

Kapitel 3:

Workflow-Modellierungssprachen – Einführung in High-Level Petrinetze

1. Überblick über Modellierungssprachen
 1. Ziel: Analyse
 2. Modellierungssprachen: Perspektiven und Anforderungen
2. Petrinetze und Workflow-Modellierung
 1. Geschichtliches
 2. Gründe und Bestandteile
 3. Interpretationen
 4. Formales
 5. High-Level Petrinetze
 6. Prozessdefinition mit Petrinetzen

Vorlesung Workflow-Management-Systeme
Unter Verwendung von Folien der Vorlesung
WFMS von Prof. Stucky, AIFB, WS 2000/01

1 Ziel der Modellierung: Analyse (I)

Arten der Analyse

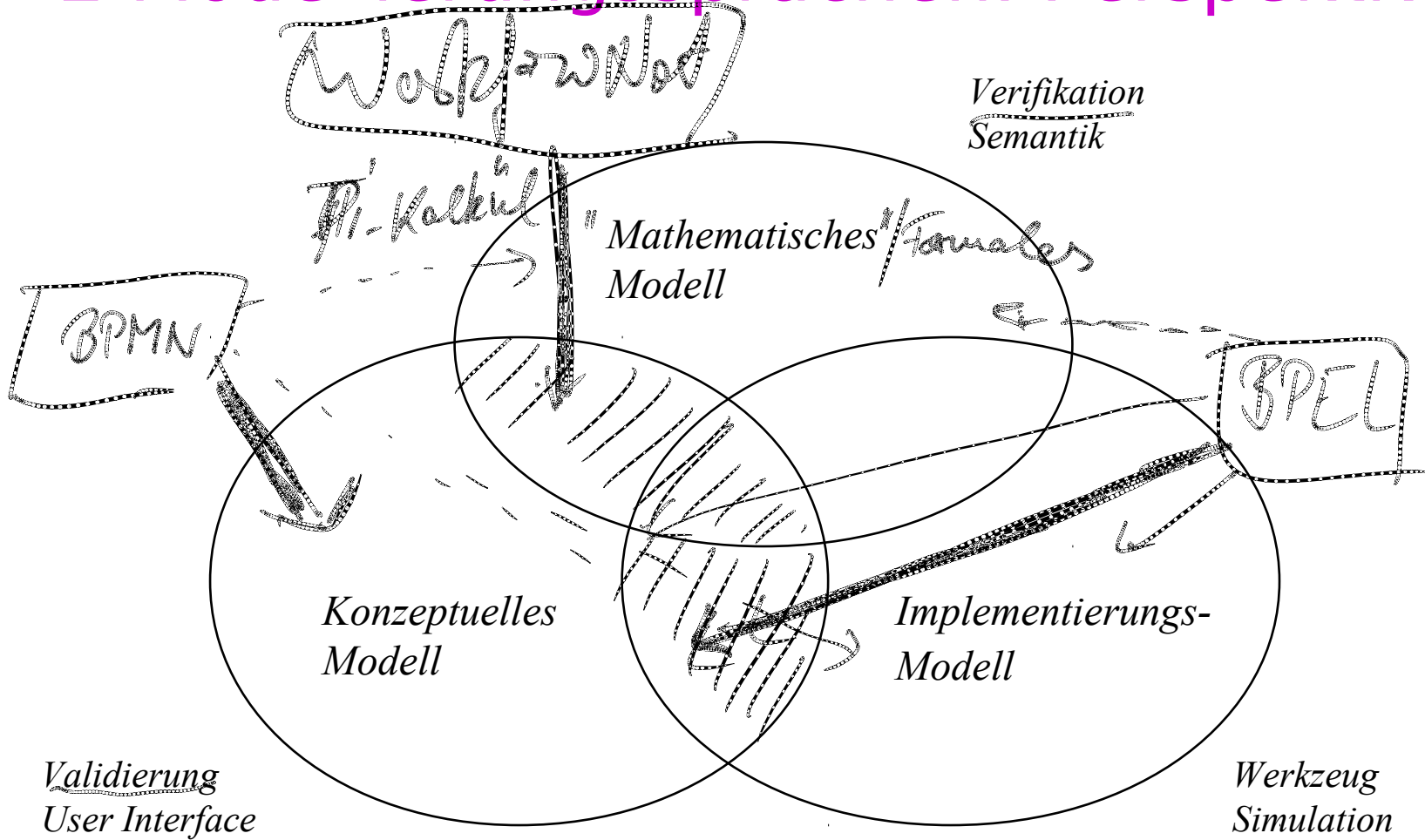
- Validierung
 - » Ist das Modell richtig bzgl. der Realität/Vorstellung?
 - » z.B. Kundenbezug, Medien- und Organisationsbrüche
- Verifikation
 - » Nachweis der Korrektheit des Geschäftsprozesses
 - » Struktur (z.B. Vor- und Nachbedingungen für alle Aktivitäten)
 - » Verhalten (z.B. Deadlocks, nie ausgeführte Aktivitäten)
- Leistungsbewertung
 - » Leistungsfähigkeit des Geschäftsprozesses
 - » z.B. Durchlaufzeit, Kostenrechnung, Ressourcenauslastung

1 Ziel der Modellierung: Analyse (II)

Analyseverfahren

- „Hinschauen“
- Korrekte Verfahren
 - » automatische Verifikation:
Ermittlung spezieller Eigenschaften (z.B. Invarianten, Ziele,...)
 - » semiautomatische Verifikation:
Beweisverfahren (z.B. Proof Checker)
- Simulation
 - » sequentielle Abläufe/ nichtsequentielle Abläufe
 - » animierte Simulation

2 Modellierungssprachen: Perspektiven



2 Modellierungssprachen: Anforderungen

- Analysemöglichkeit für Geschäftsprozesse
- Aktivitäten und Übergänge
- Rollen und ihre Struktur
- Input / Output von Aktivitäten
- externe Applikationen und ihre Einbindung in Workflows
- Daten / Datenbanken
- Simulationen für Navigation

2 Modellierungssprachen: Anforderungen

- Formale Semantik
- Graphisch
- Leicht zu lernen
- Leicht zu benutzen
- Hohe Ausdrucksmächtigkeit
- Werkzeugunterstützung
- Unabhängigkeit von Herstellern
- Explizite Darstellung von Zuständen und Ereignissen

*dynamisches
Verhalten
Interaktion mit
Umgebung/keine*

2 Modellierungssprachen: Überblick

- Flussdiagramme
- Datenflussdiagramme (DFD, ISAC, SADT, ...)
SADT: Structured Analysis and Design Technique
- Transitions-Systeme, Zustandsdiagramme
- Warteschlangen-Modelle, Markov-Ketten
- Prozess-Algebren (z.B. ACP, CSP, CCS, ...)
ACP: Algebra of Communicating Processes
CSP: Communicating Sequential Processes
- Herstellerspezifische Modellierungssprachen
 - » Ereignisgesteuerte Prozessketten (EPK)
- (High-level) Petrinetze

High-Level-Petrinetze

1. Geschichtliches
2. Gründe und Bestandteile
3. Interpretationen
4. Formales
5. High-Level Petrinetze
6. Prozessdefinition mit Petrinetzen

1 Petrinetze: Geschichtliches

- Ursprung: Dissertationsschrift
"Kommunikation mit Automaten" von Carl Adam Petri
(1962)
- Seither: mehr als 10.000 Arbeiten auf dem Gebiet
- bis 1985: hauptsächlich von Theoretikern benutzt
- seit Mitte der 80er Jahre: vermehrter Einsatz in
praktischen Anwendungen
- Gründe:
 - » Einführung der High-Level-Netze
 - » Entwurf von Werkzeugen

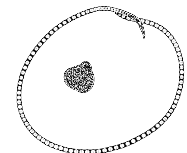
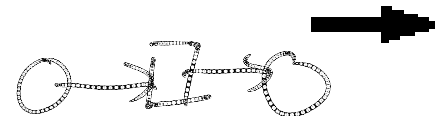
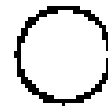
2 Petrinetze: Gründe

- Graphischer Formalismus
- Formale Syntax und Semantik
- Explizite Darstellung von Zuständen
- Herstellerunabhängig
- Viele Analyseverfahren und -werkzeuge



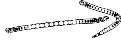
2 Petrinetze: Bestandteile

Ein Petrinetz besteht aus

- Stellen
- Transitionen
- Verbindungen zwischen Stellen und Transitionen
- Marken in Stellen



3 Petrinetze: Interpretationen (I)

- *Transitionen* 
 - » Aktionen, Handlungen, Transporte, Transformationen, Anweisungen, Programme, ...
- *Stellen* 
 - » Bedingungen, Medien, Materialbehälter, Datenträger, Puffer, Nachrichtenkanäle,...
- *Verbindungen* 
 - » Vor- und Nachbedingungen von Aktivitäten, Start und Ziel von Transporten, Eingabe und Ausgabe von Programmen,...

3 Petrinetze: Interpretationen (II)

□ *Marken*

- » Zustände einer Bedingung, Gültigkeit von Bedingungen,
Füllungsgrad von Speichern,
Daten auf Datenträgern,
Nachrichten in Puffern, ...

□ *Markierungen*

- » lokale Zustände
- » Gesamtzustände

4 Petrinetze: Formales (I)

Struktur von Petrinetzen

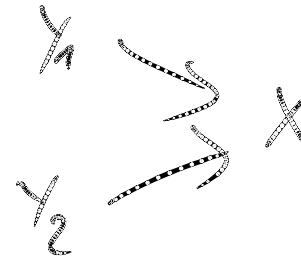
- Ein **Petrinetz** ist ein Tripel $N = (S, T, F)$ mit
 - » S (Stellen), T (Transitionen) sind endliche Mengen
 - » $S \cap T = \emptyset$
 - » $S \cup T \neq \emptyset$
 - » $F \subseteq (S \times T) \cup (T \times S)$ ist eine binäre Relation über $S \cup T$

- Alle Stellen und Transitionen eines Netzes heißen **Netzelemente**.

4 Petrinetze: Formales (II)

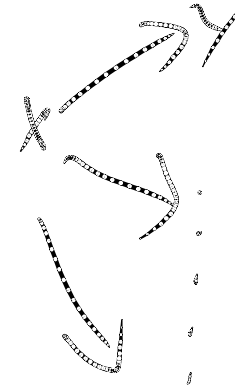
- **Vorbereich** eines Elements x :
Menge aller Eingangs-Knoten
(bzw. Input-Knoten) von x

$$\bullet x = \{y \mid (y, x) \in F\}$$



- **Nachbereich** eines Elements x :
Menge aller Ausgangs-Knoten
(bzw. Output-Knoten) von x

$$x \bullet = \{y \mid (x, y) \in F\}$$



4 Petrinetze: Formales (III)

Verzweigungen

Netzleert
Ein Knoten heißt

- **vorwärtsverzweigt**, falls $|x \bullet| > 1$,
- **rückwärtsverzweigt**, falls $|\bullet x| > 1$.

Vorwärtsverzweigte Stellen:

modellieren Alternativen (Konflikte),

rückwärtsverzweigte Transitionen:

modellieren Synchronisation.

4 Petrinetze: Formales (IV)

Teil-Netz

Ein Netz $N' = (S', T', F')$ heißt **TeilNetz** des Netzes $N = (S, T, F)$,

wenn

- i. $S' \subseteq S$ und
- ii. $T' \subseteq T$ und
- iii. $F' = F \cap ((S' \times T') \cup (T' \times S'))$.

4 Petrinetze: Formales (V)

Rand

Der **Rand** eines Teil-Netzes N' (bzgl. des Netzes N) sind diejenigen seiner Knoten, die über Kanten mit dem Restnetz verbunden sind. Er ist also definiert durch:

$$\{x \in S' \cup T' \mid (x^\bullet \cup \bullet x) \setminus (S' \cup T') \neq \emptyset\}$$

(Vor- und Nachbereich von x sind hierbei bzgl. N zu verstehen)

Ein Teil-Netz N' heißt

stellenberandet,

wenn sein Rand nur Stellen enthält,

transitionsberandet,

wenn sein Rand nur Transitionen enthält.

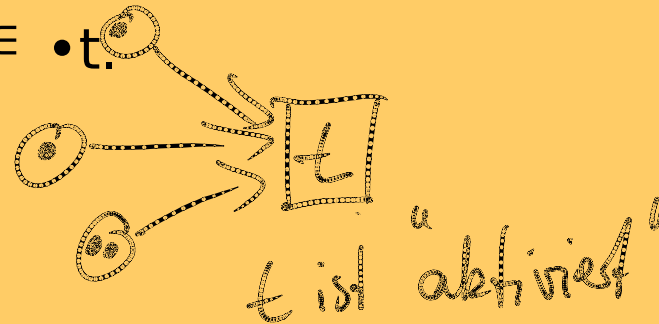
4 Petrinetze: Formales (VI)

Markierung

Eine **Markierung** m eines Netzes $N=(S,T,F)$

ist eine Abbildung $m: S \rightarrow \mathbb{N}$.

Eine Markierung m **aktiviert** eine Transition $t \in T$,
wenn $m(s) > 0$ für alle $s \in \bullet t$.



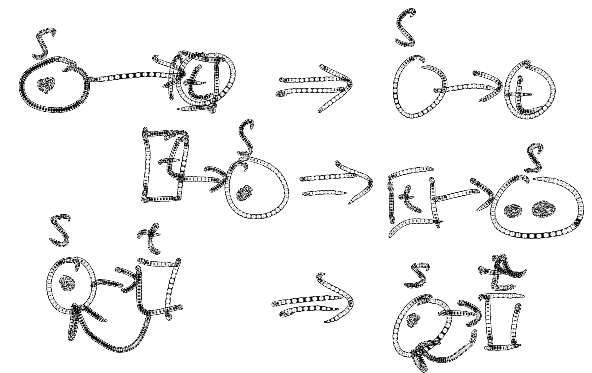
4 Petrinetze: Formales (VII)

„Schaltregel“

Falls t unter m aktiviert ist, kann t schalten.

Dies führt zu einer **Folgemarkierung** m' , definiert durch

$$m'(s) = \begin{cases} m(s) & , \text{ falls } s \notin \bullet t \text{ und } s \notin t \\ m(s) - 1 & , \text{ falls } s \in \bullet t \text{ und } s \notin t \\ m(s) + 1 & , \text{ falls } s \notin \bullet t \text{ und } s \in t \\ m(s) & , \text{ falls } s \in \bullet t \text{ und } s \in t \end{cases}$$



Schreibweise: $m \xrightarrow{t} m'$

Ein Netz N mit Markierung m wird als

Stellen/Transitions-Netz bezeichnet und durch das Paar (N, m)

angegeben: **S/T-Netz (N, m)**

4 Petrinetze: Formales (VIII)

Schaltfolge

Sei m eine Markierung eines S/T-Netzes N .

Falls $m \xrightarrow{t_1} m_1, m_1 \xrightarrow{t_2} m_2, \dots, m_{n-1} \xrightarrow{t_n} m_n$ Schaltvorgänge

sind,

ist $\tau = t_1 t_2 \dots t_n$ eine **Schaltfolge** von m nach m_n :

$(m \rightarrow m_n)$

ε

Dies gilt auch für die leere Sequenz ε : $m \rightarrow m$ für jede Markierung m .

3.2.4 Petrinetze: Formales (IX)

Wir schreiben $m \xrightarrow{\tau}^* m'$ und nennen m' von m
erreichbar,

wenn $m \xrightarrow{\tau} m'$ für irgendeine Schaltfolge τ gilt.

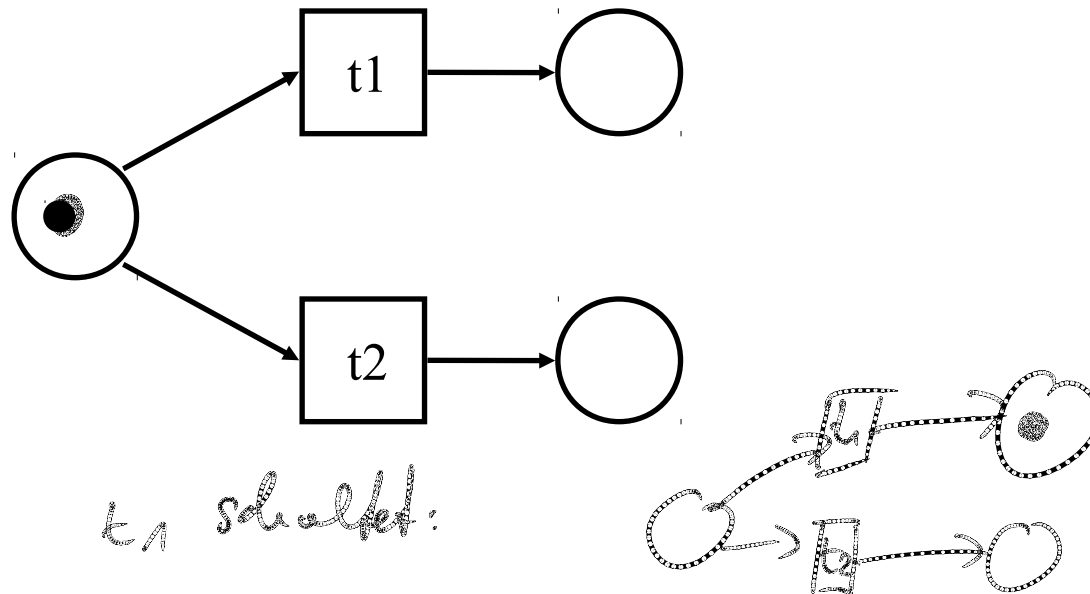
$[m>$ bezeichnet die Menge aller von m **erreichbaren**
Markierungen.

Das Verhalten eines markierten S/T-Netzes wird
 beschrieben durch die Menge seiner Schaltfolgen.
 Eine kompaktere Repräsentation liefert der
 Markierungsgraph.

4 Petrinetze: Formales (X)

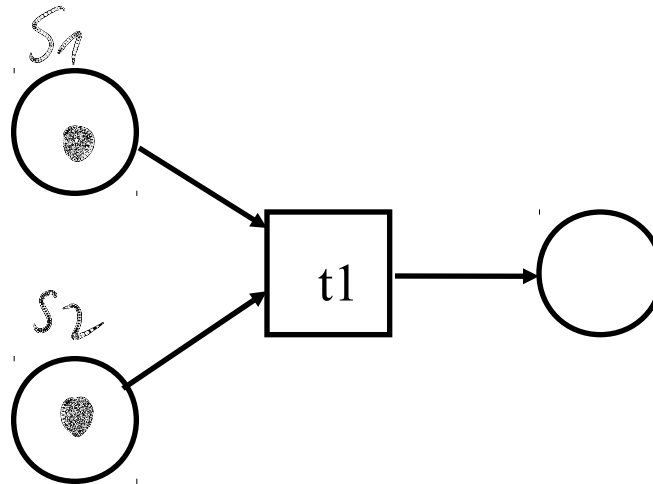
□ Konflikt

» Zwei Transitionen benötigen die gleiche Marke(n)

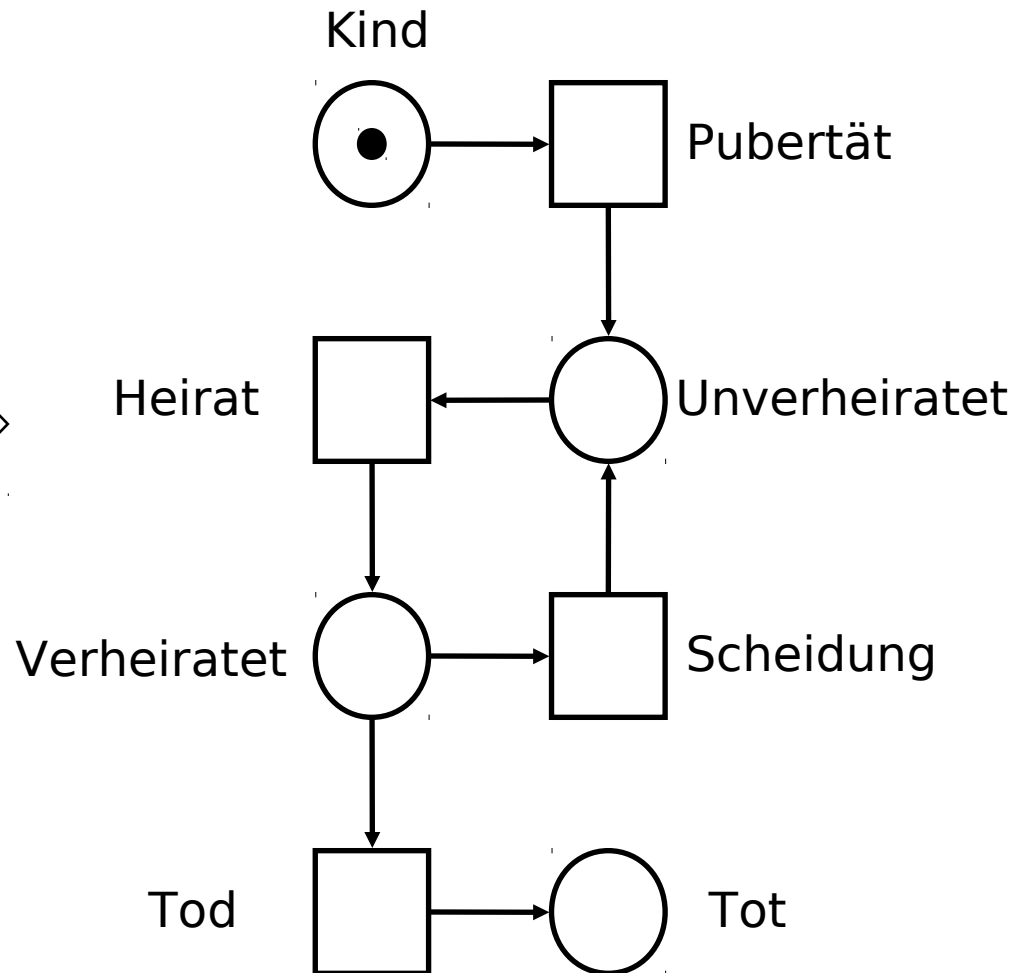
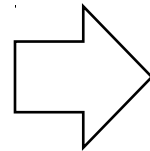


4 Petrinetze: Formales (XI)

□ Synchronisation



4 Beispiel: Lebenszyklus



Was stimmt da
nicht?

5 High-Level-Petrinetze (I)

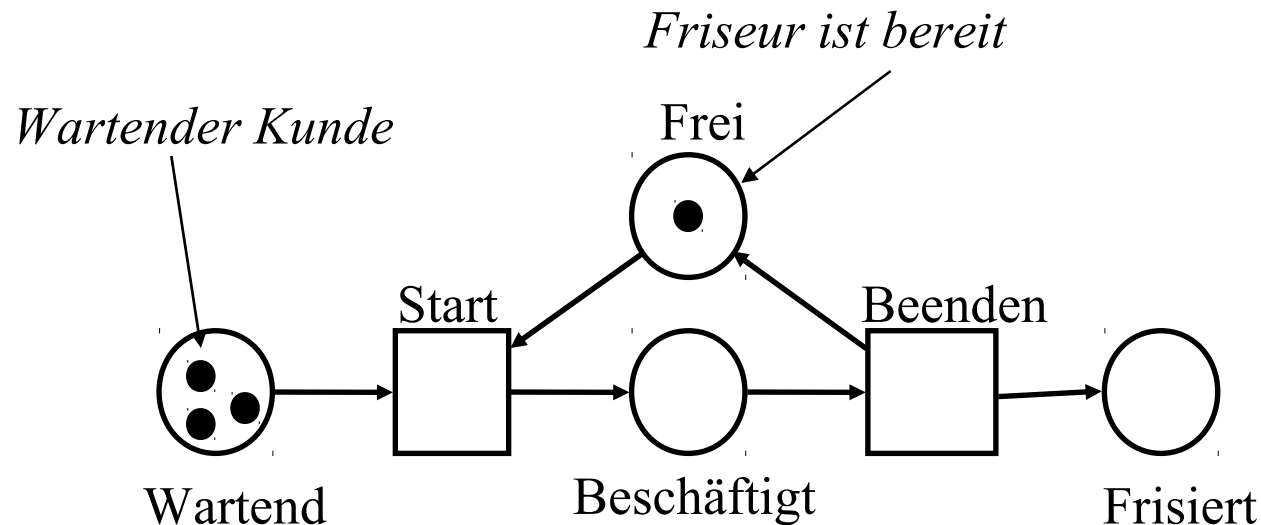
- Beim praktischen Einsatz von „klassischen“ Petrinetzen treten oft Probleme auf:
 - » Die Modelle werden zu groß und komplex.
 - » Die Modellierung ist langwierig und kompliziert.
 - » Zeit, Kosten und Daten können nicht modelliert werden.

5 High-Level-Petrinetze (II)

High-Level Petrinetze sind Petrinetze mit den folgenden Erweiterungen:

- Unterscheidbare Marken
 - » zur Modellierung von Attributen
 - » auch: gefärbte Marken, colored tokens
- Zeit
 - » zur Performance-Analyse
 - » verschiedene Zeitkonzepte
- Hierarchie
 - » zur Strukturierung der Modelle
 - » Modellierung auf verschiedenen Abstraktionsebenen

5 High-Level-Petrinetze – „Beim Friseur“ (I)

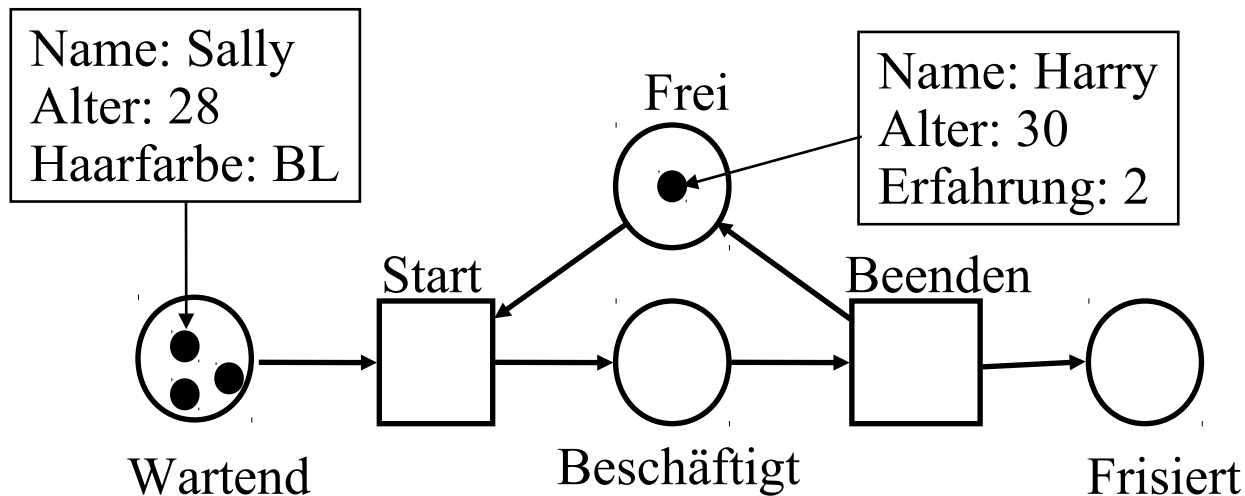


Anmerkung:

man beachte die problemlose Modellierung einer Situation mit mehreren Friseuren

5 High-Level-Petrinetze – „Beim Friseur“ (II)

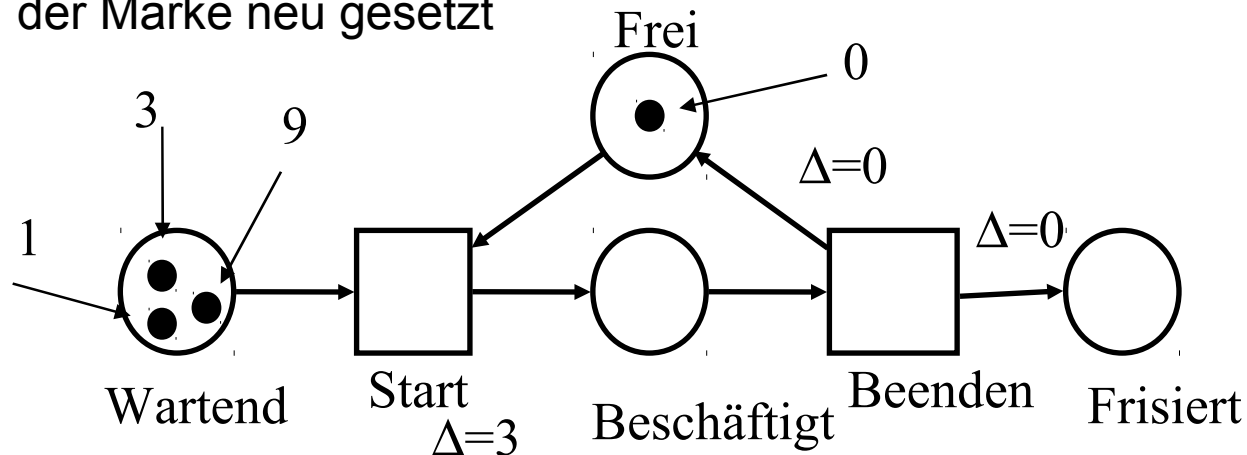
- Eine unterscheidbare Marke
 - » stellt ein Objekt mit einer Menge von Attributen dar.
 - » beinhaltet Werte für alle Attribute.



5 High-Level-Petrinetze (III)

□ Die Erweiterung mit Zeit:

- » für Performance-Analysen werden Zeitdauern, Verzögerungen etc. benötigt
- » Jede Marke bekommt einen Zeitstempel
- » Durch Transitionen wird das Alter (und damit die Verfügbarkeit) der Marke neu gesetzt

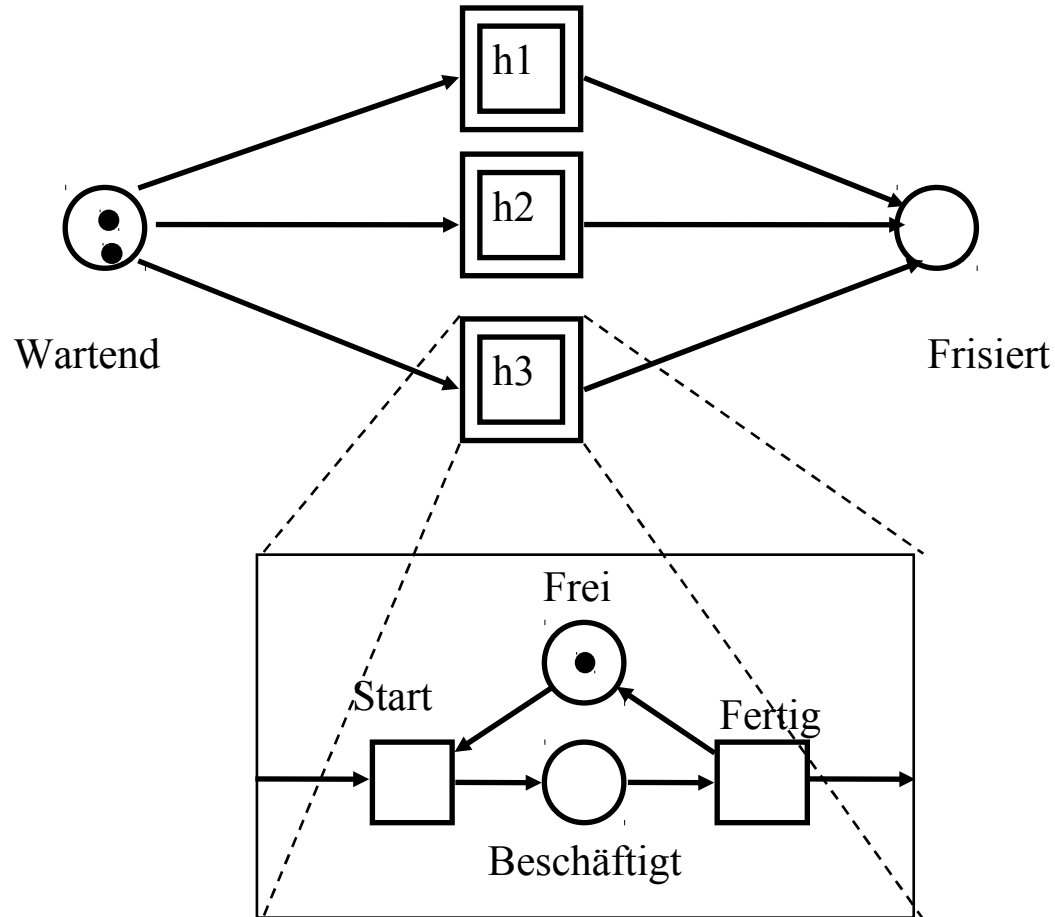


- » vier verschiedene Zeitkonzepte
(hier: Bestimmung der Verzögerung jedes Tokens)

5 High-Level-Petrinetze (IV)

- Die Erweiterung um Hierarchie-Konzepte
 - » Ein Mechanismus zur Strukturierung komplexer Modelle ermöglicht das Modellieren und die Darstellung auf verschiedenen Abstraktionsstufen
 - » Ein Netzelement wird durch ein entsprechend berandetes Teilnetz (auch: Subnetz) ersetzt.

5 High-Level-Petrinetze – „Beim Friseur“ (III)

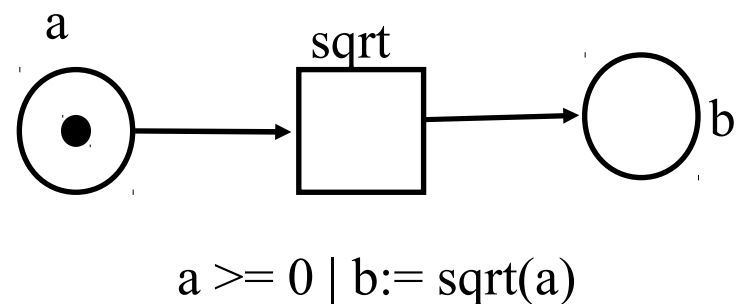
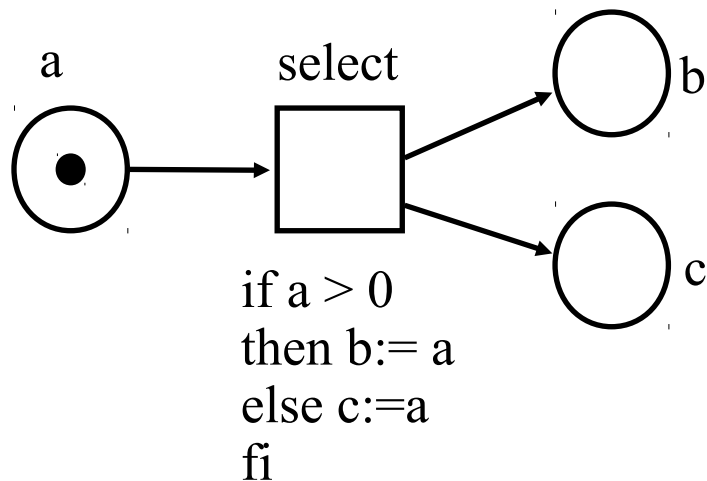
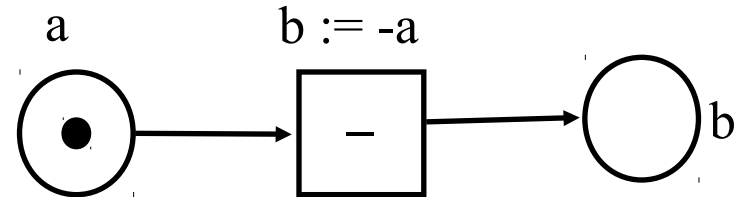
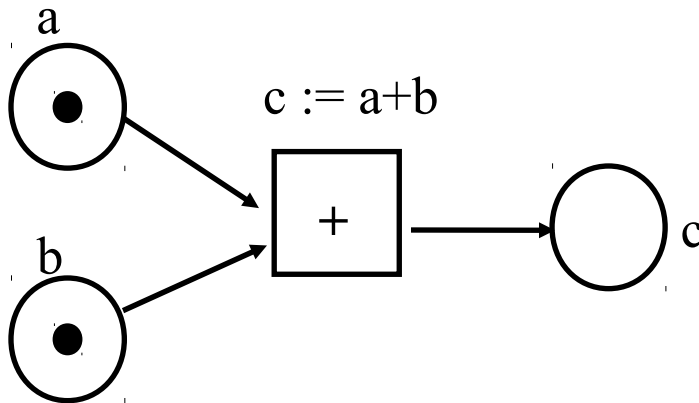


5 High-Level-Petrinetze (V)

Für jede **Transition** wird spezifiziert:

- die Anzahl der produzierten Marken
- die Werte der entsprechenden Attribute
- die Anzahl der konsumierten Marken
- (optional) eine Schaltbedingung (auch: guard)

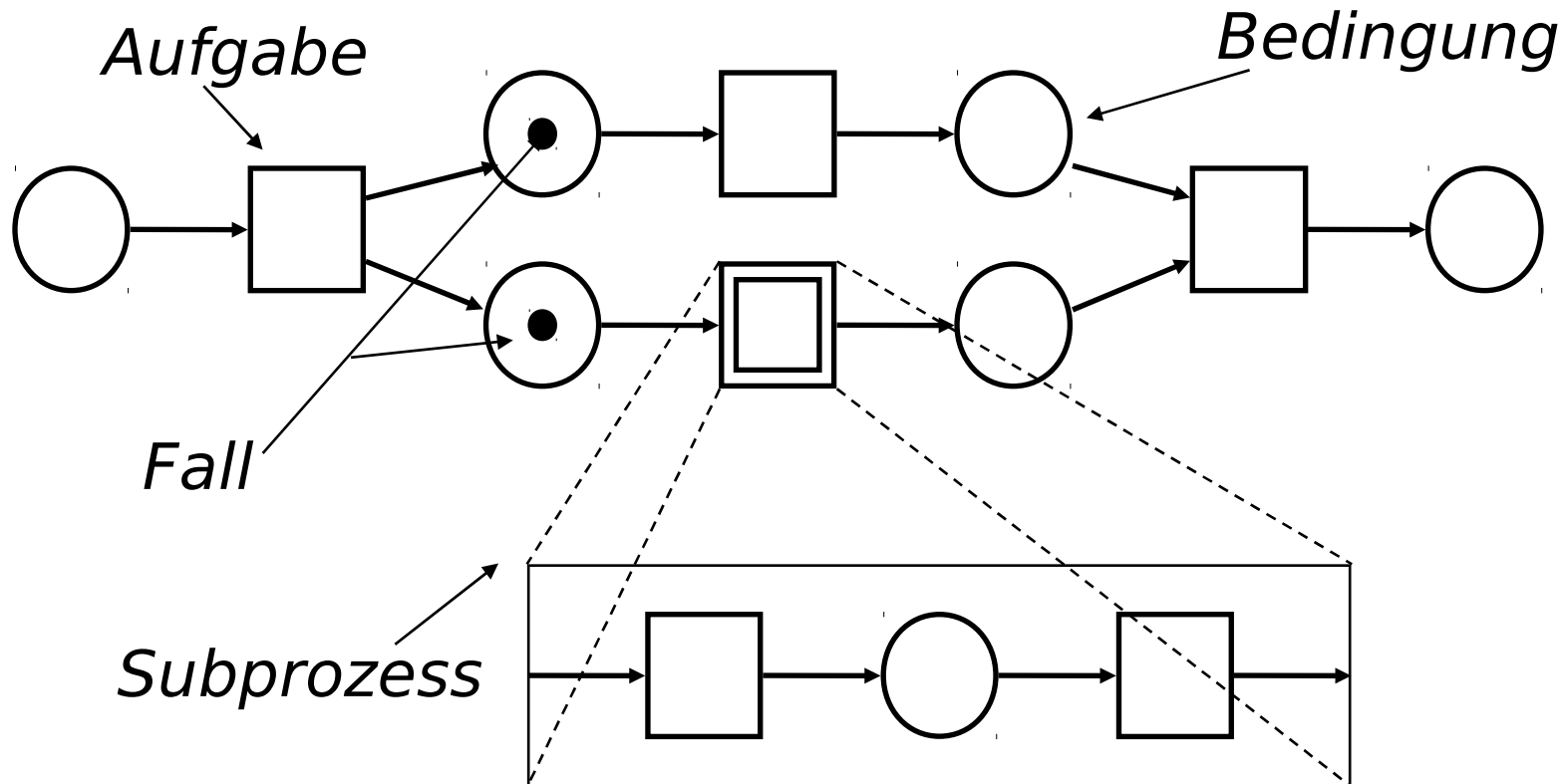
5 High-Level-Petrinetze - Beispiele



6 Terminologie bei Petri-Netzen für WFs

- A **Case** is the 'thing' which needs to be processed by following the process definition.
- Eine **Prozess-Definition** legt die Aufgaben und ihre Reihenfolge fest.
- **Bedingungen** legen die Reihenfolge der Aufgaben fest, sie können wahr oder falsch sein.
- Eine Aufgabe hat **Vor-** und **Nachbedingungen**.

6 Prozessdefinition mit Petri-Netzen



6 Routing von Fällen

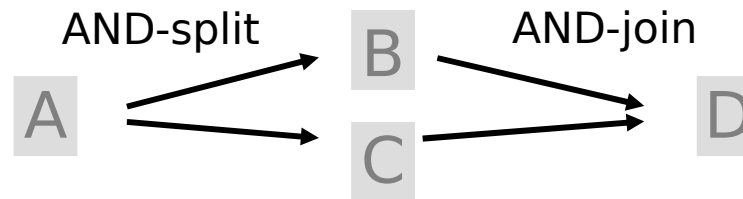
Abstraktion

A, B, C, D - Aufgaben

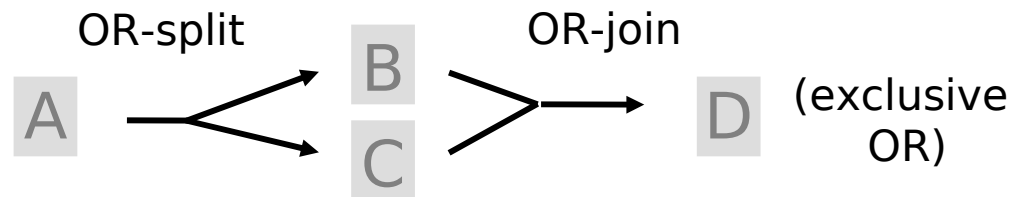
- sequentiell



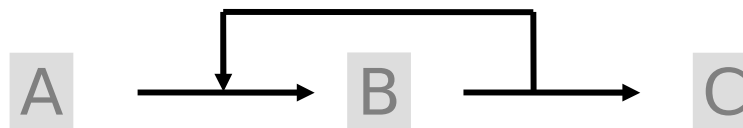
- parallel



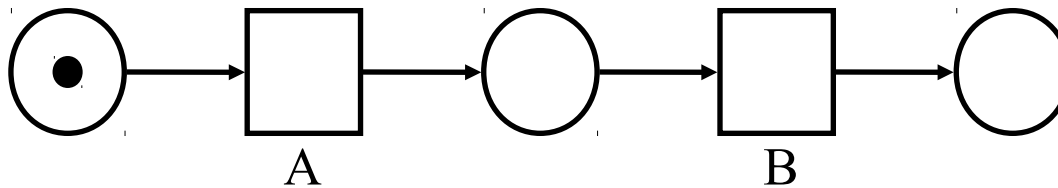
- wahlweise



- iterativ

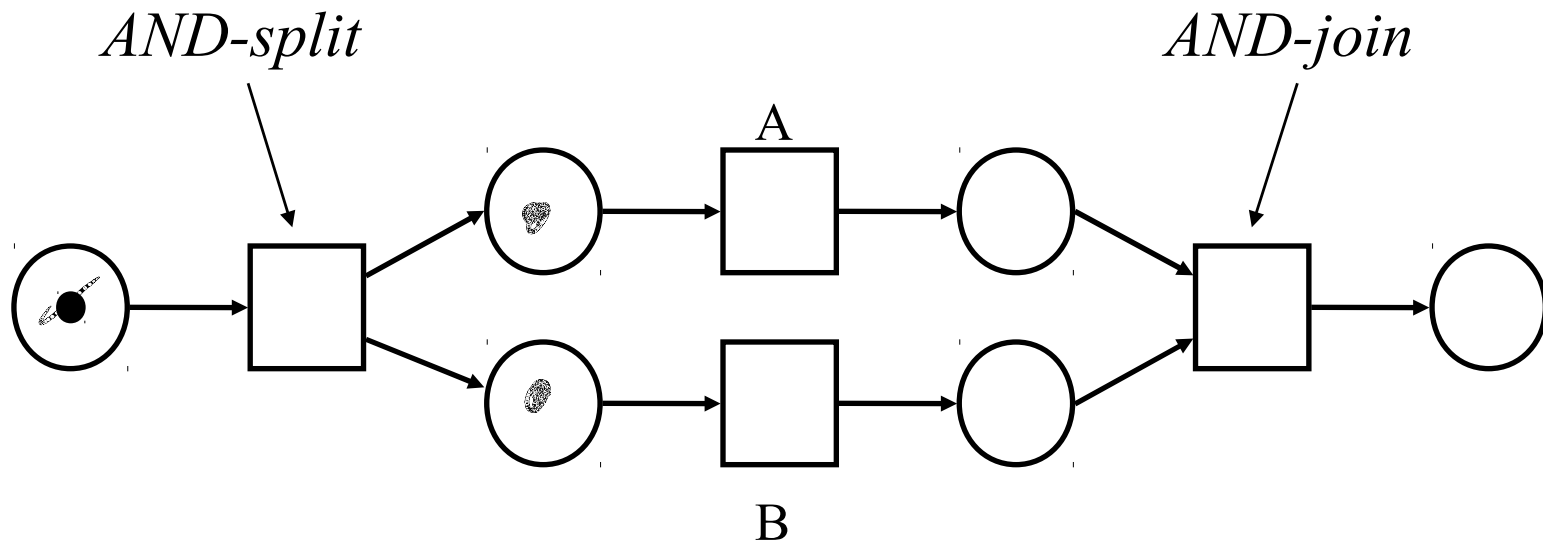


6 Sequentielles Routing



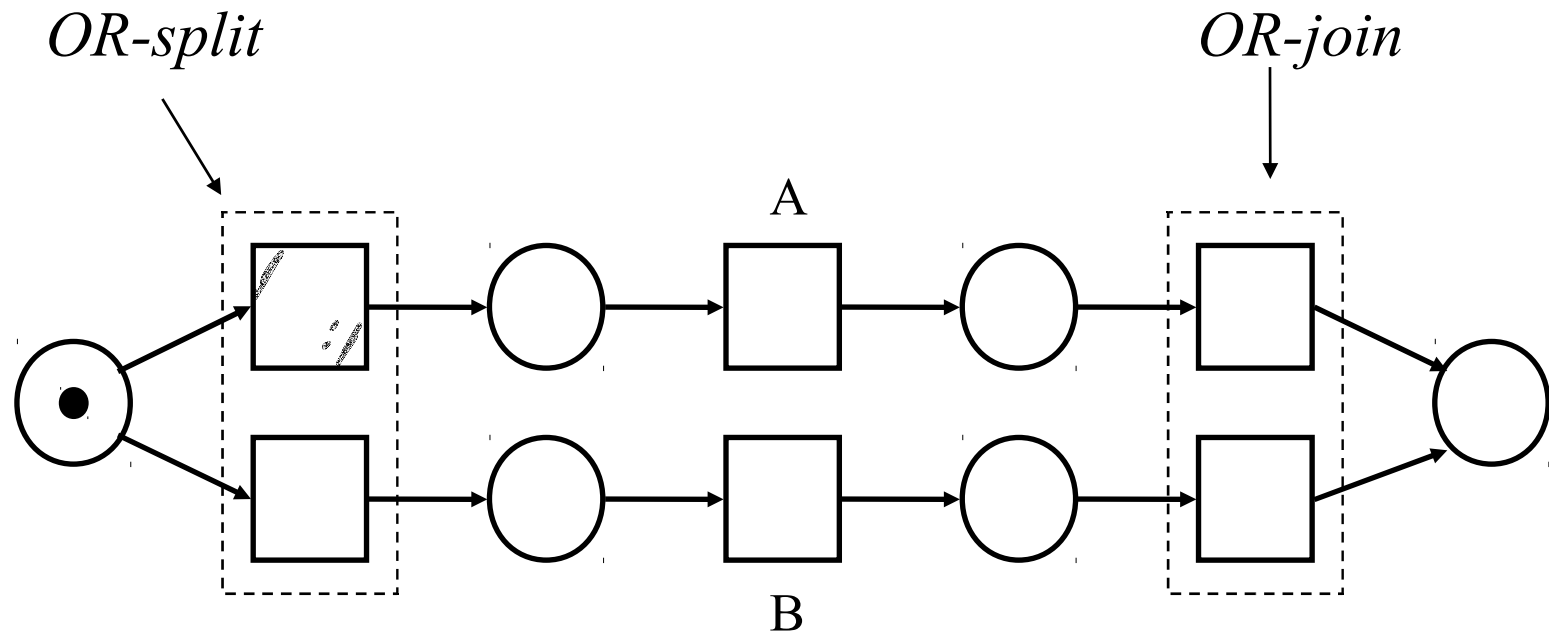
“Erst A, dann B”

6 Paralleles Routing



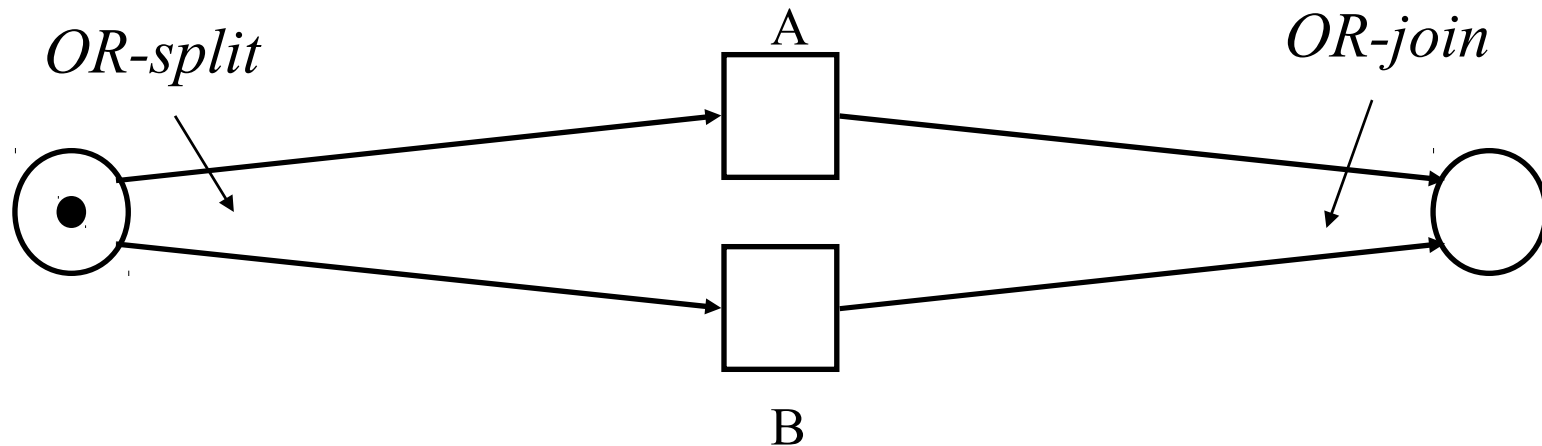
"A und B nebenläufig"

6 Auswahl (I)



"A oder B" (exklusiv)

6 Auswahl (II)

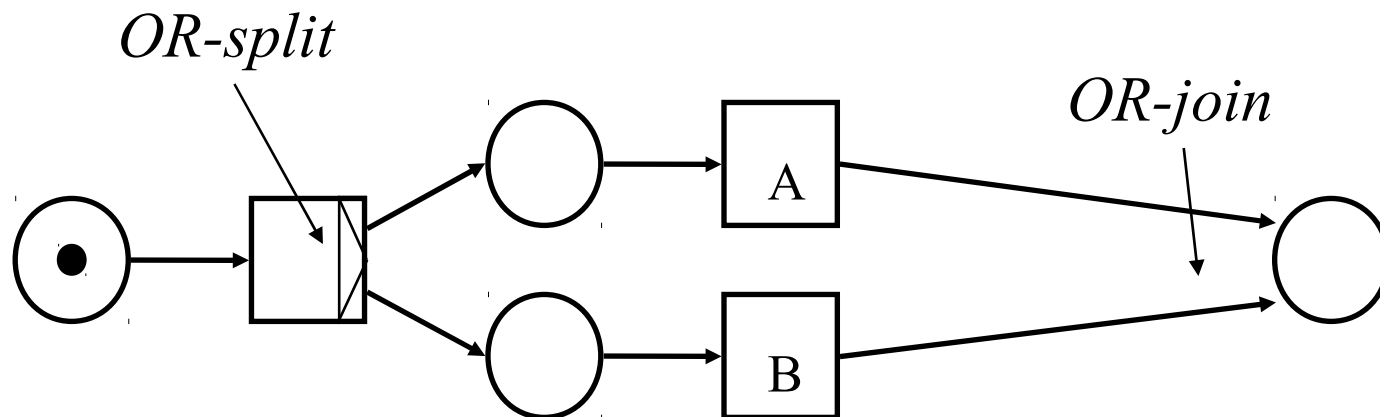


Implizite Auswahl: hängt von A und B ab

(d.h. die genaue Verzweigung steckt implizit in den Transitionen)

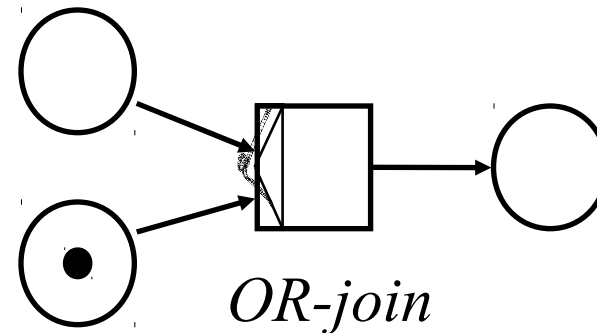
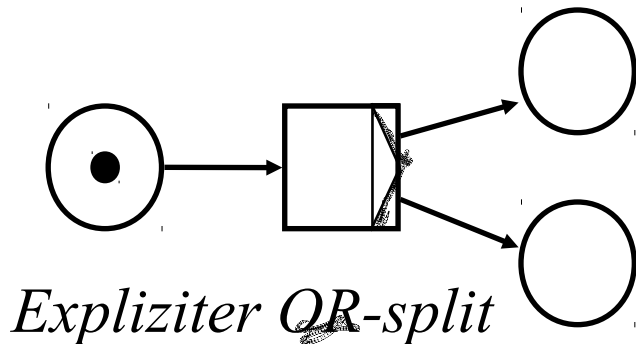
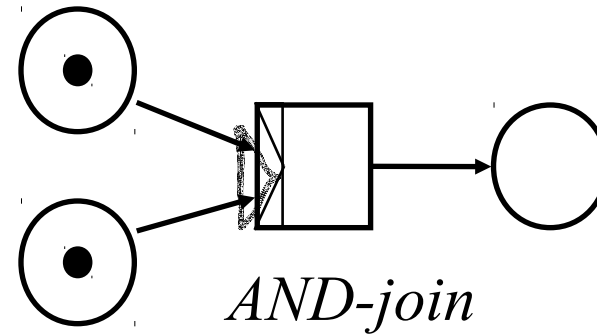
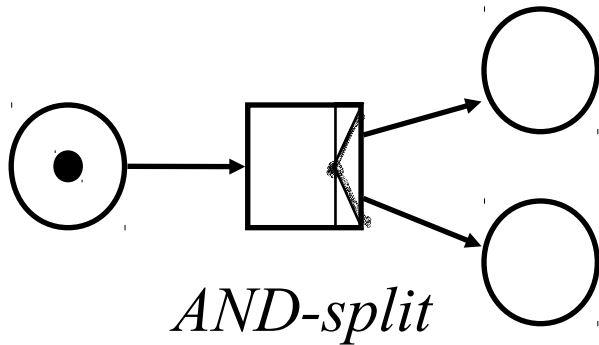
6 Auswahl (III)

Ein Modellierungsprimitiv zur Modellierung expliziter Auswahl:

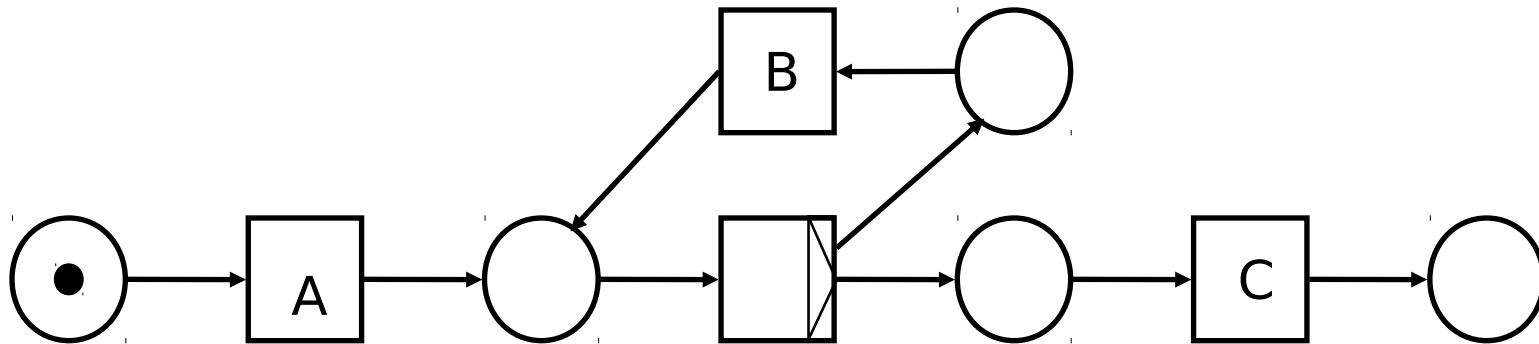


Explizite Auswahl: hängt nicht von A und B ab

6 Überblick über die Routing-Erweiterungen



6 Iteration



6 Trigger

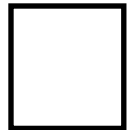
- Ein Workflowsystem ist ein reaktives System und als solches von der Umwelt abhängig:
 - » Die Ankunft einer EDI-Nachricht.
 - » Der Arbeitsbeginn einer Ressource.
 - » Die Ankunft einer Akte.
 - » Ein Anruf zur Auftragsbestätigung.

- Einige Aufgaben erfordern Trigger.

6 Trigger - Arten von Aufgaben

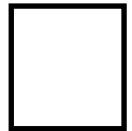
□ Automatisch

» Kein Trigger erforderlich.



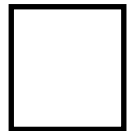
□ Benutzer

» Initiiert durch eine Ressource.



□ Extern

» Ein externer Event (z.B. Nachricht, Anruf) ist erforderlich.

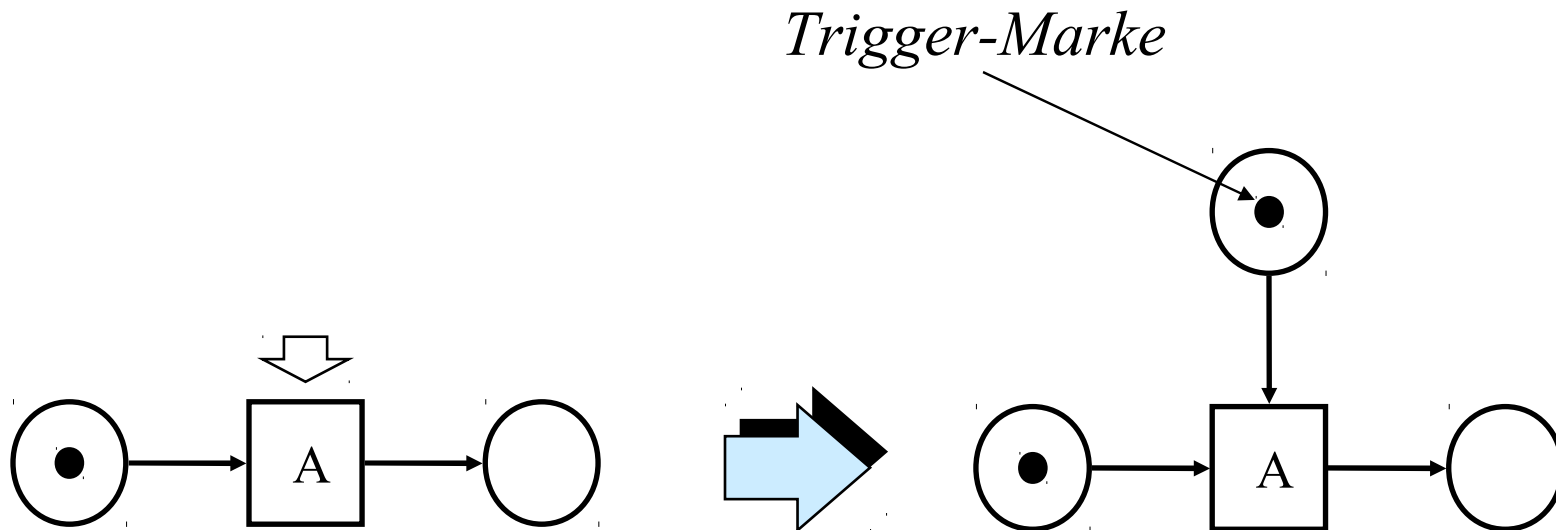


□ Zeit

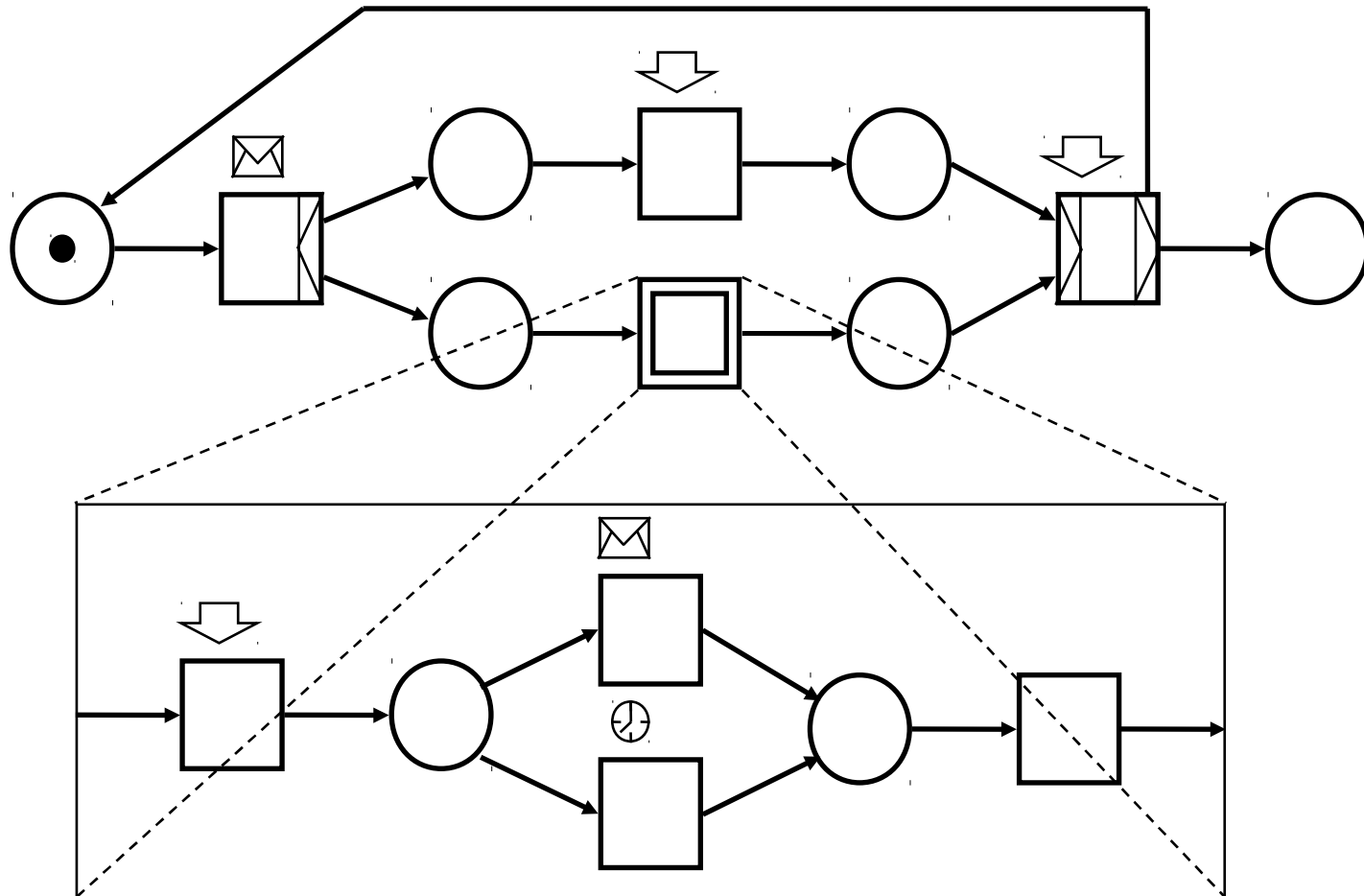
» Zeitdauern oder Zeitpunkte werden berücksichtigt.



6 Trigger – Darstellung als Stelle



6 Prozessdefinition – Alle Konstrukte



Exemplarische Fragen

- Was sind Vorteile von Petri-Netzen für die Ablaufmodellierung?
- Wozu werden Petri-Netze im Workflow-Bereich eingesetzt?
- Welche Analyseformen gibt es? Wo sind Petri-Netze nutzbar?
- Wie ist ein Petri-Netz aufgebaut?
- Welche Erweiterungen sind in High-Level-Petri-Netzen enthalten und wozu dienen diese?
- Was ist der Rand eines (Teil-)Petri-Netzes?
- Wie nutzt man Petri-Netze zur Prozessdefinition?
- Welche Routing-Konstrukte werden angeboten? Was sind die Vorteile, wenn solche Konstrukte genutzt werden?