



Formal Approaches to Business Processes through Petri Nets

Nick Russell
Arthur H. M. ter Hofstede



Acknowledgement



These slides summarize the content of Chapter 2 of the book:

A.H.M. ter Hofstede, W. van der Aalst, M. Adams, N. Russell.
Modern Business Process Automation. YAWL and its support environment. Springer, 2010.

These slides have been prepared by or inspired by slides of the following people:

- Wil van der Aalst, TUE & QUT
- Michael Adams, QUT
- Lachlan Aldred, QUT
- Bartek Kiepuszewski, Moreton, BMS, Cutter Consortium
- Marcello La Rosa, QUT
- Petia Wohed, SU/KTH
- Moe Wynn, QUT

- One of the frequent criticisms of modeling notations is that they are *imprecise* and, as a consequence, *subject to varying interpretations by different parties*.
- Describing a candidate modeling notation in terms of a **formal technique** provides an effective means of minimizing the potential for ambiguity.
- To do so, it is necessary to describe both the *syntax* and *semantics* of the modeling formalism using a well-founded technique.
- Suitable techniques for doing so generally stem from mathematical foundations and include general-purpose modeling approaches such as **Petri nets**.

- Petri nets have proven to be a particularly effective mechanism for modeling the **dynamic aspects of processes**.
- Petri Nets they have three specific advantages:
 - Formal semantics despite the graphical nature.
 - State-based instead of event-based.
 - Abundance of analysis techniques.
- Many variant exist: we will analyze ***workflow nets*** and ***reset nets***.
- Petri Nets have been chosen as the formal underpinning for the **YAWL** language (it will be discussed in an upcoming lecture).

- **Formal foundations for modeling languages in BPM**
 - Petri nets
 - Some fundamental results
 - Workflow nets
 - Mapping workflow concepts to Petri nets
 - Reset nets

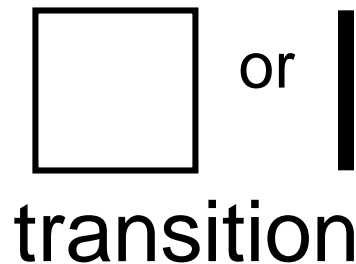
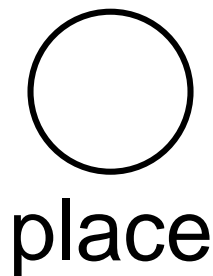
- Originate from C.A. Petri's PhD thesis (1962).
- They were originally conceived as a technique for the description and *analysis of concurrent behaviour in distributed systems*.
- Based on a few simple concepts, yet expressive.
- They have a **simple graphical format** and a **precise operational semantics** that makes them an attractive option for modeling the static and dynamic aspects of processes.
- Many analysis techniques exist.
- Many extensions and variants have been defined over the years.

- Applications in many different areas, such as databases, software engineering, formal semantics, etc.
- There are two main uses of Petri nets for workflows:
 - Specifications of workflows.
 - Formal foundation for workflows (semantics, analysis of properties).

Petri Nets: Basics

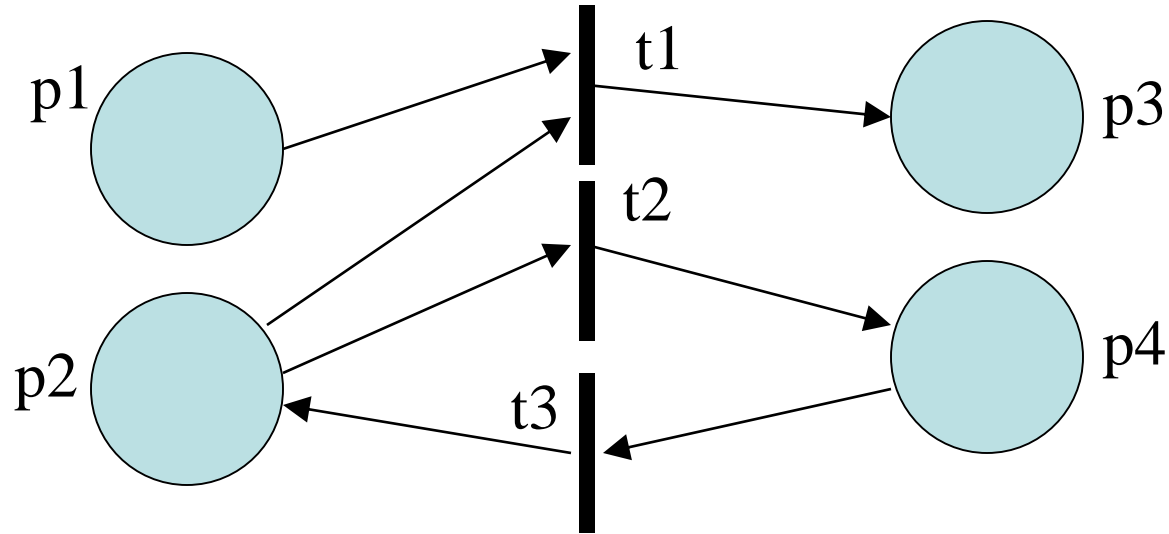


- A Petri Net takes the form of a **directed bipartite graph** where the nodes are either *places* or *transitions*.
- **Places** represent **intermediate states** that may exist during the operation of a process. Places are represented by circles.
- Places can be input/output of **transitions**. Transitions correspond to the **activities** or **events** of which the process is made up. Transitions are represented by rectangles or thick bars.
- **Arcs** connect places and transitions in a way that places can only be connected to transitions and vice-versa.



- Formally a Petri net N is a triple (P, T, F) where
 - P is a finite set of places
 - T is a finite set of transitions where $P \cap T = \emptyset$
 - $F \subseteq (P \times T \cup T \times P)$ is the set of arcs known as the **flow relation**
- A directed arc from a place p to a transition t indicates that p is an **input place** of t . Formally:
 - $\bullet t = \{p \in P \mid (p, t) \in F\}$
- A directed arc from a transition t to a place p indicates that p is an **output place** of t . Formally:
 - $t \bullet = \{p \in P \mid (t, p) \in F\}$
 - With an analogous meaning, we can define:
 - $p \bullet = \{t \in T \mid (p, t) \in F\}$ and $\bullet p = \{t \in T \mid (t, p) \in F\}$

Petri Net: Example



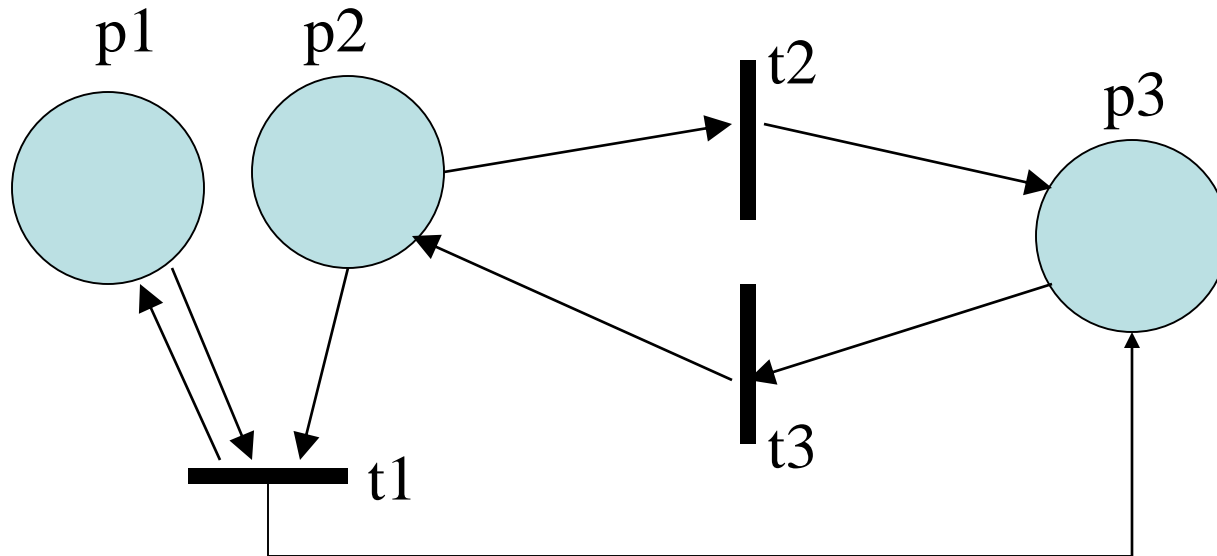
$P = \{p_1, p_2, p_3, p_4\}$

$T = \{t_1, t_2, t_3\}$

$F = \{(p_1, t_1), (p_2, t_1), (t_1, p_3), (p_2, t_2), (t_2, p_4), (p_4, t_3), (t_3, p_2)\}$

$t_1 \bullet = \{p_3\}; \bullet t_1 = \{p_1, p_2\}; \bullet p_2 = \{t_3\}; \bullet p_1 = \emptyset; p_2 \bullet = \{t_1, t_2\}$

Petri Nets: Example



$P = \dots$

$T = \dots$

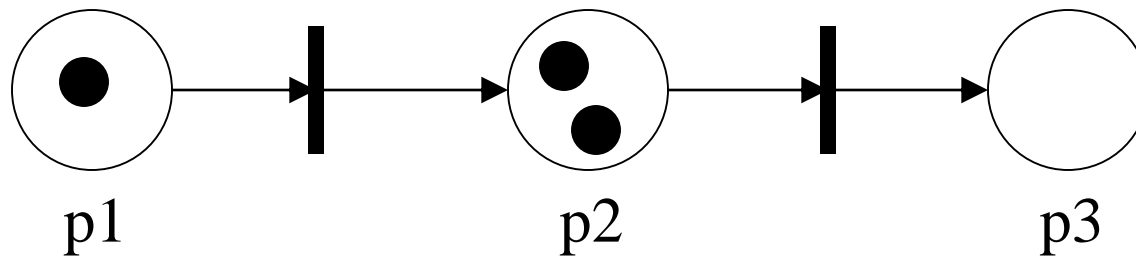
$F = \dots$

$t_1 \bullet = \dots ; \bullet t_1 = \dots ; \bullet p_2 = \dots ; p_2 \bullet = \dots$

Markings



- The operational semantics of a Petri Net is described in terms of particular marks called **tokens** (graphically represented as black dots ●).
- Places in Petri Nets can contain any number of tokens. The distribution of tokens across all of the places in a net is called a **marking**. For a Petri net an *initial marking* M_0 needs to be specified.
- Marking assigns tokens to places; formally, a marking M of a Petri net $N = (P, T, F)$ is a function **$M: P \rightarrow \mathbf{NAT}$** .

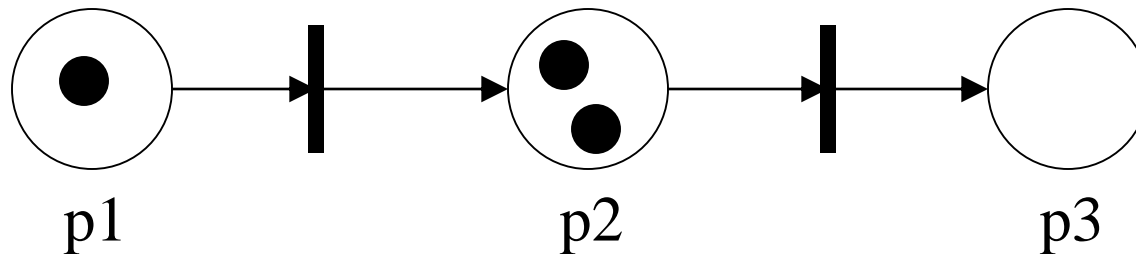


- The marking below is formally captured by the following marking $M = \{(p_1, 1), (p_2, 2), (p_3, 0)\}$.

State of a Petri Net



- A state can be compactly described as shown in the following example:
 - $1p_1 + 2p_2 + 0p_3$ is the state with one token in place p_1 , two tokens in p_2 and no tokens in p_3 .
 - We can also represent this state in the following (equivalent) way: $p_1 + 2p_2$.



- We can also describe an ordering function \geq over the set of possible states such that, given a Petri net $N = (P, T, F)$ and markings M and M' , $M \geq M'$ iff for all p in P : $M(p) \geq M'(p)$. $M > M'$ iff $M \geq M'$ and $M \neq M'$.

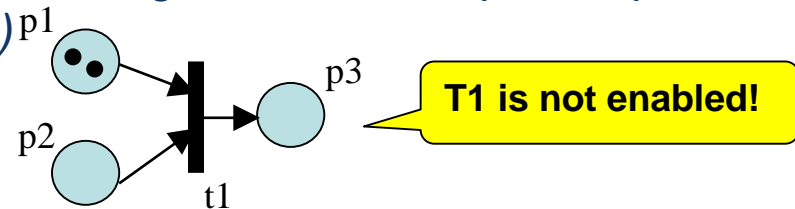
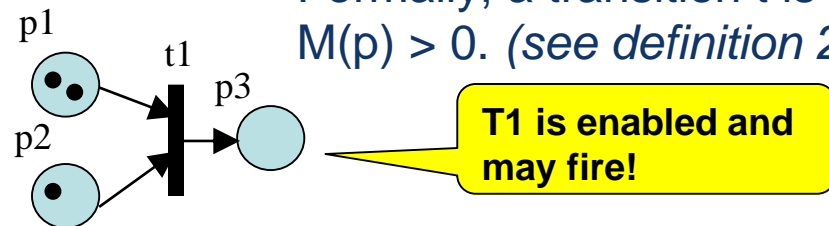
Enabled Transitions



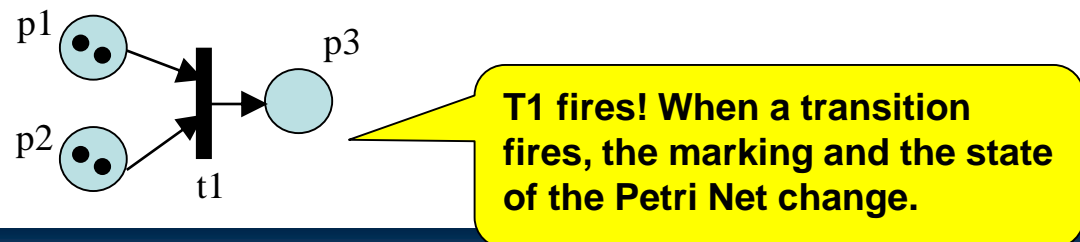
- The operational semantics of Petri nets are characterized by the notion of a transition executing or “**firing**”. A transition in a Petri net can “fire” whenever there are one or more tokens in each of its input places.
- The execution of a transition occurs in accordance with the following firing rules:

1. A transition t is said to be **enabled** if and only if each input place p of t contains at least one token. Only enabled transitions may fire.

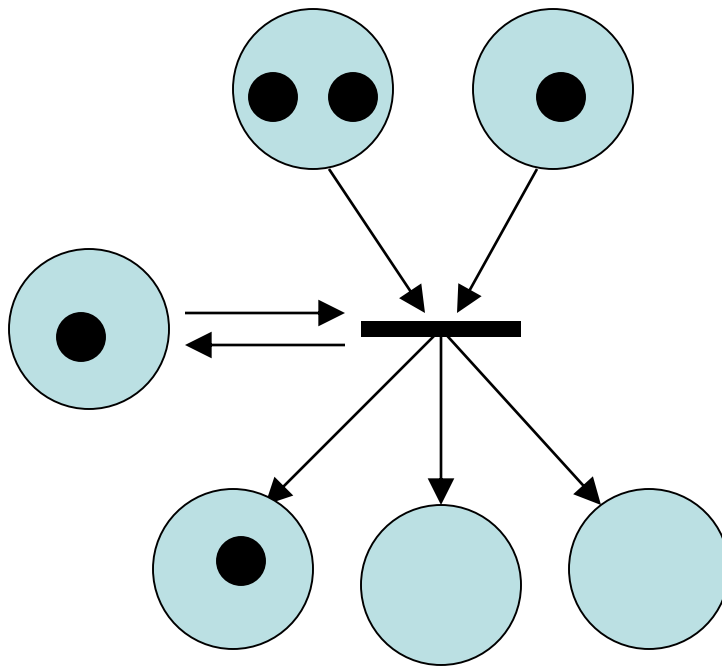
– Formally, a transition t is enabled in a marking M iff for each p , with $p \in \bullet t$, $M(p) > 0$. (see definition 2.7 of [DE95])



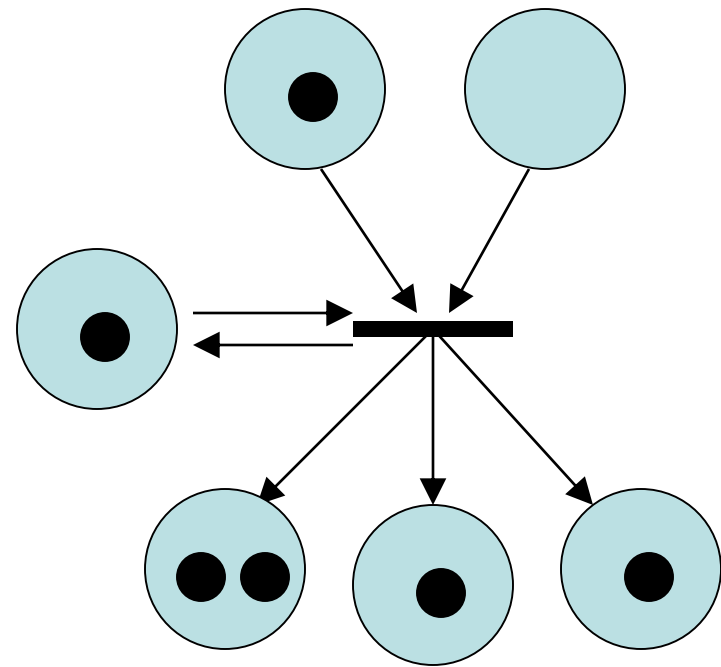
2. If transition t fires, then t **consumes one token** from each input place p of t and **produces one token** for each output place p of t .



Firing a Transition: Example

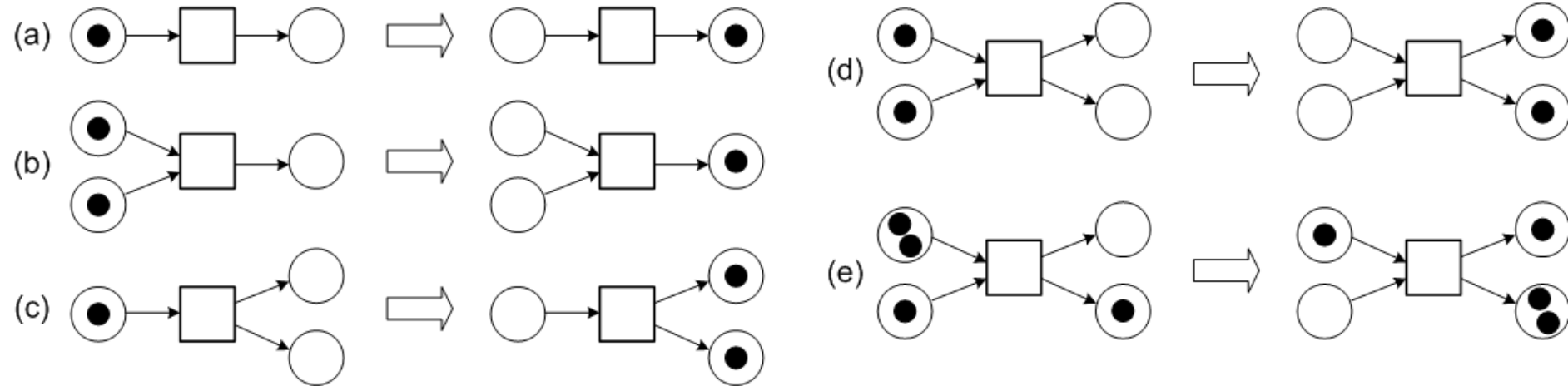


BEFORE



AFTER

Firing Transitions: Further Examples



- It is assumed that the firing of a transition is an atomic action that occurs instantaneously and cannot be interrupted.
- If there are multiple enabled transitions, any one of them may fire; however, for execution purposes, it is assumed that **they cannot fire simultaneously**.
- An enabled transition is not forced to fire immediately but can do so at a time of its choosing.
- These features make Petri nets particularly suitable for modeling concurrent process executions.

Firing Transitions



- Given a Petri Net (P, T, F) and an initial state \mathbf{M} , we have the following notations that characterize the firing of a given transition t :
 - $\mathbf{M} \rightarrow^t \mathbf{M}'$ indicates that if transition t is enabled in state \mathbf{M} , then firing t in \mathbf{M} results in state \mathbf{M}' . Formally, notation $\mathbf{M} \rightarrow^t \mathbf{M}'$, is defined by:
 - $M'(p) = M(p)$ if $p \notin \bullet t \cup t \bullet$ or $p \in \bullet t \cap t \bullet$
 - $M'(p) = M(p) - 1$ if $p \in \bullet t$ and $p \notin t \bullet$
 - $M'(p) = M(p) + 1$ if $p \in t \bullet$ and $p \notin \bullet t$
 - $\mathbf{M} \rightarrow \mathbf{M}'$ indicates that there is a transition t such that $\mathbf{M} \rightarrow^t \mathbf{M}'$.
 - $\mathbf{M} \rightarrow^\sigma \mathbf{M}'$ denotes the firing sequence $\sigma = t_0 t_1 \dots t_{n-1}$ that leads from state \mathbf{M} to state \mathbf{M}' , such that

$$\mathbf{M} = \mathbf{M}_0 \rightarrow^{t_0} \mathbf{M}_1 \rightarrow^{t_1} \mathbf{M}_2 \dots \mathbf{M}_{n-1} \rightarrow^{t_{n-1}} \mathbf{M}_n = \mathbf{M}'$$

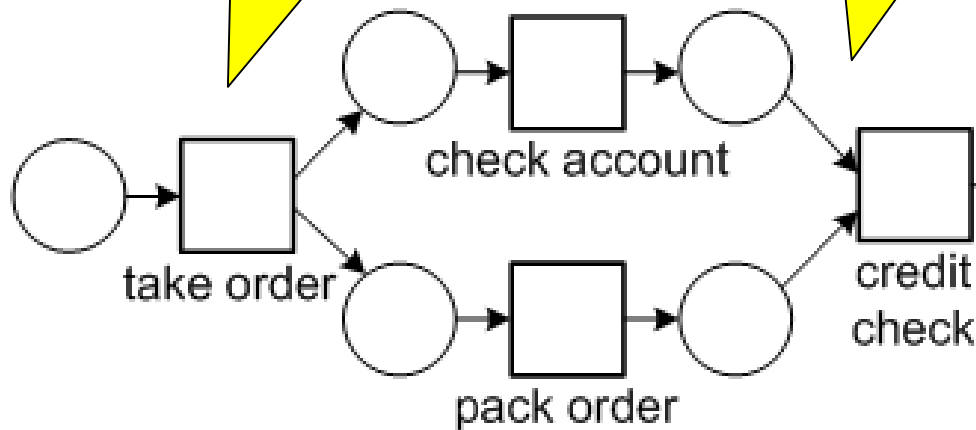
Note that the transitions do not have to be different!

- A state \mathbf{M}' is called **reachable** from state \mathbf{M} (we write $\mathbf{M} \rightarrow^* \mathbf{M}'$) iff there is a firing sequence σ that leads from state \mathbf{M} to state \mathbf{M}' .
 - Informally, a marking is reachable from another marking if there is a sequence of transitions that can fire from the first marking to arrive at the second marking.

Petri nets: Order Fulfillment Example



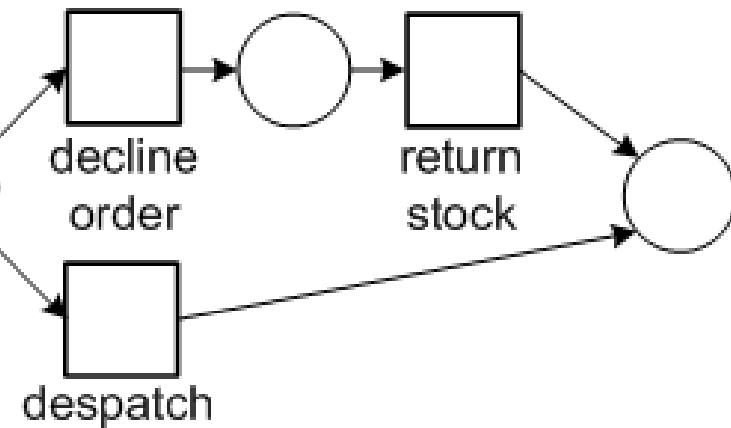
First, a **take order** task is executed.



When **pack order** and **check account** tasks have been both completed, the **credit check** task is executed.

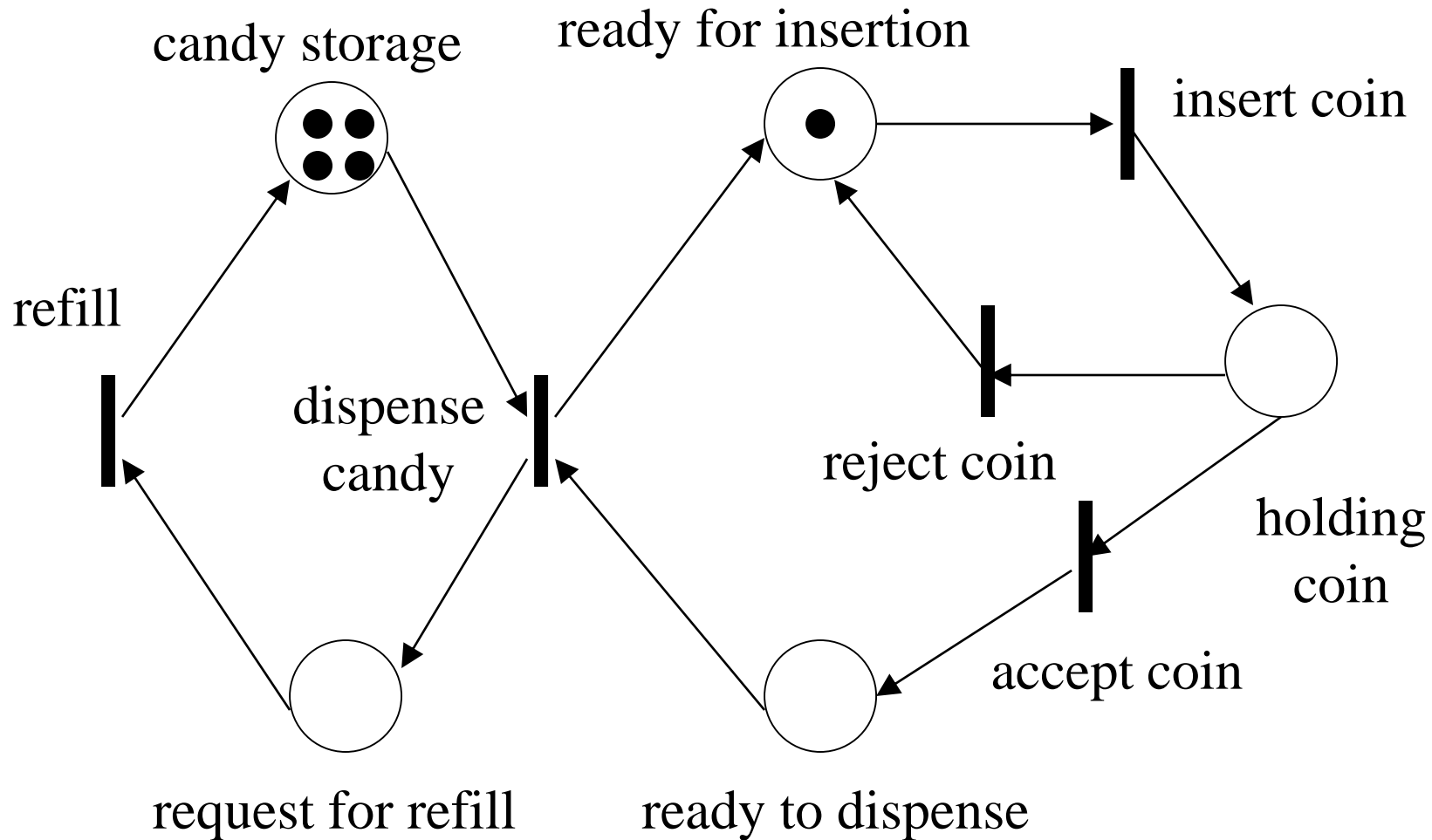
Then, **pack order** and **check account** tasks are executed in parallel.

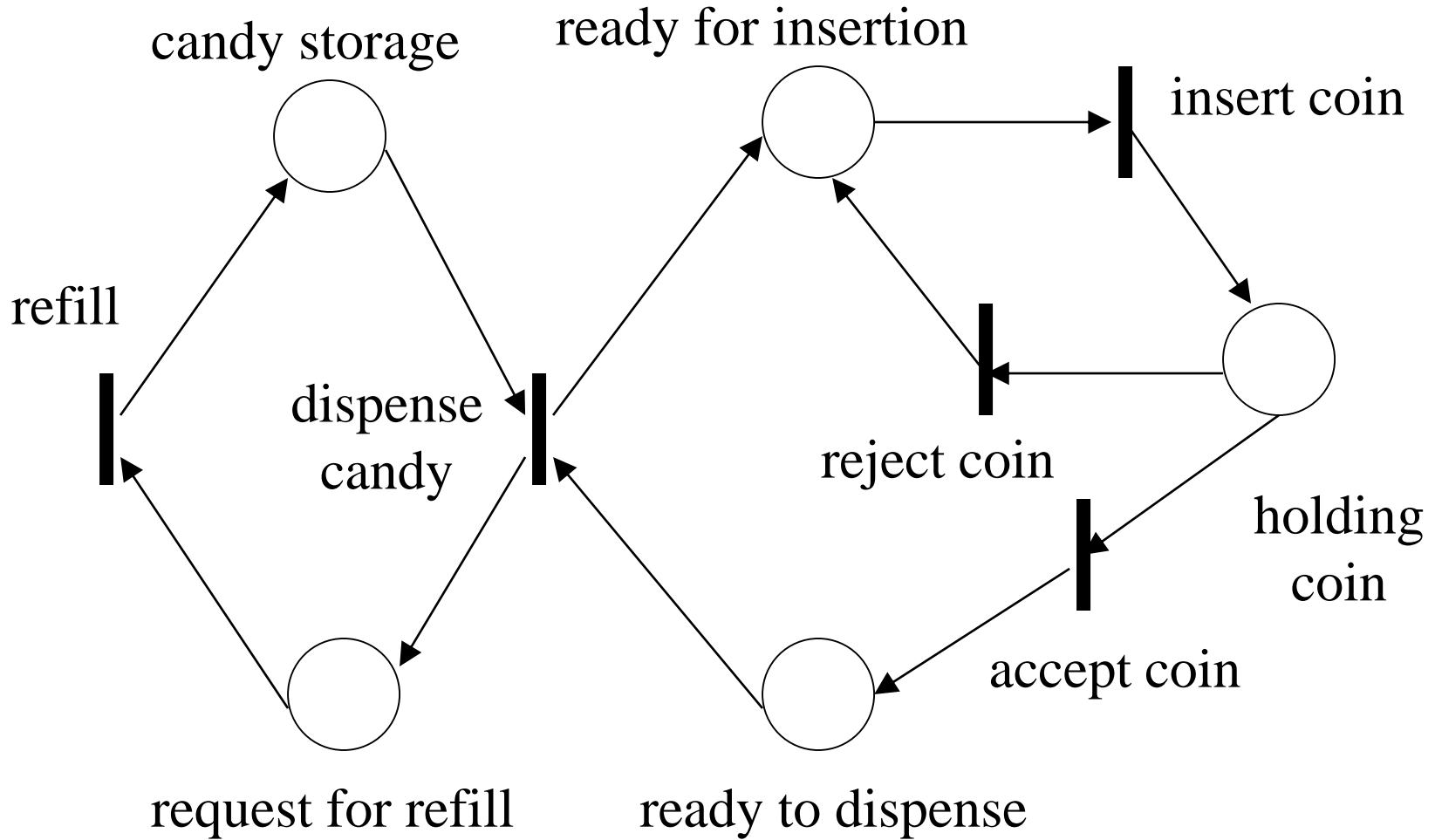
If the customer has not sufficient credit the **decline order** runs and, finally, the **return stock** task ensures that the items from the order are returned to the warehouse.



If the customer has sufficient credit remaining, the order is **despatched**.

Petri nets: Example of a vending machine (source [DE95] p. 4)






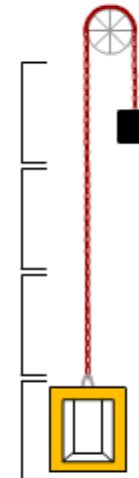
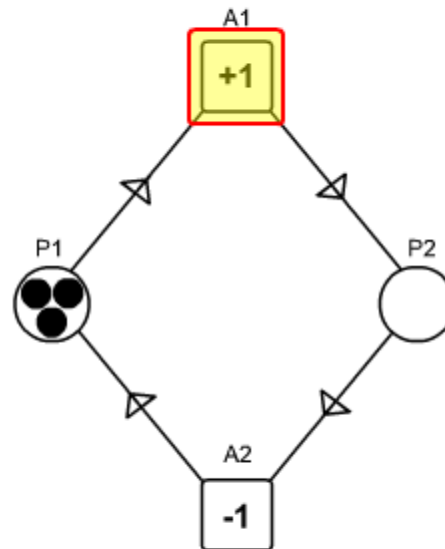
Petri net example: Elevator 1



Elevator (1)

TU/e technische universiteit eindhoven

Press: 
To fire the activity



The number of tokens represents the number of vertical movements an elevator can make upwards or downwards in a certain state of the system.

/ faculty of technology management

© Wil van der Aalst, Vincent Almering en Herman Wijbenga

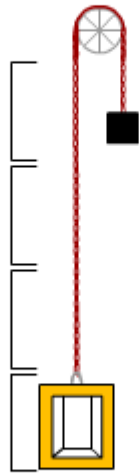
Animation by Wil van der Aalst, Vincent Almering and Herman Wijbenga

Petri net example: Elevator 2

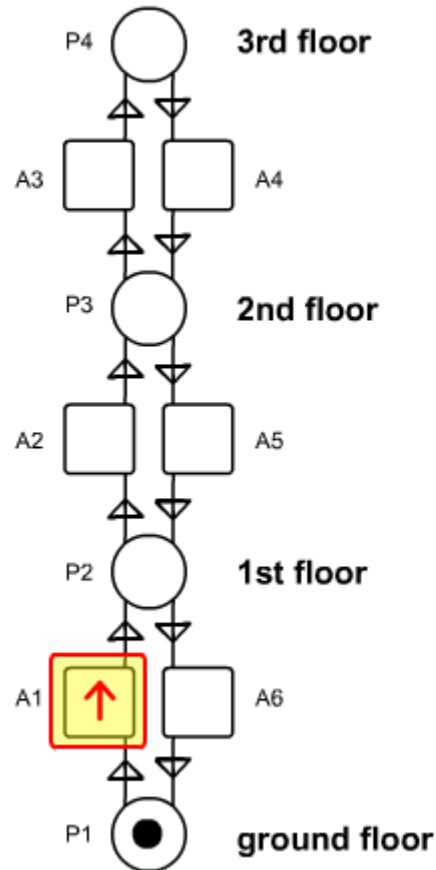



Elevator (2)

The token represents the elevator.



TU/e technische universiteit eindhoven



Press: 
To fire the activity

/ faculty of technology management

© Wil van der Aalst, Vincent Almering en Herman Wijbenga

Animation by Wil van der Aalst, Vincent Almering and Herman Wijbenga


Petri net example: Elevator 3

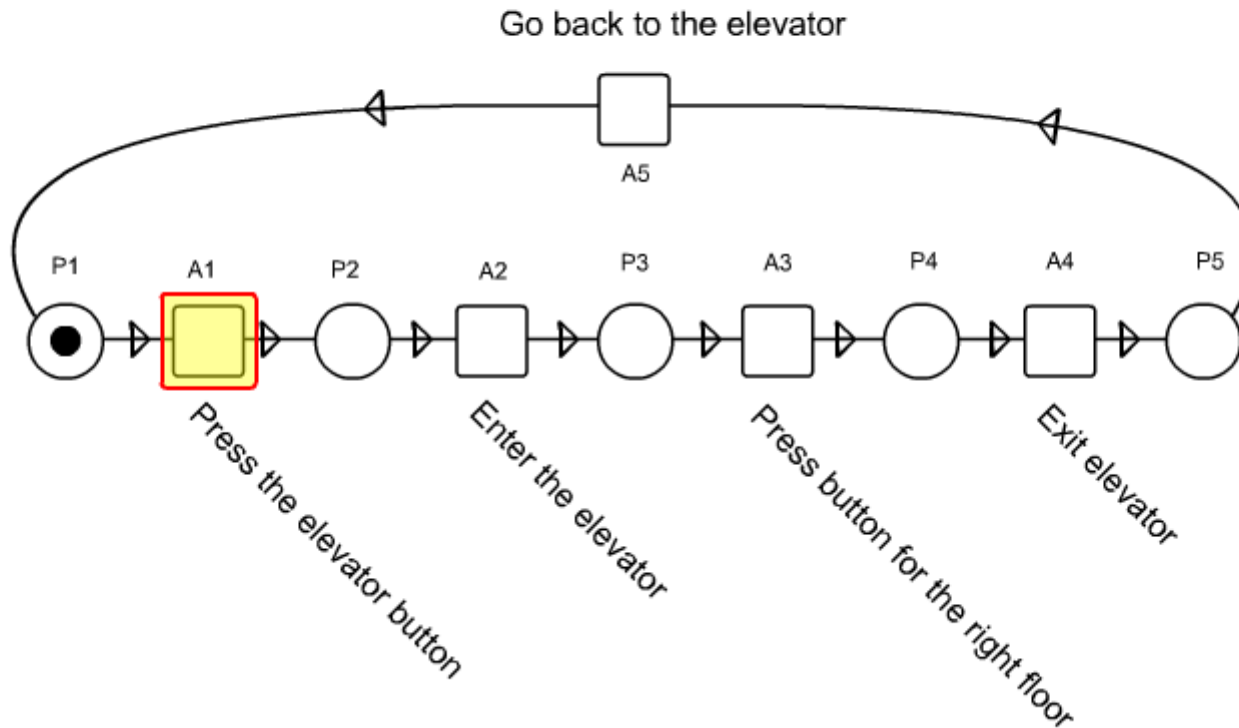


Elevator (3)

TU/e technische universiteit eindhoven

The token is a person using the elevator.

Press: 
To fire the activity



/ faculty of technology management

© Wil van der Aalst, Vincent Almering en Herman Wijbenga

Animation by Wil van der Aalst, Vincent Almering and Herman Wijbenga

Modelling Exercise



- We want to model with a Petri Net the behaviour of two traffic lights at an intersection, in a way that they cannot be green or yellow at the same time.
- Conversely, they are allowed to signal red at the same time.


Solution Traffic Lights

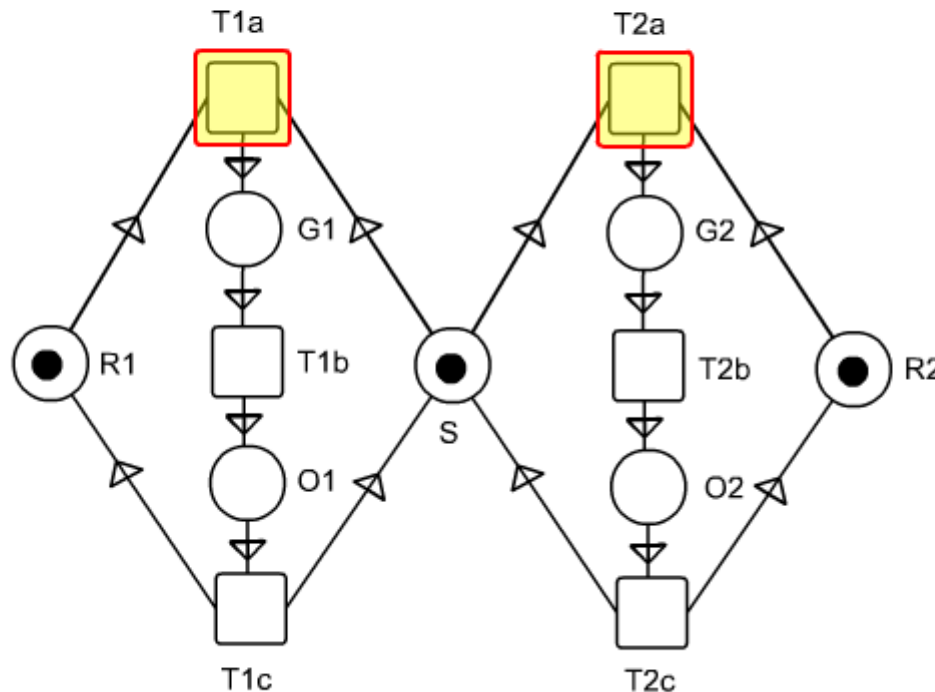


Two traffic lights

TU/e technische universiteit eindhoven

There are two traffic lights, which are not allowed to signal green at the same time.

Press:  to fire the activity



/ faculteit technologie management

© Wil van der Aalst, Vincent Almering en Herman Wijbenga

Animation by Wil van der Aalst, Vincent Almering and Herman Wijbenga

Homeworks

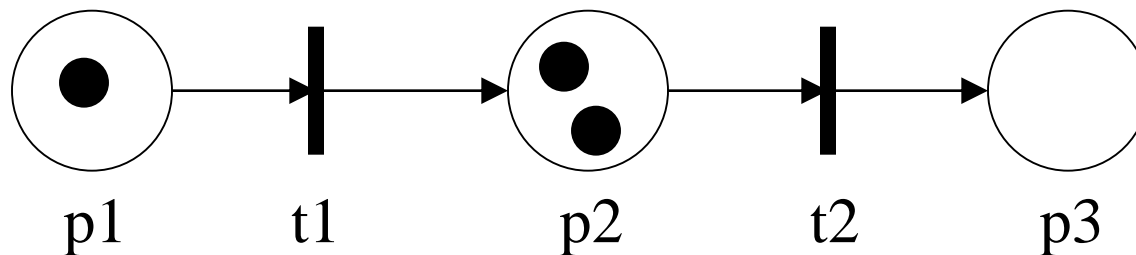


- Two traffic lights at an intersection. If one is red, the other should be green etc. (*many discussions on modelling traffic lights through Petri nets can be found on the internet*).
- A producer and a consumer producing and consuming (resp.) indefinitely. The consumer cannot consume more than the producer has produced thus far. How does your model change if the buffer between them is of limited size? (*this is a well-known concurrency problem*)
- Two parallel processes with two critical sections. If one of the two processes is in its critical section, the other process should not be able to enter its critical section and vice versa. (*this is also a well-known concurrency problem*)

Coverable Markings



- Coverability is a weaker notion than reachability. A marking M is **coverable** iff a reachable marking M' exists such that $M' \geq M$ (see e.g. Definition 5 in [HAAR09]).
- *Example:* Given the Petri net and marking in figure, $p1+p2+p3$ is a reachable marking, while $p1+p3$ is a coverable marking (but not reachable).



- To decide whether a given marking M is reachable is a $DSPACE(\exp)$ -hard problem.

Properties



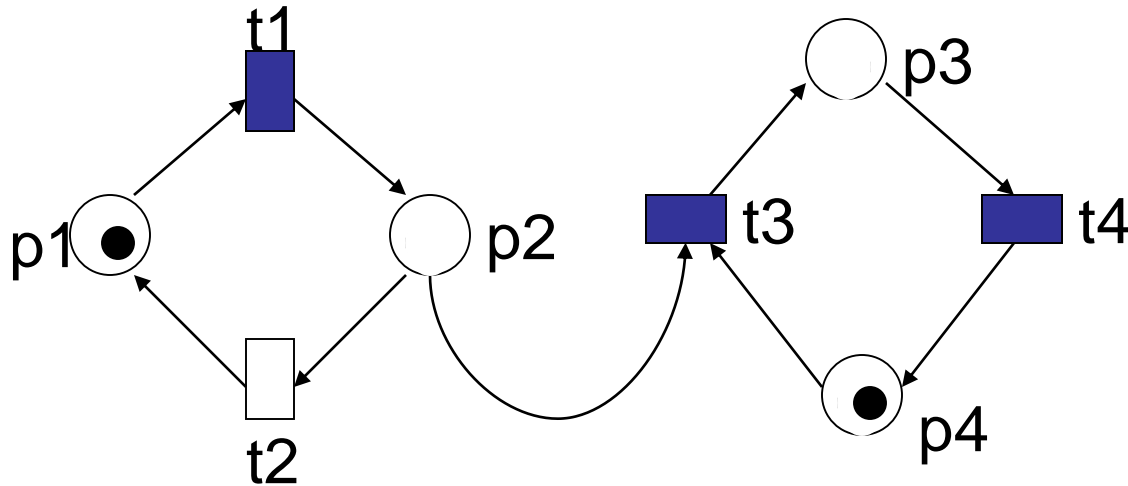
- A Petri net N with initial marking M_0 is **live** iff for every reachable marking M and every transition t there exists a marking M' reachable from M which enables t . (see definition 2.16 of [DE95]).
- More informally: A Petri net with initial marking M_0 is live if, no matter what marking has been reached from M_0 , it is possible to ultimately fire any transition by progressing through some further firing sequence.
- The notion of *liveness* is important since it demonstrates that at least one transition can fire in every reachable state. A live Petri net guarantees deadlock-free operation.
- A Petri net N with initial marking M_0 is **deadlock free** iff every reachable marking enables some transition (see definition 2.16 of [DE95]).

Properties



- A Petri net N with initial marking M_0 is ***k-bounded*** iff for every reachable marking M , $M(p) \leq k$ (k is the minimal number for which this holds). (see *definition 2.20 of [DE95]*).
 - A 1-bounded net is called ***safe***.
 - The property of *boundness* ensures that the number of tokens cannot grow arbitrarily.
- A Petri net N is ***strongly connected*** iff for every pair of nodes (places or transitions) x and y there is a path from x to y and vice-versa.

Is this Petri Net live and bounded?



$M_0 = (1, 0, 0, 1)$

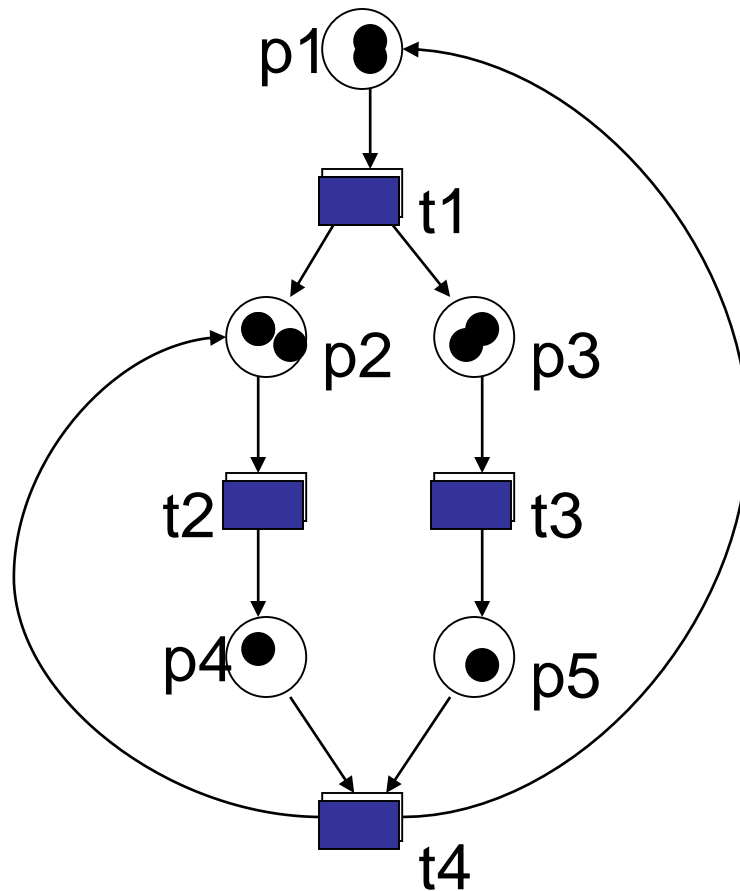
$M_1 = (0, 1, 0, 1)$

$M_2 = (0, 0, 1, 0)$

$M_3 = (0, 0, 0, 1)$

A bounded but non-live Petri net

Is this Petri Net live and bounded?



$$M0 = (1, 0, 0, 0, 0)$$

$$M1 = (0, 1, 1, 0, 0)$$

$$M2 = (0, 0, 0, 1, 1)$$

$$M3 = (1, 1, 0, 0, 0)$$

$$M4 = (0, 2, 1, 0, 0)$$

⋮

An unbounded but live Petri net

Exercise



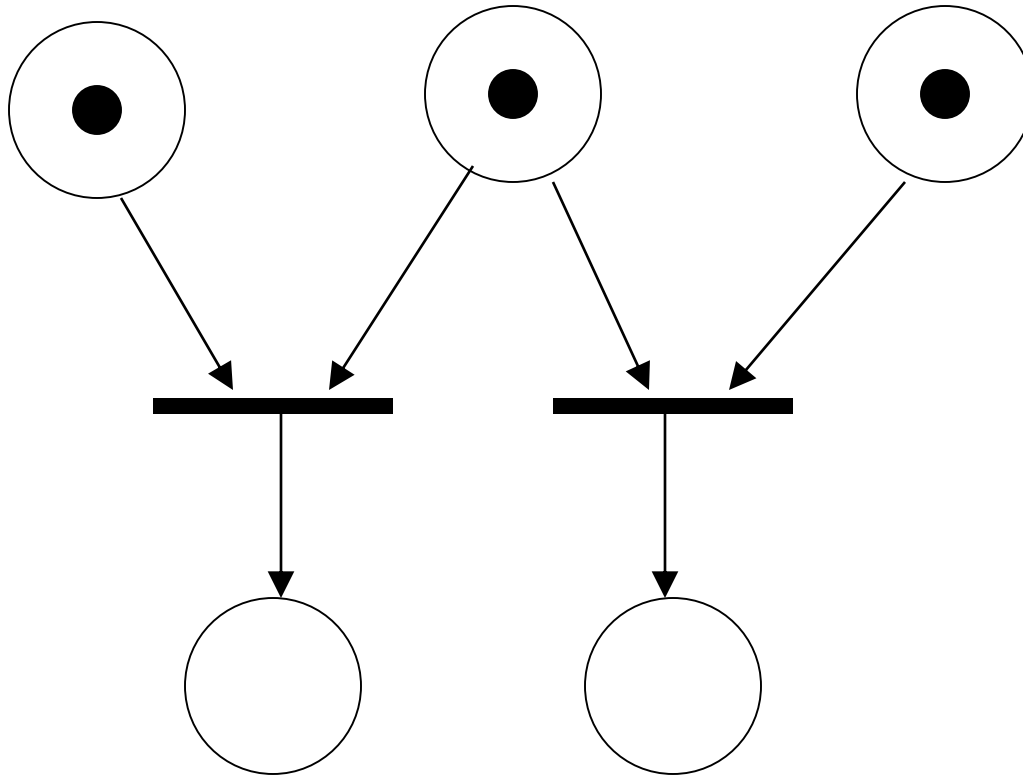
- Is the vending machine live?
- Is it deadlock free?
- Is it bounded?
- Is it strongly connected?
- Can a marking be reached with tokens both in “ready for insertion” and “ready to dispense”?
- Give an example of a marking that is coverable but not reachable.

Free Choice Petri nets



- Many verification problems in Petri nets have a high complexity.
- **Free Choice Petri nets** are a subclass of Petri nets with a “nice” tradeoff between expressiveness and analyzability (see e.g. [DE95]).
- All elementary workflow concepts are essentially free choice.
- In a Free Choice Petri net “*the result of the choice between two transitions can never be influenced by the rest of the system*” [DE95]

Example of a Conflict

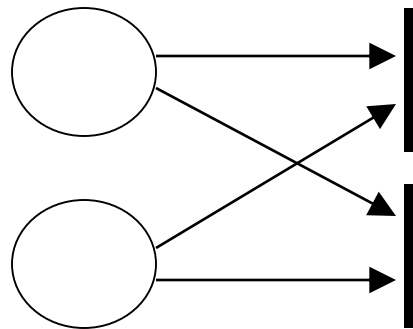


Free Choice Petri nets: Definition

(see [DE95] p63-64)



- In a Free Choice Petri net, every pair of transitions either share all their input places, or they share none.
- Formally, a Petri net $N = (P, T, F)$ is free choice iff for all transitions t, t' :
 - $\bullet t \cap \bullet t' \neq \emptyset \Rightarrow \bullet t = \bullet t'$
 - For any free-choice net, a t' in conflict with an enabled transition t , is also enabled.



Workflow nets: Motivation



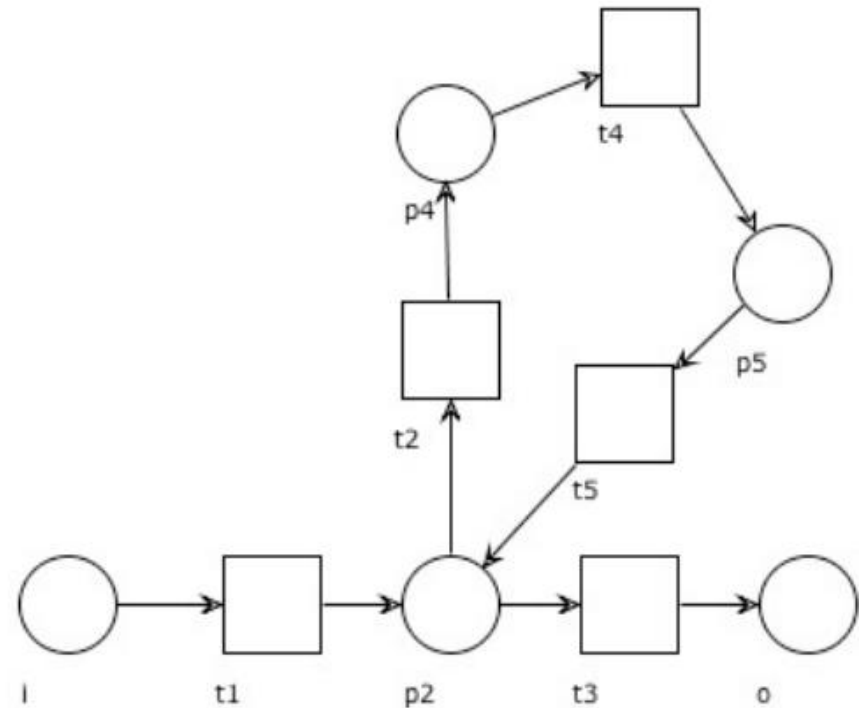
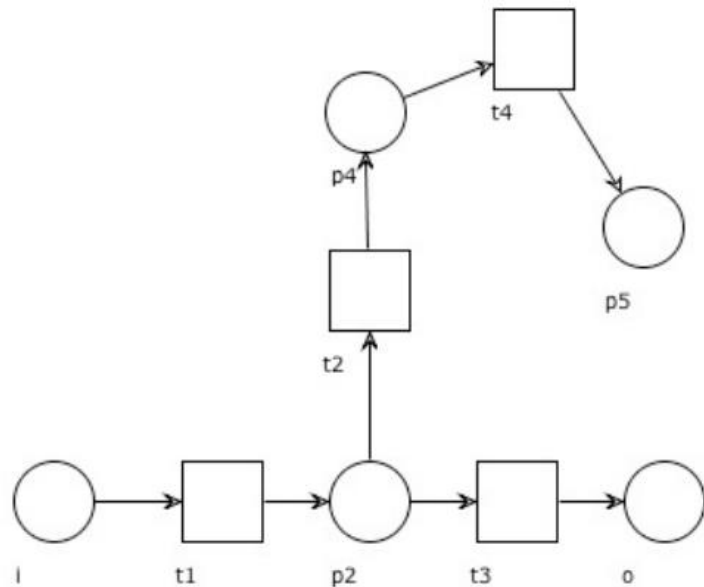
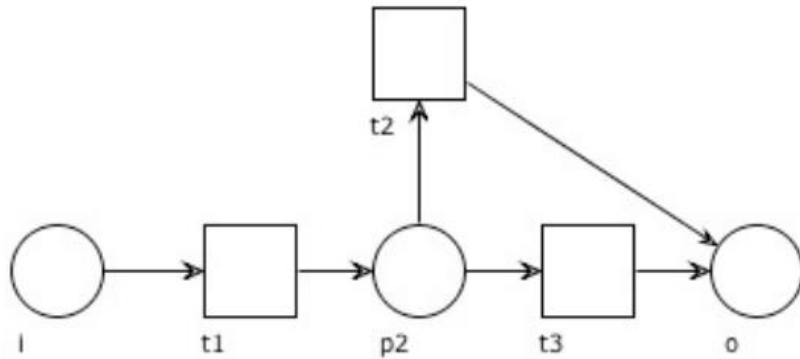
- Wil van der Aalst has proposed the use of Petri nets for **workflow modelling**. In [Aalst96] three benefits are argued:
 - Petri nets are formally defined;
 - Petri nets support the notion of being “in between” performing tasks through the notion of place;
 - Petri nets have associated analysis techniques.
- He proposes a particular subclass of Petri nets, called **Workflow nets** (WF-nets) for this purpose.
- In a workflow net, *transitions* represent the **tasks** that comprise a business process and *places* represent the **conditions** preceding and following the tasks.

Workflow nets: Definition



- A workflow net has a single **start place** and a single **end place**.
 - This means that workflow nets closely correspond to real-life processes that tend to have a specific starting point and a specific end point.
- Every transition in the workflow net is on a **path from the start to the end place**.
 - This ensures that each transition in a workflow net contributes to the progression of an executing instance towards its end state.
- **Definition [AH02, p271-272]** A Petri net $PN = (P, T, F)$ is a WF-net (Workflow net) if and only if:
 - There is one source place $i \in P$ such that $\bullet i = \emptyset$
 - There is one sink place $o \in P$ such that $o \bullet = \emptyset$
 - Every node $x \in P \cup T$ is on a path from i to o .

Exercise: Candidate WF-nets?



Workflow nets: Definition



- It is important to note that the previous definition traces the minimal requirements for a workflow net.
- However, it does not guarantee (by itself) that a candidate workflow net will not potentially be subject to *deadlock* or *livelock*.
- To ensure that any given process instance behaves in a predictable way, in [AH02] a number of so-called **soundness criteria** are formulated.

Workflow nets: Soundness



Definition [soundness]: A procedure modeled in the form of a WF-net $PN = (P, T, F)$ is **sound** if and only if:

- **[Option to Complete]** Given an initial marking i , from every marking M reachable from i , there exists a firing sequence leading from state M to state o .

Formally:
$$\forall M \left[(i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o) \right]$$

- *Basically, this means that the any executing instance of the workflow net must eventually terminate, i.e., net is free of deadlock and infinite loops.*
- **[Proper Completion]** State o is the only state reachable from state i with at least one token in place o . Formally:

$$\forall M \left[(i \xrightarrow{*} M \wedge M \geq o) \Rightarrow (M = o) \right]$$

- *When the workflow terminates no other tasks are still running and termination is signalled only once. At the moment of termination, there must be **one token** in the end place o and all other places in the WF-net **must be empty**.*
- **[No Dead Tasks]** For every transition t , a marking M reachable from i ($i \xrightarrow{*} M$) can be found that enables t .
 - *The workflow does not contain any superfluous parts that can never be activated. In a nutshell, dead transitions are not allowed.*

Workflow Net Constructs



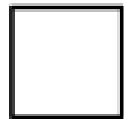
Automatic tasks execute as soon as they are enabled.

User tasks are passed to human resources for execution once enabled.

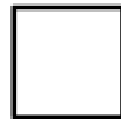
In WF-net there are some notational enhancements (often termed “syntactic sugar”) for **split** and **join** constructs that simplify the specification of a workflow net.



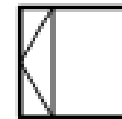
place



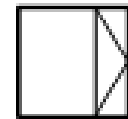
automatic task



user task



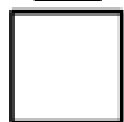
explicit OR join



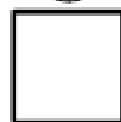
explicit OR split



arc



external task



time task



AND split



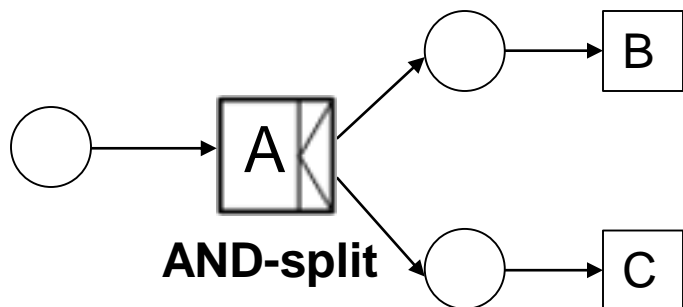
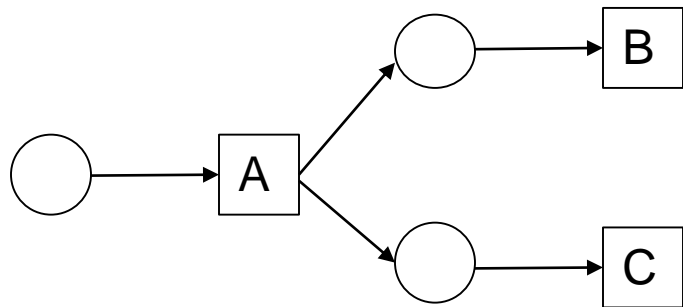
AND join

External tasks only proceed once they are enabled and a required message or signal is received from the operating environment.

Time tasks only proceed once they are enabled and a specified (time-based) deadline occurs.

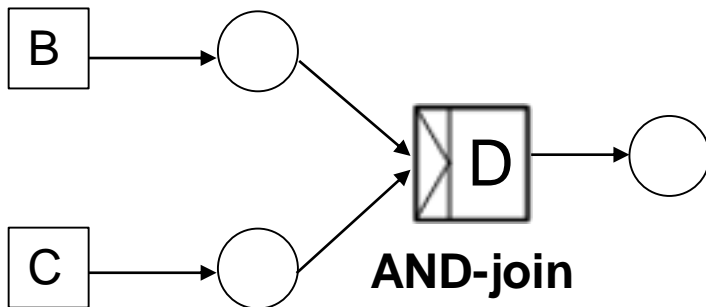
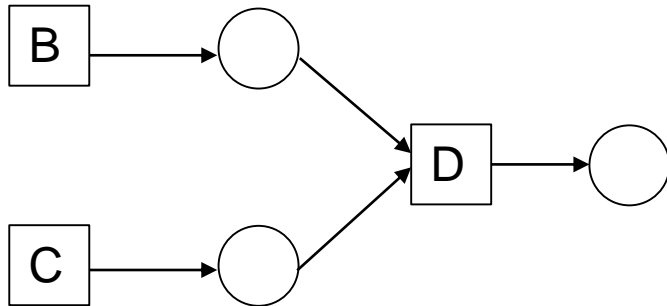
The basics of Petri nets can be used to understand the semantics of some elementary modeling concepts in WF-nets.

Parallelism: AND-split



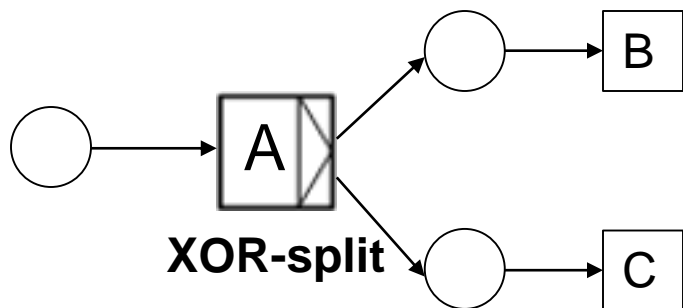
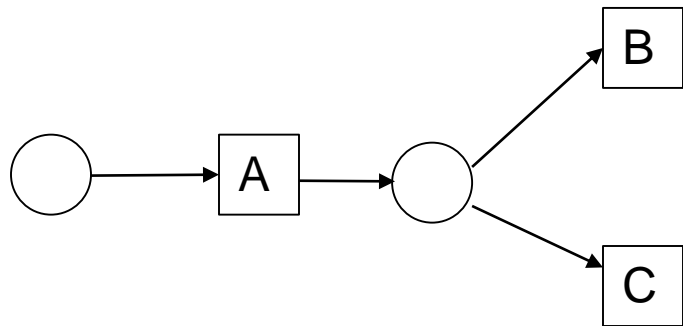
- According to the WfMC [WfMC], an **AND-split** is “a point within the workflow where a single thread of control splits into two or more threads which are executed in parallel within the workflow, allowing multiple activities to be executed simultaneously.”
- The execution of A enables both task B and task C. As a result, task B and task C are executed **in parallel** (in an arbitrary order).
- In WF-nets, a special construct for AND-split is introduced.

Parallelism: AND-join



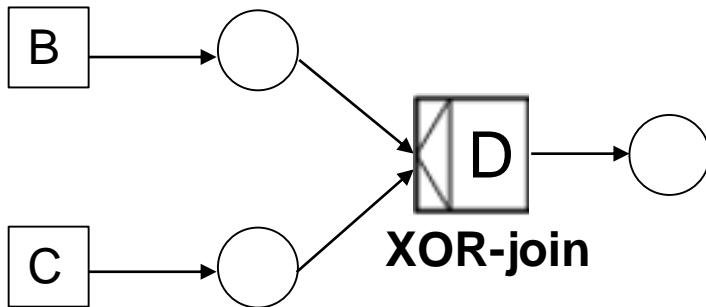
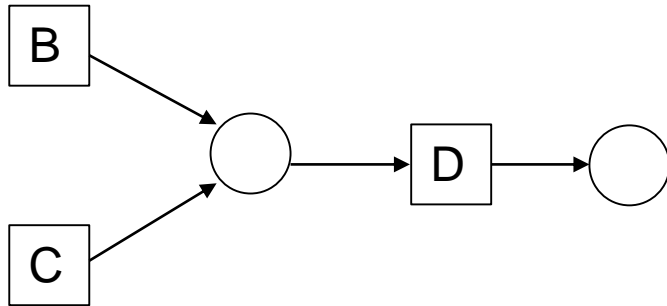
- According to the WfMC [WfMC], an **AND-join** is “a point in the workflow where two or more parallel executing activities converge into a single common thread of control.”
- Task D is enabled after execution both *B* and *C*, i.e., D is used to synchronize two subflows.
- In WF-nets, a special construct for AND-JOIN is introduced.

Conditional Routing: XOR-split



- According to the WfMC [WfMC], a **XOR-split** is “a point within the workflow where a single thread of control makes a decision upon which branch to take when encountered with multiple alternative workflow branches.”
- Note that the exclusive nature of the choice, i.e. only one of the outgoing branches can be chosen (i.e., either task B or C can be executed).
- In WF-nets, a special construct for XOR-split is introduced.

Conditional Routing: XOR-join



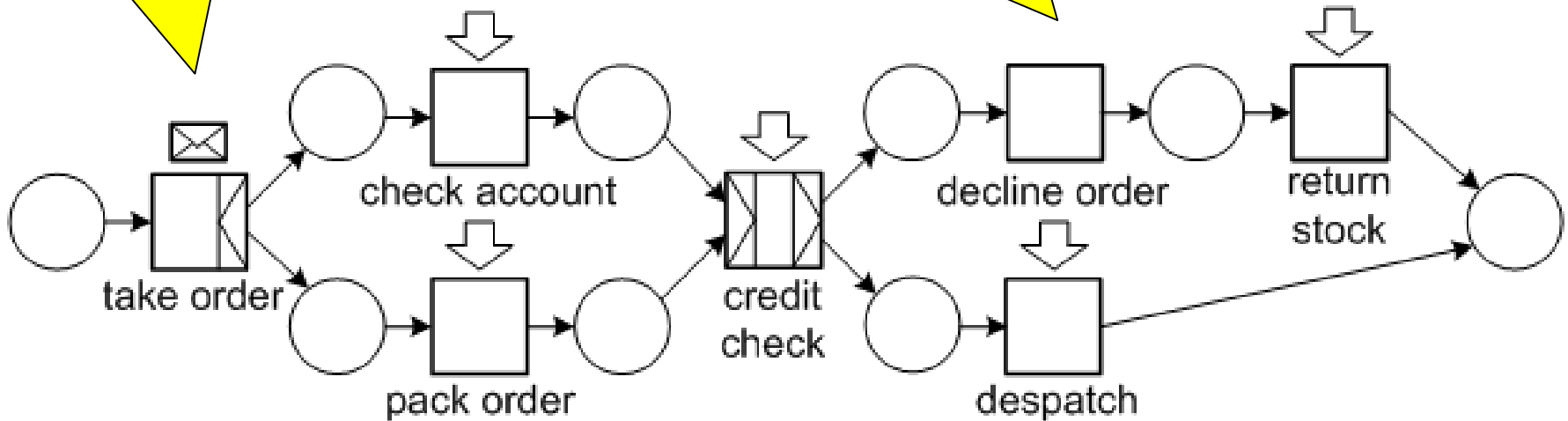
- According to the WfMC [WfMC], a **XOR-join** is “a point within the workflow where two or more alternative activity(s) workflow branches re-converge to a single common activity as the next step within the workflow.”
- As no parallel activity execution has occurred at the join point, no synchronization is required.
- Therefore, D is enabled when B or C complete.
- In WF-nets, a special construct for XOR-JOIN is introduced.

Workflow net Example – order fulfillment process



The **take order** task is externally triggered when an order request is received.

The **decline order** task runs automatically with the customer receiving a notification either by email or fax.



Most tasks are undertaken by human resources (i.e., staff).

Workflow nets: How to decide soundness?

(see [AH02] p276)



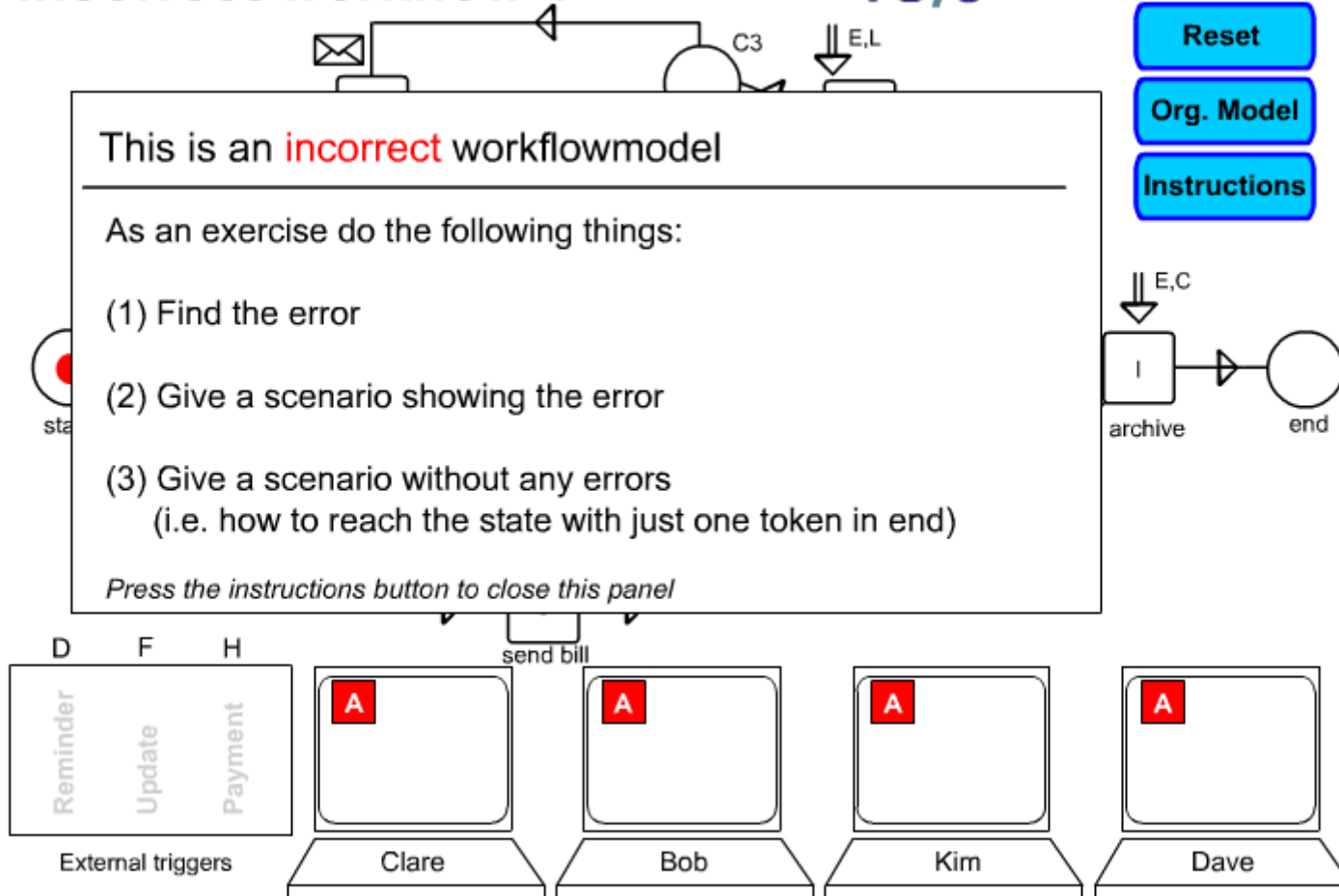
- In [Aalst97] it was shown that soundness for a WF-net could be determined in terms of liveness and boundedness. In [AH02] p.276 this is explained as determining that a workflow net PN is sound is equivalent to determining to whether the net PN' which is constructed through the addition of an extra transition t, where $\bullet t = \{o\}$ and $t\bullet = \{i\}$, is live and bounded.
- As pointed out in [AH02] p.277, the computational complexity of determining whether a WF-net is sound may be quite high. Restrictions (e.g. requiring the net to be free choice) can be imposed to make this more tractable, see the discussion in [AH02] p277-286.
- At Eindhoven University of Technology the Workflow Analyzer (WOFLAN) was developed which is freely available for download.

Workflow Animation – Erroneous WF



Incorrect workflow 1

TU/e technische universiteit eindhoven



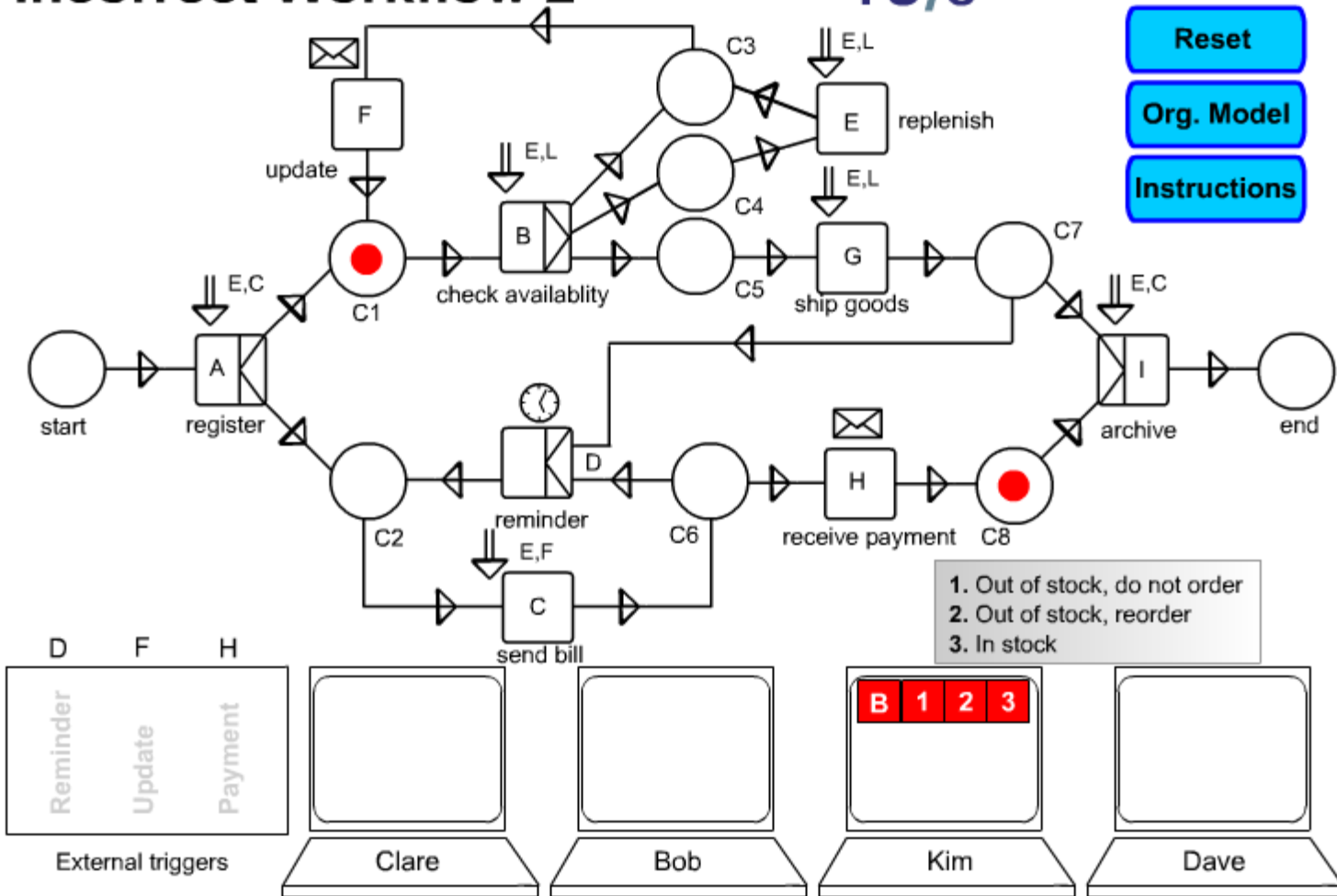
Click on a work-item to select a piece of work for a specific case.

/ faculteit technologie management

© Wil van der Aalst, Vincent Almering en Herman Wijbenga

Animation by Wil van der Aalst, Vincent Almering and Herman Wijbenga

Incorrect Workflow 2



Click on a work-item to select a piece of work for a specific case.

/ faculteit technologie management

© Wil van der Aalst, Vincent Almering en Herman Wijbenga

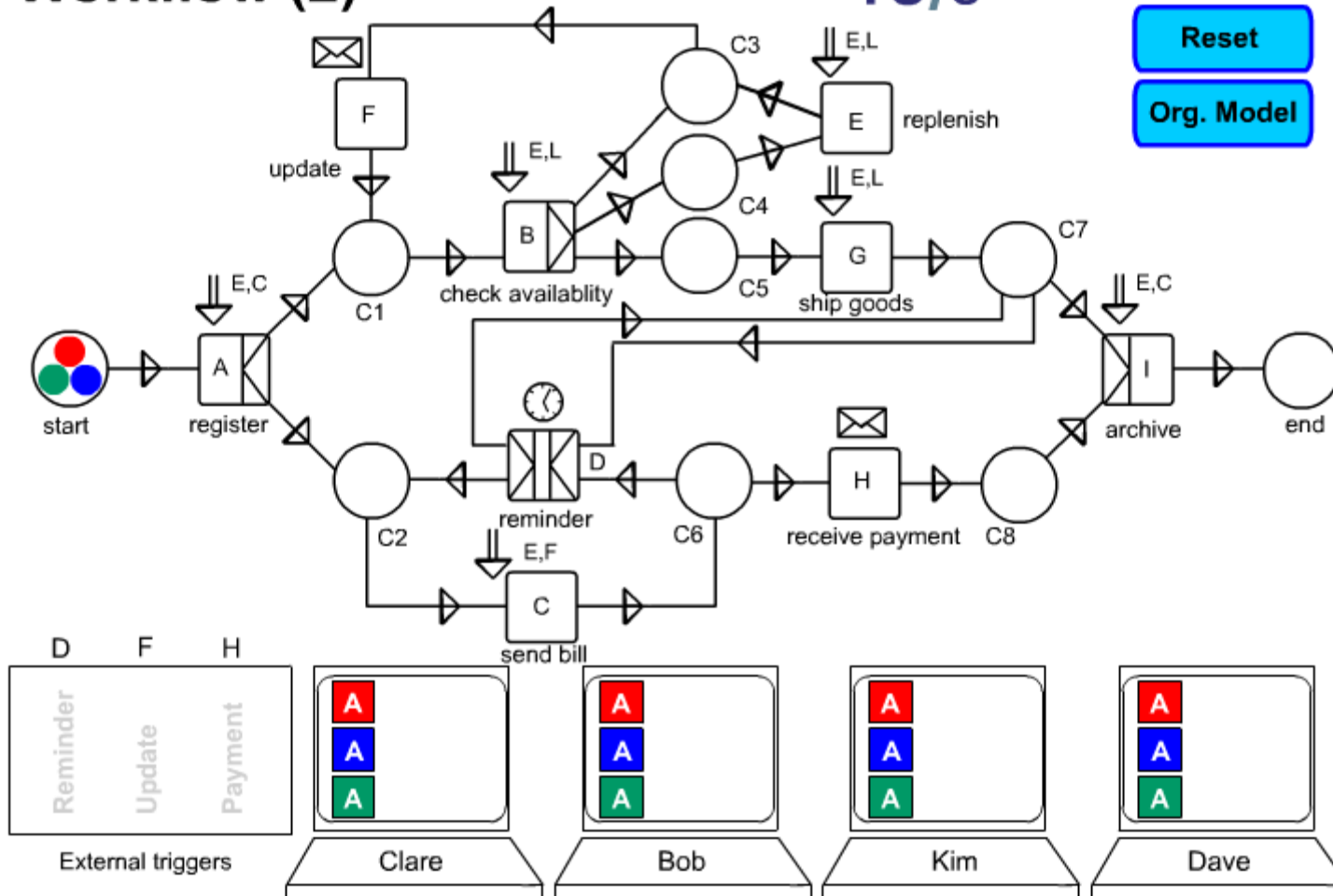
Animation by Wil van der Aalst, Vincent Almering and Herman Wijbenga

Workflow Animation – Correct WF



Workflow (2)

TU/e technische universiteit eindhoven



Reset
Org. Model

Click on a work-item to select a piece of work for a specific case.

/ faculteit technologie management

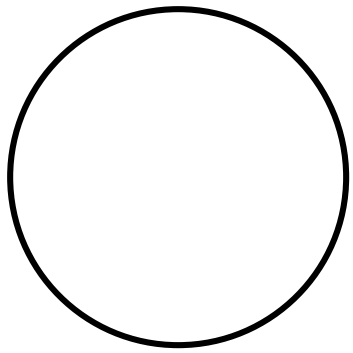
© Wil van der Aalst, Vincent Almering en Herman Wijbenga

Animation by Wil van der Aalst, Vincent Almering and Herman Wijbenga

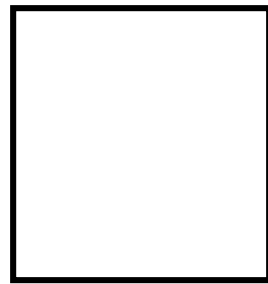


- While Petri nets and Workflow nets provide an effective means of modeling a business process, **they are not able to capture the notion of cancellation.**
 - This is a significant omission, given the relative frequency of cancellation in real-life processes.
- **Reset nets** extend Petri nets with a special type of arc, the *reset arc*, to capture the notion of cancellation.
- Like normal input arcs, reset arcs **connect places to transitions.**
- However, they operate in a different way: when a transition to which a reset arc is connected fires, **any place connected to the transition by reset arcs is emptied of any tokens that it may contain.**
- Note that the tokens in places attached to a transition by reset arcs **play no part in the enablement of the transition.**
 - This makes reachability **undecidable.**

Reset nets



place



transition



arc



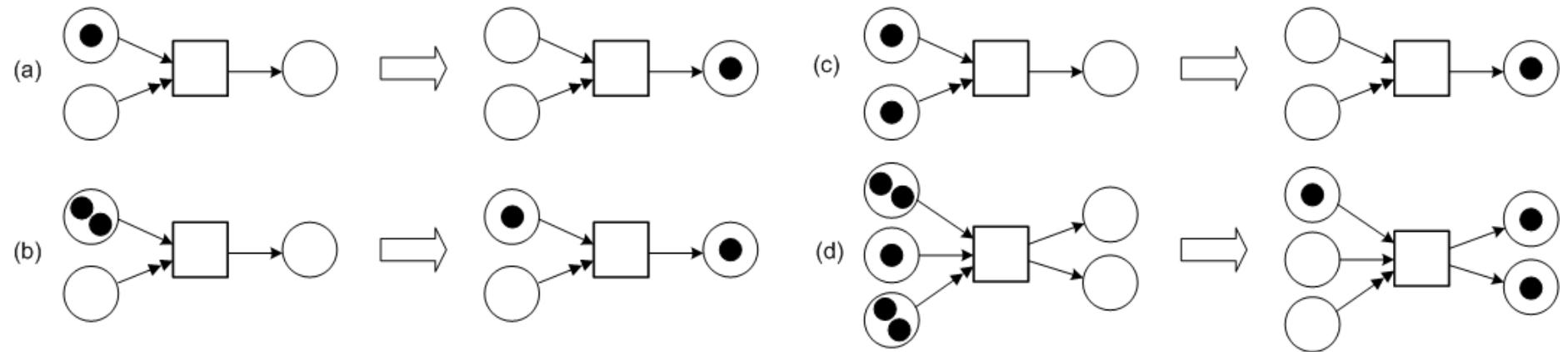
reset arc

Reset nets: formal definition

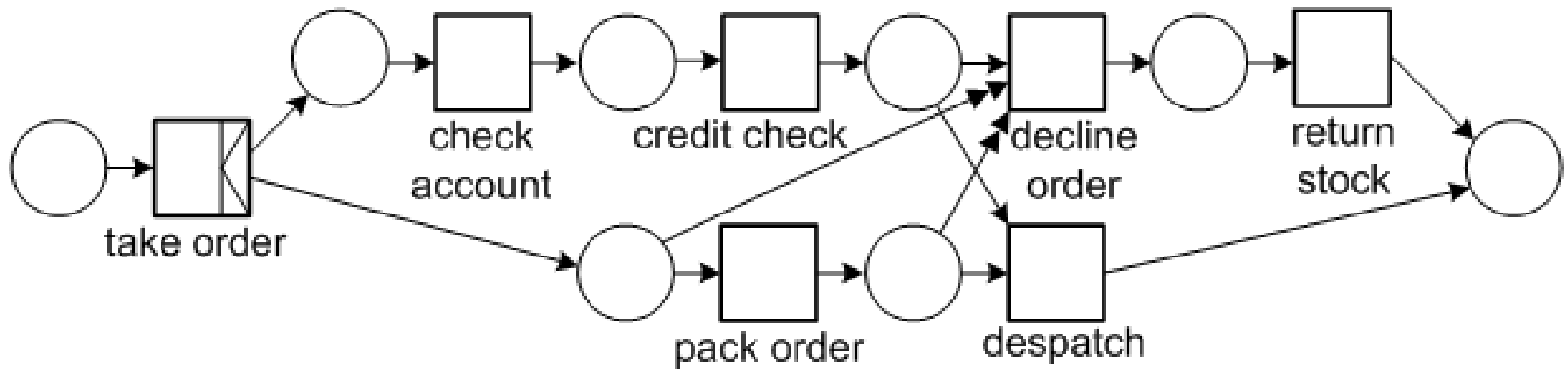


- Formal definitions are based on [DFS98, FRSB02, FS01].
- Syntactically a **Reset net** is a tuple (P, T, F, R) where
 - (P, T, F) is a Petri net with a finite set of places P , a finite set of transitions T , and a flow relation $F \subseteq (P \times T \cup T \times P)$;
 - $R: T \rightarrow 2^P$ is a function associating reset places with transitions.
- A reachable marking M' is defined by:
 - first removing the tokens needed to enable t from its input places $\bullet t$;
 - then removing all tokens from reset places associated with t (i.e., $R(t)$);
 - and finally by adding tokens to its output places $t\bullet$.
- Let $N = (P, T, F, R)$ be a Reset net and M a marking.
 - A transition $t \in T$ is enabled iff $\bullet t \leq M$.
 - An enabled transition t can fire thus changing the state to M' , denoted $M \xrightarrow{t} M'$, with $M' = (M - \bullet t)[P \setminus R(t)] + t\bullet$.
- The definition of occurrence sequence extends naturally from Petri nets.

Reset nets



Reset nets: the order fulfillment process



Sources and References



- [Aalst96] Wil M. P. van der Aalst. Three Good Reasons for Using a Petri-net-based Workflow Management System. In S. Navathe and T. Wakayama, editors, Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), pages 179-201, Cambridge, Massachusetts, November 1996.
- [Aalst97] Wil M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, Applications and Theory of Petri Nets 1997, volume 1248 of Lecture Notes in Computer Science, pp 407-426, Springer Verlag, 1997.
- [AH02] Wil M.P. van der Aalst and Kees M. van Hee. Workflow Management: Models, Methods, and Systems. The MIT Press, 2002.
- [JB96] S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture and Implementation. International Thomson Computer Press, 1996.
- [BW90] J. Baeten and W.P. Weijland. Process Algebra. Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.
- [DE95] J. Desel and J. Esparza. Free Choice Petri Nets. Cambridge Tracts in Theoretical Computer Science 40, Cambridge University Press, 1995.
- [DFS98] C. Dufourd, A. Finkel, and P. Schnoebelen. Reset nets between decidability and undecidability. In K. Larsen, S. Skyum, and G. Winskel, editors, Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98), volume 1443 of Lecture Notes in Computer Science, pages 103–115, Aalborg, Denmark, July 1998. Springer.
- [FRSB02] A. Finkel, J.-F. Raskin, M. Samuelides, and L. van Begin. Monotonic extensions of petri nets: Forward and backward search revisited. Electronic Notes in Theoretical Computer Science, 68(6):1–22, 2002.
- [FS01] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! Theoretical Computer Science, 256(1–2):63–92, April 2001.
- [JK09] K. Jensen and L.M. Kristensen. Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Springer 2009.
- [Peterson81] J.L.A. Peterson. Petri net theory and the modeling of systems. Prentice Hall, 1981.
- [HAAR09] A.H.M. ter Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell (editors). Modern Business Process Automation: YAWL and Its Support Environment. Springer, 2010.
- [WfMC] Workflow Management Coalition - Terminology & Glossary, Document number WFMC-TC-1011, Document Status 3.0, February 1999. Downloaded from <http://www.aiim.org/wfmc/mainframe.htm>. (this document contains the quoted definitions)
- [KHA03] B. Kiepuszewski, A.H.M. ter Hofstede and W.M.P. van der Aalst. Fundamentals of Control Flow in Workflows. Acta Informatica 39(3):143-209, 2003.
- [KHB00] B. Kiepuszewski, A.H.M. ter Hofstede, C. Bussler. On Structured Workflow Modelling. Proceedings CAiSE'2000, Lecture Notes in Computer Science 1789, Stockholm, Sweden, June 2000.
- [Kie03] B. Kiepuszewski. Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows. PhD thesis, Queensland University of Technology, Brisbane, Australia, 2003.
- www.workflowcourse.com (among others for the animations)

