diagrams simpler
- A superstate contains one or more substates.
- Superstates make it possible to view a state diagram at different levels of abstraction.

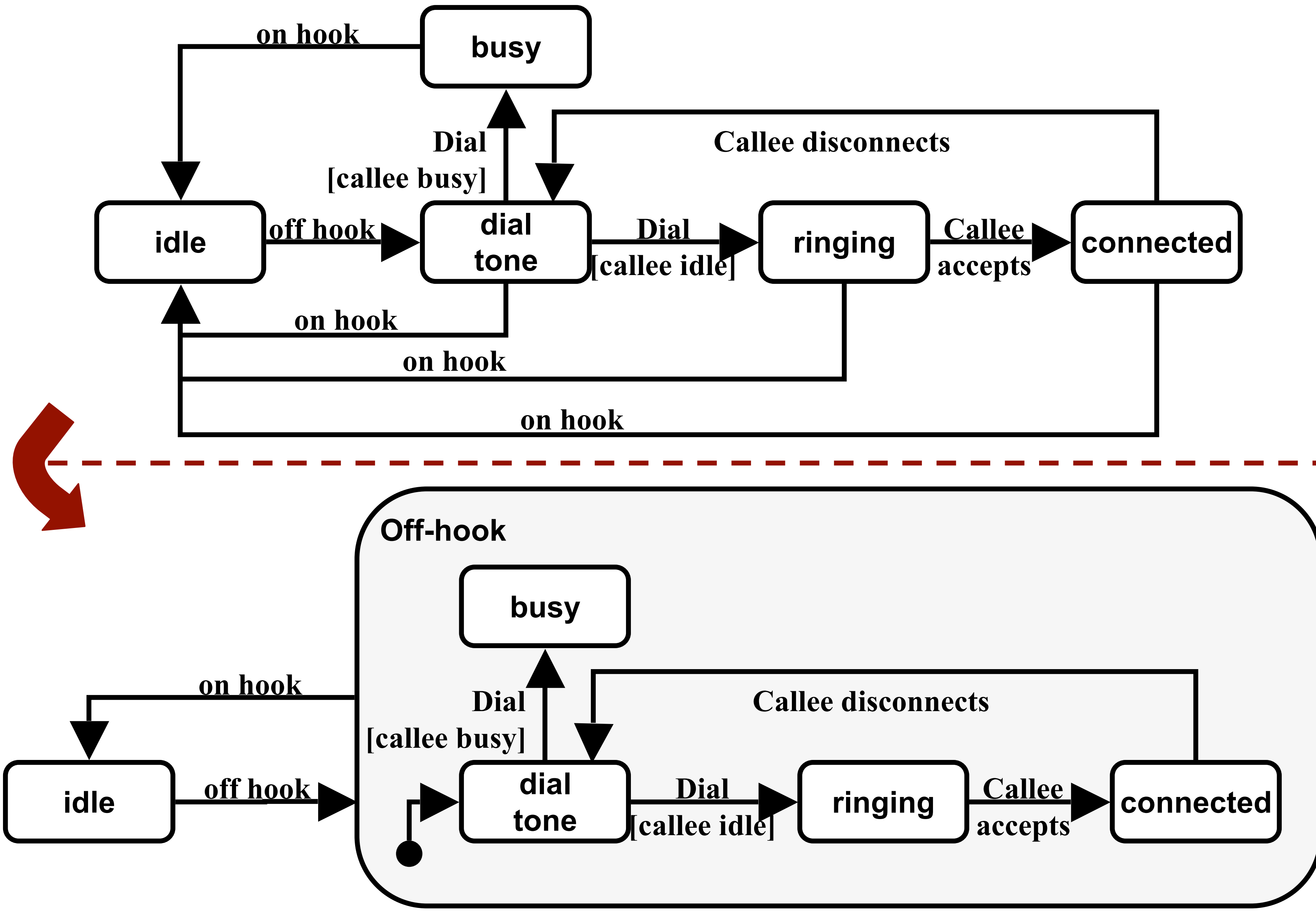
OR superstates
- when the superstate is "on", only one of its substates is "on"



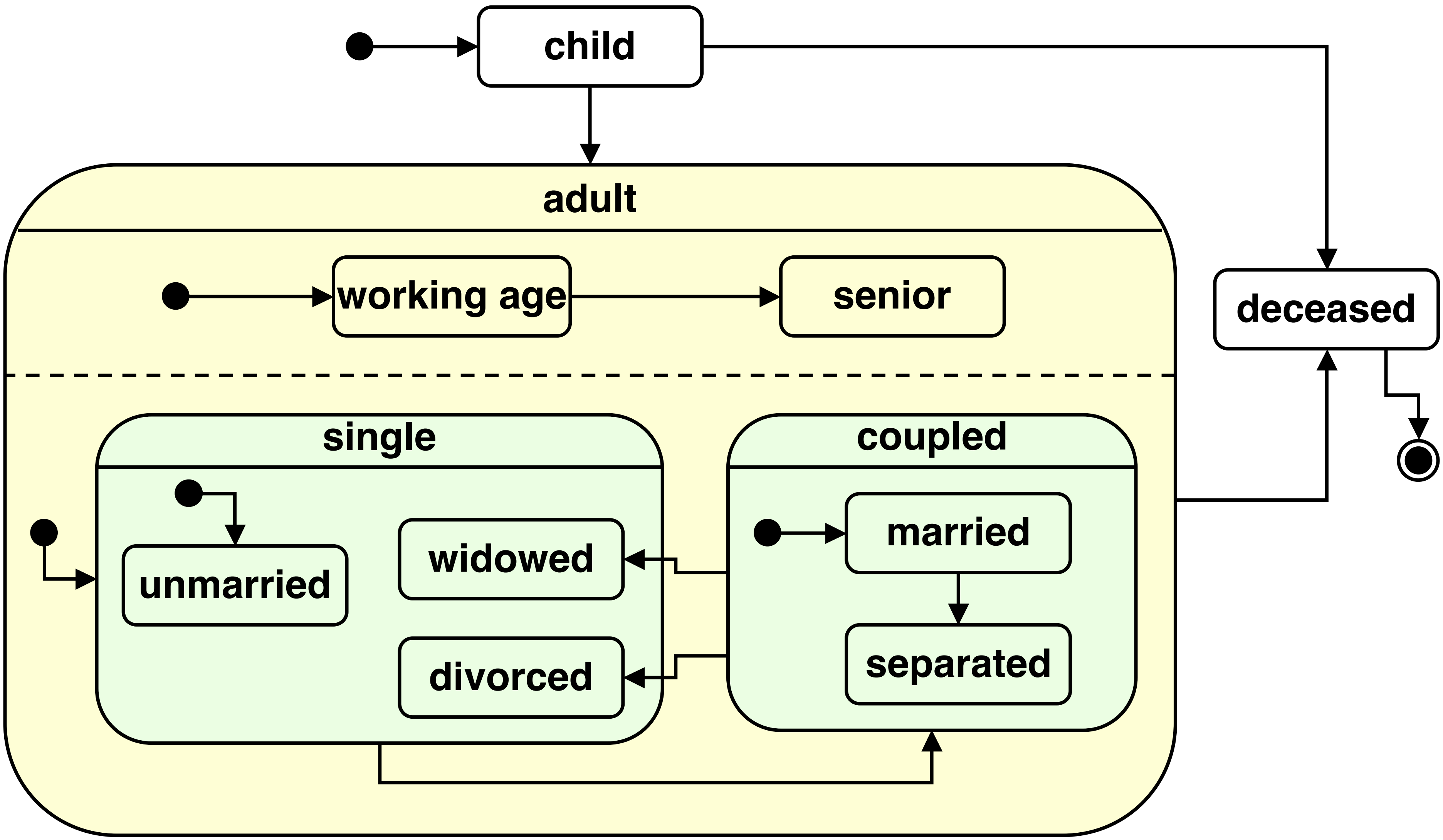
AND superstates (concurrent substates)
- When the superstate is "on", all of its states are also "on"
- Usually, the AND substates will be nested further as OR superstates



Superstates example



A more detailed example



States in UML

- A state represents a time period during which
  - A predicate is true
    - e.g. (budget - expenses) > 0,
  - An action is being performed, or an event is awaited:
    - e.g. checking inventory for order items
    - e.g. waiting for arrival of a missing order item
- States can have associated activities:
  - do/activity
    - carries out some activity for as long as the state is “on”
  - entry/action and exit/action
    - carry out the action whenever the state is entered (exited)
  - include/stateDiagramName
    - “calls” another state diagram, allowing state diagrams to be nested

Events in UML

- Events are happenings the system needs to know about
  - Must be relevant to the system (or object) being modelled
  - Must be modellable as an instantaneous occurance (from the system's point of view)
    - E.g. completing an assignment, failing an exam, a system crash
  - Are implemented by message passing in an OO Design
- In UML, there are four types of events:
  - *Change events* occur when a condition becomes true
    - denoted by the keyword 'when'
    - e.g. when[balance < 0]
  - *Call events* occur when an object receives a call for one of its operations to be perfomed
  - *Signal events* occur when an object receives an explicit (real-time) signal
  - *Elapsed-time events* mark the passage of a designated period of time
    - e.g. after[10 seconds]

Checking your Statecharts

Consistency Checks

- All events in a statechart should appear as:
  - operations of an appropriate class in the class diagram
- All actions in a statechart should appear as:
  - operations of an appropriate class in the class diagram

Style Guidelines

- Give each state a unique, meaningful name
- Only use superstates when the state behaviour is genuinely complex
- Do not show too much detail on a single statechart
- Use guard conditions carefully to ensure statechart is unambiguous
  - Statecharts should be deterministic (unless there is a good reason)

You probably shouldn't be using statecharts if:

- you find that most transitions are fired "when the state completes"
- many of the trigger events are sent from the object to itself
- your states do not correspond to the attribute assignments of the class