Methods for the specification and verification of business processes
MPB (6 cfu, 295AA)

Roberto Bruni
http://www.di.unipi.it/~bruni
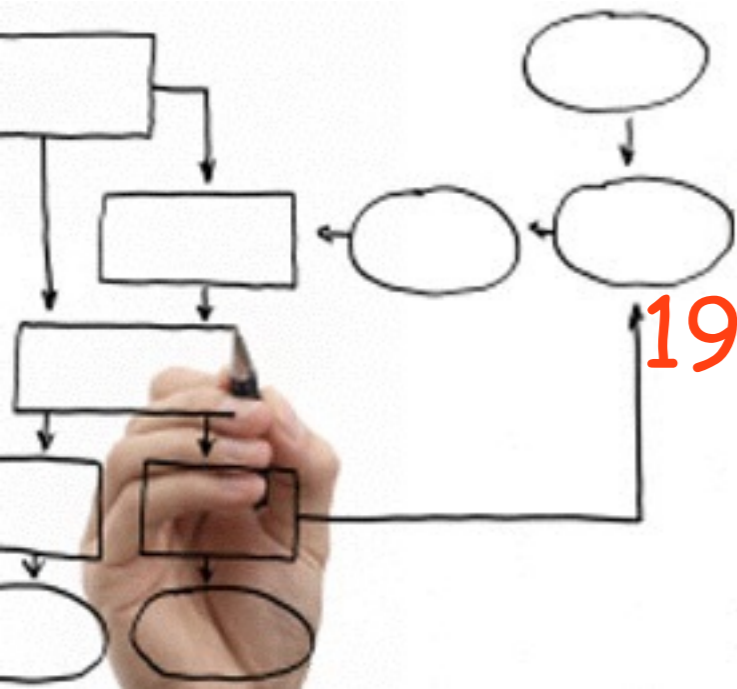
19 - Event-driven process chains

Object

We overview EPC and the main challenges that arise when analysing them with Petri nets

Ch.4.3, 6 of Business Process Management: Concepts, Languages, Architectures

Event-driven Process Chain

An **Event-driven Process Chain** (EPC) is a particular type of flow-chart that can be used for configuring an Enterprise Resource Planning (ERP) implementation

Supported by many tools (e.g. SAP R/3)

EPC Markup Language available (EPML) as interchange format

EPC overview

Important notation to model the domain aspects of business processes

Rather informal notation

EPC focus is on representing domain concepts and processes (not their formal aspects and technical realization)

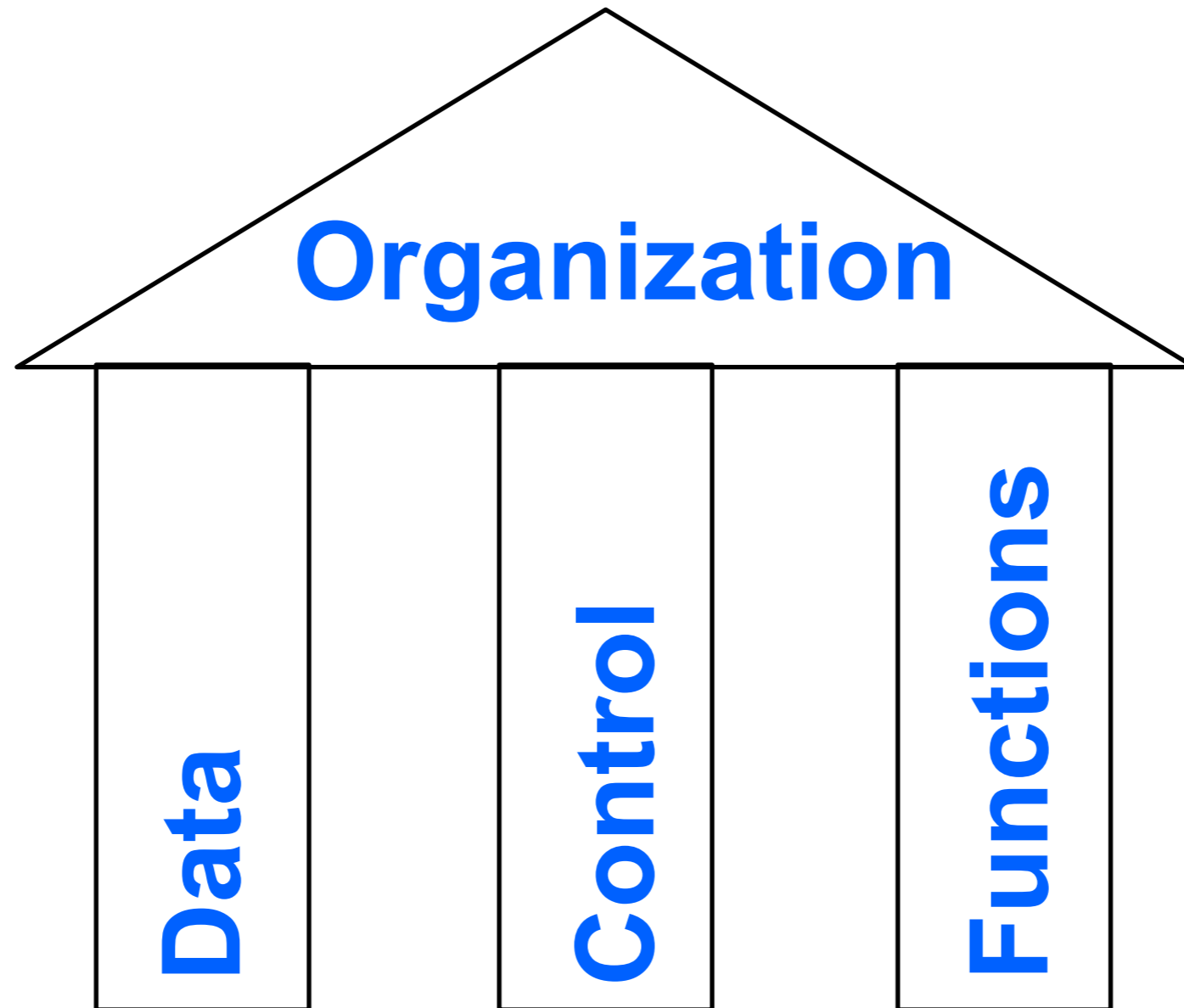It can be used to drive the modelling, analysis and redesign of business process

EPC origin

EPC method was originally developed by
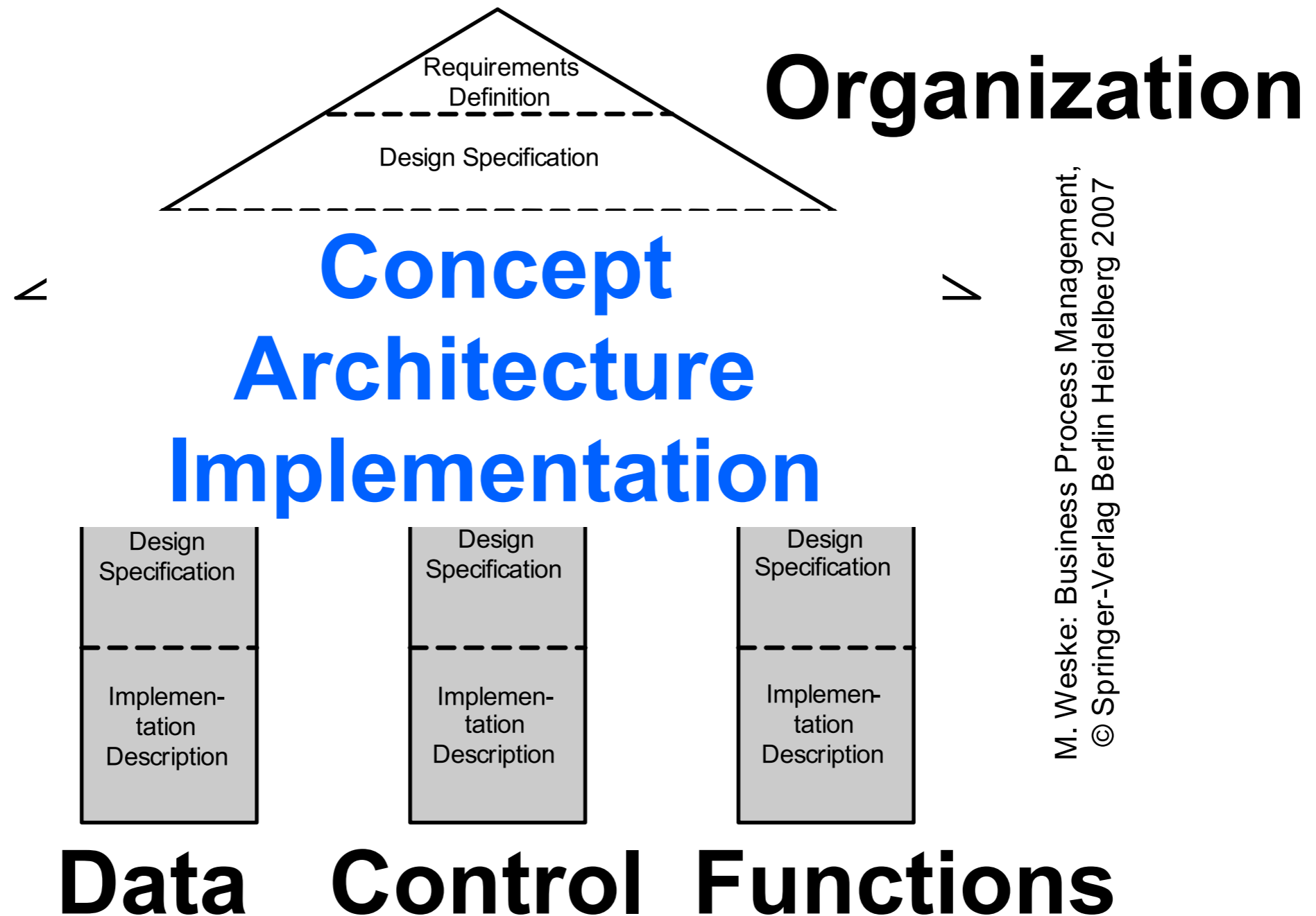Wilhelm-August Scheer (early 1990's)
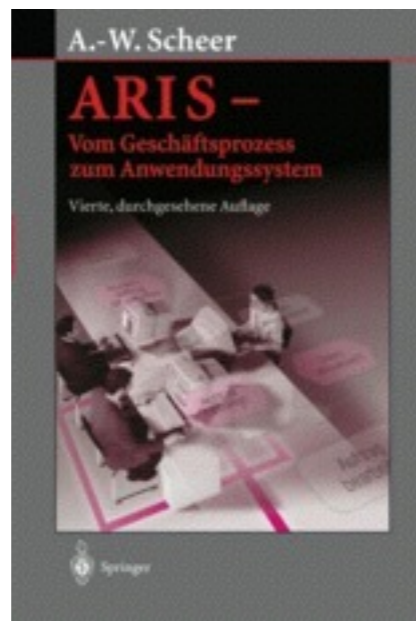
Part of a holistic modelling approach called
ARIS framework
(Architecture of Integrated Information Systems)
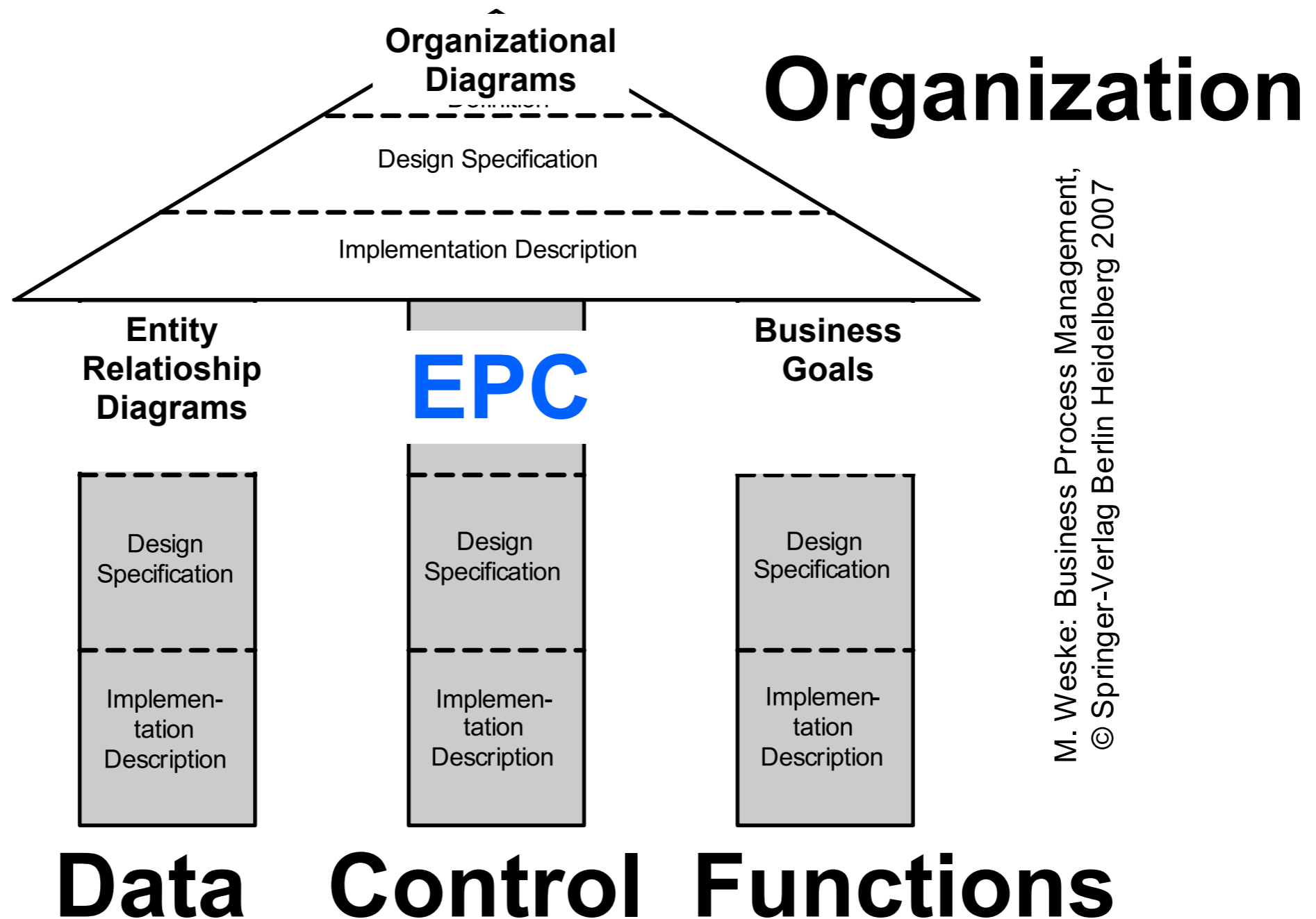
ARIS house (1999): three pillars and a roof…

...and three levels of abstraction each

Requirements
Definition

Design Specification

Organization

Concept
Architecture
Implementation

Design Specification	Design Specification	Design Specification
Implemen-tation Description	Implemen-tation Description	Implemen-tation Description

Data **Control** **Functions**

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

...and three levels of abstraction each

Organizational Diagrams

Organization

Design Specification

Implementation Description

Entity Relatioship Diagrams

EPC

Business Goals

| Design Specification | Design Specification | Design Specification |
| Implemen-tation Description | Implemen-tation Description | Implemen-tation Description |

Data   Control Functions

EPC informally

An EPC is an "ordered" graph
of events and functions

It provides various connectors that allow
alternative and parallel execution of processes

The flow is specified by logical operators
AND, XOR, OR

Simple, easy-to-understand notation

EPC ingredients: Event

Any EPC diagram must
start with **event(s)**
and end with **event(s)**

Passive elements used to describe
under which circumstances a process (or a function) works
or which state a process (or a function) results in
(like pre- / post-conditions)

Graphical representation: hexagons

EPC ingredients: Function

Any EPC diagram may involve
several **functions**

Active elements used to describe
the tasks or activities of a business process

Functions can be refined to other EPC

Graphical representation:
rounded rectangles

EPC ingredients: Logical connectors

Any EPC diagram may involve
several **connectors**

Elements used to describe
the logical relationships between elements in the diagram

Branch, merge, fork, join

Graphical representation:
circles (or also octagons)
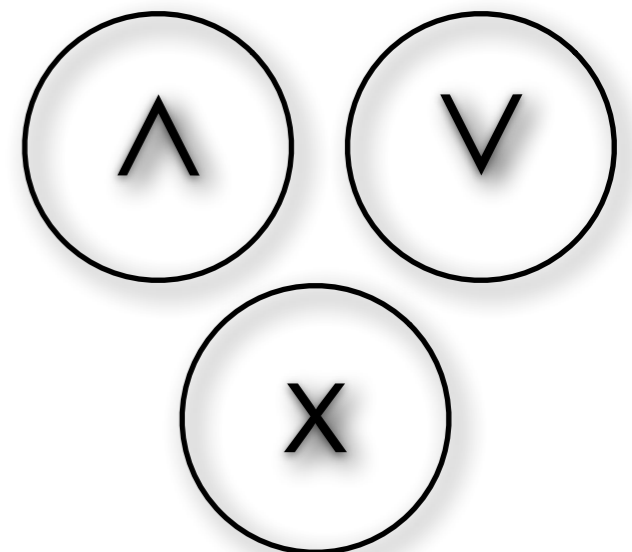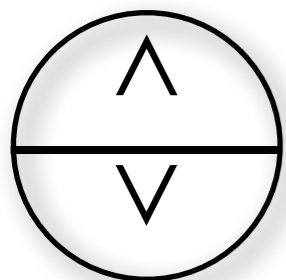
EPC ingredients: Control flow

Any EPC diagram may involve
several **control flow connections**

Control flow is used to connect
events with functions and connectors
by expressing causal dependencies

Graphical representation:
dashed arrows

EPC ingredients at a glance
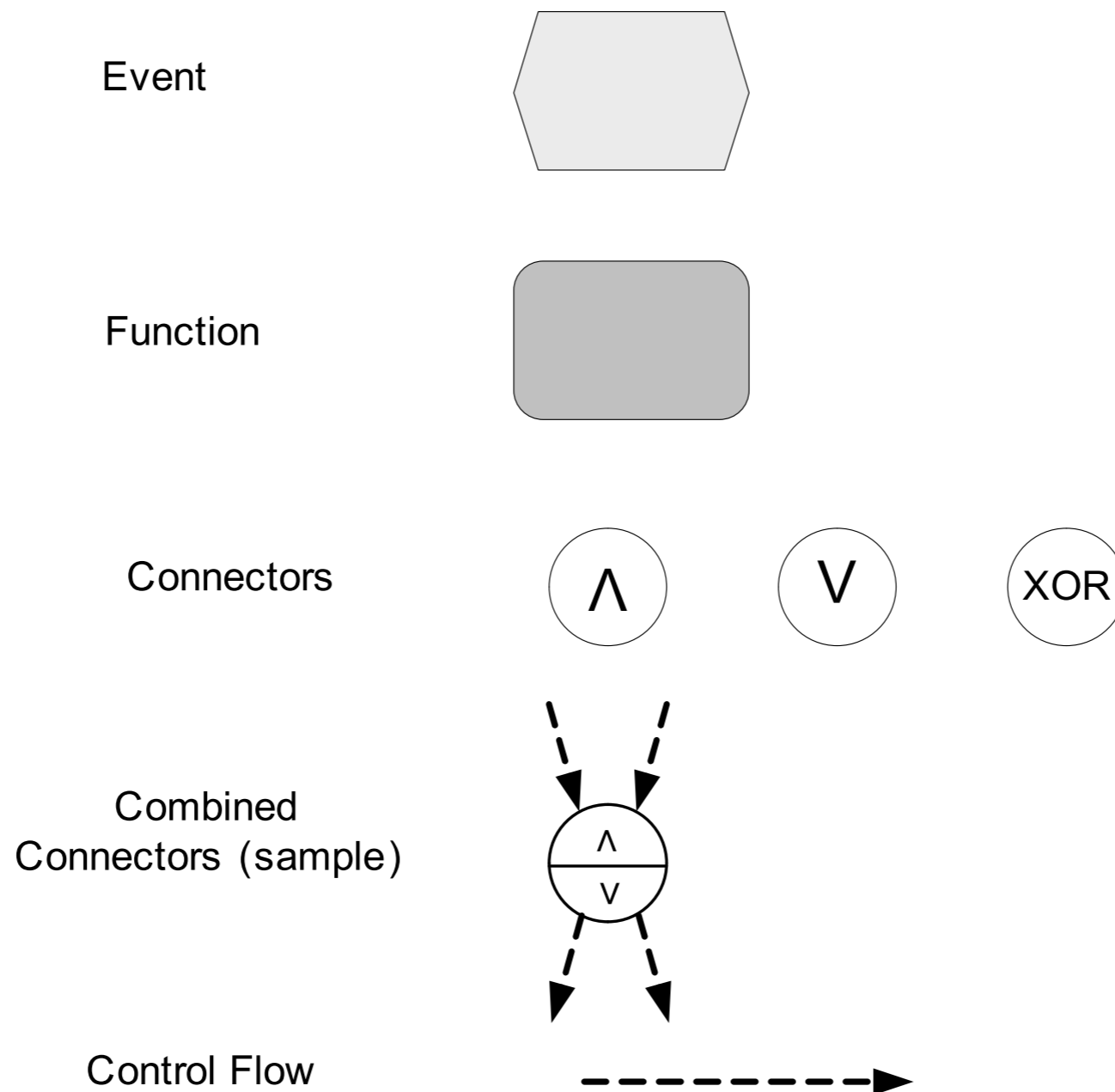
Event

Function

Connectors   ∧   ∨   XOR

Combined
Connectors (sample)

∧
∨

Control Flow

EPC ingredients: Diagrams

EPC elements can be combined in a fairly free manner
(possibly including cycles)

There must be at least one start event and one end event
Events have at most one incoming and one outgoing arc
Events have at least one incident arc

Functions have exactly one incoming and one outgoing arc

The graph is weakly connected (no isolated nodes)

Connectors have either one incoming arc and multiple outgoing arcs
or viceversa (multiple incoming arcs and one outgoing arc)

EPC ingredients: Diagrams

Other constraints are sometimes imposed

There is no arc between two events
There is no arc between two functions

Unique start / end event

No event is followed by a decision node
(i.e. (X)OR-split)

EPC allowed connections

EPC an example

Legend

| Start event | Simple flow – Sequence | End event |
| function → event → function | | |

XOR-split – Exclusive choice

AND-split – Parallel split

OR-split – Multi-choice

XOR-join – Simple merge

AND-join – Synchronization

OR-join – Gen. Sync. Merge

18
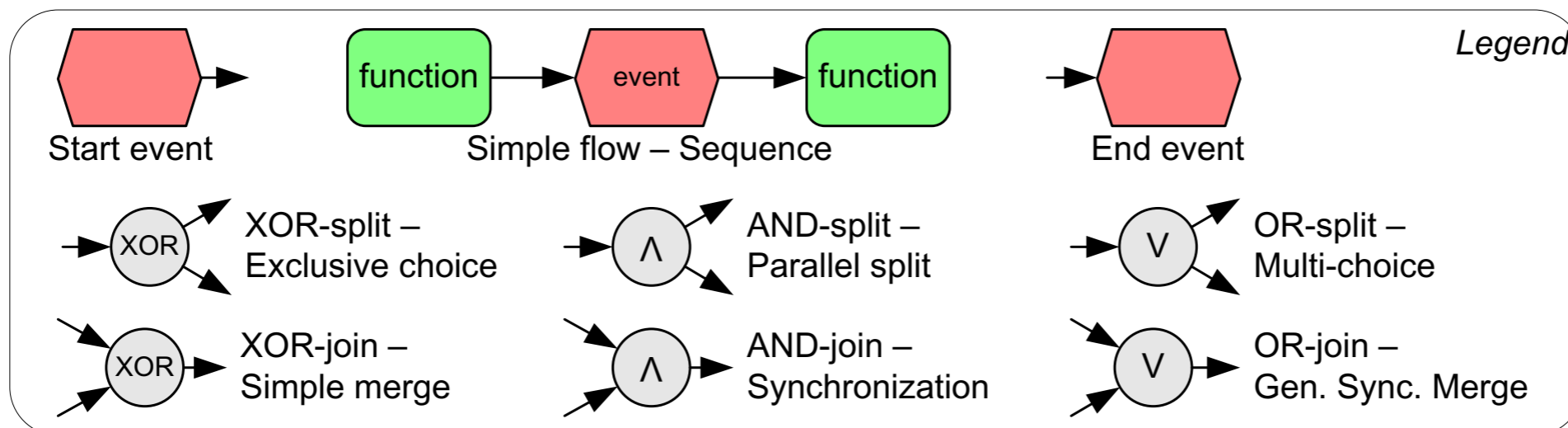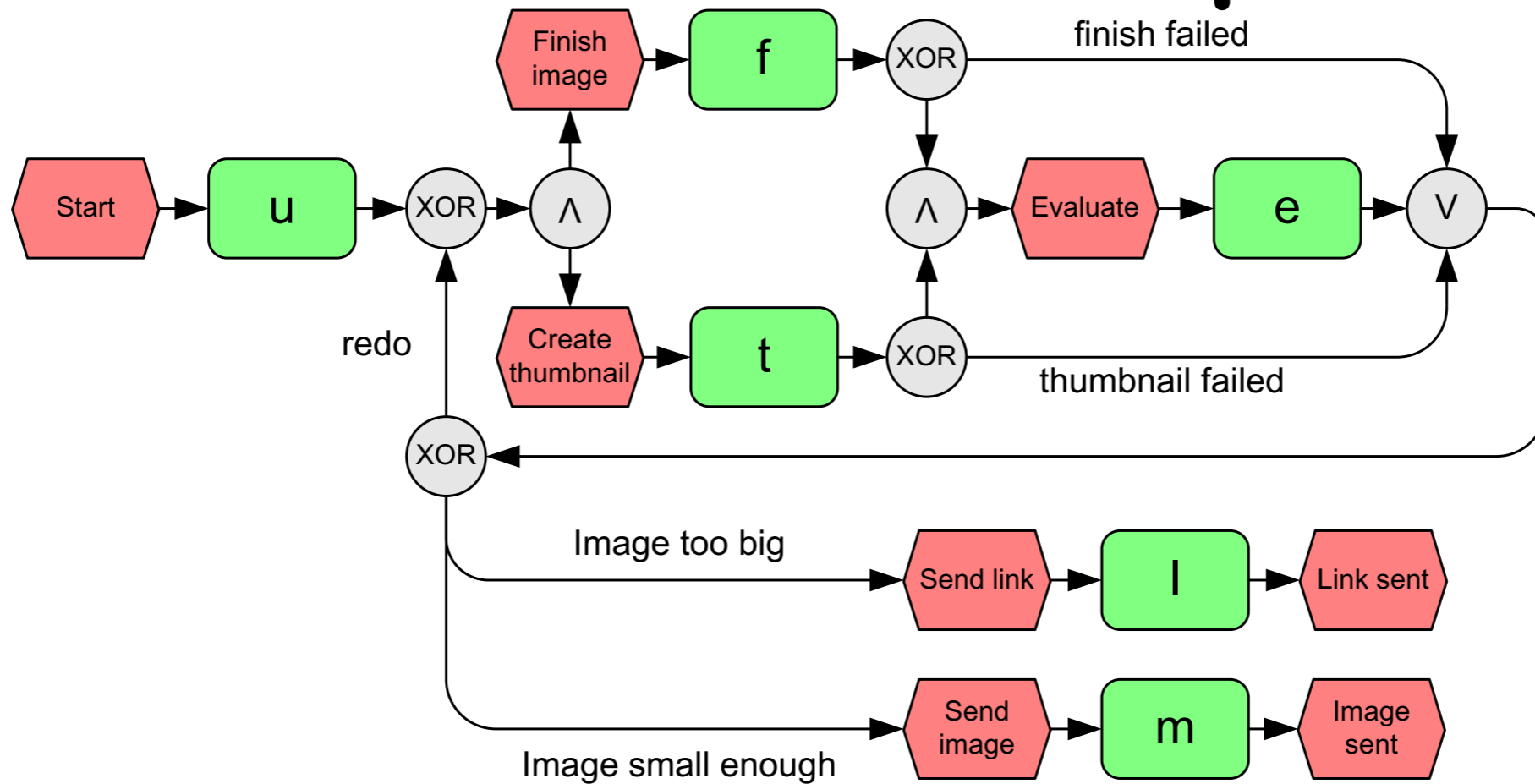
Other stuff

Other decorations / annotations for functions:

Information, material, resource object: represents objects in the real world that can be input data or output data for a function
(rectangles linked to function boxes)

Organization unit: determines the person or organization responsible for a specific function
(ellipses with a vertical line)

Supporting system: technical support
(rectangles with vertical lines on its sides)

Other stuff

EPC Diagram always starts with Event

Organization unit assignments show the connection between an organization unit and the function it is responsible for.

Event describes what circumstances a function or a process works or which state a function or process results in.

Organization units determine which person or organization within the structure of an enterprise is responsible for a specific function.

Information, material, or resource objects portray objects in the real world, for example business objects, entities, etc., which can be input data serving as the basis for a function, or output data produced by a function.

Event

Organization Unit

Function

Information or Material

XOR operator corresponds to making decision of which path to choose among several control flows.

Function describes the transformations from an initial state to a resulting state.

XOR

Event

Event

This object represents information system

System

A control flow connects events with functions, process paths, or logical connectors

Function

Information or Material

Information flows show the connection between functions and input or output data, upon which the function reads changes or writes.

Event

EPC Diagram always ends with Event

20

EPC intuitive semantics

A process starts when some initial event(s) occurs

The activities are executed according to the constraints in the diagram

When the process is finished, only final events have not been dealt with

If this is always the case, then the EPC is "correct"

Folder-passing semantics

Semantics

- State: Process folders

- Transition relation:
  Propagation of process folders

Folder-passing semantics: events
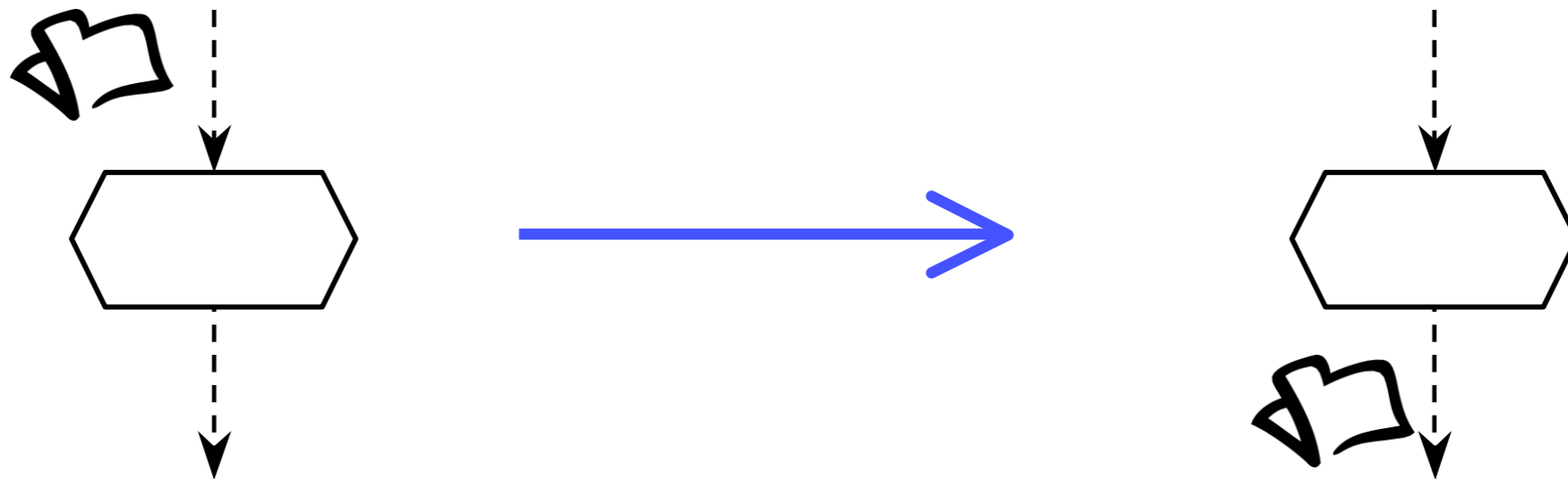
Folder-passing semantics: AND-split
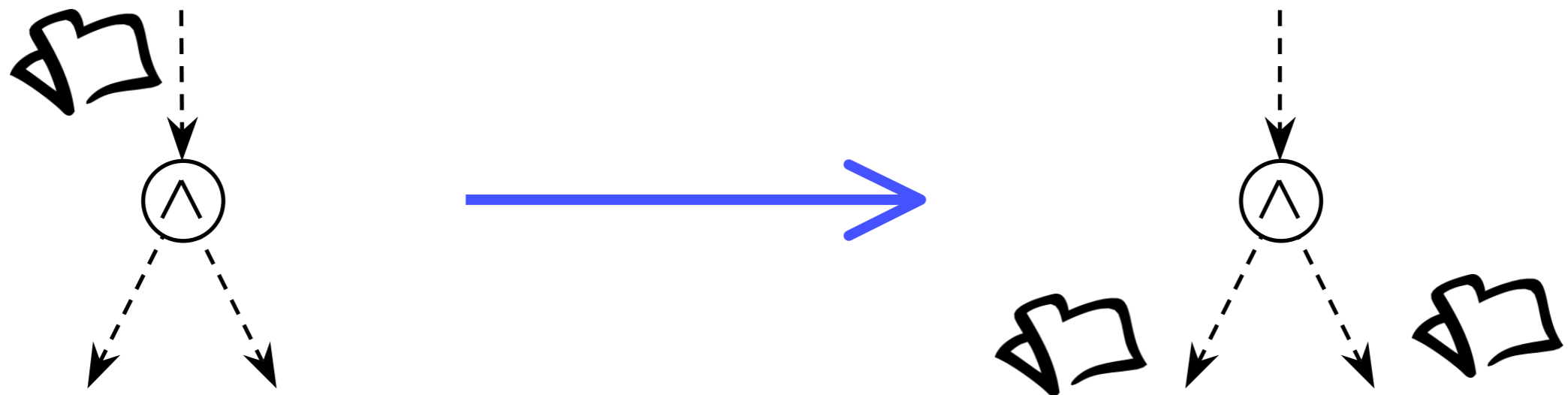
Folder-passing semantics: XOR-split
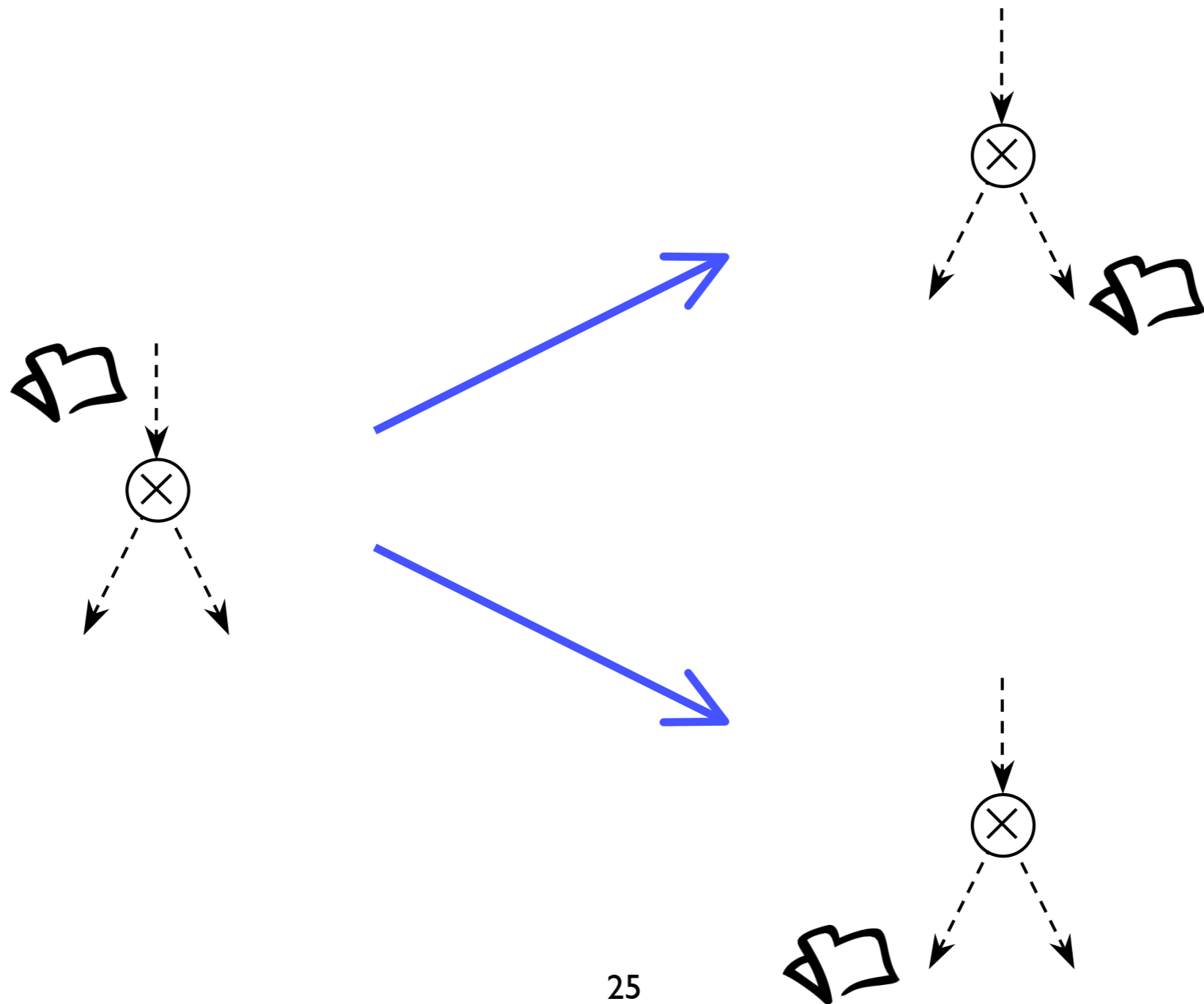
Folder-passing semantics: XOR-join
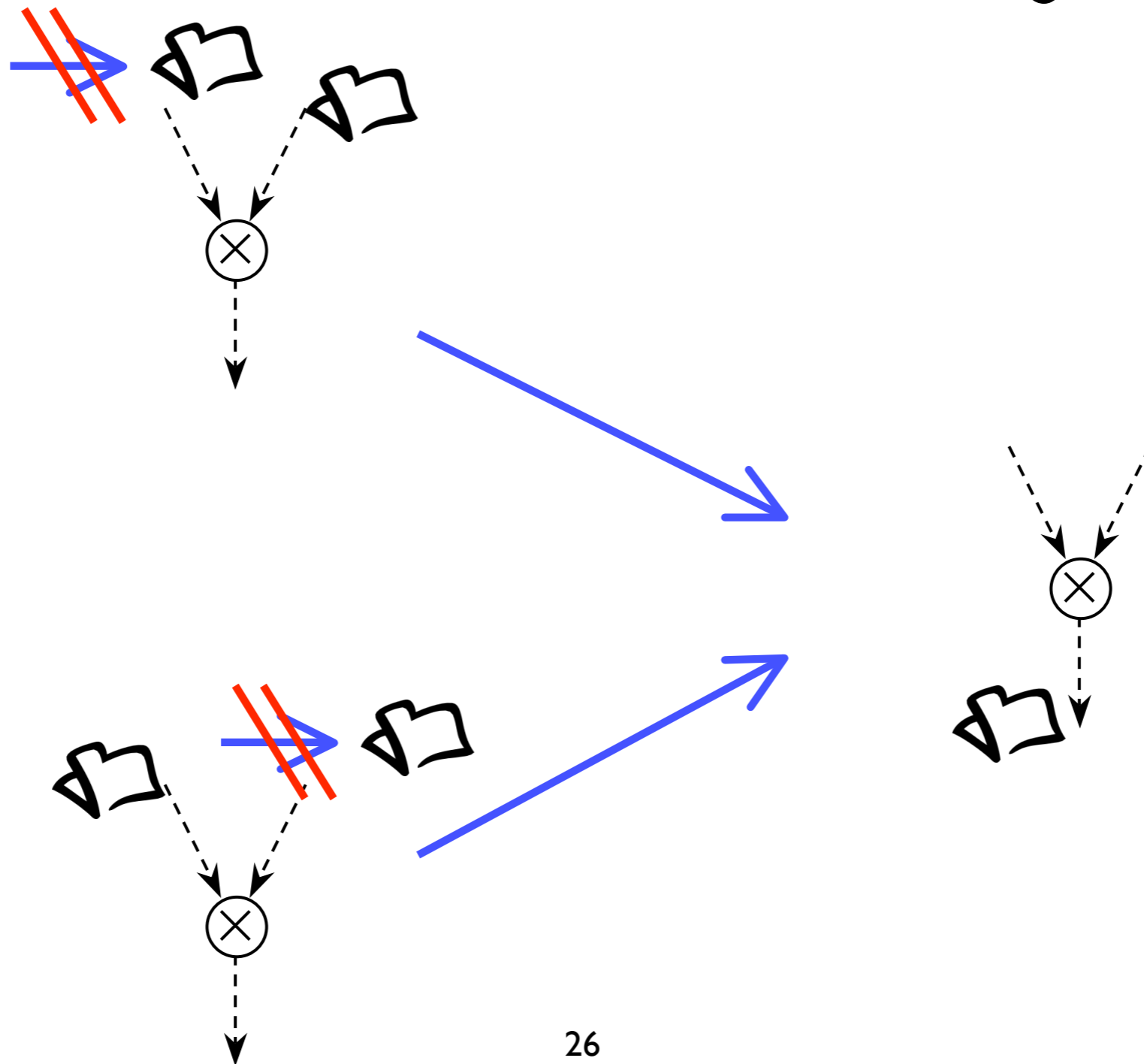
Folder-semantics in one slide
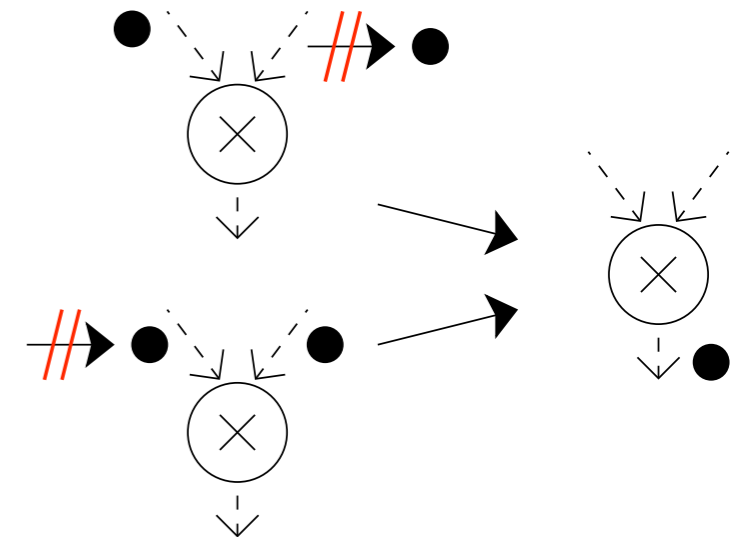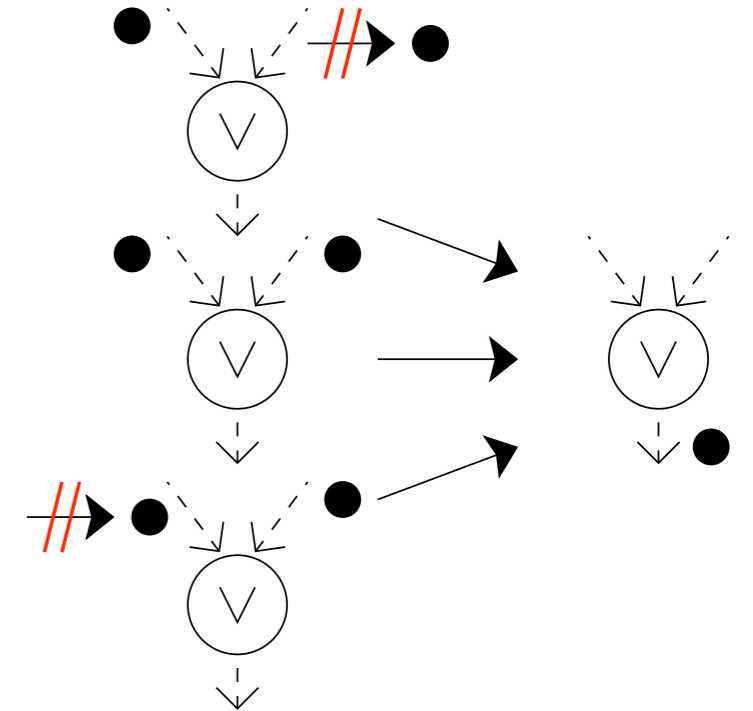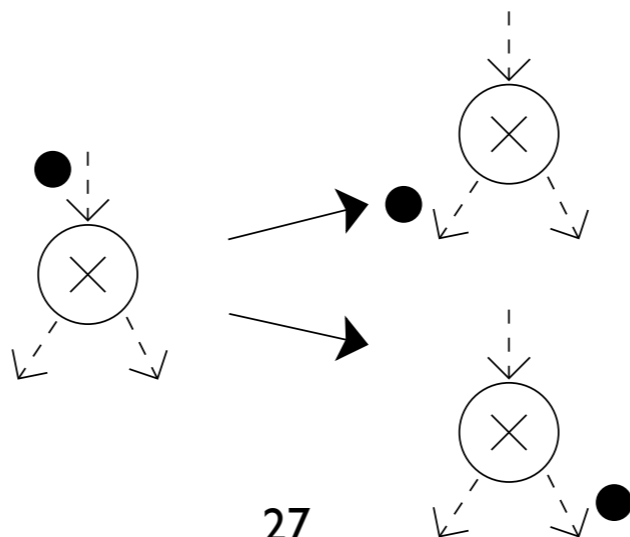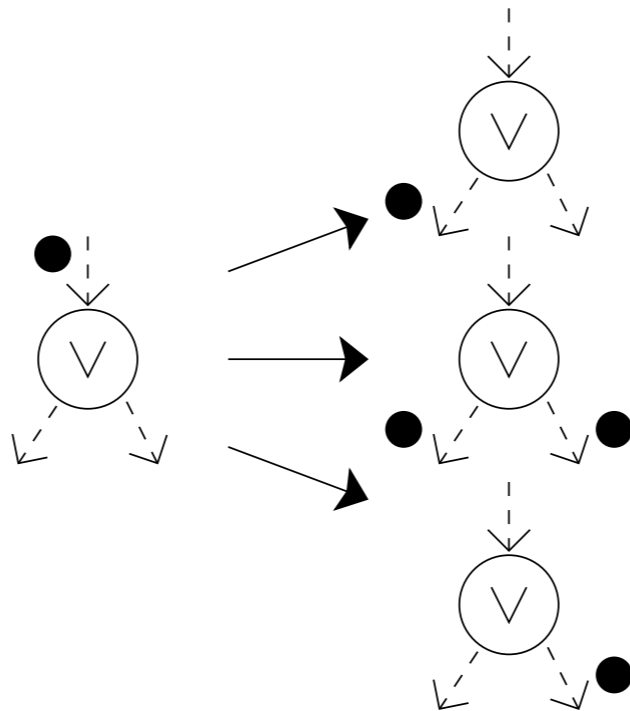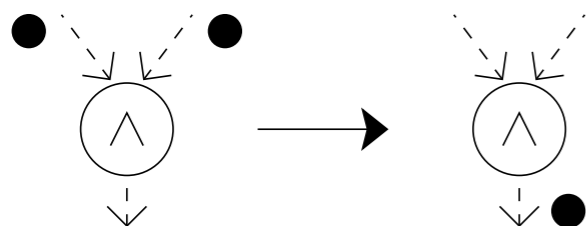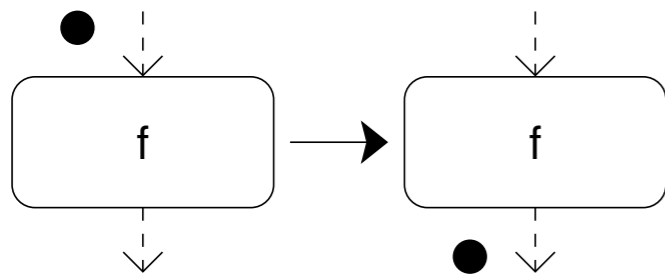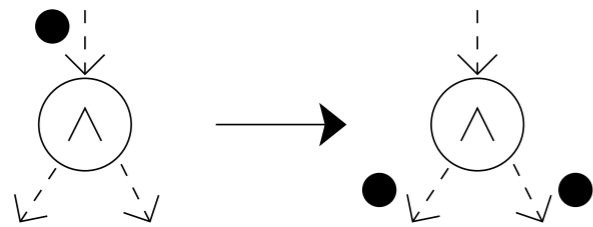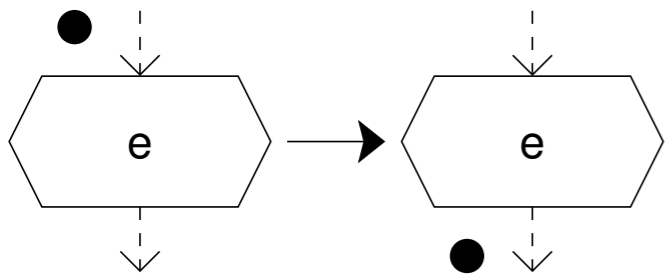
A vicious circle
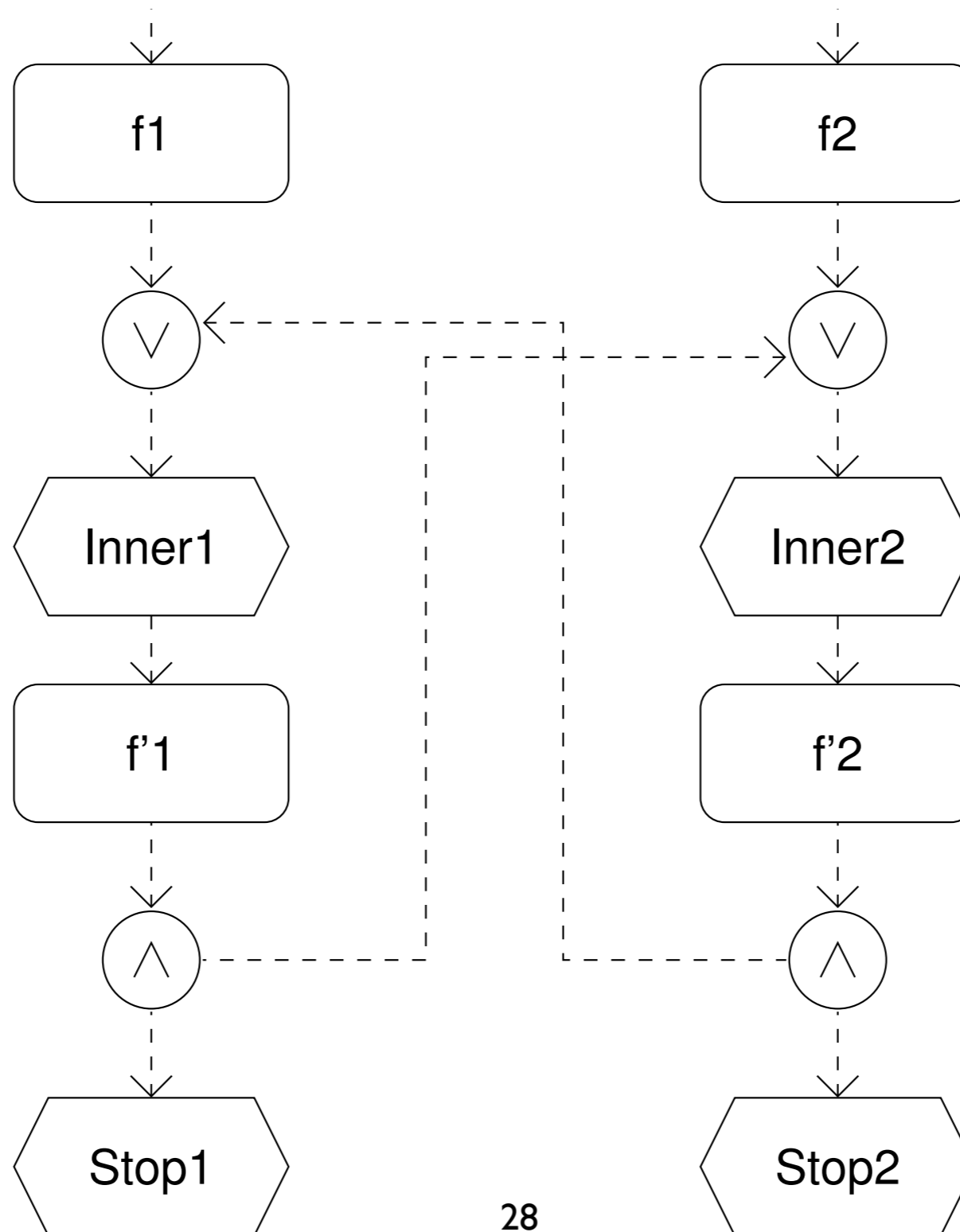
A vicious circle?
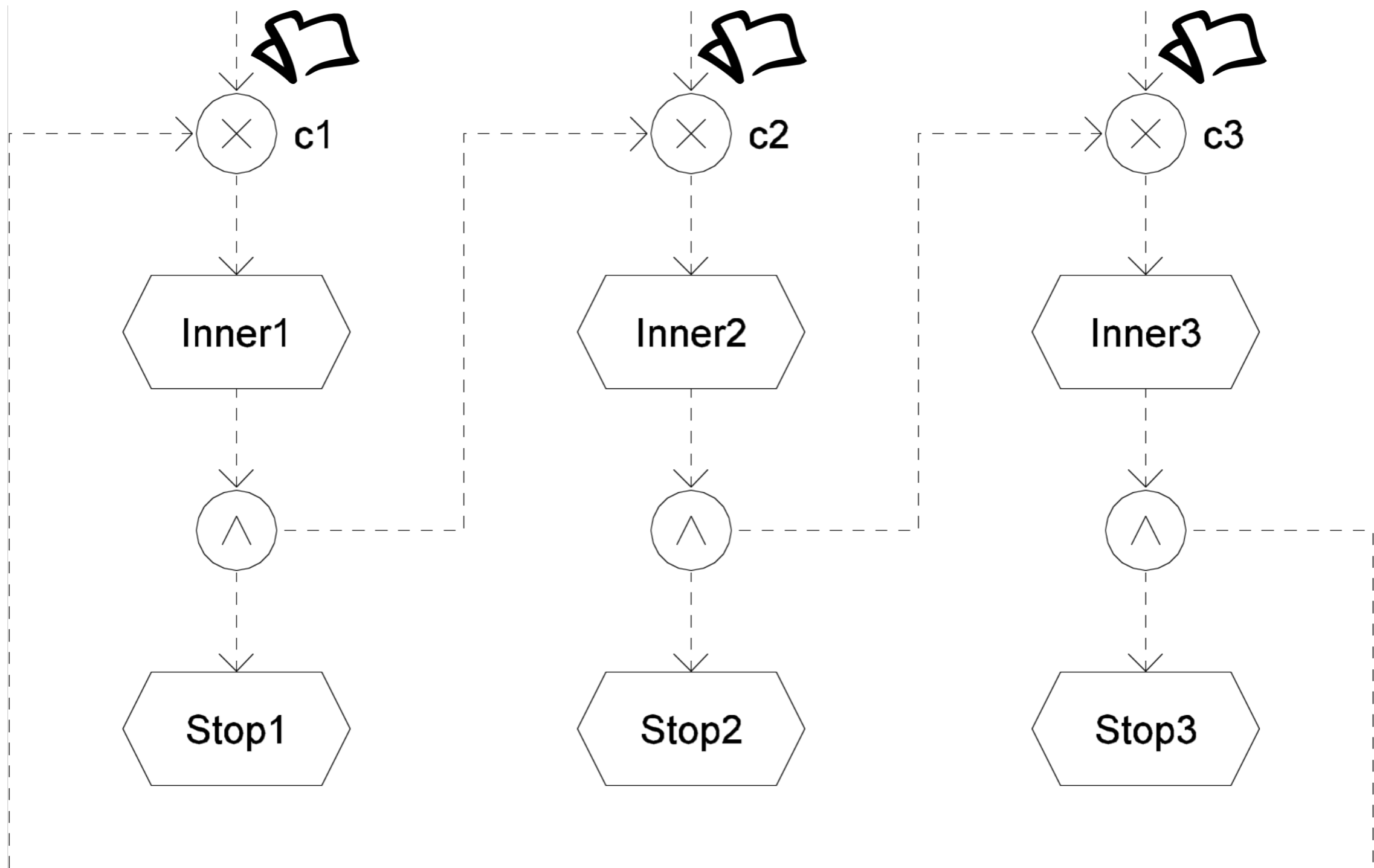
EPC semantics?

Little unanimity around the EPC semantics

Roughly described (verbal form) in the original publication by Scheer (1992)

Later, several attempts to define formal semantics (in many cases they end up attributing different meanings to the same EPC)

Discrepancies typically stem from the interpretation of (X)OR connectors (in particular, join case)
Other issues: unclear start, join/split balancing, alternation of events and functions

Problem with start events

A start event is an event with no incoming arc

A start event invokes a new execution of the process template

What if multiple start events occur?

Start events are mutually exclusive
(as if they were preceded by an implicit XOR split)

Problem with start events

hypothetical / implicit split
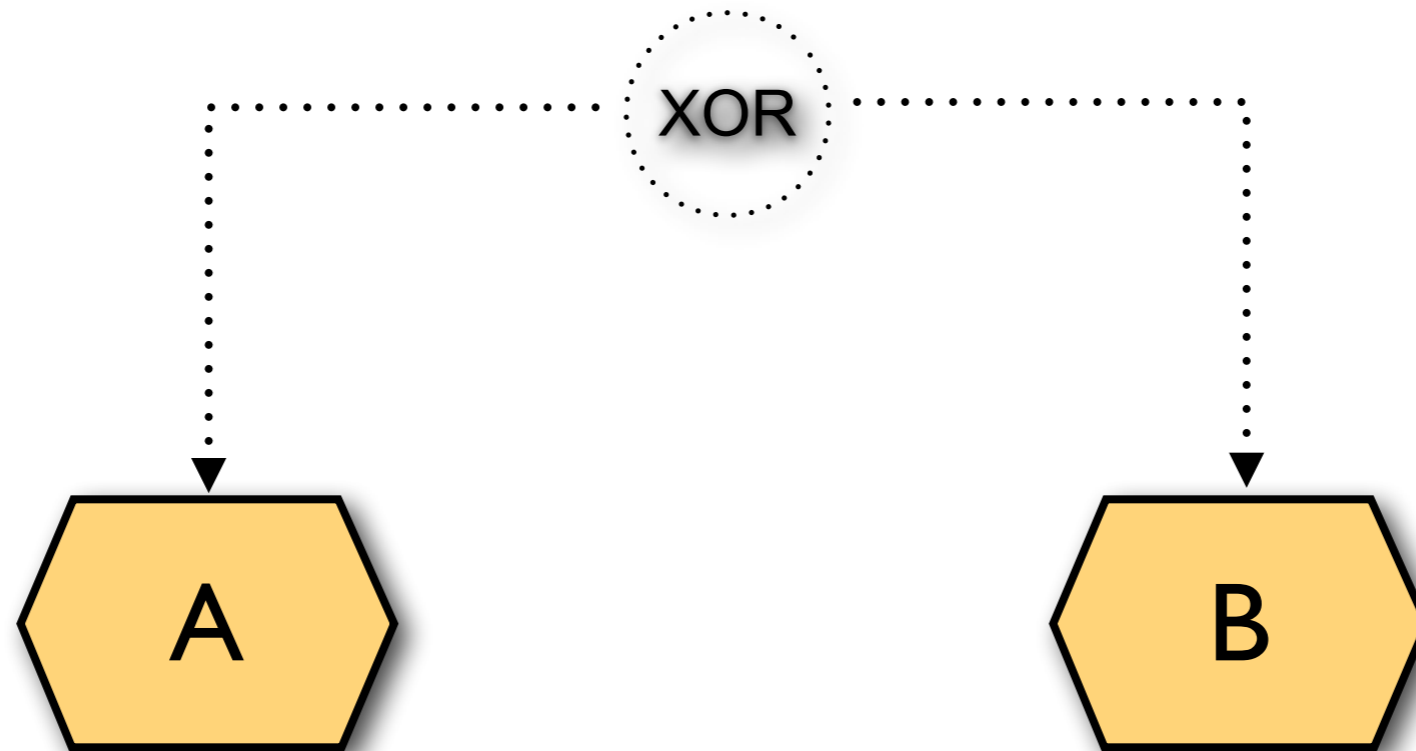
Every join has a split

In theory, every join has at least one **corresponding** split
(i.e. a split for which there is a path
from either output to the input of the join)

proof:
we trace backward the paths
leading to the join from start events;
if the start events coincide there is a split node in the path;
if start events differ, the candidate split is the implicit XOR

Problem with corresponding splits

The semantics of a join often depends on whether or not it has a corresponding split

But:
1) there can be **more candidates** to corresponding split
2) and they can have **different type** than the join

candidates of the same type of the join are called **matching** split

Some suggested to have a flag that denotes the corresponding split

Tagging
corresponding splits

s1

s2

j1 {s1}

j2 {s2}
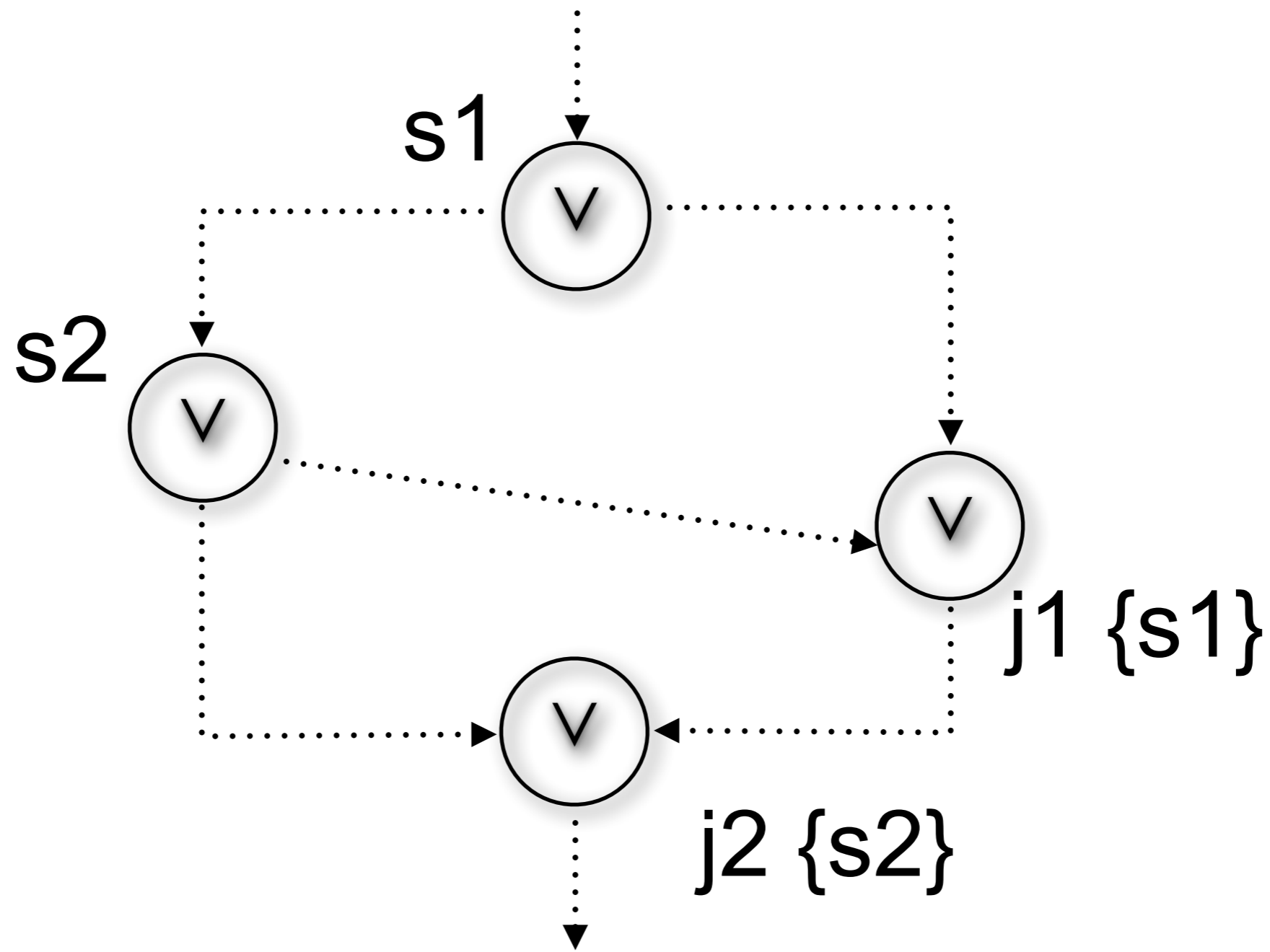
Problem with OR join

If an OR join has a matching split, the semantics is usually: "wait for the completion of all paths activated by the matching split"

If there is no matching split, some ambiguities can arise:

wait-for-all: wait for the completion of all *activated* paths (default semantics, because it coincides with that of a matched OR)

first-come: wait only for the path that is completed first and ignore the second

every-time: trigger the outgoing path on each completion (the outgoing path can be activated multiple times)

Some suggested to have different (trapezoid) symbols or suitable flags to distinguish the above cases and allow them all

Problem with XOR join

Similar considerations hold for the XOR join

If a XOR join has a matching split, the semantics is intuitive:
"it blocks if both paths are activated and it is triggered by the completion of a single activated path"

If there is no matching split:
all feasible interpretations that do not involve blocking are already covered by the OR (wait-for-all, first-come, every-time)
and **contradict the exclusivity** of the XOR
(a token from one path can be accepted only if we make sure that no second token will arrive via the other path)

Some suggest to just forbid the use of XOR in the unmatched case
(the implicit start split is allowed as a valid match)

Problem with alternation

Empirical studies have shown that middle and upper management people consider strict alternation between events and functions as too restrictive:
they find it hard to identify the necessary events at the abstract level of process description they are working at

It is safe to drop this requirement,
as dummy events might always be added later, if needed

Translation of EPC to Petri nets

Idea

We transform EPC diagrams to Workflow nets

We exploit the formal semantics of nets
to give unambiguous semantics to EPC diagrams

We apply the verification tools we have seen
to check if the net is sound:
the EPC diagram is sound if its net is so

A note about the transformation

We first transform each event, function and connector separately in small net fragments

When translating the control flow arcs we may then introduce other places / transitions to preserve the bipartite structure in the net
(no arc allowed between two places,
no arc allowed between two transitions)

We show two translations, depending on whether the join are decorated or not

First attempt (decorated EPC)

EPC
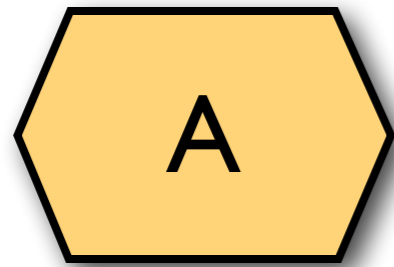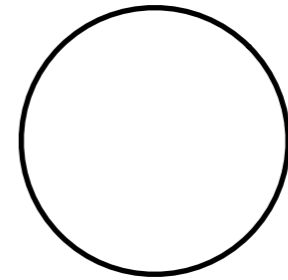
Petri net

A

event

place

EPC

Petri net

function

transition

EPC

Petri net

AND split

net

EPC

Petri net

OR split

net

EPC
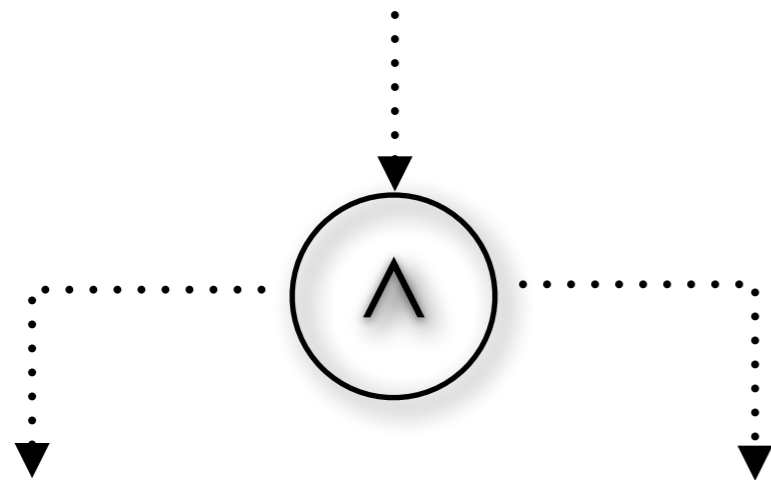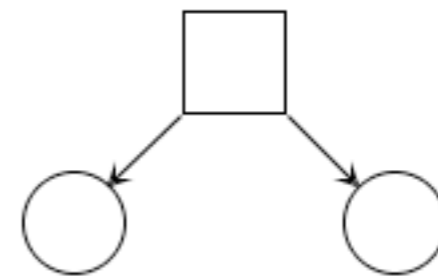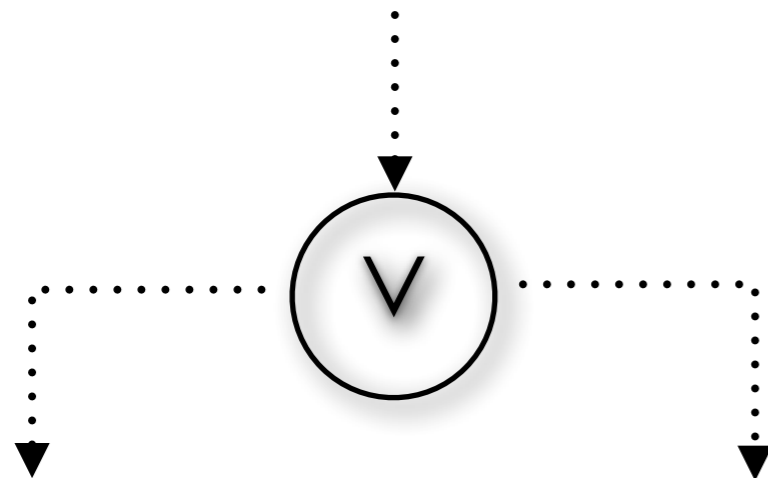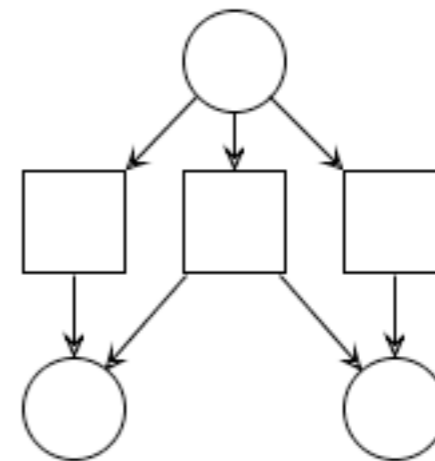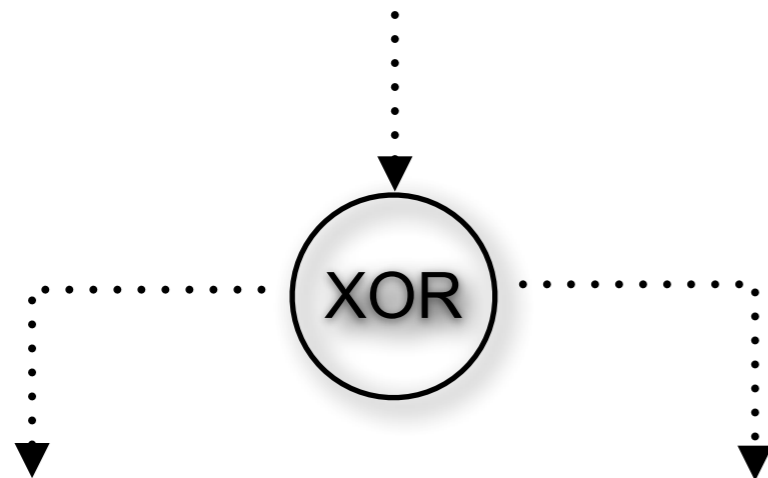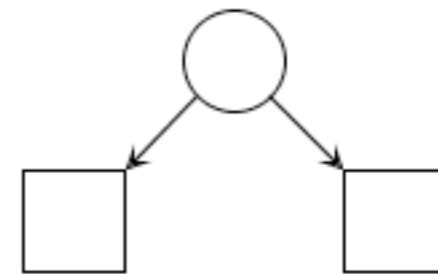
Petri net

XOR split

net

EPC
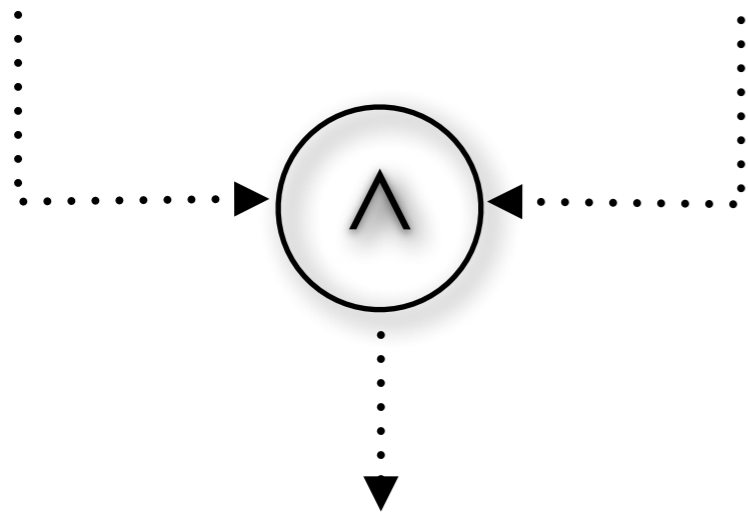
Petri net

AND join

net

EPC

Petri net

corresponding split

? ? ?

XOR

XOR join

net

49

EPC
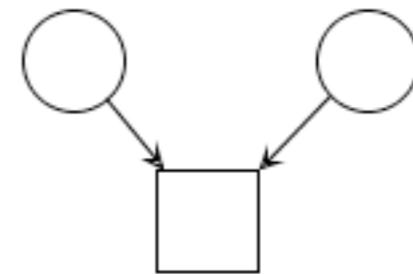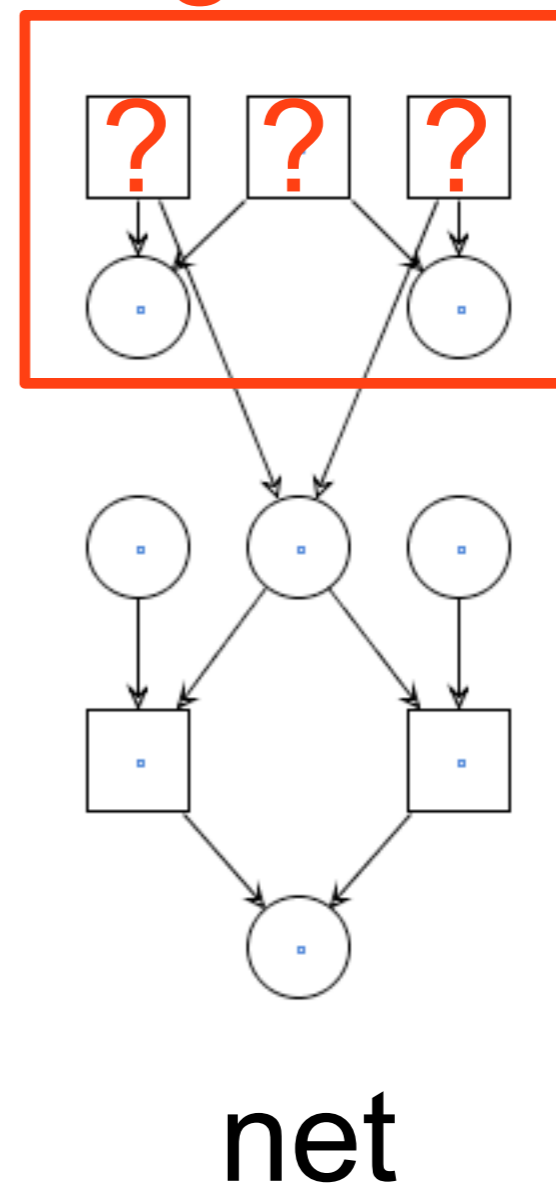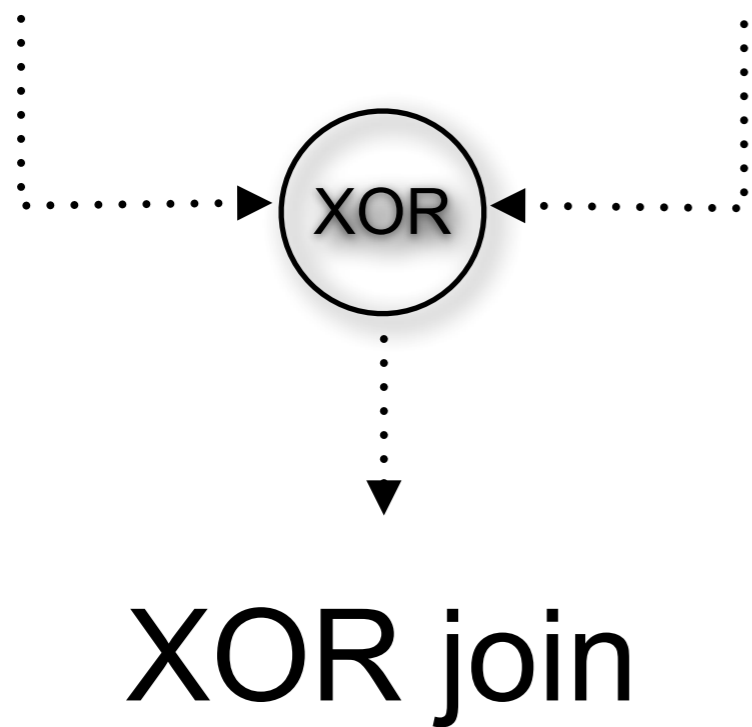
Petri net

corresponding
XOR/OR split

?

ok

net

XOR

XOR join

EPC
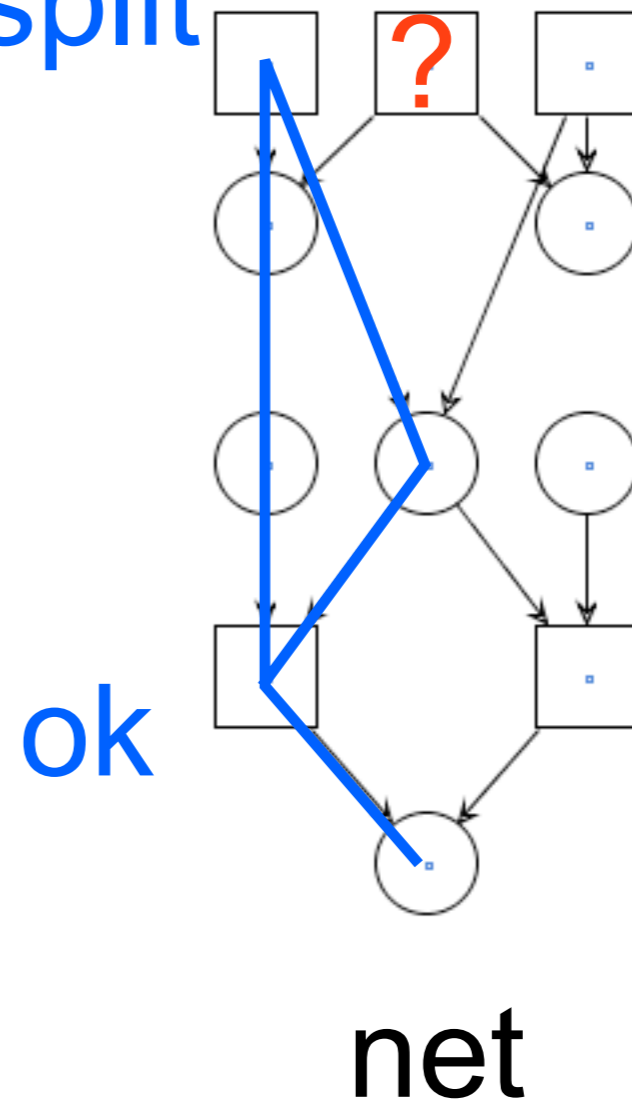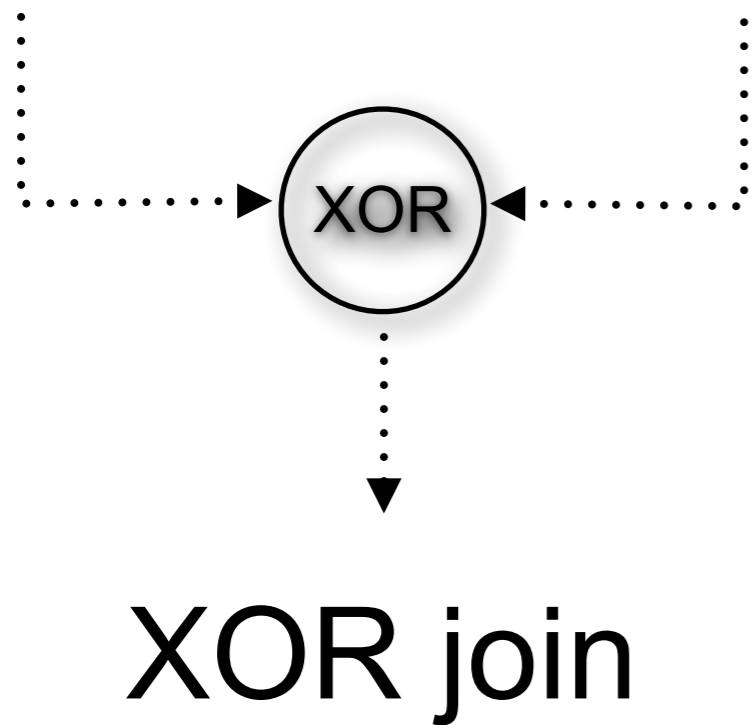
Petri net

corresponding
AND/OR split
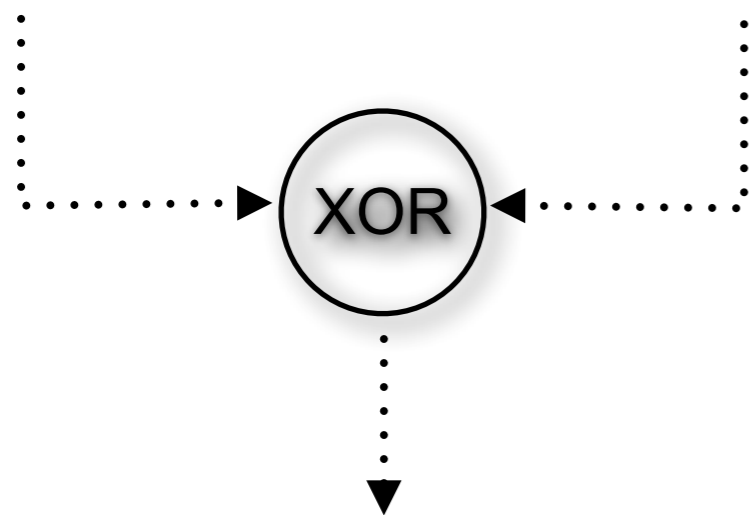
XOR
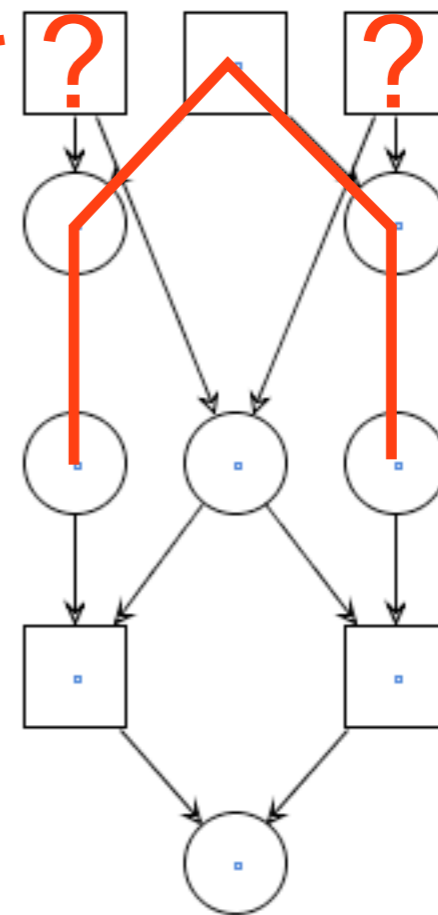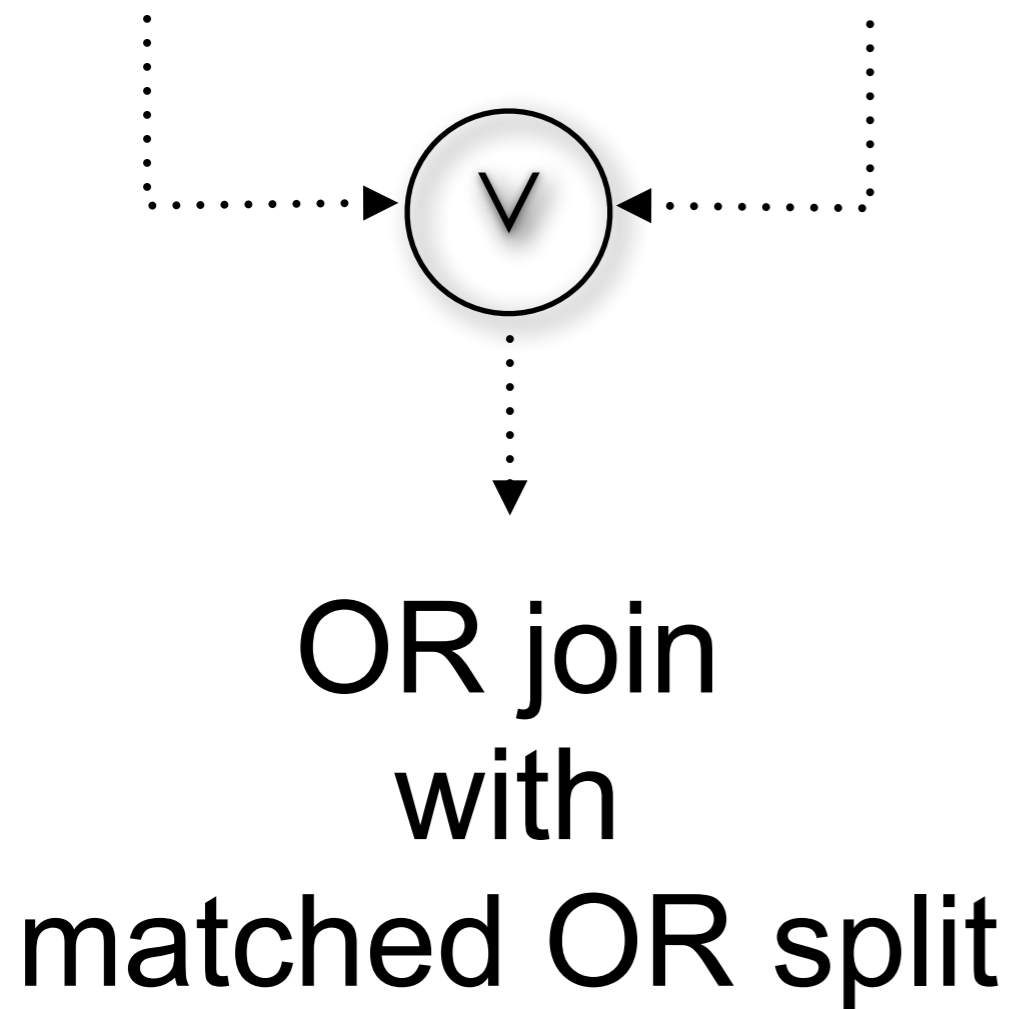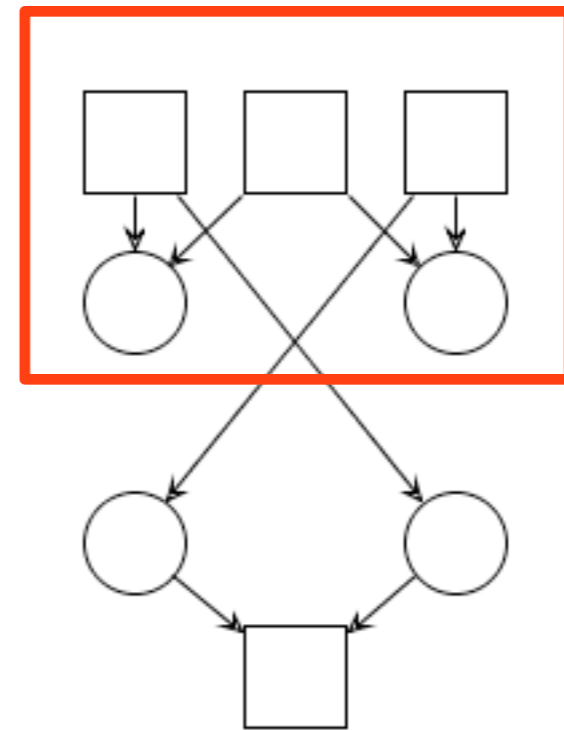
XOR join

? ?

dangling
tokens!

deadlock!

net

EPC

Petri net

OR join
with
matched OR split

net

52

EPC

Petri net

corresponding split

wfa

∨

OR join
wait-for-all
(unmatched)

? ? ?

net

53

EPC

Petri net

corresponding
AND split

wfa

∨
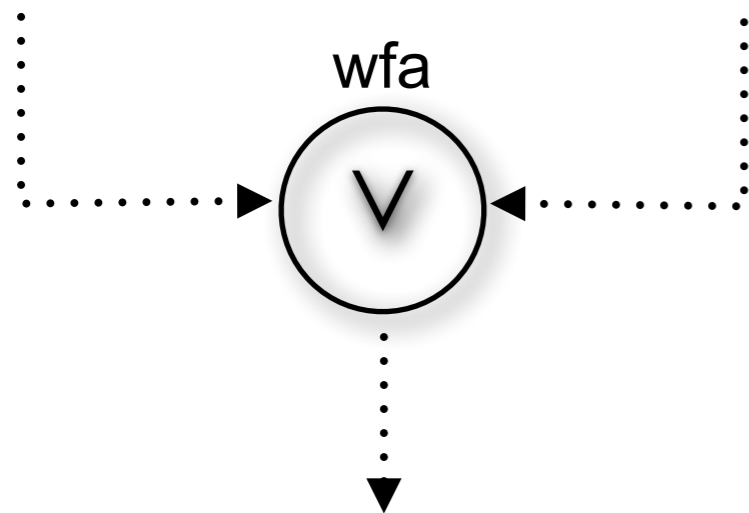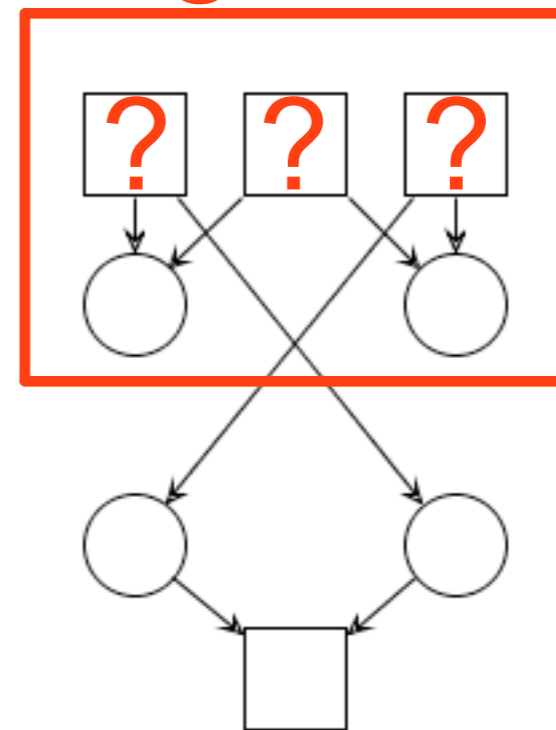
net

OR join
wait-for-all
(unmatched)

EPC

Petri net

corresponding
XOR split

wfa

∨
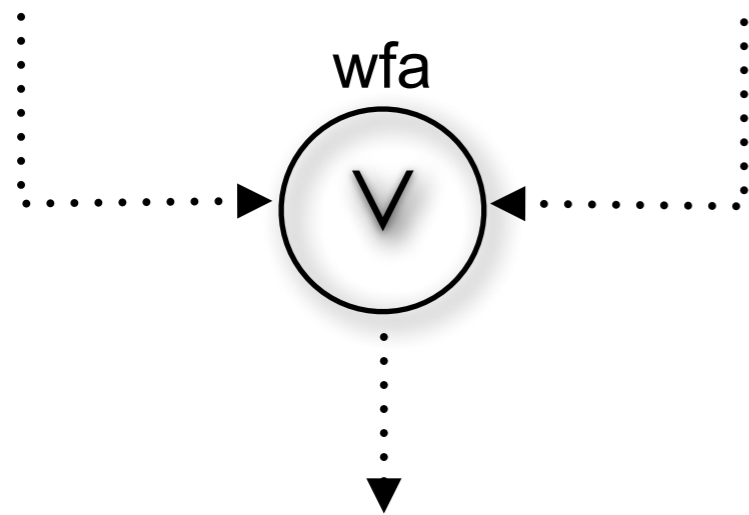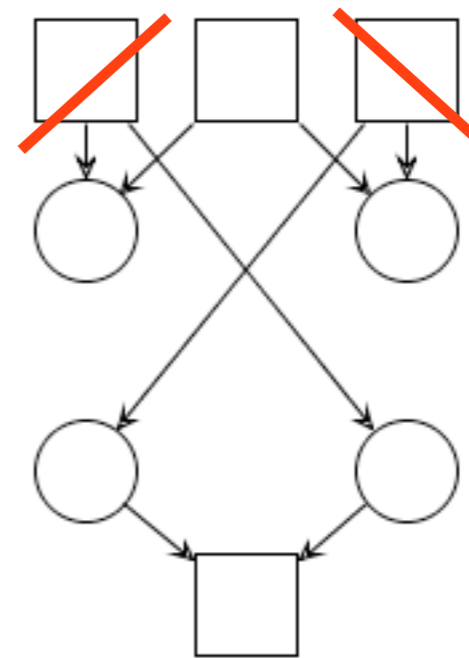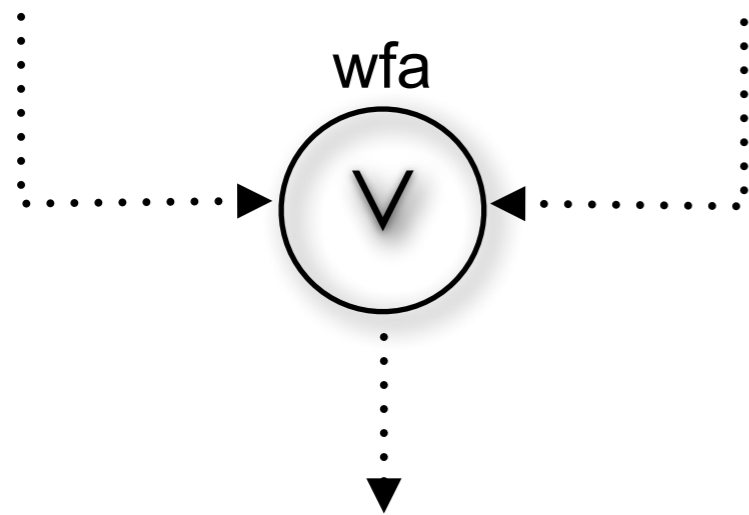
OR join
wait-for-all
(unmatched)

net

EPC

Petri net

corresponding split

? ? ?

fc

∨

OR join
first-come
(unmatched)

net

EPC

Petri net

corresponding
XOR split

fc

∨
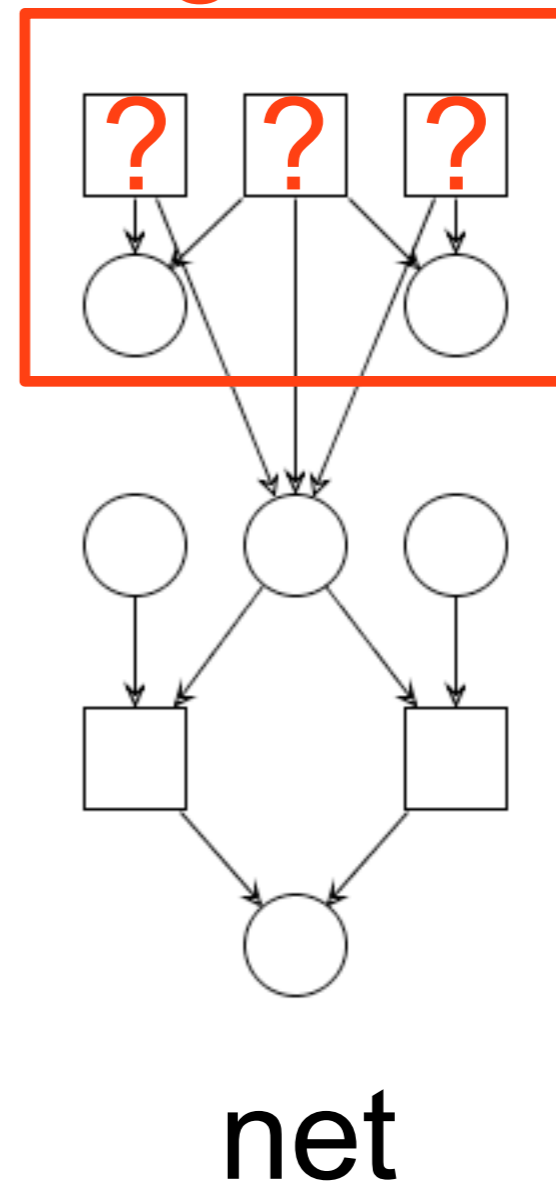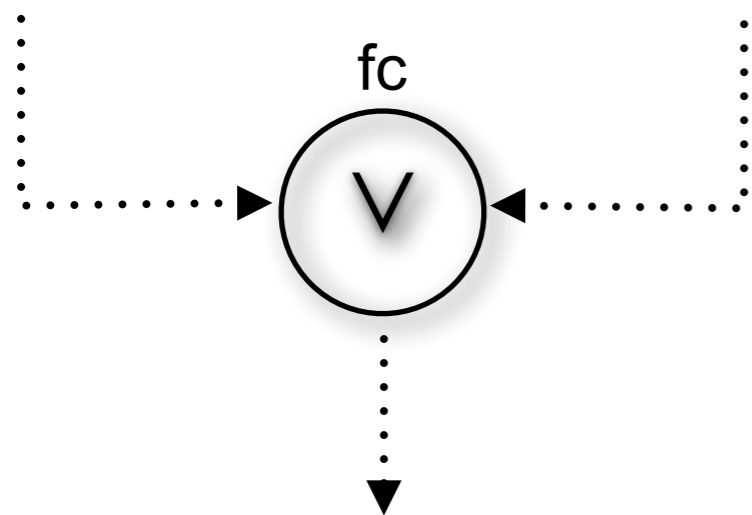
OR join
first-come
(unmatched)

ok

net

EPC

Petri net

corresponding
AND split

fc
∨
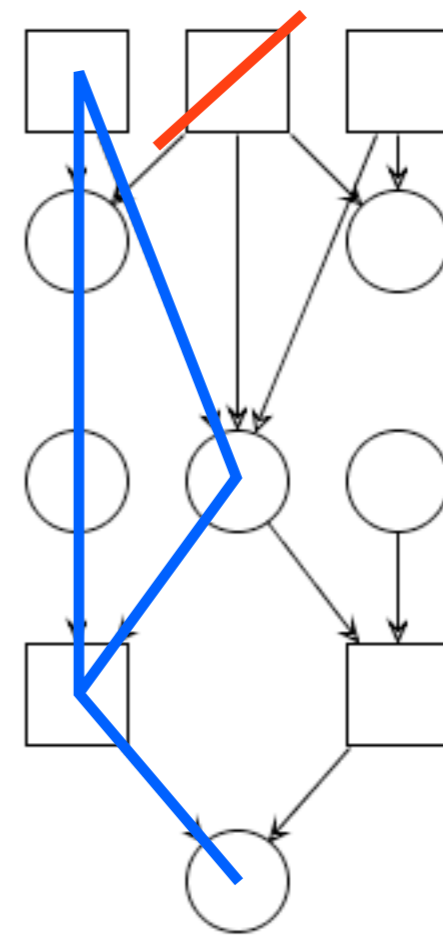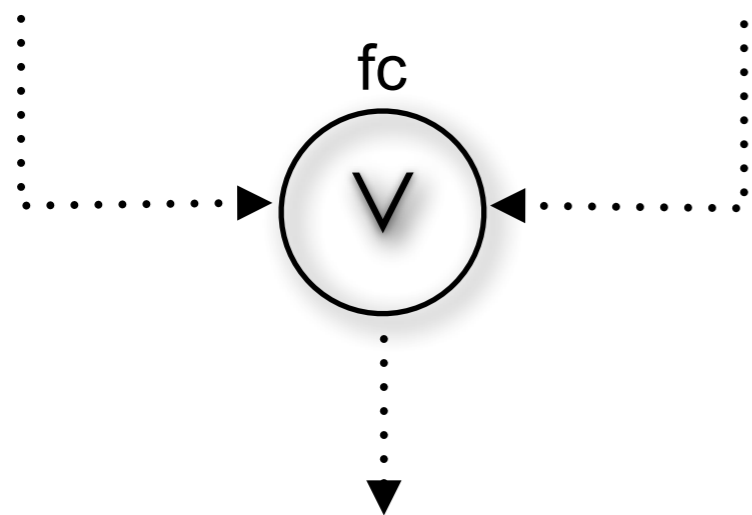
OR join
first-come
(unmatched)
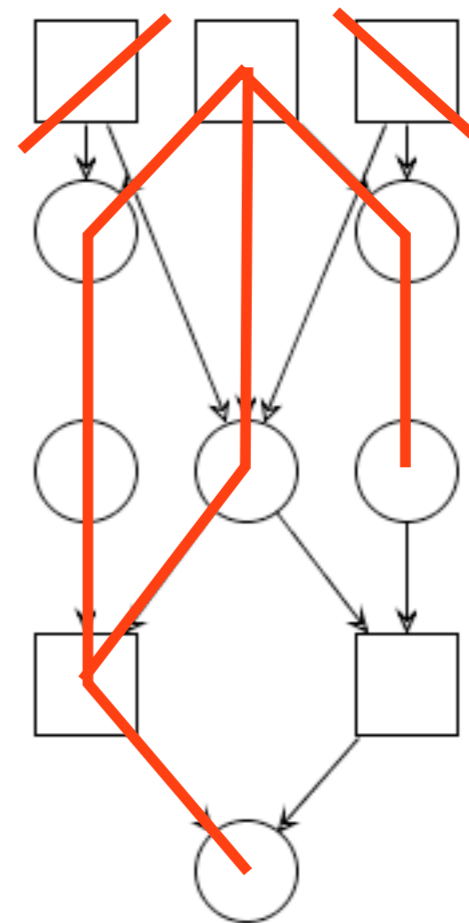
dangling
token!

net

EPC

Petri net

et

v
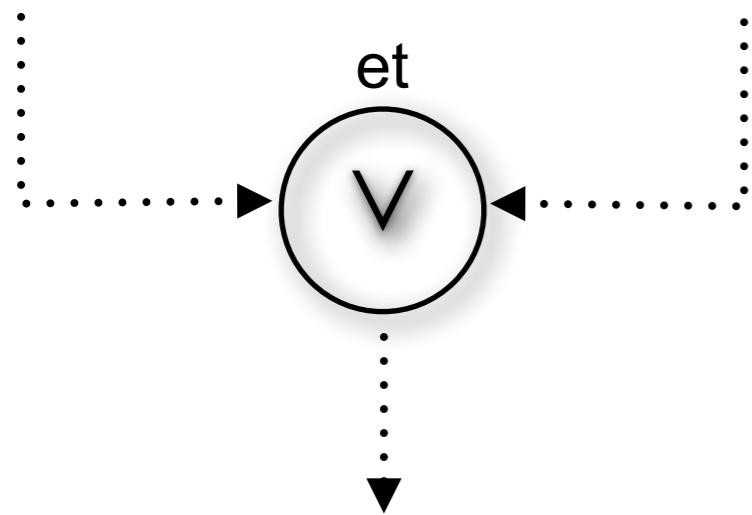
OR join
every-time
(unmatched)

net

EPC
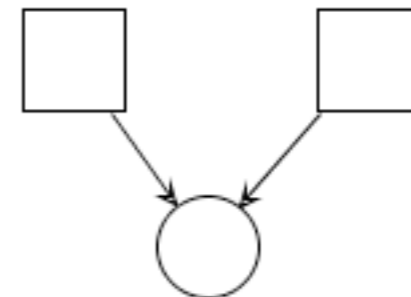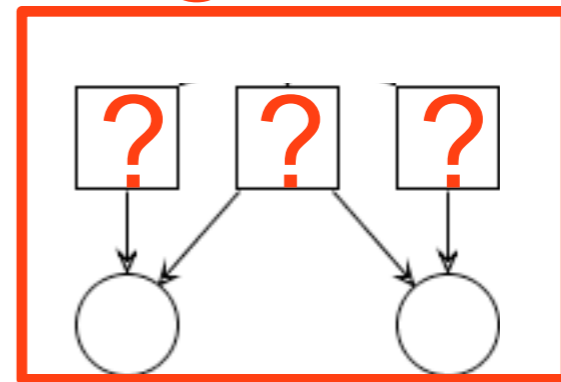
Petri net

corresponding
XOR split

ok

net

OR join
every-time
(unmatched)

EPC

Petri net

corresponding
AND split

et

∨

two
tokens!

net
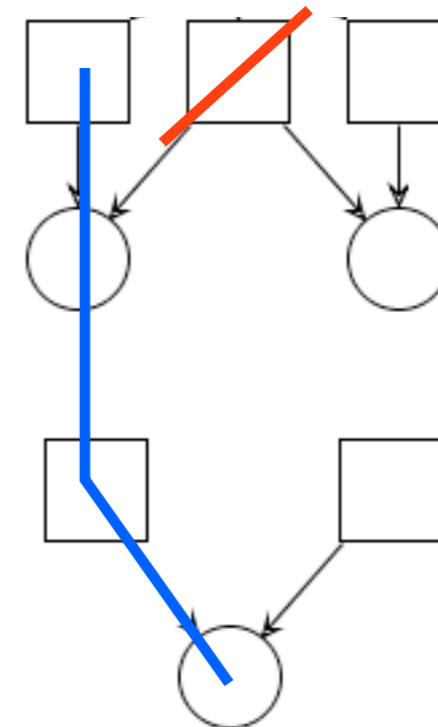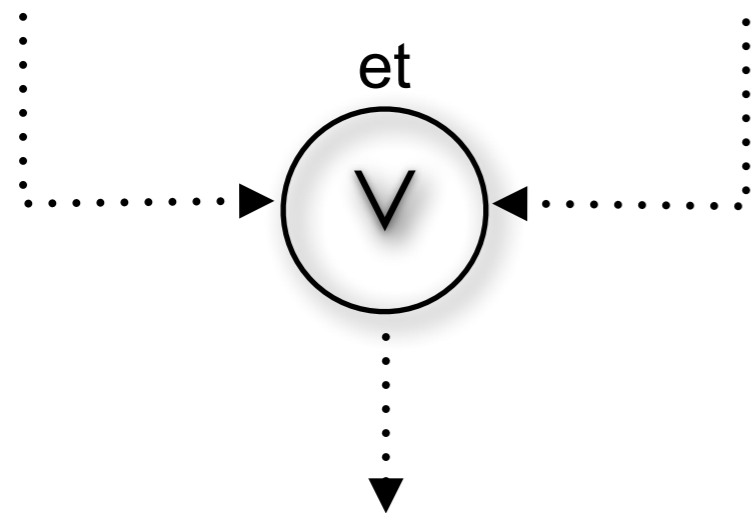
OR join
every-time
(unmatched)
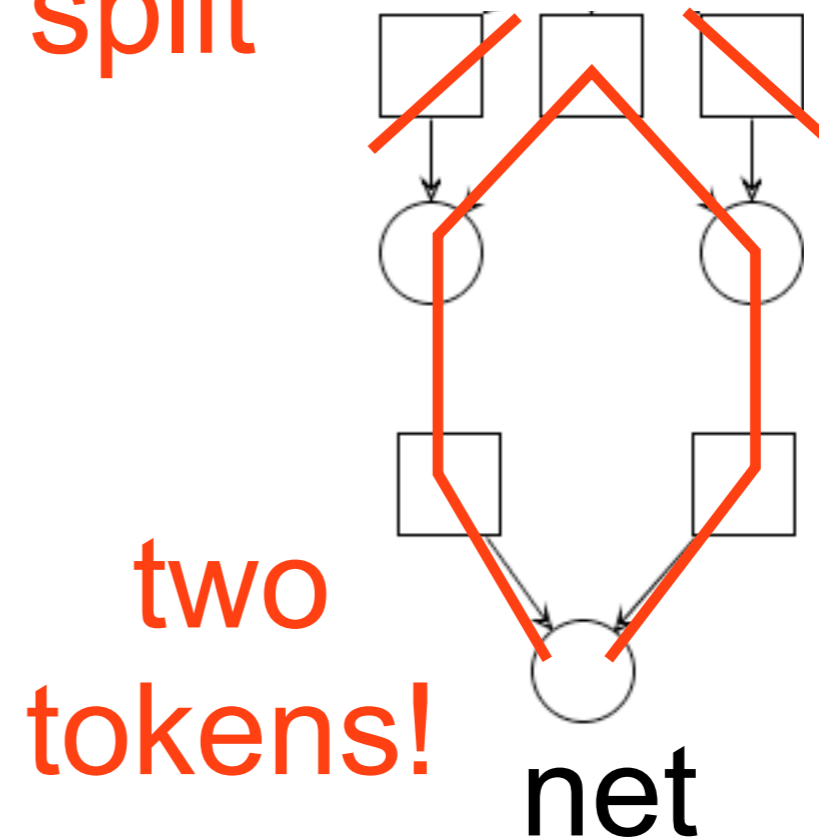
split
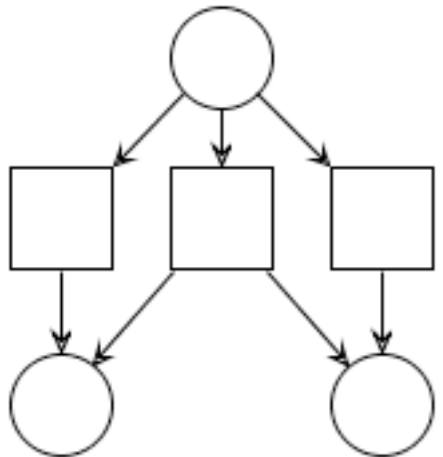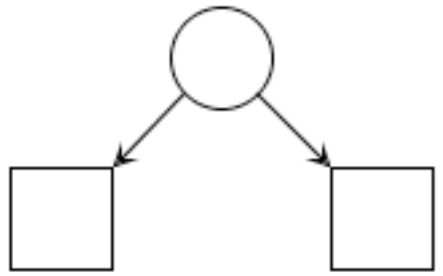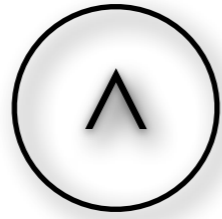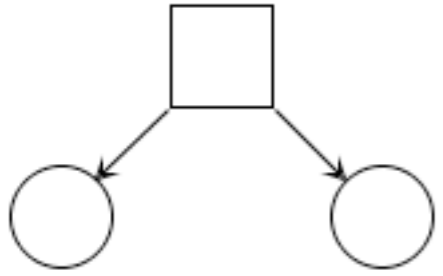
join

∧

∨
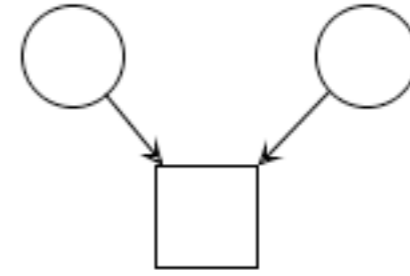
corresp.
split
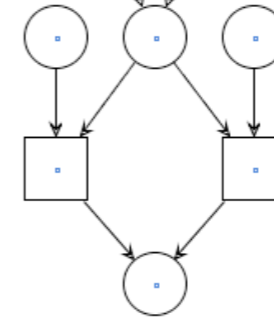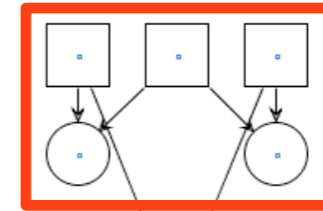
matched
OR split

corresp.
split: wfa
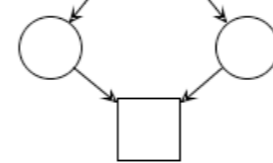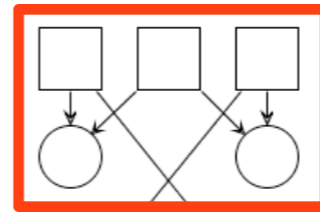
corresp.
split: fc

corresp.
split: et

XOR

62

Ill-formed net

Petri net

dummy transition
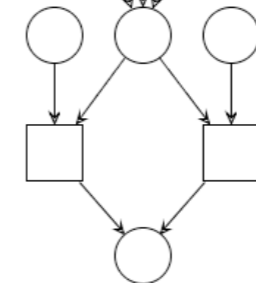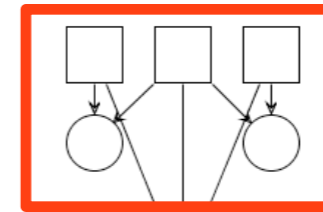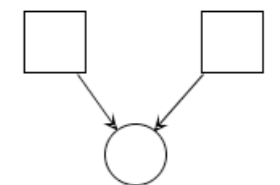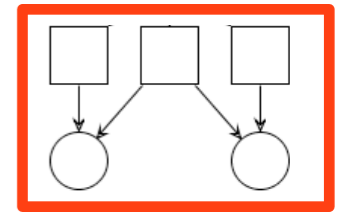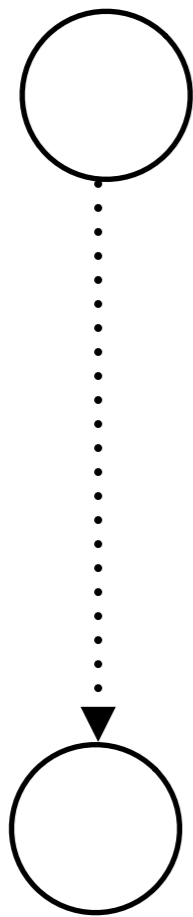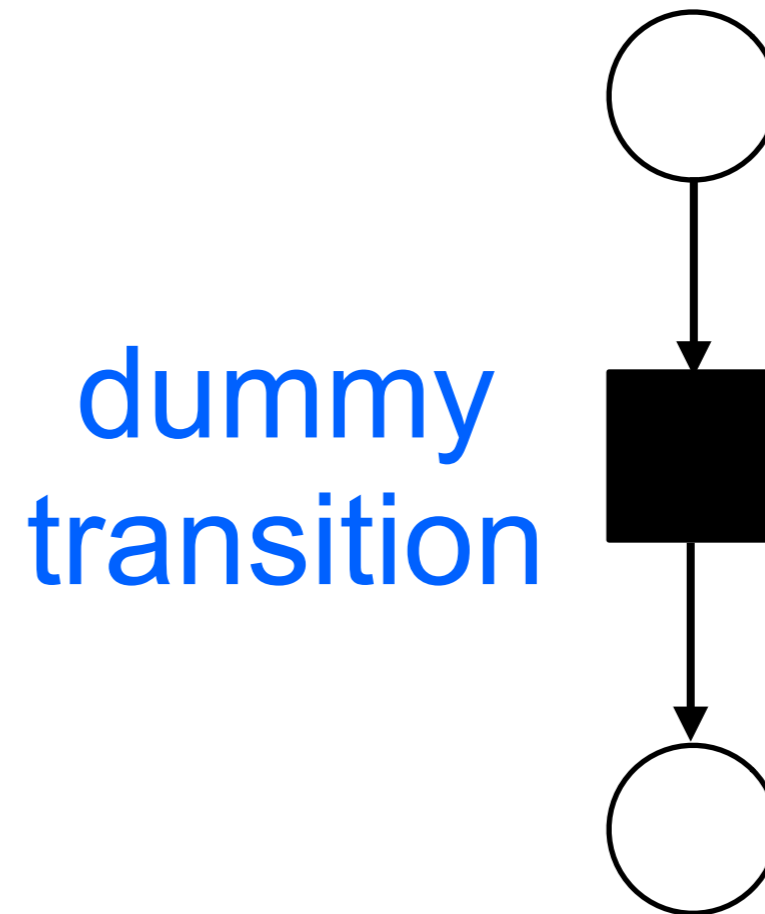
Ill-formed net
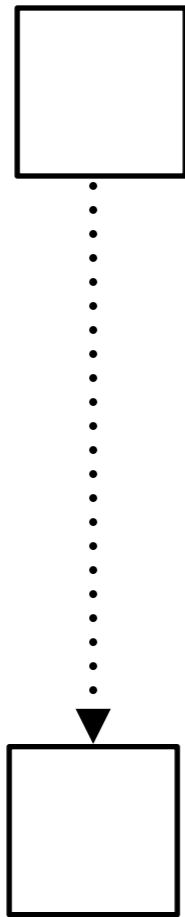
Petri net

dummy place

Exercise

Sound?

ZOOM IN

B → b → c → C → D

(first-come)

J1

A → a → S1 → H → e → J1 ∨ → F

f → S2 → G → d

J1 ∨ → XOR → E

J0

66
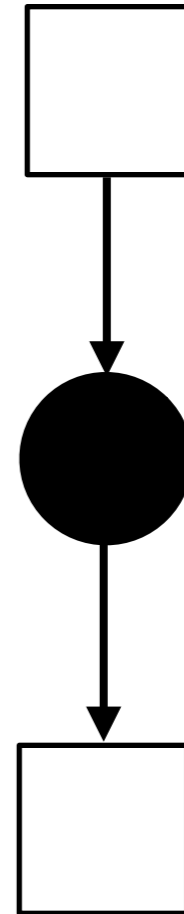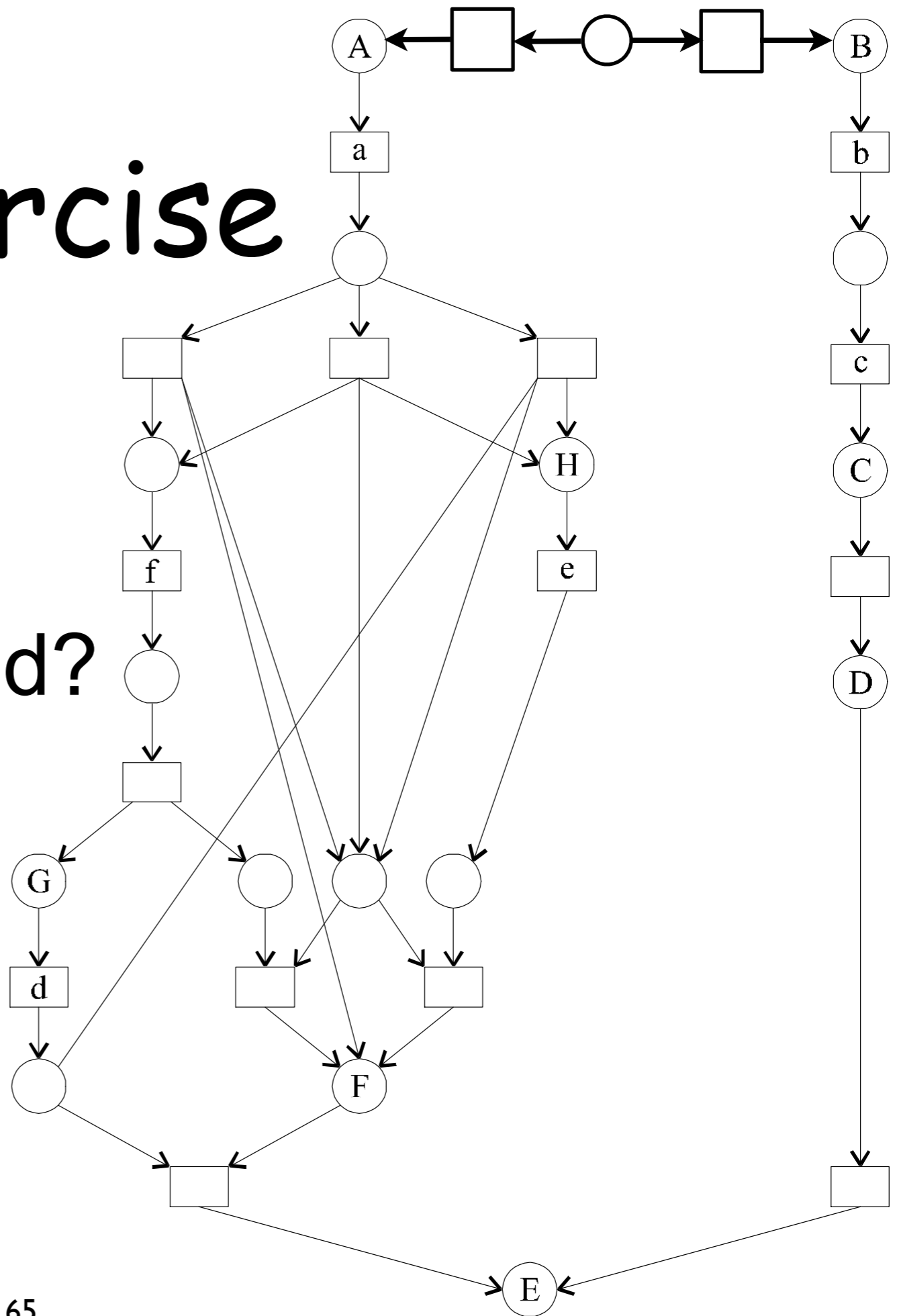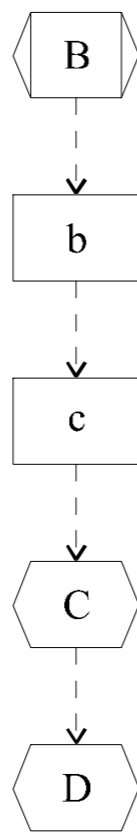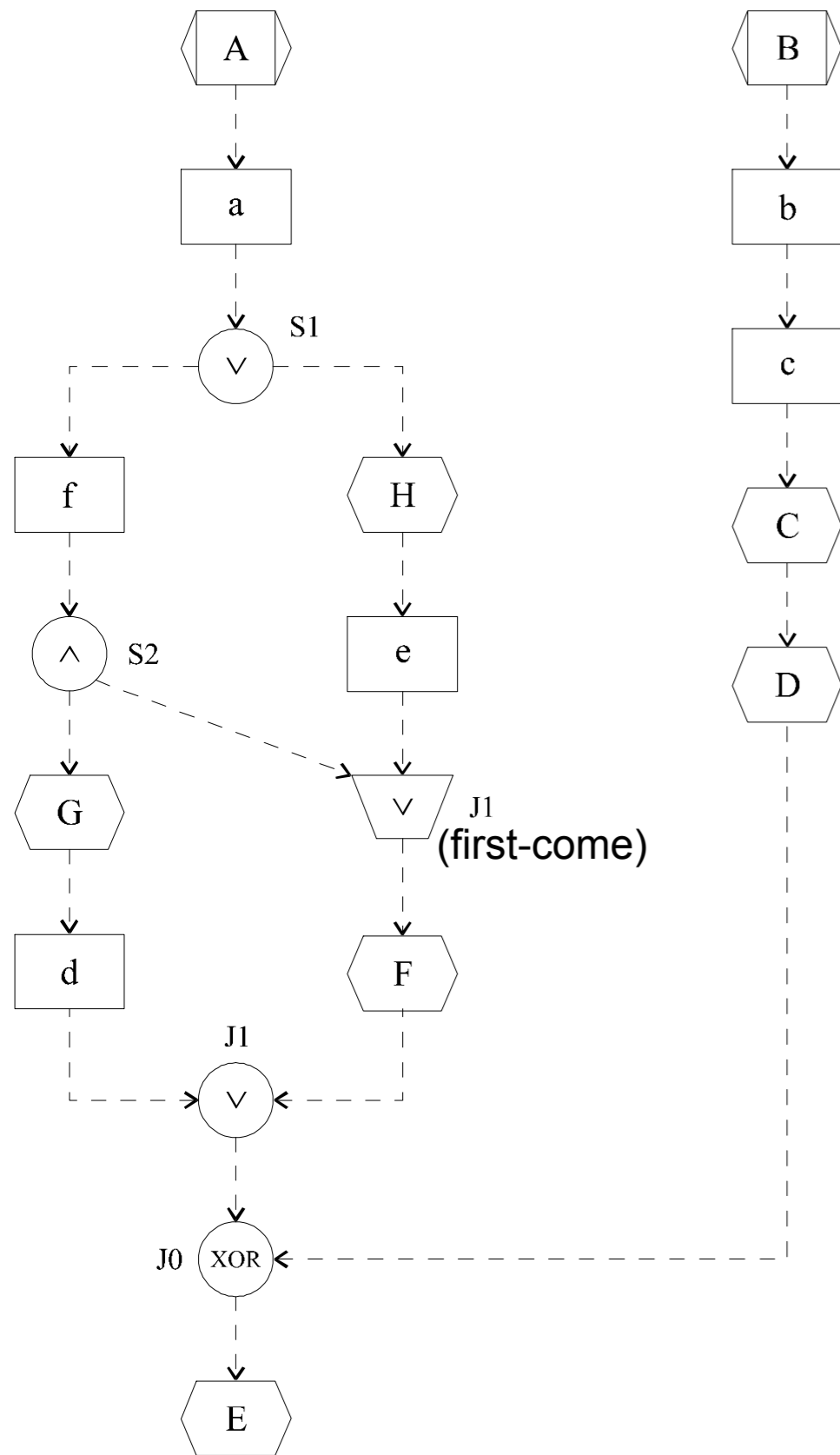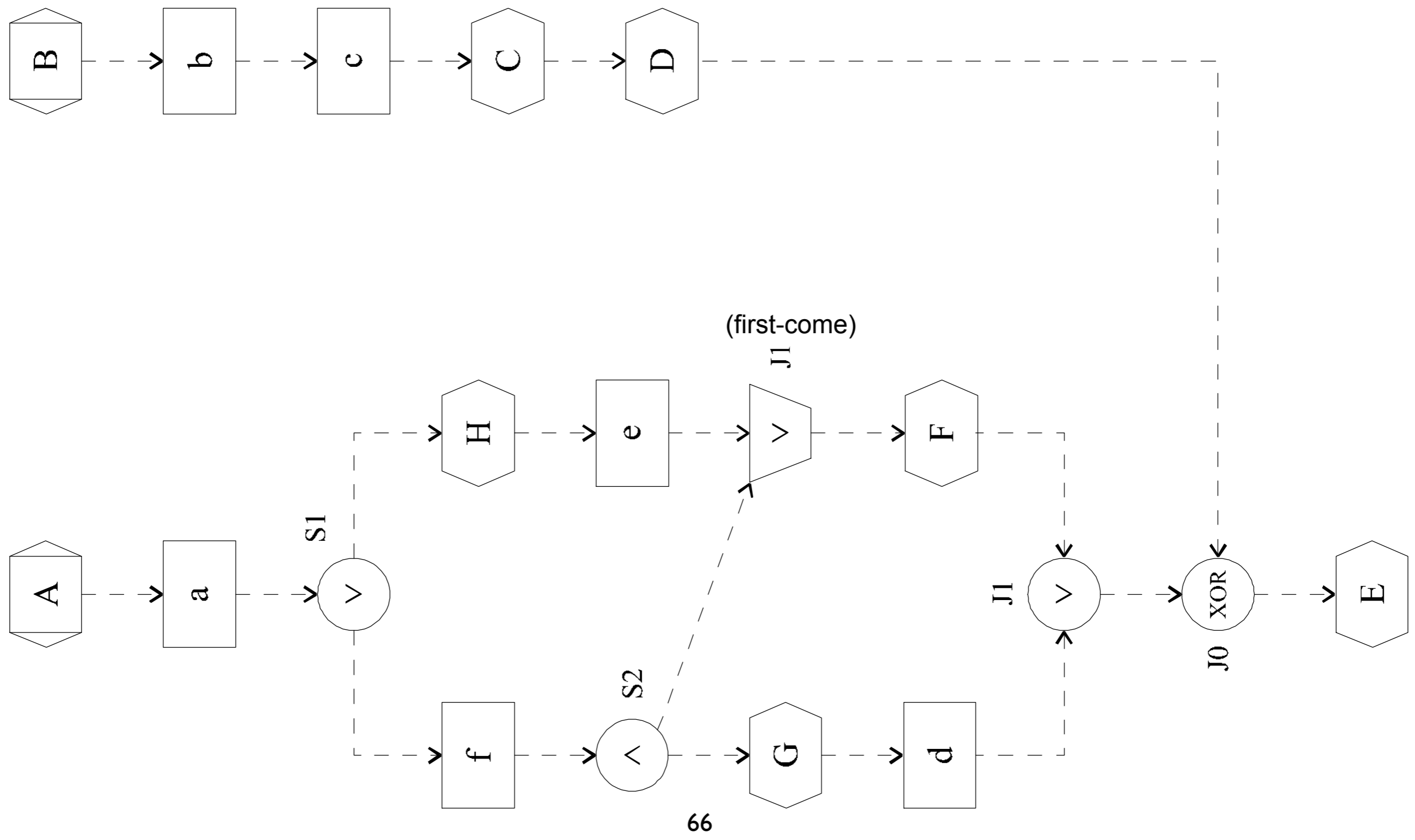
Exercise
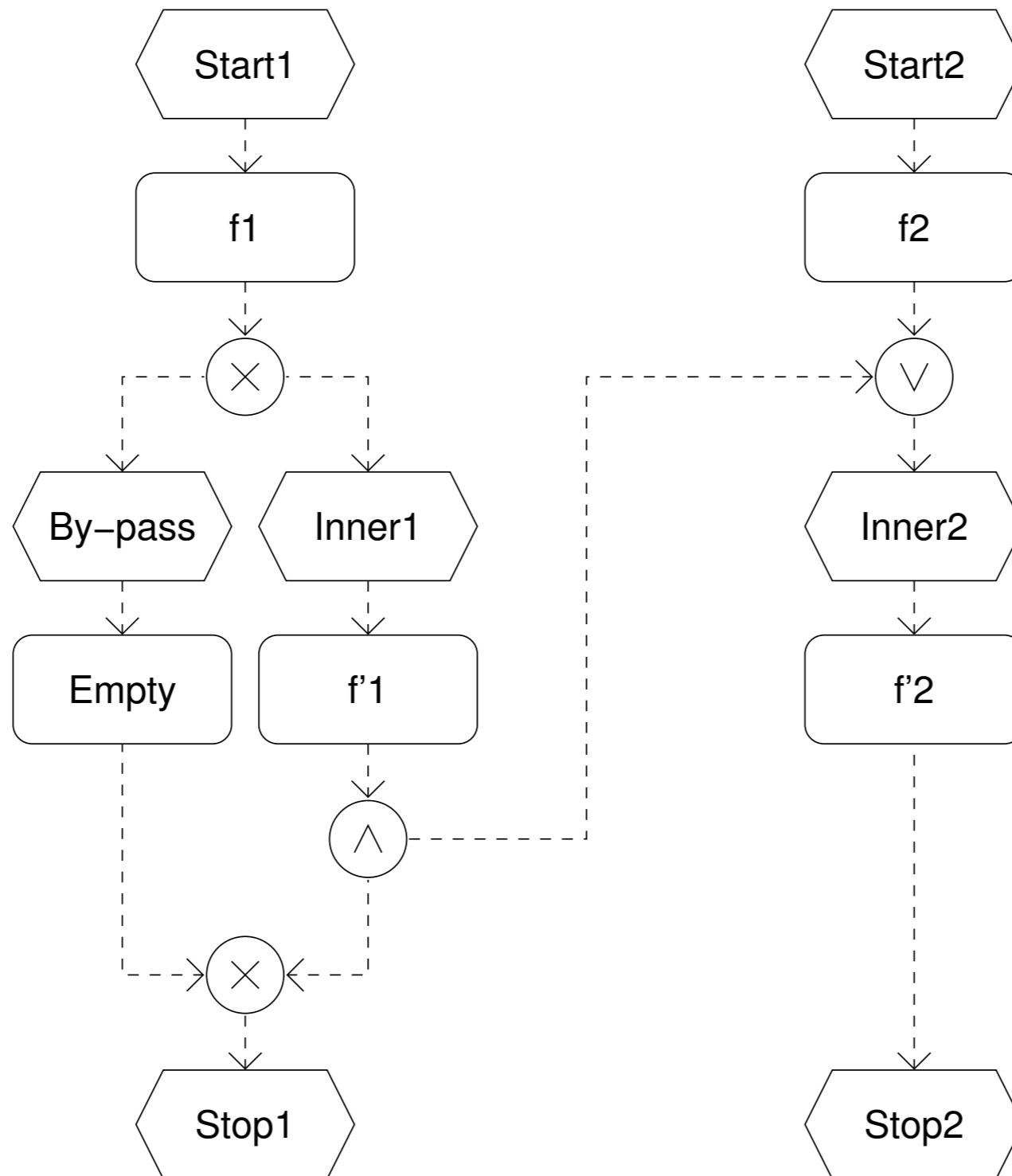
Sound?

Summary of possible problems

We need to decorate the EPC diagram
OR-join decorated with corresponding splits
OR-join decorated with policies (unmatched split)

Split / join mismatch may induce unexpected behaviour

Possible introduction of dummy places and transitions

Second attempt
(no decoration available)

Simplified EPC

We rely on event / function alternation
along paths in the diagram
and also **along paths between two connectors**
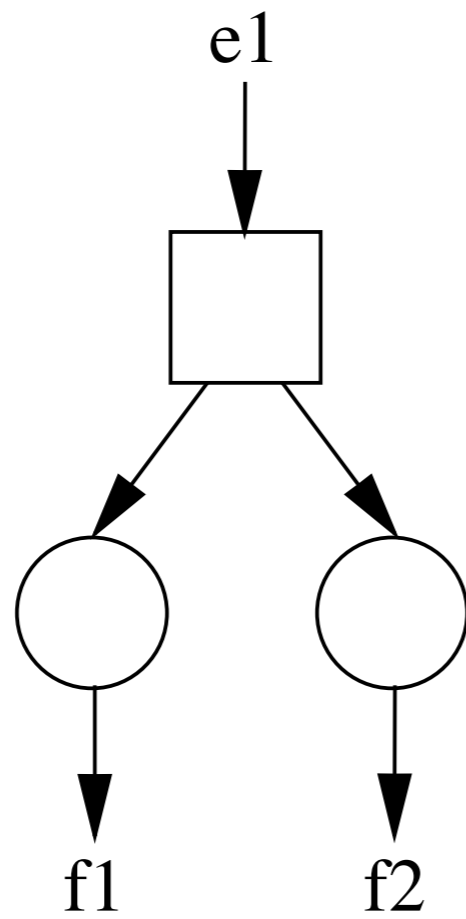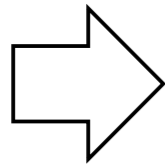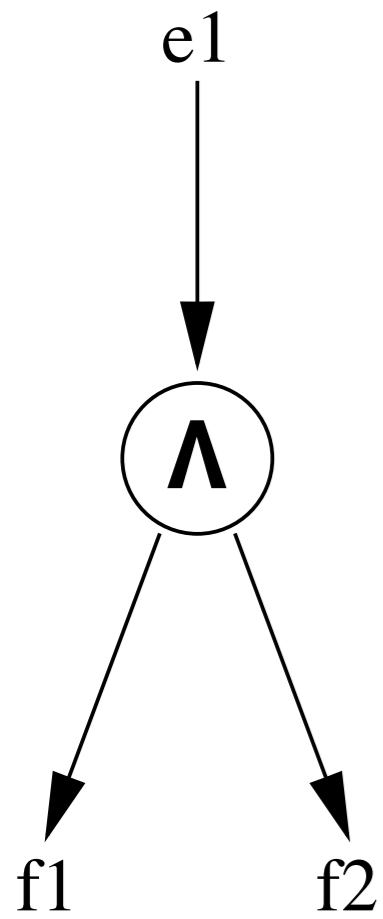
OR-connectors are not considered

EPC 2 Petri nets

event
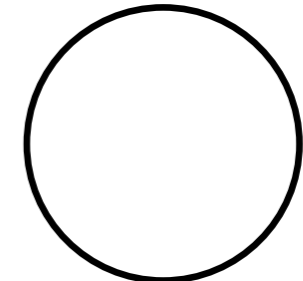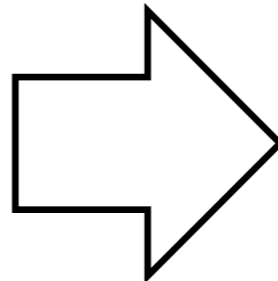
place

function
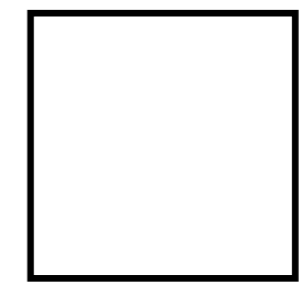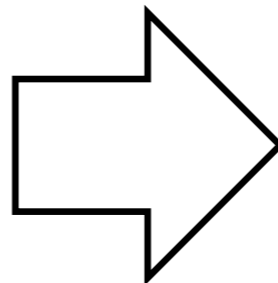
transition

EPC 2 Petri nets

e1   e2   e1   e2   f1   f2   f1   f2
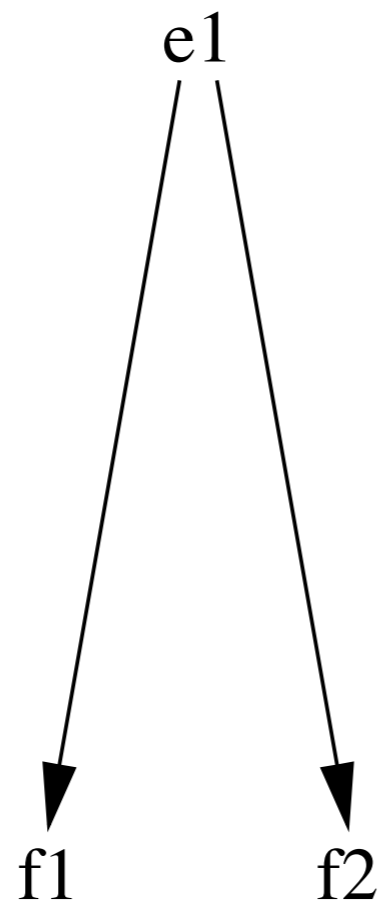
∧   ⇨   ∧   ⇨
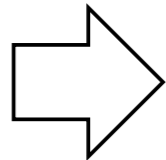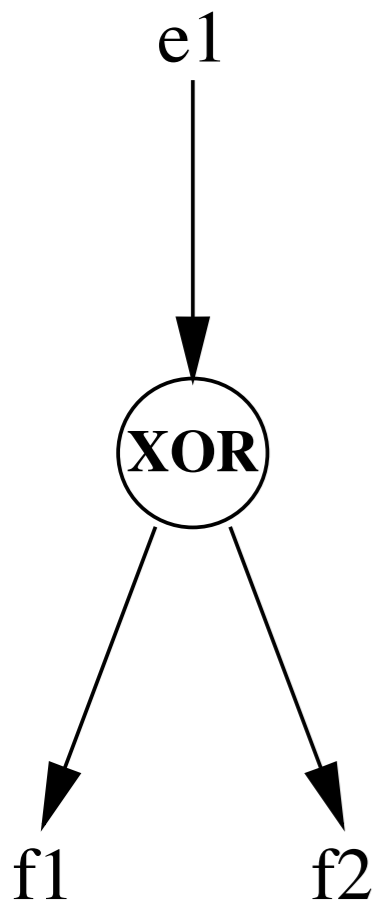
f1   f1   e1   e1
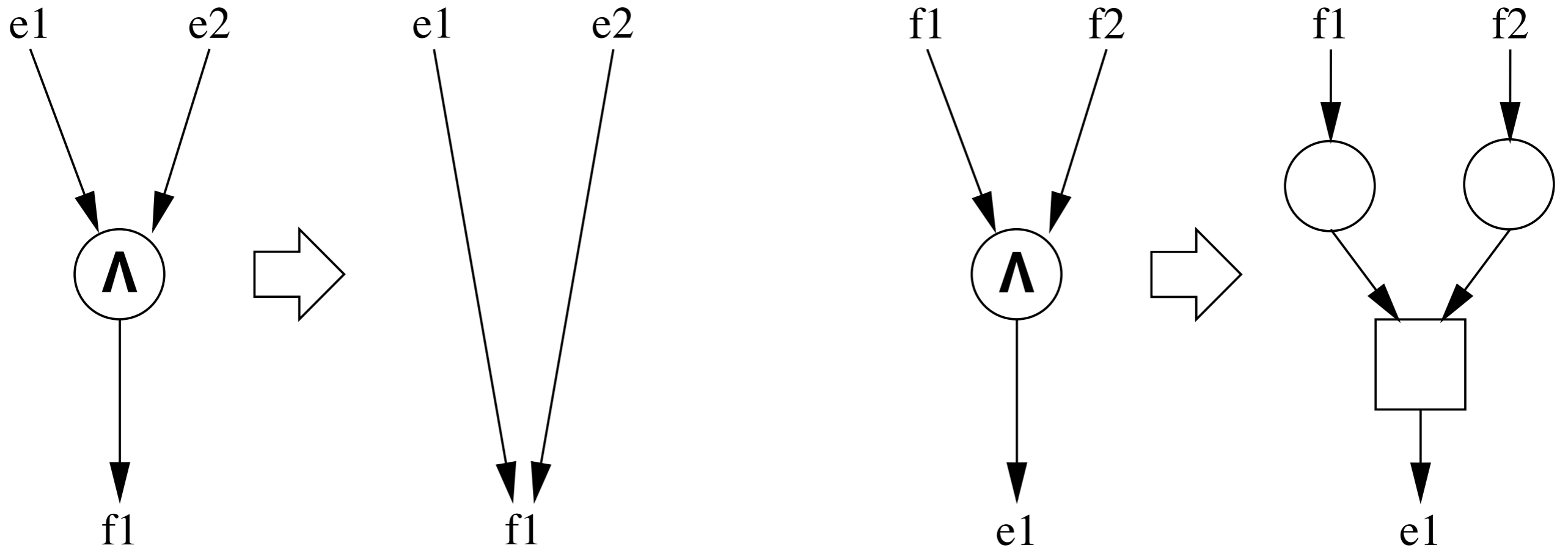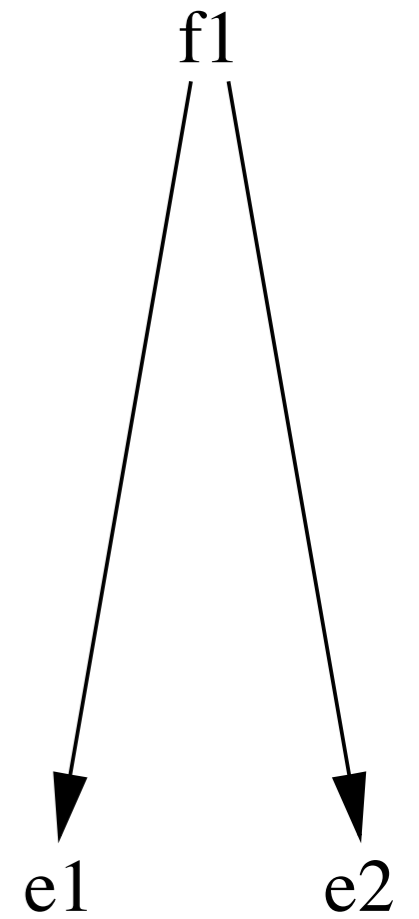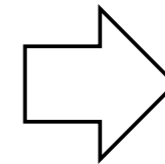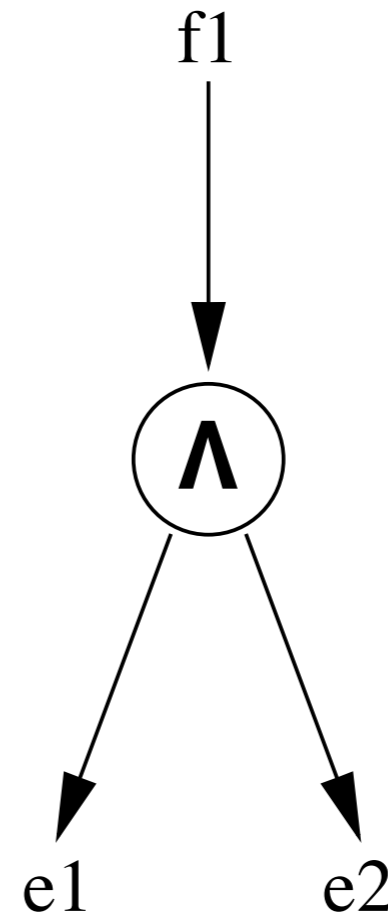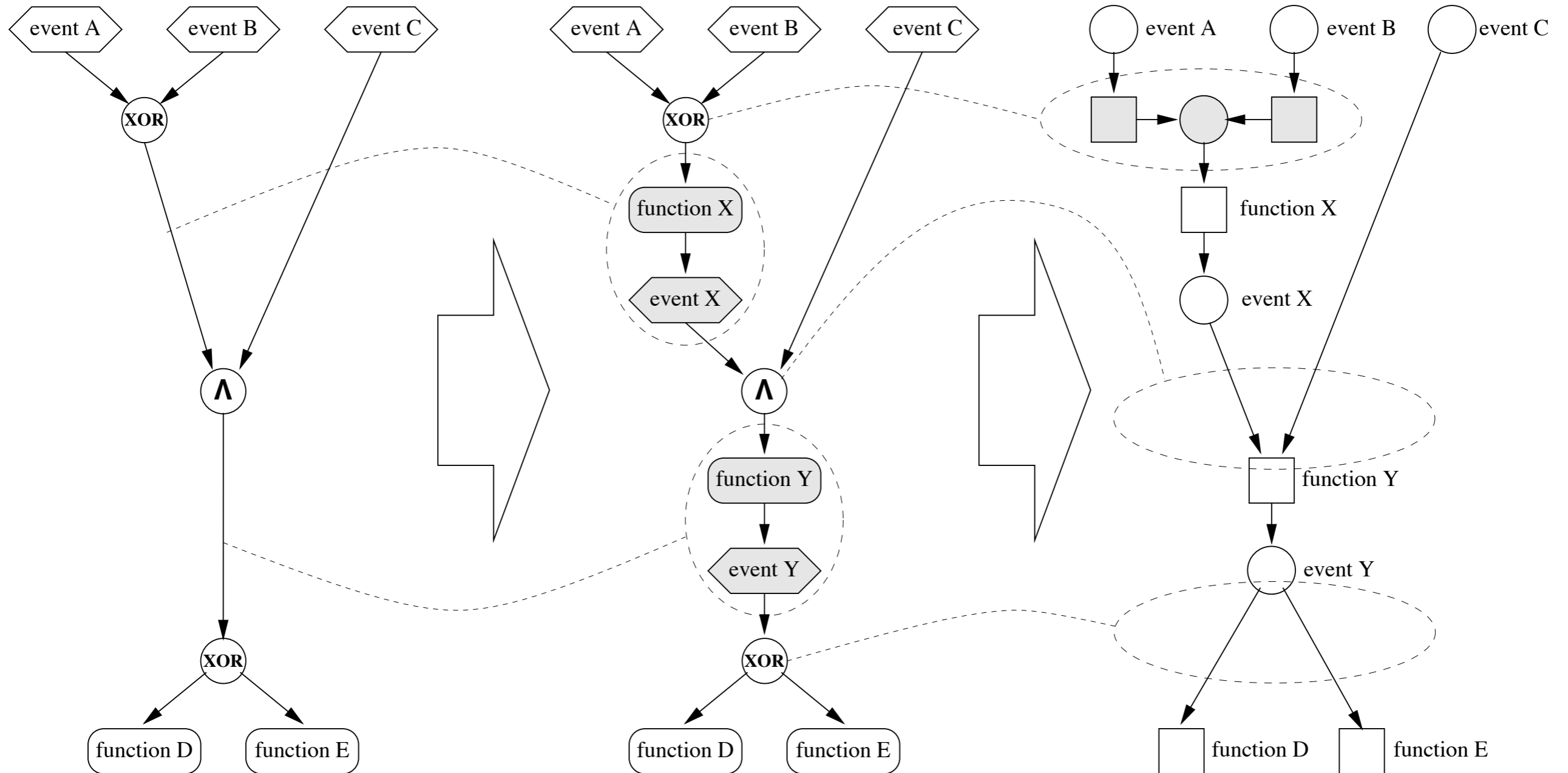
EPC 2 Petri nets
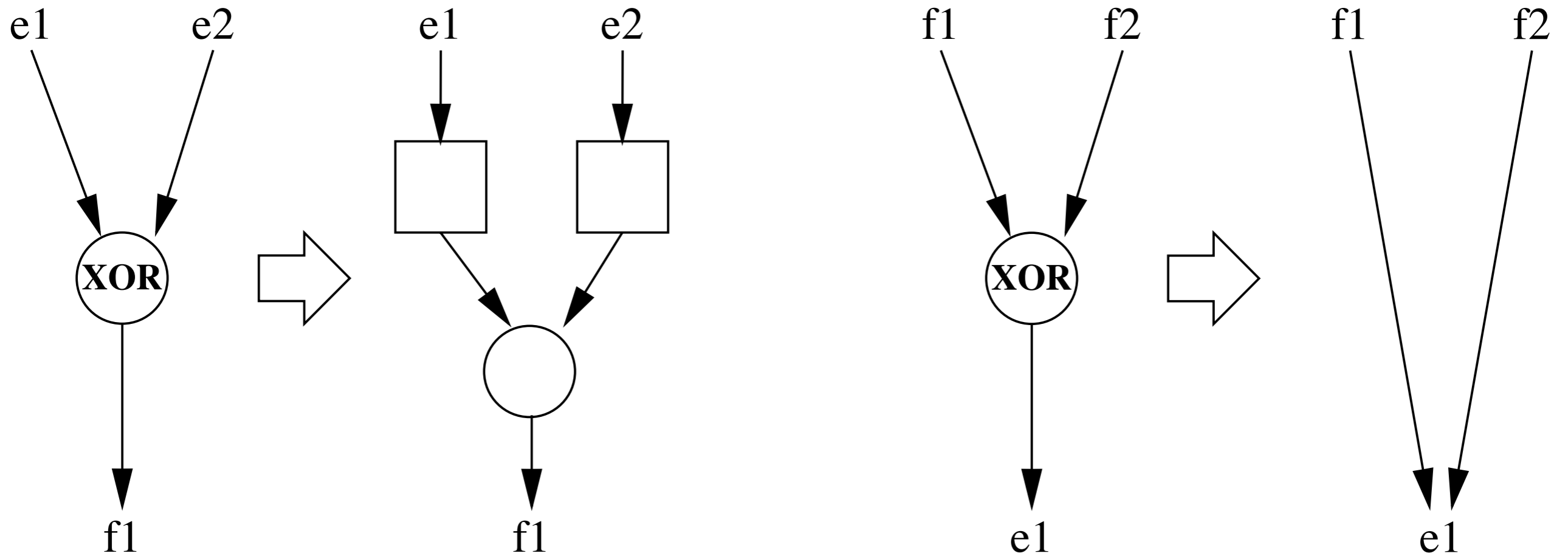
EPC 2 Petri nets

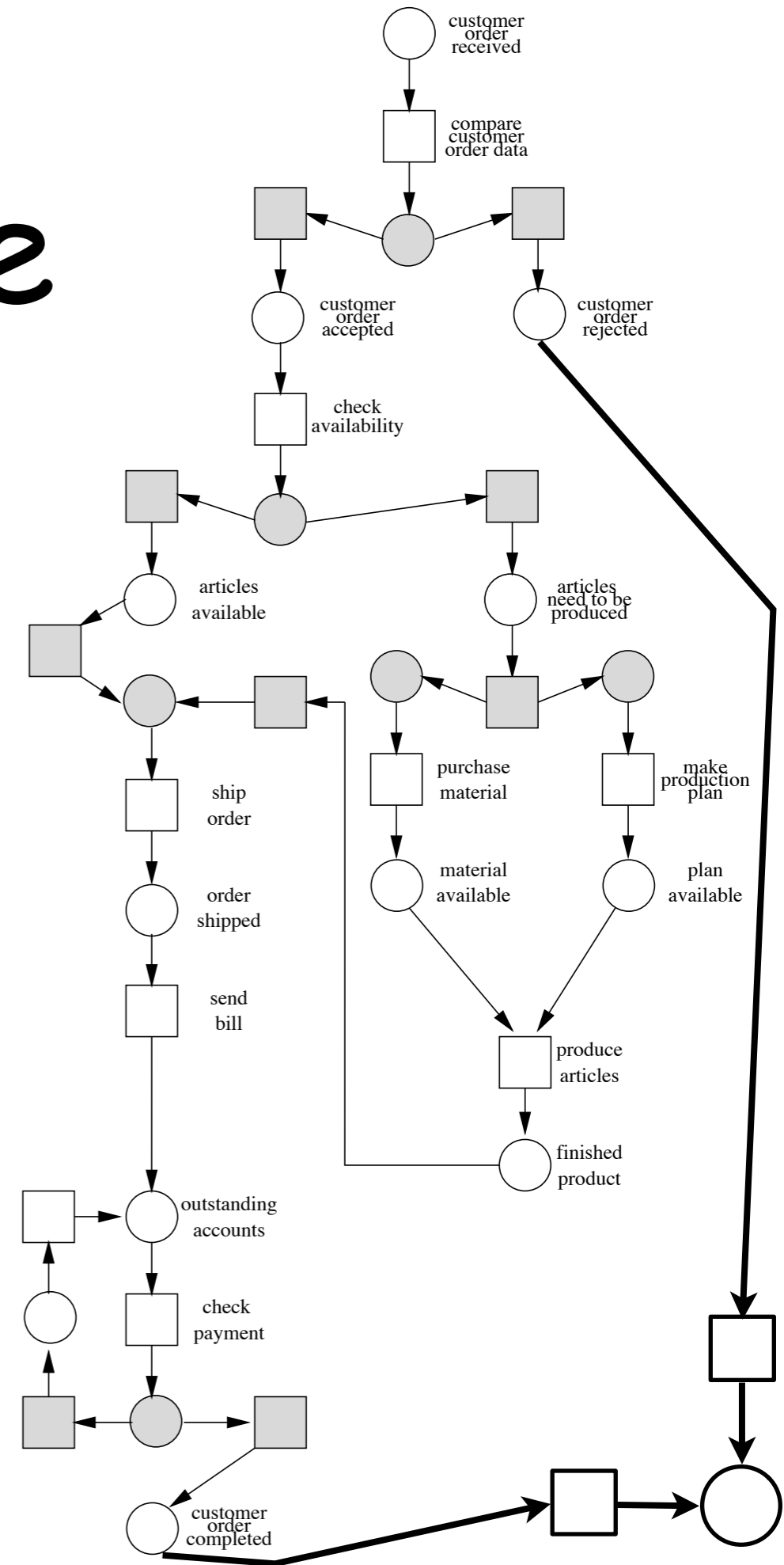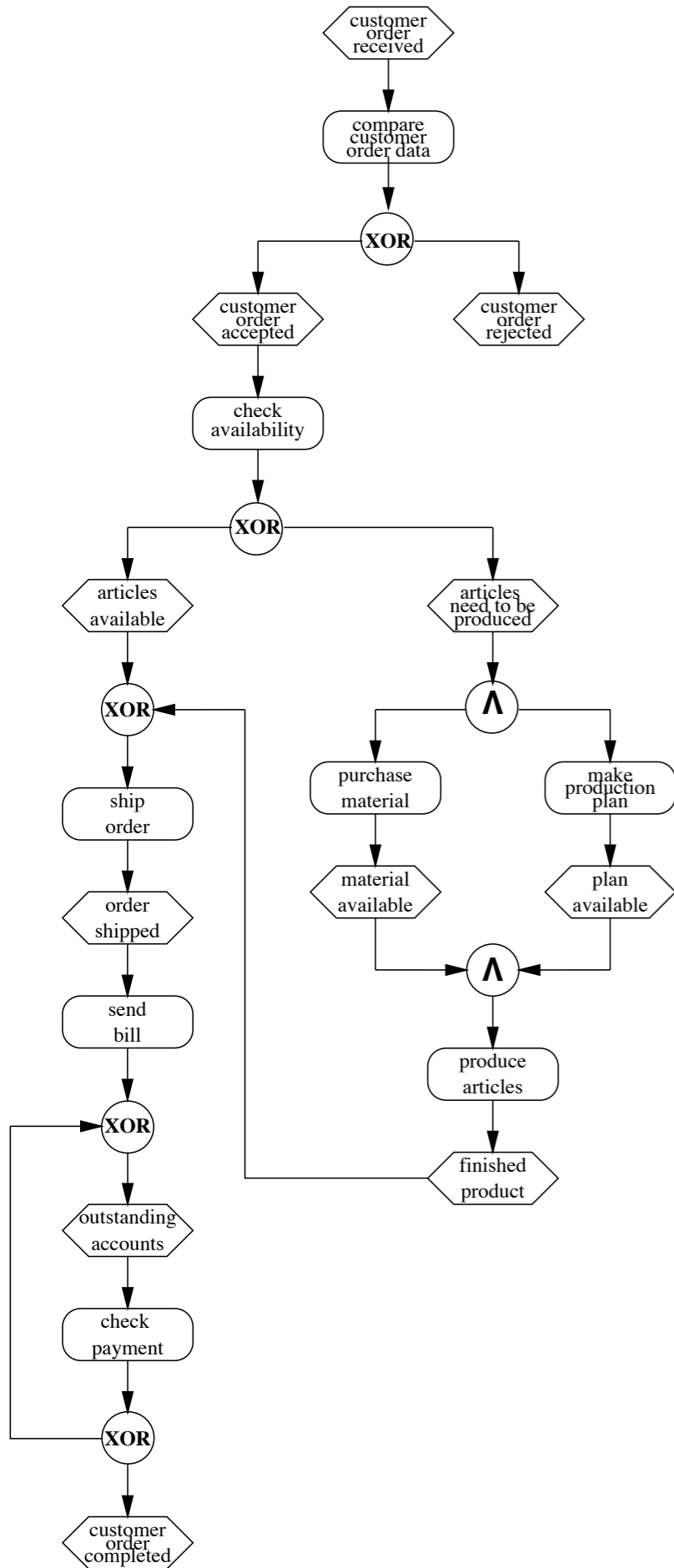# EPC 2 Petri nets
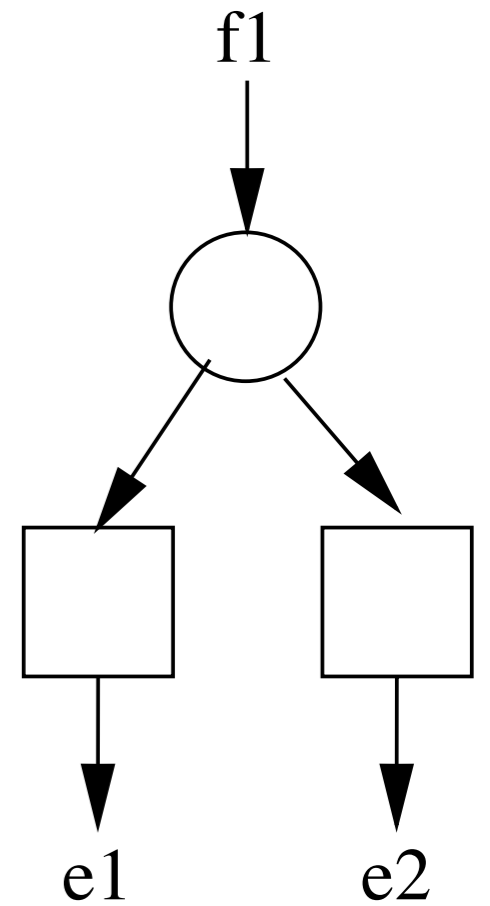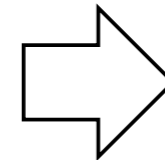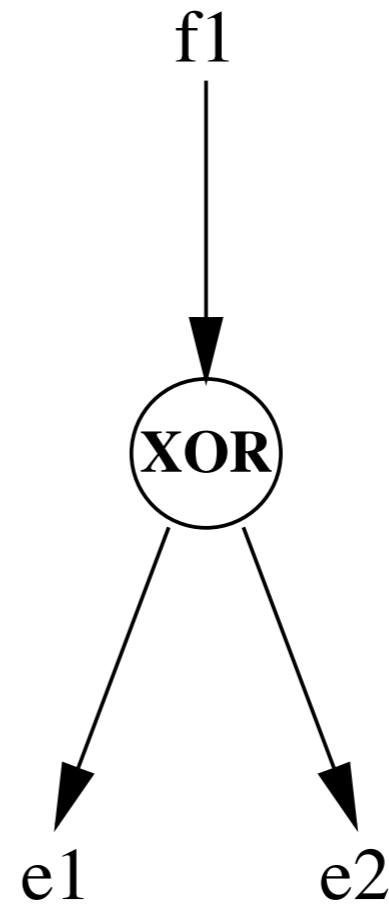
EPC 2 nets: Example

Outcome

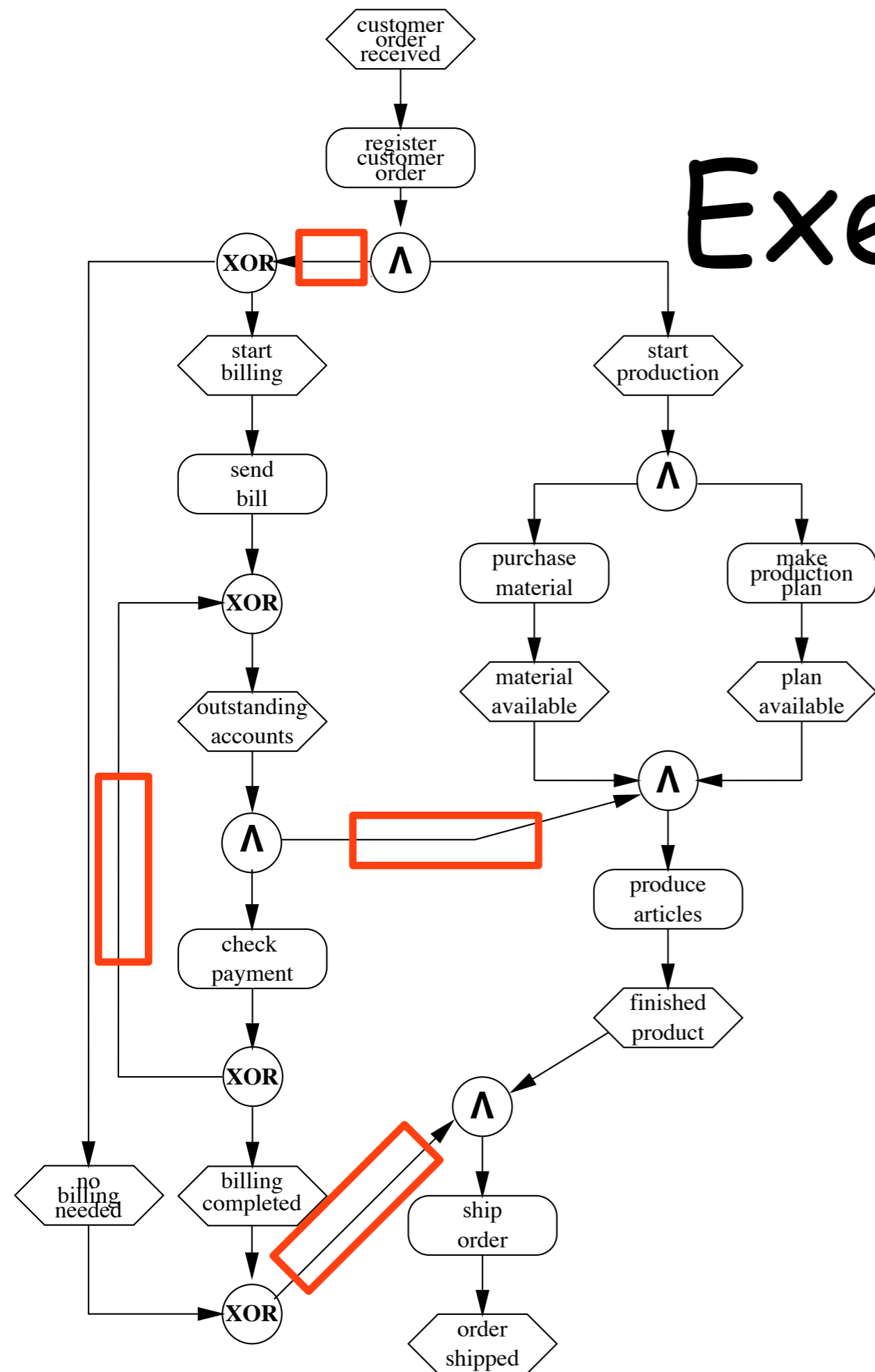**From any EPC we derive a free-choice net**

Moreover, if we add unique start / end events
(and suitable transitions attached to them)
the net is a workflow net

Exercise

Check it sound!

Exercise

customer order received → register customer order → ∧ → XOR

XOR → start billing → send bill → XOR → outstanding accounts → ∧ → check payment → XOR → billing completed → XOR

∧ → start production → ∧ → purchase material → material available → ∧ → produce articles → finished product → ∧ → ship order → order shipped

make production plan → plan available → ∧

no billing needed

Sound?

(remind to add dummy events and functions)

79

Relaxed soundness (a third attempt)

Popularity vs superiority

EPC are a quite successful, semiformal notation

They lack a comprehensive and consistent syntax
They lack even more a corresponding semantics

You may **enrich the notation**, but people will dislike or misinterpret implementation policies

You may **restrict the notation**, but people will prefer the more liberal (flexible) syntax and ignore the guidelines

Remember some good old friends

Chief Process Officer

Business engineer

EPC

Process participants

System architect

WFnet

Process designer

Knowledge worker

Process responsible

System developer

What are ultimately business process?

Graphical language to **communicate** concepts

Careful selection of symbols
shapes, colors, arrows
(the alphabet is necessary for communication)

Greatest common denominator of the people involved

Intuitive meaning
(verbal description, no math involved)

A secret not to tell

The more ways are to interpret a certain construct the more likely an agreement will be reached

A pragmatic consideration

Moreover

in the **analysis phase**
the participants may not be ready
to **finalise** the specification
and decide for the **correct interpretation**

However

ambiguous process description constitutes
a **major problem** in the **design phase**

Consequences

Yet

it is important to find out **flaws** as **soon** as possible

Therefore

we need to fix a **formal representation**
that **preserves all ambiguities**

Problem

EPC is fine (widely adopted)

WF net offer a useful tool

but

Soundness is too demanding at early stages

Relaxed soundness

A **sound** behaviour:
we move from a start event to an end event
so that nothing blocks or remains undone

Execution paths leading to **unsound** behaviour
can be used to infer potential mistakes in the EPC

If some unsound behaviour is possible
but **enough** sound paths exist
the process is called **relaxed sound**

A 3-steps approach (keep it simple!)

Step 1: straightforward element map

Step 2: element fusion

A   B

OR

C

Step 1
Mapping EPC elements to PN–modules

A   B

C

Unification of elements (Case1)

Step 2
Modul combination

Fusion of arcs (Case2)

A   B

tA   tAB   tB

C
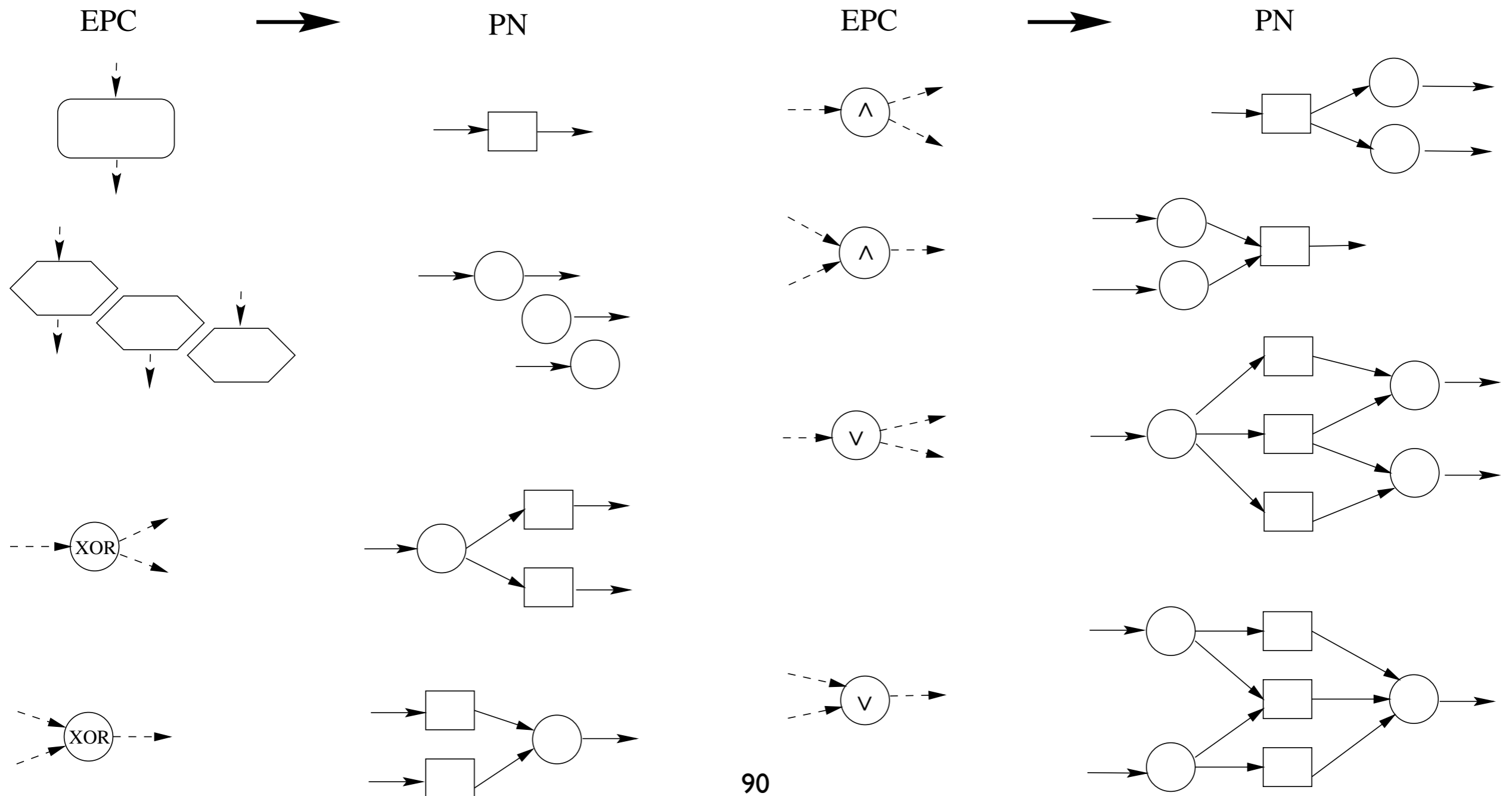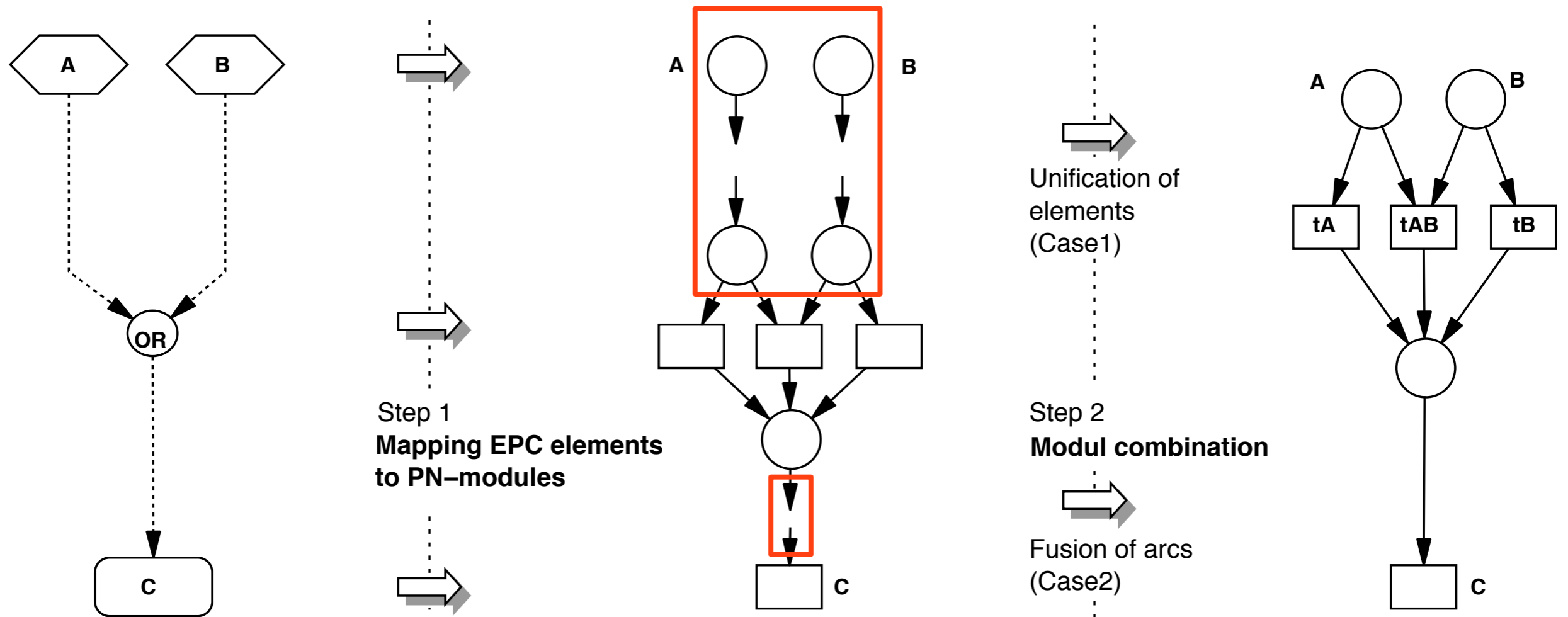
Step 3:
add unique start / end

a)

s   b)

XOR start

AND

AND
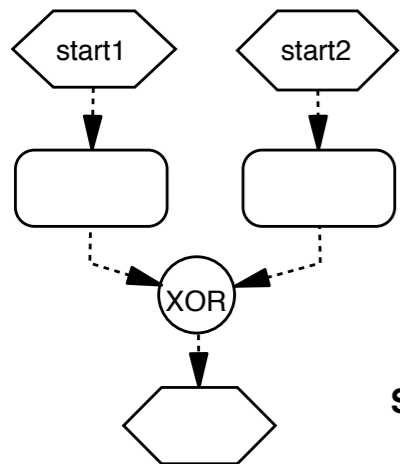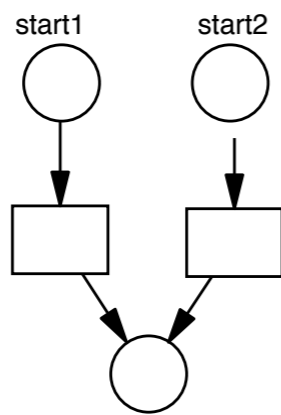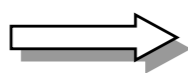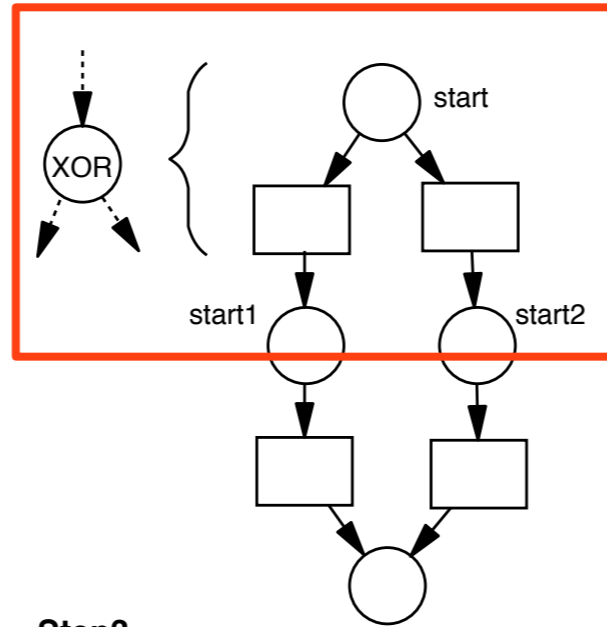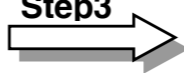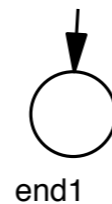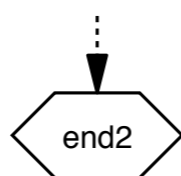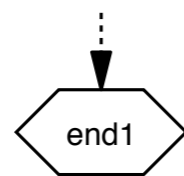
Step1 &
Step2
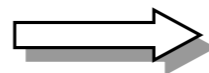
Step3

(sometimes AND can be preferred)

c

end1   end2

Step1 &
Step2   S

Step3   Step3

OR end

goods arrived E11

∧ C6

check goods F8

XOR C8

ok E15

not_ok E14

∨ C7

complaint F10

record receipt of goods F9

data revised E16

goods recorded E12

XOR C11

∧ C12

store goods F13

stored E20

Example

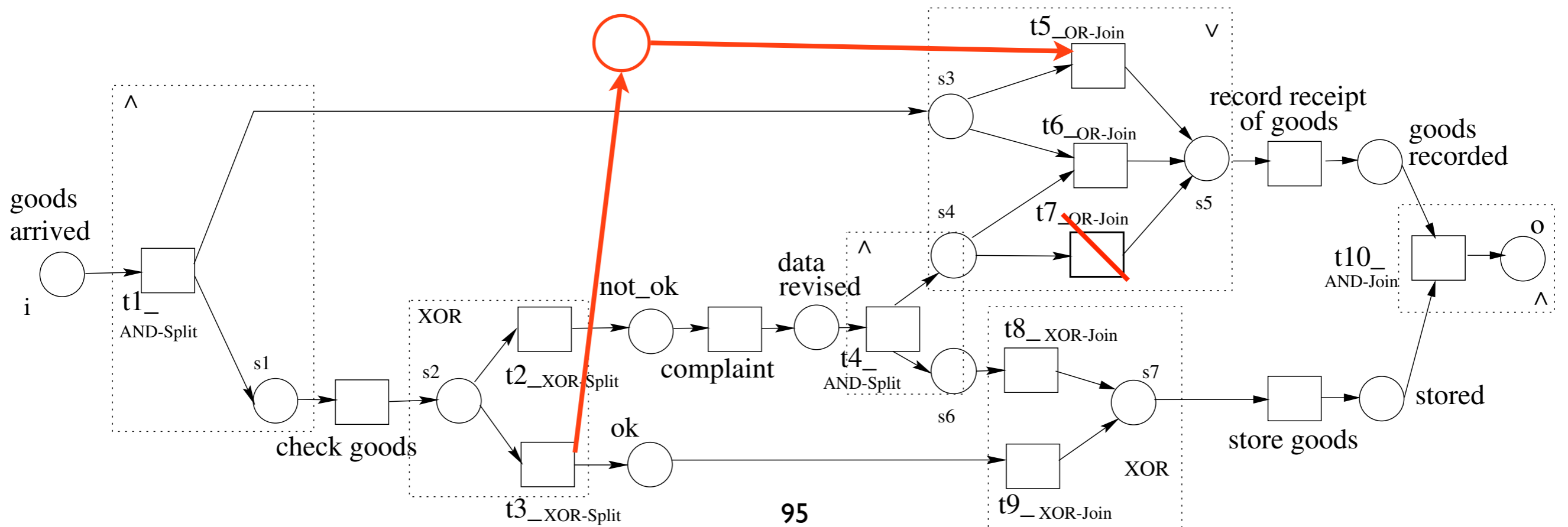# Sound?

93

Example

Not sound!

94
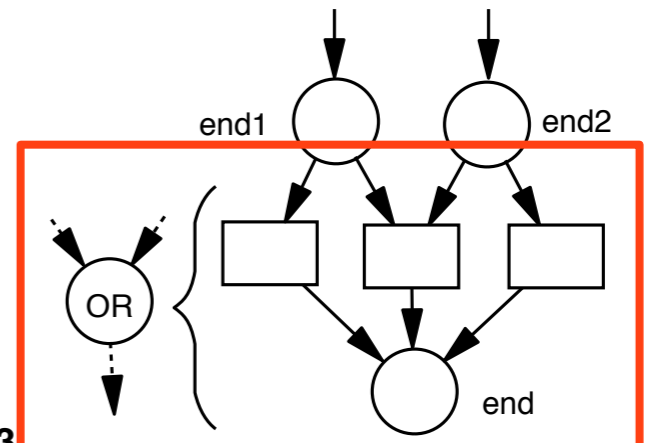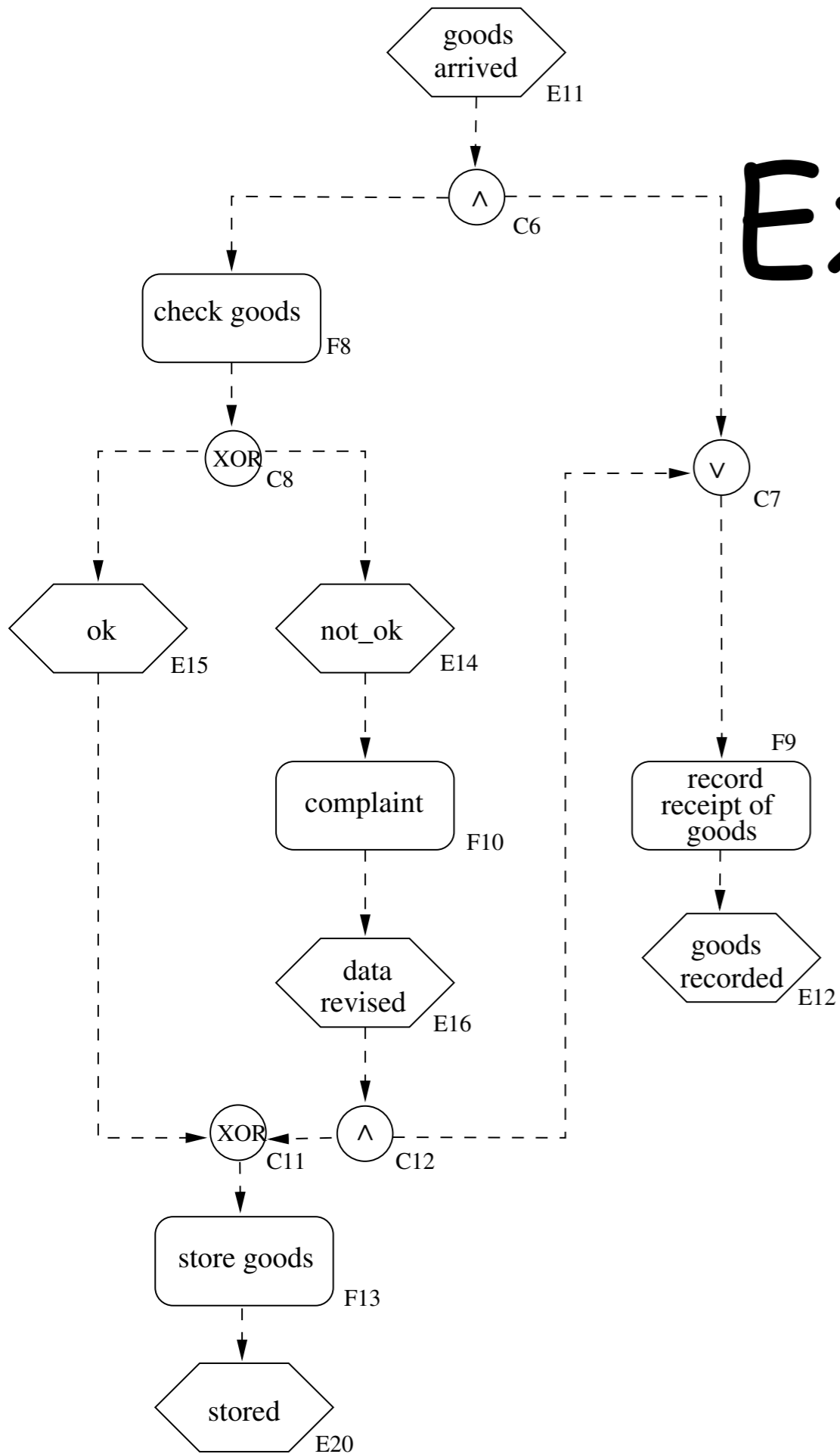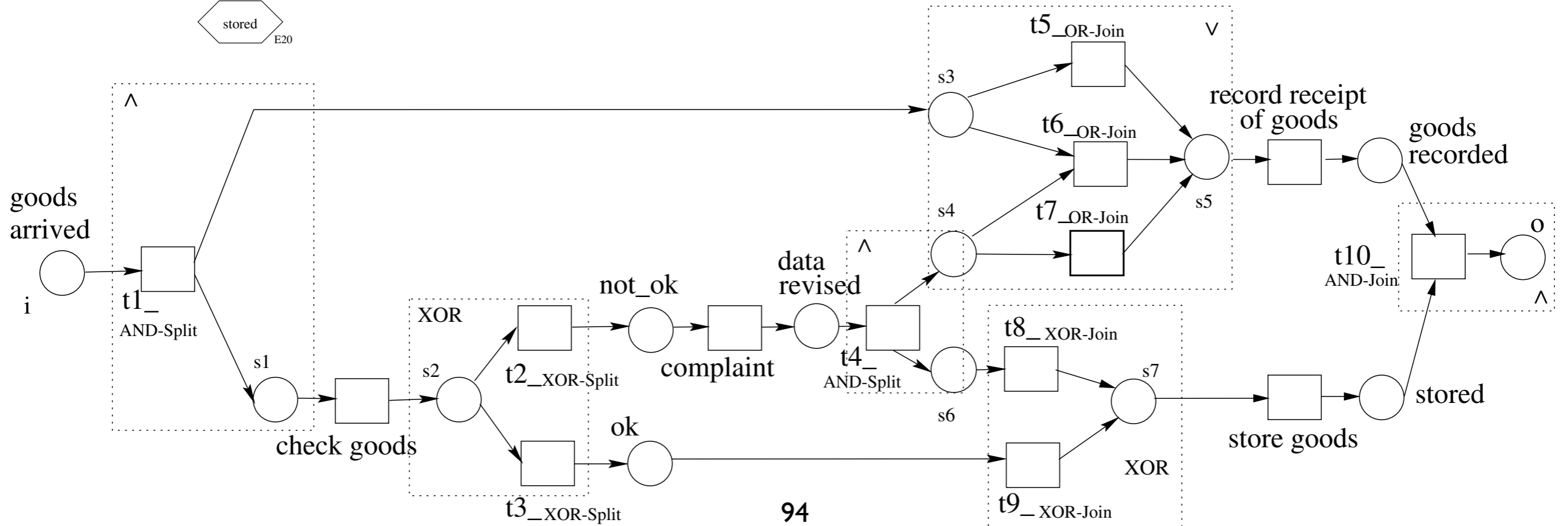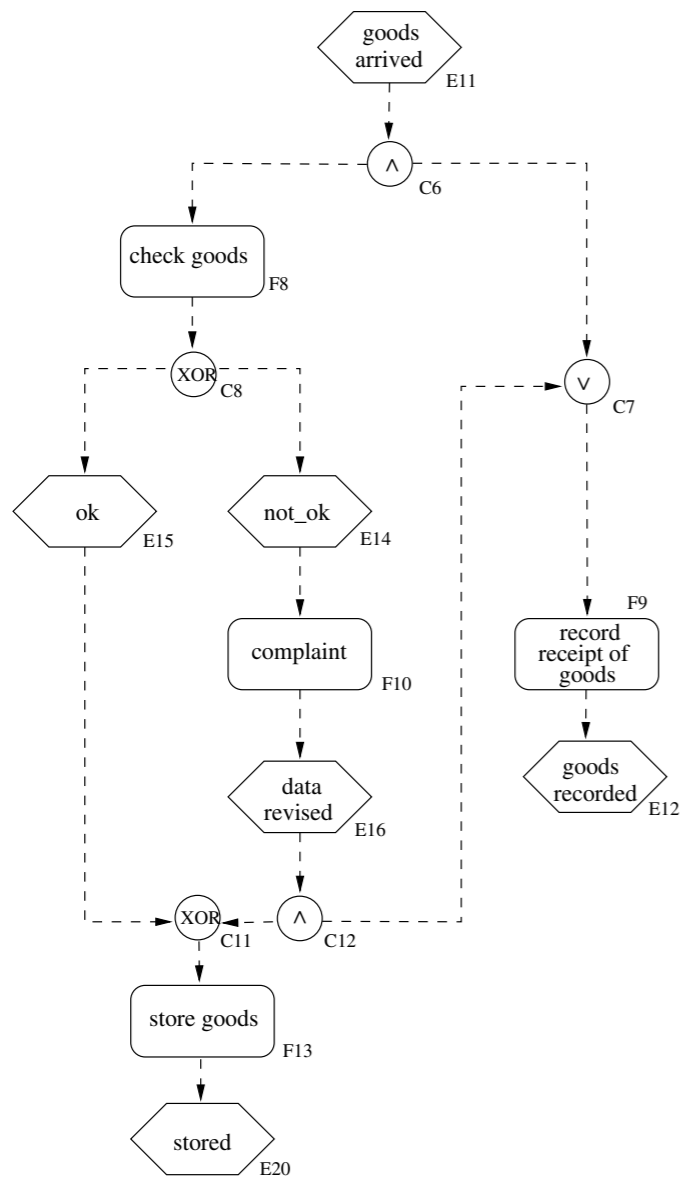
Example

We can turn it to sound, but:
small changes in the net, turn big in EPC
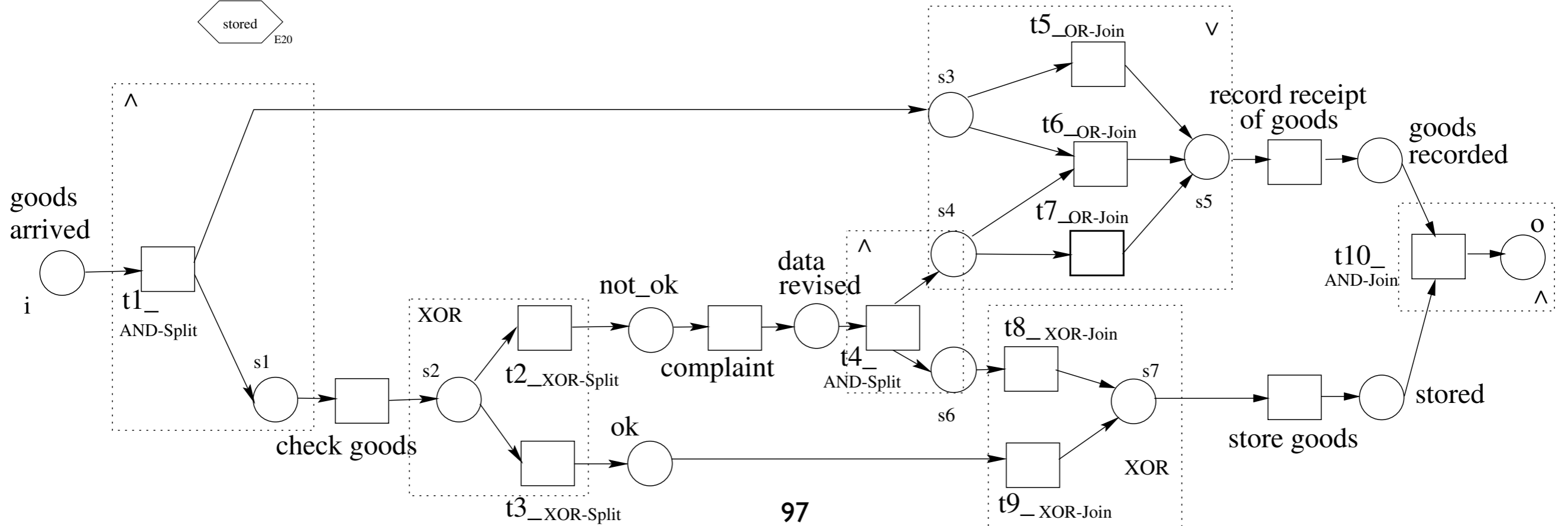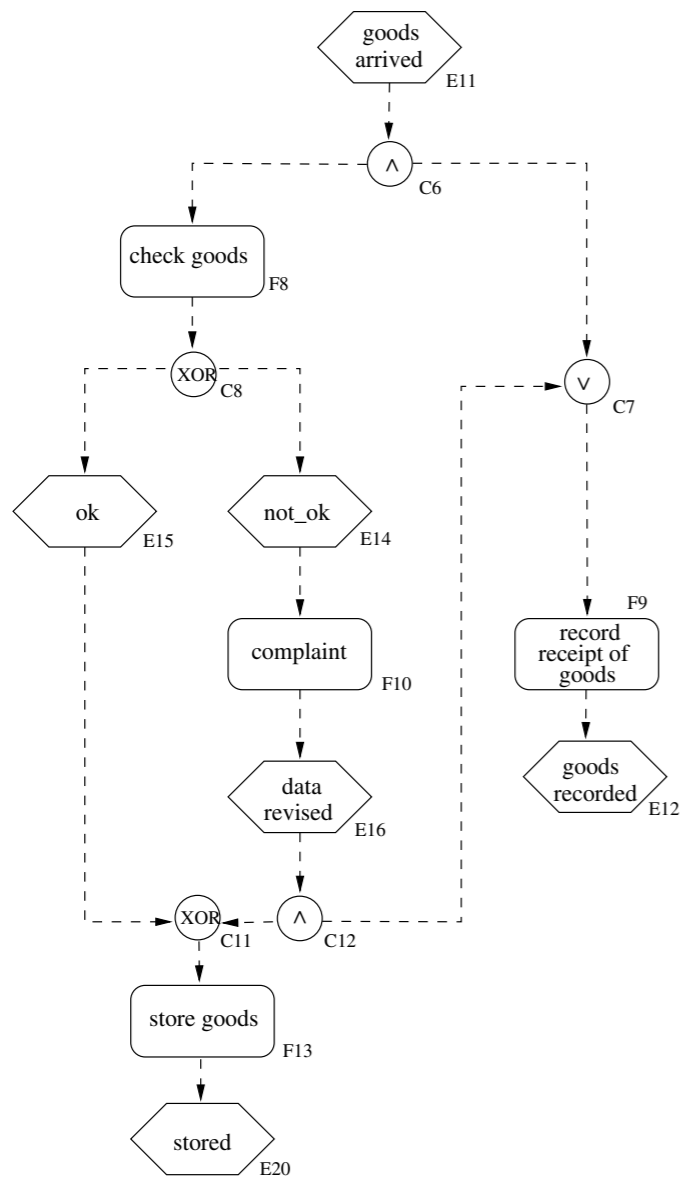
Relaxed soundness: formally

Definition: A WF net is **relaxed sound** if every transition belongs to a firing sequence that starts in state i and ends in state o

$$\forall t \in T. \, \exists M, M'. \, i \rightarrow^* M \xrightarrow{t} M' \rightarrow^* o$$

(it is sound "enough", in the sense that all transitions are covered by at least one sound execution)
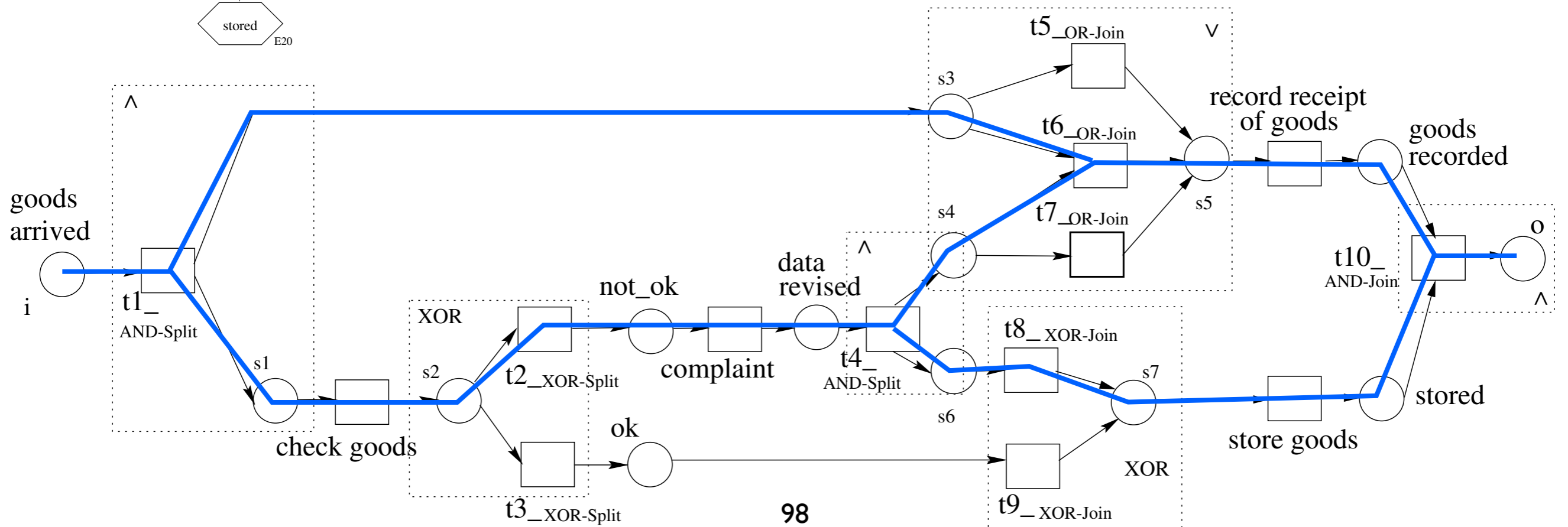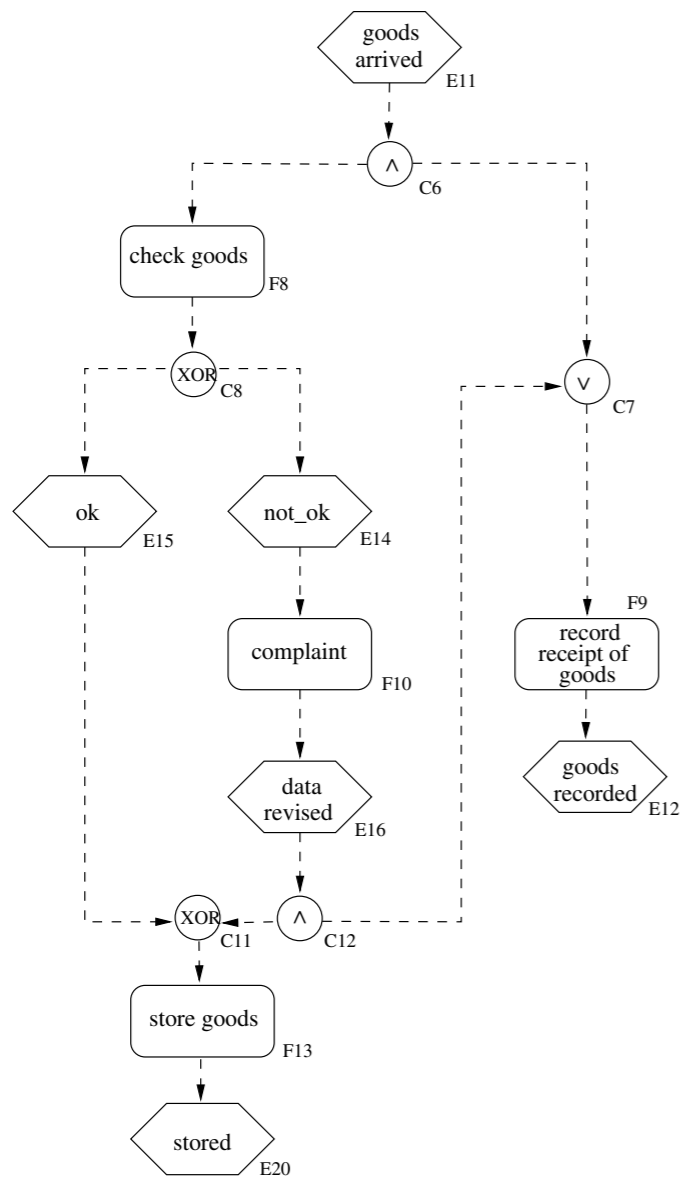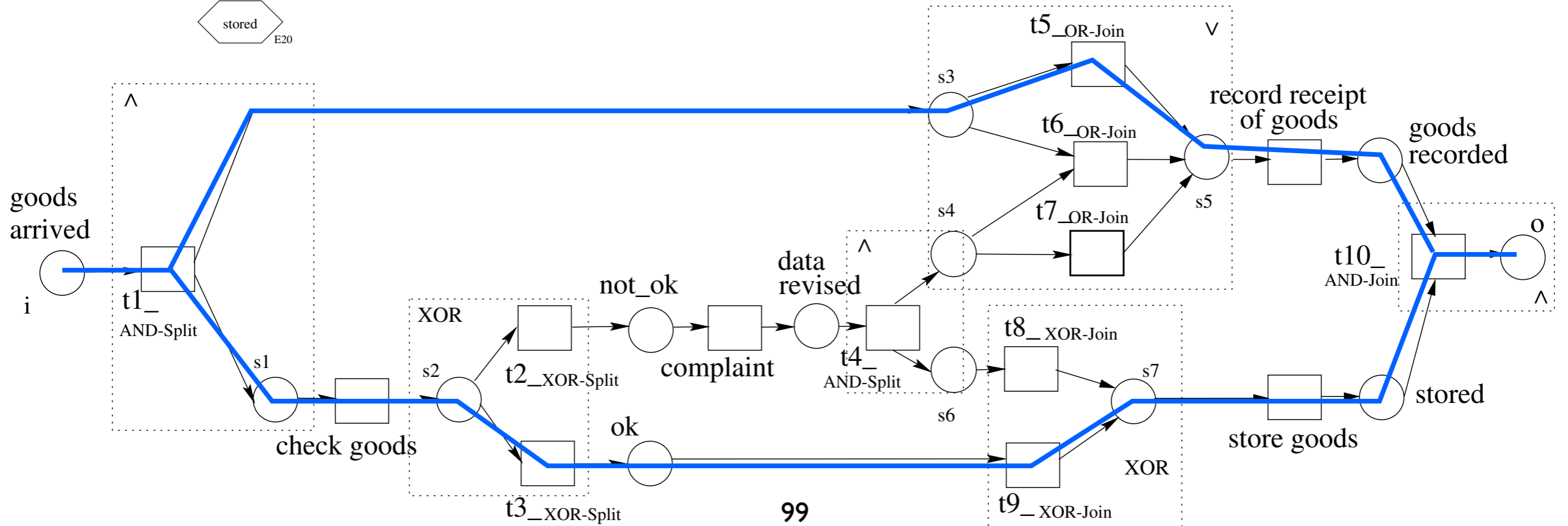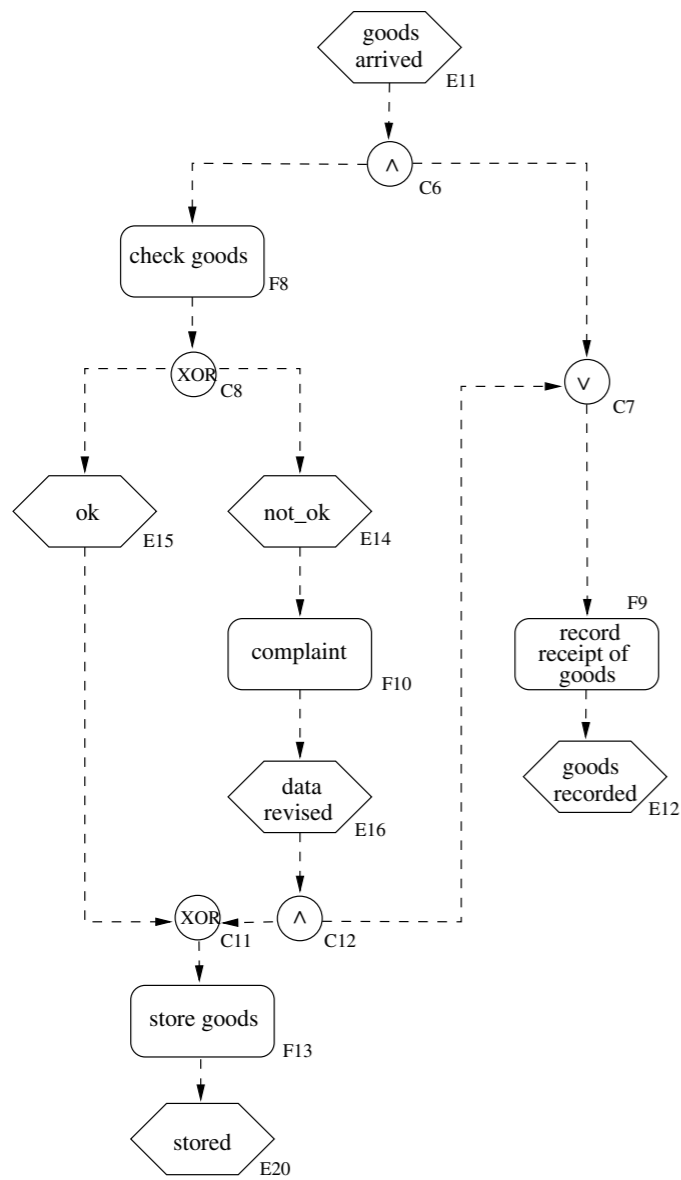
Example

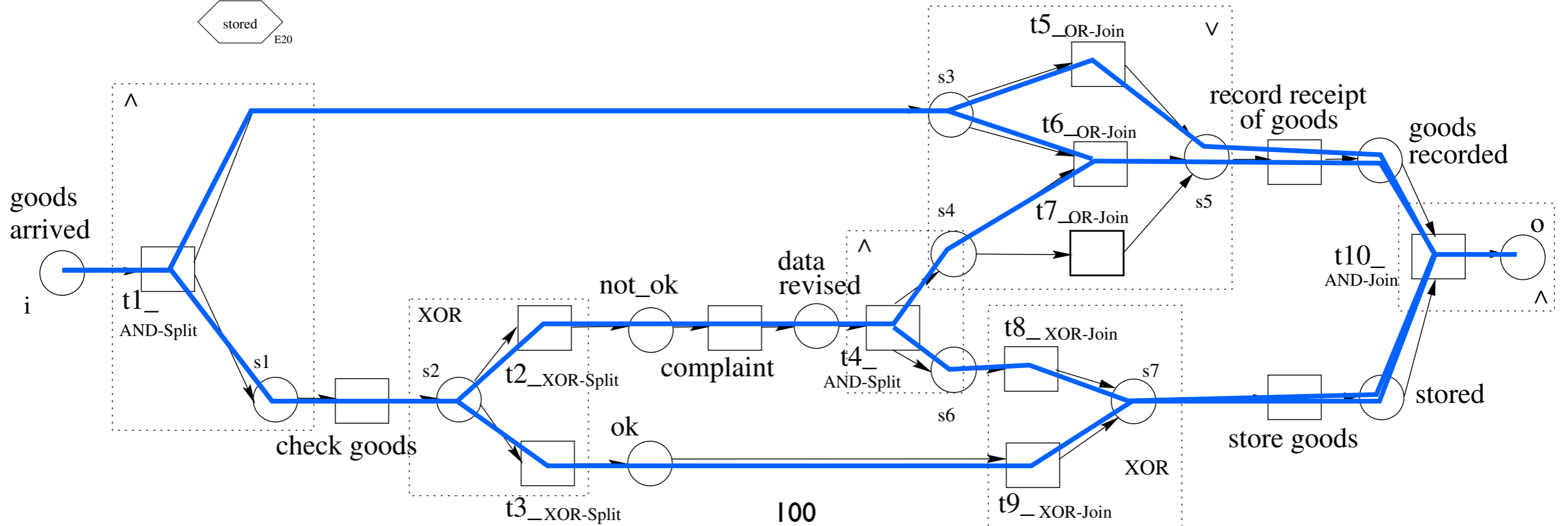## Relaxed sound?

Example

Relaxed sound?
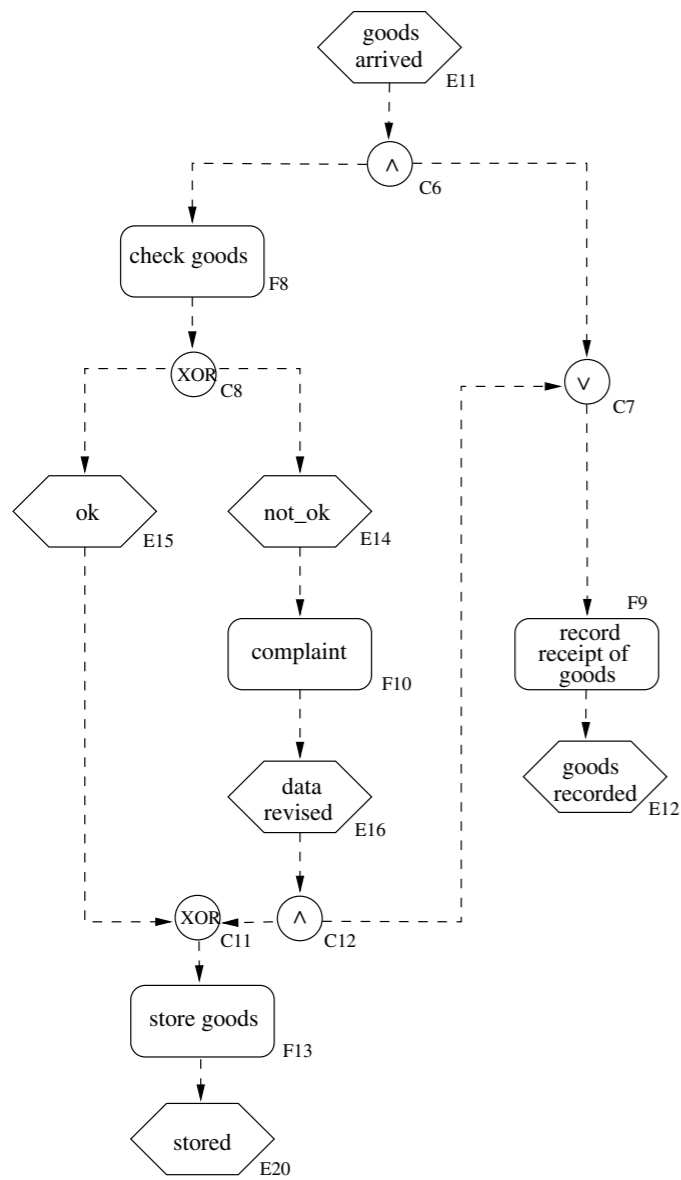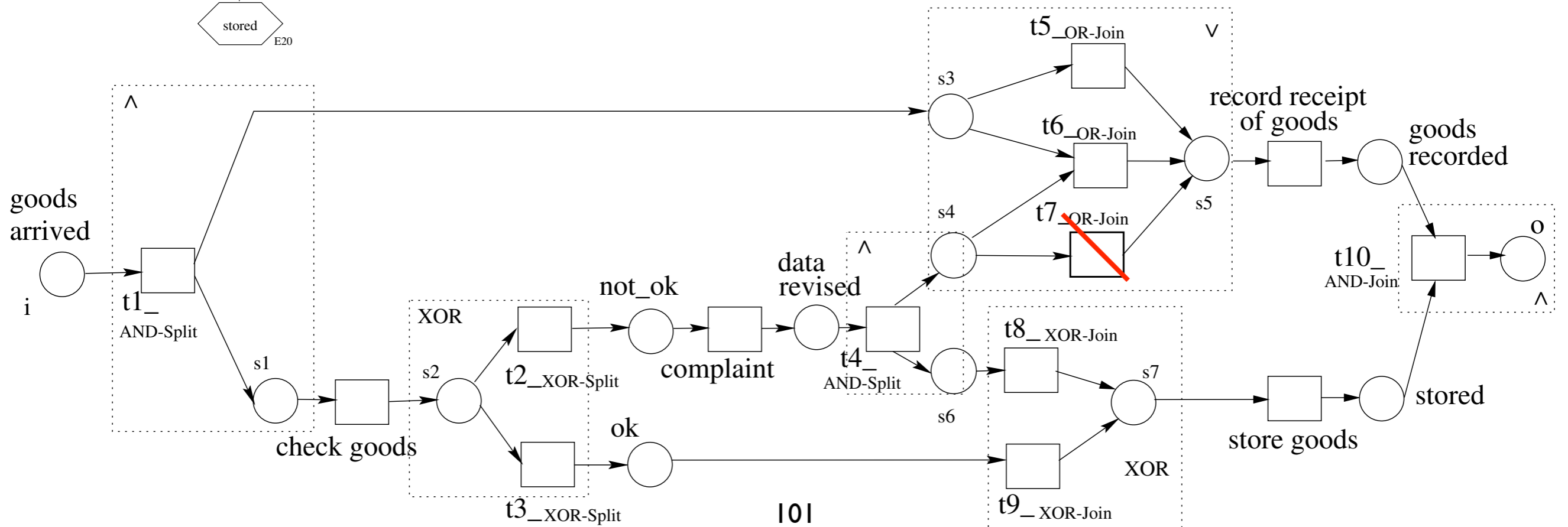

98

Example

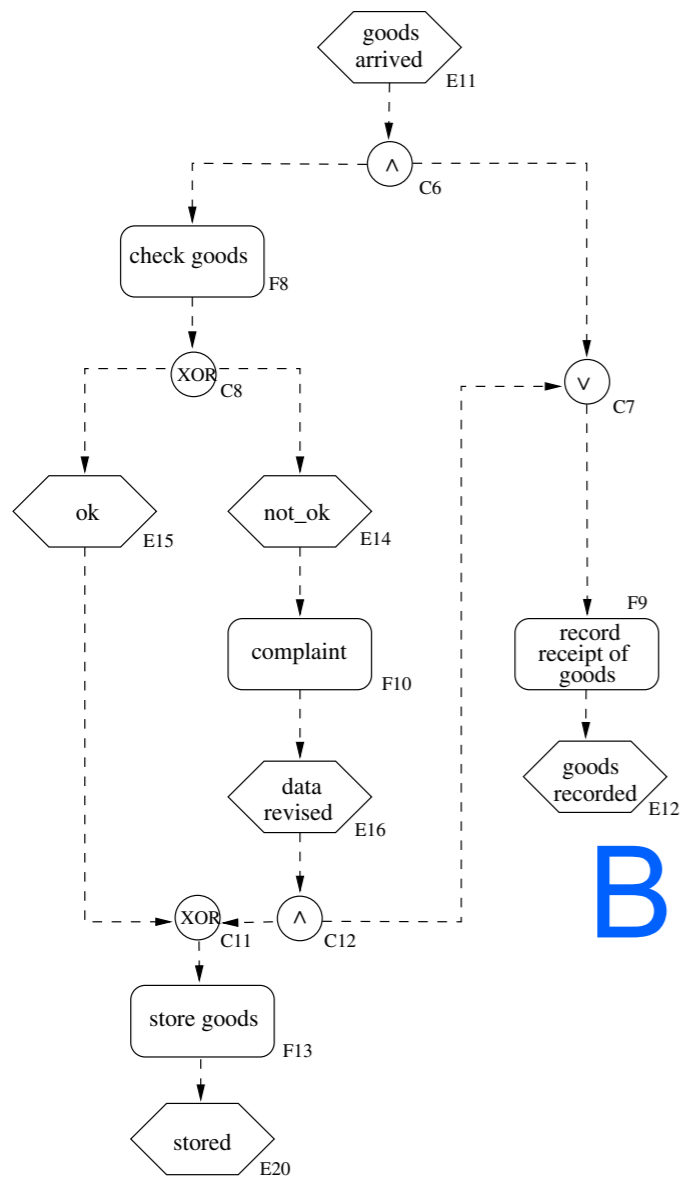## Relaxed sound?

Example

Relaxed sound?

Example

Not relaxed sound!
But relaxed sound as EPC

Pros and Cons

If the WF net is **not sound**:
there are transitions that are not contained in any sound
firing sequence

Hence their EPC counterparts need improvements

Relaxed soundness can be proven only by enumeration
(of enough sound firing sequences)

No equivalent characterisation is known
that is more convenient to check