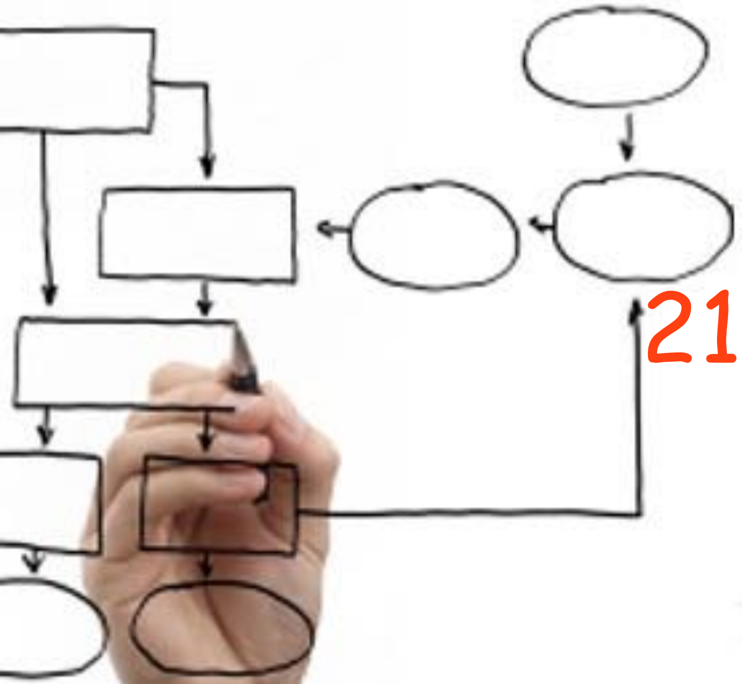
Business Processes Modelling

MPB (6 cfu, 295AA)

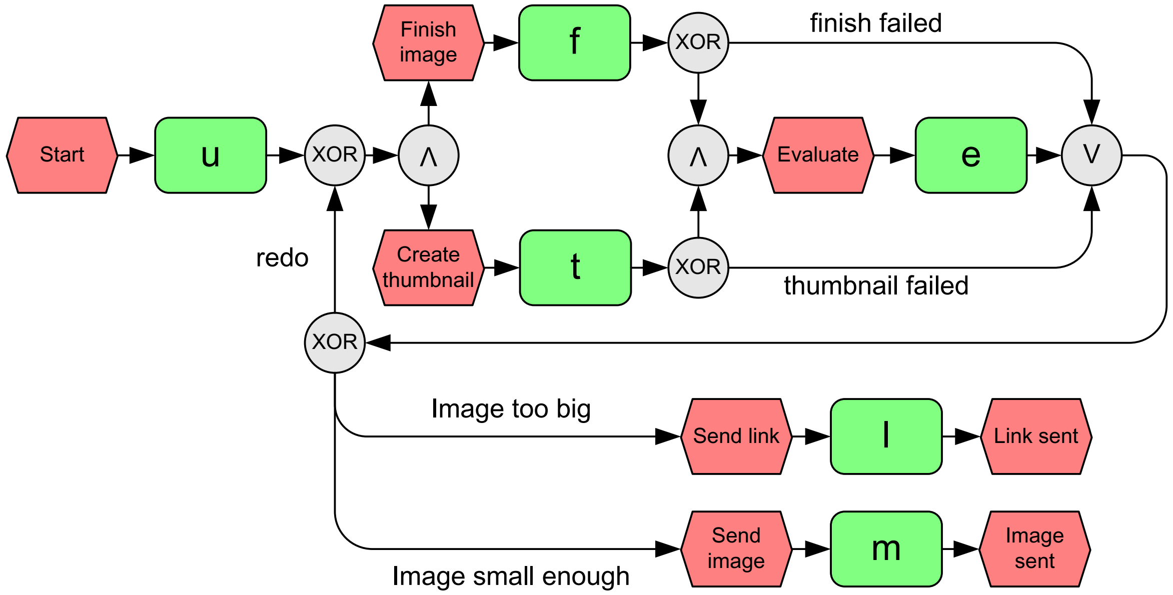
Roberto Bruni

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



21 - 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

EPC origin (early 1990's)

EPC method originally developed as part of a holistic modelling approach called

ARIS framework

(Architecture of Integrated Information Systems)

by Wilhelm-August Scheer

A.-W. Scheer

ARIS –

Vom Geschäftsprozess zum Anwendungssystem

Vierte, durchgesehene Auflage

Springer

Event-driven Process Chain

An **Event-driven Process Chain** (EPC)
is a flow-chart that can be used:
to configure an Enterprise Resource Planning implementation
to drive the modelling, analysis, redesign of business process

Informal notation: simple, intuitive and easy-to-understand

EPC represents domain concepts and processes
(neither their formal aspects nor their technical realization)

EPC Markup Language (EPML): XML interchange format

EPC Diagrams

Why do we need diagrams?

Graphical languages **communicate** concepts

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

Greatest common denominator of the people involved

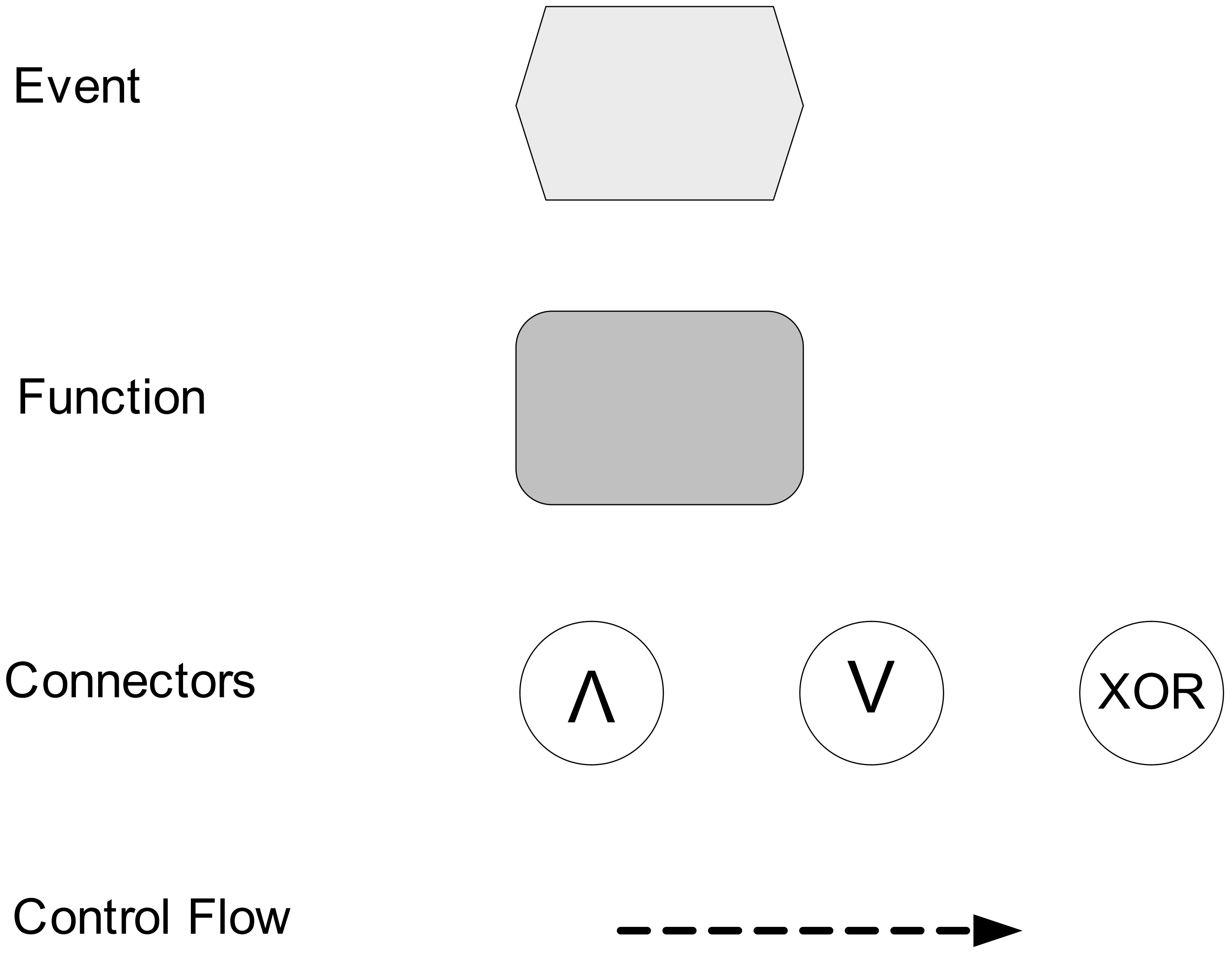
Intuitive meaning
(verbal description, no math involved)

EPC informally

An EPC is a graph of **events** and **functions**

It provides some logical **connectors** that allow alternative and parallel execution of processes (AND, XOR, OR)

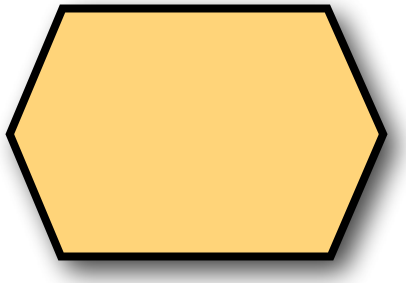
EPC ingredients at a glance



Events

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

Graphical representation: hexagons

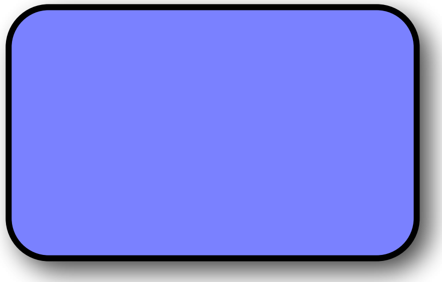


Passive elements used to describe
under which circumstances a process (or a function) works
or which state a process (or a function) results in
(like pre- / post-conditions)

Functions

Any EPC diagram may involve several **functions**

Graphical representation: rounded rectangles



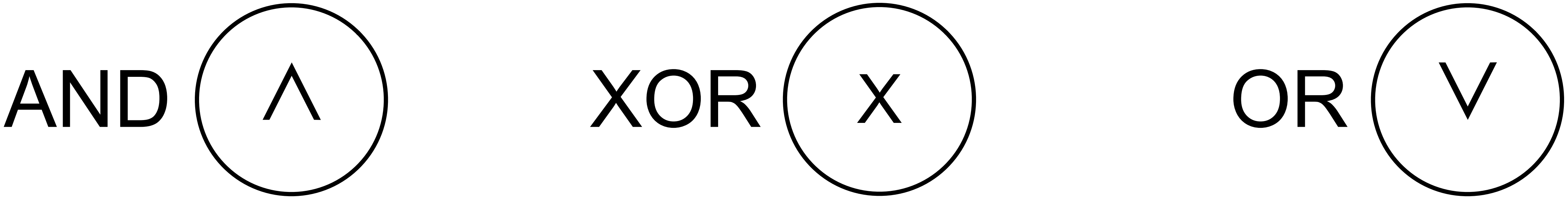
Active elements used to describe
the tasks or activities of a business process

Functions can be refined to other EPC diagrams

Logical connectors

Any EPC diagram may involve several **connectors**

Graphical representation: circles (or also octagons)



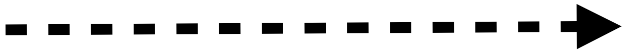
AND ∧ XOR X OR ∨

Elements used to describe
the logical relationships between split/join branches

Control flow

Any EPC diagram may involve several **connections**

Graphical representation: dashed arrows



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

EPC diagrams

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

The graph is **weakly connected** (e.g., no isolated nodes)

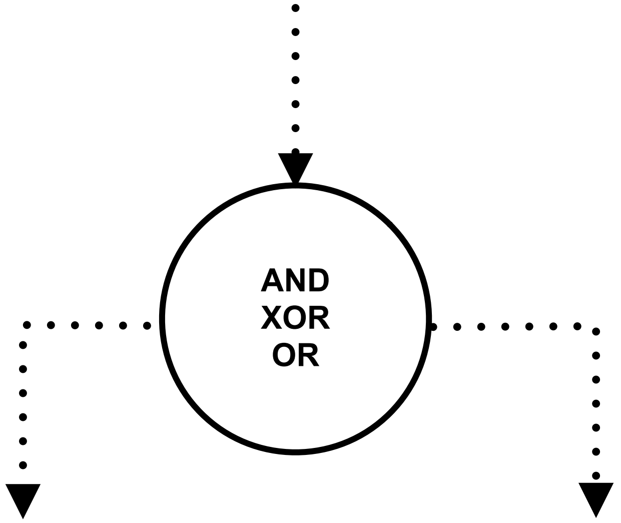
Events have at most one incoming and one outgoing arc
Events have at least one incident arc
There must be at least one start event and one end event

Functions have exactly one incoming and one outgoing arc

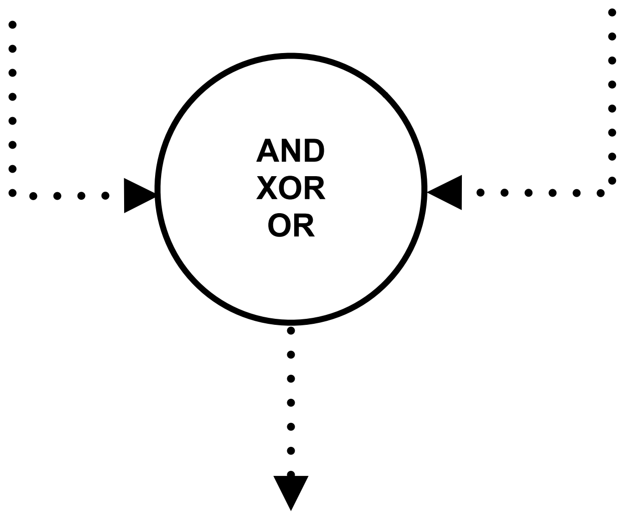
Connectors have either one incoming arc and multiple outgoing arcs
or viceversa (multiple incoming arcs and one outgoing arc)

Logical connectors: splits and joins

Splits



Joins

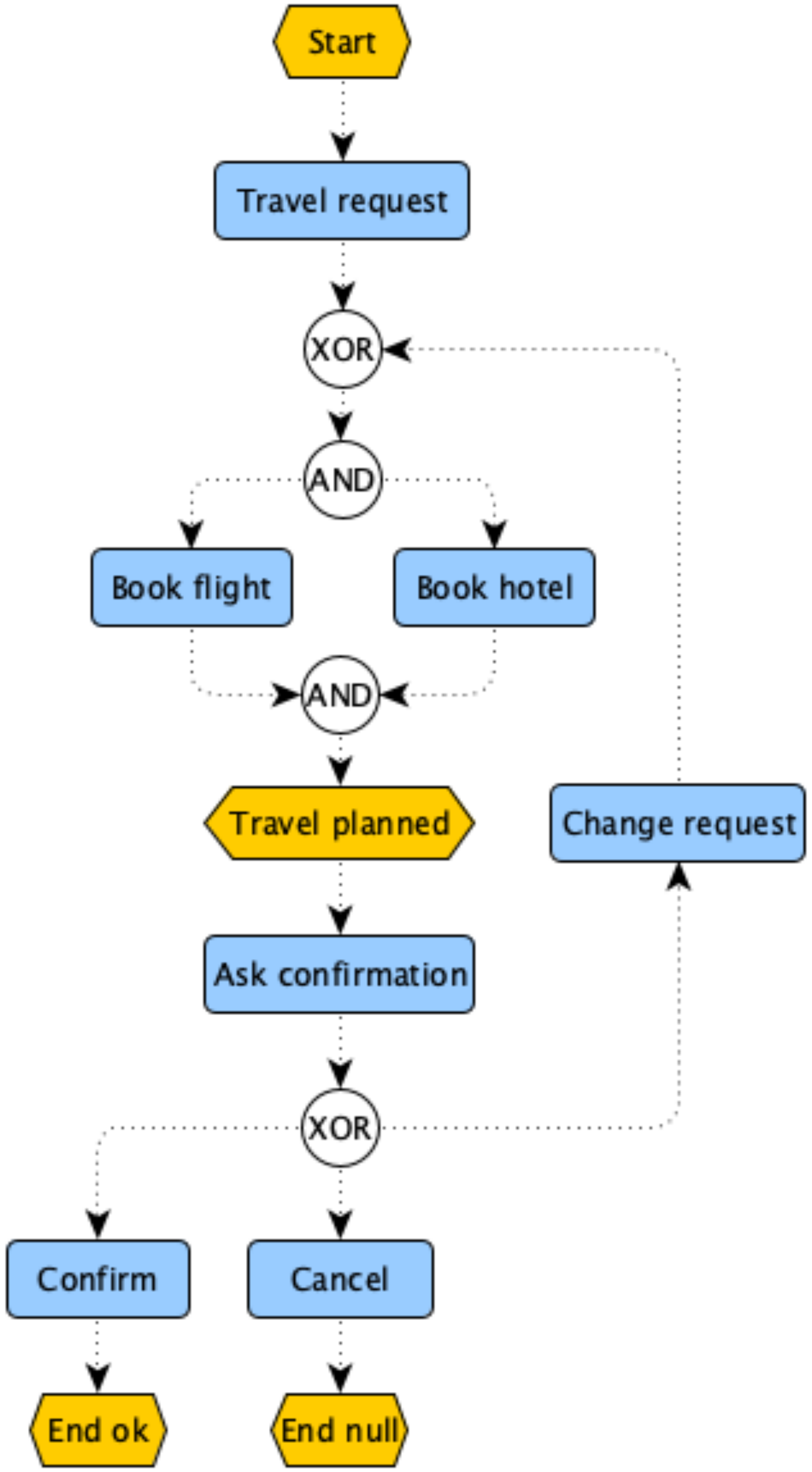


EPC: Example

∧ = AND

X = XOR

∨ = OR



EPC Diagrams: guidelines

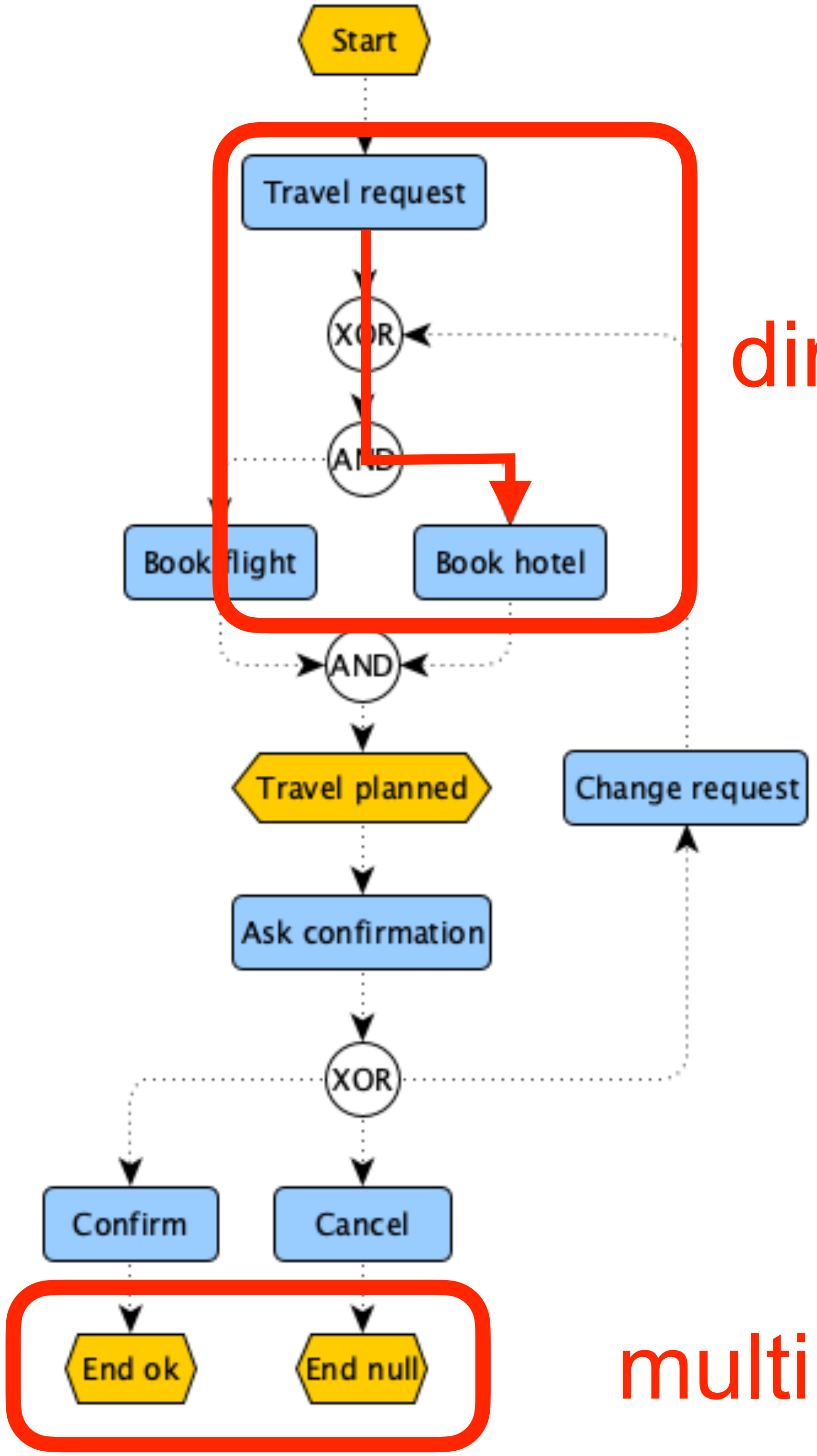
Other constraints are sometimes imposed

Unique start / end event

No direct flow between two events
No direct flow between two functions

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

EPC guidelines: Example



direct flow between functions

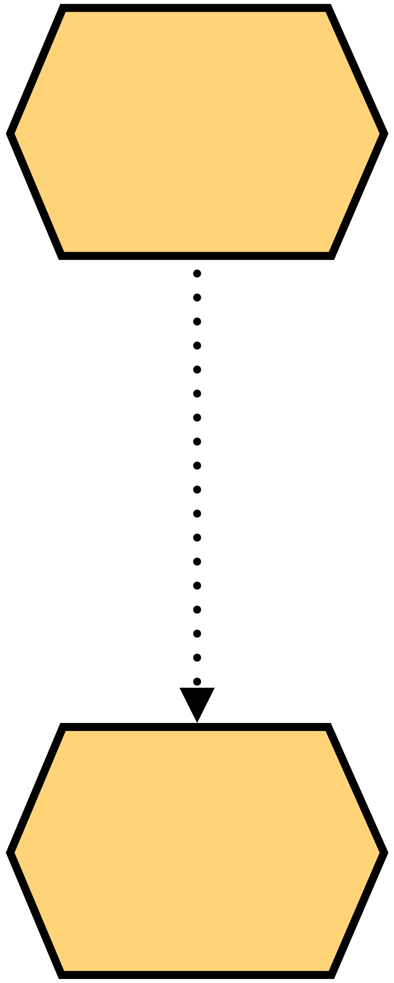
multiple end events

Problem with guidelines

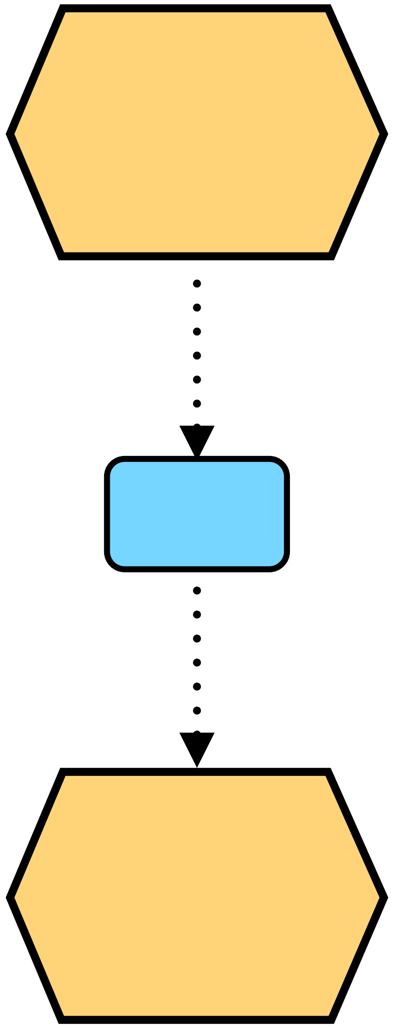
From empirical studies:
guidelines are too restrictive and people ignore them
(otherwise diagrams would get unnecessarily complicated,
more difficult to read and understand)

Solution:
It is safe to drop most constraints
(implicit dummy nodes might always be added later, if needed)

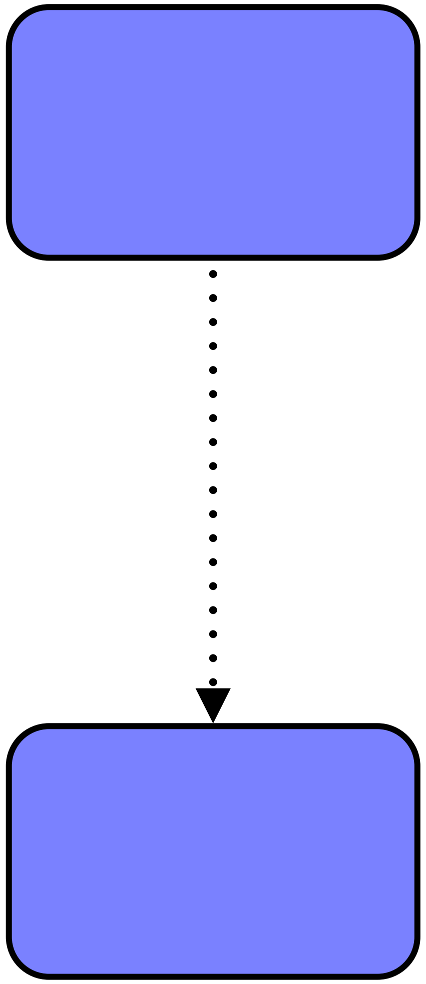
EPC: repairing alternation



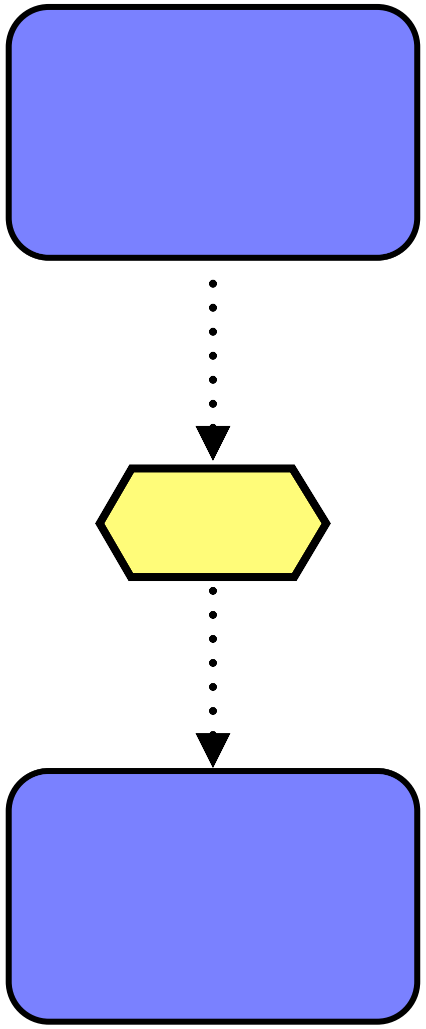
add dummy functions to guarantee alternation



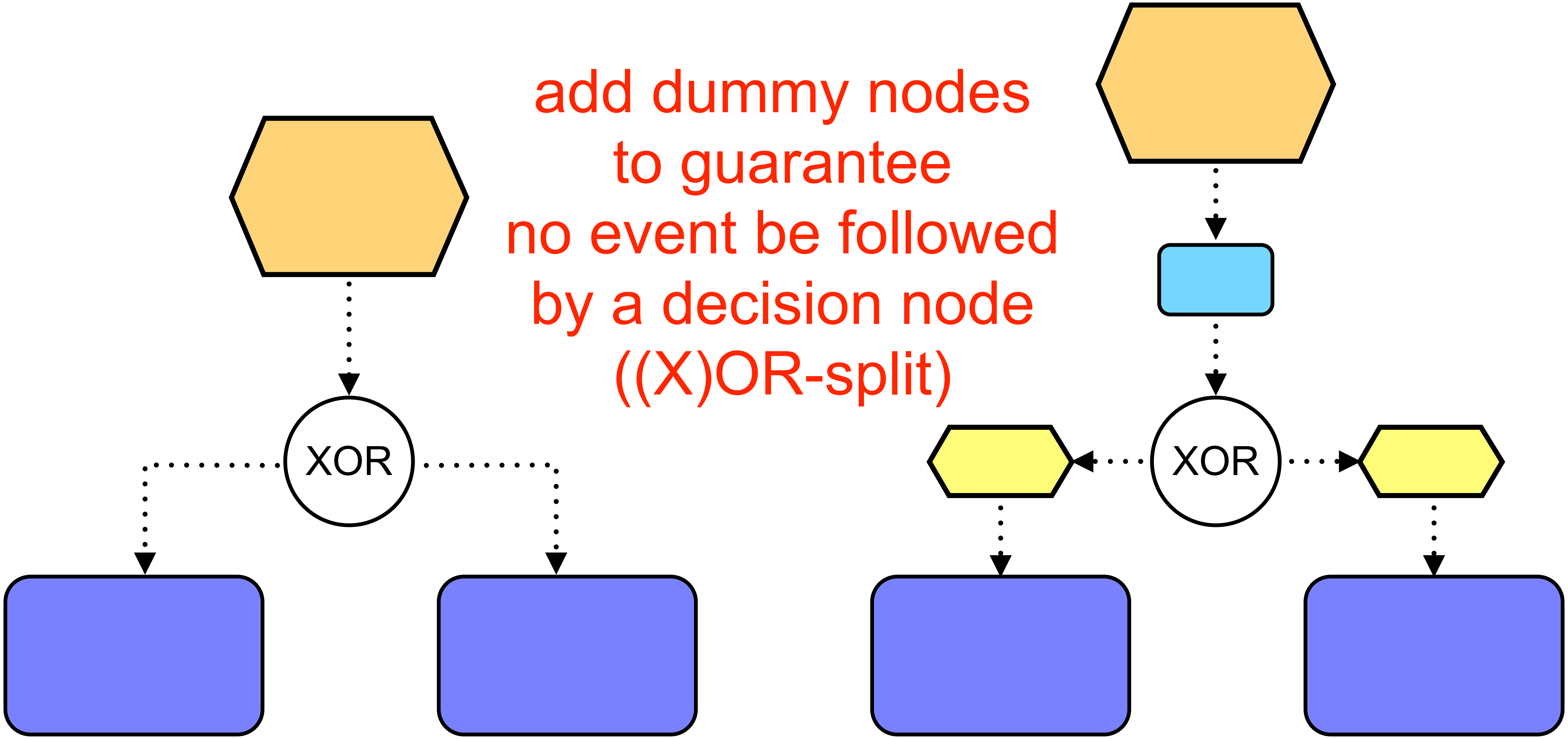
EPC: repairing alternation



add dummy
events
to guarantee
alternation



EPC: repairing decisions

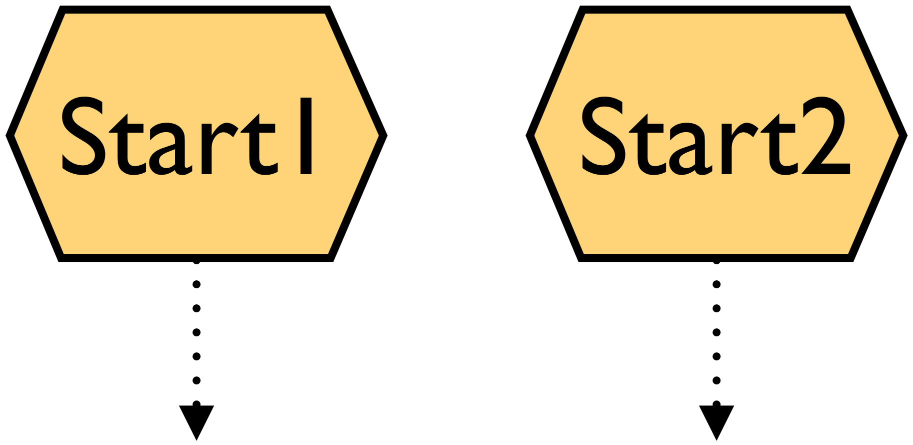


add dummy nodes
to guarantee
no event be followed
by a decision node
((X)OR-split)

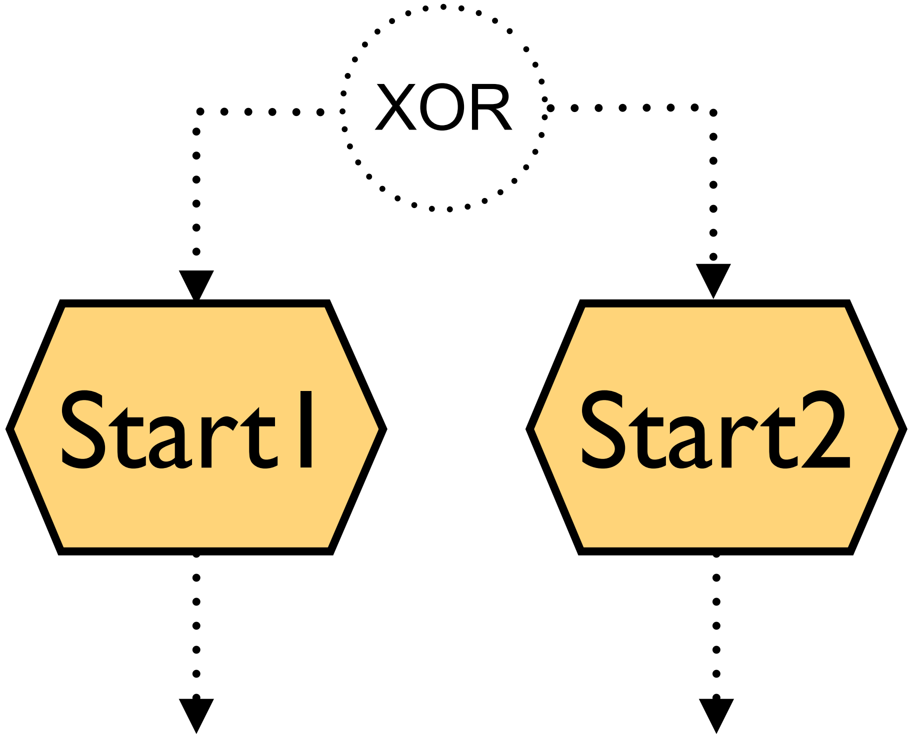
EPC: repairing multiple start events

A start event is an event with no incoming arc
it invokes a new instance of the process template

Start events are mutually exclusive

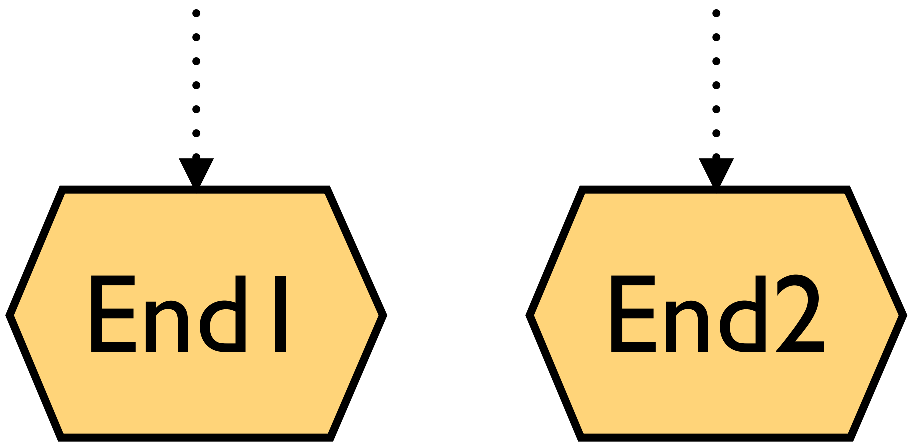


assume an implicit XOR split is present

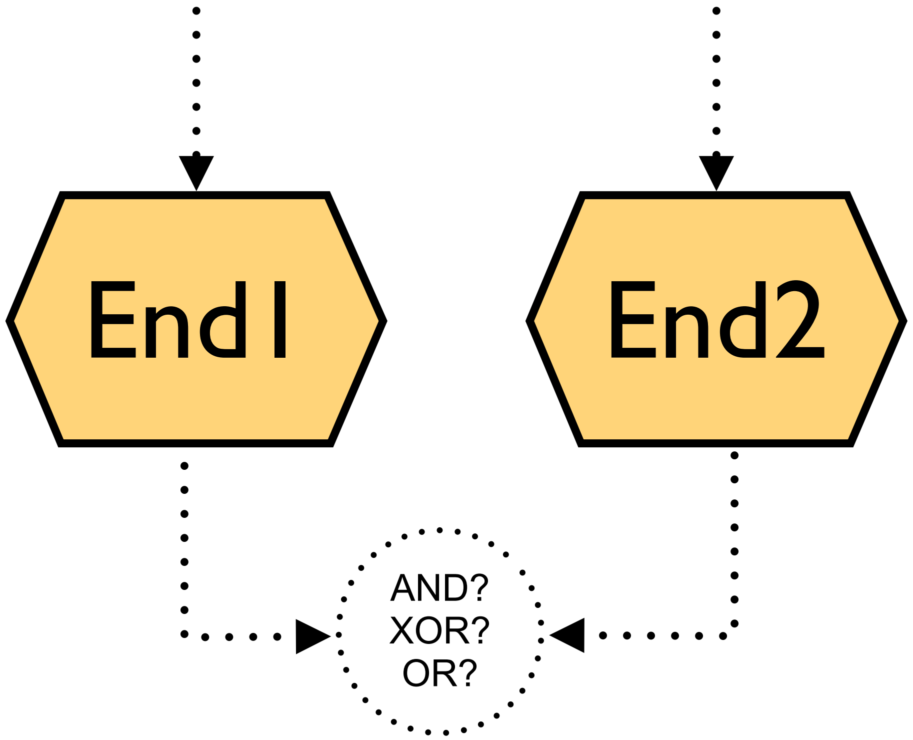


EPC: repairing multiple end events

An end event is an event with no outgoing arc
it indicates completion of some activities
What if multiple end events occur? No unanimity!
they are followed by an implicit join connector
(typically a XOR… but not necessarily so)

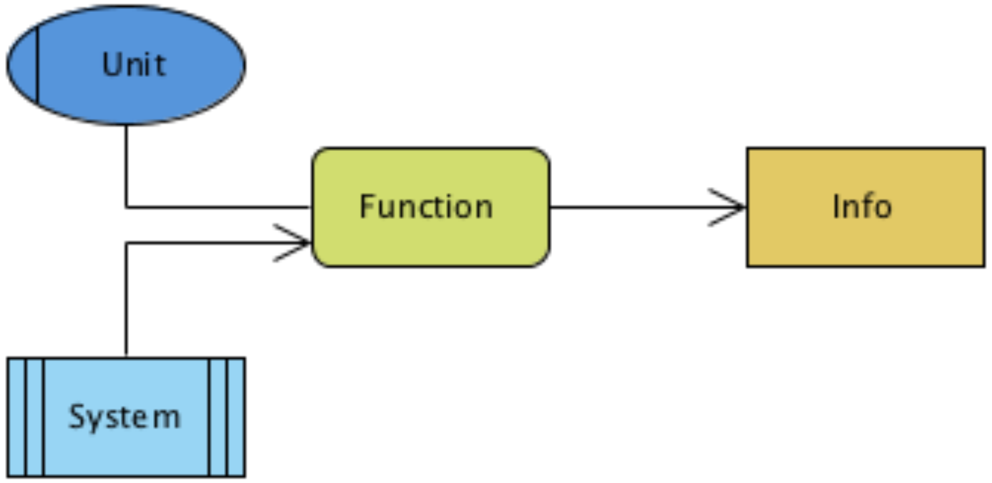


assume an
implicit
join
is present



Other ingredients: function annotations

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



Information, material, resource object:
represents objects in the real world
e.g. input data or output data for a function
(rectangles linked to function boxes)
angles with vertical lines on its sides)

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

EPC Semantics

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”

EPC formal semantics?

Little unanimity around the EPC semantics

Rough verbal description
in the original publication by Scheer (1992)

Later, several attempts to define formal semantics
(assigning different meanings to the same EPC,
sometimes leading to paradoxes)

Discrepancies typically stem from the interpretation
of (X)OR join connectors

Sound EPC diagrams

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

We transform EPC diagrams to Workflow nets:
the EPC diagram is sound if its net is so

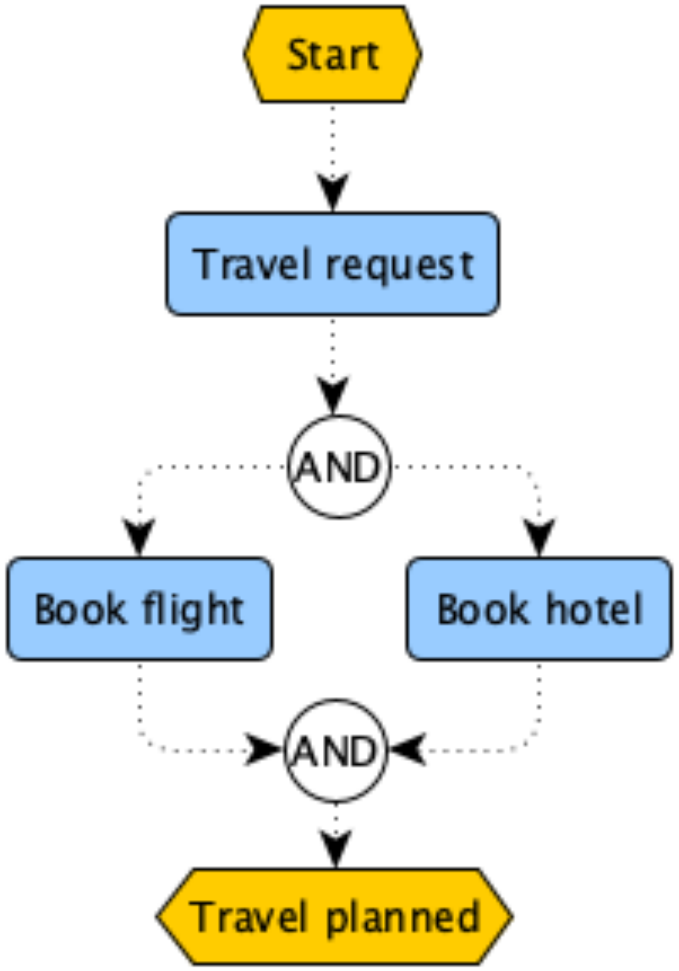
We can reuse the verification tools
to check if the net is sound

Is there a unique way to proceed? Not necessarily!

Translation of EPC to Petri nets

The idea

From EPC to wf nets in three steps



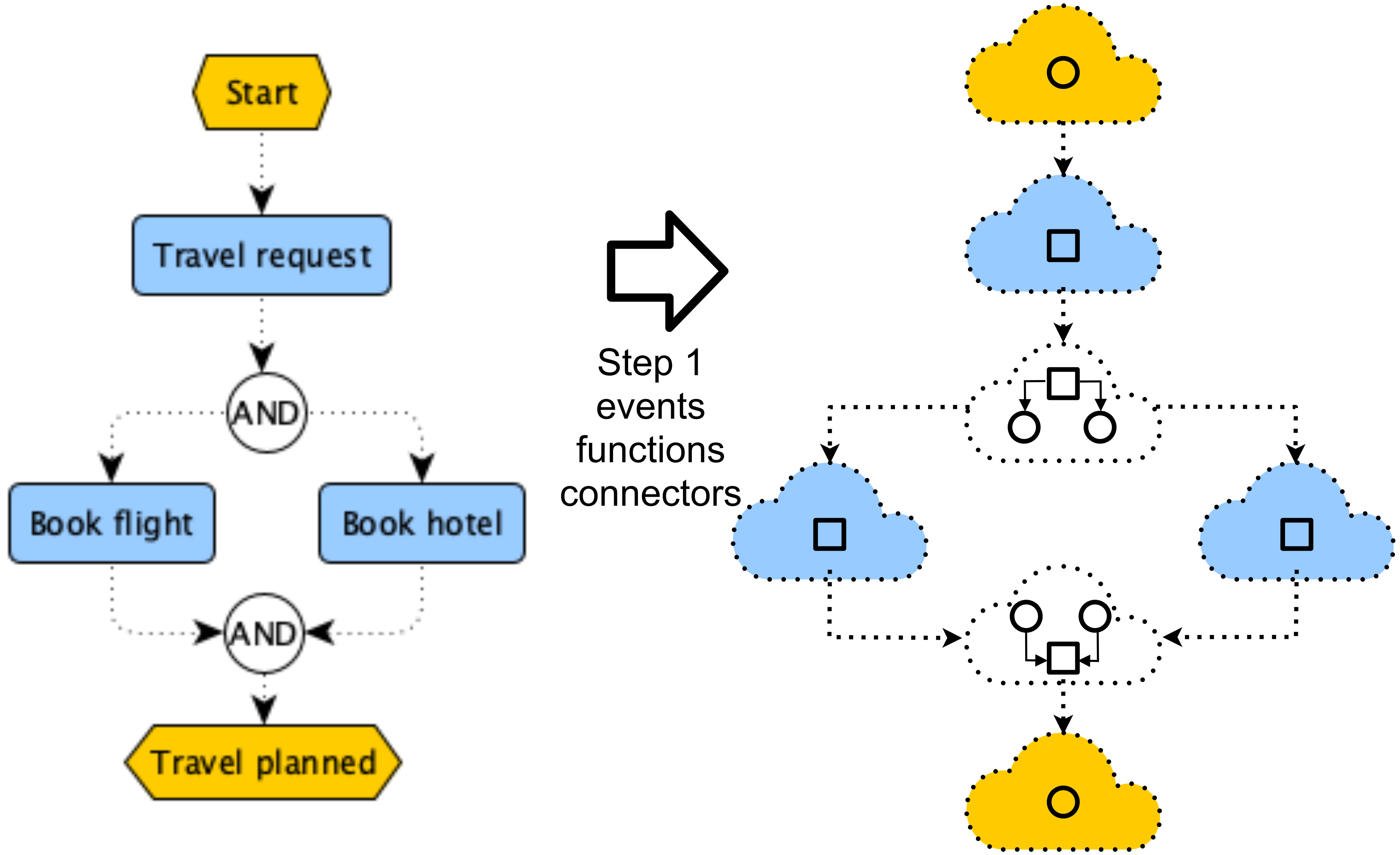
Step 1
convert each
- event
- function
- connector
to a net fragment

Step 2
connect
fragments
together

Step 3
enforce
initial place
final place

Step 1

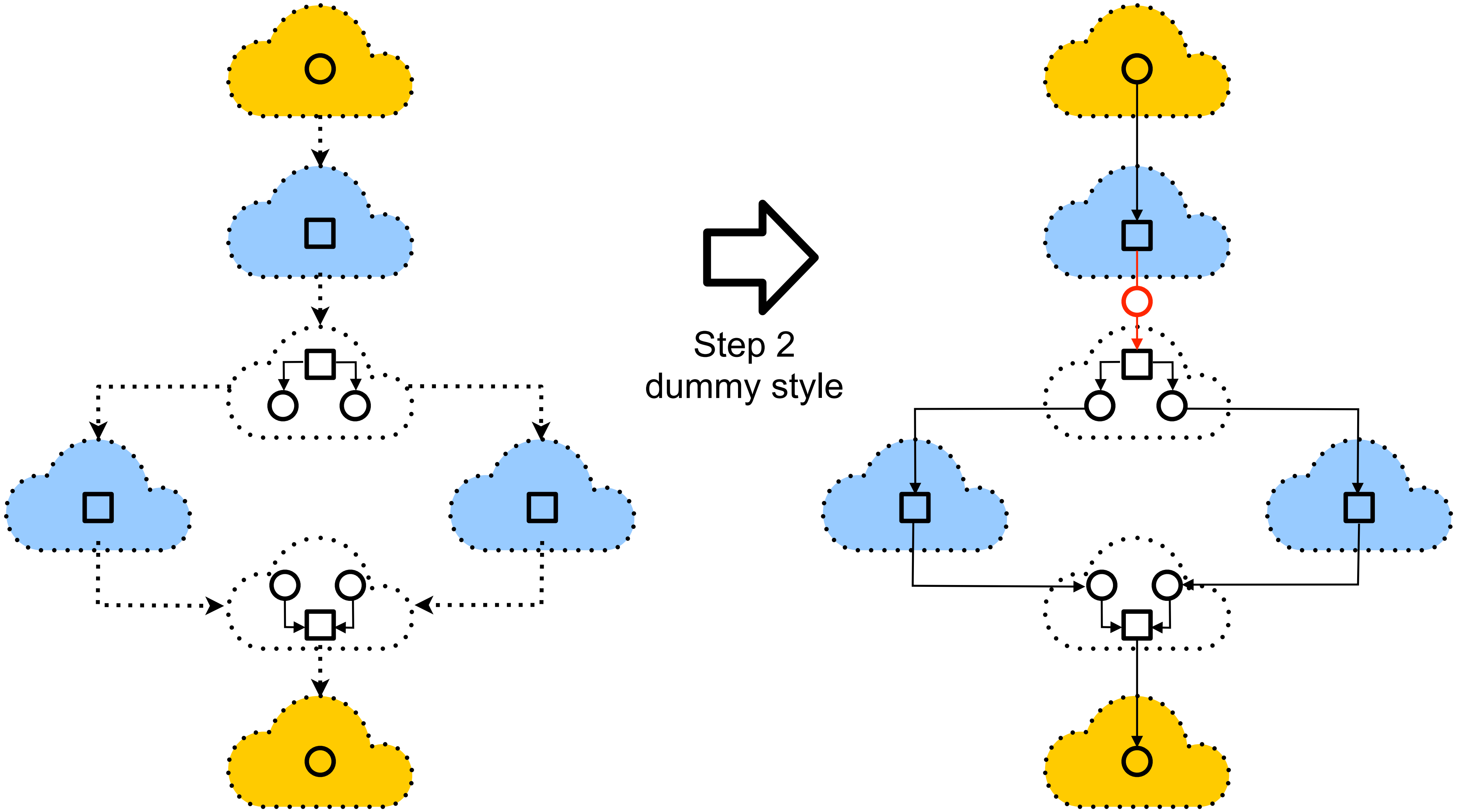
We replace each event, function and connector separately with small net fragments



Step 1
events
functions
connectors

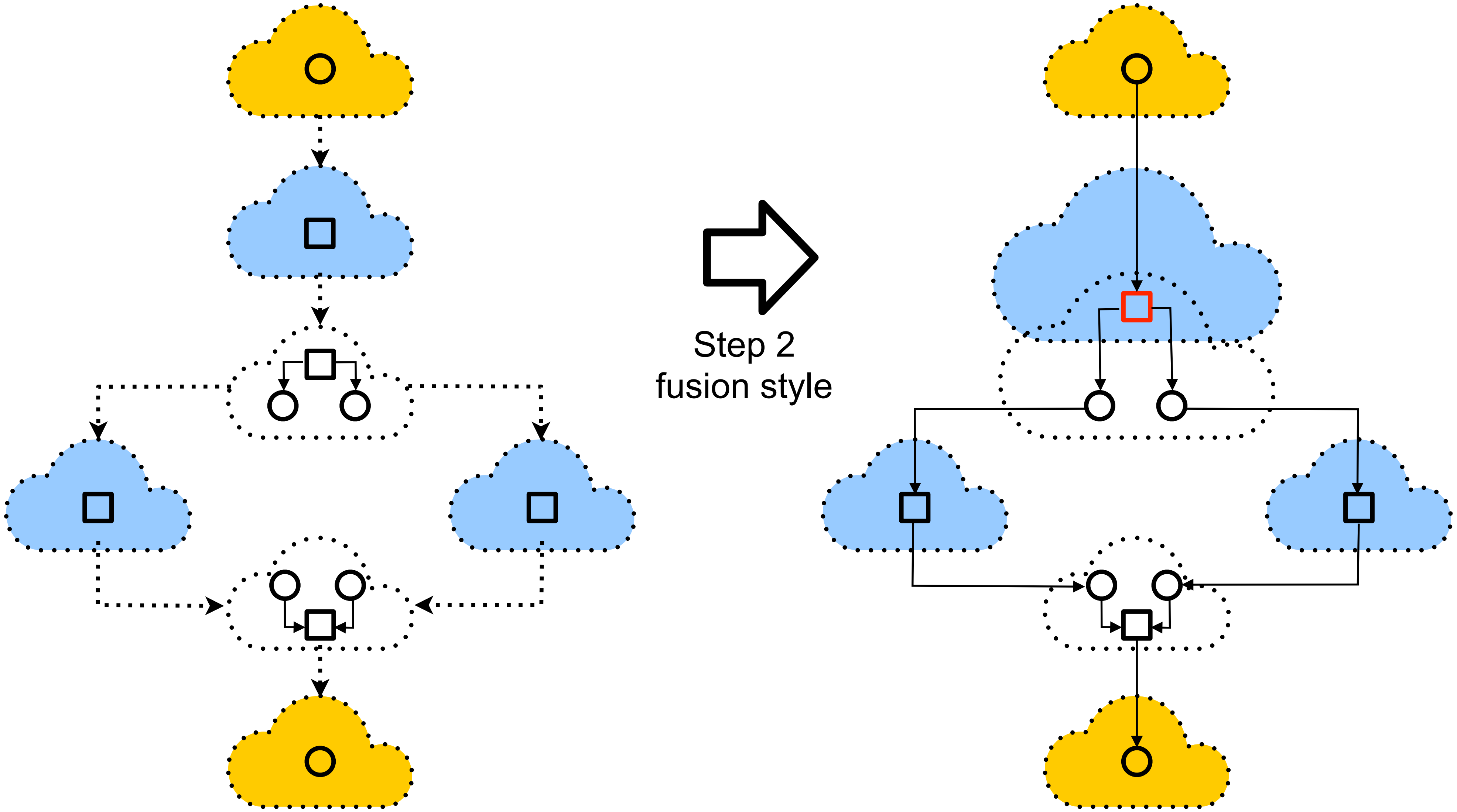
Step 2: dummy style

Then we connect the fragments together
(we may decide to introduce dummy places / transitions)

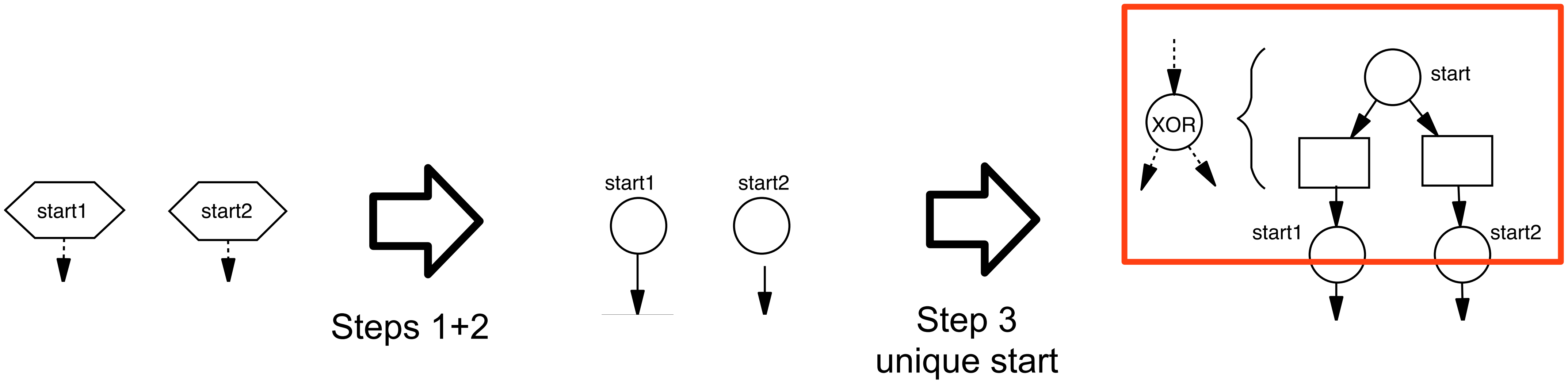


Step 2: fusion style

Then we connect the fragments together
(or we may decide to merge places / transitions)



Step 3: unique start

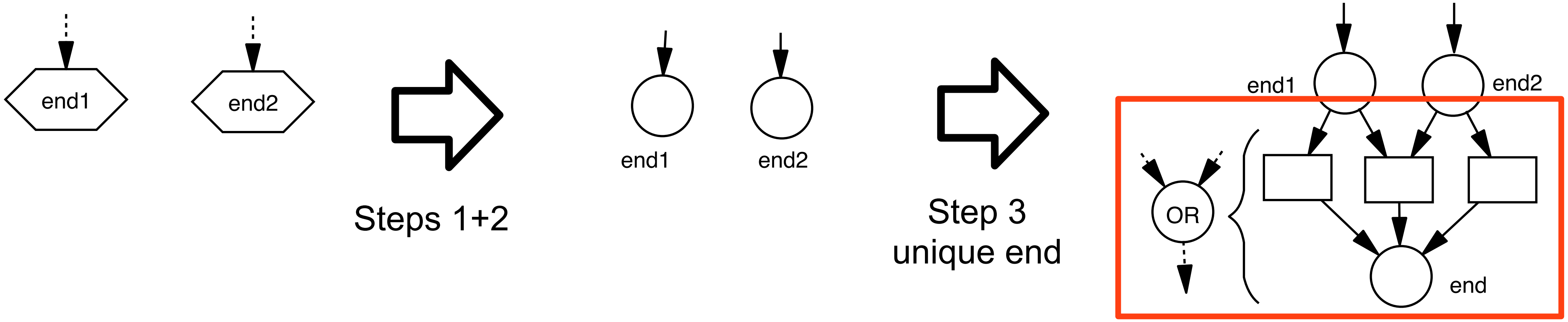


XOR start

Steps 1+2

Step 3
unique start

Step 3: unique end



Steps 1+2

Step 3
unique end

OR end

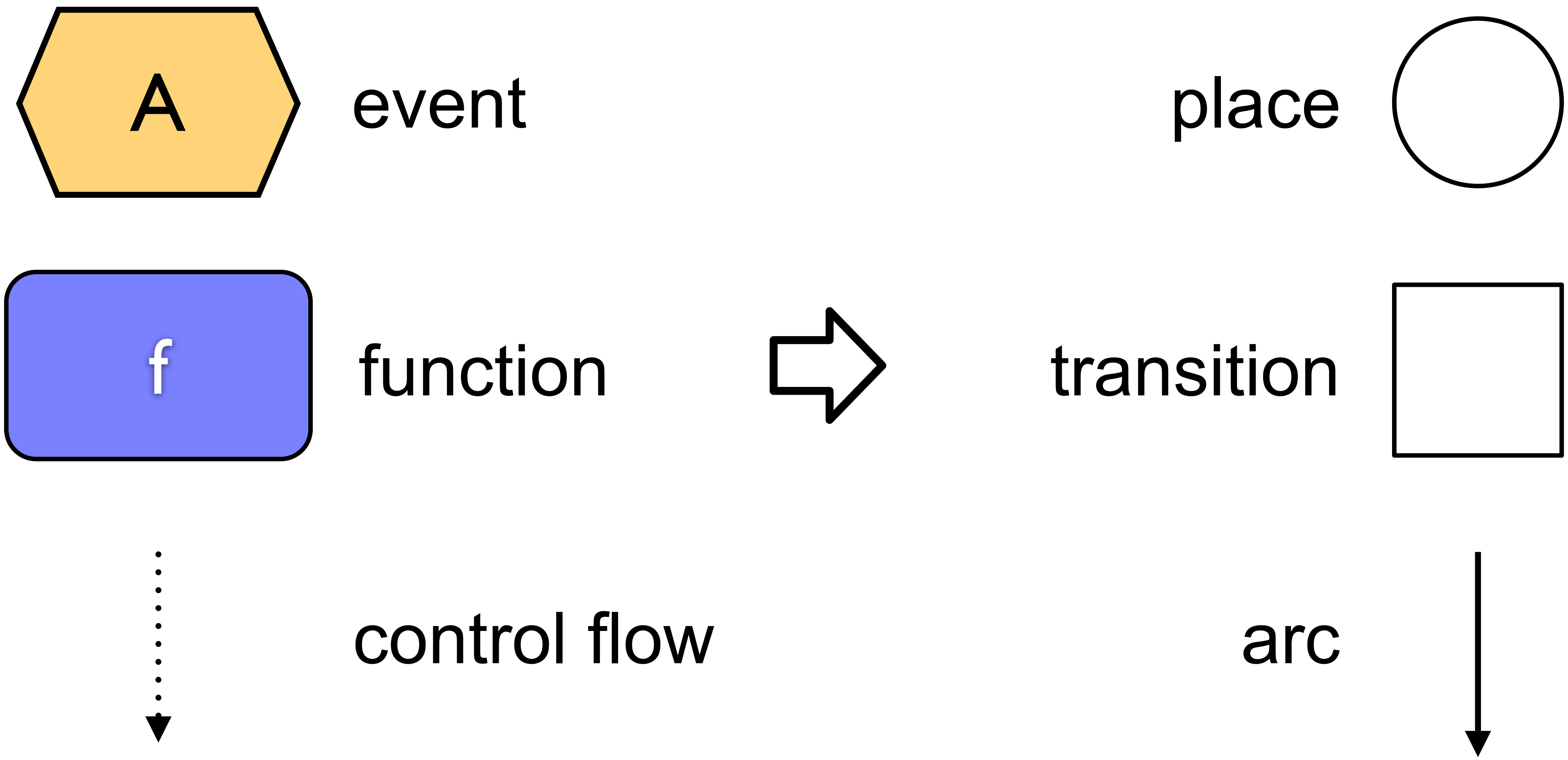
(sometimes XOR/AND can be preferred)

Three approaches

We overview **three** different translations

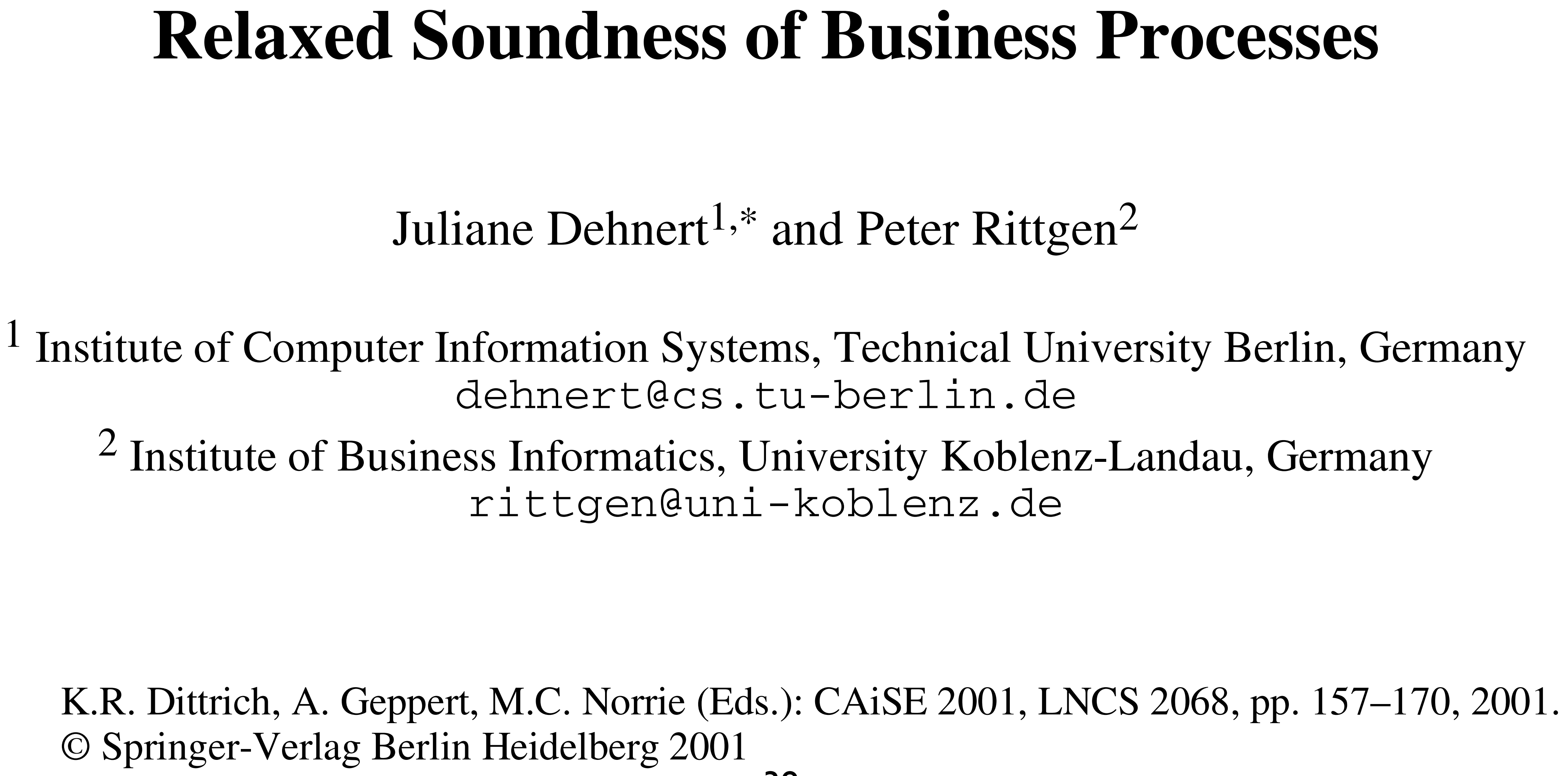
n.	**trickiness**	**style**	**applicability**	**outcome**
1st	easy	fusion	any EPC	likely unsound, (relaxed soundness)
2nd	medium, context dependent	(dummy)	simplified EPC event function alternation, no OR connectors	free-choice net
3rd	hard, context dependent	dummy	decorated EPC join-split correspondence, OR policies	accurate analysis

Commonalities



EPC element	net fragment
event	place
function	transition
control flow	arc

First attempt (straight translation)



Relaxed Soundness of Business Processes

Juliane Dehnert[1,*] and Peter Rittgen[2]

[1] Institute of Computer Information Systems, Technical University Berlin, Germany
`dehnert@cs.tu-berlin.de`

[2] Institute of Business Informatics, University Koblenz-Landau, Germany
`rittgen@uni-koblenz.de`

K.R. Dittrich, A. Geppert, M.C. Norrie (Eds.): CAiSE 2001, LNCS 2068, pp. 157–170, 2001.

Rationale

EPC success is due to its **simplicity**

EPC diagrams lack a consistent semantics:
ambiguous and flawed process descriptions
can arise in the **design phase**

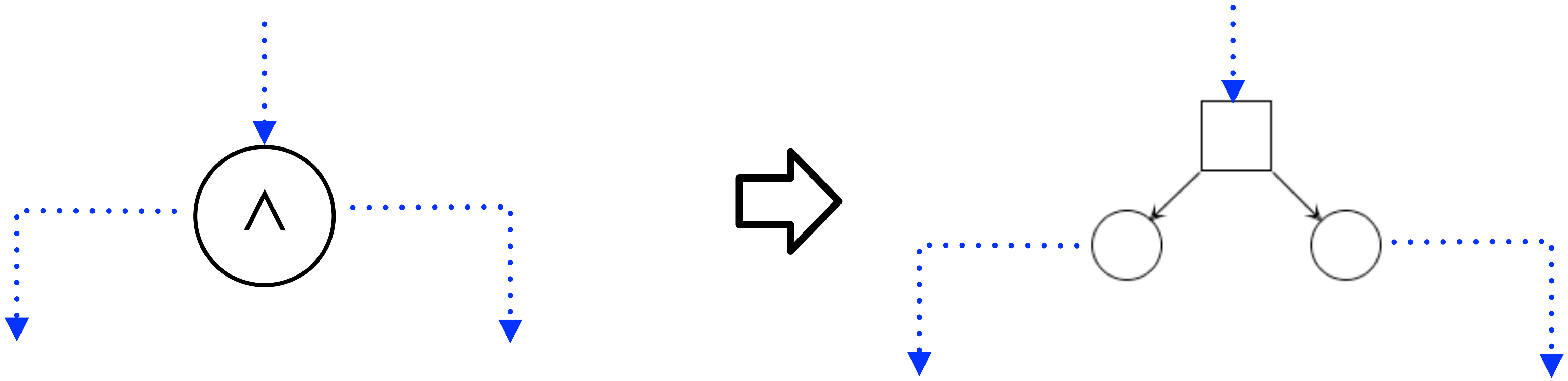
it is important to find out **flaws** as **soon** as possible

therefore

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

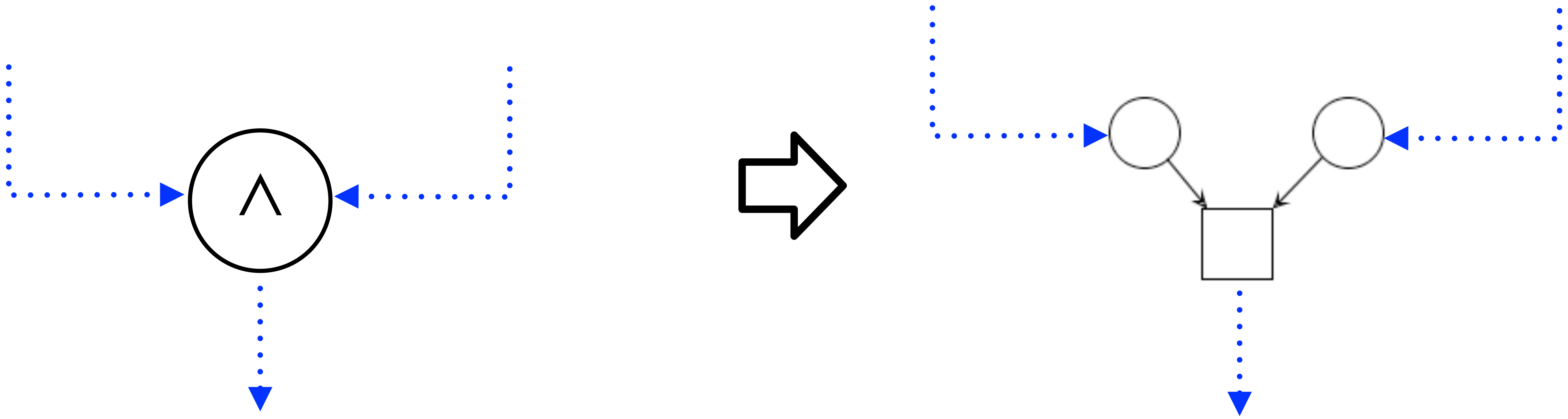
Step 1: AND split

EPC element **net fragment**



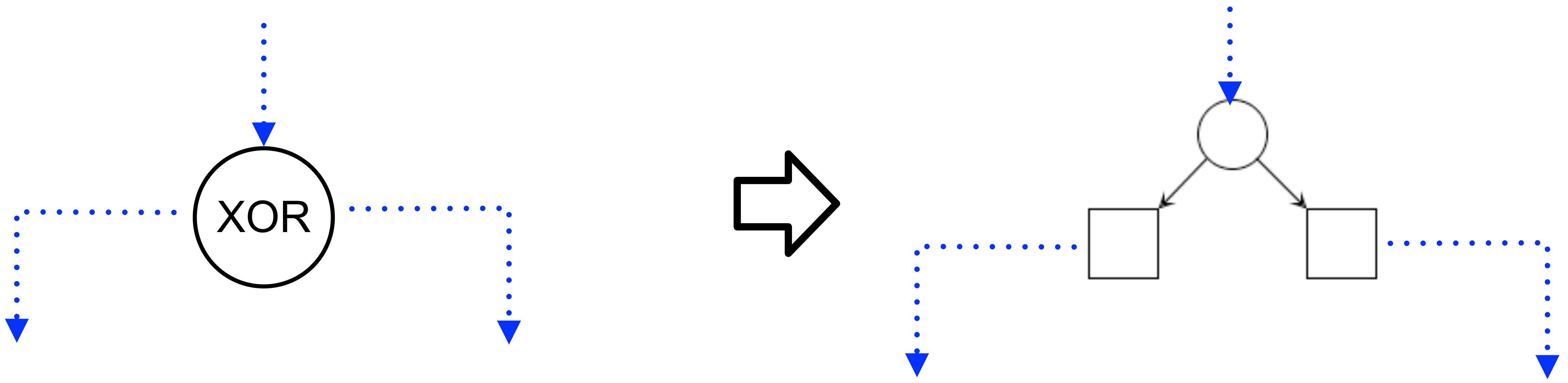
Step 1: AND join

EPC element **net fragment**



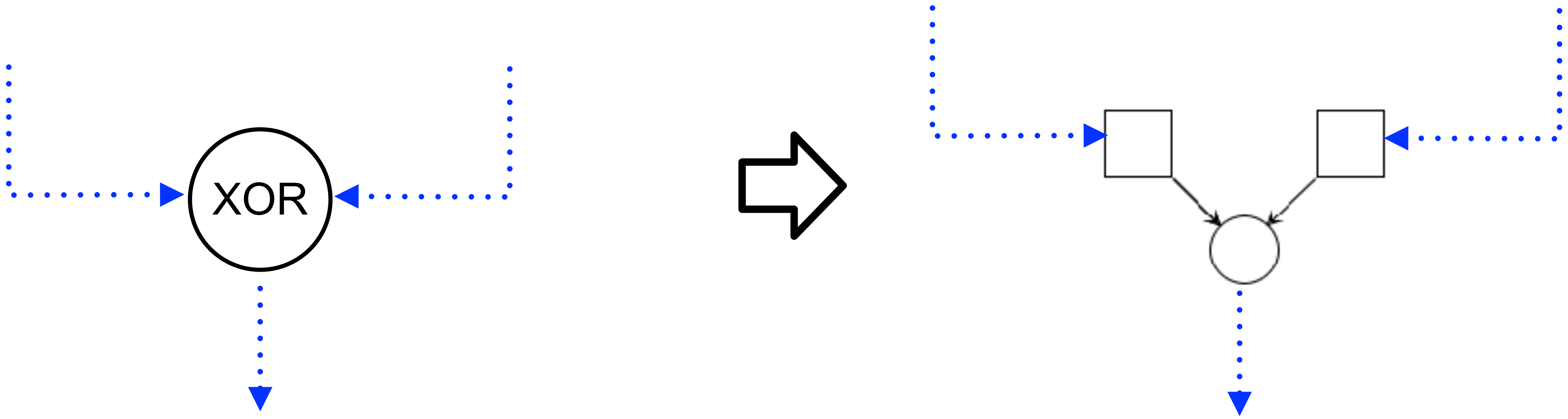
Step 1: XOR split

EPC element **net fragment**



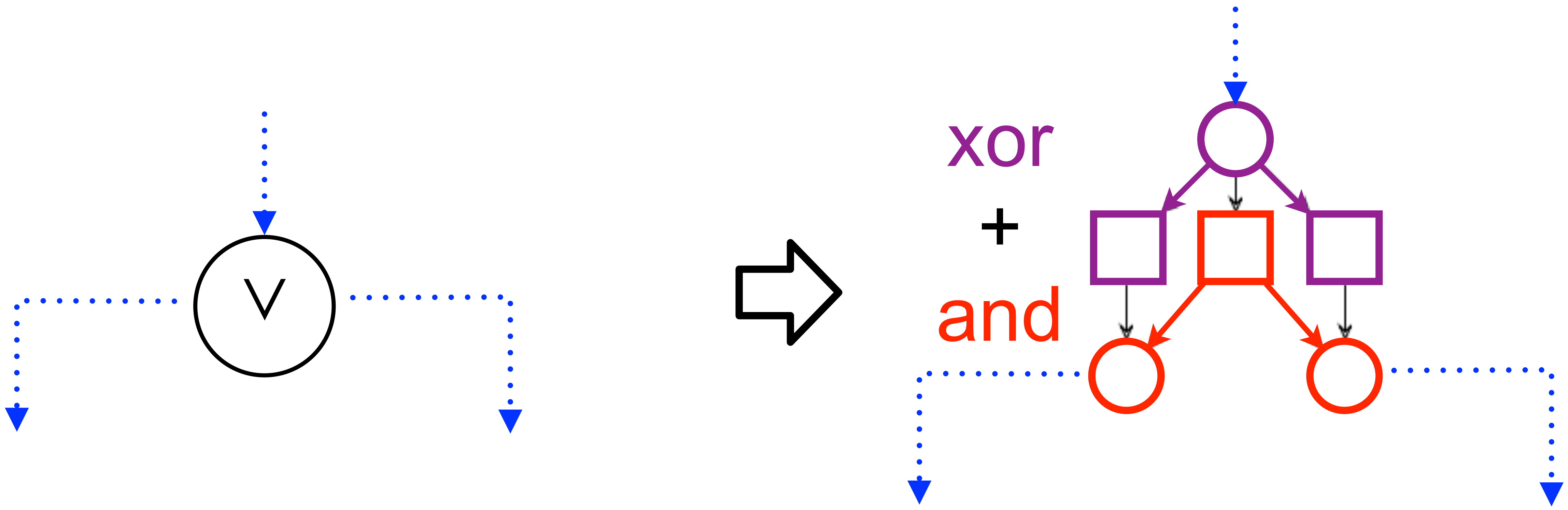
Step 1: XOR join

EPC element **net fragment**



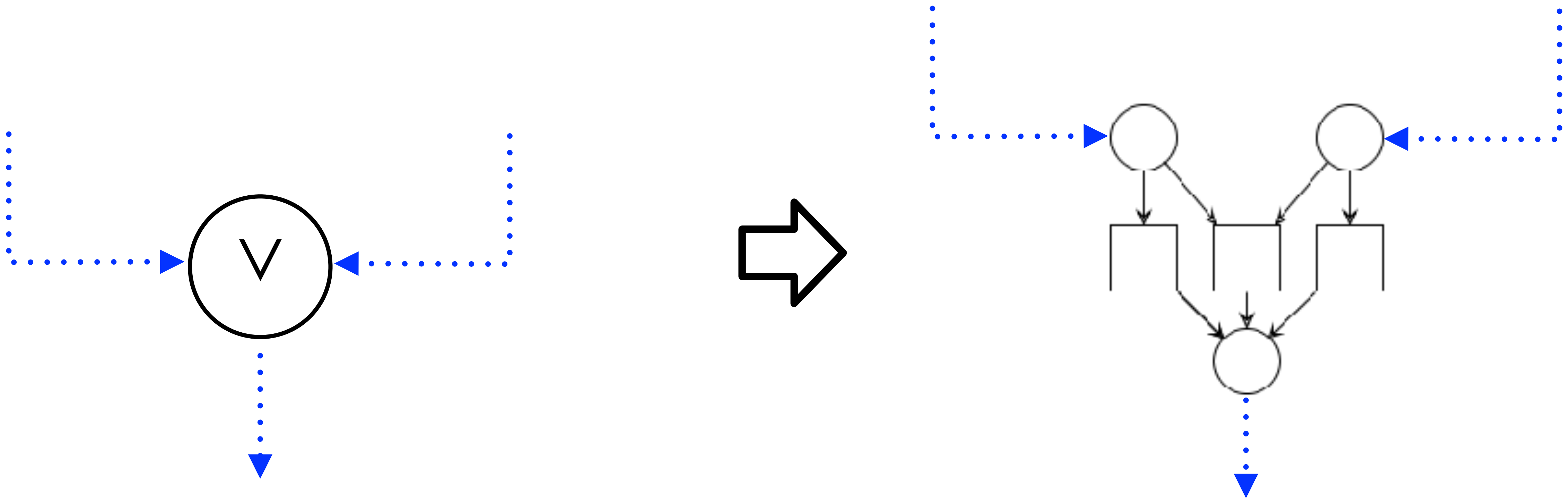
Step 1: OR split

EPC element **net fragment**

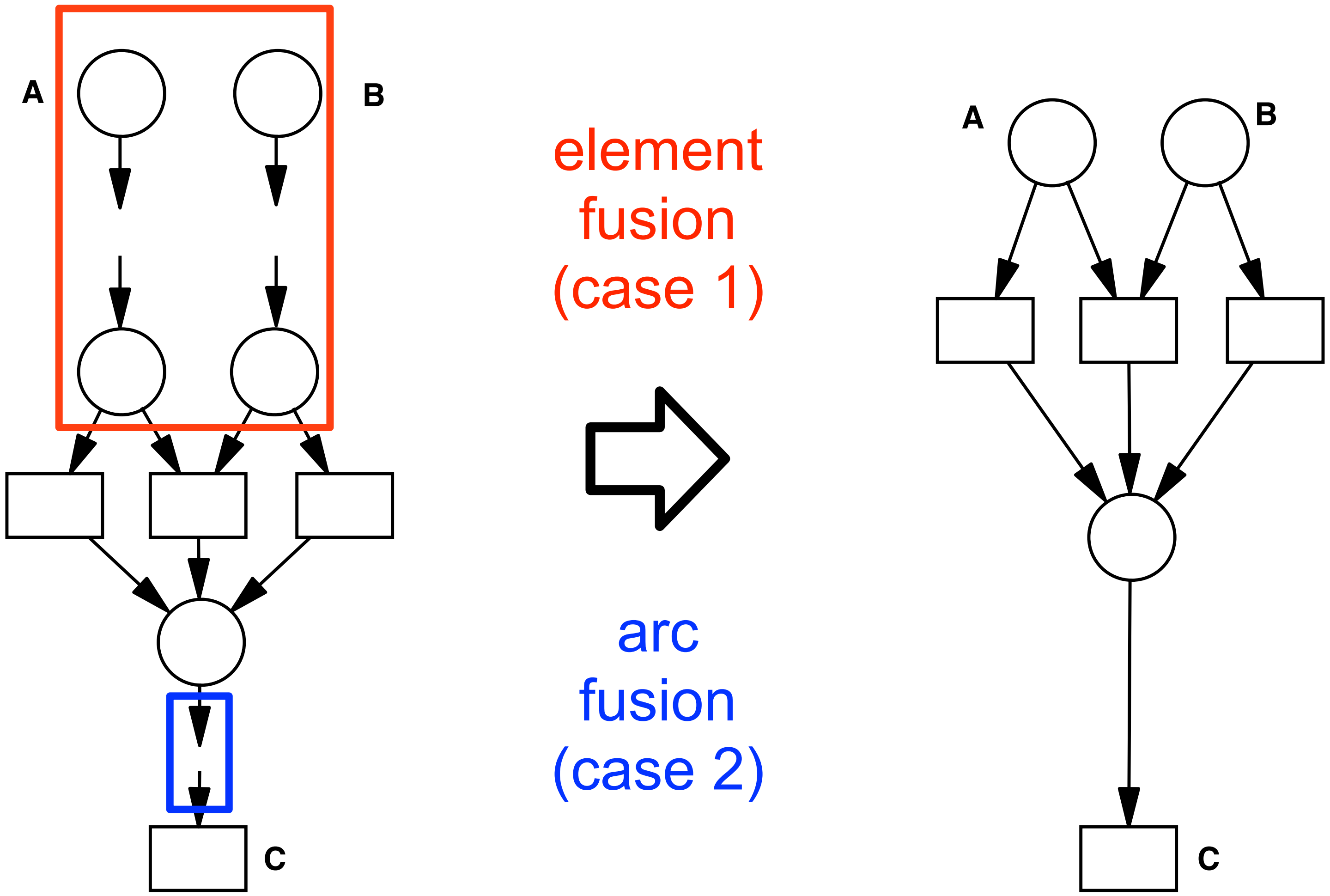


Step 1: OR join

EPC element **net fragment**



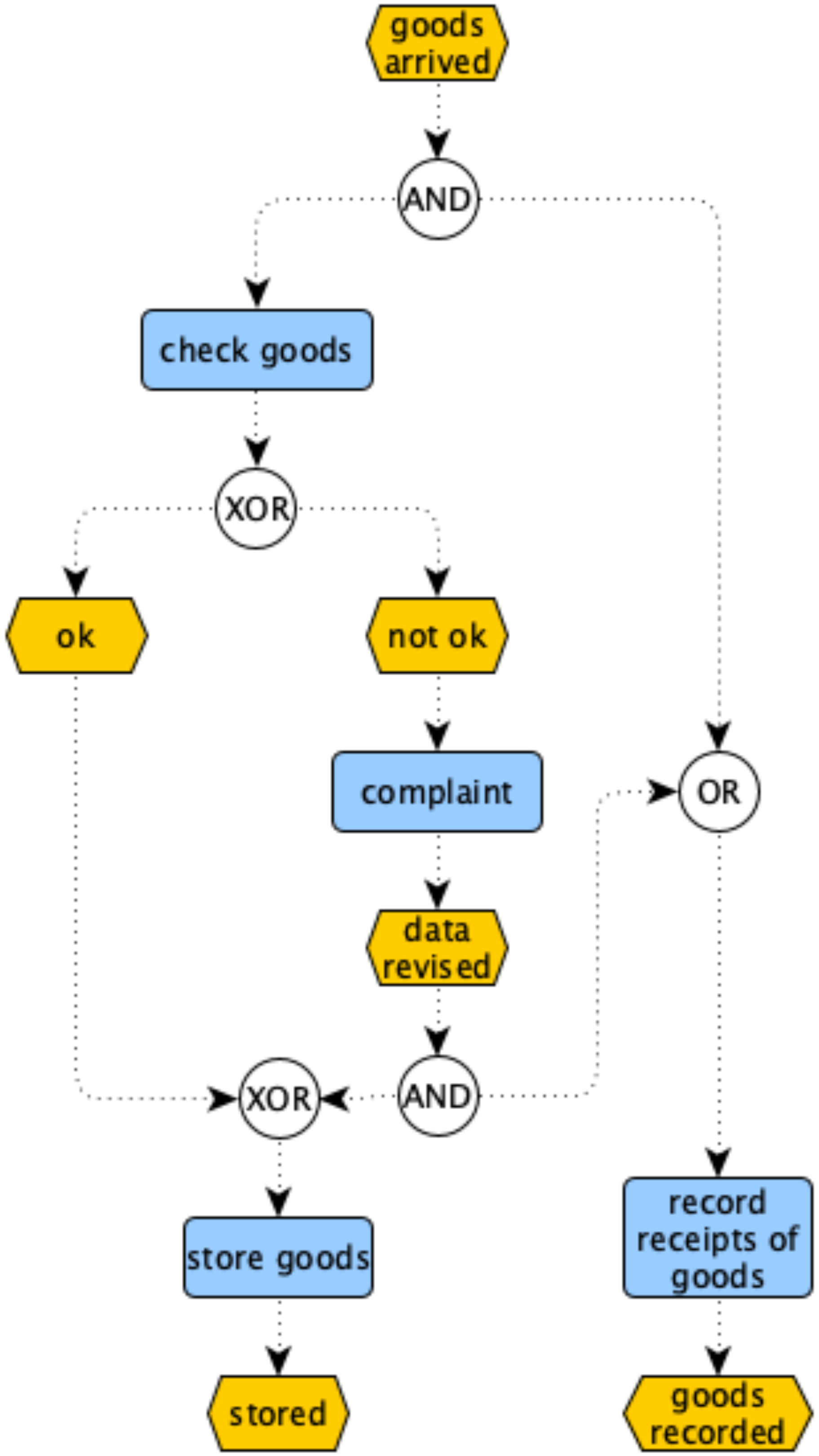
Step 2: fusion style



element fusion (case 1)

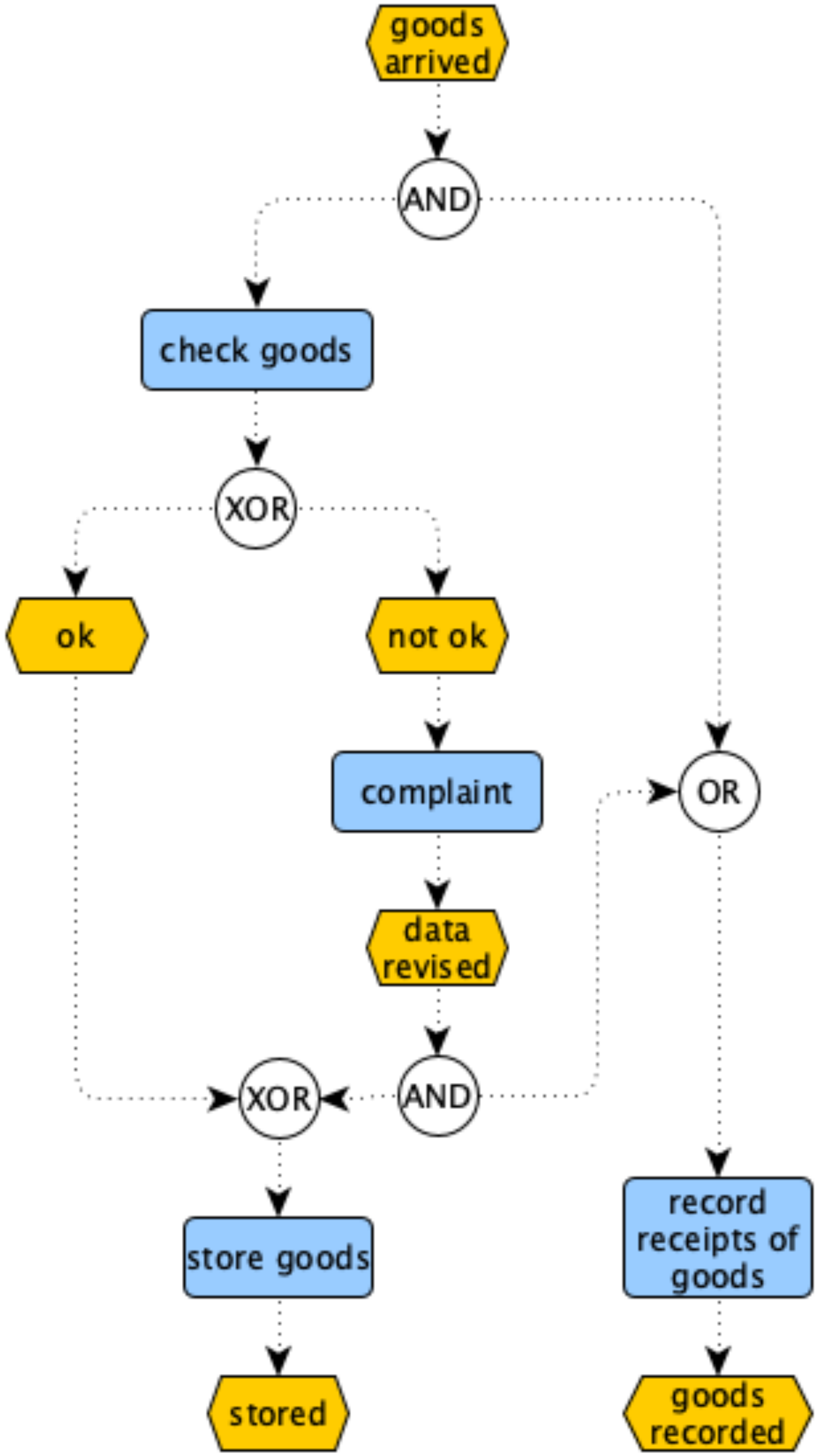
arc fusion (case 2)

Example

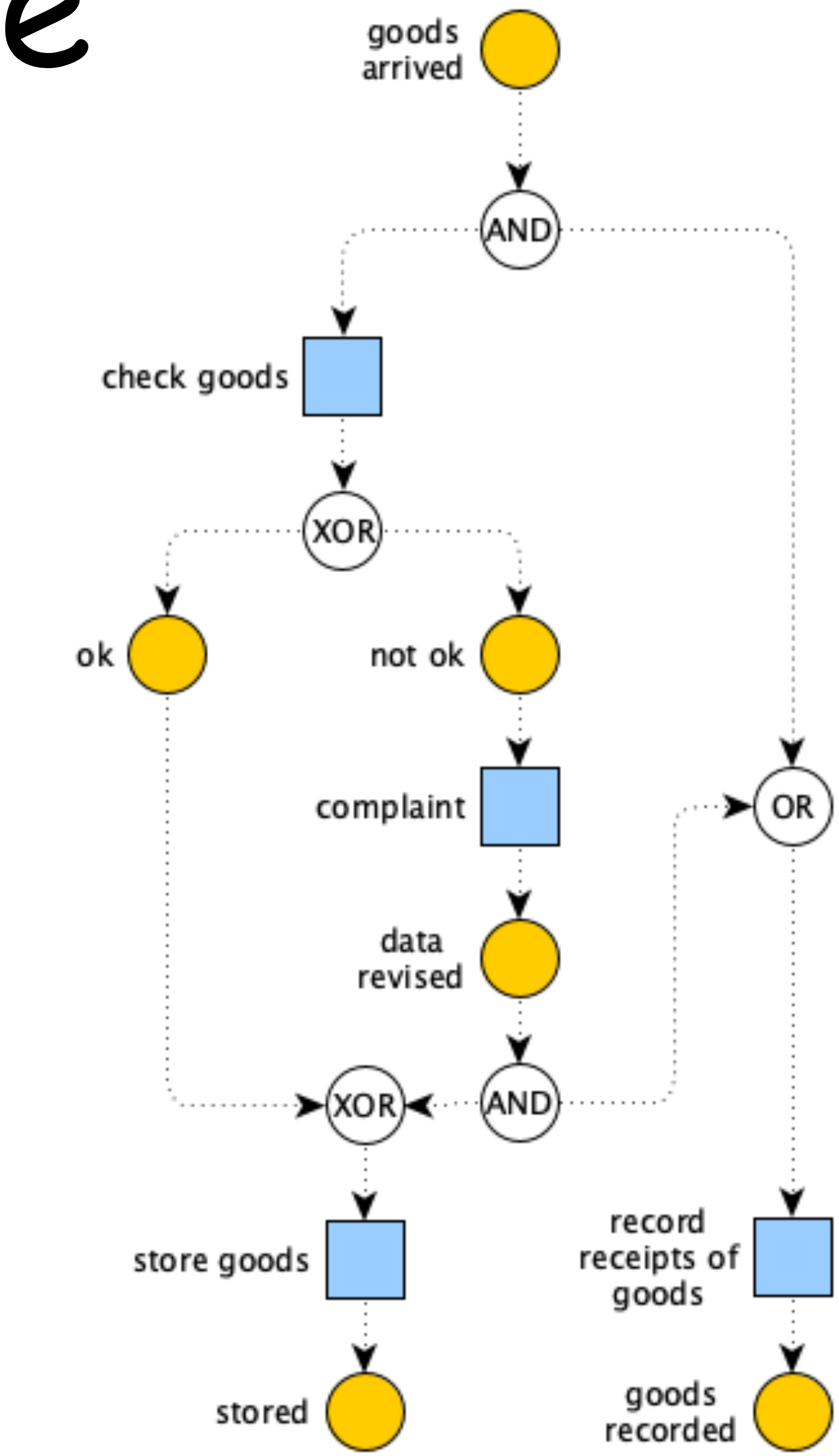


Sound?

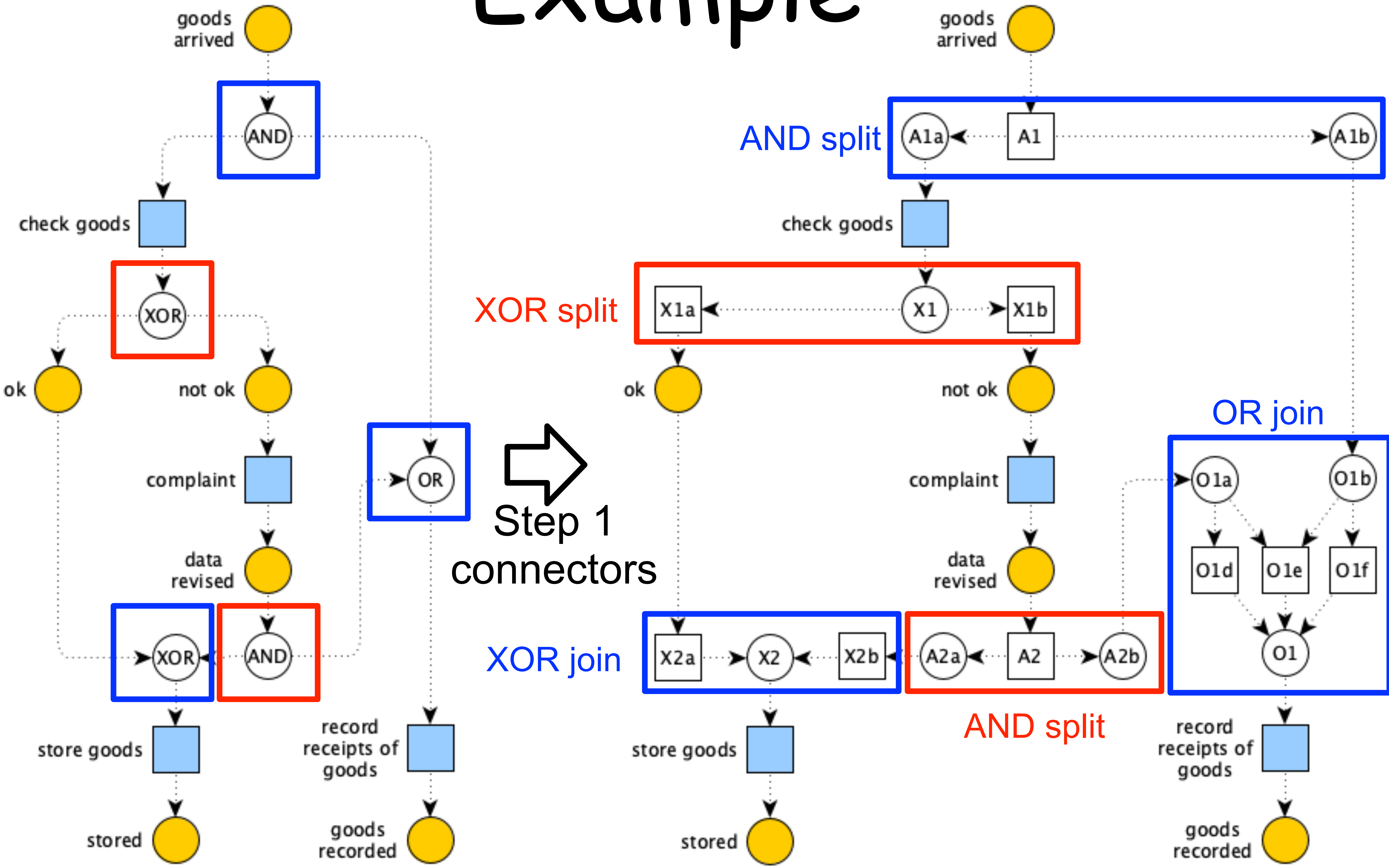
Example



Step 1
events and
functions



Example



AND split

XOR split

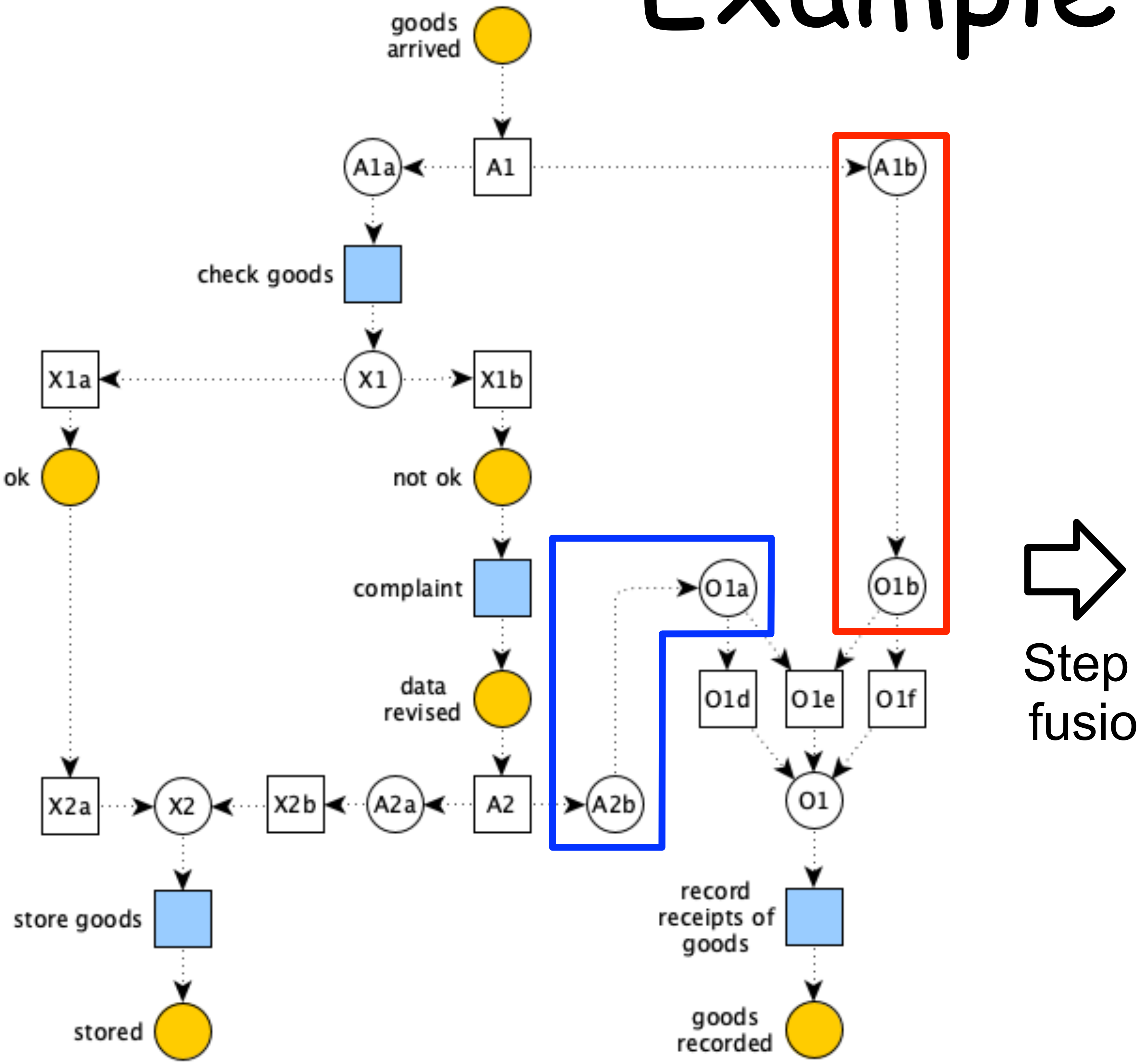
Step 1 connectors

OR join

XOR join

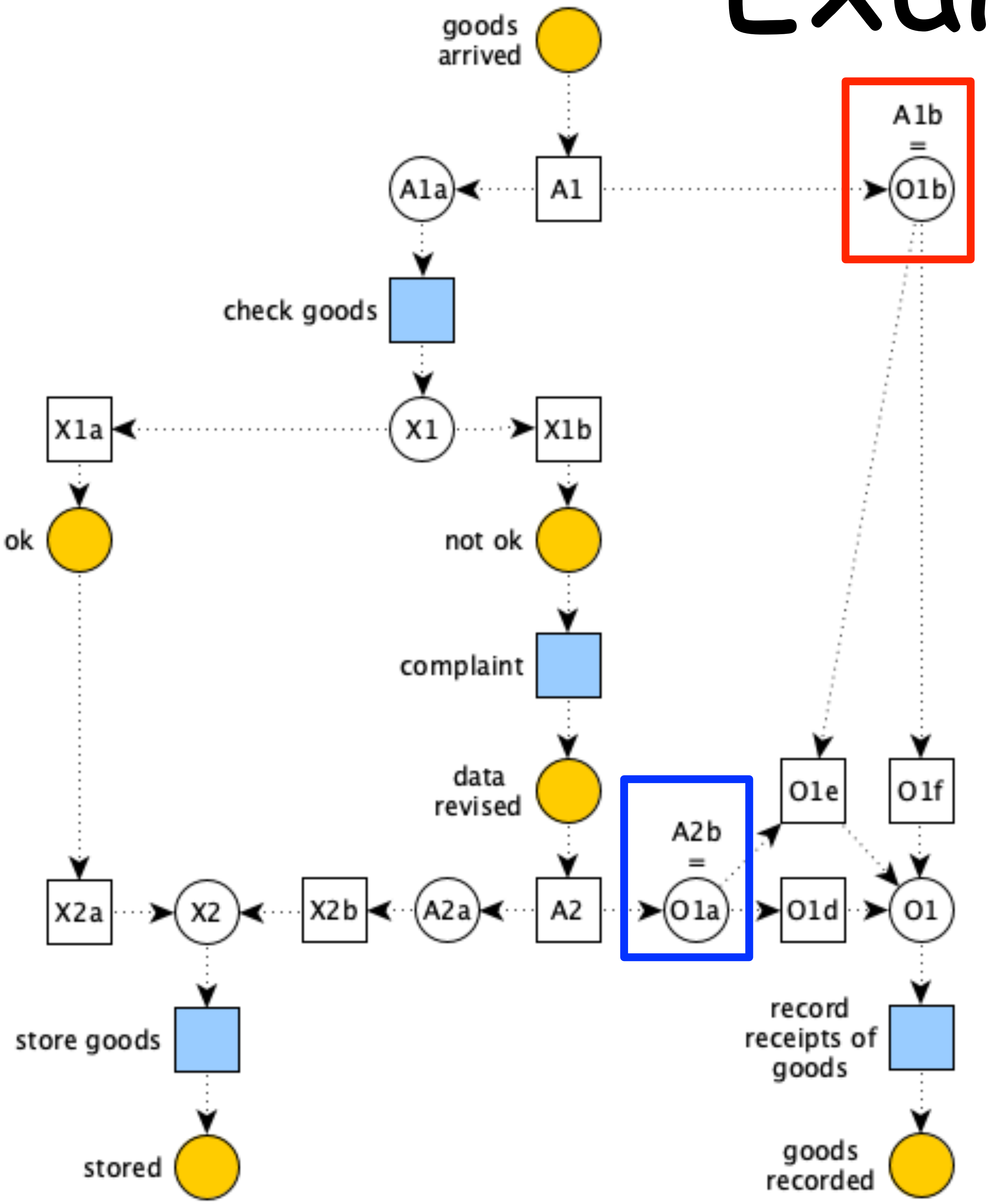
AND split

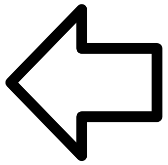
Example



Step 2 fusion

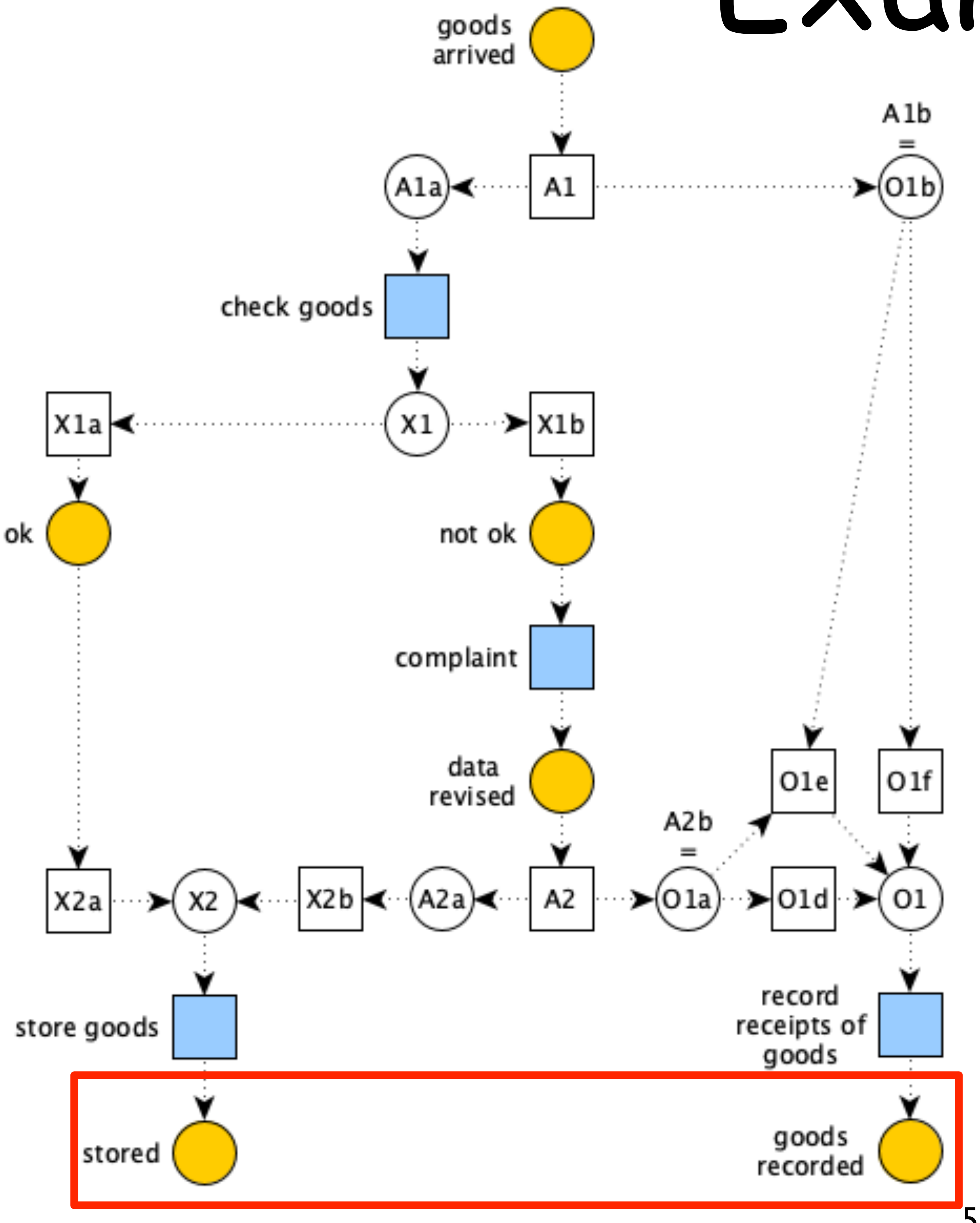
Example





Step 2
fusion

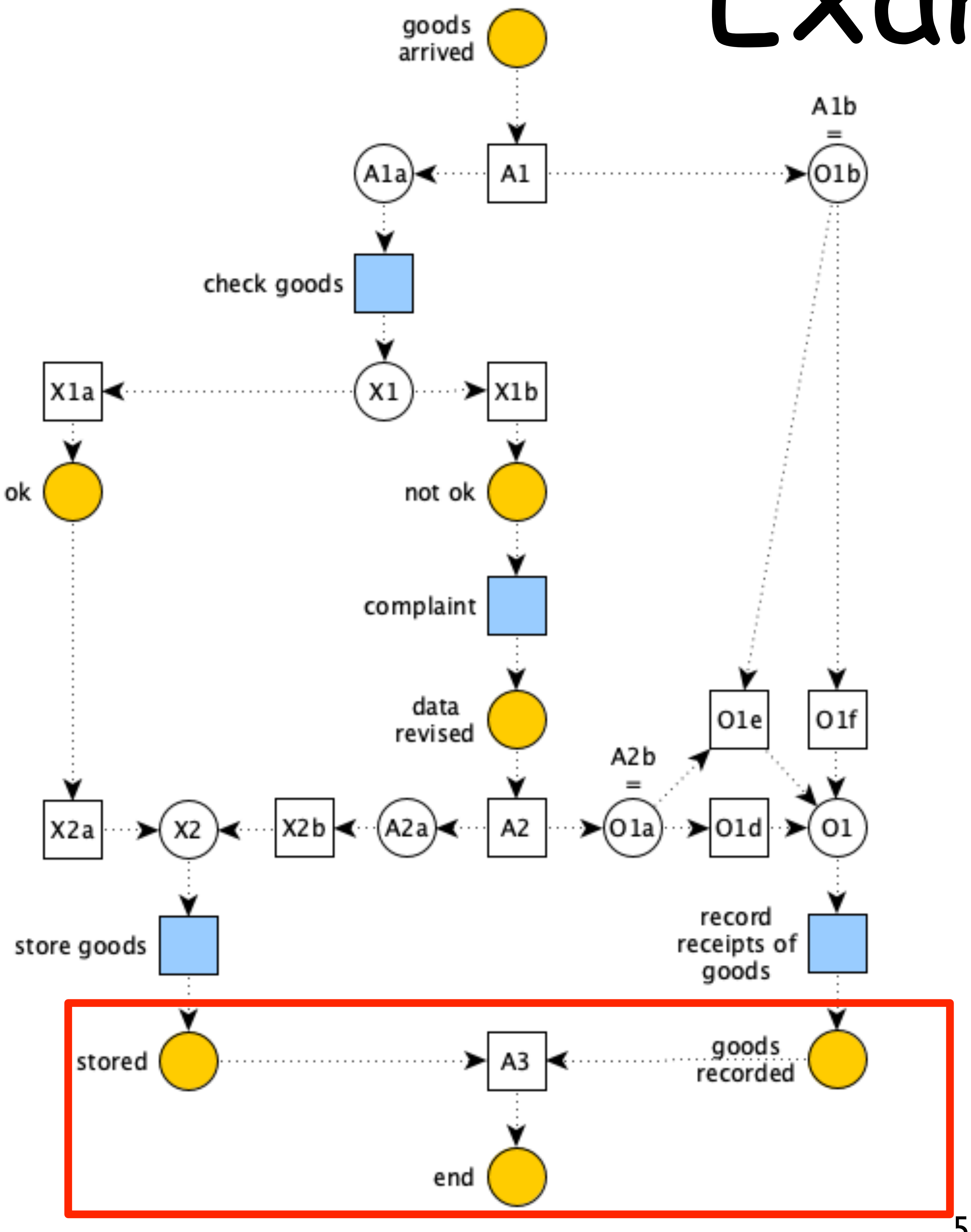
Example



Step 3
unique end

implicit AND join (because of A2)

Example

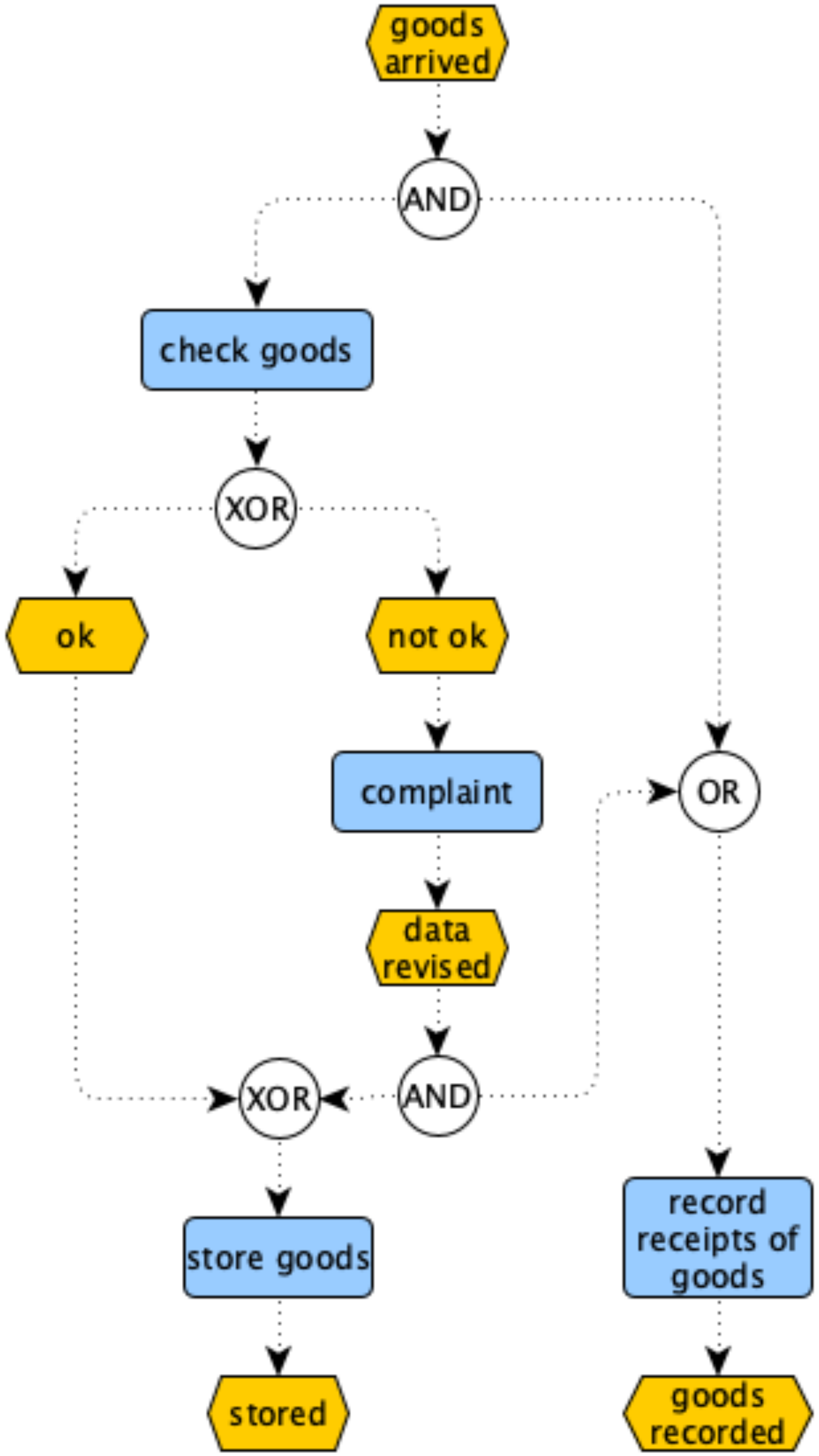


Step 3
unique end

implicit AND join (because of A2)

Example

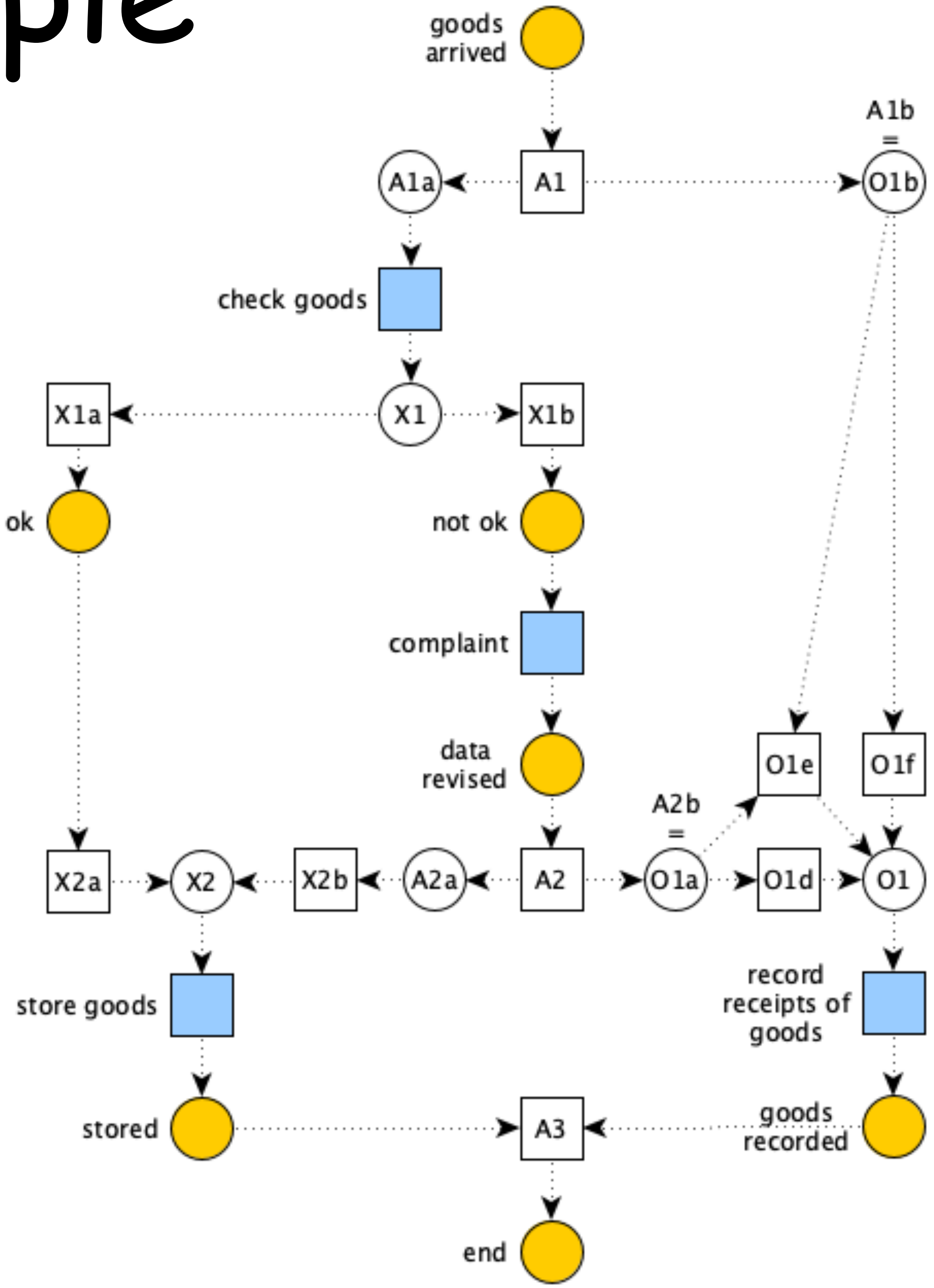
EPC



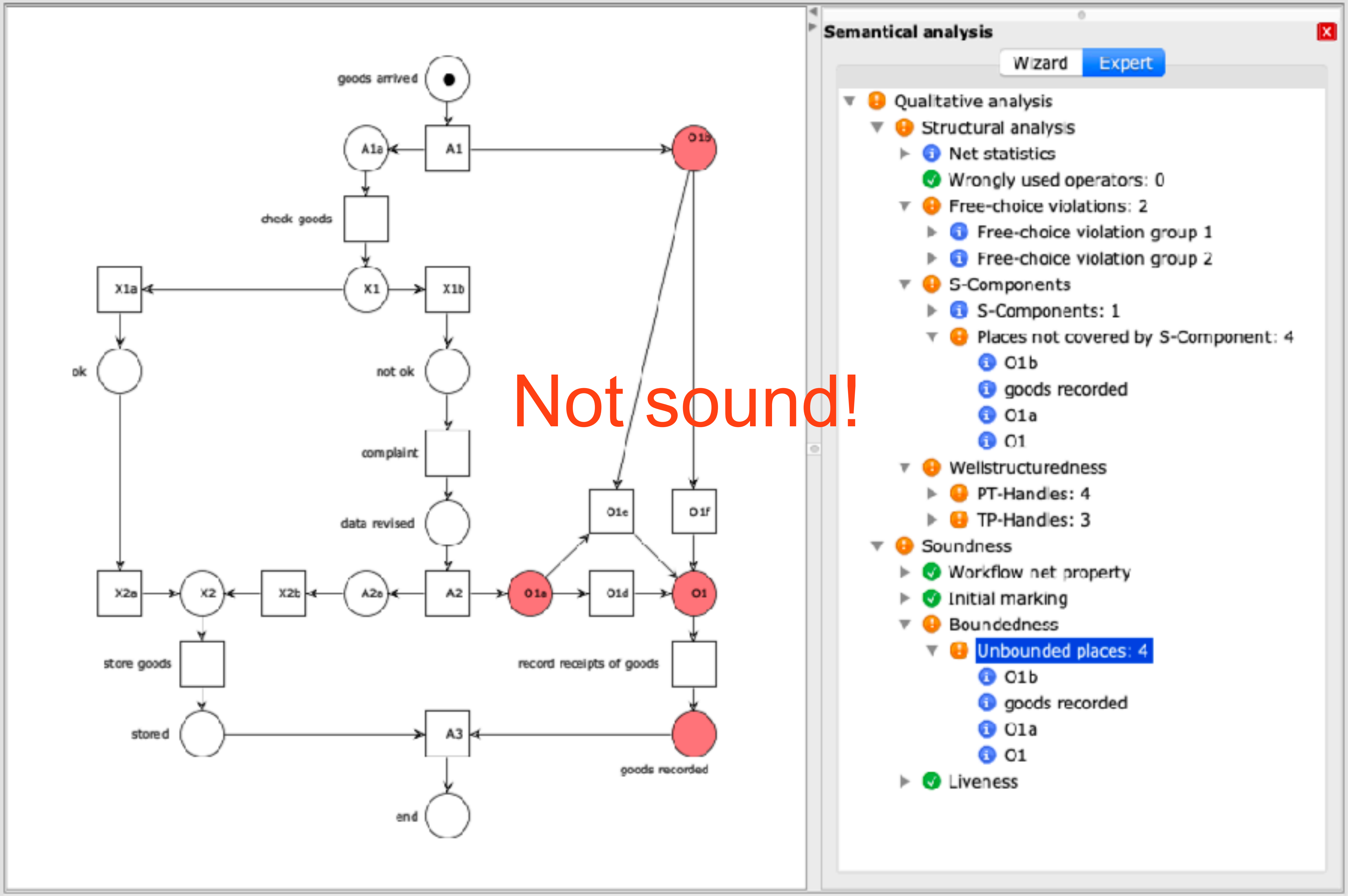
Sound?

Steps 1+2+3

wf net



Soundness analysis



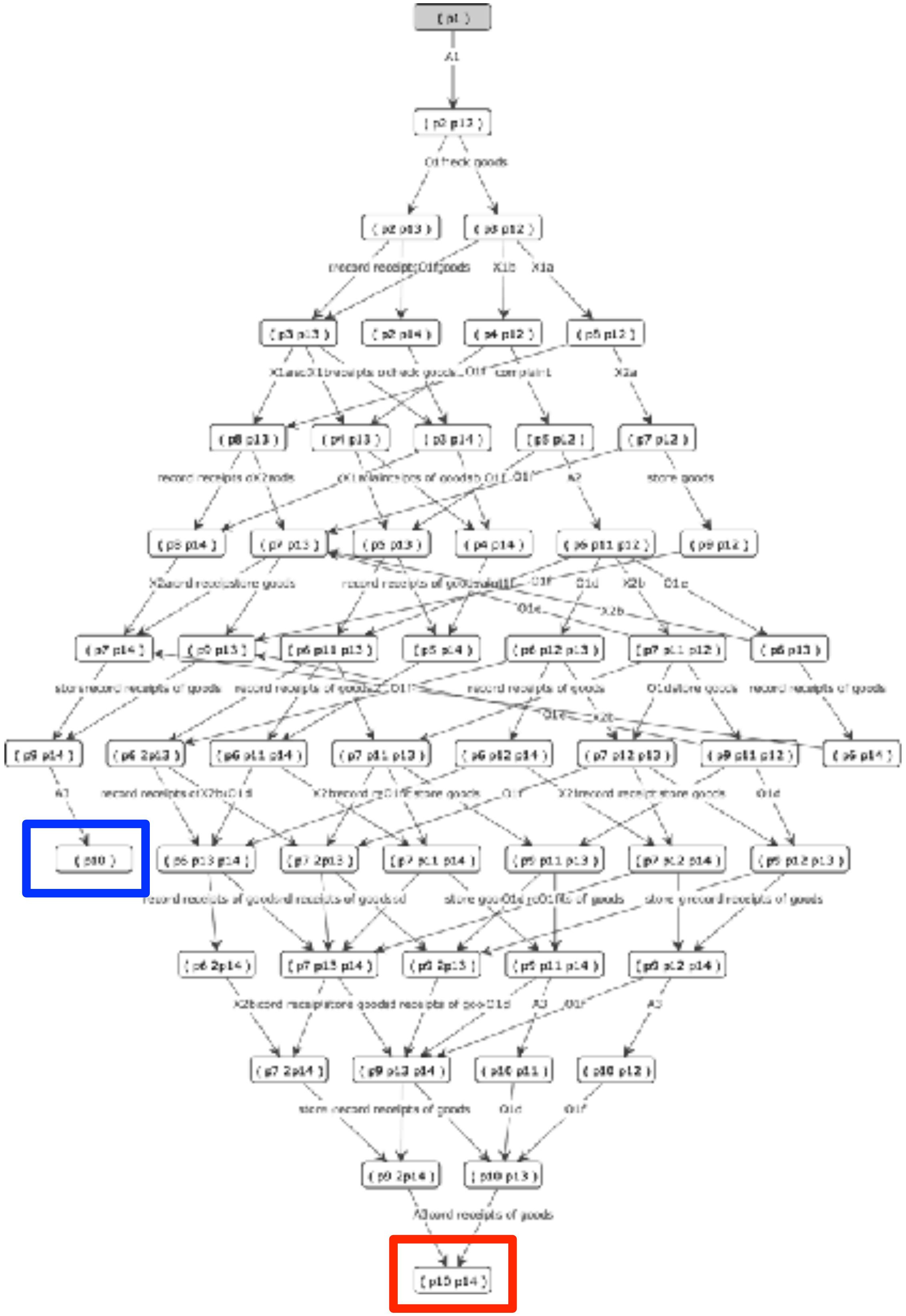
Not sound!

Semantical analysis

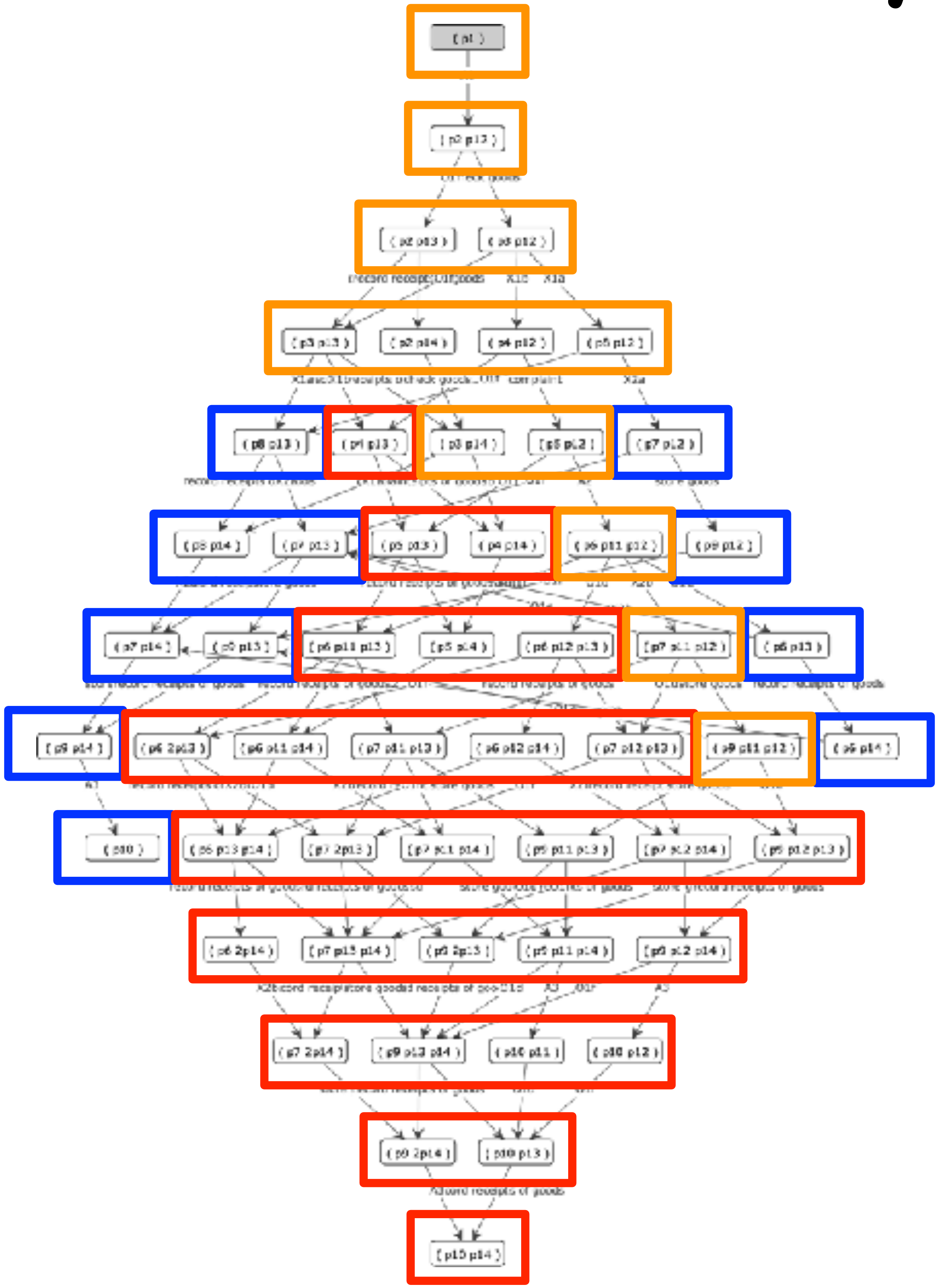
Wizard | Expert

- Qualitative analysis
  - Structural analysis
    - Net statistics
    - Wrongly used operators: 0
    - Free-choice violations: 2
      - Free-choice violation group 1
      - Free-choice violation group 2
    - S-Components
      - S-Components: 1
      - Places not covered by S-Component: 4
        - O1b
        - goods recorded
        - O1a
        - O1
    - Wellstructuredness
      - PT-Handles: 4
      - TP-Handles: 3
  - Soundness
    - Workflow net property
    - Initial marking
    - Boundedness
      - Unbounded places: 4
        - O1b
        - goods recorded
        - O1a
        - O1
    - Liveness

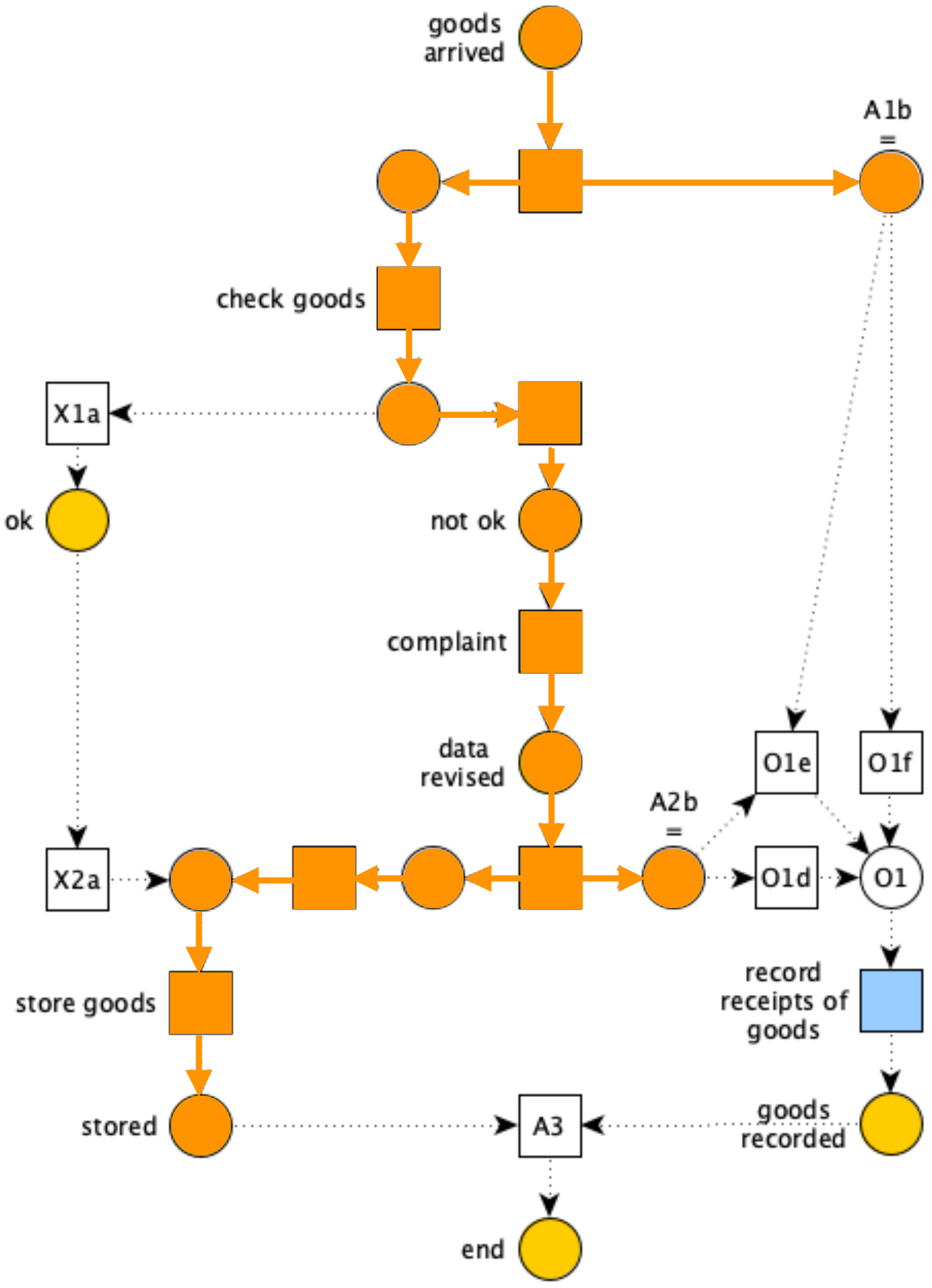
Soundness analysis



Soundness analysis

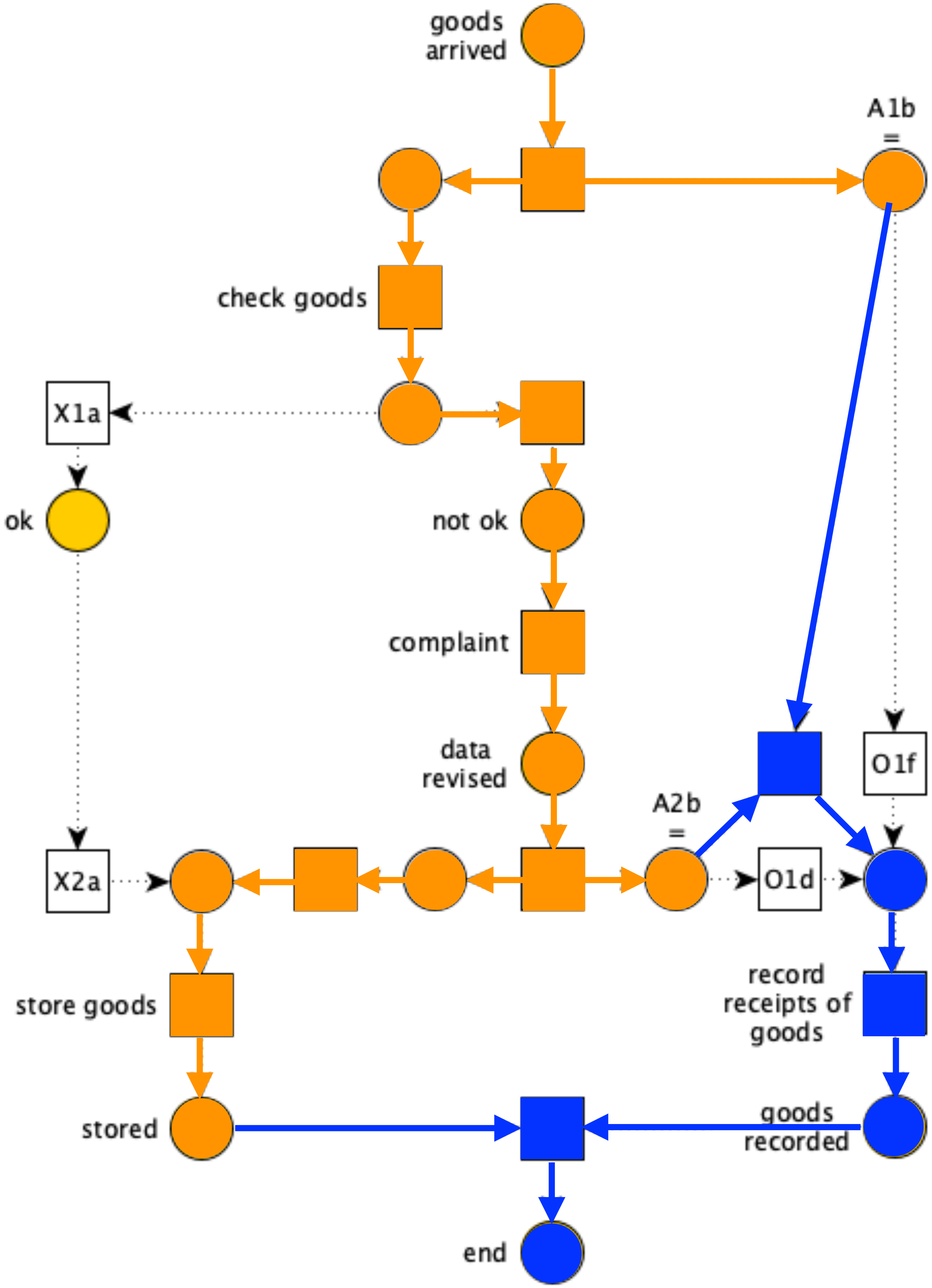


Soundness analysis



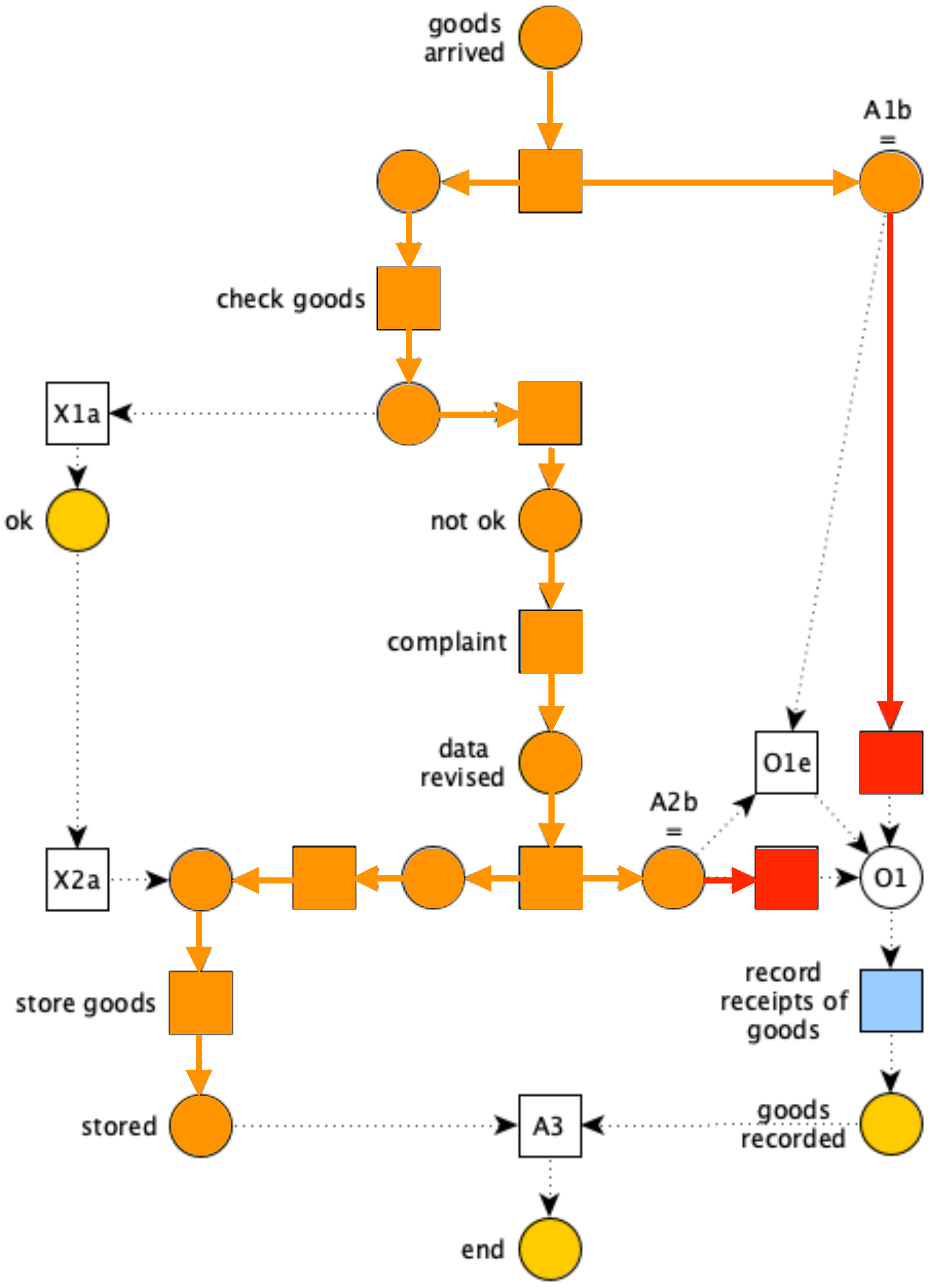
the right thing to do would be to fire O1e

Soundness analysis



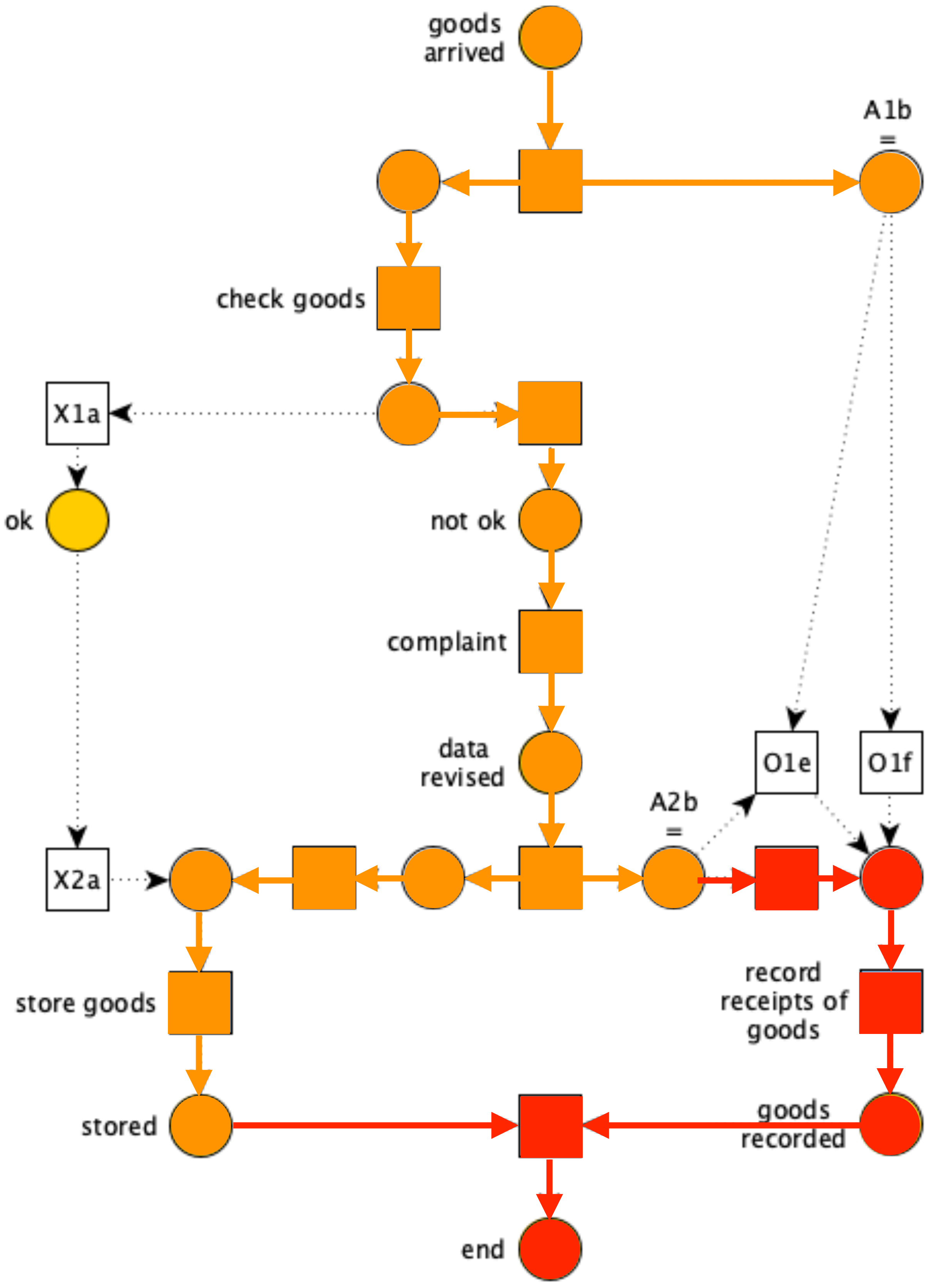
the right thing to do would be to fire O1e

Soundness analysis



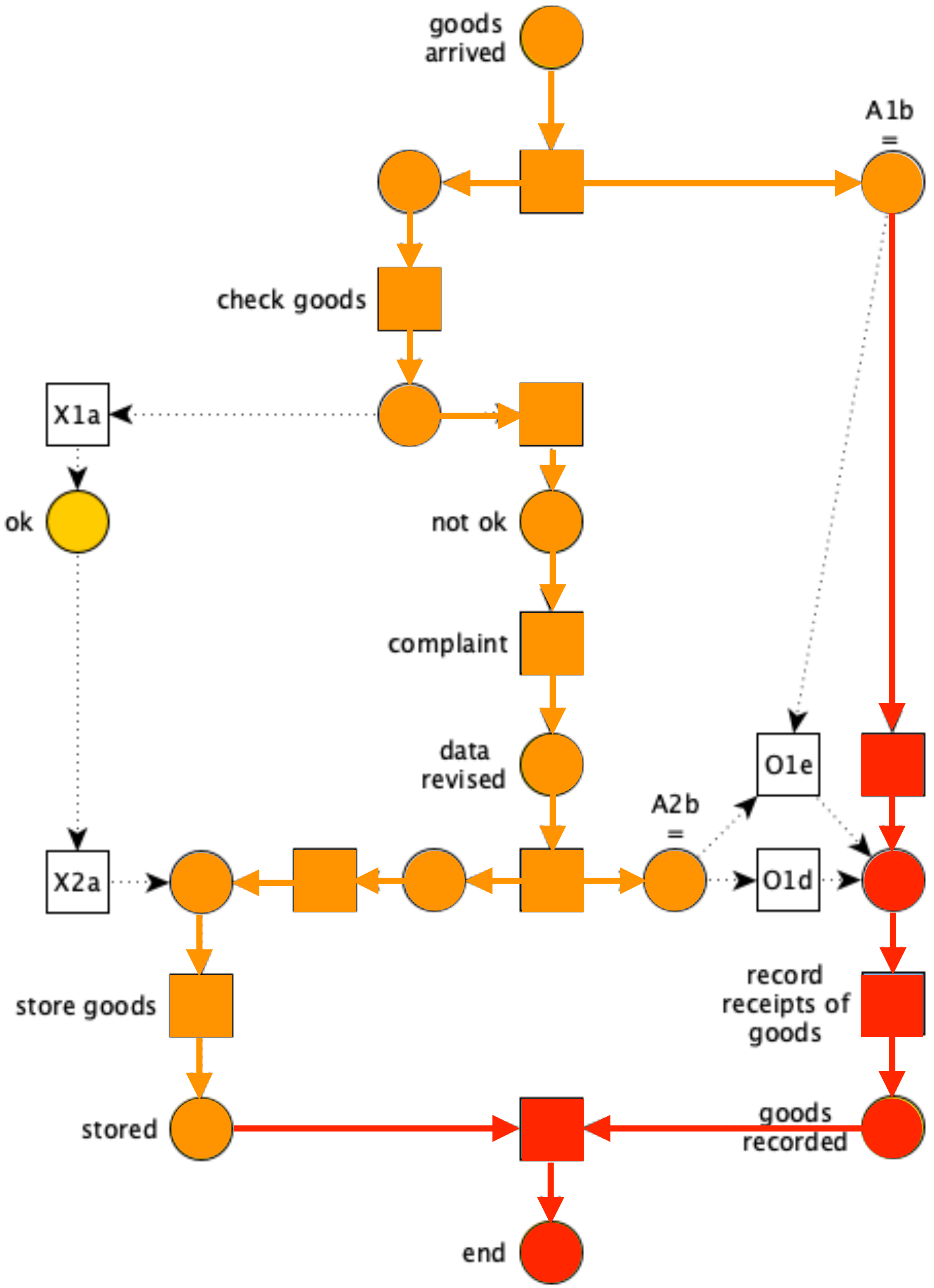
but O1f and O1d are enabled as well (OR semantics!)

Soundness analysis



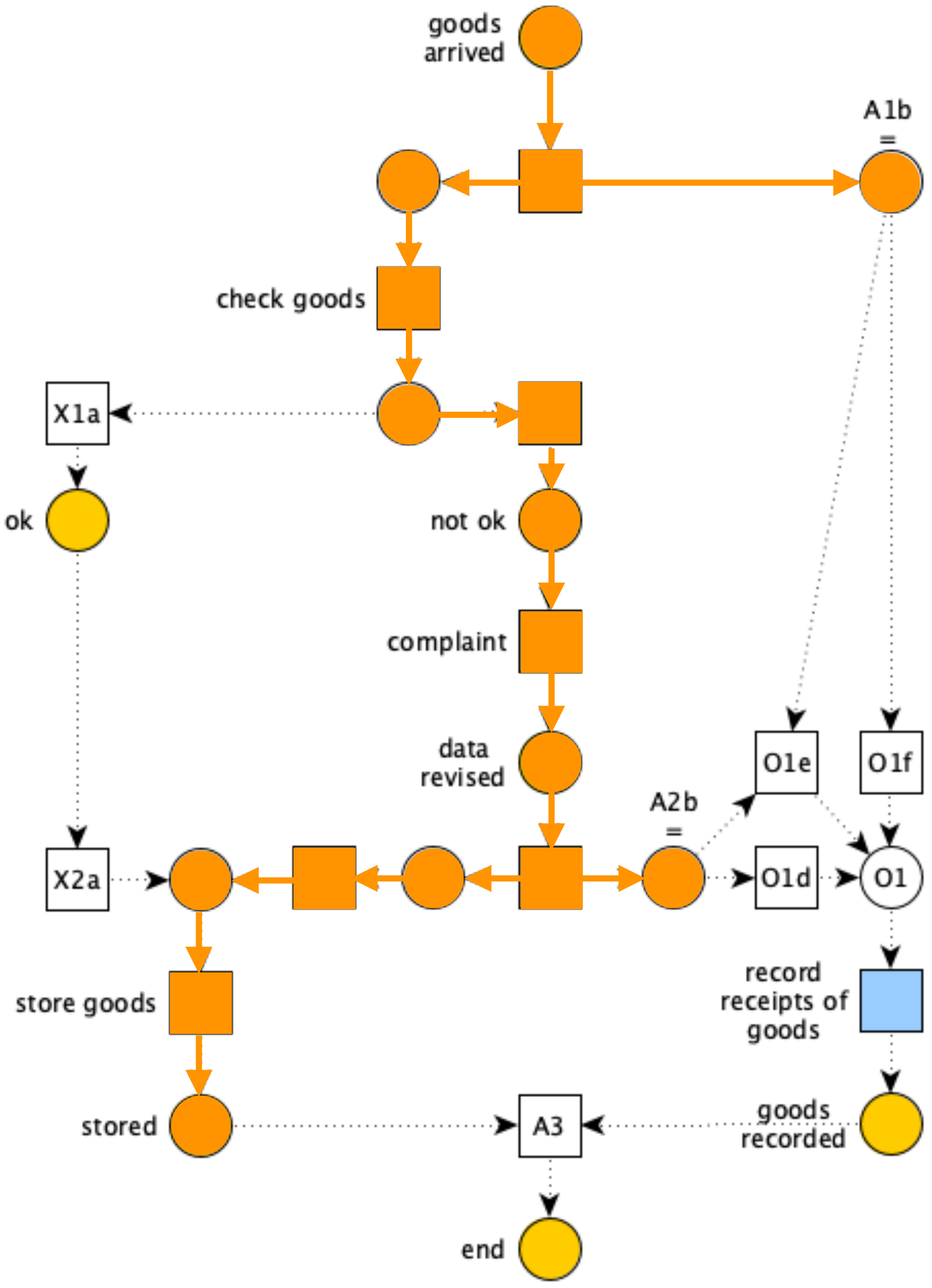
proper completion is not guaranteed (N* unbounded)

Soundness analysis



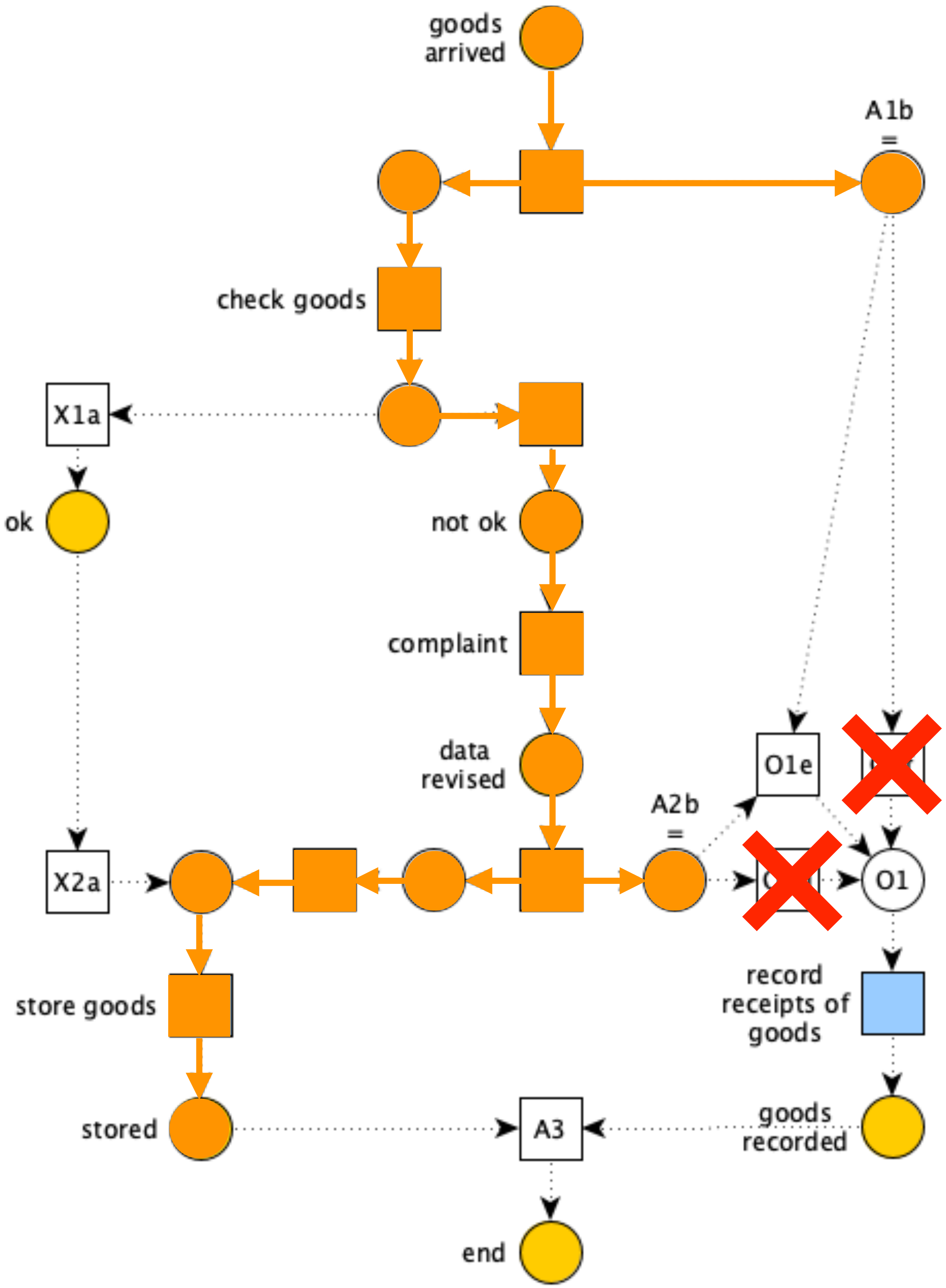
proper completion
is not guaranteed
(N* unbounded)

Soundness analysis



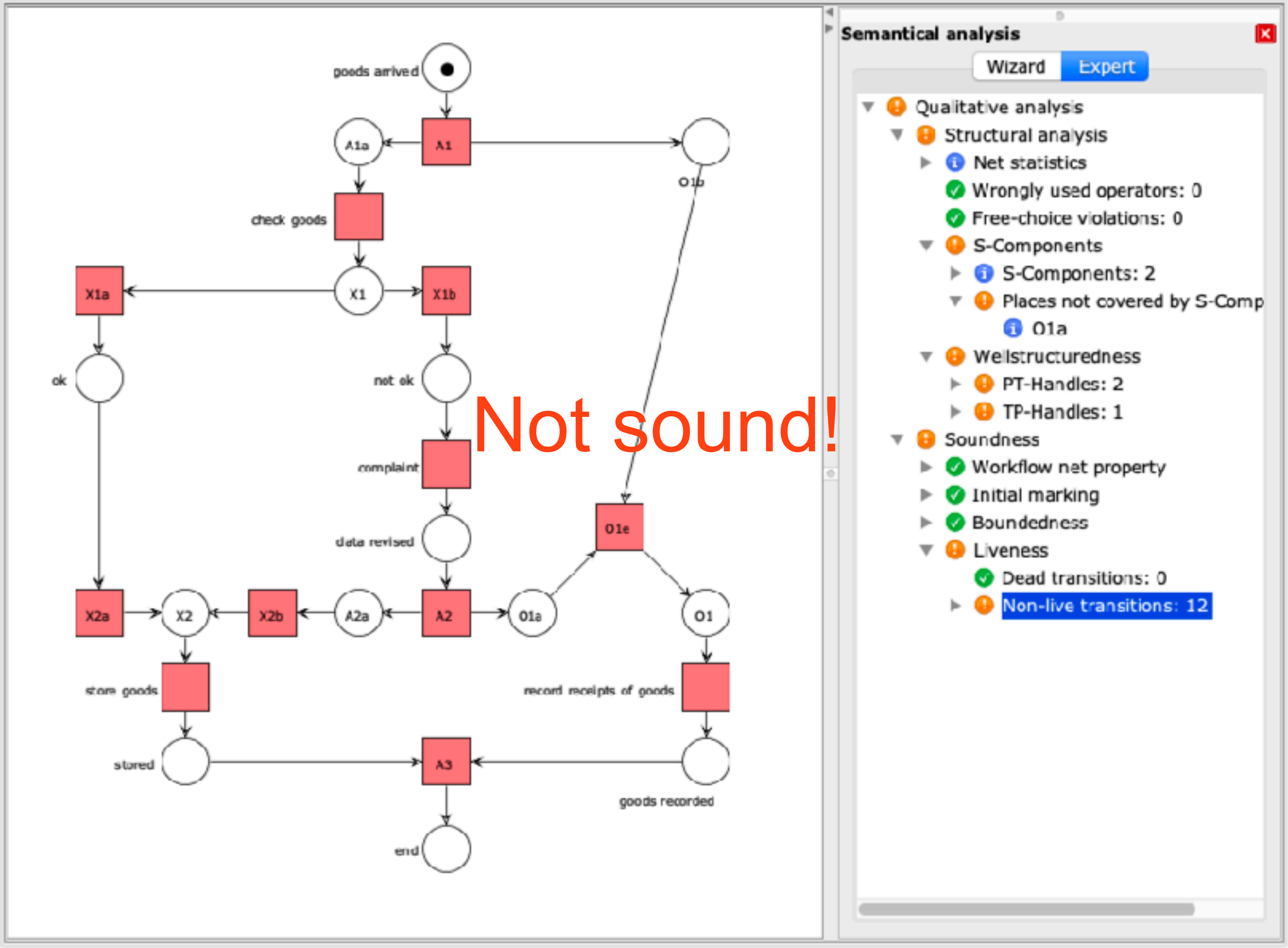
Can we repair the model?

Soundness analysis



AND join instead of OR join?

Soundness analysis

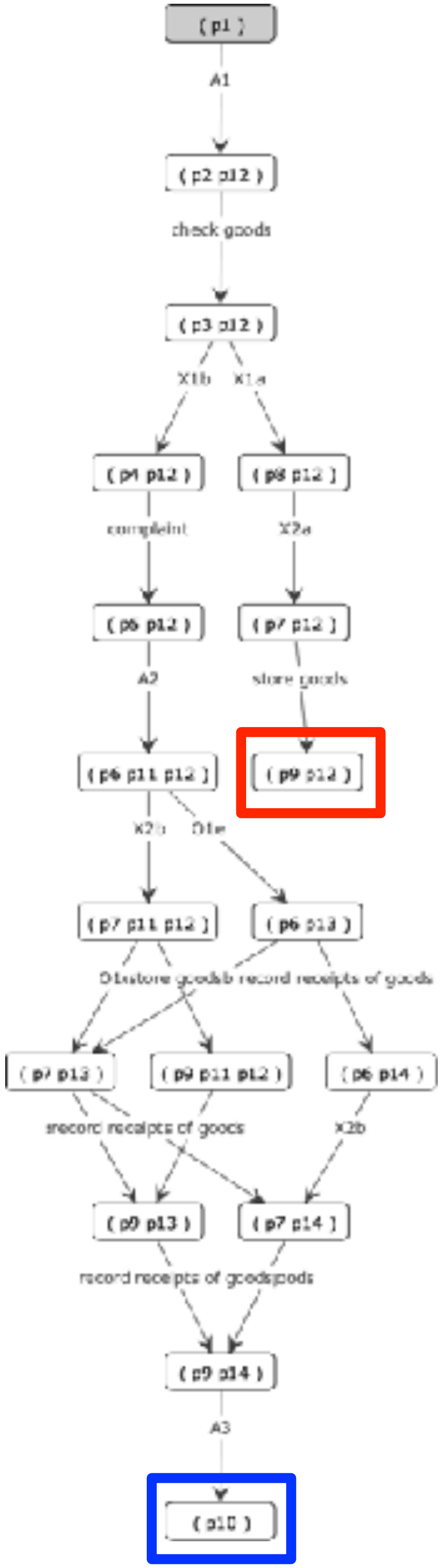


Not sound!

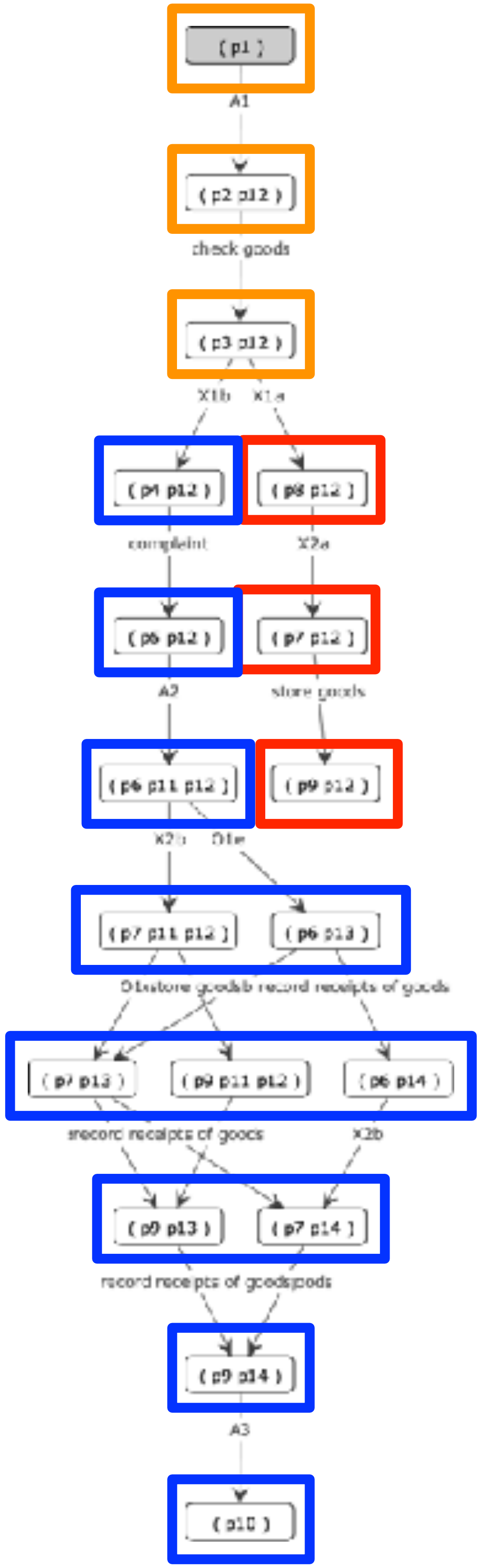
Semantical analysis

- ! Qualitative analysis
  - ! Structural analysis
    - i Net statistics
    - ✓ Wrongly used operators: 0
    - ✓ Free-choice violations: 0
    - ! S-Components
      - i S-Components: 2
      - ! Places not covered by S-Comp
        - i O1a
    - ! Wellstructuredness
      - ! PT-Handles: 2
      - ! TP-Handles: 1
  - ! Soundness
    - ✓ Workflow net property
    - ✓ Initial marking
    - ✓ Boundedness
    - ! Liveness
      - ✓ Dead transitions: 0
      - ! Non-live transitions: 12

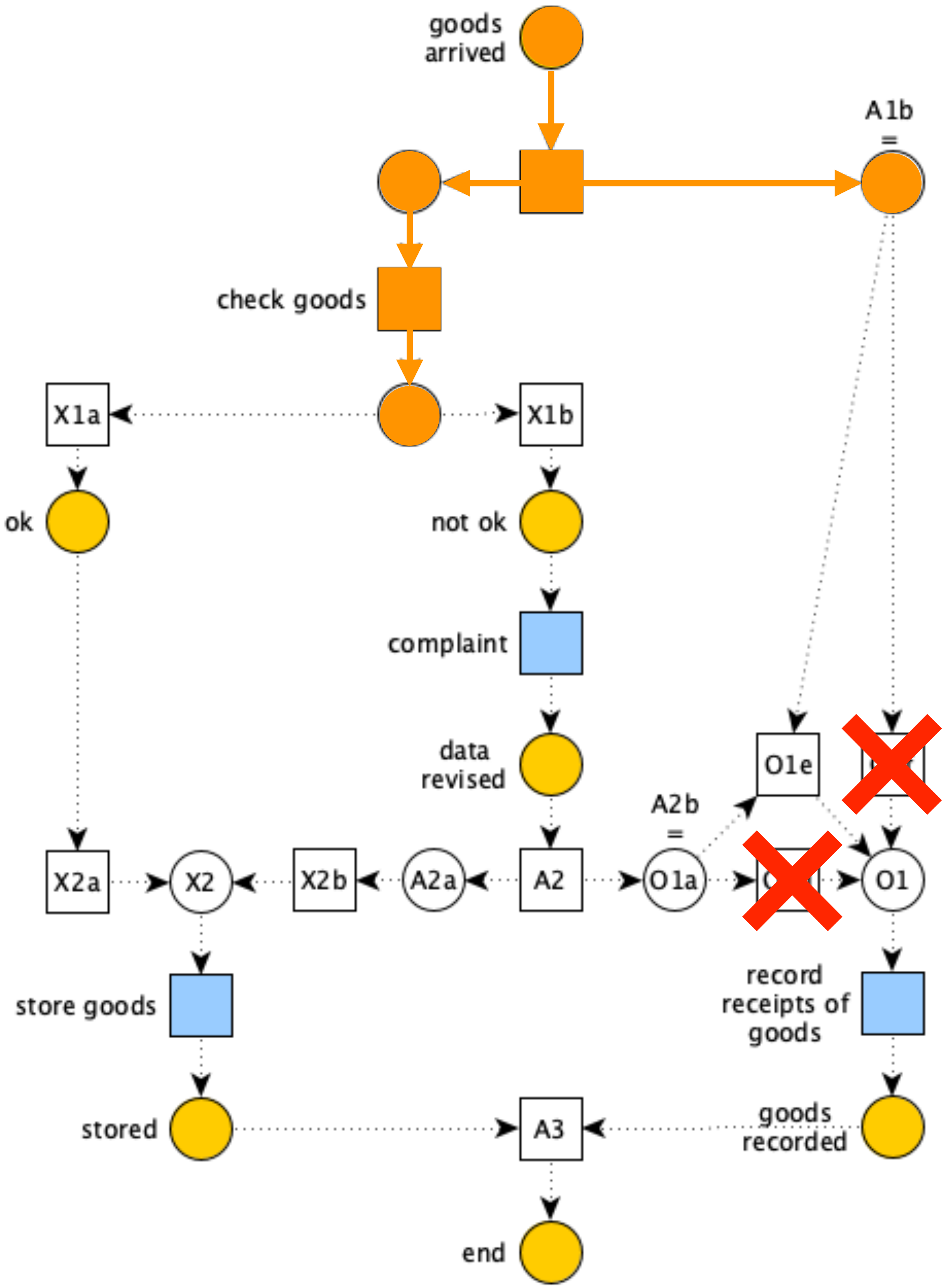
Soundness analysis



Soundness analysis



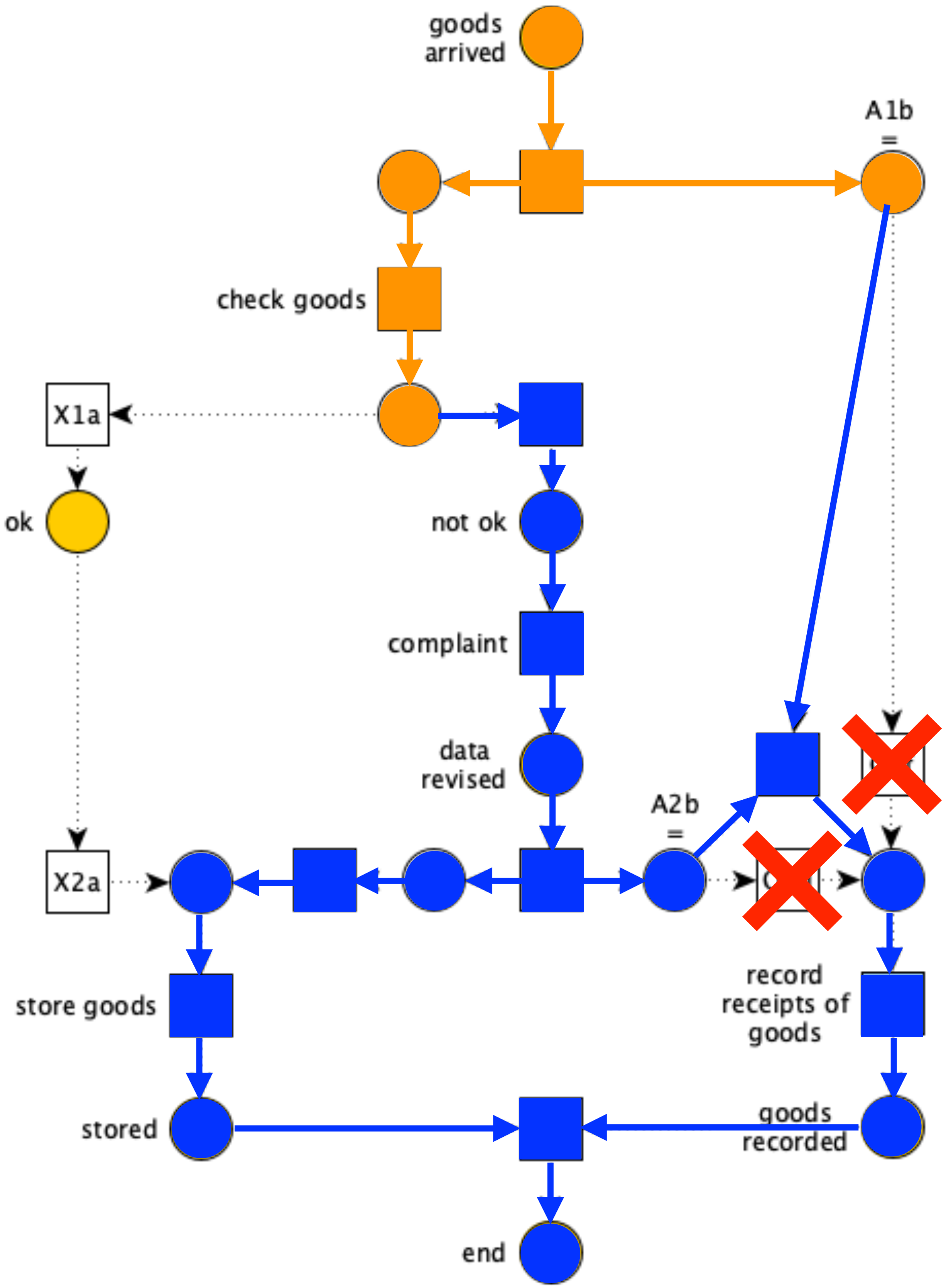
Soundness analysis



the right thing to do would be to fire X1b

AND join instead of OR join?

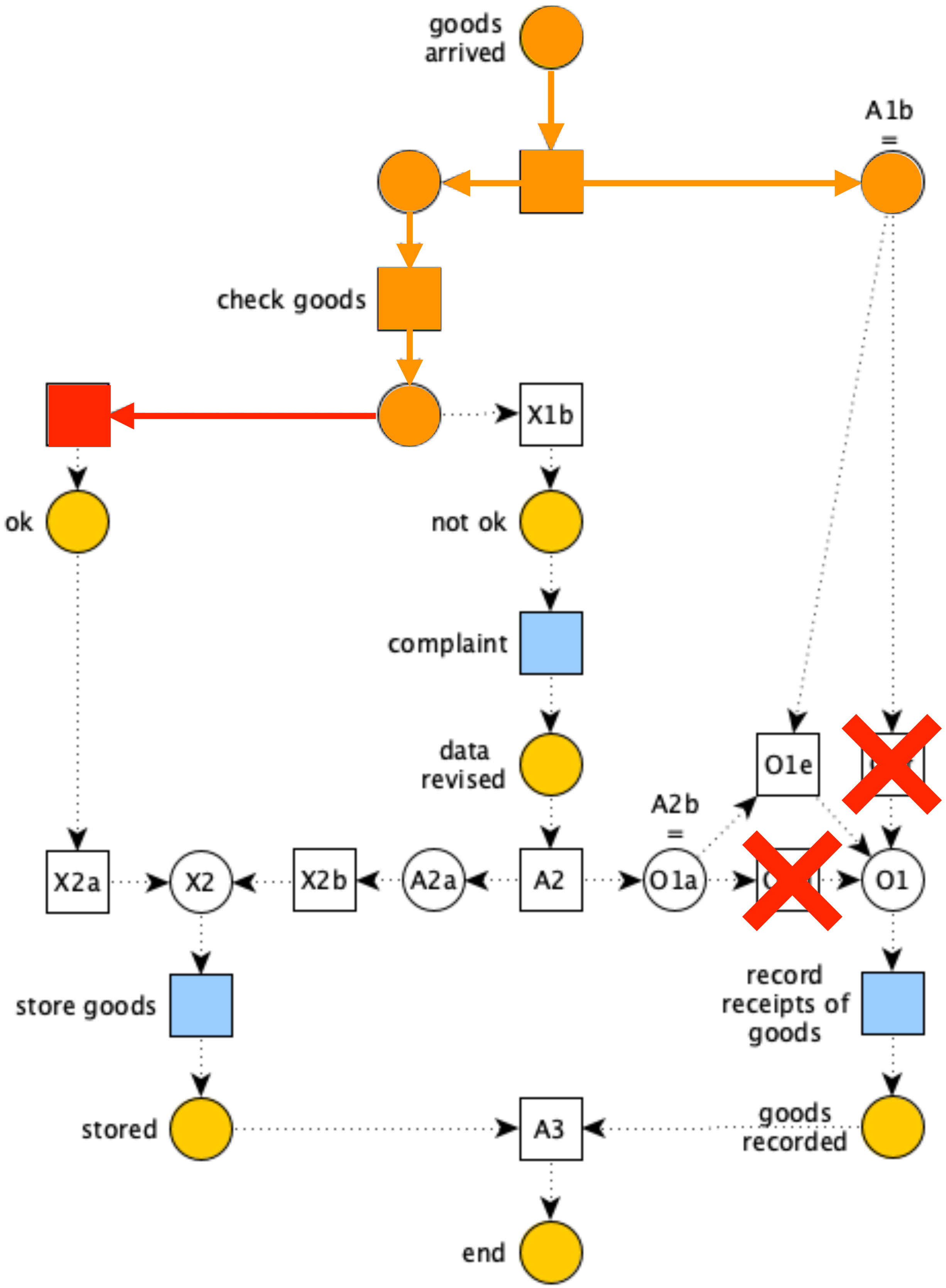
Soundness analysis



the right thing to do would be to fire X1b

AND join instead of OR join?

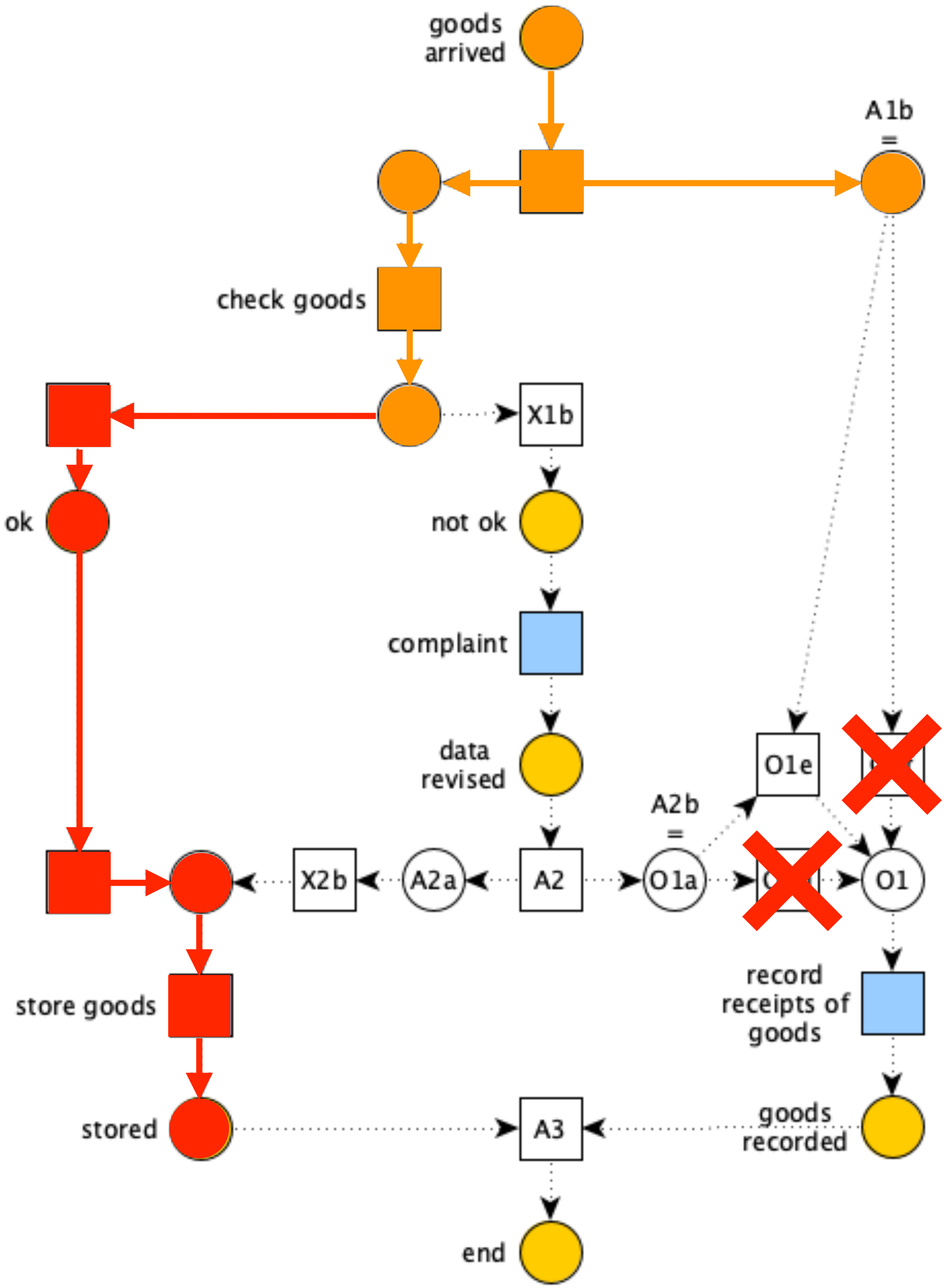
Soundness analysis



but X1a
is enabled as well

AND join instead of OR join?

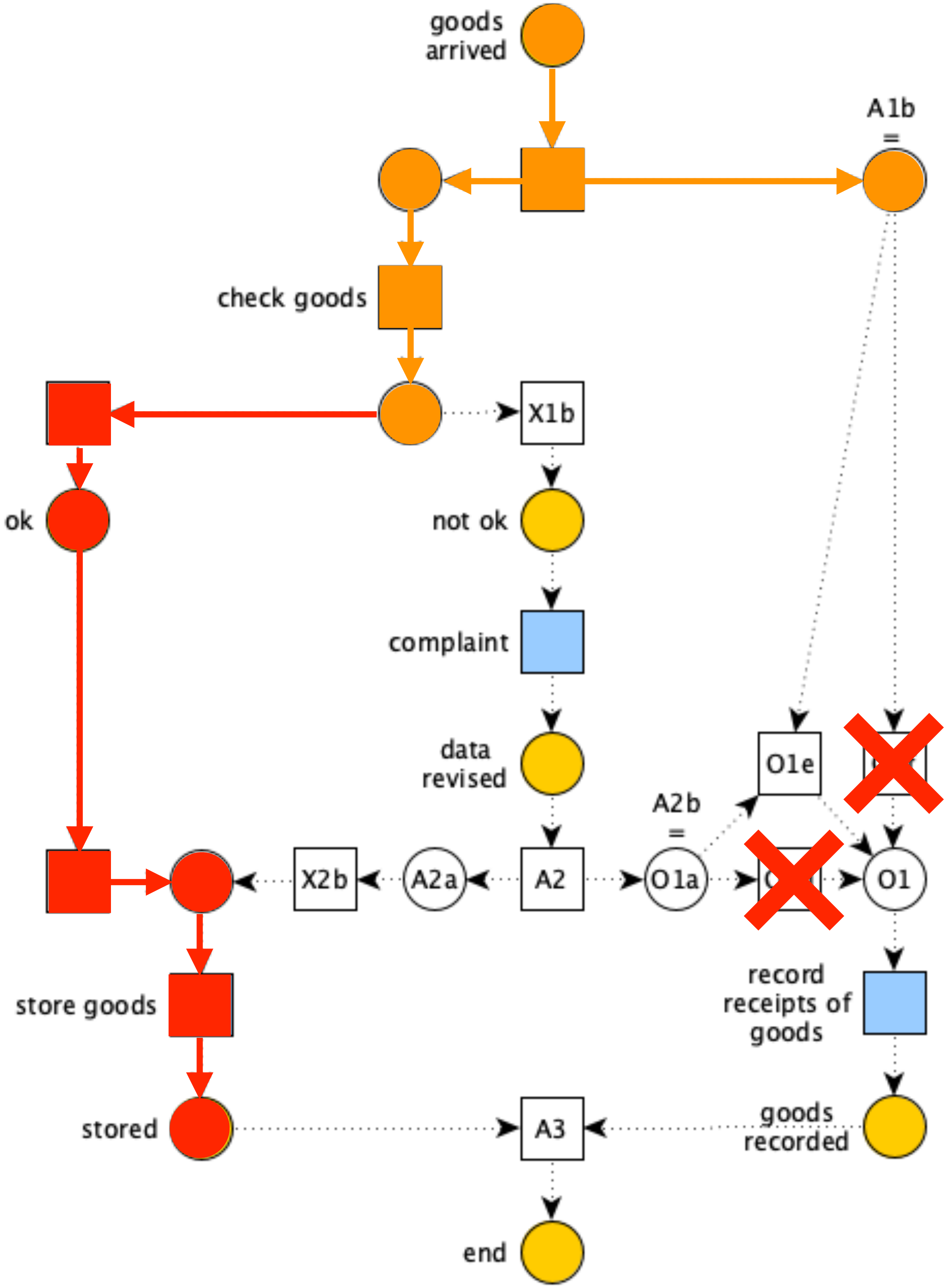
Soundness analysis



AND join instead of OR join?

possible deadlock!
option to complete is not guaranteed
(N* non-live)

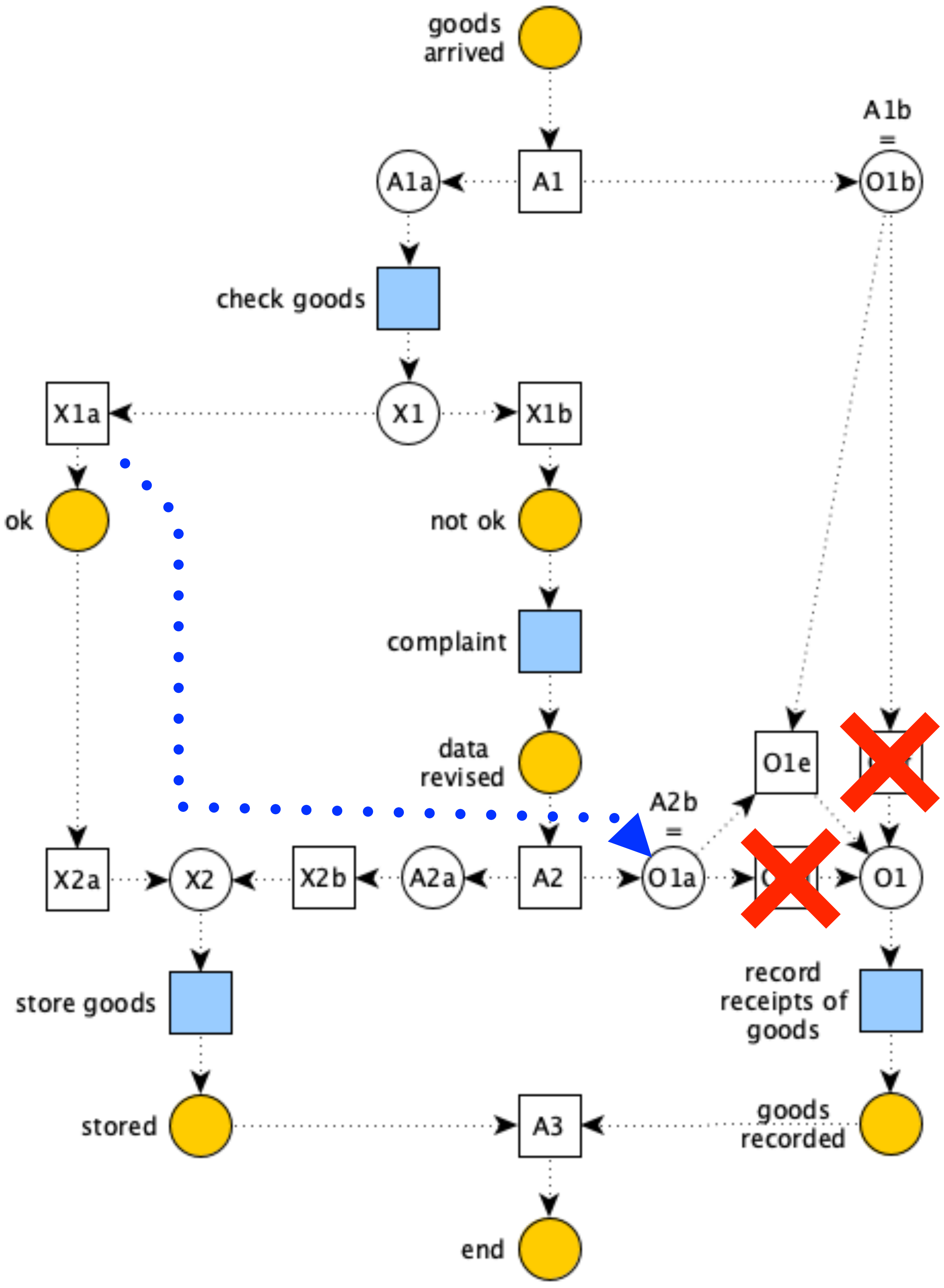
Soundness analysis



we miss a token in O1a

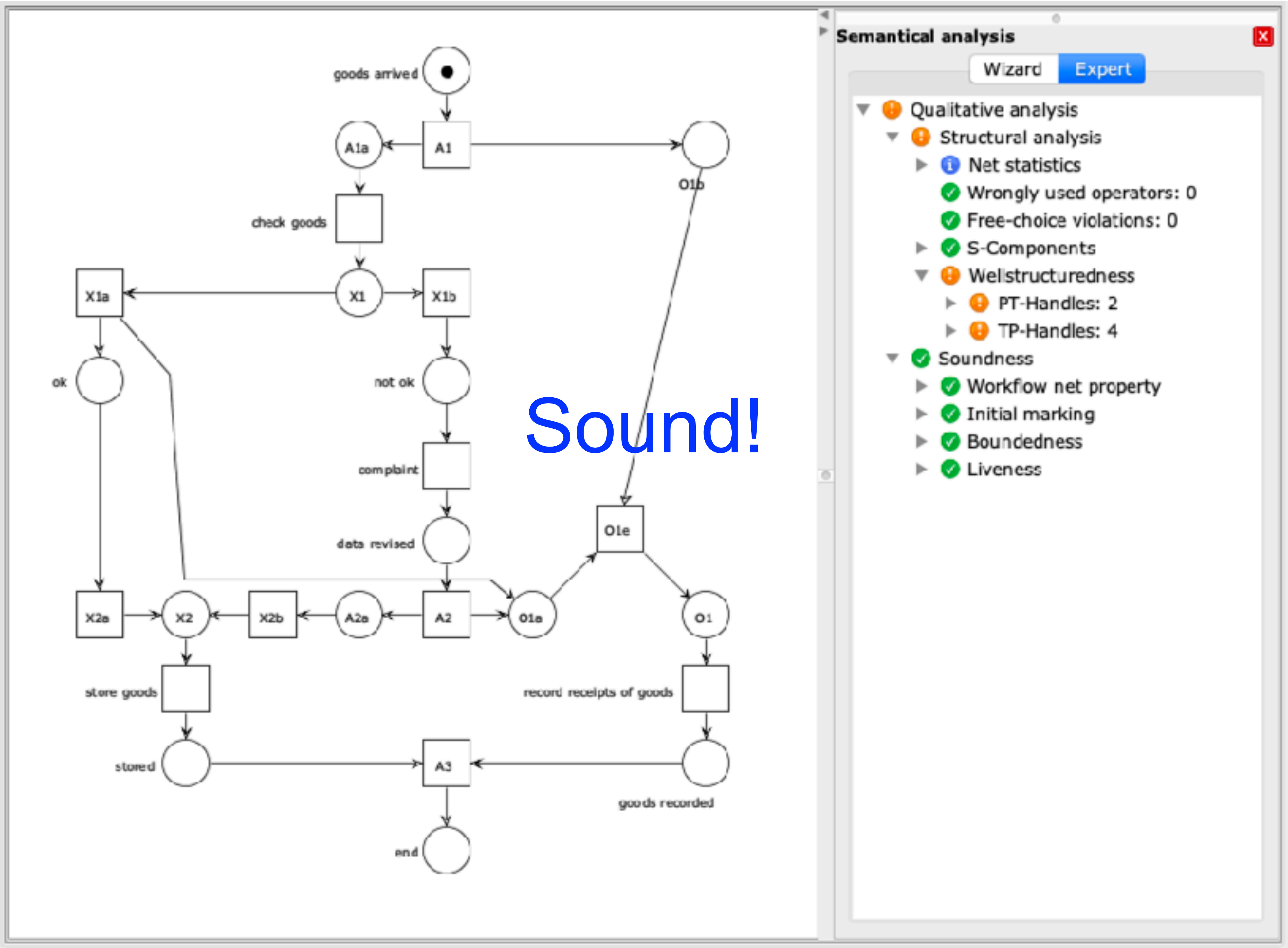
AND join instead of OR join + ad hoc flow?

Soundness analysis



AND join
instead of
OR join
+ ad hoc flow?

Soundness analysis

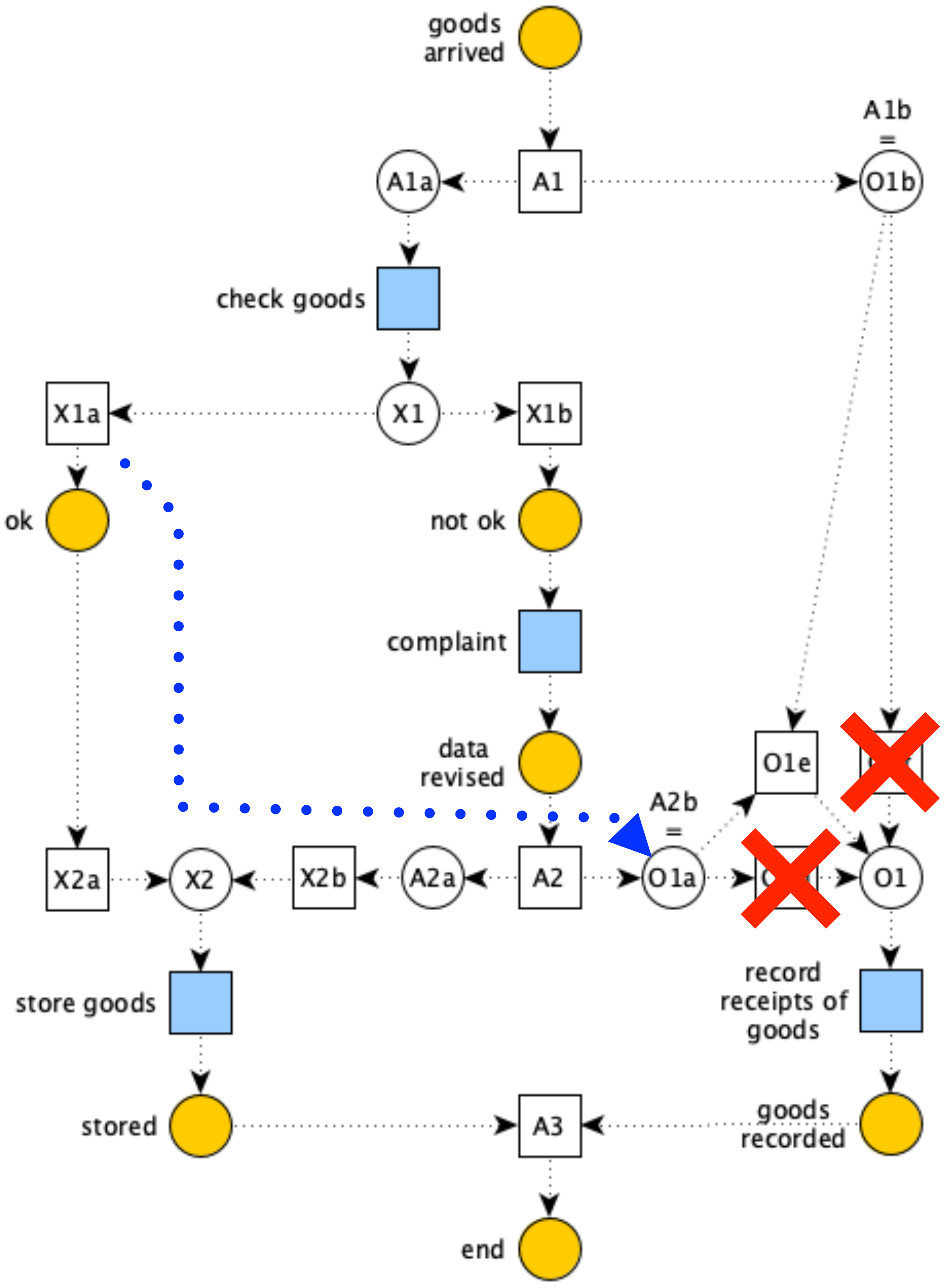


Sound!

Semantical analysis

- Qualitative analysis
  - Structural analysis
    - Net statistics
    - Wrongly used operators: 0
    - Free-choice violations: 0
    - S-Components
    - Wellstructuredness
      - PT-Handles: 2
      - TP-Handles: 4
  - Soundness
    - Workflow net property
    - Initial marking
    - Boundedness
    - Liveness

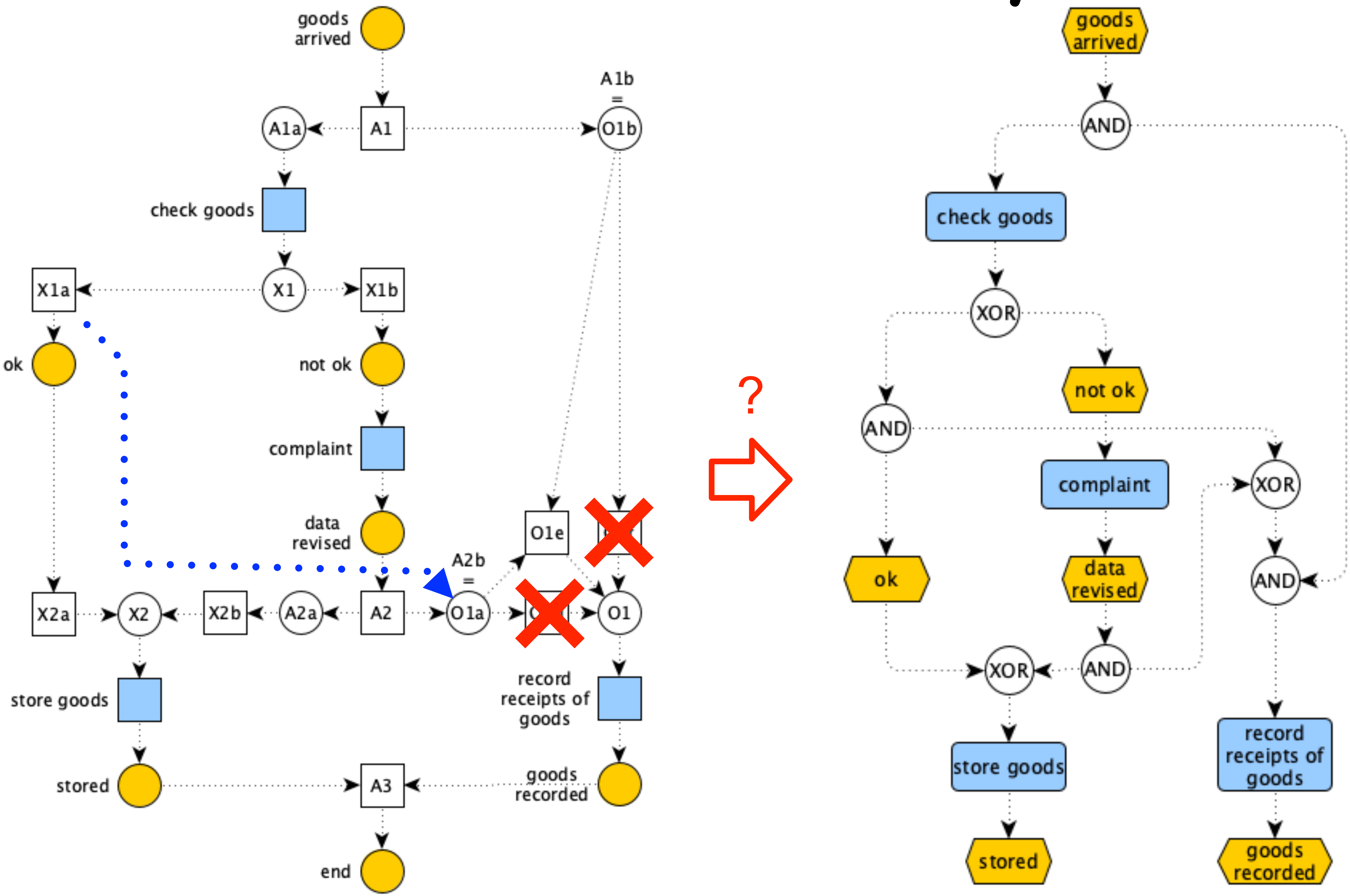
Soundness analysis



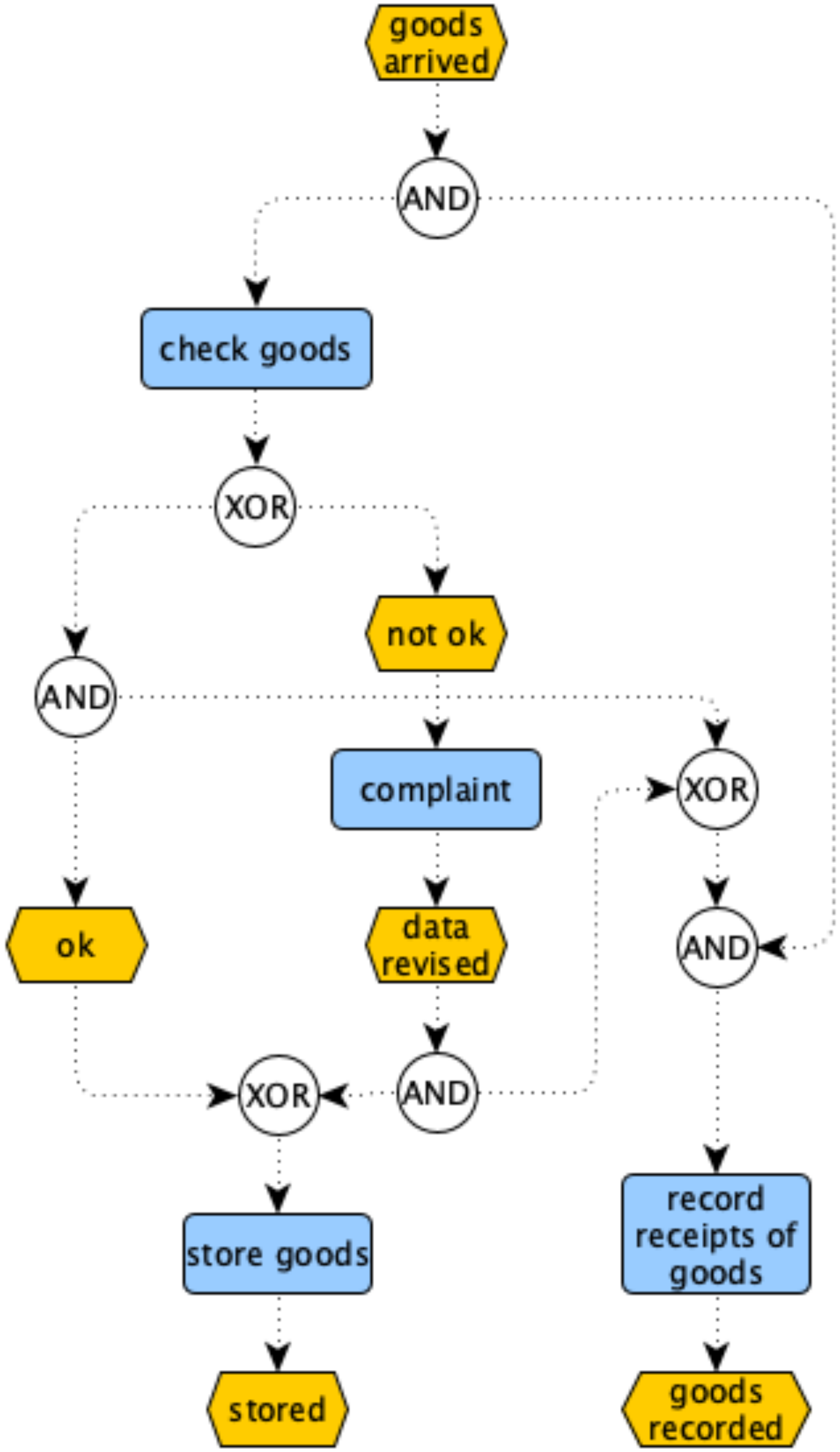
Sound, but…

we have repaired the wf net, not the original EPC diagram!

Soundness analysis



Soundness analysis



The diagram is now
more complex
and less readable
than the original one!

Are we sure that its translation
is the same sound wf net that
we have designed ad hoc?

Are we sure it is sound?

Need to restart the analysis!!

Relaxed Soundness (optional reading)

Problem

EPC is widely adopted
also at early stages of design

WF nets offer a useful tool

but

Soundness can be too demanding at early stages

(Un)sound behaviours

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

The language of the net
collects all and only
its sound behaviours

$$L(N) = \{\sigma \mid i \xrightarrow{\sigma} o\}$$

Execution paths leading to **unsound** behaviours
can be used to infer potential mistakes

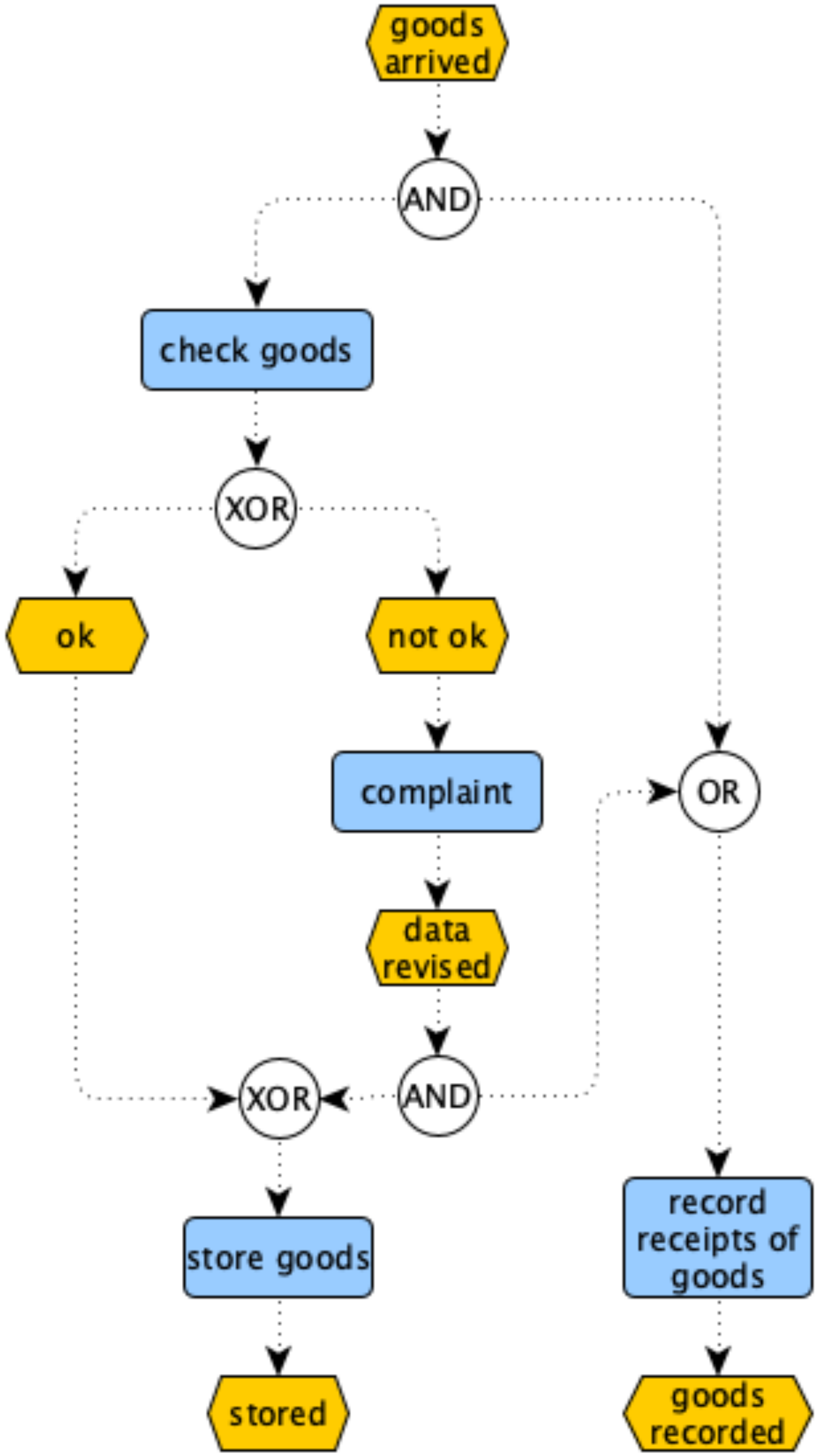
Relaxed soundness

If some unsound behaviour is possible
but any transition can take part to one sound execution,
then the process is called **relaxed sound**

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
(i.e. it appears in the language of the net)

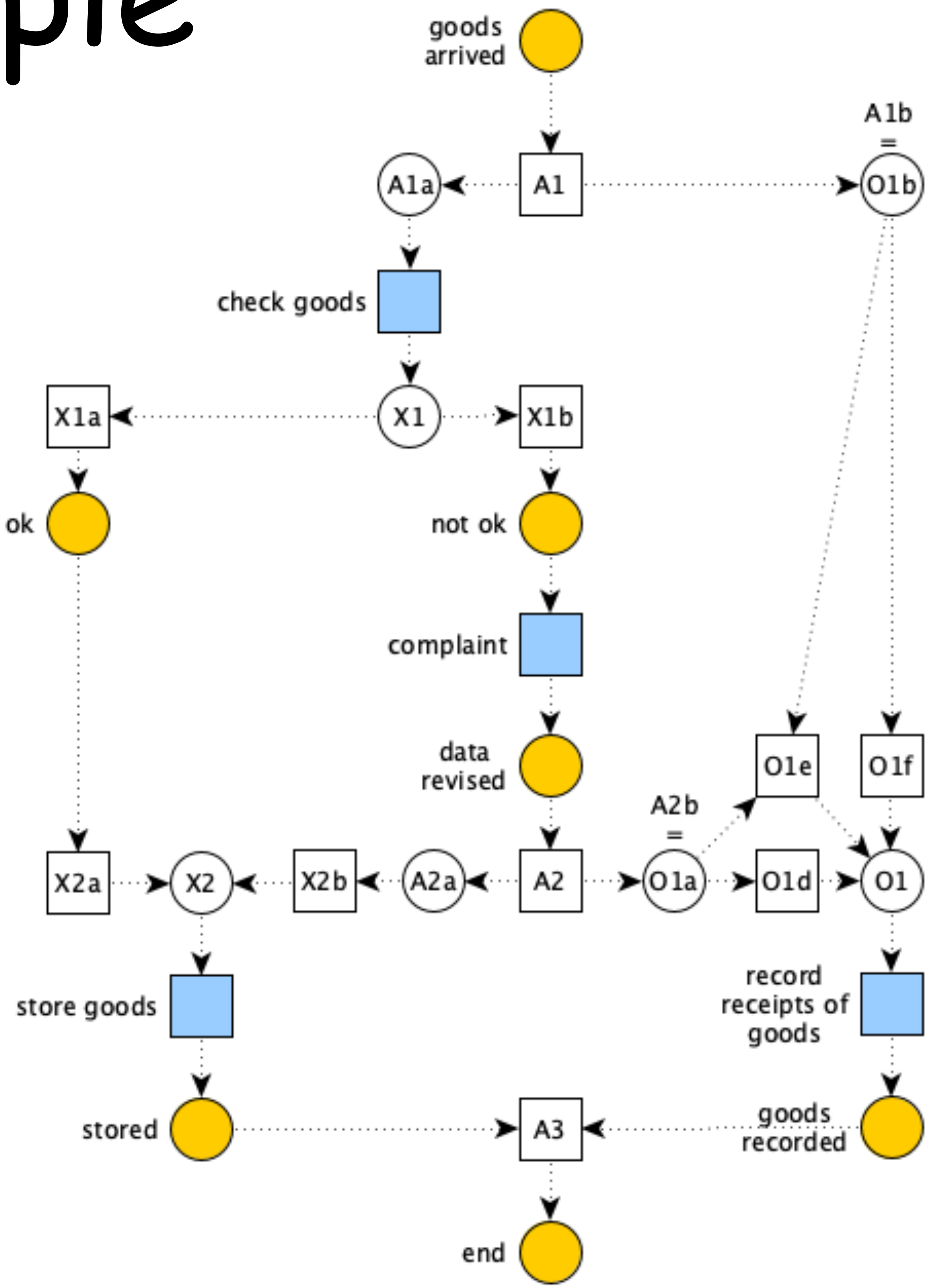
$$\forall t \in T.\ \exists \sigma \in L(N).\ \vec{\sigma}(t) > 0$$

Example

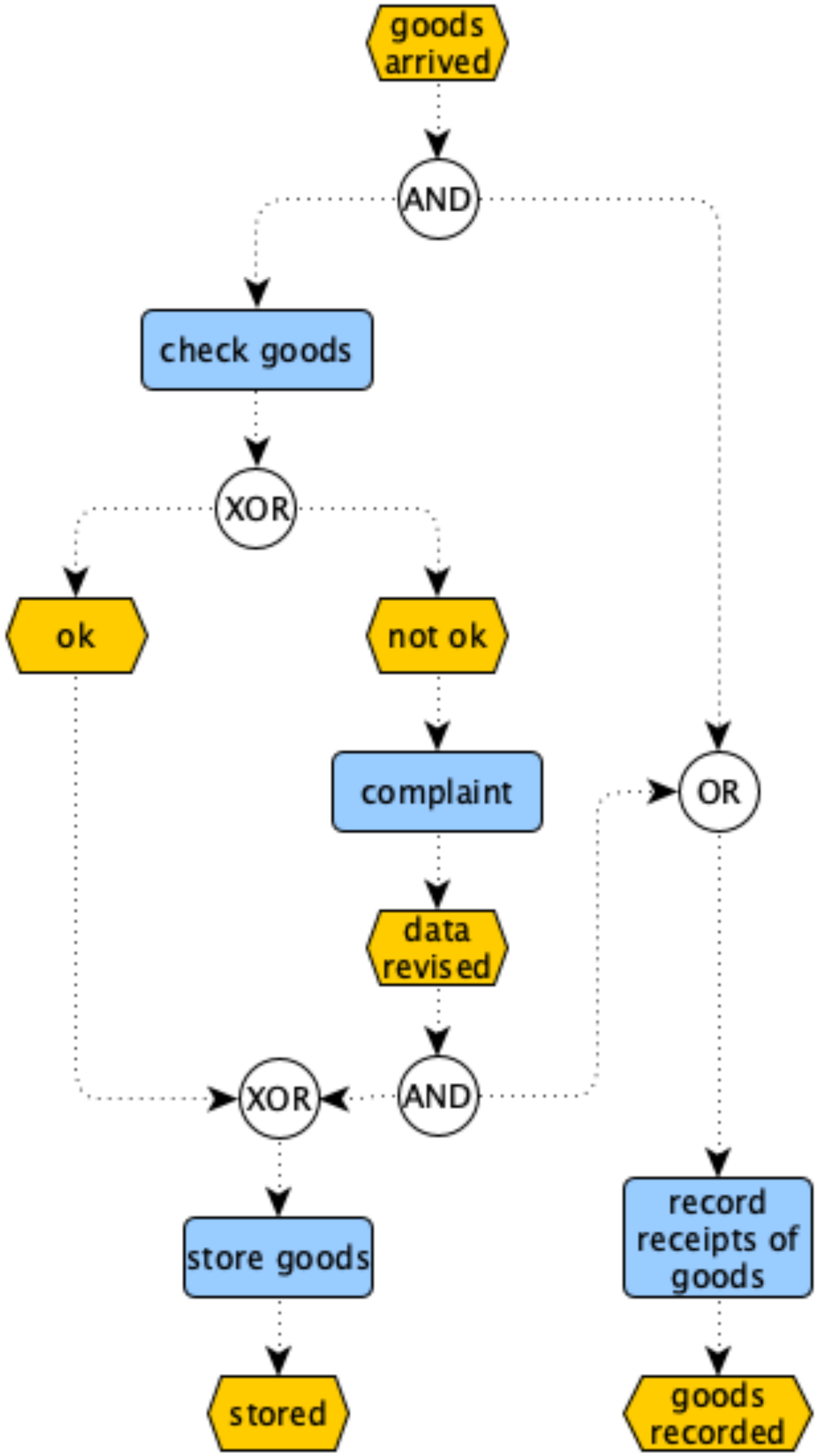


Relaxed sound?

Steps 1+2+3



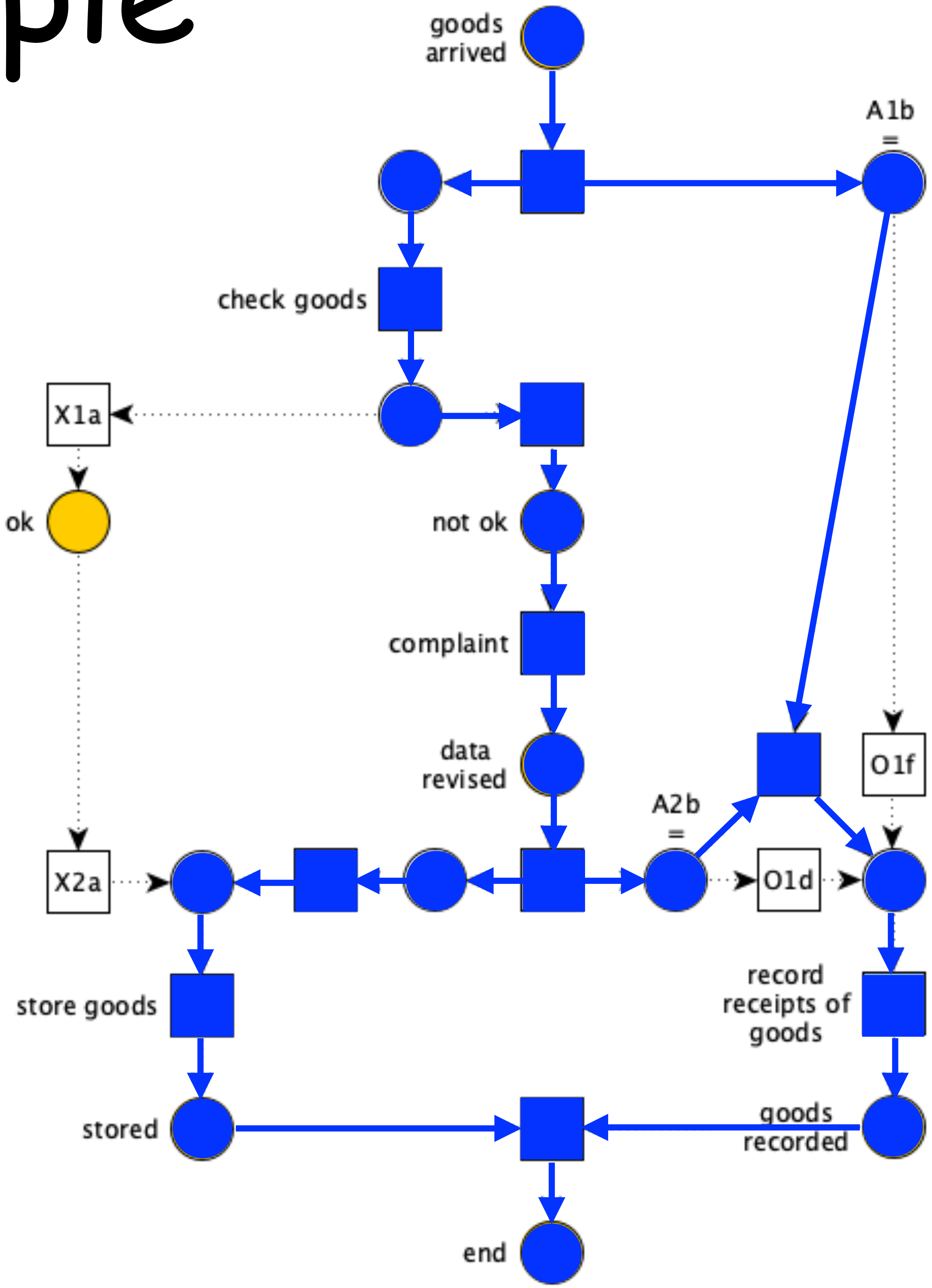
Example



Relaxed sound?

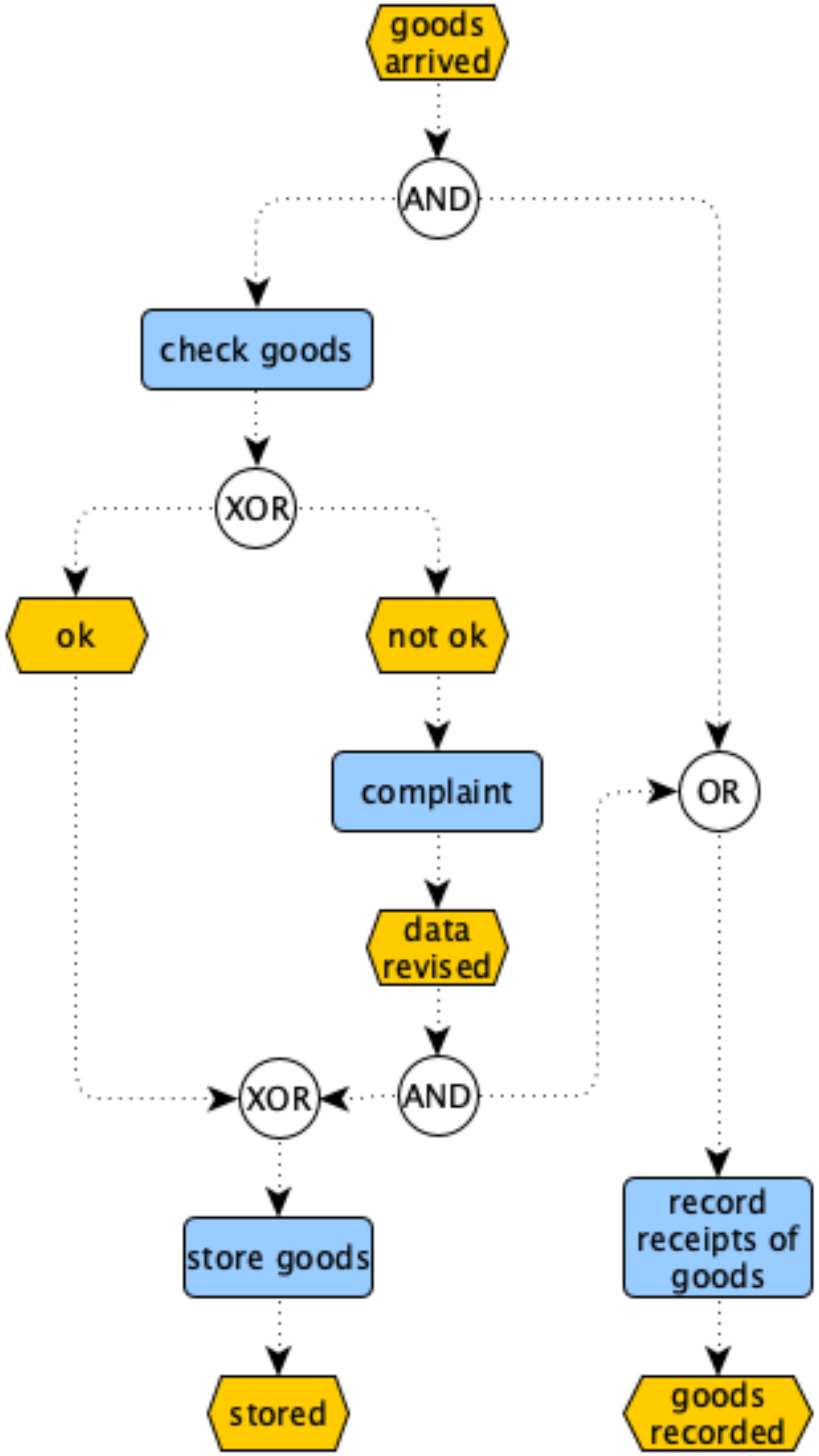
Steps 1+2+3

a sound execution



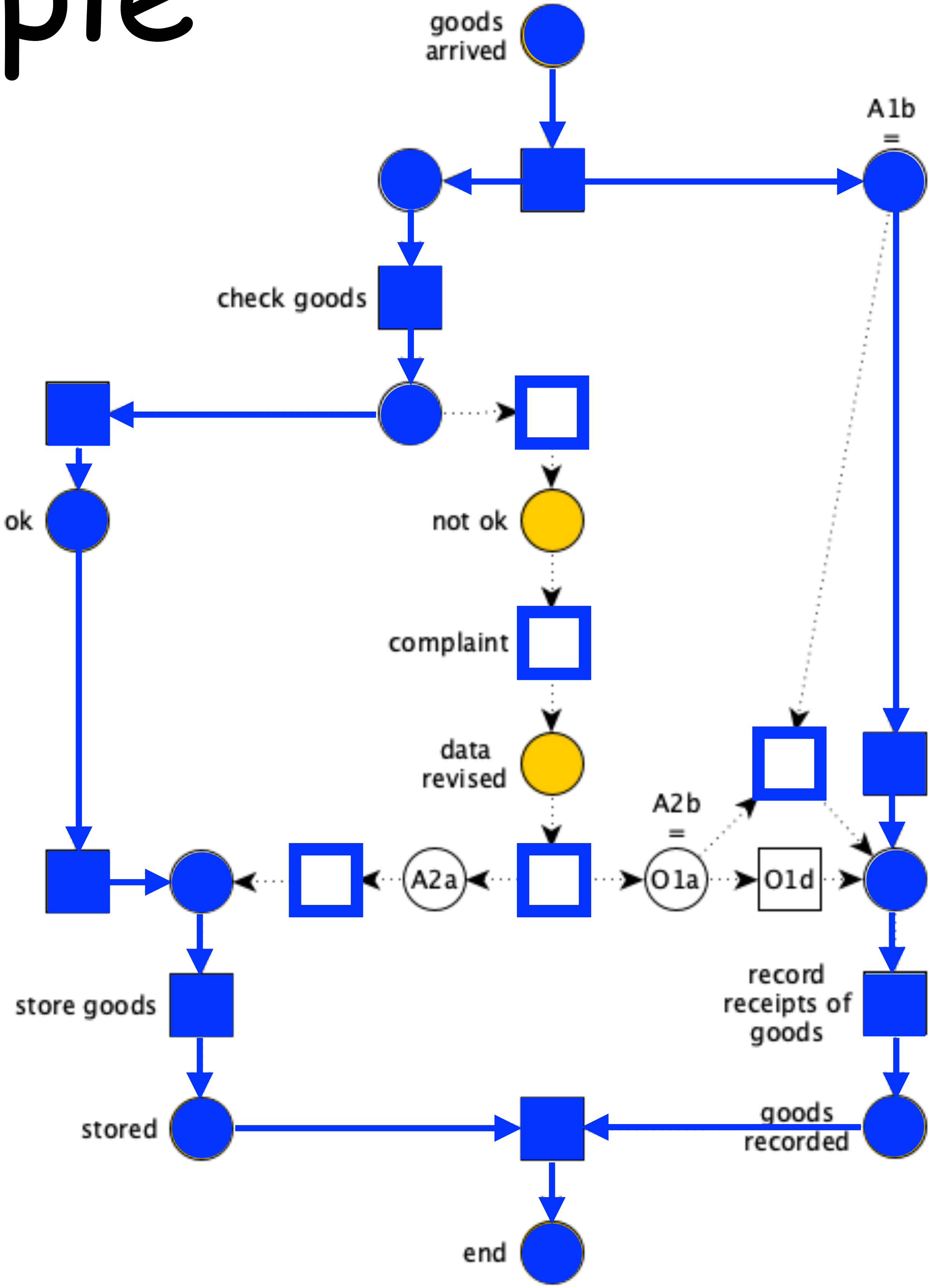
Example

another sound execution

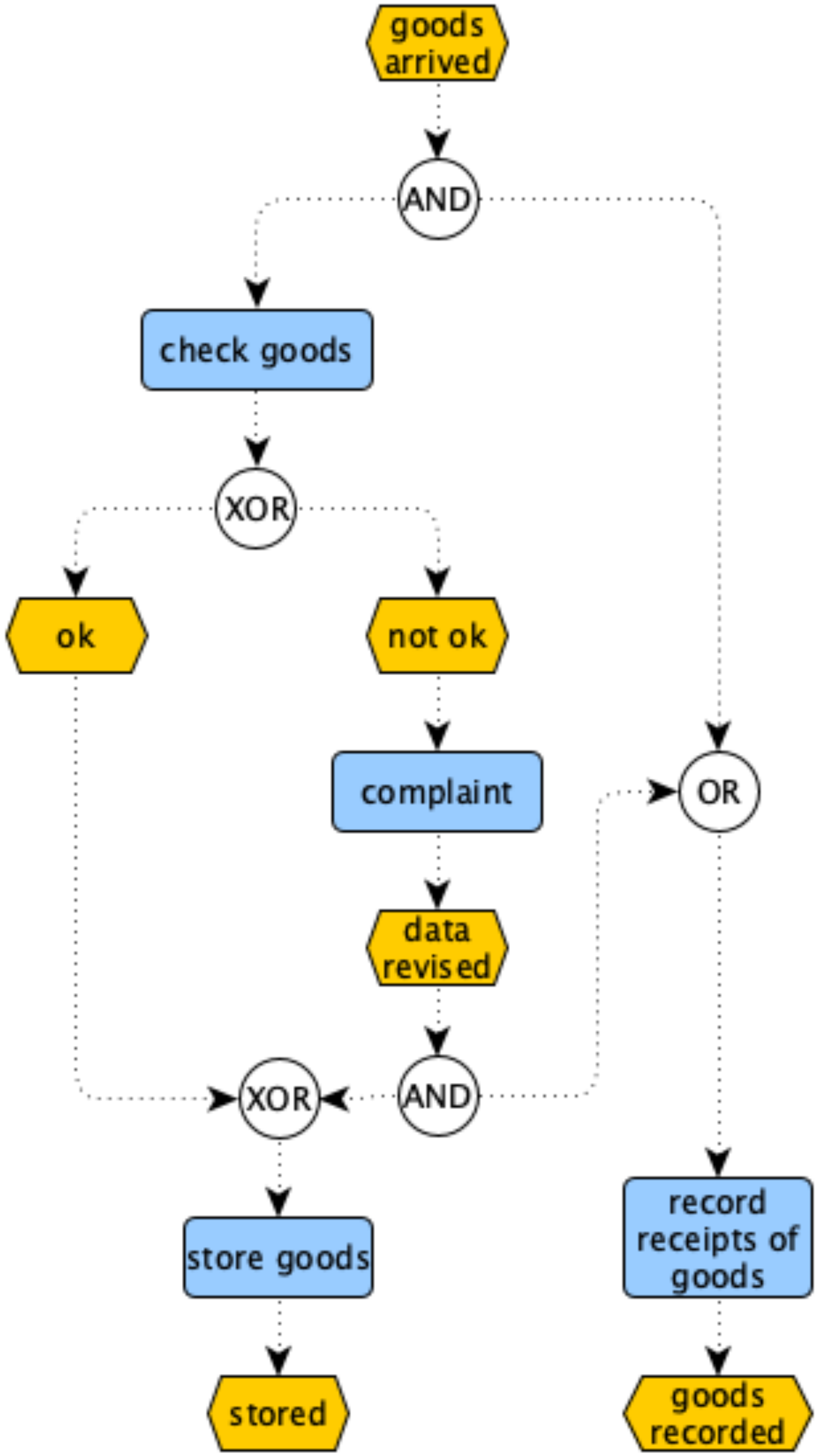


Relaxed sound?

Steps 1+2+3



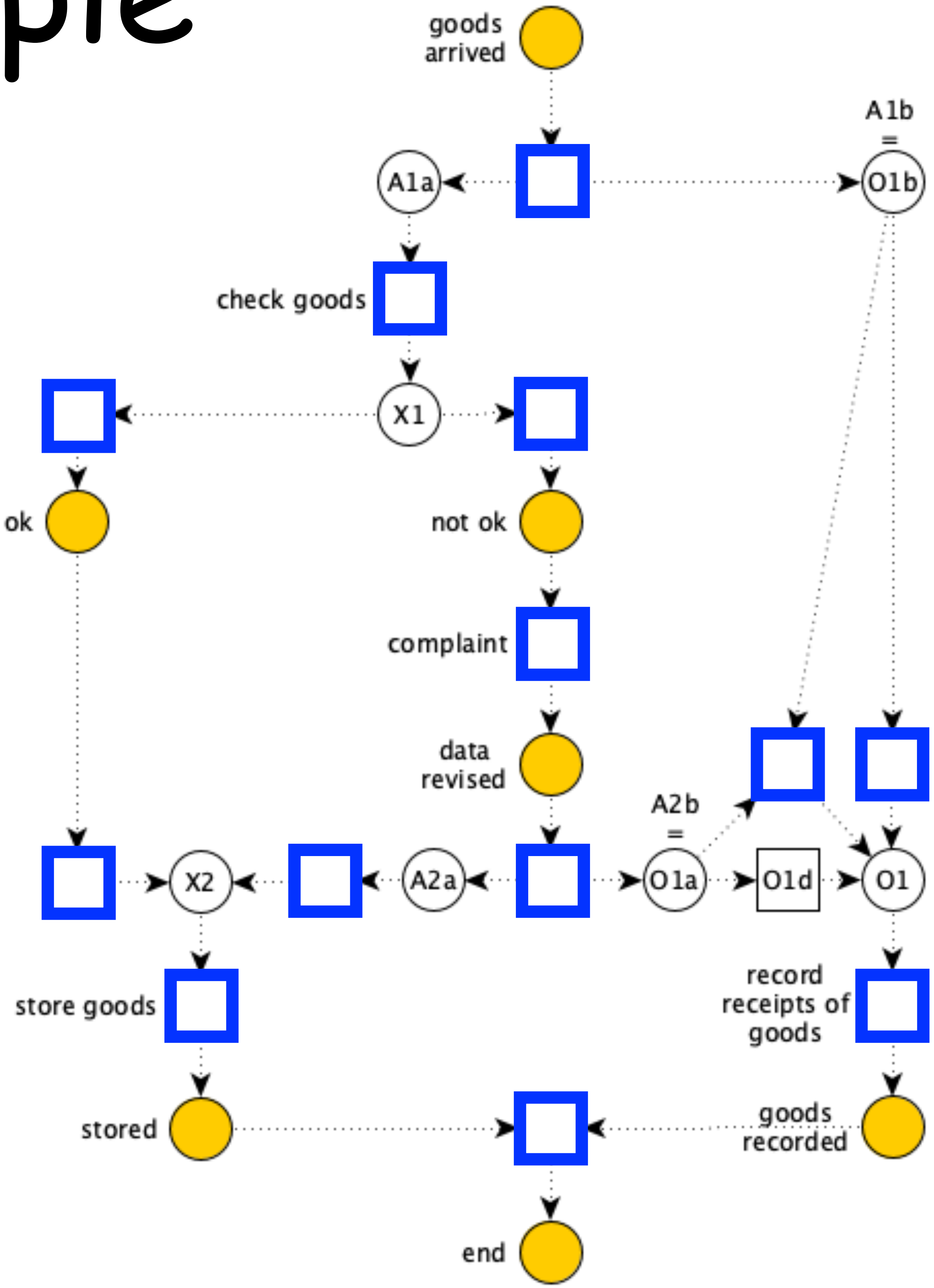
Example



Relaxed sound?

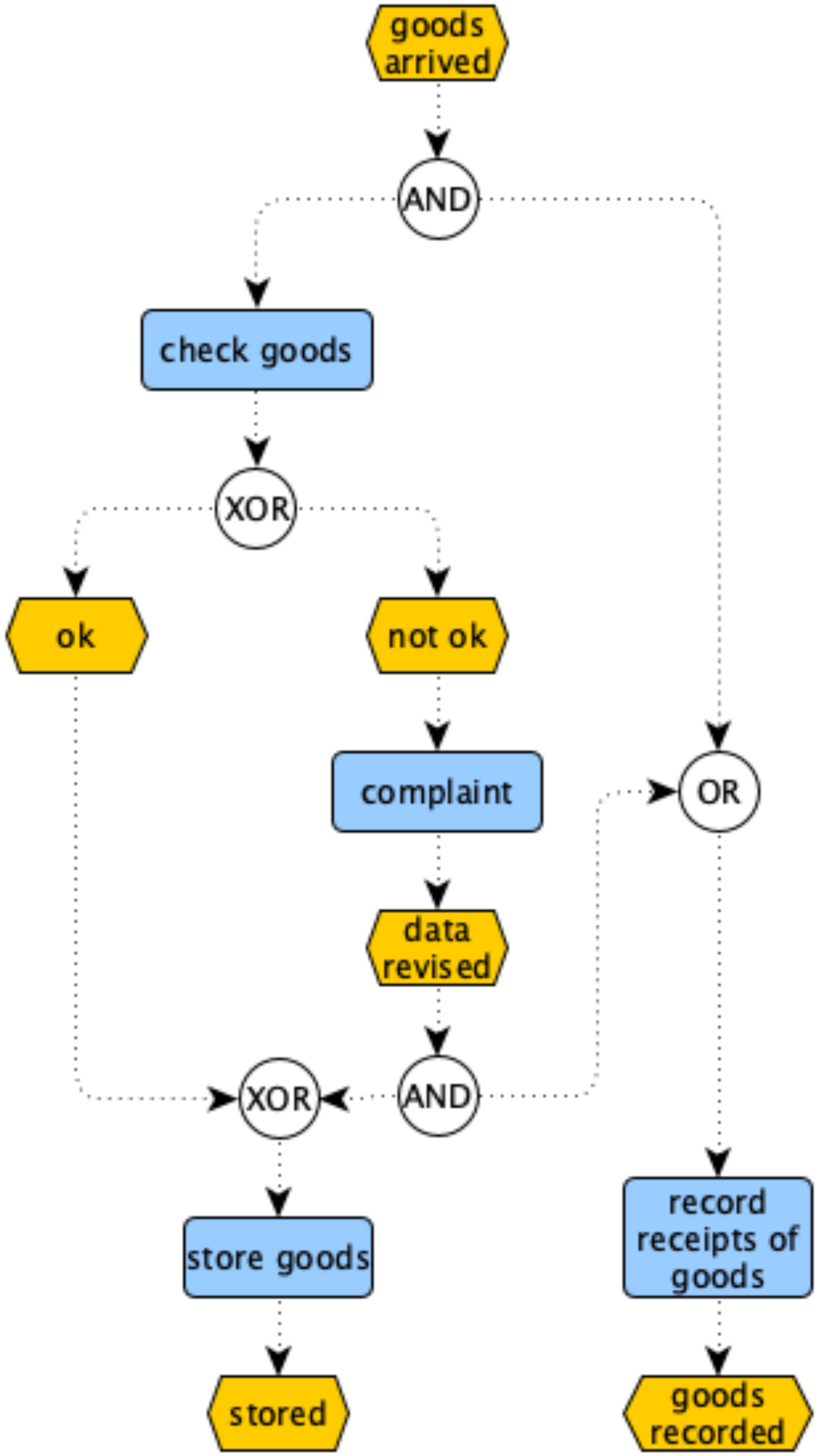
Steps 1+2+3

tasks involved in some sound execution



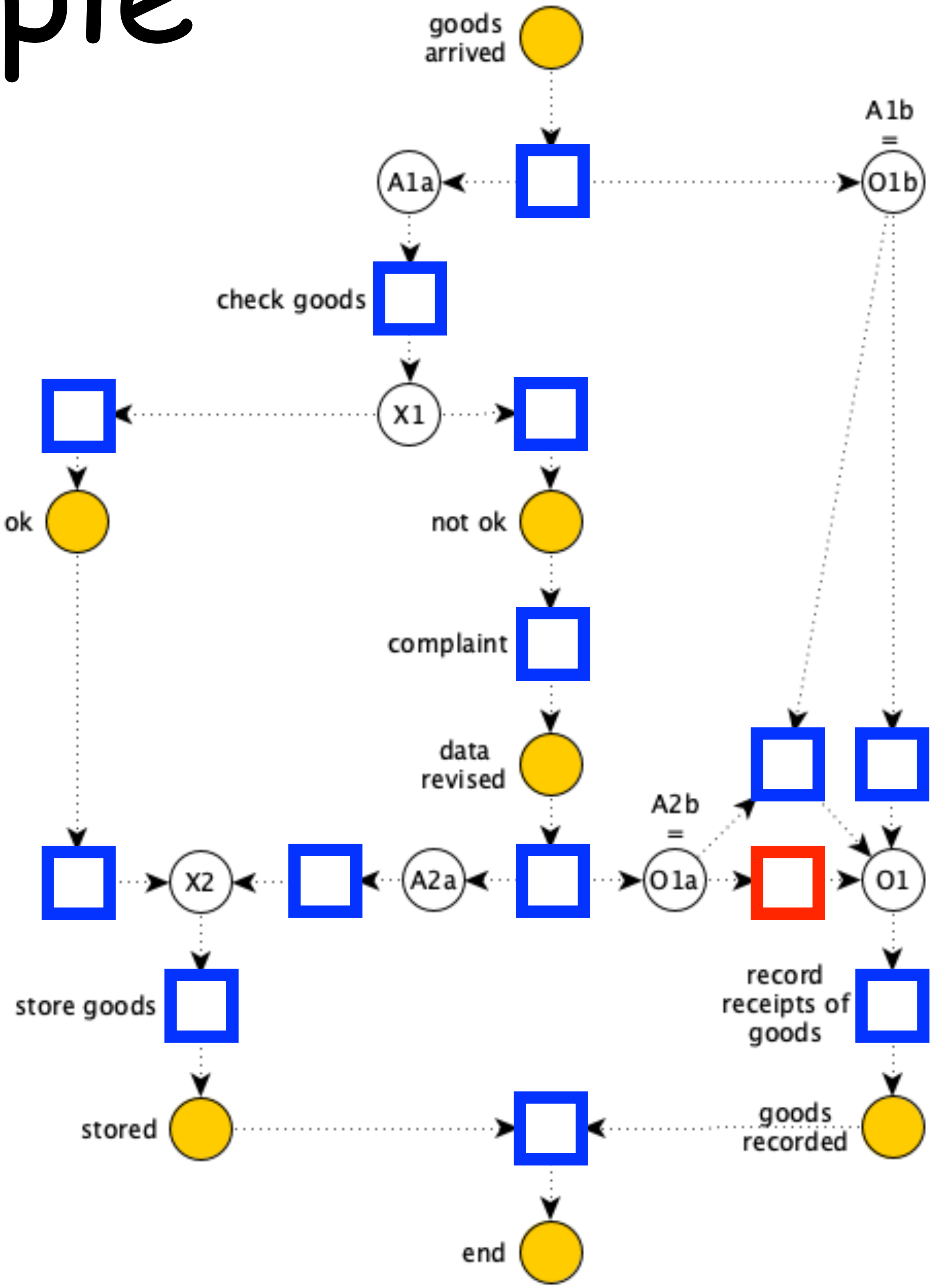
Example

one task not involved in some sound execution



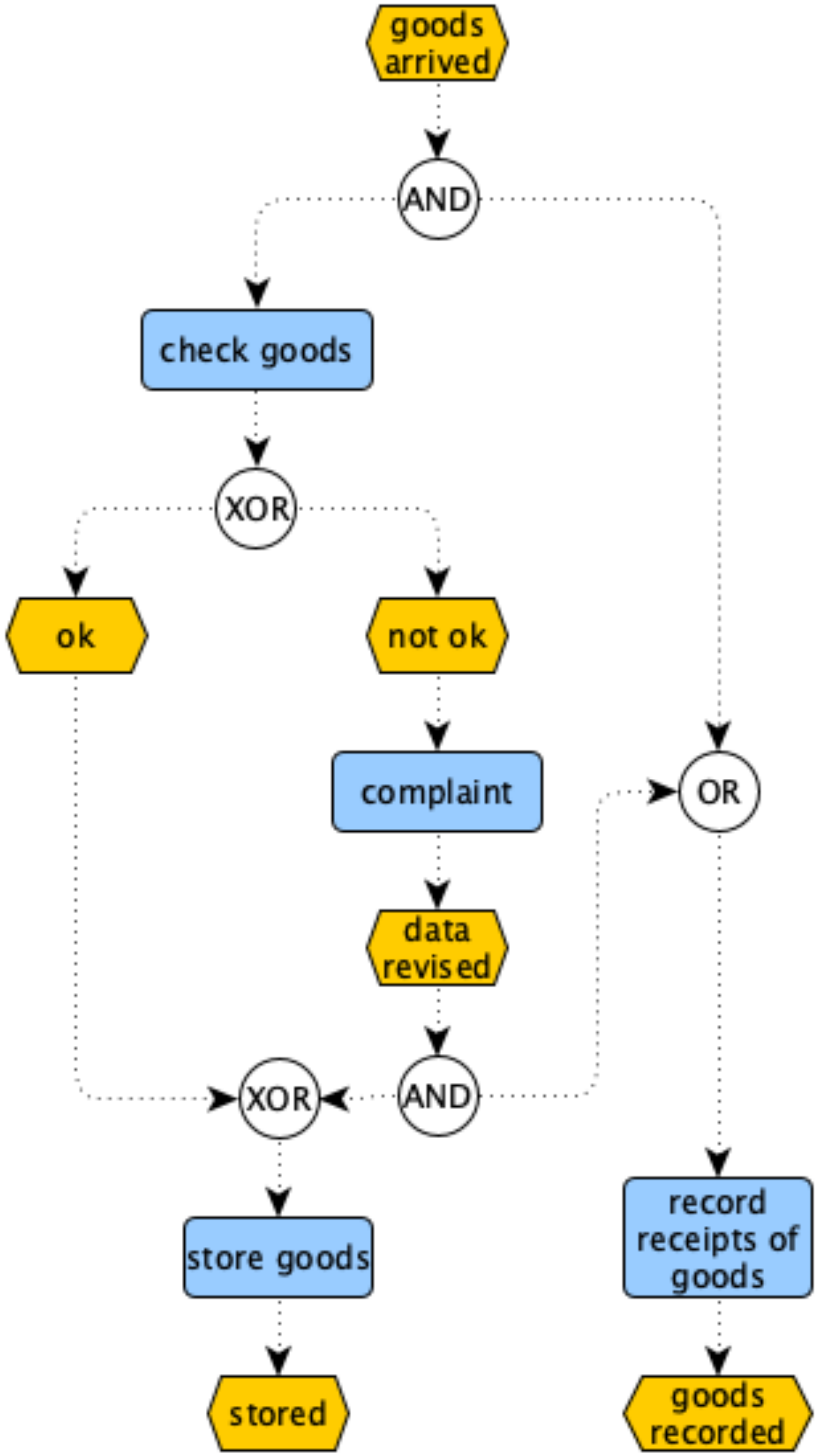
Not relaxed sound as a net!

Steps 1+2+3



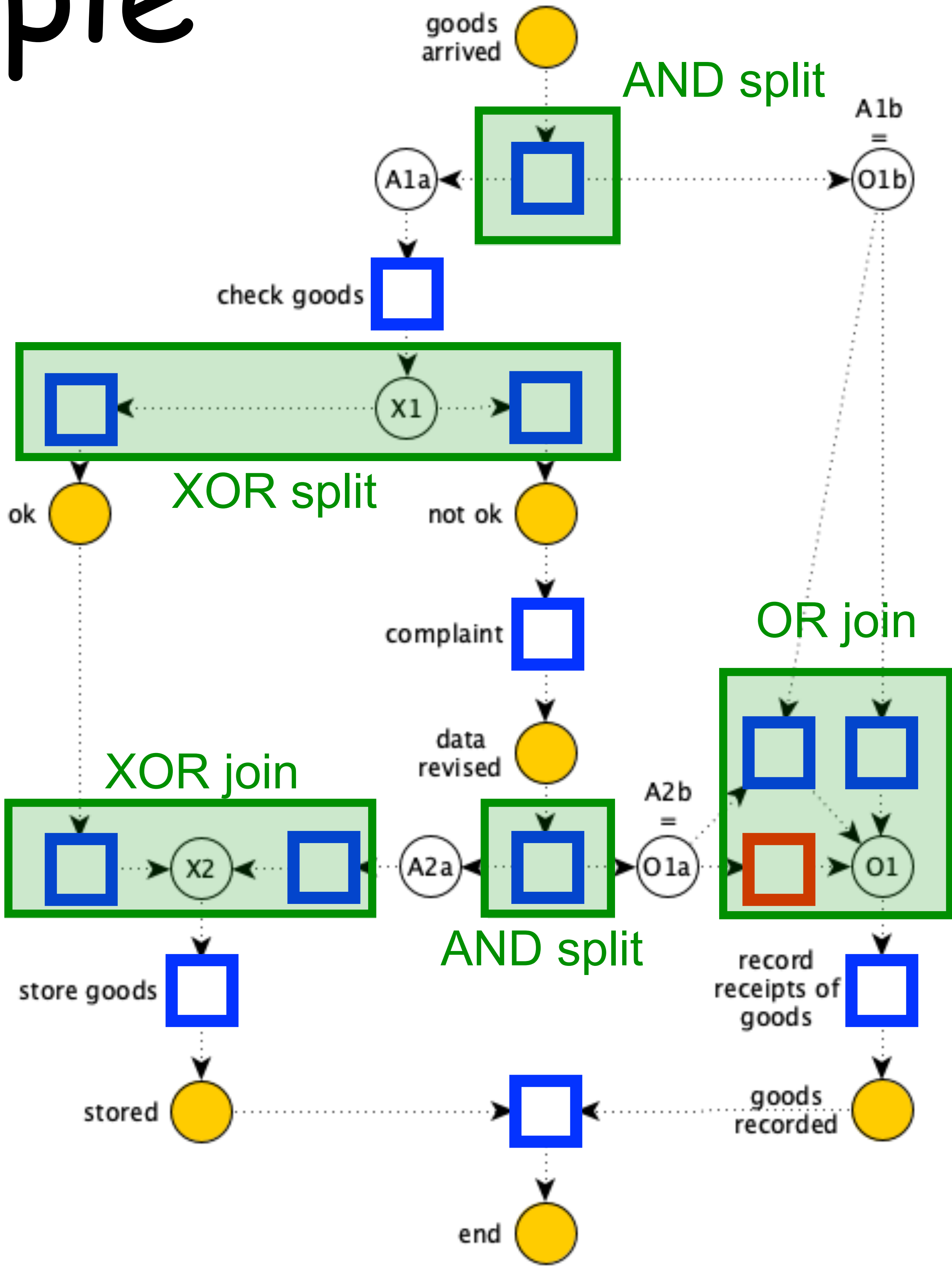
Example

all EPC nodes involved in some sound execution



Relaxed sound as EPC!

Steps 1+2+3



Relaxed soundness?

If the WF net is **not relaxed sound** there are transitions that are not involved in sound executions (not included in a firing sequence of L(N))

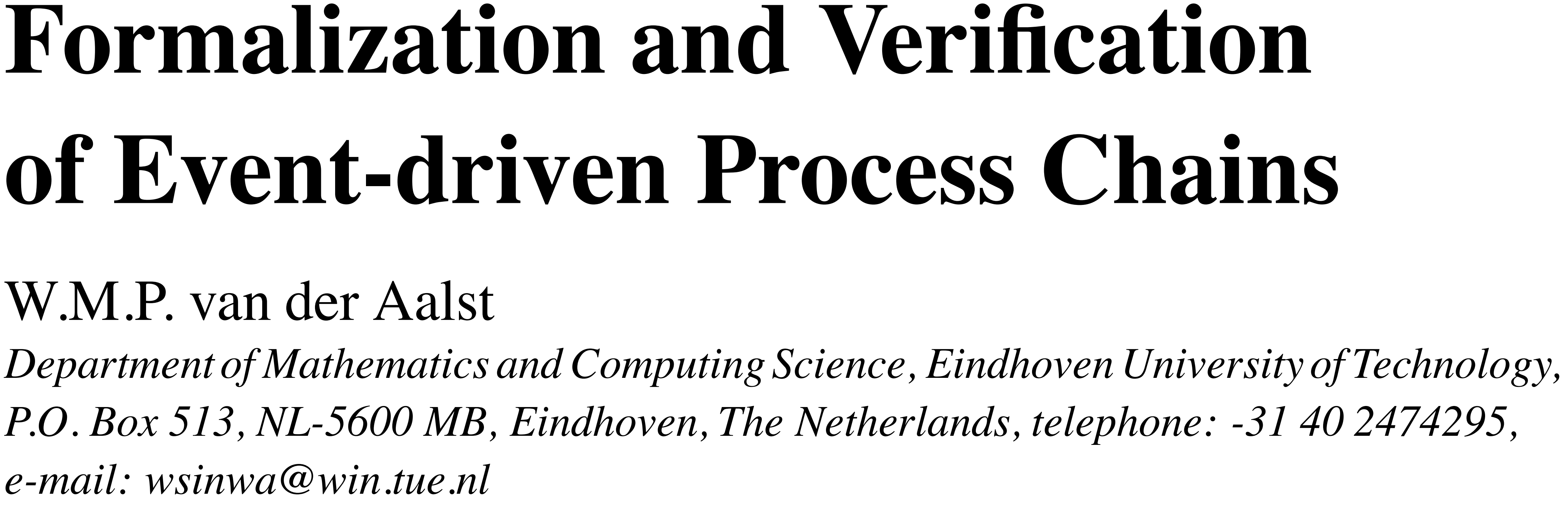
Their EPC counterparts may need improvements

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

Open problem

No equivalent characterization is known that is more convenient to check

Second attempt (no OR connectors)



Formalization and Verification of Event-driven Process Chains

W.M.P. van der Aalst

Department of Mathematics and Computing Science, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, telephone: -31 40 2474295,
e-mail: wsinwa@win.tue.nl

Simplified EPC

We restrict the analysis to a sub-class of EPC diagrams

We require:

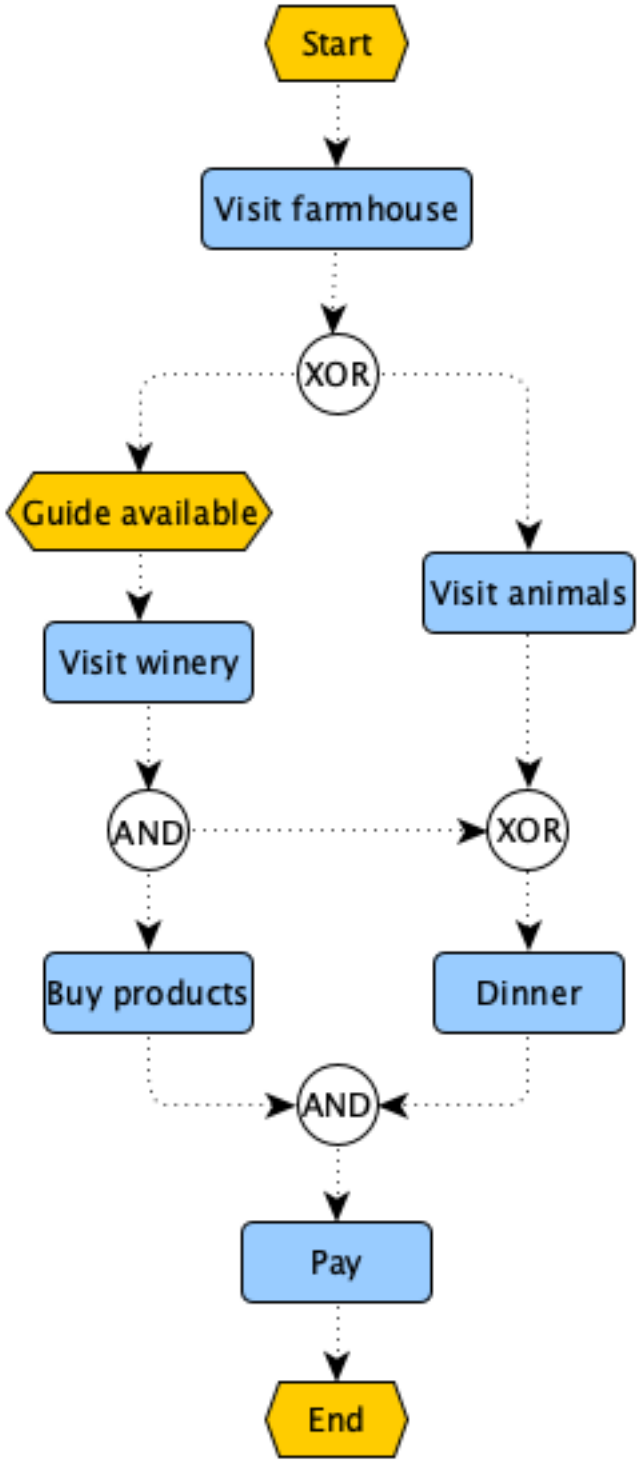
event / function alternation
(also along paths between two connectors)
(fusion not needed, dummy places/transitions not needed)

OR-connectors are not present
(avoid intrinsic problems with OR join)

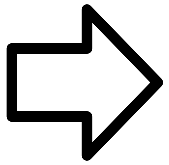
Example

OR-connectors are not present

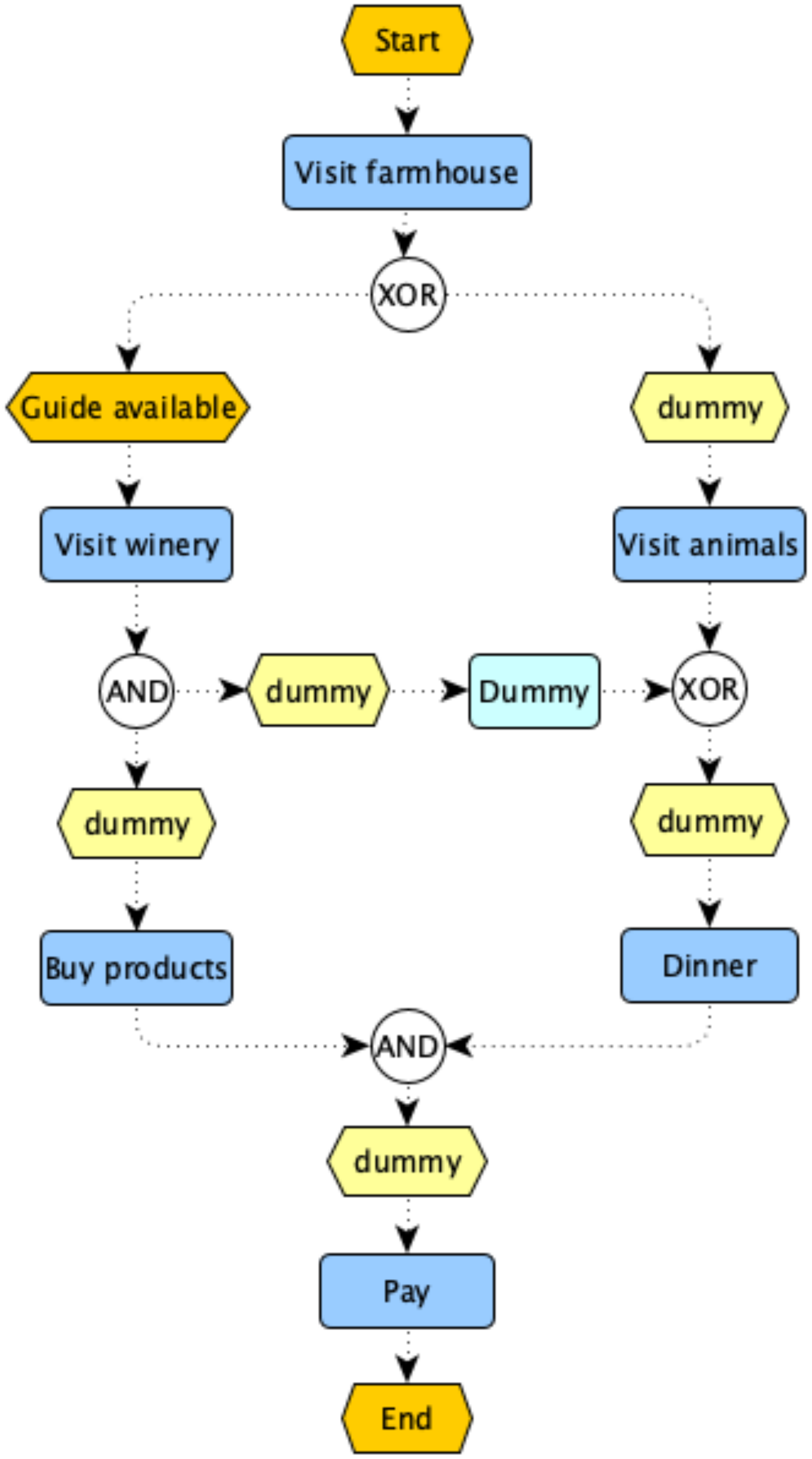
alternation is not satisfied



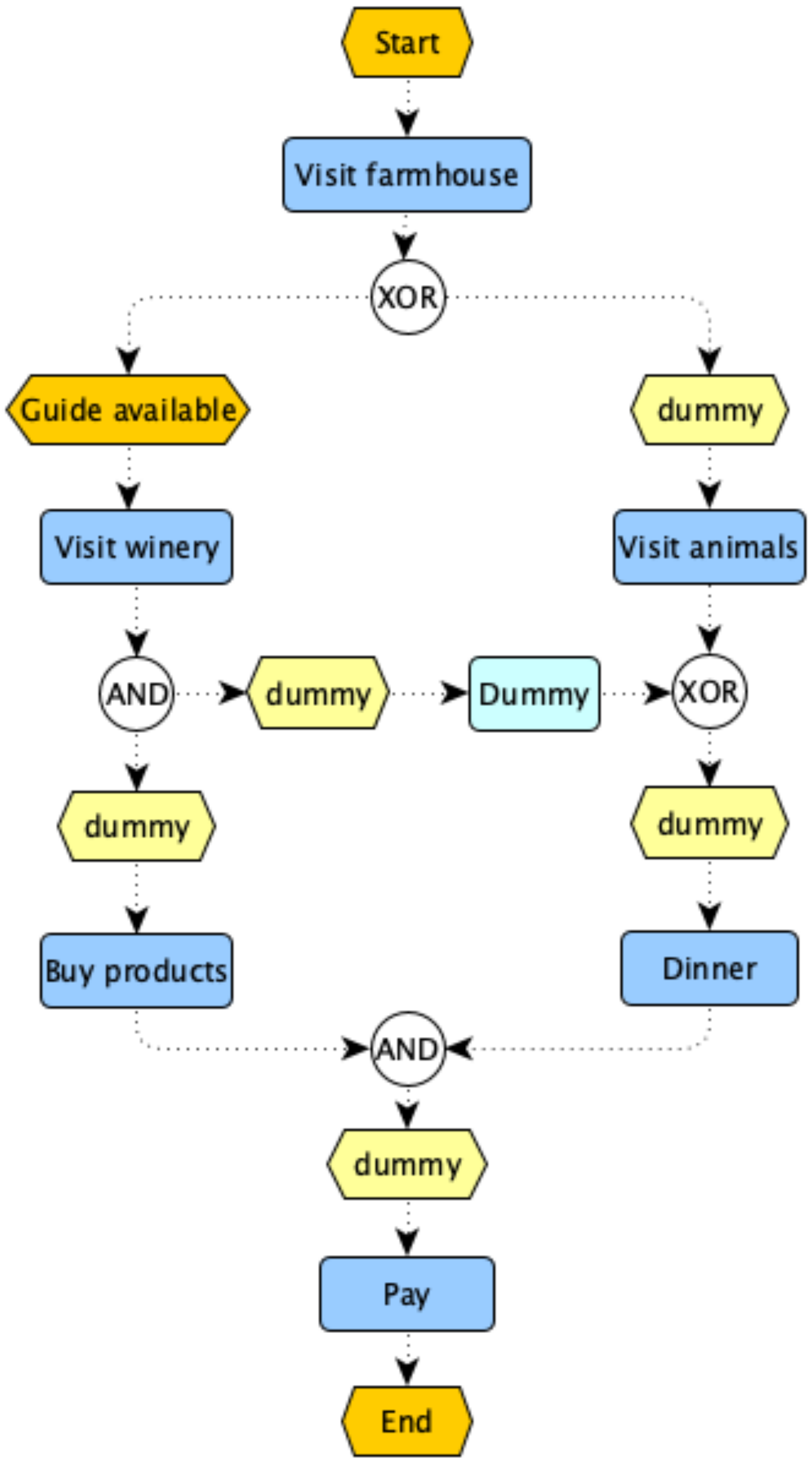
Add dummy events and functions to force alternation



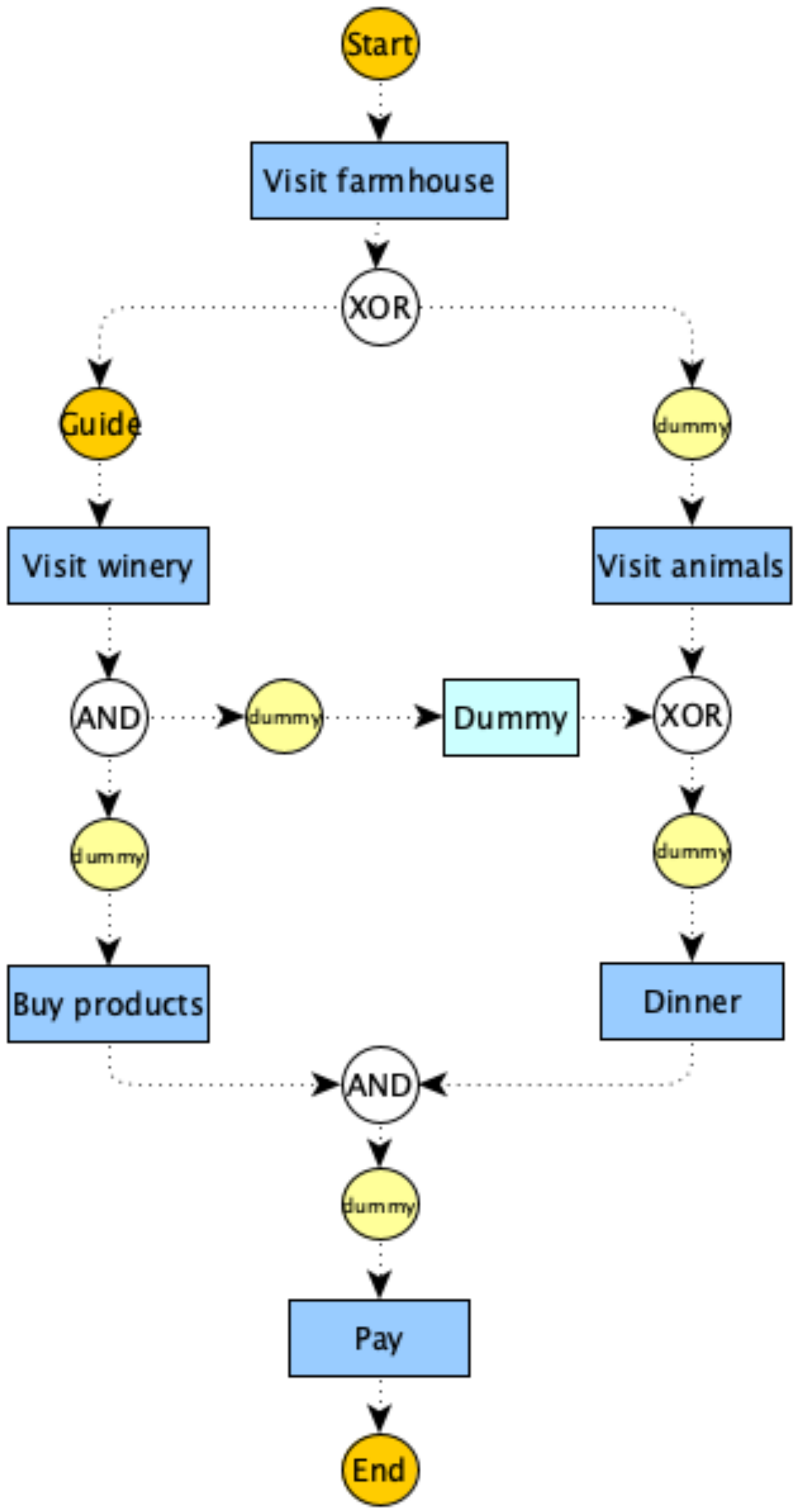
Step 0



Example



Step 1
events and
functions



Step 1: split/join connectors

The translation of logical connectors
depends on the context:

if a connector connects **functions to events**
we apply a certain translation

if it connects **events to functions**
we apply a different translation

Step 1:
split/join connectors

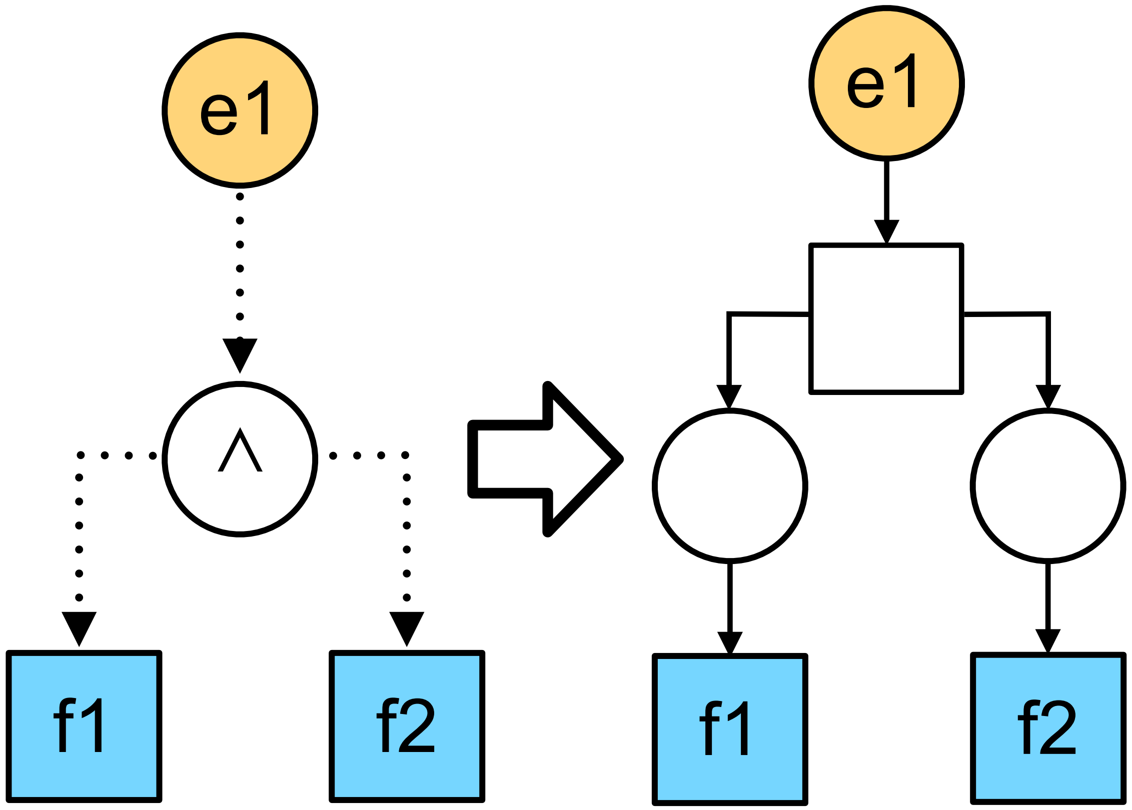
The translation of logical connectors **depends on the context**:

if a connector connects **transitions to places** we apply a certain translation

if it connects **places to transitions** we apply a different translation

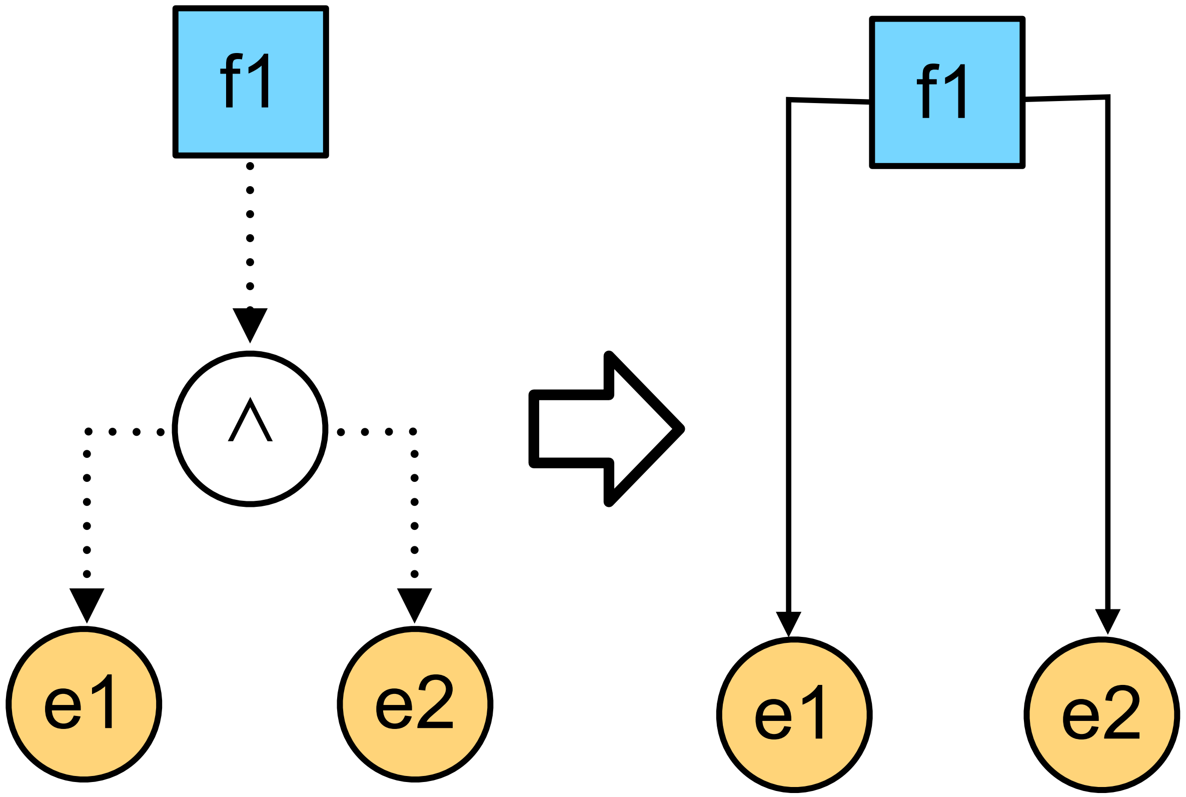
Step 1: AND split

EPC net fragment



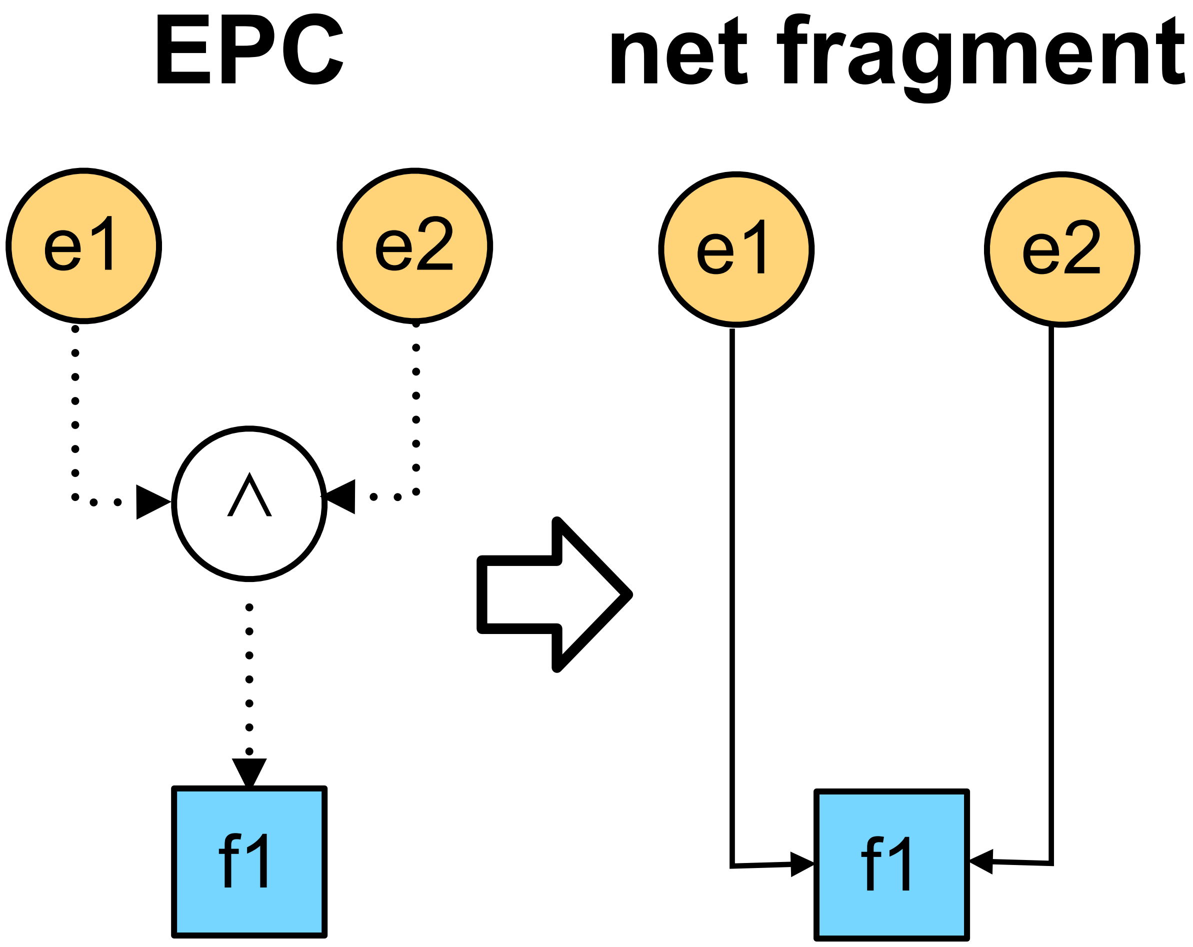
(event to functions)

EPC net fragment

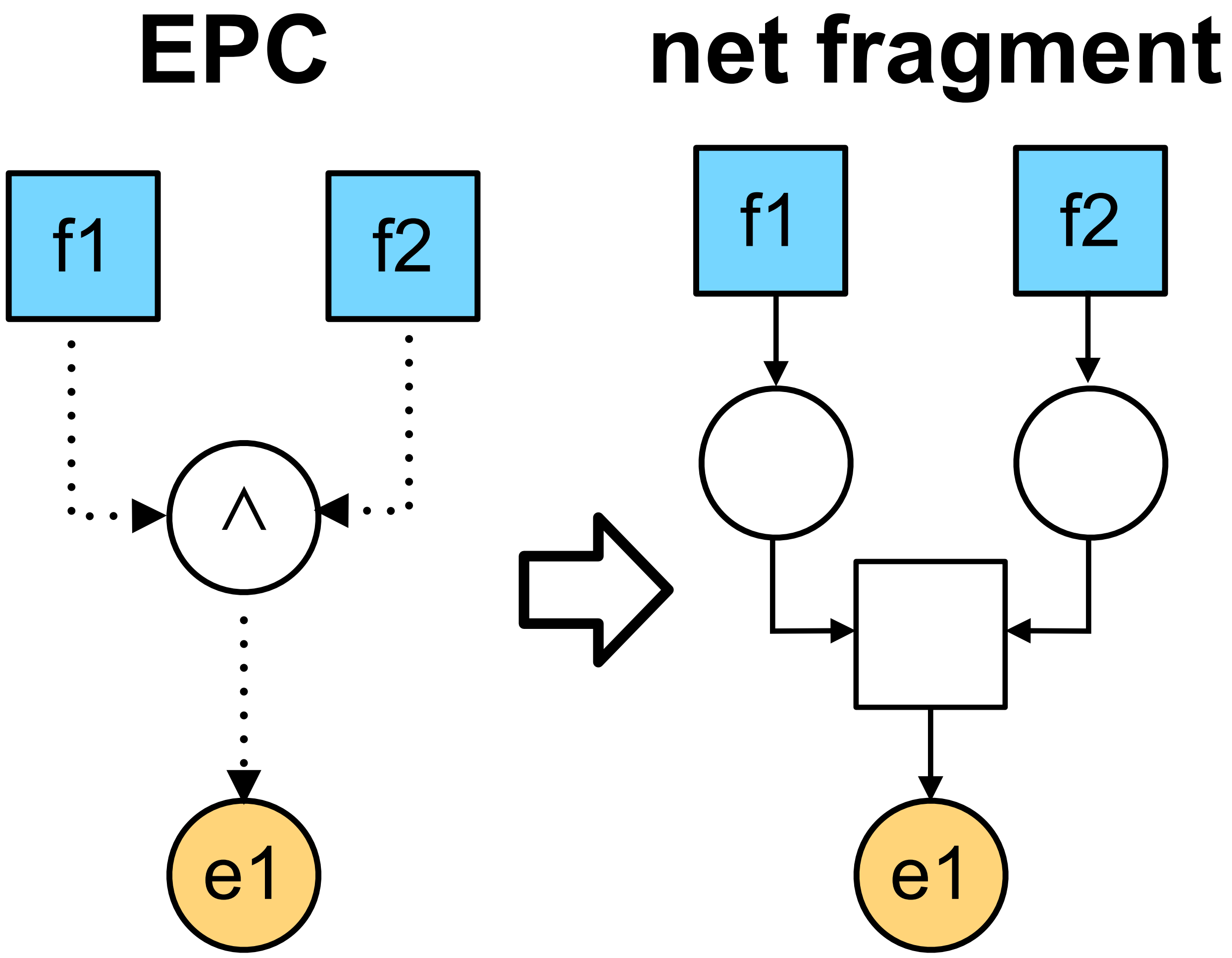


(functions to events)

Step 1: AND join

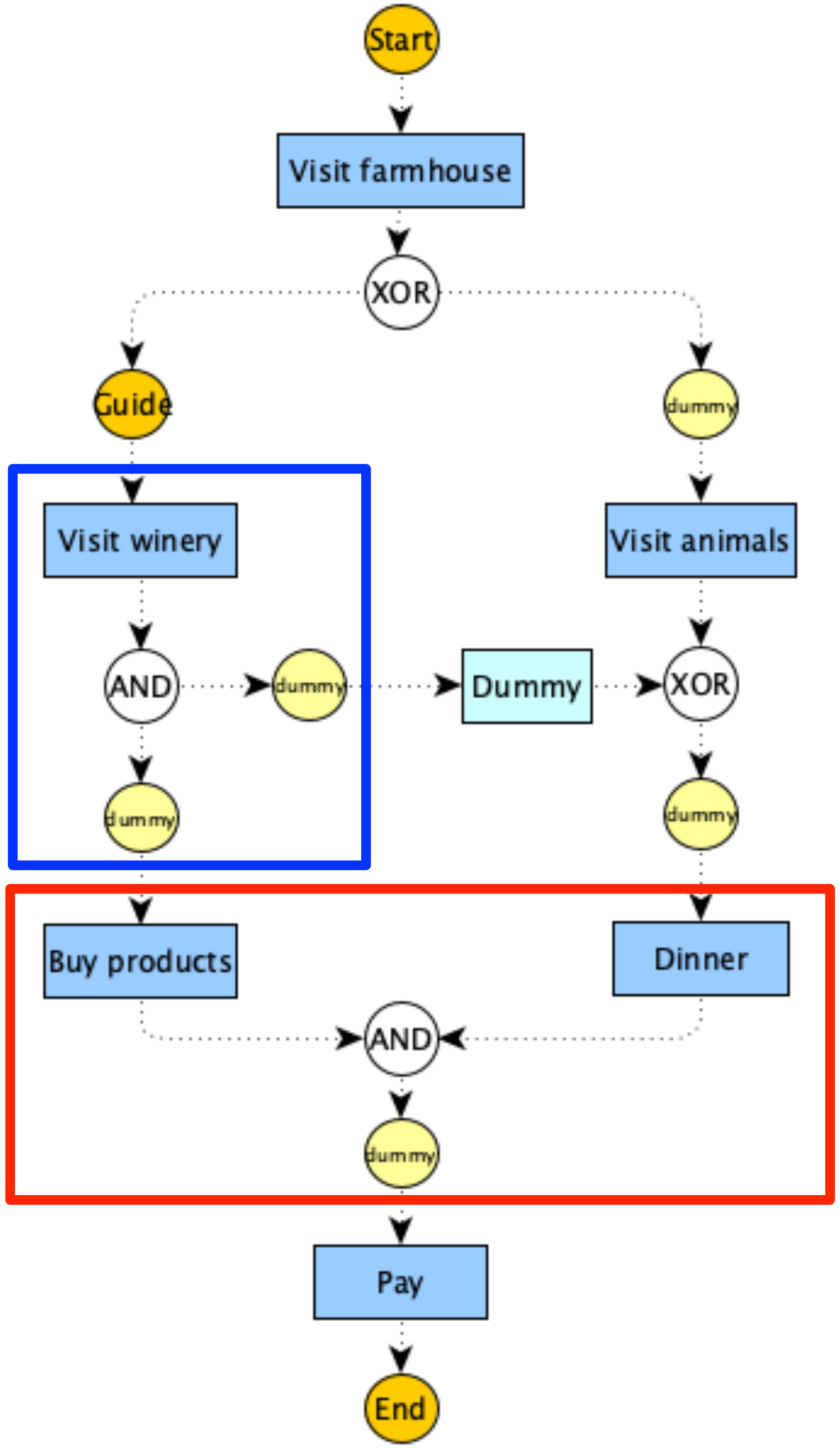


(event to functions)

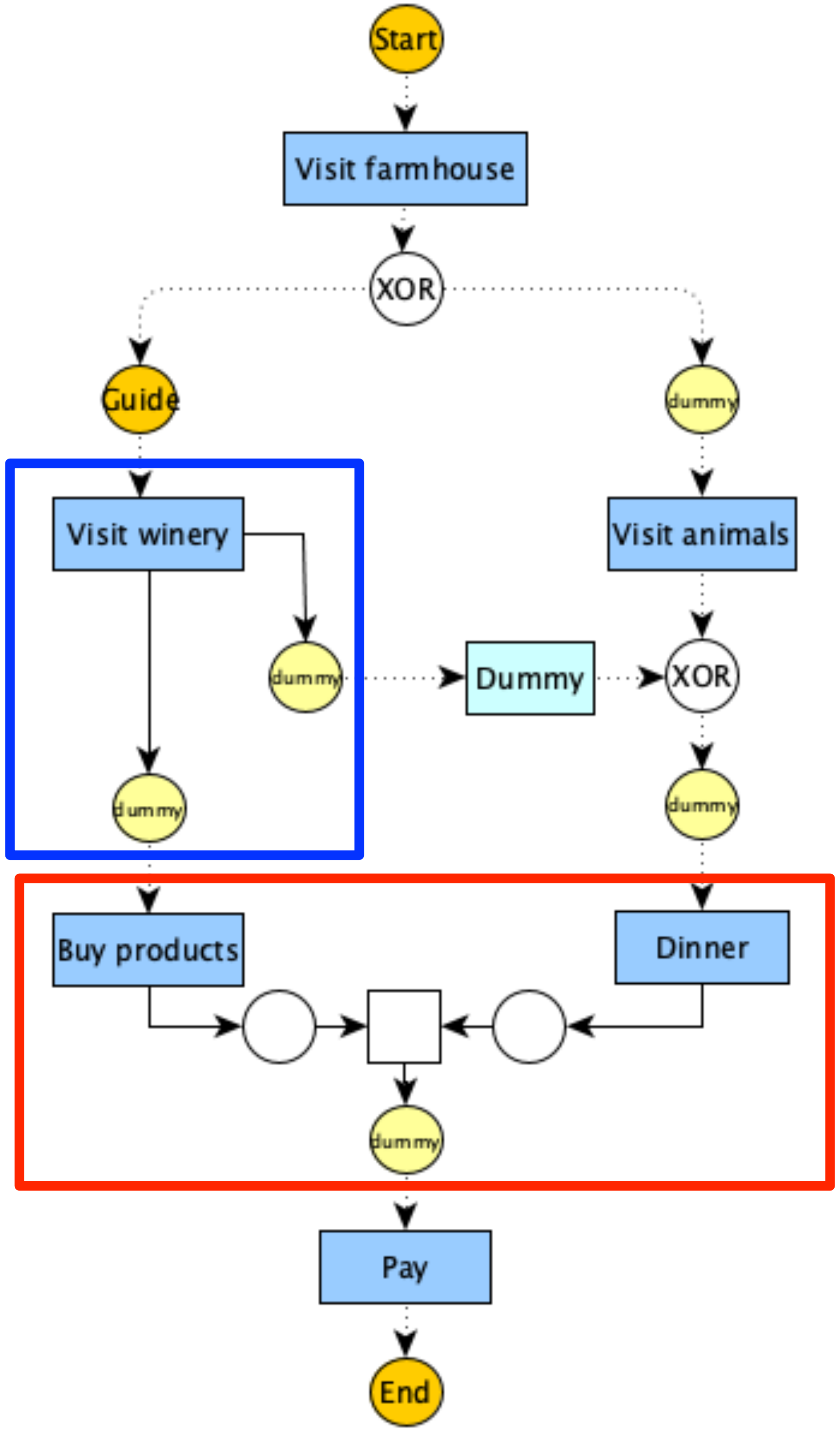


(functions to events)

Example

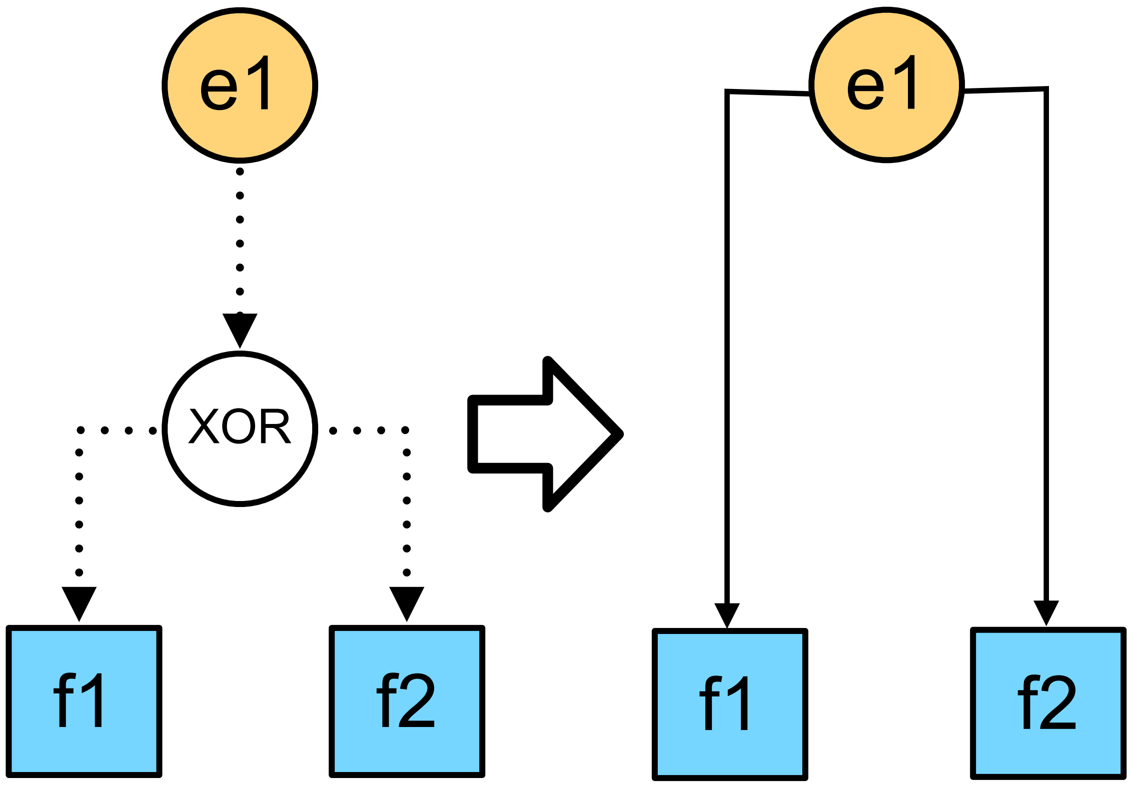


Step 1
AND
connectors



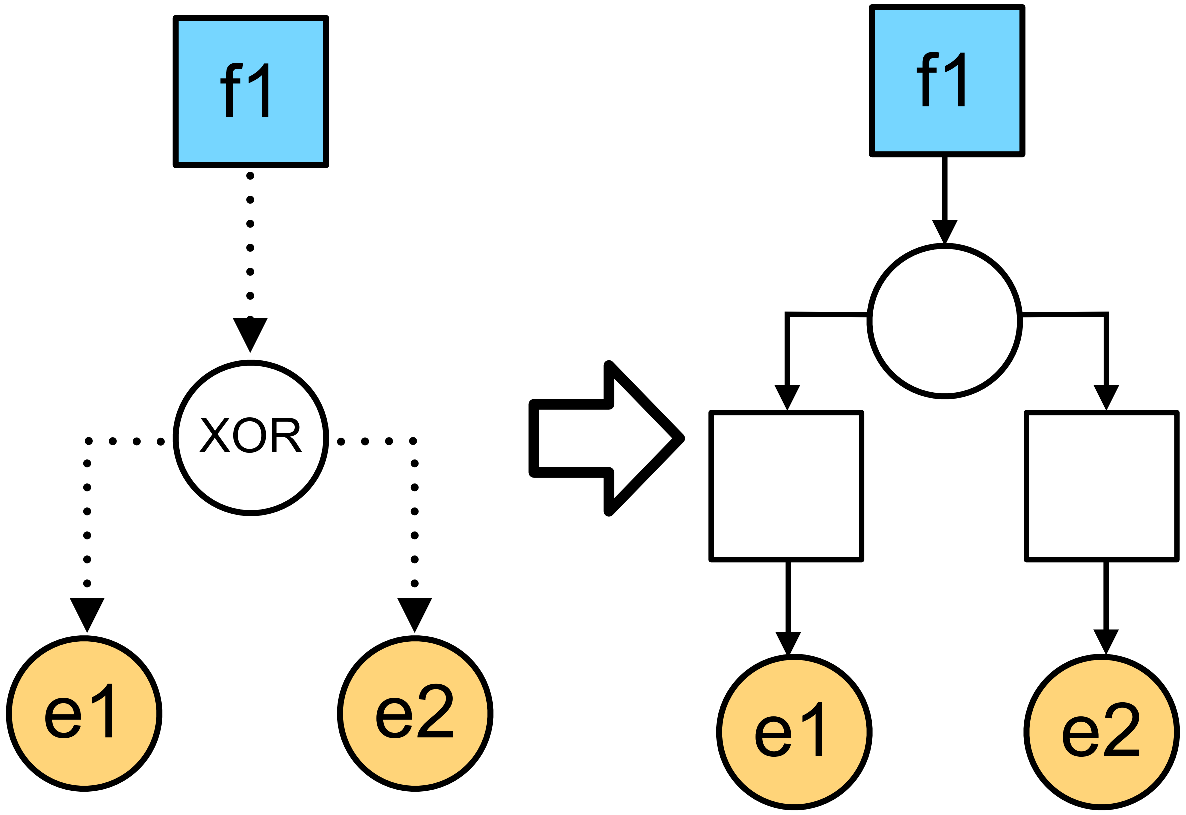
Step 1: XOR split

EPC net fragment



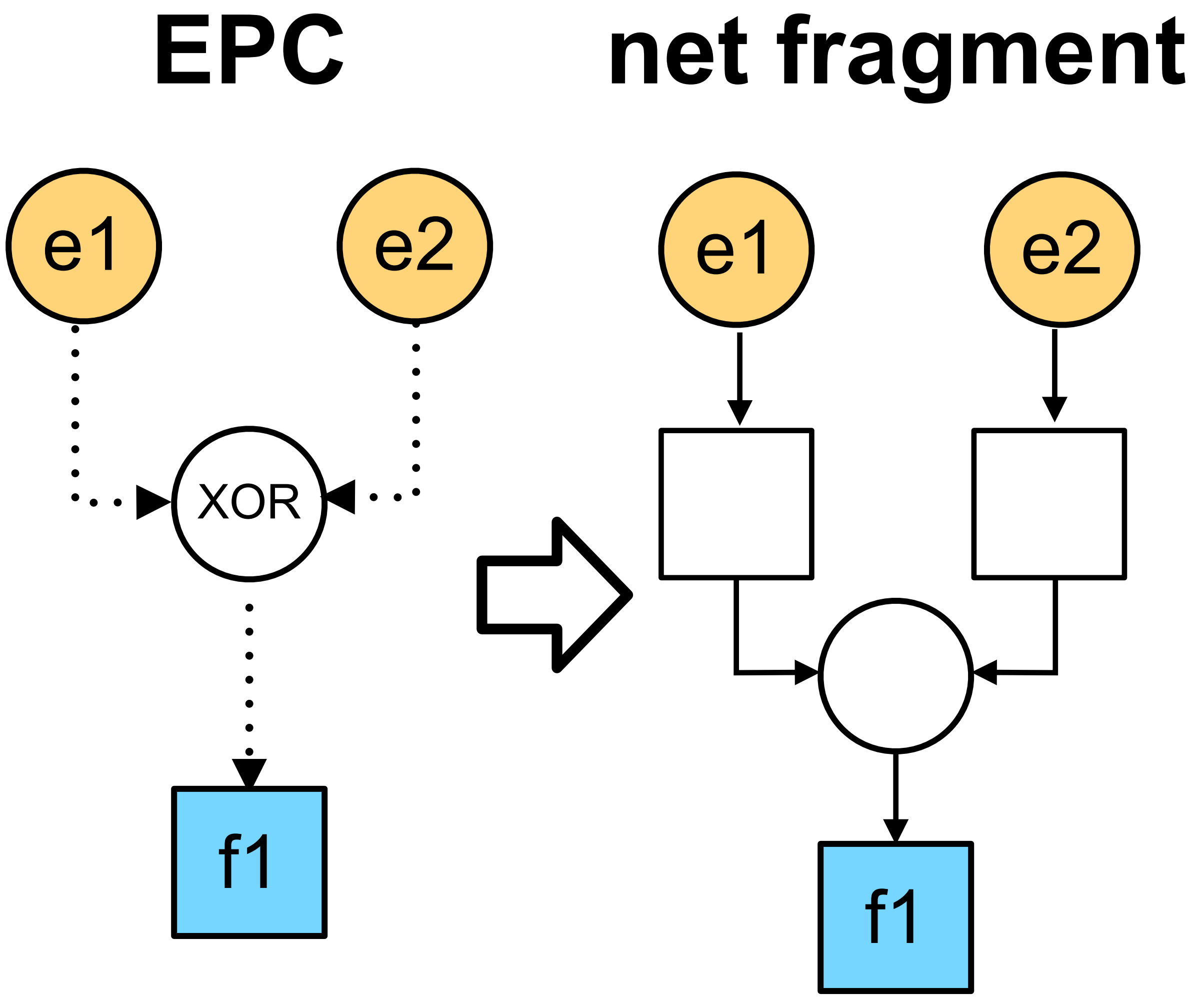
(event to functions)

EPC net fragment



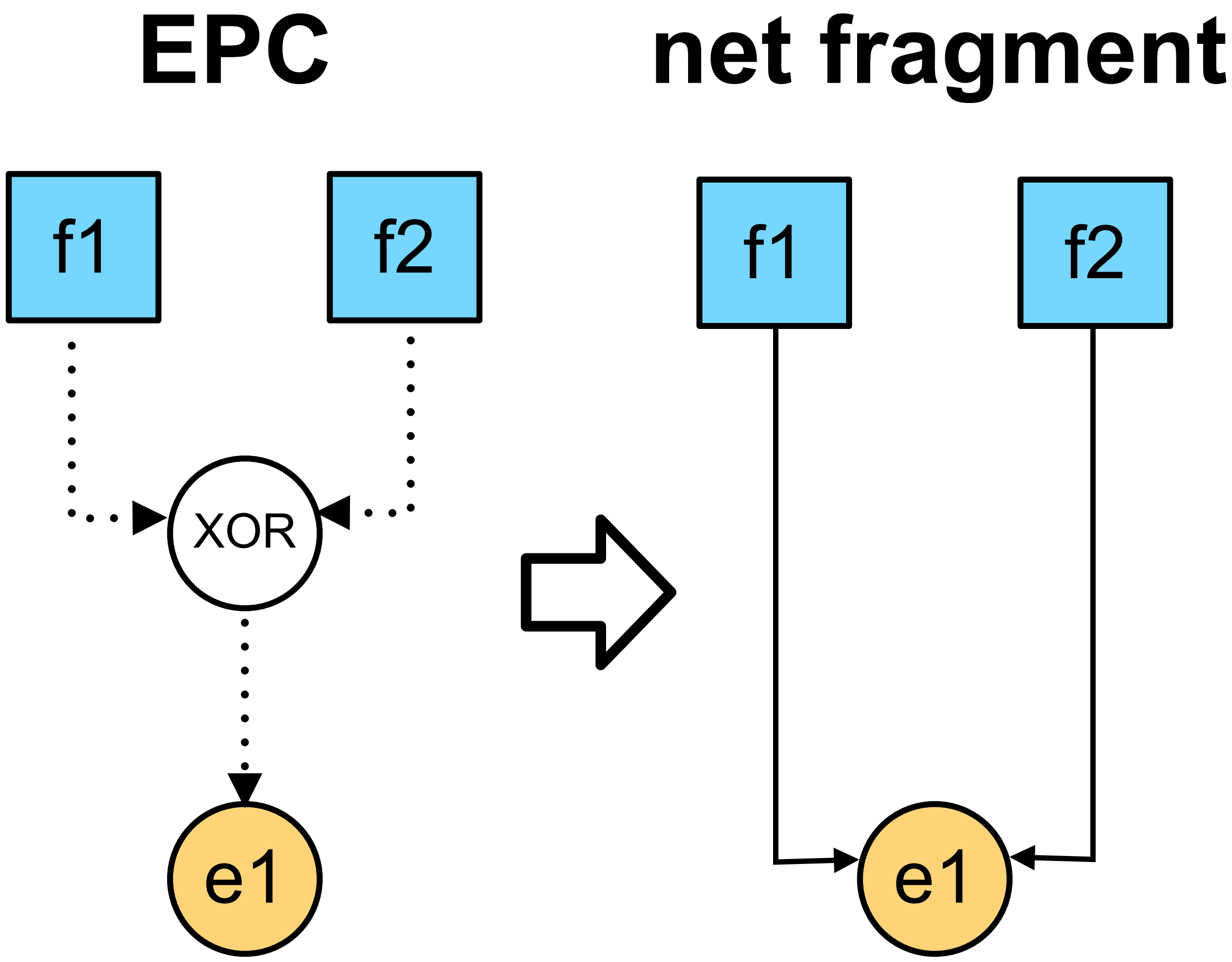
(functions to events)

Step 1: XOR join


EPC

net fragment

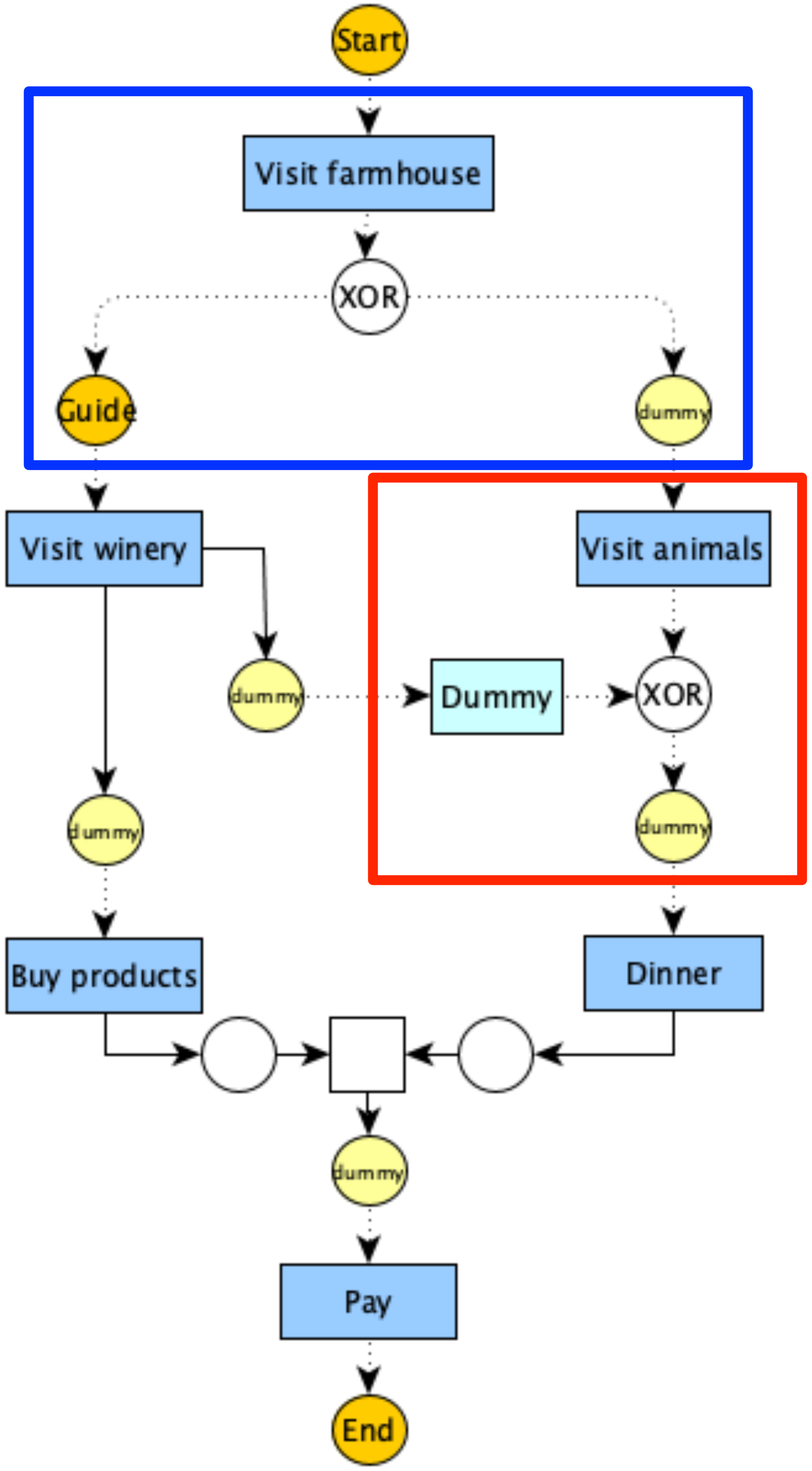
(event to functions)


EPC

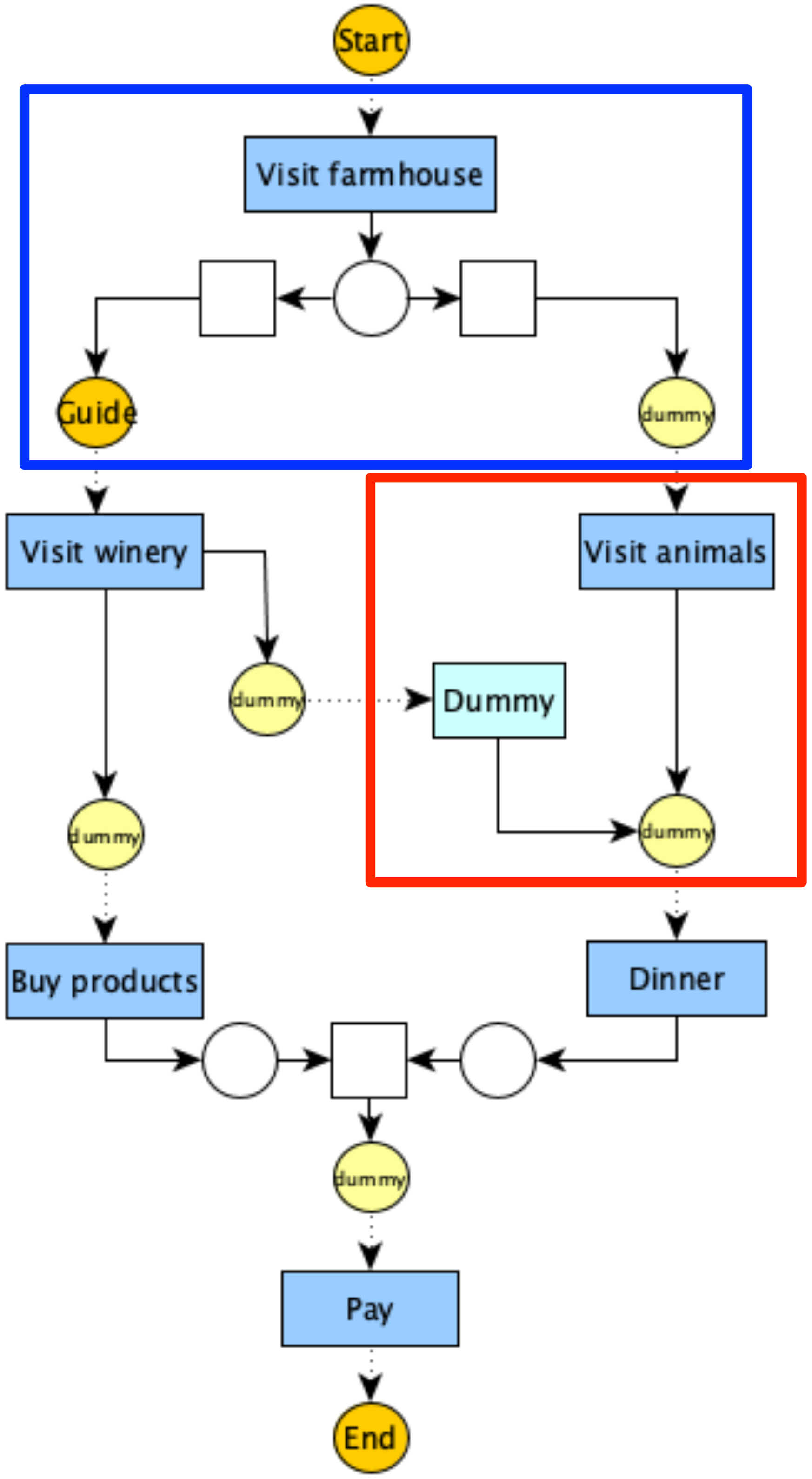
net fragment

(functions to events)

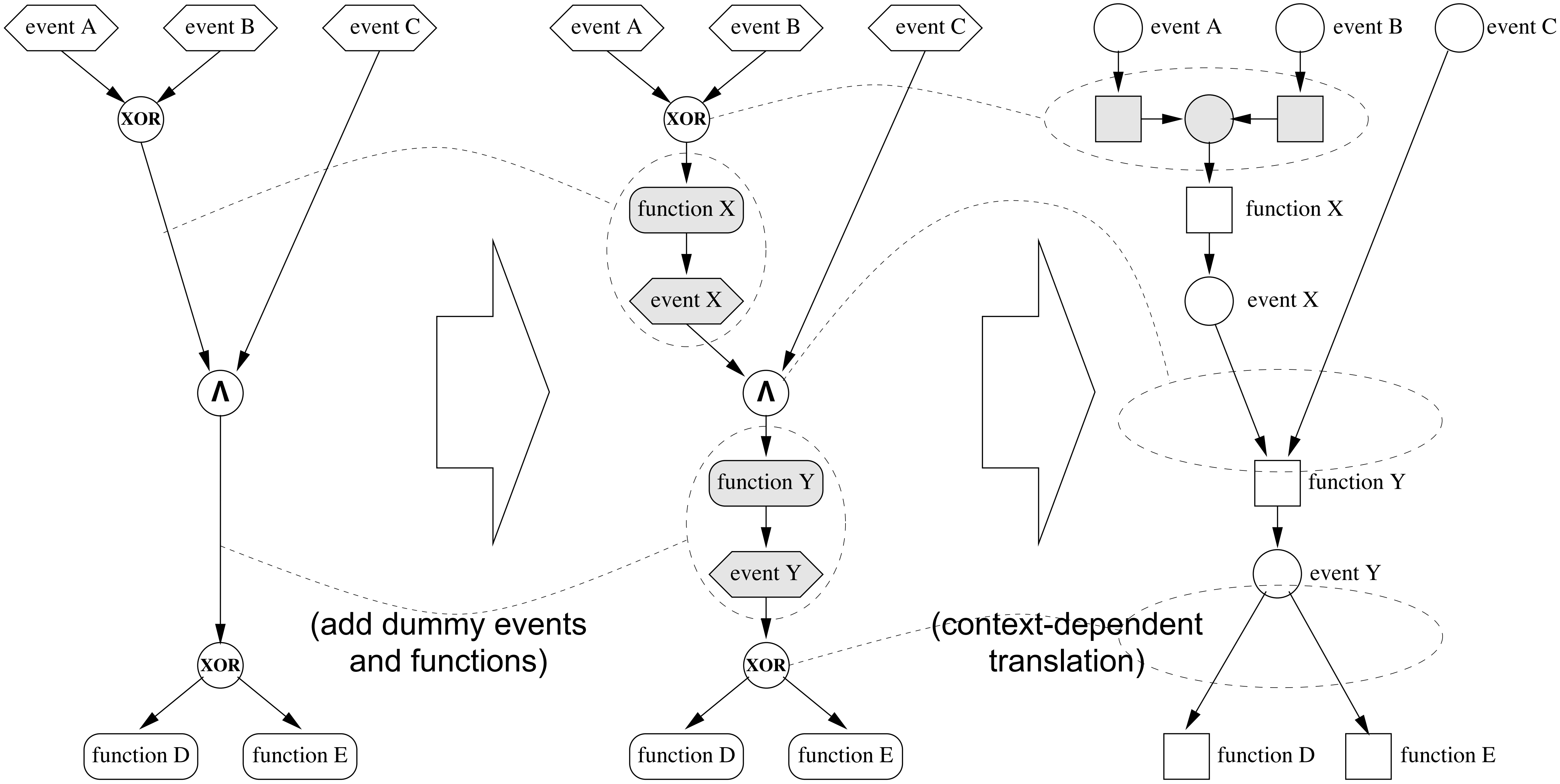
Example



Step 1
XOR
connectors



Overall strategy

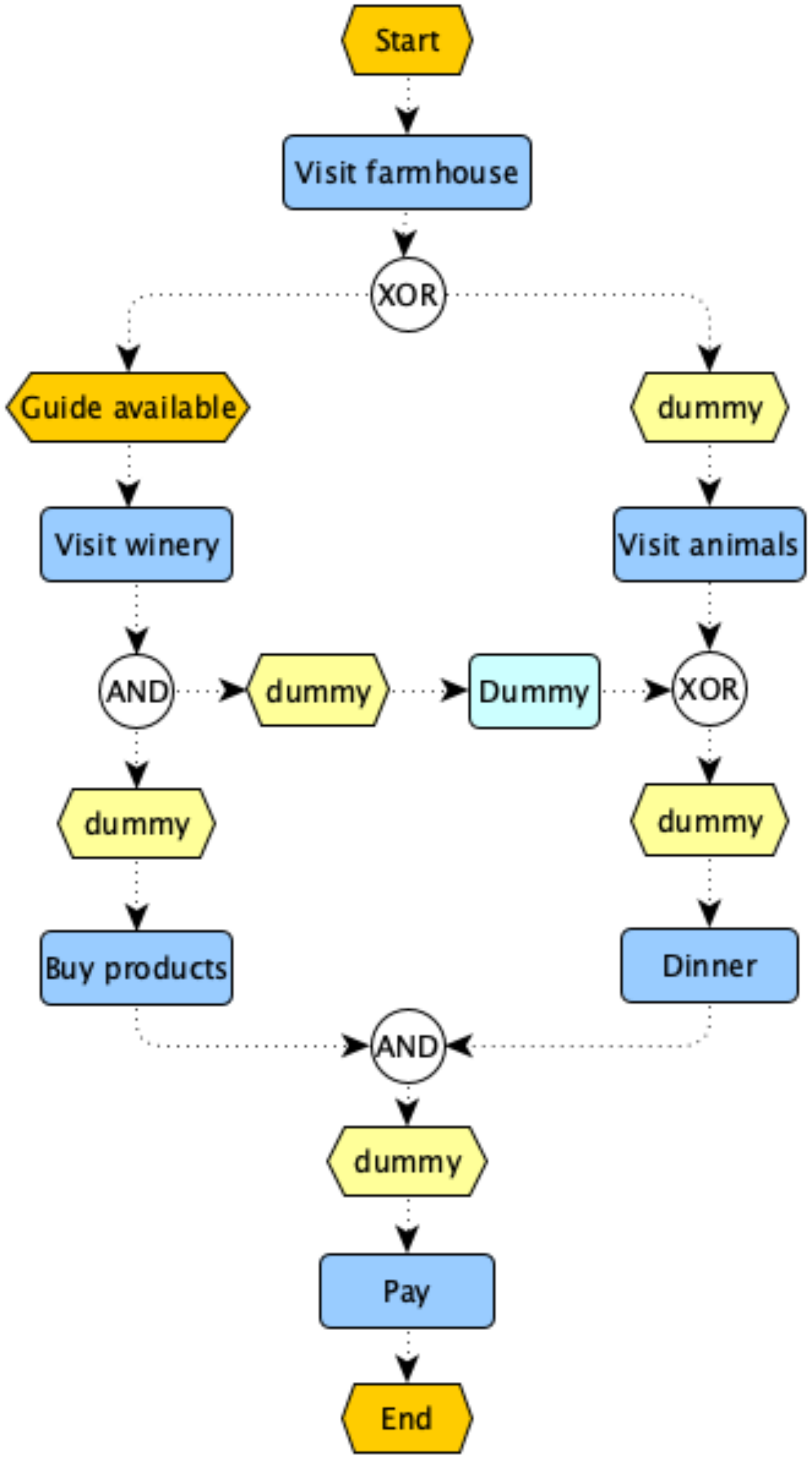


(add dummy events and functions)

(context-dependent translation)

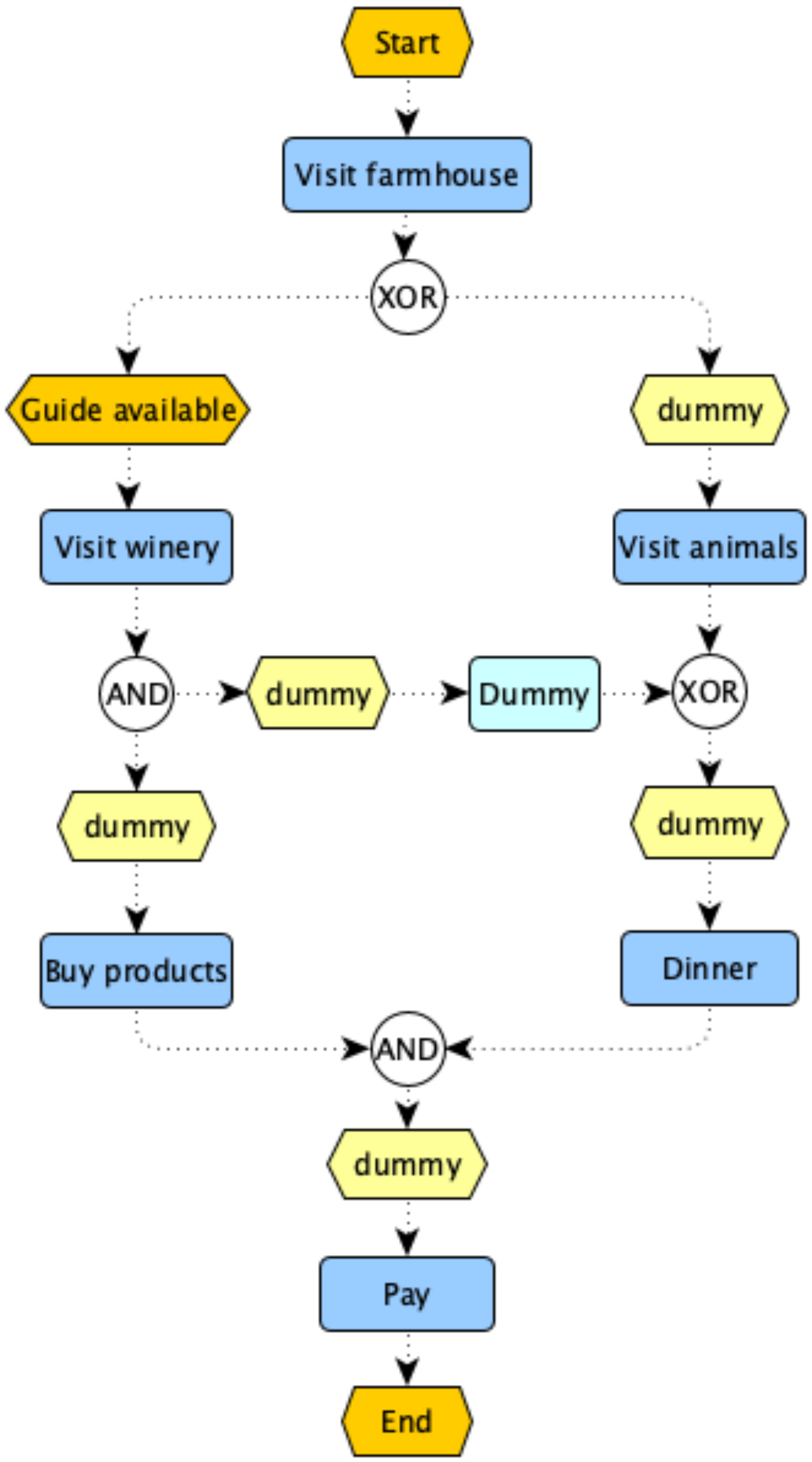
From any EPC we derive a free-choice net

Example

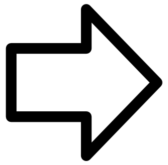


Sound?

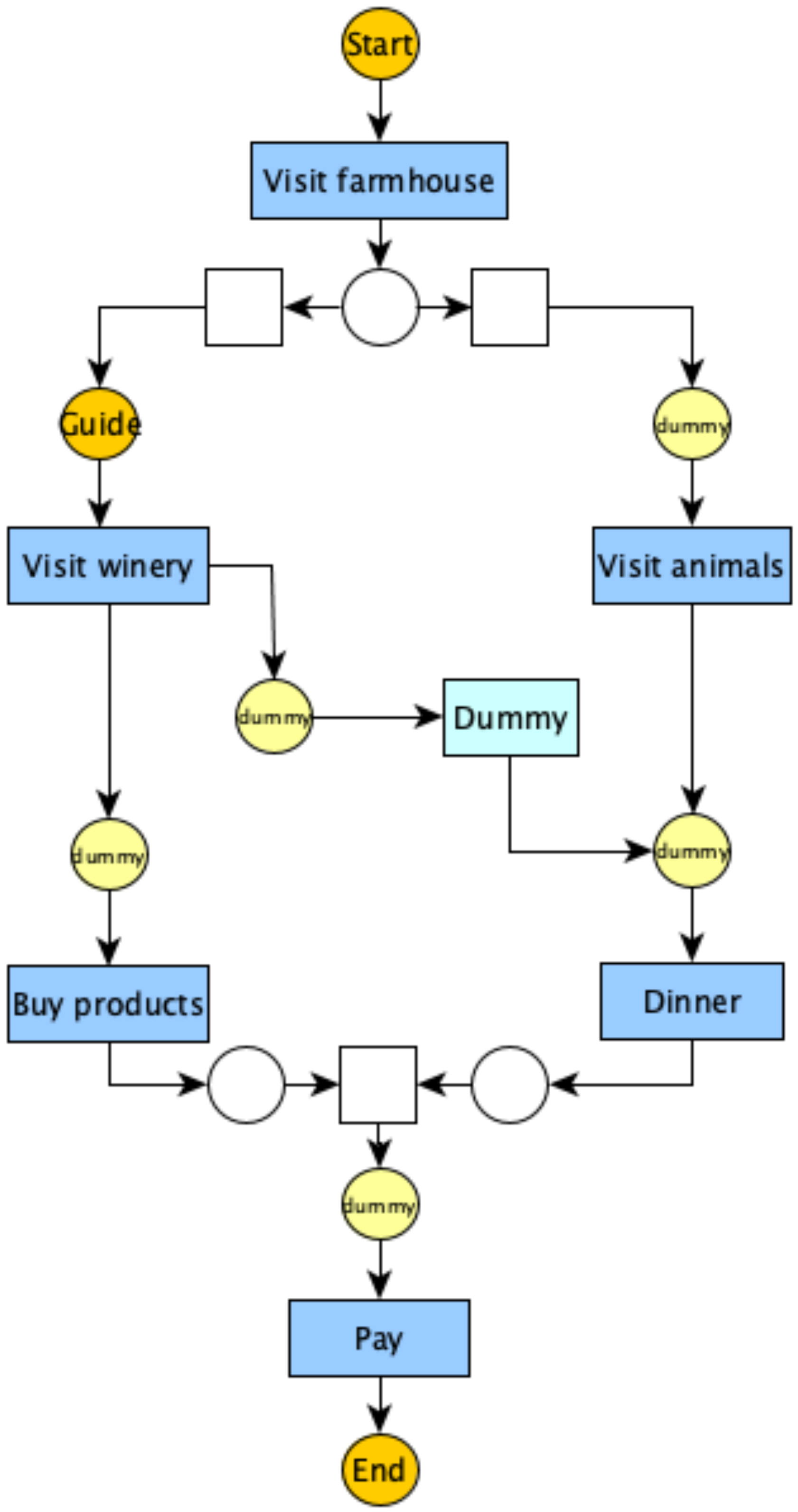
Example



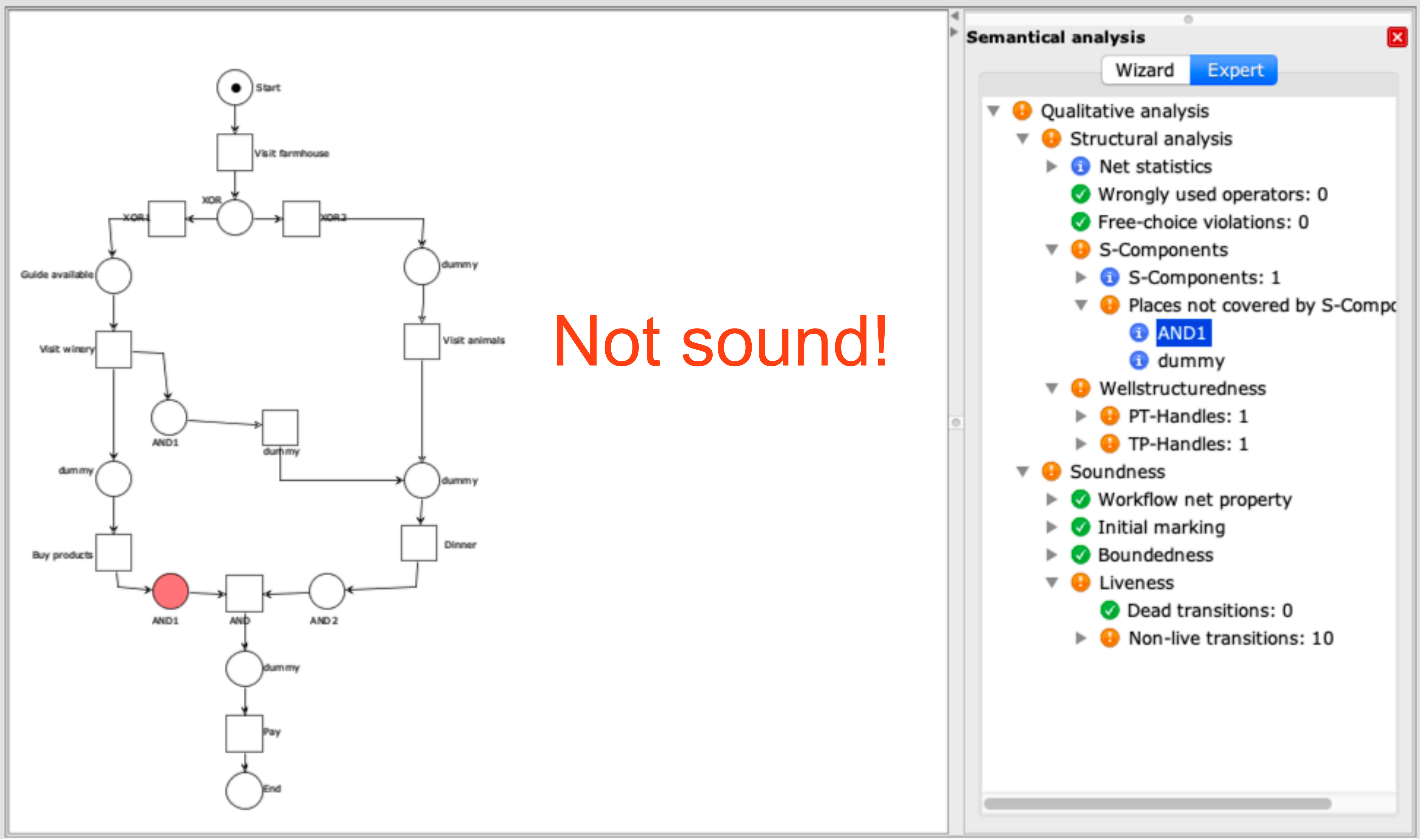
Sound?



Steps
1+2(+3)



Example



Not sound!

```
▼ ! Qualitative analysis
    ▼ ! Structural analysis
        ▶ i Net statistics
          ✓ Wrongly used operators: 0
          ✓ Free-choice violations: 0
        ▼ ! S-Components
            ▶ i S-Components: 1
            ▼ ! Places not covered by S-Compo
                  i AND1
                  i dummy
        ▼ ! Wellstructuredness
            ▶ ! PT-Handles: 1
            ▶ ! TP-Handles: 1
    ▼ ! Soundness
        ▶ ✓ Workflow net property
        ▶ ✓ Initial marking
        ▶ ✓ Boundedness
        ▼ ! Liveness
              ✓ Dead transitions: 0
            ▶ ! Non-live transitions: 10
```

Third attempt (decorated EPC)

UNIVERSITÄT KOBLENZ · LANDAU

Institut für Wirtschaftsinformatik

Fachbereich Informatik
Universität Koblenz-Landau

Decorated EPC

Applicable to any EPC diagram, provided that its designer add some information

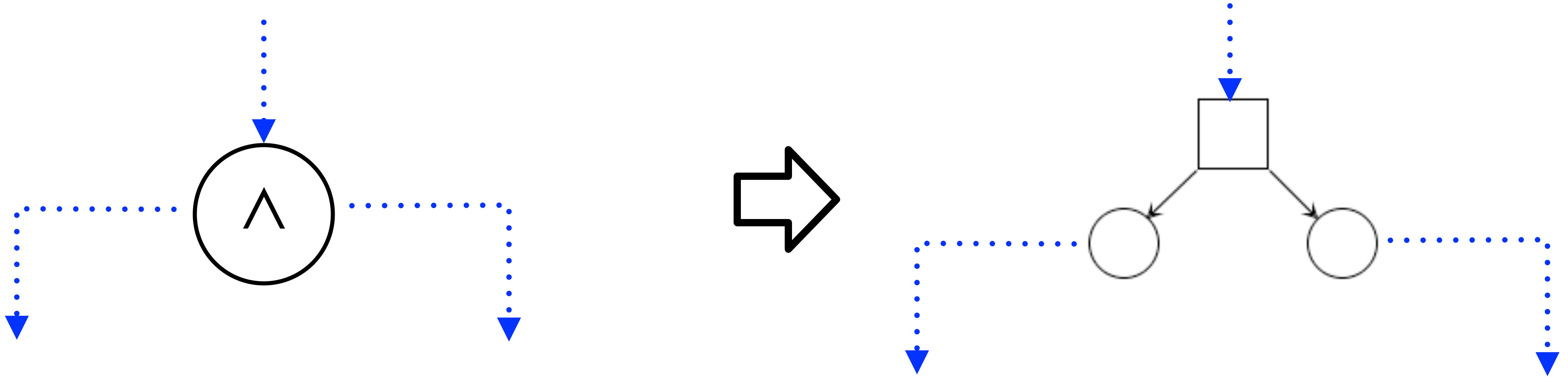
We require:

every (X)OR join is paired with a corresponding split
(possibly of the same type)

OR-joins are decorated with a policy
(avoid OR join ambiguous behaviour)

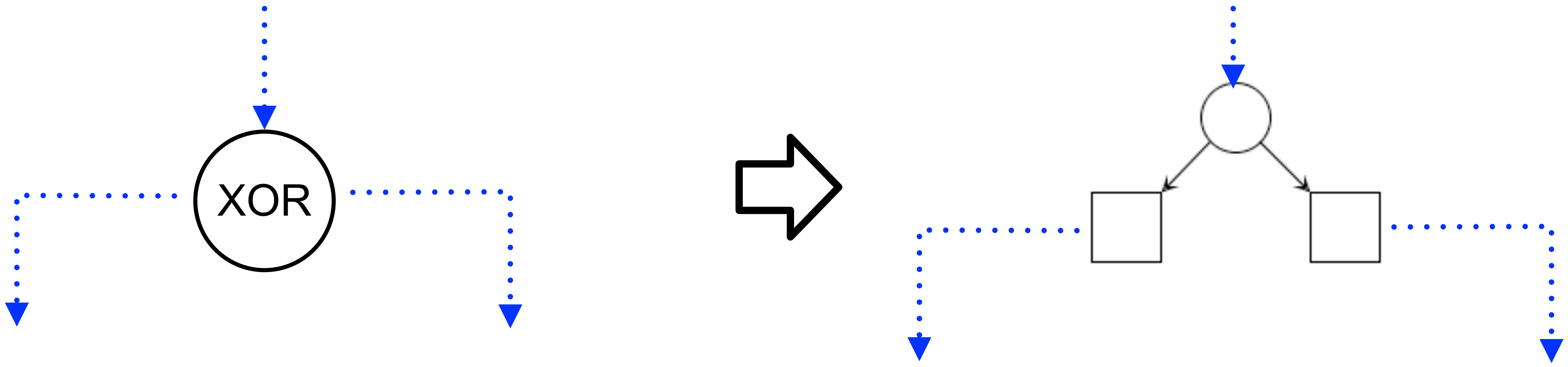
Step 1: AND split

EPC element **net fragment**



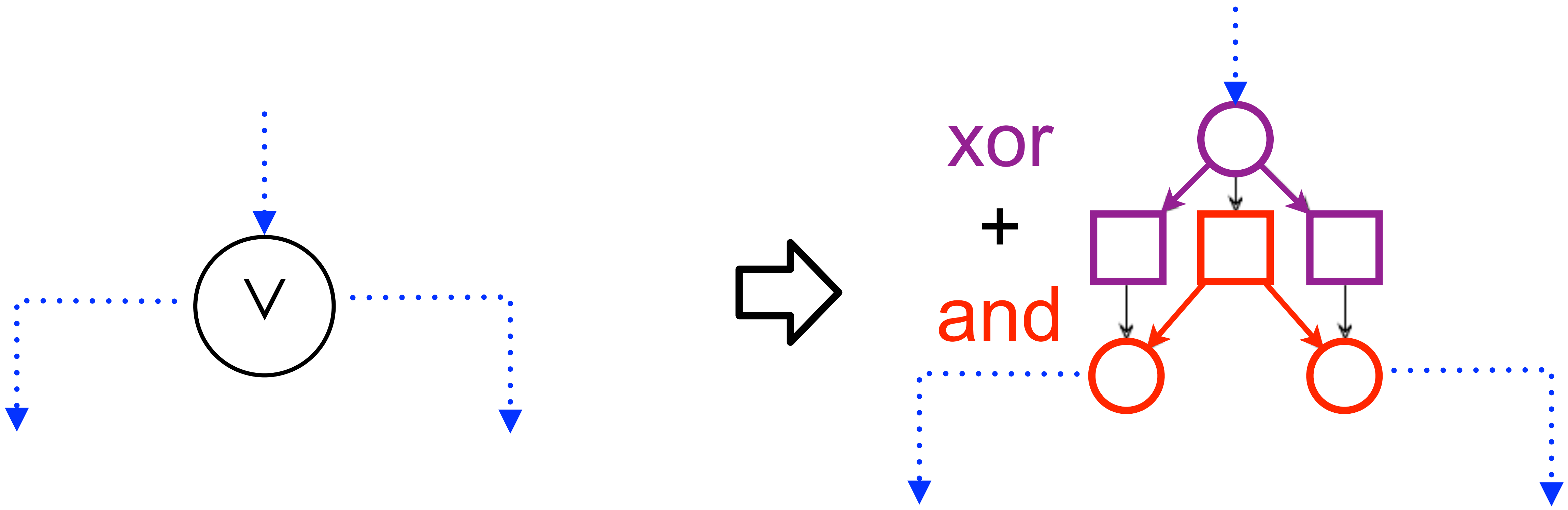
Step 1: XOR split

EPC element **net fragment**



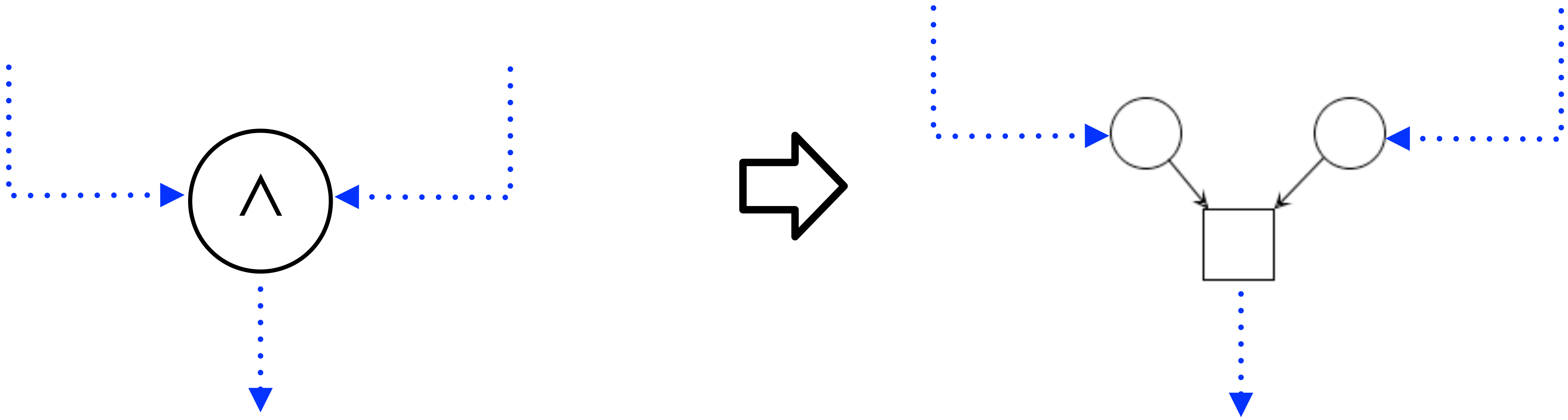
Step 1: OR split

EPC element **net fragment**

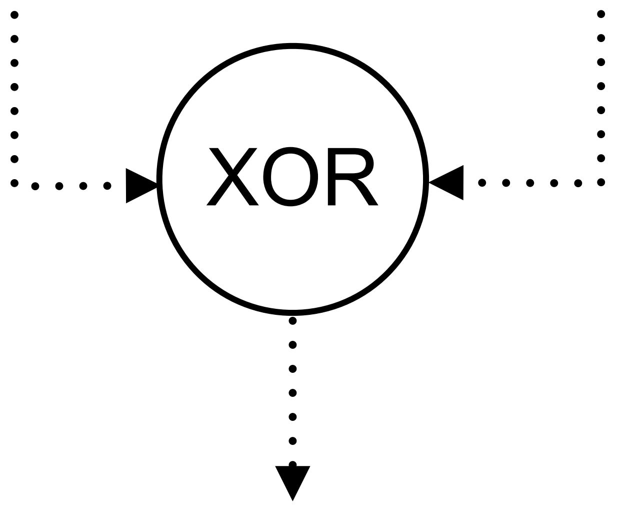


Step 1: AND join

EPC element **net fragment**



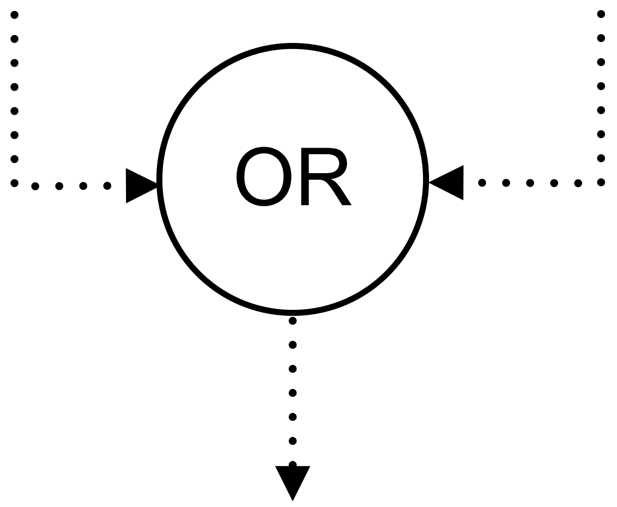
XOR join: intended meaning



if both inputs arrive,
it should block the flow

if one input arrives,
it cannot proceed unless
it is informed that
the other input will never arrive

OR join: intended meaning



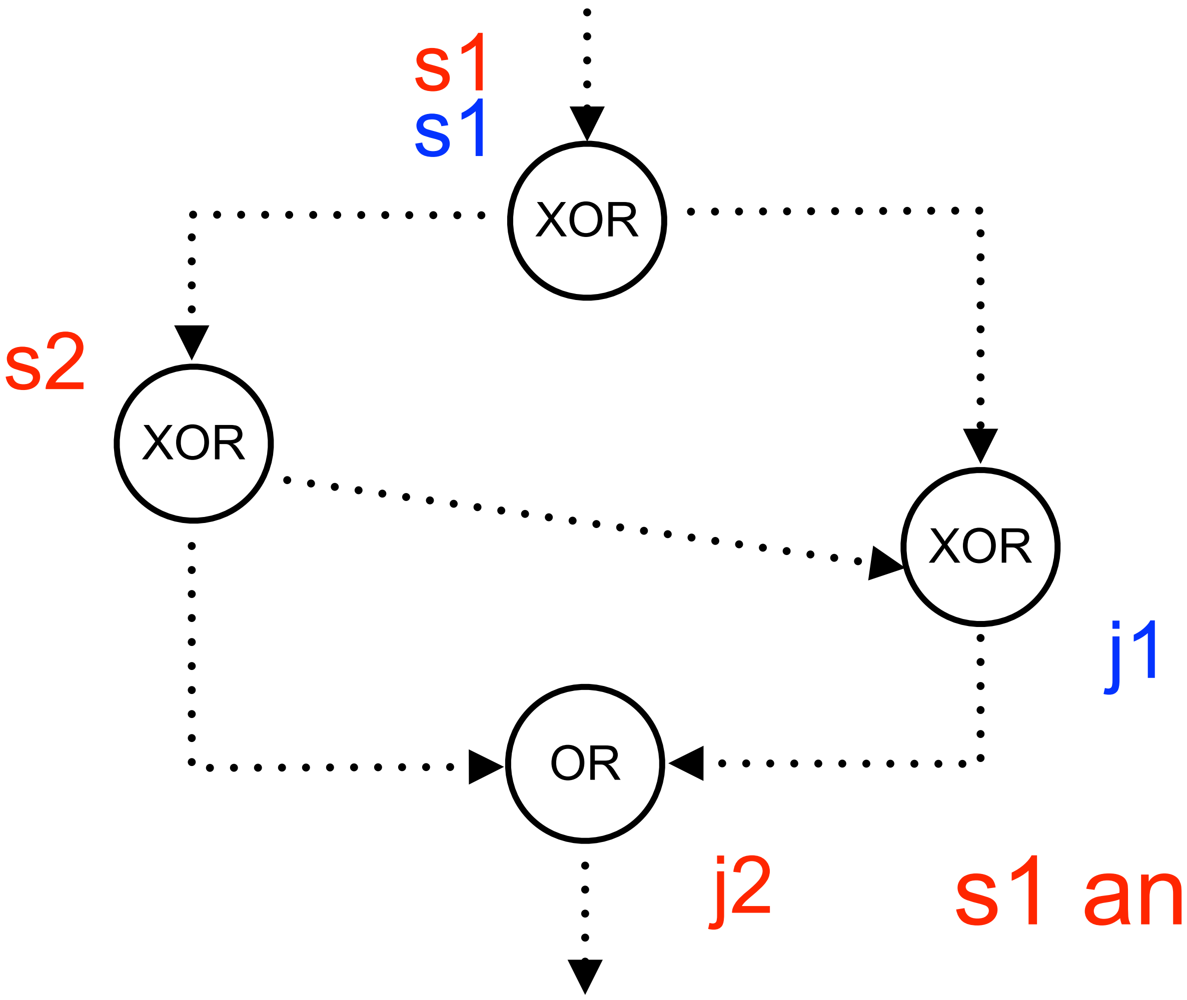
if only one input arrives,
it should release the flow

if both inputs arrive,
it should release only one output

if one input arrives,
it must wait until the other arrives or
it is guaranteed that the other will never arrive

Candidate split

A **candidate split** for a join node is any split node whose outputs are connected to the inputs of the join

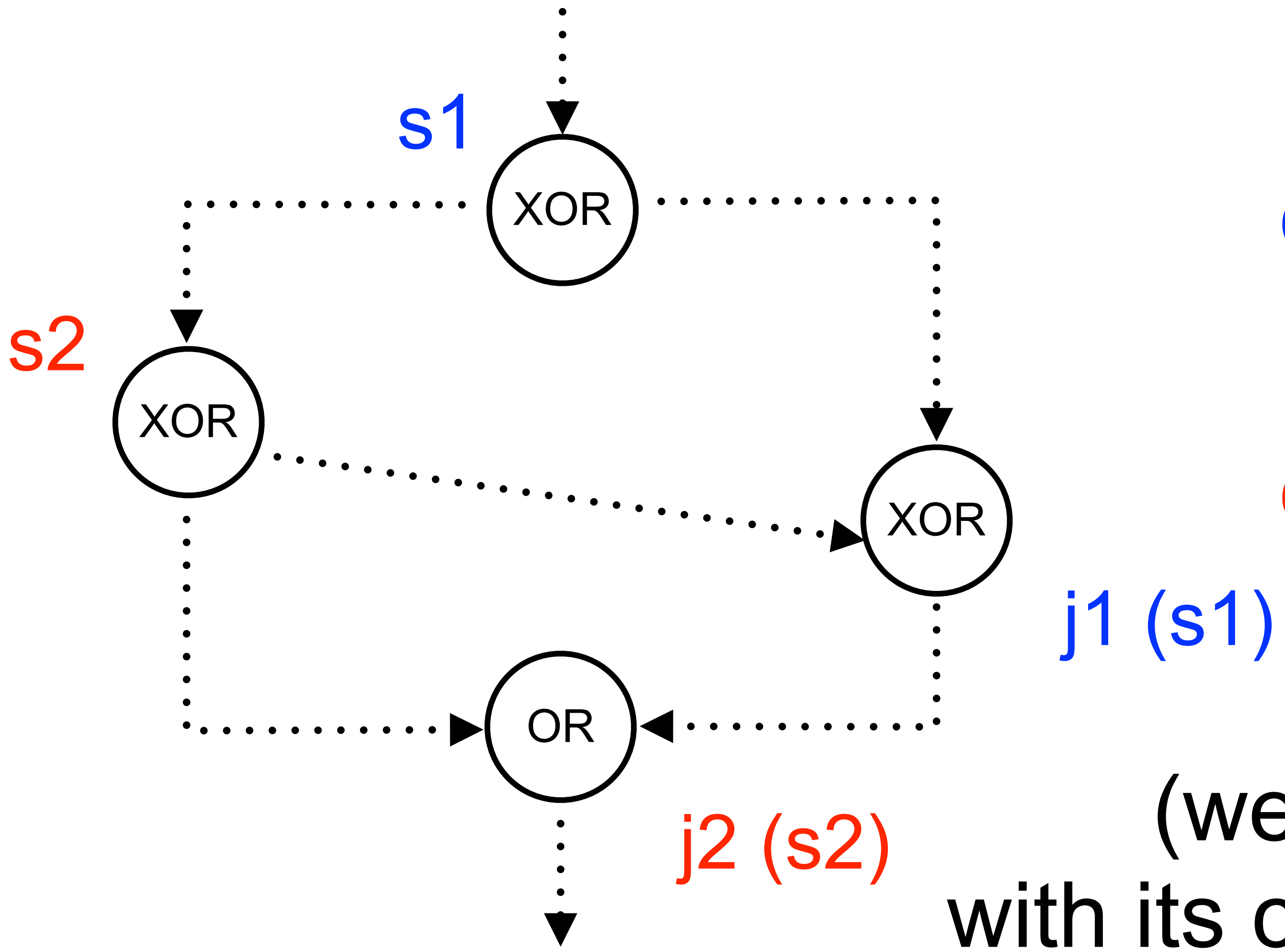


s1 is a candidate split for j1

s1 and s2 are candidate splits for j2

Corresponding split

A **corresponding split** for a join node
is a chosen candidate split



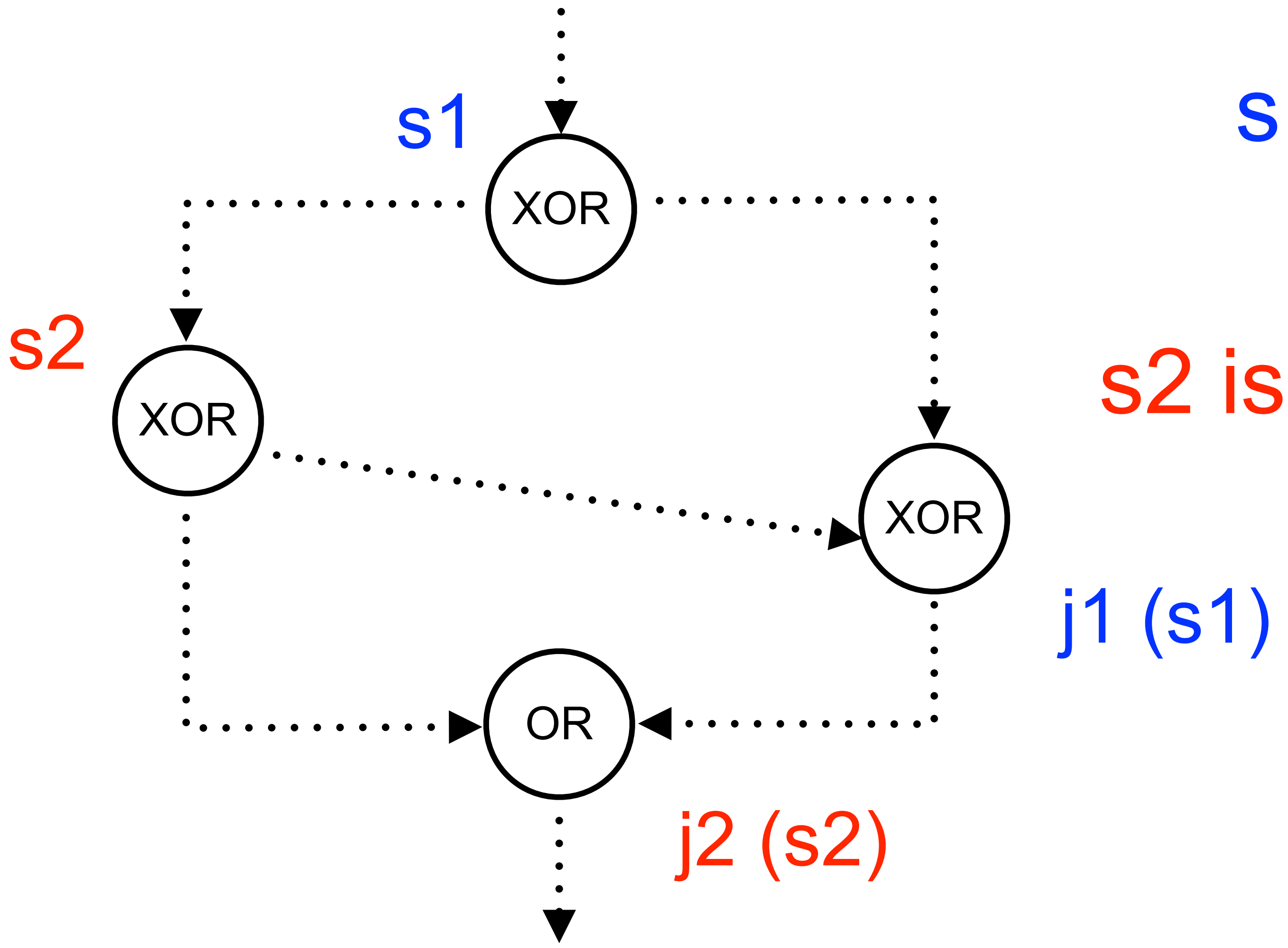
we choose s1 as a
corresponding split for j1

we choose s2 as a
corresponding split for j2

(we tag each join
with its corresponding split)

Matching split

A corresponding split for a join node is called **matching** if it has the same type as the join node



s1 is a matching split for j1

s2 is not a matching split for j2

OR join: assumption

If an OR join has a **matching split**, its semantics is **wait-for-all**: wait for the completion of all *activated* paths

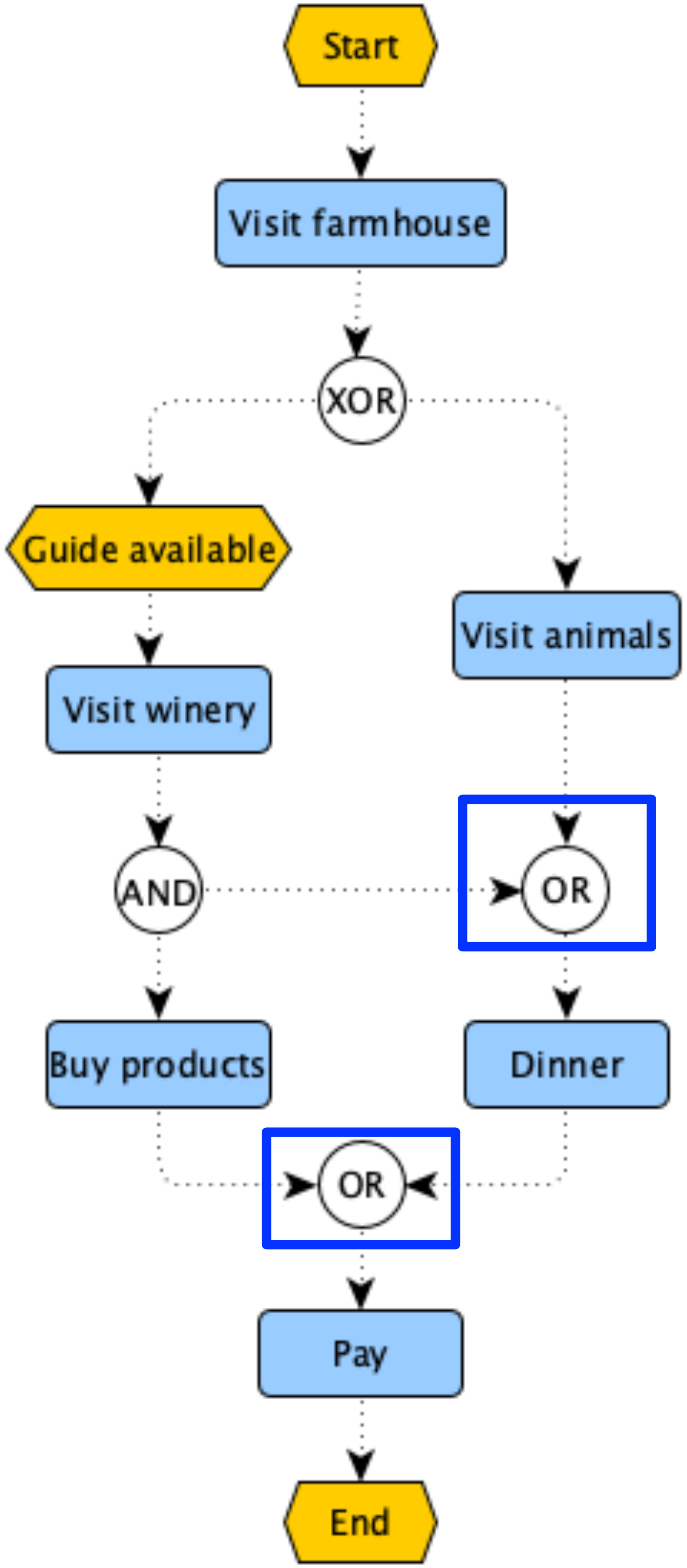
Otherwise, also other policies can be chosen:

every-time: trigger the outgoing path on each input

first-come: wait for the first input and ignore the second

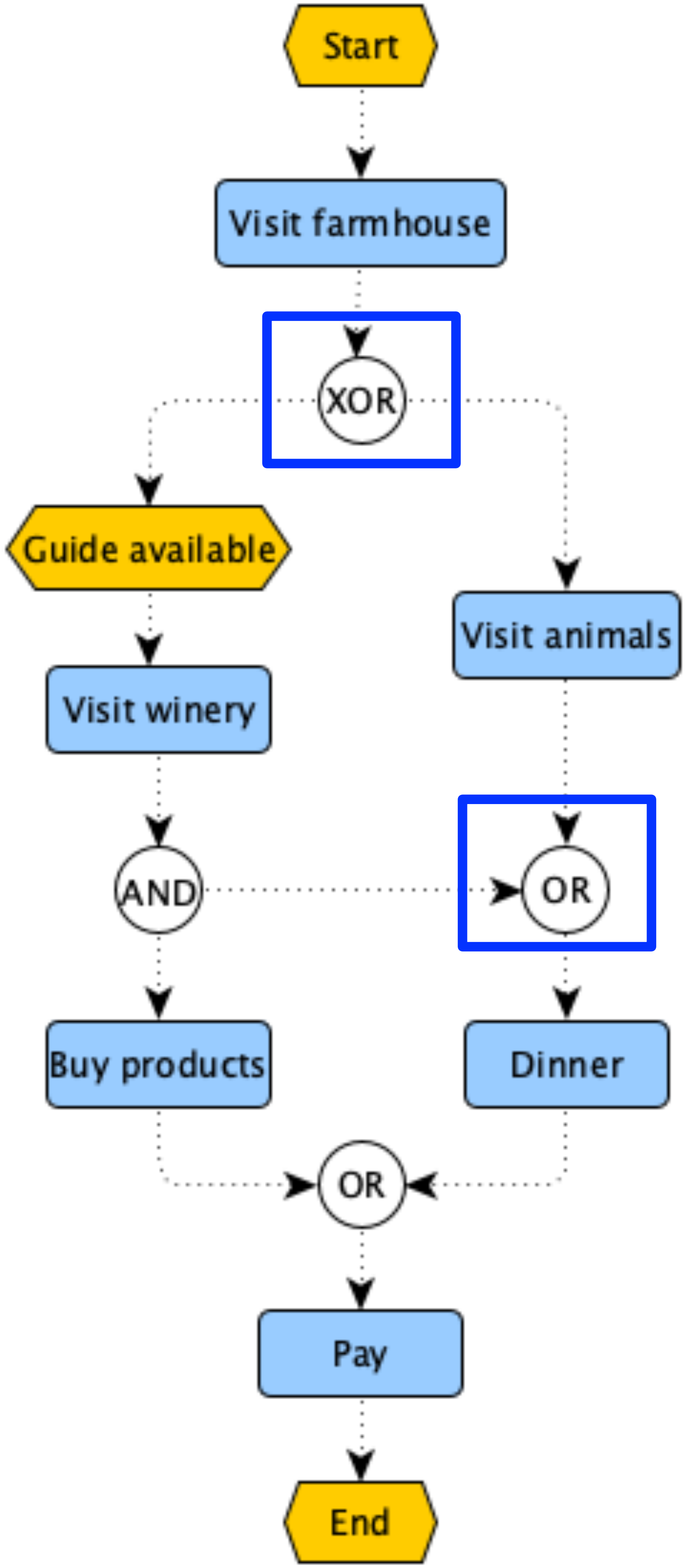
Assumption: every OR join is tagged with a policy (some suggested to have different trapezoid symbols)

Example



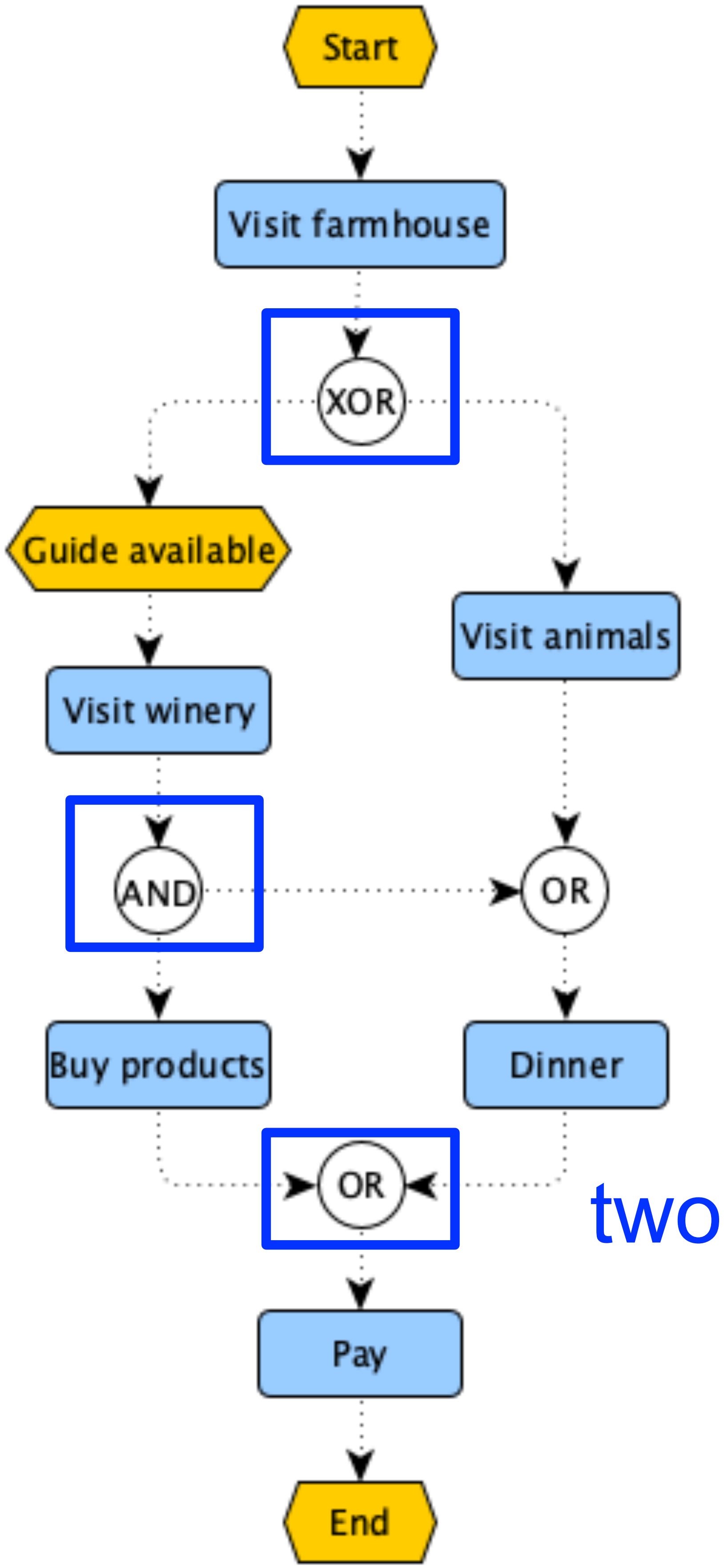
two OR joins
but no OR split

Example



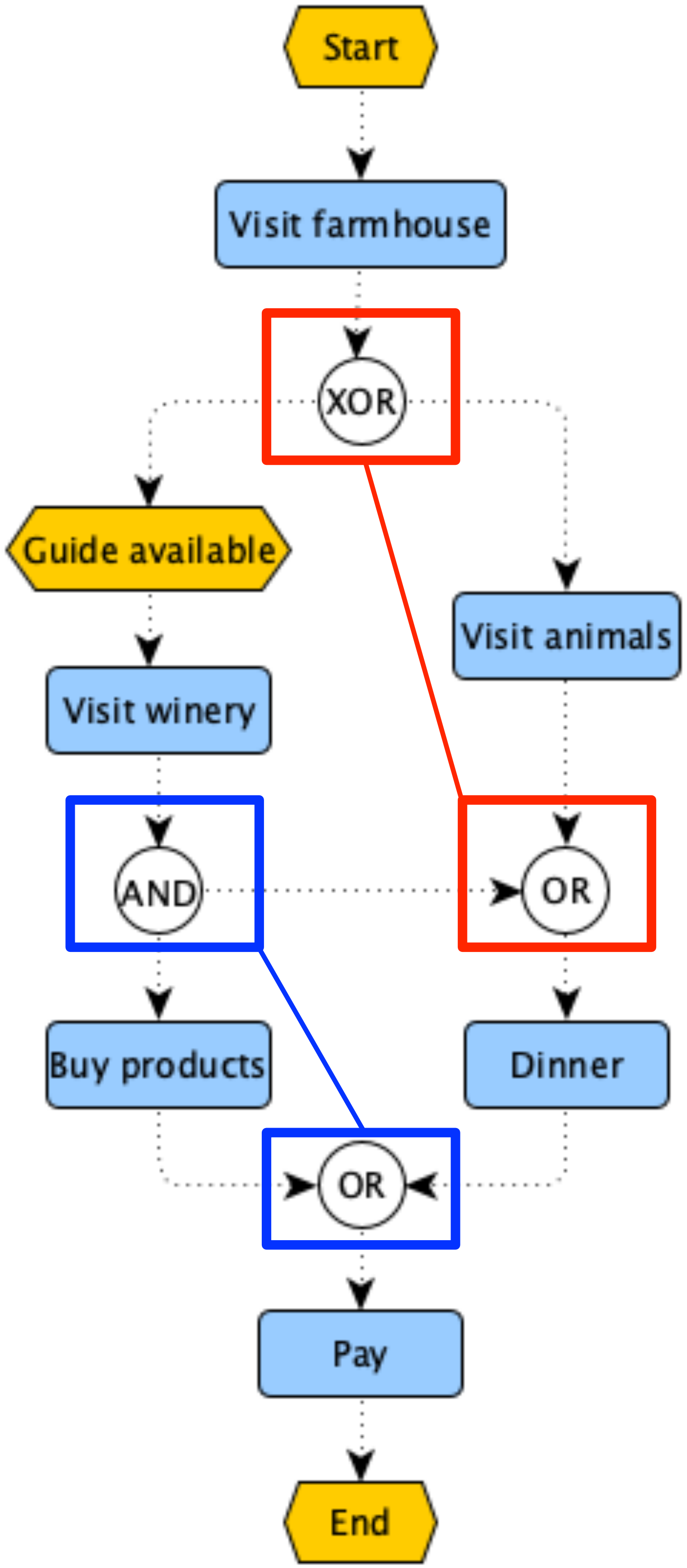
only one
candidate split

Example



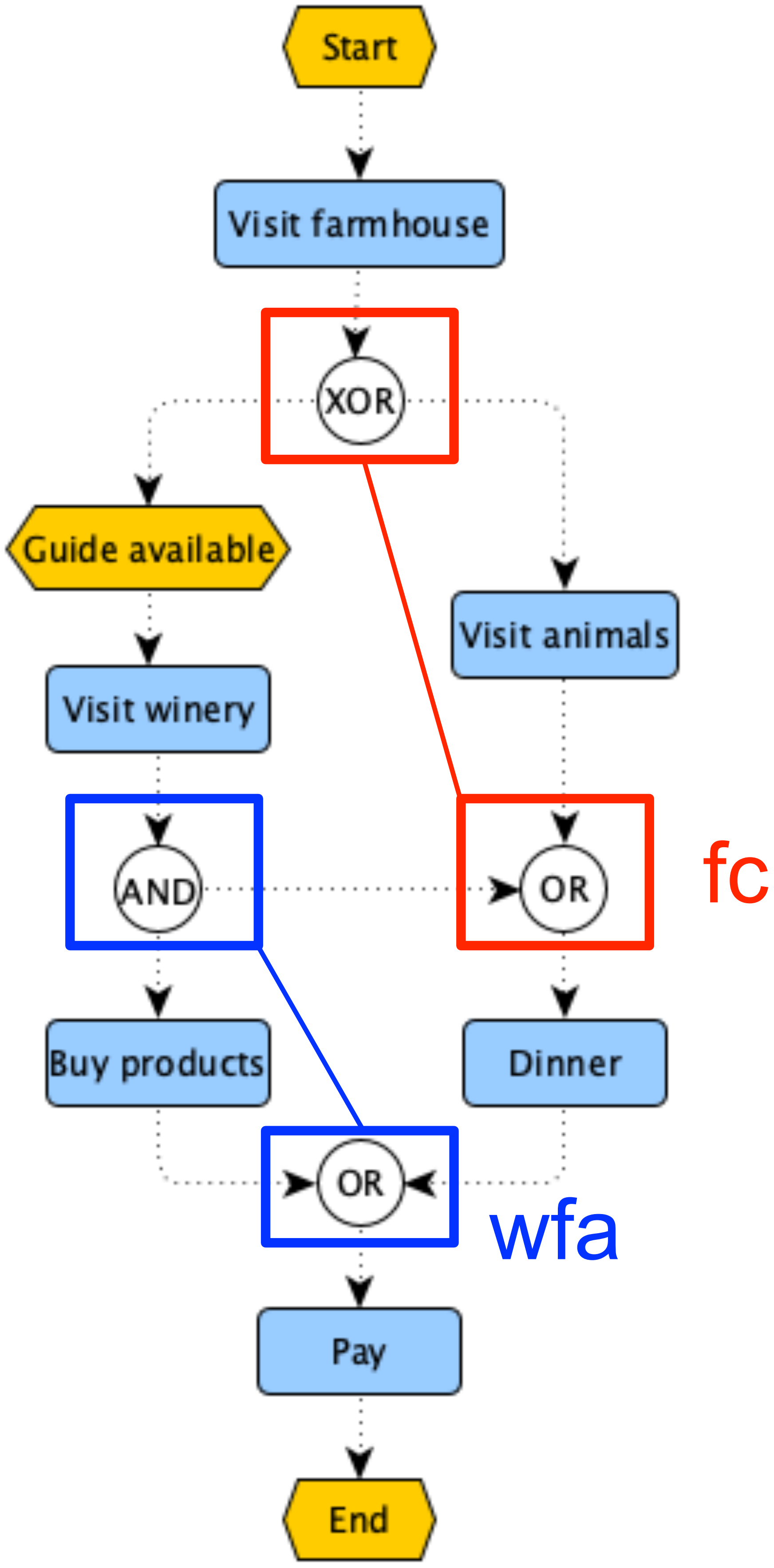
two candidate splits

Example



assign corresponding splits

Example



assign policies

Assumption

…

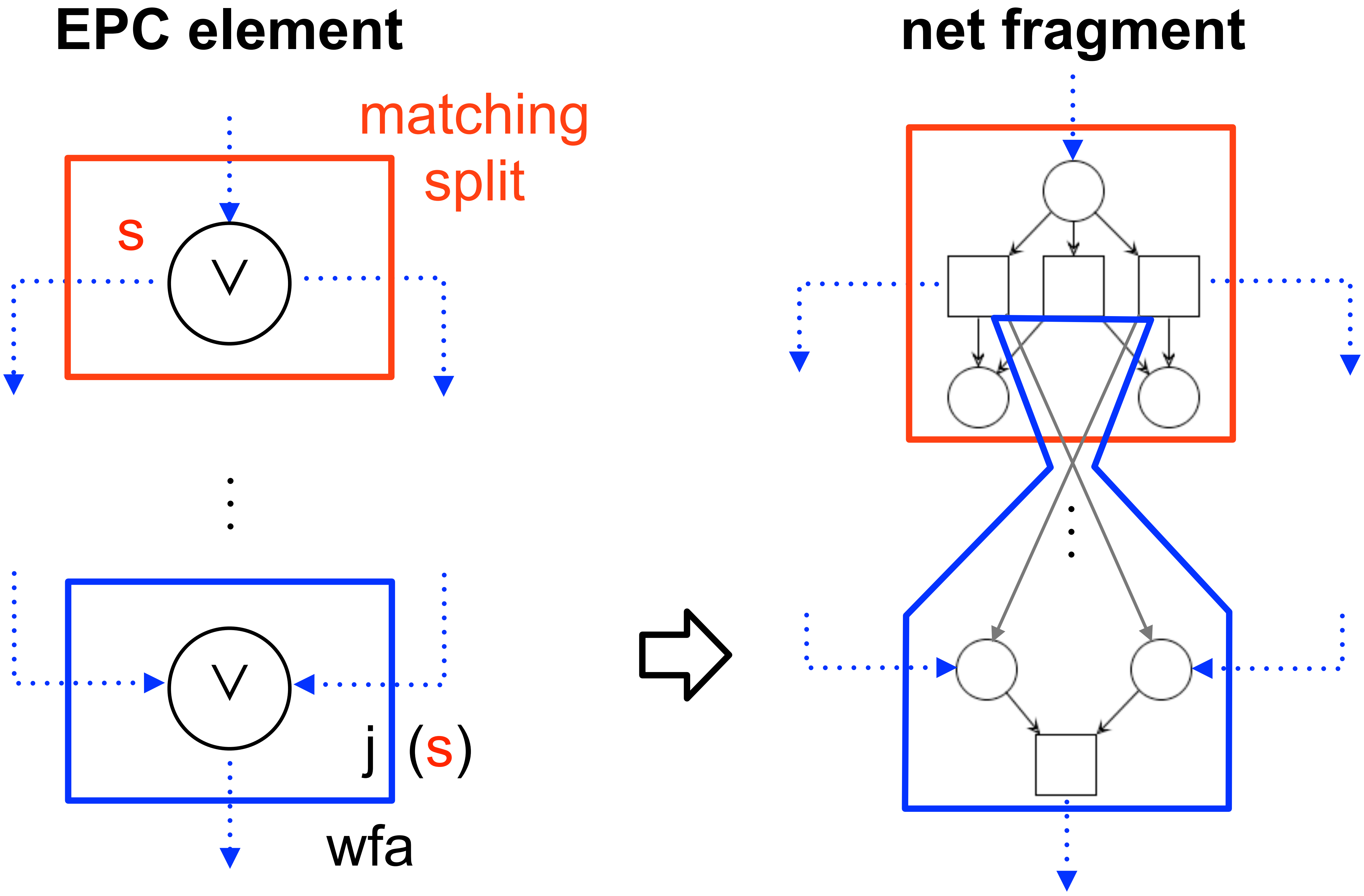
An OR join with **matching split uses wfa**

If an OR join has non-matching corresponding split
it is decorated with a policy (wfa, fc, et)

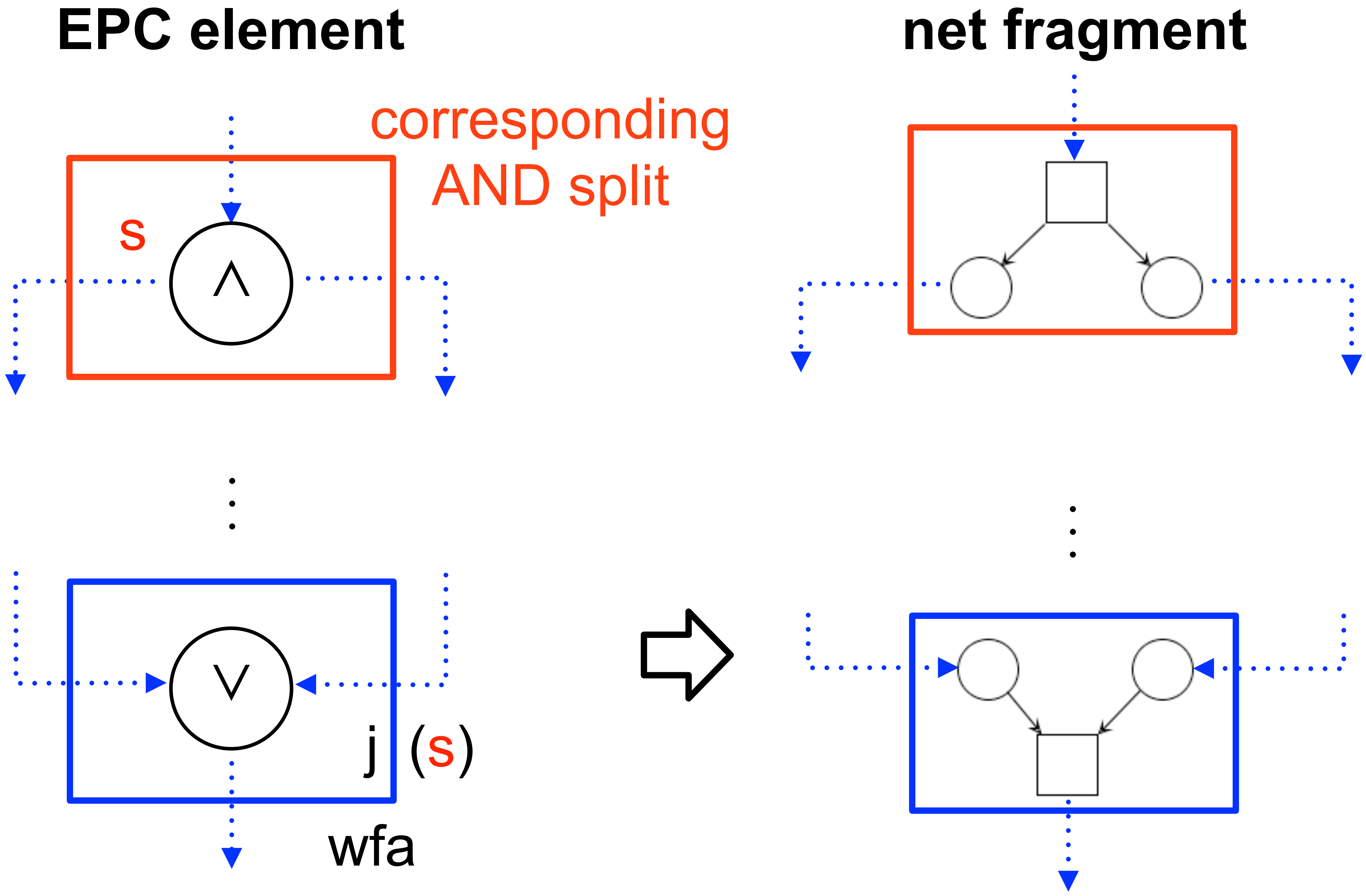
wfa: wait-for-all
works well with any corresponding split

…

Step 1: OR join (wfa)



Step 1: OR join (wfa)



EPC element

net fragment

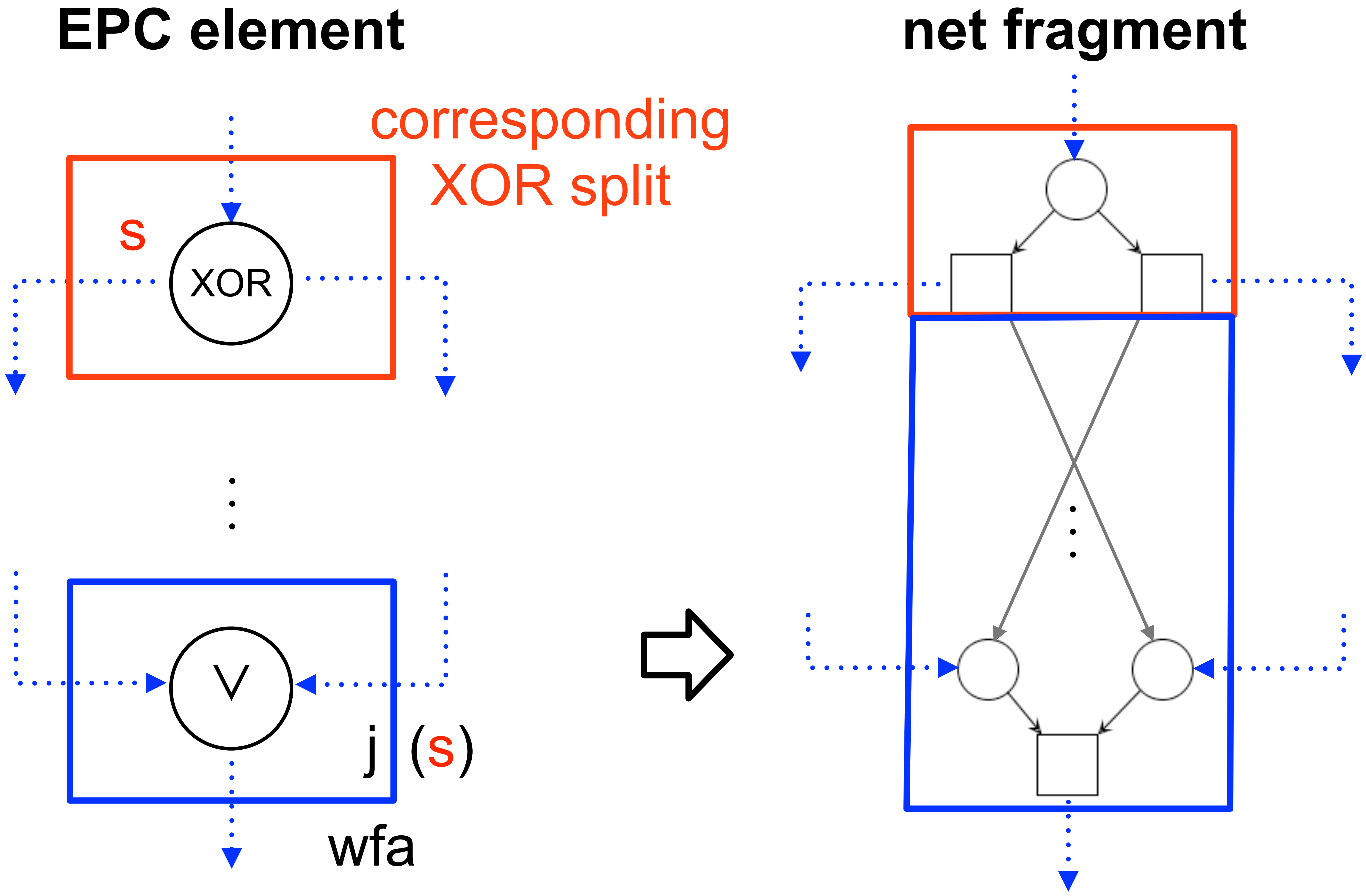
corresponding AND split

s

j (s)

wfa

Step 1: OR join (wfa)



Assumption

…

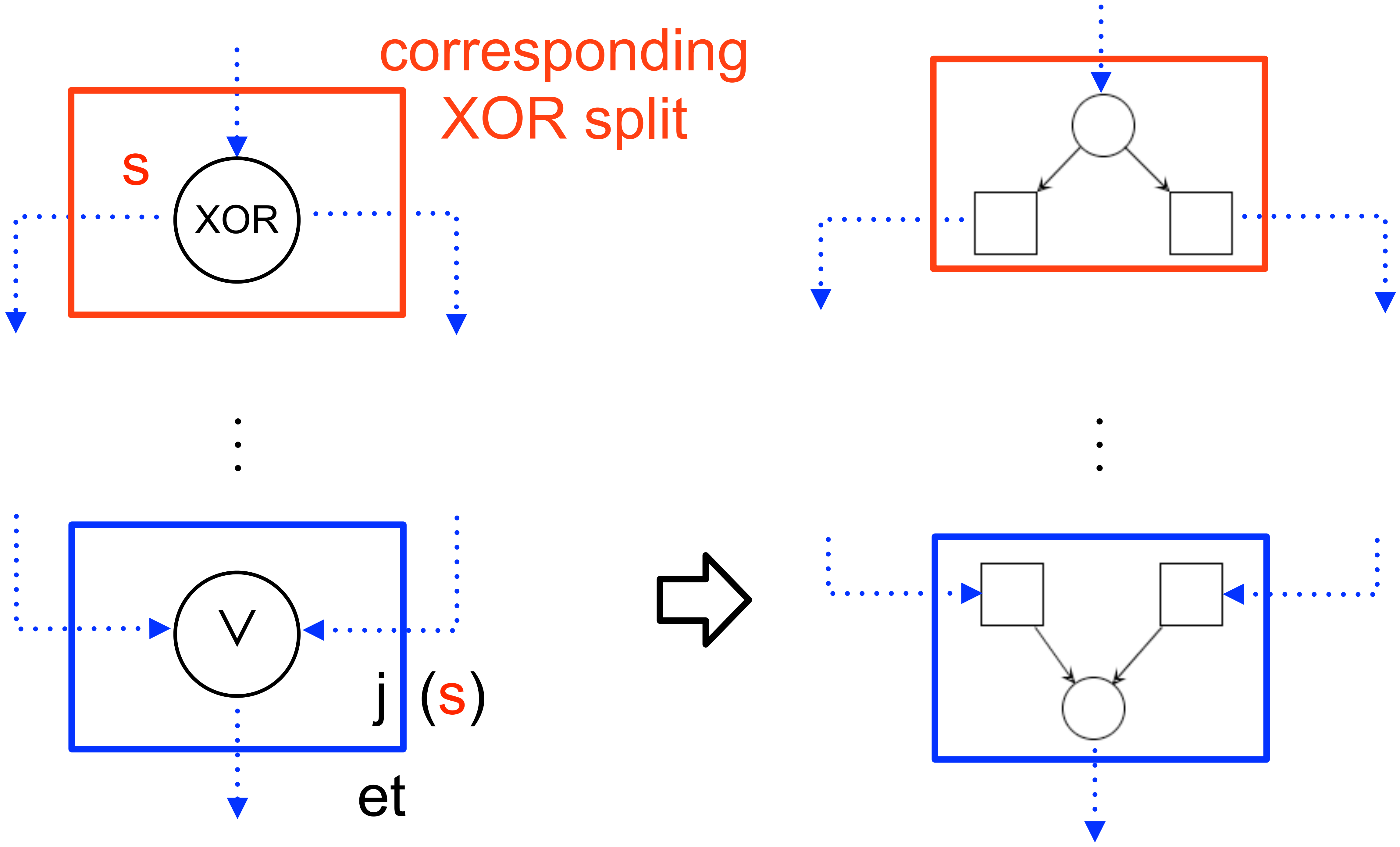
If an OR join has non-matching corresponding split
it is decorated with a policy (wfa, fc, et)

et: every-time
works well with corresponding XOR split

…

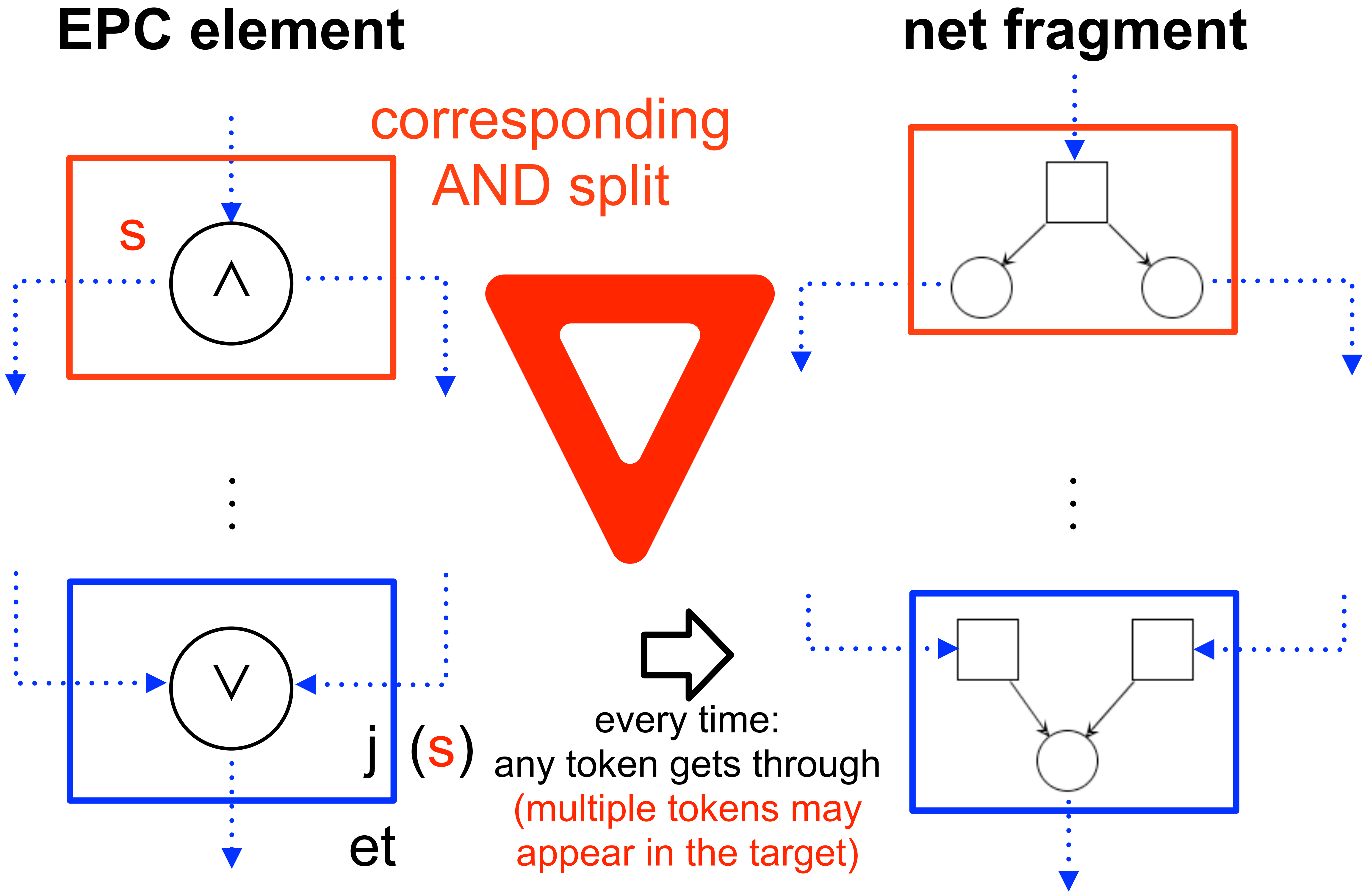
Step 1: OR join (et)

EPC element **net fragment**



corresponding XOR split

Step 1: OR join (et)



EPC element

net fragment

corresponding AND split

every time:
any token gets through
(multiple tokens may appear in the target)

Assumption

…

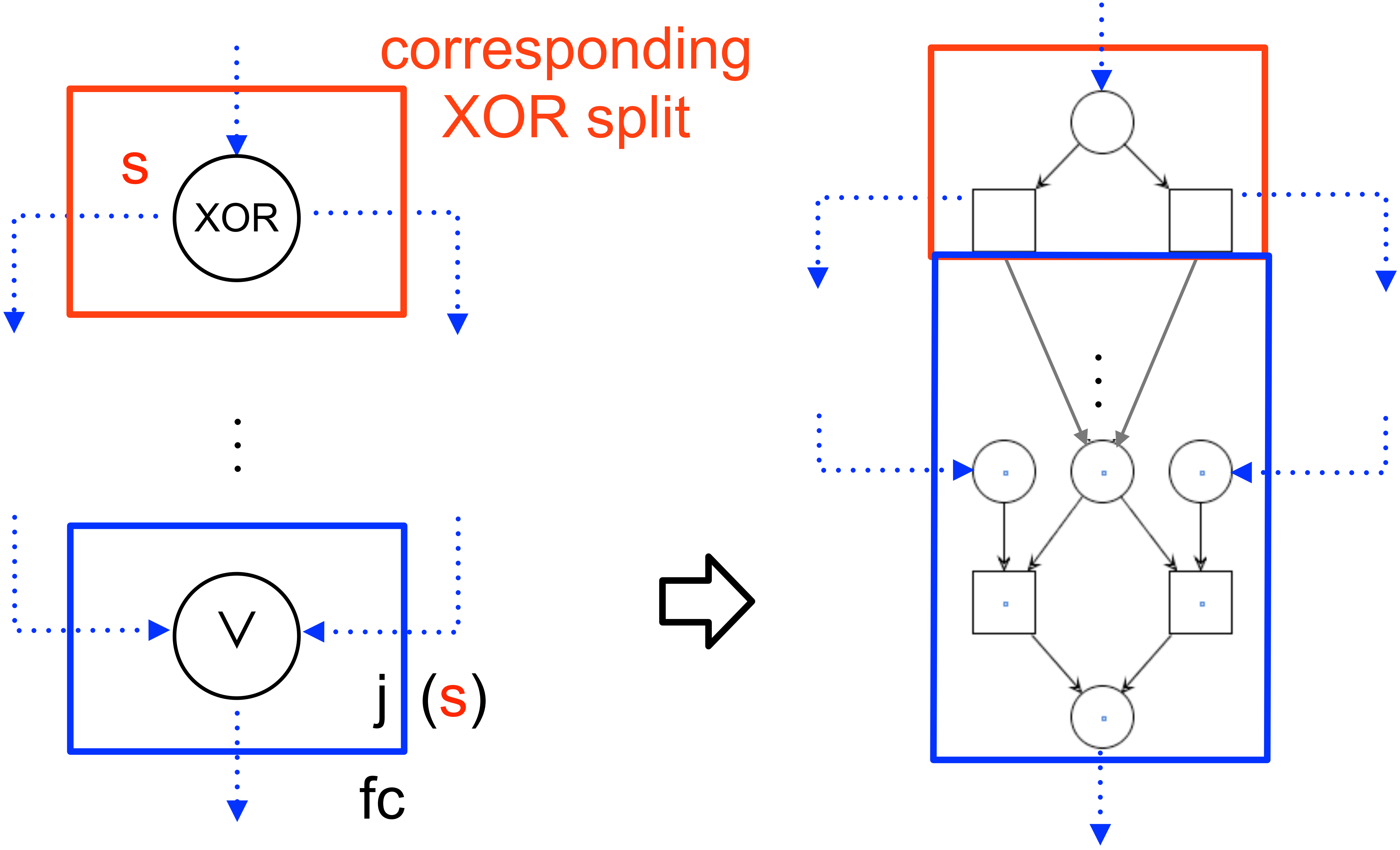
If an OR join has non-matching corresponding split
it is decorated with a policy (wfa, fc, et)

fc: first-come
works well with corresponding XOR split

…

Step 1: OR join (fc)

EPC element **net fragment**

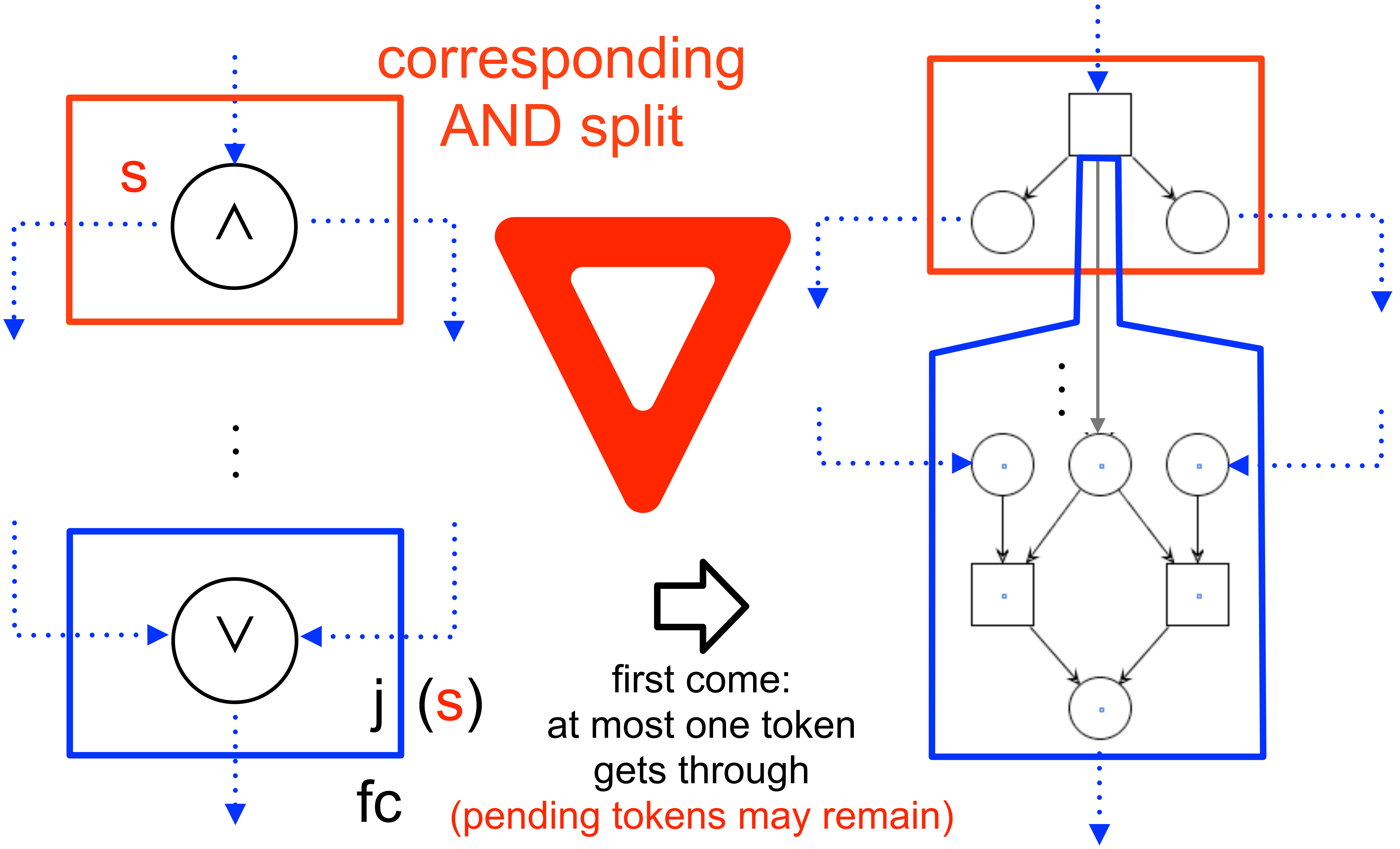


corresponding XOR split

Step 1: OR join (fc)

EPC element

net fragment



corresponding AND split

first come:
at most one token
gets through
(pending tokens may remain)

XOR join: assumption

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

Any policy (wait-for-all, first-come, every-time)
contradicts the exclusivity of XOR
(a token from one path can be accepted only if we make sure that no second token will arrive via the other path)

Assumption: every XOR join has a matching split
(the implicit start split is allowed as a valid match)

Assumption

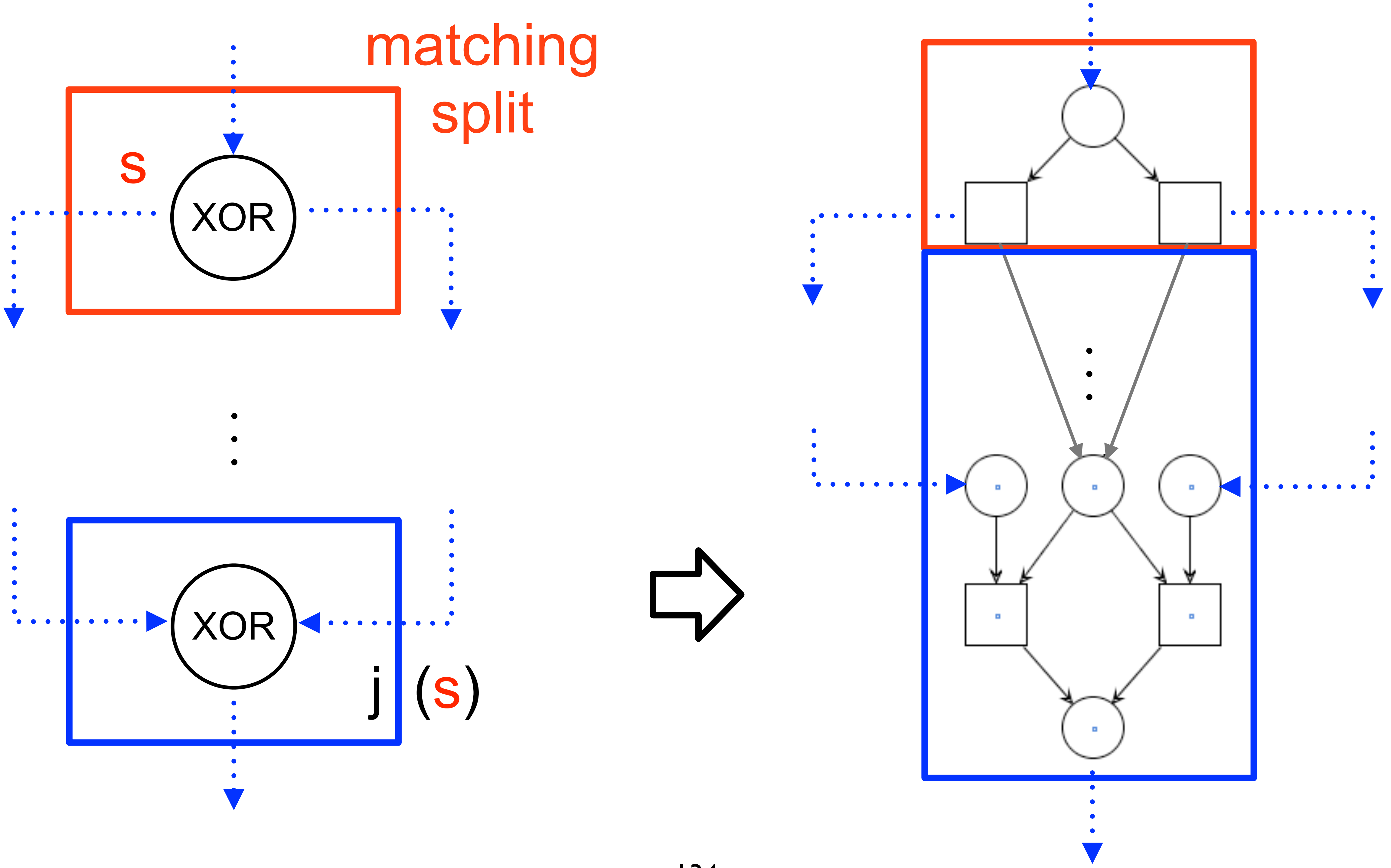
…

Any XOR join has a **corresponding matching split**

…

Step 1: XOR join

EPC element **net fragment**



matching split

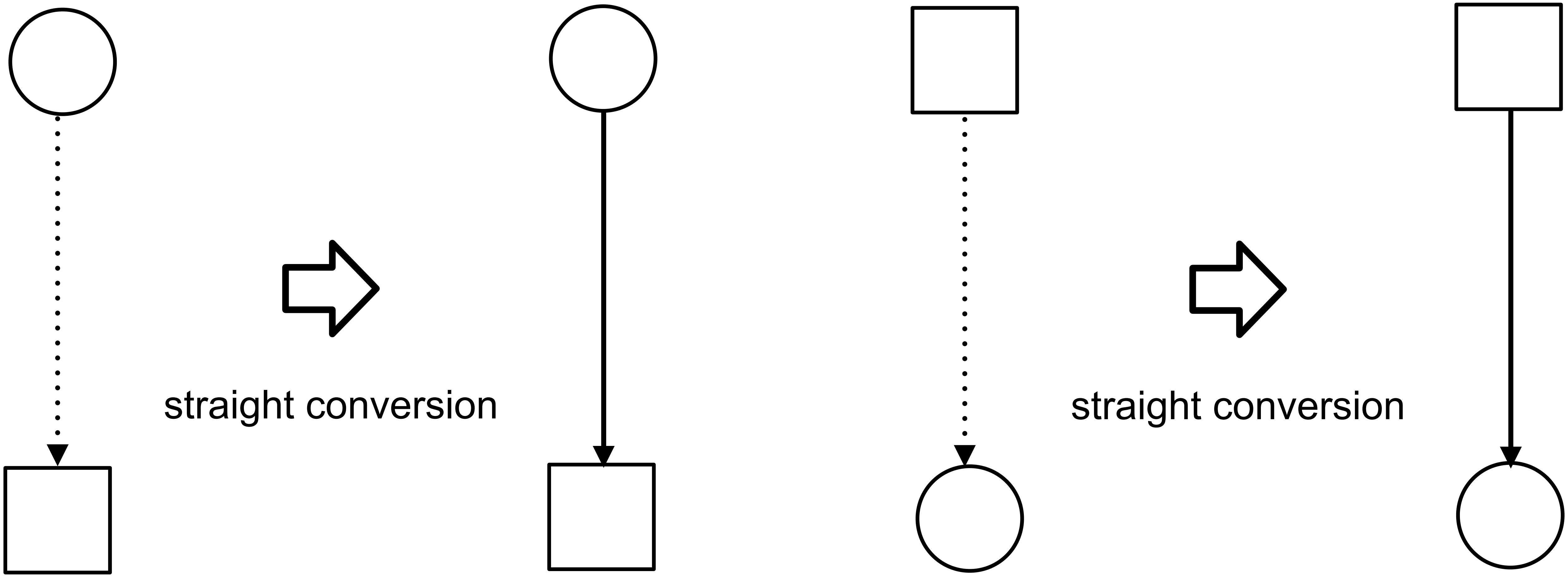
s

XOR

XOR

j (s)

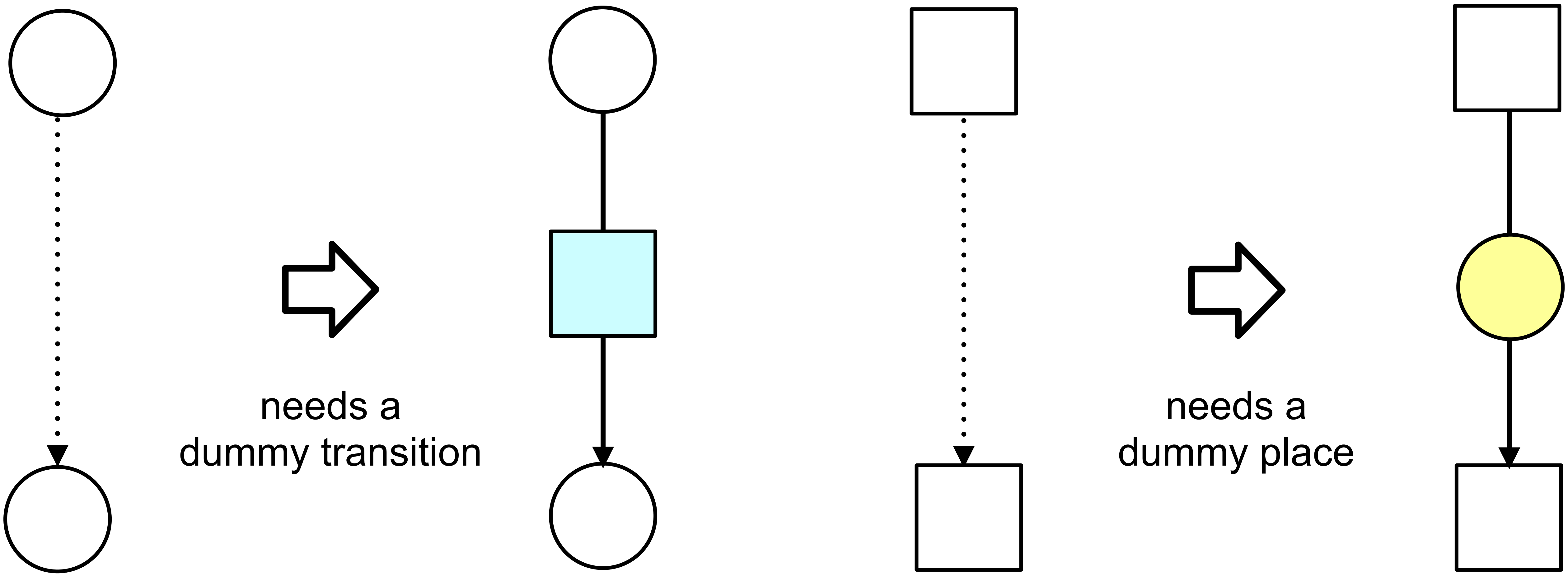
Step 2: dummy style



straight conversion

straight conversion

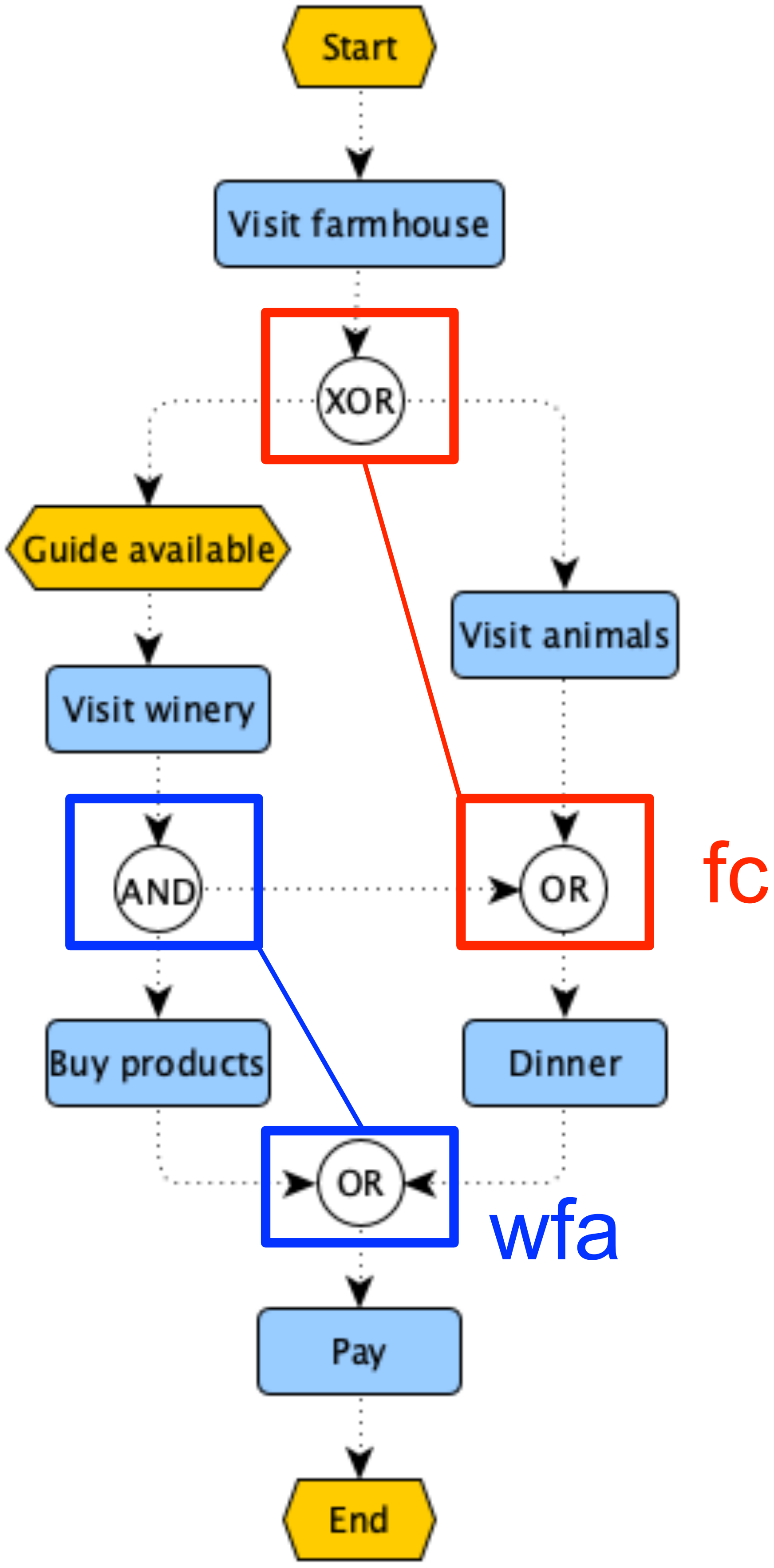
Step 2: dummy style



needs a
dummy transition

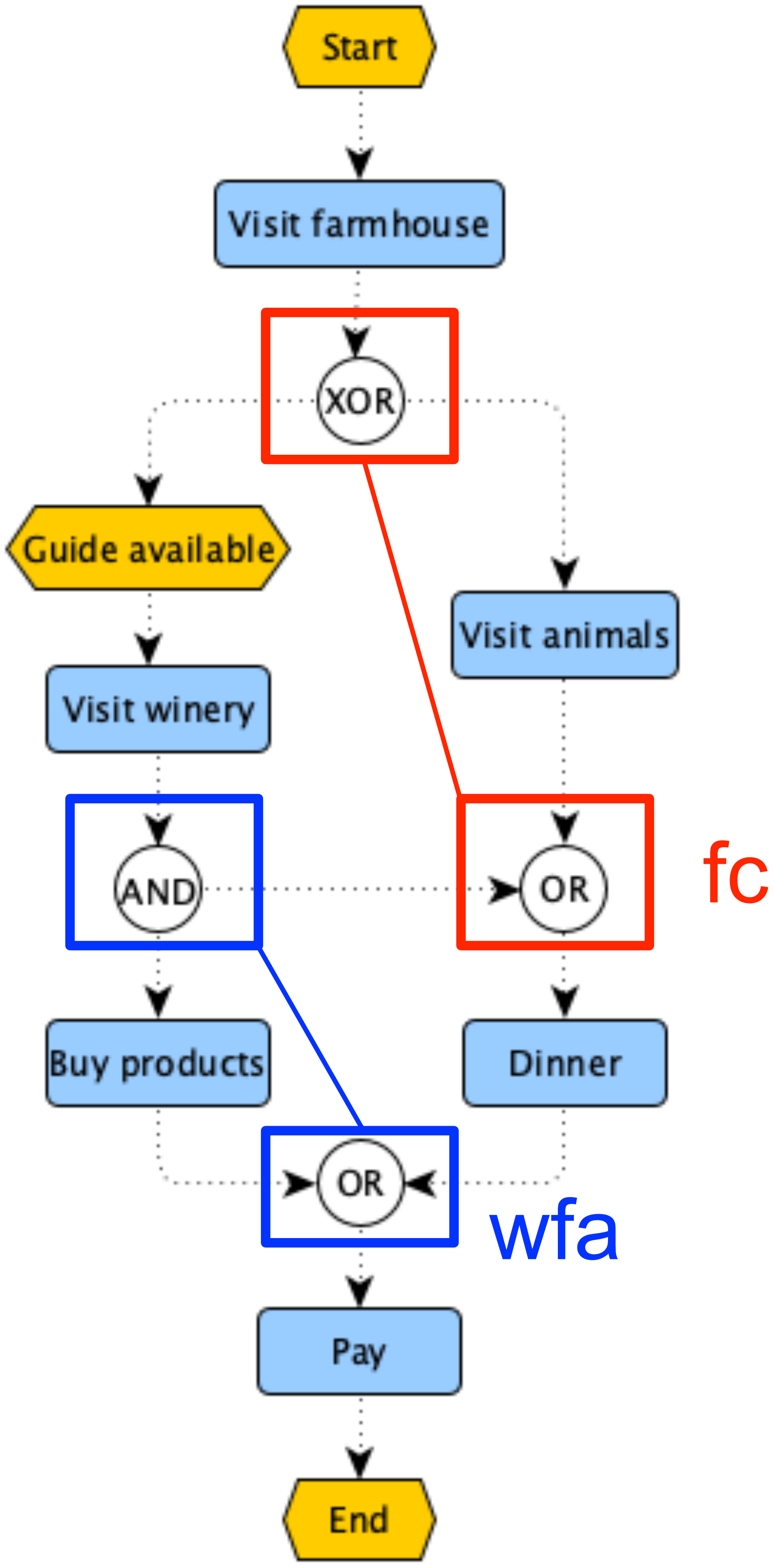
needs a
dummy place

Example

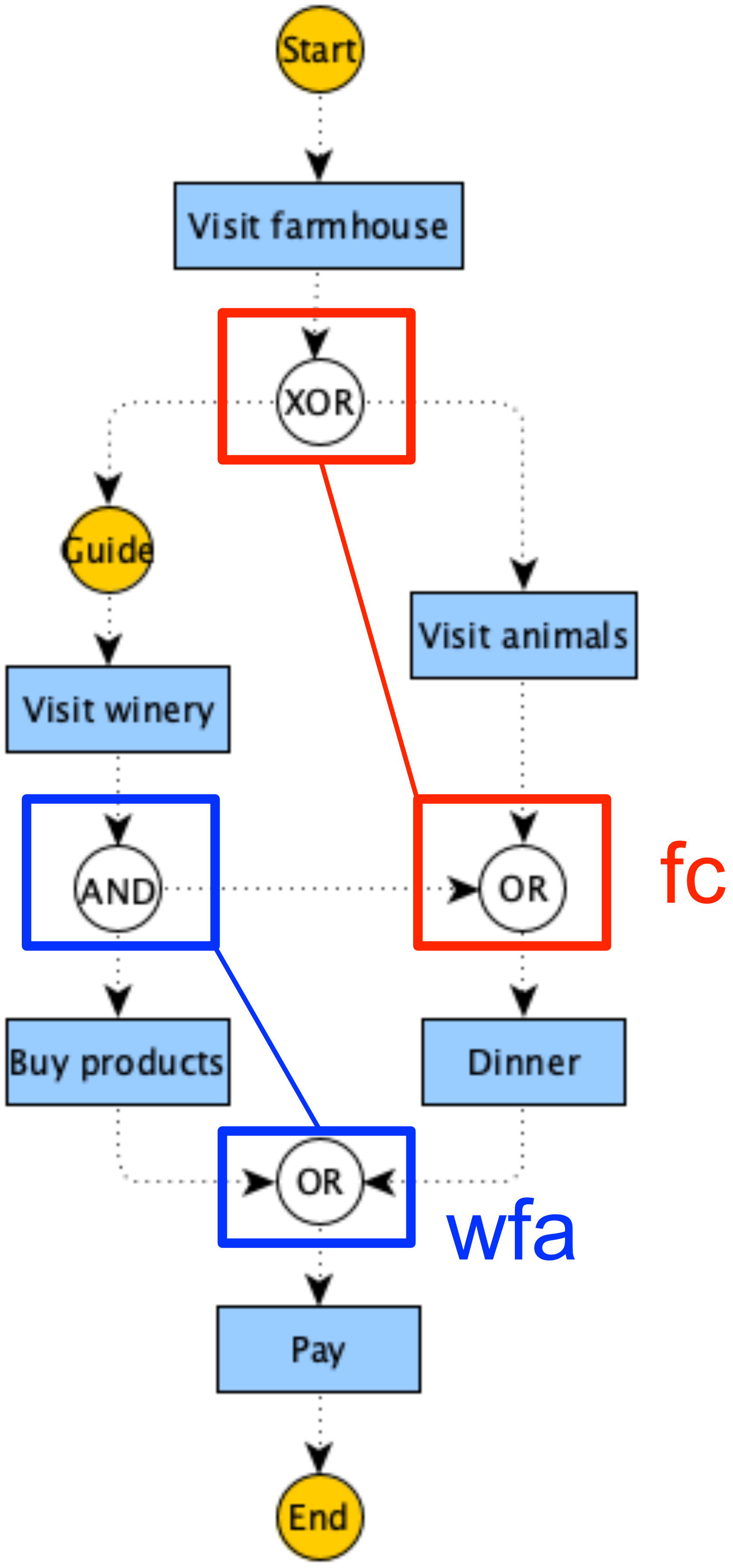


Sound?

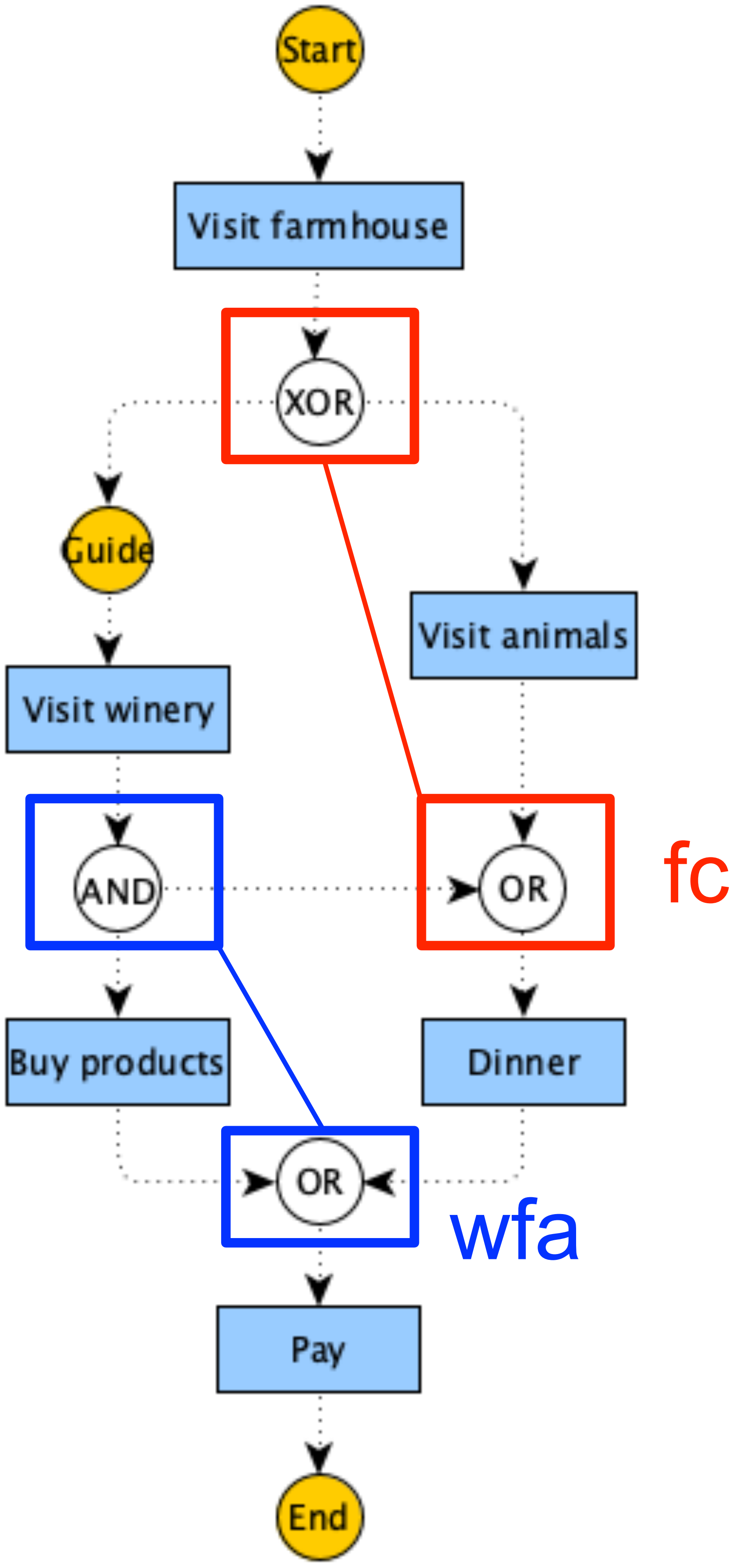
Example



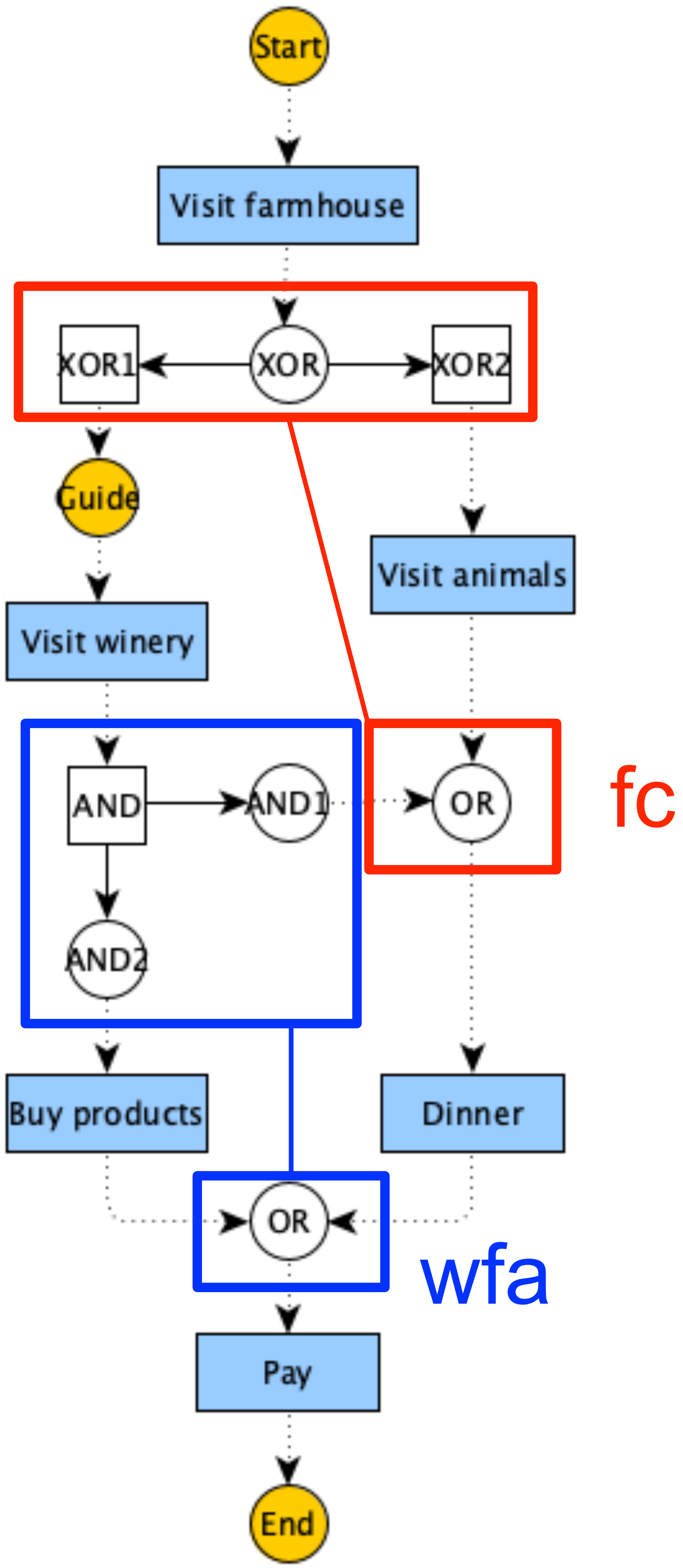
Step 1
events and
functions



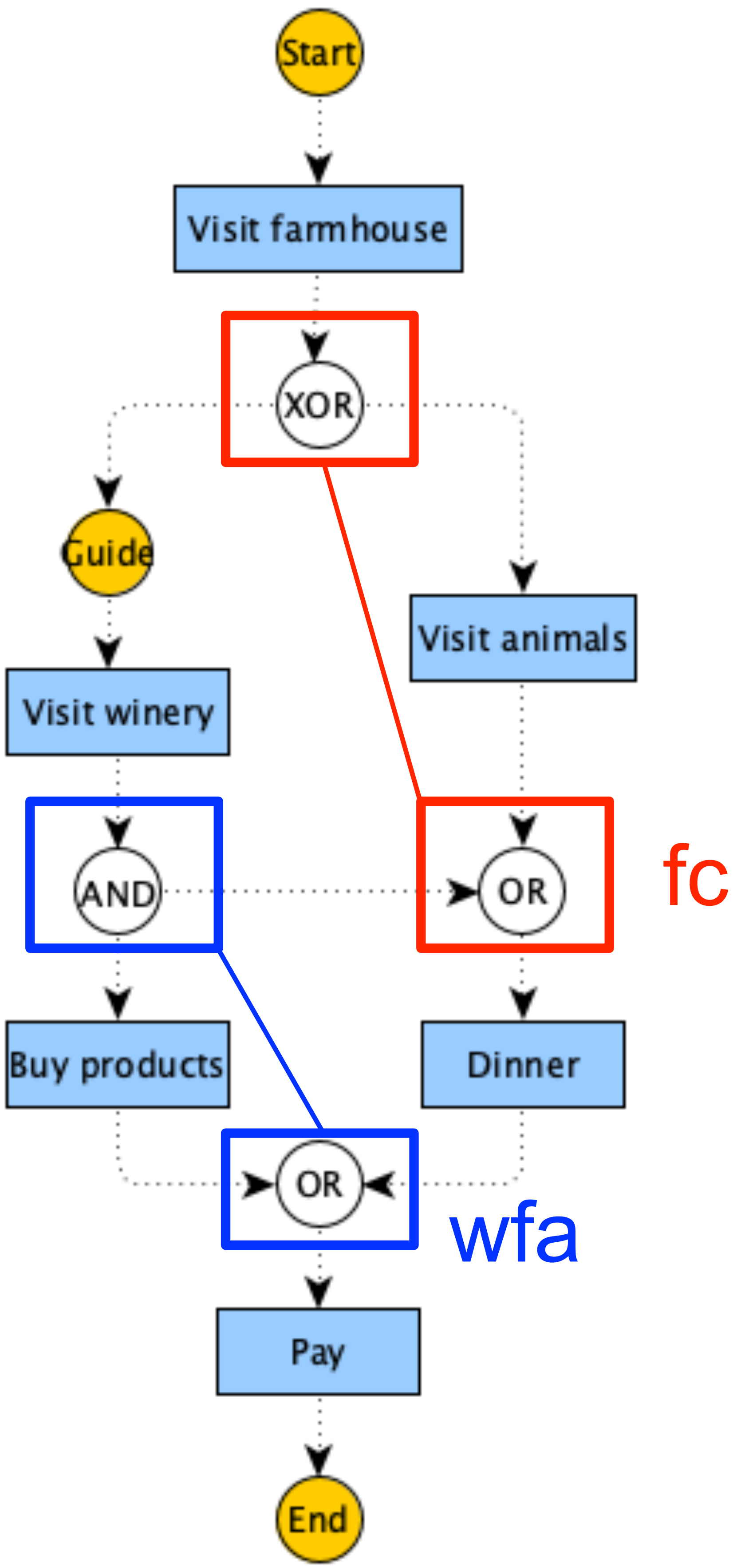
Example



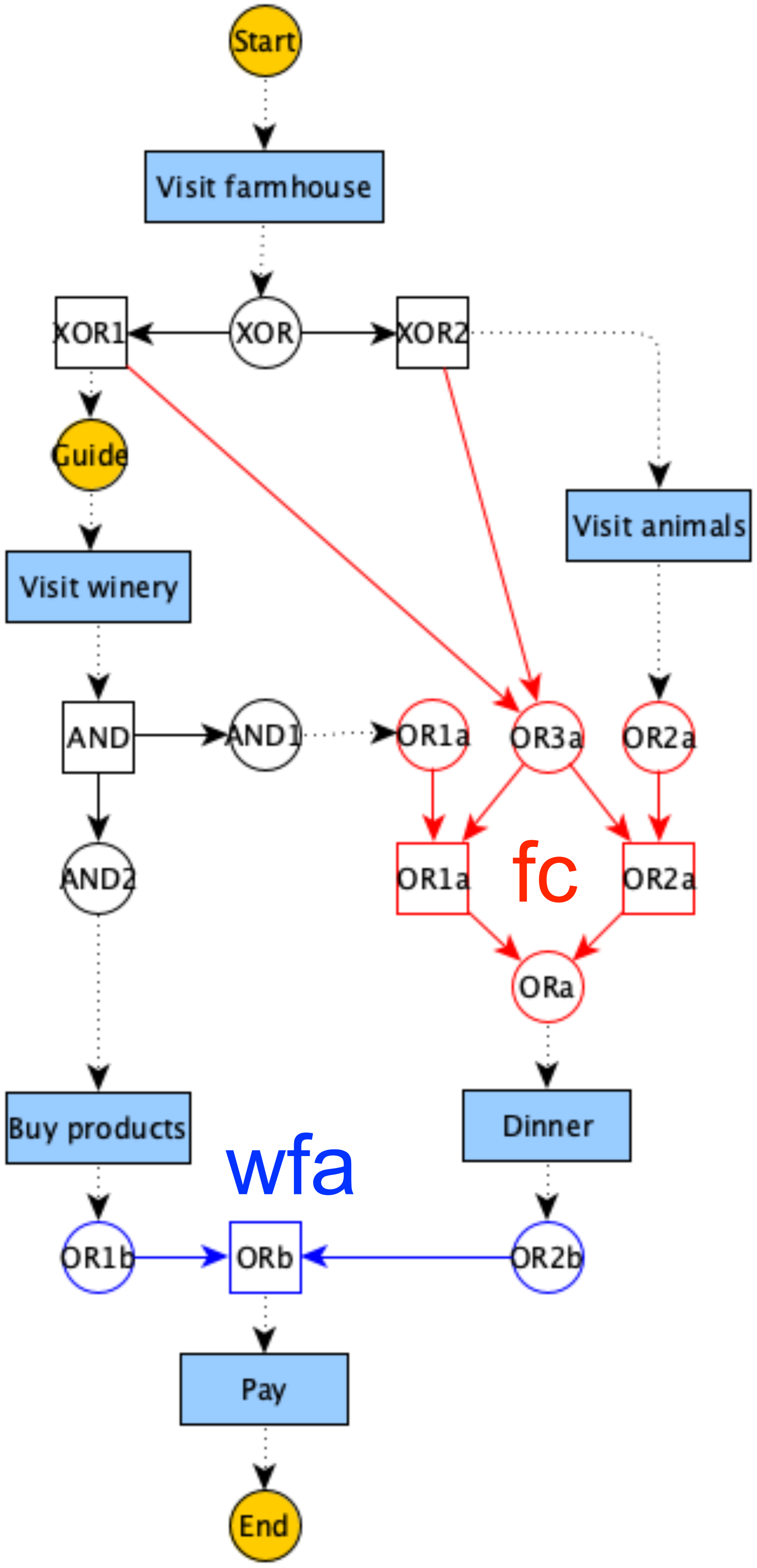
Step 1
splits



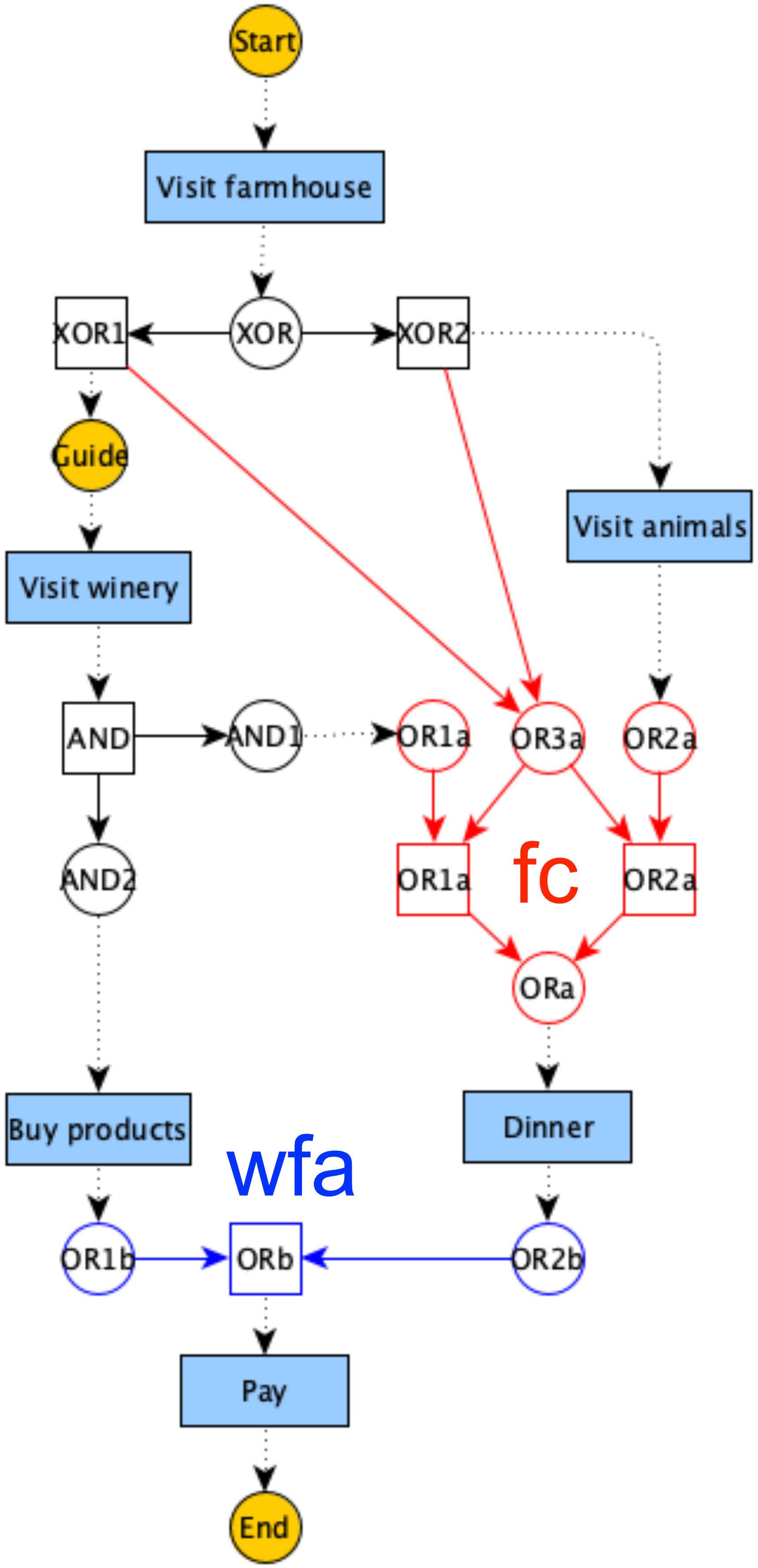
Example



Step 1 splits and joins

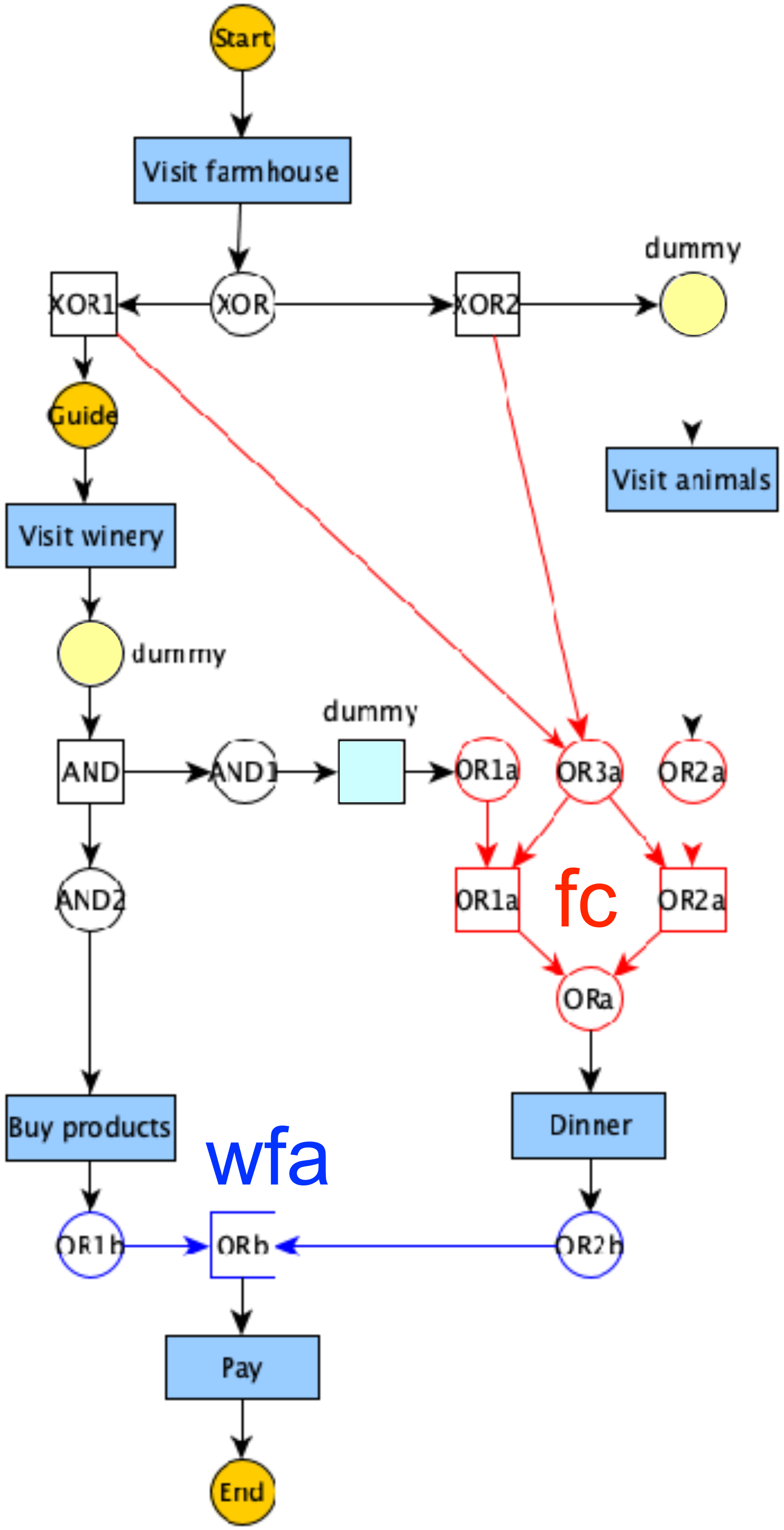


Example

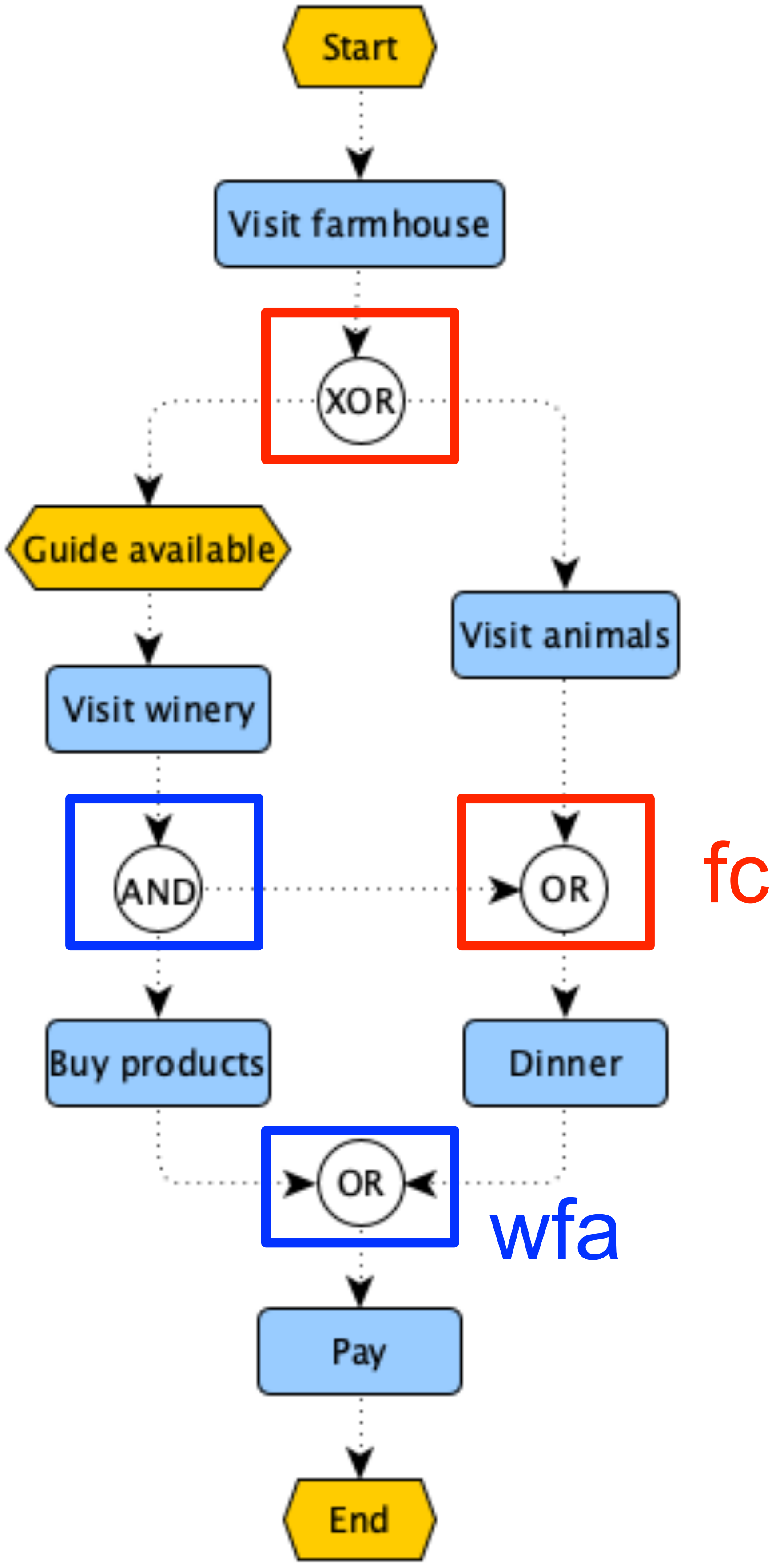


⇨

Step 2(+3)
dummy style



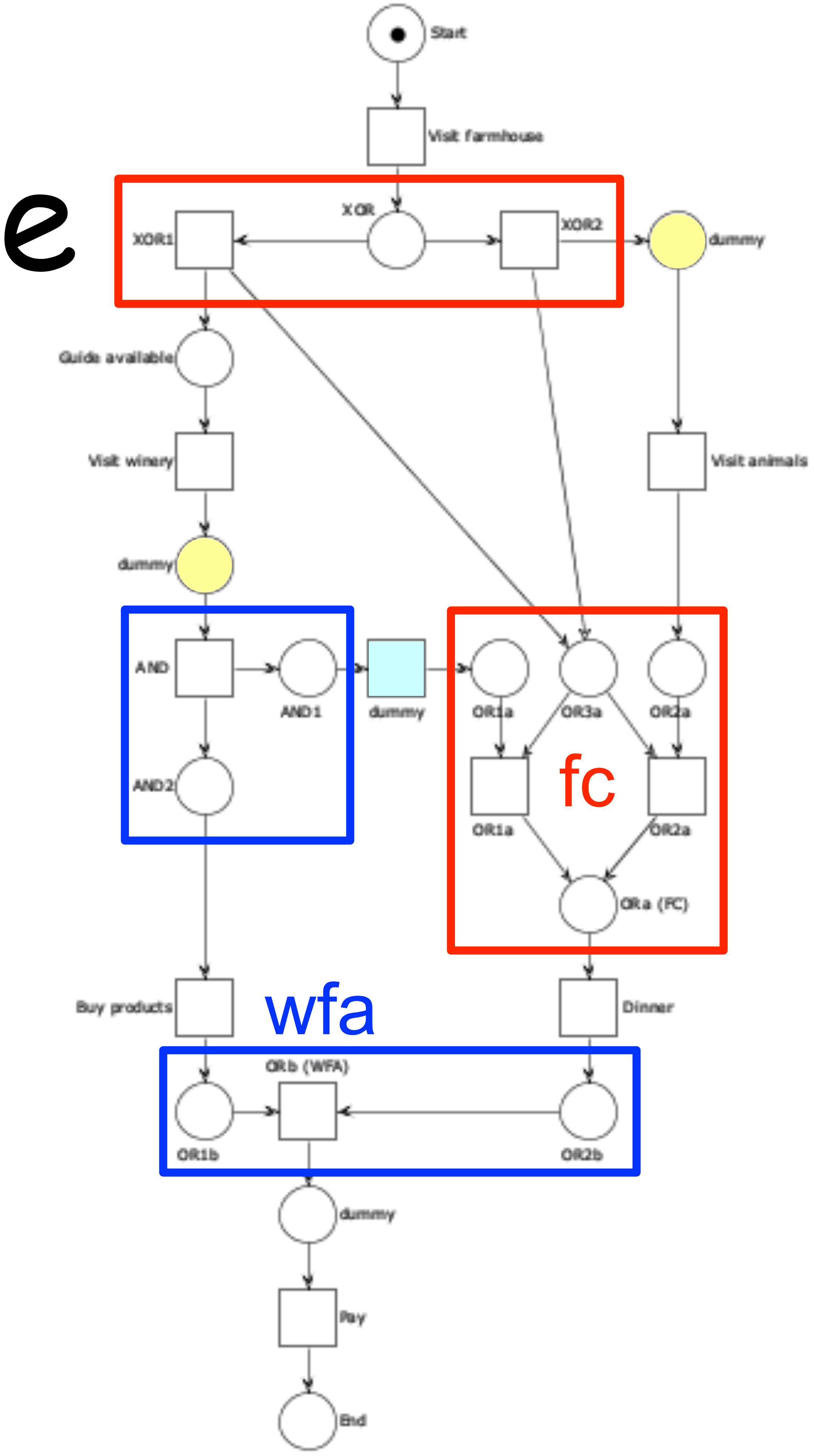
Example



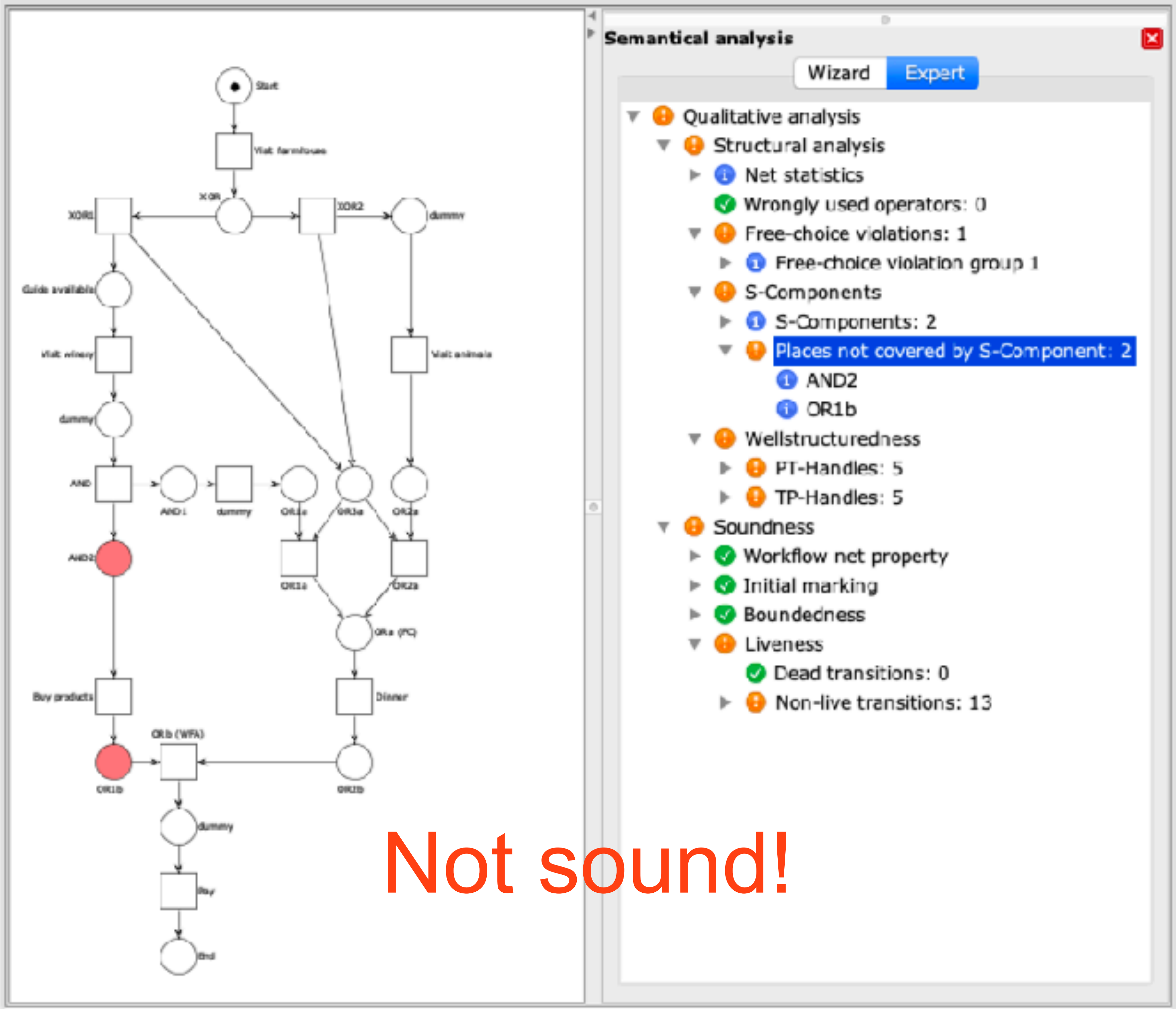
Sound?

⇨

Steps
1+2(+3)



Example



Semantical analysis

- Qualitative analysis
  - Structural analysis
    - Net statistics
    - ✓ Wrongly used operators: 0
    - Free-choice violations: 1
      - Free-choice violation group 1
    - S-Components
      - S-Components: 2
      - Places not covered by S-Component: 2
        - AND2
        - OR1b
    - Wellstructuredness
      - PT-Handles: 5
      - TP-Handles: 5
  - Soundness
    - ✓ Workflow net property
    - ✓ Initial marking
    - ✓ Boundedness
    - Liveness
      - ✓ Dead transitions: 0
      - Non-live transitions: 13

Not sound!

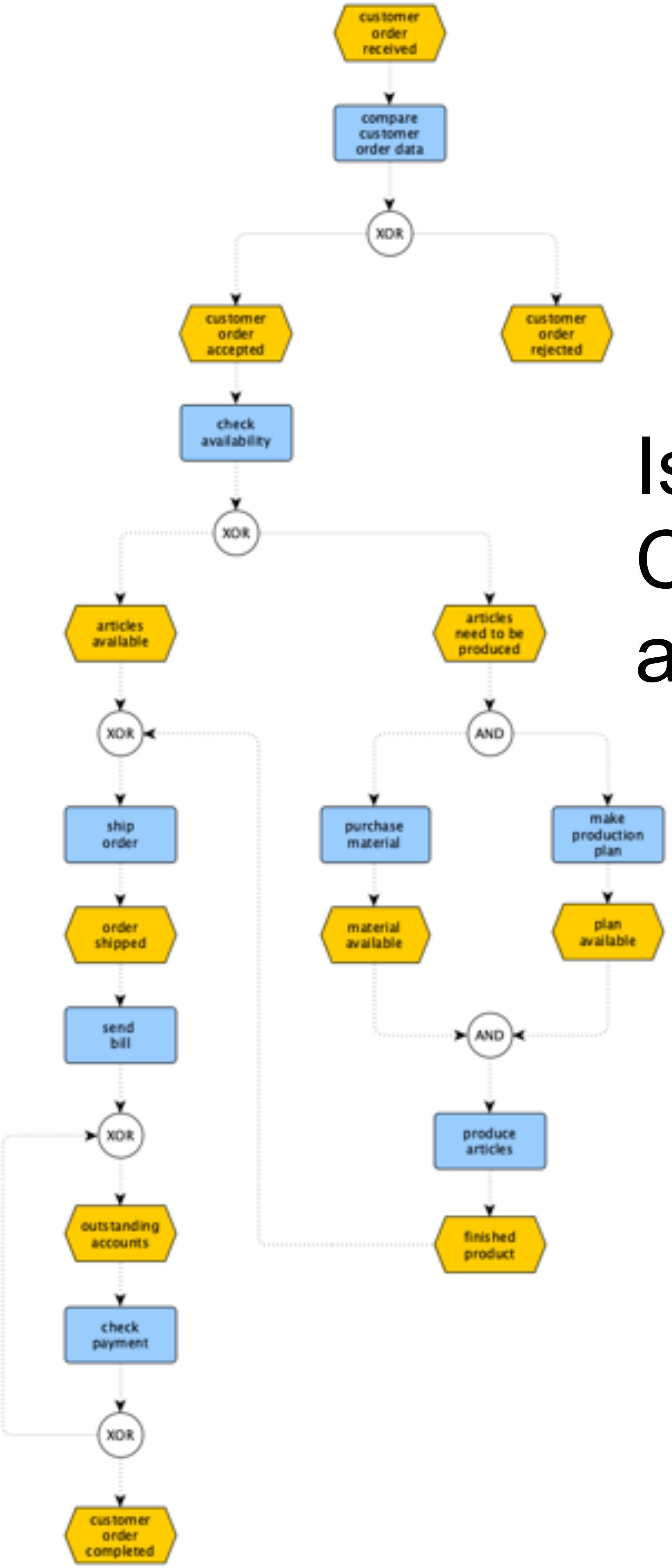
EPC pros and cons

You may **leave complete freedom**,
but most diagrams will not be sound

You may **constrain diagrams**,
but people like flexible syntax and ignore guidelines

You may **require to add decorations**,
but people will be lazy or misinterpret policies

Exercise



Is this EPC diagram sound?
Choose one of the three techniques seen and apply it to answer the above question