Business Processes Modelling

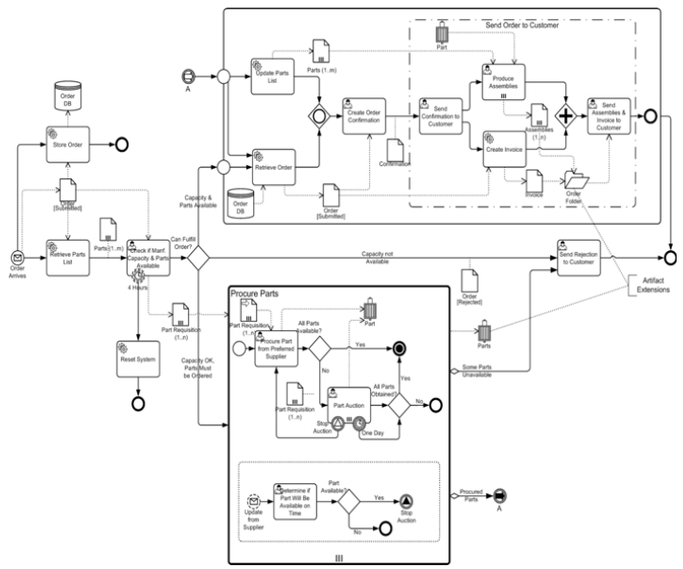
MPB (6 cfu, 295AA)

Roberto Bruni

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

21 - BPMN analysis

Object



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

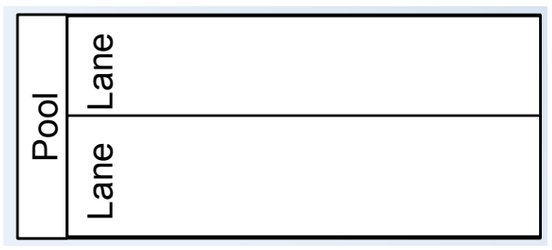
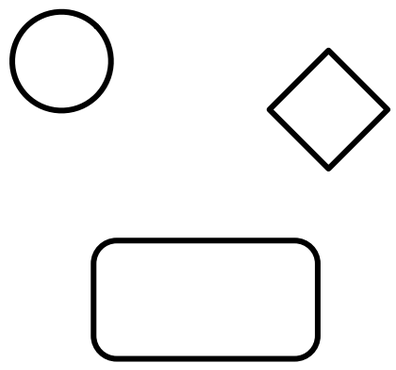
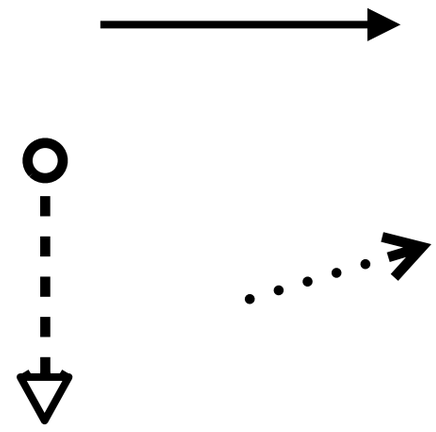
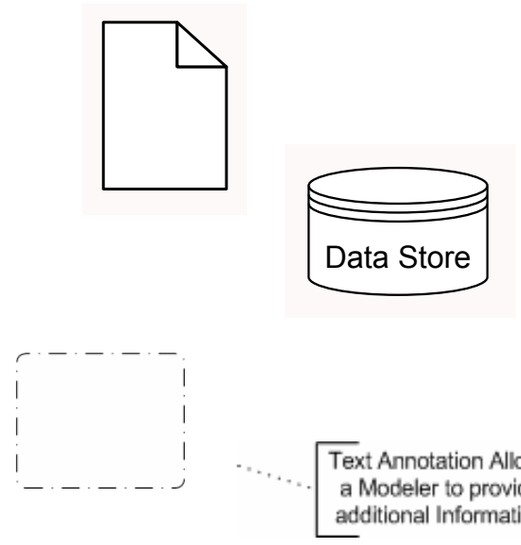
Ch.5.7 of Business Process Management: Concepts, Languages, Architectures

BPMN Diagrams

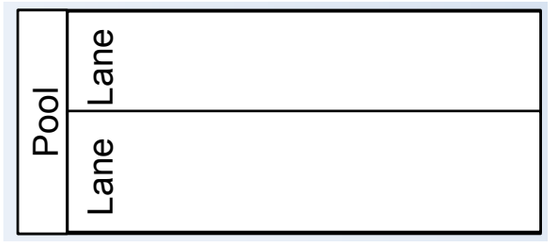
Business process diagrams

BPMN defines a standard for
Business Process Diagrams (**BPD**)
based on **flowcharting technique**

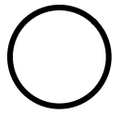
Four categories of elements

swimlanes	flow objects	connecting objects	artefacts
			

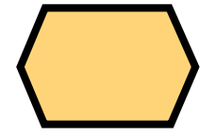
BPMN vs EPC (roughly)



swimlanes

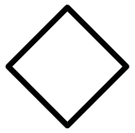


event

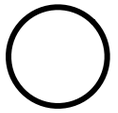
event 

activity

function 



gateway

connector 

sequence flow

control flow 

message flow

BPMN 2.0 - Business Process Model and Notation

http://bpmb.de/poster

Activities

Task

A **Task** is a unit of work, the job to be performed. When marked with a [+] symbol it indicates a **Sub-Process,** an activity that can be refined.

Transaction

A **Transaction** is a set of activities that logically belong together; it might follow a specified transaction protocol.

Event Sub-Process

An **Event Sub-Process** is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.

Call Activity

A **Call Activity** is a wrapper for a globally defined Sub-Process or Task that is reused in the current process.

Activity Markers

Markers indicate execution behavior of activities:

- Sub-Process Marker
- Loop Marker
- Parallel MI Marker
- Sequential MI Marker
- Ad Hoc Marker
- Compensation Marker

Task Types

Types specify the nature of the action to be performed:

- Send Task
- Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task
- Script Task

Sequence Flow
defines the execution order of activities.

Default Flow
is the default branch to be chosen if all other conditions evaluate to false.

Conditional Flow
has a condition assigned that defines whether or not the flow is used.

Gateways

Exclusive Gateway
When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Event-based Gateway
Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

Parallel Gateway
When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

Inclusive Gateway
When splitting, one or more branches are activated. All active incoming branches must complete before merging.

Exclusive Event-based Gateway (instantiate)
Each occurrence of a subsequent event starts a new process instance.

Complex Gateway
Complex merging and branching behavior that is not captured by other gateways.

Parallel Event-based Gateway (instantiate)
The occurrence of all subsequent events starts a new process instance.

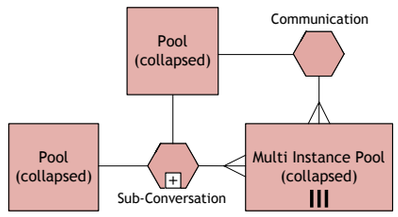
Conversations

A **Communication** defines a set of logically related message exchanges. When marked with a [+] symbol it indicates a Sub-Conversation, a compound conversation element.

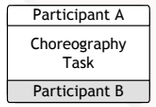
A **Conversation Link** connects Communications and Participants.

A **Forked Conversation Link** connects Communications and multiple Participants.

Conversation Diagram

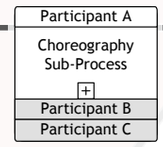


Choreographies



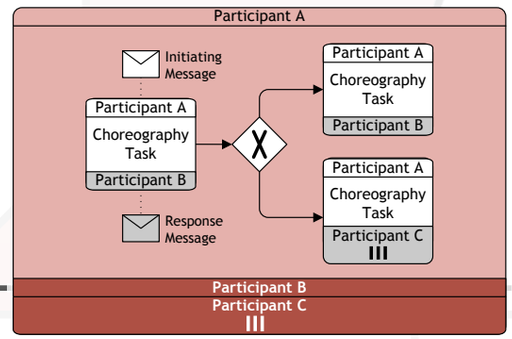
A **Choreography Task** represents an Interaction (Message Exchange) between two Participants.

Multiple Participants Marker denotes a set of Participants of the same kind.

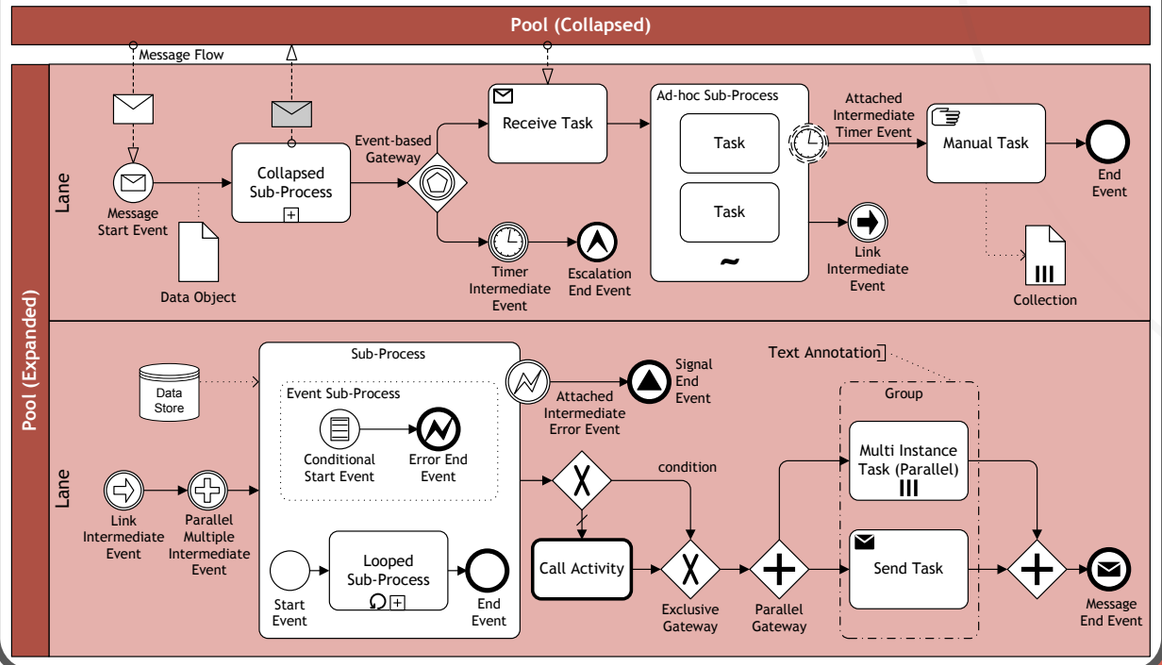


A **Choreography Sub-Process** contains a refined choreography with several Interactions.

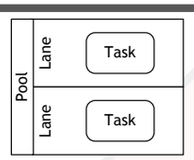
Choreography Diagram



Collaboration Diagram

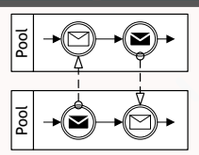


Swimlanes



Pools (Participants) and **Lanes** represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.

Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.



The order of message exchanges can be specified by combining message flow and sequence flow.

Events

	Start: Top-Level	Start: Event Sub-Process Interrupting	Start: Event Sub-Process Non-Interrupting	Intermediate: Catching	Intermediate: Boundary Interrupting	Intermediate: Boundary Non-Interrupting	Intermediate: Throwing	End
None: Untyped events, indicate start point, state changes or final states.	○						◯	●
Message: Receiving and sending messages.	✉	✉	✉	✉	✉	✉	✉	✉
Timer: Cyclic timer events, points in time, time spans or timeouts.	🕒	🕒	🕒	🕒	🕒	🕒		
Escalation: Escalating to an higher level of responsibility.		⌃	⌃		⌃	⌃	▲	▲
Conditional: Reacting to changed business conditions or integrating business rules.	☰	☰	☰	☰	☰	☰		
Link: Off-page connectors. Two corresponding link events equal a sequence flow.				⇨			➡	
Error: Catching or throwing named errors.		↯			↯			↯
Cancel: Reacting to cancelled transactions or triggering cancellation.					⊗			⊗
Compensation: Handling or triggering compensation.		◀◀			◀◀		◀◀	◀◀
Signal: Signalling across different processes. A signal thrown can be caught multiple times.	△	△	△	△	△	△	▲	▲
Multiple: Catching one out of a set of events. Throwing all events defined	⬠	⬠	⬠	⬠	⬠	⬠	⬟	⬟
Parallel Multiple: Catching all out of a set of parallel events.	✚	✚	✚	✚	✚	✚		
Terminate: Triggering the immediate termination of a process.								●

Data

A **Data Input** is an external input for the entire process. It can be read by an activity.

A **Data Output** is a variable available as result of the entire process.

A **Data Object** represents information flowing through the process, such as business documents, e-mails, or letters.

A **Collection Data Object** represents a collection of information, e.g., a list of order items.

A **Data Store** is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

A **Message** is used to depict the contents of a communication between two Participants.

BPM Offensive Berlin

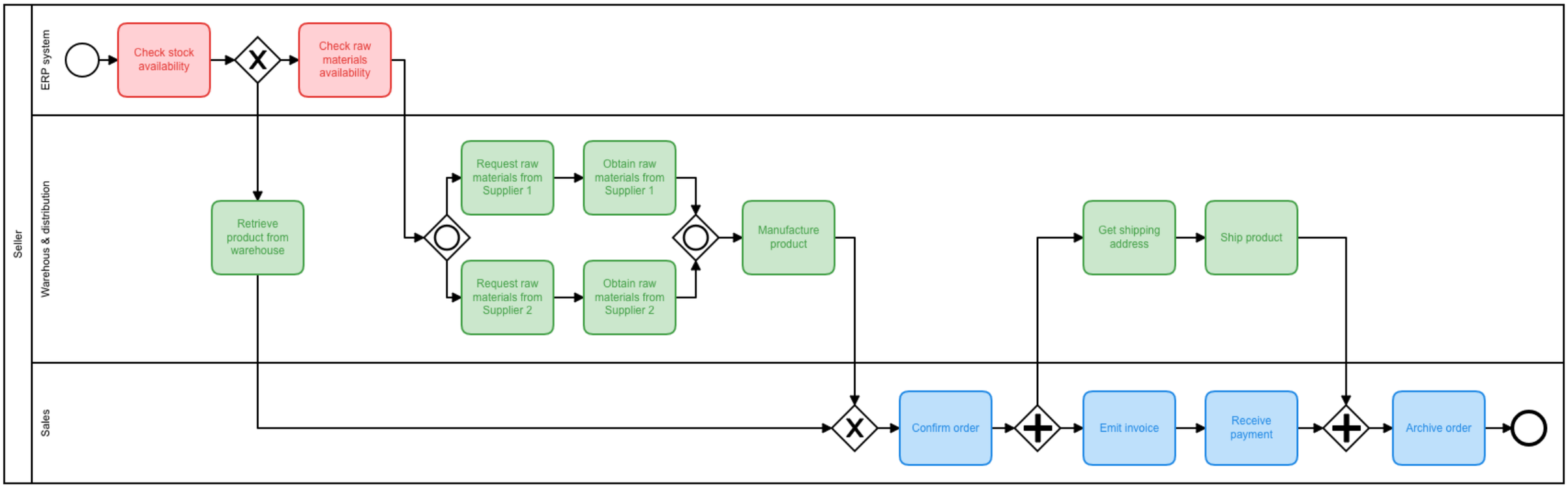
HPI Hasso Plattner Institut

camunda the business process company

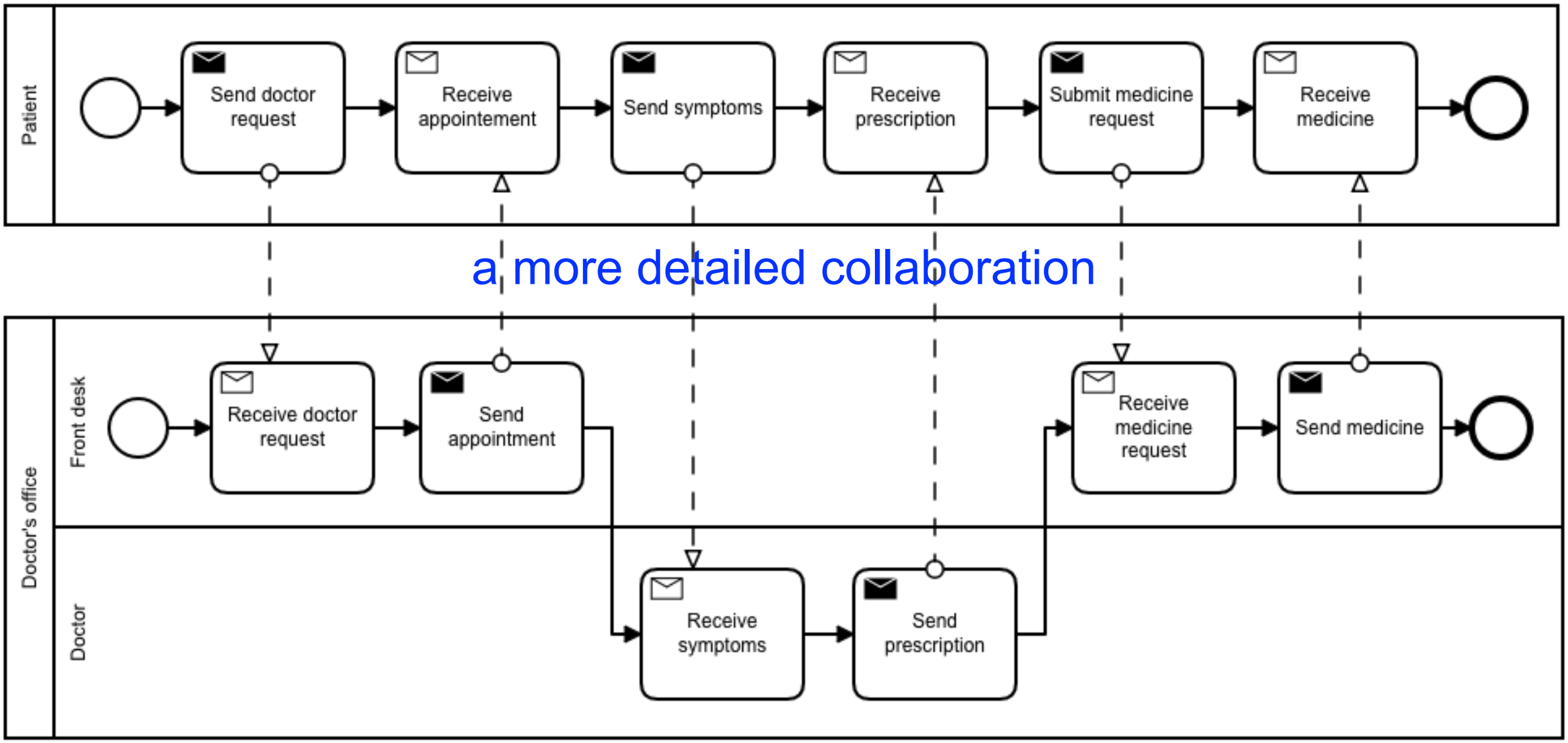
inubit integrating your business and IT

SIGNAVIO simply professional

Resources as lanes: order fulfillment

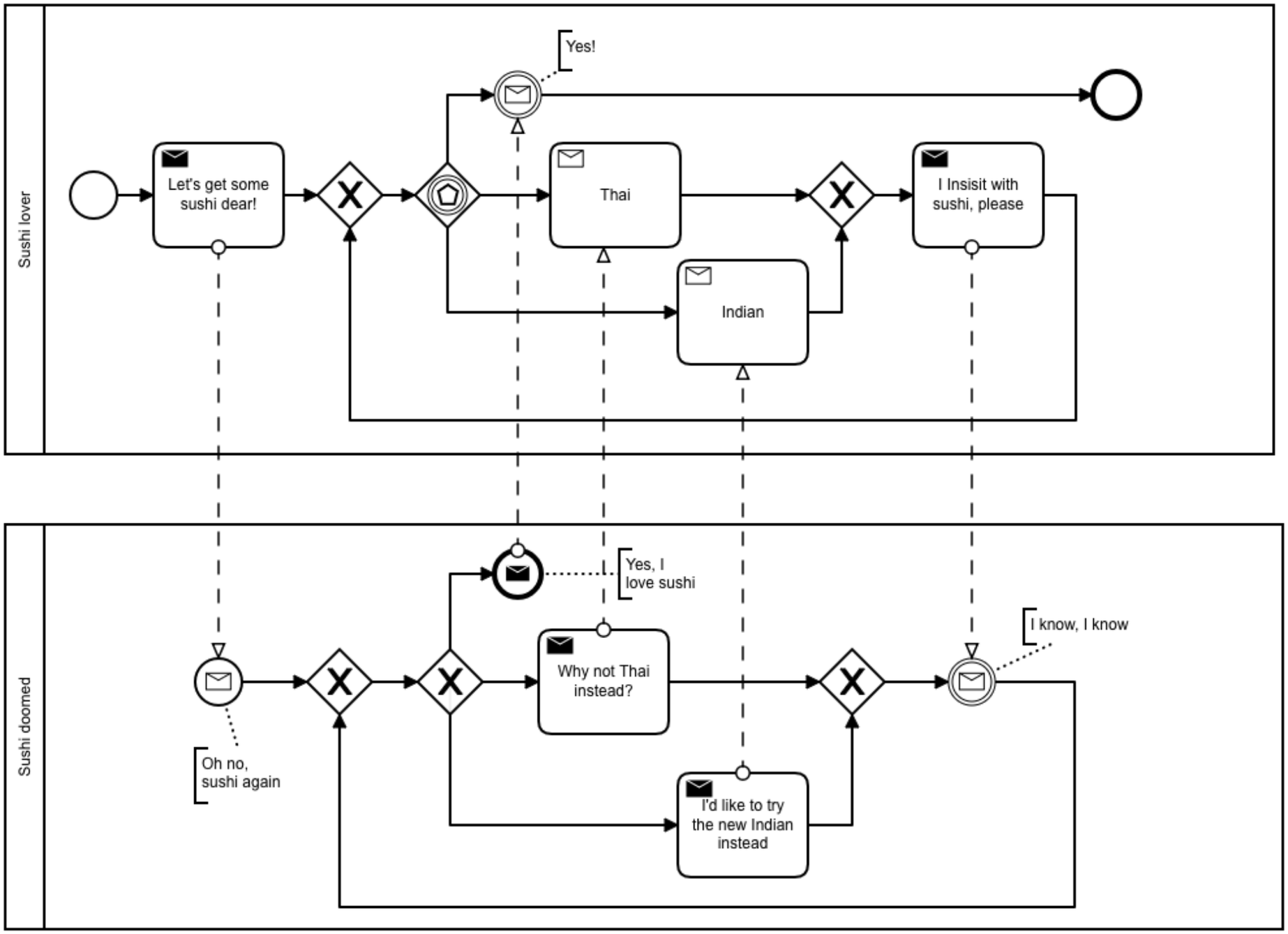


From processes to collaborations



a more detailed collaboration

A negotiation without choice



BPMN Semantics

BPMN formal semantics?

Many attempts:
Abstract State Machines (ASM)
Term Rewriting Systems
Graph Rewrite Systems
Process Algebras
Temporal Logic
…
Petri nets

(Usual difficulties with OR-join semantics)

Sound BPMN diagrams

We can exploit the formal semantics of nets
to give unambiguous semantics to
BPMN process diagrams
BPMN collaboration diagrams

We transform
BPMN process diagrams to wf nets
BPMN collaboration diagrams to wf systems

A BPMN diagram is sound if its net is so

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

Translation of BPMN to Petri nets

From BPMN to Petri nets

Available online at www.sciencedirect.com
ScienceDirect
Information and Software Technology 50 (2008) 1281–1294
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

Semantics and analysis of business process models in BPMN

Remco M. Dijkman [a], Marlon Dumas [b,c], Chun Ouyang [c,*]

[a] *Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, 5600 MB, The Netherlands*
[b] *Institute of Computer Science, University of Tartu, J Liivi 2, Tartu 50409, Estonia*
[c] *Faculty of Information Technology, Queensland University of Technology, G.P.O. Box 2434, Brisbane, Qld 4001, Australia*

14

Simplified BPMN

a start / exception event has just one outgoing flow
and no incoming flow

an end event has just one incoming flow
and no outgoing flow

all activities and intermediate events have exactly
one incoming flow and one outgoing flow

all gateways have either
one incoming flow (and multiple outgoing)
or one outgoing flow (and multiple incoming)

Simplified BPMN

The previous constraints are no real limitation:

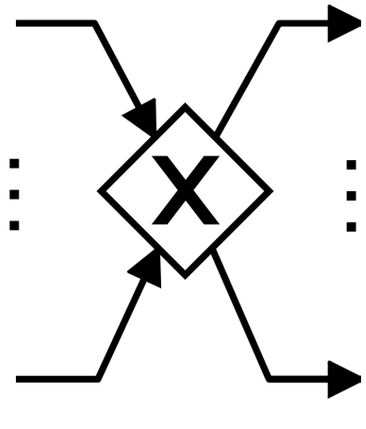
events or activities with multiple incoming flows:
insert a preceding XOR-join gateway

events or activities with multiple outgoing flows:
insert a following AND-split gateway

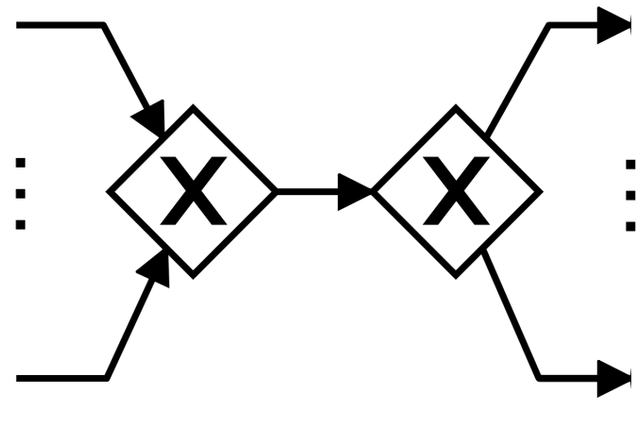
gateways with multiple incoming and outgoing flows:
decompose in two gateways

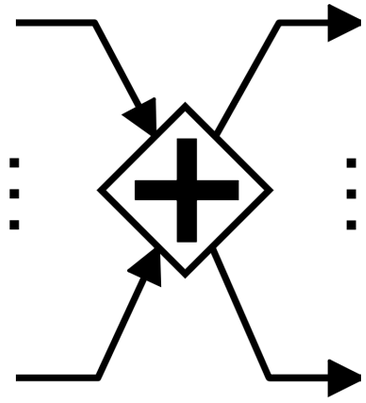
insert start / end events if needed

Pay attention to gateways

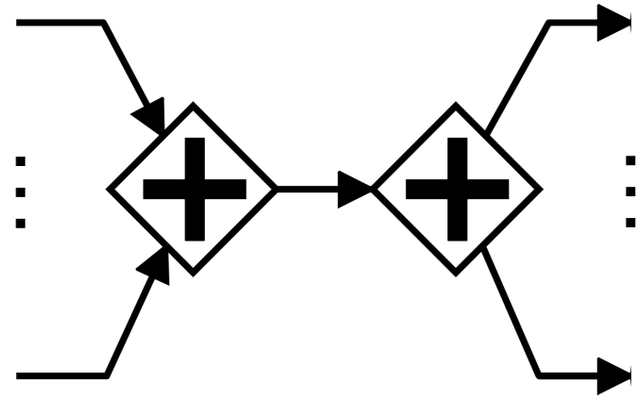


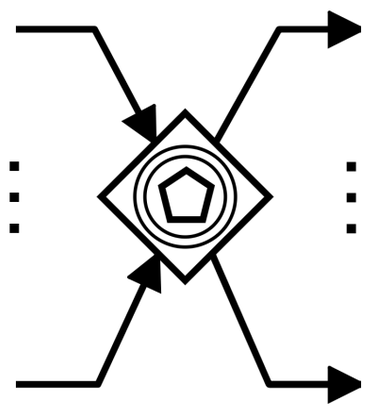
stands for



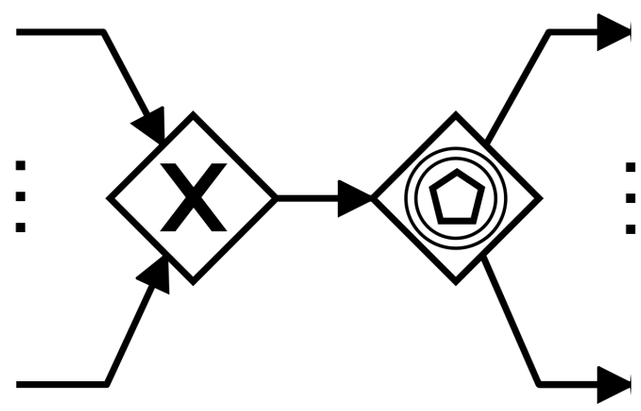


stands for

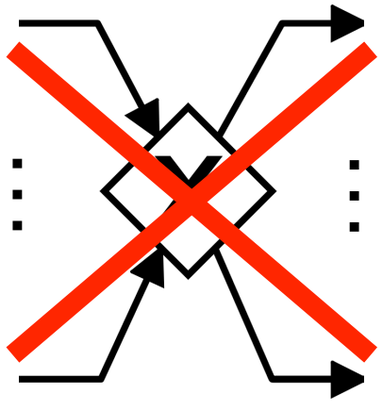




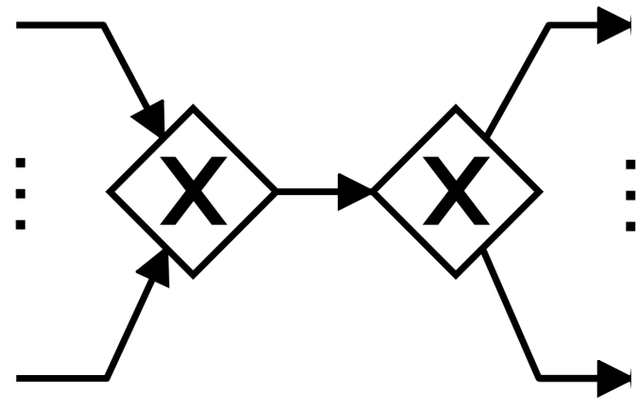
stands for

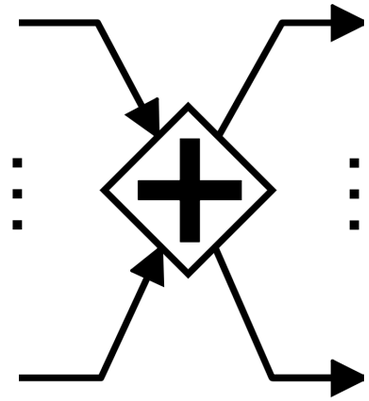


My suggestions

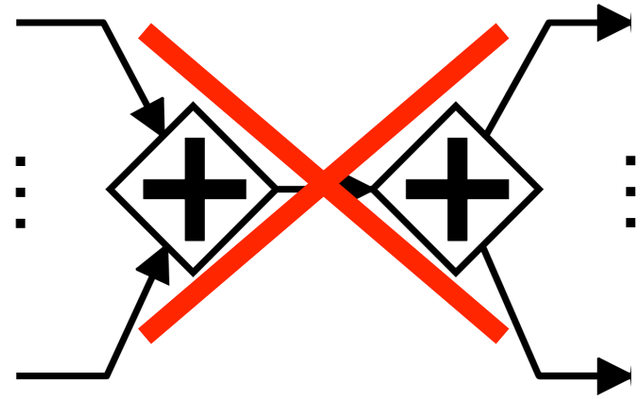


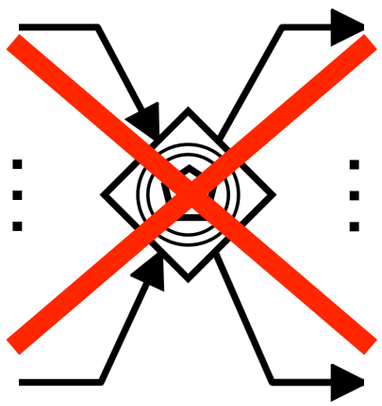
stands for



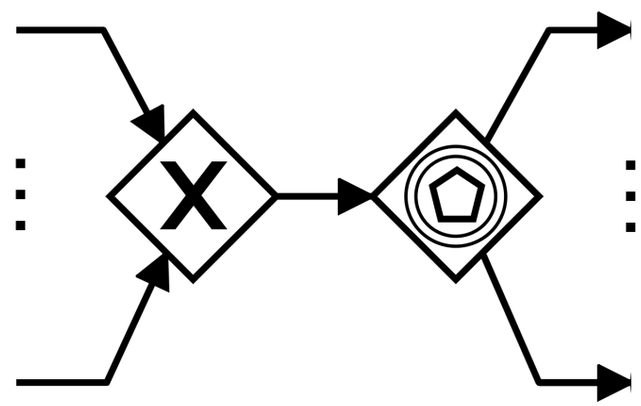


stands for



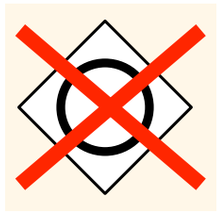


stands for



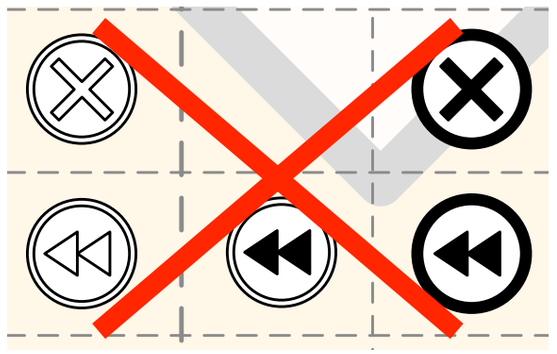
Simplified BPMN

Avoid OR-gateways
(all problems seen with EPC apply to BPMN as well)



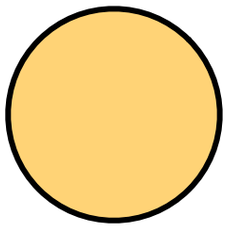
Limited form of sub-processing

No transactions and compensations

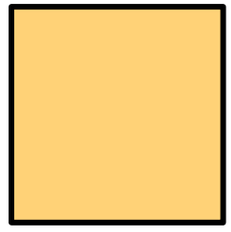


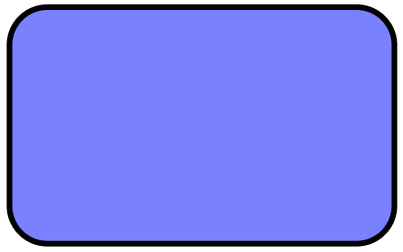
The twist!

BPMN object **net fragment**

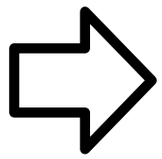


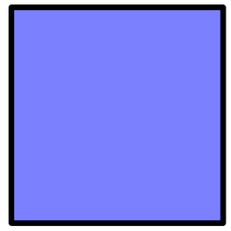
event

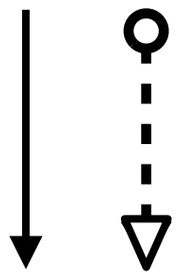
transition 



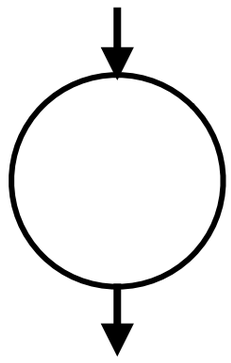
activity



transition 



sequence flow
message flow

place 

Roughly

A place for each arc

one transitions for each event

one transition for each activity

one or two transitions for each gateway

…

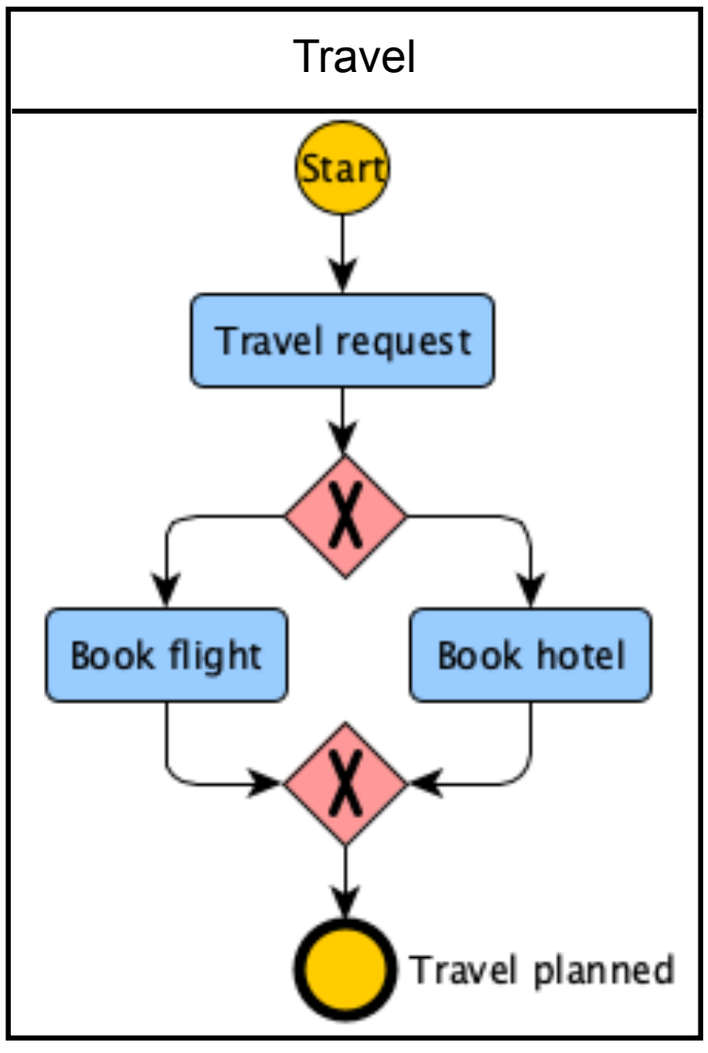
with some exceptions!

(start event, end event, event-based gateways, loops, …)

no dummy objects!

The strategy

From BPMN process diagrams to wf nets in three steps



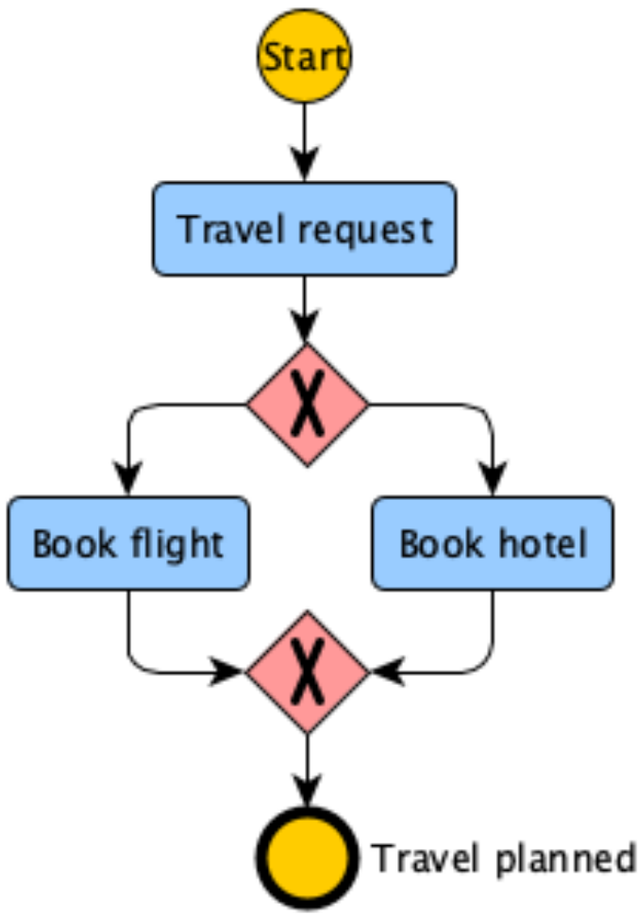
Step 1
convert
sequence flow
message flow

Step 2
convert
flow objects

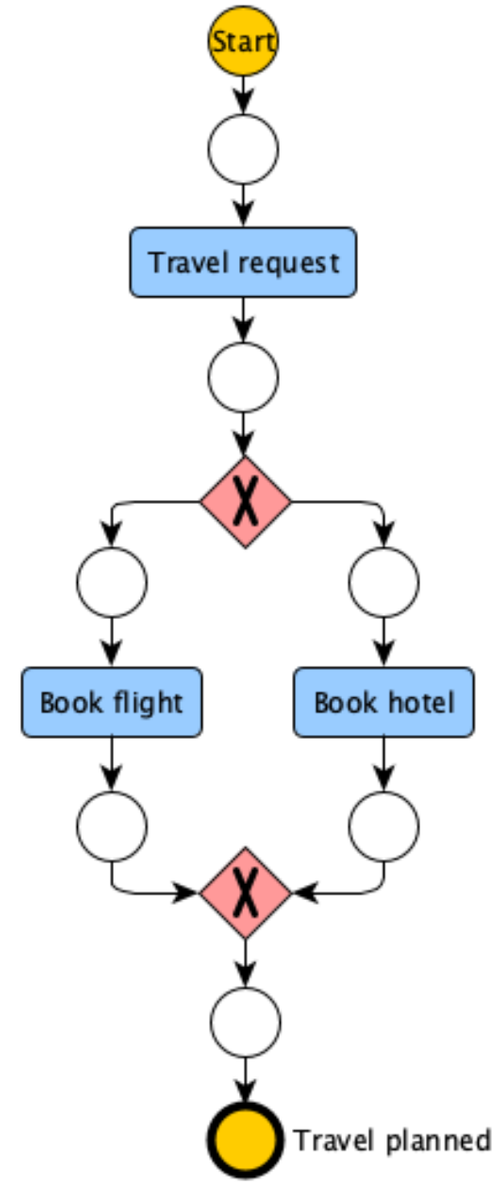
Step 3
enforce
initial place
final place

Step 1: convert flows

We insert a place for each sequence flow and message flow

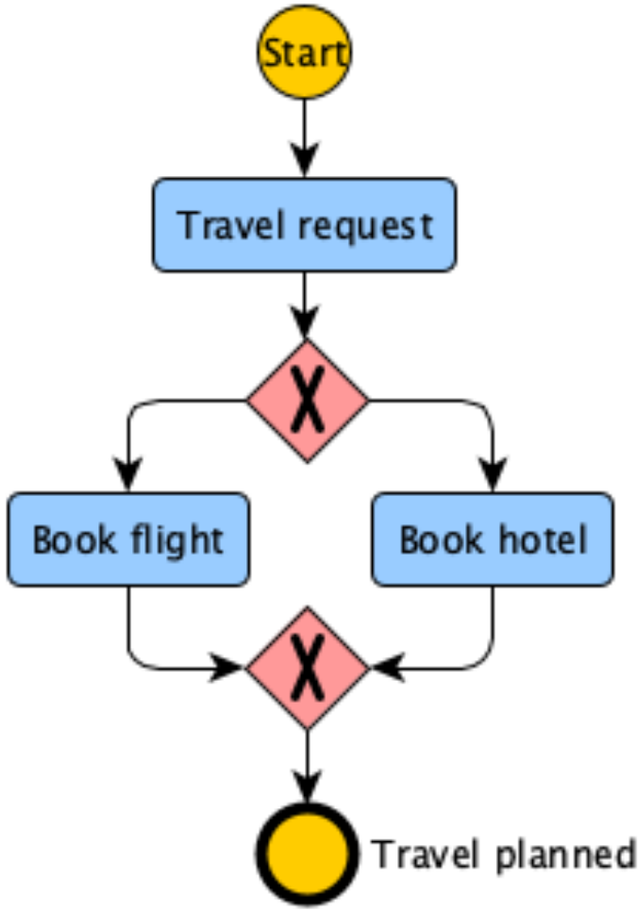


Step 1
sequence flow
message flow

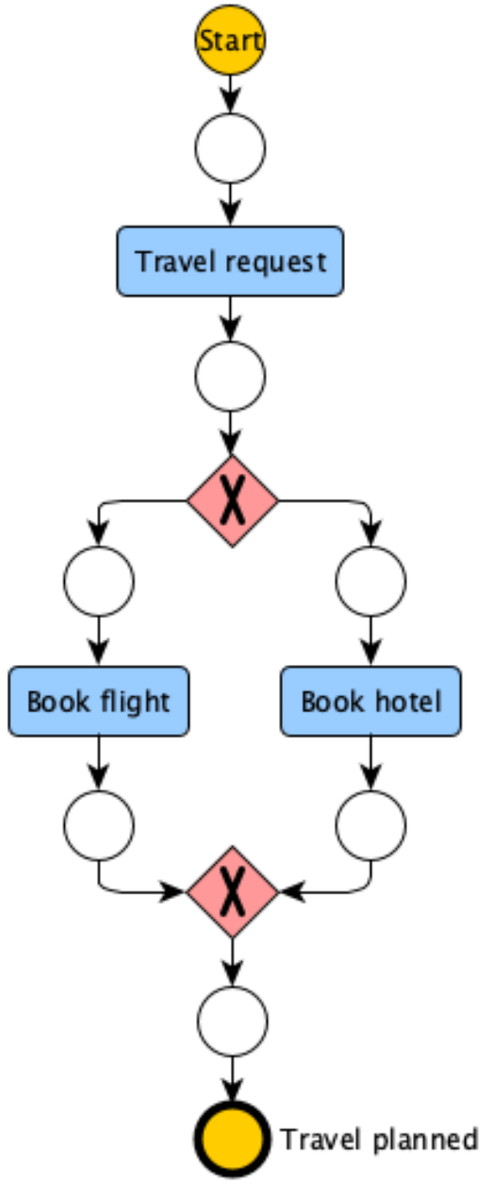


Step 2: convert flow objects

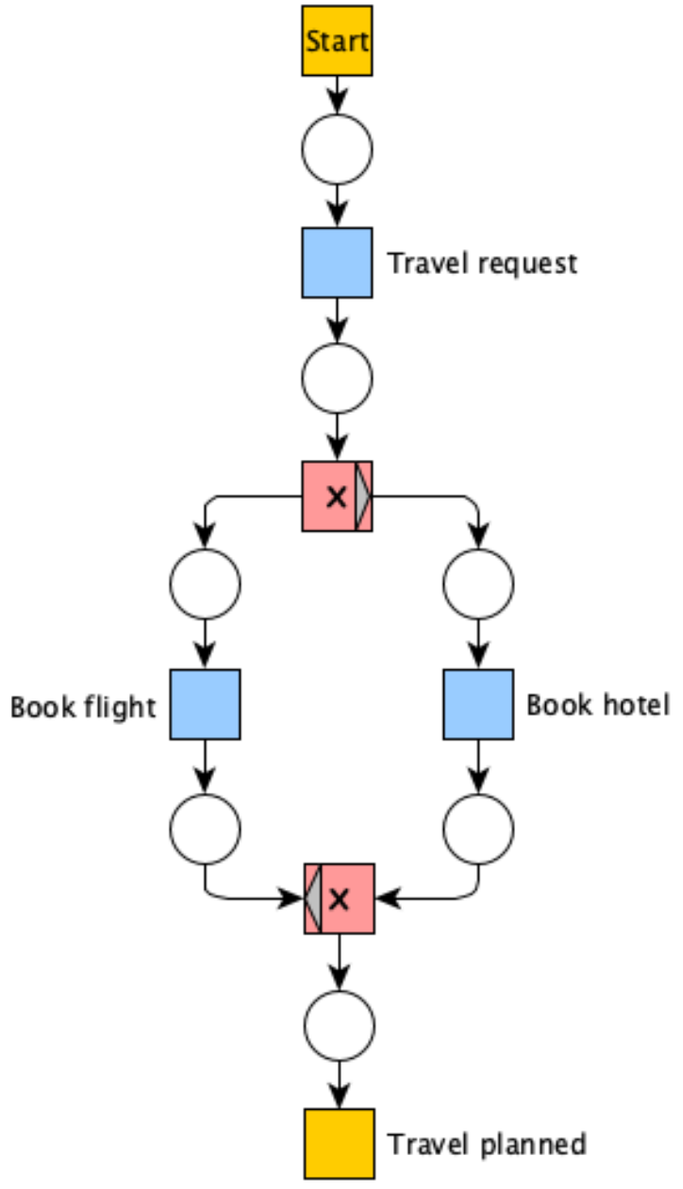
Then insert transitions



Step 1
sequence flow
message flow

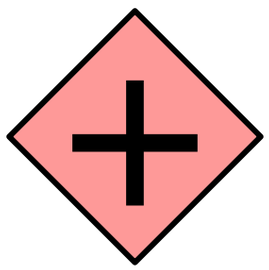


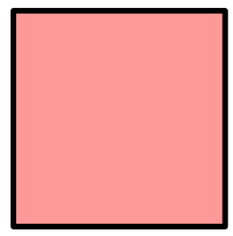
Step 2
flow objects

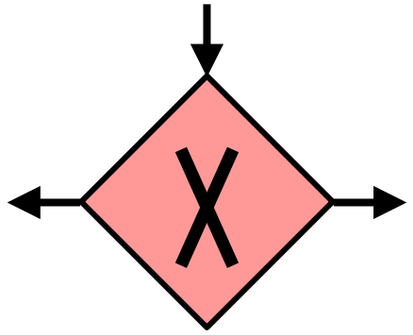


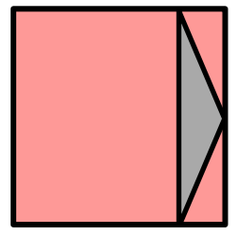
Step 2: gateways

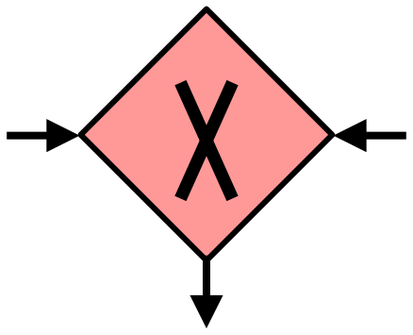
BPMN object **net fragment**

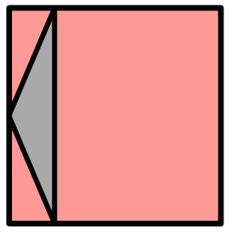


AND split / join transition 



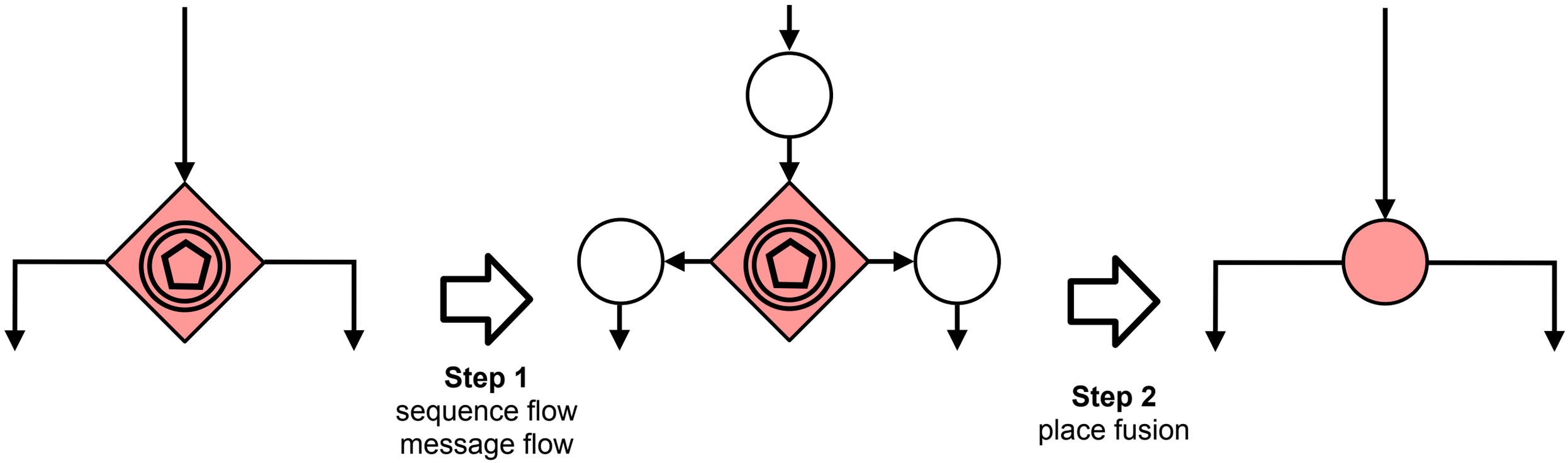
XOR split ⇨ transition 



XOR join transition 

Step 2: event-based

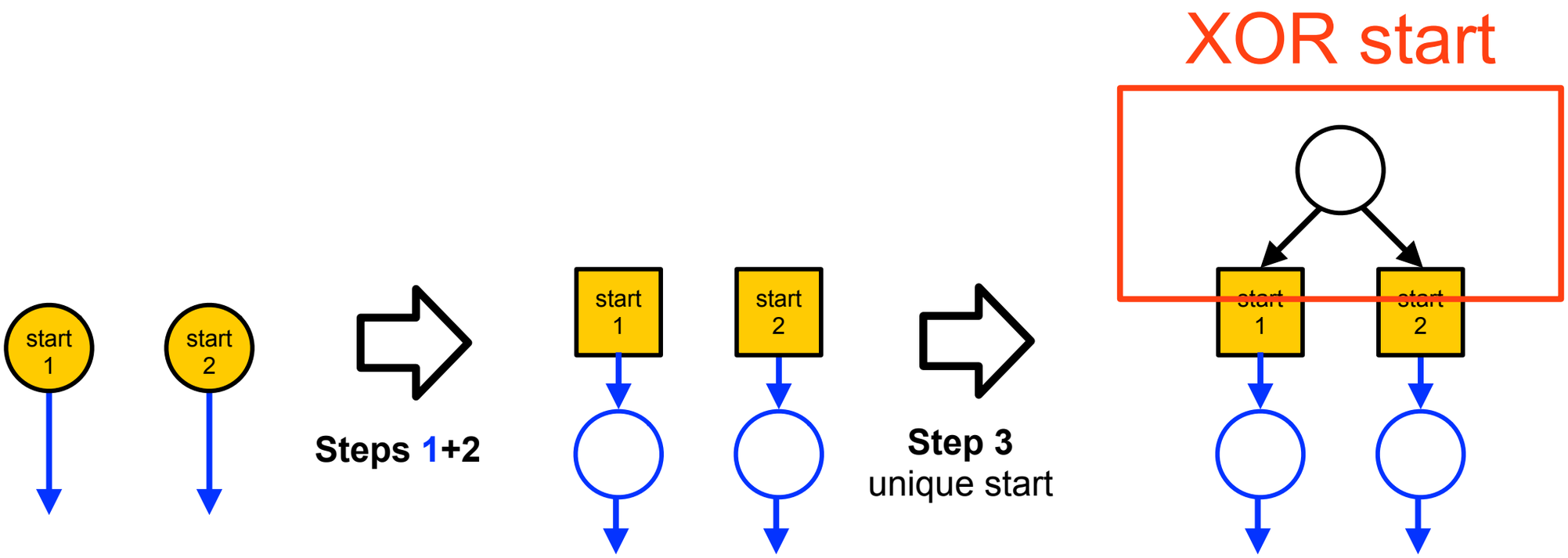
BPMN object **net fragment**



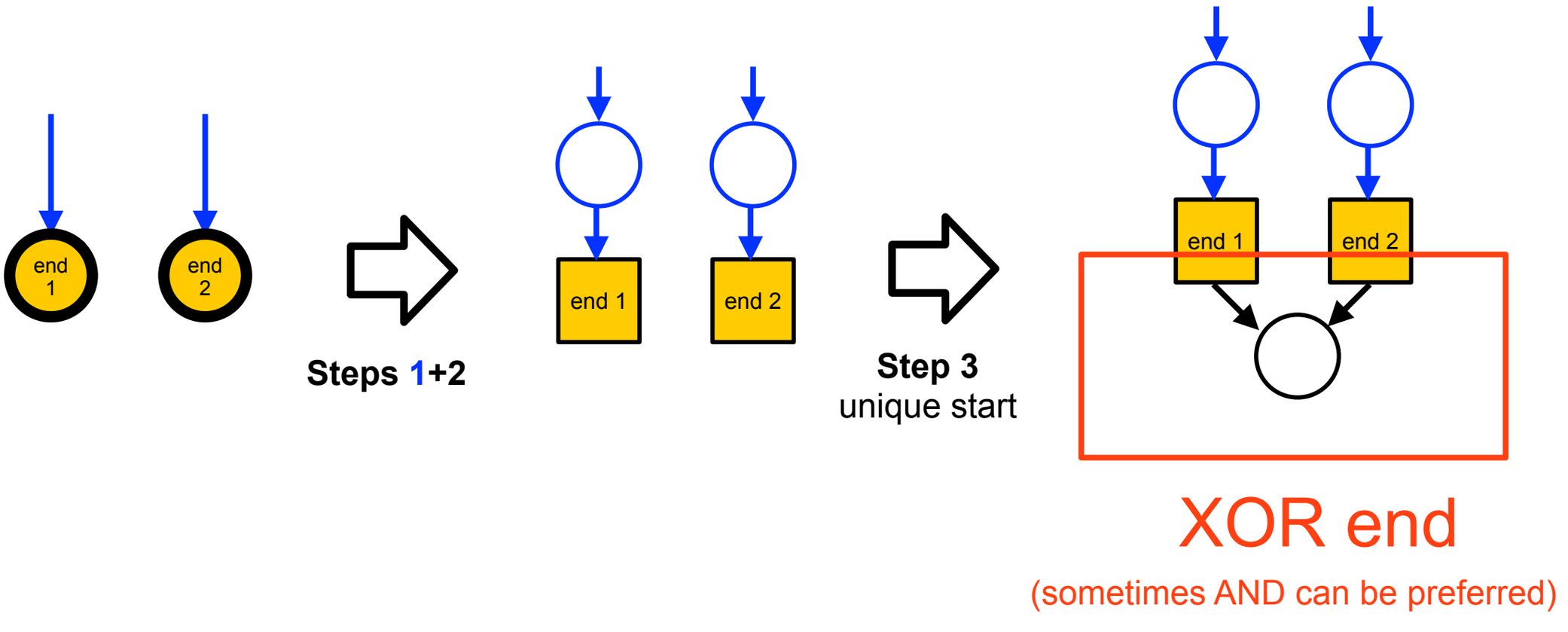
Step 1
sequence flow
message flow

Step 2
place fusion

Step 3: add unique start



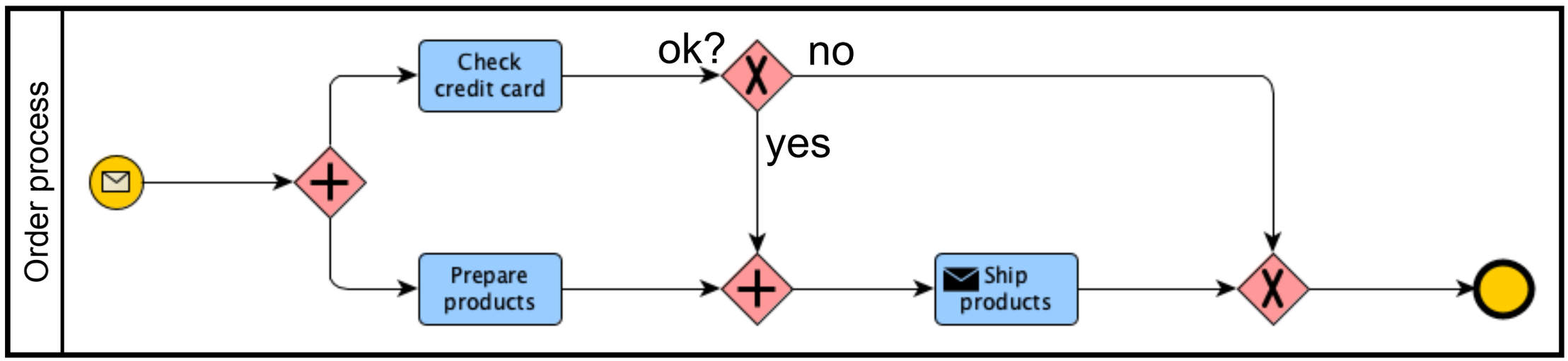
Step 3: add unique end



Example:

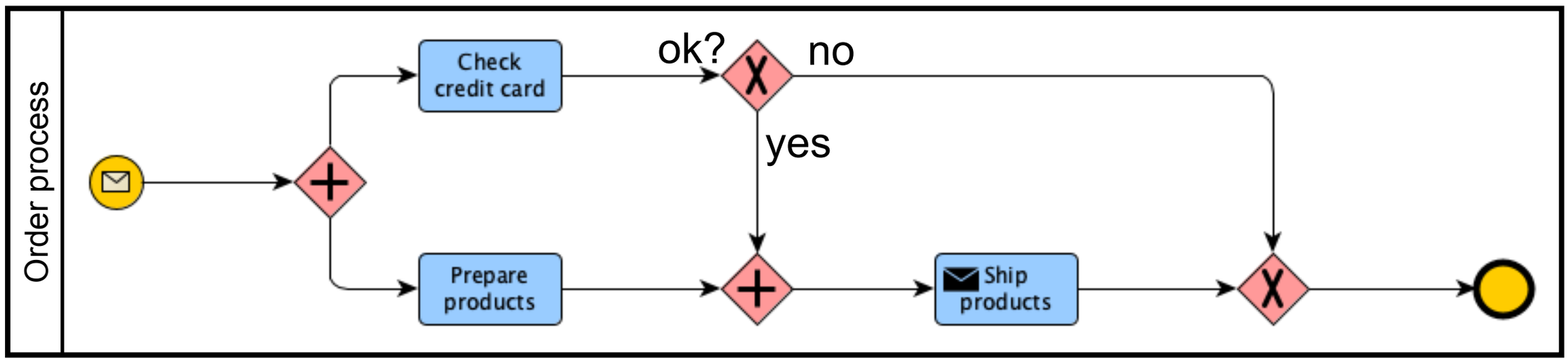
Order process

Order process

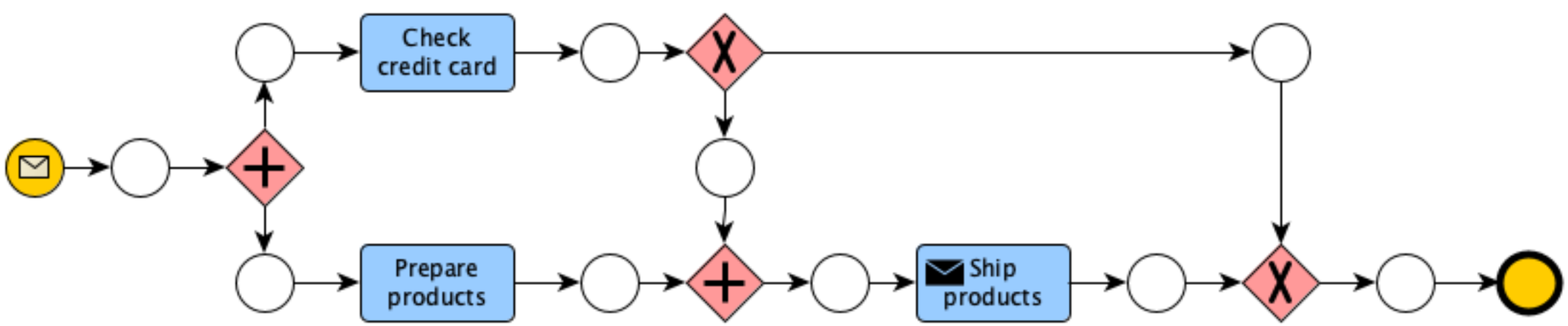


Sound?

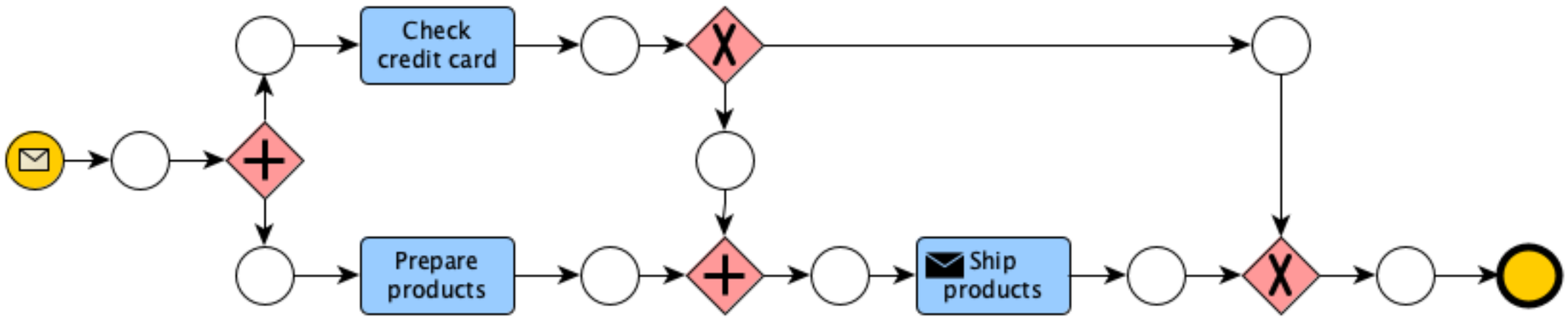
Order process: step 1

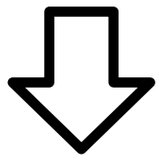


Step 1
sequence flow
message flow

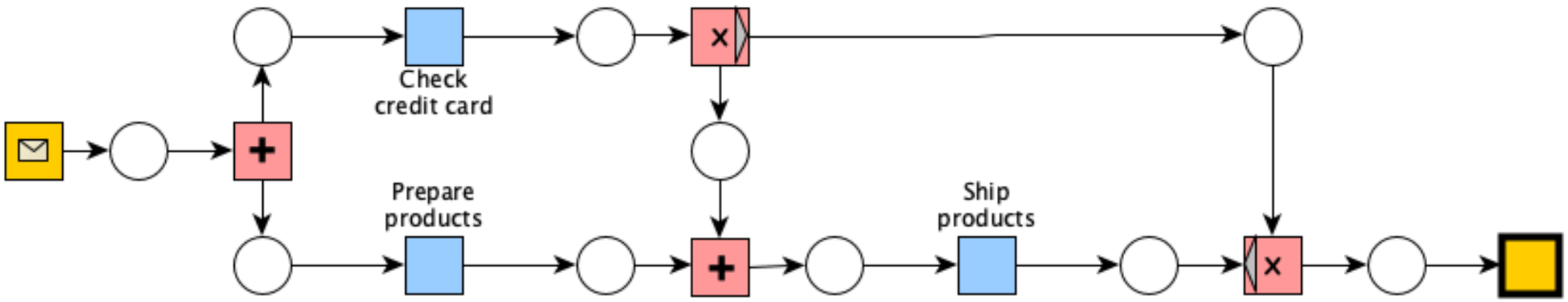


Order process: step 2

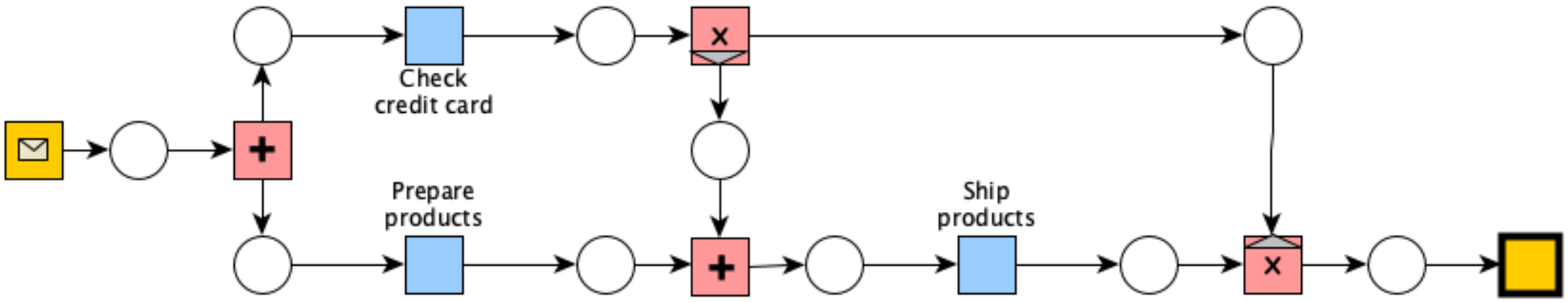


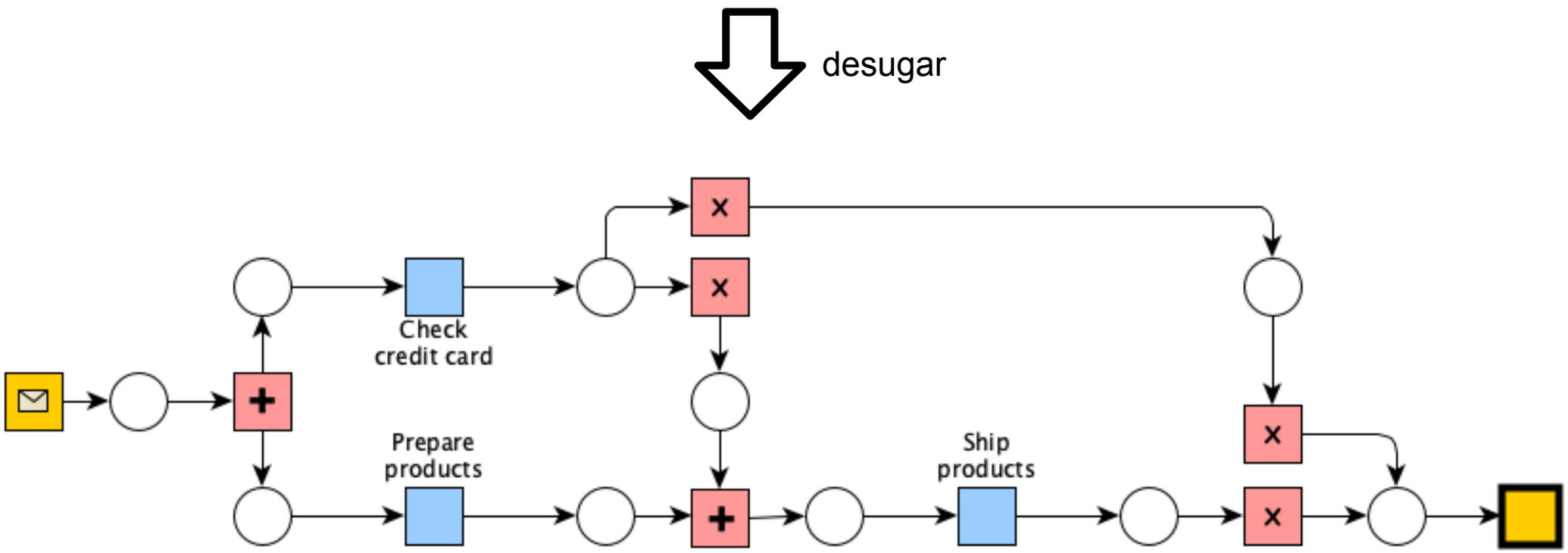


Step 2
flow objects

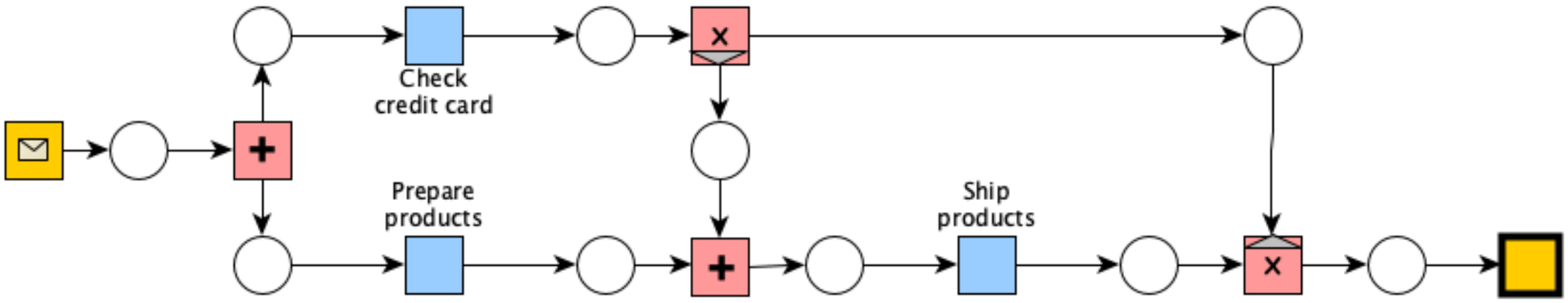


Order process: (desugar)

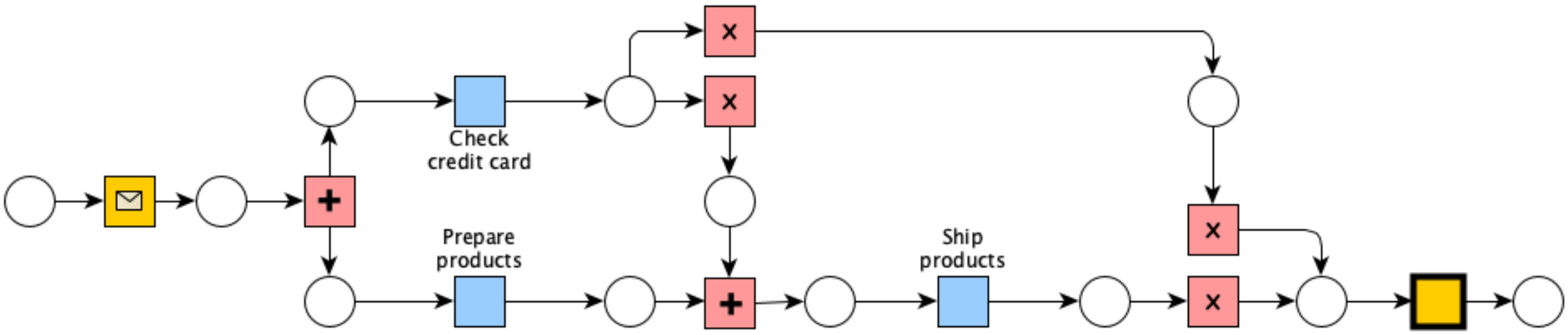




Order process: step 3

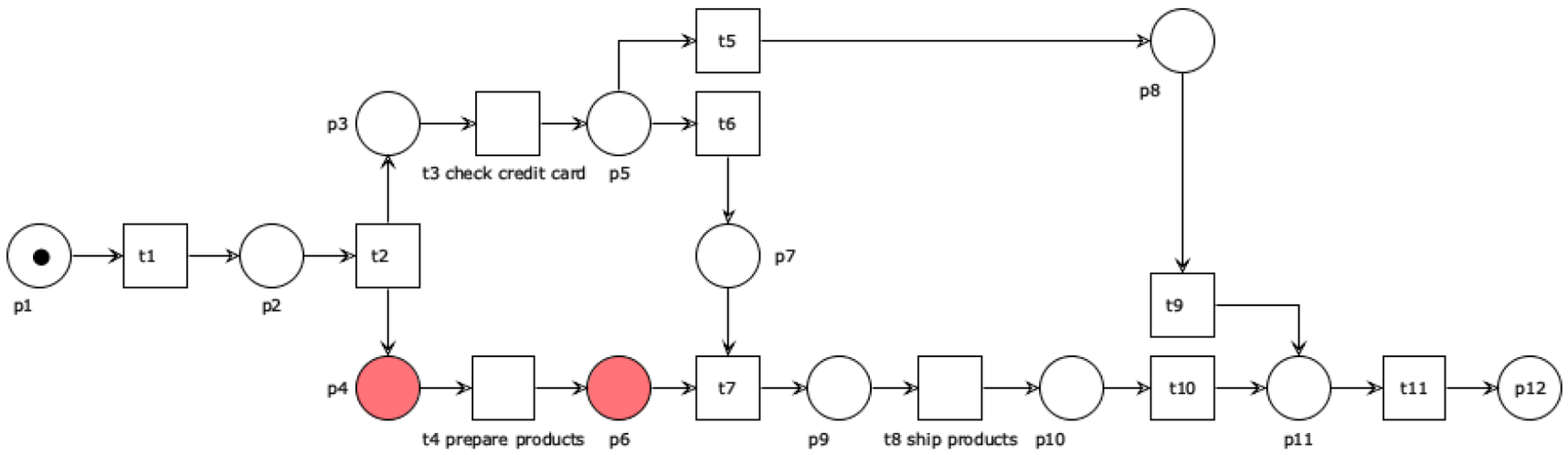


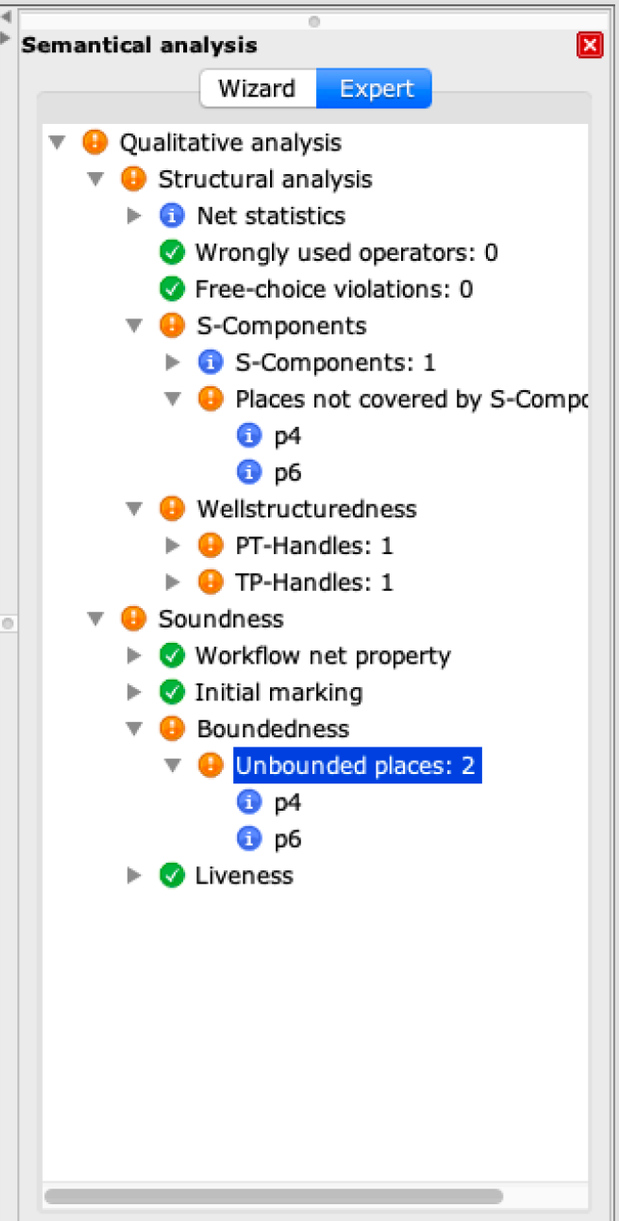
Step 3
enforce
initial place
final place



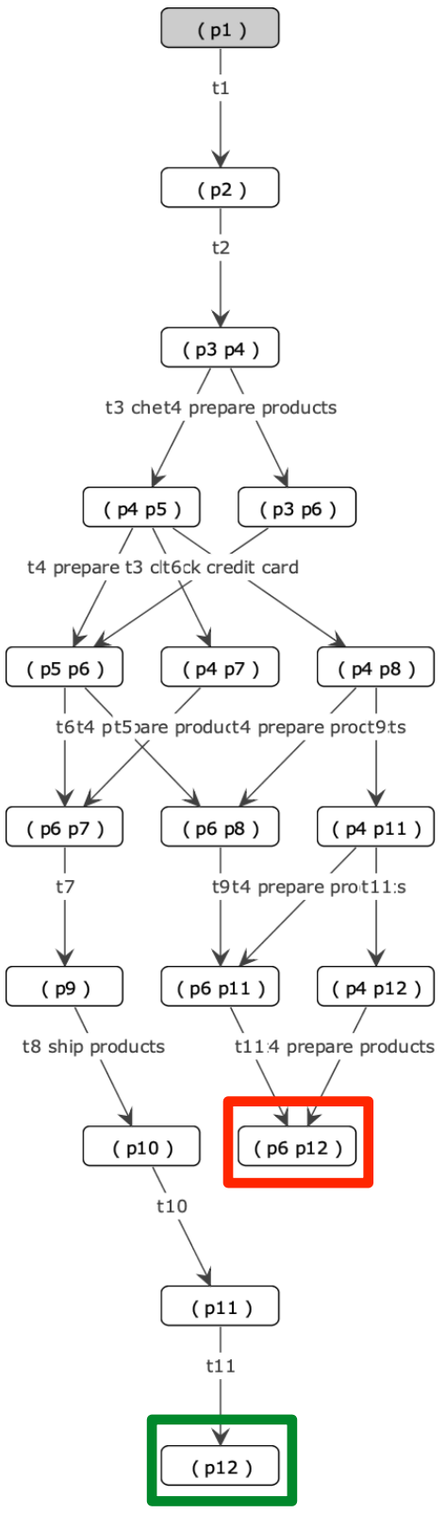
Soundness analysis

Not sound!

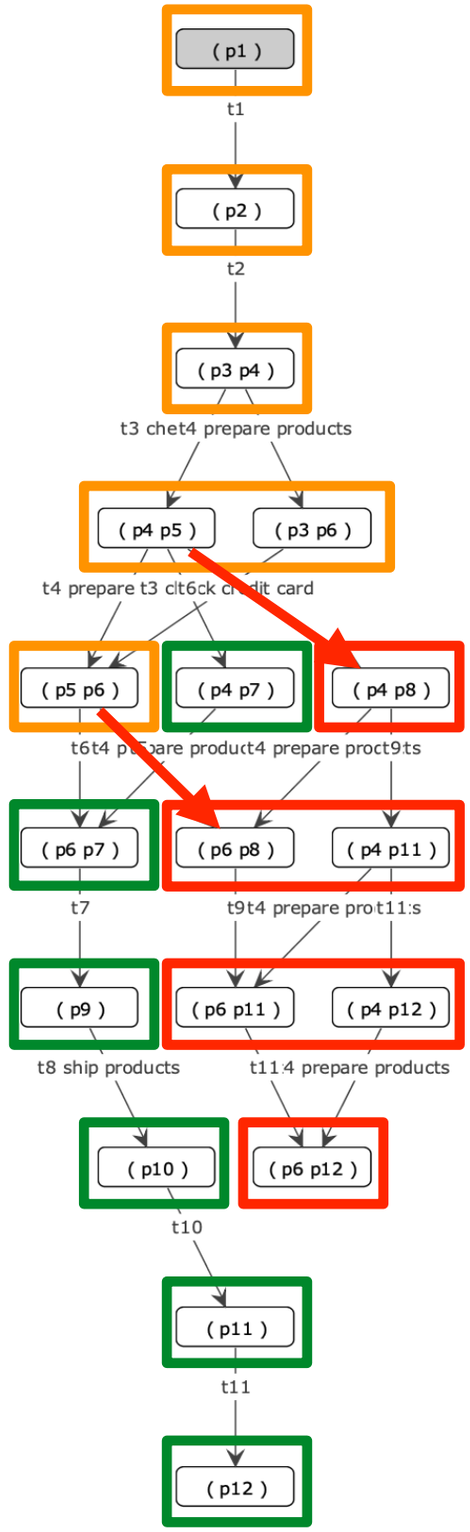




Soundness analysis



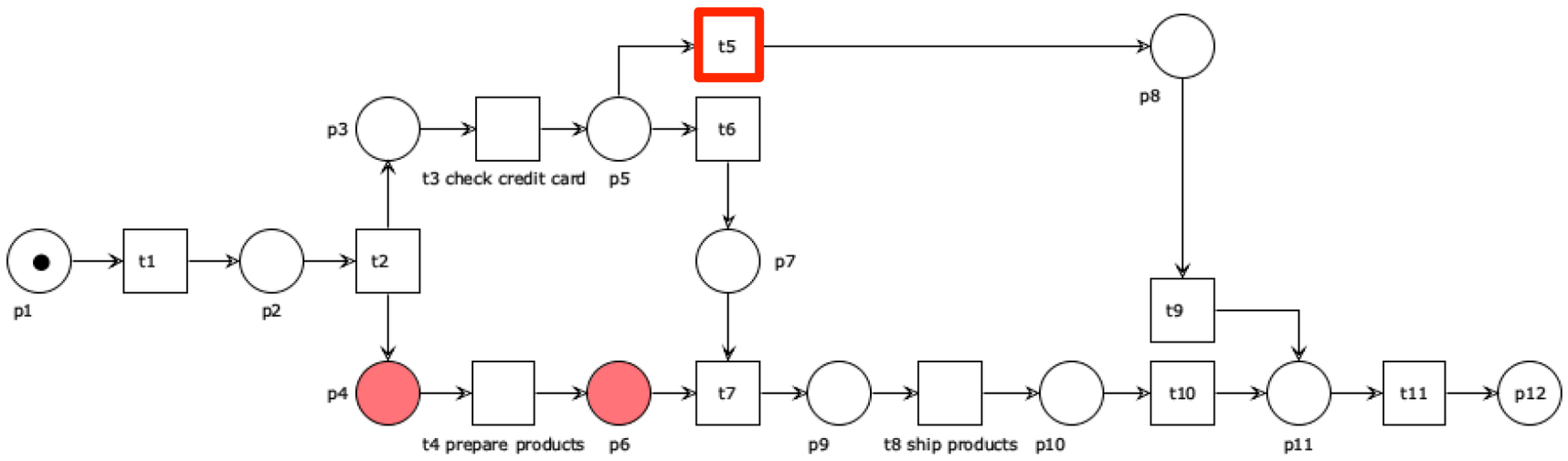
Soundness analysis

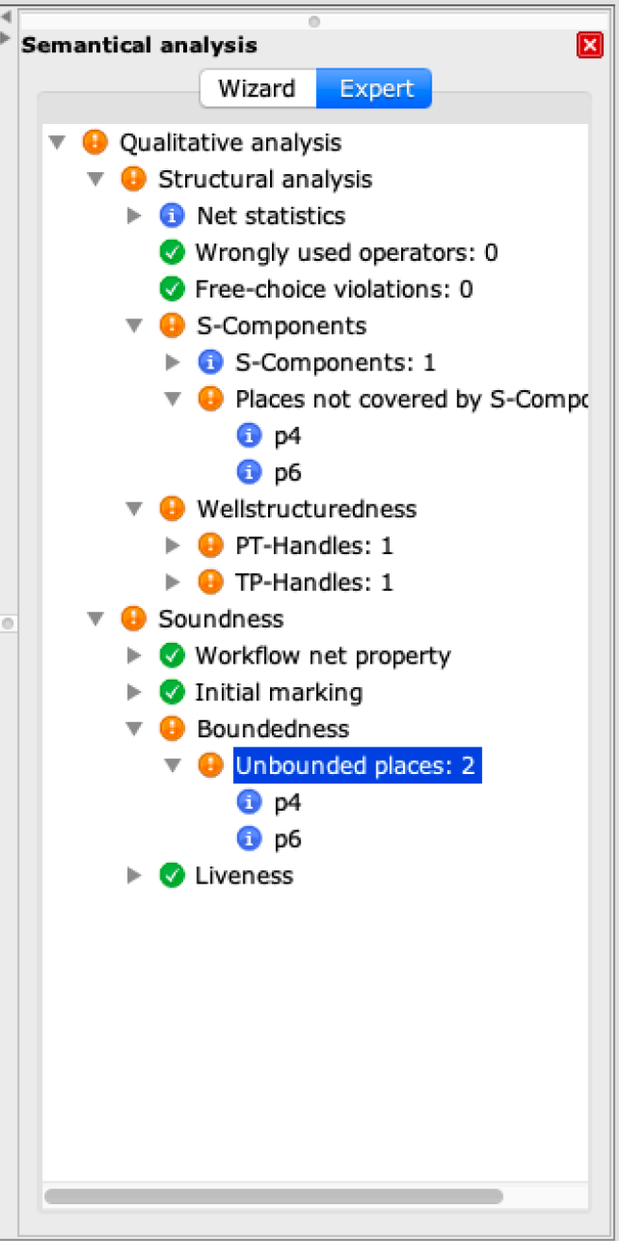


t5

Soundness analysis

Not sound!

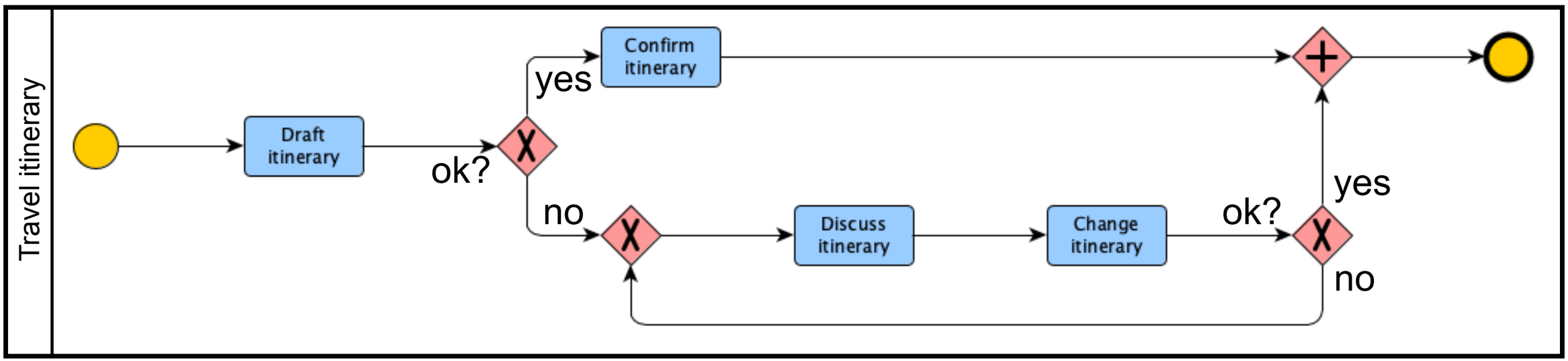




Example:

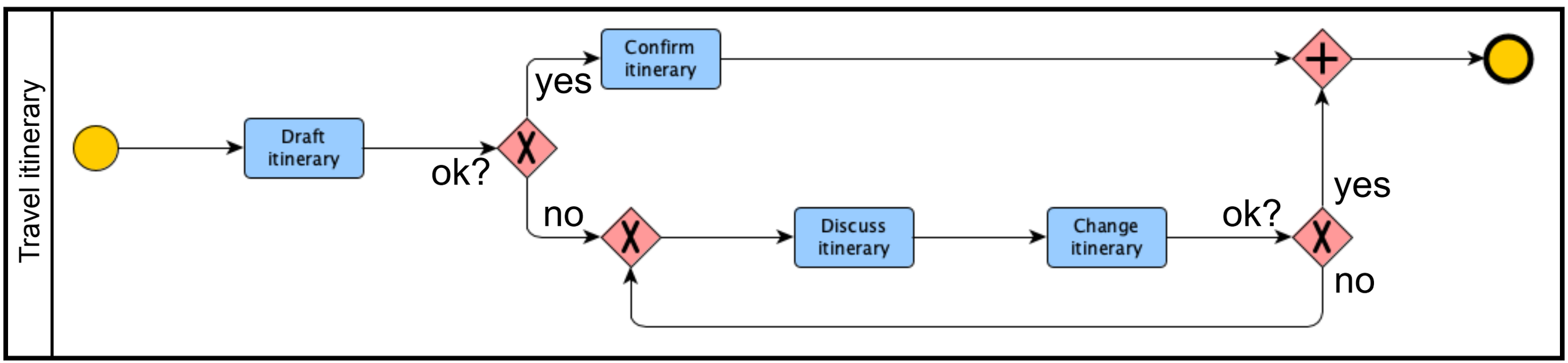
Travel itinerary

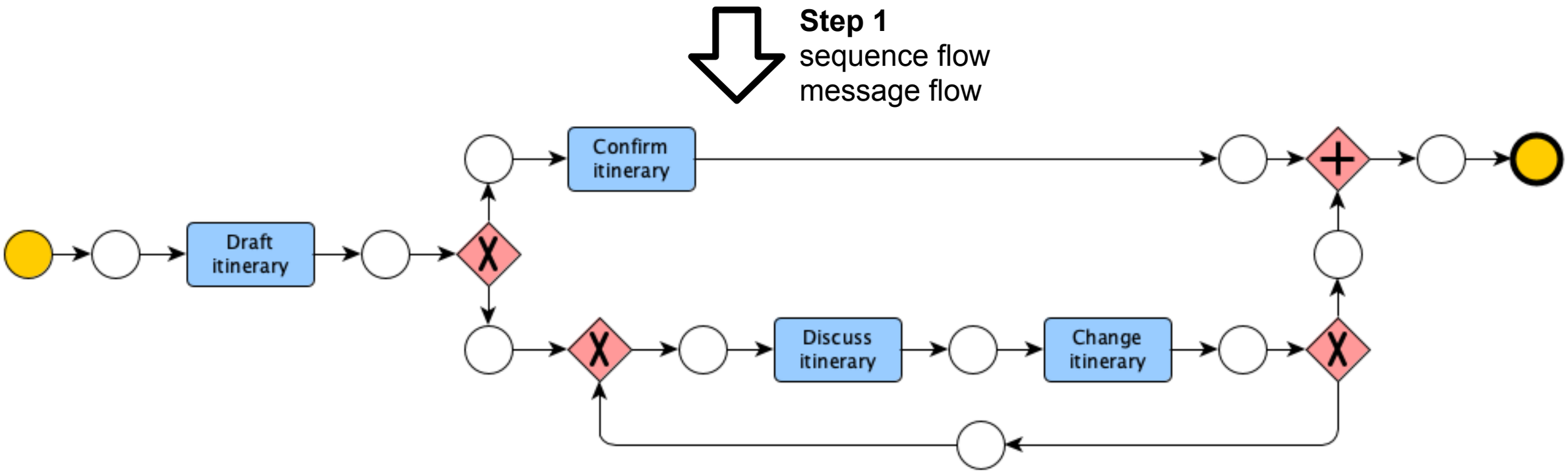
Travel itinerary



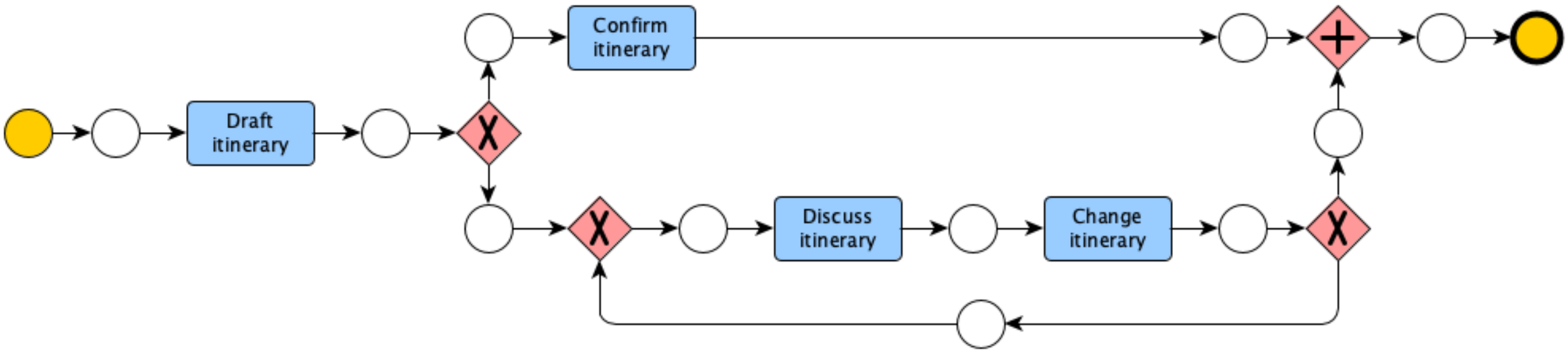
Sound?

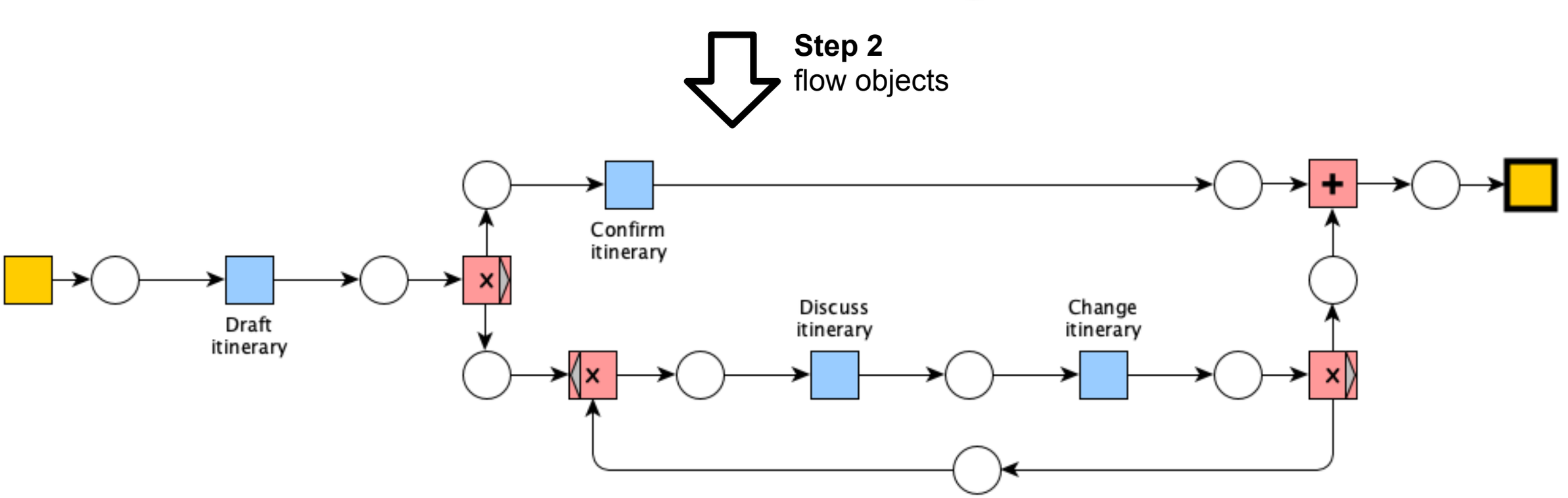
Travel itinerary: step 1



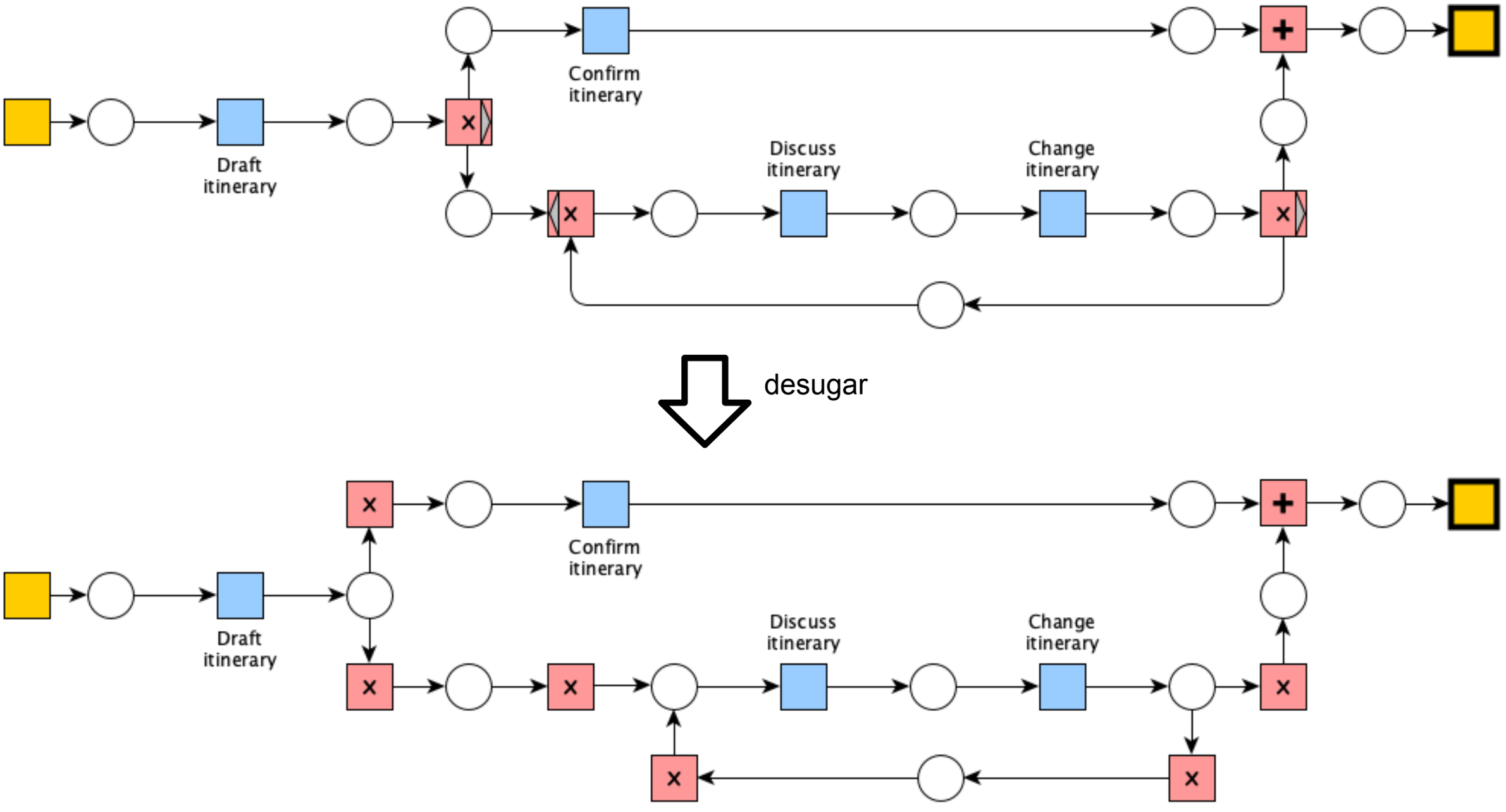


Travel itinerary: step 2

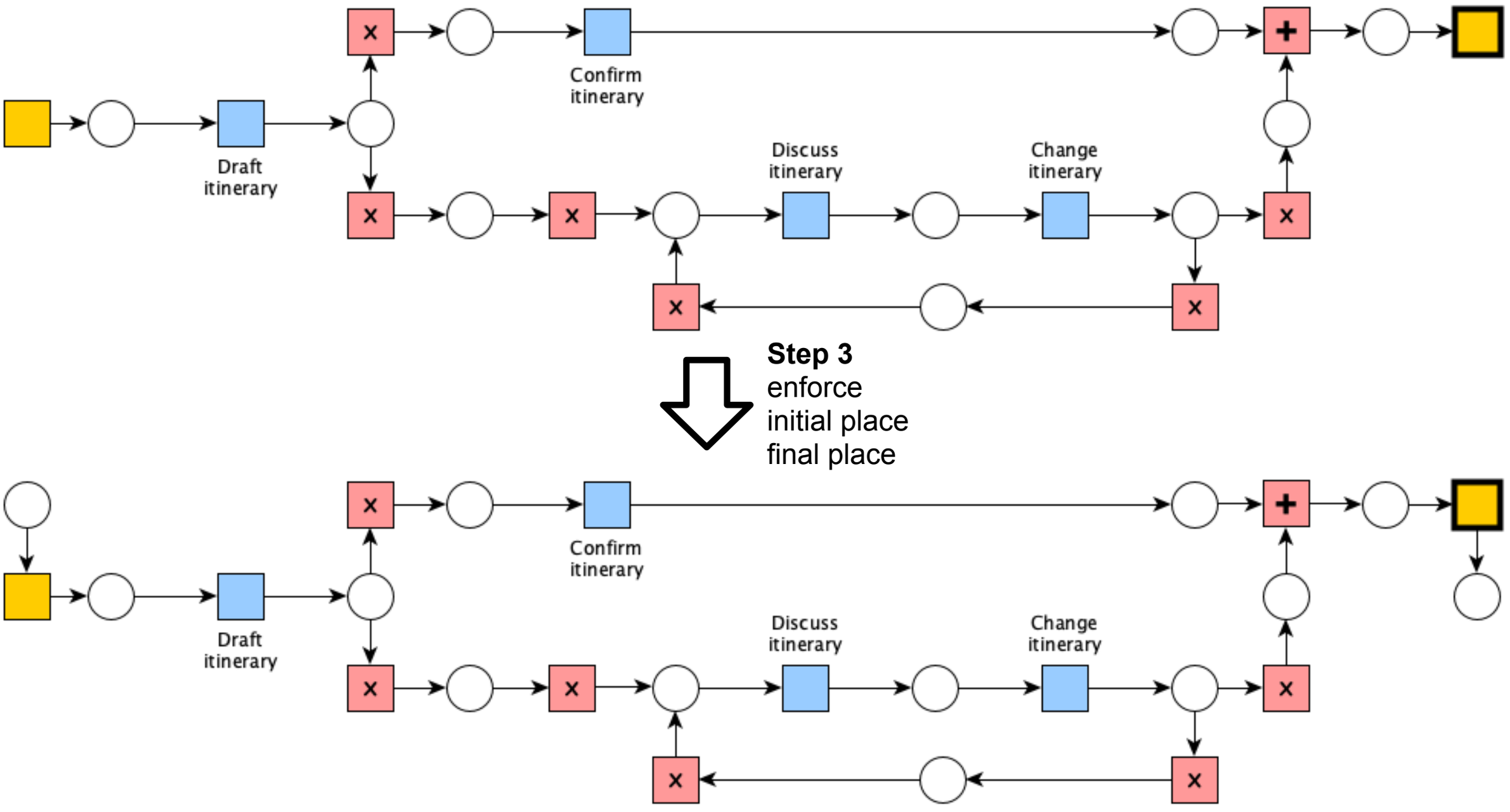




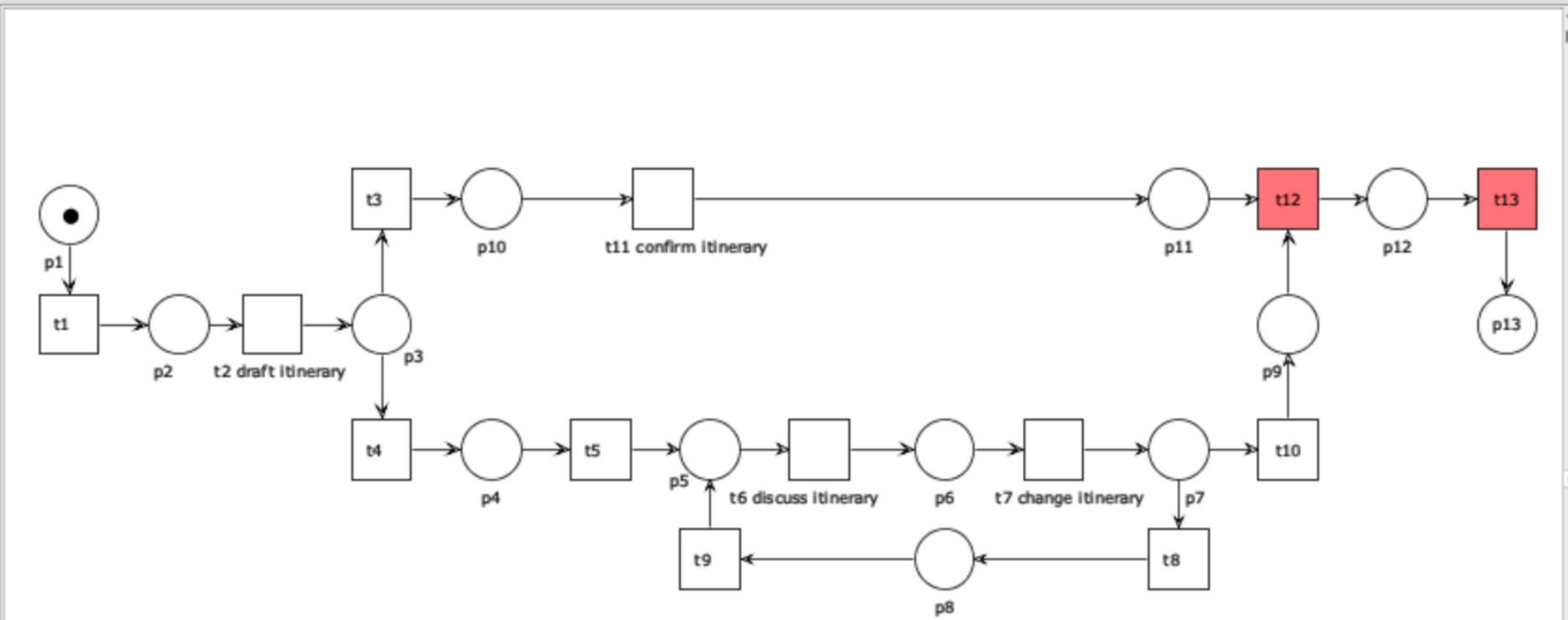
Travel itinerary: (desugar)



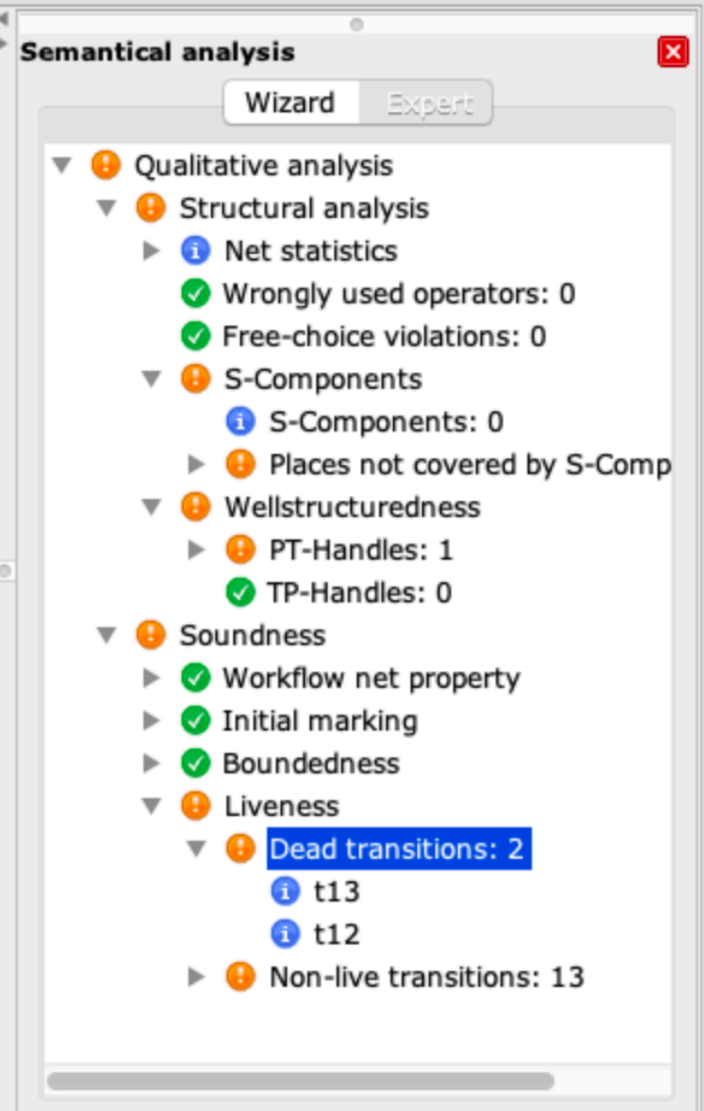
Travel itinerary: step 3



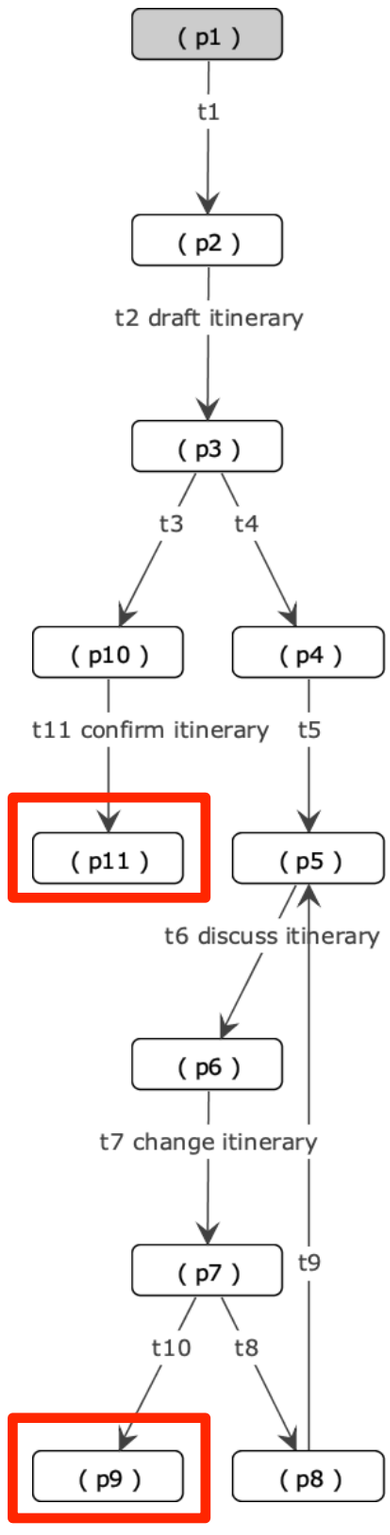
Soundness analysis



Not sound!



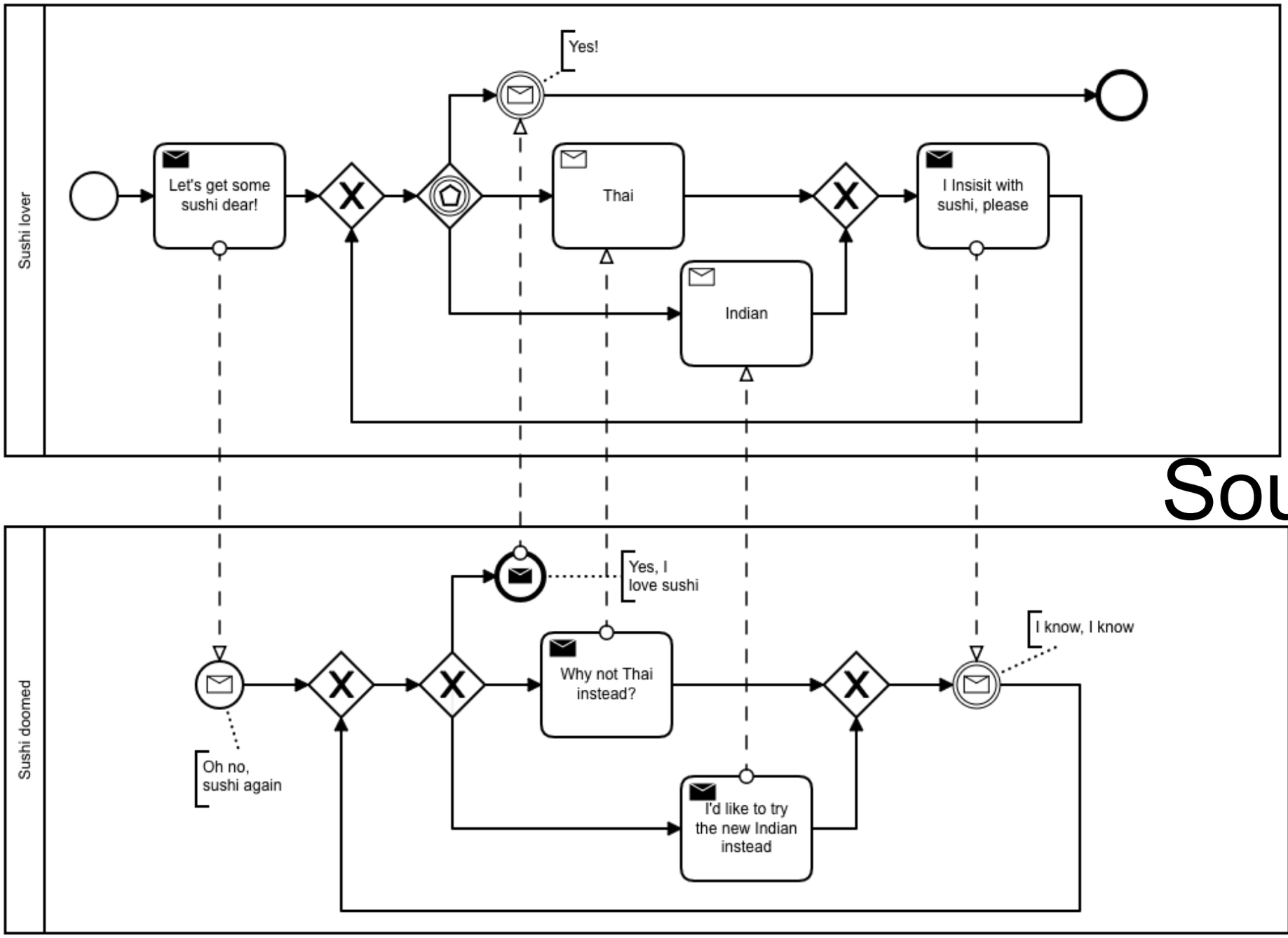
Soundness analysis



Example:

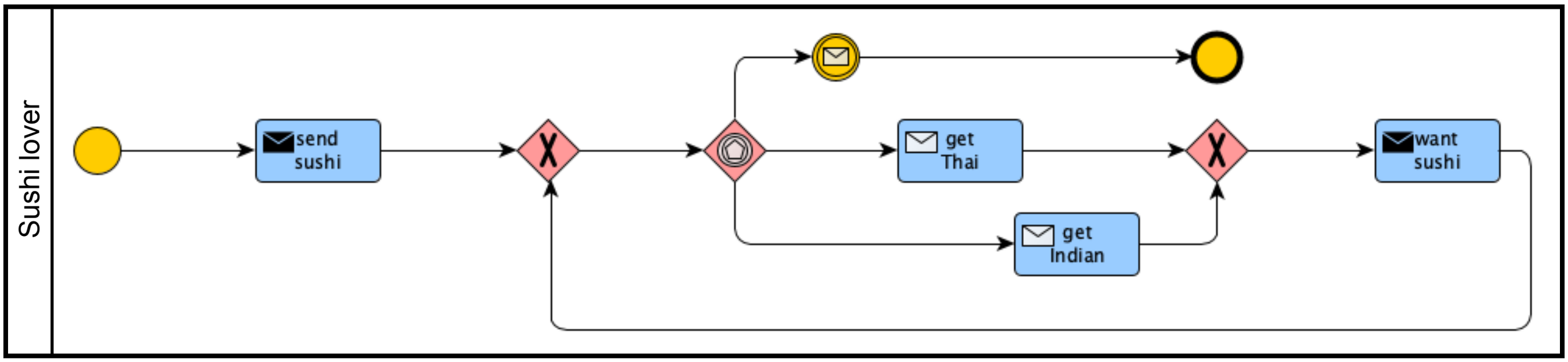
Always sushi

Always sushi



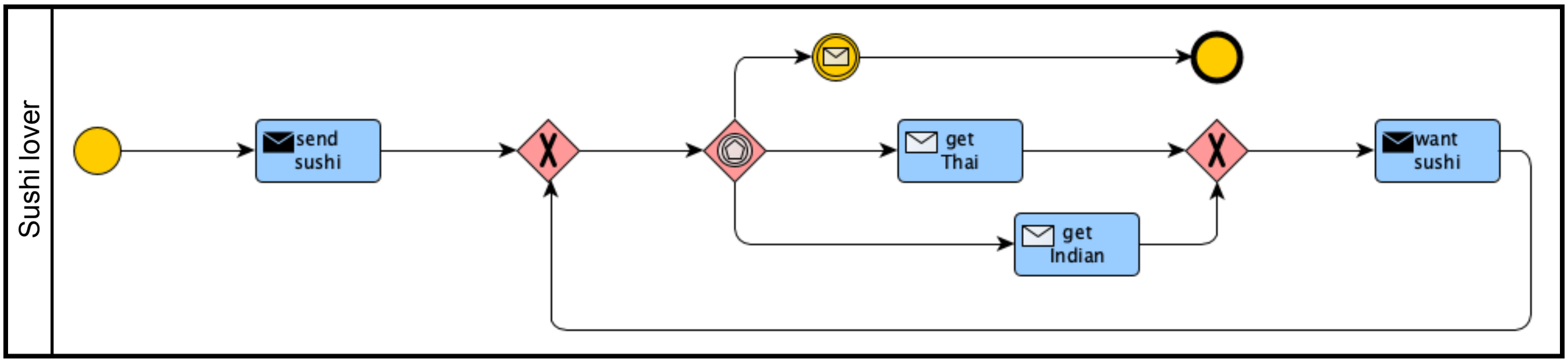
Sound?

Sushi lover

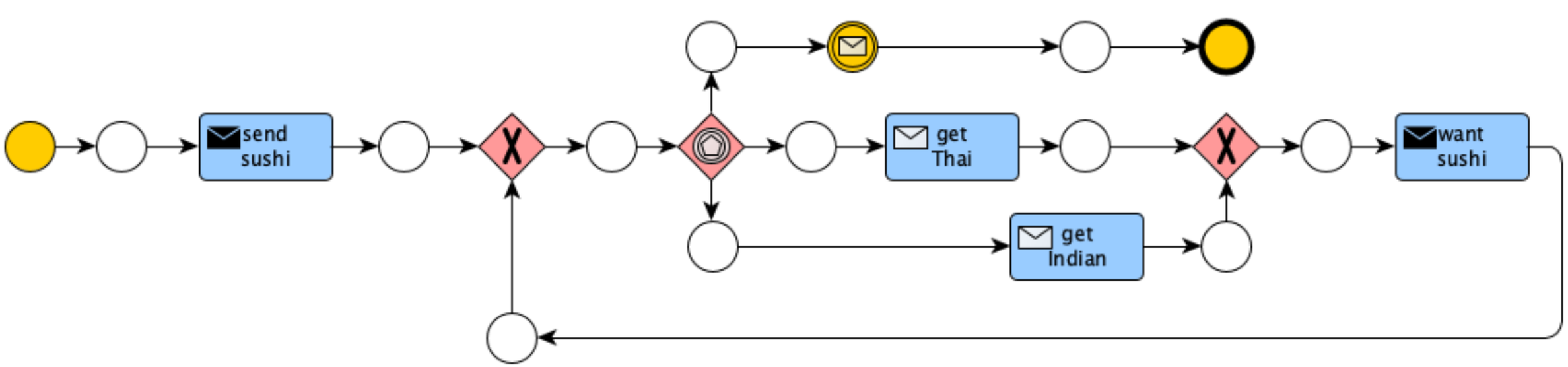


Sound?

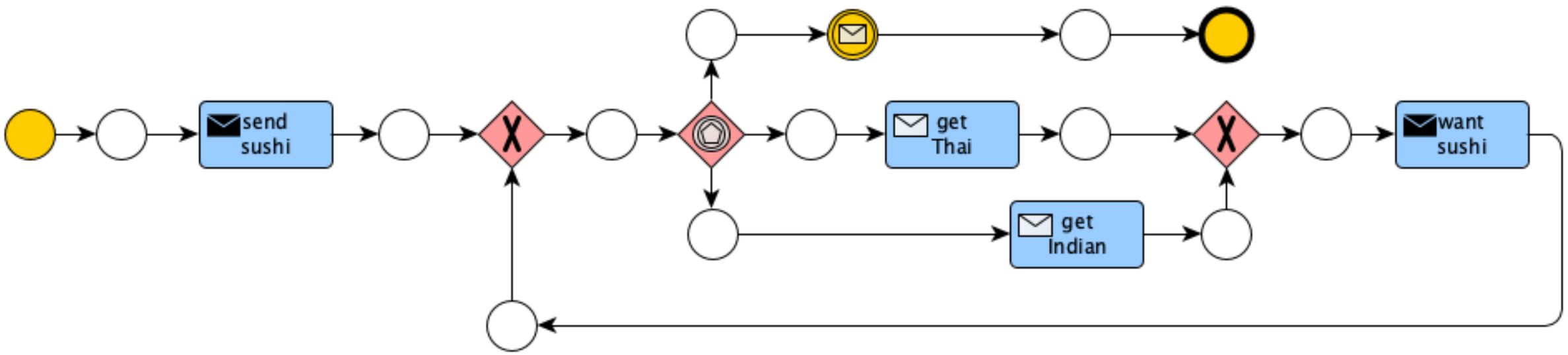
Sushi lover: step 1



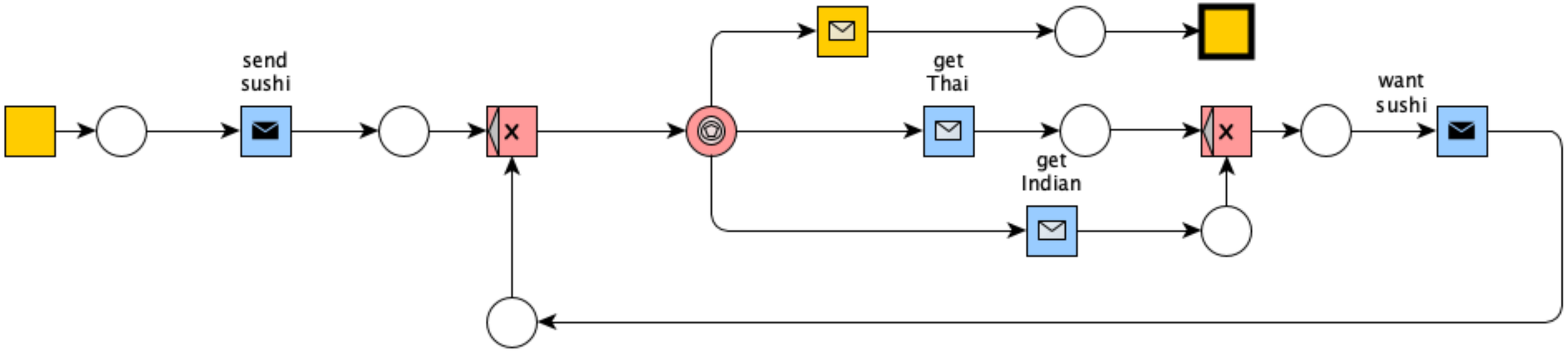
Step 1
sequence flow
message flow



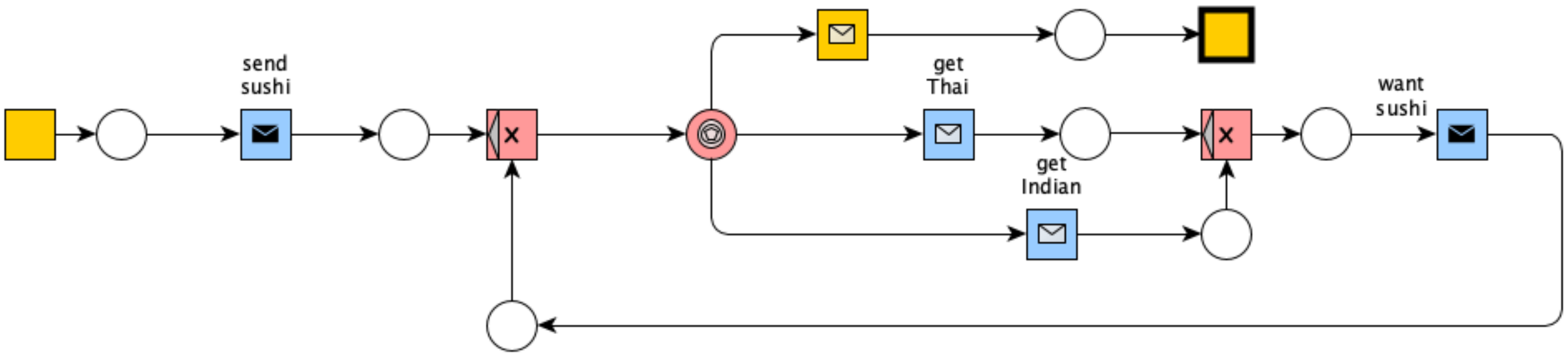
Sushi lover: step 2

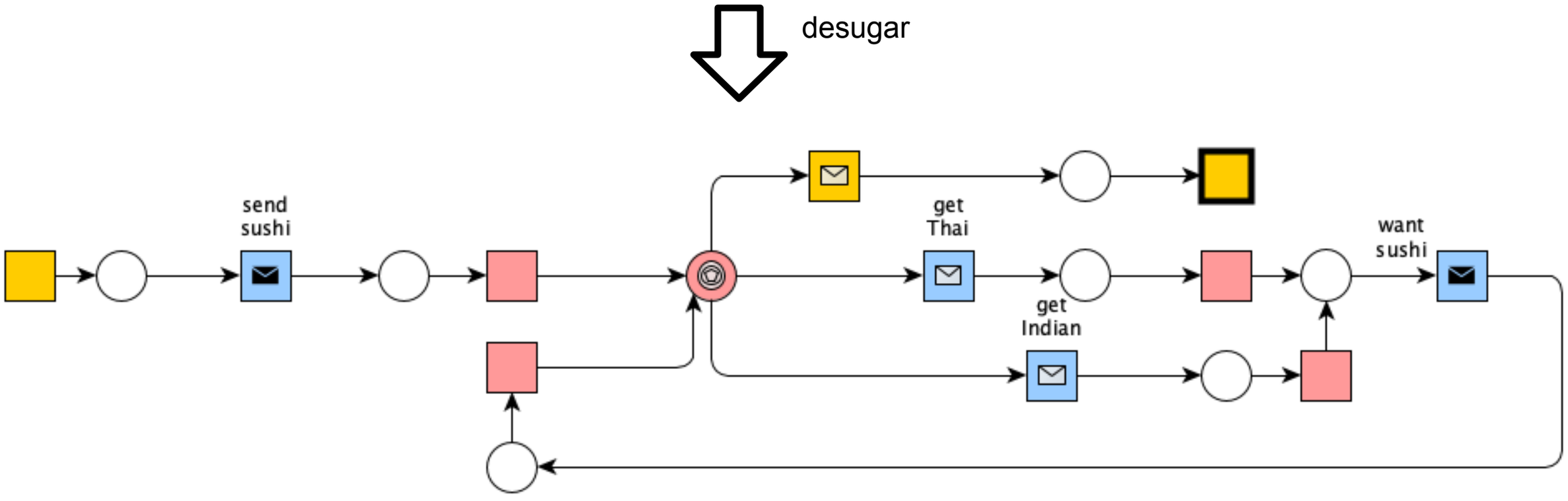


Step 2
flow objects

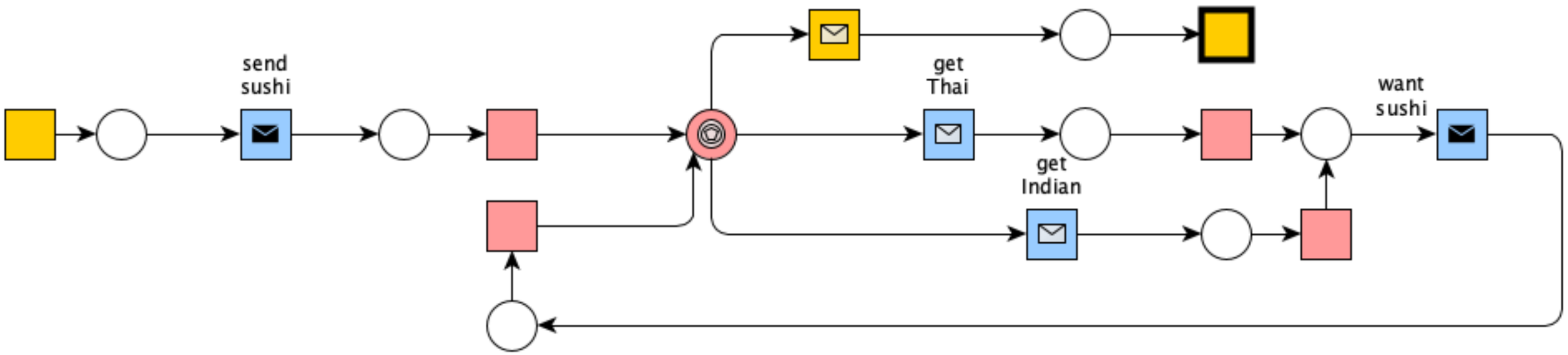


Sushi lover: (desugar)

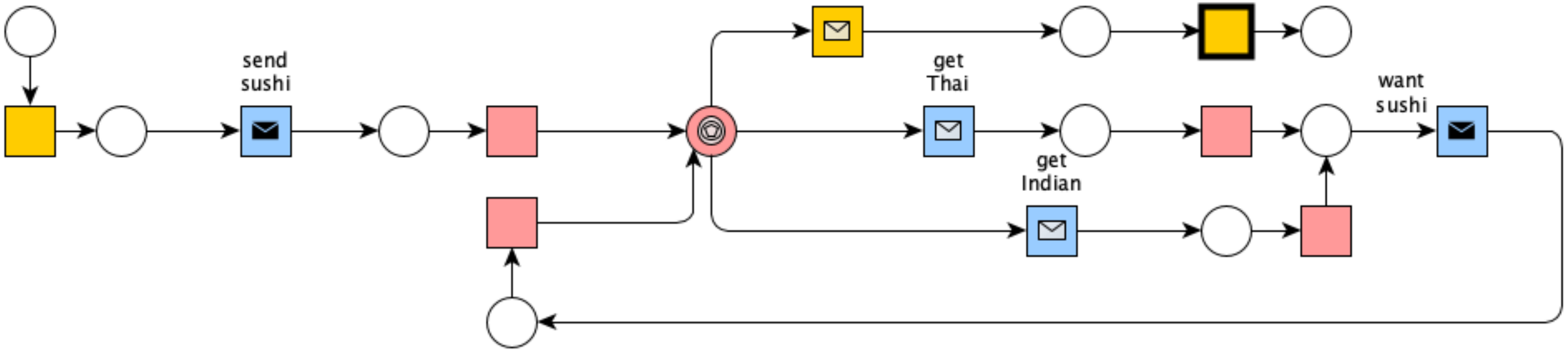




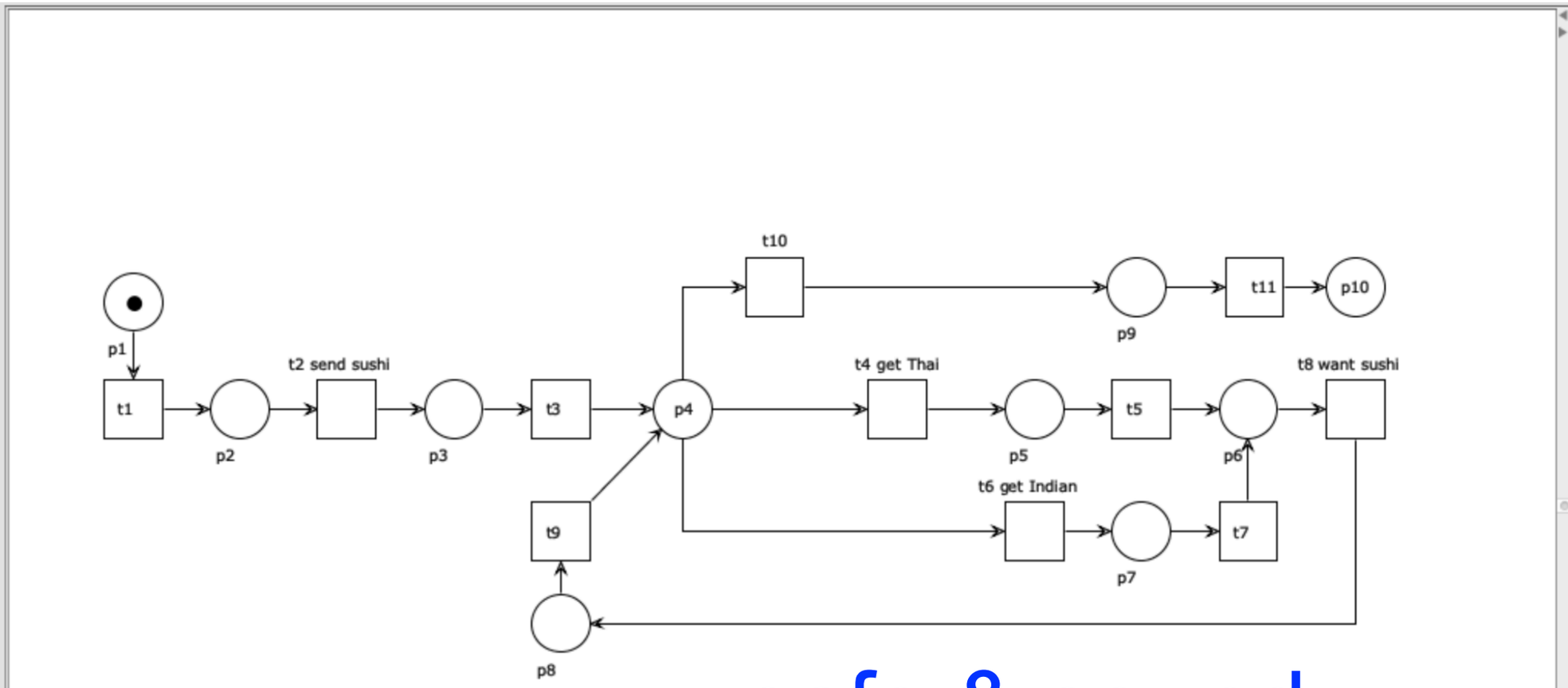
Sushi lover: step 3



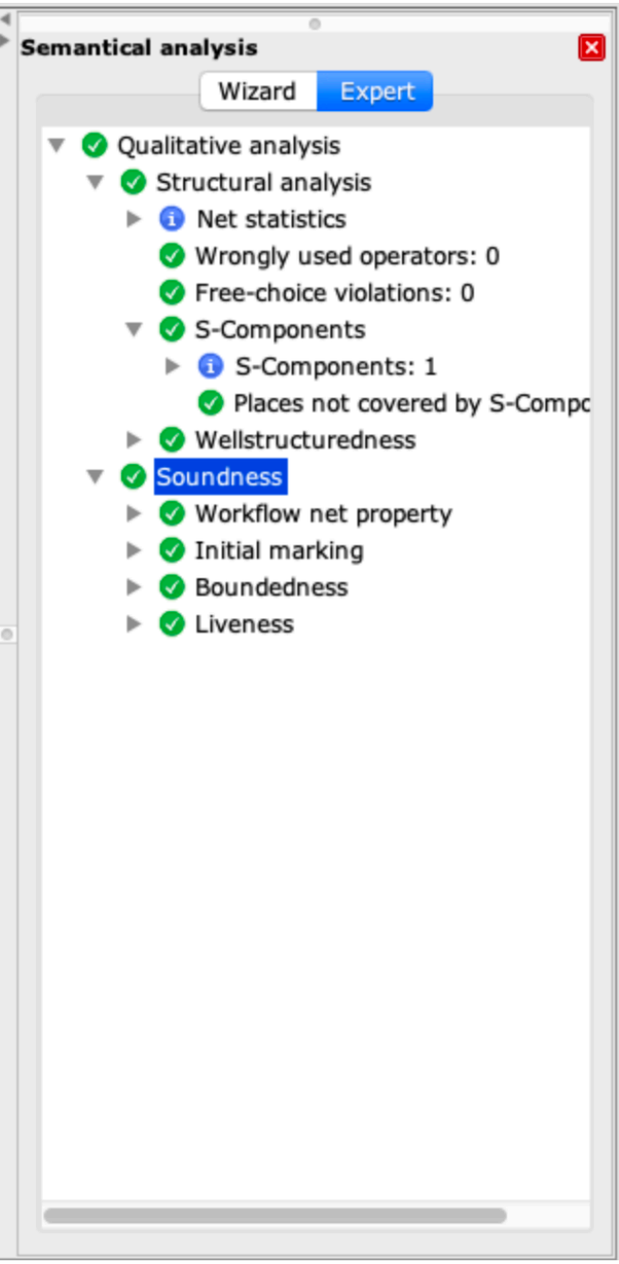
Step 3
enforce
initial place
final place



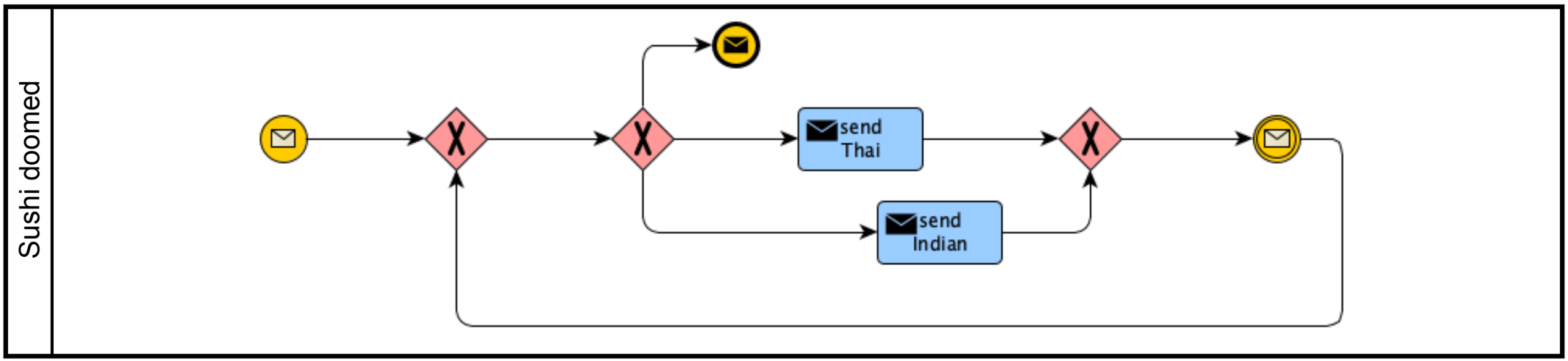
Soundness analysis



safe & sound
(s-net)

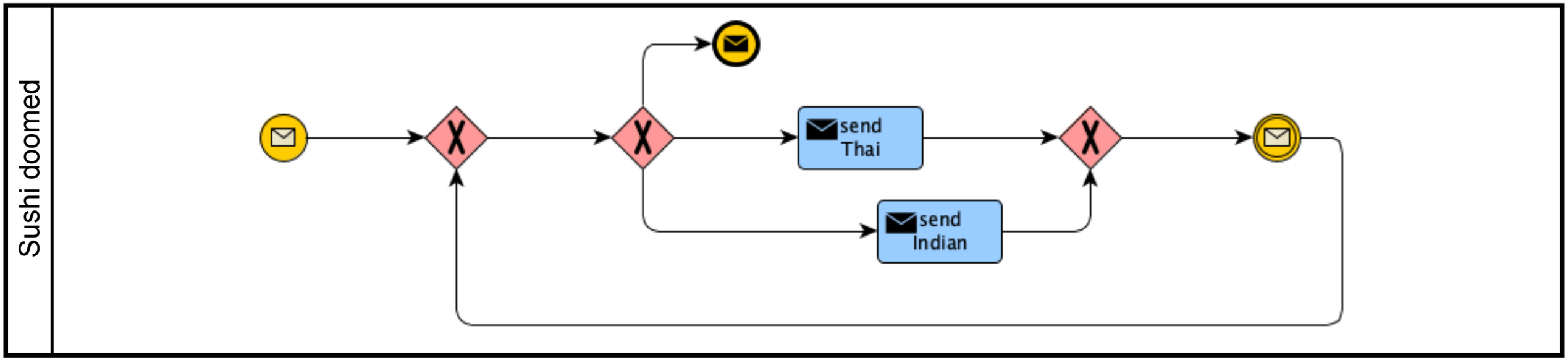


Sushi doomed

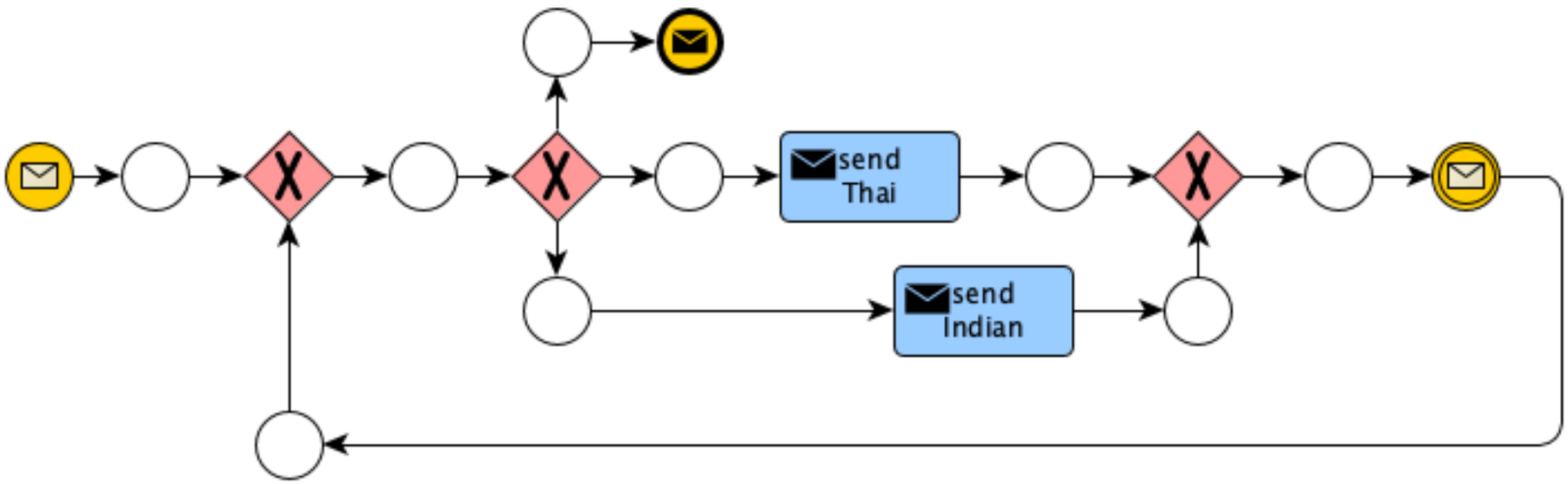


Sound?

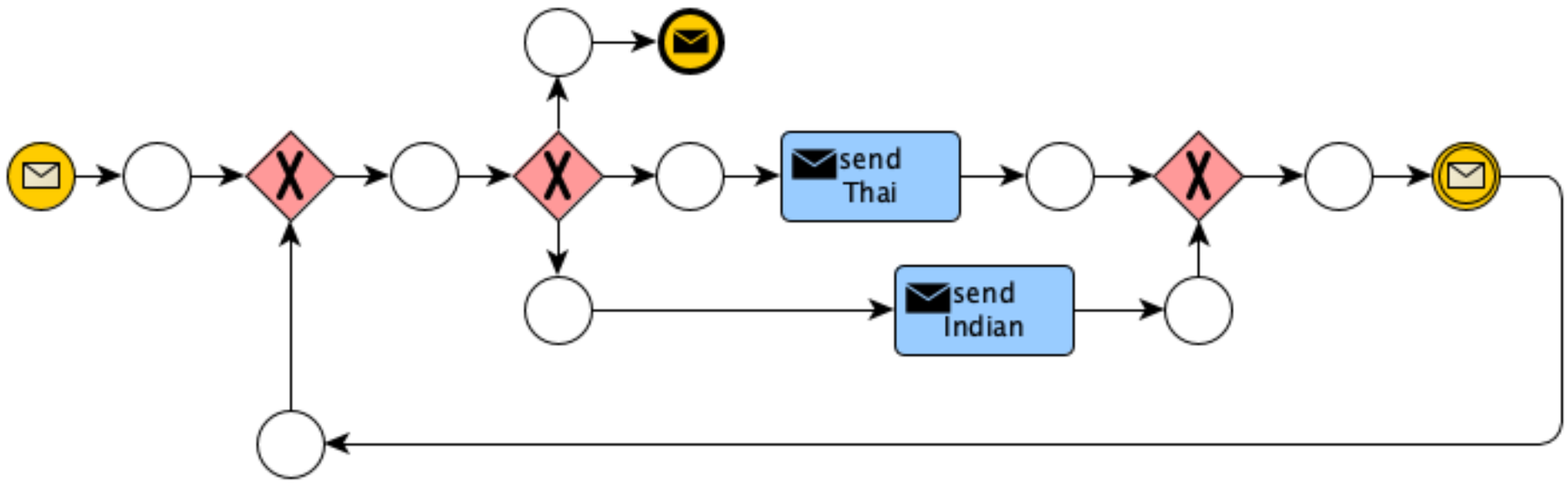
Sushi doomed: step 1



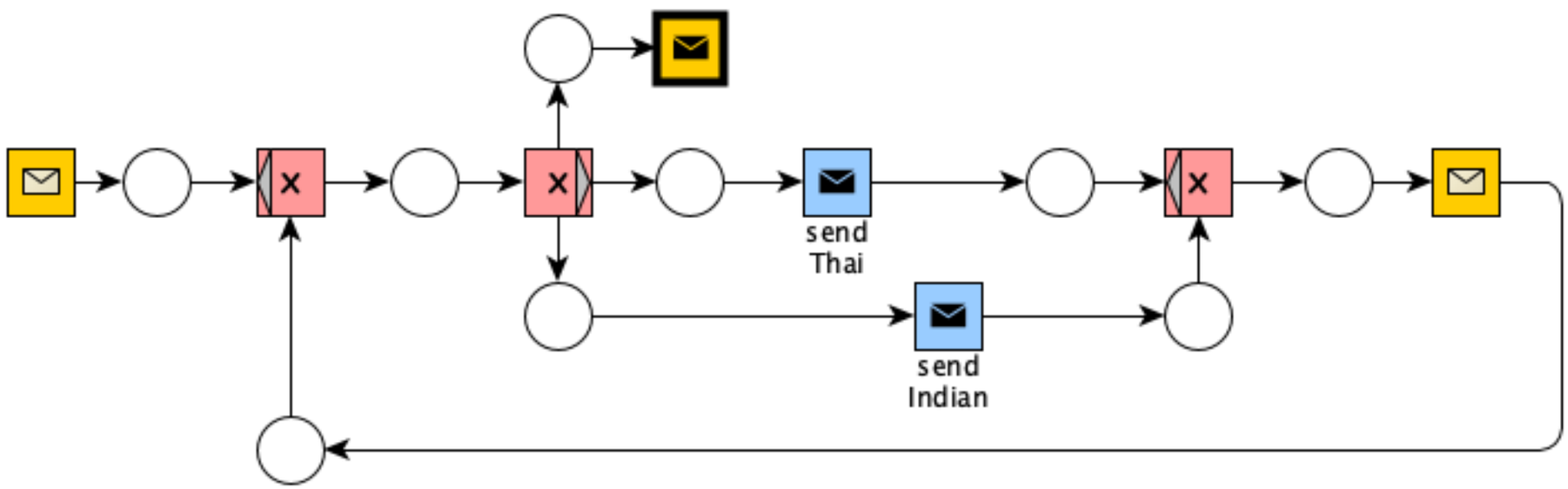
Step 1
sequence flow
message flow



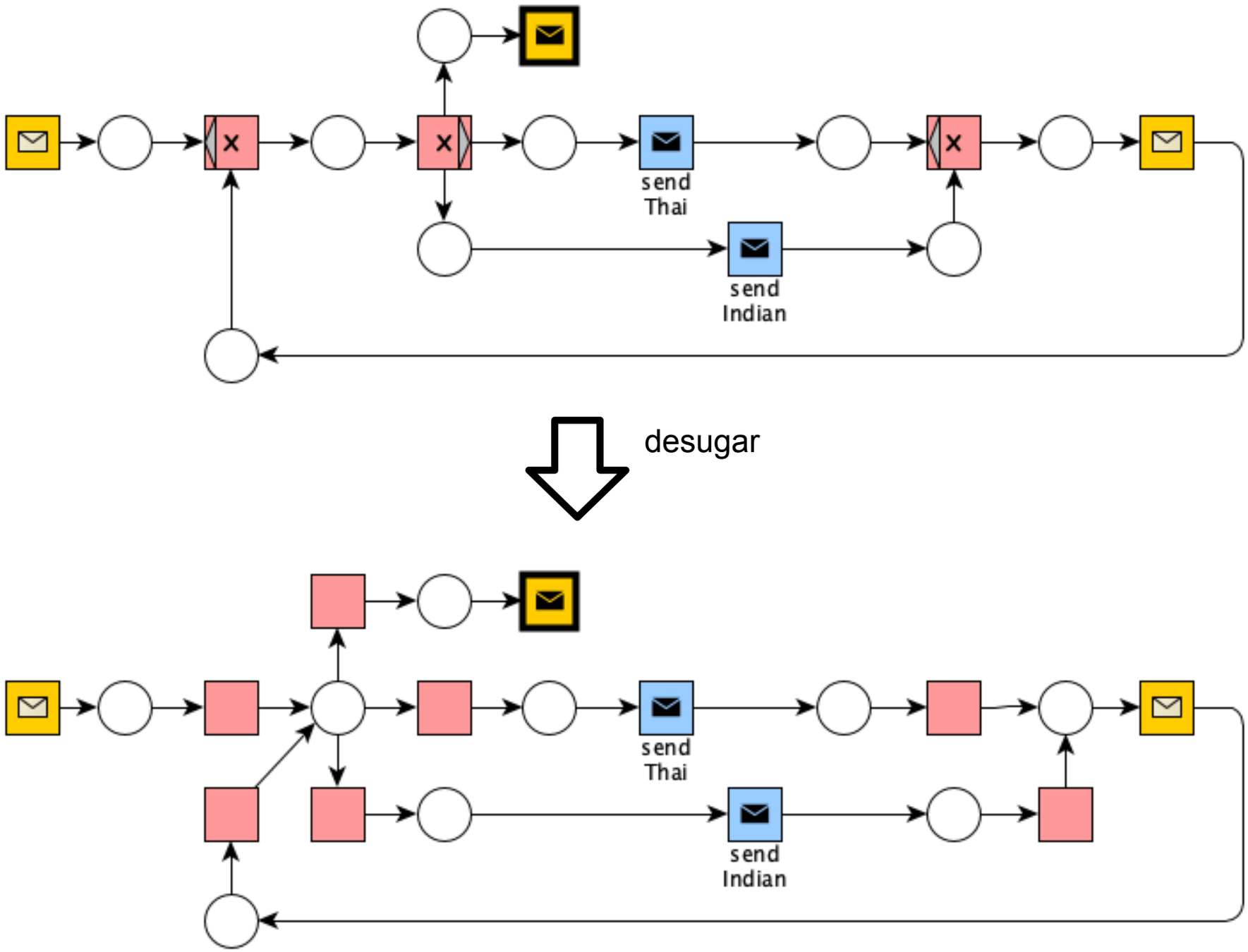
Sushi doomed: step 2



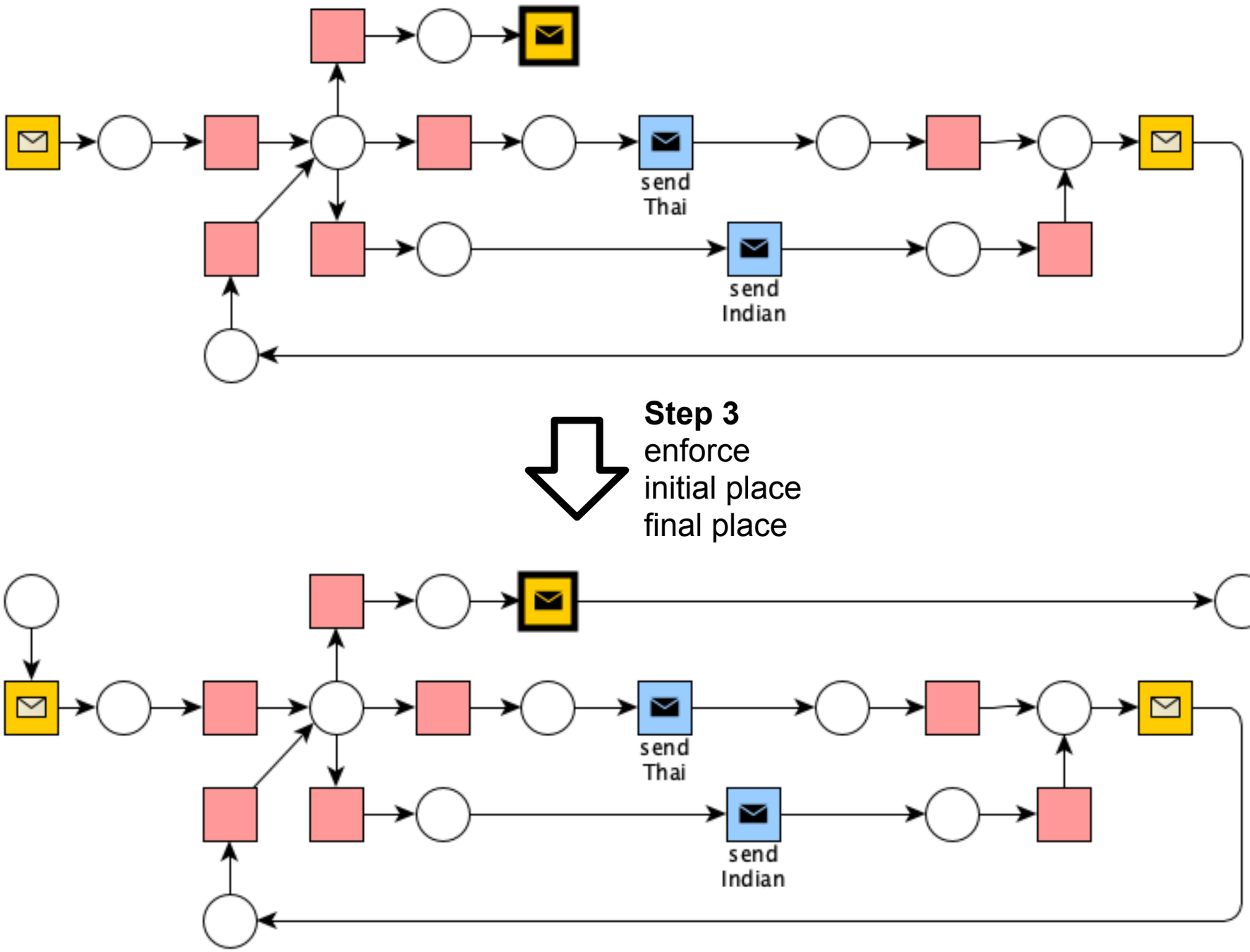
Step 2
flow objects



Sushi doomed: (desugar)

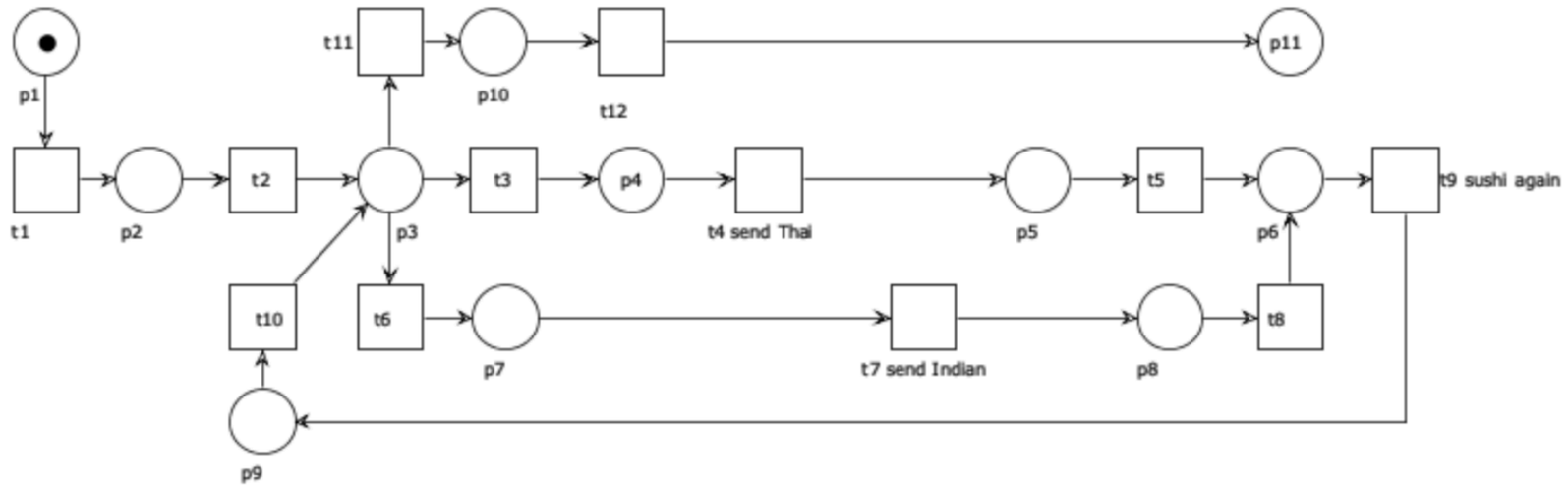


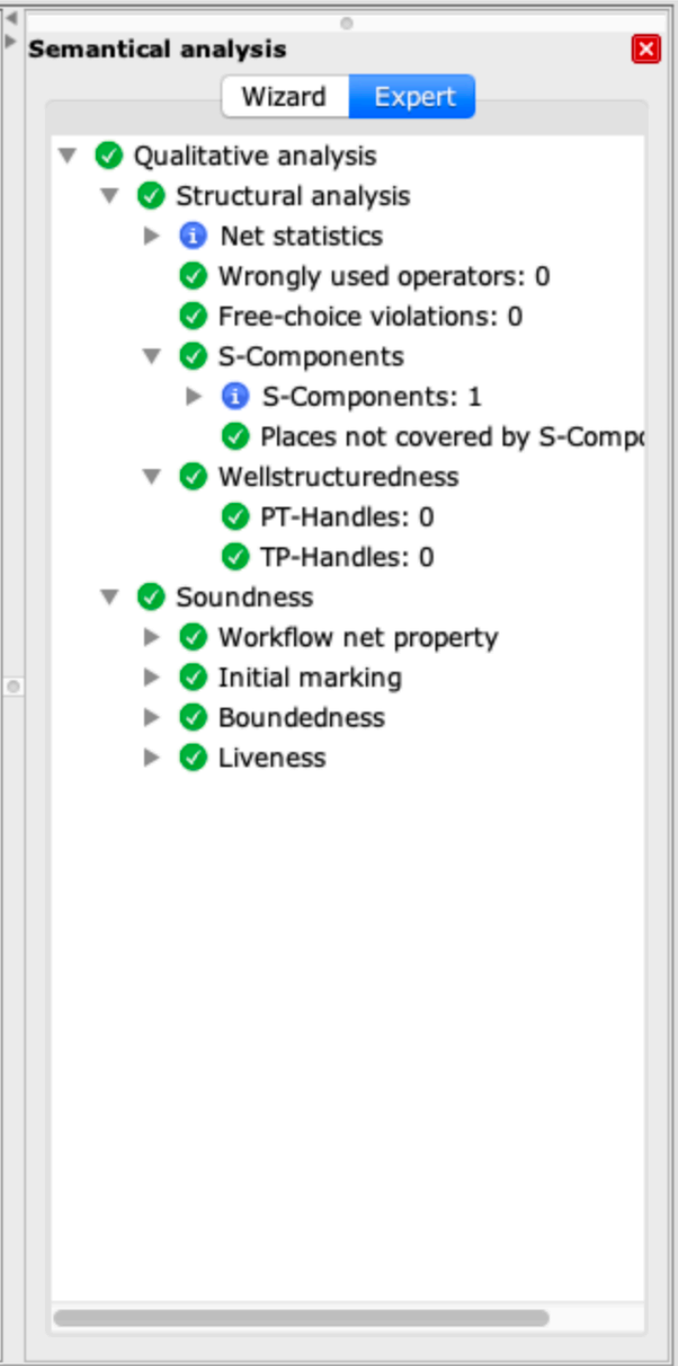
Sushi doomed: step 3



Step 3
enforce
initial place
final place

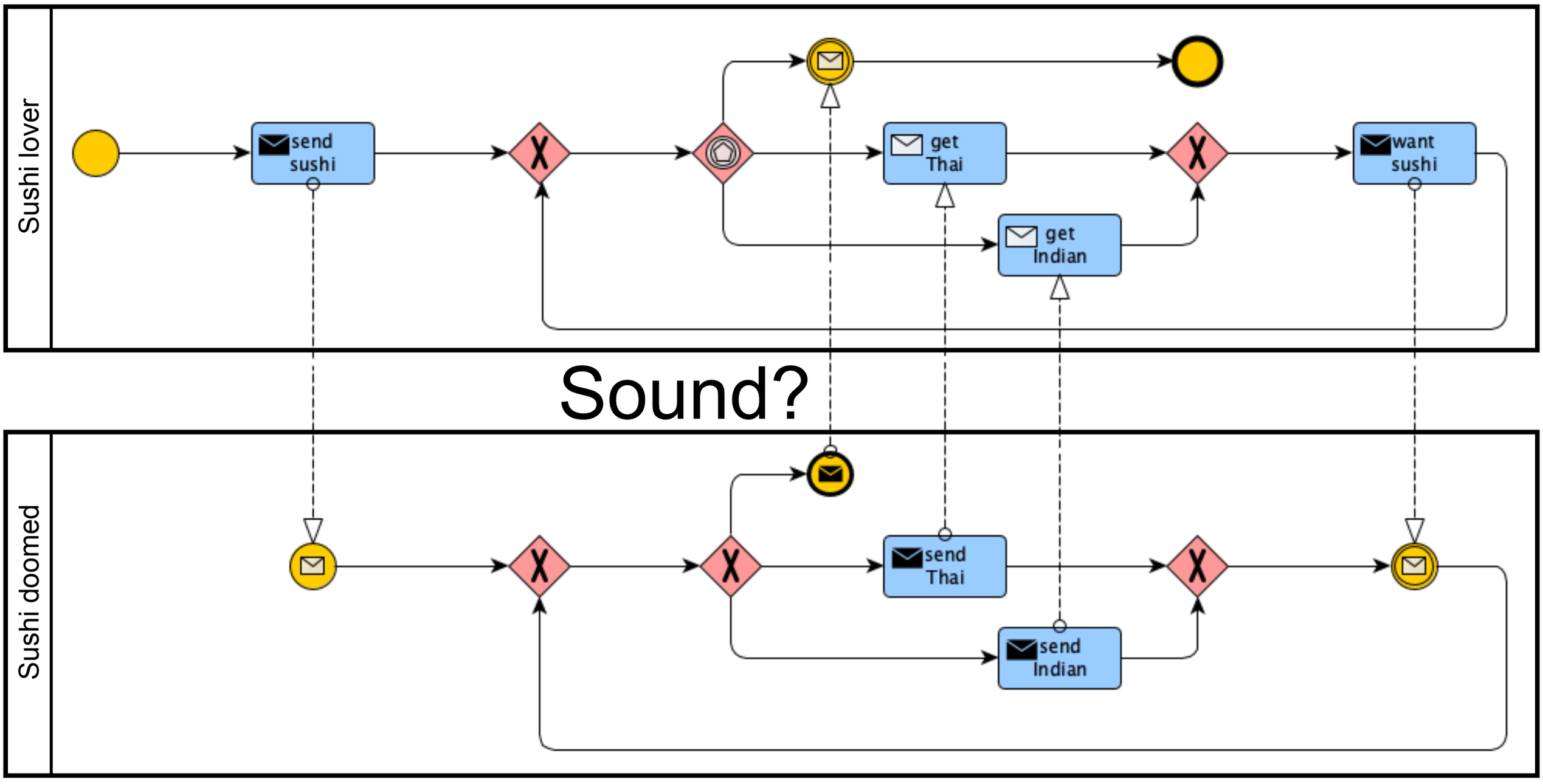
Soundness analysis





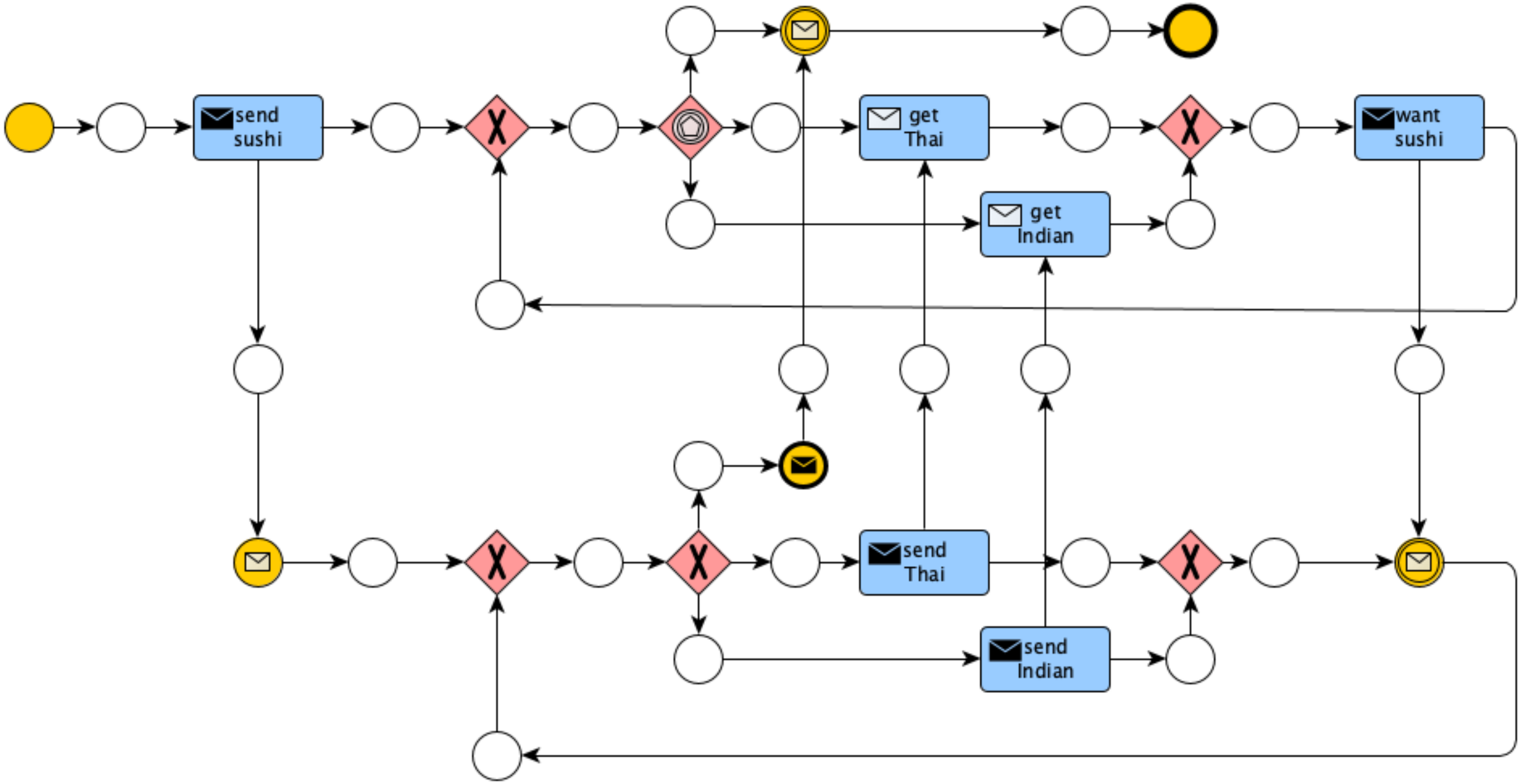
safe & sound
(s-net)

Sushi system

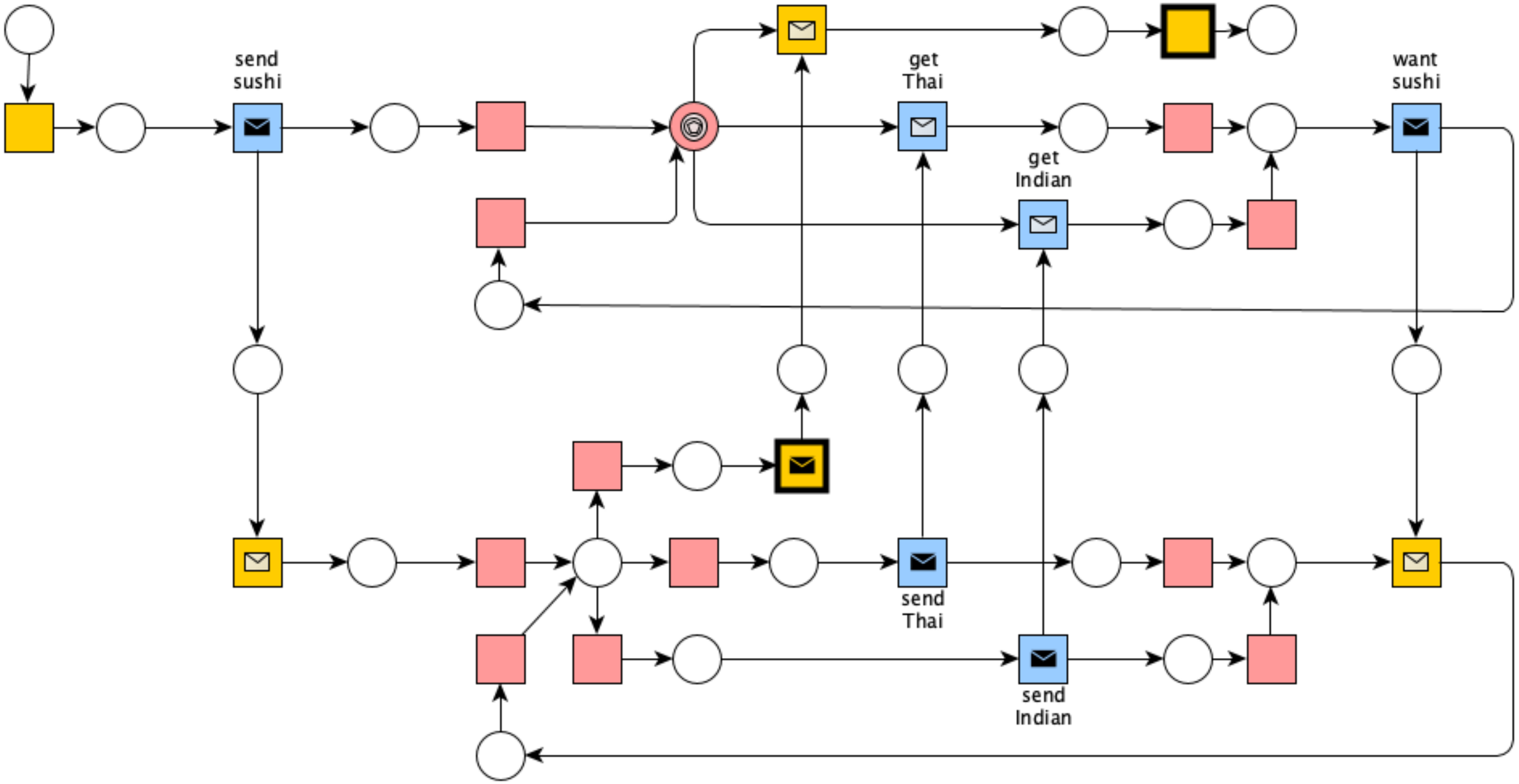


Sound?

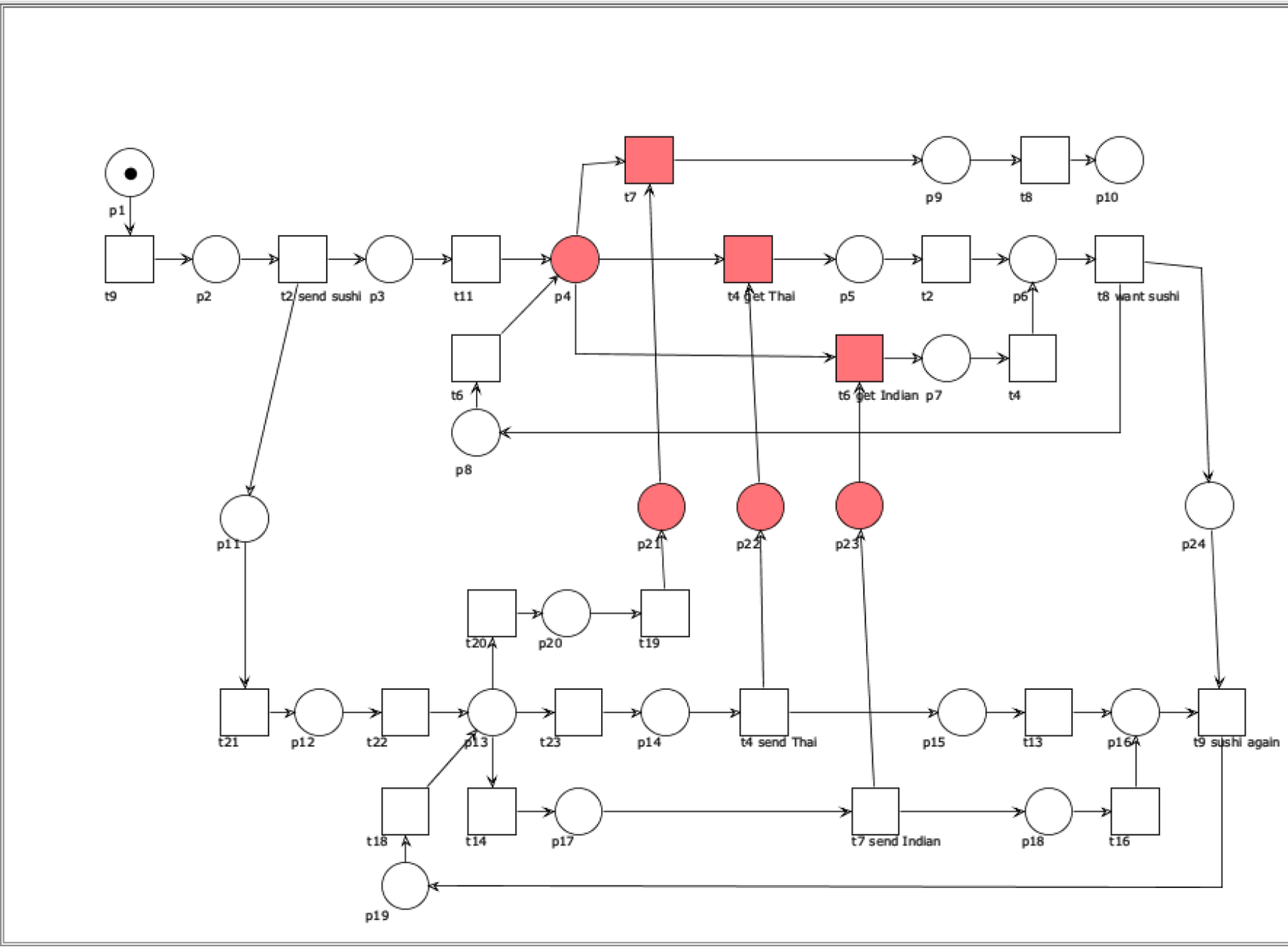
Sushi system: step 1

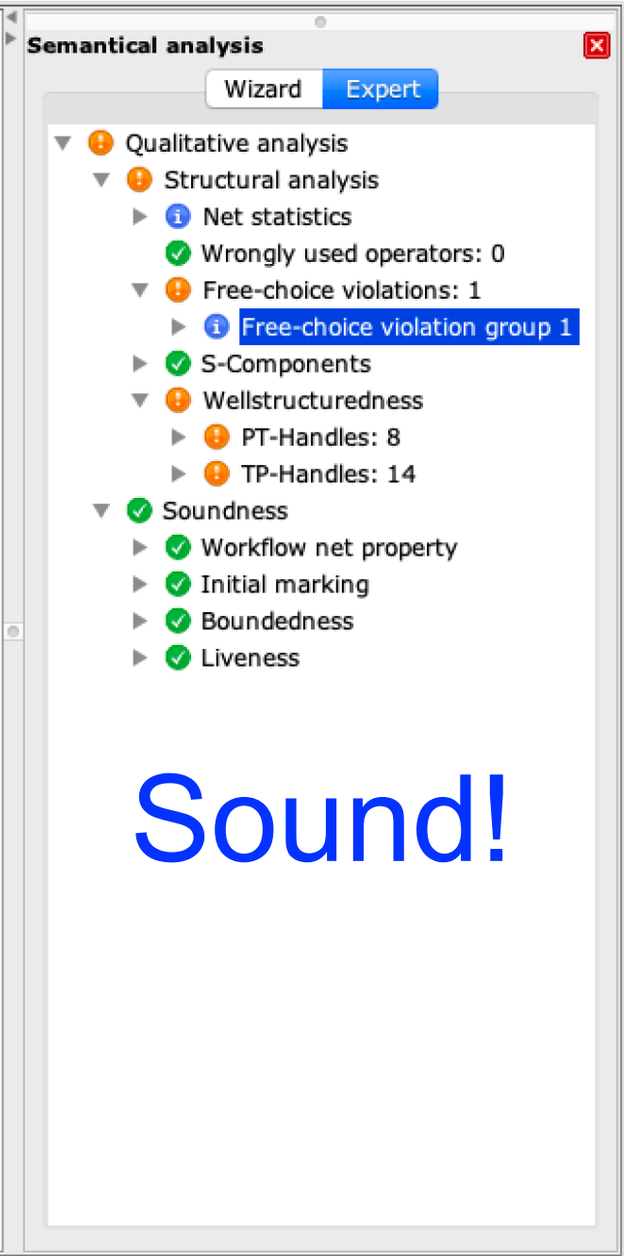


Sushi system: step 1+2+3

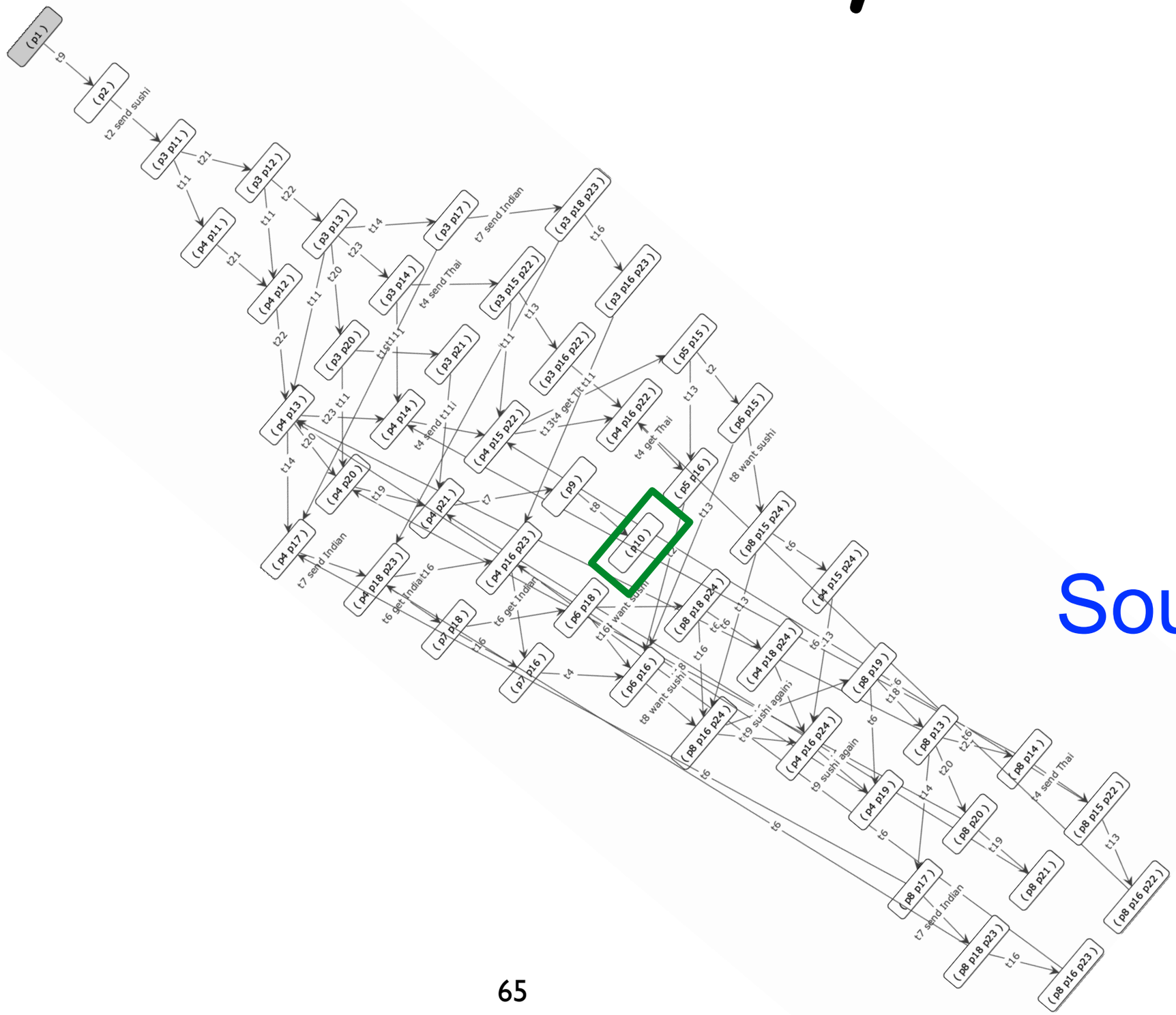


Soundness analysis





Soundness analysis

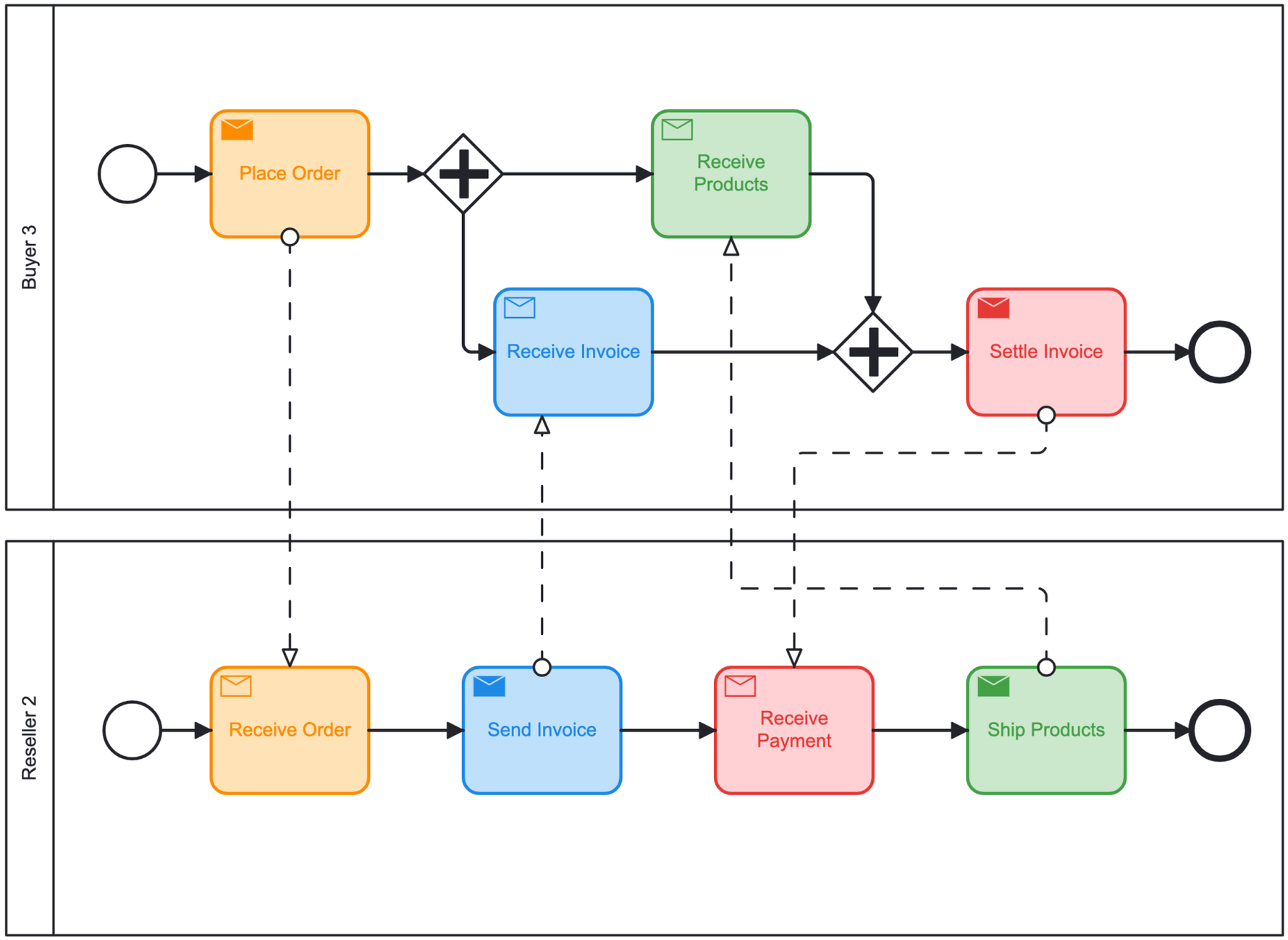


Sound!

Example:

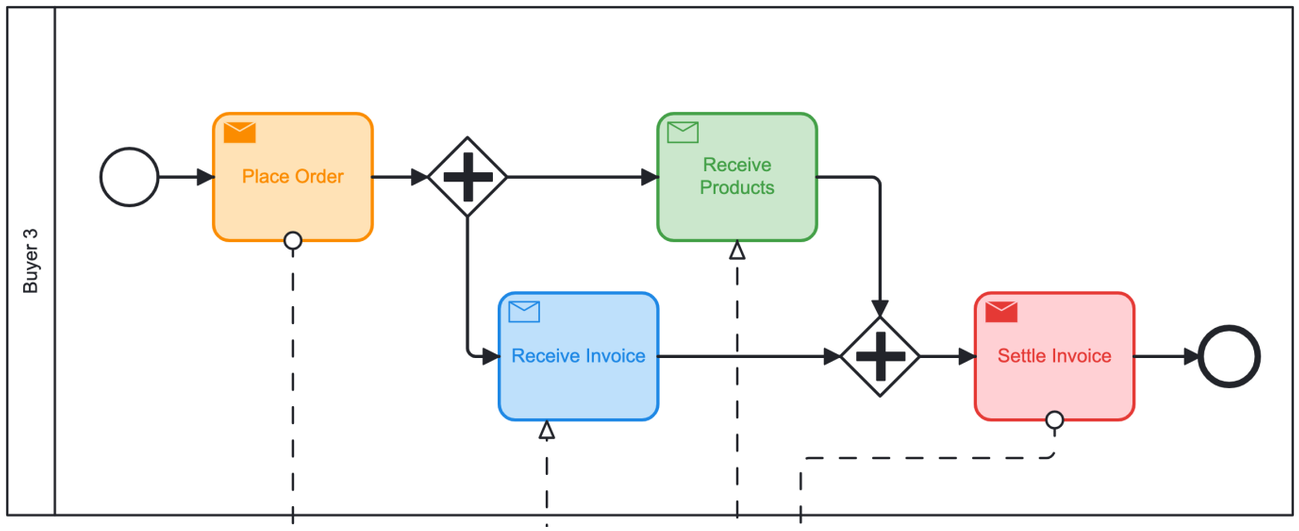
Buyer - Reseller

Buyer 3 and Reseller 2

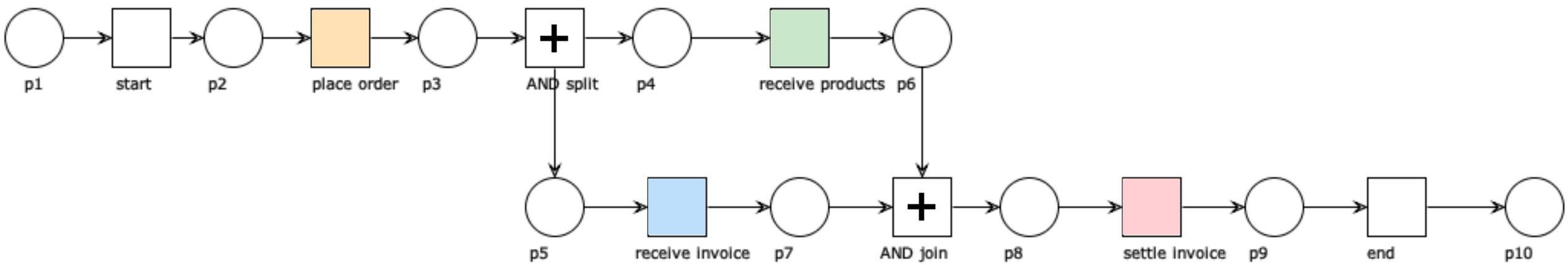


Sound?

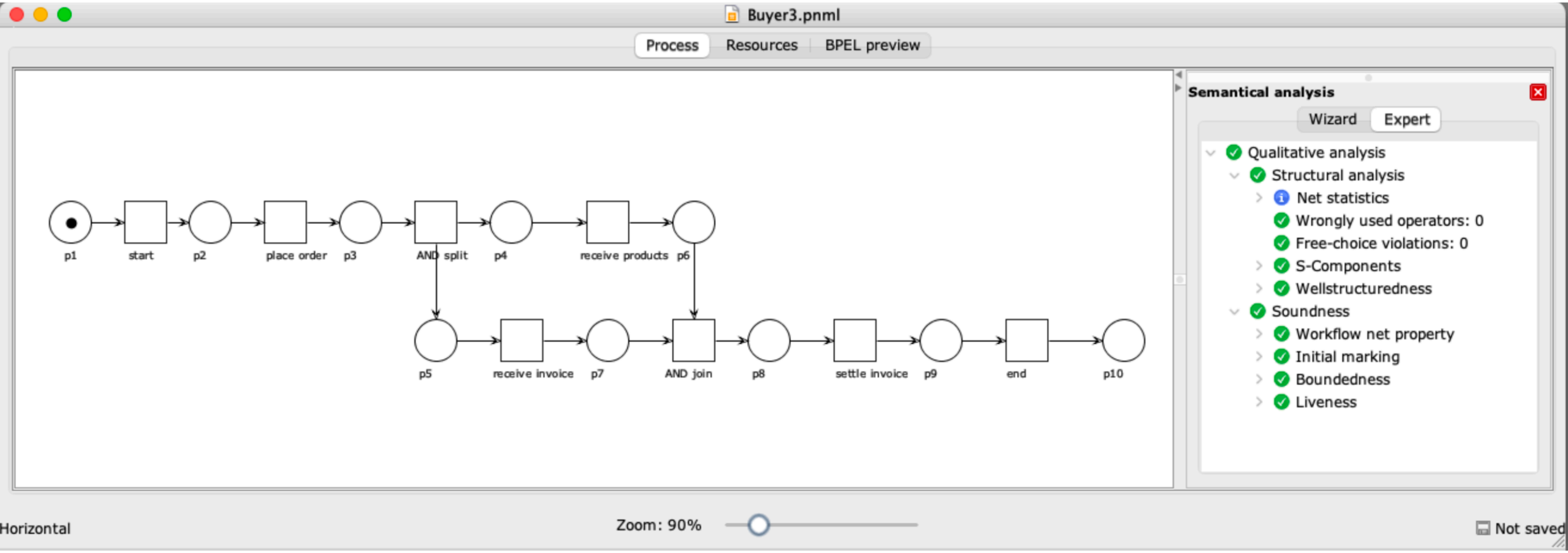
Buyer 3 sound?



Step 1 + 2 + 3

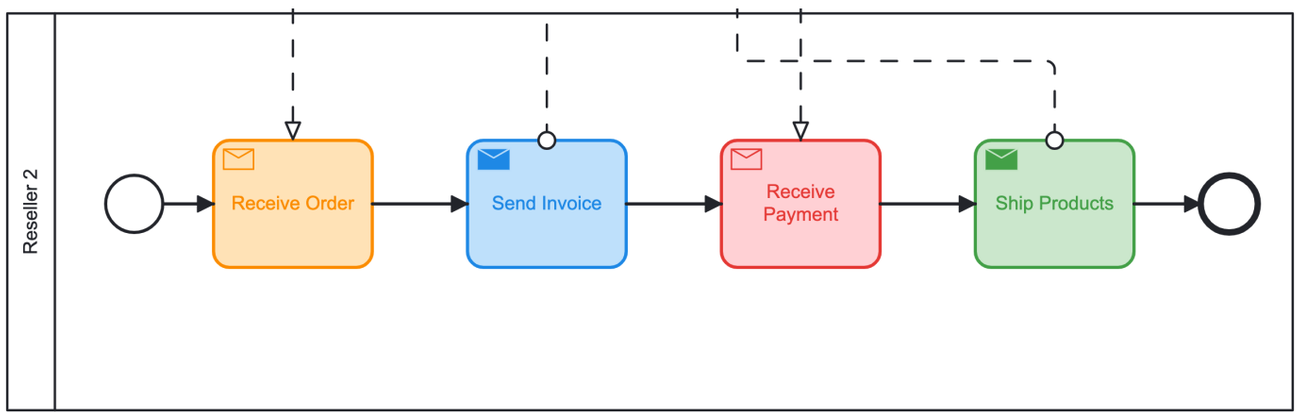


Buyer 3 soundness analysis

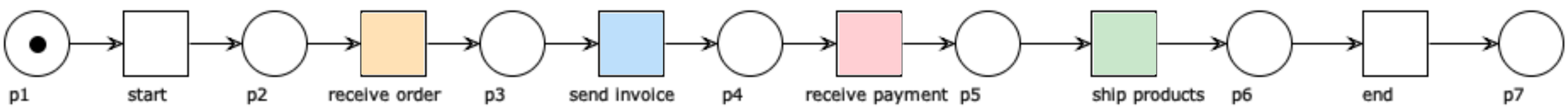


Safe and sound!

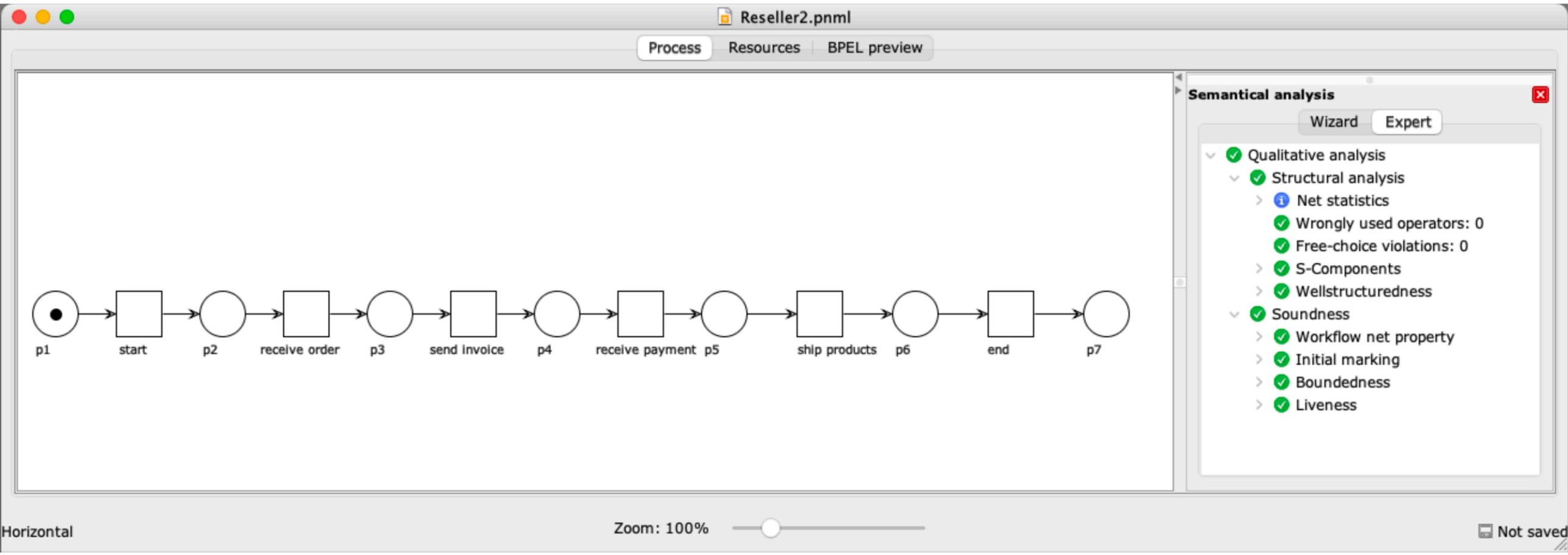
Reseller 2 sound?



Step 1 + 2 + 3

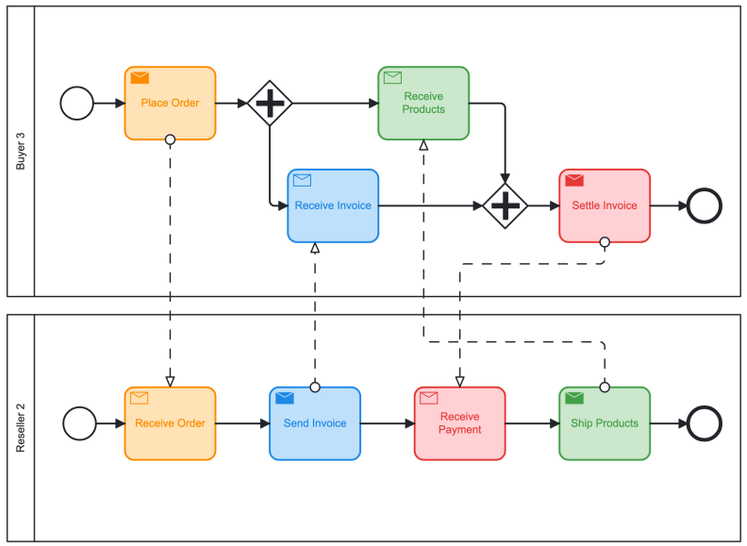


Reseller 2 soundness analysis

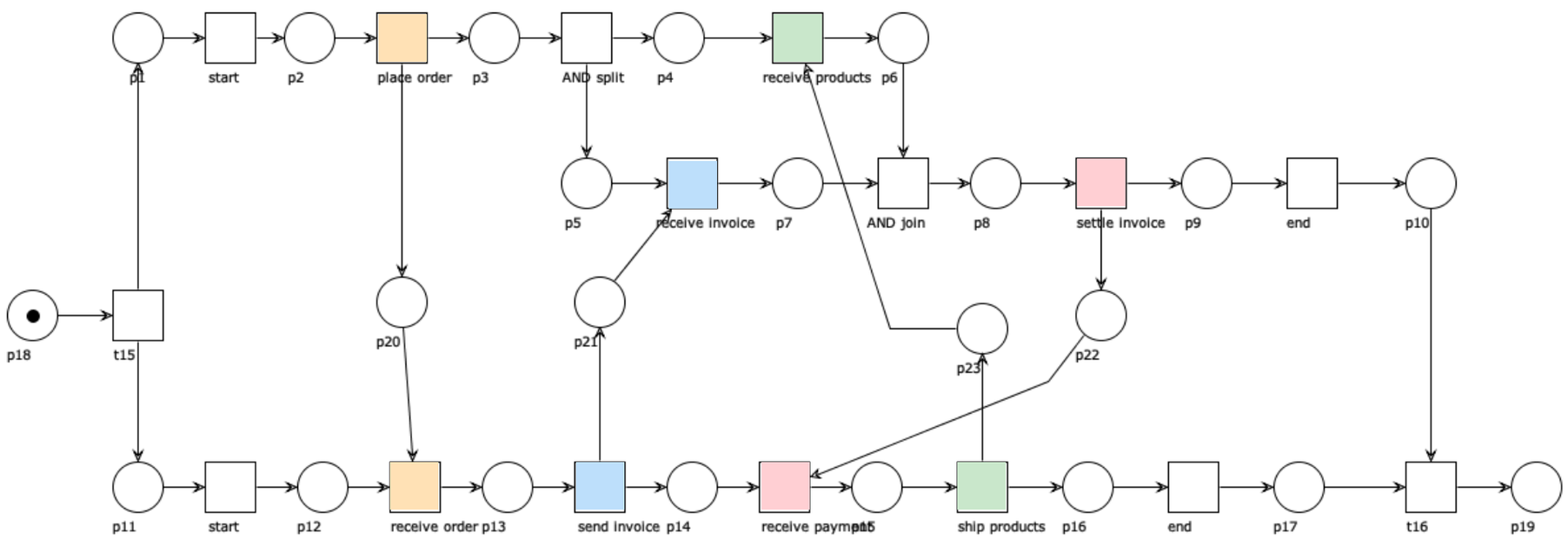


Safe and sound!

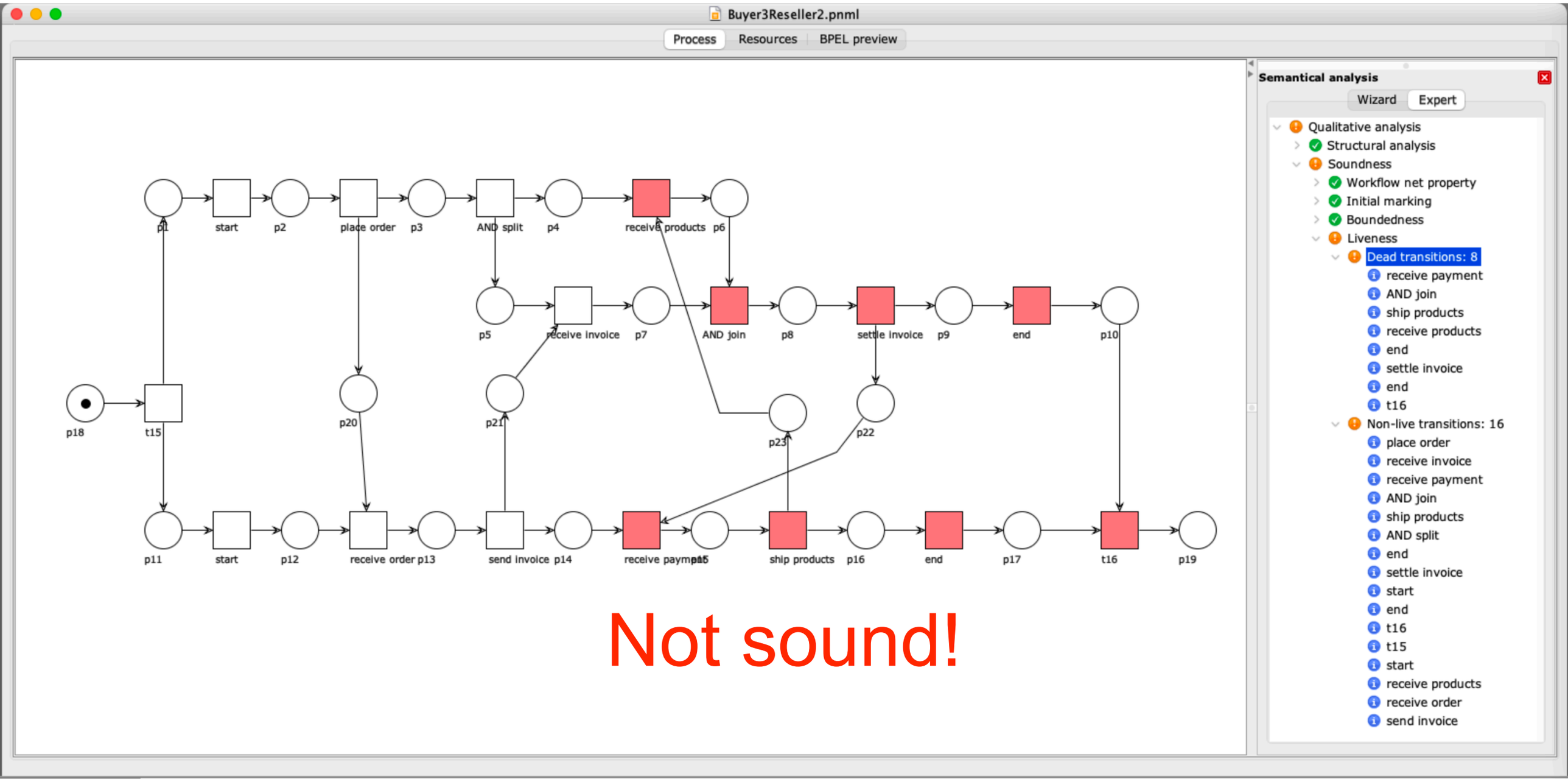
Buyer 3 + Reseller 2: sound?



Step 1 + 2 + 3

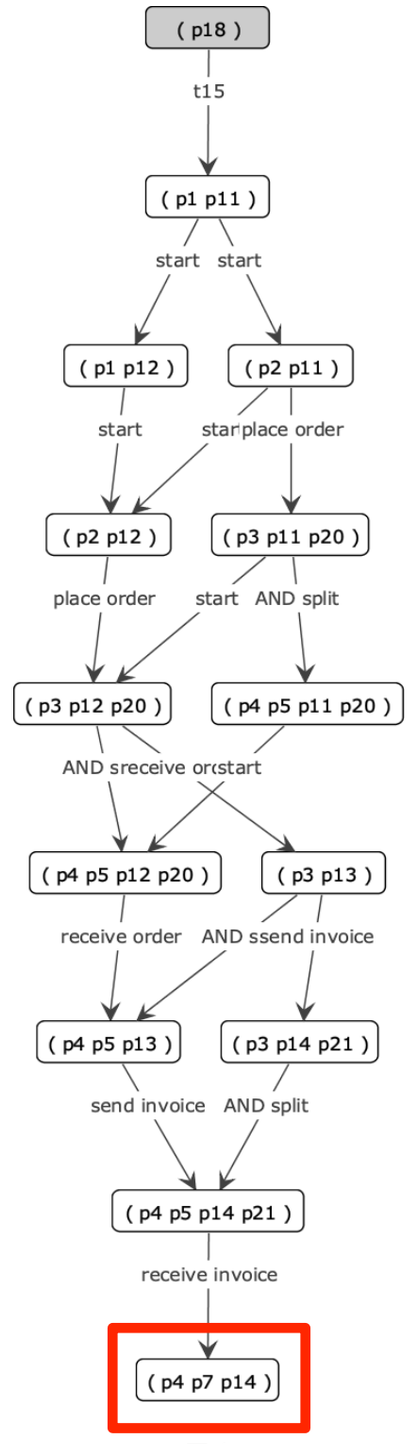


Buyer 3 + Reseller 2: analysis



Not sound!

Buyer 3 + Reseller 2: analysis



Not sound!

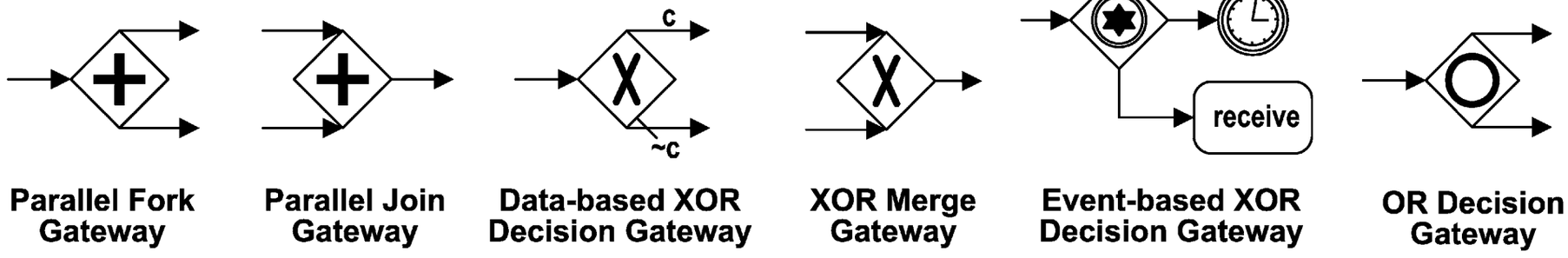
Step 0: preprocessing BPMN diagrams

Overview

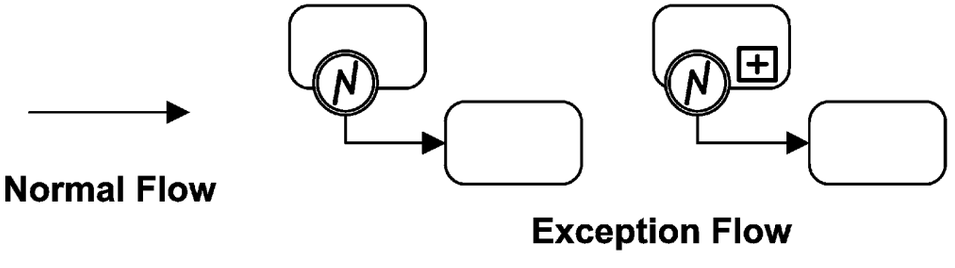
EVENT

ACTIVITY

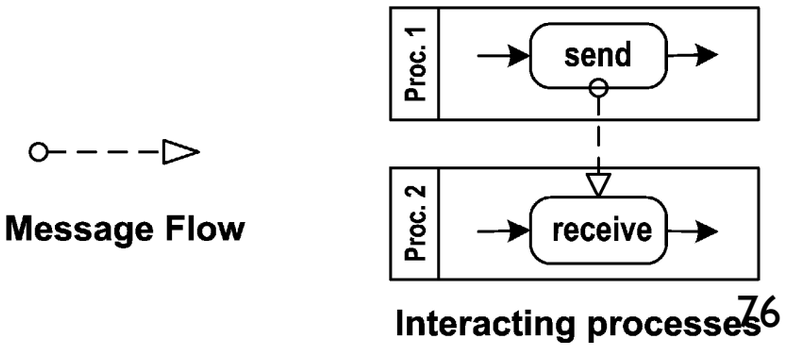
GATEWAY



SEQUENCE FLOW



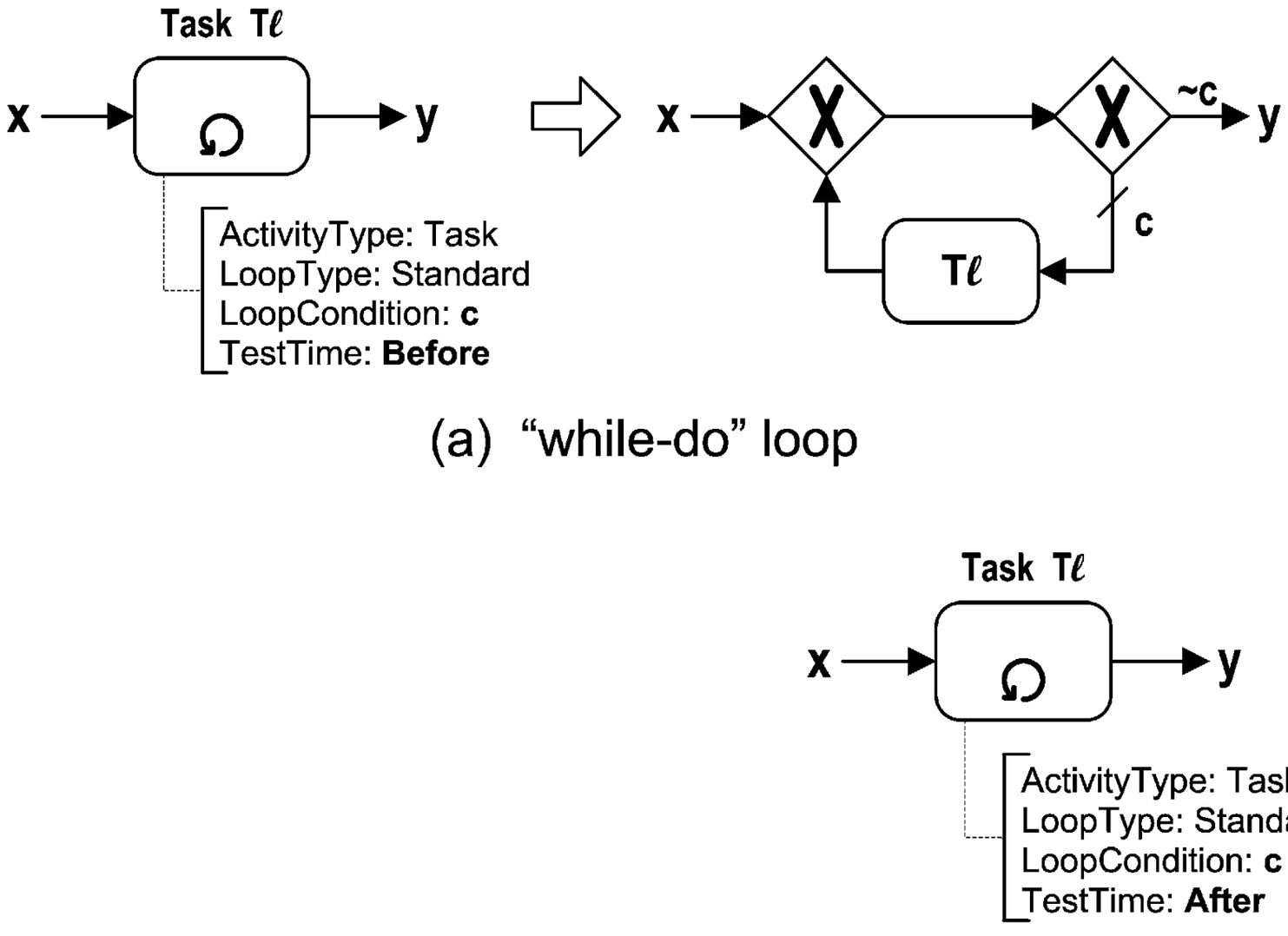
MESSAGE FLOW



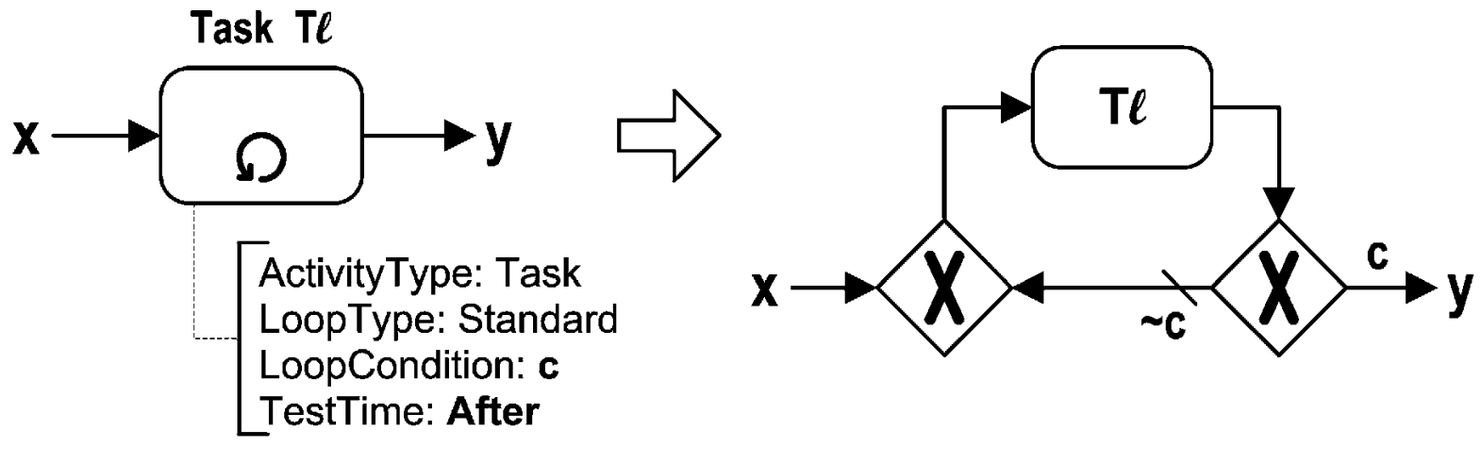
[Note]:

1. **Apart from intermediate error events, intermediate message or timer events may also be the source of exception flows.**
2. **A message flow may link task to task, end event to task, task to start event, and end event to start event.**

Activity looping

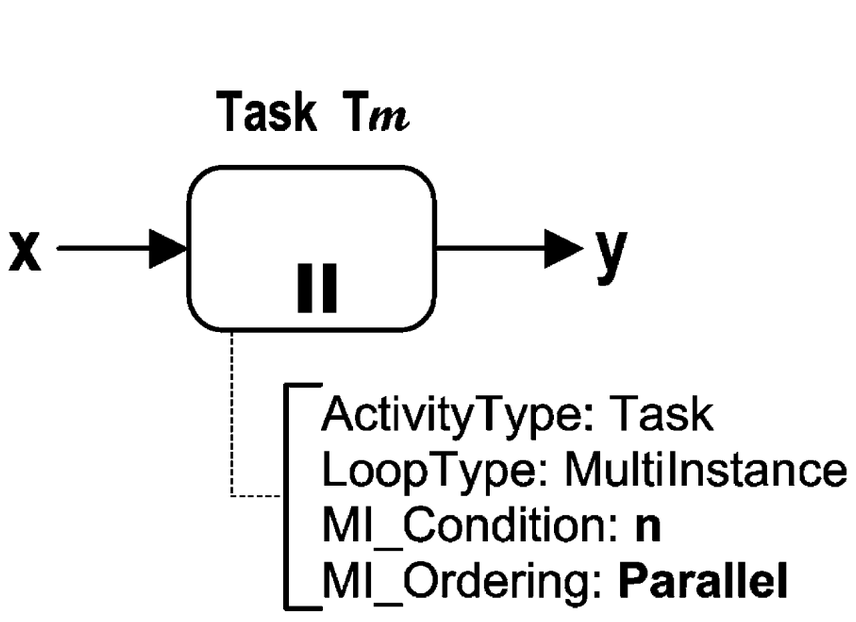


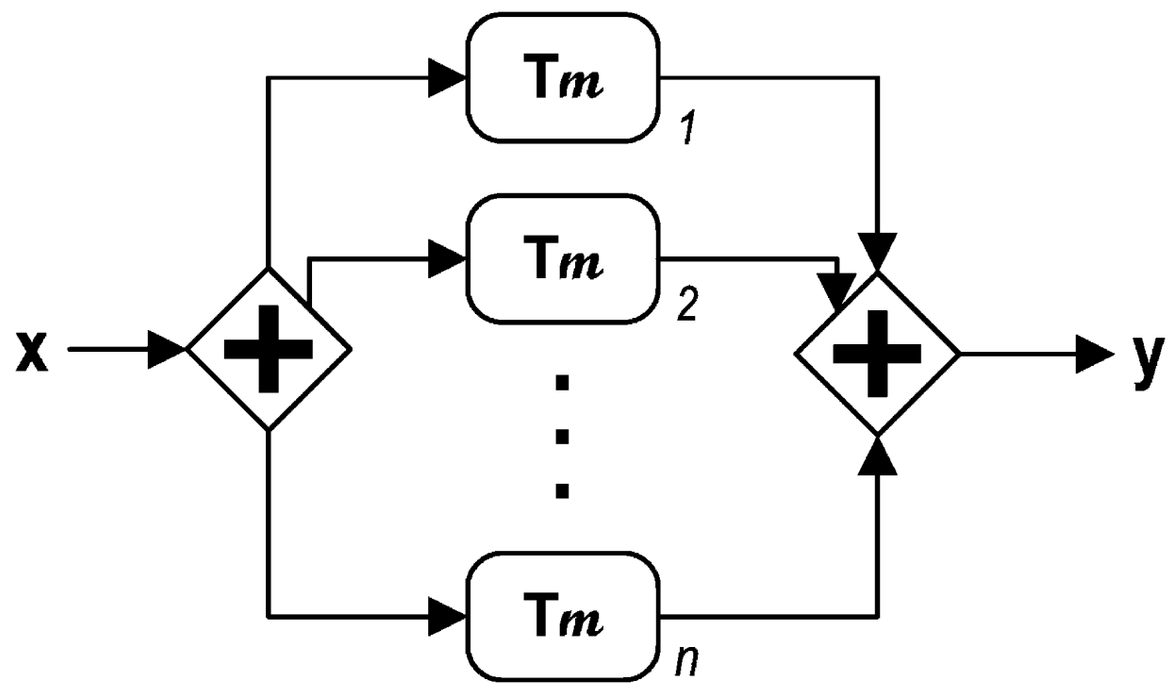
(a) “while-do” loop



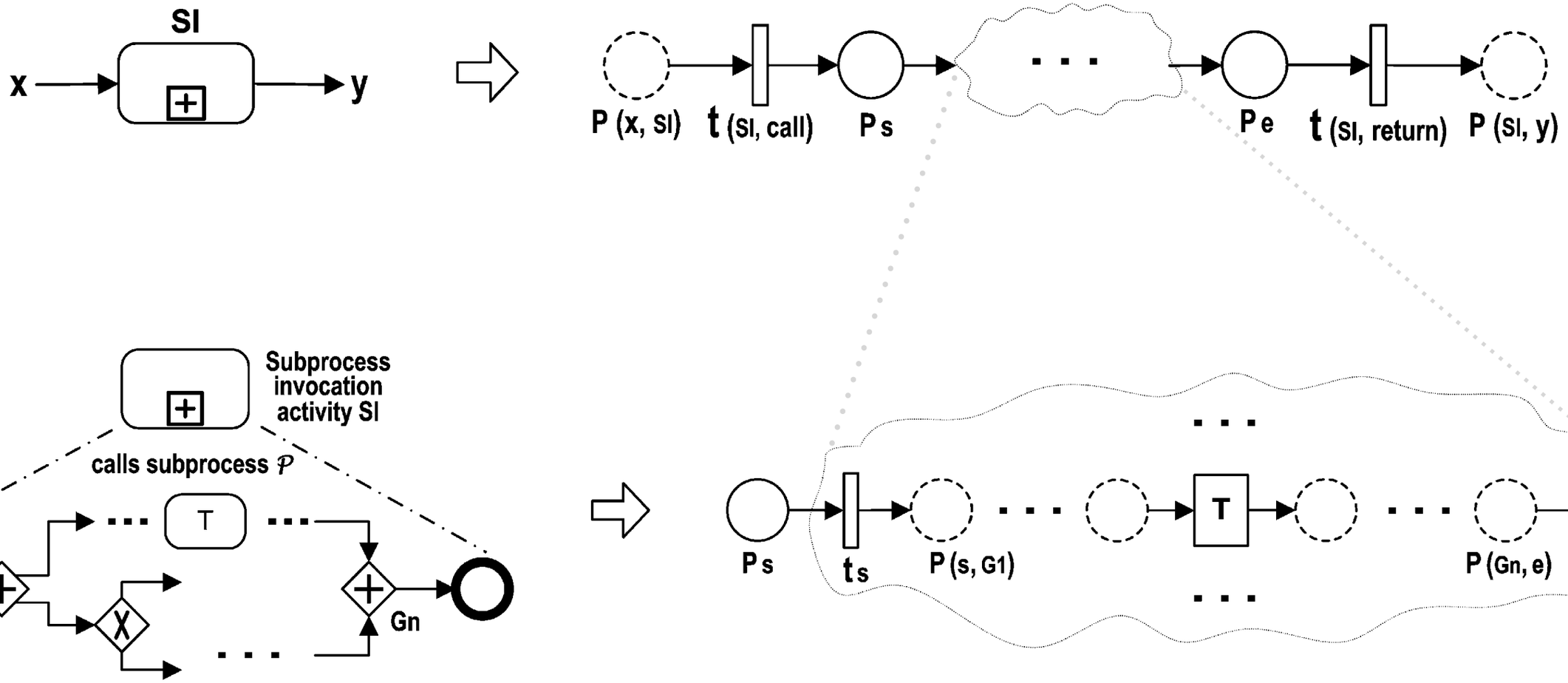
(b) “do-until” loop

Multiple instances (design-time bounded)

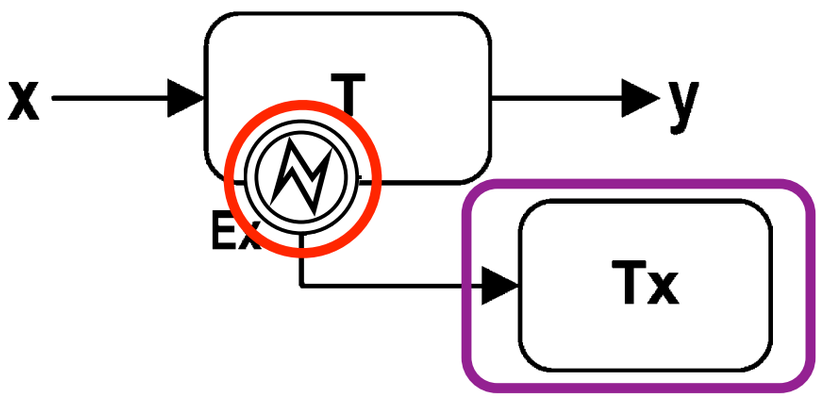


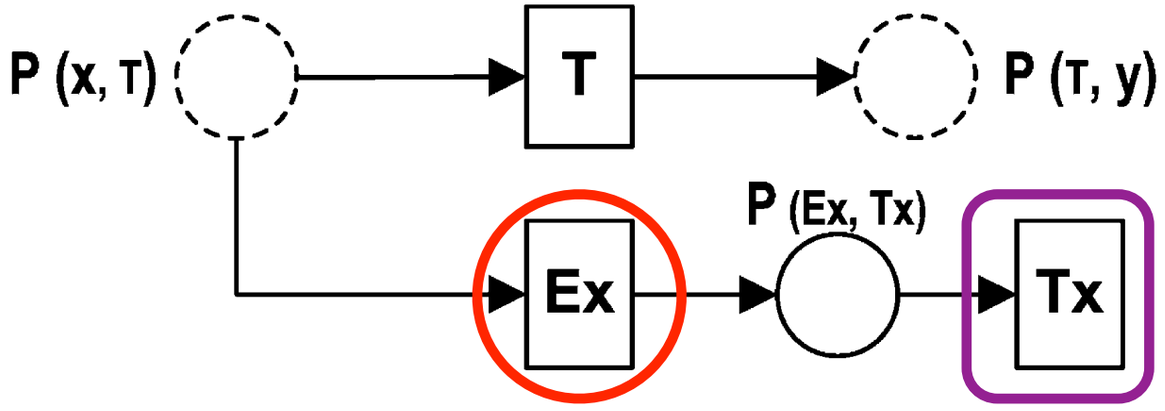


Sub-processes



Exception handling: single task





Exception handling: sub-processes