

# Business Processes Modelling

## MPB (6 cfu, 295AA)

Roberto Bruni

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

19 - Diagnosis for WF nets



# Object



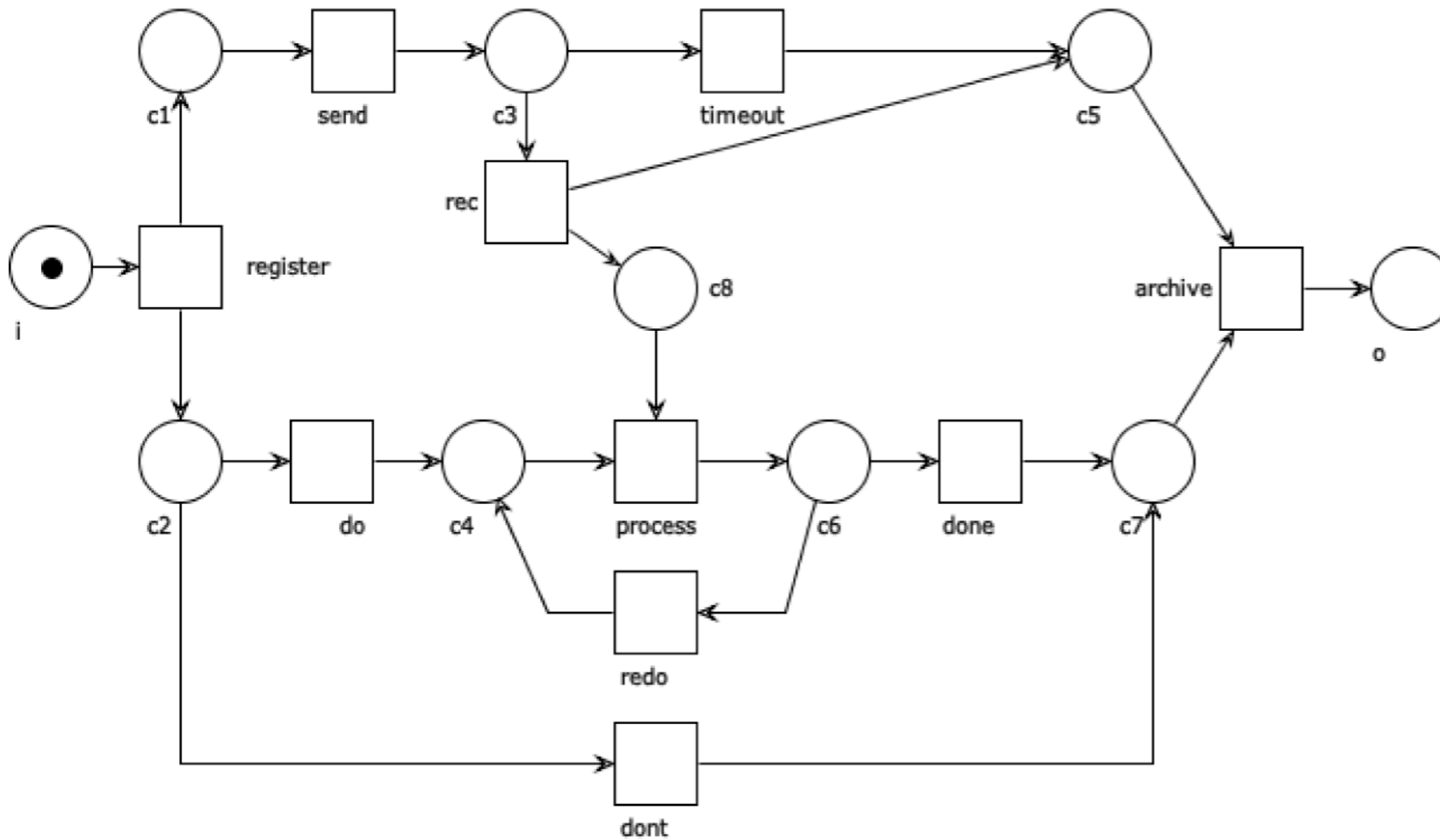
We study suitable diagnosis techniques  
for unsound Workflow nets

Diagnosing workflow processes using Woflan (article, optional reading)

<http://wwwis.win.tue.nl/~wvdaalst/publications/p135.pdf>

# Woped

what are S-components?  
and why are they relevant?



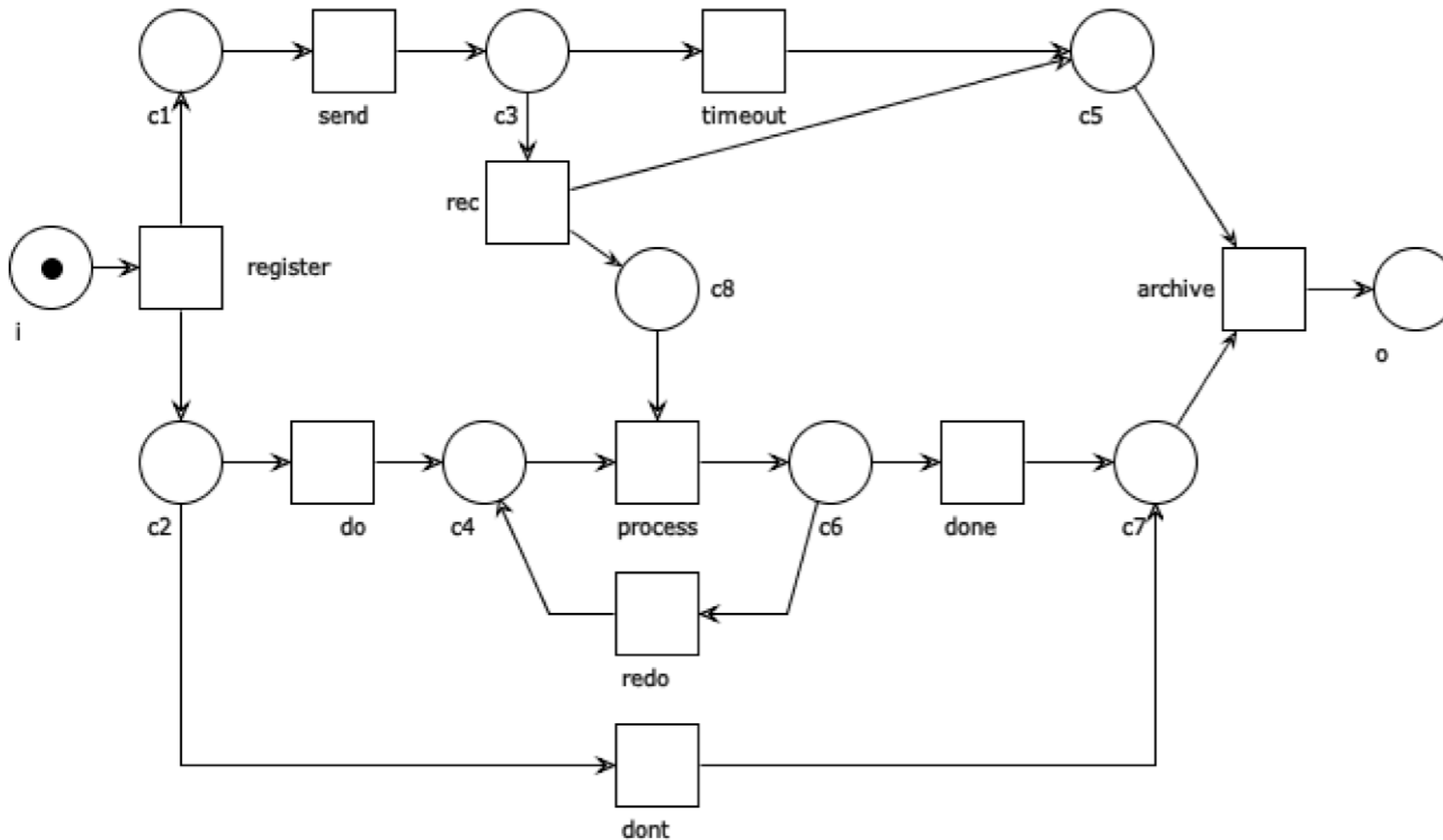
## Semantical analysis

Wizard Expert

- Qualitative analysis
  - Structural analysis
    - Net statistics
      - Places: 10
      - Transitions: 10
      - Operators: 0
      - Subprocesses: 0
      - Arcs: 24
    - Wrongly used operators: 0
    - Free-choice violations: 0
    - S-Components**
      - S-Components: 2
      - Places not covered by S-Component: 1
        - c8
    - Wellstructuredness
      - PT-Handles: 4
      - TP-Handles: 5
    - Soundness
      - Workflow net property
      - Initial marking
      - Boundedness
      - Liveness

# Woped

what are PT/TP-handles?  
and why are they relevant?



## Semantical analysis

Wizard Expert

- Qualitative analysis
  - Structural analysis
    - Net statistics
      - Places: 10
      - Transitions: 10
      - Operators: 0
      - Subprocesses: 0
      - Arcs: 24
    - Wrongly used operators: 0
    - Free-choice violations: 0
  - S-Components
    - S-Components: 2
    - Places not covered by S-Component: 1
      - c8
  - Wellstructuredness
    - PT-Handles: 4
    - TP-Handles: 5
  - Soundness
    - Workflow net property
    - Initial marking
    - Boundedness
    - Liveness

# S-Coverability

# Rank Theorem

(main result, proof omitted)

## Theorem:

A free-choice system  $(P, T, F, M_0)$  is live and bounded  
**iff**

1. it has at least one place and one transition
2. it is connected
3.  $M_0$  marks every proper siphon
- 4. it has a positive S-invariant**
5. it has a positive T-invariant
6.  $\text{rank}(N) = |C_N| - 1$

(where  $C_N$  is the set of clusters)

# A technique to find a positive $S$ -invariant

A case is often composed by parallel threads of control  
(each thread imposing some order over its tasks)

Decompose the net  $N$  in suitable  $S$ -nets  
so that any place of  $N$  belongs to some  $S$ -net  
(the same place can appear in more  $S$ -nets)

Each  $S$ -net induces a uniform  $S$ -invariant

A positive  $S$ -invariant is obtained  
as the sum of the  $S$ -invariants of each subnet

# S-component

take a set of nodes

**Definition:** Let  $N = (P, T, F)$  and  $\emptyset \subset X \subseteq P \cup T$

Let  $N' = (P \cap X, T \cap X, F \cap (X \times X))$  be a subnet of  $N$ .

$N'$  is an **S-component** if

forget the arcs to other nodes

1. it is a strongly connected S-net

2. for every place  $p \in X \cap P$ , we have  $\bullet p \cup p \bullet \subseteq X$

if a place  $p$  is taken

then all transitions attached to  $p$  must be selected



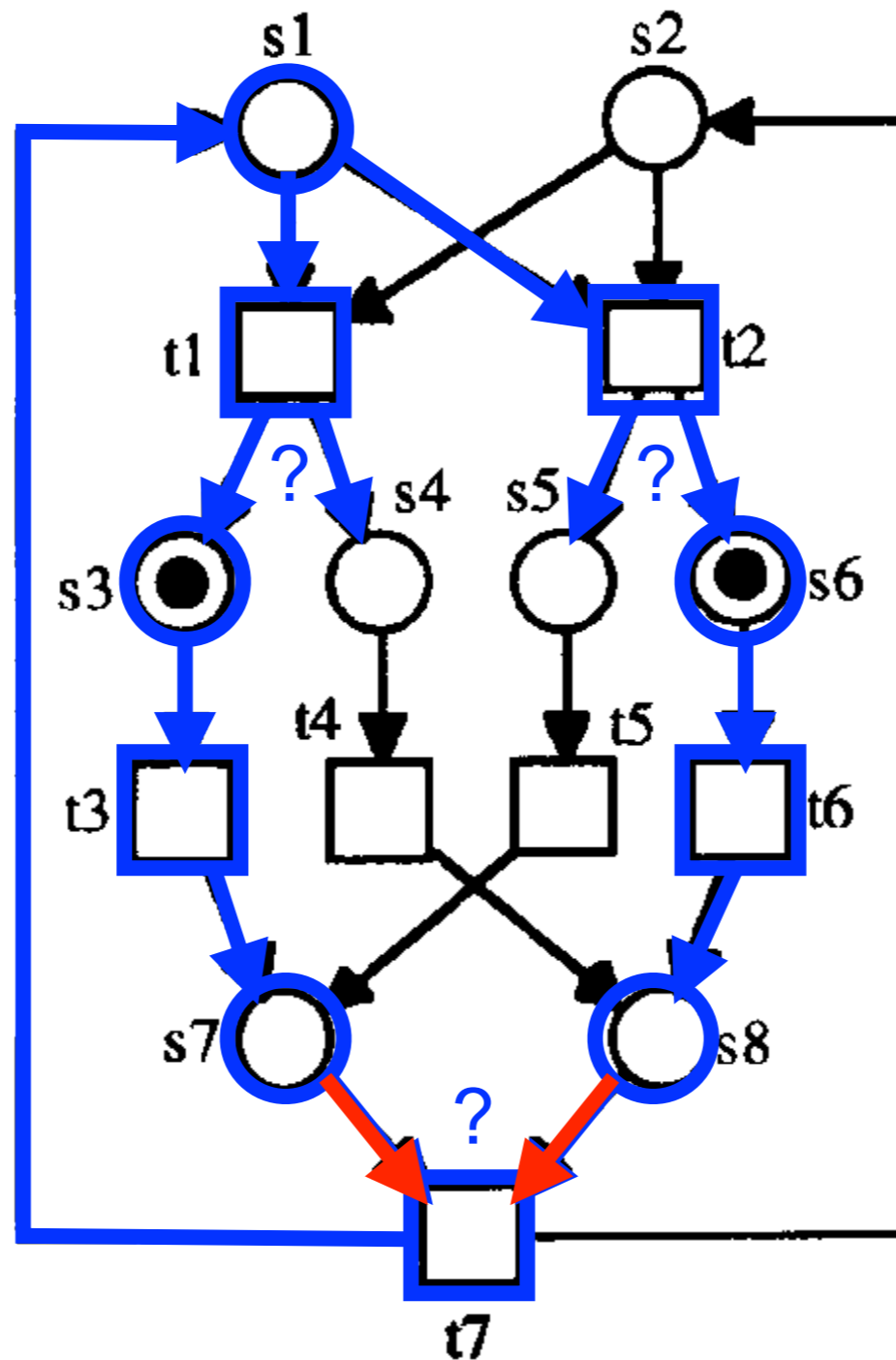
# S-cover

**Definition:** an **S-cover** of a net  $N$  is a set  $C$  of  $S$ -components of  $N$  such that every place  $p$  of  $N$  belongs to one or more  $S$ -components in  $C$

$N$  is **S-coverable** if it has an  $S$ -cover

if a place  $p$  is taken  
then all transitions attached to  $p$  must be selected

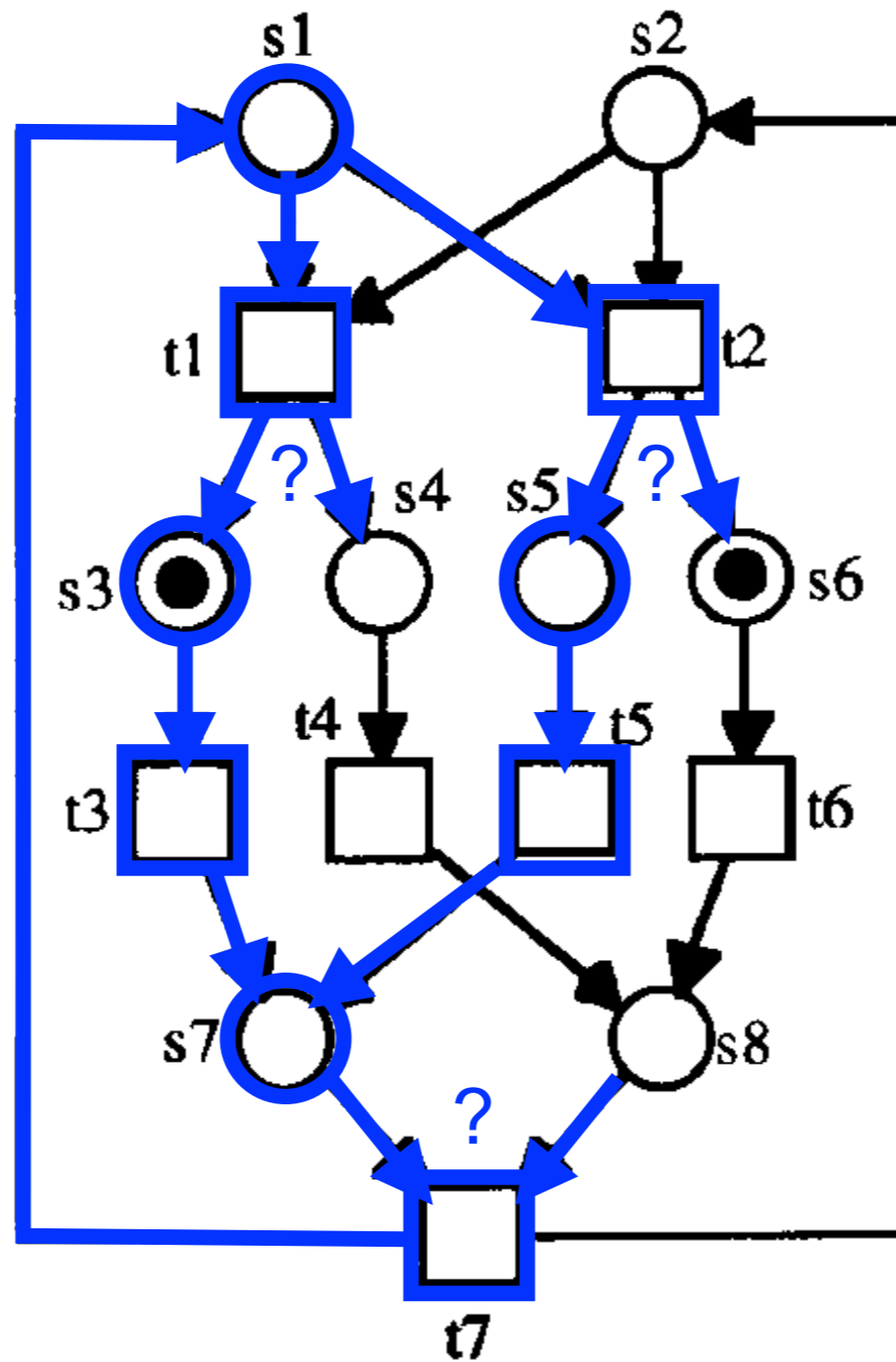
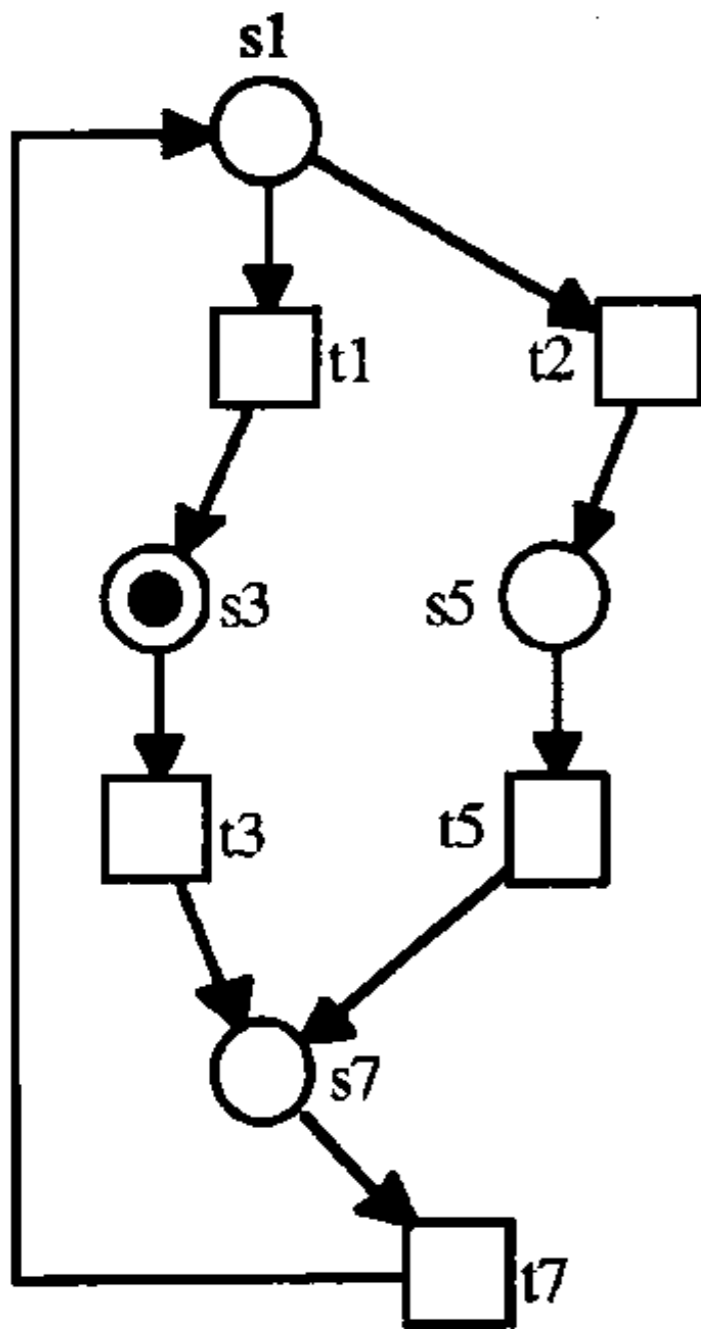
# S-cover: example



not an S-net

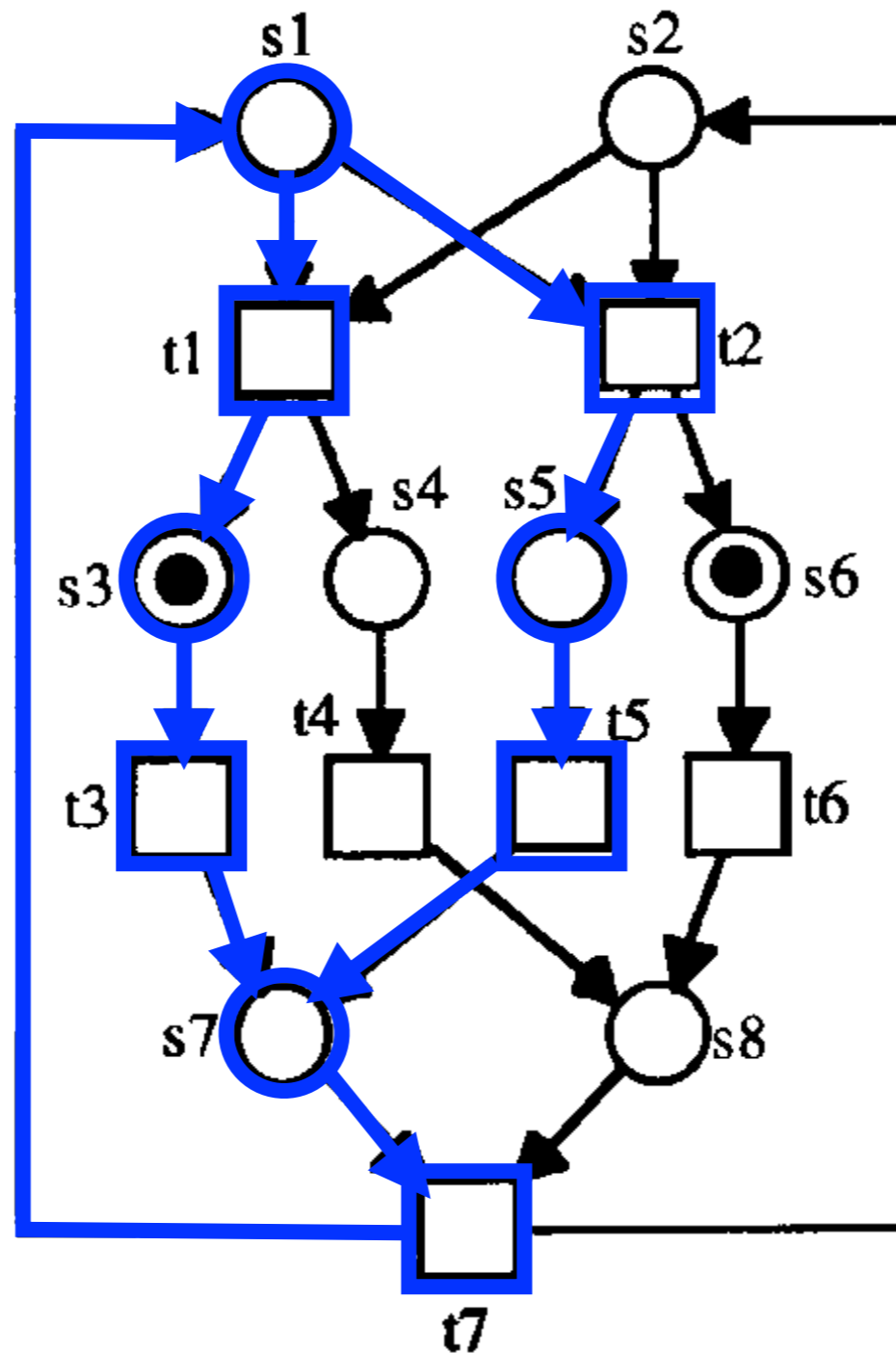
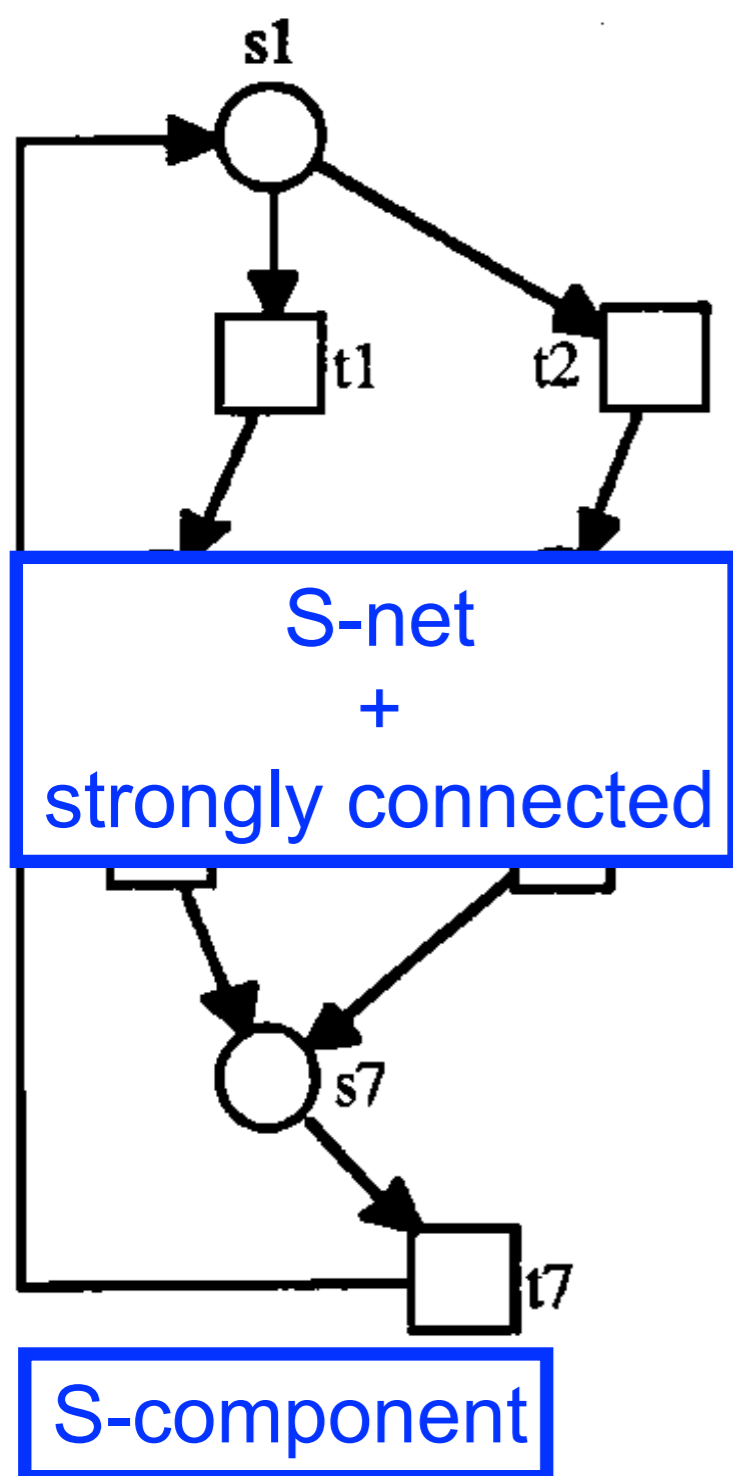
if a place  $p$  is taken  
then all transitions attached to  $p$  must be selected

# S-cover: example



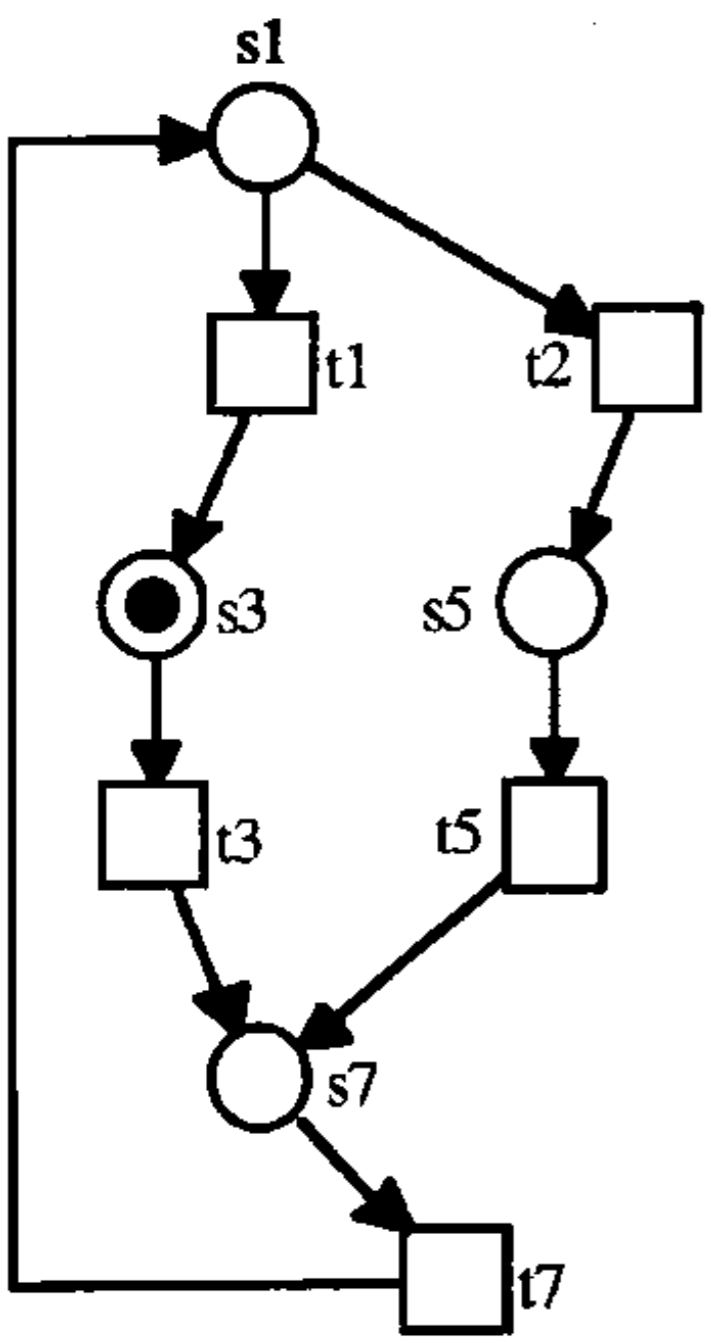
if a place  $p$  is taken  
then all transitions attached to  $p$  must be selected

# S-cover: example

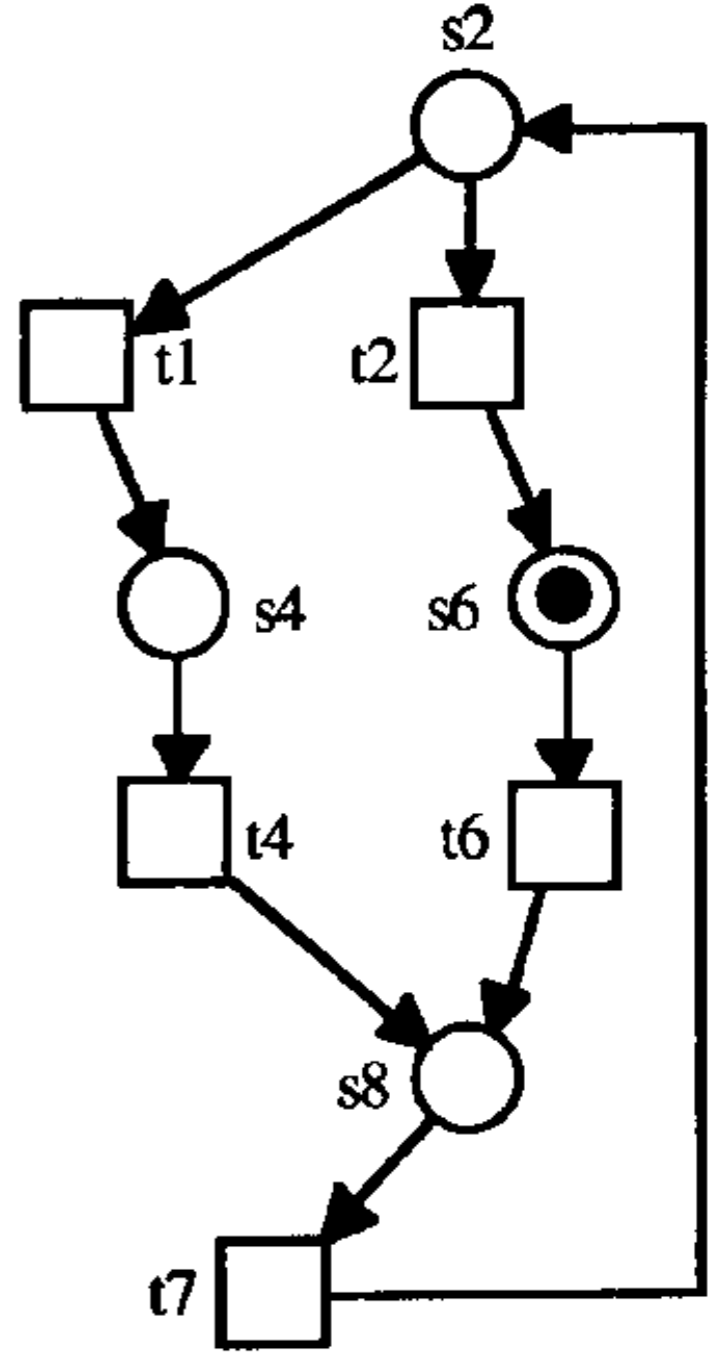
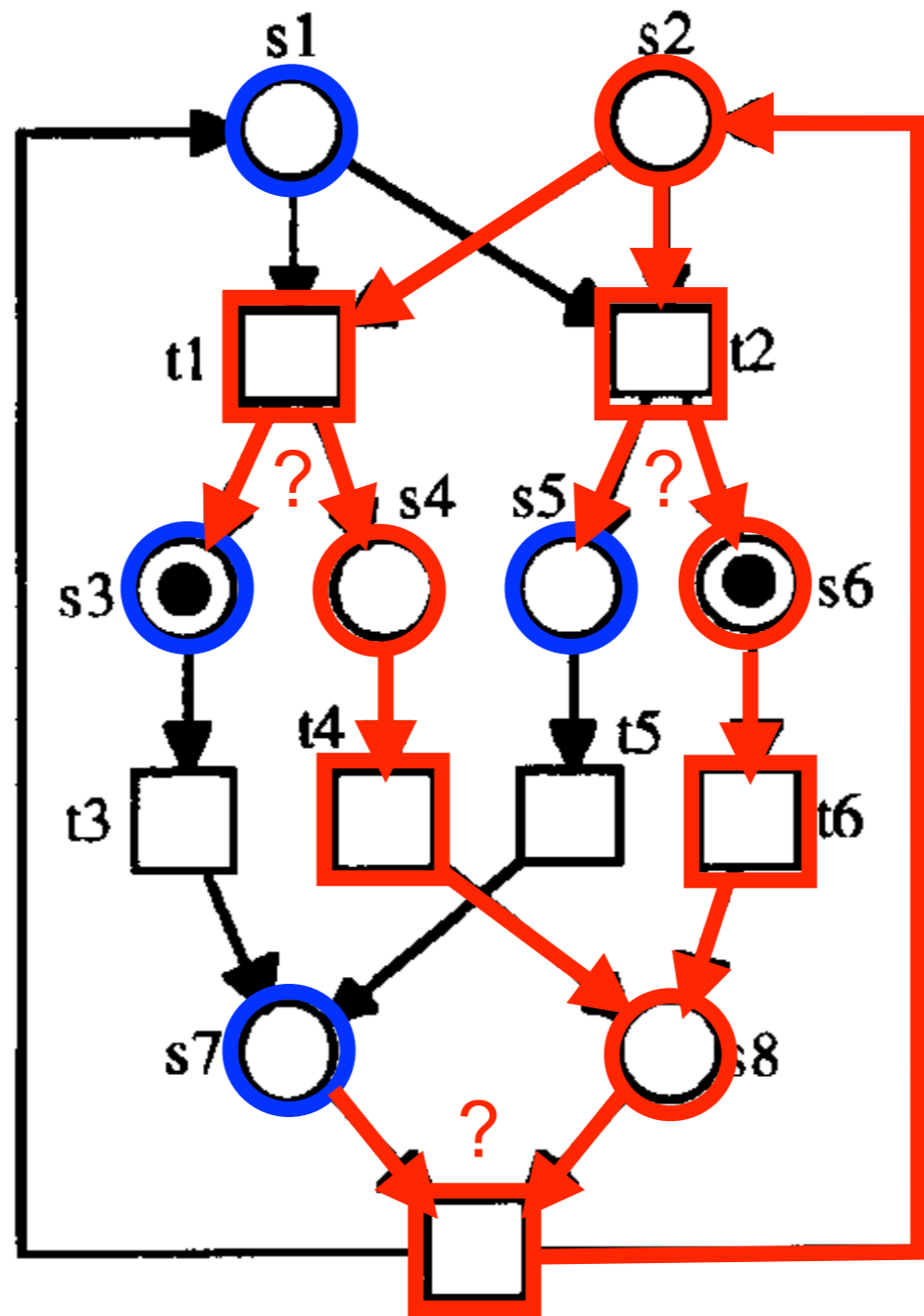


if a place  $p$  is taken  
 then all transitions attached to  $p$  must be selected

# S-cover: example

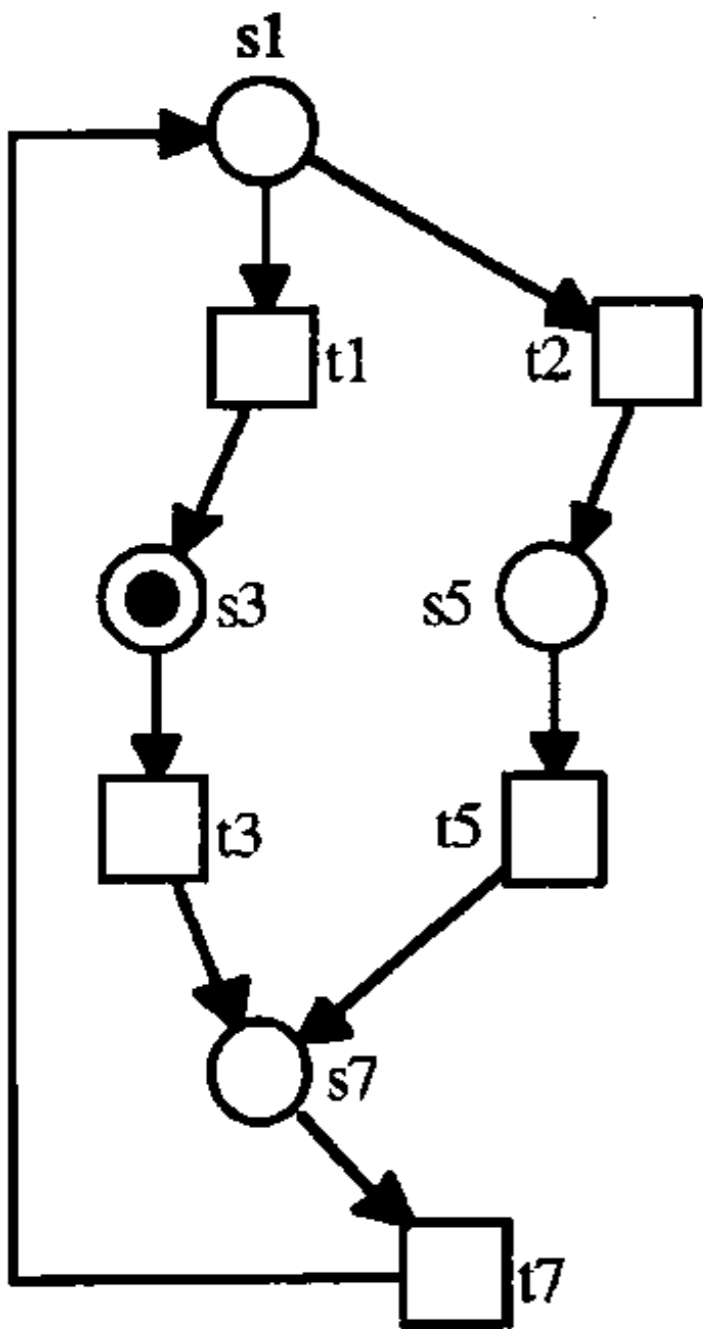


S-component

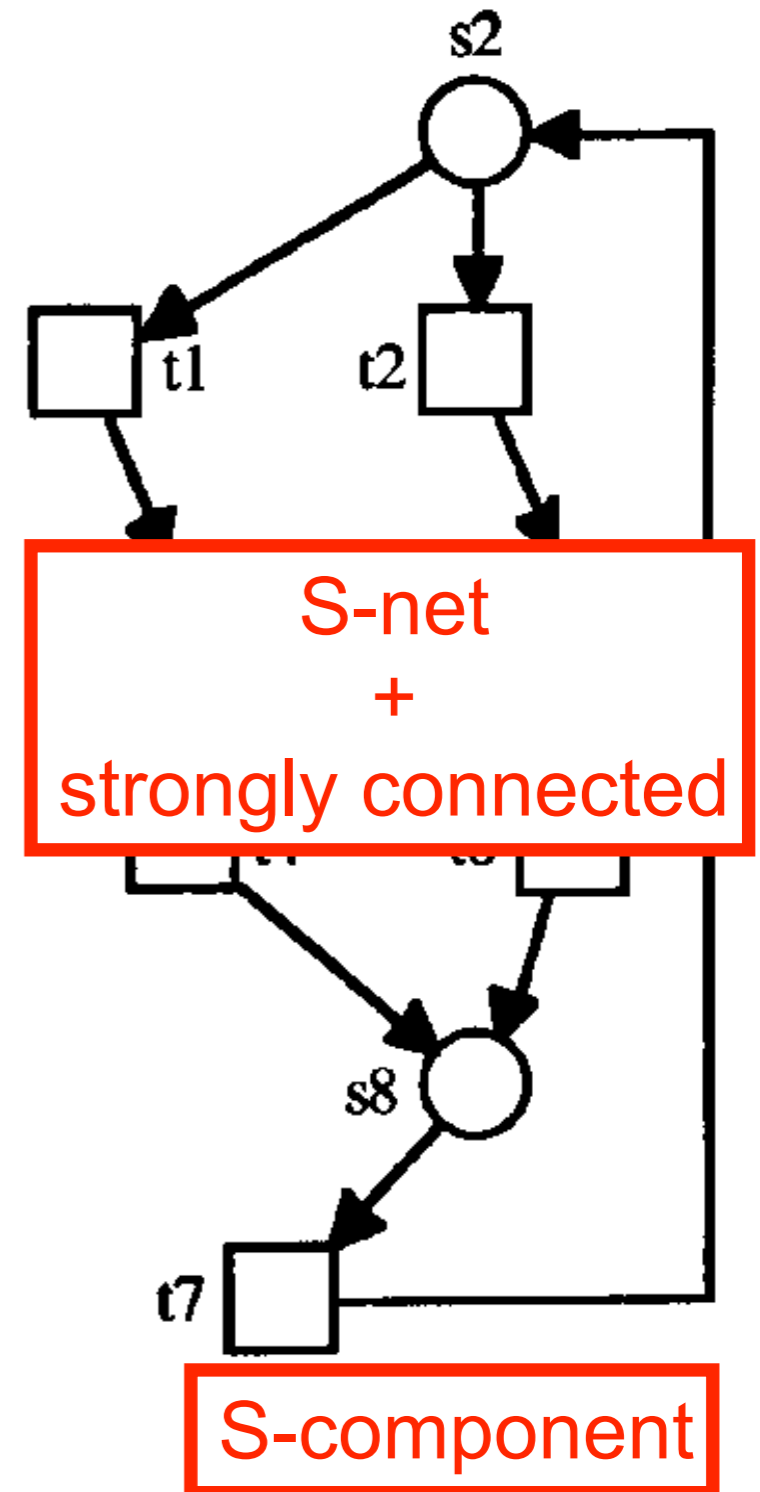
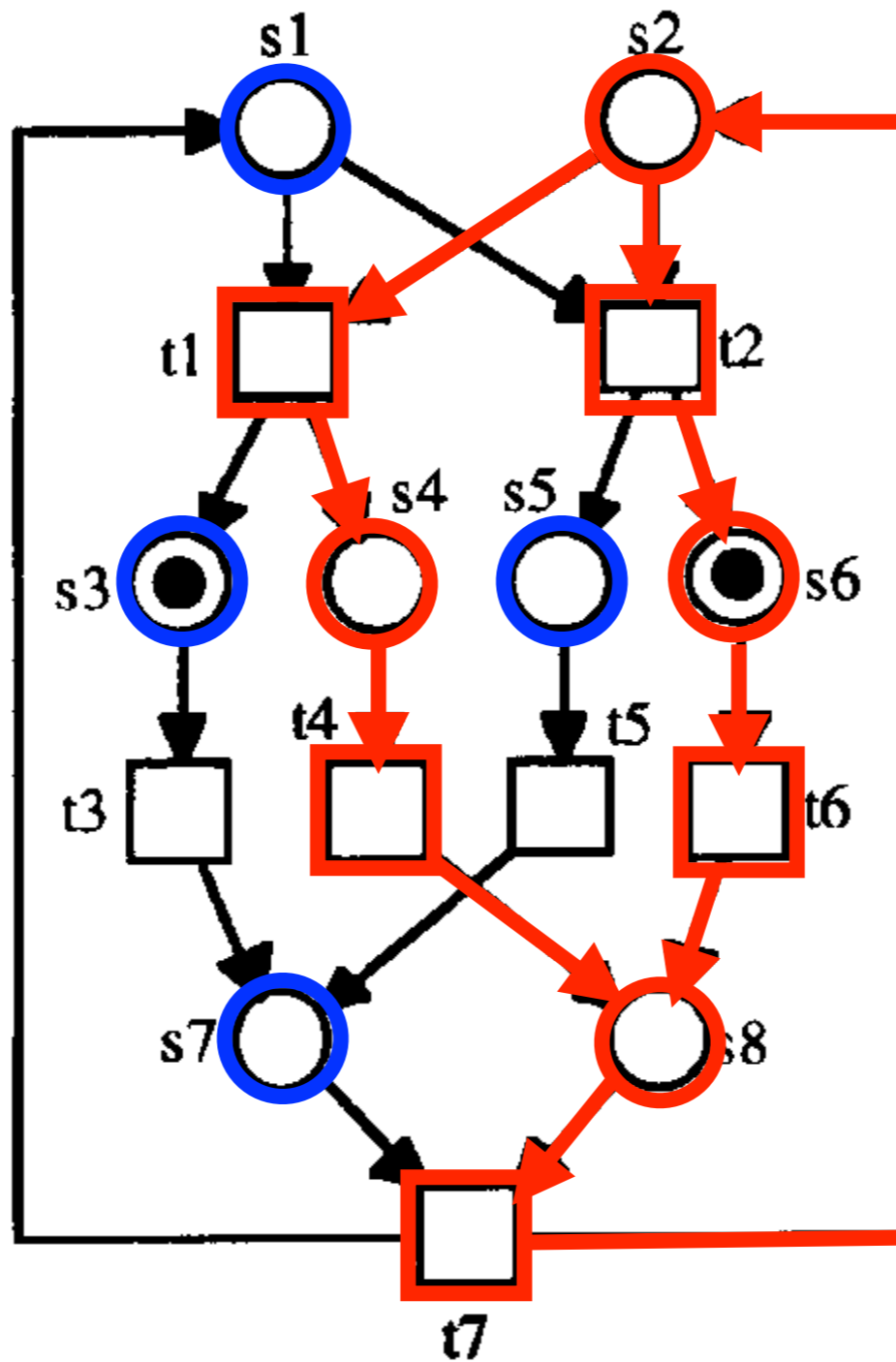


if a place  $p$  is taken  
 then all transitions attached to  $p$  must be selected

# S-cover: example



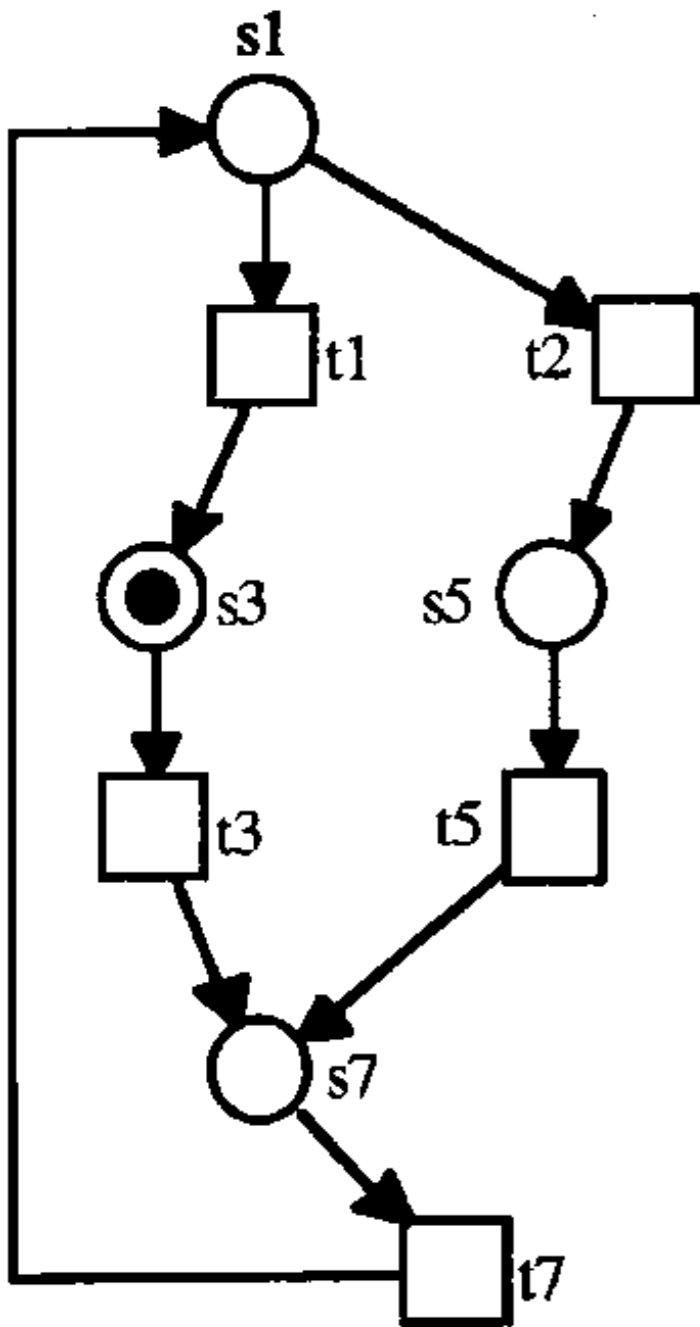
S-component



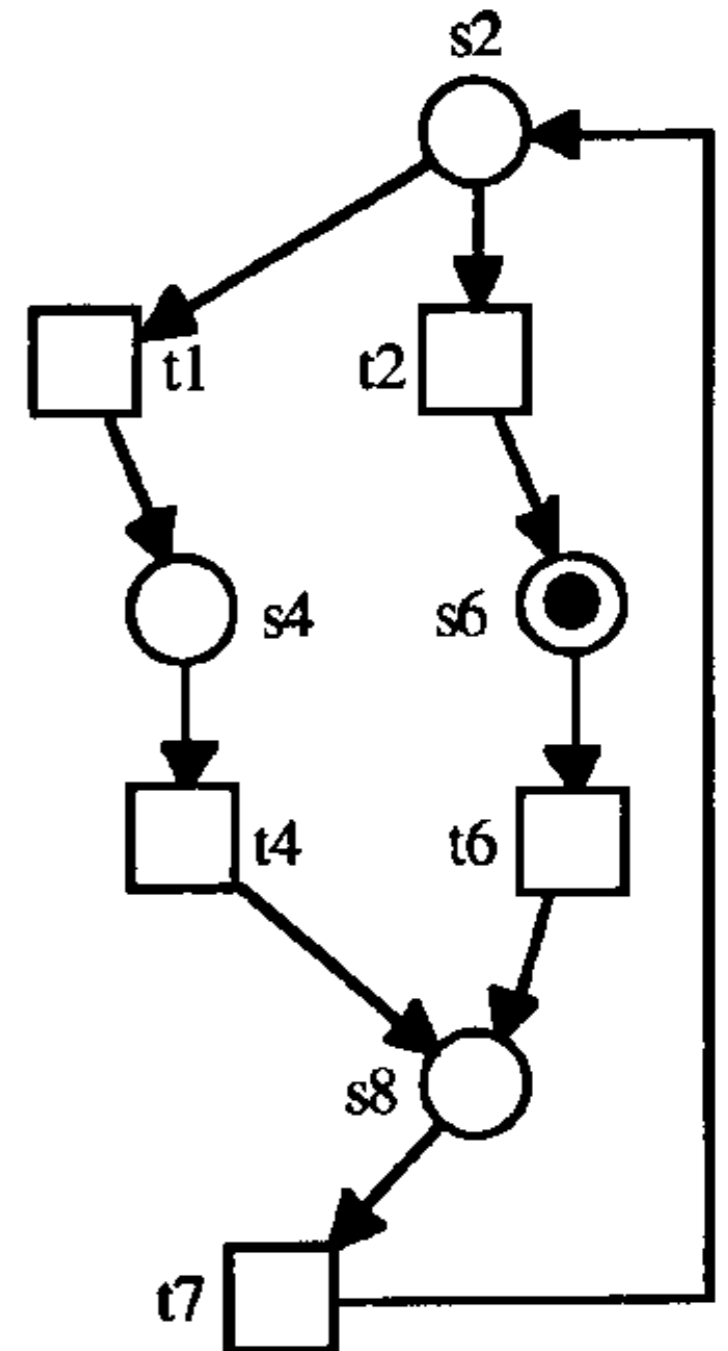
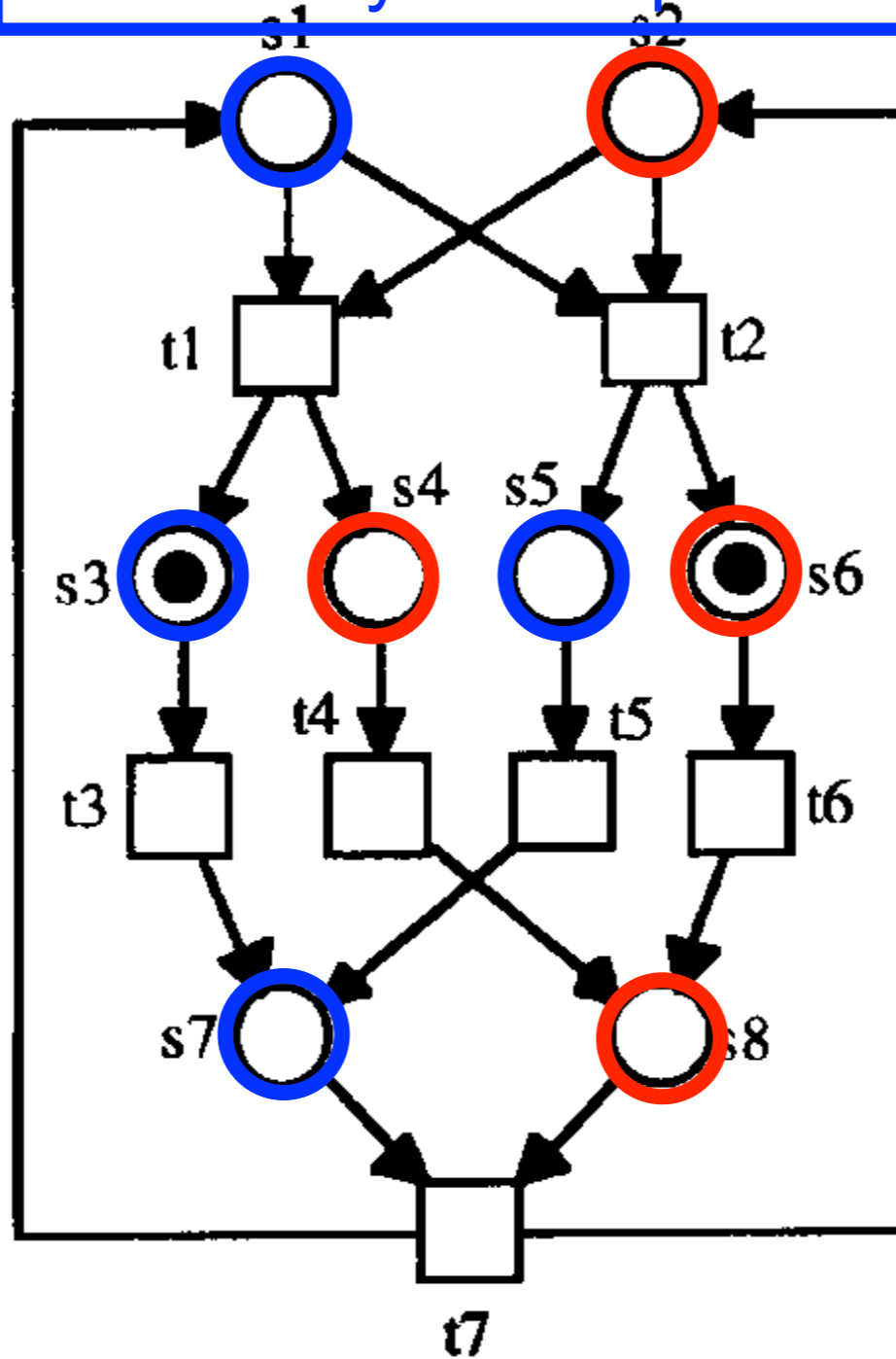
S-component

# S-cover: example

covered by S-components



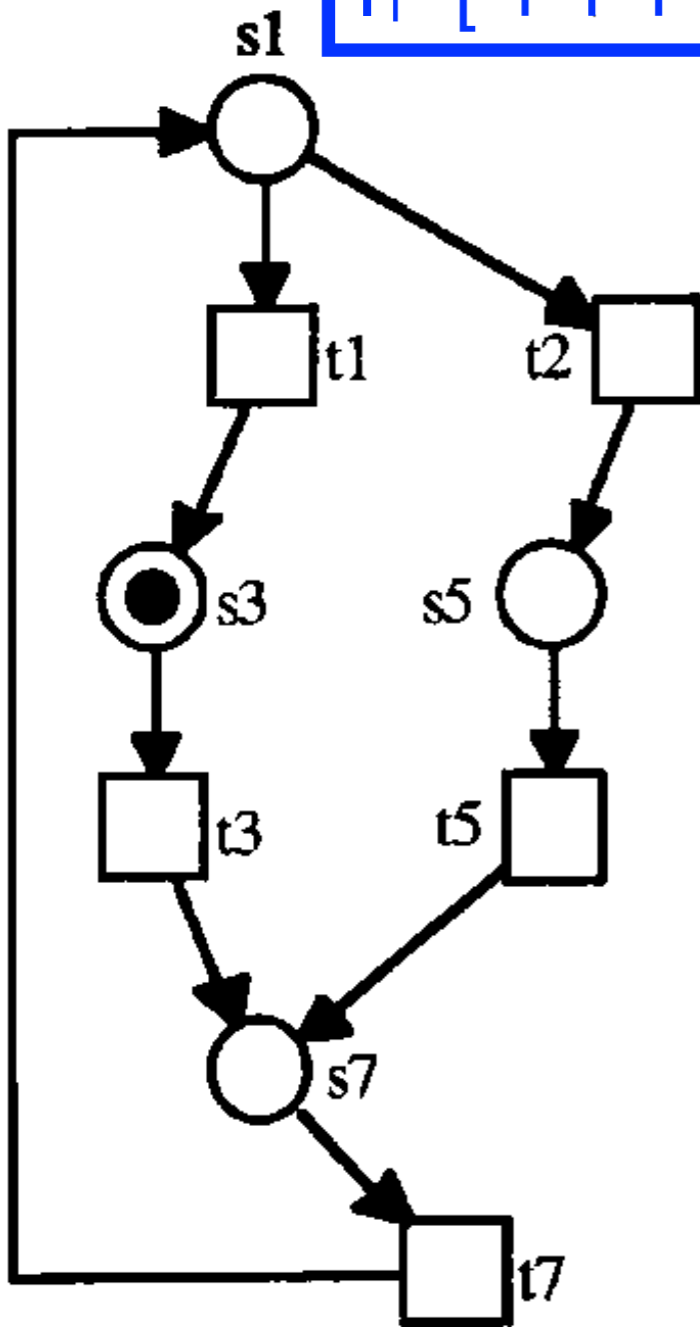
S-component



S-component

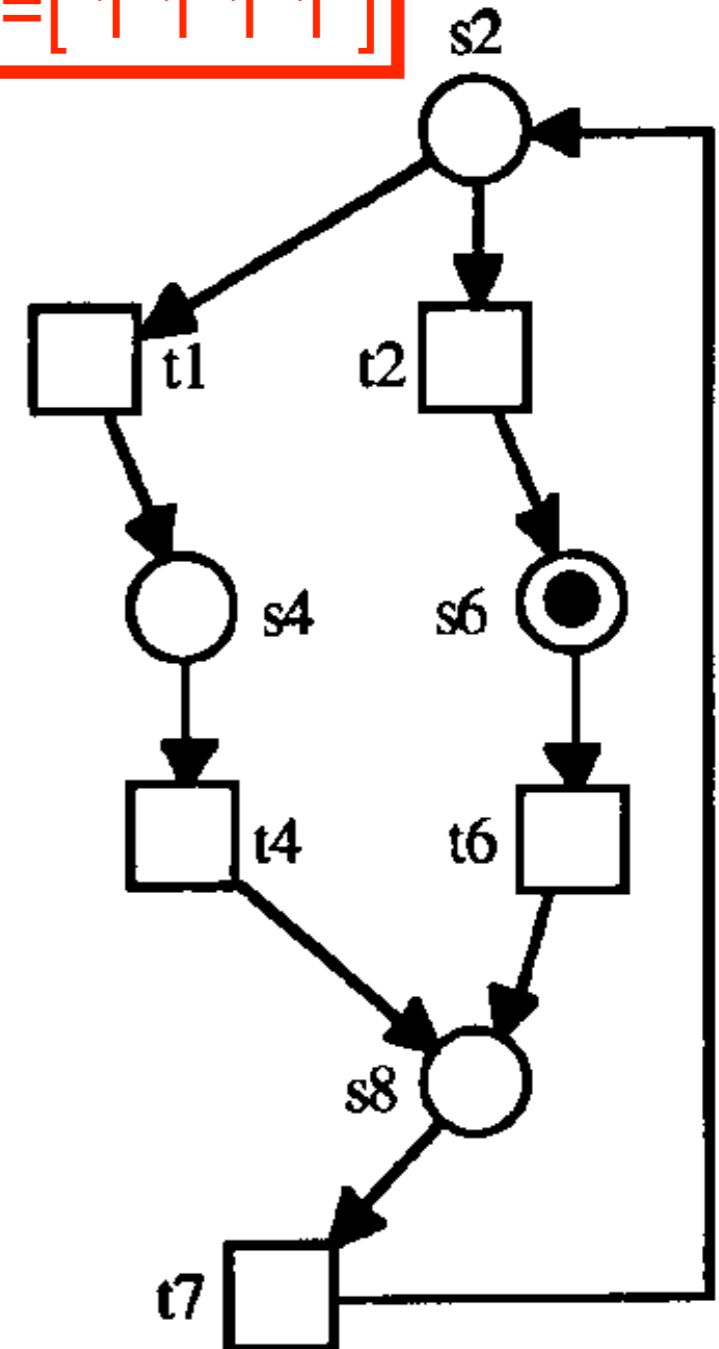
# S-cover: example

$$l_1 = [1 \ 1 \ 1 \ 1]$$



S-component

$$l_2 = [1 \ 1 \ 1 \ 1]$$

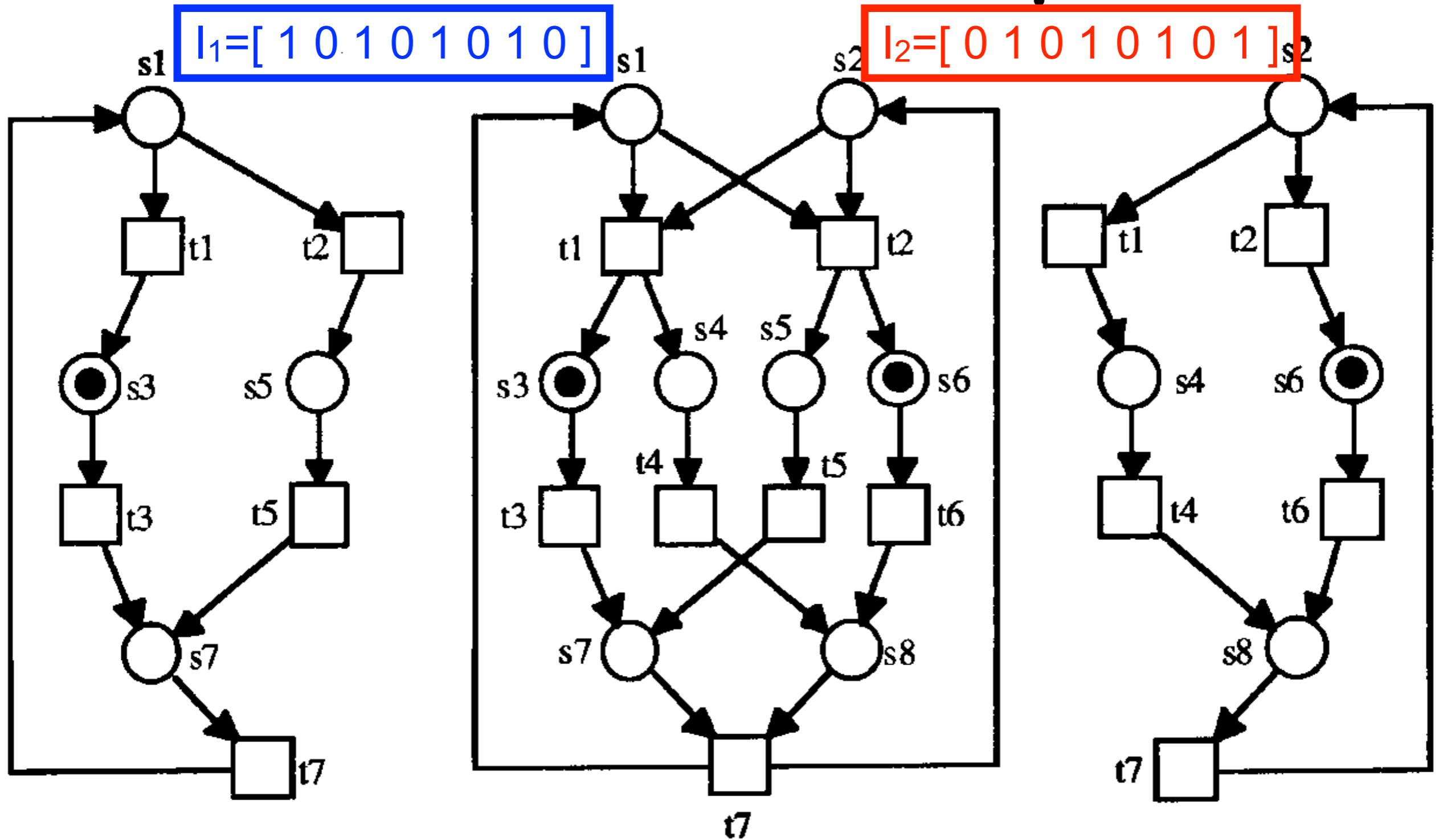


S-component

S-cover



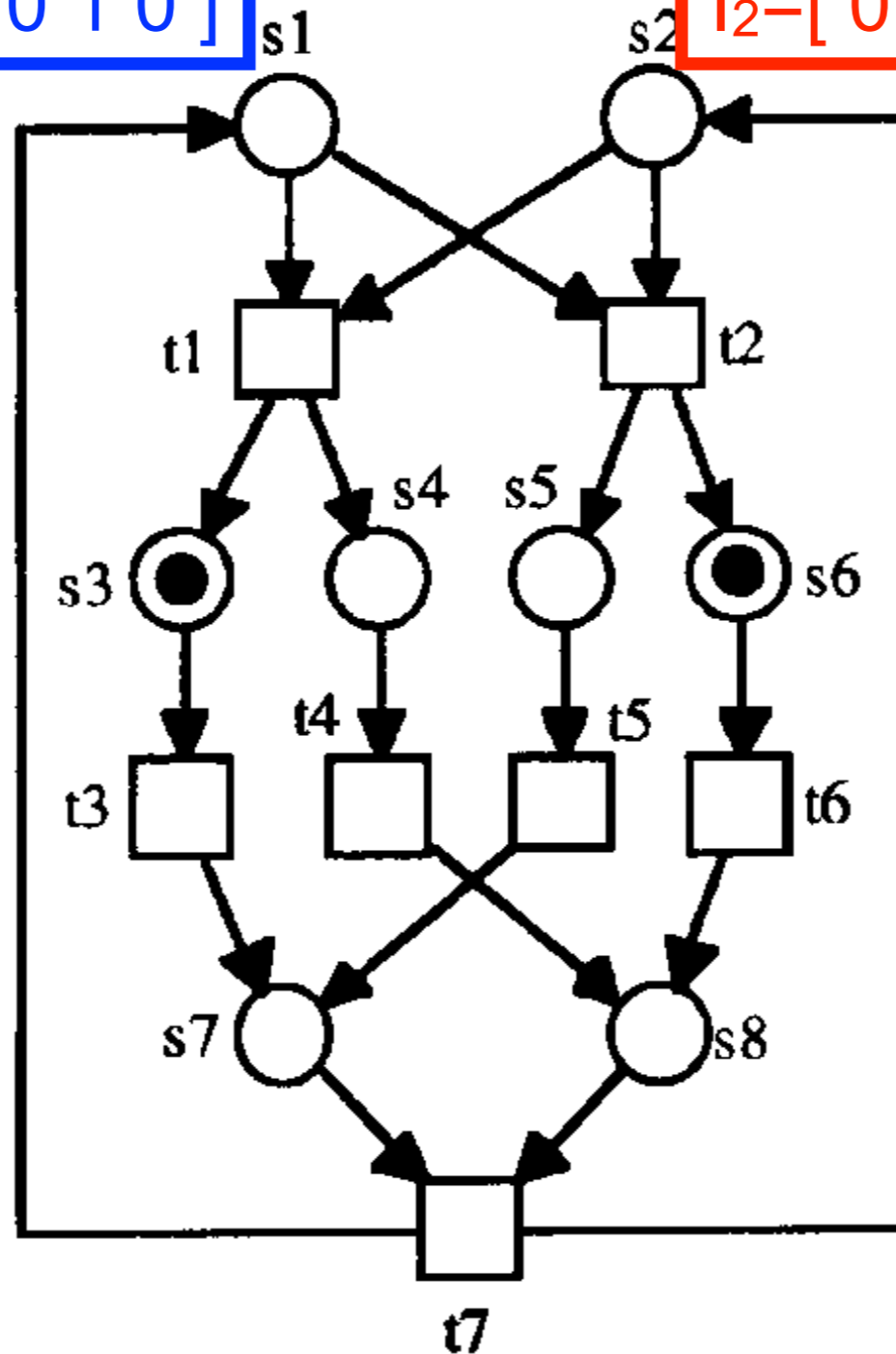
# S-cover: example



# S-cover: example

$$I_1 = [1\ 0\ 1\ 0\ 1\ 0\ 1\ 0]$$

$$I_2 = [0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]$$



$$I_1 + I_2 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$$

positive S-invariant

# S-coverability theorem

**Theorem:** If a free-choice system is live and bounded then it is S-coverable

(proof omitted)

Consequence:

**free-choice + not S-coverable  $\Rightarrow$  not (live and bounded)**

# S-Coverability diagnosis

N is sound iff  $N^*$  is live and bounded (Main Theorem)

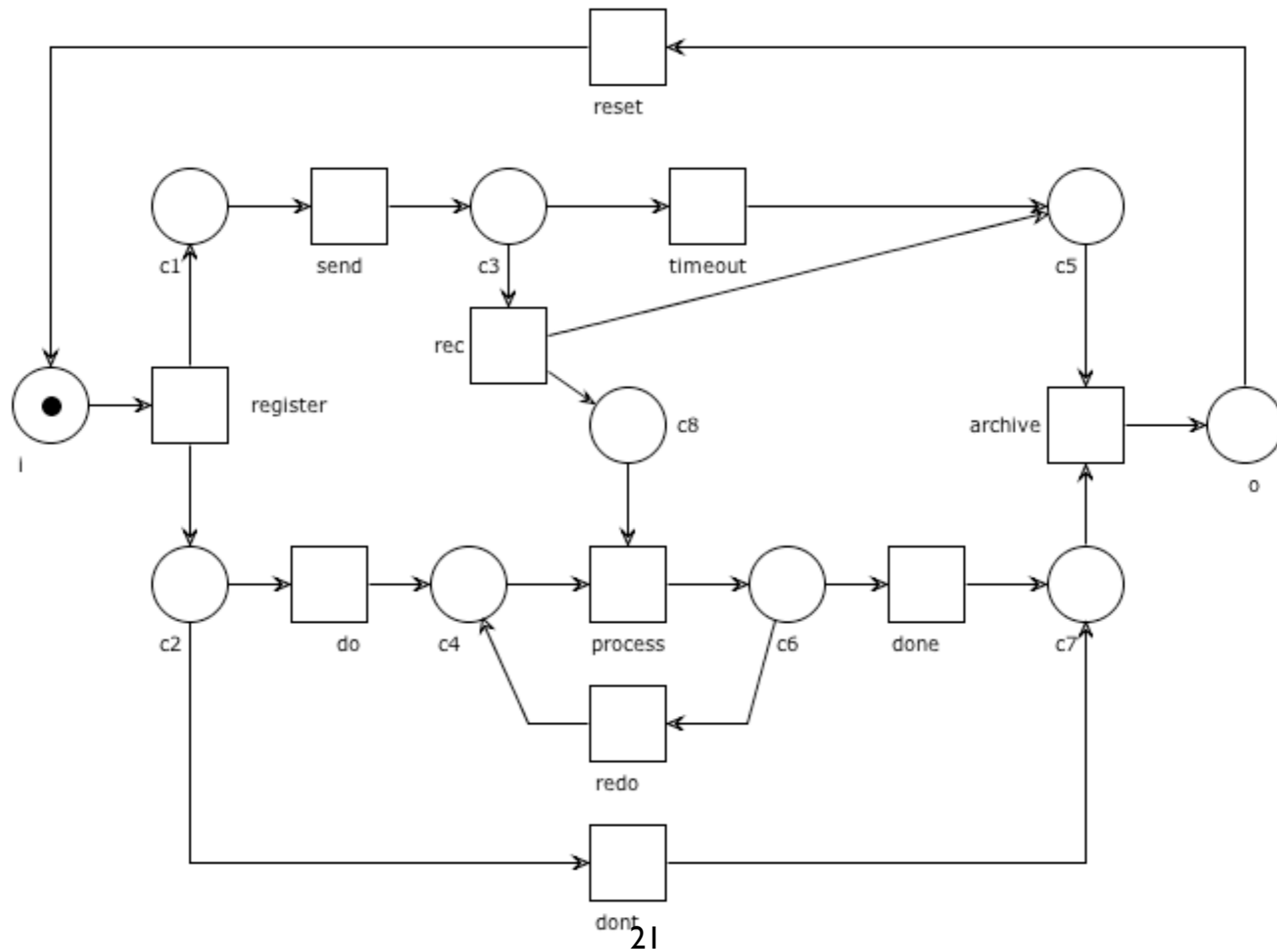
N is free-choice iff  $N^*$  is free-choice

If  $N^*$  is free-choice, live and bounded  
it must be S-coverable (S-coverability theorem)

**Corollary:** If N is sound and free-choice,  
then  $N^*$  must be S-coverable

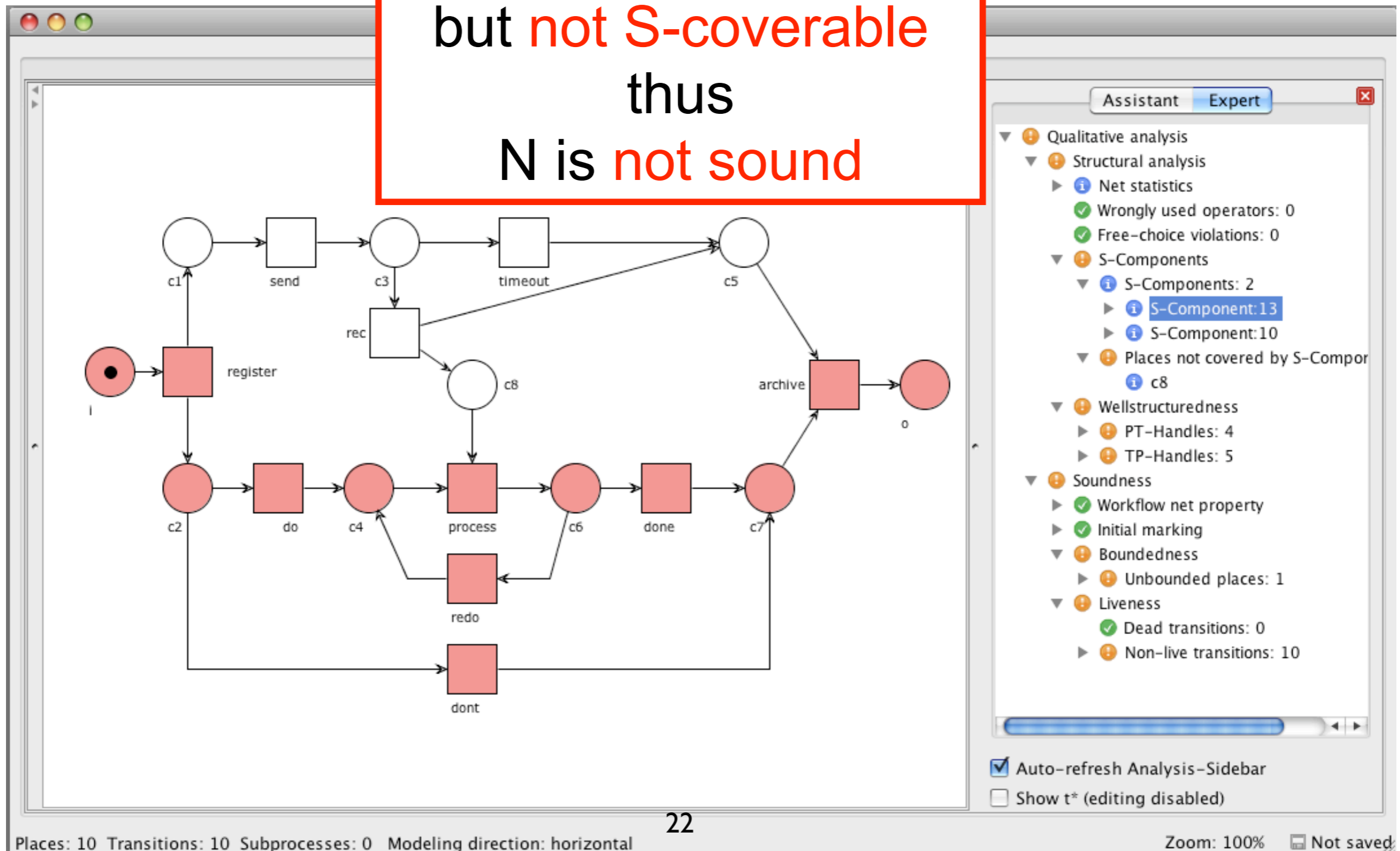
**N free-choice +  $N^*$  not S-coverable  $\Rightarrow$  N not sound**

# S-cover for $N^*$ ?



# WoPeD Diagnosis

$N^*$  is free-choice  
but not S-coverable  
thus  
 $N$  is not sound



# Be careful

reset transition is implicit in WoPeD

WoPeD shows S-components for  $N^*$   
(not for  $N$ )

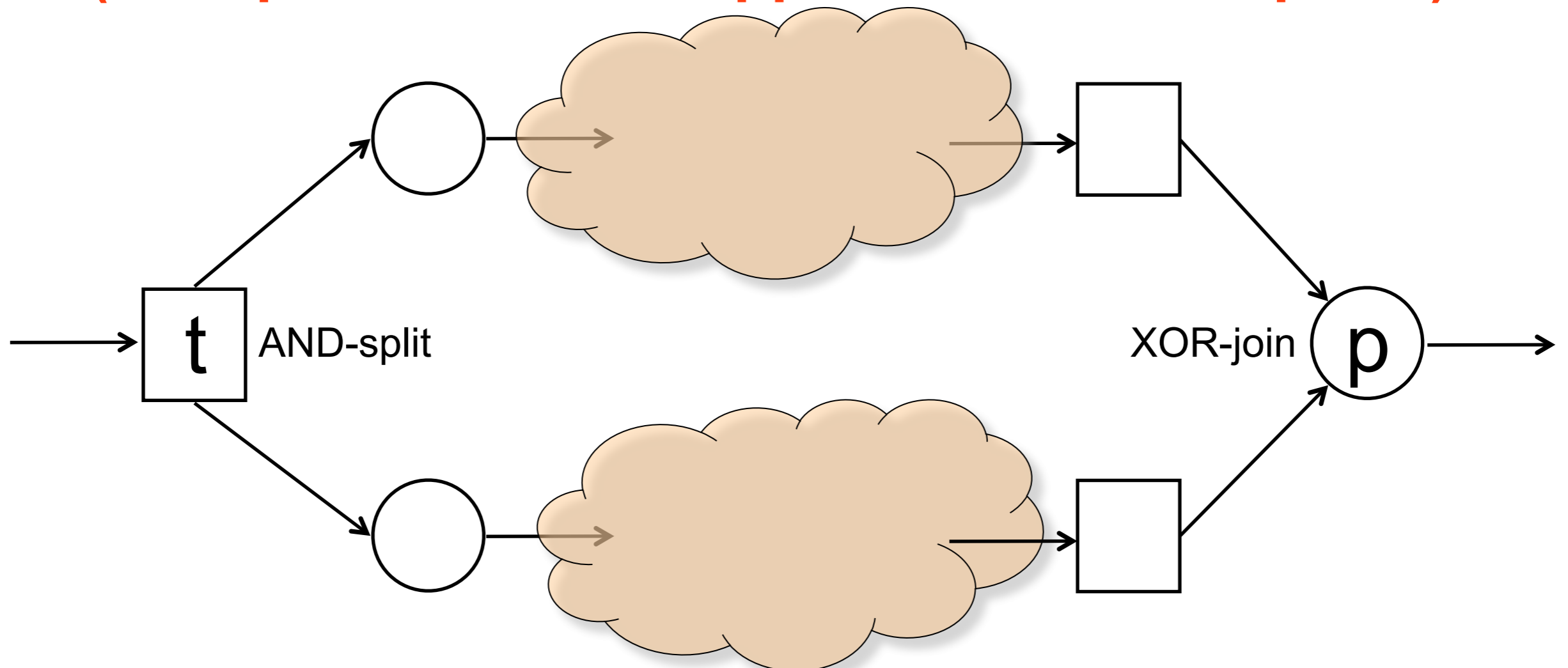
# Well-structuredness (PT/TP-handles)



# TP-handles

Two parallel flows initiated by an AND-split should not be joined by a XOR-join

**(multiple tokens can appear in the same place)**



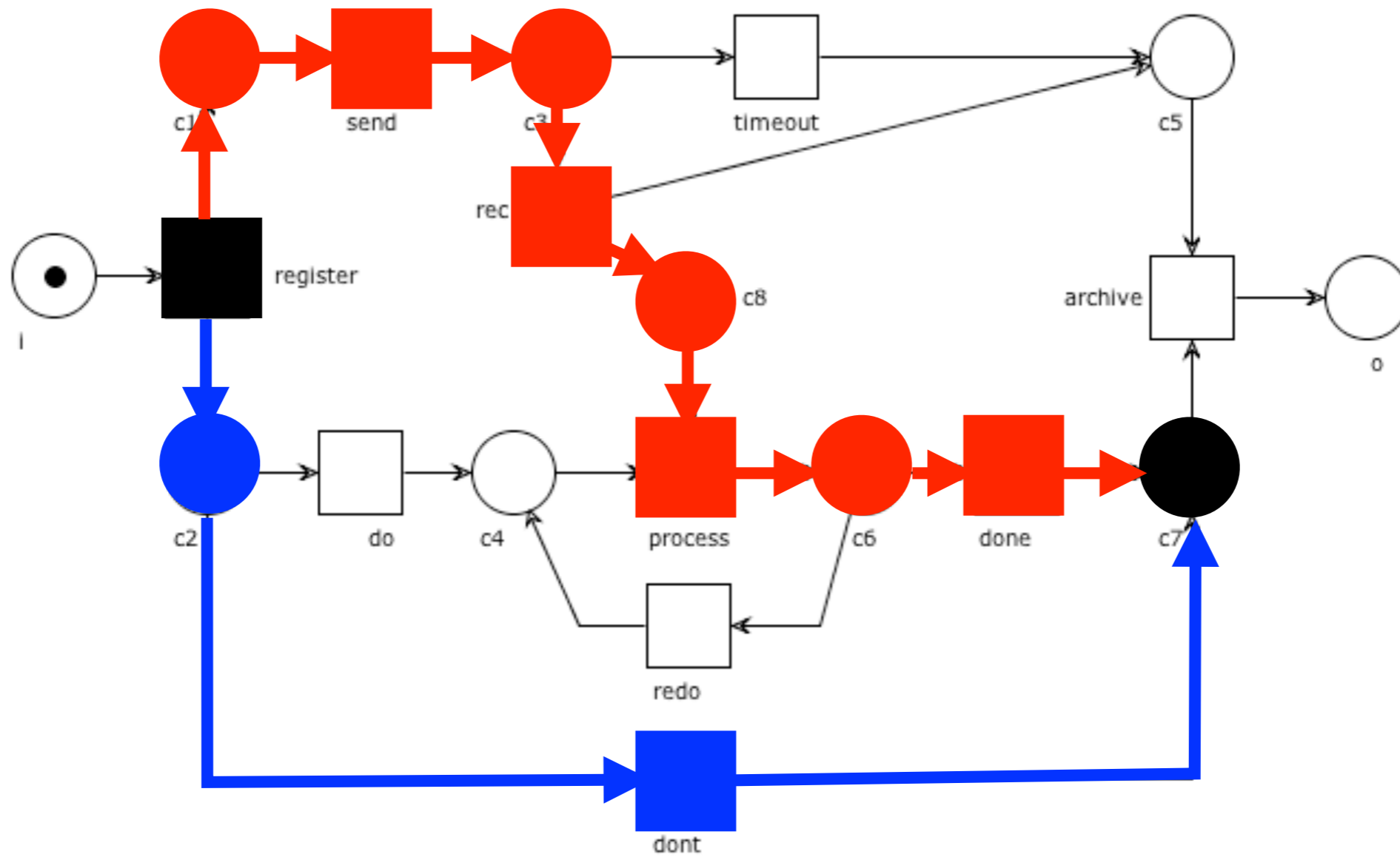
# TP-handles

## Definition:

A transition  $t$  and a place  $p$  form a **TP-handle** if there are

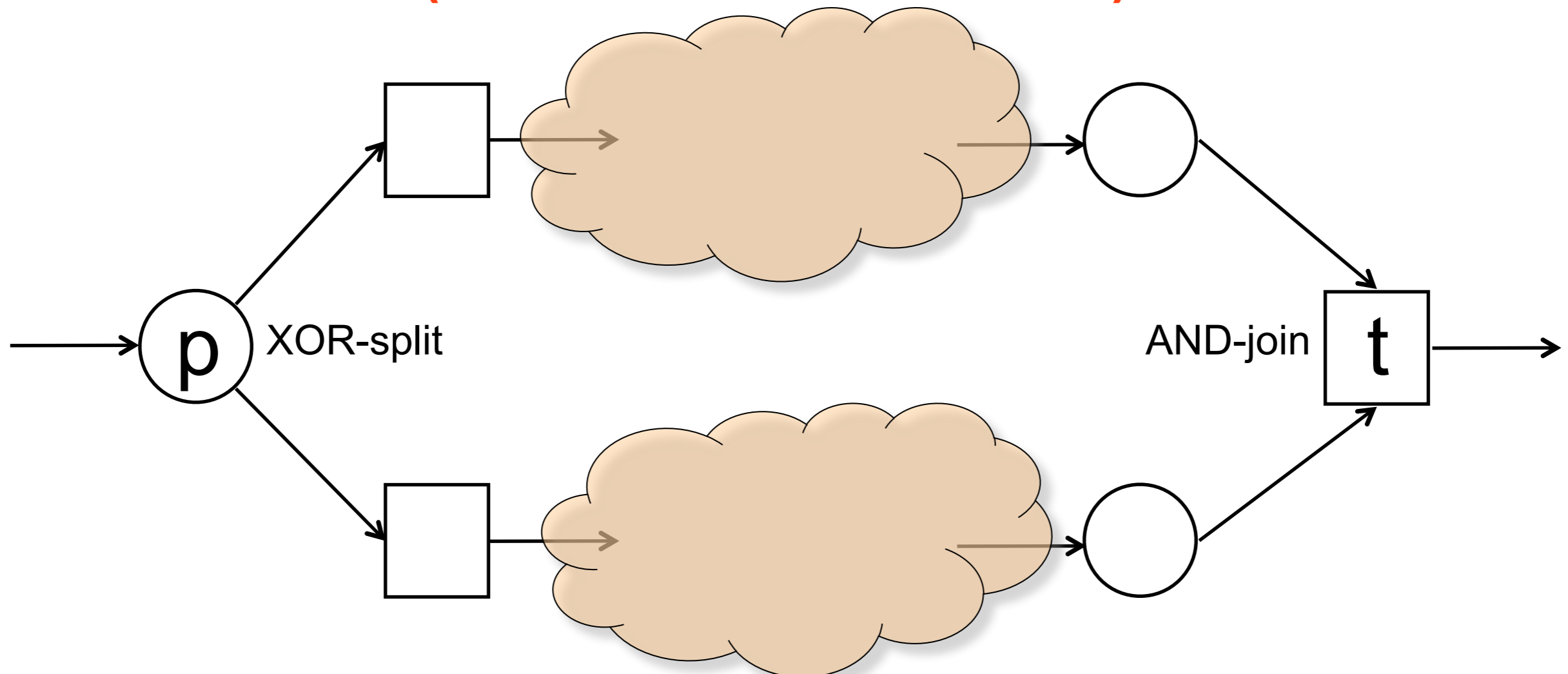
two distinct elementary paths  $c_1$  and  $c_2$  from  $t$  to  $p$  such that the only nodes they have in common are  $t, p$

# Example: TP-handle



# PT-handles

Two alternative flows created via a XOR-split  
should not be synchronized by an AND-join  
**(the net could deadlock)**



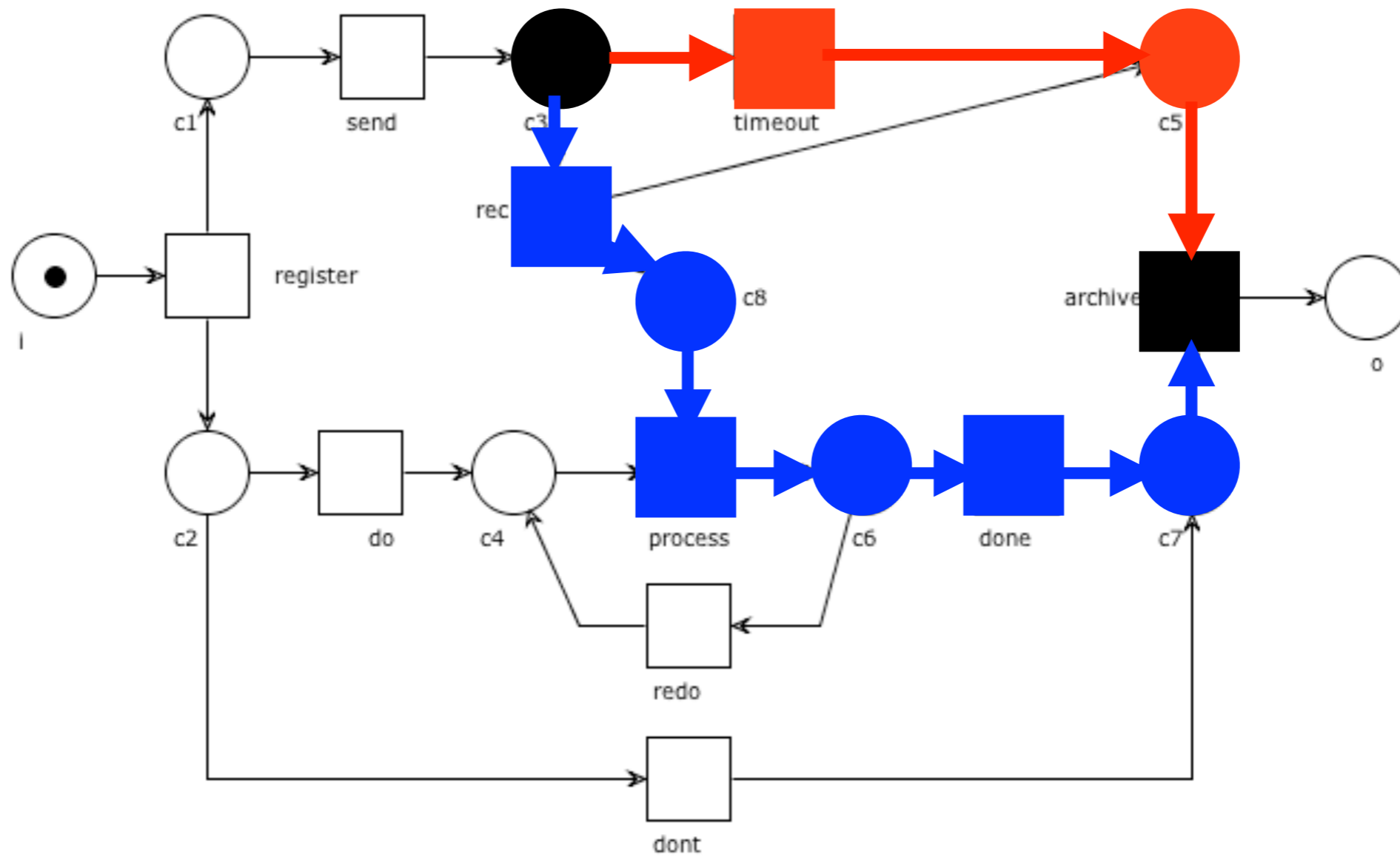
# PT-handles

## Definition:

A place  $p$  and a transition  $t$  form a **PT-handle**  
if there are

two distinct elementary paths  $c_1$  and  $c_2$  from  $p$  to  $t$   
such that the only nodes they have in common are  $p, t$

# Example: PT-handle



# Well-Structured Nets

**Definition:** A net is **well-handled** if it has neither TP-handles nor PT-handles

**Definition:** A workflow net  $N$  is **well-structured** if  $N^*$  is well-handled

# Be careful

$N$  well-structured =  $N^*$  well-handled

reset transition is implicit in WoPeD

WoPeD marks PT/TP-handles over  $N^*$   
(not over  $N$ )



# WoPeD Diagnosis

The screenshot displays the WoPeD tool interface for analyzing a Petri net. The main window shows a Petri net with places  $c1$  through  $c8$  and transitions labeled `register`, `send`, `rec`, `timeout`, `do`, `process`, `redo`, `done`, `dont`, and `archive`. Places  $c3$  and `process` are highlighted in red. The initial state is marked with  $i$  and the final state with  $o$ .

The **Semantical analysis** panel on the right is set to **Expert** mode and shows the following analysis results:

- Qualitative analysis
  - Structural analysis
    - Net statistics
      - Wrongly used operators: 0
      - Free-choice violations: 0
    - S-Components
    - Wellstructuredness
      - PT-Handles: 4
        - PT-Handle pair #1
        - PT-Handle pair #2
        - PT-Handle pair #3
        - PT-Handle pair #4**
          - $c3$
          - process
    - TP-Handles: 5
      - TP-Handle pair #1
        - register
        - $c7$
      - TP-Handle pair #2
      - TP-Handle pair #3
      - TP-Handle pair #4
      - TP-Handle pair #5
- Soundness

# WoPeD Diagnosis

The image displays the WoPeD Diagnosis interface for a workflow net (wfnet-unsound.pnml). The main window shows a Petri net diagram with places (circles) and transitions (squares). Places are labeled c1 through c8, and transitions are labeled register, send, timeout, archive, do, done, redo, and dont. A blue arrow points from place c3 to transition rec, and another blue arrow points from transition rec to place c8. The Semantical analysis panel on the right shows a tree structure of analysis results:

- Qualitative analysis
  - Structural analysis
    - Net statistics
      - Wrongly used operators: 0
      - Free-choice violations: 0
    - S-Components
    - Wellstructuredness
      - PT-Handles: 4
        - PT-Handle pair #1
        - PT-Handle pair #2
        - PT-Handle pair #3
        - PT-Handle pair #4**
          - c3
          - process
    - TP-Handles: 5
      - TP-Handle pair #1
        - register
        - c7
      - TP-Handle pair #2
      - TP-Handle pair #3
      - TP-Handle pair #4
      - TP-Handle pair #5
  - Soundness

The interface also includes a status bar at the bottom with "Horizontal", "Zoom: 100%", and "Not saved" indicators.

# WoPeD Diagnosis

The image displays the WoPeD (Web of Petri Nets) diagnosis tool interface. The main window shows a Petri net diagram for a BPEL process named 'wfnet-unsound.pnml'. The diagram includes places (circles) labeled c1 through c8 and transitions (squares) labeled 'register', 'send', 'rec', 'timeout', 'archive', 'do', 'process', 'redo', 'dont', and 'reset'. A red path highlights a sequence of transitions and places: 'register' (c1) → 'send' (c3) → 'timeout' (c5) → 'archive' (o) → 'reset' (square) → 'register' (c1). A blue path highlights: 'rec' (square) → c8 (circle) → 'process' (square) → c4 (circle) → 'redo' (square) → c2 (circle) → 'register' (square) → c1 (circle).

The 'Semantical analysis' panel on the right provides a hierarchical view of the analysis results:

- Qualitative analysis
  - Structural analysis
    - Net statistics
      - Wrongly used operators: 0
      - Free-choice violations: 0
    - S-Components
    - Wellstructuredness
      - PT-Handles: 4
        - PT-Handle pair #1
        - PT-Handle pair #2
        - PT-Handle pair #3
        - PT-Handle pair #4**
          - c3
          - process
      - TP-Handles: 5
        - TP-Handle pair #1
          - register
          - c7
        - TP-Handle pair #2
        - TP-Handle pair #3
        - TP-Handle pair #4
        - TP-Handle pair #5
- Soundness

At the bottom of the interface, there is a status bar with 'Horizontal', 'Zoom: 100%', a zoom slider, and 'Not saved'.

# Well-structuredness, S-coverability and Soundness

**Theorem:** If  $N$  is sound and well-structured,  
then  $N^*$  is S-coverable  
(proof omitted)

Consequence:

**$N$  well-structured +  $N^*$  not S-coverable  $\Rightarrow$   $N$  not sound**

# Error sequences

# Woflan

<http://www.win.tue.nl/woflan/>

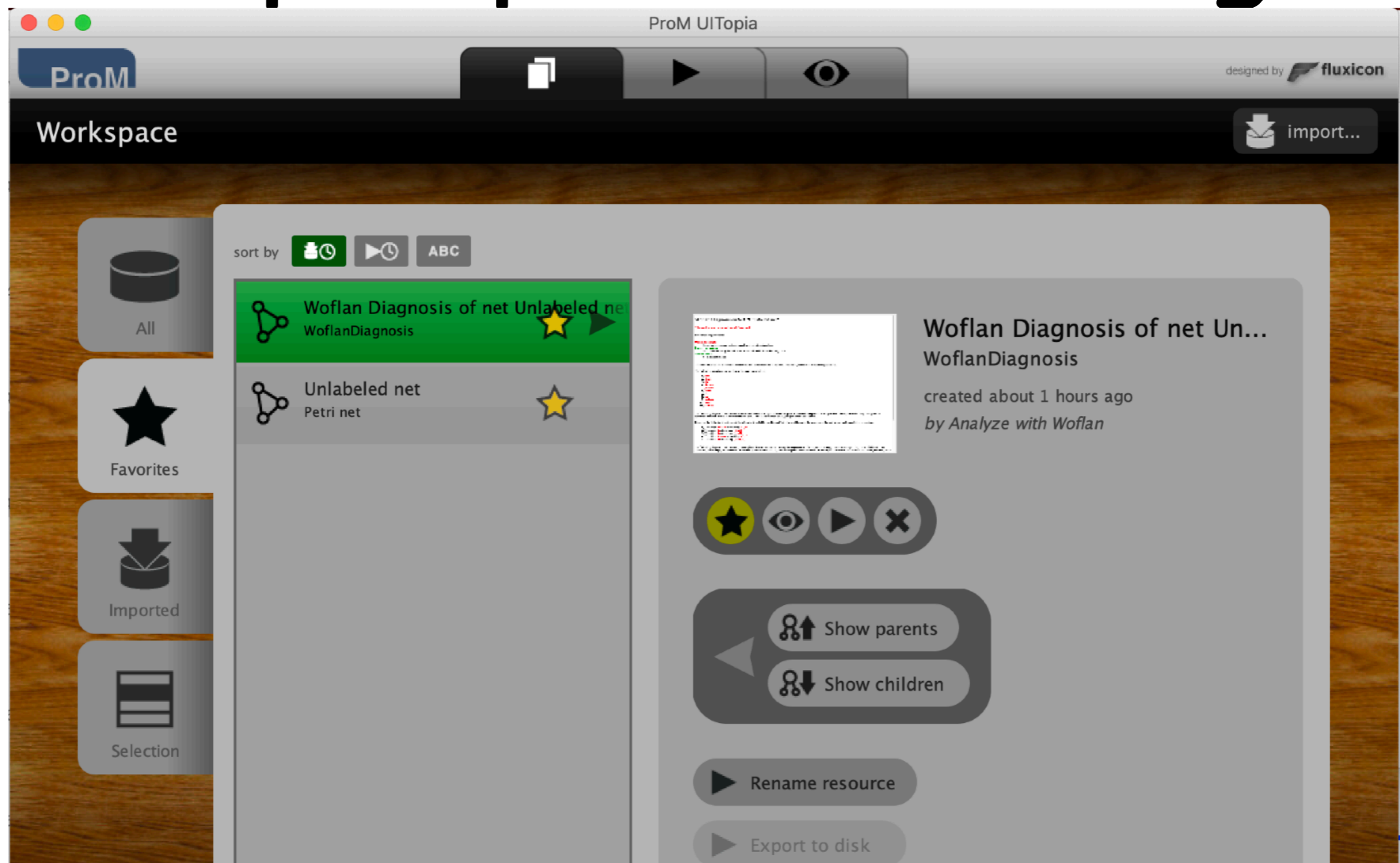
WOrkFLow ANalyzer  
(Microsoft Windows only)



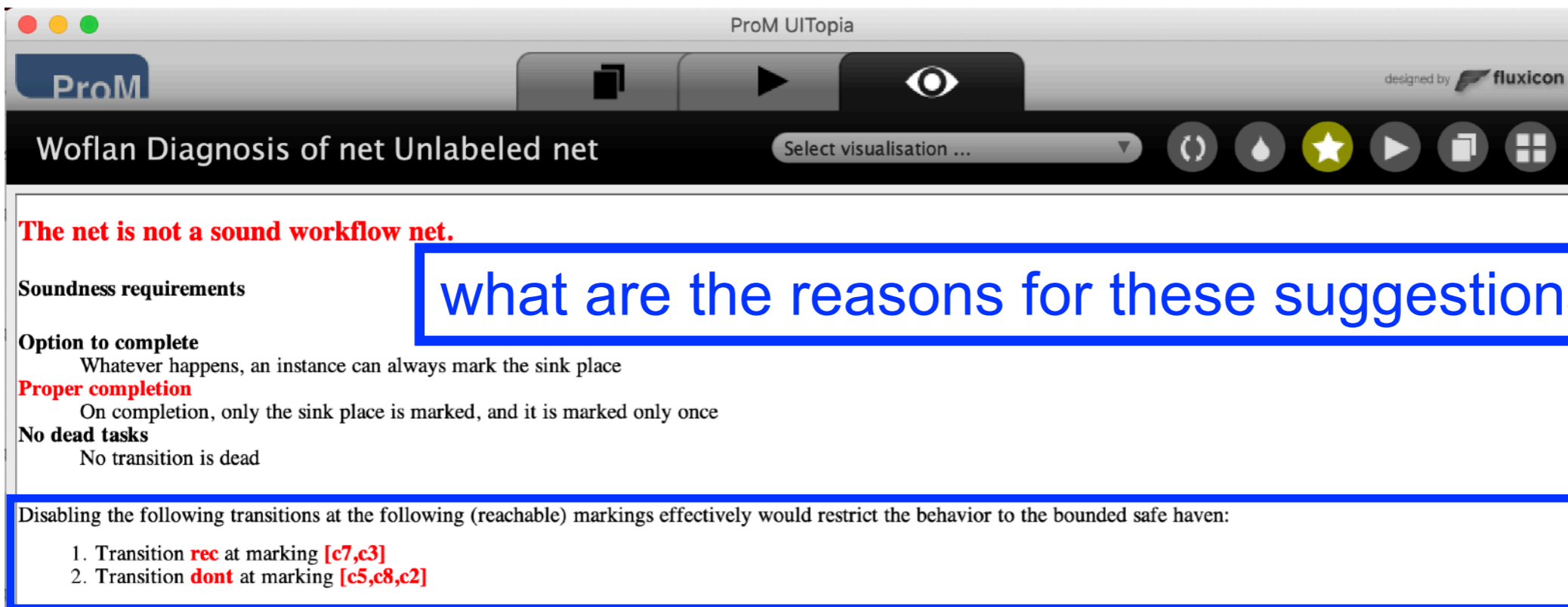
Woflan tells us if  $N$  is a sound workflow net  
(Is  $N$  a workflow net? Is  $N^*$  bounded? Is  $N^*$  live?)  
if not, provides some diagnostic information

# Woflan now a ProM plugin

<http://promtools.org/>



# Woflan (in ProM)



ProM UI Topia

ProM

Woflan Diagnosis of net Unlabeled net

Select visualisation ...

**The net is not a sound workflow net.**

**Soundness requirements**

**Option to complete**  
Whatever happens, an instance can always mark the sink place

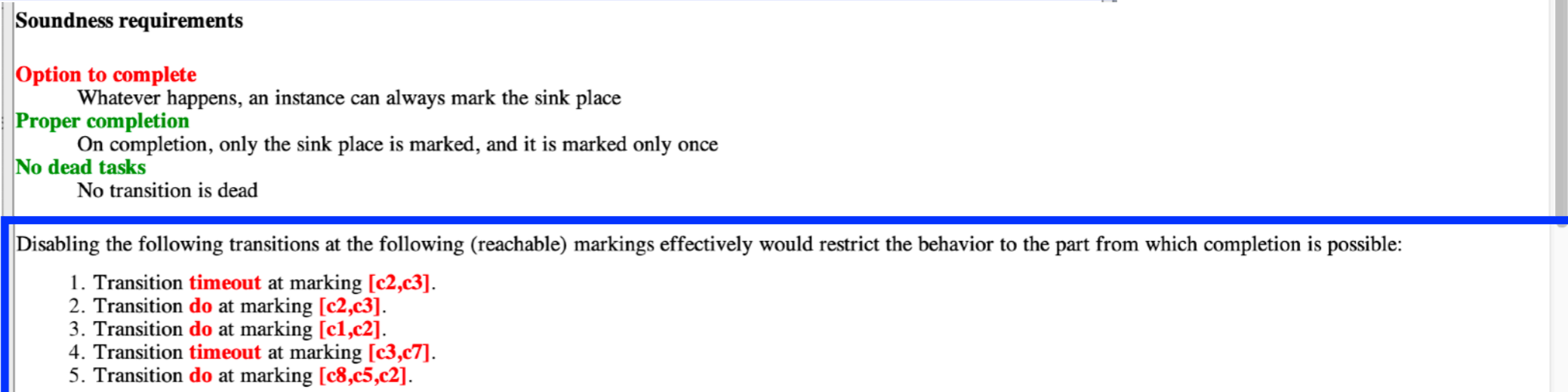
**Proper completion**  
On completion, only the sink place is marked, and it is marked only once

**No dead tasks**  
No transition is dead

Disabling the following transitions at the following (reachable) markings effectively would restrict the behavior to the bounded safe haven:

1. Transition **rec** at marking **[c7,c3]**
2. Transition **dont** at marking **[c5,c8,c2]**

what are the reasons for these suggestions?



Soundness requirements

**Option to complete**  
Whatever happens, an instance can always mark the sink place

**Proper completion**  
On completion, only the sink place is marked, and it is marked only once

**No dead tasks**  
No transition is dead

Disabling the following transitions at the following (reachable) markings effectively would restrict the behavior to the part from which completion is possible:

1. Transition **timeout** at marking **[c2,c3]**.
2. Transition **do** at marking **[c2,c3]**.
3. Transition **do** at marking **[c1,c2]**.
4. Transition **timeout** at marking **[c3,c7]**.
5. Transition **do** at marking **[c8,c5,c2]**.



# Diagnostic information

The sets of:  
unbounded places of  $N^*$   
dead transitions of  $N^*$   
non-live transitions of  $N^*$

may provide useful information for  
the diagnosis of behavioural errors

Unfortunately, this information is not always sufficient  
to determine the exact cause of the error

**Behavioural error sequences** help us to locate problems

# Error sequences

Rationale:

We want to find firing sequences such that:

1. every continuation of such sequences will lead to an error
2. they are as short as possible  
(none of their prefixes satisfies the above property)

Informally:

error sequences are scenarios that capture the essence of errors made in the workflow design (violate “option to complete” or “proper completion”)

Error sequences:  
Non-live sequences

# Non-Live sequences: informally

A **non-live sequence** is a firing sequence as short as possible such that **completion of the case is no longer possible**

i.e. a witness for transition reset being non-live in  $N^*$

# Non-Live sequences: fundamental property

Let  $N$  be such that:  
 $N^*$  is bounded  
 $N$  (or equivalently  $N^*$ ) has no dead task

Then,  $N^*$  is live  
**iff**  
 $N$  has no non-live sequences

# Non-Live sequences: graphically

**The analysis is possible in bounded systems only**

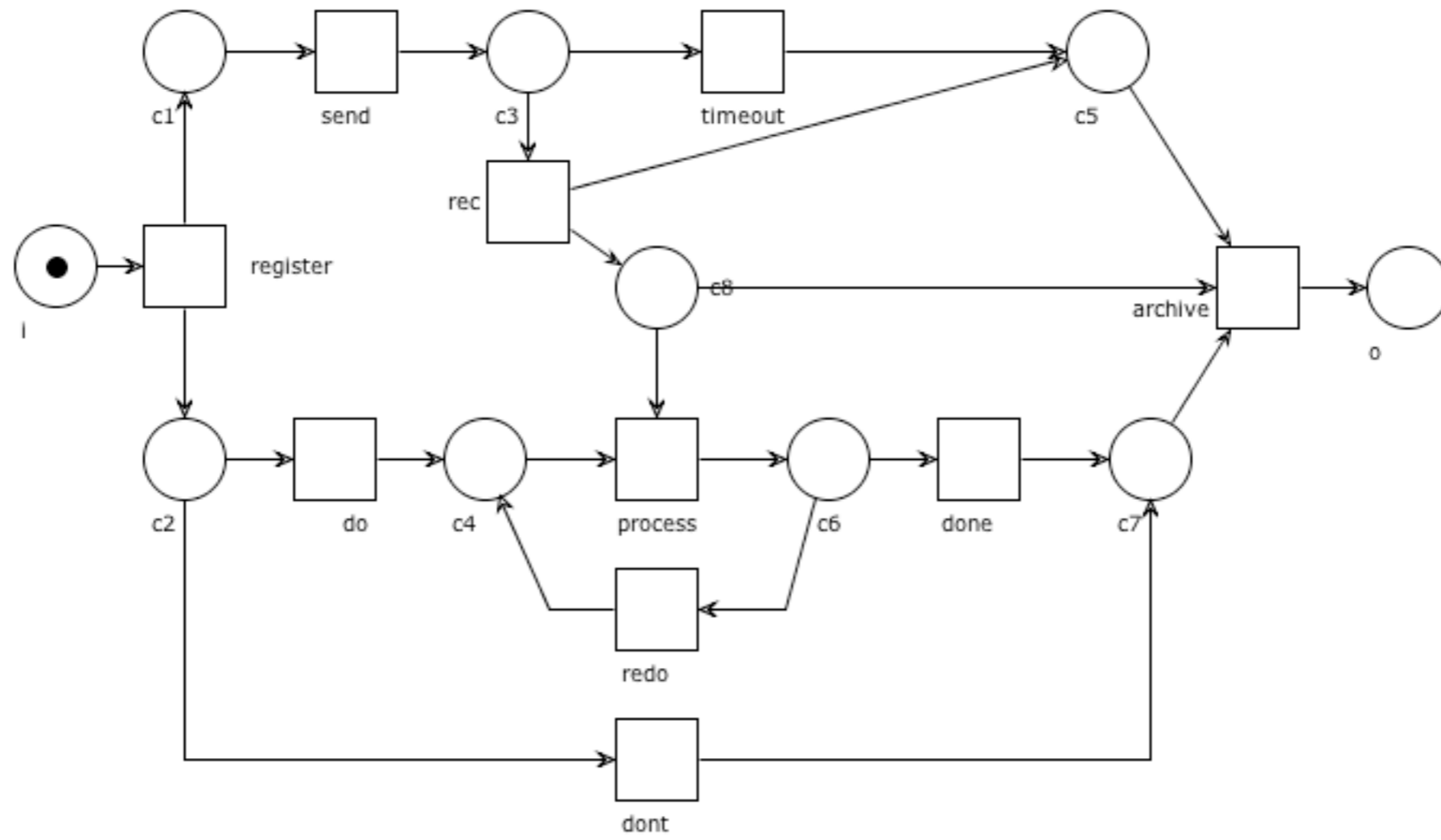
Compute the RG of  $N^*$

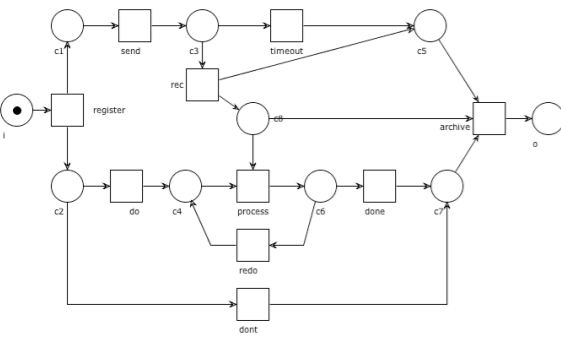
Color in **red** all nodes from which there is **no path** to  $o$

Color in **green** all nodes from which **all paths** lead to  $o$

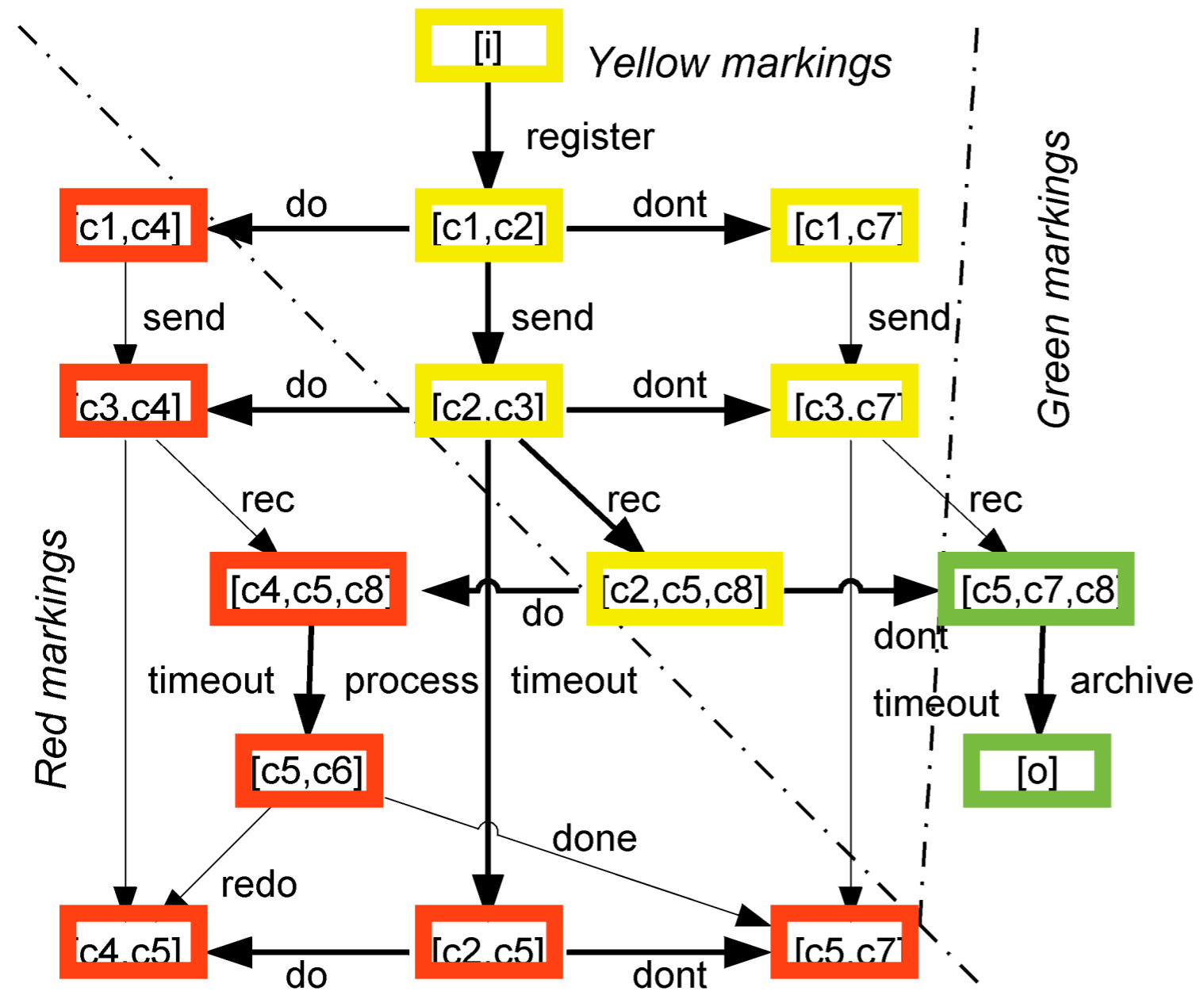
Color in **yellow** all remaining nodes  
(some but not all paths lead to  $o$ )

# Example: N





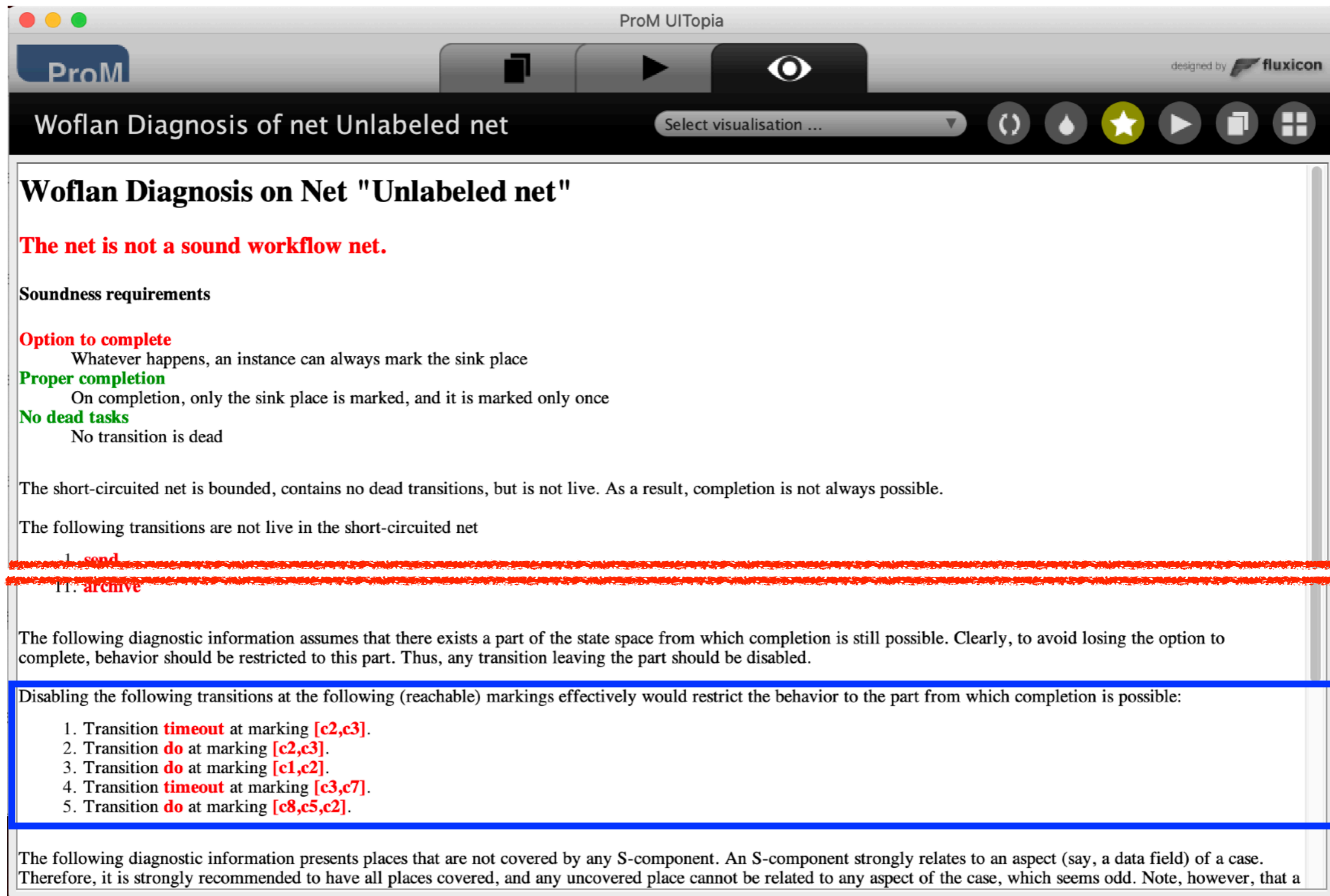
# Example: RG (N)



- Non-live sequences:
- register, **do**
  - register, send, **do**
  - register, send, **timeout**
  - register, send, rec, **do**
  - register, send, dont, **timeout**
  - register, dont, send, **timeout**



# Woflan (in ProM)



The screenshot shows the ProM UI with the title "Woflan Diagnosis of net Unlabeled net". The main content area displays the following text:

**Woflan Diagnosis on Net "Unlabeled net"**

**The net is not a sound workflow net.**

**Soundness requirements**

- Option to complete**  
Whatever happens, an instance can always mark the sink place
- Proper completion**  
On completion, only the sink place is marked, and it is marked only once
- No dead tasks**  
No transition is dead

The short-circuited net is bounded, contains no dead transitions, but is not live. As a result, completion is not always possible.

The following transitions are not live in the short-circuited net

- 1. ~~send~~
- 11. ~~archive~~

The following diagnostic information assumes that there exists a part of the state space from which completion is still possible. Clearly, to avoid losing the option to complete, behavior should be restricted to this part. Thus, any transition leaving the part should be disabled.

Disabling the following transitions at the following (reachable) markings effectively would restrict the behavior to the part from which completion is possible:

- 1. Transition **timeout** at marking [c2,c3].
- 2. Transition **do** at marking [c2,c3].
- 3. Transition **do** at marking [c1,c2].
- 4. Transition **timeout** at marking [c3,c7].
- 5. Transition **do** at marking [c8,c5,c2].

The following diagnostic information presents places that are not covered by any S-component. An S-component strongly relates to an aspect (say, a data field) of a case. Therefore, it is strongly recommended to have all places covered, and any uncovered place cannot be related to any aspect of the case, which seems odd. Note, however, that a

Error sequences:  
Unbounded sequences

# Unbounded sequences: informally

An **unbounded sequence** is a firing sequence of **minimal length** such that every continuation **invalidates proper completion**

i.e. a witness for unboundedness

# Unbounded sequences: fundamental property

$N^*$  is bounded  
**iff**

$N$  has no unbounded sequences

Undesired markings:  
**infinite-weighted markings or markings greater than 0**

# Unbounded sequences: graphically

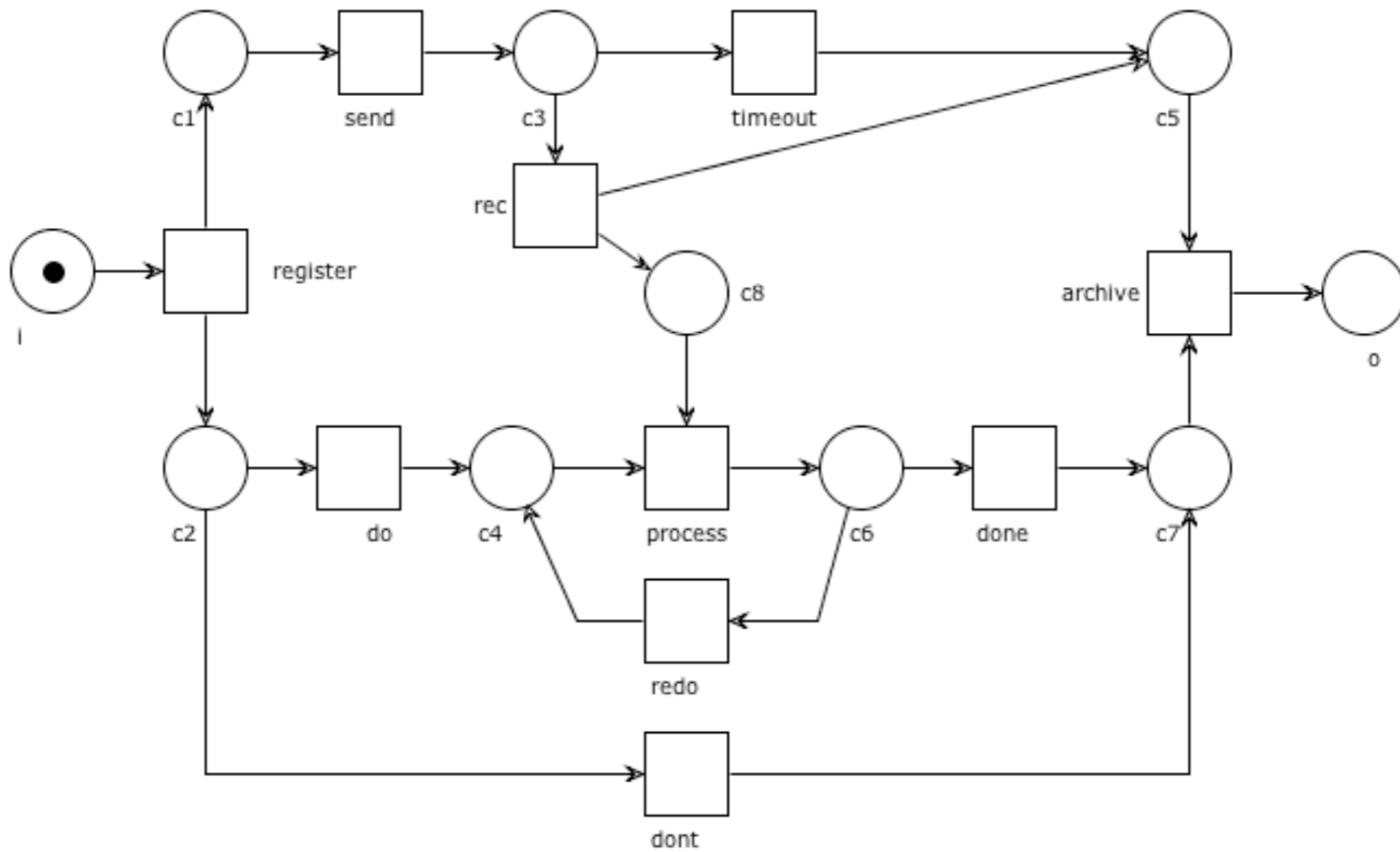
Compute the CG of  $N^*$

Color in **green** all nodes from which  
**undesired markings are not reachable**

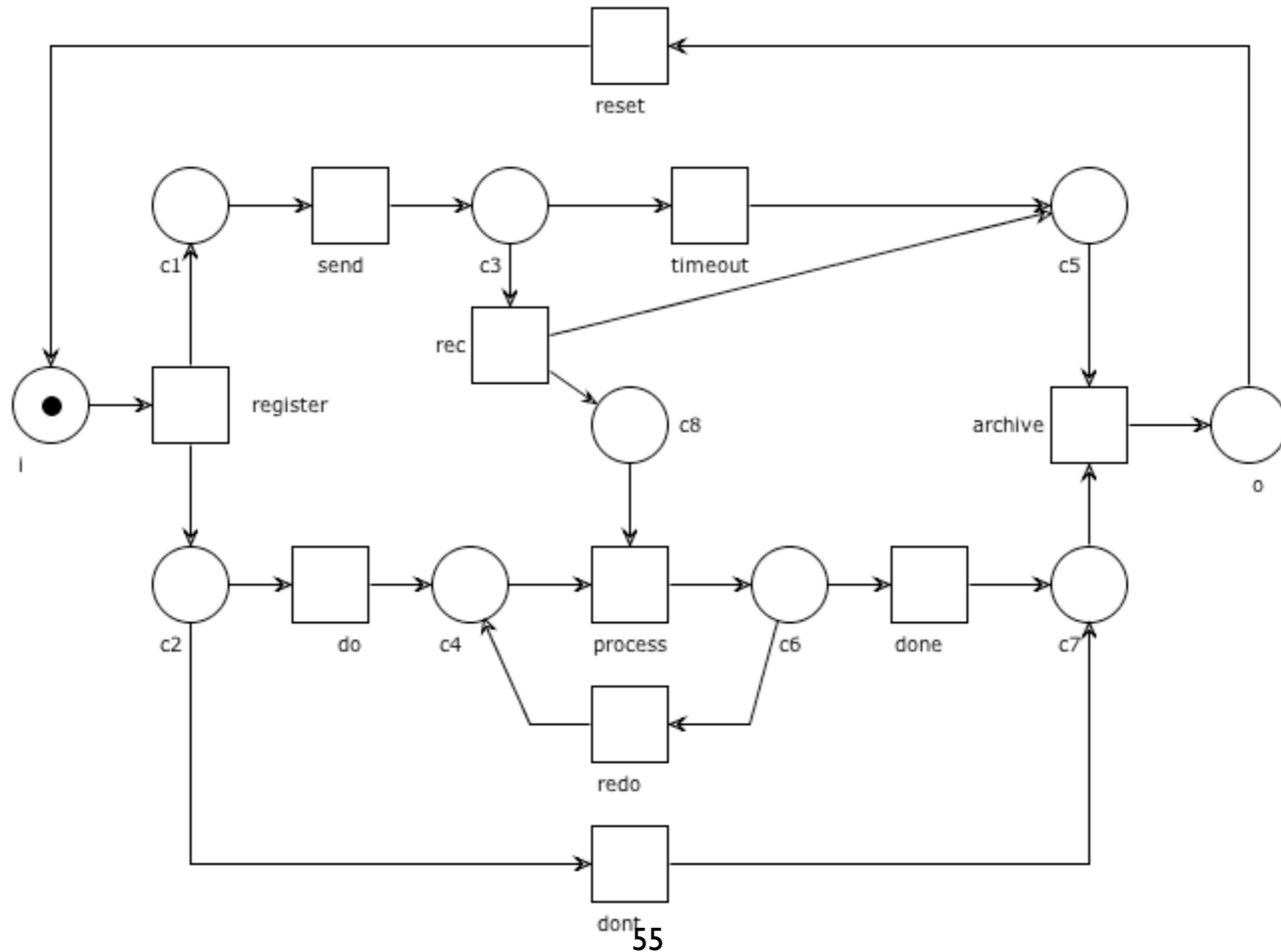
Color in **red** all nodes from which  
**no green marking is reachable**  
(undesired markings are unavoidable)

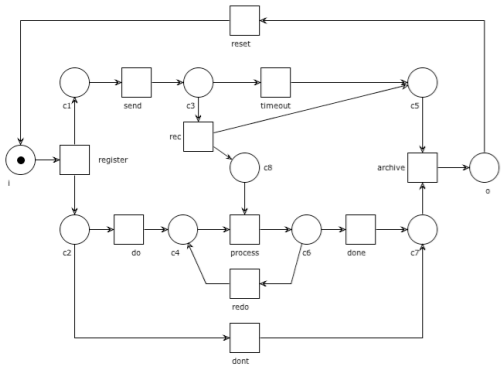
Color in **yellow** all remaining nodes  
(undesired markings are reachable but avoidable)

# Example: N

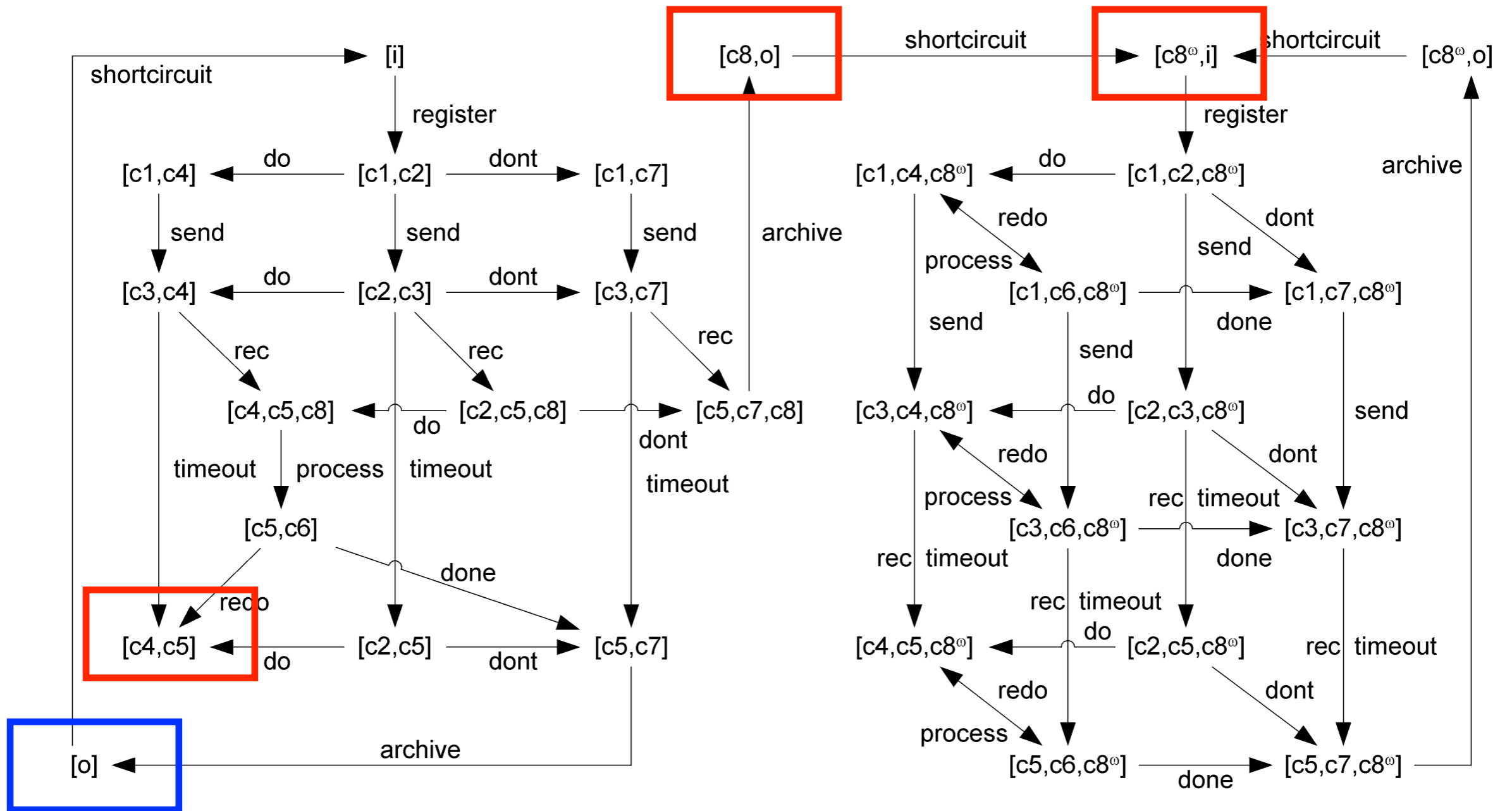


# Example: $N^*$





# Example: CG ( $N^*$ )





# Restricted coverability graph (RCG)

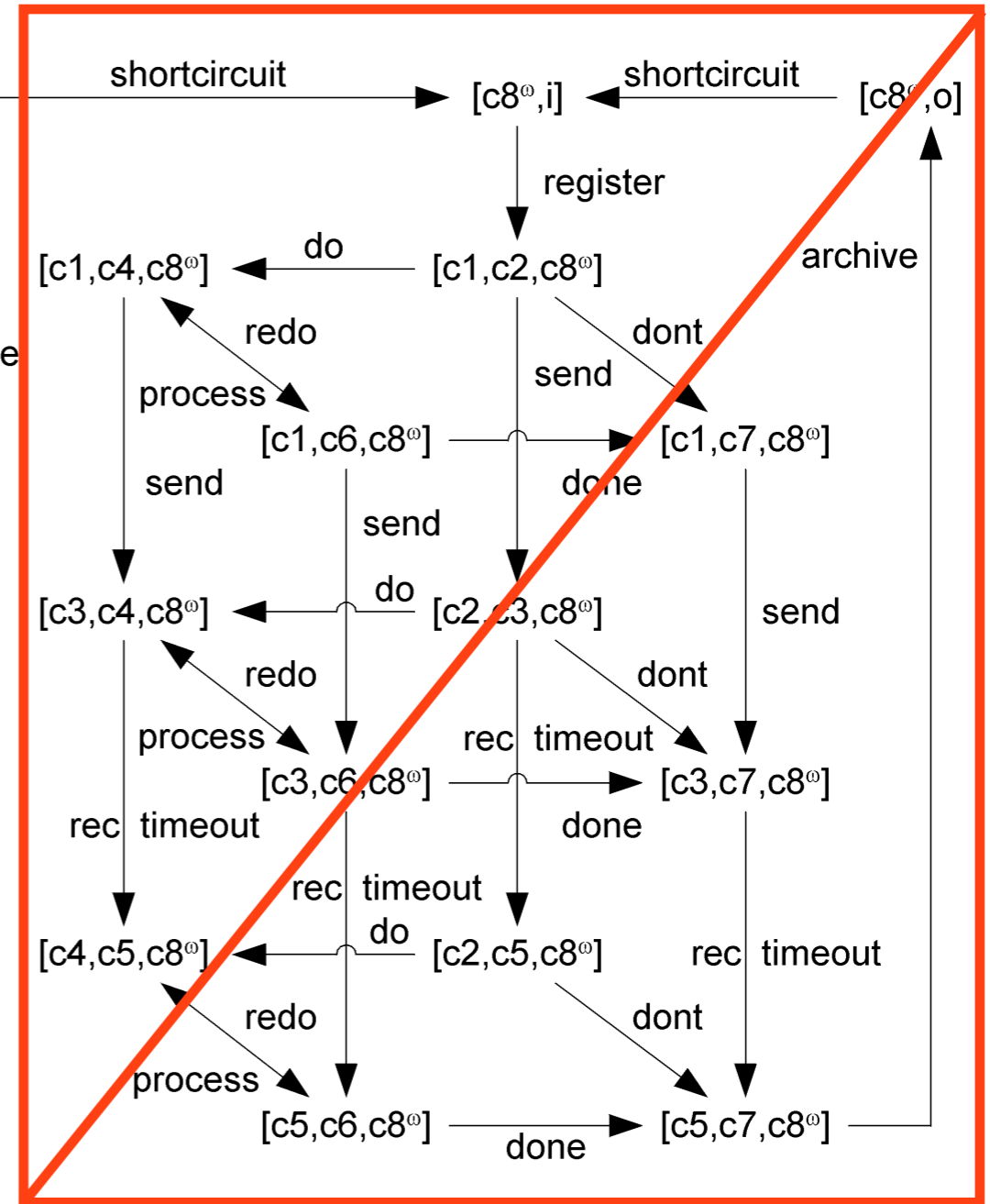
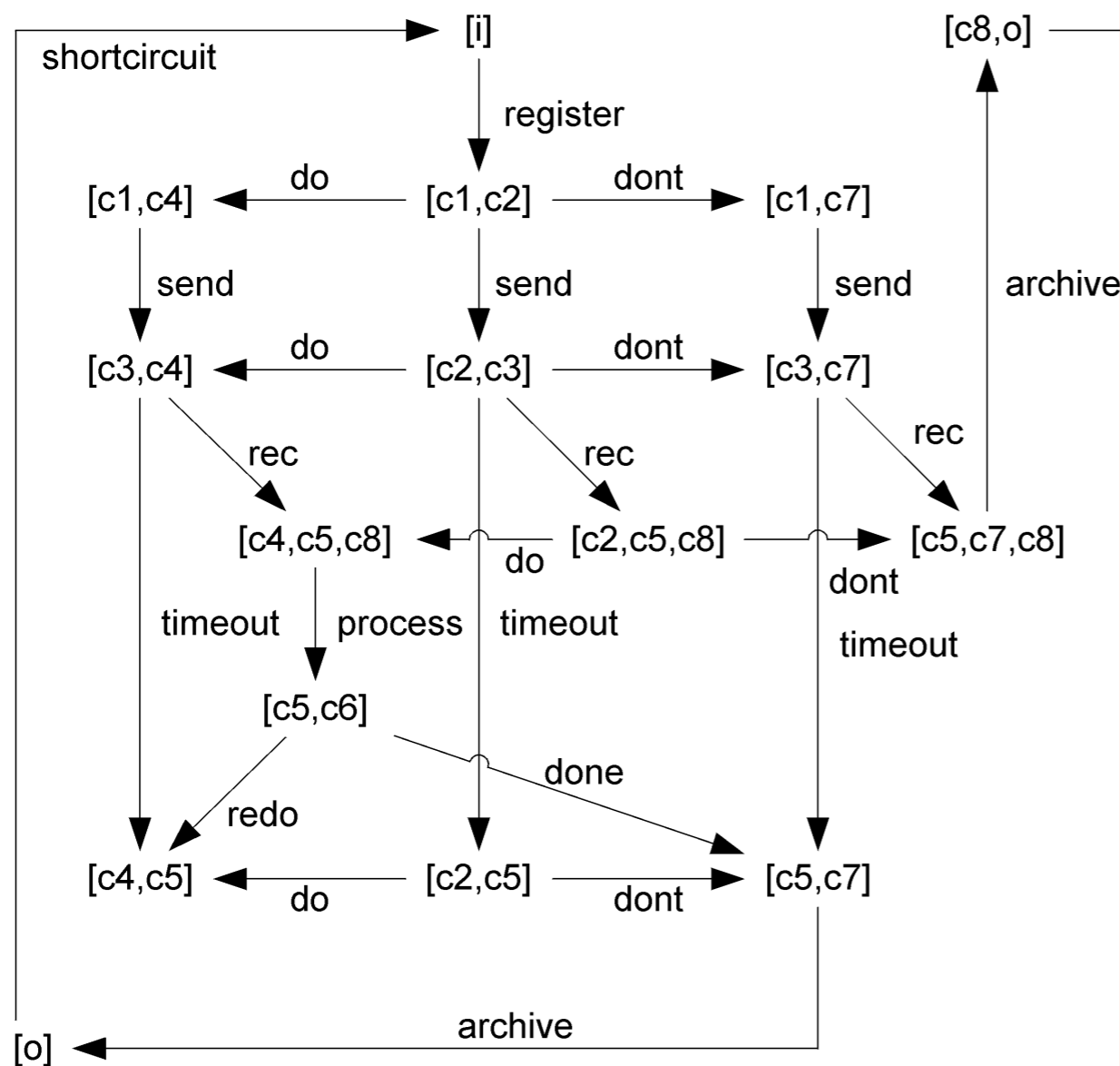
CG can become very large

Basic observation:

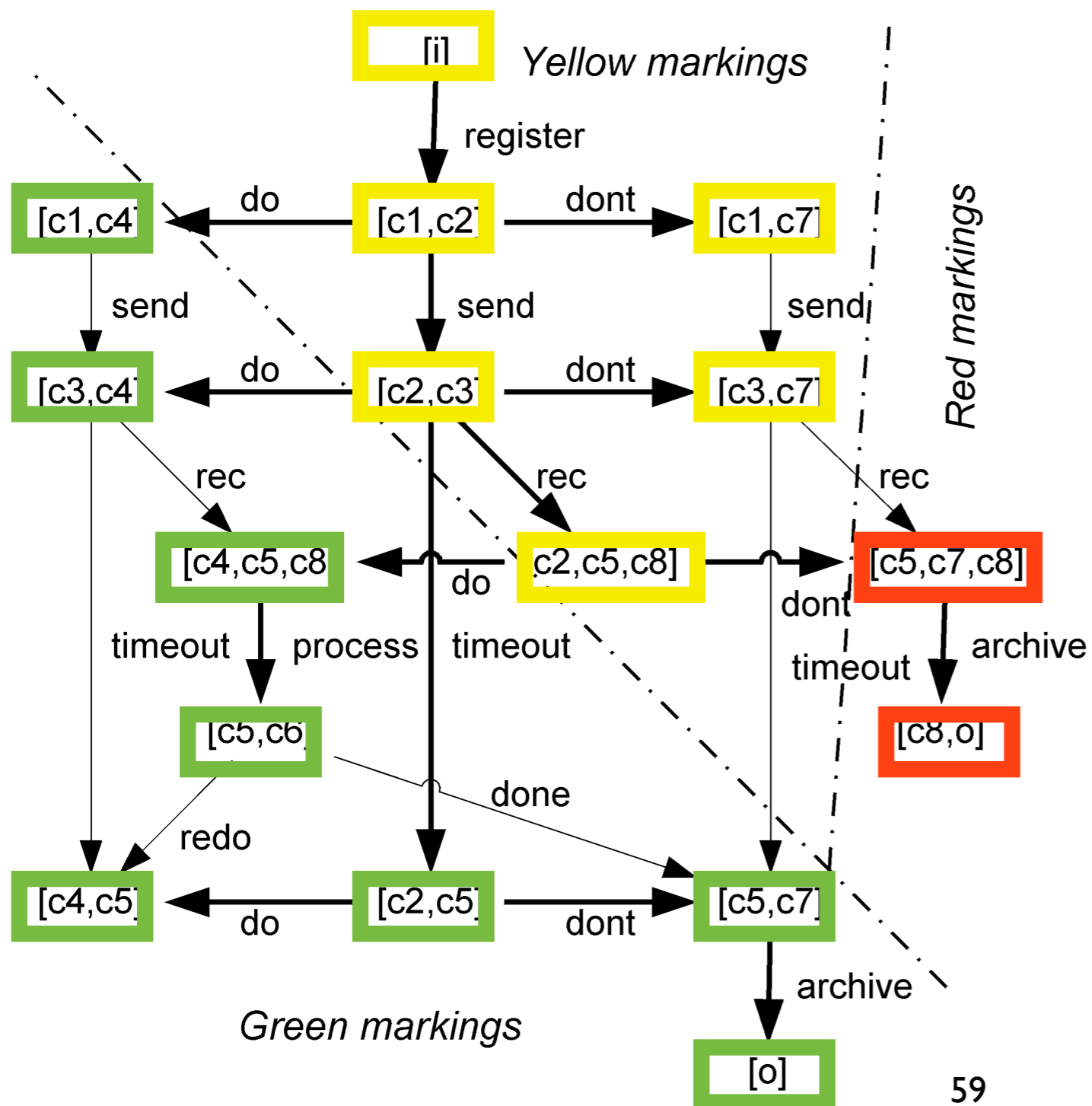
infinite-weighted markings leads to infinite-weighted markings  
and they will be all red

We can just avoid computing them!

# Example: Restricted CG vs CG



# Example: RCG (N\*)



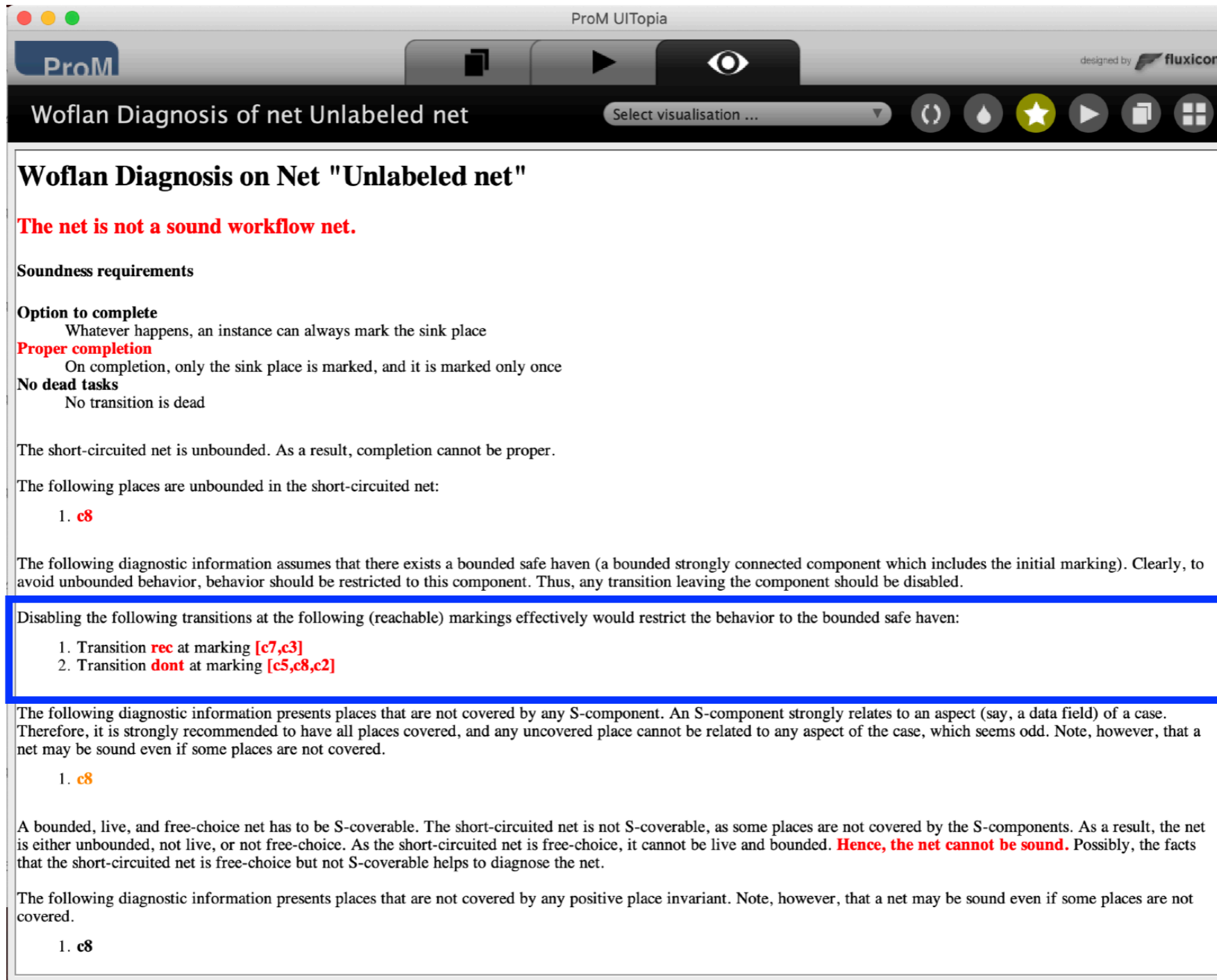
Unbounded sequences:

register, dont, send, **rec**

register, send, dont, **rec**

register, send, rec, **dont**

# Woflan (in ProM)



The screenshot shows the ProM UI with the title "Woflan Diagnosis of net Unlabeled net". The main content area displays the following text:

**Woflan Diagnosis on Net "Unlabeled net"**

**The net is not a sound workflow net.**

**Soundness requirements**

- Option to complete**  
Whatever happens, an instance can always mark the sink place
- Proper completion**  
On completion, only the sink place is marked, and it is marked only once
- No dead tasks**  
No transition is dead

The short-circuited net is unbounded. As a result, completion cannot be proper.

The following places are unbounded in the short-circuited net:

- c8**

The following diagnostic information assumes that there exists a bounded safe haven (a bounded strongly connected component which includes the initial marking). Clearly, to avoid unbounded behavior, behavior should be restricted to this component. Thus, any transition leaving the component should be disabled.

Disabling the following transitions at the following (reachable) markings effectively would restrict the behavior to the bounded safe haven:

- Transition **rec** at marking **[c7,c3]**
- Transition **dont** at marking **[c5,c8,c2]**

The following diagnostic information presents places that are not covered by any S-component. An S-component strongly relates to an aspect (say, a data field) of a case. Therefore, it is strongly recommended to have all places covered, and any uncovered place cannot be related to any aspect of the case, which seems odd. Note, however, that a net may be sound even if some places are not covered.

- c8**

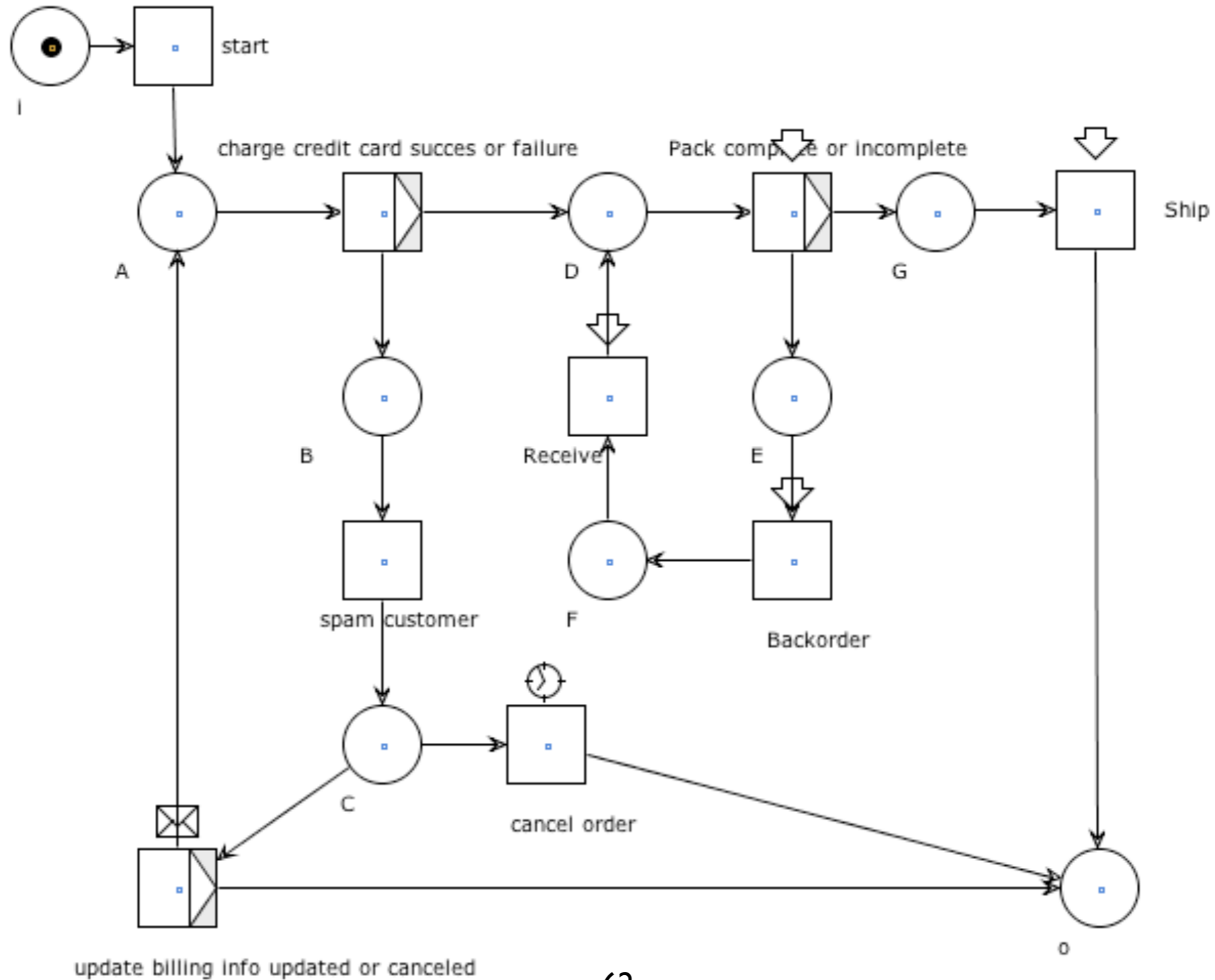
A bounded, live, and free-choice net has to be S-coverable. The short-circuited net is not S-coverable, as some places are not covered by the S-components. As a result, the net is either unbounded, not live, or not free-choice. As the short-circuited net is free-choice, it cannot be live and bounded. **Hence, the net cannot be sound.** Possibly, the facts that the short-circuited net is free-choice but not S-coverable helps to diagnose the net.

The following diagnostic information presents places that are not covered by any positive place invariant. Note, however, that a net may be sound even if some places are not covered.

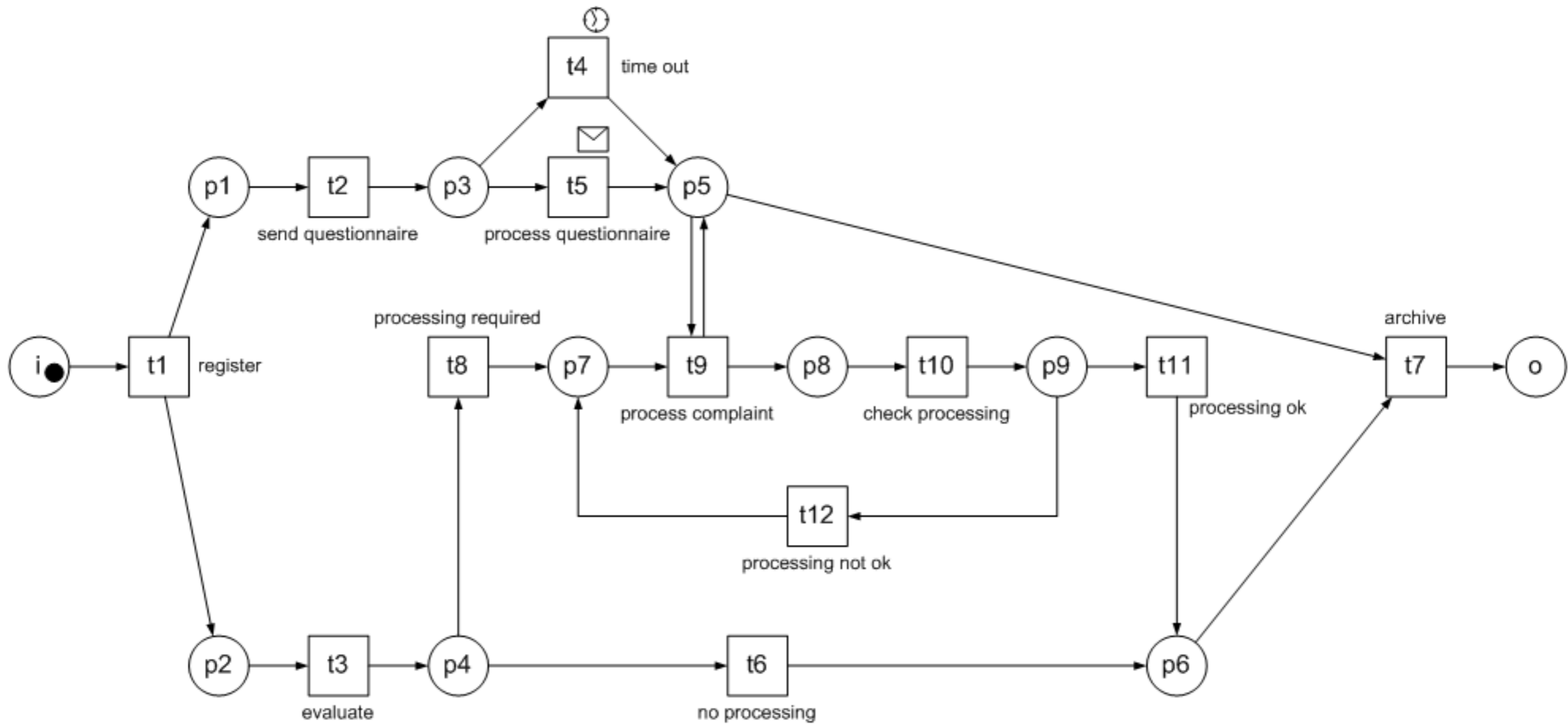
- c8**

# Practice with WoPeD (and Woflan)

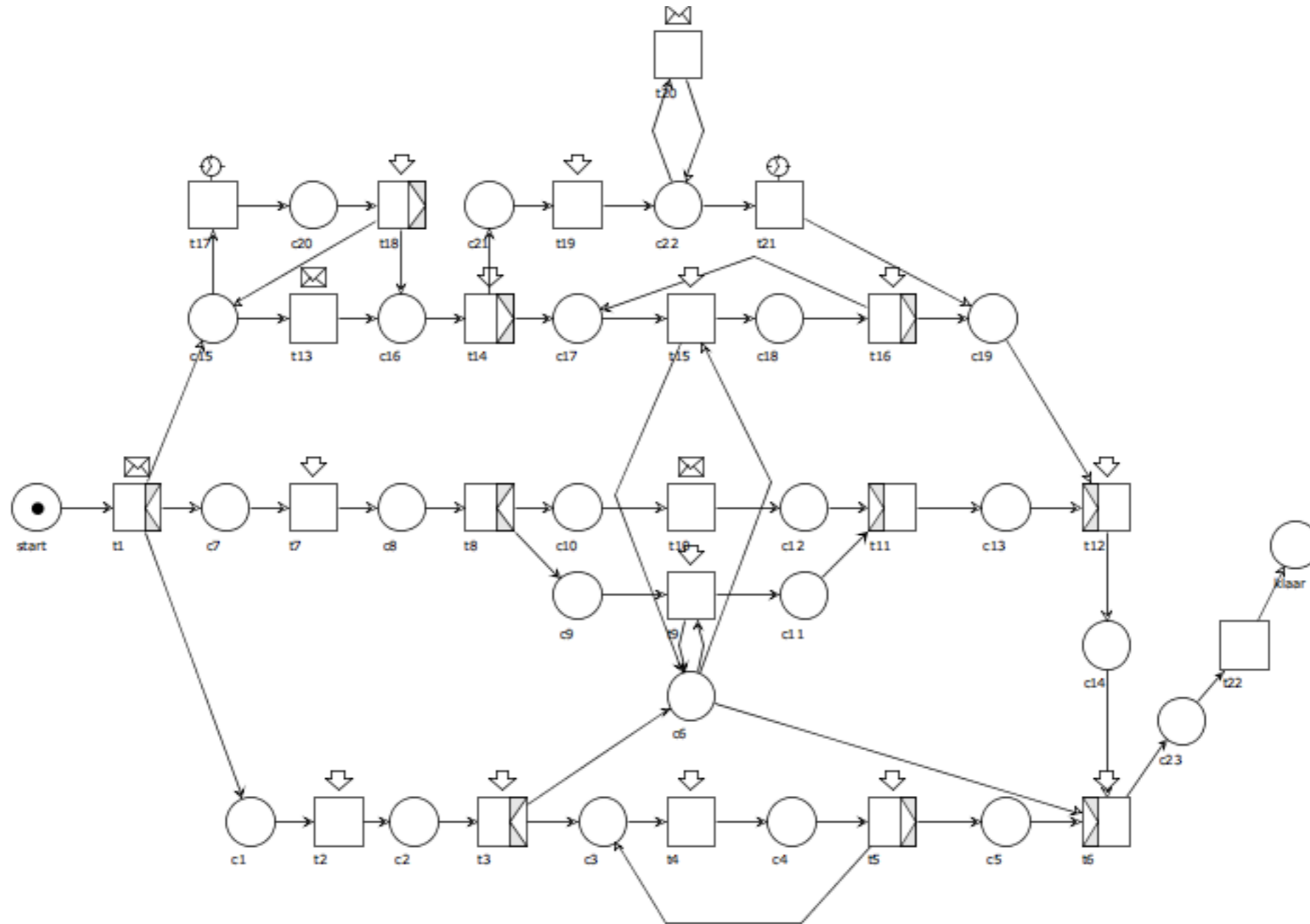
# Analyse this net



# Analyse this net

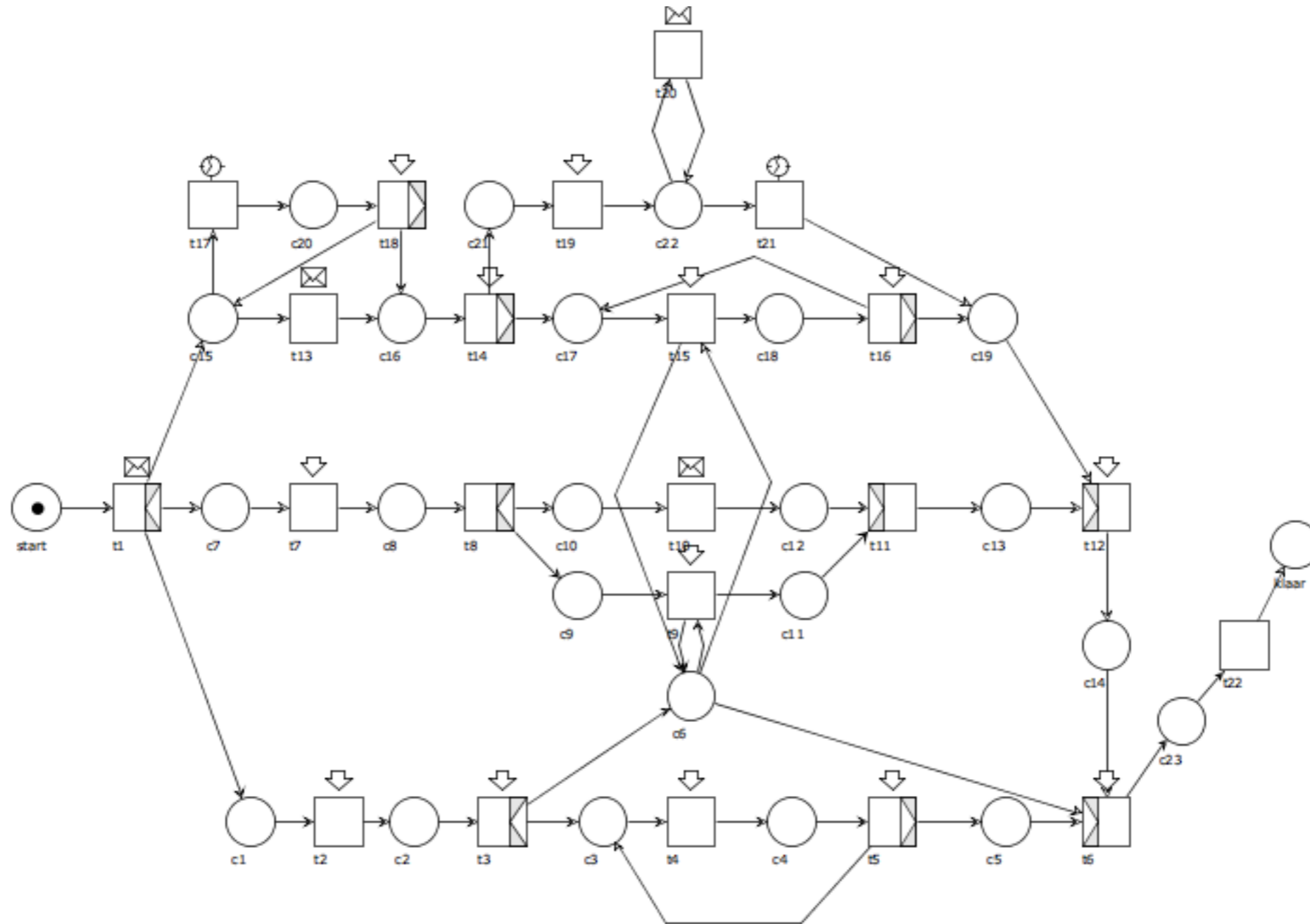


# Is this net free-choice?

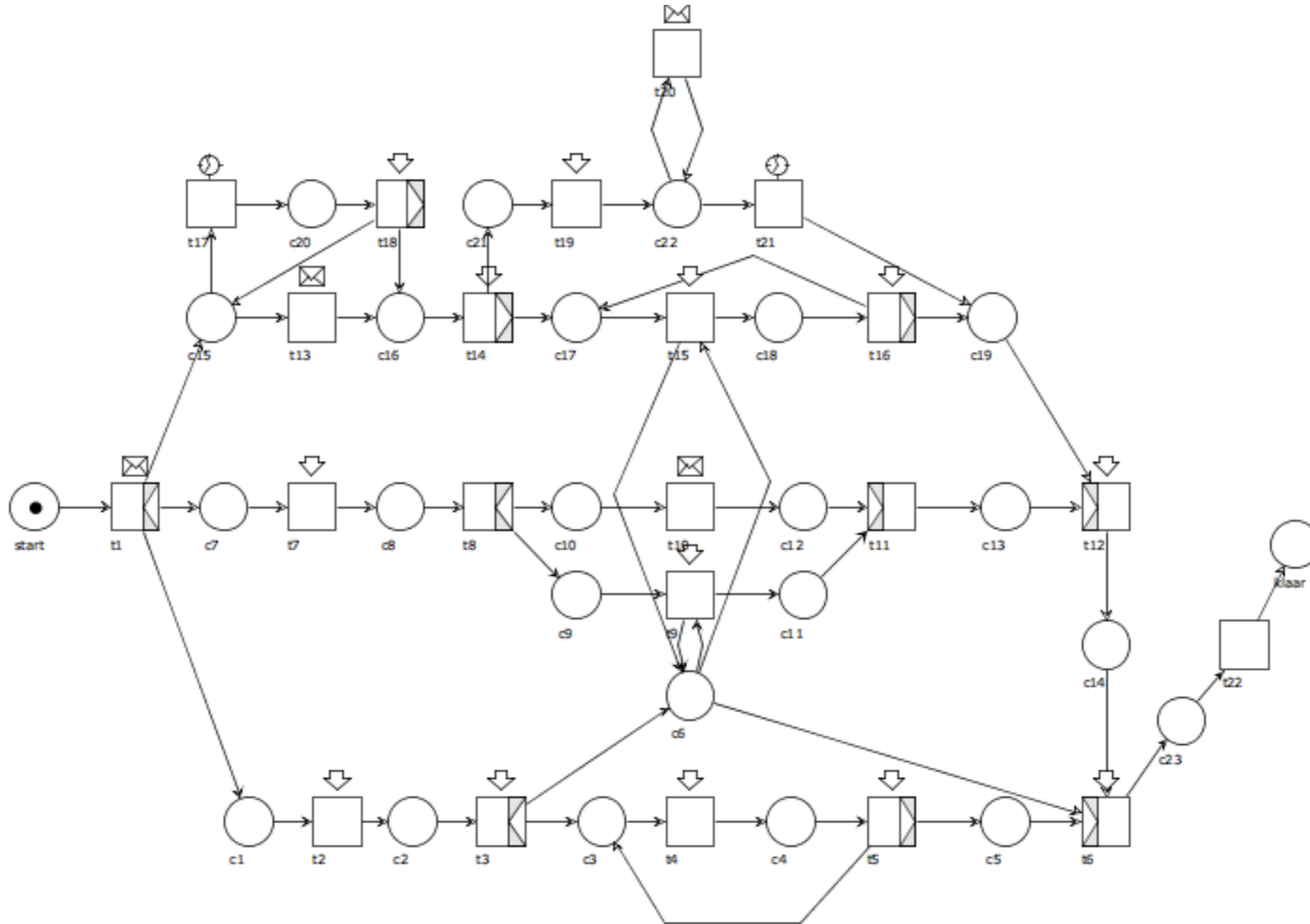




# Is this net S-coverable?



# Is this net sound?



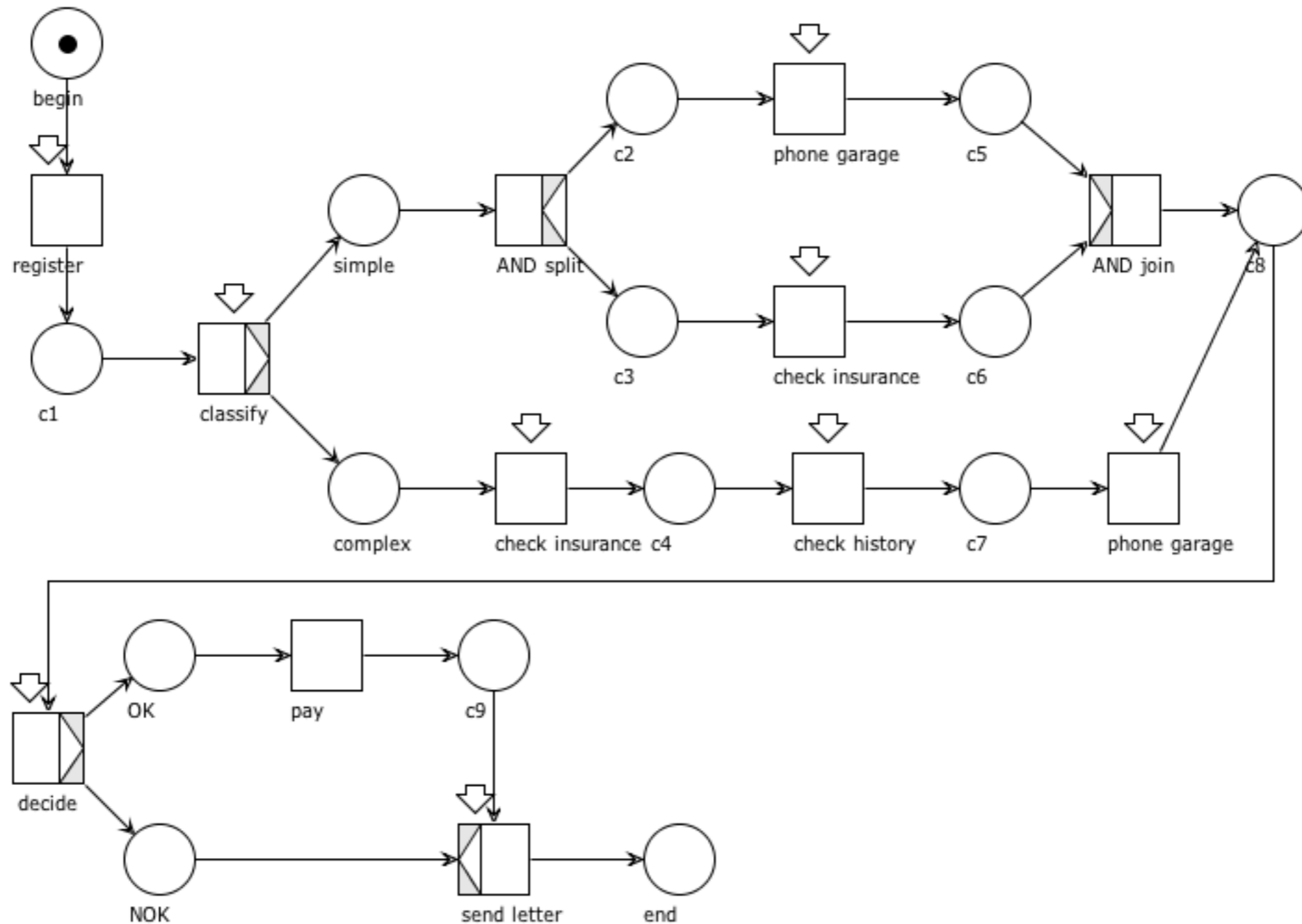
# Design Example: Car Damage

- An insurance company uses the following procedure for the processing of the claims
- Every claim, reported by a customer, is registered
- After the registration, the claim is classified
- There are two categories: simple and complex claims.
  - For simple claims two tasks need to be executed: check insurance and phone garage.  
These tasks are *independent* of each other.
  - The complex claims require three tasks: check insurance, check damage history and phone garage.  
These tasks need to be *executed sequentially* in the order specified.
- After executing the two/three tasks a decision is taken with two possible outcomes: OK (positive) or NOK (negative).
- If the decision is positive, then insurance company will pay.
- In any event, the insurance company sends a letter to the customer.

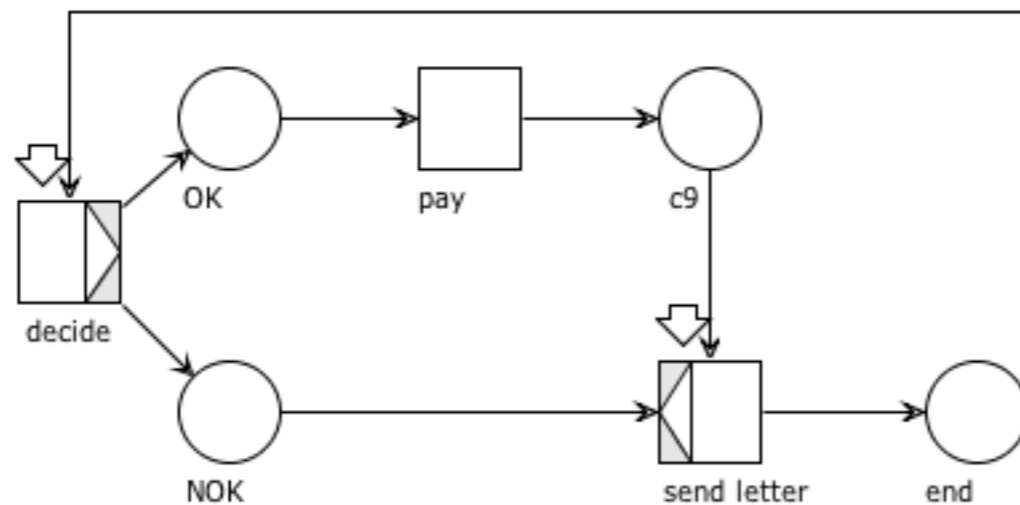
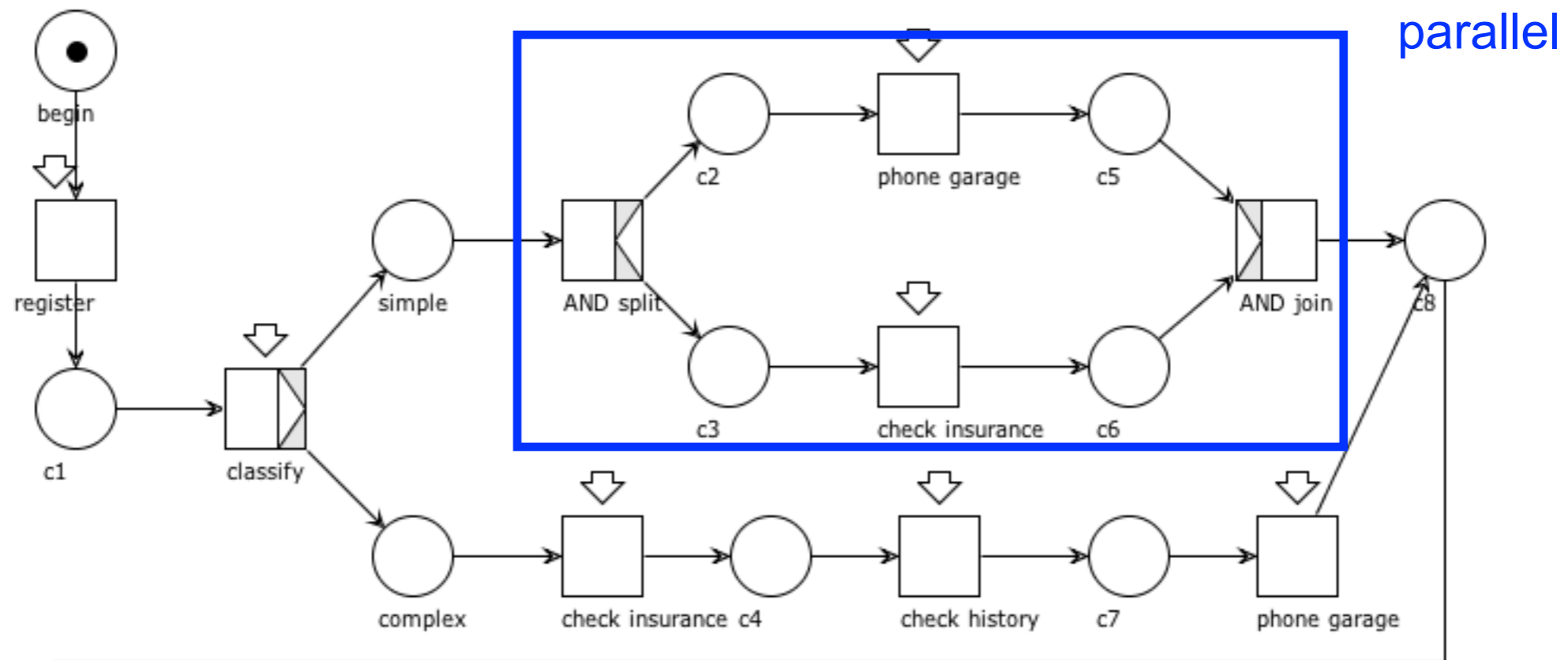
# Design Example: Car Damage

- An insurance company uses the following procedure for the processing of the claims
- Every claim, reported by a customer, is **registered**
- After the registration, the claim is **classified**
- There are two categories: **simple** and **complex** claims.
  - For simple claims two tasks need to be executed: **check insurance** and **phone garage**.  
These tasks are *independent* of each other.
  - The complex claims require three tasks: **check insurance**, **check damage history** and **phone garage**.  
These tasks need to be *executed sequentially* in the order specified.
- After executing the two/three tasks a **decision** is taken with two possible outcomes: **OK** (positive) or **NOK** (negative).
- If the decision is positive, then insurance company will **pay**.
- In any event, the insurance company **sends a letter** to the customer.

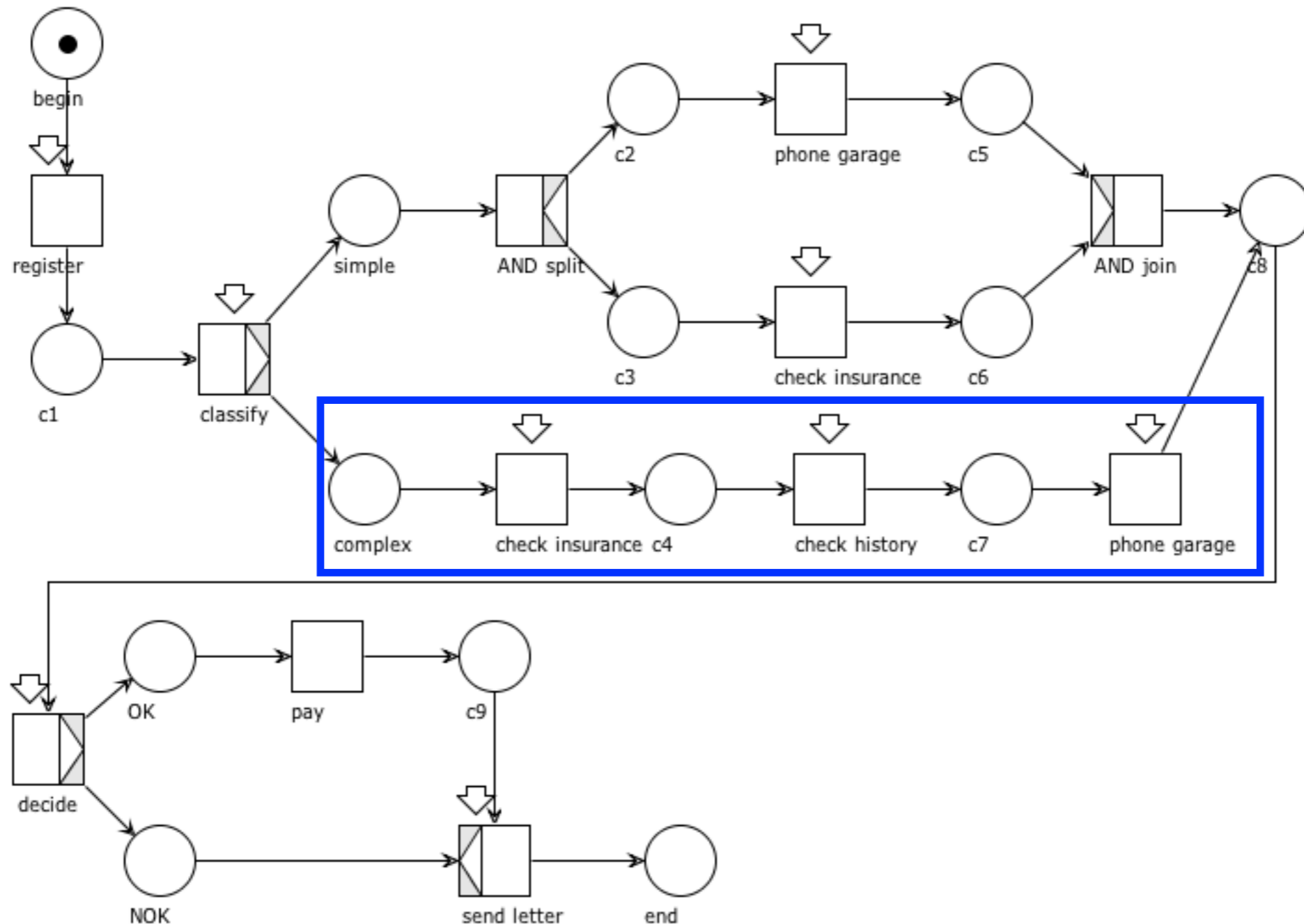
# Design Example: Car Damage



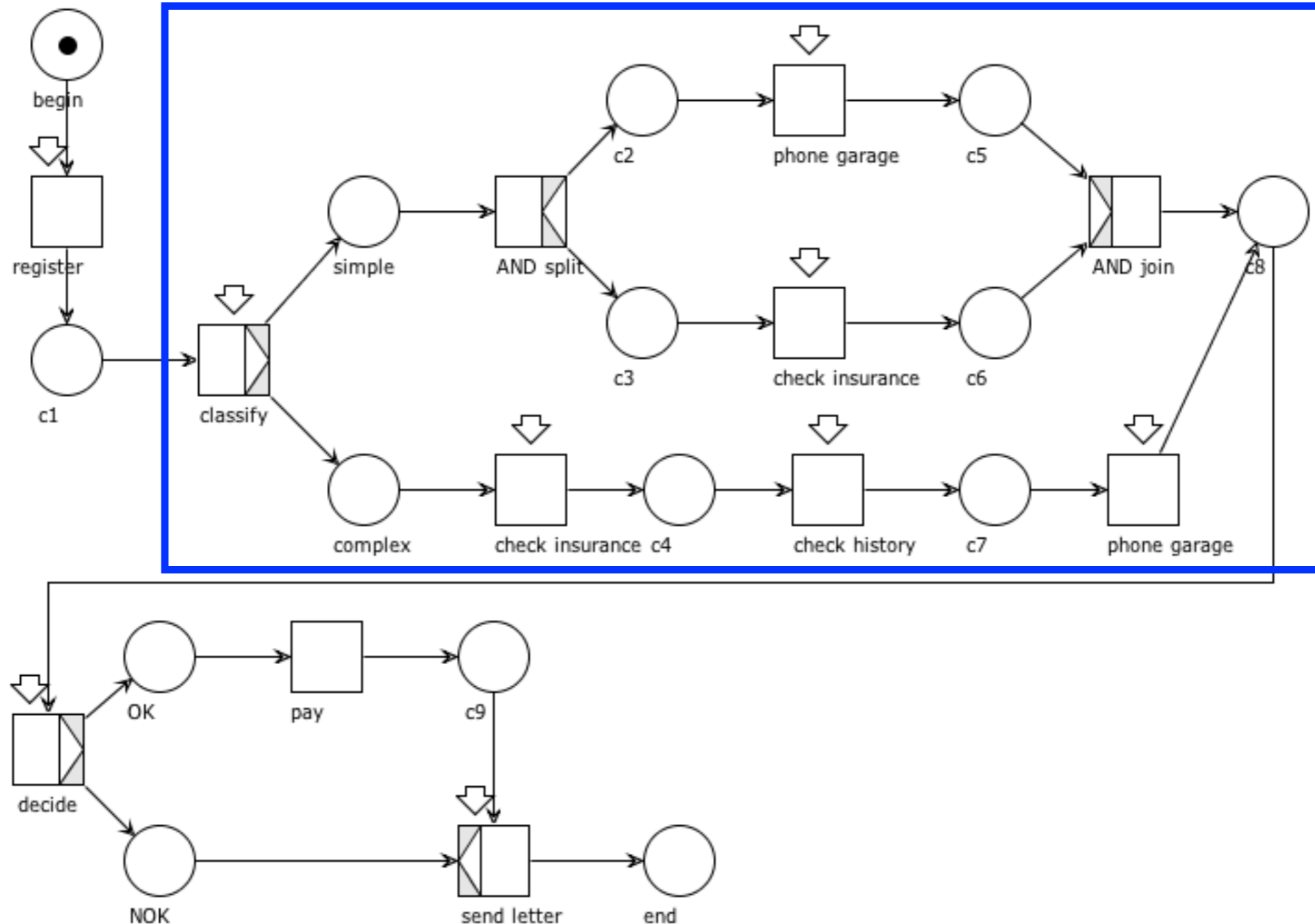
# Design Example: Car Damage



# Design Example: Car Damage



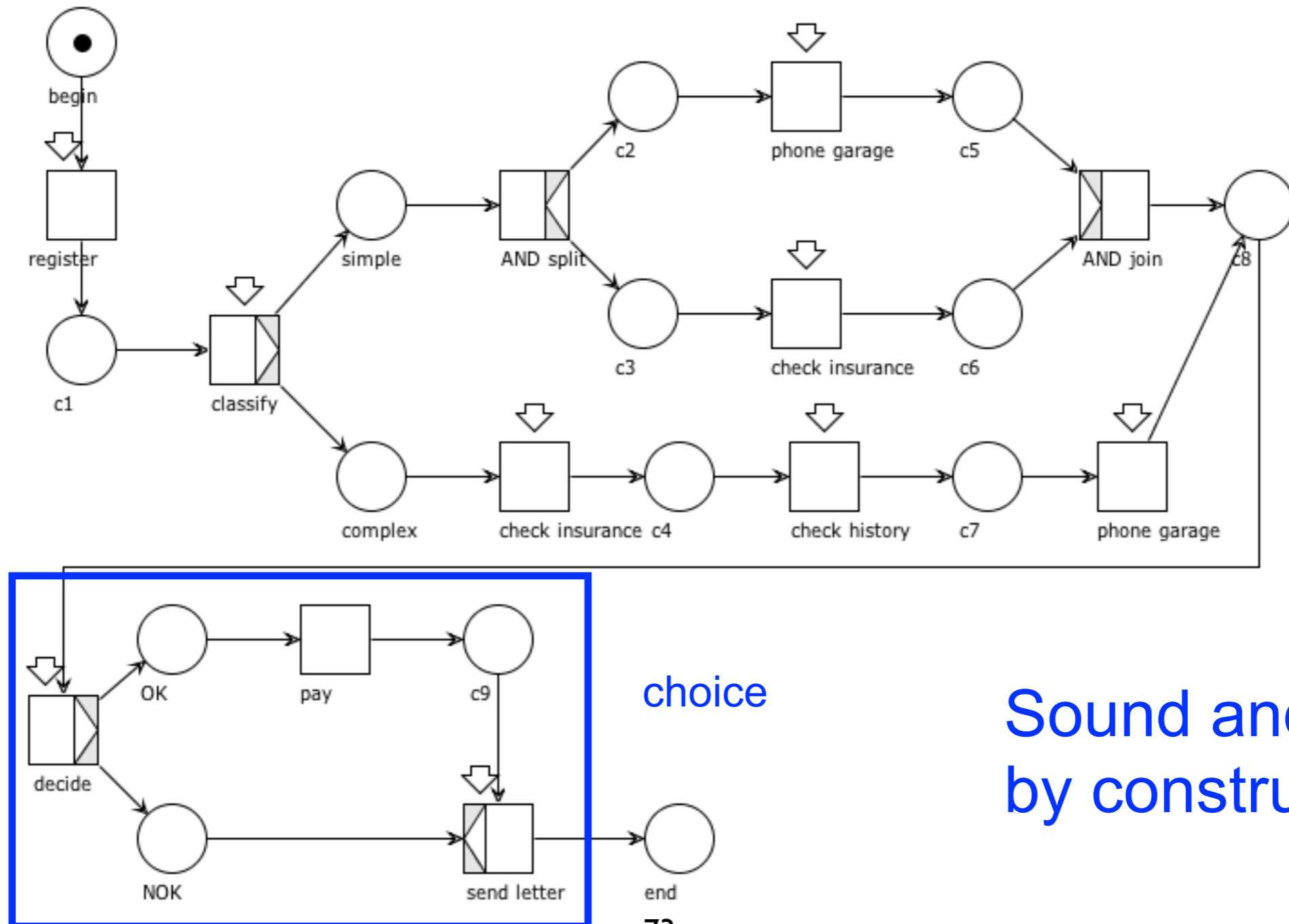
# Design Example: Car Damage



choice



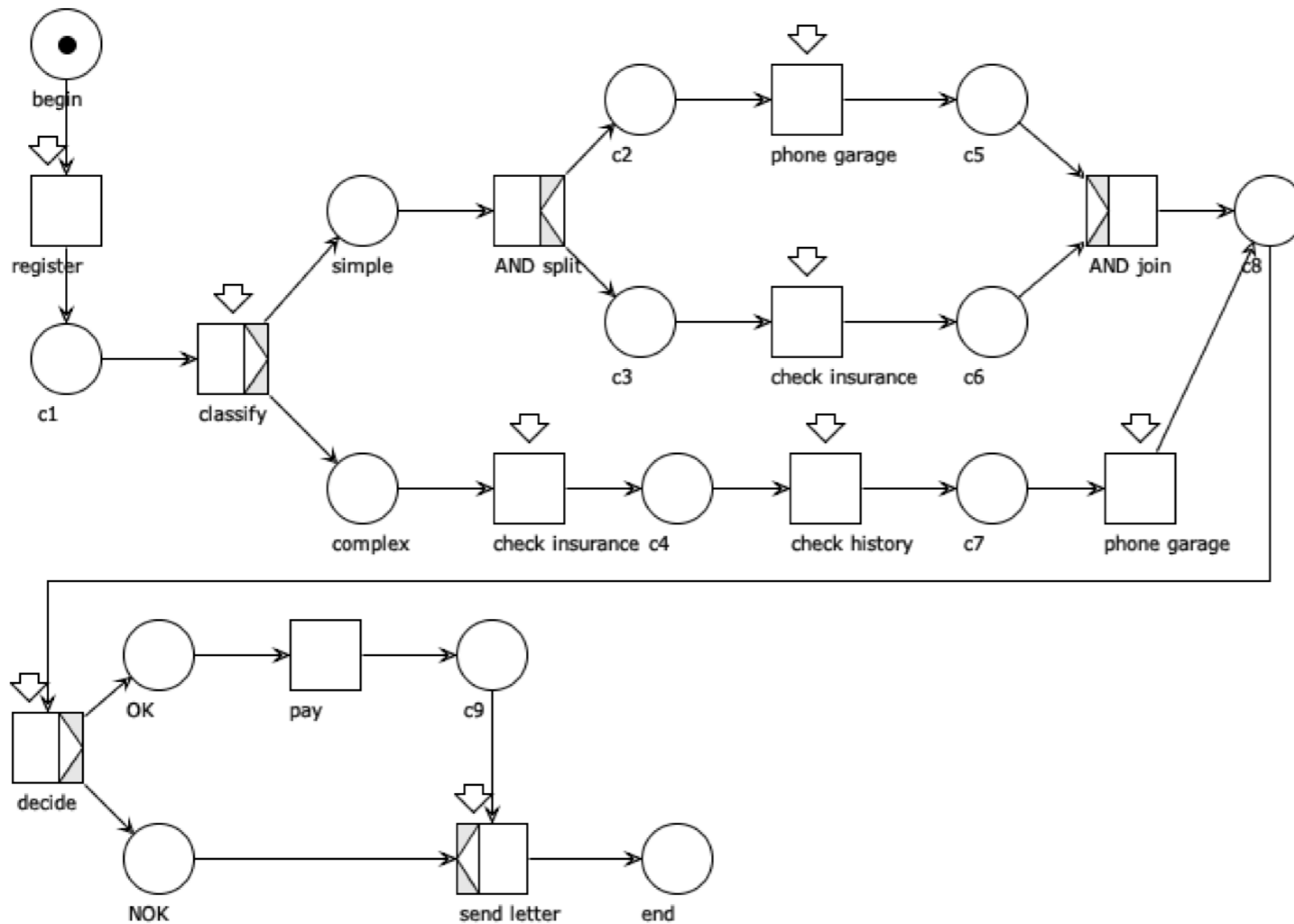
# Design Example: Car Damage



choice

Sound and safe  
by construction!

# Design Example: Car Damage



**Semantical analysis**

Wizard Expert

- ✓ Qualitative analysis
  - ✓ Structural analysis
    - ▶ Net statistics
      - ✓ Wrongly used operators: 0
      - ✓ Free-choice violations: 0
  - ✓ S-Components
    - ▶ S-Components: 2
      - ✓ Places not covered by S-Comp
  - ✓ Wellstructuredness
    - ✓ PT-Handles: 0
    - ✓ TP-Handles: 0
- ✓ Soundness
  - ▶ Workflow net property
  - ▶ Initial marking
- ✓ Boundedness
  - ✓ Unbounded places: 0
- ✓ Liveness
  - ✓ Dead transitions: 0
  - ✓ Non-live transitions: 0

# 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.