

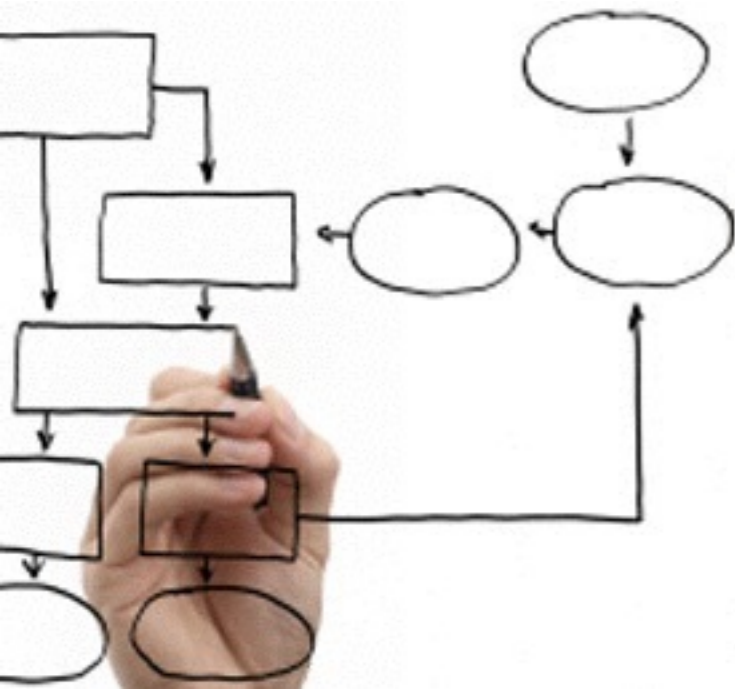
Methods for the specification and verification of business processes

MPB (6 cfu, 295AA)

Roberto Bruni

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

18 - Workflow modules



Object

We study Workflow modules to model interaction between workflows

Problem

Not all tasks of a workflow net are automatic:
they can be triggered manually or by a message

they can be used to trigger other tasks

How do we represent this?

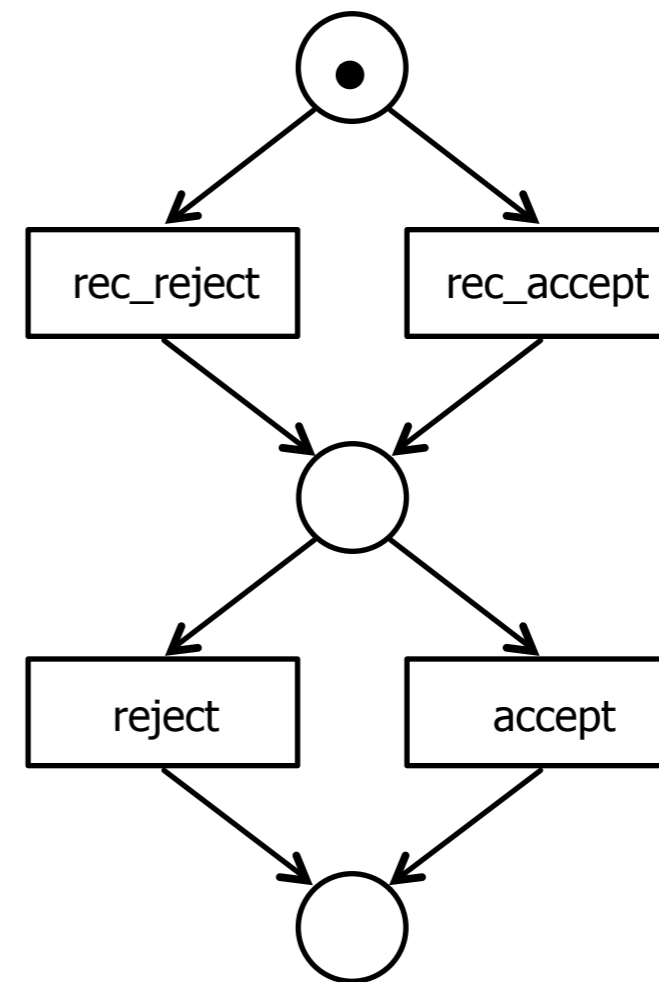
Implicit interaction

Separately developed
workflow

Some activities can
input messages

Some activities can
output messages

Seller

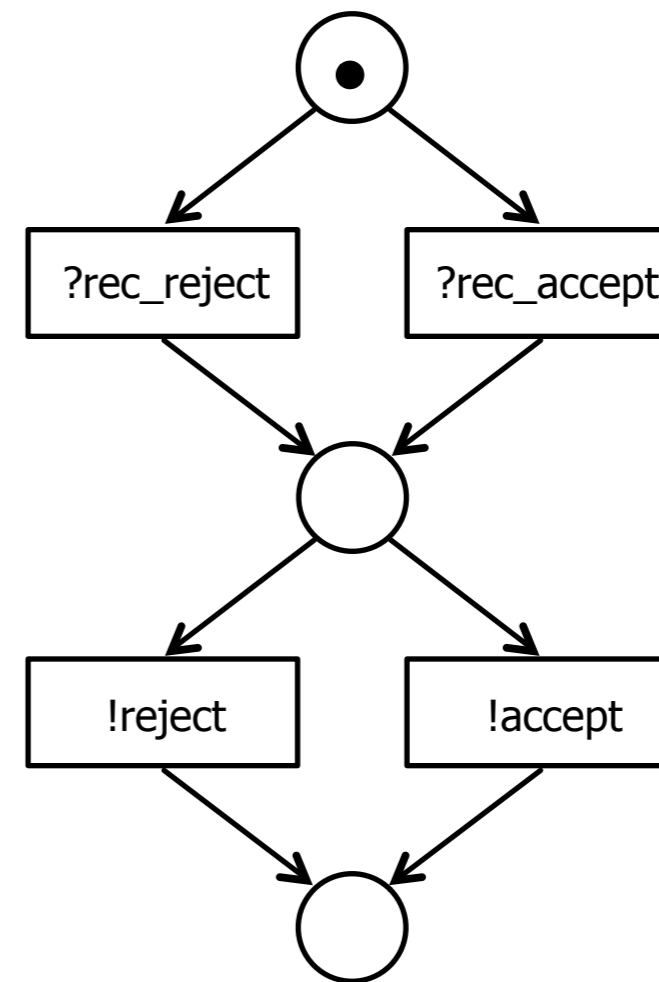


Implicit interaction

Seller can receive
(symbol ?)
recommendations

Seller can send
(symbol !)
decisions

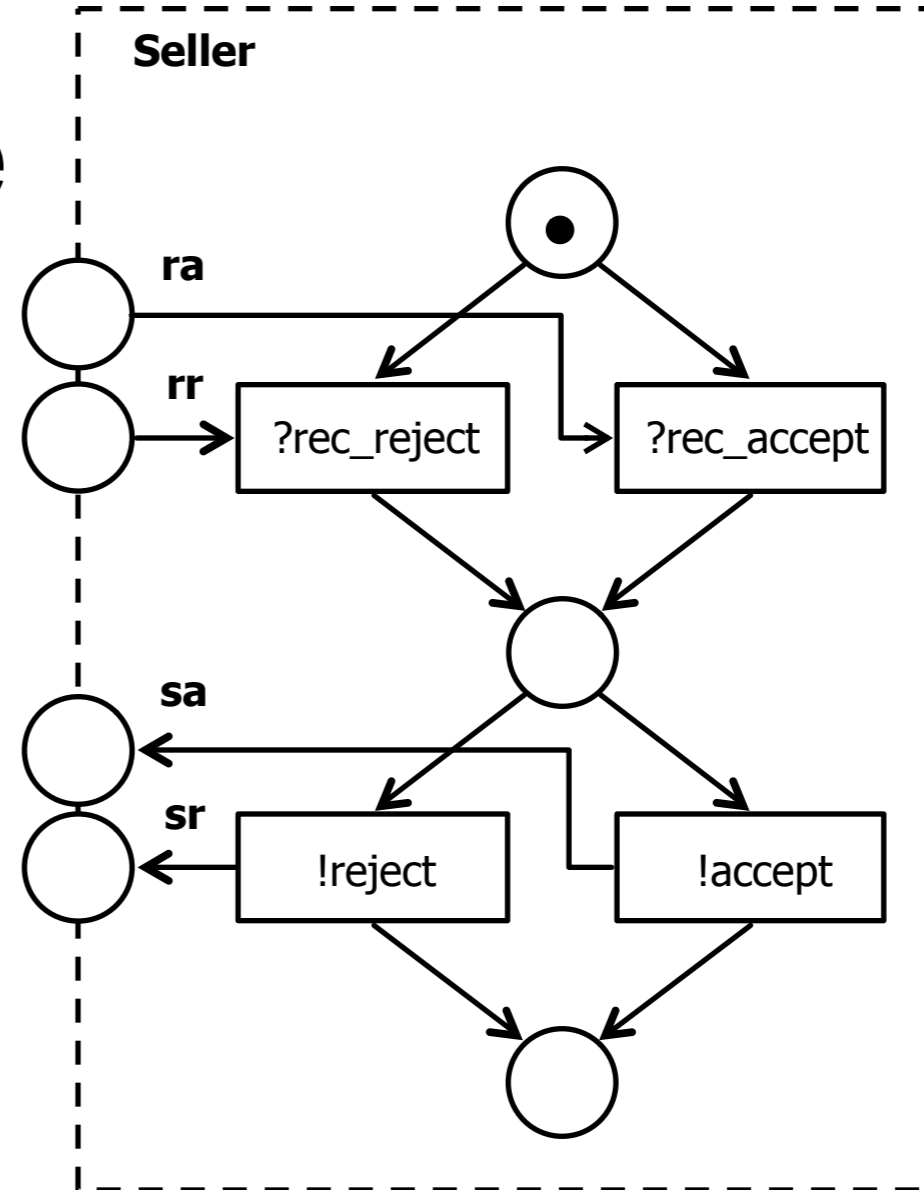
Seller



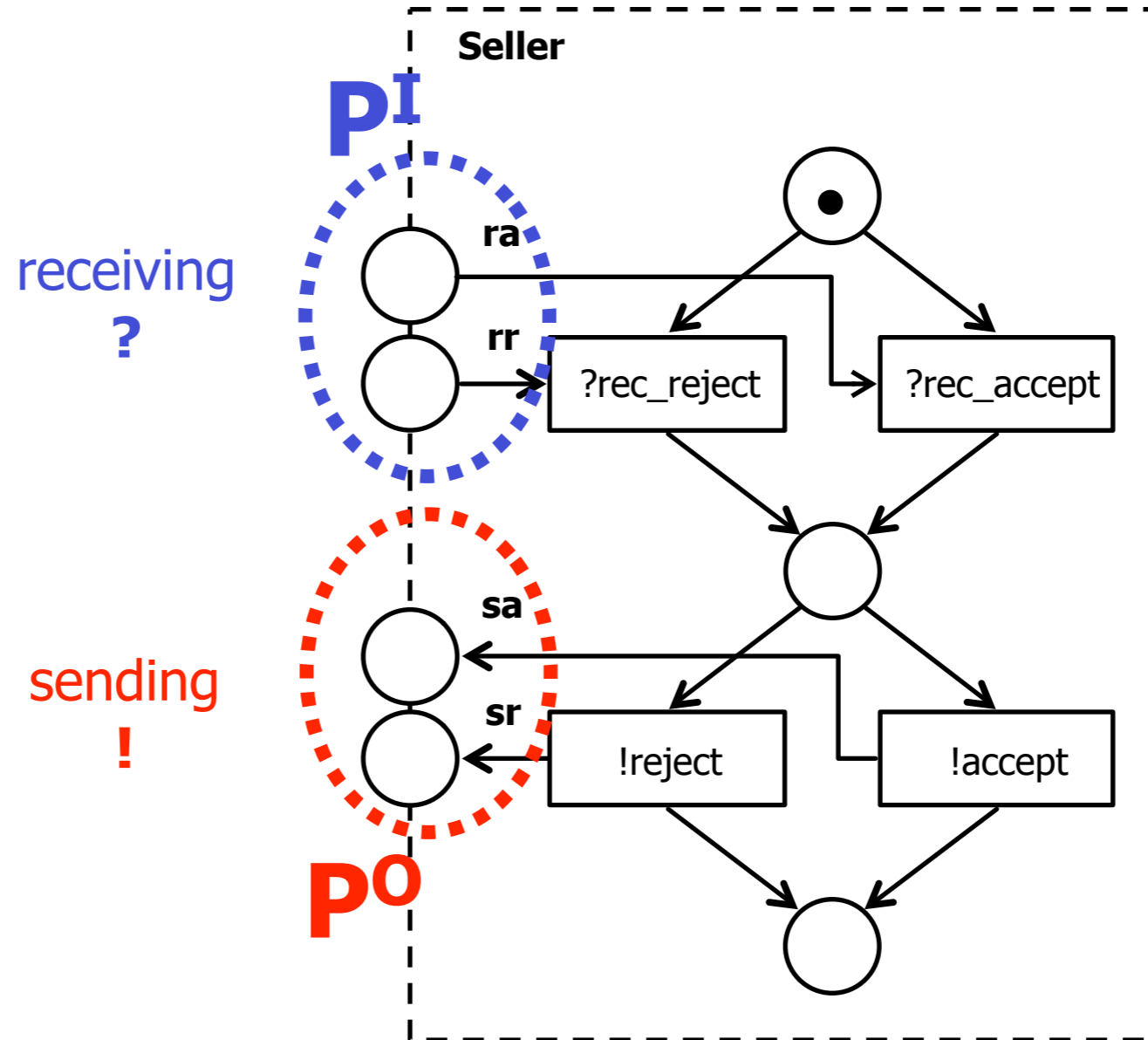
Interface

Seller has an interface
for interaction

It consists of
some input places
and
some output places



Interface



Problem

Assume the original workflow net has been validated:

it is a sound (and maybe safe) workflow net

When we add the interface
it is no longer a workflow net!

Workflow Modules

Definition: A **workflow module** consists of

a workflow net (P, T, F)

plus a set P^I of incoming places

plus a set of incoming arcs $F^I \subseteq (P^I \times T)$

plus a set P^O of outgoing places

plus a set of outgoing arcs $F^O \subseteq (T \times P^O)$

such that no transition is connected to both
an incoming and an outgoing place

Problem

Workflow modules must be capable to interact

How do we check that their interfaces match?

How do we combine them together?

Strong structural compatibility

A set of workflow modules is called
strongly structural compatible

if

for every message that can be sent there is a
workflow who can receive it,

and

for every message that can be received there is a
participant who can send it

(formats of message data are assumed to match)

Weak structural compatibility

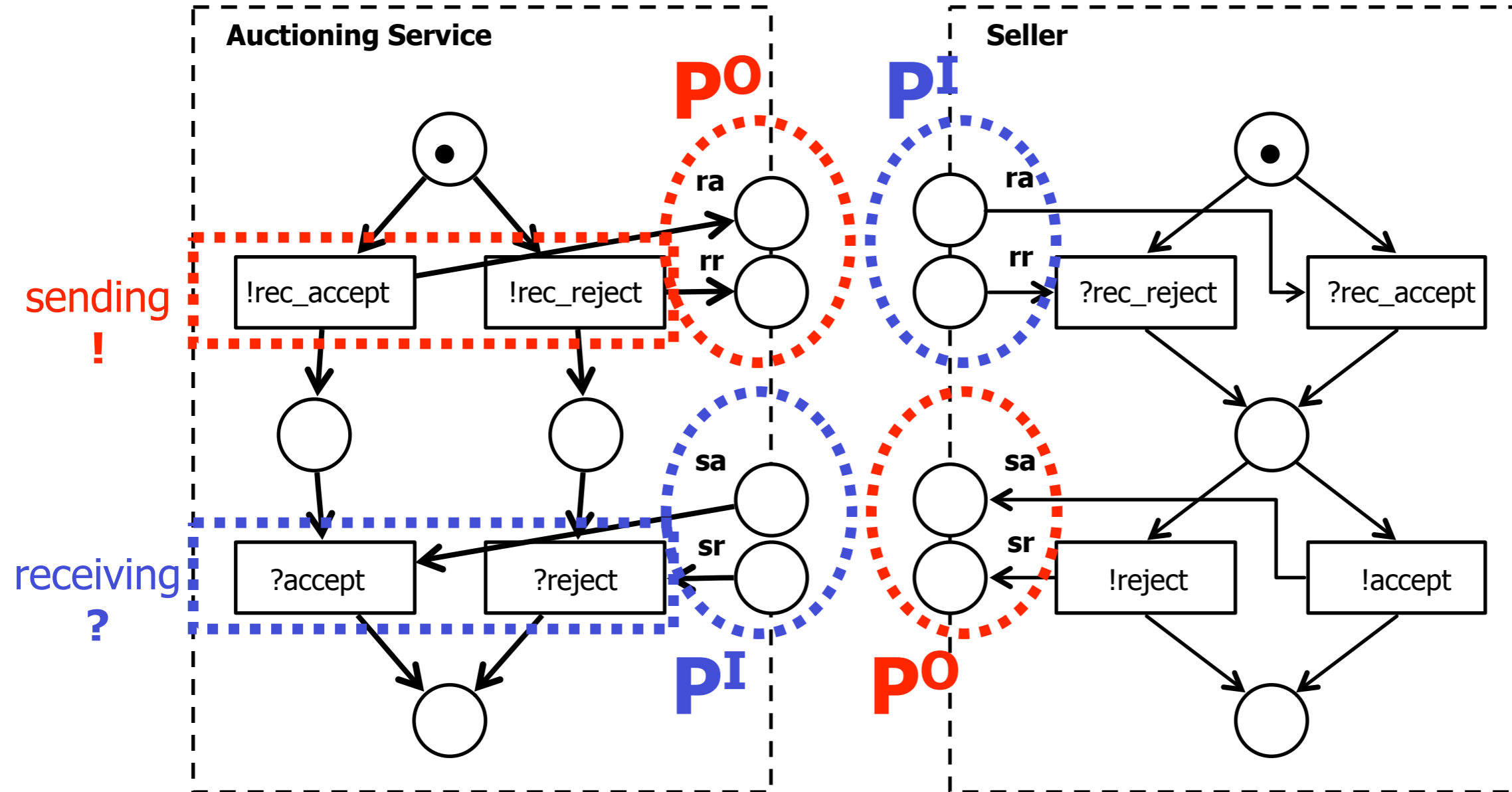
A set of workflow modules is called
weakly structural compatible

if

all messages sent by participants
can be received by other participants

more likely than a complete structural match
(workflow modules are developed separately)

Interaction



Problem

We have added places and arcs to single nets

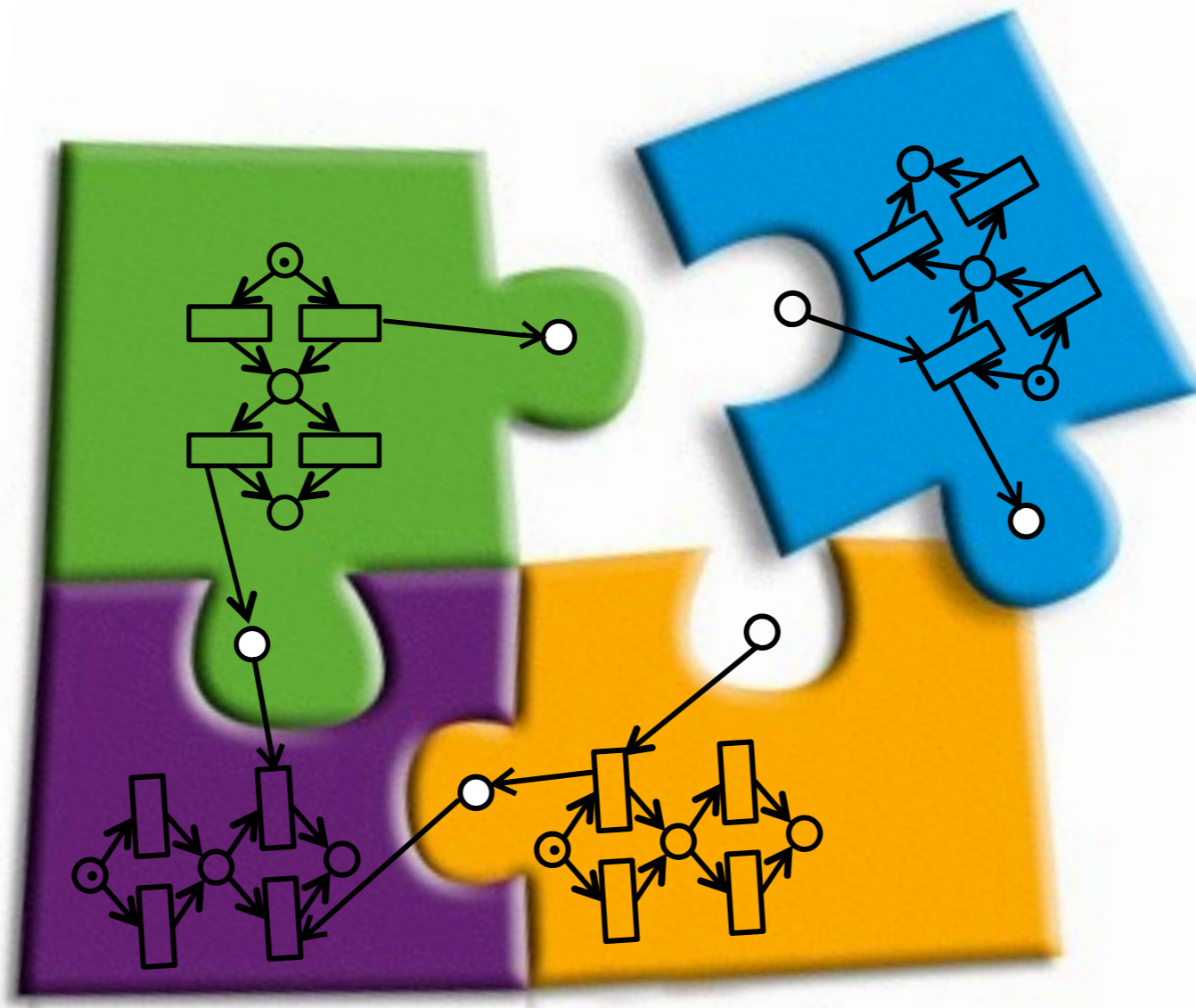
We have joined places of different nets

We have joined their initial markings

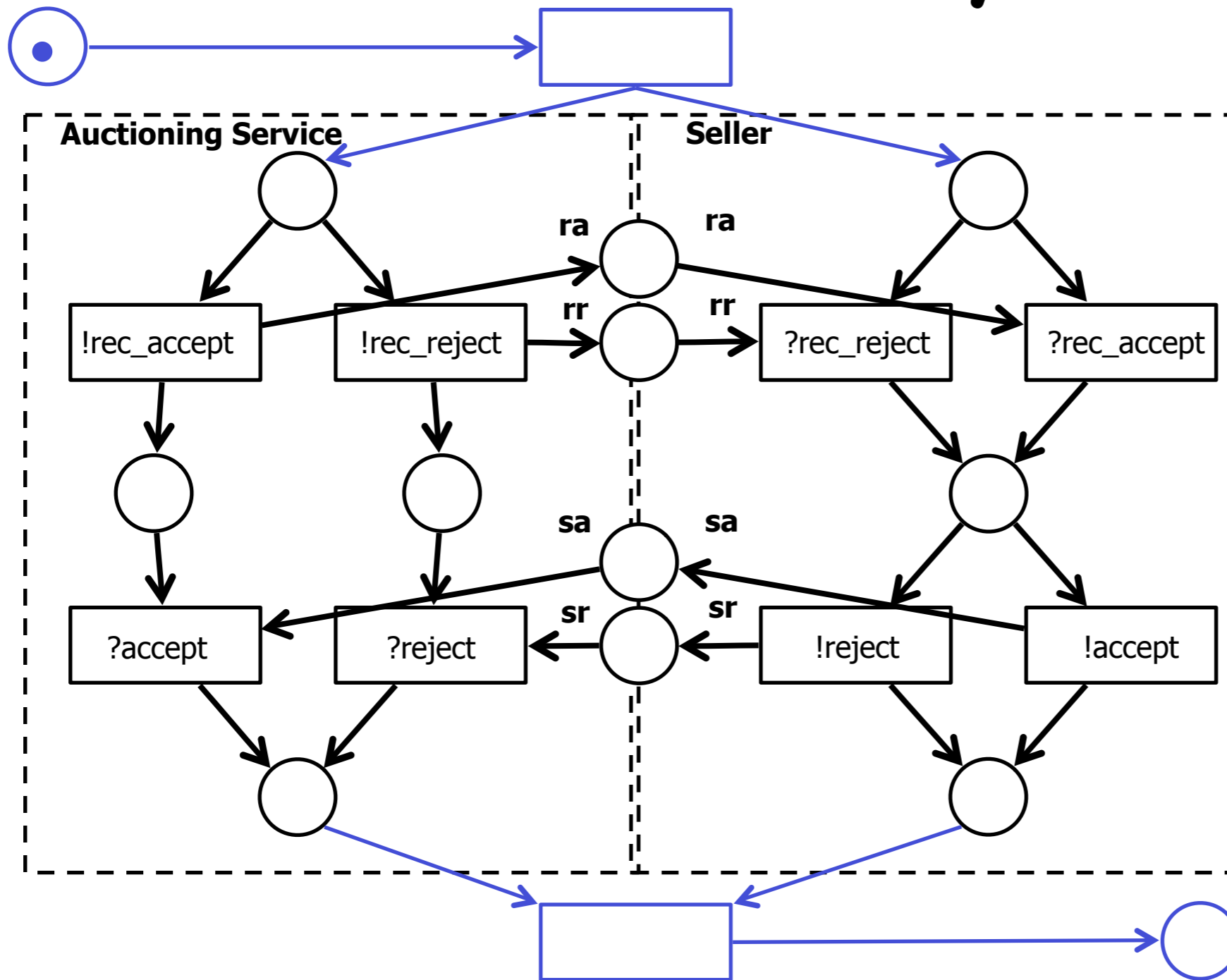
How do we check that the system behaves well?

What has this check to do with WF net soundness?

Workflow systems



Workflow system



Workflow system

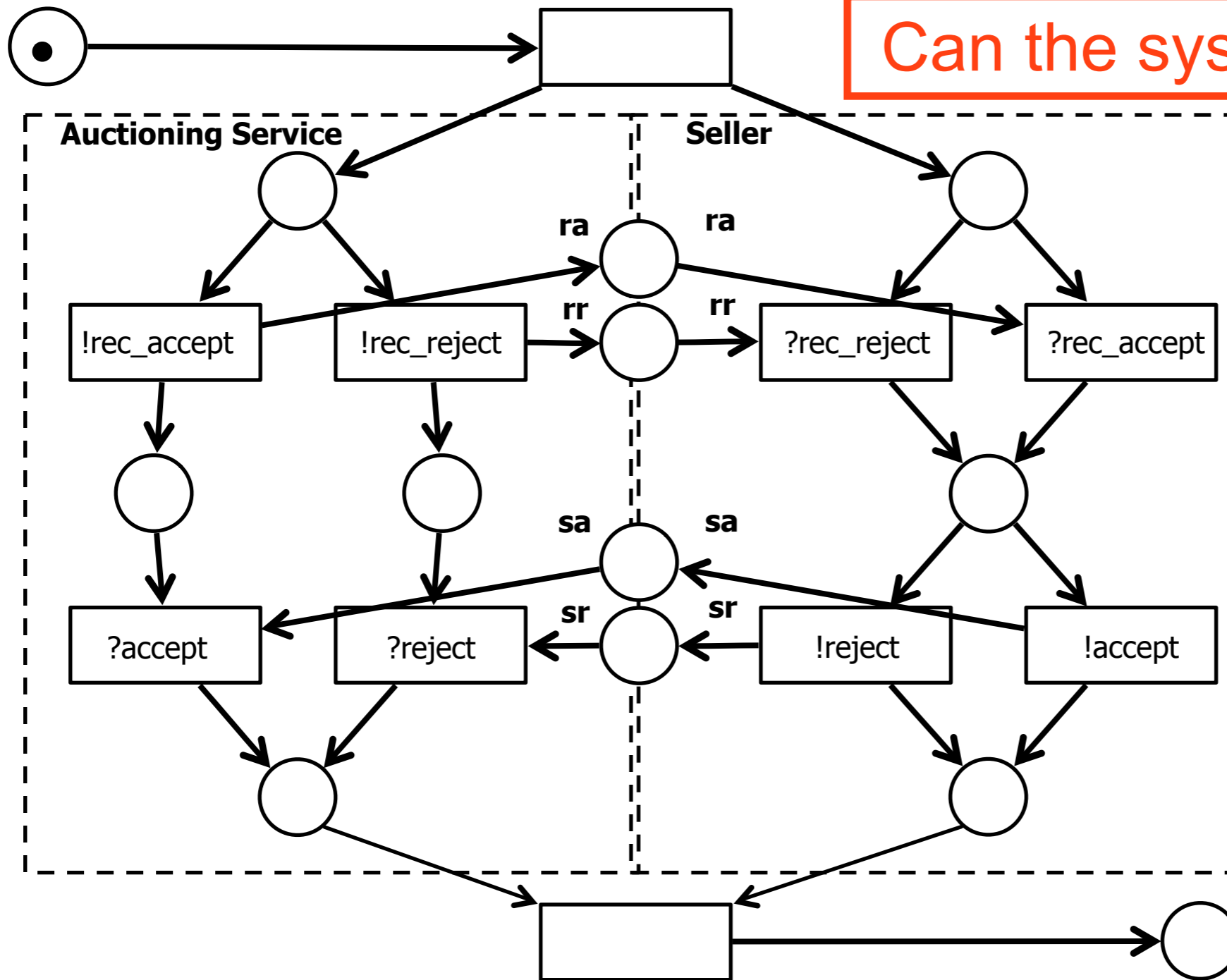
Definition: A **workflow system** consists of
a set of n structurally compatible workflow modules
(initial places i_1, \dots, i_n , final places o_1, \dots, o_n)

plus an initial place **i**
and a transition **t_i** from **i** to **i_1, \dots, i_n**

plus a final place **o**
and a transition **t_o** from **o_1, \dots, o_n** to **o**

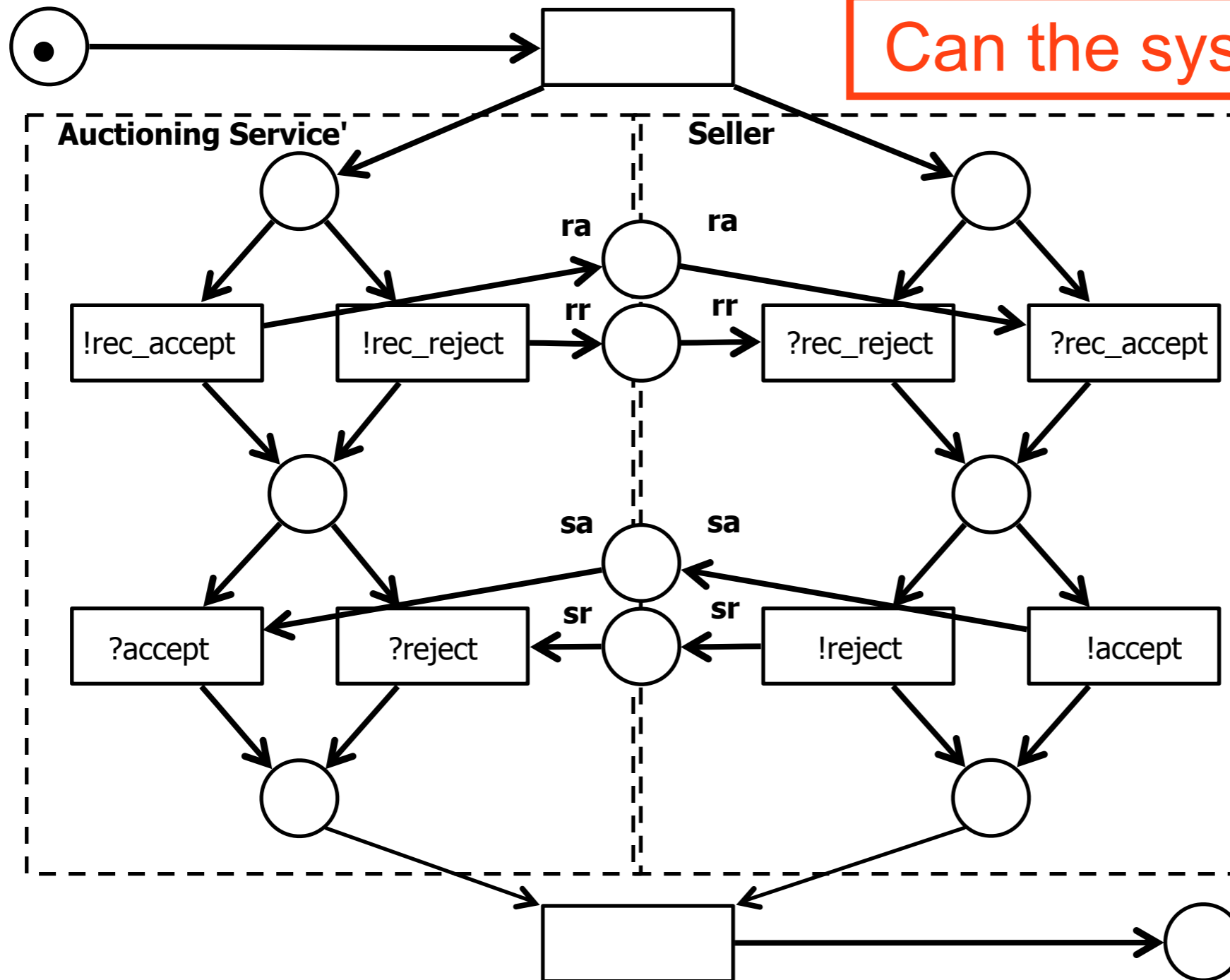
Exercise

Can the system deadlock?



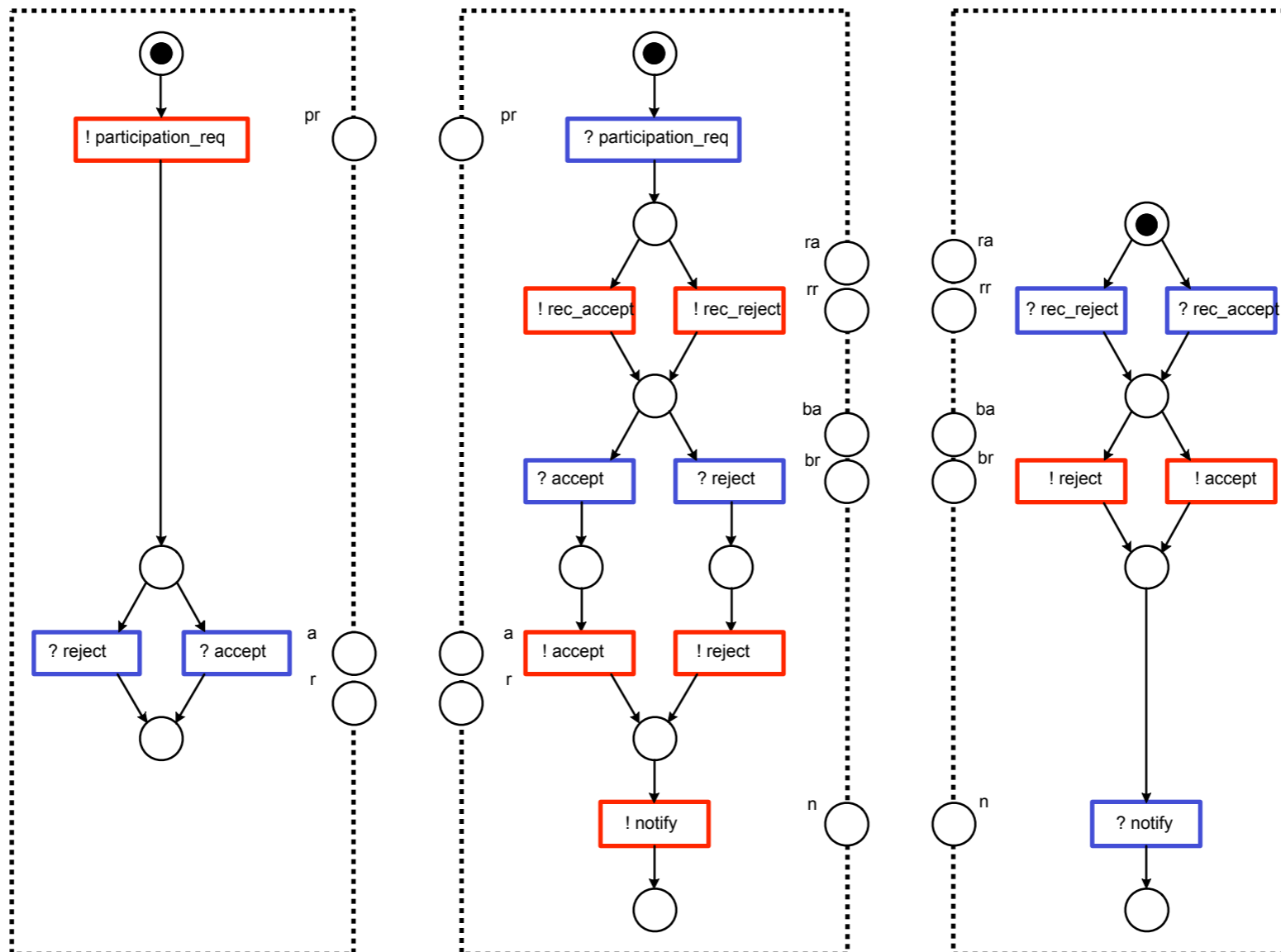
Exercise

Can the system deadlock?



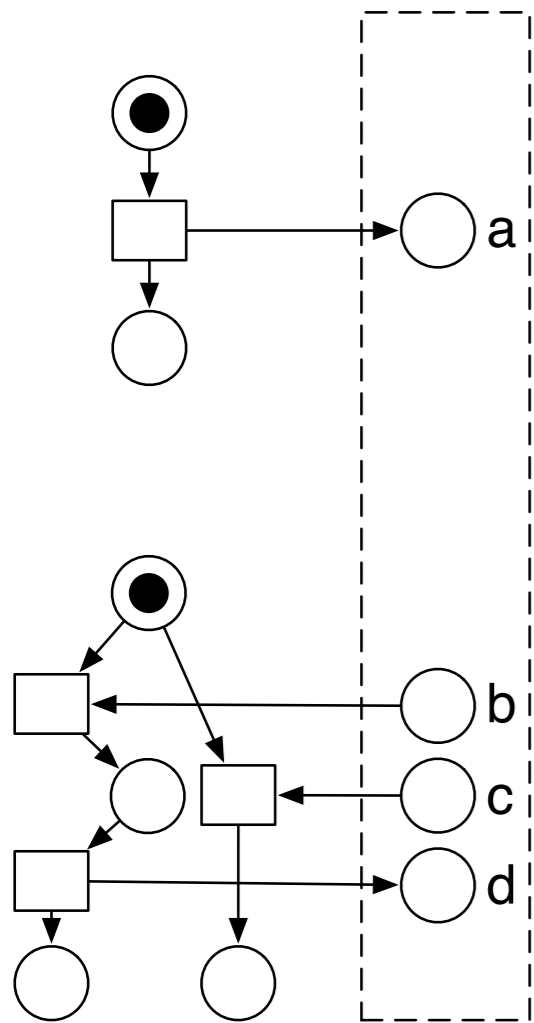
Exercise

Complete with missing arcs the following behavioural interfaces and check their compatibility

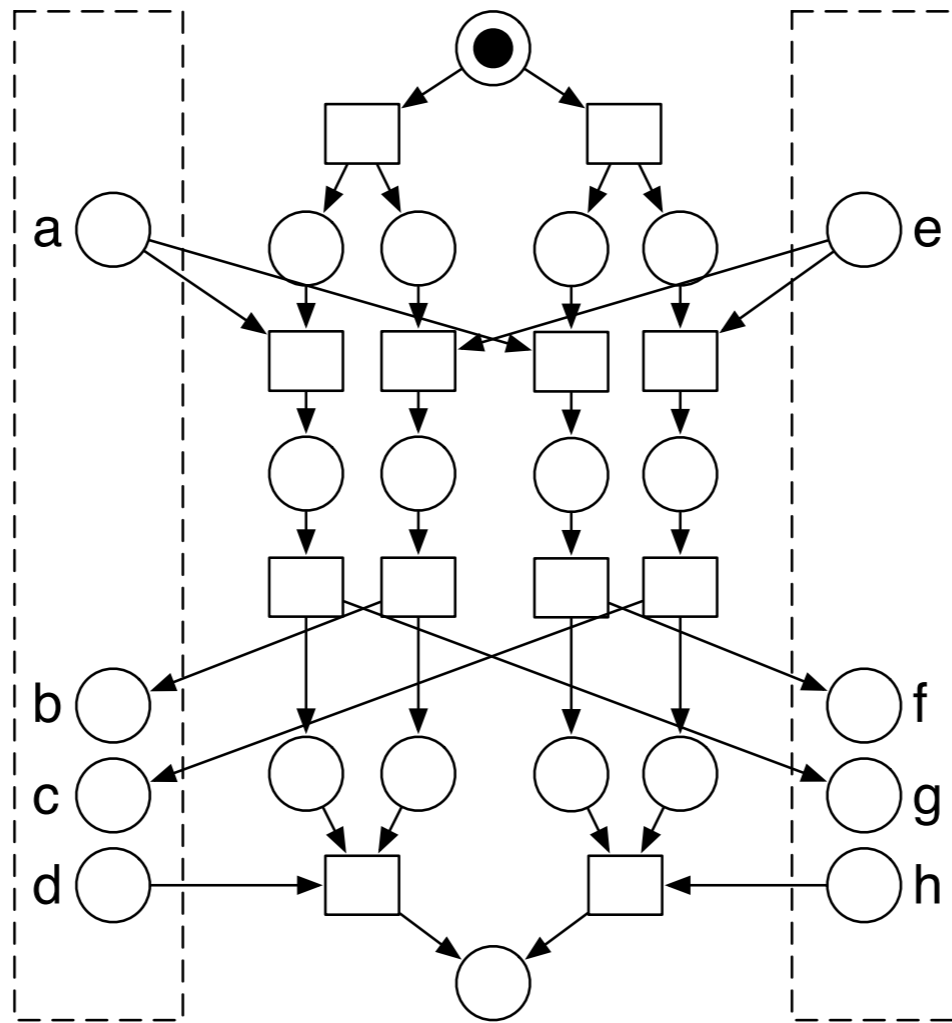


Exercise

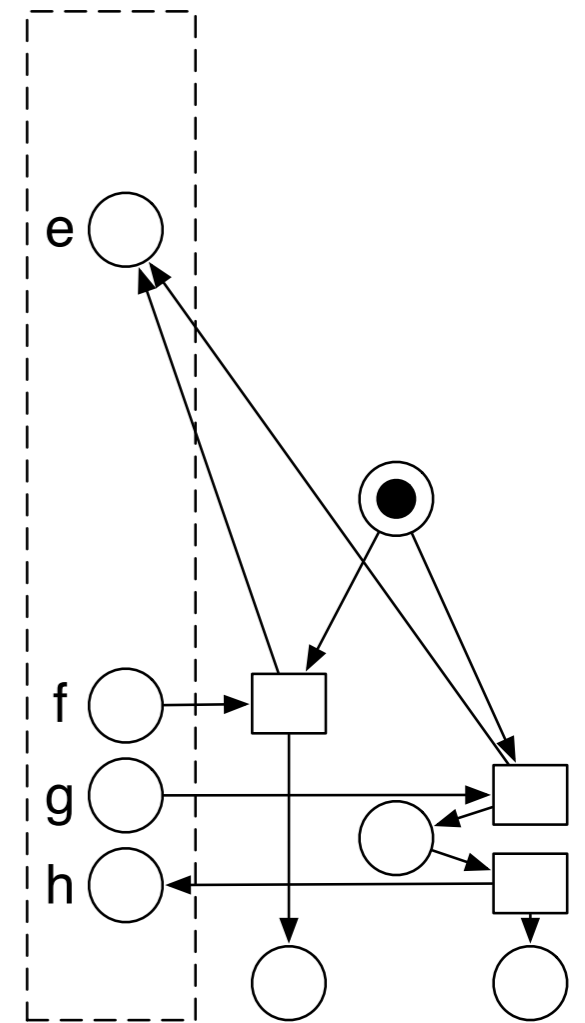
Check compatibility of WF modules below



(a)



(b)



(c)

Weak soundness

Problem

When checking behavioural compatibility
the soundness of the overall net
is a too restrictive requirement

Workflow modules are designed separately
possibly reused in several systems
it is unlikely that every functionality they offer is
involved in each system

Problem

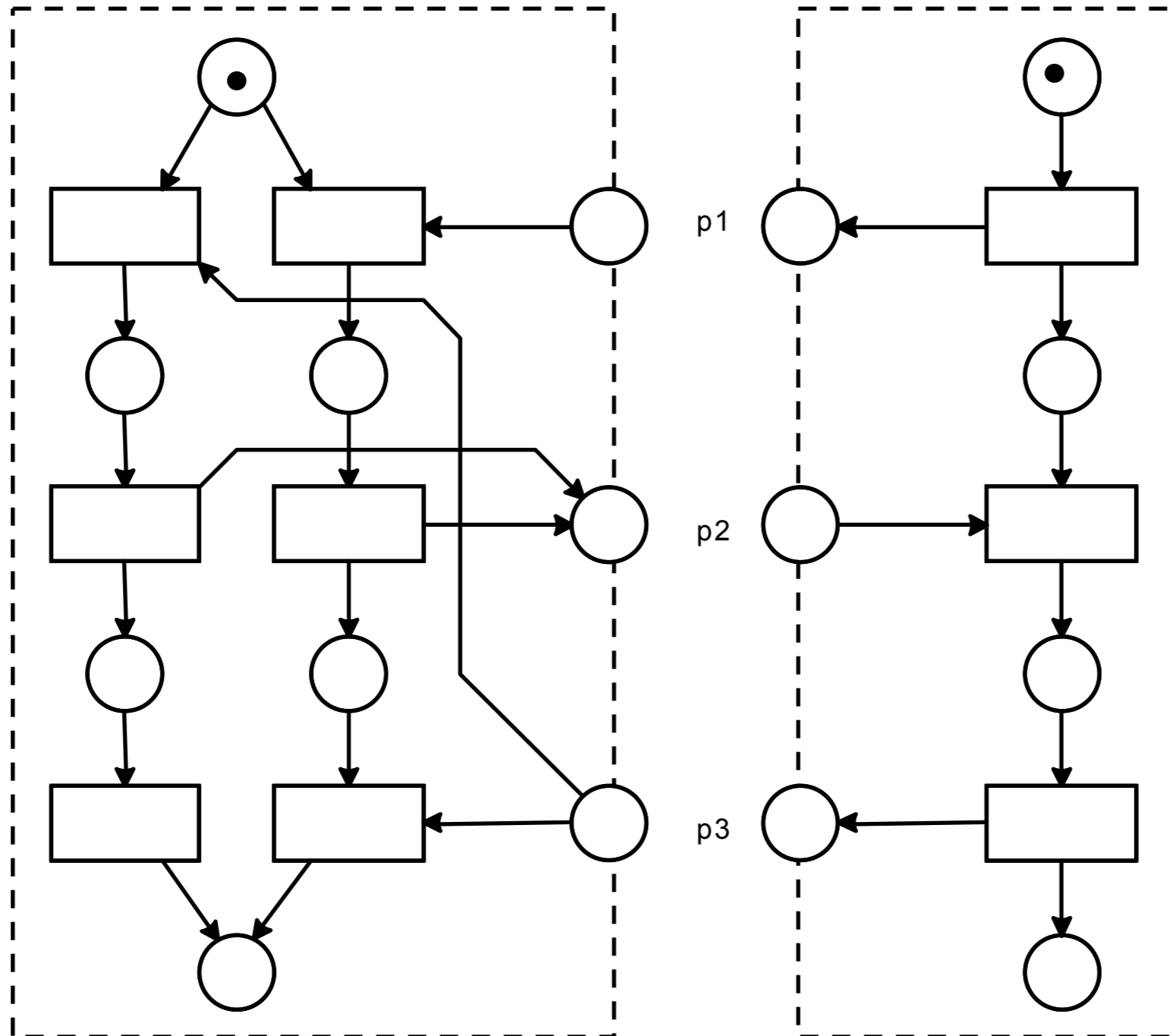
Definition: A workflow net is **weak sound** if it satisfies “option to complete” and “proper completion”

(dead tasks are allowed)

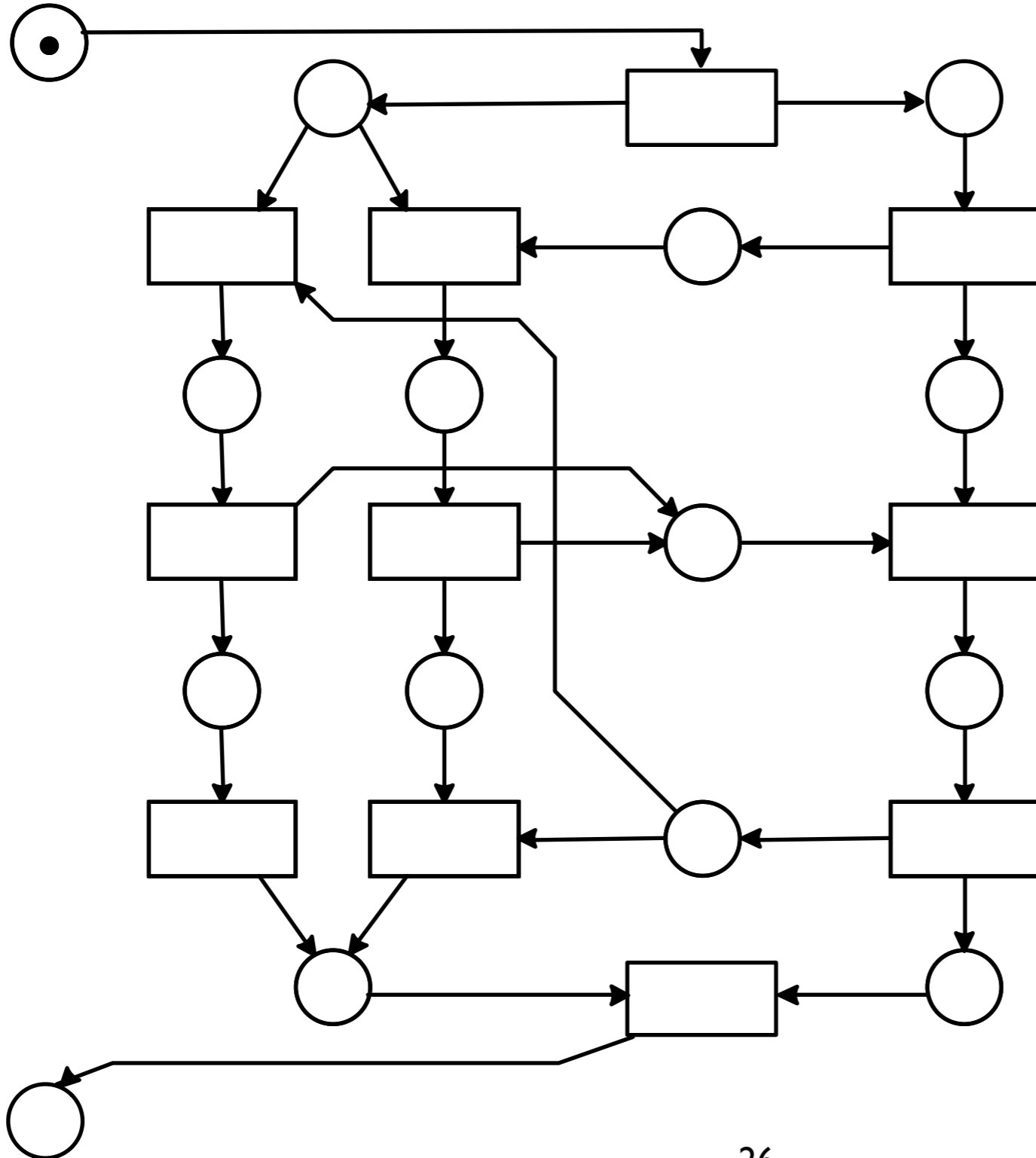
Weak soundness can be checked on the RG

It guarantees deadlock freedom and proper termination of all modules

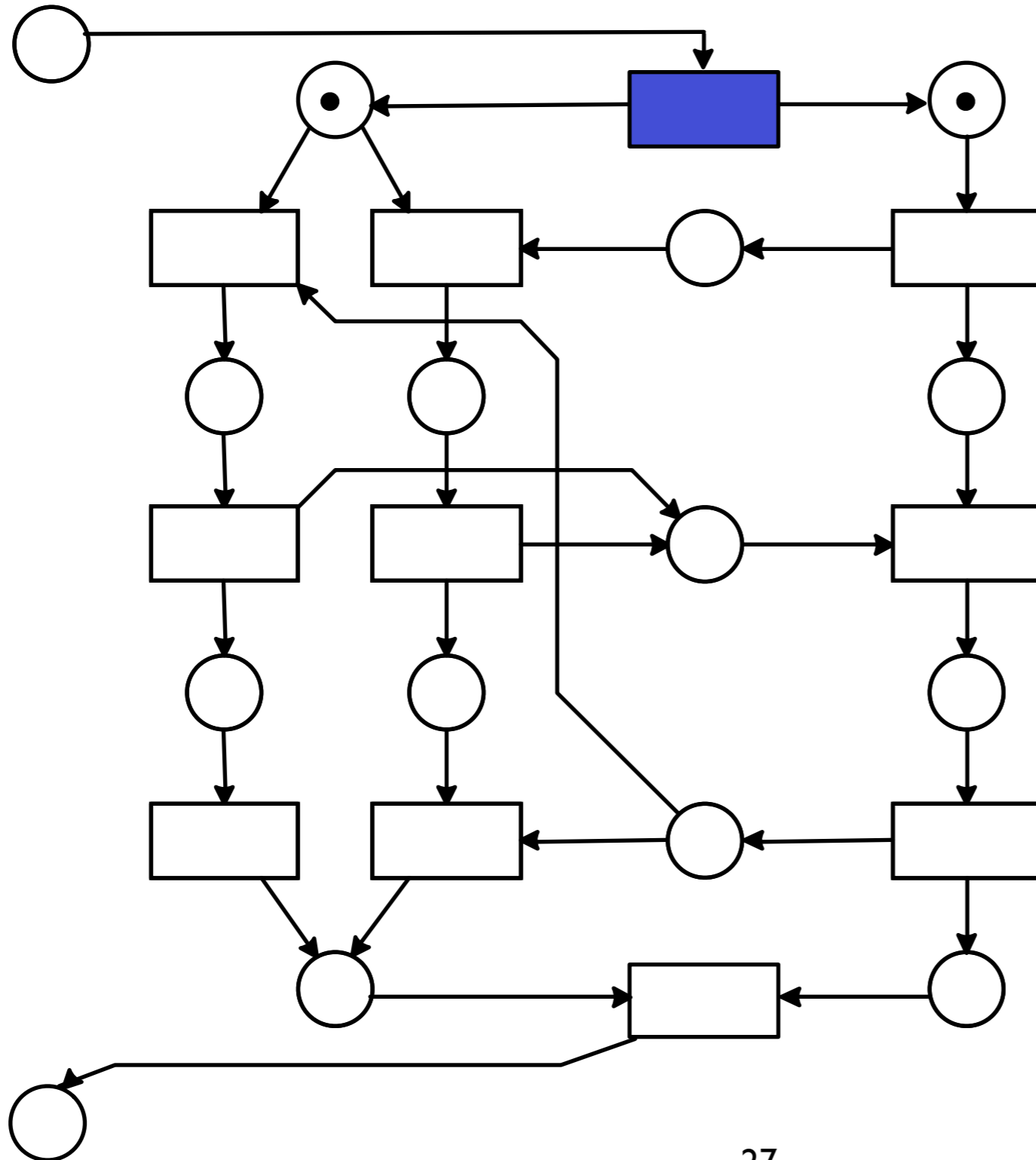
Sound + Sound = ?



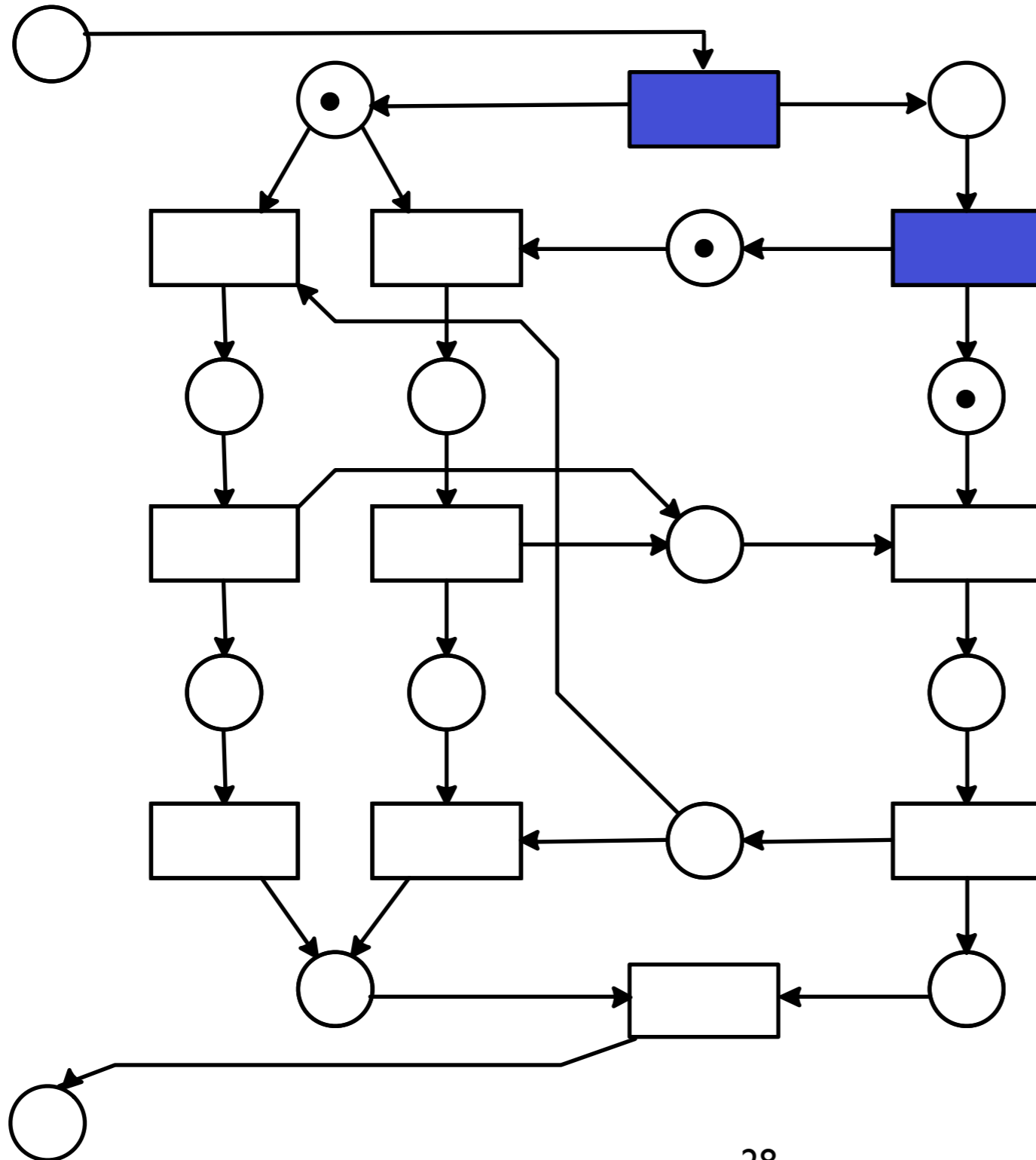
Sound + Sound = not sound



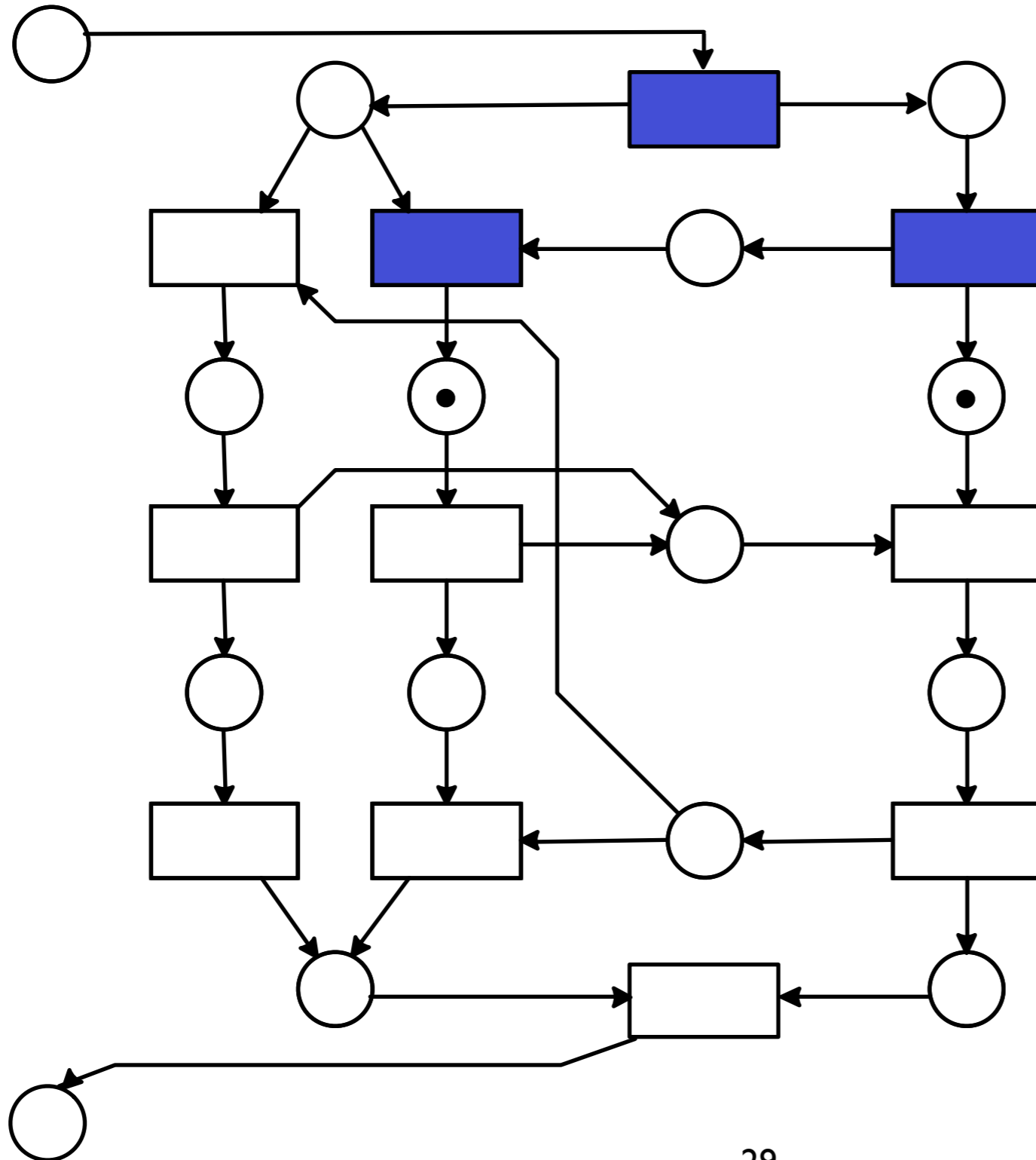
Sound + Sound = not sound



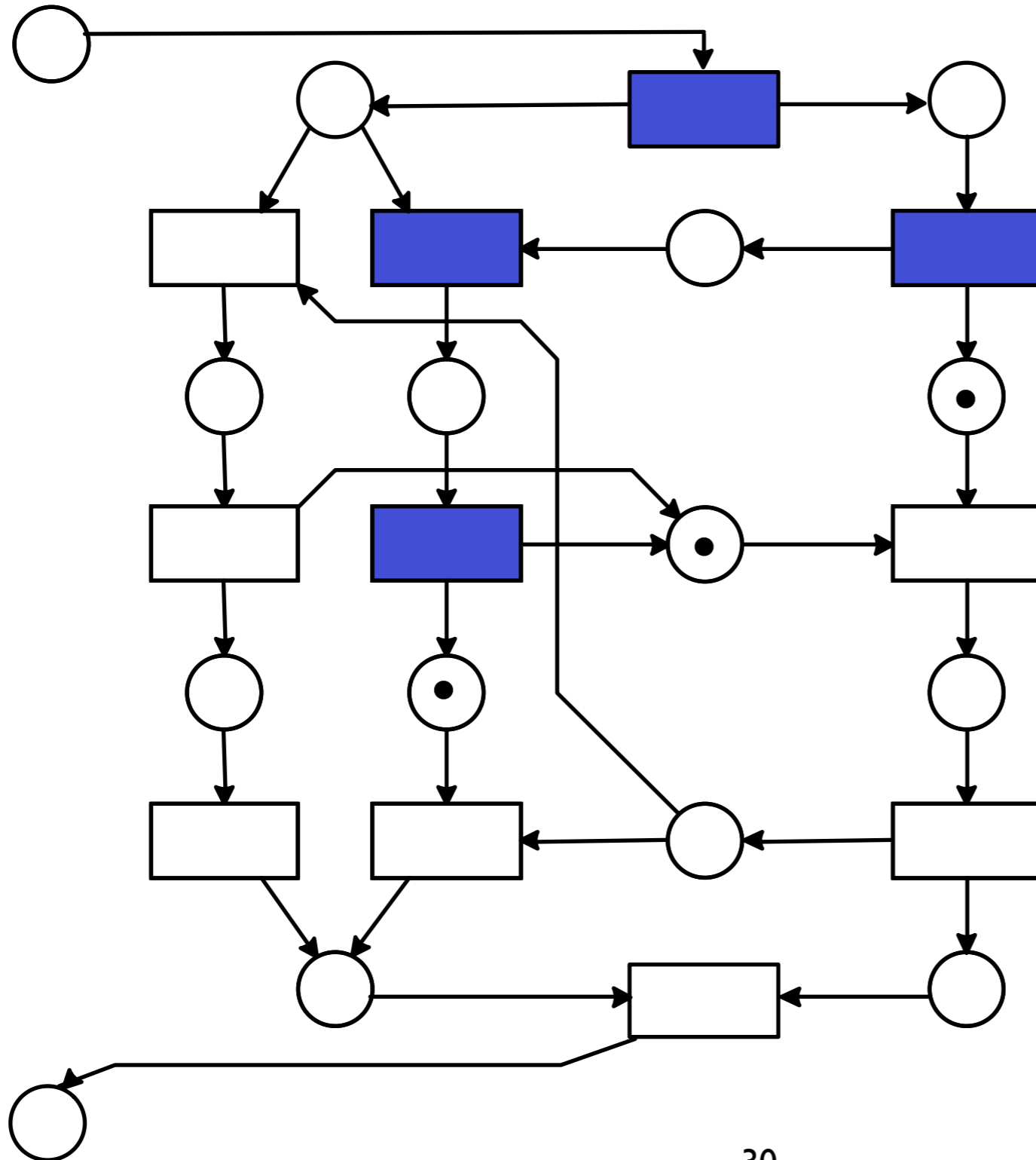
Sound + Sound = not sound



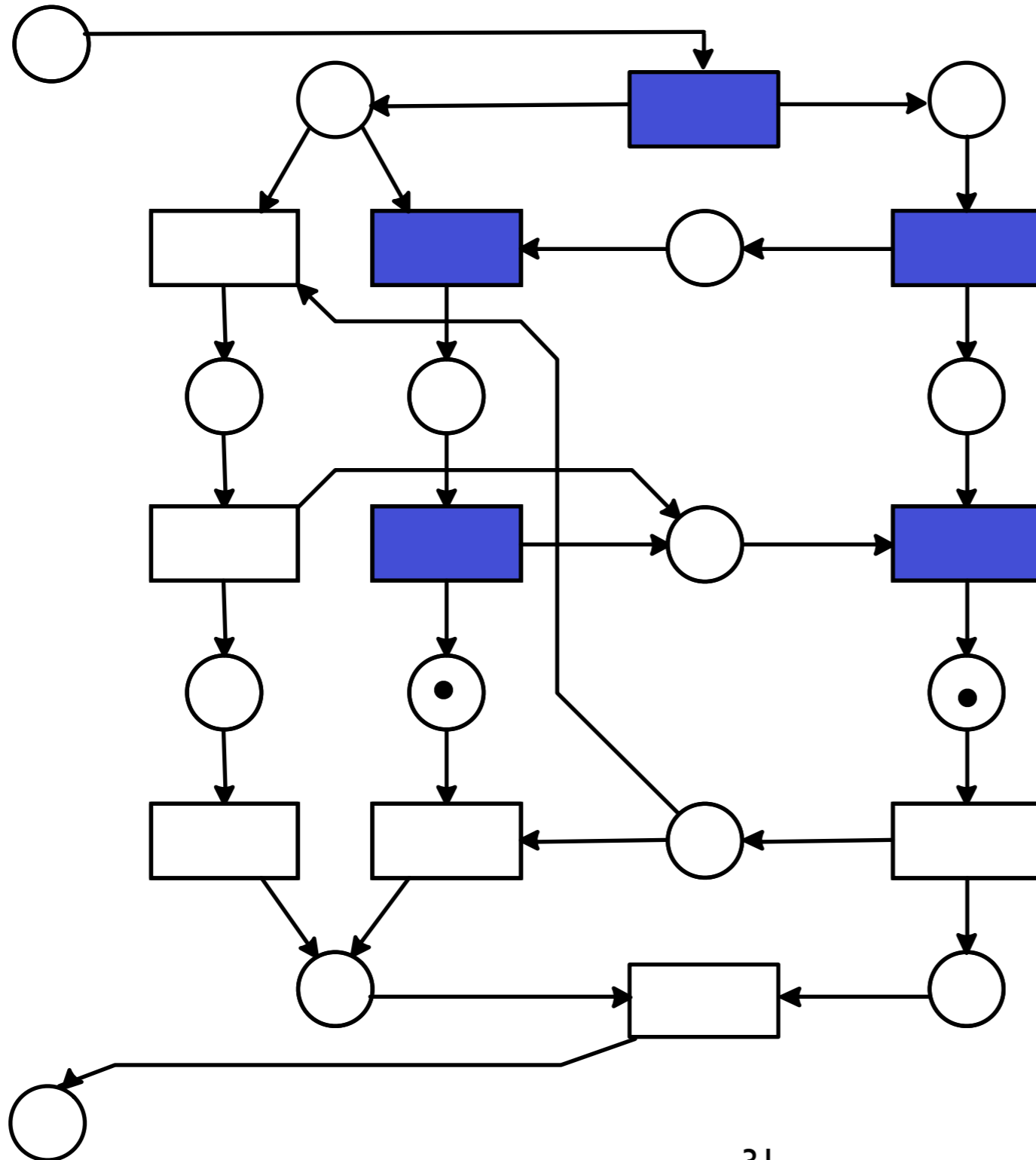
Sound + Sound = not sound



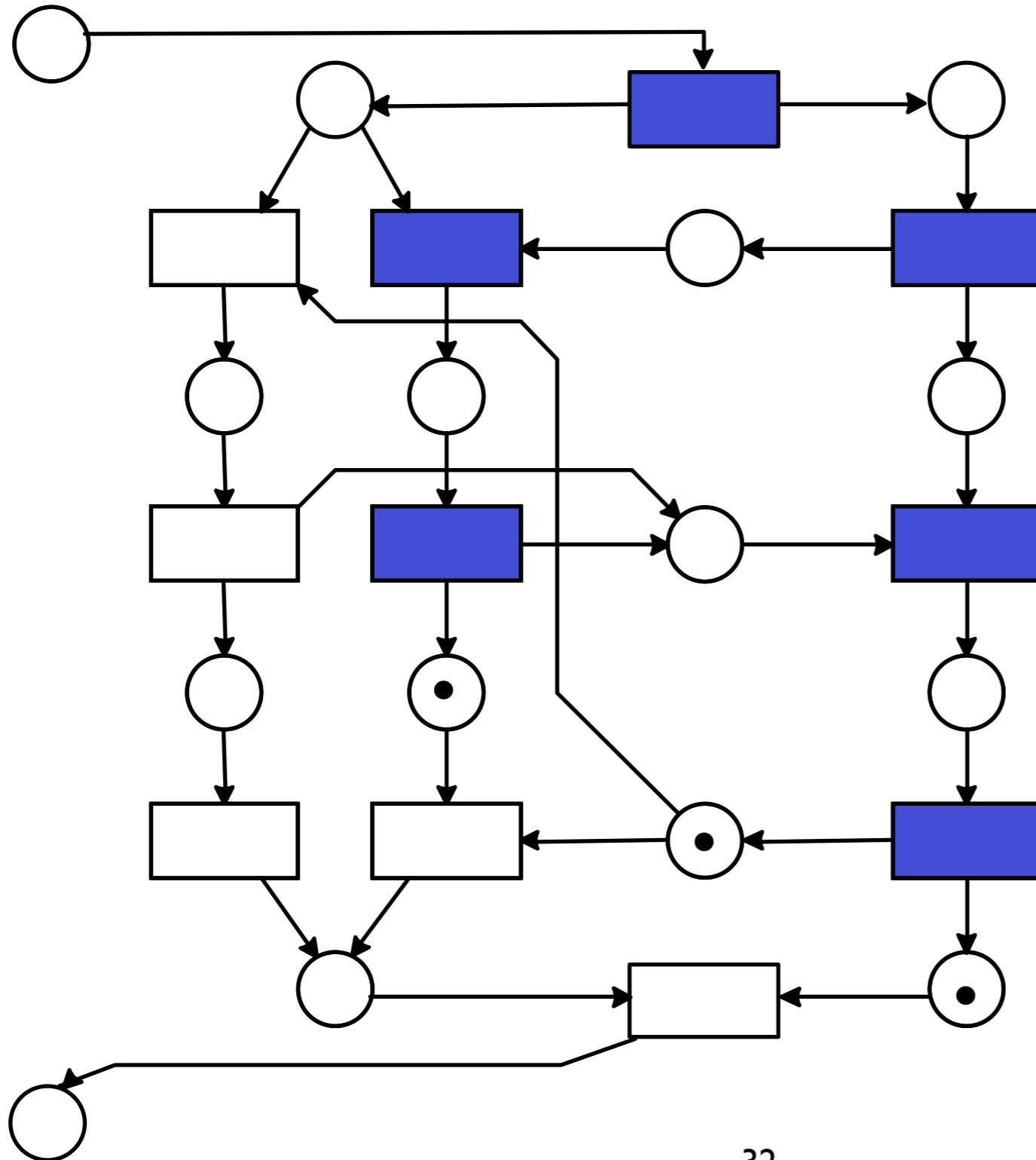
Sound + Sound = not sound



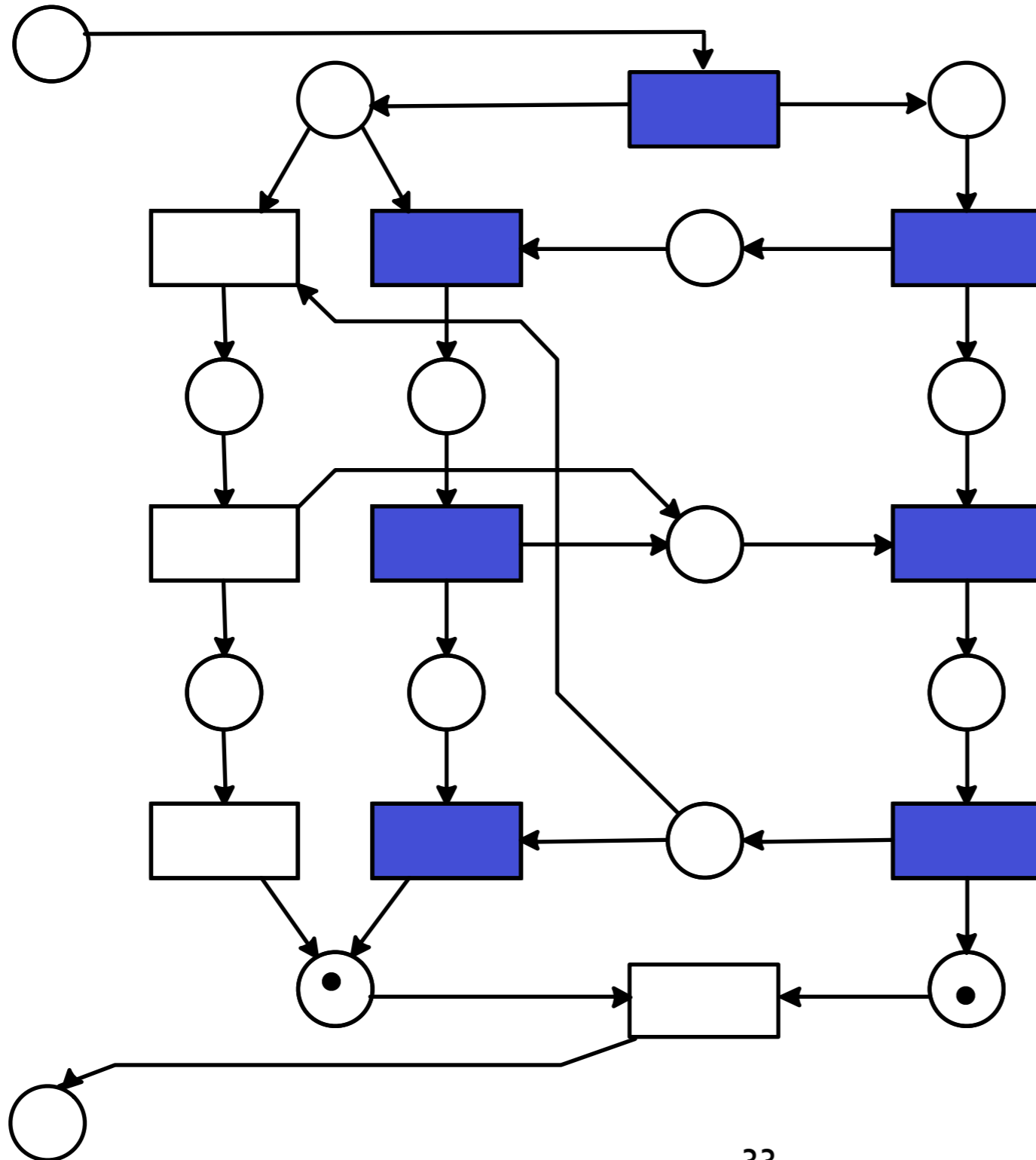
Sound + Sound = not sound



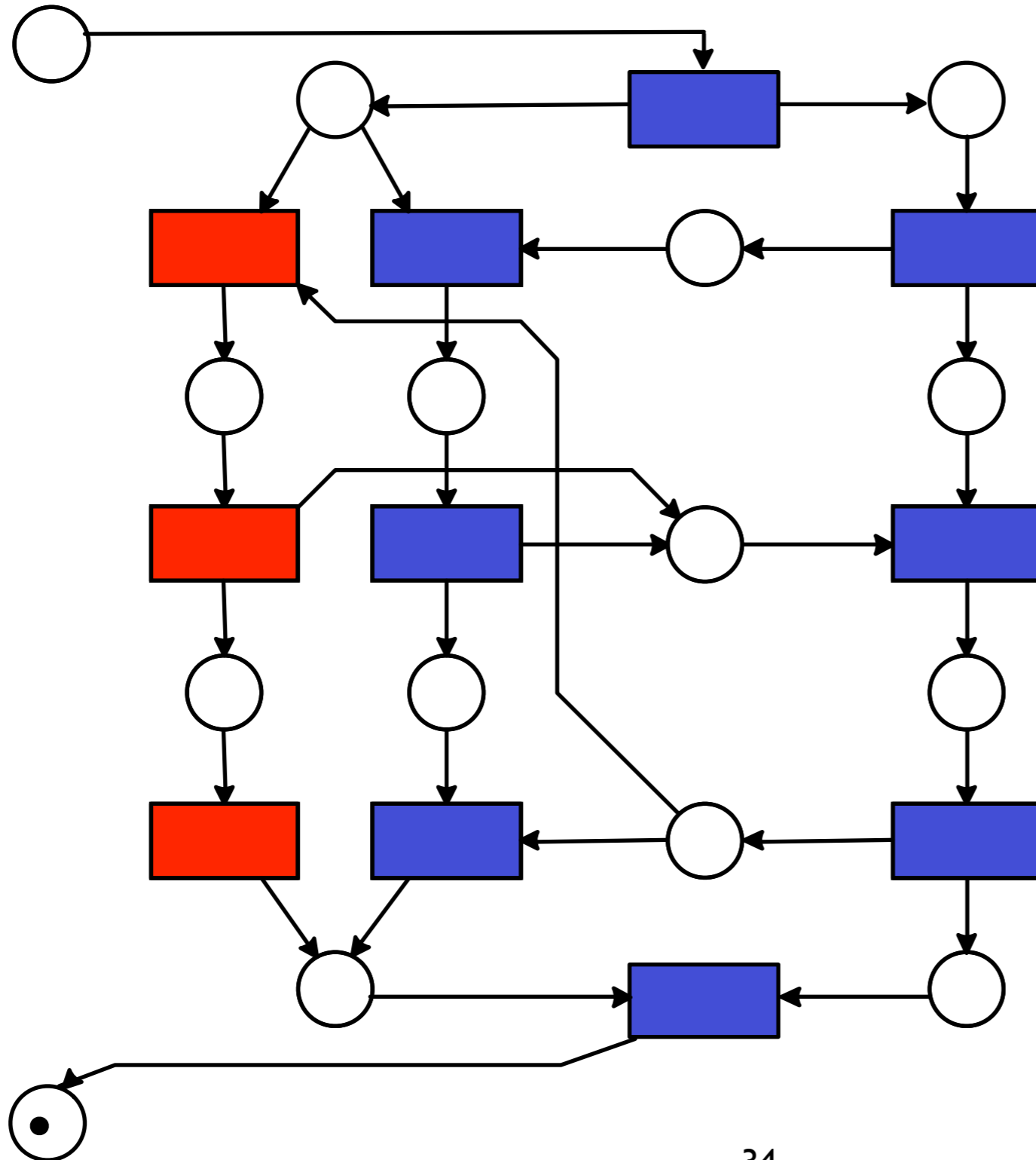
Sound + Sound = not sound



Sound + Sound = not sound

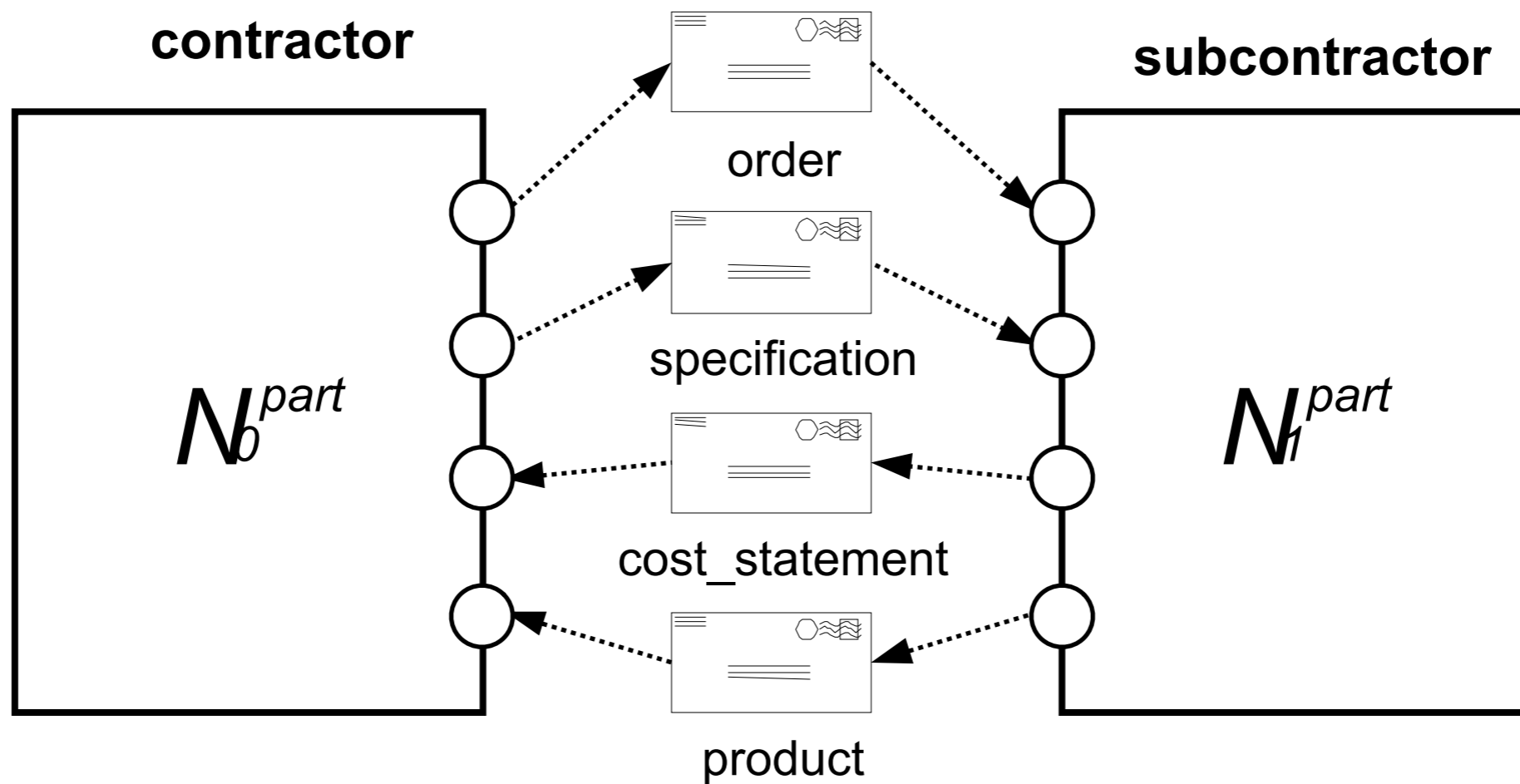


Sound + Sound = not sound

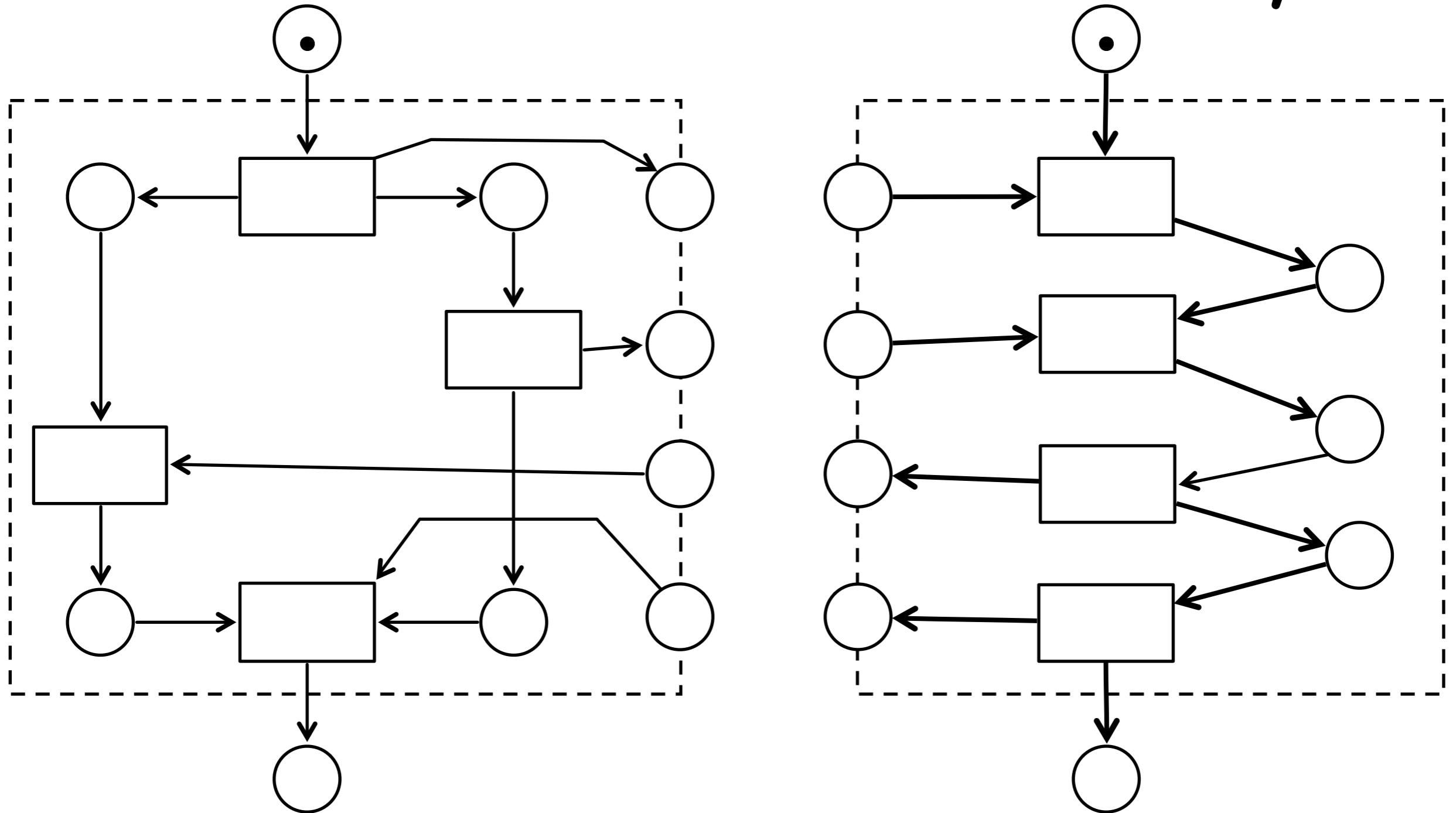


Weak
Sound!

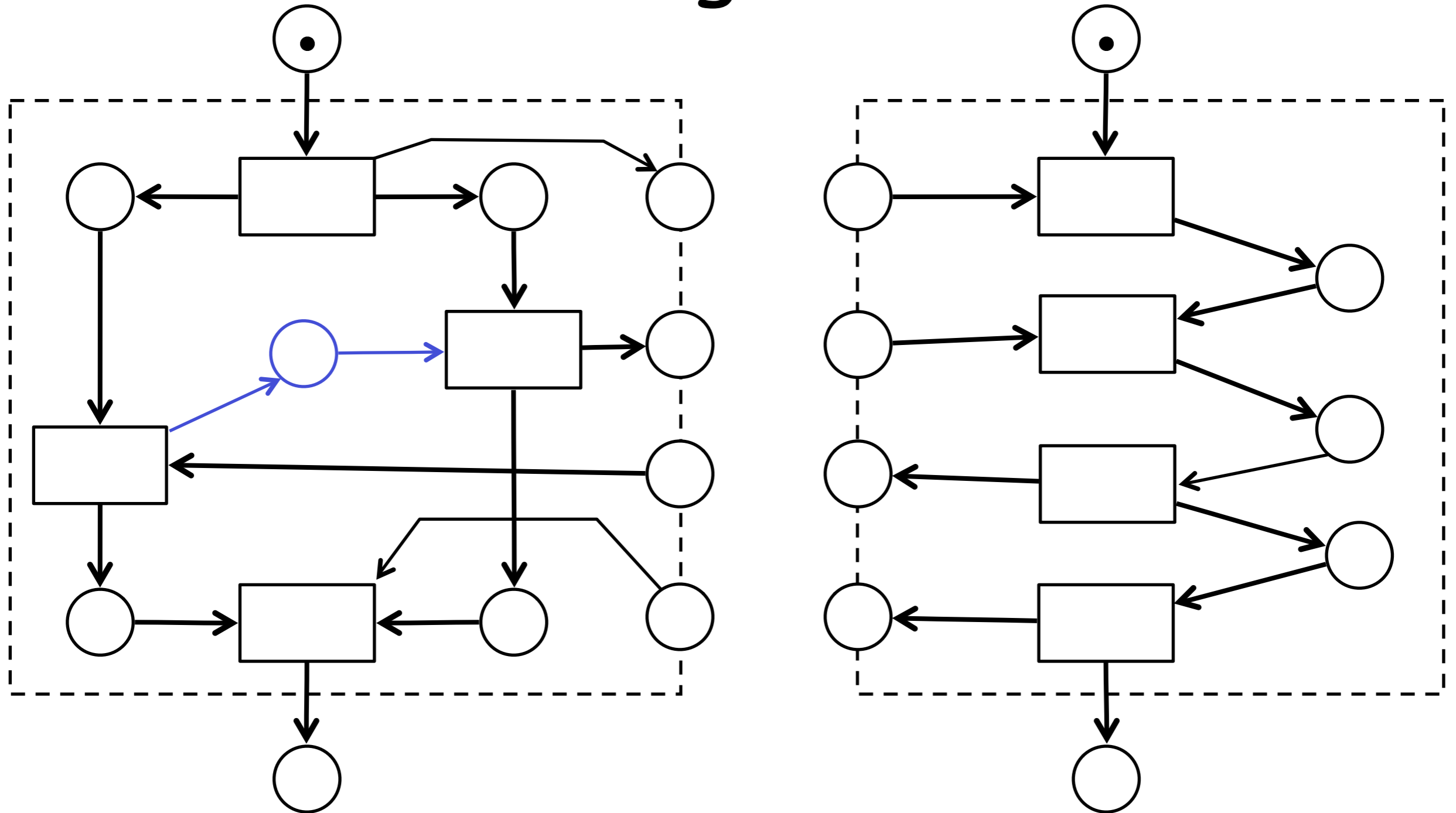
Exercise: Preliminaries



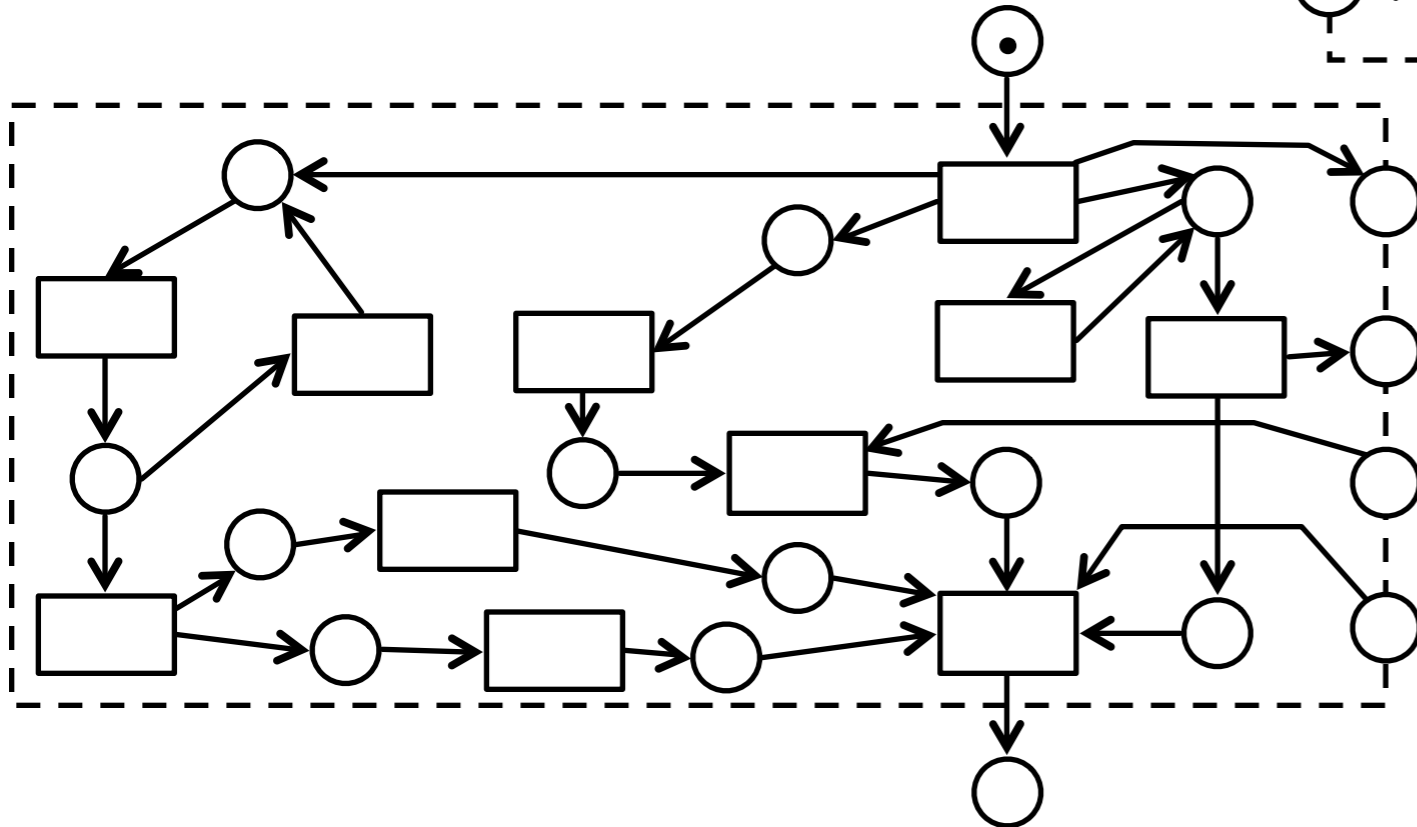
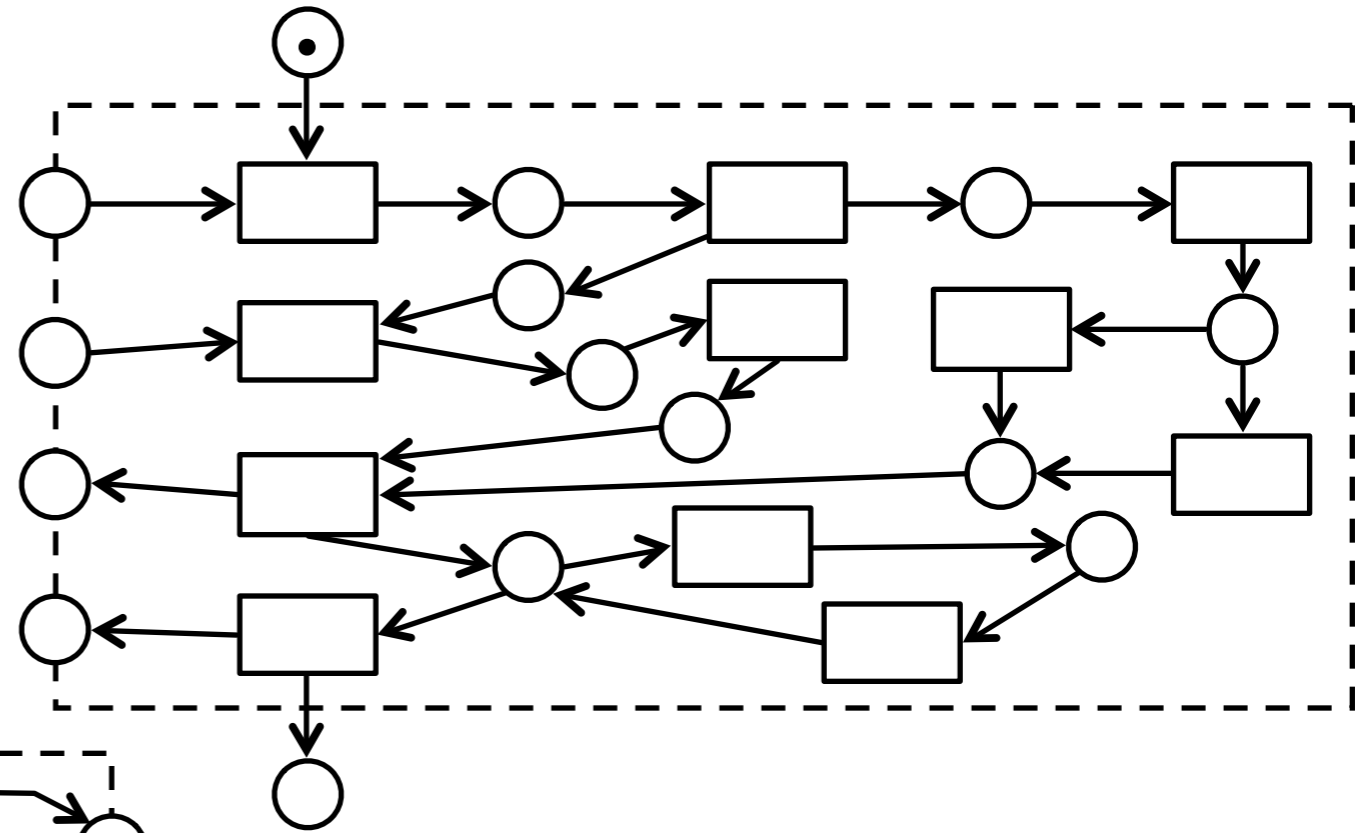
Exercise: Check Weak Soundness of The Assembly



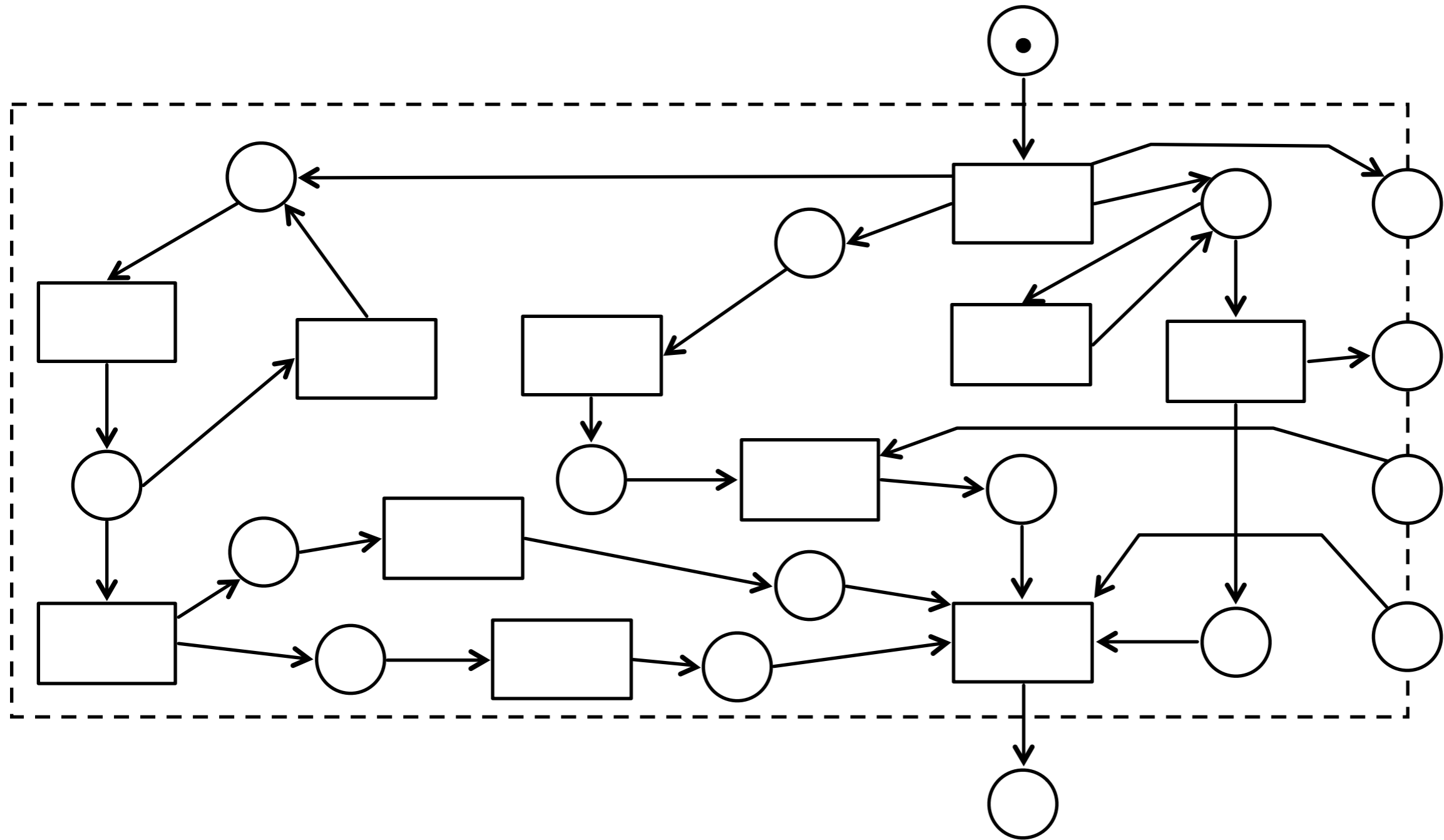
Exercise: Check Again After Refactoring Contractor



Exercise: Check Again After Refactoring Both



(Contractor zoom-in)



(Subcontractor zoom-in)

