Process modelling and analysis with High level Petri nets

Process modelling and analysis with High-Level Petri nets

- Recap first lecture
- High-Level Petri nets
- Running case

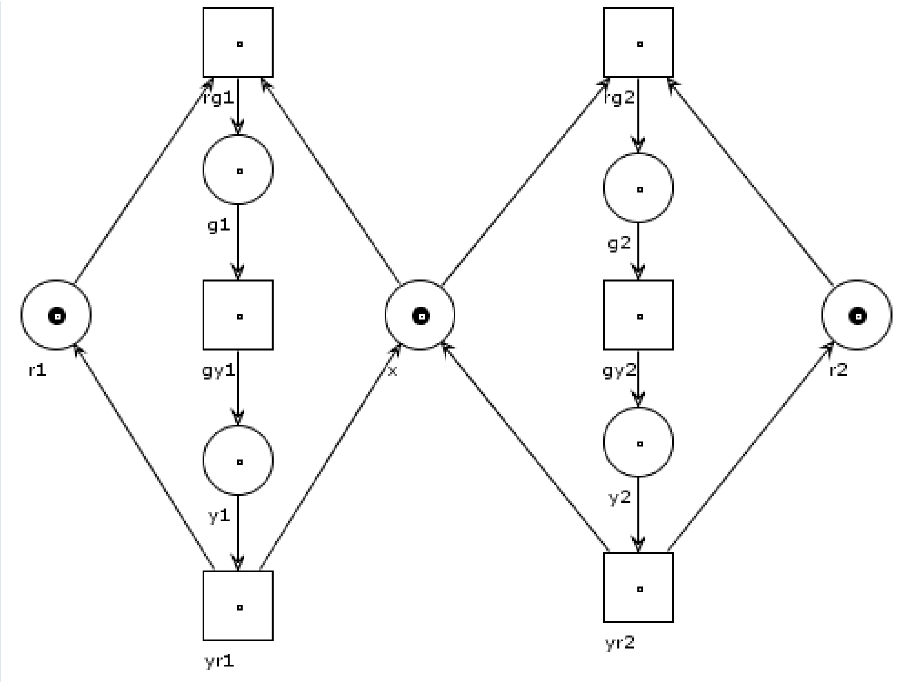
Recap first lecture

- Formal approach process modelling
- Elementary net systems
  - Exercise 1.1.1.
  - Exercise 1.2.2.

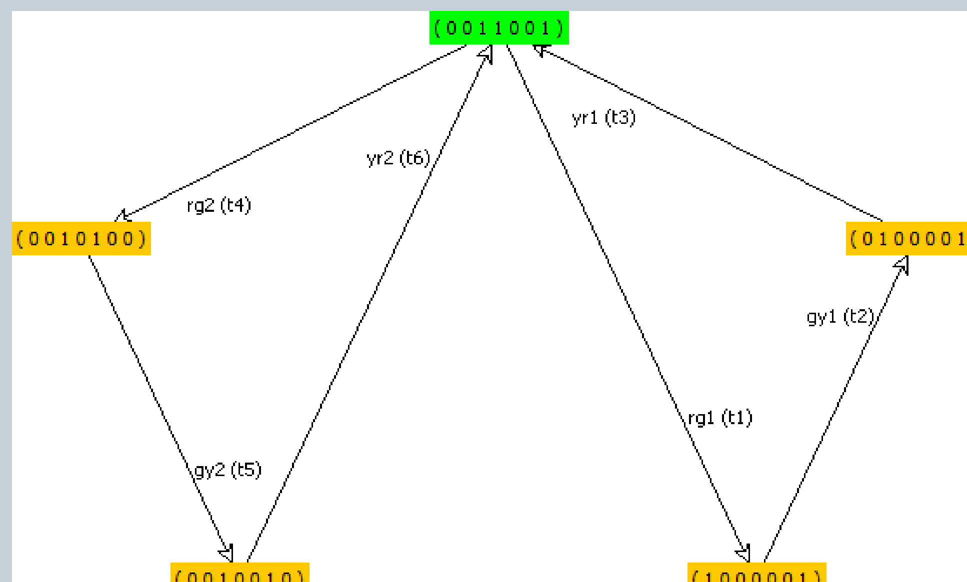
Recap first lecture

- Arguments for formal approach to process modelling:
  - Formal models allow (automated) verification of properties
  - Graphical representation ease validation
  - Formal models can be used as unambiguous blueprints for implementation

Recap first lecture



Generate reachability graph



Requirement :
both lights green
at the same time
is not allowed !

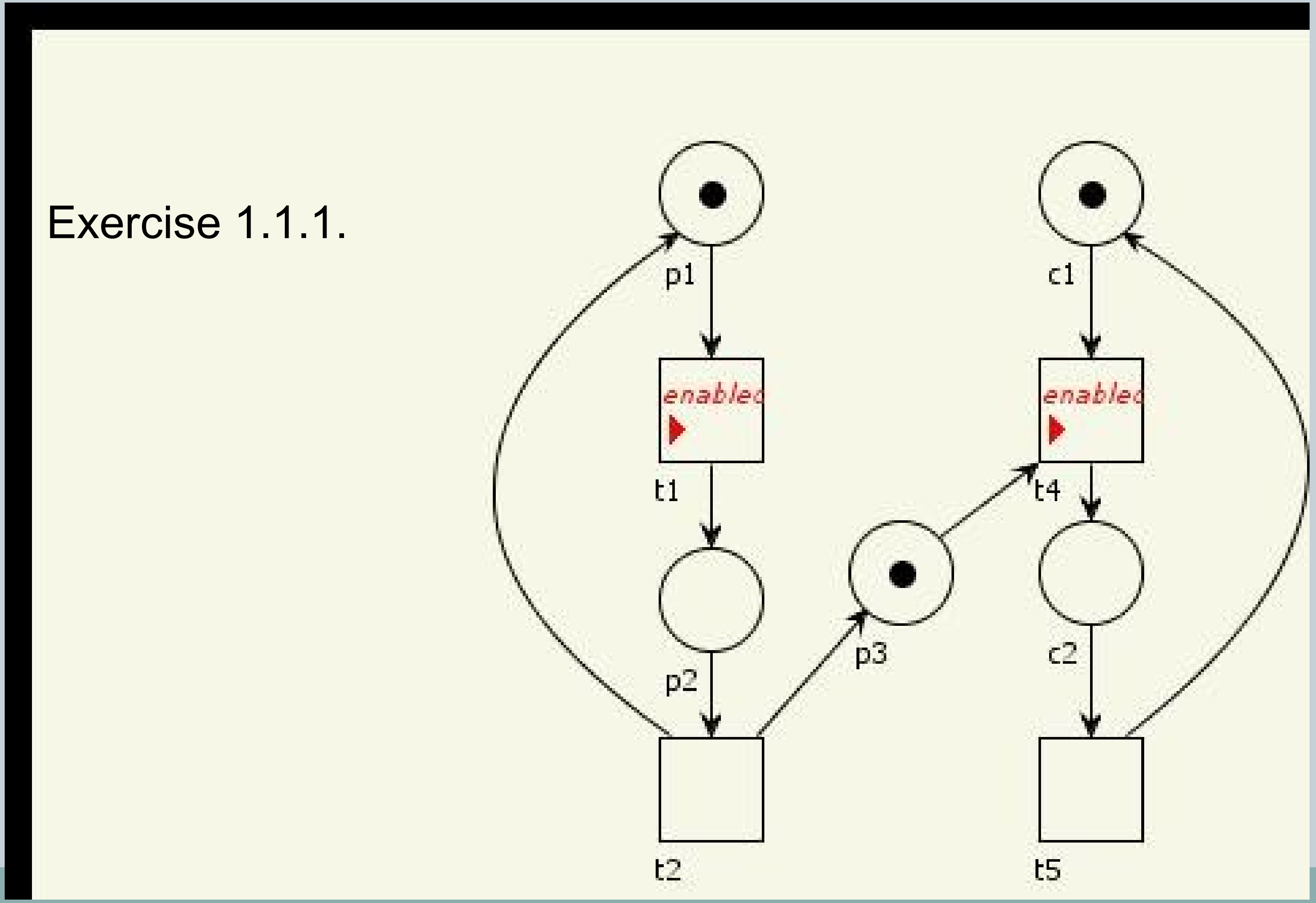
verify

Recap first lecture

- Formal approach process modelling
- Elementary net systems
  - Exercise 1.1.1.
  - Exercise 1.2.2.

Recap first lecture

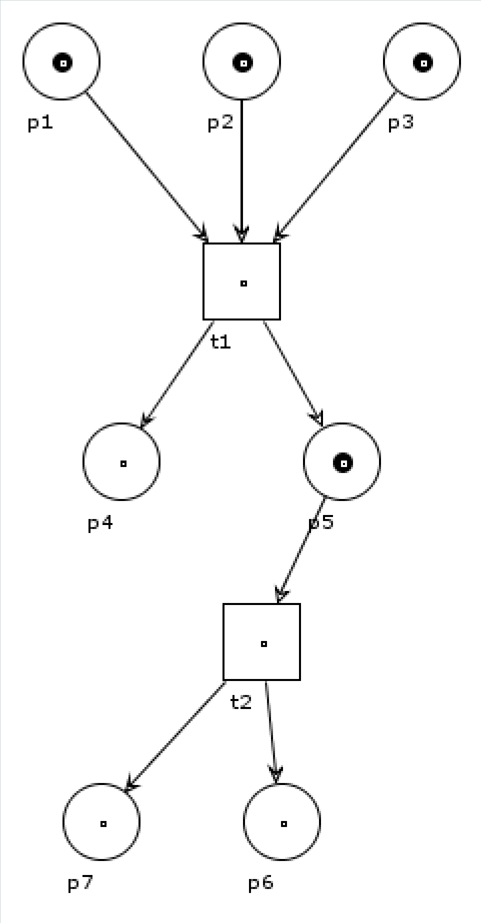
- Is this a contact free net ?



Recap first lecture

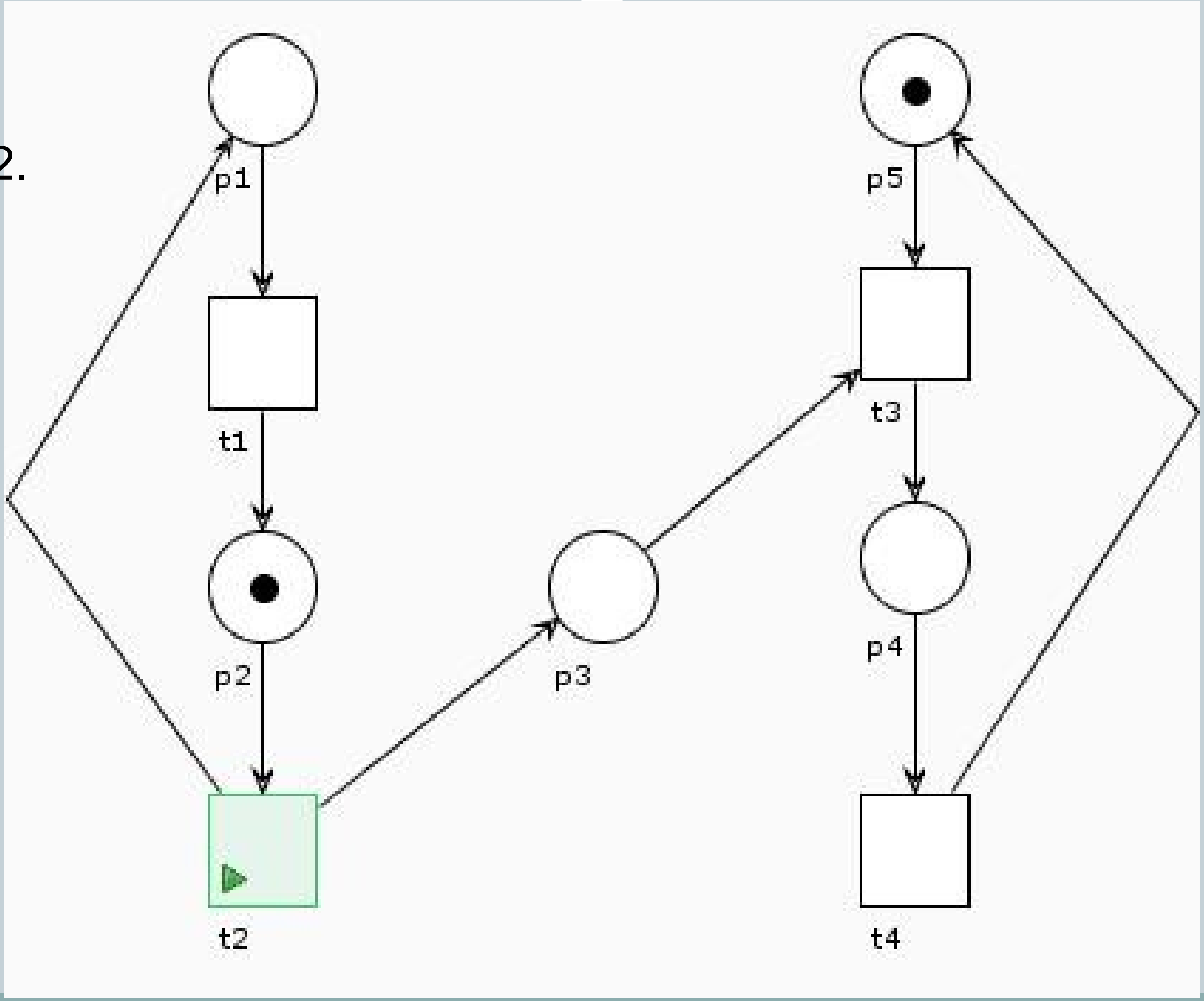
CONTACT :

Exercise 1.1.1.



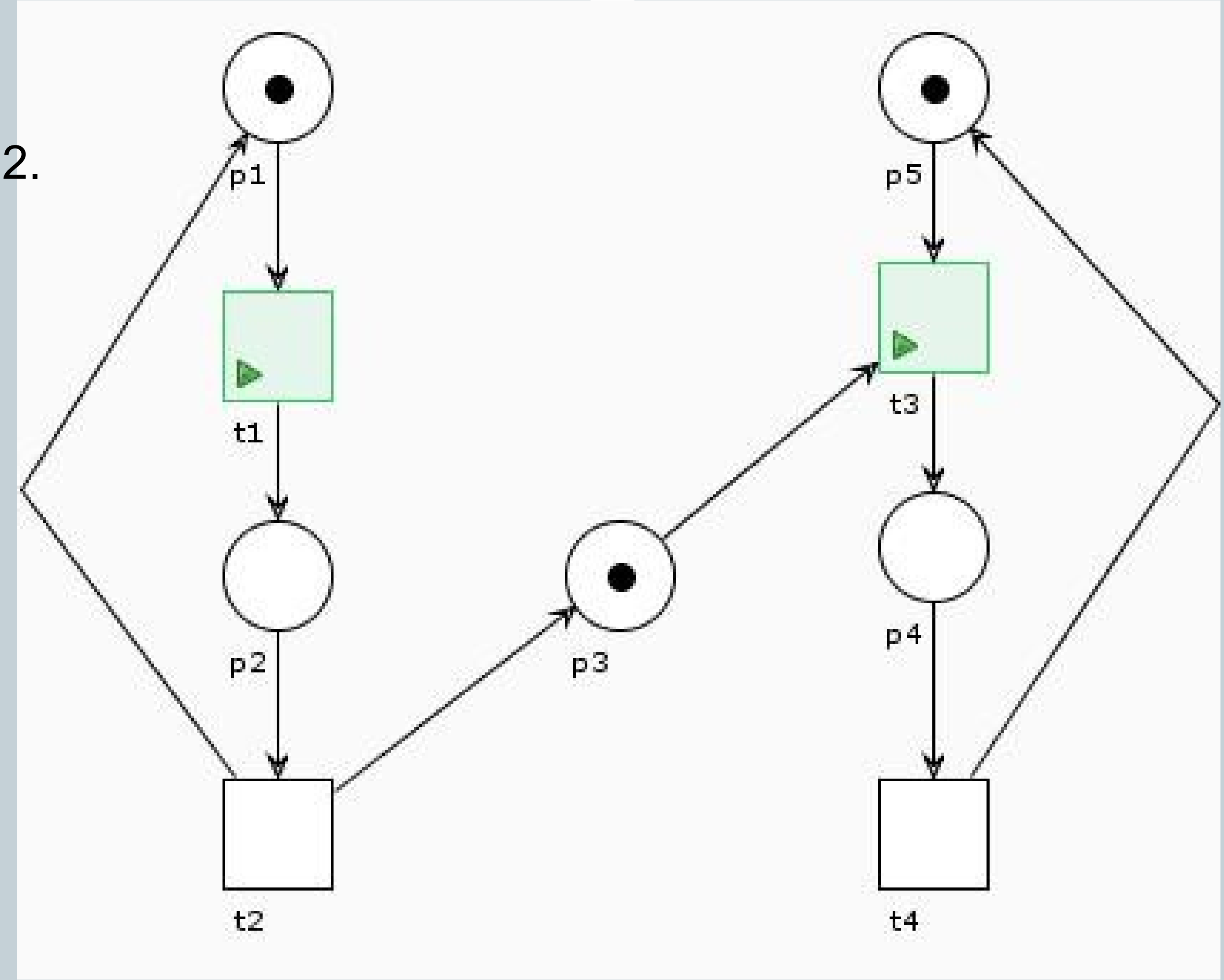
Recap first lecture

Exercise 1.2.2.



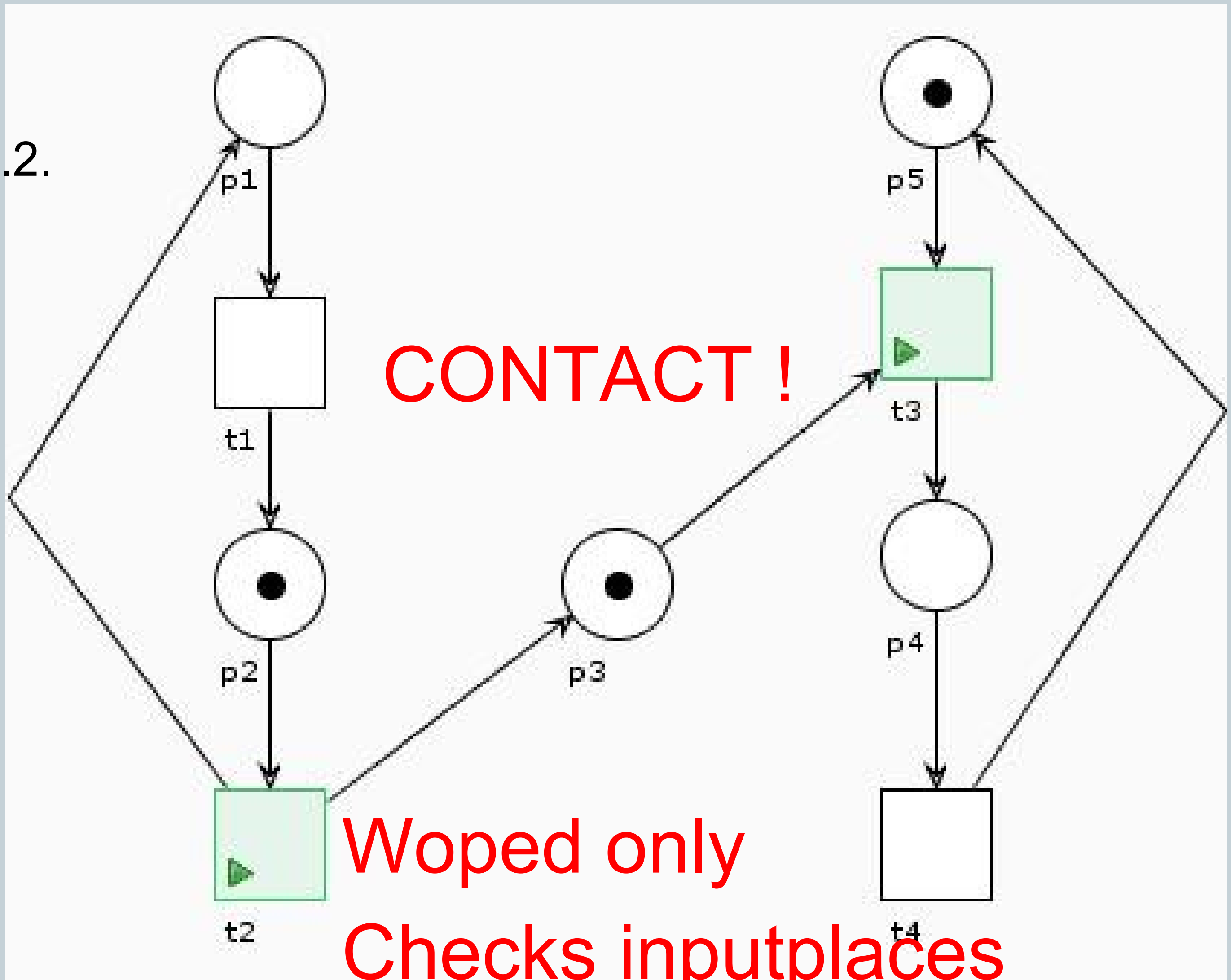
Recap first lecture

Exercise 1.2.2.



Recap first lecture

Exercise 1.2.2.



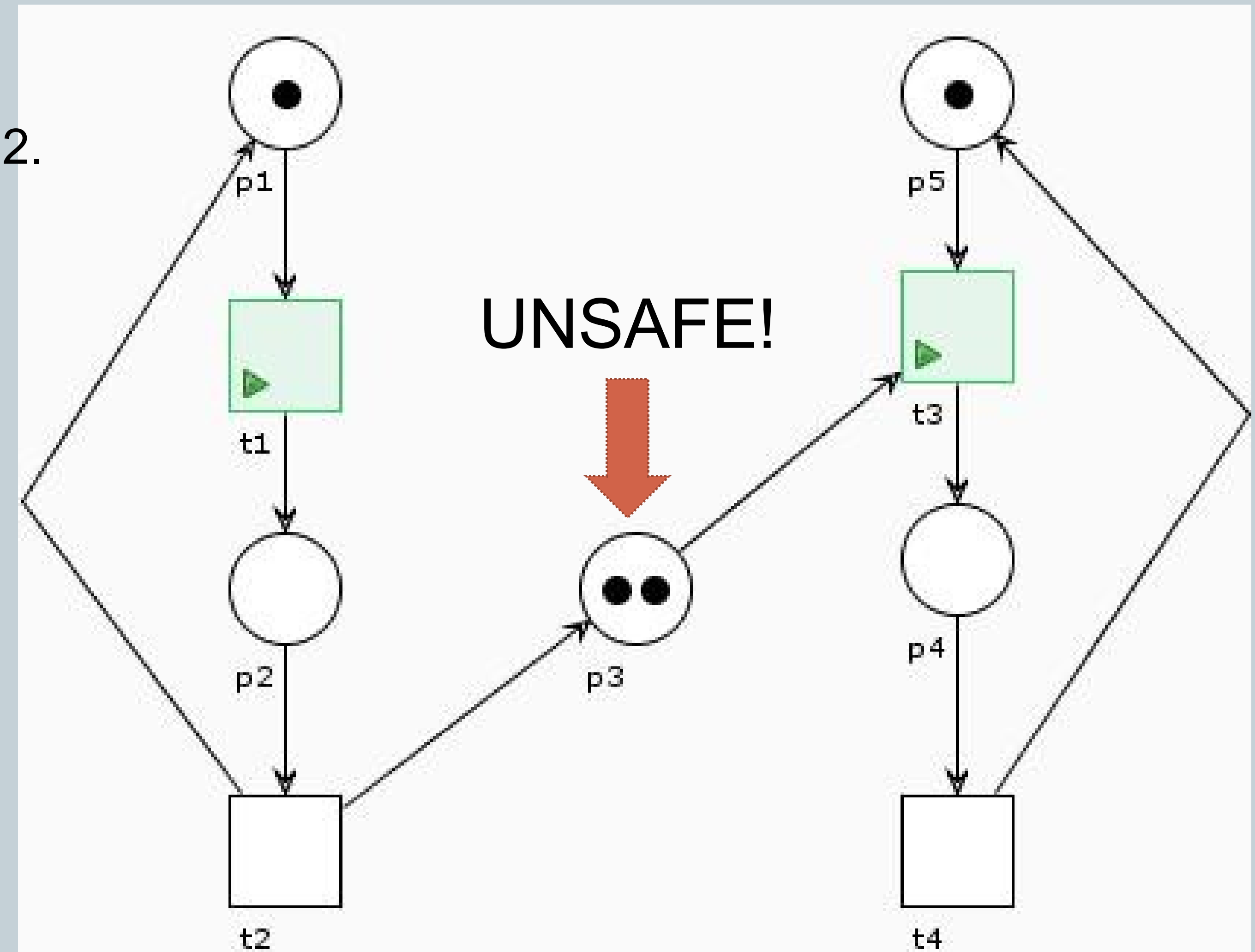
CONTACT !

Woped only

Checks inputplaces

Recap first lecture

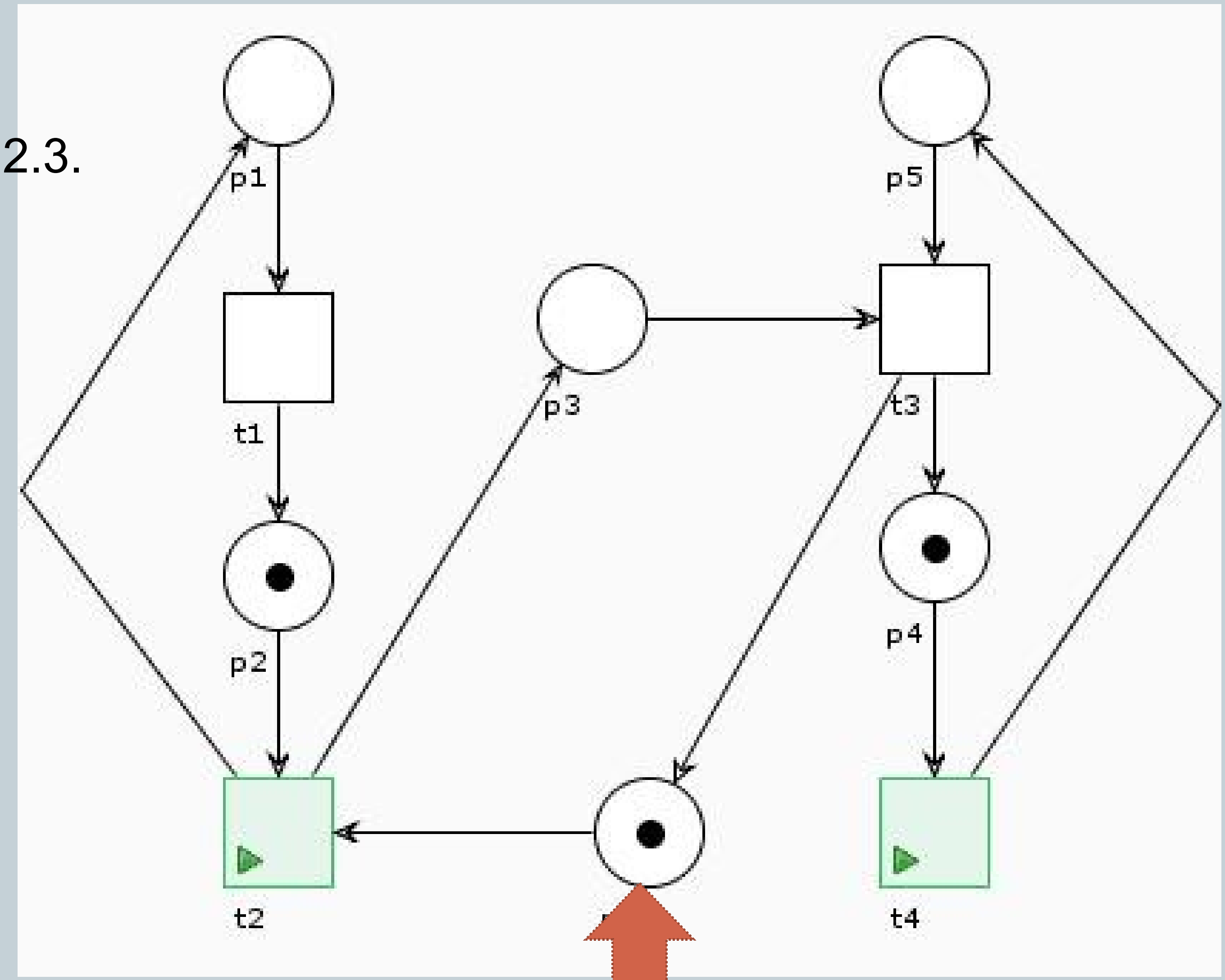
Exercise 1.2.2.



How do we solve this ?

Recap first lecture

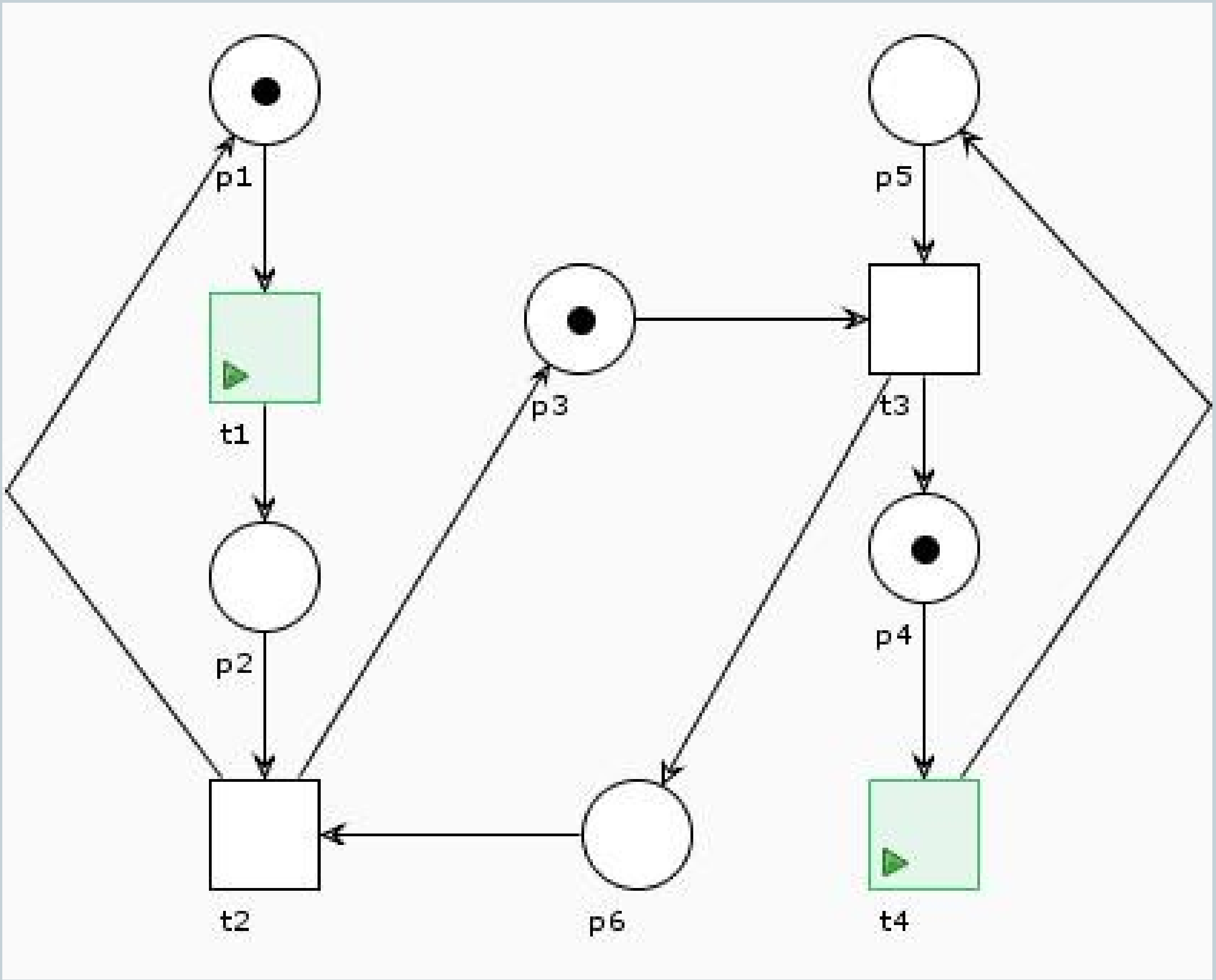
Exercise 1.2.3.



ADD COMPLEMENT PLACE

Recap first lecture

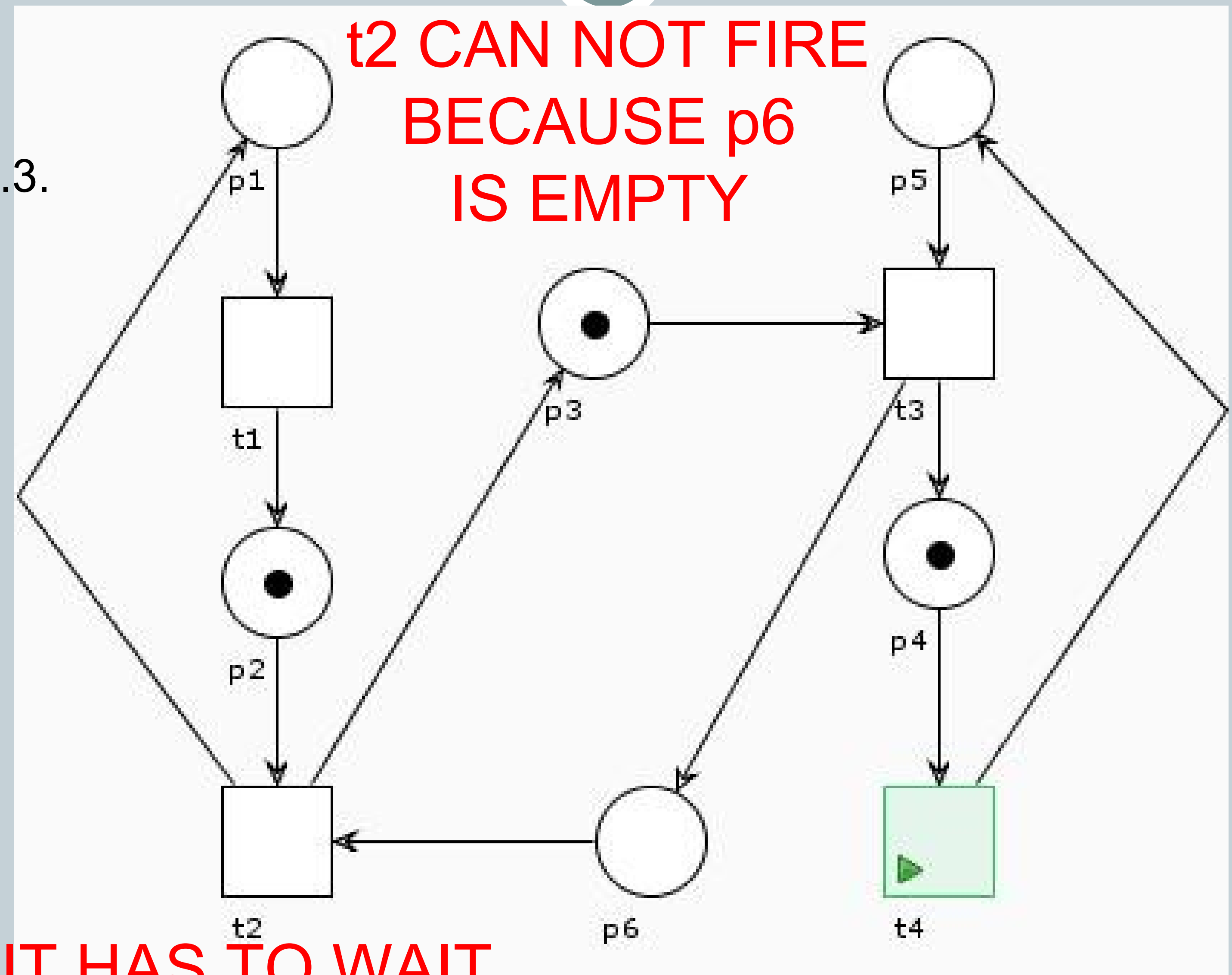
Exercise 1.2.3.



Recap first lecture

Exercise 1.2.3.

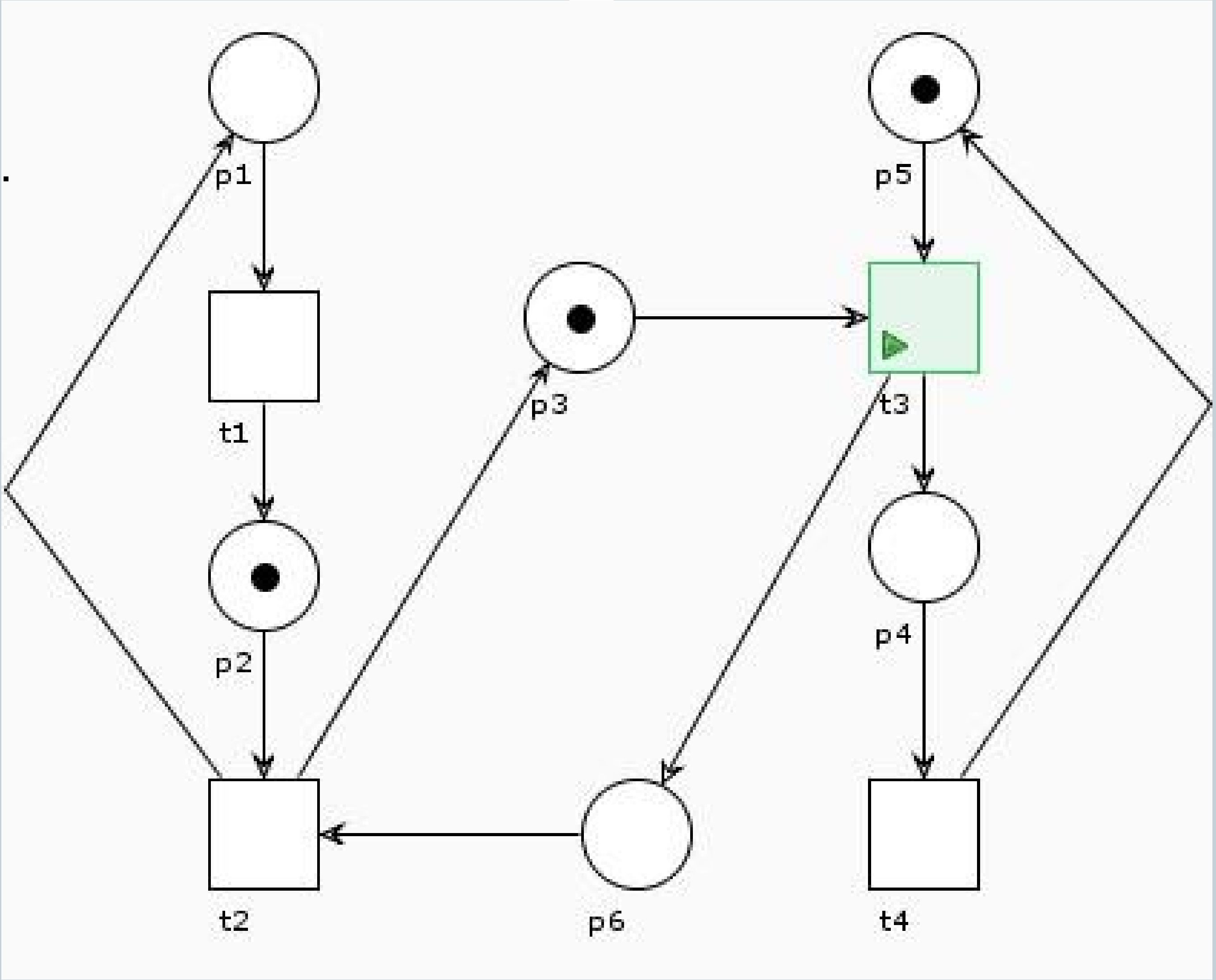
t2 CAN NOT FIRE
BECAUSE p6
IS EMPTY



SO IT HAS TO WAIT
UNTIL t3 HAS FIRED

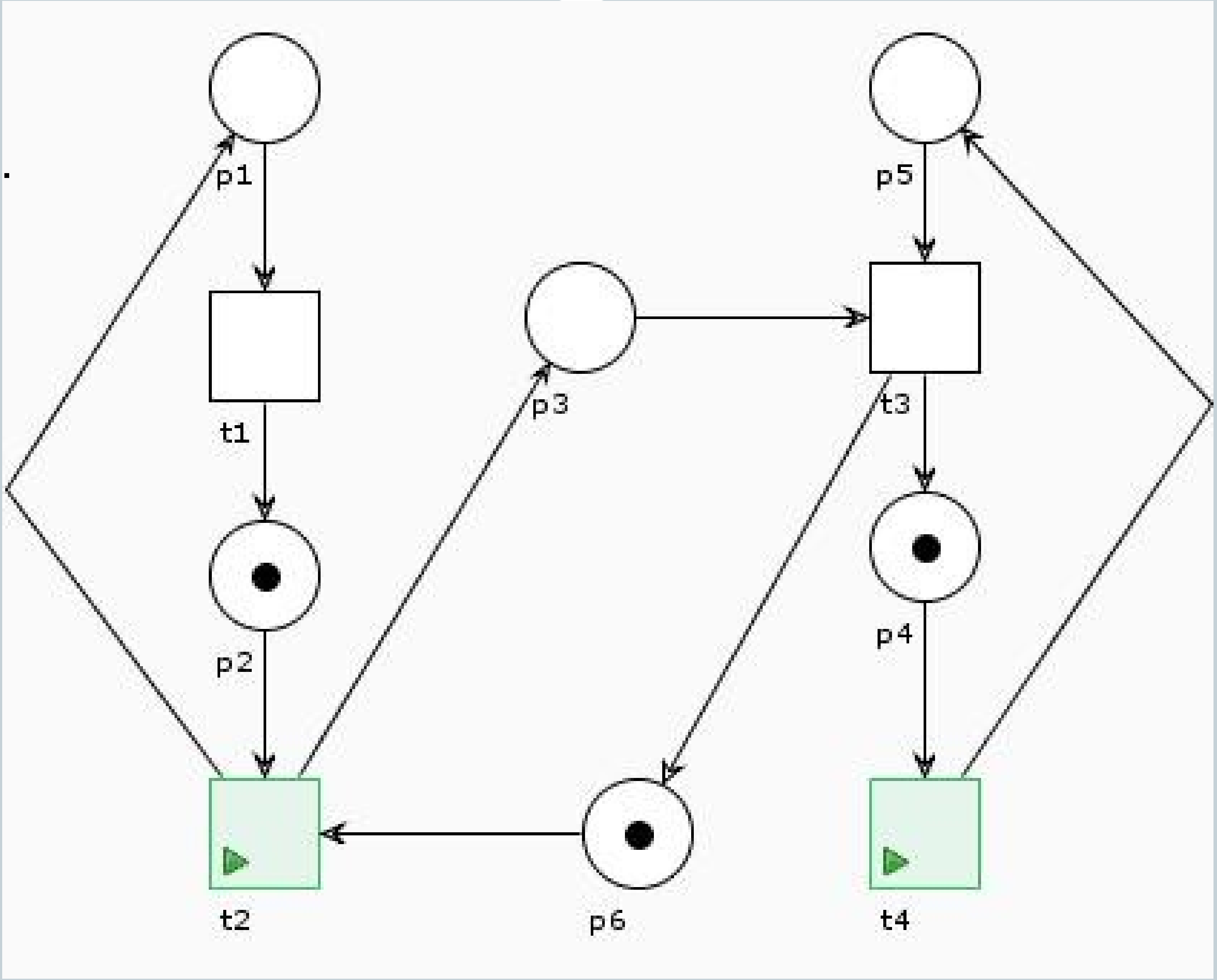
Recap first lecture

Exercise 1.2.3.



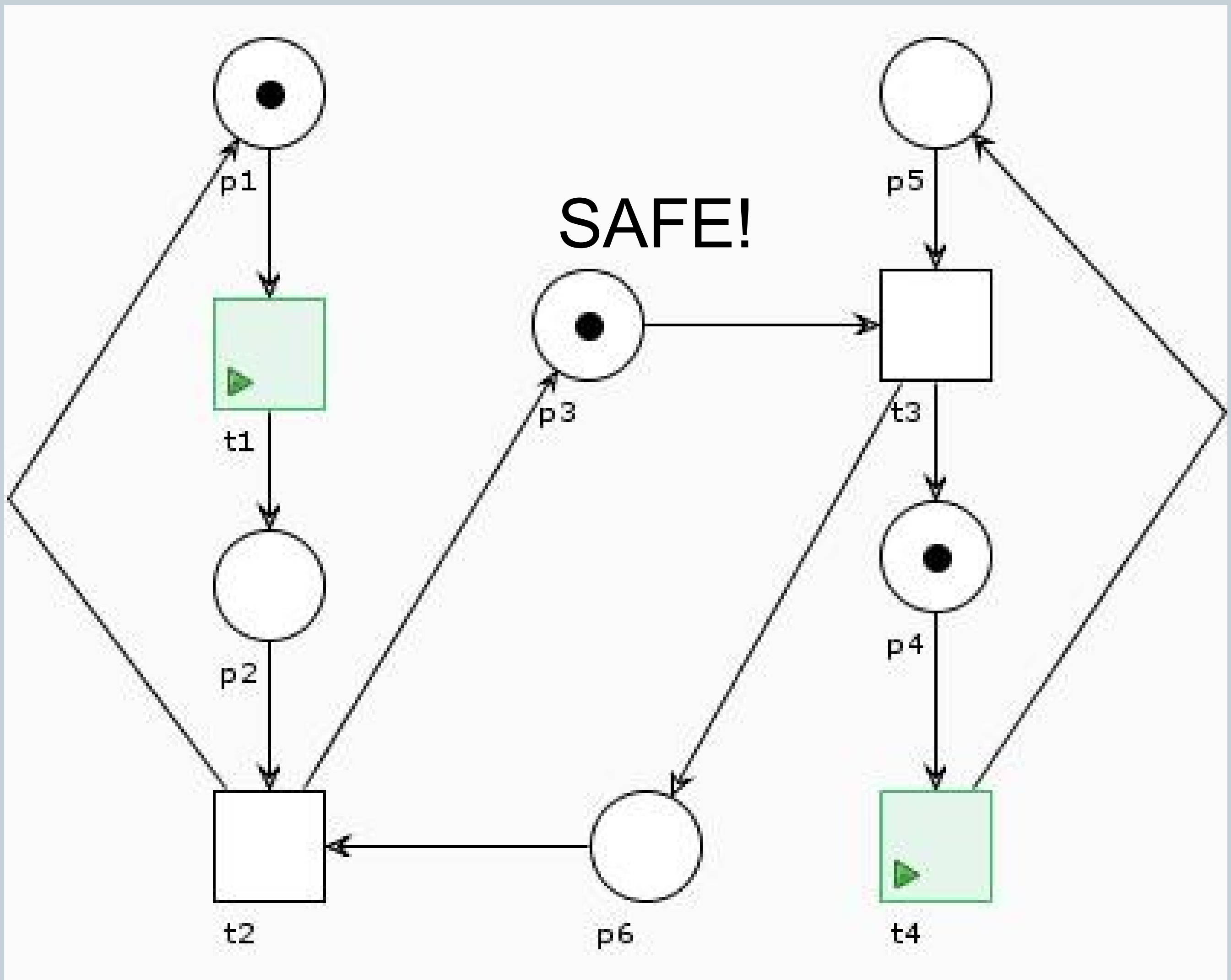
Recap first lecture

Exercise 1.2.3.



Recap first lecture

Exercise 1.2.3.



Recap first lecture

Exercise 1.1.1.b.

All input places … **Have a token**

1. A transition t is enabled in a marking M iff the following property is satisfied :

$$\forall p \in In(t) \quad M(p) = true \quad \wedge$$
$$\forall p \in Out(t) \quad M(p) = false.$$

All output places **Have no token**

Which condition is redundant in a contact-free EN-system ?

Recap first lecture

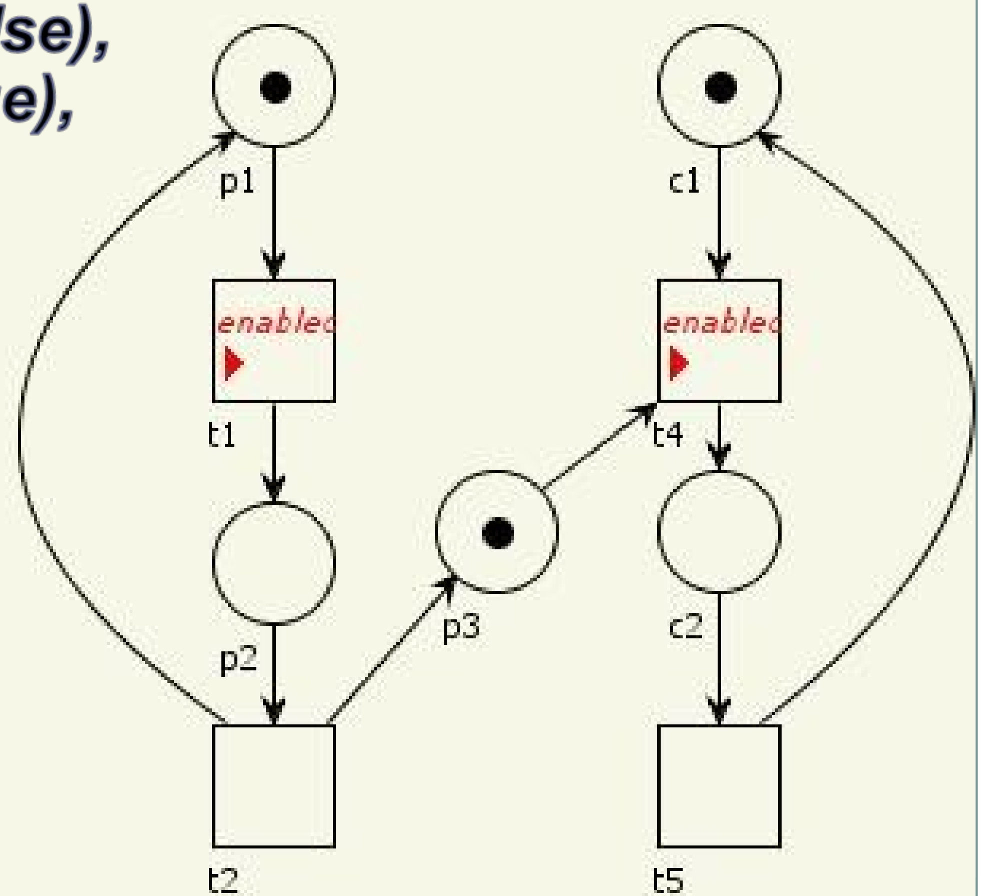
- The firing rule : formal notation for Marking

Exercise 1.1.1.b.

Marking M=
{(p1,true),(p2,false),
(p3,true),(c1,true),
(c2,false)}

Or

M(p1)=true,
M(p2)=false
Etc.



Recap first lecture

Exercise 1.1.2.

- Woped implemented the firing rule of a PT-system (no check on output places)
- Contact-free EN-systems have the same firing rule as and are equivalent to (safe) PT-systems
- So we can build EN-systems with Woped but only if they are contact-free (safe) !!

- Recap first lecture
- High-Level Petri nets
- Running case

High level Petri nets

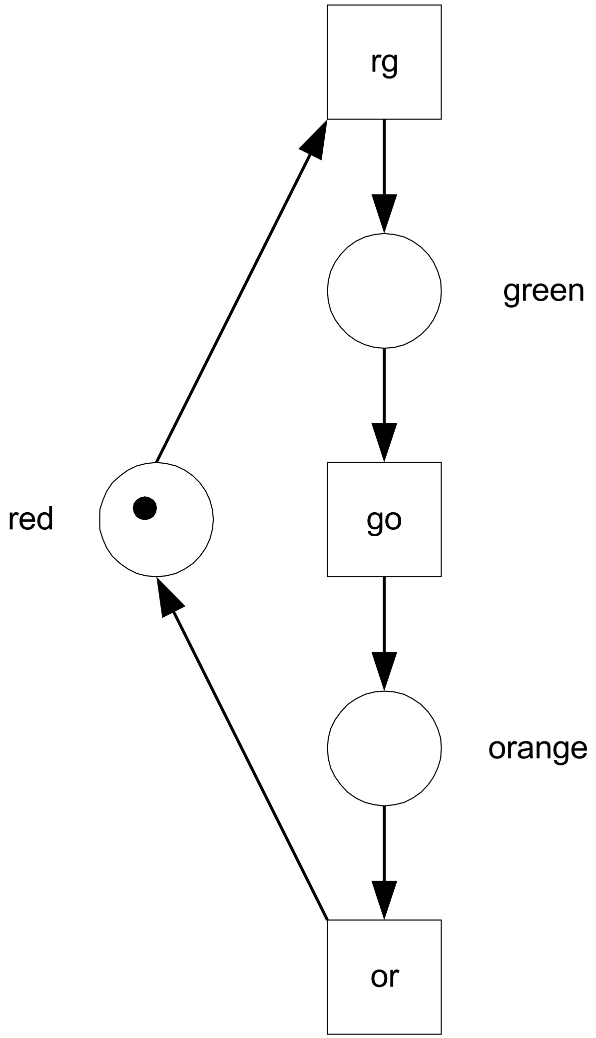
- High level Petri nets
  - PT-systems
  - WF-nets
  - Coloured Petri nets
  - Timed Petri nets
  - Hierarchical Petri nets

PT-systems

- PT-systems
  - Modelling with PT-systems
  - Comparing EN- and PT-systems
  - Analysis of PT-systems

Modelling with PT-systems

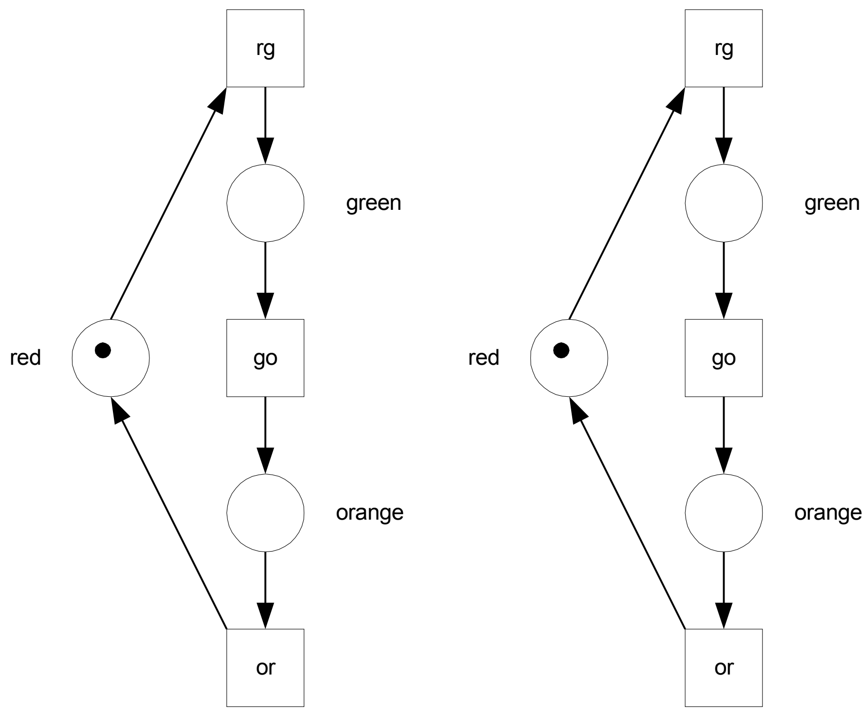
Example : Single traffic light



Modelling with PT-systems

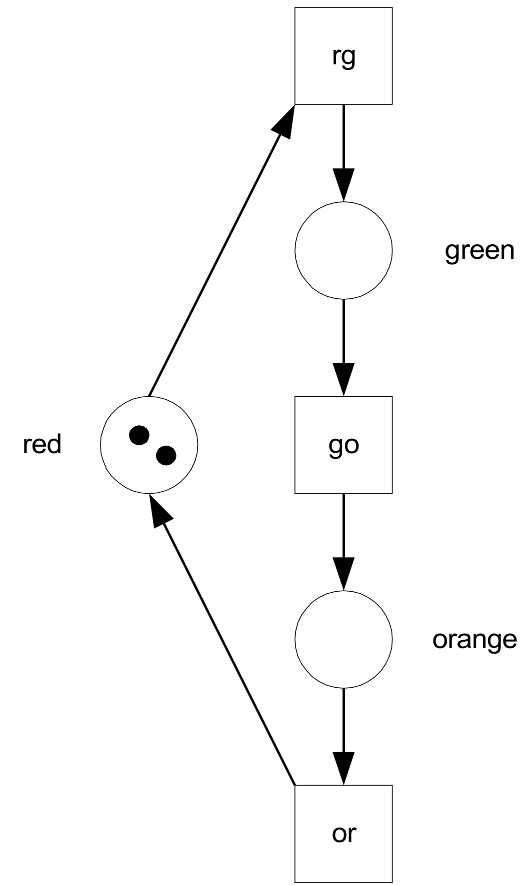
Example : two traffic lights

2 EN systems



OR

1 single PT system

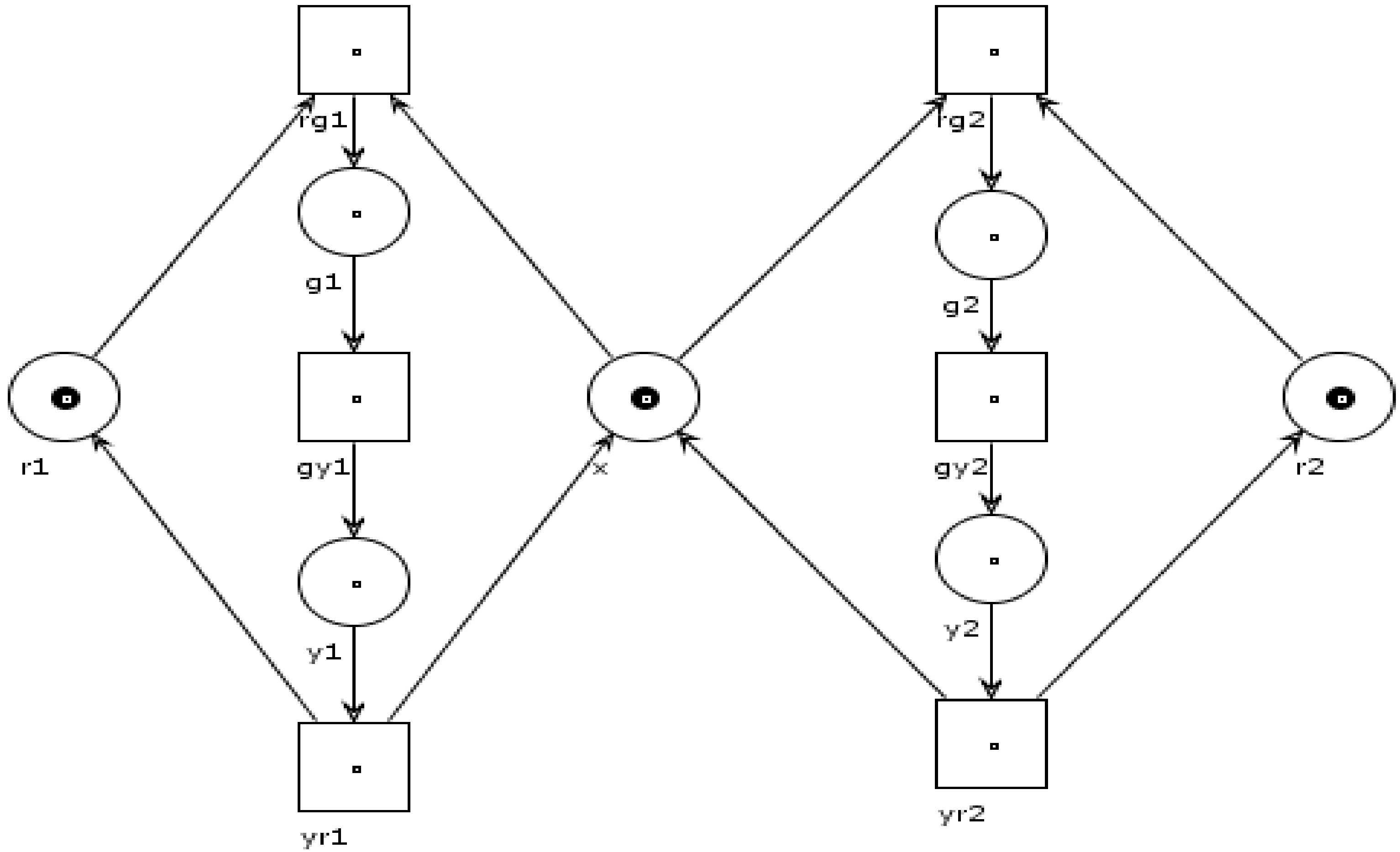


Requirement :

Both lights green at the same time not allowed !

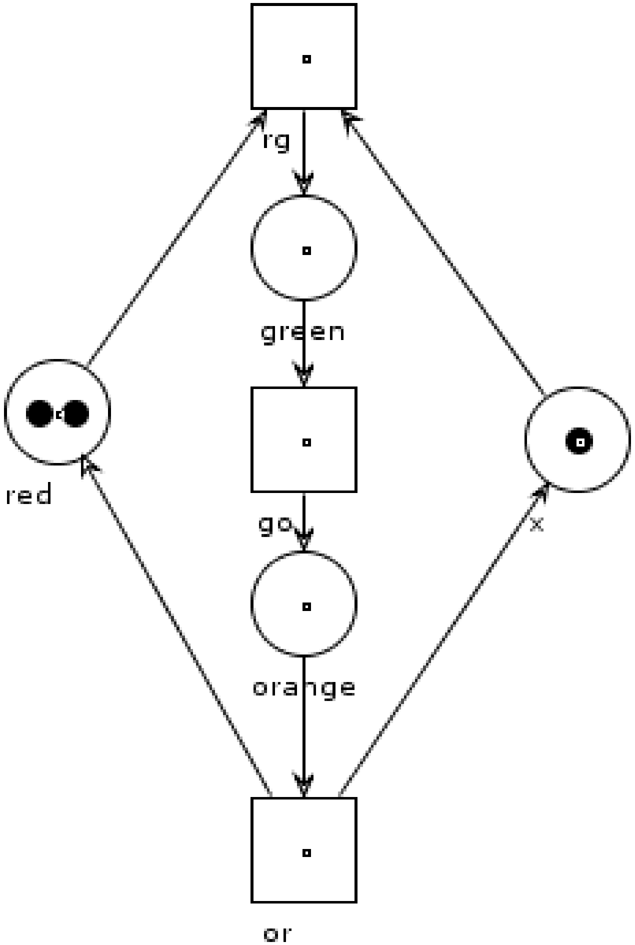
Modelling with PT-systems

This was the EN-system solution :

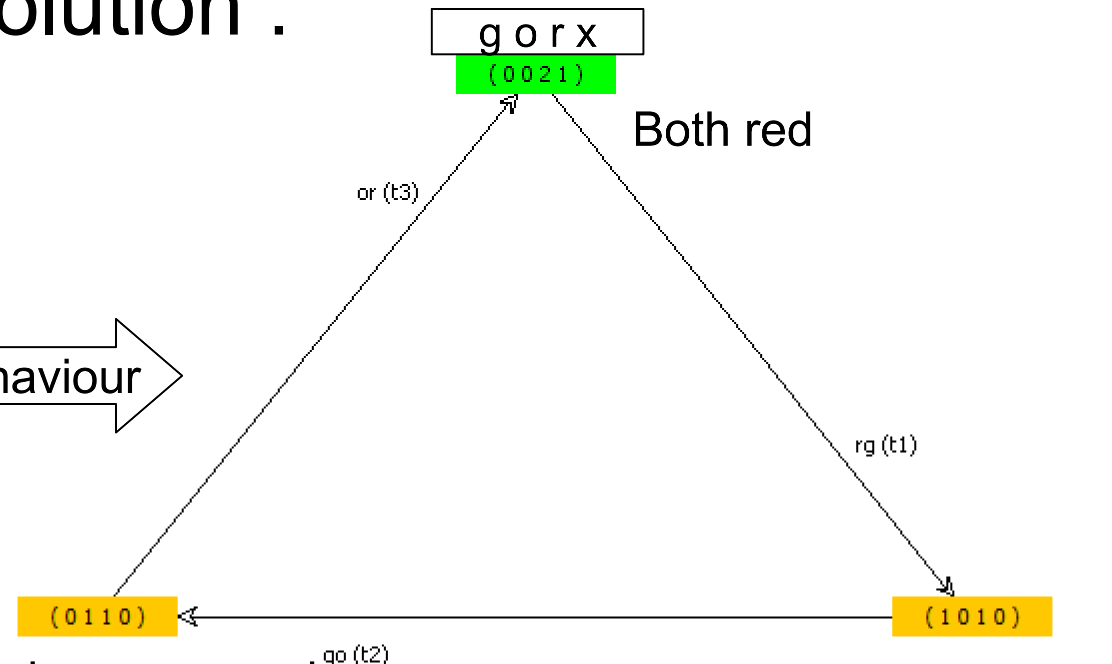


Modelling with PT-systems

This is the PT-system solution :



Behaviour



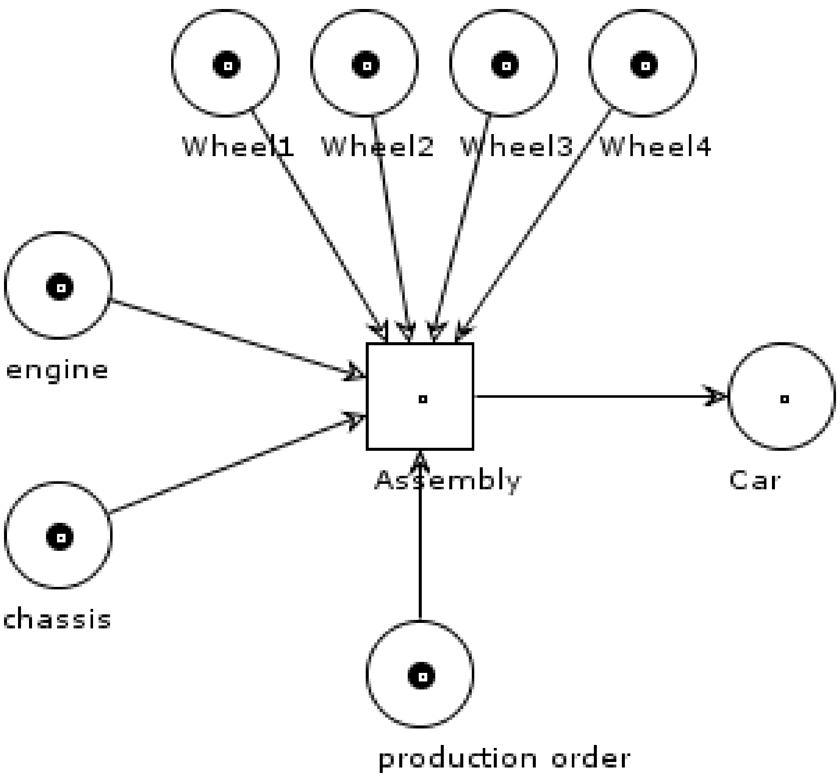
So you can verify that the model also behaves correctly.

The model is more compact, but you don't know which traffic light has which colour!

Modelling with PT-systems

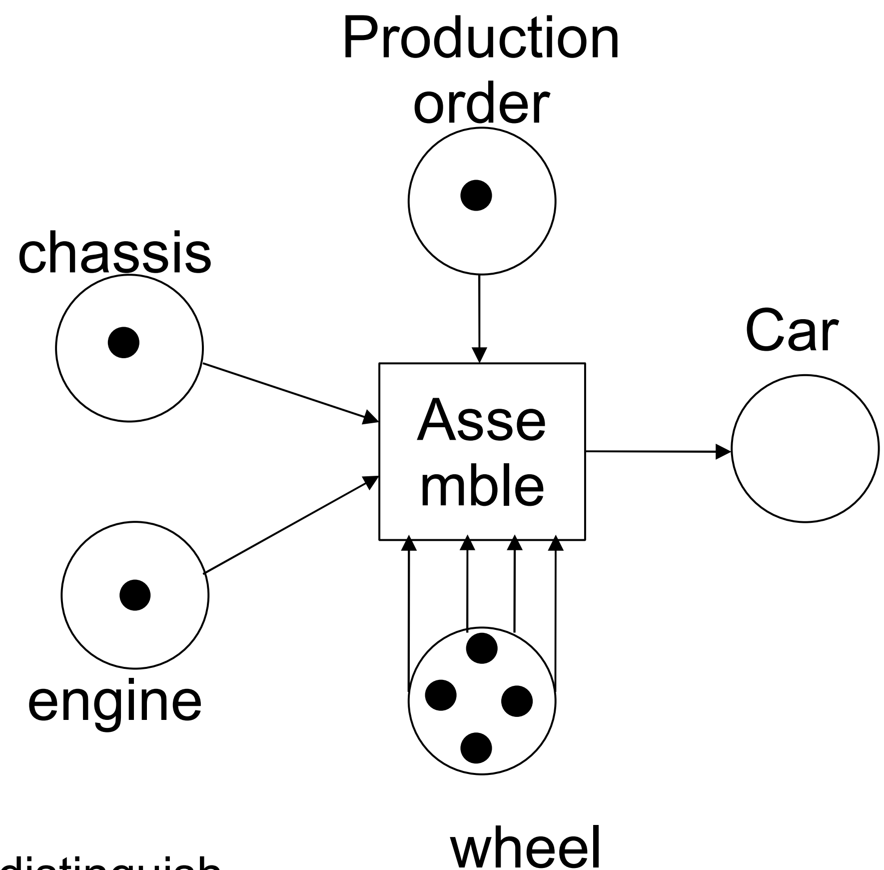
Example : Car assembly

EN system



OR

PT system



The PT system is more compact, but you can not distinguish between the 4 wheels on one car (which you might not want) !

PT-systems

- PT-systems
  - Modelling with PT-systems
  - Comparing EN- and PT-systems
  - Analysis of PT-systems

Comparing EN systems and PT systems

- Systematic comparison :
  - Elements
  - Structure
  - Dynamics
  - Behaviour

Comparing EN systems and PT systems

- EN systems
  - **Elements** : places, transitions, arcs
- PT systems
  - **Elements** : idem

Comparing EN systems and PT systems

- Systematic comparison :
  - Elements
  - Structure
  - Dynamics
  - Behaviour

Comparing EN systems and PT systems

- EN systems
  - **Structure :**
    - One place can have ***zero*** or ***one*** token
    - Between a place and a transition there are ***zero*** or ***one*** arcs

- PT systems
  - **Structure :**
    - One place can have ***multiple*** tokens
    - Between a place and a transition ***multiple*** arcs are possible

(Note: this is not true for the "Classical Petri nets" in the book of van der Aalst)

Comparing EN systems and PT systems

- Systematic comparison :
  - Elements
  - Structure
  - Dynamics
  - Behaviour

Comparing EN systems and PT systems

- EN systems
  - **Dynamics :**
    - A transition is enabled if:
      - each input place has ***one*** token
      - (each outputplace is empty)

- PT systems
  - **Dynamics :**
    - A transition is enabled if:
      - Each input place has ***enough*** tokens (i.e. One for each arc)
      - output places need not be empty

Comparing EN systems and PT systems

- EN systems
  - **Dynamics :**
    - When a transition has <u>fired </u>:
      - **One token** from each input place is **removed**
      - **One token inserted** in each outputplace

- PT systems
  - **Dynamics :**
    - When a transition has <u>fired:</u>
      - **One token *per arc*** from each inputplace is **removed**
      - **One token per arc inserted** in each outputplace

Comparing EN systems and PT systems

- Systematic comparison :
  - Elements
  - Structure
  - Dynamics
  - Behaviour

Comparing EN systems and PT systems

- EN systems
  - **Behaviour**
    - Reachability graph is finite
- PT systems
  - **Behaviour**
    - Reachability graph can be infinite

PT-systems

- PT-systems
  - Modelling with PT-systems
  - Comparing EN- and PT-systems
  - Analysis of PT-systems

Analysis of PT systems

- Analysis of PT-systems
  - Qualitative analysis
    - General properties of PT Systems
    - State space analysis of PT Systems
  - Quantitative analysis (next lecture)

Analysis of PT systems

- General properties of PT-systems
  - Reachability
  - Liveness
  - Boundedness
  - Safeness
  - (Fairness)

Analysis of PT systems

- Reachability :
  - a state M* is reachable from a state M if there is a path in the reachability graph between M and M*.
- Liveness :
  - a transition t is live if from each reachable state M a state M* can be reached where t is enabled
  - a petri net is live if all its transitions are live
  - A Petri Net with a given marking is in deadlock iff no transition is enabled in that marking.
- Boundedness :
  - a Petri net is n-bounded if the number of tokens in each place never exceeds some number n (safe if n=1)

Analysis of PT systems

- Analysis of PT-systems
  - Qualitative analysis
    - General properties of PT Systems
    - State space analysis of PT Systems
  - Quantitative analysis (next lecture)

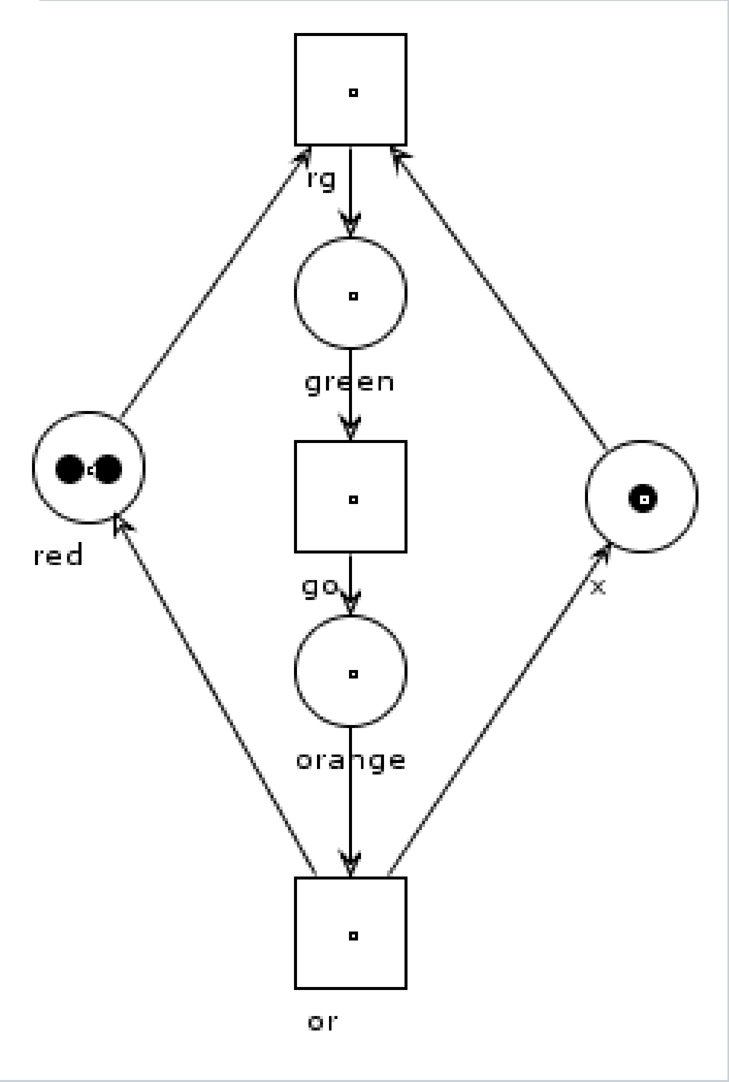
Analysis of PT systems

- State space analysis of PT-systems
  - Calculate state space
  - Specify required properties
  - Verify state space for presence/absence of properties

Analysis of PT systems

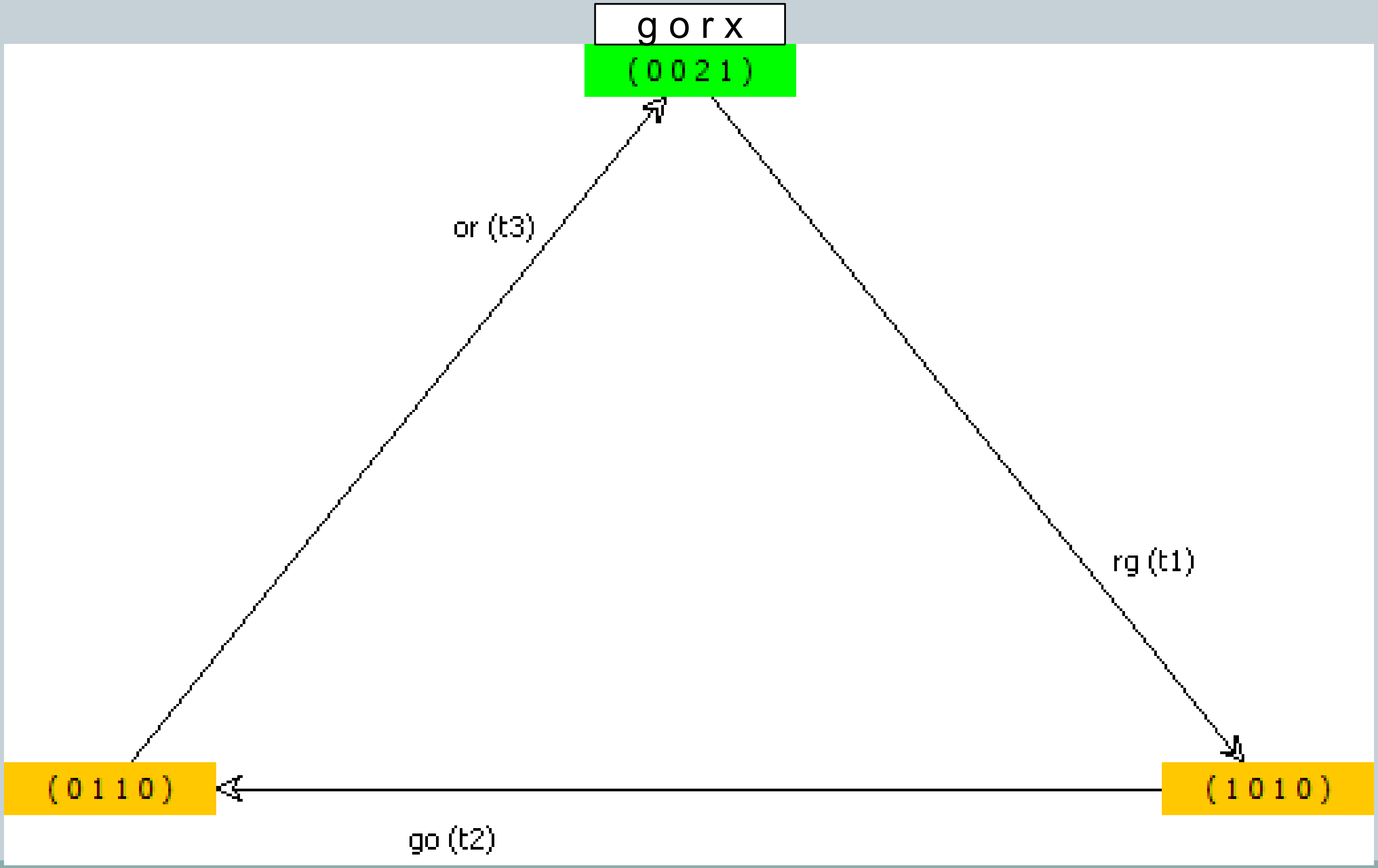
- Calculate state space :

  Traffic light example



Analysis of PT systems

- State space of traffic light example :



Analysis of PT systems

- An algorithm for calculating the state space:
  - Given:
    - V is set of nodes in the graph
    - E is the set of edges between the nodes

Analysis of PT systems

- Algorithm:
  - Initial marking is M, M is untagged
    - $V = \{M_1\}$, $E = \emptyset$
  - While there are untagged nodes in V do :
    - Select an untagged node $M \in V$ and tag it
    - For each enabled transition, t, at M do :
      - Compute M^*= state after firing t
      - $V = V \cup \{M^*\}$
      - $E = E \cup \{(M, t, M^*)\}$

Analysis of PT systems

- The algorithm does the following:
  1) Let V be the set containing just the initial state M_1 and E the empty set (so you start with an empty reachability graph)
  2) Take an untagged element M from V and tag it (to remember that you already processed it).
  3) Calculate all states reachable for M by firing all enabled transitions t, giving (M,t,M*).
  4) Each successor state M* that is not already in V is added to V, and the edge (M,t,M*) in the reachability graph is added to E.
  5) If V has no more untagged elements stop, otherwise goto 2.

Analysis of PT systems

- State space analysis of PT-systems
  - Calculate state space
  - Specify required properties
  - Verify state space for presence/absence of properties

Analysis of PT systems

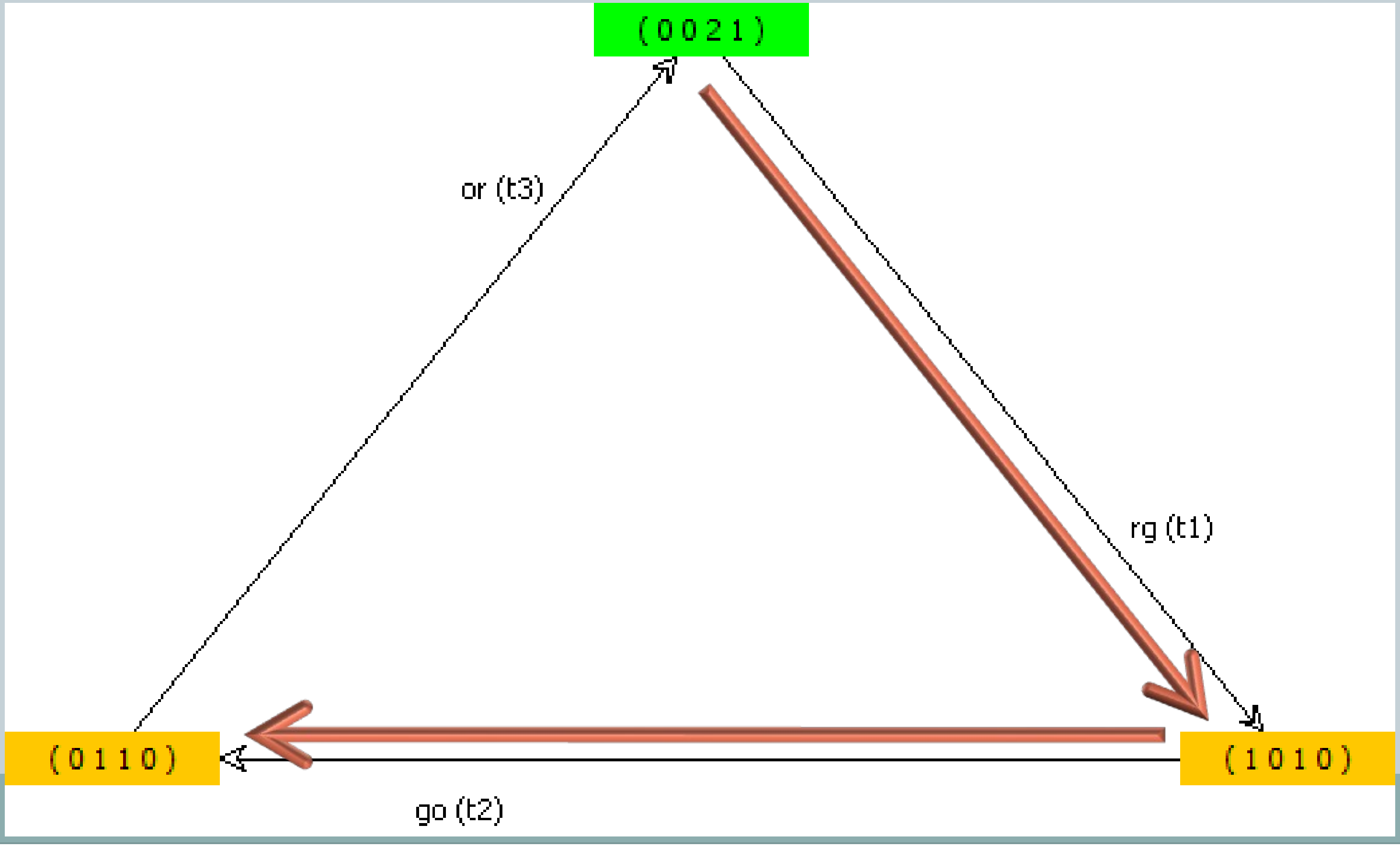
- State space analysis of PT-systems
  - Calculate state space
  - Specify required properties
    - Reachability
    - Liveness
    - Etc.
  - Verify RG for presence/absence of properties

Analysis of PT systems

- State space analysis of PT-systems
  - Calculate state space
  - Specify required properties
    - Reachability
    - Liveness
    - Etc.
  - Verify RG for presence/absence of properties

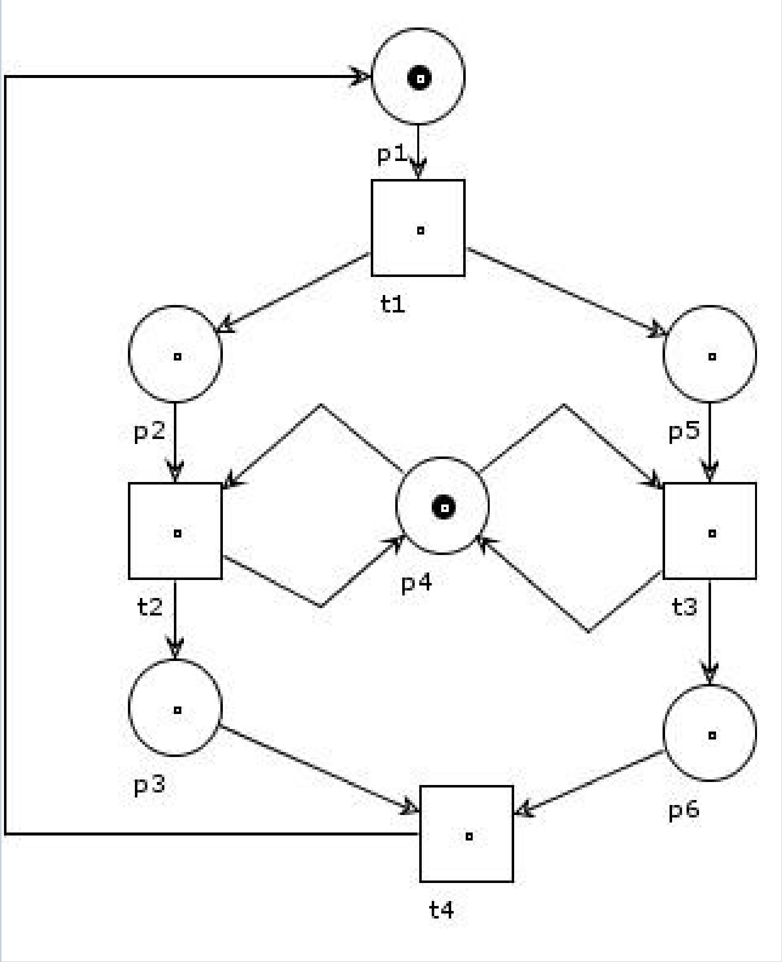
Example : Verifying reachability

Can one light be orange while the other one is red? → Is there a path in the RG from the initial marking (0 0 2 1) to the marking (0 1 1 0)



Analysis of PT systems

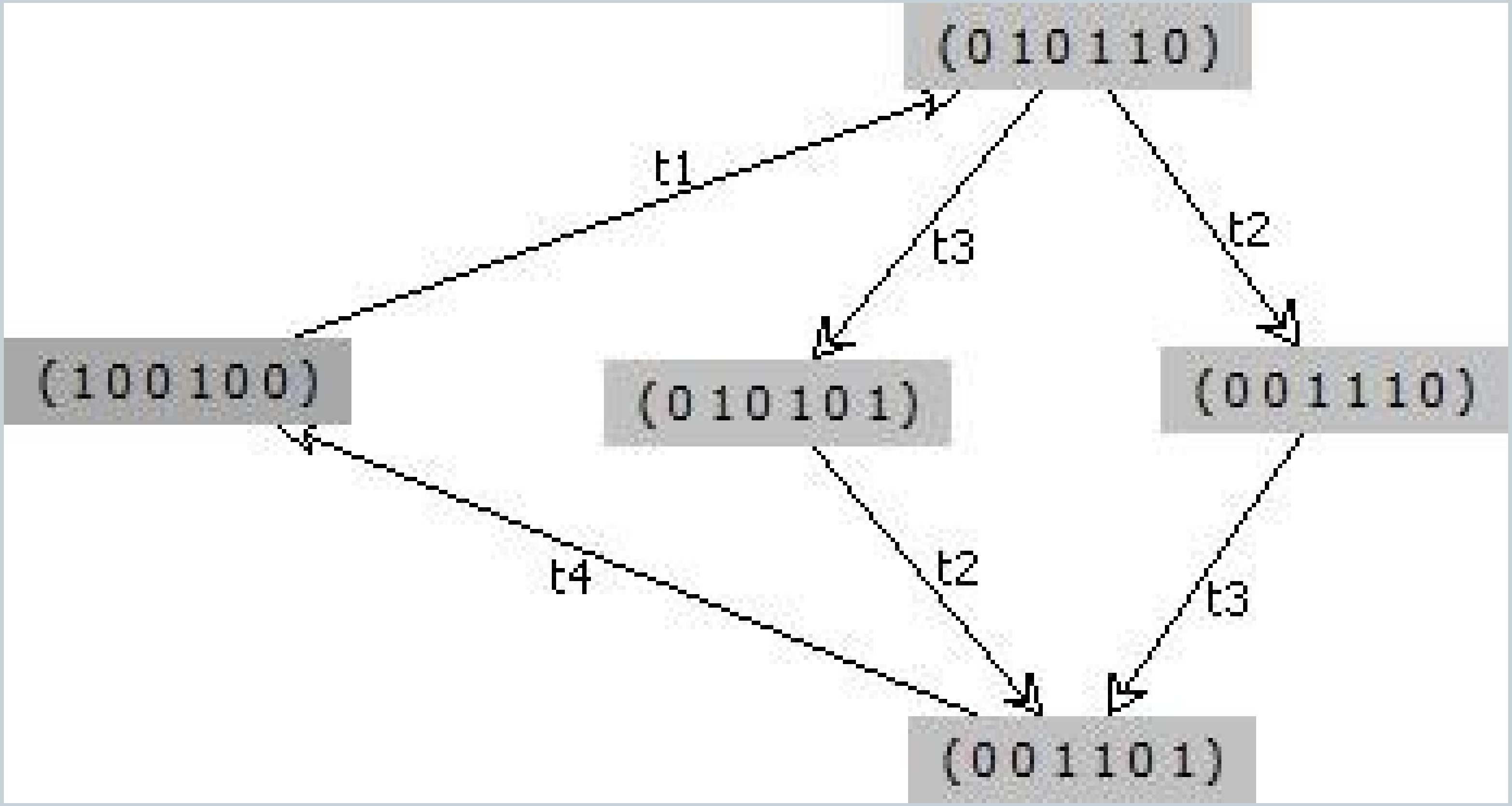
Example : Verifying Liveness



Analysis of PT systems

Example : Verifying Liveness

Is the Petri net system live? → **Is there not one single marking in the RG which has no outgoing arcs?**



Analysis of PT systems

- There are algorithms, based on the reachability graph, to decide
  - boundedness of a PT-system (Karp-Miller)
  - liveness for a bounded PT-system
  - reachability for a bounded PT-system (Lipton)

- However, the size of the reachability graph can be exponential in relation to the size of the PT-system (the "**state space explosion problem**")

- So therefore this approach might become impractical

Analysis of PT systems

- One way to address the problem of state space explosion is to put restrictions on the structure of the net, i.e. to make it more simple and its behaviour easier to analyse

- We will look at a type of PT-systems called Work Flow-nets (WF-nets) which are specifically taylored to model Workflows

High level Petri nets

- High level Petri nets
  - PT-systems
  - WF-nets
  - Coloured Petri nets
  - Timed Petri nets
  - Hierarchical Petri nets

WF-nets

- WF-nets
  - Definition of a WF-net
  - Analysis of WF-nets

Definition of a WF-net

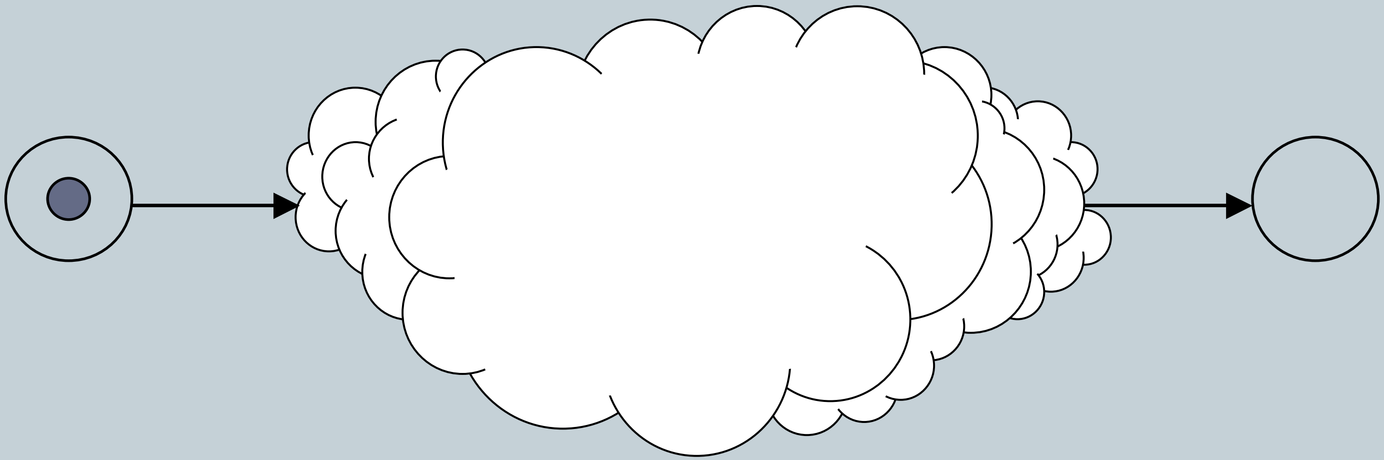
- A workflow-net is a kind of PT-system taylored to model the control-flow dimension of Workflows (of a single case)

- A Workflow is a case-based business Process:
  - Handling of a Customer order
  - Handling of an Insurance claim
  - Handling of a Mortgage request

- Mass assembly of bicycles is not a Workflow process, but production of bicycles on order is

Definition of a WF-net

- A WF-net is a PT-system with:
  - One start condition
  - One end condition
  - Each transition (task) is on a path from the start condition to the end condition



WF-nets

- WF-nets
  - Definition of a WF-net
  - Analysis of WF-nets

Analysis of WF-nets

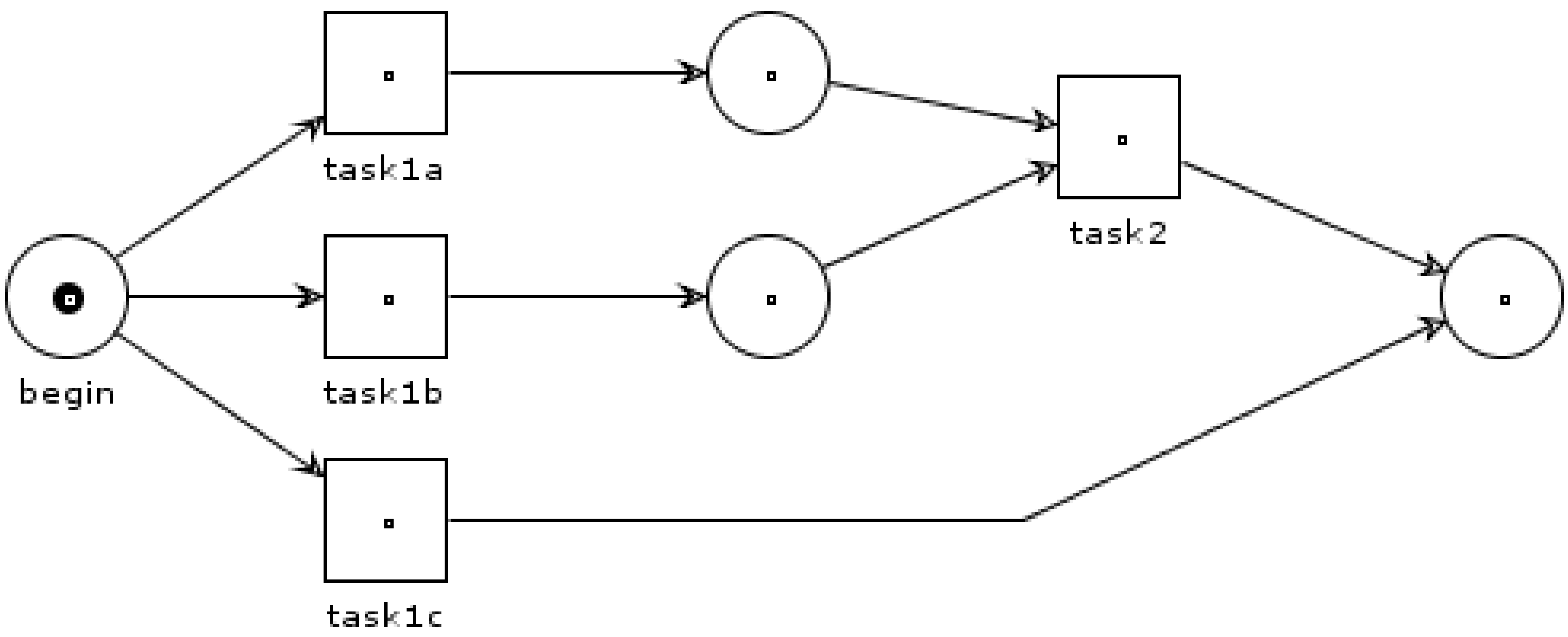
- Analysis of WF-nets
  - Qualitative analysis
    - General properties of WF-nets
    - State space analysis of WF-nets
  - Quantitative analysis (next lecture)

Analysis of WF-nets

- A general property of WF-nets : soundness

- Soundness is a minimum quality requirement for WF nets, implying :
  - The option to complete
  - “Proper completion”
  - No dead tasks

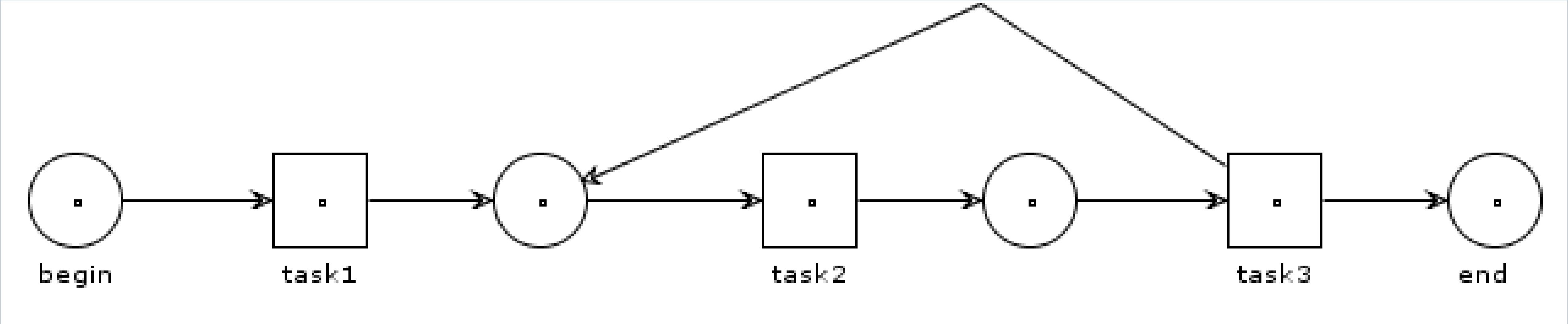
Analysis of WF-nets



Dead task
and
No option to complete

Analysis of WF-nets

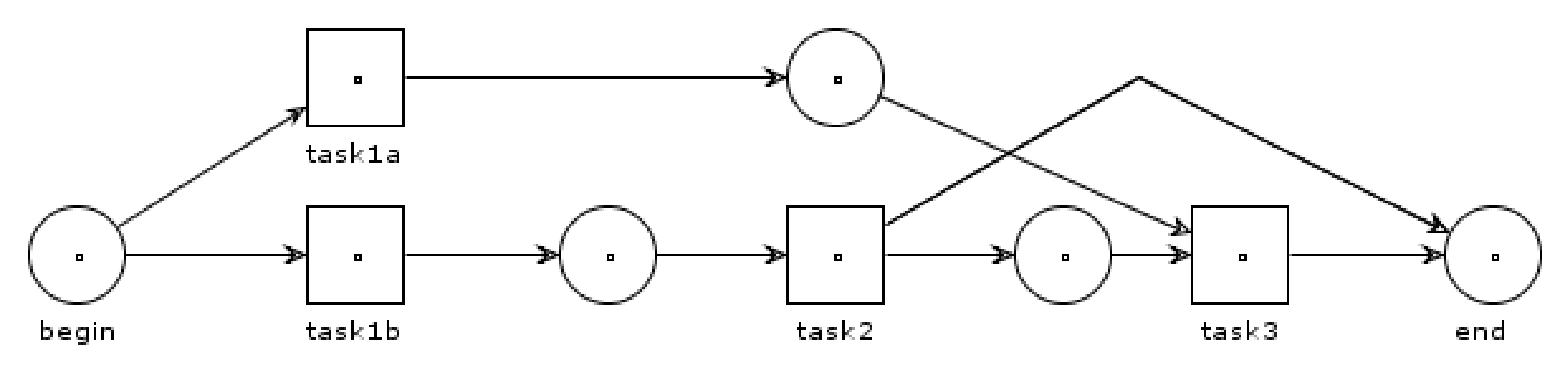
Unbounded and never ending ("livelock")



No proper completion

Analysis of WF-nets

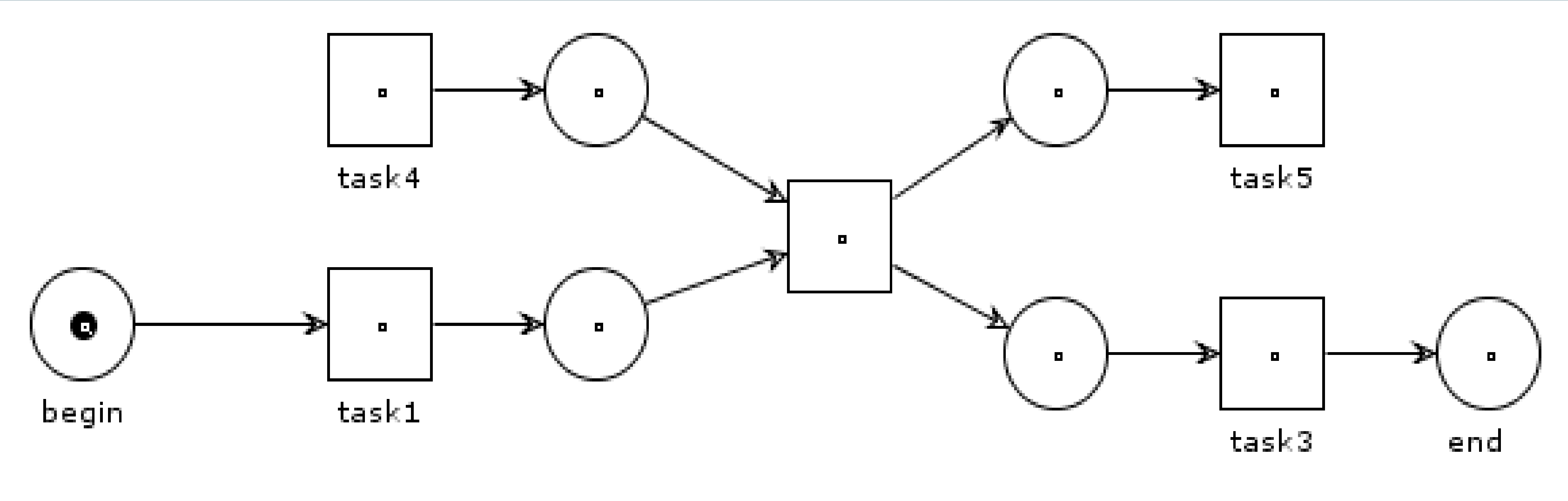
Deadlock before or after termination



No option to complete
and
no proper termination

Analysis of WF-nets

Dangling tasks



Not a WF-net

Analysis of WF-nets

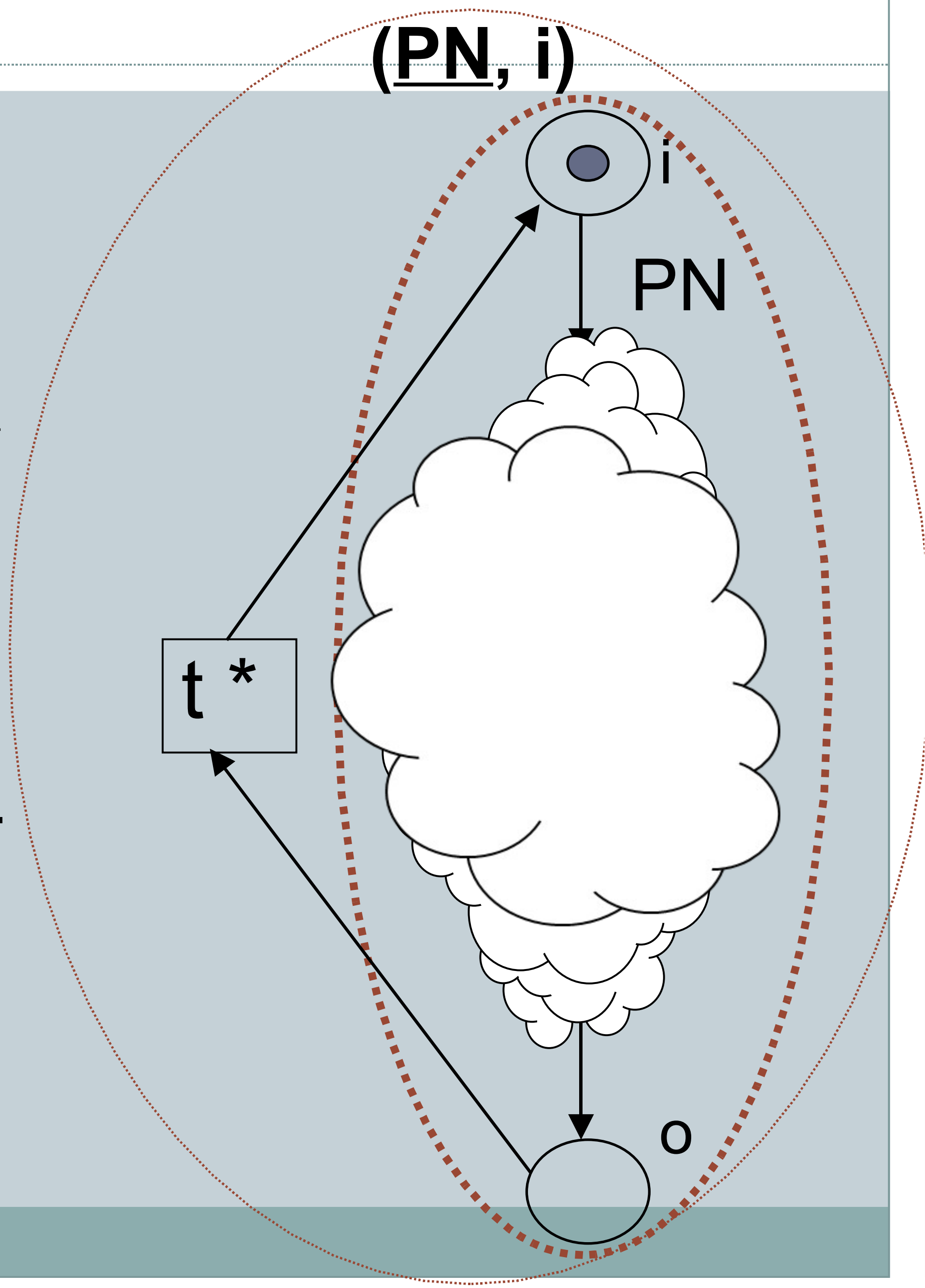
- Formal definition of soundness of a WF-net PN=(P,T,F) : *(See page 275 van der Aalst)*
  - 1) Option to complete : $\forall_M (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o);$
  - 2) Proper termination : $\forall_M (i \xrightarrow{*} M \wedge M \geq o) \Rightarrow (M = o);$
  - 3) No dead transitions : $\forall_{t \in T} \exists_{M,M'} i \xrightarrow{*} M \xrightarrow{t} M';$

Analysis of WF-nets

- Analysis of WF-nets
  - Qualitative analysis
    - General properties of WF-nets
    - State space analysis of WF-nets
  - Quantitative analysis (next lecture)

Analysis of WF-nets

- A WF-net PN is sound if and only if (<u>PN</u>, i) is life and bounded
- <u>PN</u> is the short-circuited PT-net of PN, created by adding t*



Analysis of WF-nets

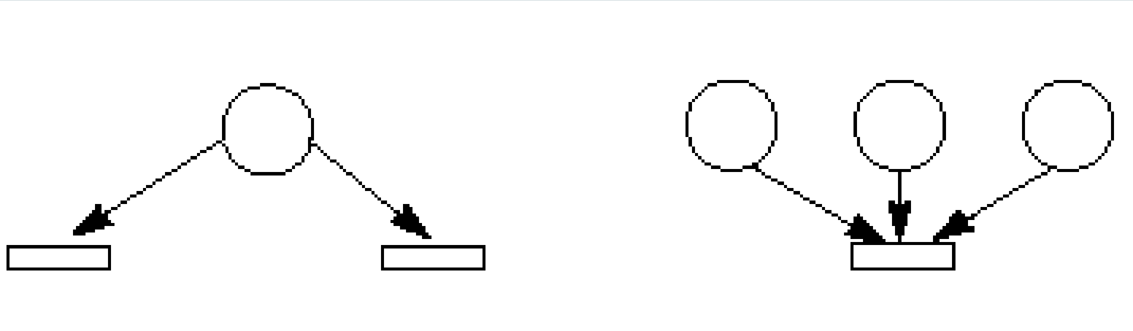
- PT nets with a finite state space (bounded) still might suffer from state space explosion problem :
  - Eg. State space of an EN system with n places < (2^n)
  - Analysis of general PT-systems intractable
- State space analysis of soundness general WF-nets has the same problem
- Therefore we will look for structural characterizations of soundness of WF-nets

Analysis of WF-nets

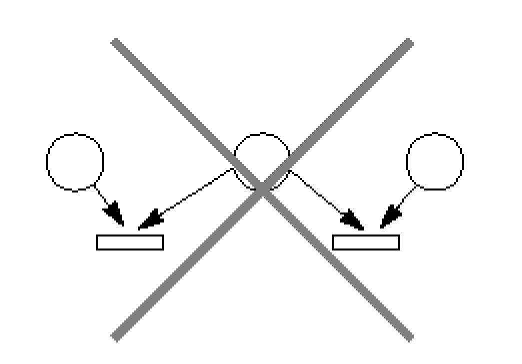
- See van der Aalst App. A.4.
  - Free choice WF-nets
  - Well structured WF-nets
  - S-coverable WF-nets

Analysis of WF-nets

Free choice Petri nets



A net with transitions in structural Conflict is not a free choice net



Analysis of WF-nets

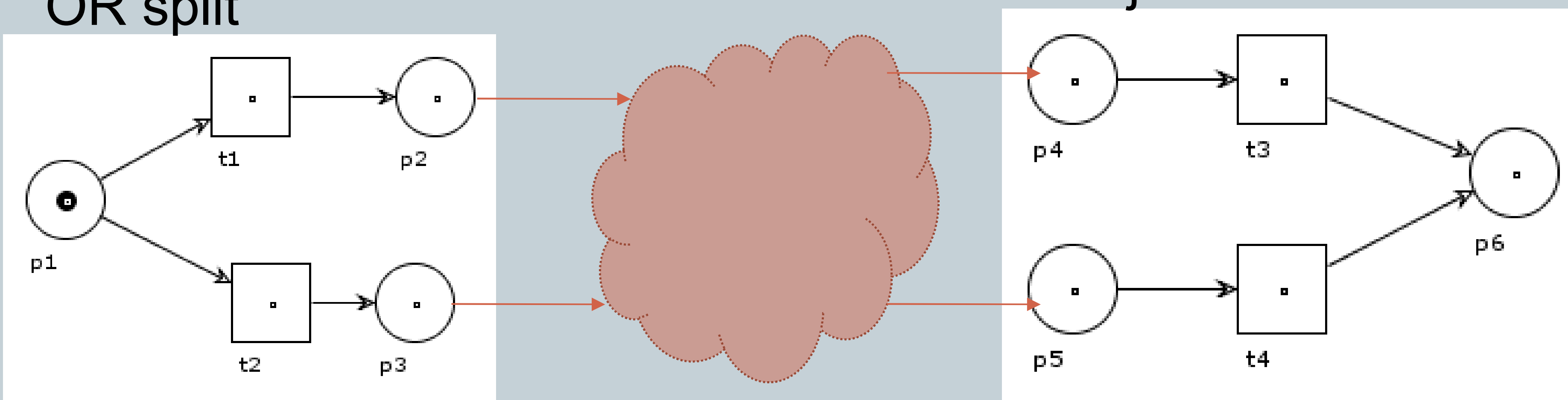
- For free choice WF-nets, soundness can be decided in polynomial time
- Free choice nets are suited to model sequence, choice and concurrency in many cases
- There are however useful sound WF-nets that are not free choice (see eg. exercise 1.2./HO II)

Analysis of WF-nets

Well structured WF-nets

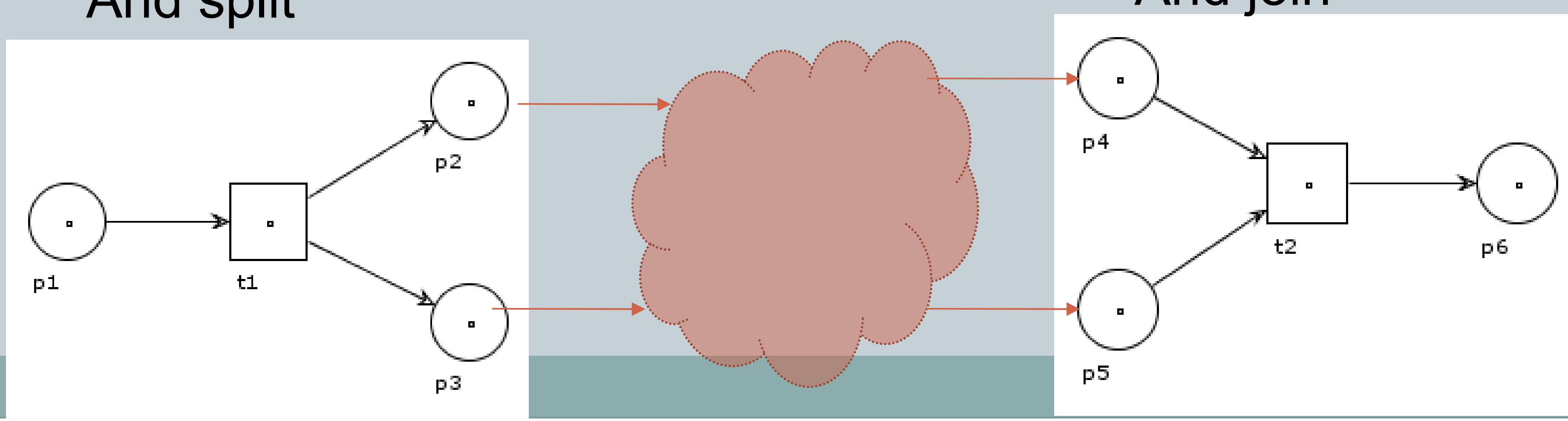
OR split

OR join



And split

And join

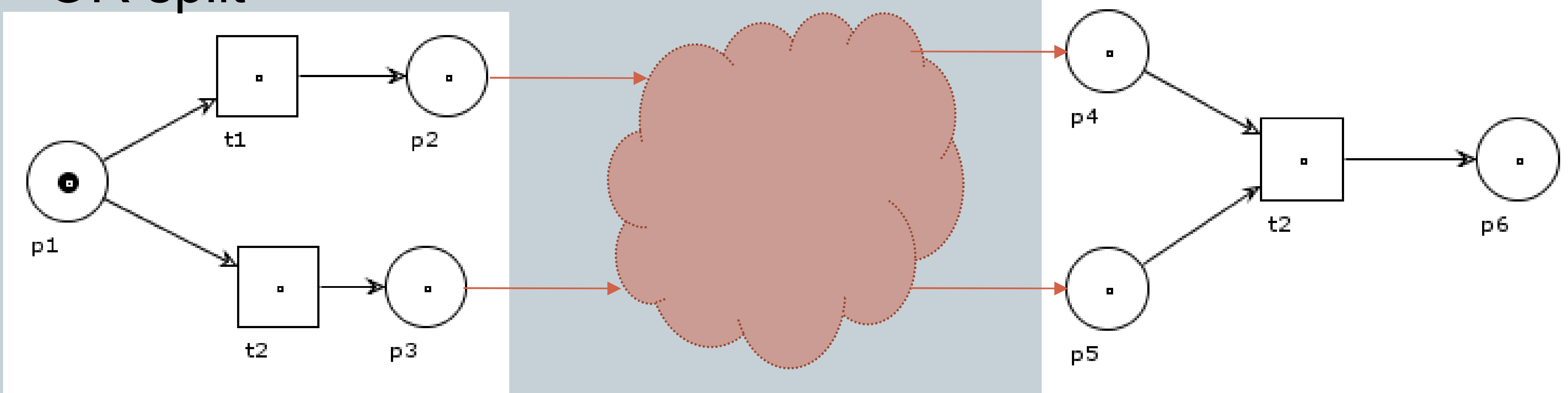


Analysis of WF-nets

Not Well structured WF-nets

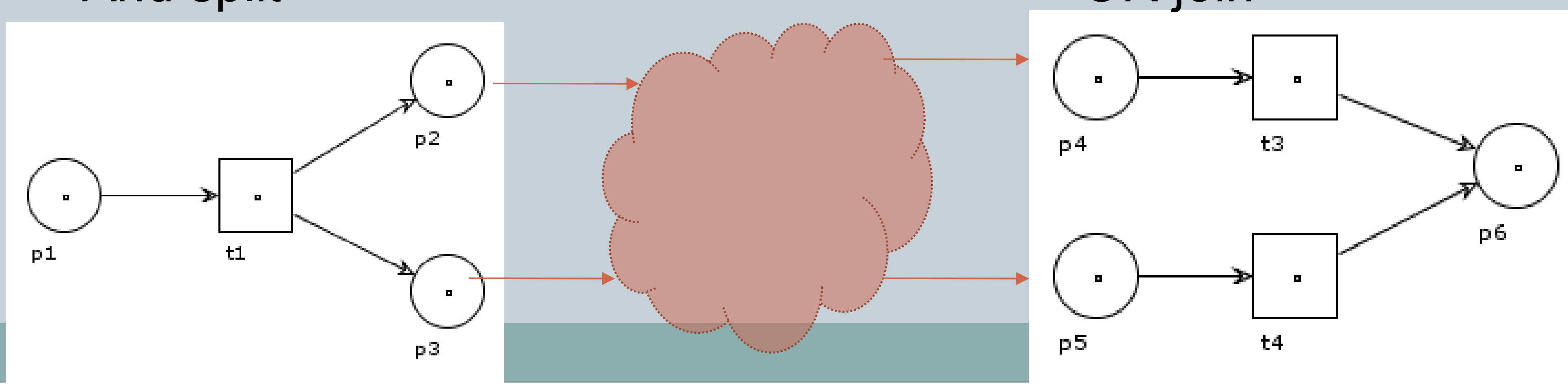
OR split

And join



And split

OR join



Analysis of WF-nets

- For Well-structured WF-nets, soundness can also be decided in polynomial time

- Well structured nets are suited to model sequence, choice and concurrency in many cases

- However Free choice nets need not be Well structured, or vice versa

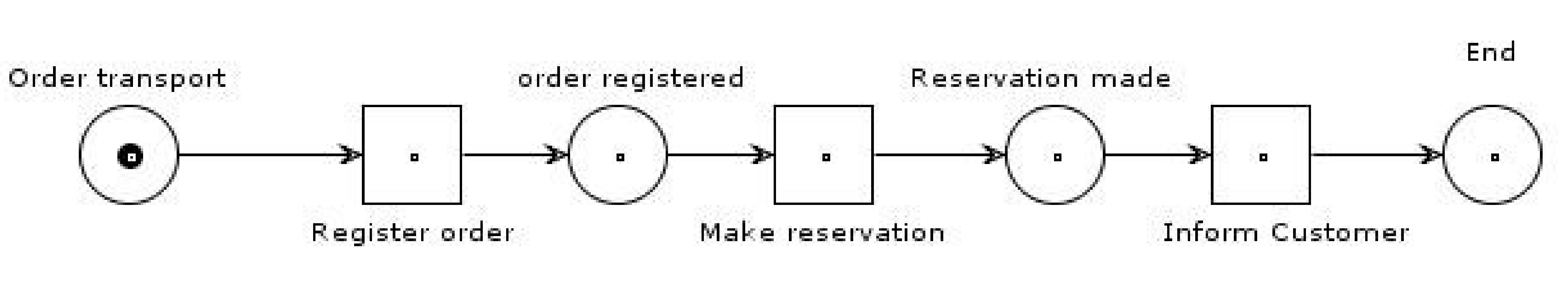
- In fact there are sound WF-nets which are neither

Analysis of WF-nets

- Definitions:
  - A WF-net is S-coverable if the short-circuited WF-net is S-coverable
  - The short-circuited WF-net is S-coverable if it is covered by S-components
  - A (part of a) Petri net is an S-component if:
    - It is a state machine and
    - Strongly connected

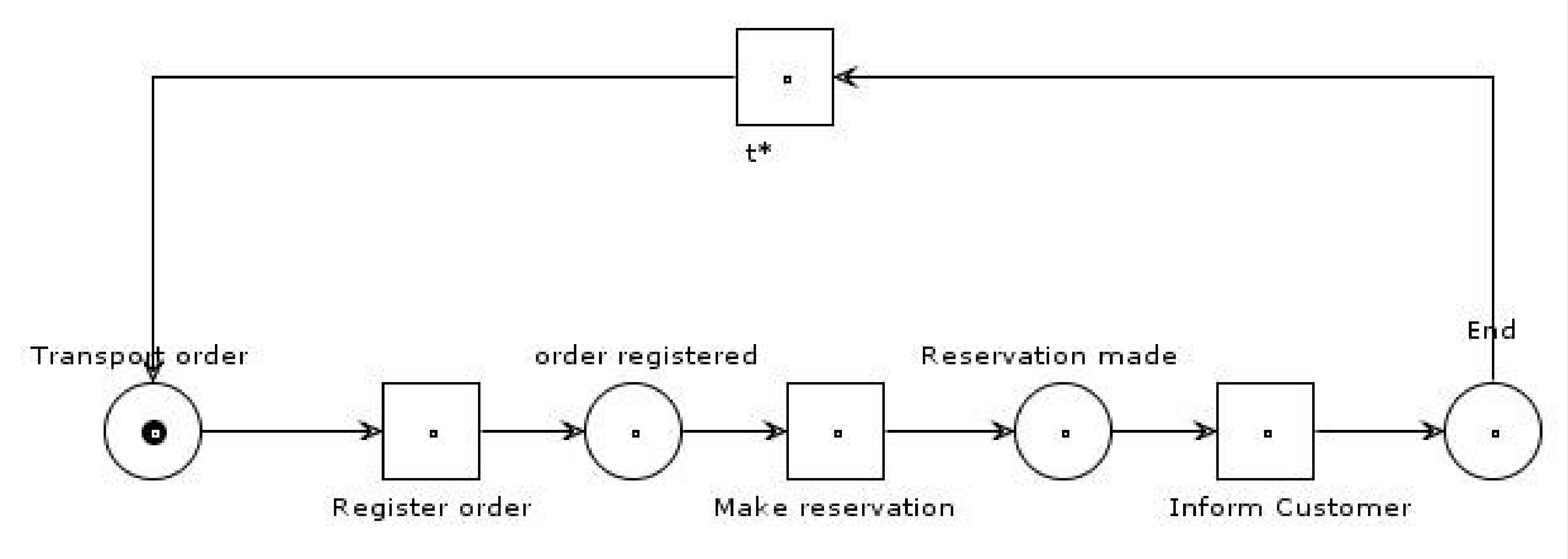
Analysis of WF-nets

A Petri net is a *state machine* *iff each transition has exactly one input and one outputplace, eg. :*



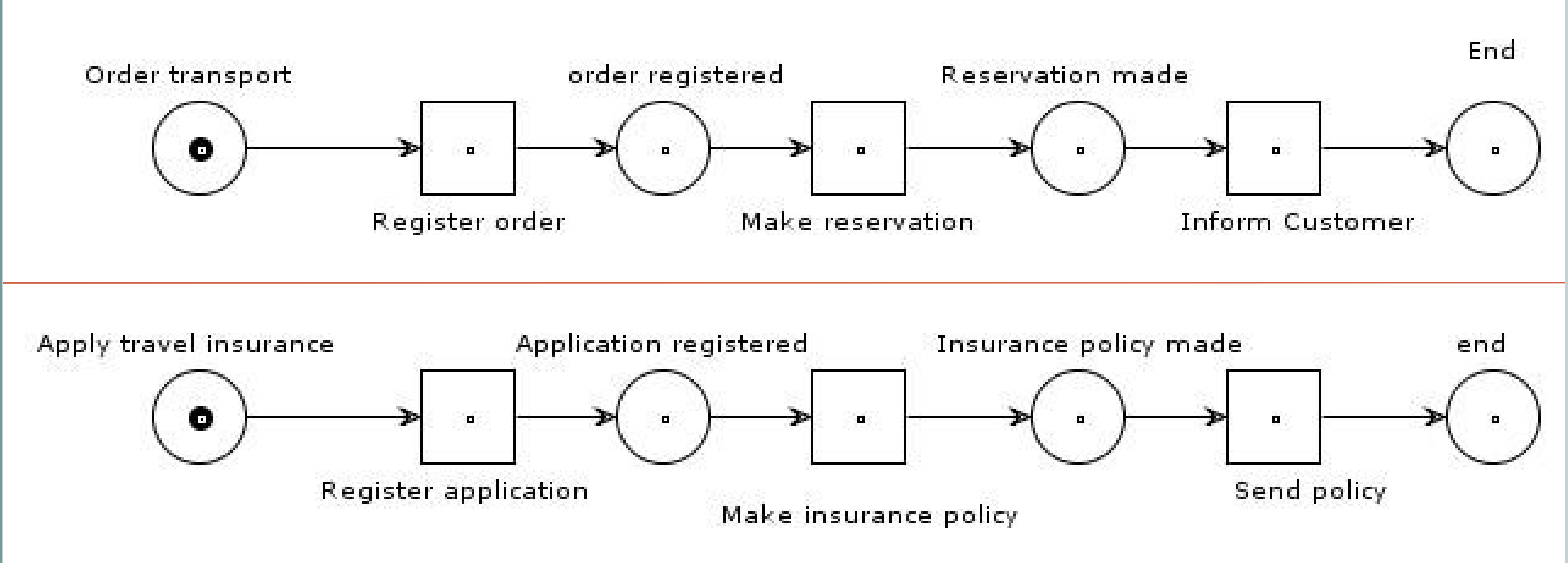
Analysis of WF-nets

An *S-component* is a strongly connected state machine, eg.:

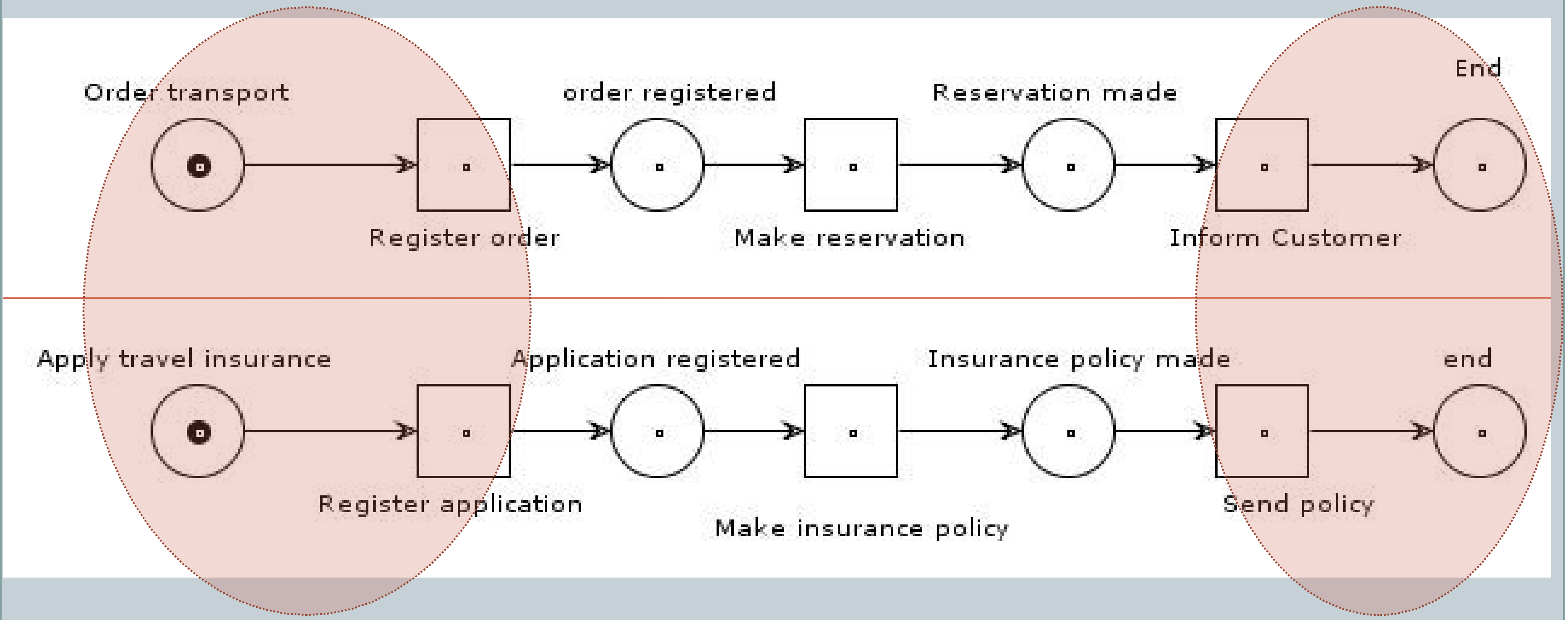


Analysis of WF-nets

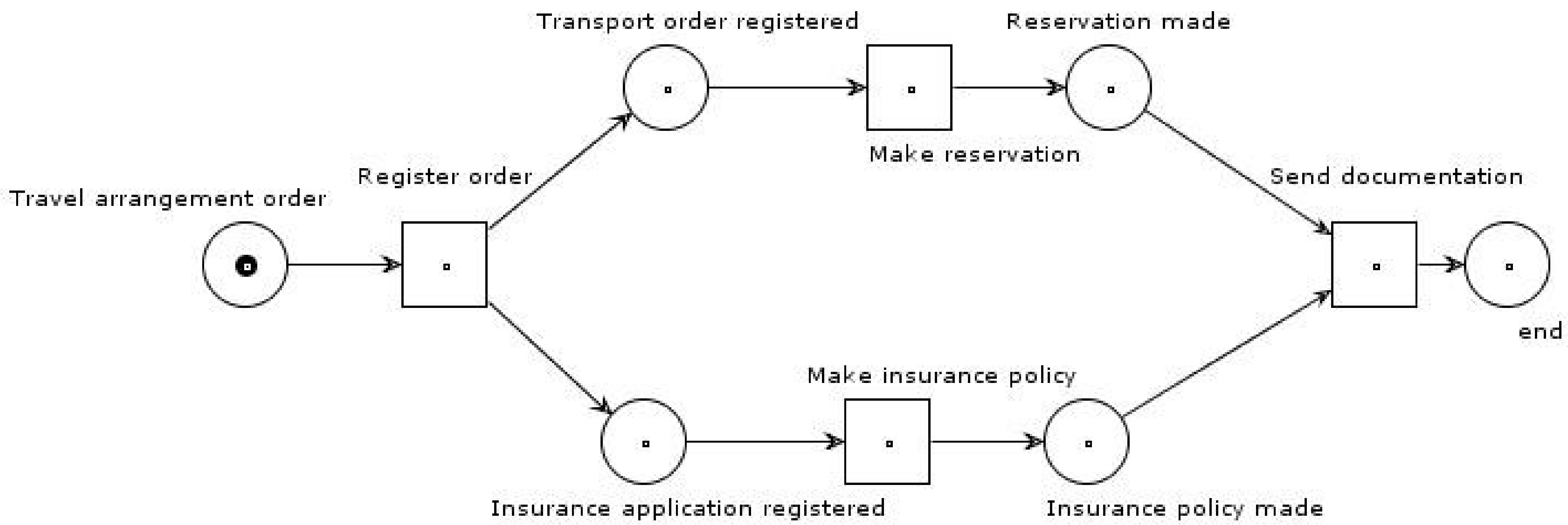
Two State machines :



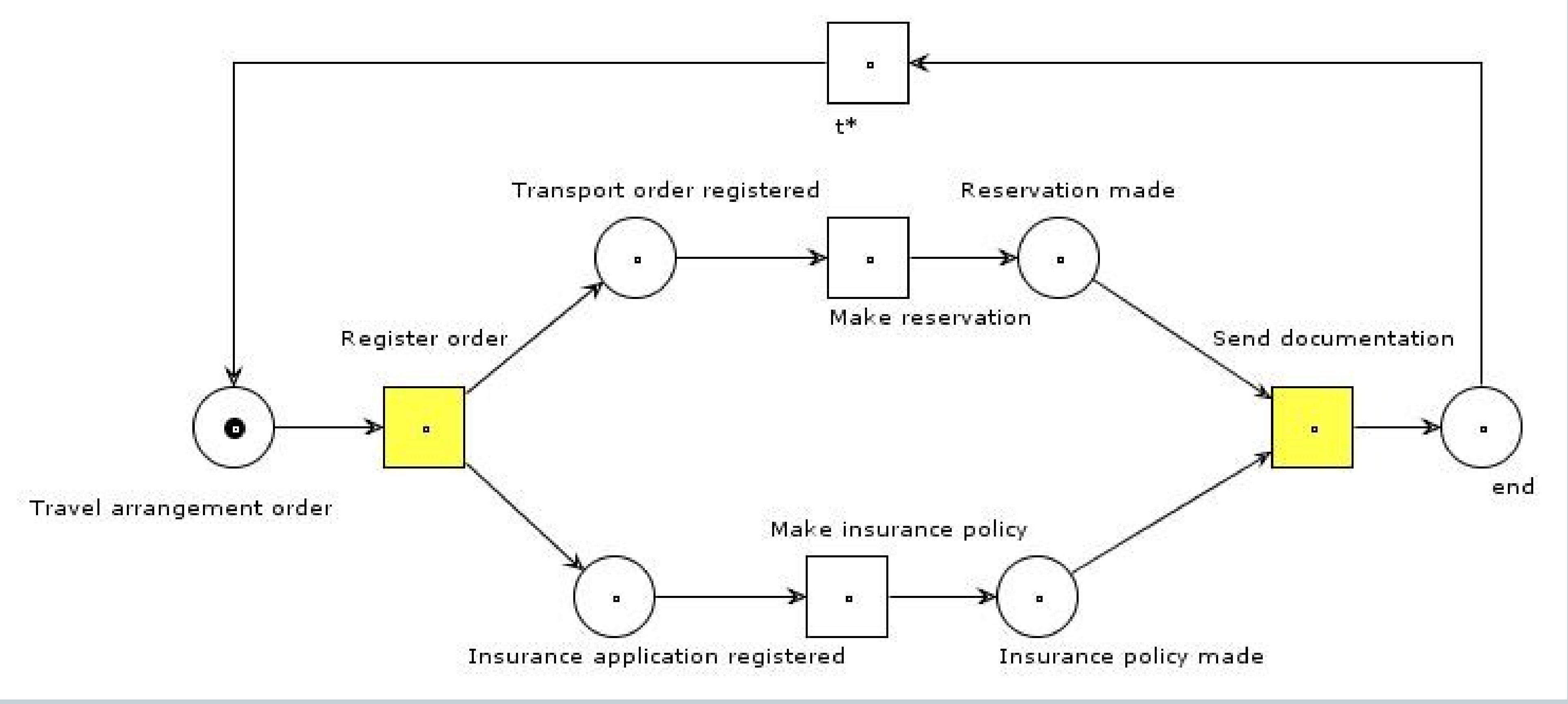
Merging two State machines .....



Into one WF-net consisting of 2 state machines:

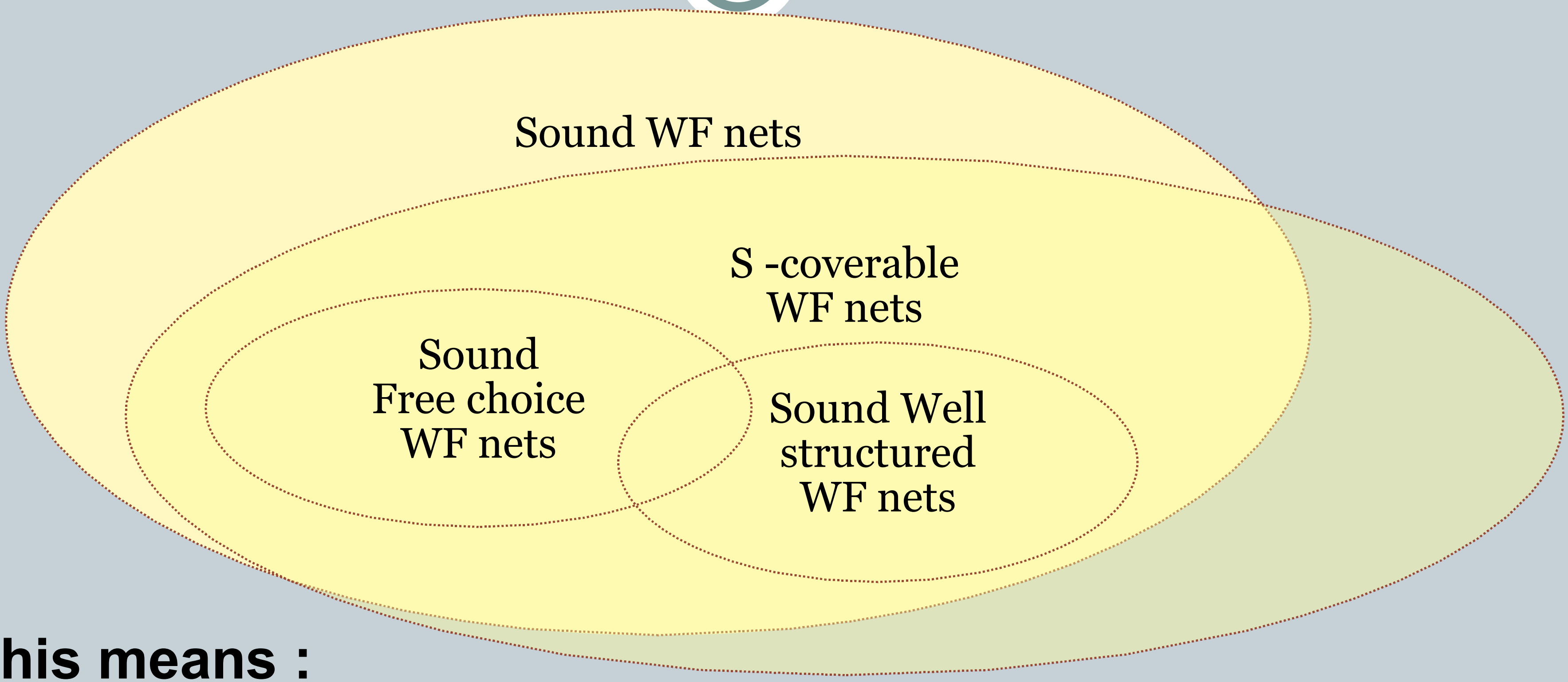


Or a short-circuited WF-net covered by 2 S-components :



Therefore the WF-net is S-coverable

Analysis of WF-nets



So, this means :

- a sound Free choice WF-net is S-coverable (and safe)

- a sound Well-structured WF-net is S-coverable (and safe)

But, there are S-coverable sound WF-nets :

- that are not Free Choice!

- that are not well-structured!

Analysis of WF-nets

Deciding soundness for subclasses is easier!

Petri net class	**Complexity soundness analysis**	
WF-net	Intractable (EXPSPACE: ”very very hard”!)	
Free Choice WF-net	Tractable (P : “easy”)	
Well-structured WF-net	Tractable (P : “easy”)	
S-coverable WF-net	Intractable (PSPACE: “very hard”)	

So if you can model a Workflow as a Free-choice WF-net or a Well-handled WF-net than you should !

But be ware, this is not always possible!

Overview Petri net family