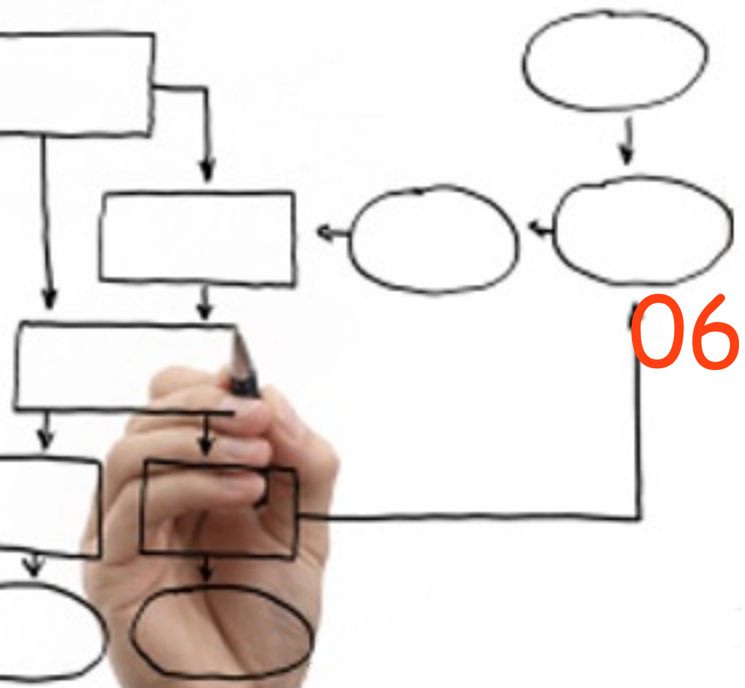
Business Processes Modelling

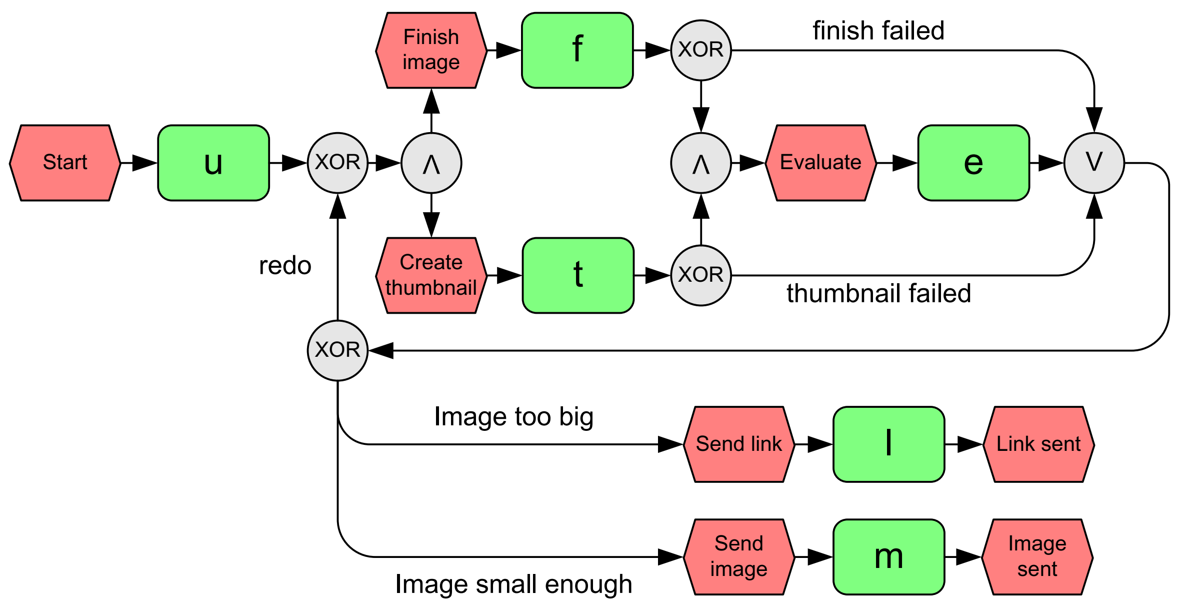
MPB (6 cfu, 295AA)

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



06 - Event-driven Process Chains

Object



We overview the EPC notation

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

Event-driven Process Chain

Definition: An **Event-driven process chain** (**EPC**) is an ordered graph of **events** and **functions**.
It provides various **connectors** that allow alternative and parallel execution of processes.
Furthermore it is specified by the usages of **logical operators**, such as OR, AND, and XOR.

A major strength of EPC is claimed to be its ***simplicity*** and ***easy-to-understand*** notation.
This makes EPC a widely acceptable technique to denote business processes.

EPC in a nutshell

Flow-chart language that can be used:
to configure an Enterprise Resource Planning implementation
to drive the modelling, analysis, redesign of business process

Informal notation (no "legenda" needed):
simple, minimal, intuitive and easy-to-understand

XML interchange format:
EPC Markup Language (.epml)

EPC origin (early 1990's)

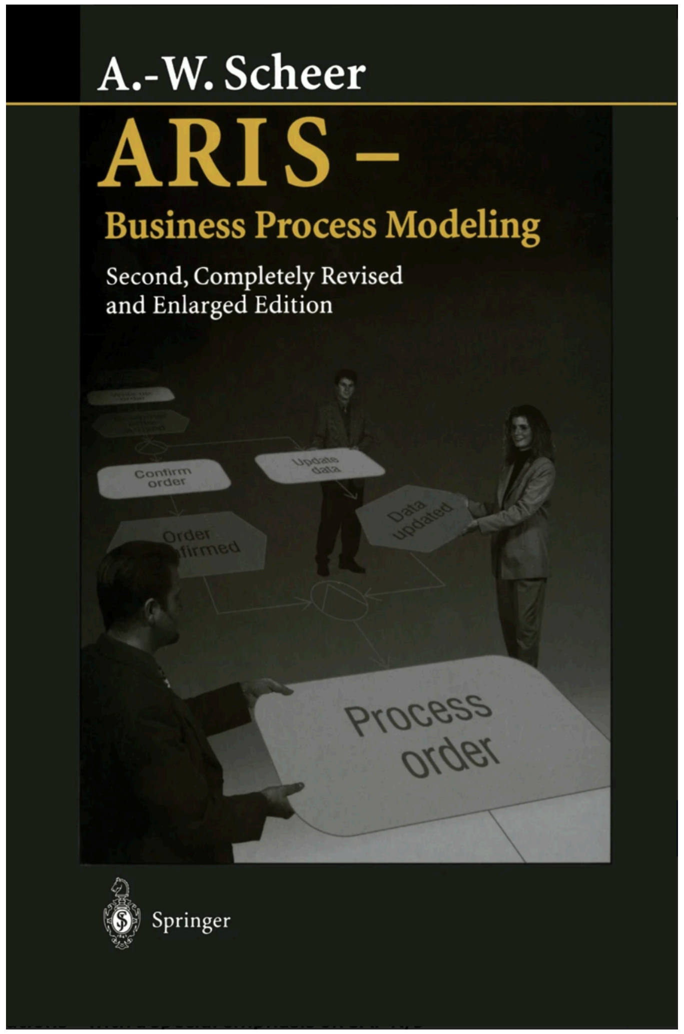
A.-W. Scheer

ARIS –

Vom Geschäftsprozess zum Anwendungssystem

Vierte, durchgesehene Auflage

Springer



A.-W. Scheer

ARIS –

Business Process Modeling

Second, Completely Revised and Enlarged Edition

Springer

EPC method originally developed as part of a holistic modelling approach called **ARIS framework** (Architecture of Integrated Information Systems) by Wilhelm-August Scheer

EPC Diagrams

Why do we need diagrams?

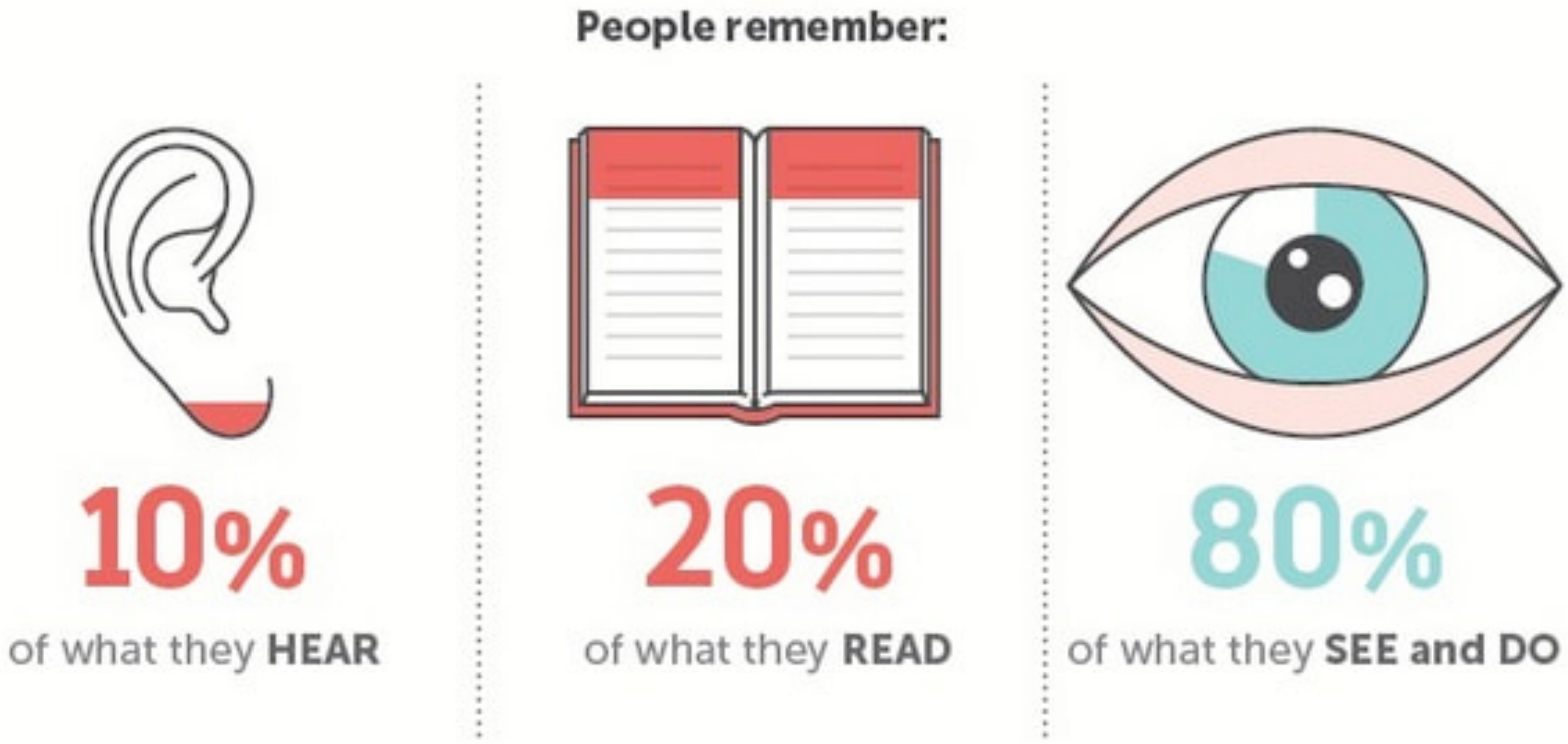
Graphical languages **communicate** concepts

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

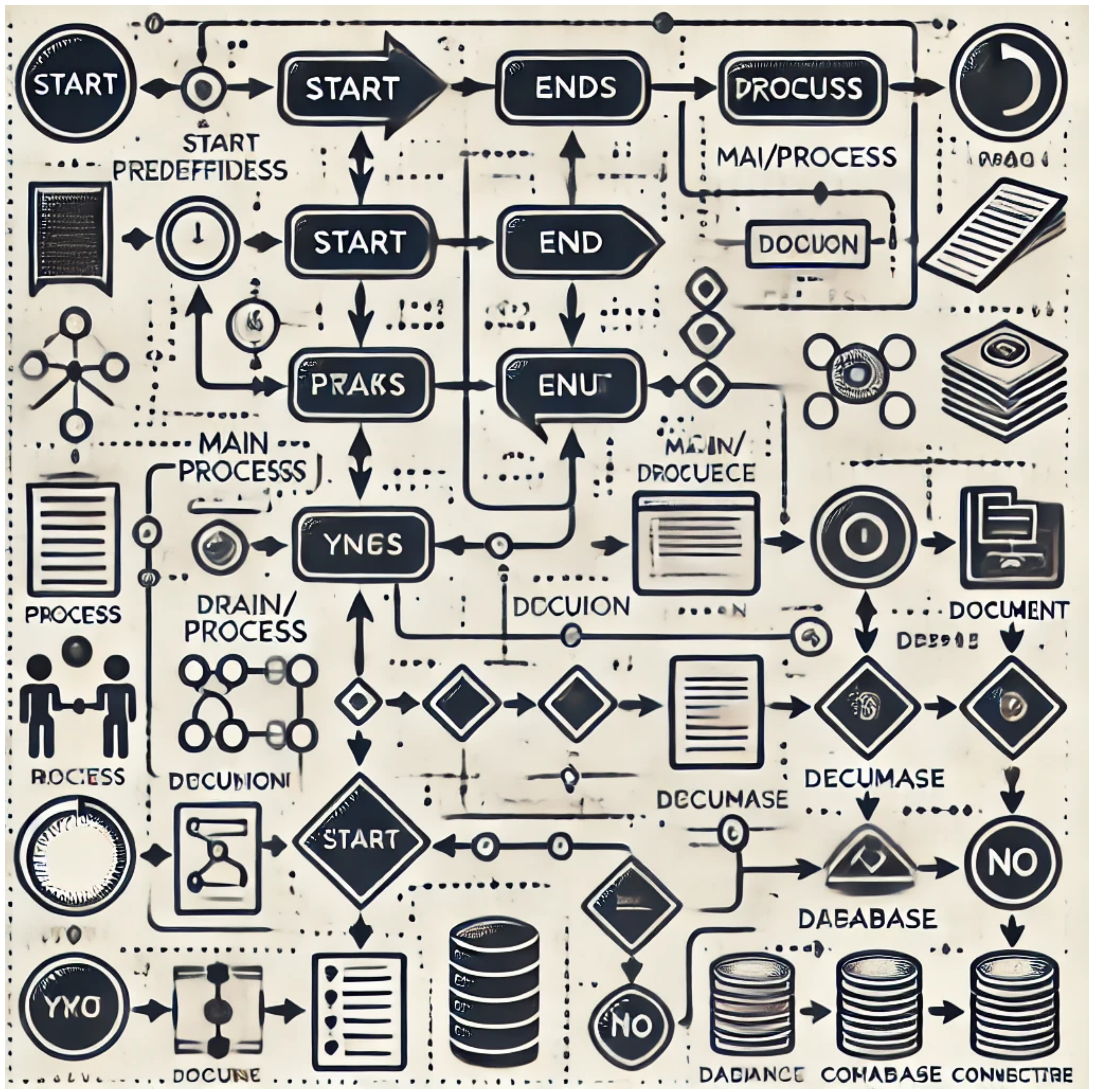
Greatest common denominator of the people involved

Intuitive meaning
(verbal description, no math involved)

Why do we need diagrams?



Keep it simple!



(OpenAI ChatGPT generated)

EPC informally

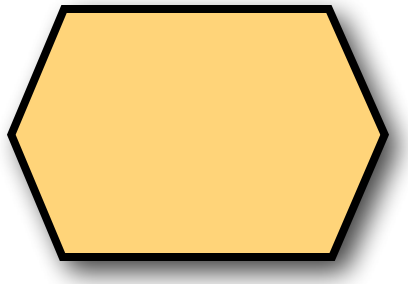
An EPC is a graph of **events** and **functions**

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

Events

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

Graphical representation: hexagons

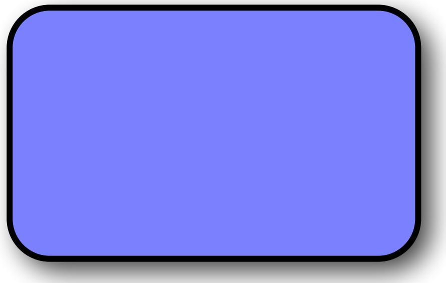


Passive elements used to describe
under which circumstances a process (or a function) works
or which state a process (or a function) results in
(like pre- / post-conditions)

Functions

Any EPC diagram may involve several **functions**

Graphical representation: rounded rectangles



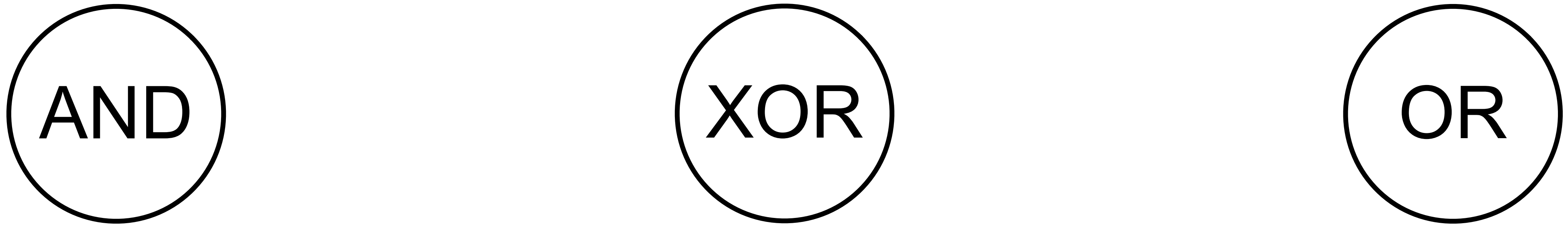
Active elements used to describe
the tasks or activities of a business process

Functions can be refined to other EPC diagrams

Logical connectors

Any EPC diagram may involve several **connectors**

Graphical representation: circles (or also octagons)



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

Logical connectors: logical symbols

$\wedge$ = AND

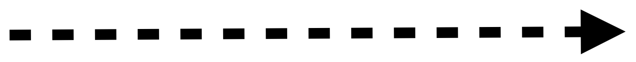
X = XOR

$\vee$ = OR

Control flow

Any EPC diagram may involve several **connections**

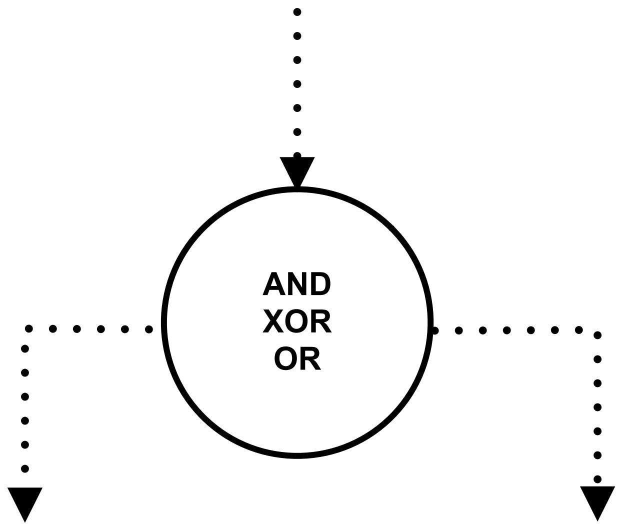
Graphical representation: dashed arrows



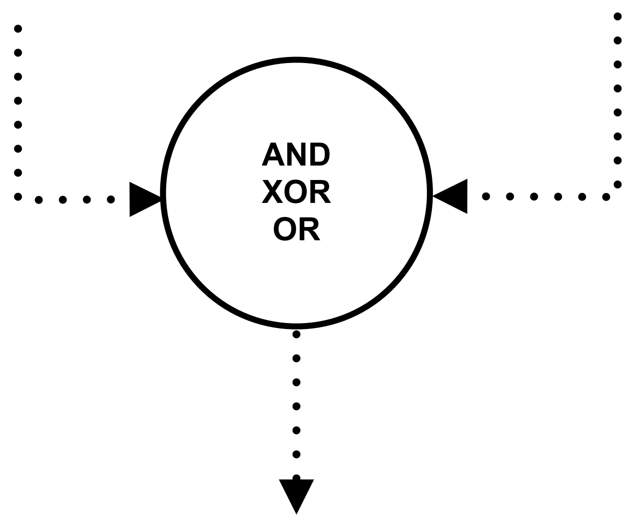
Control flow is used to connect
events with functions and connectors
by expressing causal dependencies

Logical connectors: splits and joins

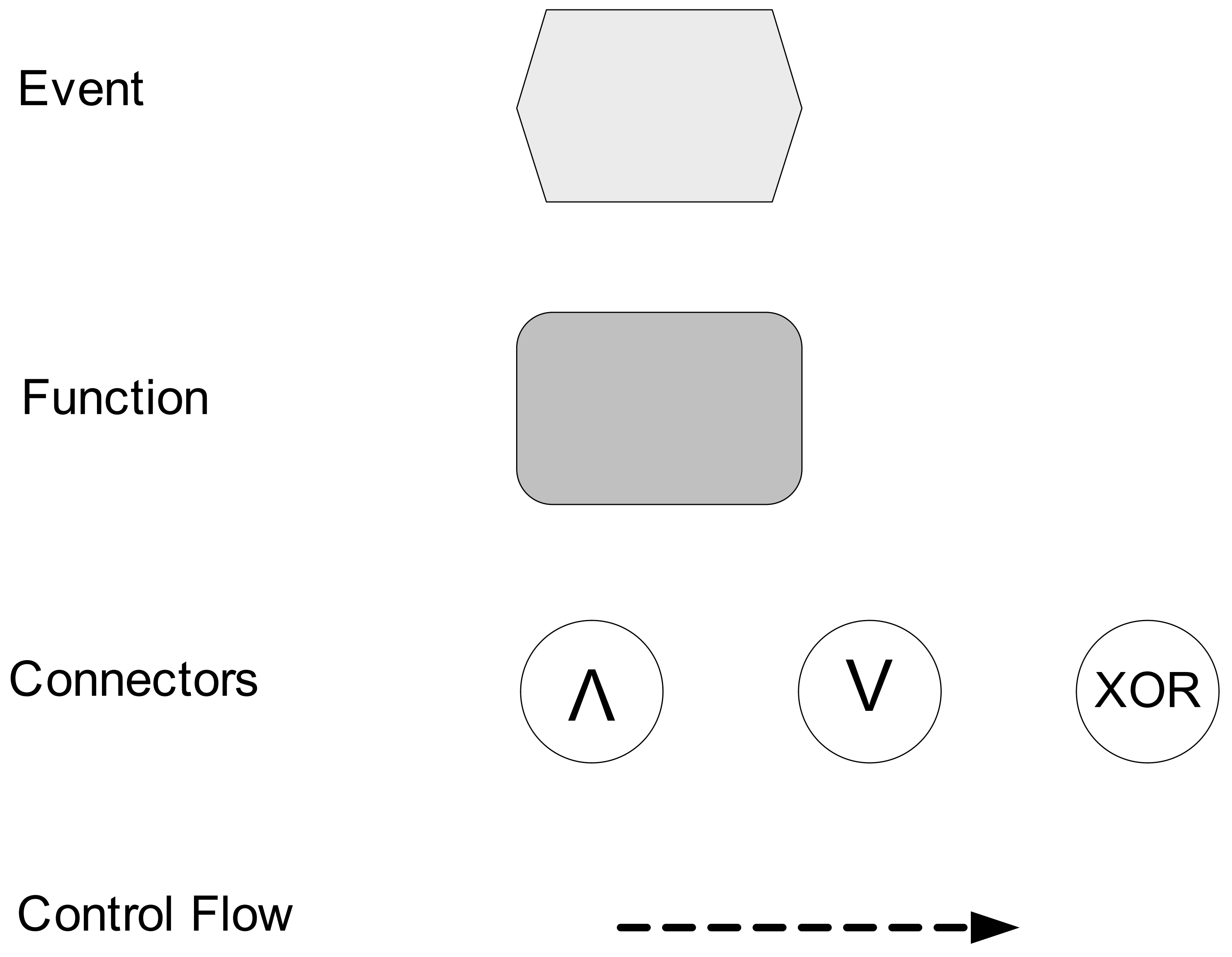
Splits



Joins



EPC ingredients at a glance



EPC Diagrams: Requirements

EPC diagrams: requirements

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

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

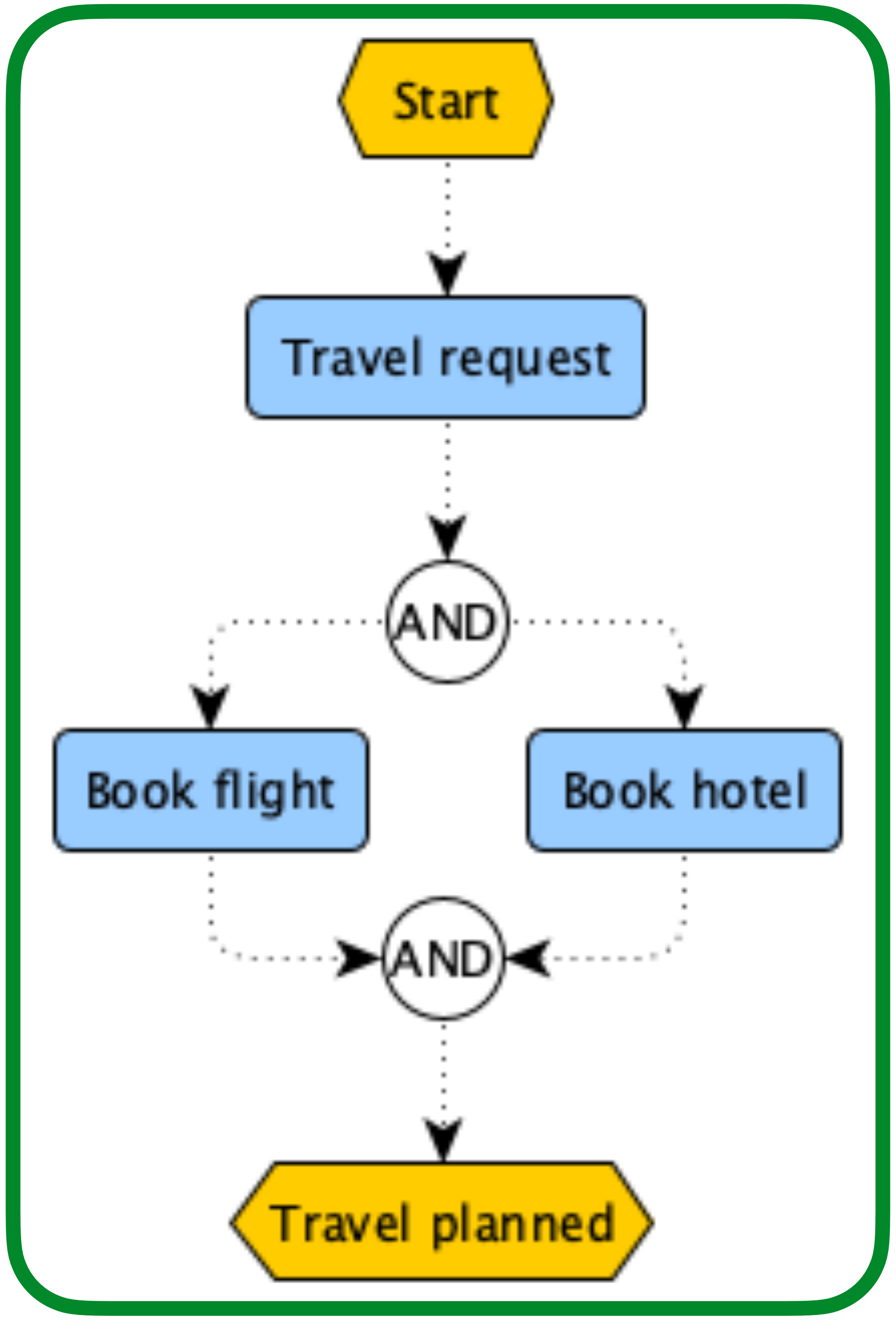
Events have at most one incoming and one outgoing arc
Events have at least one incident arc
There must be at least one start event and one end event

Functions have exactly one incoming and one outgoing arc

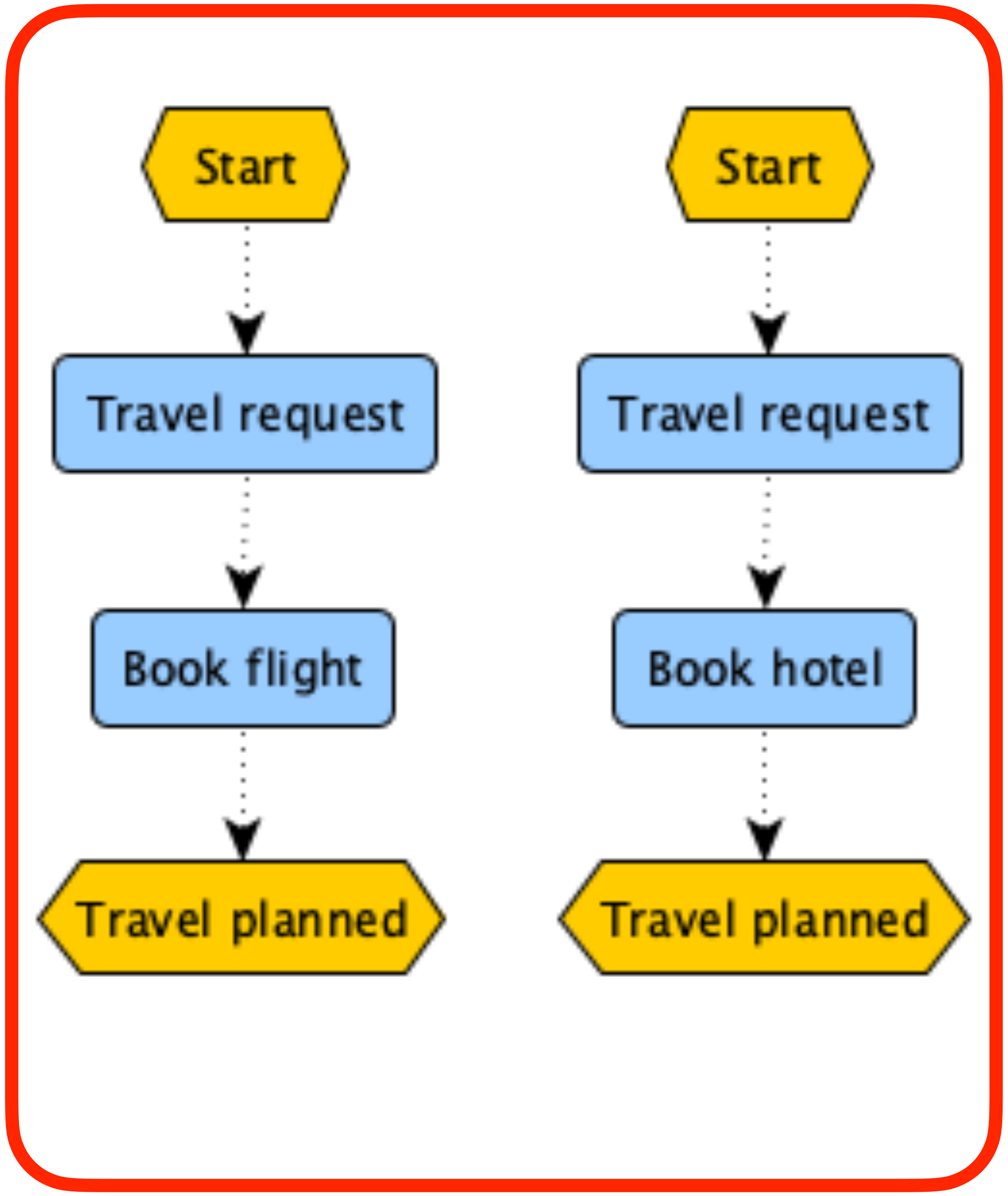
Connectors have either one incoming arc and multiple outgoing arcs
or viceversa (multiple incoming arcs and one outgoing arc)

Weak connectivity

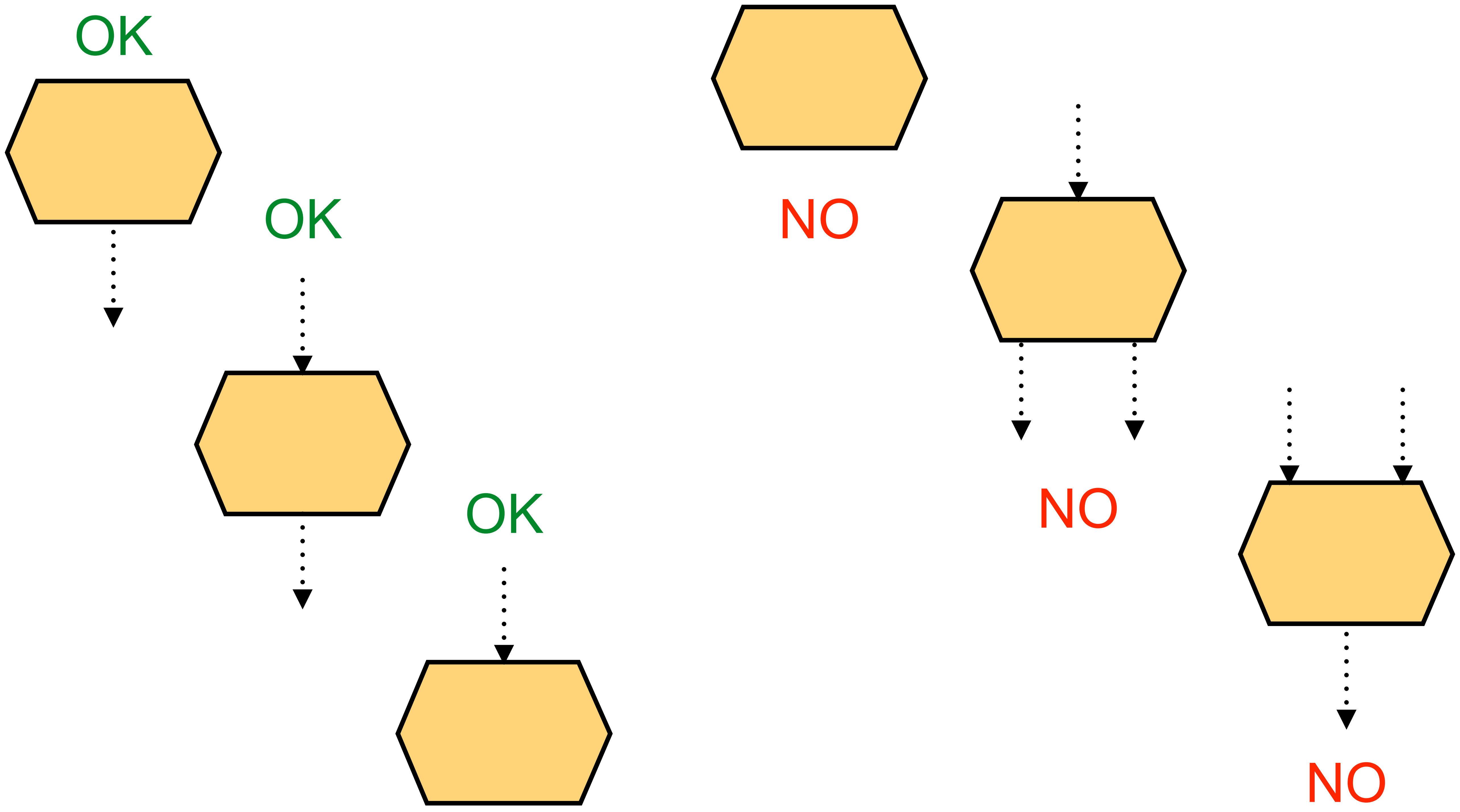
Weakly connected



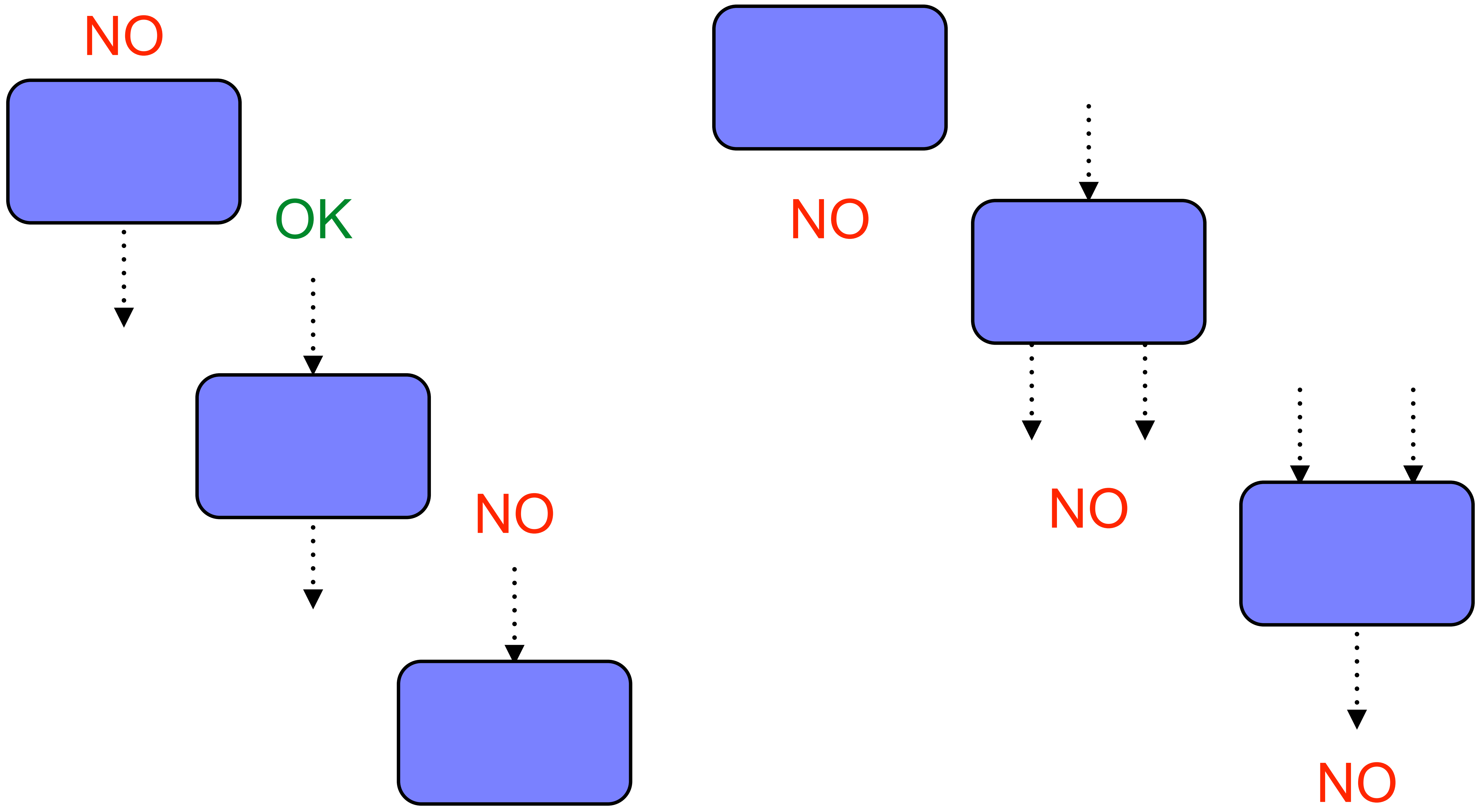
Non weakly connected



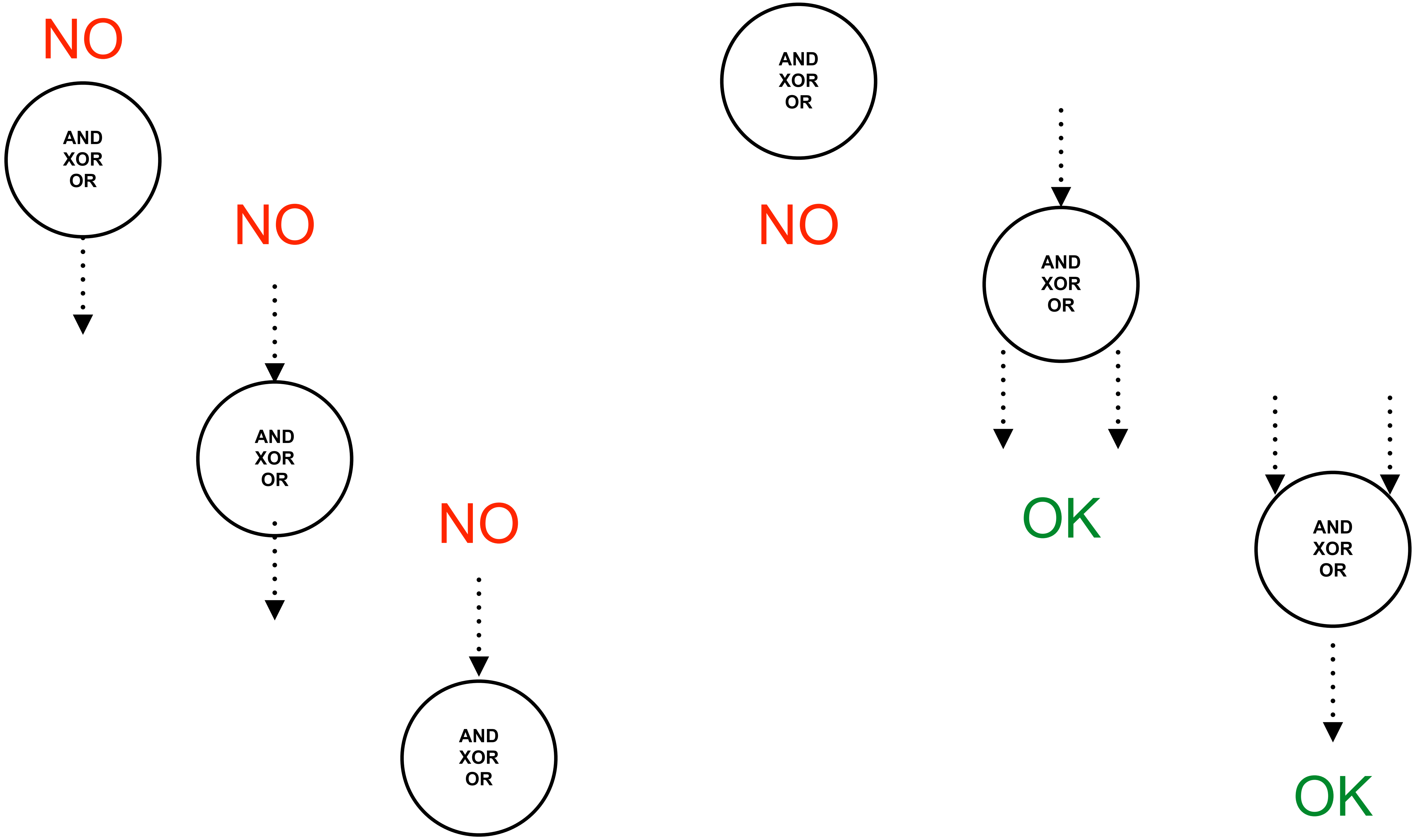
Event connectivity



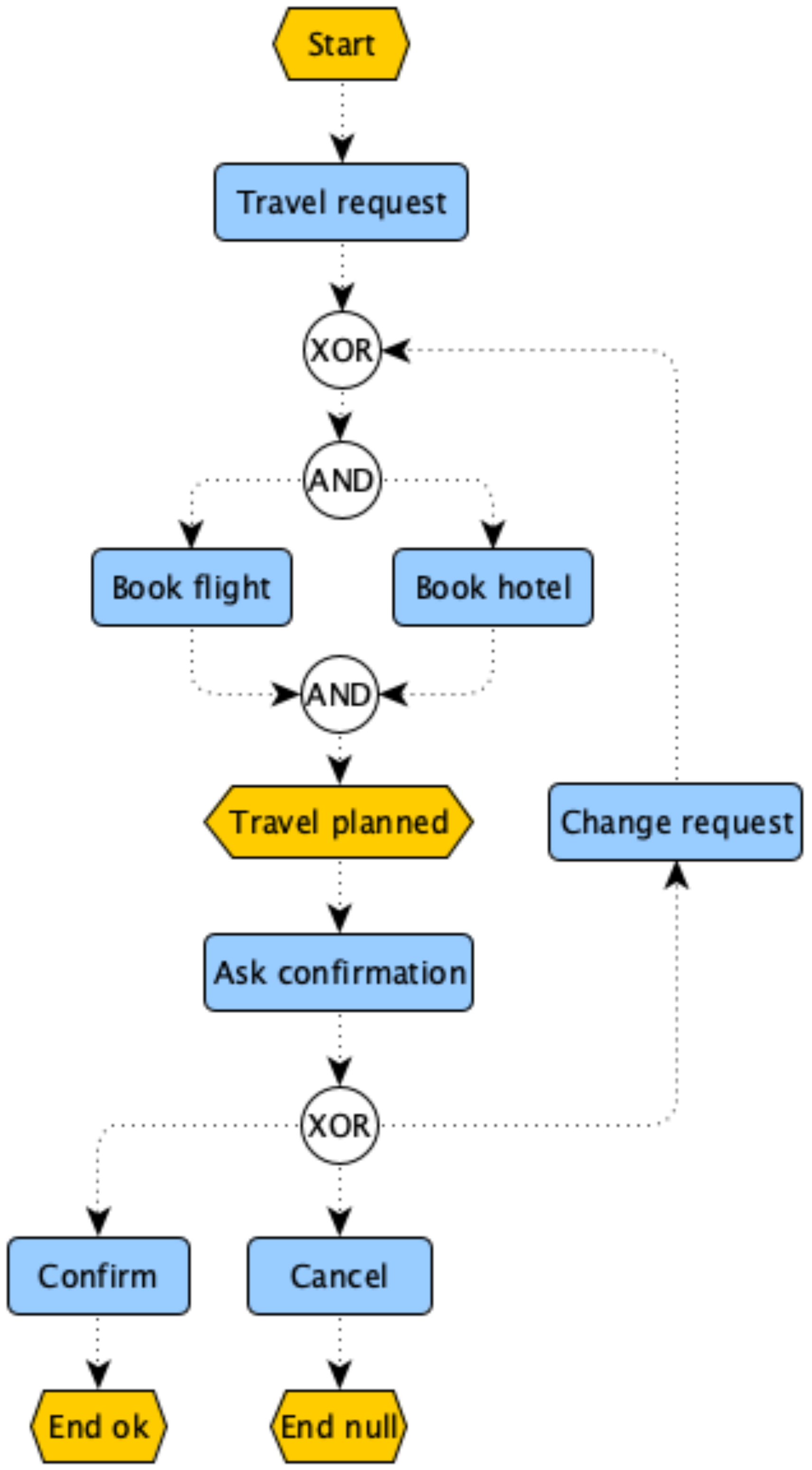
Function connectivity



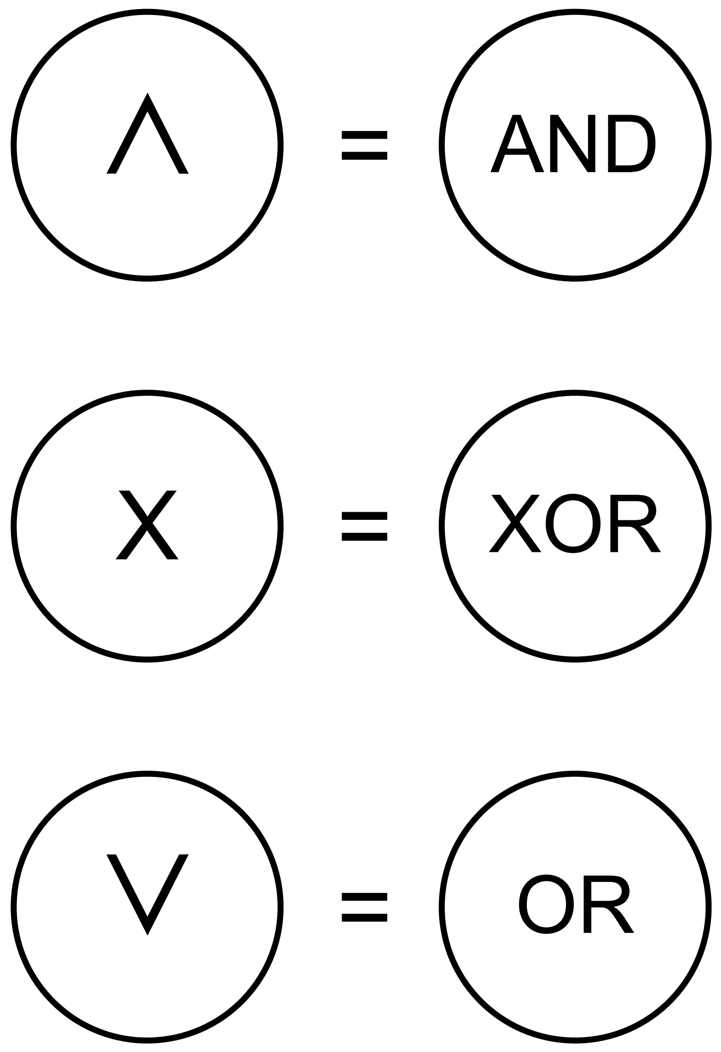
Split/Join connectivity

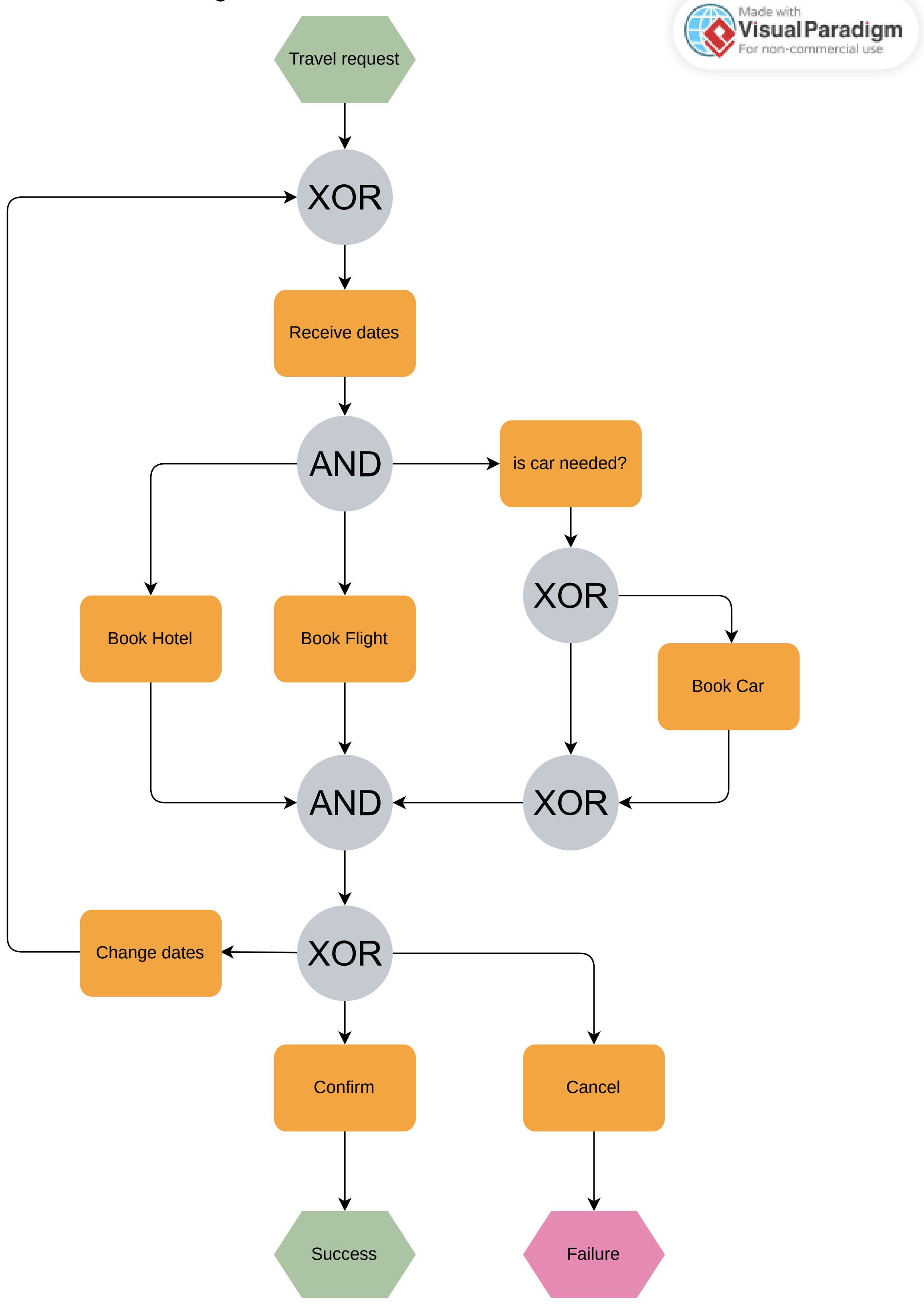


EPC: Example (yEd)

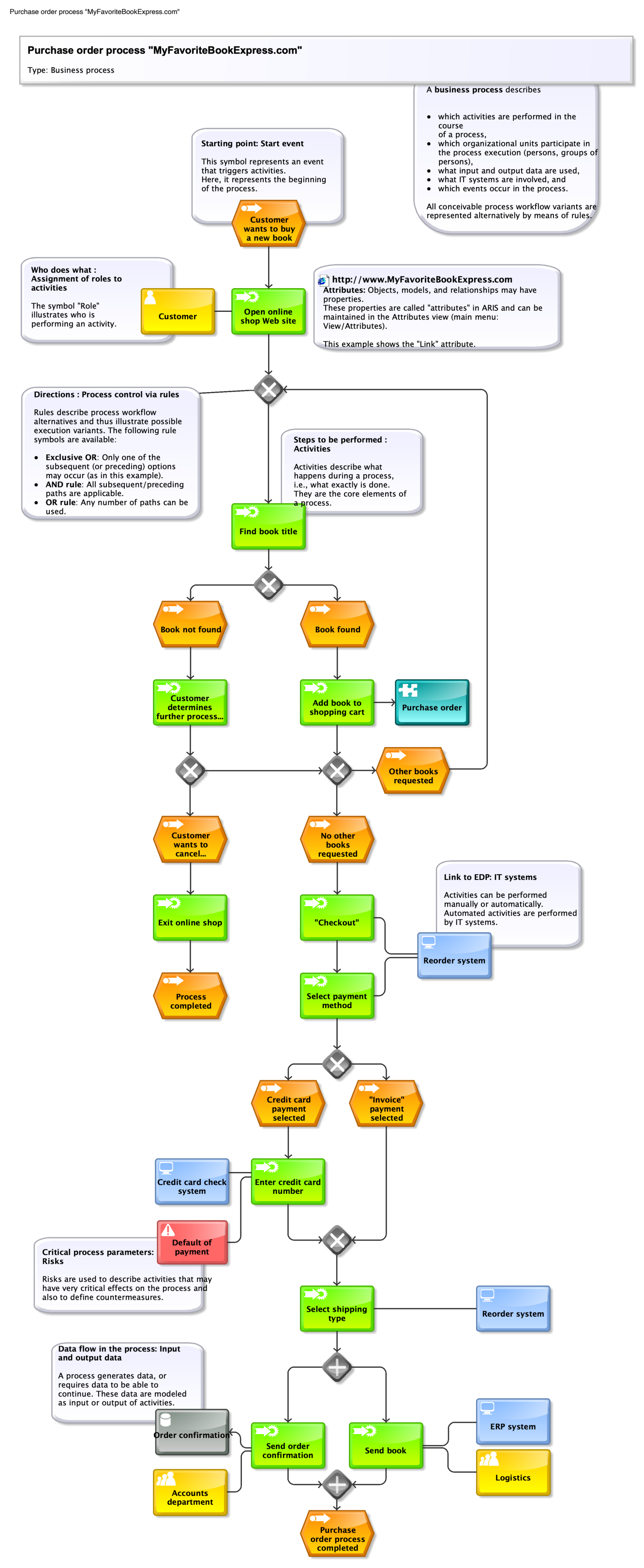


EPC: Example (VP online)





EPC: Example (ARIS Express)


Purchase order process "MyFavoriteBookExpress.com"

Purchase order process "MyFavoriteBookExpress.com"

Type: Business process

A **business process** describes

- which activities are performed in the course of a process,
- which organizational units participate in the process execution (persons, groups of persons),
- what input and output data are used,
- what IT systems are involved, and
- which events occur in the process.

All conceivable process workflow variants are represented alternatively by means of rules.

Starting point: Start event

This symbol represents an event that triggers activities.
Here, it represents the beginning of the process.

Who does what : Assignment of roles to activities

The symbol "Role" illustrates who is performing an activity.

http://www.MyFavoriteBookExpress.com
Attributes: Objects, models, and relationships may have properties.
These properties are called "attributes" in ARIS and can be maintained in the Attributes view (main menu: View/Attributes).

This example shows the "Link" attribute.

Directions : Process control via rules

Rules describe process workflow alternatives and thus illustrate possible execution variants. The following rule symbols are available:

- **Exclusive OR**: Only one of the subsequent (or preceding) options may occur (as in this example).
- **AND rule**: All subsequent/preceding paths are applicable.
- **OR rule**: Any number of paths can be used.

Steps to be performed : Activities

Activities describe what happens during a process, i.e., what exactly is done.
They are the core elements of a process.

Link to EDP: IT systems

Activities can be performed manually or automatically.
Automated activities are performed by IT systems.

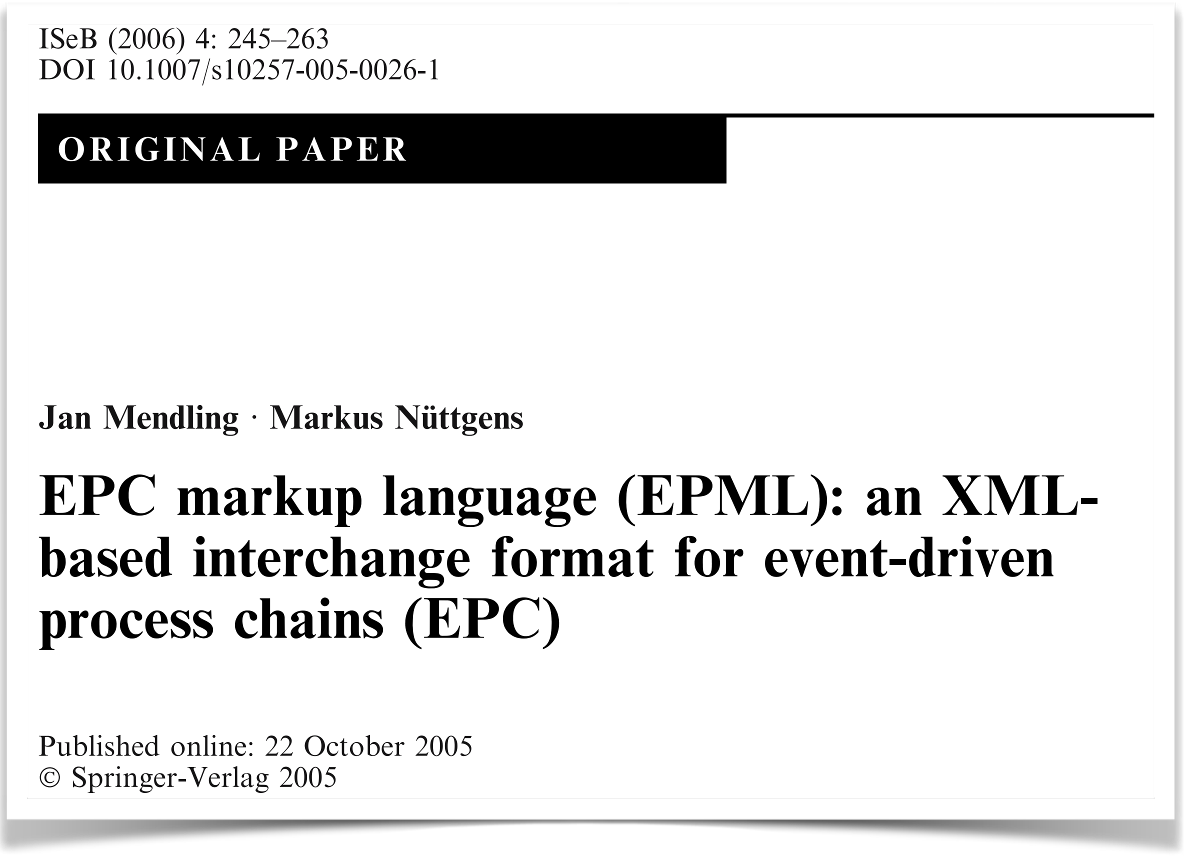
Critical process parameters: Risks

Risks are used to describe activities that may have very critical effects on the process and also to define countermeasures.

Data flow in the process: Input and output data

A process generates data, or requires data to be able to continue. These data are modeled as input or output of activities.

A taste of EPML

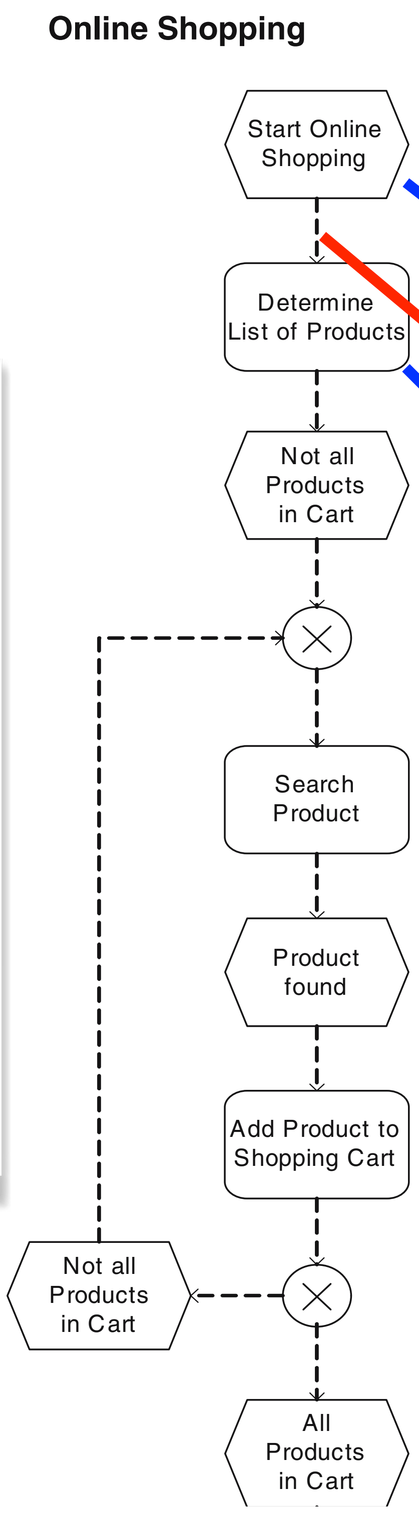

ISeB (2006) 4: 245–263
DOI 10.1007/s10257-005-0026-1

ORIGINAL PAPER

Jan Mendling · Markus Nüttgens

EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC)

Published online: 22 October 2005
© Springer-Verlag 2005



```
<?xml version="1.0" encoding="UTF-8"?>
<epml:epml xmlns:epml="http://www.epml.de">

  <coordinates
    xOrigin="leftToRight"
    yOrigin="topToBottom"/>

  <directory name="Root">

    <epc epcId="1"
        name="Online Shopping">

      <event id="1">
        <name>Start Online Shopping</name>
      </event>

      <arc id="10">
        <flow source="1" target="2"/>
      </arc>

      <function id="2">
        <name>Determine List of Products</name>
      </function>

      <arc id="11">
        <flow source="2" target="3"/>
      </arc>

      <event id="3">
        <name>Not all Products in Cart</name>
      </event>

      <arc id="12">
        <flow source="3" target="4"/>
      </arc>

      <xor id="4"/>

      <arc id="13">
        <flow source="4" target="5"/>
      </arc>

      ...

    </epc>

  </directory>

</epml:epml>
```

EPC Diagrams: Guidelines

EPC Diagrams: guidelines

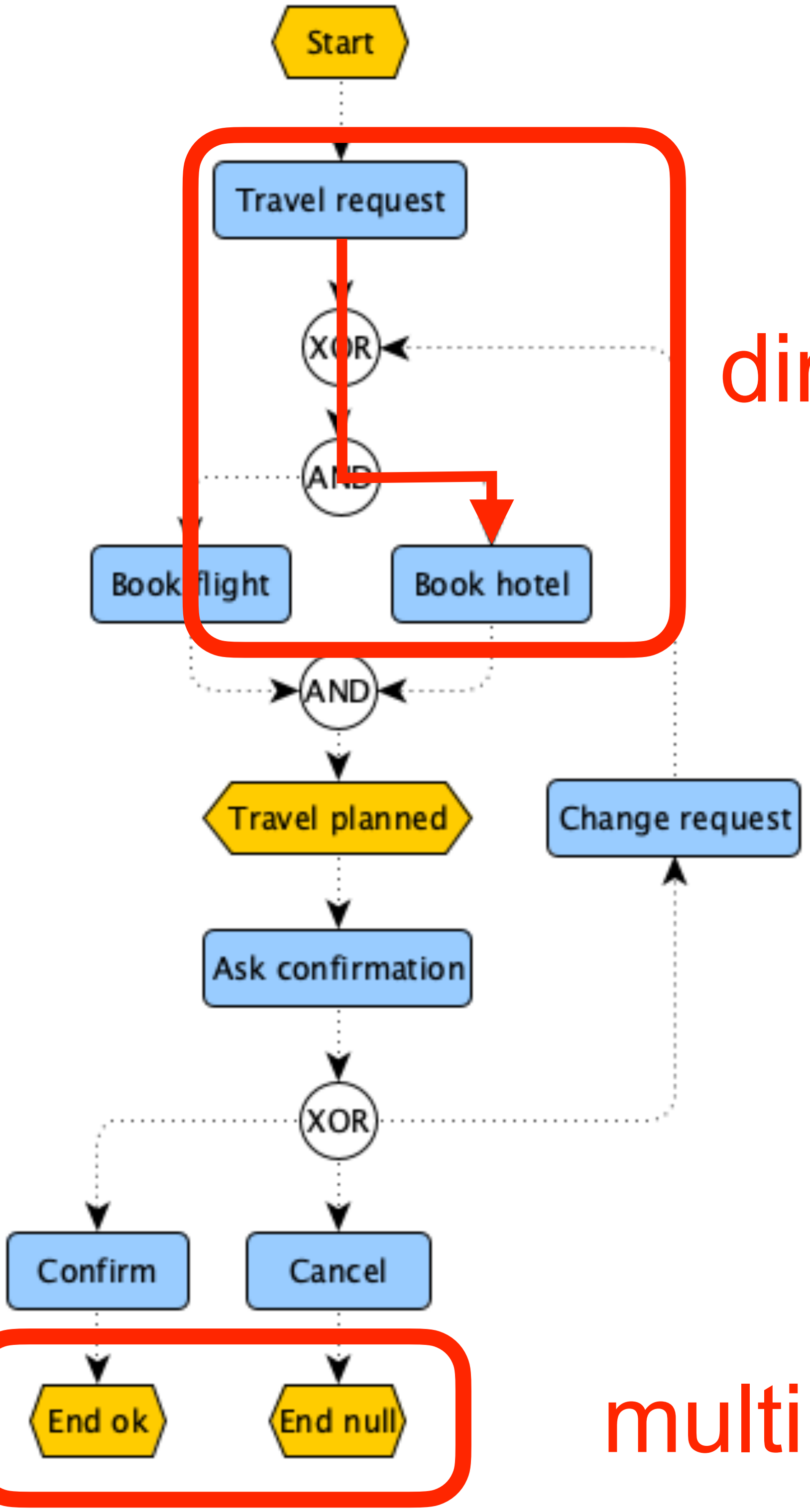
Other constraints are sometimes imposed

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

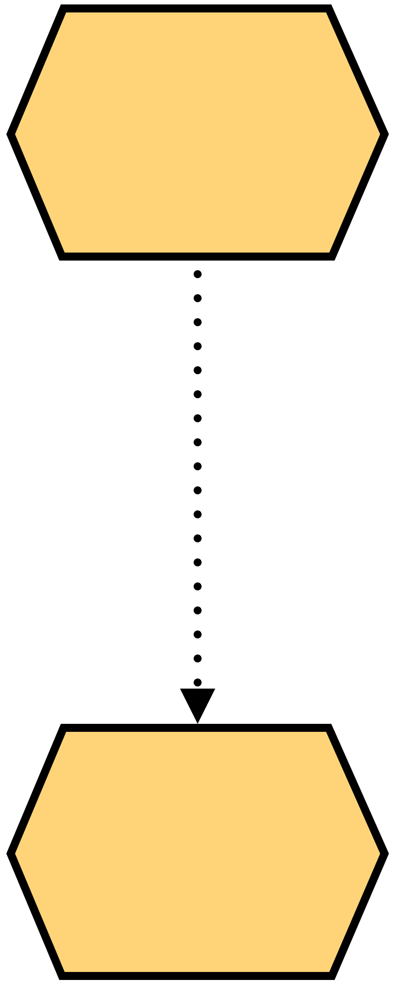
multiple end events

Problem with guidelines

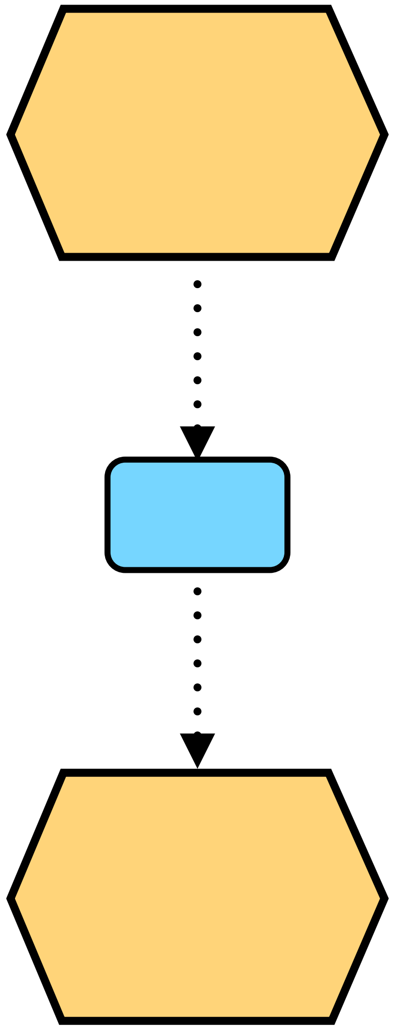
From empirical studies:
guidelines are too restrictive and people ignore them
(otherwise diagrams would get unnecessarily complicated,
more difficult to read and understand)

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

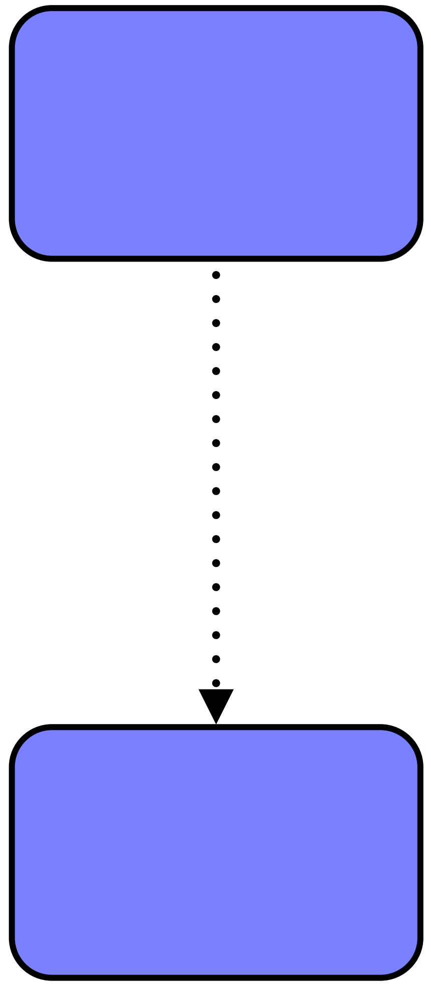
EPC: repairing alternation



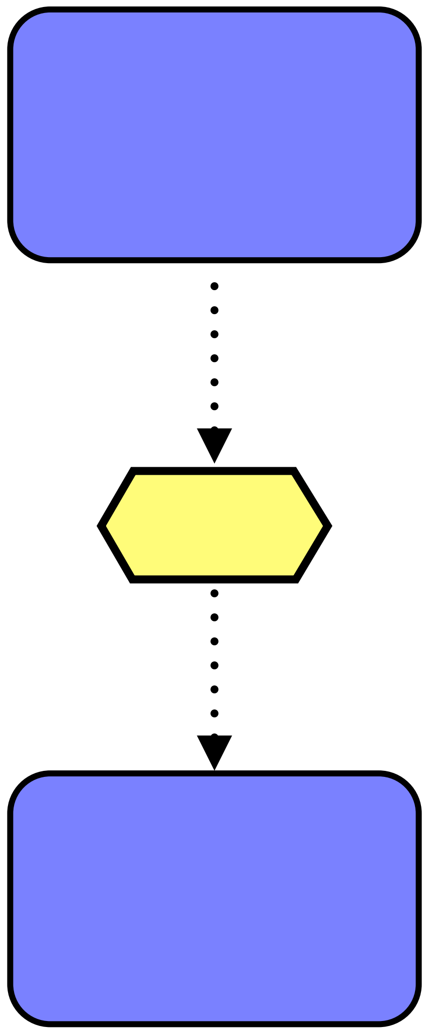
add dummy
functions
to guarantee
alternation



EPC: repairing alternation

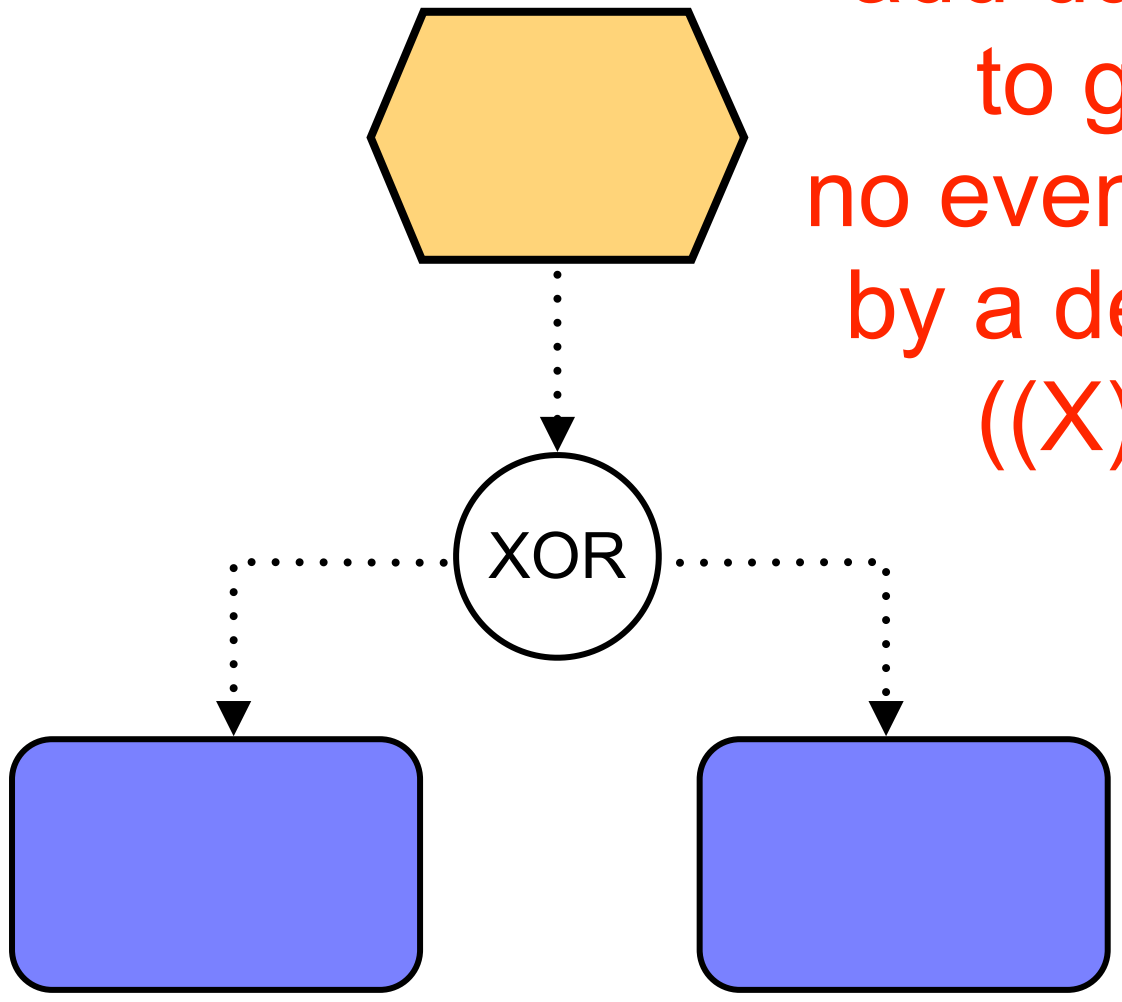


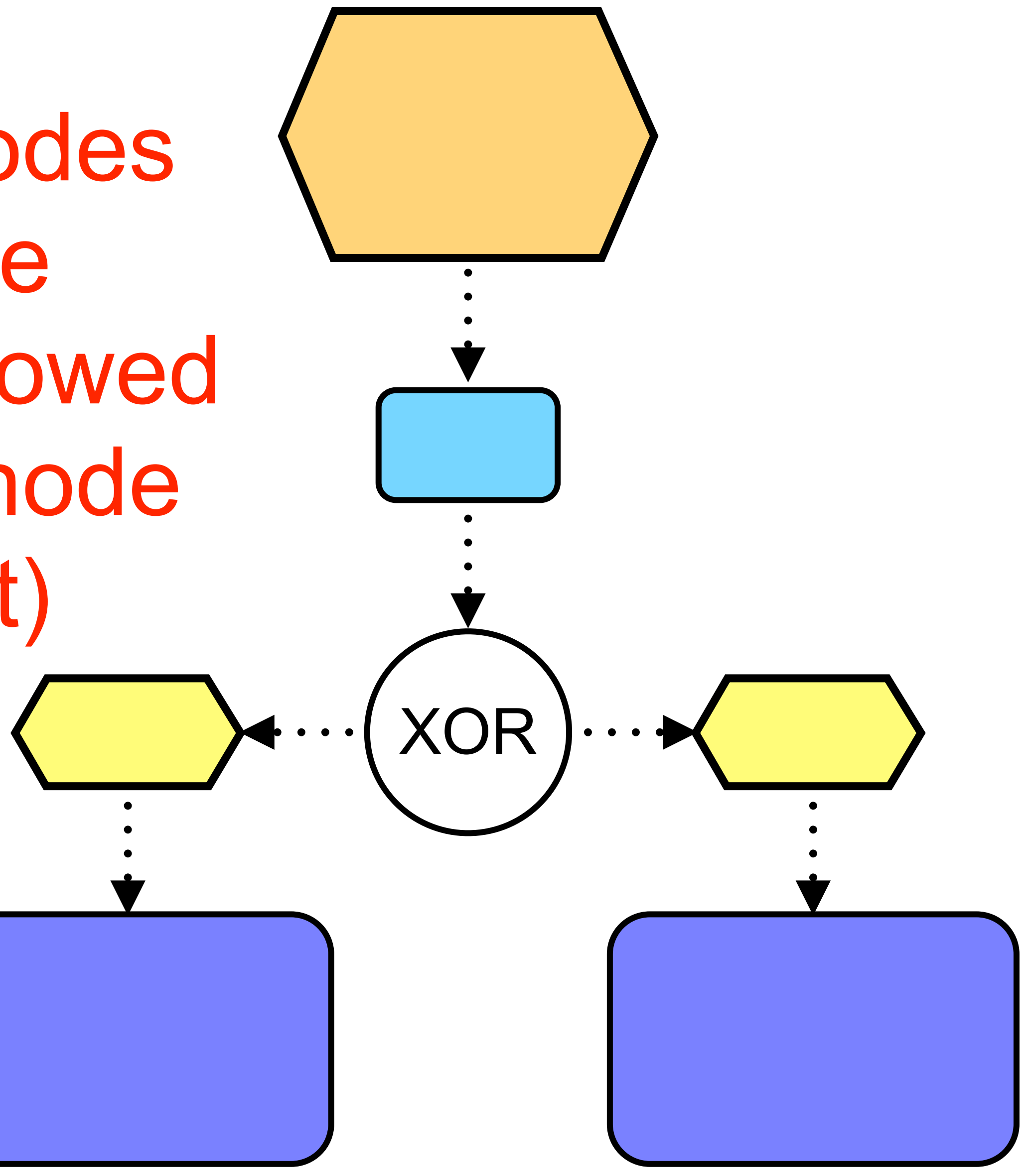
add dummy
events
to guarantee
alternation



EPC: repairing decisions

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

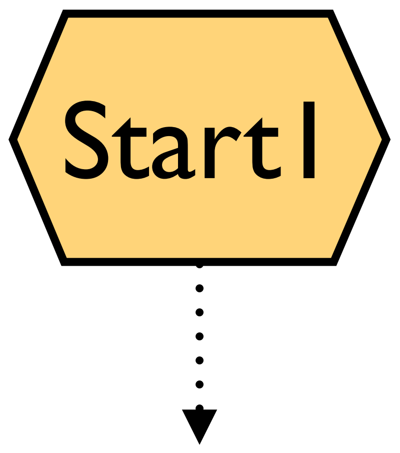




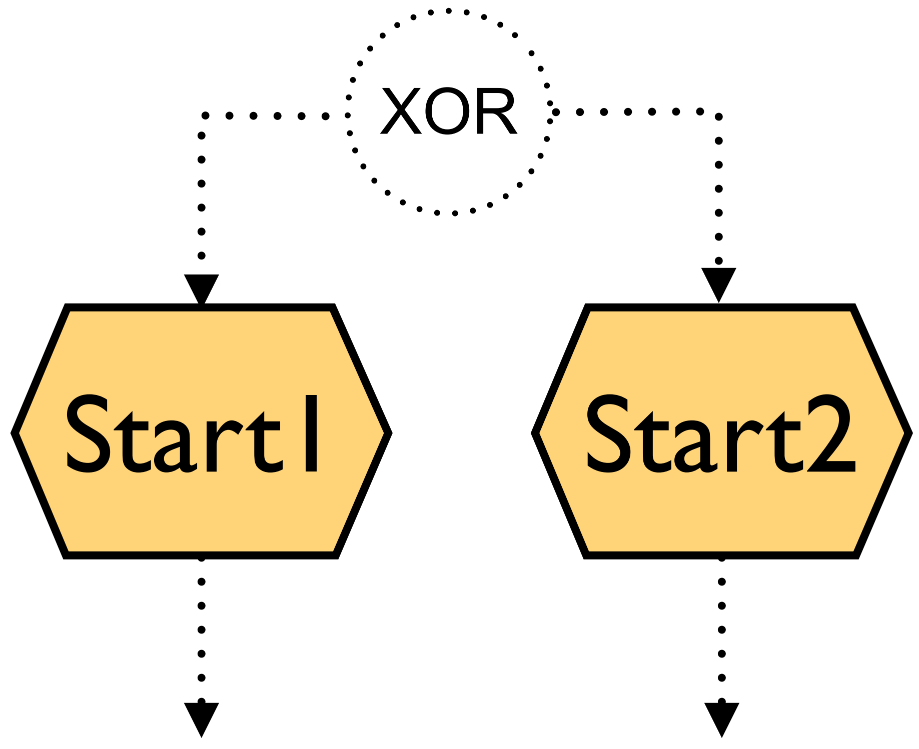
EPC: repairing multiple start events

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

Start events are mutually exclusive

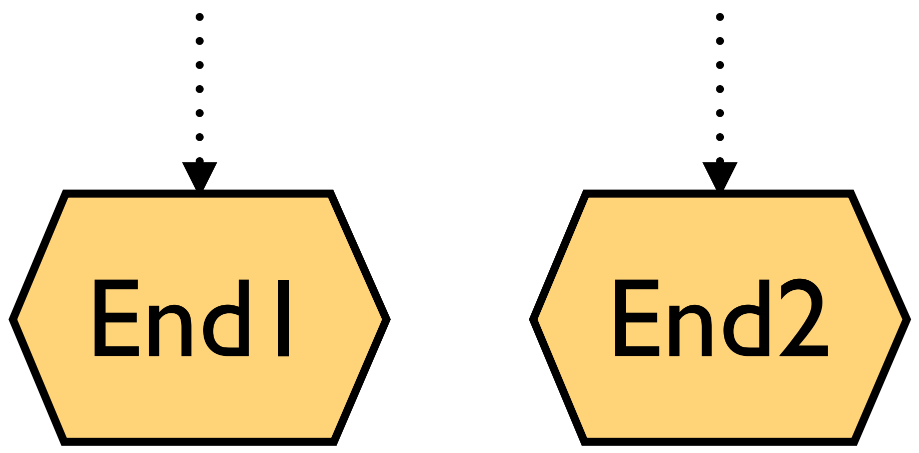


assume an implicit XOR split is present

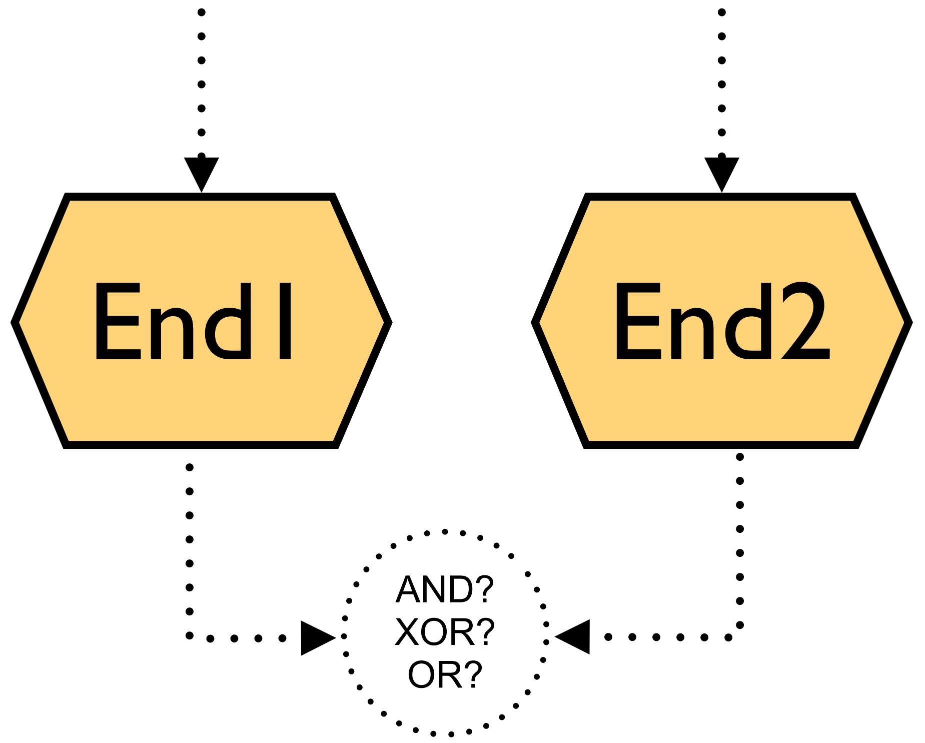


EPC: repairing multiple end events

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)

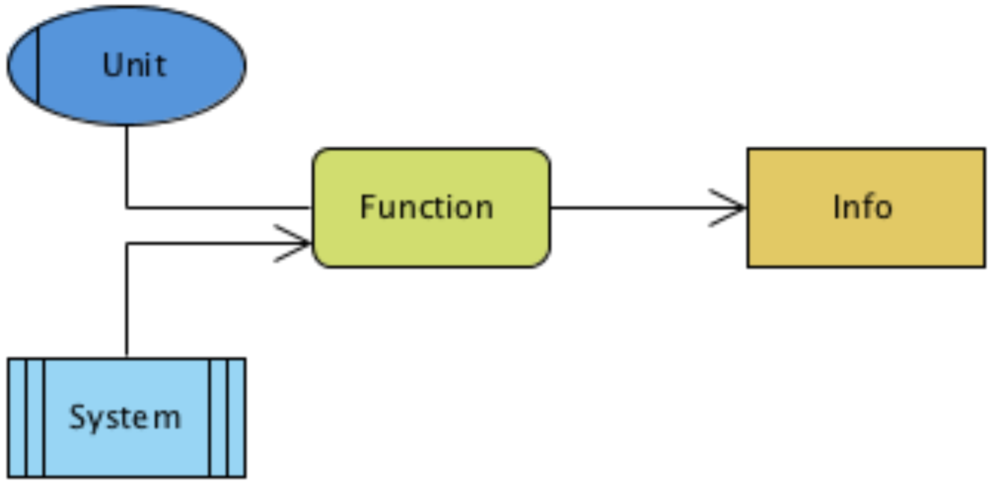


assume an implicit join is present



Other ingredients: function annotations

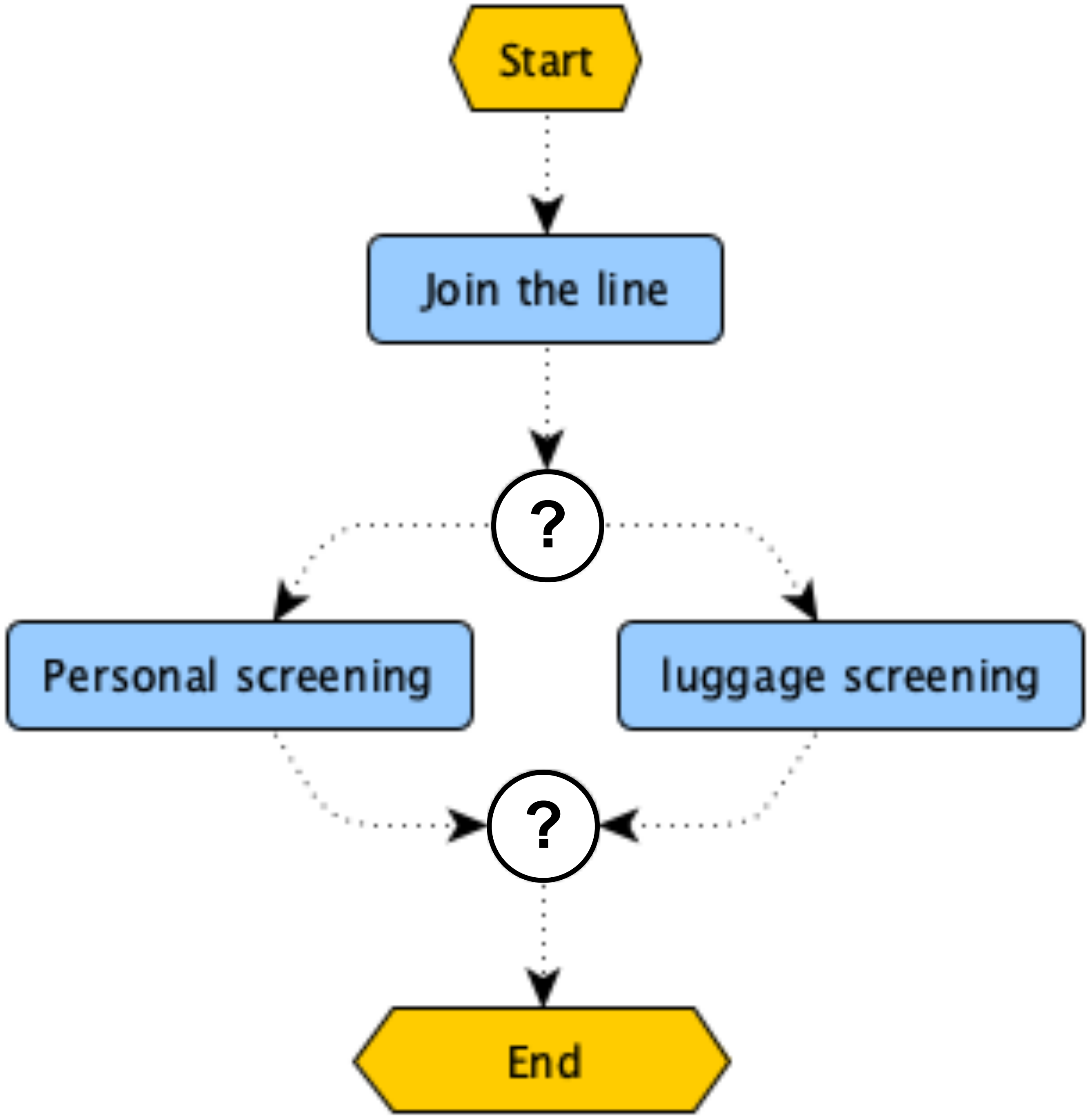
Organization unit:
determines the person or organization responsible for a specific function
(ellipses with a vertical line)



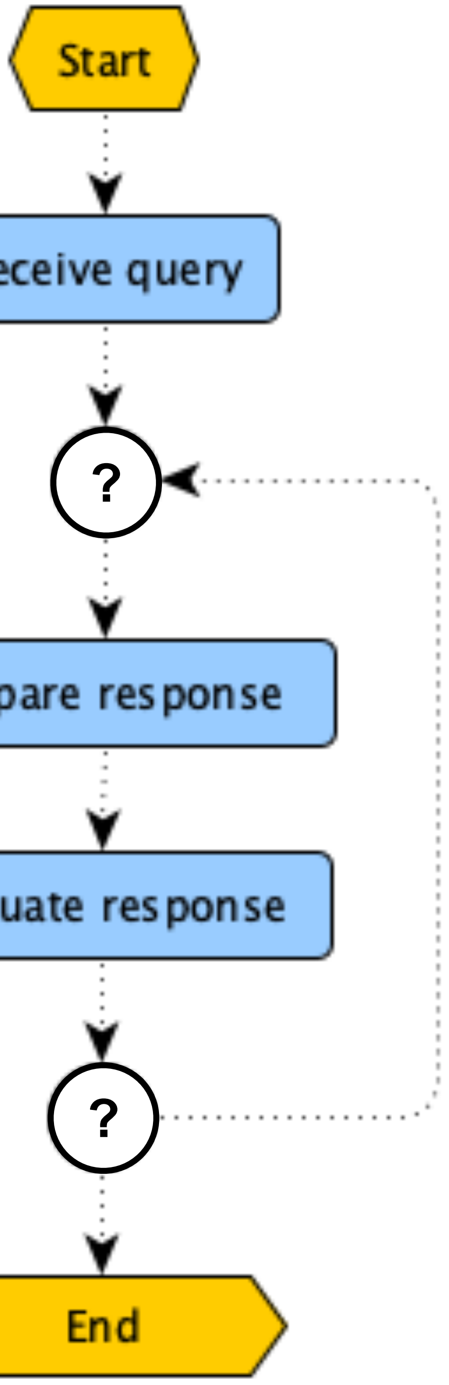
Information, material, resource object:
represents objects in the real world
e.g. input data or output data for a function
(rectangles linked to function boxes)
angles with vertical lines on its sides)

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

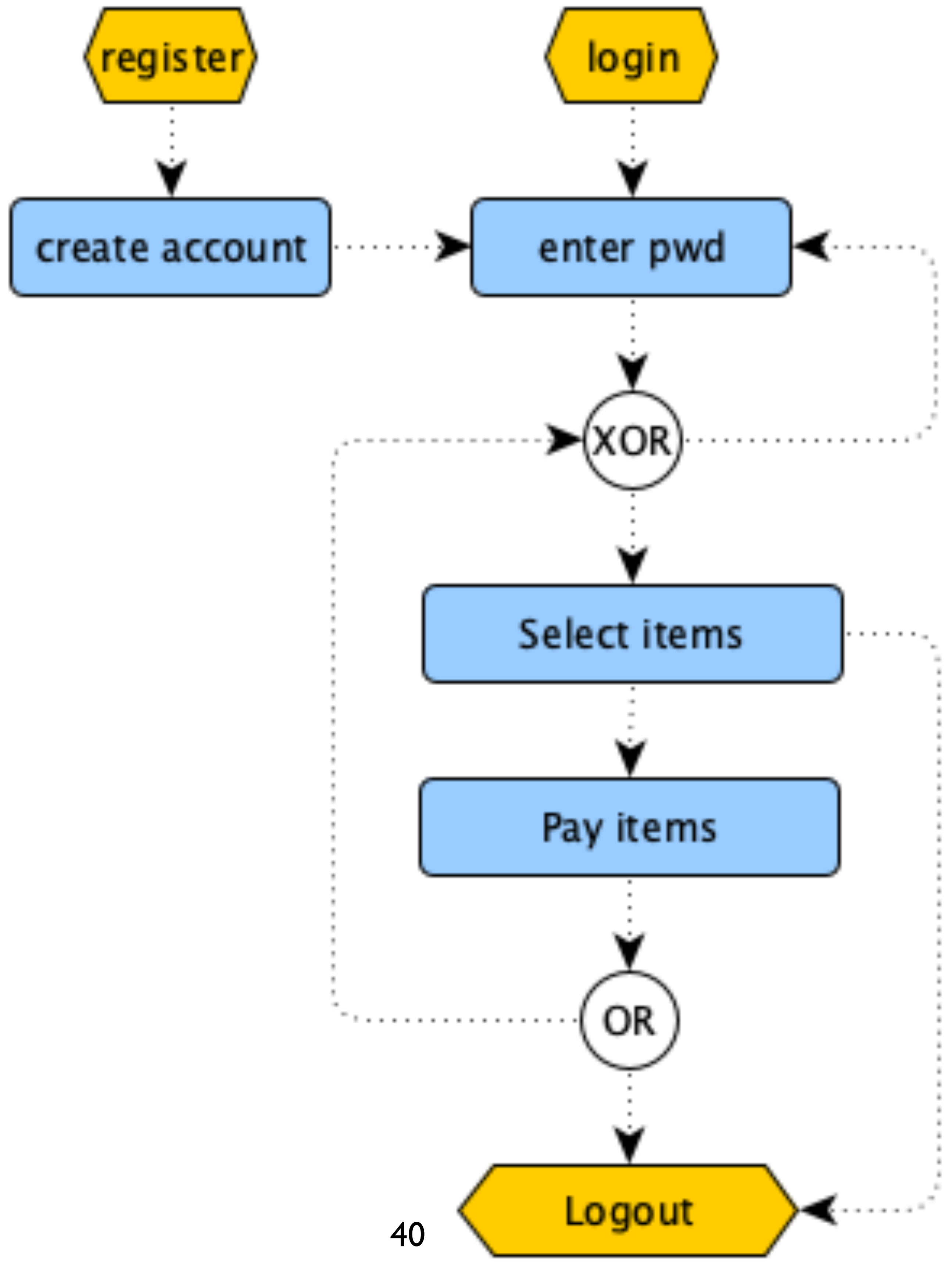
Question time: which connectors?



Question time: which connectors?



Question time: what's wrong?



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”

Folder-passing semantics

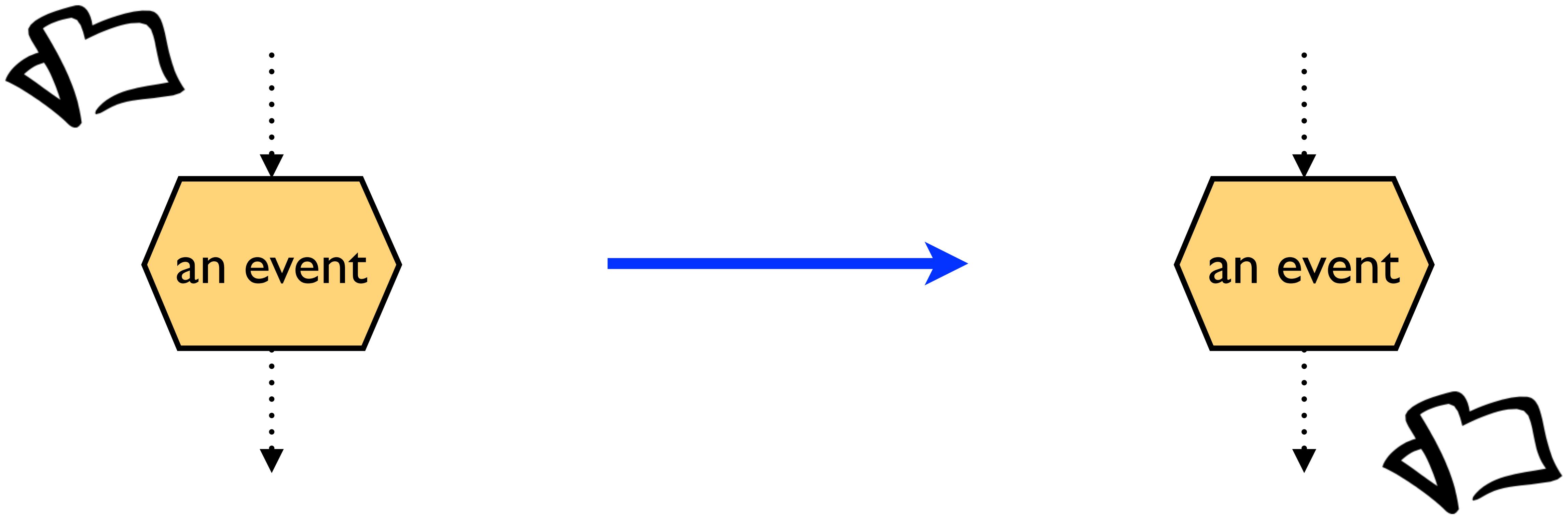
The current state of the process is determined by placing folders over the diagram

A transition relation explains how to move from one state to the next state

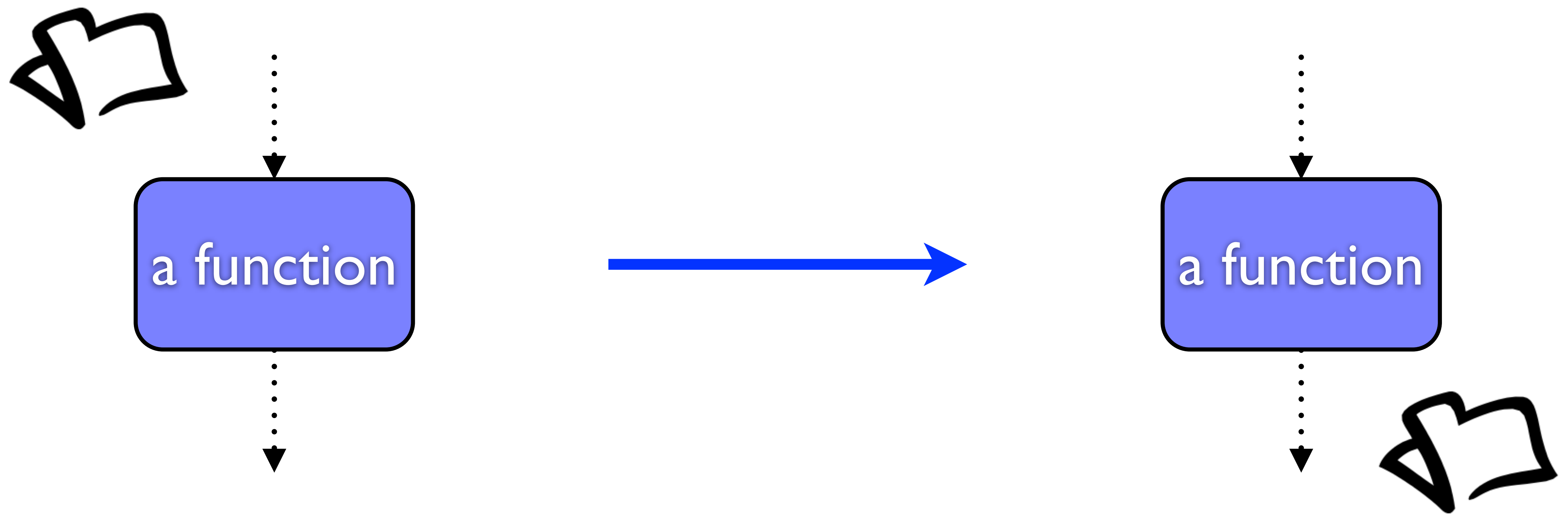
$$\longrightarrow$$

The transition relation is possibly nondeterministic

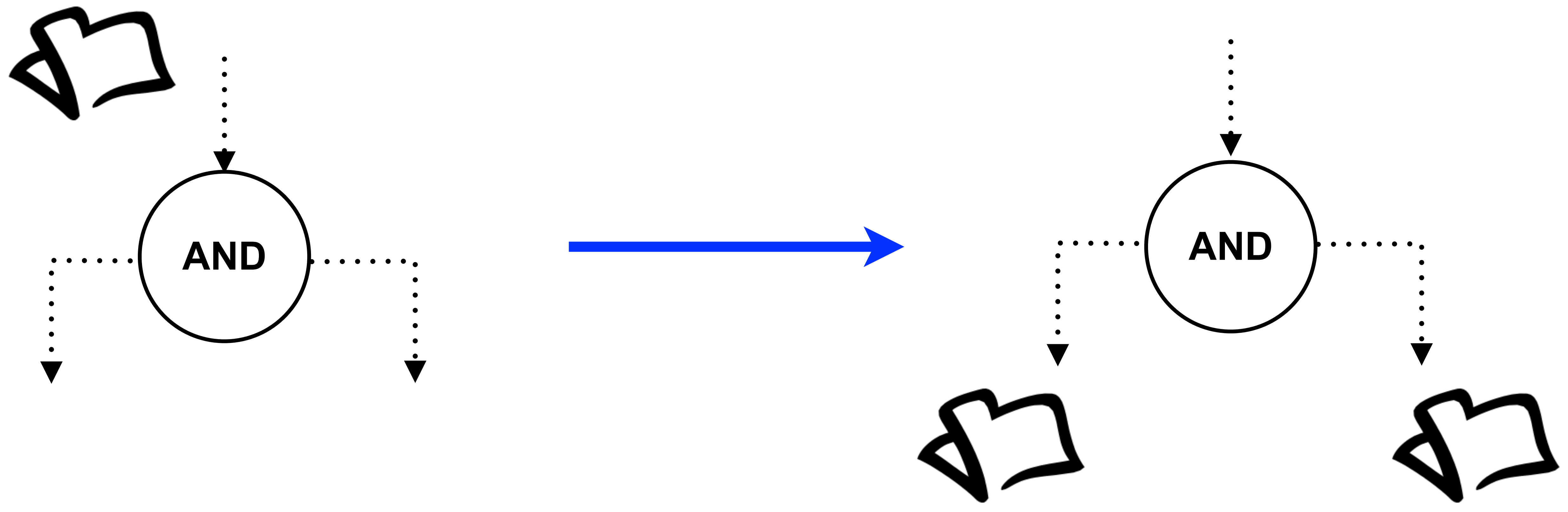
Folder-passing semantics: events



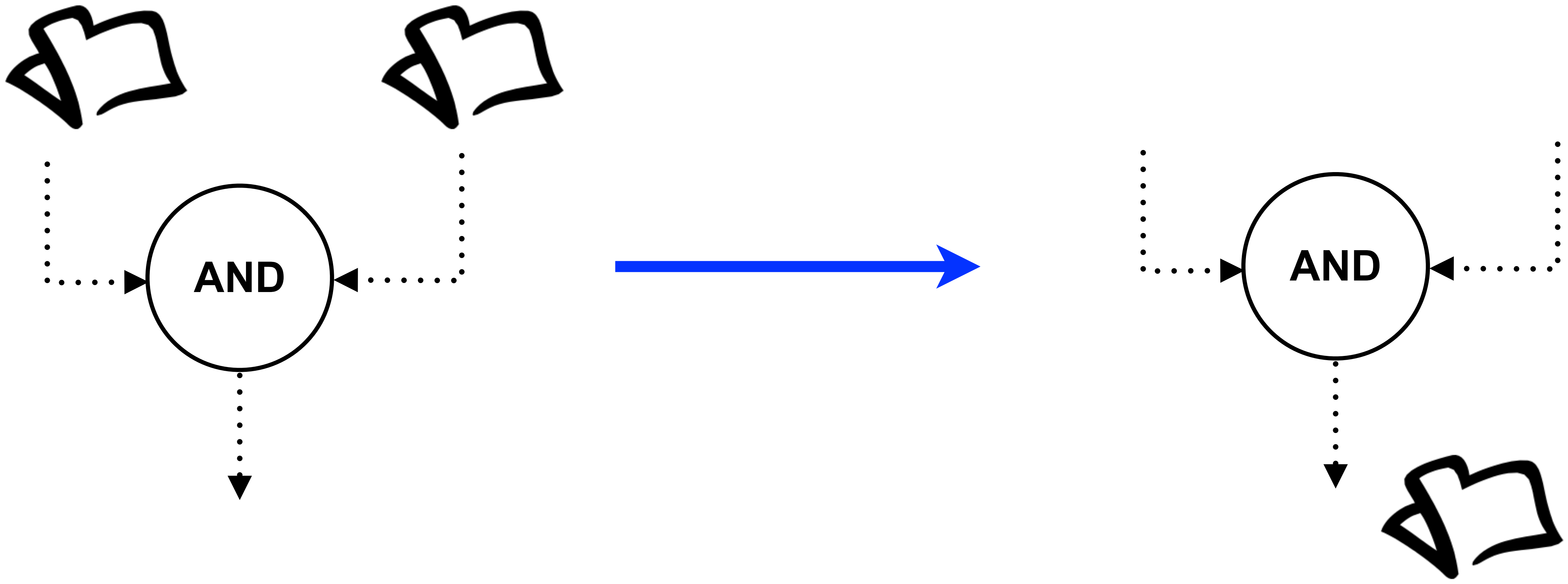
Folder-passing semantics: functions



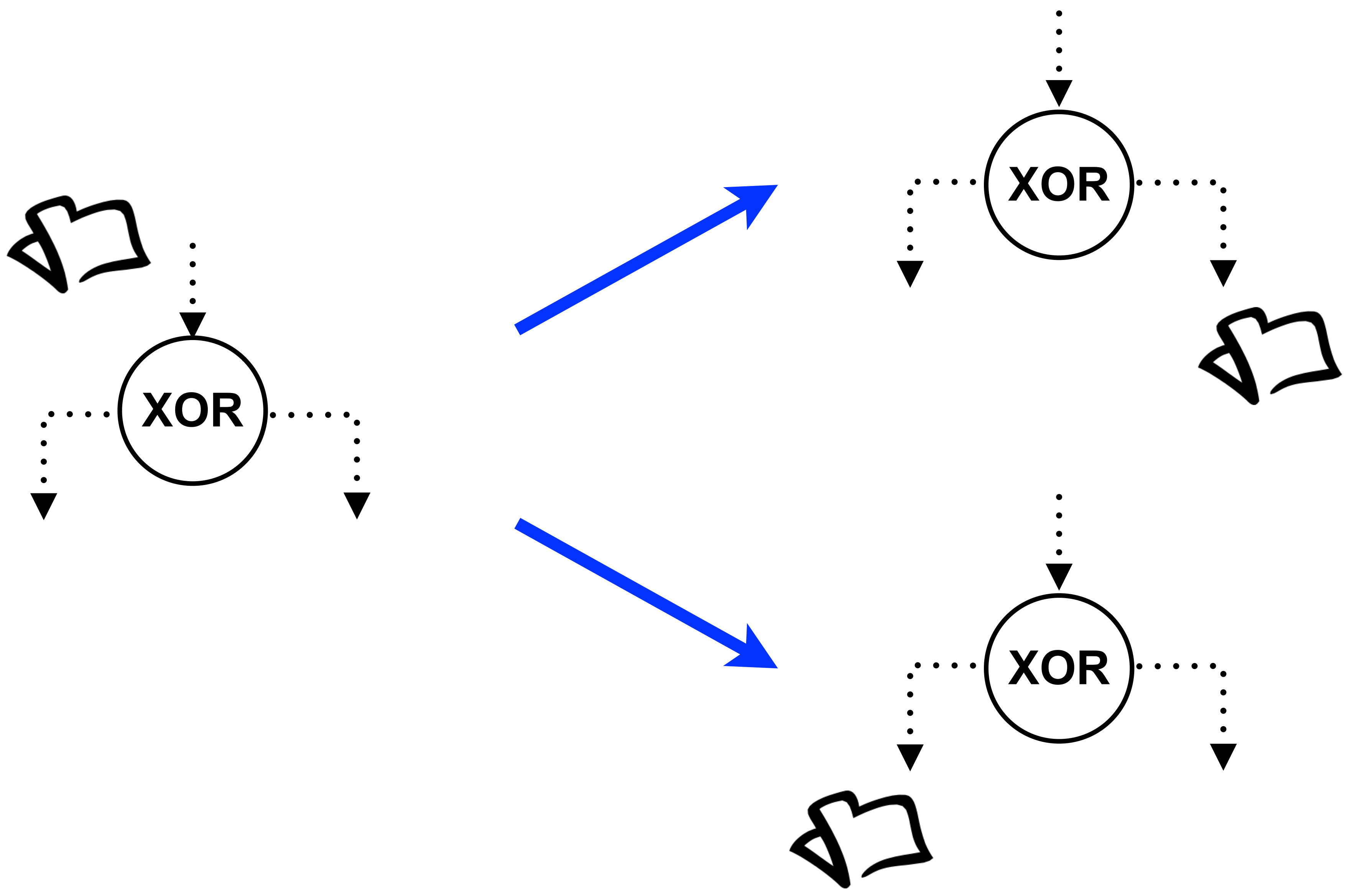
Folder-passing semantics: AND-split



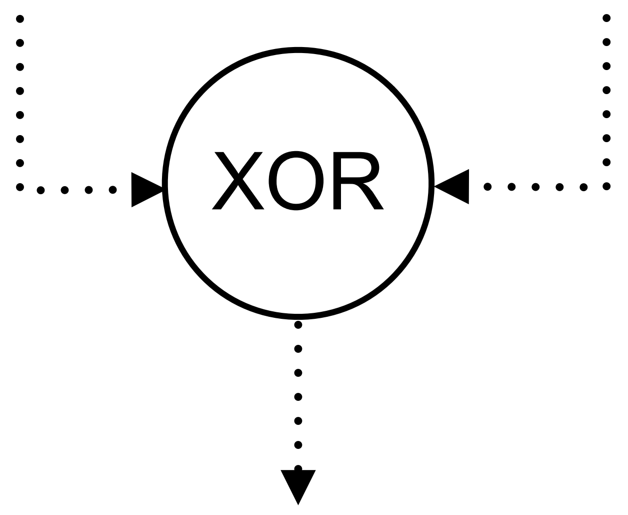
Folder-passing semantics: AND-join



Folder-passing semantics: XOR-split



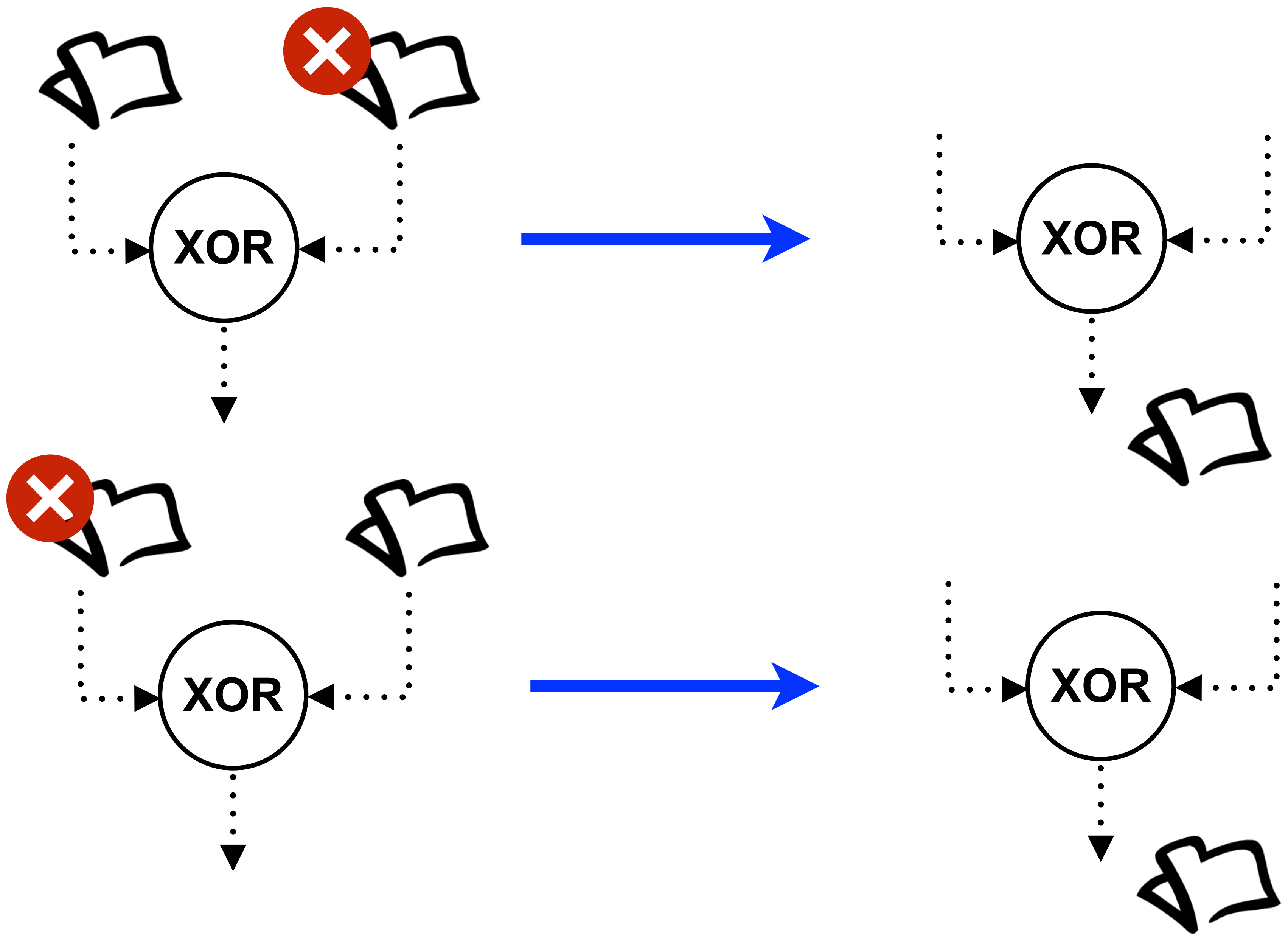
XOR join: intended meaning



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

Folder-passing semantics: XOR-join

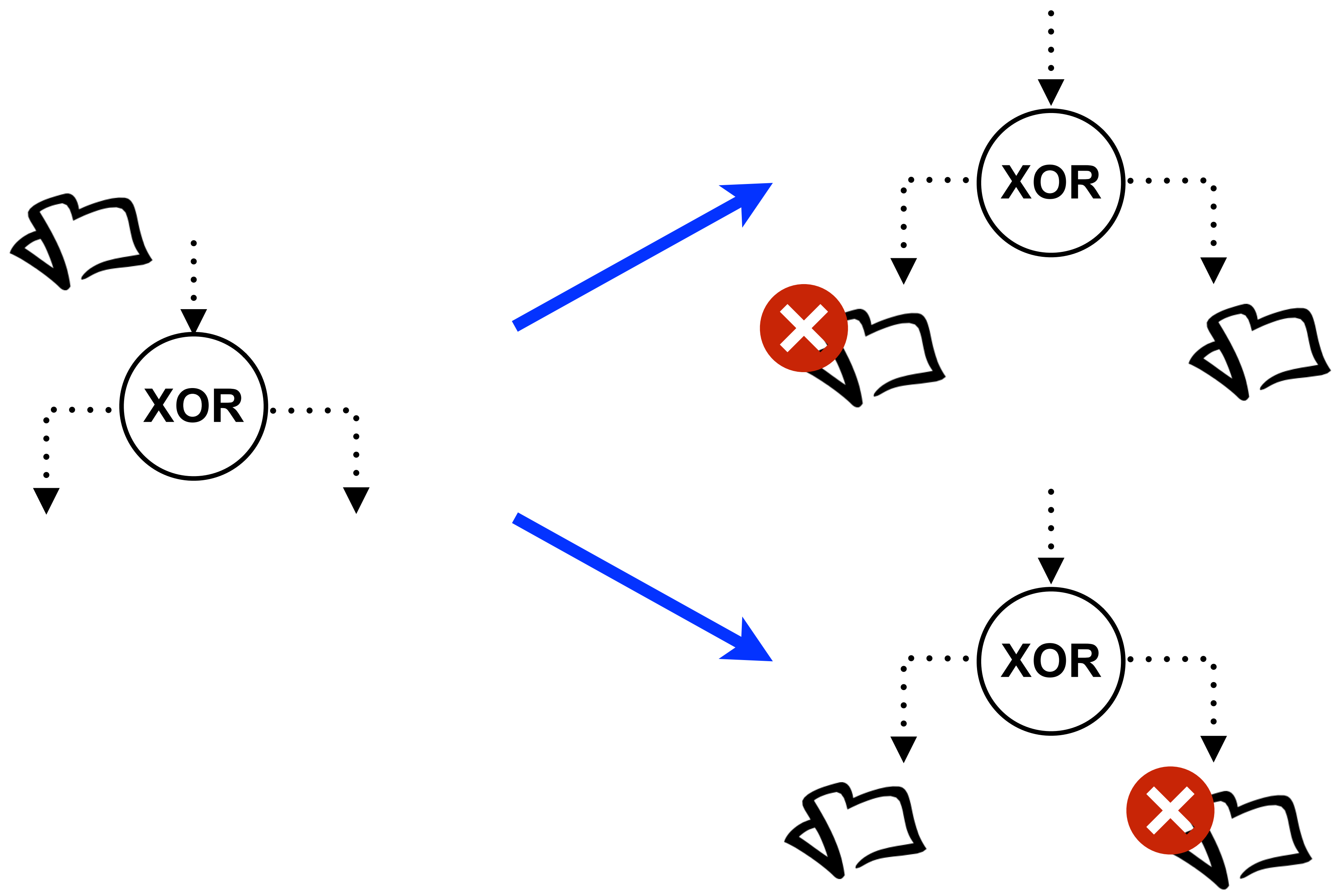


Folder-passing semantics?

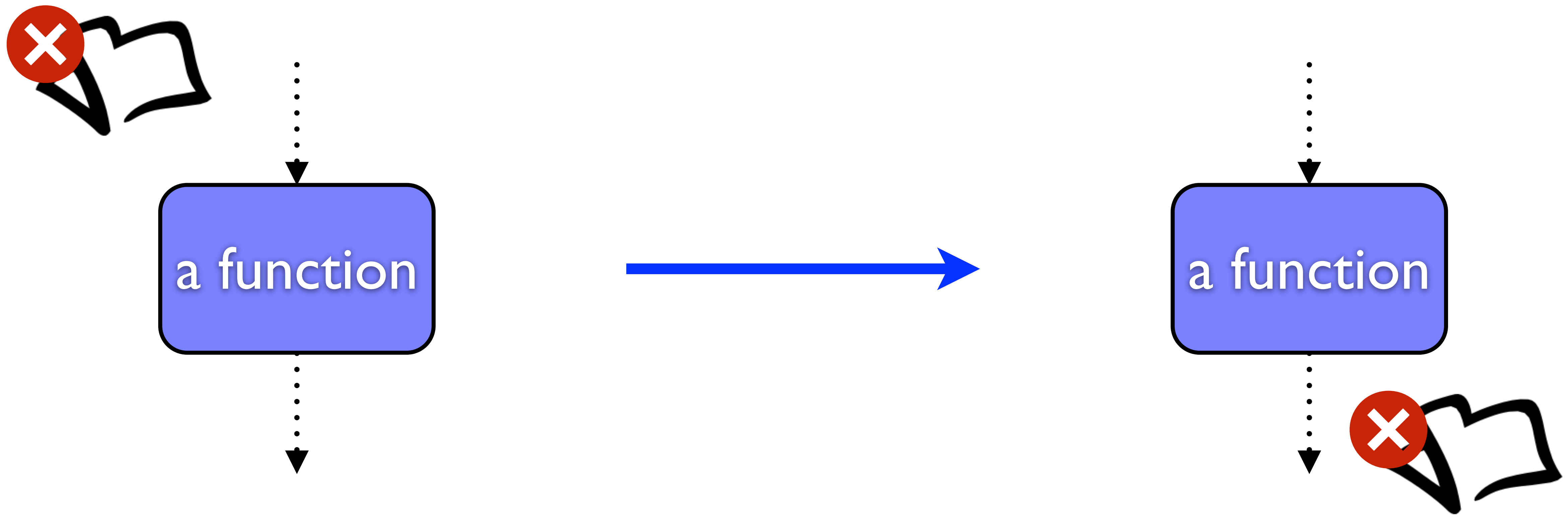
How can we infer the absence of folders?

When and how should such information be propagated?

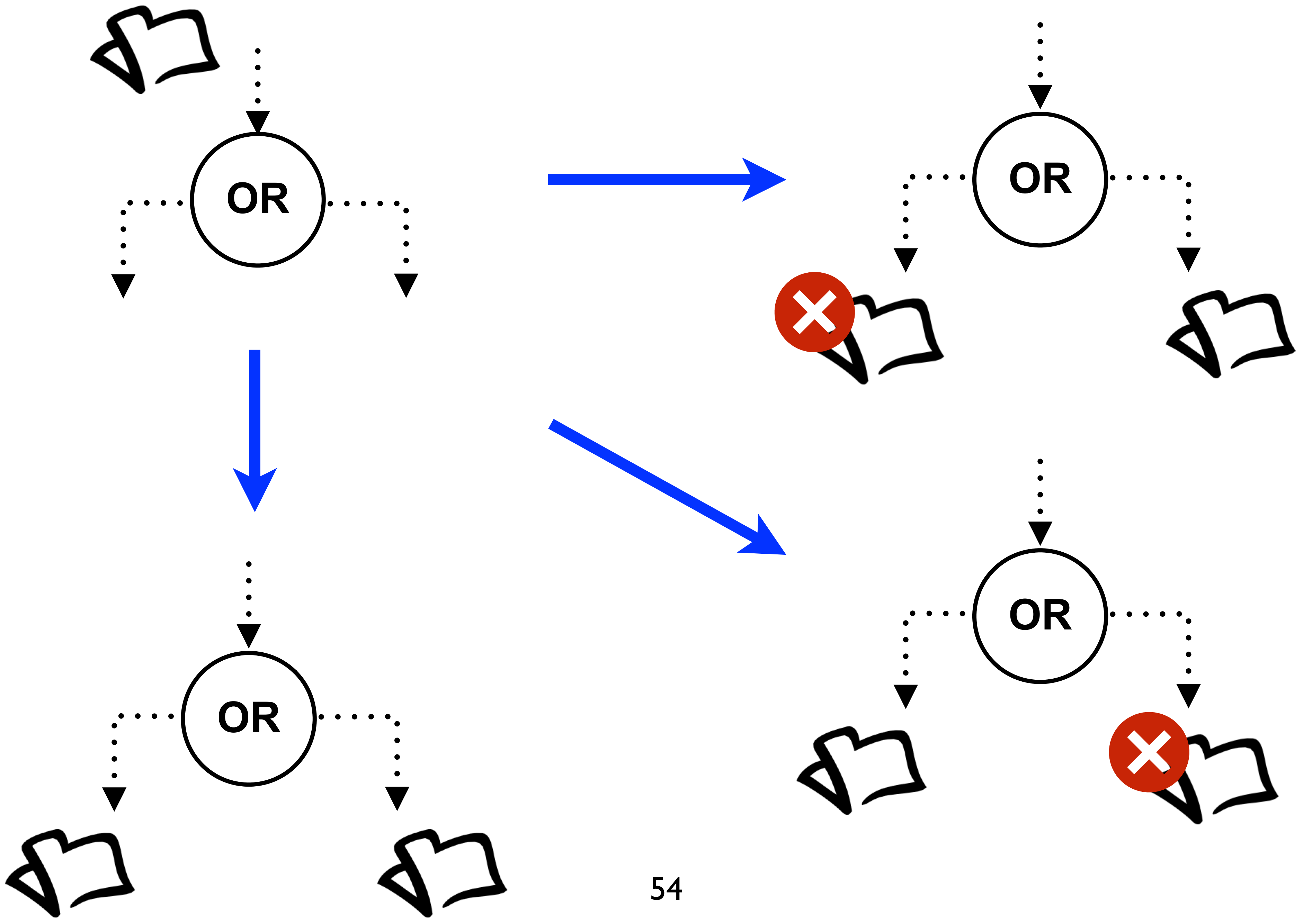
Absence of folders: creation



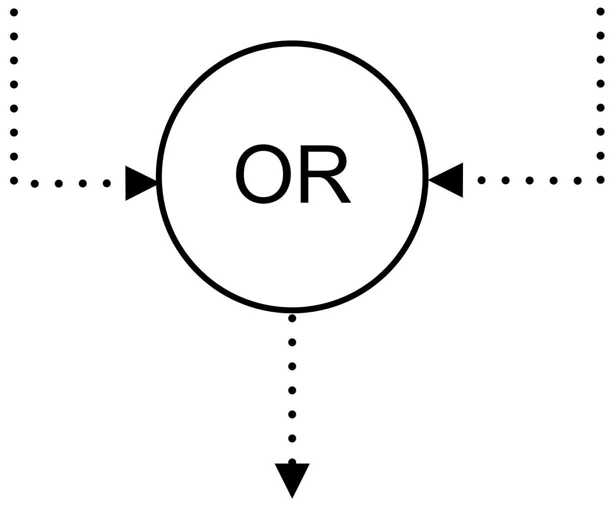
Absence of folders: propagation (example)



Folder-passing semantics: OR-split



OR join: intended meaning

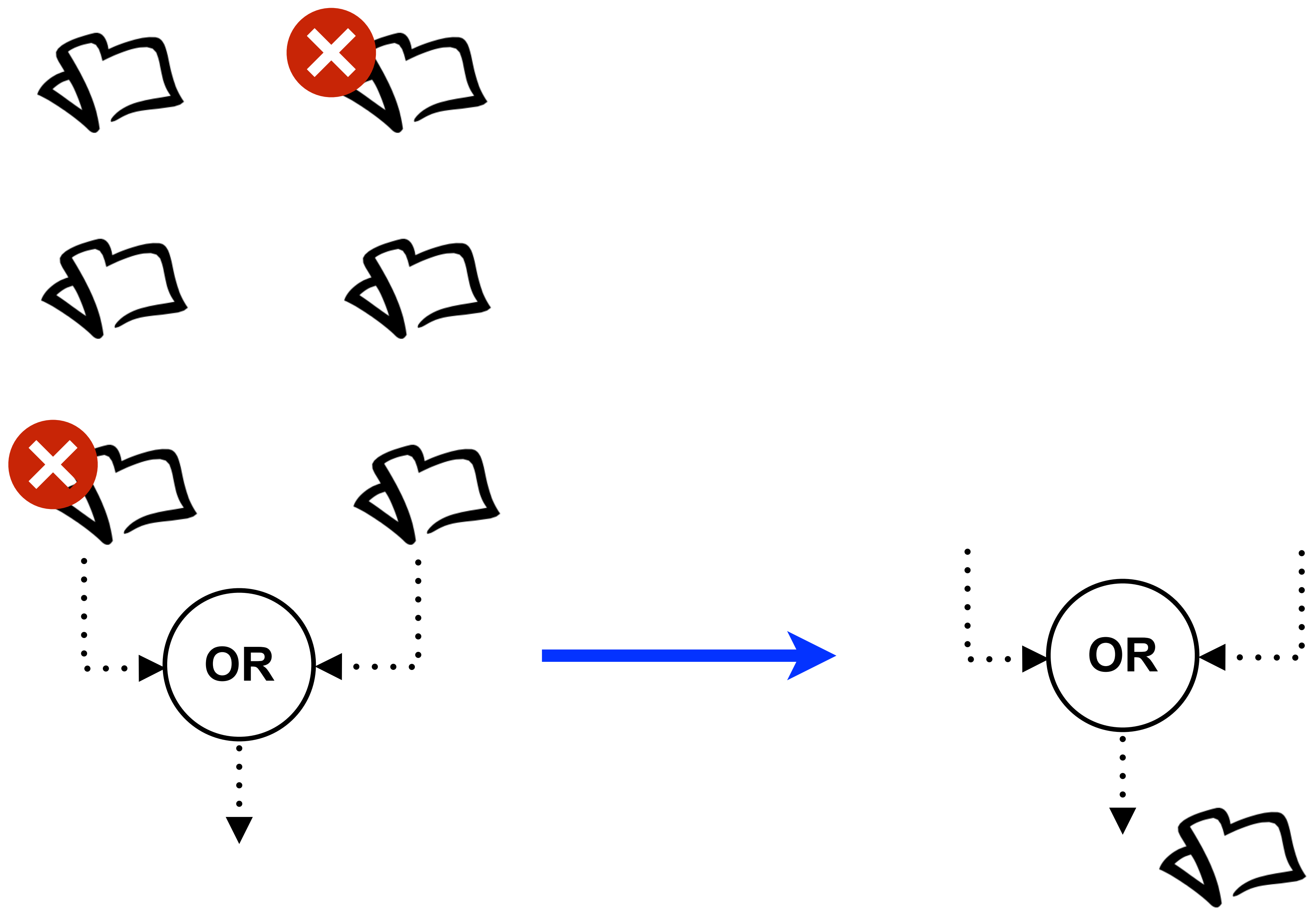


if only one input arrives,
it should release the flow

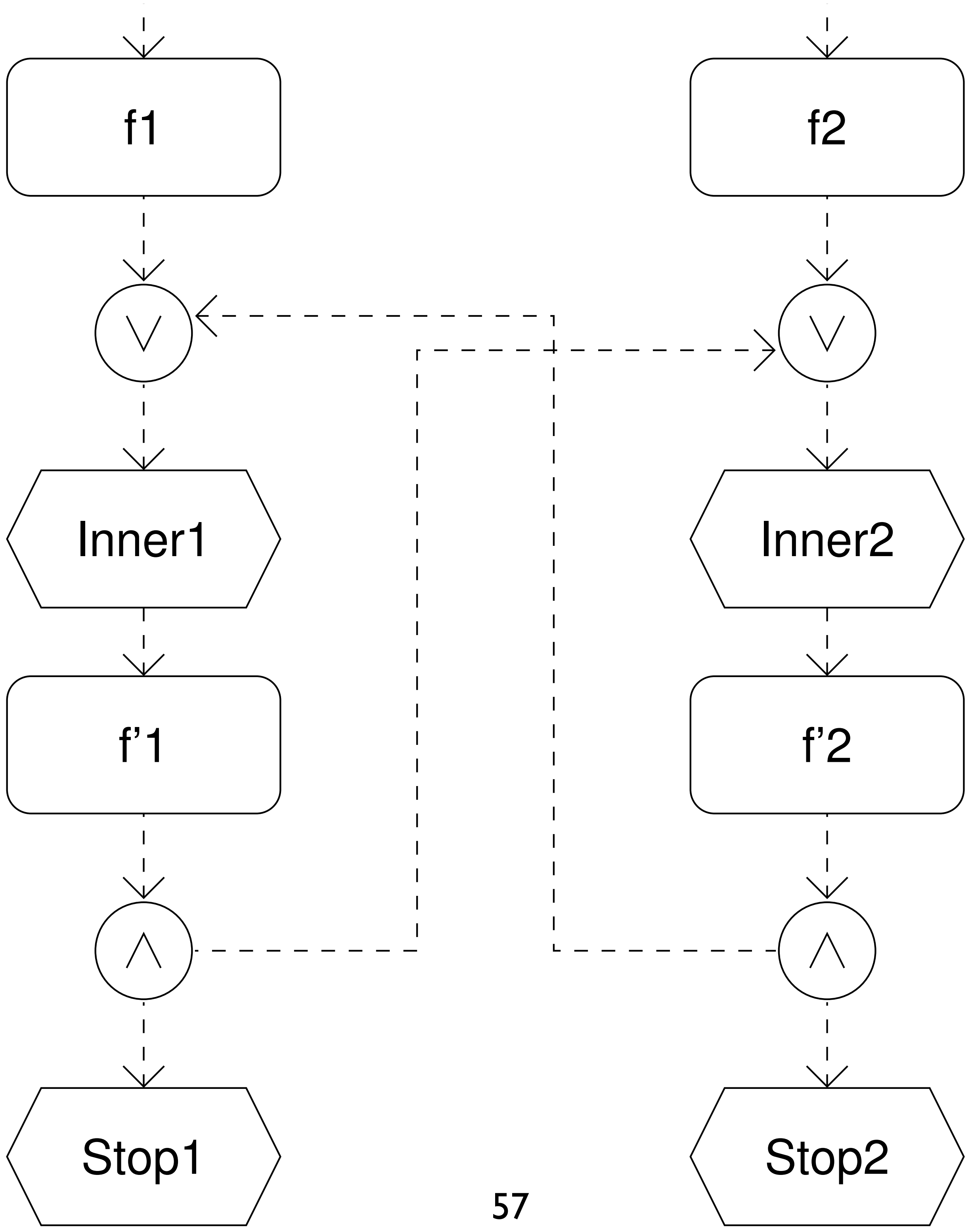
if both inputs arrive,
it should release only one output

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

Folder-passing semantics: OR-join?



A vicious circle?



Decorated EPC

To remove ambiguous behaviour for join connectors, designers can further annotate EPC diagrams

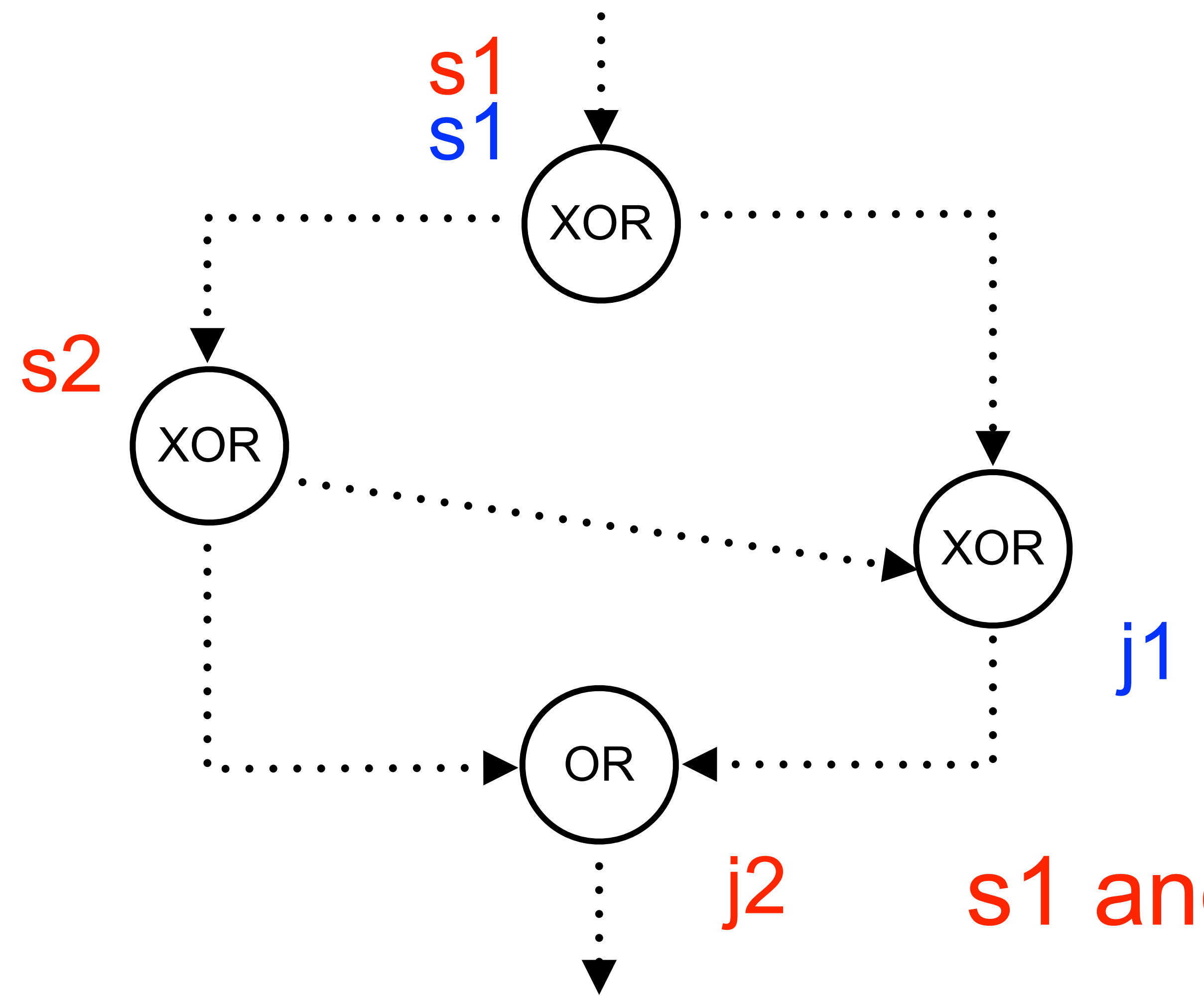
In particular we require to know:

corresponding split
which node separated the flows we are joining
(in the case of XOR/OR join)

applicable policy
how to trigger outgoing flow
(avoid OR join ambiguous behaviour)

Candidate split

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

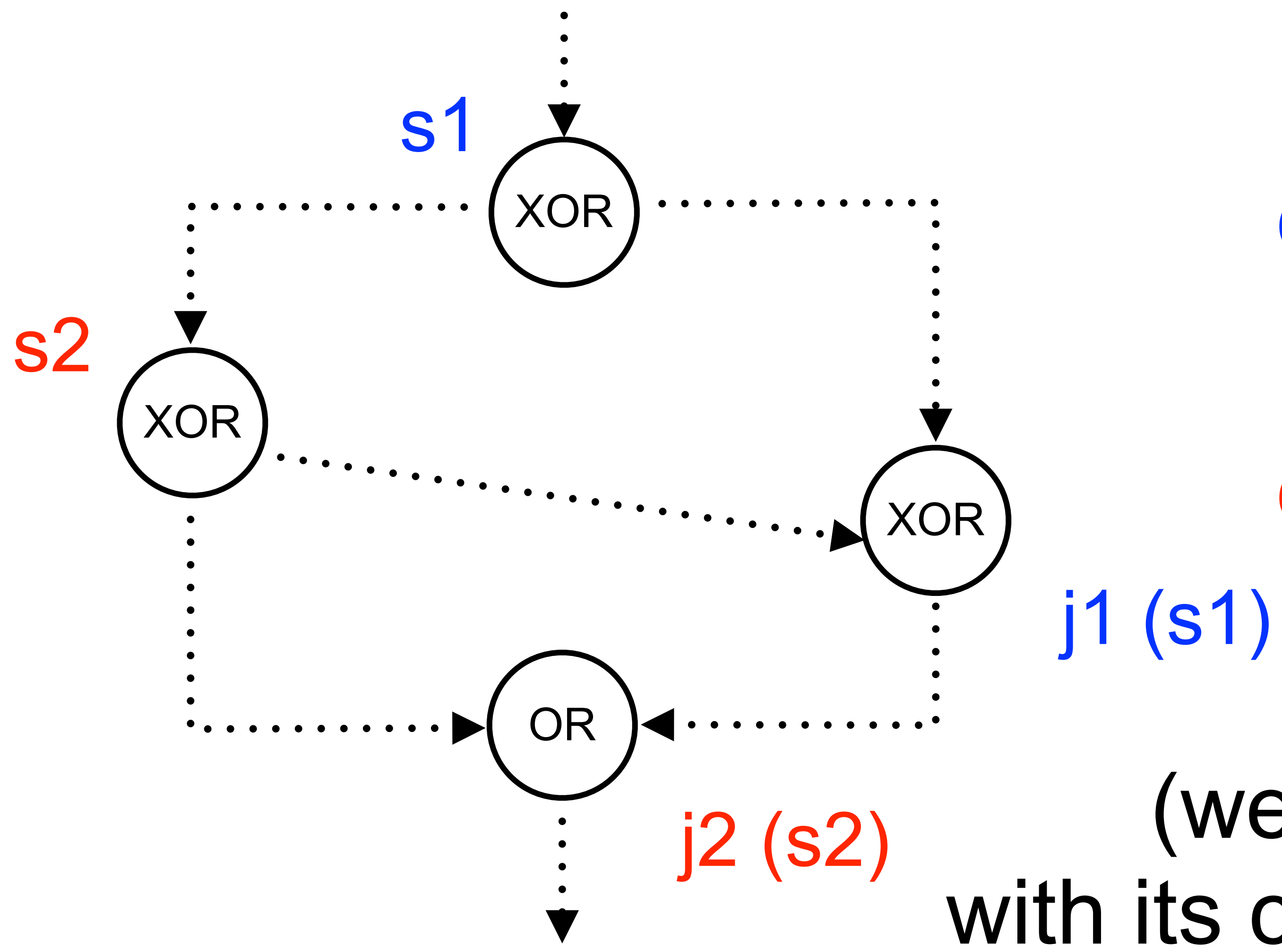


s1 is a candidate split for j1

s1 and s2 are candidate splits for j2

Corresponding split

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



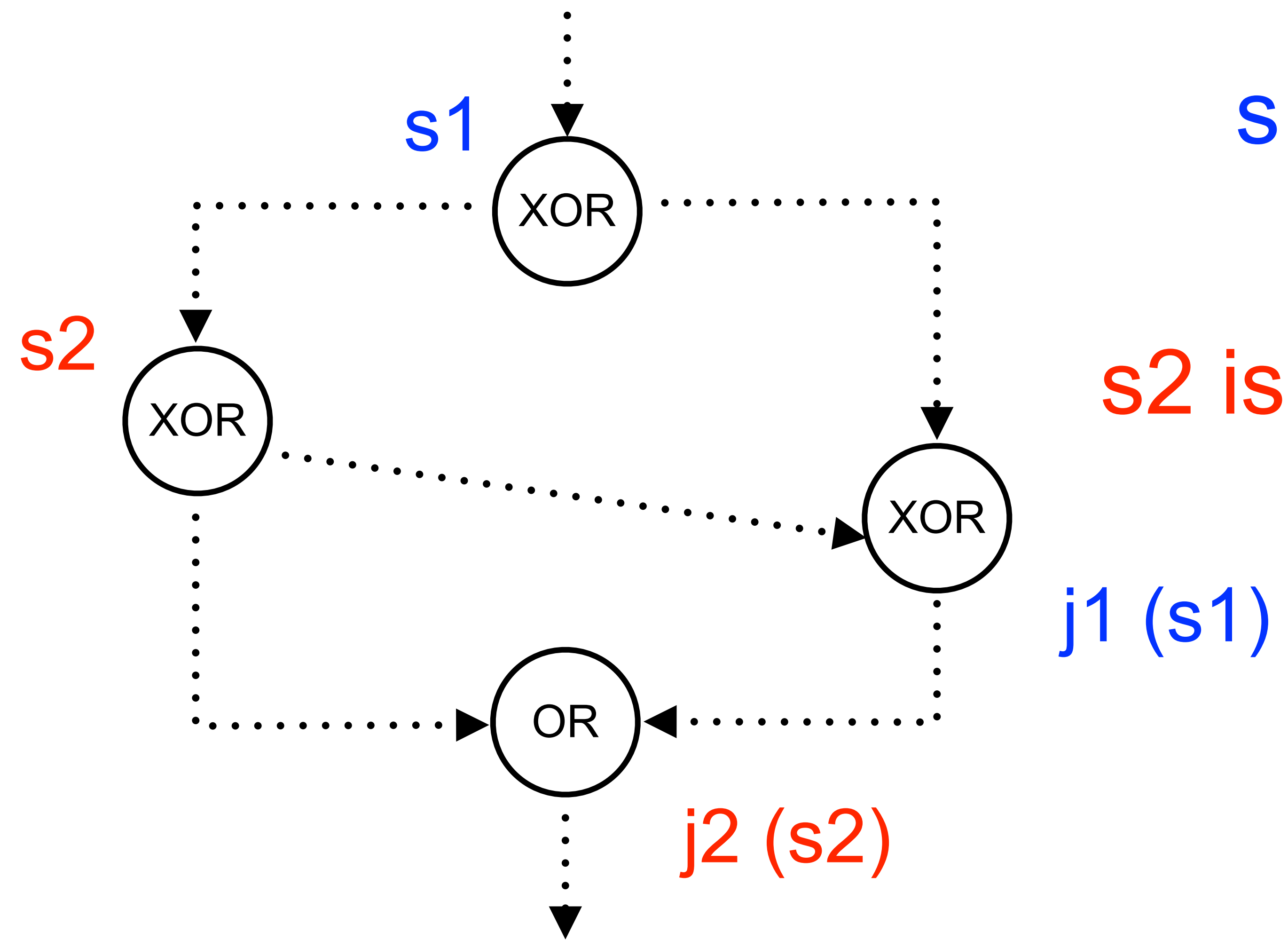
we choose s1 as a
corresponding split for j1

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: policies

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

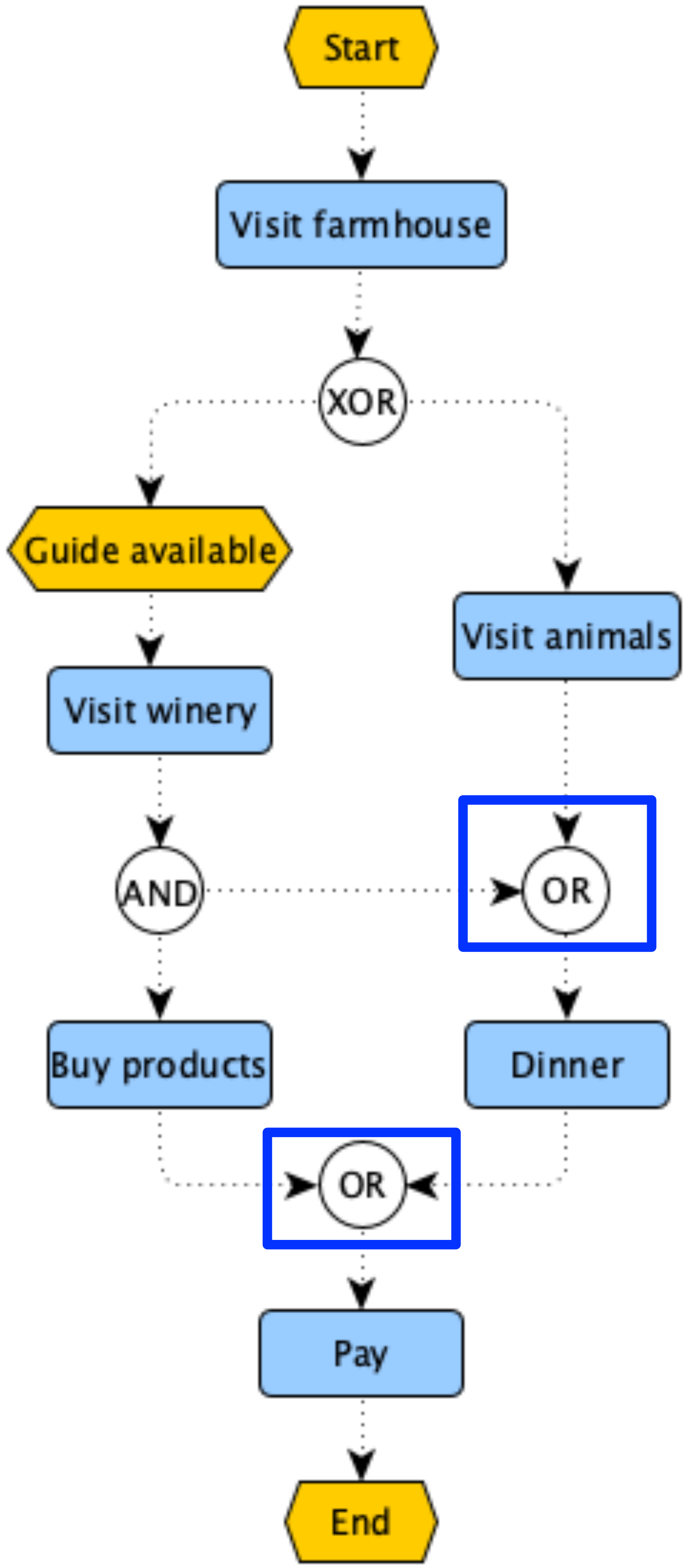
Otherwise, also other policies can be chosen:

every-time: trigger the outgoing path on each input

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

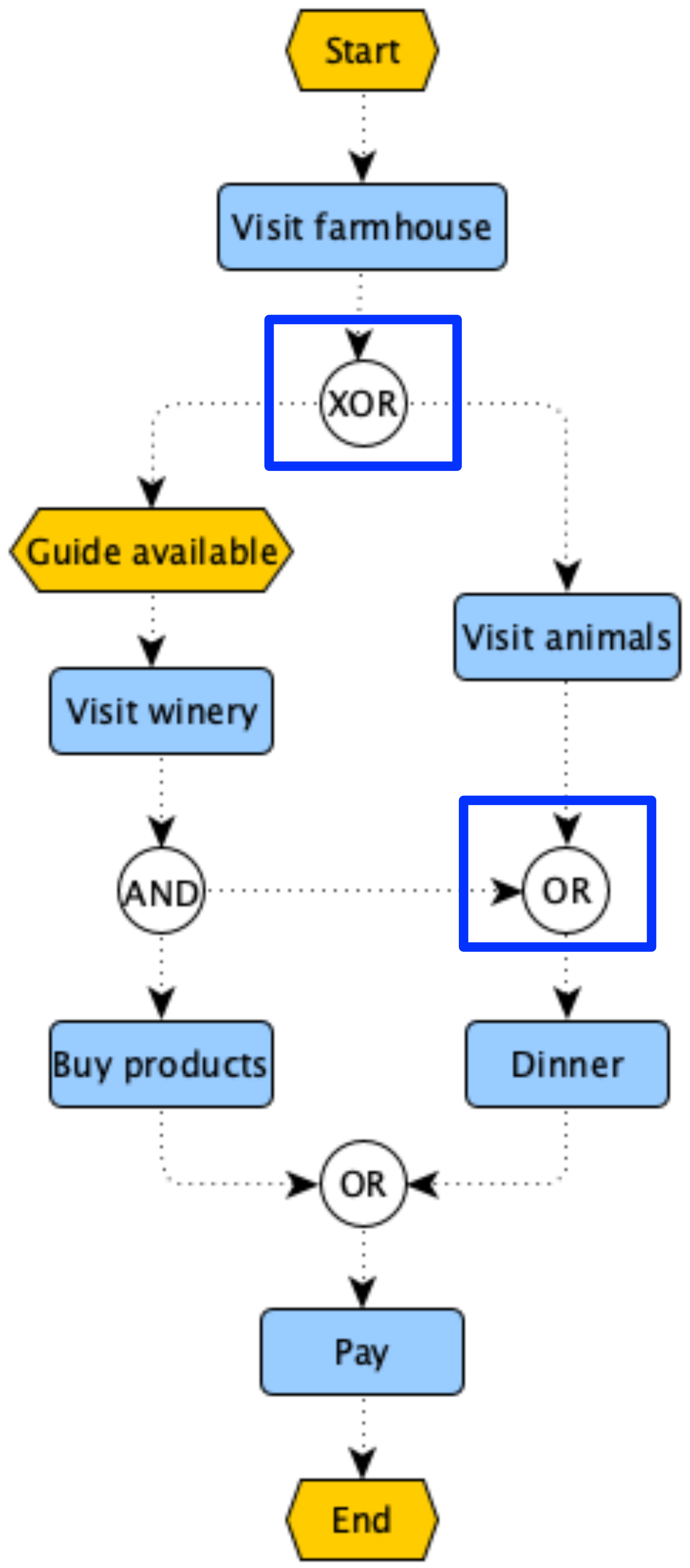
Assumption: every OR join is tagged with a policy (some suggested to have different trapezoid symbols)

Example



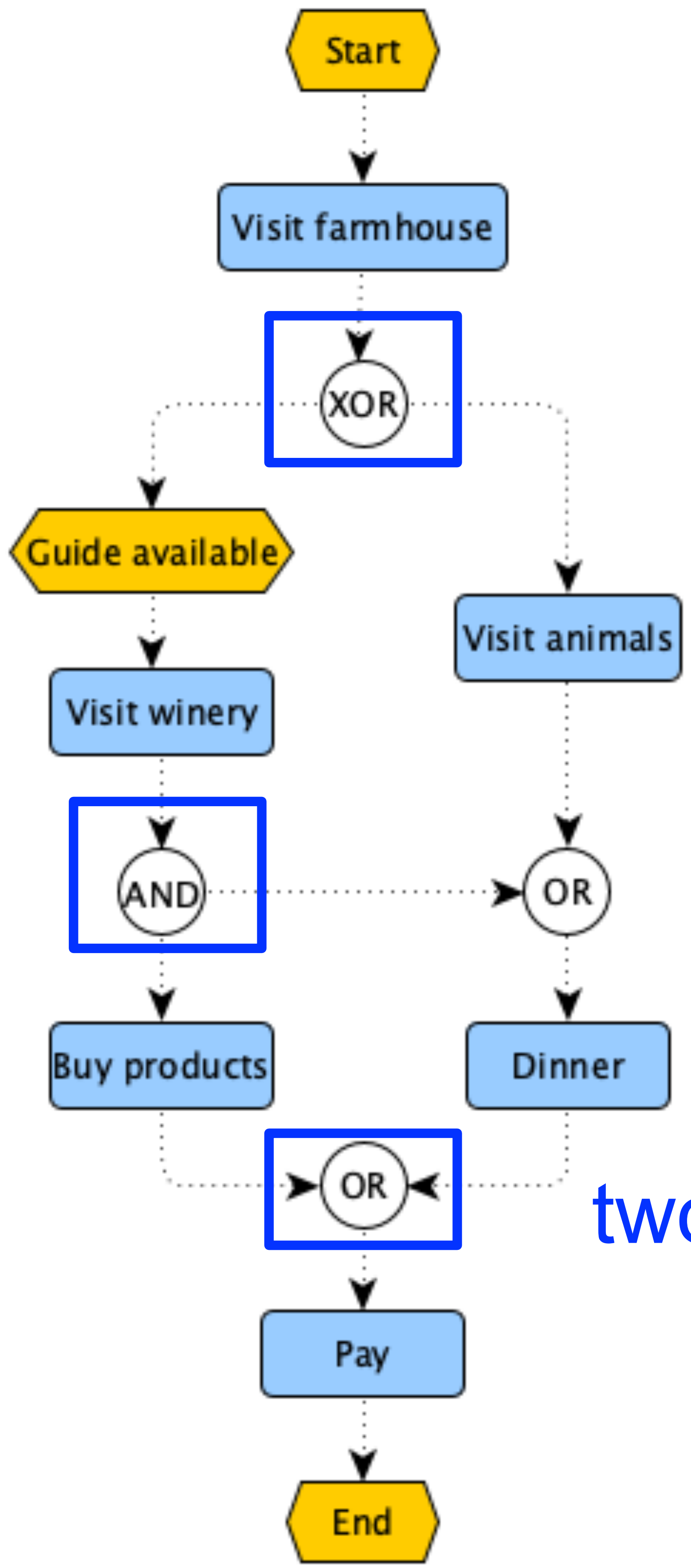
two OR joins
but no OR split

Example

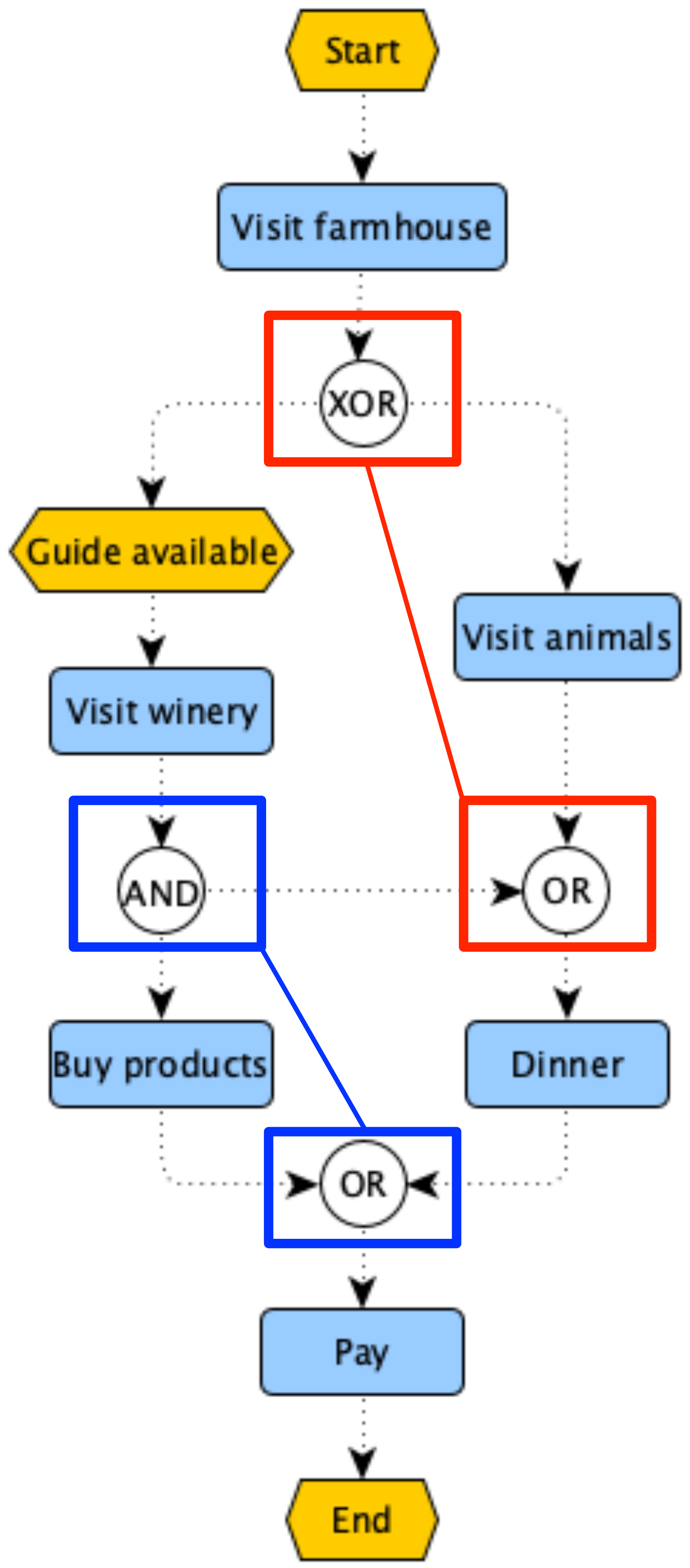


only one
candidate split

Example



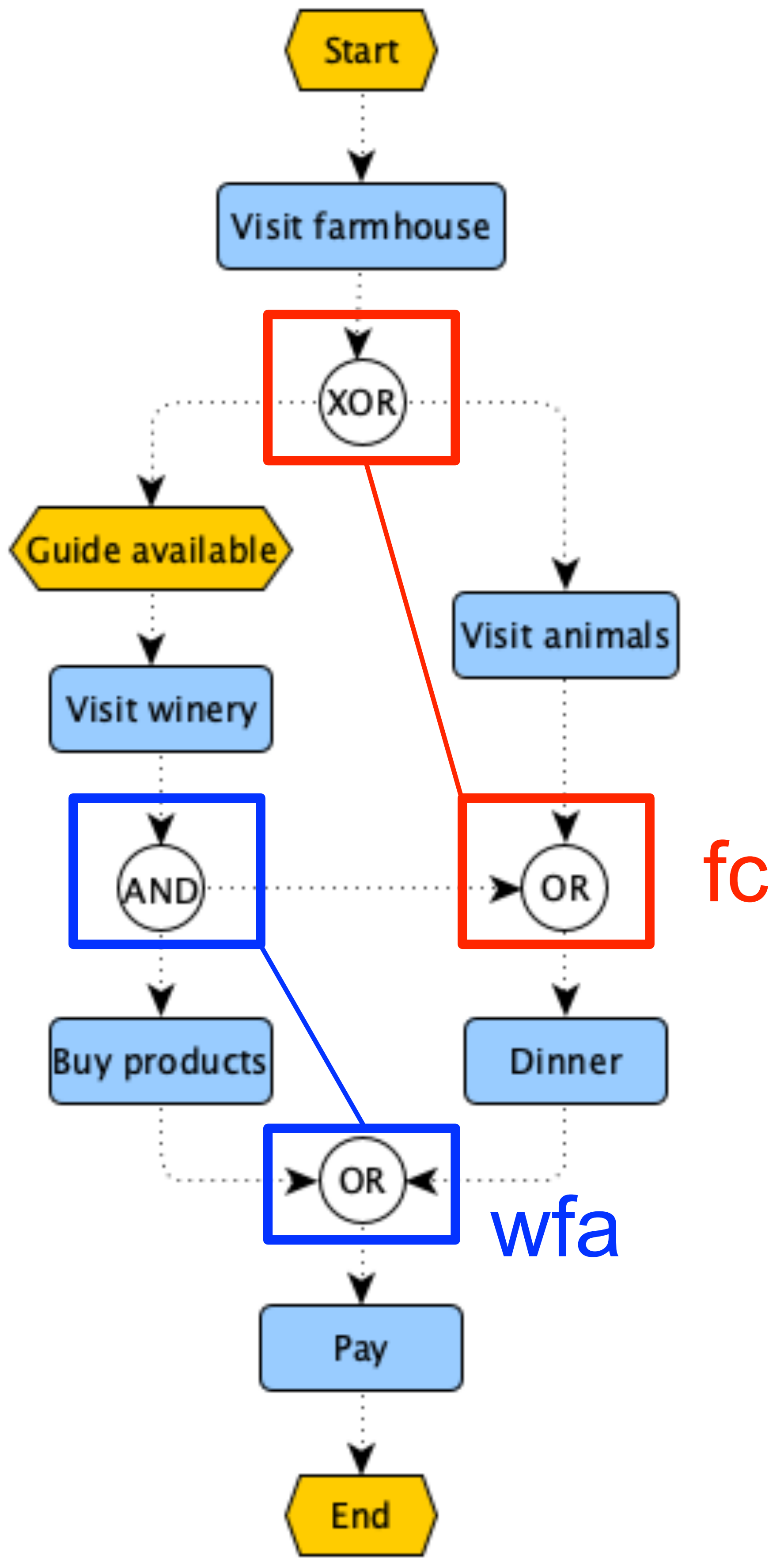
two candidate splits



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

et: every-time
fc: first-come
work well with corresponding XOR split

XOR join: assumption

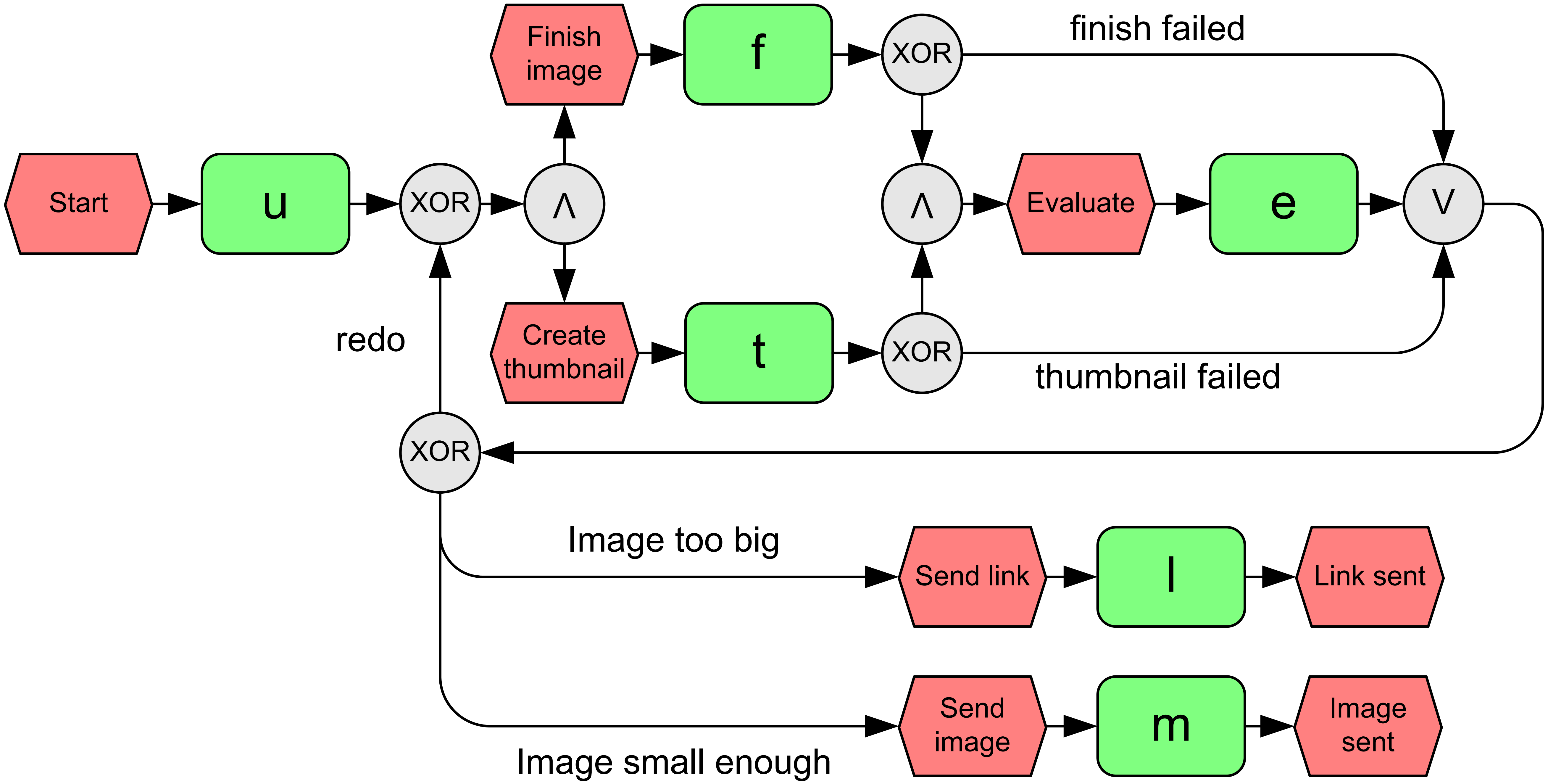
If a XOR join has a **matching split**, the semantics is:
"it blocks if both paths are activated and
it is triggered by a unique activated path"

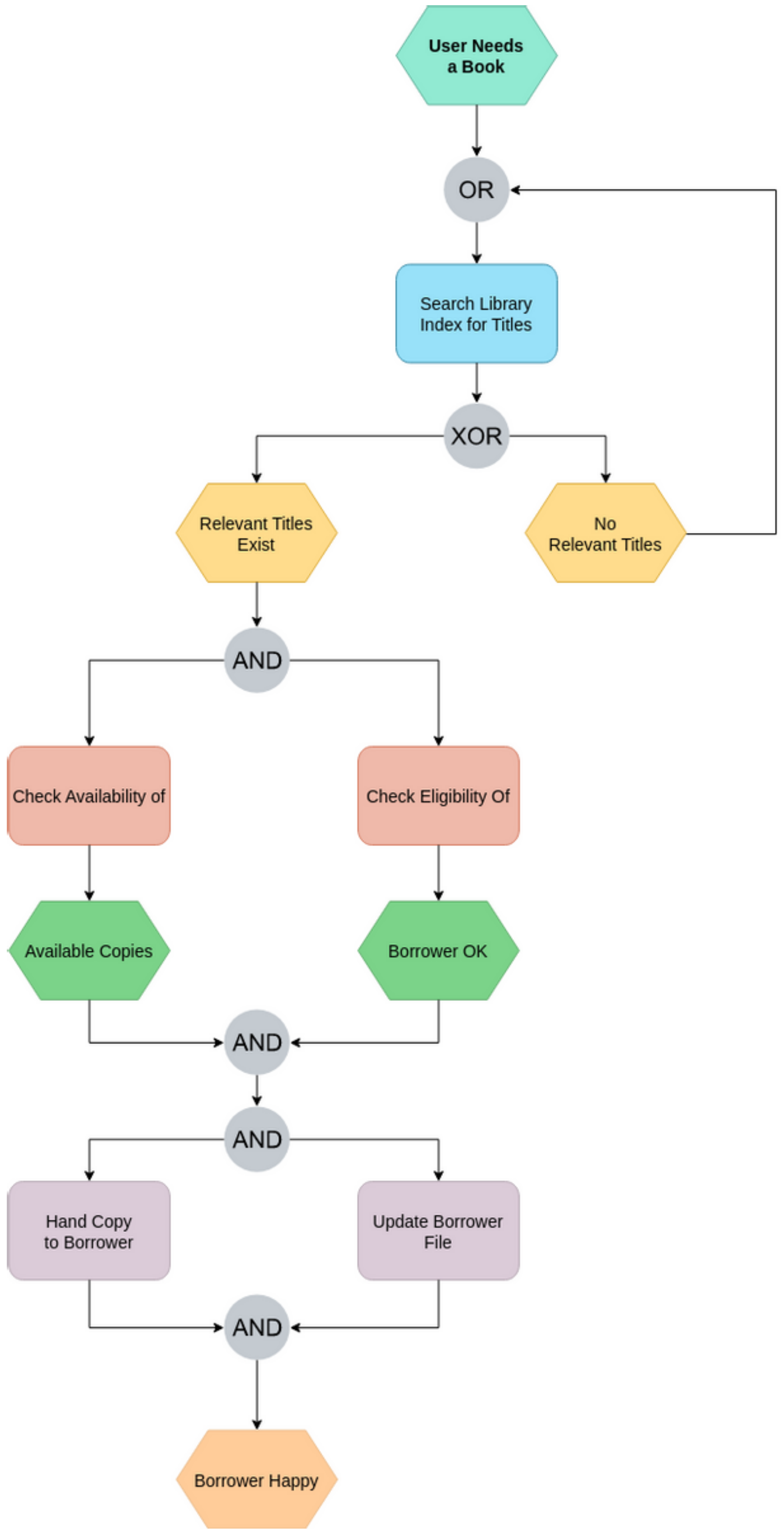
Any policy (wait-for-all, first-come, every-time)
contradicts the exclusivity of XOR
(a token from one path can be accepted only if we make sure that no second token will arrive via the other path)

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

EPC Sample Diagrams

Example: any comment?

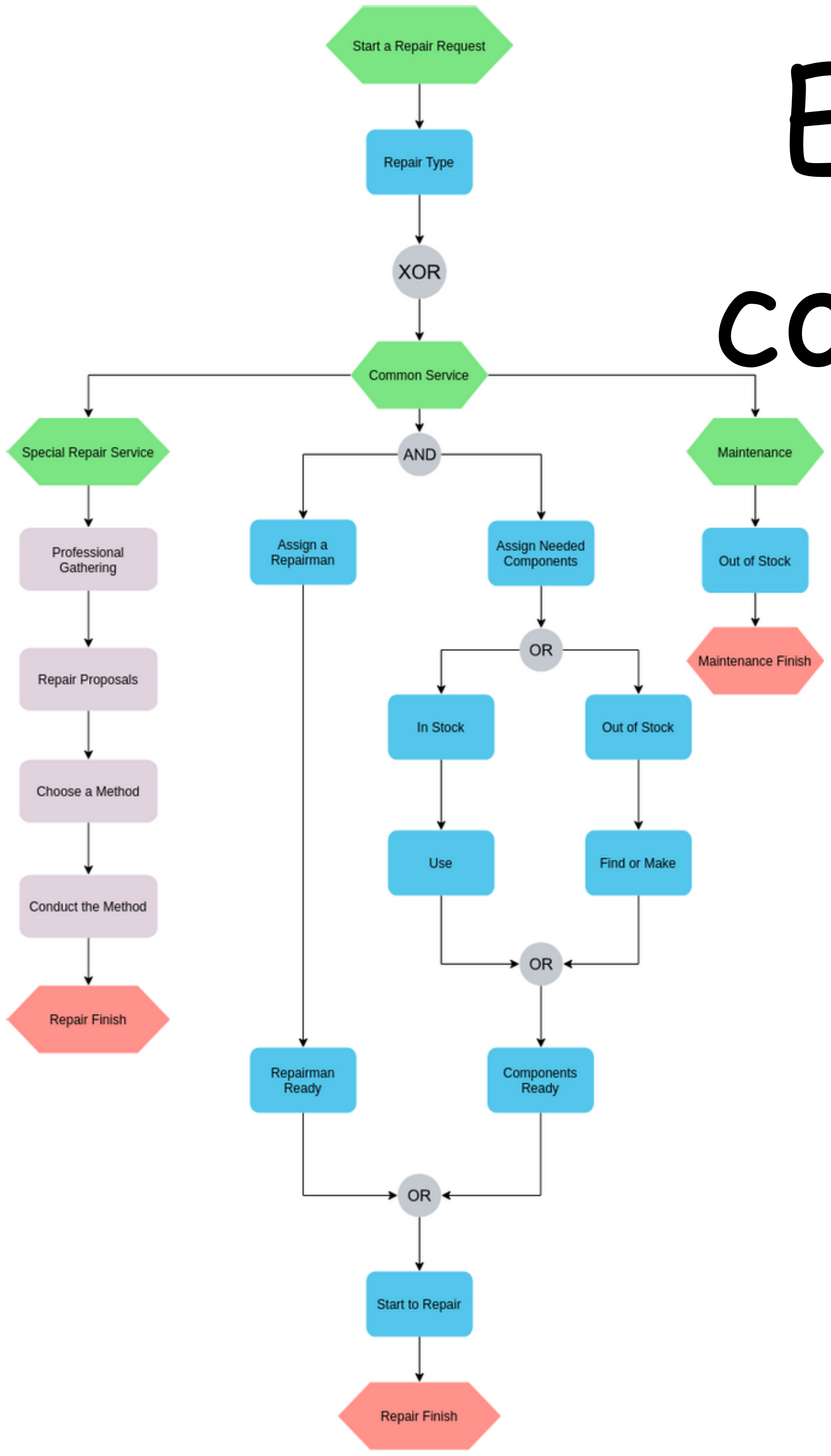




Example: comments?

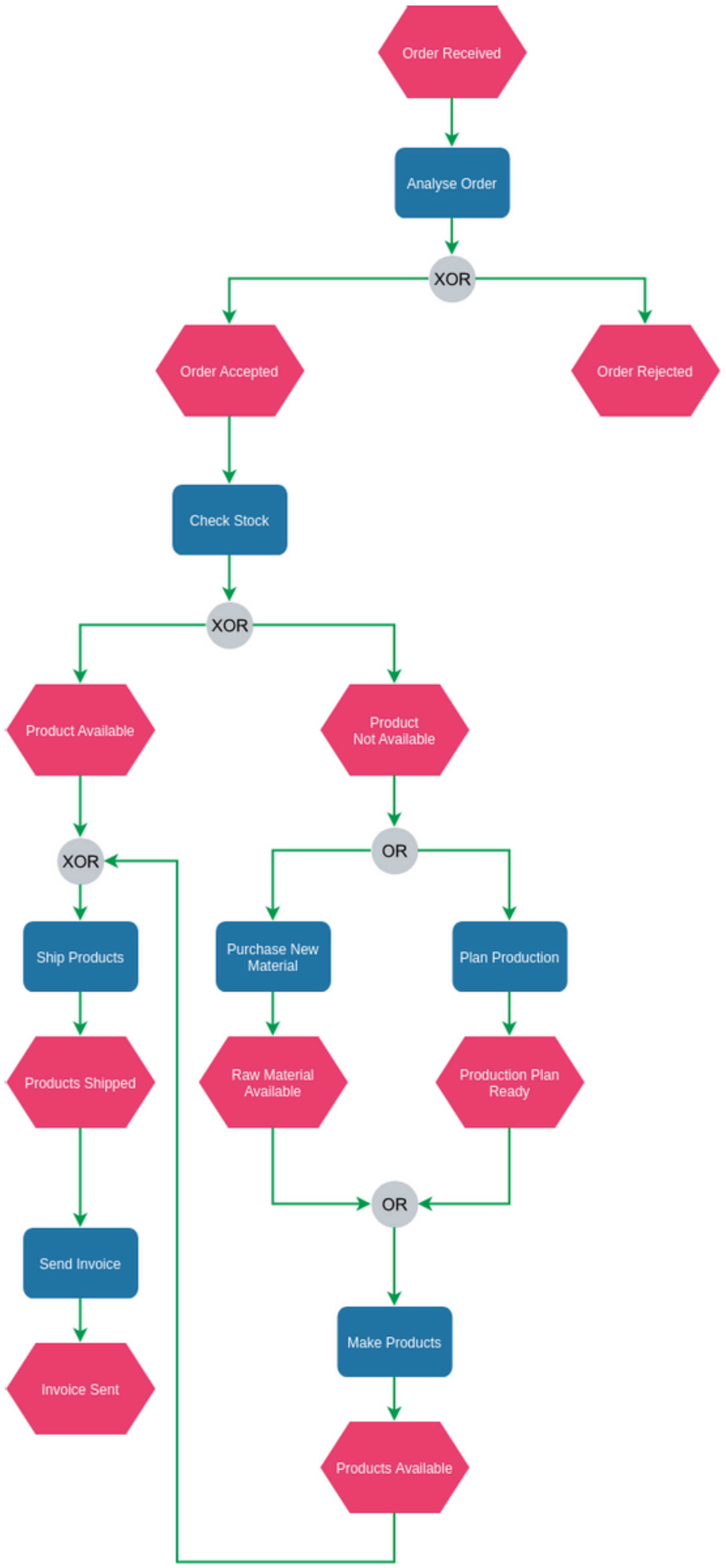
Visual Paradigm Online

Example: comments?



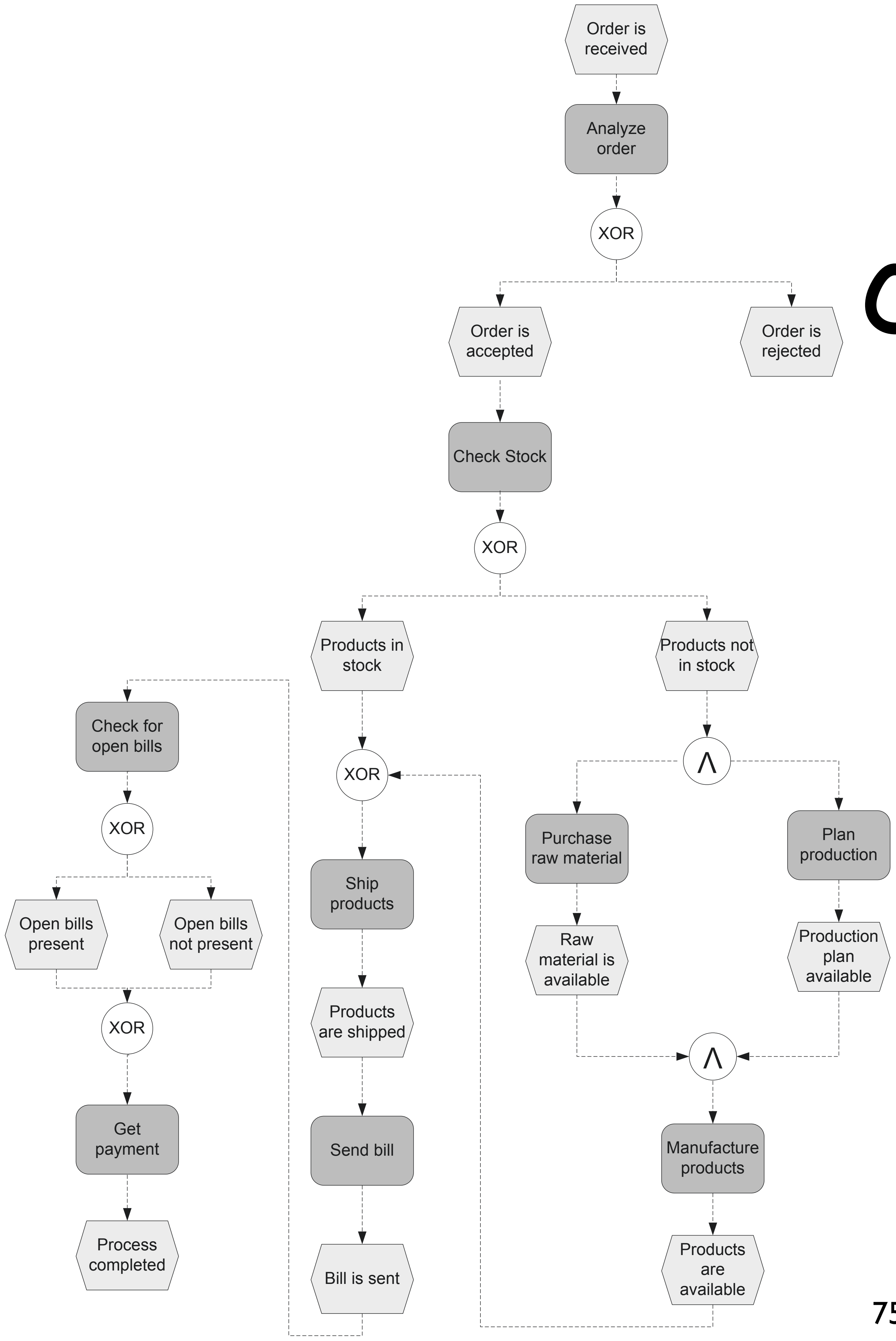
Visual Paradigm Online

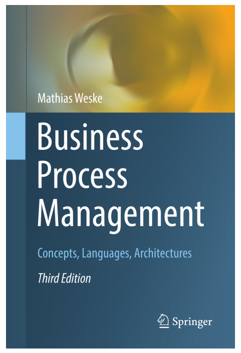
Example: comments?



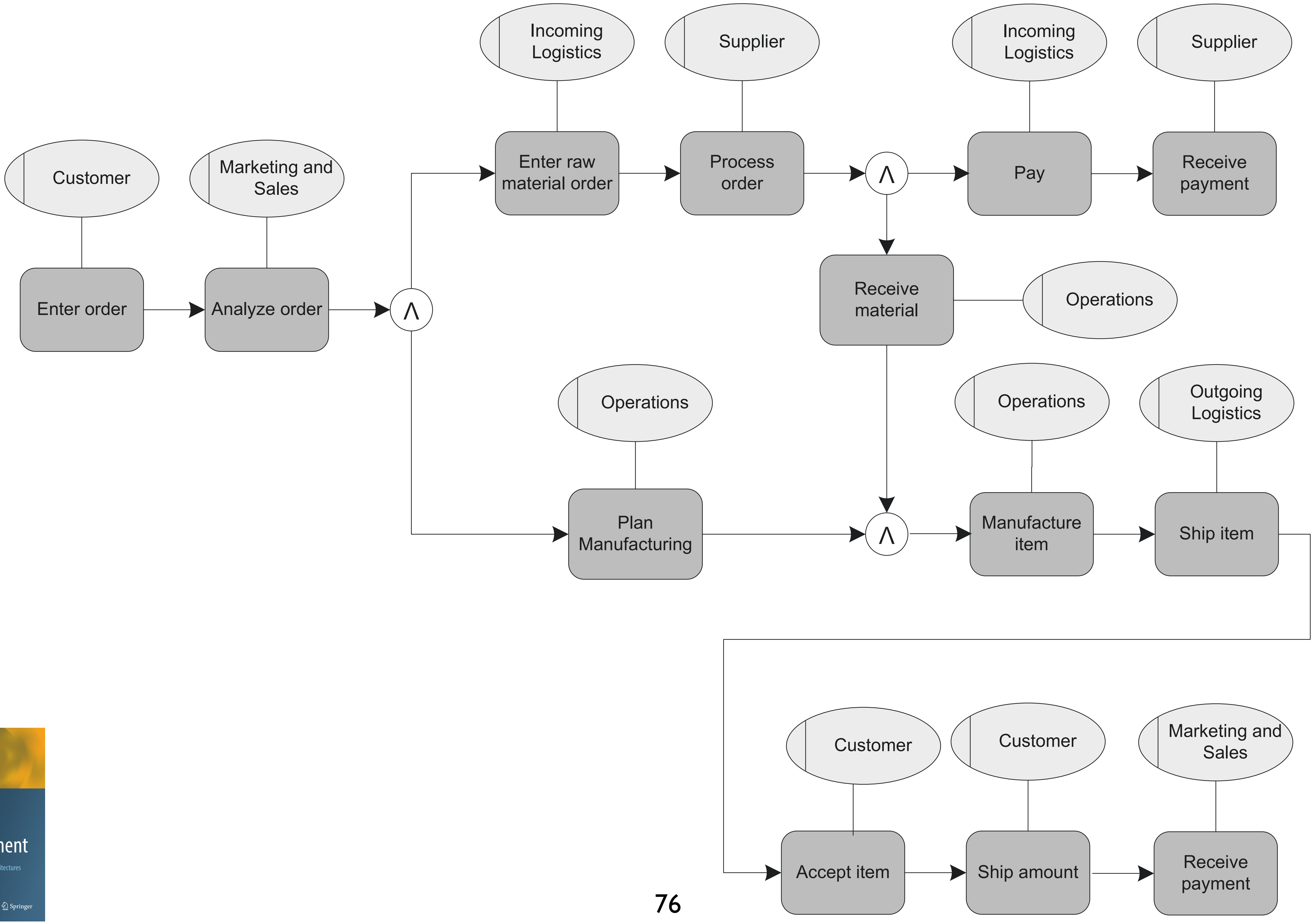
Visual Paradigm Online

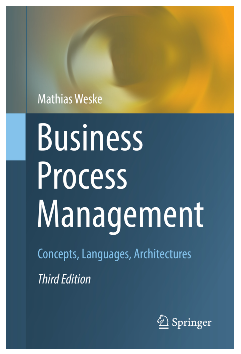
Example: comments?




Mathias Weske
Business Process Management
Concepts, Languages, Architectures
Third Edition
Springer

Example: any comment?





Mathias Weske
Business Process Management
Concepts, Languages, Architectures
Third Edition
Springer

Example: comments?

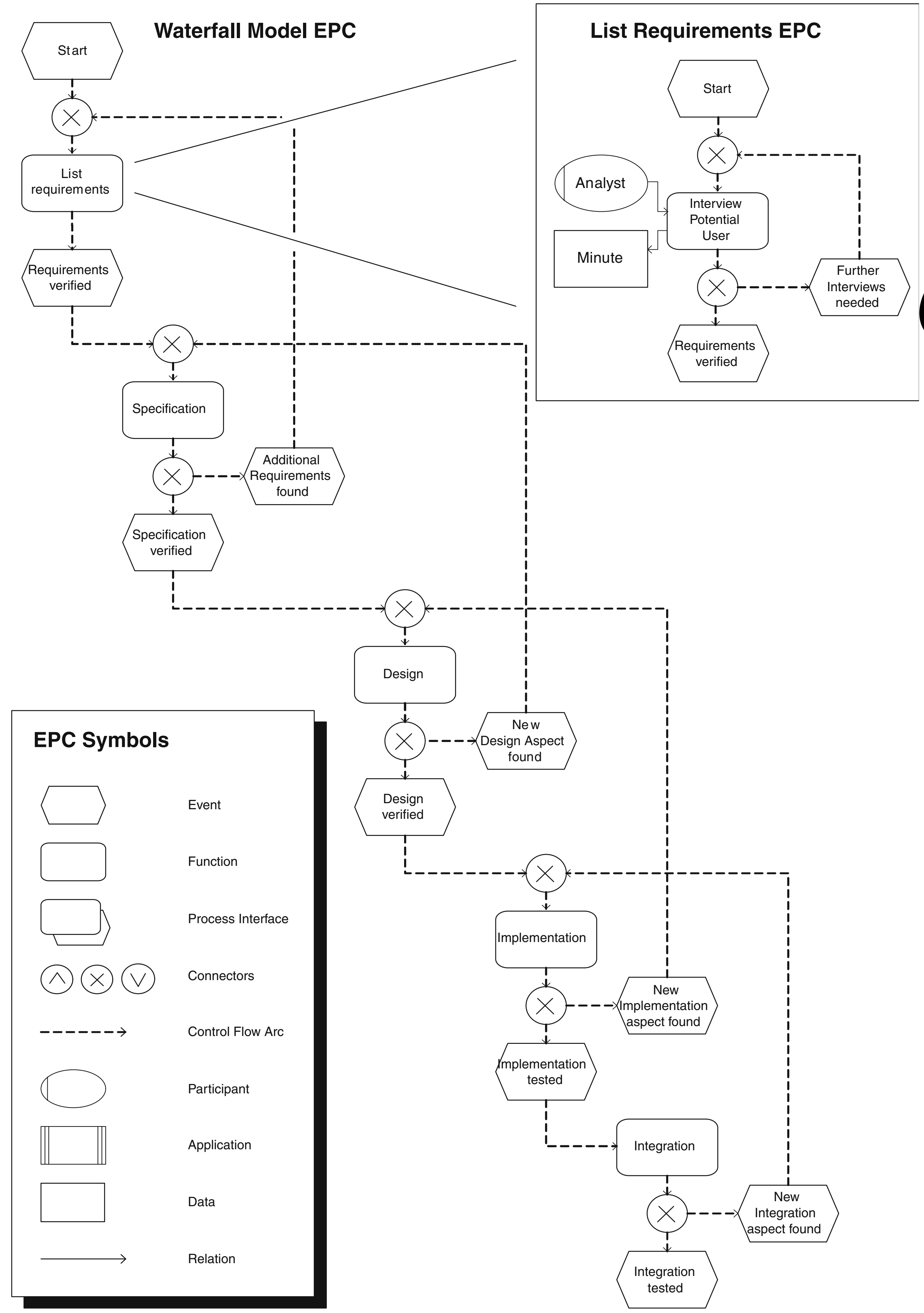
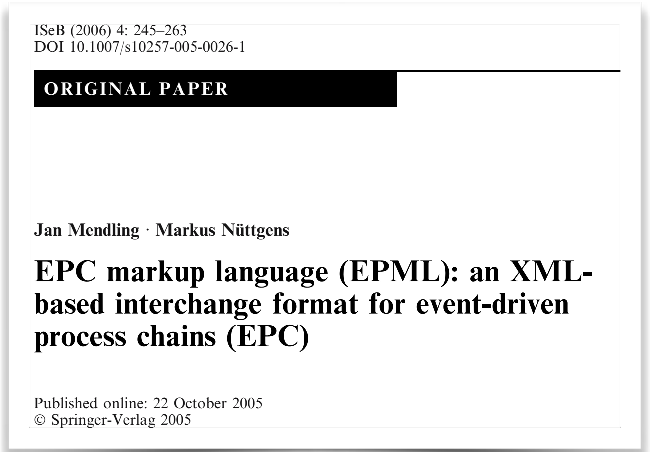


Fig. 1 Event-driven process chains representing the waterfall model for software engineering


ISeB (2006) 4: 245–263
DOI 10.1007/s10257-005-0026-1

ORIGINAL PAPER

Jan Mendling · Markus Nüttgens

EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC)

Published online: 22 October 2005
© Springer-Verlag 2005

Exercises

Search for EPC diagram drawing software products.
For each product found, annotate the following features:

1) which OS is supported? (Windows, Apple, Linux,...)

2) is it free? if not, describe its pricing.

3) is .epml format supported?

4) if you install the product, rate your user experience / usability (on the scale 1-5 stars)

Send your findings to: **`bruni@di.unipi.it`**

Exercises

Transfer the following verbal description into an EPC diagram

You are tasked with modeling the Customer Order Process of a small e-commerce company.
The process starts when a customer places an order online and ends when the order is successfully delivered.
The process must involves at least the following activities:
checking if the items are available in stock,
a notification to the customer if the items are not available,
the preparation of the order for shipment,
and the processing of the payment.

Send your solutions to: `bruni@di.unipi.it`