EEE3096S

Specifications and Modeling (7)

Section 2.6 – Pertri Nets *(simplified version)*

L11

Embedded Systems II

Dr Simon Winberg

Electrical Engineering
University of Cape Town

Outline of Lecture

- What are Petri Nets
- History of Petri Nets
- Formal Definition of Petri Net
- Basics of Petri Nets
- Properties of Petri Nets
- More Examples of Petri Nets

Models of computation in this course

Communication/ local computations	Shared memory	Message passing Synchronous	Asynchronous
Undefined components	Plain text ✓, use cases ✓	Sequence Charts ✓, ICD		
Communicating finite state machines✓	StateCharts ✓		SDL
Data flow✓	Scoreboarding + Tomasulo Algorithm (☞ Comp.Archict.)		Kahn networks✓, SDF✓
Petri nets		C/E nets, P/T nets, …	
Discrete event (DE) model	VHDL*, Verilog*, SystemC*, …	Only experimental systems, e.g. distributed DE in Ptolemy (Ptolemy only discussed briefly)	
Von Neumann model	C, C++, Java	C, C++, Java with libraries CSP, ADA		

* Classification based on implementation with centralized data structures
SystemC will not be delved into detail. Only brief flavour of VHDL and Verilog given

What are Petri Nets?

Petri Nets def.: an abstract model for information flow and scheduling.

Major use:
Modeling events in a system where some events may occur concurrently; but there are constraints on the occurrences, precedence, and/or frequency of these occurrences.

History of Petri Nets

- Originally invented by Carl Adam Petri*, presented in his PhD thesis “Kommunikation mit Automaten” (1962).
- They were originally conceived as a technique for description and analysis of concurrent behaviour in distributed systems (seriously ahead of the times).
- Idea based on a few simple concepts, but very expressive.
- They have a simple graphical format and a precise operational semantics that makes them attractive for modeling static and dynamic aspects of processes.
- Many analysis techniques and tools exist that use them.
- Many extensions and variants have been defined over the years (we focus on the basic version).

* You can read the short version of his biography, if you are interested in such things, at http://www.informatik.uni-hamburg.de/TGI/PetriNets/history/CAPetriAndPetriNets.pdf

Definition of Petri Net

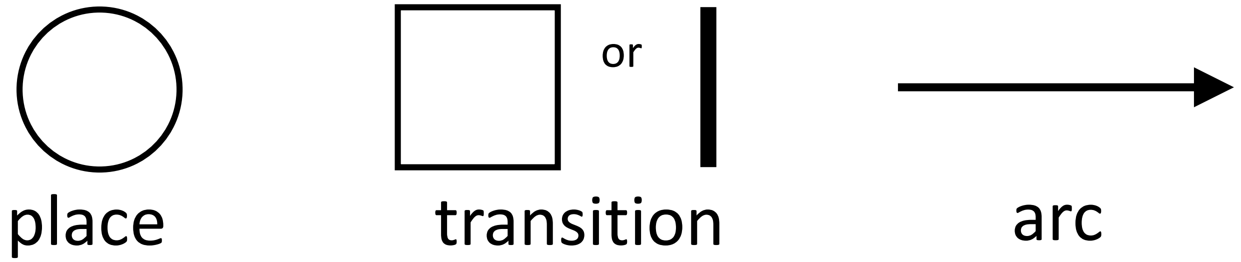
- ***C = (P, T, I, O)***
  - Places
    $P = \{ p_1, p_2, p_3, \ldots, p_n\}$
  - Transitions
    $T = \{ t_1, t_2, t_3, \ldots, t_n\}$
  - Input
    $I : T \rightarrow P^r$ (r = number of places)
  - Output
    $O : T \rightarrow P^q$ (q = number of places)

- marking μ : assignment of tokens to the places of Petri net $\mu = \mu_1, \mu_2, \mu_3, \ldots \mu_n$

Don't worry about this formal description now, let's just get into the models that are much more intuitive

Petri Nets: Basics

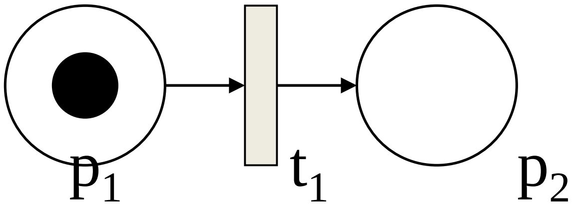
- A Petri Net diagrams are in the form of a ‘**directed bipartite graph*’** where the nodes are either *places* or *transitions.*
- **Places** : Represent **intermediate states** that may exist during a process. Places are represented by circles. Places can be the input/output of **transitions**.
- **Transitions :** These correspond to **activities** or **events** of which the process is made up. Transitions are represented by rectangles or thick bars.
- **Arcs** : Connect places and transitions in a way that places can only be connected to transitions and vice-versa.



* bipartite meaning consisting of pared (2-piece) parts

Basics of Petri Nets

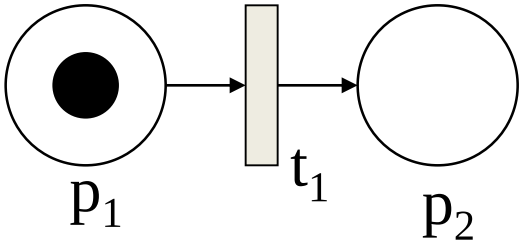
- Petri net essentially have two types of nodes: *places* and *transitions*. And arc exists only from a **place** to a **transition** or from a **transition** to a **place**.
- A place may have zero or more *tokens*.
- As indicated previously places, transitions and arcs are represented respectively by: circles, bars, arrows. A token (or 'execution ticket') is represented by a black dot.
- An example Petri Net is thus:



Space P1 holds an execution token

Basics of Petri Nets

- Below is an example Petri net with two places and one transaction.
- The transition node is ready to ***fire*** if and only if there is at least one execution token at each of its input places… for example in the diagram below transition t_1 is ready to fire.



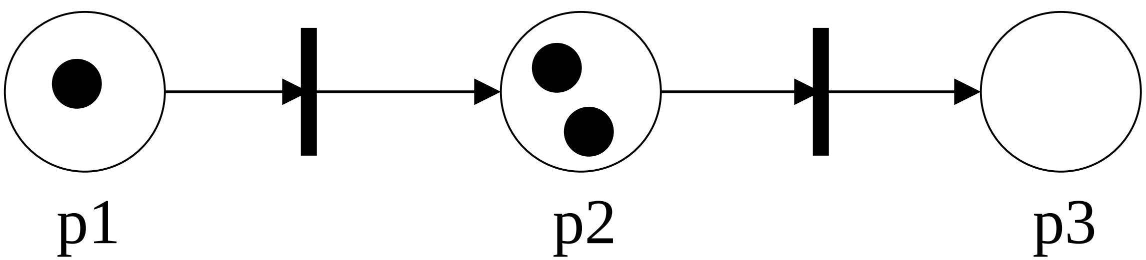
state transition of form

(1, 0) → (0, 1)
(i.e. this state of the spaces can be listed in this format)

p_1 : input place p_2: output place

Petri Net Marking

- Places in Petri nets can contain any number of tokens.
- The distribution of tokens across all of the places in a net is called a **marking**. For a Petri net an *initial marking* M_0 needs to be specified.
- Marking assigns tokens to places; formally, a marking M of a Petri net N = (P,T,F) is a function **M: P -> NAT**.



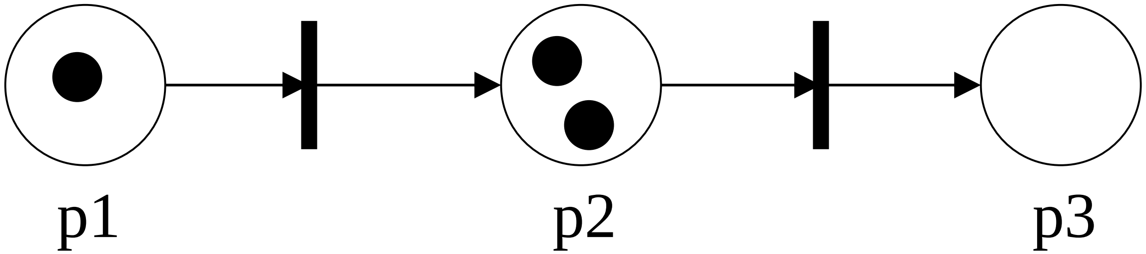
- The marking above is formally captured by the following marking $M = \{(p_1,1),(p_2,2),(p_3,0)\}$.
- Can use M(p) = markings (i.e. number tokens) in space p, e.g. $M(p_1,) = 1$

State of a Petri Net

- The state of a Petri net can be compactly described as indicated in this example:

  $\mathbf{1p_1+2p_2 + 0p_3}$ is the state with one token in place p1, two tokens in p2 and no tokens in p3. (the same as {(p1,1),(p2,2),(p3,0)})

  We can also represent this state in the following (equivalent) way: $\mathbf{p_1+2p_2}$. *This is how the Petri net could look:*



The ordering function ≥ over a set of possible states is defined as:

Given Petri net N = (P,T,F) and markings M and M',

M ≥ M' iff for all p in P: **M(p) ≥ M'(p).**

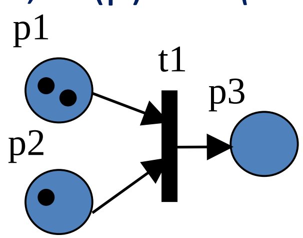
similarly can define M>M' or more succinctly M>M' iff M≥M' and M≠M'

Transitions Firing

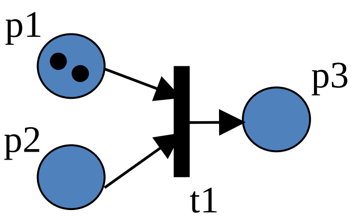
- The operational semantics of Petri nets are explained by the notion of a transition executing or “**firing**”. A transition in a Petri net can “fire” whenever there are one or more tokens in each of its input places.
- The execution of a transition occurs in accordance with the following firing rules:

1. A transition t is said to be **enabled** if and only if each input place p of t contains at least one token. <u>Only enabled transitions may fire</u>.
   – Formally, a transition t is enabled in a marking M iff for each p, with p $\in \bullet$t, M(p) > 0. *(see definition 2.7 of [DE95])*

t1 enabled and may fire:

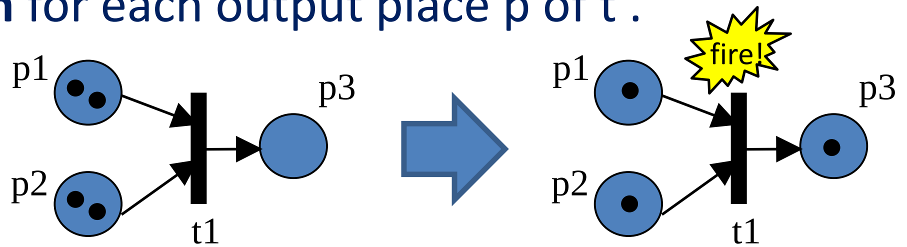


t1 is not enabled:



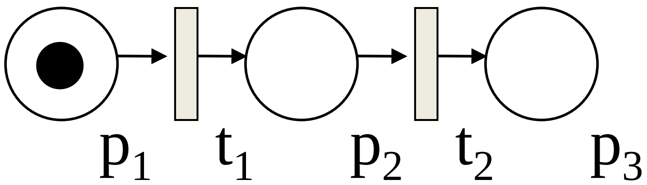
2. If transition t fires, then t **consumes one token** from each input place p of t and **produces one token** for each output place p of t .

t1 fires. When a transition fires, the marking and the state of the Petri Net change.

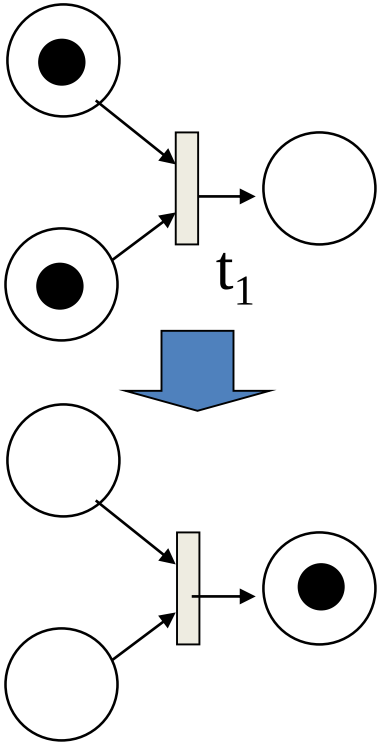


Properties of Petri Nets

- **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.

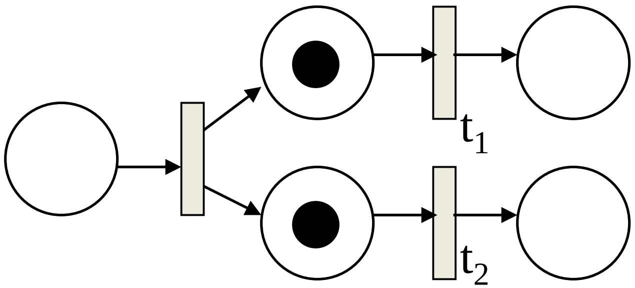
These properties are all just consequences of the Petri net behavior explained previously

Properties of Petri Nets

- Concurrency

  t_1 and t_2 are concurrent.

- With this property, the Petri Net is able to model systems of distributed control with multiple processes executing concurrently in time.

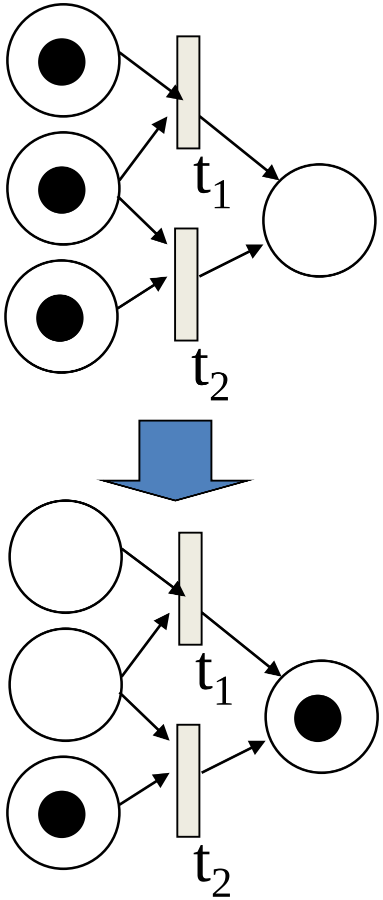


(Undesirable) Properties of Petri Nets

- Conflict

  t1 and t2 are both ready to fire but the firing of any leads to the disabling of the other transitions.

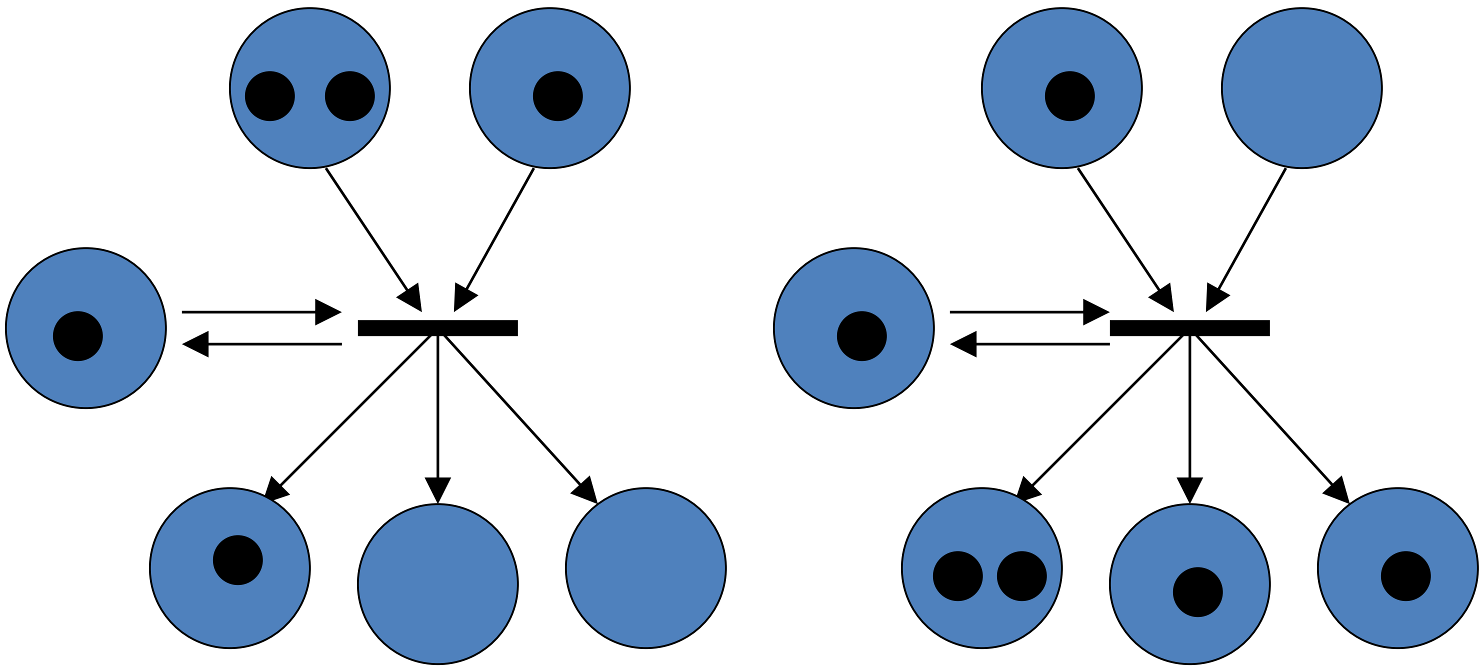
- This is not necessarily a desirable property to have in a design



Adapted from Nick Russell & Arthur Hofstede (2009) "Formal Approaches to Business Processes through Petri Nets" Yawl. Presentation slides.

More Examples of Petri Nets

Firing a Transition: Example



BEFORE

AFTER

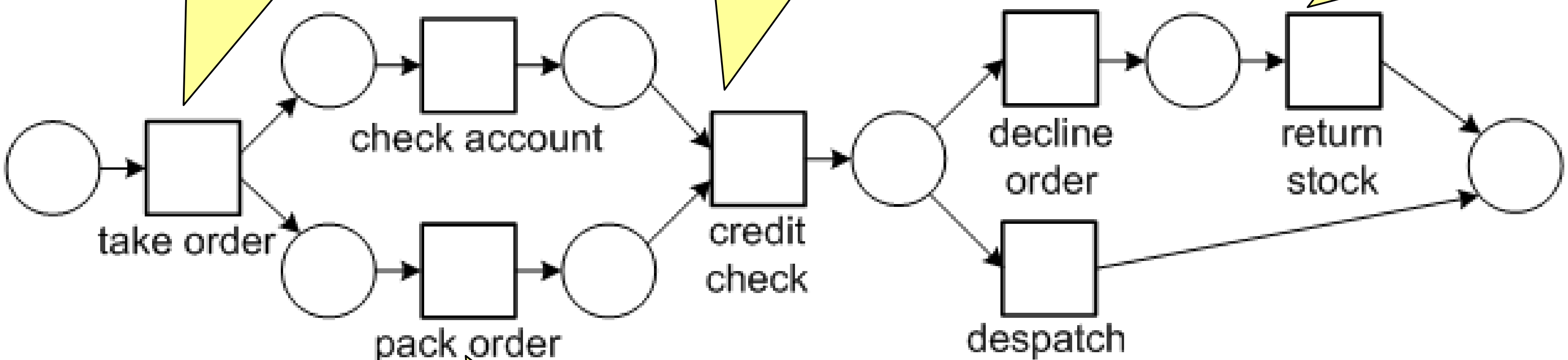
Adapted from Nick Russell & Arthur Hofstede (2009) "Formal Approaches to Business Processes through Petri Nets" Yawl. Presentation slides.

Petri nets: Order Fulfillment Example

First, a ***take order*** task is executed.

When ***pack order*** and ***check account*** tasks have been both completed, the ***credit check*** task is executed.

If the customer has not sufficient credit the ***decline order*** runs and, finally, the ***return stock*** task ensures that the items from the order are returned to the warehouse.

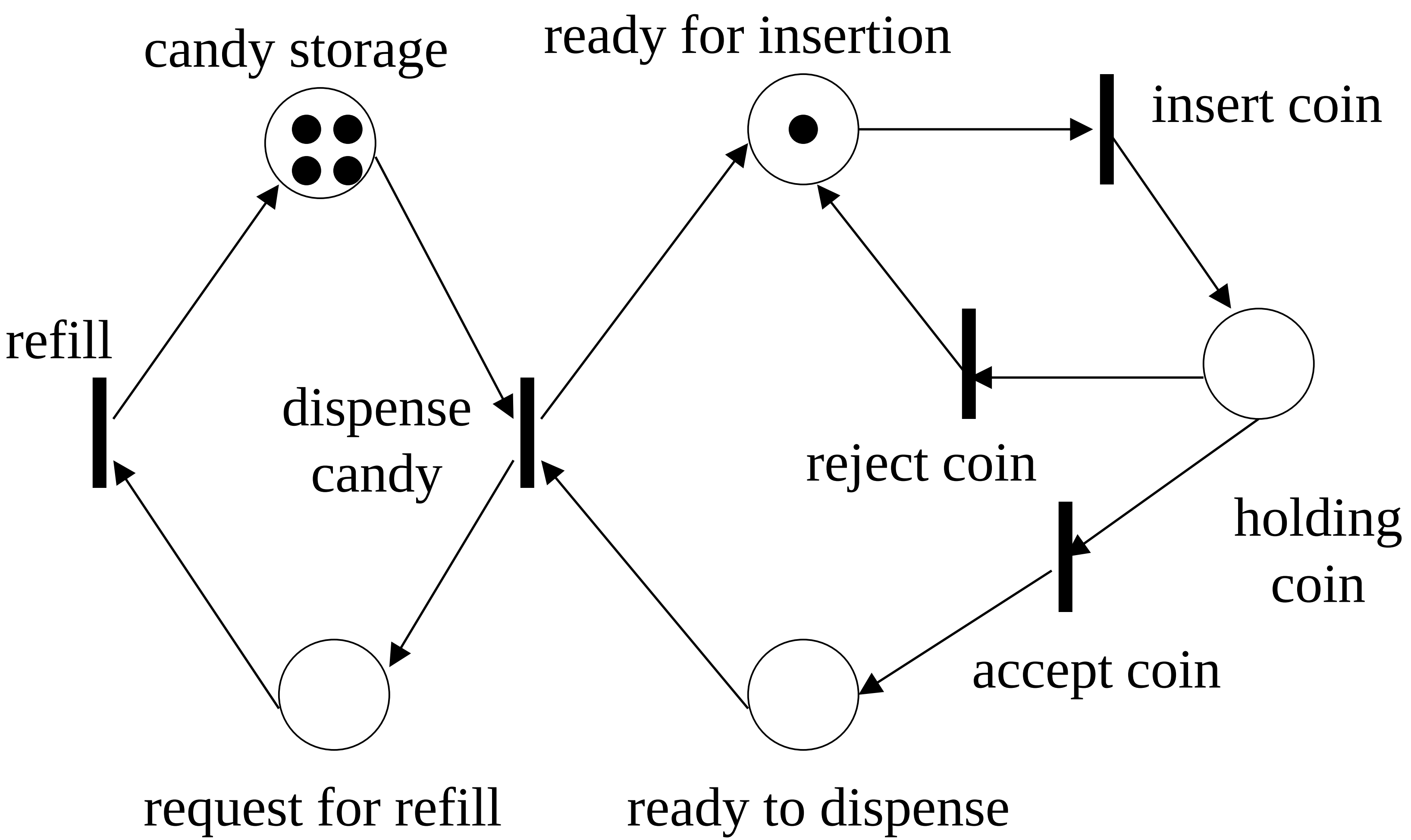


Then, ***pack order*** and ***check account*** tasks are executed in parallel.

If the customer has sufficient credit remaining, the order is ***despatched***.

Adapted from Nick Russell & Arthur Hofstede (2009) "Formal Approaches to Business Processes through Petri Nets" Yawl. Presentation slides.

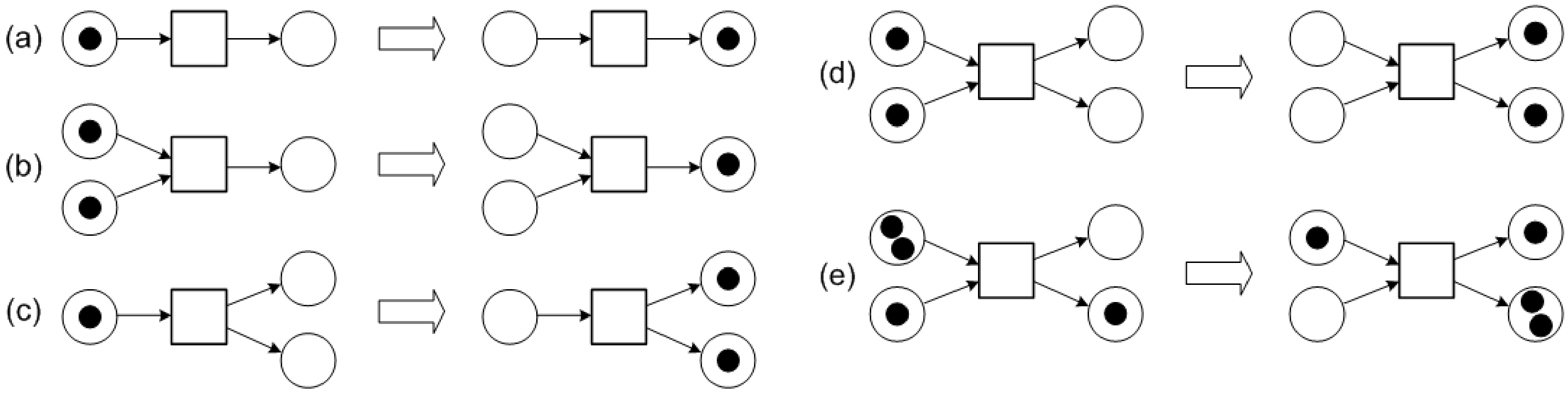
Petri nets: Example of a vending machine



If you follow through the process, you will see that you cannot complete the dispense candy until the accept coin process is completed.

Adapted from Nick Russell & Arthur Hofstede (2009) "Formal Approaches to Business Processes through Petri Nets" Yawl. Presentation slides.

Firing Transitions: Further Examples



- It is assumed firing of a transition is an atomic action that occurs instantaneously and cannot be interrupted.
- If there are multiple enabled transitions, any one of them may fire; however, for execution purposes, it is assumed that **they cannot fire simultaneously.**
- An enabled transition is not forced to fire immediately but can do so at a time of its choosing.
- These features make Petri nets particularly suitable for modeling concurrent process executions.

Properties

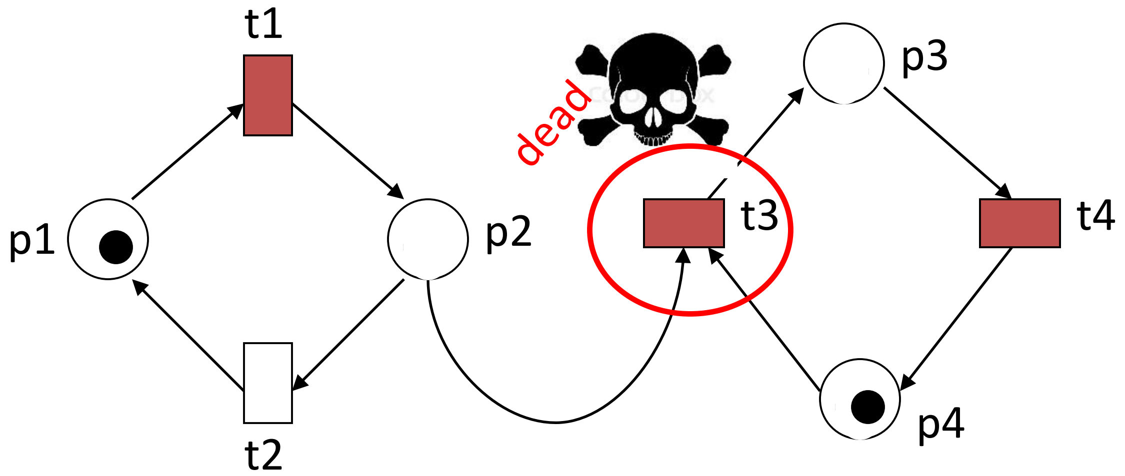
- A Petri net with initial marking M0 is live if, no matter what marking has been reached from M0, it is possible to ultimately fire *any transition* by progressing through some further firing sequence.
- The notion of *liveness* is important since it demonstrates that at least one transition can fire in every reachable state. i.e.:

  A live Petri net guarantees deadlock-free operation.
- A Petri net N with initial marking M_0 is ***deadlock free*** iff every reachable marking enables some transition

Properties

- A Petri net N with initial marking M_0 is ***k-bounded*** iff for every reachable marking M, M(p) $\leq$ k (k is the minimal number for which this holds)
  - A 1-bounded net is called ***safe***.
  - The property of *boundness* ensures that the number of tokens cannot grow arbitrarily big.
- A Petri net N is ***strongly connected*** iff for every pair of nodes (places or transitions) x and y there is a path from x to y and vice-versa.

A bounded but non-live Petri net



M0 = (1,0,0,1)

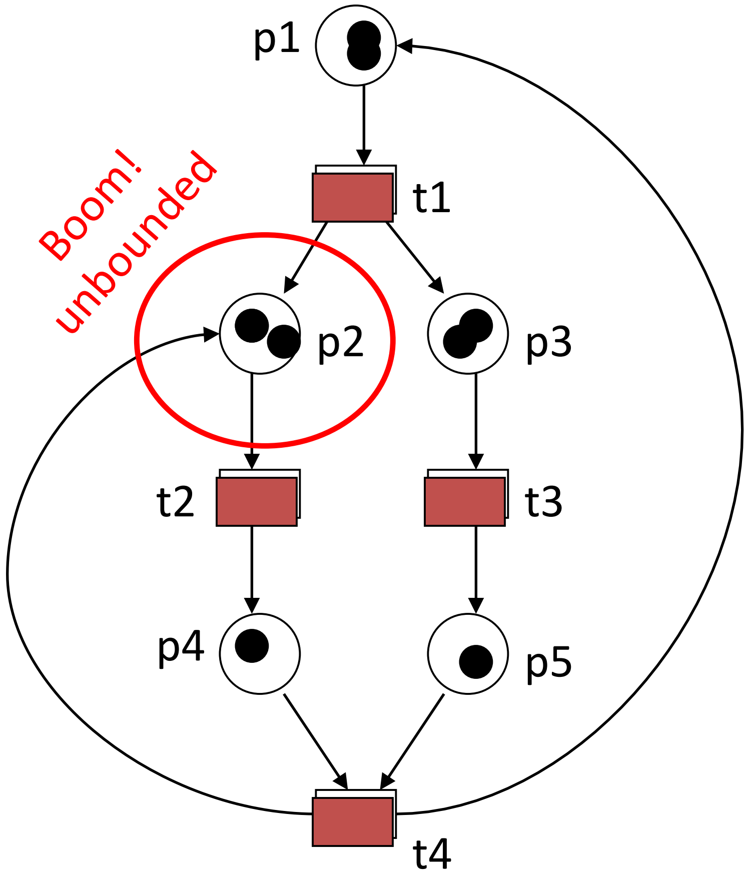
M1 = (0,1,0,1)

M2 = (0,0,1,0)

M3 = (0,0,0,1)

A bounded but non-live Petri net

Petri Net unbounded but live?



$M0 = (1, 0, 0, 0, 0)$

$M1 = (0, 1, 1, 0, 0)$

$M2 = (0, 0, 0, 1, 1)$

$M3 = (1, 1, 0, 0, 0)$

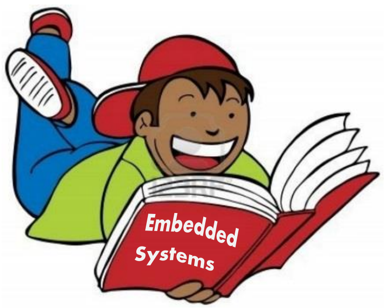
$M4 = (0, 2, 1, 0, 0)$

An unbounded but live Petri net

The Next Episode…

Lecture P03

P03: RPi GPIO



Reminder: Read section 2.6, 2.7, 2.10
(section 2.7, 2.8 optional reading)

References

- “Petri net” https://en.wikipedia.org/wiki/Petri_net
- Nick Russell & Arthur Hofstede (2009) "Formal Approaches to Business Processes through Petri Nets" Yawl. Presentation slides.
- “C. A. Petri ‘Petri Nets’” a short biography by Wilfried Brauer, Wolfgang Reisig http://www.informatik.uni-hamburg.de/TGI/PetriNets/history/CAPetriAndPetriNets.pdf