CS 4072 - Topics in CS
Process Mining

Lecture # 06

March 01, 2022

Spring 2022

FAST - NUCES, CFD Campus

Dr. Rabia Maqsood

rabia.maqsood@nu.edu.pk

Today’s Topics

- Petri-net
  - Properties: deadlock, liveness
  - Modeling by petri nets

Properties of Petri Nets

- Petri nets properties are basis for their formal analysis

- The following properties are commonly used:
  - Reachability
  - Liveness
  - Boundedness

Reachability

- Reachability set $R(M_0)$ is a set of all possible markings reachable from the initial marking M_0
- If a marking M is in the reachability set $R(M_0)$, it means that there exists a firing sequence transforming M_0 into M
- This is a useful property for analysis of systems. We can ask whether:
  - A desired state can be reached at all?
  - Is it possible for a system to arrive at an undesired (erroneous) state?

Reachability and Reachability Graph

- Reachability can be analysed by building a reachability graph
- The graph is constructed by finding all possible transitions from the initial marking – this gives a set of markings reachable from the initial one, then all possible transitions from the previously discovered markings, and so on
- May lead to extremely large graphs

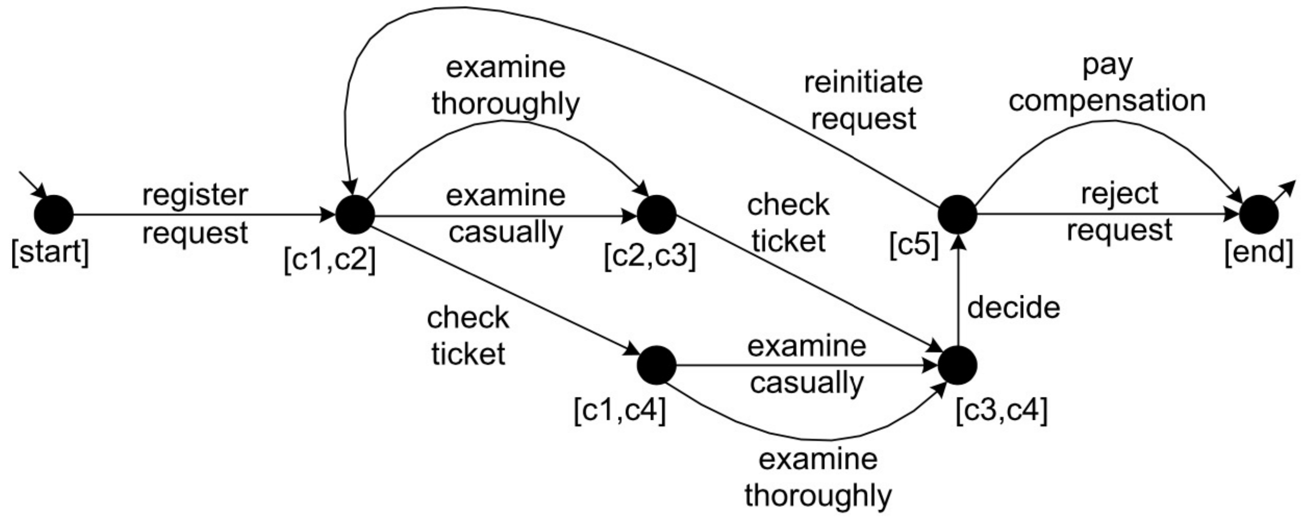
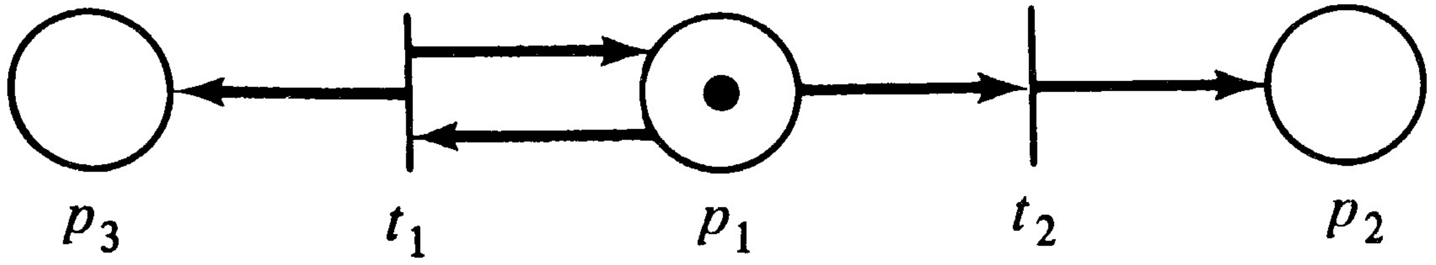


Fig. 3.3 The reachability graph of the marked Petri net shown in Fig. 3.2

Practice Work

You will see deadlocks in the reachability graph.

- Draw the reachability graph of this Petri net.



Liveness

- A marking M_i of network is **live** if, for any transition t and for every reachable marking M_j there exists a firing sequence from M_i that includes t

- In other words, every transition of the net can fire an infinite number of times

- A Petri Net N is structurally live, if any initial marking of N is live

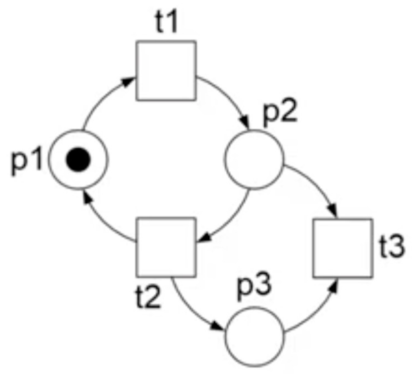
- Liveness may be used to model the occurrences of deadlocks

Deadlock

A marking M of a petri net $N = \langle P,T,F,W,M_0 \rangle$ is called a deadlock if no transition $t \in T$ is enabled in M. A net N is deadlock-free if no reachable marking is a deadlock.

- Using the reachability graph: Check whether there is a vertex without an outgoing edge.
- Application: Deadlocks tend to indicate errors (classical example: philosophers may starve).

Deadlock



$$\begin{tikzcd}
{[p1]} \arrow[r, "t1"] & {[p2]} \arrow[r, "t2"] & {[p1,p3]} \arrow[r, "t1"] & {[p2,p3]} \arrow[r, "t2"] \arrow[d, "t3"'] & {[p1,p3^2]} \arrow[r, "t1"] & {[p2,p3^2]} \arrow[r, "t2"] \arrow[d, "t3"'] & \dots \\
 & & & {[\,]} & & {[p3]} &
\end{tikzcd}$$

Boundedness

- A marking Mi is **bounded** if there exists a positive integer k such that for every reachable marking M_j - element of the reachability set $R(M_i)$ - the number of tokens in each place is bounded by k.
- A Petri net is structurally (inherently) bounded if all of its initial markings are bounded
- In other words, no reachable state can at any place contain more than k tokens
- If k equals one, the marking is said to be safe
- This property is useful for modelling limited (bounded) resources

Modeling with Petri Nets

Applications

Modeling with Petri Nets

- Petri nets were designed for and are used mainly for ***modeling***.
- The systems may be of many different kinds: computer hardware, computer software, physical systems, social systems, business processes, and so on.
- Petri nets are used to model the occurrence of various events and activities in a system.
- Petri nets may model the flow of information or other resources within a system.

Modeling with Petri Nets

- Real life systems/processes contain events/actions and conditions.
- In petri net models, events are represented by transitions and conditions are represented by places.
- Events are actions so they may *occur*.
- For an event to occur, it may be necessary for certain condition(s) to hold – that is, *preconditions* of an event.
- Occurrence of the event may cause the preconditions to cease to hold and may cause other conditions, *postconditions*, to become true.

Order-Machine Example

- The machine shop waits until an order appears and then machines the ordered part and sends it out for delivery.

The conditions for the system are:

1. The machine shop is waiting.
2. An order has arrived and is waiting.
3. The machine shop is working on the order.
4. The order is complete.

The events would be:

1. An order arrives.
2. The machine shop starts on the order.
3. The machine shop finishes the order.
4. The order is sent for delivery.

event	*preconditions*	*postconditions*
1	none	*b*
2	*a*, *b*	*c*
3	*c*	*d*, *a*
4	*d*	none

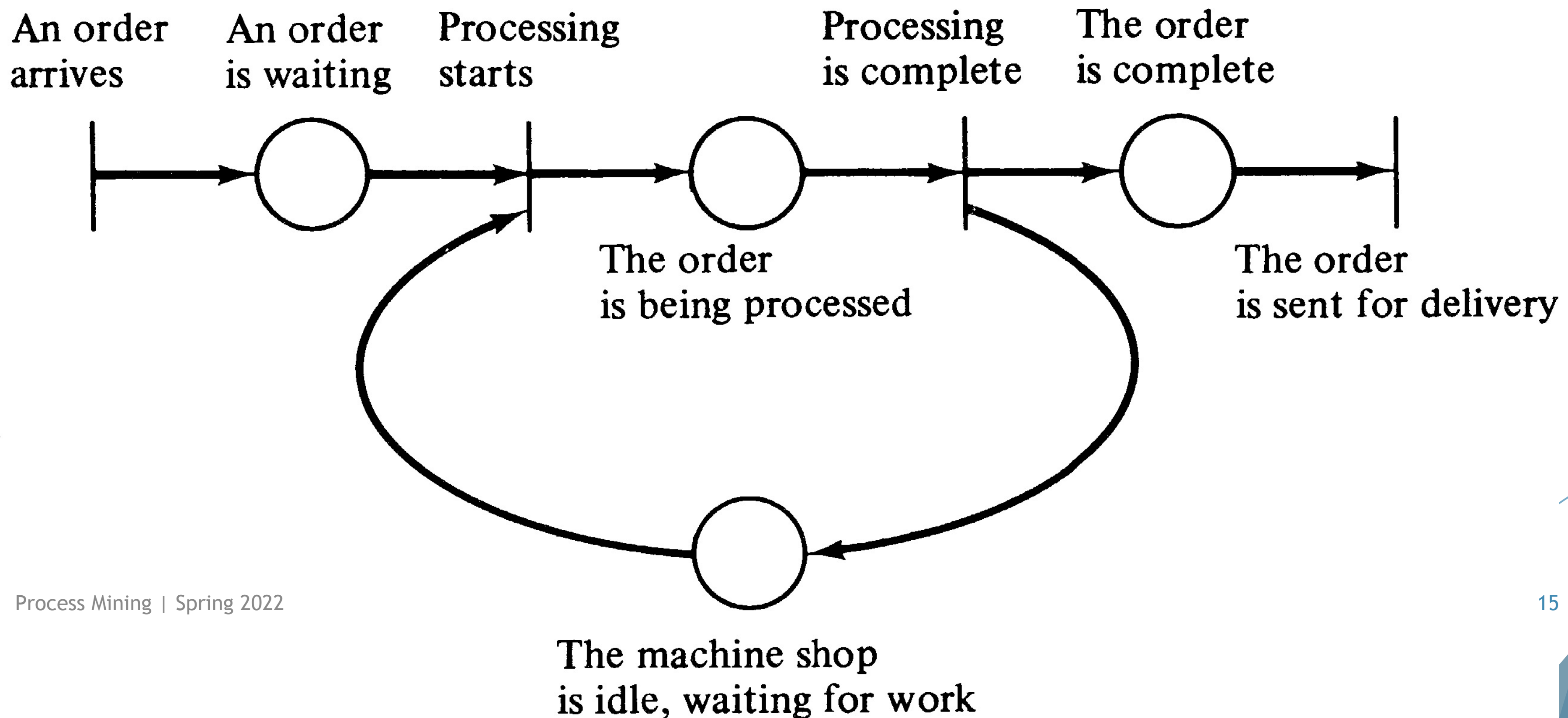
Order-Machine Example

The conditions for the system are:

1. The machine shop is waiting.
2. An order has arrived and is waiting.
3. The machine shop is working on the order.
4. The order is complete.

The events would be:

1. An order arrives.
2. The machine shop starts on the order.
3. The machine shop finishes the order.
4. The order is sent for delivery.


An order arrives
An order is waiting
Processing starts
Processing is complete
The order is complete
The order is being processed
The order is sent for delivery
The machine shop is idle, waiting for work

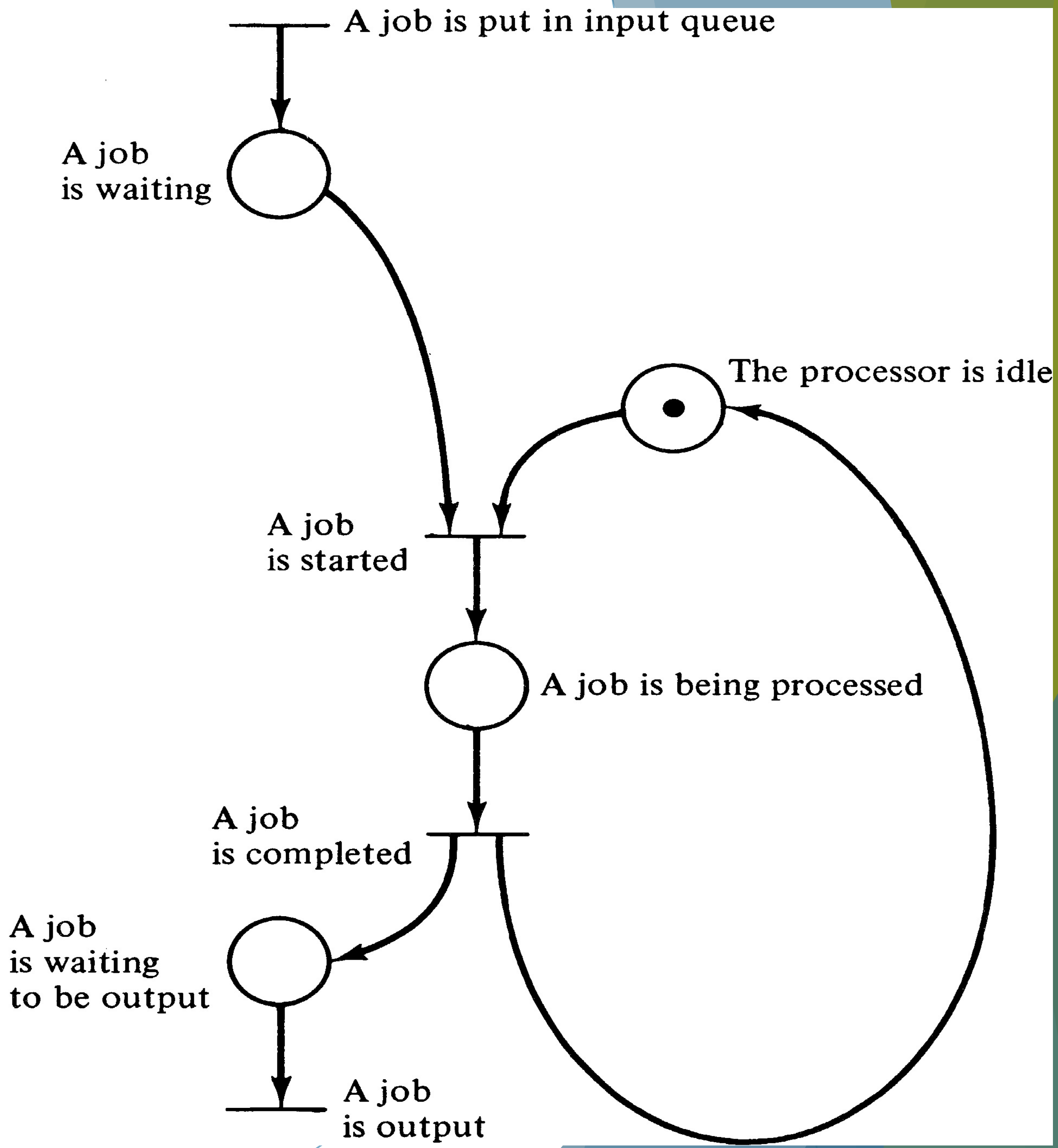
Computer System Job Processing Example

A computer system processes jobs from an input device and outputs the results on an output device.

Jobs appear on the input device.

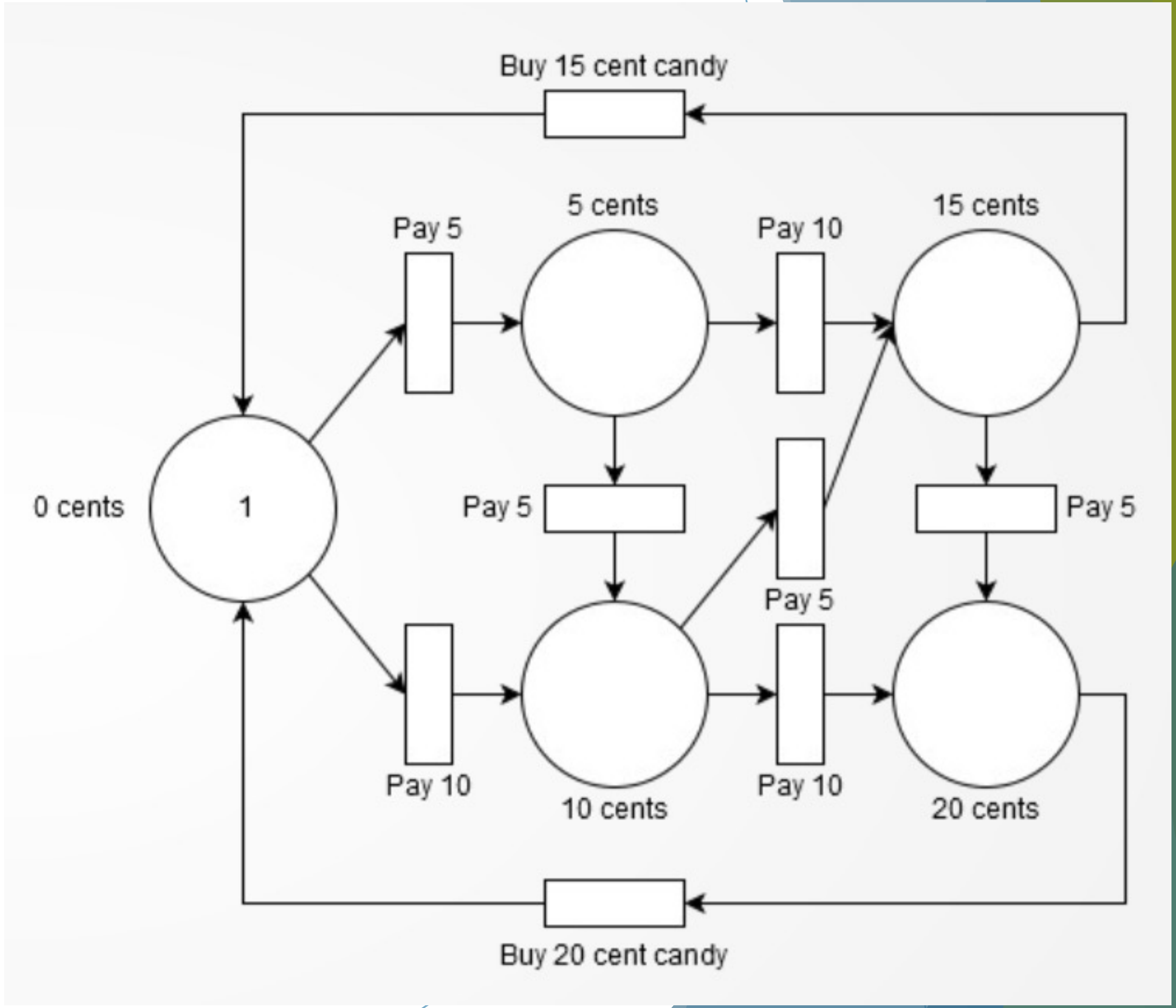
When the processor is free and there is a job on the input device, the processor starts to process the job.

When the job is complete, it is sent to the output device; the processor either continues with another job if one is available or waits until one arrives if there is no job yet on the input device.



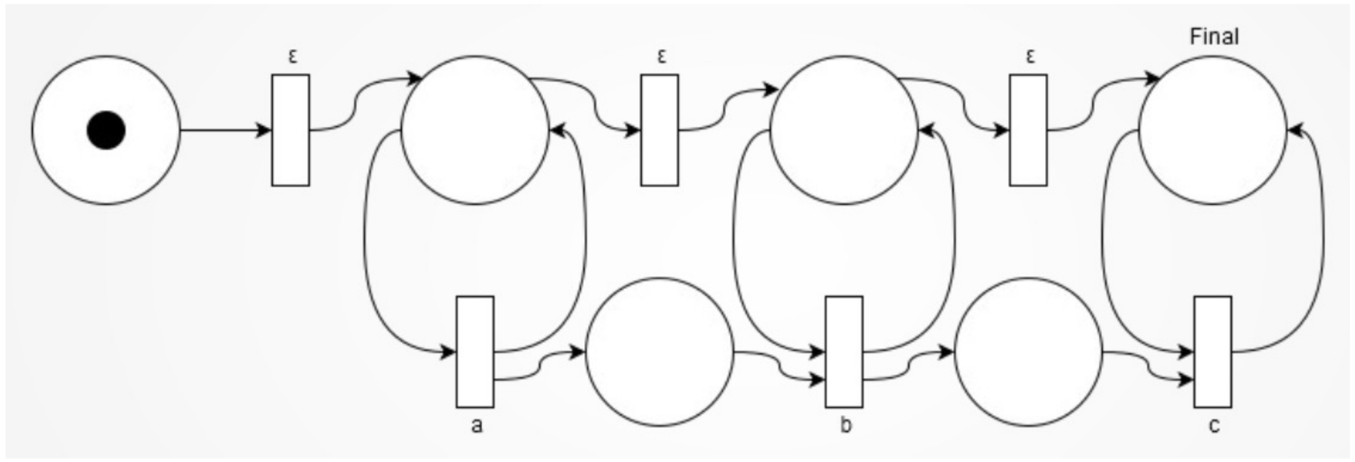
Candy Machine Example

Imagine a candy vending machine which only accepts nickles (5-cents) and dimes (10-cents), doesn't return change, and sells 15 and 20 cent candies only.



Formal Language Example

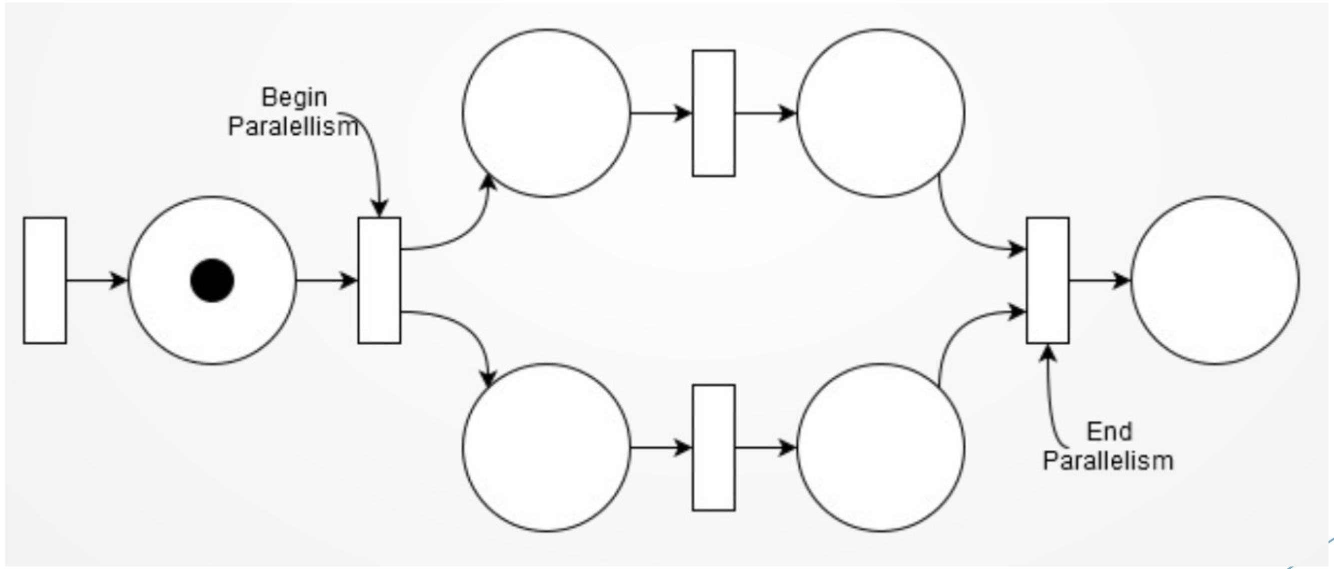
- Consider a formal language L = {$a^n b^n c^n$ | n >= 0), which is a context sensitive language.



- A Petri Net can recognize any context-sensitive, context-free, or regular language.

Parallelism Example

- Consider the simple case where we have a program that does one thing, then splits into two threads each of which performs an independent task, then continues in a single thread once both threads are done. We could model that behavior as follows.

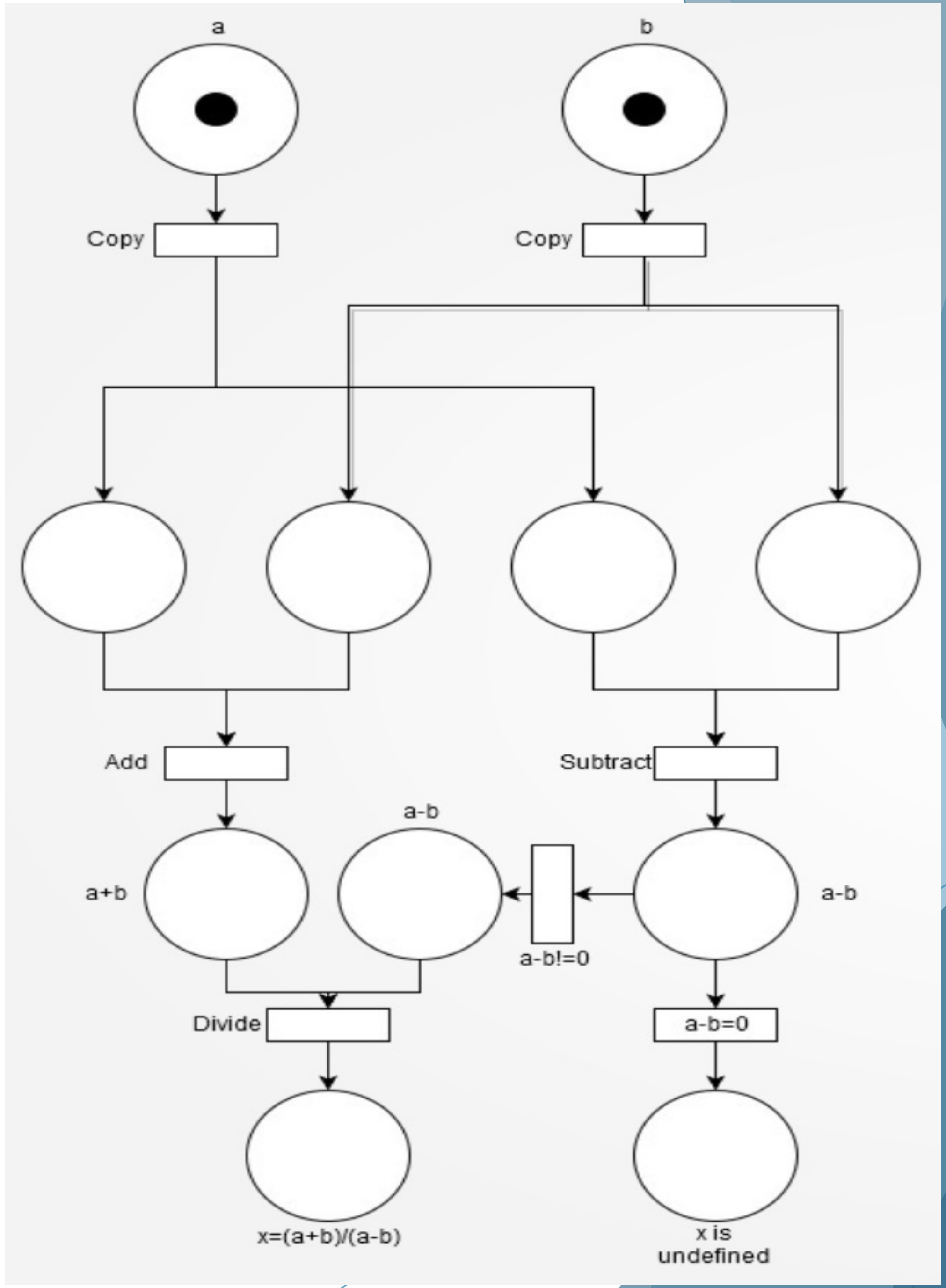


Data Flow Example

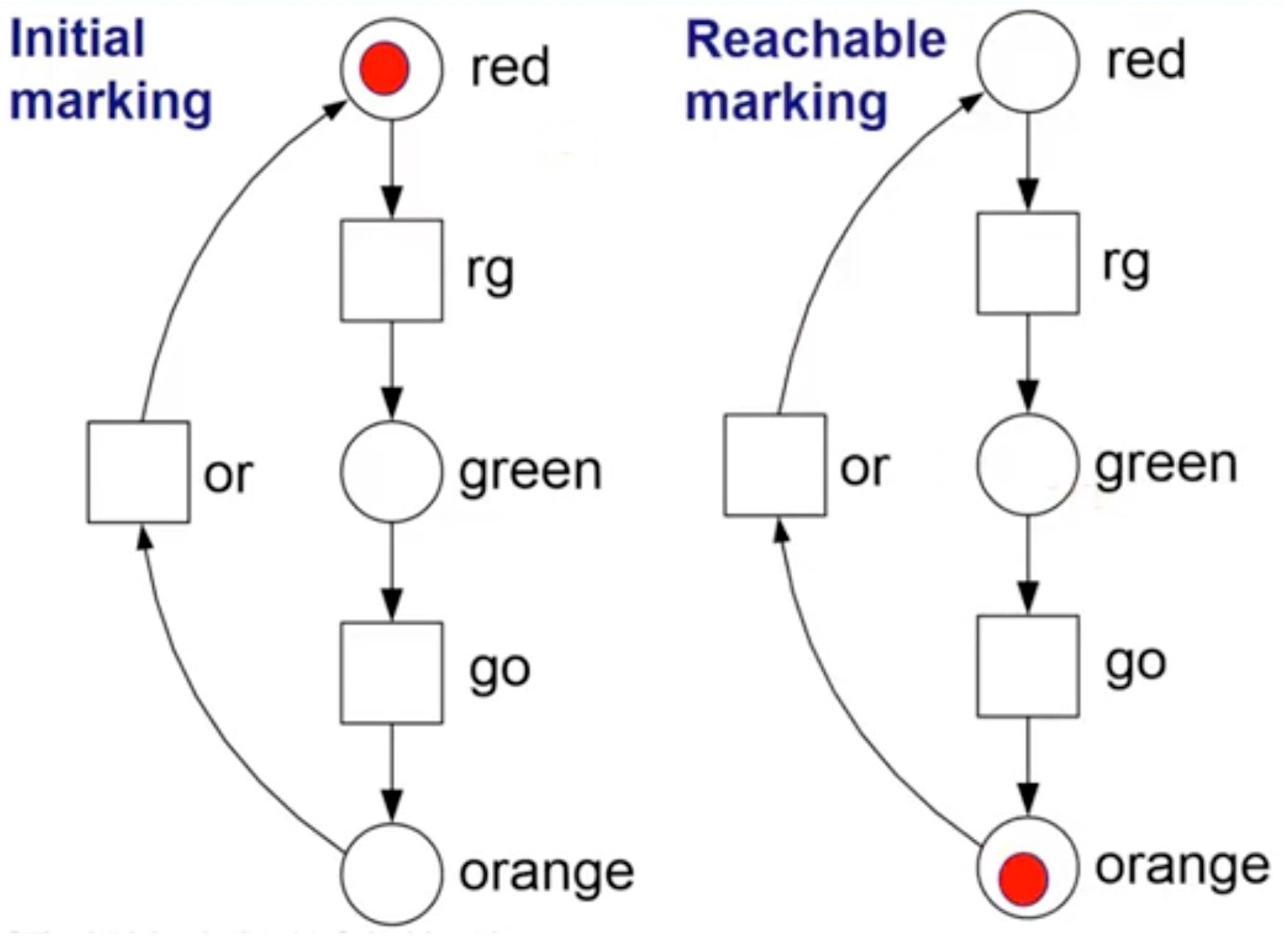
- A petri net can be used to map out the flow of data through a calculation.
- By assigning names to the places which represent value, we can easily model a calculation, where a token in a place marks that the data it represents is available for use.
- Let's model the calculation:

$$x=(a+b)/(a-b)$$

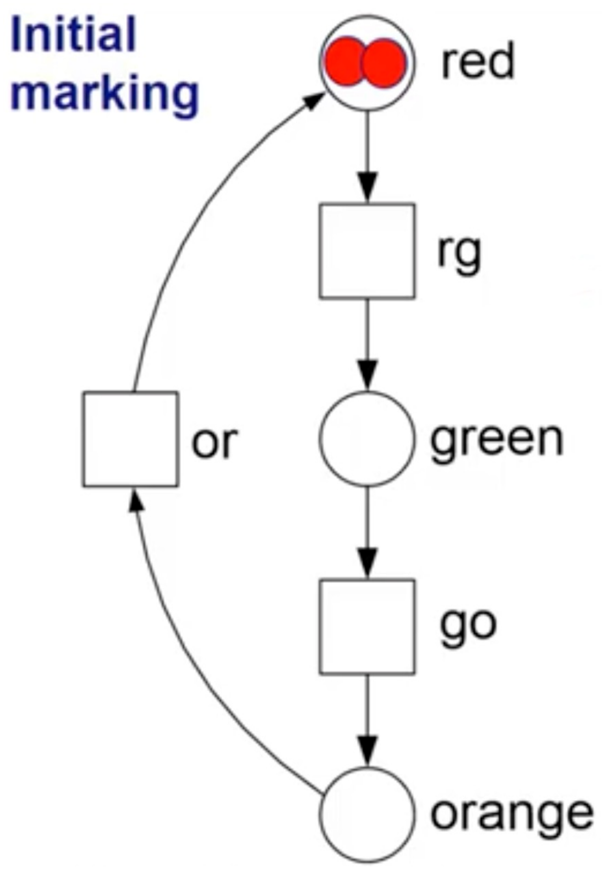
Note that the petri net does not contain any mechanism to actually perform the calculations as depicted.

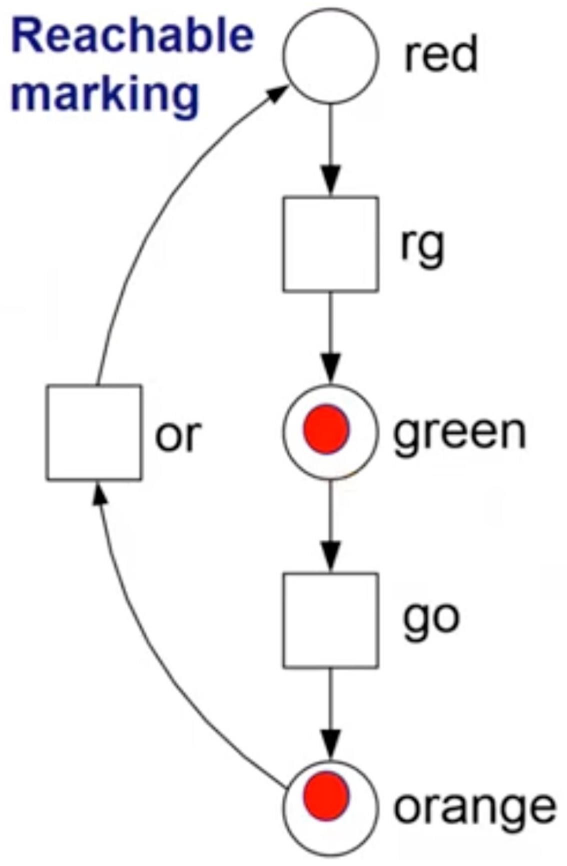


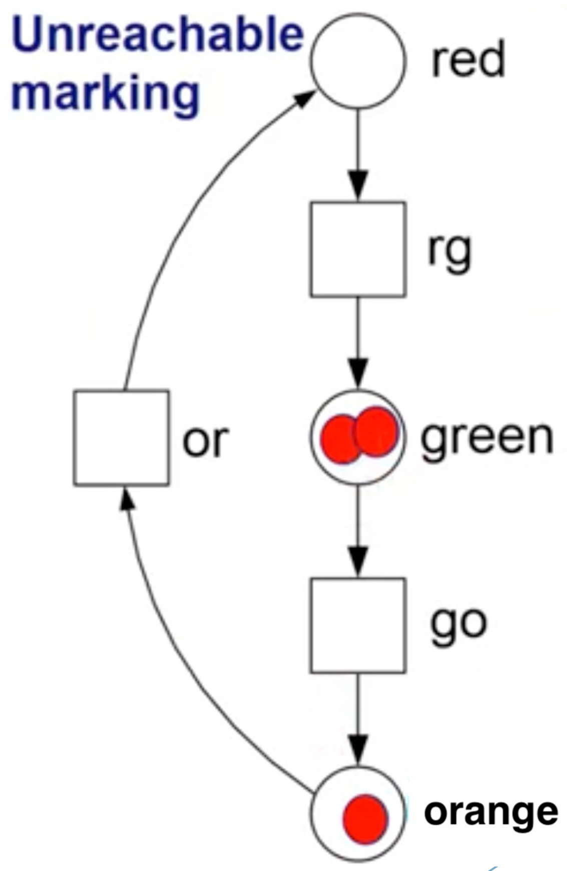
Traffic Light Example



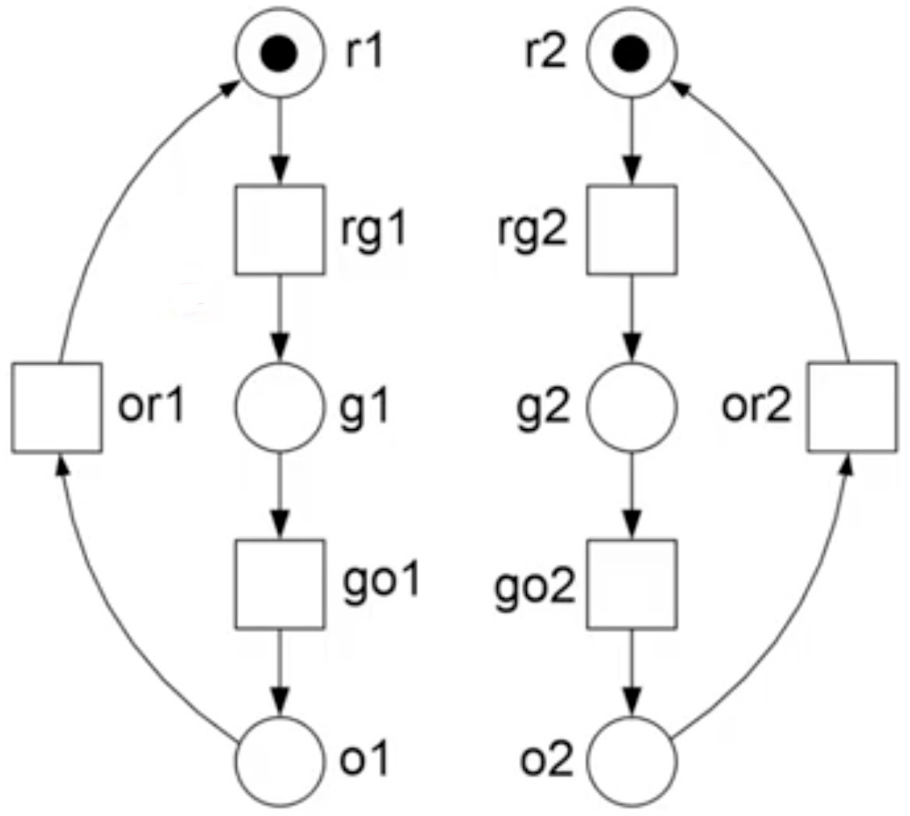
Traffic Light Example





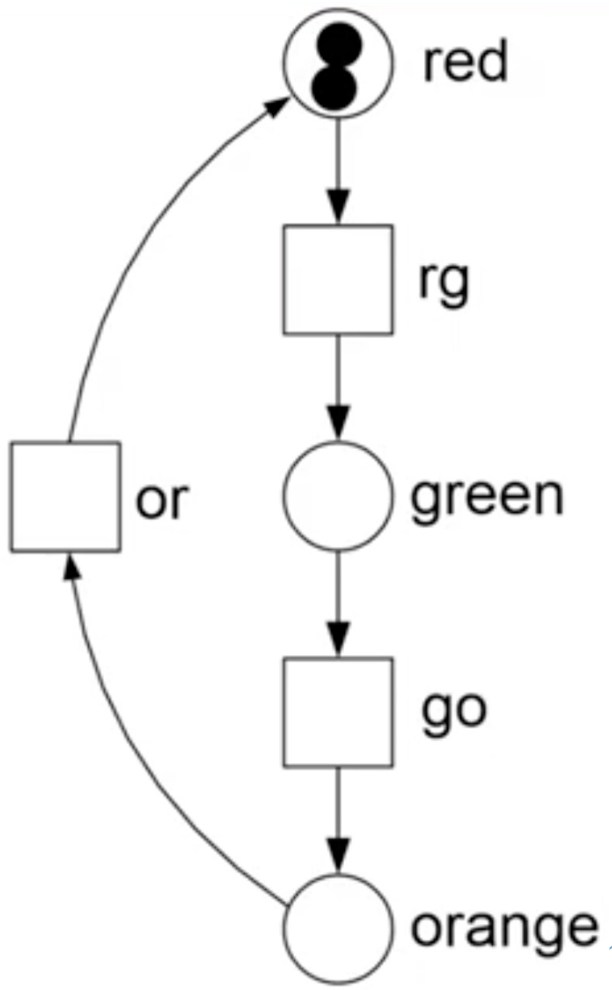


Two Traffic Lights



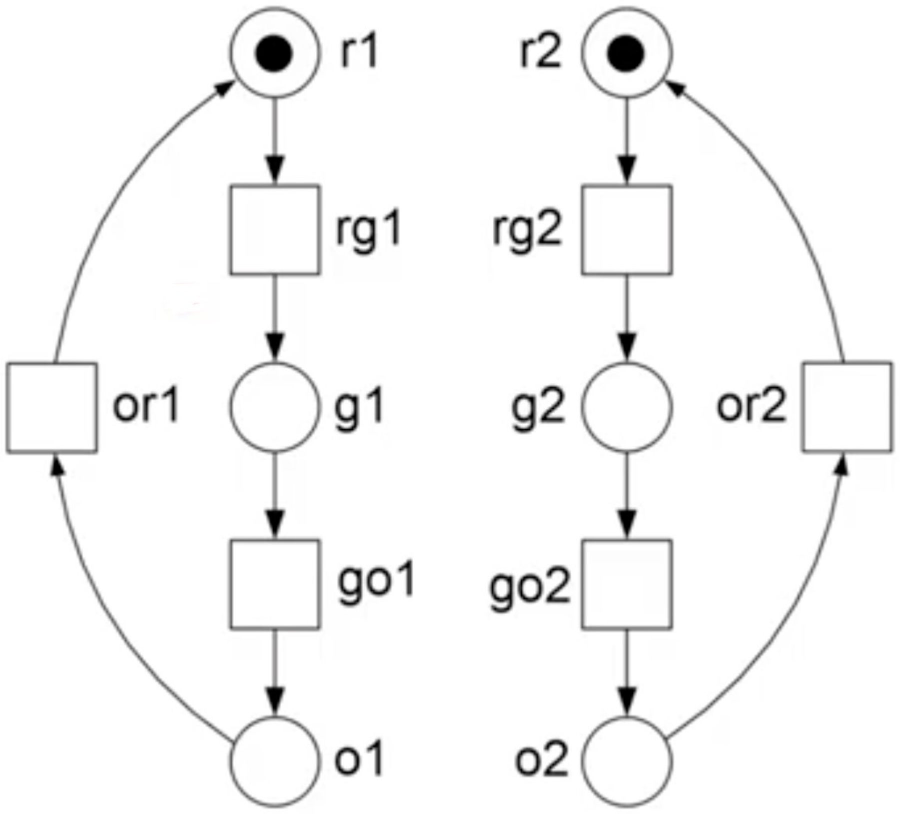
3x3 = 9 possible markings

OR



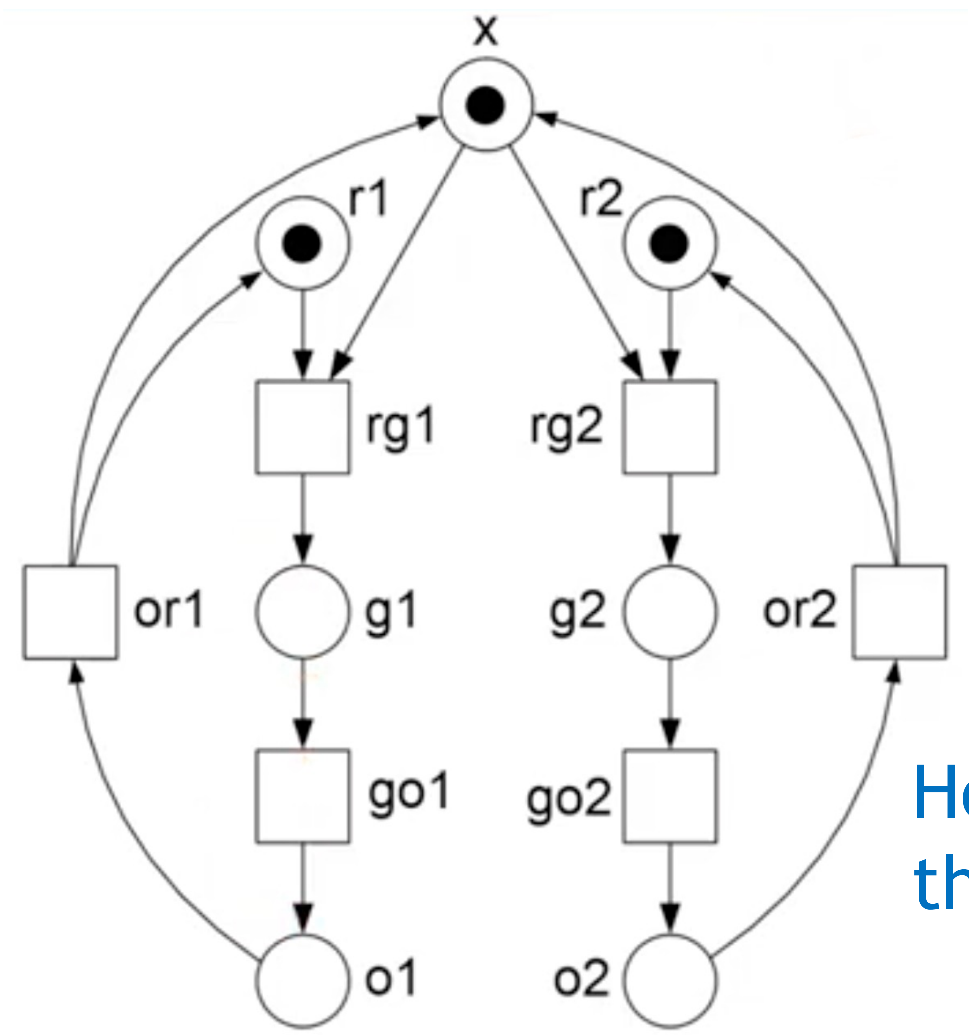
6 possible markings

Two Traffic Lights: Problem



How to make them safe?

Safe Traffic Lights: non-determinism



Any problem?

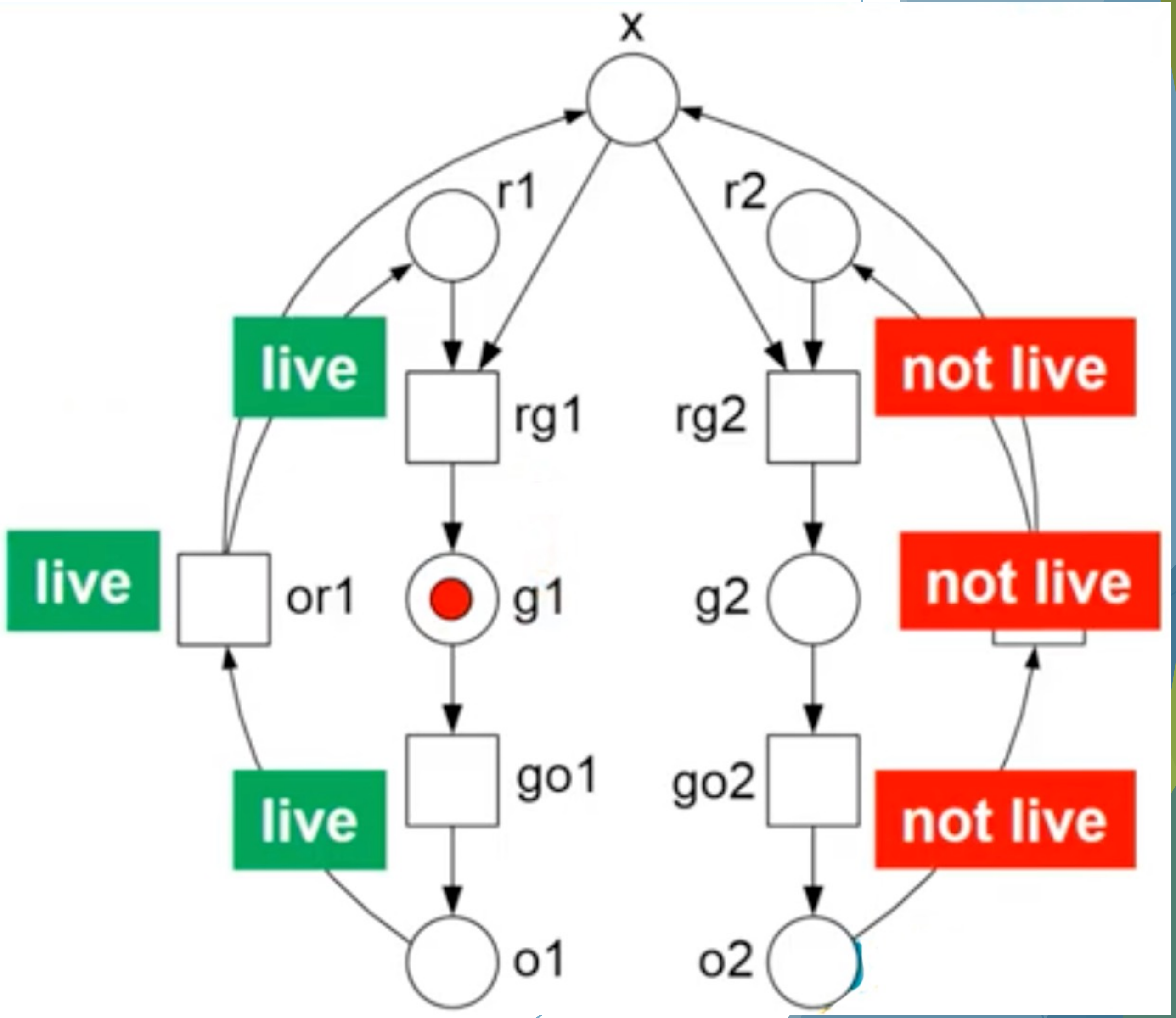
We want the traffic lights to alternate!

How can we do that in Petri nets?

Traffic Light: liveness

Deadlock-free but not live

- A transition t is live if from any reachable marking it is possible to reach a marking that enables t.
- A petri net is live if all transitions are live.
- A petri net that is live is deadlock-free



Reading Material

- Chapter 3: Aalst