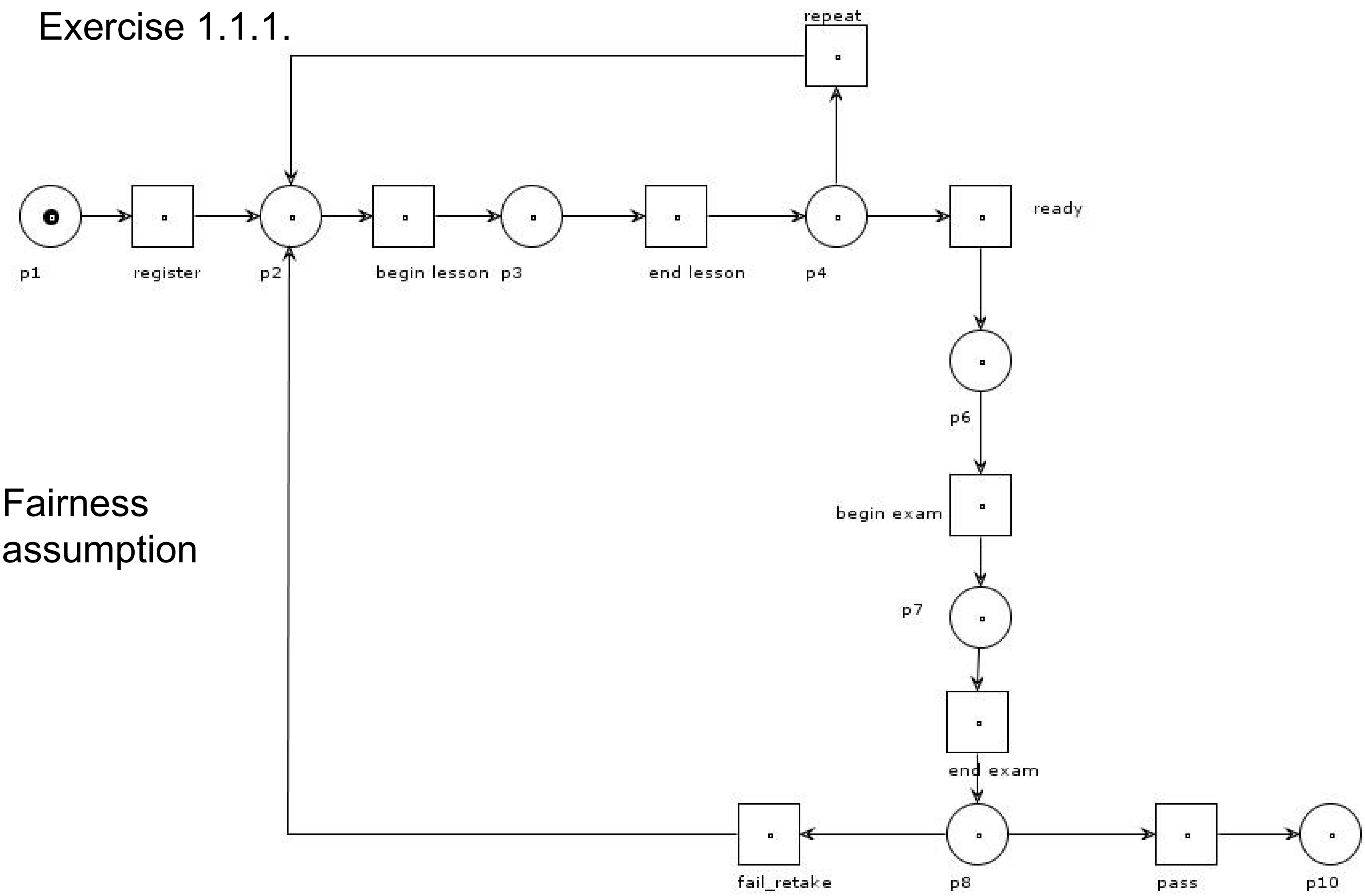
Performance analysis with Timed Petri nets

Outline third Lecture

- Recap second lecture
- Finish remainder of WF-net theory
- Performance analysis with Petri nets

Recap 2nd lecture

Exercise 1.1.1.

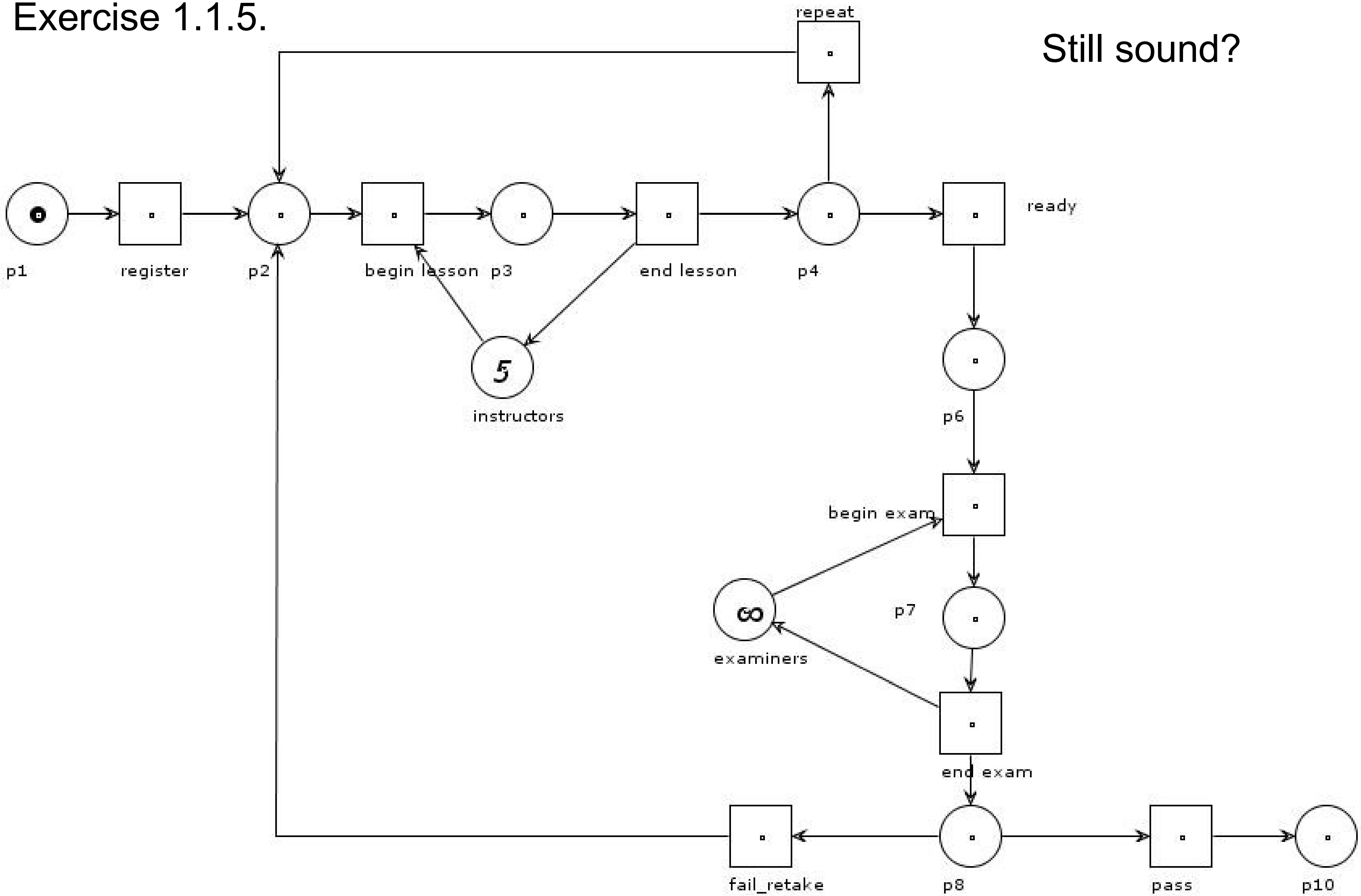


Fairness assumption

Recap 2nd lecture

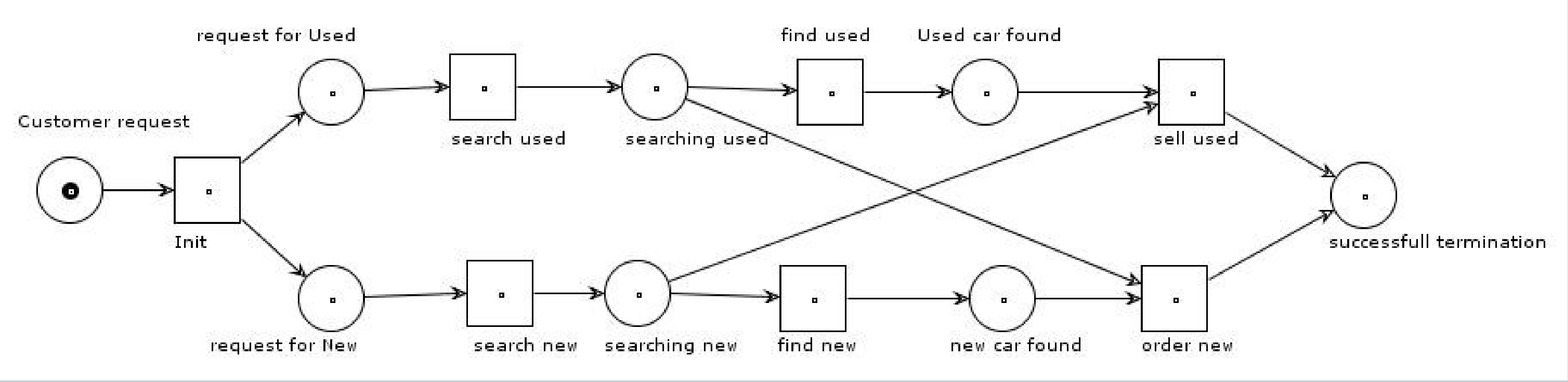
Exercise 1.1.5.

Still sound?



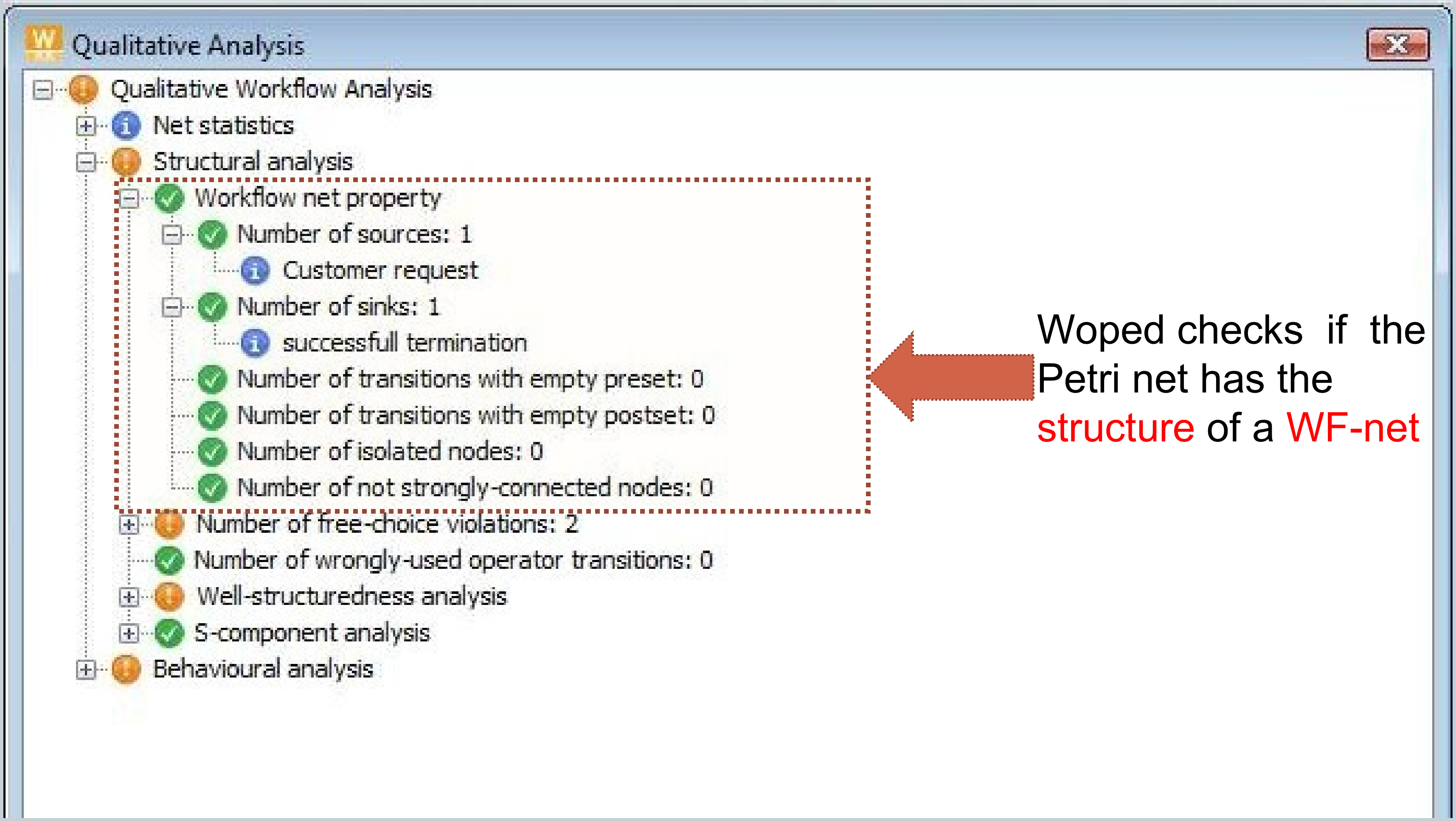
Recap 2nd lecture

Exercise 1.2.



Recap 2nd lecture

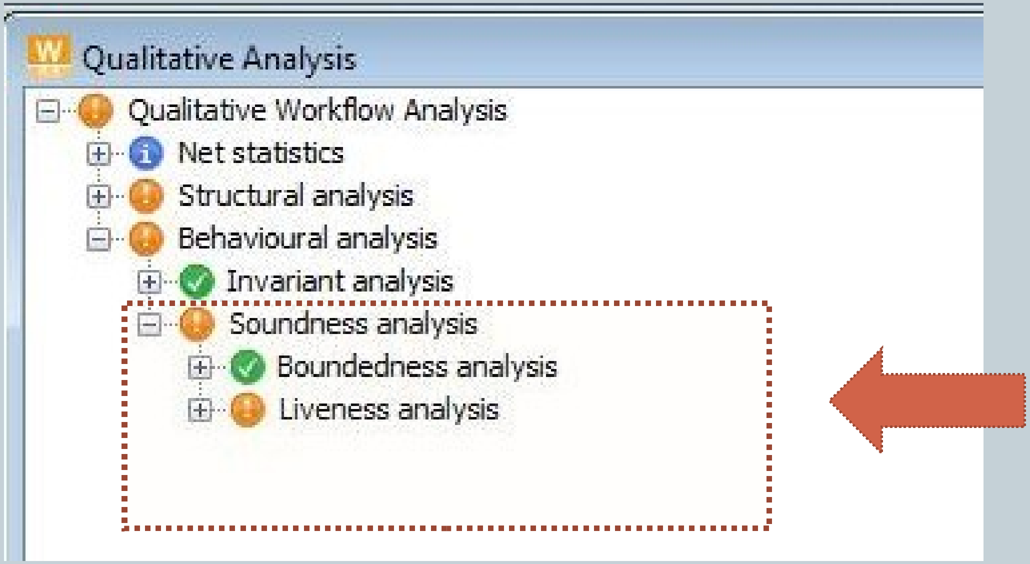
Exercise 1.2.6.



Woped checks if the Petri net has the structure of a WF-net

Recap 2nd lecture

Exercise 1.2.6.

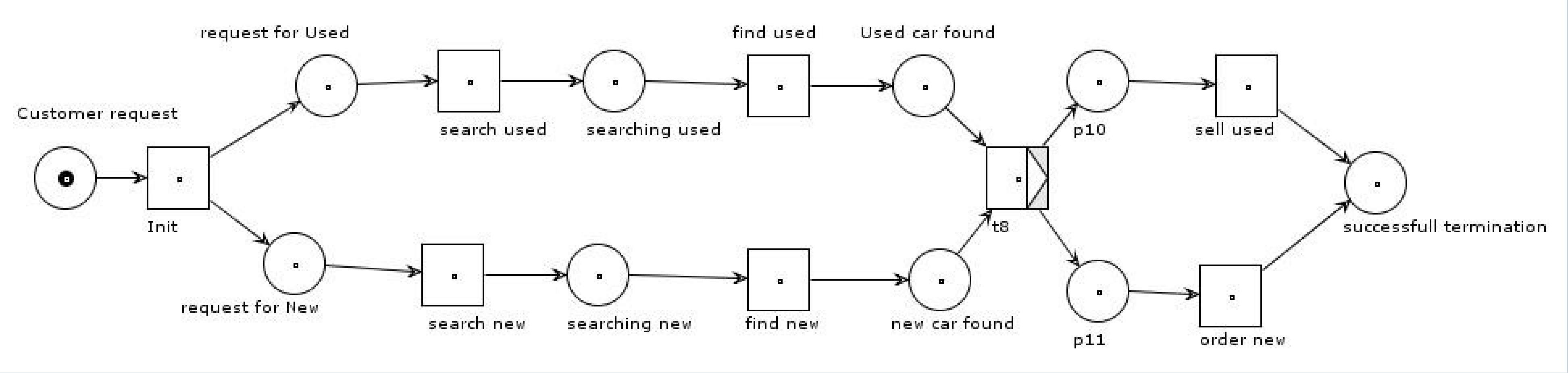


Woped checks if the Petri net has the behaviour of a sound WF-net

Recap 2nd lecture

Exercise 1.2.5.

Solution 1 : this is indeed a sound workflow net

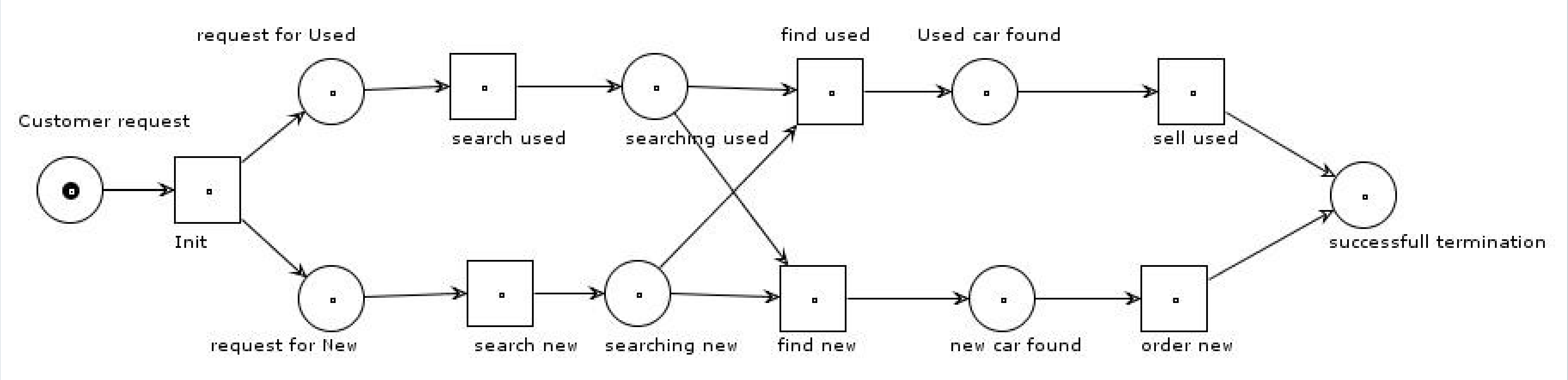


But is it still a valid workflow net?

Recap 2nd lecture

Exercise 1.2.5.

Solution 2 : this is also a sound workflow net

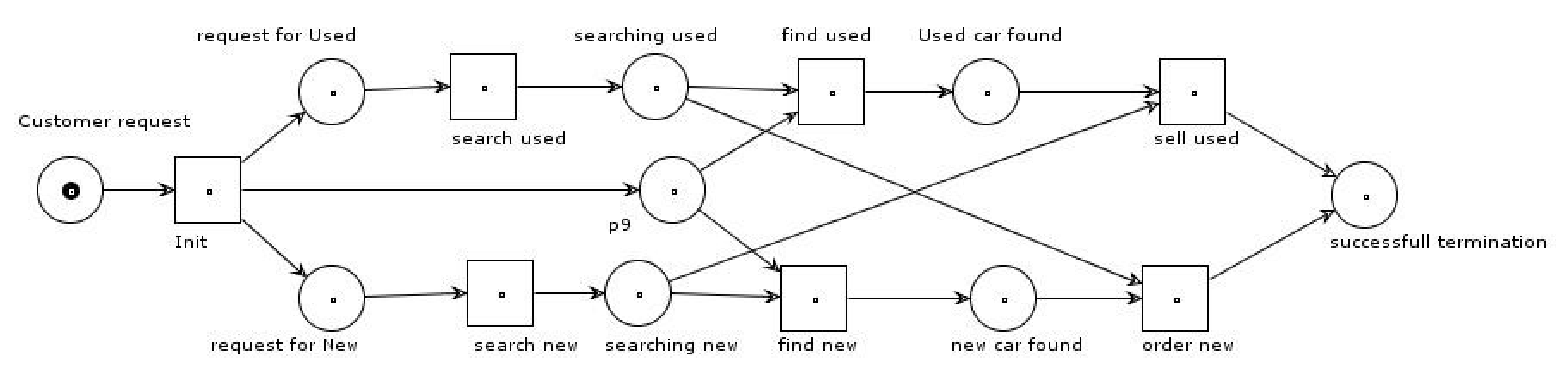


But again : is it still a valid workflow net?

Recap 2nd lecture

Exercise 1.2.5.

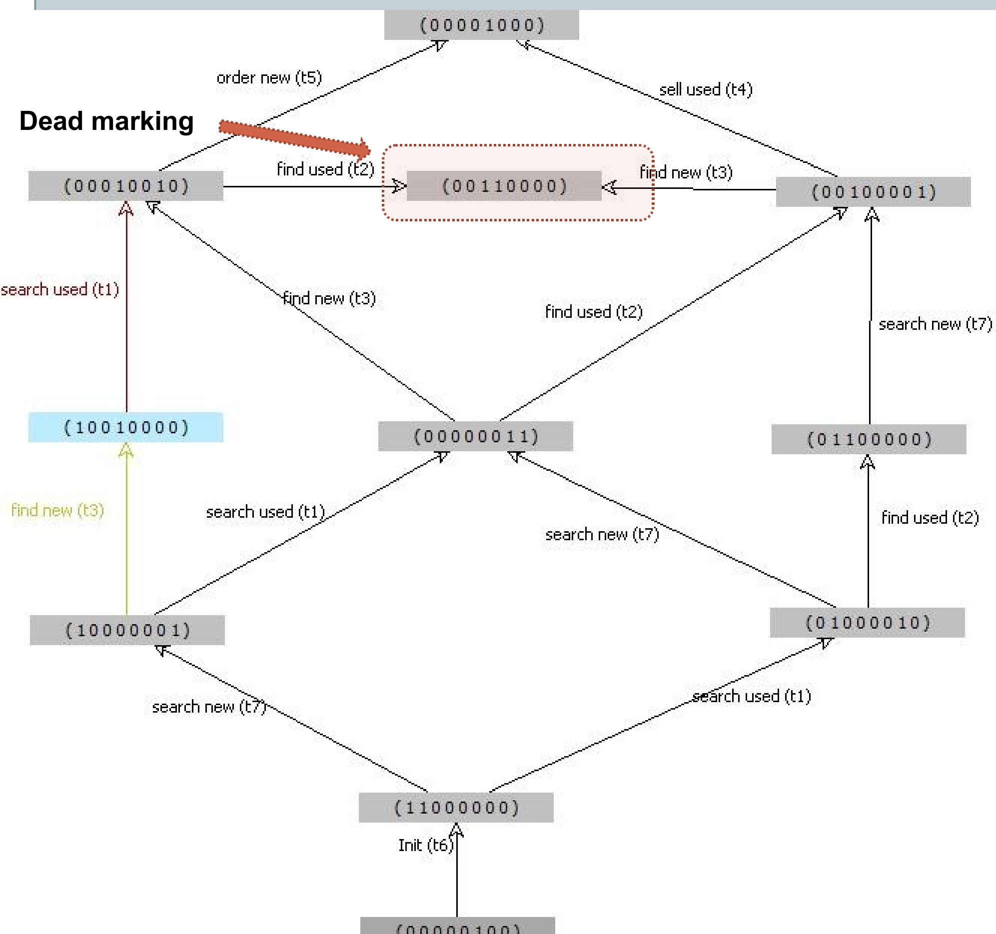
Solution 3 : this is a sound workflow net



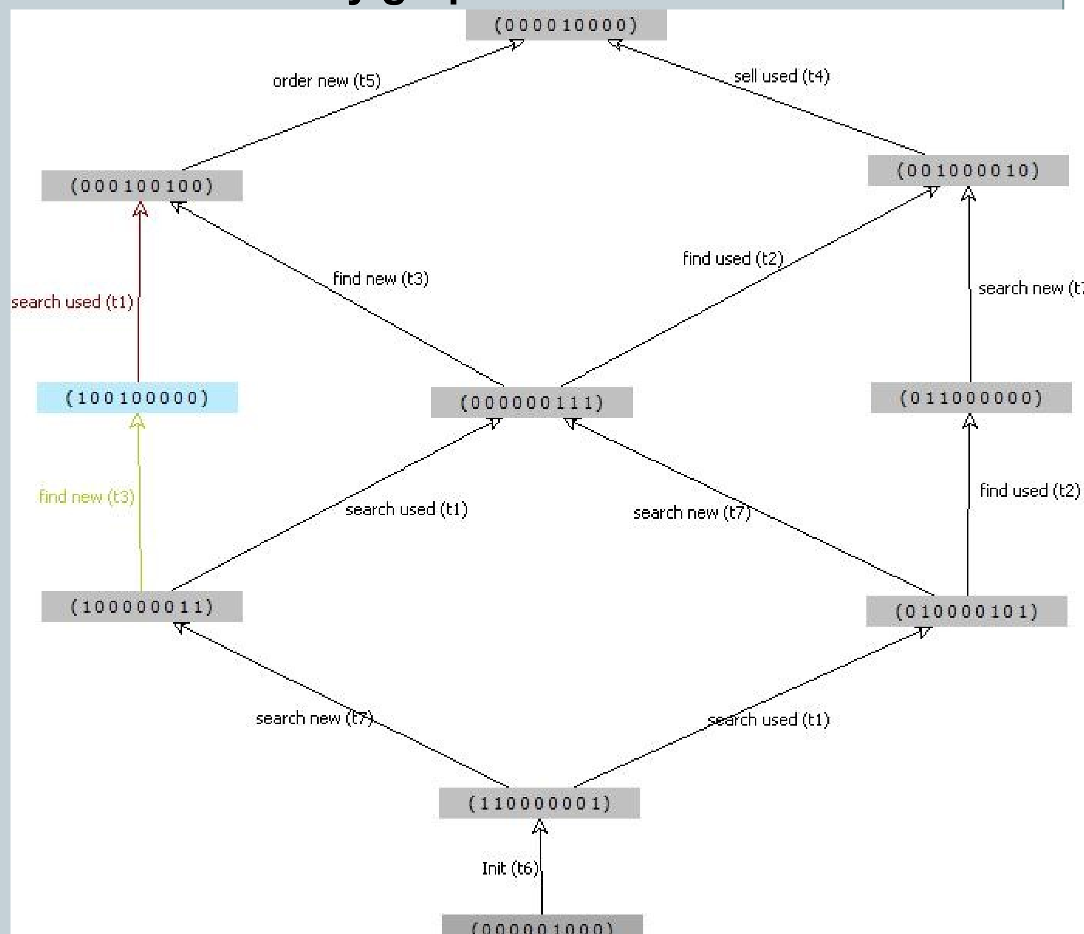
And appears also still a valid workflow net

Exercise 1.2.5.

Reachability graph of the Unsound WF-net



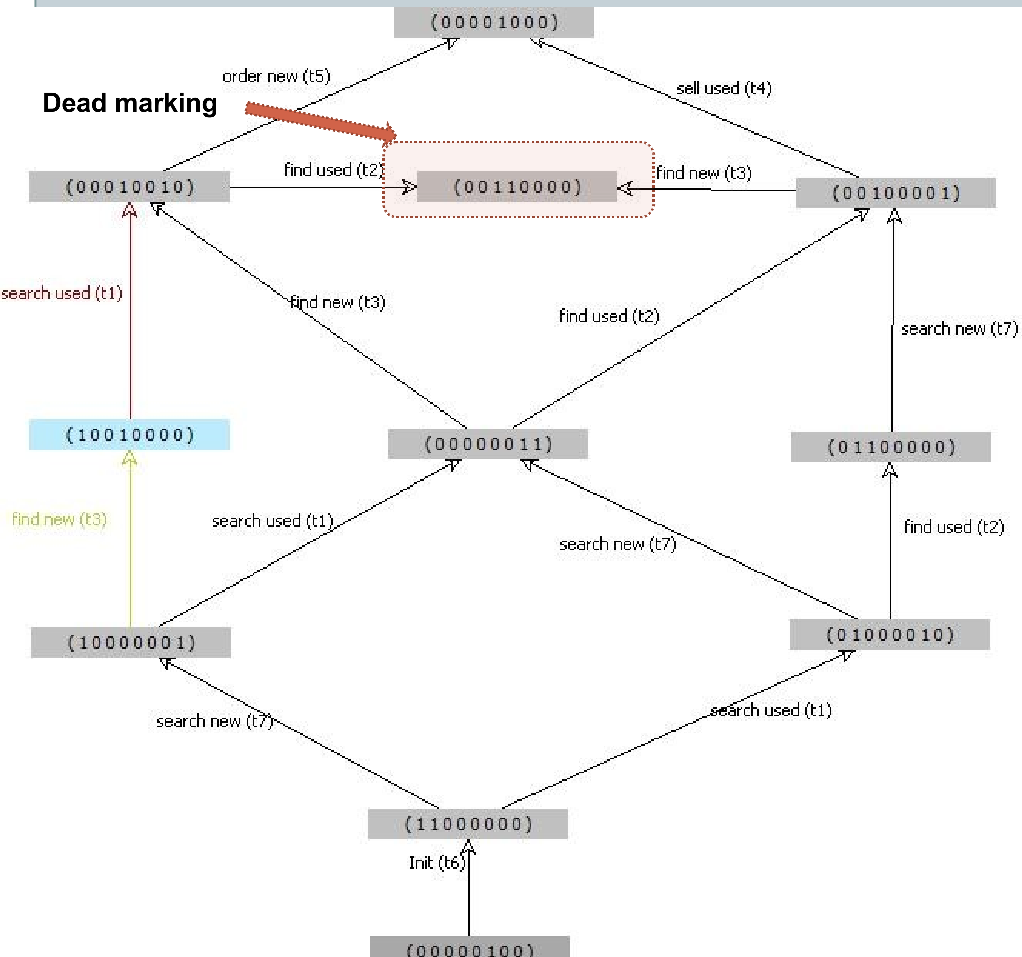
Reachability graph of solution 3

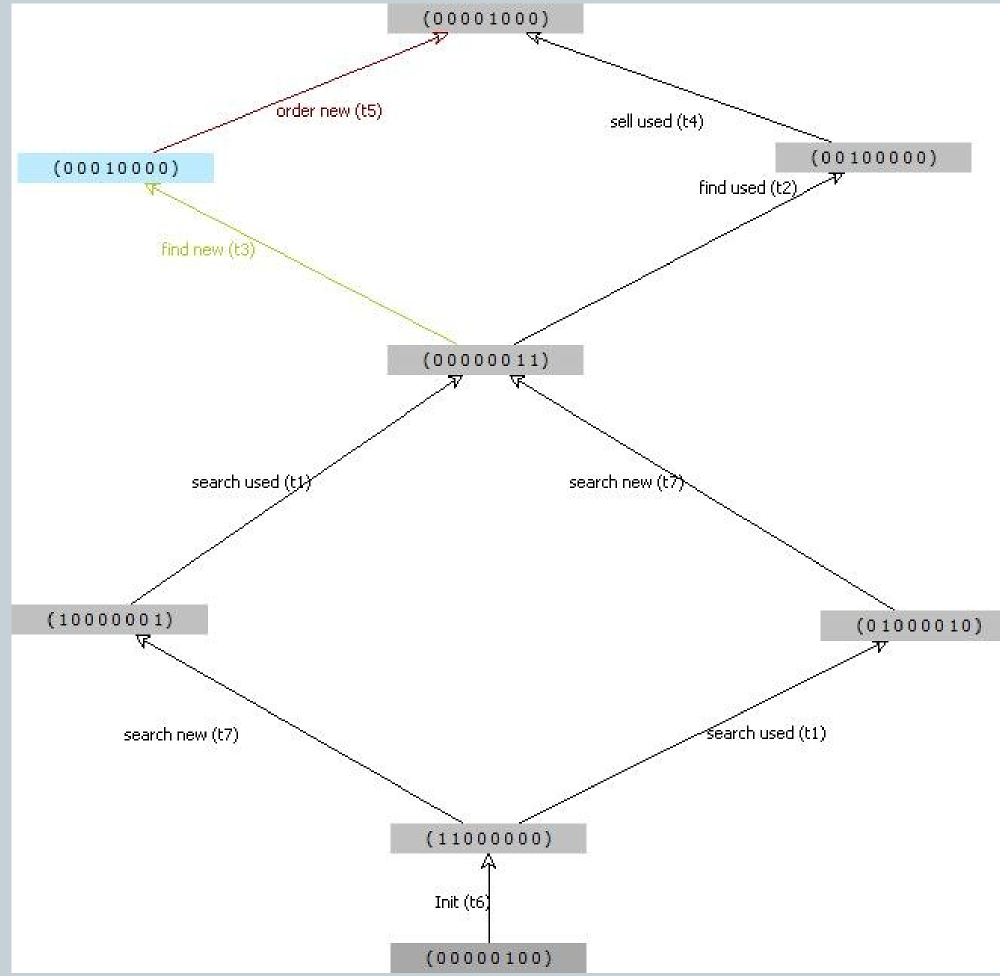


Recap 2nd lecture

Exercise 1.2.5.

Reachability graph of the Unsound WF-net → **Reachability graph of solution 2**

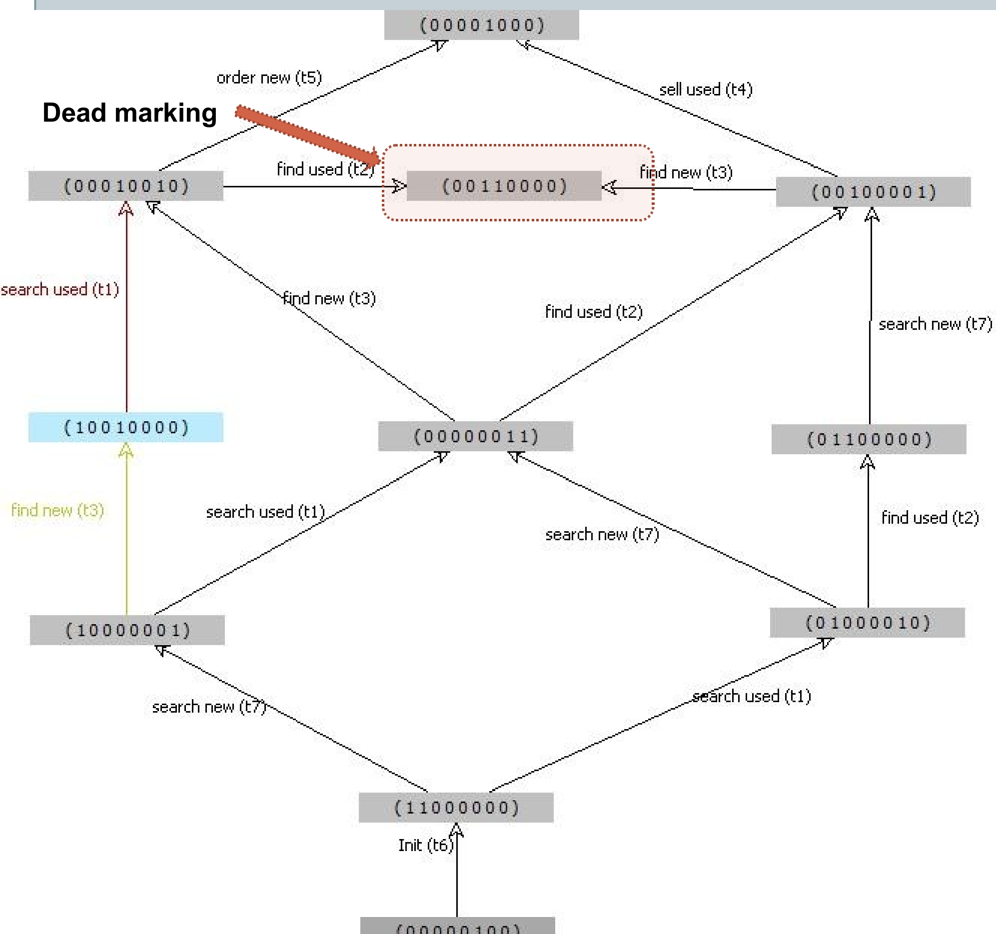




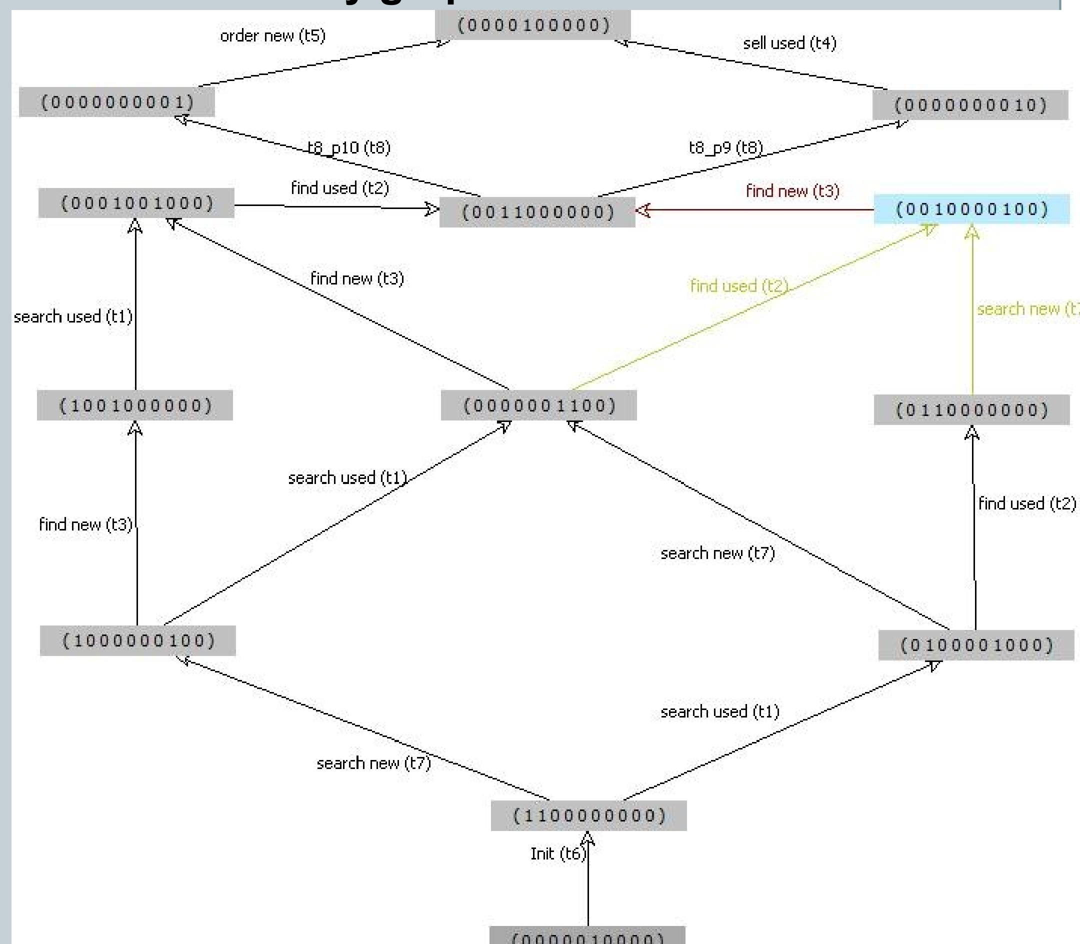
Recap 2nd lecture

Exercise 1.2.5.

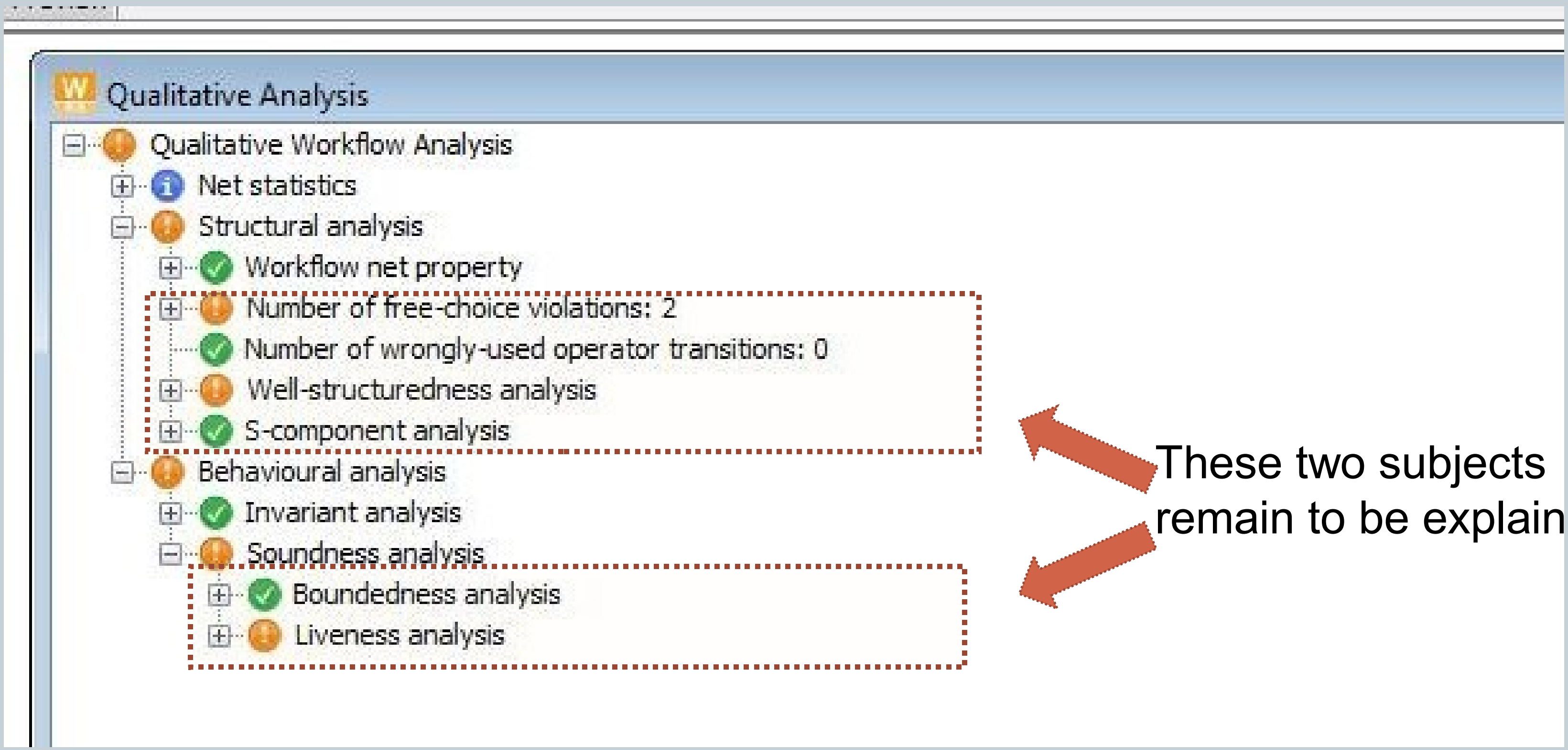
Reachability graph of the Unsound WF-net



Reachability graph of solution 1



Recap 2nd lecture



These two subjects remain to be explained

Outline third Lecture

- Recap second lecture
- Finish remainder of WF-net/soundness theory
- Performance analysis with Petri 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)

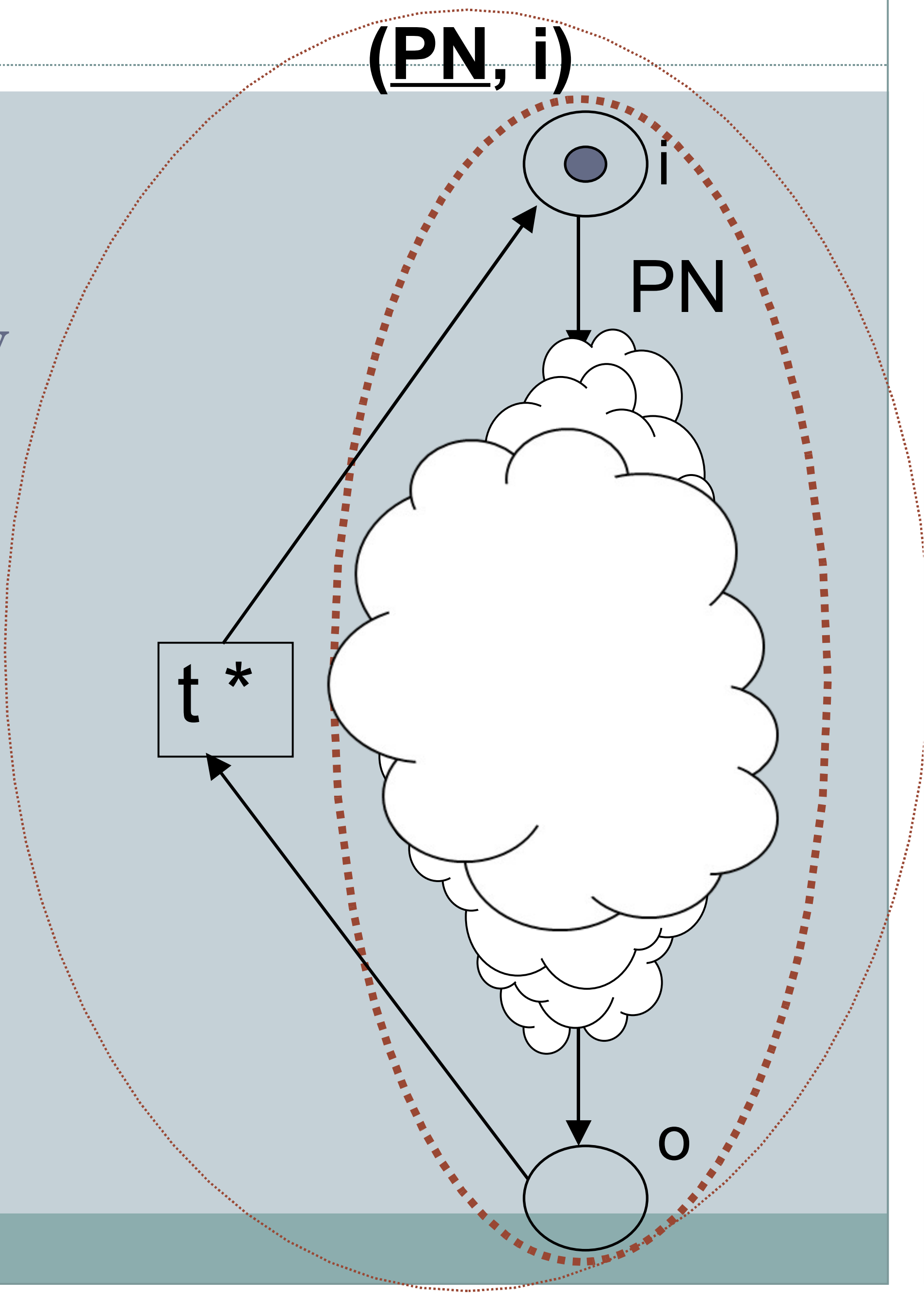
Analysis of WF-nets

- Theory of soundness

(See App. A3, page 276 van der Aalst)

  - A WF-net PN is sound if and only if (PN, i) is life and bounded

  - PN is the short-circuited PT-net of PN, created by adding t*



- To conclude :
  - We already had algorithms to prove liveness and boundedness of a PT-system (see 2nd lecture)
  - Using these results, we now also have an algorithm to prove soundness of a WF-system

Analysis of WF-nets

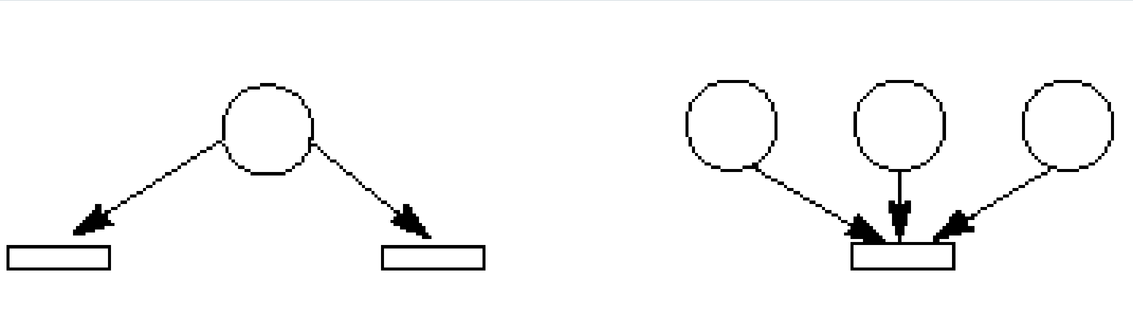
- PT nets with a finite state space (bounded) still might suffer from state space explosion problem :
  - Eg. State space of an EN system with n places < (2^n)
  - Analysis of general PT-systems intractable
- State space analysis of soundness of general WF-nets has the same problem
- Therefore we will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

Analysis of WF-nets

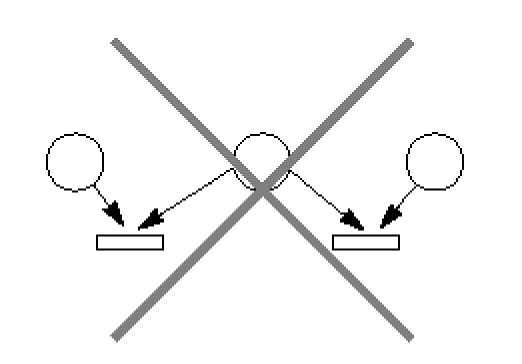
- See van der Aalst App. A.4.
  - Free choice WF-nets
  - Well structured WF-nets
  - S-coverable WF-nets

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

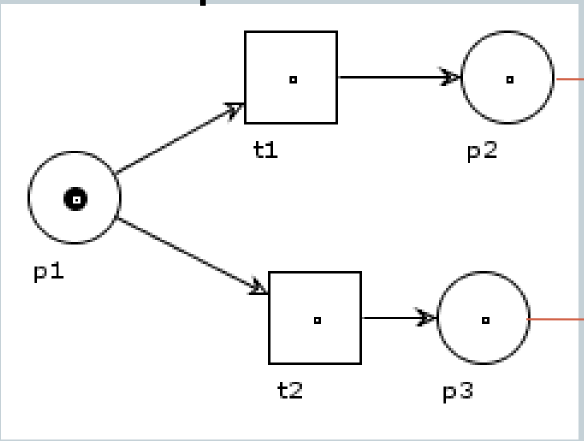
- Free choice nets are suited to model sequence, choice and concurrency in many cases

- There are however useful sound WF-nets that are not free choice (see eg. exercise 1.2./HO II)

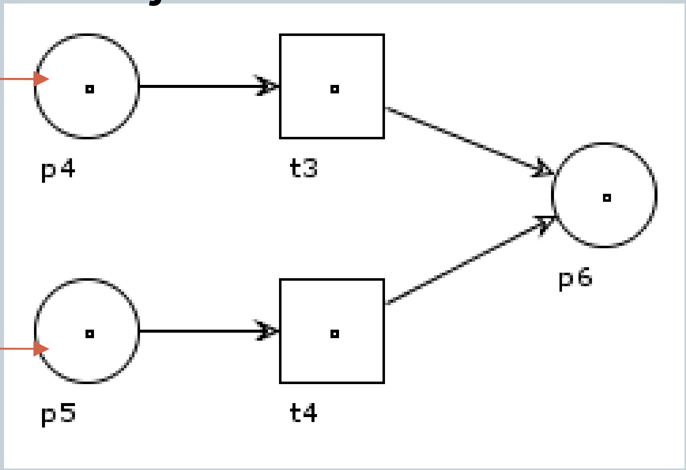
Analysis of WF-nets

Well structured WF-nets

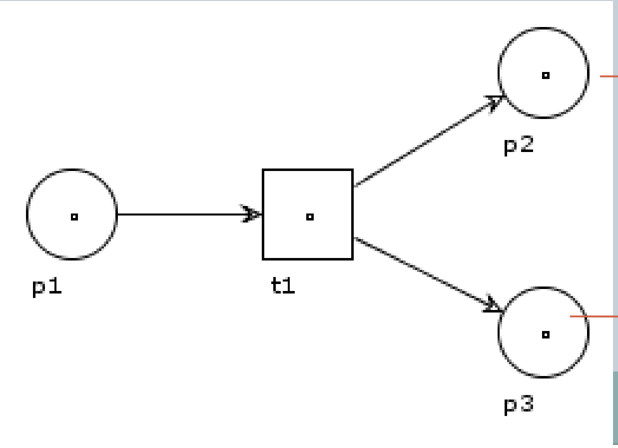
OR split



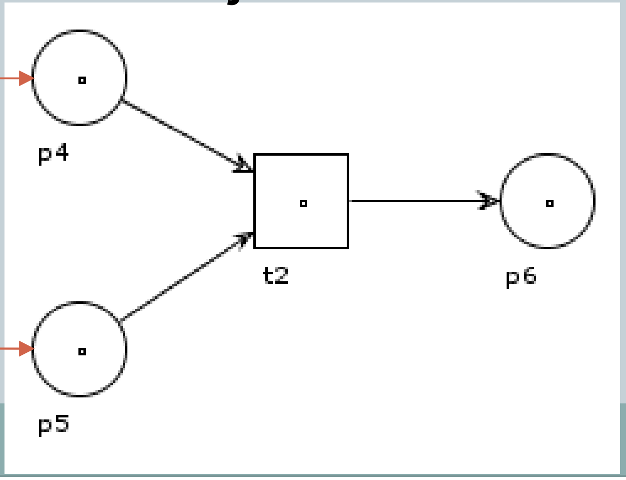
OR join



And split



And join

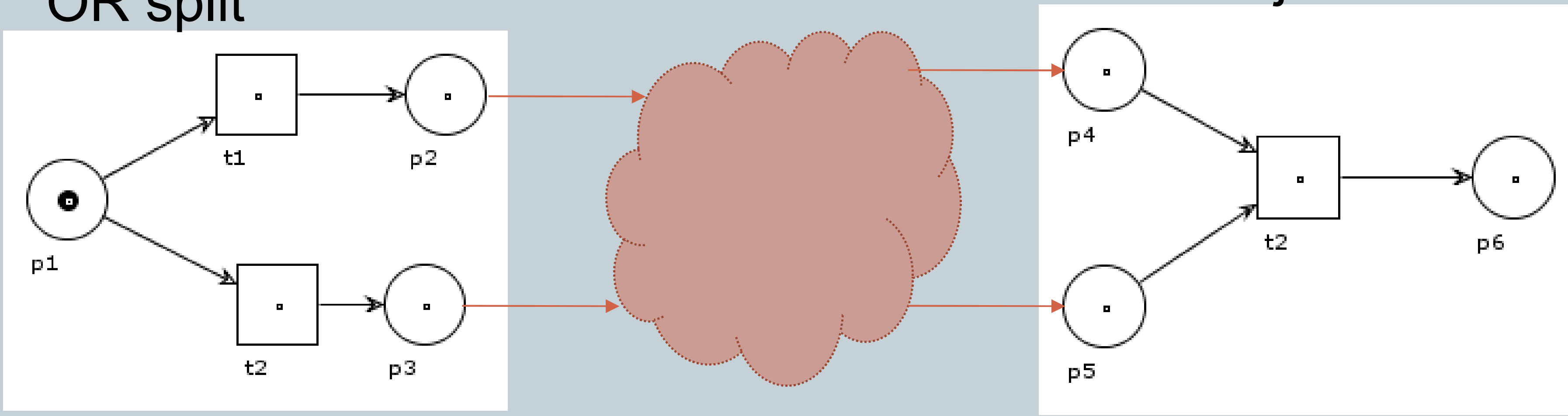


Analysis of WF-nets

Not Well structured WF-nets

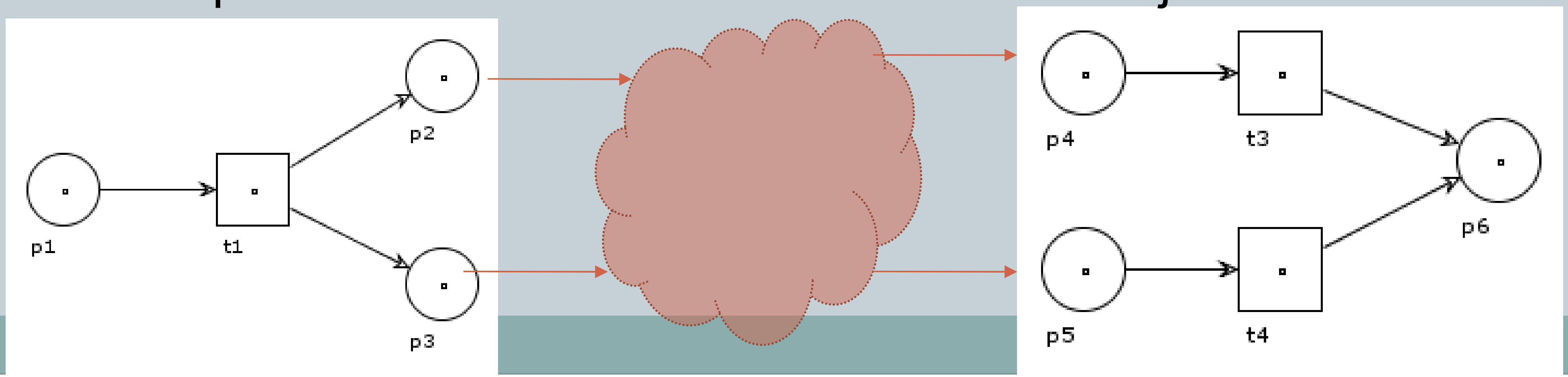
OR split

And join



And split

OR join



Analysis of WF-nets

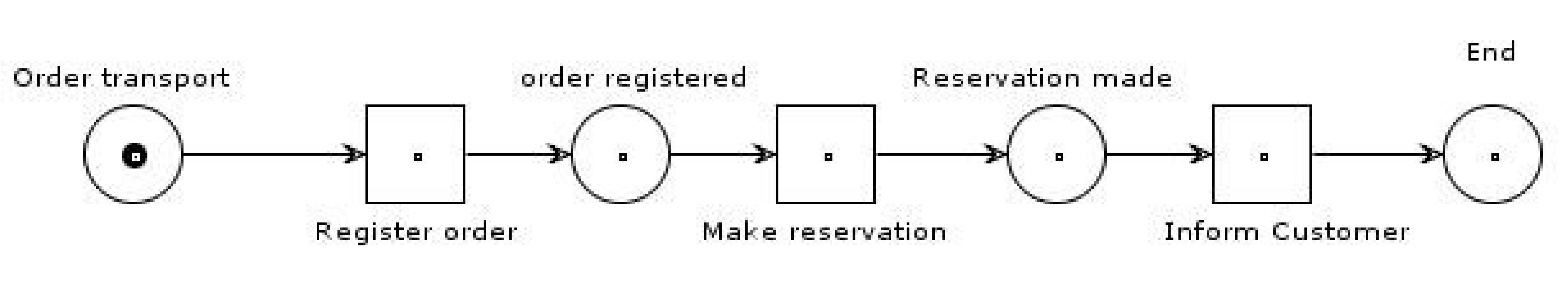
- For Well-structured WF-nets, soundness can also be decided in polynomial time
- Well structured nets are suited to model sequence, choice and concurrency in many cases
- However Free choice nets need not be Well structured, or vice versa
- In fact there are sound WF-nets which are neither

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

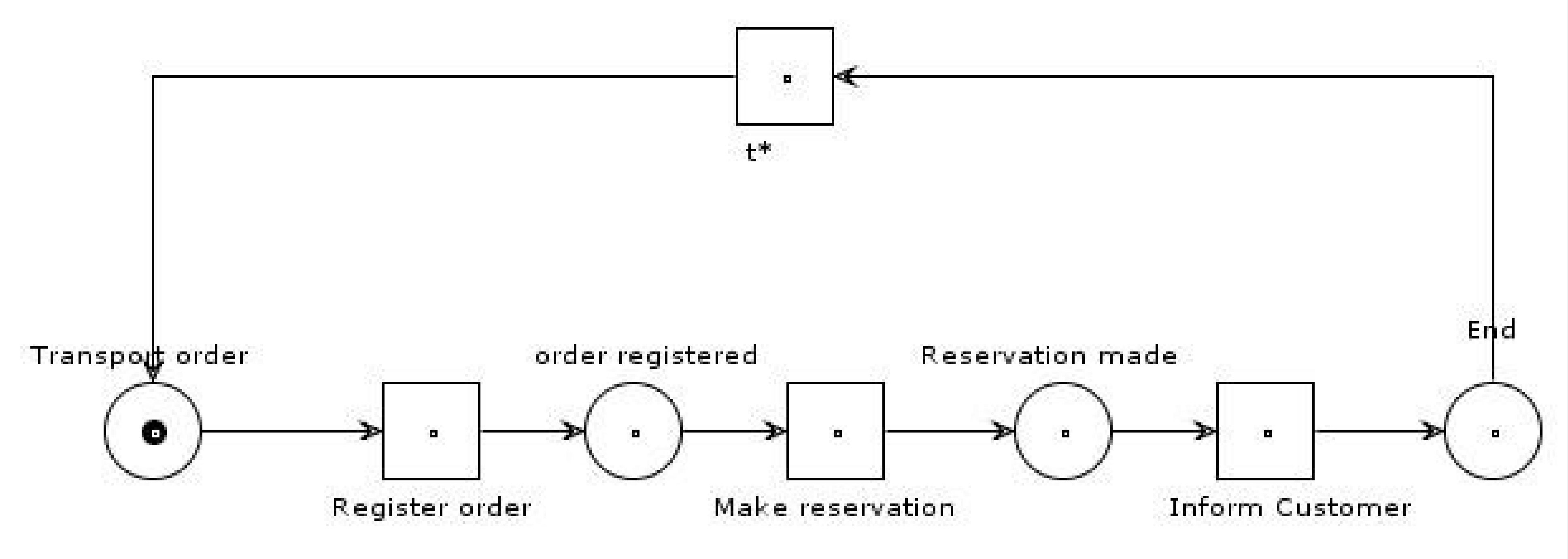
Analysis of WF-nets

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

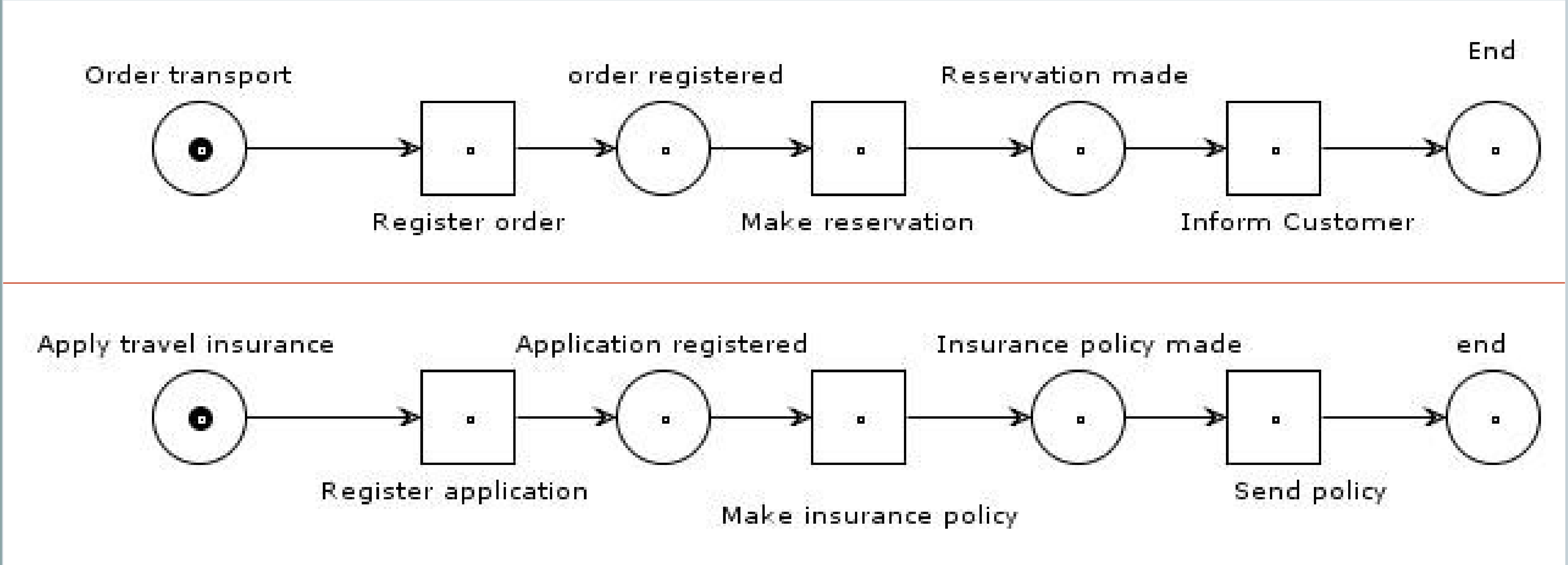


Analysis of WF-nets

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

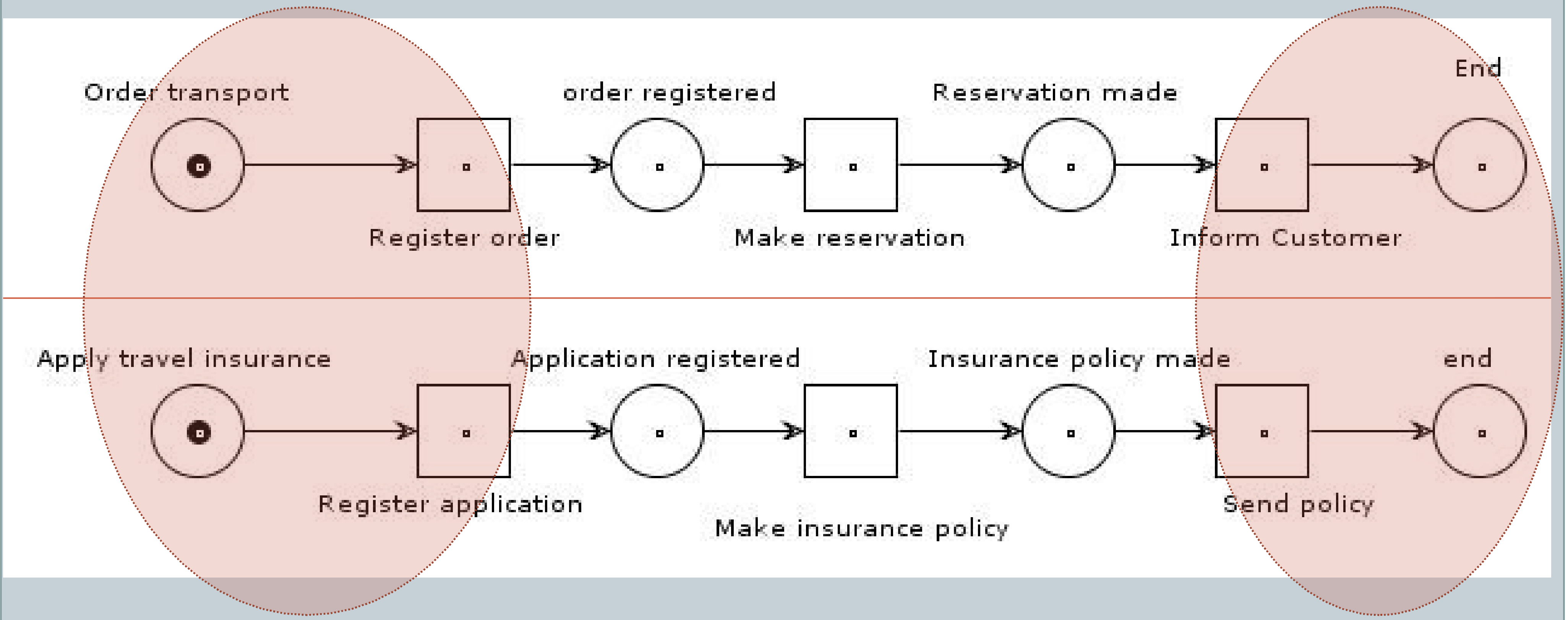


Two State machines :



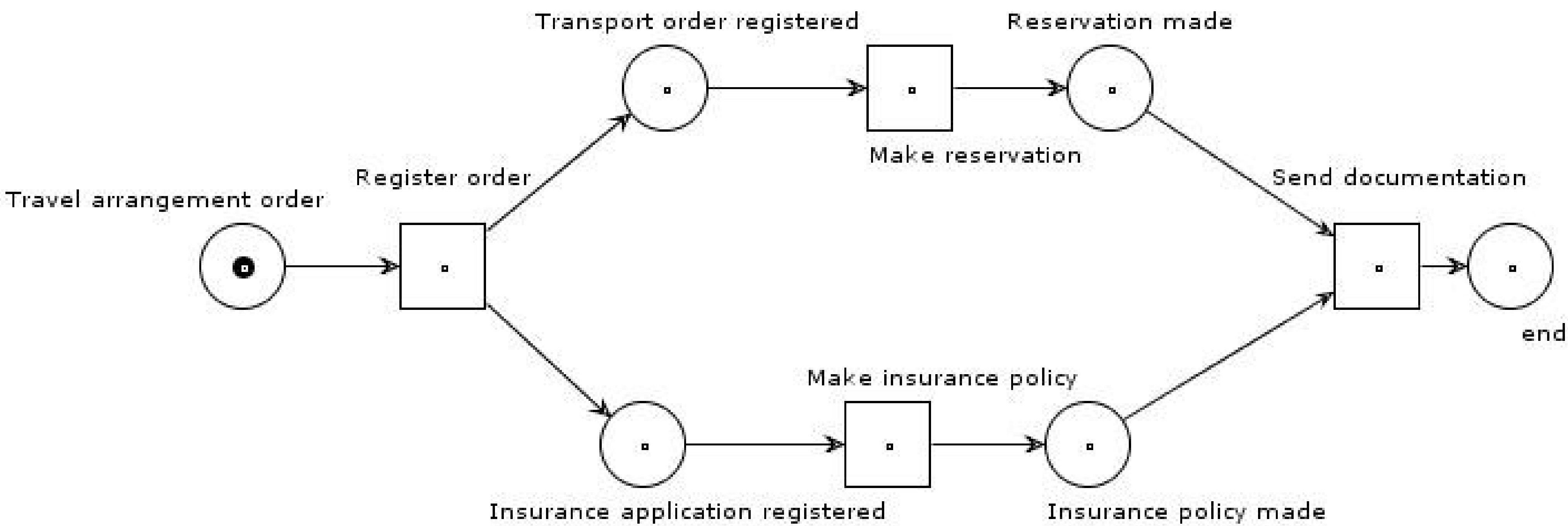
Analysis of WF-nets

Merging two State machines .....



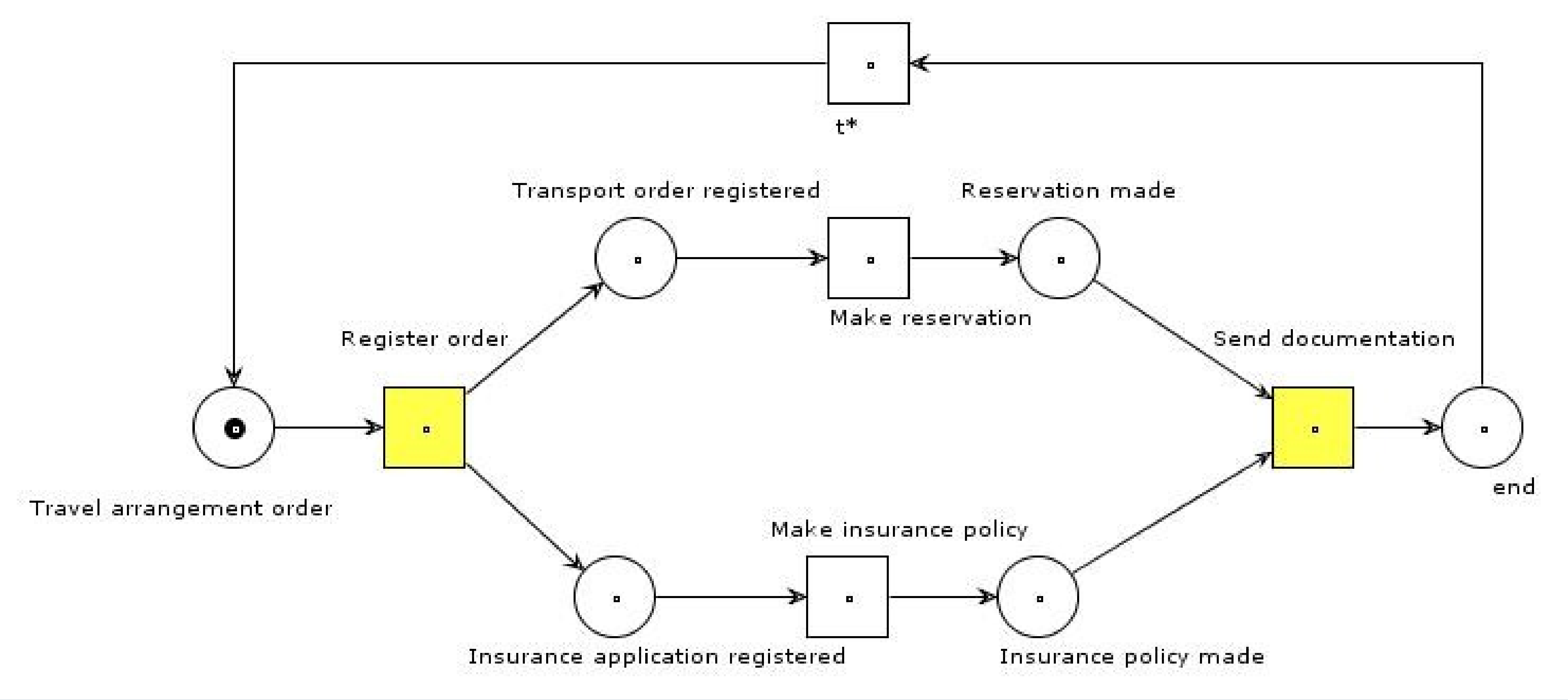
Analysis of WF-nets

Into one WF-net consisting of 2 state machines:



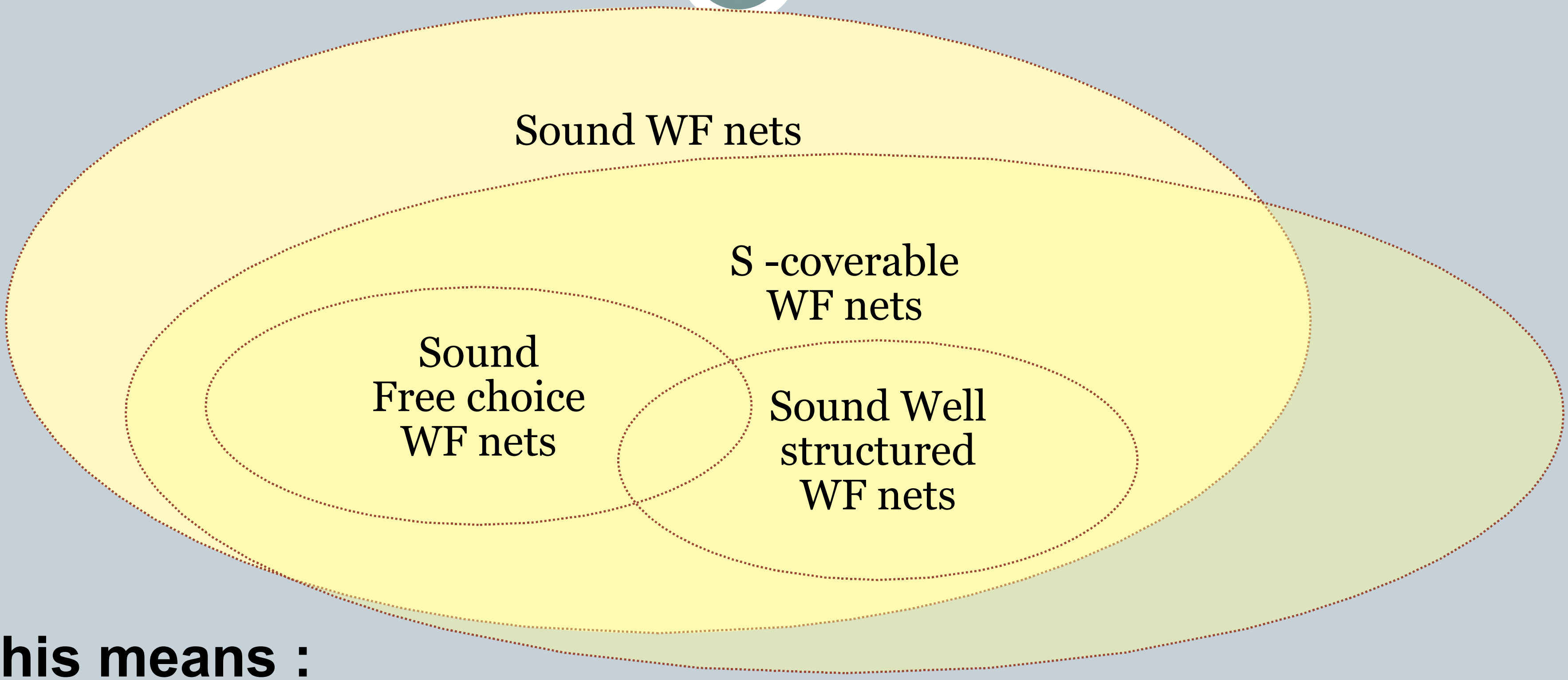
Analysis of WF-nets

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

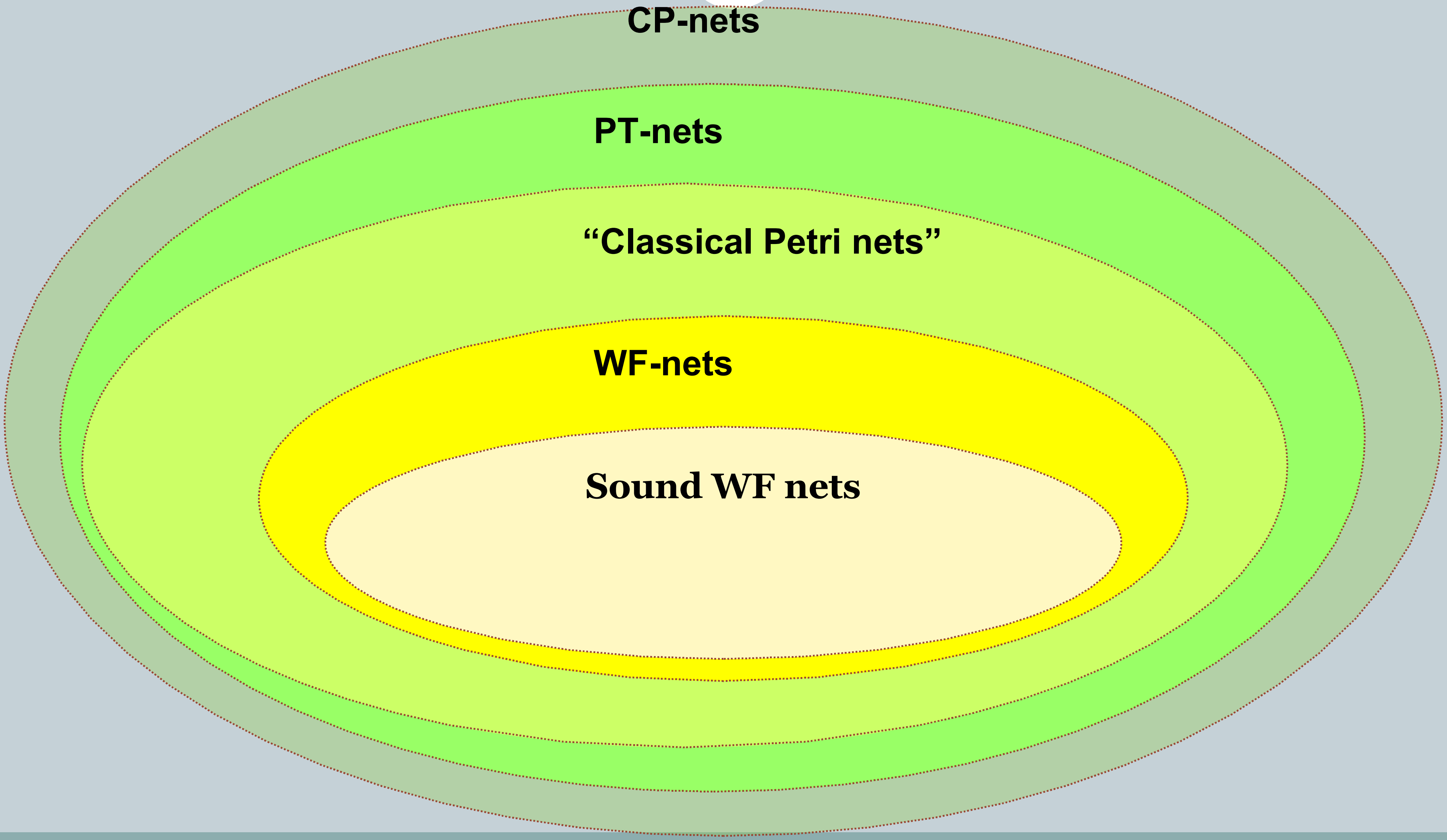
Deciding soundness for subclasses is easier!

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



Outline third lecture

- Recap second lecture
- Finish remainder of WF-net/soundness theory
- Performance analysis with Petri nets

Performance analysis

- Introduction performance analysis
- Modelling for performance analysis
  - Extension of PT-nets with time
  - Some process patterns
  - Extension of Coloured Petri nets with colour
  - Including resources in the model
- Simulation with CPN tools

Introduction performance analysis

Introduction performance analysis

- To establish that our design meets all our **requirements** and expectations, before we start construction :
  - Validity
  - Correctness
  - Performance

Introduction performance analysis

- Types of performance requirements :
  - **External** performance (end result, customer perspective, effectiveness)
  - **Internal** performance (resource use, management perspective, efficiency)

Introduction performance analysis

- Indicators external performance:
  - Completion **time**
    - Average completion time
    - Reliability of completion time
    - Meeting Deadlines
  - Other
    - Quality
    - Service
    - ……

Introduction performance analysis

- Indicators internal performance :
  - Resource use (per case) :
    - Labour (Man hours per case)
    - Capital (Machine hours per case)
    - Capacity utilization (Number of cases per time unit)
    - Raw materials (weight/volume per case)

Introduction performance analysis

- Conclusion : time plays a critical role in performance analysis, eg. :
  - The ***duration*** of an ***activity***
  - The ***completion time*** of processing a ***case***
  - The ***output*** or ***throughput*** of a process in number of cases ***per time unit***
  - Resource utilization (% idle time)

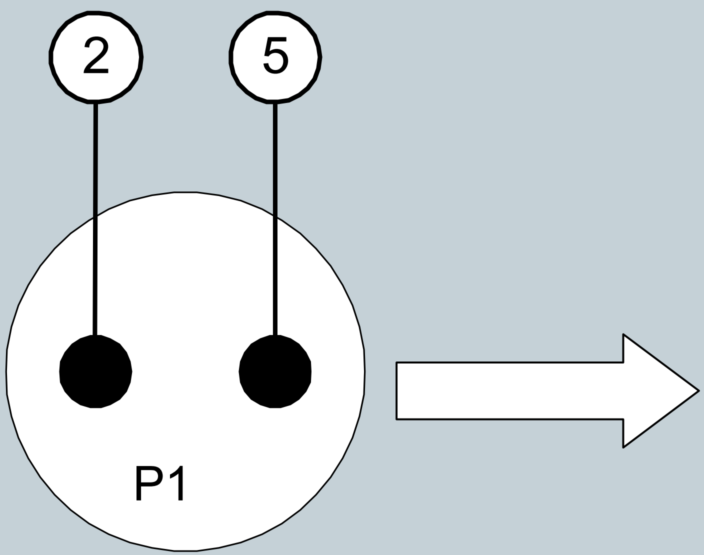
- **We need to include time into our Petri net modelling language**

Performance analysis

- Introduction performance analysis
- Modelling for performance analysis
  - Extension of Coloured Petri nets with time
  - Some process patterns
  - Extension of Coloured Petri nets with colour
  - Including resources in the model
- Simulation with Petri nets

Extension of Petri nets with time

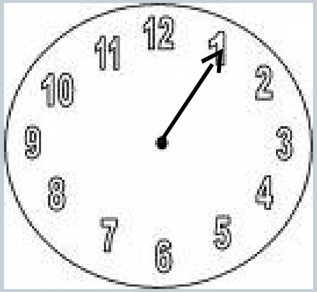
- Each token gets a **timestamp**.
- The timestamp specifies the **earliest time** when it can be consumed.



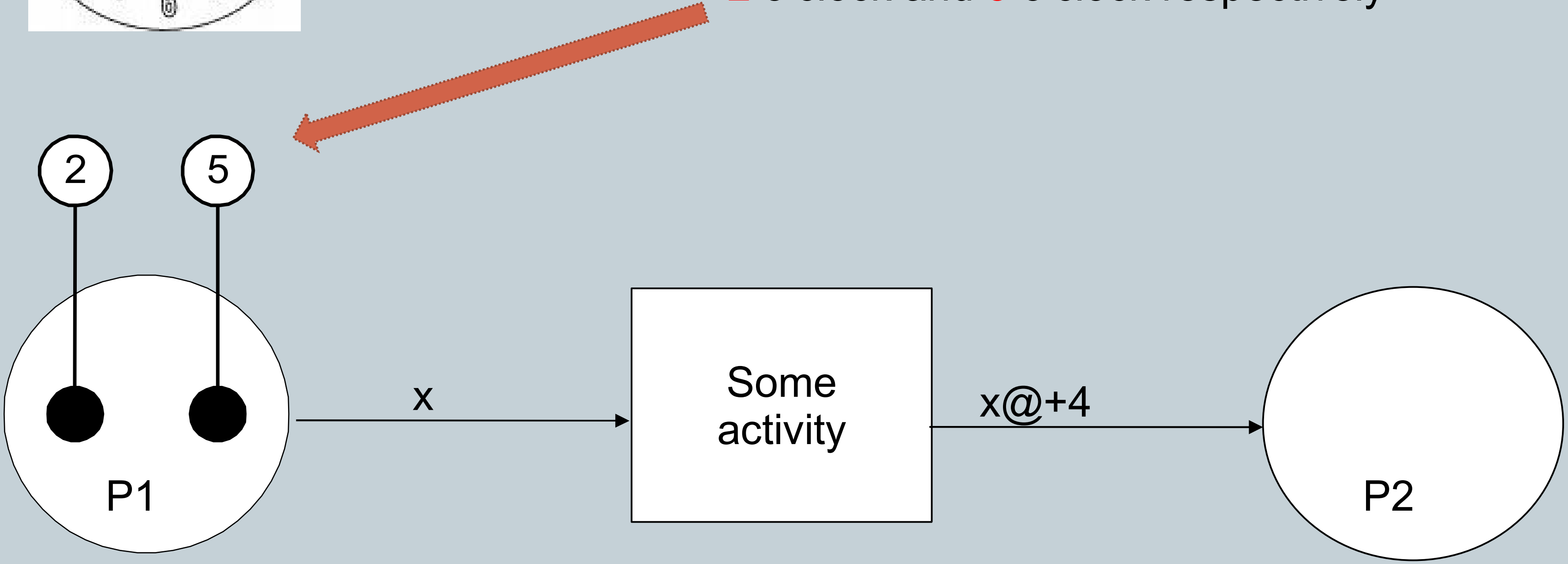
Declarations in CPN Tools

(* Standard declarations *)
colset UNIT = unit **timed;**
colset INT = int **timed;**
colset BOOL = bool **timed;**
colset STRING = string **timed**;

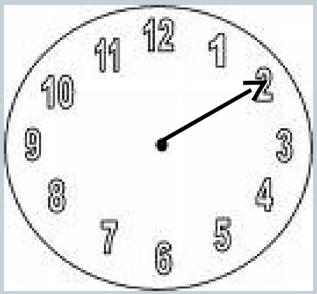
Extension of Petri nets with time

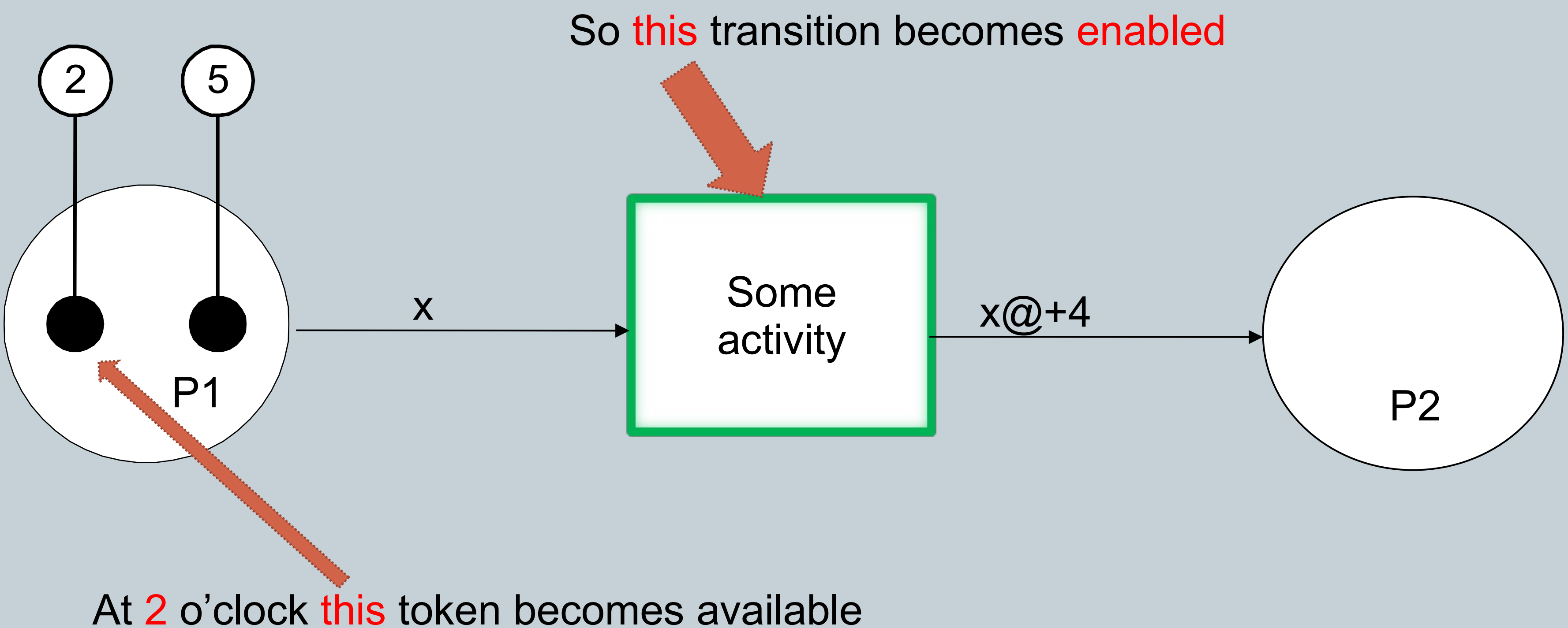


Tokens will become available at
2 o'clock and 5 o'clock respectively

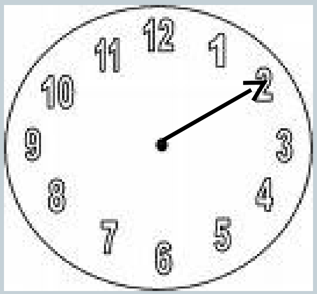


Extension of Petri nets with time



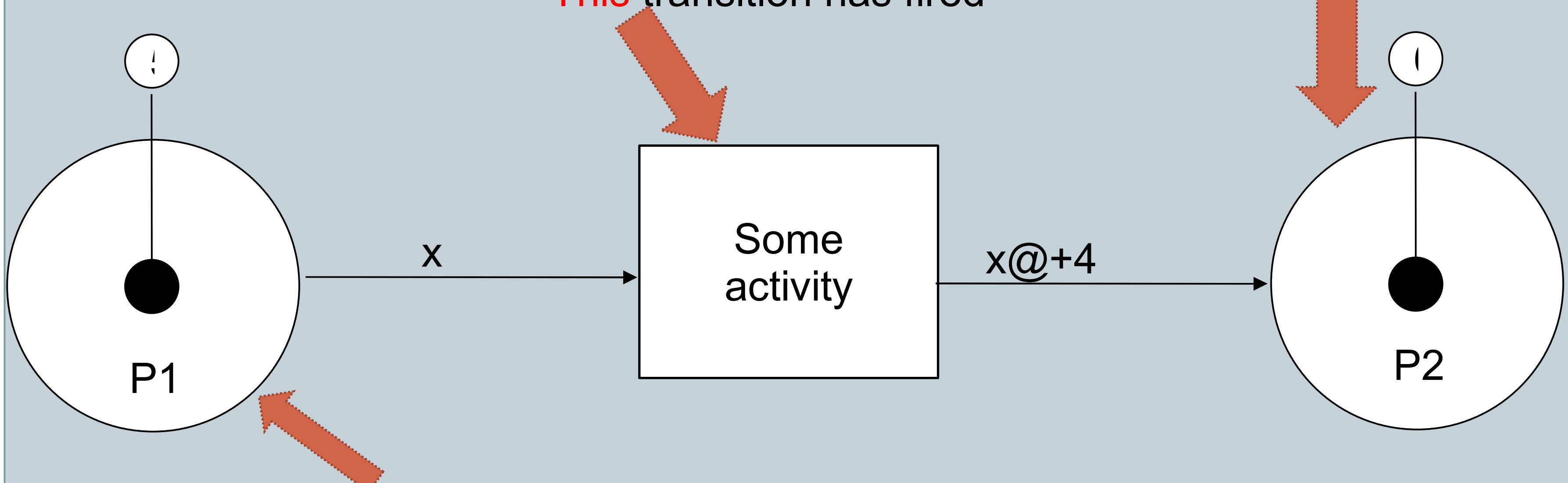


Extension of Petri nets with time



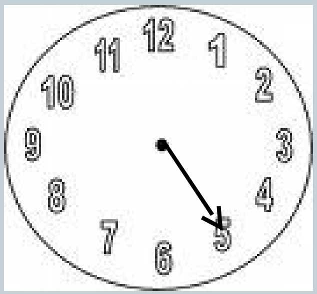
The transition has produced one token with a timestamp (2+4)=6 in this place

This transition has fired

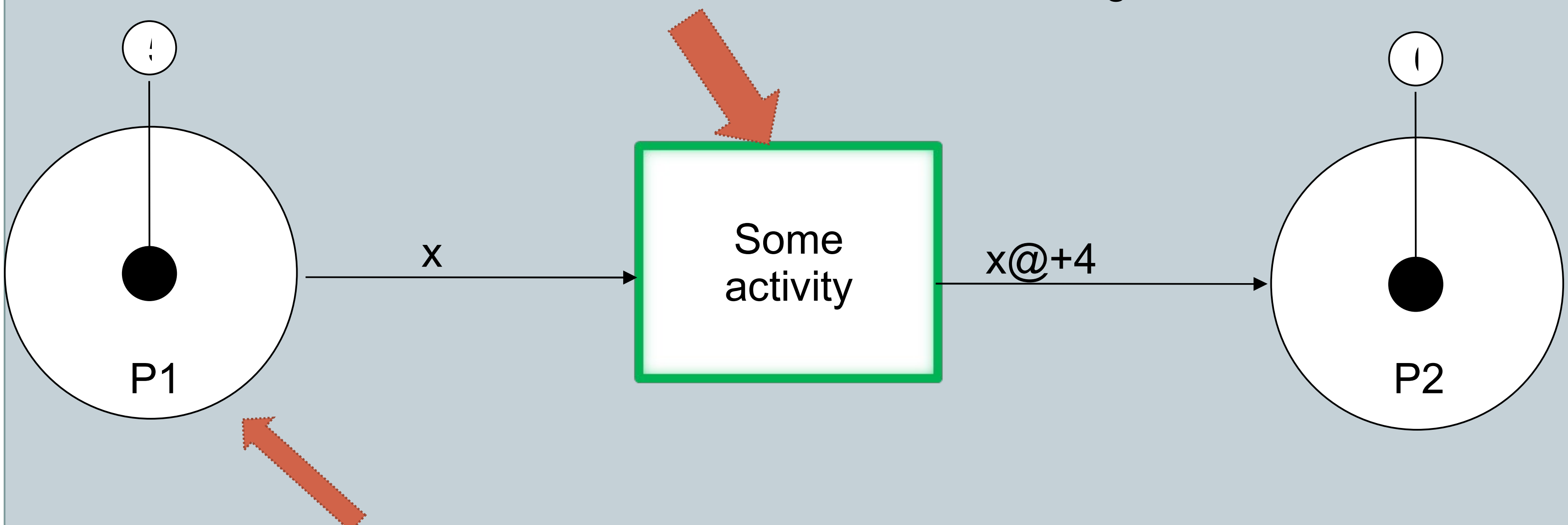


The transition has consumed one token with timestamp 2 from this place

Extension of Petri nets with time

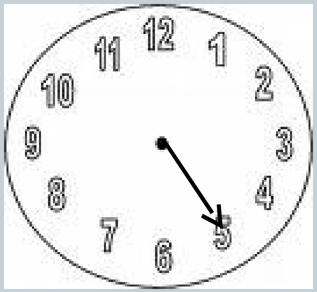


So this transition becomes again enabled



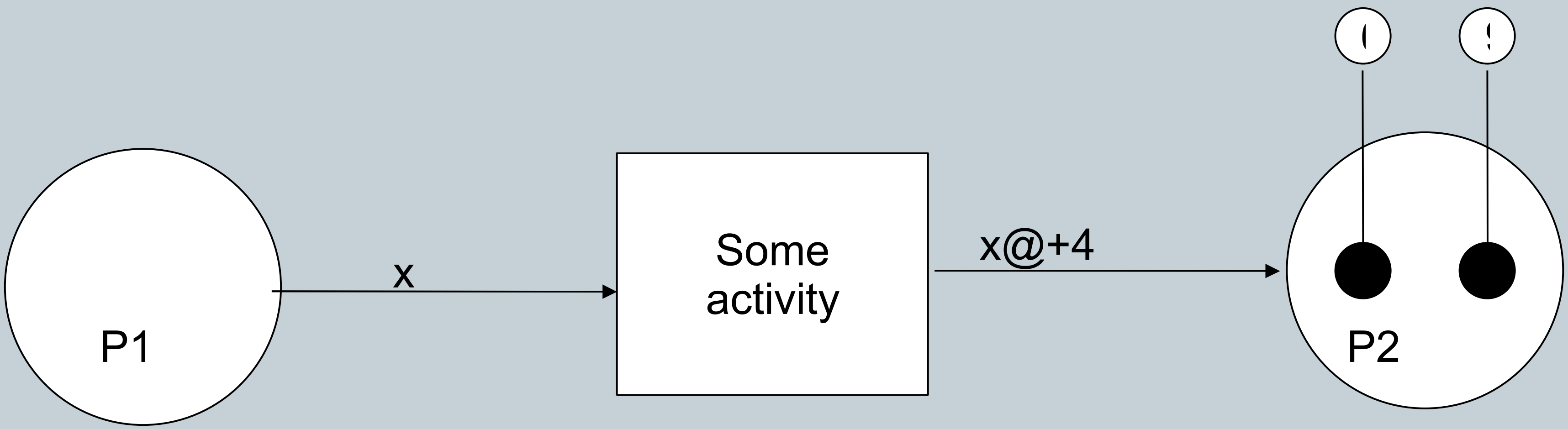
At 5 o'clock this token becomes available

Extension of Petri nets with time



The situation after firing a second time

The transition has produced another token with a timestamp (5+4)=9



So the time stamps in P2 are all 4 hours later then in P1

This means that the duration of "some activity" is 4 hours

This is due to the time clause x@+4 on the outgoing arc

Extension of Petri nets with time

- New elements :
  - *Time stamps* attached to *tokens*
    - *A token becomes **available** for consumption by a transition from the moment of its timestamp*
  - *A global clock models the passing of time*
  - Rules for *enabling* of transitions
    - A transition is enabled if ***enough*** tokens are ***available***
  - ***Time clauses* to calculate the time stamps**
    - **The time stamps produced are equal to the *firing time* plus the *delay* specifed by the time clause**

Extension of Petri nets with time

- Analyzing completion time of a business process with timed Petri nets :
  - The ***completion time*** of a process depends on duration of each activity and the routing of the case through each activity in the process
  - We can use timed Petri nets to analyze and quantify the effect of both

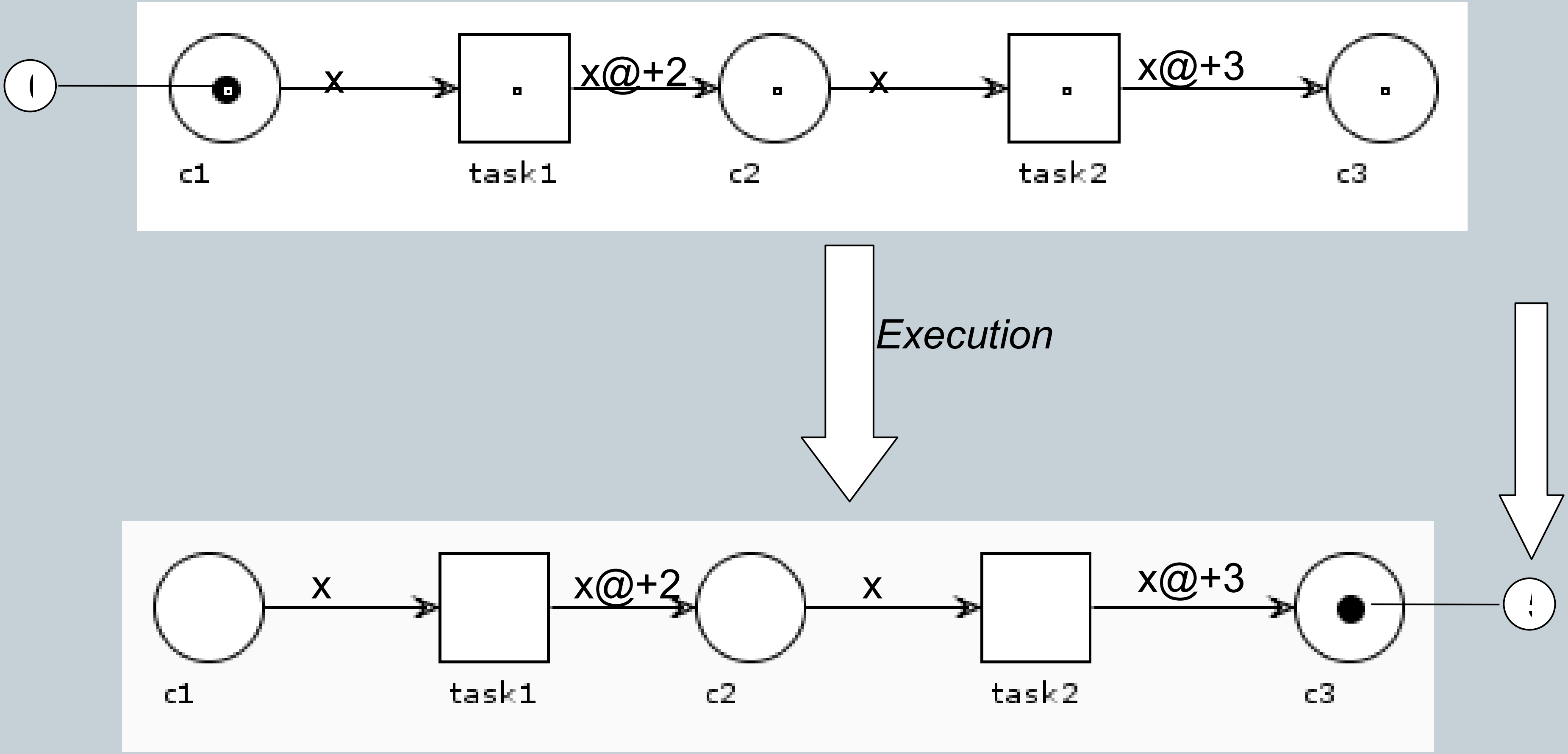
Extension of Petri nets with time

- Example : a process with two tasks
  - Task 1 : 2 minutes (eg. Have a phone call)
  - Task 2 : 3 minutes (eg. Have a short walk)

- Analyze the difference between sequential and parallel routing

Extension of Petri nets with time

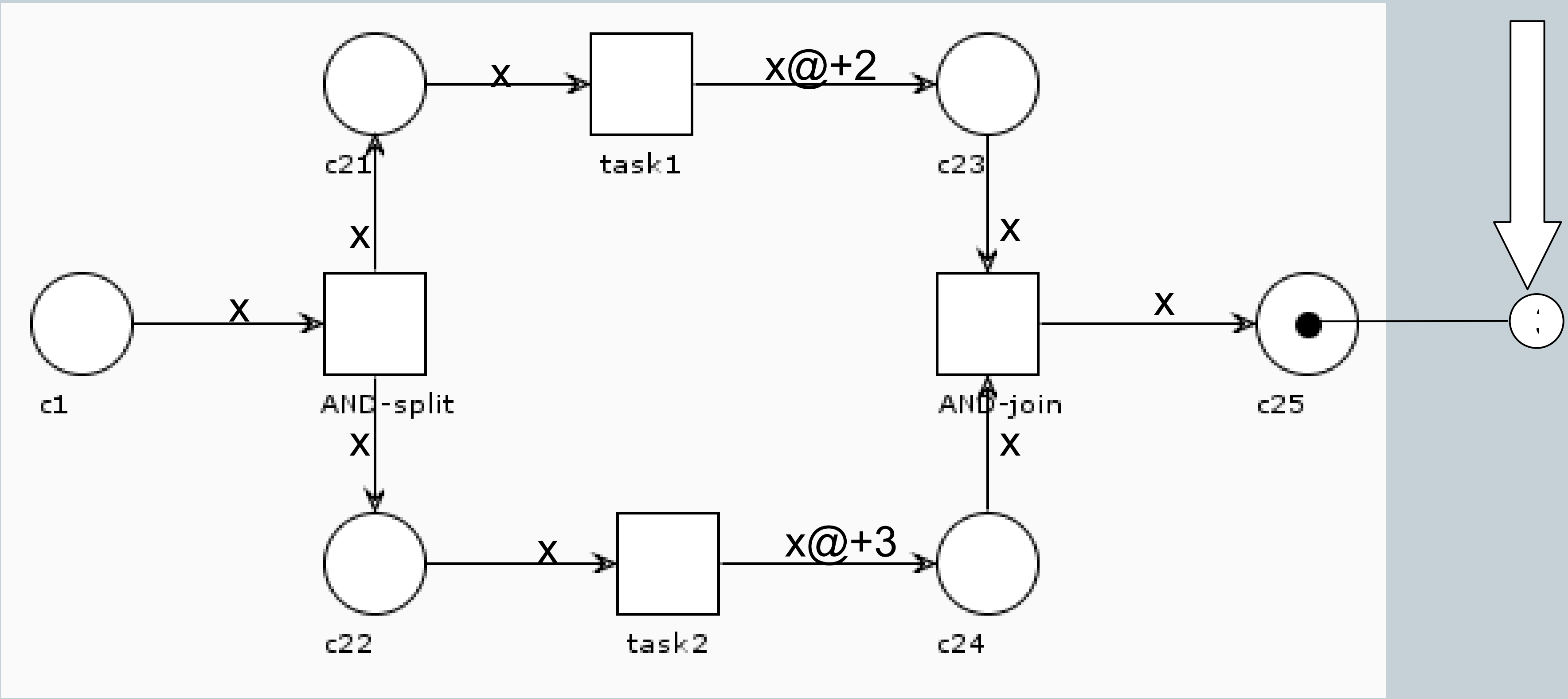
- Example sequential routing



Completion time takes **5 time units** (eg.minutes) for each case

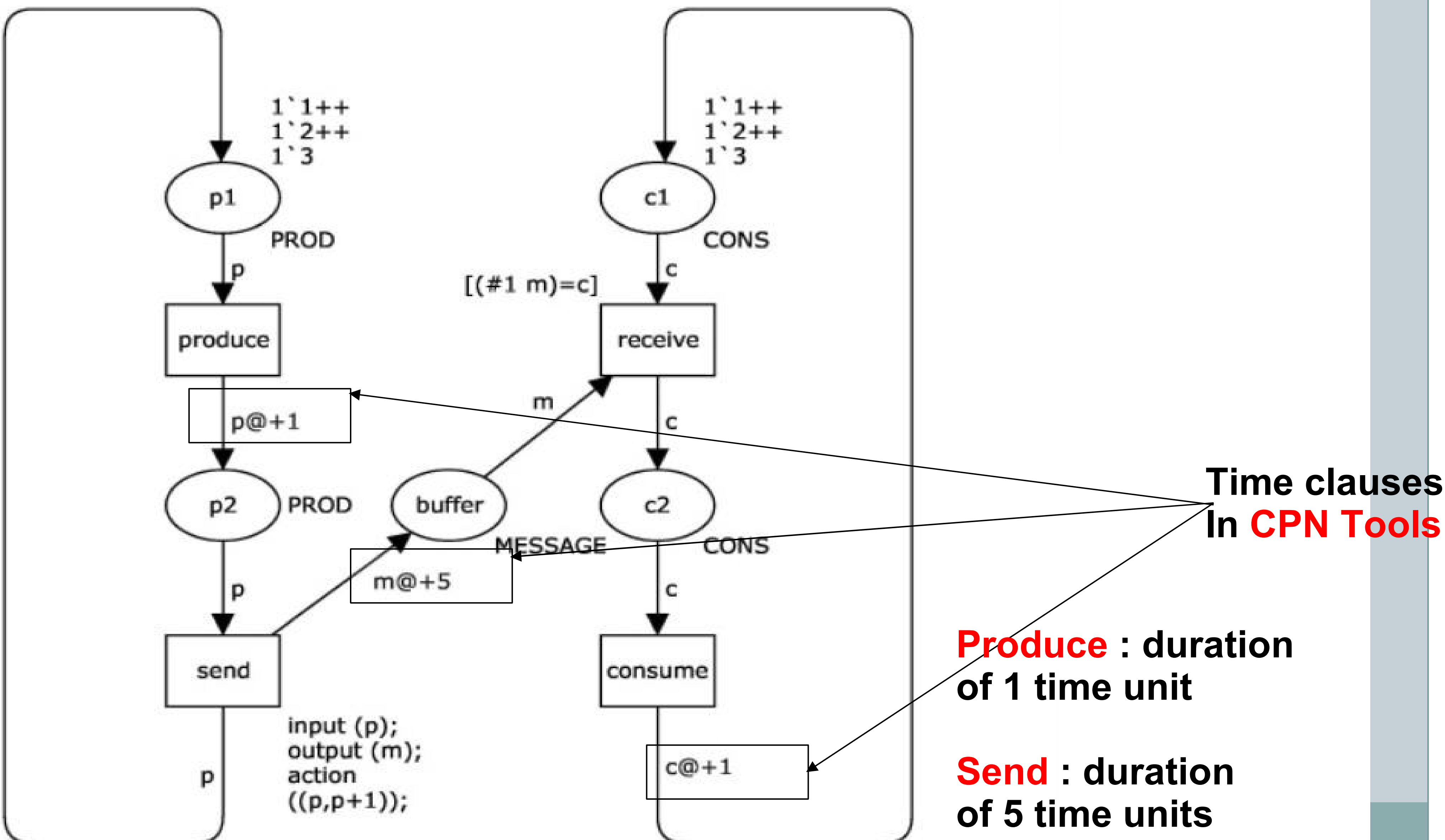
Extension of Petri nets with time

- Example Parallel routing :



Completion time would take 3 minutes for each case

Extension of Petri nets with time



Produce : duration of 1 time unit

Send : duration of 5 time units

Performance analysis

- Introduction performance analysis
- Modelling for performance analysis
  - Extension of Coloured Petri nets with time
  - Some process patterns
  - Extension of Coloured Petri nets with colour
  - Including resources in the model
- Simulation with Petri nets

Process patterns

- Best practices (van der Aalst p. 120-125):
  - Paralellization
  - Combination of tasks
  - Resource flexibilization (*)
  - Triage (*)

Process patterns

- As you have seen, tokens in a Petri net can be used to model the "cases" handled by a business process

- The route in a business process might depend on the ***case attributes***, ***eg.***
  - ***Triage in handling insurance claims : big claims are handled differently from small claims***

- However, the tokens in the EN system and PT system are indistinguishable, so we have a problem….

Outline of this lecture

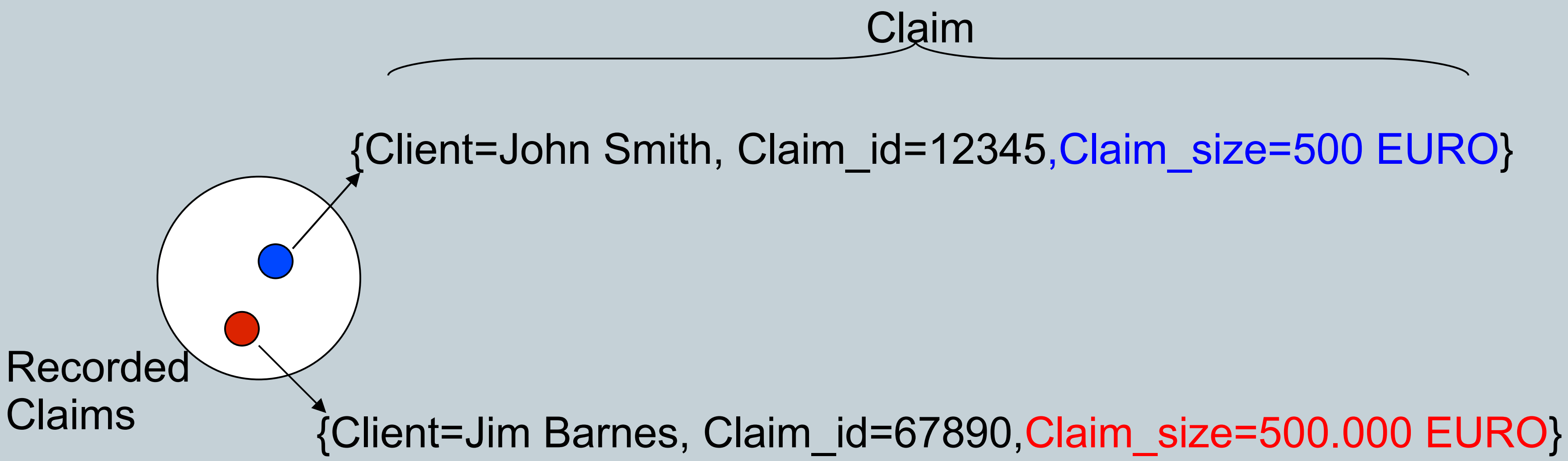
- Introduction performance analysis
- Modelling for performance analysis
  - Extension of Petri nets with time
  - Some process patterns
  - Extension of Petri nets with colour
  - Including resources in the model
- Simulation with Petri nets

Extension of Petri nets with colour

- Processing of insurance claims :
  - Big claims (> EURO 500) must be investigated and assessed by expert before paying out compensation
  - Small claims (< EURO 501) can be paid out without further investigation

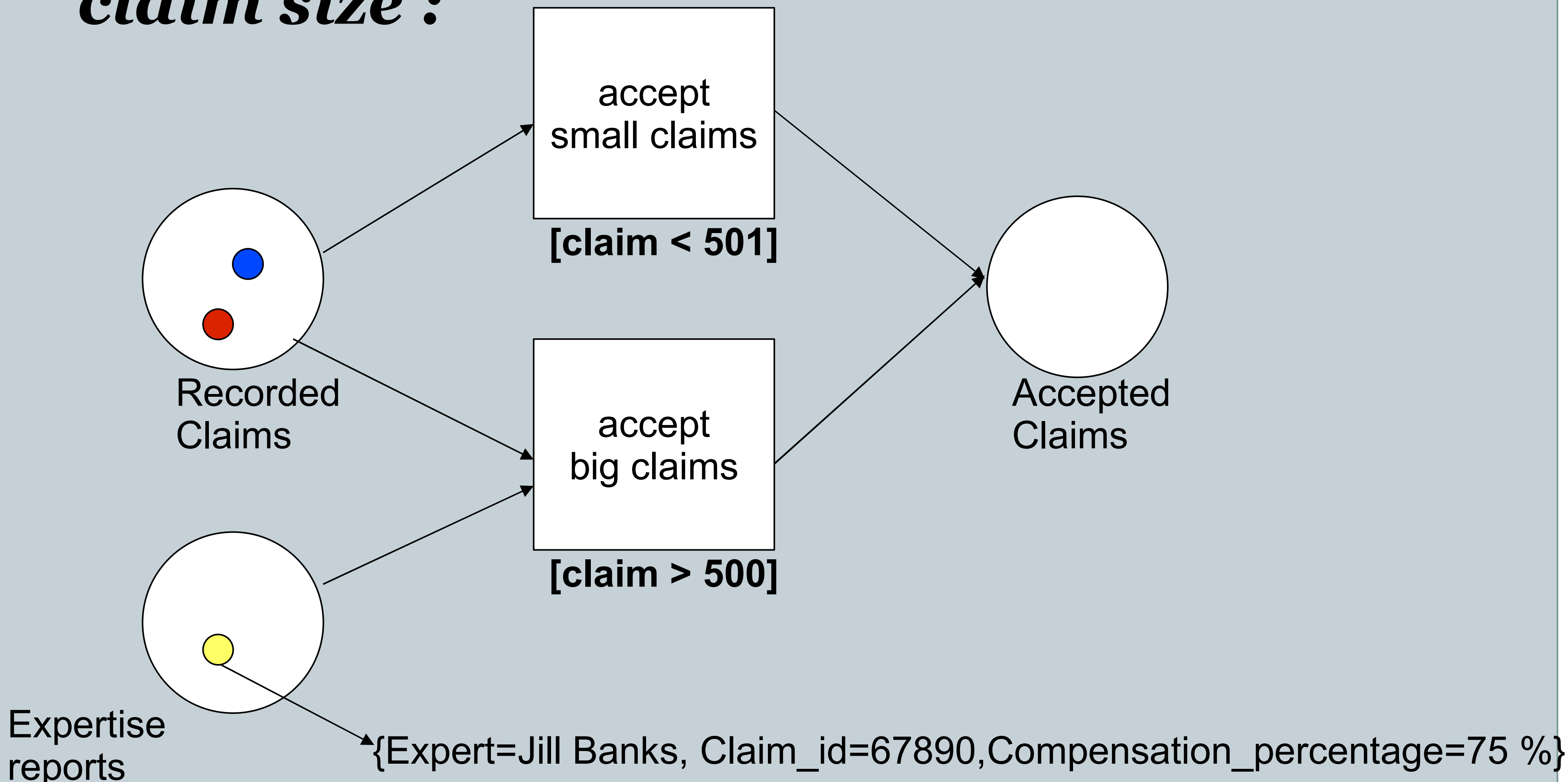
Extension of Petri nets with colour

- Give each ***token*** a ***value***, *for instance the size of the insurance claim*



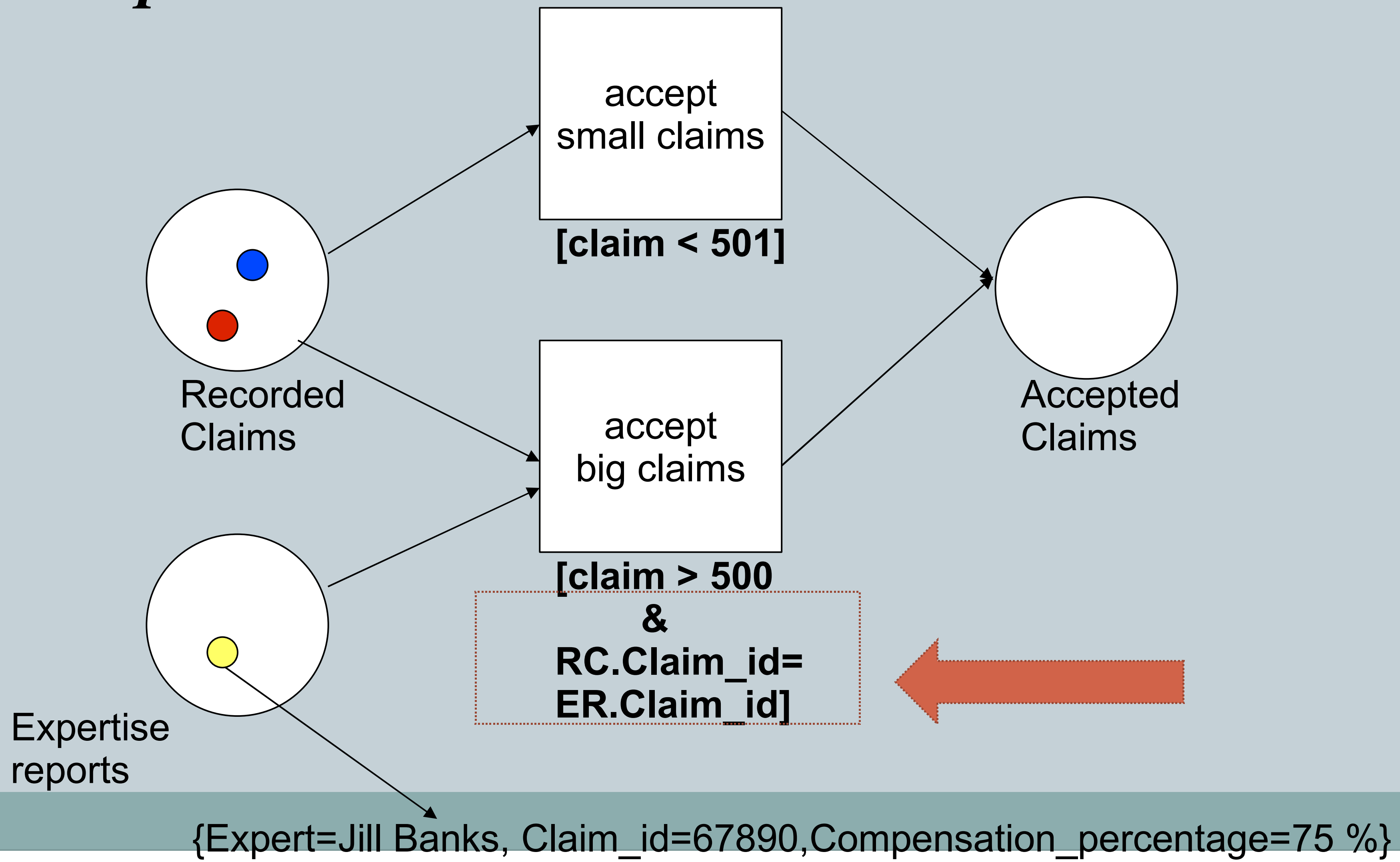
Extension of Petri nets with colour

- ***And a guard to make routing dependent on claim size :***



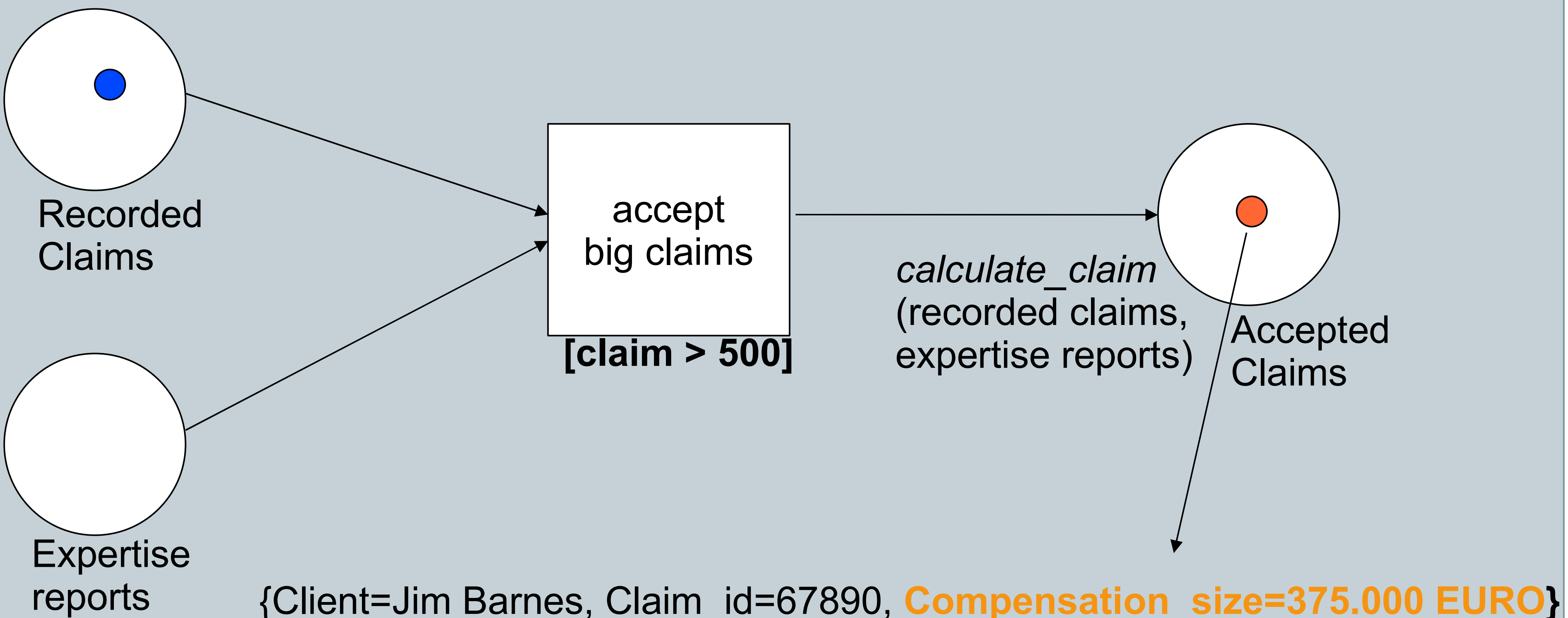
Extension of Petri nets with colour

- ***We would also need a guard to match "Expertise reports" to "Claims"***



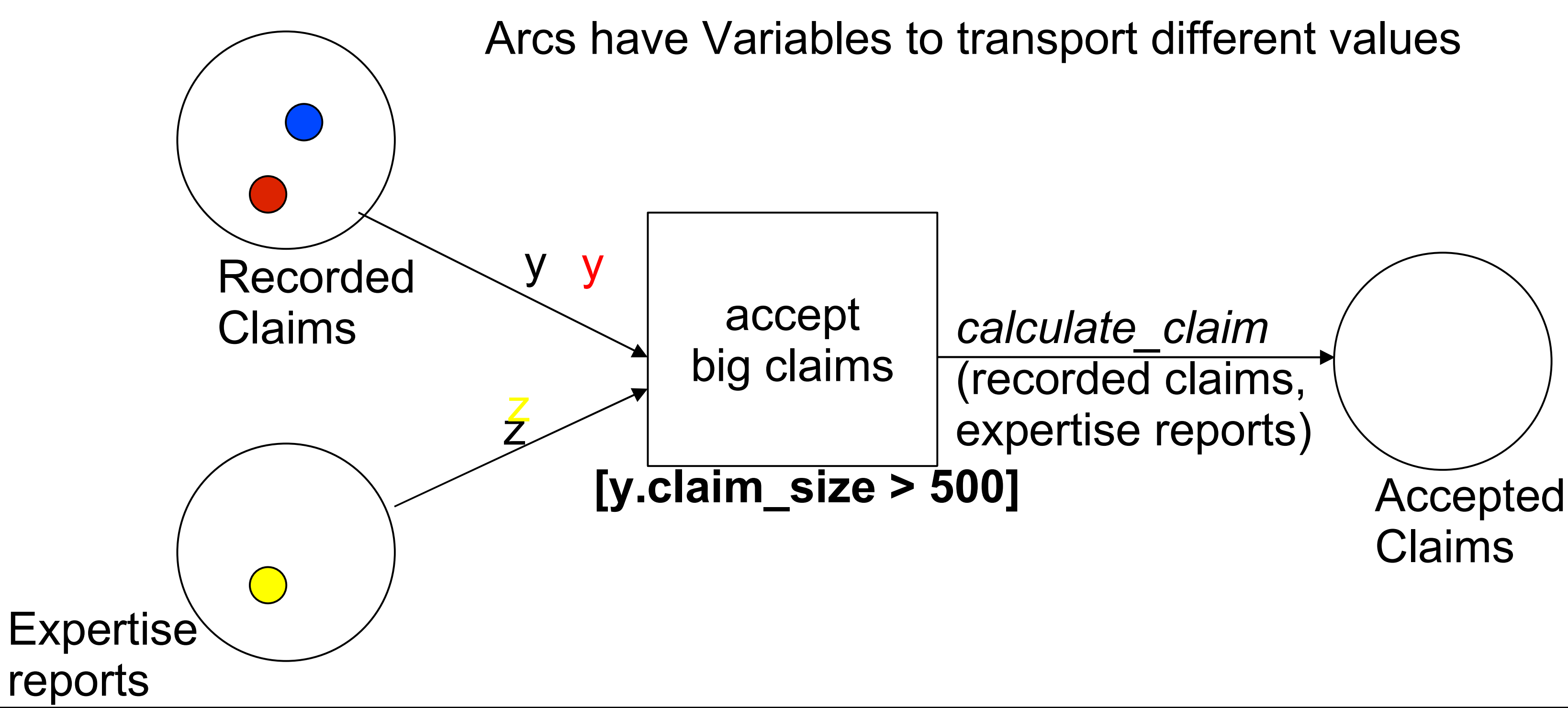
Extension of Petri nets with colour

- ***arc expression or* function *to calculate output tokens from input tokens***



Extension of Petri nets with colour

Bindings for "accept big claims"



Extension of Petri nets with colour

- *Before a transition is enabled, all variables on input and output arcs are bound to a value:*

- *A binding will only occur if it matches the guard*

Extension of Petri nets with colour

- 3 producers (id=1,2 and 3)
- Each can produce messages containing some data (a number)
- 3 consumers (id=1,2 and 3)
- Each consumer can only consume messages of producer with the same id

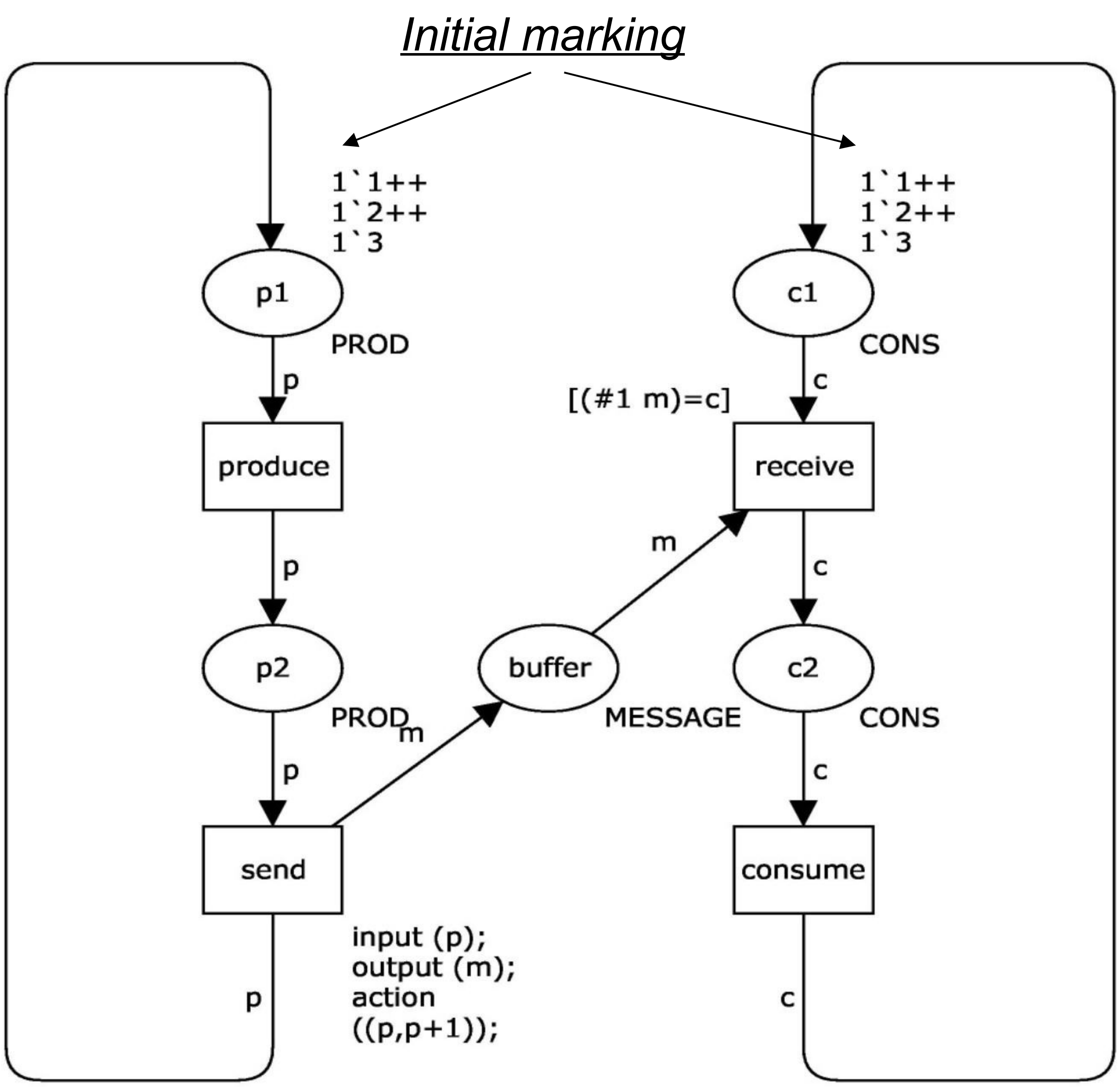
Extension of Petri nets with colour

- Colourset PROD = Integer
- Colourset DATA = Integer
- Colourset MESSAGE = PROD * DATA
- Colourset CONS = Integer

So for example :

Producer 1 (element of PROD) can produce a message (1,32) (element of MESSAGE) that can be consumed by consumer 1 (element of CONS)

Extension of Petri nets with colour



PROD, CONS, DATA are *coloursets* of type Integer

MESSAGE is colourset of type PROD*DATA

SEND produces tokens of type MESSAGE with *output function* : (p,p+1)

RECEIVE consumes a token of type MESSAGE provided it matches the *guard* (token in place c1=first element in token in the buffer)

Performance analysis

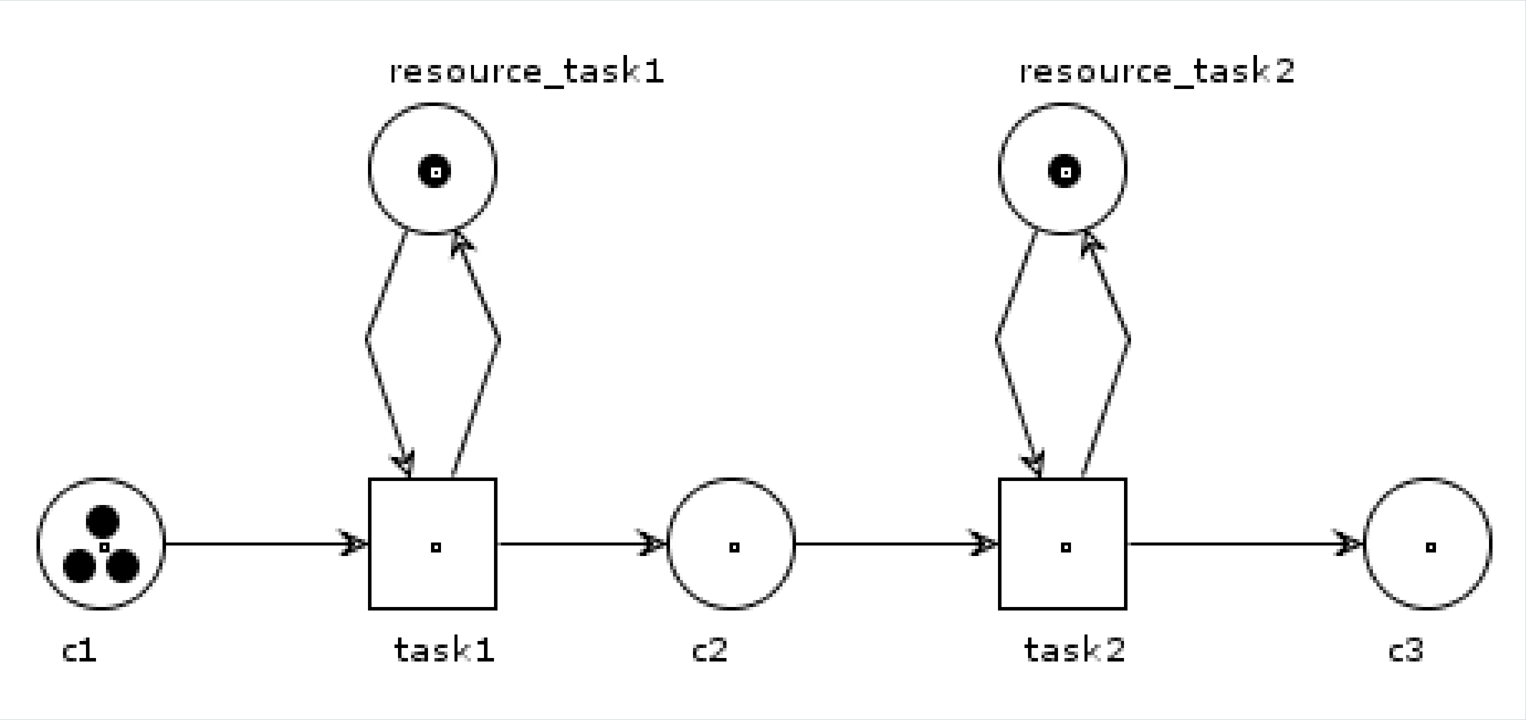
- Introduction performance analysis
- Modelling for performance analysis
  - Extension of Petri nets with time
  - Some process patterns
  - Extension of Petri nets with colour
  - Including resources in the model
- Simulation with Petri nets

Including resources in the model

- Not only routing but also **resource availability** will influence completion time:
  - If a task can not be executed due to resources that are unavailable, the case must ***wait en thus completion time will increase***
- ***So we should also model resources in our Petri net, so how do we do that?***

Including resources in the model

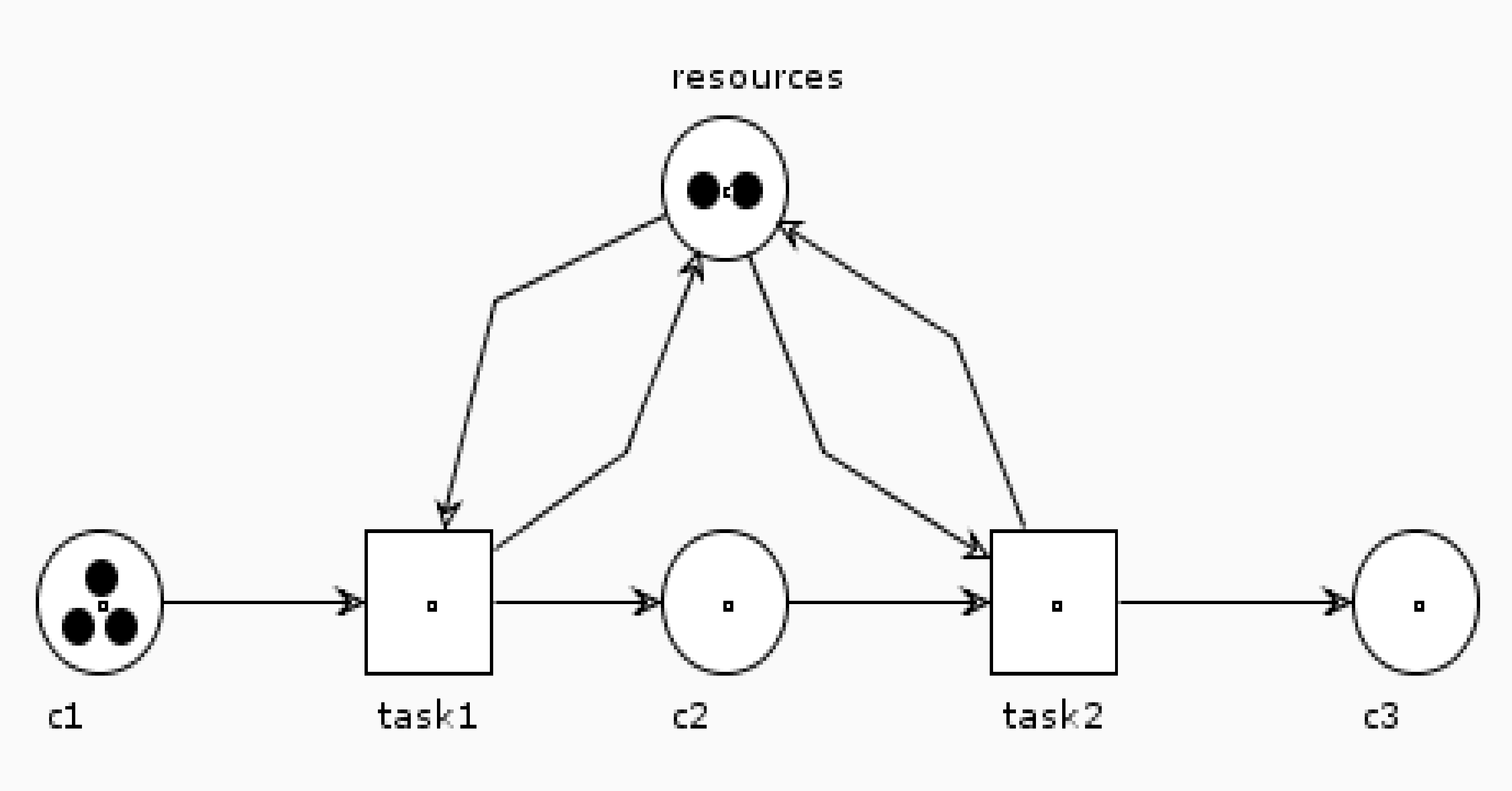
A simpel sequential process with two dedicated resouces



Resource task 1 has 3 cases to process and
Resource task 2 has nothing to do

Including resources in the model

A simpel sequential process with resouces *flexible* enough to perform all tasks by merging the resource places



Now, as long as there are cases, the resources are always utilized

Outline of this lecture

- Introduction performance analysis
- Modelling for performance analysis
  - Extension of Petri nets with time
  - Some process patterns
  - Extension of Petri nets with colour
  - Including resources in the model
- Simulation with Petri nets

Simulation with Petri nets

- Different approaches :
  - Markovian analysis
  - Queuing theory
  - **Simulation**
    - Generally applicable to analysis of workflows
    - No mathematical sophistication required
    - Also useful tool for validation

Timed coloured petri nets are well suited for running simulations

Simulation with Petri nets

- Simulation:
  - Execute the model
  - Gather data produced during execution of the model
  - Calculate performance indicators based on gathered data
  - Compare calculated performance with performance requirements

Simulation with Petri nets

- Model is composed of :
  - Model of processes in scope
  - Model of the environment
- Model of the environment
  - Approximation only by using:
  - Probablity distributions
- Executing the model is then a random walk through the reachability graph

Simulation with Petri nets

- Simulation:
  - Gather data produced during execution of the model
  - Execute the model
  - Calculate performance indicators based on gathered data
  - Compare calculated performance with performance requirements

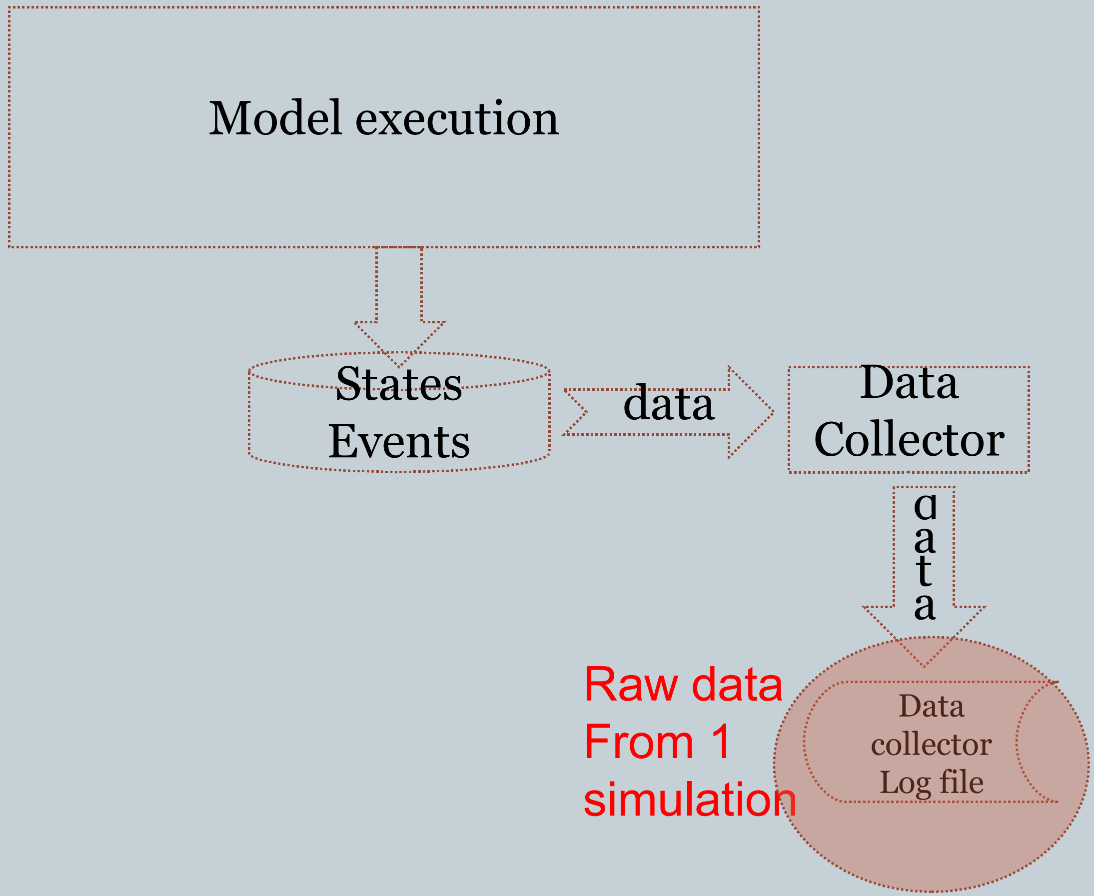
Simulation with Petri nets

- A *monitor* is a mechanism that is used to observe, inspect, control or modify a simulation of a CP-net.
- Important characteristics of monitors:
  - They can inspect the *states* and *events* of a simulation, and take appropriate actions based on the observations.
  - There is an explicit separation between monitoring the behavior of a net, and modeling the behavior of the system.

Simulation with Petri nets

- Data collection monitors for calculating performance measures, such as:
  - Expected average delay in queue
  - Expected average queue length
  - Expected utilization of the server

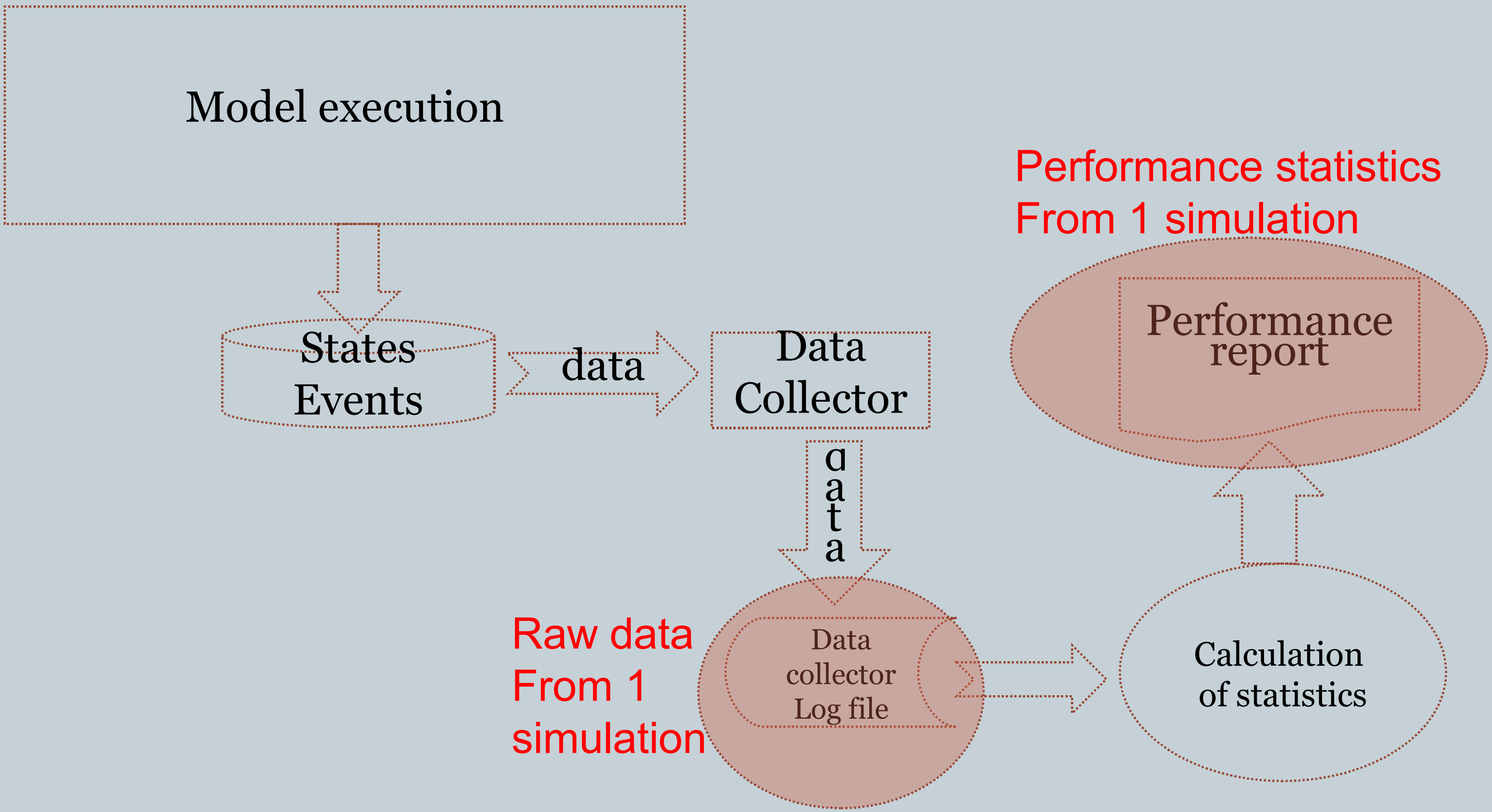
Simulation with Petri nets



Simulation with Petri nets

- Simulation:
  - Execute the model
  - Gather data produced during execution of the model
  - Calculate performance indicators based on gathered data
  - Compare calculated performance with minimum performance requirements

Simulation with Petri nets



Simulation with Petri nets

Note that these statistics have been calculated for data that is not necessarily independent or identically distributed.

Timed statistics					
Name	Count	Avrg	Min	Max	Time Interval
List_length_dc_System'Queue_1	285	5.339428	0	12	13673
Marking_size_Server'Busy_1	277	0.920427	0	1	13673
Queue_Length	285	5.339428	0	12	13673
Queue_Length_Advanced	281	5.339428	0	12	13673
Queue_Length_NoInit	284	5.562362	0	12	13125
Queue_Length_NoInitNoStop	275	5.562362	0	12	13125
Queue_Length_NoStop	276	5.339428	0	12	13673
Server_Utilization	268	0.920427	0	1	13673
Server_Utilization_Advanced	147	0.920427	0	1	13673

Untimed statistics						
Name	Count	Sum	Avrg	StD	Min	Max
Count_trans_occur_Arrivals'Arrive_1	141	141	1.000000	0.000000	1	1
Long_Delay_Times	134	122	0.910448	0.286611	0	1
Processed_A_Jobs	85	85	1.000000	0.000000	1	1
Queue_Delay	134	70580	526.716418	243.416584	0	1082
Queue_Delay_IntInf	134	70580	526.716418	243.416584	0	1082
Queue_Delay_Real	134	70580.000000	526.716418	243.416584	0.000000	1082.000000
Server_Utilization_Estimate_by_ProcTime	9	7.517767	0.835307	0.112929	0.579436	0.949755
Server_Utilization_by_ProcTime	9	7.420739	0.824527	0.110175	0.579436	0.920427
Total_Processing_Time	134	12986.000000	96.910448	106.917751	1.000000	602.000000

Simulation with Petri nets

- Improving reliability by calculating statistics based on data gathered from multiple executions
- If the subruns are assumed to be mutually independent, one can calculate a **confidence interval** ➔ **see exercise “Queue system configuration”**

Performance analysis with Petri nets

- CPN tools
  - More powerful then woped
  - Woped is specifically taylored for workflow definition and analysis with Classical Petri nets, CPN tools is more generic but therefore also a bit more difficult to use.
- Exercises
  - 1.1. Getting acquainted with CPN tools
  - 1.2. The assembly line
  - 1.3. Advanced features : is an optional exercise
  - 1.4. Running case