

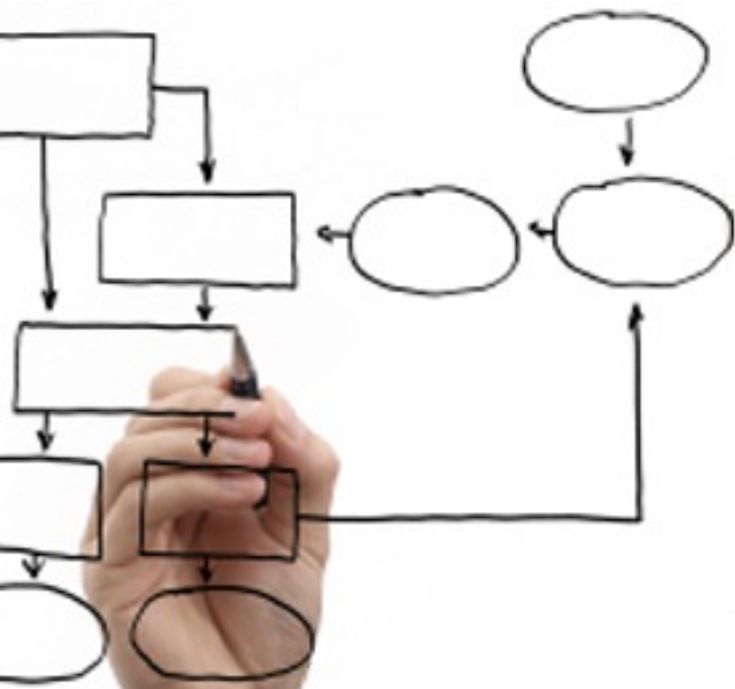
Business Processes Modelling

MPB (6 cfu, 295AA)

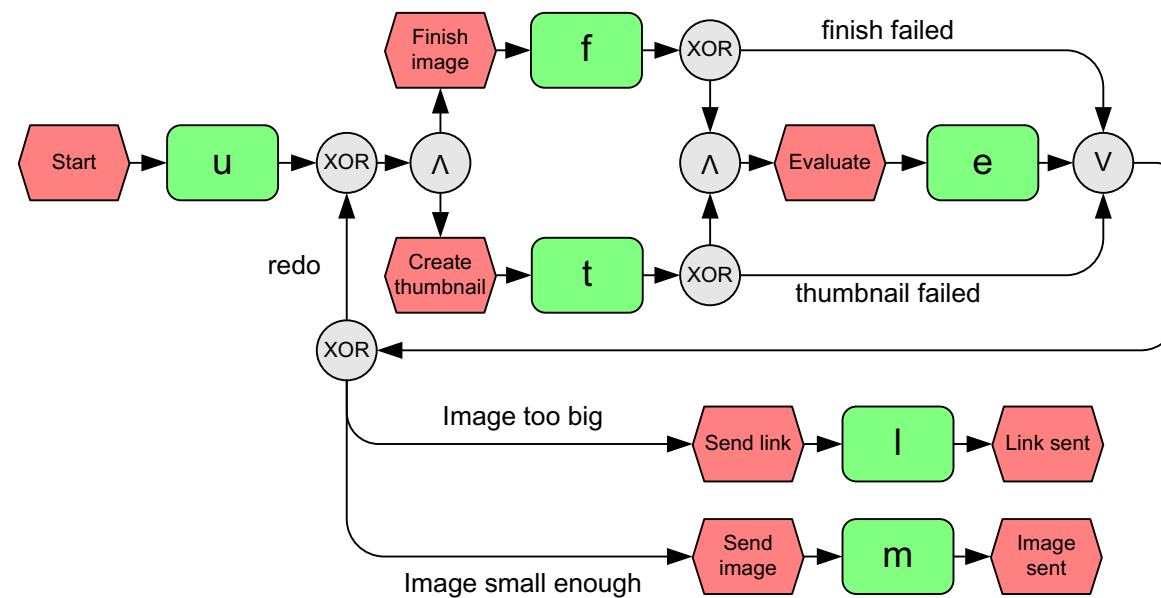
Roberto Bruni

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

16a - EPC analysis



Object



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

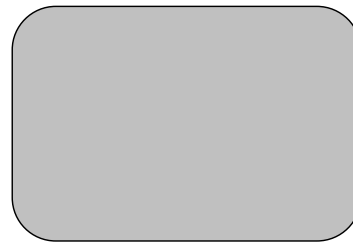
EPC Diagrams

EPC ingredients at a glance

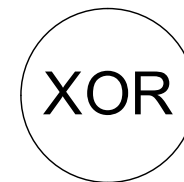
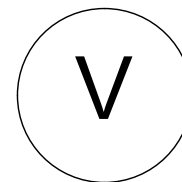
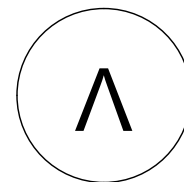
Event



Function



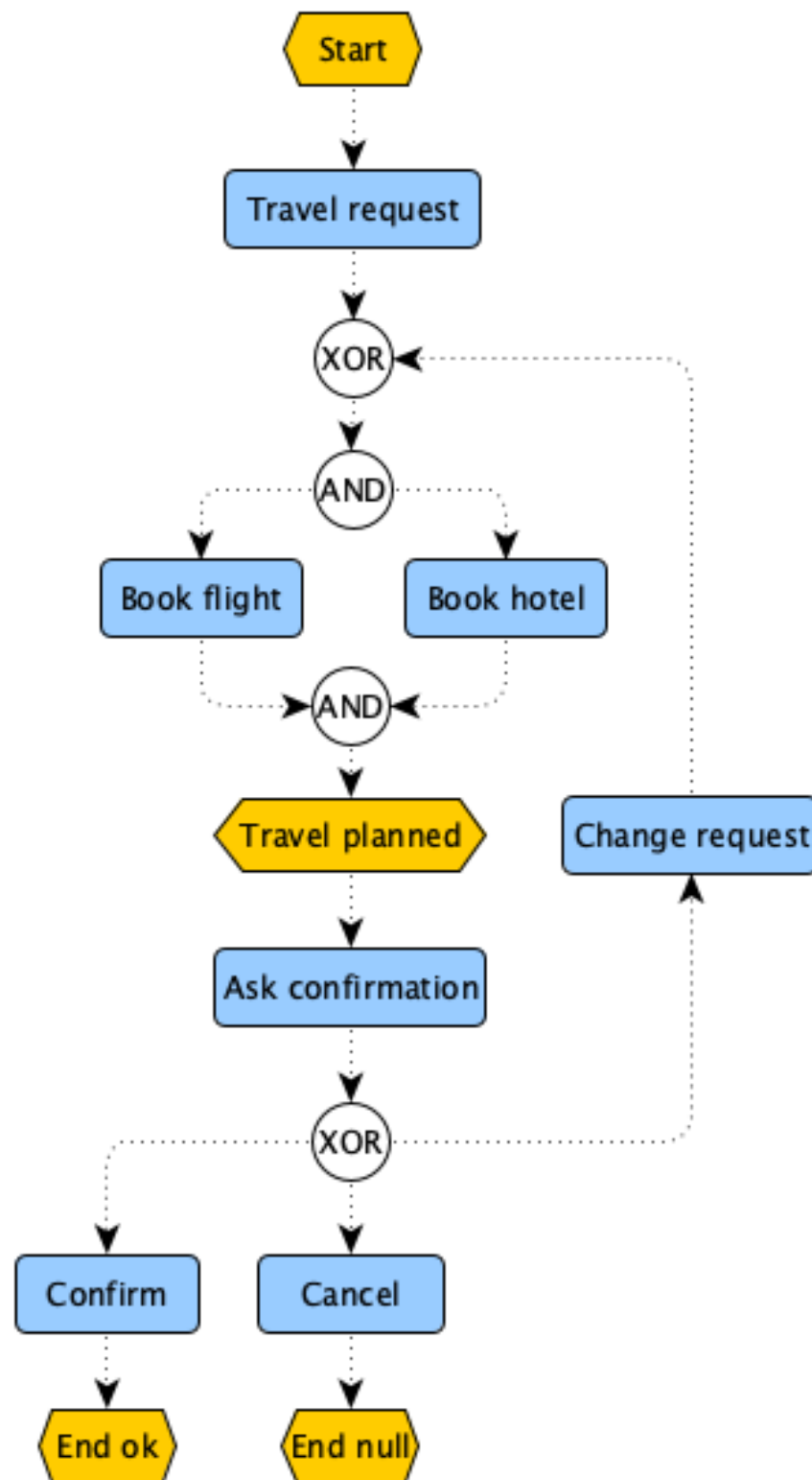
Connectors



Control Flow



EPC: Example



EPC Semantics

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

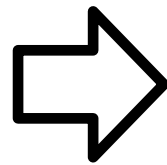
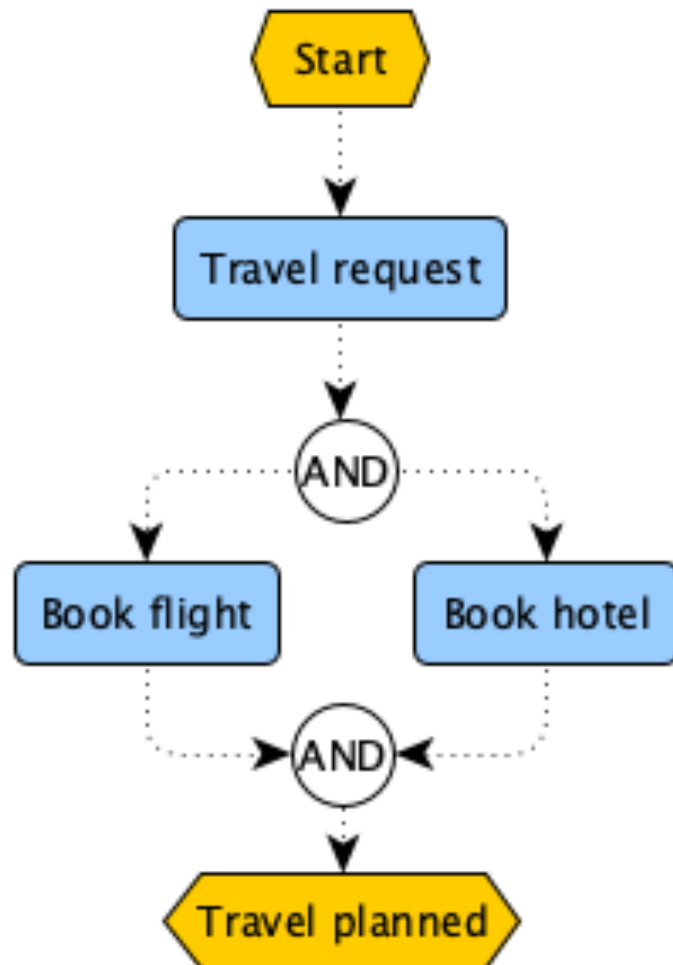
We can reuse the verification tools
to check if the net is sound

Is there a unique way to proceed? Not necessarily!

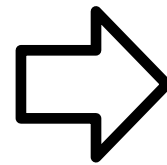
Translation of EPC to Petri nets

The idea

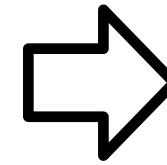
From EPC to wf nets in three steps



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



Step 2
connect
fragments
together

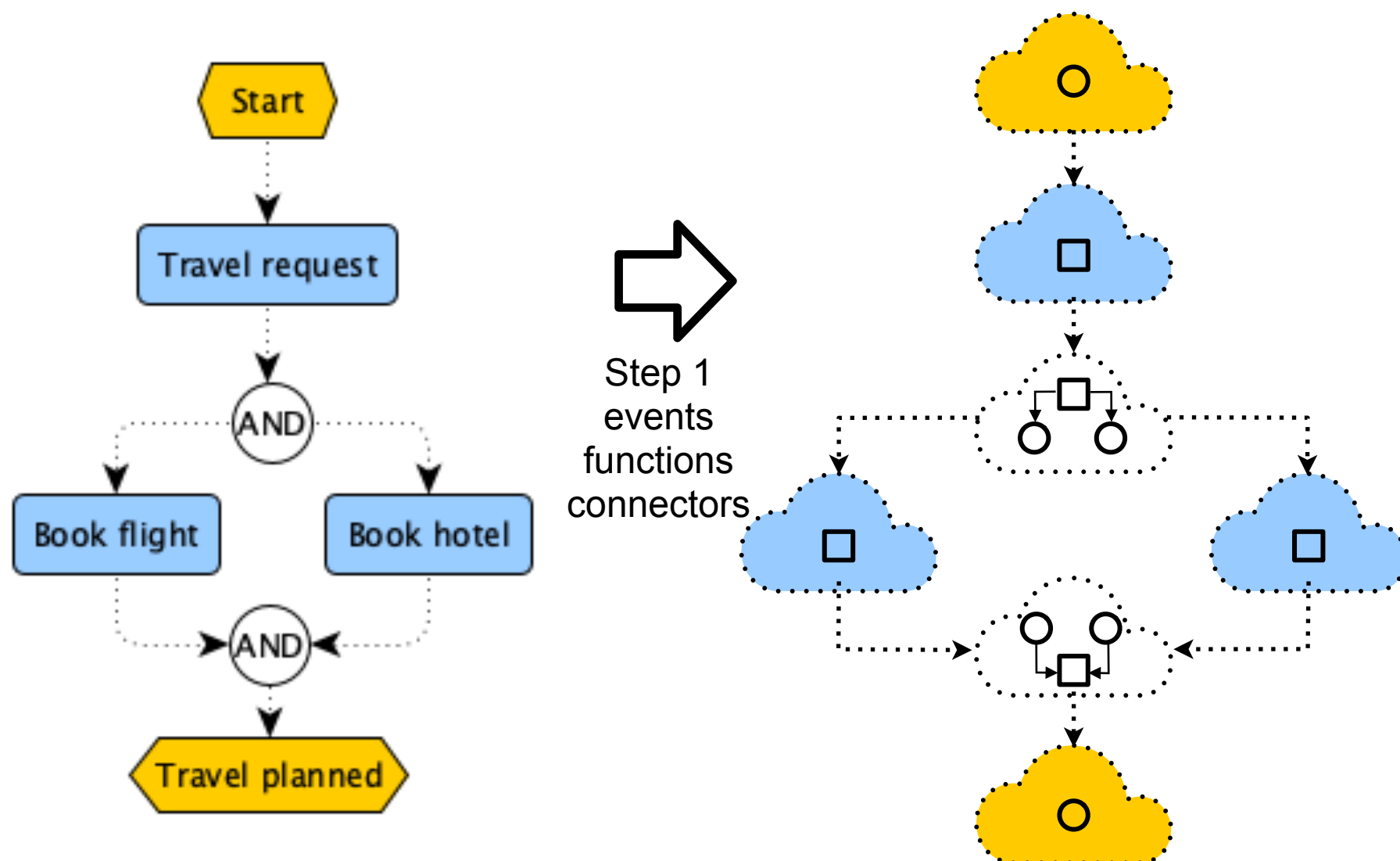


Step 3
enforce
initial place
final place



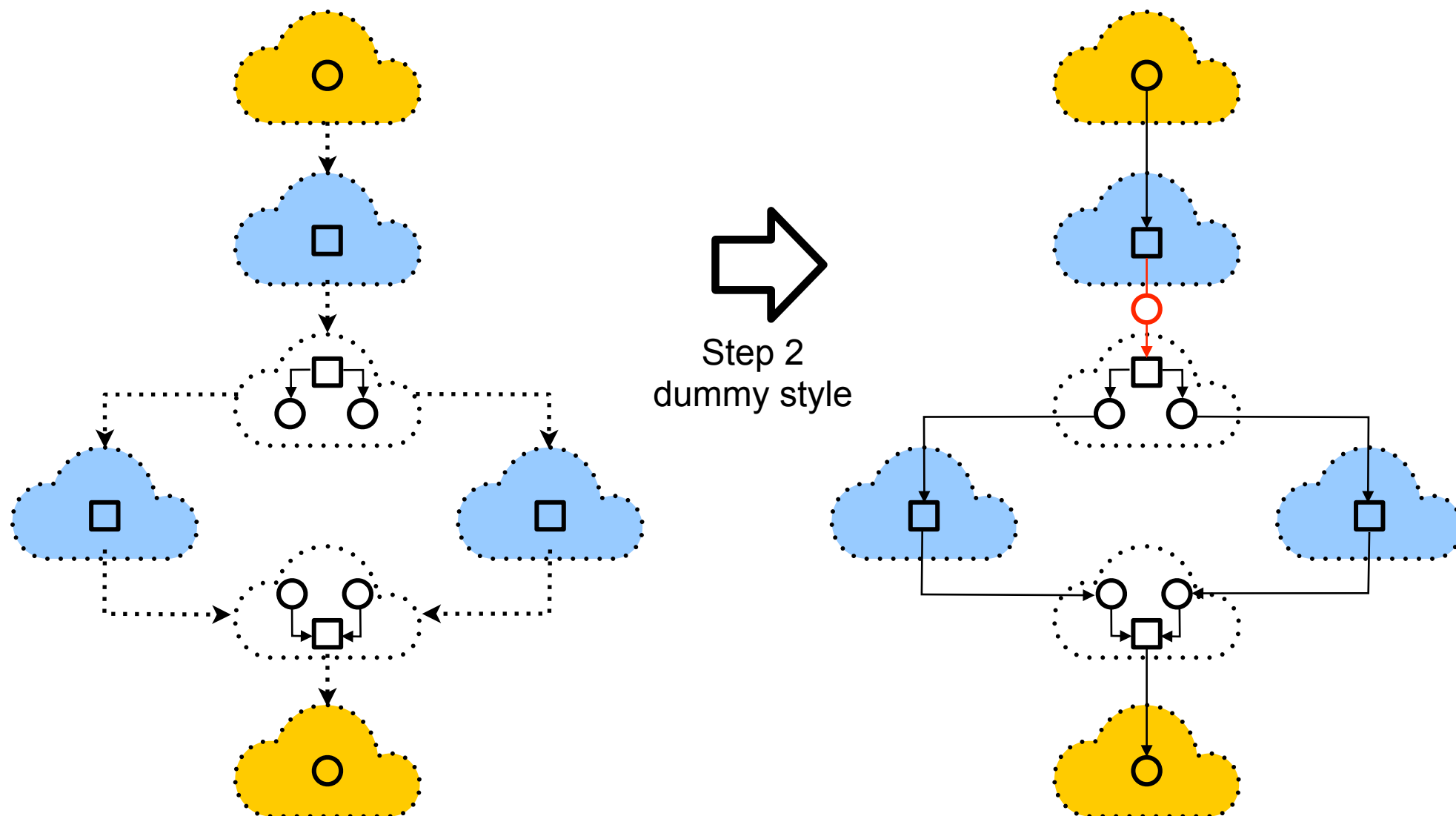
Step 1

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



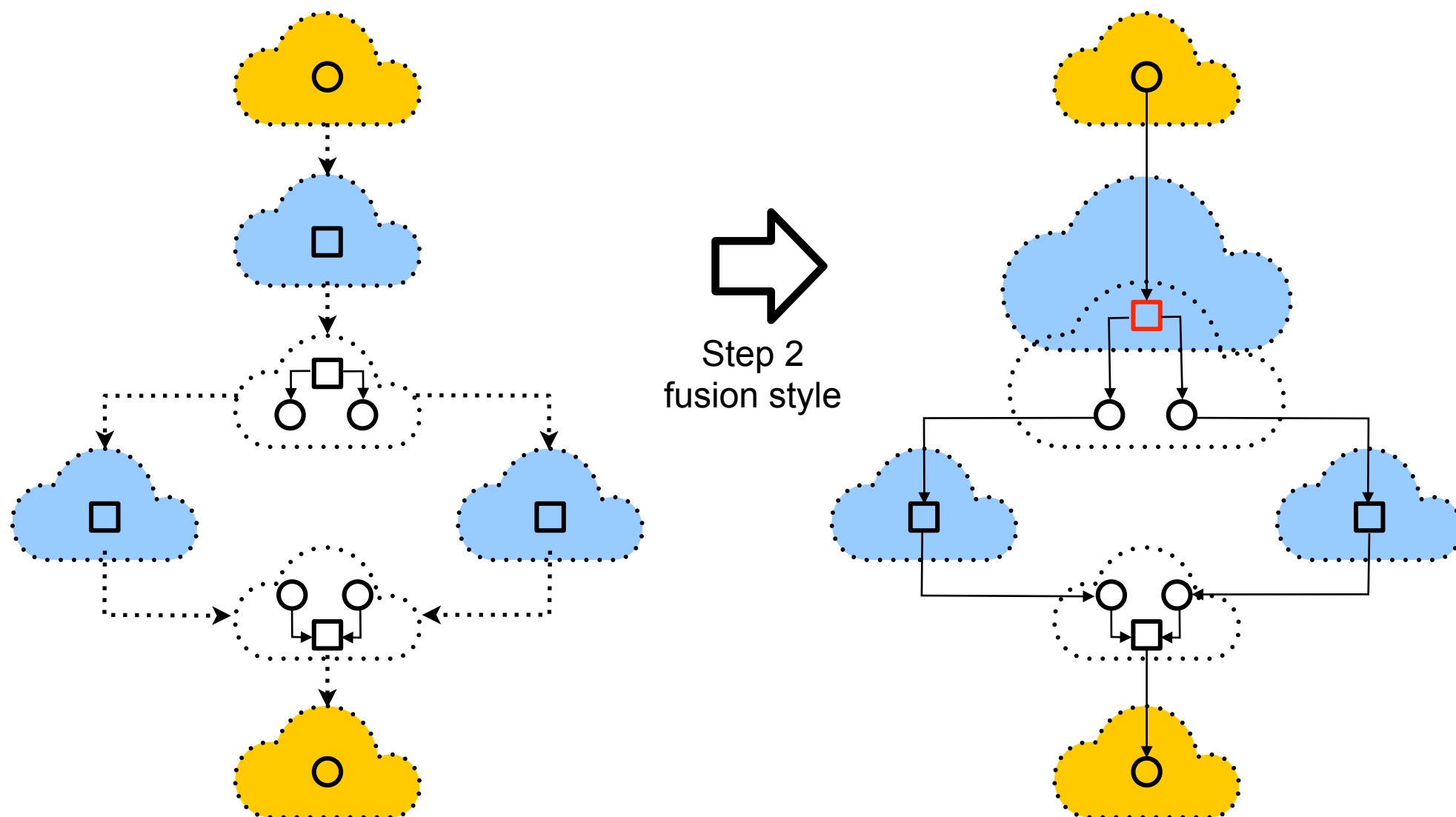
Step 2: dummy style

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

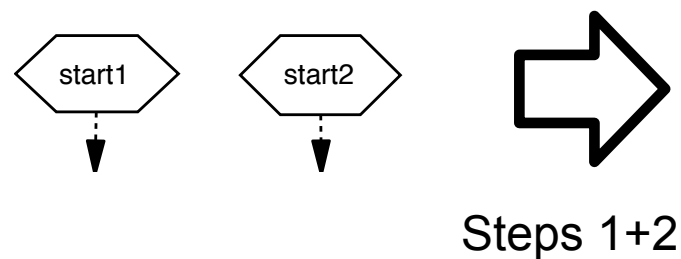


Step 2: fusion style

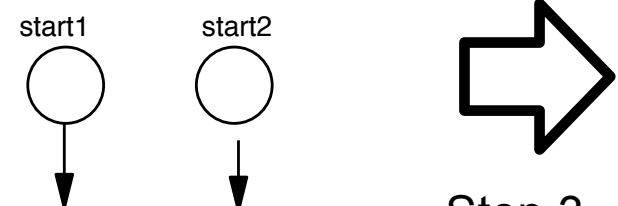
Then we connect the fragments together
(or we may decide to merge **places** / **transitions**)



Step 3: unique start

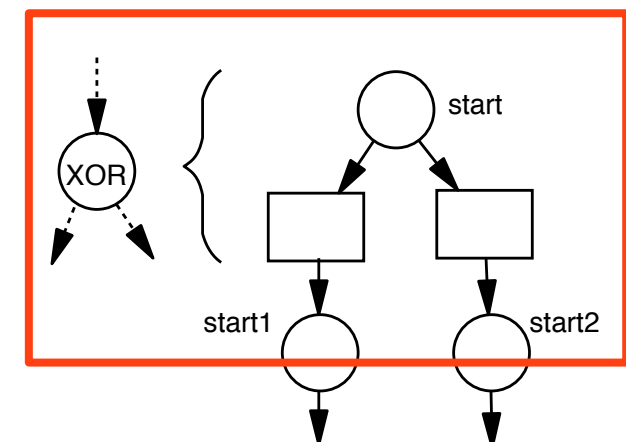


Steps 1+2

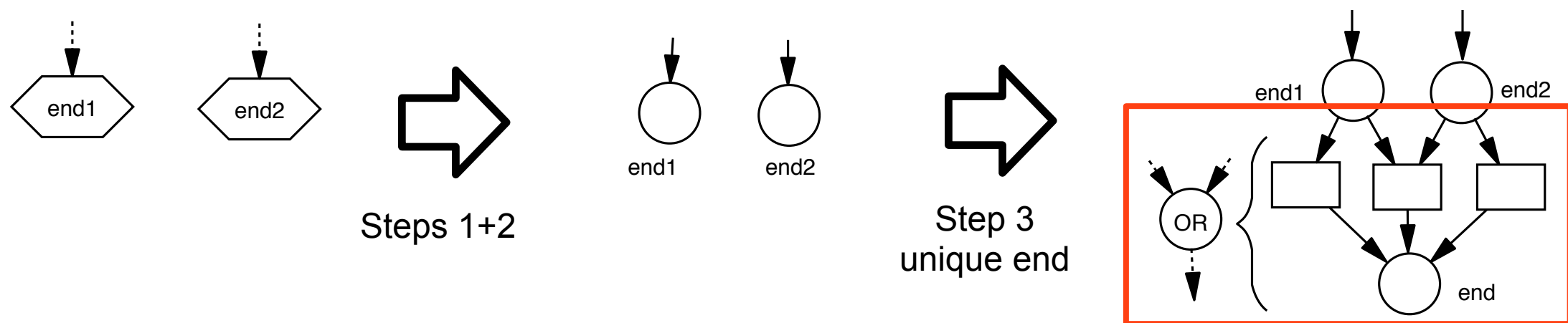


Step 3
unique start

XOR start



Step 3: unique end



OR end

(sometimes XOR/AND can be preferred)

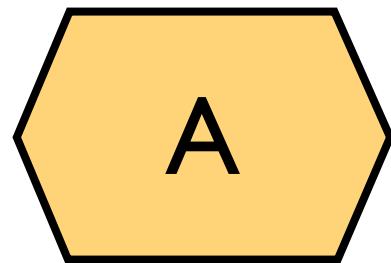
Three approaches

We overview **three** different translations

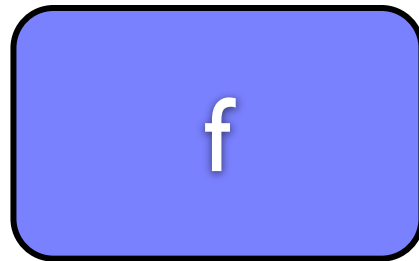
n.	ingenuity	style	applicability	outcome
1st	easy	fusion	any EPC	likely unsound, (relaxed soundness)
2nd	medium, context dependent	(dummy)	simplified EPC: event function alternation, no OR connectors	free-choice net
3rd	hard, context dependent	dummy	decorated EPC: join-split correspondence, OR policies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

Commonalities

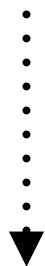
EPC element



event



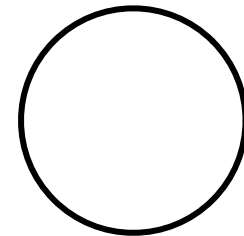
function



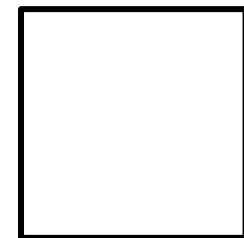
control flow

net fragment

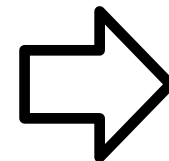
place



transition



arc



First attempt (straight translation)

Relaxed Soundness of Business Processes

Juliane Dehnert^{1,*} and Peter Rittgen²

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

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**

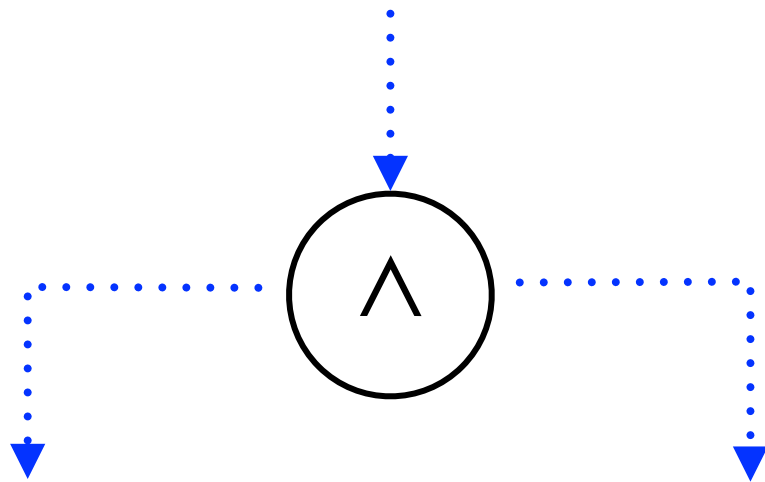
it is important to find out **flaws** as **soon** as possible

therefore

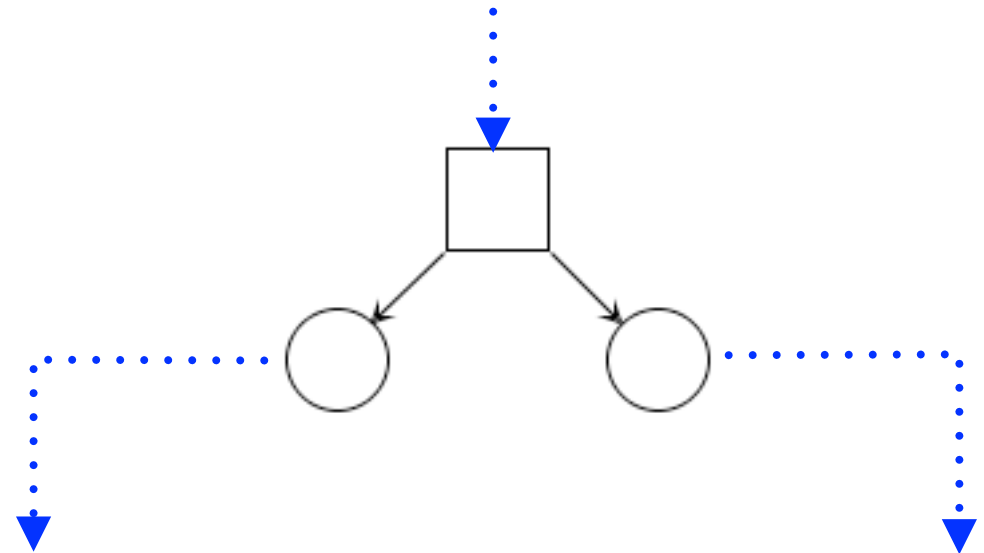
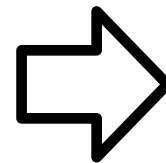
we need to fix a **formal representation**
that **preserves all ambiguities**

Step 1: AND split

EPC element

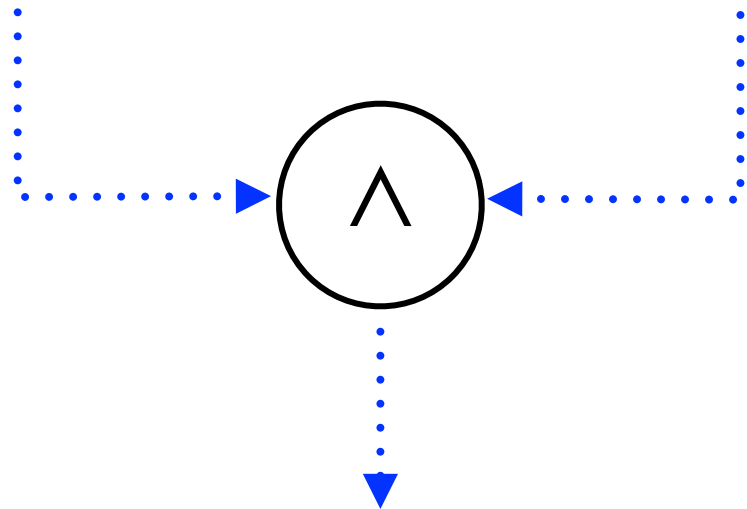


net fragment

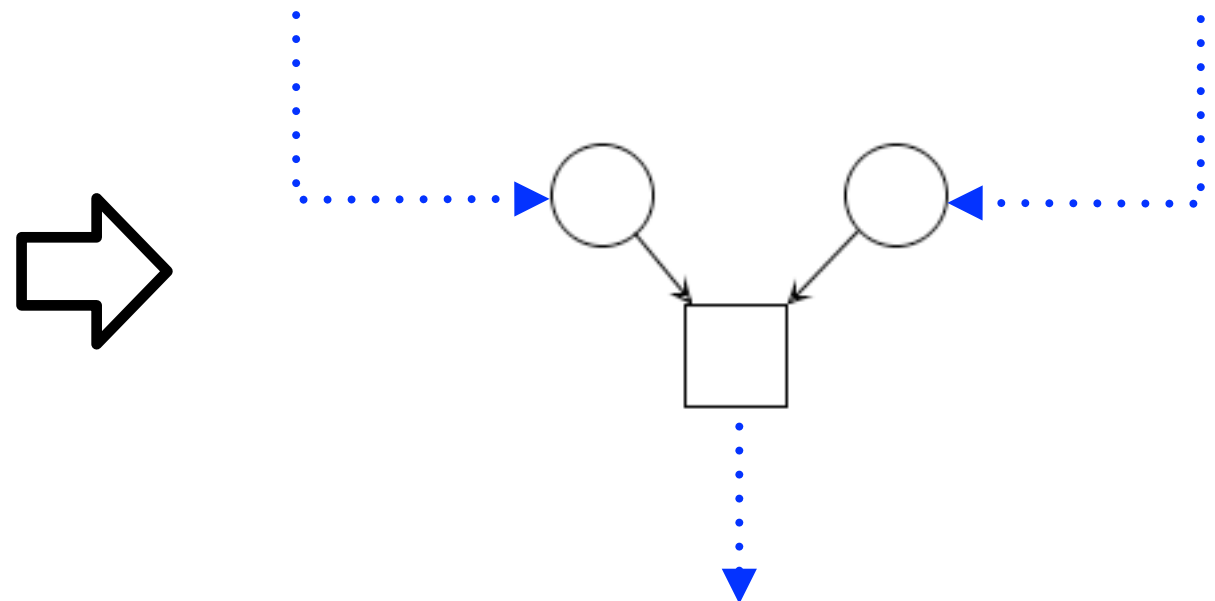


Step 1: AND join

EPC element

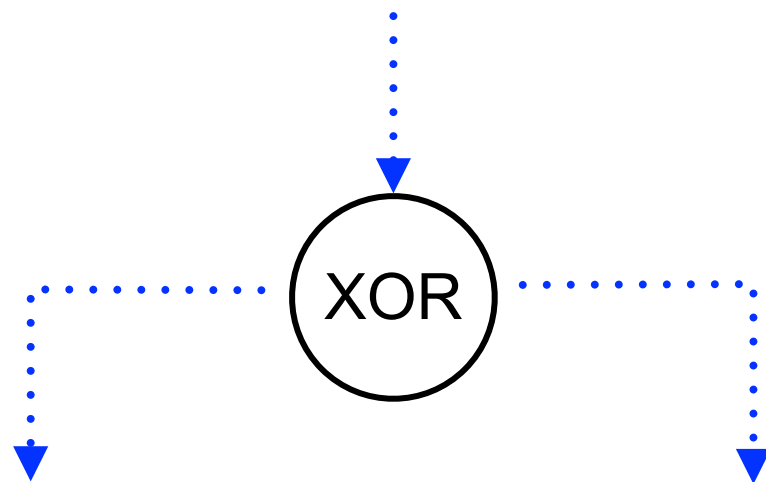


net fragment

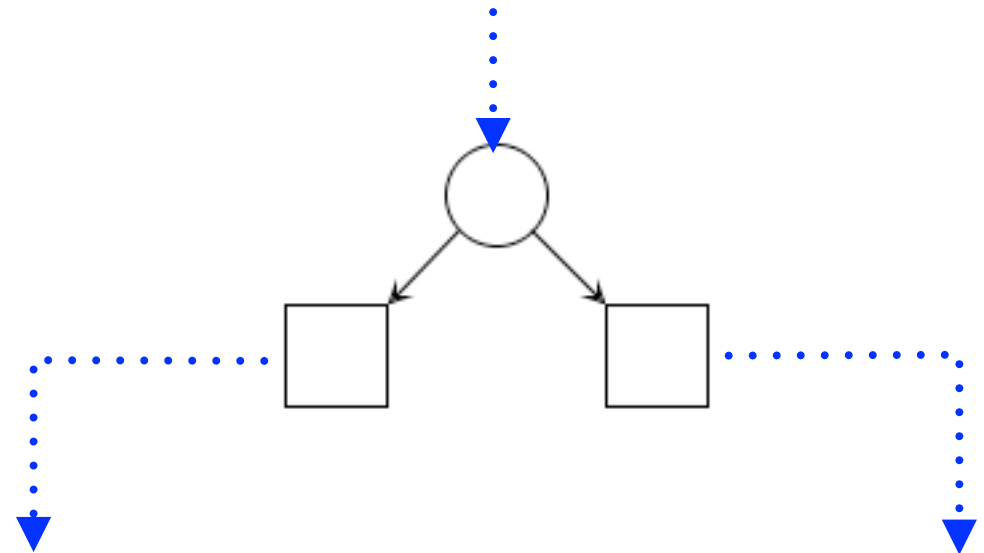
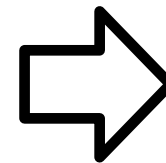


Step 1: XOR split

EPC element

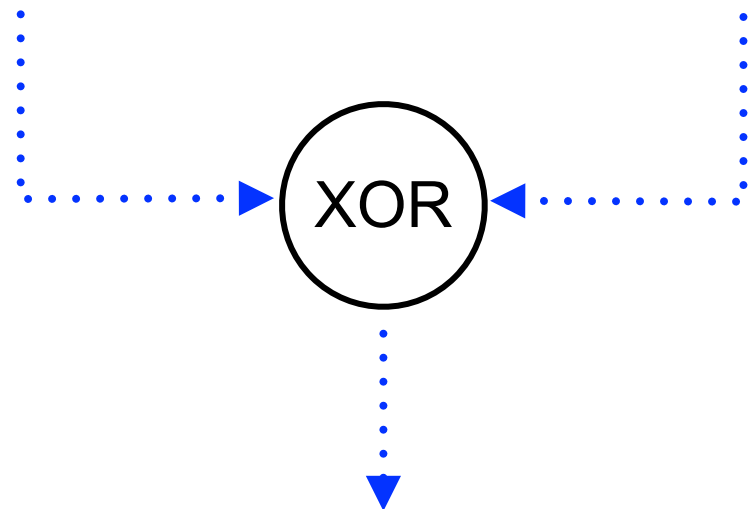


net fragment

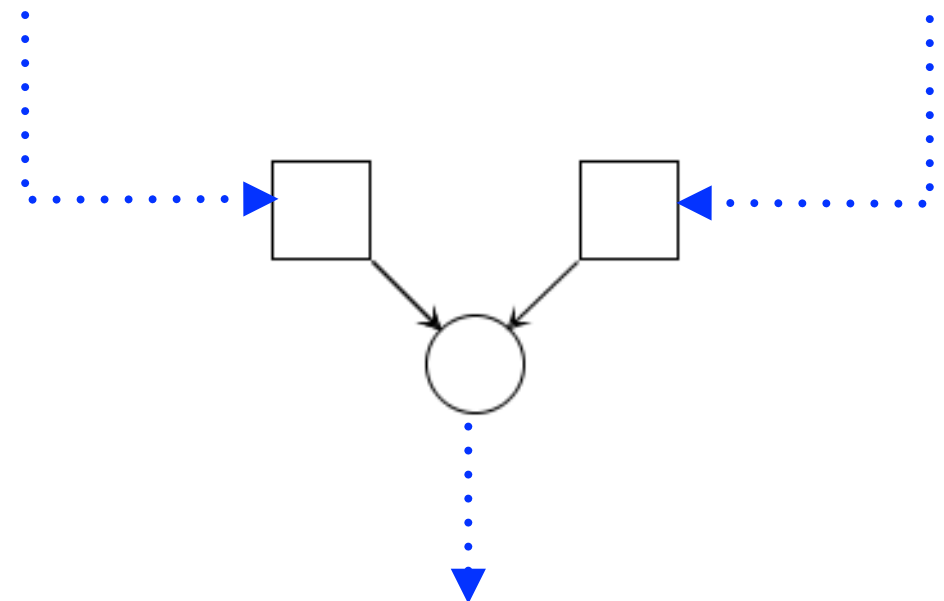
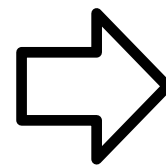


Step 1: XOR join

EPC element

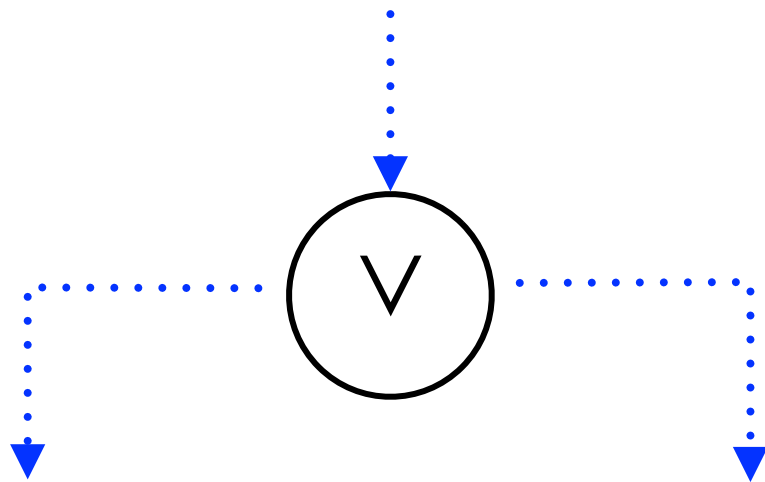


net fragment

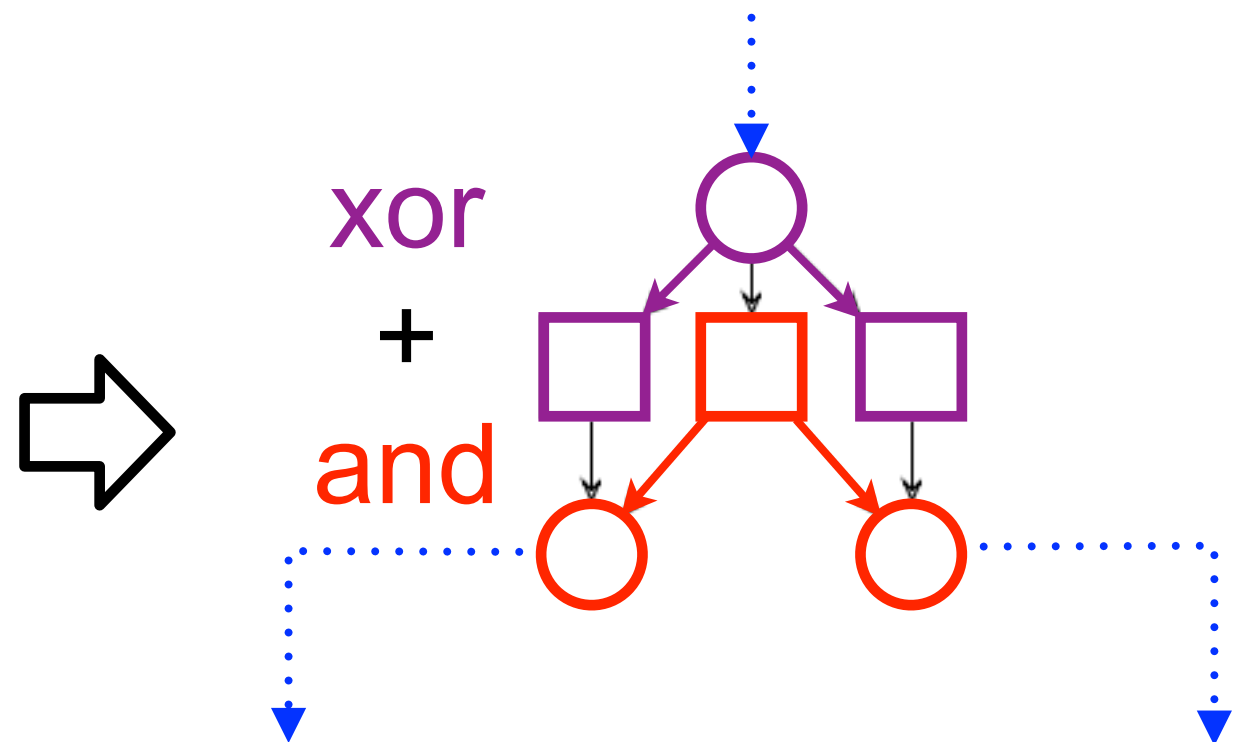


Step 1: OR split

EPC element

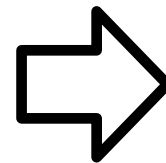
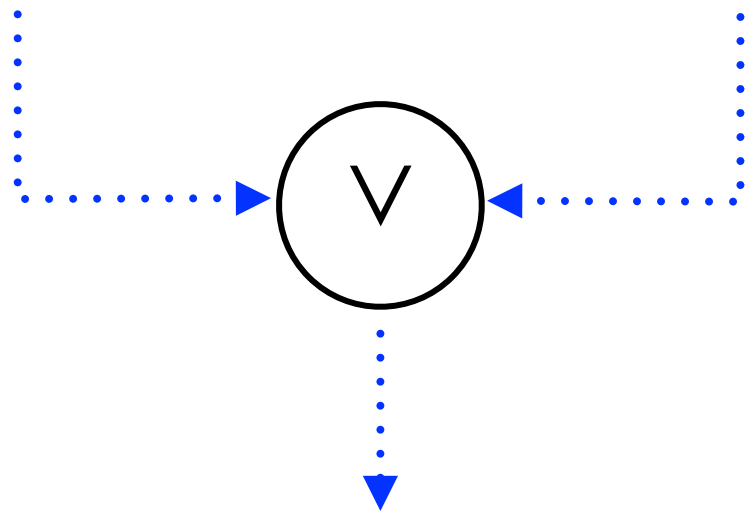


net fragment

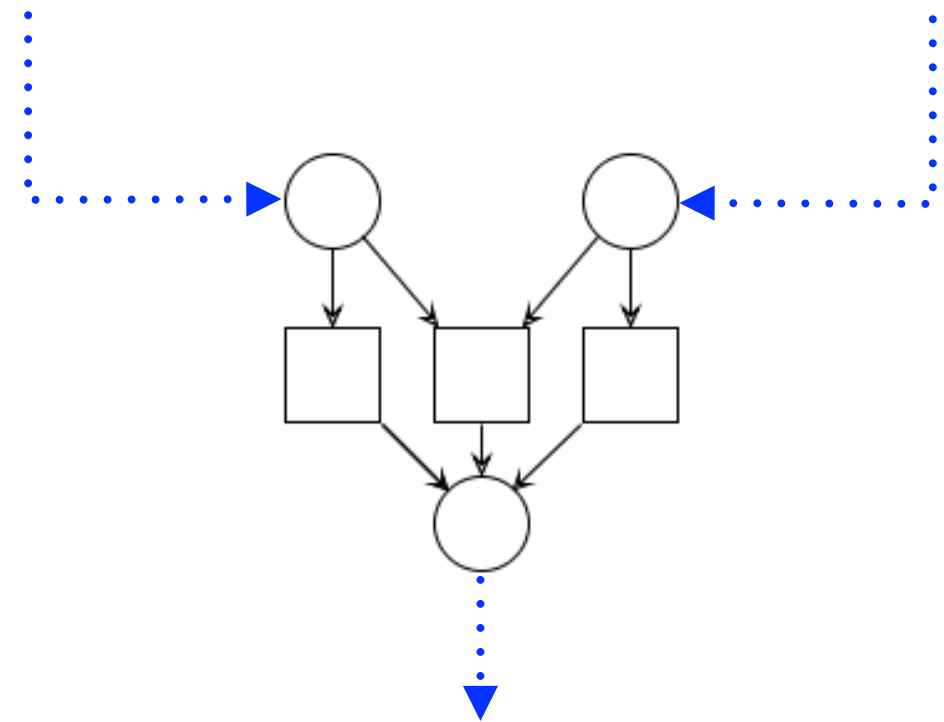


Step 1: OR join

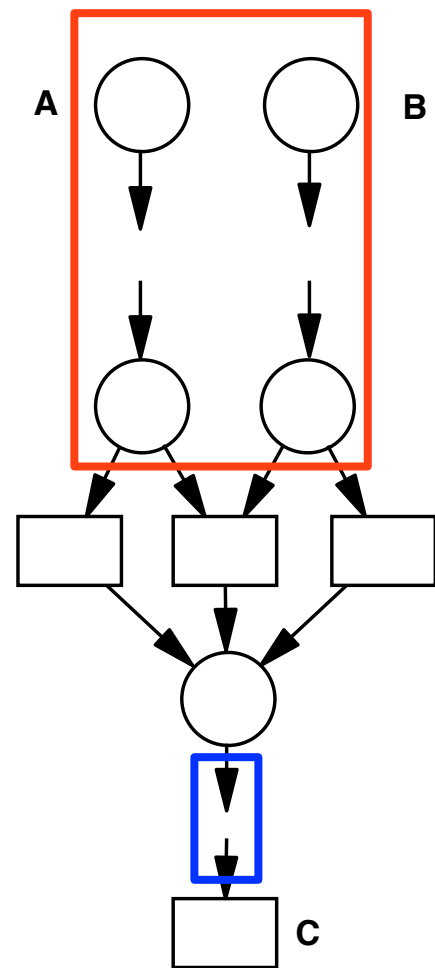
EPC element



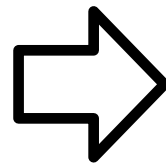
net fragment



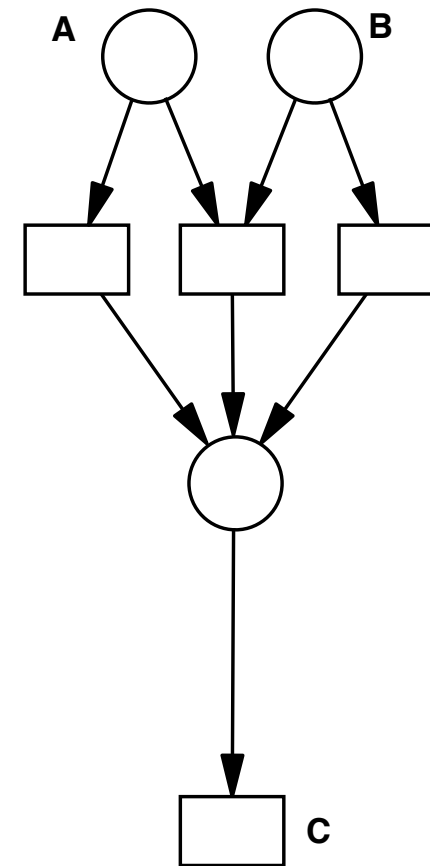
Step 2: fusion style



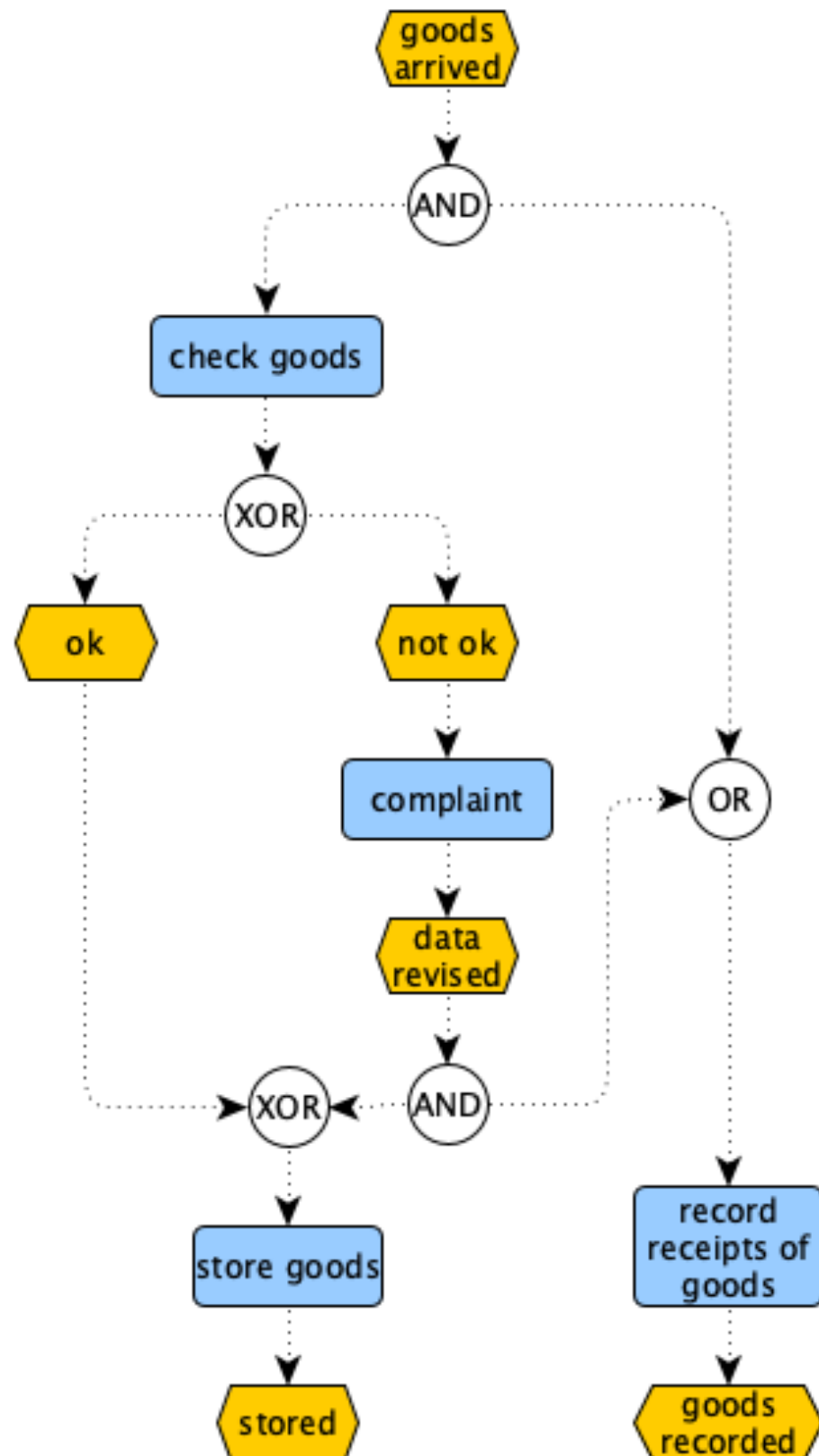
element
fusion
(case 1)



arc
fusion
(case 2)

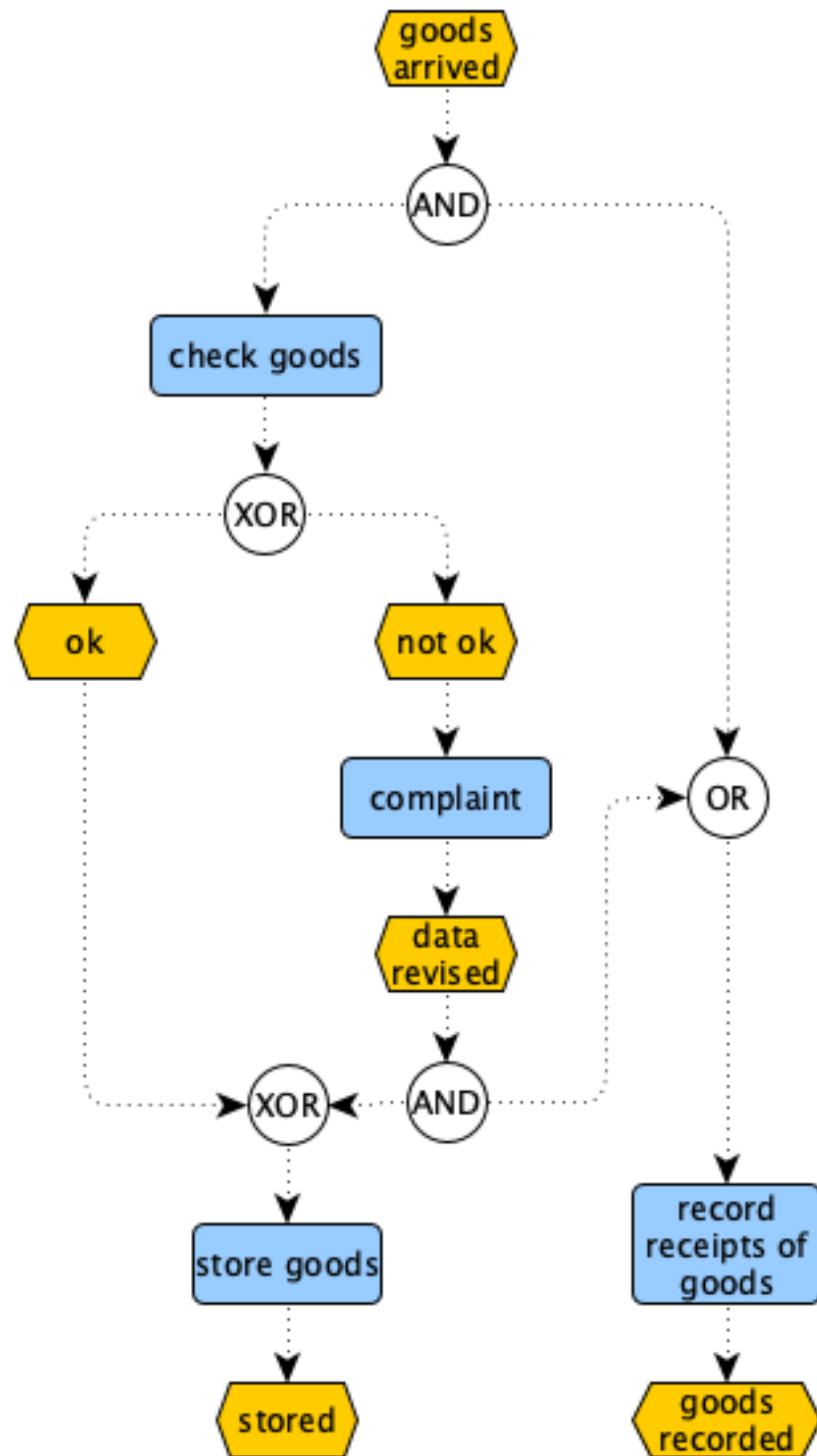


Example

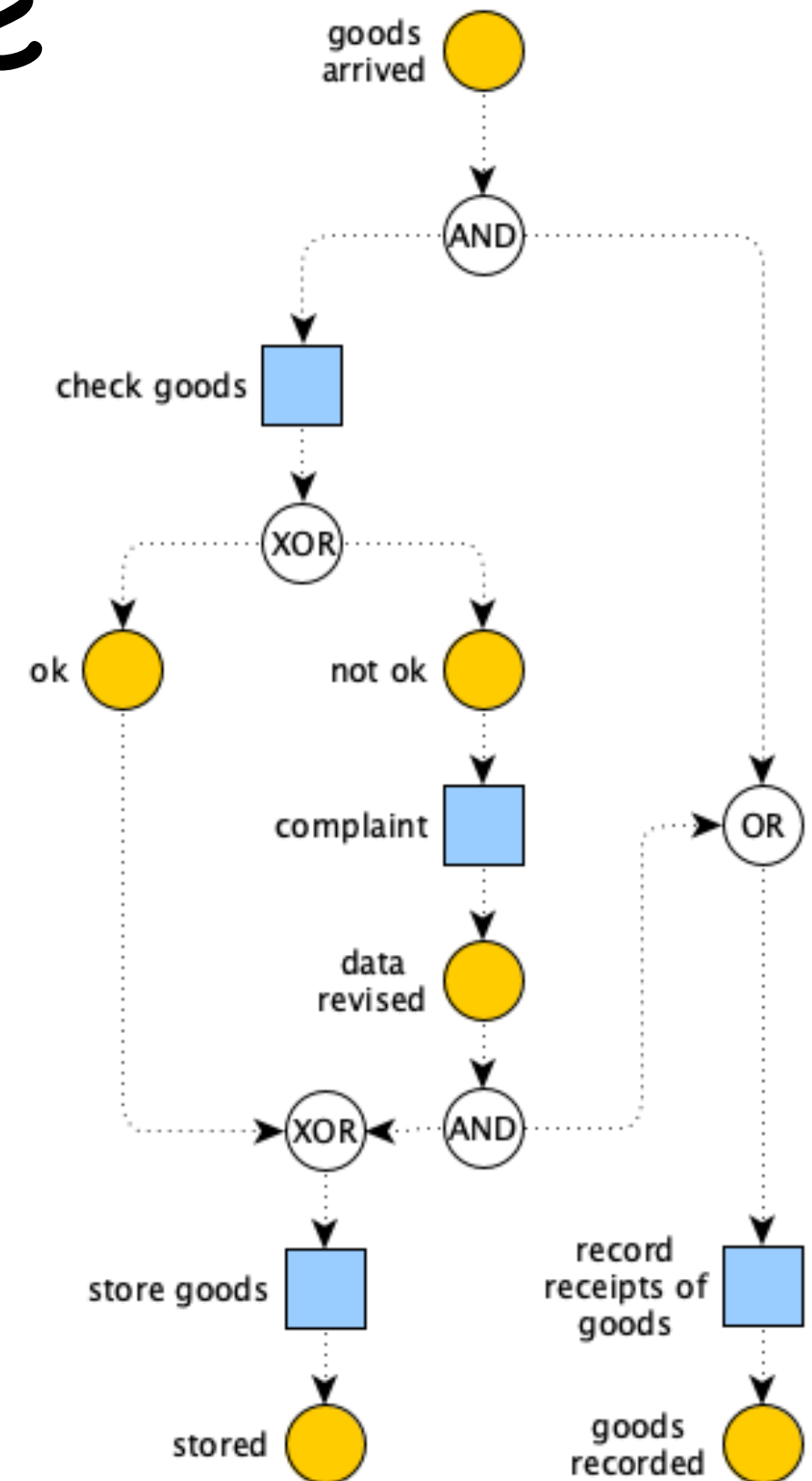


Sound?

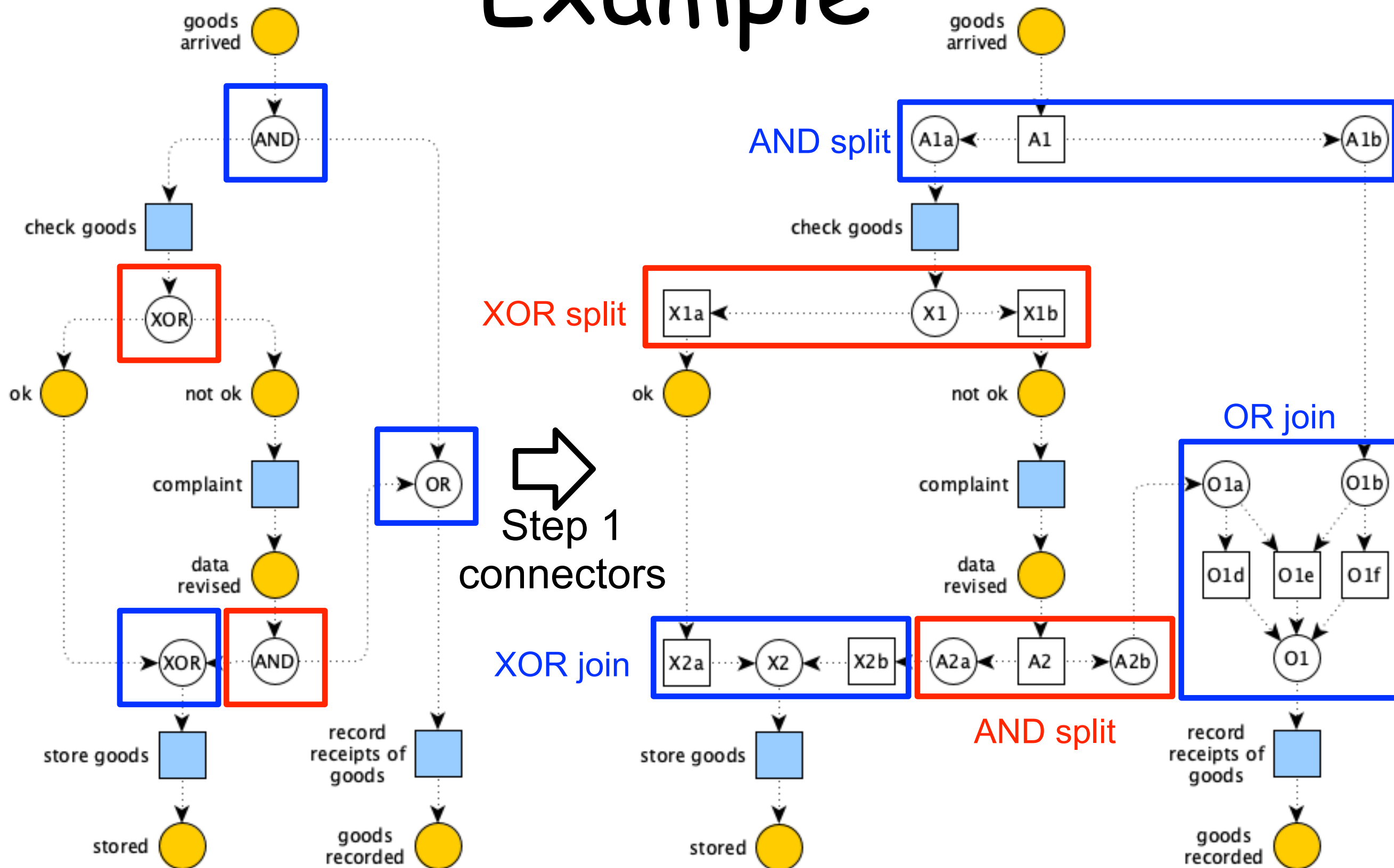
Example



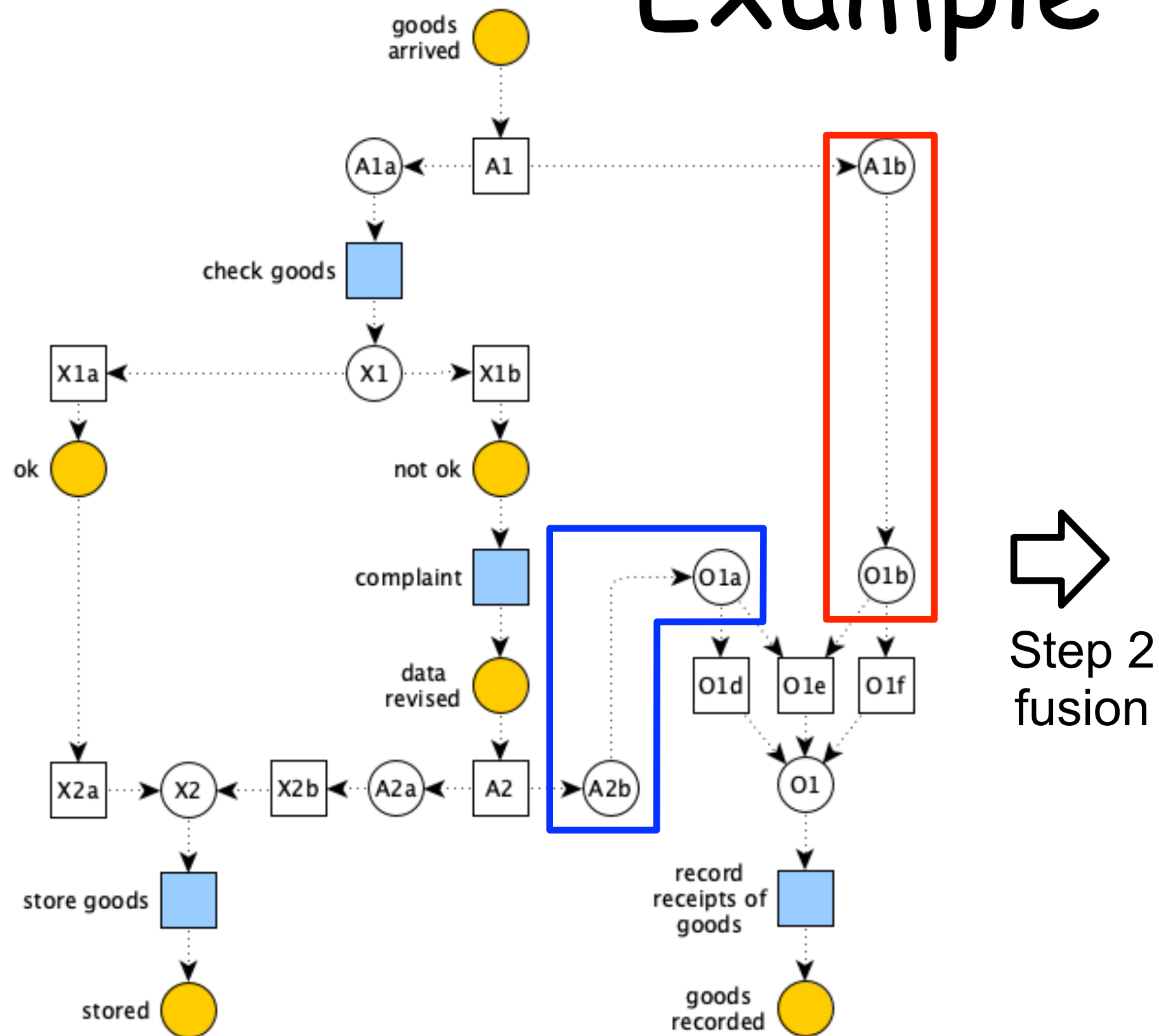
➡
Step 1
events and
functions



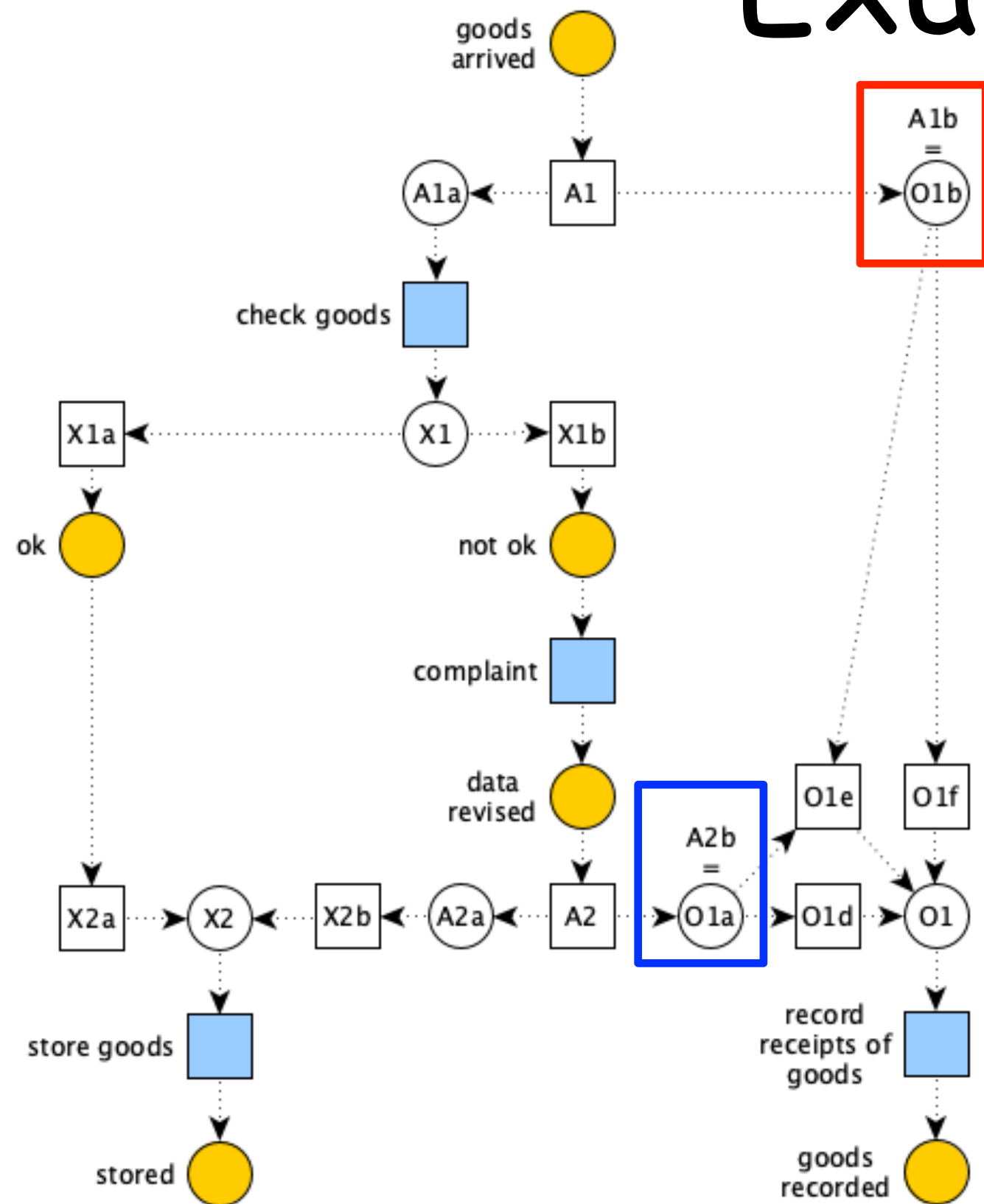
Example



Example

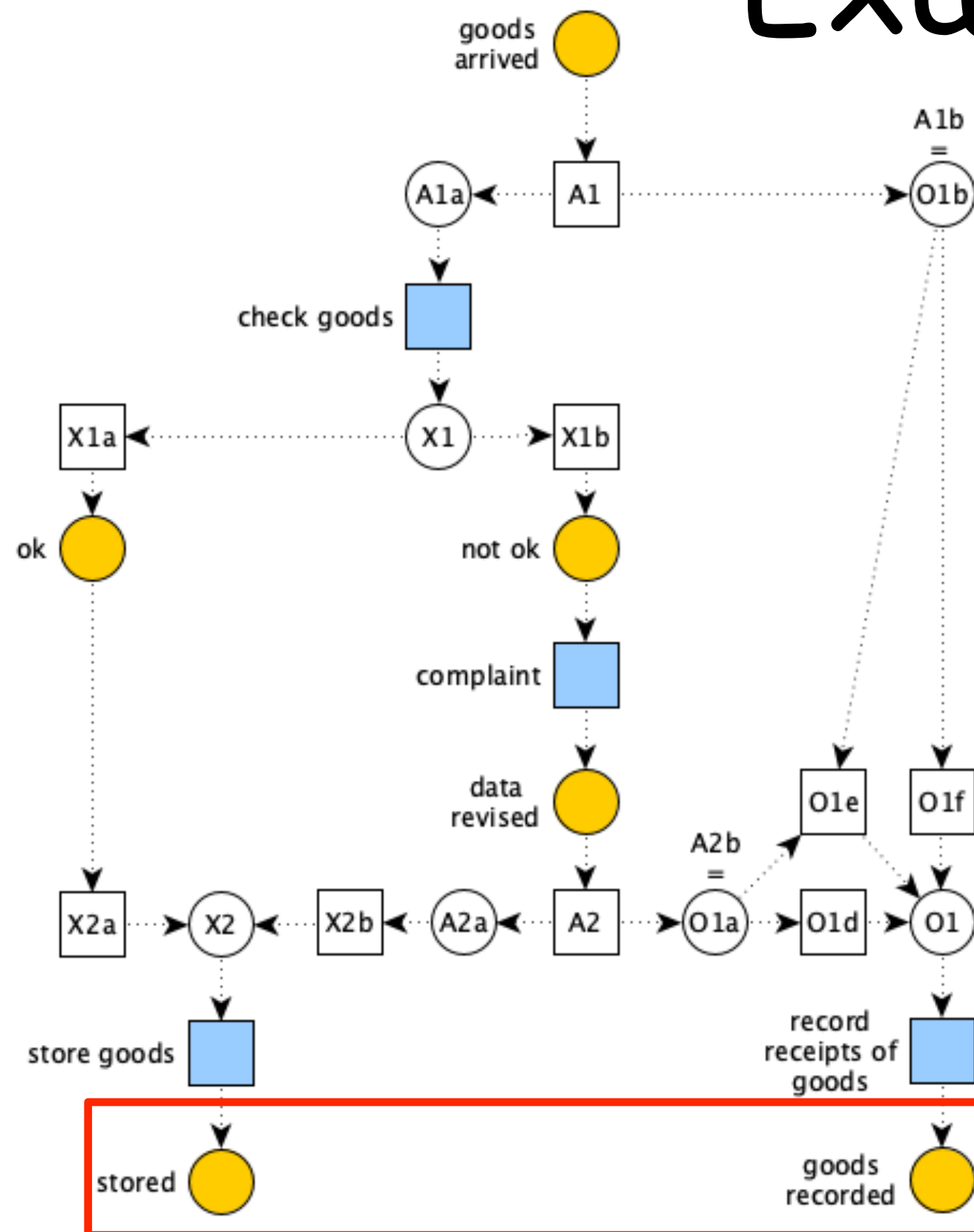


Example



←
Step 2
fusion

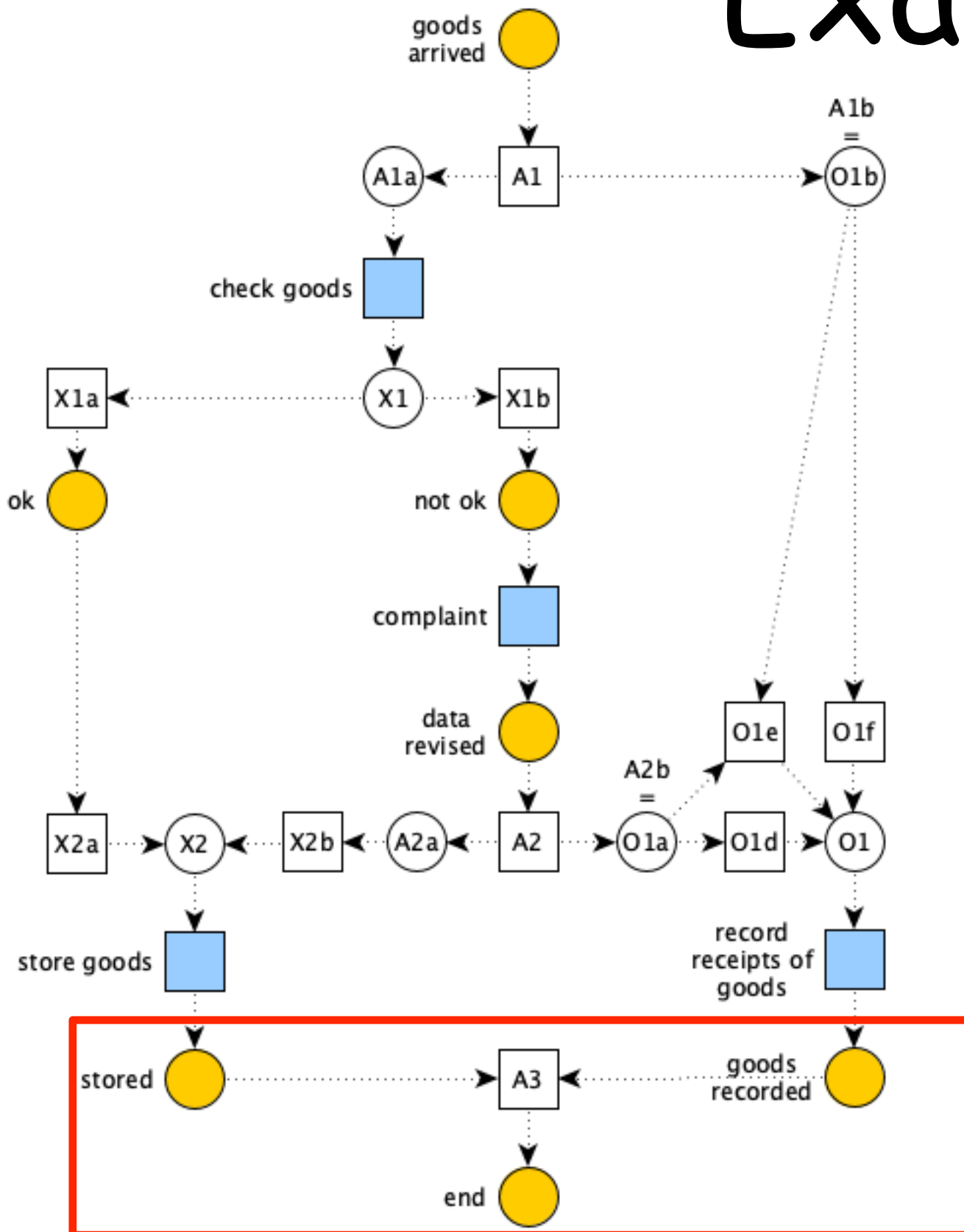
Example



➡
Step 3
unique end

implicit AND join (because of A2)

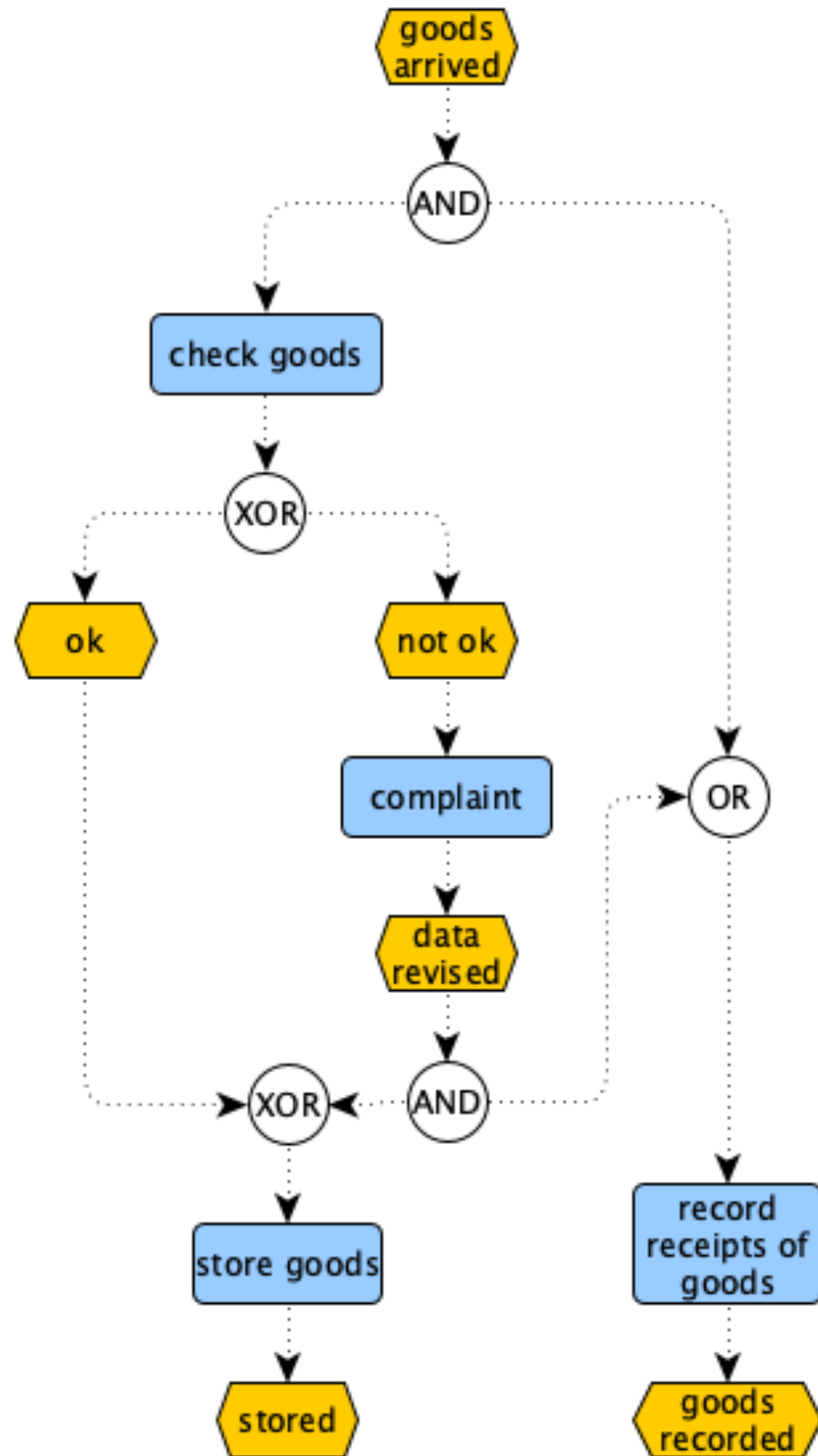
Example



←
Step 3
unique end

implicit AND join (because of A2)

EPC

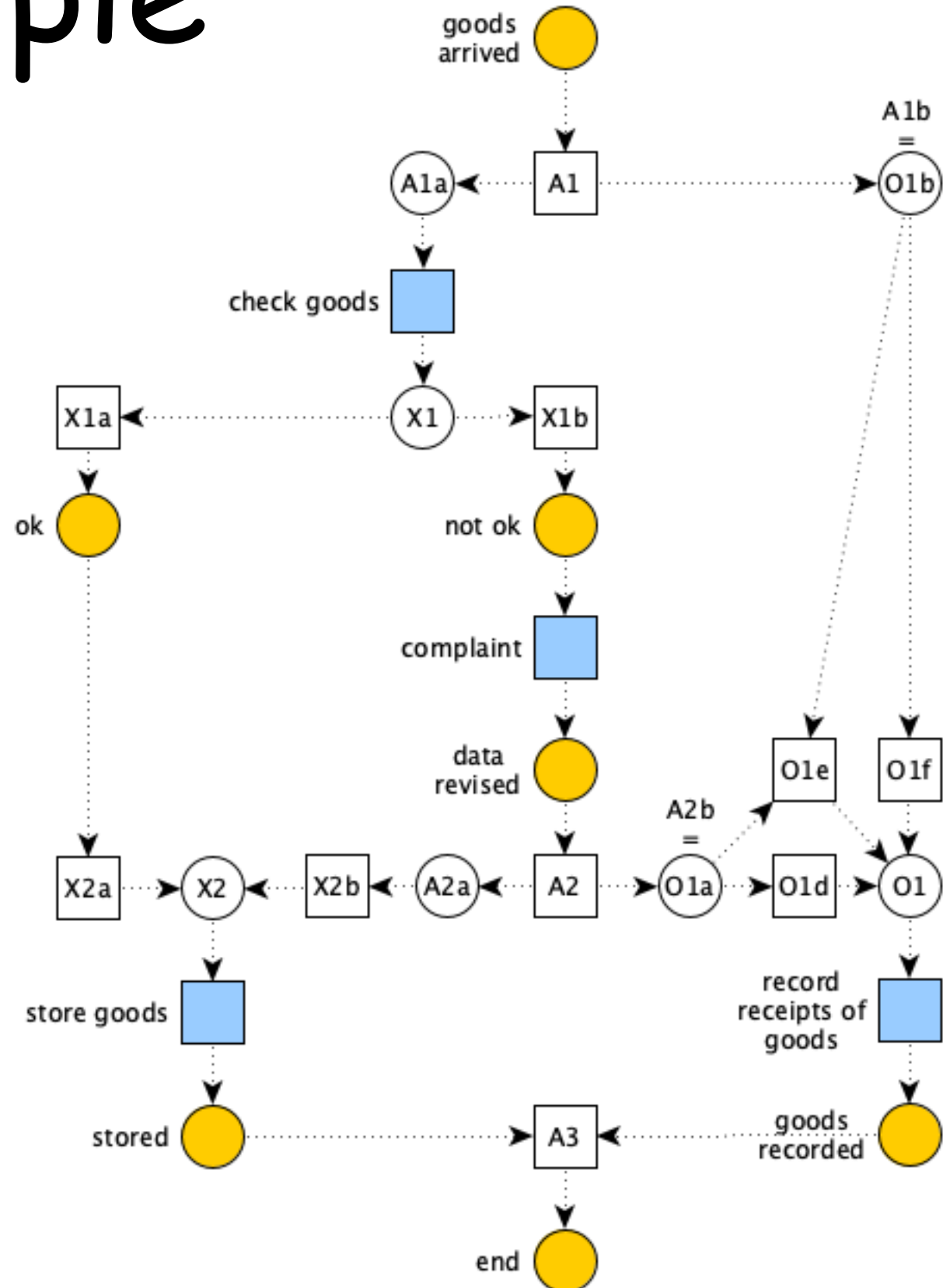


Example

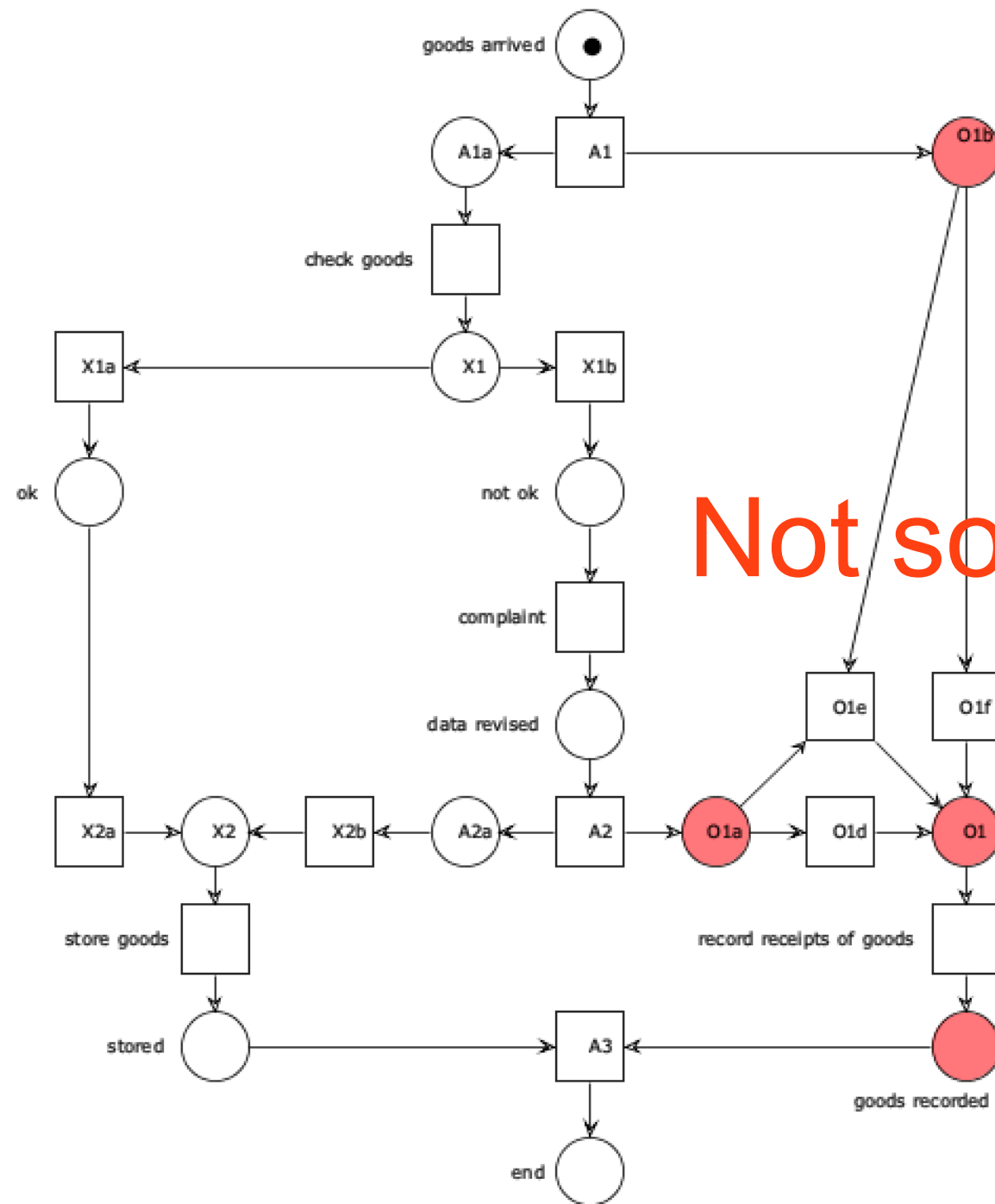
Sound?

➔
Steps
1+2+3

wf net



Soundness analysis



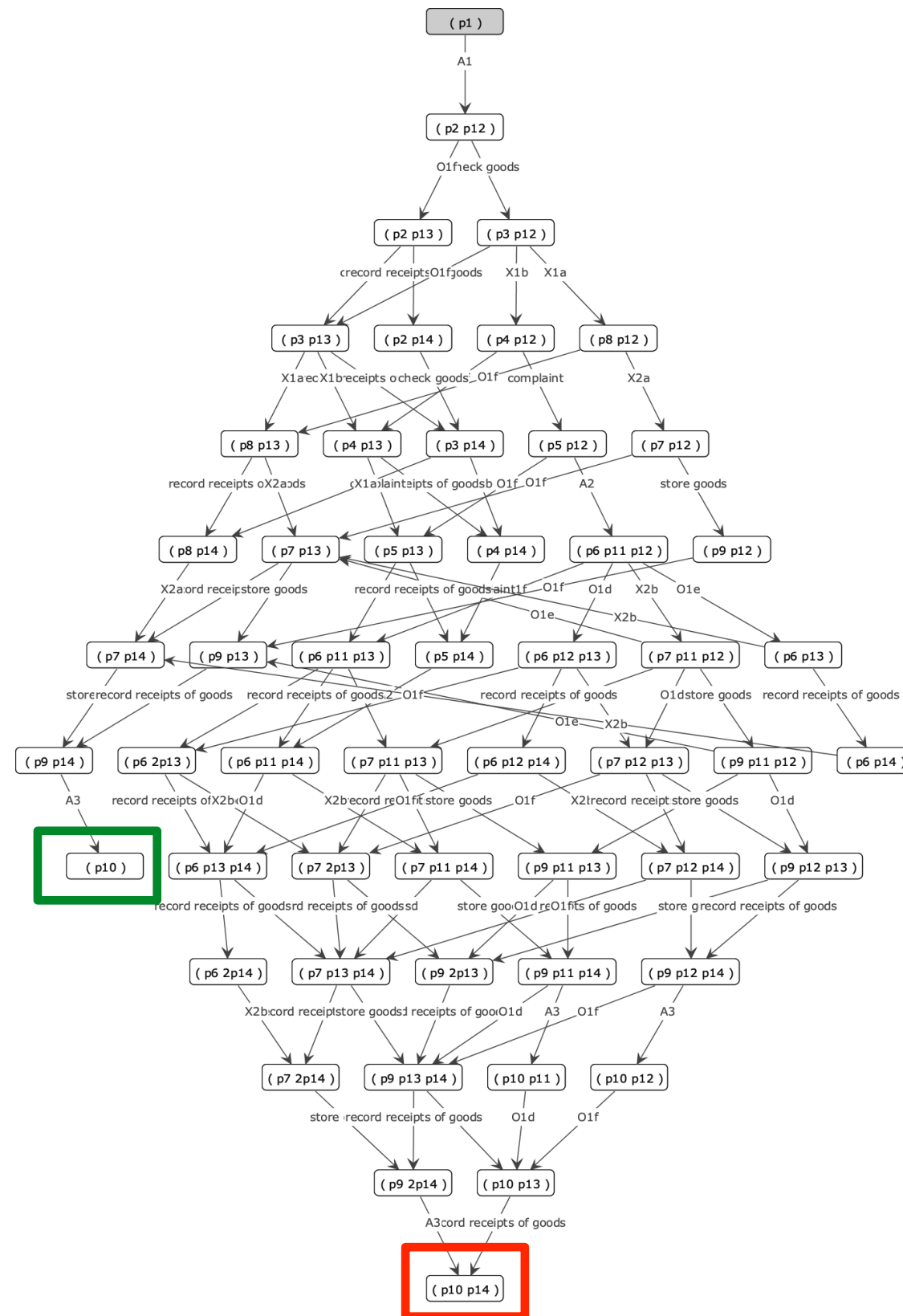
Not sound!

Semantical analysis

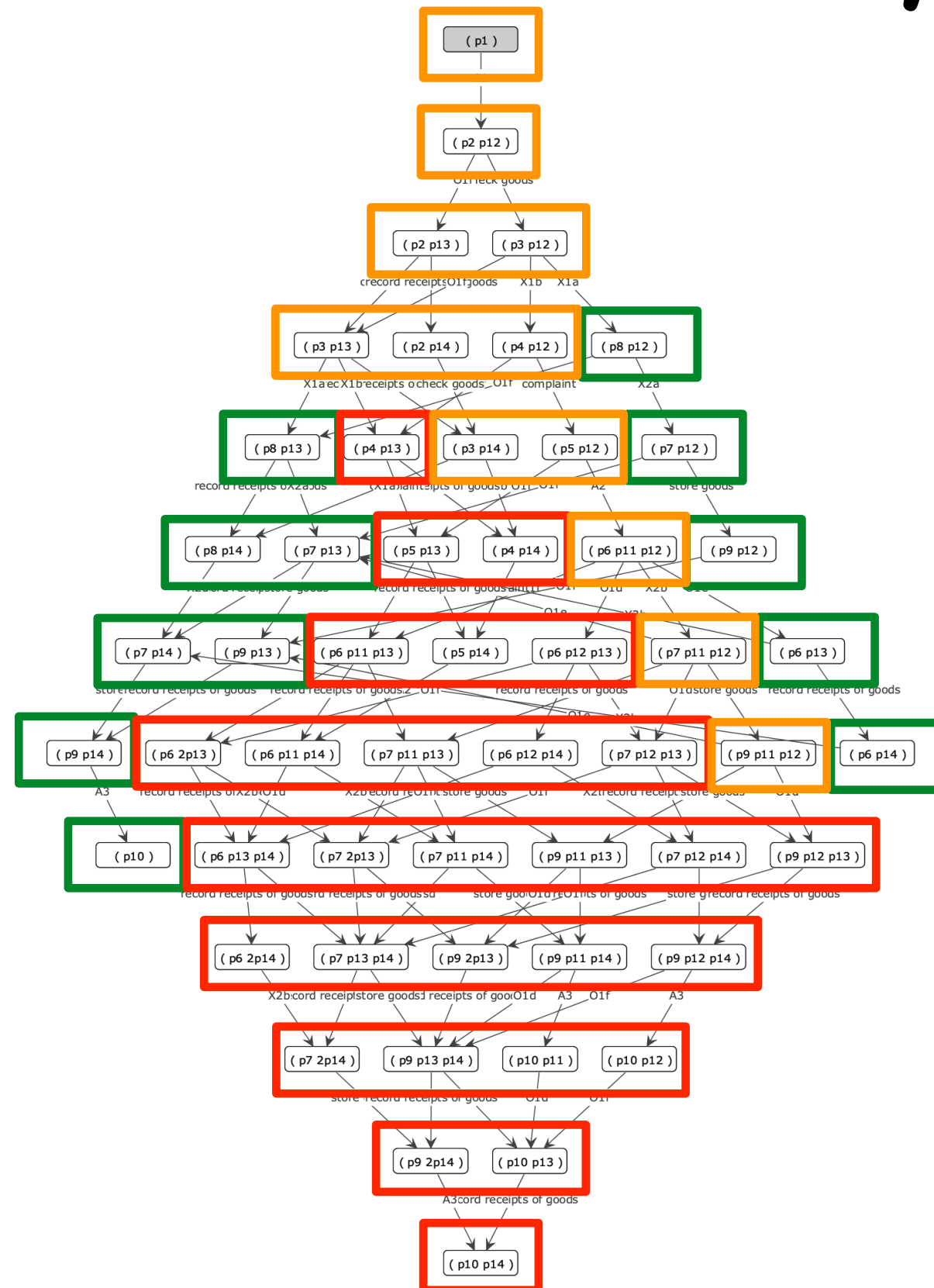
Wizard Expert

- 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

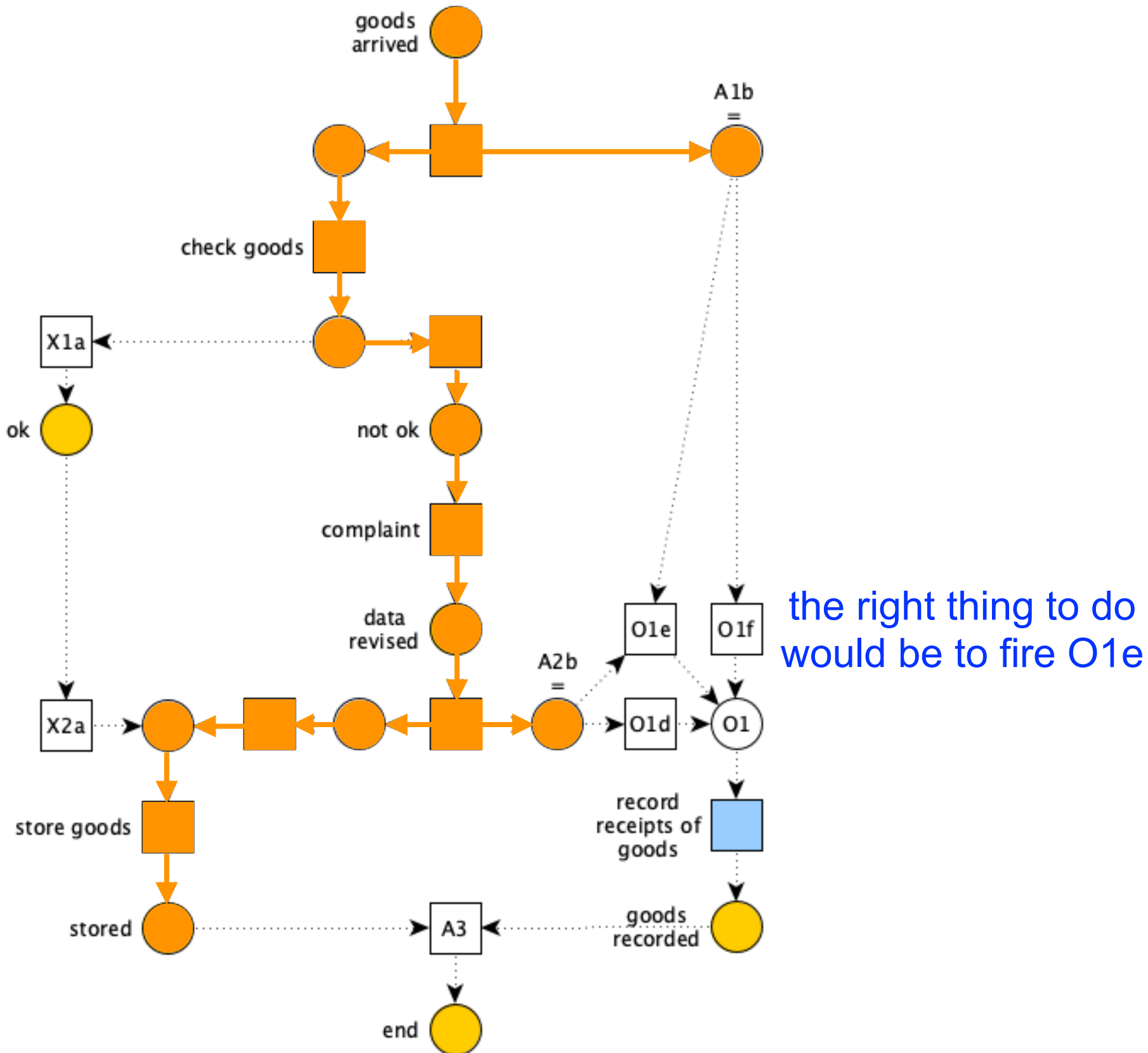
Soundness analysis



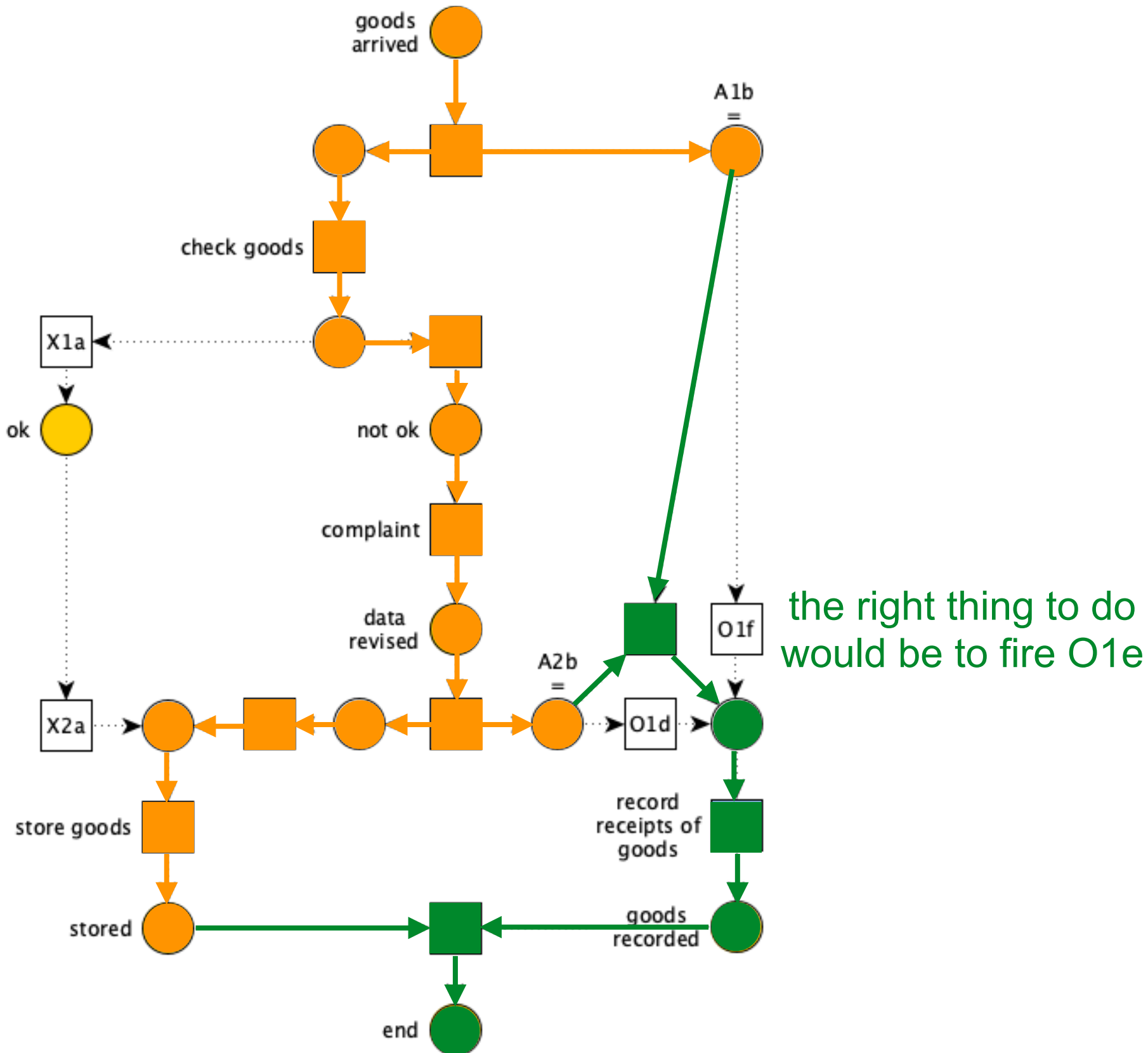
Soundness analysis



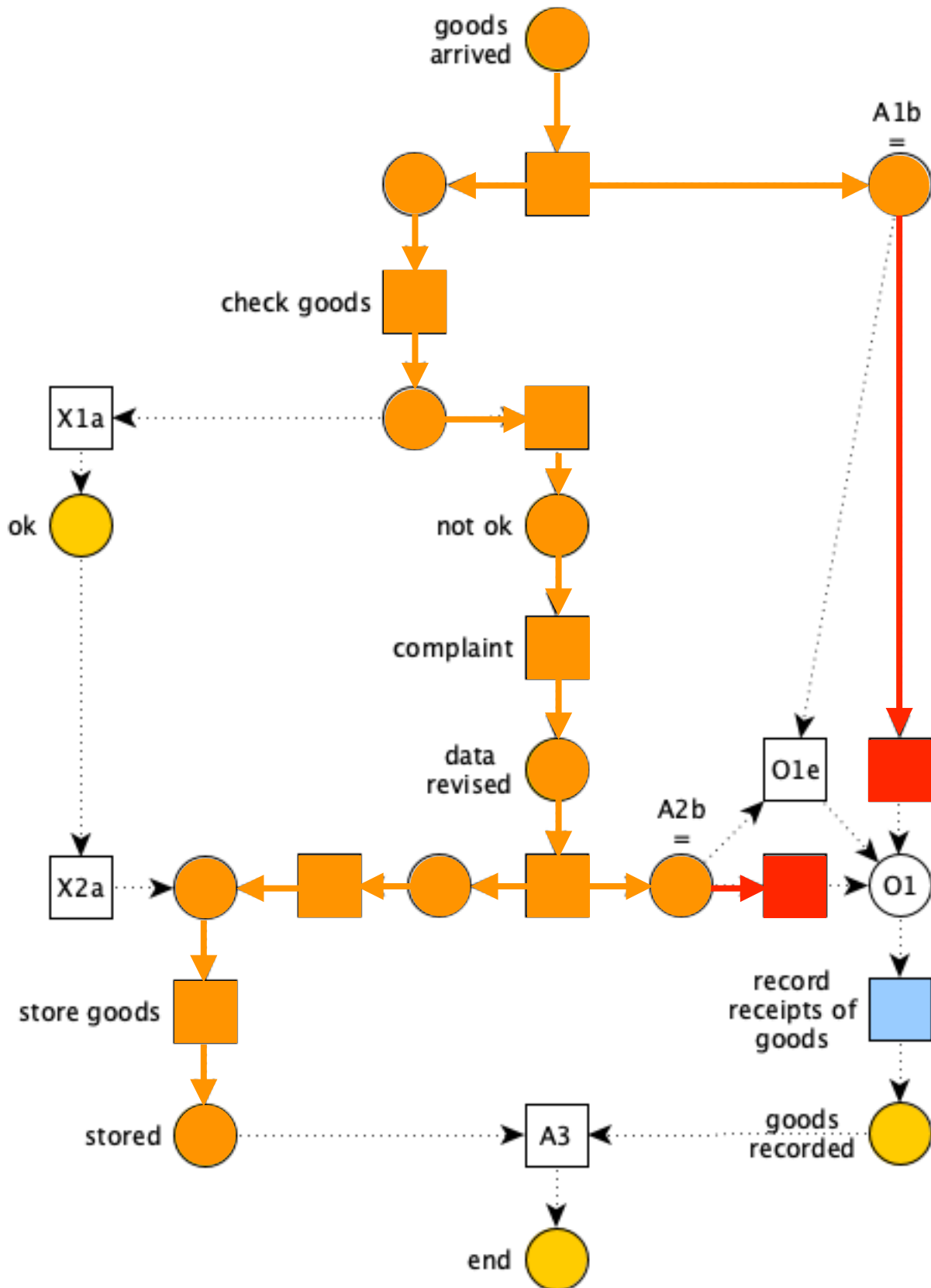
Soundness analysis



Soundness analysis

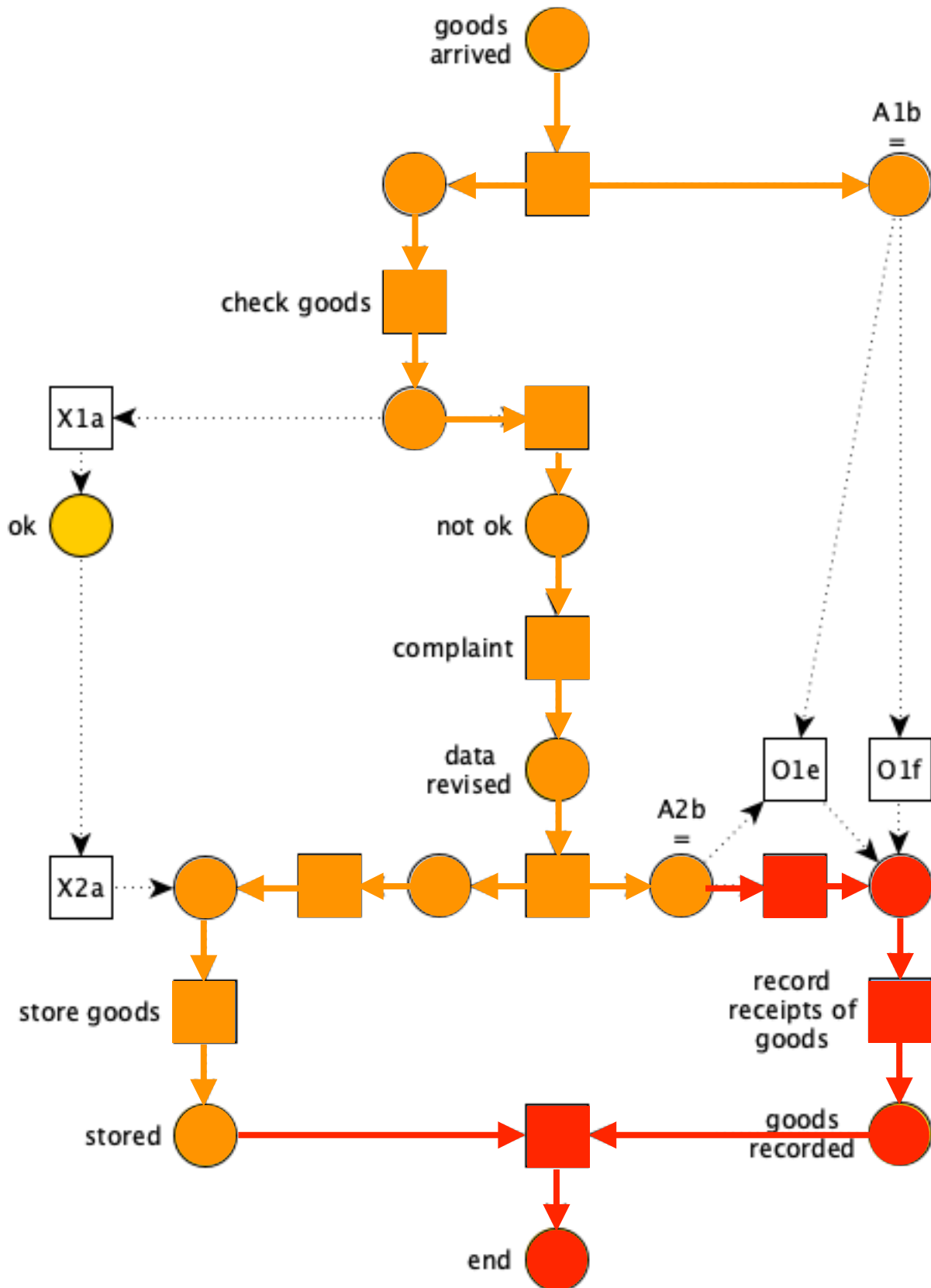


Soundness analysis



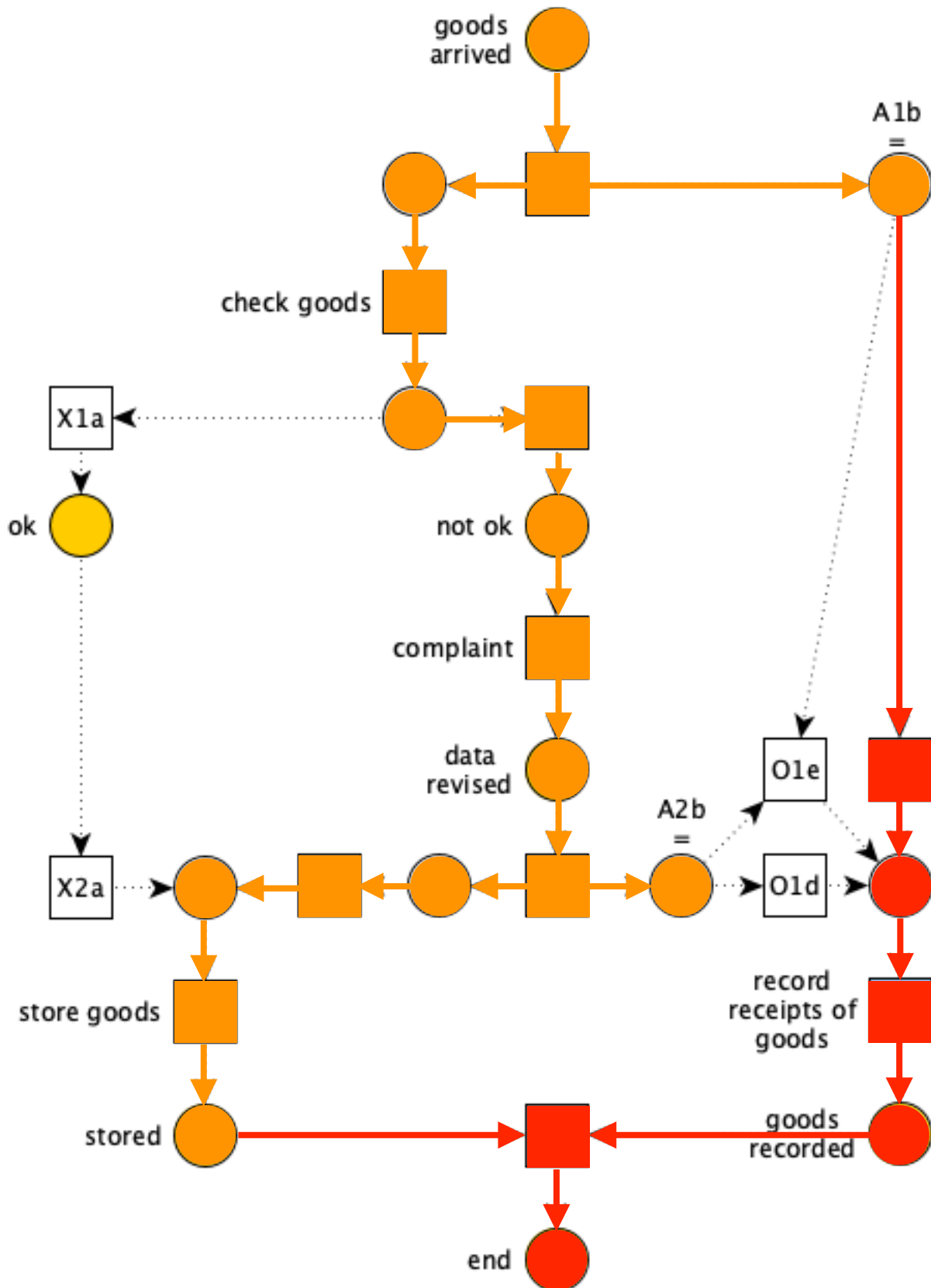
but O1f and O1d
are enabled as well
(OR semantics!)

Soundness analysis



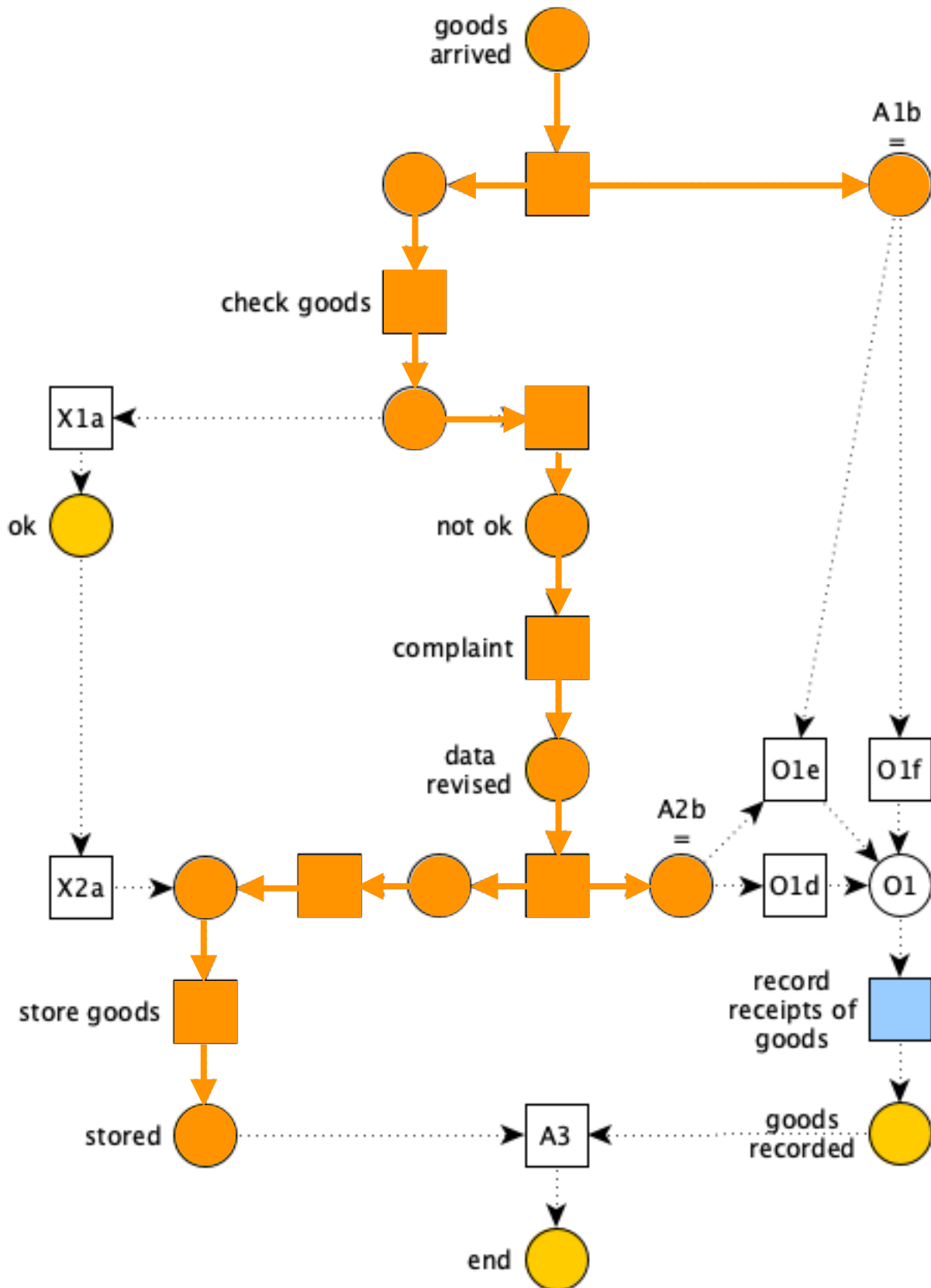
proper completion
is not guaranteed
(N* unbounded)

Soundness analysis



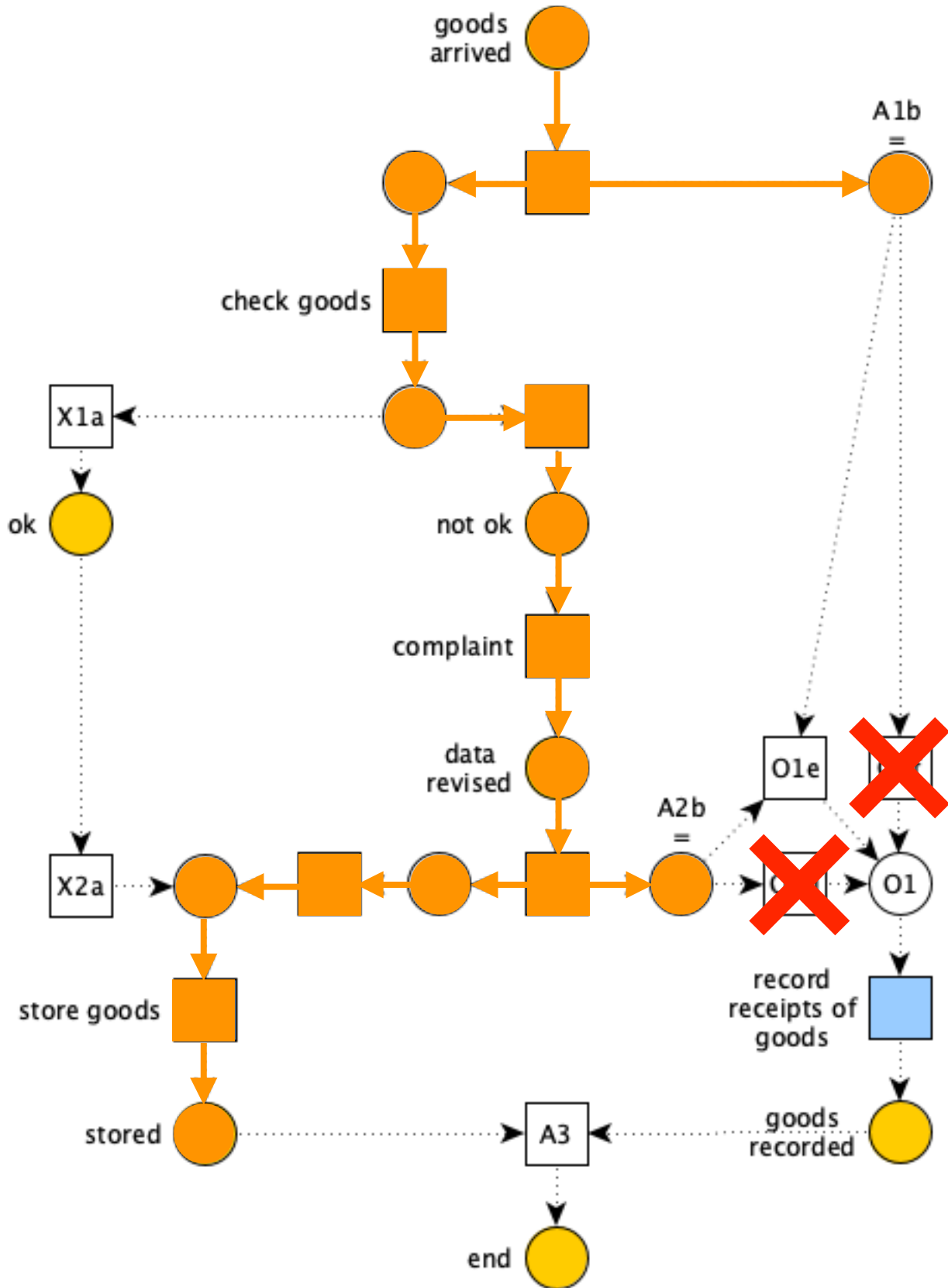
proper completion
is not guaranteed
(N* unbounded)

Soundness analysis



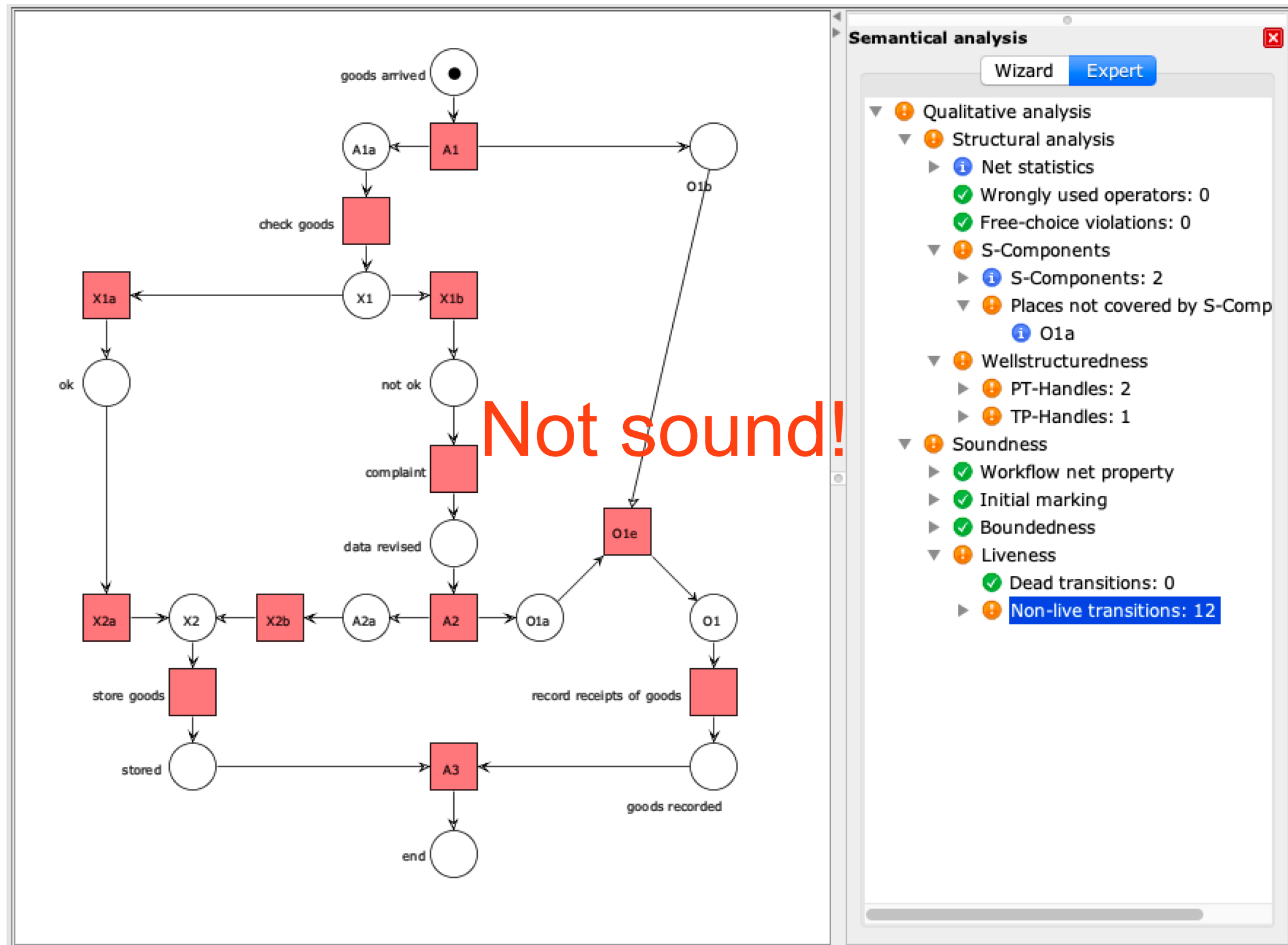
Can we repair the model?

Soundness analysis

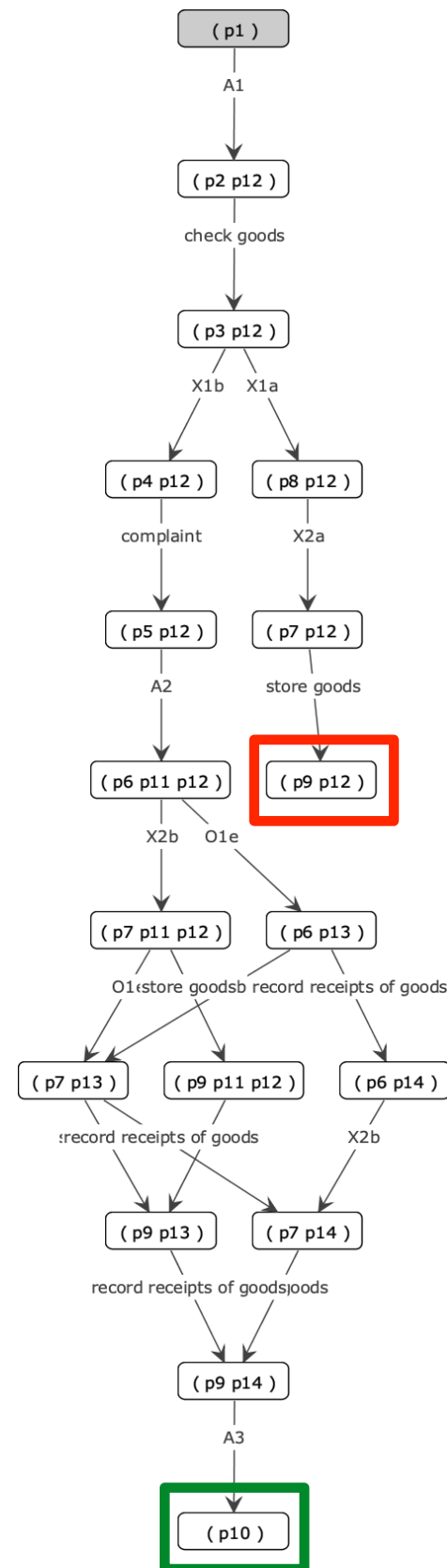


AND join
instead of
OR join?

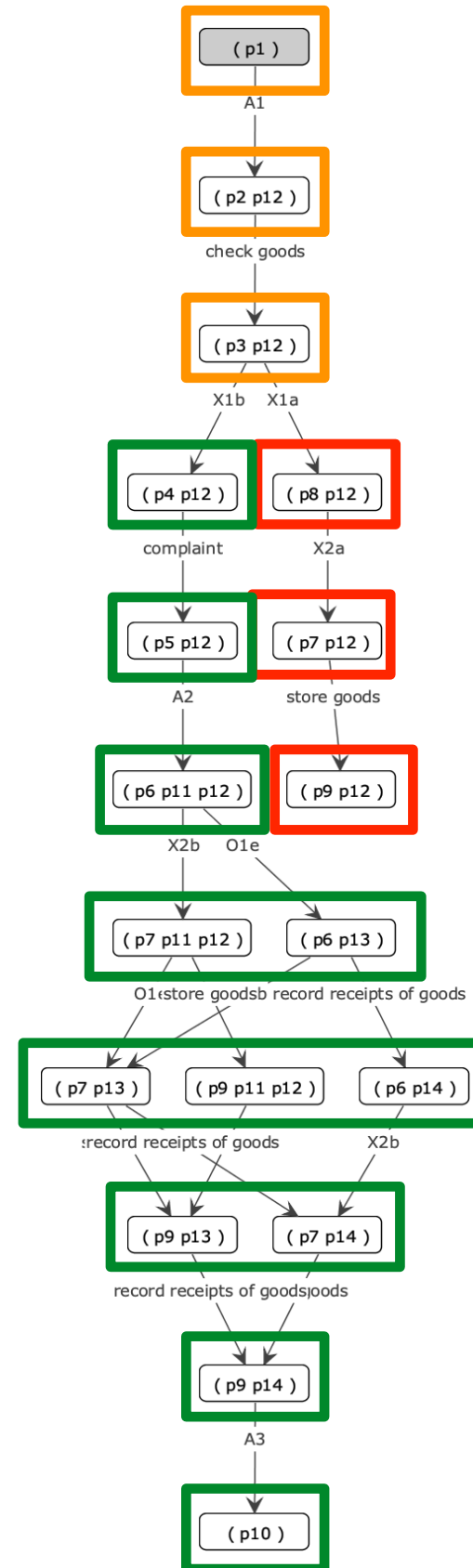
Soundness analysis



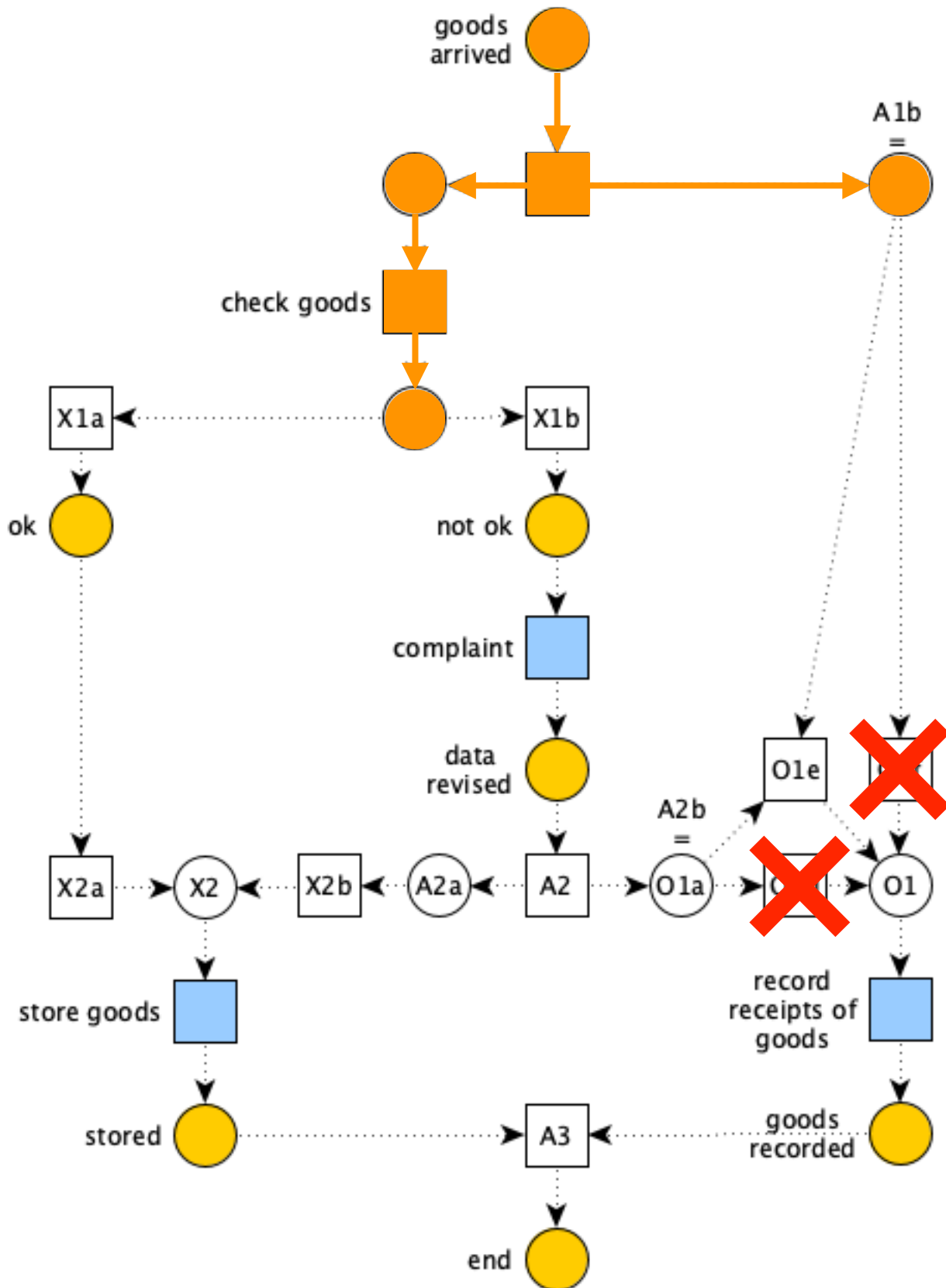
Soundness analysis



Soundness analysis



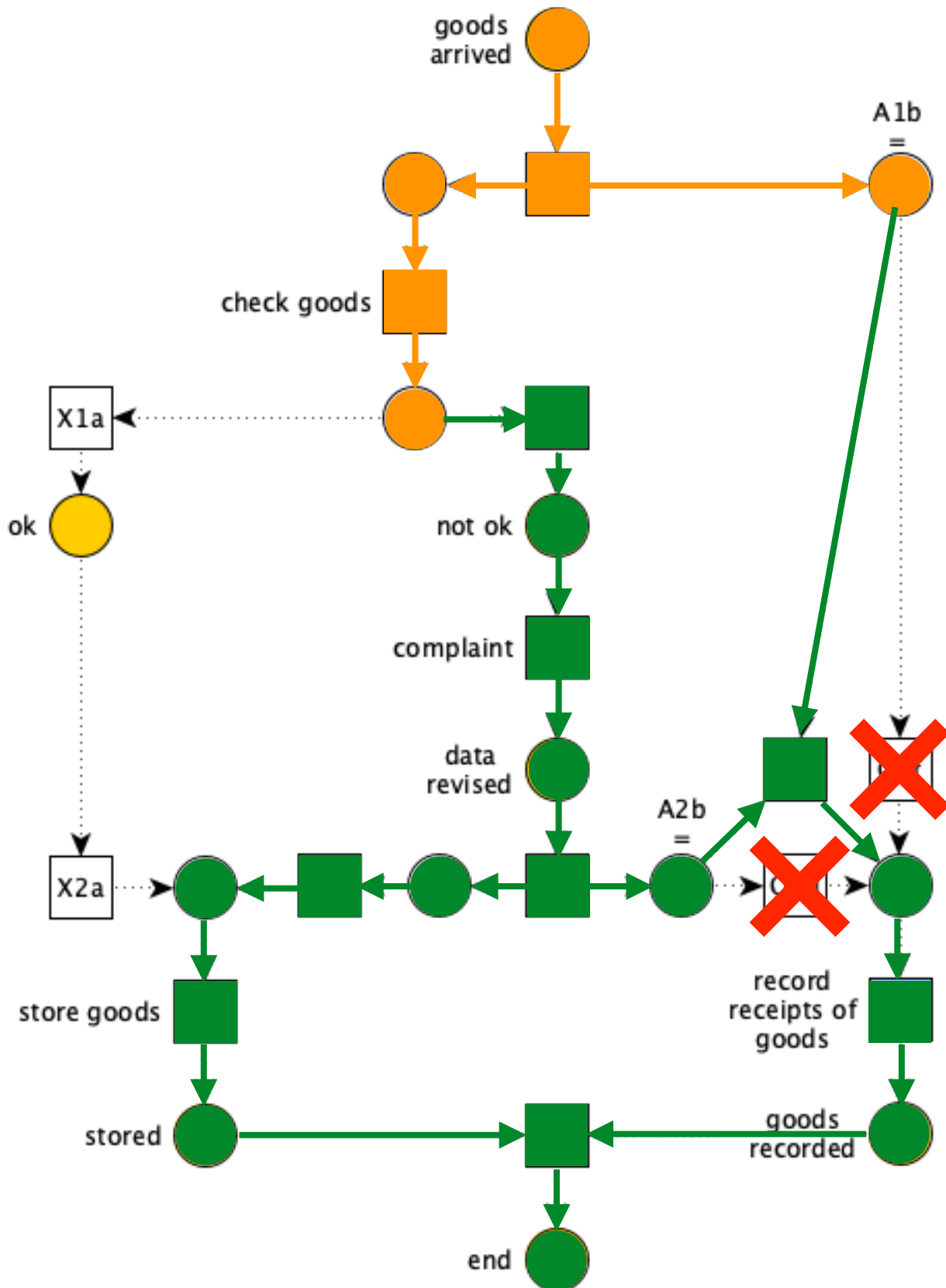
Soundness analysis



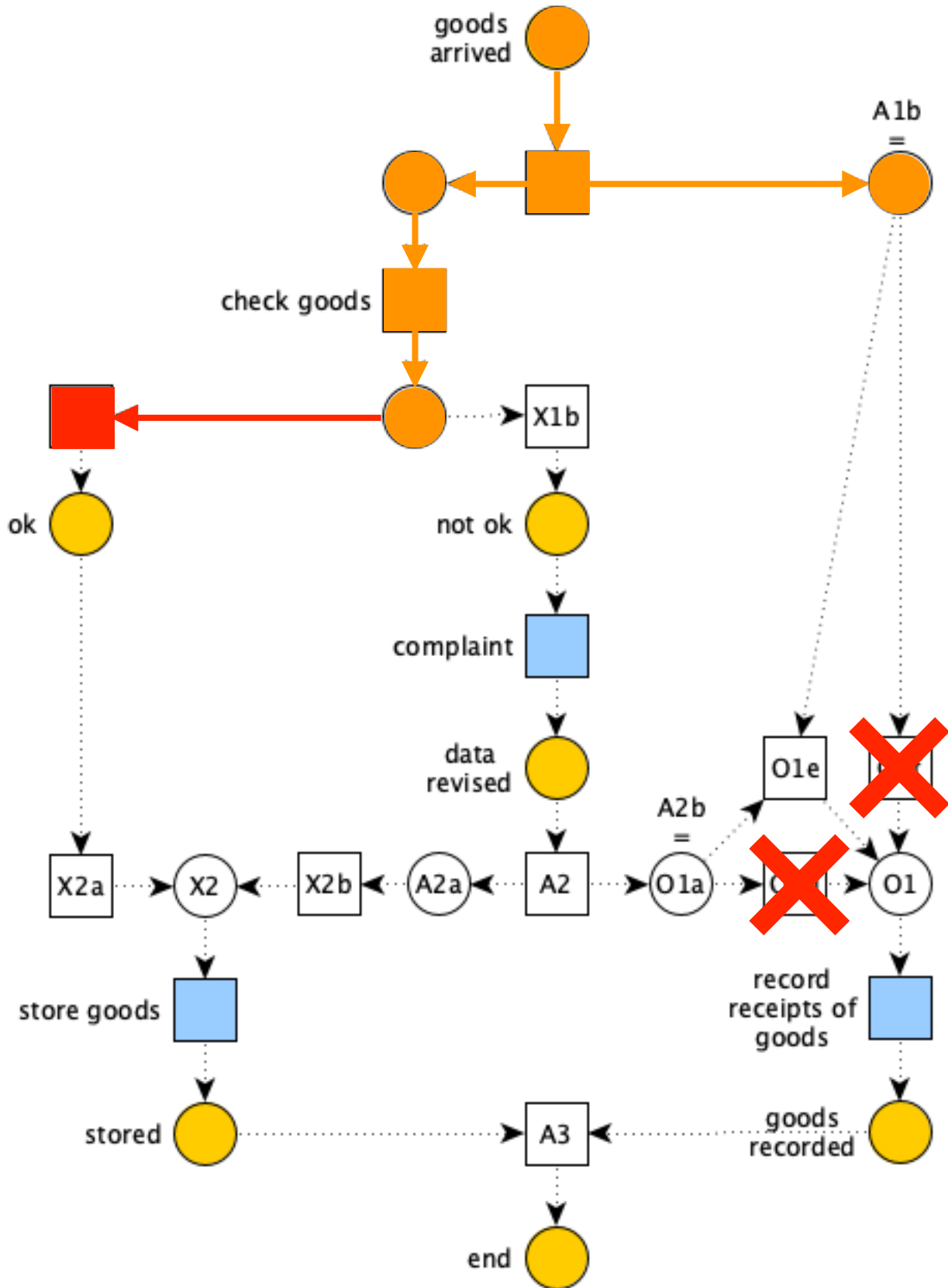
the right thing to do
would be to fire X1b

AND join
instead of
OR join?

Soundness analysis



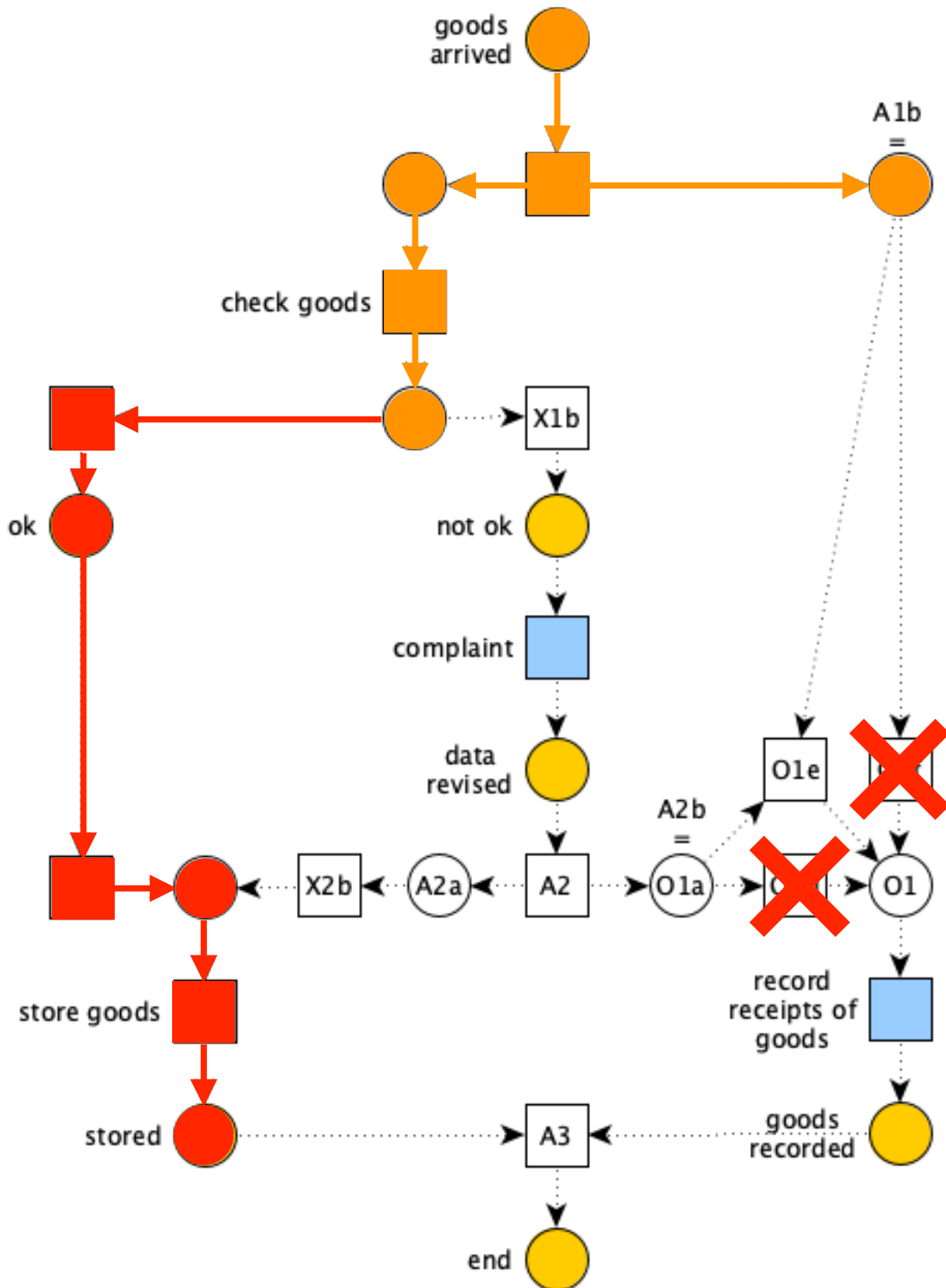
Soundness analysis



but X1a
is enabled as well

AND join
instead of
OR join?

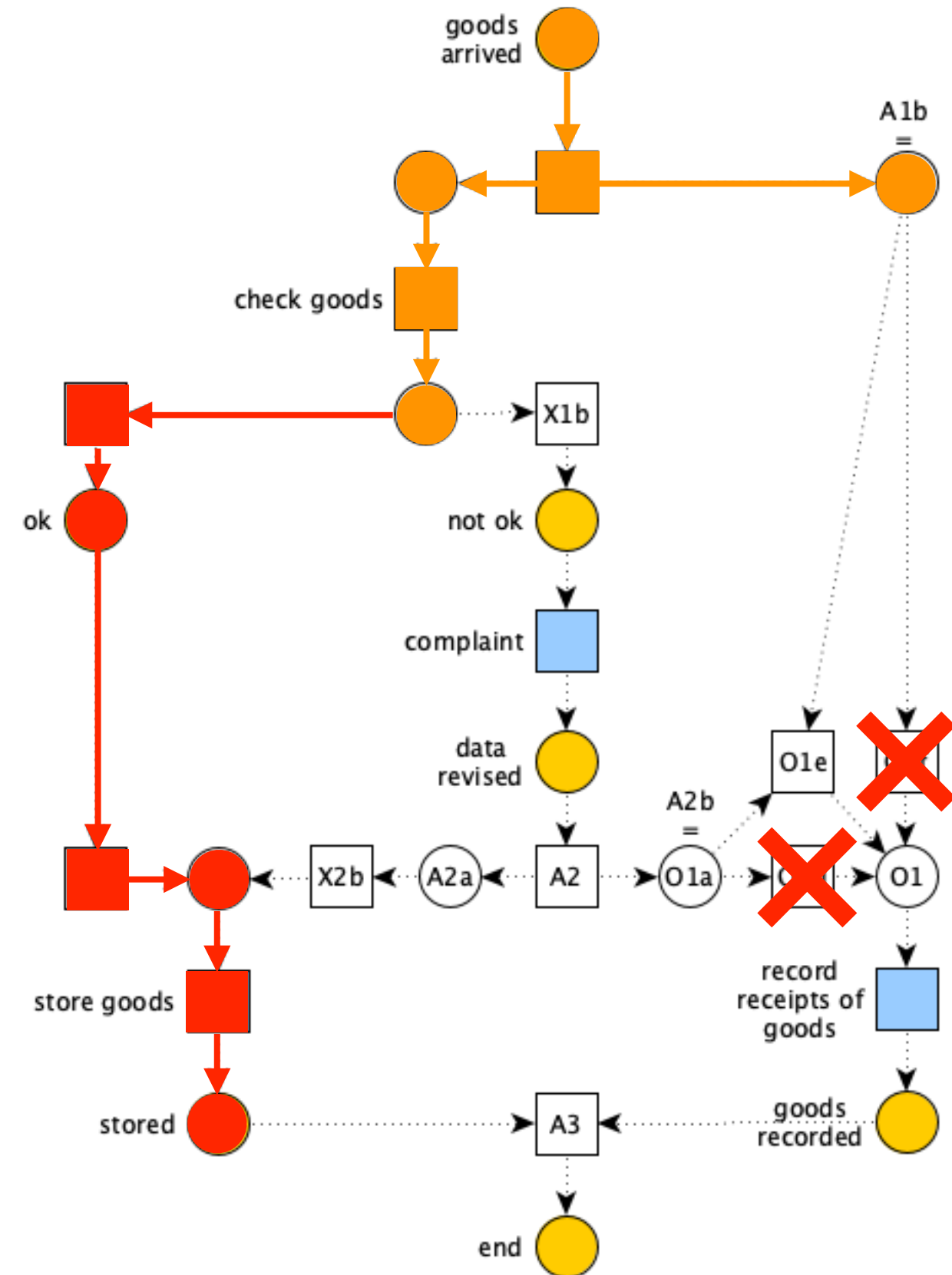
Soundness analysis



AND join
instead of
OR join?

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

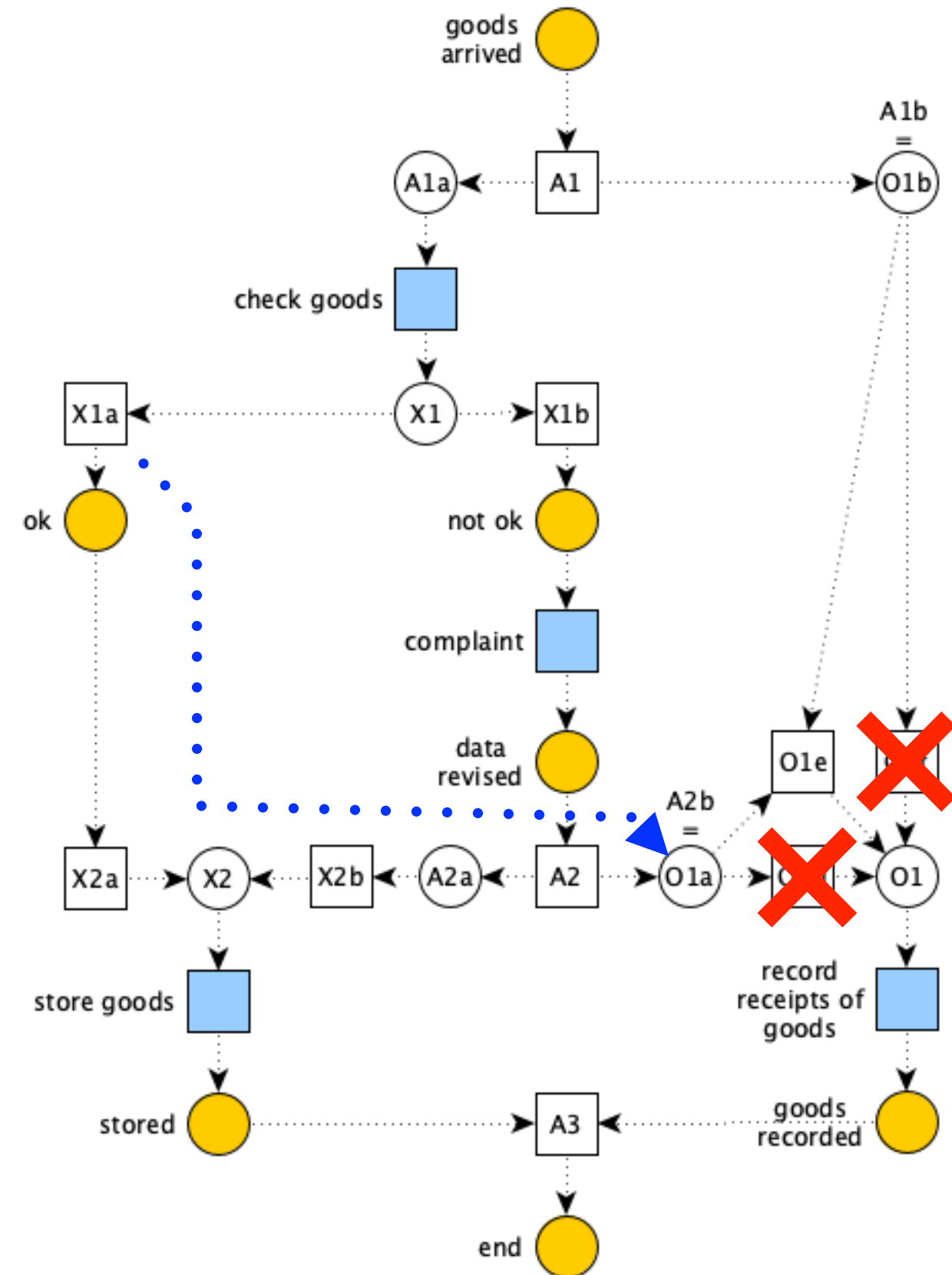
Soundness analysis



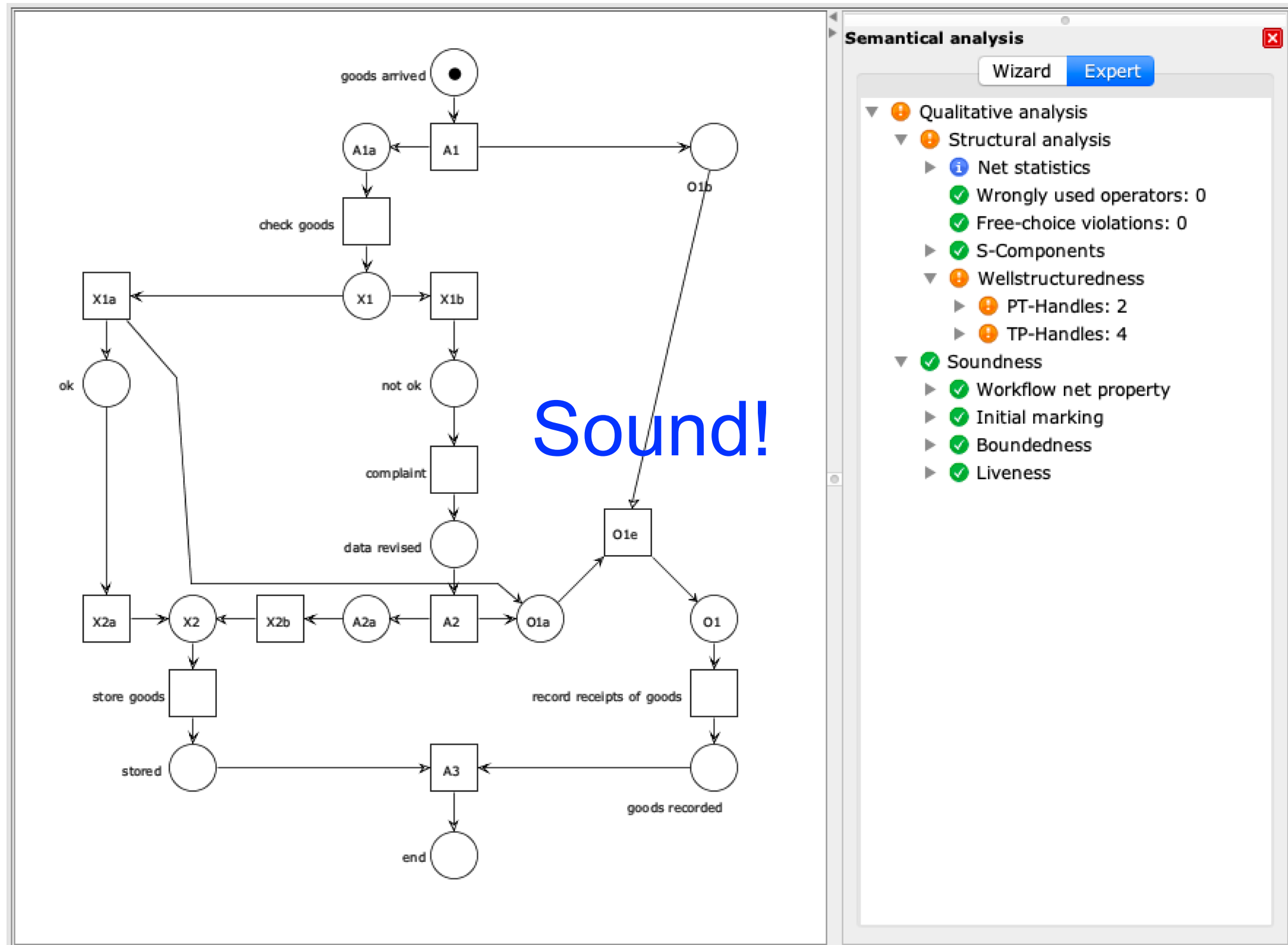
we miss a
token
in O1a

AND join
instead of
OR join
+ ad hoc flow?

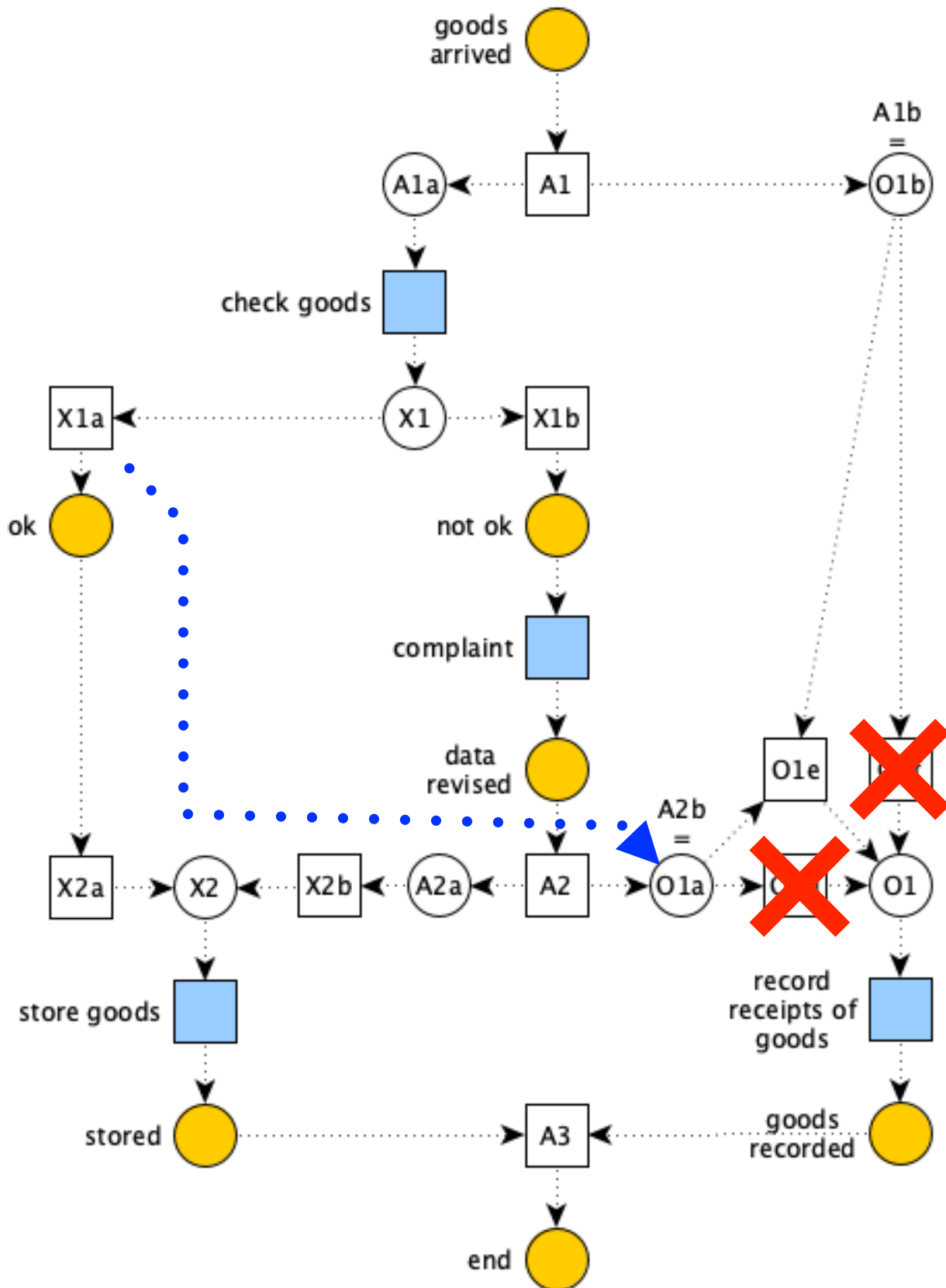
Soundness analysis



Soundness analysis

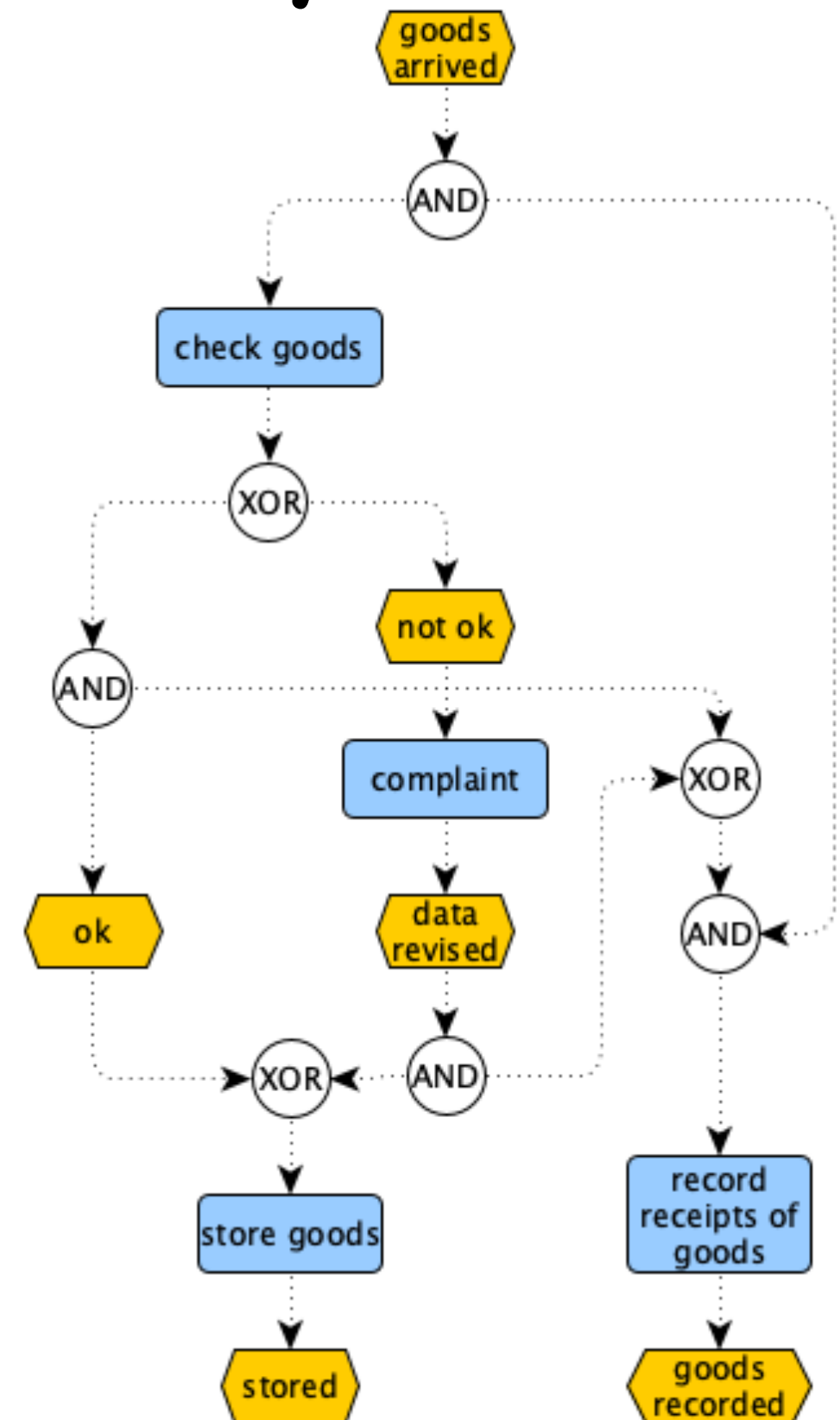
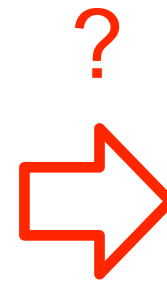
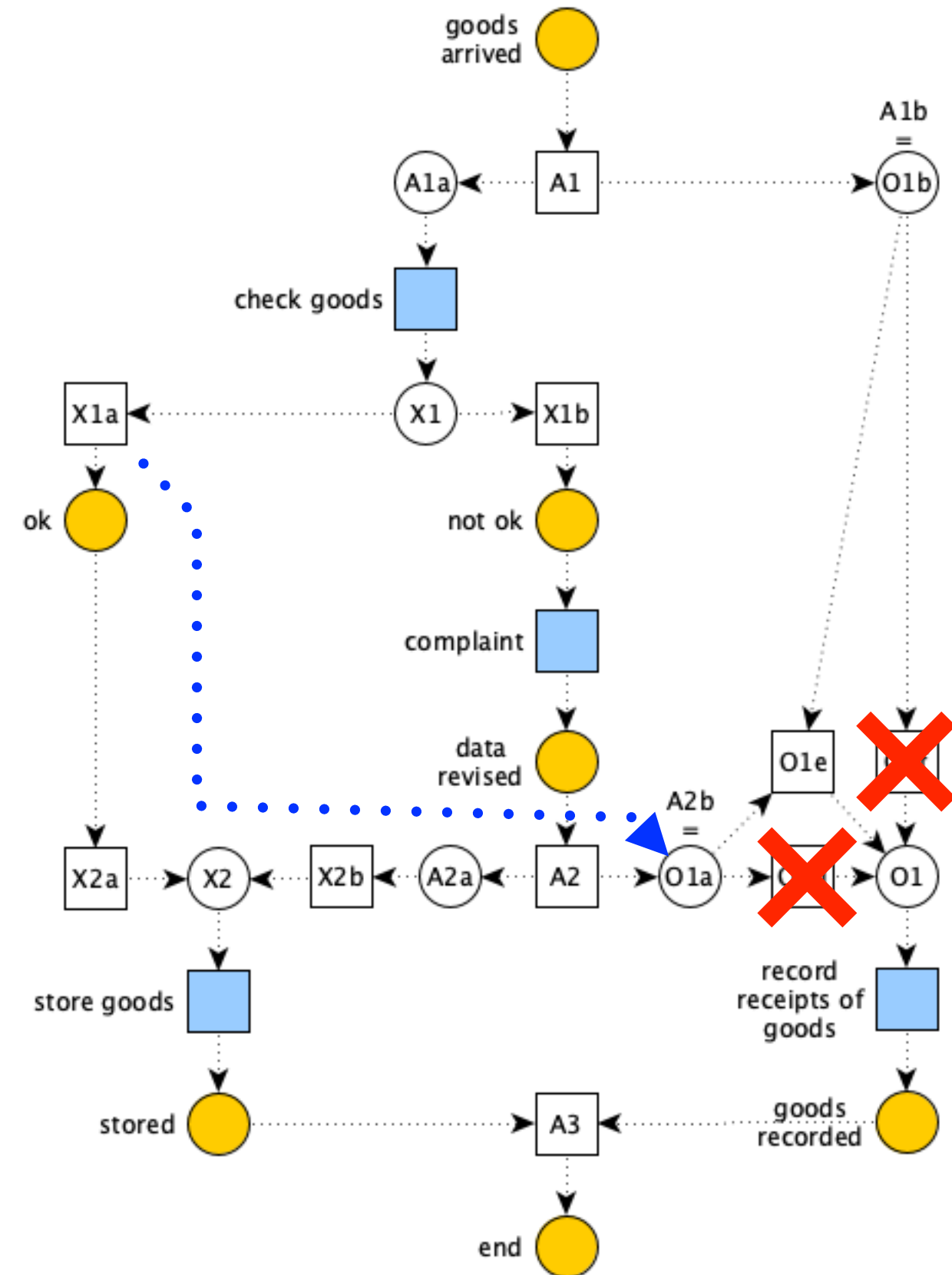


Soundness analysis

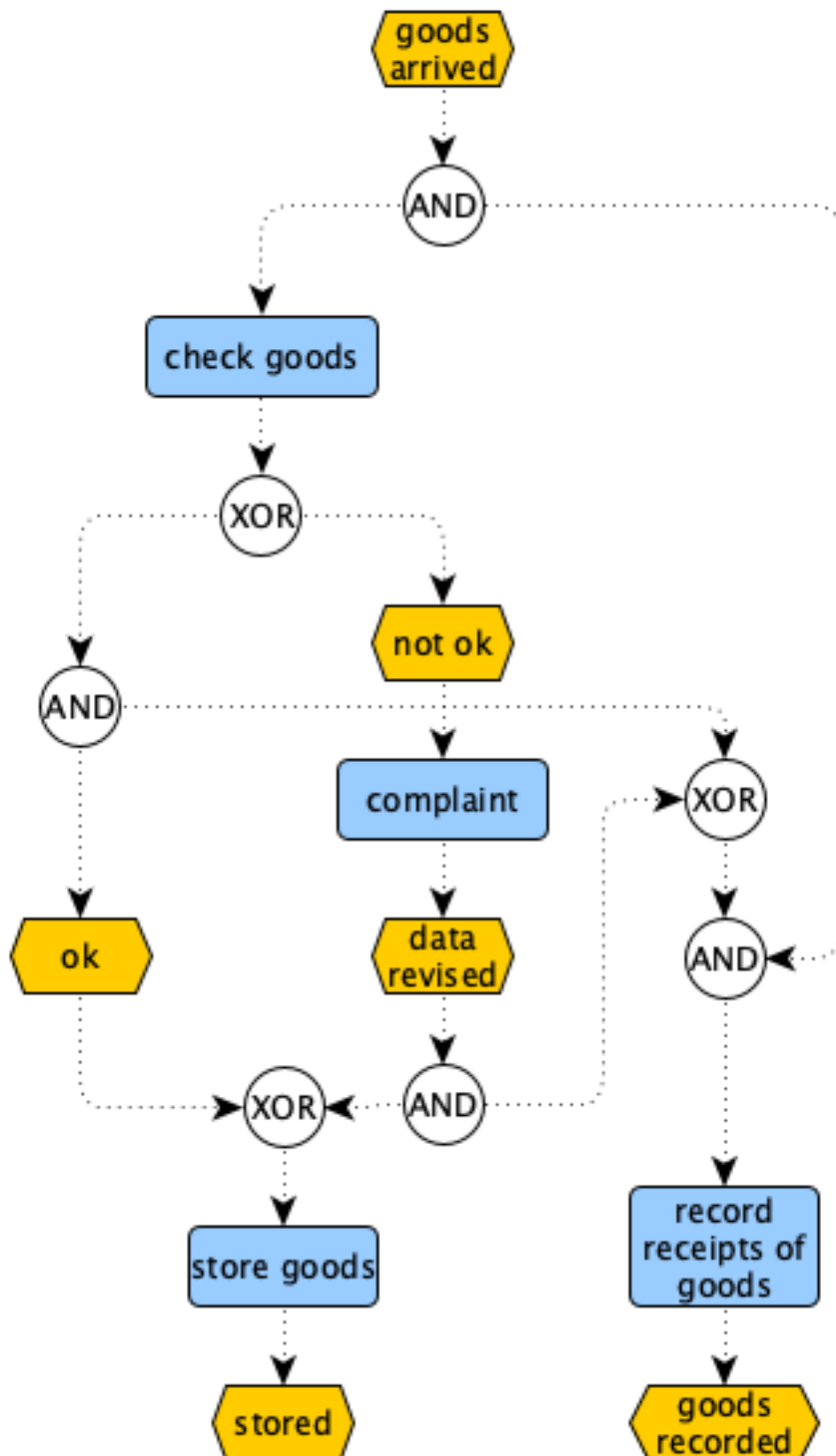


Sound, but...
we have repaired the wf net,
not the original EPC diagram!

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

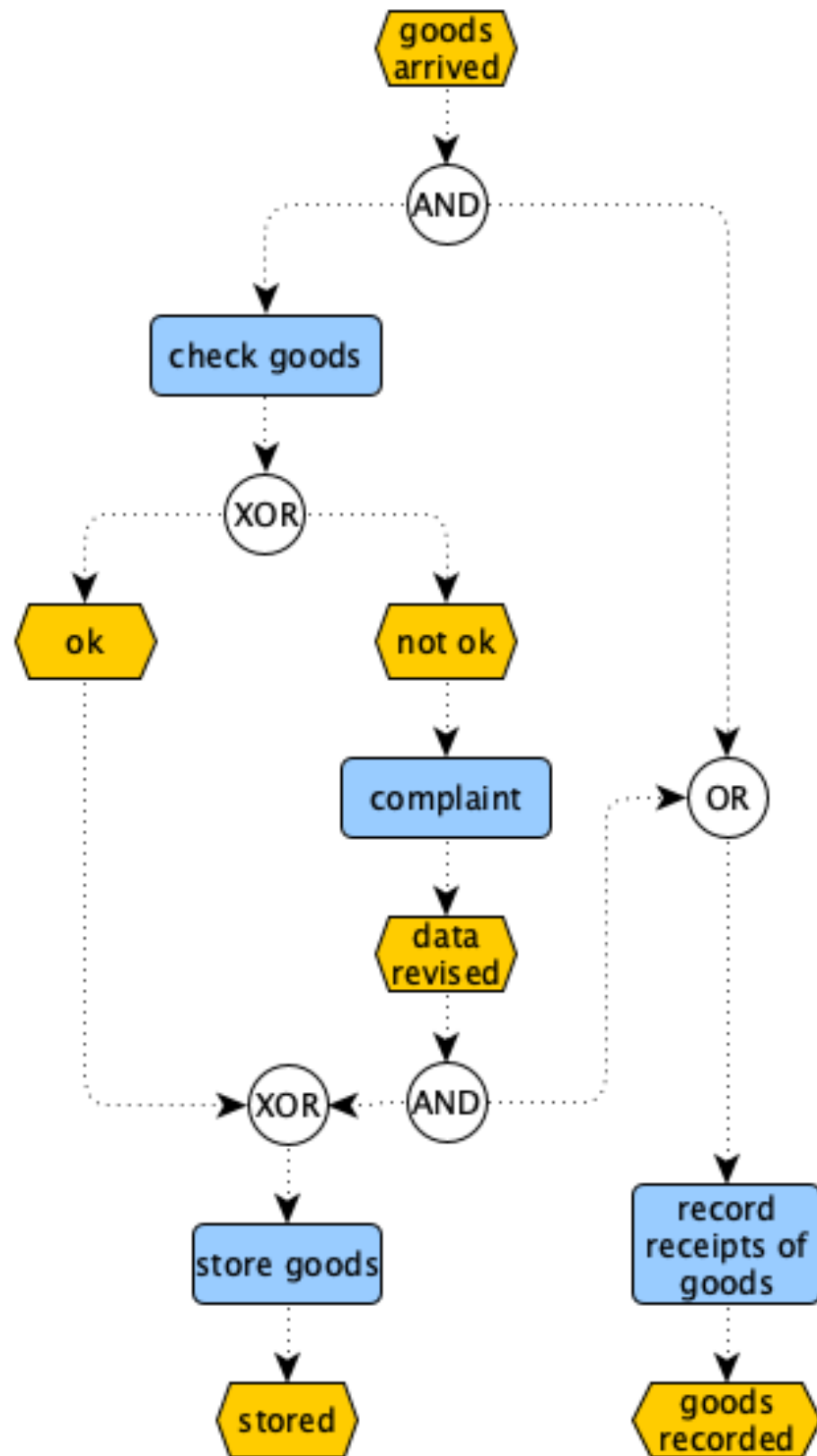
Relaxed soundness

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)

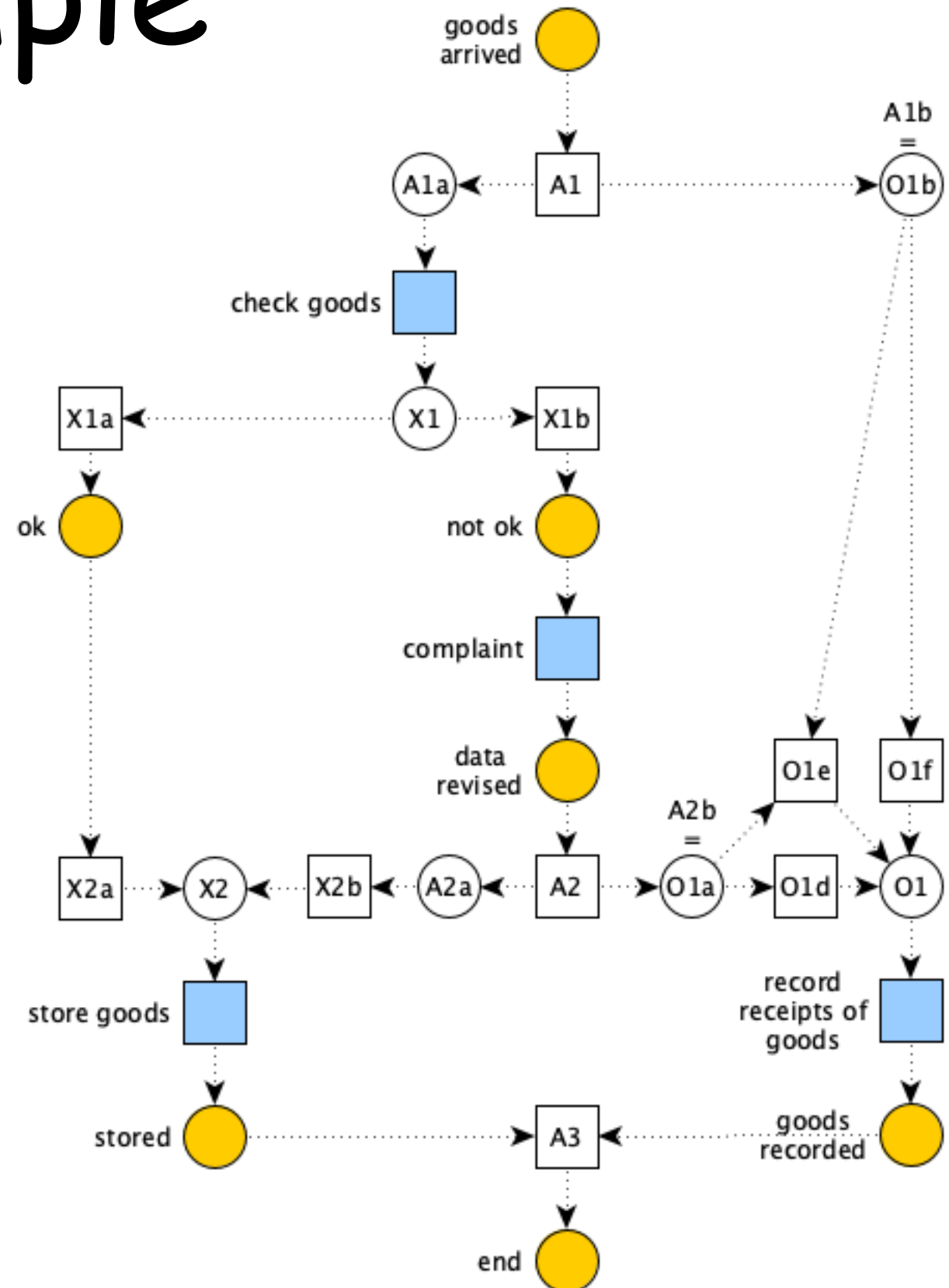
$$\forall t \in T. \exists \sigma \in L(N). \vec{\sigma}(t) > 0$$

Example



Relaxed
sound?

➡
Steps
1+2+3



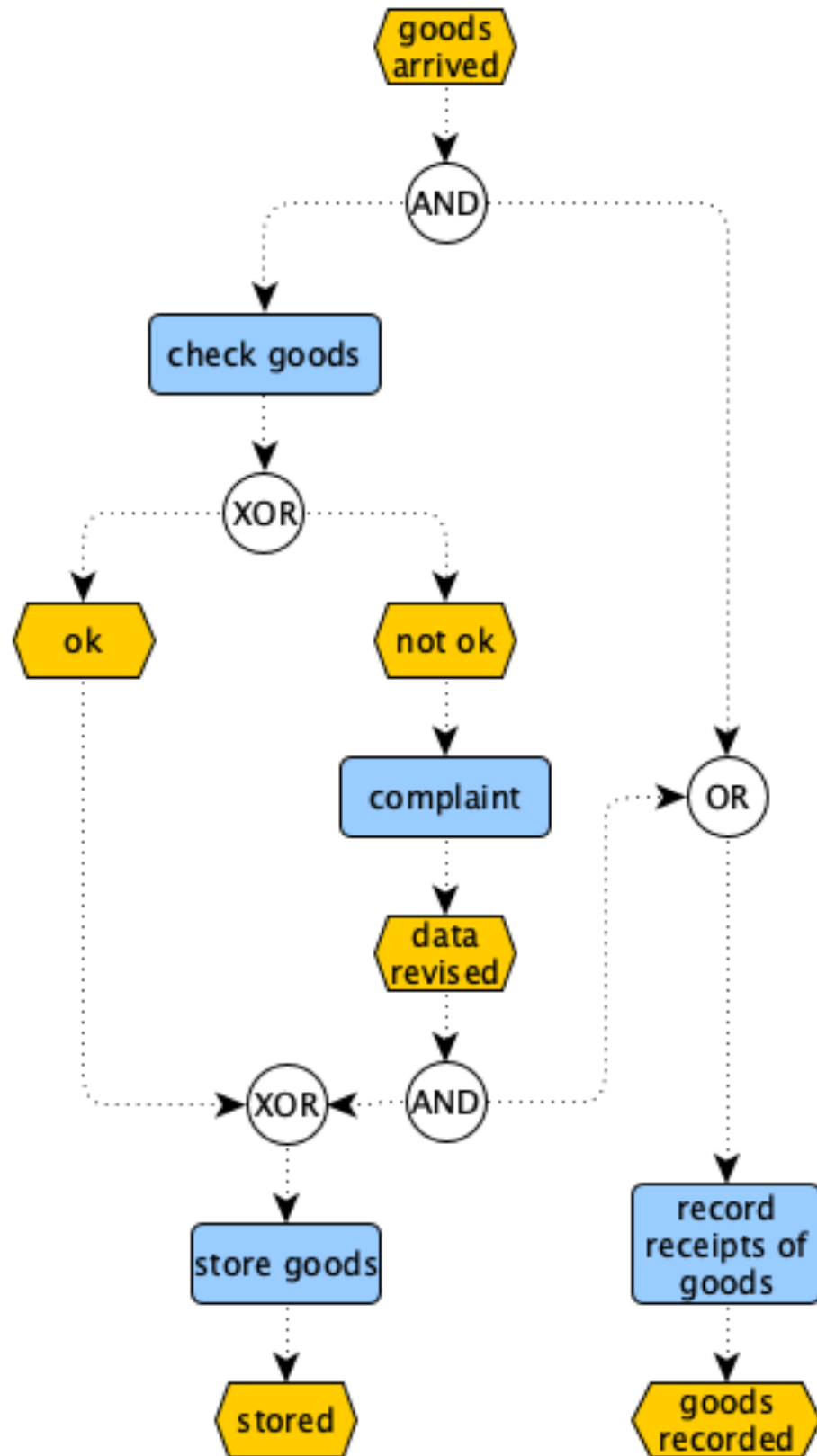
a sound execution



```

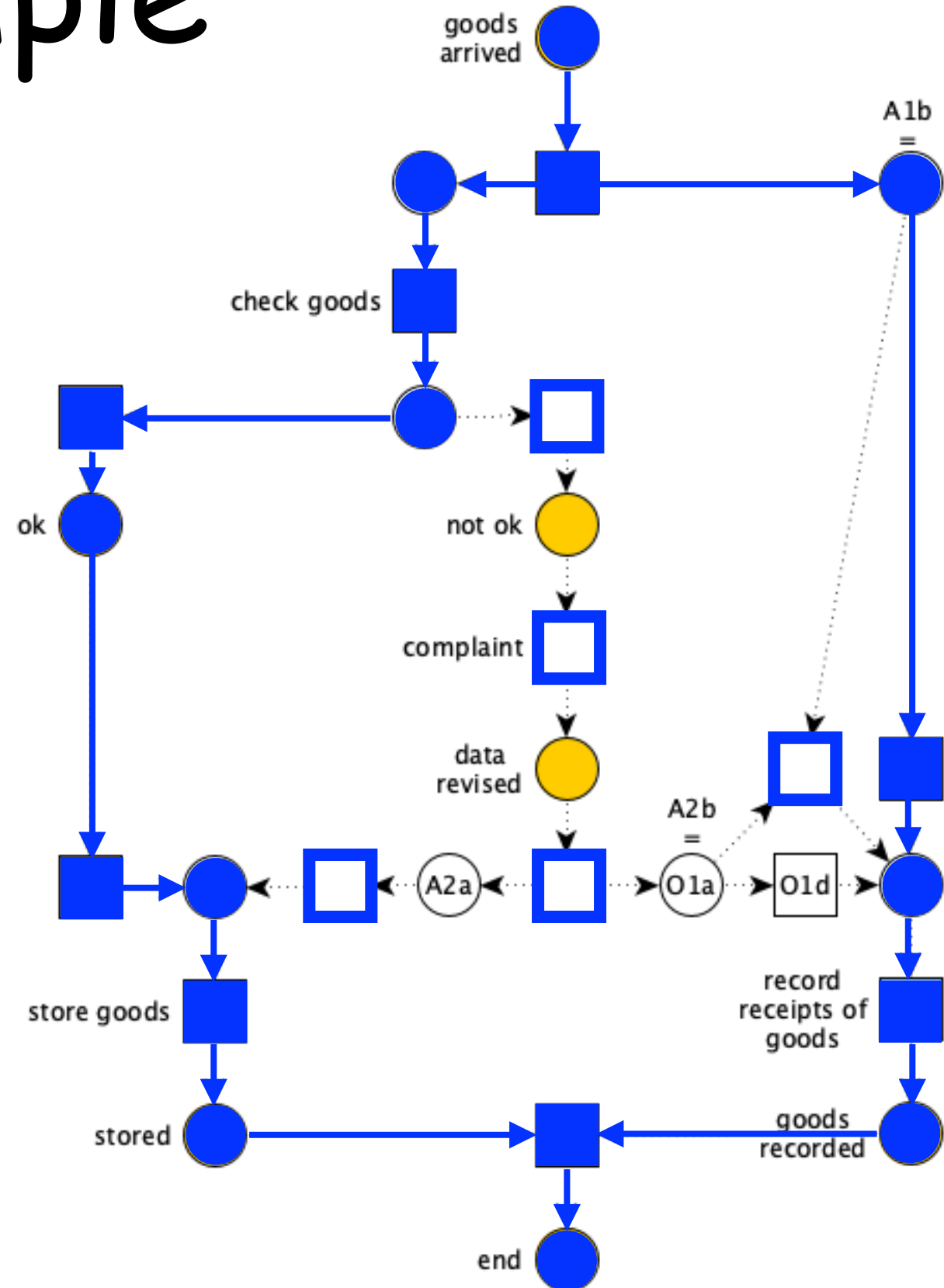
graph TD
    GA((goods arrived)) --> T1[ ]
    T1 --> C1(( ))
    T1 --> A1b((A1b = ))
    C1 --> T2[ ]
    T2 --> C2(( ))
    C2 --> T3[ ]
    T3 --> C3(( ))
    C3 --> T4[ ]
    T4 --> C4(( ))
    C4 --> T5[ ]
    T5 --> C5(( ))
    C5 --> T6[ ]
    T6 --> C6(( ))
    C6 --> T7[ ]
    T7 --> C7(( ))
    C7 --> T8[ ]
    T8 --> C8(( ))
    C8 --> T9[ ]
    T9 --> C9(( ))
    C9 --> T10[ ]
    T10 --> C10(( ))
    C10 --> T11[ ]
    T11 --> C11(( ))
    C11 --> T12[ ]
    T12 --> C12(( ))
    C12 --> T13[ ]
    T13 --> C13(( ))
    C13 --> T14[ ]
    T14 --> C14(( ))
    C14 --> T15[ ]
    T15 --> C15(( ))
    C15 --> T16[ ]
    T16 --> C16(( ))
    C16 --> T17[ ]
    T17 --> C17(( ))
    C17 --> T18[ ]
    T18 --> C18(( ))
    C18 --> T19[ ]
    T19 --> C19(( ))
    C19 --> T20[ ]
    T20 --> C20(( ))
    C20 --> T21[ ]
    T21 --> C21(( ))
    C21 --> T22[ ]
    T22 --> C22(( ))
    C22 --> T23[ ]
    T23 --> C23(( ))
    C23 --> T24[ ]
    T24 --> C24(( ))
    C24 --> T25[ ]
    T25 --> C25(( ))
    C25 --> T26[ ]
    T26 --> C26(( ))
    C26 --> T27[ ]
    T27 --> C27(( ))
    C27 --> T28[ ]
    T28 --> C28(( ))
    C28 --> T29[ ]
    T29 --> C29(( ))
    C29 --> T30[ ]
    T30 --> C30(( ))
    C30 --> T31[ ]
    T31 --> C31(( ))
    C31 --> T32[ ]
    T32 --> C32(( ))
    C32 --> T33[ ]
    T33 --> C33(( ))
    C33 --> T34[ ]
    T34 --> C34(( ))
    C34 --> T35[ ]
    T35 --> C35(( ))
    C35 --> T36[ ]
    T36 --> C36(( ))
    C36 --> T37[ ]
    T37 --> C37(( ))
    C37 --> T38[ ]
    T38 --> C38(( ))
    C38 --> T39[ ]
    T39 --> C39(( ))
    C39 --> T40[ ]
    T40 --> C40(( ))
    C40 --> T41[ ]
    T41 --> C41(( ))
    C41 --> T42[ ]
    T42 --> C42(( ))
    C42 --> T43[ ]
    T43 --> C43(( ))
    C43 --> T44[ ]
    T44 --> C44(( ))
    C44 --> T45[ ]
    T45 --> C45(( ))
    C45 --> T46[ ]
    T46 --> C46(( ))
    C46 --> T47[ ]
    T47 --> C47(( ))
    C47 --> T48[ ]
    T48 --> C48(( ))
    C48 --> T49[ ]
    T49 --> C49(( ))
    C49 --> T50[ ]
    T50 --> C50(( ))
    C50 --> T51[ ]
    T51 --> C51(( ))
    C51 --> T52[ ]
    T52 --> C52(( ))
    C52 --> T53[ ]
    T53 --> C53(( ))
    C53 --> T54[ ]
    T54 --> C54(( ))
    C54 --> T55[ ]
    T55 --> C55(( ))
    C55 --> T56[ ]
    T56 --> C56(( ))
    C56 --> T57[ ]
    T57 --> C57(( ))
    C57 --> T58[ ]
    T58 --> C58(( ))
    C58 --> T59[ ]
    T59 --> C59(( ))
    C59 --> T60[ ]
    T60 --> C60(( ))
    C60 --> T61[ ]
    T61 --> C61(( ))
    C61 --> T62[ ]
    T62 --> C62(( ))
    C62 --> T63[ ]
    T63 --> C63(( ))
    C63 --> T64[ ]
    T64 --> C64(( ))
    C64 --> T65[ ]
    T65 --> C65(( ))
    C65 --> T66[ ]
    T66 --> C66(( ))
    C66 --> T67[ ]
    T67 --> C67(( ))
    C67 --> T68[ ]
    T68 --> C68(( ))
    C68 --> T69[ ]
    T69 --> C69(( ))
    C69 --> T70[ ]
    T70 --> C70(( ))
    C70 --> T71[ ]
    T71 --> C71(( ))
    C71 --> T72[ ]
    T72 --> C72(( ))
    C72 --> T73[ ]
    T73 --> C73(( ))
    C73 --> T74[ ]
    T74 --> C74(( ))
    C74 --> T75[ ]
    T75 --> C75(( ))
    C75 --> T76[ ]
    T76 --> C76(( ))
    C76 --> T77[ ]
    T77 --> C77(( ))
    C77 --> T78[ ]
    T78 --> C78(( ))
    C78 --> T79[ ]
    T79 --> C79(( ))
    C79 --> T80[ ]
    T80 --> C80(( ))
    C80 --> T81[ ]
    T81 --> C81(( ))
    C81 --> T82[ ]
    T82 --> C82(( ))
    C82 --> T83[ ]
    T83 --> C83(( ))
    C83 --> T84[ ]
    T84 --> C84(( ))
    C84 --> T85[ ]
    T85 --> C85(( ))
    C85 --> T86[ ]
    T86 --> C86(( ))
    C86 --> T87[ ]
    T87 --> C87(( ))
    C87 --> T88[ ]
    T88 --> C88(( ))
    C88 --> T89[ ]
    T89 --> C89(( ))
    C89 --> T90[ ]
    T90 --> C90(( ))
    C90 --> T91[ ]
    T91 --> C91(( ))
    C91 --> T92[ ]
    T92 --> C92(( ))
    C92 --> T93[ ]
    T93 --> C93(( ))
    C93 --> T94[ ]
    T94 --> C94(( ))
    C94 --> T95[ ]
    T95 --> C95(( ))
    C95 --> T96[ ]
    T96 --> C96(( ))
    C96 --> T97[ ]
    T97 --> C97(( ))
    C97 --> T98[ ]
    T98 --> C98(( ))
    C98 --> T99[ ]
    T99 --> C99(( ))
    C99 --> T100[ ]
    T100 --> C100(( ))
    C100 --> T101[ ]
    T101 --> C101(( ))
    C101 --> T102[ ]
    T102 --> C102(( ))
    C102 --> T103[ ]
    T103 --> C103(( ))
    C103 --> T104[ ]
    T104 --> C104(( ))
    C104 --> T105[ ]
    T105 --> C105(( ))
    C105 --> T106[ ]
    T106 --> C106(( ))
    C106 --> T107[ ]
    T107 --> C107(( ))
    C107 --> T108[ ]
    T108 --> C108(( ))
    C108 --> T109[ ]
    T109 --> C109(( ))
    C109 --> T110[ ]
    T110 --> C110(( ))
    C110 --> T111[ ]
    T111 --> C111(( ))
    C111 --> T112[ ]
    T112 --> C112(( ))
    C112 --> T113[ ]
    T113 --> C113(( ))
    C113 --> T114[ ]
    T114 --> C114(( ))
    C114 --> T115[ ]
    T115 --> C115(( ))
    C115 --> T116[ ]
    T116 --> C116(( ))
    C116 --> T117[ ]
    T117 --> C117(( ))
    C117 --> T118[ ]
    T118 --> C118(( ))
    C118 --> T119[ ]
    T119 --> C119(( ))
    C119 --> T120[ ]
    T120 --> C120(( ))
    C120 --> T121[ ]
    T121 --> C121(( ))
    C121 --> T122[ ]
    T122 --> C122(( ))
    C122 --> T123[ ]
    T123 --> C123(( ))
    C123 --> T124[ ]
    T124 --> C124(( ))
    C124 --> T125[ ]
    T125 --> C125(( ))
    C125 --> T126[ ]
    T126 --> C126(( ))
    C126 --> T127[ ]
    T127 --> C127(( ))
    C127 --> T128[ ]
    T128 --> C128(( ))
    C128 --> T129[ ]
    T129 --> C129(( ))
    C129 --> T130[ ]
    T130 --> C130(( ))
    C130 --> T131[ ]
    T131 --> C131(( ))
    C131 --> T132[ ]
    T132 --> C132(( ))
    C132 --> T133[ ]
    T133 --> C133(( ))
    C133 --> T134[ ]
    T134 --> C134(( ))
    C134 --> T135[ ]
    T135 --> C135(( ))
    C135 --> T136[ ]
    T136 --> C136(( ))
    C136 --> T137[ ]
    T137 --> C137(( ))
    C137 --> T138[ ]
    T138 --> C
```

Example another sound execution



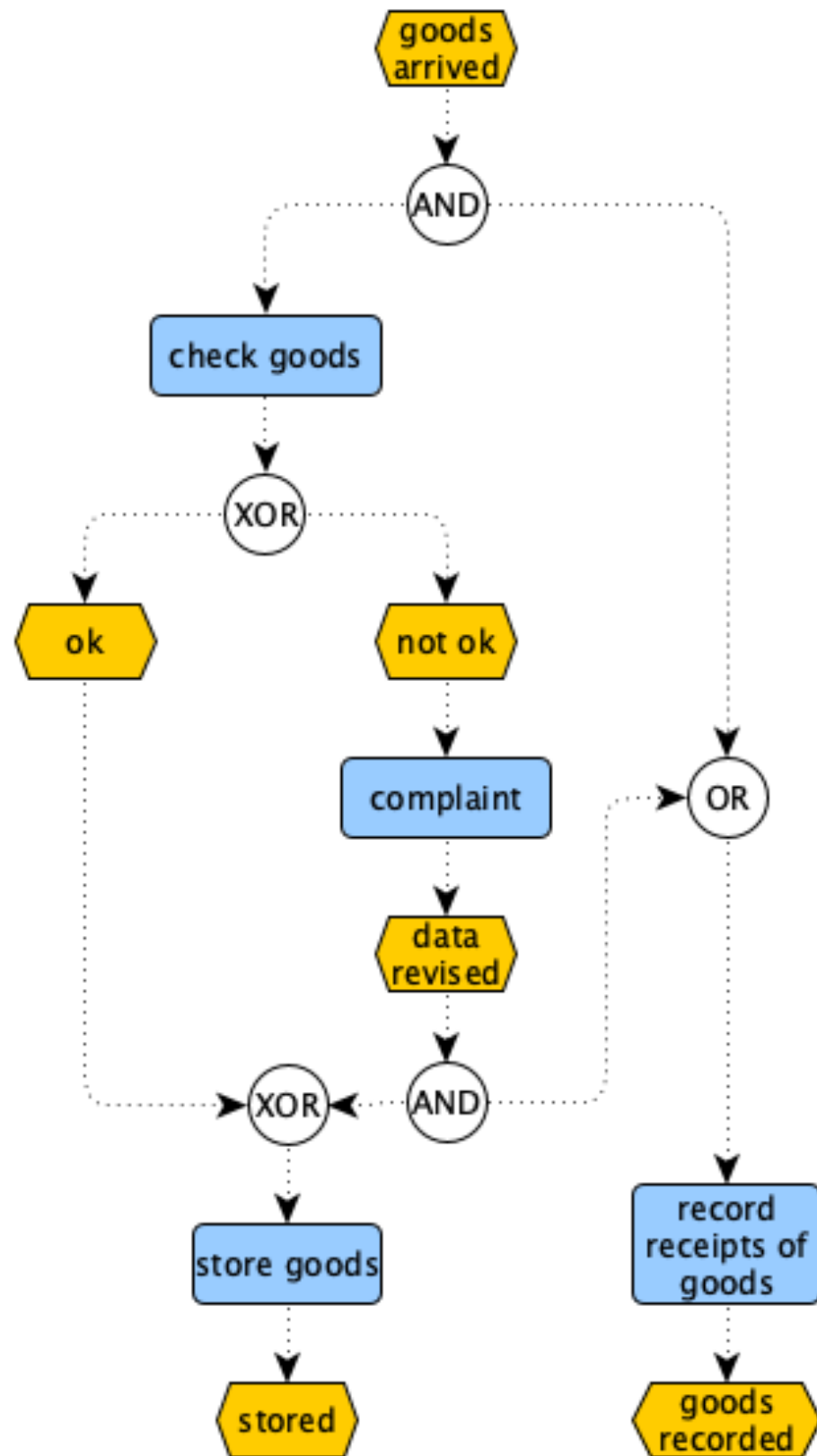
Relaxed
sound?

➡
Steps
1+2+3



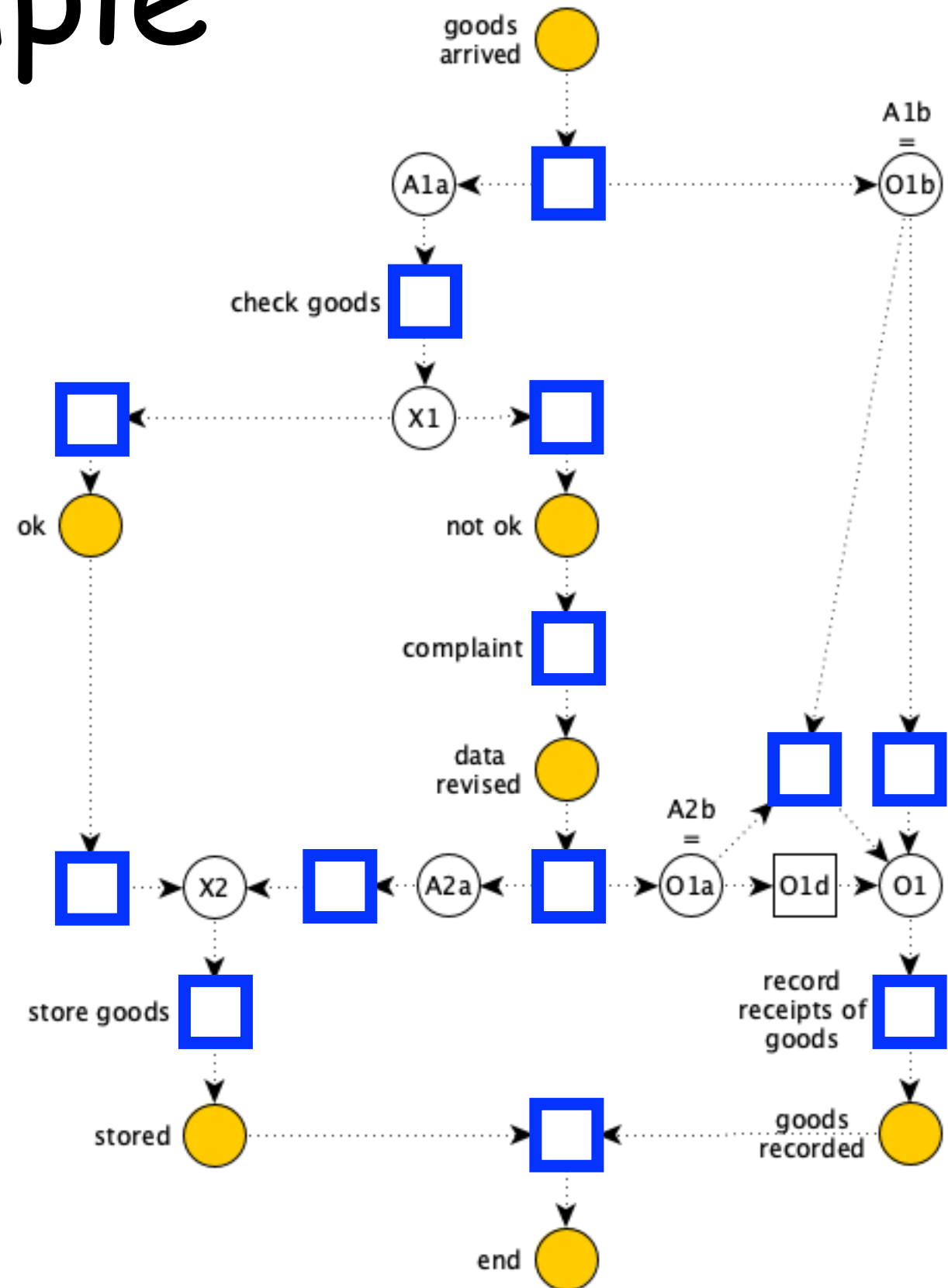
tasks involved in
some sound execution

Example



Relaxed
sound?

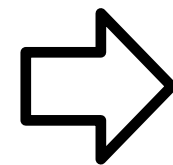
➡
Steps
1+2+3



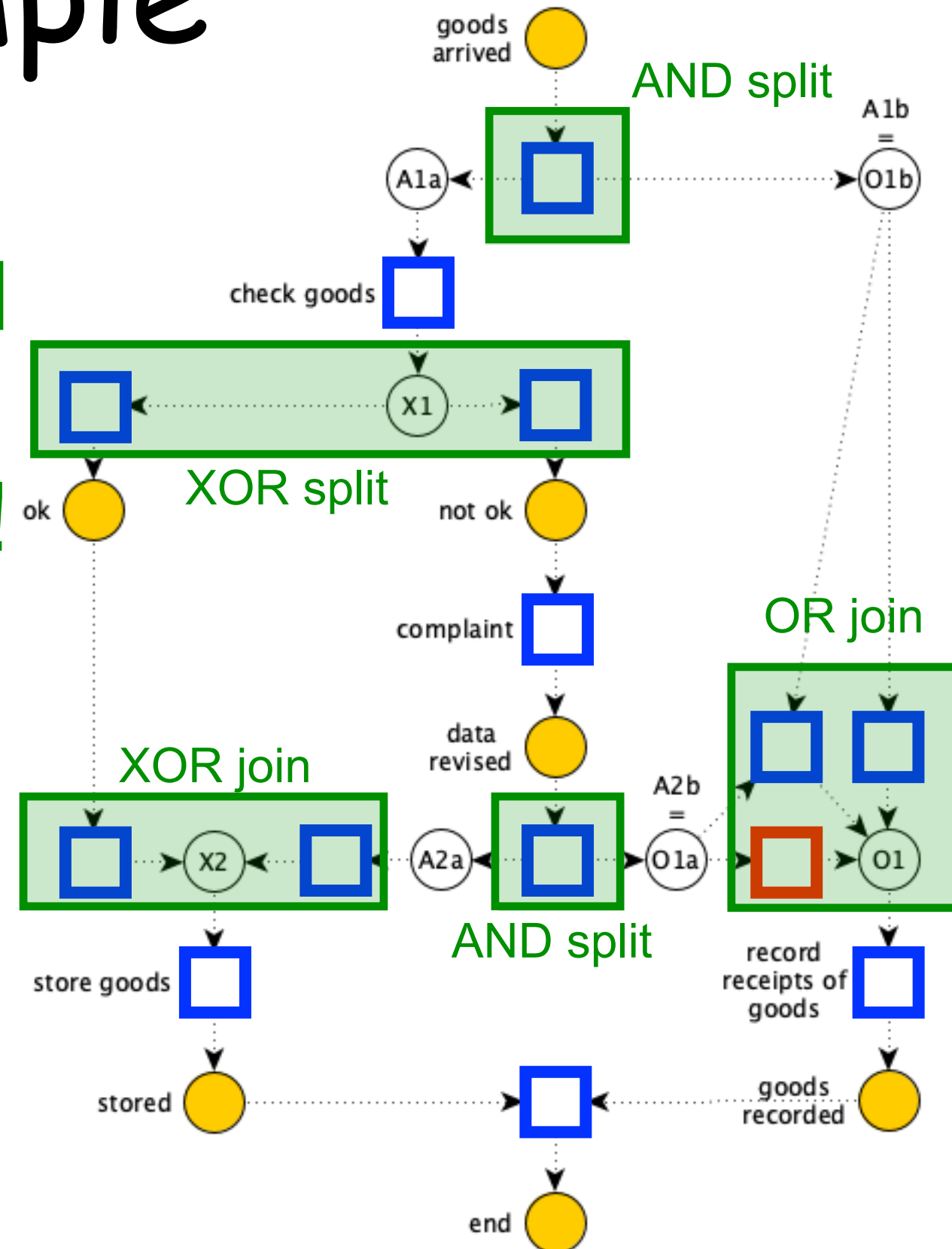
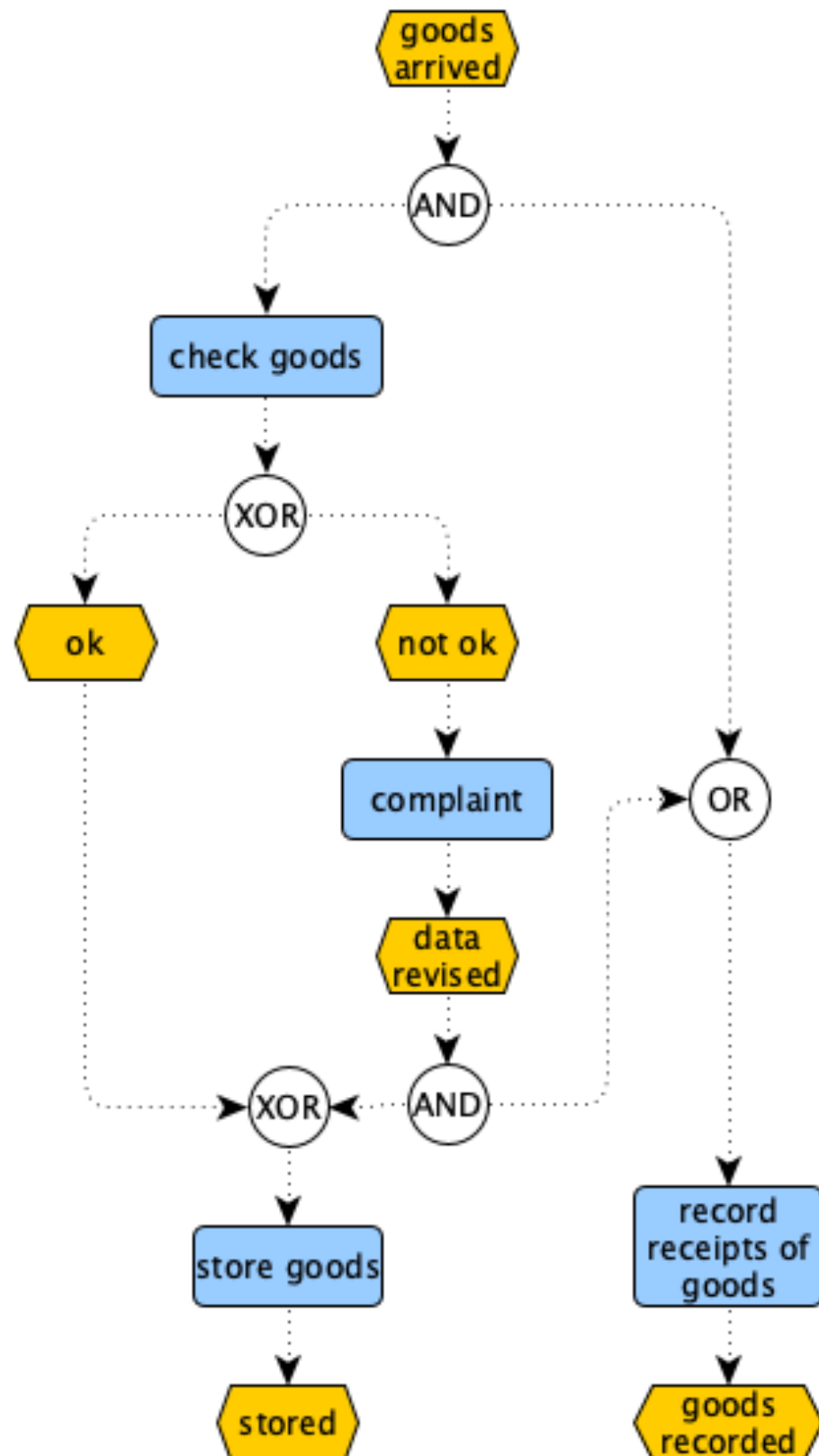
all EPC nodes involved in
some sound execution

Example

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

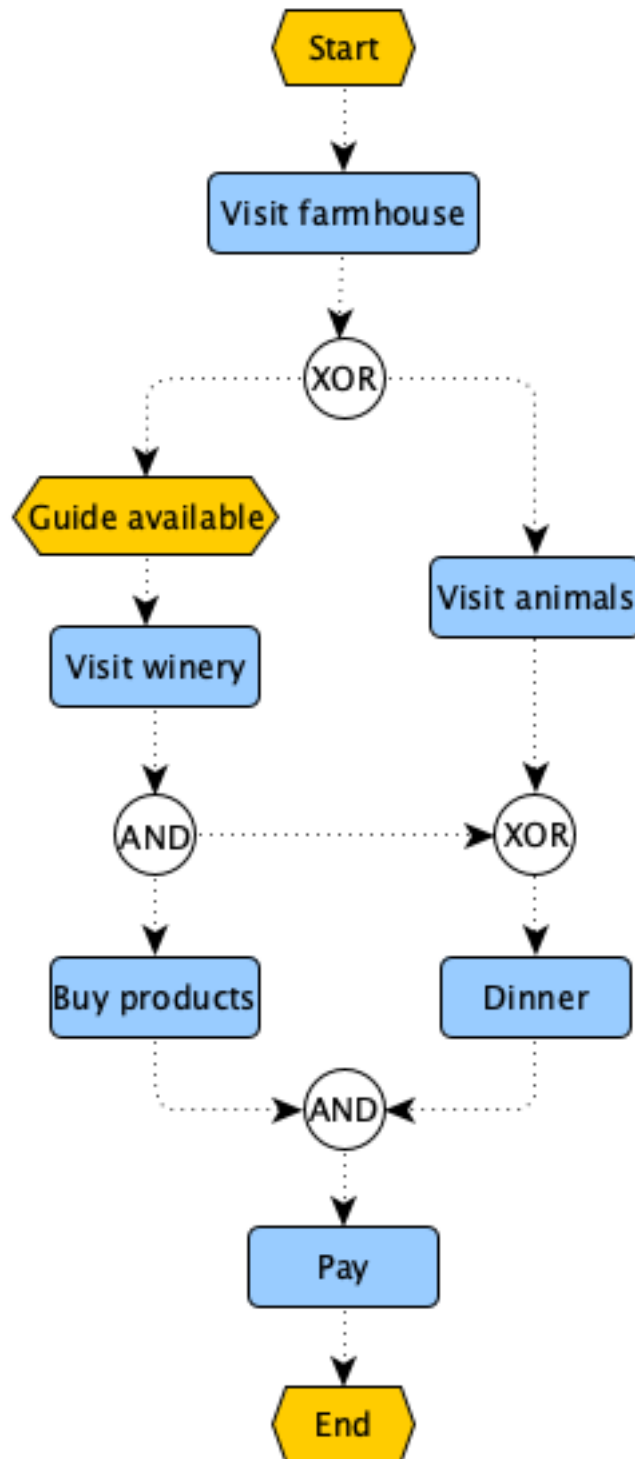
(also along paths between two connectors)

(fusion not needed, dummy places/transitions not needed)

OR-connectors are not present

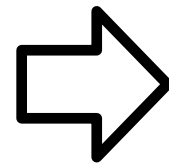
(avoid intrinsic problems with OR join)

OR-connectors
are not present
alternation
is not satisfied

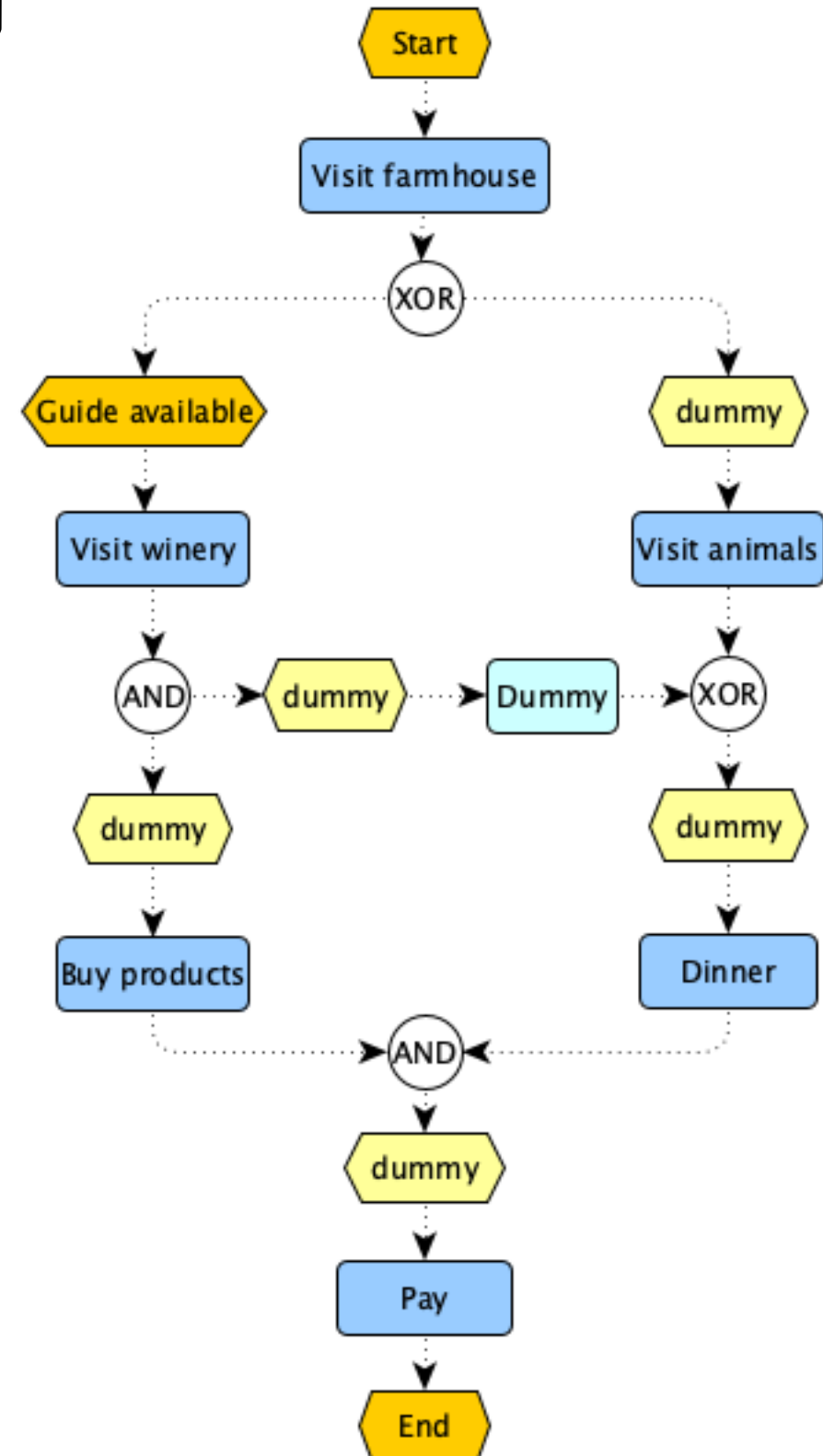


Example

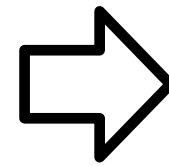
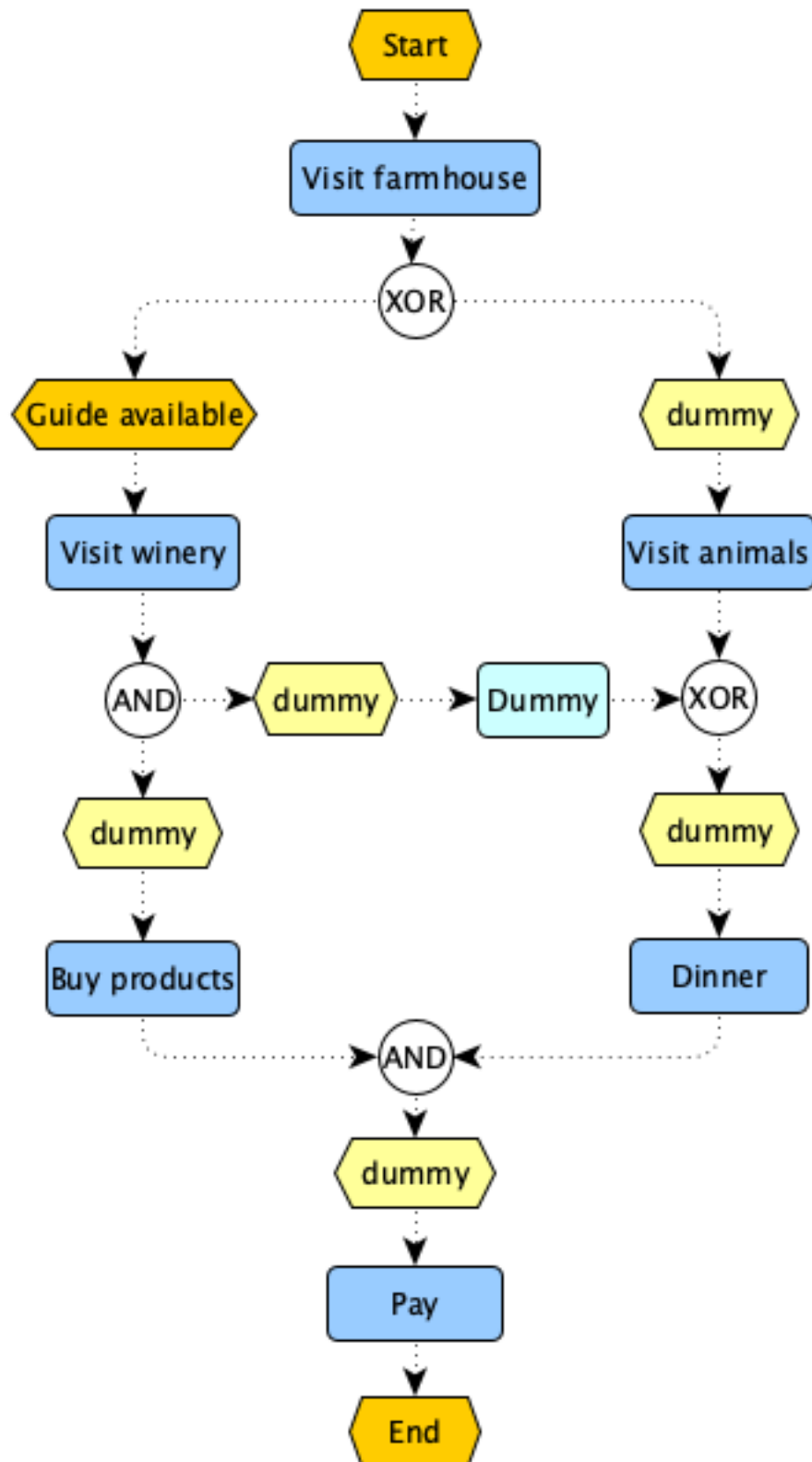
Add dummy events
and functions
to force alternation



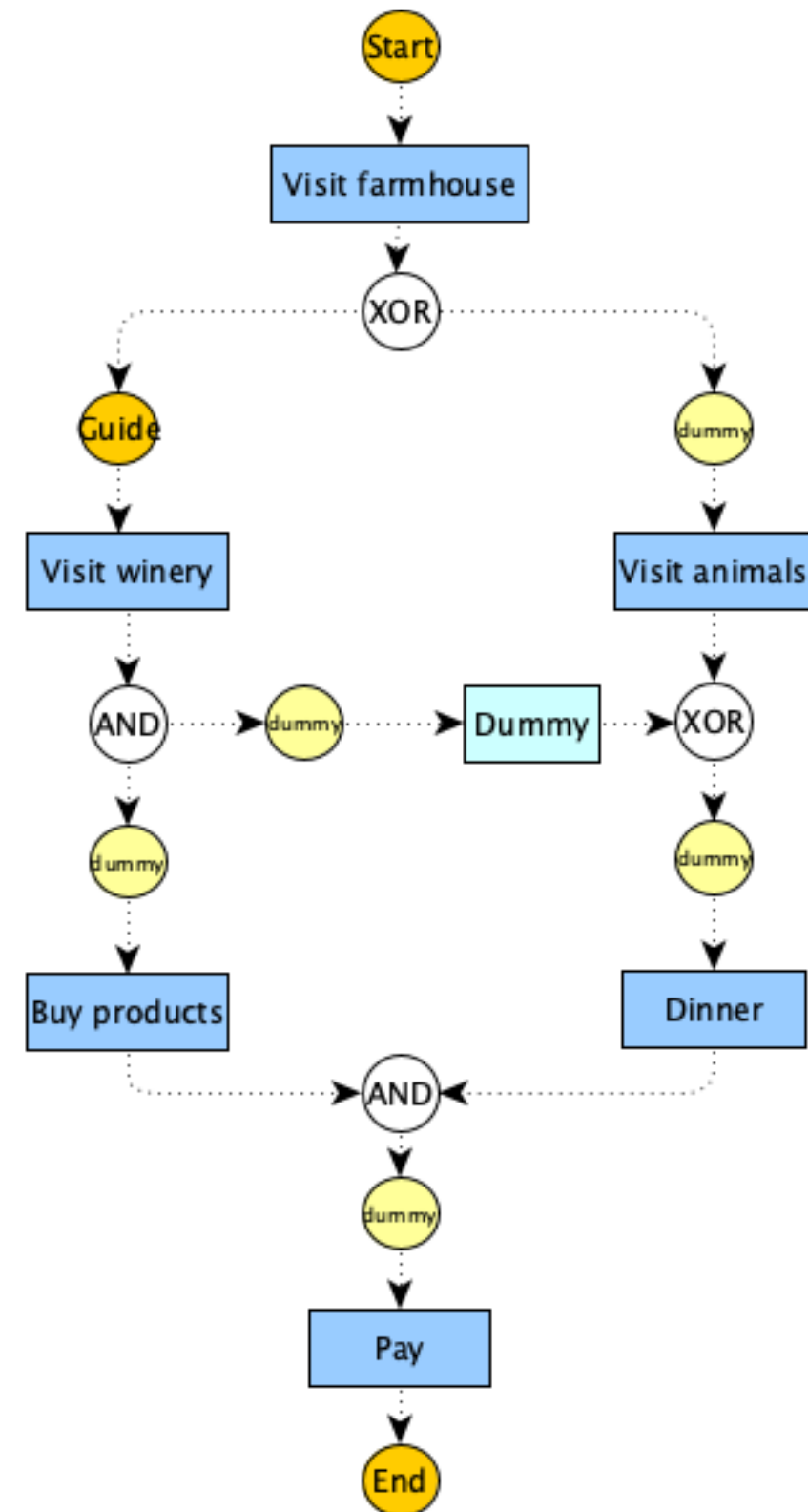
Step 0



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

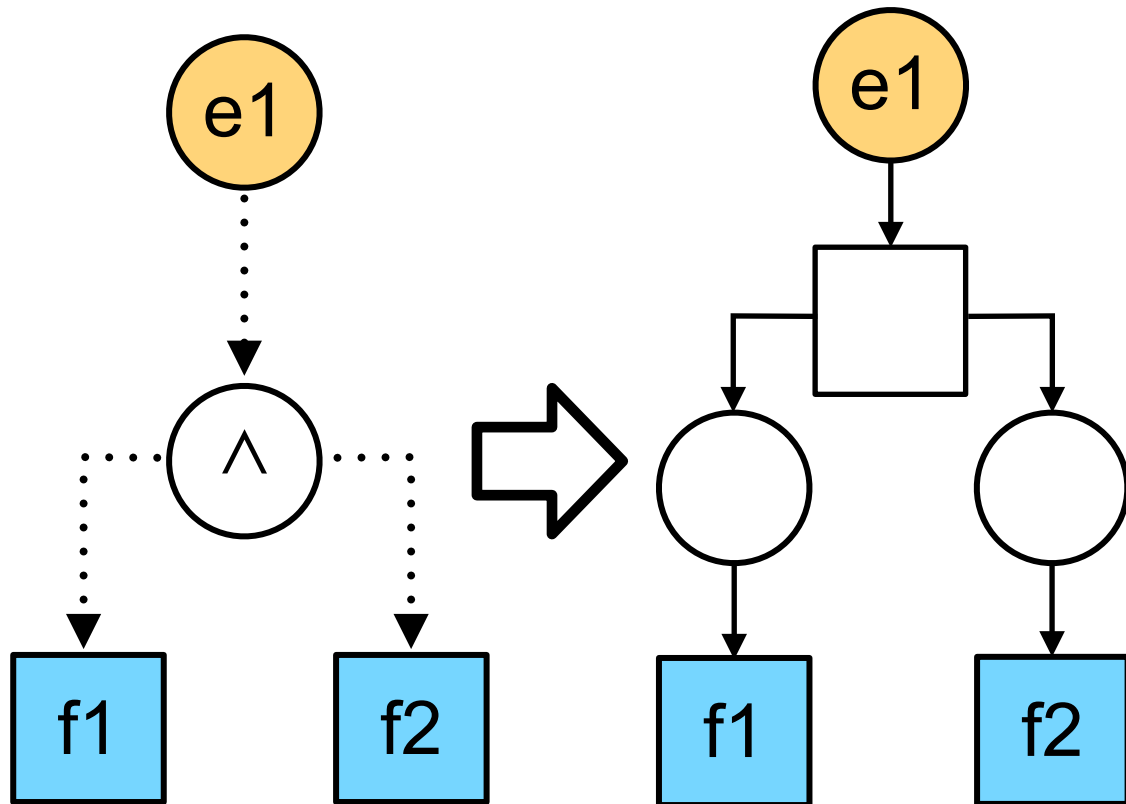
The translation of logical connectors
depends on the context:

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

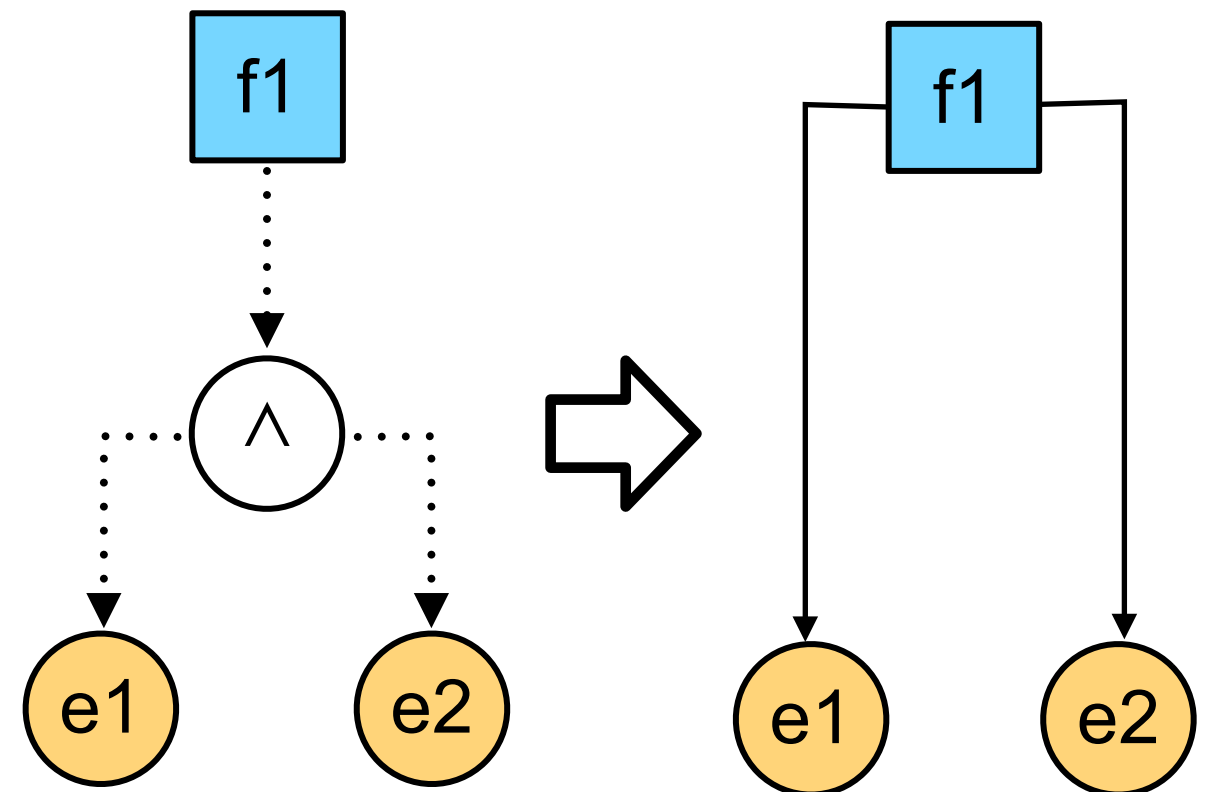
EPC



(event to functions)

net fragment

EPC



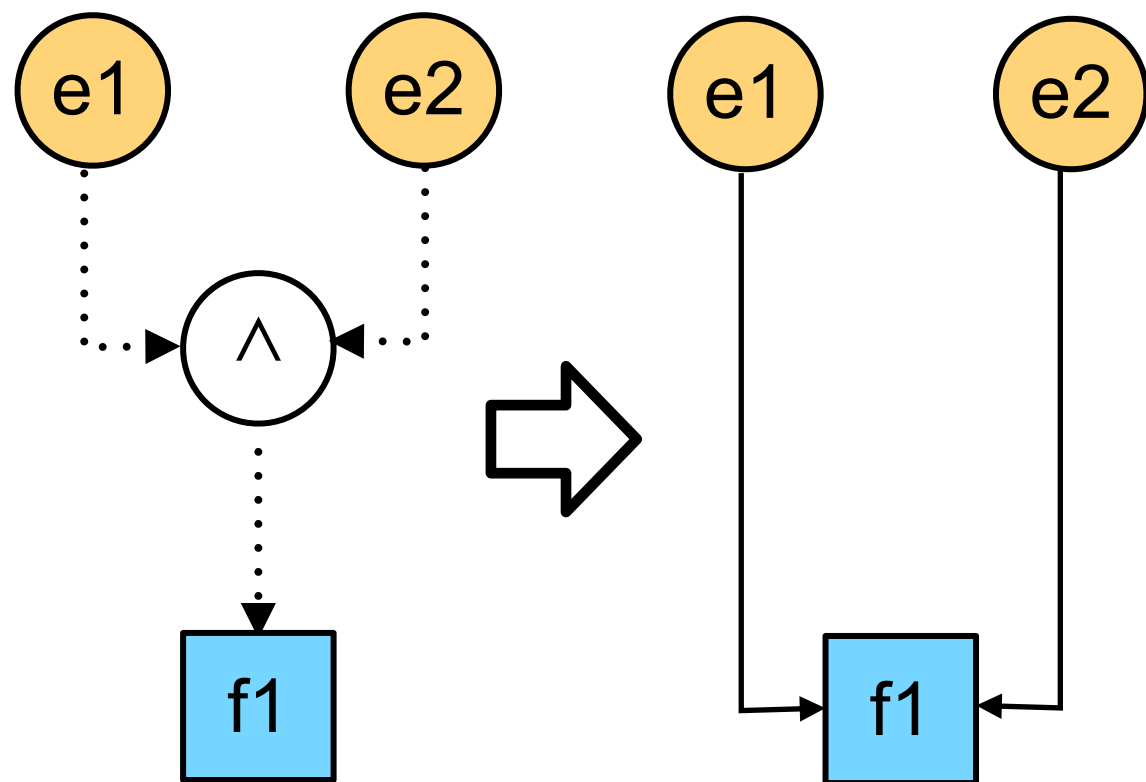
(functions to events)

net fragment

Step 1: AND join

EPC

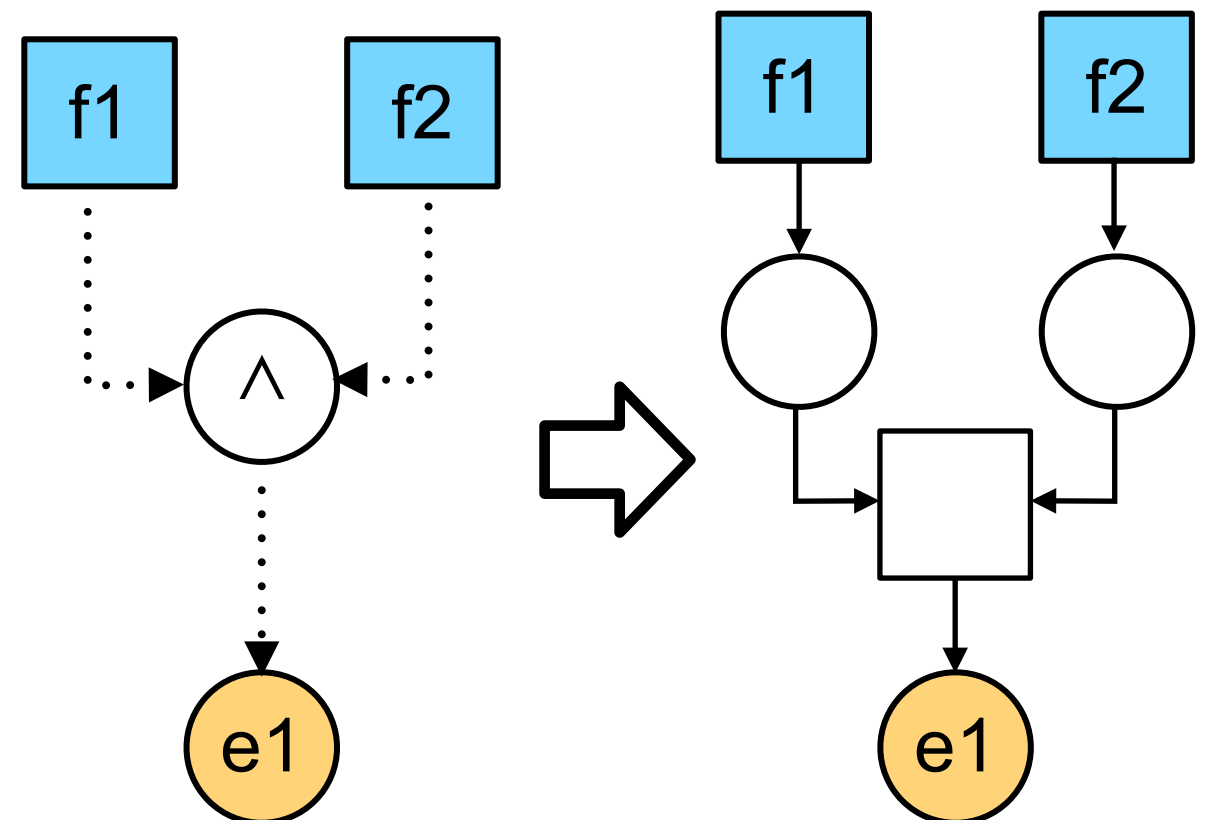
net fragment



(event to functions)

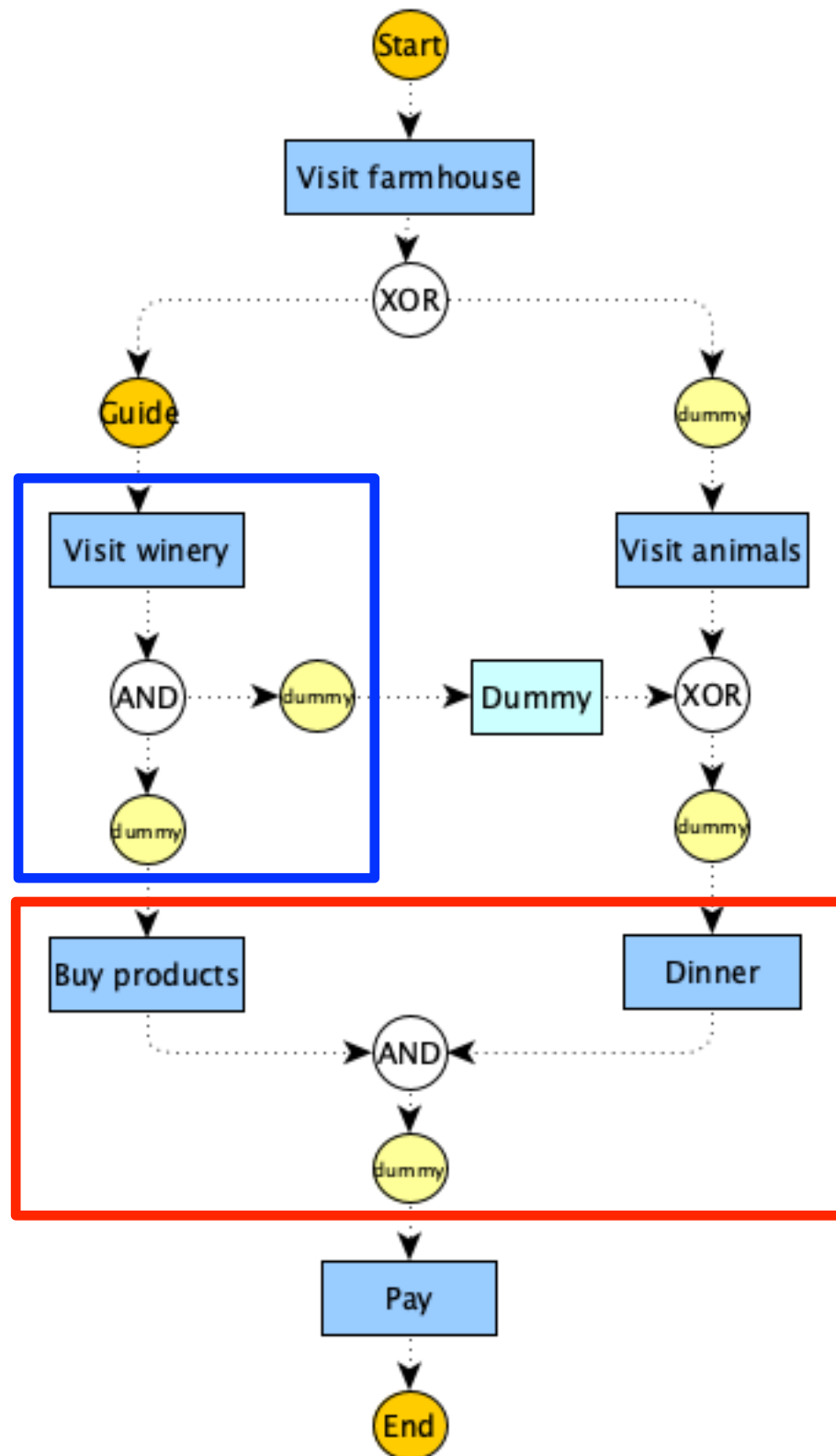
EPC

net fragment

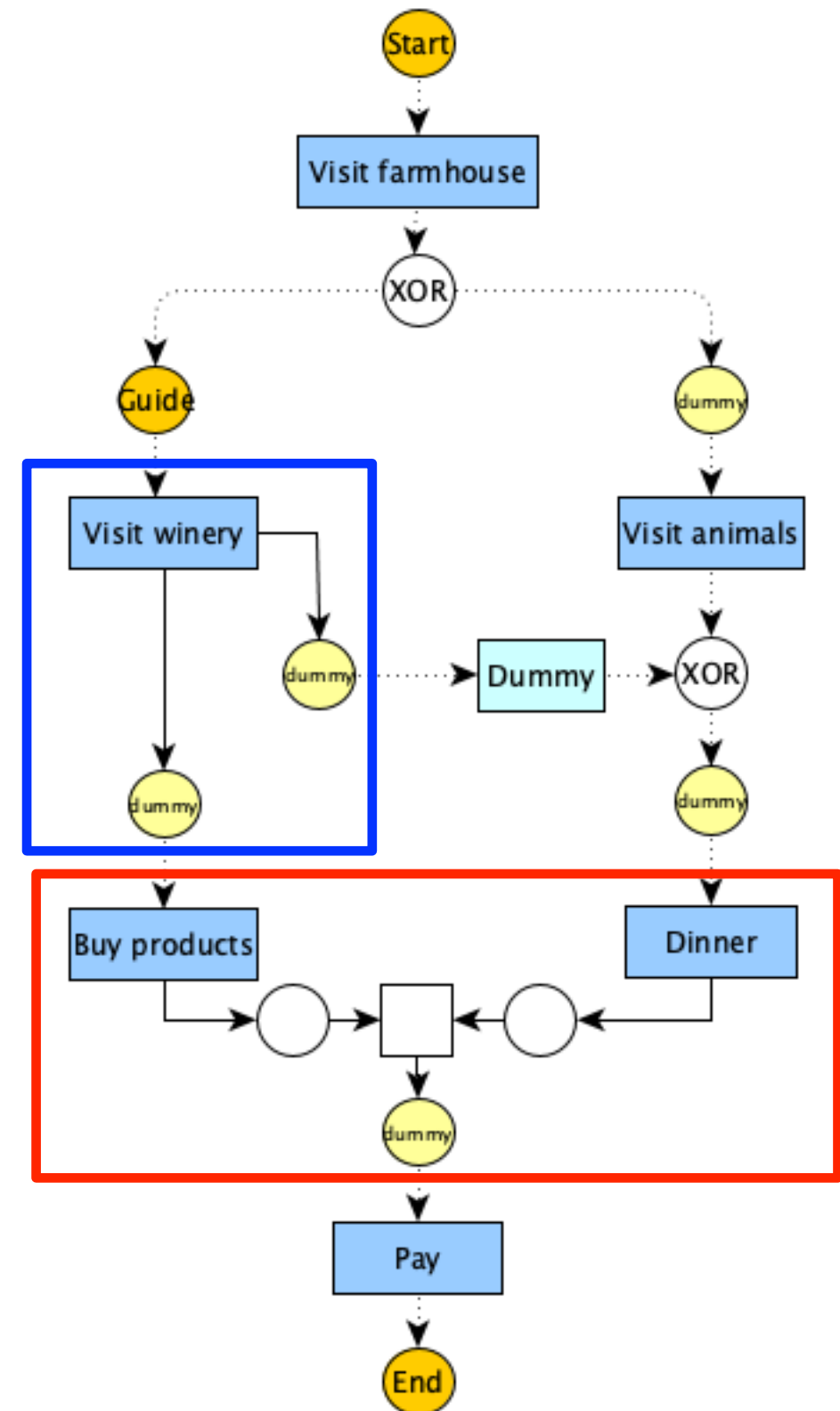


(functions to events)

Example

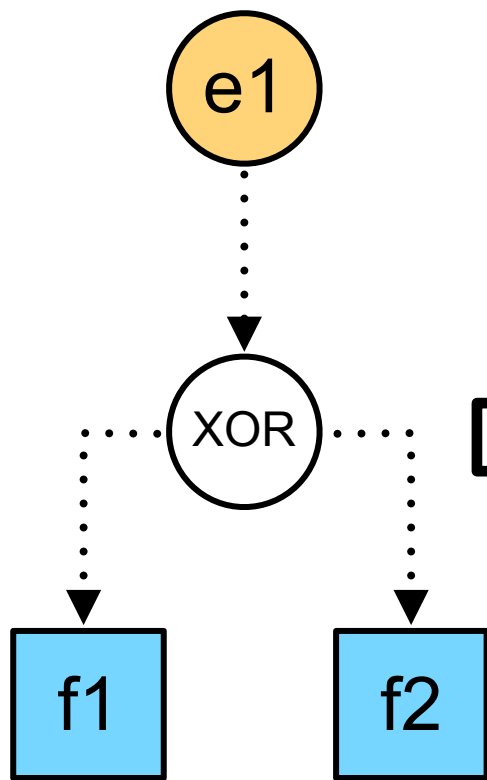


➡
Step 1
AND
connectors



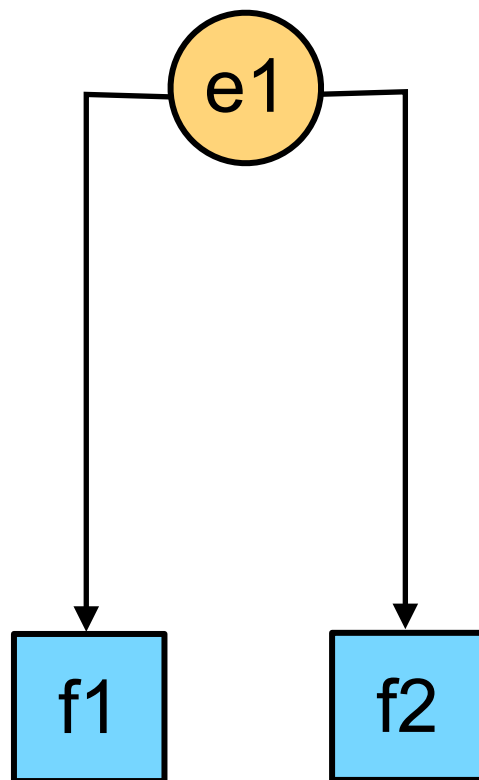
Step 1: XOR split

EPC

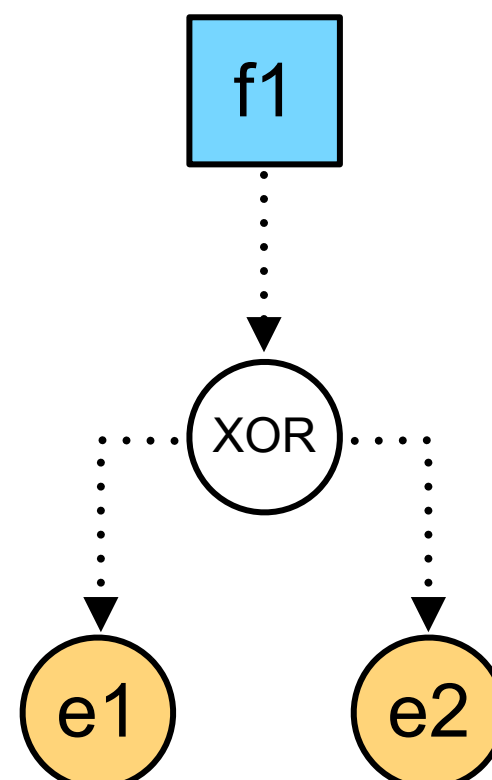


(event to functions)

net fragment

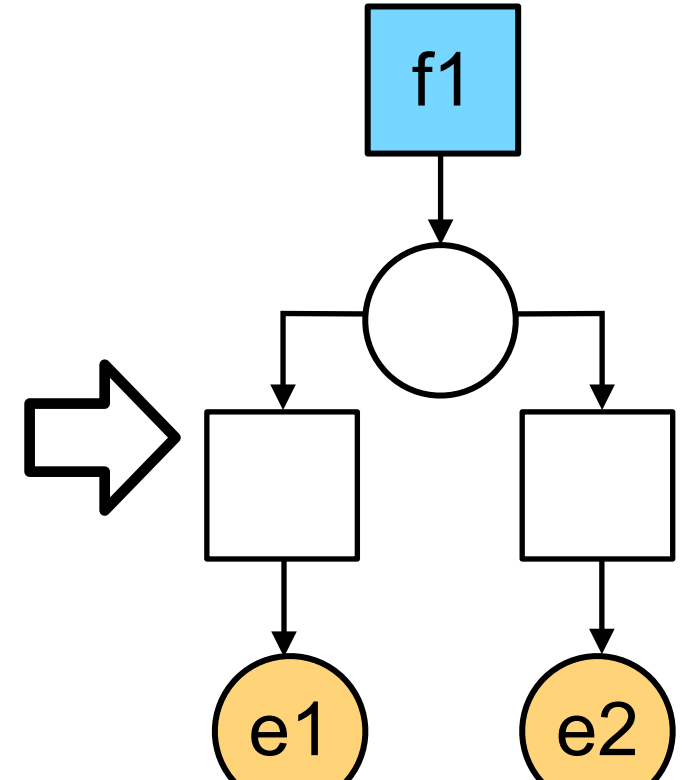


EPC



(functions to events)

net fragment



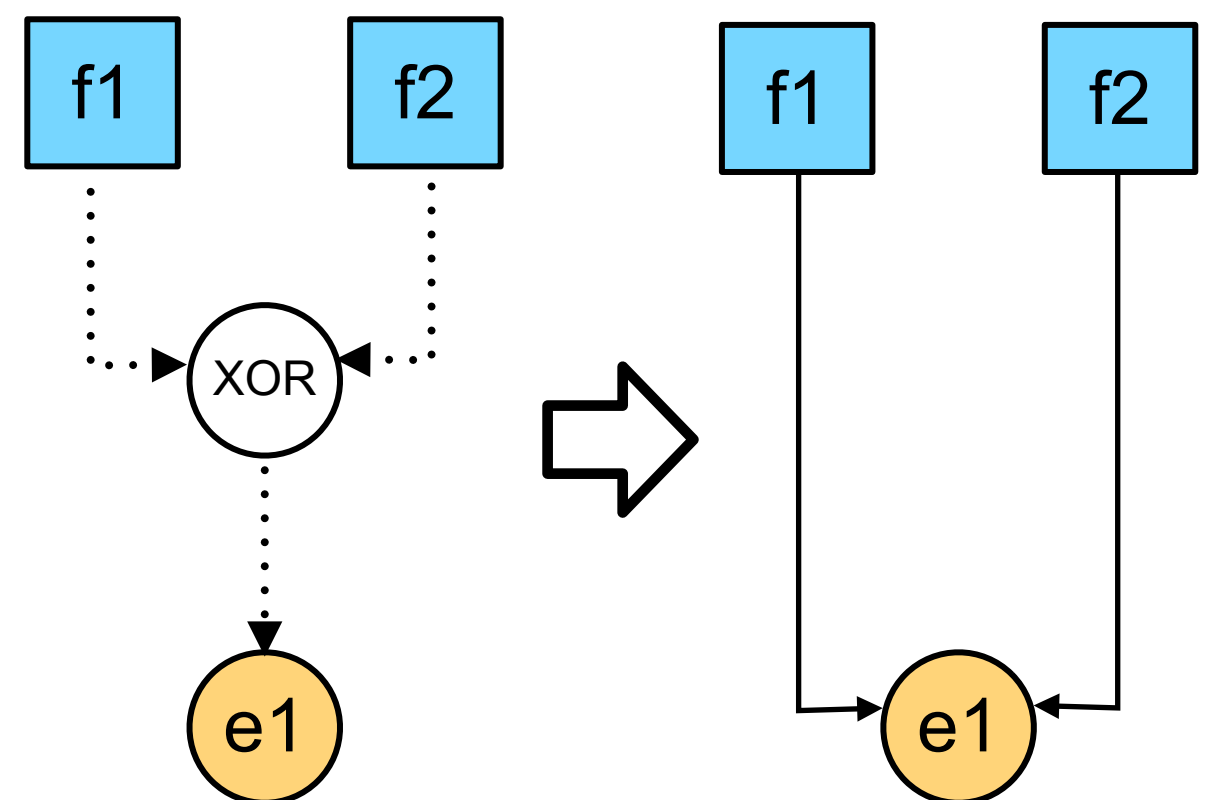
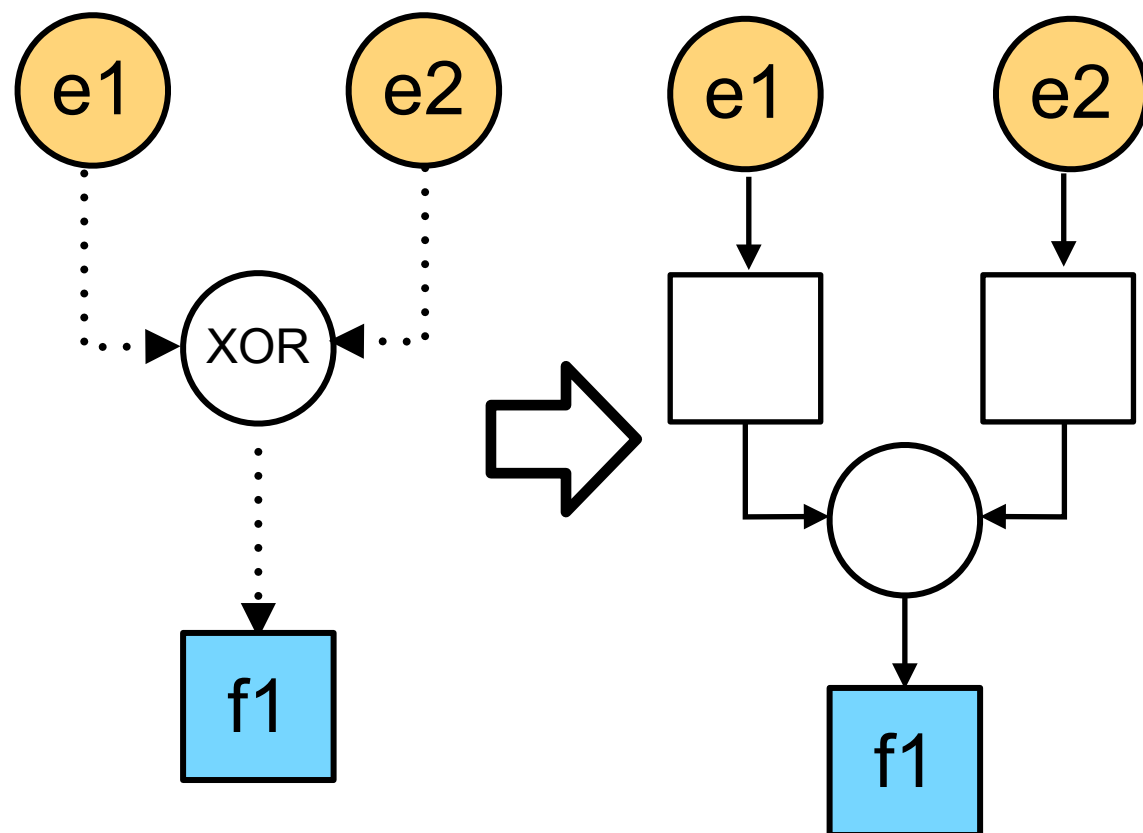
Step 1: XOR join

EPC

net fragment

EPC

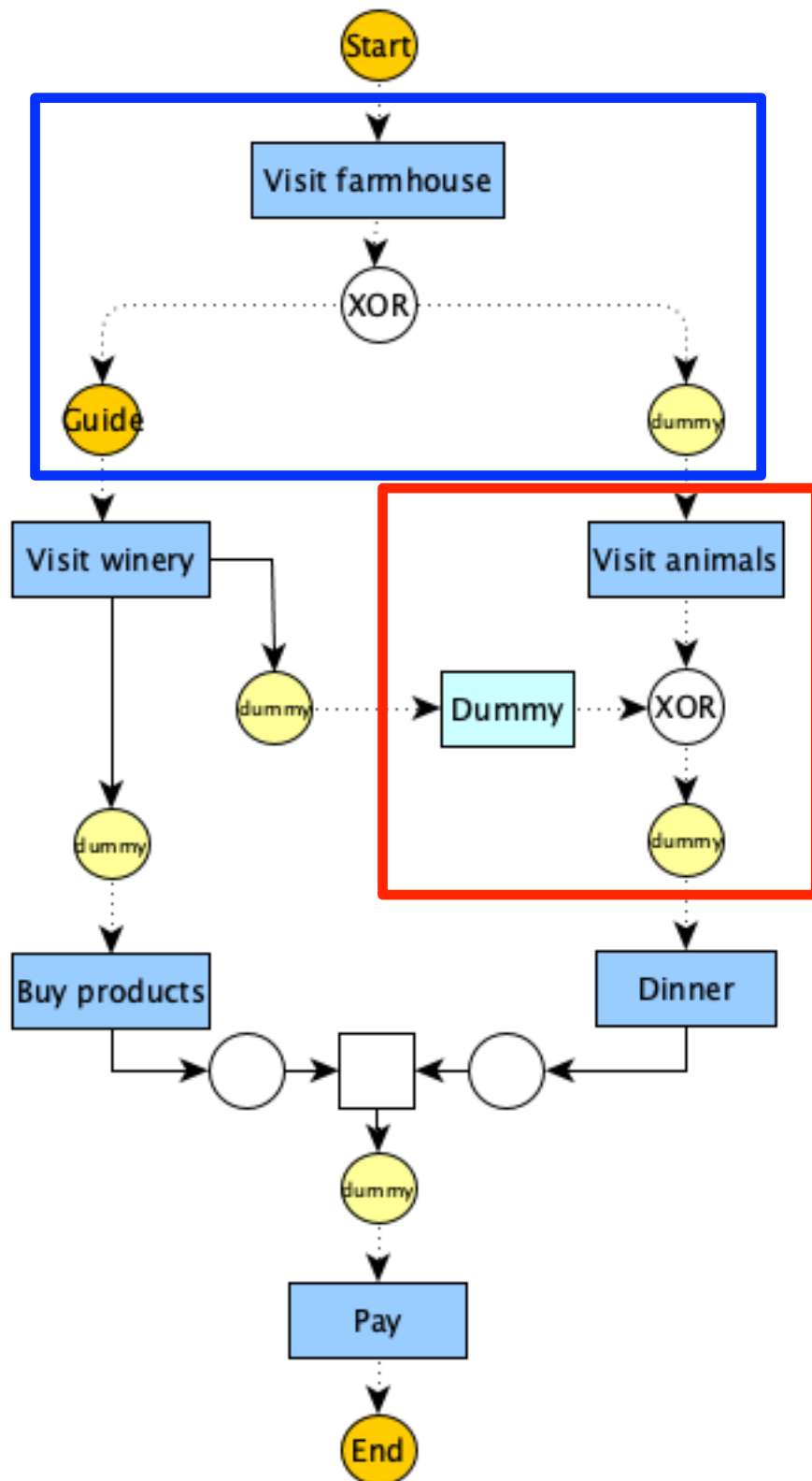
net fragment



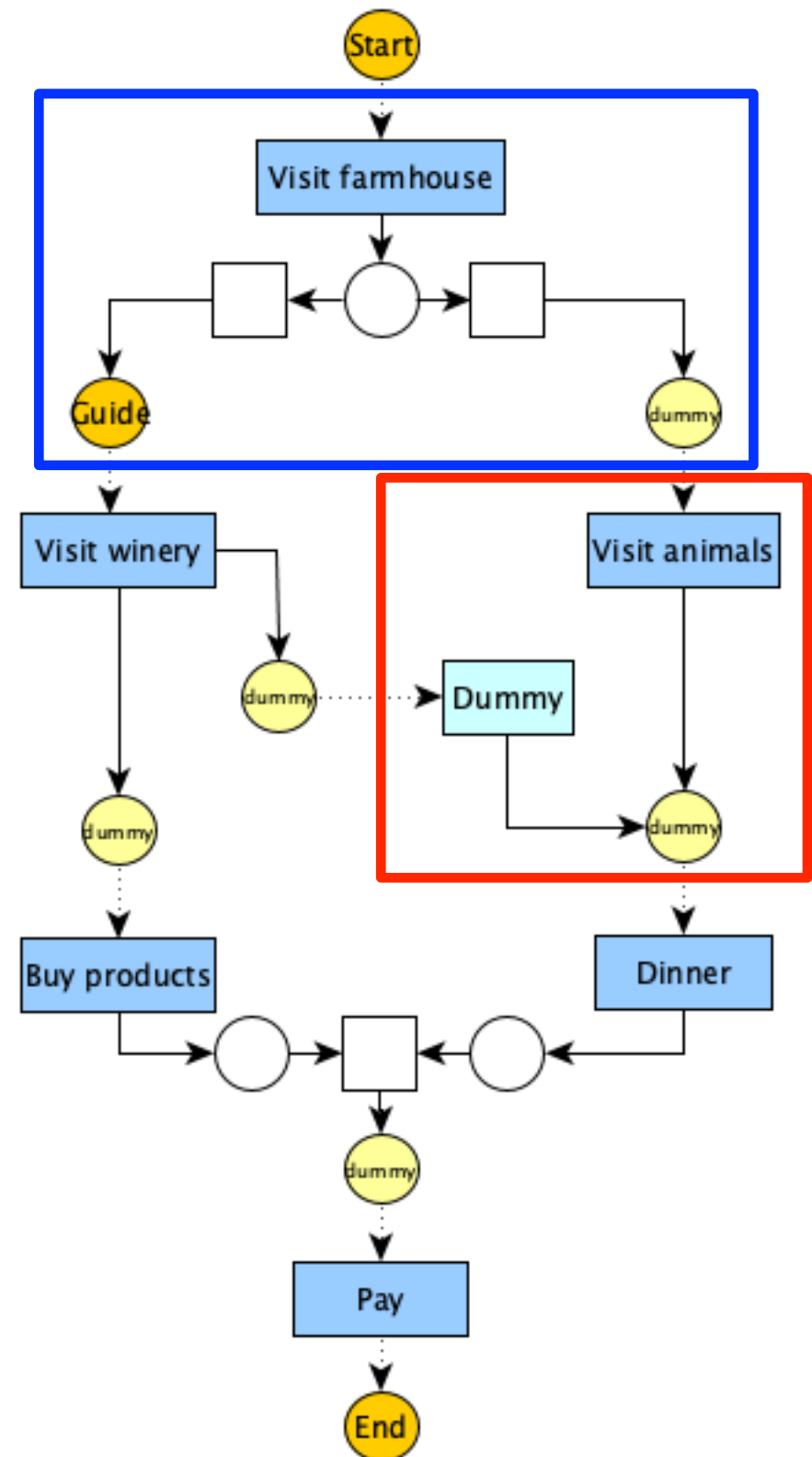
(event to functions)

(functions to events)

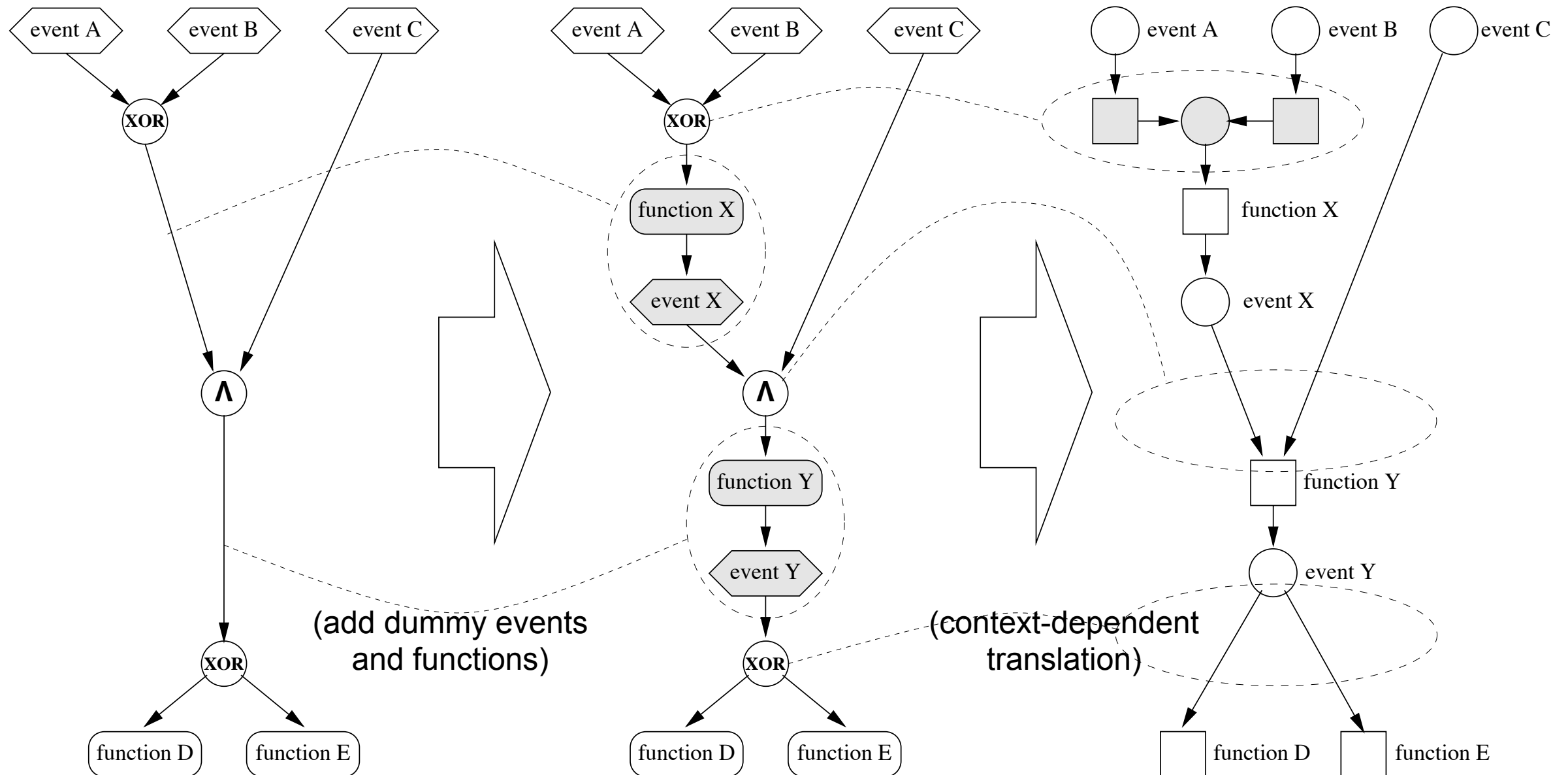
Example



➡
Step 1
XOR
connectors



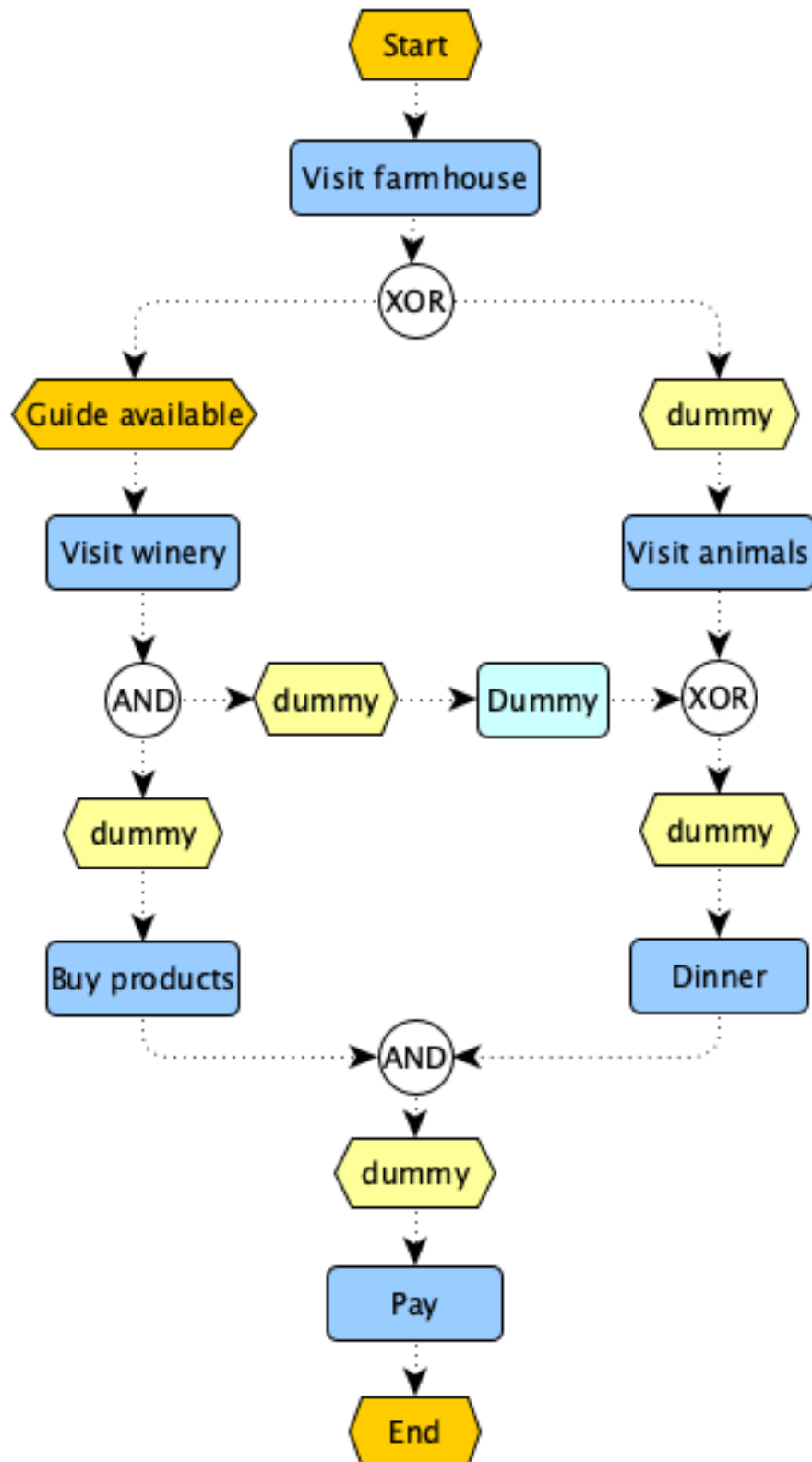
Overall strategy



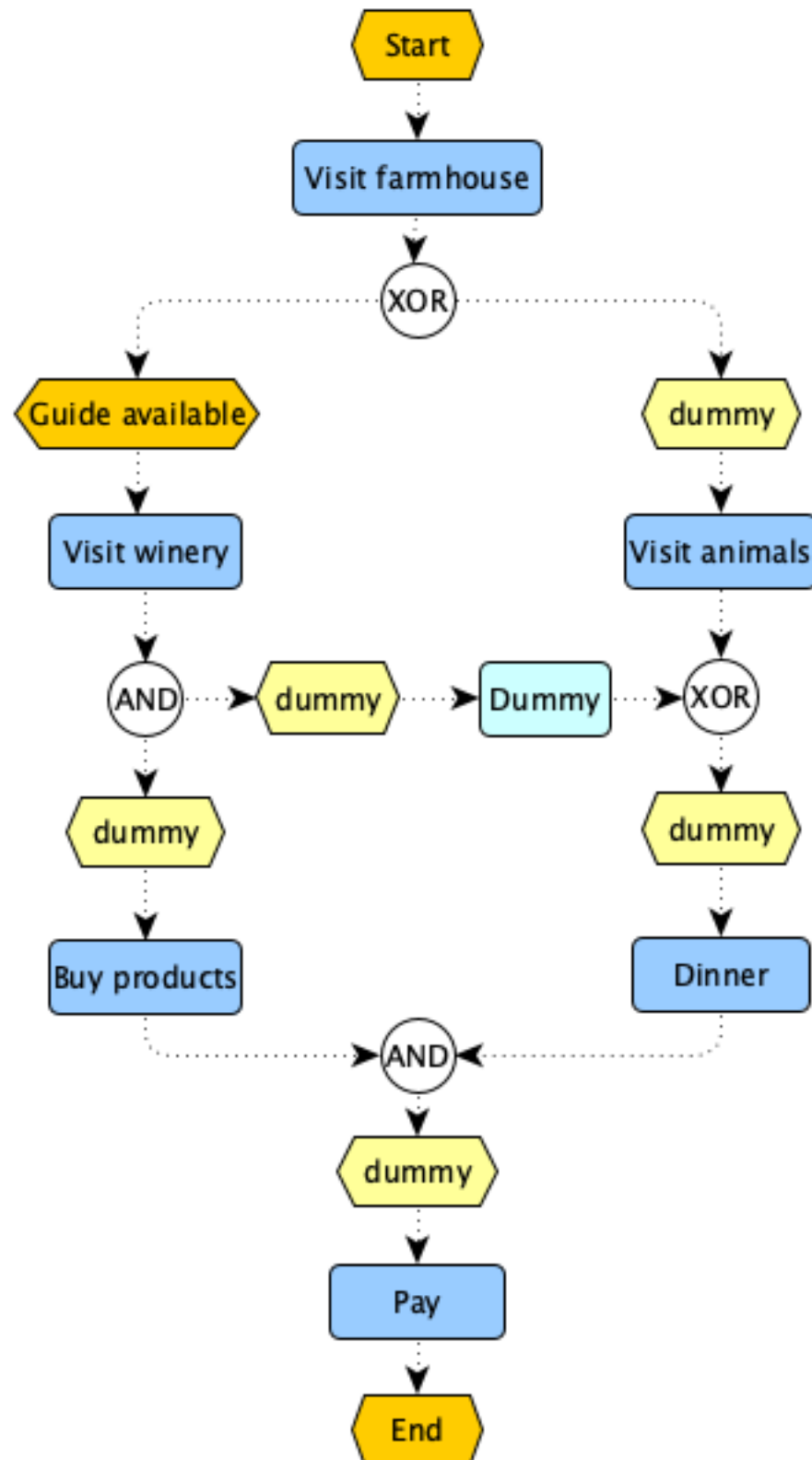
From any EPC we derive a free-choice net

Example

Sound?

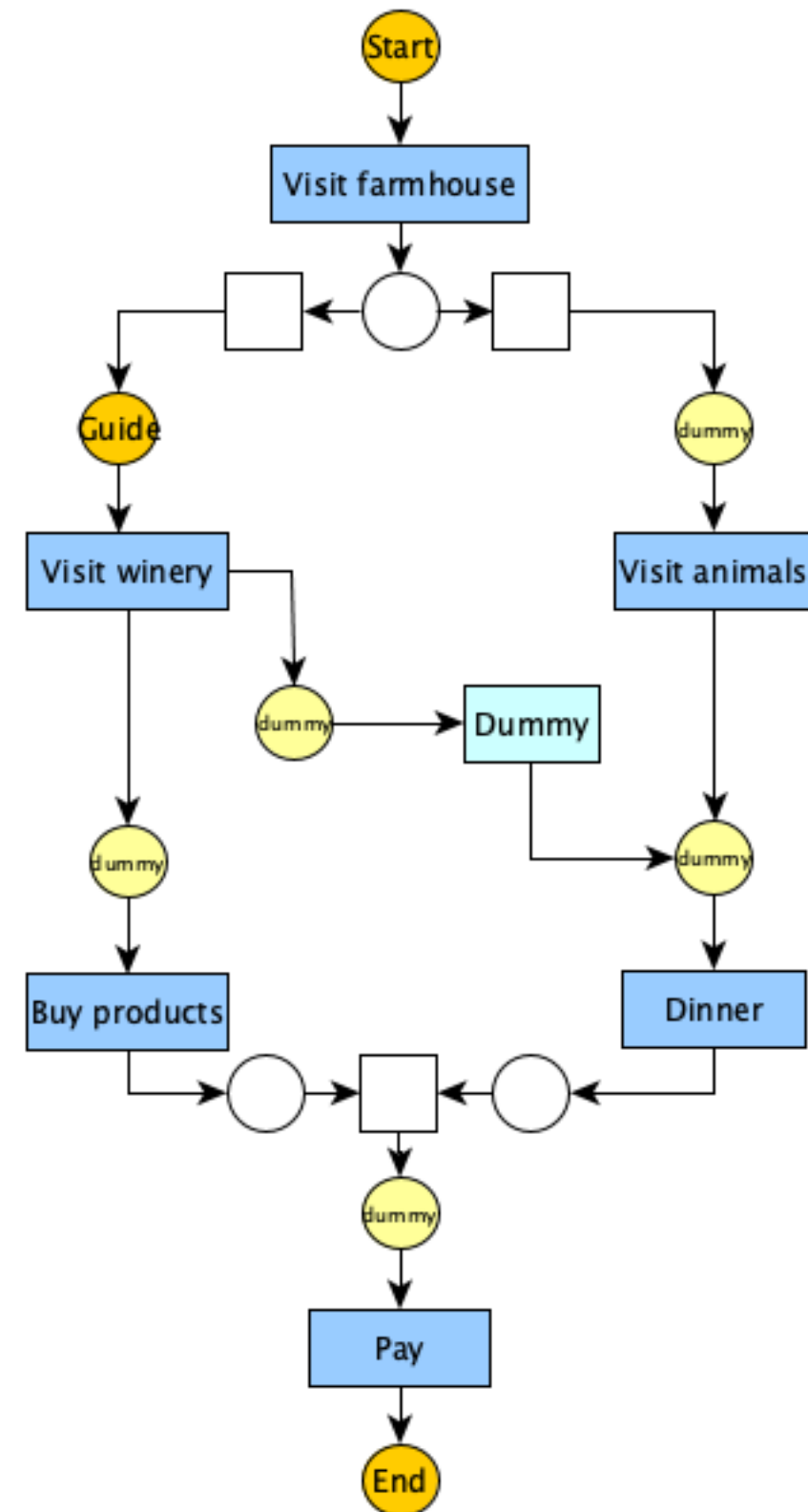


Example

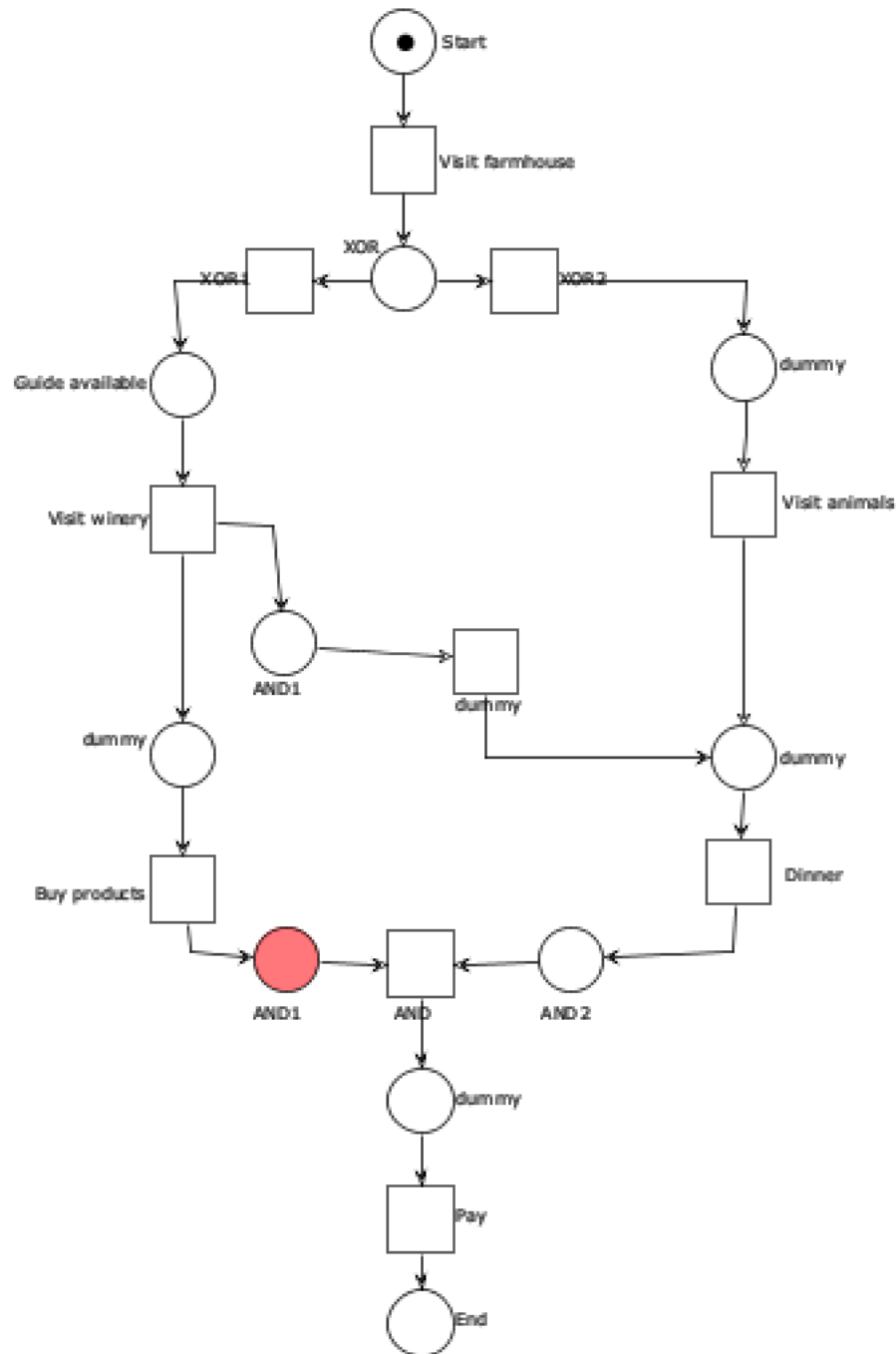


Sound?

➡
Steps
1+2(+3)



Example



Not sound!

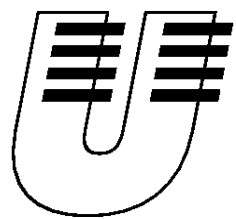
Semantical analysis

Wizard

Expert

- Qualitative analysis
 - Structural analysis
 - Net statistics
 - Wrongly used operators: 0
 - Free-choice violations: 0
 - S-Components
 - S-Components: 1
 - Places not covered by S-Component
 - AND1
 - 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

PETER RITTGEN

MODIFIED EPCs AND THEIR FORMAL SEMANTICS

Decorated EPC

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

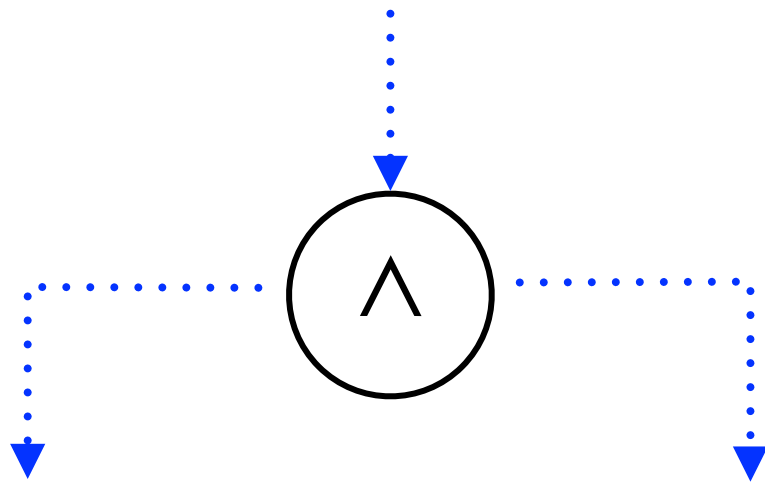
We require:

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

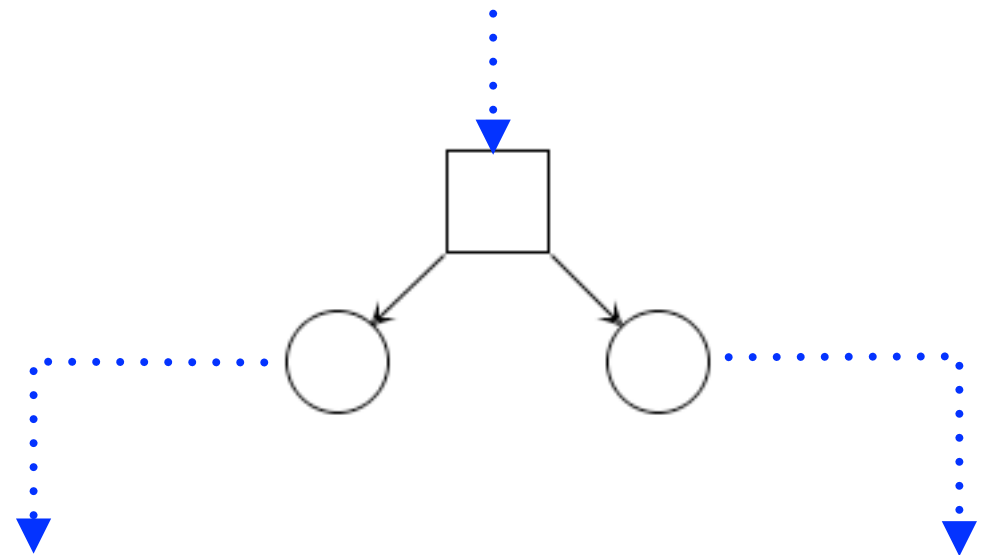
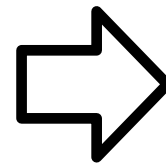
OR-joins are decorated with a policy
(avoid OR join ambiguous behaviour)

Step 1: AND split

EPC element

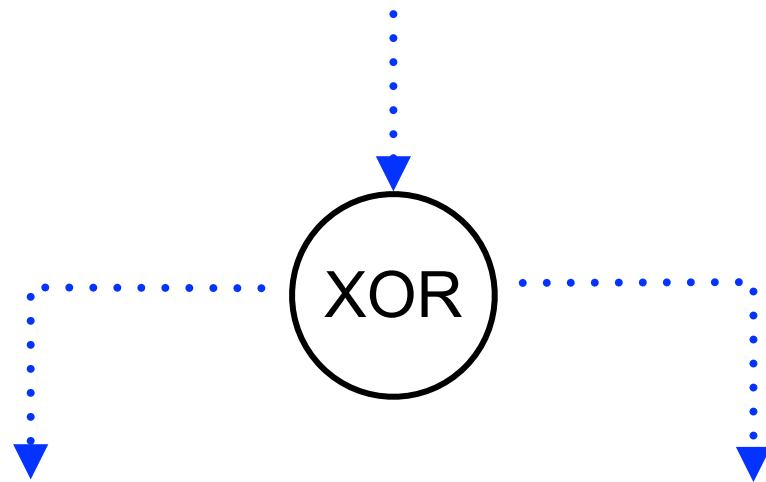


net fragment

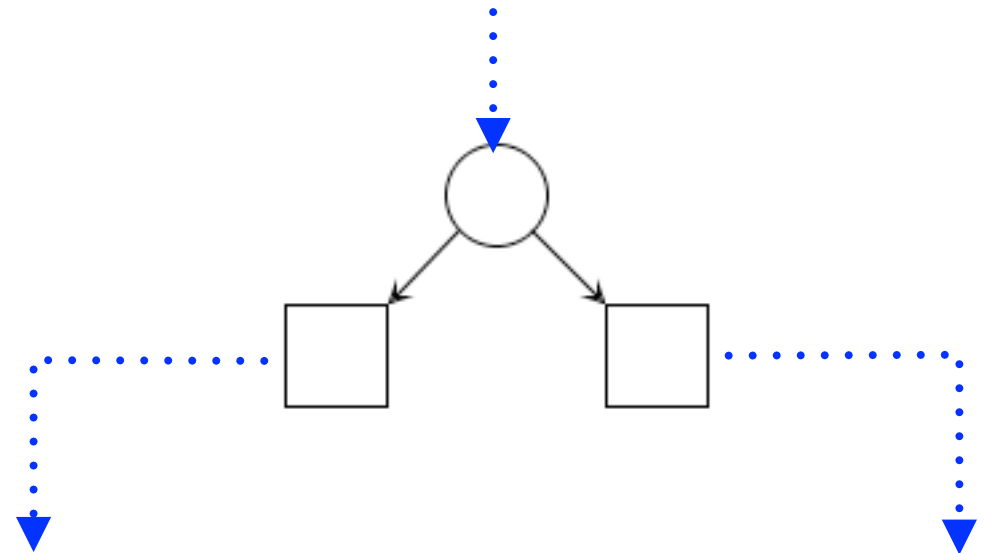
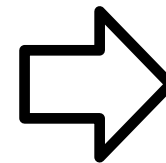


Step 1: XOR split

EPC element

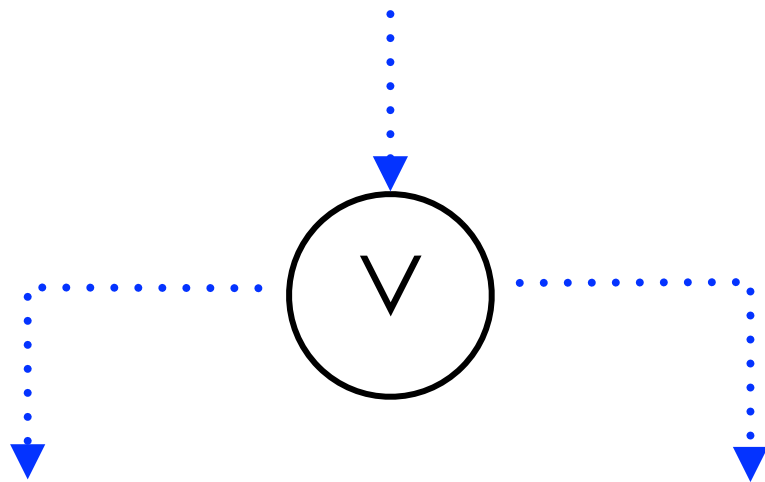


net fragment

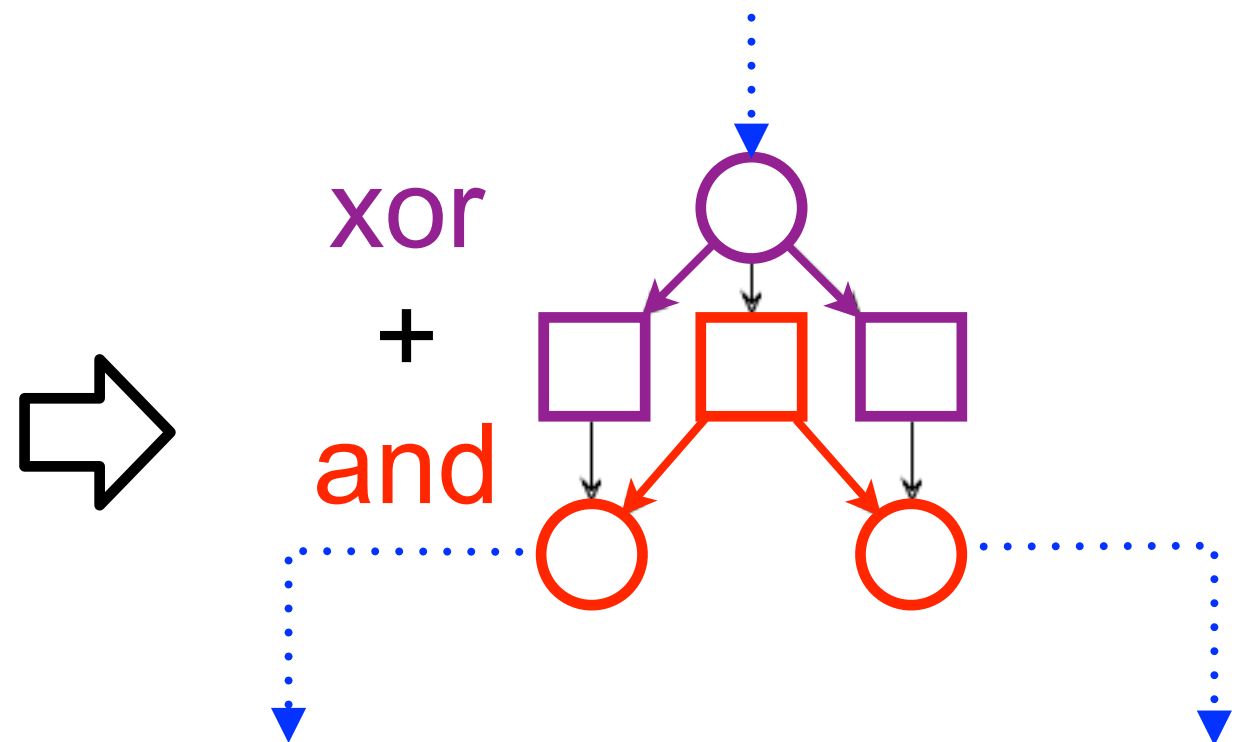


Step 1: OR split

EPC element

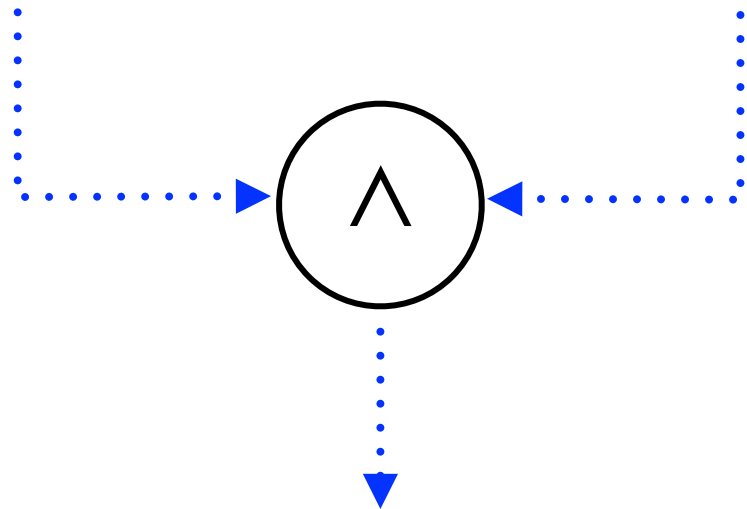


net fragment

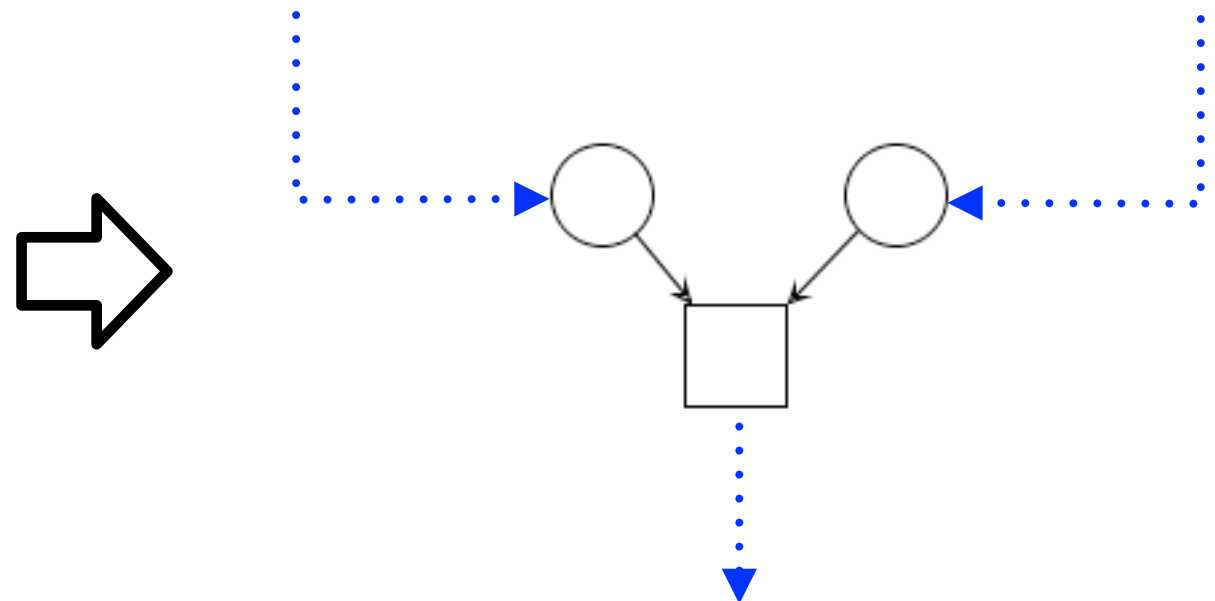


Step 1: AND join

EPC element

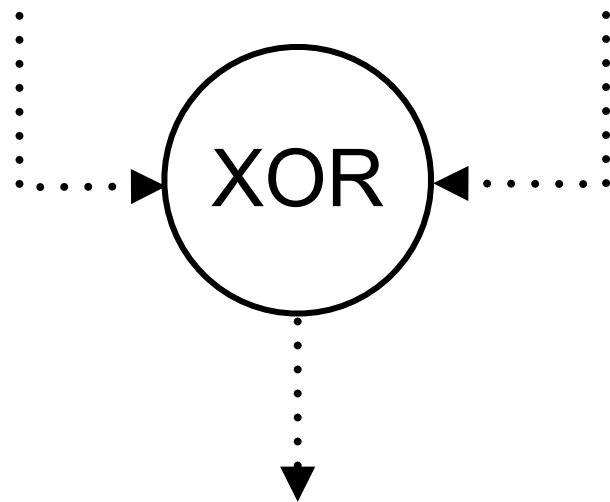


net fragment



XOR join: intended meaning

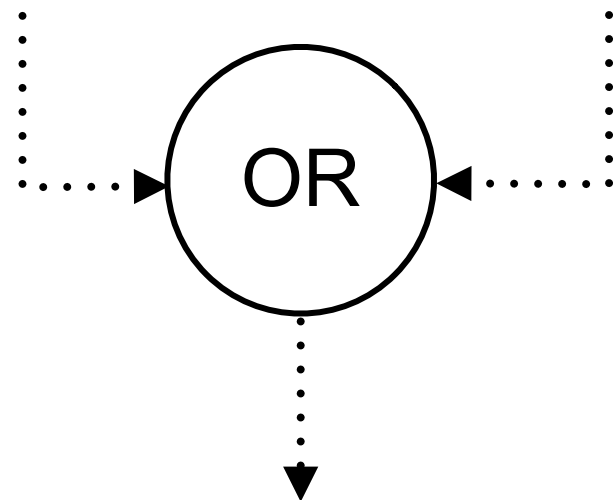
if both inputs arrive,
it should block the flow



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

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

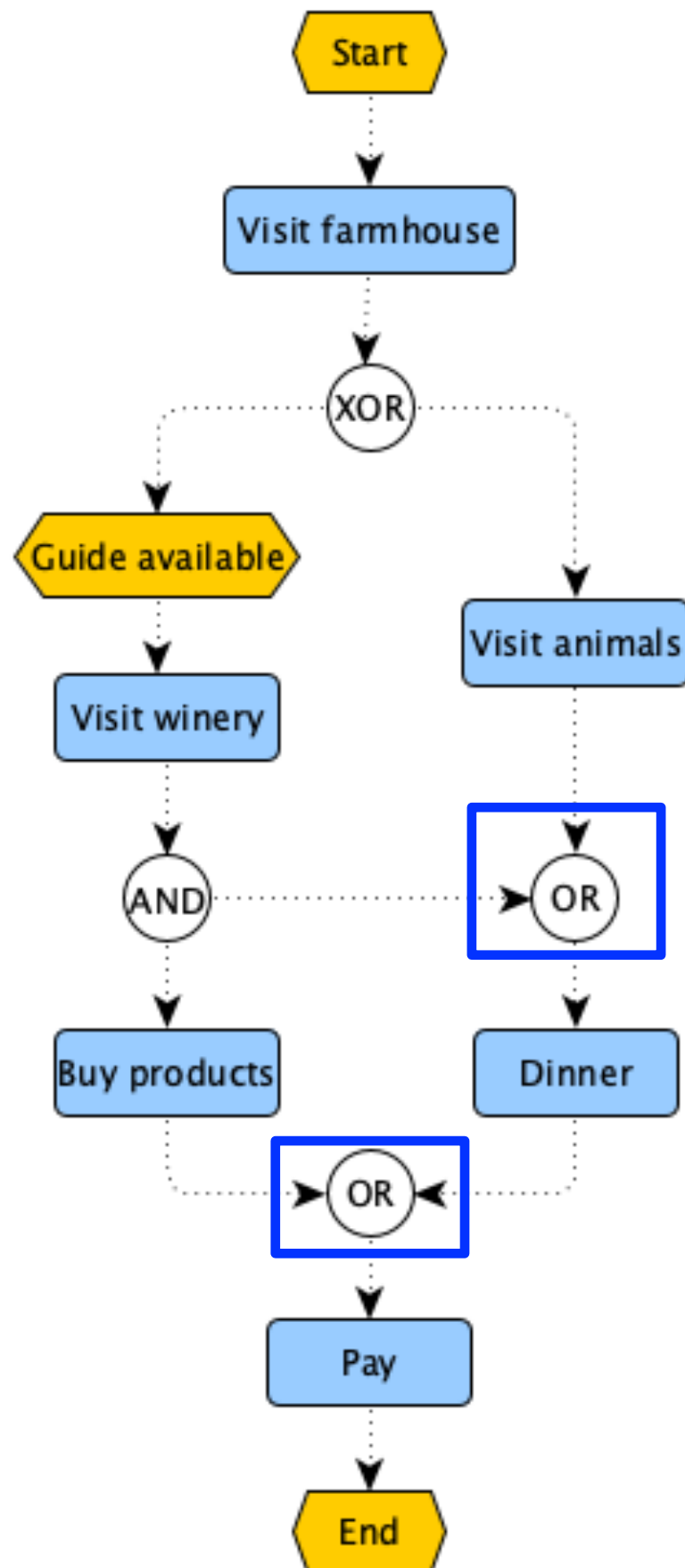
Otherwise, also other policies can be chosen:

every-time: trigger the outgoing path on each input

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

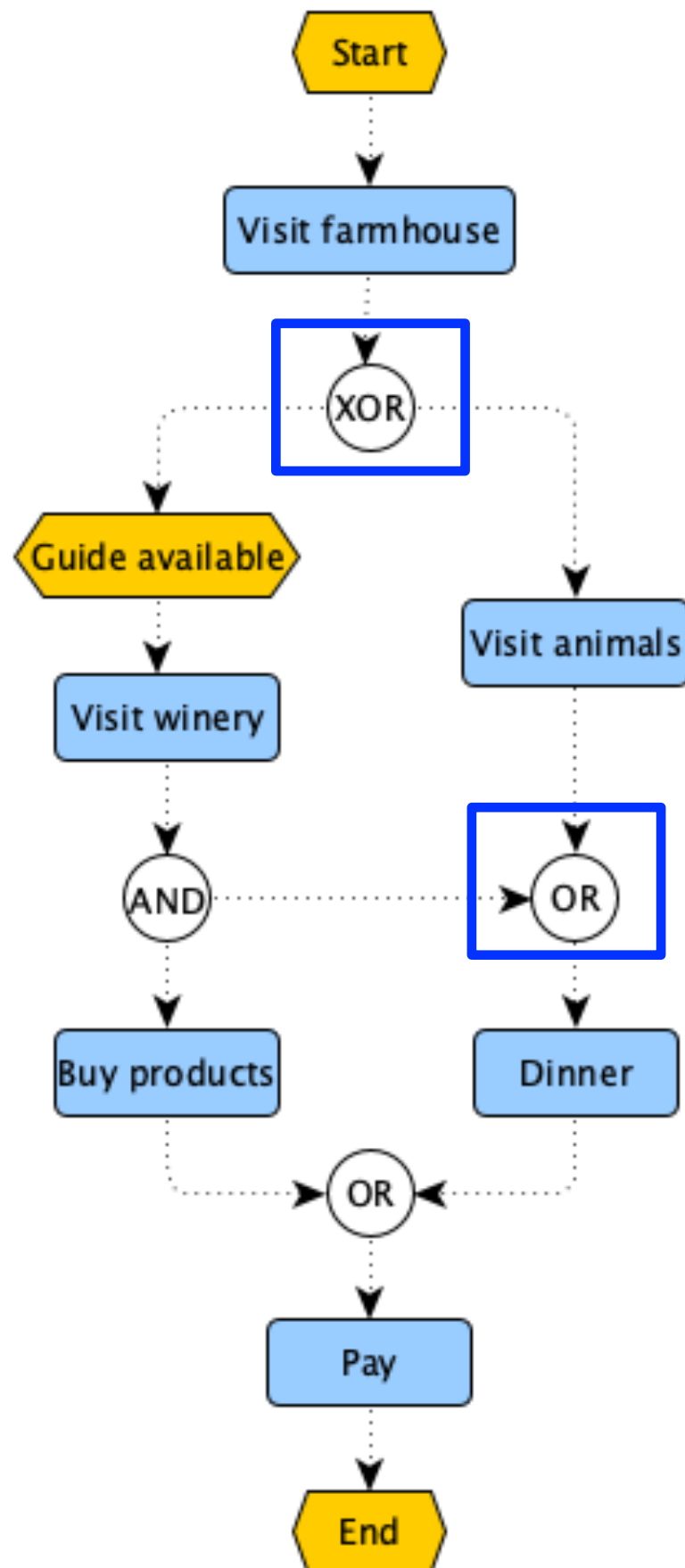
Assumption: every OR join is tagged with a policy
(some suggested to have different trapezoid symbols)

Example



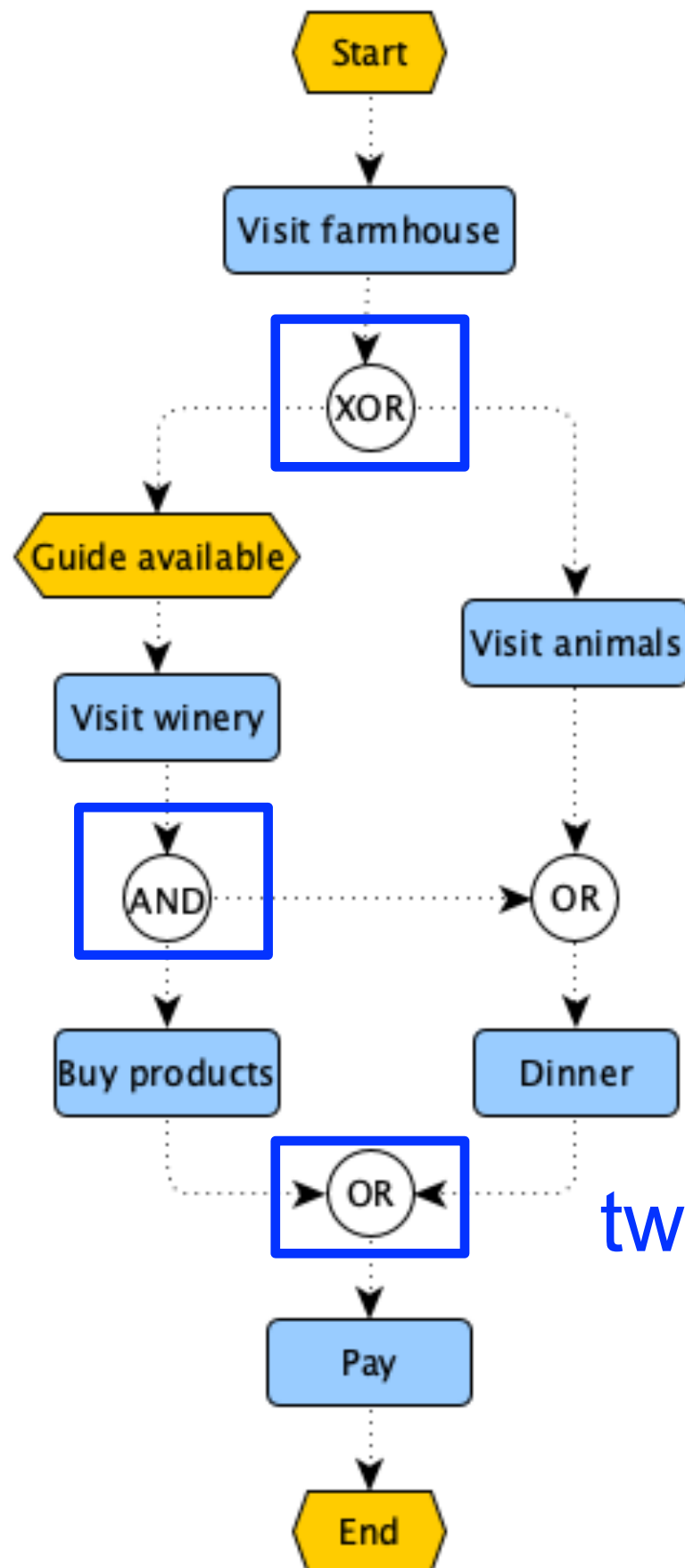
two OR joins
but no OR split

Example



only one
candidate split

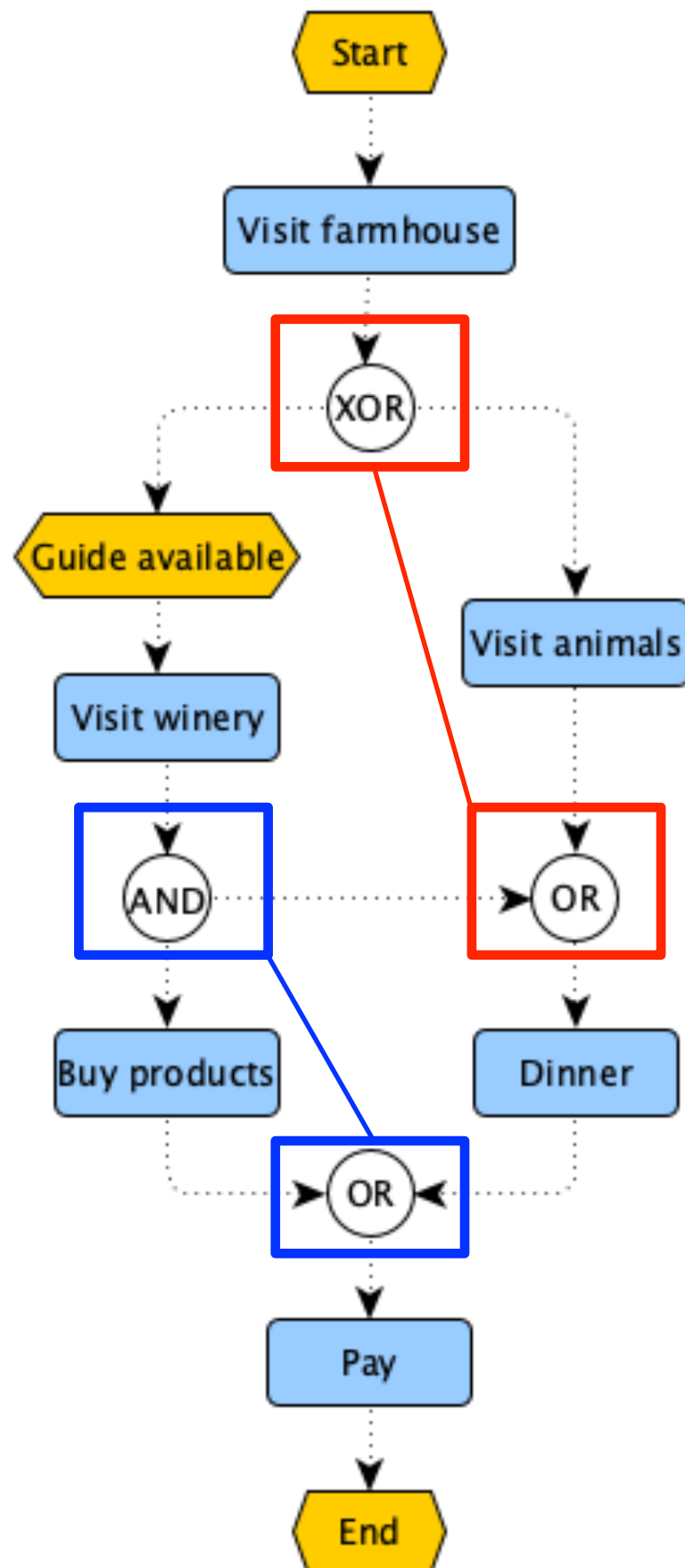
Example



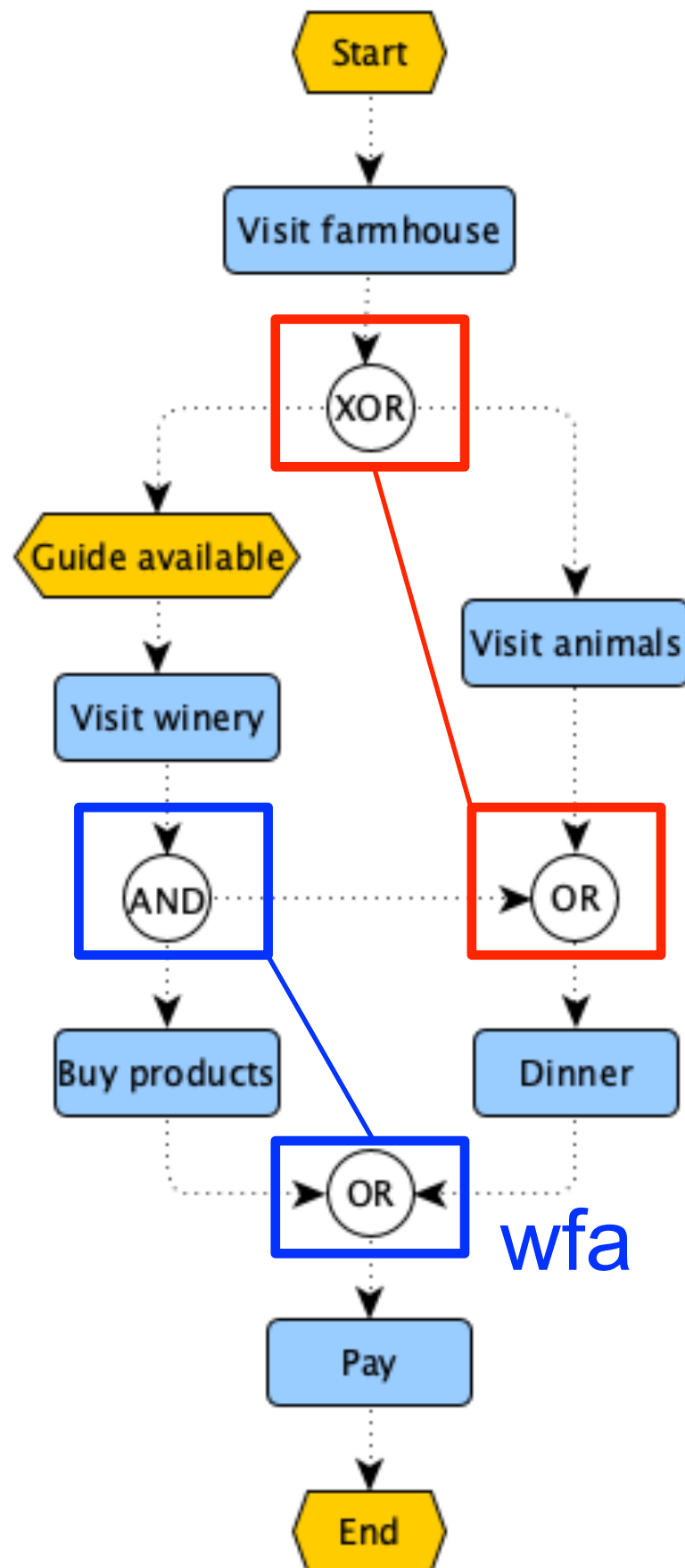
two candidate
splits

Example

assign corresponding splits



Example



fc

assign policies

wfa

Assumption

...

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

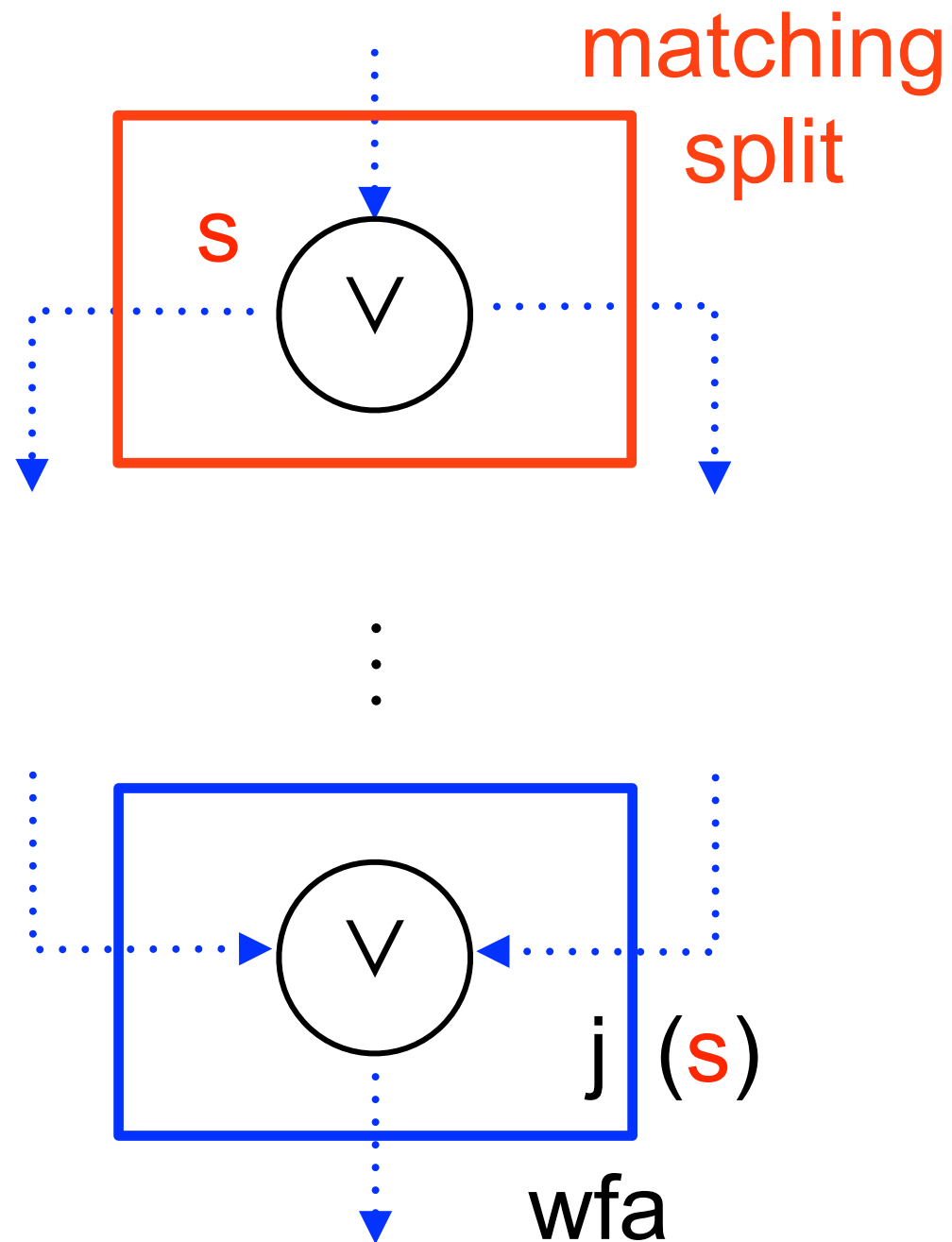
If an OR join has non-matching corresponding split
it is decorated with a policy (wfa, fc, et)

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

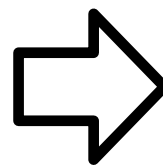
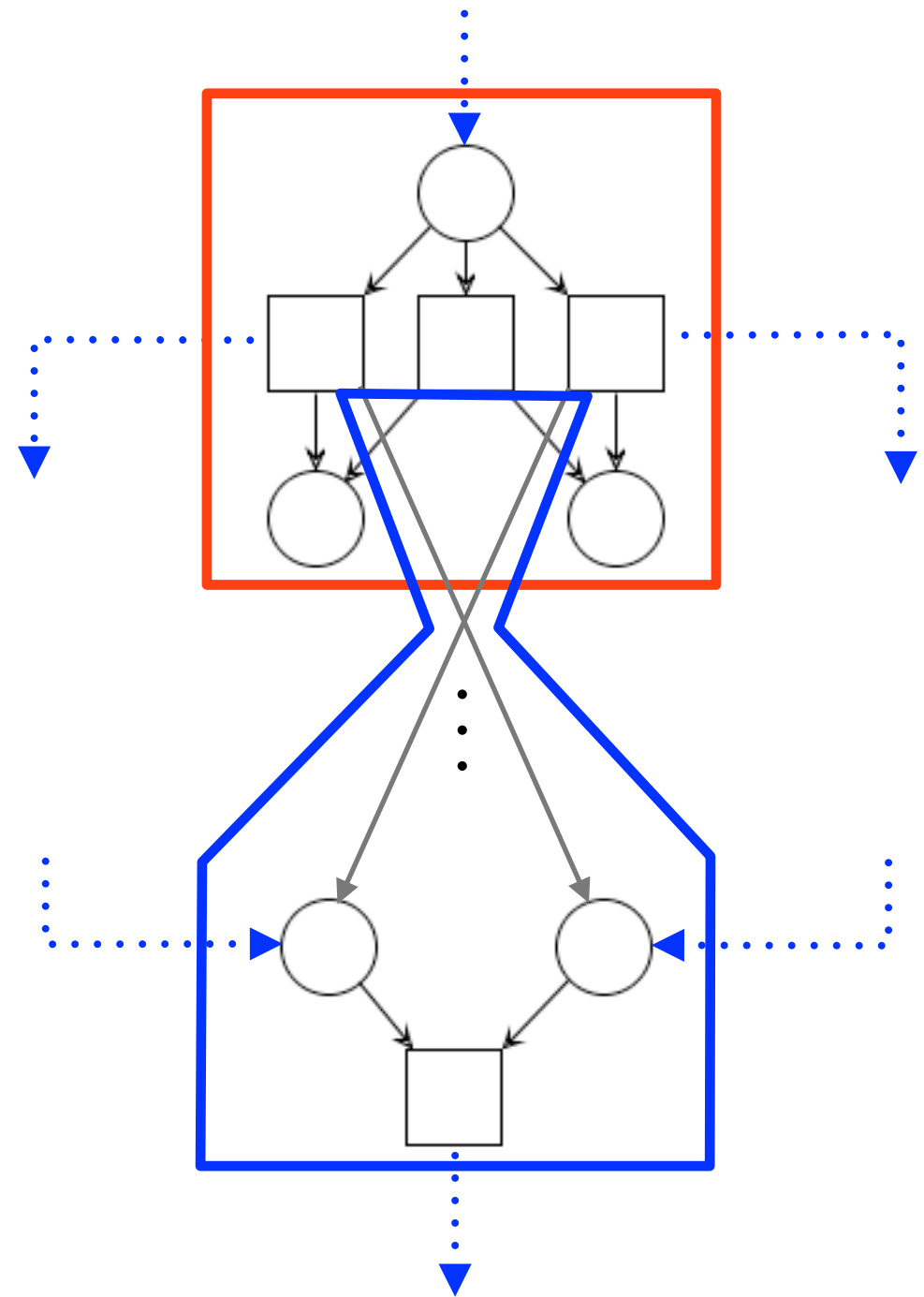
...

Step 1: OR join (wfa)

EPC element

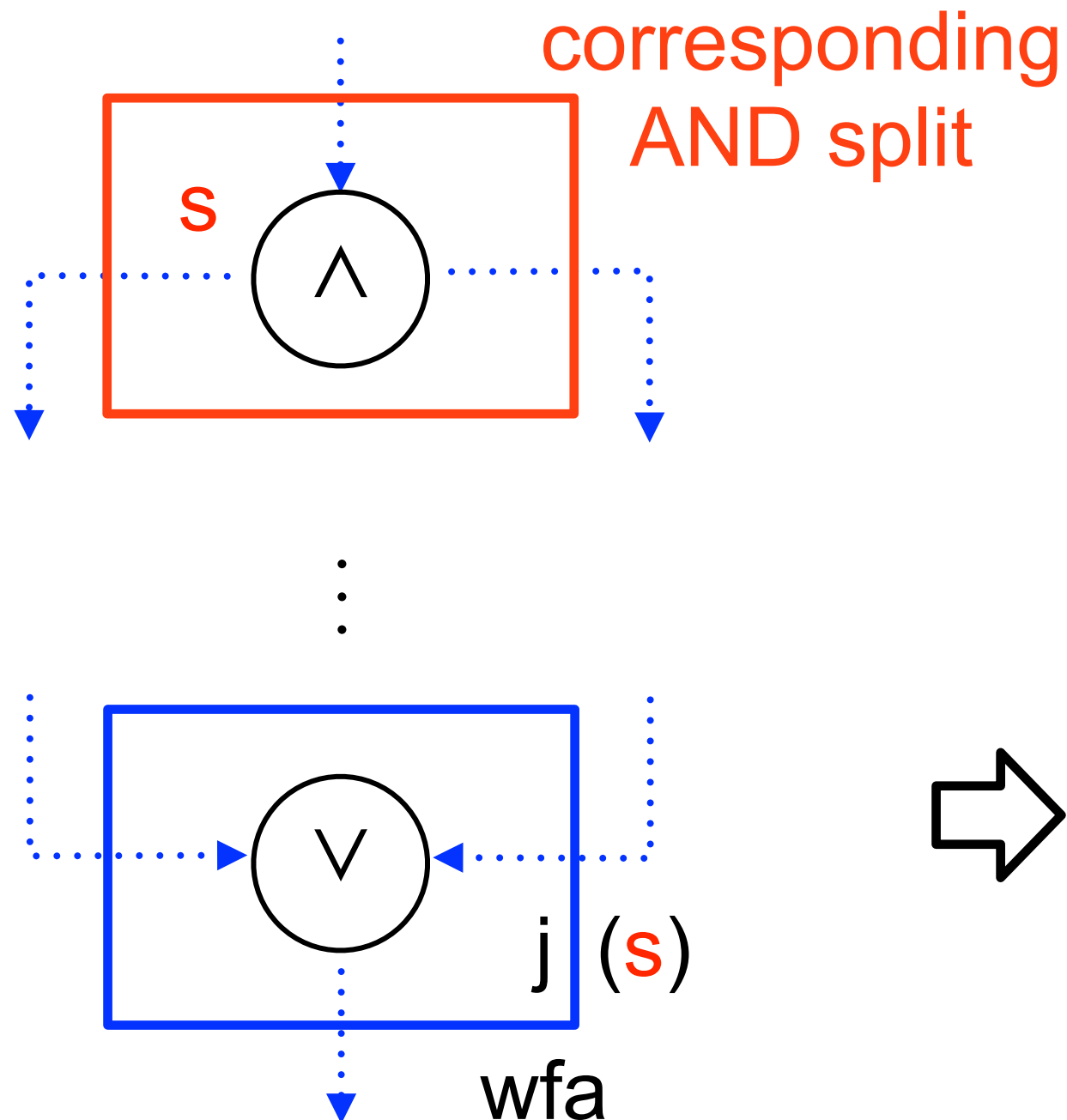


net fragment

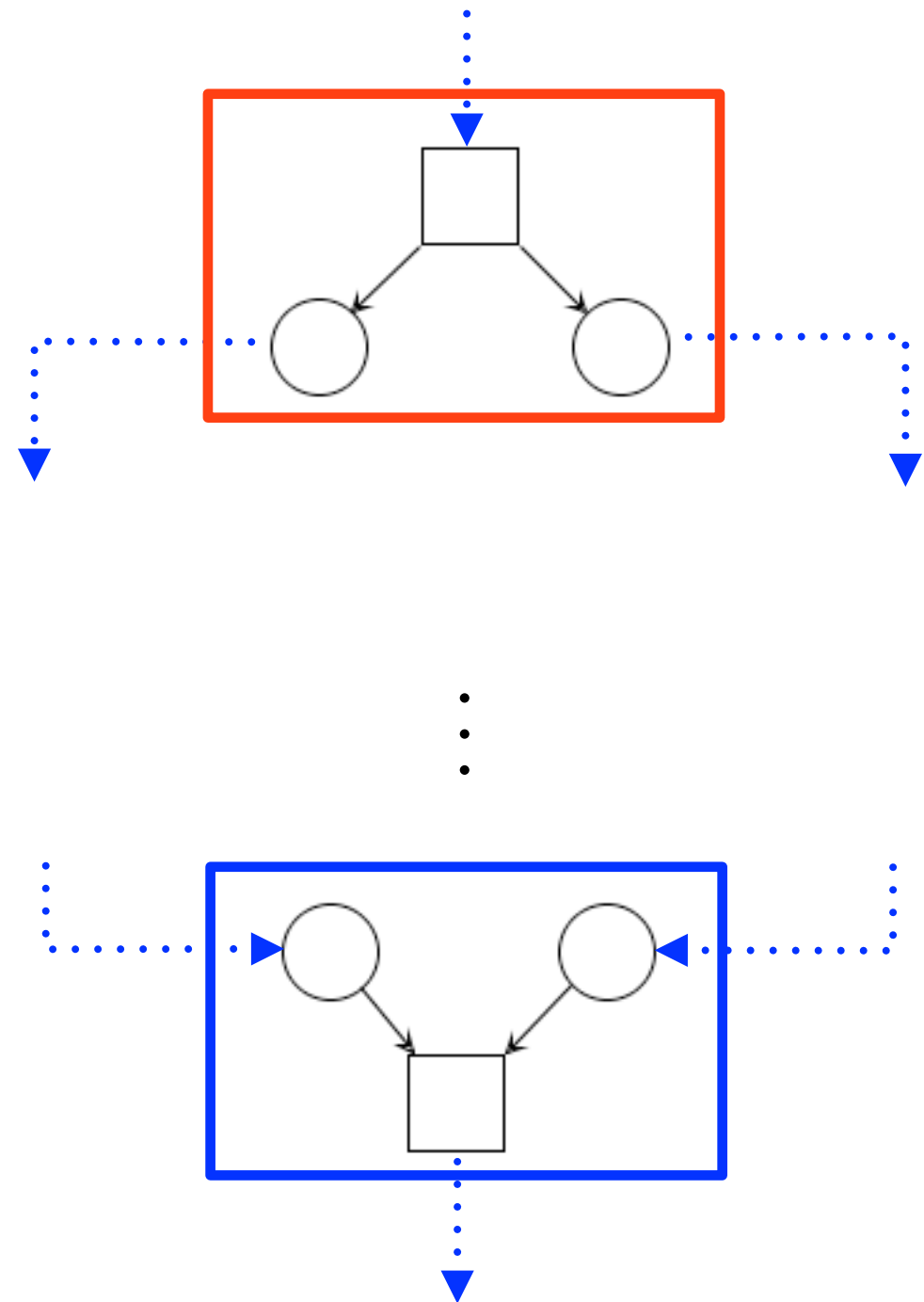


Step 1: OR join (wfa)

EPC element

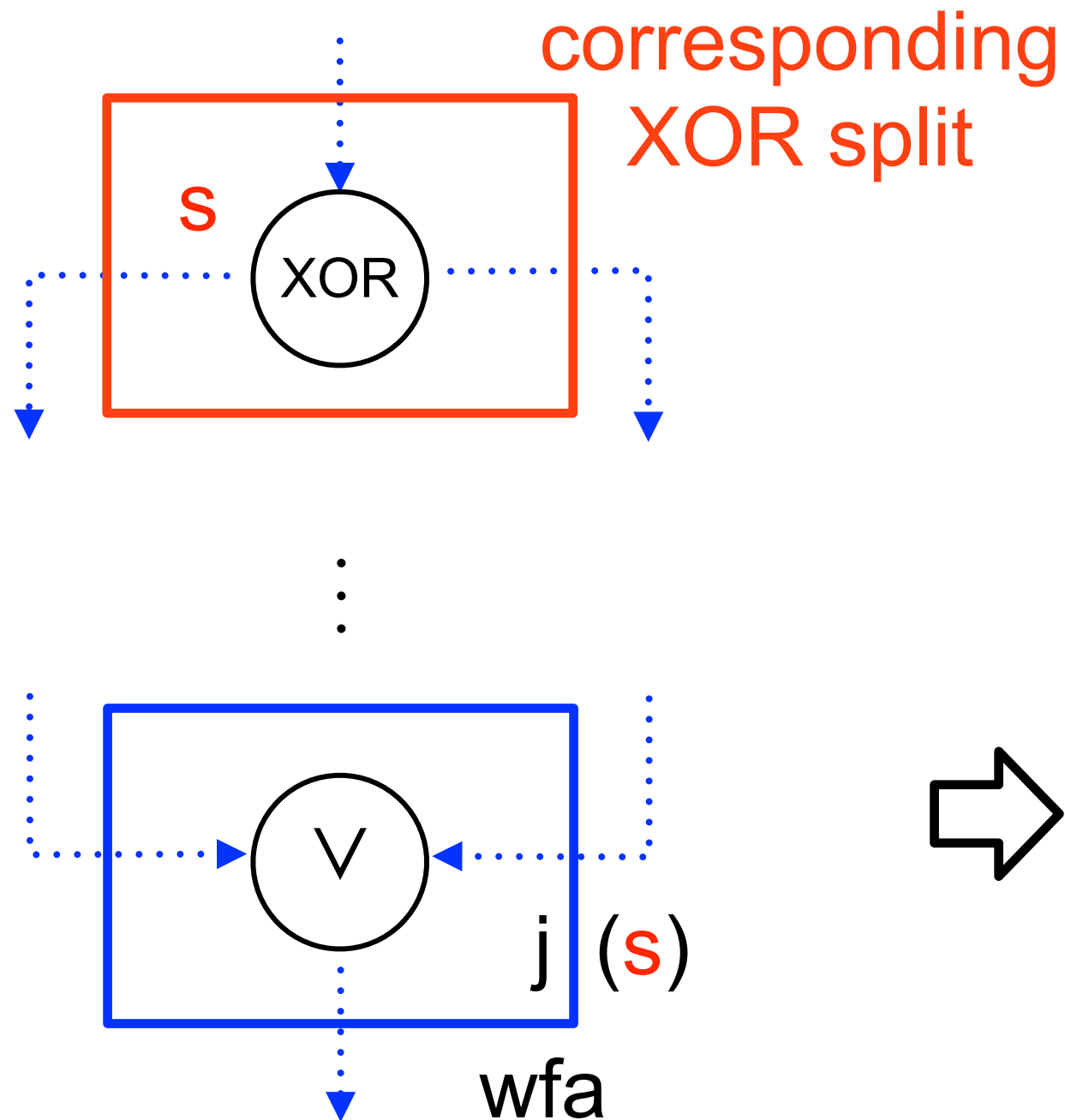


net fragment

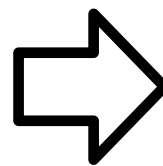
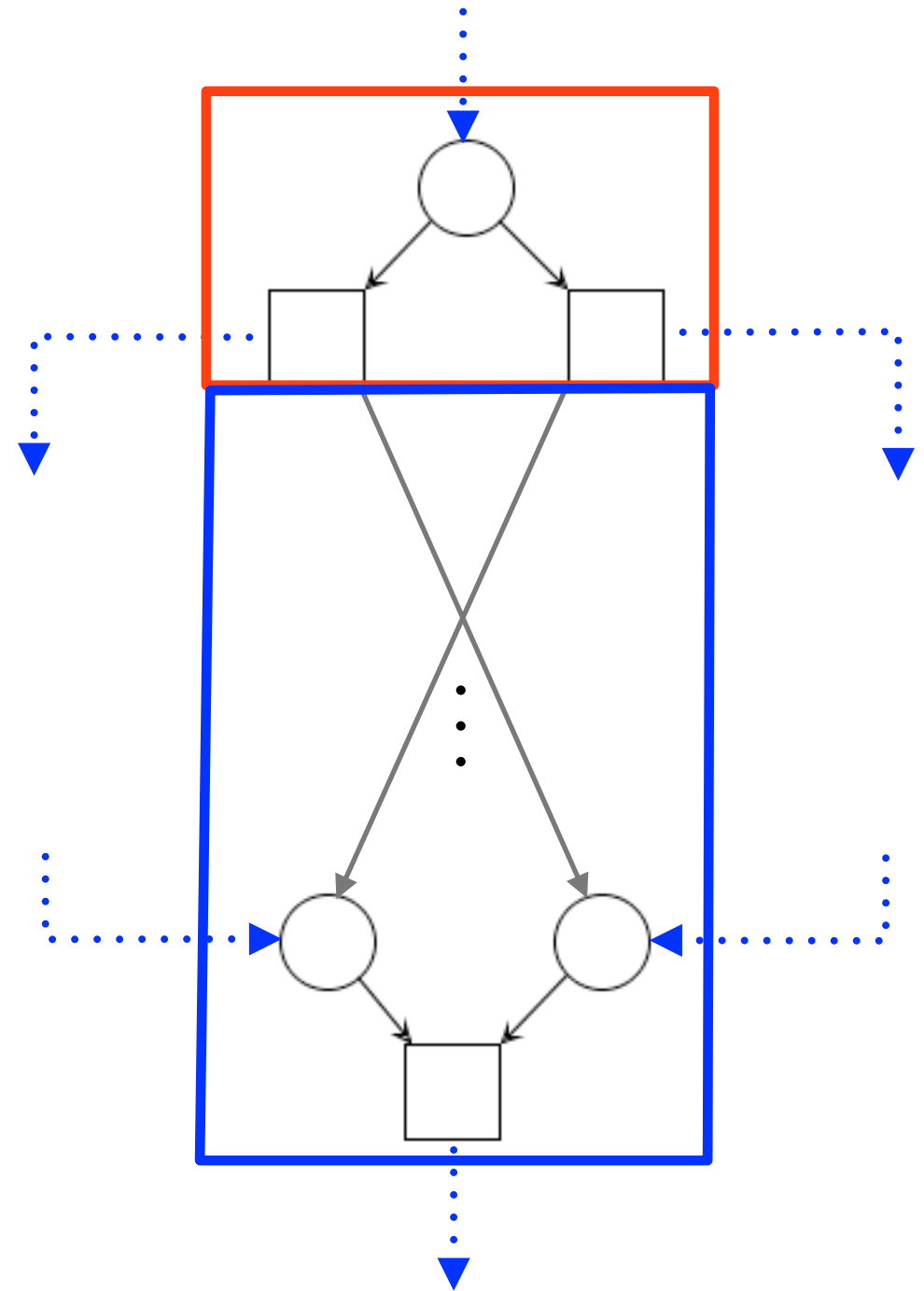


Step 1: OR join (wfa)

EPC element



net fragment



Assumption

...

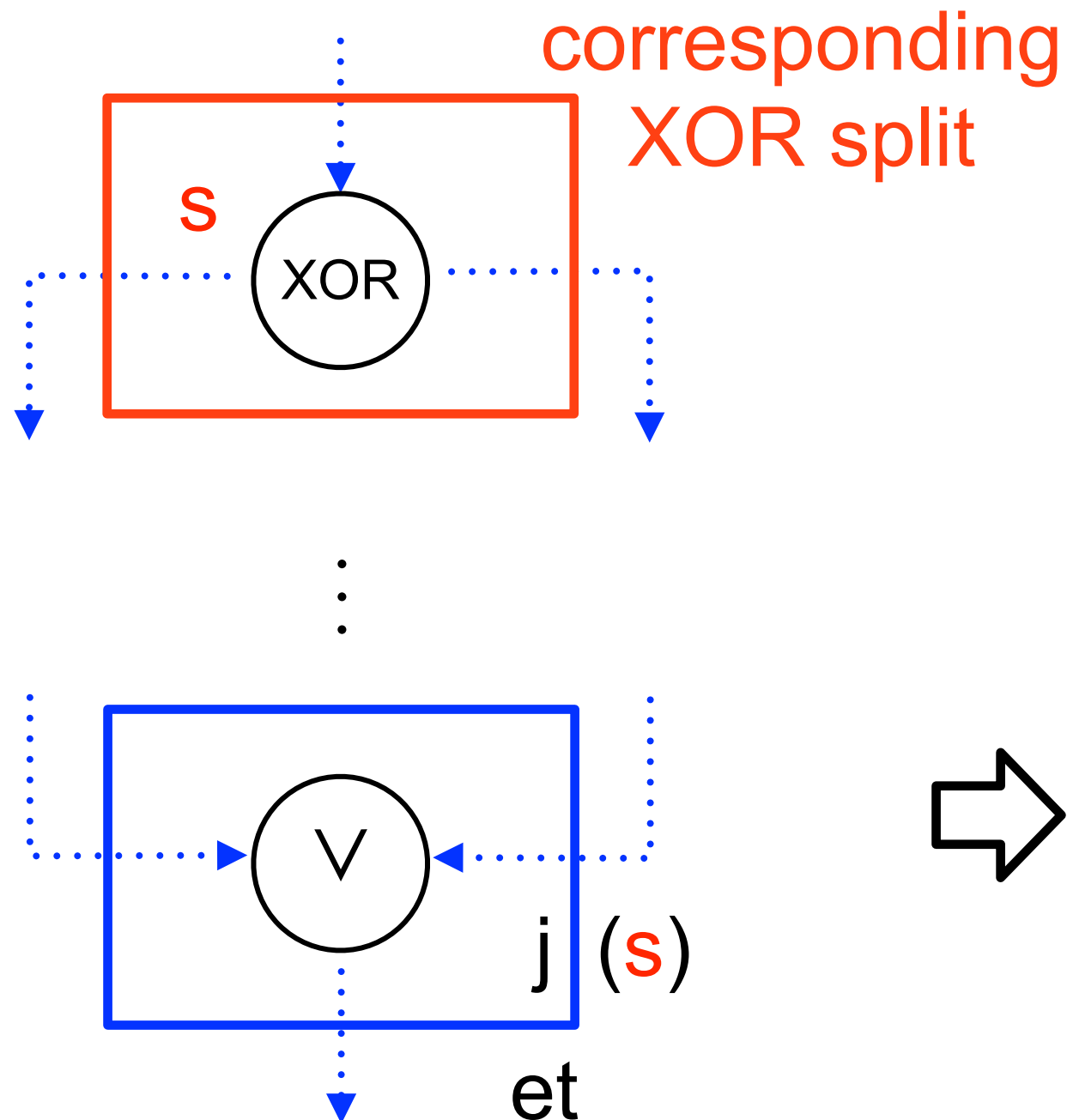
If an OR join has non-matching corresponding split
it is decorated with a policy (wfa, fc, et)

et: every-time
works well with corresponding XOR split

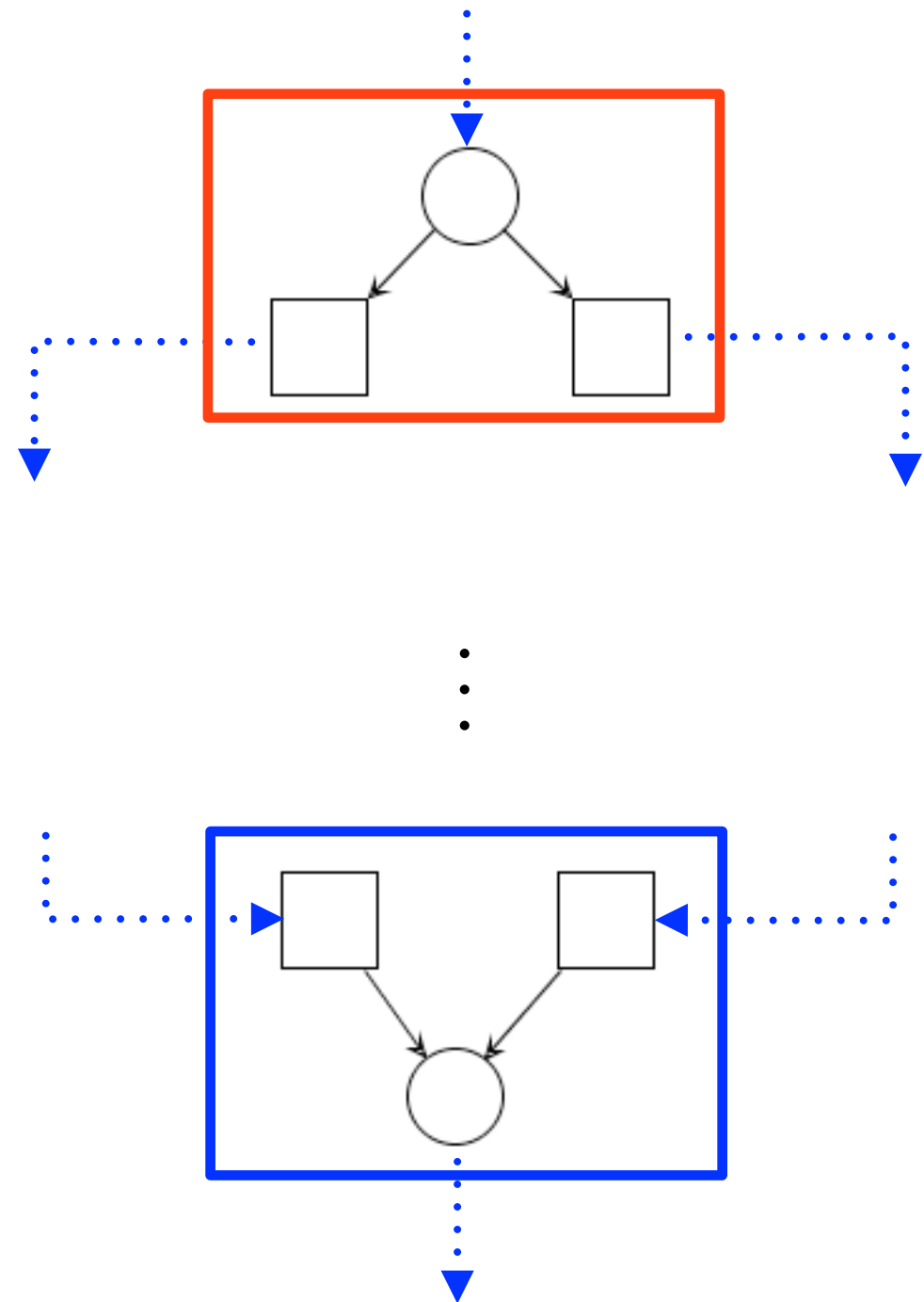
...

Step 1: OR join (et)

EPC element



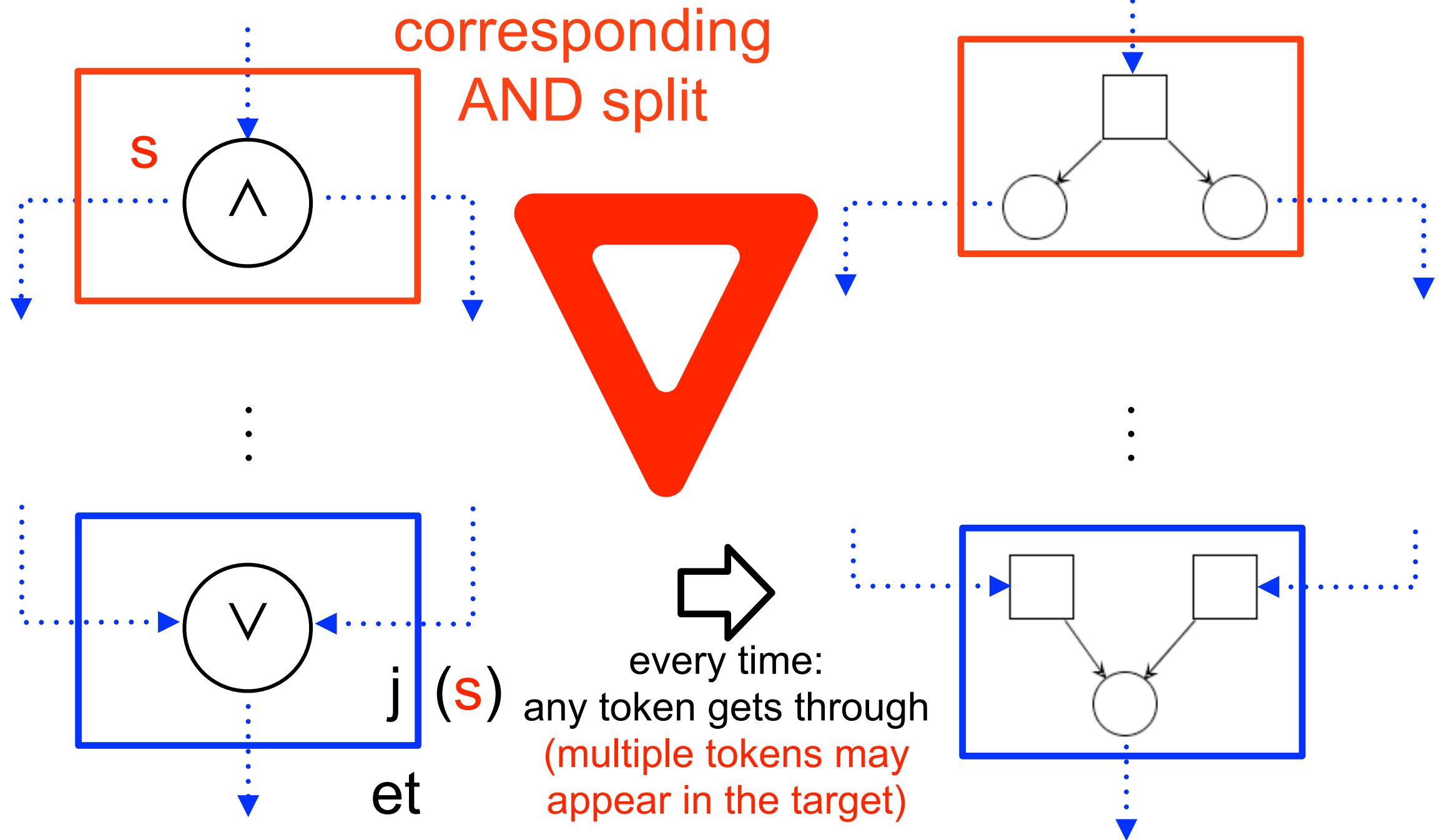
net fragment



Step 1: OR join (et)

EPC element

net fragment



Assumption

...

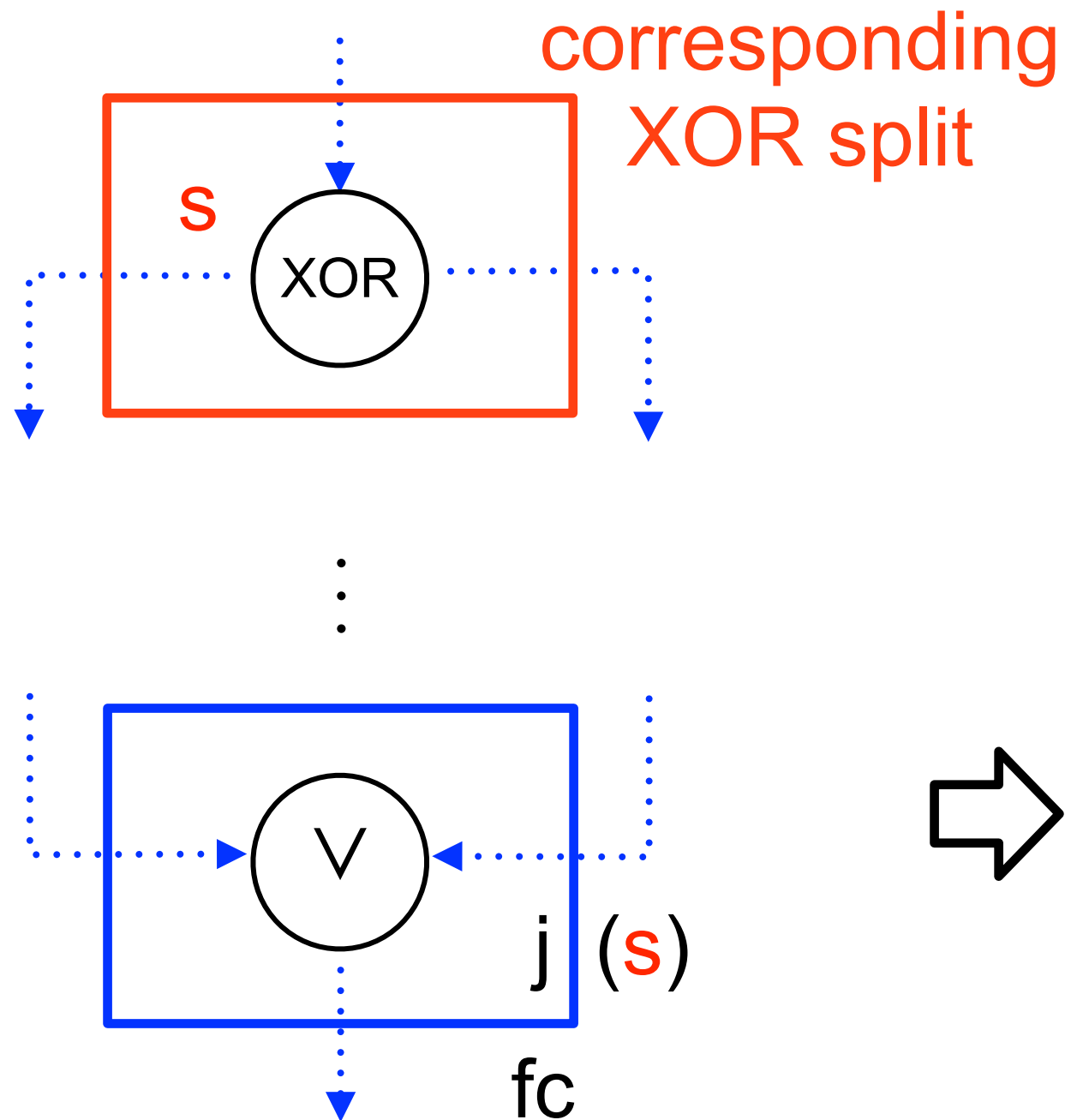
If an OR join has non-matching corresponding split
it is decorated with a policy (wfa, fc, et)

fc: first-come
works well with corresponding XOR split

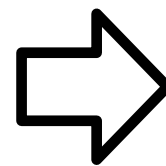
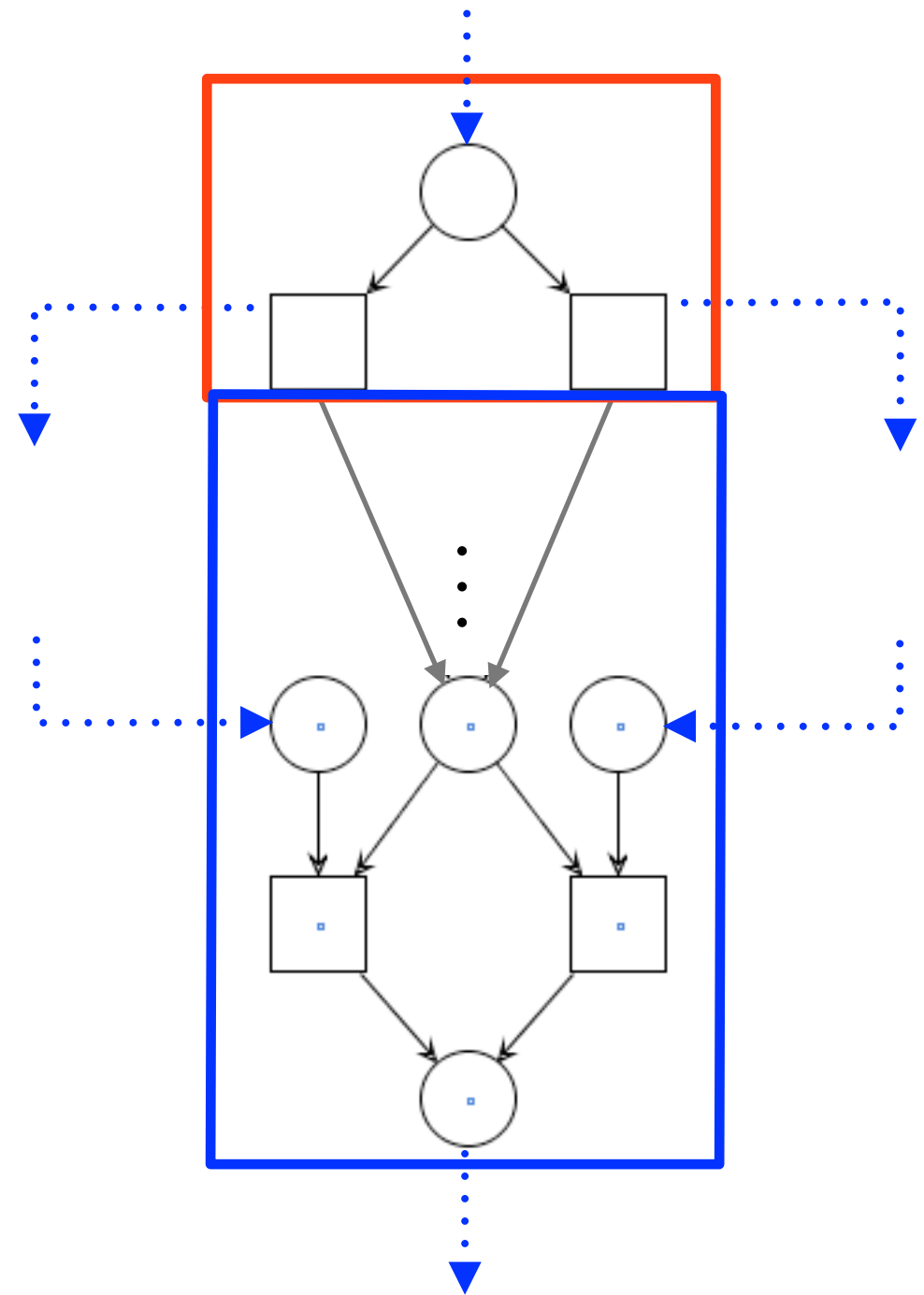
...

Step 1: OR join (fc)

EPC element



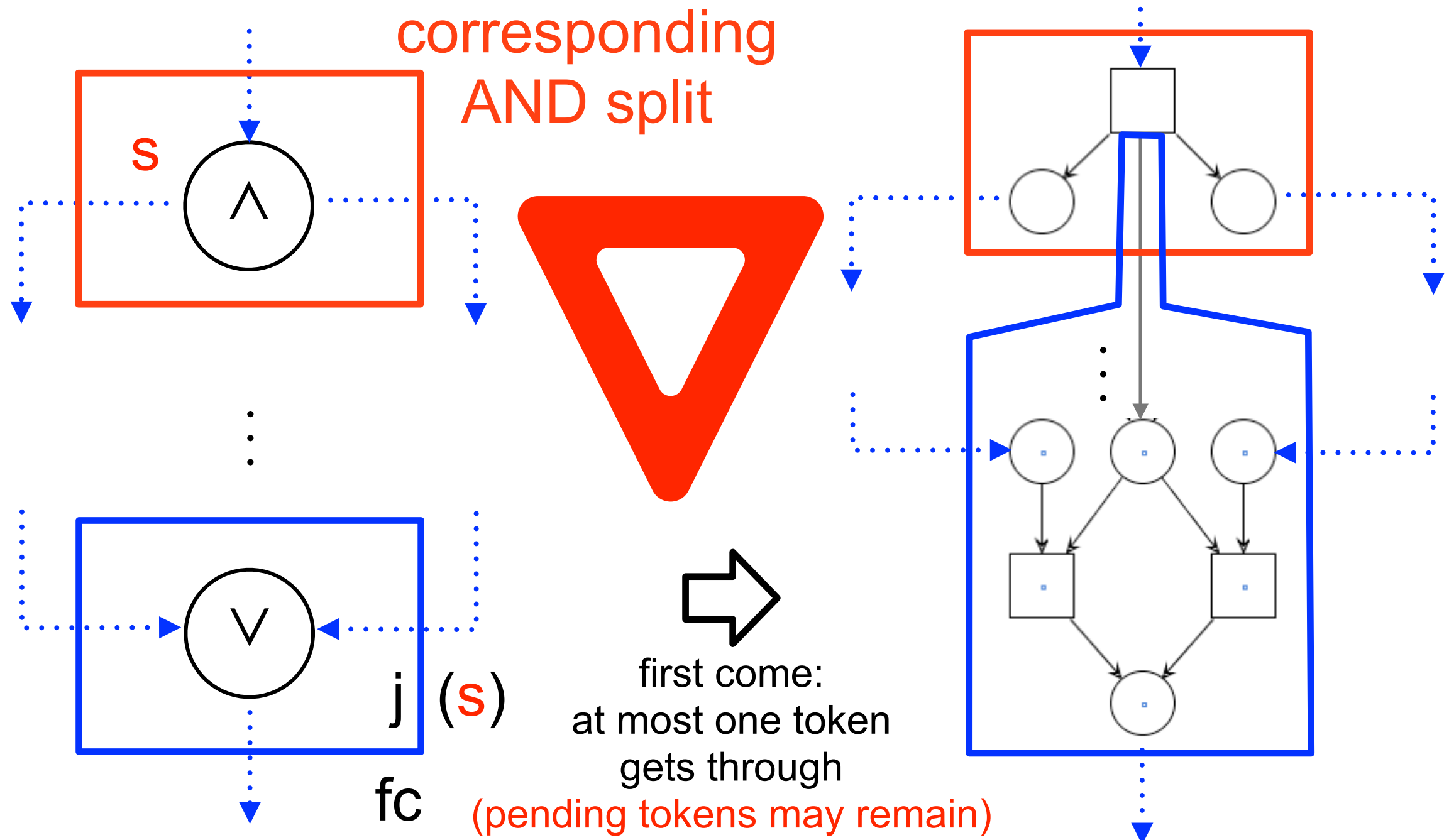
net fragment



Step 1: OR join (fc)

EPC element

net fragment



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

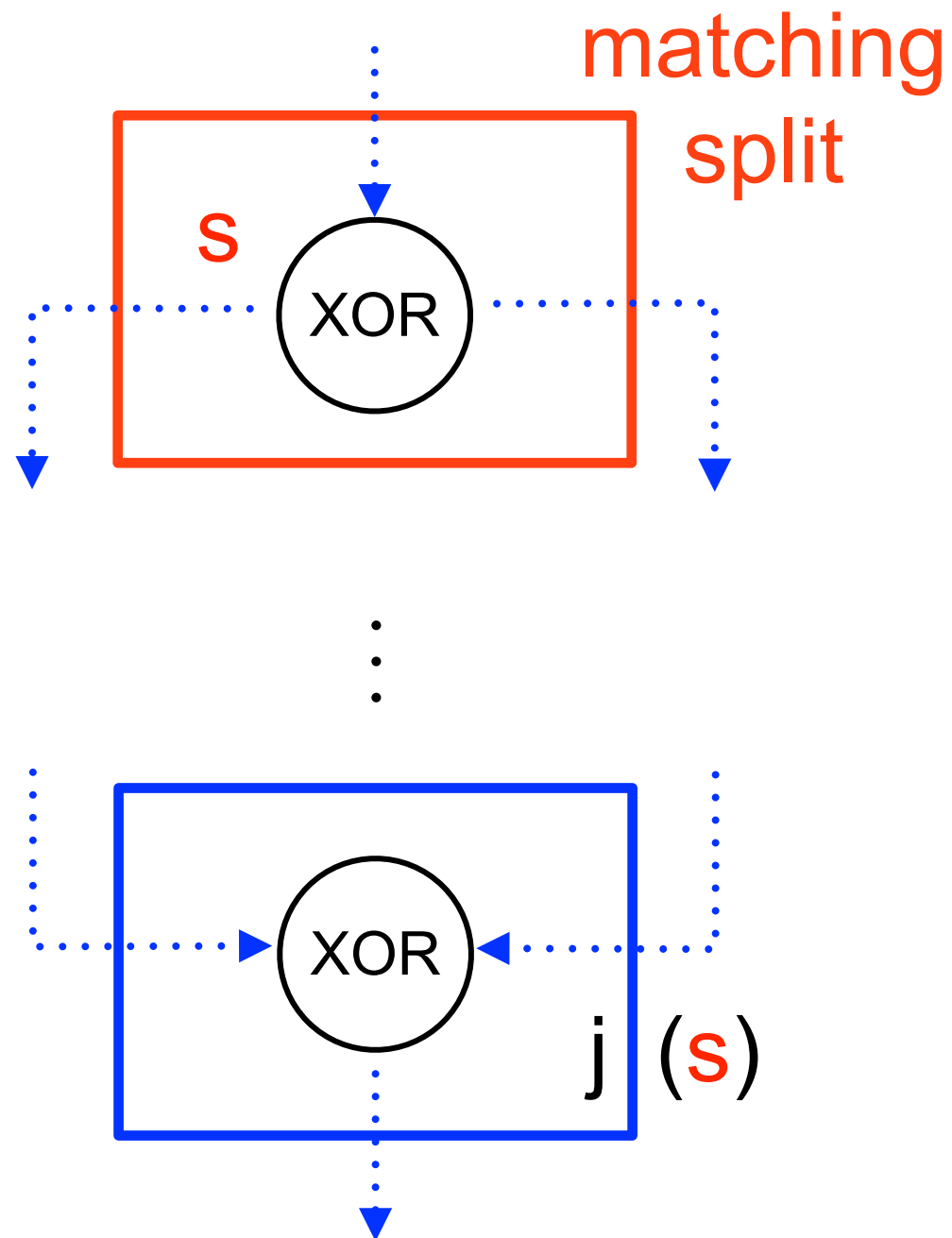
...

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

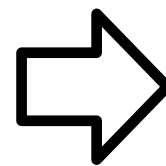
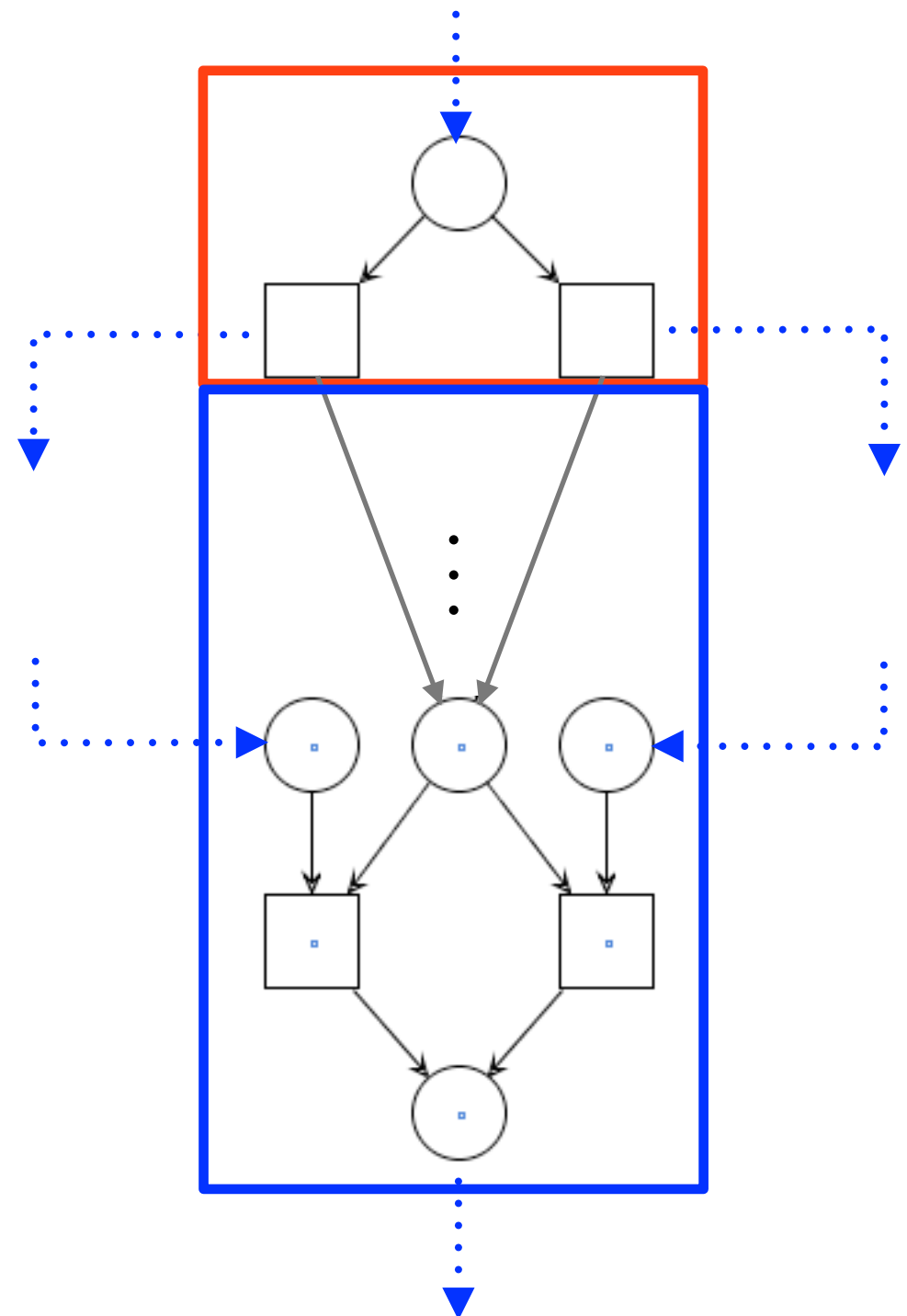
...

Step 1: XOR join

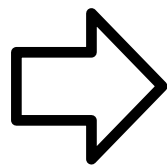
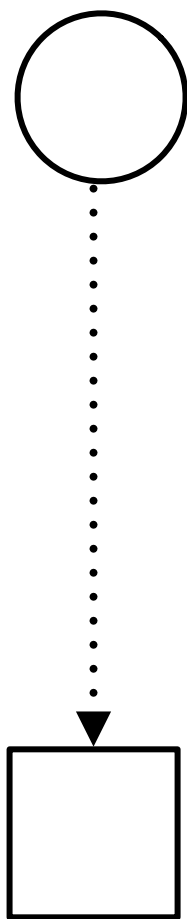
EPC element



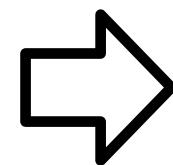
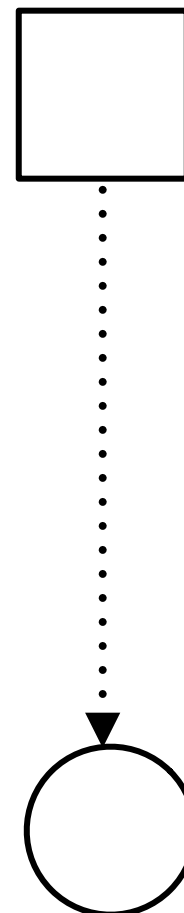
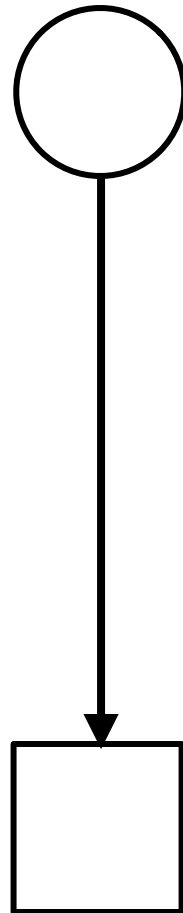
net fragment



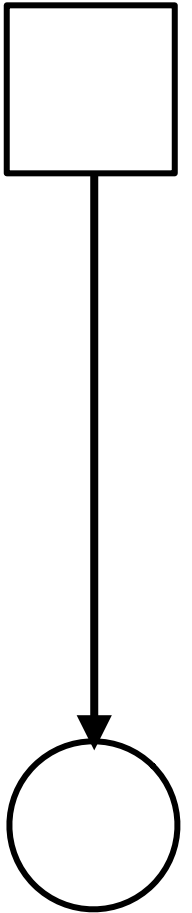
Step 2: dummy style



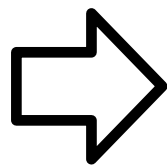
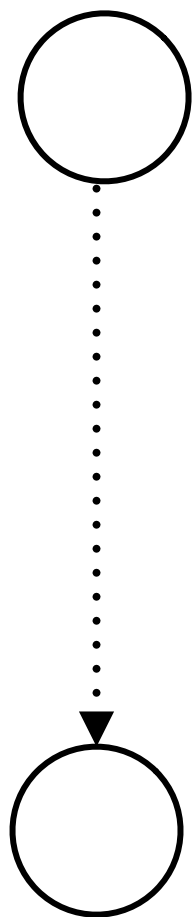
straight conversion



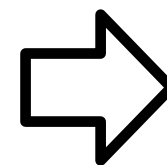
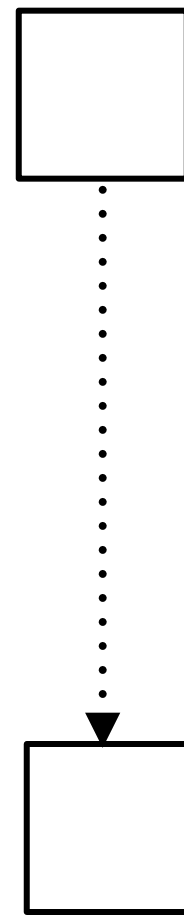
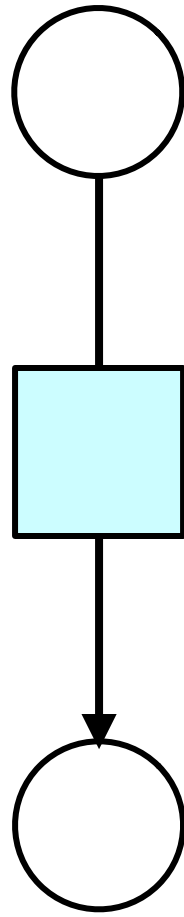
straight conversion



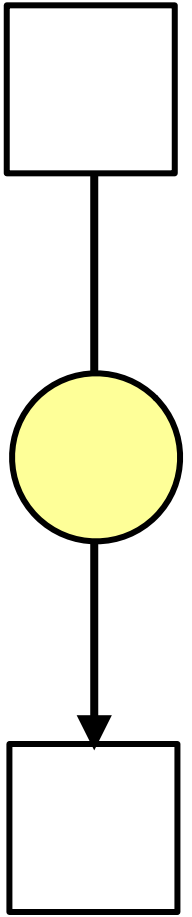
Step 2: dummy style



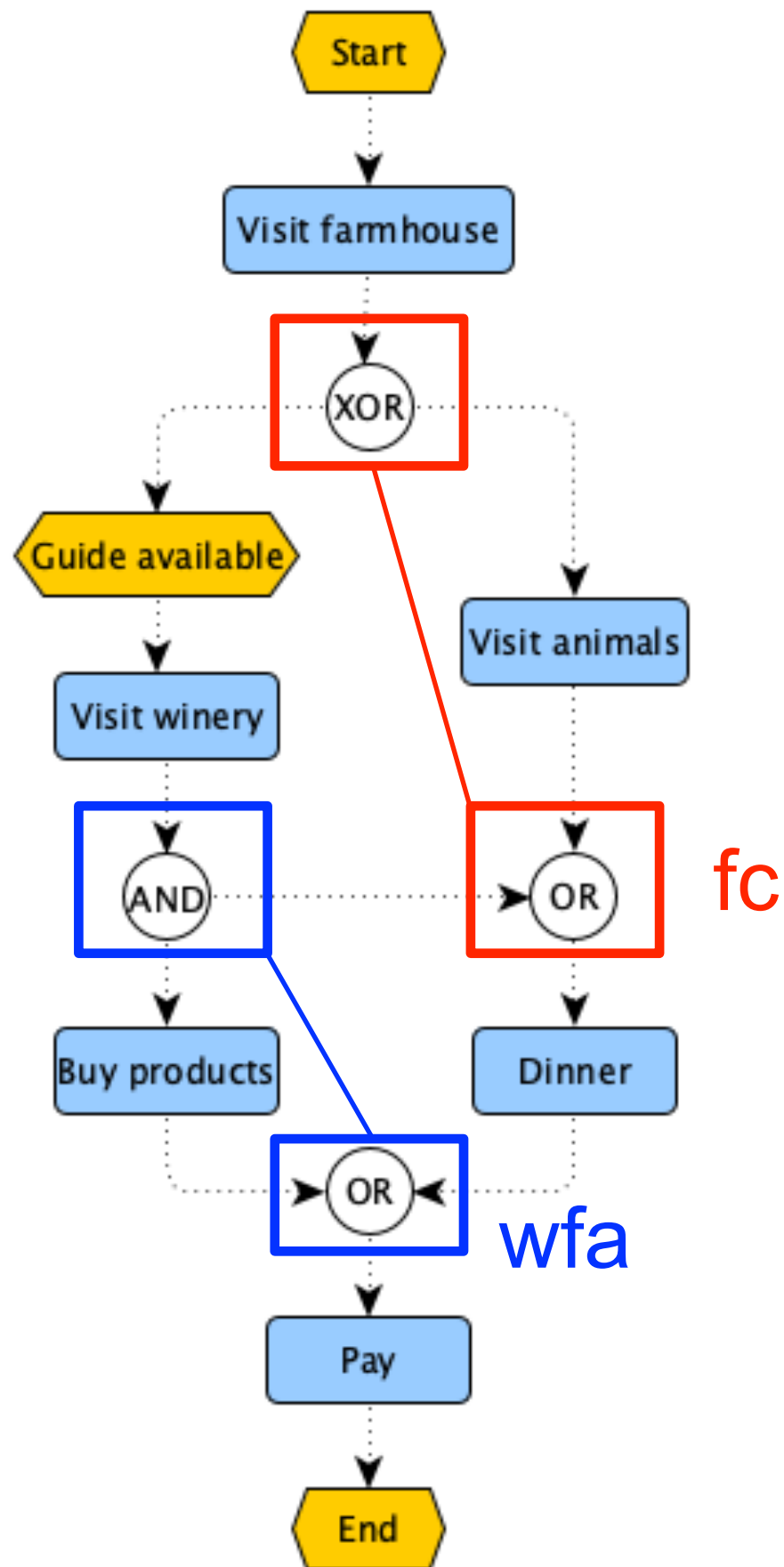
needs a
dummy transition



needs a
dummy place

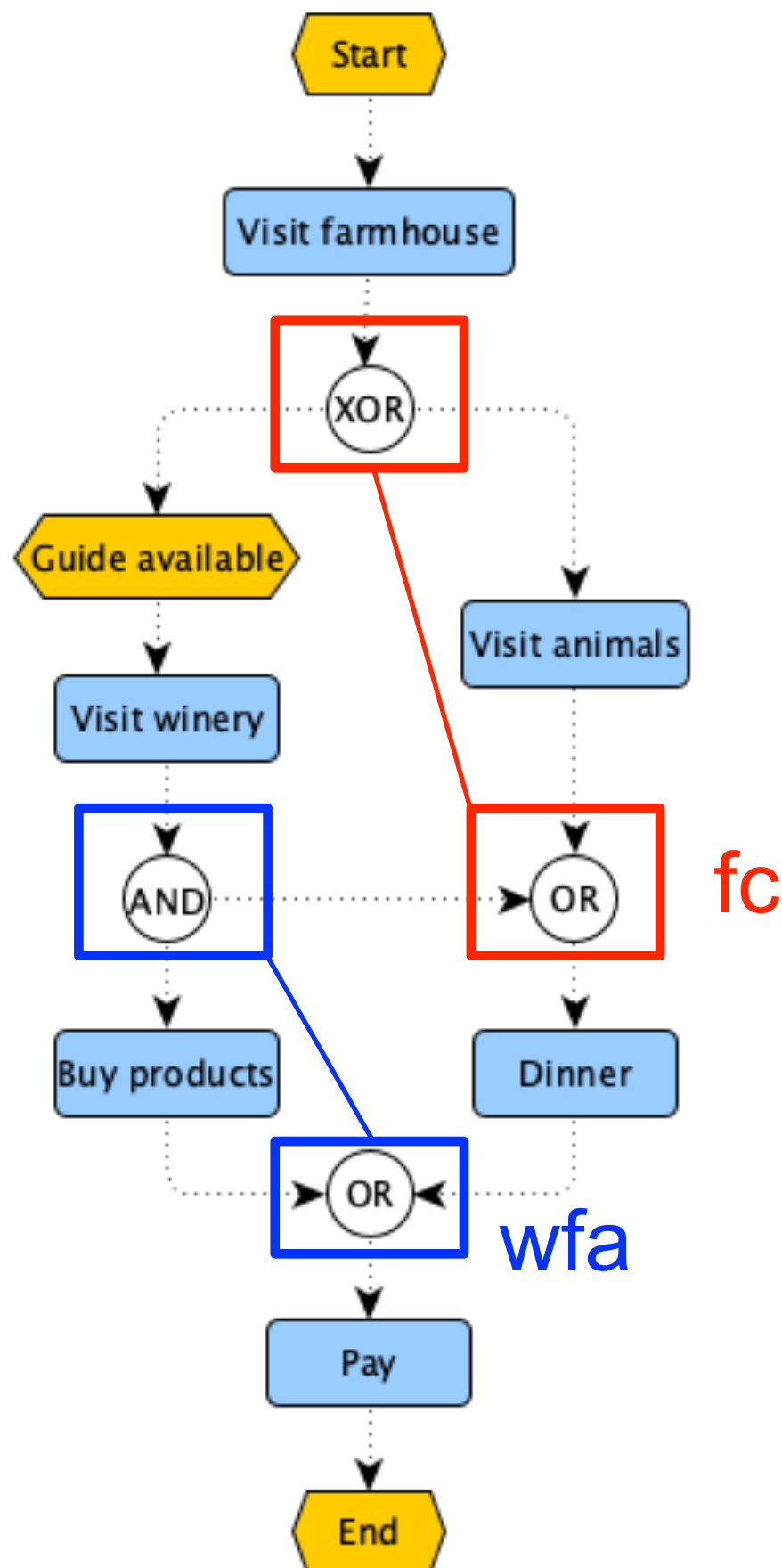


Example

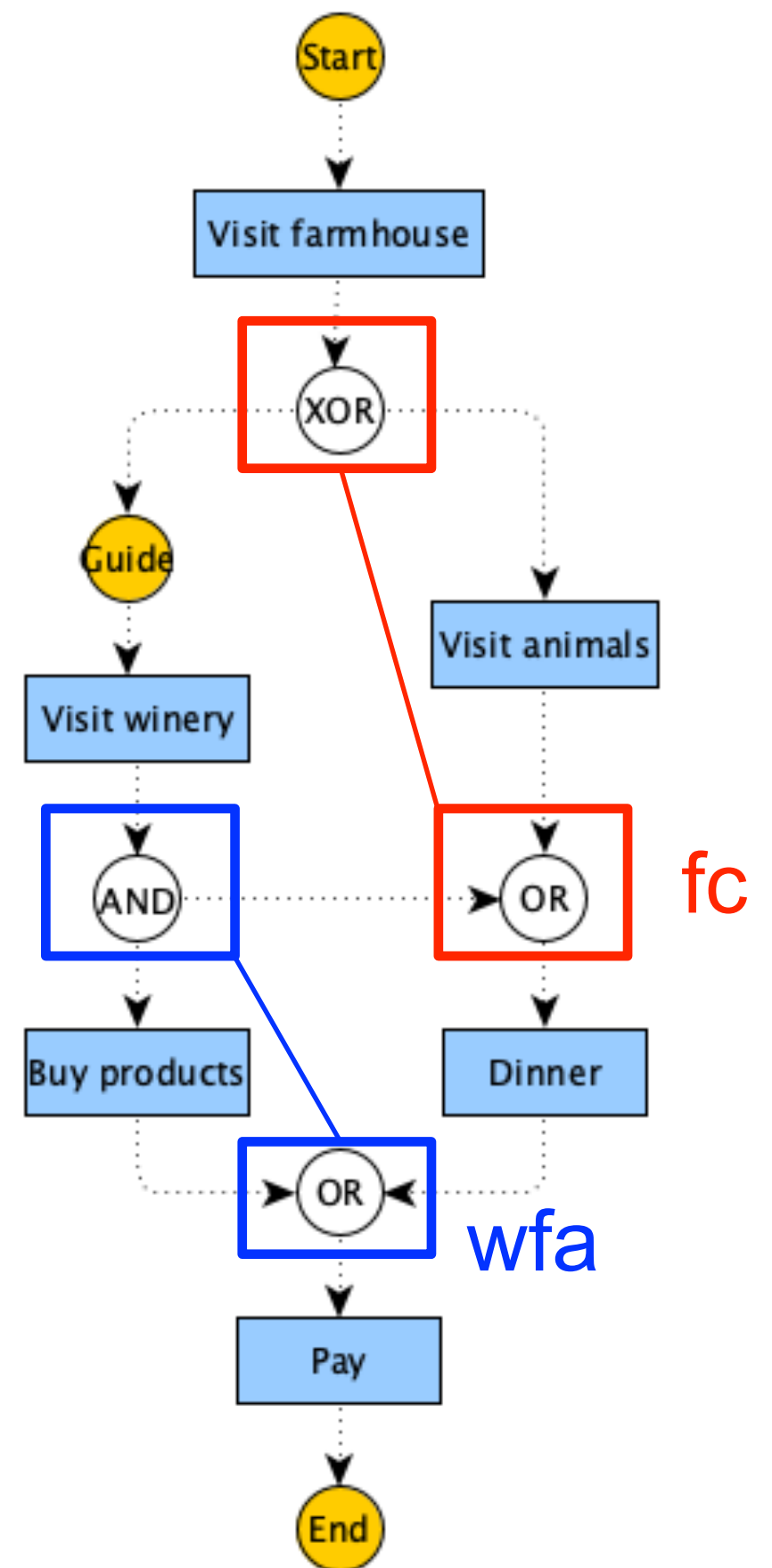


Sound?

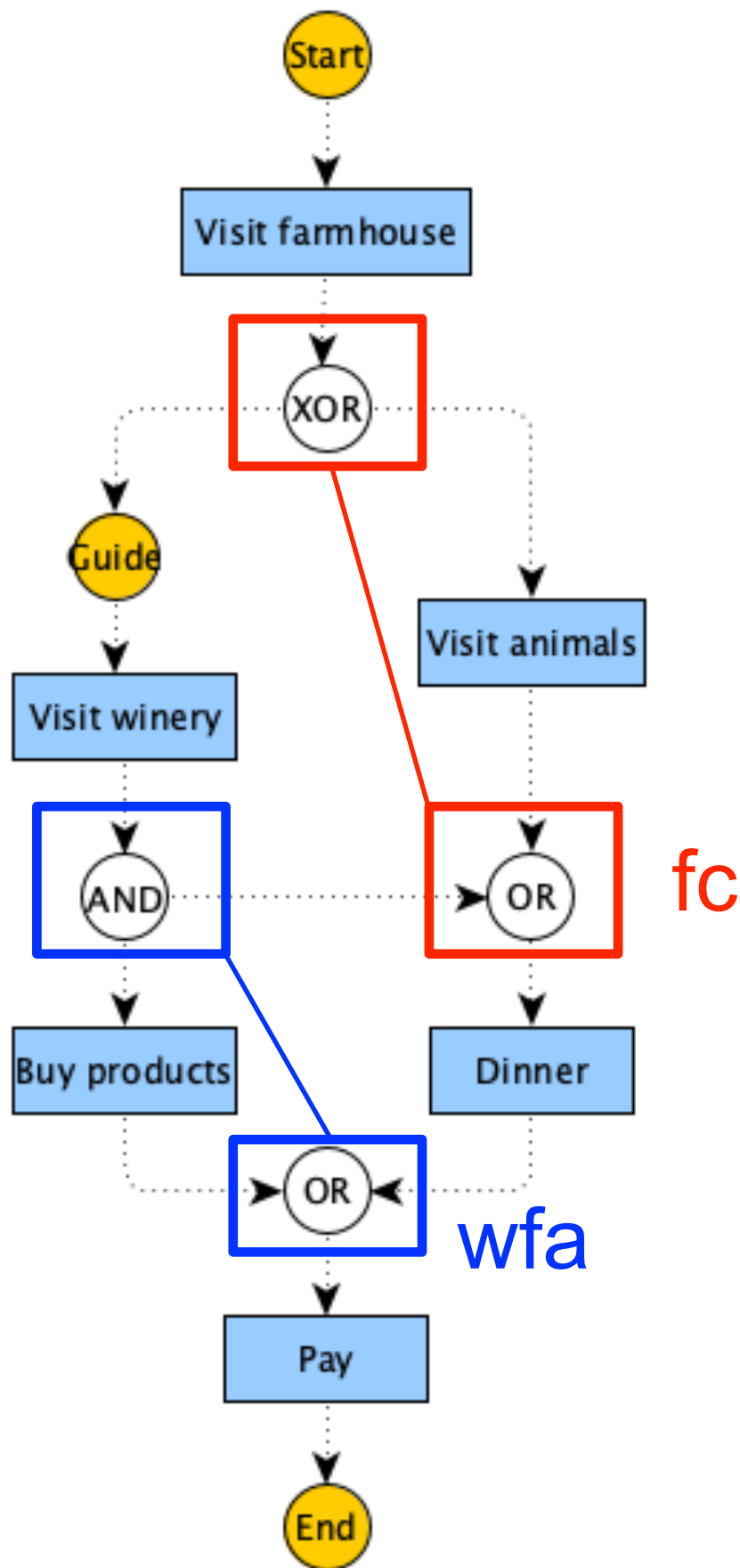
Example



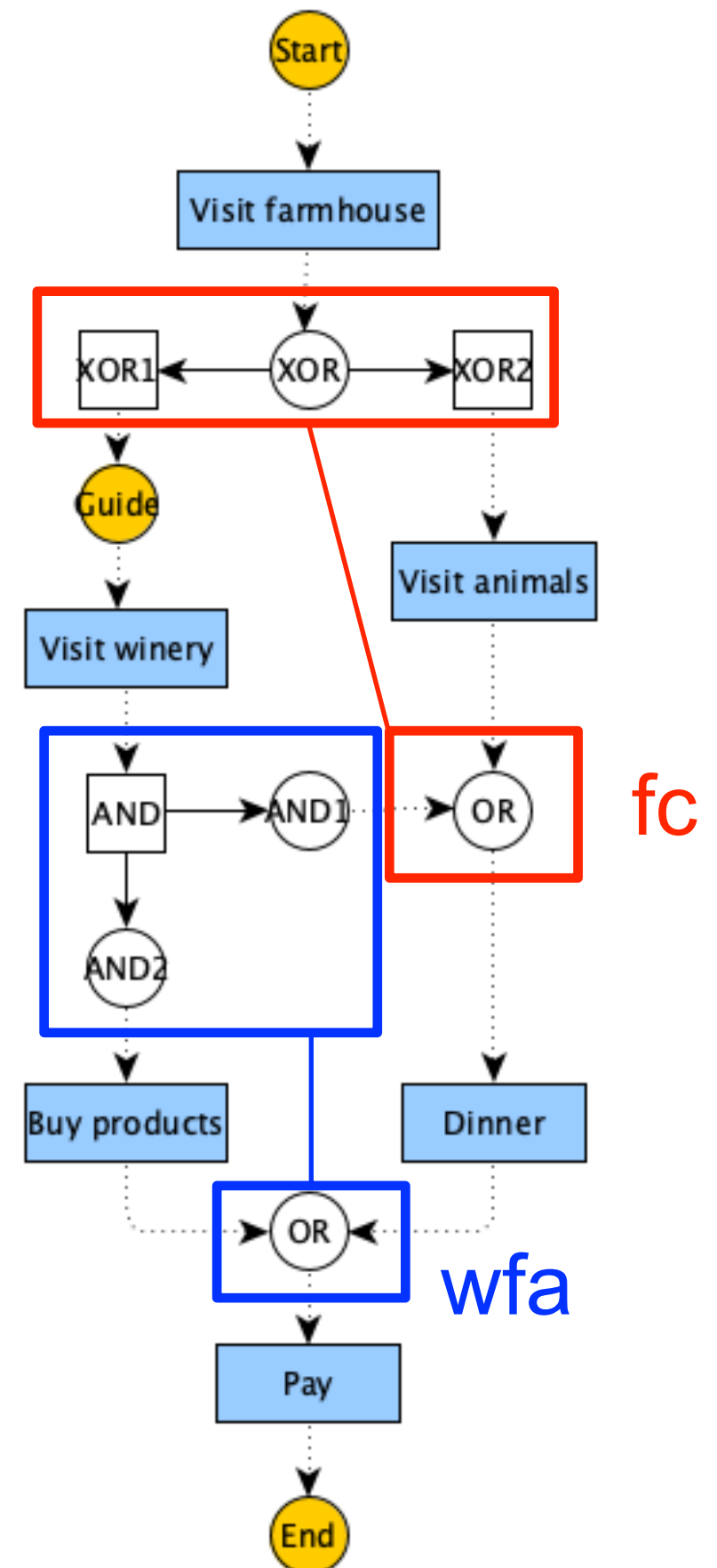
➡
Step 1
events and
functions



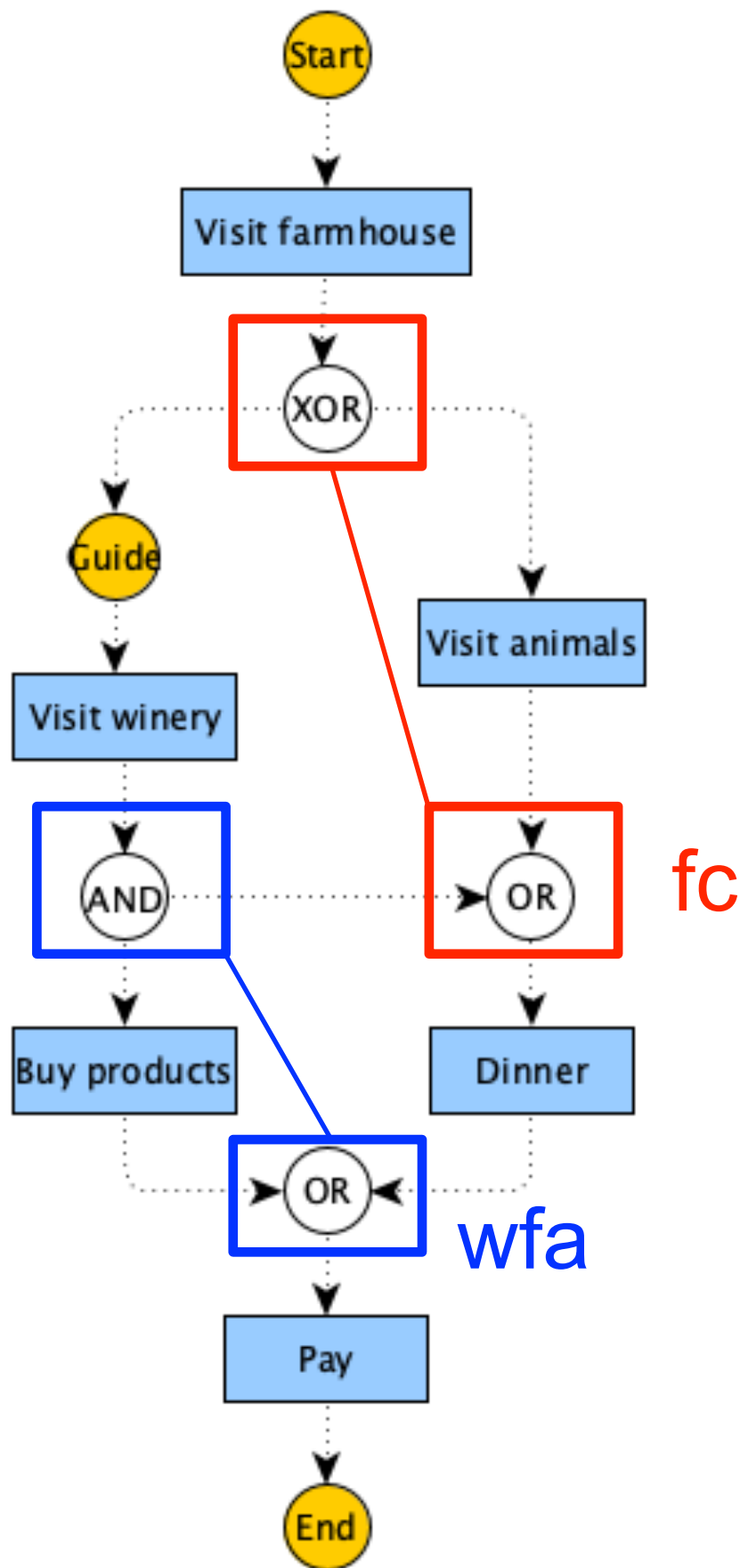
Example



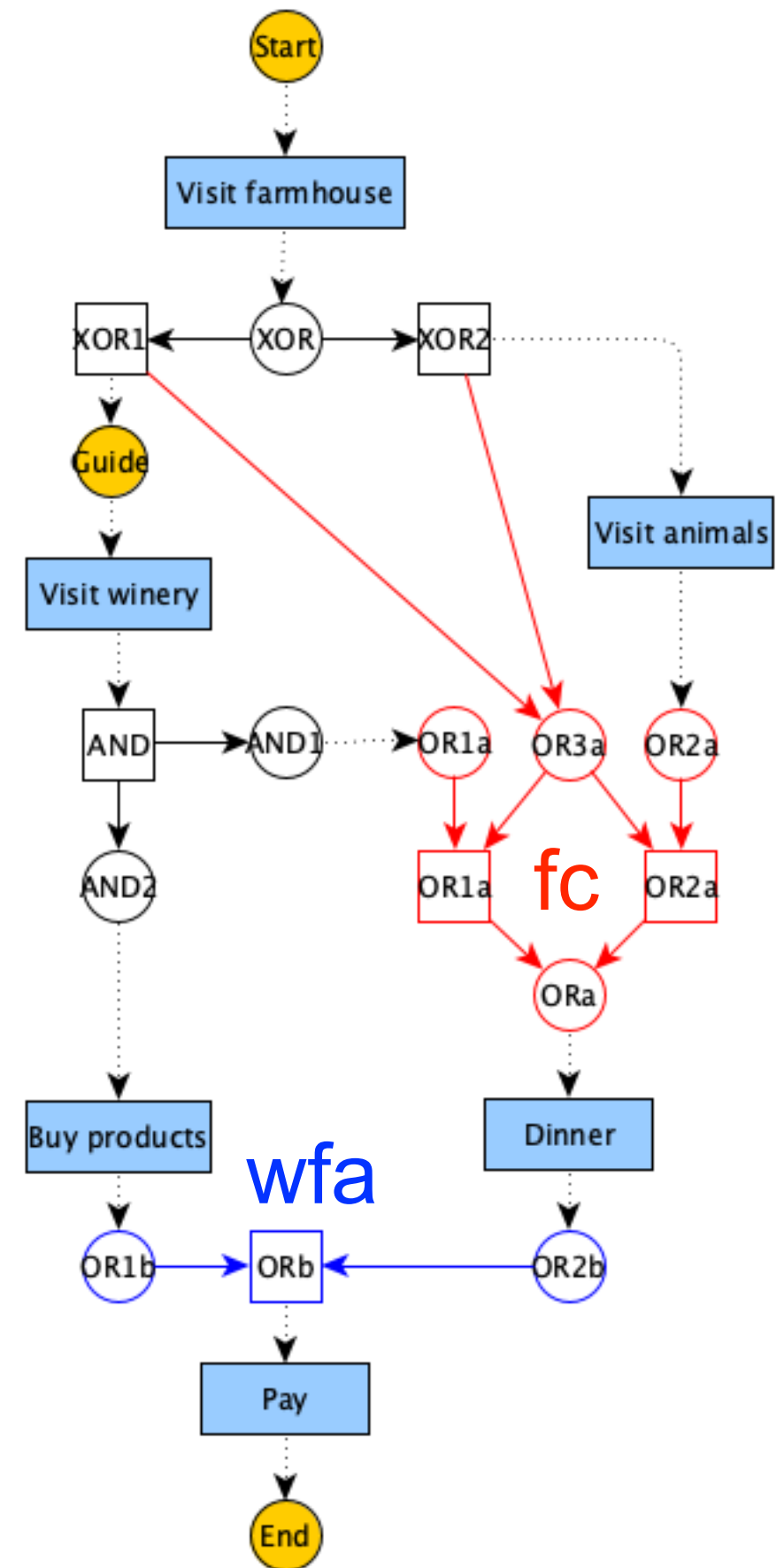
Step 1
splits



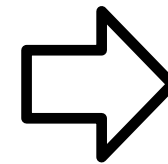
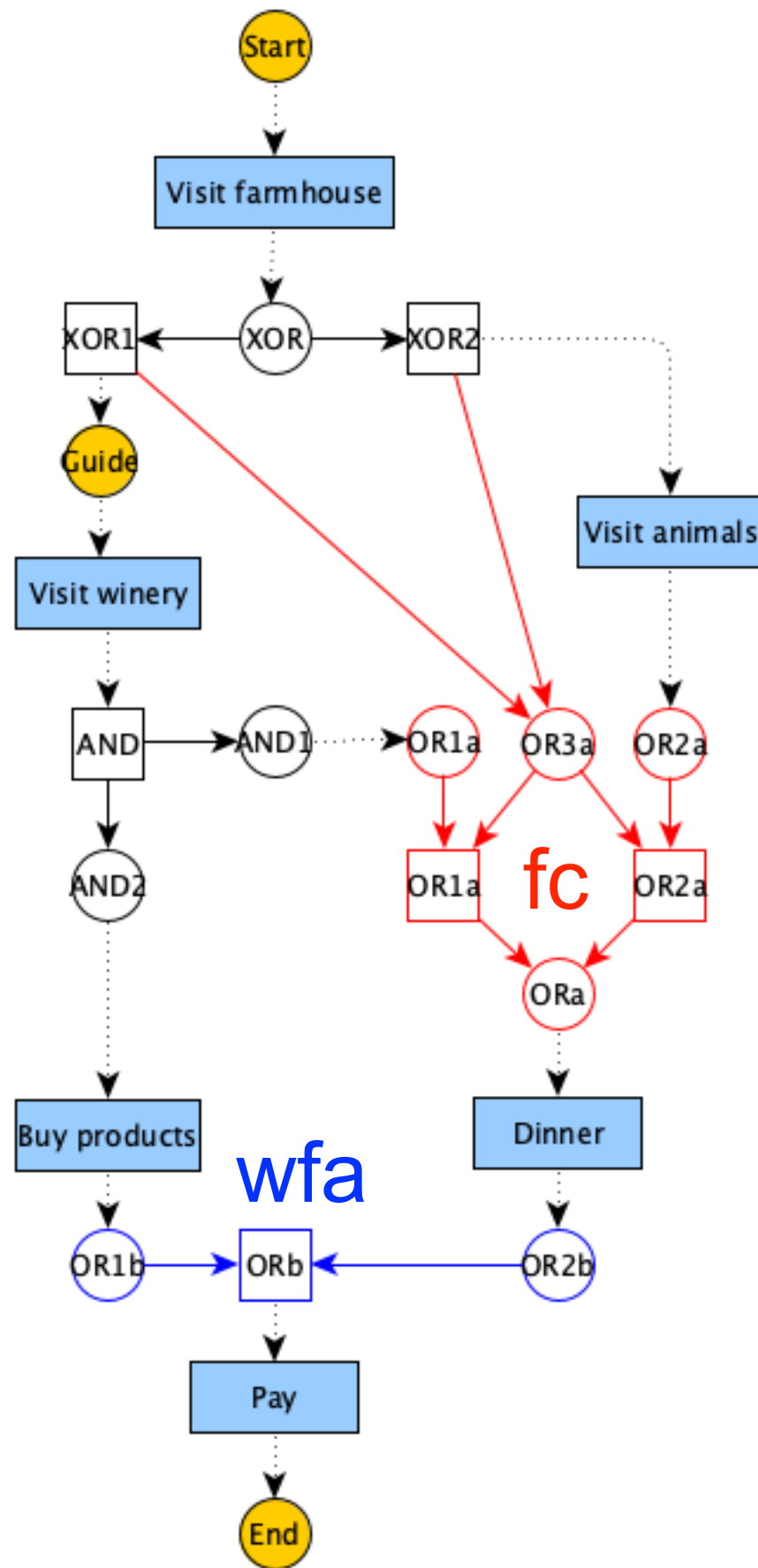
Example



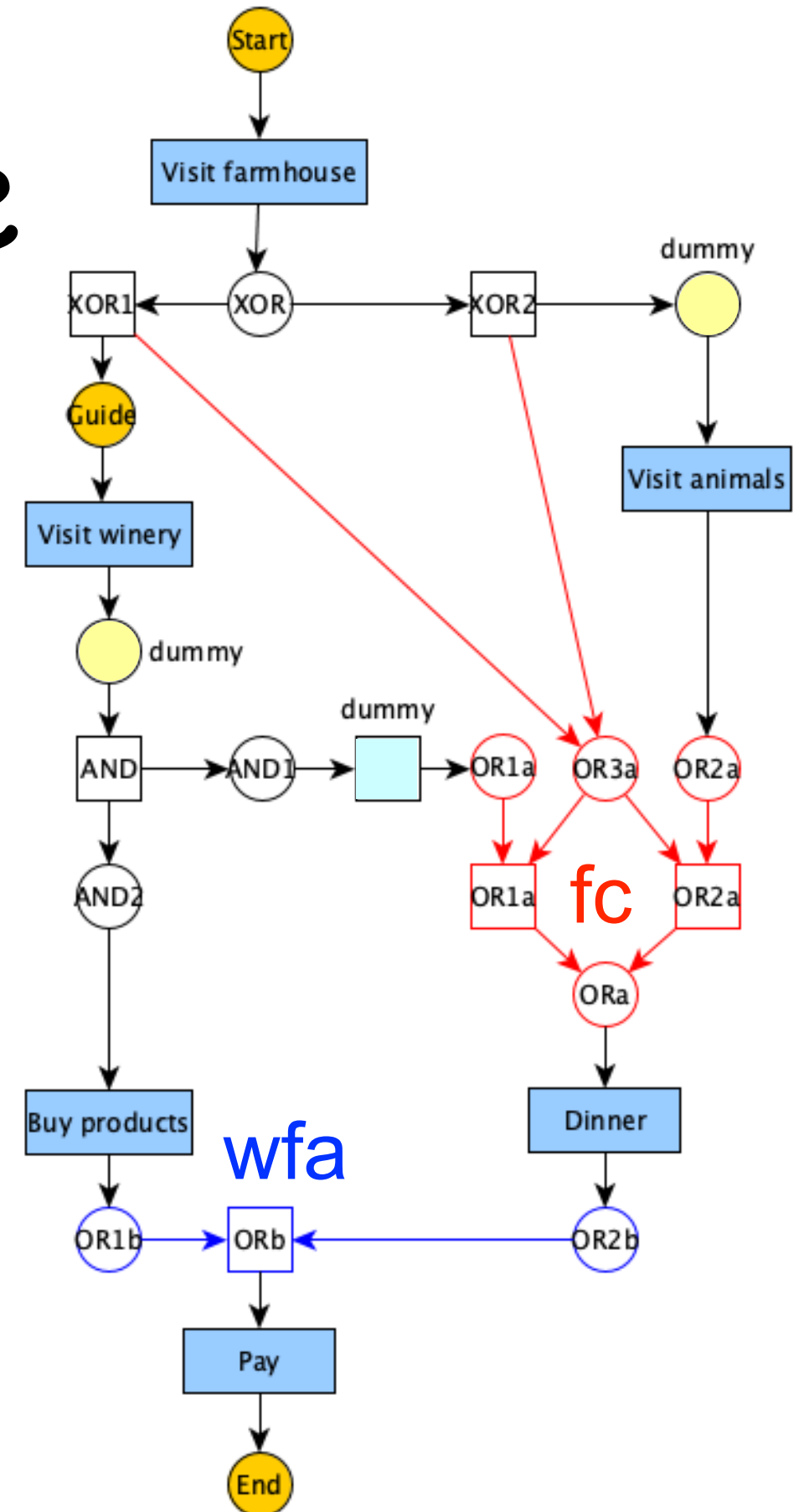
➡
Step 1
splits and
joins



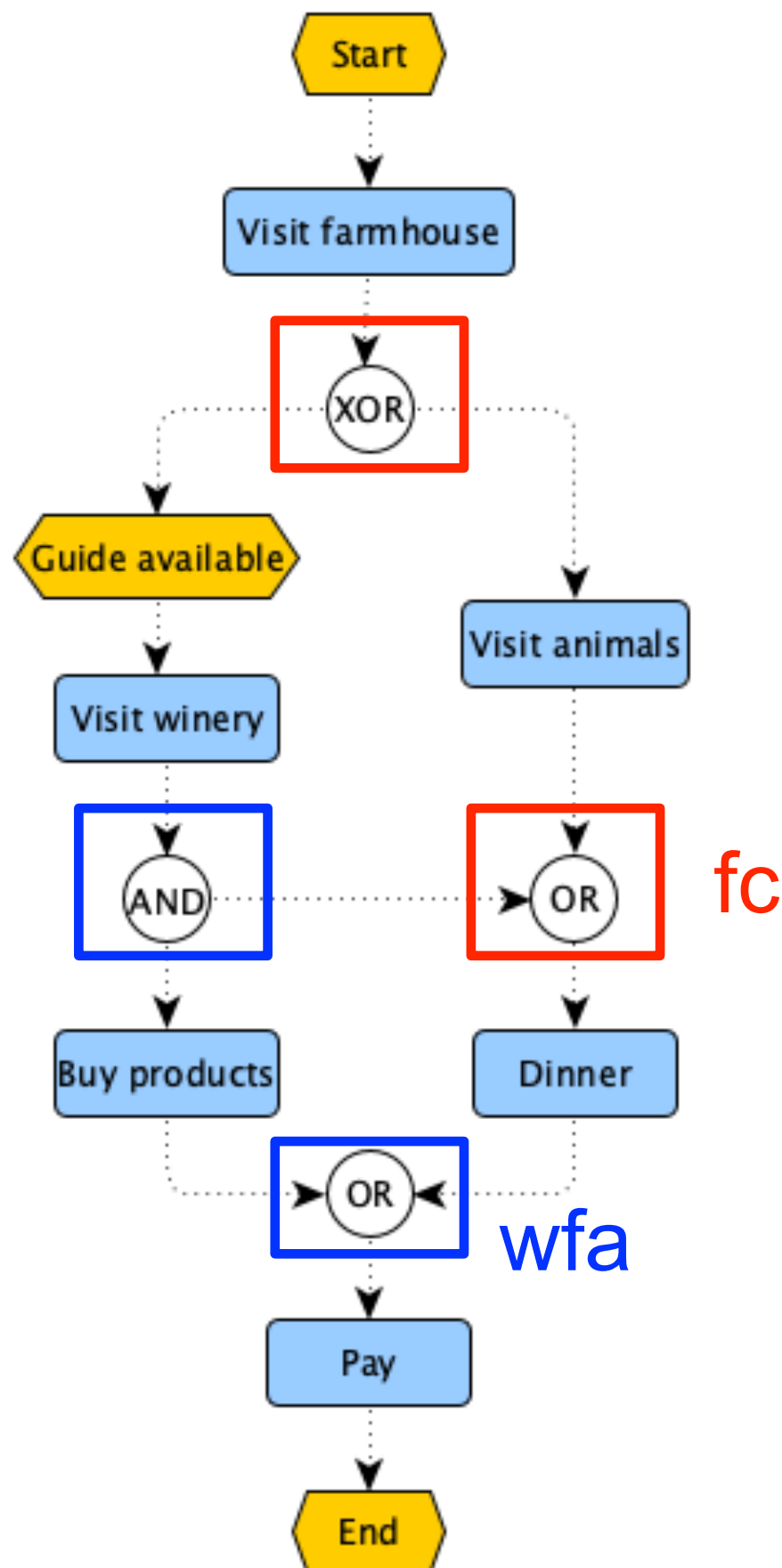
Example



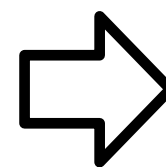
Step 2(+3)
dummy style



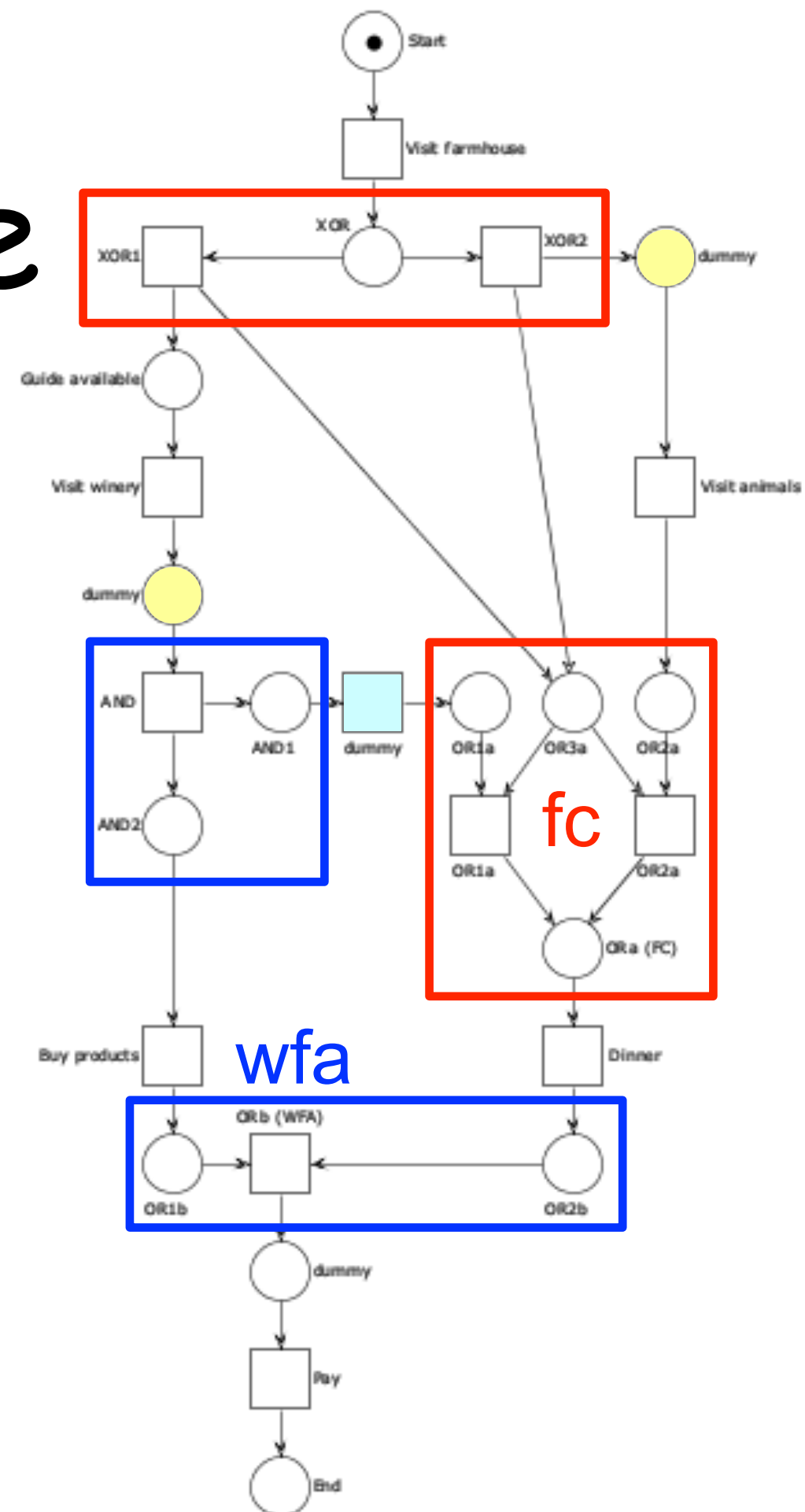
Example



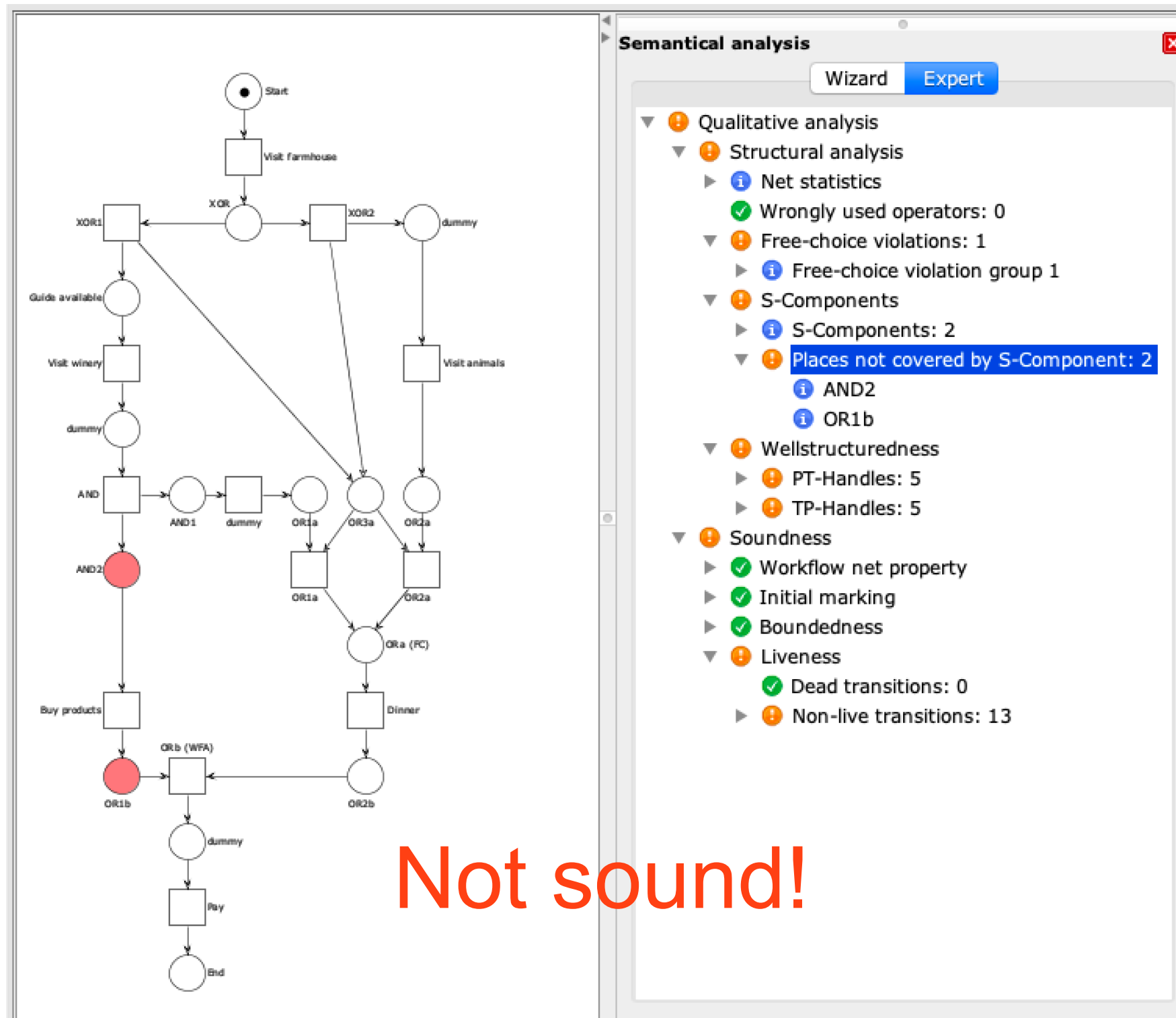
Sound?



Steps
1+2(+3)



Example



Not sound!

EPC pros and cons

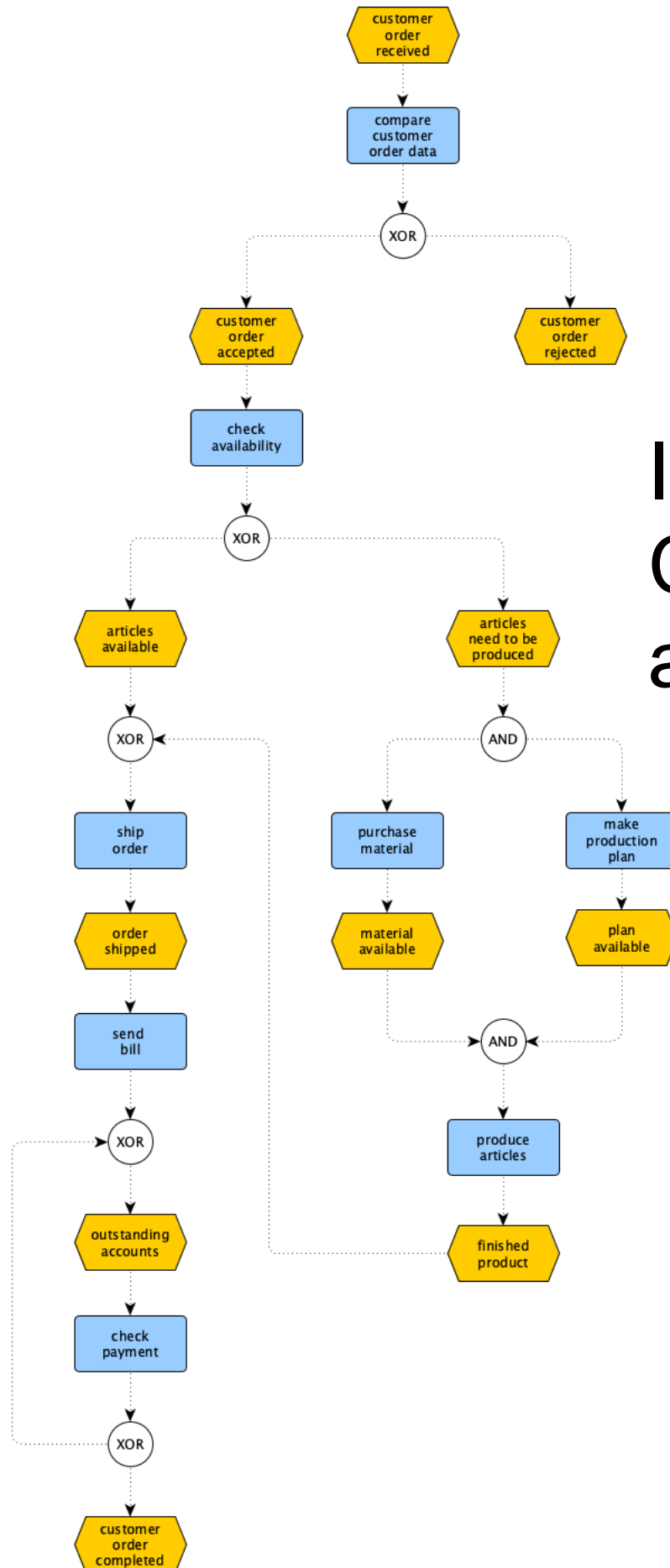
You may **leave complete freedom**,
but most diagrams will not be sound

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

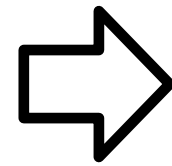
Exercise

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

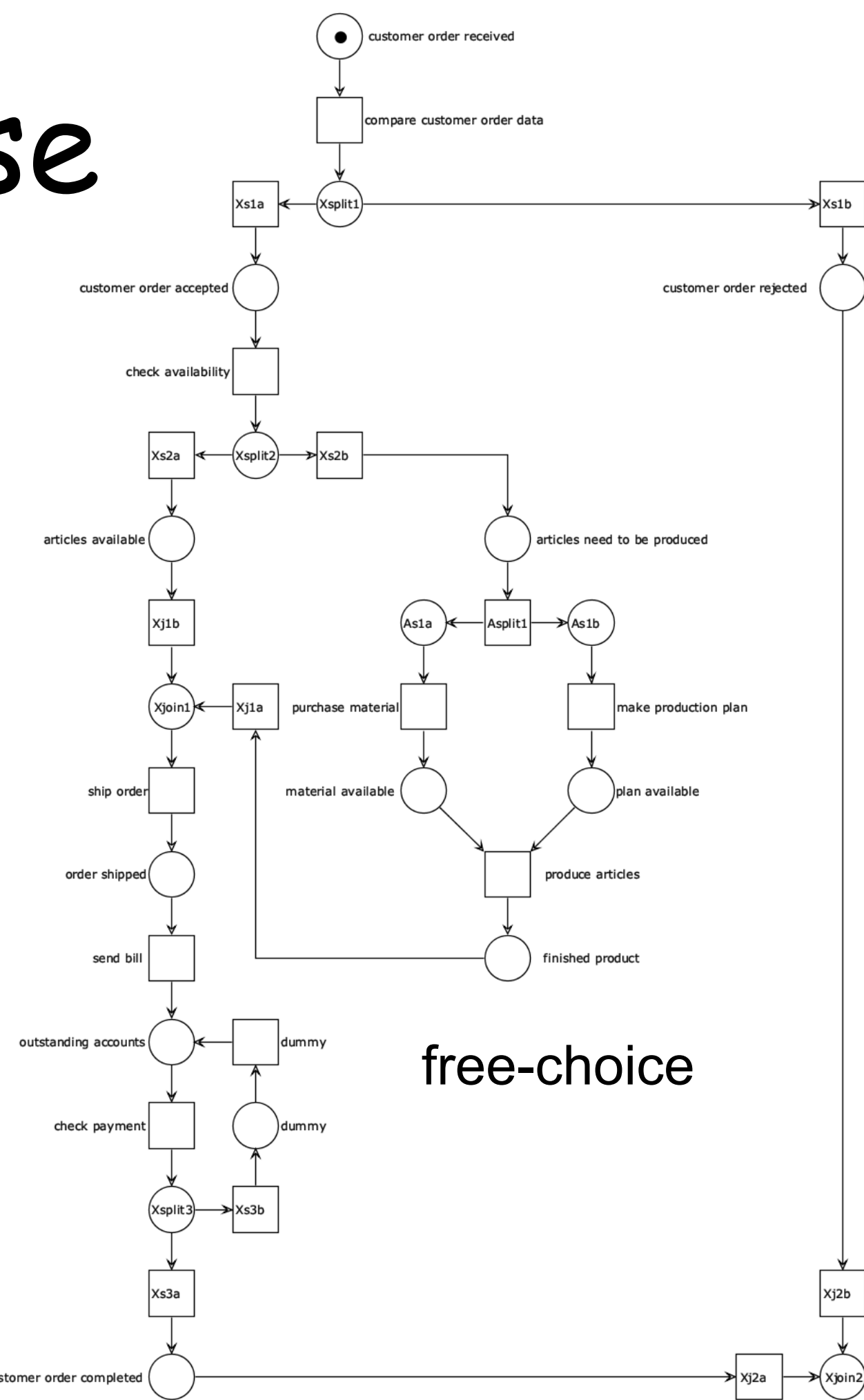
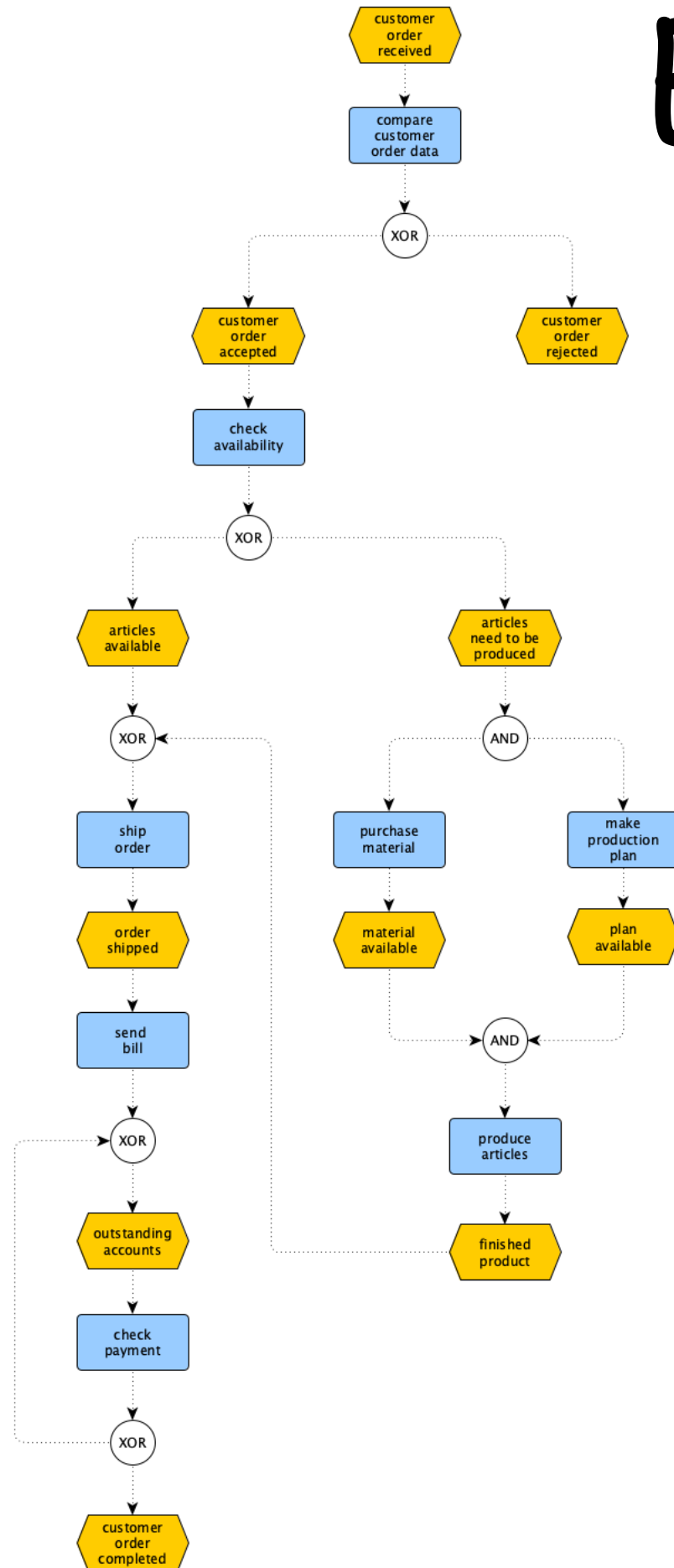


Exercise

2nd technique
(fusion style)



Steps
1+2+3
(XOR end)



Sound!



Sound!