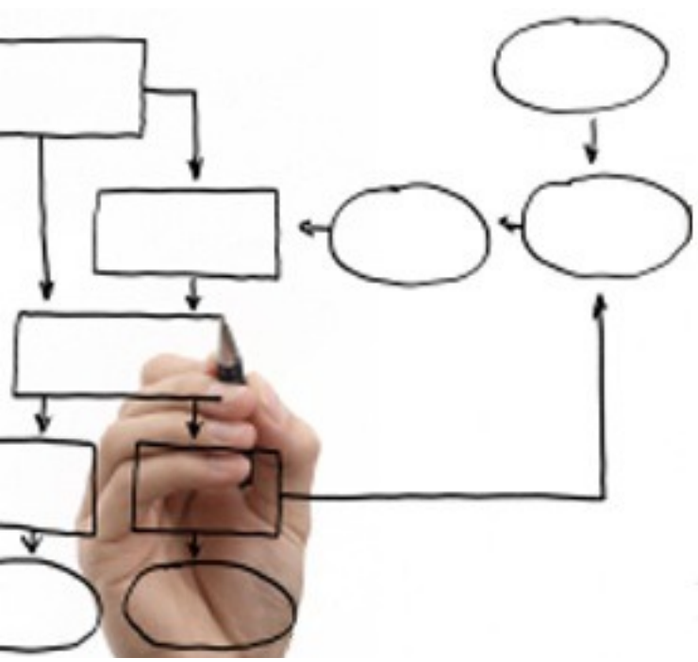
Methods for the specification and verification of business processes

MPB (6 cfu, 295AA)



Roberto Bruni

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

12 - Workflow nets

Object

We study some special kind of Petri nets, that are suitable models of workflows

There are many, many variants of Petri nets

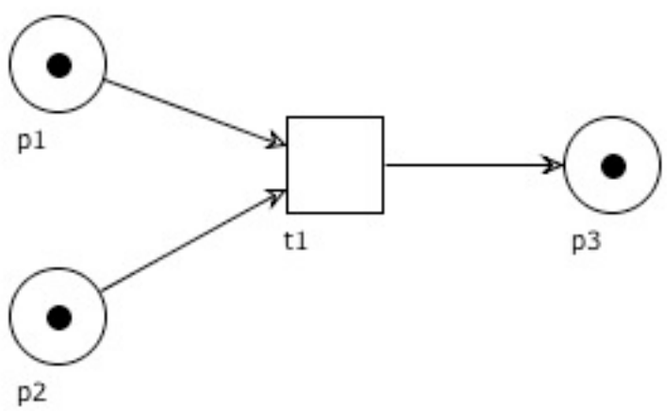
Condition / Event Systems

A **C/E system** is a Petri net whose places have all capacity equal to 1
(i.e. each place can contain one token at most)

Markings are just subsets of P (not multisets)

Firing rule is more restrictive:
t is enabled at M if •t ⊆ M **and t• ∩ M = ∅**

Is t_1 enabled?



Place / Transition Petri nets

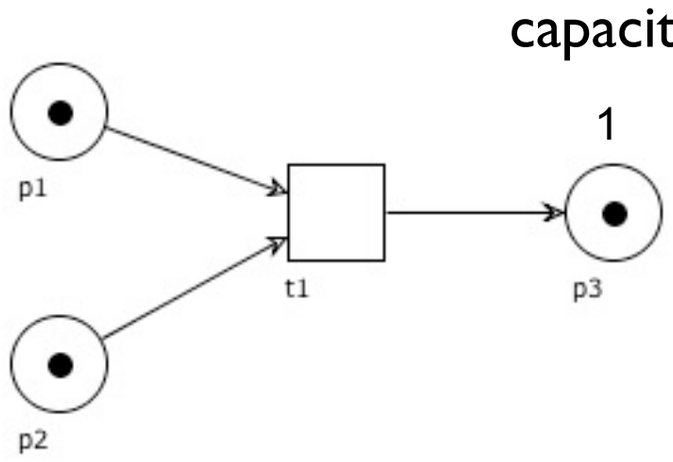
A **P/T net** is a Petri net (P,T,F) together with a weight function w : F → Nat

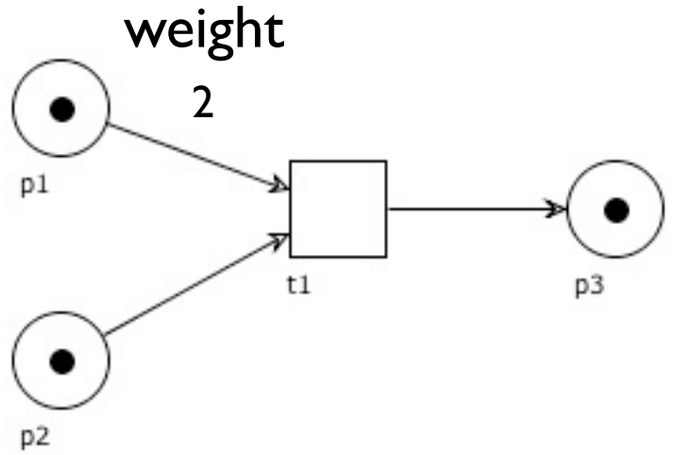
Firings consume and produce tokens according to the weight function

Sometimes a place capacity function c : P → Nat ∪ {∞} is also considered

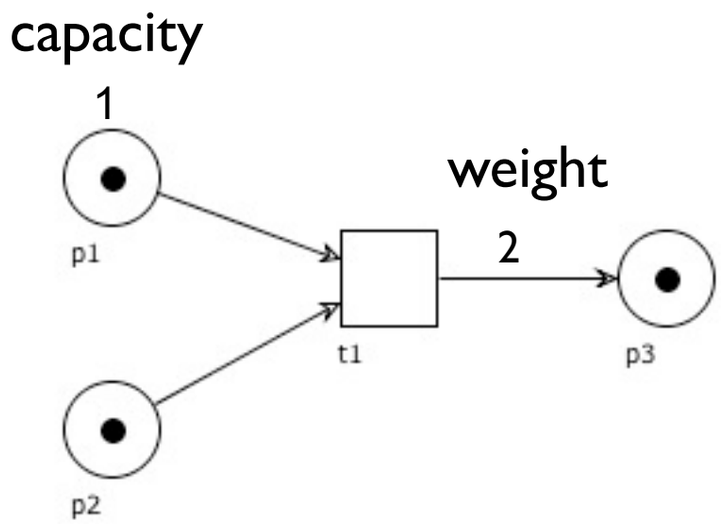
Firings cannot lead to markings where the capacity of a place is exceeded

P/T net: examples



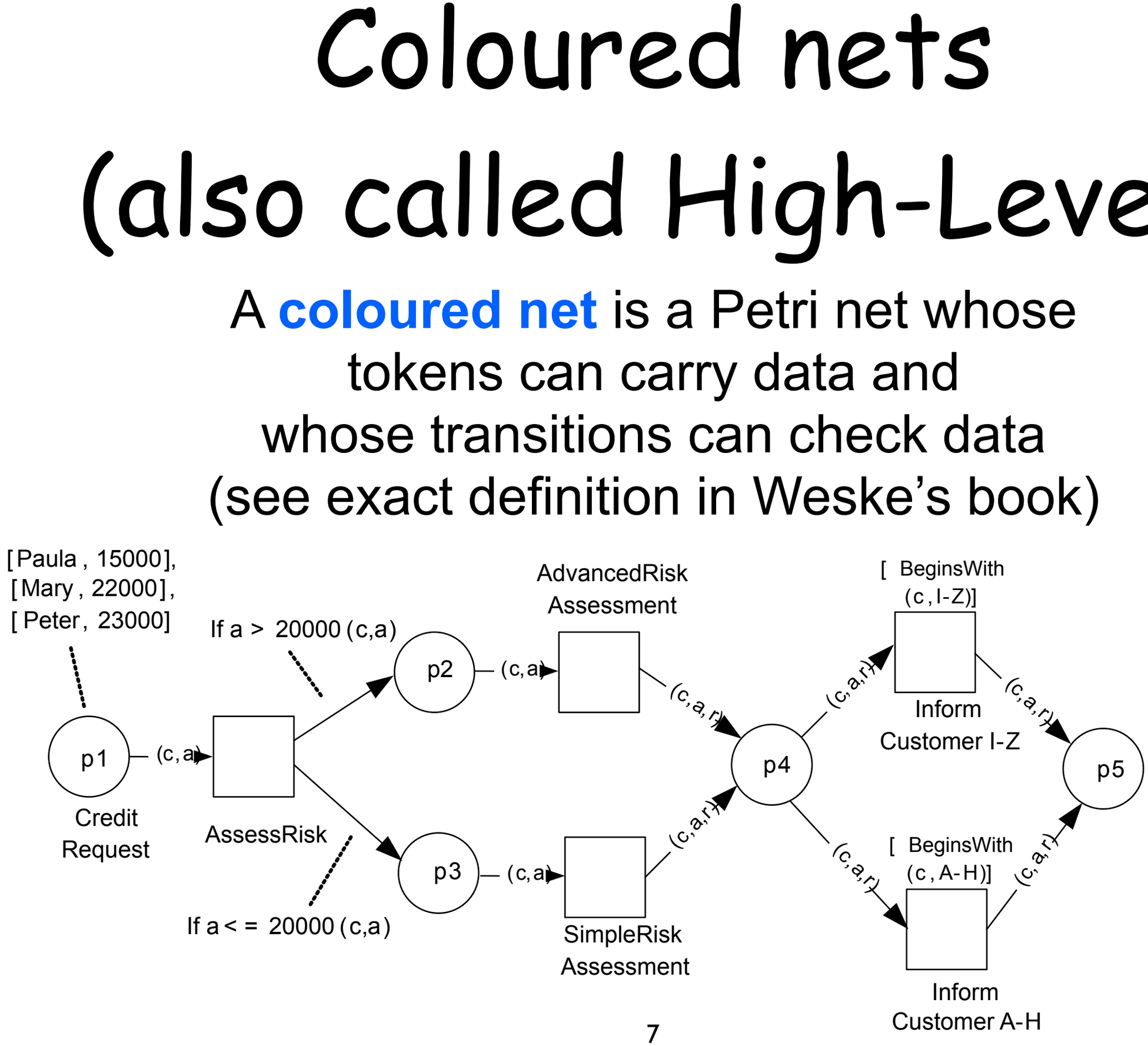


Is t_1 enabled?



Coloured nets
(also called High-Level)

A **coloured net** is a Petri net whose
tokens can carry data and
whose transitions can check data
(see exact definition in Weske's book)



Workflow nets

Workflow nets features

Aim: To ease the representation of business processes

Formal (unambiguous) semantics

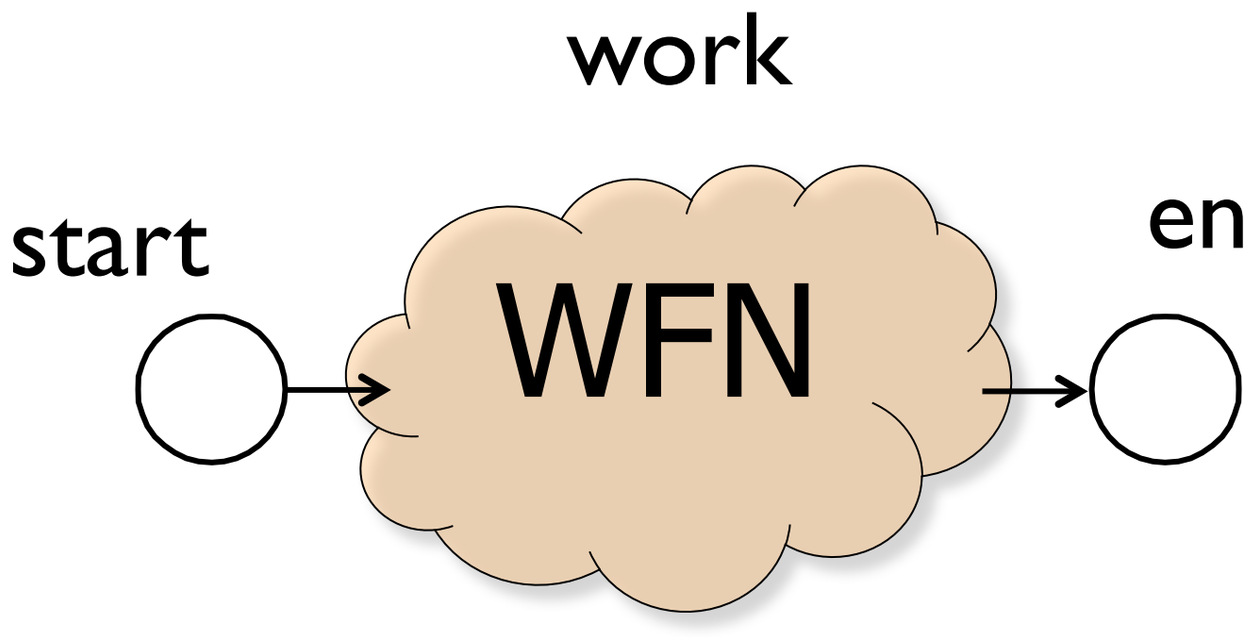
Decorated graphical representation

Structural restrictions

Efficient analysis of process properties

Tool independence (.pnml standard)

Workflow net: idea



Workflow net

Definition:
A Petri net (P, T, F) is called **workflow net** if:

1. there is a distinguished *initial place* $i \in P$ with $\bullet i = \emptyset$
2. there is a distinguished *final place* $o \in P$ with $o\bullet = \emptyset$
3. every other place and transition belongs to a path from i to o

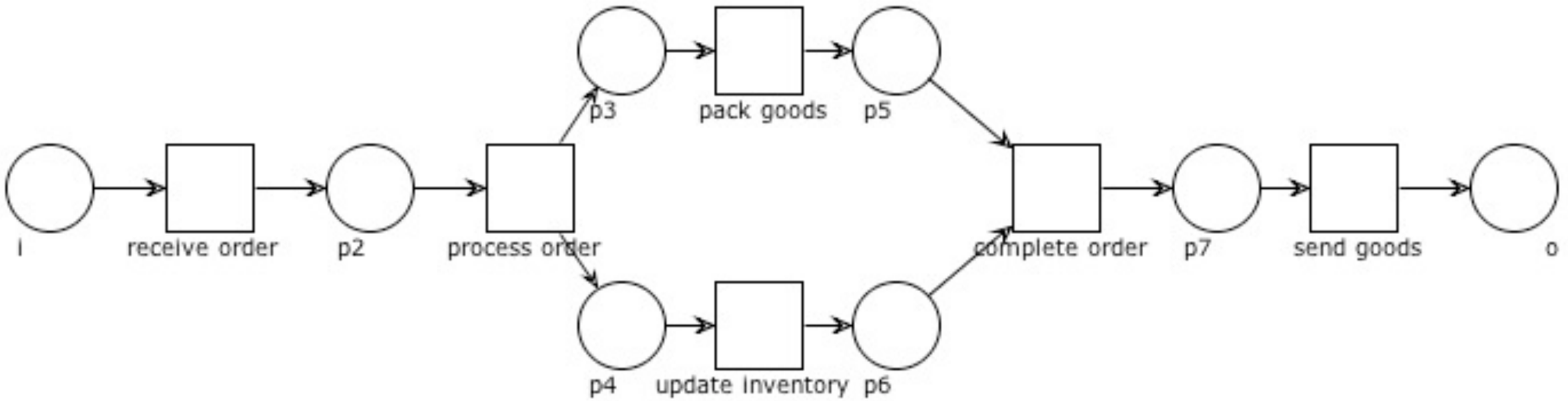
Workflow net: Rationale

1. a token in i represents a process instance not yet started
2. a token in o represents a finished case
3. each place and each transition can participate in a case

Definition:
A Petri net (P, T, F) is called **workflow net** if:

1. there is a distinguished *initial place* $i \in P$ with $\bullet i = \emptyset$
2. there is a distinguished *final place* $o \in P$ with $o\bullet = \emptyset$
3. every other place and transition belongs to a path from i to o

WF net: Example



Basic properties

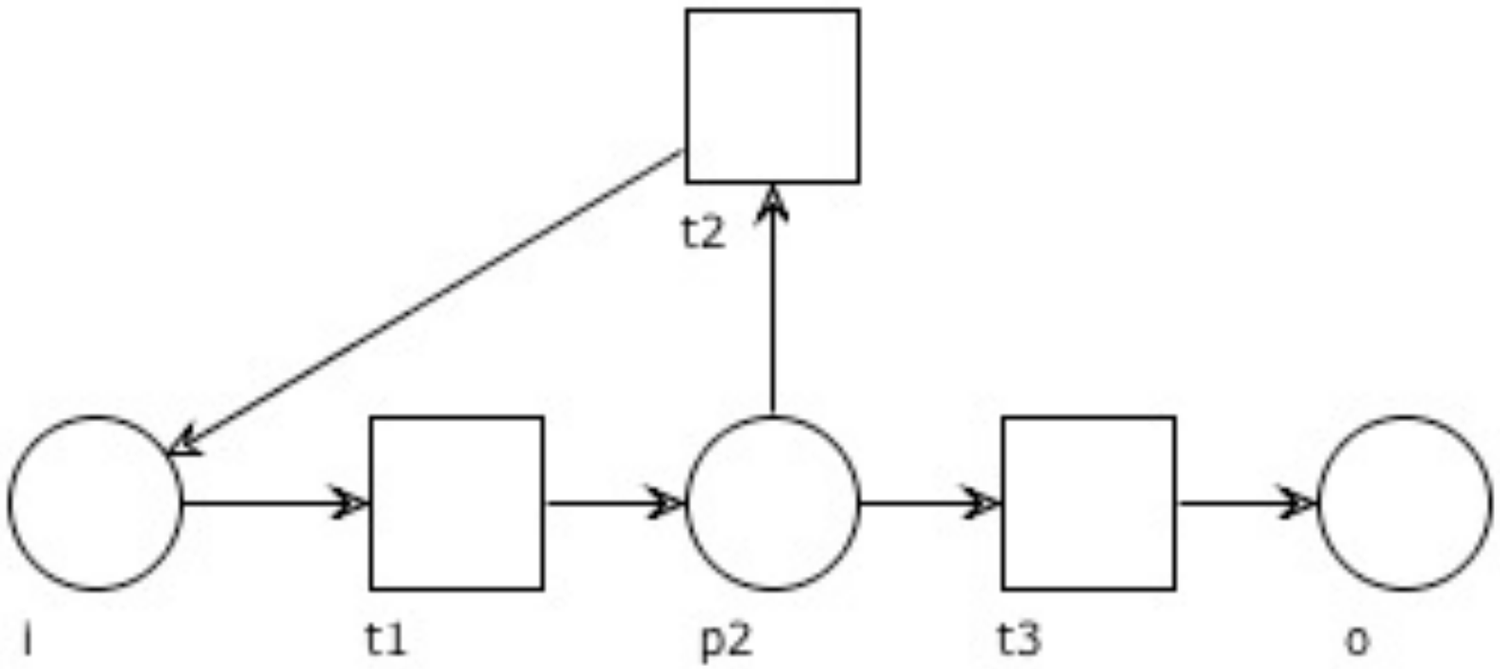
Lemma: In a workflow net there is a **unique** node with no incoming arc

Lemma: In a workflow net there is a **unique** node with no outgoing arc

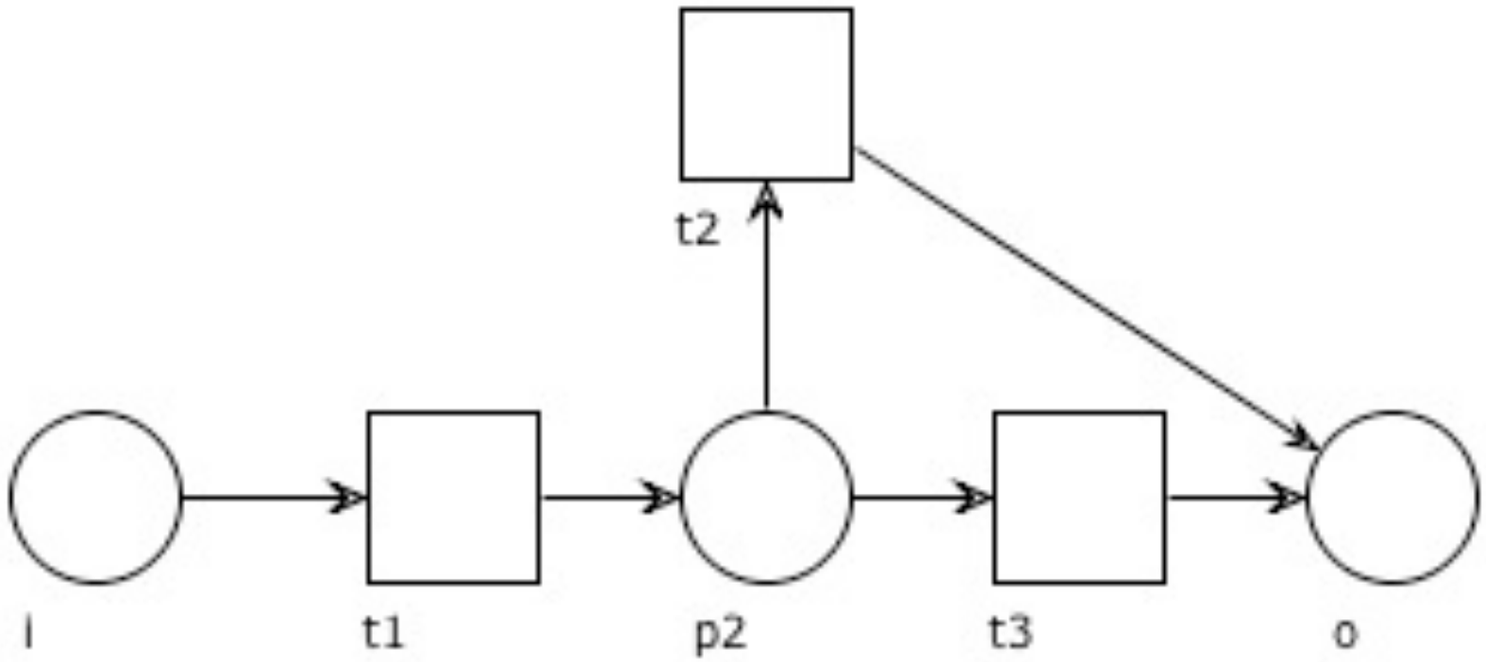
Exercise: Guess which nodes are those

Exercise: Prove the above lemmas (hint: suppose the nodes are not unique reach a contradiction)

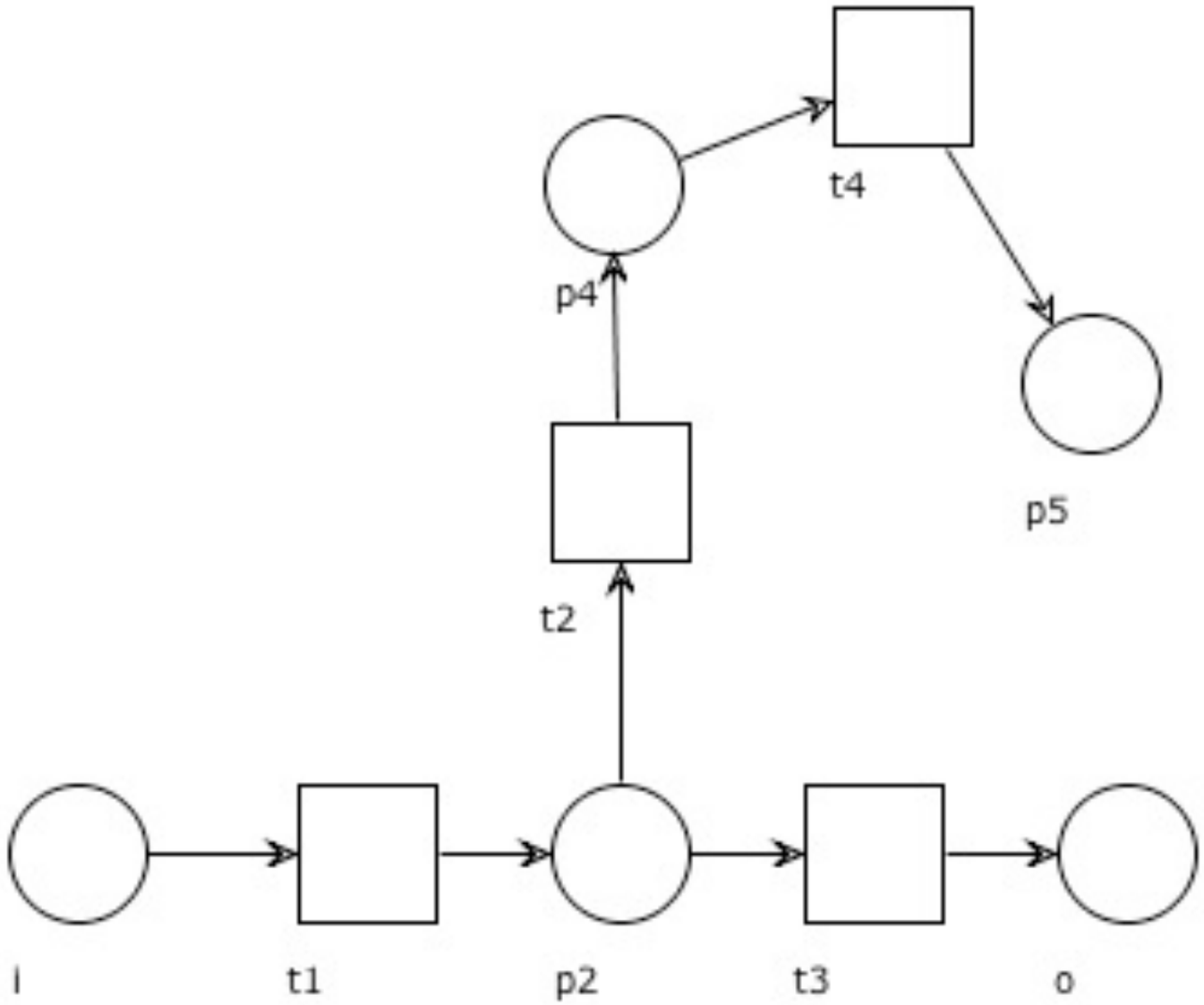
Exercise: WF net?



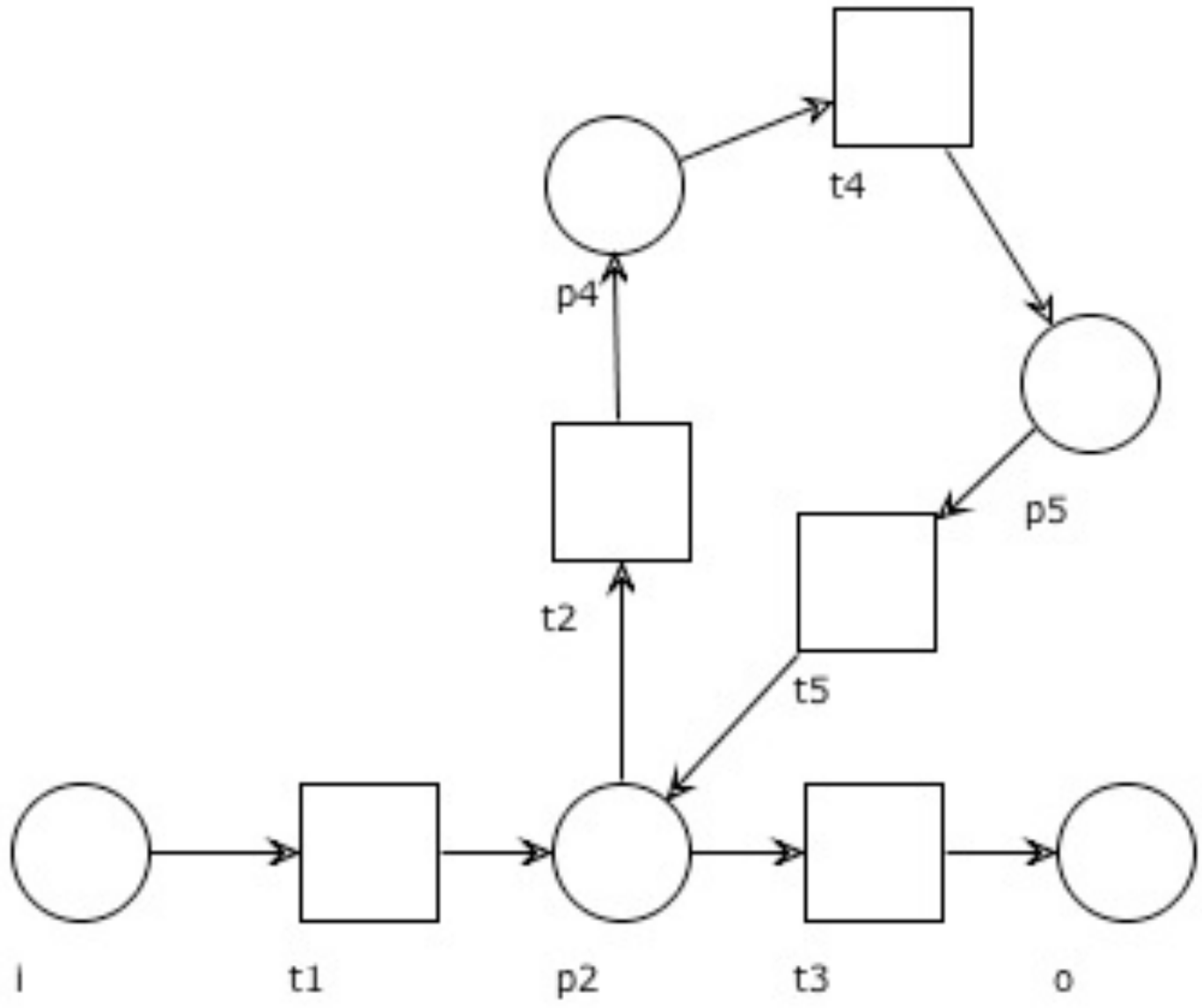
Exercise: WF net?



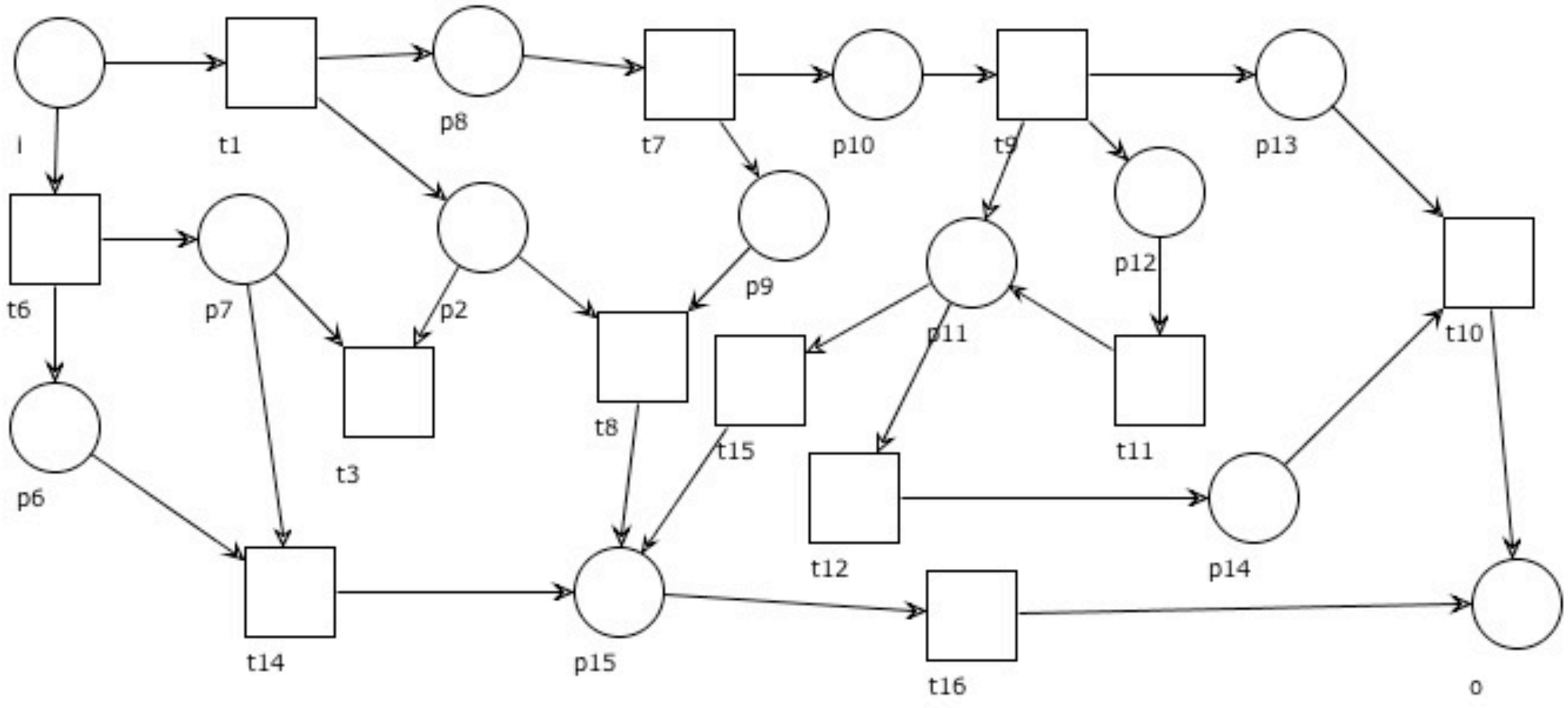
Exercise: WF net?



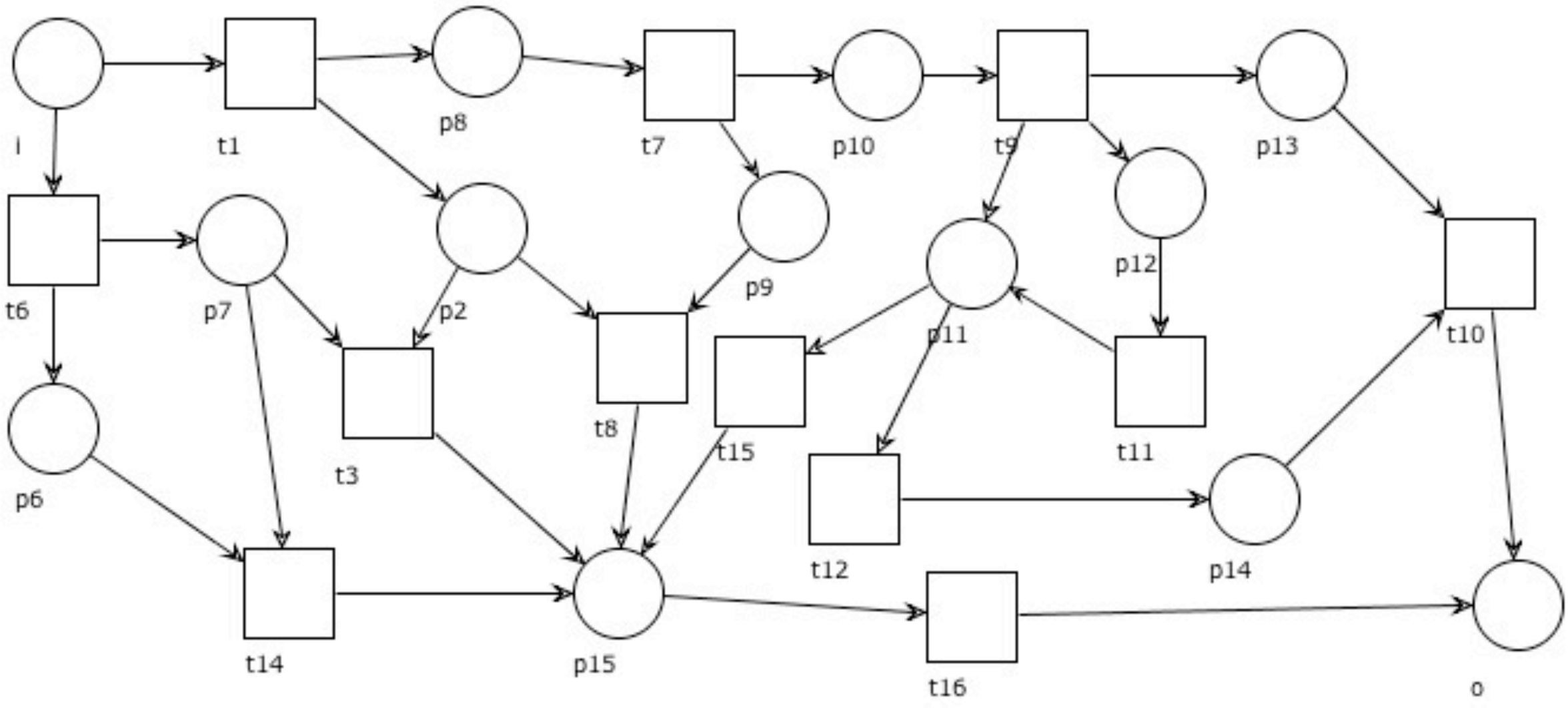
Exercise: WF net?



Exercise: WF net?

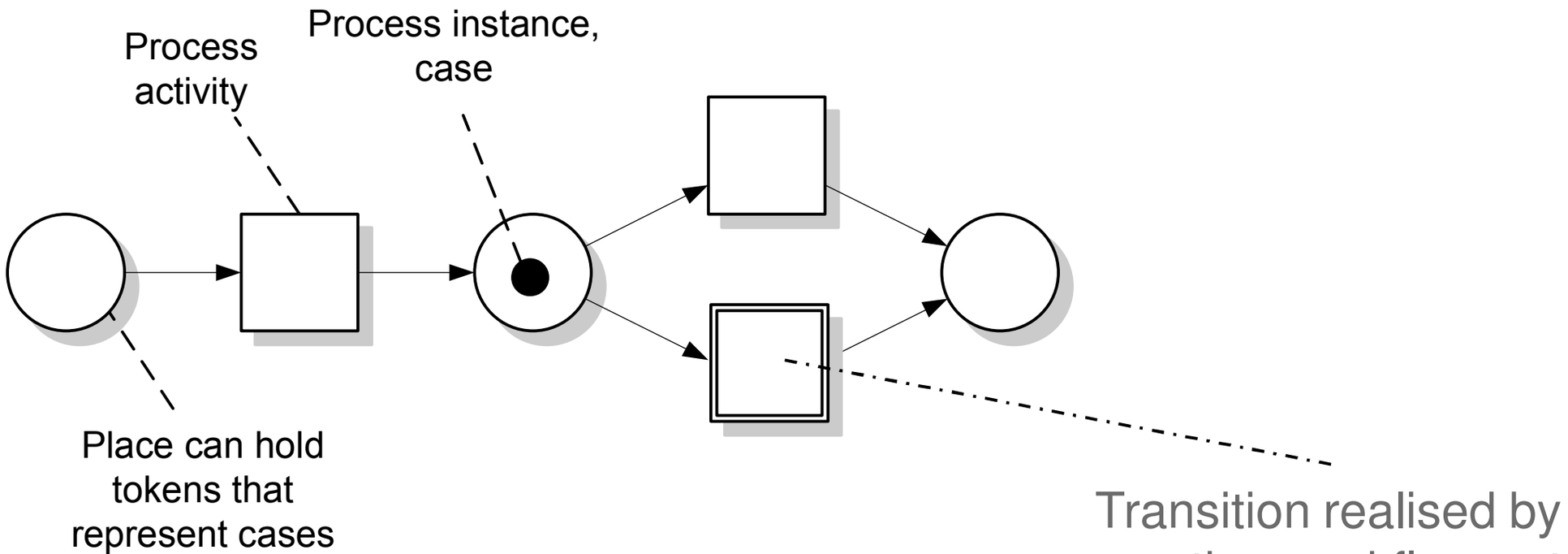


Exercise: WF net?

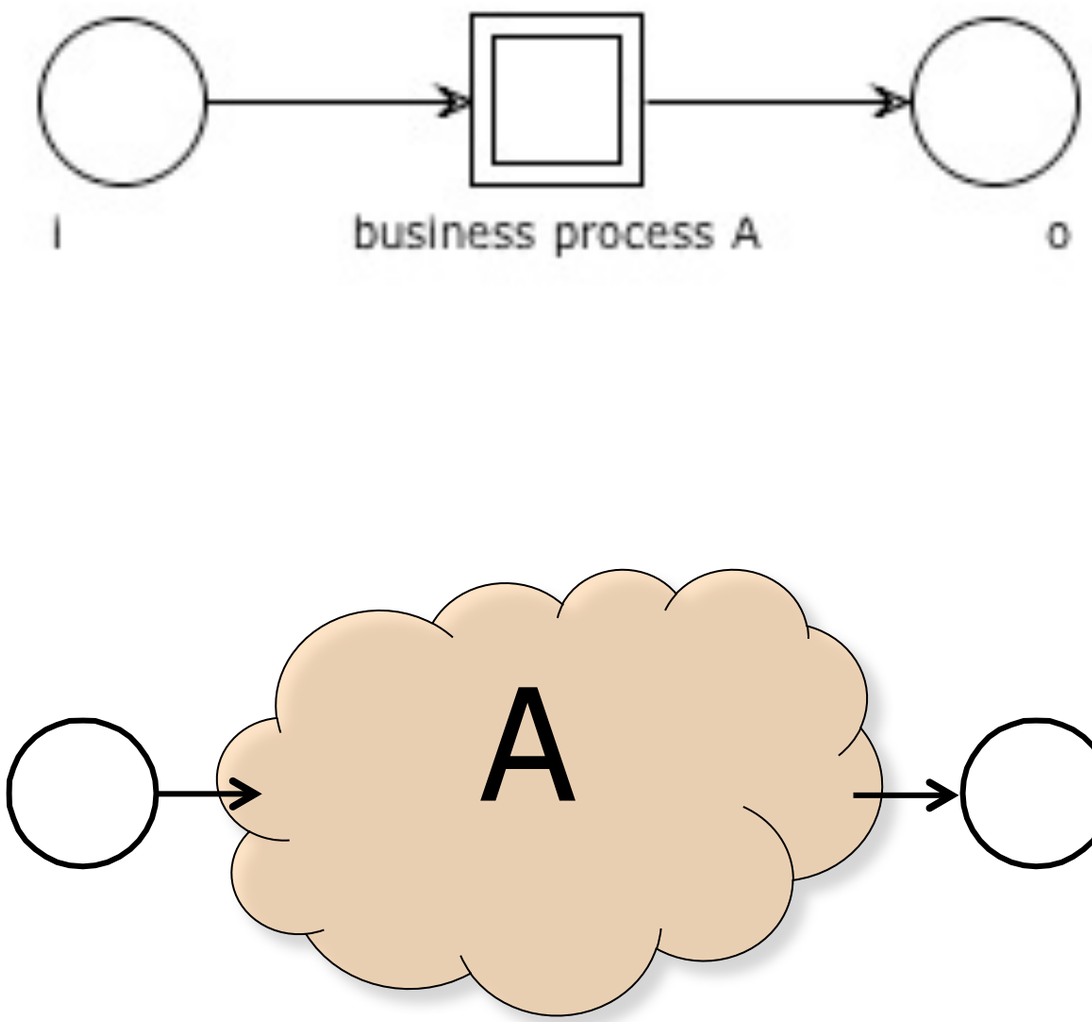


Hierarchical structuring

Uniqueness of entry / exit point facilitate the hierarchical structuring of WF nets



Abstract view



Typical control flow aspects

Sequencing

Parallelism (AND-split + AND-join)

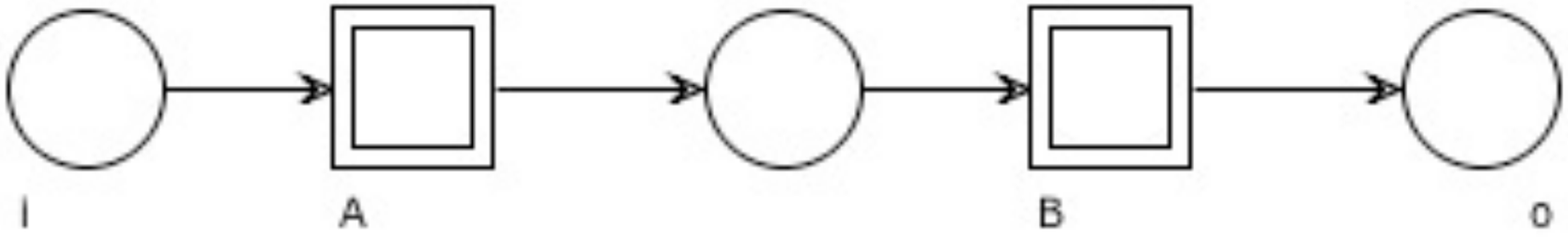
Selection (XOR-split + XOR-join)

Iteration (XOR-join + XOR-split)

Capacity constraints:
Feedback loop
Mutual exclusion
Alternating

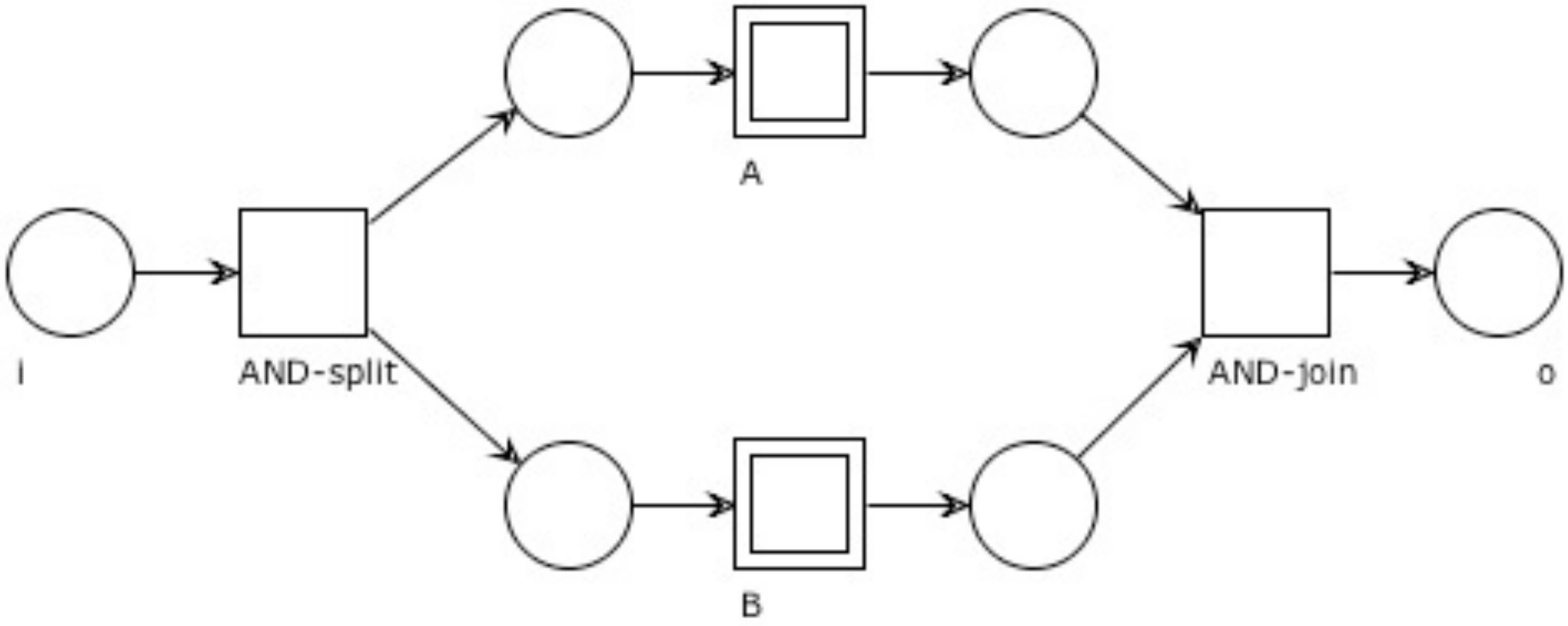
Sequencing

B is executed after A



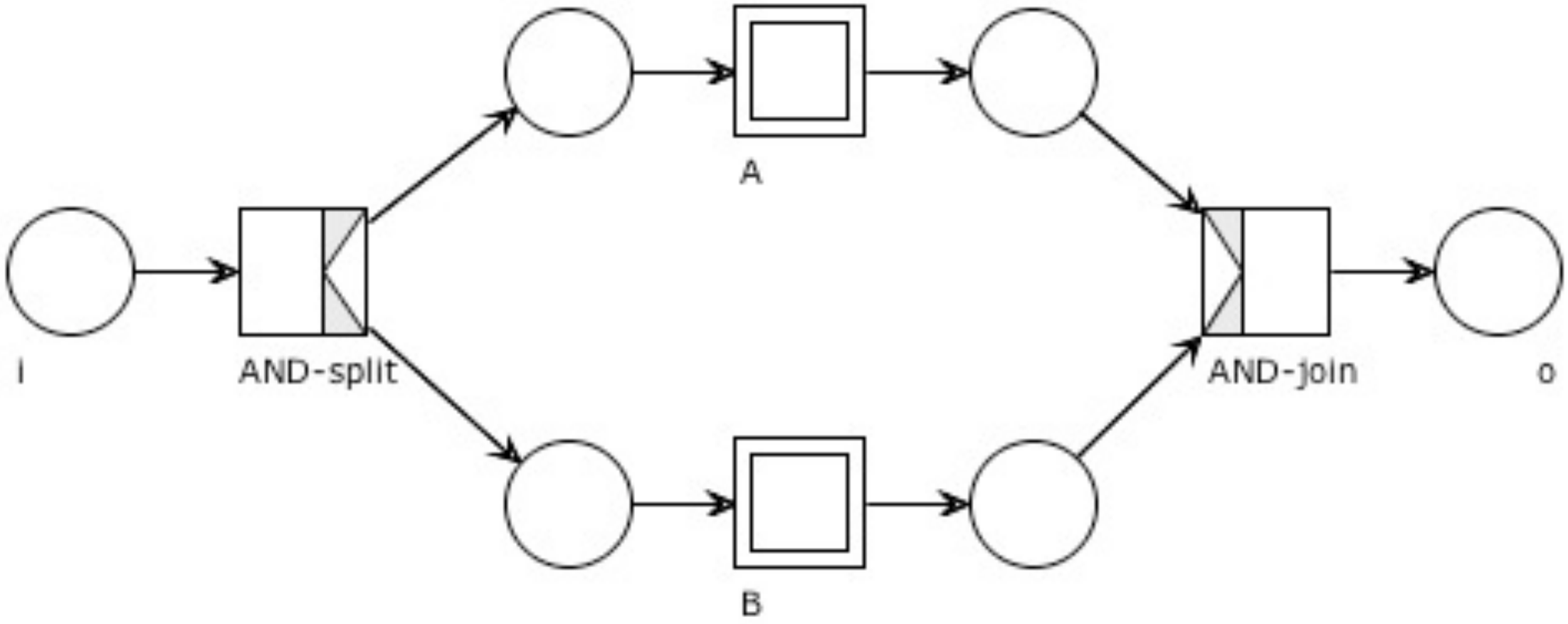
Parallelism (AND-split + AND-join)

A and B are both executed in no particular order



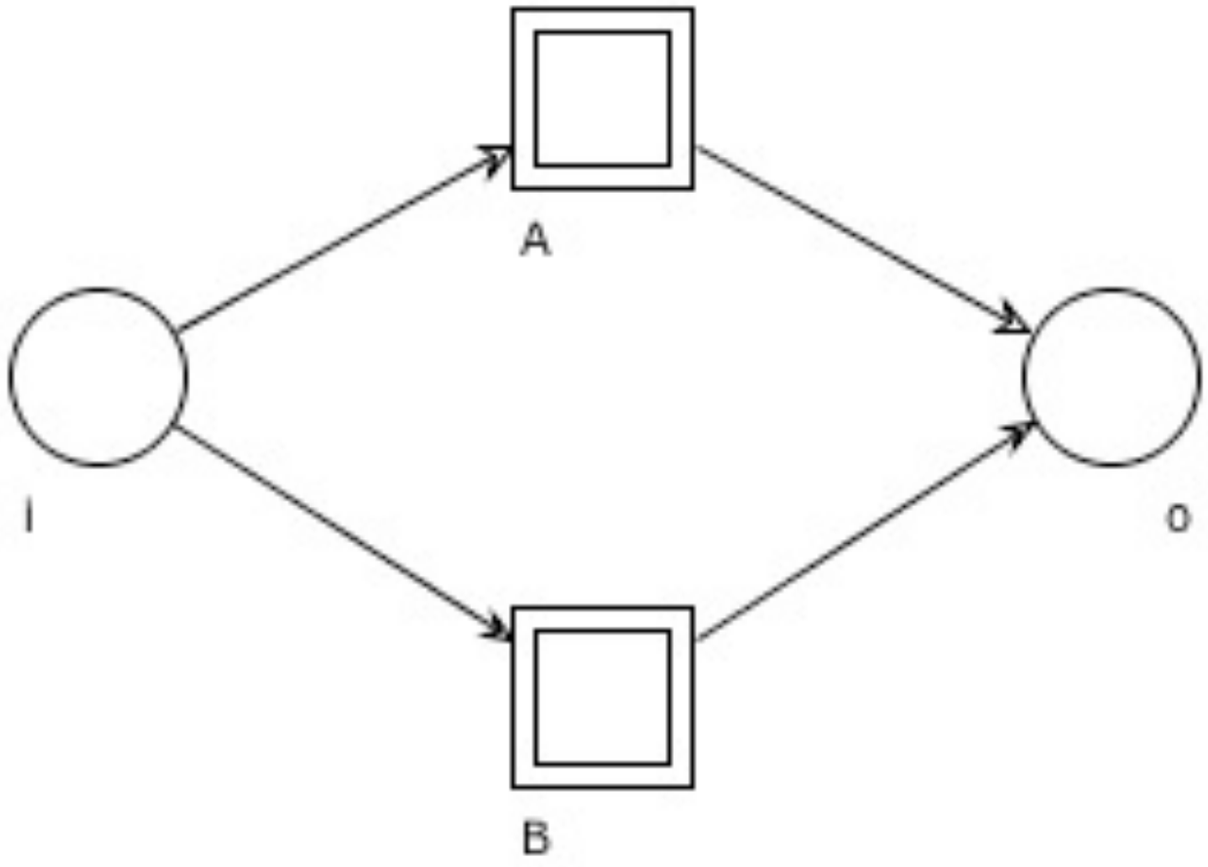
Parallelism ("sugared" version)

Decorated version for business process stakeholders



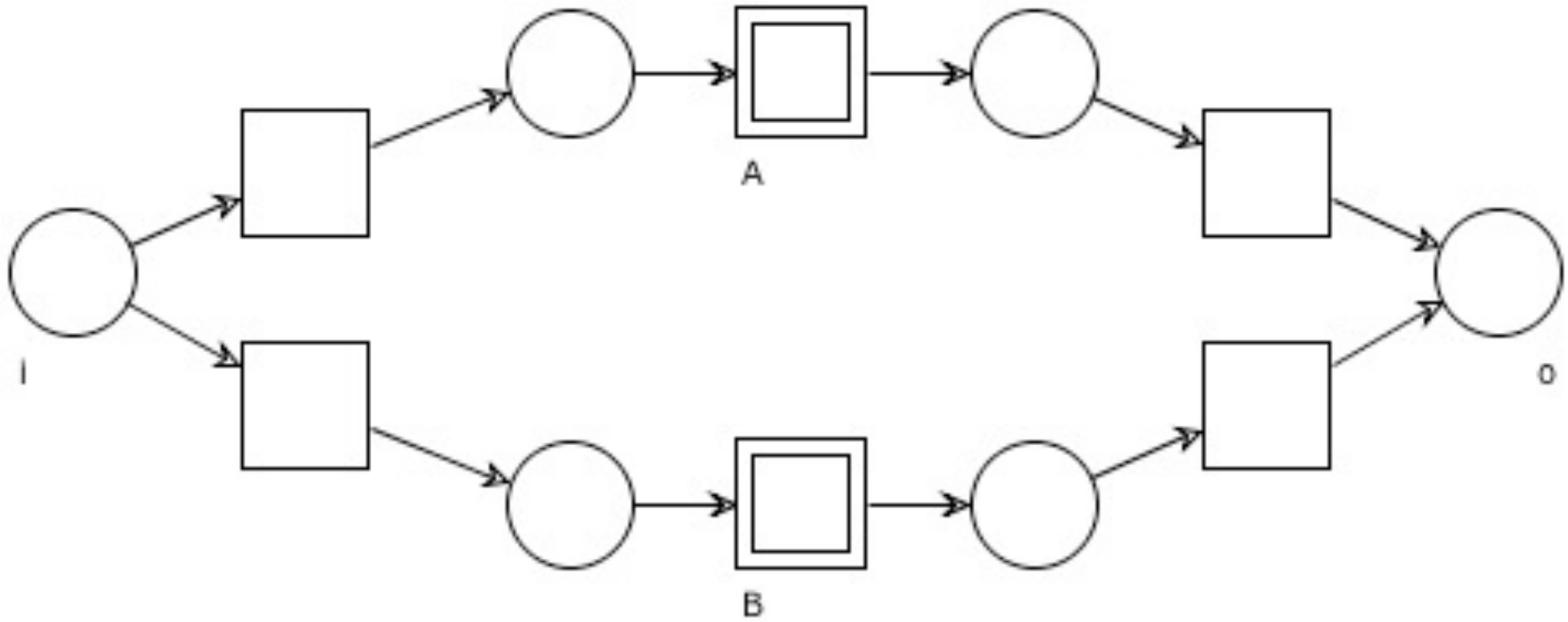
Deferred choice (XOR-split + XOR-join)

Either A or B is executed (choice is **implicit**)



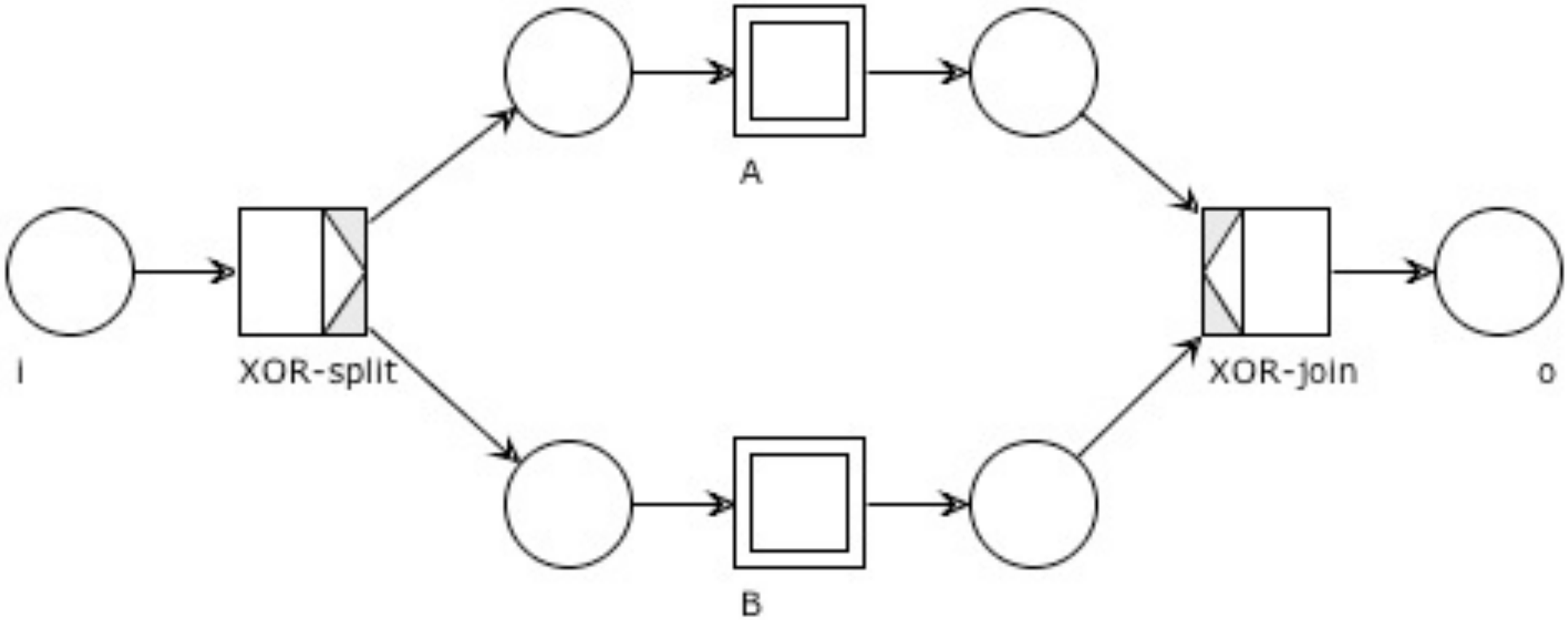
Explicit choice (XOR-split + XOR-join)

Either A or B is executed (choice is **explicit**)



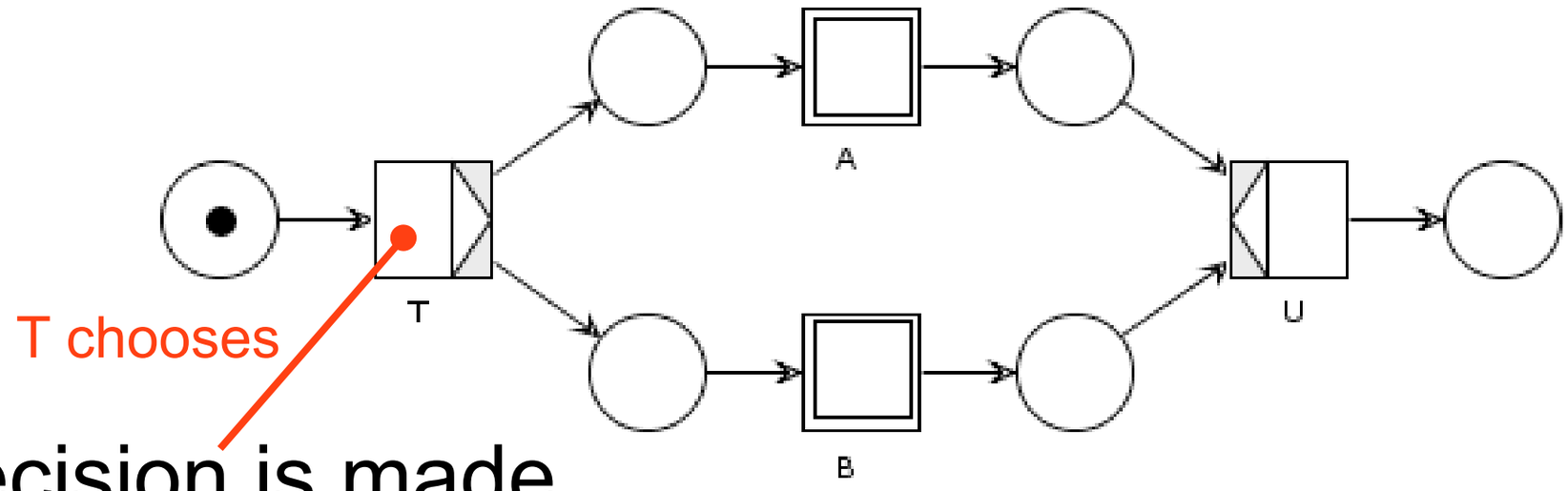
Choice
("sugared" version)

Decorated version for business process stakeholders

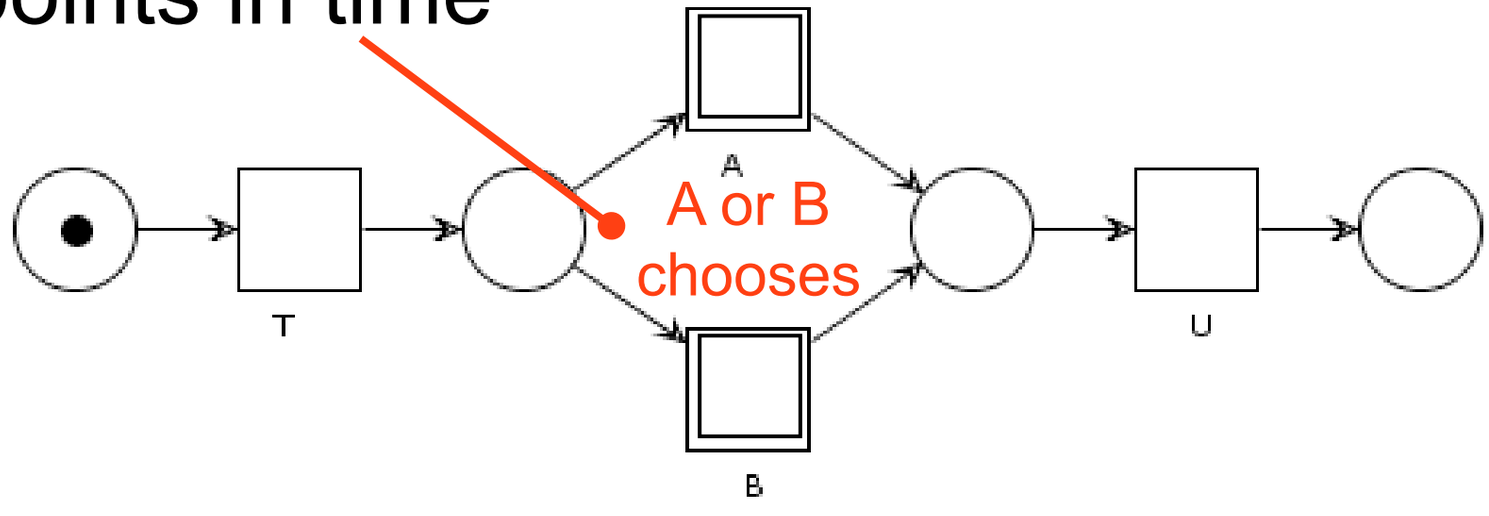


Remember

Explicit choice ≠ Implicit choice

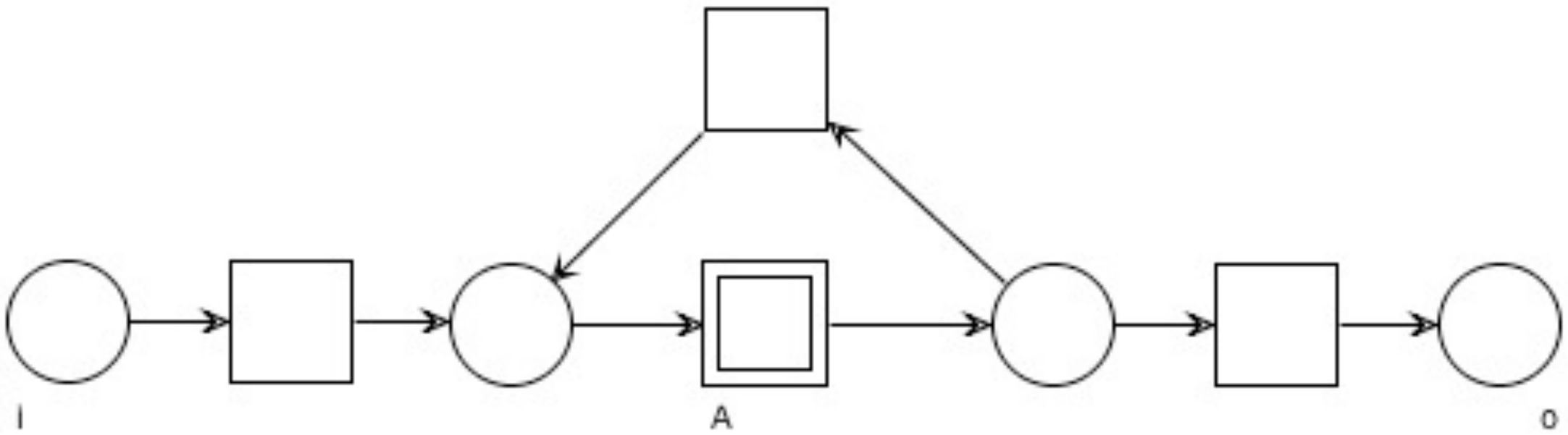


The decision is made at different points in time



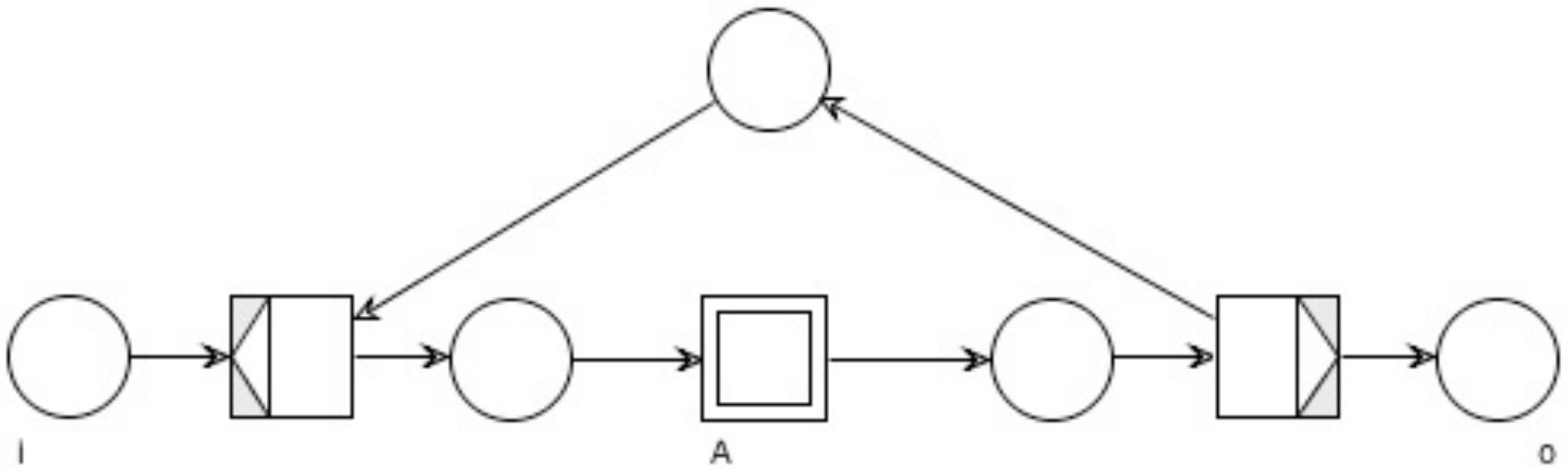
Iteration
(one or more time)

A is executed 1 or more time



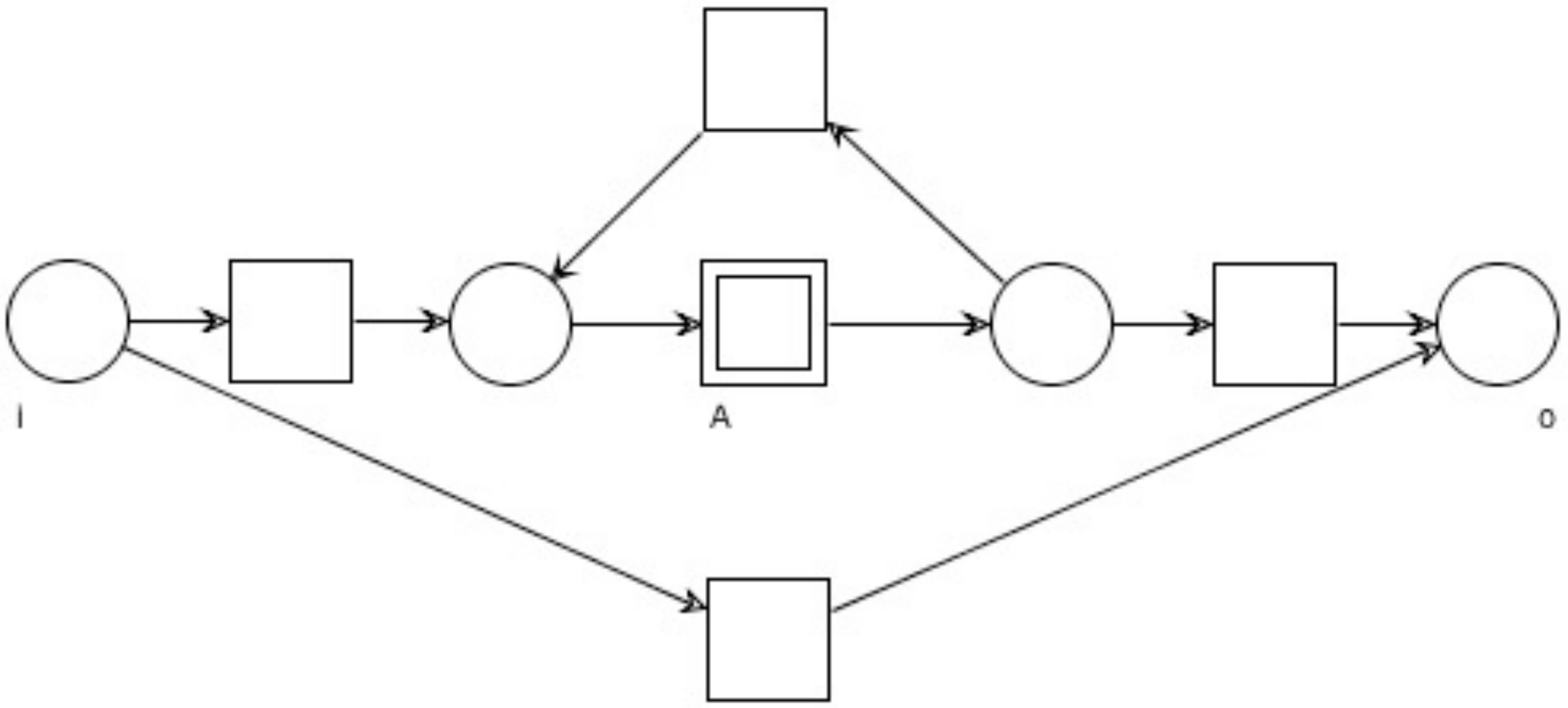
One-or-more iteration ("sugared" version)

Decorated version for business process stakeholders



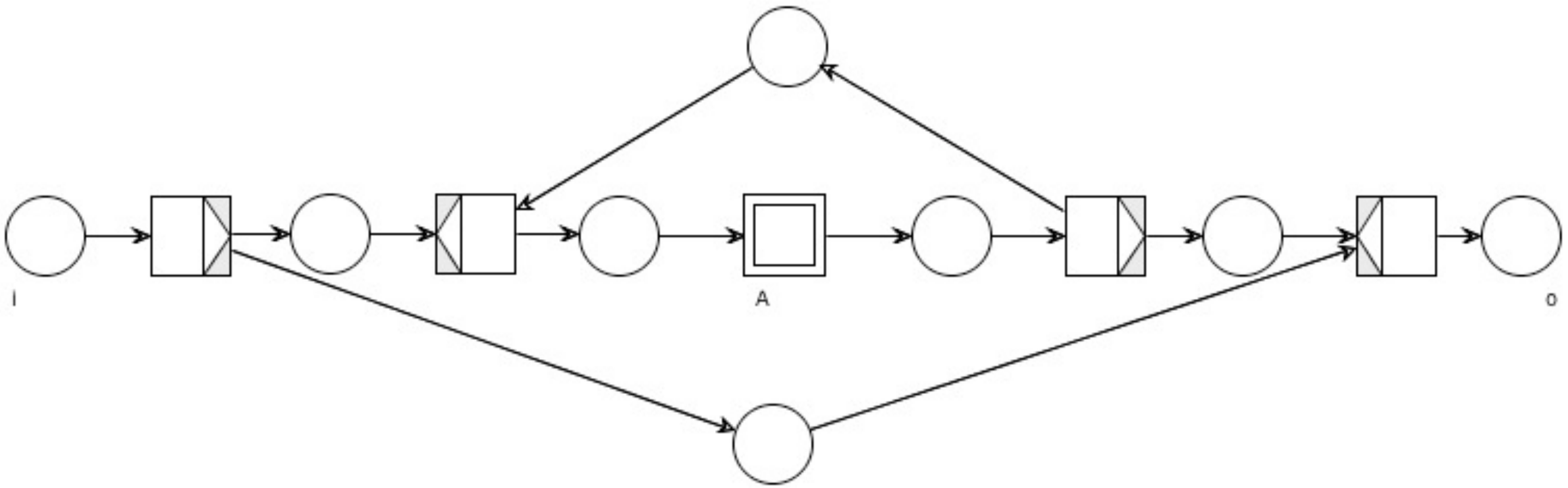
Iteration
(zero or more time)

A is executed 0 or more time



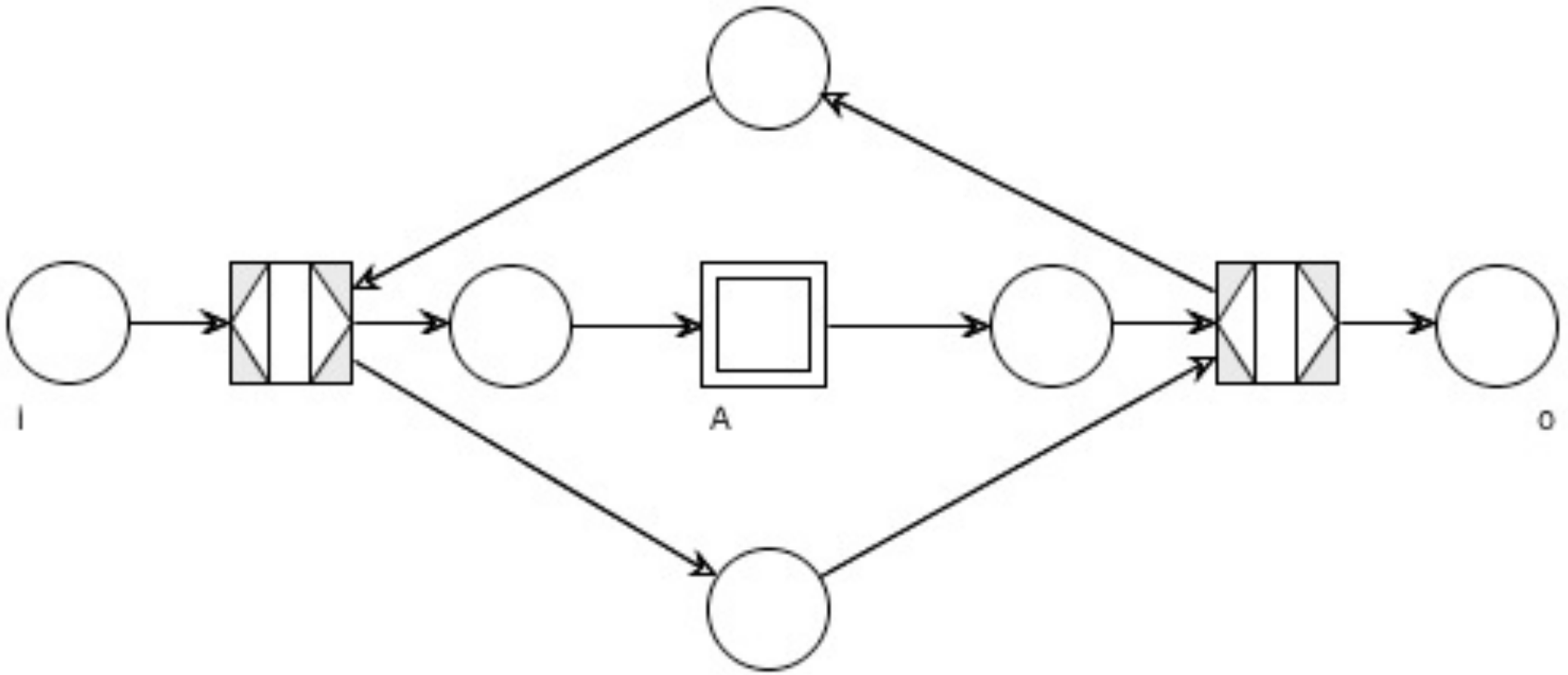
Zero-or-more iteration ("sugared" version)

Decorated version for business process stakeholders



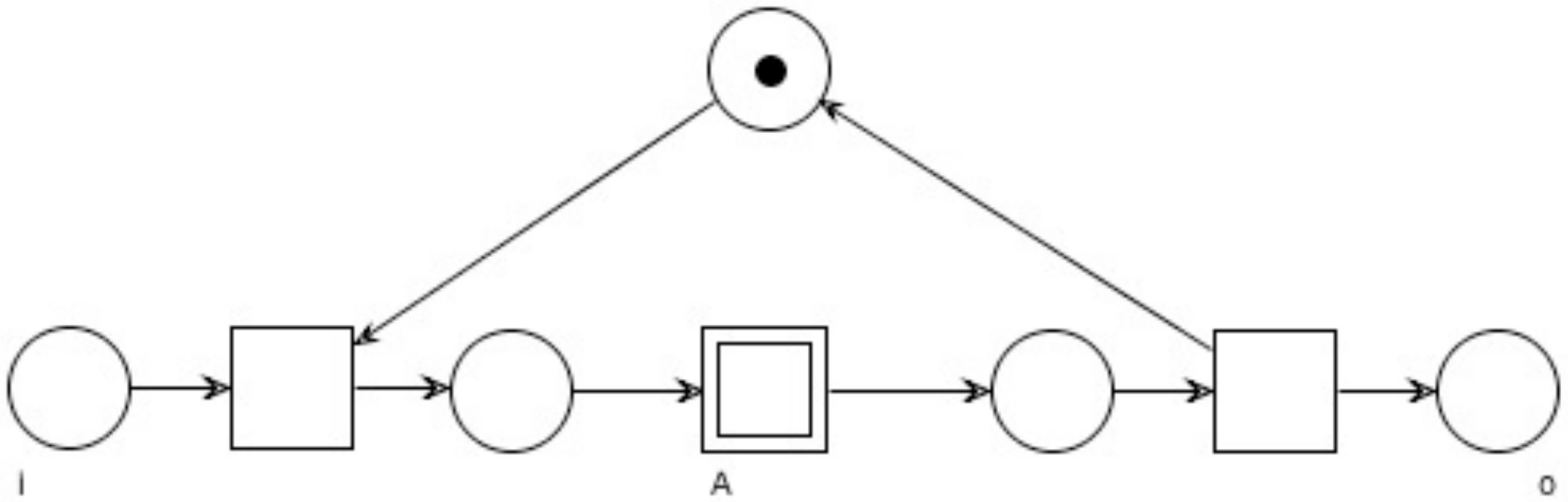
Zero-or-more iteration (simplified version)

Decorated version for business process stakeholders



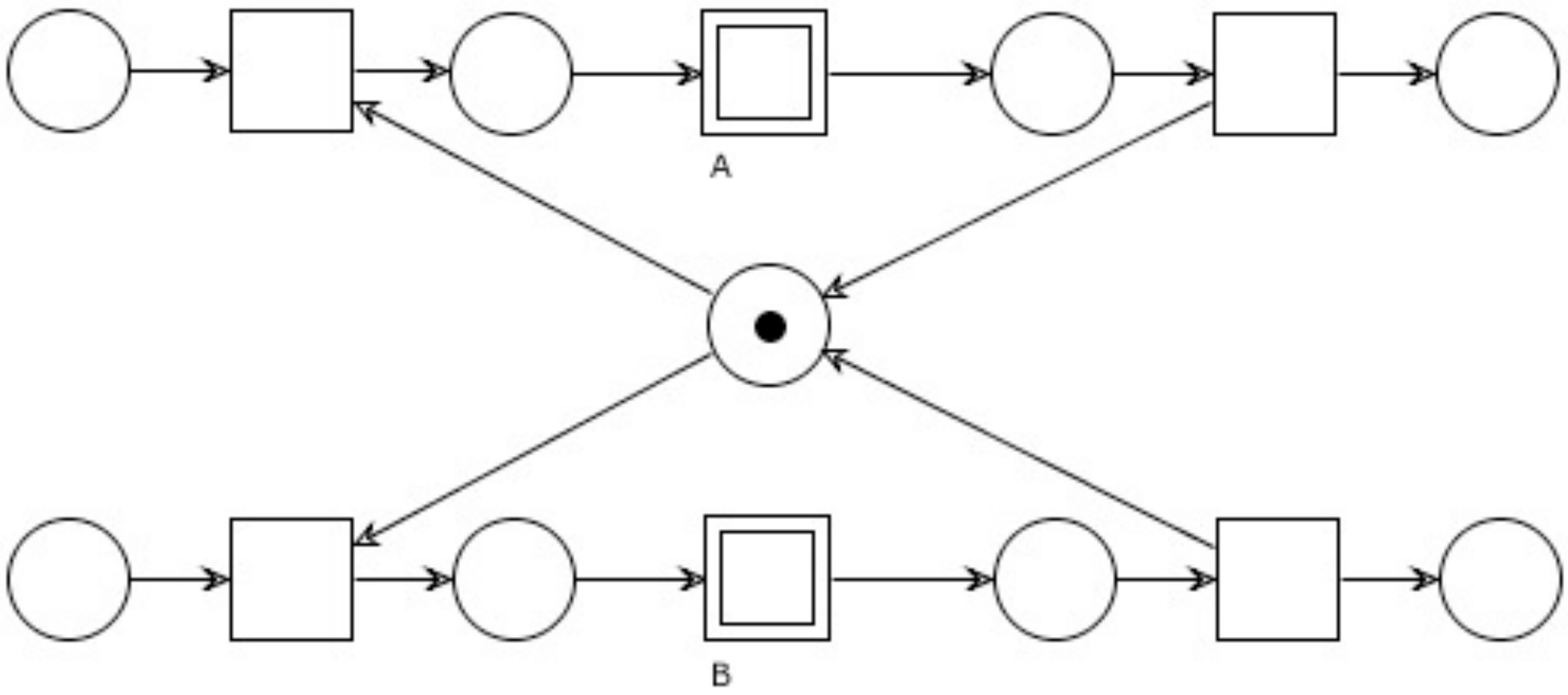
One serve per time

Multiple activations are handled one by one



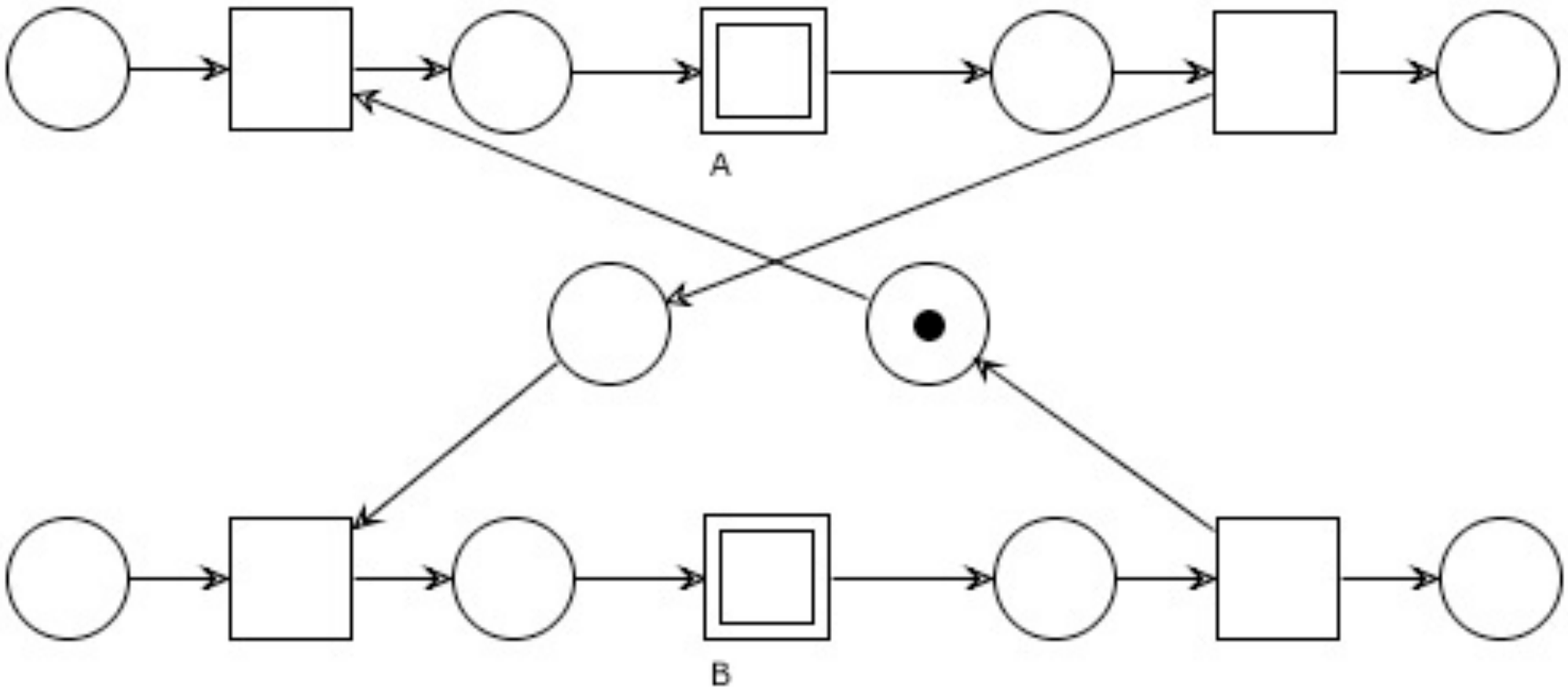
Mutual exclusion

A and B cannot execute concurrently

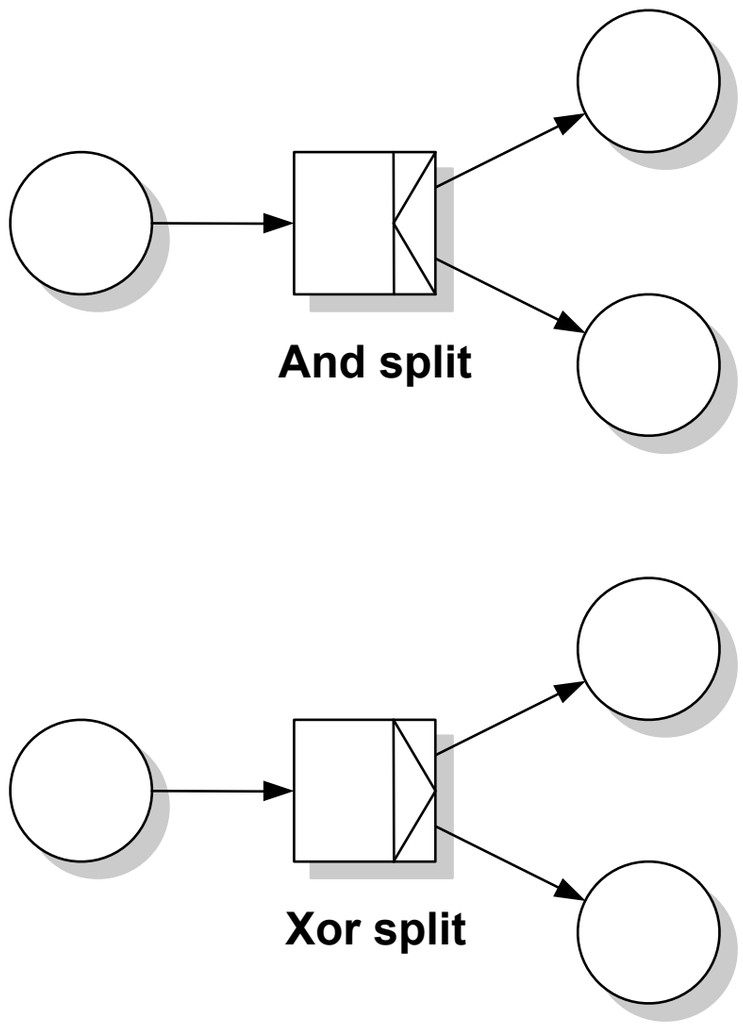


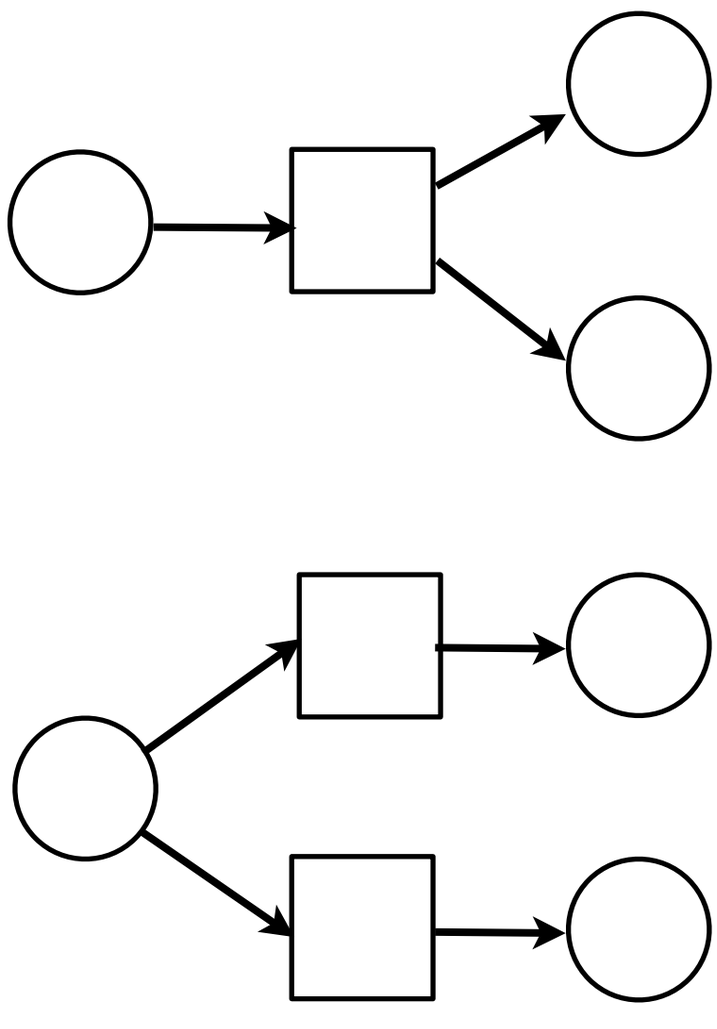
Alternation

A and B execute one time each (A first)



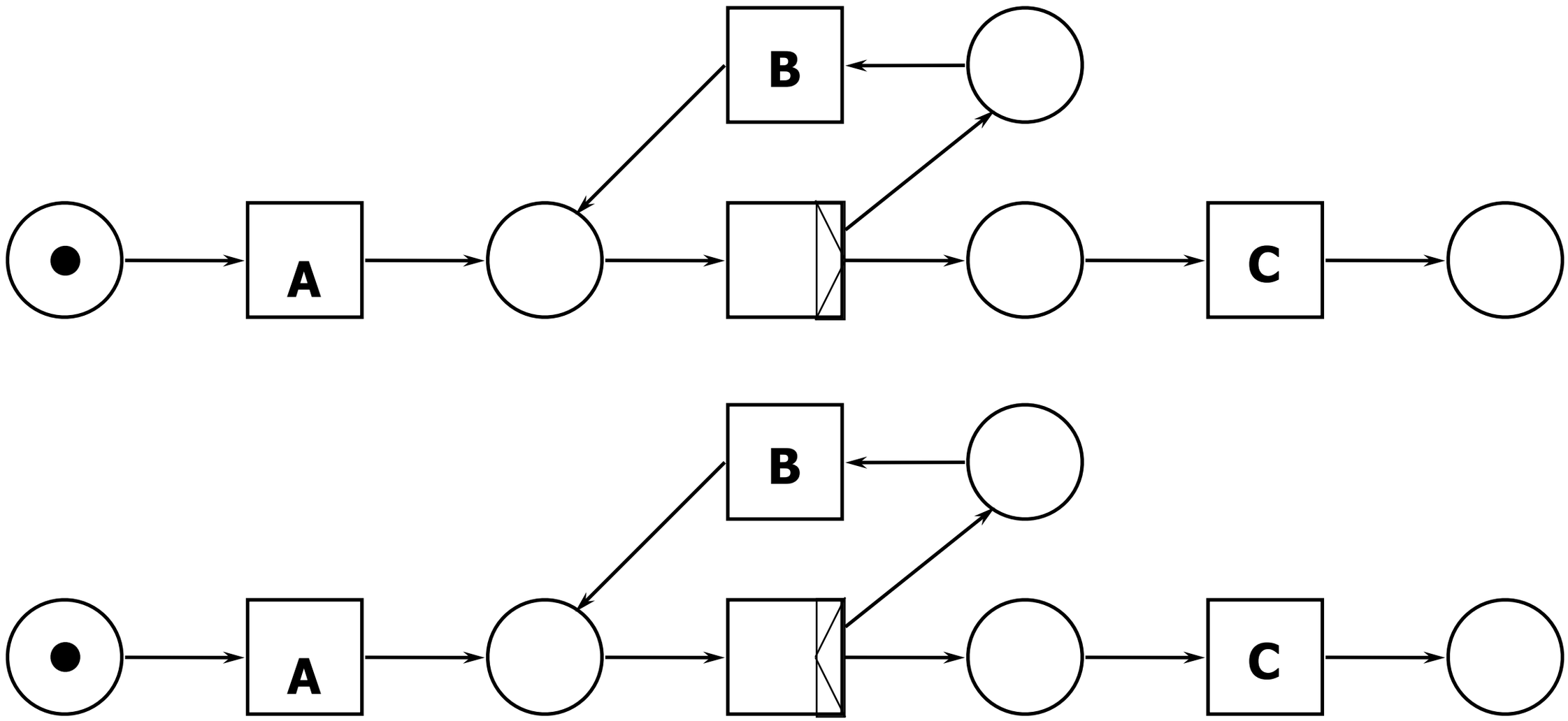
Syntax Sugar





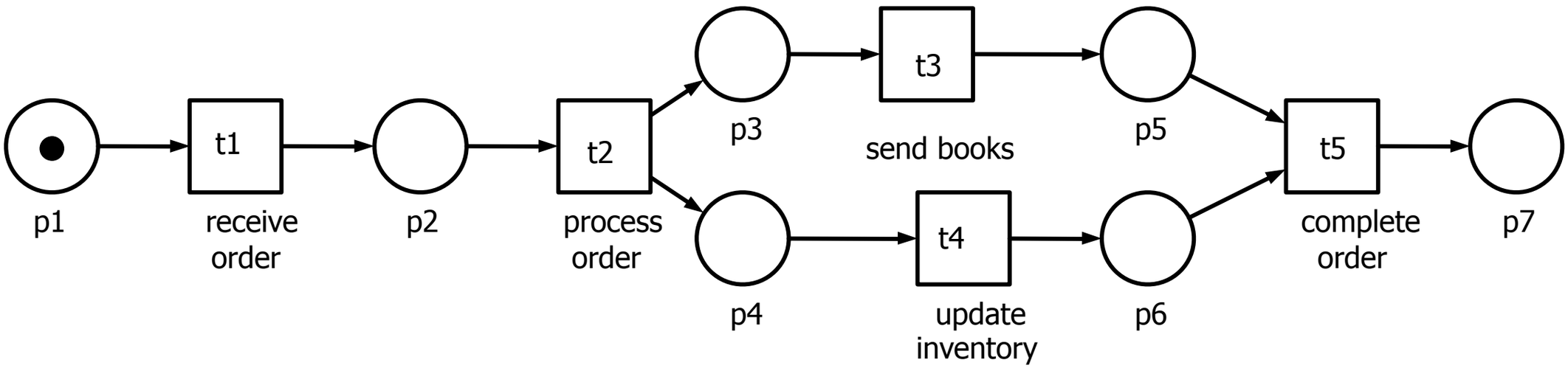
Question time

What's the difference (also in terms of firing sequences)?



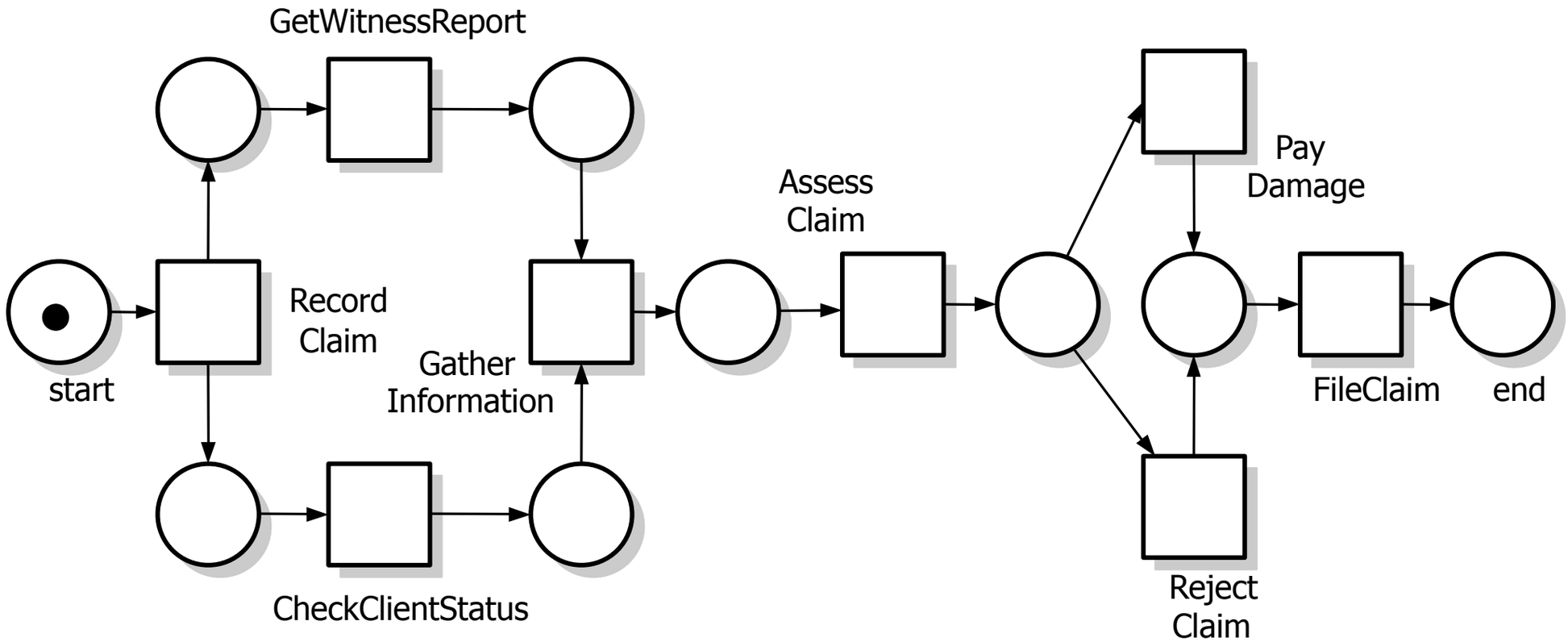
Exercises

- Which "patterns" can be found in the workflow net below?
- "Sugarize" the net
- Draw the corresponding Reachability Graph
- What are the possible firing sequences?



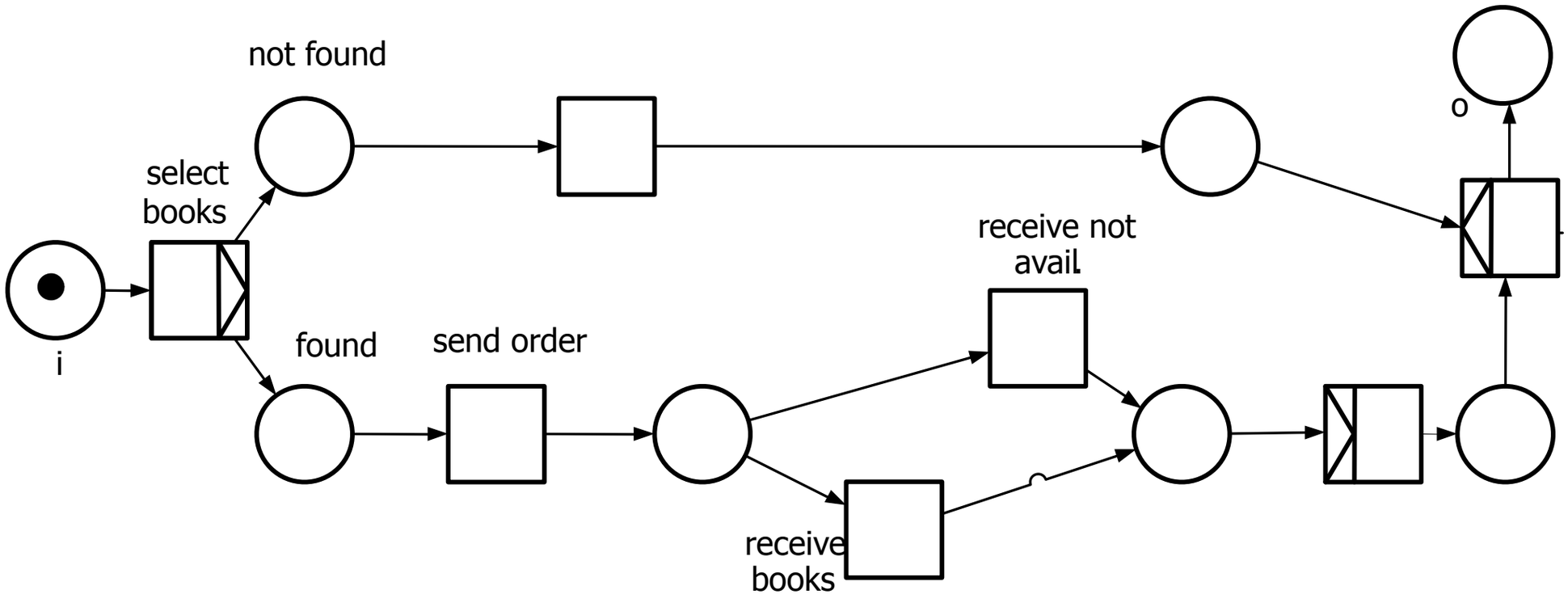
Exercises

- Which "patterns" can be found in the workflow net below?
- "Sugarize" the net (where it makes sense)
- Name all places and draw the Reachability Graph



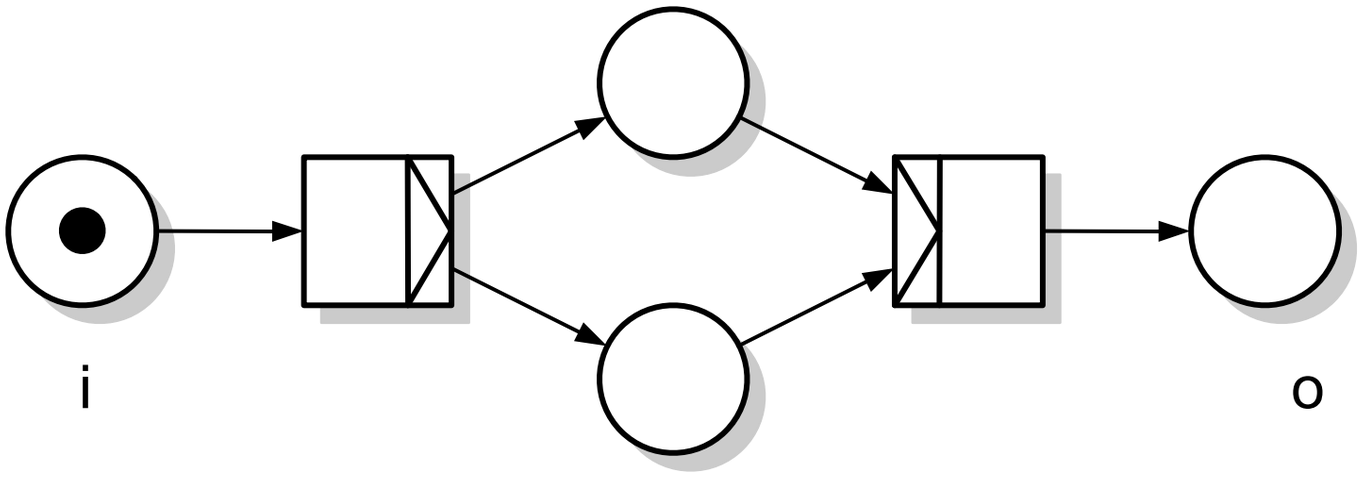
Exercises

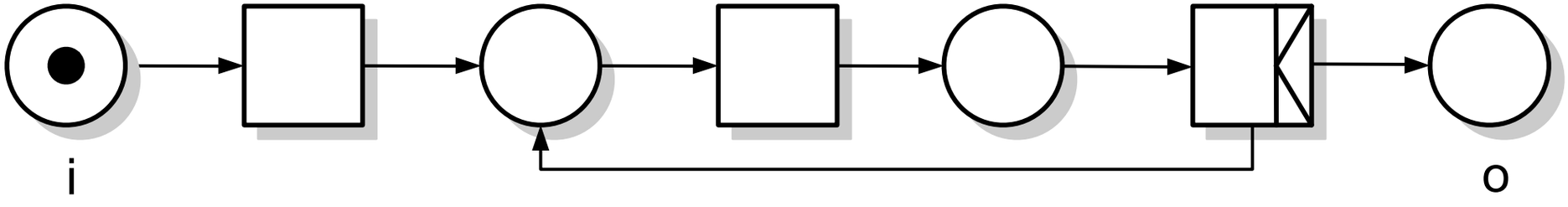
- "Desugarize" the workflow net below, then name all places and all transitions
- Draw the corresponding Reachability Graph
- What are the possible firing sequences?



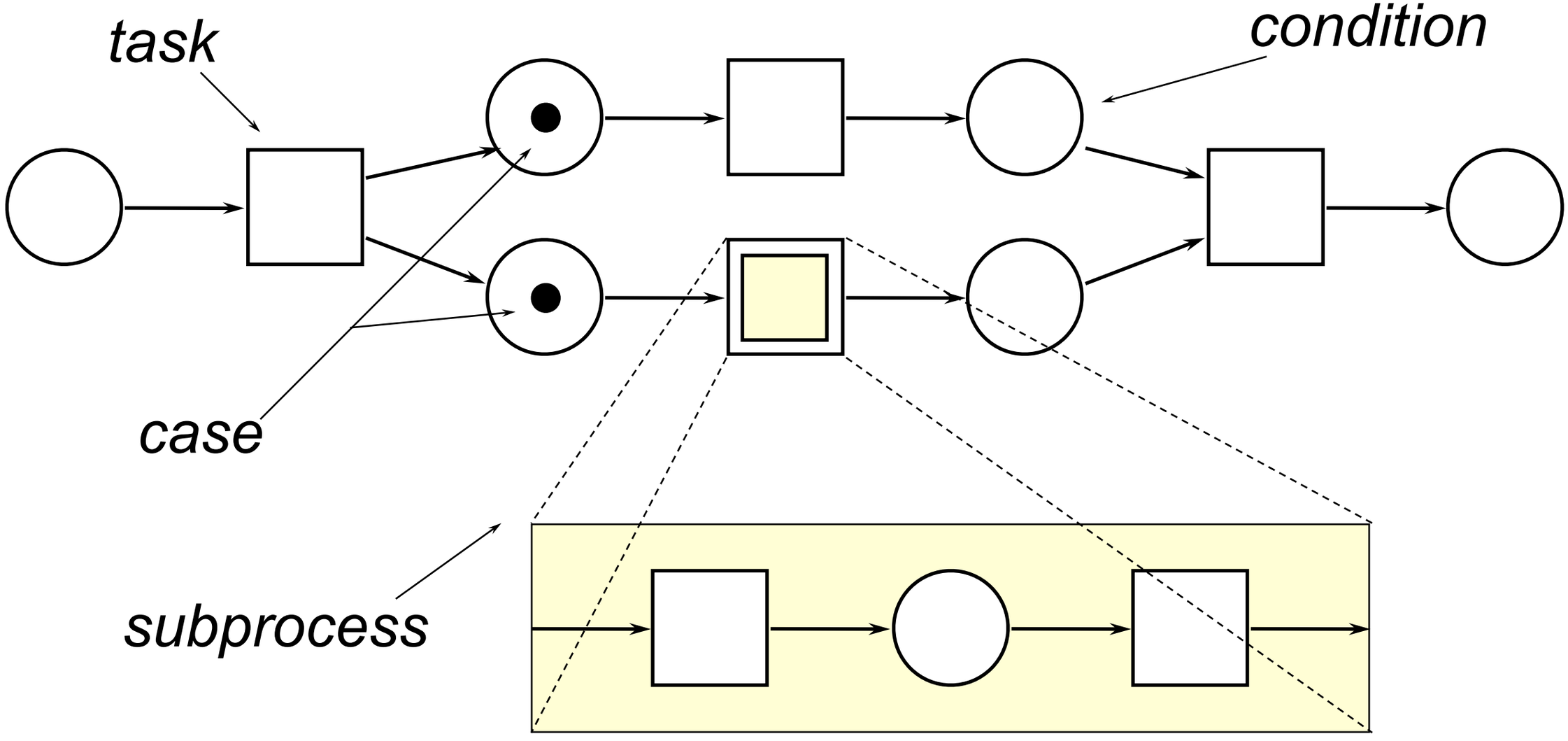
Exercises

- "Desugarize" the workflow nets below, then name all items
- Draw the corresponding Reachability Graphs
- What are their possible firing sequences?





Subprocesses



Triggers

Execution constraints can depend on the environment in which processes are enacted.

In the contexts of workflow nets, transitions can be annotated with the information on who (or what) is responsible for the "firing" of that task.

All transitions that are not annotated can fire automatically

Such annotations are called **triggers**

Triggers

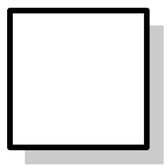
Triggers can be:

the receipt of a message

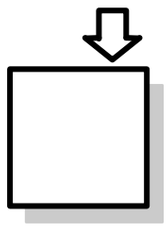
the expiration of a time-out

a human interaction

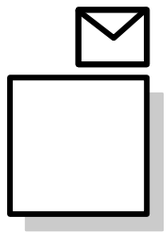
Symbols for triggers



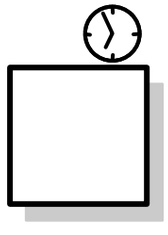
Automatic Trigger: Task enacted automatically



User Trigger: A human user takes initiative and starts activity

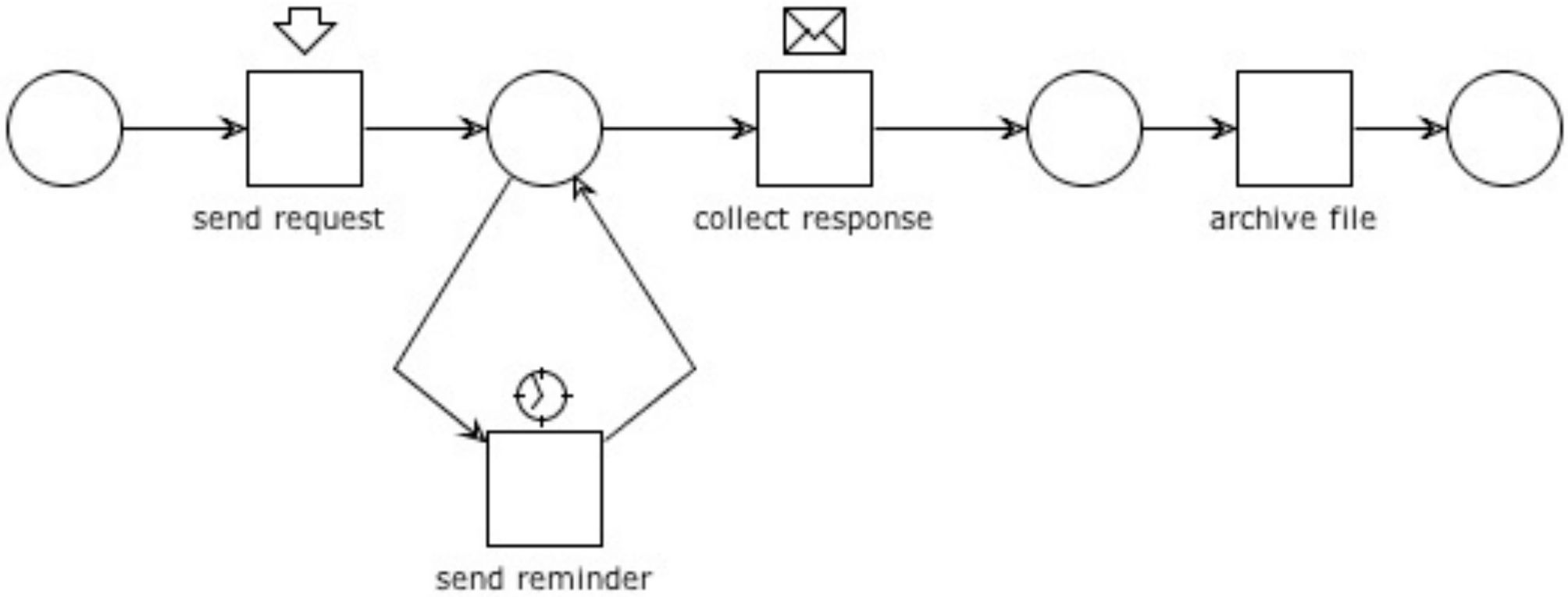


External Trigger: External event required to start activity

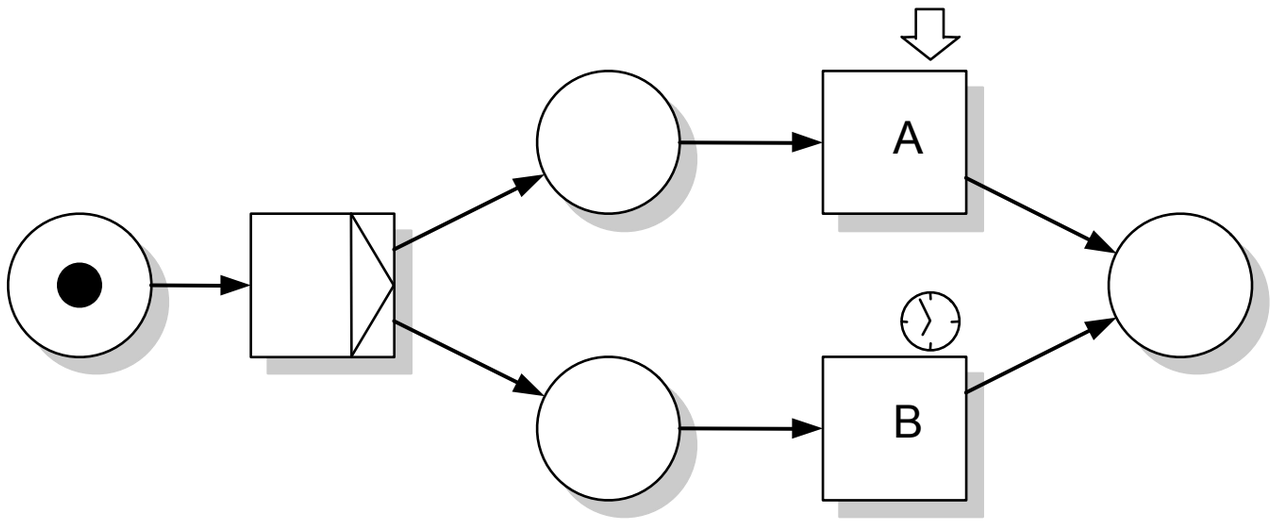


Time Trigger: Activity started when timer elapses

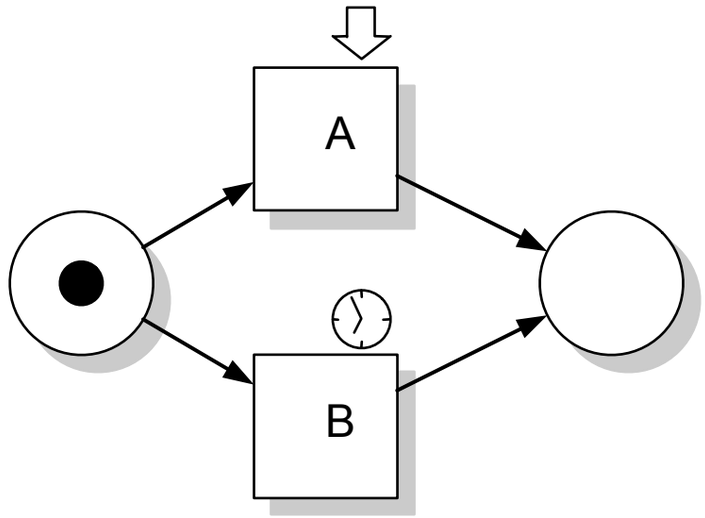
Triggers: example



Explicit vs Implicit XOR-split



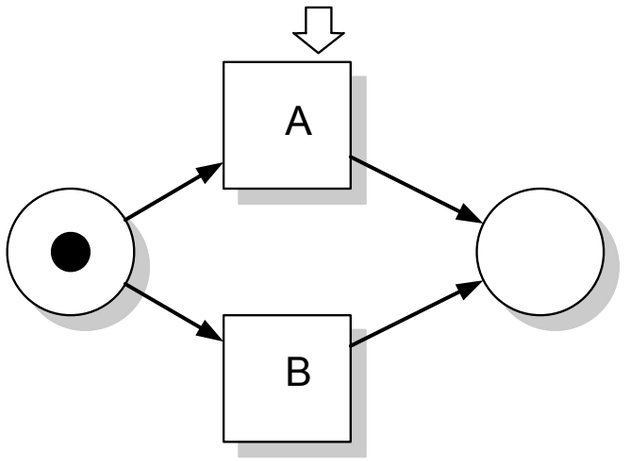
(a) *Explicit xor split* does not enable A and B concurrently



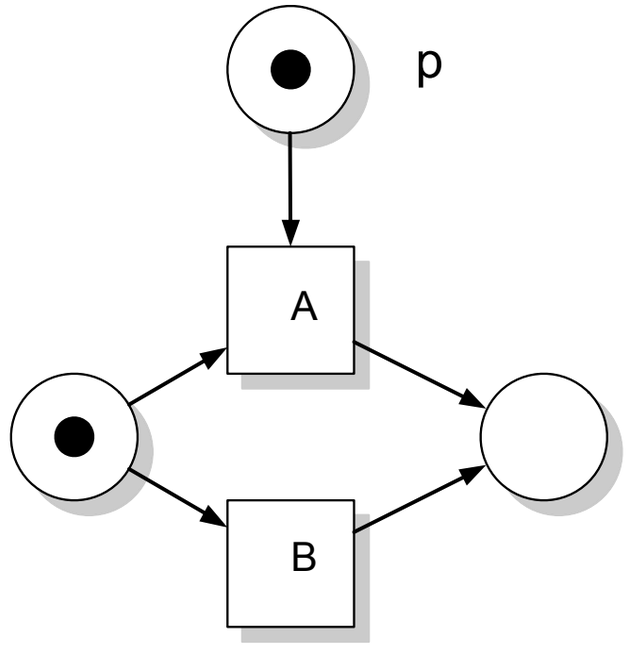
(b) *Implicit xor split* enables A and B concurrently

Encoding triggers

Trigger activities can formally be represented by places with an arc to the respective transition...



(a) Transition A started by user trigger



(b) Representation of user trigger by additional place and additional arc

...but the resulting nets would not be workflow nets!

Terminology: revisited

task: A logical step which may be executed for many cases

work item = task + case

A logical step which may be executed for a specific case

activity = task + case + (trigger) + (resource)
The actual execution of a task for a specific case

(work items and activities are task instances)

Motivation for the analysis

Old BPs generally had simple structures and a physical document linked to each case (a sort of token that serializes tasks)

ICT developments (databases and networks) allowed terrific enhancements... and dangers

information is shared

parallelization is possible

completion times can be shortened

BPs are larger, with increasing complexity

flawed situations are more frequent

Is this WF net ok?