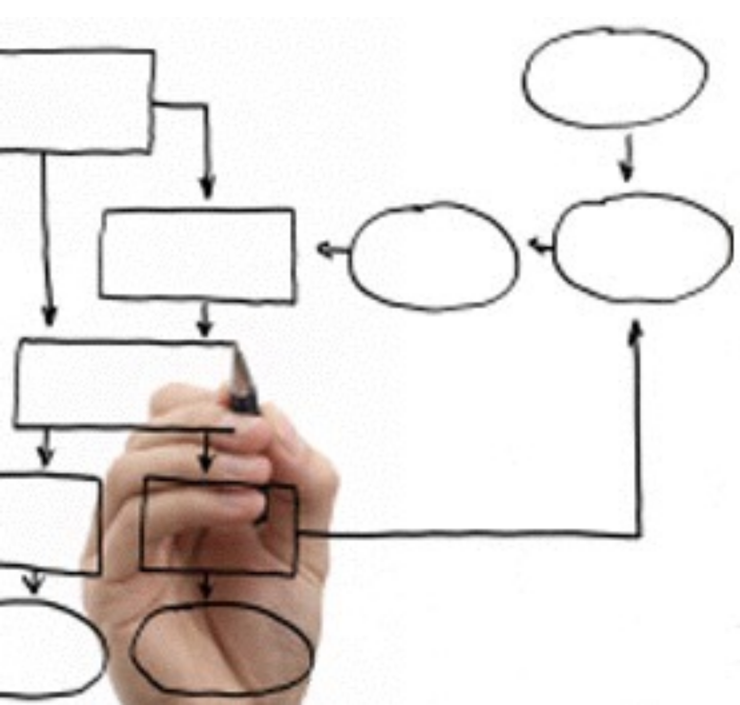
Methods for the specification and verification of business processes

MPB (6 cfu, 295AA)

Roberto Bruni
http://www.di.unipi.it/~bruni



20 - Workflow modules

Object

We study Workflow modules to model interaction between workflows

Ch.6 of Business Process Management: Concepts, Languages, Architectures

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?

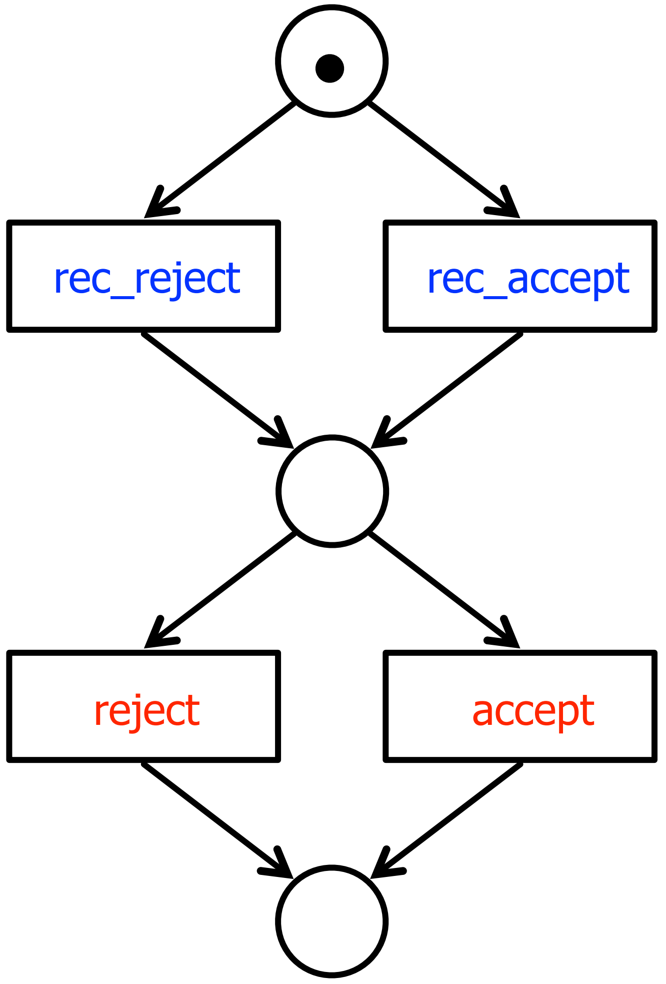
Implicit interaction

Separately developed workflow

Some activities can input messages

Some activities can output messages

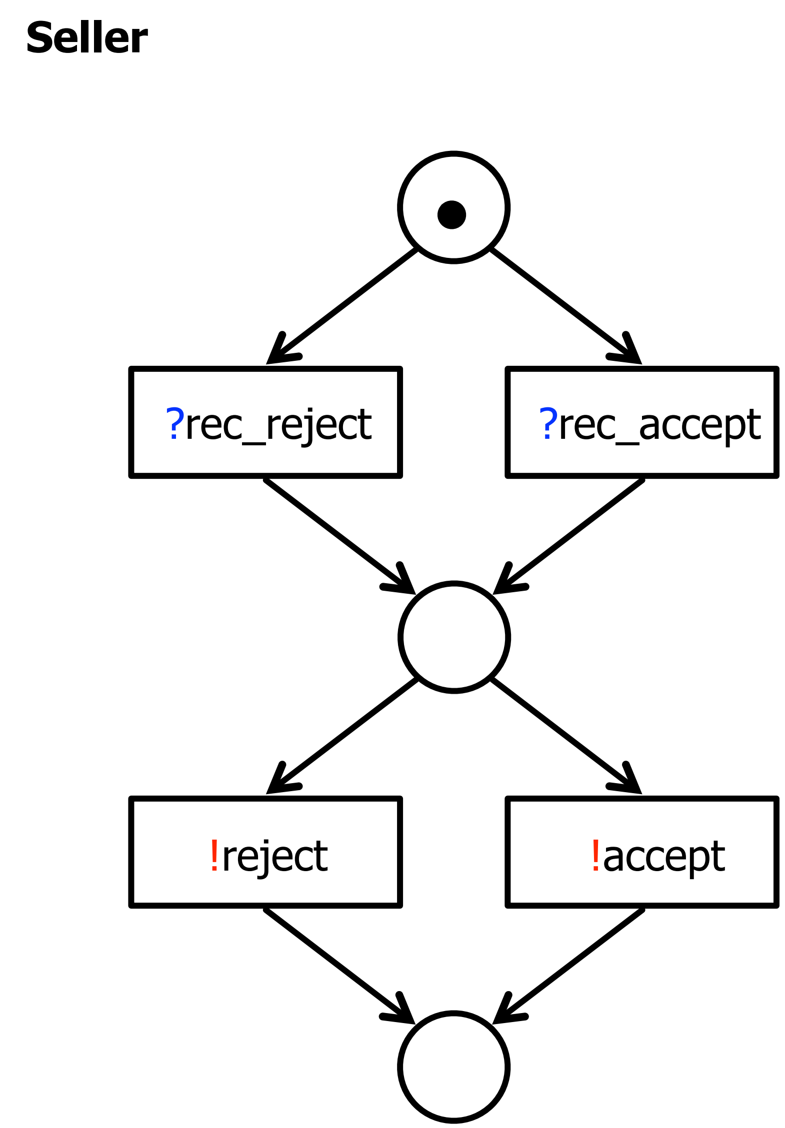
Seller



Implicit interaction

Seller can receive (symbol ?) recommendations

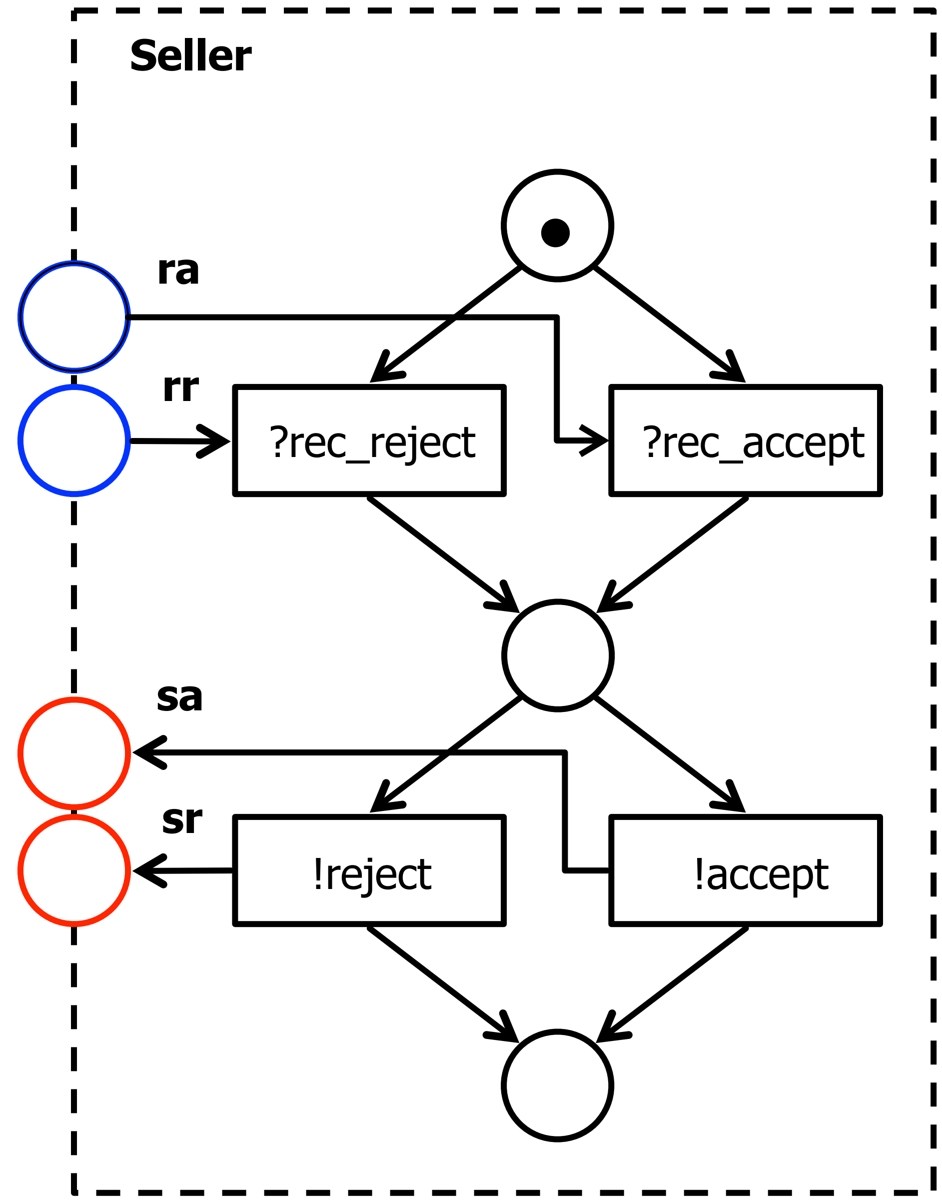
Seller can send (symbol !) decisions



Interface

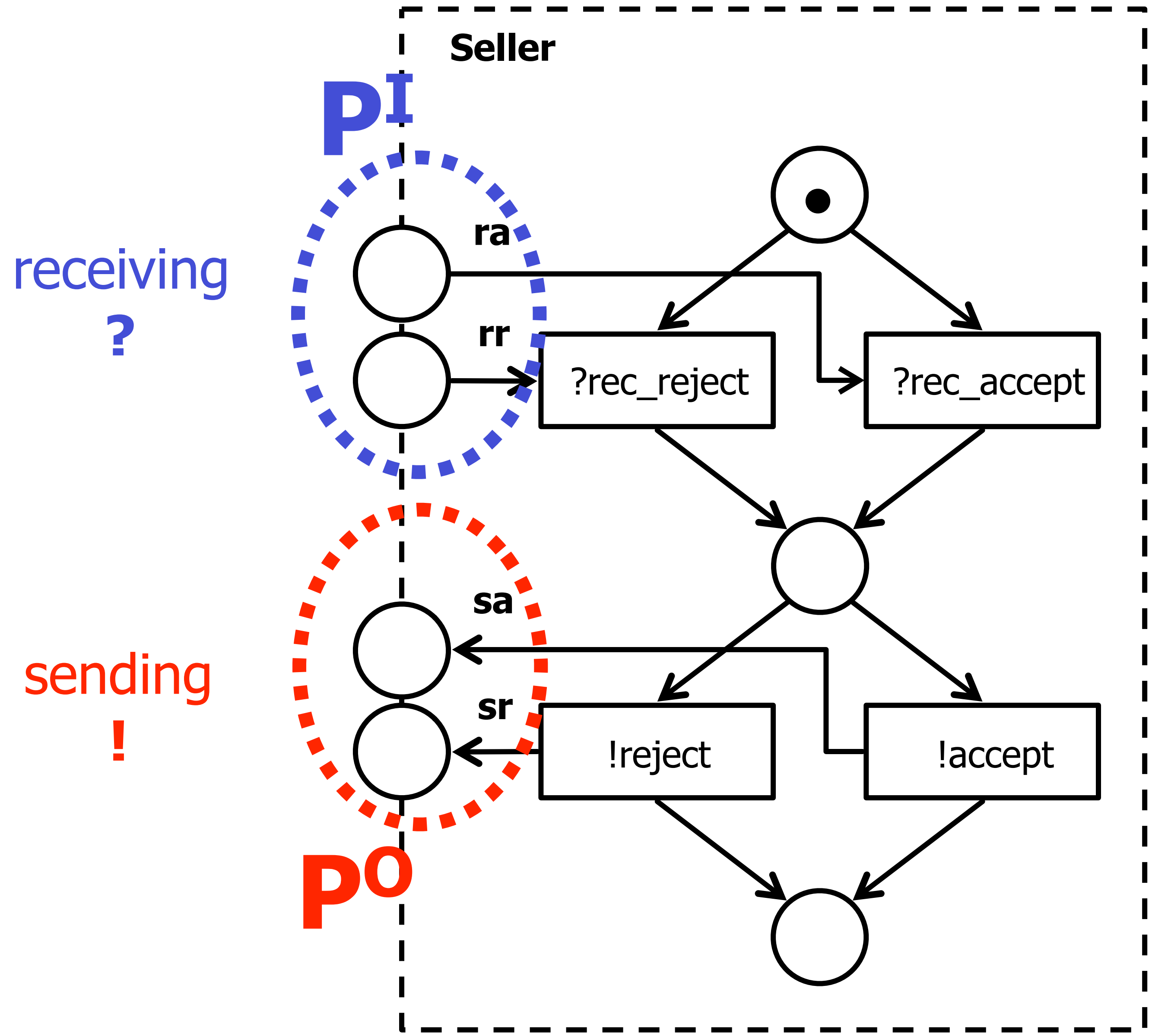
Seller has an interface for interaction

It consists of some input places and some output places



M. Weske: Business Process Management,

Interface


receiving
?
sending
!
P^I
P^O
Seller
ra
rr
sa
sr
?rec_reject
?rec_accept
!reject
!accept

Problem

Assume the original workflow net has been validated:

it is a sound (and maybe safe) workflow net

When we add the (places in 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 $\mathbf{P^I}$ of incoming places
plus a set of incoming arcs $\mathbf{F^I \subseteq (P^I \times T)}$

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

such that each transition has
at most one connection to places in the interface

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 module who can receive it,
and
for every message that can be received
there is a module who can send it

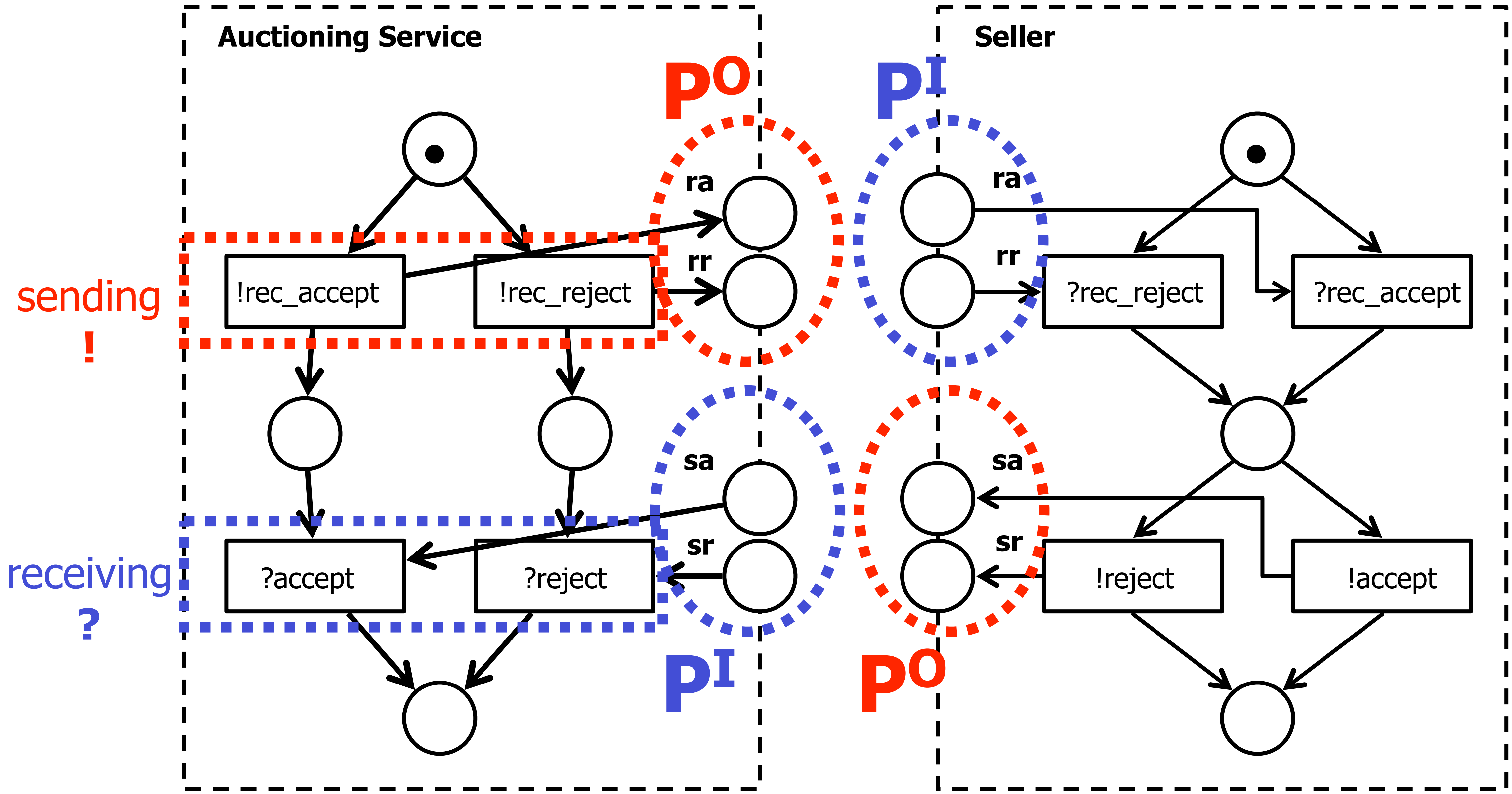
(formats of message data are assumed to match)

Weak structural compatibility

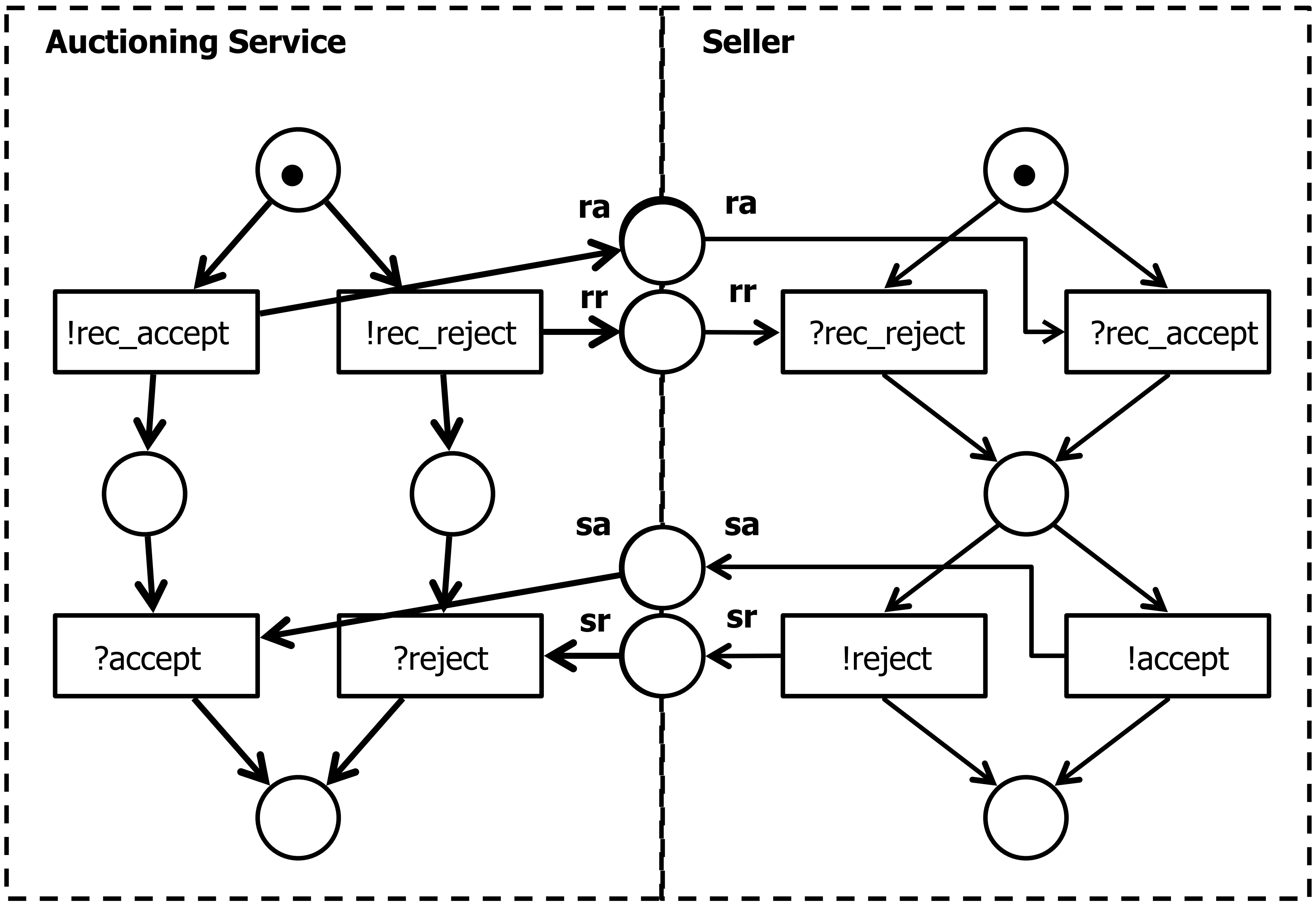
A set of workflow modules is called
weakly structural compatible
if
all messages sent by modules
can be received by other modules

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

Interaction



Interaction



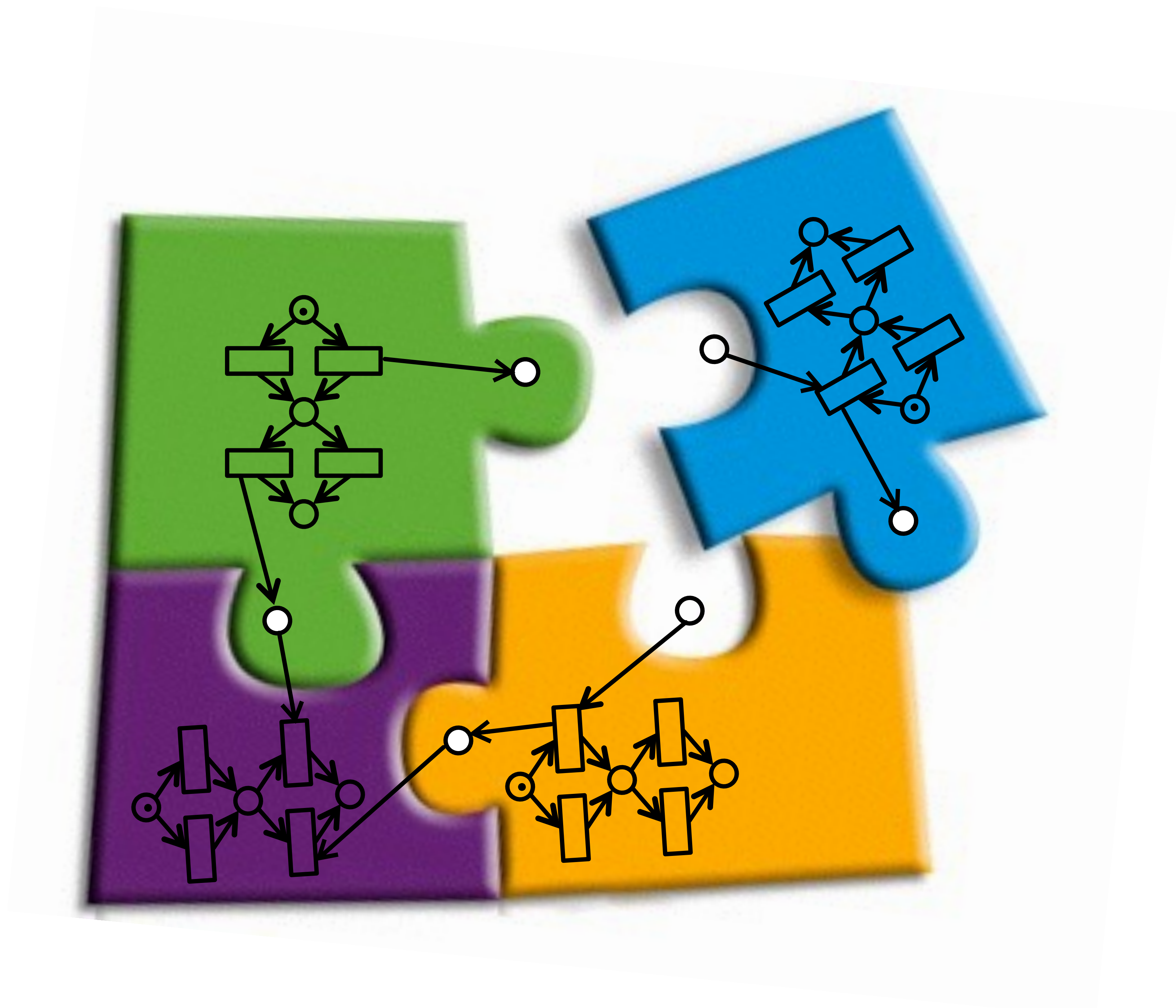
Problem

We have added places and arcs to single nets
We have joined places of different nets
We have paired their initial markings

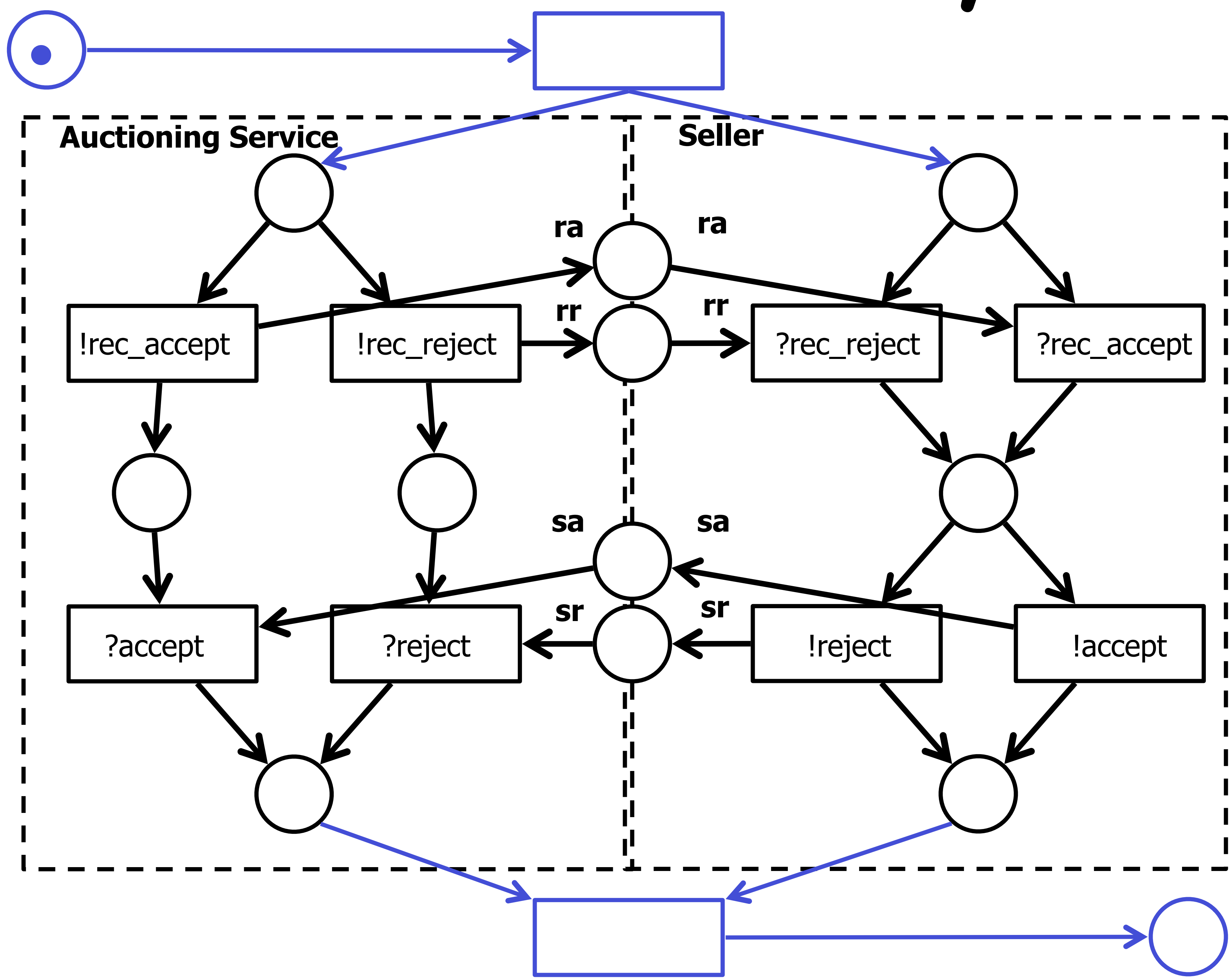
How do we check that the system behaves well?

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 $i_1,\ldots,i_n$, final places $o_1,\ldots,o_n$)

plus an initial place $\mathbf{i}$
and a transition $\mathbf{t_i}$ from $\mathbf{i}$ to $\mathbf{i_1,\ldots,i_n}$

plus a final place $\mathbf{o}$
and a transition $\mathbf{t_o}$ from $\mathbf{o_1,\ldots,o_n}$ to $\mathbf{o}$

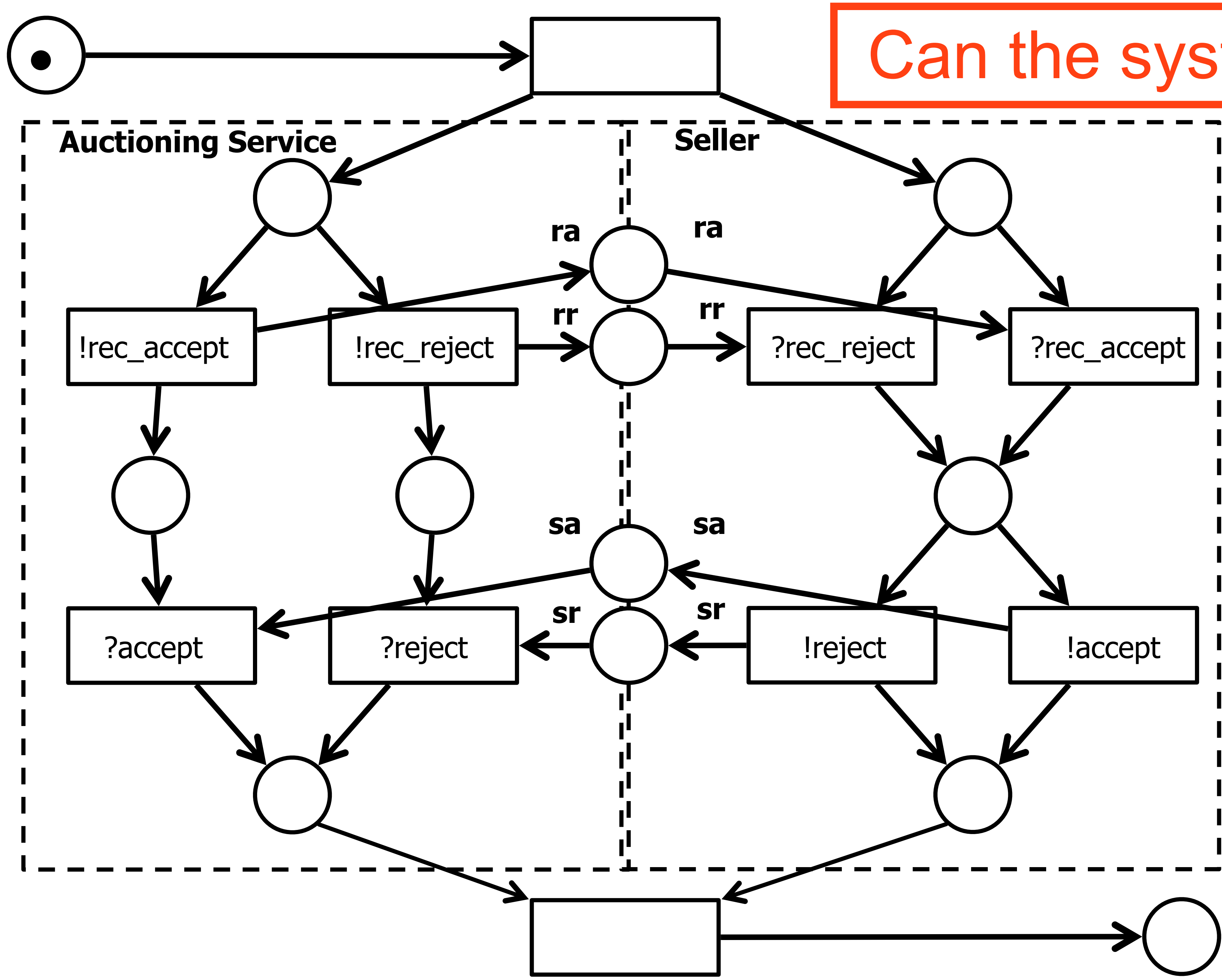
Soundness of workflow systems

A workflow system is just an ordinary workflow net

We can check its **soundness** as usual

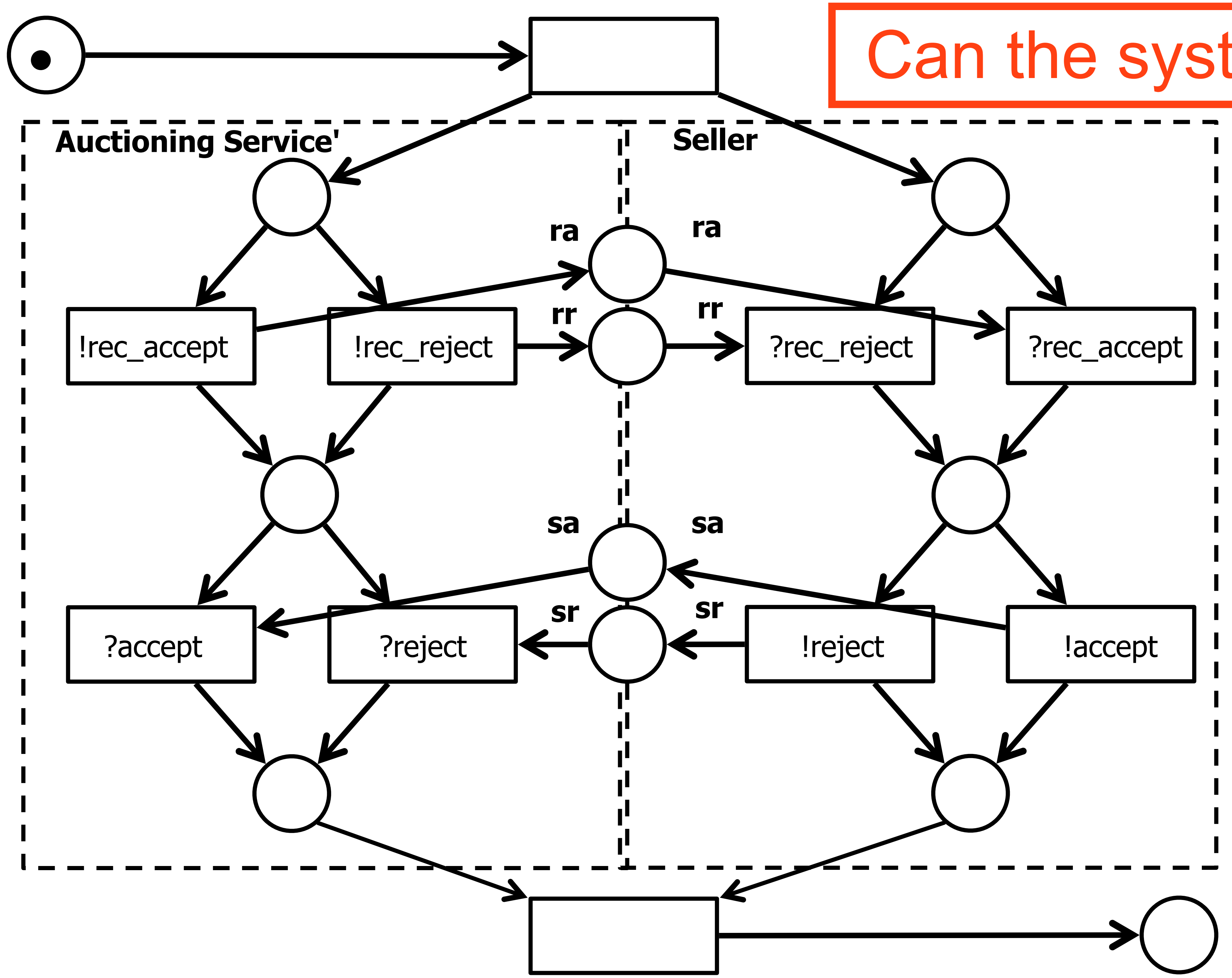
Exercise

Can the system deadlock?



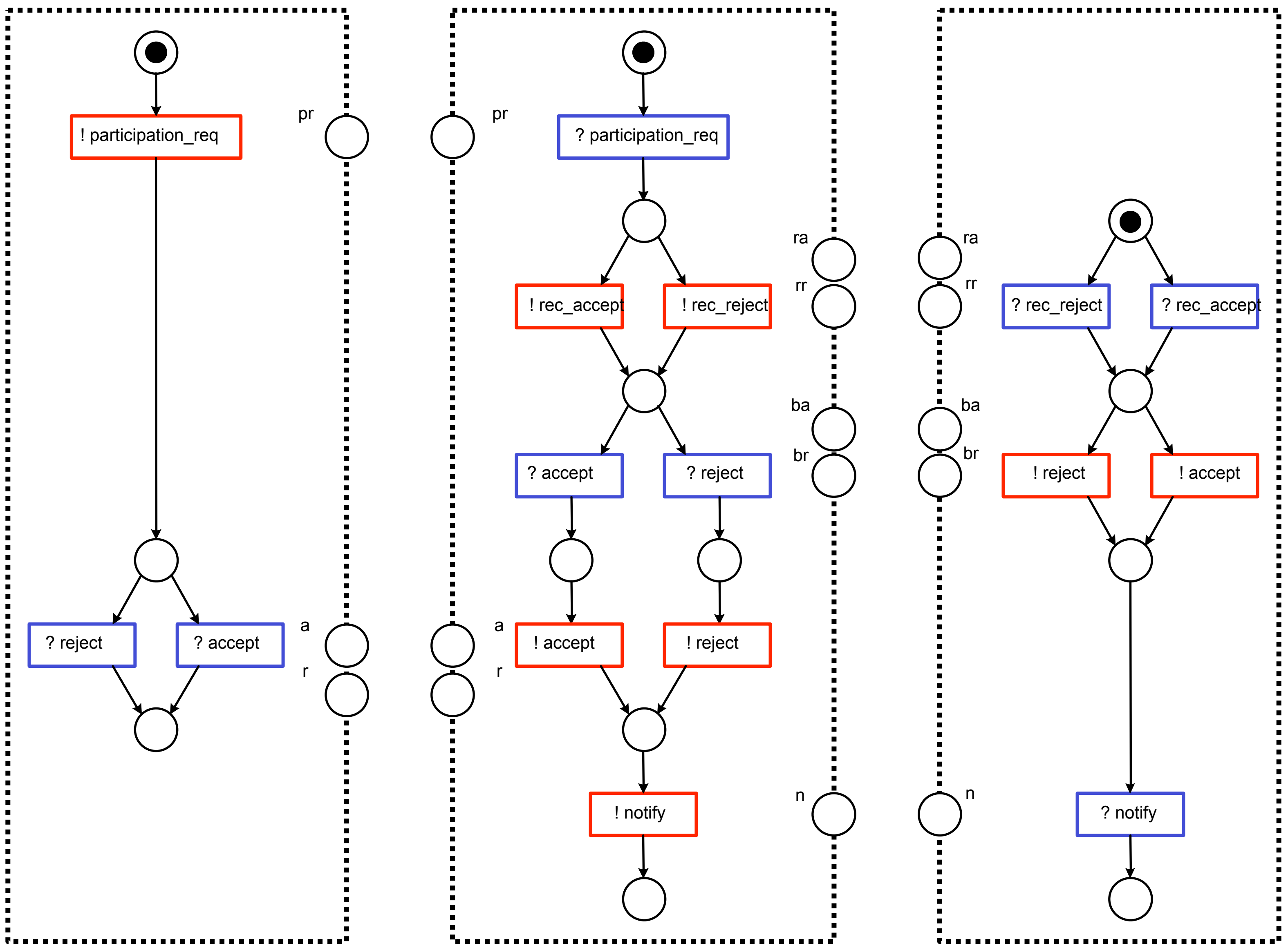
Exercise

Can the system deadlock?



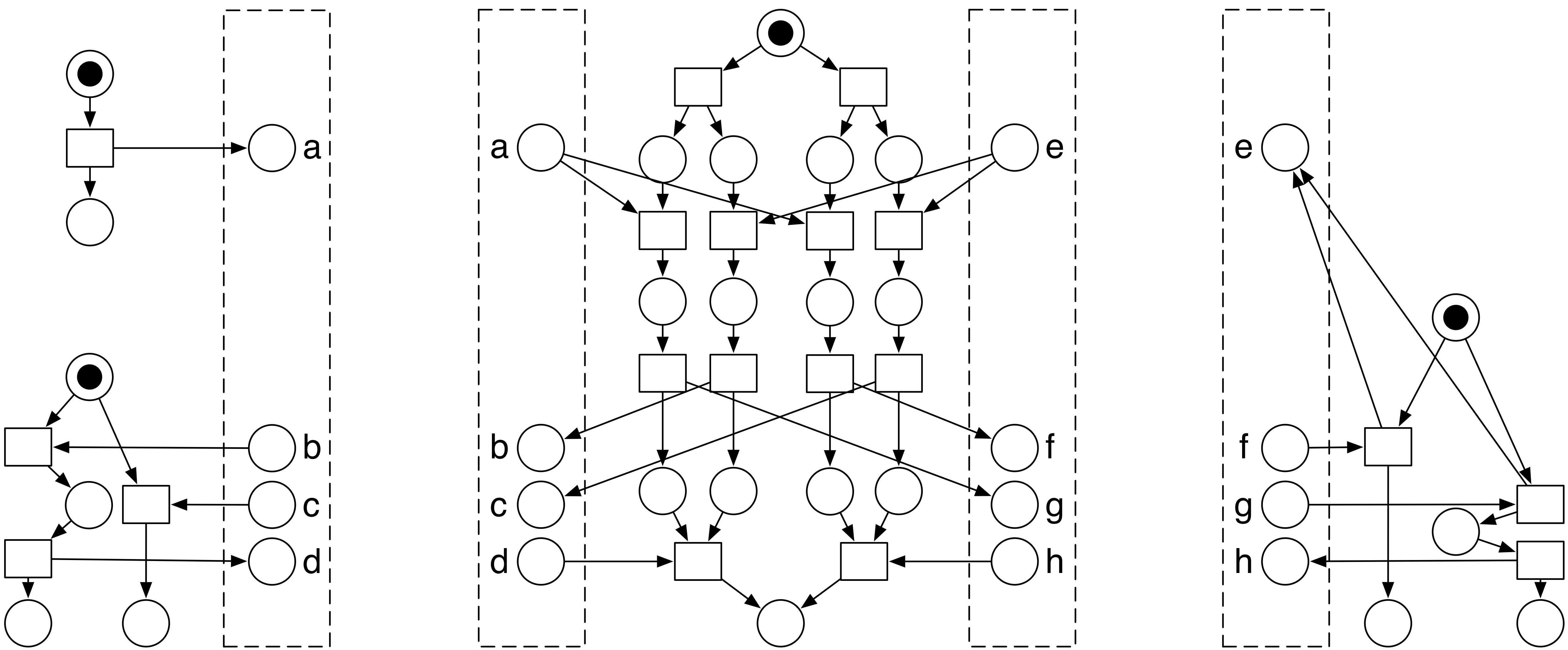
Exercise

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

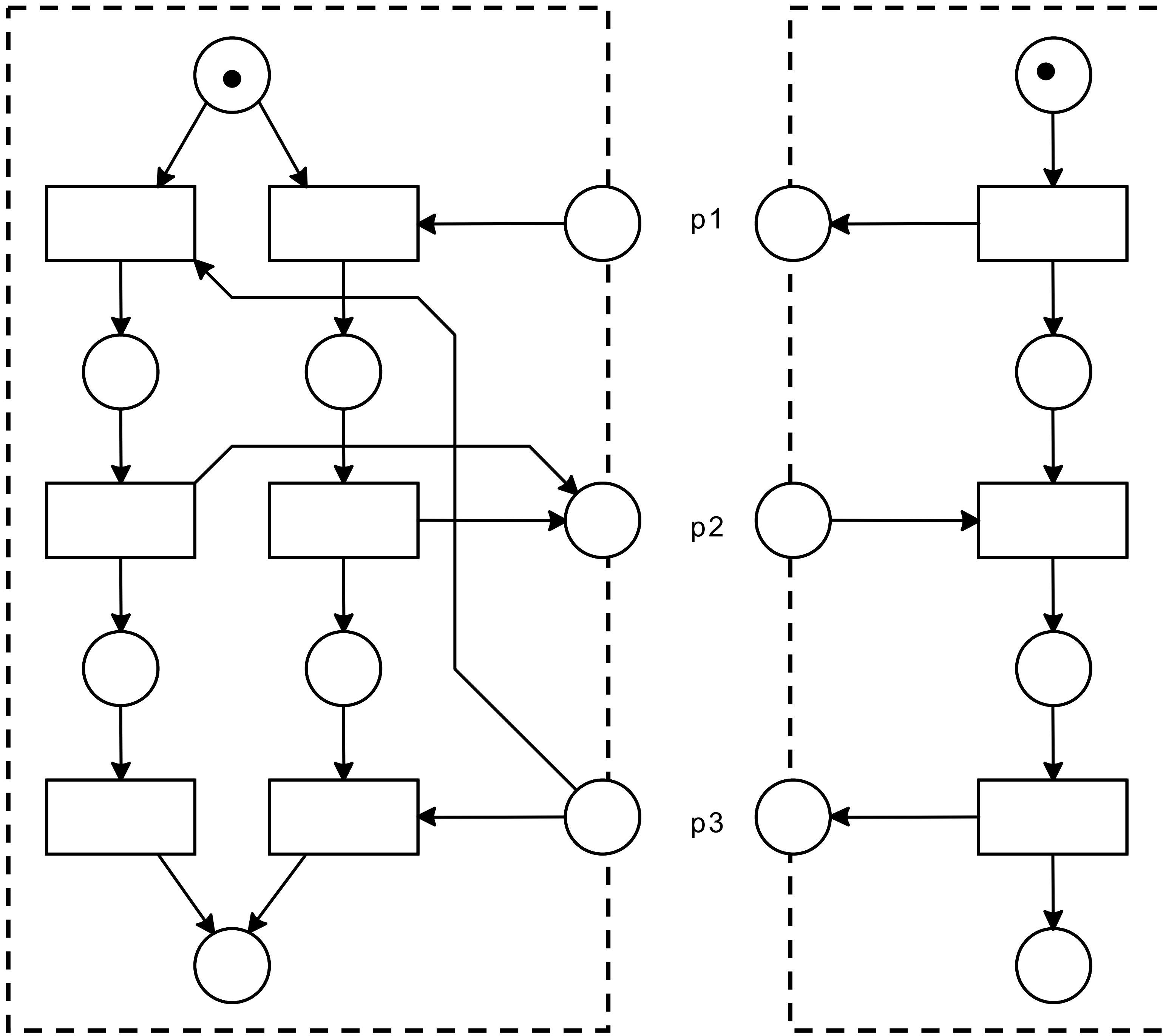
Definition: A workflow net is **weak sound** if it satisfies "option to complete" and "proper completion"

(dead tasks are allowed)

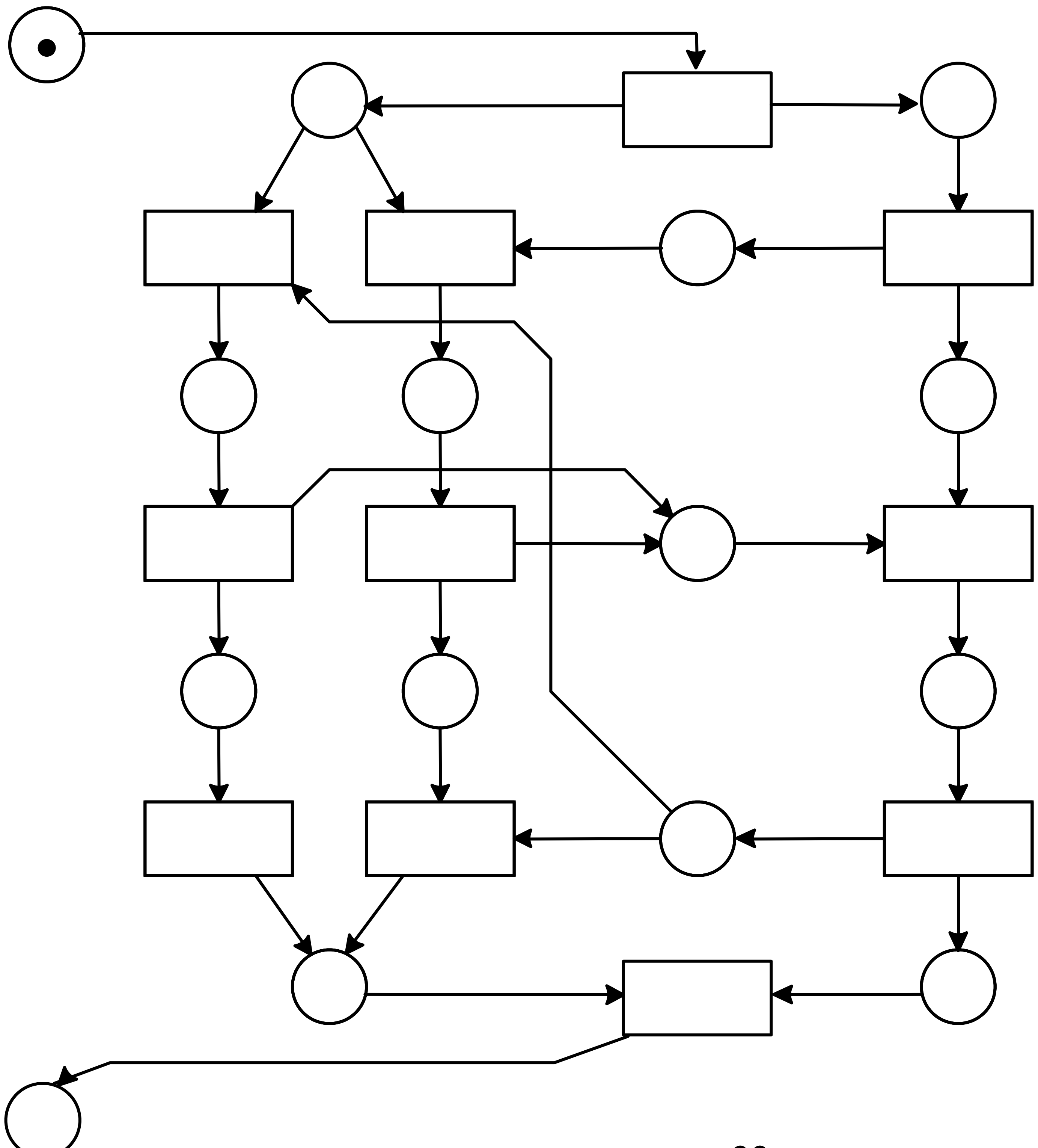
Weak soundness can be checked on the RG

It guarantees deadlock freedom and proper termination of all modules

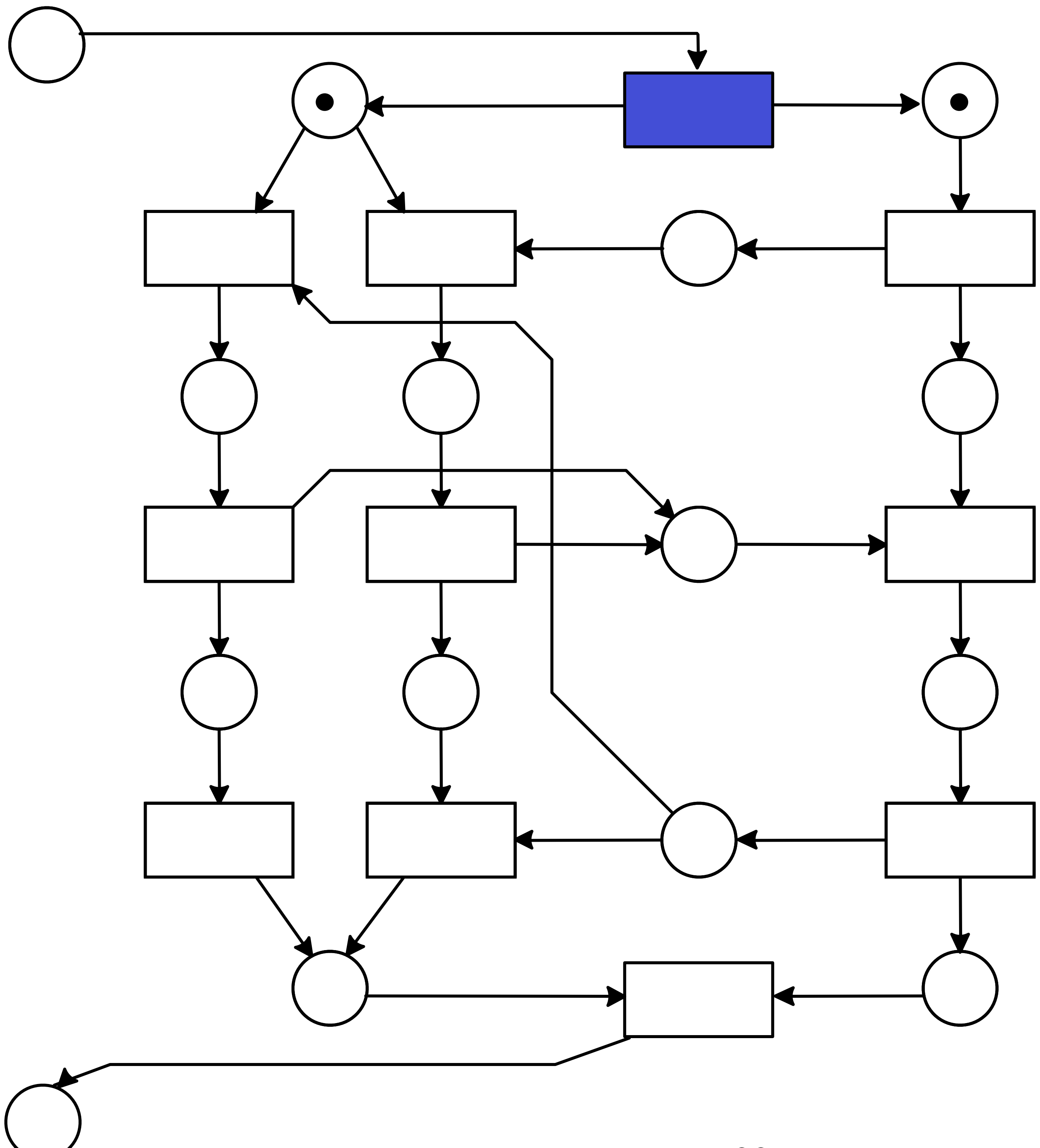
Sound + Sound = ?



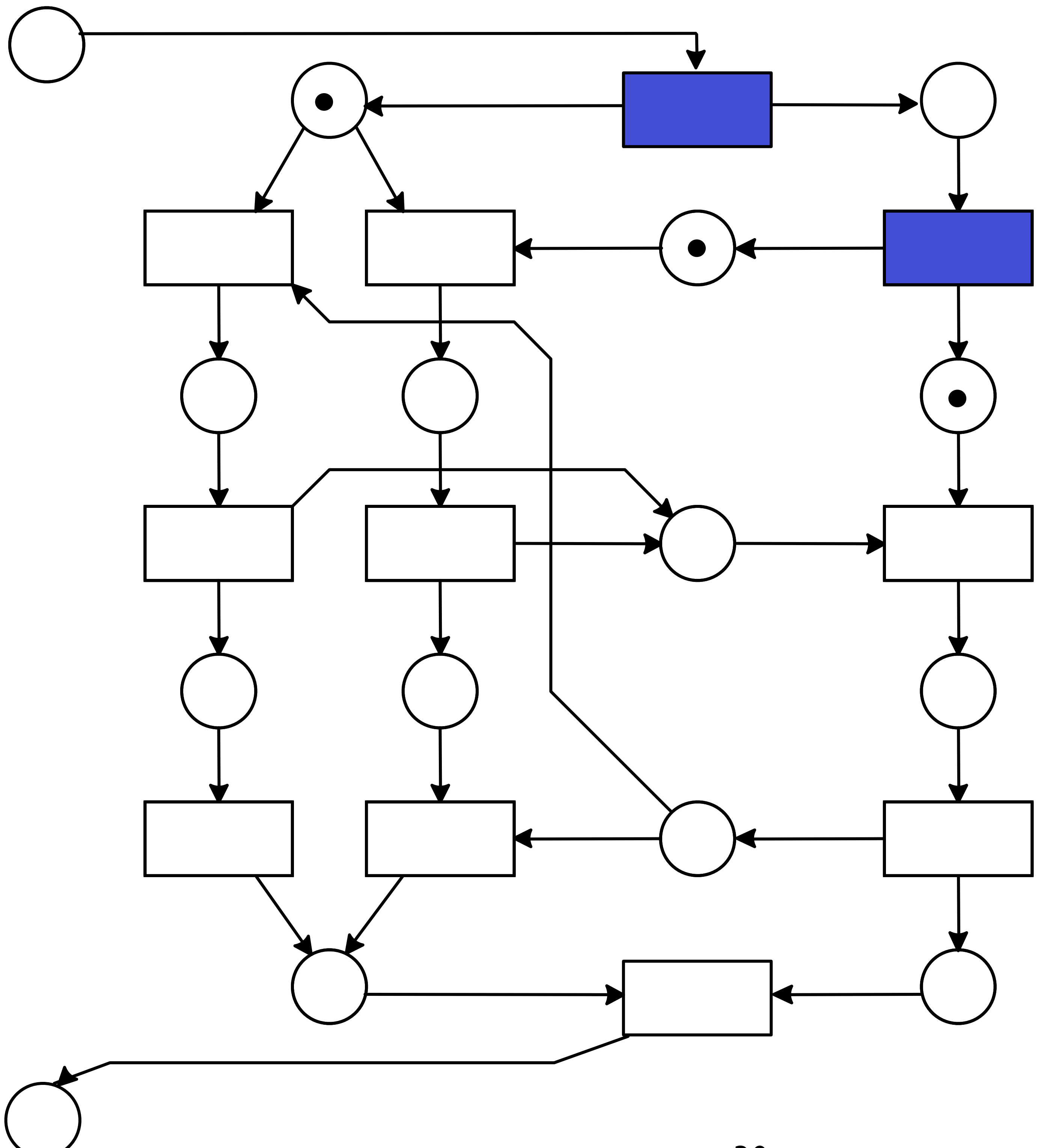
Sound + Sound = not sound



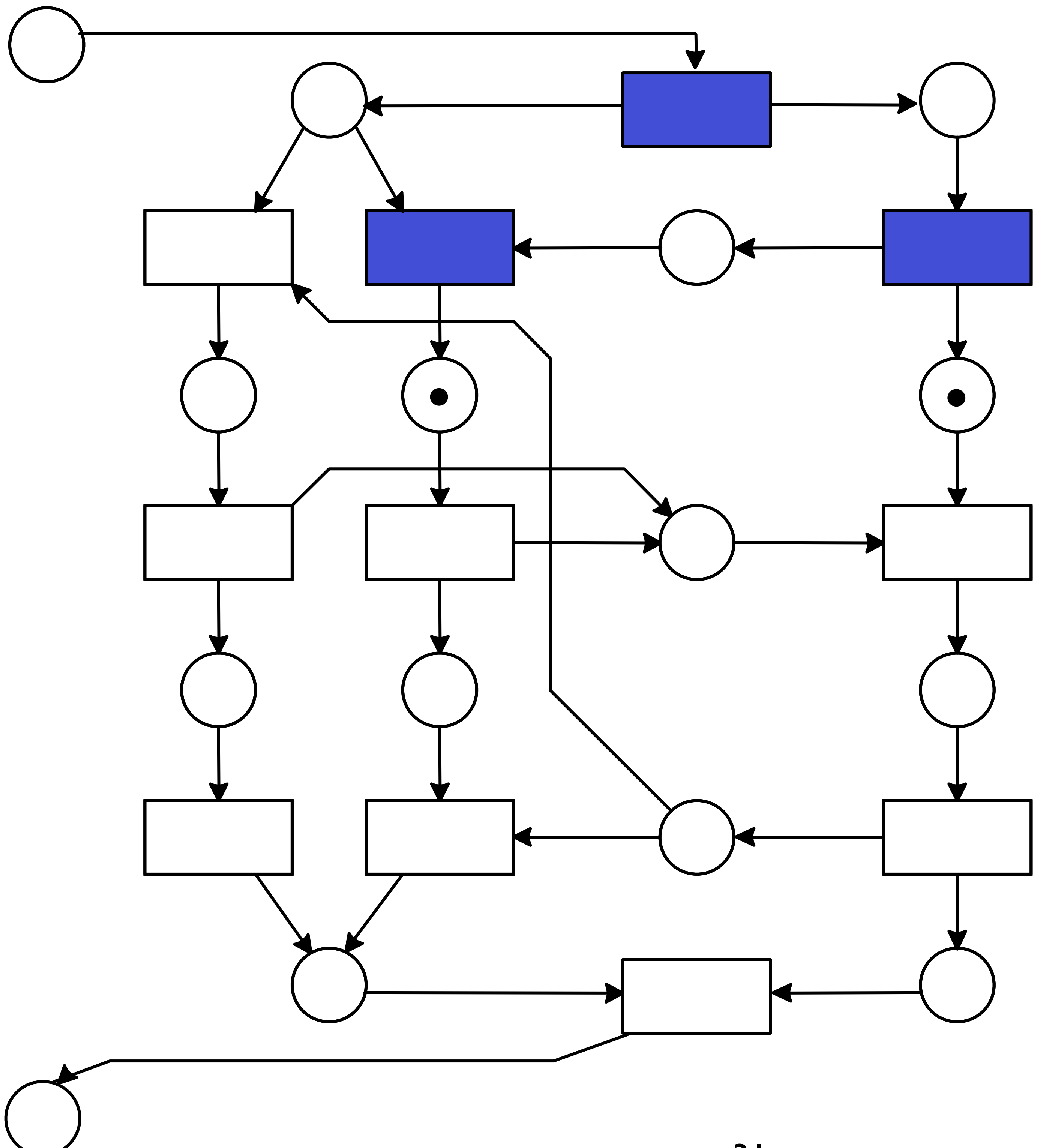
Sound + Sound = not sound



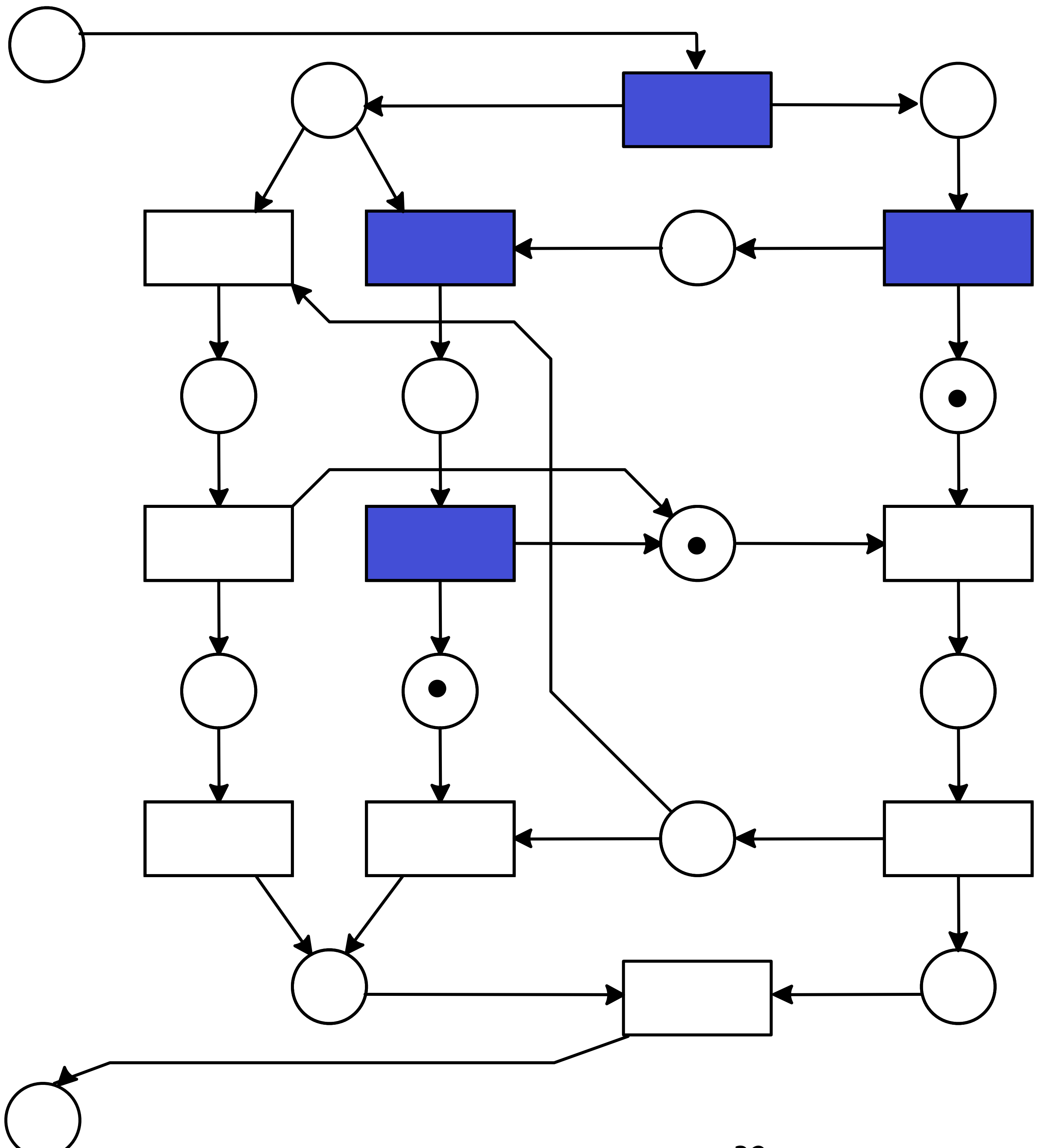
Sound + Sound = not sound



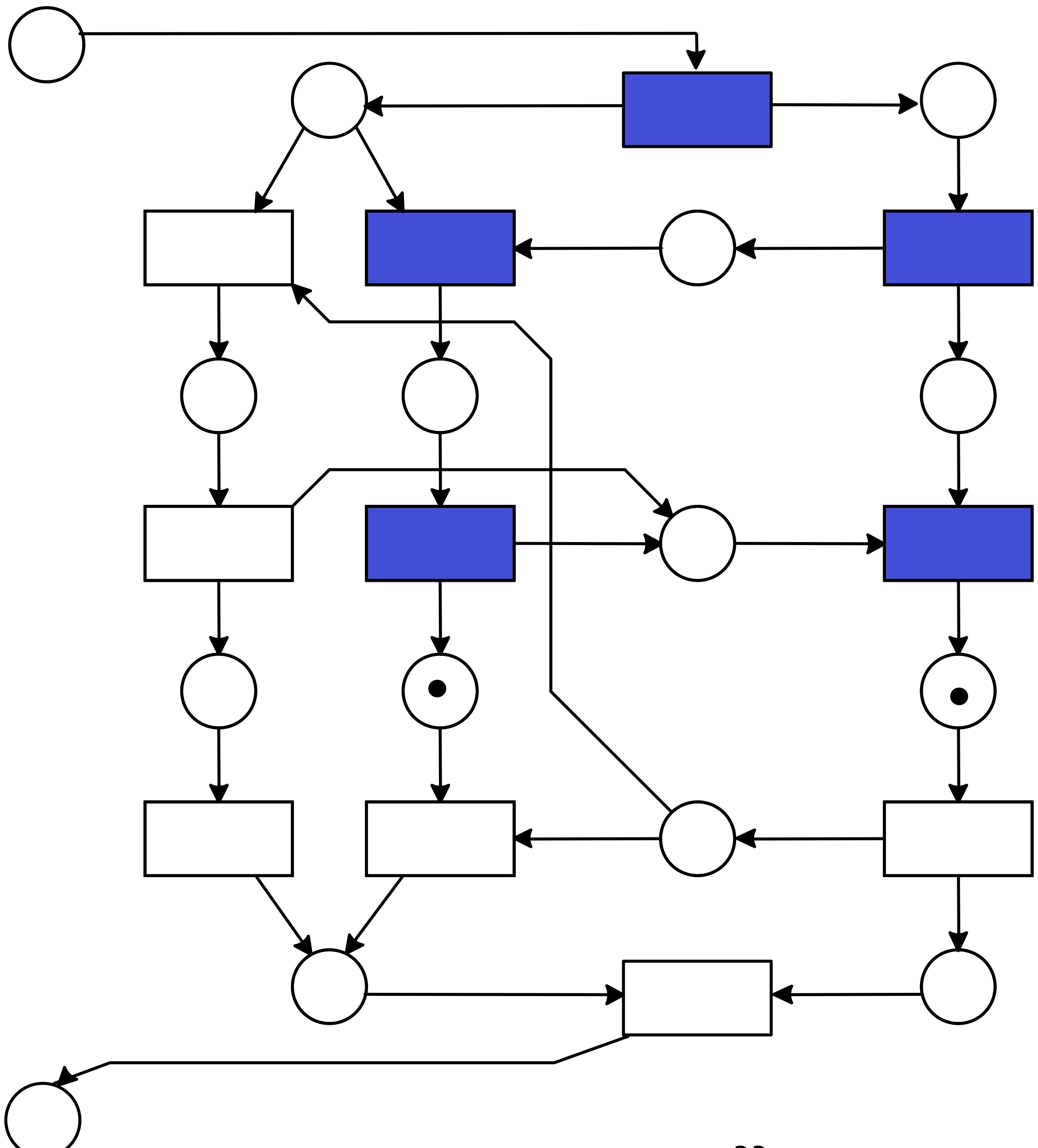
Sound + Sound = not sound



Sound + Sound = not sound

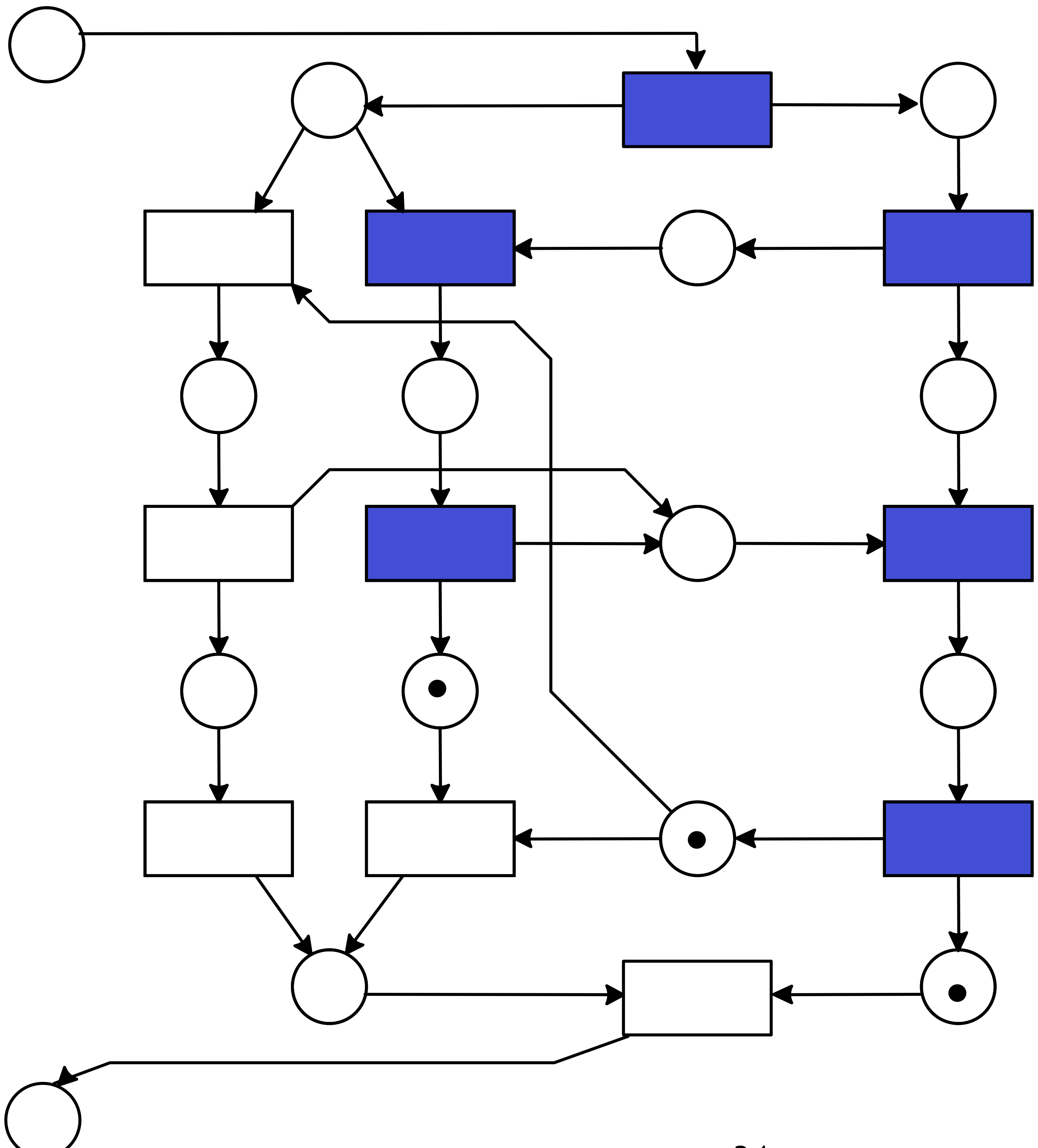


Sound + Sound = not sound

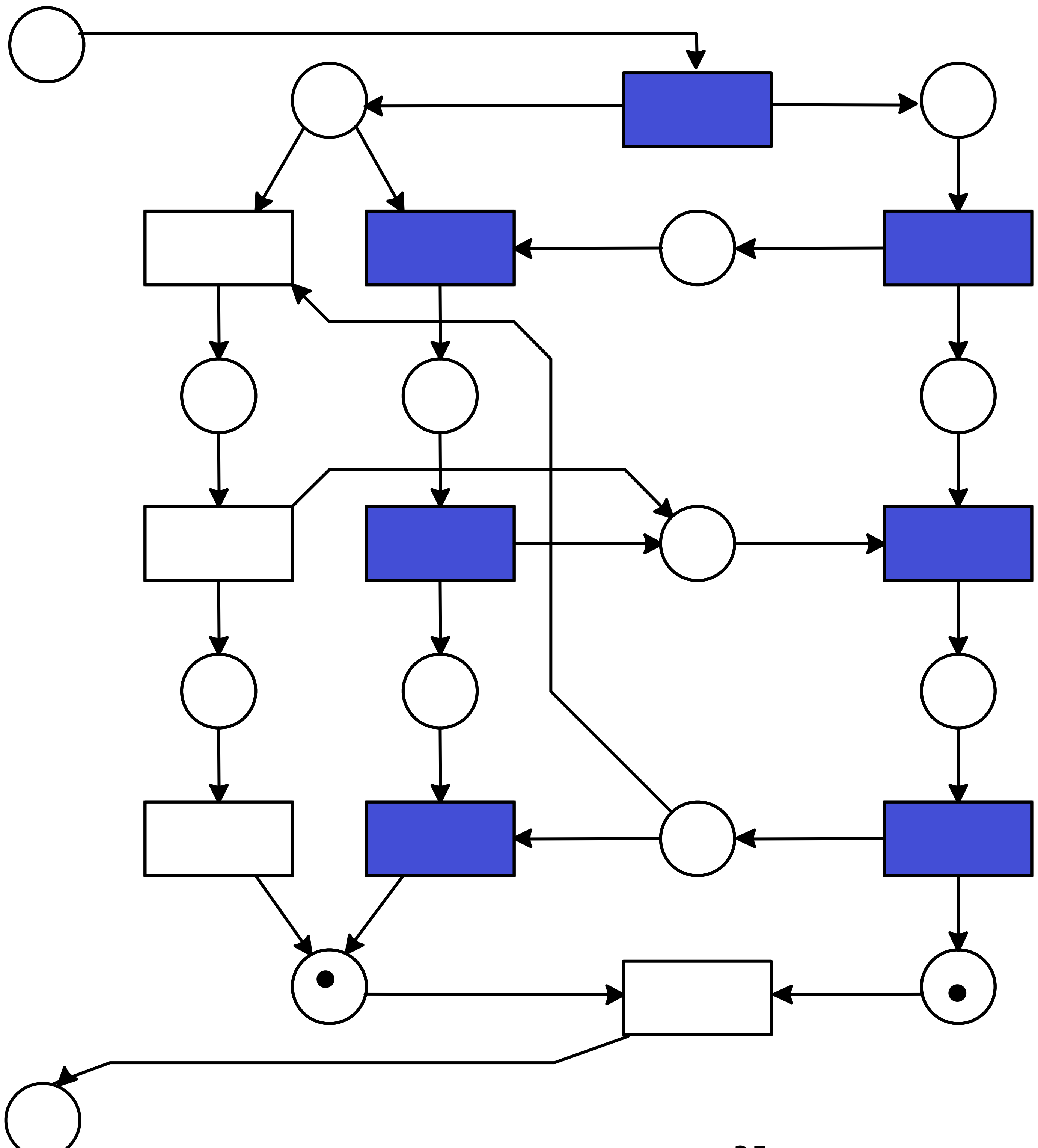


M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

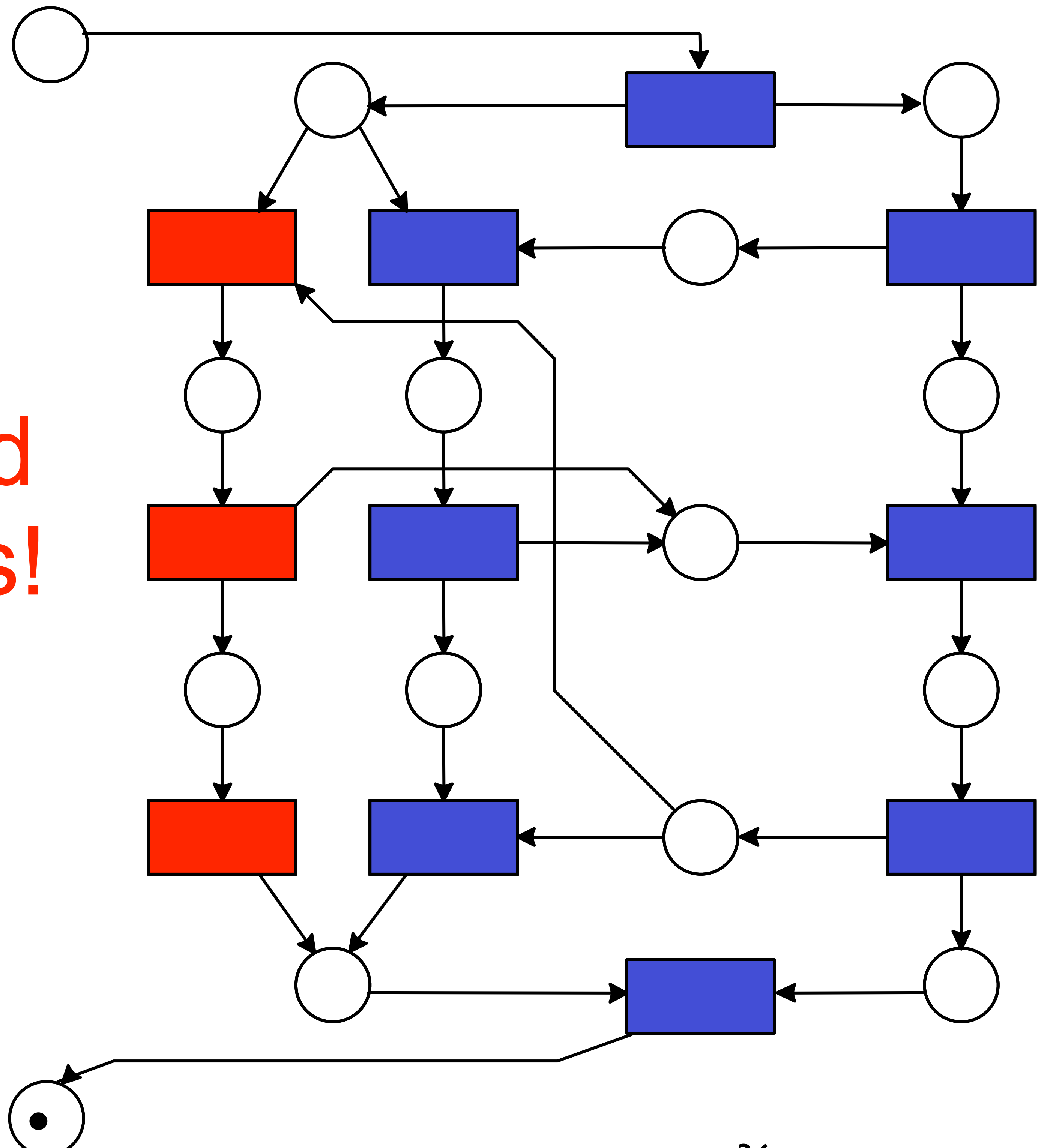
Sound + Sound = not sound



Sound + Sound = not sound



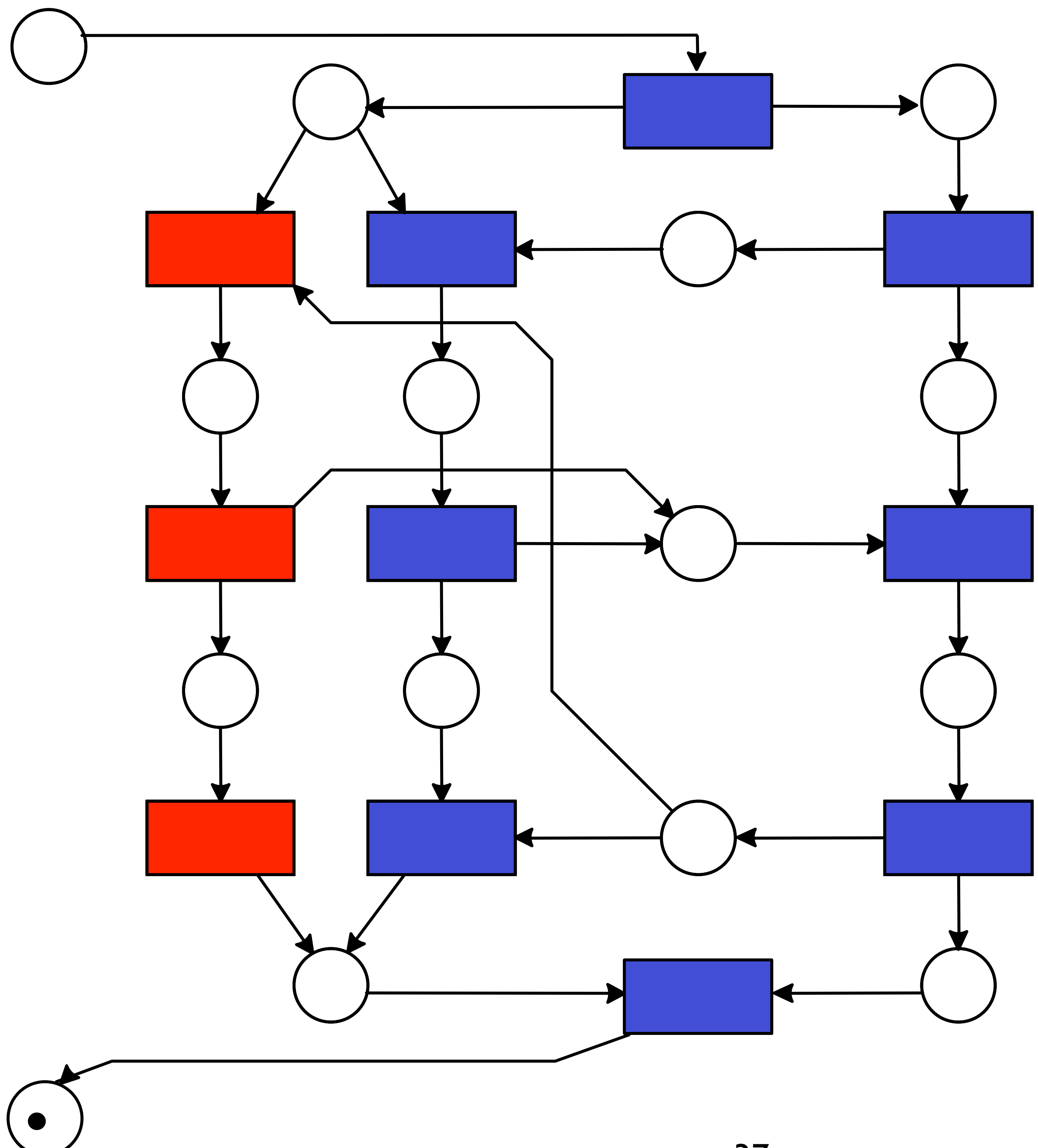
Sound + Sound = not sound



Dead tasks!

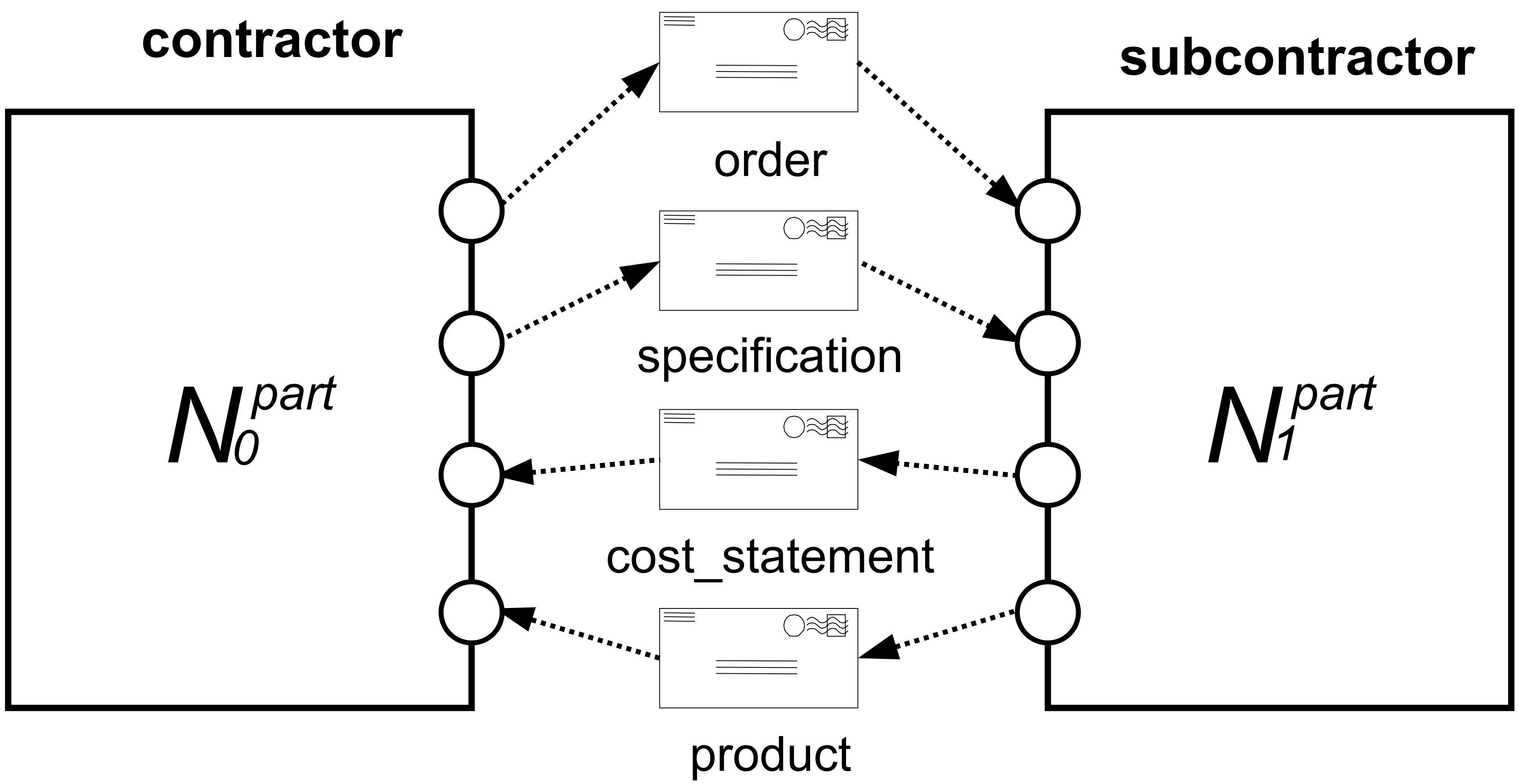
M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2007

Sound + Sound = not sound

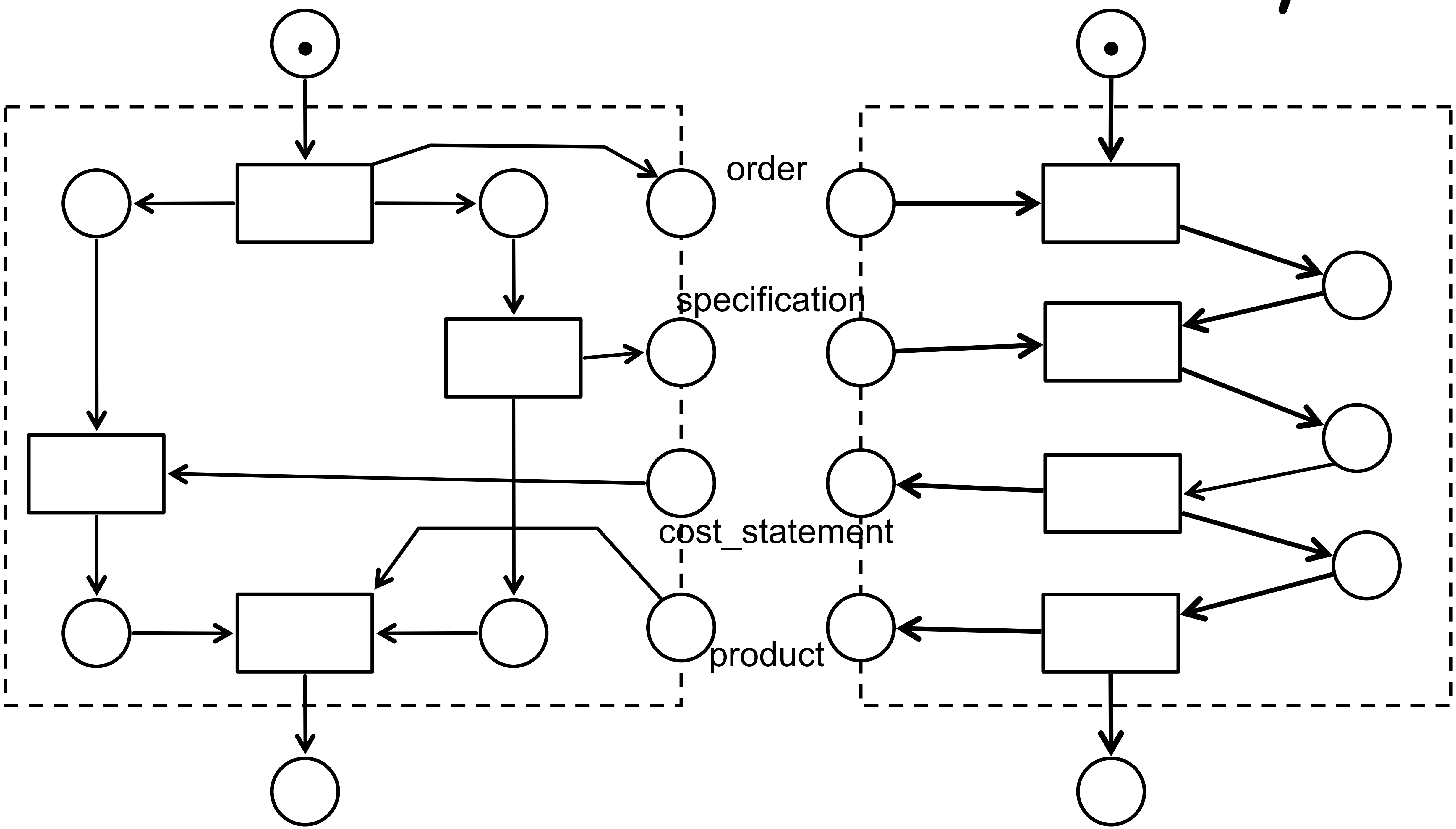


Weak Sound!

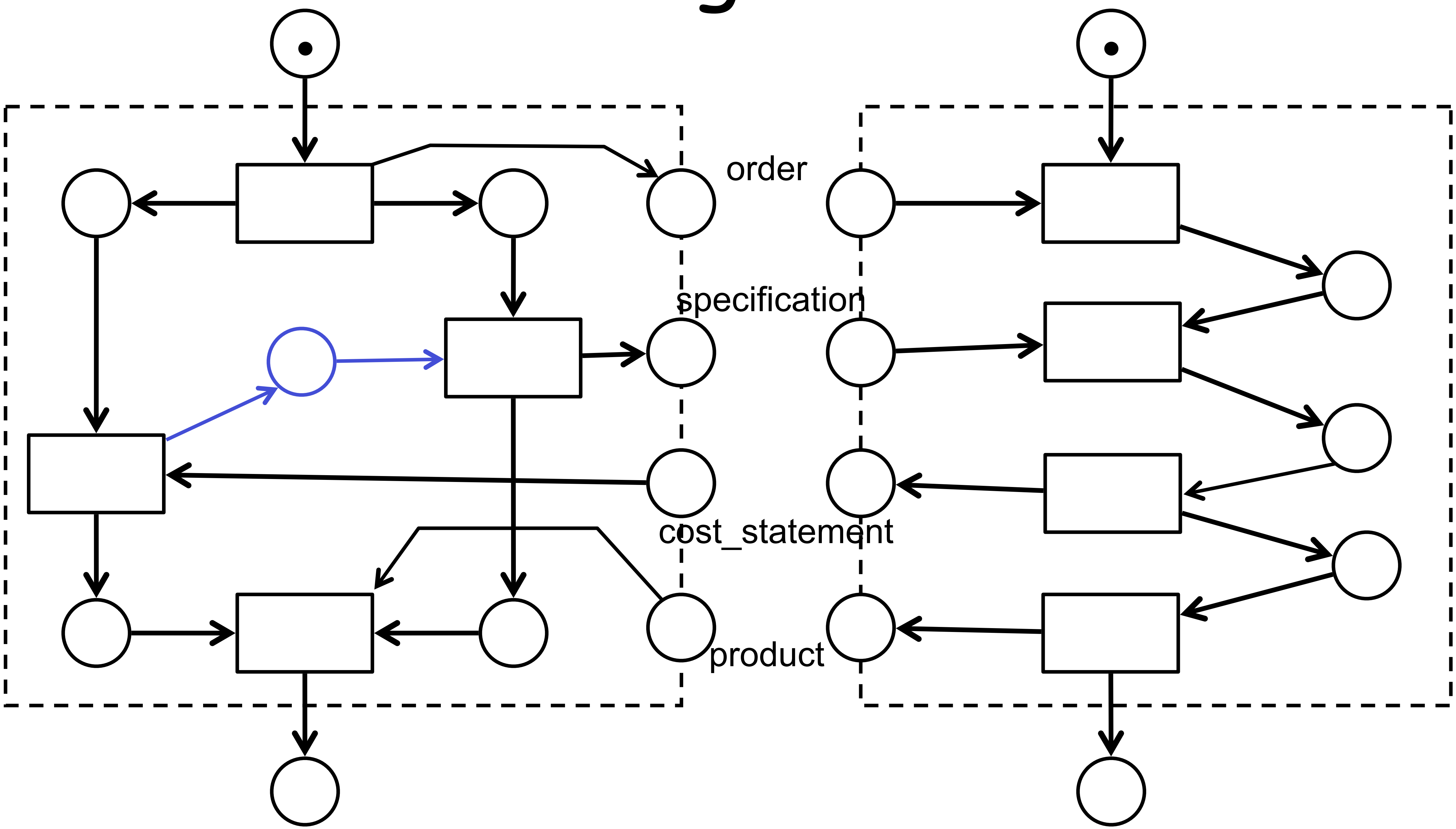
Exercise: Preliminaries



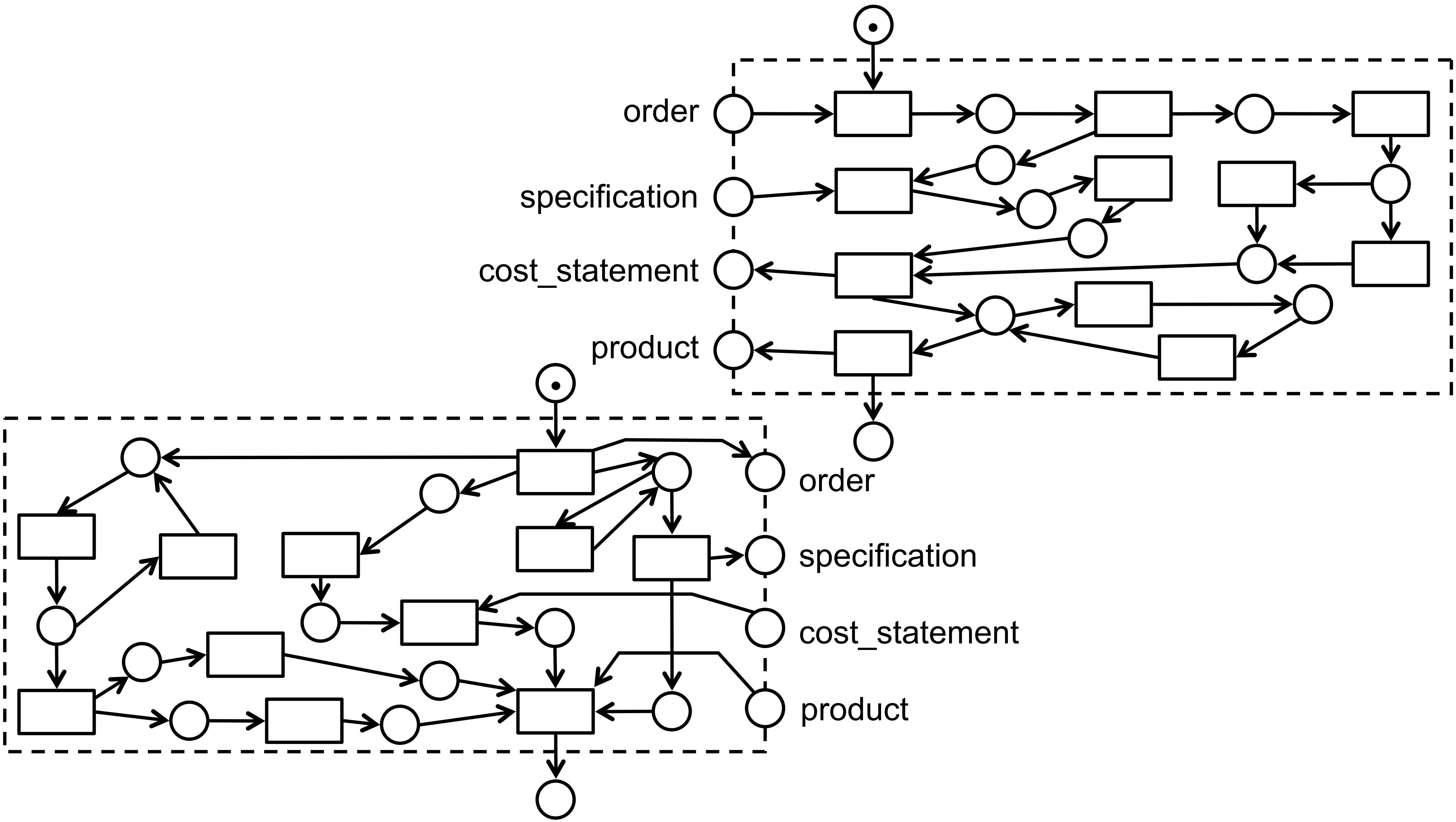
Exercise: Check Weak Soundness of The Assembly



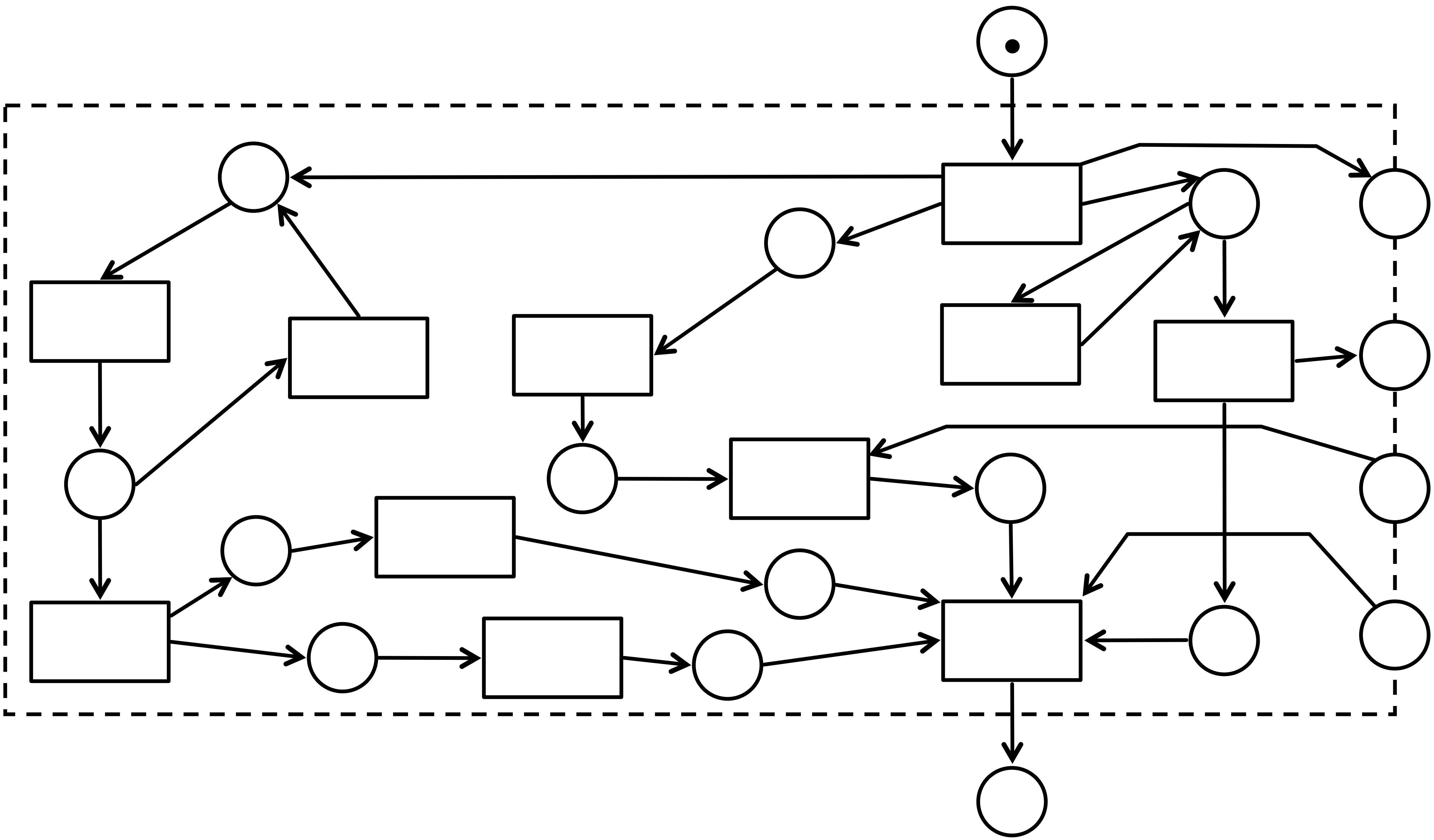
Exercise: Check Again After Refactoring Contractor



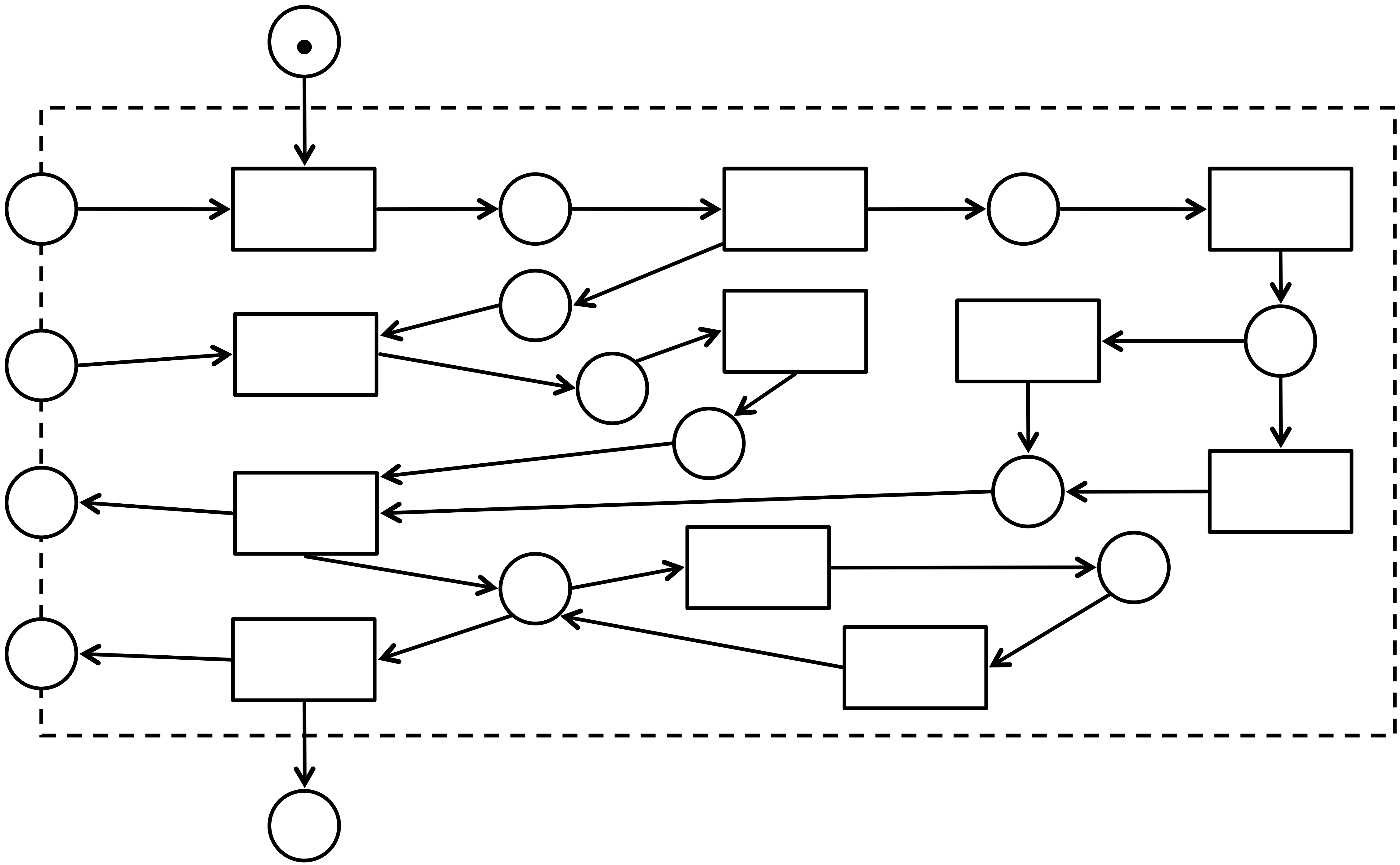
Exercise: Check Again After Refactoring Both



(Contractor zoom-in)



(Subcontractor zoom-in)



Partner existence (aka controllability)

Does My Service Have Partners?

Karsten Wolf

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
karsten.wolf@uni-rostock.de

K. Jensen and W. van der Aalst (Eds.): ToPNoC II, LNCS 5460, pp. 152–171, 2009.

Problem

Processes are designed in isolation
(loose coupling)
We would like their composition to be well-behaved

Given a process:

Can we guarantee that at least one partner exists?

If so, can we synthesize the most permissive partner?

Controllability

Assume a notion of well-behaving is defined

We say that a process N is **controllable**
if it has at least one partner N' such that
the composition of N with N' is well-behaving

Controllability: idea

Given a process N
we aim to construct an automaton that
over-approximates the behaviour of any partner,
then we iteratively remove states and arcs that
invalidate the behavioural property we are after:

if we end up with the empty automaton,
then the process N is **uncontrollable**

otherwise, the automaton defines the most general
strategy to collaborate with N
(guaranteeing the behavioural property we are after)

Open nets

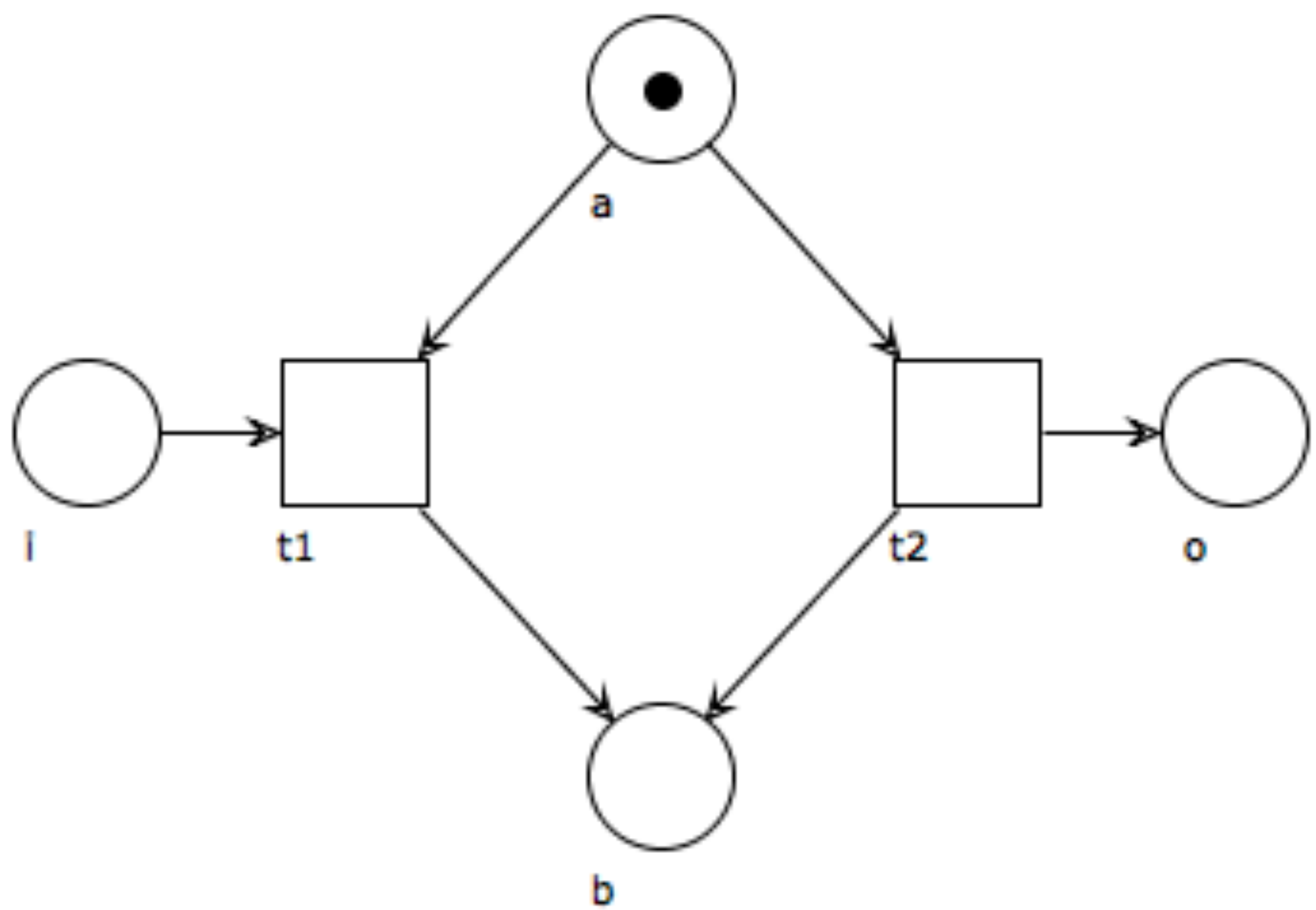
Definition: An **open net** consists of a net (P,T,F,m_0)
plus incoming places $\mathbf{P^I \subseteq P}$
plus outgoing places $\mathbf{P^O \subseteq P}$
plus a finite set $\mathbf{M_f}$ of **final markings**

such that

each transition has
at most one connection to places in the interface

any initial or final marking
does not mark any place in the interface

Example: open net



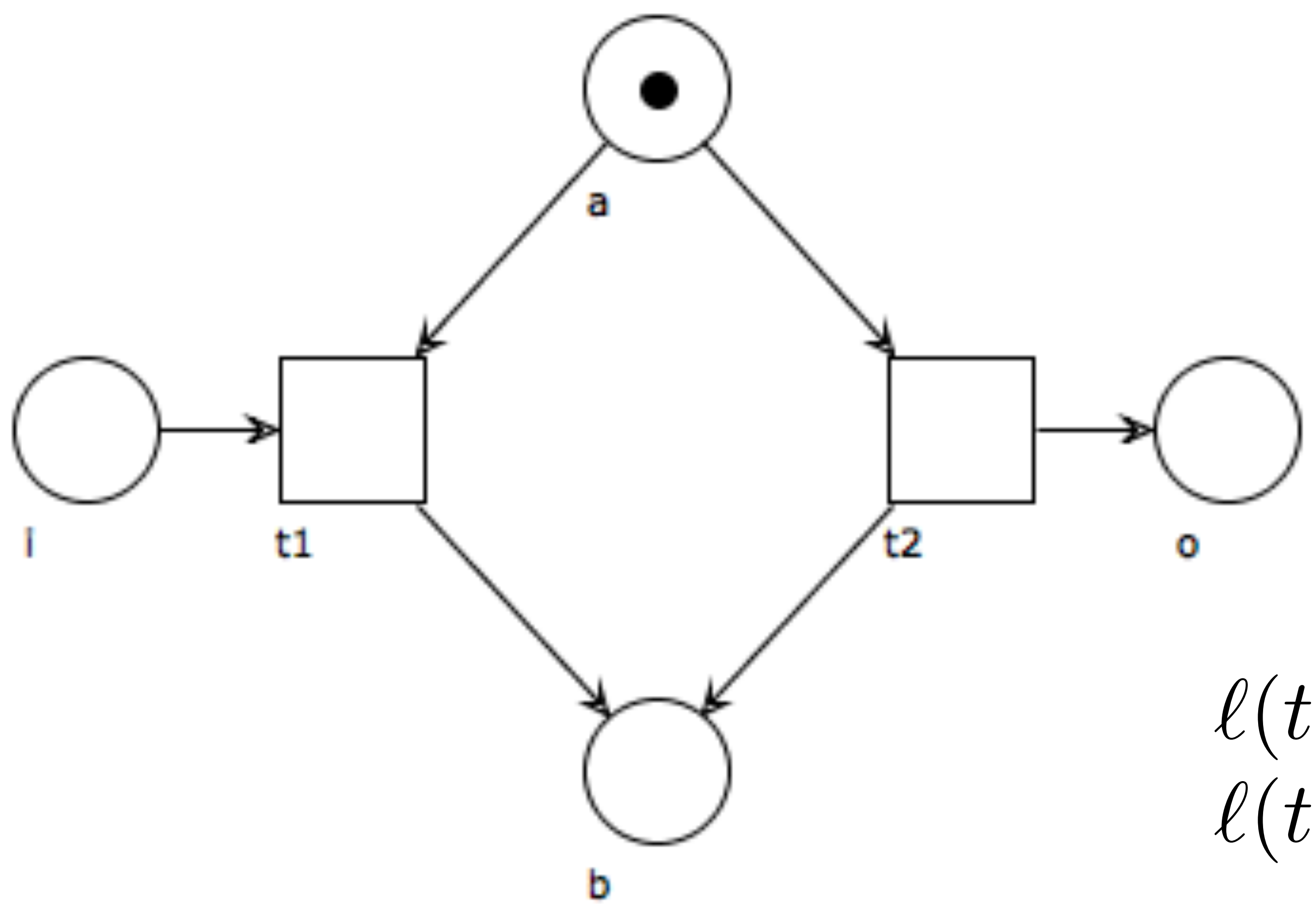
$m_0 = a$
$P^I = \{ i \}$
$P^O = \{ o \}$
$M_f = \{ b \}$

Notation

Definition: The label of a transition t
is the interface place connected to t, if any,
or the special silent action tau otherwise

$$\ell(t) = \begin{cases} x & \text{if } x \in (P^I \cup P^O) \text{ and } (t,x) \in F \vee (x,t) \in F \\ \tau & \text{otherwise} \end{cases}$$

Example: open net



m_0 = a
P^I = { i }
P^O = { o }
M_f = { b }

$$\begin{aligned} \ell(t_1) &= i \\ \ell(t_2) &= o \end{aligned}$$

Closed nets

An open net

$N = (P,T,F,m_0,P^I,P^O,M_f)$

is called **closed** if

$P^I = P^O = \varnothing$

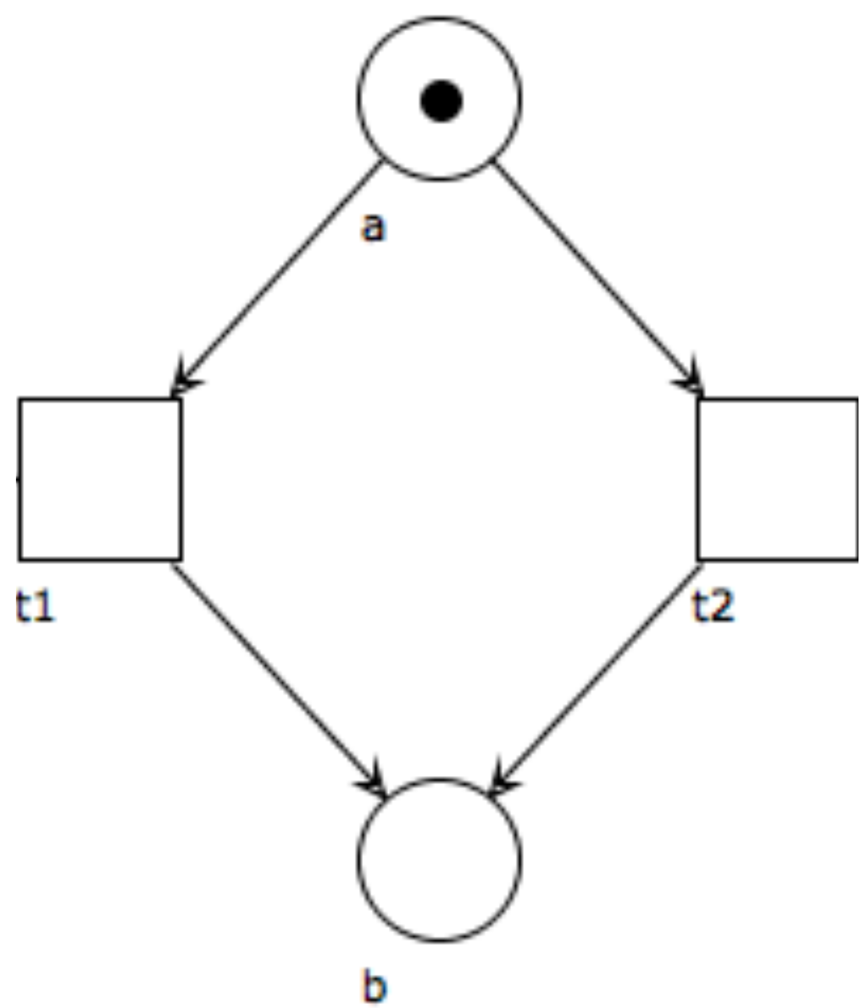
Inner nets

Let $N = (P,T,F,m_0,P^I,P^O,M_f)$ be an open net and let $IO = (P^I \cup P^O)$ be its interface

its **inner** net **In(N)** is the closed net obtained by removing the interface

$$In(N) = (\ P \setminus IO\ ,\ T\ ,\ F \setminus ((IO \times T) \cup (T \times IO))\ ,\ m_0\ ,\ \varnothing\ ,\ \varnothing\ ,\ M_f\ )$$

Example: inner net



Bounded and responsive nets

We focus on open nets N such that
their inner nets In(N) are

bounded
(in the usual sense, they have finite occurrence graphs)

responsive
from any marking m
either a final marking is reachable
or m enables a transition t connected to the interface

Composition of open nets

Given two open nets

$N_1 = (P_1,T_1,F_1,m_{01},P^I_1,P^O_1,M_{f1})$

$N_2 = (P_2,T_2,F_2,m_{02},P^I_2,P^O_2,M_{f2})$

such that $P^I_1=P^O_2$ and $P^O_1= P^I_2$

…

Composition of open nets

Given two open nets

$N_1 = (P_1, T_1, F_1, m_{01}, P^I, P^O, M_{f1})$

$N_2 = (P_2, T_2, F_2, m_{02}, P^O, P^I, M_{f2})$

their composition $N = N_1 \oplus N_2$ is the closed net

$N = (P_1 \cup P_2 \ , \ T_1 \cup T_2 \ , \ F_1 \cup F_2 \ , \ m_{01} + m_{02} \ , \ \varnothing \ , \ \varnothing \ , \ M_{f1} \times M_{f2} \ )$

A final marking of the composed net
is any combination of final markings of the original nets

note that even if N_1 and N_2 are bounded and responsive,
their composition $N_1 \oplus N_2$ is not necessarily so

Behavioural properties: DF

A closed net N = $(P,T,F,m_0,\varnothing,\varnothing,M_f)$

is **DF** (**deadlock-free**)
if any non-final reachable marking enables a transition

$$\forall m \in ([m_0\rangle \setminus M_{\mathrm{f}}).\ \exists t.\ M \stackrel{t}{\rightarrow}$$

DF,k-controllable nets

A bounded and responsive open net
$N = (P,T,F,m_0,P^I,P^O,M_f)$
is **DF,k-controllable**

if there is a (bounded and responsive) partner N'
such that

$N \oplus N'$ is DF
and any place $p \in (P^I \cup P^O)$ is k-bounded in $N \oplus N'$

such an N' is called a **DF,k-strategy** of N

Approach

Start with a bounded and responsive open net
$N = (P,T,F,m_0,P^I,P^O,M_f)$

1st step

Define a strategy **TS_0** which is an automaton
such that a state q of TS_0
represents the set of markings N can be in
while TS_0 is in q

i.e. q is the view of N
according to the interactions observed so far

Note that **TS_0 can be infinite**

Notation

A special symbol **#** tags **final states**

Given a set of markings **M** of N,
we denote by **cl(M)$_N$** the **closure** of **M**

i.e., the set of markings reachable in N
from any of the markings in M

$$cl(M)_N = \{\ m' \ \mid\ \exists m \in M.\ m' \in [M\rangle_N\ \}$$

TS_0

$TS_0 = (Q, E, q_0, Q_\mathrm{f})$

$q_0 = cl(\{m_0\})_N \quad q_0 \in Q$

if $q \in Q$

if $\# \notin q$ any state q has a final counterpart

then $q' = q \cup \# \in Q$, $q \xrightarrow{\tau} q \in E$, $q \xrightarrow{\tau} q' \in E$

if $x \in P^I \wedge \# \notin q$ TS_0 simulates the production of messages in input places

then $q' = cl(\{m + x \mid m \in q\}) \in Q$, $q \xrightarrow{x} q' \in E$

if $x \in P^O$ TS_0 simulates the consumption of messages from output places

then $q' = \{m - x \mid m \in q, m(x) > 0\} \setminus \{\#\} \in Q$, $q \xrightarrow{x} q'$

$Q_\mathrm{f} = \{q \in Q \mid \# \in q\}$

Approach

Starting from the strategy TS_0

2nd step

Define a strategy **TS_1**

that removes from TS_0 all states q

that contains a marking m that exceeds the capacity

bound k for some place in the interface

(as a consequence remove all adjacent edges

and all states that become unreachable)

TS_1 is always a finite automaton

(actually it can be constructed directly from N)

TS_1

$$TS_0 = (Q, E, q_0, Q_\mathrm{f})$$

$$Q_1 = Q \setminus \{q \in Q \mid \exists m \in q.\ \exists x \in (P^I \cup P^O).\ m(x) > k\}$$

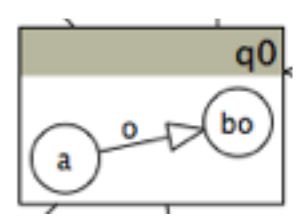
$$E_1 = \{q \xrightarrow{\alpha} q' \in E \mid q, q' \in Q_1\}$$

$$Q_{\mathrm{f}1} = Q_\mathrm{f} \cap Q_1$$

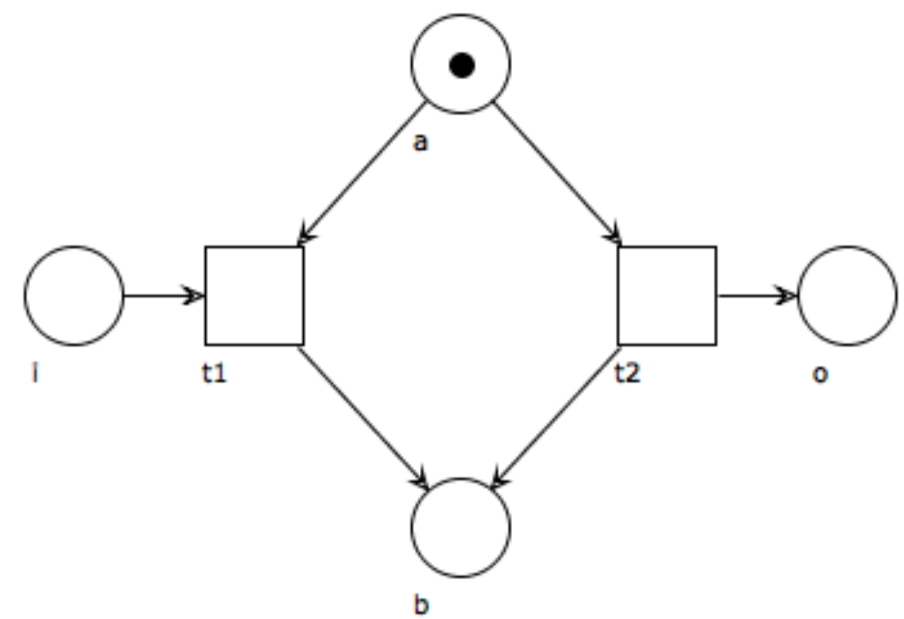
$$TS_1 = (Q_1, E_1, q_0, Q_{\mathrm{f}1})$$

DF,1-controllability example: TS_1

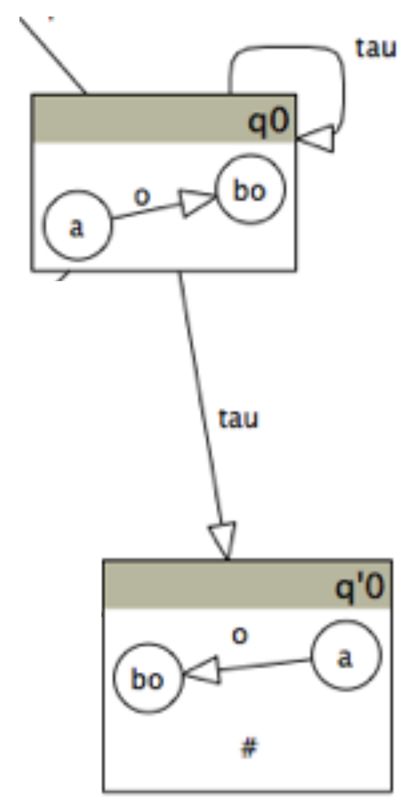
$cl(\{a\})_N$



N

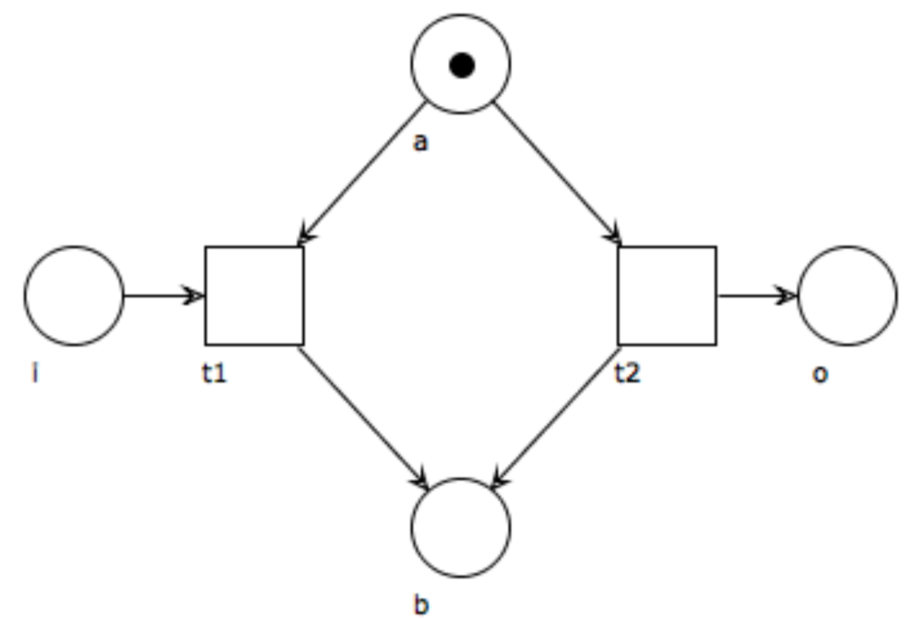


DF,1-controllability example: TS_1



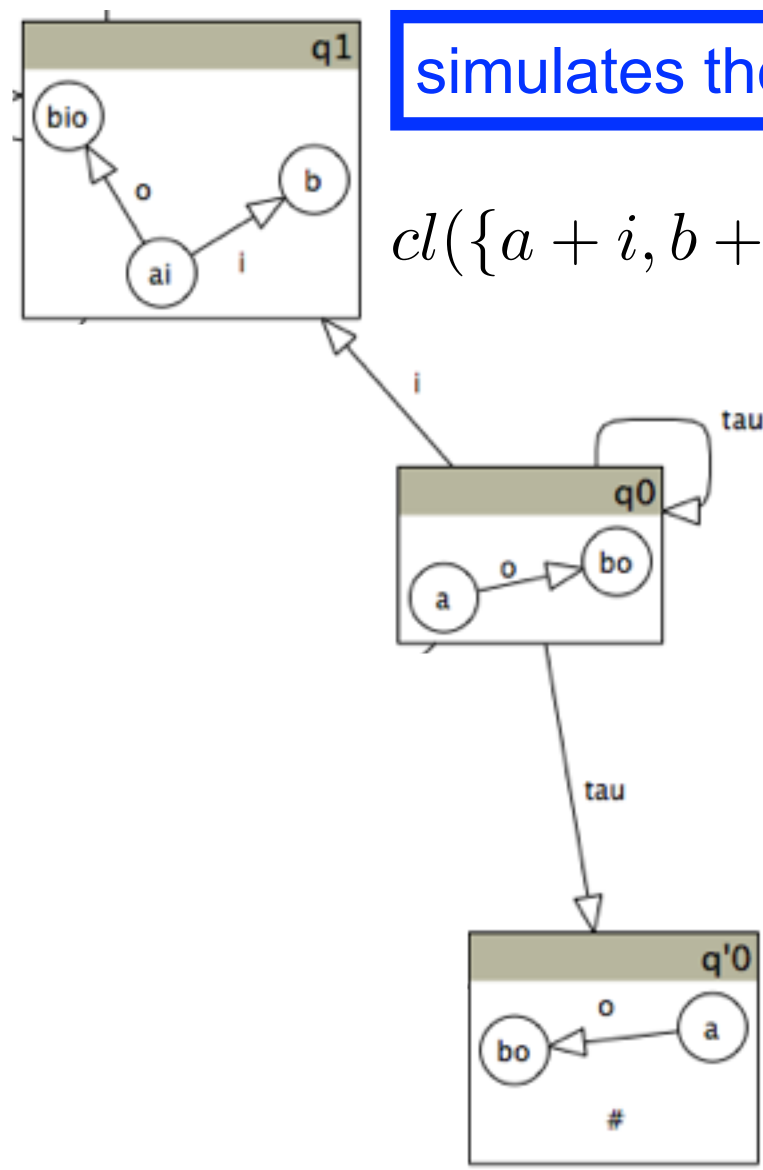
any state q has a final counterpart

N



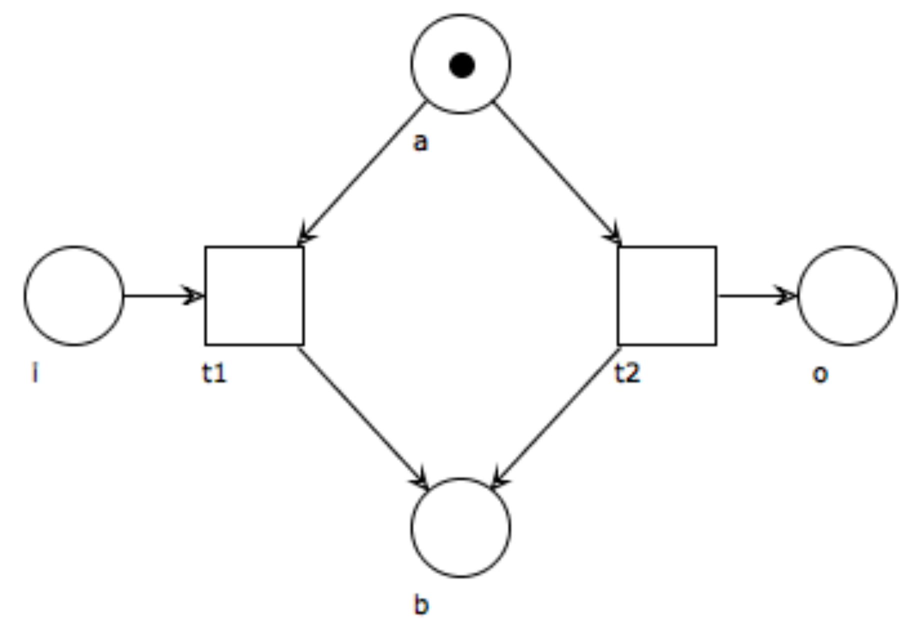
DF,1-controllability example: TS_1

simulates the production of messages in input places

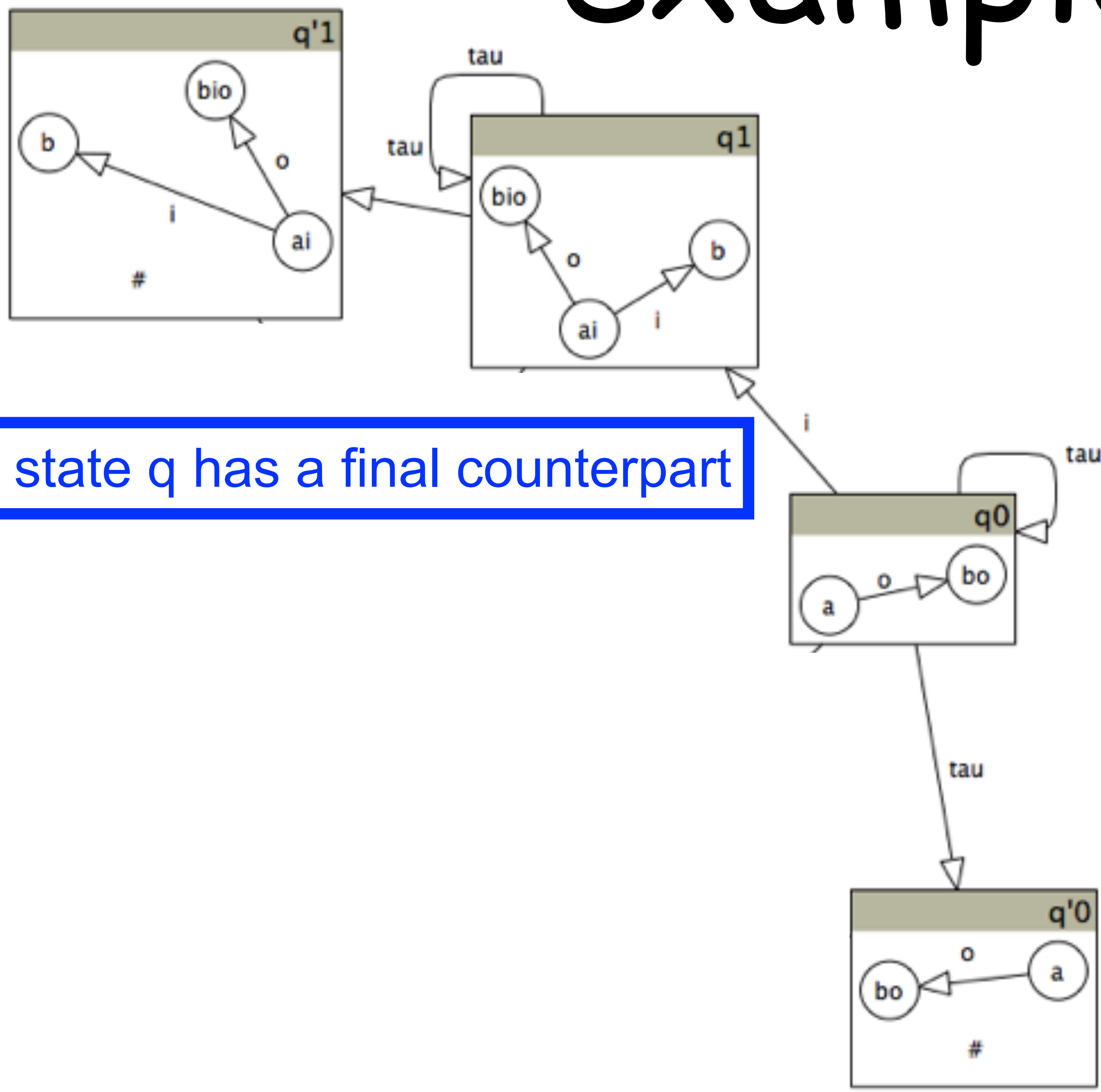


$$cl(\{a+i, b+o+i\})_N$$

N

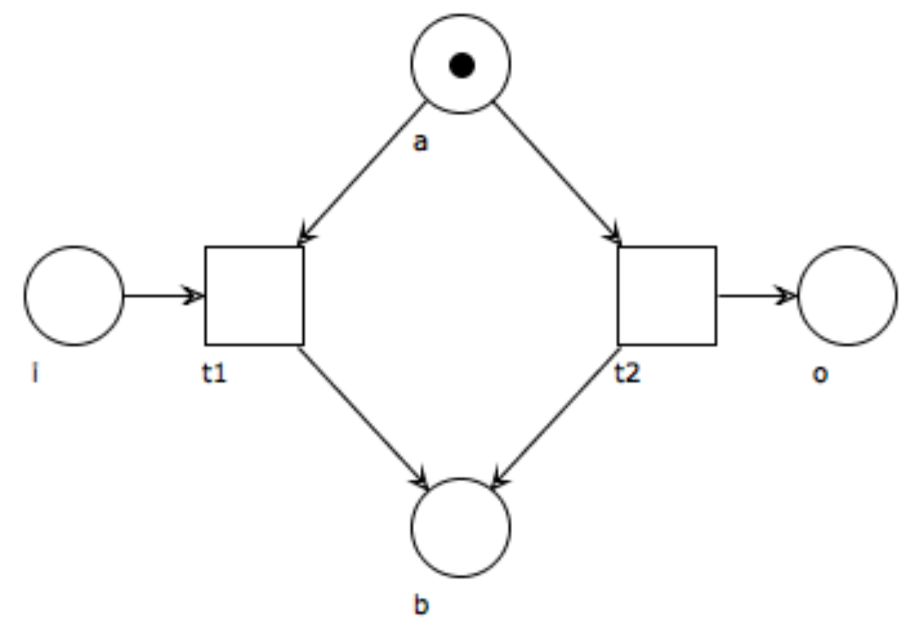


DF,1-controllability example: TS_1

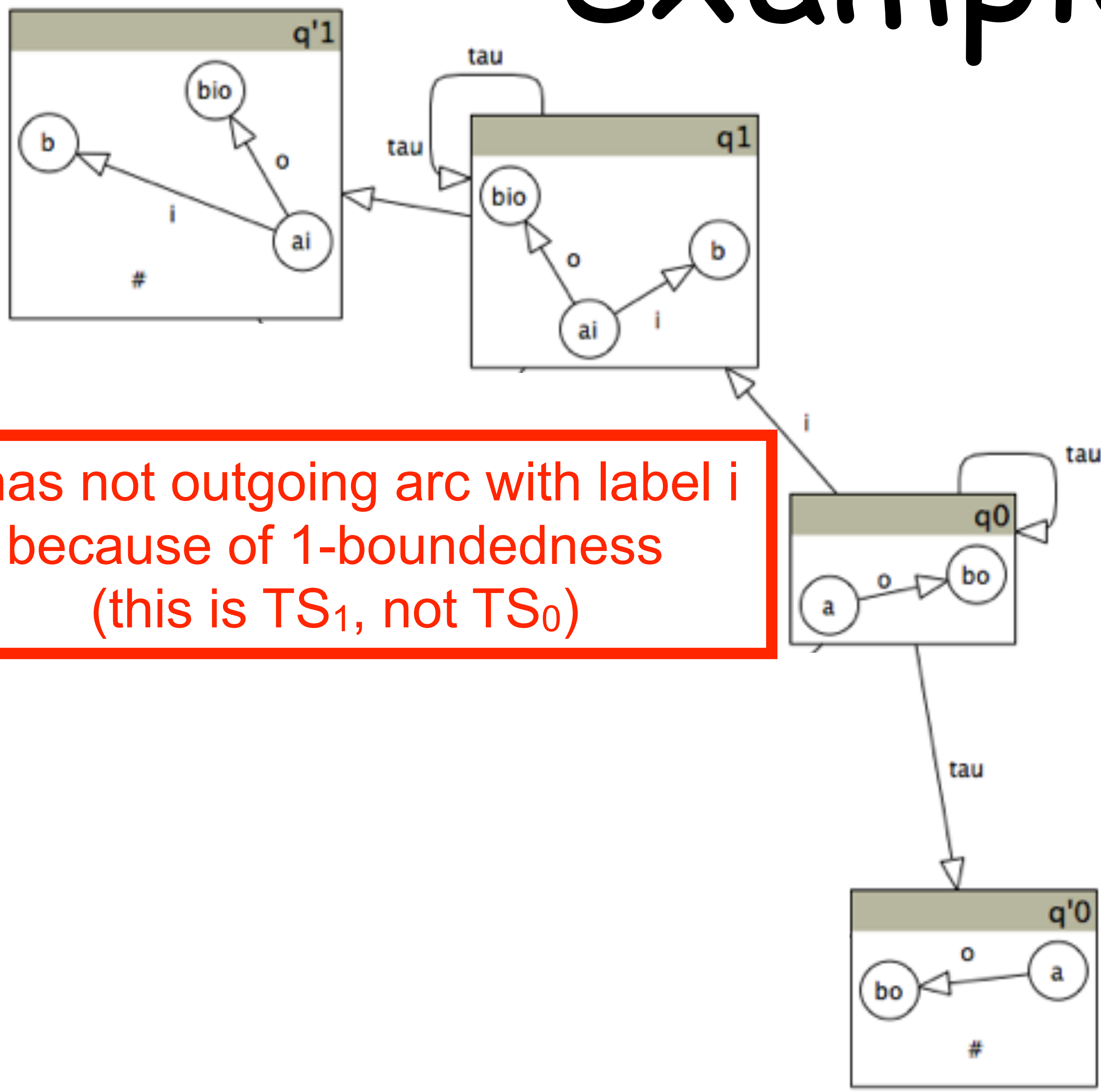


any state q has a final counterpart

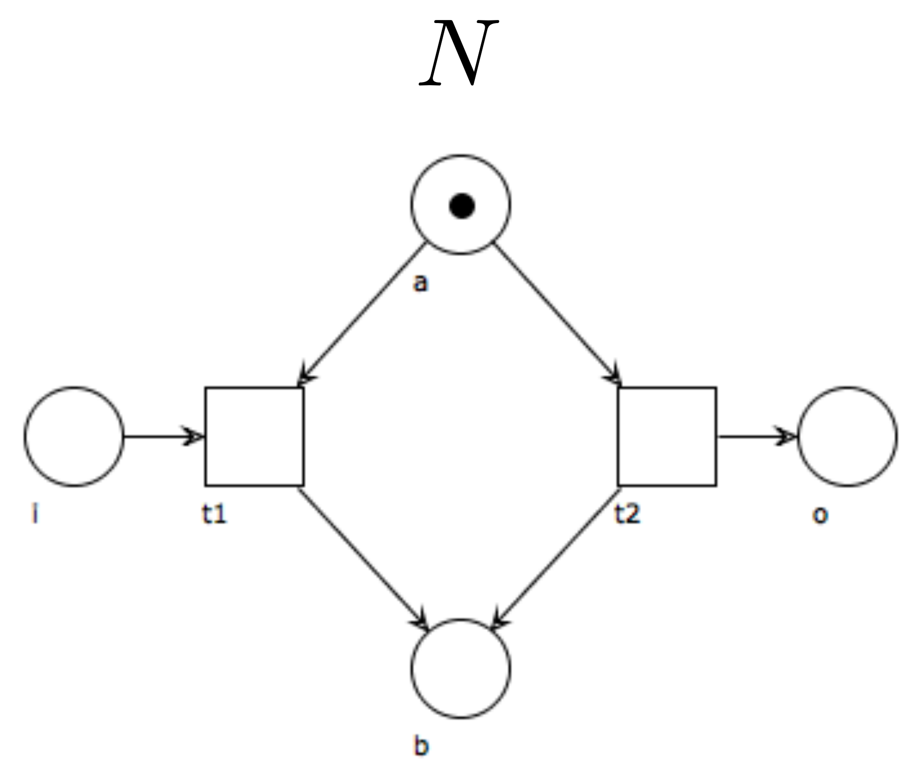
N



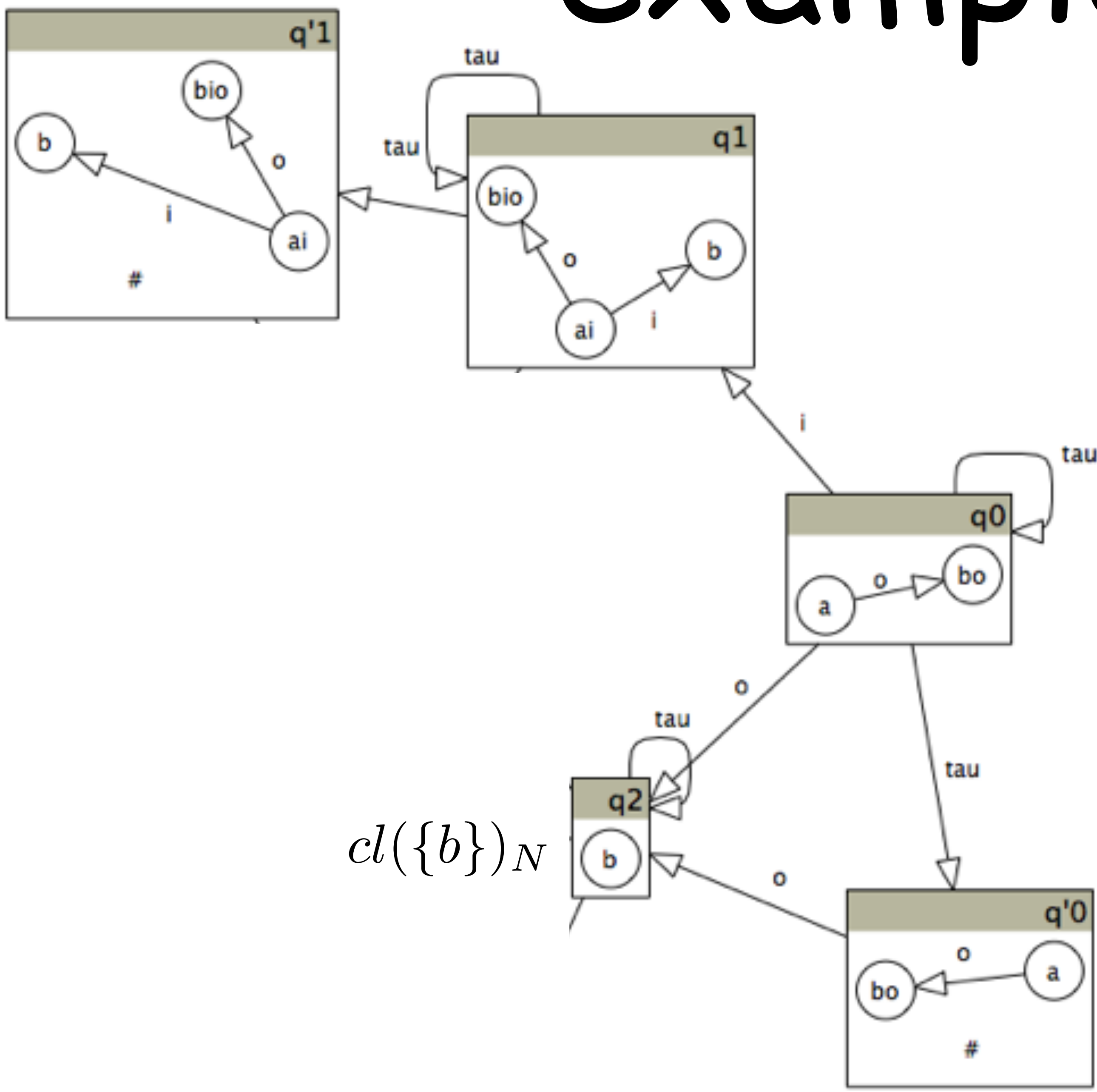
DF,1-controllability example: TS_1

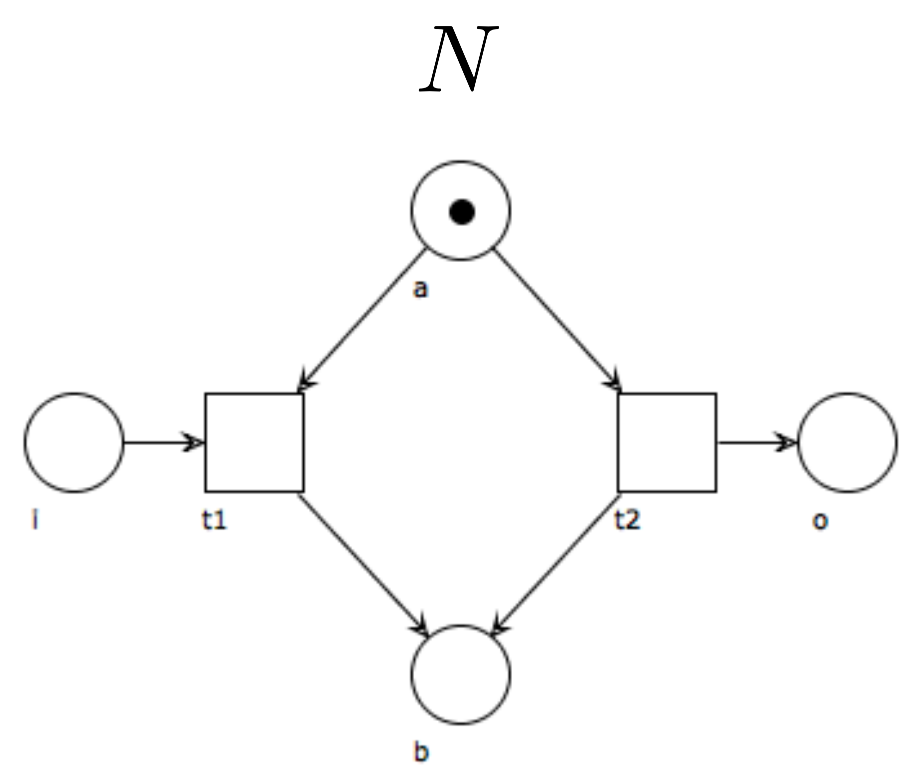


q_1 has not outgoing arc with label i because of 1-boundedness (this is TS_1, not TS_0)



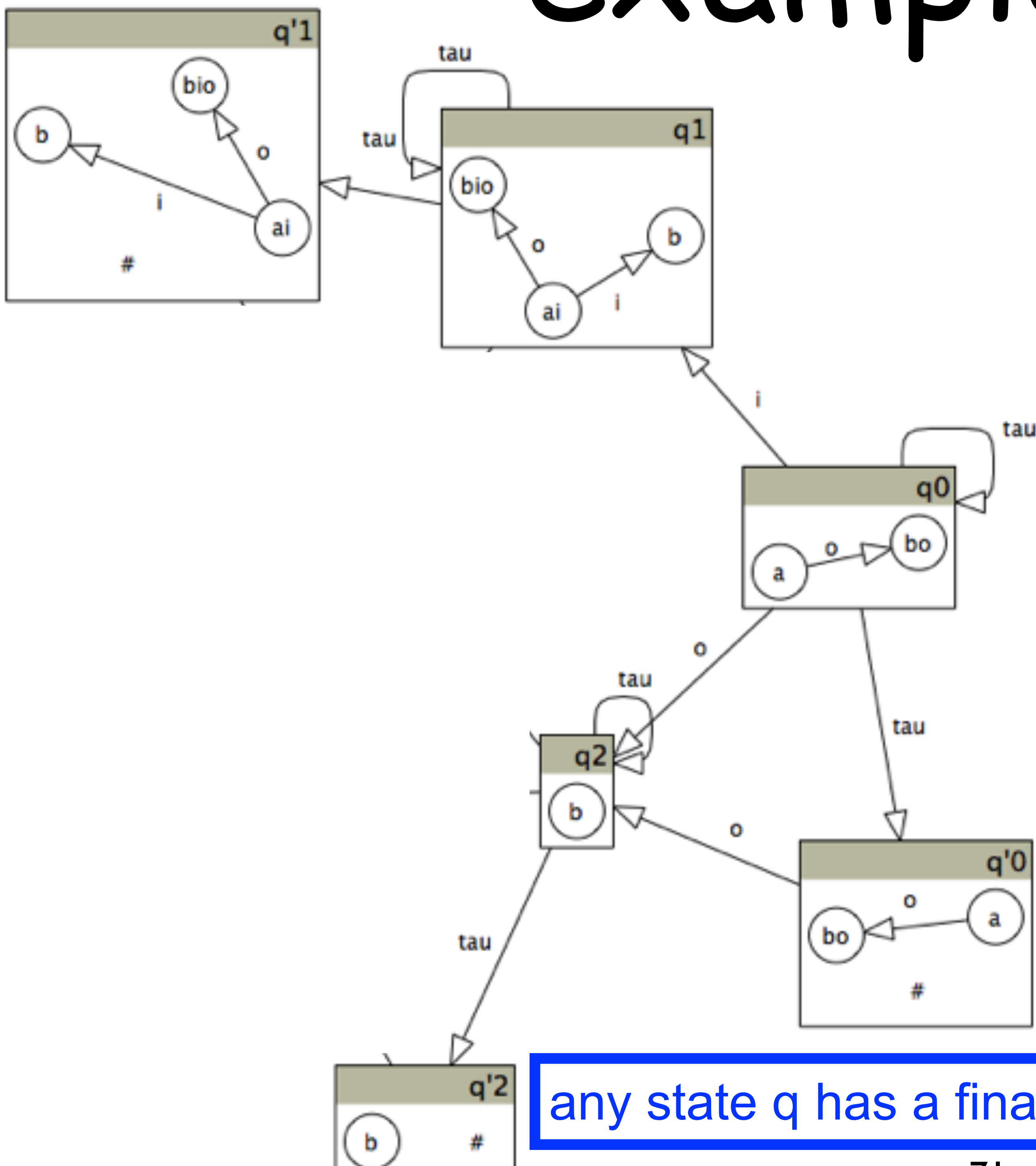
DF,1-controllability example: TS_1



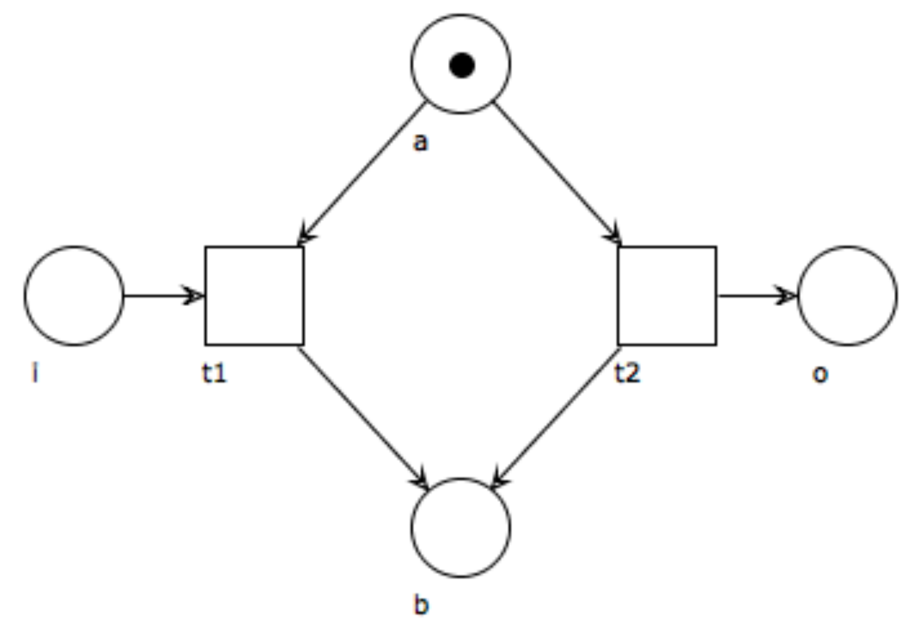


simulates the consumption of messages from output places

DF,1-controllability example: TS_1

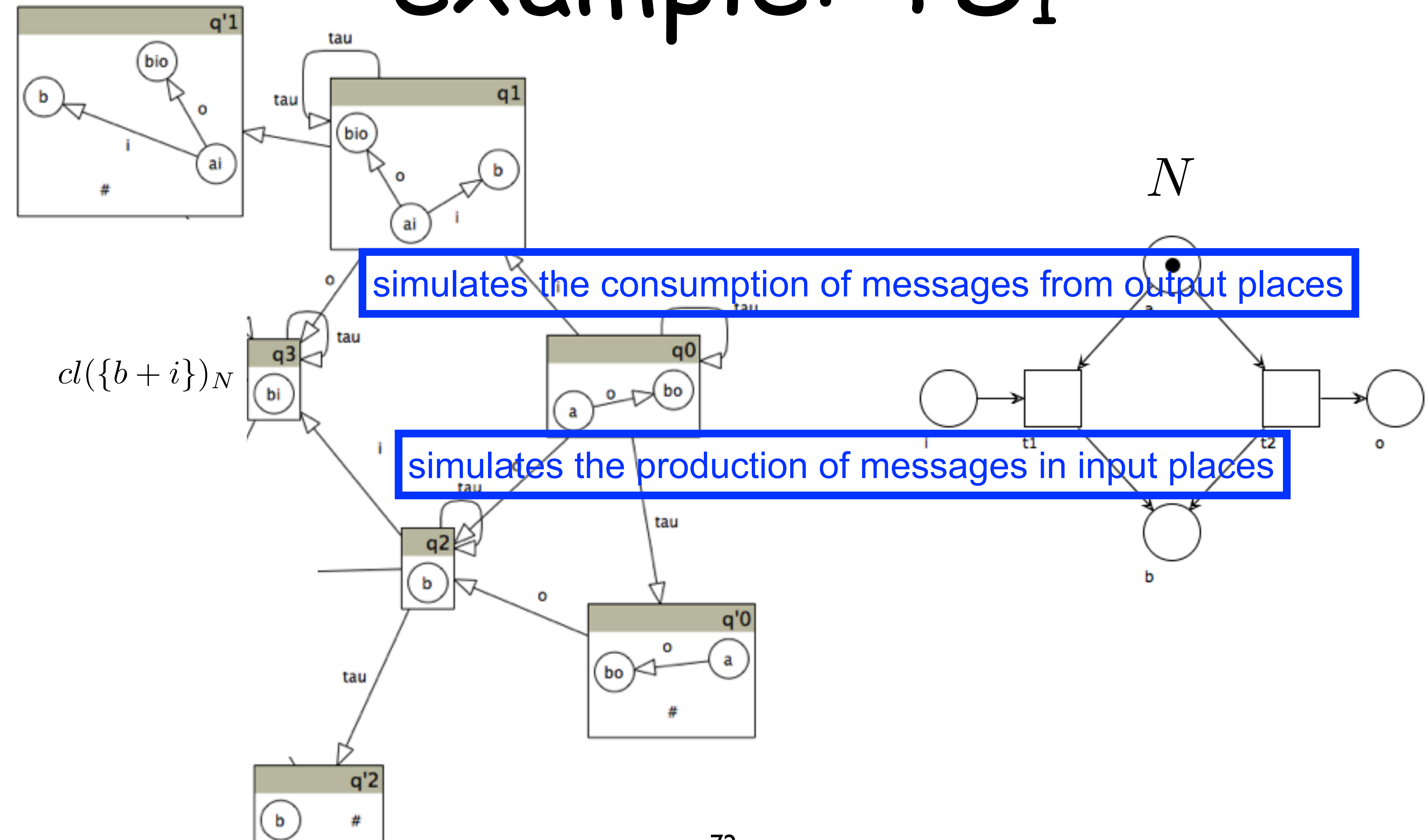


N

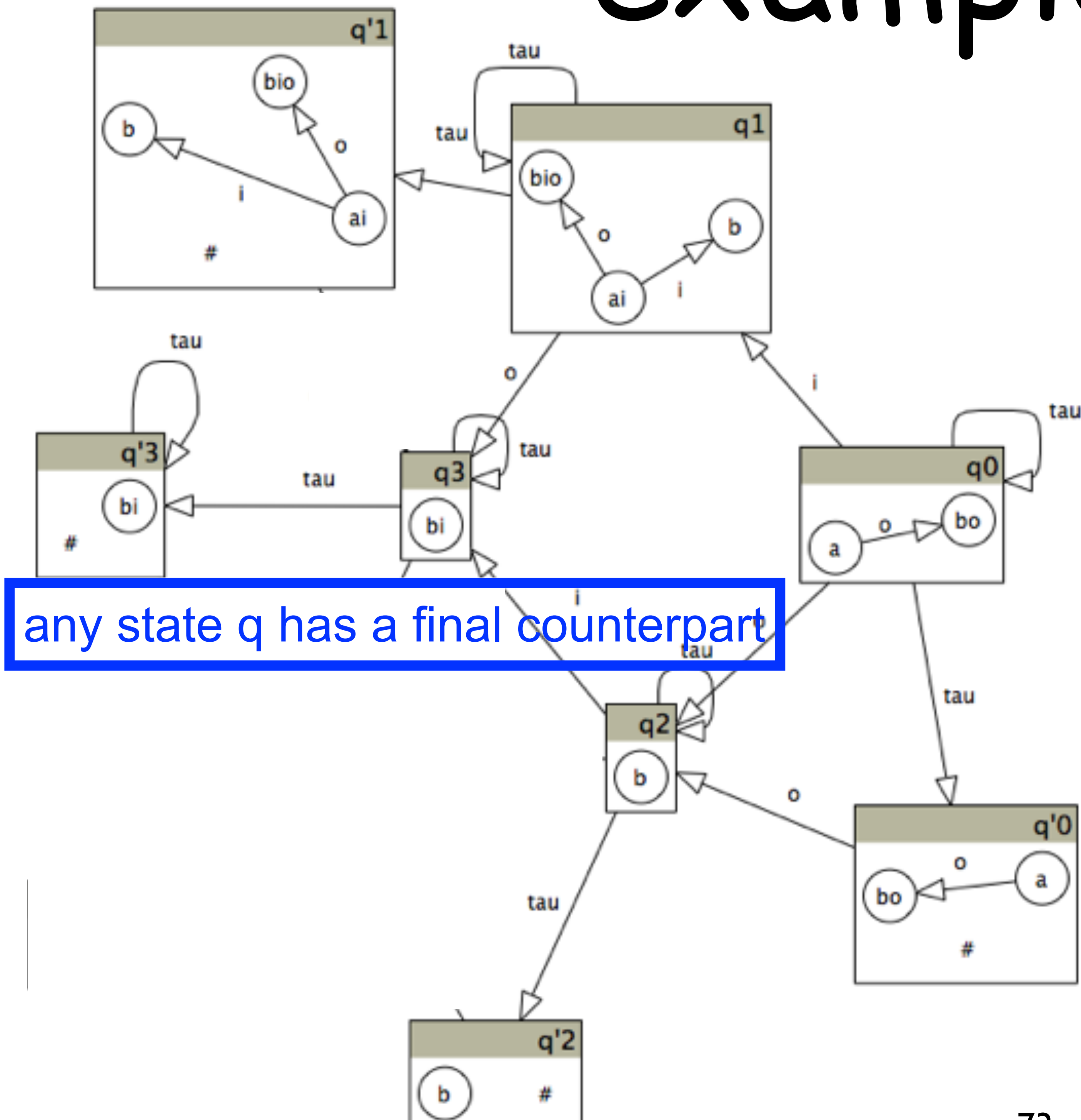


any state q has a final counterpart

DF,1-controllability example: TS_1

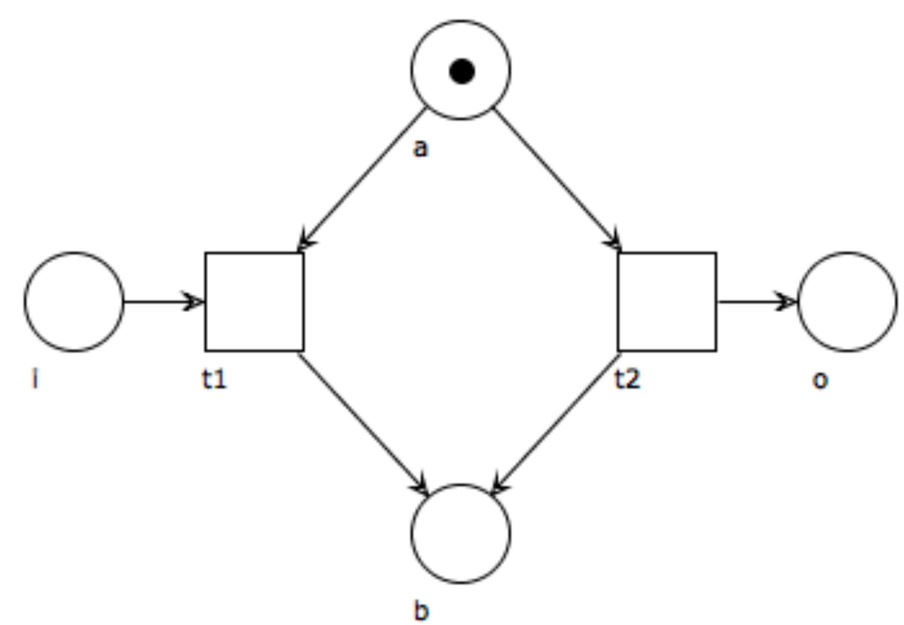


DF,1-controllability example: TS_1

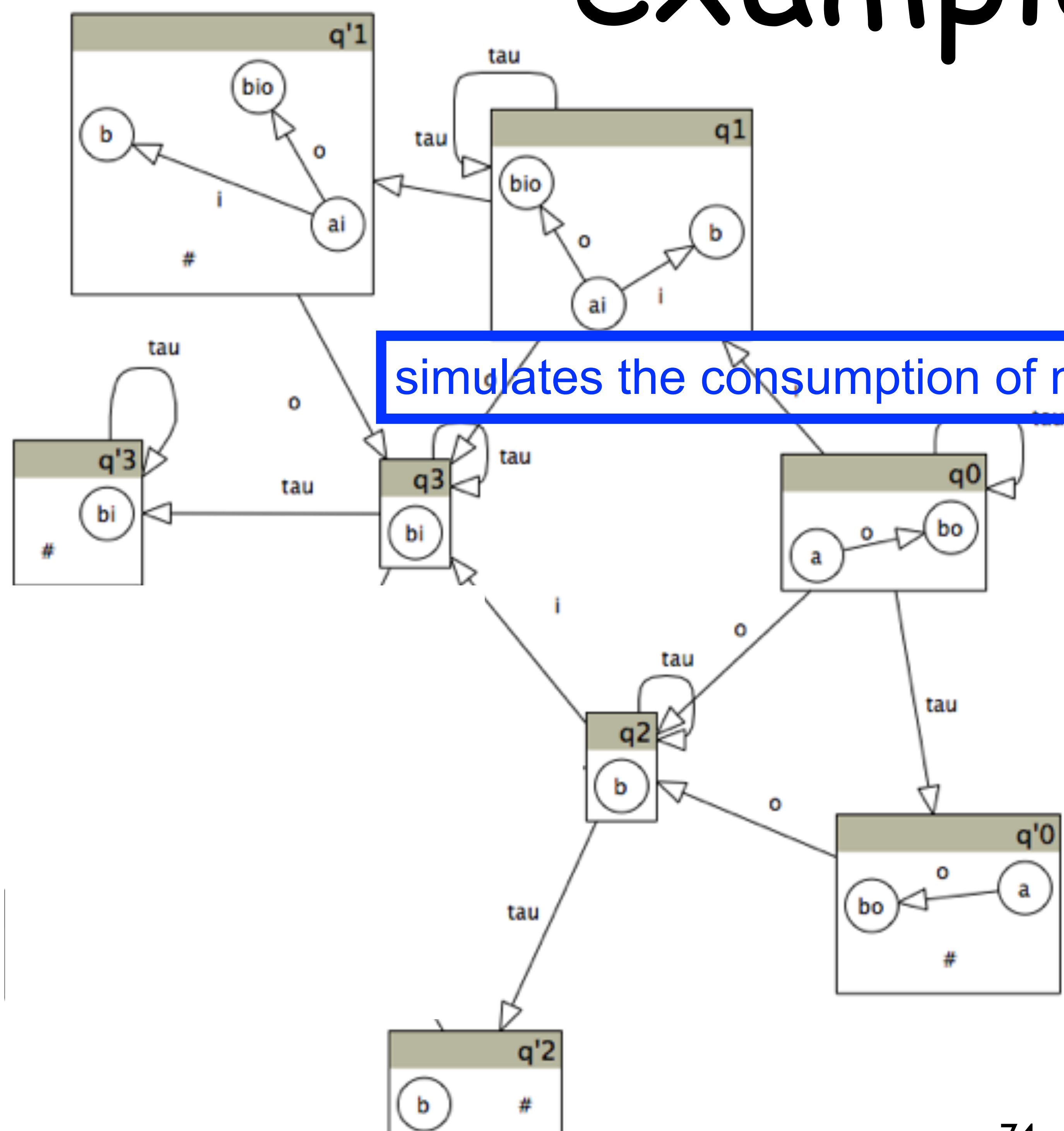


any state q has a final counterpart

N

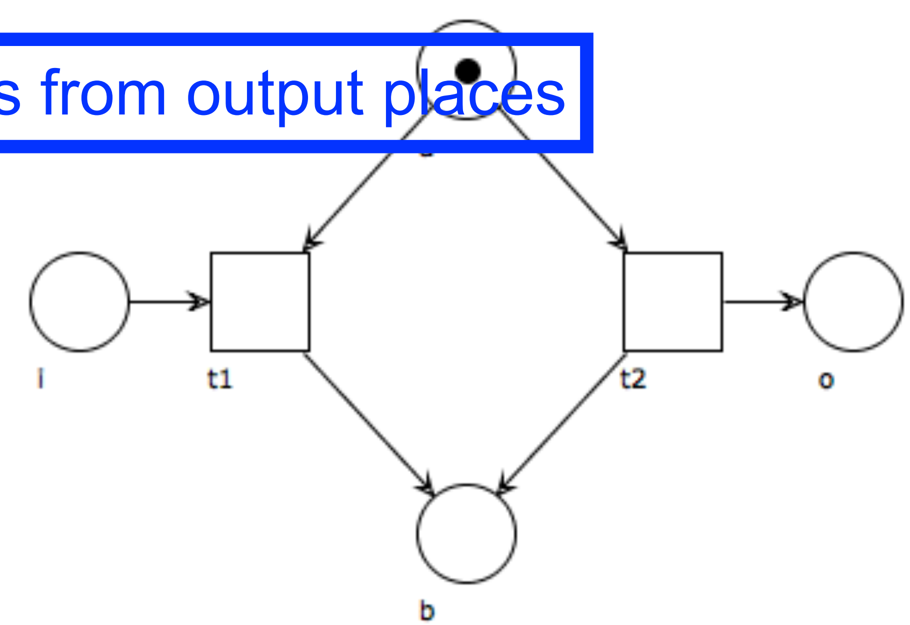


DF,1-controllability example: TS_1

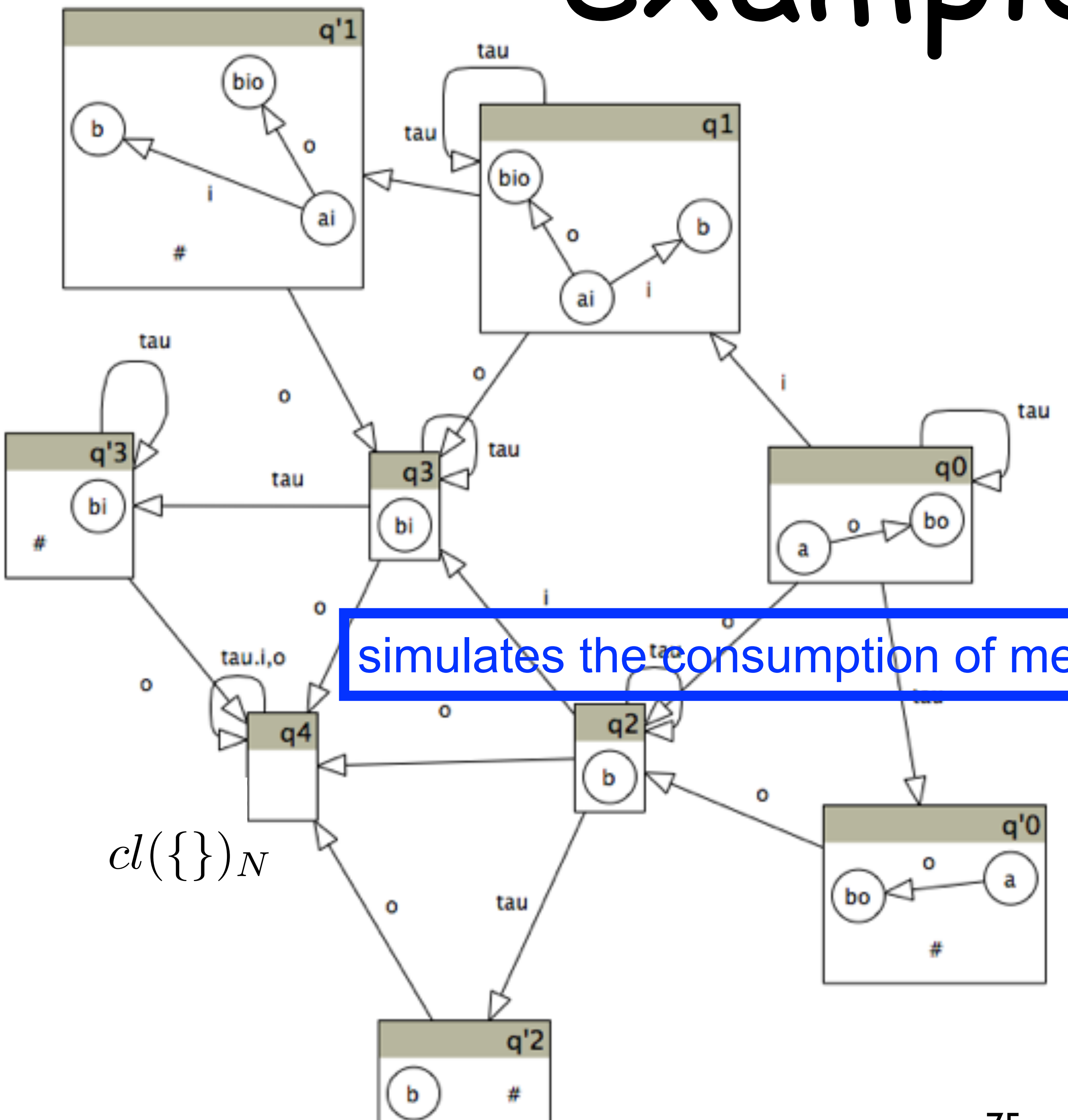


simulates the consumption of messages from output places

N

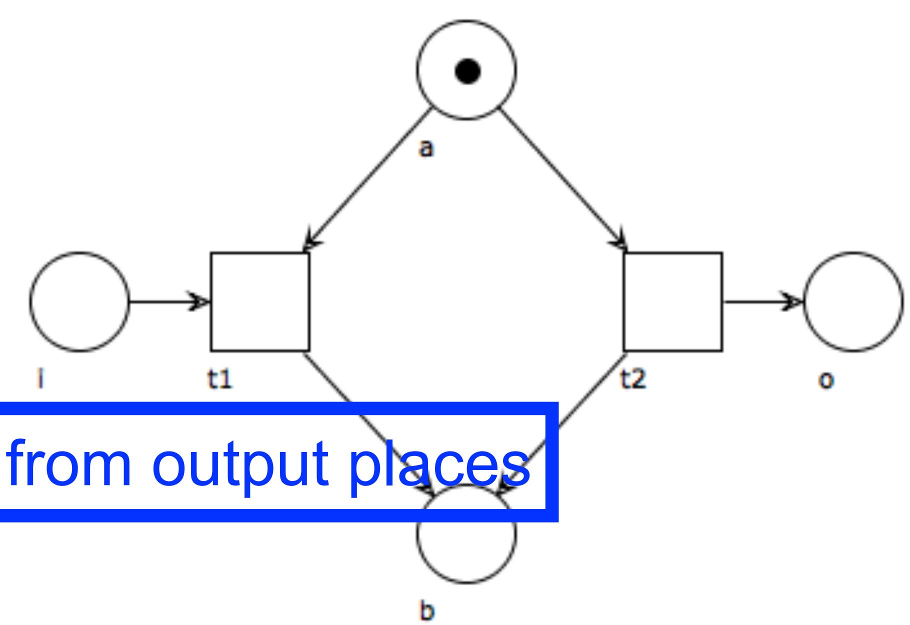


DF,1-controllability example: TS_1



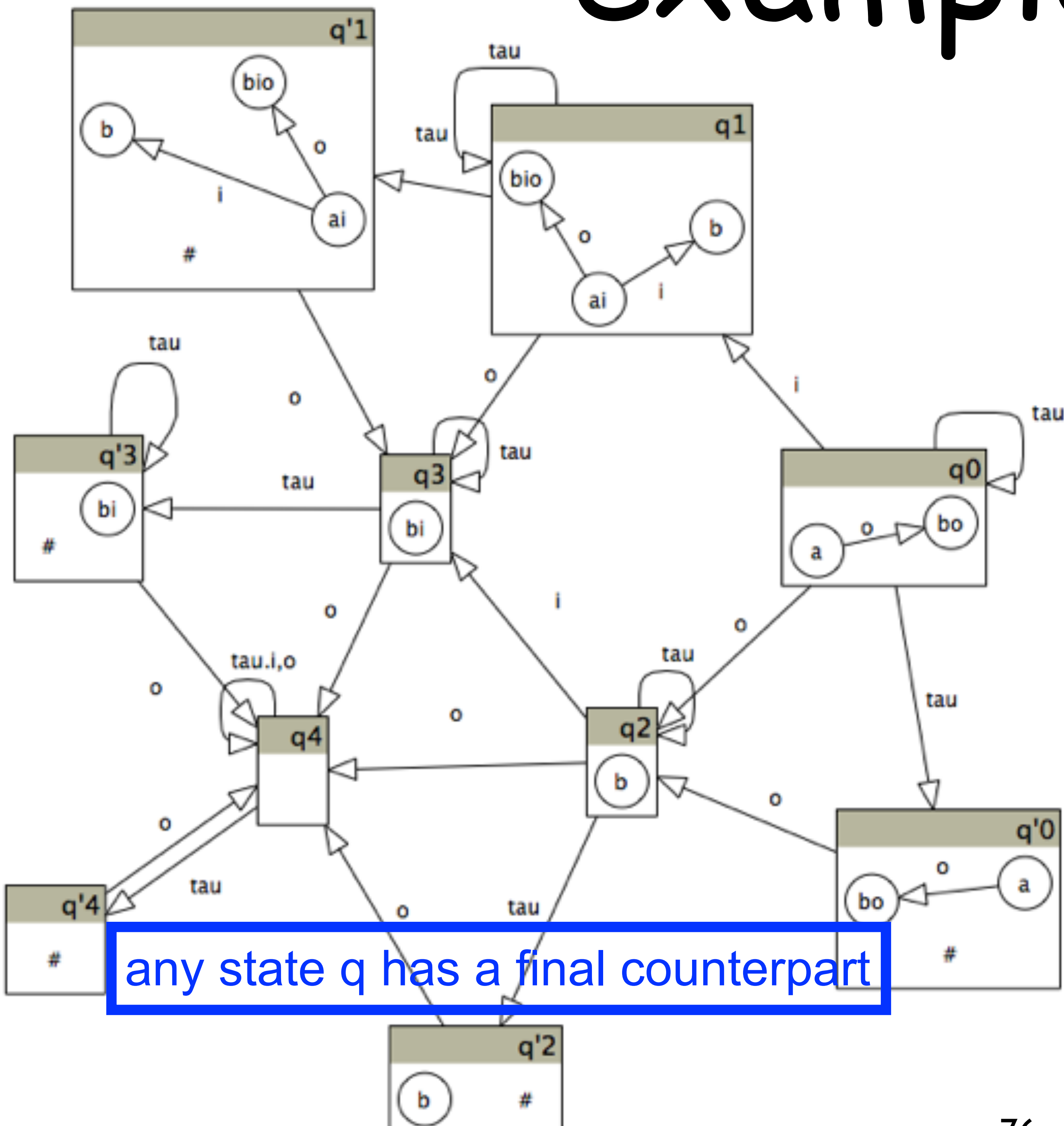
$cl(\{\})_N$

N

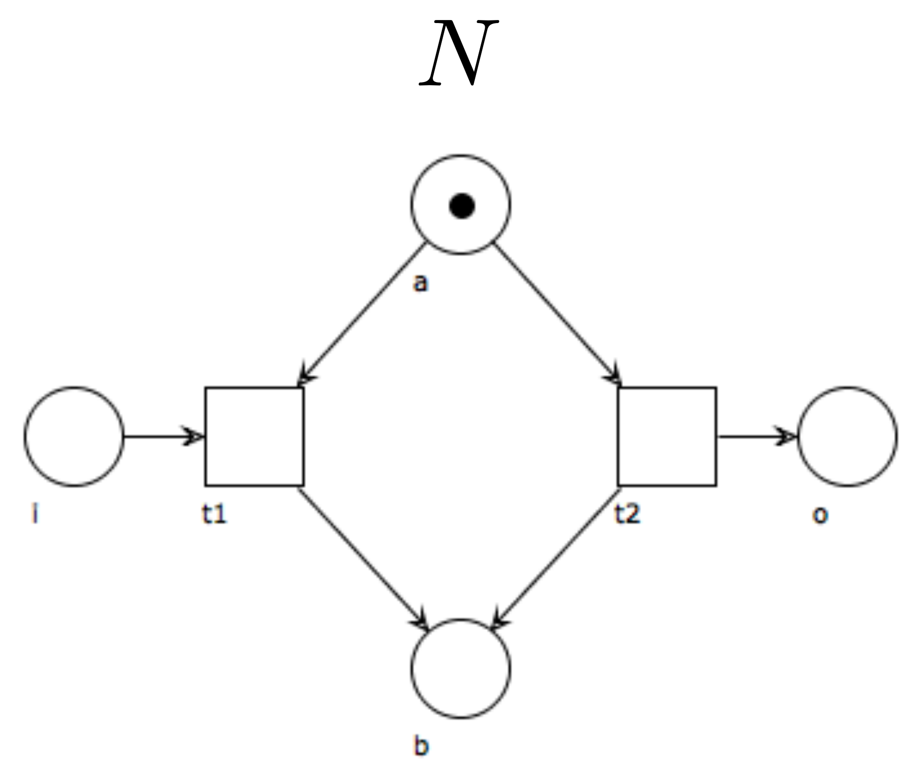


simulates the consumption of messages from output places

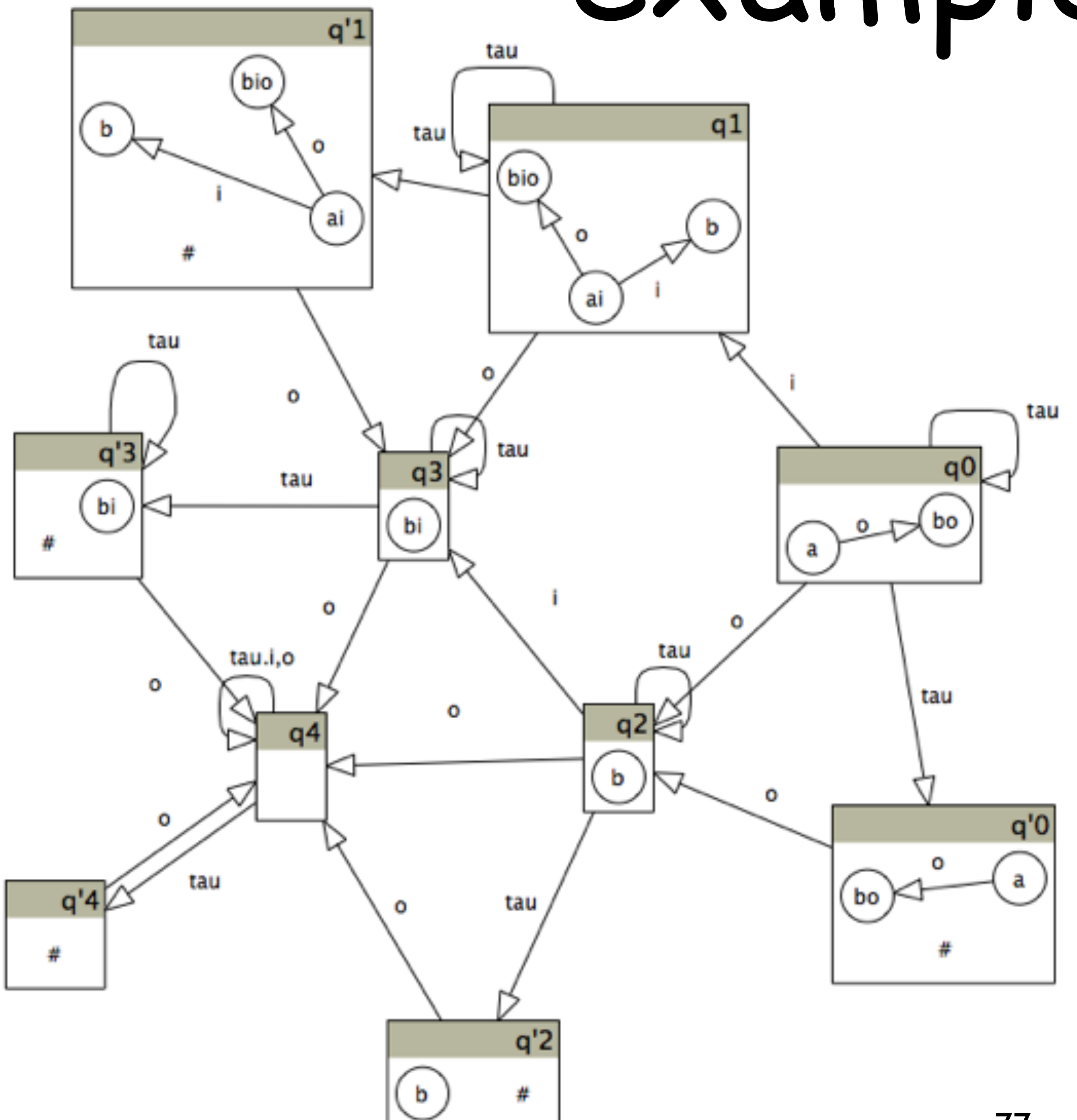
DF,1-controllability example: TS_1



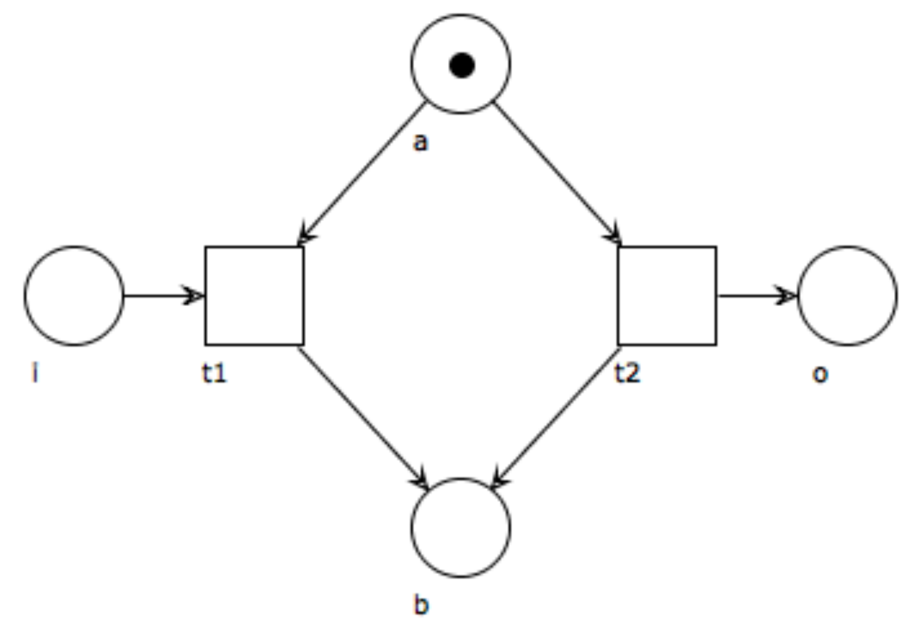
any state q has a final counterpart



DF,1-controllability example: TS_1



N



Approach

Starting from the strategy TS_i

Iterative step

Define a strategy $\mathbf{TS_{i+1}}$
that removes from TS_i all states q
that can invalidate the property of interest

(as a consequence remove all adjacent edges
and all states that become unreachable)

At some point $TS_{j+1} = TS_j$
and we terminate

Notation

We can compose TS_i with N, written $TS_i \oplus N$

States are pairs $[q, m]$ with $q \in Q_i$ and $m \in q$

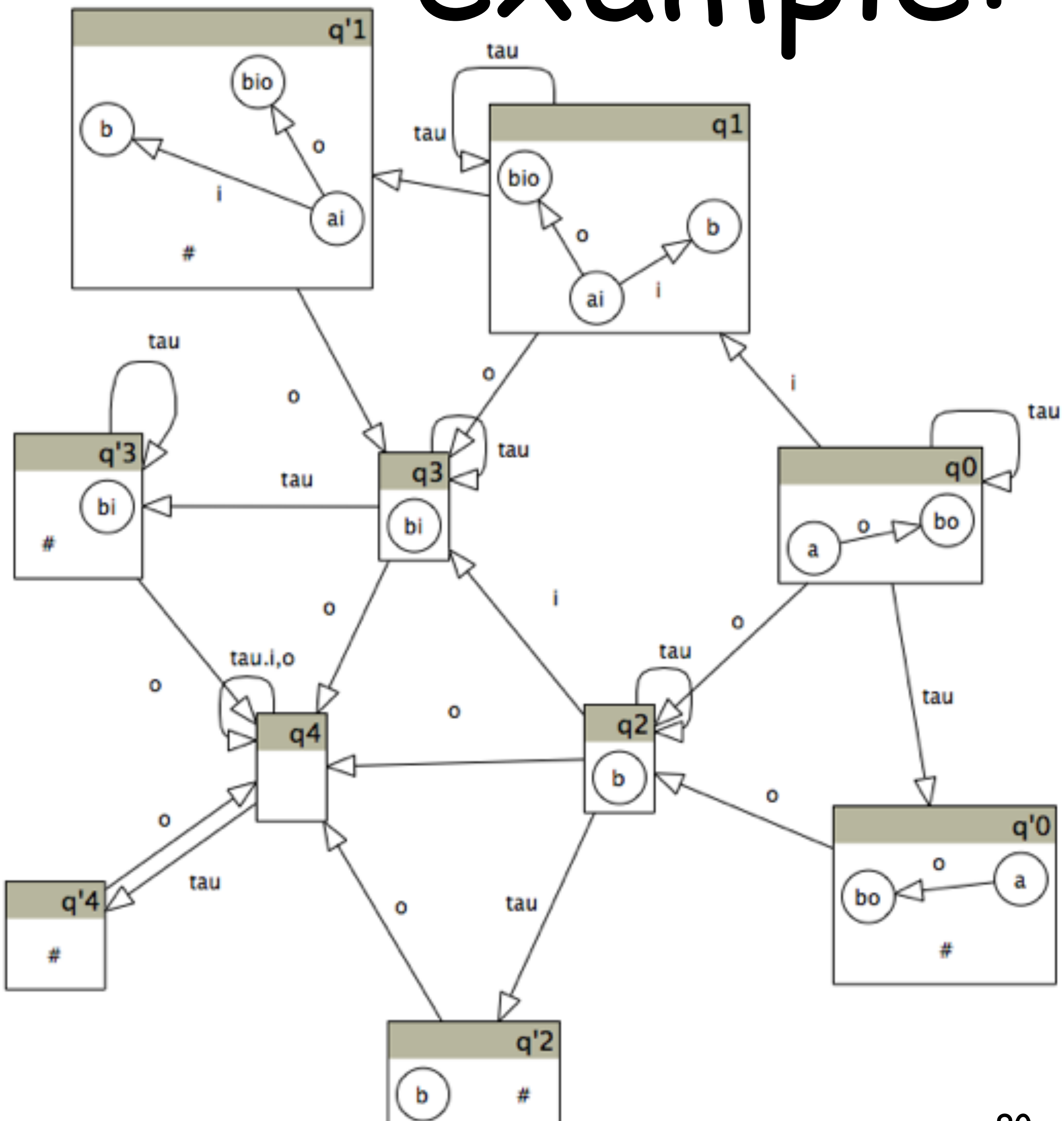
if $m \xrightarrow{t} m'$ in N then $[q, m] \xrightarrow{\ell(t)} [q, m']$

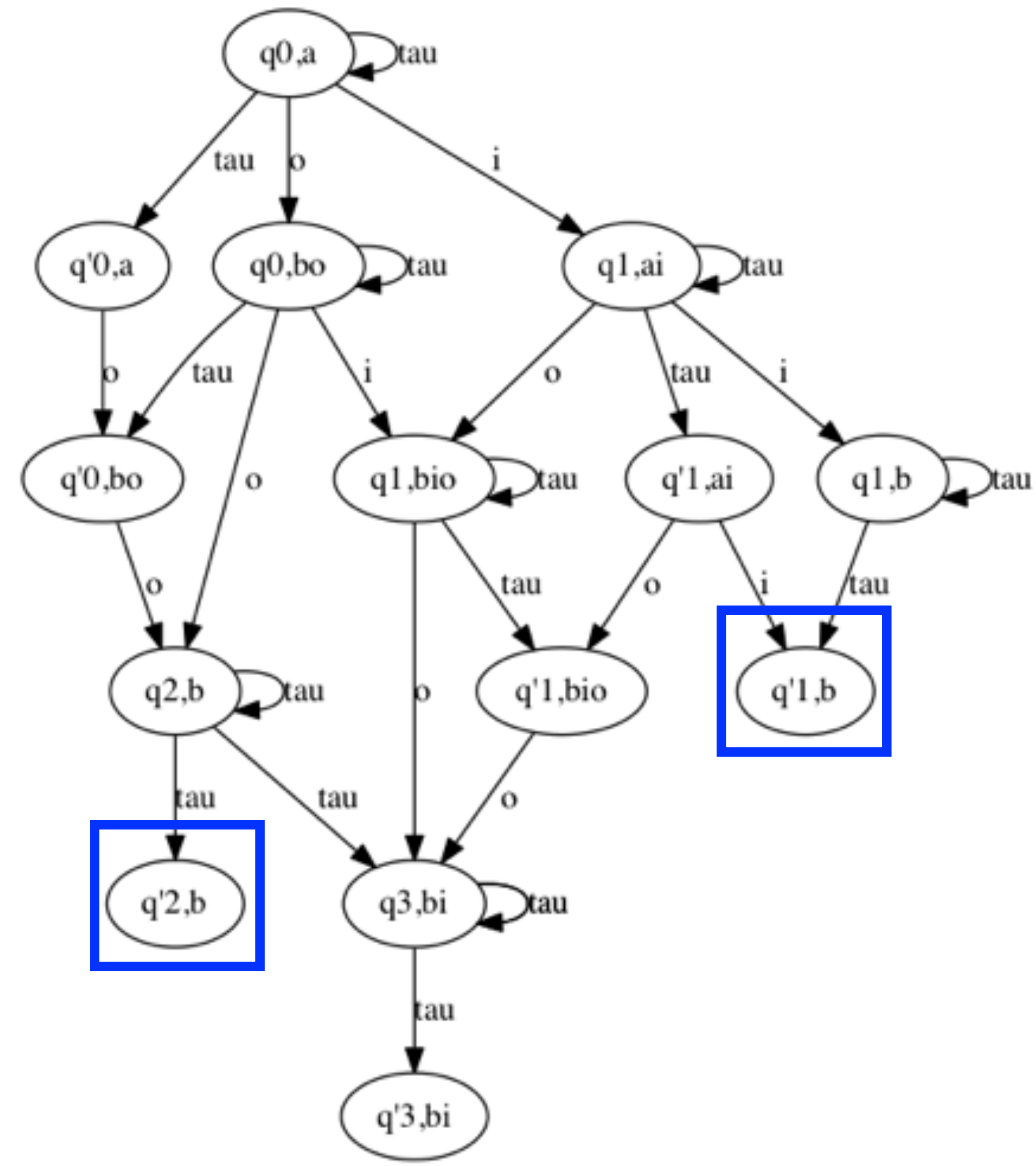
if $q \xrightarrow{\tau} q' \in E_i$ then $[q, m] \xrightarrow{\tau} [q', m]$

if $q \xrightarrow{x} q' \in E_i$ with $x \in P^I$ then $[q, m] \xrightarrow{x} [q', m + x]$

if $q \xrightarrow{x} q' \in E_i$ with $x \in P^O$ and $\boxed{m(x) > 0}$
then $[q, m] \xrightarrow{x} [q', m - x]$

DF,1-controllability example: $TS_1 \oplus N$





TS_{i+1}

$TS_i = (Q_i, E_i, q_0, Q_{\mathrm{f}i})$

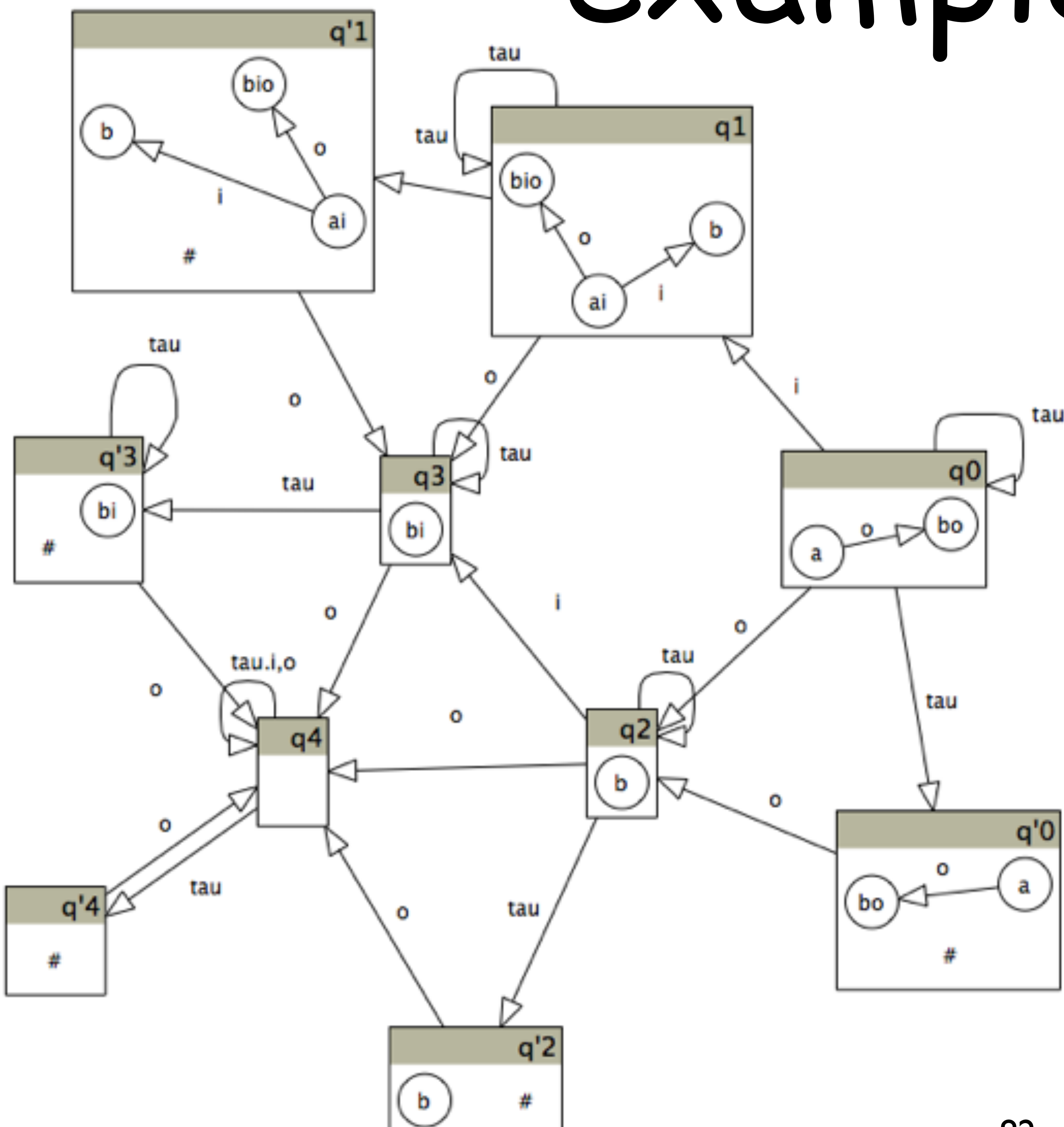
remove the state q if $q \xrightarrow{\tau} q$ is the only edge with source q

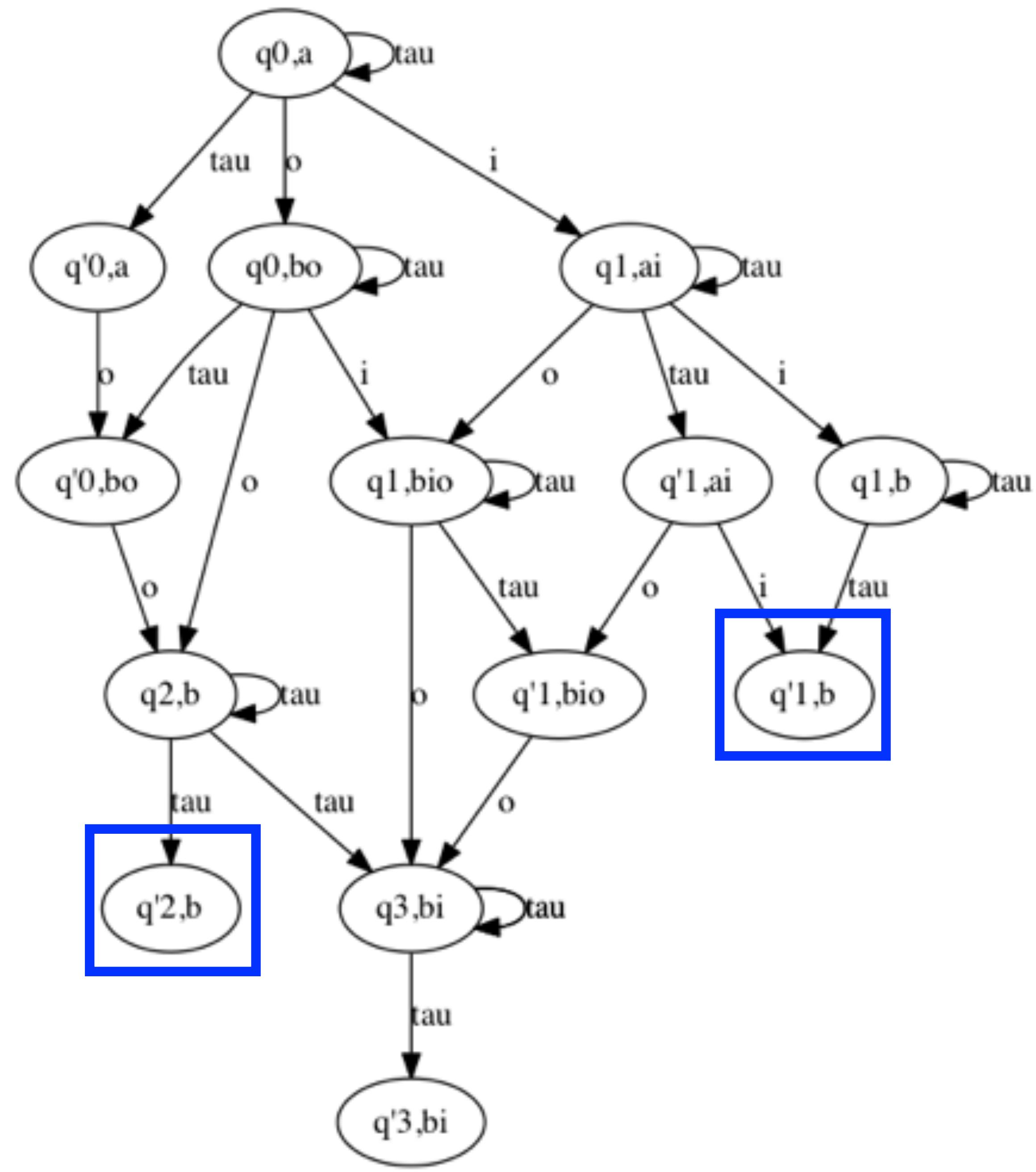
remove the state q if $\exists m \in q$ such that in $TS_i \oplus N$

if $[q', m']$ is reachable from $[q, m]$ then $m' \notin M_{\mathrm{f}}$

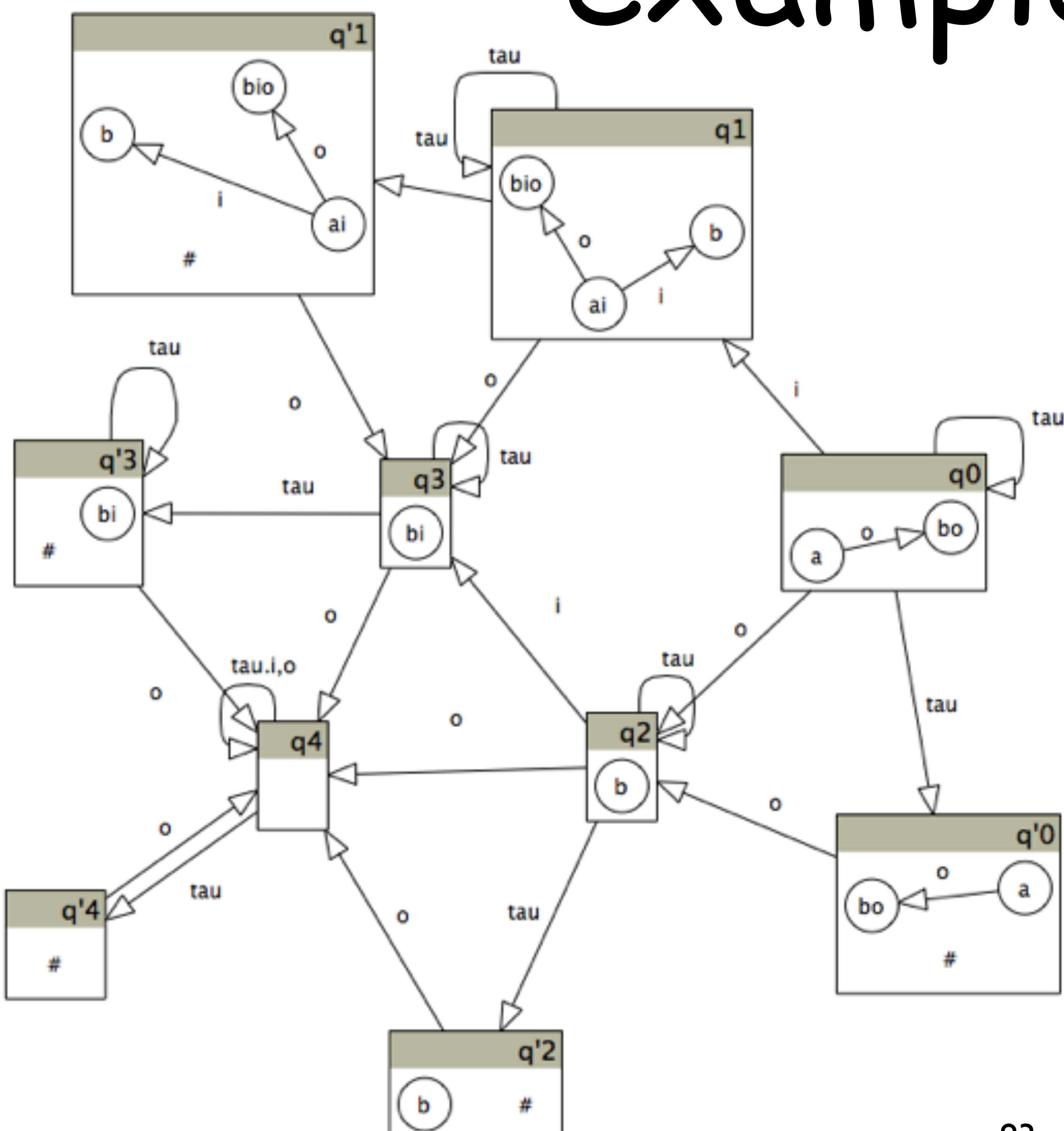
and only sequences of τ-steps are possible from $[q, m]$

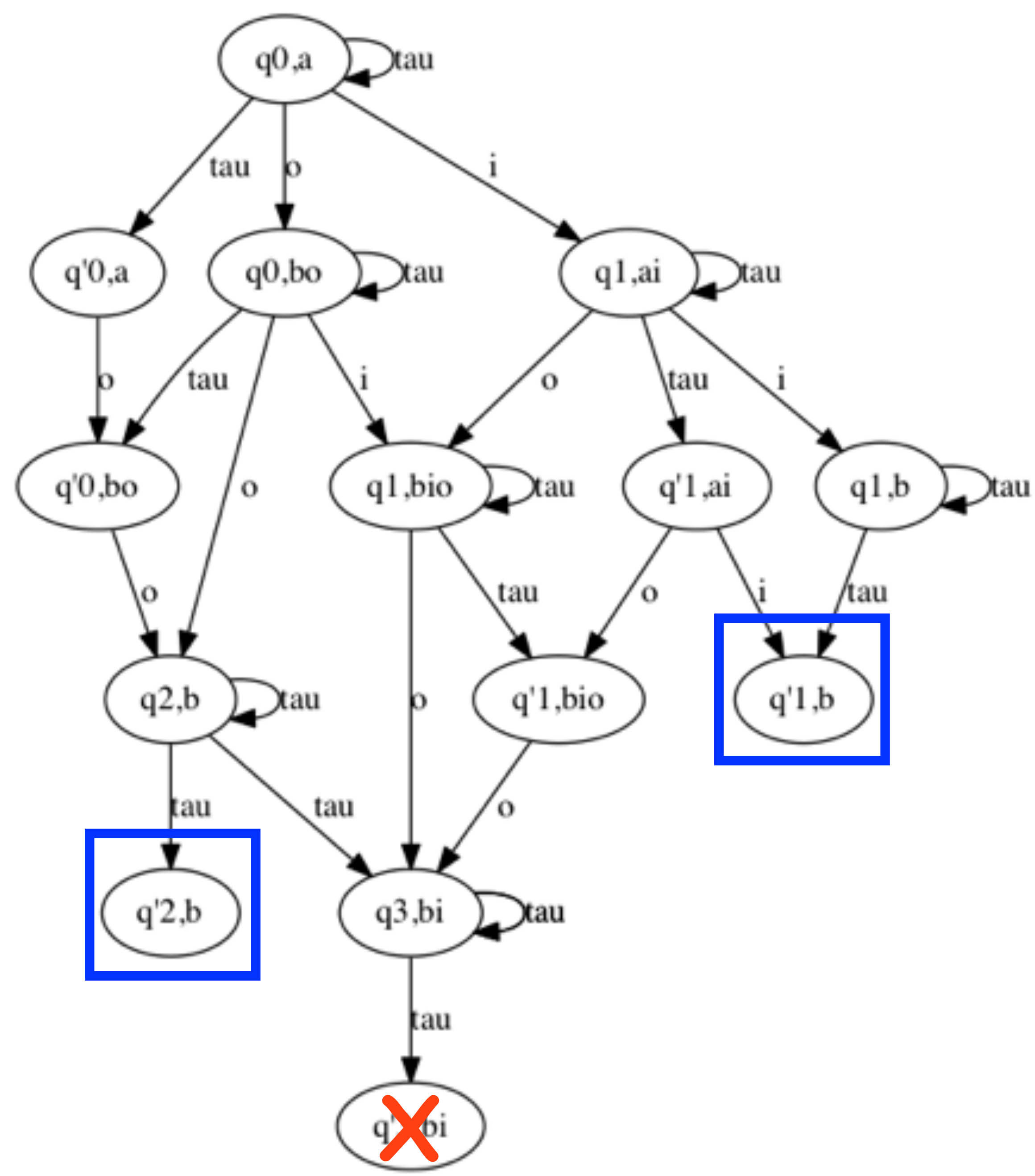
DF,1-controllability example: TS_2



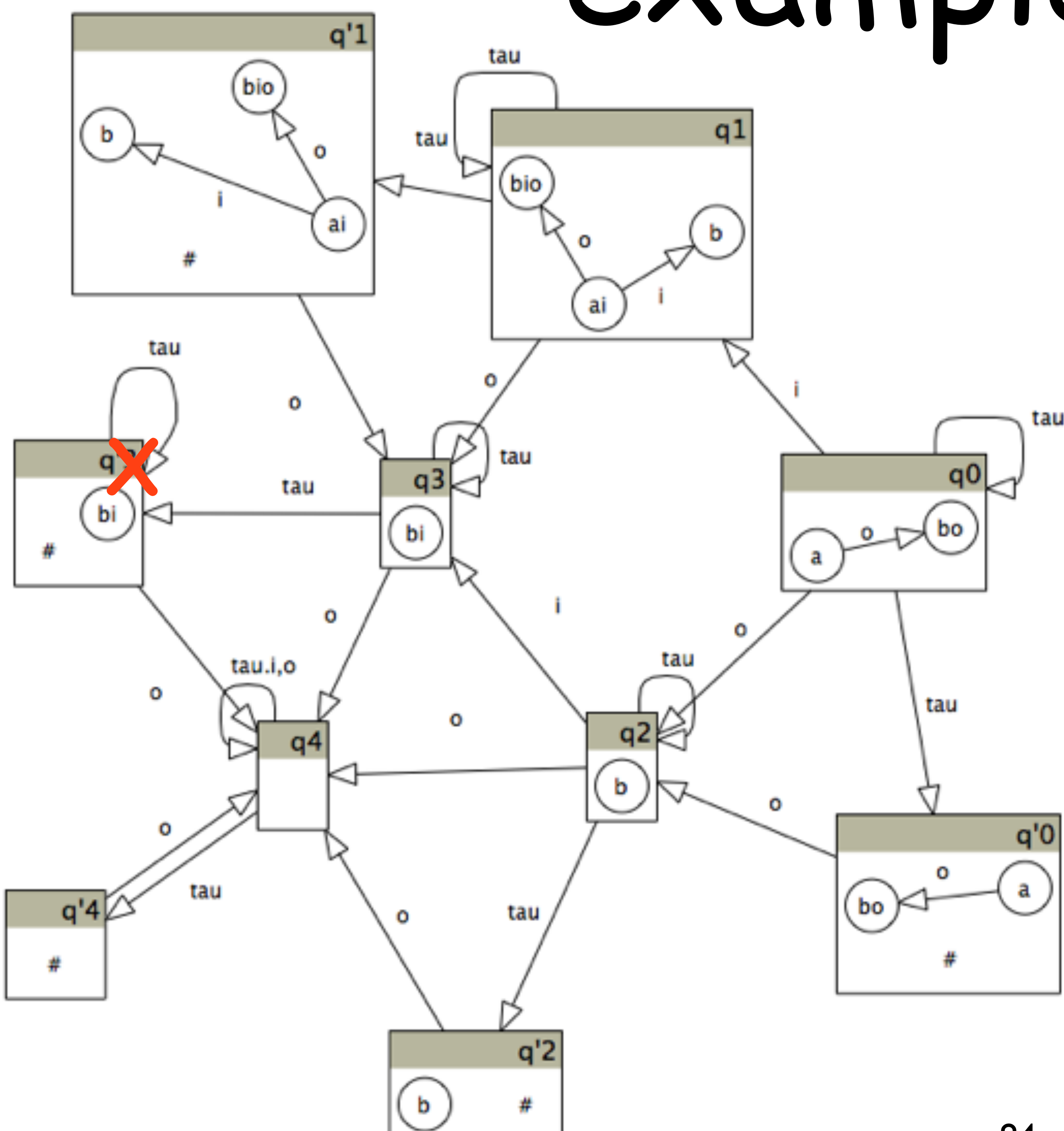


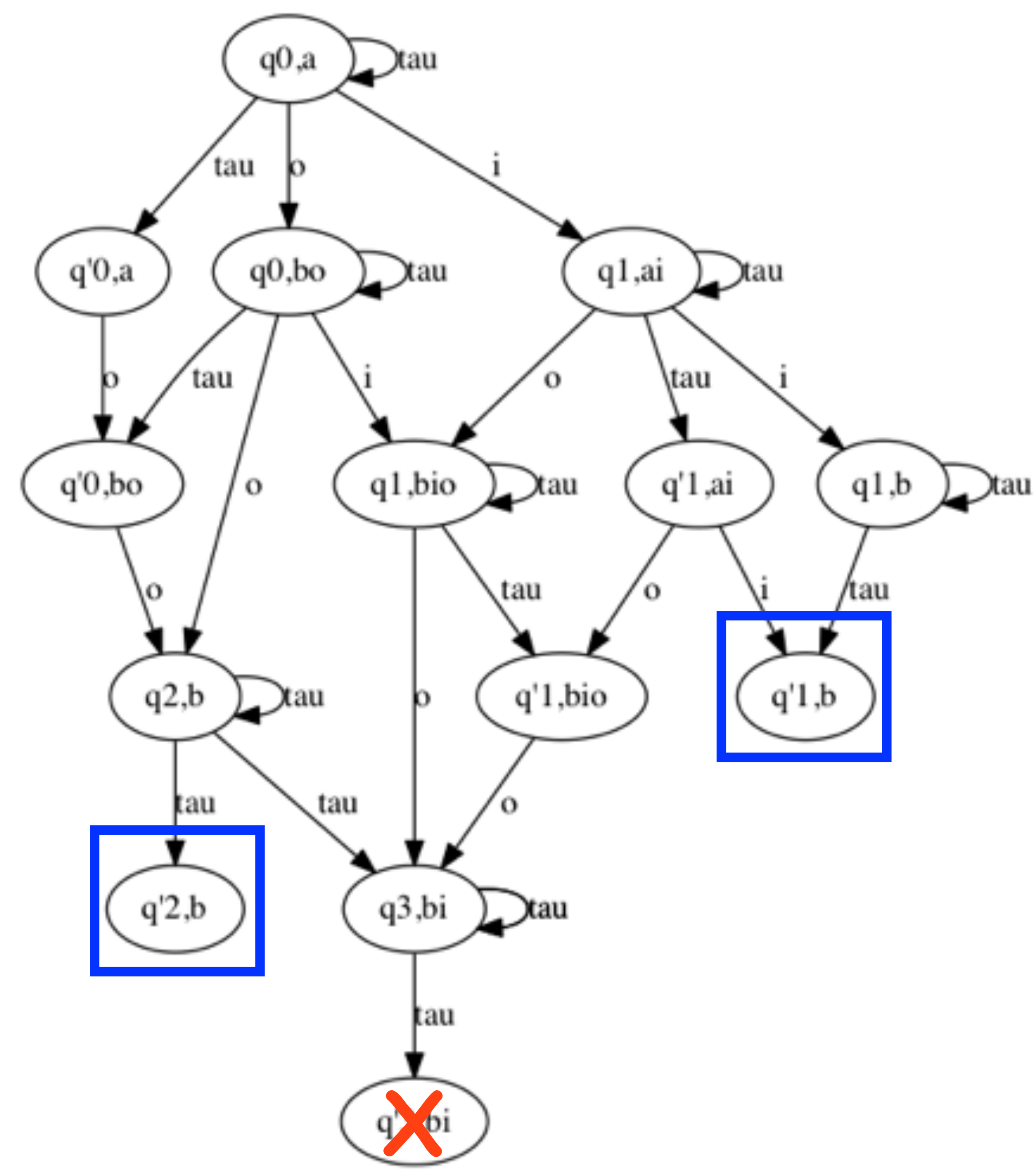
DF,1-controllability example: TS_2



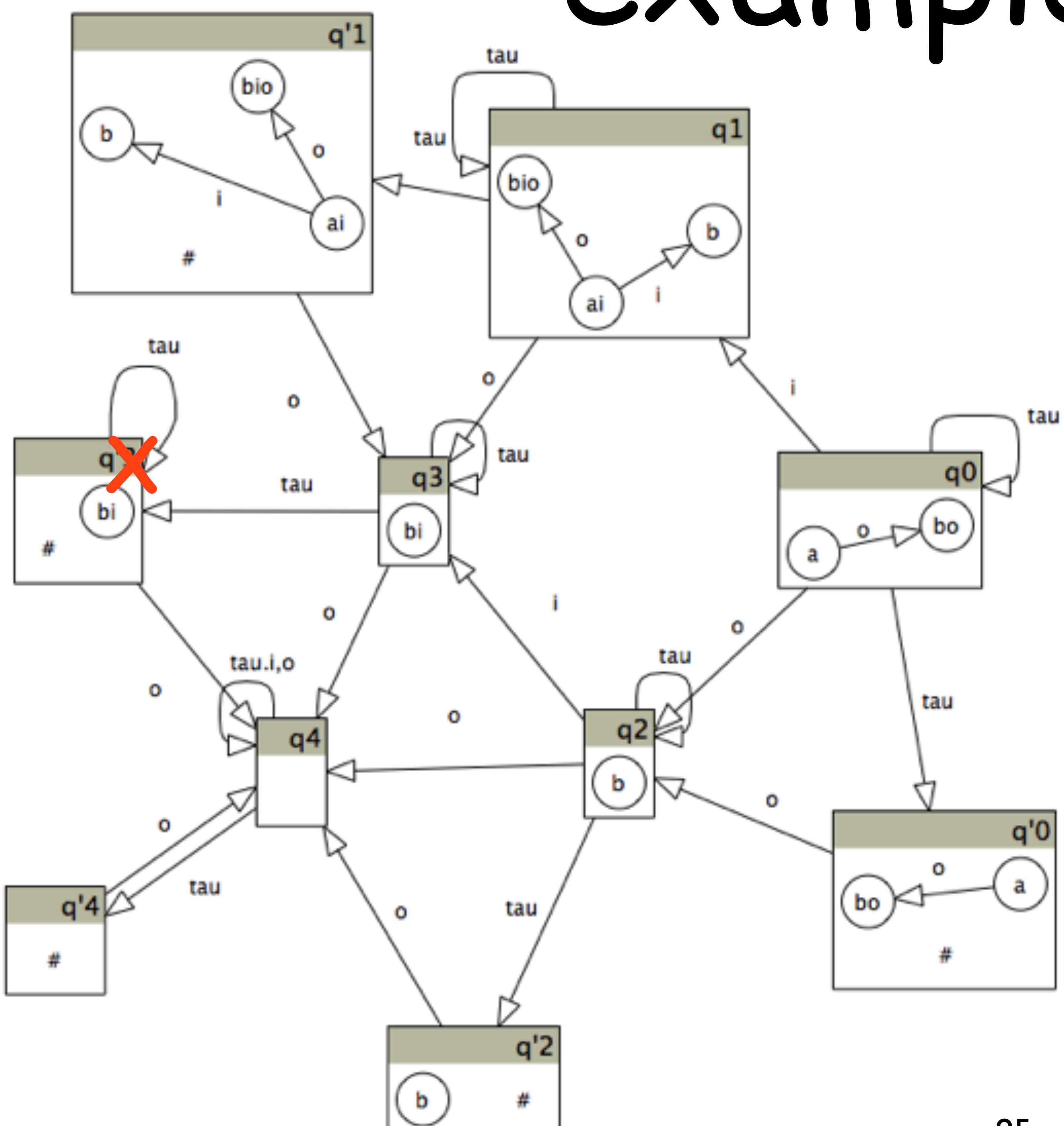


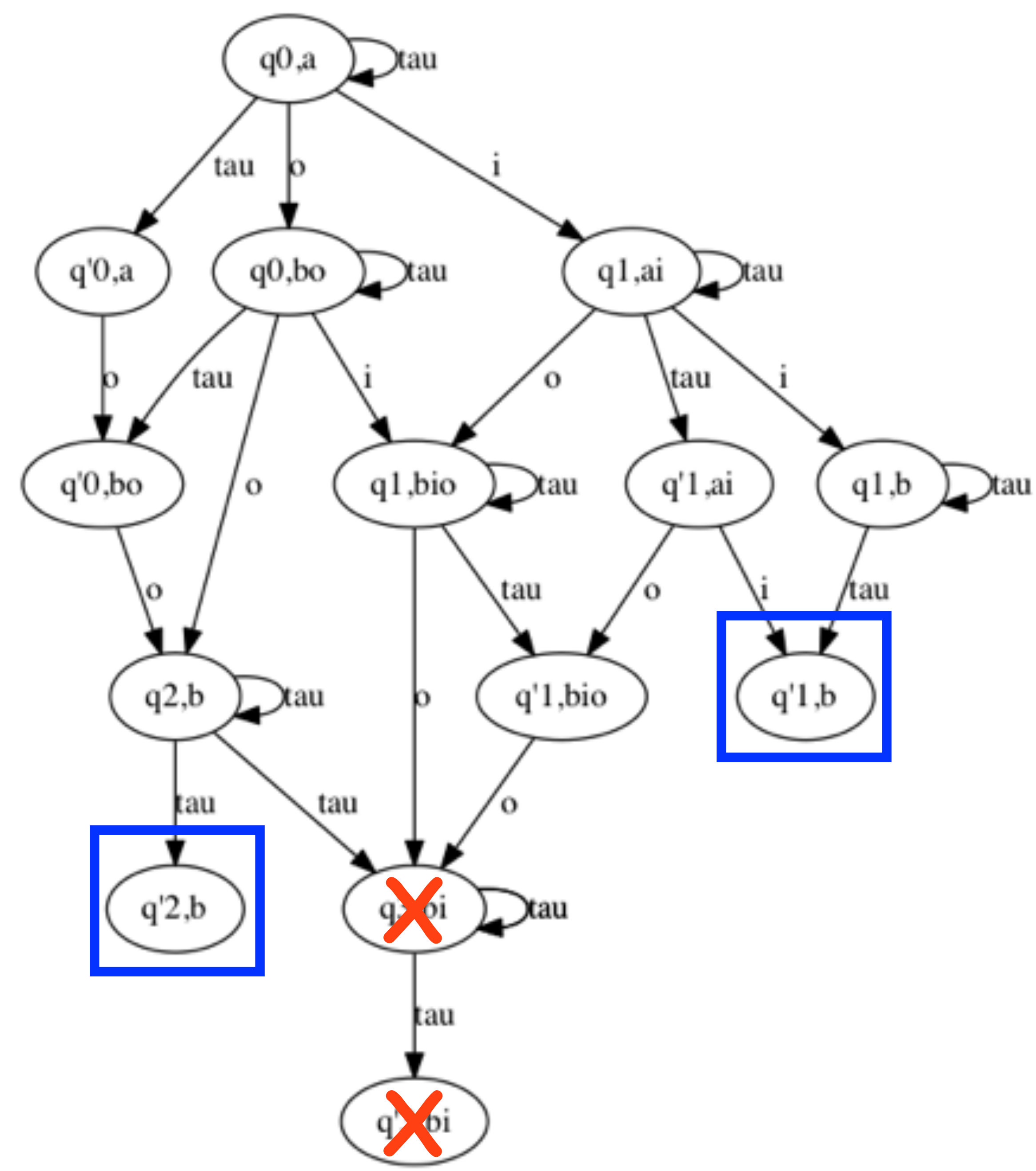
DF,1-controllability example: TS_2



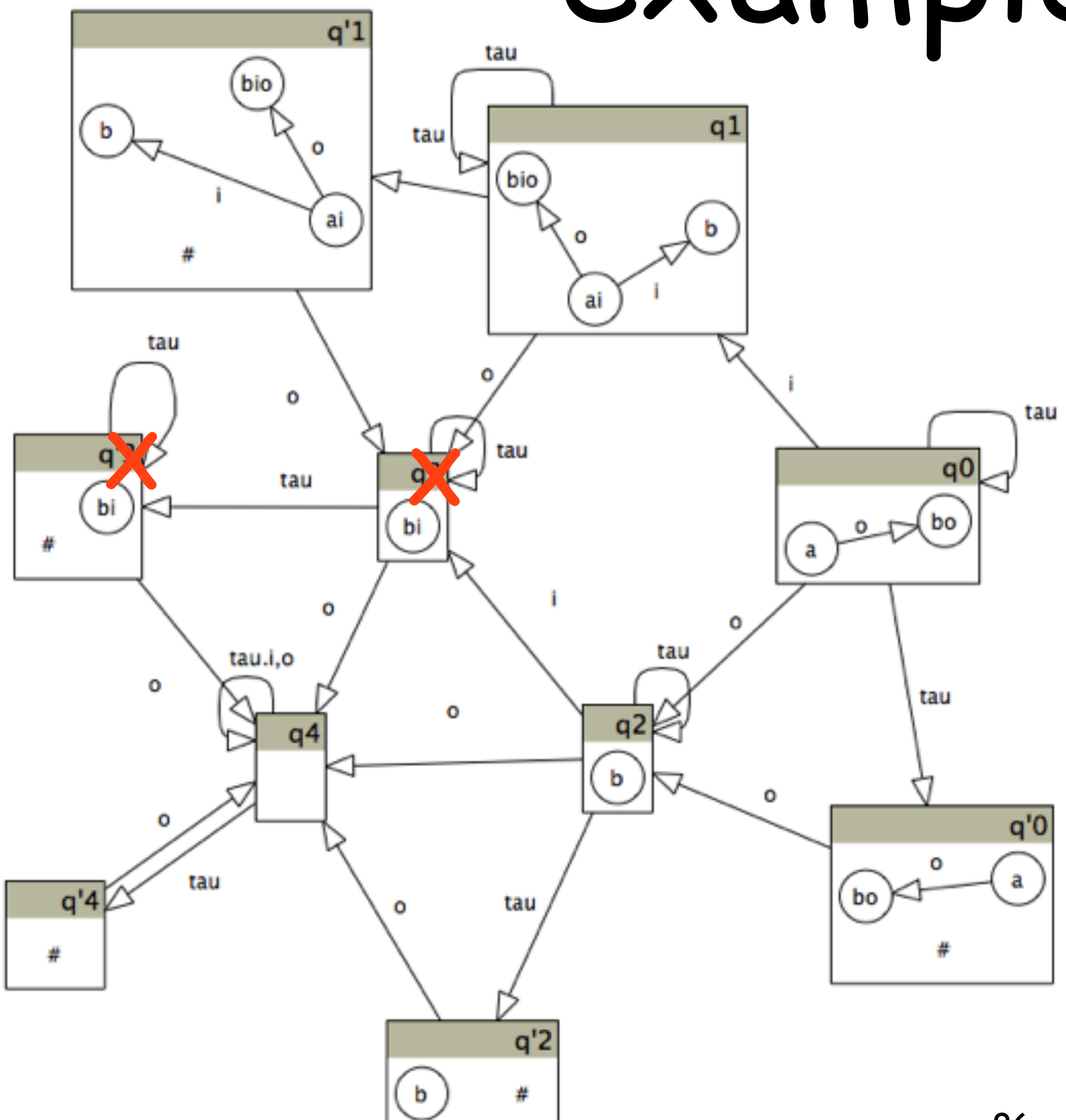


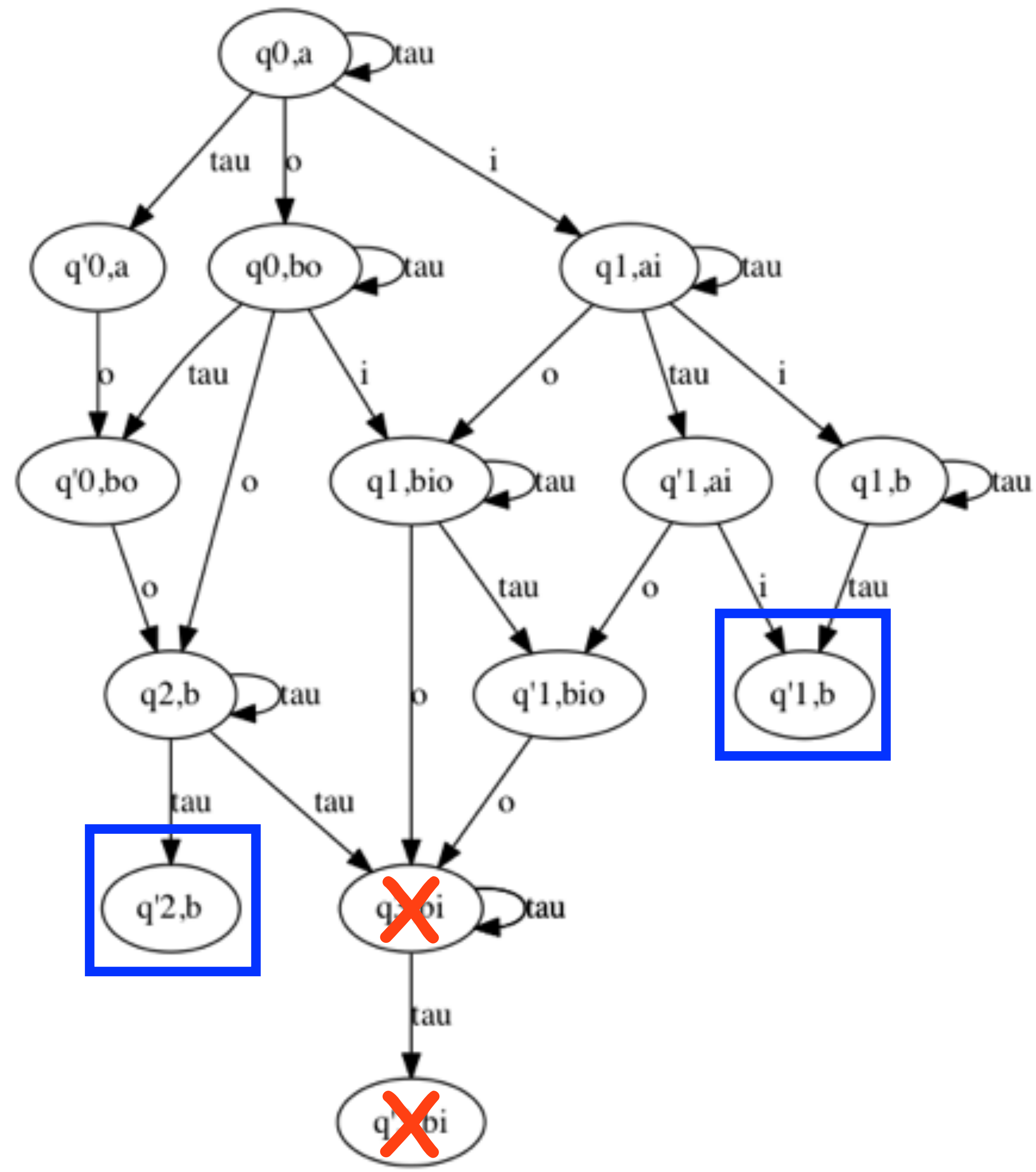
DF,1-controllability example: TS_2



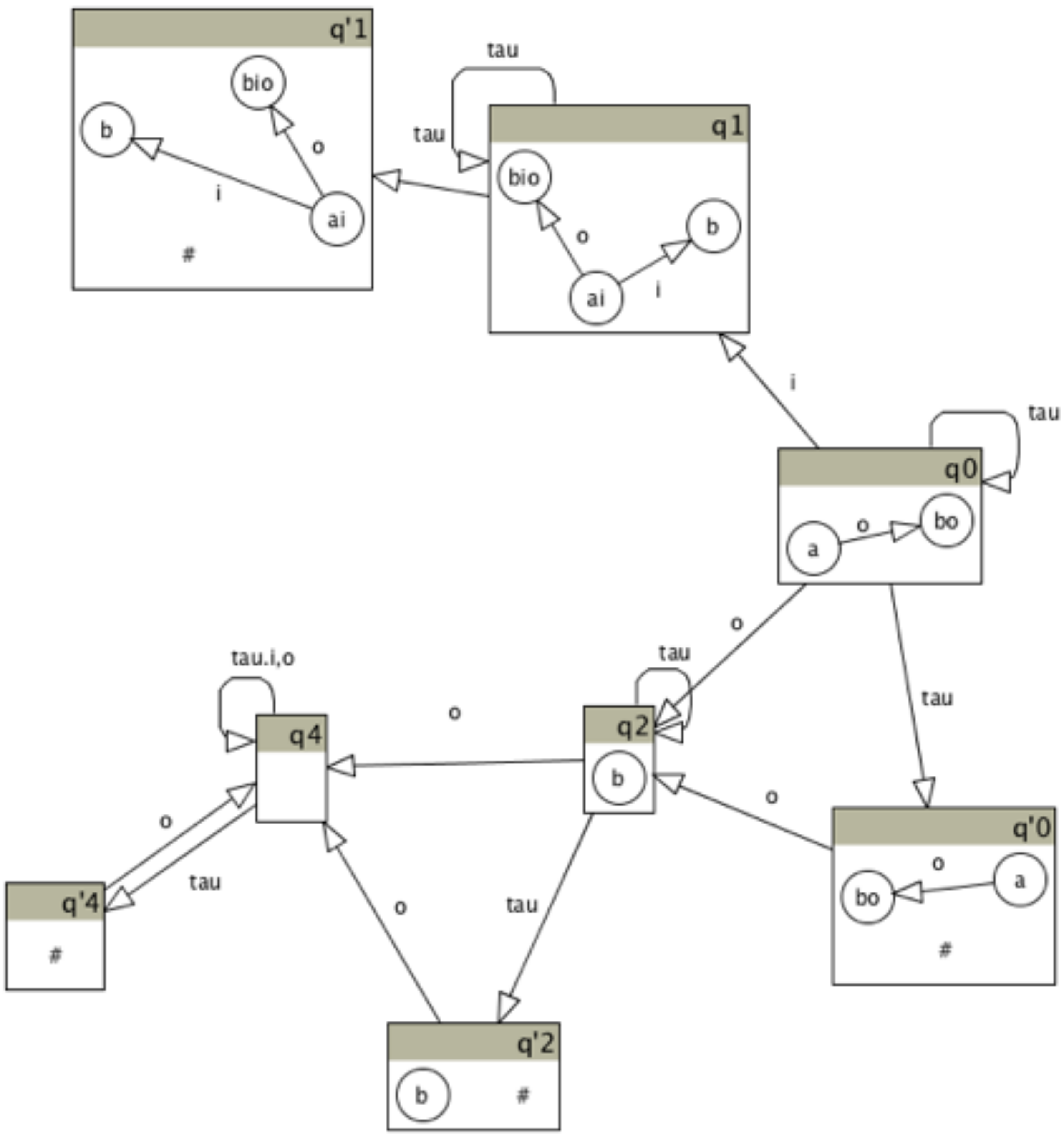


DF,1-controllability example: TS_2

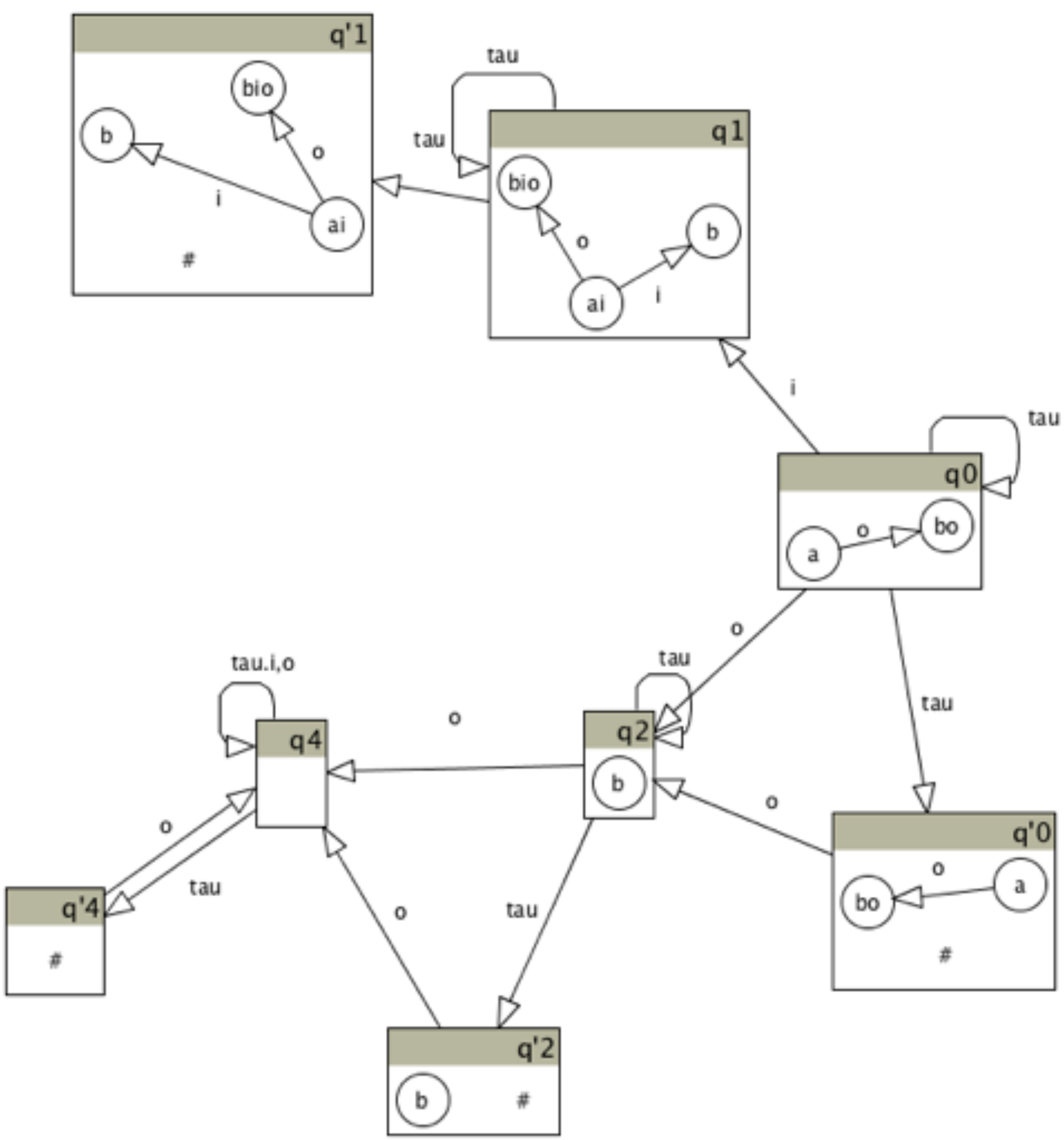


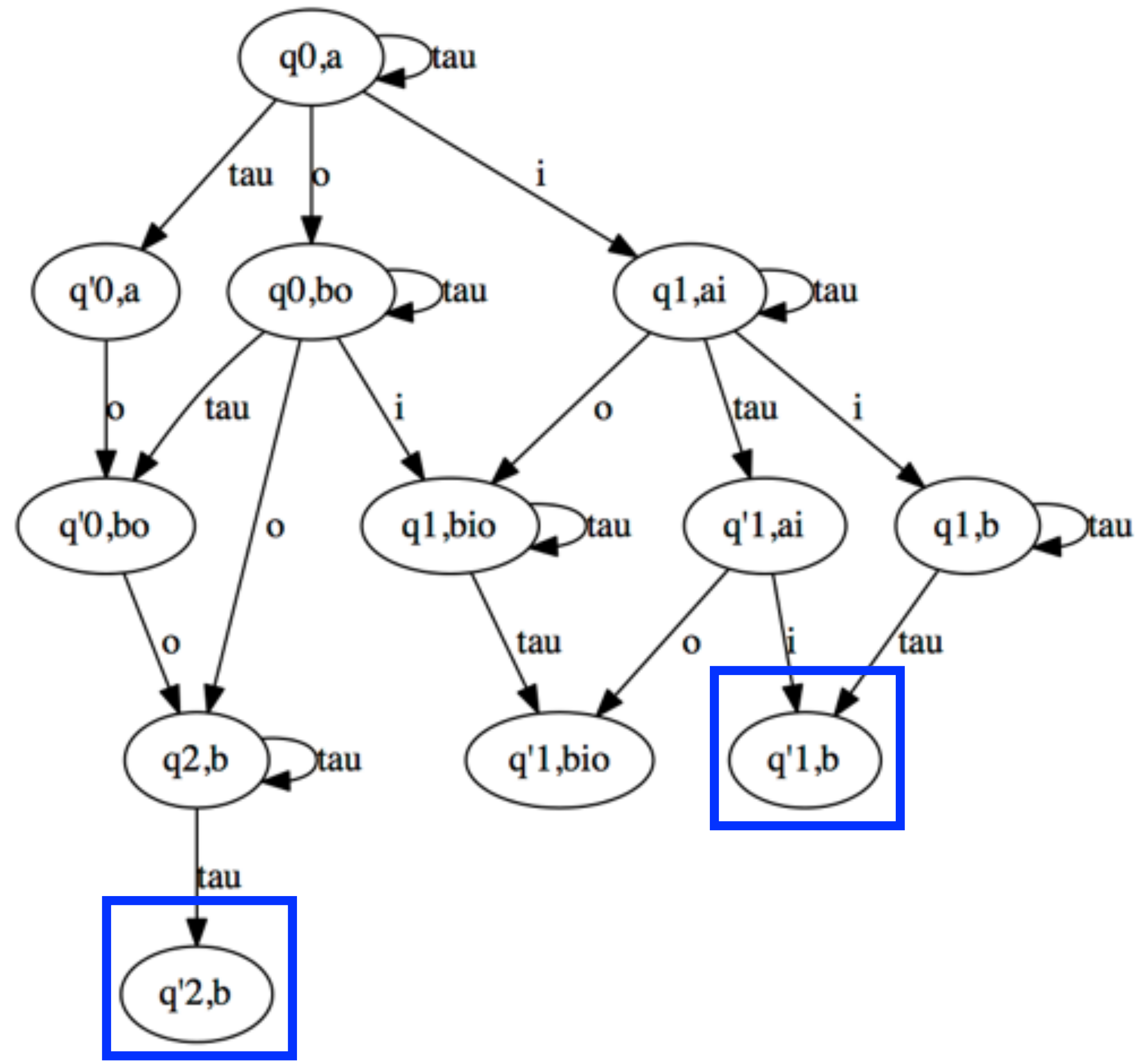


DF,1-controllability example: TS_2

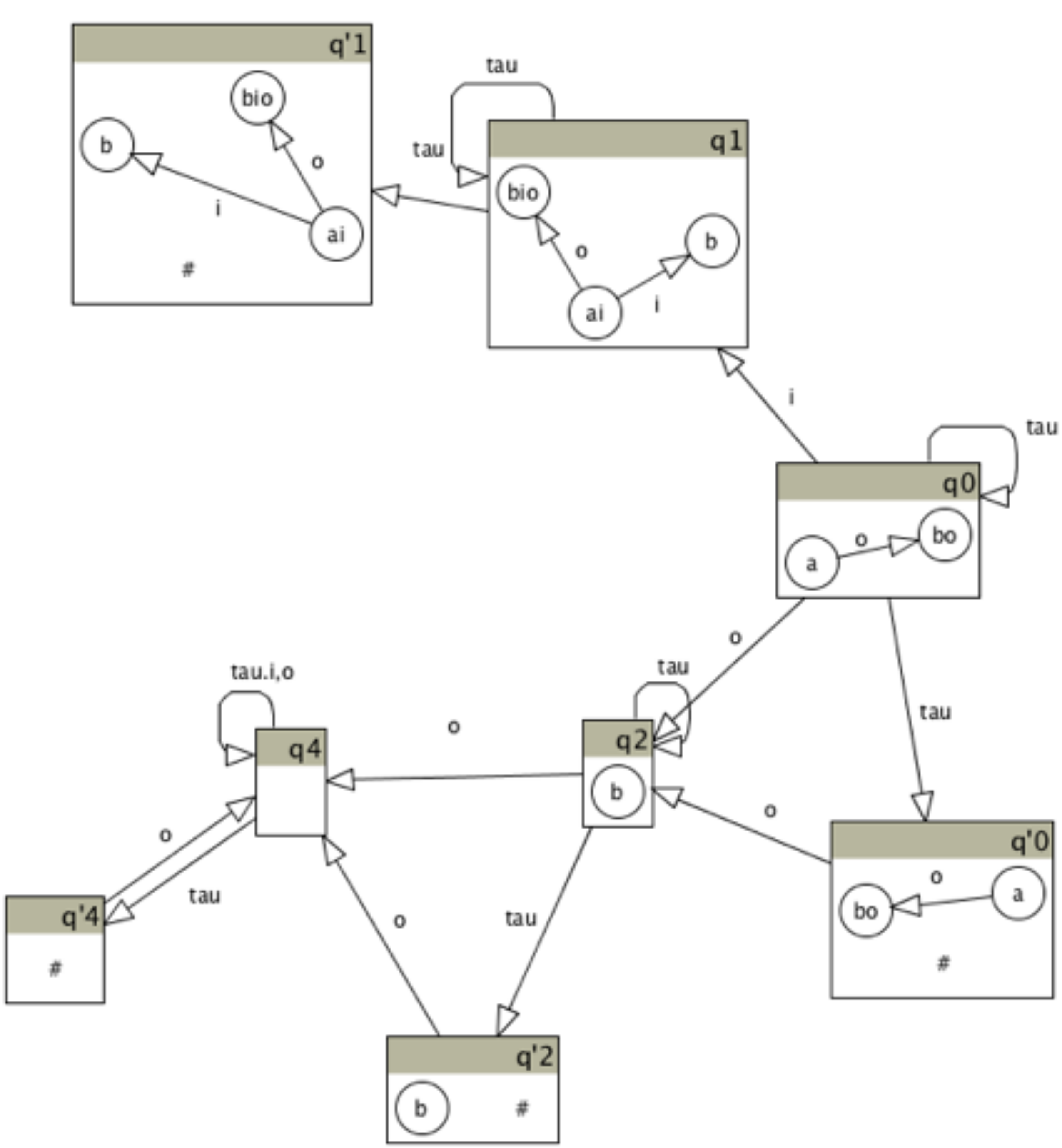


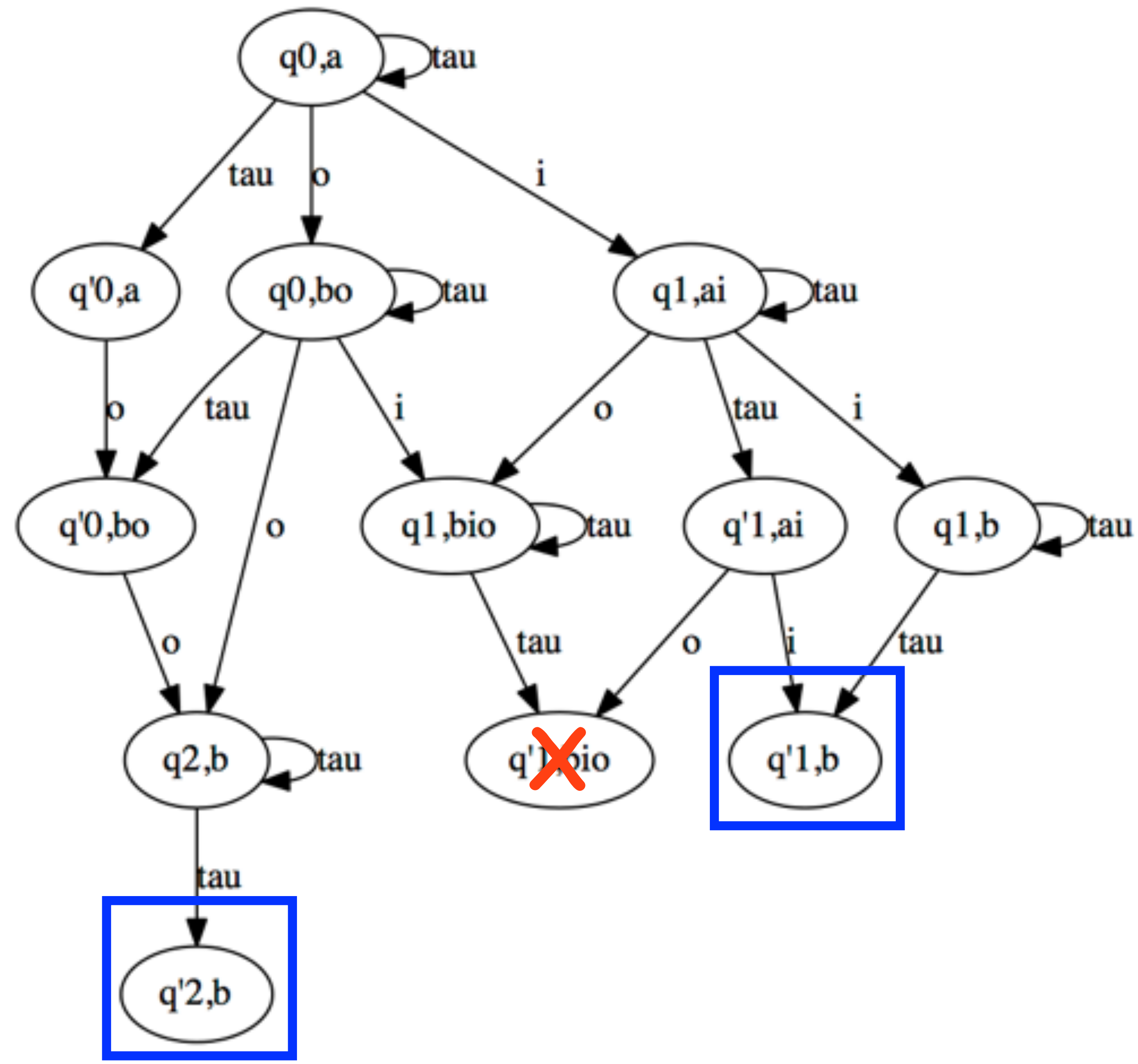
DF,1-controllability example: $TS_2 \oplus N$



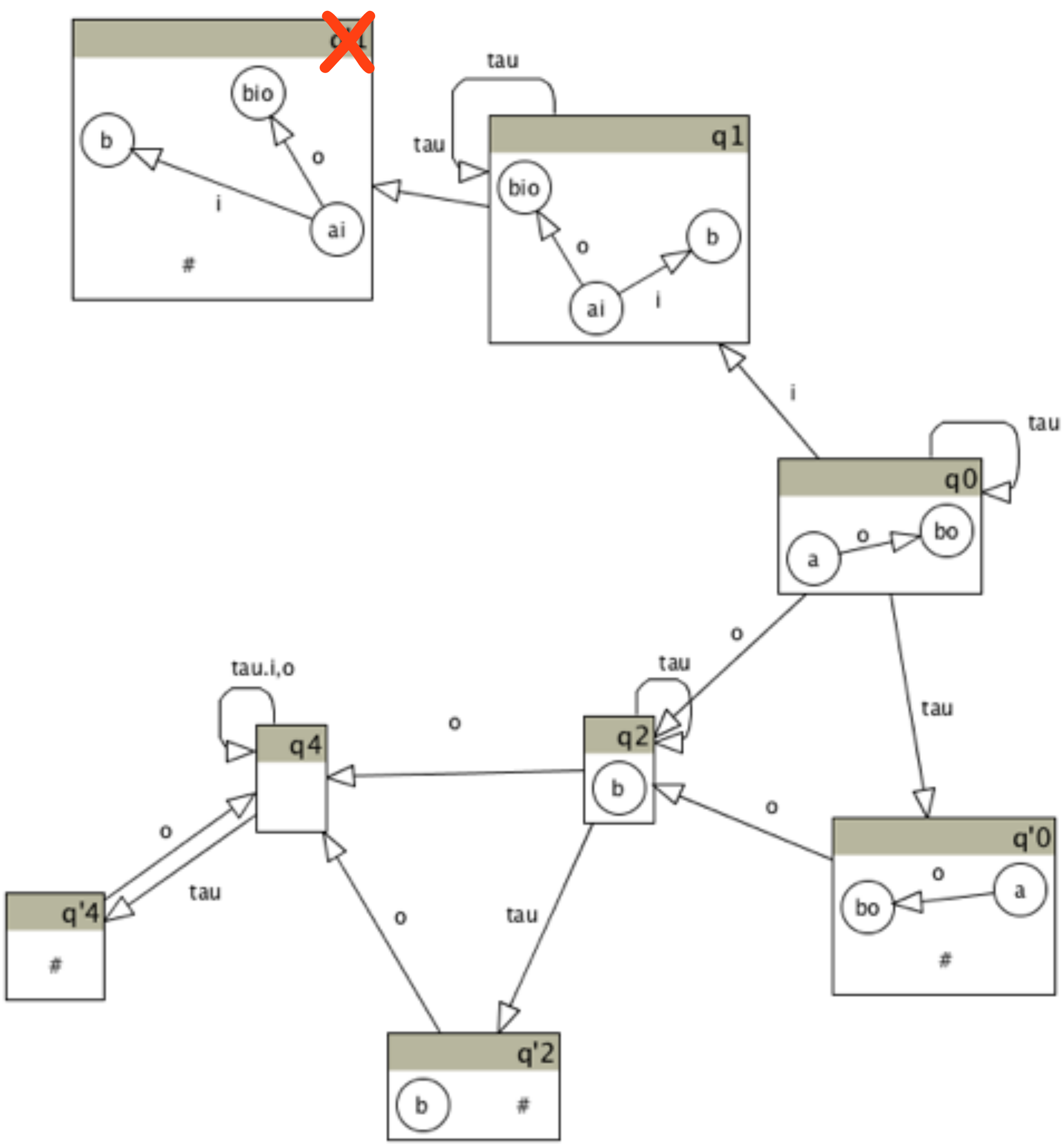


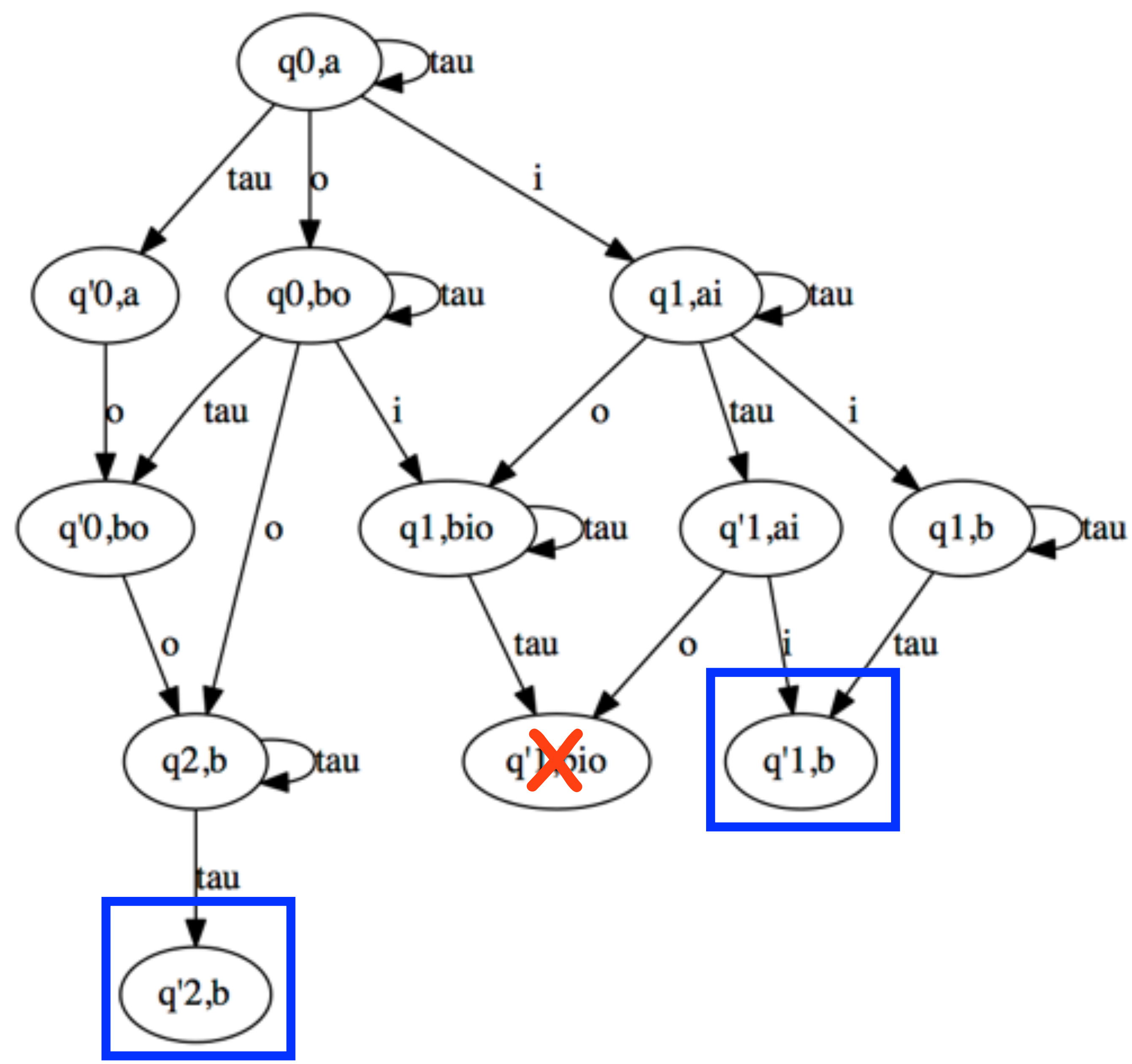
DF,1-controllability example: $TS_2 \oplus N$



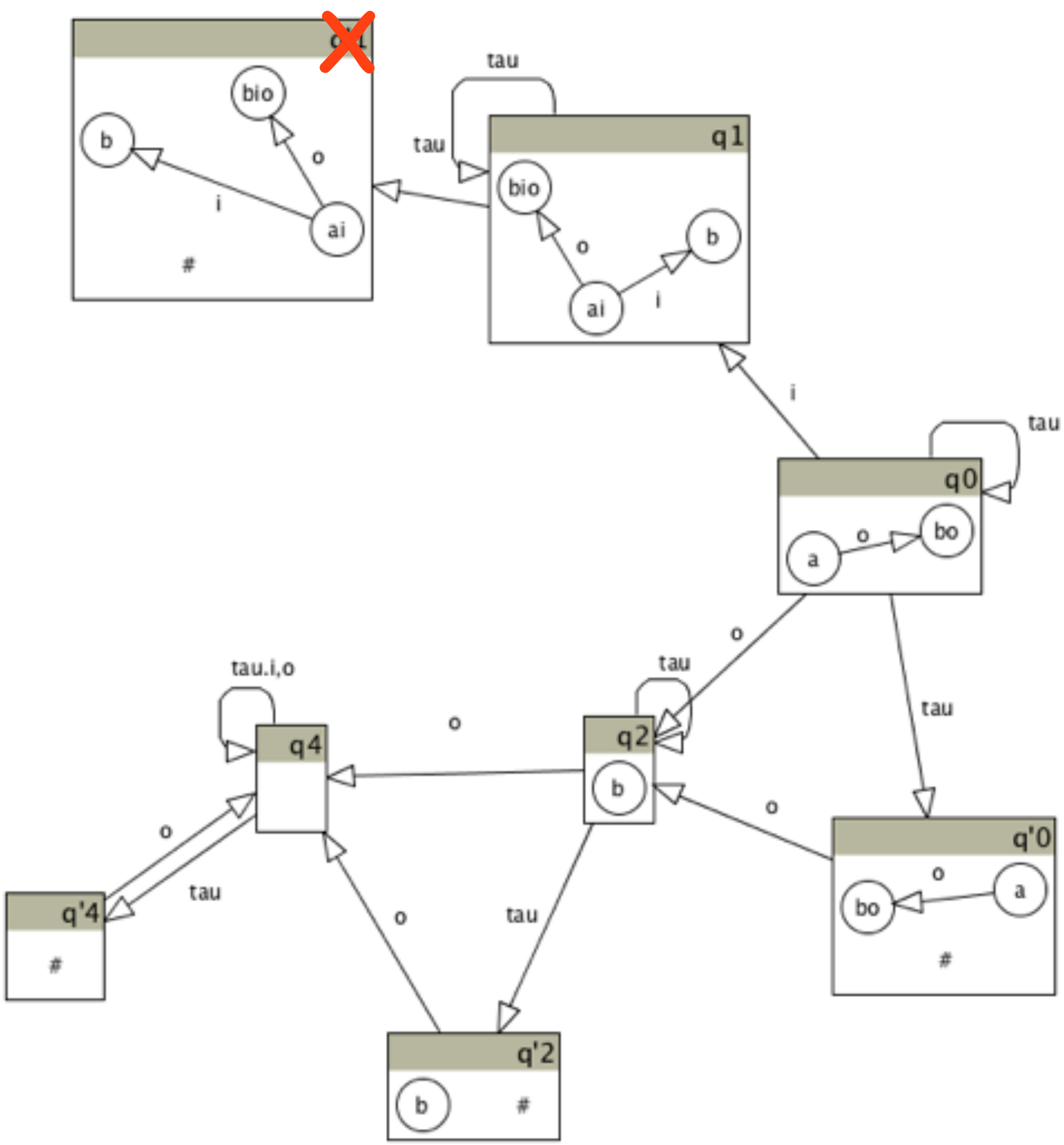


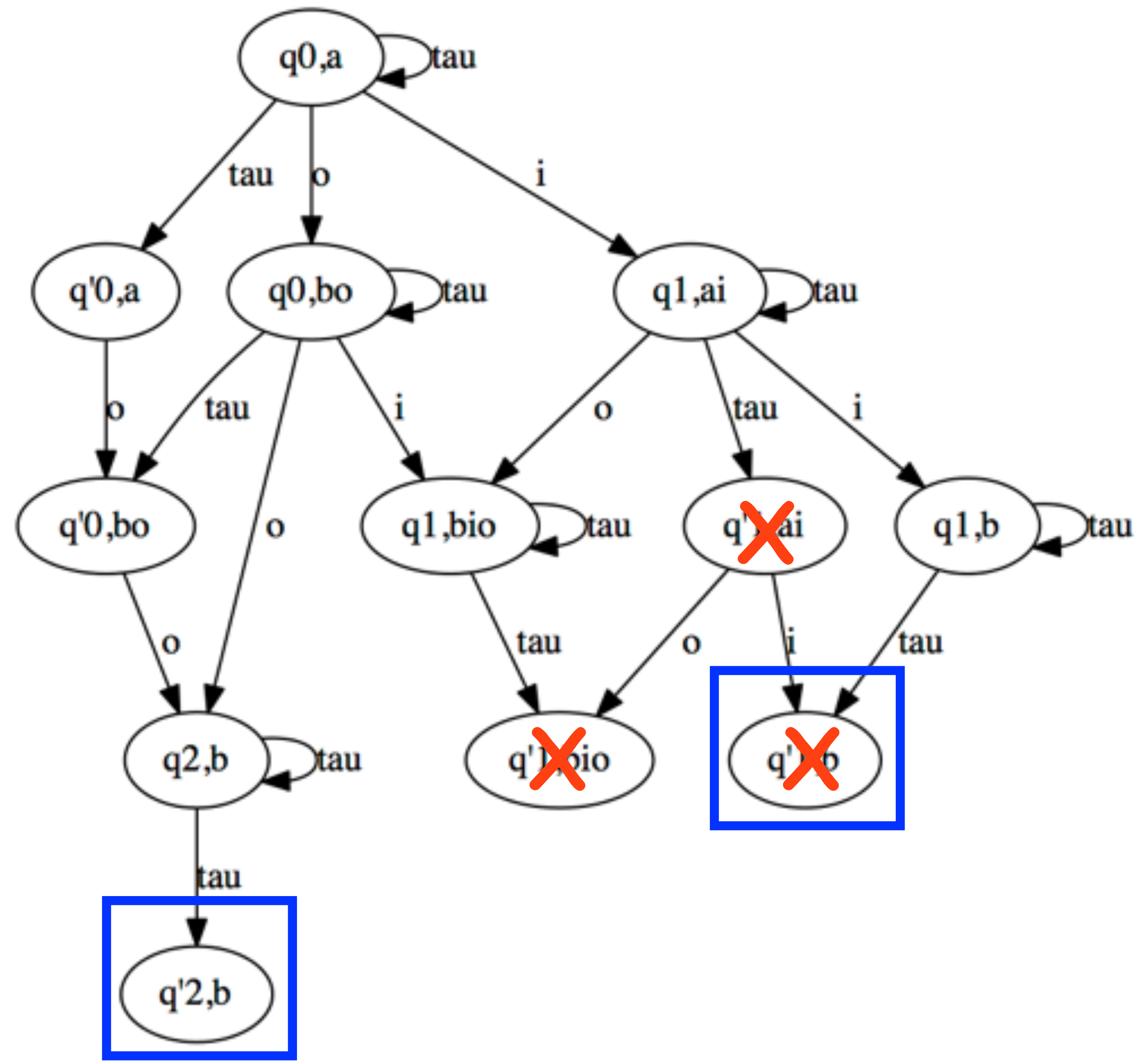
DF,1-controllability example: $TS_2 \oplus N$



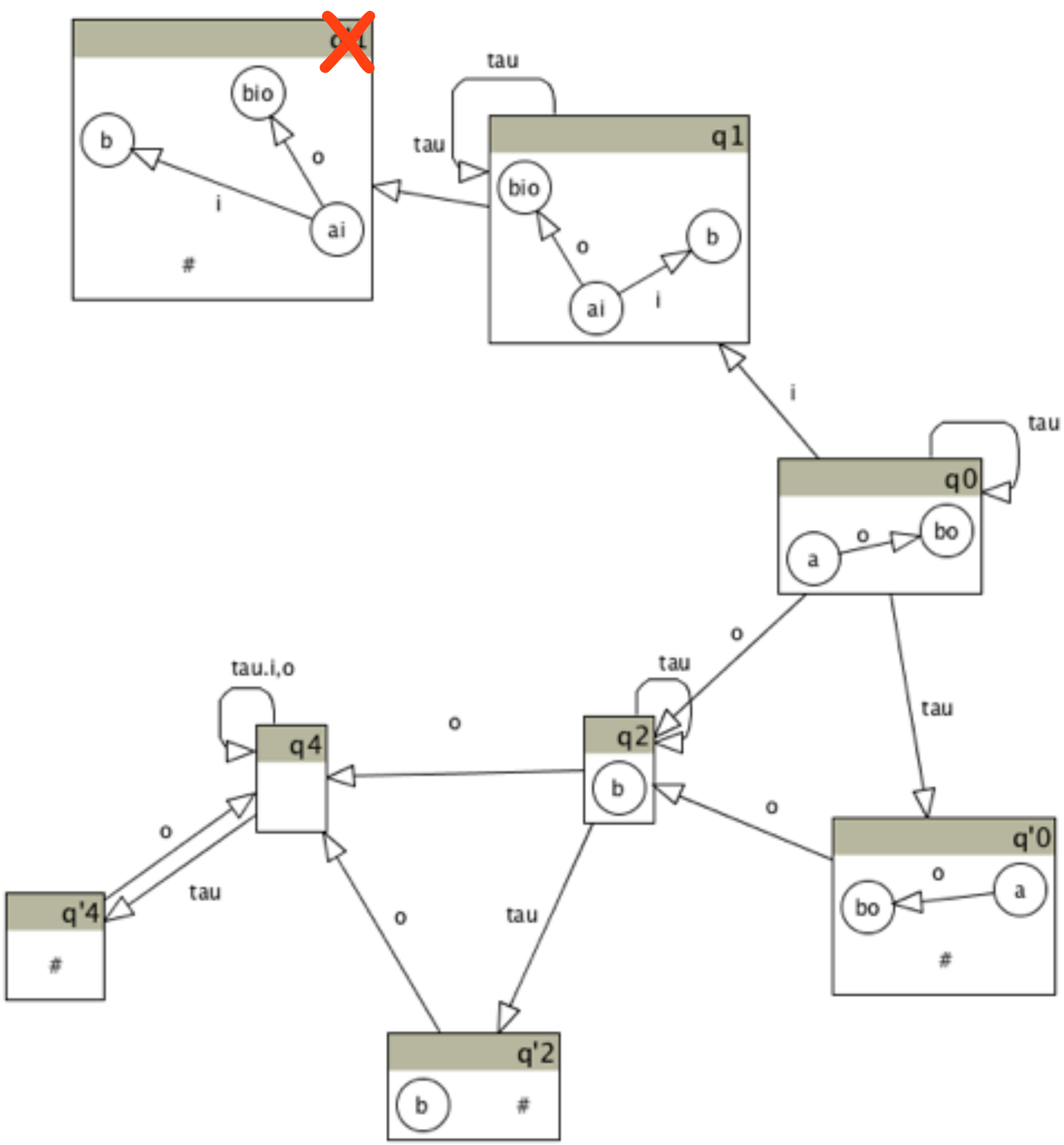


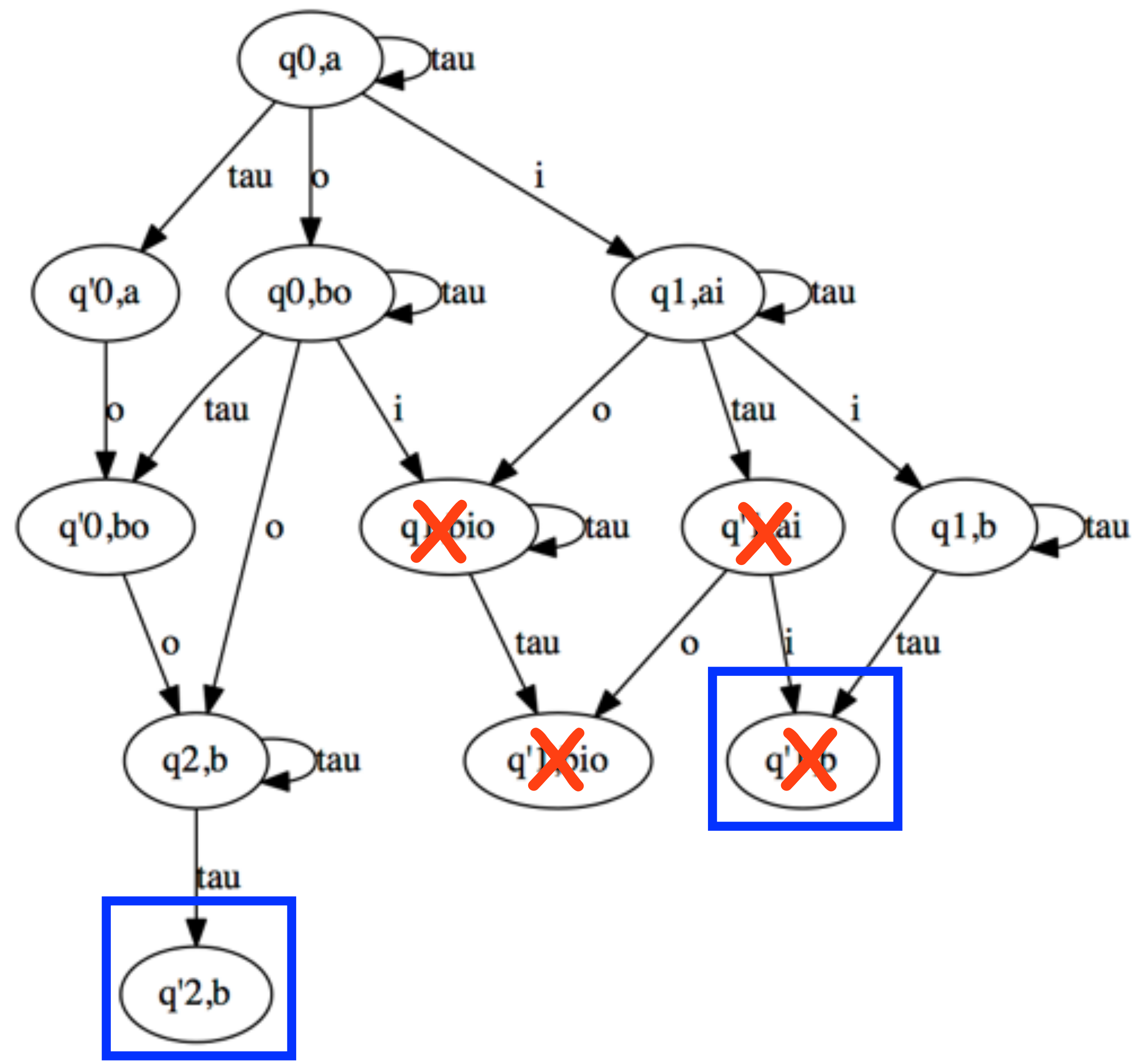
DF,1-controllability example: $TS_2 \oplus N$



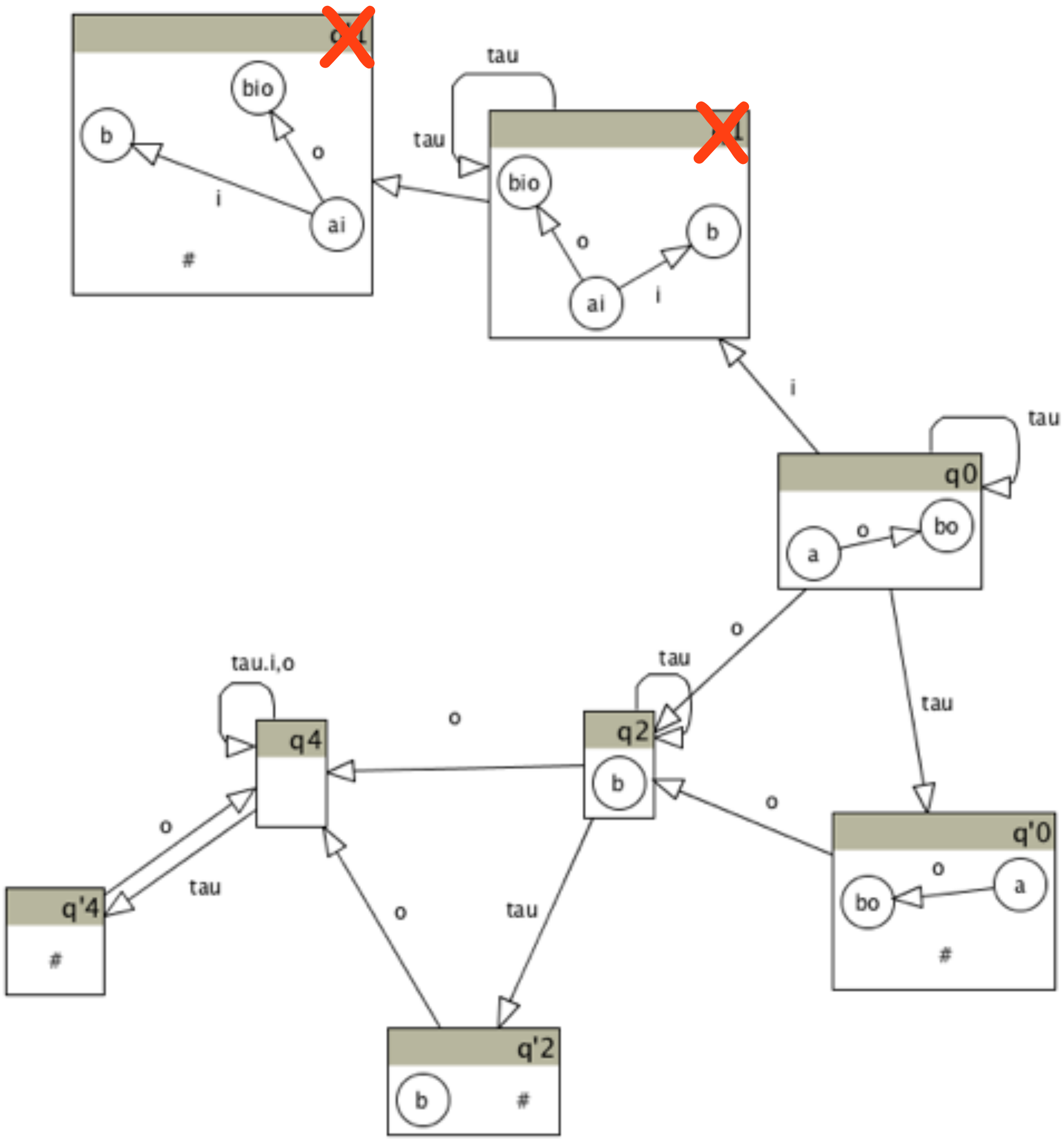


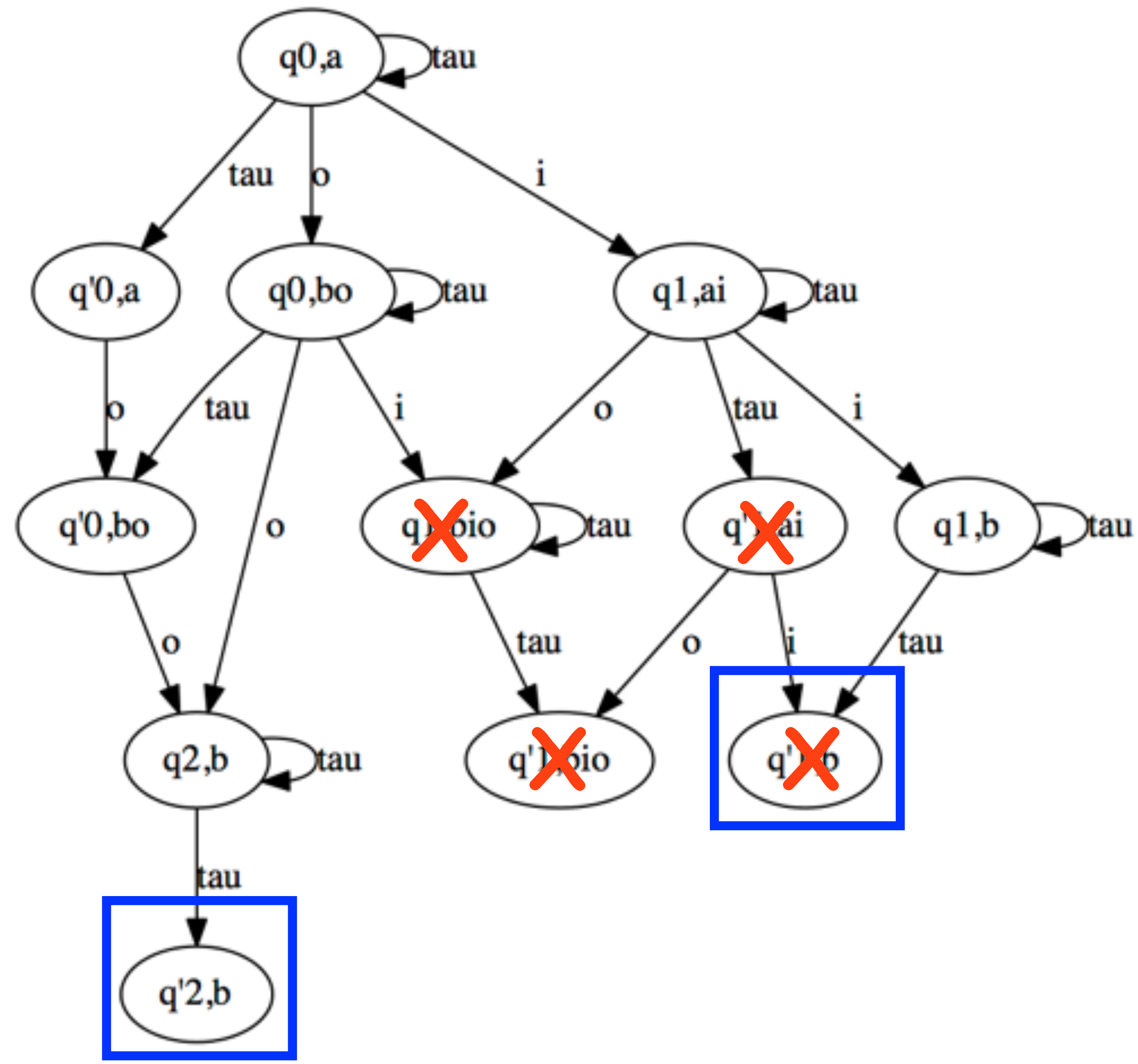
DF,1-controllability example: $TS_2 \oplus N$



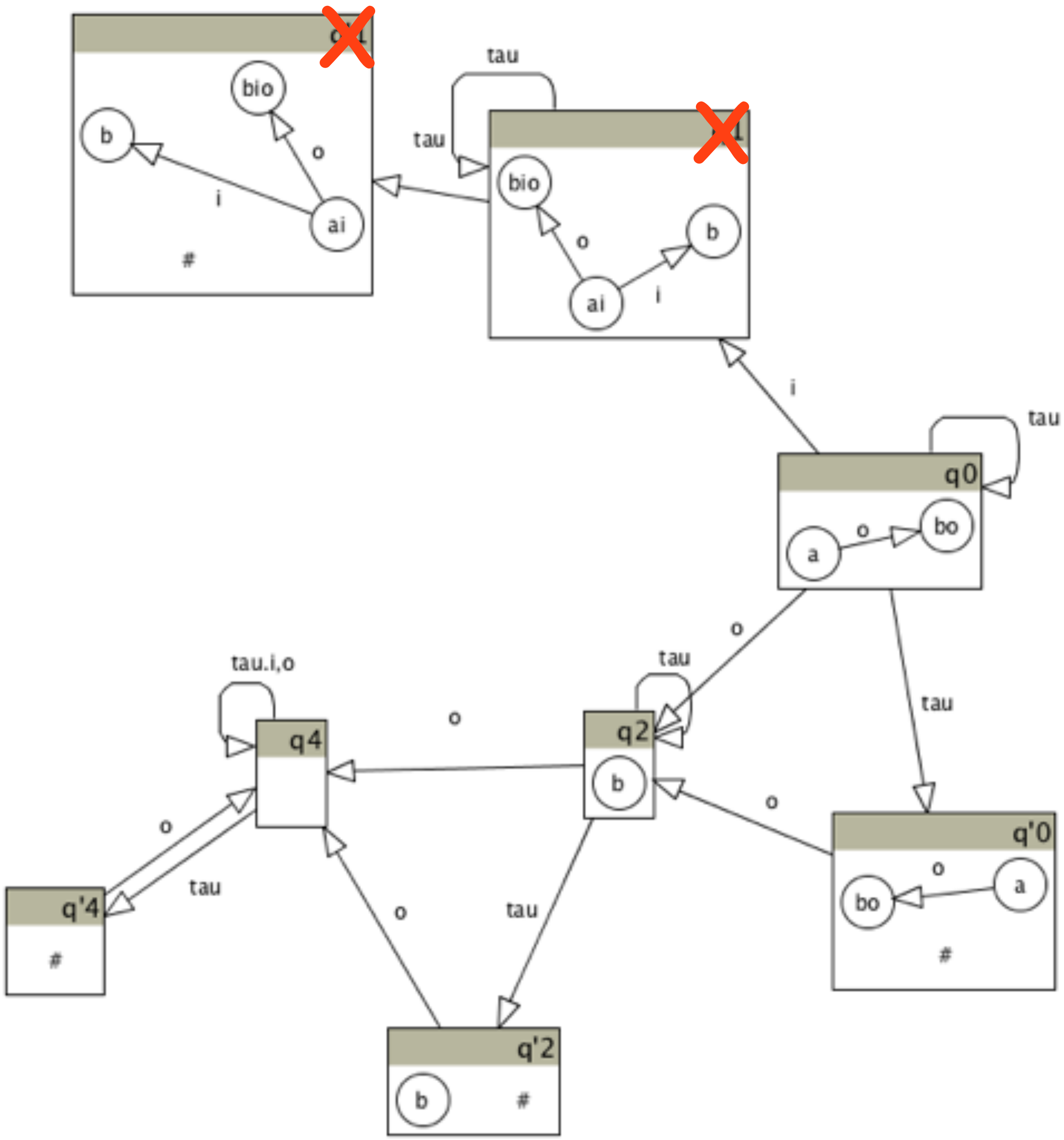


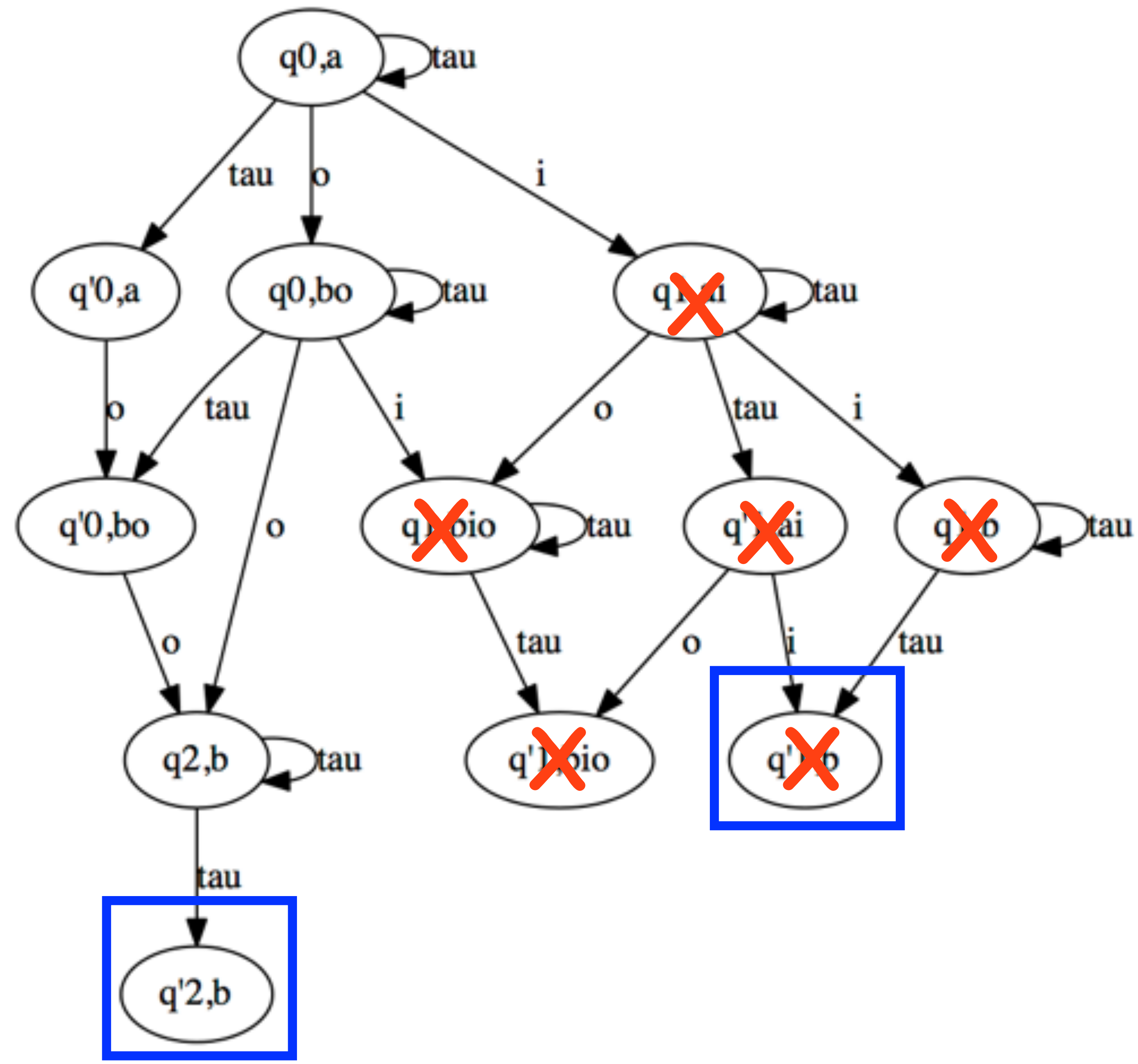
DF,1-controllability example: $TS_2 \oplus N$



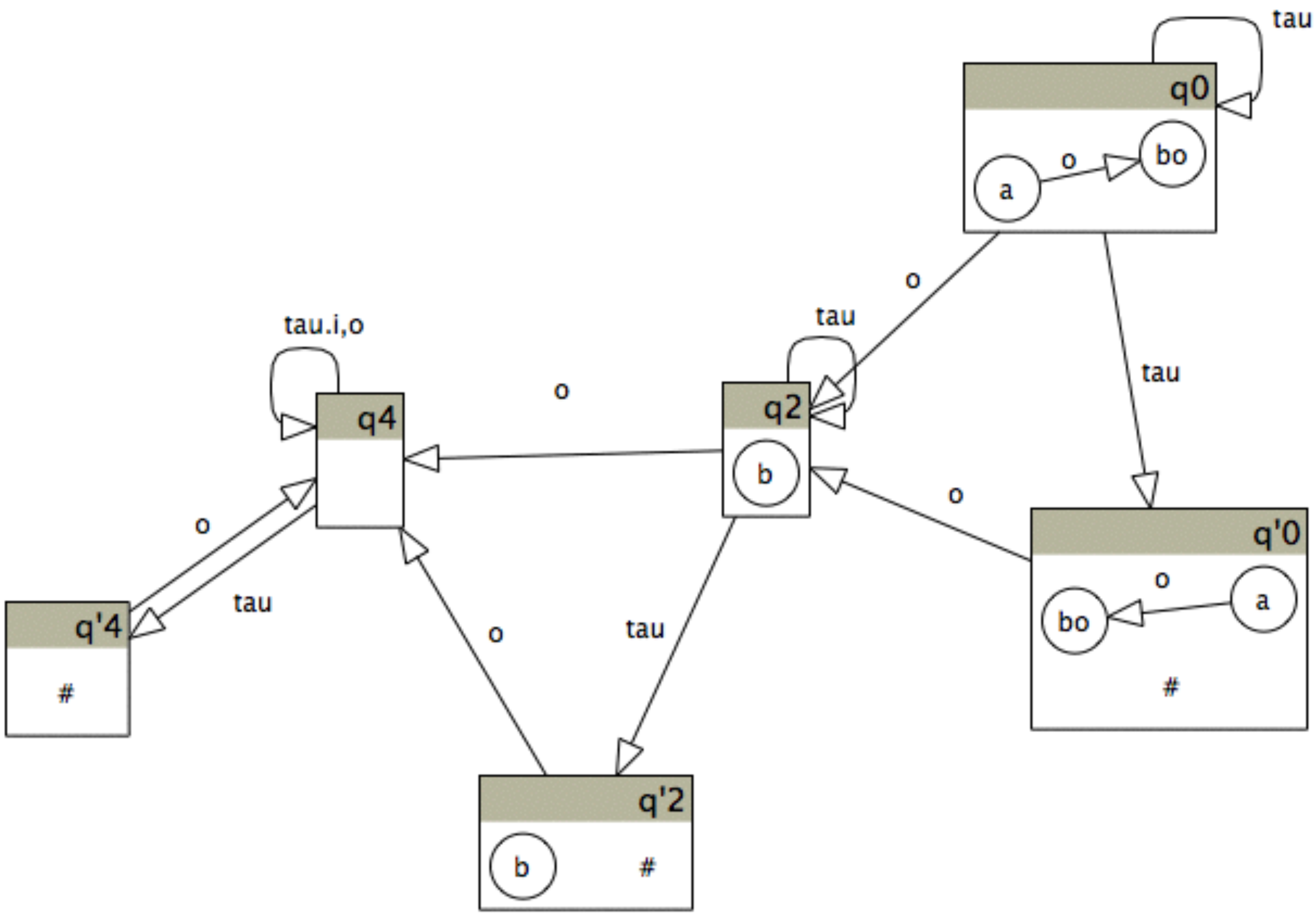


DF,1-controllability example: $TS_2 \oplus N$

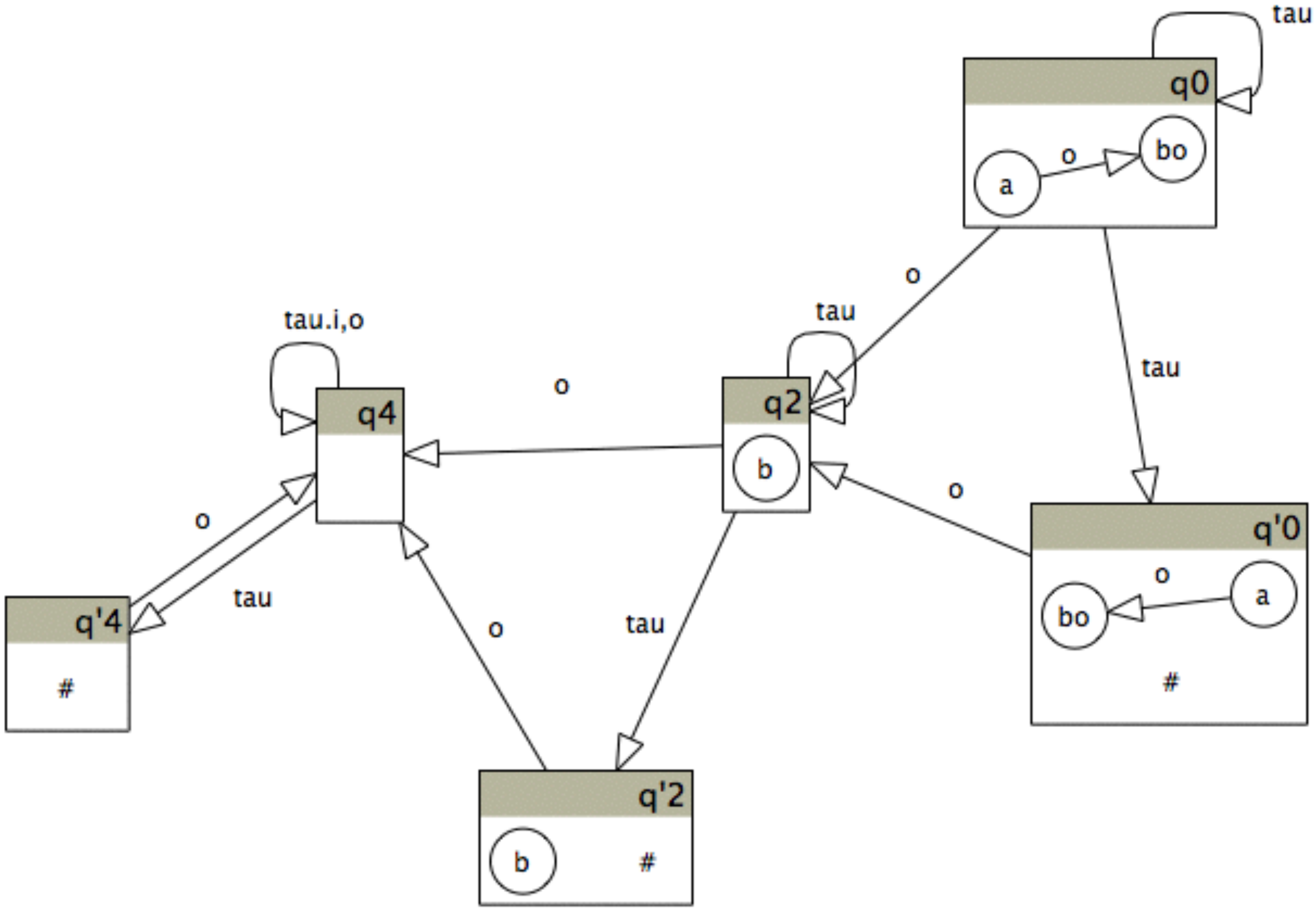


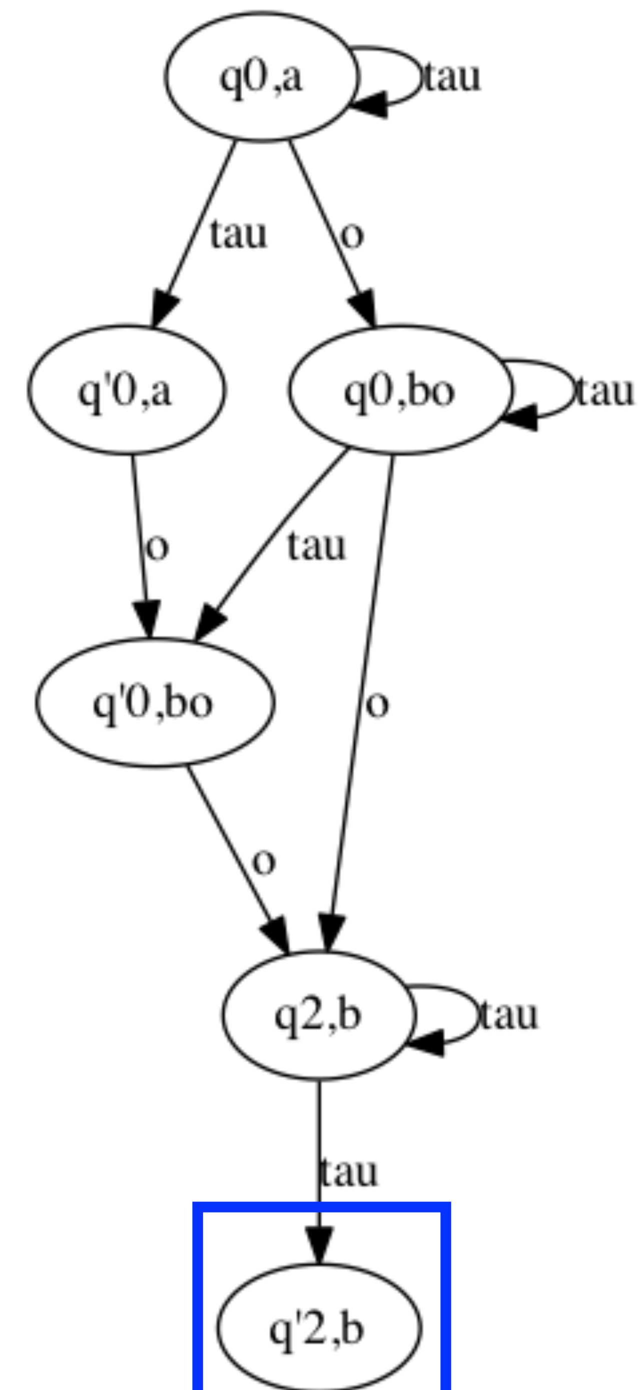


DF,1-controllability example: TS_3



DF,1-controllability example: $TS_3 \oplus N$





Note

Note the presence of a state associated with
the empty set of markings

it can appear in a strategy,
but is actually unreachable in the composition with N

for DF,k-controllability it makes no harm

DF,k-controllability theorem

Let TS = (Q,E,q_0,Q_f) be the automaton produced by applying the above procedure to the open net N

Theorem

N is DF,k-controllable

iff

$Q \neq \emptyset$

(proof omitted)

Behavioural properties: LF

A closed net N = $(P,T,F,m_0,\varnothing,\varnothing,M_f)$

is **LF** (**livelock-free**)
if from any reachable marking a final marking is reachable

$$\forall m \in [m_0\rangle.\ [m\rangle \cap M_\mathrm{f} \neq \emptyset$$

LF,k-controllable nets

A bounded and responsive open net
$N = (P,T,F,m_0,P^I,P^O,M_f)$
is **LF,k-controllable**

if there is a (bounded and responsive) partner N'
such that

$N \oplus N'$ is LF
and any place $p \in (P^I \cup P^O)$ is k-bounded in $N \oplus N'$

such an N' is called a **LF,k-strategy** of N

Approach

We construct TS_0 and TS_1 as before.
We change only the iterative step

Iterative step

Define a strategy $\mathbf{TS_{i+1}}$
that removes from TS_i all states q
that can invalidate the property of interest

(as a consequence remove all adjacent edges
and all states that become unreachable)

At some point $TS_{j+1} = TS_j$
and we terminate

TSi+1

$TS_i = (Q_i, E_i, q_0, Q_{\mathrm{f}i})$

remove the state q if $\exists m \in q$ such that in $TS_i \oplus N$
no $[q', m']$ with $q' \in Q_{\mathrm{f}i} \wedge m' \in M_{\mathrm{f}}$ is reachable from $[q, m]$

LF,k-controllability theorem

Let TS = (Q,E,q_0,Q_f) be the automaton produced by applying the above procedure to the open net N

Theorem

N is LF,k-controllable

iff

$Q \neq \emptyset$

(proof omitted)