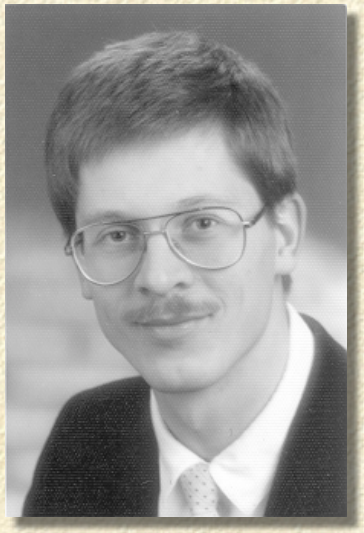
FGI 2: Formale Grundlagen der Informatik II

Modellierung & Analyse

von

Informatik-Systemen

Rüdiger Valk und Daniel Moldt und Michael Köhler-Bußmeier



Di 12:15 - 13:00 | 5 Minuten Pause | 13:05 - 13:50 Erz H

Do 12:15 - 13:00 | 5 Minuten Pause | 13:05 - 13:50 Phil A

Skript und Folien unter

http://www.informatik.uni-hamburg.de/TGI/

Benutzername: lossol Passwort: capetri

Papierversion in den Übungsgruppen

Email-Verteiler:

Inf-Bacc@informatik.uni-hamburg.de

Inf-Diplom@informatik.uni-hamburg.de

MCI-Bacc@informatik.uni-hamburg.de

CIS-Bacc@informatik.uni-hamburg.de

SSE-Bacc@informatik.uni-hamburg.de

weitere bitte an :
valk@informatik.uni-hamburg.de

Übungen am Montag und Dienstag

Beginn: gestern

Dr. Michael Köhler

koehler@informatik.uni-hamburg.de

Modalitäten siehe
FGI-2-Seite

Ba - Stud. zur Vorlesung anmelden !

Ba - Stud. in der Übungsgruppe anmelden !

auf der FGI2-Seite:

Prüfungsstoff für Diplomstudierende, die FGI 2 im WiSe 2010/2011 gehört haben und sich in PNL prüfen lassen.

Der Vorlesungsstoff der Vorlesung enthält Teile, die Diplomstudierenden auch in anderen Vorlesungen geboten werden. Er wird deshalb nicht in der mündlichen Prüfung zu PNL abgefragt. Falls der Stoff in der Vorlesung F4 vorkam, so kann in der mündlichen Prüfung so darauf Bezug genommen werden, wie dies für jeden Stoff des Grundstudiums gilt.

Verbindlich sind alle Teile des FGI2-Skriptes zum WiSe 2010/2011 bis auf den Abschnitt 1.4.4 und die Kapitel 4 und 6.

Rüdiger Valk
September 2010

auf der FGI2-Seite:

Prüfungsstoff für Studierende der Wirtschaftsinformatik, die FGI 2 im WiSe 2010/2011 gehört haben und sich in F4 prüfen lassen.

Der Vorlesungsstoff der Vorlesung enthält Teile, die dem Stoff der F4-Vorlesung entsprechen. Nach wie vor ist es möglich, sich nach dem F4-Skript prüfen zu lassen.

Alternativ kann die Prüfung auch nach den Kapiteln 1, 2, 3 und 6 des FGI 2 Skriptes 2010/11 erfolgen. In beiden Fällen ist es sinnvoll die entsprechenden Teile in der Vorlesung FGI 2 zu hören.

Wenn FGI 2 so als Prüfungsstoff gewählt wird, ist es nicht mehr möglich, FGI 2 als Ersatz für PNL im Vertiefungsgebiet zu wählen.

Rüdiger Valk, Daniel Moldt
September 2010

Inhalt

- *verschiedene* Modelle
  - Transitionssysteme, Petrinetze, Prozessalgebra
- *verschiedene* Prozessbegriffe
  - lineare Ordnung, partielle Ordnung, Zeitstempel, Prozessgraphen
- *verschiedene* Algorithmen
  - Konsistenz, parallele Alg., Beschränktheit
- *verschiedene Formen der* Verifikation
  - Model Checking, Temporale Logik

Inhalt

Transitionssysteme	**Kapitel 1 - Teil 1**
Platz/Transitions-Netze	**Kapitel 3 - Teil 1**
Workflow-Netze	**Kapitel 6 - Teil 1**
Höhere Petrinetze	**Kapitel 7**

Modelle

Partielle Ordnungen	**Kapitel 2**
Zeitstempel	

Prozesse

Verifikation durch Model-Checking	**Kapitel 1 - Teil 2**
Verifikation bei Petrinetzen	**Kapitel 3 - Teil 2**
Prozessalgebra	**Kapitel 5**

Verifikation

Konsistenz	**Kapitel 6 - Teil 2**
RAM, PRAM, parallele Algorithmen	**Kapitel 4**

Algorithmen

Anhang A Referenzen auf englische Literatur

Literaturverzeichnis

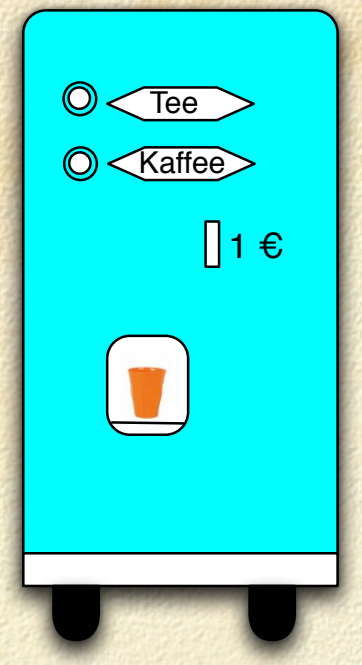
Index

Kapitel 1

1 Transitionssysteme und Verifikation

Teil 1

zentrale Begriffe des Kapitels



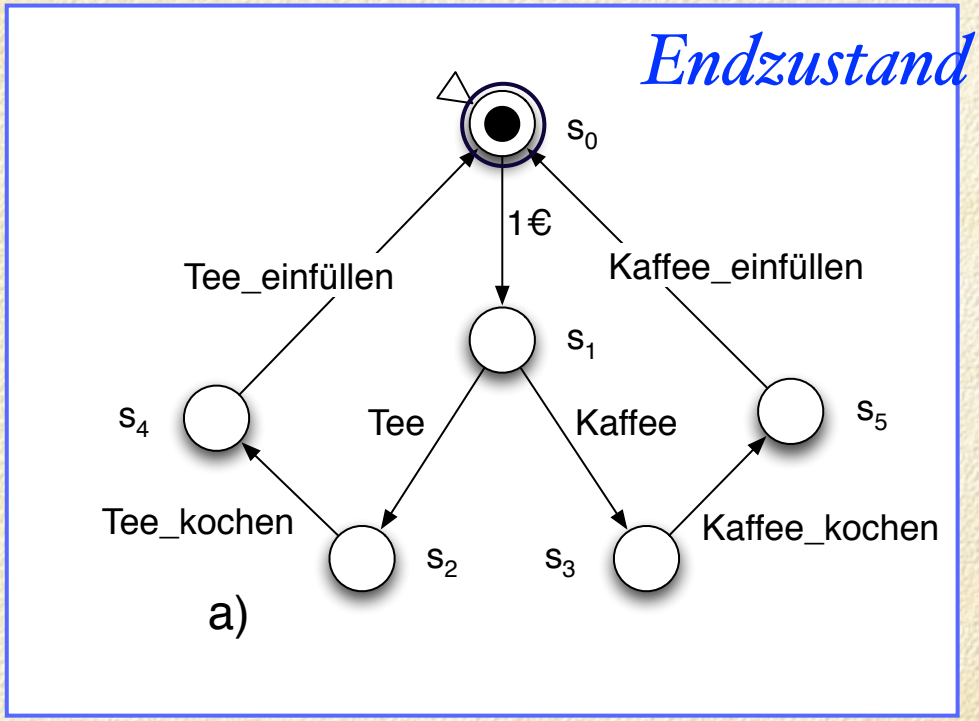
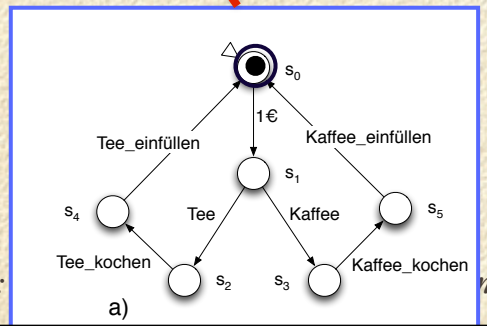
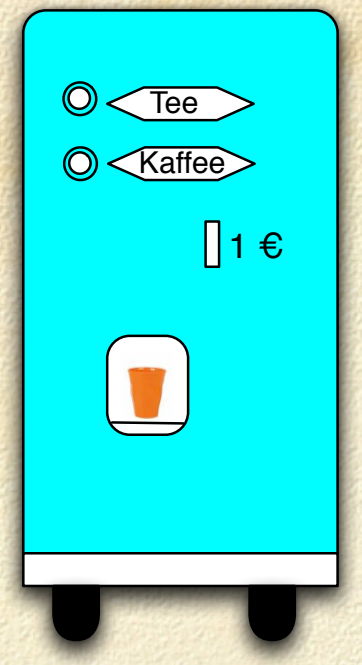


Abbildung 1.1: Getränkeautomat

dazu:
„akzeptierender Automat“





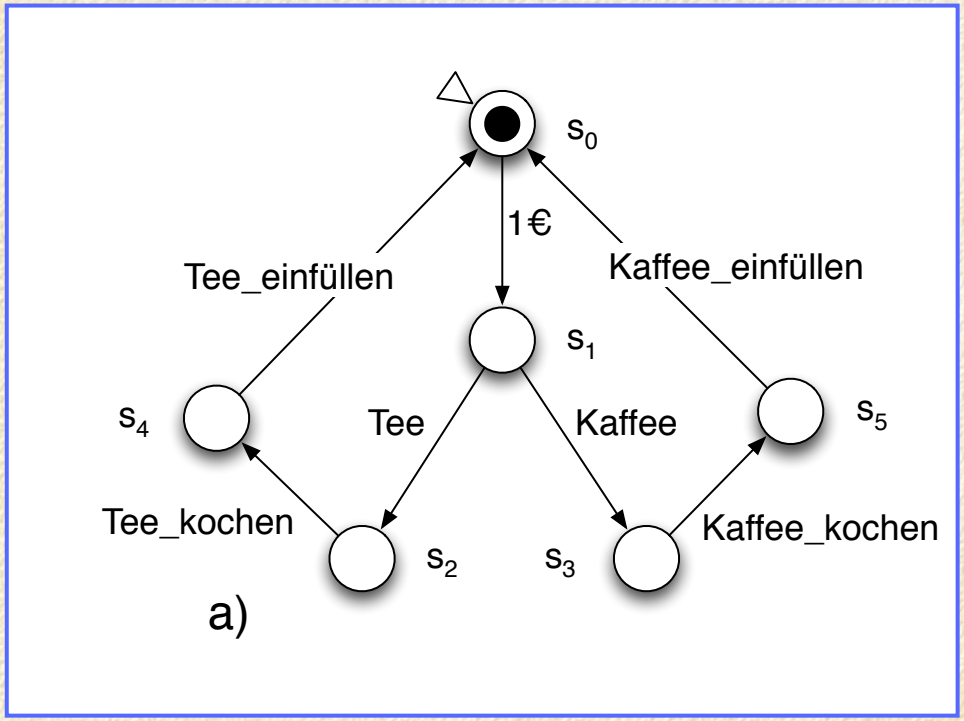
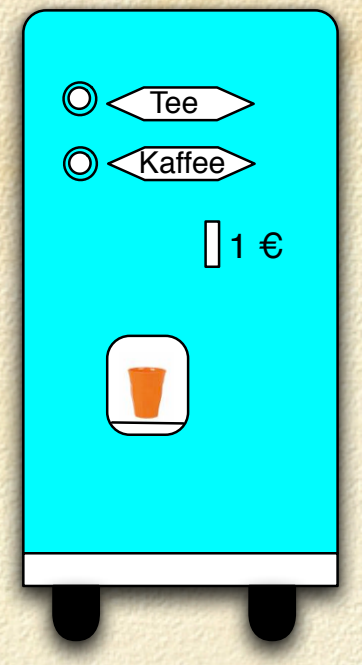


Abbildung 1.1: Getränkeautomat

1€	Kaffee	Kaffee_kochen	Kaffee_einfüllen	#			

dazu:
„akzeptierender Automat“



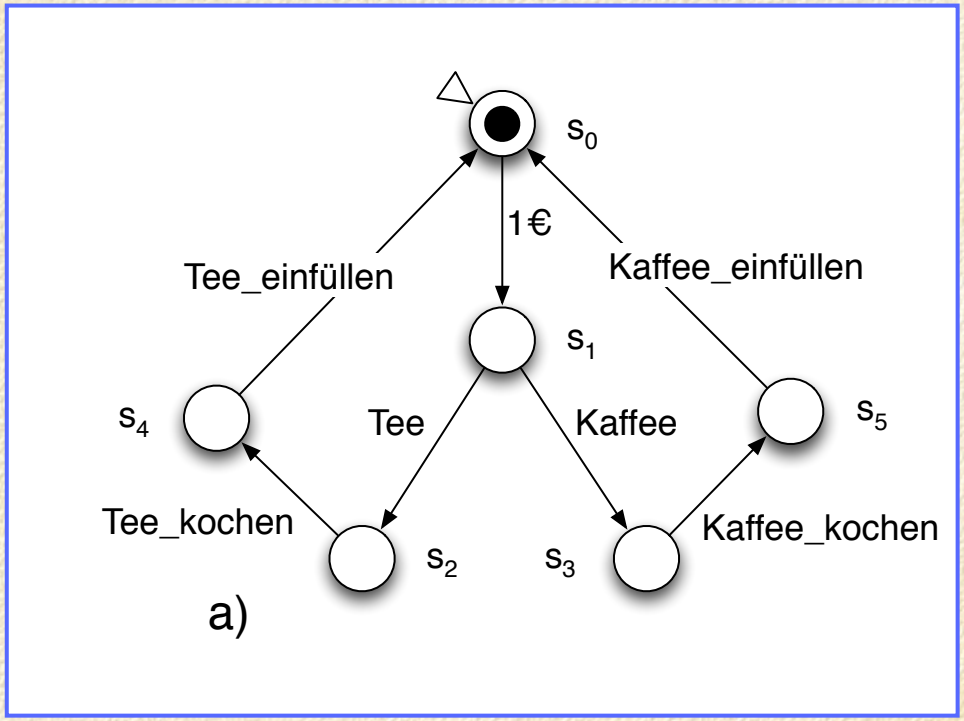
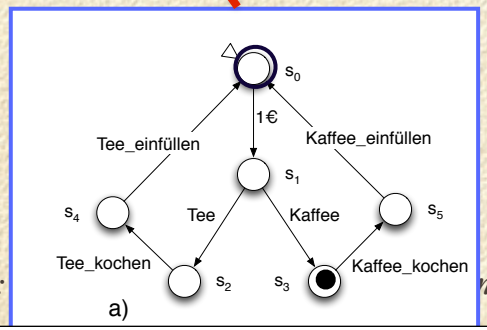
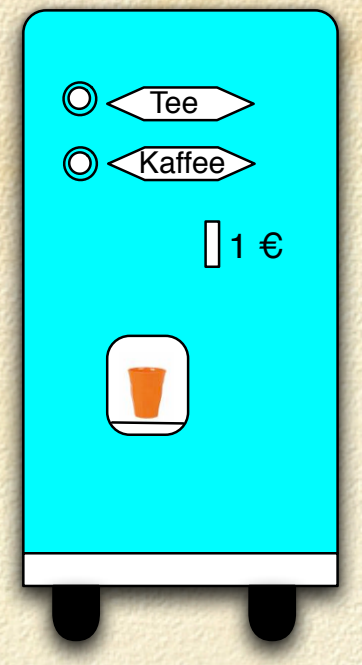


Abbildung 1.1: Getränkeautomat

1€	Kaffee	Kaffee_kochen	Kaffee_einfüllen	#			



dazu:
„akzeptierender Automat“



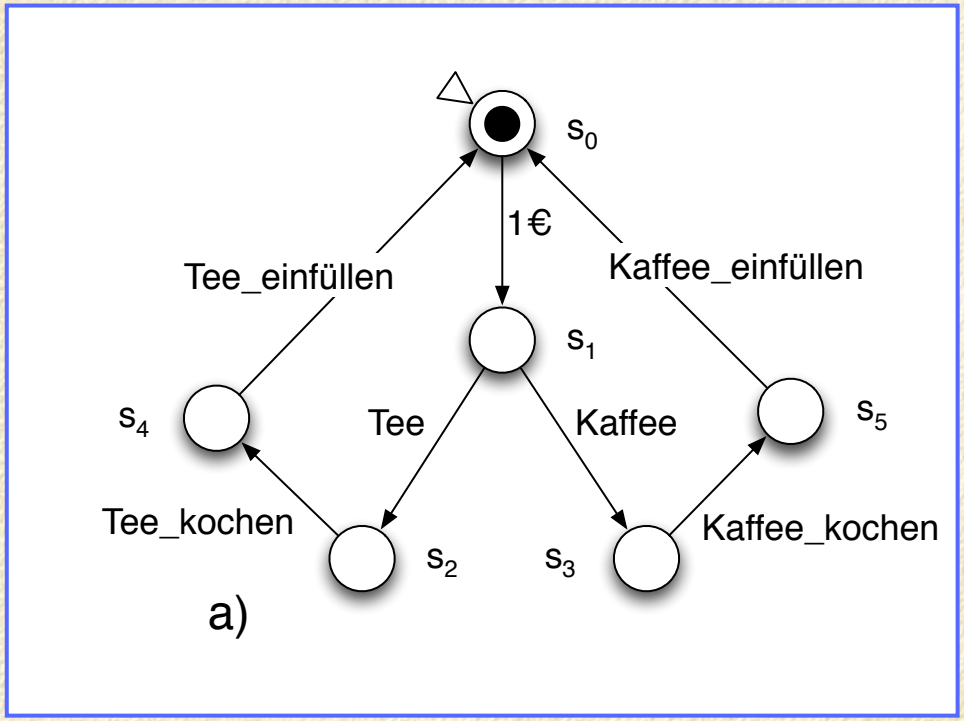
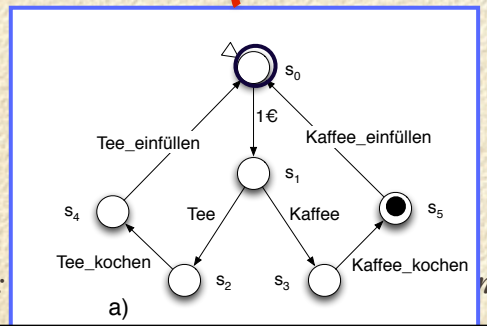
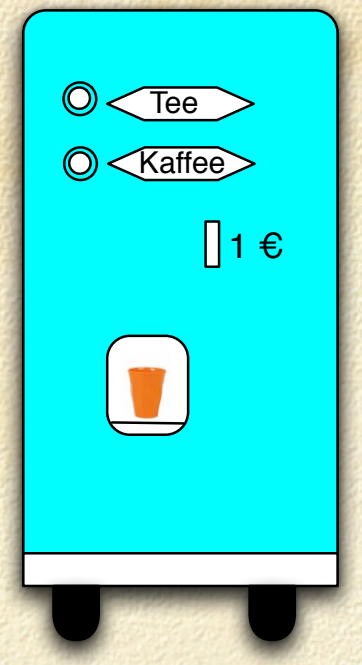


Abbildung 1.1: Getränkeautomat

1€	Kaffee	Kaffee_kochen	Kaffee_einfüllen	#			



dazu:
„akzeptierender Automat“



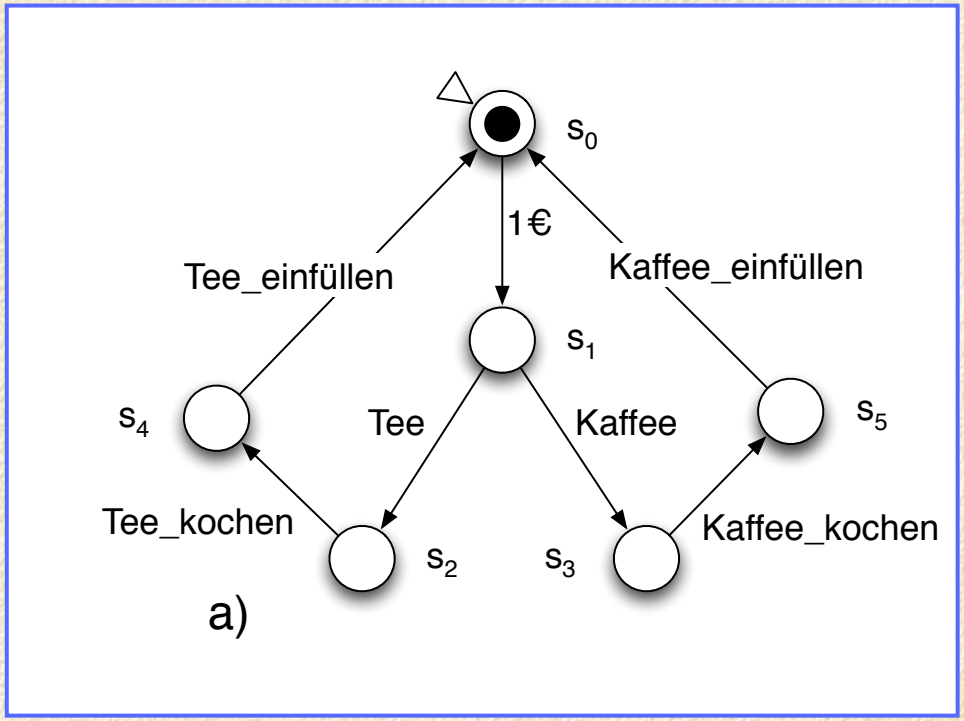
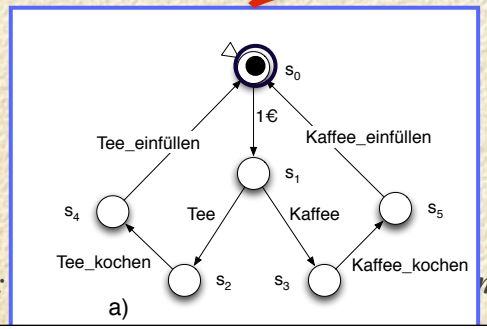
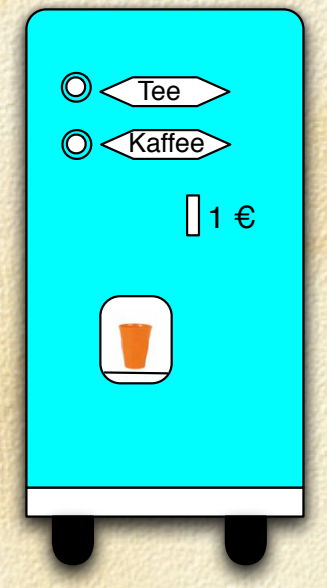


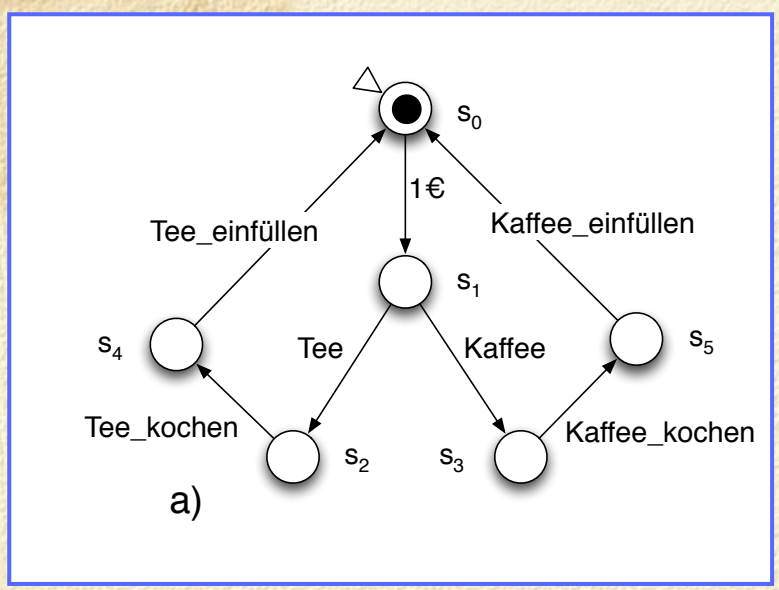
Abbildung 1.1: Getränkeautomat

1€	Kaffee	Kaffee_kochen	Kaffee_einfüllen	#			

dazu:
„akzeptierender Automat“







Definition 1.1 *Ein* Transitionssystem $TS = (S, A, tr, S^0, S^F)$ *besteht aus*

$\mathcal{A} = (Q, \Sigma, \delta, S_0, F)$ *NFA*

a) einer Menge S von Zuständen,

$S = \{s_0, \cdots, s_5\}$

b) einer endlichen Menge A von Aktionen *oder* Transitionen,

$A = \{1€, \text{Tee}, \text{Kaffee}, \text{Tee\_kochen}, \text{Kaffee\_kochen}, \text{Tee\_einfüllen}, \text{Kaffee\_einfüllen}\}$

c) einer Transitionsrelation $tr \subseteq S \times A \times S$,

$tr = \{(s_0, 1€, s_1), (s_1, \text{Tee}, s_2), \cdots\}$

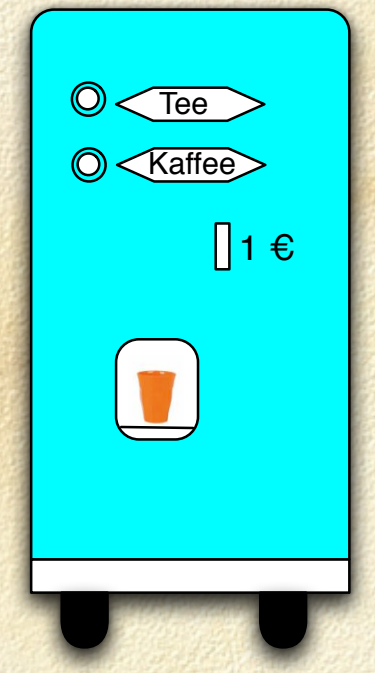
$s_0 \xrightarrow{1€} s_1$

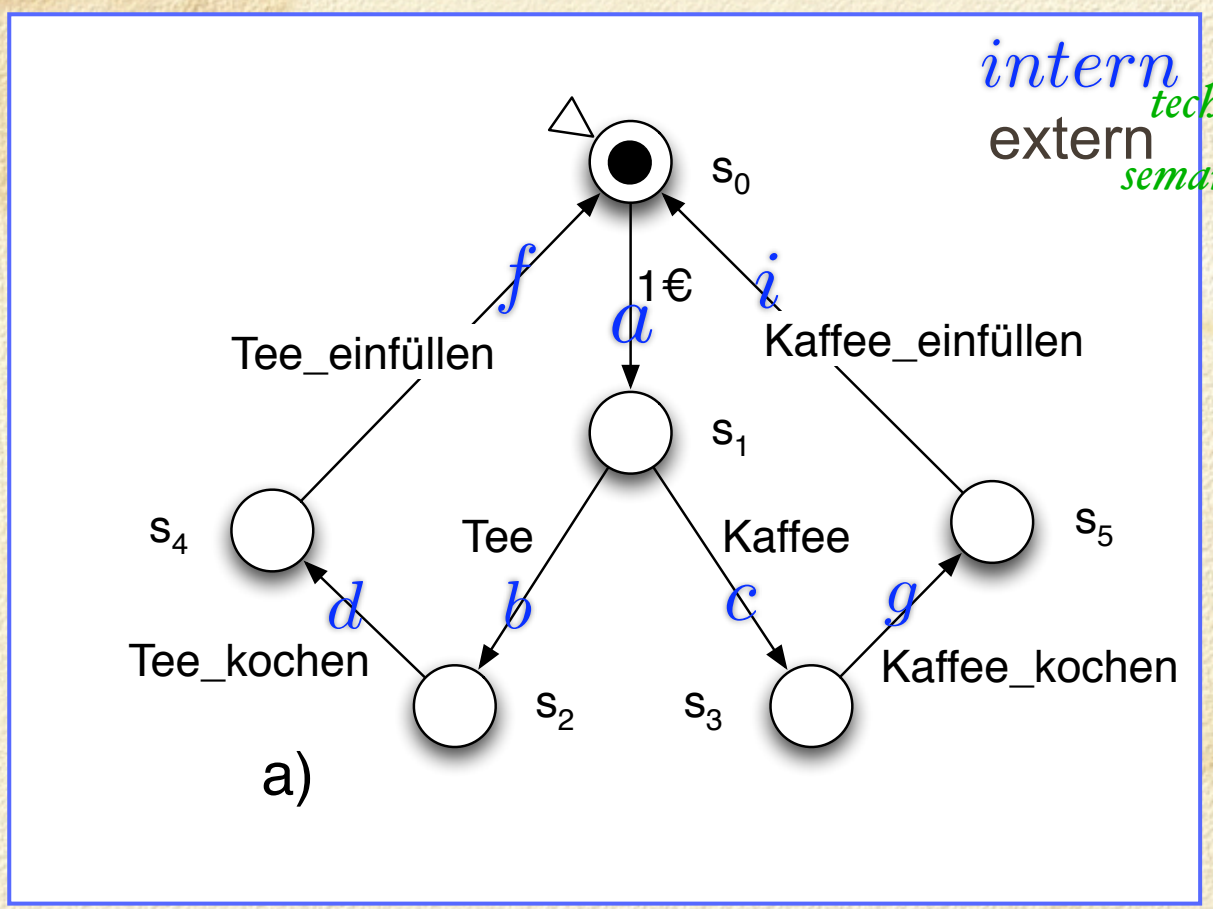
d) einer Menge $S^0 \subseteq S$ von Anfangszuständen *und*

$S^0 = \{s_0\}$

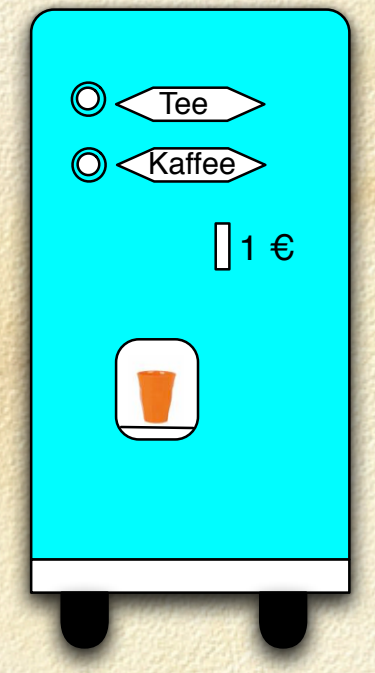
e) einer Menge $S^F \subseteq S$ von Endzuständen,

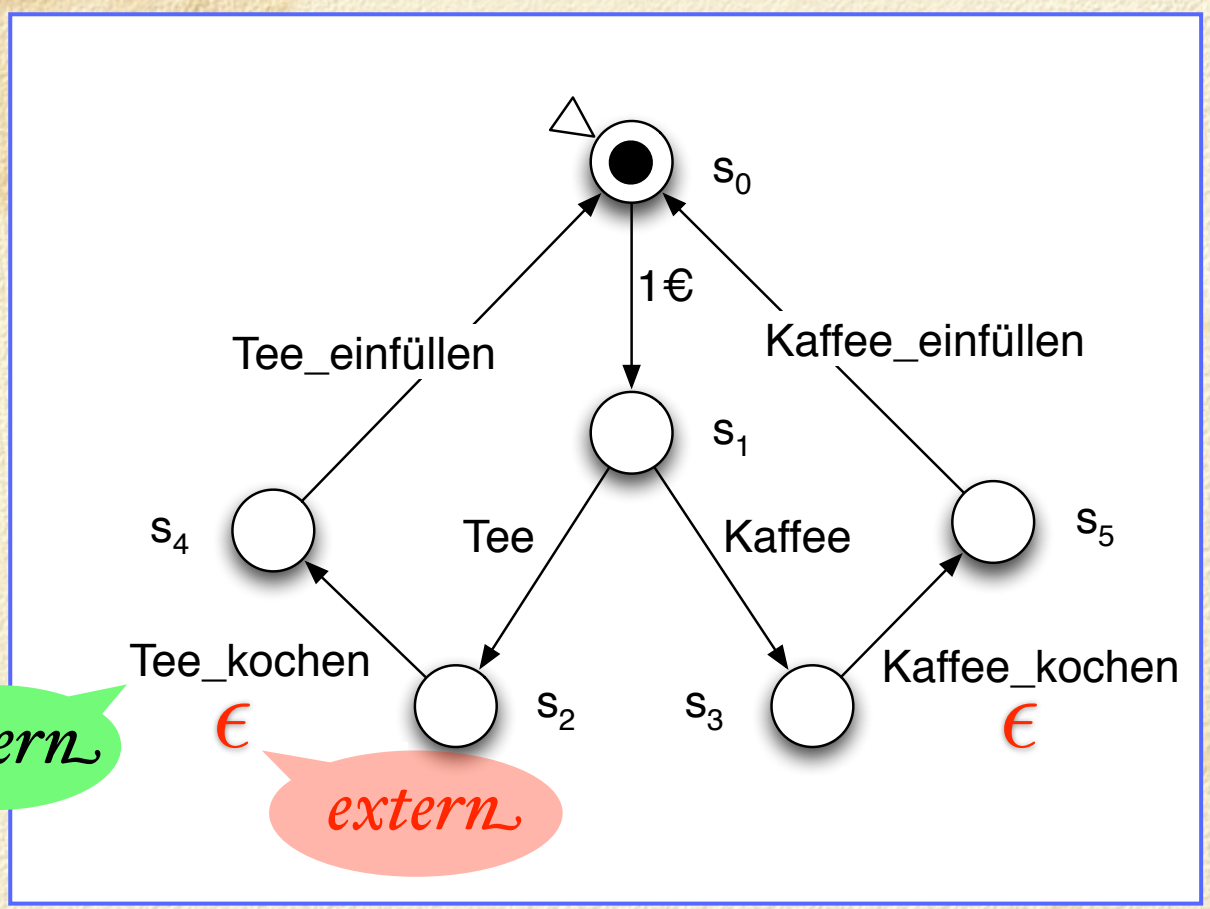
$S^F = \{s_0\}$





Definition 1.2 *Eine* Transitionsetikettenfunktion *eines Transitionssystems* $TS = (S, A, tr, S^0, S^F)$ *ist eine Abbildung* $E_A : A \to \Sigma \cup \{\epsilon\}$. *Dabei ist* Σ *das Etikettenalphabet und* ϵ *das leere Wort* [2].

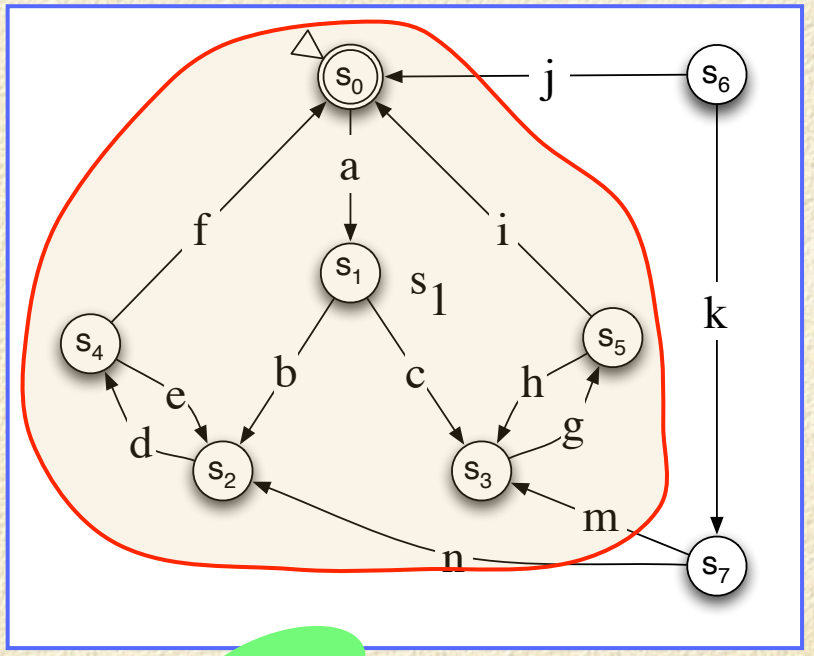




Definition 1.2 *Eine* Transitionsetikettenfunktion *eines Transitionssystems* $TS = (S, A, tr, S^0, S^F)$ *ist eine Abbildung* $E_A : A \to \Sigma \cup \{\epsilon\}$. *Dabei ist* Σ *das Etikettenalphabet und* ϵ *das leere Wort* [2].

mögliche Zustände: *Erreichbarkeitsmenge*

mögliche (terminale) *Aktionsfolgen:* *Sprache*



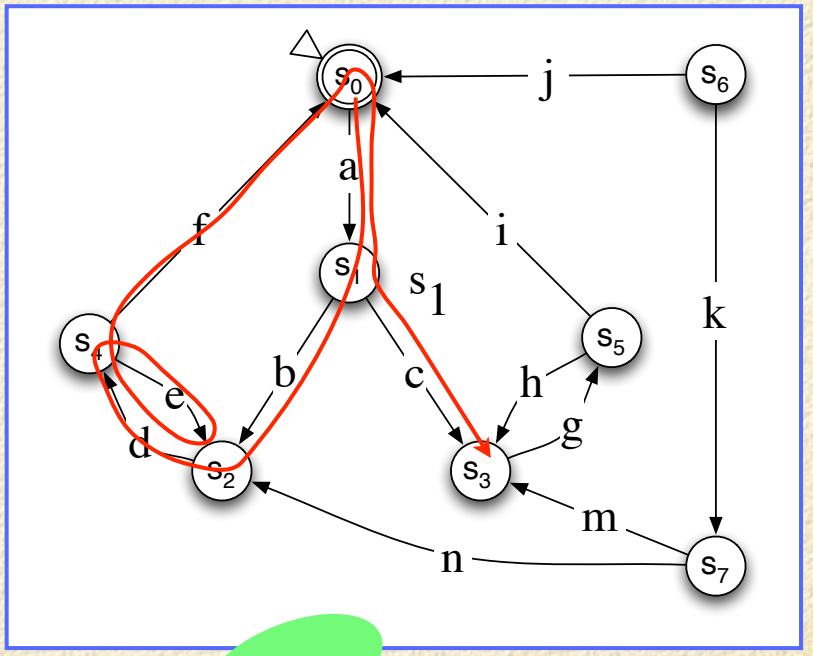
im Beispiel oben: s0

- $R(TS, S_1) := \{s \in S \mid \exists w \in A^*, s_1 \in S_1 : s_1 \xrightarrow{w} s\}$ *als* Menge der von S_1 aus in TS erreichbaren Zustände *und*
- $R(TS) := R(TS, S^0)$ *als* Erreichbarkeitsmenge von TS.

mögliche Zustände: *Erreichbarkeitsmenge*

$$abdedfac \in AS(TS).$$

$$acgiabdededf \in L(TS)$$



Aktions-Sequenzen

- $AS(TS, S_1) := \{w \in A^* \mid \exists s_1 \in S_1, s_1' \in S : s_1 \xrightarrow{w} s_1'\}$ *als* Menge der von S_1 aus in TS möglichen Aktionsfolgen *und*

- $AS(TS) := AS(TS, S^0)$ *als* Aktionsfolgenmenge von TS *definiert.*

Finale Aktions-Sequenzen

- $FS(TS) := \{w \in A^* \mid \exists s_1 \in S_0, s_1' \in S^F : s_1 \xrightarrow{w} s_1'\}$ *ist die* terminale Aktionsfolgenmenge von TS. *Diese Menge heißt oft auch* akzeptable Aktionsfolgenmenge *oder* akzeptierte Sprache *und wird als* $L(TS)$ *bezeichnet.*

mögliche (terminale) Aktionsfolgen: Sprache $E_A(TS)$

Sprachen unendlicher Folgen/Wörter

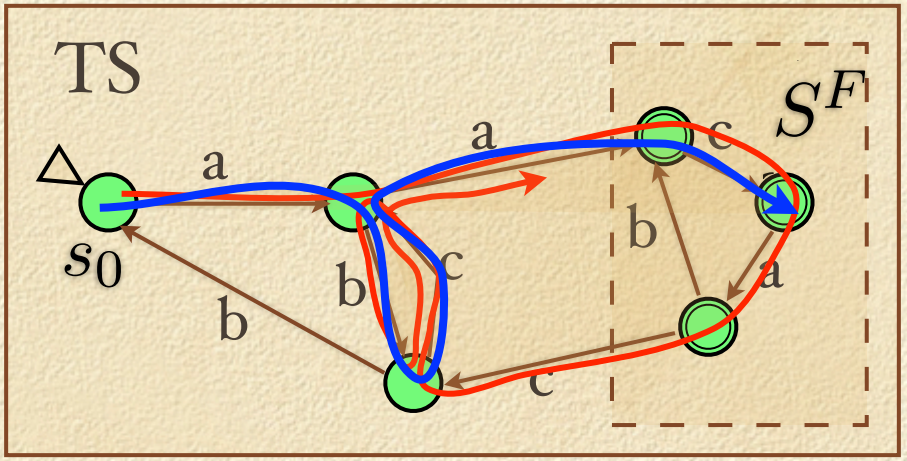
sei A^ω die Menge der unendlichen

Folgen von Zeichen aus A

$w \in A^\omega$ wird als $w = a_0 a_1 a_2 \cdots$ dargestellt.

$L(TS)$ $\quad \text{infinite}(w) := \{a \in A | a \text{ kommt in } w \text{ unendlich oft vor}\}$

$L^\omega(TS) = ?$

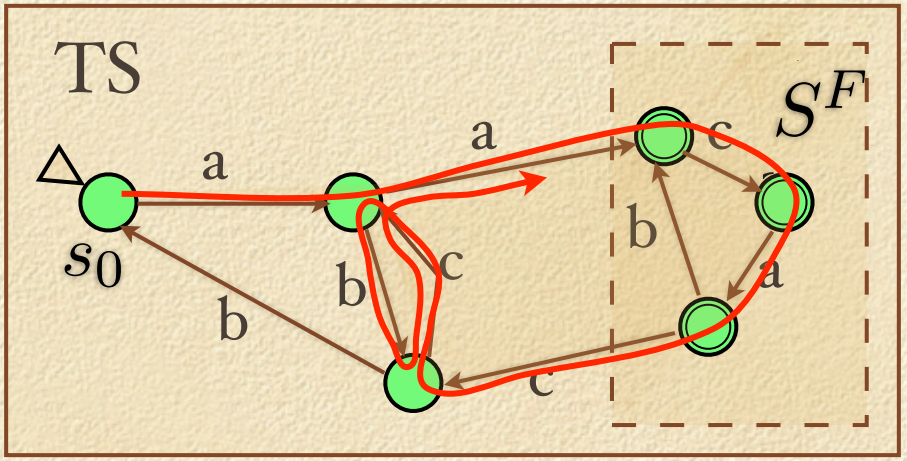


$w = a_0 a_1 a_2 \cdots$ $\quad Pfad\ s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \cdots$

$\text{infinite}(s_0 s_1 s_2 \cdots) \cap S^F \neq \emptyset \Leftrightarrow w$ akzeptabel.

die von TS akzeptierte ω-Sprache

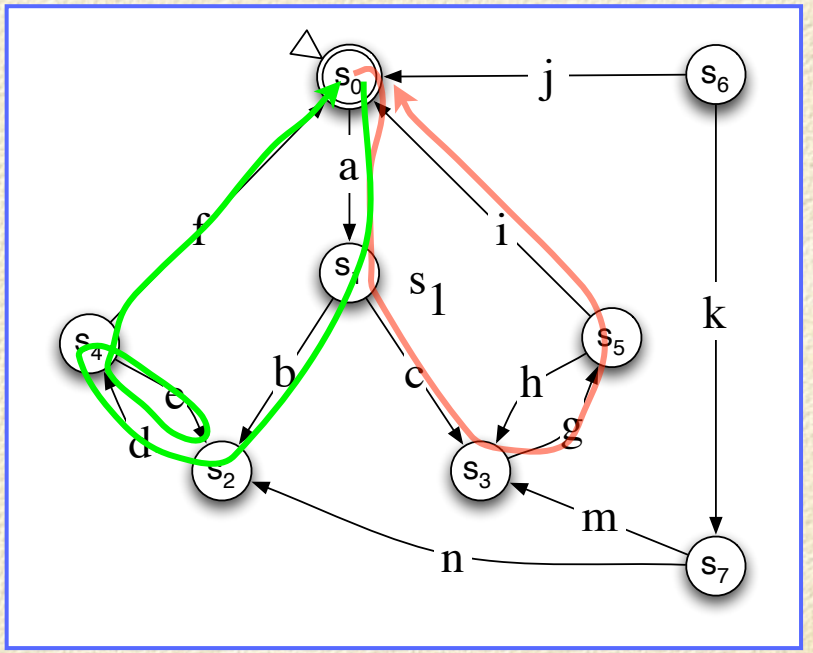
$L^\omega(TS) = ?$



$$abdedef(acgi)^\omega$$

$$\in L^\omega(TS)$$

$$abdededef(acgi)^\omega$$



Zwei Transitionssysteme

$$TS_1 = (S_1, A, tr_1, S_1^0)$$

$$TS_1 = (S_1, A, tr_1, S_1^0)$$

„folgenäquivalent“, falls

$$TS_1 \sim_{AS} TS_2 :\Leftrightarrow AS(TS_1) = AS(TS_2)$$

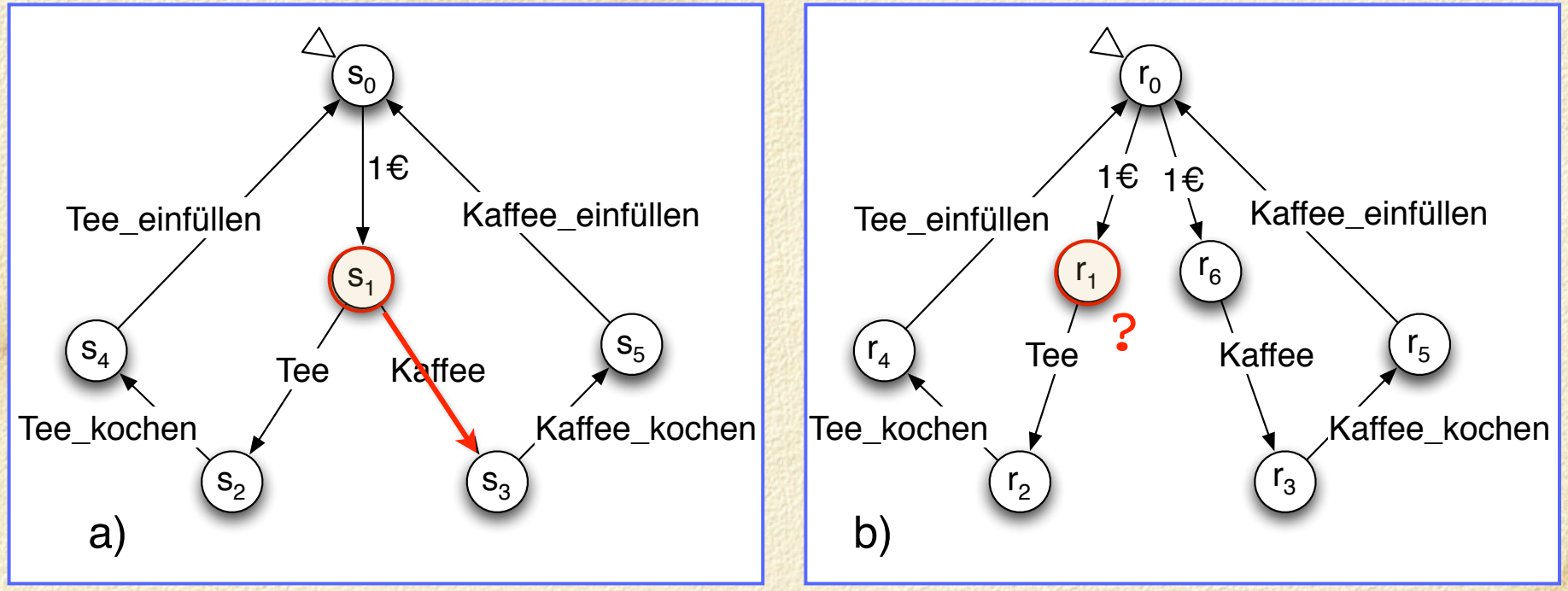


Abbildung 1.4: Zwei folgenäquivalente Transitionssysteme

Haben die beiden Systeme tatsächlich dasselbe „Verhalten“?

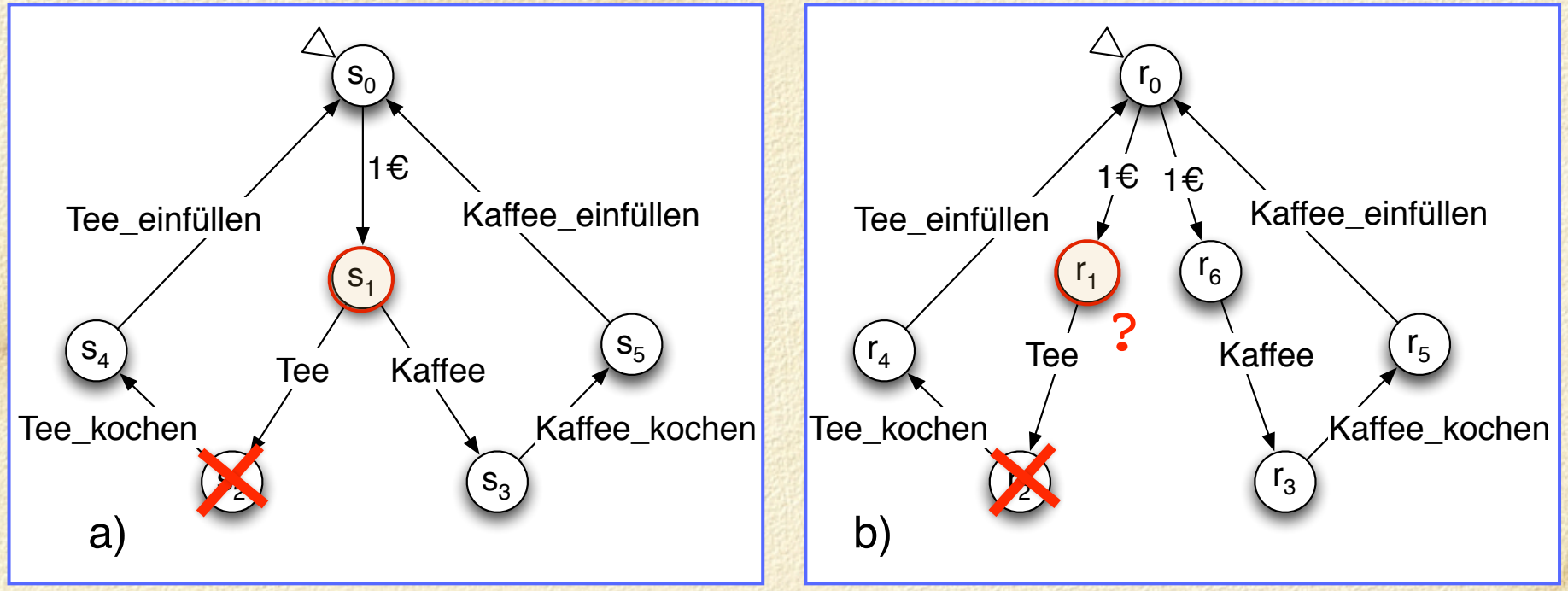


Abbildung 1.4: Zwei folgenäquivalente Transitionssysteme

Haben die beiden Systeme tatsächlich dasselbe „Verhalten“?

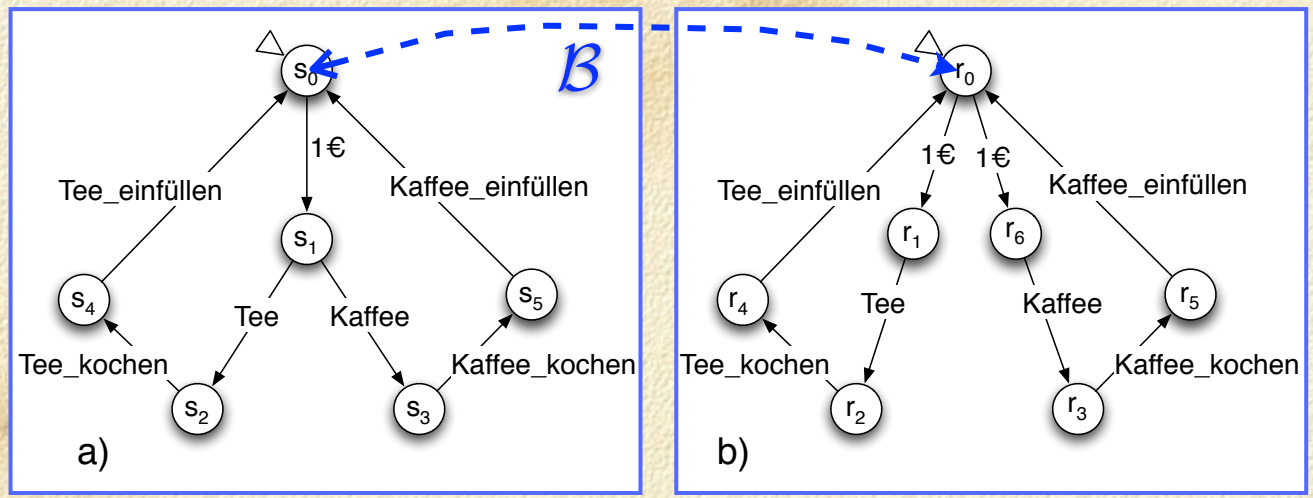


Abbildung 1.4: Zwei folgenäquivalente Transitionssysteme

Definition 1.7 *Gegeben seien für $i \in \{1,2\}$ zwei Transitionssysteme $TS_i = (S_i, A, tr_i, S_i^0, S_i^F)$ mit Endzuständen (und der gleichen Aktionsmenge A) . Eine (aktionsbasierte)* Bisimulation *für (TS_1, TS_2) ist eine binäre Relation $\mathcal{B} \subseteq S_1 \times S_2$, für die folgendes gilt:*

$$a)\ \forall s_0 \in S_1^0\ \exists r_0 \in S_2^0 : (s_0, r_0) \in \mathcal{B}$$
$$\forall r_0 \in S_2^0\ \exists s_0 \in S_1^0 : (s_0, r_0) \in \mathcal{B}$$

$$c)\ \forall (s,r) \in \mathcal{B} : s \in S_1^F \Leftrightarrow r \in S_2^F$$

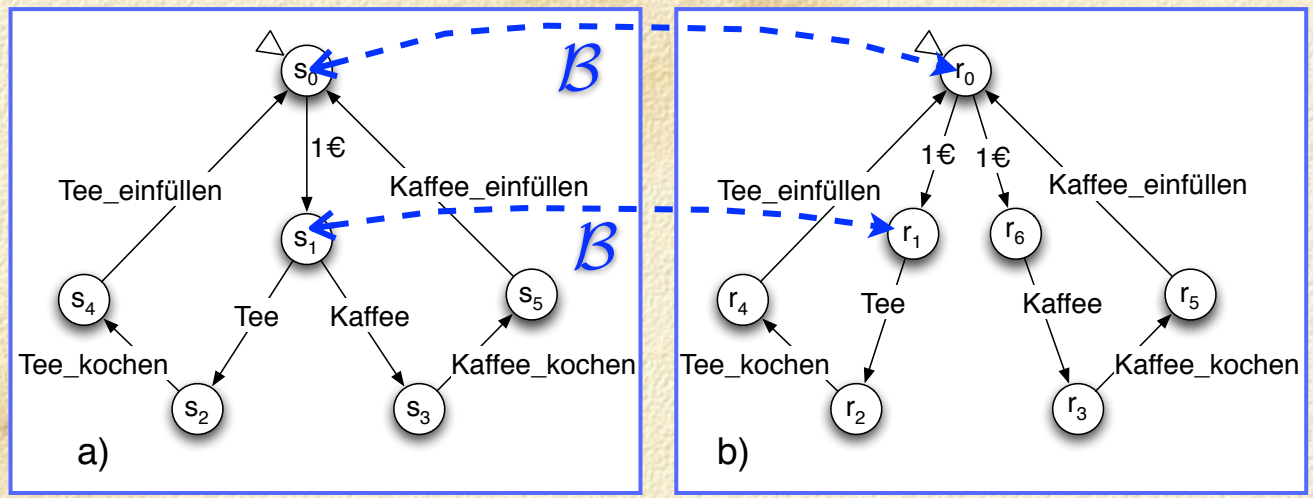


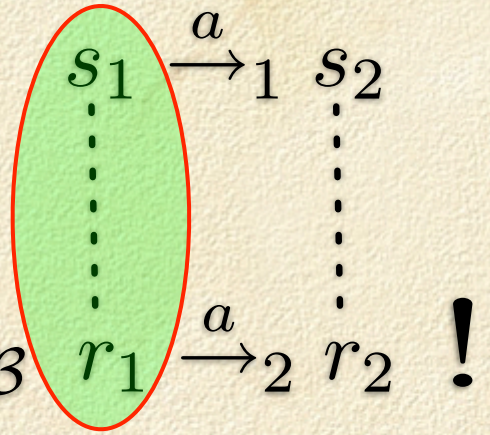
Abbildung 1.4: Zwei folgenäquivalente Transitionssysteme

Definition 1.7 *Gegeben seien für* $i \in \{1,2\}$ *zwei Transitionssysteme* $TS_i = (S_i, A, tr_i, S_i^0, S_i^F)$ *mit Endzuständen (und der gleichen Aktionsmenge* A*)*[5]*. Eine (aktionsbasierte)* Bisimulation *für* (TS_1, TS_2) *ist eine binäre Relation* $\mathcal{B} \subseteq S_1 \times S_2$*, für die folgendes gilt:*

b) Für alle $(s_1, r_1) \in \mathcal{B}$ *gilt:*

$$s_1 \xrightarrow{a}_1 s_2 \Rightarrow \exists r_2 \in S_2 : r_1 \xrightarrow{a}_2 r_2 \wedge (s_2, r_2) \in \mathcal{B}$$

$$r_1 \xrightarrow{a}_2 r_2 \Rightarrow \exists s_2 \in S_1 : s_1 \xrightarrow{a}_1 s_2 \wedge (s_2, r_2) \in \mathcal{B}$$



TS_1 und TS_2 heißen bisimilar *(in Zeichen $TS_1 \underline{\leftrightarrow} TS_2$) falls eine solche Bisimulationsrelation $\mathcal{B}$ existiert. Zustände mit $(s, r) \in \mathcal{B}$ heißen* bisimilar *(in Zeichen $s \underline{\leftrightarrow} r$).*

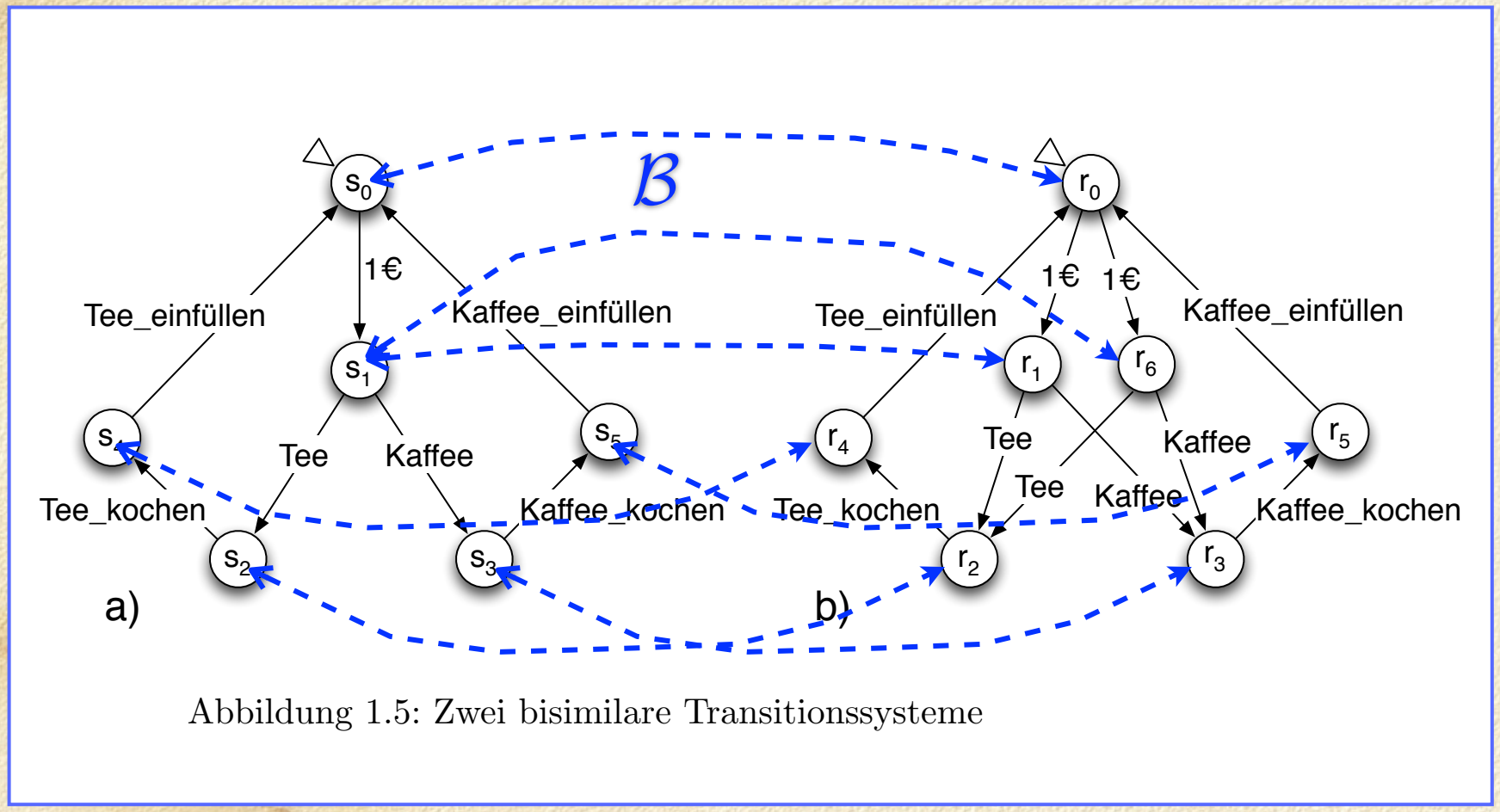


Abbildung 1.5: Zwei bisimilare Transitionssysteme

Satz 1.8 *Wenn zwei Transitionssysteme $TS_i = (S_i, A, tr_i, S_i^0, S_i^F)$ mit Endzuständen (und der gleichen[7] Aktionsmenge A und $i \in \{1, 2\}$) bisimilar sind, dann sind sie auch akzeptanzäquivalent, aber nicht umgekehrt,*
d.h.: $TS_1 \leftrightarrows TS_2 \Rightarrow TS_1 \sim_L TS_2$).

zum Beweis: vollständige Induktion über die Wortlänge

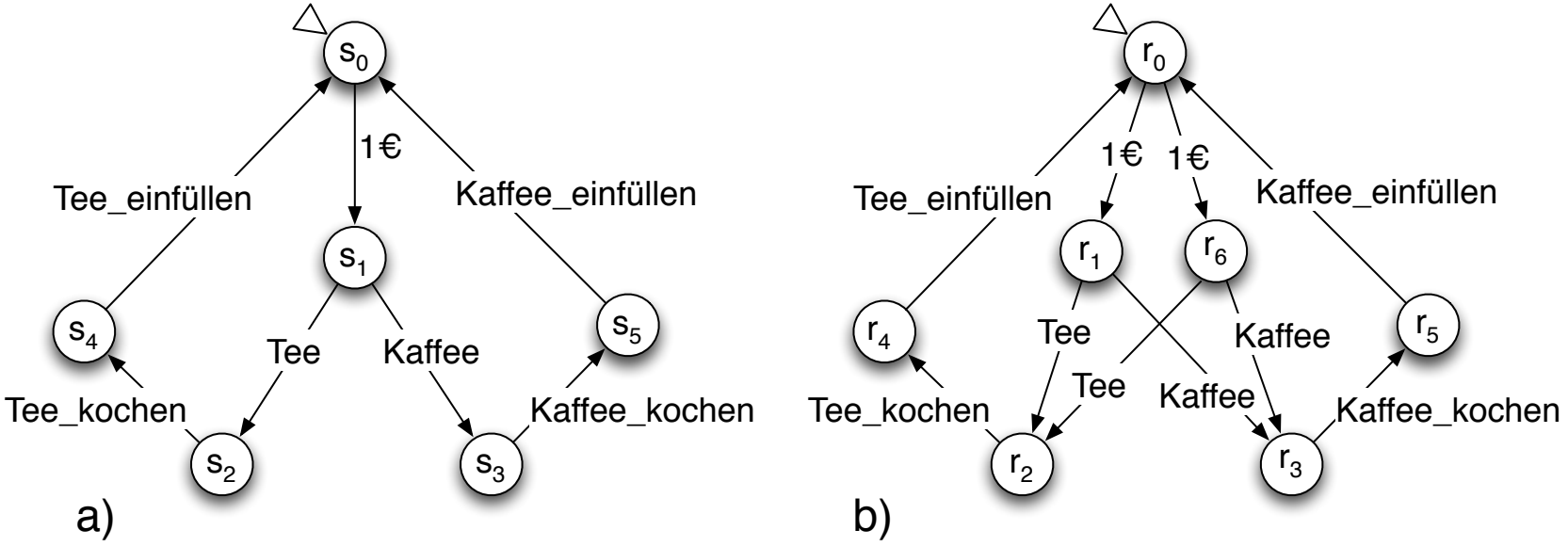


Abbildung 1.5: Zwei bisimilare Transitionssysteme

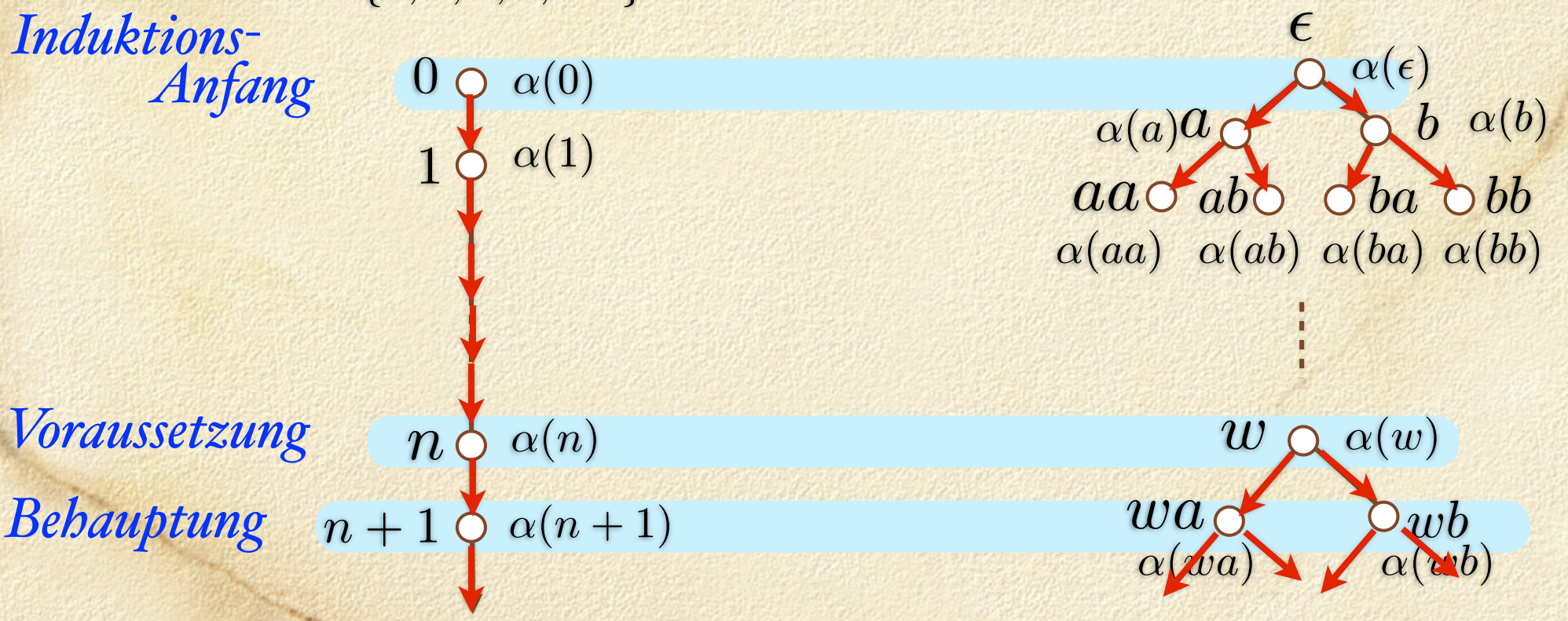
Satz 1.8 *Wenn zwei Transitionssysteme $TS_i = (S_i, A, tr_i, S_i^0, S_i^F)$ mit Endzuständen (und der gleichen[7] Aktionsmenge A und $i \in \{1,2\}$) bisimilar sind, dann sind sie auch akzeptanzäquivalent, aber nicht umgekehrt,*
d.h.: $TS_1 \leftrightarrows TS_2 \Rightarrow TS_1 \sim_L TS_2$).

zum Beweis einer Eigenschaft $\alpha(n)$ *bzw* $\alpha(w)$*:*

vollständige Induktion über:

$$\mathbb{N} = \{0, 1, 2, 3, \cdots\}$$

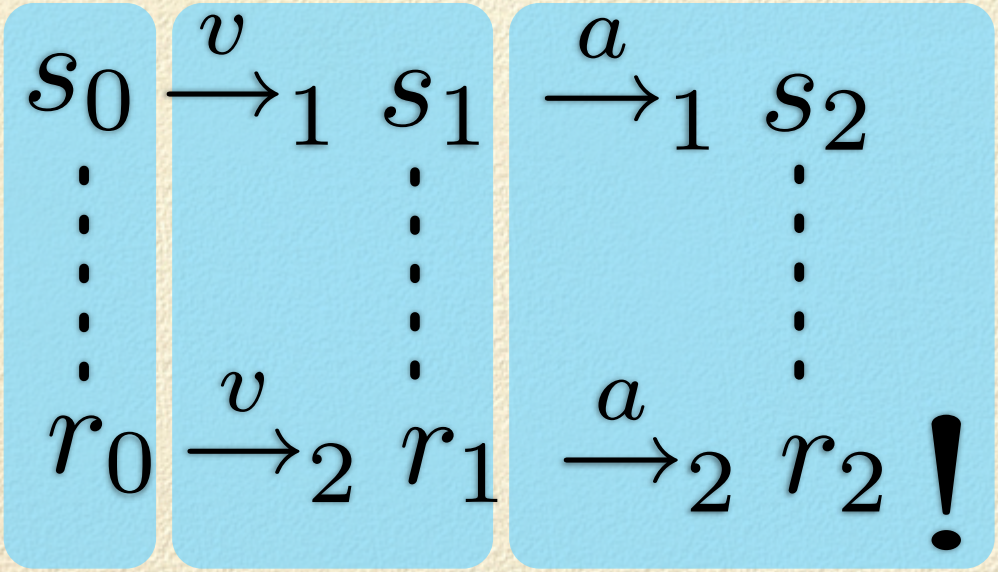
$$A^* = \{a, b\}^*$$



Satz 1.8 *Wenn zwei Transitionssysteme $TS_i = (S_i, A, tr_i, S_i^0, S_i^F)$ mit Endzuständen (und der gleichen[7] Aktionsmenge A und $i \in \{1,2\}$) bisimilar sind, dann sind sie auch akzeptanzäquivalent, aber nicht umgekehrt,*
d.h.: $TS_1 \leftrightarrows TS_2 \Rightarrow TS_1 \sim_L TS_2$).

zum Beweis: ~vollständige Induktion über die Wortlänge

$$v \in A^* \qquad a \in A$$



Induktions-

Anfang *Voraussetzung* *Behauptung*

Satz 1.8 *Wenn zwei Transitionssysteme $TS_i = (S_i, A, tr_i, S_i^0, S_i^F)$ mit Endzuständen (und der gleichen[7] Aktionsmenge A und $i \in \{1, 2\}$) bisimilar sind, dann sind sie auch akzeptanzäquivalent, aber nicht umgekehrt,*
d.h.: $TS_1 \leftrightarrows TS_2 \Rightarrow TS_1 \sim_L TS_2)$.

$\Leftarrow$?

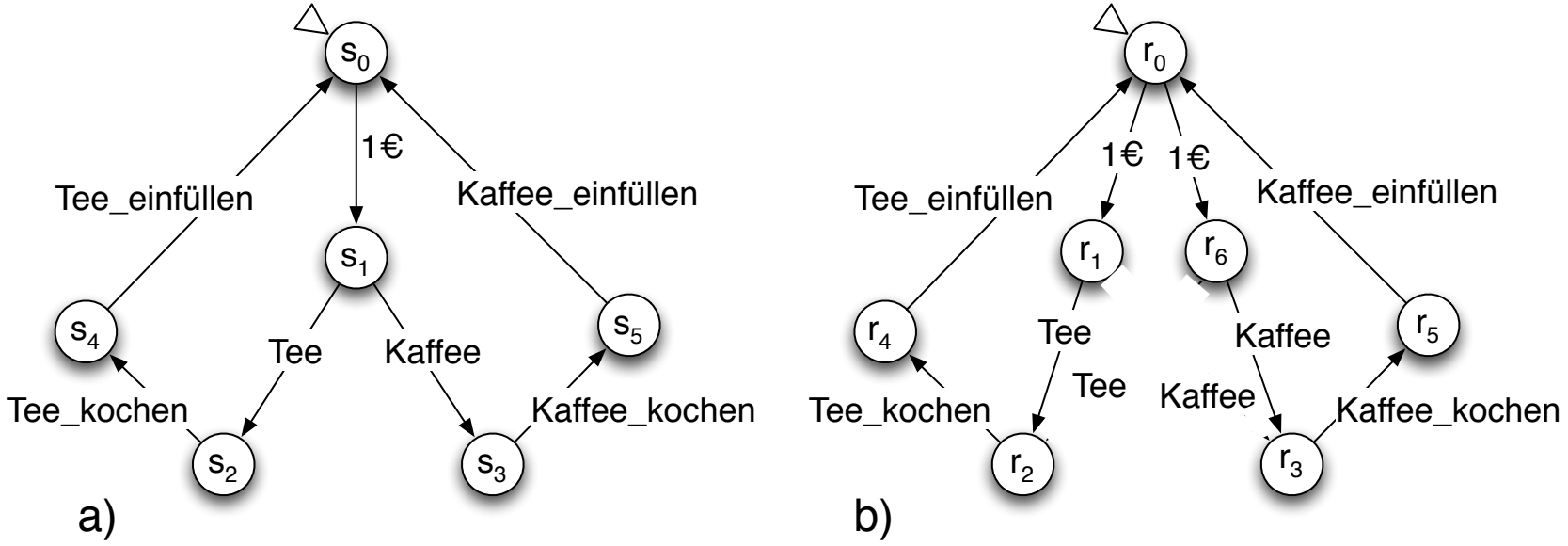
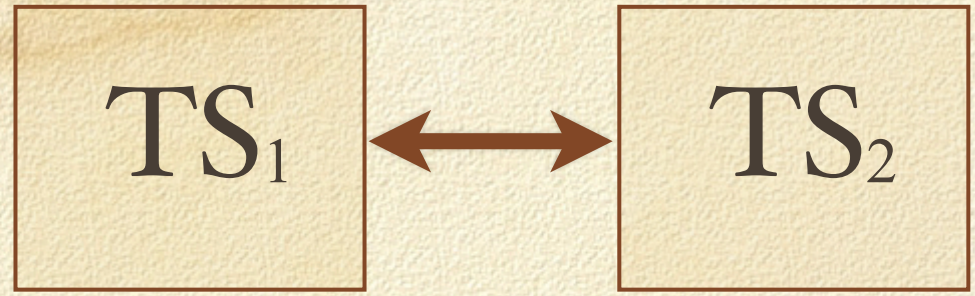


Abbildung 1.5: Zwei bisimilare Transitionssysteme

nicht

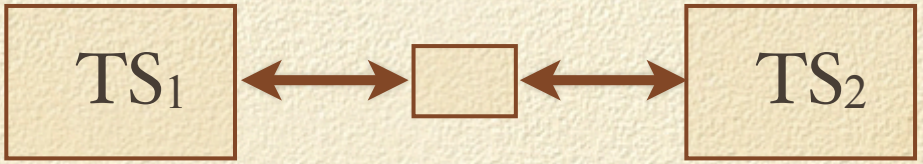
1.2 Produkte von Transitionssystemen



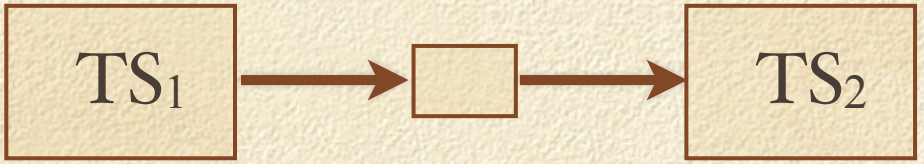
a) *Rendezvous-Synchronisation*

handshake synchronisation

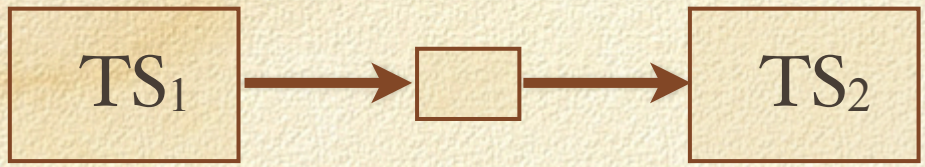
b) *Speicher-Synchronisation*

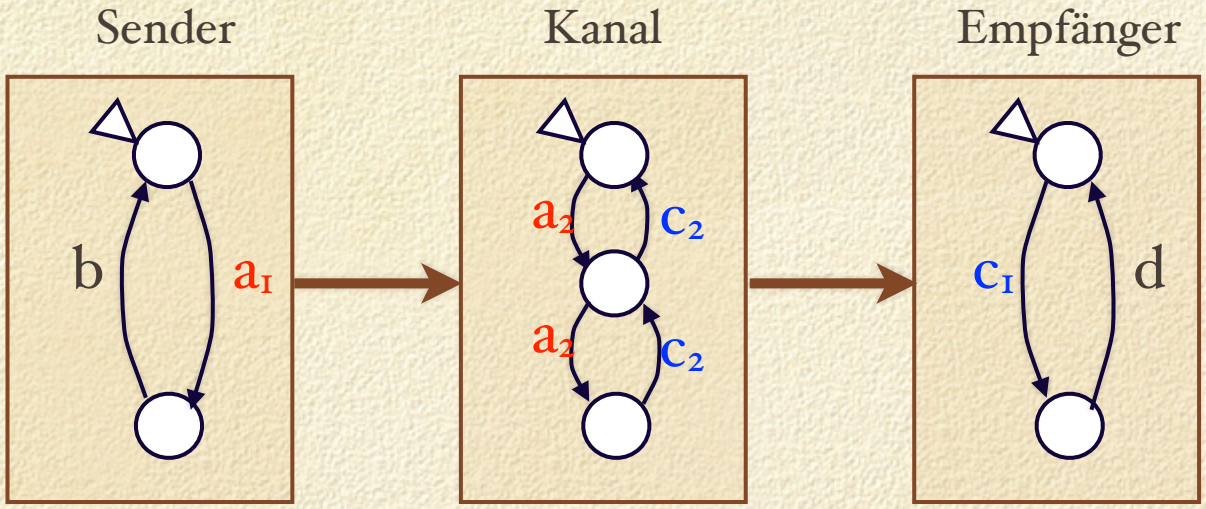


c) *Nachrichten-Synchronisation*



c) *Nachrichten-Synchronisation*

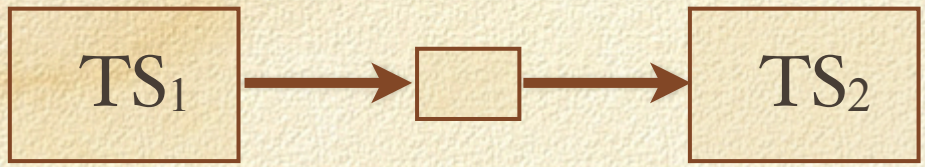


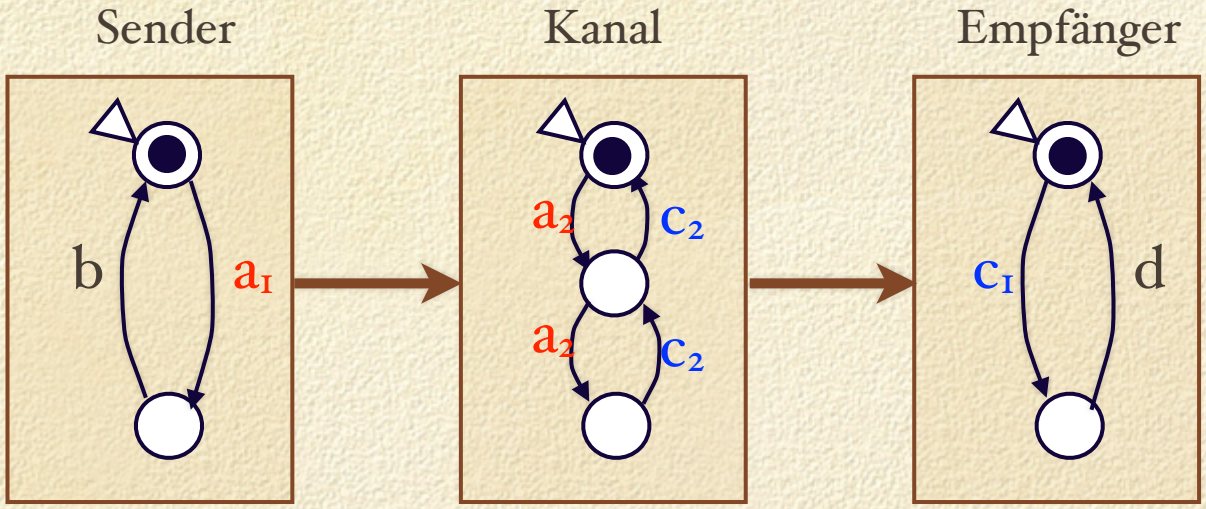


$$Sync = \{(a_1,a_2),(c_1,c_2)\}$$

$$\gamma(a_1,a_2) = a, \quad \gamma(c_1,c_2) = c$$

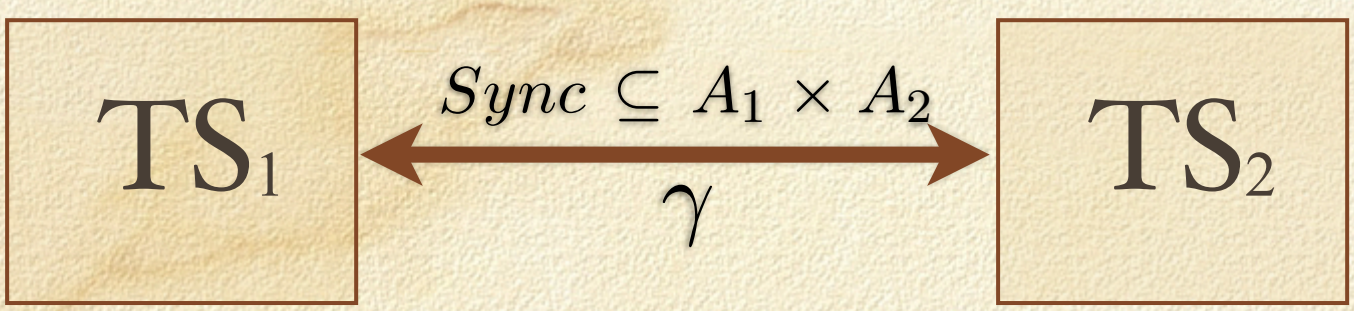
c) *Nachrichten-Synchronisation*





$$Sync = \{(a_1,a_2),(c_1,c_2)\}$$

$$\gamma(a_1,a_2) = a,\ \ \gamma(c_1,c_2) = c\}$$



Definition 1.9 *Gegeben seien zwei Transitionssysteme* $TS_i = (S_i, A_i, tr_i, S_i^0, S_i^F)$ $(i \in \{1,2\})$ *(mit nicht notwendig gleichen Aktionsmengen* A_i*). Das* Produkttransitionssystem *von* TS_1 *und* TS_2 *wird als Transitionssystem*

$$TS_1 \otimes_\gamma TS_2 = (S_1 \times S_2, A_1 \cup A_2 \cup Sync, tr_3, S_1^0 \times S_2^0, S_1^F \times S_2^F, \gamma)$$

definiert, wobei

A_3

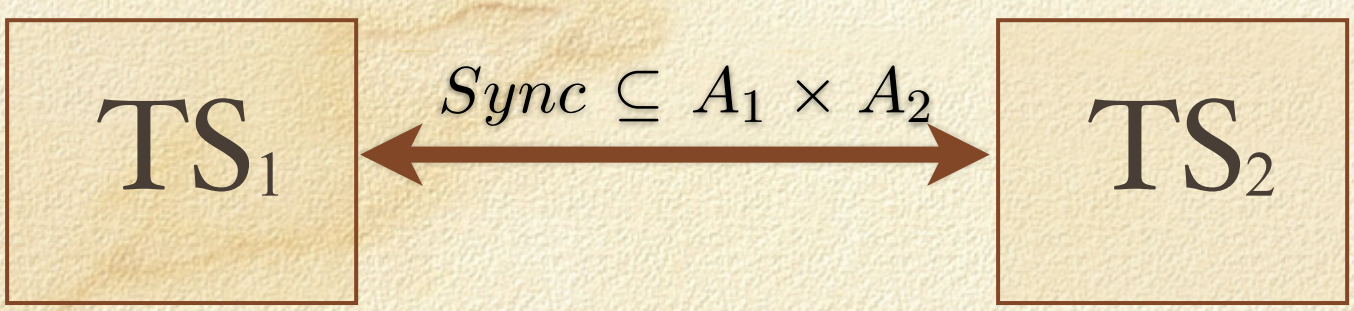
a) $Sync \subseteq A_1 \times A_2$ *eine* Synchronisations-Relation,

b) $\gamma : Sync \to A_3$ *eine Abbildung von Sync in ein* Kommunikationsalphabet[7] A_3 *ist und*

Die Kommunikationsabbildung γ *ist optional.*

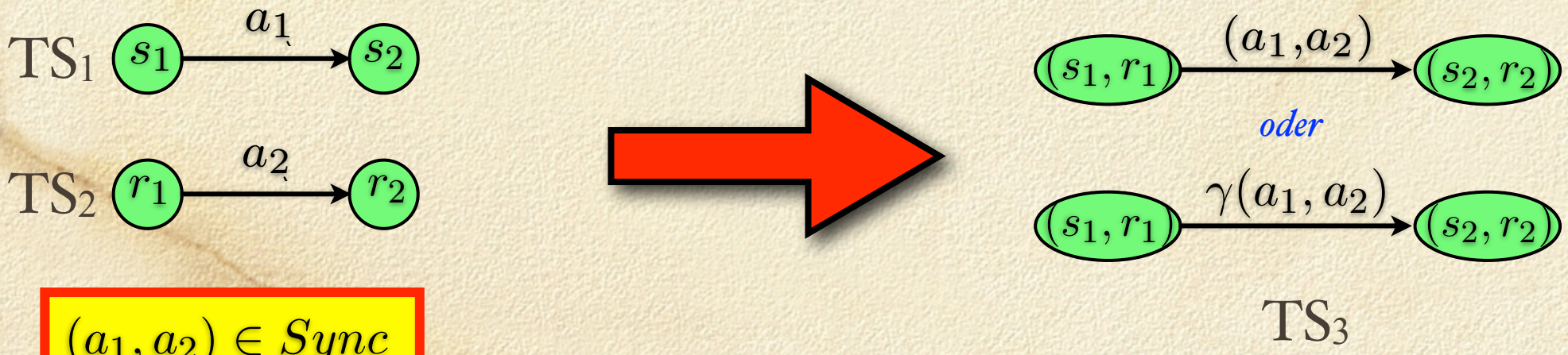
Transitionen

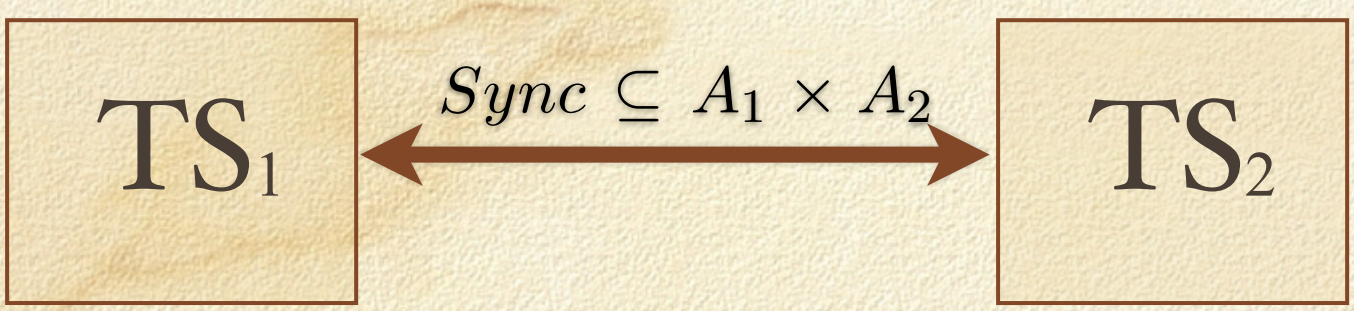
durch Regeln



Regel $Sy3$: Kommunikation

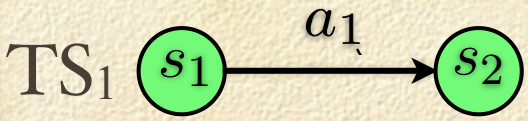
$$\frac{s_1 \xrightarrow{a_1}_1 s_2, \quad r_1 \xrightarrow{a_2}_2 r_2, \quad (a_1, a_2) \in Sync}{(s_1, r_1) \xrightarrow{(a_1,a_2)}_3 (s_2, r_2)}$$

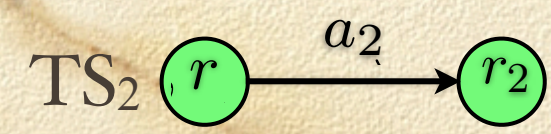


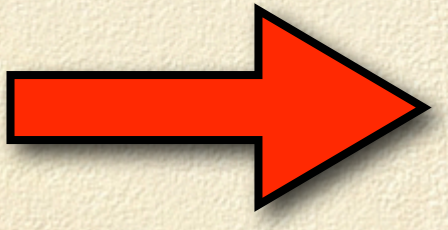


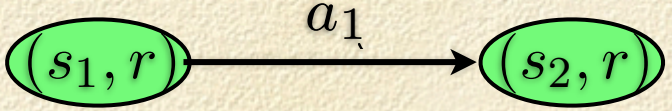
Regel Sy1: Erste Komponente

$$\frac{s_1 \xrightarrow{a_1}_1 s_2, \quad a_1 \notin pr_1(Sync), \quad r \in S_2}{(s_1, r) \xrightarrow{a_1}_3 (s_2, r)}$$



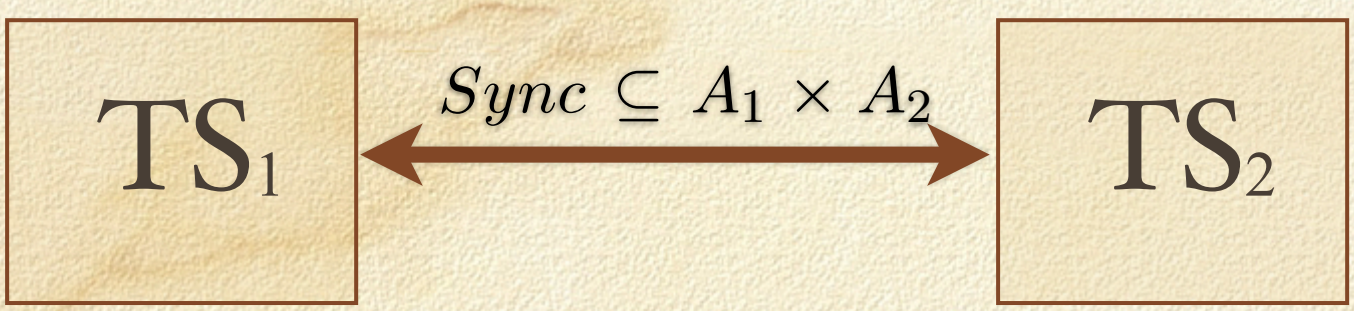






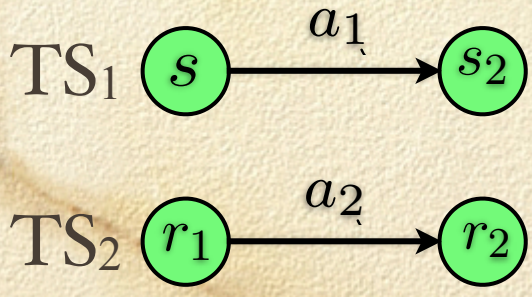
TS3

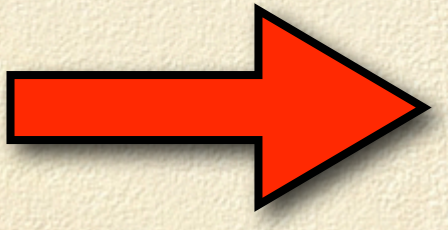
$a_1 \notin pr_1(Sync)$

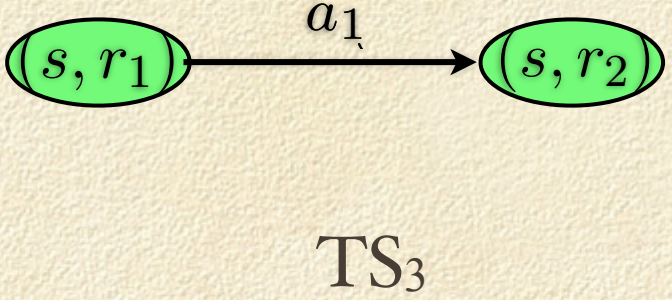


Regel $Sy2$: Zweite Komponente

$$\frac{r_1 \xrightarrow{a_2}_2 r_2,\ a_2 \notin pr_2(Sync), \quad s \in S_1}{(s, r_1) \xrightarrow{a_2}_3 (s, r_2)}$$







$a_2 \notin pr_2(Sync)$

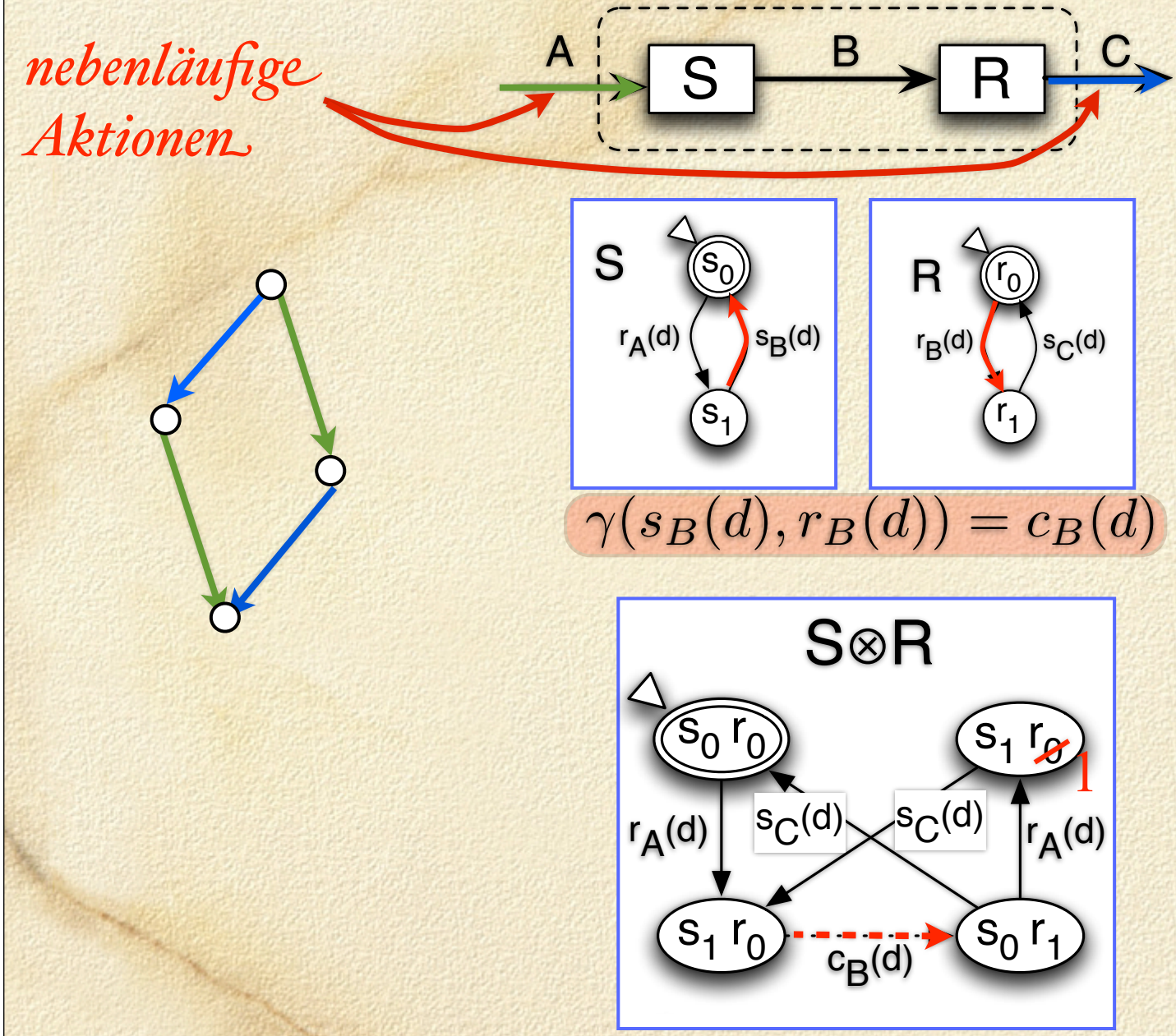


Abbildung 1.6: Ein einfaches Sender-Empfänger-System

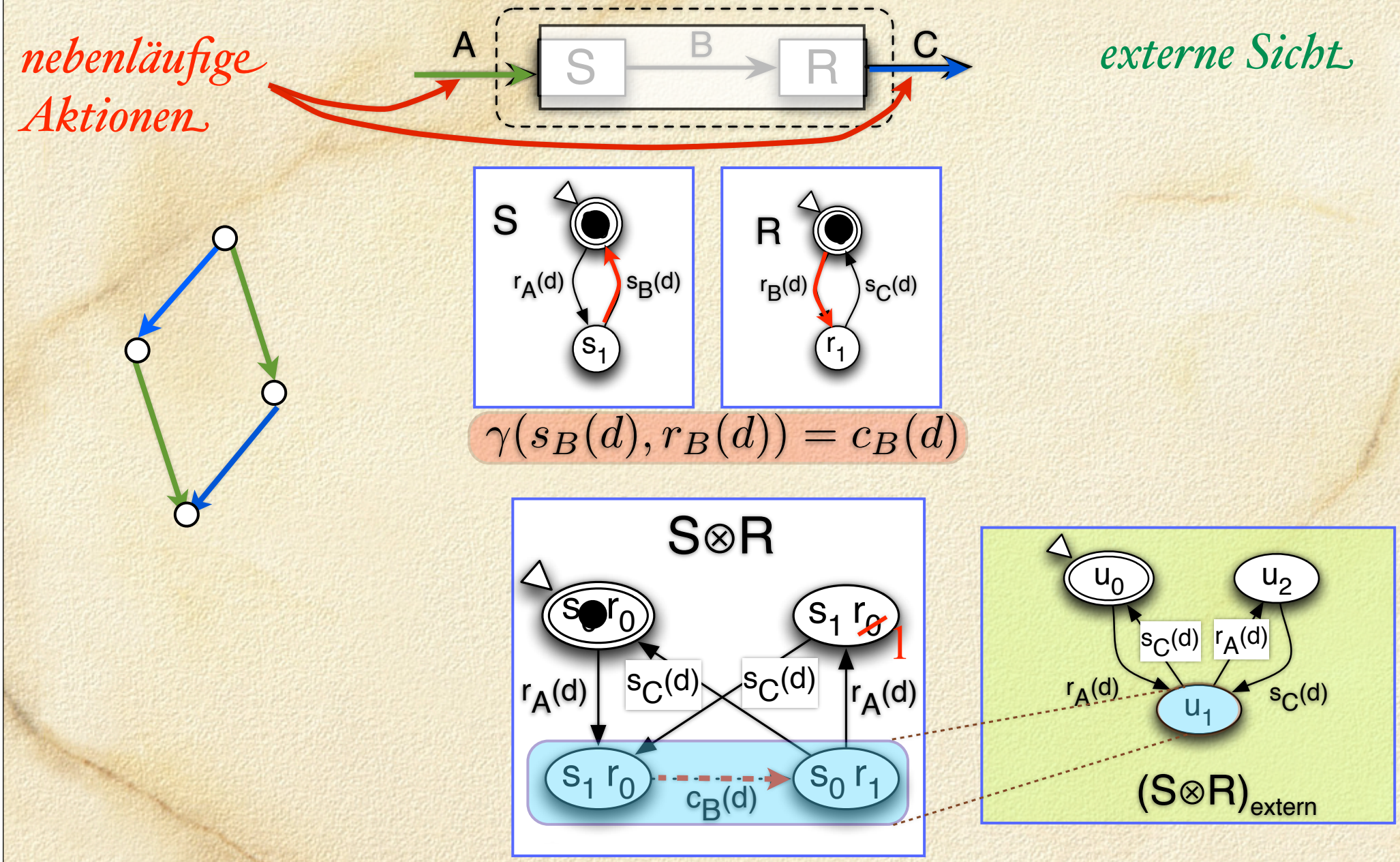
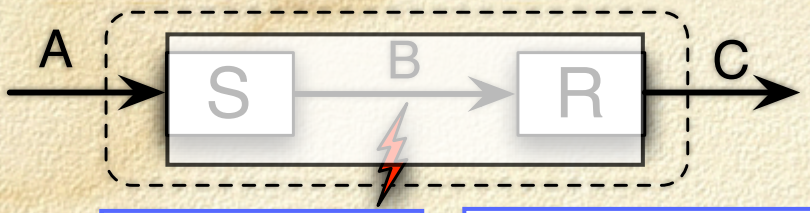
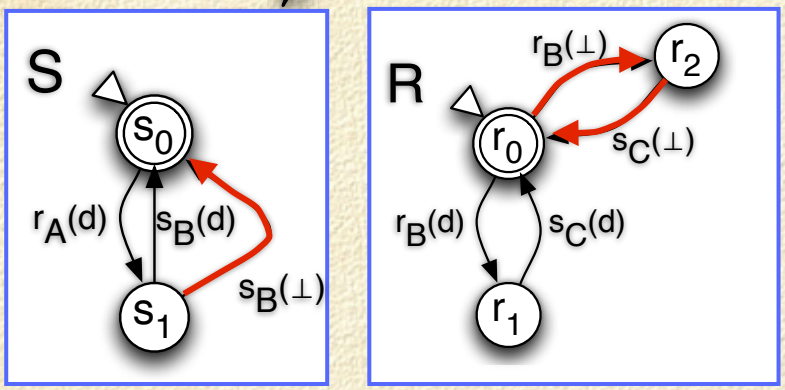


Abbildung 1.6: Ein einfaches Sender-Empfänger-System



externe Sicht



$$\gamma(s_B(d), r_B(d)) = c_B(d)$$
$$\gamma(s_B(\perp), r_B(\perp)) = c_B(\perp)$$

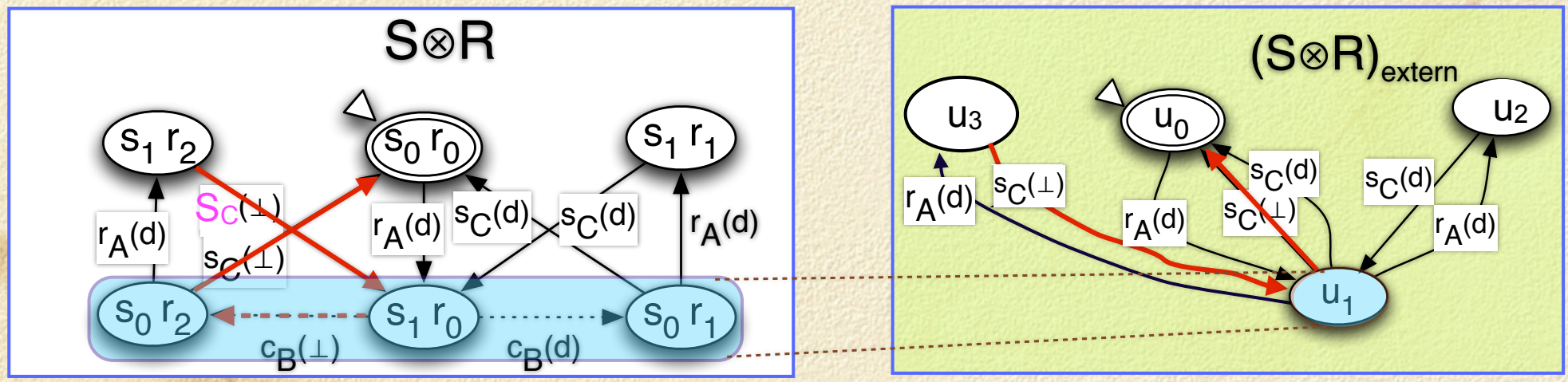


Abbildung 1.7: Ein gestörtes Sender-Empfänger-System

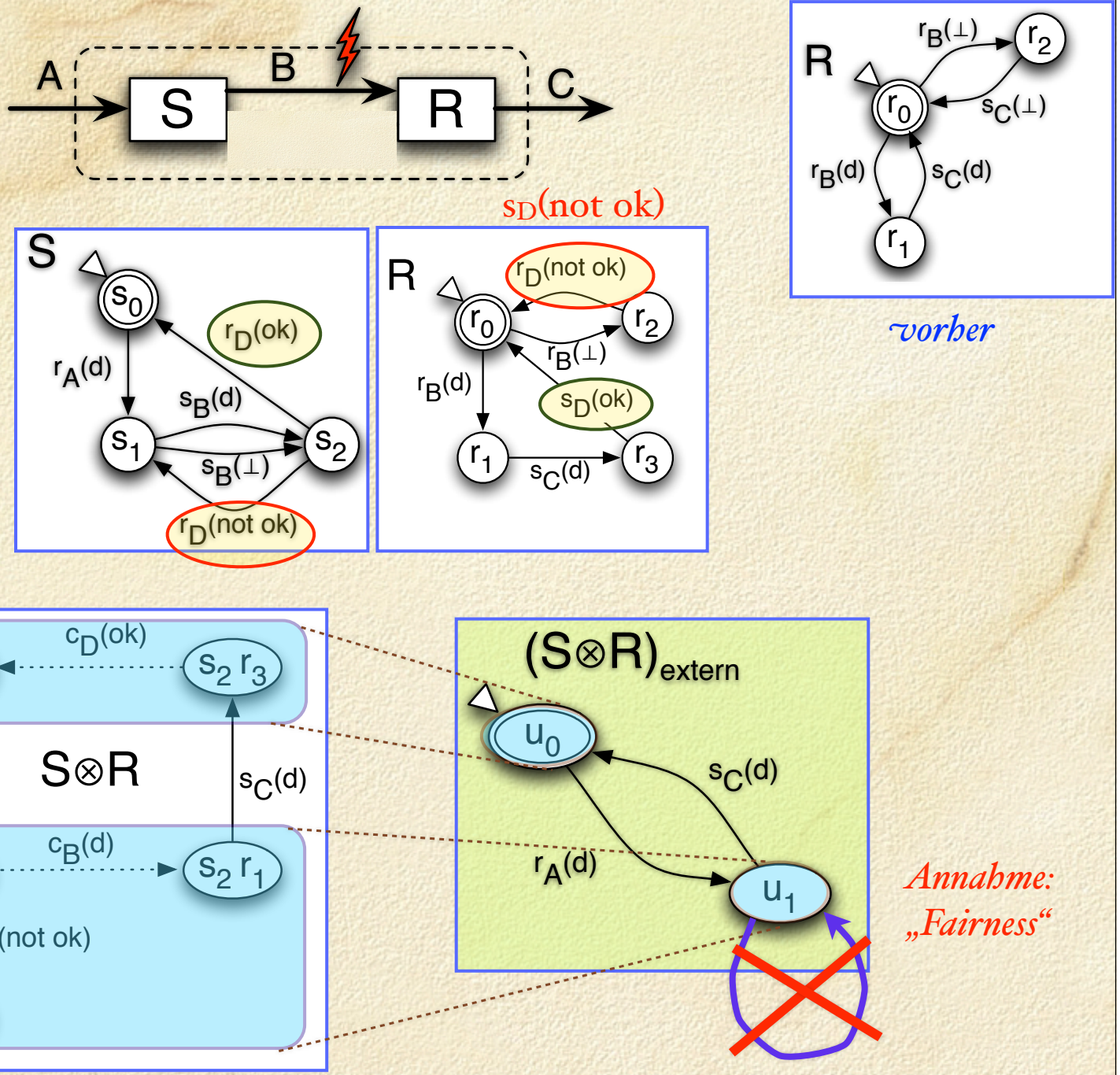
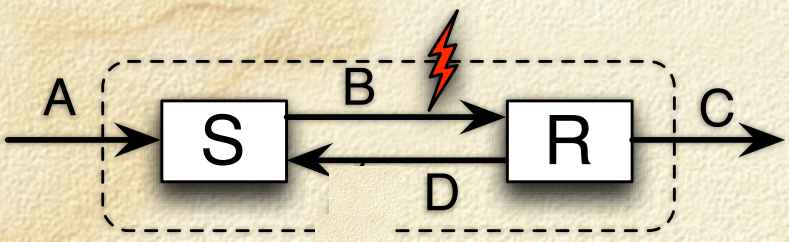
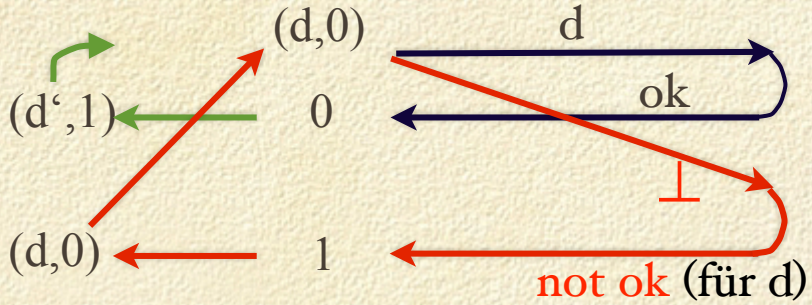


Abbildung 1.8: Entstörtes Sender-Empfänger-System





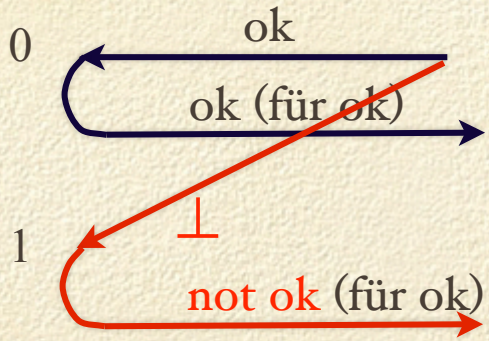
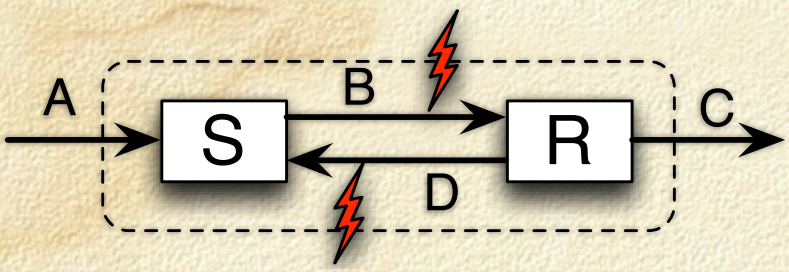
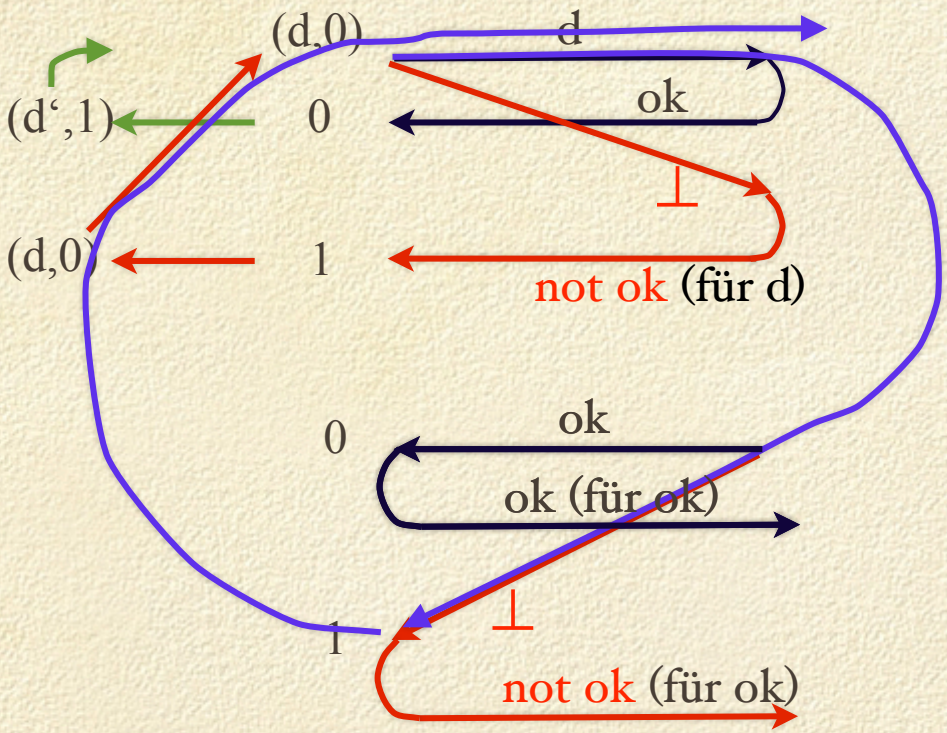


Abbildung 1.9: Das Alternierbit-Protokoll

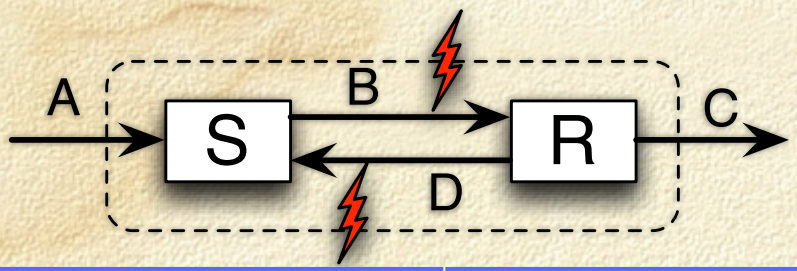


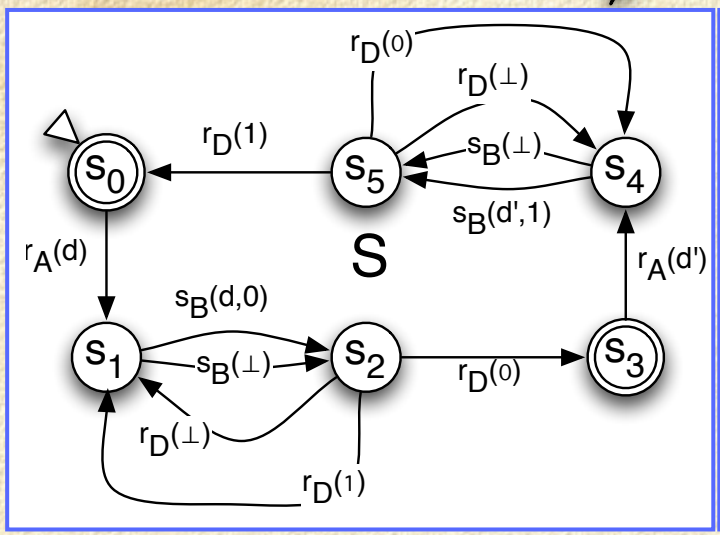


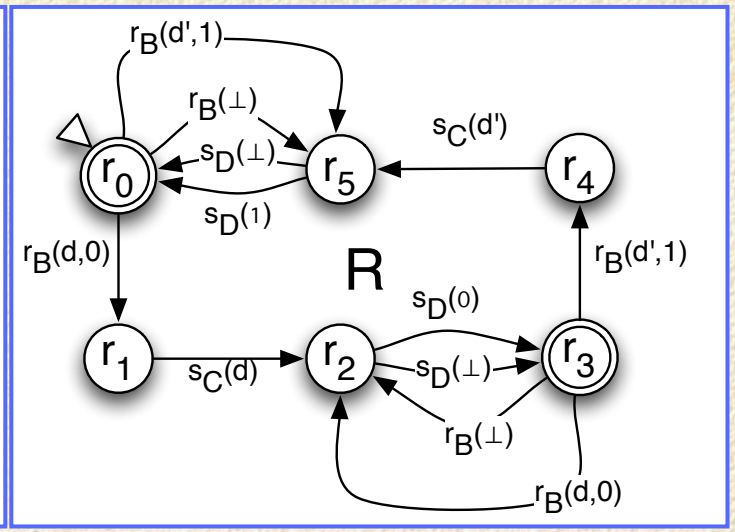
Was passiert bei korrektem Versand, aber gestörter Quittung?

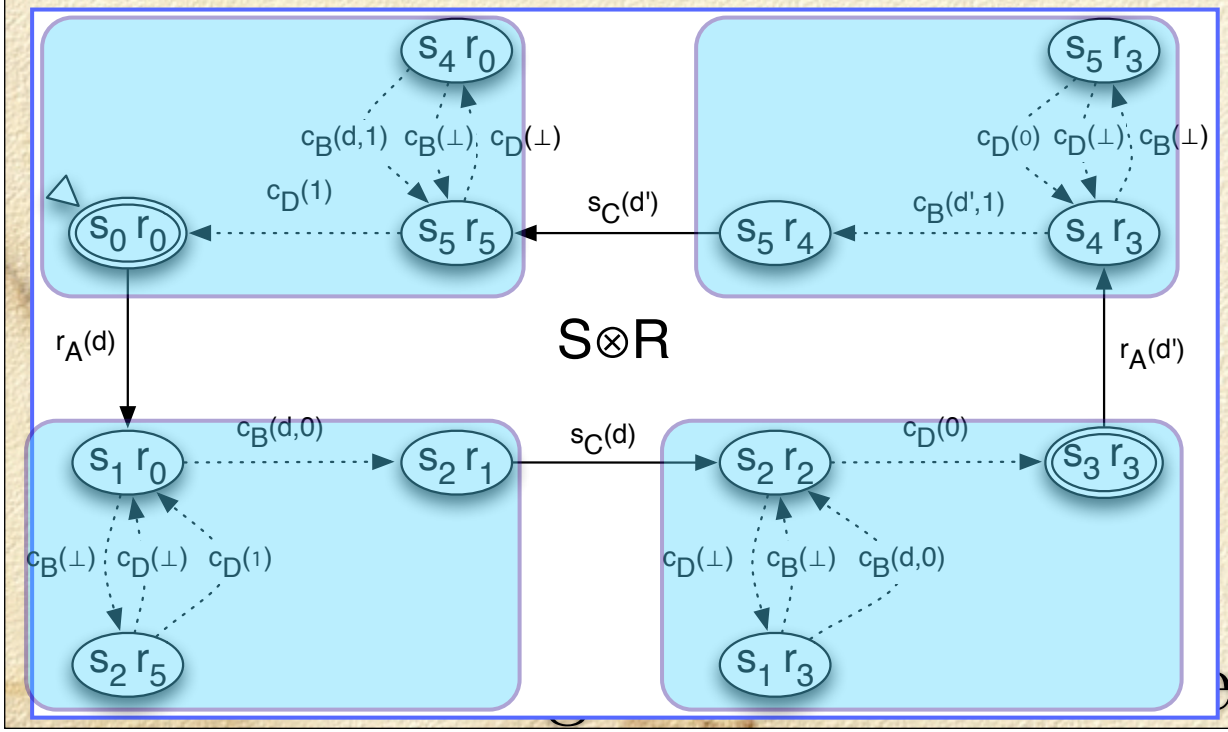
(d,0) *wird doppelt versandt!!!!*

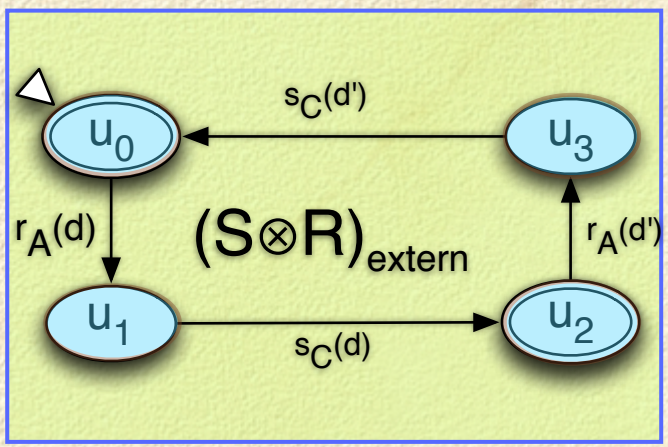
Abbildung 1.9: Das Alternierbit-Protokoll

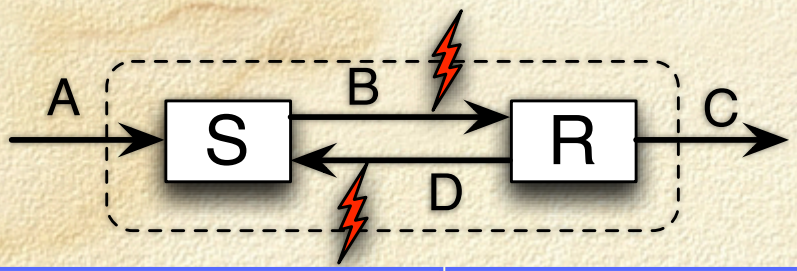

A
S
B
R
C
D

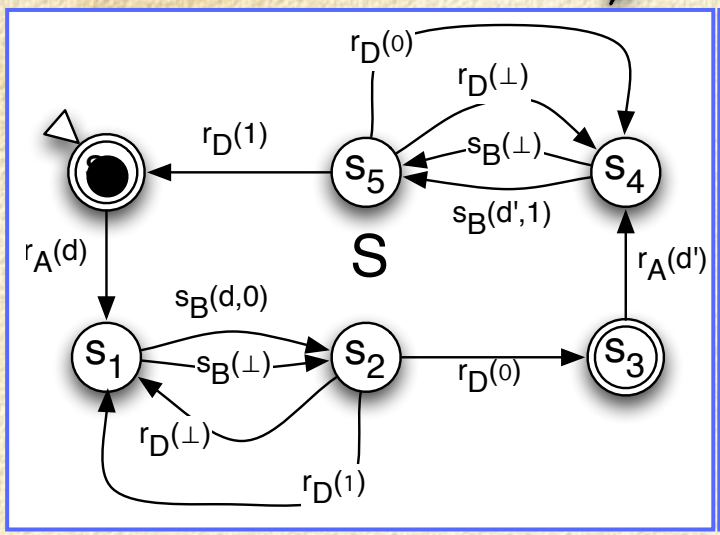

s_0
s_1
s_2
s_3
s_4
s_5
S
$r_D(0)$
$r_D(\perp)$
$r_D(1)$
$s_B(\perp)$
$s_B(d',1)$
$r_A(d)$
$r_A(d')$
$s_B(d,0)$
$s_B(\perp)$
$r_D(0)$
$r_D(\perp)$
$r_D(1)$

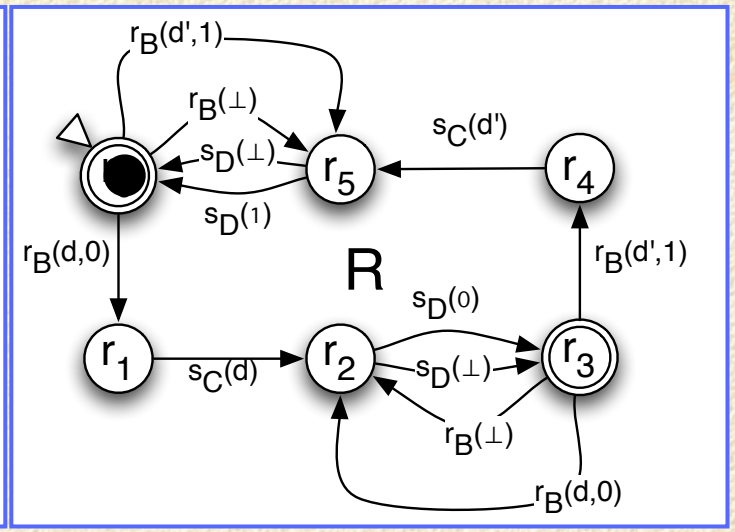

r_0
r_1
r_2
r_3
r_4
r_5
R
$r_B(d',1)$
$r_B(\perp)$
$s_D(\perp)$
$s_C(d')$
$s_D(1)$
$r_B(d,0)$
$r_B(d',1)$
$s_D(0)$
$s_C(d)$
$s_D(\perp)$
$r_B(\perp)$
$r_B(d,0)$

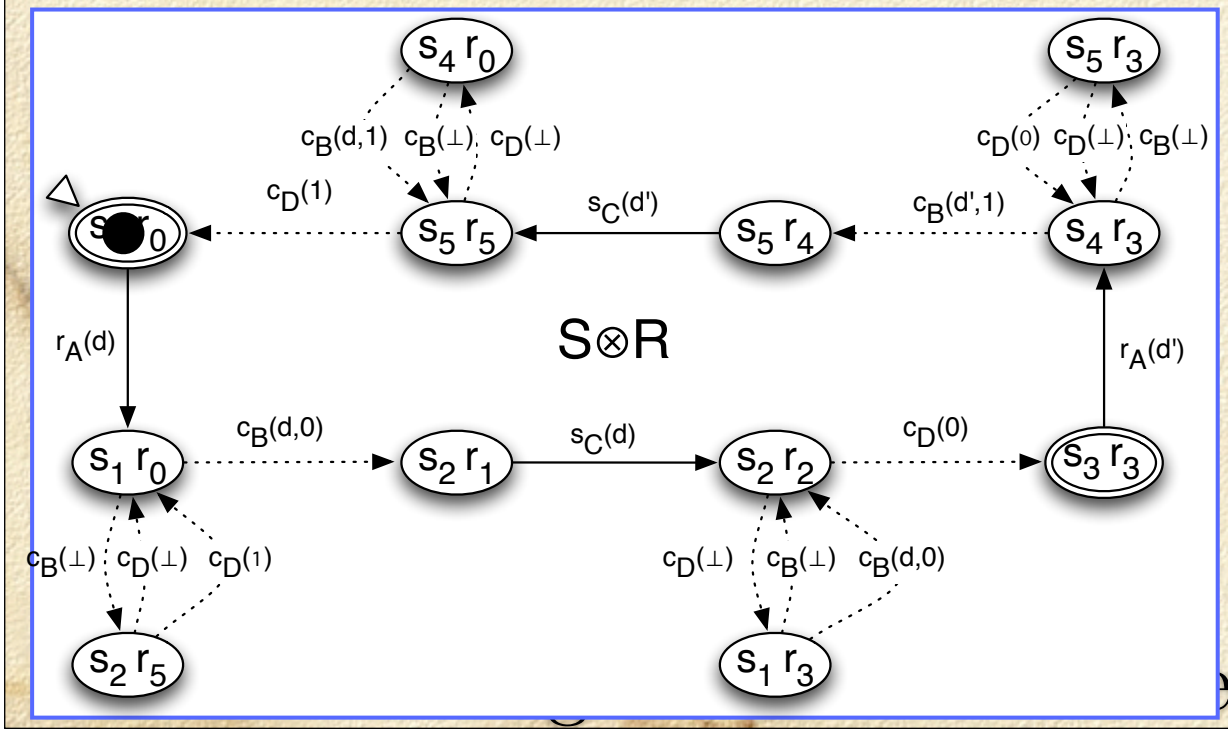

$s_0\ r_0$
$s_1\ r_0$
$s_2\ r_1$
$s_2\ r_2$
$s_3\ r_3$
$s_4\ r_0$
$s_5\ r_3$
$s_5\ r_5$
$s_5\ r_4$
$s_4\ r_3$
$s_2\ r_5$
$s_1\ r_3$
S⊗R
$c_B(d,1)$ $c_B(\perp)$ $c_D(\perp)$
$c_D(1)$
$s_C(d')$
$c_D(0)$ $c_D(\perp)$ $c_B(\perp)$
$c_B(d',1)$
$r_A(d)$
$r_A(d')$
$c_B(d,0)$
$s_C(d)$
$c_D(0)$
$c_B(\perp)$ $c_D(\perp)$ $c_D(1)$
$c_D(\perp)$ $c_B(\perp)$ $c_B(d,0)$

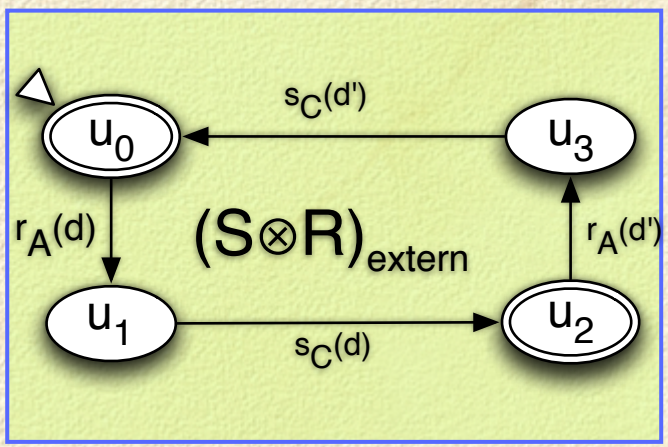

u_0
u_1
u_2
u_3
$(S⊗R)_{extern}$
$s_C(d')$
$r_A(d)$
$r_A(d')$
$s_C(d)$



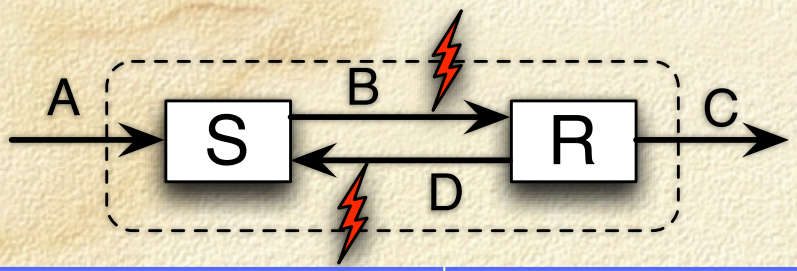


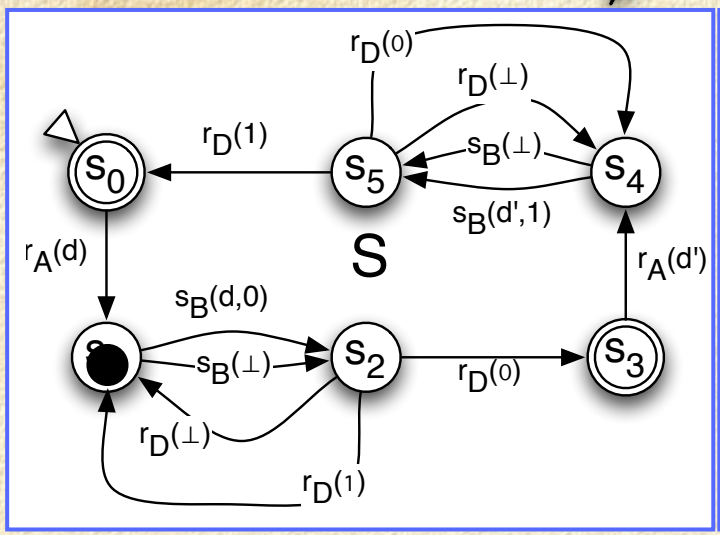


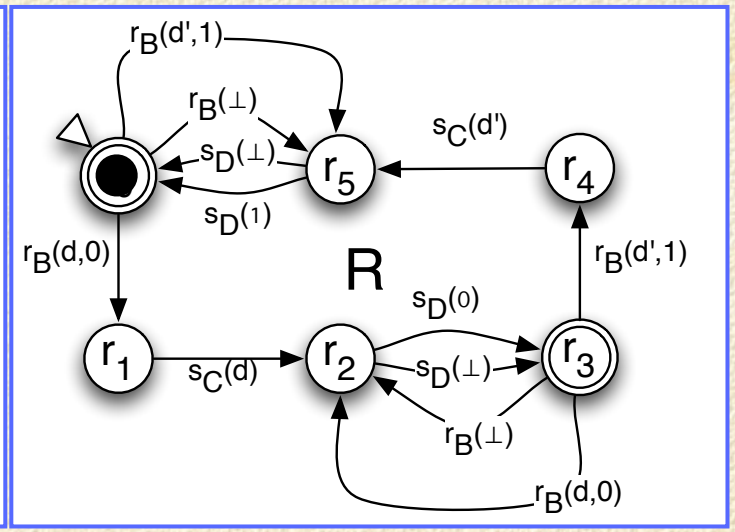


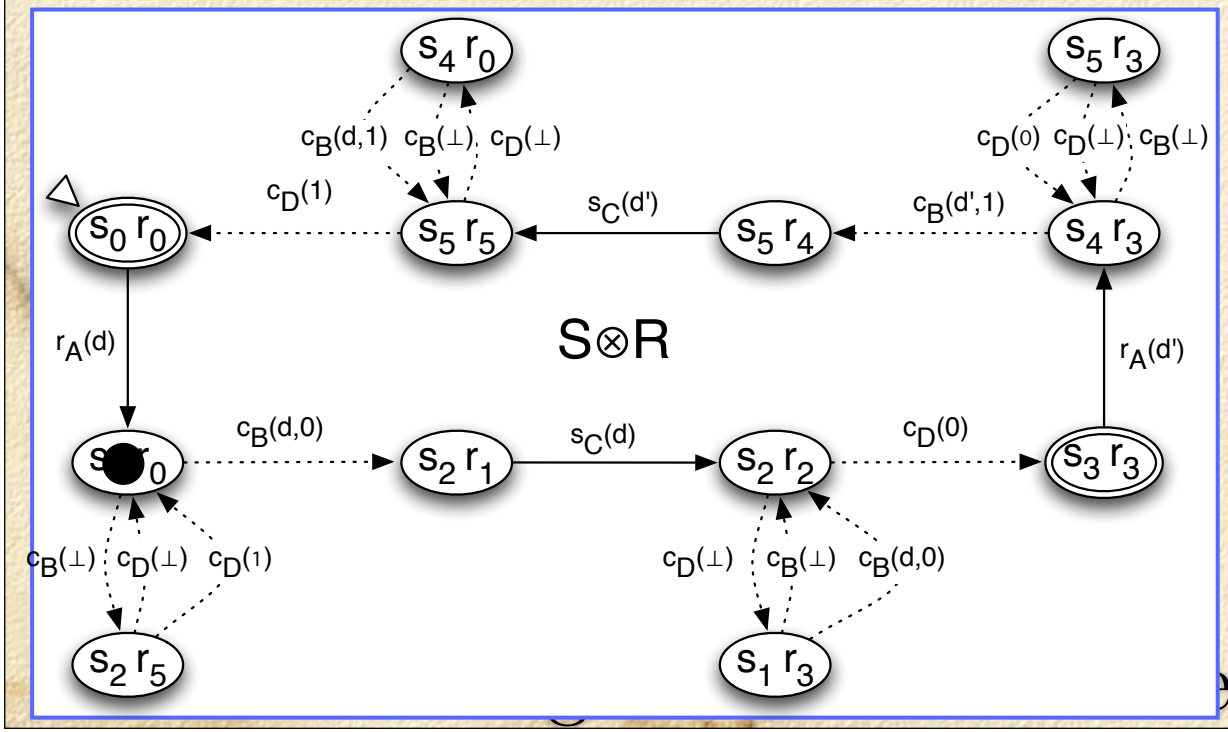


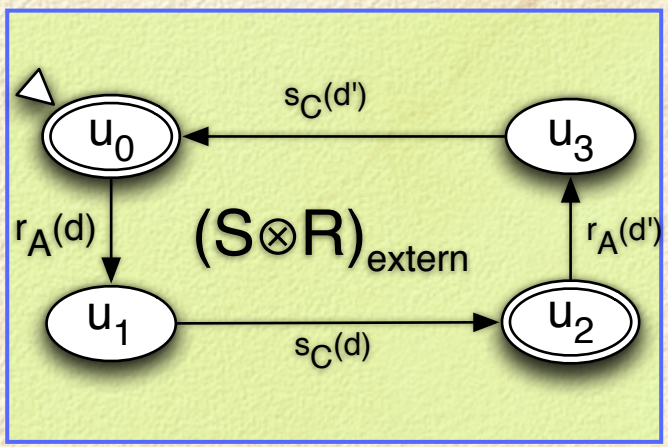
erbit-Protokoll



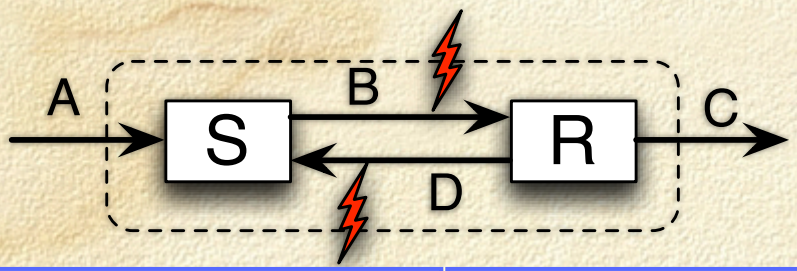


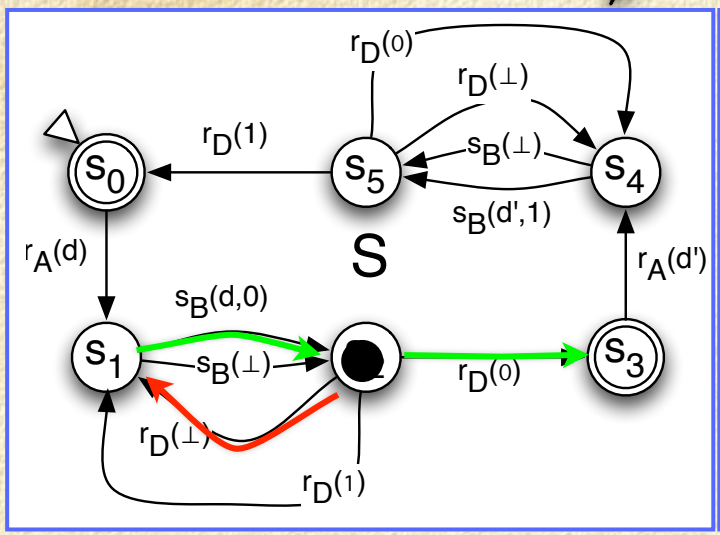


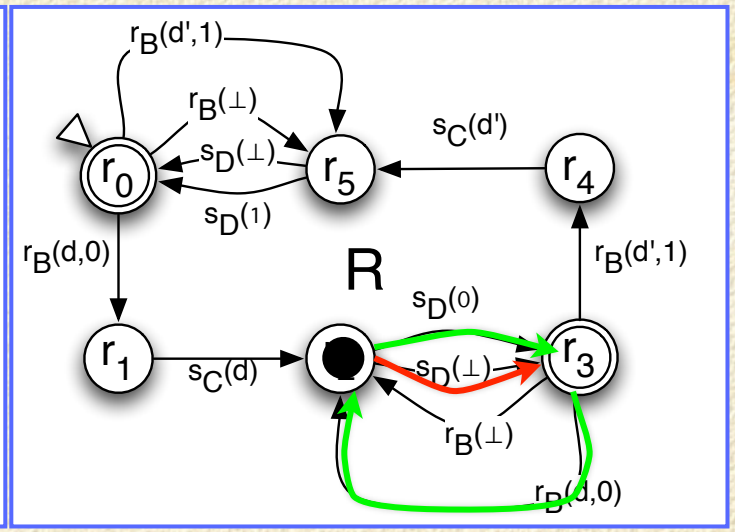


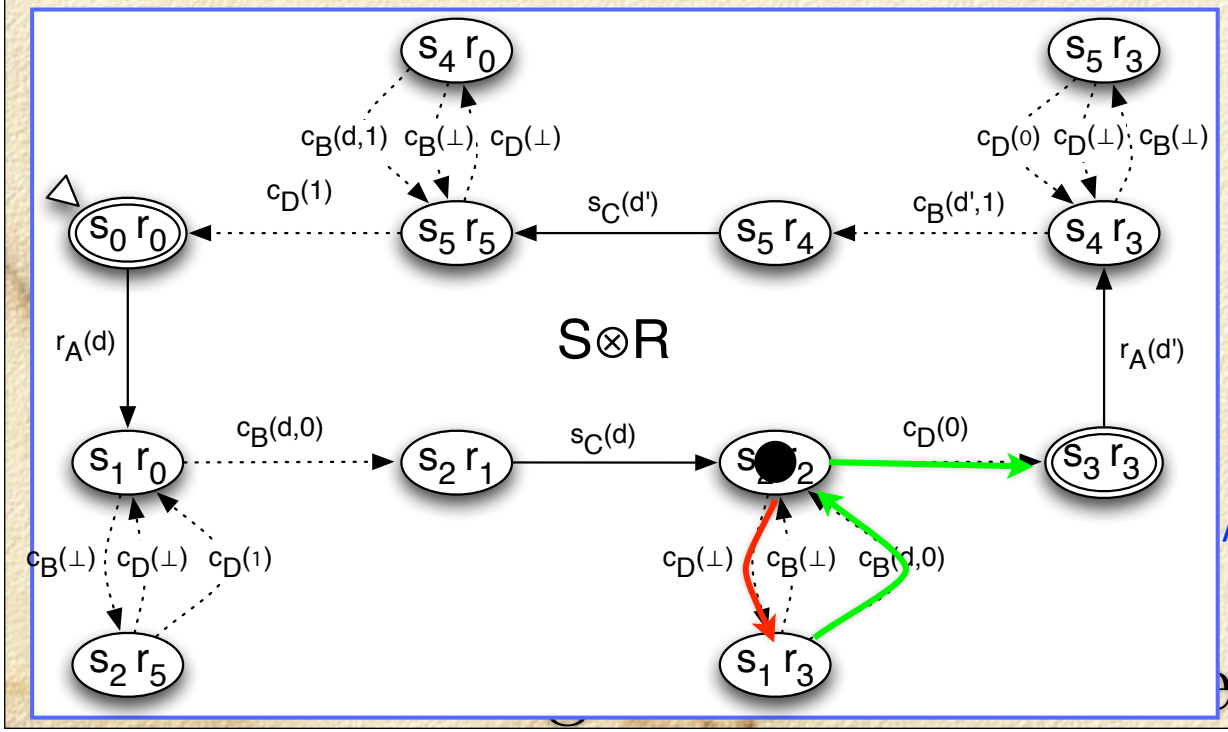


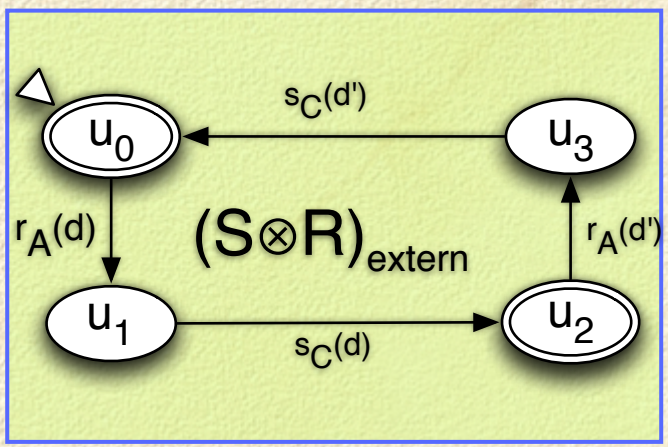
erbit-Protokoll









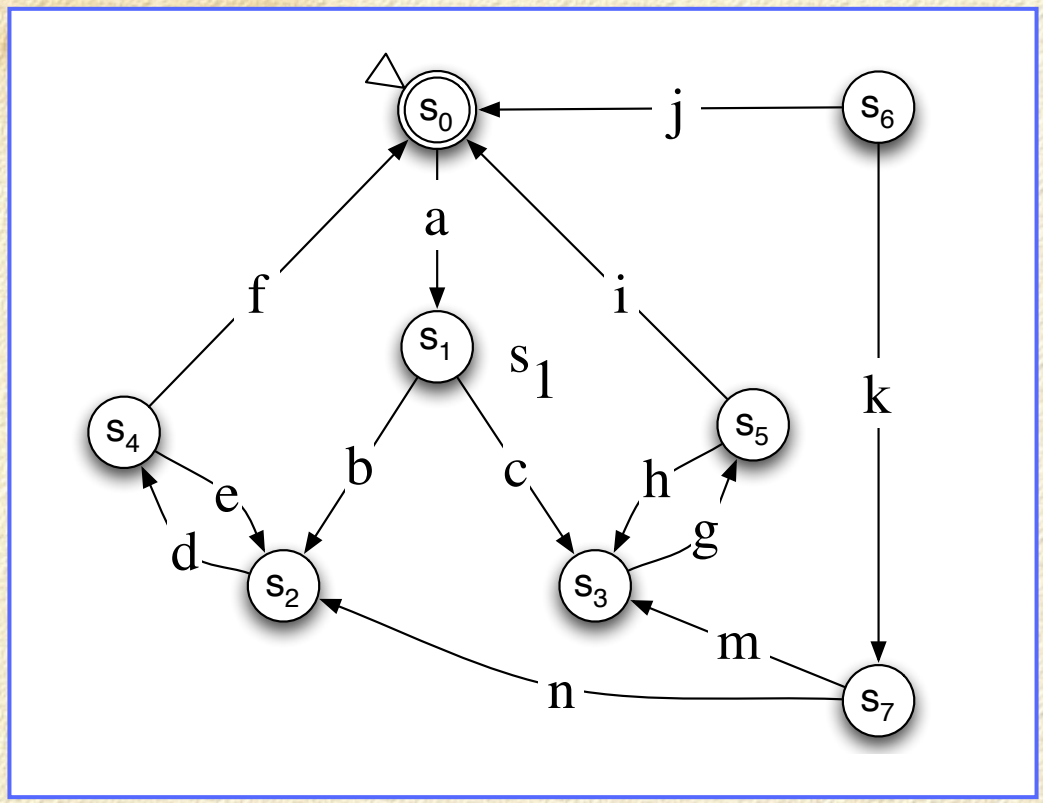


korrekter Versand, aber gestörter Quittung

(d,0) *wird doppelt versandt????*

erbit-Protokoll

1.3 Automaten, reguläre Sprachen und Model-Checking



$$L(TS) = (a(bd(ed)^*f + cg(hg)^*i))^*.$$

$$L(TS) = (a(bd(ed)^* f + cg(hg)^* i))^*.$$

rationale

Definition 1.14 *Die regulären Ausdrücke über einem endlichen Alphabet Σ sind definiert durch:*

1. *$\emptyset$ ist ein regulärer Ausdruck, der die (leere) Menge $M_\emptyset := \emptyset$ beschreibt.*

2. *Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck, der die Menge $M_a := \{a\}$ beschreibt.*

3. *Sind A und B reguläre Ausdrücke, welche die Mengen M_A und M_B beschreiben, dann sind induktiv folgende reguläre Ausdrücke definiert:*

   - *$(A + B)$ beschreibt die Menge $M_A \cup M_B$,*
   - *$(A \cdot B)$ beschreibt die Menge $M_A \cdot M_B$,*
   - *A^* beschreibt die Menge M_A^*,*
   - *A^+ beschreibt die Menge M_A^+.*

4. Nur ...

$$\Sigma = \{a, b\}$$

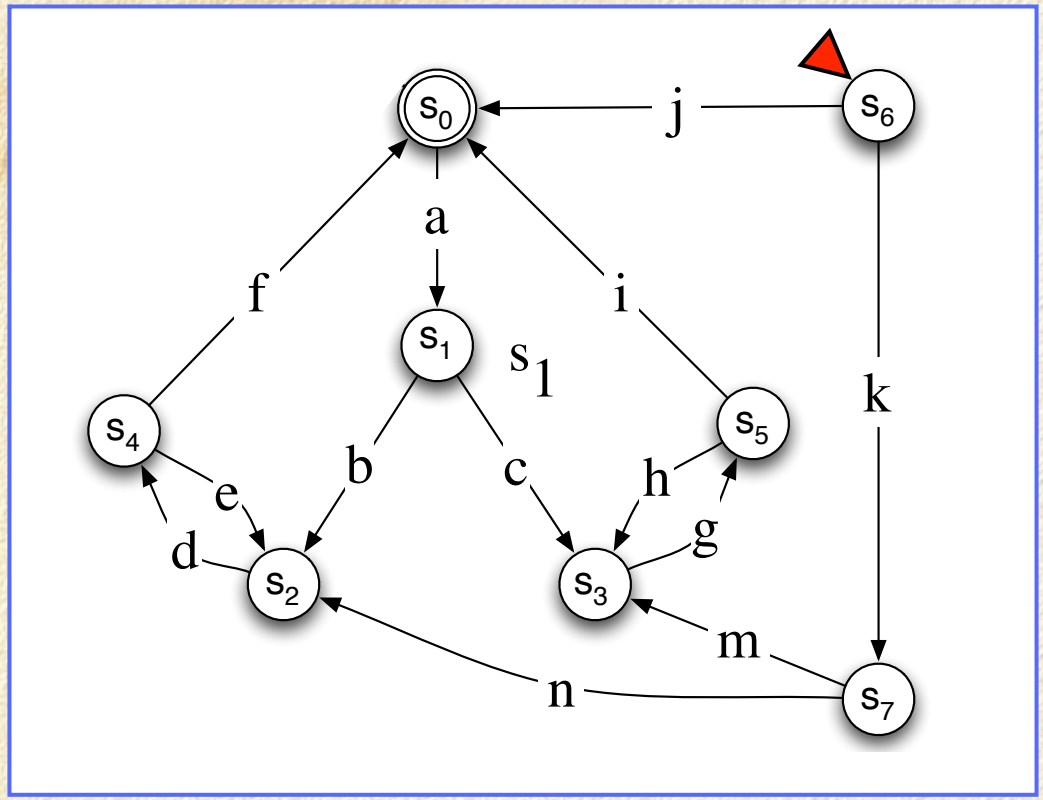
$$(a+b)\Sigma^* + (a+b)(a+b)\Sigma^*$$

$$(a+b)(\Sigma^* + (a+b)\Sigma^*)$$

$$\Sigma^+$$

Menge aller Wörter über Σ*, die mindestens ein Zeichen enthalten.*

ω-reguläre Ausdrücke



$$L^{\omega}(TS) = knd(ed)^{*}fC^{\omega} + kmg(hg)^{*}iC^{\omega} + jC^{\omega}$$

$$L(TS) = (\underbrace{a(bd(ed)^{*}f + cg(hg)^{*}i)}_{C})^{*}.$$

$$L^\omega(TS) = knd(ed)^* fC^\omega + kmg(hg)^* iC^\omega$$

Definition 1.15

$$+ j\, C^\omega$$

$$w = a_1 a_2 \cdots a_n \in \Sigma^*$$

$$u = b_1 b_2 \cdots \in \Sigma^\omega$$

$$w \cdot u := a_1 a_2 \cdots a_n b_1 b_2 \cdots \in \Sigma^\omega$$

$$W \cdot U := \{w \cdot u | w \in W, u \in U\} \subseteq \Sigma^\omega$$

$$W \subseteq \Sigma^* \quad U \subseteq \Sigma^\omega$$

$$W^\omega := \{w_1 \cdot w_2 \cdot w_3 \cdots | w_i \in W \backslash \{\epsilon\}, i \geq 1\}$$

ω-Abschluss

$$L^\omega(TS) = knd(ed)^* fC^\omega + kmg(hg)^* iC^\omega + j\,C^\omega$$

$$G = A_1 \cdot B_1^\omega + \cdots + A_n \cdot B_n^\omega$$

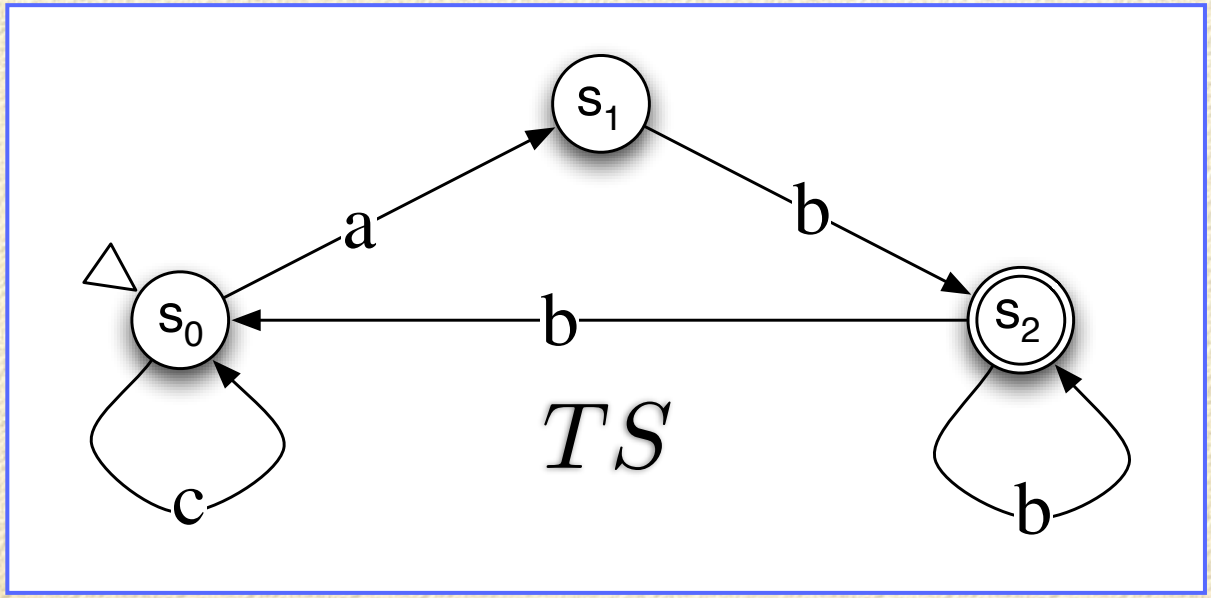
$$M_G := M_{A_1} \cdot M_{B_1}^\omega \cup \cdots \cup M_{A_n} \cdot M_{B_n}^\omega .$$

ω-reguläre Menge

ω - rationale

Satz 1.17 *a) Die durch endliche Transitionssysteme $TS = (S, \Sigma, tr, S^0, S^F)$ akzeptierten Sprachen $L(TS)$ sind genau die regulären Sprachen über Σ.* *Satz von Kleene*

b) Die durch endliche Transitionssysteme $TS = (S, \Sigma, tr, S^0, S^F)$ akzeptierten ω-Sprachen $L^\omega(TS)$ sind genau die ω-regulären Sprachen über Σ.



$$L^\omega(TS) = c^* ab(b^+ + bc^* ab)^\omega$$

„Produkt - Automat“

Produkt-Transitionssystem für Durchschnitt

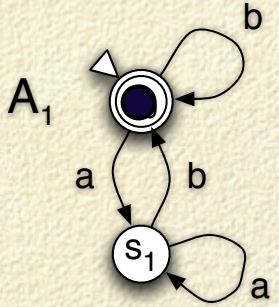
Satz 1.18 *Gegeben seien für $i \in \{1,2\}$ zwei Transitionssysteme $TS_i = (S_i, A_i, tr_i, S_i^0, S_i^F)$ mit Endzuständen. Dann können Transitionssyteme TS_3 und TS_4 effektiv konstruiert werden, die den Durchschnitt der akzeptierten Sprachen akzeptieren:*

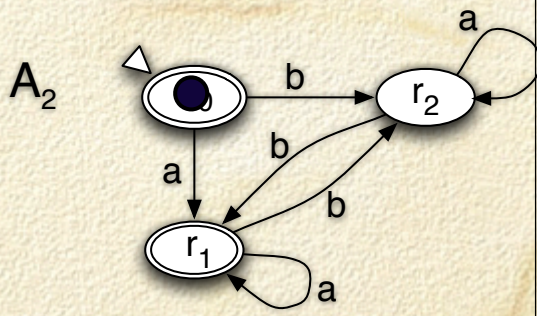
a) $L(TS_3) = L(TS_1) \cap L(TS_2)$

a) Wir definieren TS_3 als das Transitionssystem $TS_1 \otimes_{Sync} TS_2$ mit $Sync = \{(a,a) | a \in A_1 \cap A_2\}$ und $\gamma(a,a) = a$ für $a \in A_1 \cap A_2$ und streichen alle Transitionen $(s_1, r_1) \xrightarrow{a} (s_2, r_2)$ mit $a \notin A_1 \cap A_2$, d.h. bei der Produktdefinition von Definition 1.9 auf Seite 10 kommt nur die Regel $Sy3$ zur Anwendung.

Produkttransitionssystem für Durchschnittbildung

Satz 1.18 *Gegeben seien für $i \in \{1,2\}$ zwei Transitionssysteme $TS_i = (S_i, A_i, tr_i, S_i^0, S_i^F)$ mit Endzuständen. Dann können Transitionssyteme TS_3 und TS_4 effektiv konstruiert werden, die den Durchschnitt der akzeptierten Sprachen akzeptieren:*





a) $L(TS_3) = L(TS_1) \cap L(TS_2)$

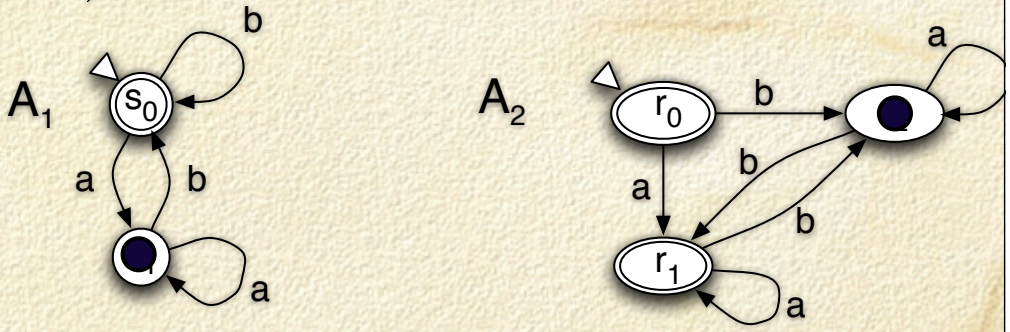
$w \in L(TS_1) \cap L(TS_2)$ $\quad w = a_1 a_2 \cdots a_n$:

abababab

Produkttransitionssystem für Durchschnittbildung

Satz 1.18 *Gegeben seien für $i \in \{1,2\}$ zwei Transitionssysteme $TS_i = (S_i, A_i, tr_i, S_i^0, S_i^F)$ mit Endzuständen. Dann können Transitionssyteme TS_3 und TS_4 effektiv konstruiert werden, die den Durchschnitt der akzeptierten Sprachen akzeptieren:*

a) $L(TS_3) = L(TS_1) \cap L(TS_2)$



$$\Delta\ s_0 \xrightarrow{a_1}_1 s_1 \xrightarrow{a_2}_1 s_2 \cdots s_{n-1} \xrightarrow{a_n}_1 s_n$$

$$\Delta\ r_0 \xrightarrow{a_1}_2 r_1 \xrightarrow{a_2}_2 r_2 \cdots r_{n-1} \xrightarrow{a_n}_2 r_n$$

$$\Delta\ (s_0, r_0) \xrightarrow{a_1}_3 (s_1, r_1) \xrightarrow{a_2}_3 (s_2, r_2) \cdots (s_{n-1}, r_{n-1}) \xrightarrow{a_n}_3 (s_n r_n)$$

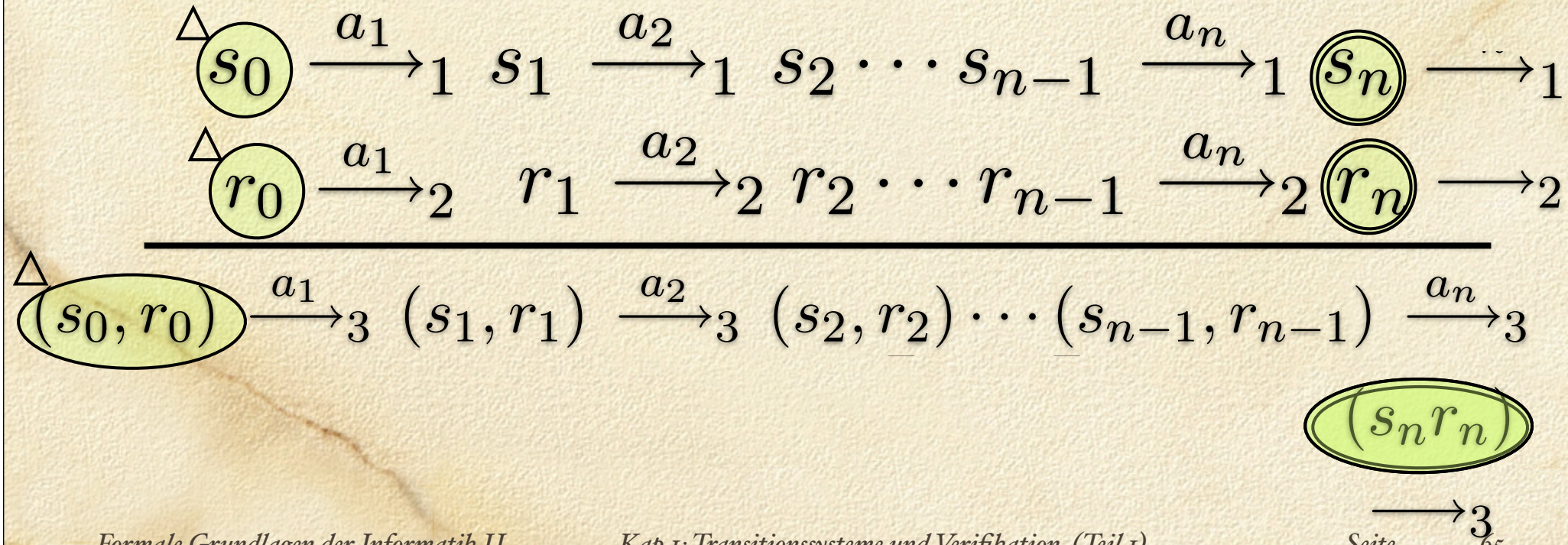
Produkttransitionssystem für Durchschnittbildung

Satz 1.18 *Gegeben seien für $i \in \{1,2\}$ zwei Transitionssysteme $TS_i = (S_i, A_i, tr_i, S_i^0, S_i^F)$ mit Endzuständen. Dann können Transitionssyteme TS_3 und TS_4 effektiv konstruiert werden, die den Durchschnitt der akzeptierten Sprachen akzeptieren:*

a) $L(TS_3) = L(TS_1) \cap L(TS_2)$

b) $L^\omega(TS_4) = L^\omega(TS_1) \cap L^\omega(TS_2)$.

$w \in L(TS_1) \cap L(TS_2)$ $\quad w = a_1 a_2 \cdots a_n$.

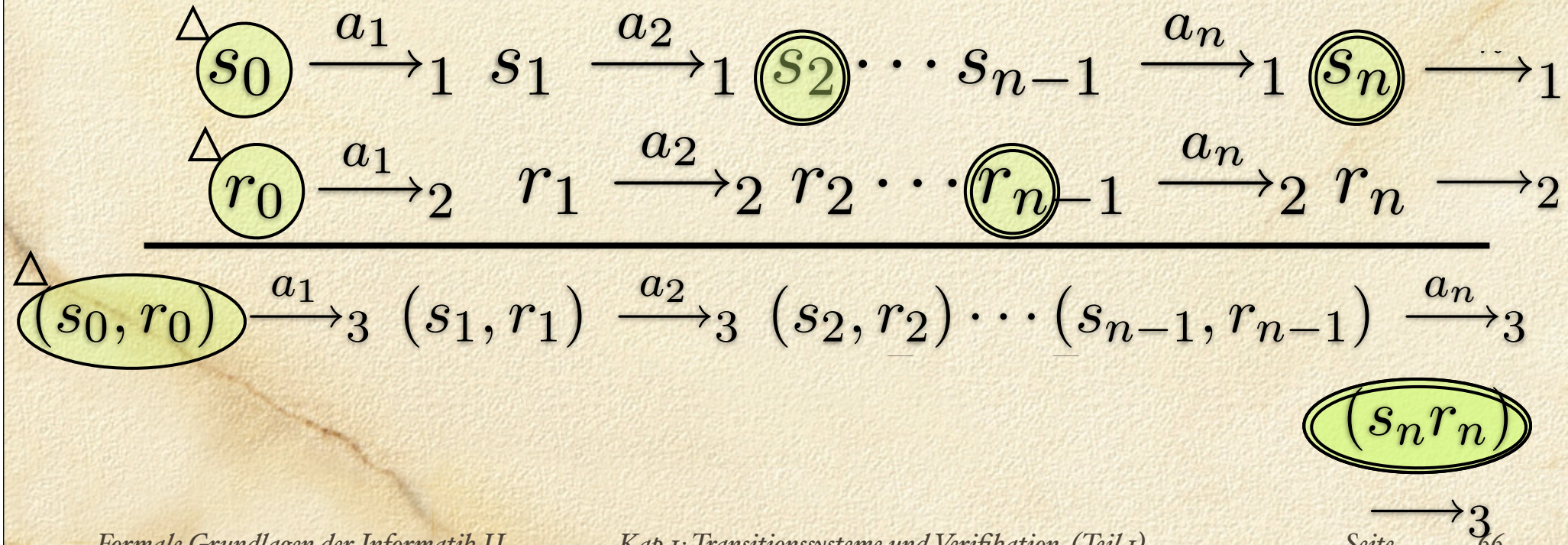


Produkttransitionssystem für Durchschnittbildung

Satz 1.18 *Gegeben seien für $i \in \{1,2\}$ zwei Transitionssysteme $TS_i = (S_i, A_i, tr_i, S_i^0, S_i^F)$ mit Endzuständen. Dann können Transitionssyteme TS_3 und TS_4 effektiv konstruiert werden, die den Durchschnitt der akzeptierten Sprachen akzeptieren:*

a) $L(TS_3) = L(TS_1) \cap L(TS_2)$ *b)* $L^\omega(TS_4) = L^\omega(TS_1) \cap L^\omega(TS_2)$.

$w \in L(TS_1) \cap L(TS_2)$ $\quad w = a_1 a_2 \cdots a_n$,

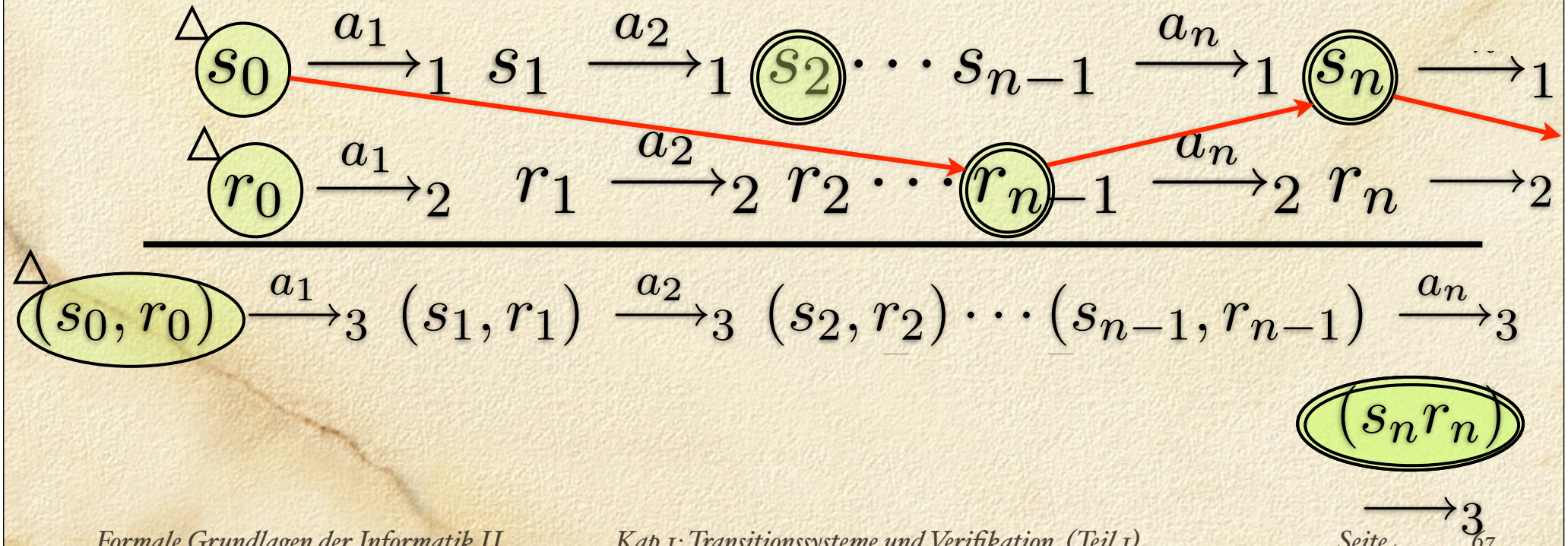


Produkttransitionssystem für Durchschnittbildung

Satz 1.18 *Gegeben seien für $i \in \{1,2\}$ zwei Transitionssysteme $TS_i = (S_i, A_i, tr_i, S_i^0, S_i^F)$ mit Endzuständen. Dann können Transitionssyteme TS_3 und TS_4 effektiv konstruiert werden, die den Durchschnitt der akzeptierten Sprachen akzeptieren:*

a) $L(TS_3) = L(TS_1) \cap L(TS_2)$ *b)* $L^\omega(TS_4) = L^\omega(TS_1) \cap L^\omega(TS_2)$.

$w \in L(TS_1) \cap L(TS_2)$ $\quad w = a_1 a_2 \cdots a_n$,



Formal definieren wir $TS_4 = (S_4, A_1 \cap A_2, tr_4, S_4^0, S_4^F)$ durch

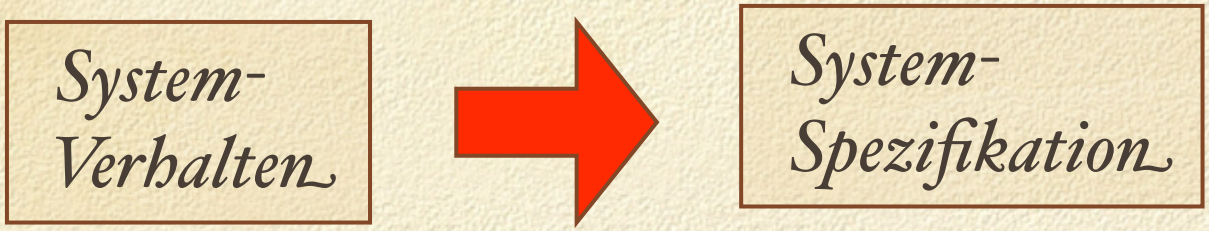
a) $S_4 = \{(s,r,q) | (s,r) \in S_3, q \in \{1,2\}\}$

b) $(s,r,q) \xrightarrow{a}_4 (s',r',q') :\Longleftrightarrow (s,r) \xrightarrow{a}_3 (s',r') \ \wedge \ q' := \begin{cases} 2 & \text{falls } q = 1 \wedge s \in S_q^F \\ 1 & \text{falls } q = 2 \wedge r \in S_q^F \\ q & \text{sonst} \end{cases}$

c) $S_4^0 := \{(s,r,1) | (s,r) \in S_3^0 = S_1^0 \times S_2^0\}$

d) $S_4^F := \{(s,r,1) | s \in S_1^F\}$

Verifikation eines Systems

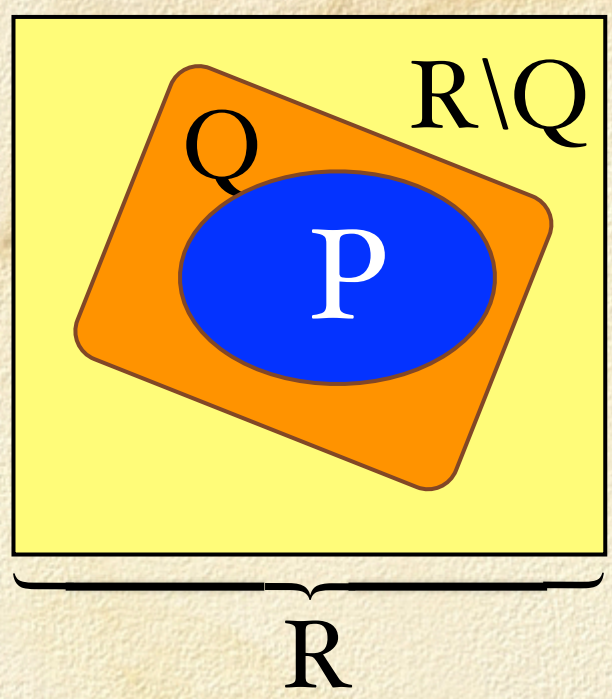


$$L(TS_{sys}) \subseteq L(TS_{spec})$$

$$L^{\omega}(TS_{sys}) \subseteq L^{\omega}(TS_{spec}).$$

(temporal-)logische Formel f_{spec}

$$L(TS_{sys}) \subseteq L(TS_{spec})$$

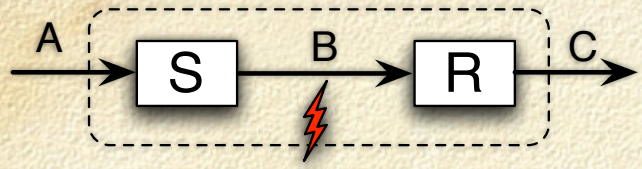


$$P \subseteq Q \Leftrightarrow P \cap (R \backslash Q) = \emptyset$$

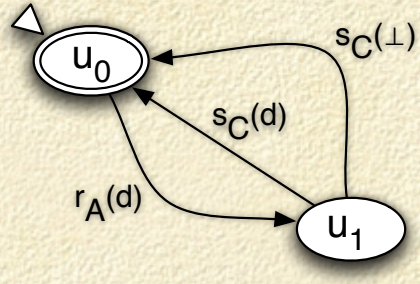
$$L(TS_{sys}) \cap (A^* \backslash L(TS_{spec})) = \emptyset$$

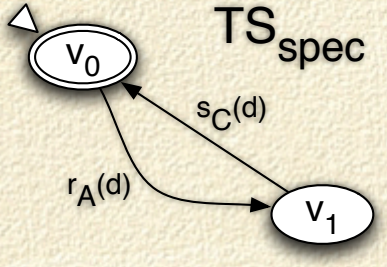
$$L^\omega(TS_{sys}) \cap (A^\omega \backslash L^\omega(TS_{spec})) = \emptyset$$

$$\underbrace{f_{spec}}_{\neg f_{spec}}$$



$TS_{sys} = (S \otimes R)_{extern}$





$$L(TS_{sys}) \stackrel{?}{\subseteq} L(TS_{spec})$$

$$L(TS_{sys}) \cap (A^* \setminus L(TS_{spec})) \stackrel{?}{=} \emptyset$$