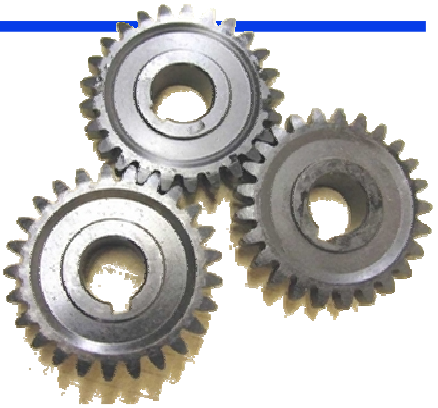
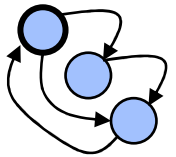
Modellierung und Analyse eingebetteter und verteilter Systeme

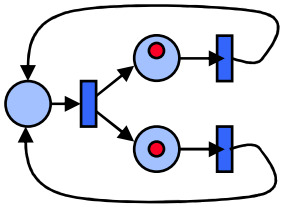
—

Thread „Funktionalität“ *Teil 1*



- Einleitung
- Zustandstransitionssysteme
- Petrinetz und Partialordnungsmodelle
- Prozessalgebra: CCS
- Temporale Logik: LTL, CTL, CTL*
- Erreichbarkeitsanalyse und Model Checking
- Eigenschaftsbeweise im STS
  - Safety: Zustandsinvarianten und Induktionsbeweis
  - Liveness: Leads-to-Ketten, Fairness- und Lattice-Regeln





X = (a.b.c.Y + D) | X

E((EX.P)U(AG.Q))

F: Funktionaler Thread – Inhalte

1. Einleitung
2. Erweiterter Mealy-Automat
3. Petri Netz
4. Gefärbtes Transitionssystem (LTS)
5. Calculus of Communicating Systems (CCS)
6. Einfaches Zustandstransitionssystem (STS)
7. Safety und Liveness im STS
8. Erreichbarkeitsanalyse
9. Logiken (LTL, CTL, CTL*)
10. Model Checking
11. Safety- und Livenessbeweise

F1: Einleitung zum Thread „Funktionalität“

- Anwendungsdomänen und Systeme
- Eigenschaften: Safety und Liveness
- Komponenten, Ereignisse
- Kopplung und Gesamtsystem
- Prinzipien

F1: Übersicht: Anwendungsdomänen

Systeme

- die aus mehreren, nebenläufig agierenden und miteinander interagierenden Komponenten bestehen
- die über längere Zeiträume unterbrechungs- und störungsfrei arbeiten sollen
- an die besondere Funktionssicherheitsanforderungen gestellt werden

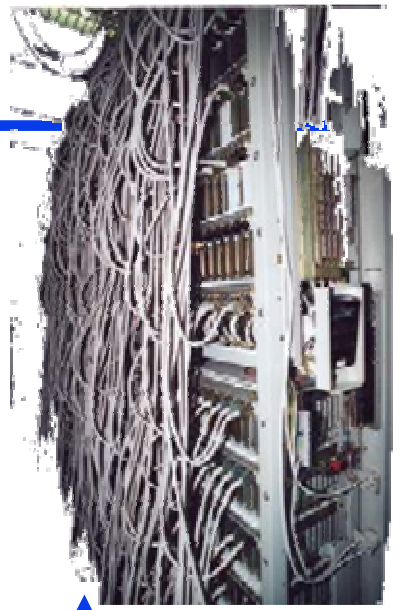
Domänen

- Telekommunikationssysteme
- Steuerungs- und Überwachungssysteme für
  - Produktionsanlagen
  - Energieanlagen
  - Chemieanlagen
  - Verkehrstechnische Anlagen

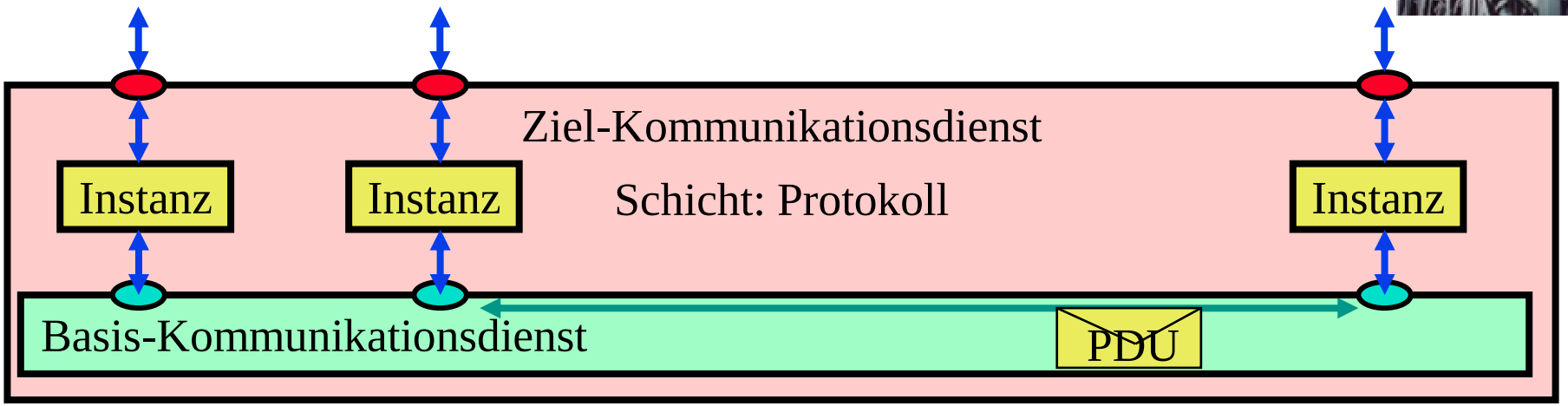
::

- Unternehmenskritische betriebswirtschaftliche Anwendungen
  - Verfügbarkeitsanforderungen
  - Korrektheitsanforderungen
- Informationsverarbeitung mit Datensicherheitsanforderungen
  - Datensicherheit bedingt Funktionssicherheit
    (die Schutzfunktionen müssen funktionssicher implementiert sein)

F1: Anwendungsdomäne: Telekommunikation



- Dienstleistende Systeme bestehend aus Schichten interagierender Protokollinstanzen
- Protokolle und Dienste



- Zuverlässige Systemfunktion trotz des Auftretens von Übertragungsfehlern

F1: Anwendungsdomäne: Steuerungssysteme

- Mit technischen Prozessen über Aktoren und Sensoren verbunden
- Physikalische Zustandsgrößen
- nicht nur funktionale Anforderungen:
  - Rechtzeitigkeit
  - Datensicherheit
  - Zuverlässigkeit

[DIN66201]: Ein Prozess ist eine Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Materie, Energie oder Information umgeformt, transportiert oder gespeichert wird.

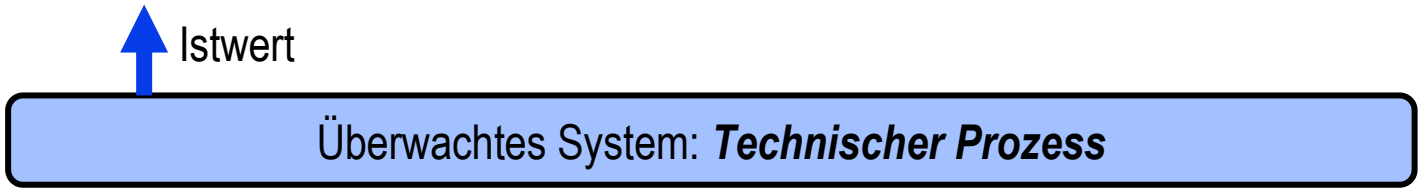
Technischer Prozess: Material oder Energie wird umgeformt, transportiert oder gespeichert.

Rechenprozess: Information wird umgeformt, transportiert oder gespeichert.

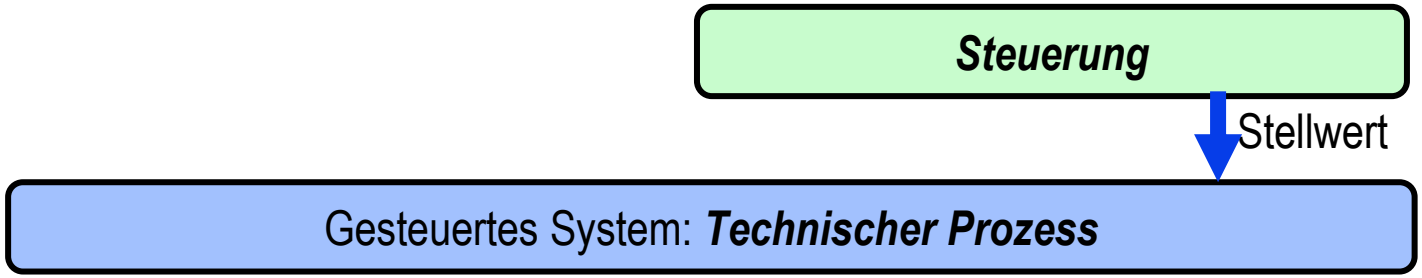
Prozess: „das Fortschreitende“– Vorgang im Ablauf der Zeit – Verhalten

F1: Automatisierung – Steuerung und Regelung

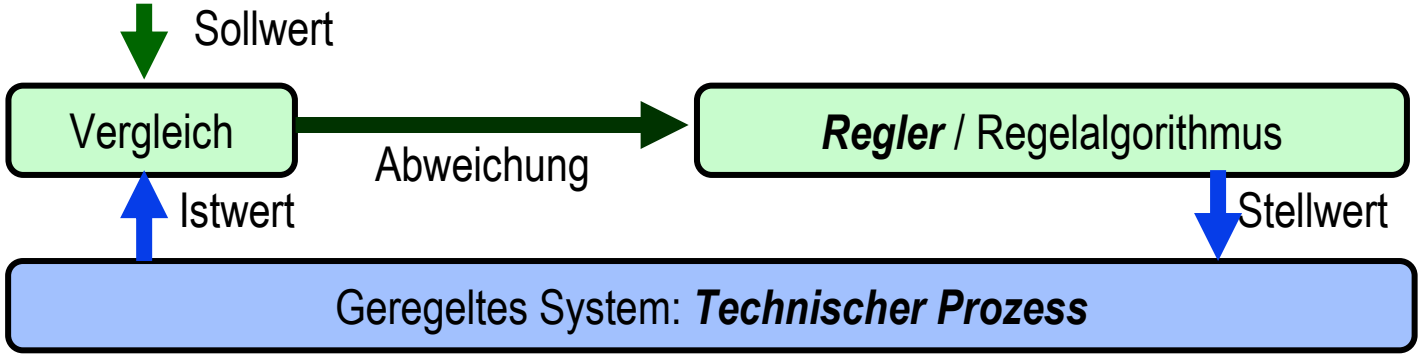
- Überwachung



- Steuerung

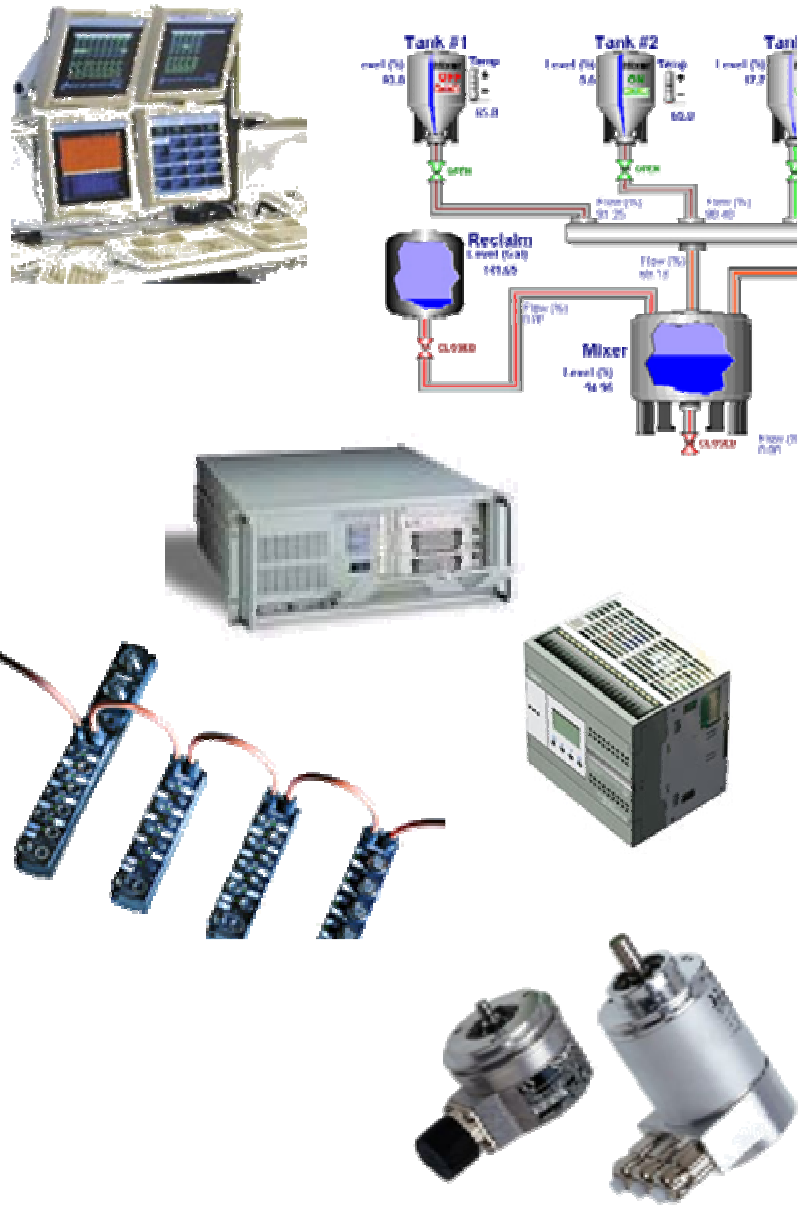


- Regelung, der Regelkreis



F1: Automatisierung – Komponenten

- Verbindungen zum Enterprise Information System (EIS)
- Leitstand
- Industrie PCs (IPCs)
- Programmable Logic Controller (PLCs)
- Kommunikation, Feldbusse
- Instrumentierung des technischen Prozesses, Sensoren, Aktoren



F1: Automatisierung – System- und Prozesstypen

- Typen dynamischer Systeme
  - Analoge Systeme
    - » z.B. Temperaturregelung
  - Diskrete Systeme
    - » z.B. Weichenstellung
  - Hybride Systeme
    - » z.B. Rezeptgesteuerte Chemieanlagen

- Technische Prozess-Typen
  - Fließprozesse
    - » z.B. Stahl-Walzen
  - Folgeprozesse
    - » z.B. Fräsen und Bohren eines Motorblocks
  - Stückgut-Prozesse
    - » z.B. Transport eines Containers

F1: Automatisierung – Anforderungen

- Funktionssicherheit soll verhindern:
  - Fehlerhafte Funktion
  - Ungewünschte Funktion
  - ::
  - Zerstörung der technischen Anlage
  - Gefährdung der Umgebung

- Ziel von Modellierung und Analyse

  Möglichst viele Fehler finden

  Sie kann ein System nicht „als korrekt absegnen“, denn

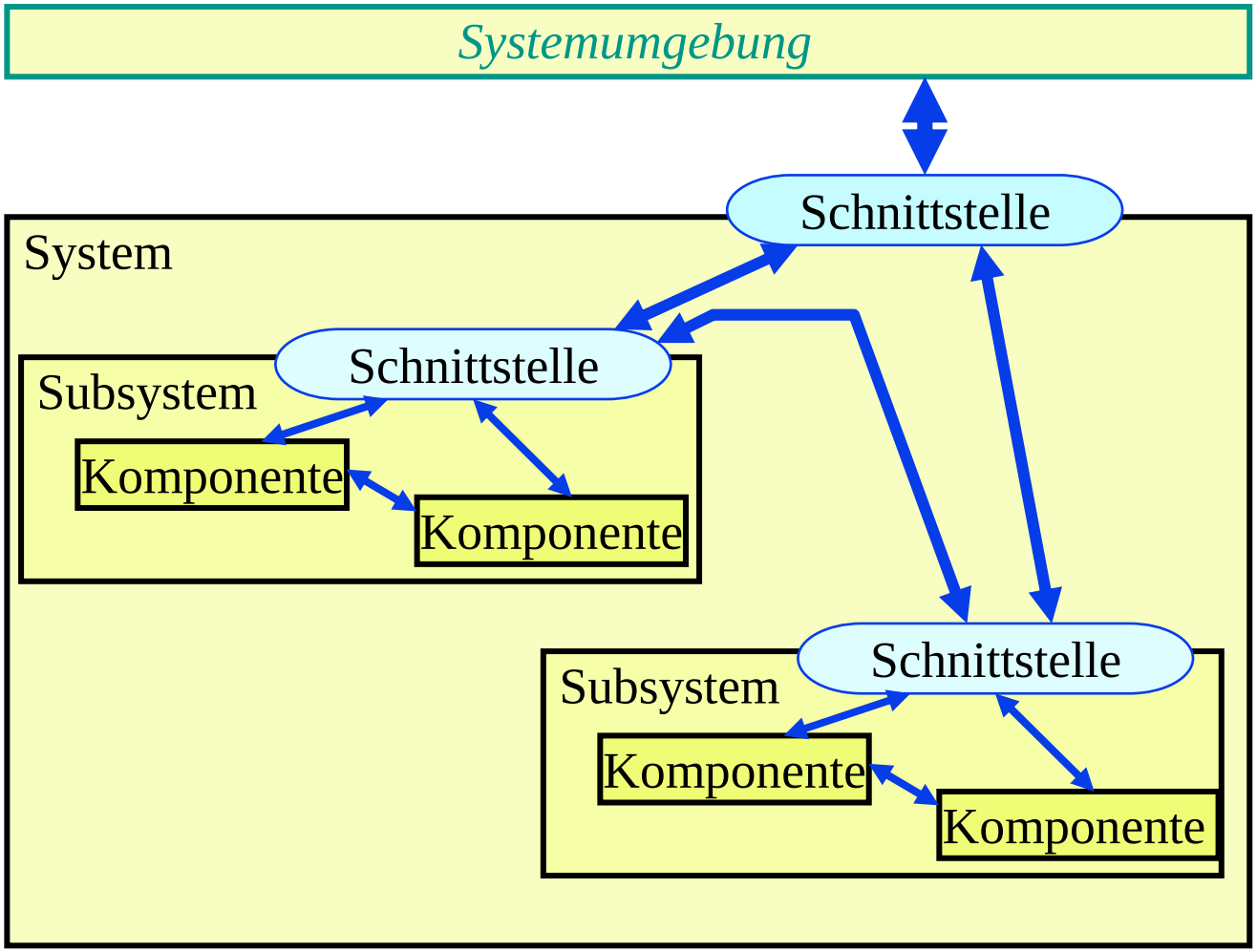
  Ein Modell entspricht nicht unbedingt in allen relevanten Belangen der Realität !

F1: Eigenschaften: Safety und Liveness

- **Safety**
  „Was darf geschehen?“, „Was darf auf keinen Fall geschehen?“
  Spielraum, des Systemverhaltens
  - Beispiele:
    „Die Etagentür des Aufzugs darf sich nur öffnen,
    wenn sich die Kabine auch auf Etagenhöhe befindet!“
    „Der Behälterdruck darf den Grenzwert nicht überschreiten!“
    „Die Geschwindigkeit darf erhöht werden, wenn der Bremsweg eingehalten werden kann.“
- **Liveness**
  „Was soll geschehen?“, „Worauf wollen wir nicht unendlich lang warten?“
  Gewünschte Fortschritte des Verhaltens
  - Beispiele:
    „Wenn der Aufzug-Rufknopf gedrückt wird, soll die Kabine in der Etage ankommen!“
    „Wenn der Auftrag eintrifft, soll er bearbeitet werden!“
- Problem
  Vermeide Widersprüche zwischen Safety- und Liveness-Anforderungen
  - Beispiel:
    „Wenn der Aufzug-Rufknopf gedrückt wird, soll sich die Etagentür nach endlicher Zeit öffnen!“ ABER
    „Die Kabine befindet sich in der falschen Etage und der Kabinenantrieb ist defekt.“

F1: Modelle ereignisdiskreter Systeme

- Komponenten haben Verhalten
  - interne Aktionen
  - externe Aktionen
- Komponenten sind gekoppelt
  - externe Aktionen → Interaktionen
- Modelle für
  - Komponenten-verhalten
  - Kopplung / Verkabelung
  - Aggregation / Systembildung / Komposition / Kapselung

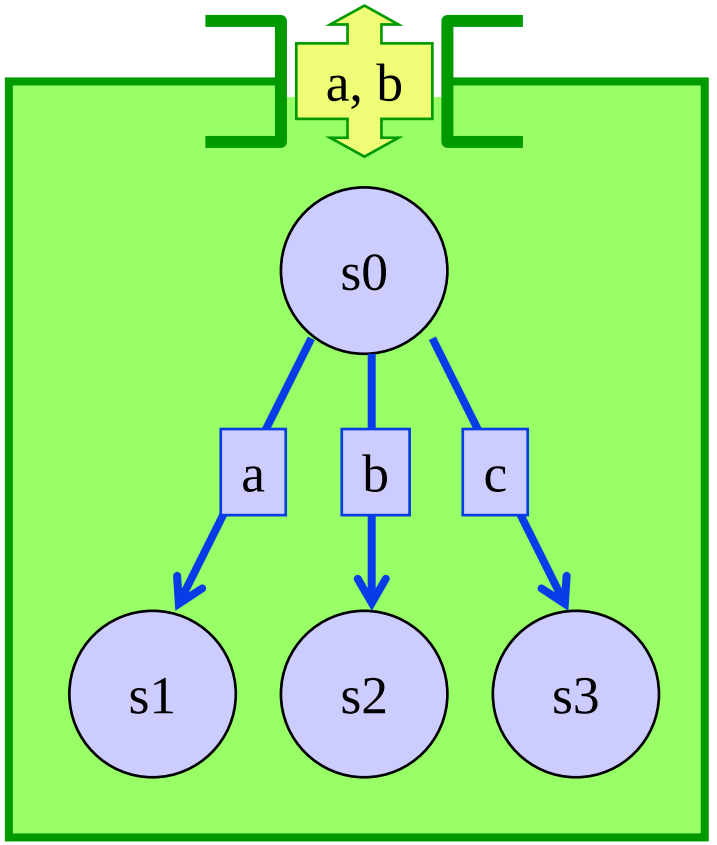


F1: Modelle ereignisdiskreter Systeme: Verhalten

- Verhalten
  - Serie von Verhaltensschritten
  - Schritt:
    Zustand mit Bereitschaft zu
    bestimmten Aktionen
    Aktion
    Folgezustand mit Folgeverhalten

- Interne Aktionen, Interne Ereignisse,
  Spontane Ereignisse

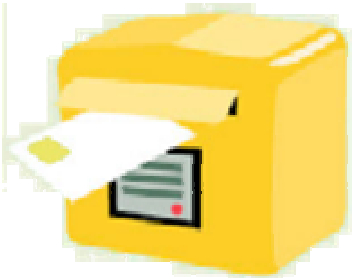
- Externe Ereignisse, Stimuli, Reaktionen,
  mehrseitig wirkende Interaktionen



F1: Modelle ereignisdiskreter Systeme: Interaktionen

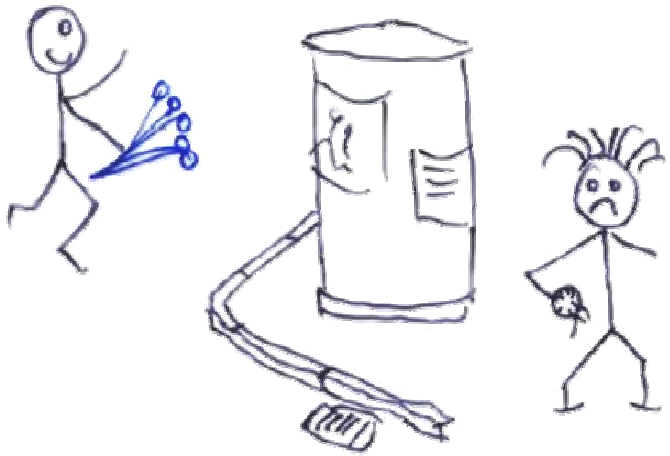
- Physikalische Interaktionen
  - nicht rückwirkungsfrei, jede Seite beeinflusst die andere

- Logische Interaktionen
  - oft rückwirkungsfrei, Initiator und Antworter, Sender und Empfänger
  - es gibt aber auch symmetrische Interaktionskonzepte
    - » zeitliches Rendezvous
    - » inhaltliche Übereinkunft / Abstimmung

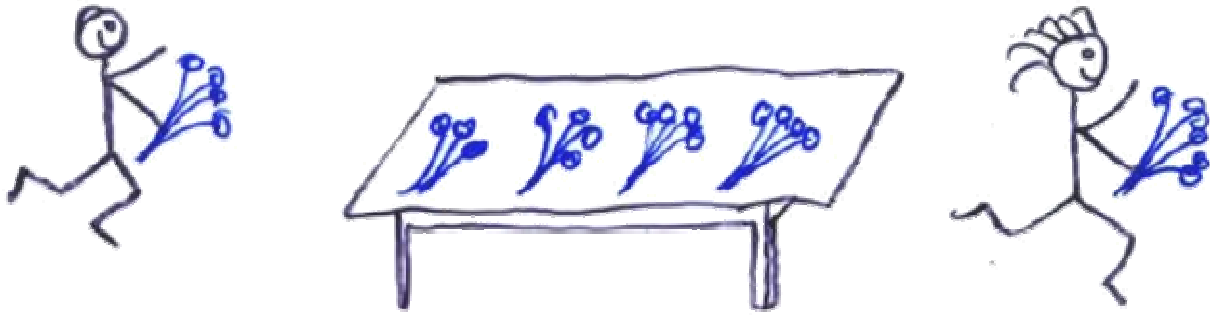


F1: Kopplung

- Rendezvous-Kopplung (nicht puffernd, „synchron“)

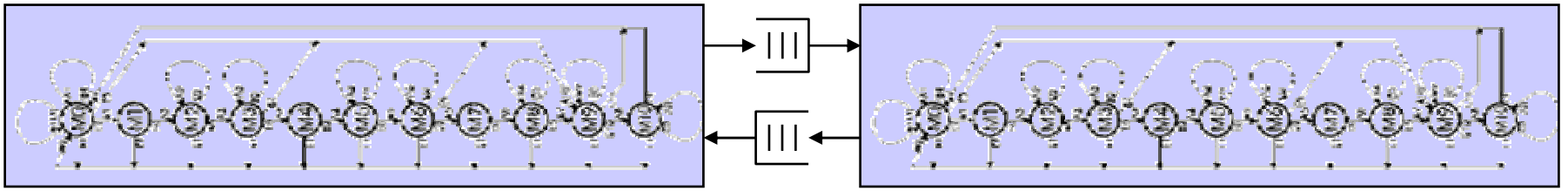


- Kanal-Kopplung (puffernd, „asynchron“)

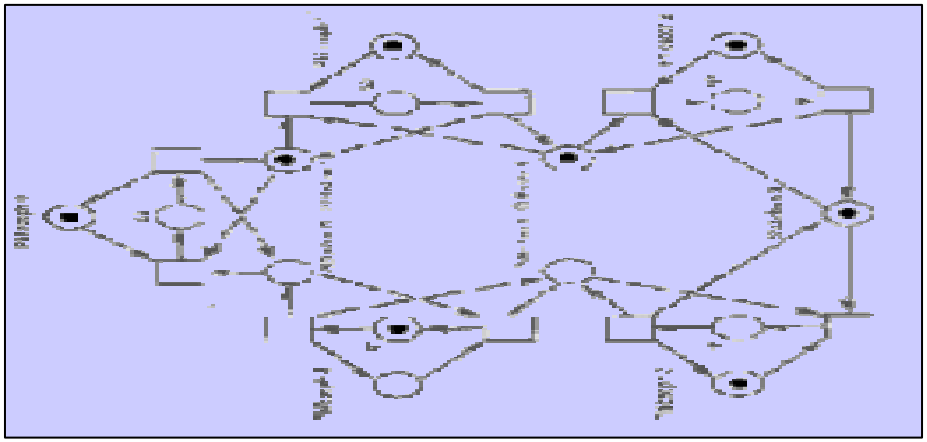


F1: Modelle ereignisdiskreter Systeme – Grobe Einteilung

- Komponenten-konzentriert
  komplexe Komponenten, einfache Kopplungen

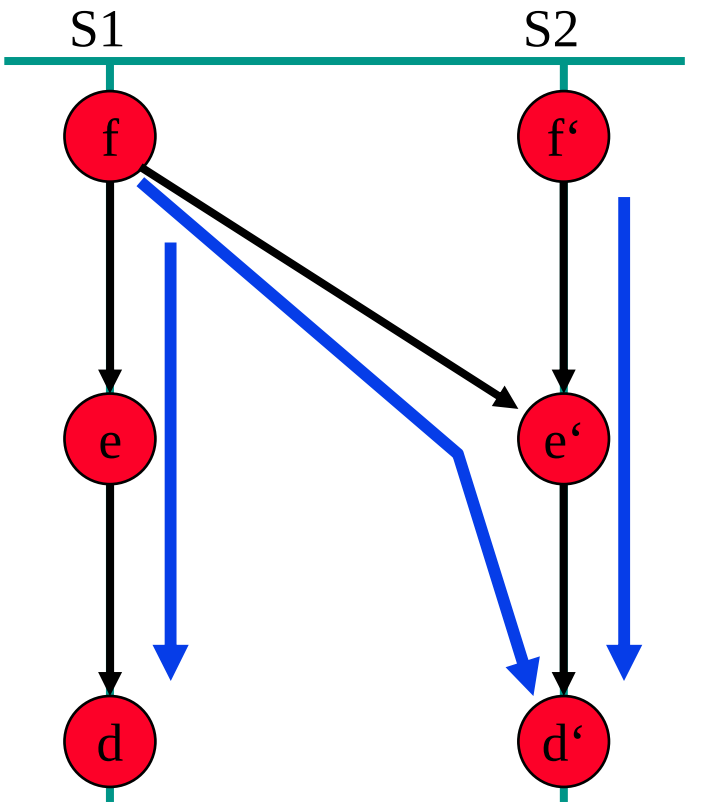


- Kopplungskonzentriert
  einfache Komponenten, komplexe Kopplungen



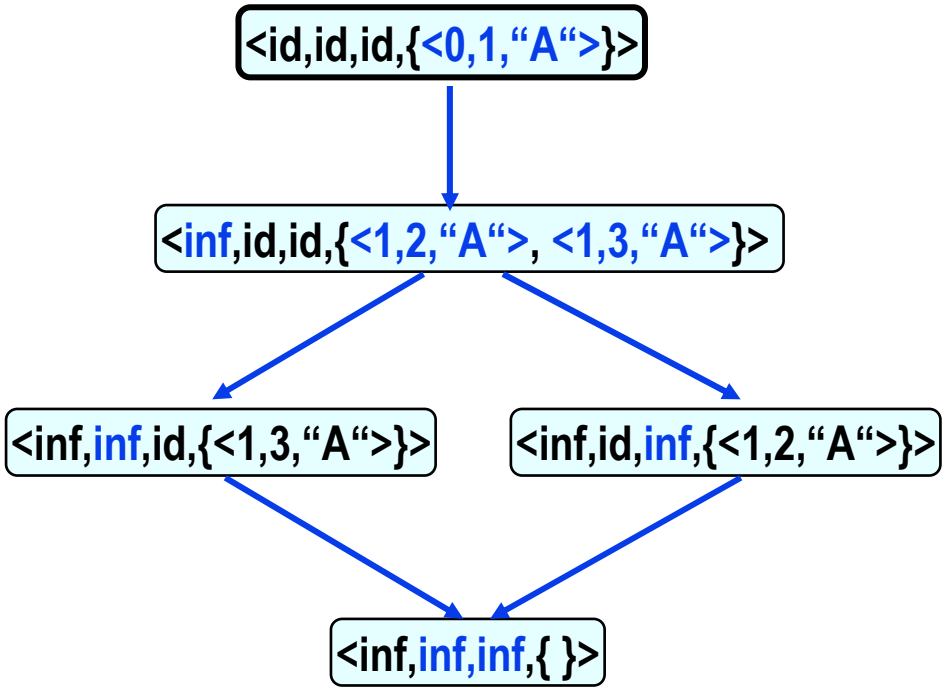
F1: Modelle ereignisdiskreter Systeme – Nebenläufigkeit

- „True Concurrency“ Partialordnungsmodelle



Kausalitätsstruktur

- „Interleaving“ Totalordnungsmodelle



Erreichbarkeitsgraph

F2: Erweiterter Mealy-Automat

- Mealy-Automat
- Unvollständiger und nichtdeterministischer Mealy-Automat
- Erweiterter Mealy-Automat
- Kanalkopplung

Literatur

- Hartmut König: „Protocol Engineering: Prinzip, Beschreibung und Entwicklung von Kommunikationsprodokollen“, Vieweg+Teubner Verlag, 2003

Standards

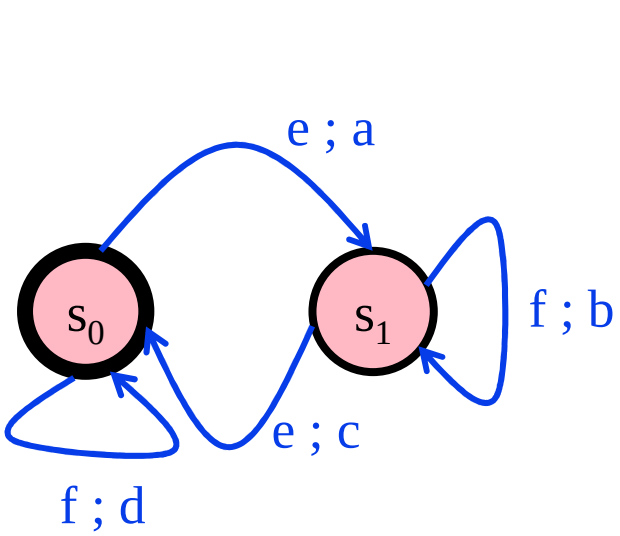
- ISO: ESTELLE (extended state transition language), ISO Internationaler Standard 9074
- ITU/CCITT: SDL (Specification and Description Language), ITU-T/CCITT-Recommendation Z.100

G. H. Mealy: A method for synthesizing sequential circuits.
The *Bell System Technical* Journal, Vol. 34, pp. *1045-1079, 1955.*

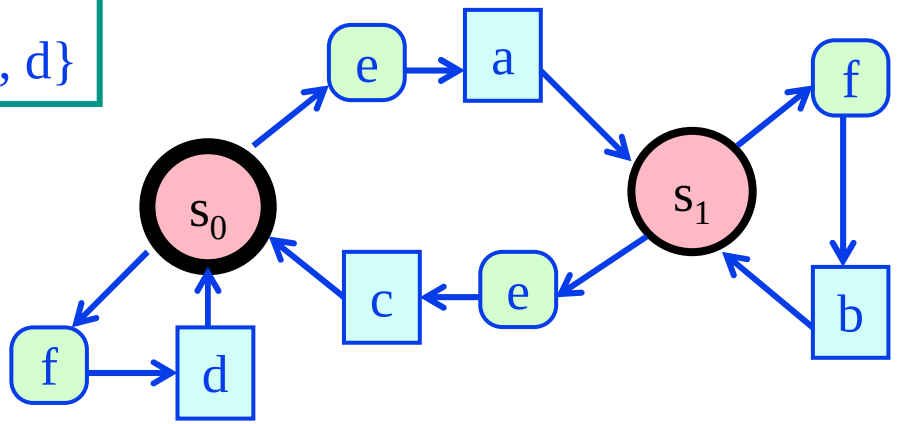
F2: Finite State Machines – Mealy-Automat

Mealy-Automat $< S, E, A, \delta, \lambda, s_0 >$

- S endliche Menge von Zuständen
- E endliche Menge von Eingabezeichen
- A endliche Menge von Ausgabezeichen
- δ Zustandsübergangsfunktion, $\delta : S \times E \rightarrow S$
- λ Ausgabefunktion, $\lambda : S \times E \rightarrow A$
- s_0 Startzustand, $s_0 \in S$



$S = \{s_0, s_1\}$
$E = \{e, f\}$
$A = \{a, b, c, d\}$



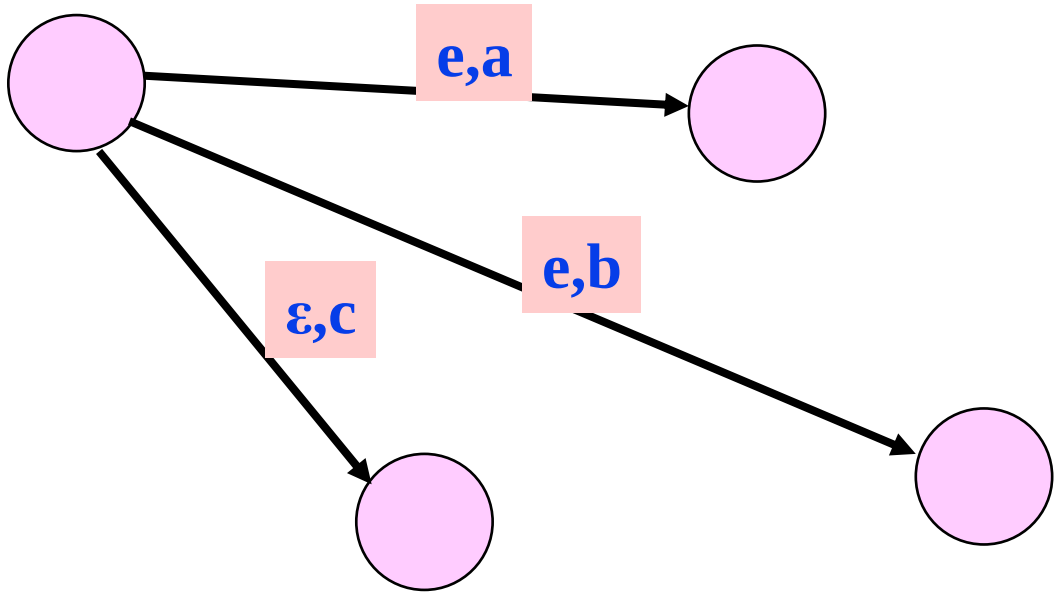
F2: Unvollständiger und nichtdeterministischer Mealy-Automat

Unvollständigkeit	*Nicht in jedem Zustand ist für alle Eingaben eine Transition vorhanden*
Nichtdeterminismus	*Es gibt u.U. pro Momentanzustand-Eingabe-Kombination mehr als eine Transition*
Spontane Transitionen	*Es gibt u.U. Transitionen ohne Eingabe*

keine Transition für Eingabe **f**,

2 Transitionen für Eingabe **e**,

Transition mit leerer Eingabe **ε**



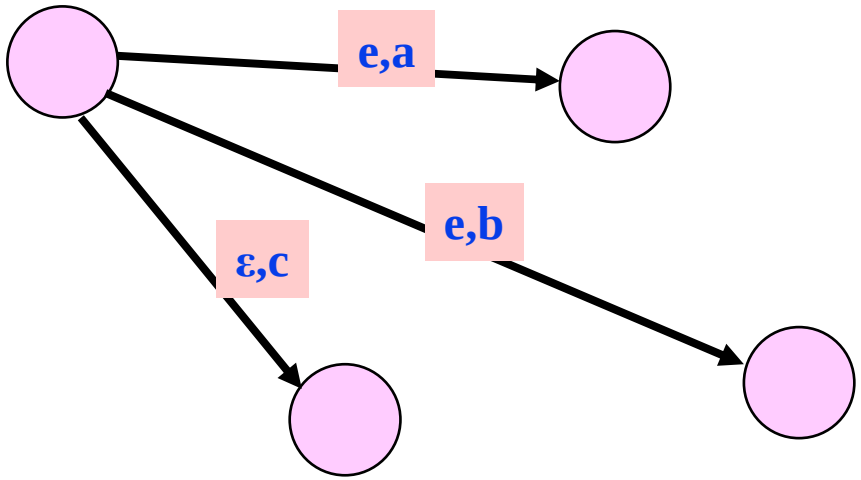
F2: Unvollständiger und nichtdeterministischer Mealy-Automat

- Nichtdeterministischer, unvollständiger Mealy-Automat $< S, E, A, \delta, \lambda, s_0 >$
- S endliche Menge von Zuständen
- E endliche Menge von Eingabezeichen
- A endliche Menge von Ausgabezeichen
- δ Zustandsübergangs**relation**, $\delta \subset (S \times E) \times S$
- λ Ausgabe**relation**, $\lambda \subset (S \times E) \times A$
- s_0 Startzustand, $s_0 \in S$

keine Transition für Eingabe **f**,

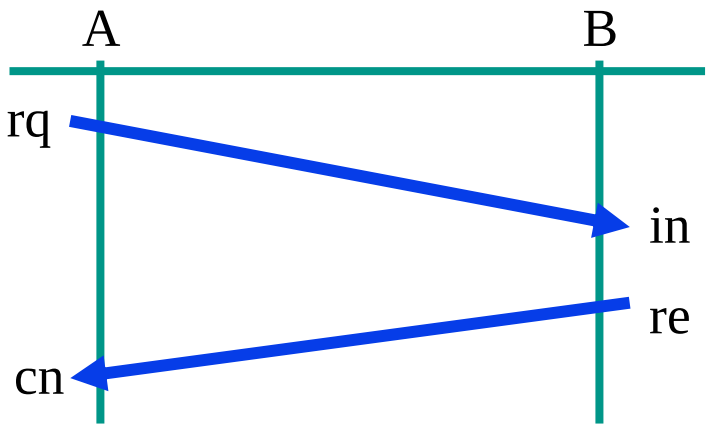
2 Transitionen für Eingabe **e**,

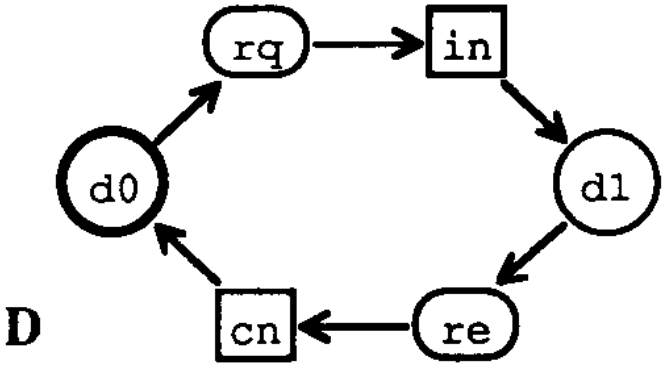
Transition mit leerer Eingabe **ε**



F2: Beispiel – Dienst D

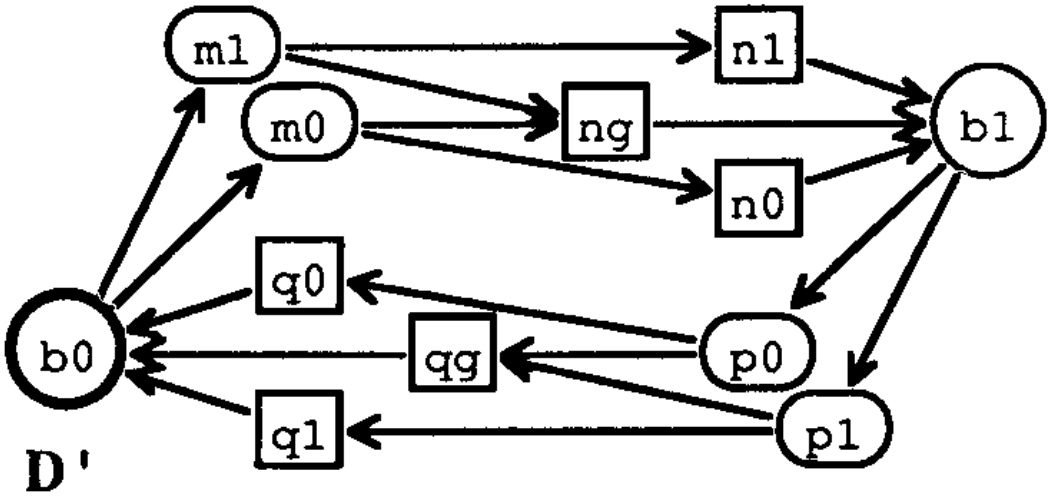
- Ein einfacher Kommunikationsdienst, der eine bestätigte Dienstleistung anbietet.
  - Initiator nur an Ort A
  - Responder nur an Ort B
  - keine Fehler

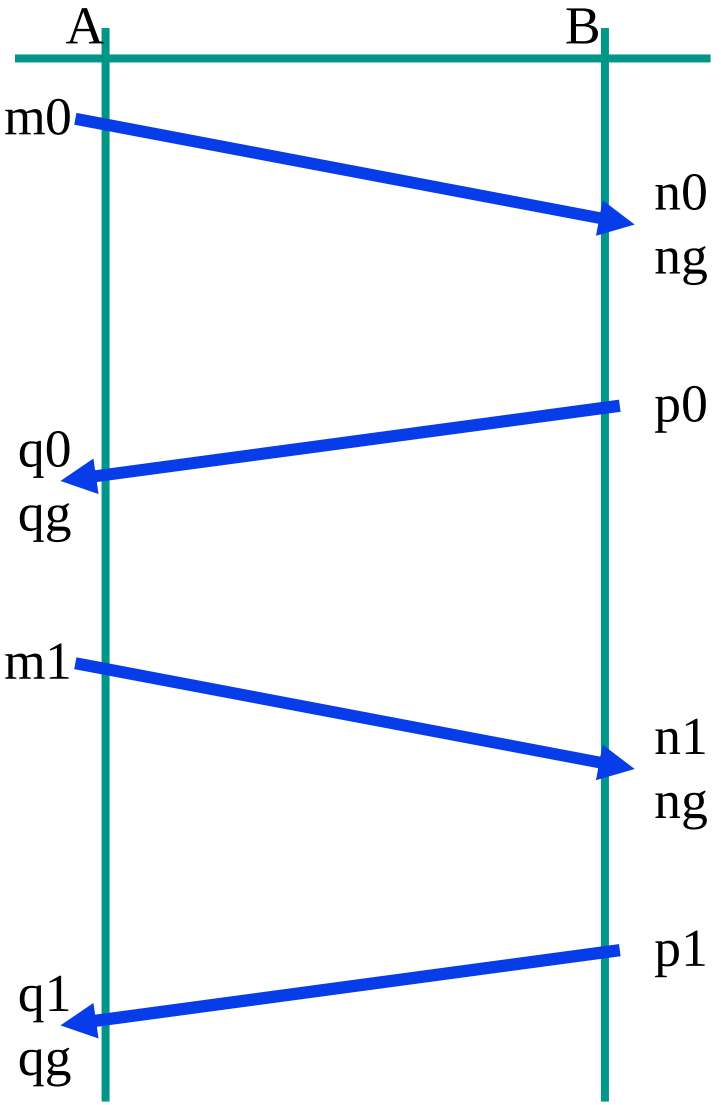




F2: Beispiel – Dienst D‘

Ein Halduplex-Kommunikationsdienst,
welcher von A nach B Nachrichen-PDUs m0, m1
und von B nach A Quittungs-PDUs p0, p1
überträgt.
Die PDUs können verfälscht werden.
Die Verfälschungen werden erkannt und angezeigt
(ng, qg)





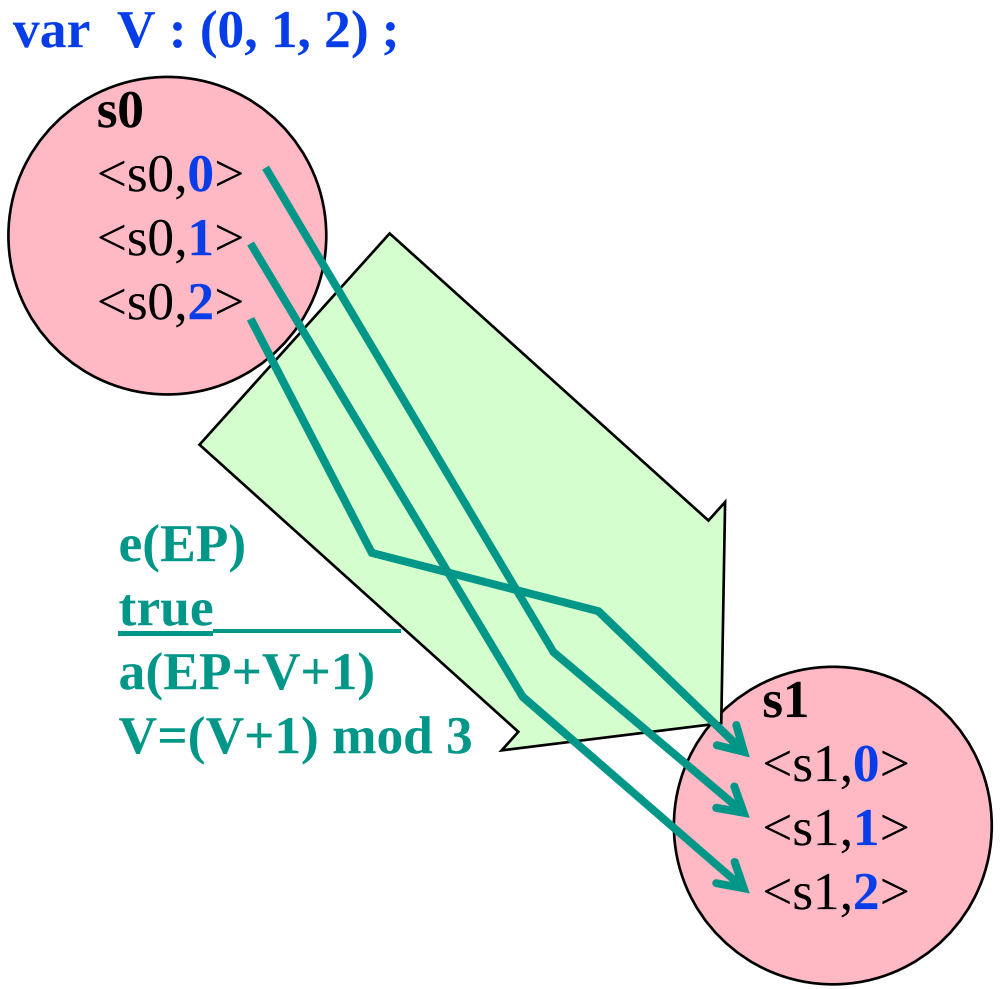
F2: Erweiterter Mealy-Automat

Problem:

Große Mengen von Zuständen, Eingaben, Ausgaben und Transitionen

Erweiterungen:

- Nebenzustandsraum: Variablen
- Eingaben und Ausgaben haben Datenparameter
- Variableninitialisierung
- Transitionsklausen

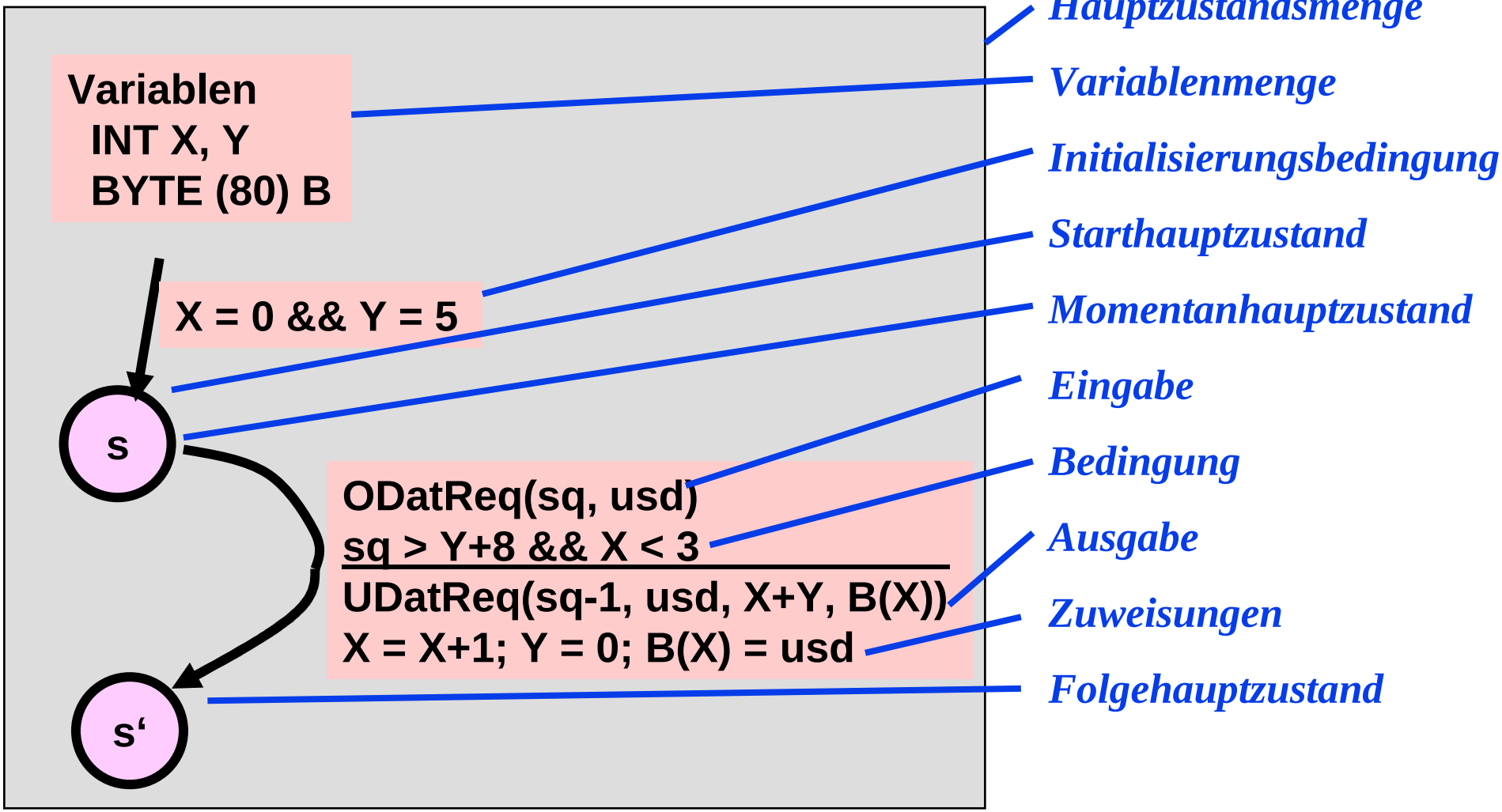


F2: Erweiterter Mealy-Automat

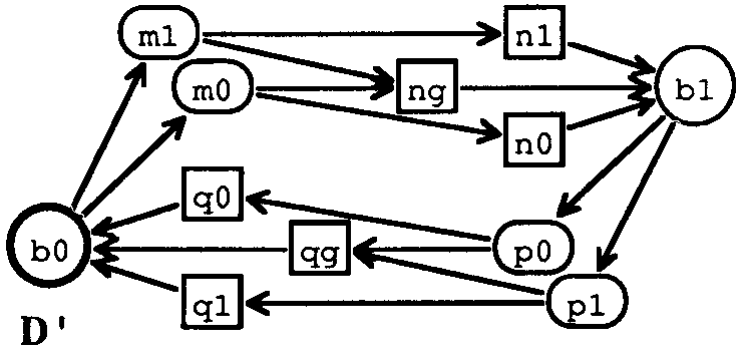
Erweiterter Mealy-Automat, definiert durch

- Menge von Variablen V_1, .. V_n mit Wertebereichen W_1, ..,W_n
- Menge von Eingaben E_1,.., E_m jeweils mit Parametern EP_{i1},.., EP_{im}
- Menge von Ausgaben A_1,.., A_p jeweils mit Parametern AP_{i1},.., AP_{ip}
- Menge von Hauptzuständen HS
- Ein ausgezeichneter Start-Hauptzustand hs_0
- Initialisierungsbedingung als boolescher Ausdruck über Variablen
- Menge von Transitionsklausen TK_1,.., TK_q,
  jeweils definierend eine Menge von Transitionen T_1, .., T_q
  - Momentanhauptzustand: $s \in HS$
  - Eingabe $e(w_1, w_2, ..)$: Term aus E_i über Eingabeparametern
  - Bedingung: Boolescher Ausdruck über Eingabeparametern und Variablen
  - Folgehauptzustand: $s' \in HS$
  - Ausgabe $a(u_1, u_2, ..)$: Term aus A_j über E_i-Eingabeparametern und Variablen
  - Variablenzuweisungen $V_k = aus_k$, Term über E_i-Eingabeparametern und Variablen

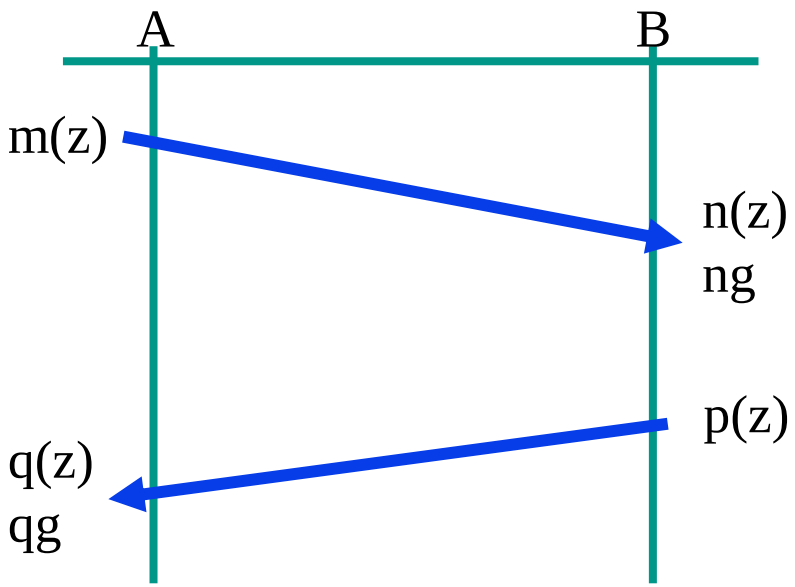
F2: Erweiterter Mealy-Automat

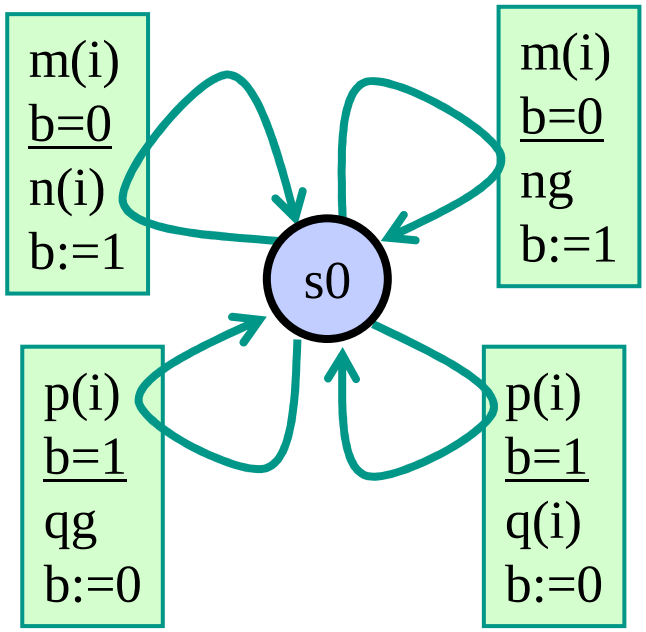


F2: Beispiel – Dienst D‘ als erweiterter Automat



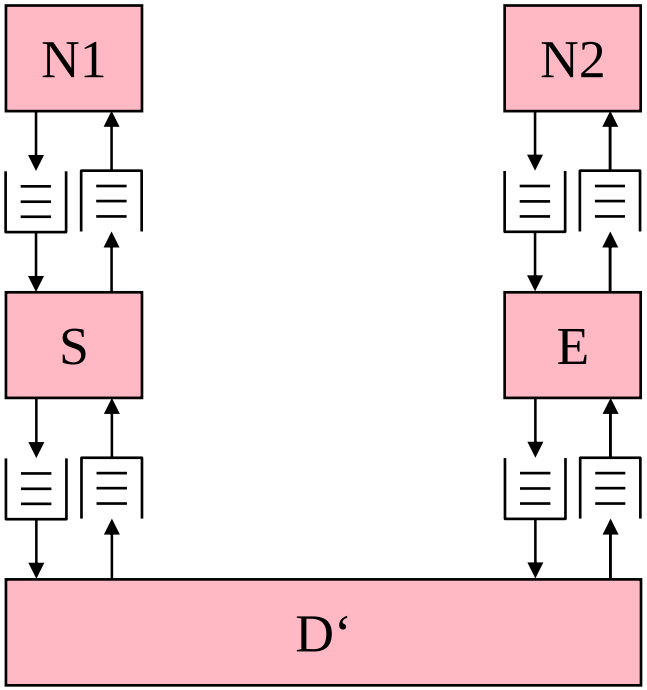
Eingaben: m (i : (0, 1)) ; p (i : (0,1)) ;
Ausgaben: n (i : (0, 1)) ; q (i : (0,1)) ;
ng ; qg ;
Hauptzustände: S={s0} ;
Variablen: b : (0, 1) ; init b := 0 ;

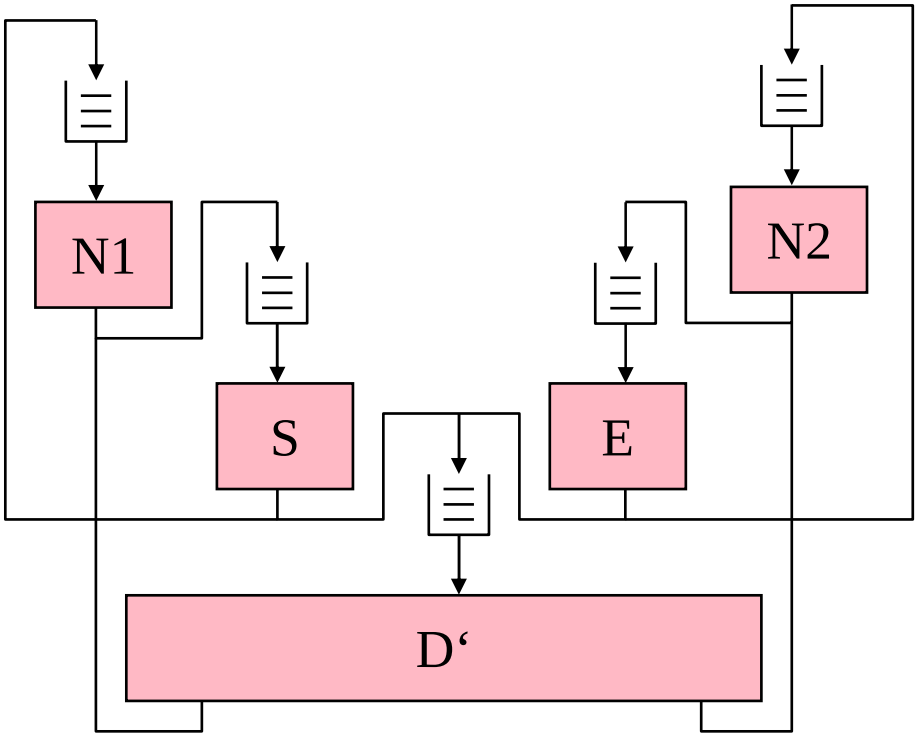




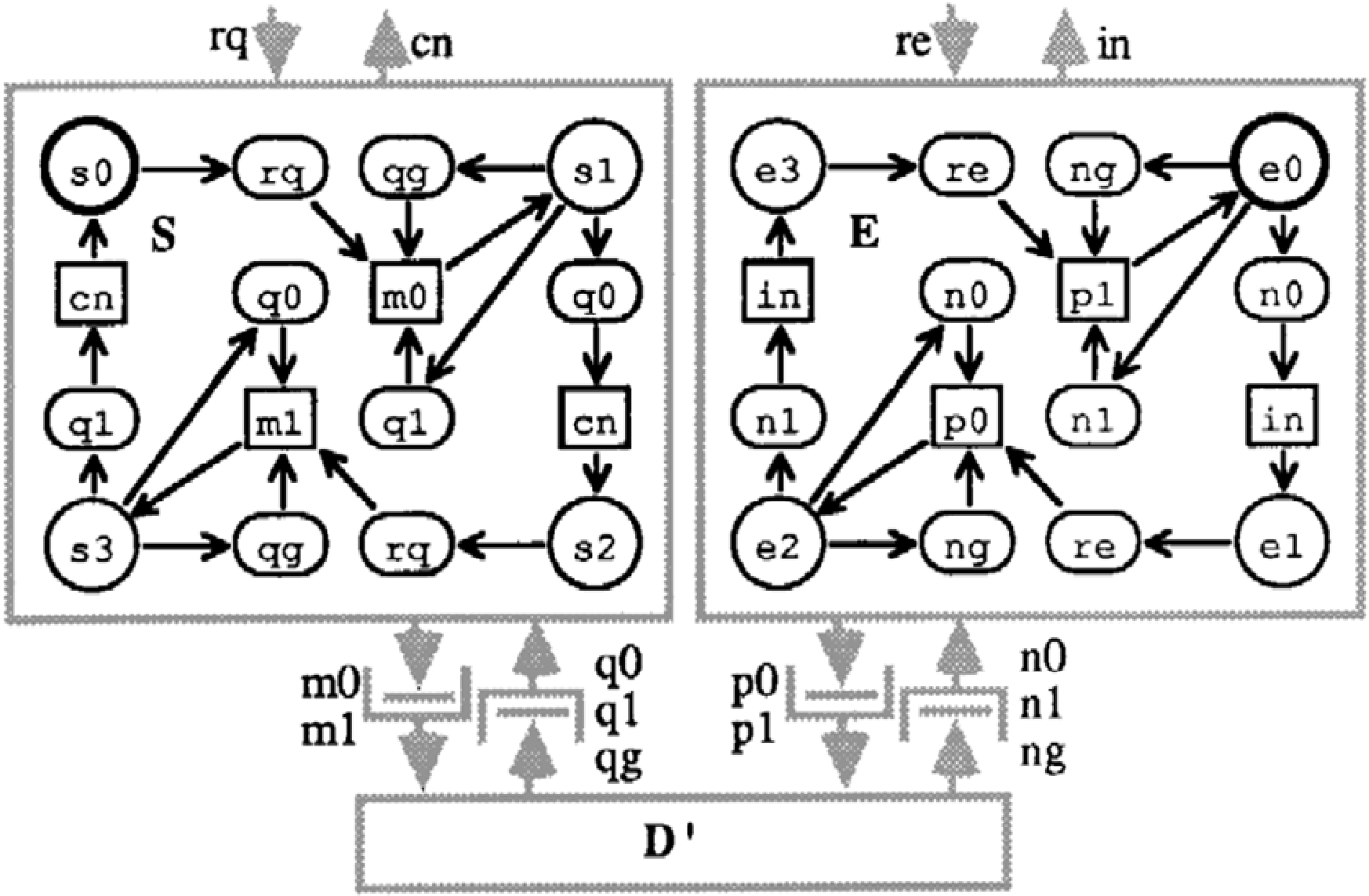
F2: Kanalkopplung

- Zur Systembildung werden Mealy-Automaten durch Puffer verbunden.
- Ein Puffer hat in der Regel FIFO-Eigenschaften (Queue).
- Varianten:
  - ein **Verbindungskanal** je gekoppeltem Automatenpaar und Richtung
  - eine **Mailbox** je Automat
  - frei definierbare Queues

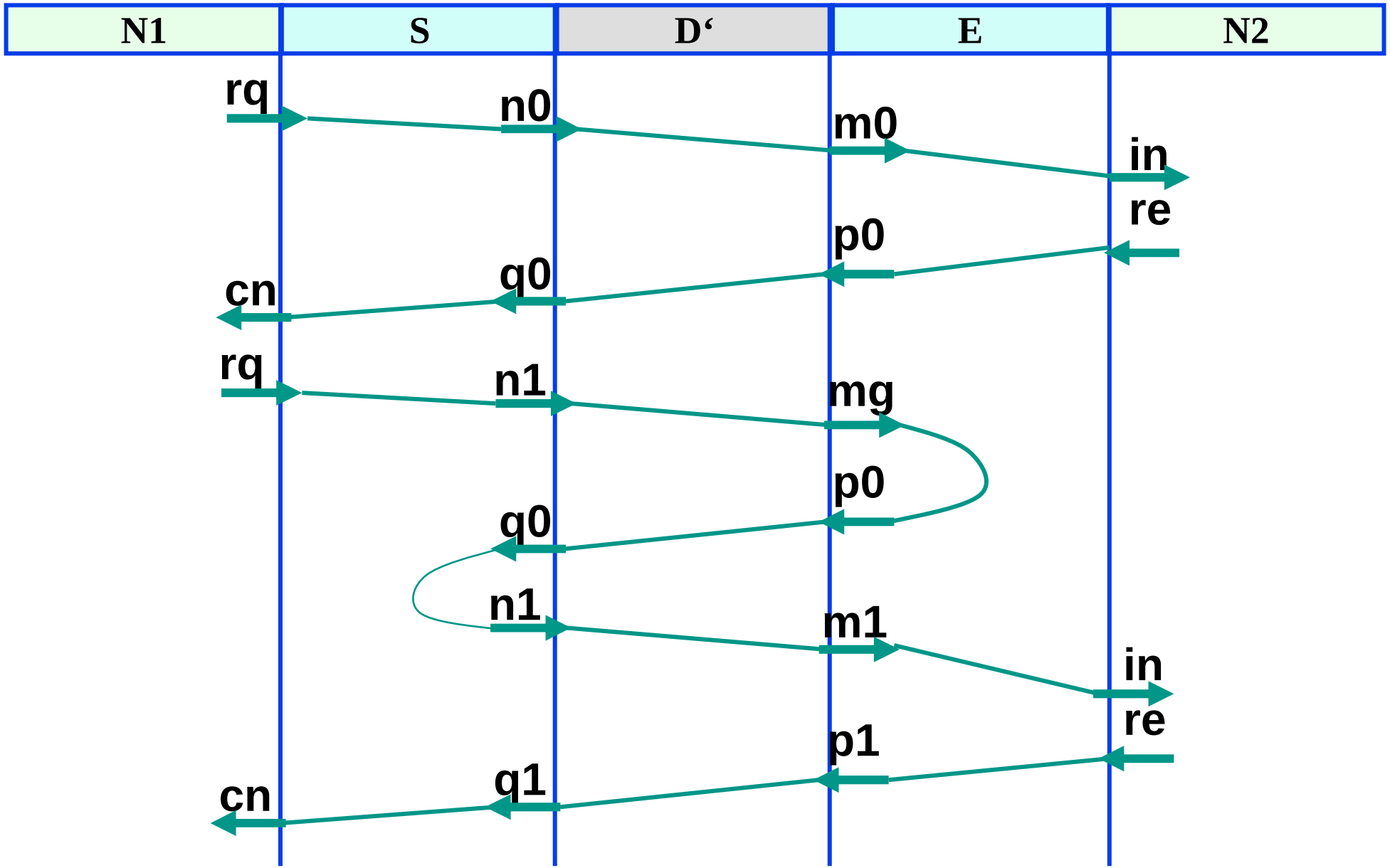




F2: Beispiel – Protokollsystem AB



F2: Beispiel – Protokollsystem AB



F3: Petri Netz

Nebenläufige, nichtdeterministische, räumlich verteilte, ereignisdiskrete Systeme

- *Fokus: Räumliche Struktur, kausale Zusammenhänge*
- *Viele Varianten, hier: Stellen-Transitionsnetze*

- Stellen-Transitionsnetz
- Schaltregel
- Schritte
- Eigenschaften
- Stellen- und Transitions-Invarianten

- W. Reisig: „Petrinetze: Eine Einführung“, Springer-Verlag, Berlin, Heidelberg, 1986
- Carl Adam Petri: „Introduction to general net theory“, in: W. Brauer (ed.), Net Theory and Applications, Lecture Notes in Computer Science 84, pp. 1-19, 1980.

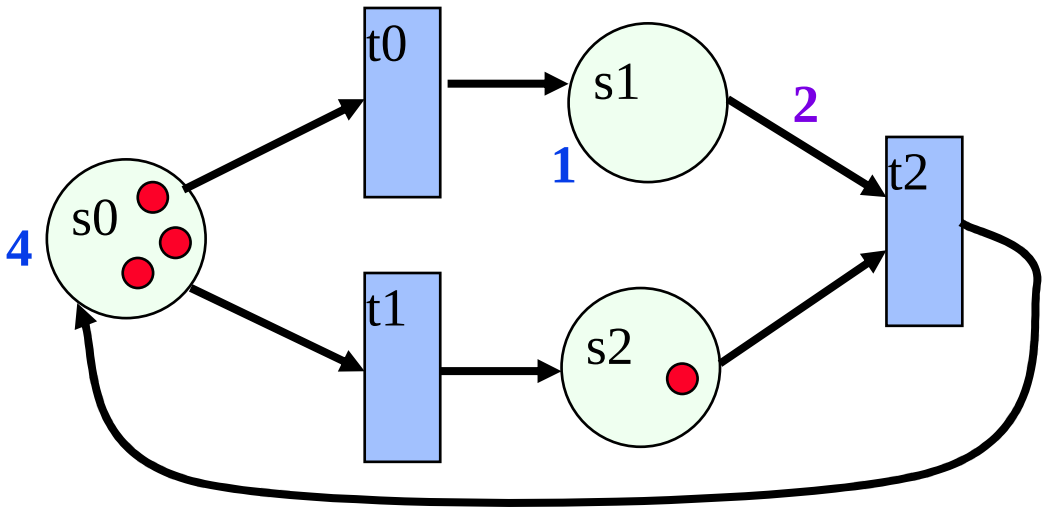
Carl Adam Petri: „Kommunikation mit Automaten“, Dissertation, Schriften des Rheinisch-Westfälischen Institutes für Instrumentelle Mathematik an der Universität Bonn Nr. 2, 1962

F3: Stellen-Transitionsnetz

Stellen-Transitions-Netz $< S, T, F, K, W, m_0 >$

- S — Menge von Stellen (auch Plätze)
- T — Menge von Transitionen, $S \cap T = \{\}$
- F — Flussrelation, Menge von Kanten (zweigeteilt): $F \subset (S \times T) \cup (T \times S)$
- K — Stellenkapazität, $K : S \rightarrow IN \cup \{\infty\}$
- W — Kantengewicht, $W : F \rightarrow IN$
- **m_0** — Startmarkierung, $m_0 : S \rightarrow IN$

} Netzgraph <S, T, F>



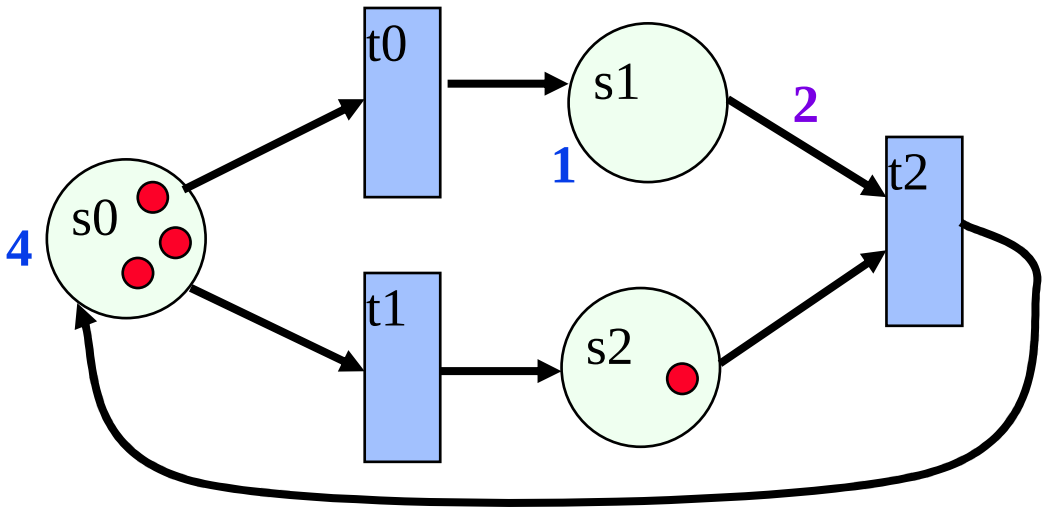
S= {s0, s1, s2}
T= {t0, t1, t2}
F= {<s0,t0>, <s0,t1>, <t0, s1>, <t1, s2>, <s1, T2>, <s2, t2>, <t2, S0>}

K= {s0→4, s1→1, s2→oo}
W= {<s1,t2>→2, sonst <*,*>→1}
m0= { s0→3, s1→0, s2→1 }

F3: Stellen-Transitionsnetz

Im Stellen-Transitions-Netz $< S, T, F, K, W, m_0 >$:

- Vorbereich ●t einer Transition t:
  $\{ s: <s, t> \in F \}$
- Nachbereich t● einer Transition t
  $\{ s: <t, s> \in F \}$



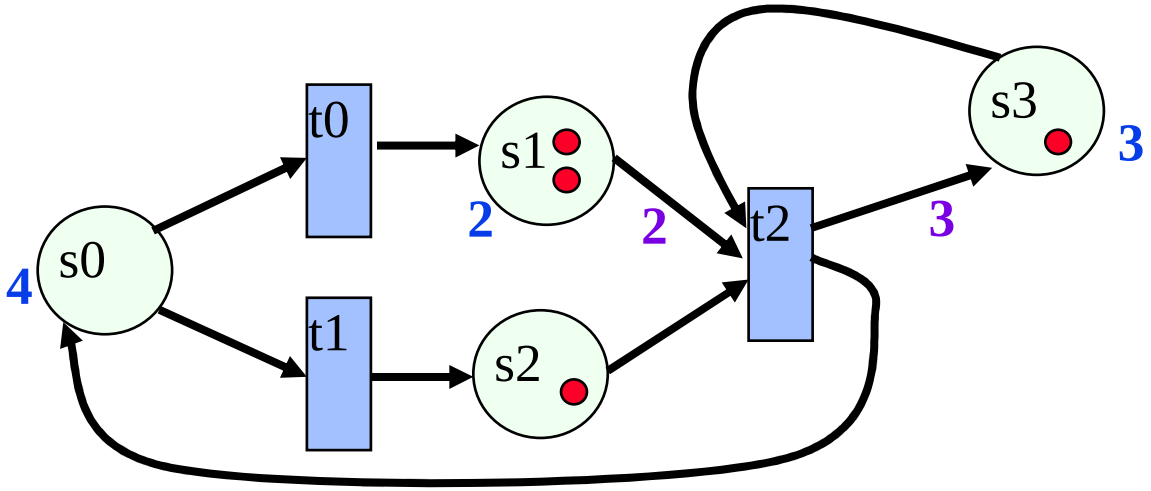
●t0= {s0}
t0●= {s1}

●t1= {s0}
t1●= {s2}

●t2= {s1, s2}
t2●= {s0}

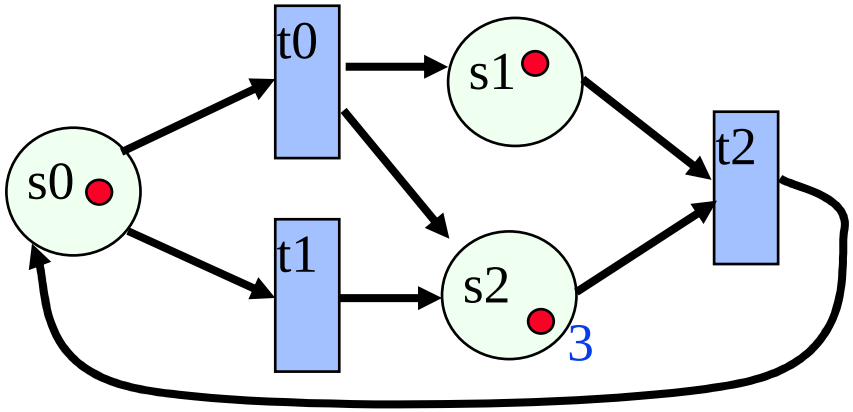
F3: Petri Netz - Dynamik – Schaltregeln

- Transition t heißt aktiviert oder schaltbereit:
  - $\forall s \in \bullet t: m(s) \geq W(<s,t>)$! Eingänge verfügbar
  - $\forall s \in t\bullet \setminus \bullet t: K(s) \geq m(s)+W(<t,s>)$! Platz für Ausgänge
  - $\forall s \in t\bullet \cap \bullet t: K(s) \geq m(s)-W(<s,t>)+W(<t,s>)$! Schleifenbilanz passt
- Schalten einer Transition t
  (m steht für momentane Markierung, m‘ für die Folgemarkierung)
  - m‘(s) = m(s)-W(<s,t>) falls s nur im Vorbereich von t
  - m‘(s) = m(s)+W(<t,s>) falls s nur im Nachbereich von t
  - m‘(s) = m(s)-W(<s,t>)+W(<t,s>) falls s im Vor- und im Nachbereich (Schleife)



F3: Petri Netz – Dynamik: Systemablauf

- Ein Schritt
  - Momentanmarkierung liegt vor
    - Es ist eine bestimmte Menge von Transitionen schaltbereit
      - Eine Teilmenge dieser schaltet (***Konflikte*** und ***Kontakte*** beachten)
        - Dadurch entsteht Folgemarkierung



- Maximale Schritte
  - Alle schaltbereiten Transitionen schalten
    (Konflikte: Maximale Teilmenge der schaltbereiten Transitionen)

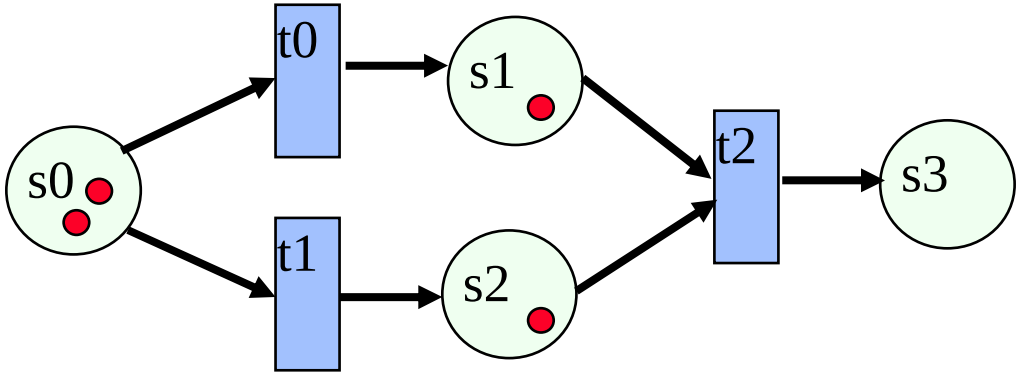
- Einer-Schritte
  - Jeweils nur eine schaltbereite Transition schaltet

- t0, t1, t2 sind alle schaltbereit

*- t0 und t1 sind im **Konflikt** (wg. s0)*
*-t0 und t1 sind im **Kontakt** (wg. s2)*

F3: Petri Netz – Dynamik: Interleaving

- Betrachtet wird total geordnete Folge von Ereignissen



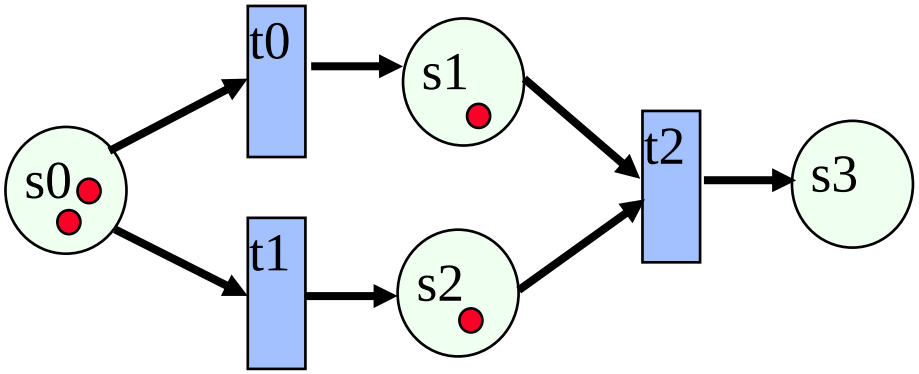
z.B.
<t0>, <t2>, <t1>, <t2> ODER
<t2>, <t1>, <t1> ODER
<t0>, <t0>, <t2> ODER
…

Nichtdeterministische alternative Systemabläufe

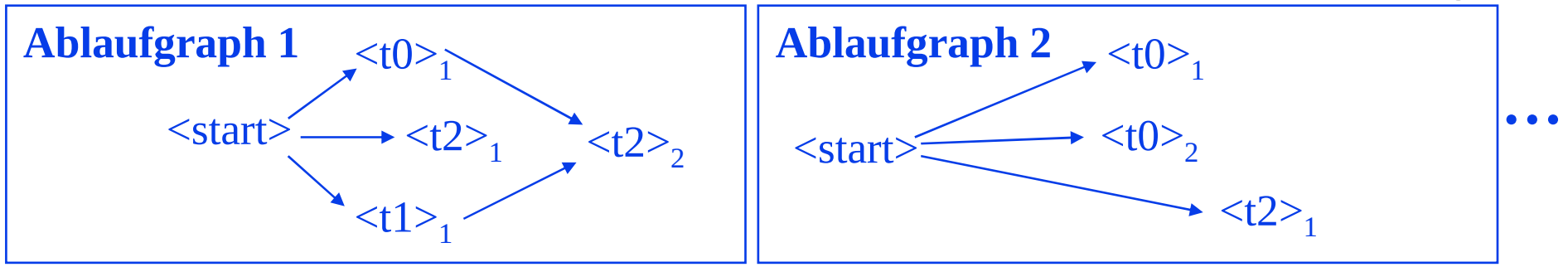
Nebenläufigkeit wird als nichtdeterministische Auswahl modelliert.

F3: Petri Netz – Dynamik: Kausalitätsstruktur

- Betrachtet werden kausale Abhängigkeiten zwischen Ereignissen



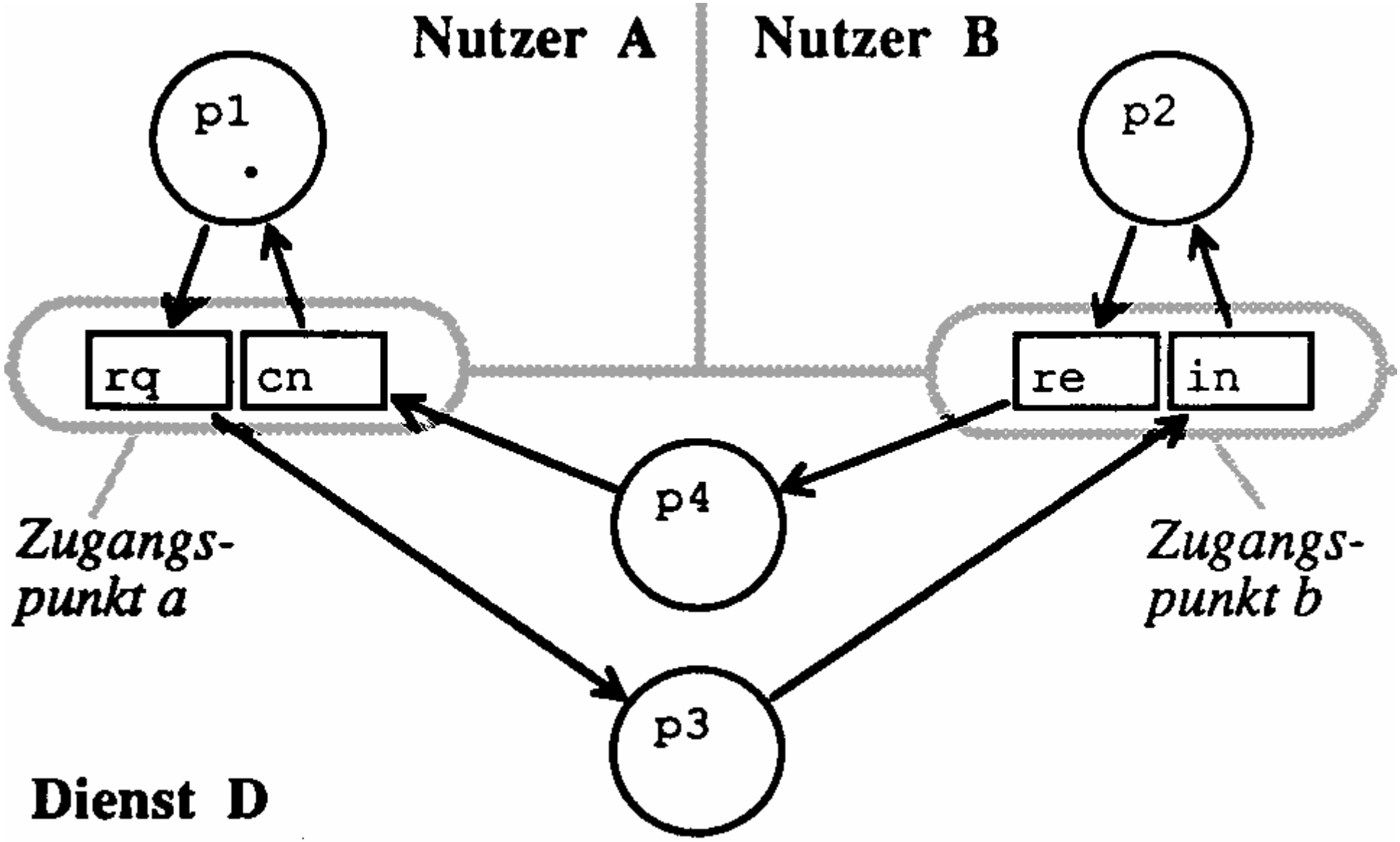
ODER ODER



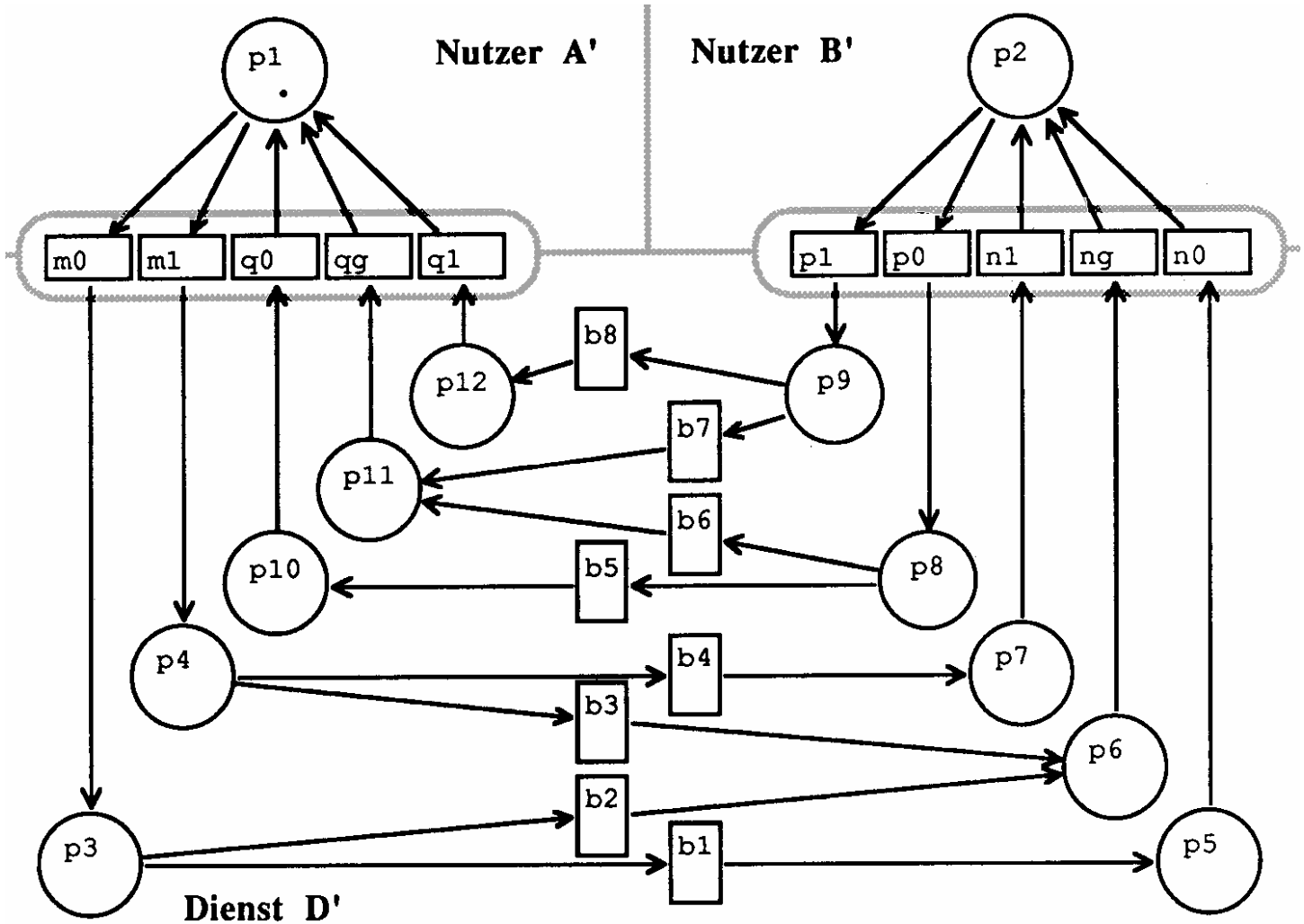
Nebenläufigkeit wird als Kausalitätsstruktur / Halbordnung modelliert.
Wechselweise kausal unabhängige Ereignisse bleiben als solche erkennbar.

Nichtdeterminismus gibt es dennoch: Unterschiedliche Ablaufgraphen sind möglich.

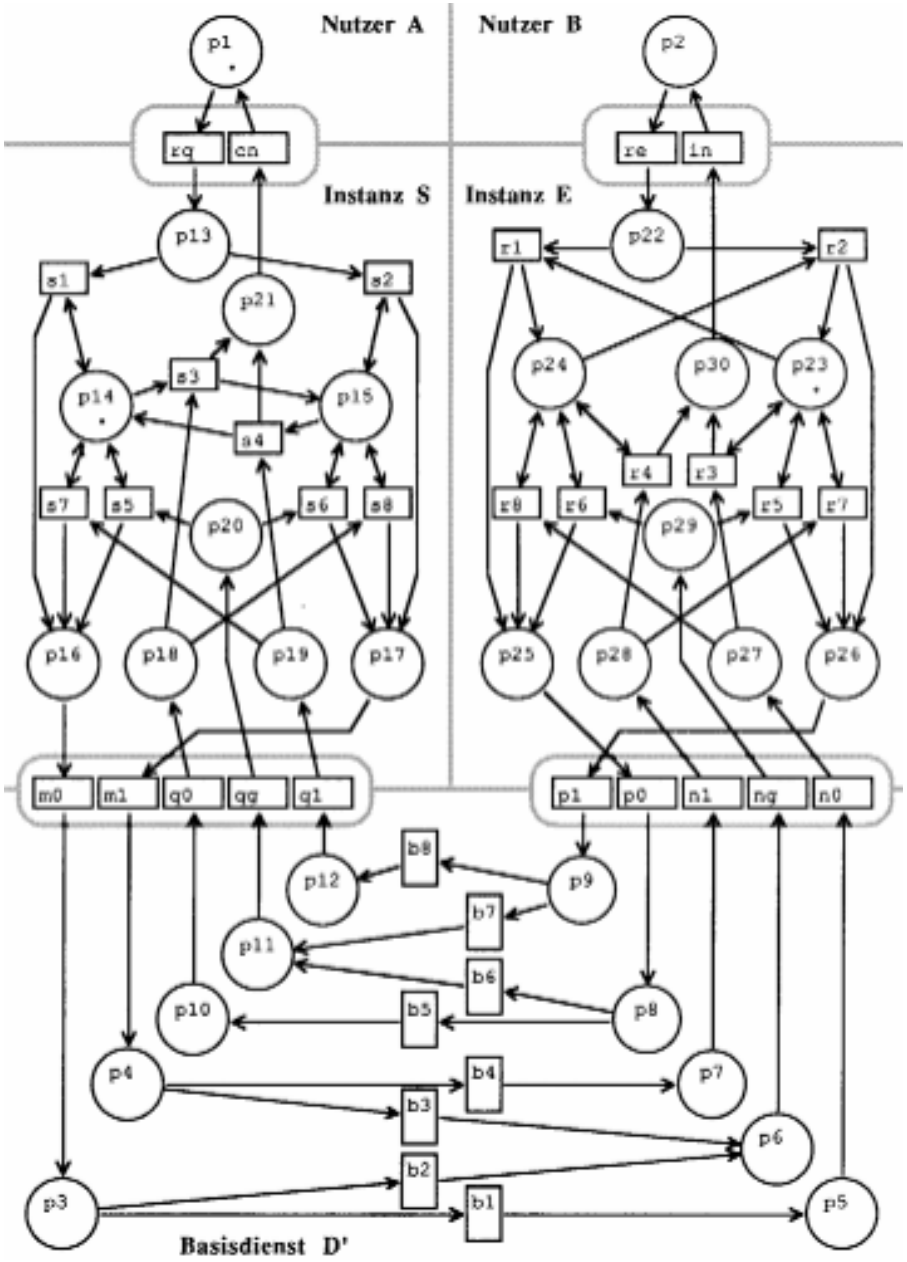
F3: Beispiel – Dienst D



F3: Beispiel – Dienst D‘



F3: Beispiel – Protokollsystem AB



Anmerkung
Man kann in einem Netz Zustandsautomaten darstellen, wenn man sich nur eine einzige Marke bewegen lässt (z.B. die Instanzen S und E)

F3: Petri Netz - Dynamik – Begriffe

- Eine Markierung m ist von einer Anfangsmarkierung m_0 aus **erreichbar**, wenn es einen mögliche Schrittfolge gibt, die zu dieser Markierung führt.
- Lebendigkeit einer Transition t bei Anfangsmarkierung m_0, t heißt
  - **tot**, falls t in allen erreichbaren Markierungen nicht schaltbereit ist
  - **aktivierbar**, falls es mindestens eine erreichbare Markierung gibt, in der t schaltbereit ist,
  - **lebendig**, falls es zu jeder erreichbaren Markierung eine davon aus erreichbare Markierung gibt, in der t schaltbereit ist.
- Lebendigkeit eines Petri Netzes mit Anfangsmarkierung m0, das Netz heißt
  - **tot**, falls alle seine Transitionen tot sind,
  - **verklemmungsfrei**, falls es keine erreichbare Markierung gibt, von der aus alle Transitionen tot sind,
  - **lebendig**, falls alle Transitionen lebendig sind.
- Ein Petri-Netz heißt **beschränkt** mit Schranke *b*, falls in allen erreichbaren Markierungen in allen Stellen jeweils nicht mehr als *b* Marken liegen.
- Ein Petri-Netz heißt **sicher**, falls es beschränkt mit Schranke 1 ist.

F3: Petri Netz - Dynamik – Invarianten

- Stelleninvariante
  Aussage zur Belegung der Stellen, welche bei jeder erreichbaren Markierung zutrifft.
- Transitionsinvariante
  Aussage zur Häufigkeit des Schaltens von Transitionen, welche in jedem möglichen Ablauf zutrifft.

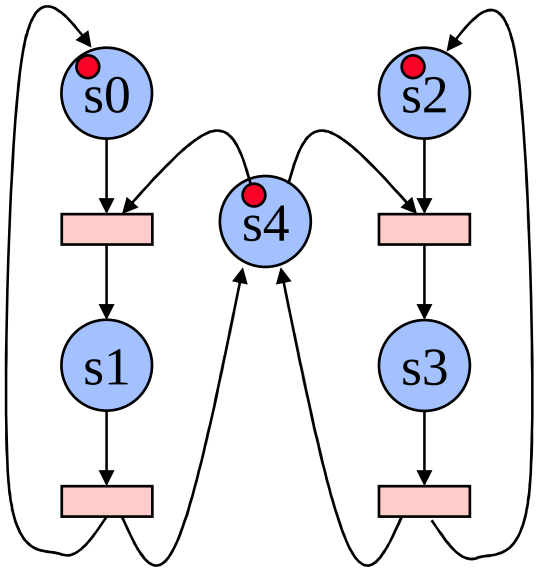
Relevante Stelleninvariante dieses Beispiels zum gegenseitigen Ausschluss:

#s1+#s3+#s4=1

(#si: Anzahl Marken in Stelle si)

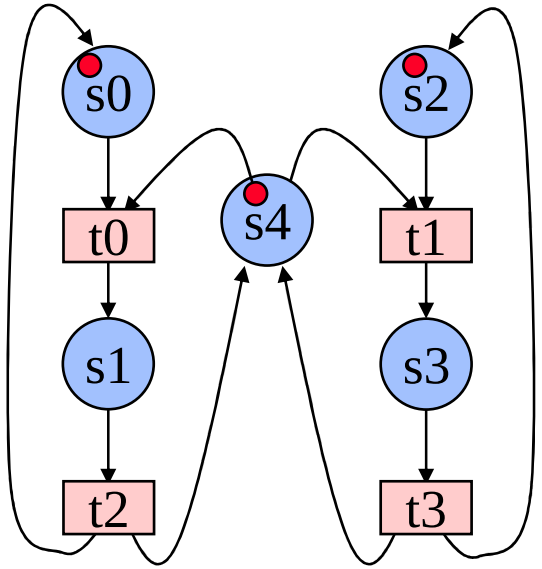
Eine andere Stelleninvariante dieses Netzes ist:

#s0 + 2*#s1 + #s2 + 2*#s3 +#s4 = 3



F3: Petri Netz - Dynamik – Invarianten

- Inzidenzmatrix C eines Stellen-Transitionsnetzes < S, T, F, K, W, m_0 >
  - $C = (c)_{ij}$ mit $i \in S, j \in T$
  - c_{ij} = (Summe der gewichteten Kanten von t_j nach s_i) – (Summe der gewichteten Kanten von s_i nach t_j)
    ! Markenveränderung für s_i, falls t_j schaltet
- Stelleninvariante v
  - $v \neq <0,0,0, ..>$, $v = <k>_i$ mit $i \in S$
  - Ganzzahlige Lösungen für: $v \bullet C = <0,0,0,..>$
    $\Leftrightarrow$ für alle erreichbaren Markierungen $m = <m_1, m_2, m_3, ..>$ gilt:
    $\sum_{i \in S} k_i * m_i$ = Konstante



#s1+#s3+#s4=1

s0	s1	s2	s3	s4
0	1	0	1	1

$\bullet$

	t0	t1	t2	t3
s0	-1	0	1	0
s1	1	0	-1	0
s2	0	-1	0	1
s3	0	1	0	-1
s4	-1	-1	1	1

= <0,0,0,0,0>

F3: Petri Netz – Weitere Netzformen

- Bedingungs-Ereignis-Netze
  - Platz hat höchstens 1 Marke entspechend zutreffender Bedingung
- Stellen-Transitions-Netze
  - Platz kann mehrere Marken aufnehmen
- Inhibitornetze
  - Inhibitor-Plätze

- Gefärbte Petri Netze
  - Marken gefärbt
- Prädikat-Transitionsnetze
  - Marken enthalten Daten, Transitionen sind bedingt

- Stochastische Petri Netze
- Zeitbehaftete Petri Netze

…

F4: Gefärbtes Transitionssystem (LTS)

- Labelled Transition System (LTS)
- Verhalten einer Komponente eines Systems
  - Komponente hat (internen) Zustand und es gibt Transitionen.
  - Komponente führt Aktionen aus:
    Transitionen werden mit Aktionen markiert (gefärbt).
  - Zur Kopplung werden Aktionen extern sichtbar und (zusammen mit Aktionen anderer Komponenten) Teil von Interaktionen
  - Interaktionen haben den Charakter von Joint Actions

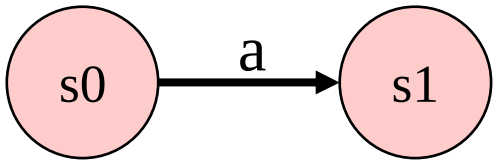
- Es ist bei LTS üblich, die Zustände in den Hintergrund treten zu lassen und sich auf die Abfolgen der Aktionen (z.B. auch nur der extern sichtbaren Aktionen) zu konzentrieren.
- Im Vergleich zum „Endlichen Automaten“ / „Endlichen Akzeptor“ sind
  - aufzählbar unendlich viele Zustände und Markierungen möglich
  - keine Akzeptorzustände definiert
  - in den meisten Varianten auch keine Startzustände definiert (potentiell Start in jedem Zustand)

Literatur
z.B.: R. de Nicola: “*Extensional equivalences for transition systems*”,
Acta Informatica,Vol. 24,2, pp. 211-237, 1987.

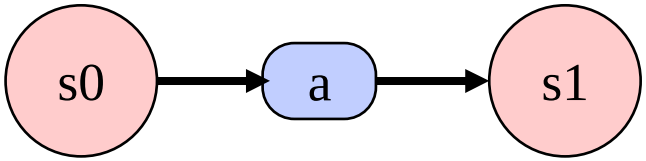
F4: Gefärbtes Transitionssystem (LTS)

- Labelled Transition System (LTS) < S, A, T (, S_0) >
  - S aufzählbare Menge von Zuständen
  - A aufzählbare Menge von Aktionen (Labels)
  - T Menge von Transitionen, T ⊂ S x A x S
  - S_0 Menge von Startzuständen

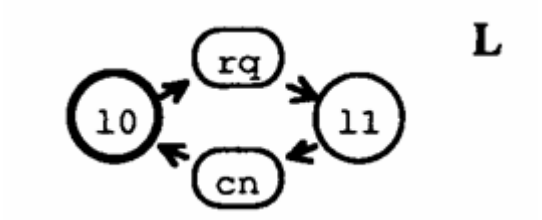
Diagramm

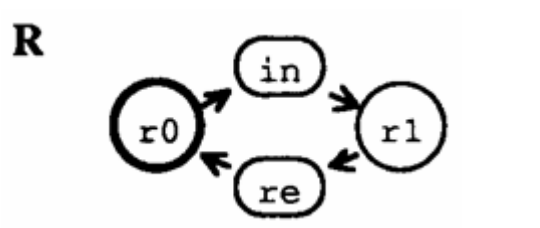


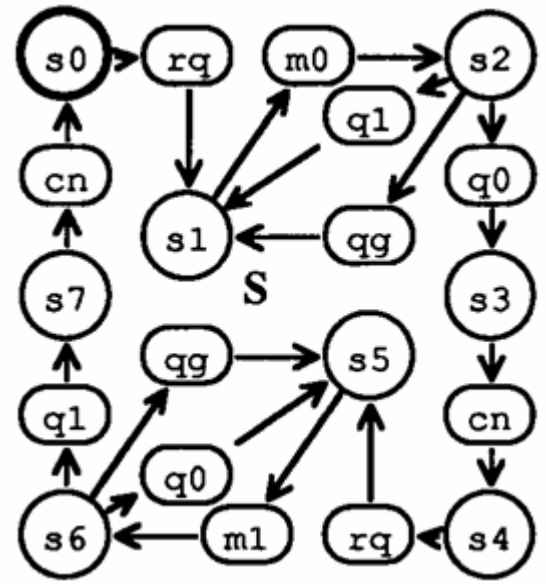
oder

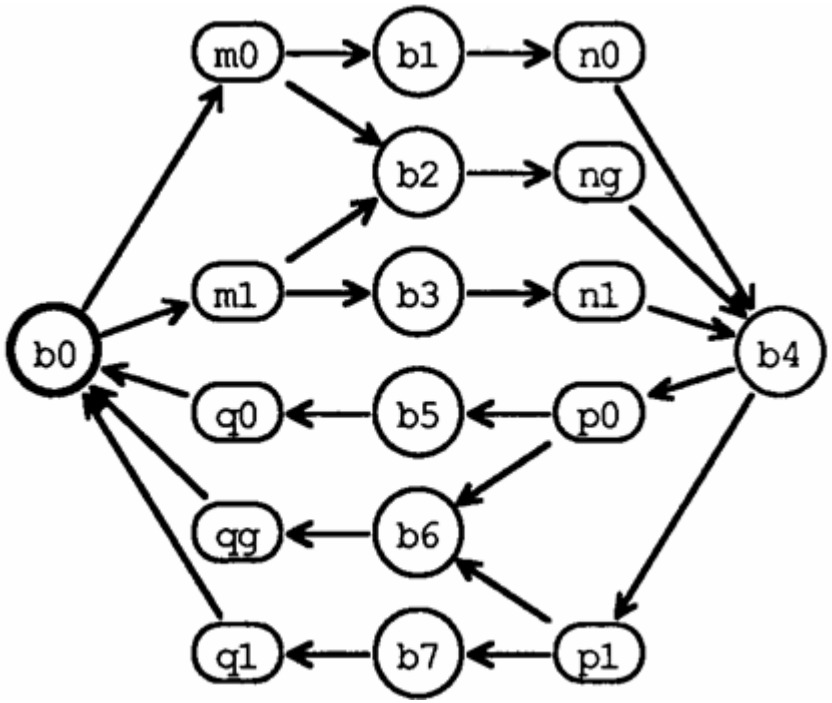


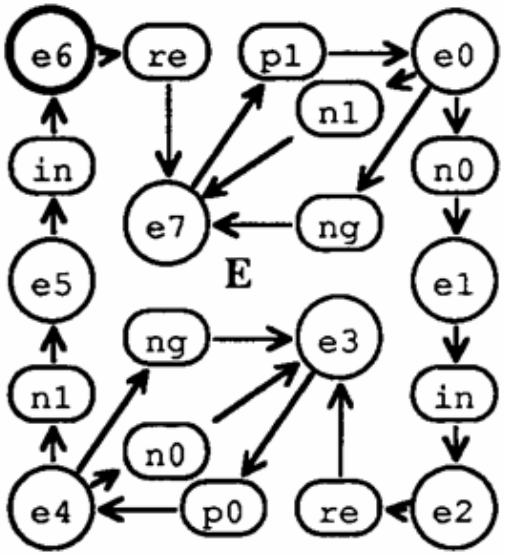
F4: Gefärbtes Transitionssystem (LTS) – Beispiele







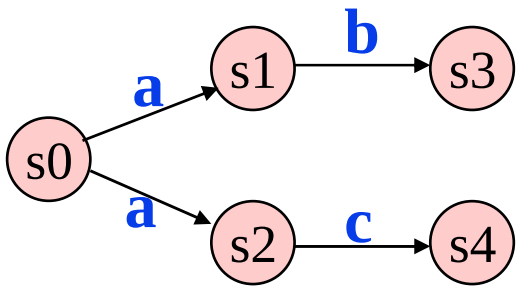




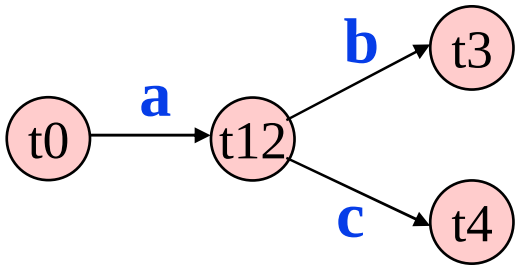
D'

F4: Bisimulation

- Endliche Akzeptoren, akzeptierte Sprache und äquivalente Zustände: Unterschied zur Bisimulation
- Unterschiede in der Menge möglicher nächster Verhaltensschritte!
- Beispiel



akzeptierte Sprache ab s0: **a(b|c)**
aber:
in s1 wird nur **b** für den nächsten Schritt akzeptiert



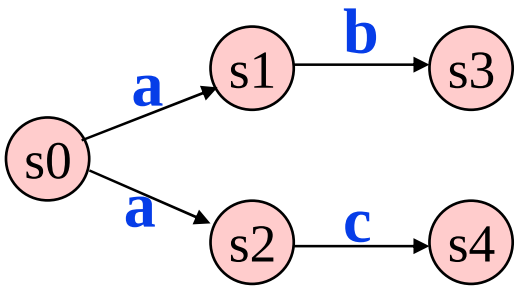
akzeptierte Sprache ab t0: **a(b|c)**
aber
in t12 werden sowohl **b** als auch **c** für den nächsten Schritt akzeptiert

➔ **gleiche Sprache, doch beobachtbar unterschiedliches Verhalten**

F4: Bisimulation – Verhaltensäquivalenz

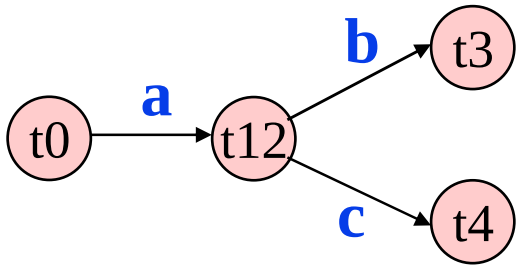
- Gegeben:
  - LTS < S, A, T >
    (Es können auch mehrere sein, z.B. zwei:
    $< S_1 \cup S_2, A_1 \cup A_2, T_1 \cup T_2 >$)
  - Binäre Relation R ⊂ S x S mit der Eigenschaft:
    - » $\forall s_1, s_2 : [s_1 R, s_2 \Rightarrow$
      $\forall a, s' \text{ mit } <s_1, a, s'> \in T:$
      $[\exists s'' : <s_2, a, s''> \in T \wedge s' R s'']]$

  R heißt Simulationsrelation.



- Gegeben:
  - LTS
  - Relation R, R ist Simulation und R^{-1} ist Simulation.

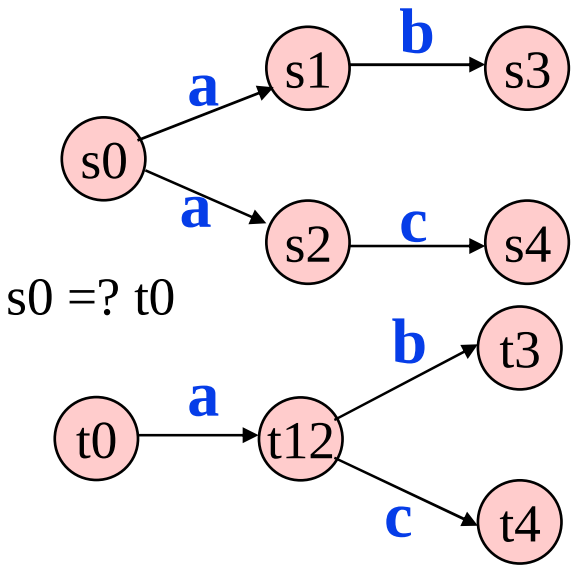
  R heißt Bisimulationsrelation.



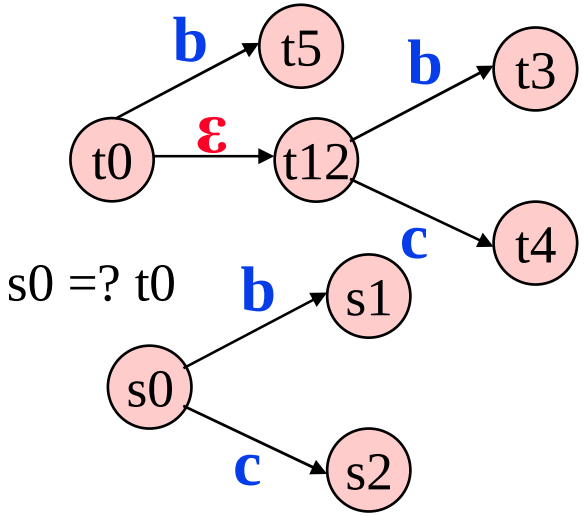
- Die Identät ist eine Bisimulationsrelation.
- Die Vereinigung zweier Bisimulationsrelationen ist wieder einer Bisimulationsrelation.
- Die umfassendste Bisimulationsrelation heißt üblicherweise einfach ***Bisimulation*** und liefert einen Äquivalenzbegriff, der sich auf die Unterscheidbarkeit von außen bezieht: **Verhaltensäquivalenz**.

F4: Starke und schwache Trace-Äquivalenz und Bisimulation …

- Trace-Äquivalenz versus Bisimulation
  - Trace-Äquivalenz
    Vergleich der möglichen Aktionsfolgen ohne Berücksichtigung der möglichen Verhaltensschritte (Sprach-Vergleich)
  - Bisimulation
    Vergleich entsprechend der Verhaltensschritt-Möglichkeiten.

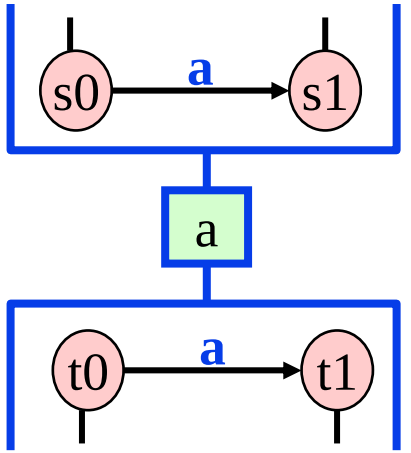


- Stark versus Schwach
  - Stark
    Alle Labels werden als gleich behandelt.
  - Schwach
    Sonderrolle interner ε-Aktionen und ε-Aktionsfolgen.



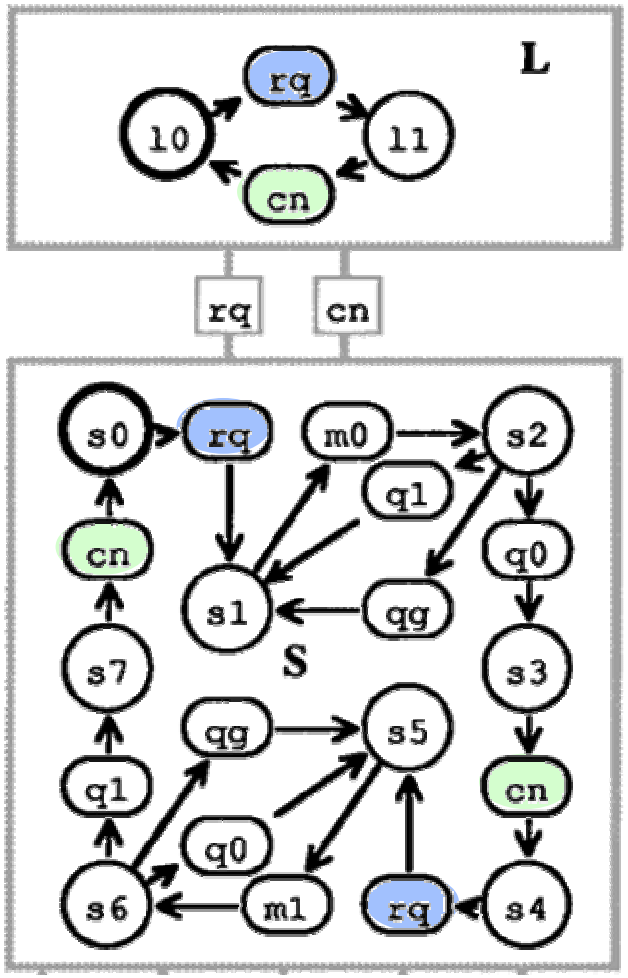
F4: Joint Action Kopplung

- Kopplung von LTSen über Ports, an welchen gemeinsame Aktionen ausgeführt werden.
  - 2-Parteien-Interaktionen, aber auch Mehr-Parteien-Interaktionen möglich
  - Synchronisation:
    Rendezvous, alle werden verzögert, bis alle bereit sind, dann findet die Interaktion dadurch statt, dass alle teilnehmenden Parteien gleichzeitig ihre jeweilige Aktion ausführen.
  - Kommunikation, Datenaustausch:
    Durch Abstimmung / Auswahl einer für alle Parteien möglichen Interaktionsvariante.



- Beispiele entsprechender realer Interaktionen
  - Ein Mensch zieht eine Schublade am Warenautomaten:
    Der Mensch muss an einer Schublade ziehen, welche der Automat auch freigegeben hat, wenn die Interaktion stattfinden können soll.
  - Das Arbeitstreffen:
    Die Akteure treffen sich, erarbeiten ein gemeinsames Ergebnis und trennen sich danach wieder.

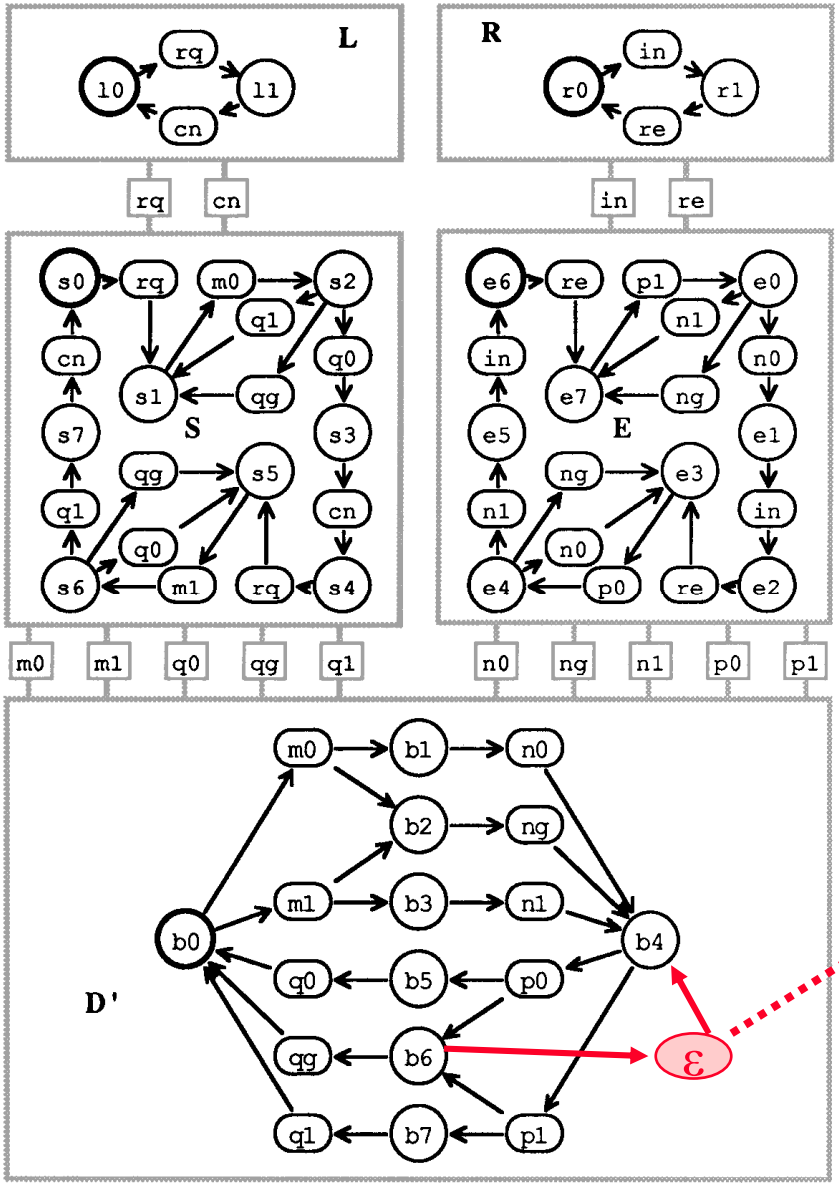
F4: Beispiel – Protokollsystem AB – 2 Teile davon



L kann den Übergang l0 –rq→ l1 nur ausführen,
wenn S gleichzeitig den Übergang s0–rq→s1 oder
den Übergang s4–rq→s5 ausführt.
Analog kann L den Übergang l1–cn→l0 nur ausführen,
wenn S gleichzeitig den Übergang s3–cn→s4 oder
den Übergang s7–cn→s0 ausführt.

Die nicht gekoppelten Übergänge (z.B. s1–m0→s2)
dürfen von S ohne Interaktion mit L ausgeführt werden.

F4: Beispiel – Protokollsystem AB



Hier ergibt sich ein geschlossenes Gesamtsystem.

Es gibt jeweils nur 2-Parteien-Kopplungen.

Alle Aktionen sind Teil einer Interaktion, d.h. es gibt keine internen Aktionen, welche eine Instanz in alleiniger Regie ausführt.

Man kann sich vorstellen, rein interne Aktionen, zu einer Aktion **ε** zu abstrahieren, deren Identität nach außen nicht wahrnehmbar wird (wohl aber ihre Wirkung auf das Folgeverhalten).

So könnte z.B. ein spontaner Verlust einer in Übertragung befindlichen Nachricht modelliert werden.

F5: Calculus of Communicating Systems (CCS)

- Systeme statt durch Diagramme mit algebraischen Gleichungssystemen definieren.
  - kompakte, präzise Darstellungen
  - Hintergrund LTS, aber es findet eine komplette Zustandsabstraktion statt

  Prozess-Algebren, prominentester Vertreter ist **CCS**

- *Anmerkuung:*
  Für viele gilt allerdings gerade das Zustandskonzept als die geeignetste Abstraktion einer Historie. Denn in einem Zustand lässt sich (bei klug gewählter Strukturierung) der für die Zukunft relevante Einfluss einer Historie kompakt repräsentieren.

Literatur:
Robin Milner: “A Calculus of Communicating Systems”, Springer Verlag, Lecture Notes in Computer Science, Vol. 92, 1980.
Standard
LOTOS: A formal description technique based on the temporal ordering of observational behavior, ISO International Standard 8807.

F5: CCS – Operationen – Agent und Variablen, **nil**

- Ein System heißt Agent
  (Ein Agent kann ein einzelner sequentieller Prozess oder auch ein System aus nebenläufigen Prozessen sein).
- Ein Agent kann durch eine Agentenvariable repräsentiert werden.
  Die Variable wird mit dem Verhalten des Agenten gleichgesetzt.
- Es gibt eine Agentenkonstante: **nil**, sie steht für das Stop-Verhalten.

Agent:

X = *Ausdruck, der Verhalten von X beschreibt.*

Agent:

Y = nil

Agent:

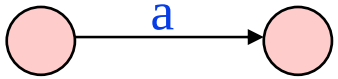
Z = Y

Y und Z sind Agenten, die stoppen.
X besitzt ein mit dem Ausdruck beschriebenes Verhalten.
→ ***Operationen zur Bildung von Ausdrücken***

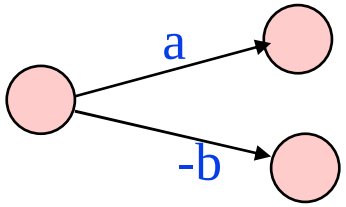
F5: CCS – Operationen – Endlicher sequentieller Agent

- Aktionsnamen (Marken):
  - positive Aktionen, z.B. a, b, c
  - komplementäre Aktionen, z.B. -a, -b, -c oder $\overline{a}$, $\overline{b}$, $\overline{c}$
  - die interne Aktion **τ**
- Operationen
  - Aktionspräfix: Agent ::= Aktion „*“ Agent, z.B.: a*-b*nil
  - Auswahl: Agent ::= Agent „+“ Agent, z.B.: (a*-b*nil)+(c*nil)
  - *Anmerkung: „*nil“ wird häufig weggelassen, z.B. a*b+c*
- Zugehöriges LTS

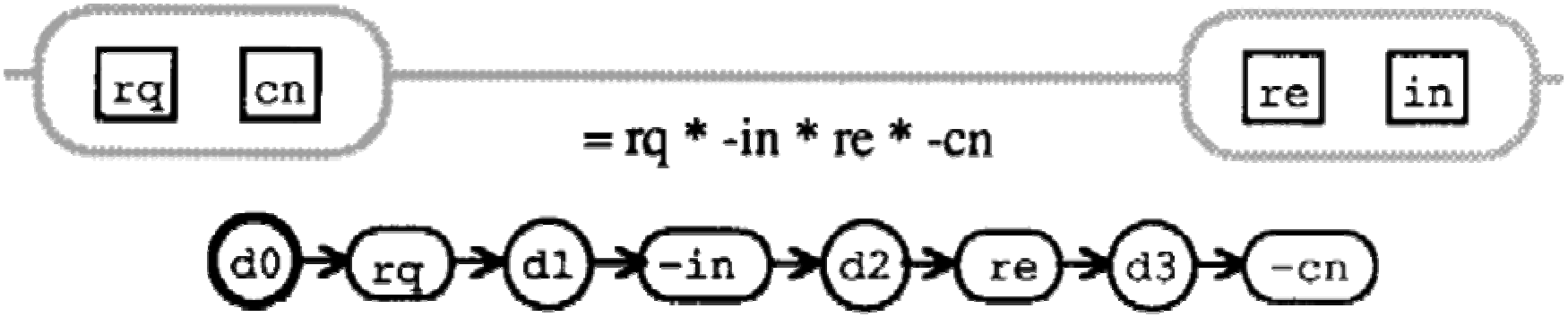
X = a * nil



Y = a * nil + -b * nil



F5: Beispiel – Dienst D_e

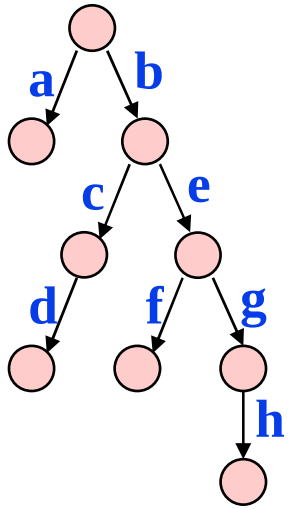


- Aktionen: **rq, -in, re, -cn**
  (Es gibt Sende- und Empfangsaktionen. Sie sind komplementär.)

- Verhalten:
  1. rq Empfange rq
  2. -in Sende in
  3. re Empfange re
  4. -cn Sende cn
  5. nil Stop

F5: CCS – Synchronisationsbaum ST

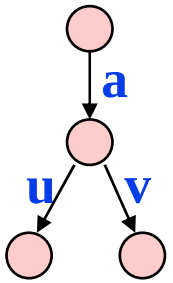
- Synchronisationsbaum ST_X eines Agenten X
  Baum-Darstellung des LTS, Zustände anonym
- Beispiel:
  X = **a + b*(c*d + e*(f + g*h))**



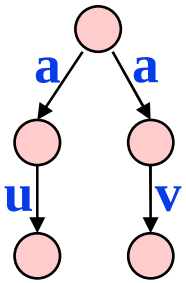
Anmerkung:
Im zyklenfreien LTS werden einfach die Zustandsinschriften weggelassen.
Bei Zyklen werden diese „abgewickelt“. Dies führt zu unendlich tiefen Bäumen.

F5: CCS – Fehlendes Distributivgesetz

- Beachte!
  - **X=a*(U+V)** ist nicht verhaltensgleich zu **Y=a*U + a*V**
- Denn:
  - X kann eine **a** Aktion ausführen und sich danach wie X oder Y verhalten.
  - mit **U=u*nil** und **V=v*nil** z.B.
    kann X nachdem es **a** ausführte, sowohl **u** als auch **v** ausführen.



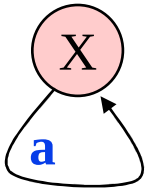
  - Y kann ebenfalls zunächst eine a Aktion ausführen.
    Dabei fällt jedoch schon die Entscheidung, ob danach nur ein Folgeverhalten X oder ein Folgeverhalten Y möglich ist.
  - mit **U=u*nil** und **V=v*nil** z.B.
    kann y nachdem es **a** ausführte, entweder nur **u** oder nur **v** ausführen.



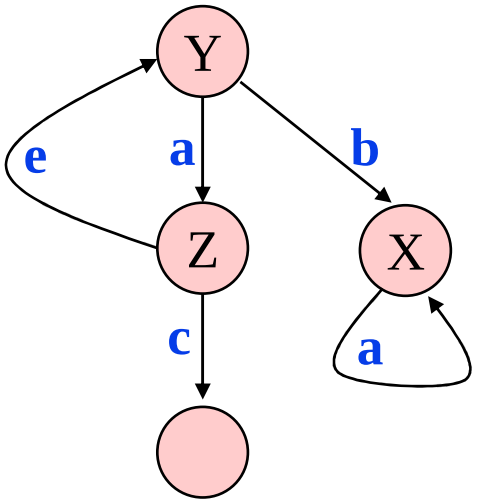
F5: CCS – Operationen – Unendlicher sequentieller Agent

- Agentenvariablen und Rekursion
  - Unendliche Agentenverhalten werden durch rekursive Gleichungssysteme definiert.
  - Sie entsprechen LTSen mit Zyklen.

X = a * X



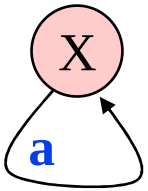
Y = a * Z + b * X
Z = e * Y + c * nil

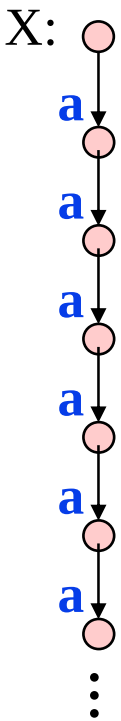


F5: CCS – Operationen – Unendlicher sequentieller Agent

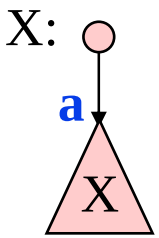
- Das abgewickelte LTS bildet den (unendlichen) Synchronisationsbaum

X = a * X





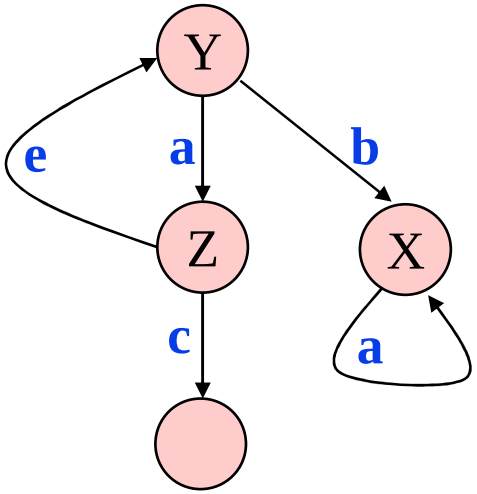
Bildungsgesetz:

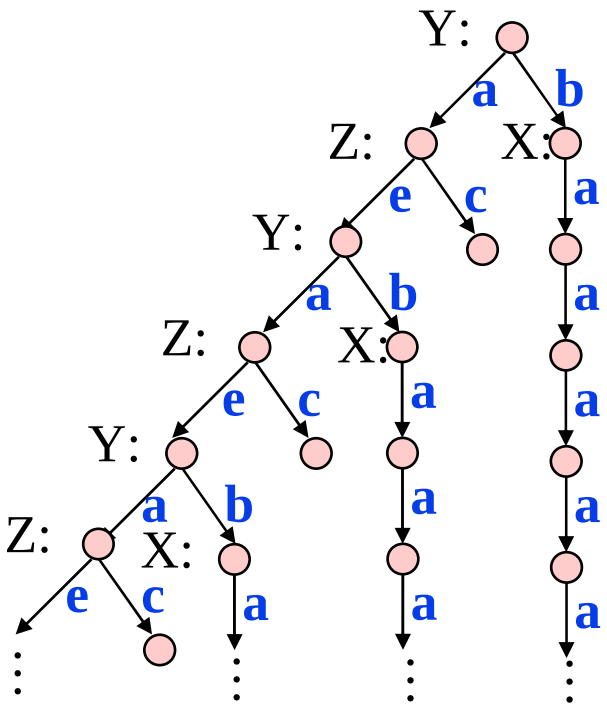


F5: CCS – Operationen – Unendlicher sequentieller Agent

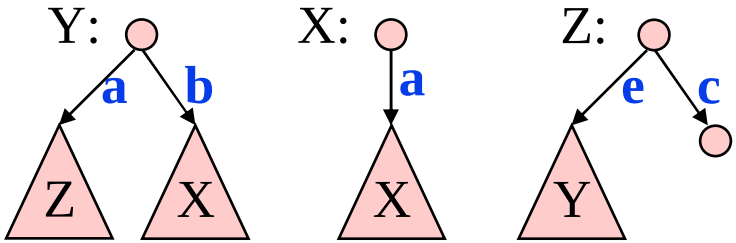
- Das abgewickelte LTS bildet den (unendlichen) Synchronisationsbaum

Y = a * Z + b * X
X = a * X
Z = e * Y + c * nil

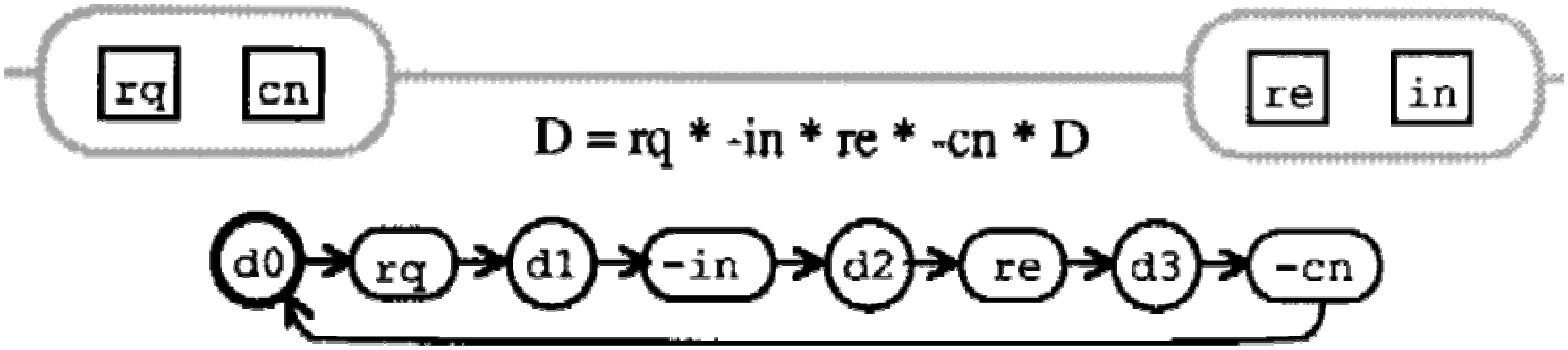


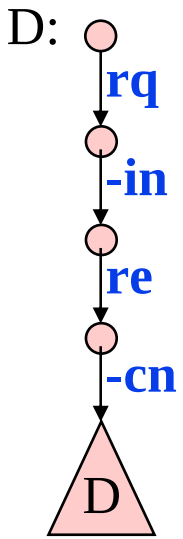


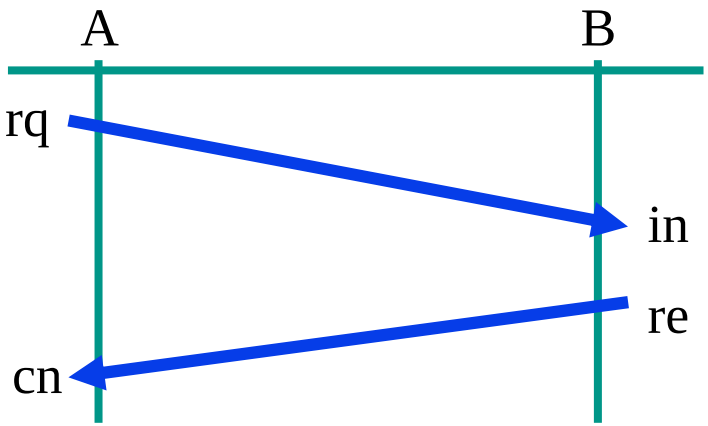
Bildungsgesetze:



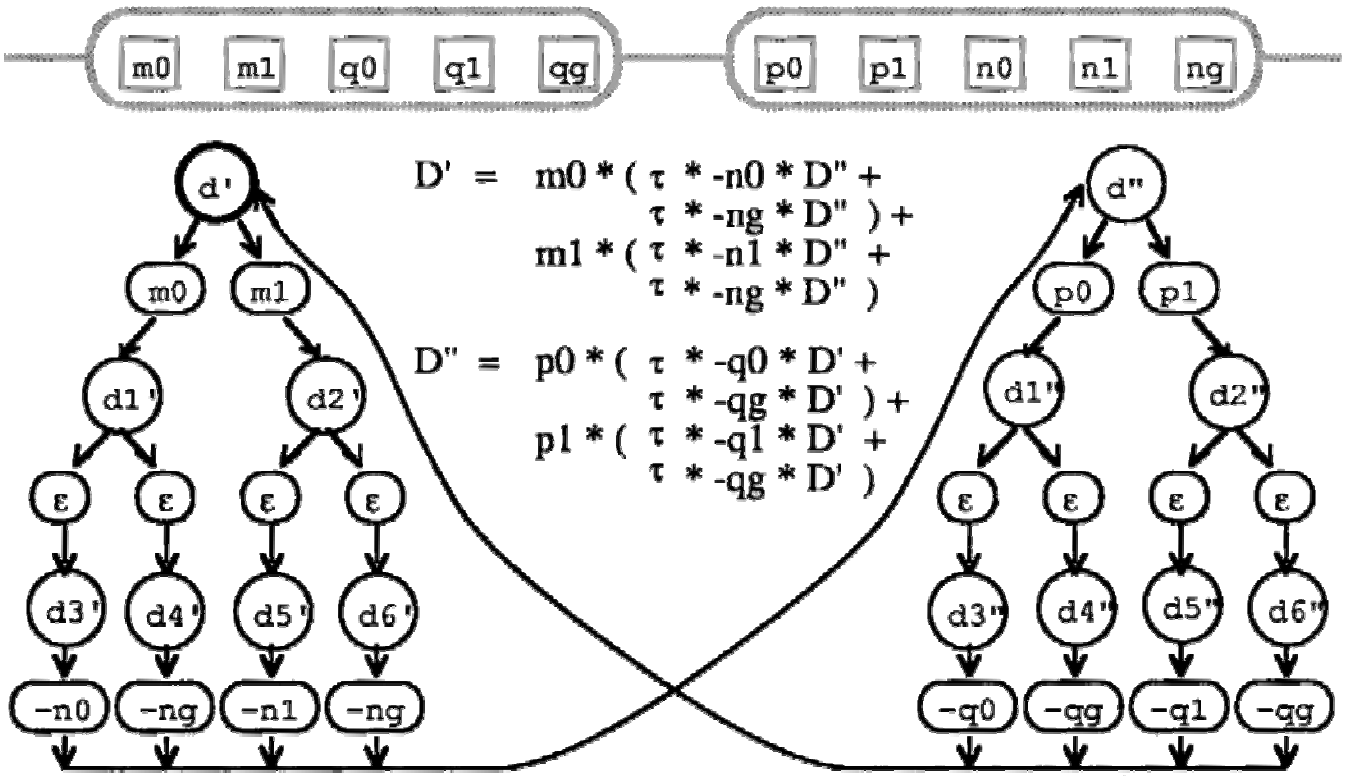
F5: Beispiel – Dienst D

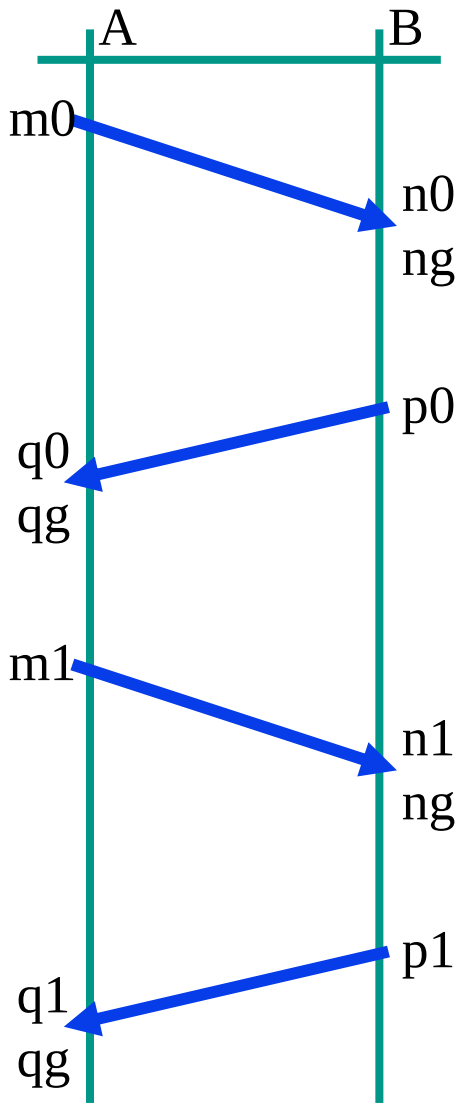






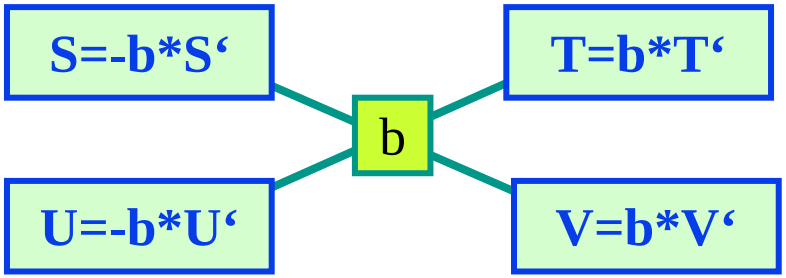
F5: Beispiel – Dienst D‘





F5: CCS – Operationen – Systembildung: Interaktionen

- In CCS sind Interaktionen generell gerichtete 2-Parteien-Interaktionen:
  - Ein (sendender) Agent führt eine Aktion **–a** aus.
  - Gleichzeitig führt ein empfangender Agent **a** aus.
- Es können aber beliebig viele Parteien Aktionen eines Typs (**a** , **-a**) ausführen.



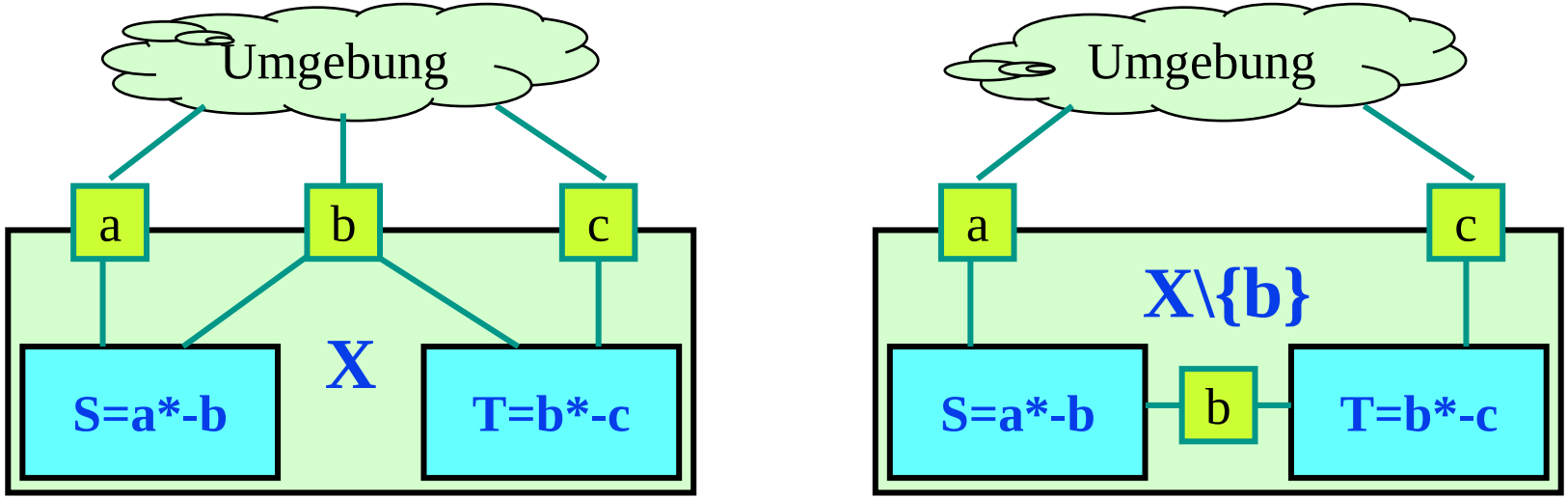
Hier sind die 4 Agenten S, T, U, V vorhanden und können
im nächsten Schritt **b** bzw. **-b** ausführen.
Es gibt folgende Möglichkeiten für den nächsten Schritt:

- S und T interagieren, U und V bleiben stehen, Schrittergebnis ist S‘, T‘, U, V
- U und V interagieren, S und T bleiben stehen, Schrittergebnis ist S, T, U‘, V‘
- S und V interagieren, U und T bleiben stehen, Schrittergebnis ist S‘, T, U, V‘
- U und T interagieren, S und V bleiben stehen, Schrittergebnis ist S, T‘, U‘, V

Beachte: Es ist in allen 4 Fällen je ein Sender und je ein Empfänger beteiligt.

F5: CCS – Operationen – Systembildung: Hiding und τ

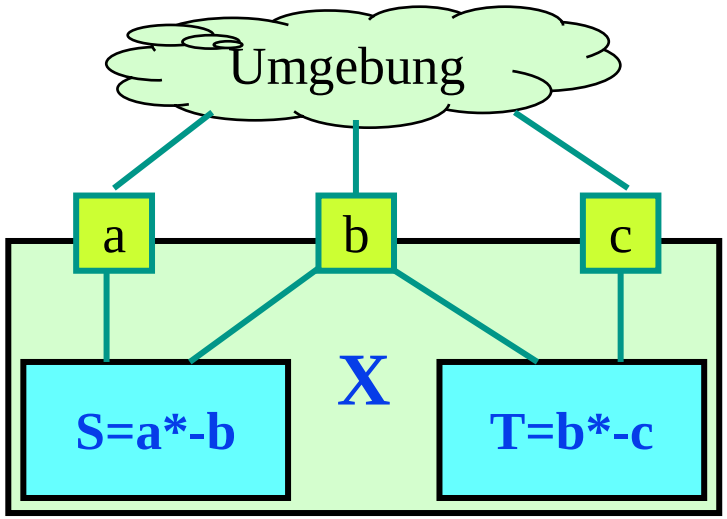
- Alle Aktionen eines Agenten, außer **τ**, sind potentielle Interaktionen mit der Umgebung des Agenten.
- Wie verberge ich die internen Interaktionen eines Subsystems nach außen?
- Operation: Hiding **X = X \ { a1, a2, a3, ..}**
  schirmt a1, a2, a3, .. und deren Komplemente nach außen ab.



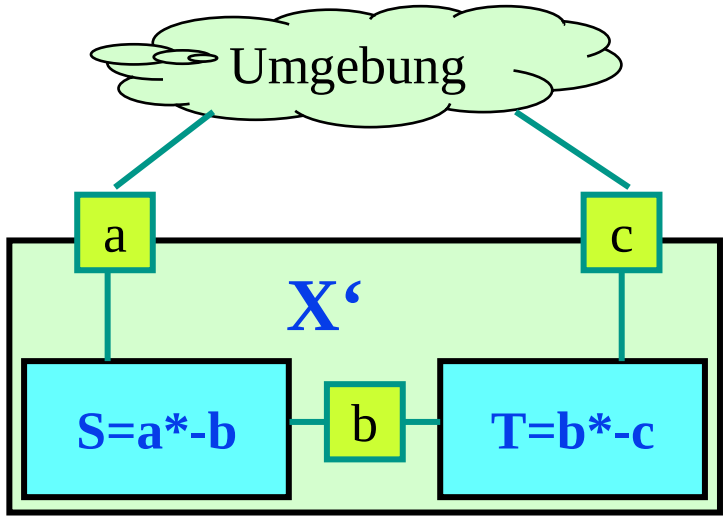
- Verborgene innere Interaktionen erscheinen nach außen als **τ**

F5: CCS – Operationen – Systembildung: Komposition

- Zur Bildung eines aus Agenten (oder Subsystemen) zusammengesetzten Systems dient die Kompositionsoperation „|“
- **X = S | T** steht für ein aus S und T gebildetes System
- **X‘ = (S | T) \ {b}** steht für ein aus S und T gebildetes System, das per **b**, **-b** nur intern interagiert.



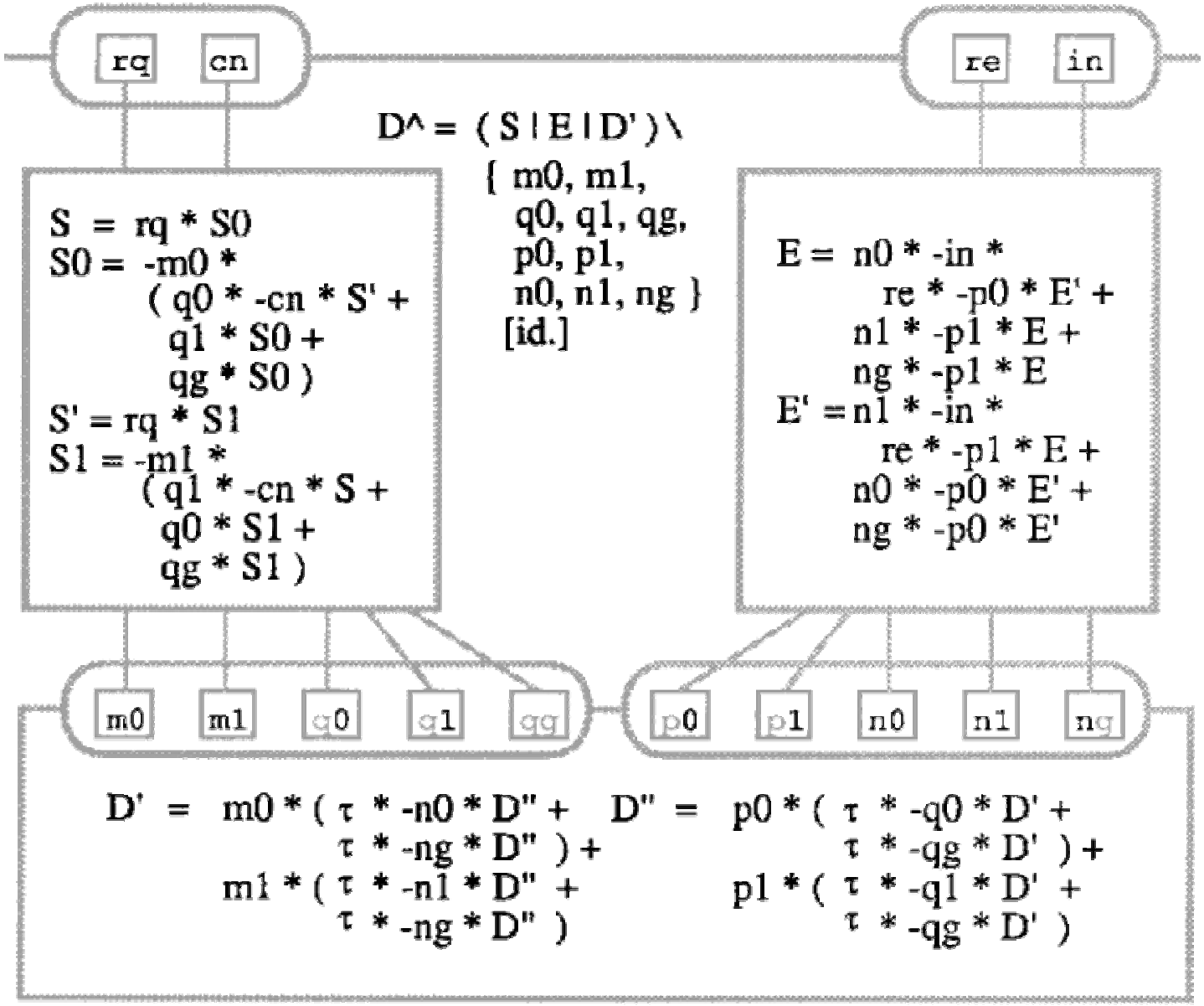
X = S | T



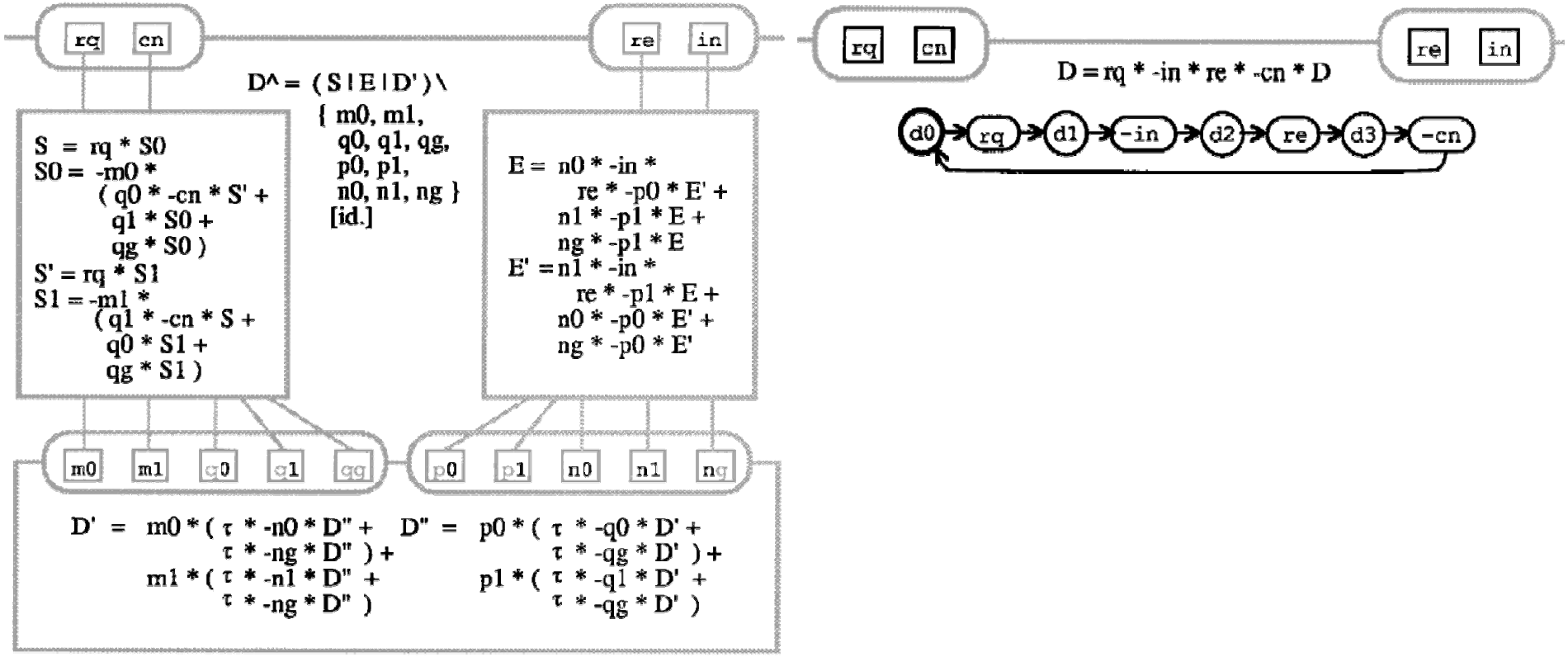
X‘ = (S | T) \ { b }

Es gilt hier übrigens:
X‘ = a τ*-c*nil*

F5: Beispiel – Protokollsystem AB



F5: Beispiel – Protokollsystem AB und Dienst D



D^ wirkt nach außen wie D:

Das Protokollsystem erbringt / implementiert den Zieldienst.

Lässt sich das ausrechnen?

F5: CCS – Expansionstheorem

- Systembildung in der Regel per Kombination aus Komposition und Hiding (Modulbildung), z.B. **S = (U | V) \ I**
- Die Operatoren **|** und **** sind definierbar über *****, **+**, **nil** *(nach Interleaving-Konzept)*
- Aus den Definitionen lässt sich das Expansionstheorem ableiten.

Beispiel

S = a * u + b * v + c * w

T = d * x + -b * y + -c * z

P = (S | T) \ {b}

P = a * (u | T) \ {b} + c * (w | T) \ {b} — *S interagiert mit Umgebung*

\+

d * (S | x) \ {b} + -c * (S | z) \ {b} — *T interagiert mit Umgebung*

\+

τ * (v | y) \ {b} + **τ** * (w | z) \ {b} — *S und T interagieren*

F5: CCS – Expansionstheorem

$$S = \sum_{\mu_i \in M} \mu_i . S_i$$

$$T = \sum_{\nu_j \in N} \nu_j . t_j$$

$$I \subset M \cup N$$

$$P = (S \mid T) \setminus I$$

$$P = \sum_{\substack{\mu_i \in M, \\ \mu_i, \overline{\mu_i} \notin I}} \mu_i . (S_i \mid T) \setminus I$$

S-Schritte mit Umgebung

$$+ \sum_{\substack{\nu_j \in N, \\ \nu_j, \overline{\nu_j} \notin I}} \nu_j . (S \mid t_j) \setminus I$$

T-Schritte mit Umgebung

$$+ \sum_{\substack{\mu_i \in M, \\ \nu_j \in N, \\ \mu_i = \overline{\nu_j}}} \tau . (S_i \mid t_j)\, I$$

S-T-Interaktionen

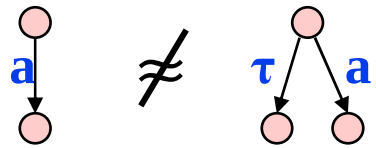
F5: CCS – Observational Equivalence

- Schwache Bisimulation $\approx = \bigcup_{\text{R ist schwache Bisimulation}} R$

- Einige Algebraische Regeln
  - *$\approx$ ist Äquivalenzrelation*
    - » $A \approx A$
    - » $A \approx B \Rightarrow B \approx A$
    - » $A \approx B \wedge B \approx C \Rightarrow A \approx C$
  - *$\approx$ ist Kongruenz für alle Operationen außer „+“*
    - » $A \approx B \Rightarrow a * A \approx a * B$
    - » $A \approx B \Rightarrow A \mid D \approx B \mid D$
    - » $A \approx B \Rightarrow A \setminus \{a\} \approx B \setminus \{a\}$
  - *Das τ ist unsichtbar*
    - » $\tau * A \approx A$

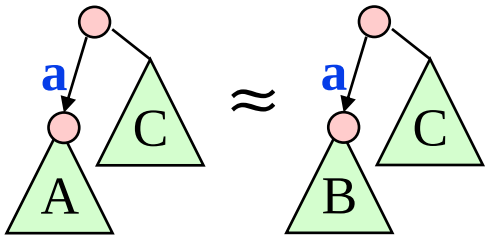
F5: CCS – Observational Equivalence

- ≈ keine Kongruenz für „+“
  - wegen τ * A ≈ A gilt nil ≈ τ * nil
  - es gilt aber
    nil + a * nil ≉ τ * nil + a * nil



- Def: A ist stabil ⇔
  A kann nicht ≈ als ersten
  Übergang ausführen

A ≈ B
a * A + C ≈ a * B + C



- A, B stabil ∧ A ≈ B
  ⇒ A + C ≈ B + C

F5: CCS – Observational Congruence

- Def: A = B („=“ Observational Congruence) ⇔
  „=“ ist die größte Teilmenge von „ ≈“ bei der für alle Paare A, B die Kongruenz bezüglich „+“ gegeben ist.
- Einige Algebraische Regeln
  - *Verhältnis zu ≈*
    - » A = B ⇒ A ≈ B
    - » A, B stabil ^ A ≈B ⇒ A = B
    - » A ≈ B ⇒ a*A = a*B
  - *Äquivalenzrelation*
    - » A = A
    - » A = B ⇒ B = A
    - » A = B ∧ B = C ⇒ A = C
  - *Kongruenz für alle Operationen*
    - » A = B ⇒ a * A = a * B
    - » A = B ⇒ A | D = B | D
    - » A = B ⇒ A \ {a} = B \ {a}
    - » A = B ⇒ A + D = B + D
  - *Eigenschaften von „+“*
    - » A + B = B + A
    - » A + (B + C) = (A + B) + C
    - » A + nil = A
    - » A + A = A
  - *Eigenschaften von „|“*
    - » A | B = B | A
    - » A | (B | C) = (A | B) | C
    - » A | nil = A
    - » **Def. von „|“**
    - » **Expansionstheorem**
  - *Interner Übergang*
    - » a * **τ** * A = a * A
    - » A + **τ** * A = **τ** * A

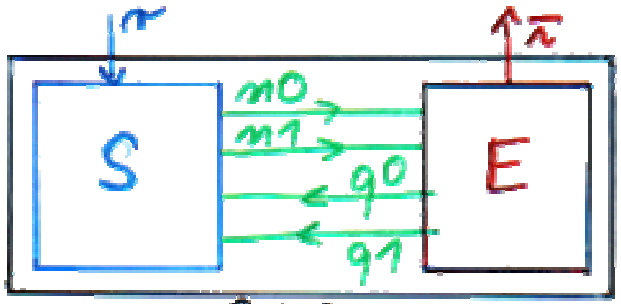
…

→ *Algebraisches Rechnen mit Prozessen*

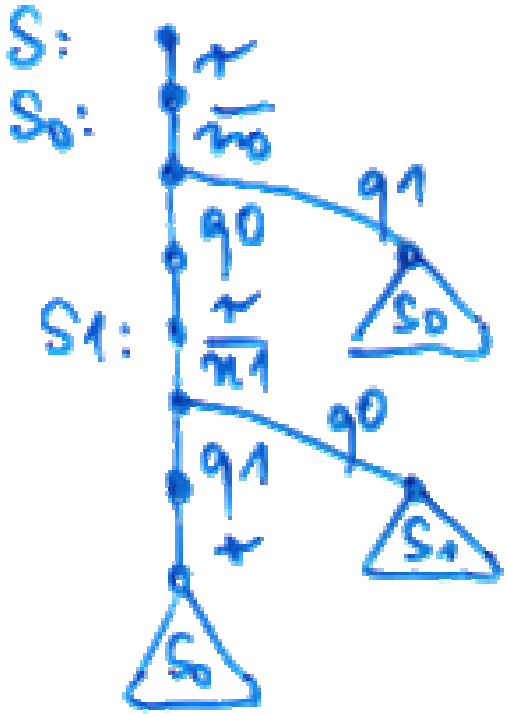
F5: CCS – Verifikationsbeispiel

AB-Protokoll
„Ohne Störungen,
Instanzen direkt gekoppelt“

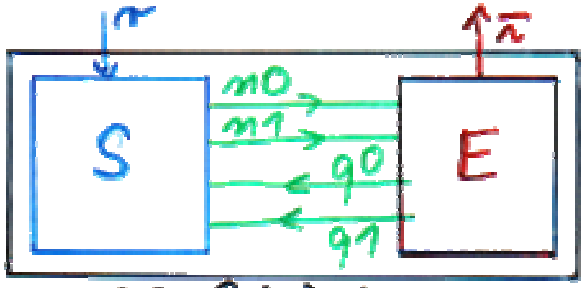
Gilt D = AB ?



$$S = \tau . S_0$$
$$S_0 = \overline{m0} . (q0 . \tau . S_1 + q1 . S_0)$$
$$S_1 = \overline{m1} . (q1 . \tau . S_0 + q0 . S_1)$$



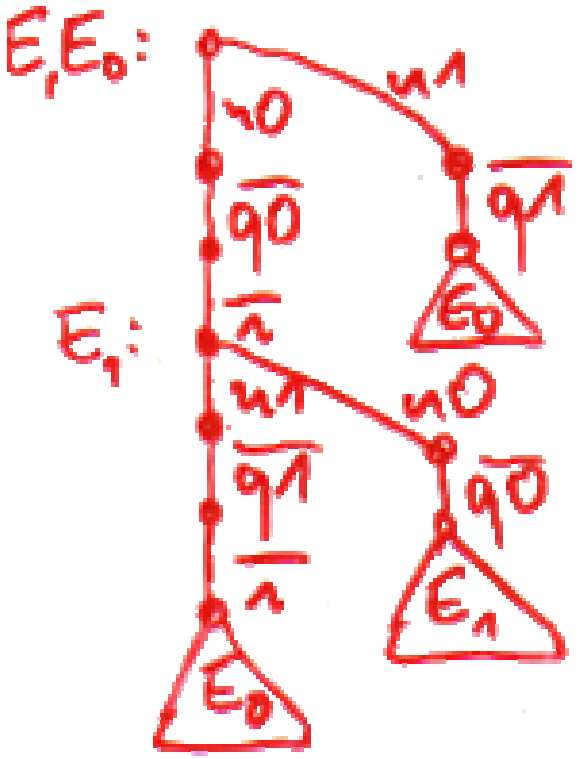
F5: CCS – Verifikationsbeispiel



$$E = E_0$$

$$E_0 = m0 \cdot \overline{q0} \cdot \bar{\pi} \cdot E_1 + m1 \cdot \overline{q1} \cdot E_0$$

$$E_1 = m1 \cdot \overline{q1} \cdot \bar{\pi} \cdot E_0 + m0 \cdot \overline{q0} \cdot E_1$$



F5: CCS – Verifikationsbeispiel

$$
\begin{aligned}
AB &= (S \mid E) \setminus I \\
&= r^* ((S0 \mid E0) \setminus I) && \text{! Expansionstheorem} \\
AB &= r^* X \\
X &= (S0 \mid E0) \setminus I \\
X &= (S0 \mid E0) \setminus I \\
&= T^*((q0^* r^* S1 \mid -q0^* - i^* E1) \setminus I) && \text{! Expansionstheorem} \\
&= T^* T^*((r^* S1 \mid -i^* E1) \setminus I) && \text{! Expansionstheorem} \\
&= T^* T^*(r^*((S1 \mid -i^* E1) \setminus I) + \\
&\qquad -i^*((r^* S1 \mid E1) \setminus i)) && \text{! Expansionstheorem} \\
&= T^* T^*(r^* - i^*((S1 \mid E1) \setminus I) + \\
&\qquad -i^* r^*((S1 \mid^* E1) \setminus I)) && \text{! Expansionstheorem}
\end{aligned}
$$

F5: CCS – Verifikationsbeispiel

$$
\begin{aligned}
X &= T^*T^*(r^*\text{-}i^*Y + \text{-}i^*r^*Y) \\
Y &= (S1|E1)\backslash I \\
Y &= (S1|E1)\backslash I \\
&= T^*((q1^*r^*S0|\text{-}q1^*\text{-}i^*E0)\backslash I) && \text{! Expansionstheorem} \\
&= T^*T^*((r^*S0|\text{-}i^*E0)\backslash I) && \text{! Expansionstheorem} \\
&= i^*T^*(r^*((S0|\text{-}i^*E0)\backslash I) + \\
&\qquad \text{-}i^*((r^*S0|E0)\backslash i)) && \text{! Expansionstheorem} \\
&= T^*T^*(r^*\text{-}i^*((S0|E0)\backslash I) + \\
&\qquad \text{-}i^*r^*((S0|^*E0)\backslash I)) && \text{! Expansionstheorem} \\
Y &= T^*T^*(r^*\text{-}i^*X + \text{-}i^*r^*X) \\
X &\approx Y \\
X &\approx r^*\text{-}i^*X + \text{-}i^*r^*X
\end{aligned}
$$

F5: CCS – Daten

- Datenvariablen und Ausdrücke
  - Es können Ausdrücke und Datenvariablen vorkommen
    - » Ein/Ausgabe
    - » IF ausdruck THEN Agent1 ELSE Agent2
  - Vorbelegung beim Agentenaufruf
    z.B. X { ausdruck / variable, ausdruck / variable, .. }
- Datenaustausch
  - Beim Senden können Ausdrücke und beim Empfangen Variablen angegeben werden.
    X { **3** / **i** } = **-a!**(**i+5**) * **b?x** * **c!**(**x+i+3**) * X { **x+8** / **i** }
- Aus Synchronisationsbäumen werden Kommunikationsbäume.

Grundidee ähnlich Mealy-Erweiterung, Beispiel

Y { x : (0,1,2) } = -a!(x+1 mod 3) * b?x * Y(x)

steht für

Y = Y0 + Y1 + Y2

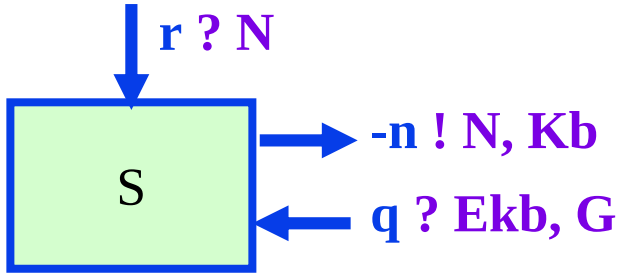
Y0 = -a1 * (b0 * Y0 + b1 * Y1 + b2 * Y2)

Y1 = -a2 * (b0 * Y0 + b1 * Y1 + b2 * Y2)

Y2 = -a0 * (b0 * Y0 + b1 * Y1 + b2 * Y2)

F5: CCS – Daten

- Beispiel




```
S =    S‘ { 0 / Kb }                          ! S ist S‘ mit 0 als Kontrollbit

S‘ =   r ? N * S“ { Kb / Kb , N / N }         ! Empfange N, weiter mit S“

S“ =   -n ! N, Kb *                           ! Sende N-PDU mit Kontrollbit
       q ? Ekb, G *                           ! Empfange Quittungsbit , Störanzeige
       if not G and Ekb = Kb                  ! Positive Quittung?
         then S‘ { not Kb / Kb }              ! Weiter mit Kb gekippt!
         else S“ {Kb / Kb , N / N }           ! Wiederholen!
```