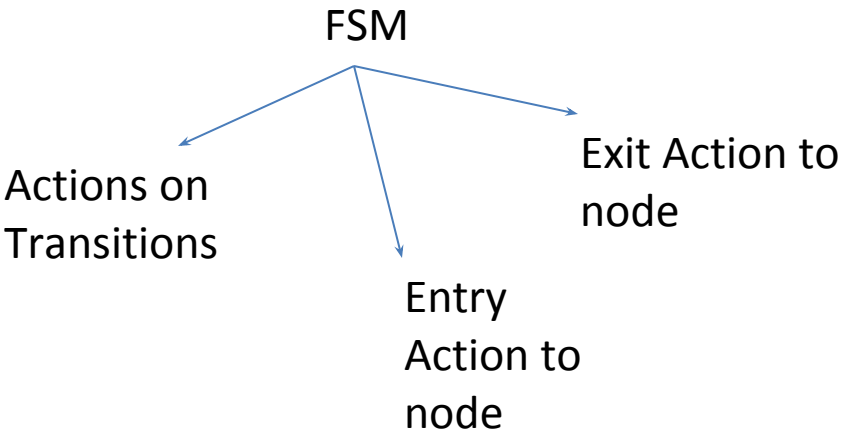
Finite State Machine

- A **finite-state machine** (**FSM**) or **finite-state automaton** (**FSA**, plural: *automata*), **finite automaton**, or simply a **state machine**, is a mathematical model of computation.

- It is an abstract machine that can be in exactly one of a finite number of states at any given time.
- The FSM can change from one state to another in response to some external inputs; the change from one state to another is called a *transition*. An FSM is defined by a list of its states, its initial state, and the conditions for each transition.
- Finite state machines are of two types - deterministic finite state machine and non deterministic finite state machine
- Examples: Vending machine, Elevator, Traffic lights etc

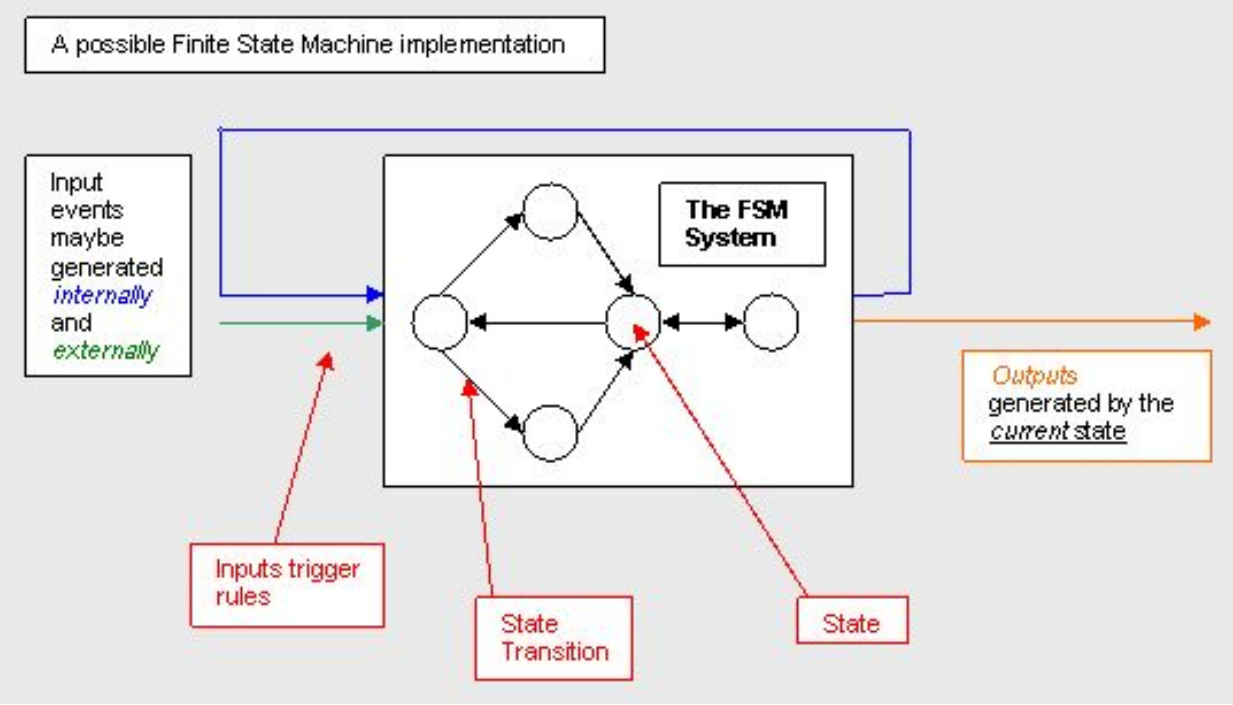
- FSM is a graph that describes how software variables are modified during execution.

- FSM helps to analyze early detection of errors through analysis of the model such as UML, State tables, Boolean logic etc. Since, these are modeled even before the code is generated.
- FSM is used for testing since 30 years
- It is best suited for control intensive applications such as elevators and not best suited for data control applications like web apps



- Nodes – represent states which in turn indicates the set of values for the key variables
- Edges – represent transitions which will have guards (conditions) and or actions.
- Preconditions (guards) – Conditions to be present for transition to occur
- Triggering events – changes to variables that cause the transitions
- Testers thus have to create test cases where every state has to be visited and hence every path will be checked for i.e for every transition there should be test cases generated
- Thus, testers have to know about FSM since they should be able to draw FSM models with complete state coverage. Also they have to define the data associated with each state

- Initial state - provides a starting point.
- Current state - remembers the product of the last state transition.



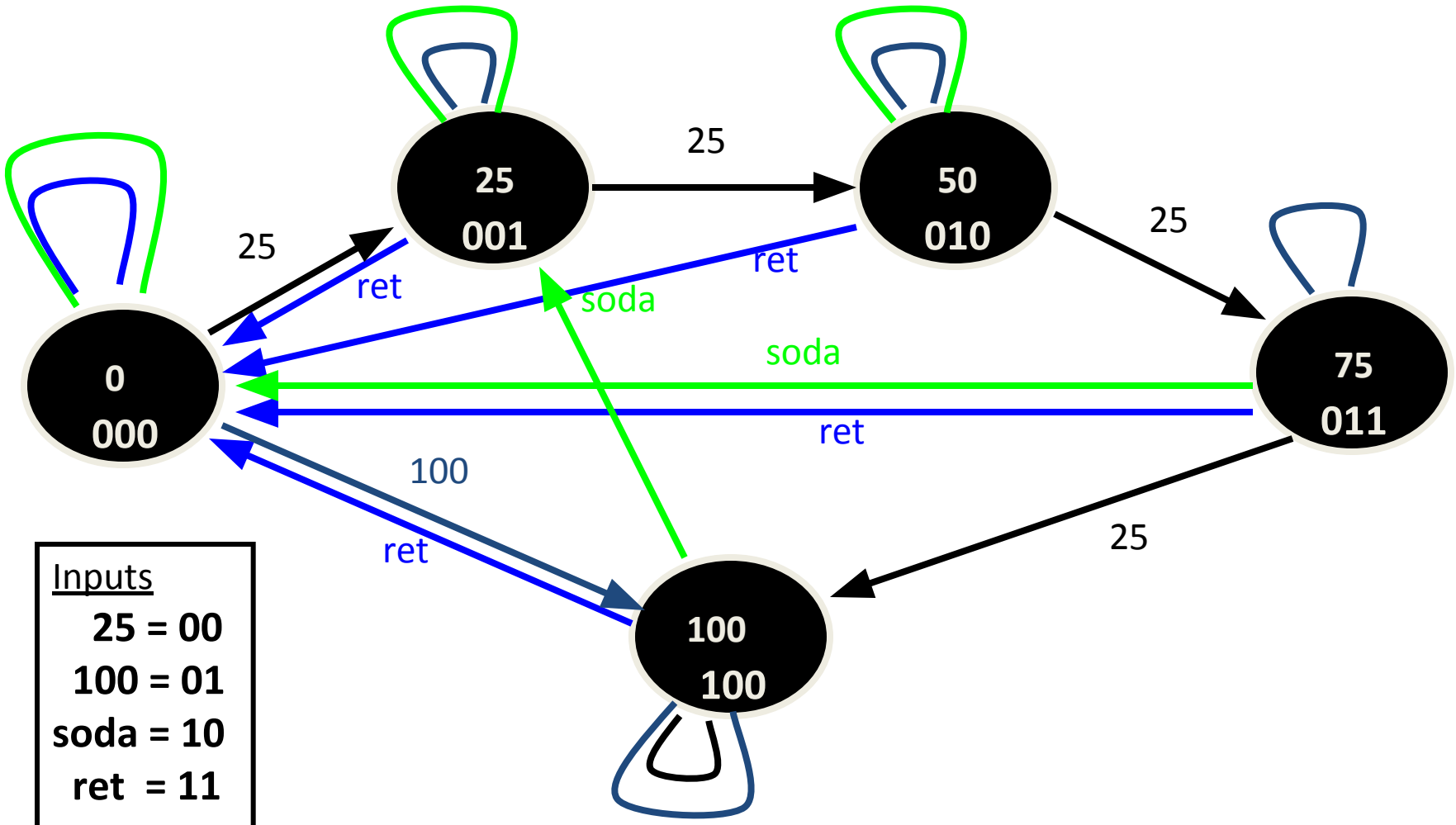
Example: Vending Machine

- Takes only quarters and dollar bills
- Won't hold more than $1.00
- Sodas cost $.75
- Possible actions (inputs)
  - deposit $.25 (25)
  - deposit $1.00 ($)
  - push button to get soda (soda)
  - push button to get money returned (ret)

Classic Example: Vending Machine

- State: description of the internal settings of the machine, e.g. how much money has been deposited and not spent
- Finite states: 0, 25, 50, 75, 100,
- Rules: determine how inputs can change state

Example: Vending Machine



Petri Net Overview

- Petri nets were invented by Carl Petri in 1966 to explore cause and effect relationships
- Expanded to include deterministic time
- Then stochastic time
- Then logic

Definition

- A Petri Nets (PN) comprises places, transitions, and arcs
  - Places are system states
  - Transitions describe events that may modify the system state
  - Arcs specify the relationship between places
- Tokens reside in places and are used to specify the state of a PN

Definition of Petri Net

- **C = (P, T, I, O)**
  - Places
    $P = \{ p_1, p_2, p_3, \ldots, p_n\}$
    - Represents an entity or a pathway component
  - Transitions
    $T = \{ t_1, t_2, t_3, \ldots, t_n\}$
    - Represent a process between places and / or used to model dependencies between places
  - Input
    $I : T \rightarrow P^r$ (r = number of places)
  - Output
    $O : T \rightarrow P^q$ (q = number of places)
    - Tokens
    - Tokens reside in places, and are used to specify the state of a Petri Net
- marking μ : assignment of tokens to the places of Petri net $\mu = \mu_1, \mu_2, \mu_3, \ldots \mu_n$

Applications of Petri Net

- Petri net is primarily used for studying the dynamic concurrent behavior of network-based systems where there is a discrete flow.

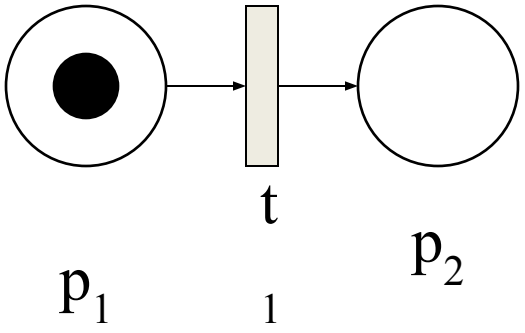
- Petri Nets are applied in practice by industry, academia, and other places.

Basics of Petri Nets

- Petri net consist two types of nodes: *places* and *transitions*. An arc exists only from a place to a transition or from a transition to a place.

- A place may have zero or more *tokens*.

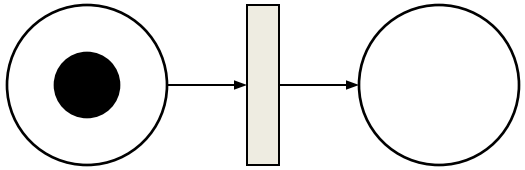
- Graphically, places, transitions, arcs, and tokens are represented respectively by: circles, bars, arrows, and dots.



Basics of Petri Nets –continued

- Below is an example Petri net with two places and one transaction.

- Transition node is ready to ***fire*** if and only if there is at least one token at each of its input places

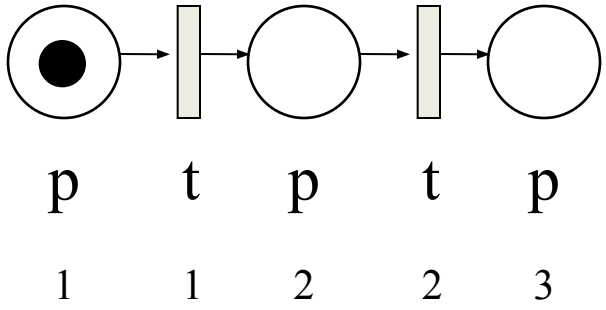


state transition of form (1, 0) □ (0, 1)

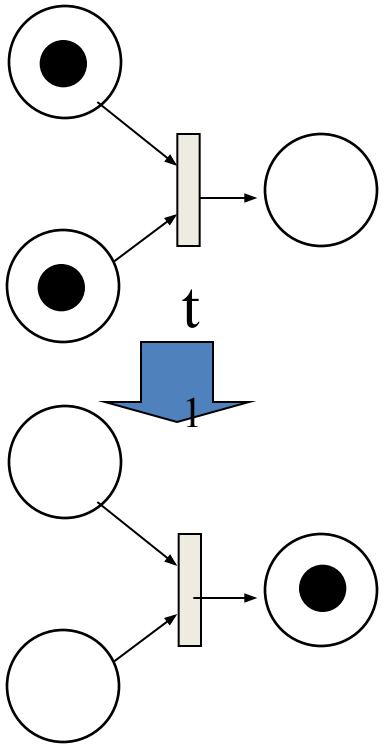
p_1 : input place p_2: output place

Properties of Petri Nets

- Sequential Execution:
  Transition t_2 can fire only after the firing of t_1. This impose the precedence of constraints "t_2 after t_1."



- Synchronization:
  Transition t_1 will be enabled only when a token there are at least one token at each of its input places.

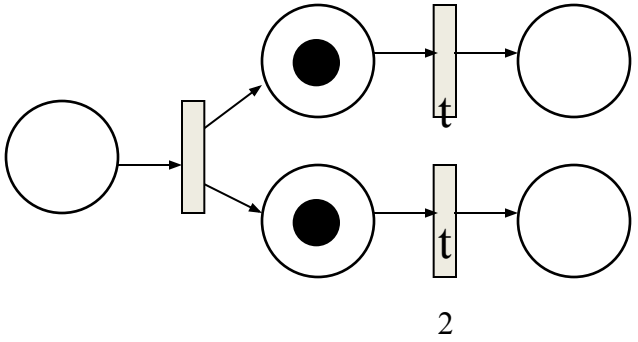


- Merging:
  Happens when tokens from several places arrive for service at the same transition.

Properties of Petri Nets –continued

- Concurrency:

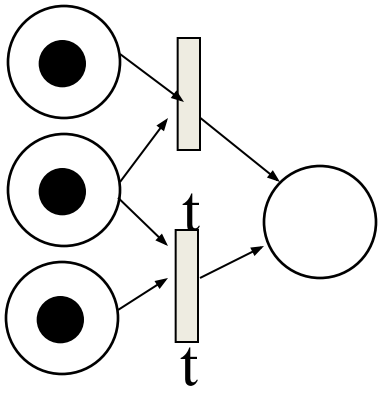
  t_1 and t_2 are concurrent.
  - with this property, Petri net is able to model systems of distributed control with multiple processes executing concurrently in time.

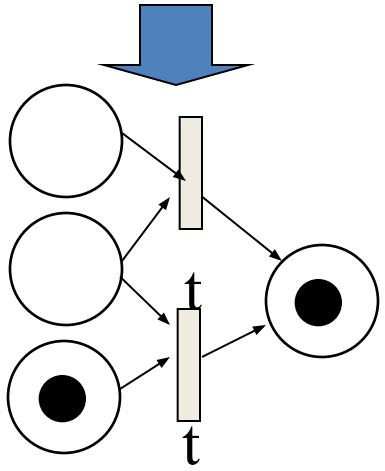


Properties of Petri Nets –continued

- Conflict

  t_1 and t_2 are both ready to fire but the firing of any leads to the disabling of the other transitions.



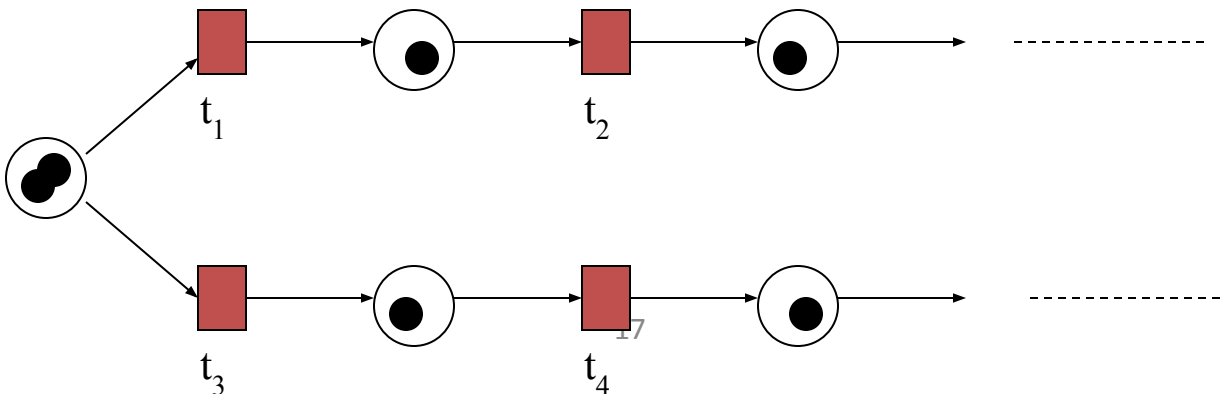


2

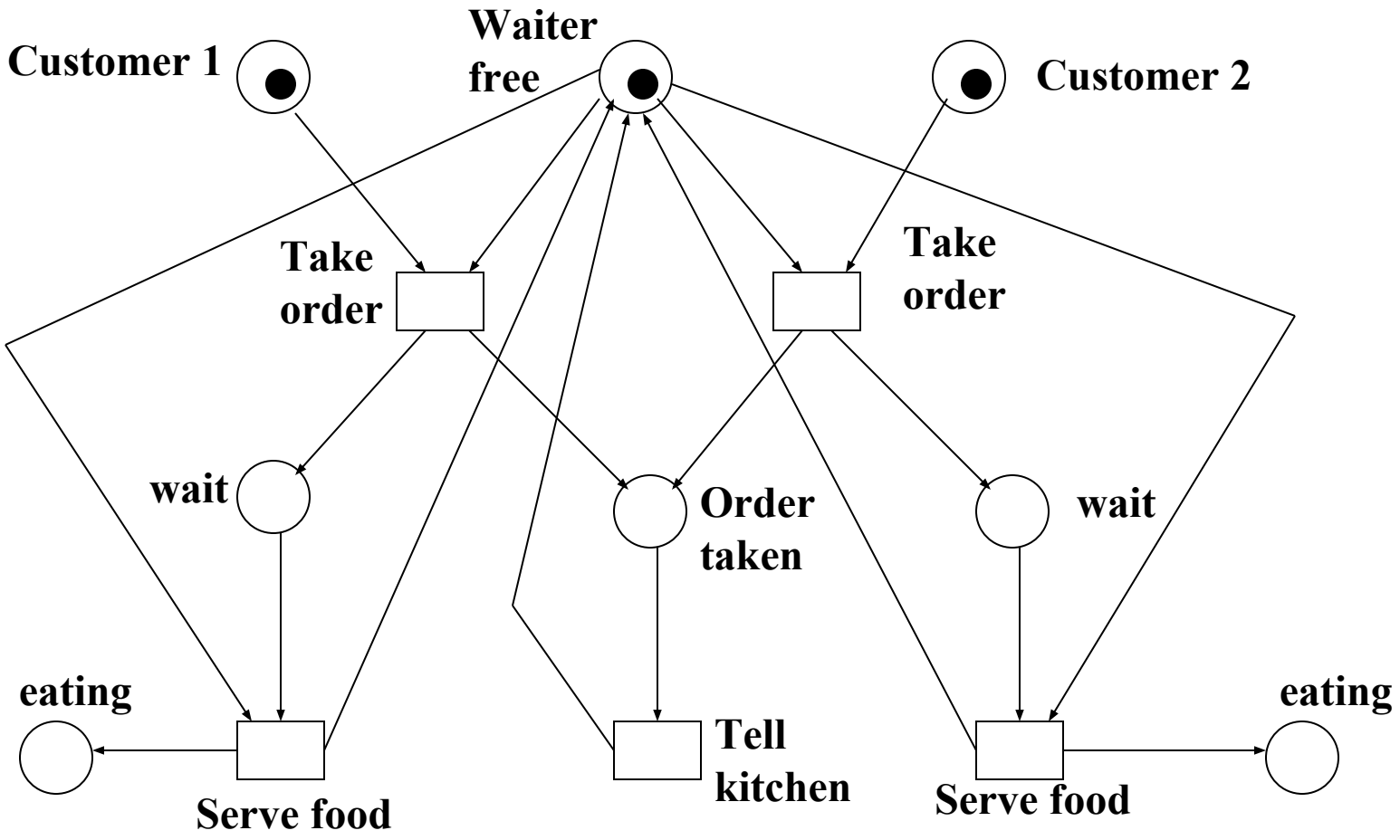
Properties of Petri Nets –continued

- Conflict - continued
  - the resulting conflict may be resolved in a purely non-deterministic way or in a probabilistic way, by assigning appropriate probabilities to the conflicting transitions.

there is a choice of either t_1 and t_2, or t_3 and t_4



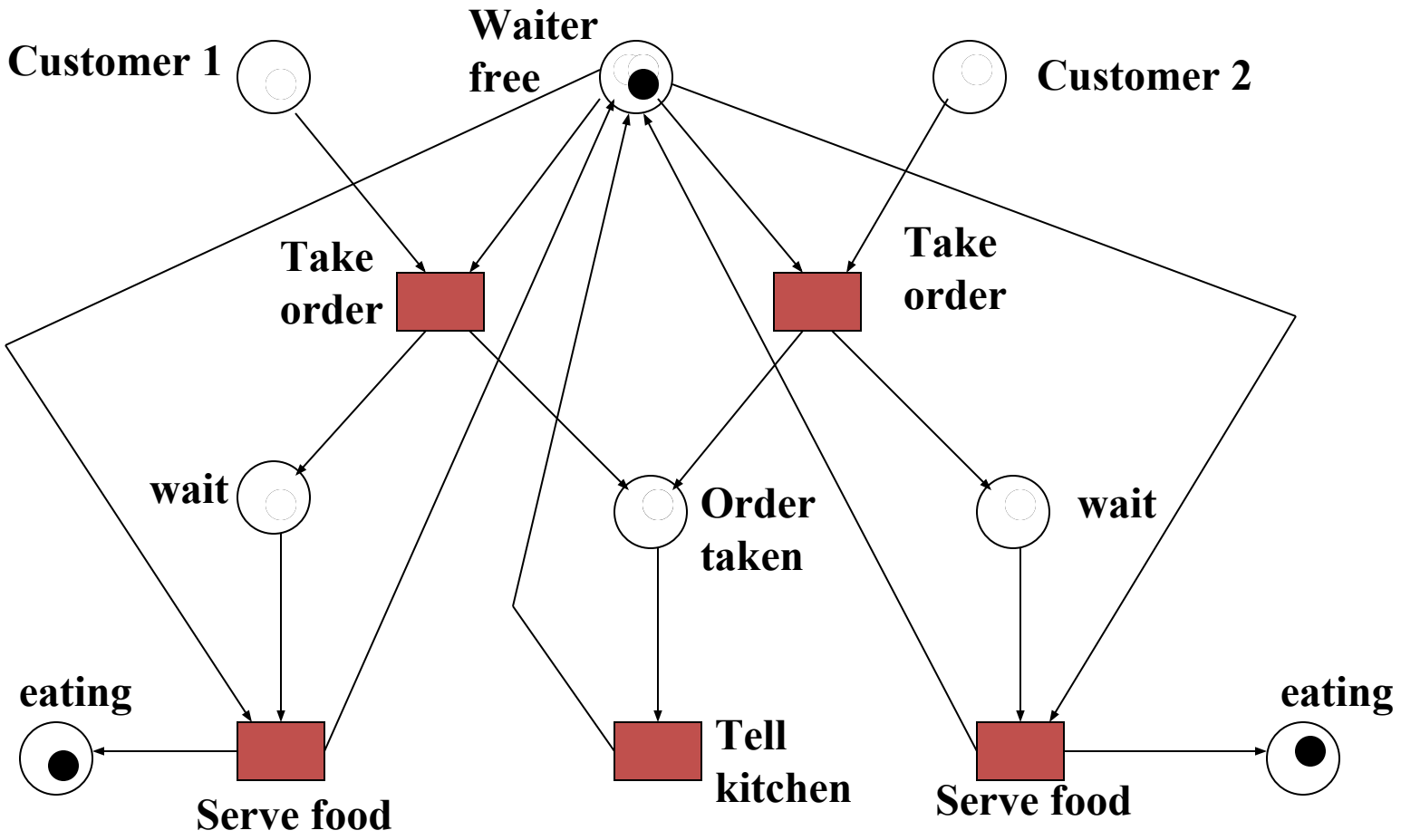
Example: In a Restaurant (A Petri Net)



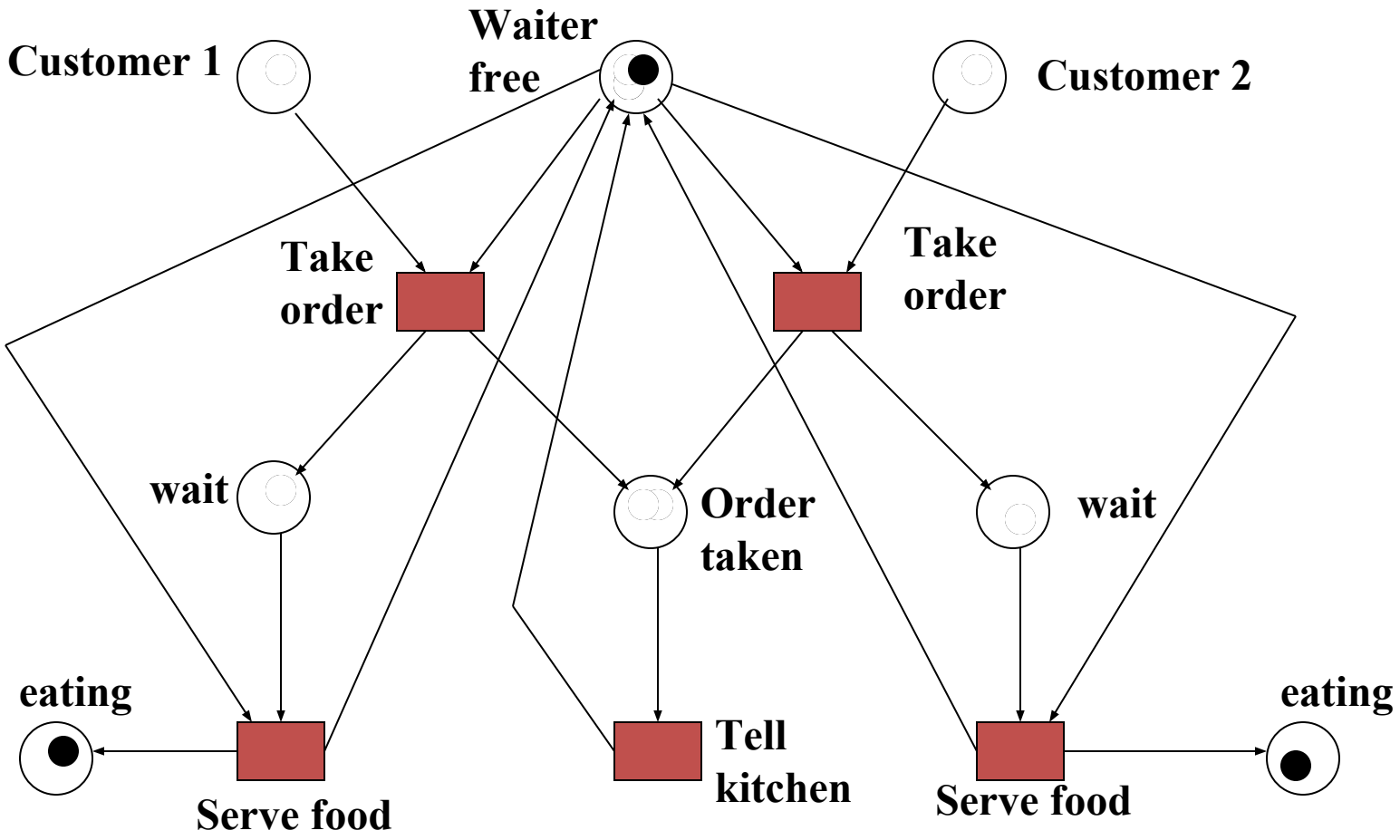
Example: In a Restaurant (Two Scenarios)

- Scenario 1:
  - Waiter takes order from customer 1; serves customer 1; takes order from customer 2; serves customer 2.
- Scenario 2:
  - Waiter takes order from customer 1; takes order from customer 2; serves customer 2; serves customer 1.

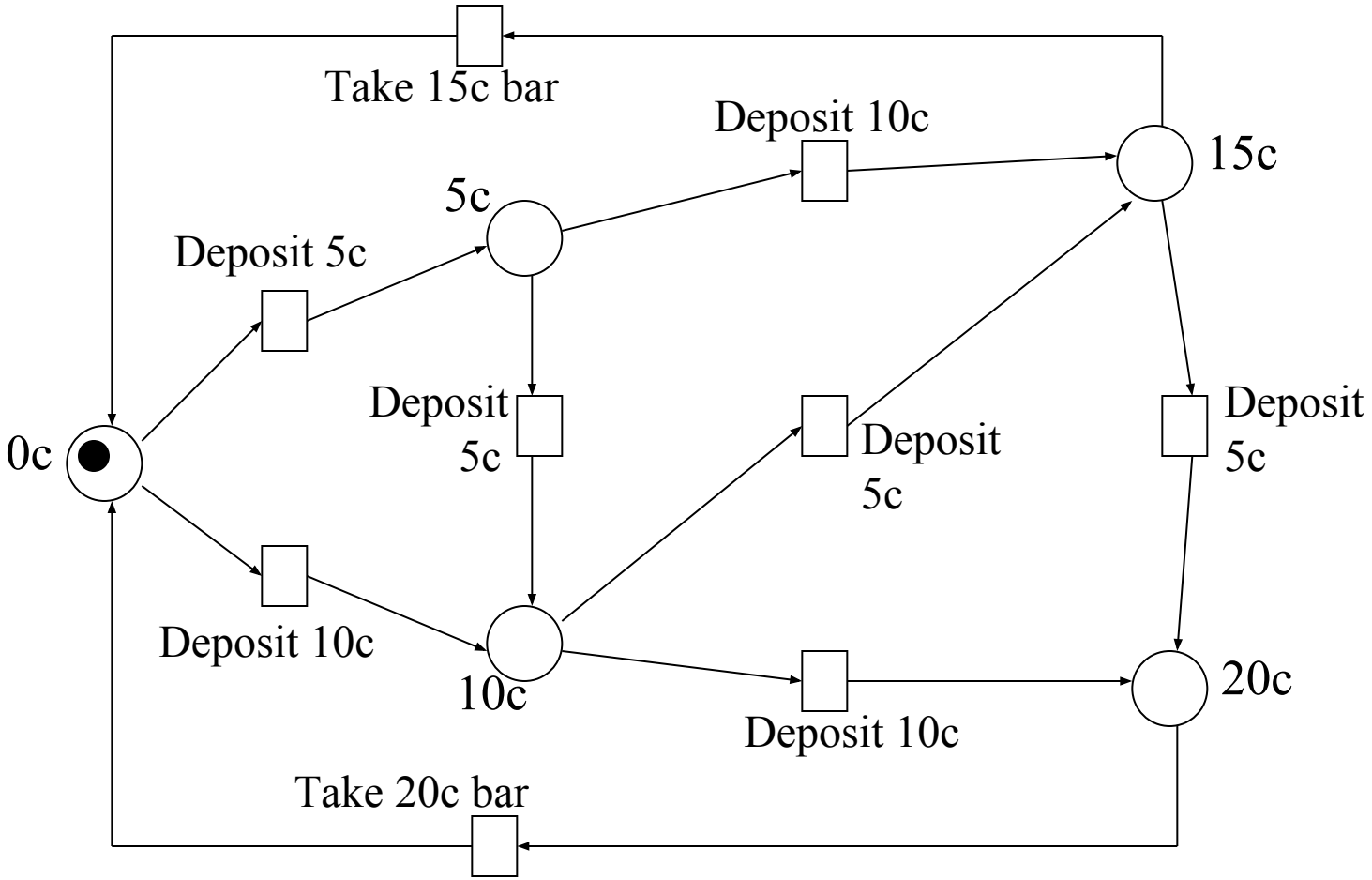
Example: In a Restaurant (Scenario 1)



Example: In a Restaurant (Scenario 2)



Example: Vending Machine (A Petri net)



Example: Vending Machine (3 Scenarios)

- Scenario 1:
  - Deposit 5c, deposit 5c, deposit 5c, deposit 5c, take 20c snack bar.
- Scenario 2:
  - Deposit 10c, deposit 5c, take 15c snack bar.
- Scenario 3:
  - Deposit 5c, deposit 10c, deposit 5c, take 20c snack bar.

Example: Vending Machine (Token Games)