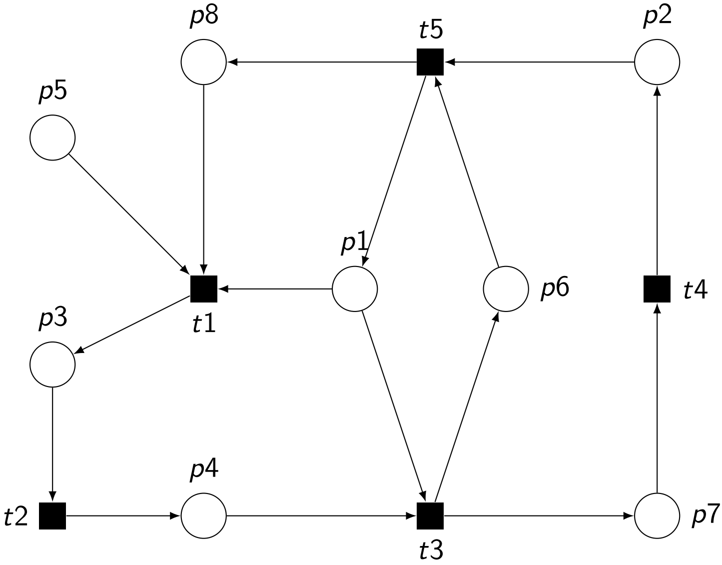
Introduction to Petri Nets

Adrián Puerto Aubel

February 7, 2025


p8
t5
p2
p5
p1
p6
t4
t1
p3
p4
t2
t3
p7

Models of Concurrency

Petri Nets

Class of formal models of *concurrency*

- They are formal models:
  Directed graphs - “Distributed version of automata”.
- They model concurrency:
  Automata are *sequential*. Petri nets represent processes that can run “in parallel”.
- They are a whole class of models:
  Different types of Petri Nets for different problems.

Petri Net Systems

Petri Net + Dynamic Behaviour

- A Petri Net is a static structure “shape of a network”
- A Petri Net System can “run” - execute actions.

Motivation

Real Computers - μ-architecture

- CPU = Control Unit + Arithmetic Units + Registers
- Control Unit = Finite State Automaton
- Communication between units uses lines:
  Cable with a binary value
- Clock is crucial to know which message is on a line.
- $\Rightarrow$ Sequential Computation : Synchronous Systems

Distributed System (Several Computers)

- Different Clocks: Asynchronous Systems
- A CPU cannot always guess the state of other CPU's
- Relies on Communication Protocols
- The state of the system is determined by the local states of each CPU

Like puzzles? $\rightarrow$ www.nandgame.com

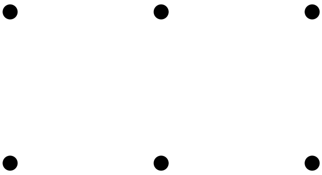
Basic Definitions

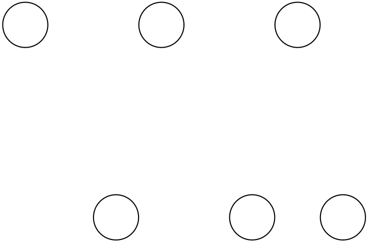
Automata

- States: Control
- Alphabet: Instructions
- Arcs (arrows): Effect of an instruction at a given state

Petri Nets

- Places: local states
- Transitions: change of state
- Arcs: Effect of a transition on the local states.



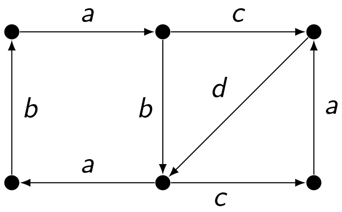


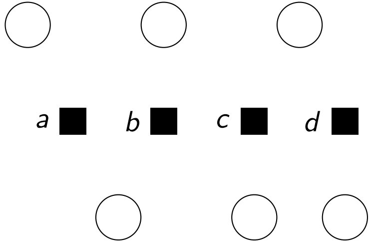
Automata

- States: Control
- Alphabet: Instructions
- Arcs (arrows): Effect of an instruction at a given state

Petri Nets

- Places: local states
- Transitions: change of state
- Arcs: Effect of a transition on the local states.





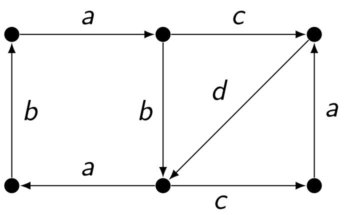
Basic Definitions

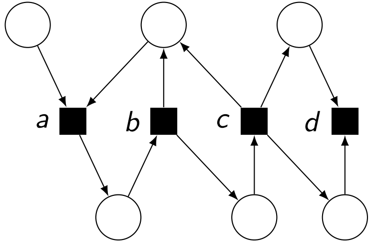
Automata

- States: Control
- Alphabet: Instructions
- Arcs (arrows): Effect of an instruction at a given state

Petri Nets

- Places: local states
- Transitions: change of state
- Arcs: Effect of a transition on the local states.





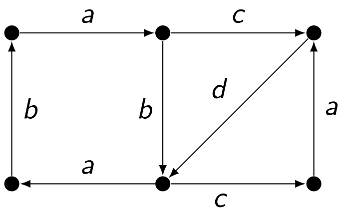
Basic Definitions

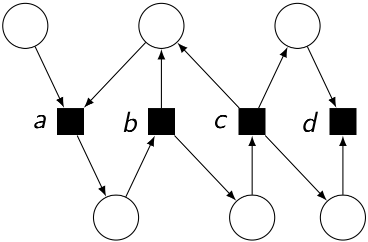
Automata

- States: Control
- Alphabet: Instructions
- Arcs (arrows): Effect of an instruction at a given state

Petri Nets

- Places: local states
- Transitions: change of state
- Arcs: Effect of a transition on the local states.





Q: What's missing?

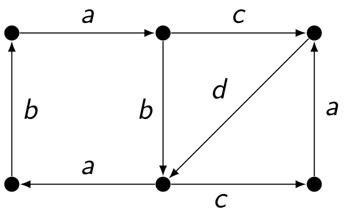
Basic Definitions

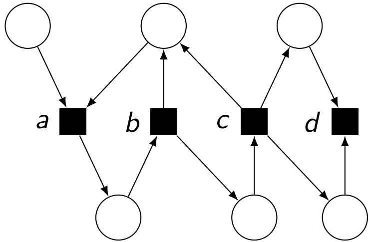
Automata

- States: Control
- Alphabet: Instructions
- Arcs (arrows): Effect of an instruction at a given state

Petri Nets

- Places: local states
- Transitions: change of state
- Arcs: Effect of a transition on the local states.





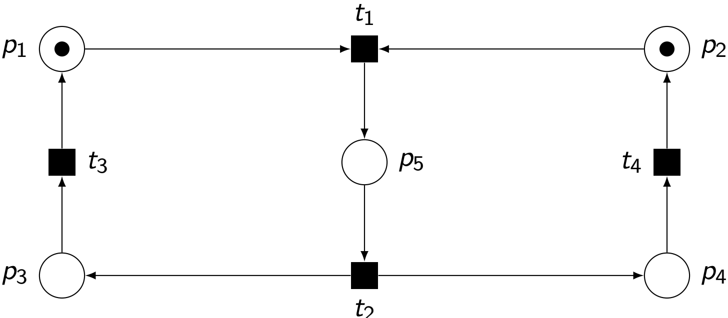
Q: What's missing? A: Initial state, accepting states

Most Basic Model

Elementary Net Systems

An E.N.S. is a tuple $\Sigma = (B, E, \mathcal{F}, m_0)$

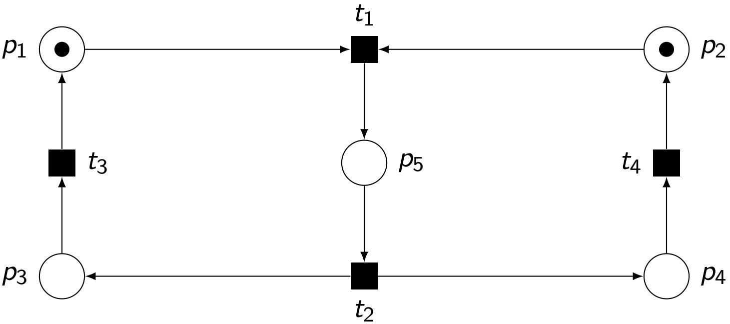
- B: Places $\rightarrow$ Conditions
- E: Transitions $\rightarrow$ Events
- $\mathcal{F} \subseteq (B \times E \cup E \times B)$: Arrows $\rightarrow$ Flow relation
- $m_0 : B \rightarrow \{0, 1\}$: Initial Marking (Global State): assigns 0 or 1 token to each condition.



Neighbourhoods of Events

Given an Event $e \in E$

- Pre-conditions: ${}^{\bullet}e = \{b \in B \mid (b, e) \in \mathcal{F}\}$
  conditions that "feed" the event
- Post-conditions: $e^{\bullet} = \{b \in B \mid (e, b) \in \mathcal{F}\}$
  conditions which are "fed" by the event.
- Two events e_1, e_2 are *independent* iff $({}^{\bullet}e_1 \cup e_1^{\bullet}) \cap ({}^{\bullet}e_2 \cup e_2^{\bullet}) = \emptyset$



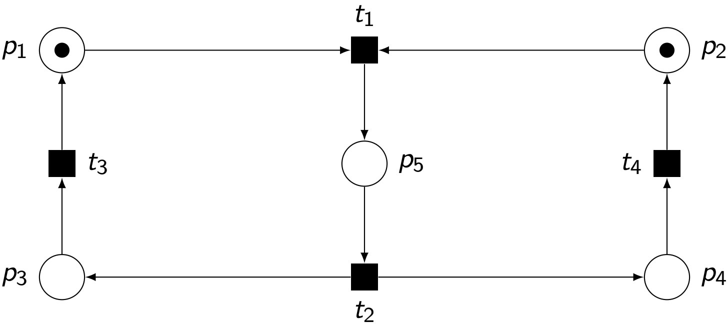
E.N.S. Dynamics

Enabled Events

Event e is enabled at marking m: $m[e\rangle$ iff:

- each $b \in {}^\bullet e$ has $m(b) = 1$ (all pre-conditions are true), AND
- each $b \in e^\bullet$ has $m(b) = 0$ (all post-conditions are false)

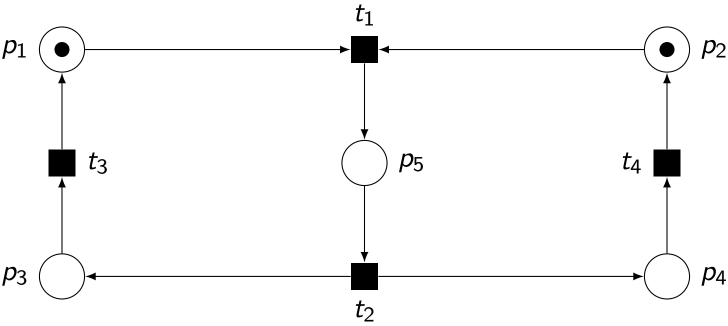
Two events e_1, e_2 are *independent* iff $({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset$ Note: several events can be enabled at the same marking.



E.N.S. Dynamics

Firing Rule

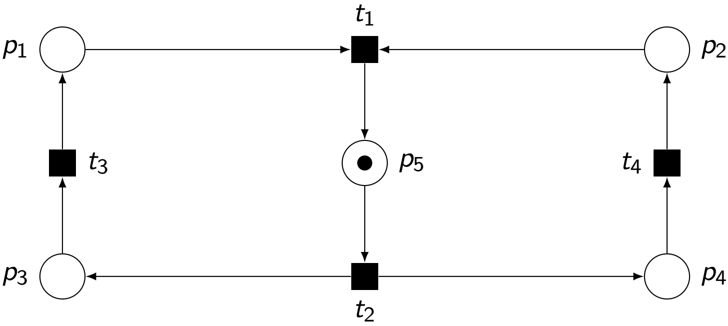
- When an event is enabled, it may *fire*:
- $m_1[e\rangle m_2$ means that:
  - e is enabled at m_1, AND
  - $m_2 = (m_1 \setminus {}^{\bullet}e) \cup e^{\bullet}$



E.N.S. Dynamics

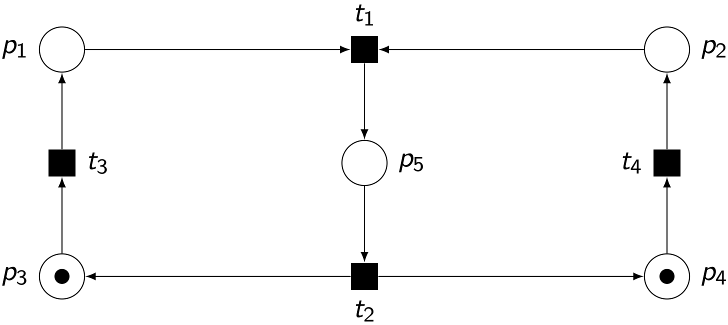
Firing Rule

- When an event is enabled, it may *fire*:
- $m_1[e\rangle m_2$ means that:
  - e is enabled at m_1, AND
  - $m_2 = (m_1 \setminus {}^\bullet e) \cup e^\bullet$



Firing Rule

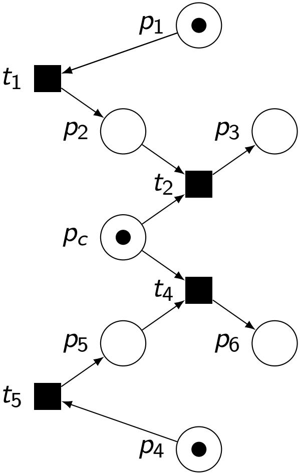
- When an event is enabled, it may *fire*:
- $m_1[e\rangle m_2$ means that:
  - e is enabled at m_1, AND
  - $m_2 = (m_1 \setminus {}^\bullet e) \cup e^\bullet$

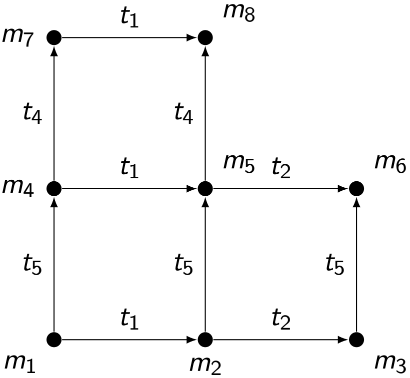


Global Behaviour: Automaton

Marking Graph

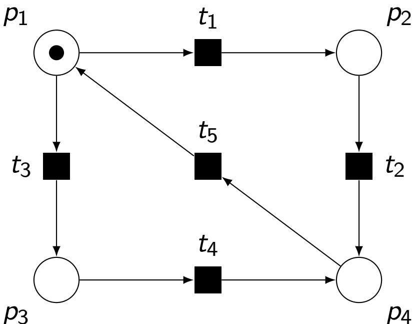
- Each node is a marking m
- We add an arc (m_1, m_2) with label e, if $m_1[e\rangle m_2$.

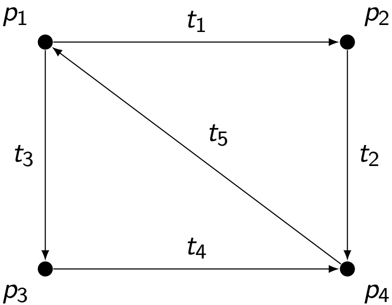




F.S.A.

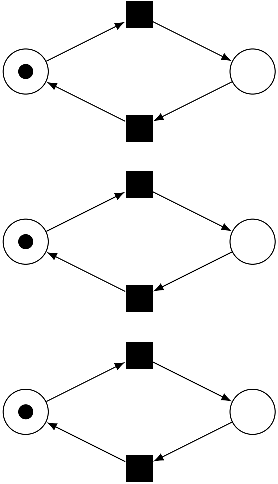
- E.N.S such that each event has ONLY 1 pre-condition and 1 post-condition
- only 1 token in the whole system
- then E.N.S. $\simeq$ Marking Graph
- Finite State Automaton



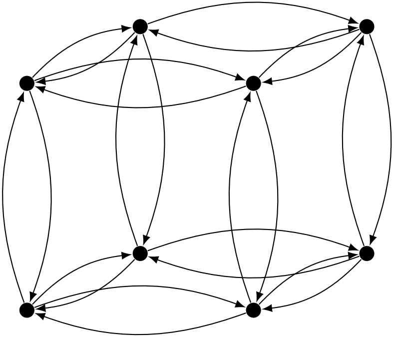


State Space Explosion

K components of size N



Size $= N * K$



Size$= N^K$

Motivation Part 2

Popularity

- Petri nets are widely used in the industry:
  System design: Software, Hardware, Logistics, etc. . .
- Two main reasons:
  - Modelling Power: Expressivity, Readability
  - Analysable: Algorithms for Verification

Modelling

Extensions of the basic model give flexibility.

Useful design tools.

• More features = More expressivity

Analysis

Good algorithms exist for verification.

Safety, Serviceability, Security, etc. . .

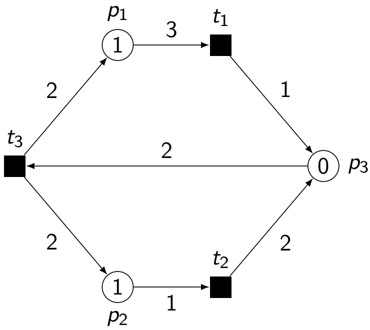
• Restrictions $\Rightarrow$ More algorithms (or more efficient).

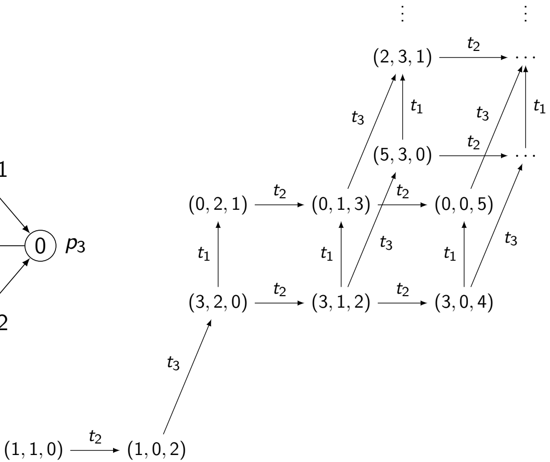
Place Transition Systems: Definitions

Place Transition Systems

- P: Places $\rightarrow$ Counters
- T: Transitions $\rightarrow$ Consume and Produce
- $\mathcal{F} : (B \times E \cup E \times B) \rightarrow \mathbb{N}$: Arcs are now weighted
- $m : P \rightarrow \mathbb{N}$: Marking assigns a number to each place
- Firing rule: $m_1[t\rangle m_2$
  - $\forall p \in P : \mathcal{F}(p, t) \leq m_1(p)$
    Places have enough tokens for the transition to fire, AND
  - $\forall p \in P : m_2(p) = m_1(p) - \mathcal{F}(p, t) + \mathcal{F}(t, p)$
    The weights on the arcs indicate how much is consumed and produced.

Place Transition Systems are VAS





$$t_1 = (-3, 0, 1); t_2 = (0, -1, 2); t_3 = (2, -2, 0)$$

Note: Unbounded Behaviour!

Problems

Boundedness

A P/T net system is *bounded* iff its set of reachable markings is finite.

- General case: PSPACE and PSPACE-complete if $|P| \geq 4$,
- PTIME for conflict-free nets (no choices)

Problems

Reachability

Given a P/T net system with initial marking m_0 and target marking m_t, decide whether m_t is reachable from m_0.

- In general, decidable but primitive recursive space!
- Undecidable if we allow for (at least) two zero-test arcs.
- 2EXPTIME if $|P| \leq 5$
- PSPACE-complete for 1-safe nets ($\simeq$ E.N.S)
- NP-complete for nets without cycles, and also for conflict-free nets (no choices).
- PTIME for bounded conflict-free nets
- PTIME for marked graphs
- PTIME for nets that are live, bounded, cyclic and free-choice.

Problems

Liveness

Deciding whether for any transition t, and from any reachable marking, there is another reachable marking that enables t.

- In general, primitive recursive equivalent to reachability, hence decidable.
- General complexity is an open problem.
- PSPACE-complete for 1-safe nets.
- co-NP-complete free-choice nets.
- PTIME for bounded bounded free-choice nets
- PTIME for conflict-free nets

Deadlock-freedom

A net is *deadlock-free* iff every reachable marking enables some transition.

- In general, reduction to reachability in PTIME
- PSPACE-complete for 1-safe nets
- NP-complete 1-safe free-choice nets
- PTIME for conflict-free nets.