

# Business Processes Modelling

## MPB (6 cfu, 295AA)

Roberto Bruni

<http://www.di.unipi.it/~bruni>

14 - Analysis of WF nets

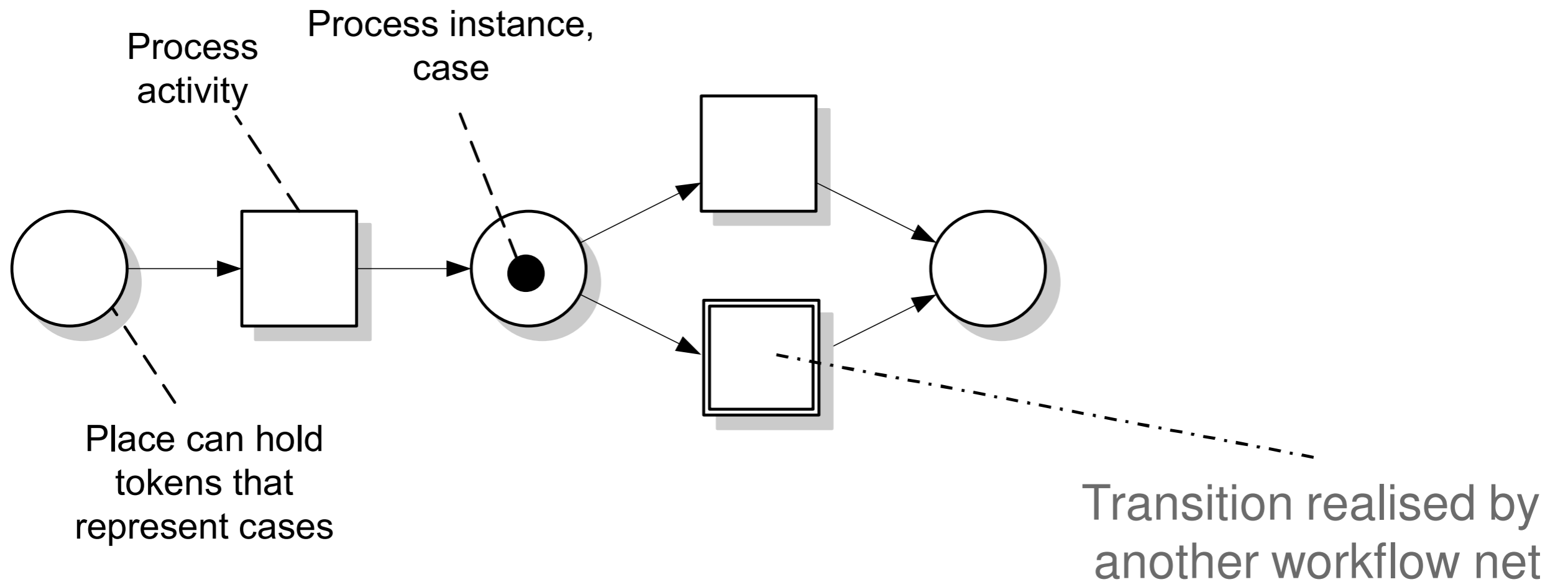


# Object

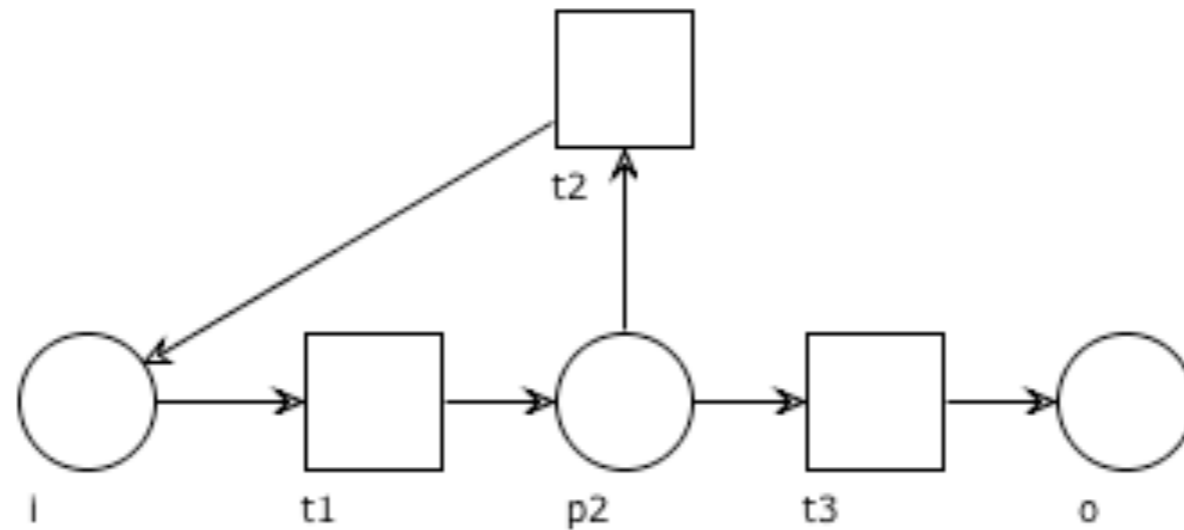


We study suitable soundness properties  
of Workflow nets

# WF nets as business processes



# Structural analysis



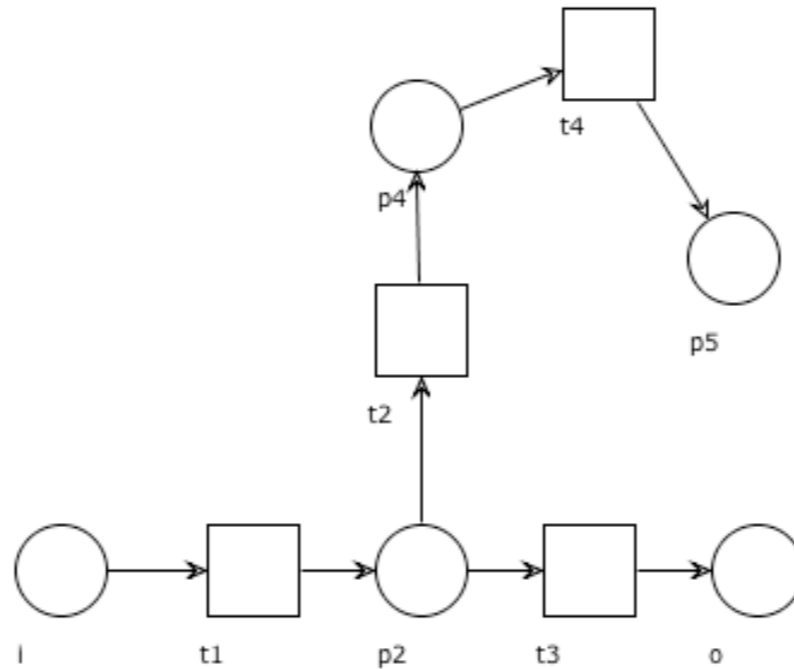
No distinguished entry / exit point

**no entry**: when should the case start?

**no exit**: when should the case end?

**not a workflow net!**

# Structural analysis



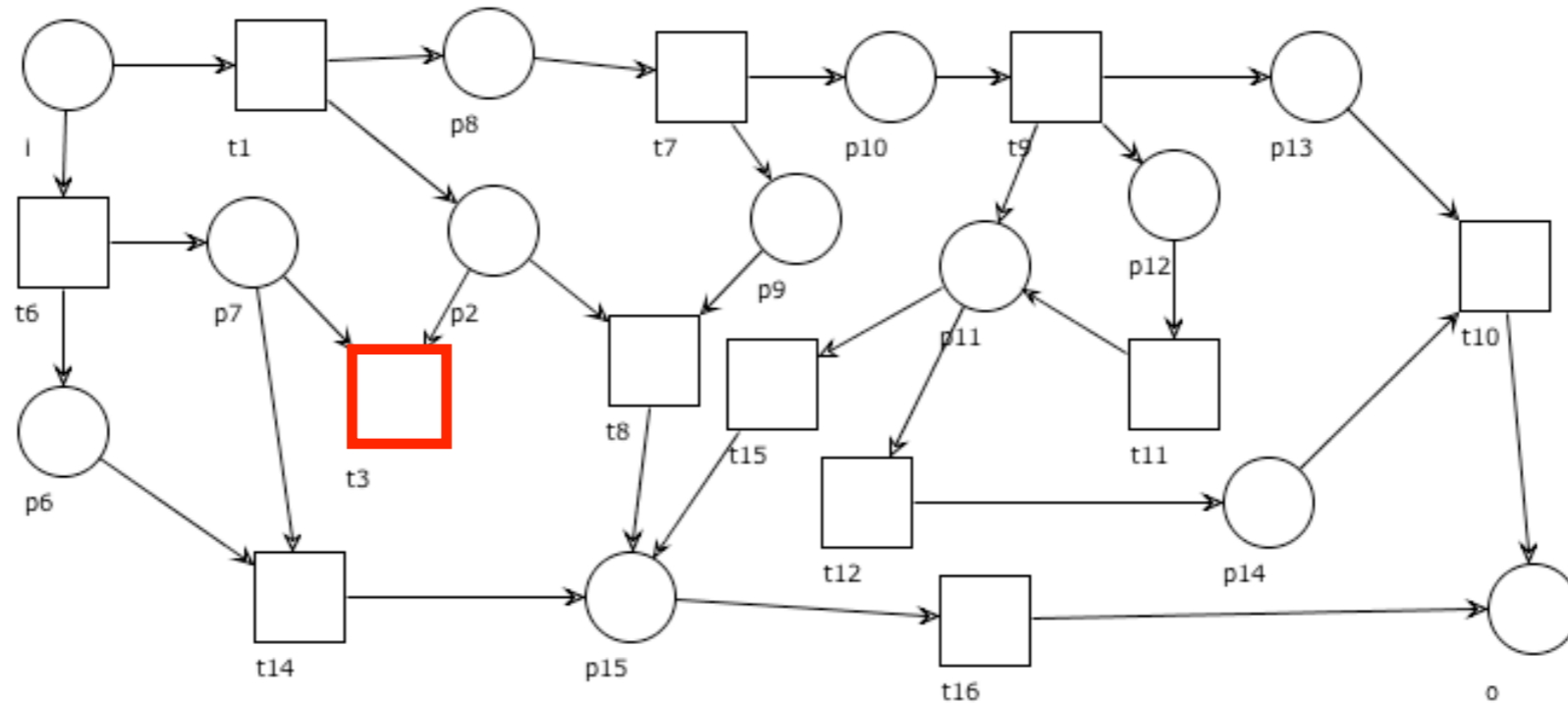
Multiple entry / exit points

**multiple entries:** when should the case start?

**multiple exit:** when should the case end?

**not a workflow net!**

# Structural analysis



Tasks t without incoming and/or outgoing arcs

**no input:** when should t be carried out?

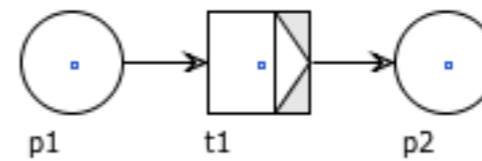
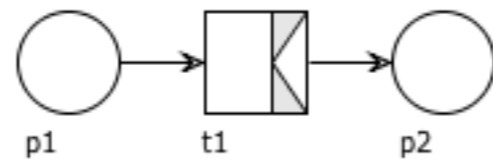
**no output:** t does not contribute to case completion

**not a workflow net!**

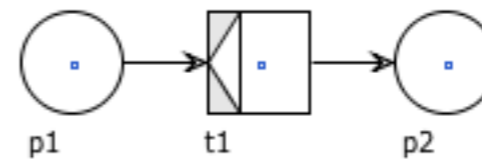
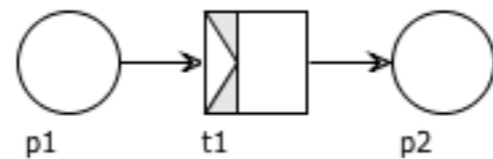
# Structural analysis

Wrong decorations of transitions

split with only one outgoing arc



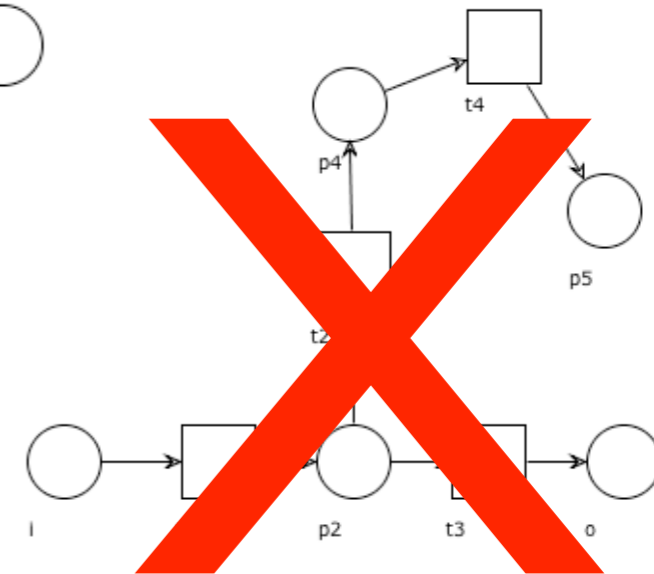
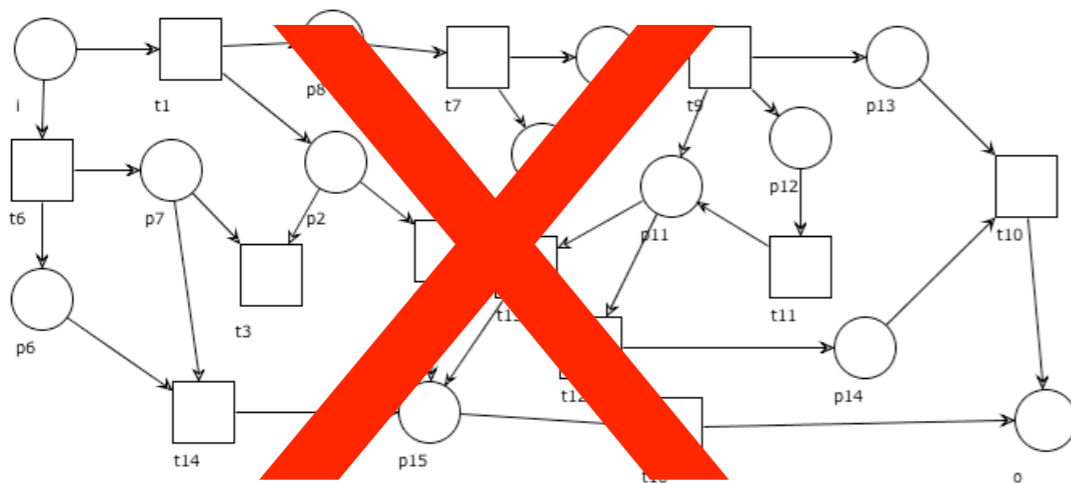
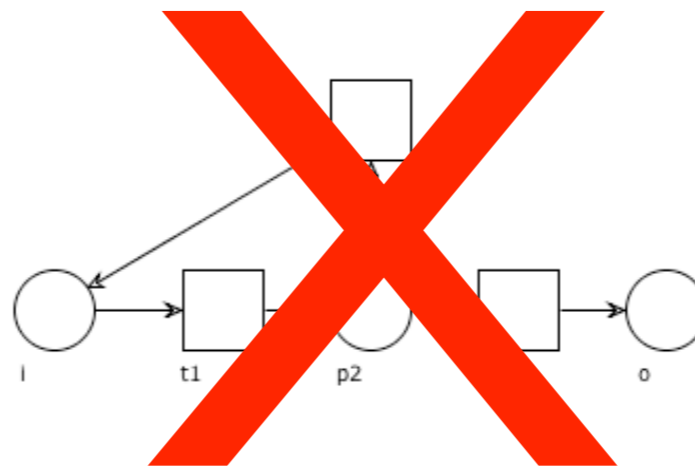
join with only one incoming arc



**non-sense: left to designer responsibility**

# Structural analysis

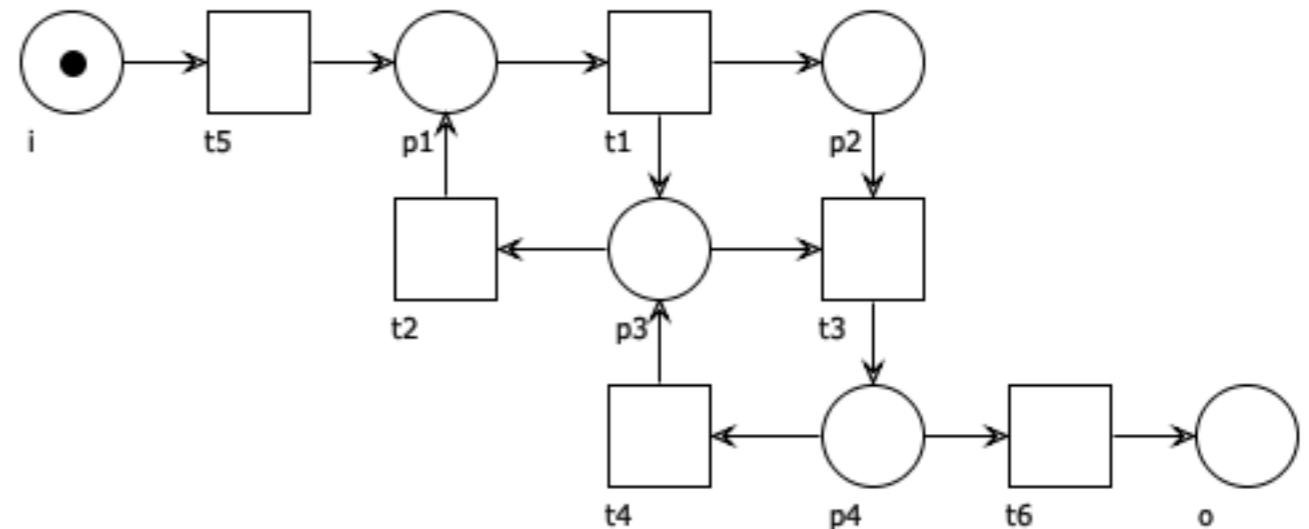
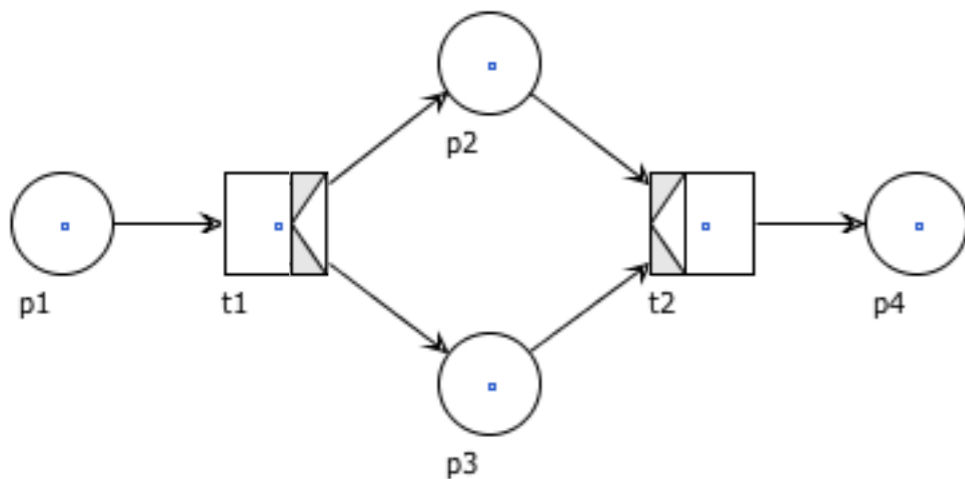
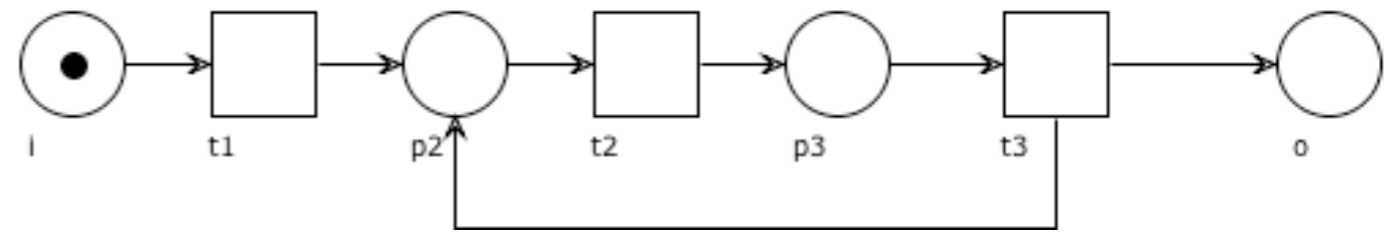
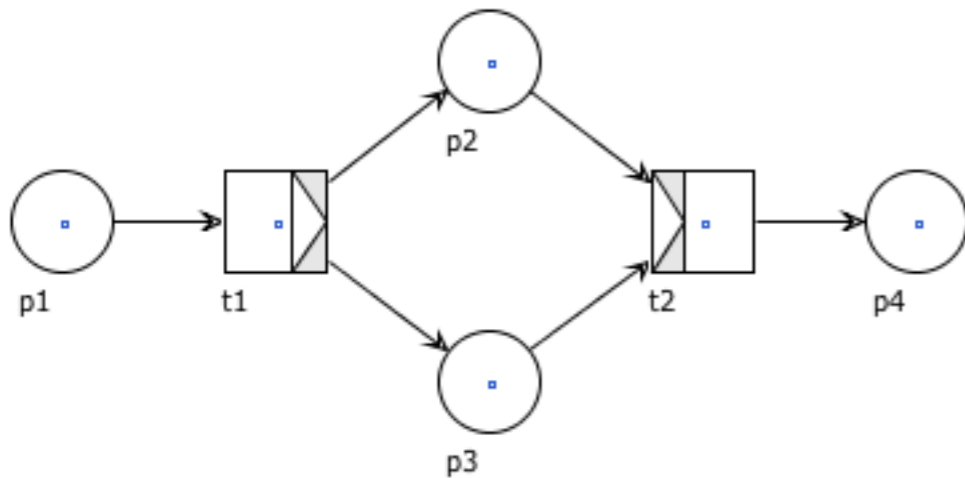
The definition of Workflow nets is purely structural but already rules out many erroneous models





# Behavioural analysis

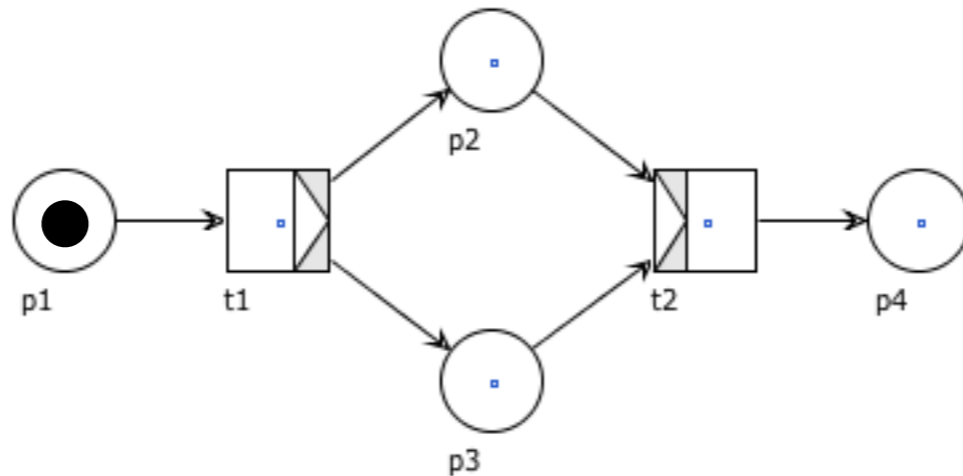
Still many problematic workflow nets  
can be defined...



# Activity analysis

## Dead tasks

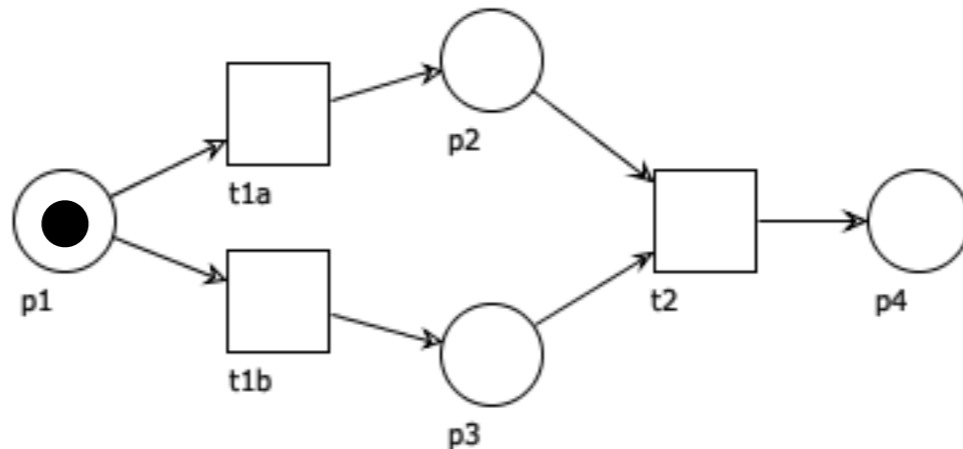
Tasks that can never be carried out  
(each transitions lies on a path from i to o: **not sufficient**)



# Activity analysis

## Dead tasks

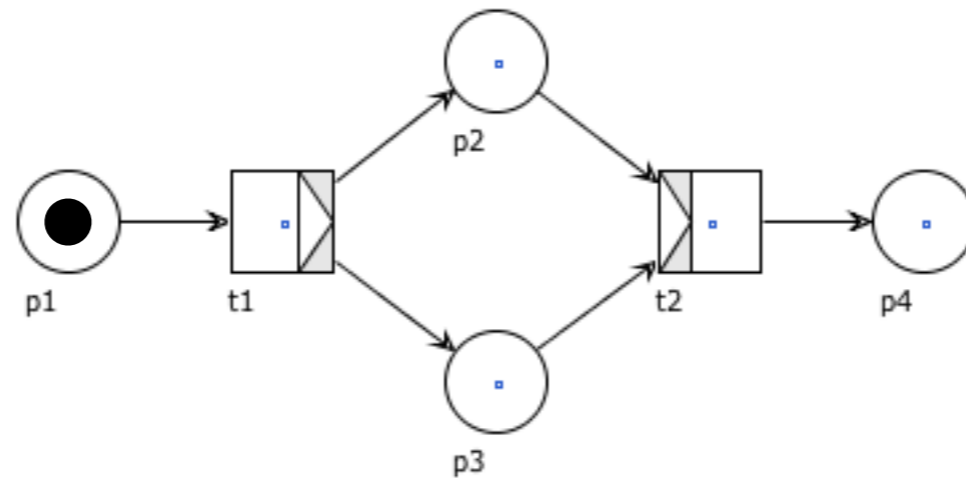
Tasks that can never be carried out  
(each transitions lies on a path from i to o: **not sufficient**)



**workflow nets can contain dead tasks!**

# Net analysis

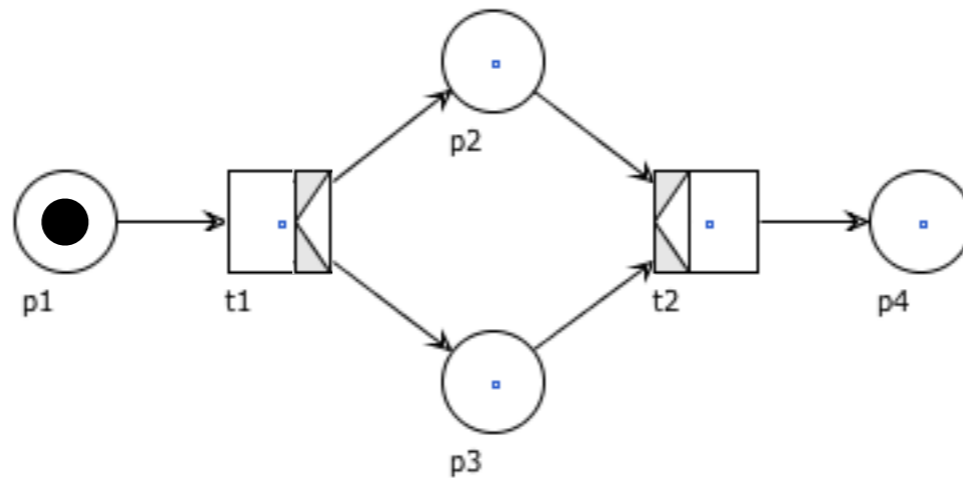
## Deadlock



a case blocks without coming to an end  
**can arise in workflow nets**

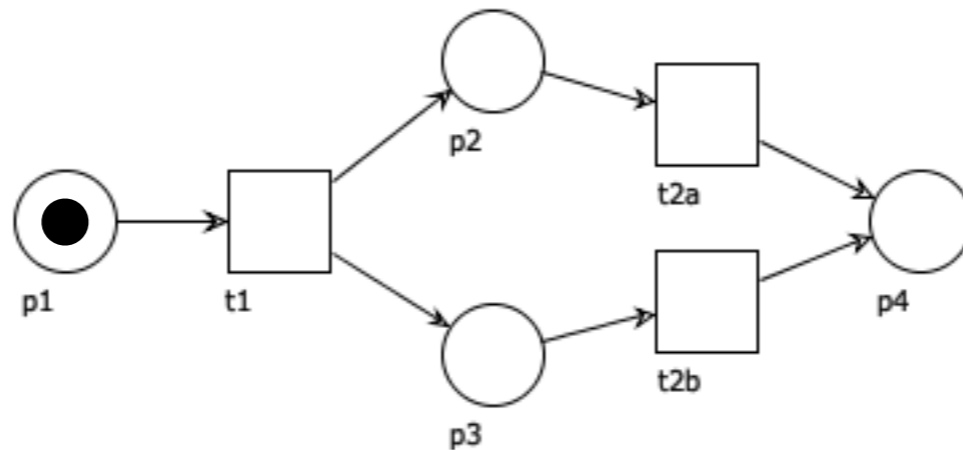
# Token analysis

Some tokens left in the net after case completion



# Token analysis

Some tokens left in the net after case completion



(when a token is in the final place the case should end)  
**can arise in workflow nets**

# Activity analysis

If tokens are left in the net after case completion  
then  
activities may still take place after case completion

it is a (worse) consequence of the previous flaw  
**can arise in workflow nets**

# Token analysis

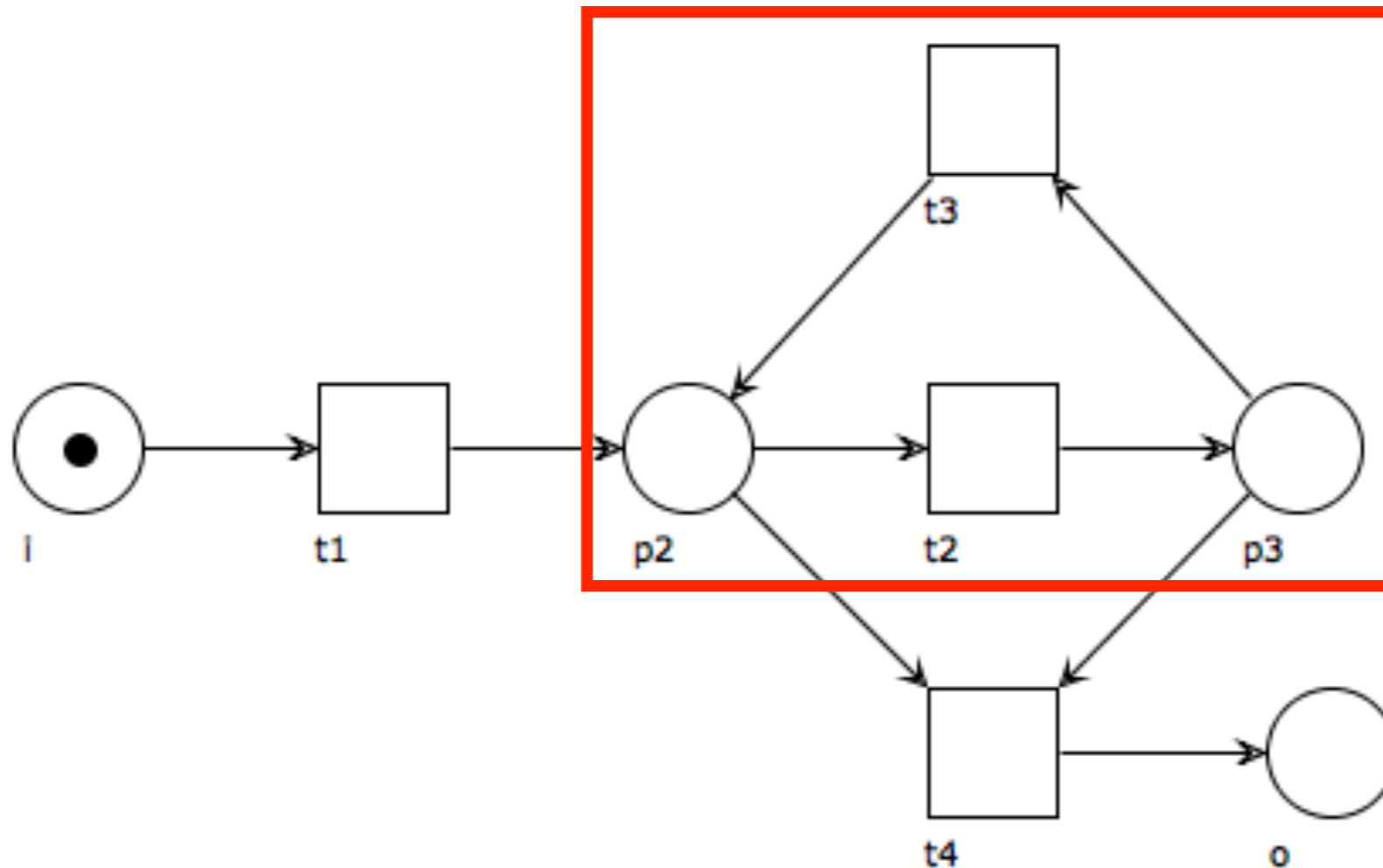
More than one token reach the end place

it can be a consequence of the previous flaws  
**can arise in workflow nets**



# Livelock

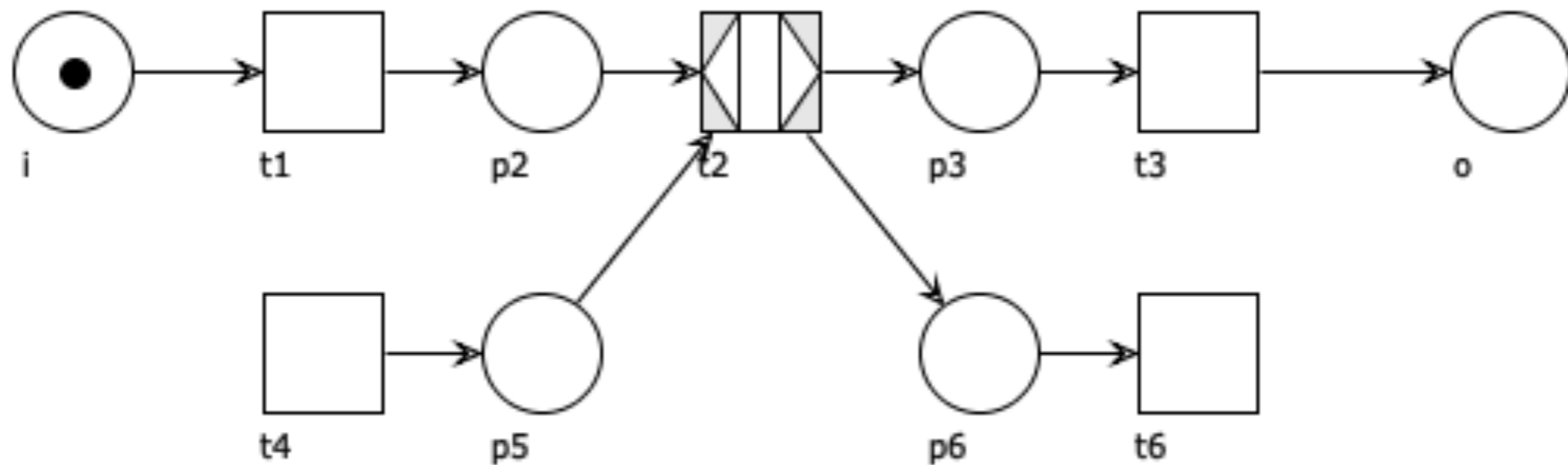
Livelock = divergence without producing output  
a case is trapped in a cycle with no opportunity to end



**can arise in workflow nets**

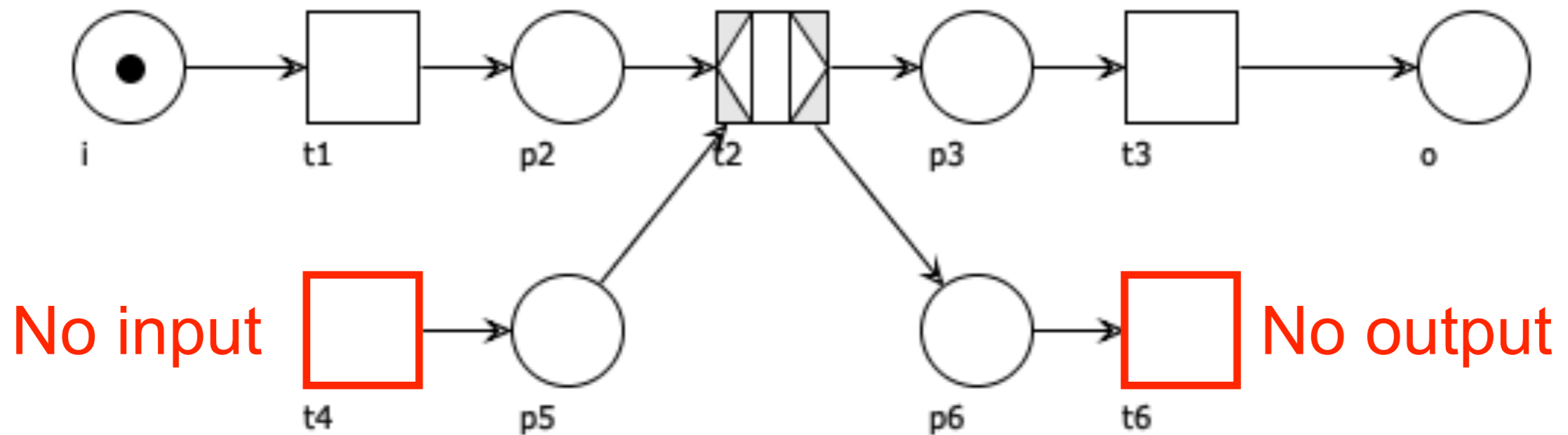
# Question time

Do you see any problem in the net below?



# Question time

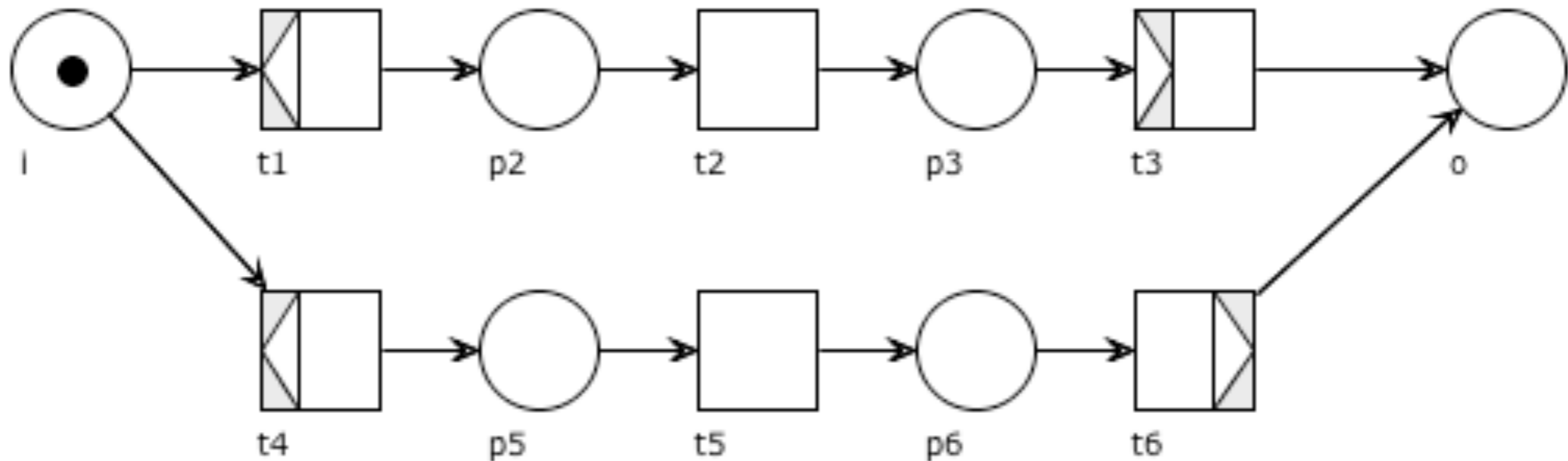
Do you see any problem in the net below?



Not a workflow net  
(not all nodes are on a path from i to o)

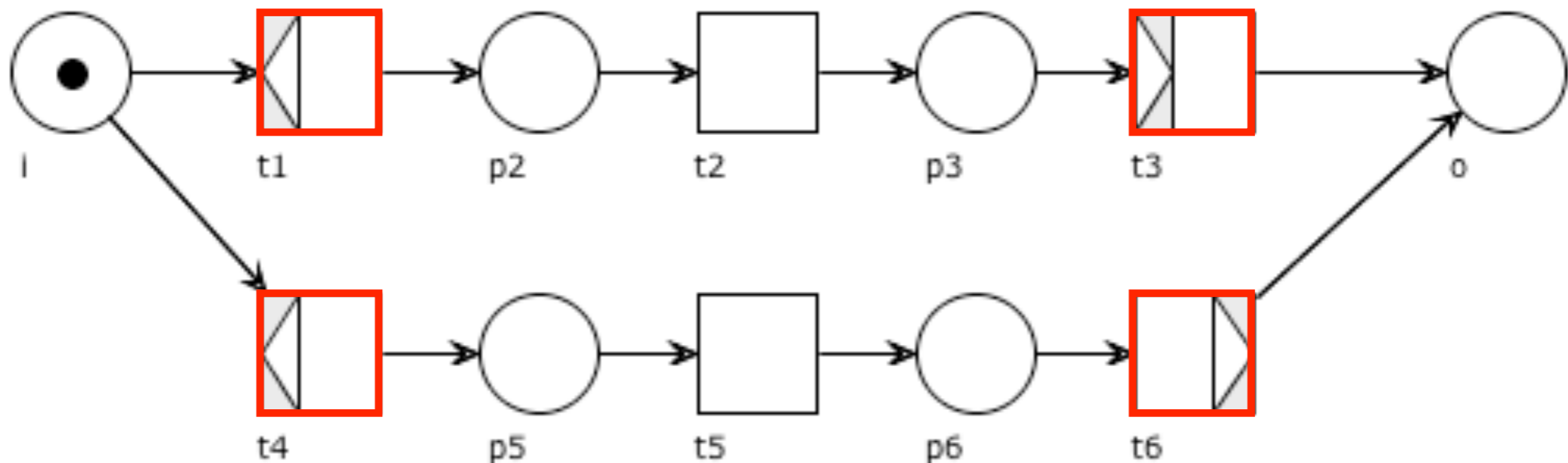
# Question time

Do you see any problem in the workflow net below?



# Question time

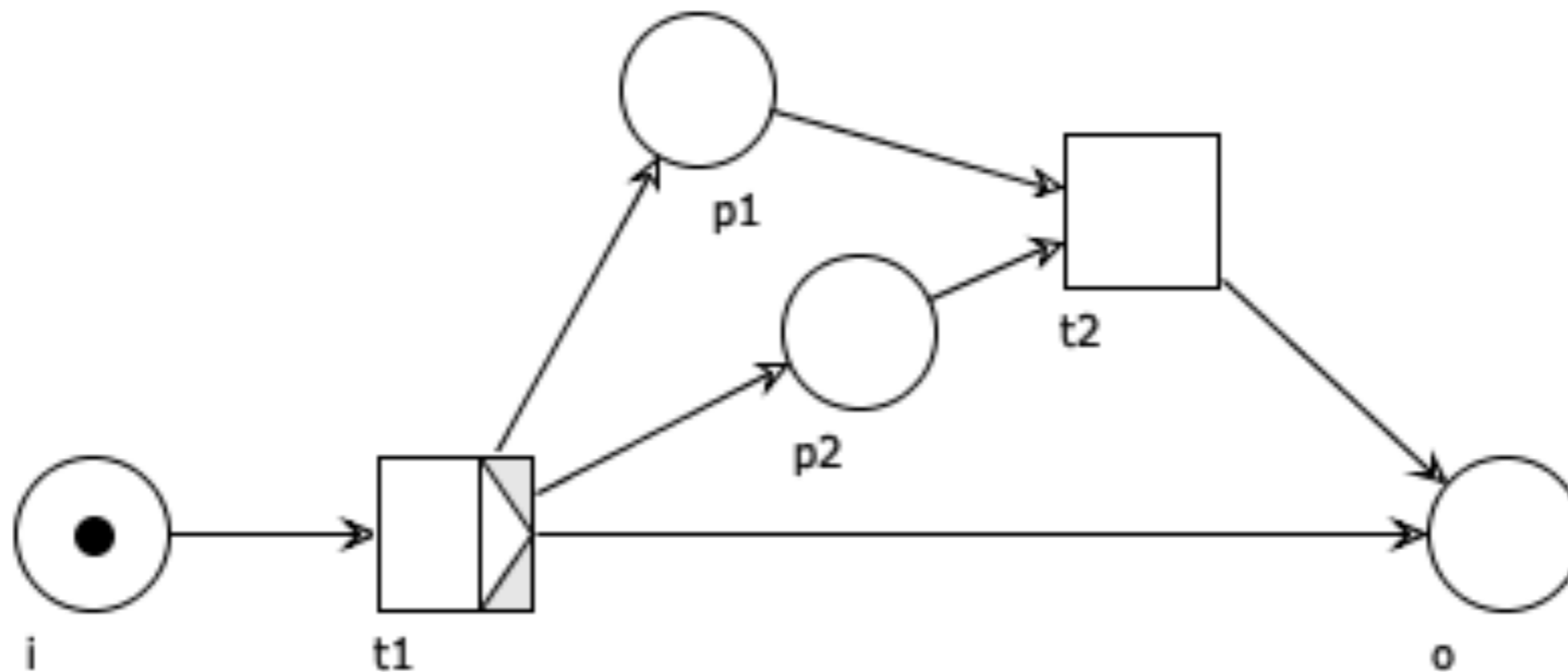
Do you see any problem in the workflow net below?



Wrong decorations!

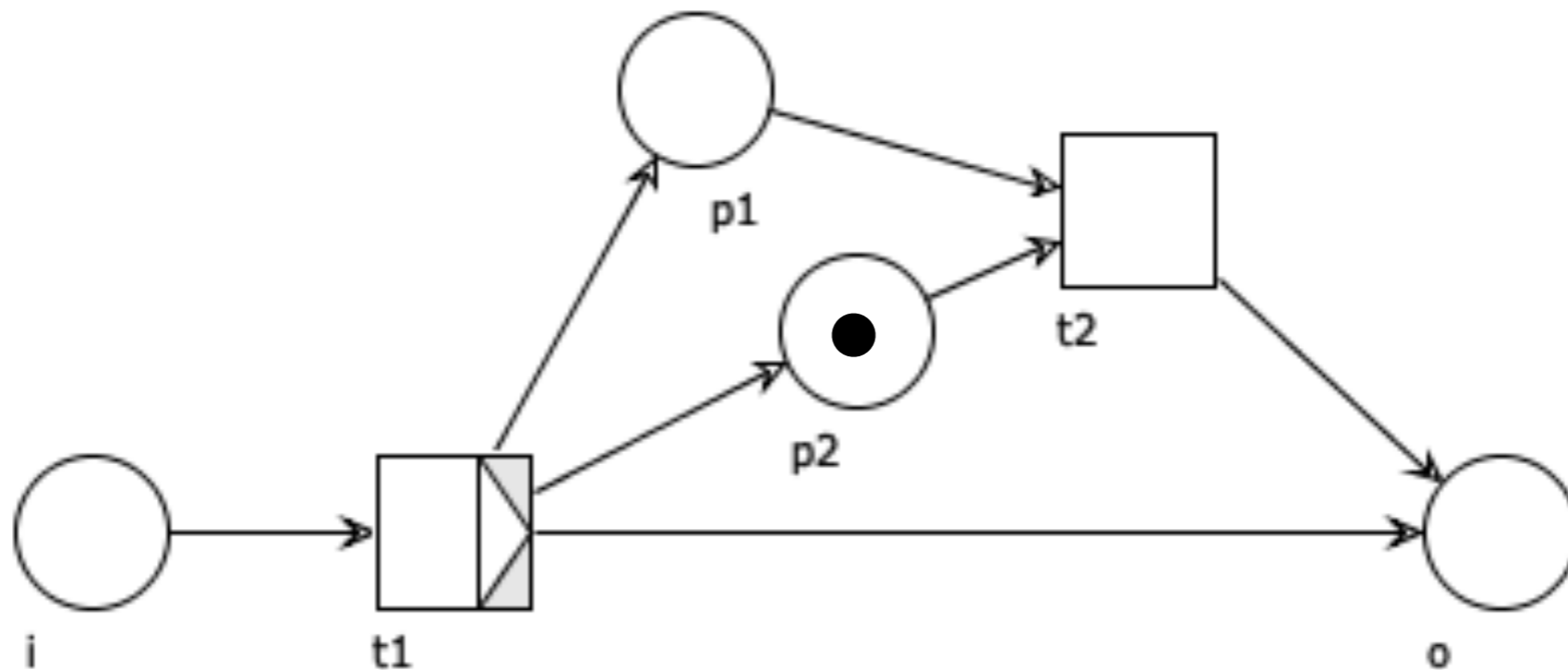
# Question time

Do you see any problem in the [workflow net](#) below?



# Question time

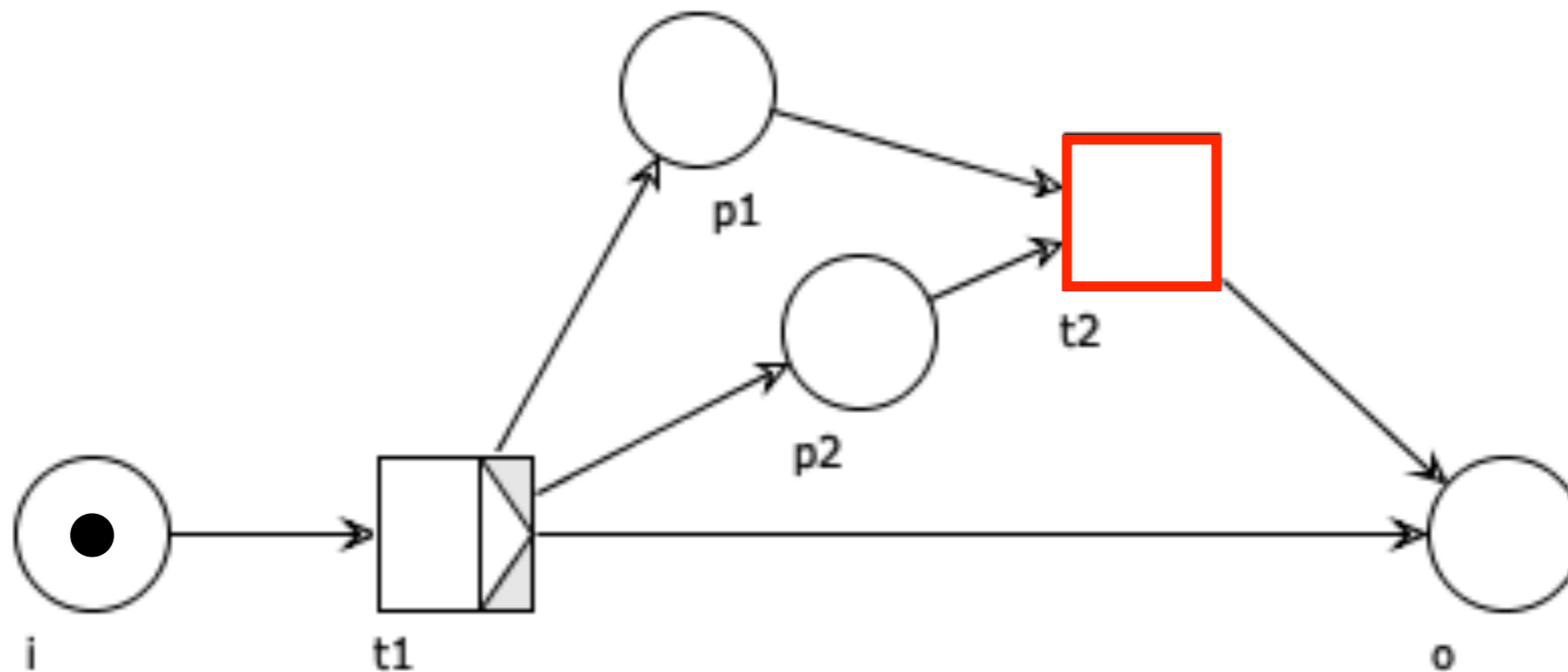
Do you see any problem in the [workflow net](#) below?



Possible deadlock

# Question time

Do you see any problem in the [workflow net](#) below?

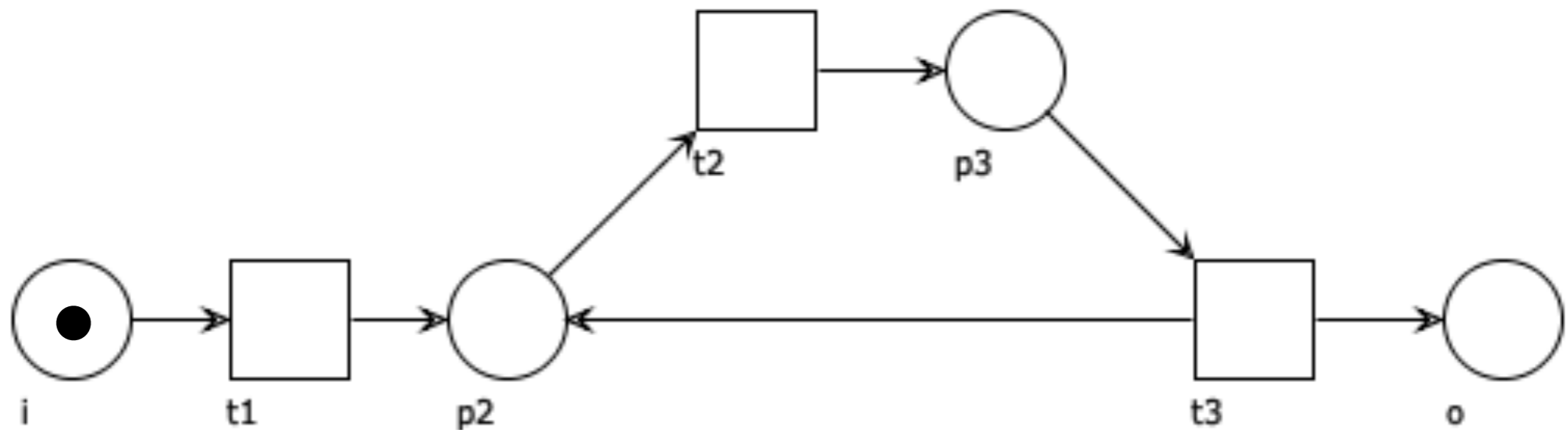


Dead task



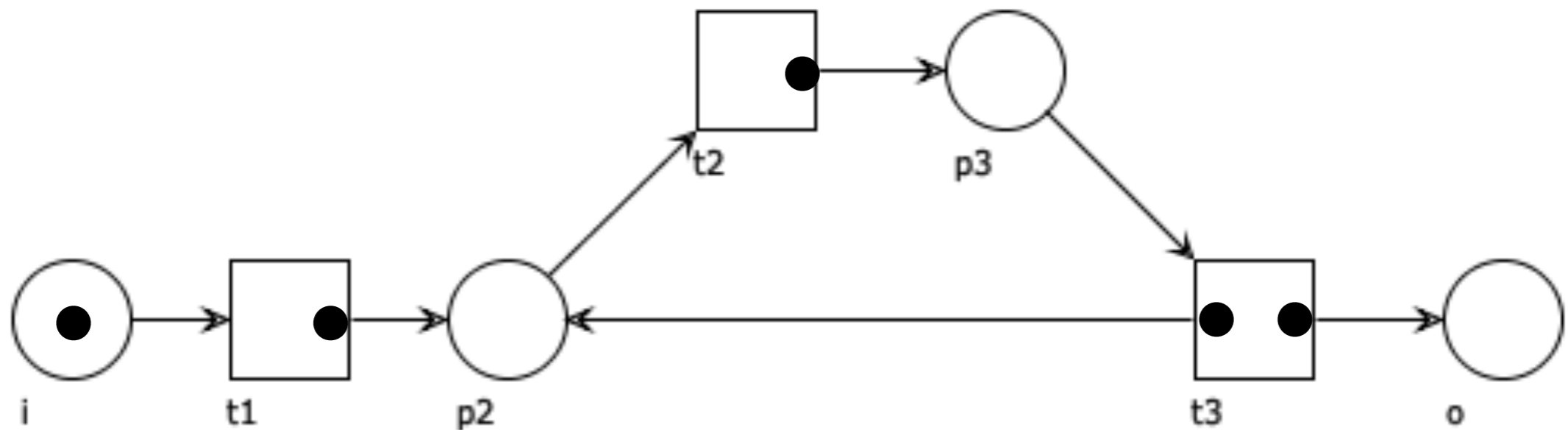
# Question time

Do you see any problem in the [workflow net](#) below?



# Question time

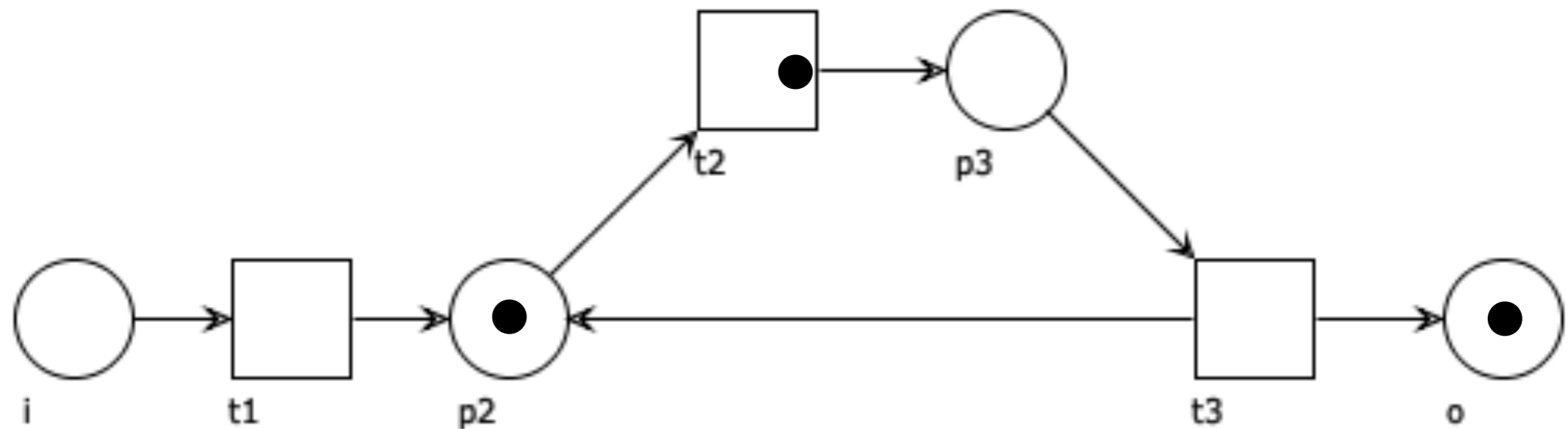
Do you see any problem in the [workflow net](#) below?



Some tokens left in the net after case completion

# Question time

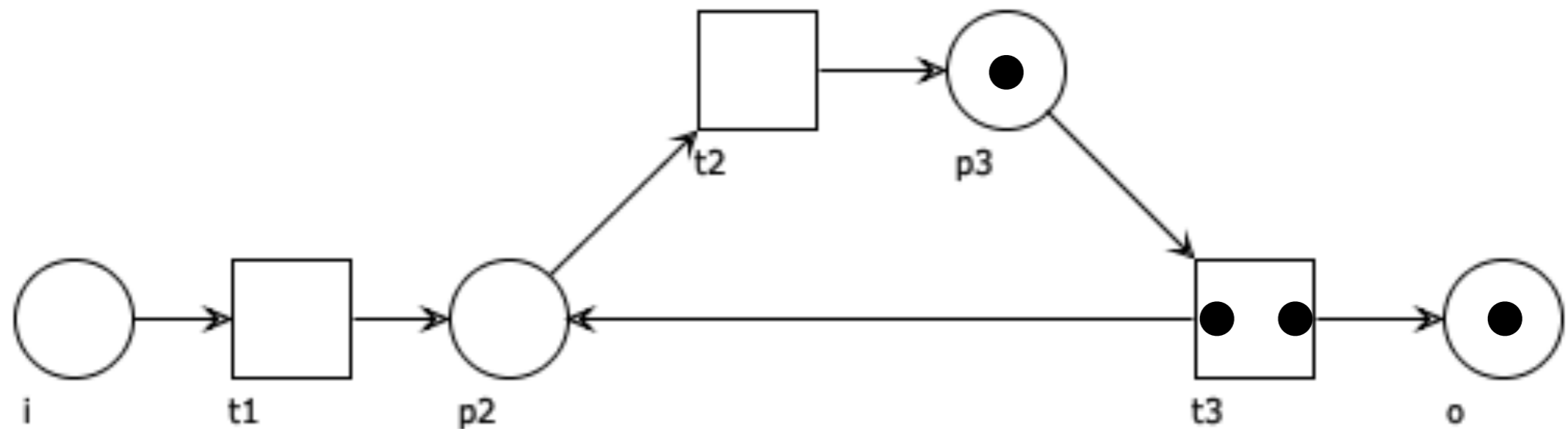
Do you see any problem in the [workflow net](#) below?



Activities still take place after case completion

# Question time

Do you see any problem in the [workflow net](#) below?



More than one token can reach the end place

# Remark

All the previous flaws are typical errors that  
can be detected  
without any knowledge about the actual content  
of the Business Process

# Verification and validation

**Validation** is concerned with the relation between the model and the reality

How does a model fit log files?

Which model does fit better?

**Verification** aims to answer qualitative questions

Is there a deadlock possible?

Is it possible to successfully handle a specific case?

Will all cases terminate eventually?

Is it possible to execute a certain task?

# Simulation techniques

## Test analysis

Try and see if certain firing sequences are allowed by the workflow net

## Using WoPeD:

Play (forward and backward) with net tokens

Record certain runs (to replay or explain)

Randomly select alternatives

**Problem:** how to make sure that all possible runs have been examined?

# Reachability analysis

## Verification by inspection

All possible runs of a workflow net are represented in its Reachability Graph (**when it is finite**)

Using WoPeD:

all reachable states are shown

(a single run does not necessarily visit all nodes)

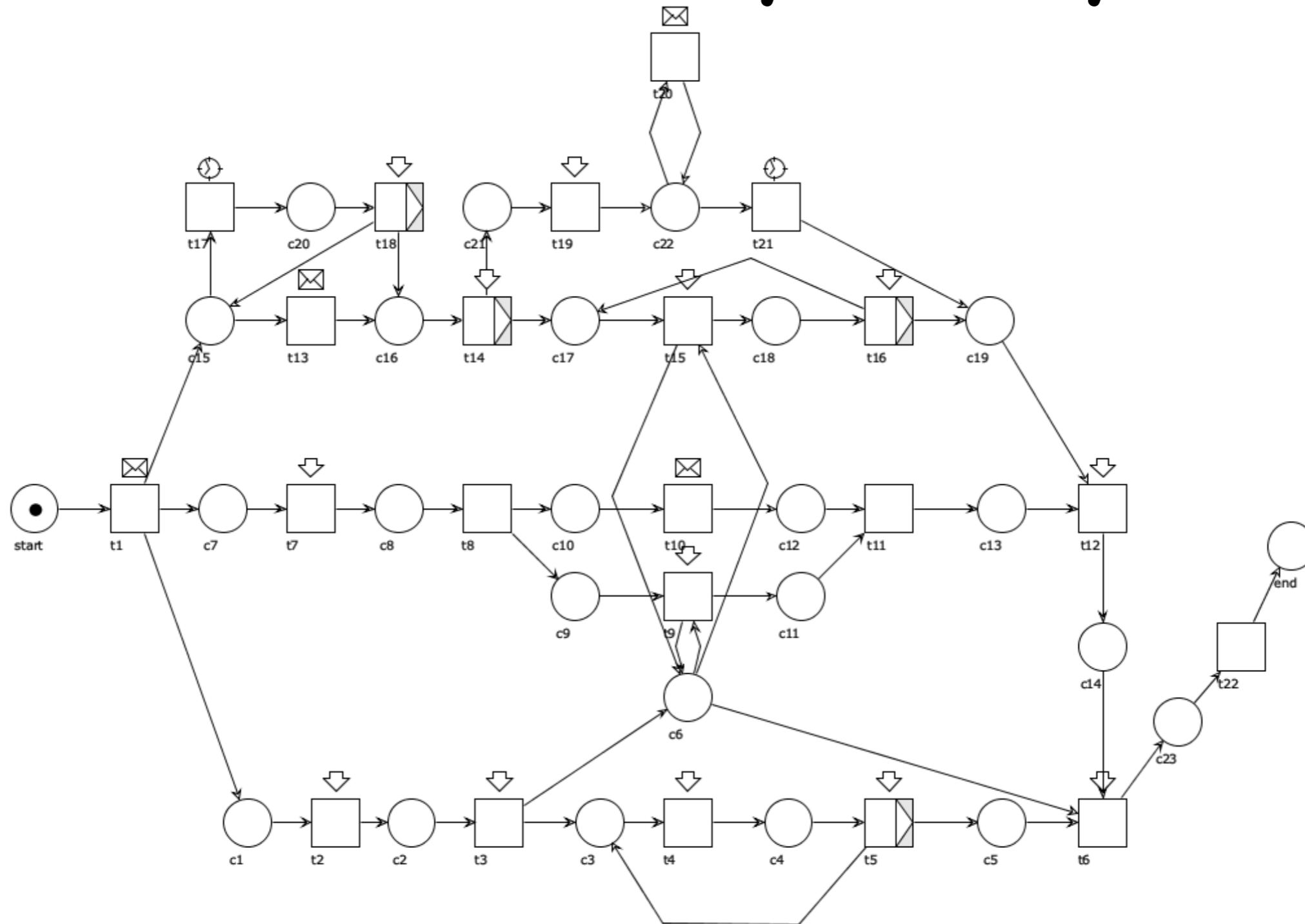
End states are evident (no outgoing arc)

Useful to check if dangerous or undesired states can arise (e.g. the green-green state in the two-traffic-lights)

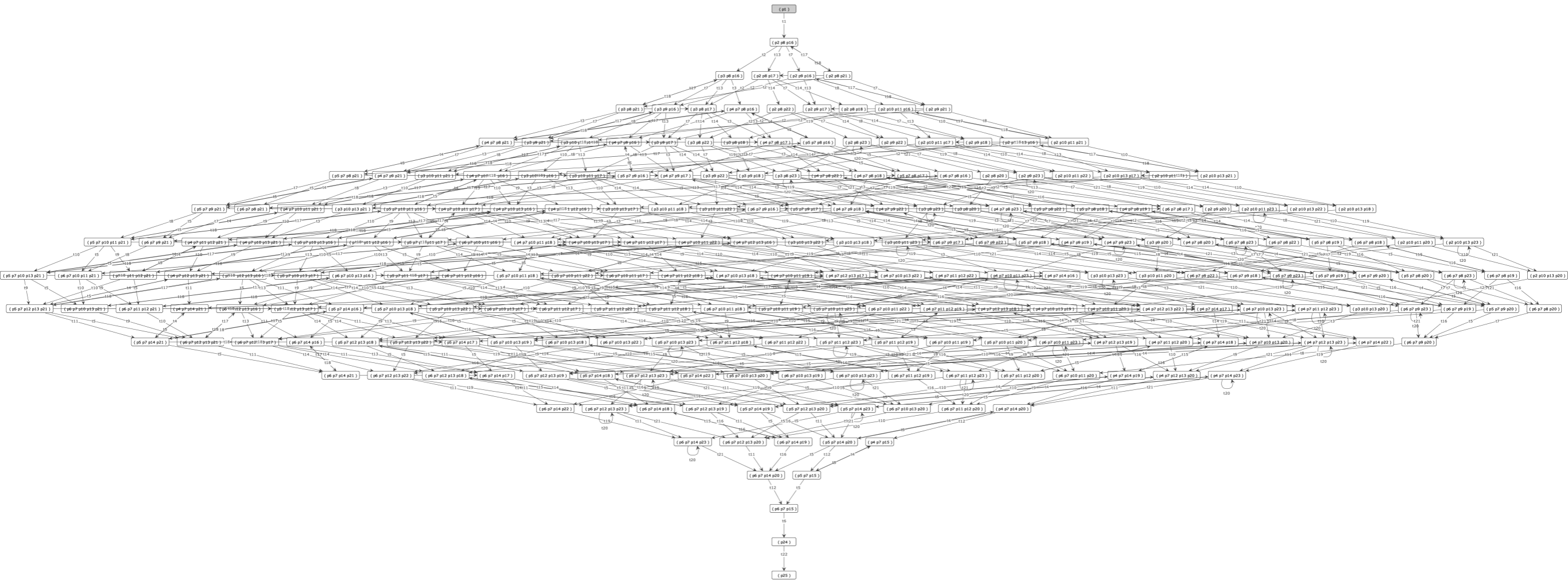
**Problem:** state explosion



# Reachability analysis



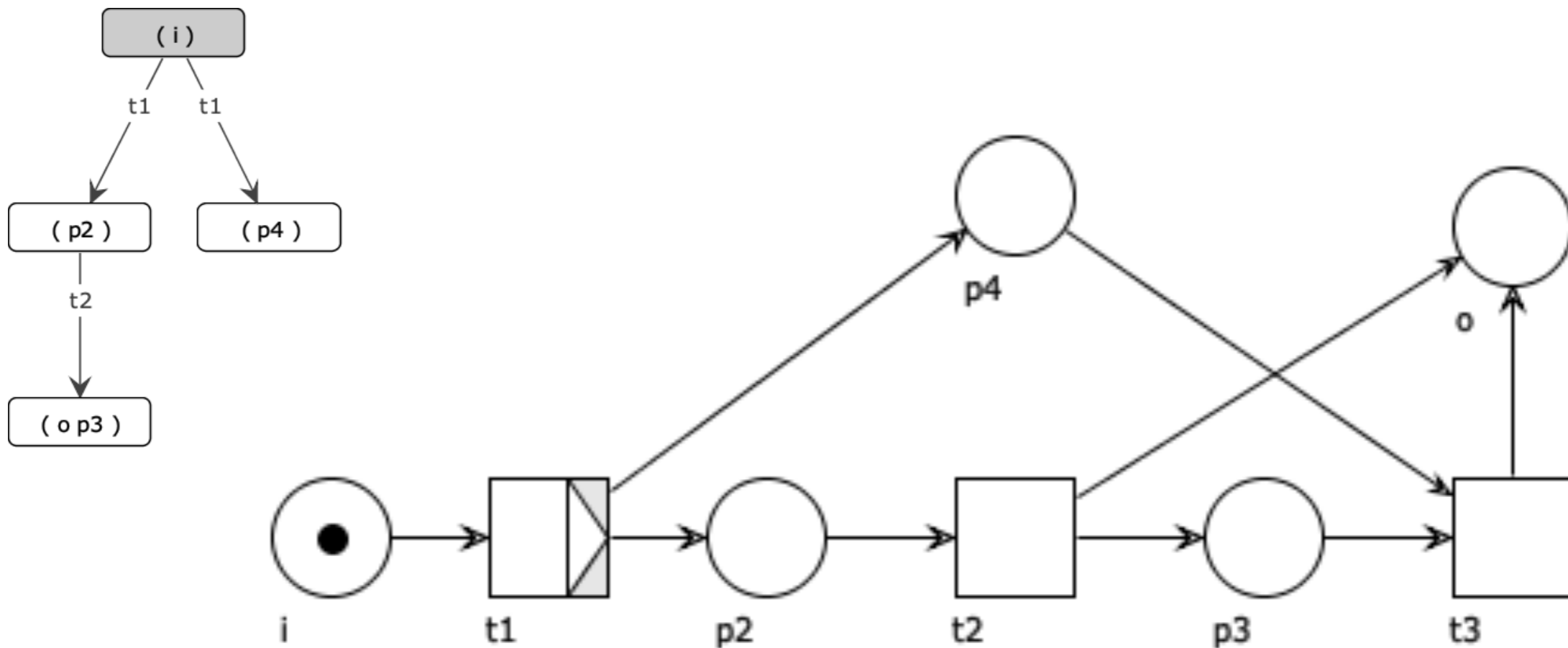
# Reachability analysis



**Problem:** state explosion

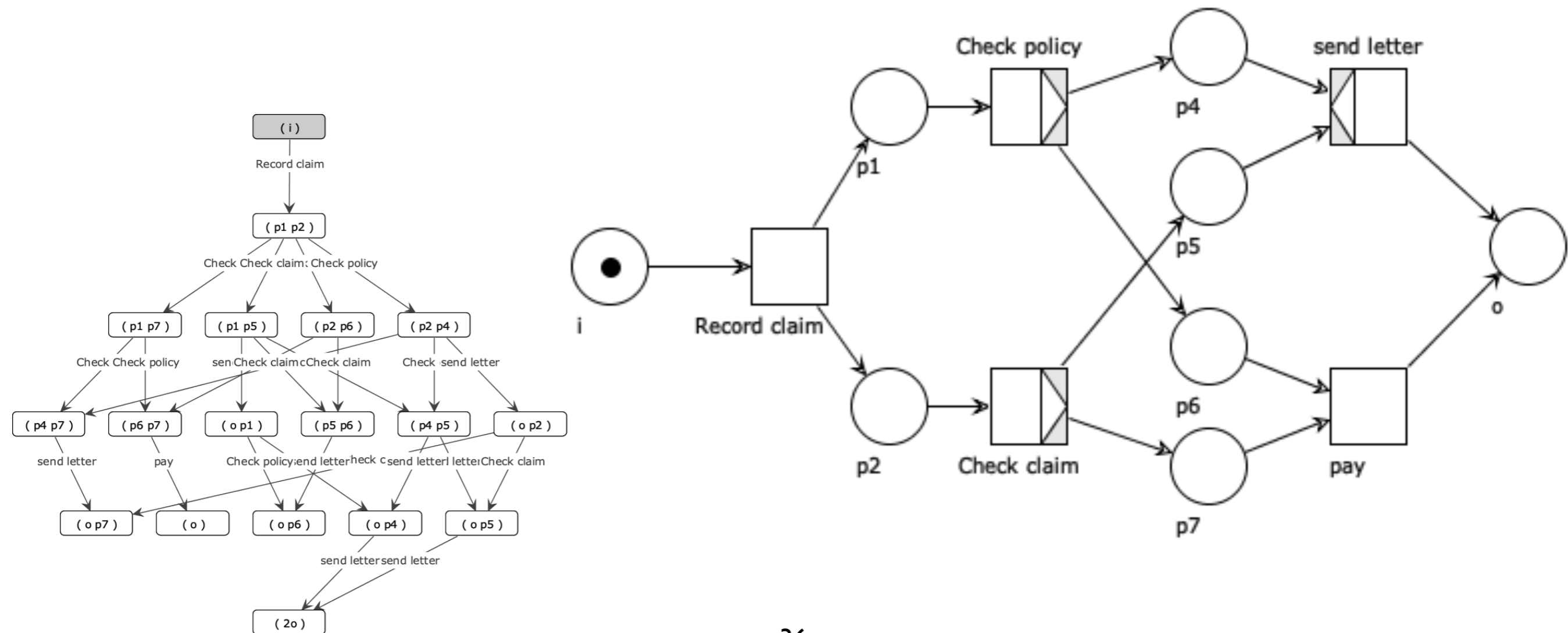
# Exercise

Do you see any problem in the workflow net below?



# Exercise

Which problem(s) in the workflow net below?  
How would you redesign the business process?



# Coverability

# Reachability analysis: finiteness?

## **Proposition:**

The reachability graph of a net is finite

if and only if

the net is bounded

# Reachability analysis: finiteness?

**Proposition:**  
A net is unbounded

if and only if

its reachability graph is not finite

# Coverability graph

A **coverability graph** is a finite over-approximation of the reachability graph

It allows for markings with infinitely many tokens in one place (called extended bags)

$$B : P \longrightarrow \mathbb{N} \cup \{\infty\}$$



# Discover unbounded places

Suppose

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \dots \xrightarrow{t_i} M_i \dots \xrightarrow{t_j} M_j$$

with  $M_i \subset M_j$

Let  $M = M_i$  and  $M' = M_j$  and  $L = M' - M$

By the monotonicity Lemma we have, for any  $n \in \mathbb{N}$ :

$$M \rightarrow^* M + L \rightarrow^* M + 2L \rightarrow^* \dots \rightarrow^* M + nL$$

Hence all places  $p$  marked by  $L$  (i.e. if  $L(p) > 0$ ) are unbounded

# Account for unbounded places

## Idea:

When computing the RG, if  $M'$  is found s.t.

$$M_0 \rightarrow^* M \rightarrow^* M' \text{ with } M \subset M'$$

Add the extended bag  $B$  (instead of  $M'$ ) to the graph

$$\text{where } B(p) = \begin{cases} M'(p) & \text{if } M'(p) - M(p) = 0 \\ \infty & \text{otherwise} \end{cases}$$

# A few remarks

**Idea:** mark unbounded places by  $\infty$

**Remind:**  $M \subset M'$  means that  $M \subseteq M' \wedge M \neq M'$ , i.e.,

1. for any  $p \in P$ ,  $M'(p) \geq M(p)$
2. there exists at least one place  $q \in P$  such that  $M'(q) > M(q)$

**Remark:**

Requiring  $M_0 \rightarrow^* M \rightarrow^* M'$  is different than requiring  $M, M' \in [M_0 \rangle$

# Operations on extended bags

**Inclusion:** Let  $B, B' : P \rightarrow \mathbb{N} \cup \{\infty\}$

We write  $B \subseteq B'$  if for any  $p$  we have

$$B'(p) = \infty \text{ or } B(p), B'(p) \in \mathbb{N} \wedge B(p) \leq B'(p)$$

**Sum:** Let  $B, B' : P \rightarrow \mathbb{N} \cup \{\infty\}$

$$(B + B')(p) = \begin{cases} \infty & \text{if } B(p) = \infty \text{ or } B'(p) = \infty \\ B(p) + B'(p) & \text{if } B(p), B'(p) \in \mathbb{N} \end{cases}$$

**Difference:** Let  $B : P \rightarrow \mathbb{N} \cup \{\infty\}$  and  $M : P \rightarrow \mathbb{N}$  with  $M \subseteq B$

$$(B - M)(p) = \begin{cases} \infty & \text{if } B(p) = \infty \\ B(p) - M(p) & \text{if } B(p) \in \mathbb{N} \end{cases}$$

# Operations on extended bags: examples

$$2a + b + \infty c \subseteq 2a + 2b + \infty c \subseteq 2a + \infty b + \infty c$$

$$2a + b + \infty c \not\subseteq a + 2b + \infty c \not\subseteq 2a + \infty b + 3c$$

$$(3a + 2b + \infty c) + (2a + \infty b + \infty c) = (5a + \infty b + \infty c)$$

$$(5a + \infty b + \infty c) - (3a + 2b + \infty c) = ?$$

**must be a marking!**

$$(5a + \infty b + \infty c) - (3a + 2b + 4c) = (2a + \infty b + \infty c)$$

# Compute a reachability graph

1. Initially  $N = \{ M_0 \}$  and  $A = \emptyset$

(all bags are finite in this case)

# Compute a reachability graph

1. Initially  $N = \{ M_0 \}$  and  $A = \emptyset$
2. Take a bag  $B \in N$  and a transition  $t \in T$  such that
  1.  $B$  enables  $t$  and there is no arc labelled  $t$  leaving from  $B$

(all bags are finite in this case)

# Compute a reachability graph

1. Initially  $N = \{ M_0 \}$  and  $A = \emptyset$
2. Take a bag  $B \in N$  and a transition  $t \in T$  such that
  1.  $B$  enables  $t$  and there is no arc labelled  $t$  leaving from  $B$
3. Let  $B' = B - \cdot t + t \cdot$

(all bags are finite in this case)



# Compute a reachability graph

1. Initially  $N = \{ M_0 \}$  and  $A = \emptyset$
2. Take a bag  $B \in N$  and a transition  $t \in T$  such that
  1.  $B$  enables  $t$  and there is no arc labelled  $t$  leaving from  $B$
3. Let  $B' = B - \cdot t + t \cdot$
4. Add  $B'$  to  $N$  and  $(B, t, B')$  to  $A$

(all bags are finite in this case)

# Compute a reachability graph

1. Initially  $N = \{ M_0 \}$  and  $A = \emptyset$
2. Take a bag  $B \in N$  and a transition  $t \in T$  such that
  1.  $B$  enables  $t$  and there is no arc labelled  $t$  leaving from  $B$
3. Let  $B' = B - \cdot t + t \cdot$
4. Add  $B'$  to  $N$  and  $(B, t, B')$  to  $A$
5. Repeat steps 2,3,4 until no new arc can be added

(all bags are finite in this case)

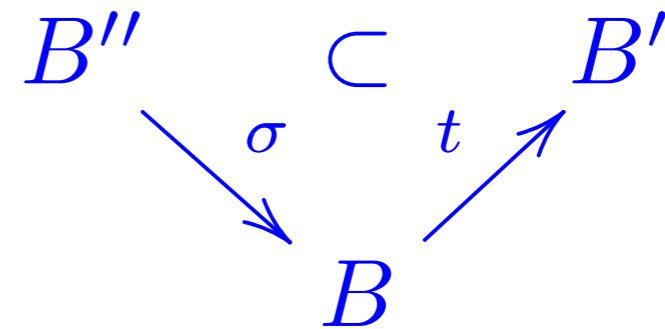
# Compute a reachability graph

1. Initially  $N = \{ M_0 \}$  and  $A = \emptyset$
2. Take a bag  $B \in N$  and a transition  $t \in T$  such that
  1.  $B$  enables  $t$  and there is no arc labelled  $t$  leaving from  $B$
3. Let  $B' = B - \cdot t + t \cdot$
4. Add  $B'$  to  $N$  and  $(B, t, B')$  to  $A$
5. Repeat steps 2,3,4 until no new arc can be added

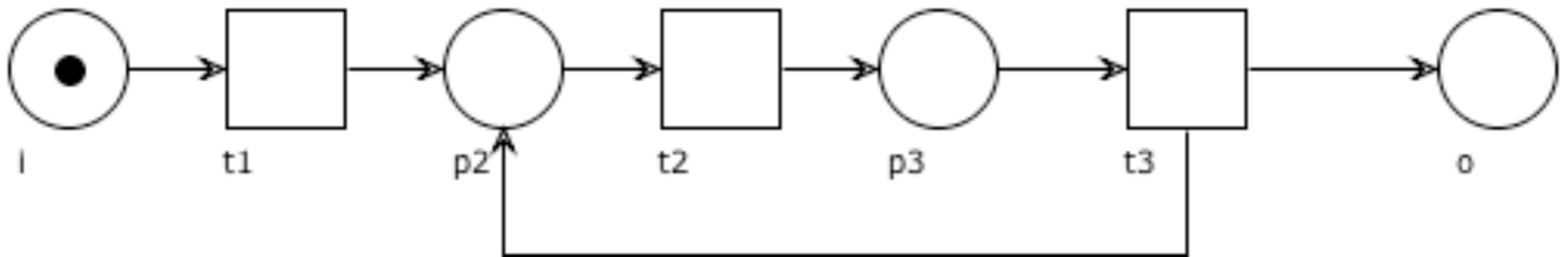
(all bags are finite in this case)

# Compute a coverability graph

1. Initially  $N = \{ M_0 \}$  and  $A = \emptyset$
2. Take a bag  $B \in N$  and a transition  $t \in T$  such that
  1.  $B$  enables  $t$  and there is no arc labelled  $t$  leaving from  $B$
3. Let  $B' = B - \cdot t + t \cdot$
4. Let  $B_c'$  such that for any  $p \in P$ 
  1.  $B_c'(p) = \infty$  if there is a node  $B'' \in N$  such that
    1. there is a directed path from  $B''$  to  $B$  in the graph  $(N, A)$
    2.  $B'' \subset B'$ ,
    3.  $B''(p) < B'(p)$
  2.  $B_c'(p) = B'(p)$  otherwise
5. Add  $B_c'$  to  $N$  and  $(B, t, B_c')$  to  $A$
6. Repeat steps 2,3,4,5 until no new arc can be added

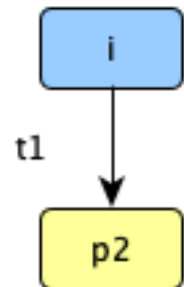
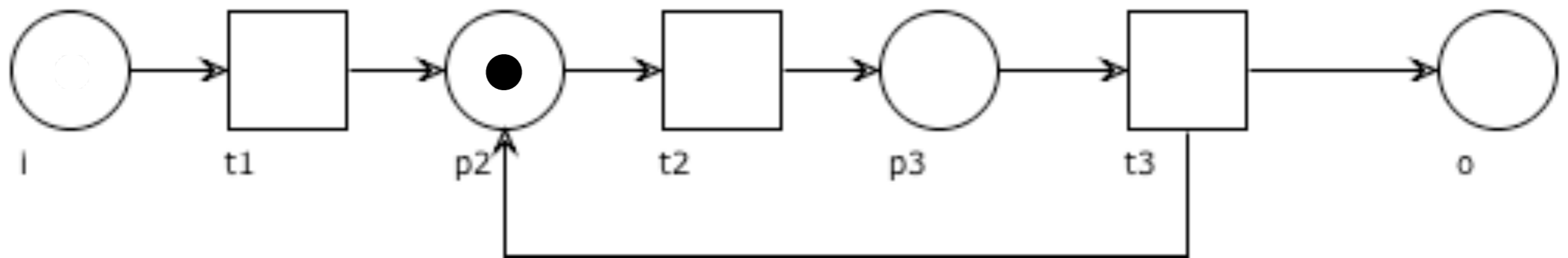


# Example

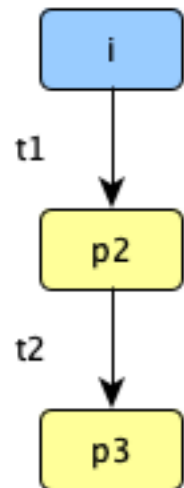
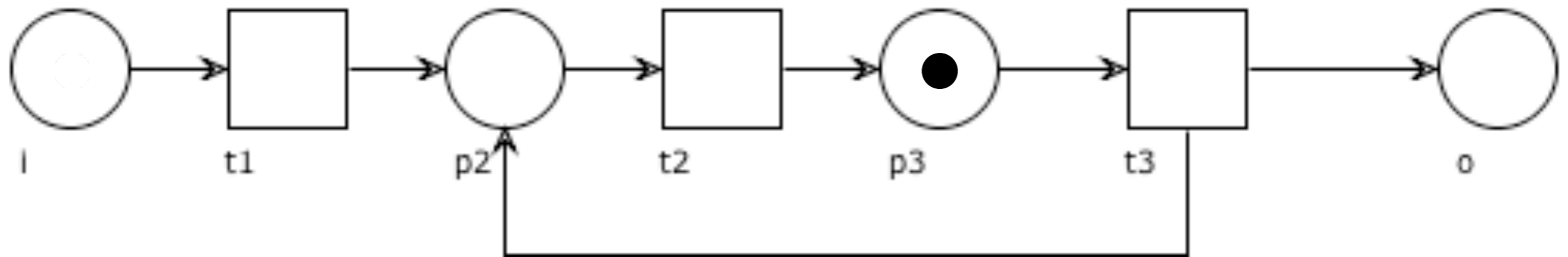


$i$

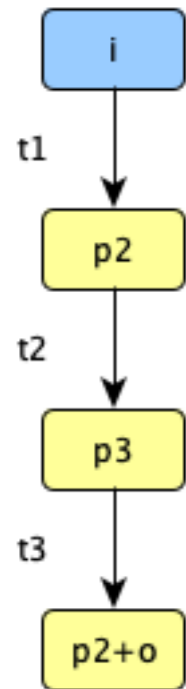
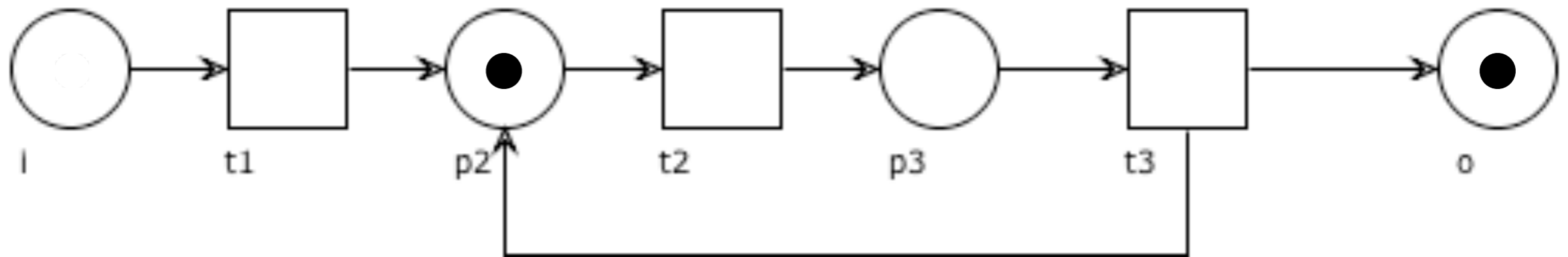
# Example



# Example

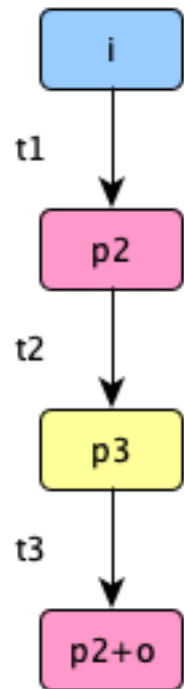
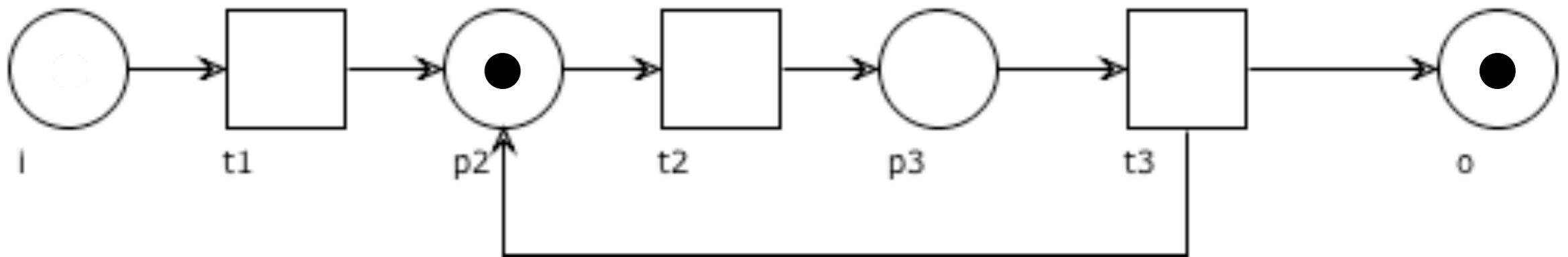


# Example

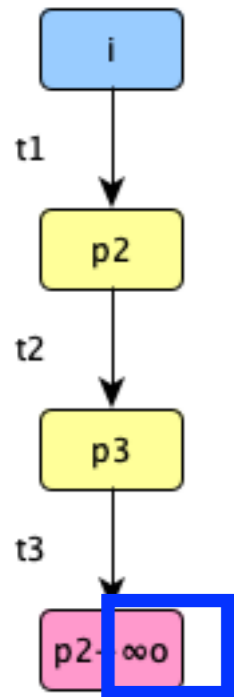
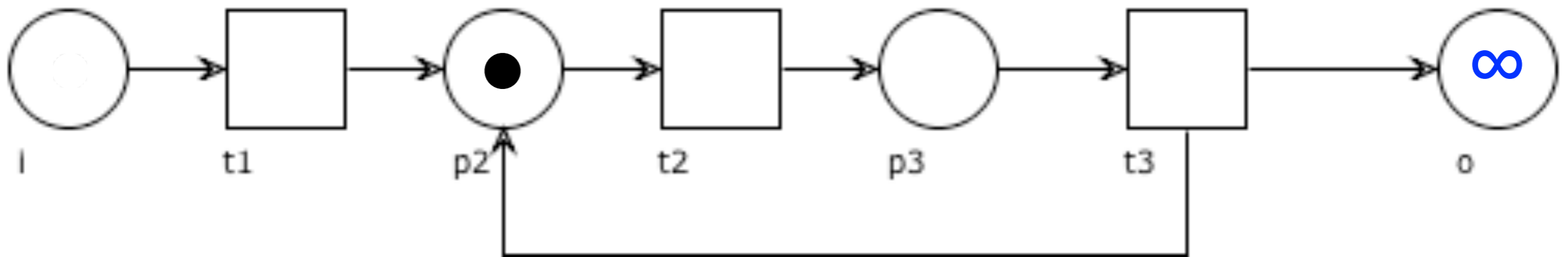




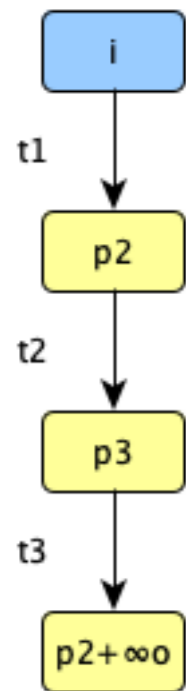
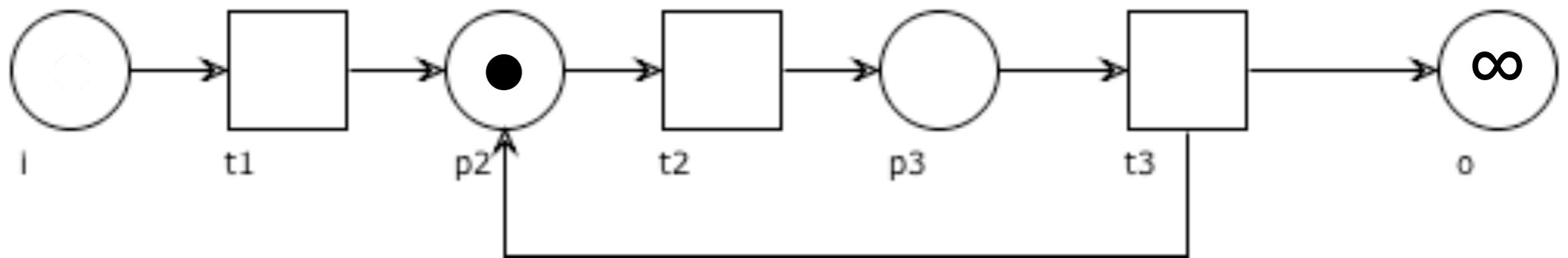
# Example



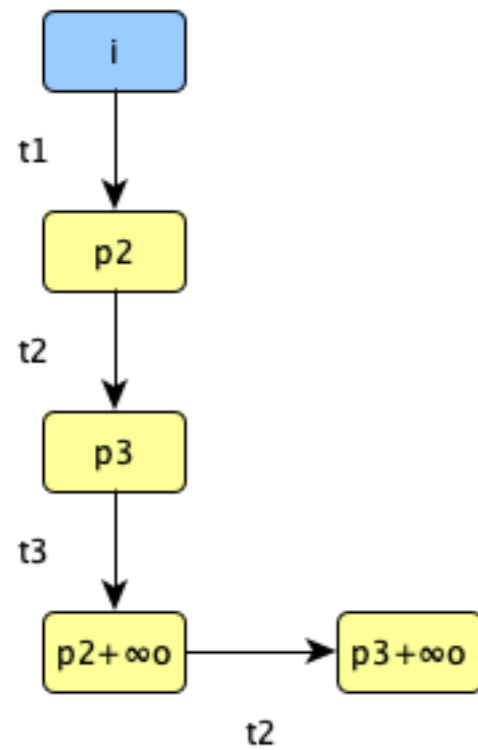
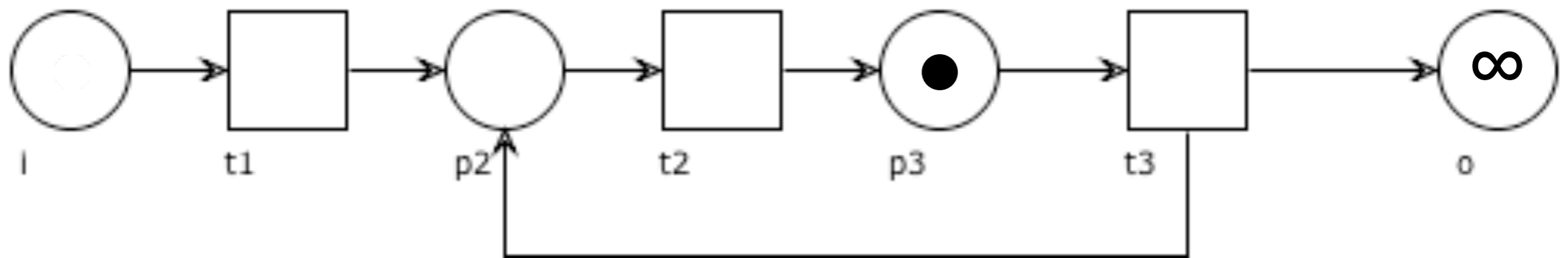
# Example



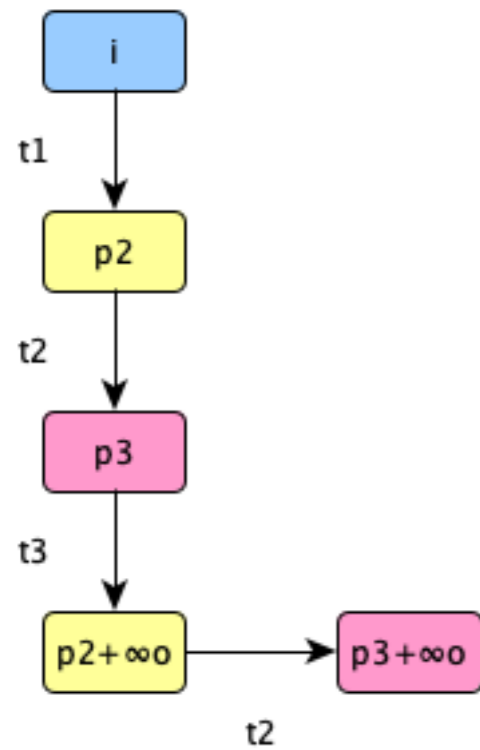
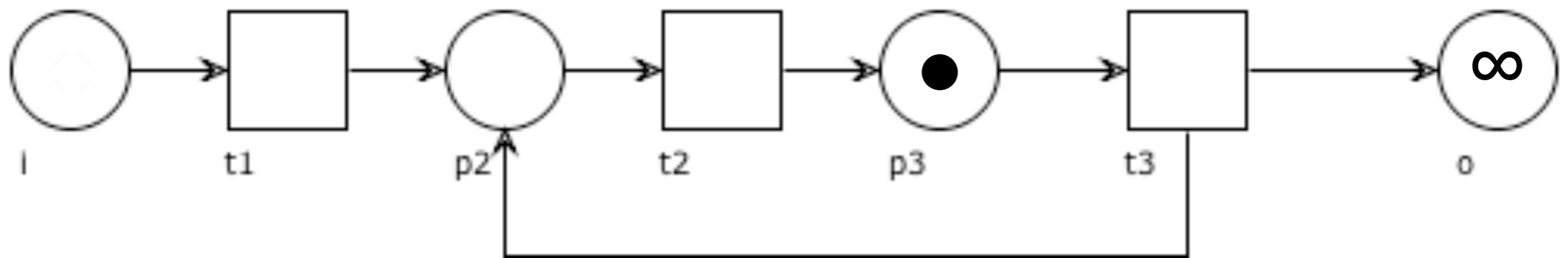
# Example



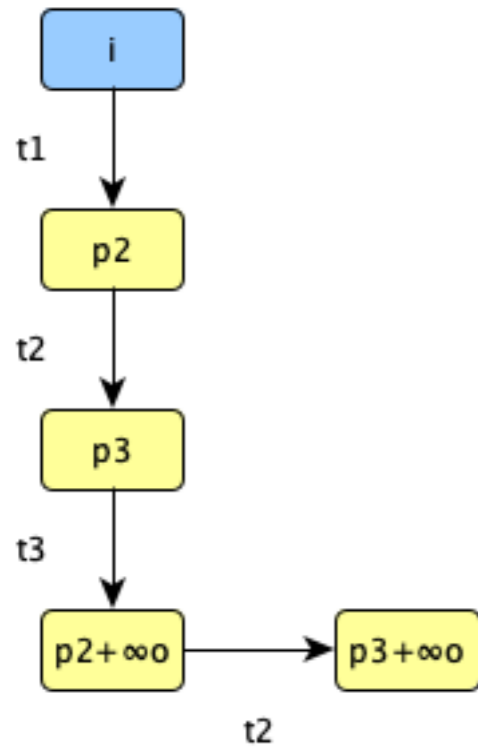
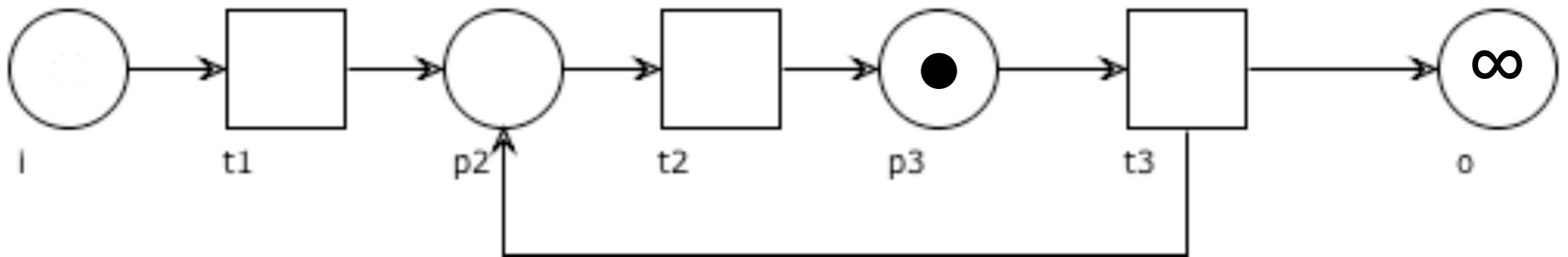
# Example



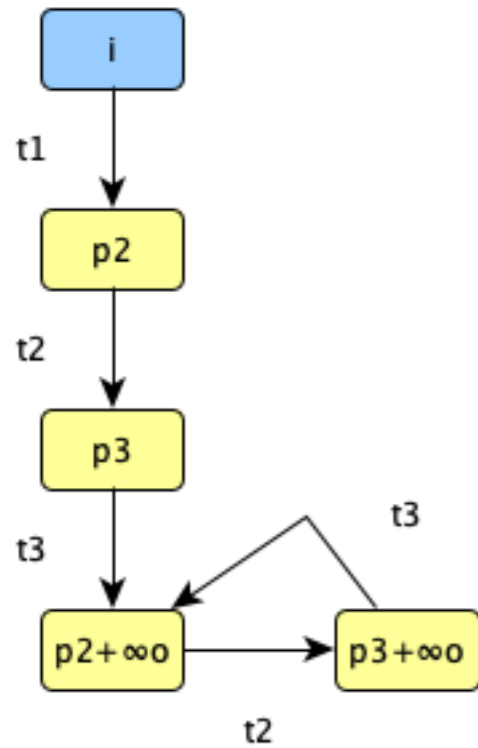
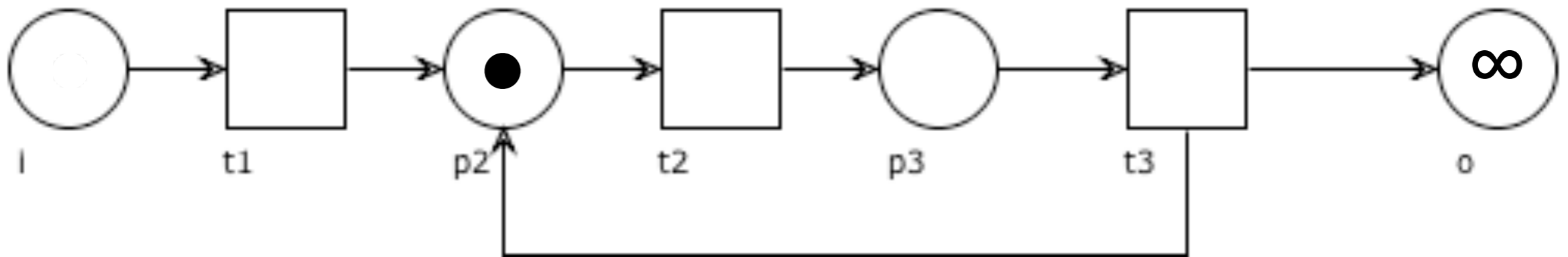
# Example



# Example



# Example



# Properties of coverability graphs

A coverability graph **is always finite**,  
but in general it **is not uniquely defined**  
(it depends on which  $B$  and  $t$  are selected at step 2)

Every firing sequence has a corresponding path in the CG  
(the converse is not necessarily true)

Any path in a CG that visits only finite markings  
corresponds to a firing sequence

If the RG is finite, then it coincides with the CG



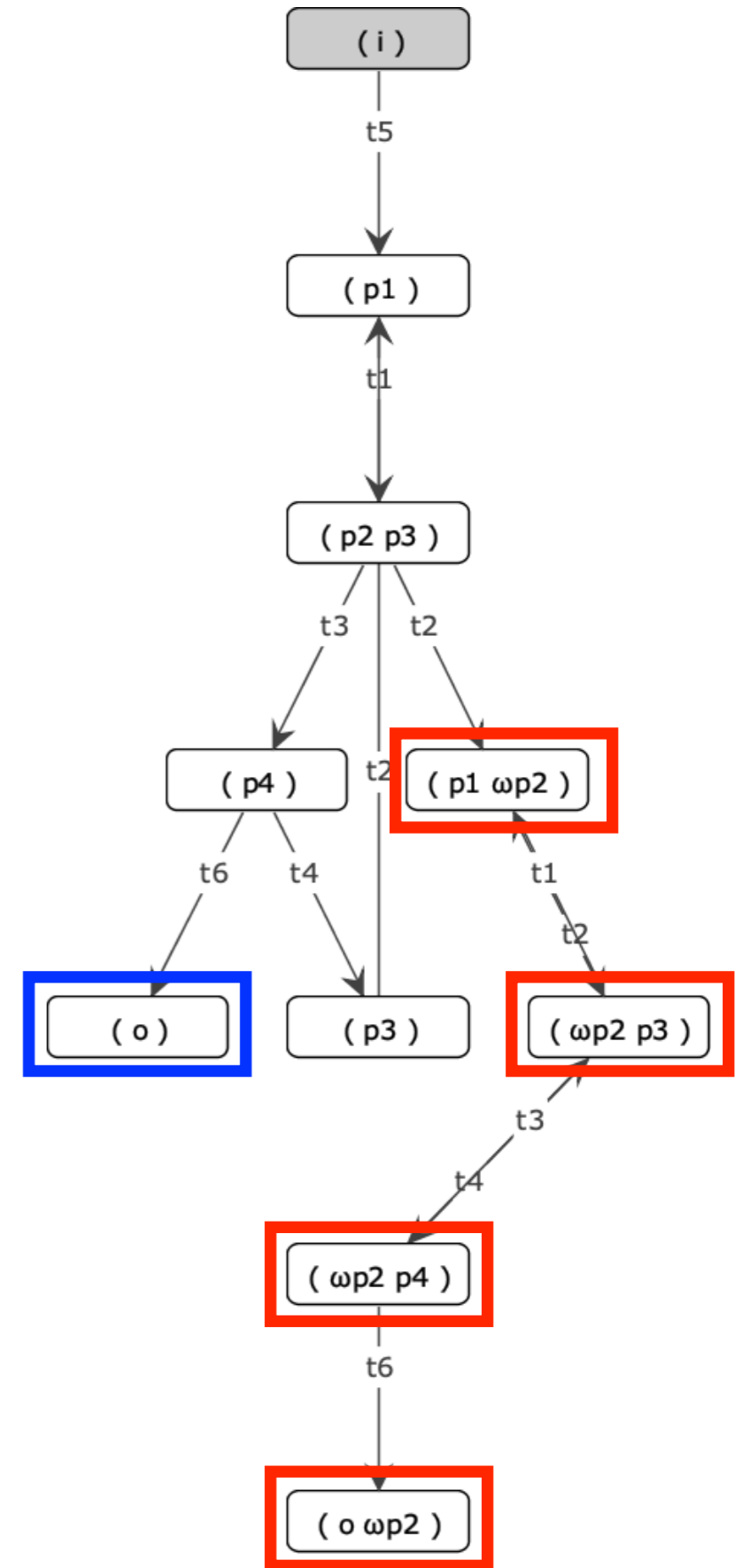
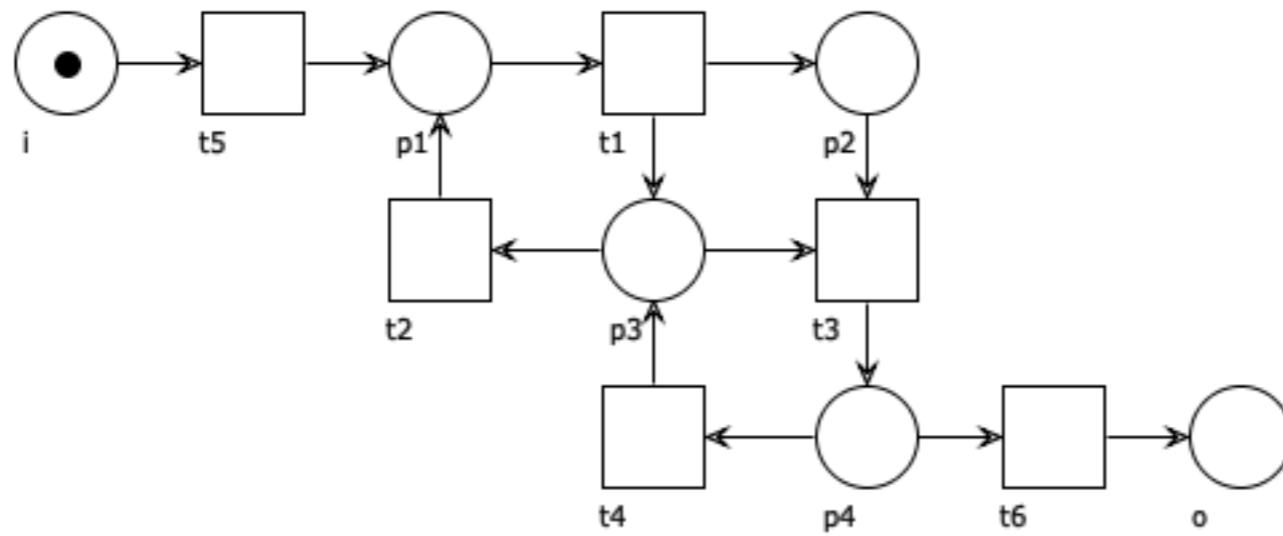
# Reachability analysis by coverability

All possible behaviours of a workflow net are represented exactly in the Reachability Graph (if finite)

We use Coverability Graph when necessary (RG not finite)

WoPeD computes a Coverability Graph

# Example



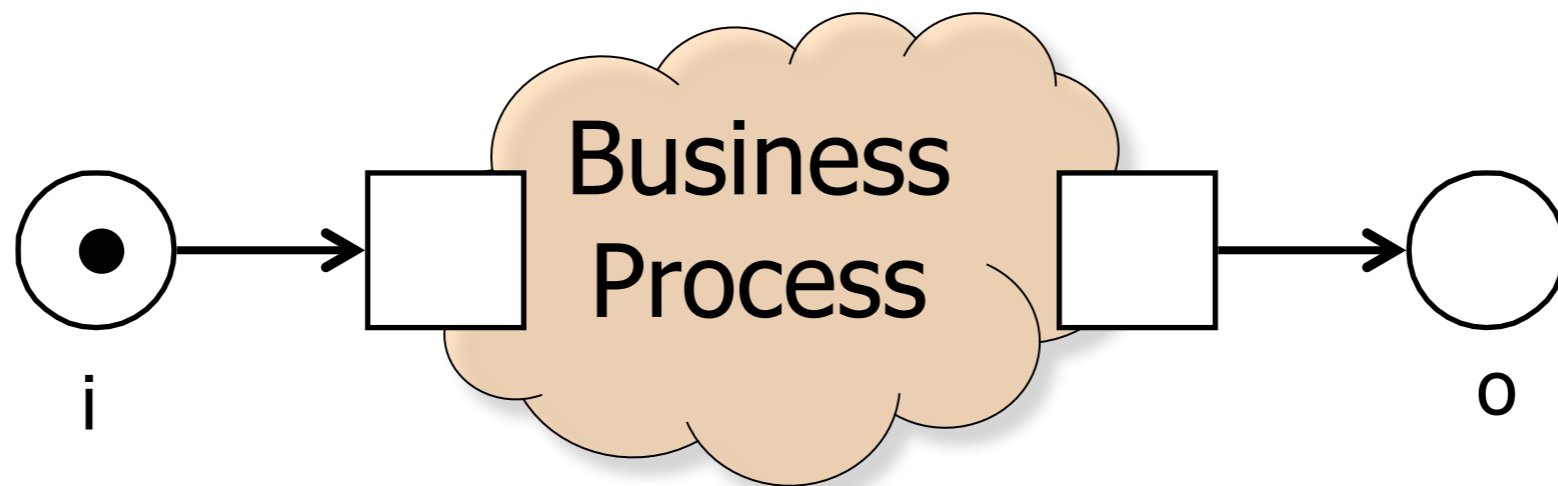
# Soundness

# Soundness of Business Processes

A process is called **sound** if

1. it contains no unnecessary tasks
2. every case is always completed in full
3. no pending items are left after case completion

# Soundness of Business Processes



# Soundness of Workflow nets

A workflow net is called **sound** if

1. for each transition  $t$ ,  
there is a marking  $M$  (reachable from  $i$ ) that enables  $t$
2. for each token put in place  $i$ ,  
one token eventually appears in the place  $o$
3. when a token is in place  $o$ , all other places are empty

# Fairness assumption

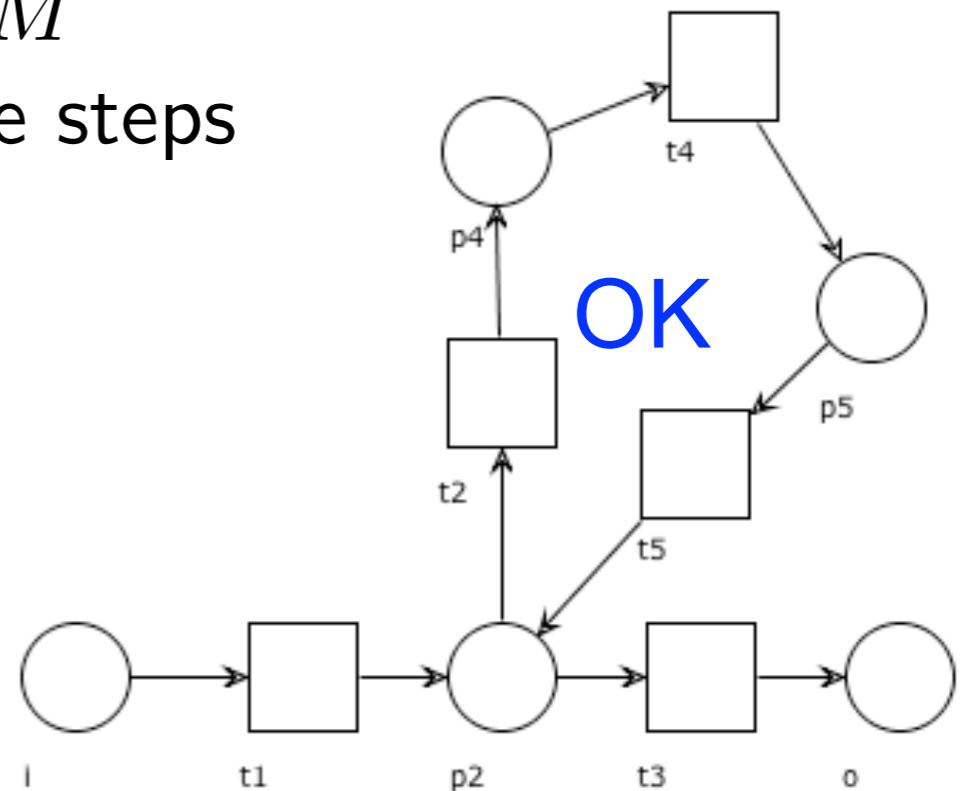
## Remark:

Condition 2 does not mean that iteration must be forbidden or bound

It says that from any reachable marking  $M$  there must be possible to reach  $o$  in some steps

## Fairness assumption:

A task cannot be postponed indefinitely



# Soundness, Formally

A workflow net is called **sound** if

**no dead task** no transition is dead

$$\forall t \in T. \exists M \in [i \rangle. M \xrightarrow{t}$$

**option to complete** place  $o$  is eventually marked

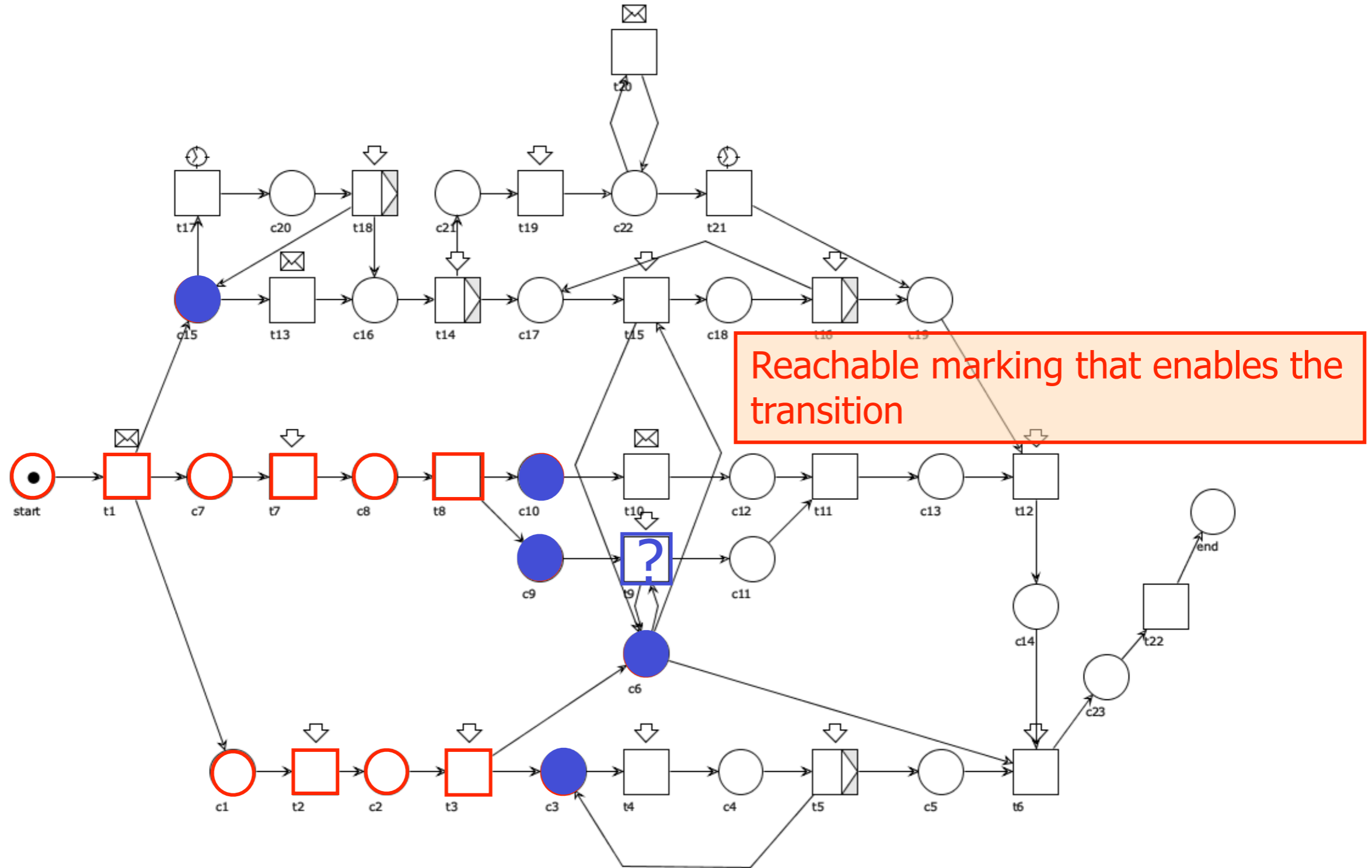
$$\forall M \in [i \rangle. \exists M' \in [M \rangle. M'(o) \geq 1$$

**proper completion** when  $o$  is marked, no other token is left

$$\forall M \in [i \rangle. M(o) \geq 1 \Rightarrow M = o$$



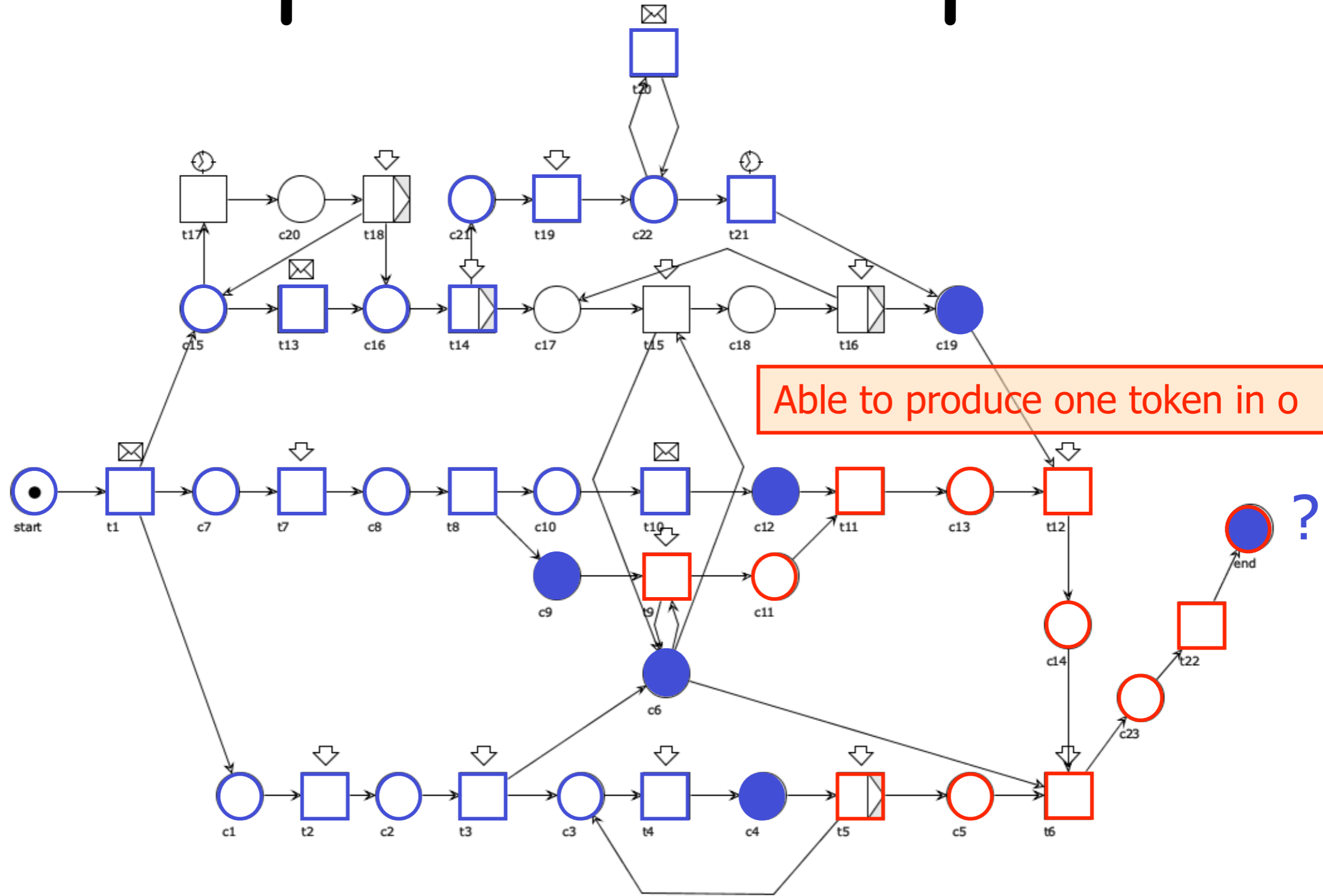
# 1: no dead tasks



# 1: no dead tasks

The check must be repeated for each task

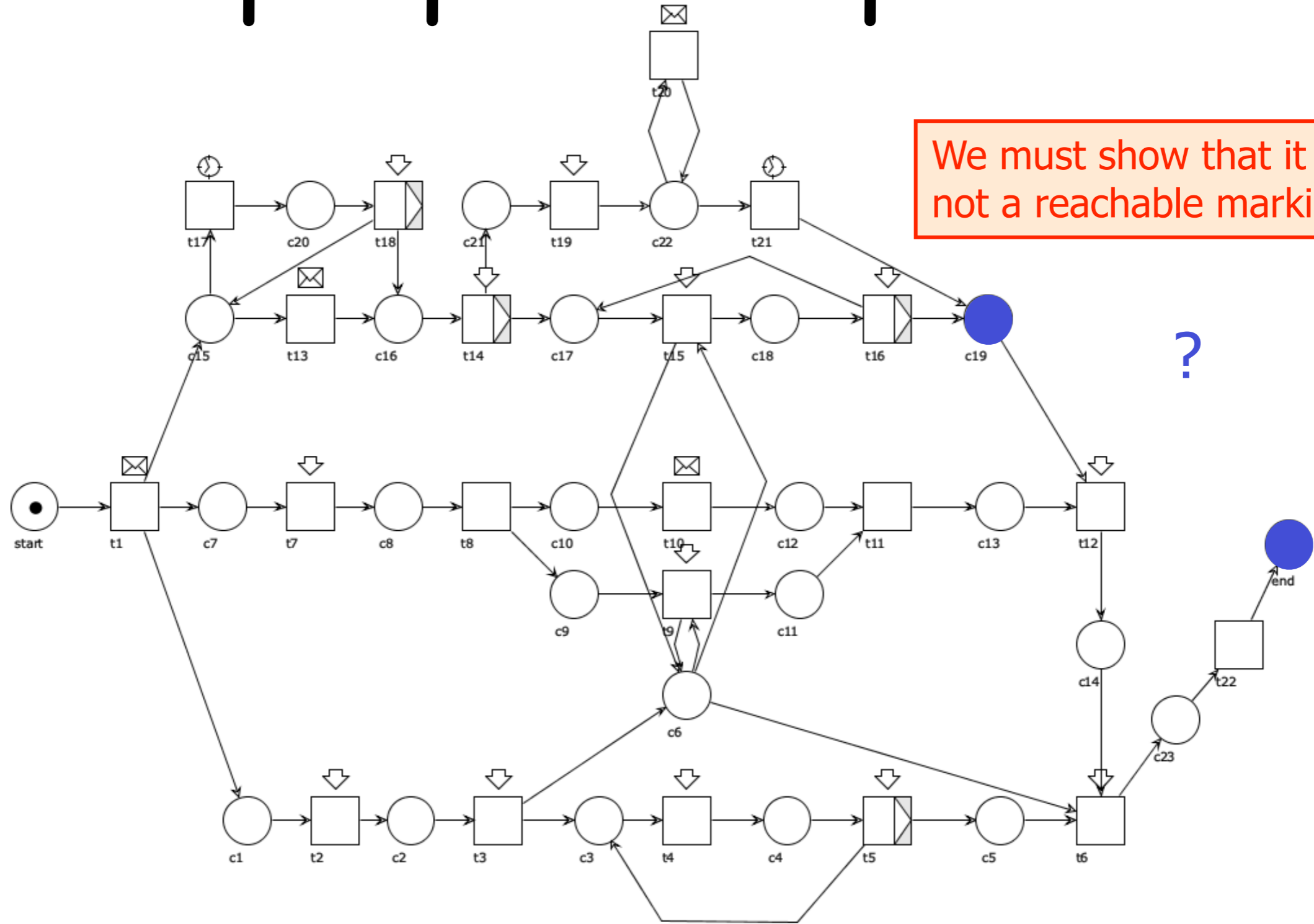
# 2: option to complete



# 2: option to complete

The check must be repeated for each reachable marking

# 3: proper completion



# 3: proper completion

The check must be repeated for each marking  $M$   
such that  $M > 0$

# Brute-force analysis

First, check if the Petri net is a **workflow net**  
easy "structural" check

Second, check if it is **sound** (more difficult):  
build the Reachability Graph

**to check 1:** for each transition  $t$  there must be an arc in the  
RG that is labelled with  $t$

**to check 2&3:** the RG must have only one final state (sink),  
that consists of one token in  $o$   
and is reachable from any other state,  
and no other marking has a token in  $o$

# Some Pragmatic Considerations

All checks can better be done automatically  
(computer aided)

but nevertheless RG construction...

1. can be computationally expensive for large nets  
(because of state explosion)
2. provides little support in repairing unsound processes
3. can be infinite (CG can be used, but it is not exact)



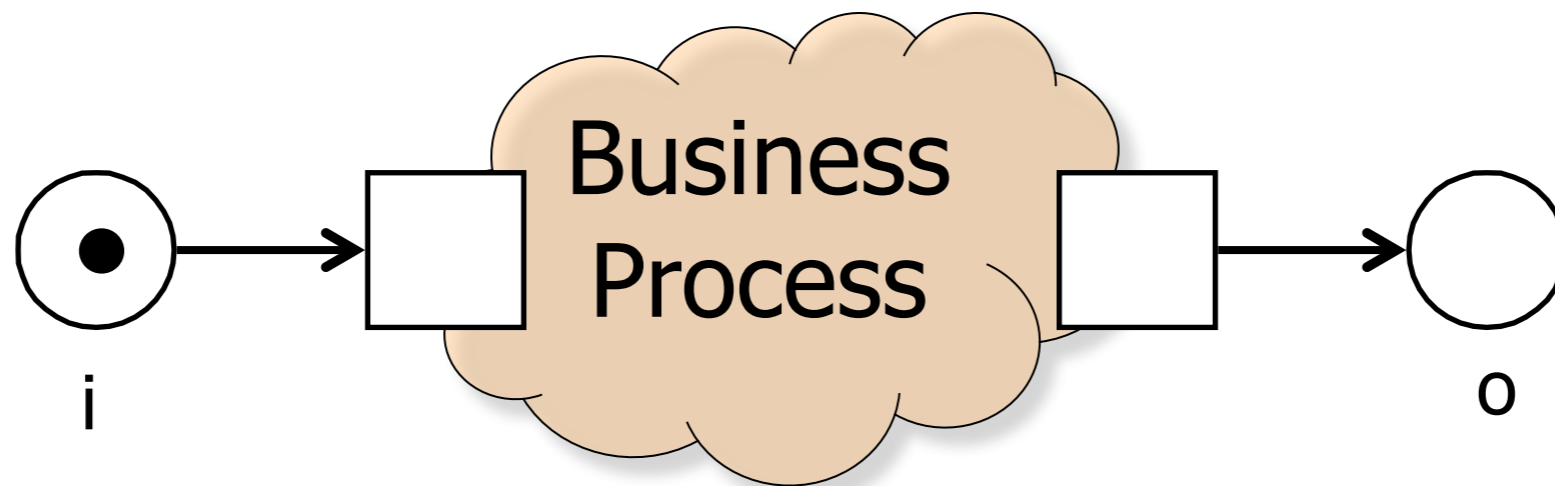
# Advanced support

Translate soundness to other well-known properties that can be checked more efficiently:

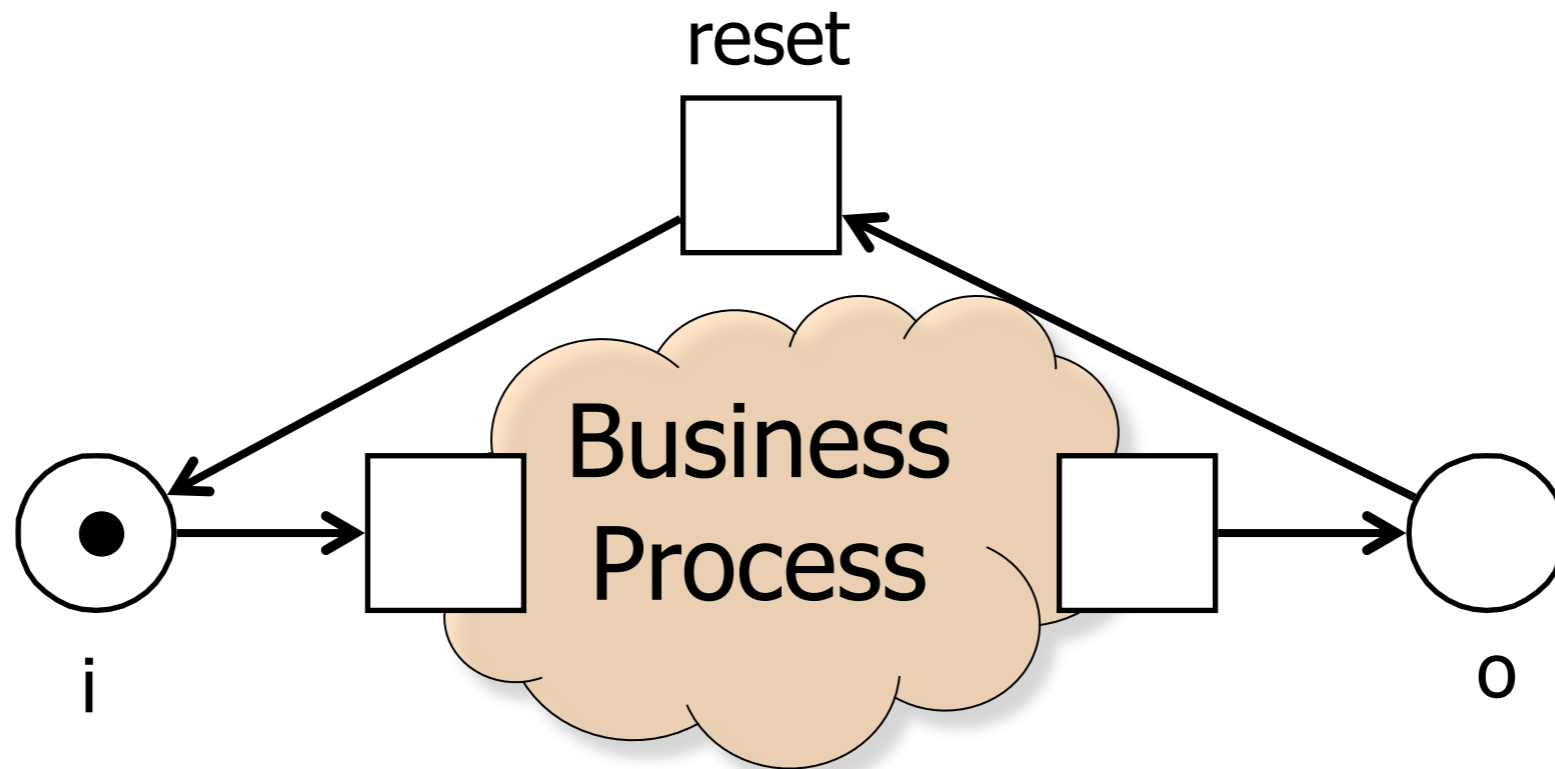
boundedness and liveness

$N^*$

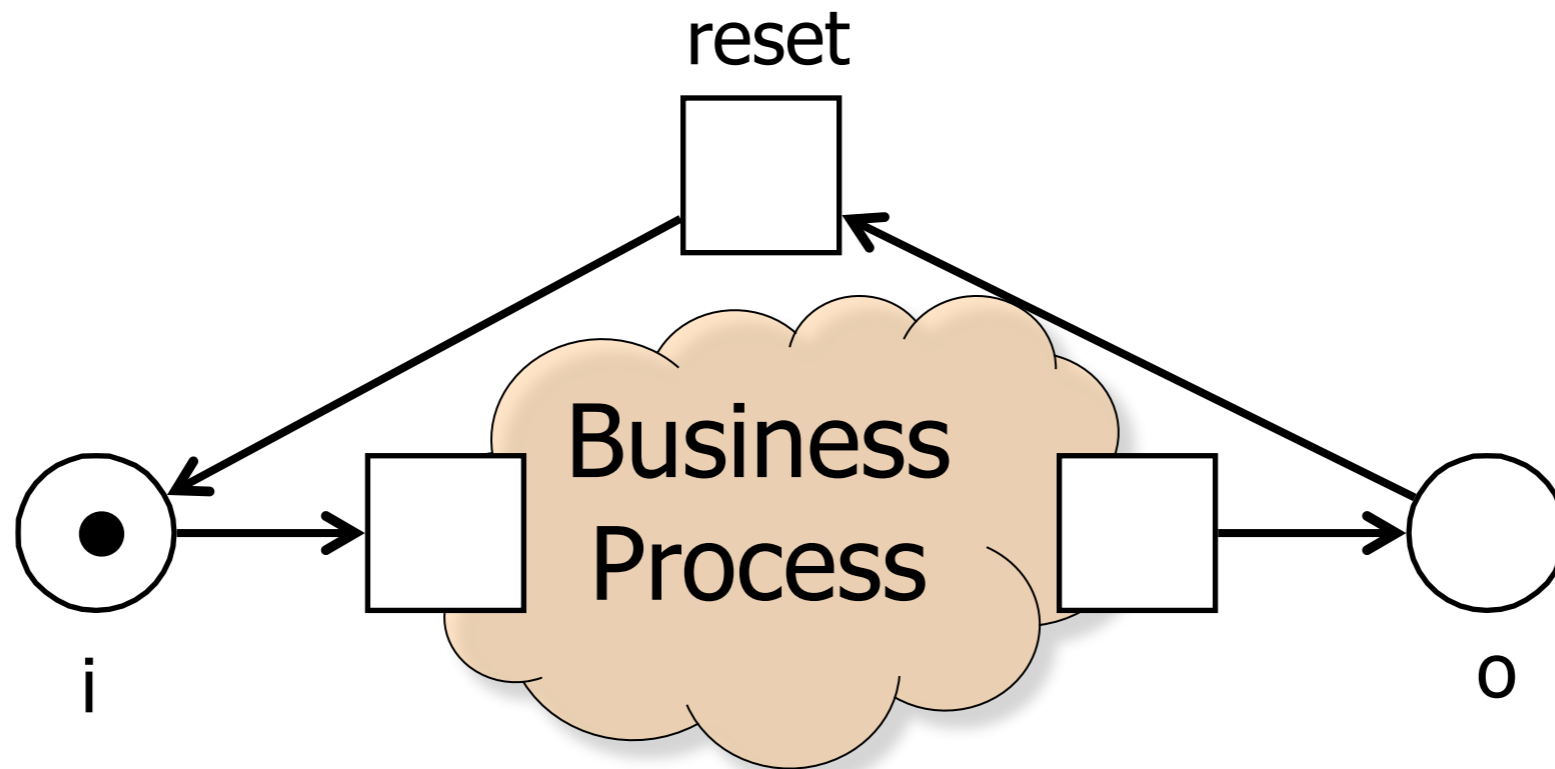
# Play once



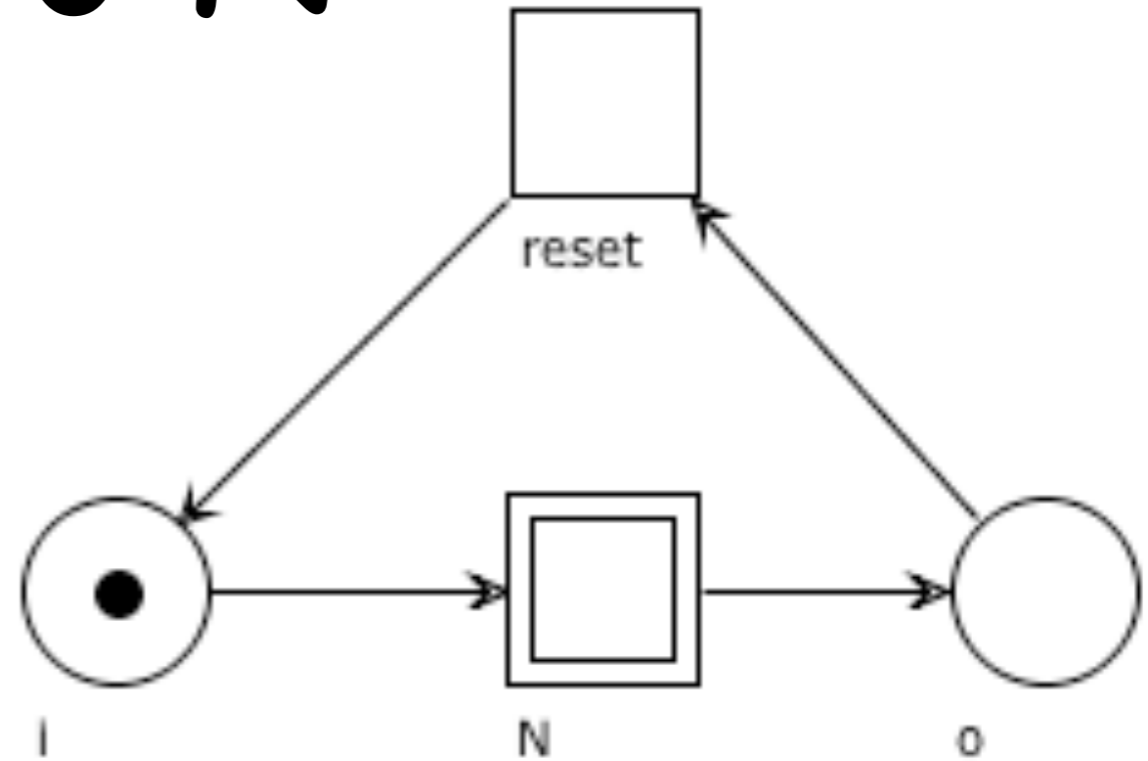
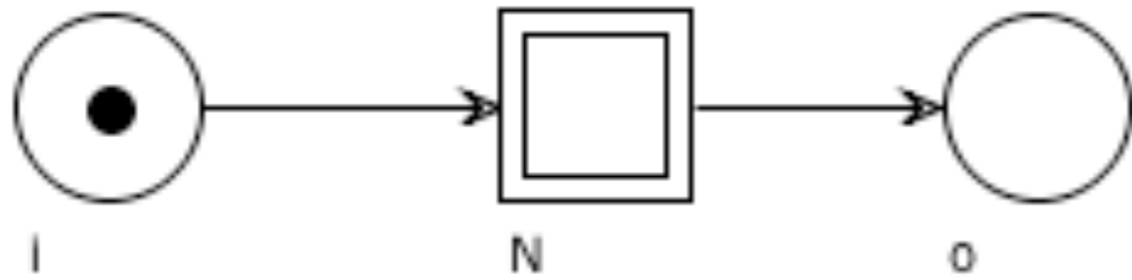
# Play twice



# Play any number of times



# From $N$ to $N^*$



Let us denote by  $N : i \rightarrow o$  a workflow net with entry place  $i$  and exit place  $o$ .

Let  $N^*$  be the net obtained by adding the “*reset*” transition to  $N$   
 $reset : o \rightarrow i$ .

# MAIN THEOREM

**Theorem:**

$N$  is sound iff  $N^*$  is live and bounded

# Proof of MAIN THEOREM (1)

$N^*$  **live and bounded** implies  $N$  **sound**:

Since  $N^*$  is **live**: for each  $t \in T$  there is  $M \in [i \rangle$ .  $M \xrightarrow{t}$

Take any  $M \in [i \rangle$  enabling  $reset : o \rightarrow i$ , hence  $M \supseteq o$

Let  $M \xrightarrow{reset} M'$ . Then  $M' \in [i \rangle$  and  $M' \supseteq i$

Since  $N^*$  is bound, it must be  $M' = i$  (and  $M = o$ )

Otherwise all places marked by  $M' - i = M - o$  would be unbounded

Hence  $N^*$  just allows multiple runs of  $N$ :

"option to complete" and "proper completion" hold (see above)

"no dead task" holds because  $N^*$  is live



# A technical lemma

## Lemma:

If  $N$  is sound,  $M$  is reachable in  $N$  iff  $M$  is reachable in  $N^*$

$\Rightarrow$ ) straightforward

$\Leftarrow$ ) Let  $i \xrightarrow{\sigma} M$  in  $N^*$  for  $\sigma = t_1 t_2 \dots t_n$

We proceed by induction on the number  $r$  of instances of *reset* in  $\sigma$

If  $r = 0$ , then *reset* does not occur in  $\sigma$  and  $M$  is reachable in  $N$

If  $r > 0$ , let  $k$  be the least index such that  $t_k = \textit{reset}$

Let  $\sigma = \sigma' t_k \sigma''$  with  $\sigma' = t_1 t_2 \dots t_{k-1}$  fireable in  $N$

Since  $N$  is sound:  $i \xrightarrow{\sigma'} o$  and  $i \xrightarrow{\sigma''} M$

Since  $\sigma''$  contains  $r - 1$  instances of *reset*:

by inductive hypothesis  $M$  is reachable in  $N$

# Proof of MAIN THEOREM (2)

**$N$  sound implies  $N^*$  bounded :**

We proceed by contradiction, assuming  $N^*$  is unbounded

Since  $N^*$  is unbounded:

$\exists M, M'$  such that  $i \rightarrow^* M \rightarrow^* M'$  with  $M \subset M'$

Let  $L = M' - M \neq \emptyset$

Since  $N$  is sound:

$\exists \sigma \in T^*$  such that  $M \xrightarrow{\sigma} o$

By the monotonicity Lemma:  $M' \xrightarrow{\sigma} o + L$  and thus  $o + L \in [i \rangle$

Which is absurd, because  $N$  is sound

# Proof of MAIN THEOREM (3)

**$N$  sound implies  $N^*$  live:**

Take any transition  $t$  and let  $M$  be a marking reachable in  $N^*$

By the technical lemma,  $M$  is reachable in  $N$

Since  $N$  is sound:  $\exists \sigma \in T^*$  with  $M \xrightarrow{\sigma} o$

Since  $N$  is sound:  $\exists \sigma' \in T^*$  with  $i \xrightarrow{\sigma'} M'$  and  $M' \xrightarrow{t}$

Let  $\sigma'' = \sigma \text{ reset } \sigma'$ , then:

$M \xrightarrow{\sigma''} M'$  in  $N^*$  and  $M' \xrightarrow{t}$

# Recall: consequences of strong connectedness theorem

If a (weakly-connected) net is not strongly connected

then

It is not “live and bounded”

If it is live, it is not bounded

If it is bounded, it is not live

# Strong connectedness of $N^*$

**Proposition:**

$N^*$  is strongly connected.

Take two nodes of  $(x, y) \in F_{N^*}$ ,  
we want to build a path from  $y$  to  $x$

If  $x, y \neq reset$ , then

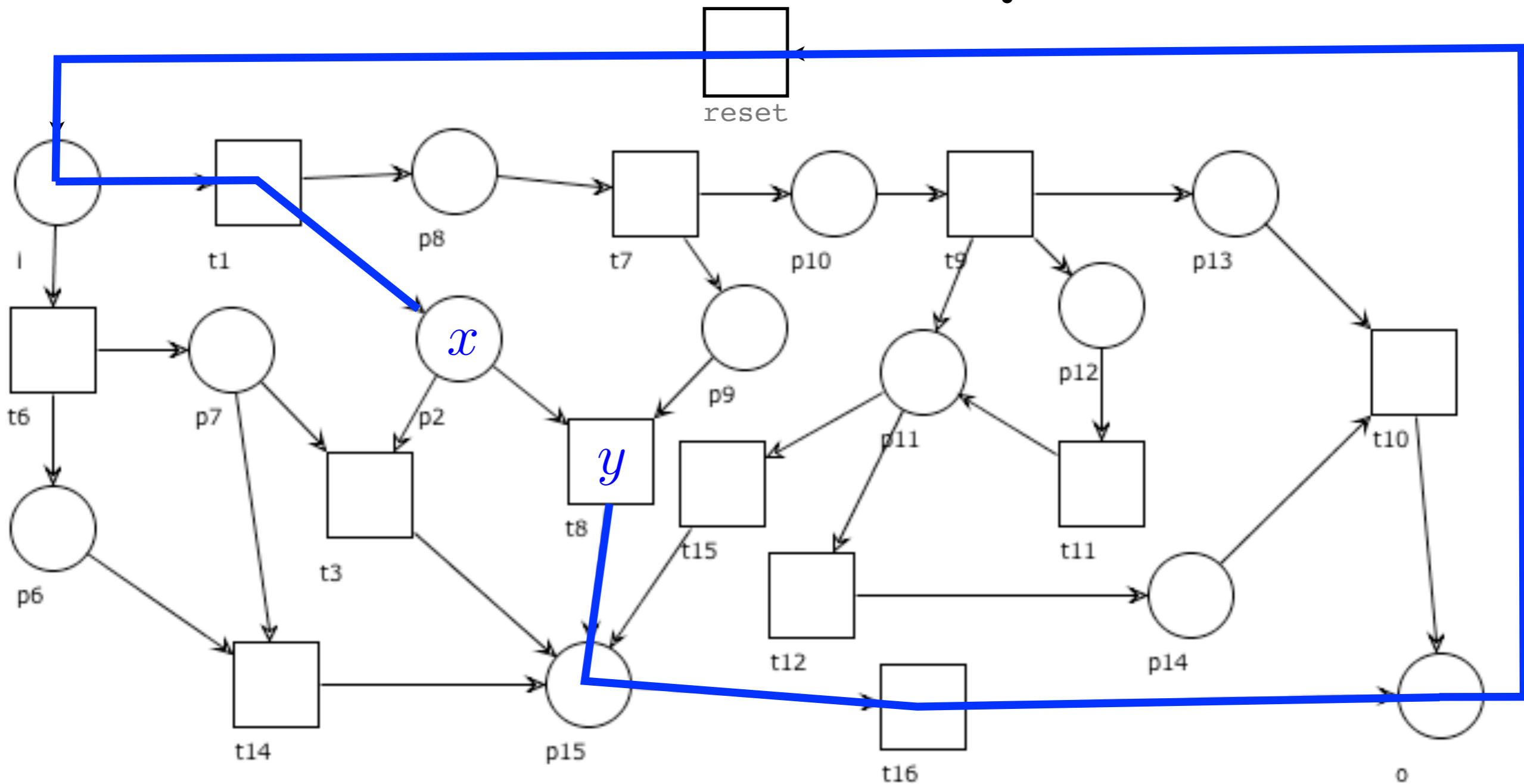
$y$  lies on a path  $i \rightarrow^* \boxed{y \rightarrow^* o}$ , because  $N$  is a workflow net,  
 $x$  lies on a path  $\boxed{i \rightarrow^* x} \rightarrow^* o$ , because  $N$  is a workflow net,  
 we combine the paths  $y \rightarrow^* o \rightarrow reset \rightarrow i \rightarrow^* x$

If  $x = o, y = reset$ , then  
 take any path  $i \rightarrow^* o$ ,

we build the path  $reset \rightarrow i \rightarrow^* o$

If  $x = reset, y = i$ , then  
 take any path  $i \rightarrow^* o$ ,  
 we build the path  $i \rightarrow^* o \rightarrow reset$

# Strong connectedness of $N^*$ : example



<http://woped.dhbw-karlsruhe.de/woped/>

# WoPeD

Workflow Petri Net Designer

*Download WoPeD at sourceforge!*



WoPeD 3.2.0

File Edit Analysis & Help Community

Tokengame Operator coloring Semantical analysis Capacity planning Quantitative simulation Coverability graph Show metrics Metrics mass analysis Metrics builder

Process Resources BPEL preview

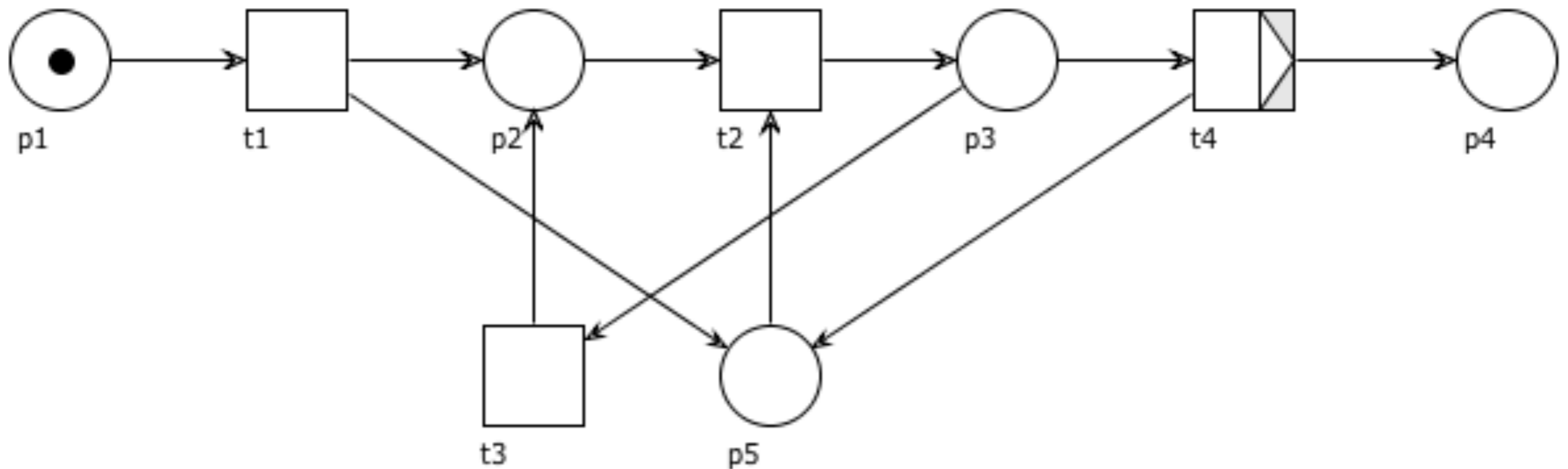
08-cube.pnml

Horizontal Zoom: 100%

08-cube.pnml

# Exercise

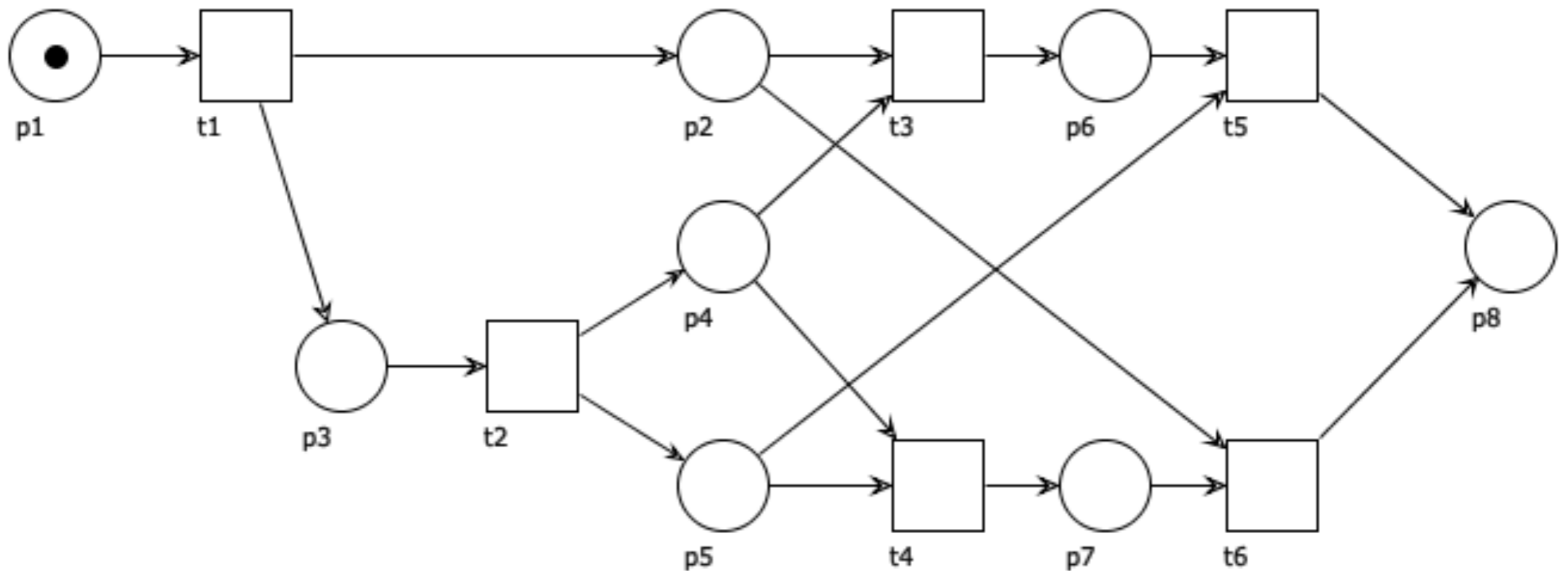
Use some tools to check if the net below is a sound workflow net or not





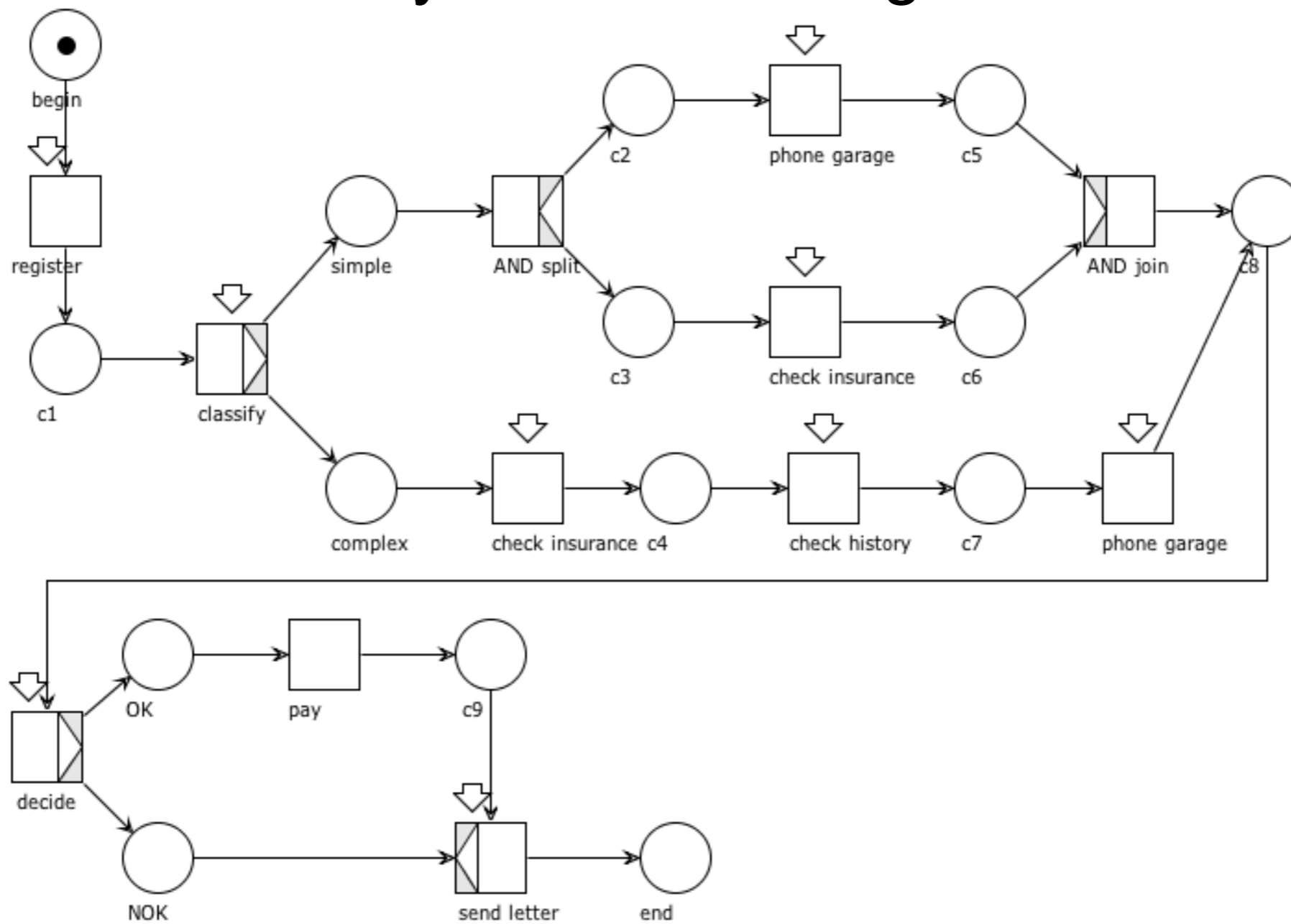
# Exercise

Use some tools to check if the net below is a sound workflow net or not



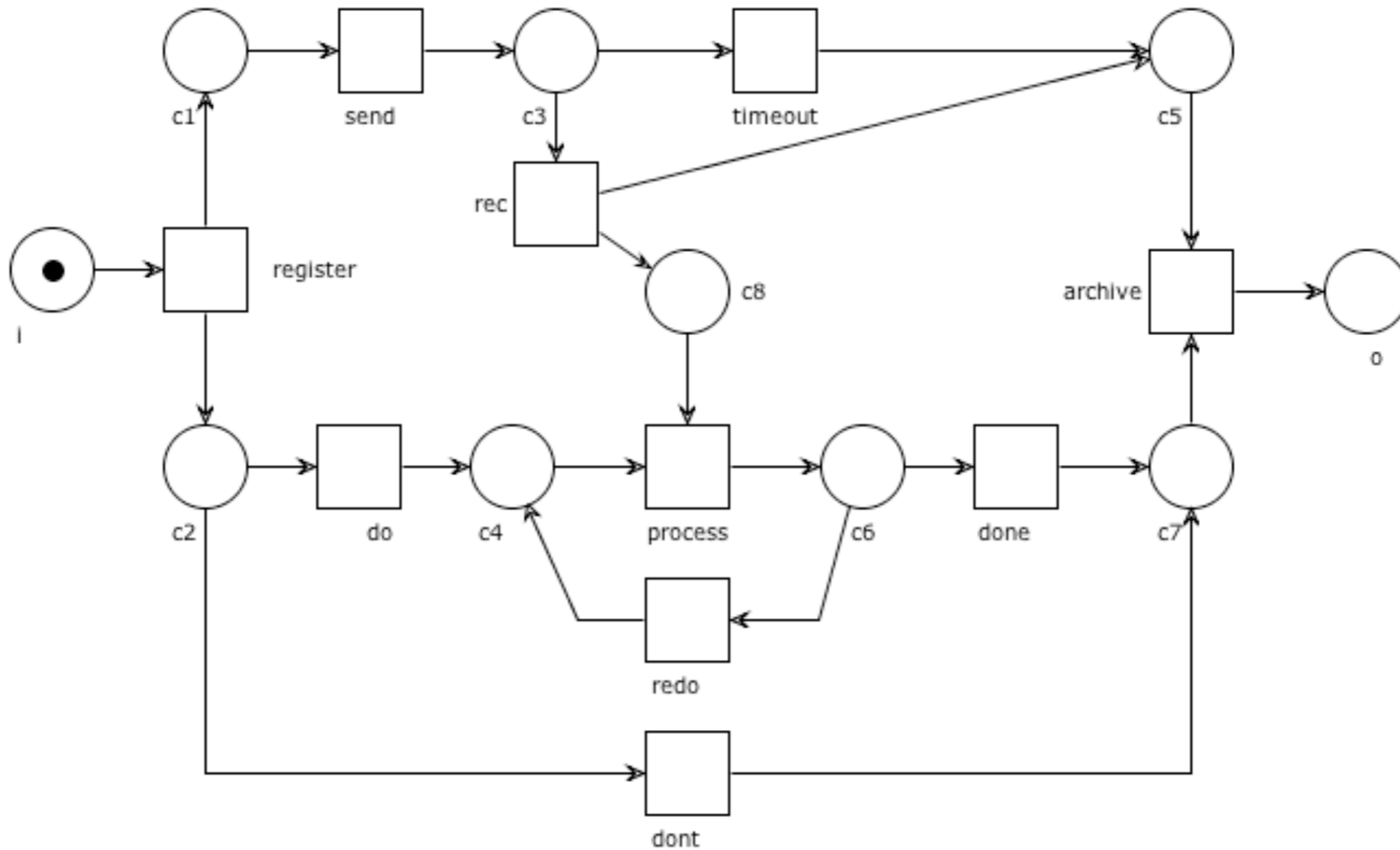
# Exercise

Analyse the following net



# Exercise

Analyse the following net



# Design and analysis of WF-nets

The workflow of a computer repair service (CRS) can be described as follows. A customer brings in a defective computer and the CRS checks the defect and hands out a repair cost calculation back.

If the customer decides that the costs are acceptable, the process continues, otherwise she takes her computer home, unrepaired.

The ongoing repair consists of two activities, which are executed sequentially but in an arbitrary order.

One activity is to check and repair the hardware, whereas the other activity is to check and configure the software.

After both activities are completed, the proper system functionality is tested.

If an error is detected the repair procedure is repeated, otherwise the repair is finished and the computer is returned.

Model the described workflow as a sound workflow net.