CoNe
Freiburg

Informatik III

3.1. Berechenbarkeit

Christian Schindelhauer

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik

Wintersemester 2007/08

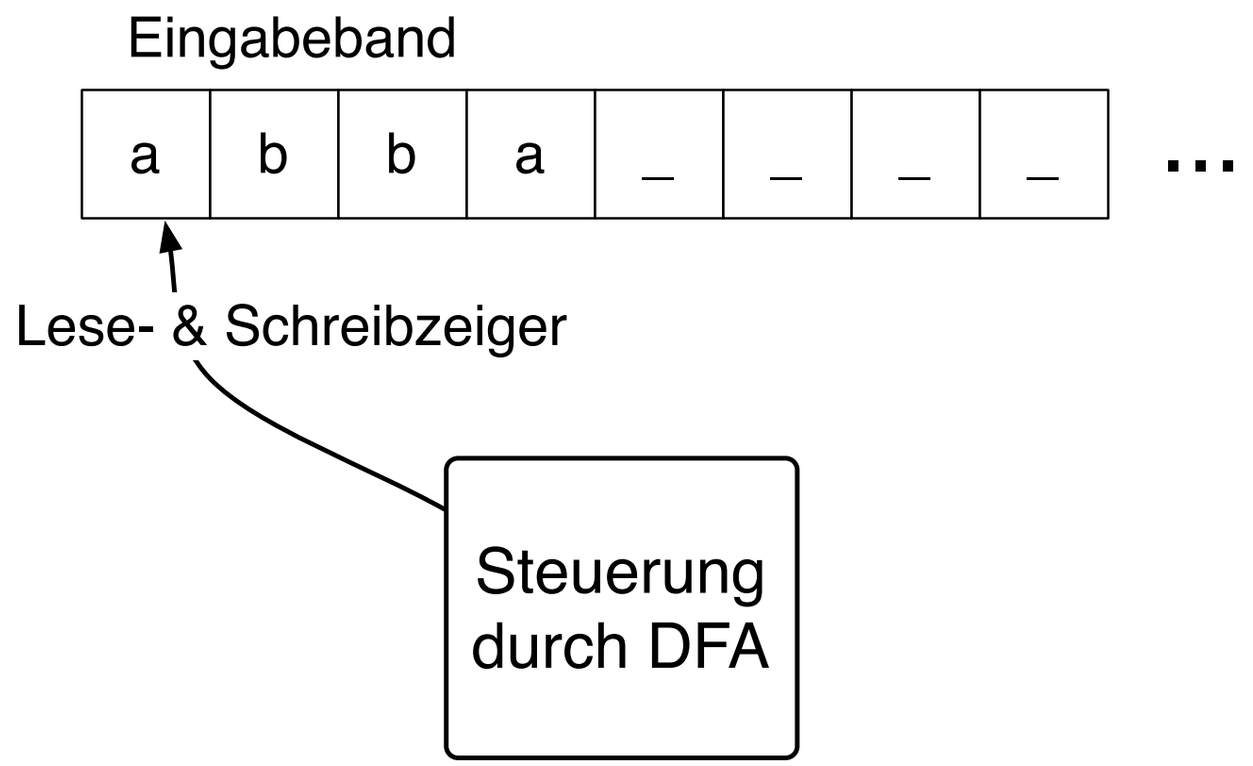
Berechenbarkeitstheorie

Die Turing-Maschine

2

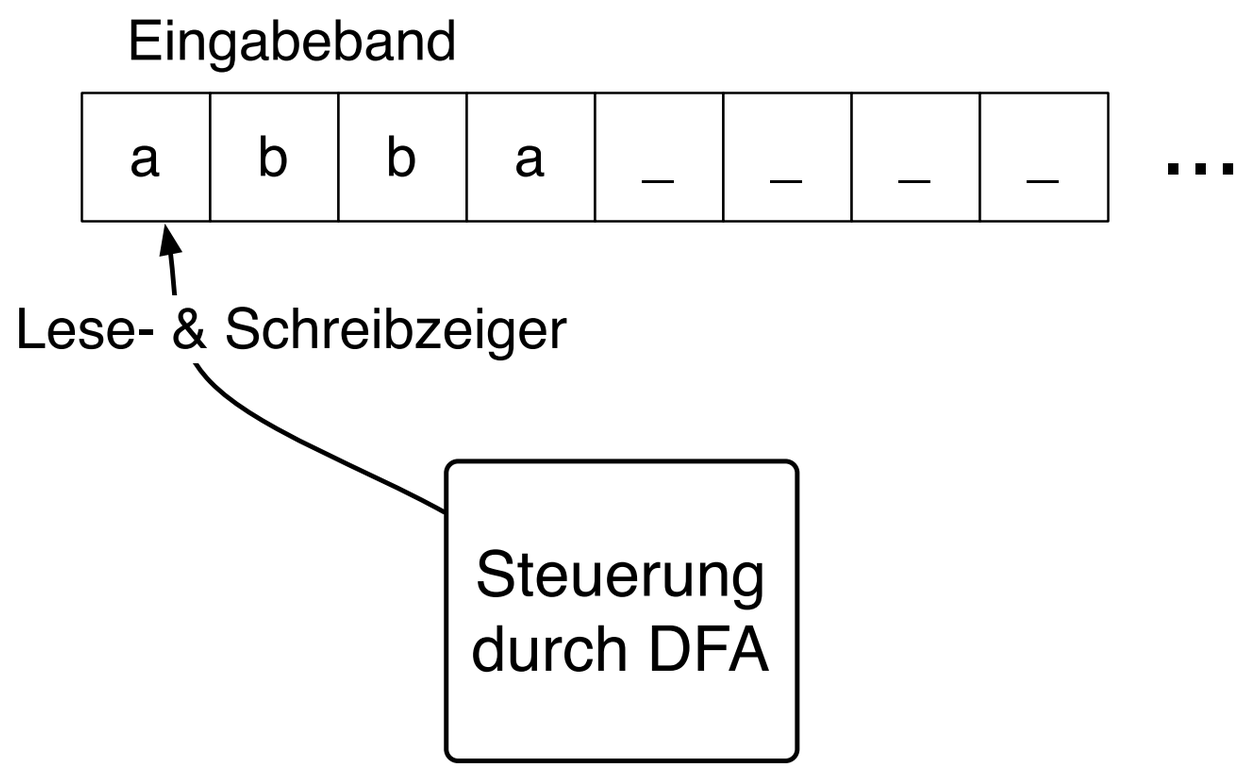
Turingmaschinen

- **DFA mit**
  - unbeschränktem Band
- **Eingabe steht zu Beginn am Anfang des Bands**
- **Auf dem Rest des Bandes steht _ (Blank)**
- **Position auf dem Band wird durch Lesekopf beschrieben**



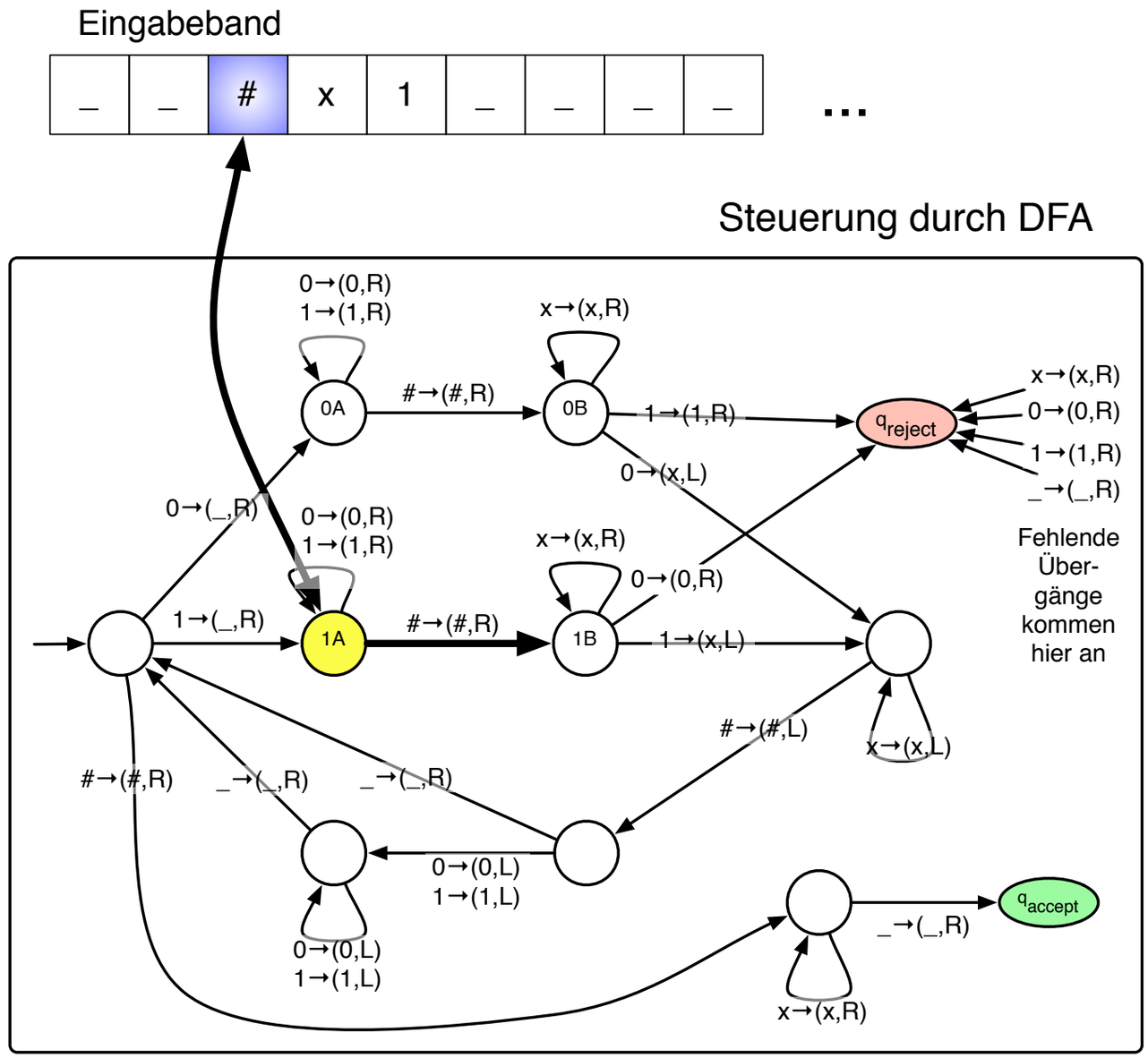
Turingmaschinen

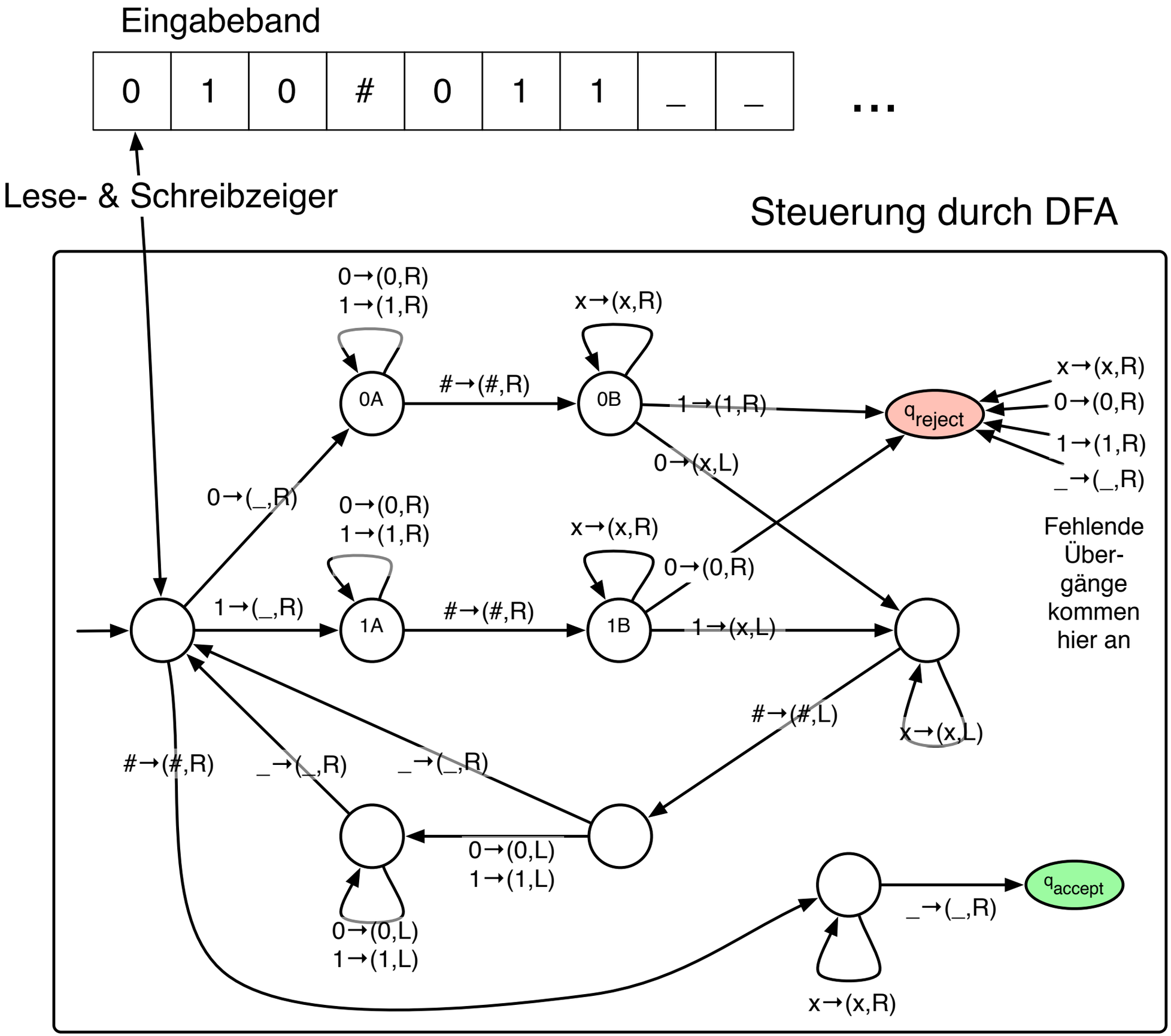
- **Der nächste Rechenschritt ist eindeutig festgelegt (determiniert) durch**
  - aktuellen Zustand und
  - aktuelles Zeichen
- **Rechenschritt**
  - überschreibt das aktuelle Zeichen durch ein neues
  - Kopf bewegt sich nach rechts oder links
  - Zustand verändert sich



Beispiel einer Turing-Maschine

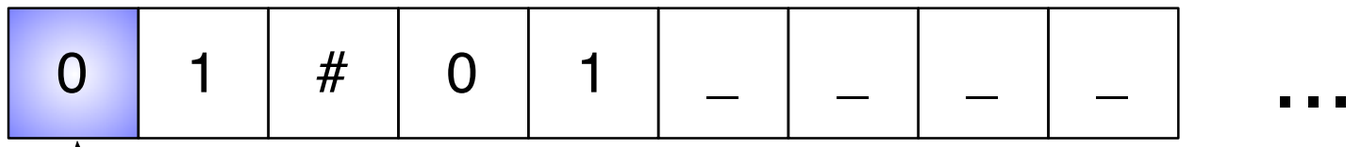
- **L = {w#w | w ∈ {0, 1}*}**
- **Vorgehensweise:**
  - Von links nach rechts über das Wort laufen
  - Erstes Zeichen links merken und löschen
  - Erstes Zeichen rechts von # vergleichen und löschen
  - Für alle Zeichen wiederholen
  - Links dürfen dann nur noch x folgen
  - Falls Zeichen an einer Stelle nicht übereinstimmen ablehnen, sonst am Ende akzeptieren



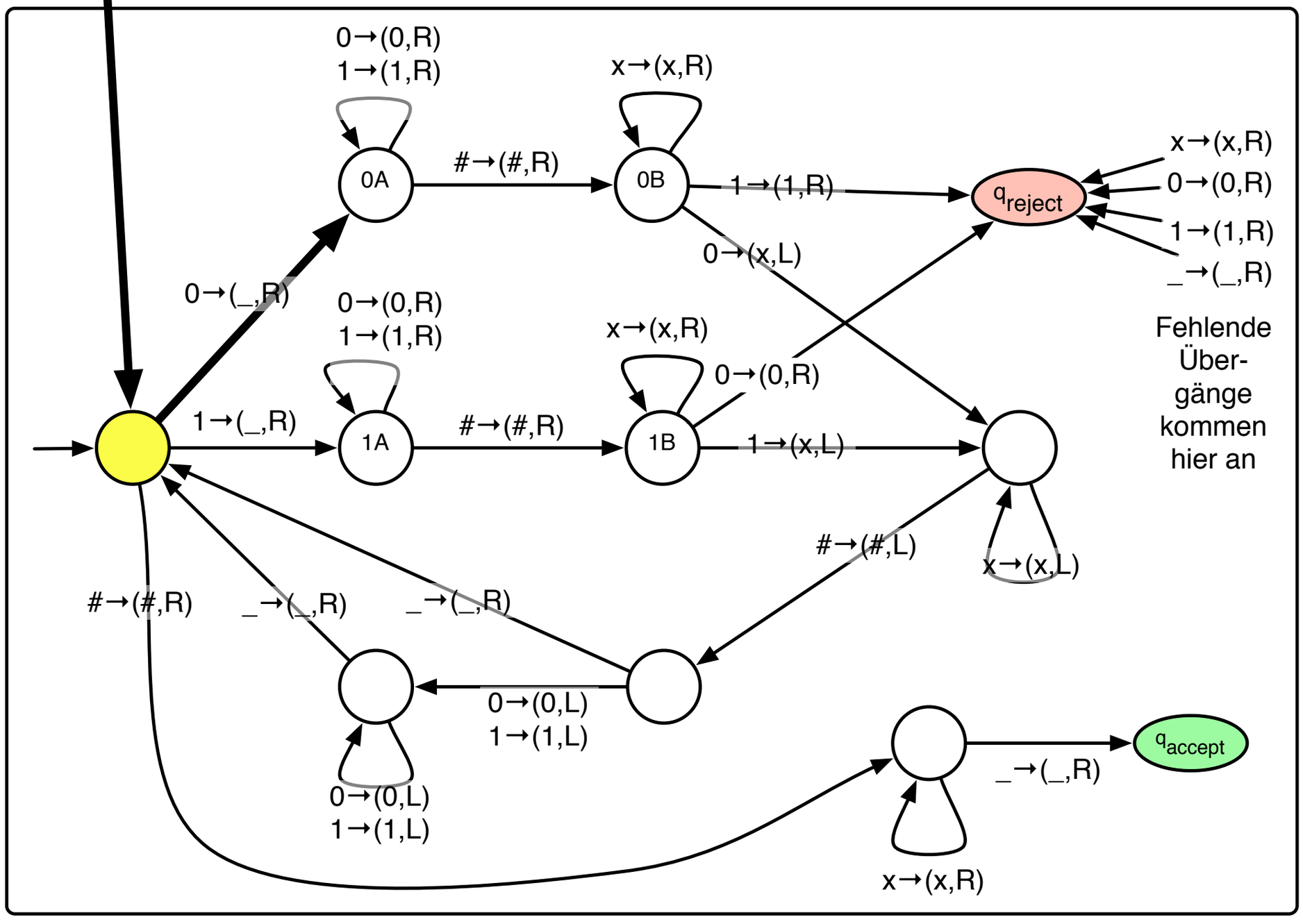

Eingabeband
0
1
0
#
0
1
1
_
_
...
Lese- & Schreibzeiger
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
0→(0,R)
1→(_,R)
1A
#→(#,R)
1B
1→(x,L)
Fehlende Über-gänge kommen hier an
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

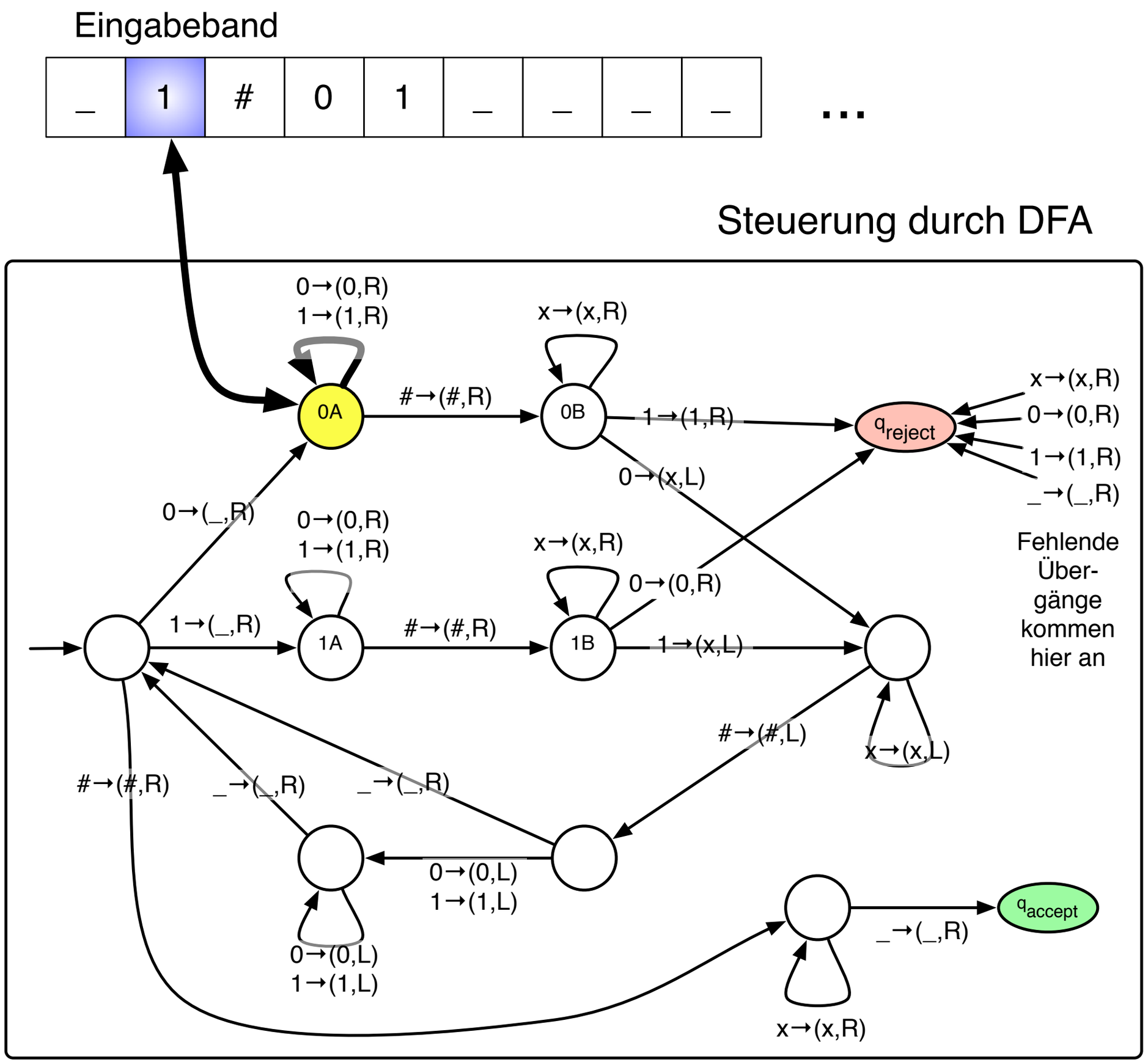
6

Eingabeband



Steuerung durch DFA



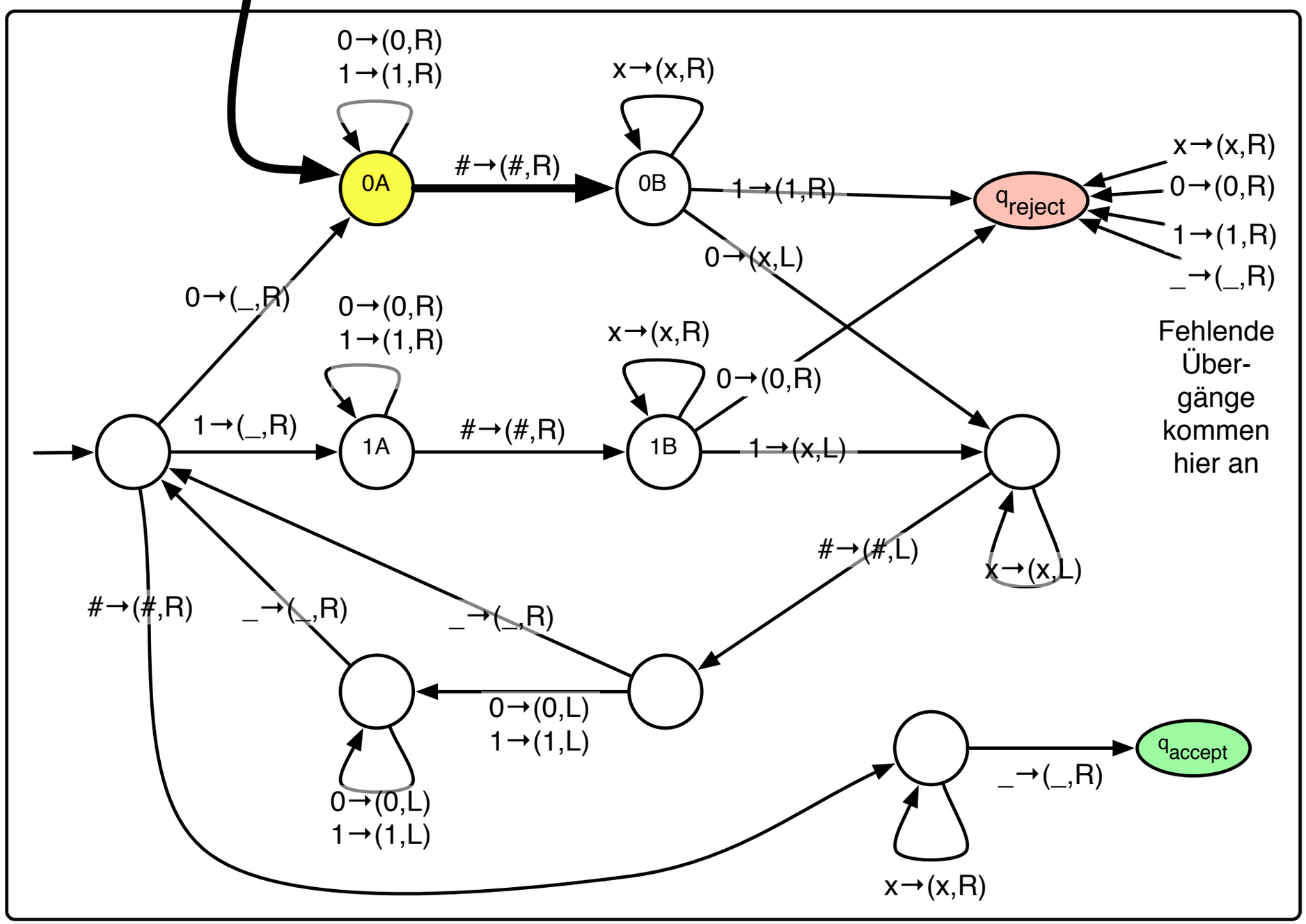

Eingabeband
_
1
#
0
1
_
_
_
_
...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
0→(x,L)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
1→(_,R)
1A
#→(#,R)
x→(x,R)
0→(0,R)
1B
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

8

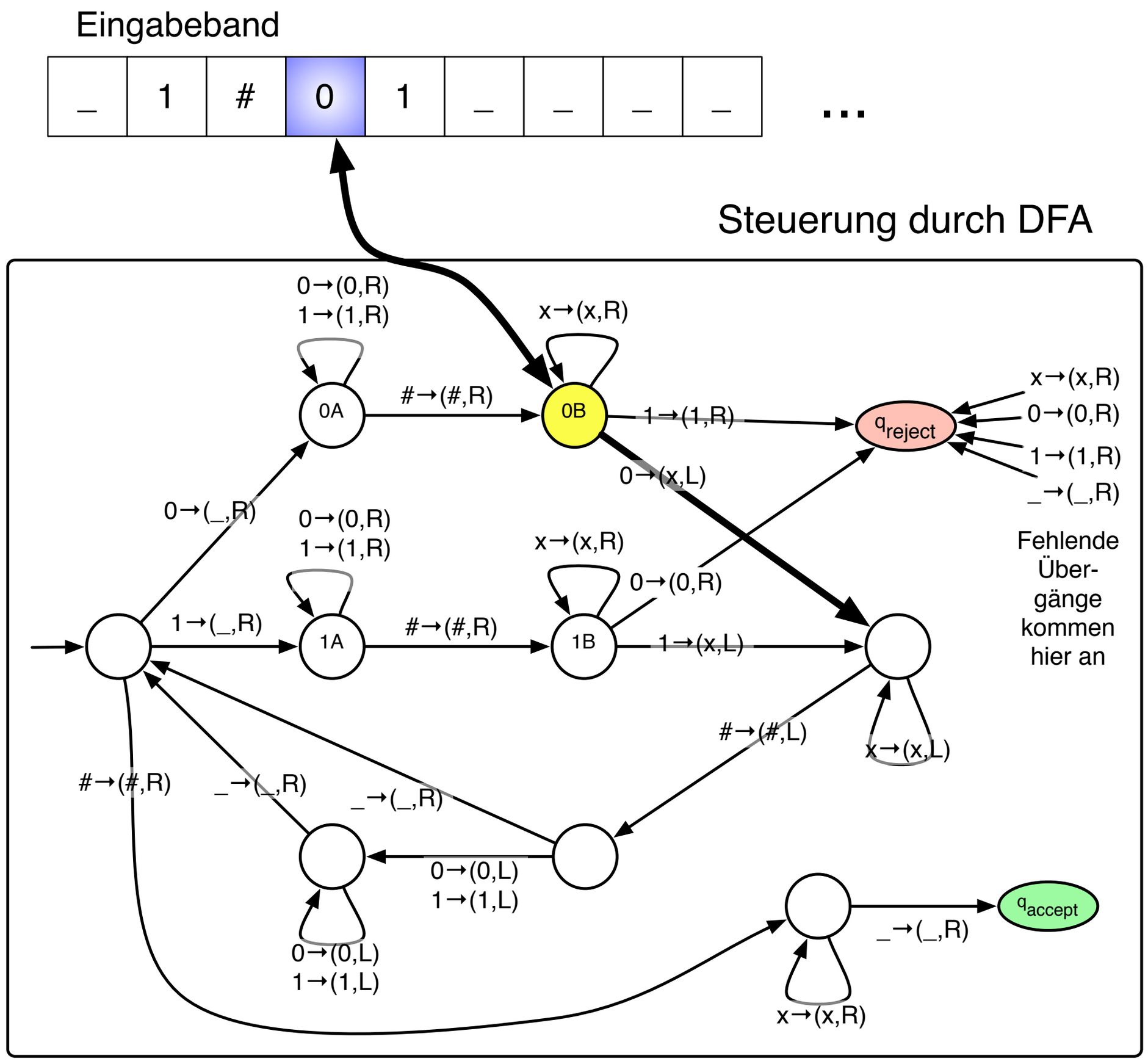
Eingabeband

_	1	#	0	1	_	_	_	_	...

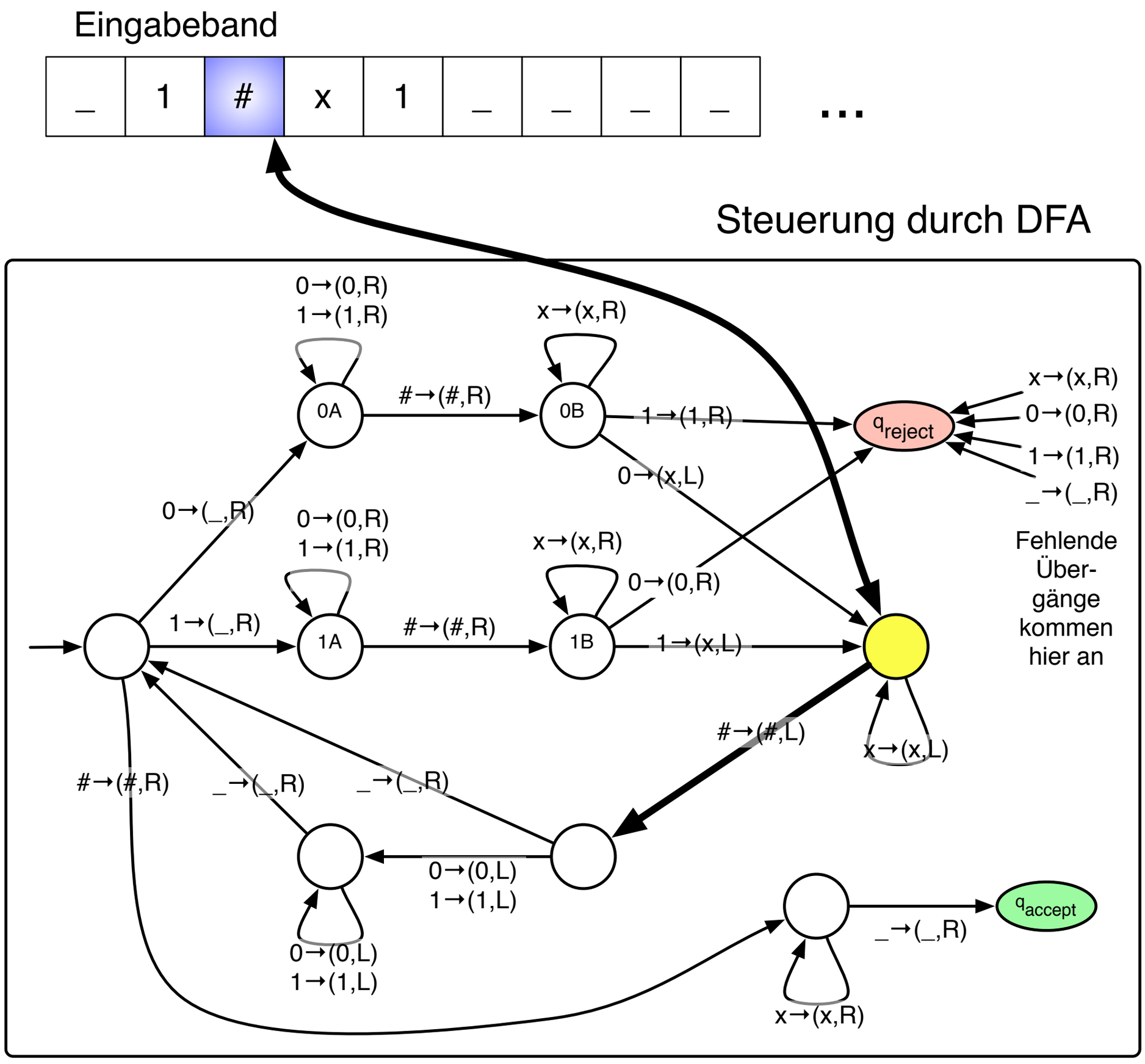
Steuerung durch DFA

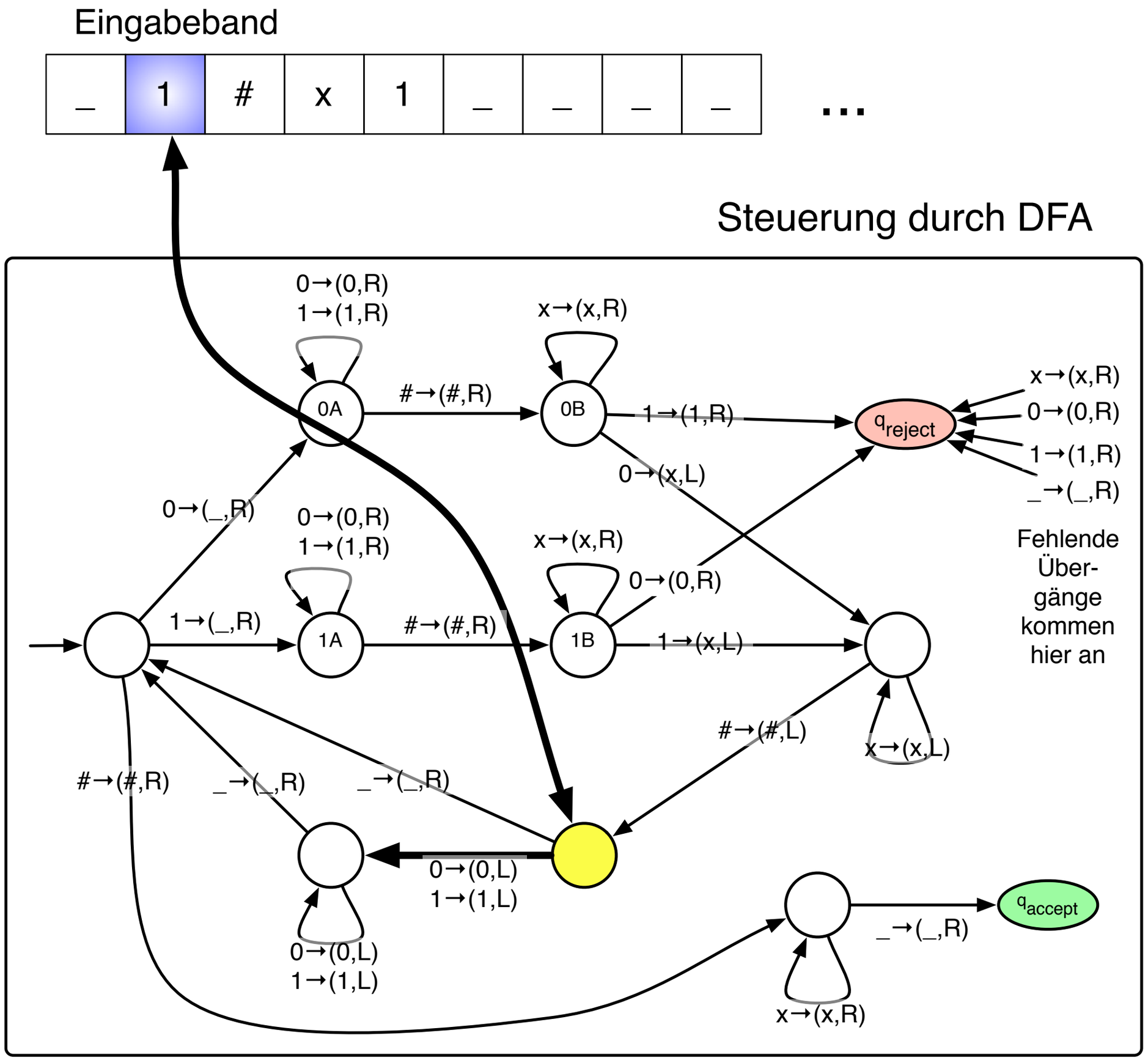


9

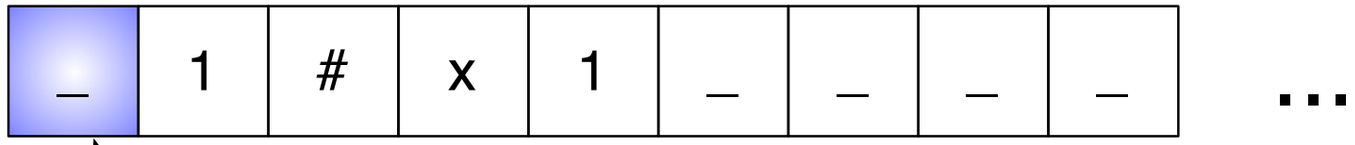

Eingabeband
_ 1 # 0 1 _ _ _ _ _
...
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
1→(_,R)
1A
#→(#,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

10

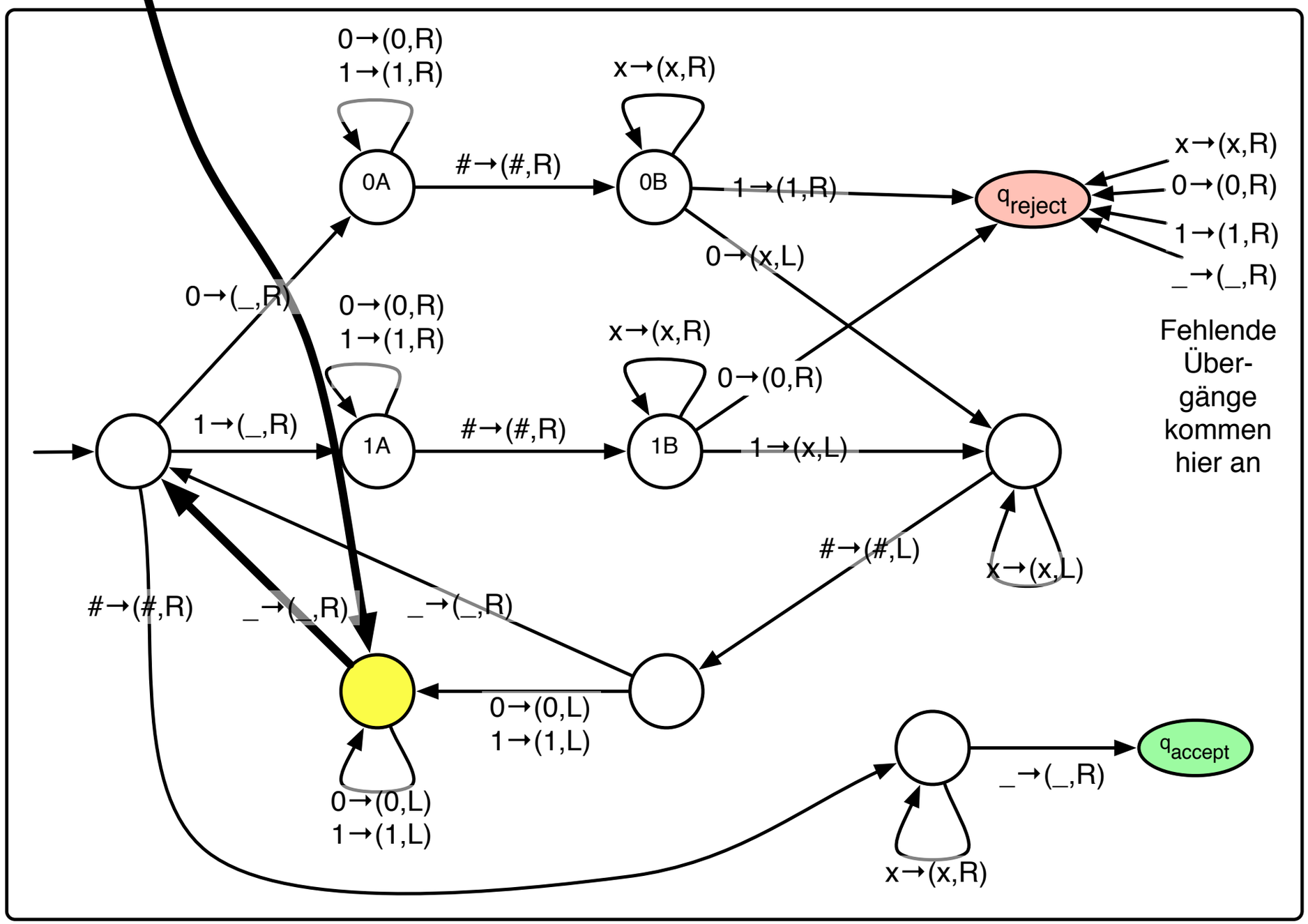

Eingabeband
_
1
#
x
1
_
_
_
_
...
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
0→(x,L)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
1→(_,R)
1A
#→(#,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)


Eingabeband
_
1
#
x
1
_
_
_
_
...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
0→(x,L)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
1A
1→(_,R)
#→(#,R)
x→(x,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

Eingabeband



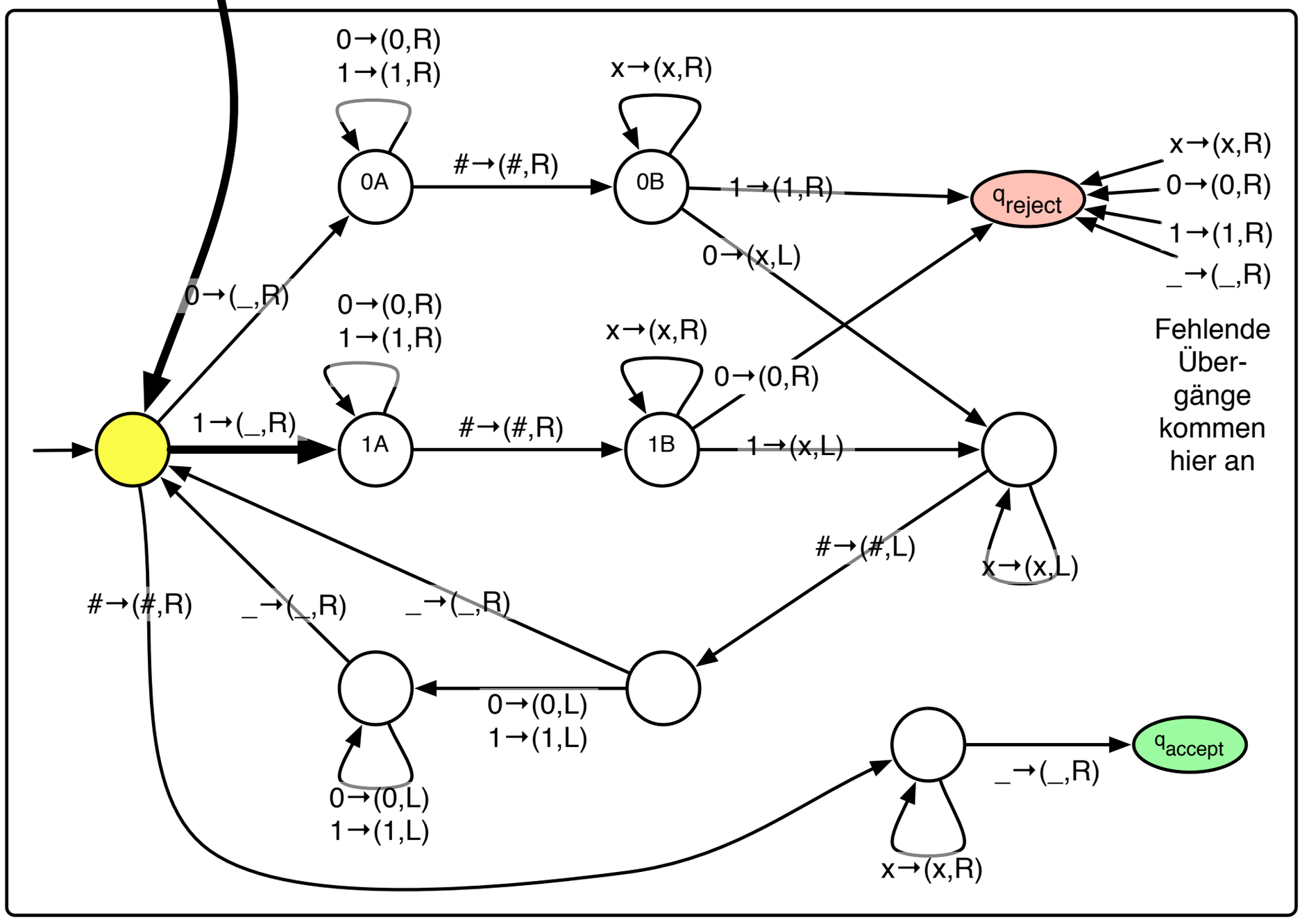
Steuerung durch DFA



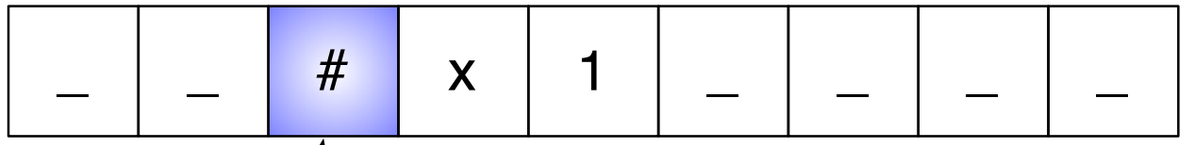
Eingabeband

_	1	#	x	1	_	_	_	_	...

Steuerung durch DFA

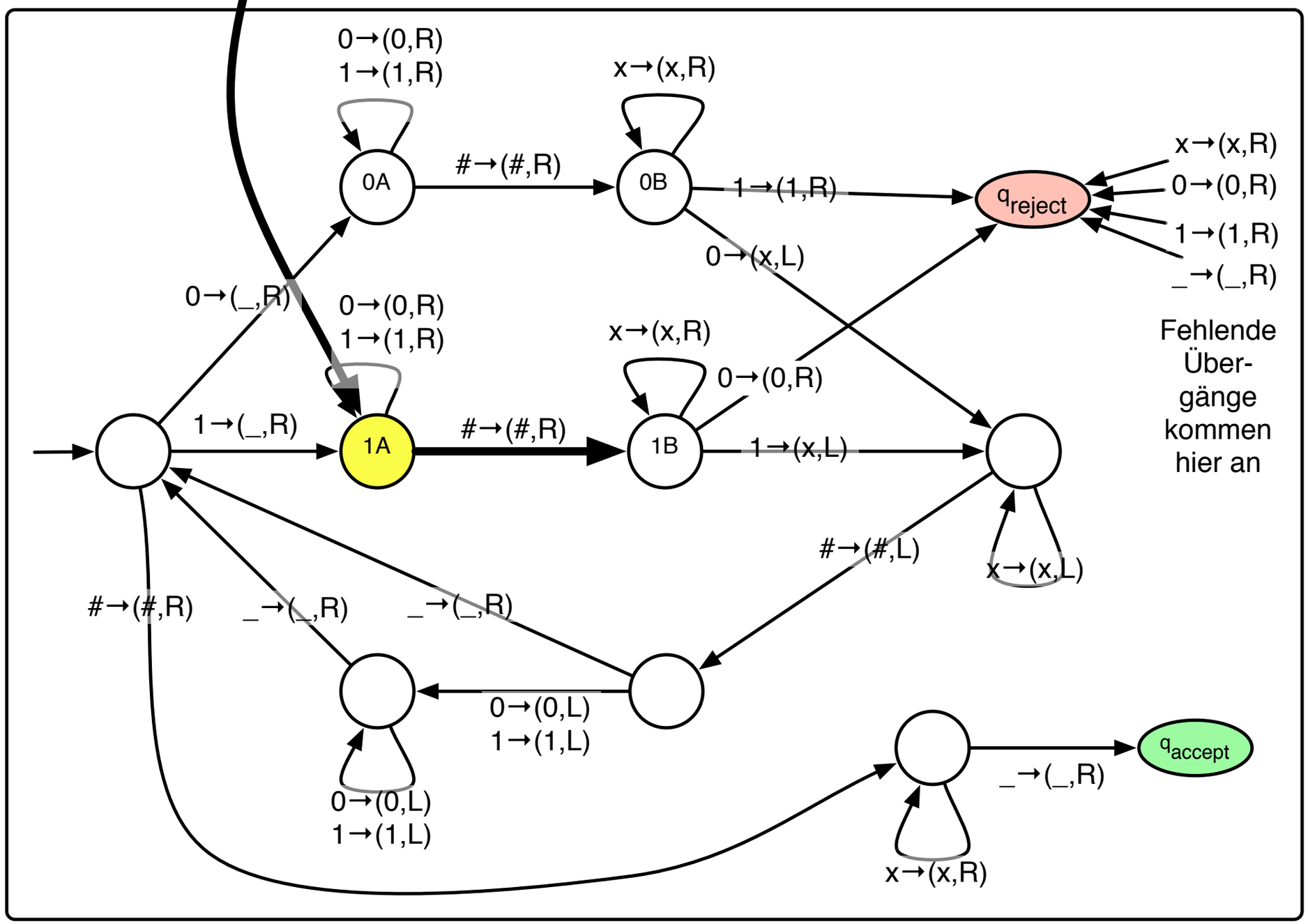


Eingabeband

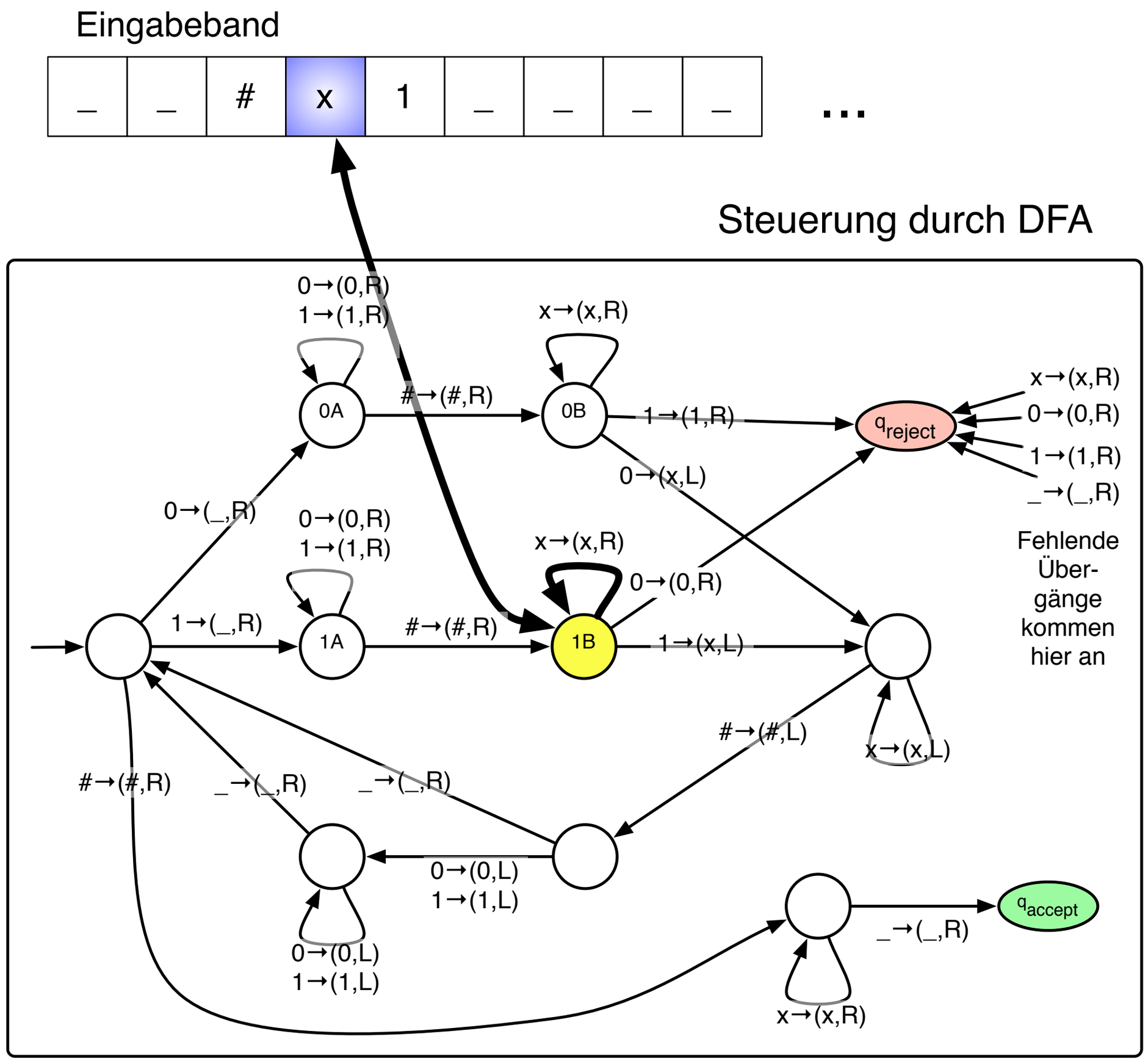


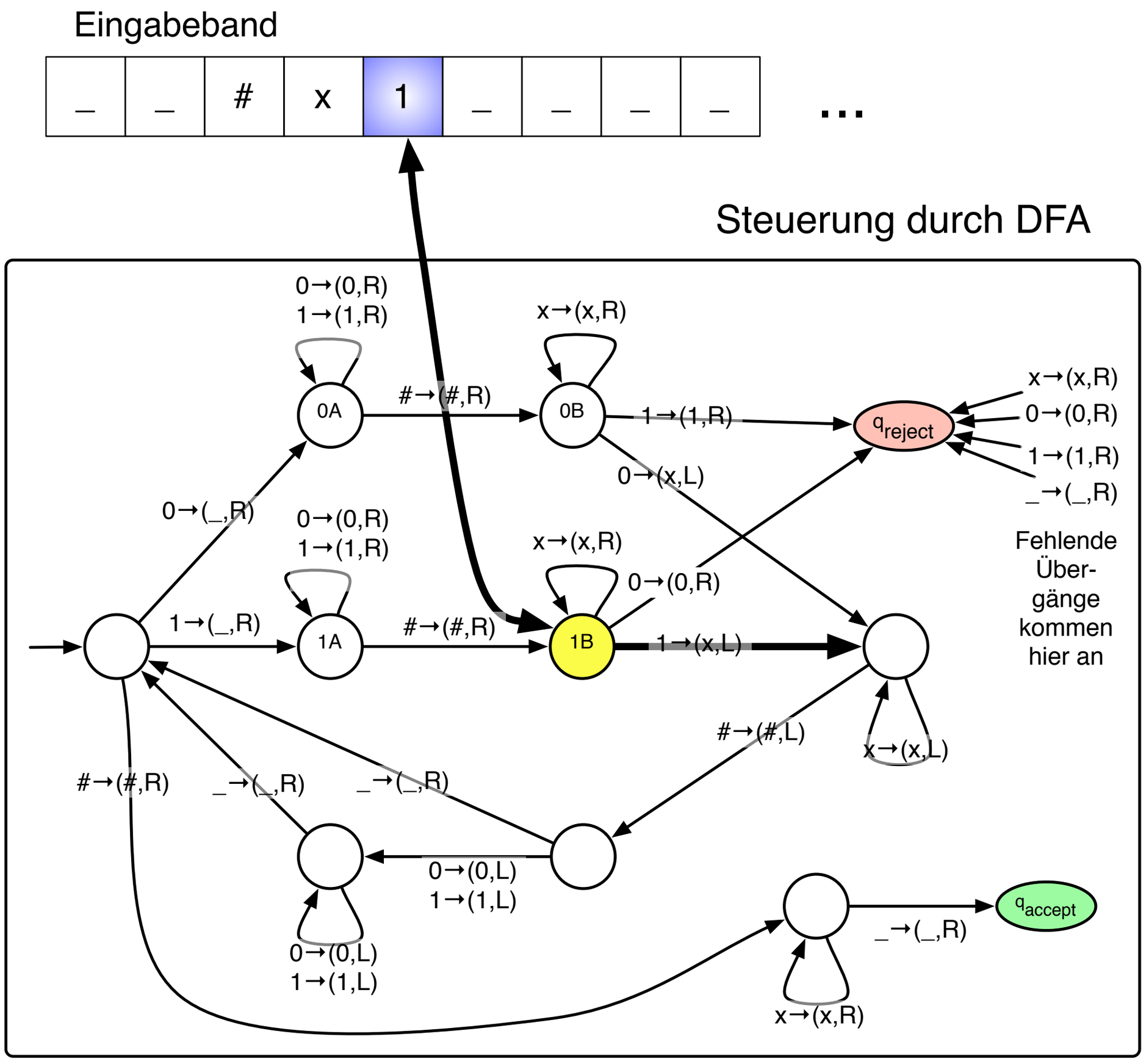
...

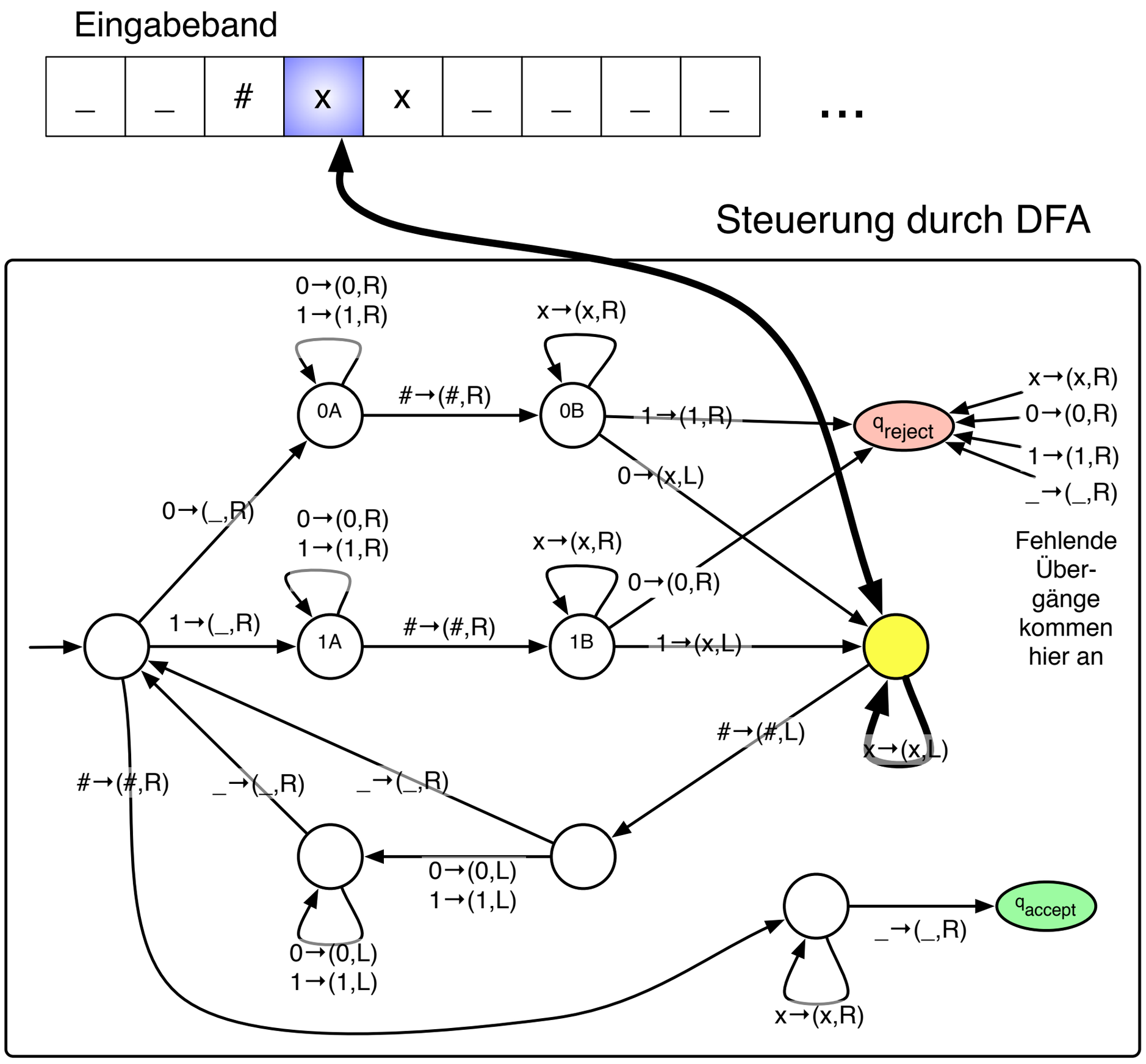
Steuerung durch DFA

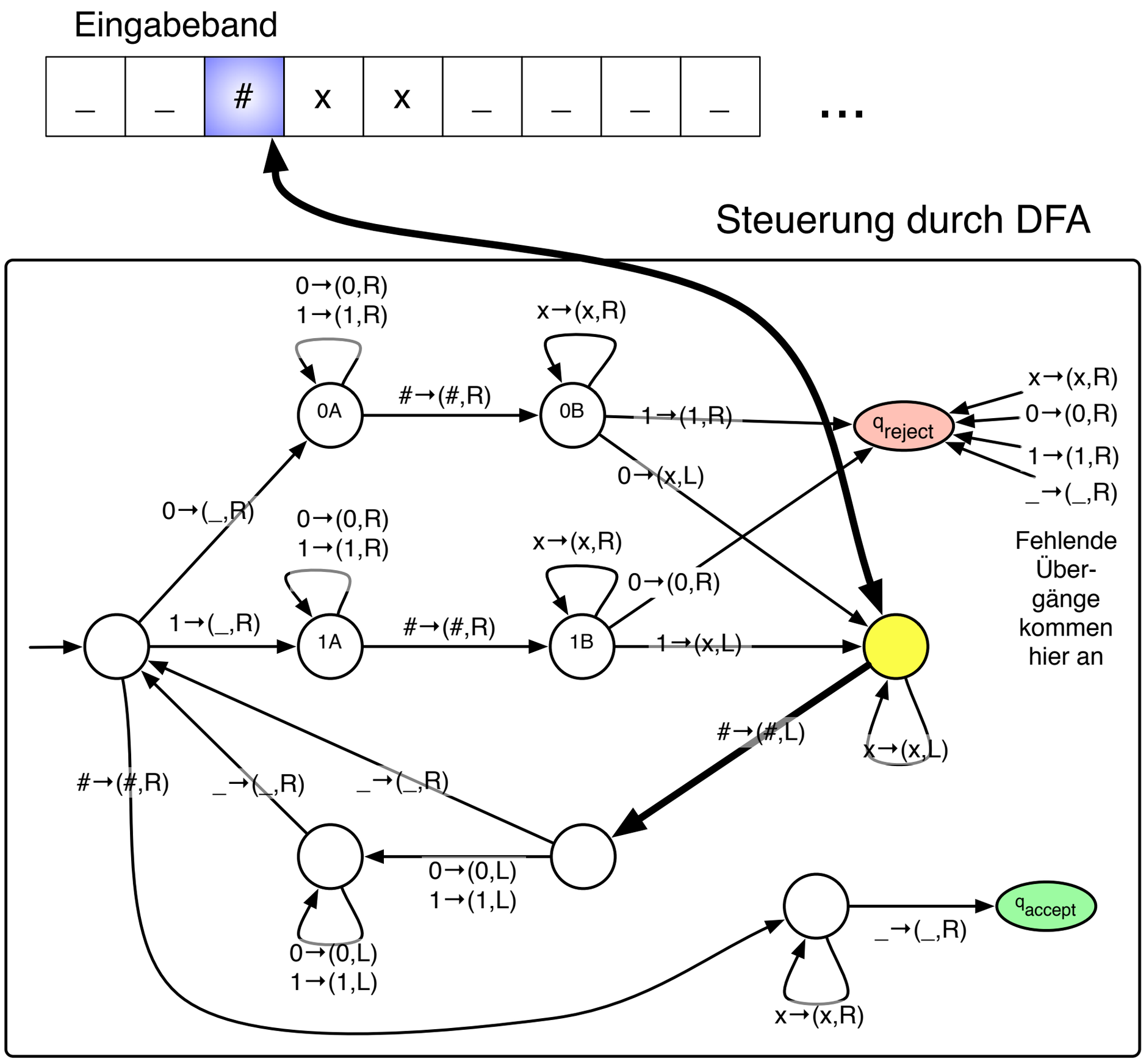


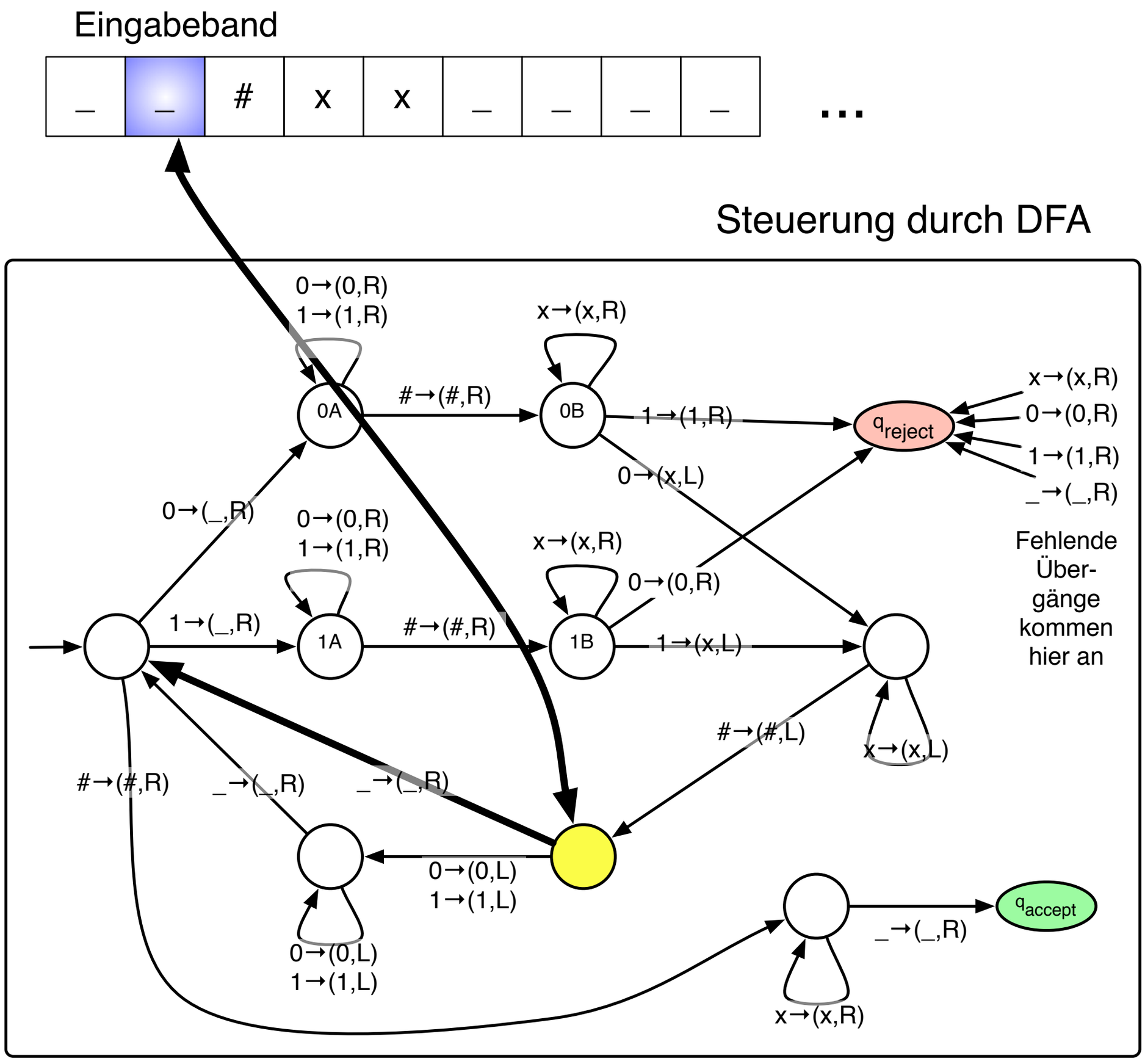
Fehlende Über-gänge kommen hier an


Eingabeband
_ _ # x 1 _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
0→(x,L)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
1→(_,R)
1A
#→(#,R)
1B
x→(x,R)
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

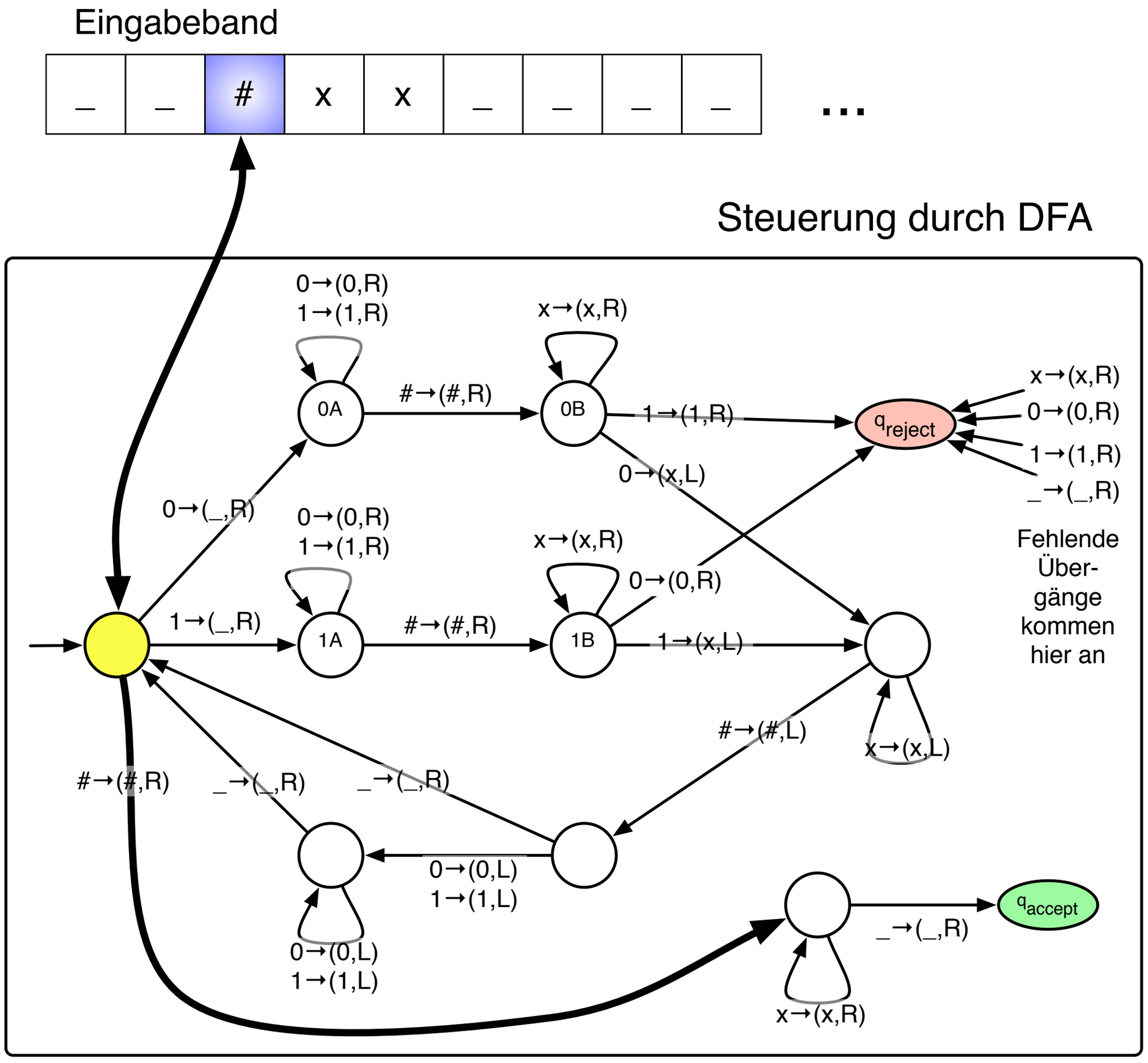

Eingabeband
_ _ # x 1 _ _ _ _
...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
0→(x,L)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
1A
1→(_,R)
#→(#,R)
1B
x→(x,R)
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

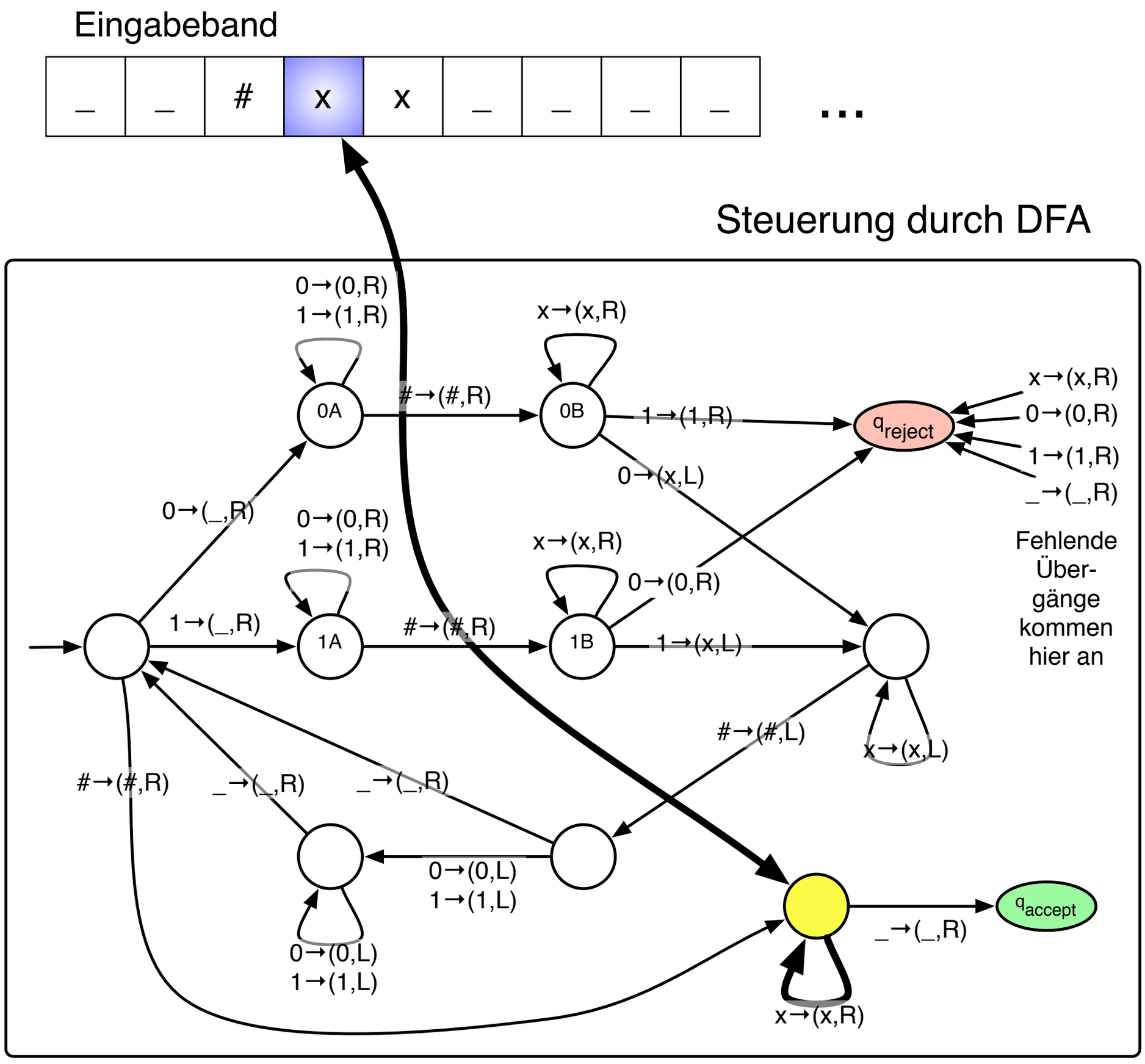

Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
0→(x,L)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
1A
1→(_,R)
#→(#,R)
x→(x,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

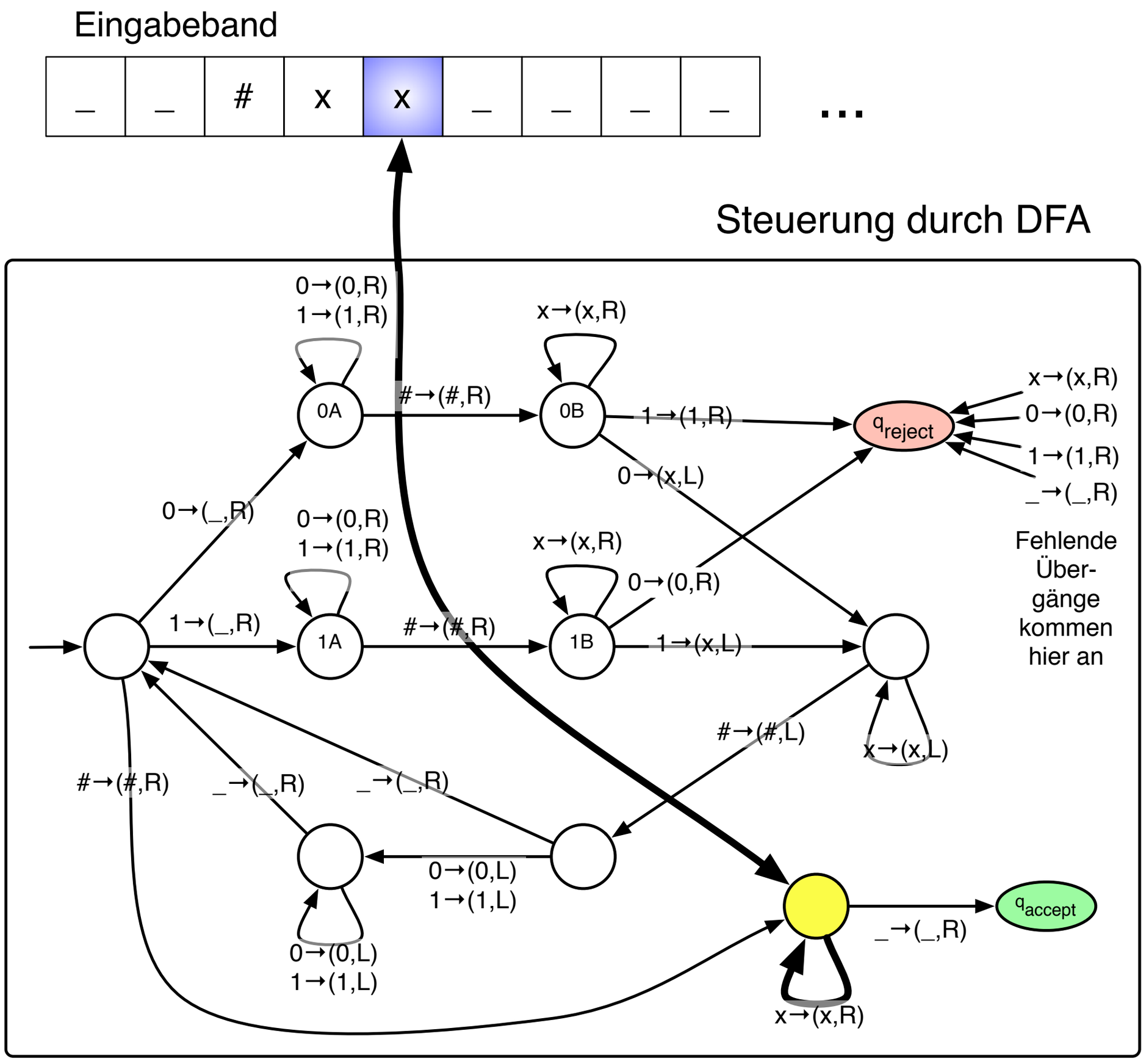

Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
0→(x,L)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
1A
1→(_,R)
#→(#,R)
x→(x,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)

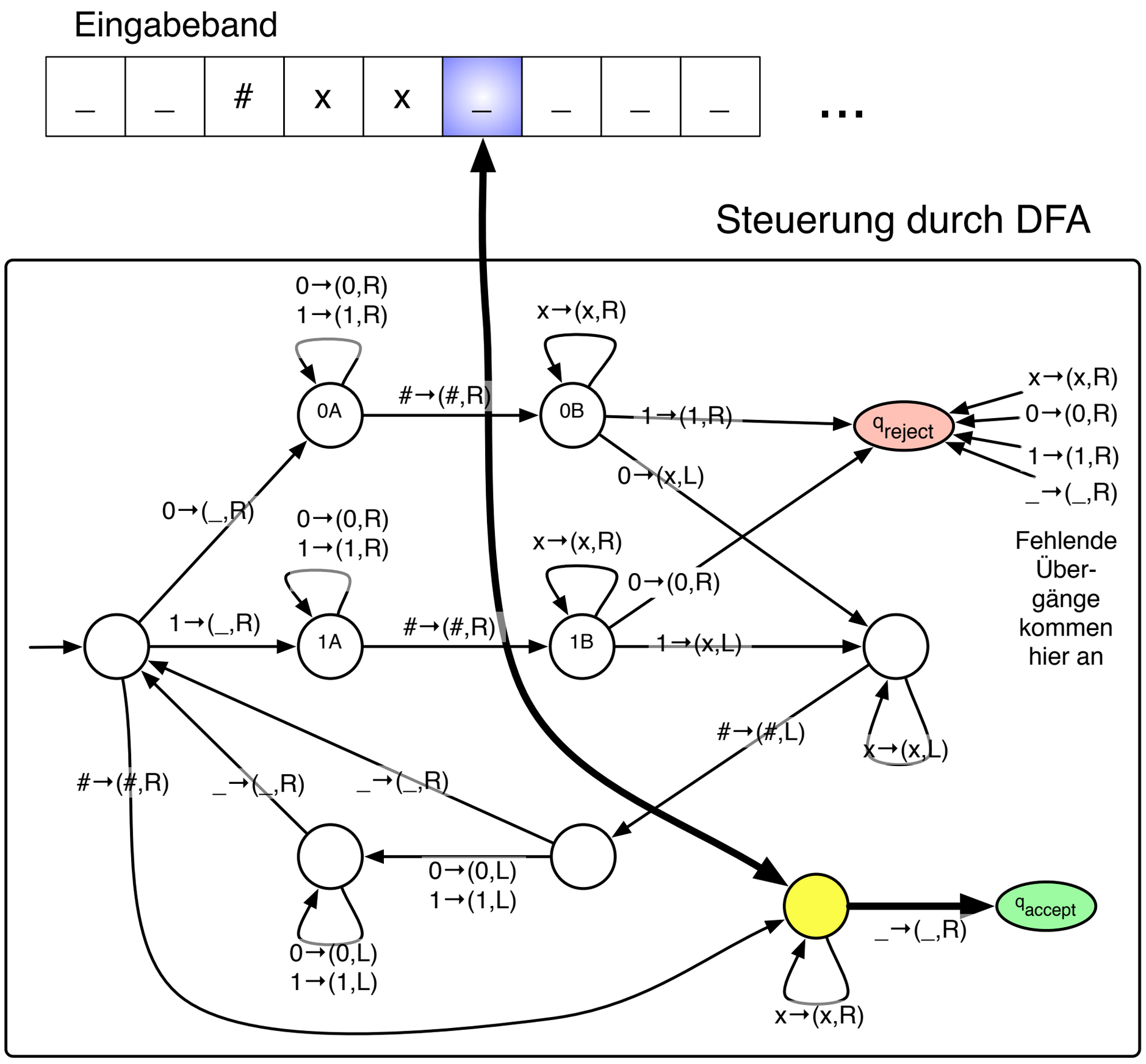

Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
qreject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
Fehlende Über-gänge kommen hier an
1→(_,R)
1A
#→(#,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
qaccept
_→(_,R)
0→(0,L)
1→(1,L)
x→(x,R)

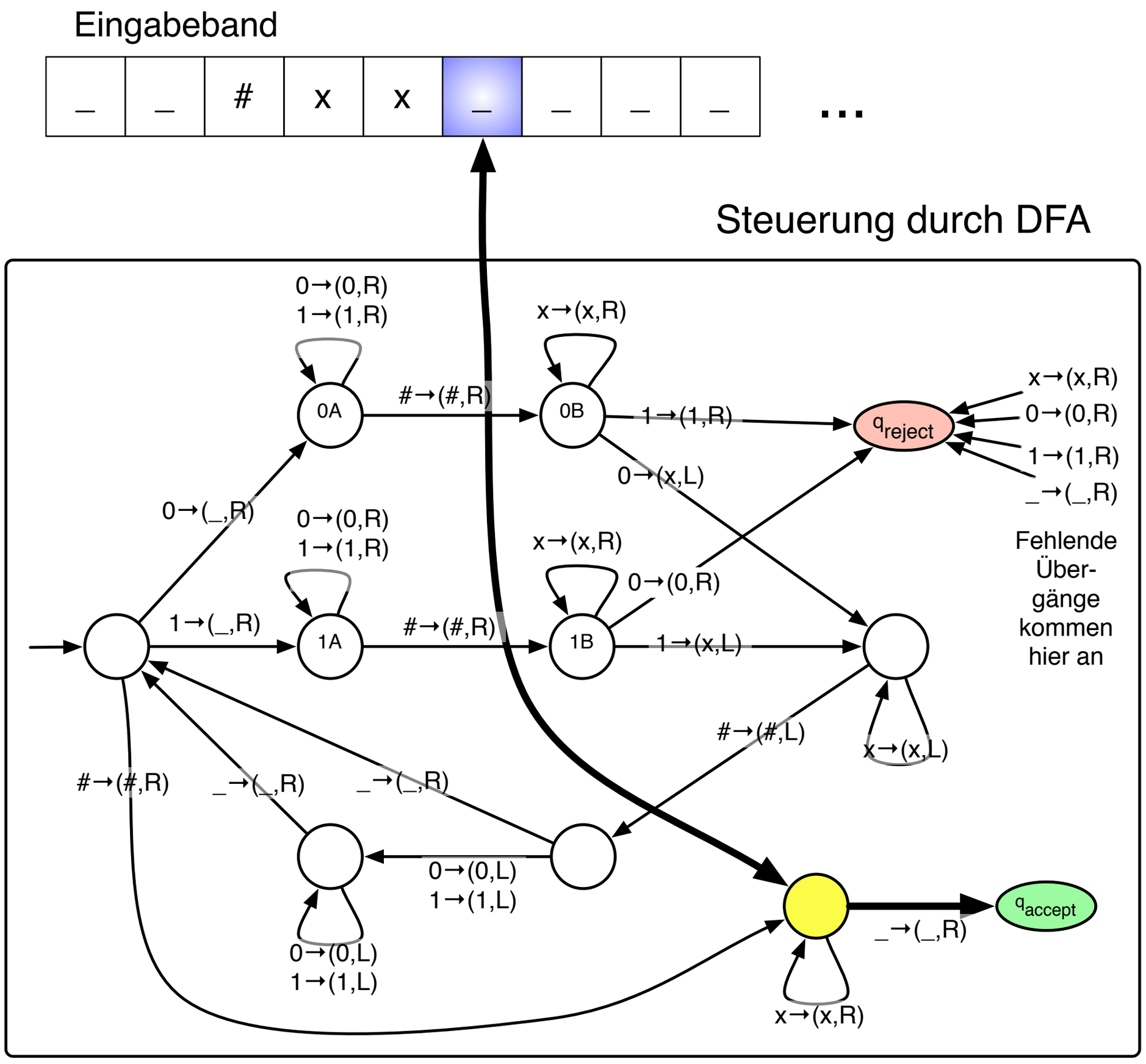
20

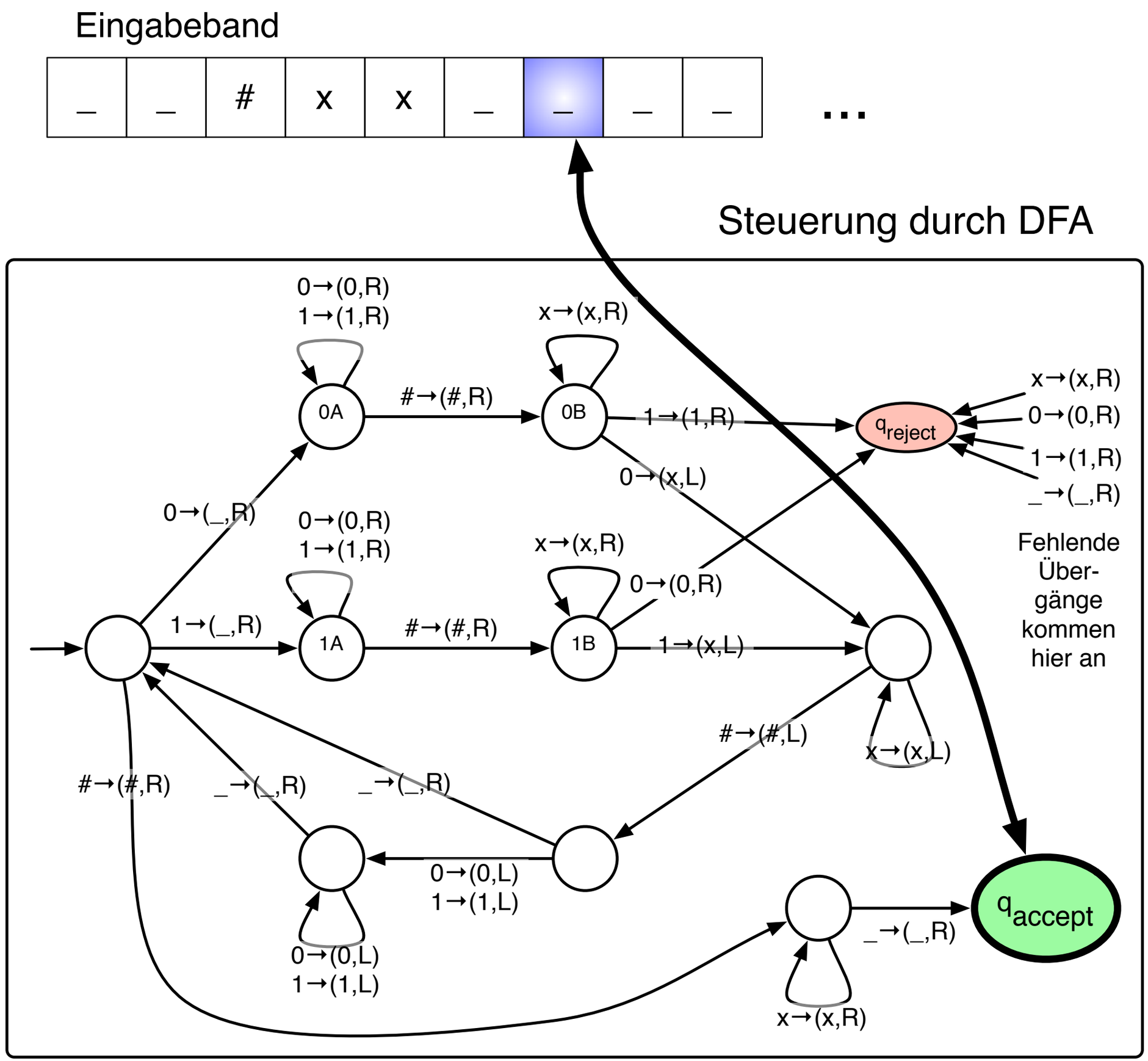

Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
Fehlende Über-gänge kommen hier an
0→(0,R)
1→(_,R)
1A
#→(#,R)
1B
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)


Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
0→(0,R)
1→(_,R)
1A
#→(#,R)
1B
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
x→(x,R)


Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
qreject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
Fehlende Über-gänge kommen hier an
1→(_,R)
1A
#→(#,R)
1B
1→(x,L)
0→(0,R)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
qaccept
_→(_,R)
0→(0,L)
1→(1,L)
x→(x,R)


Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
x→(x,R)
0A
#→(#,R)
0B
1→(1,R)
q_reject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
Fehlende Über-gänge kommen hier an
1→(_,R)
1A
#→(#,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
q_accept
_→(_,R)
0→(0,L)
1→(1,L)
x→(x,R)


Eingabeband
_ _ # x x _ _ _ _ ...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
q_{reject}
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
Fehlende Über-gänge kommen hier an
0→(_,R)
0→(0,R)
1→(1,R)
1A
1→(_,R)
#→(#,R)
x→(x,R)
1B
0→(0,R)
1→(x,L)
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
0→(0,L)
1→(1,L)
q_{accept}
_→(_,R)
x→(x,R)

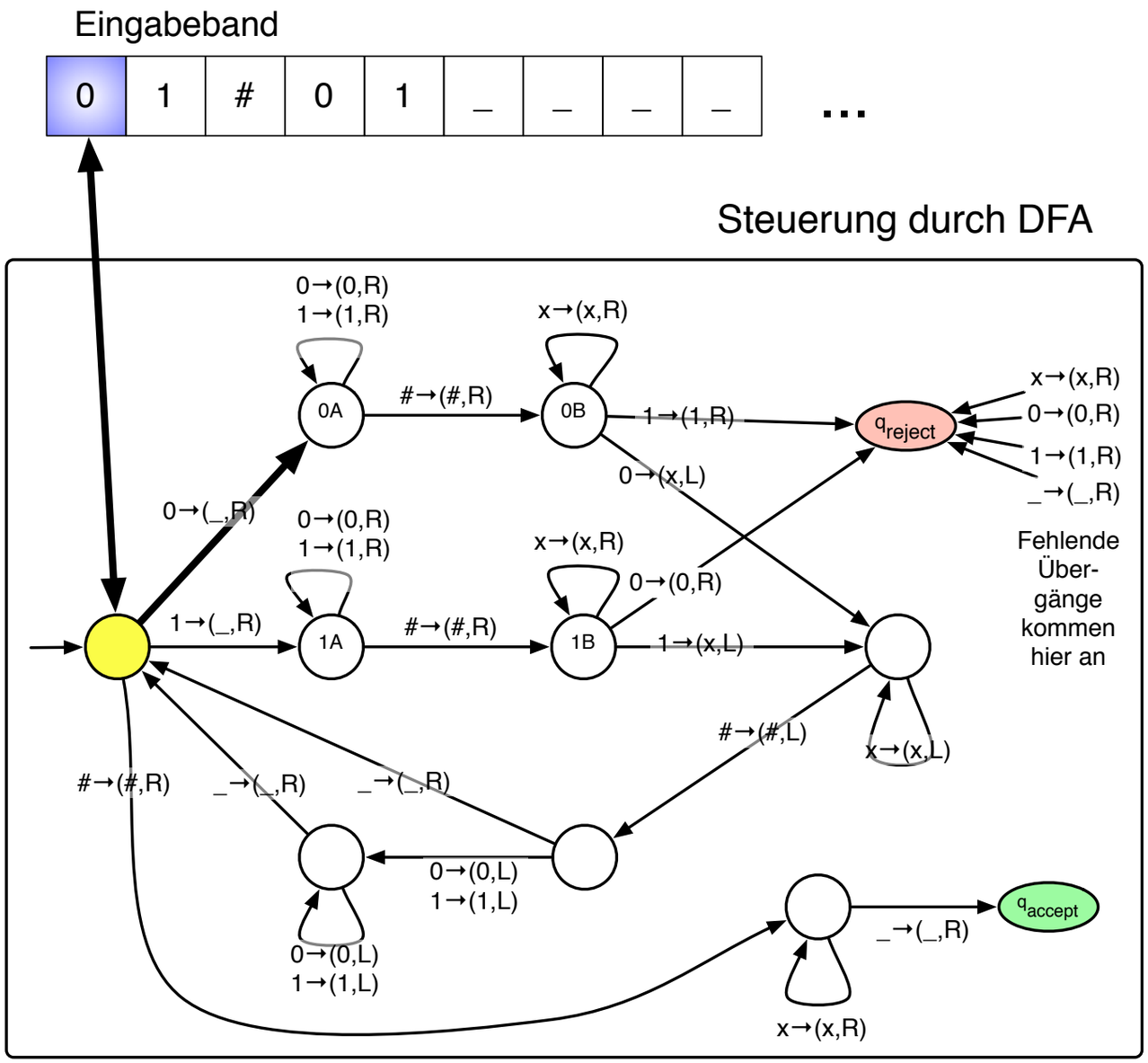

Eingabeband
_ _ # x x _ _ _ _
...
Steuerung durch DFA
0→(0,R)
1→(1,R)
0A
#→(#,R)
x→(x,R)
0B
1→(1,R)
qreject
x→(x,R)
0→(0,R)
1→(1,R)
_→(_,R)
0→(x,L)
0→(_,R)
0→(0,R)
1→(1,R)
x→(x,R)
0→(0,R)
1→(_,R)
1A
#→(#,R)
1B
1→(x,L)
Fehlende Über-gänge kommen hier an
#→(#,L)
x→(x,L)
#→(#,R)
_→(_,R)
_→(_,R)
0→(0,L)
1→(1,L)
qaccept
_→(_,R)
0→(0,L)
1→(1,L)
x→(x,R)

DTM

- **Eine (deterministische 1-Band) Turingmaschine (DTM) wird beschrieben durch ein 7-Tupel**
  - $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.
  - Q, Σ, Γ sind endliche, nichtleere Mengen und es gilt:
    - $\Sigma \subseteq \Gamma$, $q_0 \in Q$
    - $\_ \in \Gamma \cap \Sigma$ ist das Blanksymbol.
    - Σ ist das Eingabealphabet
    - Γ das Bandalphabet.
  - Zustandsmenge Q
    - $q_0 \in Q$ ist der Startzustand.
    - $q_{accept} \in Q$ ist der akzeptierende Endzustand
    - $q_{reject} \in Q$ ist der ablehnende Endzustand
  - **(partielle) Übergangsfunktion**
    - $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
    - nicht definiert für $q \in \{q_{accept}, q_{reject}\} \subseteq Q$ definiert

Arbeitsweise einer DTM

- **Initial:**
  - Eingabe steht links auf dem Band
  - Der Rest des Bands ist leer
  - Kopf befindet sich ganz links
- **Berechnungen finden entsprechend der Übergangsfunktion statt**
  - Wenn der Kopf sich am linken Ende befindet und nach links bewegen soll, bleibt er an seiner Position
  - Wenn der Kopf sich am rechten Ende befindet, wird ein _ an das Band angehängt.
- **Wenn q_{accept} oder q_{reject} erreicht wird, ist die Bearbeitung beendet**



Konfiguration

- **Momentaufnahme einer DTM**
  - Bei Bandinschrift uv
  - dabei beginnt u am linken des Bandes und hinter v stehen nur Blanks
  - Zustand q,
  - Kopf auf erstem Zeichen von v

- **Konfiguration C = uqv**

Aufeinanderfolgende Konfigurationen

- **Gegeben: Konfigurationen C_1, C_2**
- **Wir sagen:**
  - Konfiguration C_1 führt zu C_2, falls die TM von C_1 in einem Schritt zu C_2 übergehen kann.
- **Formal:**
  - Seien $a,b,c \in \Gamma$, $u,v \in \Gamma^*$ und Zustände q_i,q_j gegeben
- **Wir sagen**
  - $u\ a\ q_i\ b\ v$ führt zu $u\ q_j\ c\ b\ v$, falls $\delta(q_i,b) = (q_j,c,L)$ und
  - $u\ a\ q_i\ b\ v$ führt zu $u\ a\ c\ q_j\ v$, falls $\delta(q_i,b) = (q_j,c,R)$

Konfigurationen

- **Startkonfiguration:**
  - q_0w, wobei w die Eingabe ist
- **Akzeptierende Konfiguration:**
  - Konfigurationen mit Zustand q_{accept}
- **Ablehnende Konfiguration:**
  - Konfigurationen mit Zustand q_{reject}
- **Haltende Konfiguration:**
  - akzeptierende oder ablehnende Konfigurationen

Akzeptanz der DTM

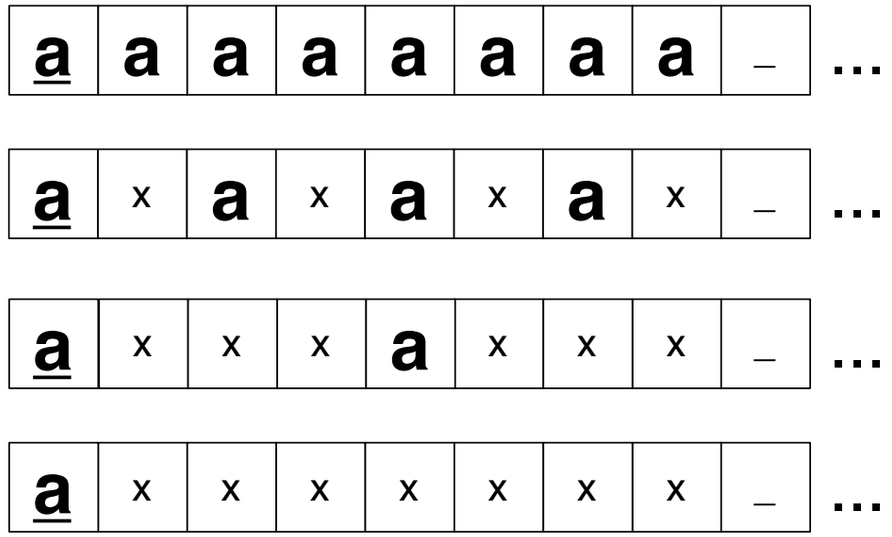
- **Eine Turingmaschine M akzeptiert eine Eingabe w, falls es eine Folge von Konfigurationen C_1, C_2, …, C_k gibt, so dass**
  - C_1 ist die Startkonfiguration von M bei Eingabe w
  - C_i führt zu C_{i+1}
  - C_k ist eine akzeptierende Konfiguration
- **Die von M akzeptierten Worte bilden die von M akzeptierte Sprache L(M)**
- **Eine Turingmaschine *entscheidet* eine Sprache, wenn jede Eingabe in einer haltende Konfiguration C_k resultiert**

Berechenbarkeitstheorie

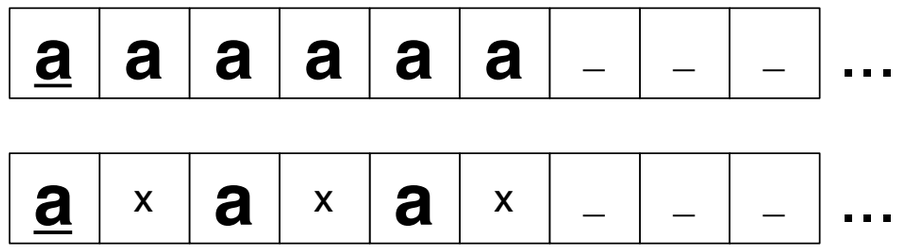
DTM-Beispiel

Beispiel für eine Turingmaschine

- **Gesucht: Turingmaschine, die**
  - $L = \{a^{2^n} \mid n \in \{0, 1, 2, \ldots\}\}$
  - entscheidet
- **Arbeitsweise:**
  - Wenn nur ein a auf dem Band ist, akzeptiere
  - Falls die Anzahl der a‘s ungerade ist, lehne ab
  - Gehe von links nach rechts über die Eingabe und ersetze jedes zweite a durch x
  - Bewege den Kopf zurück an das linke Ende



Akzeptiert



Abgelehnt

Beispiel

- **Definition der Turingmaschine:**
  - $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$

    $\Sigma = \{a\}$

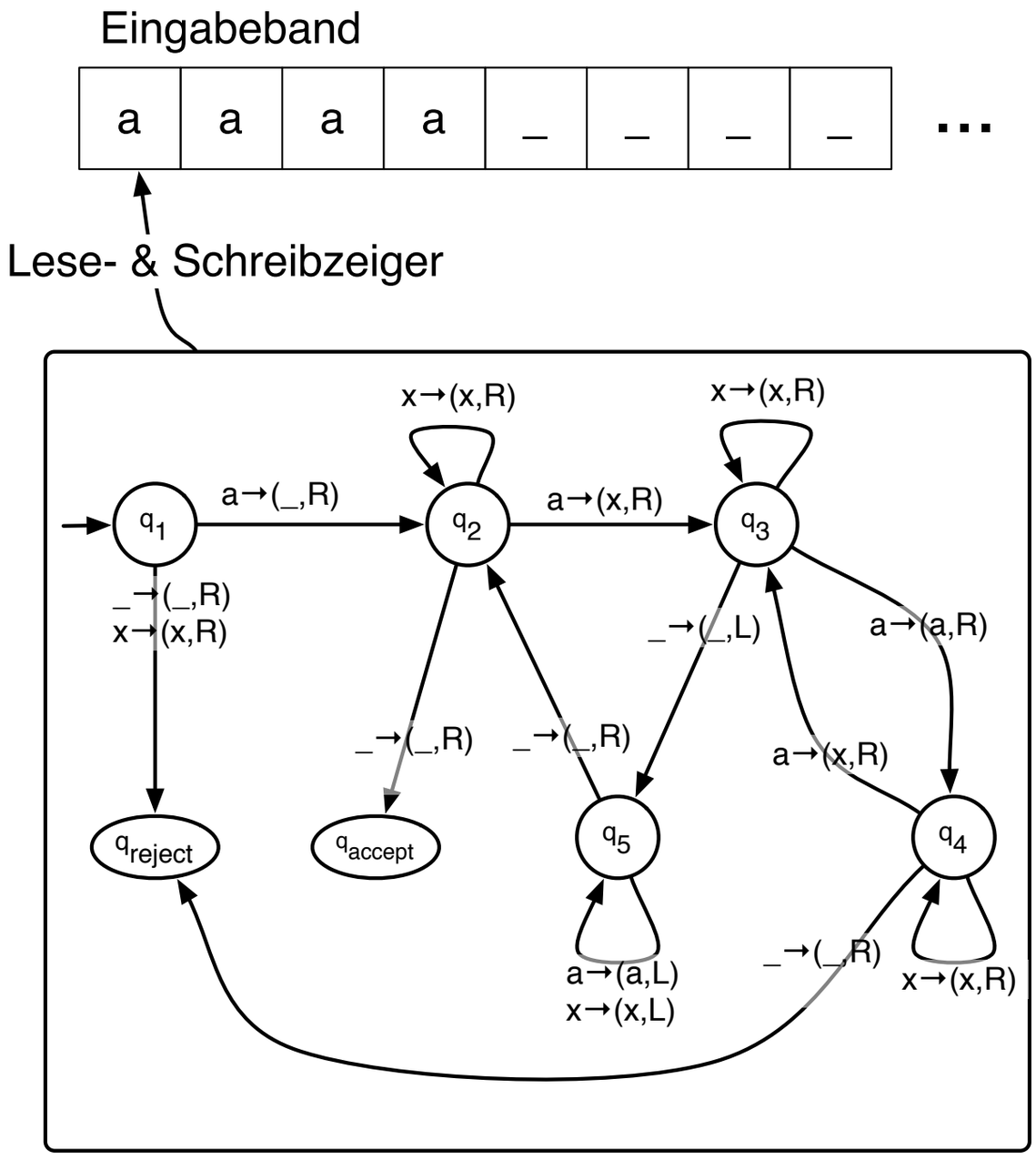
    $\Gamma = \{a, x, \_\}$
- **Besondere Zustände:**
  - Startzustand q_1
  - Akzeptierender Endzustand q_{accept}
  - Ablehnender Endzustand q_{reject}
- **Übergangsfunktion** δ

δ	a	x	_
q_1	$(q_2, \_, R)$	(q_{reject}, x, R)	$(q_{reject}, \_, R)$
q_2	(q_3, x, R)	(q_2, x, R)	$(q_{accept}, \_, R)$
q_3	(q_4, a, R)	(q_3, x, R)	$(q_5, \_, L)$
q_4	(q_3, x, R)	(q_4, x, R)	$(q_{reject}, \_, R)$
q_5	(q_5, a, L)	(q_5, x, L)	$(q_2, \_, R)$

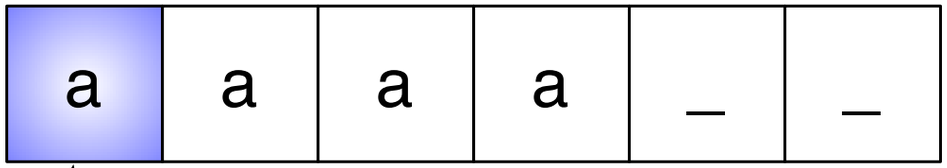
Abarbeitung eines Beispielworts

- **Beispiel**
  - Wort: 0000
- **Startkonfiguration:**
  - $q_1 0000$
- **Übergangsfunktion**

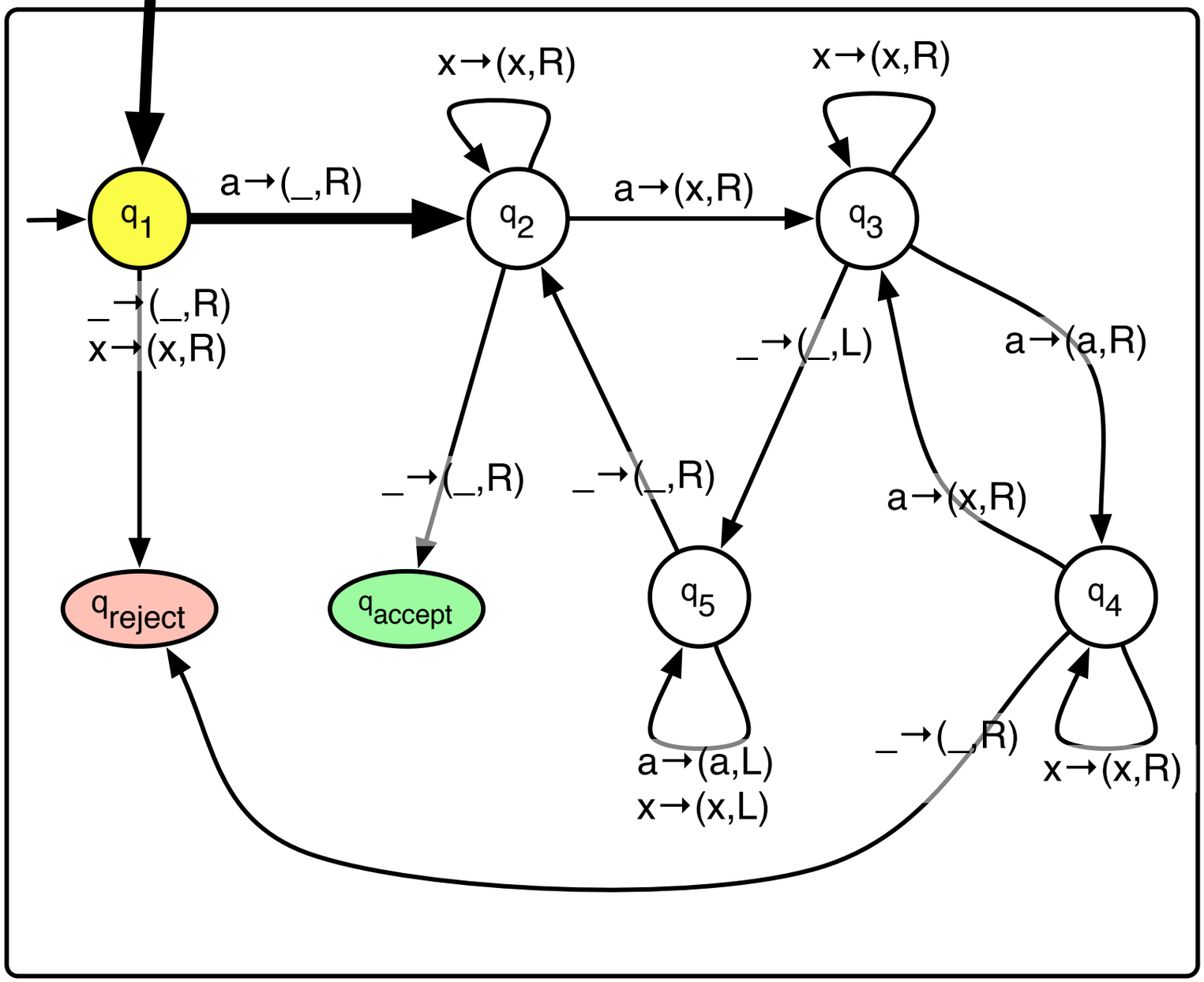
δ	a	x	_
q_1	$(q_2, \_, R)$	(q_{reject}, x, R)	$(q_{reject}, \_, R)$
q_2	(q_3, x, R)	(q_2, x, R)	$(q_{accept}, \_, R)$
q_3	(q_4, a, R)	(q_3, x, R)	$(q_5, \_, L)$
q_4	(q_3, x, R)	(q_4, x, R)	$(q_{reject}, \_, R)$
q_5	(q_5, a, L)	(q_5, x, L)	$(q_2, \_, R)$



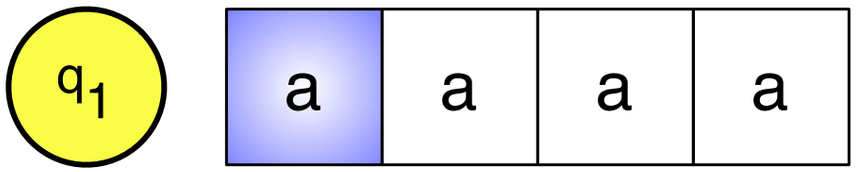
Eingabeband

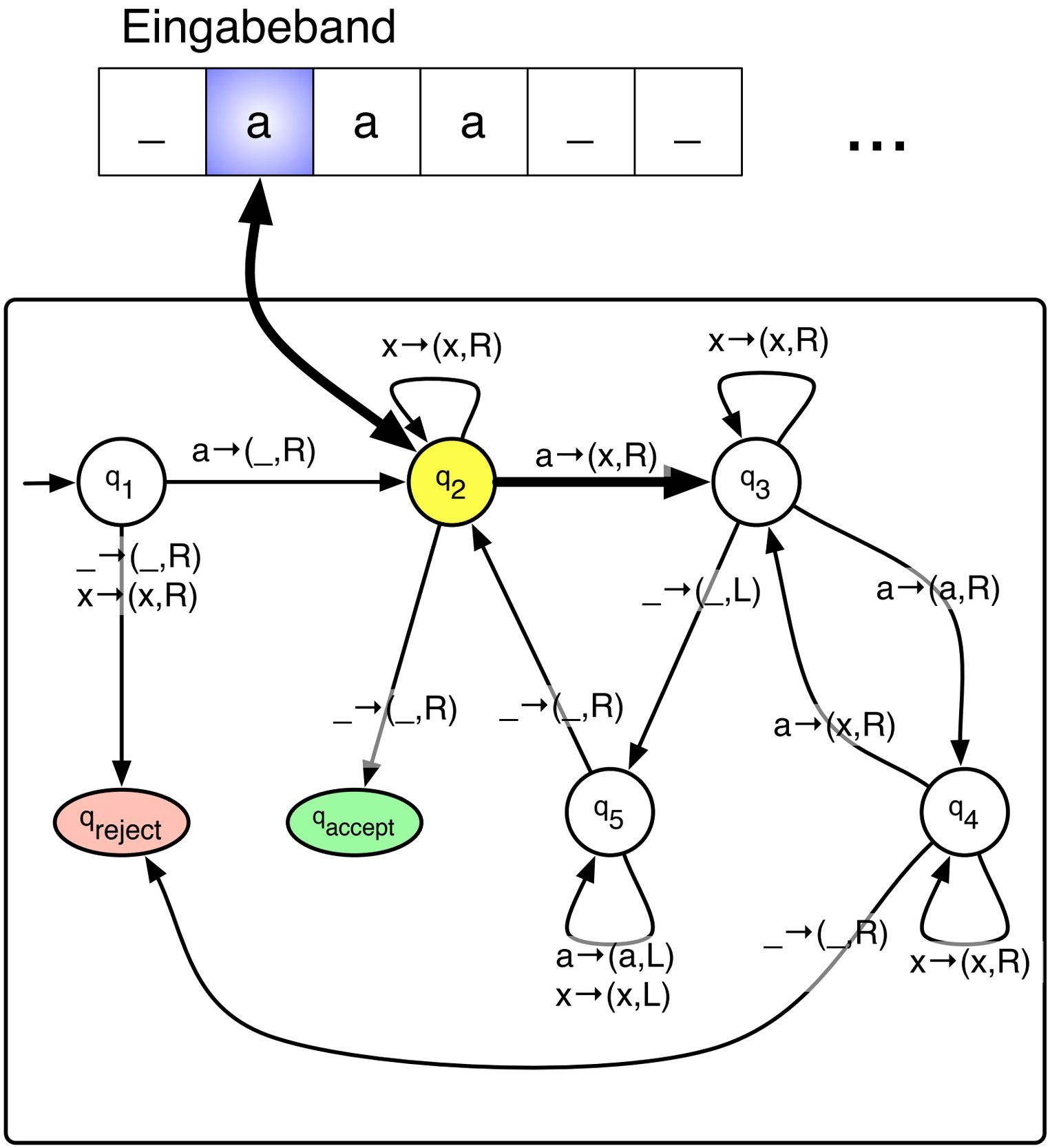


...

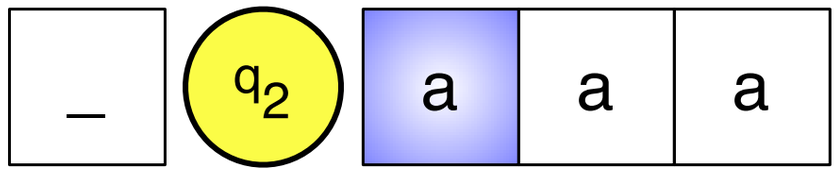


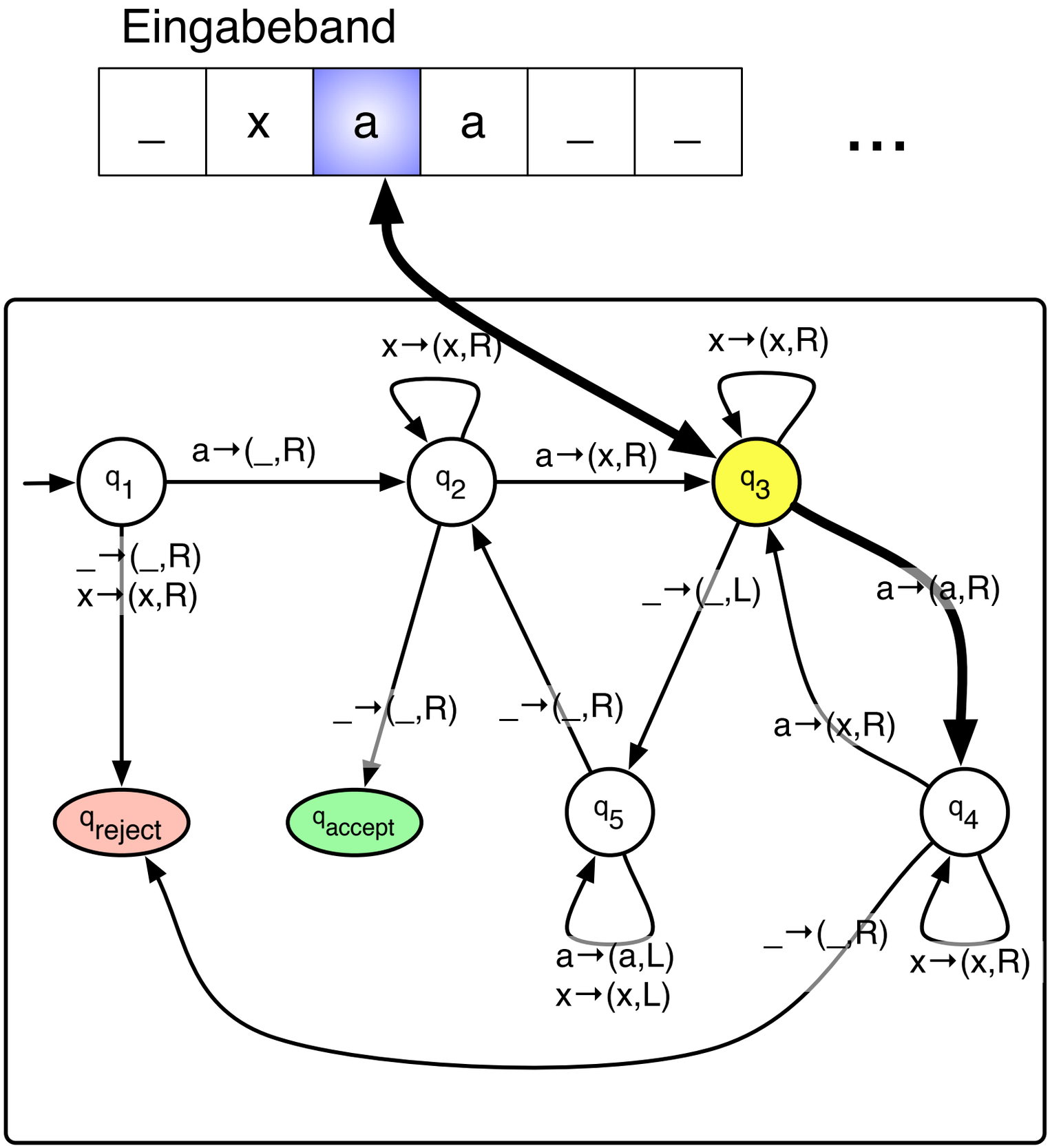
Konfiguration



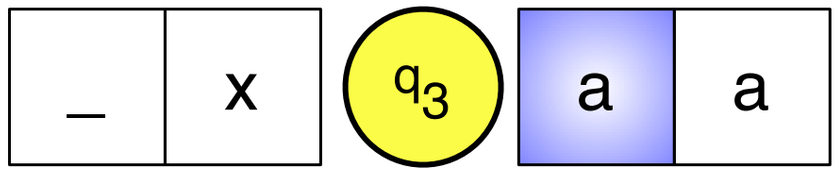


Konfiguration

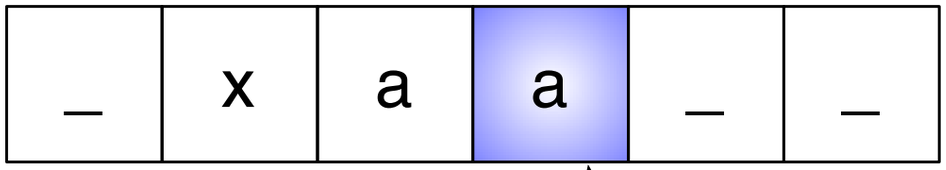




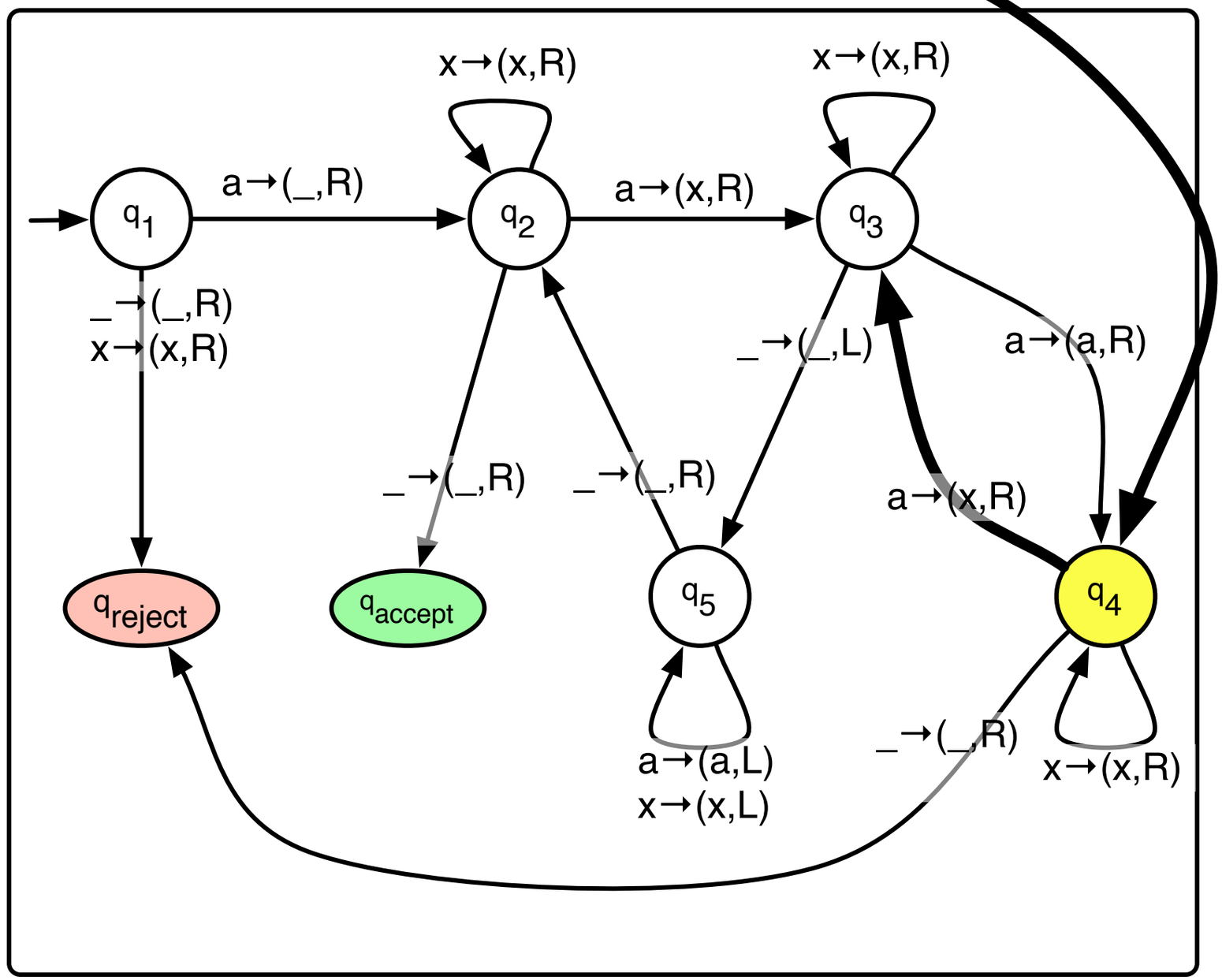
Konfiguration



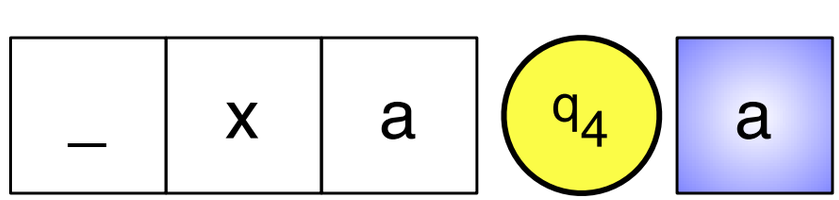
Eingabeband



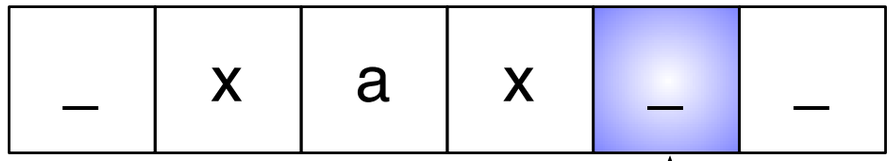
...



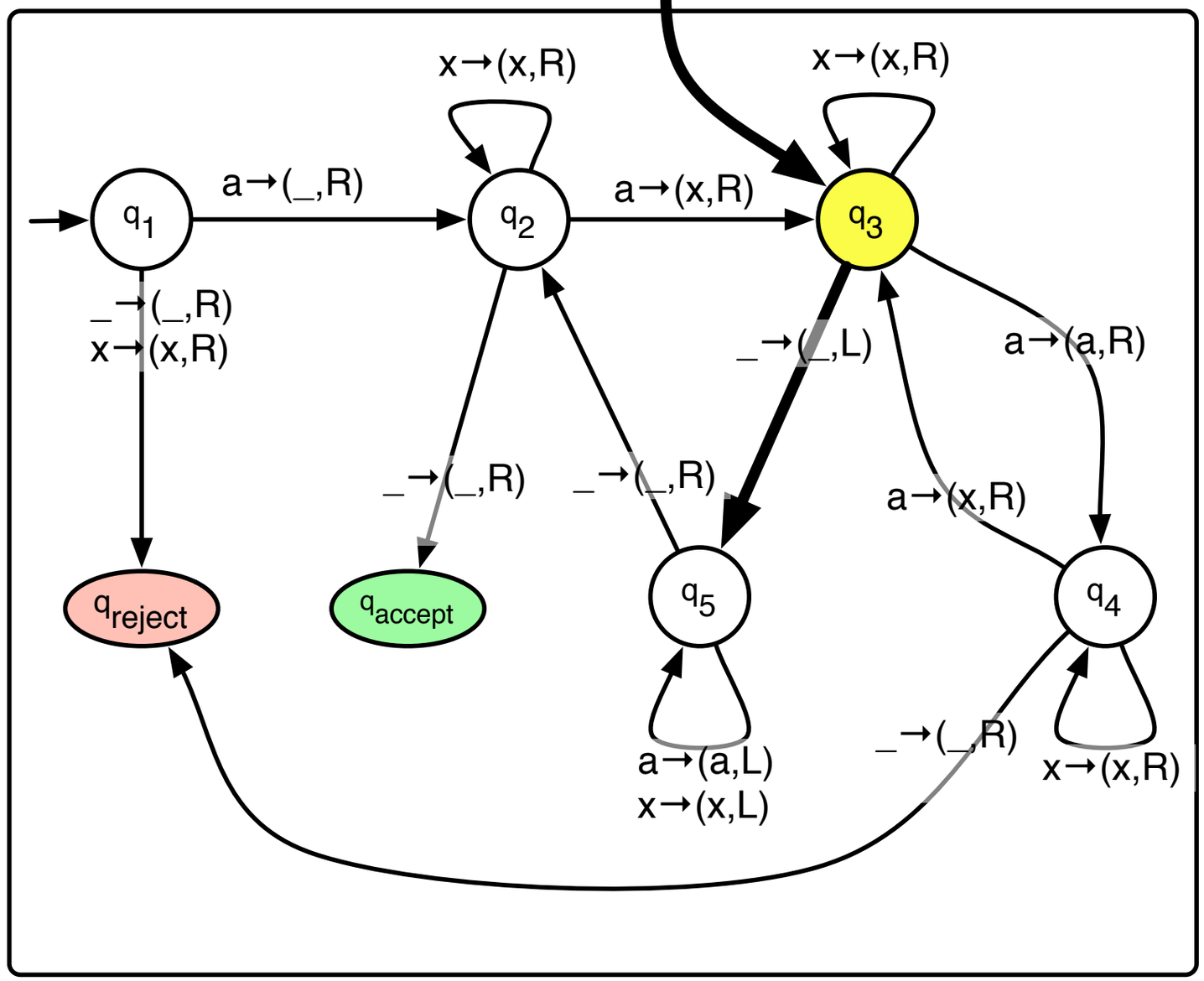
Konfiguration



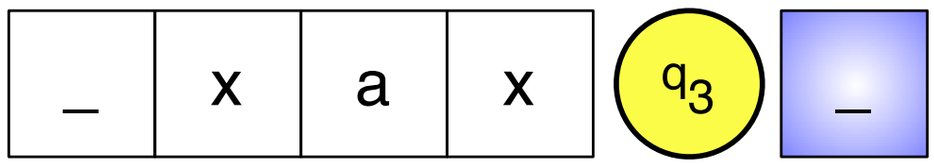
Eingabeband



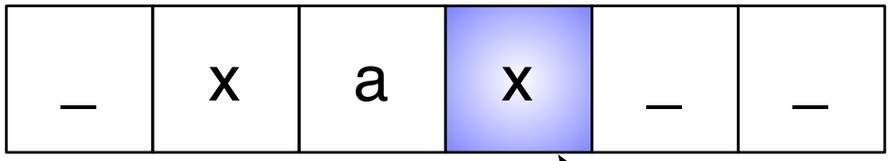
...



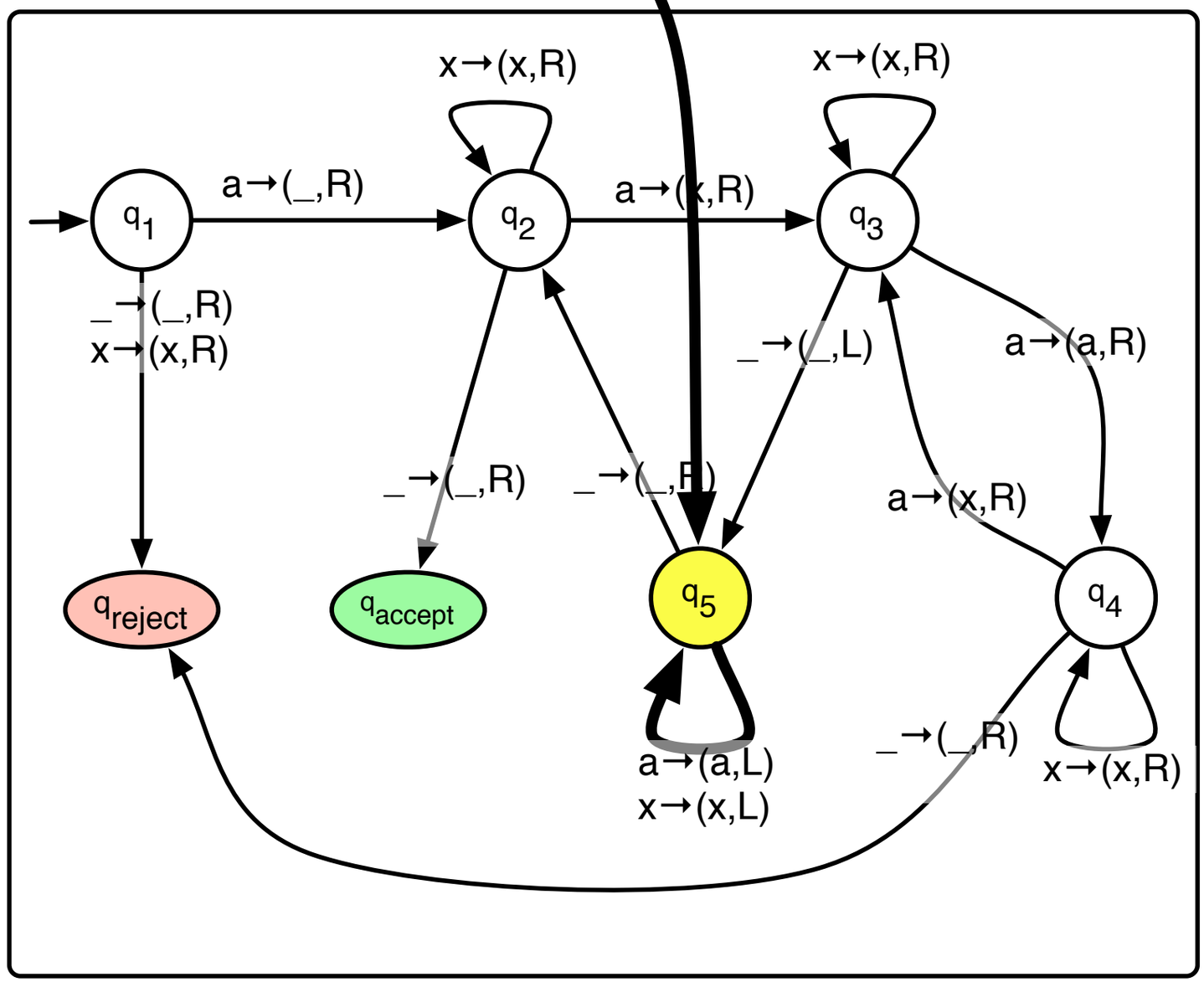
Konfiguration



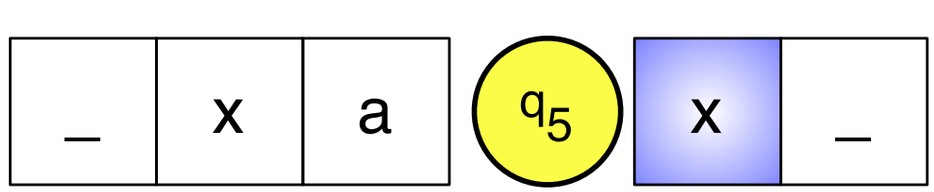
Eingabeband



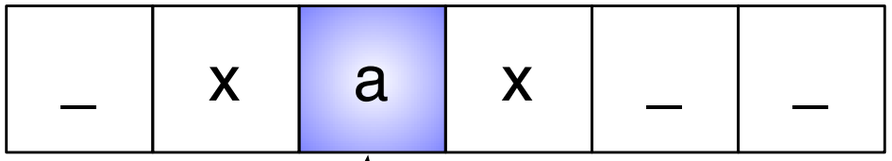
...



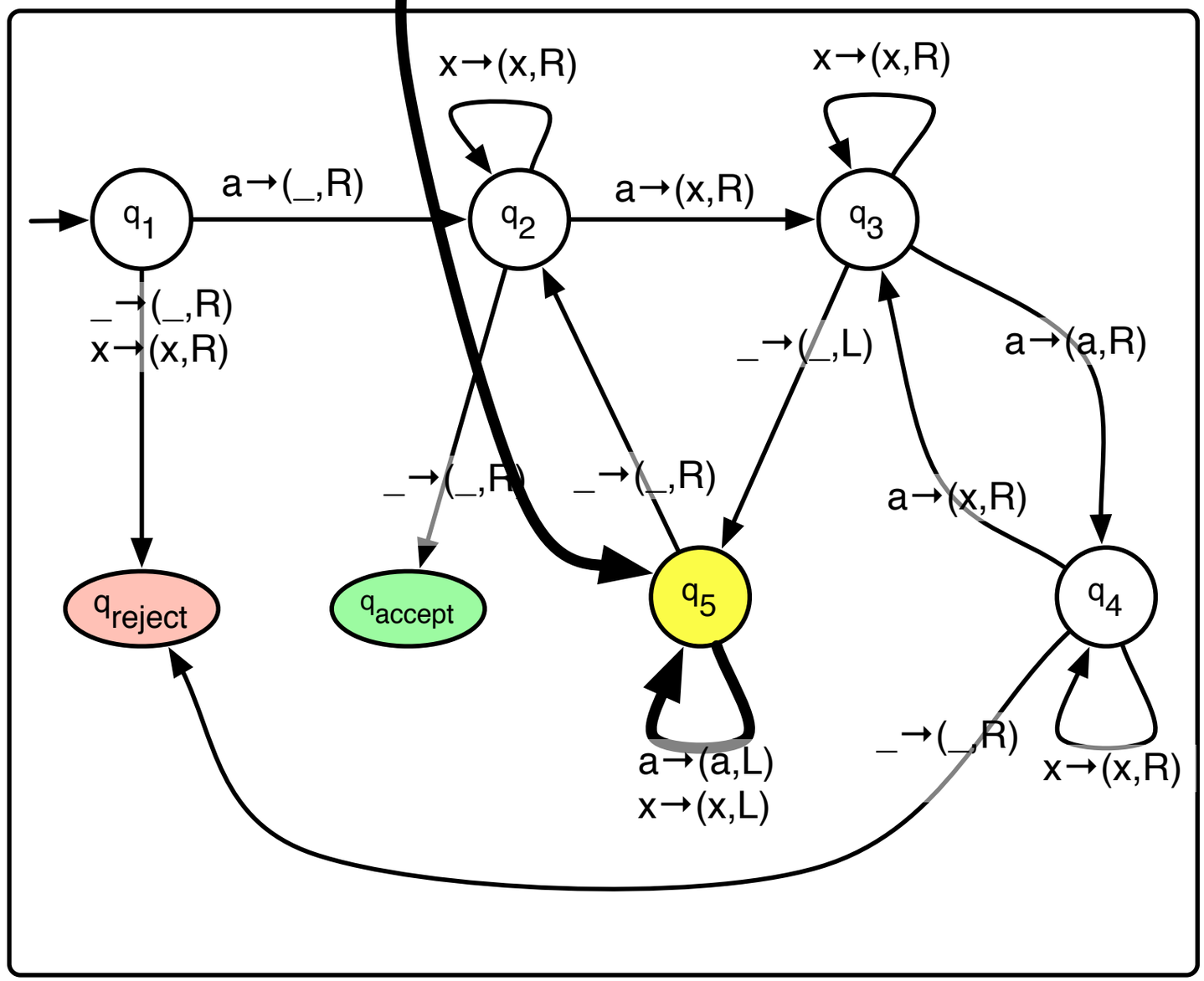
Konfiguration



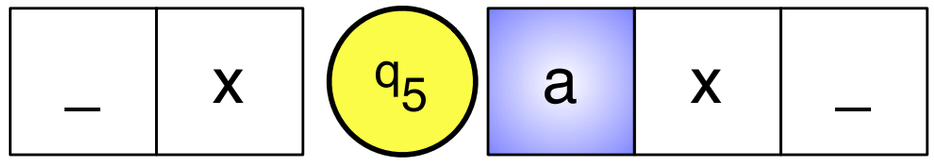
Eingabeband



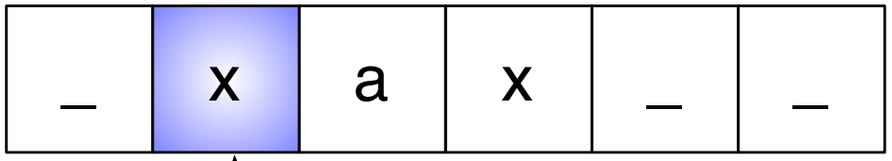
...



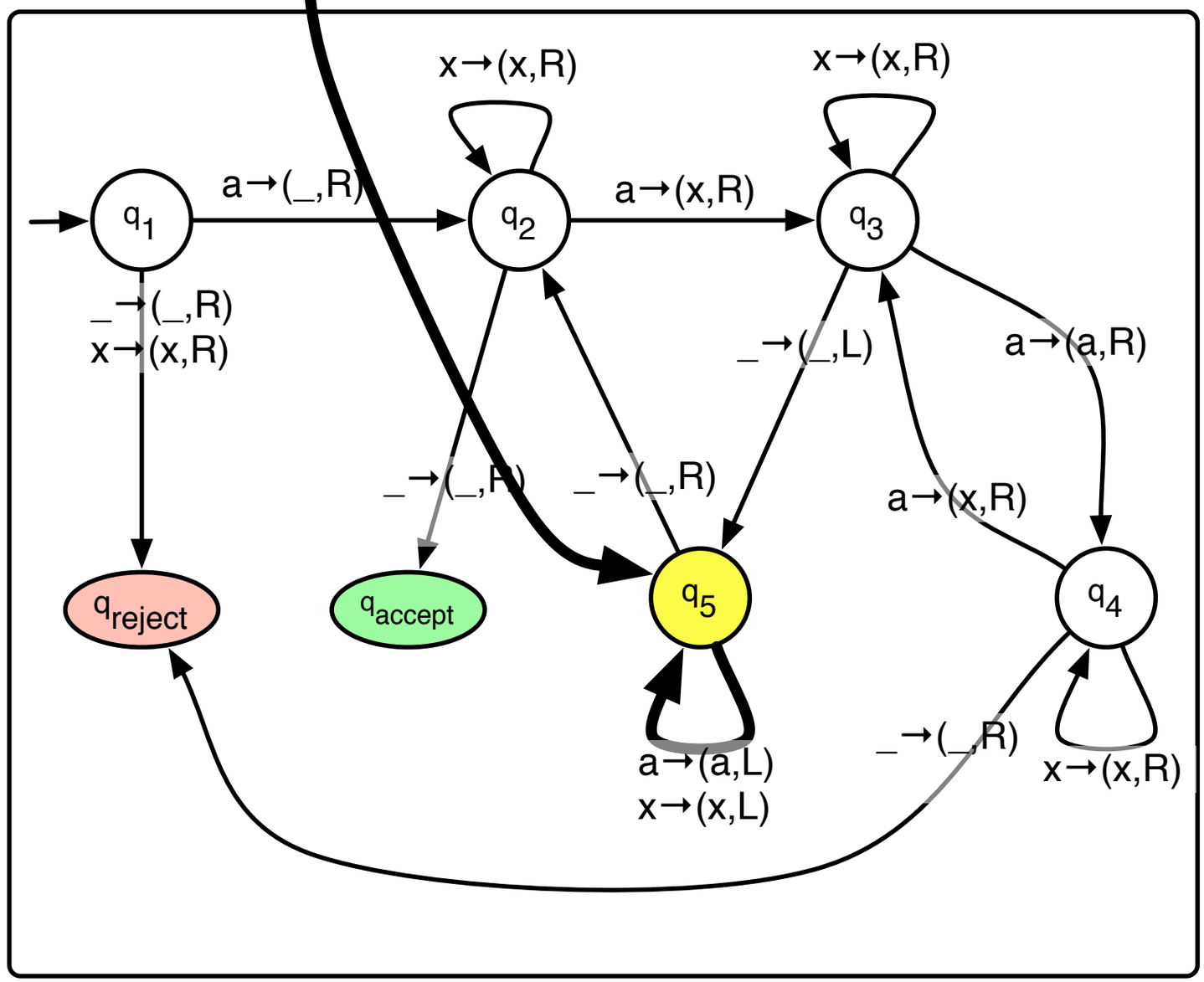
Konfiguration



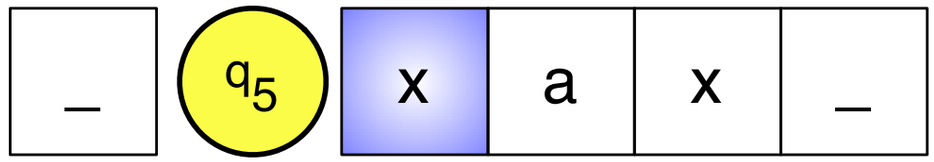
Eingabeband



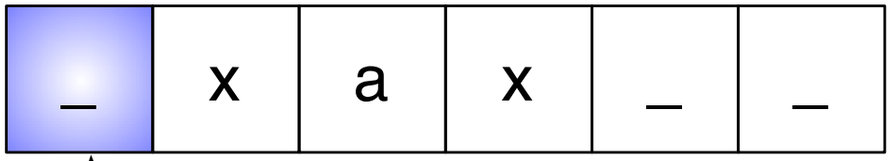
...



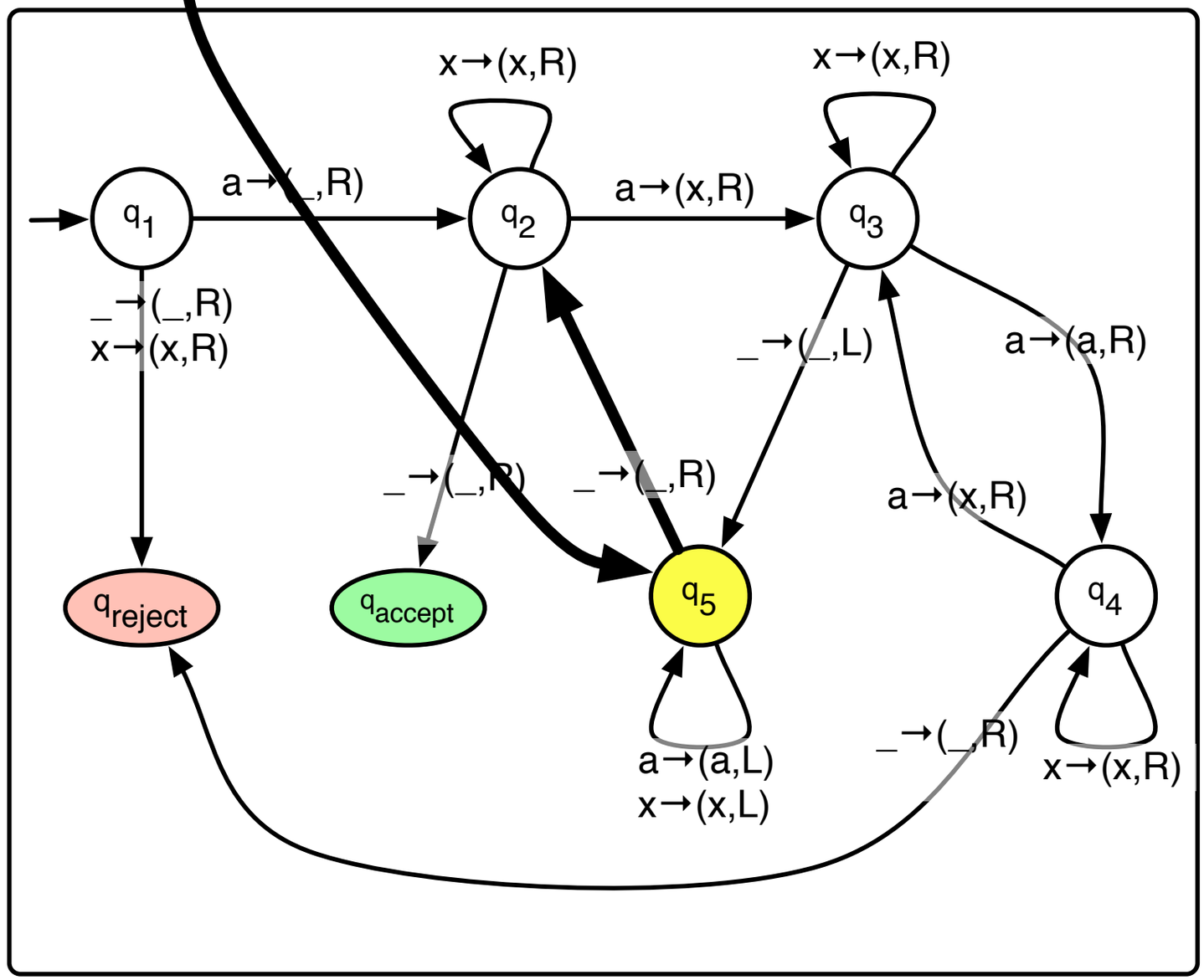
Konfiguration



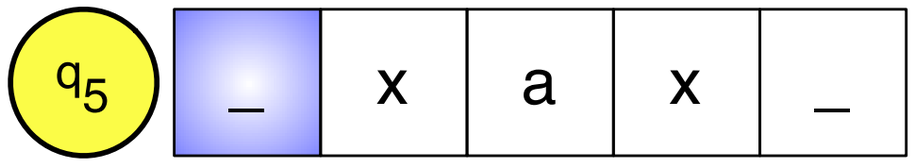
Eingabeband

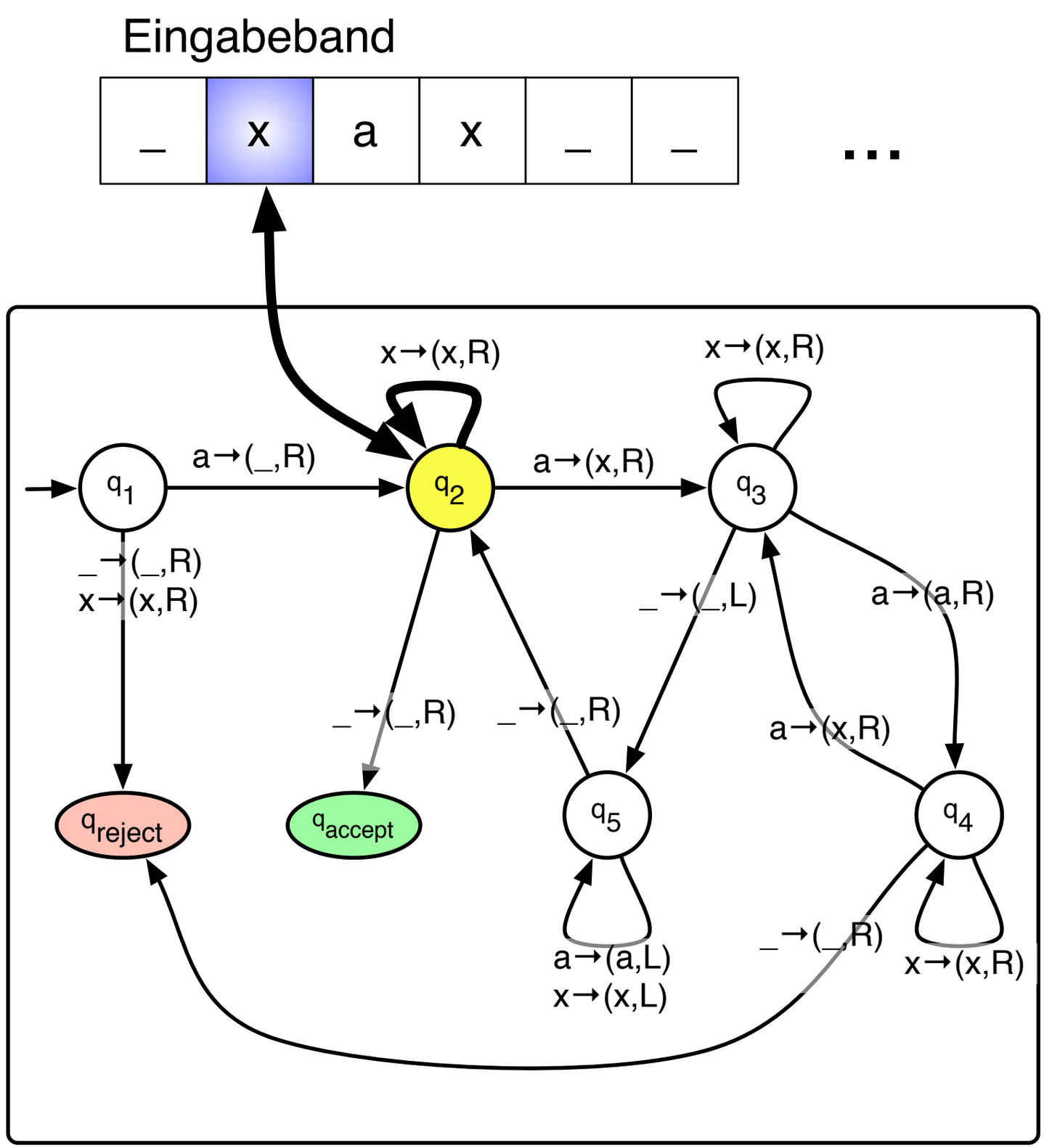


...

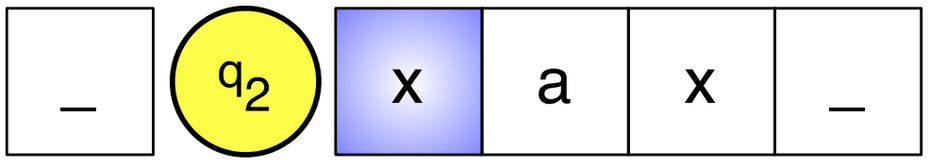


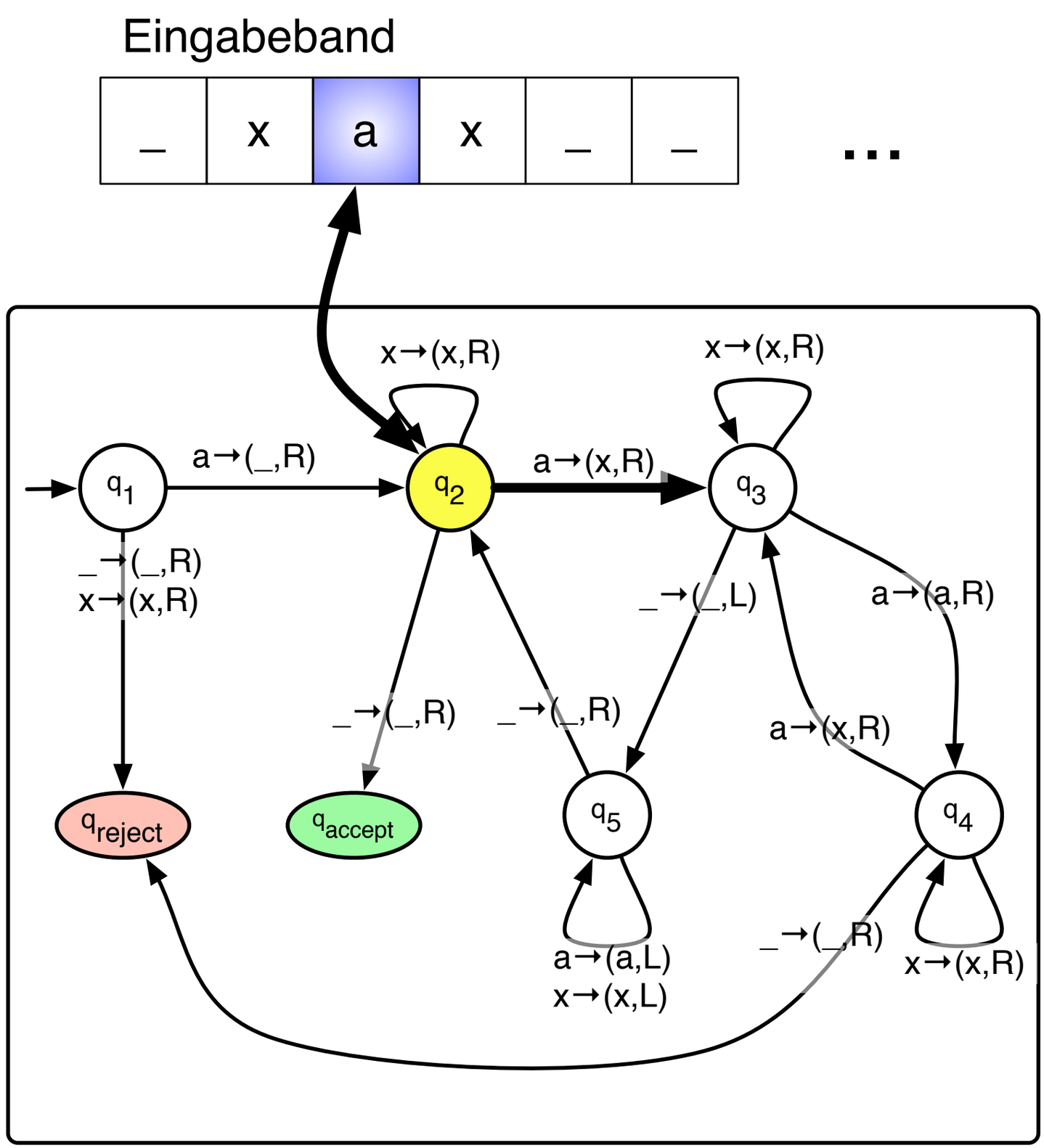
Konfiguration



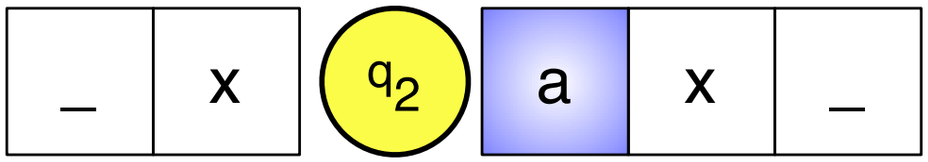


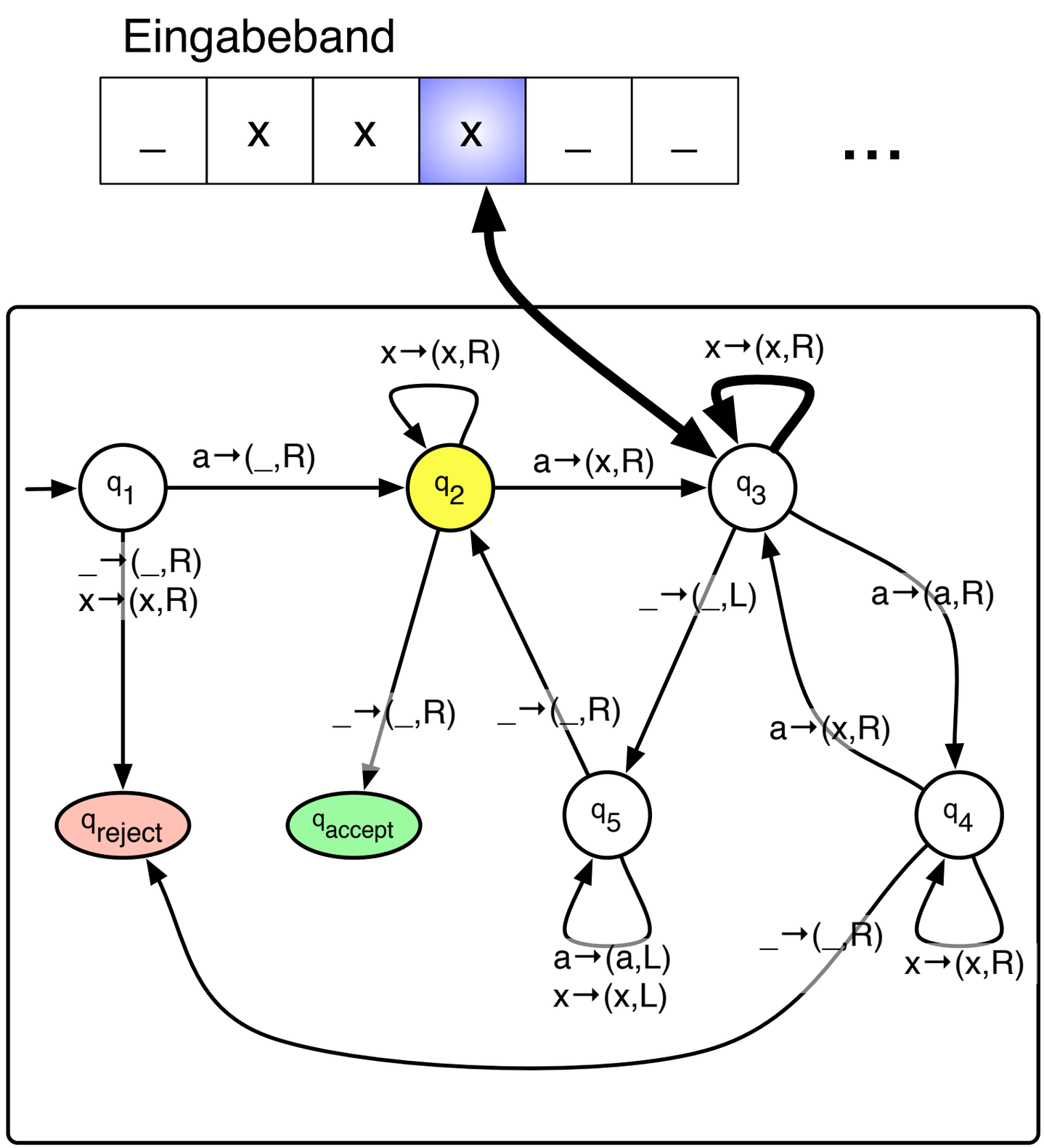
Konfiguration



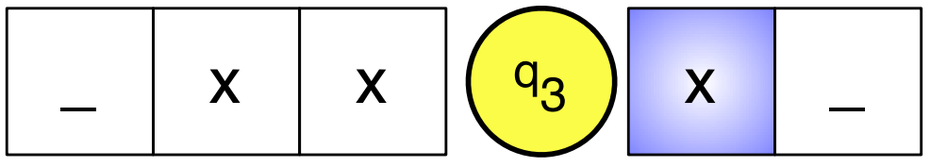


Konfiguration

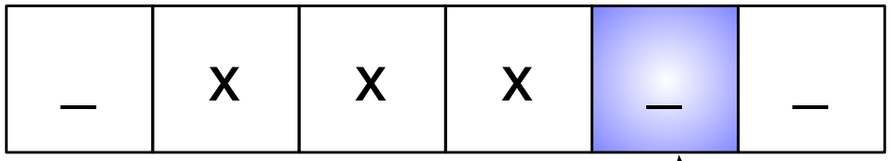




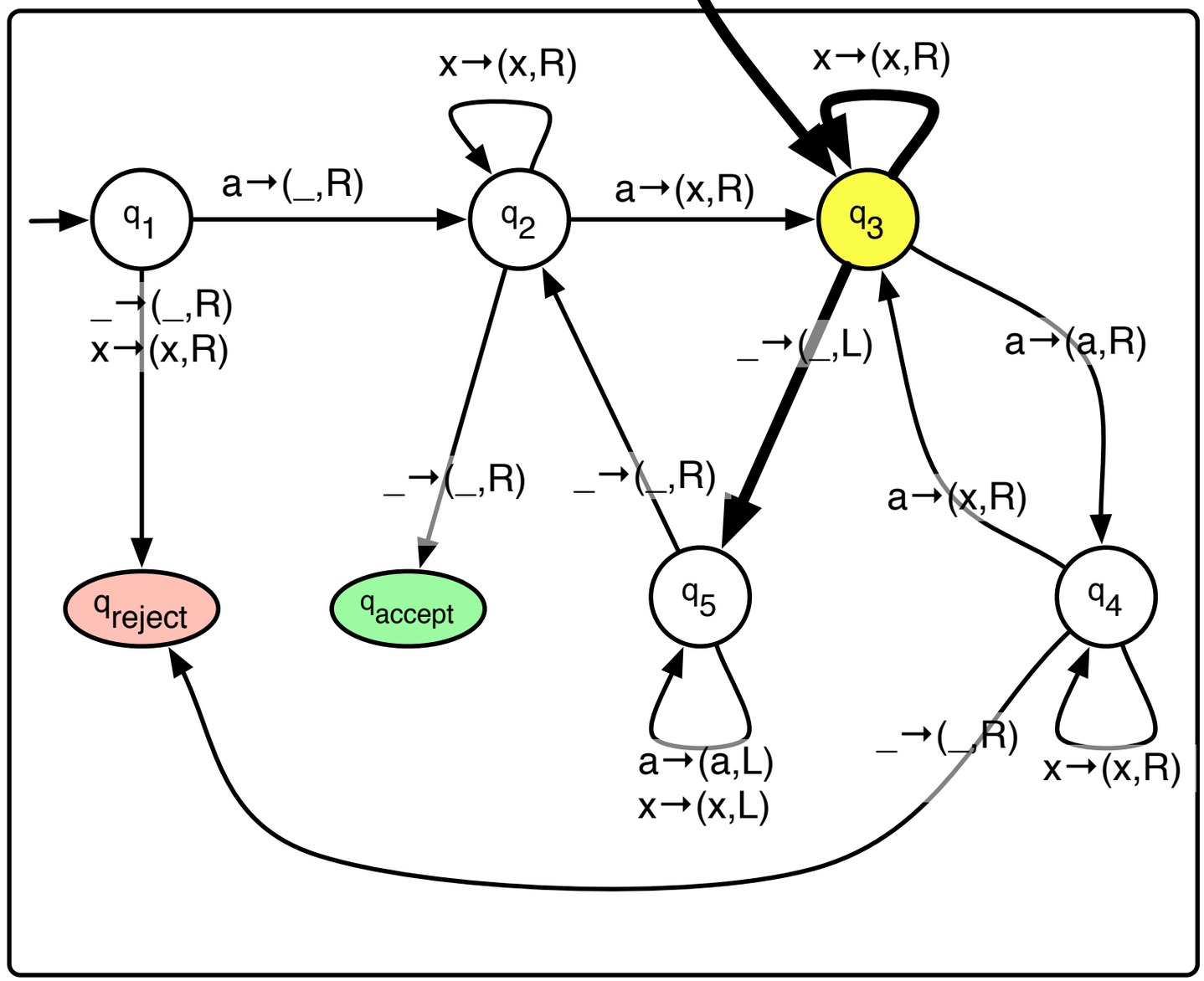
Konfiguration



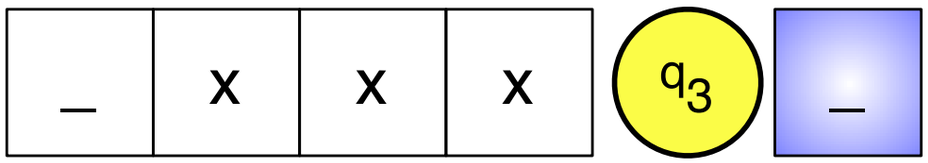
Eingabeband

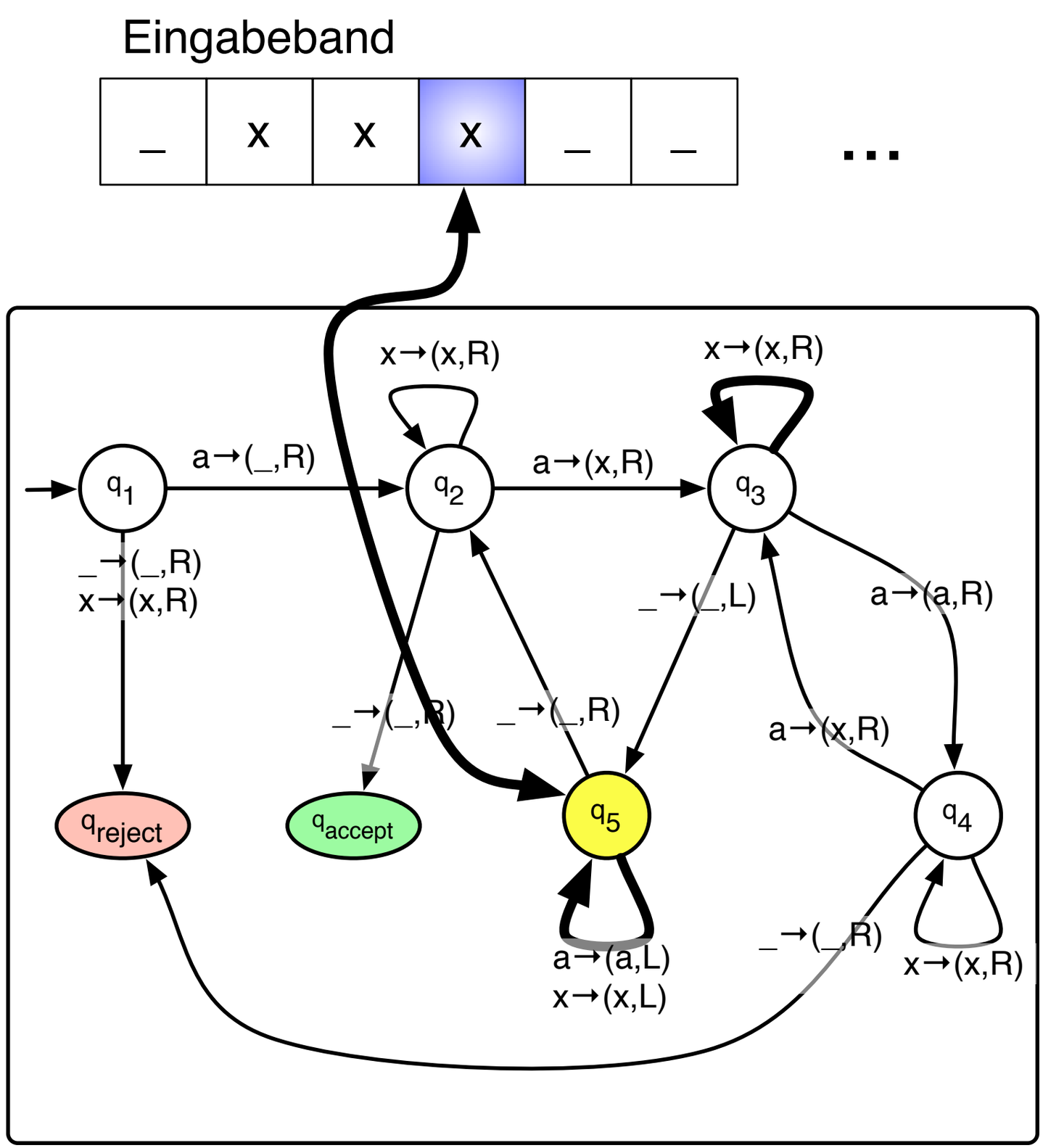


...

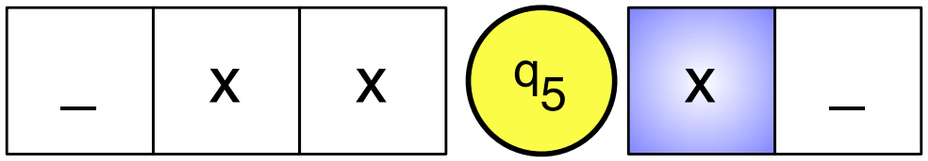


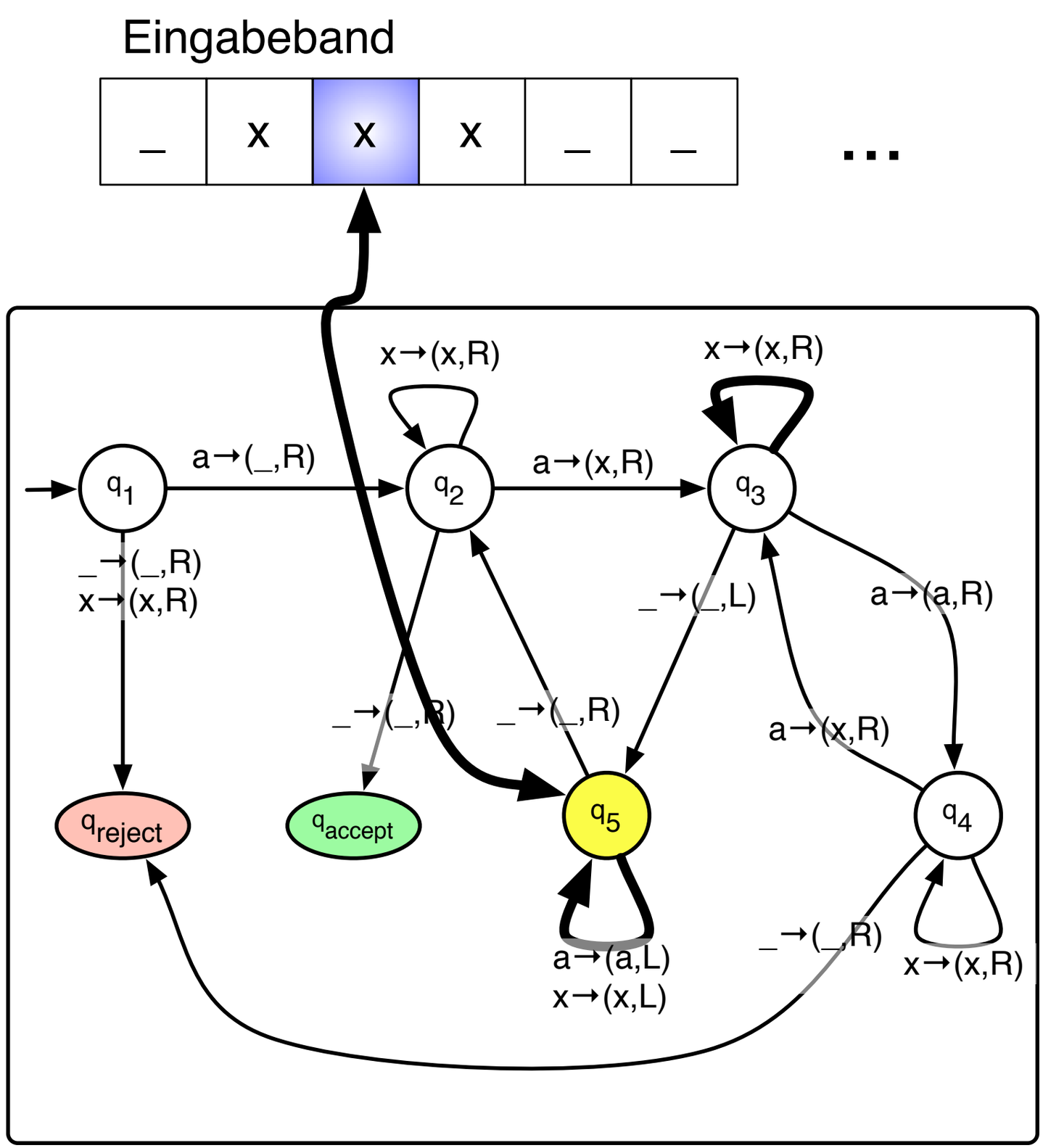
Konfiguration



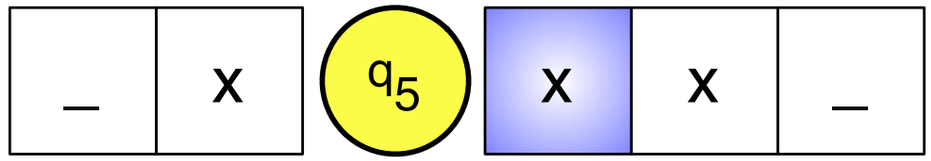


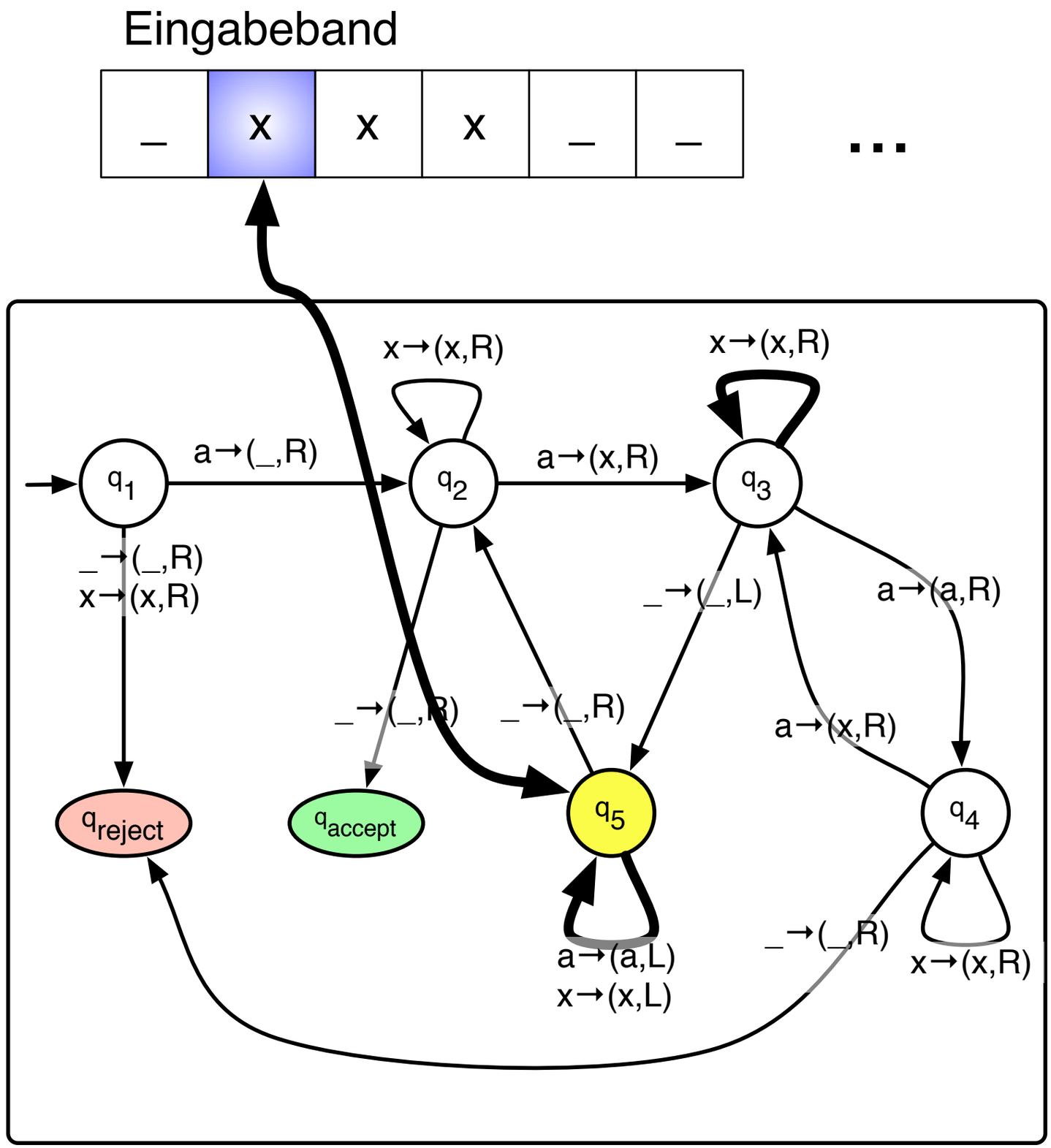
Konfiguration



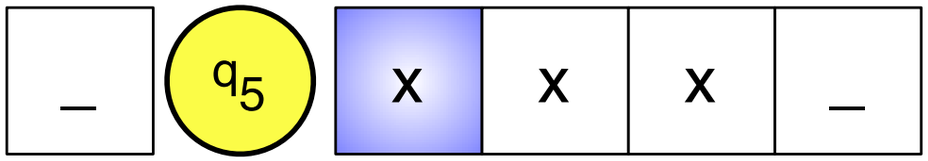


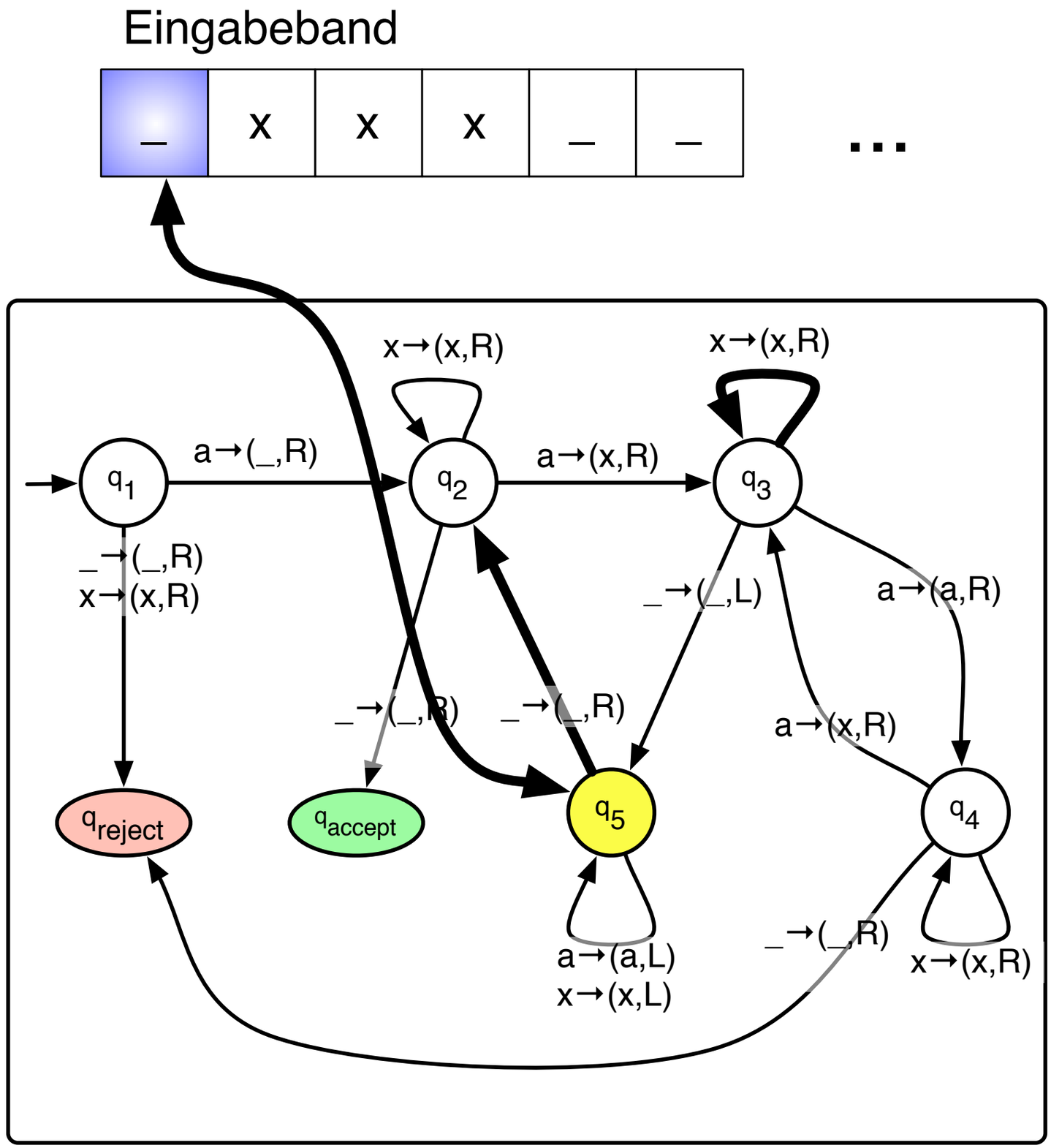
Konfiguration



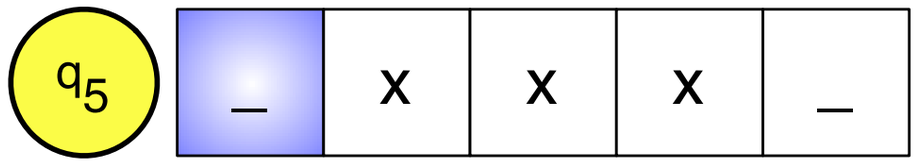


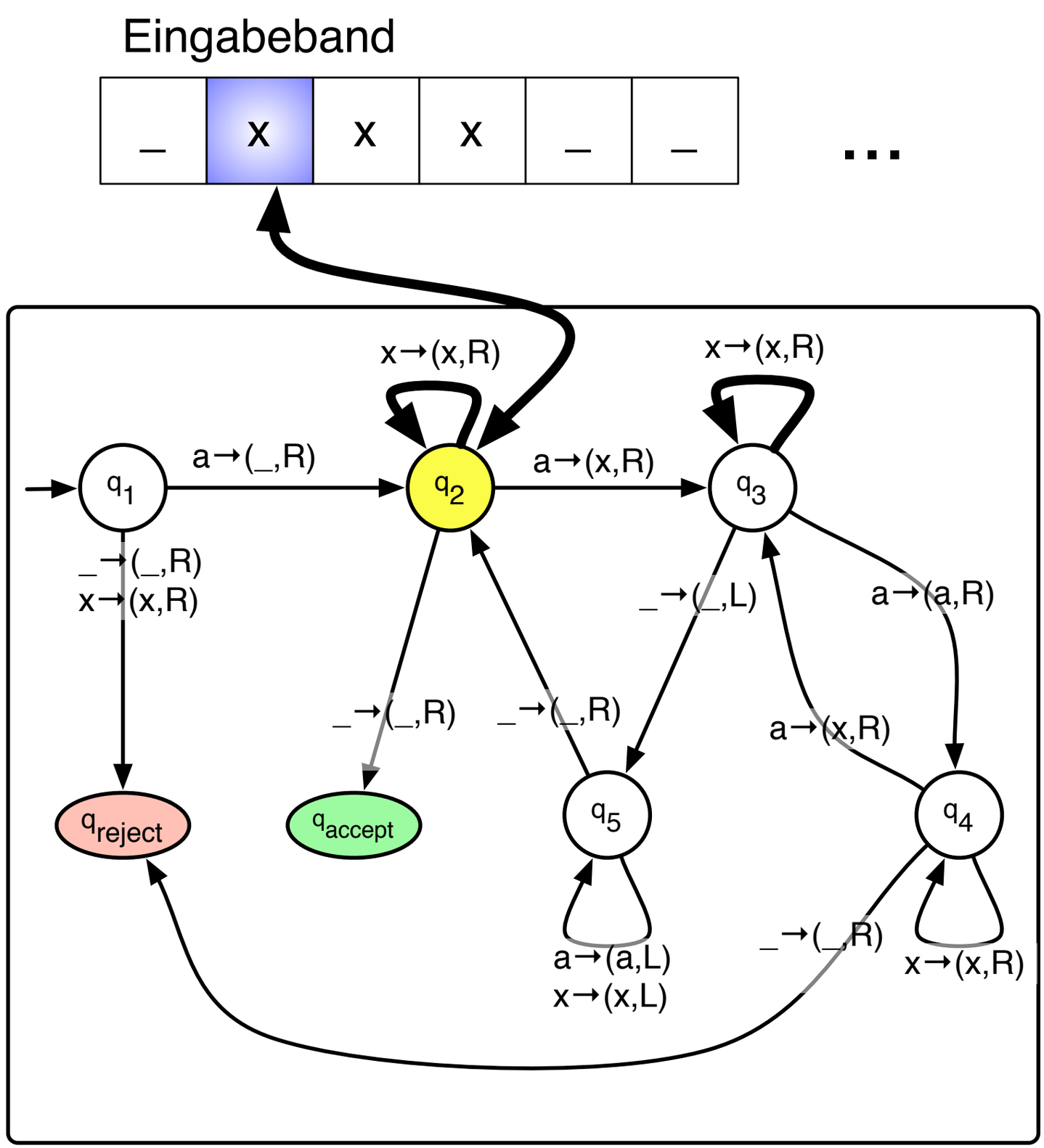
Konfiguration



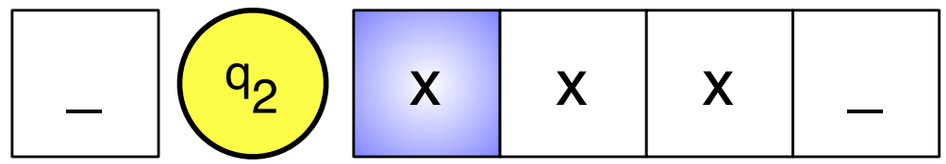


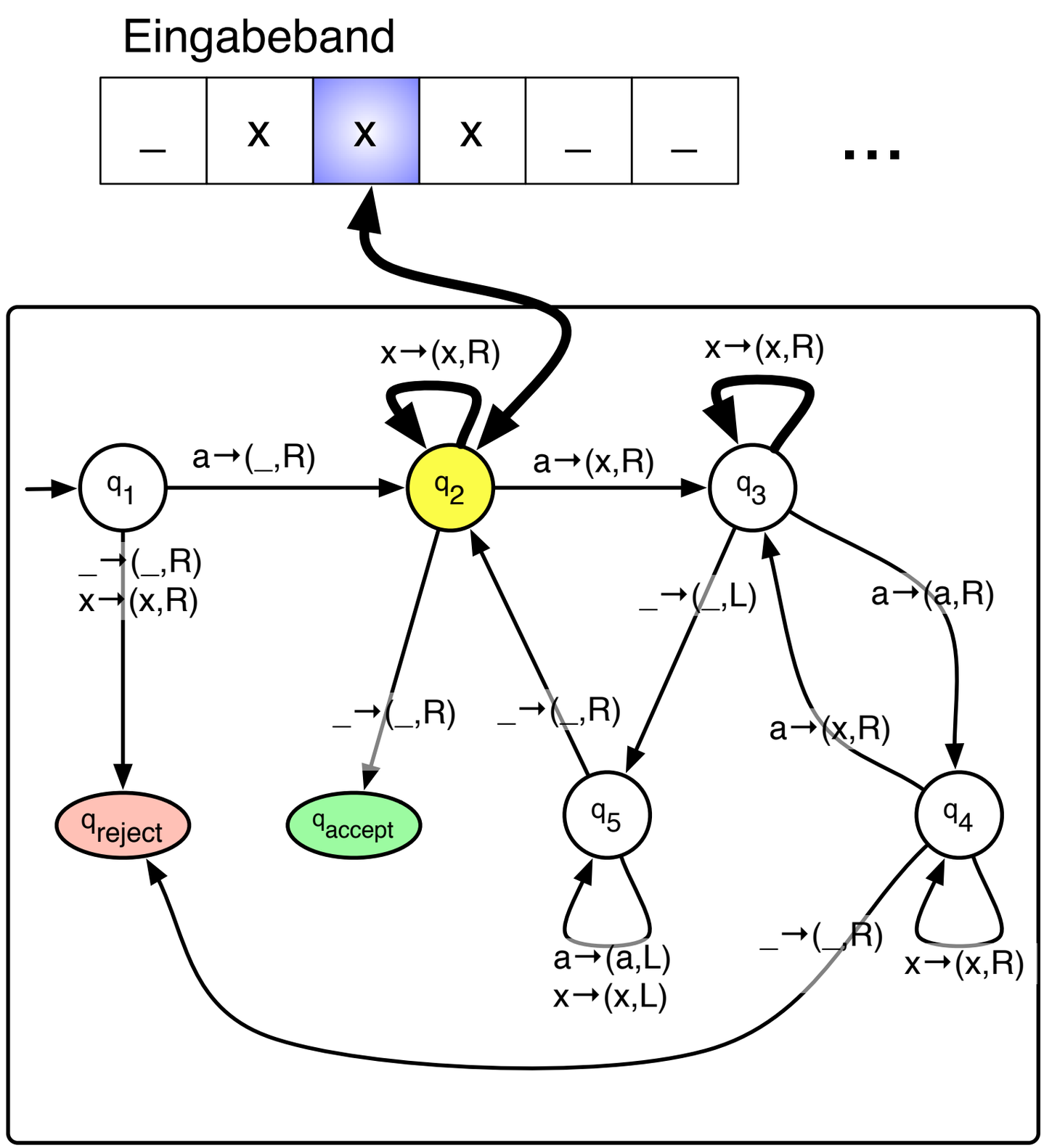
Konfiguration



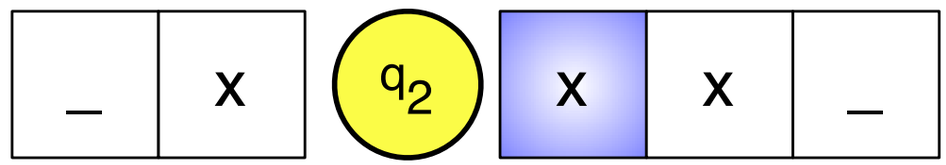


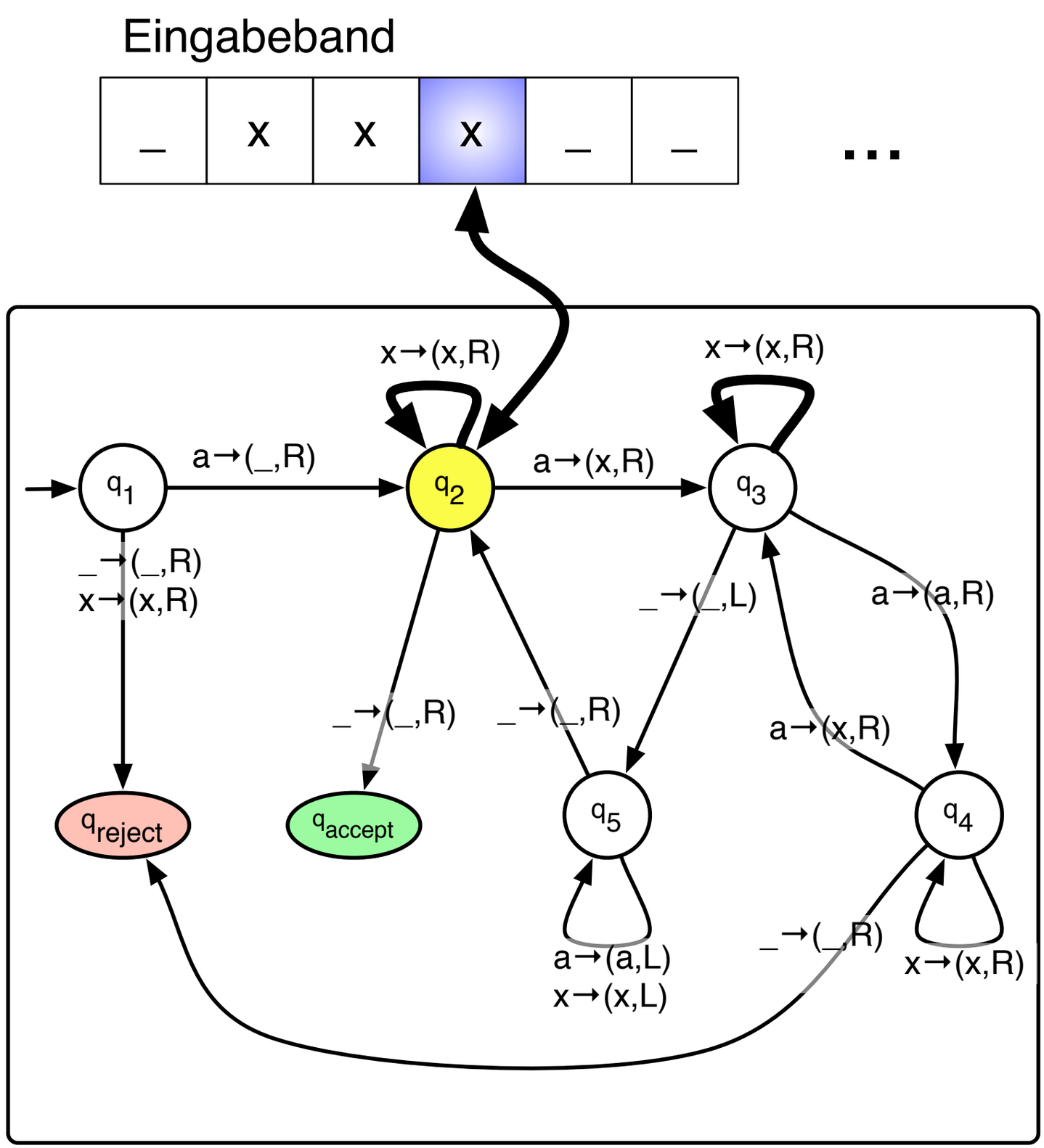
Konfiguration



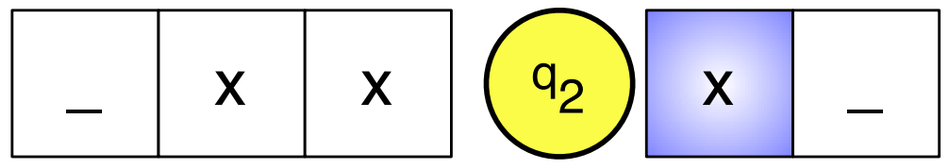


Konfiguration

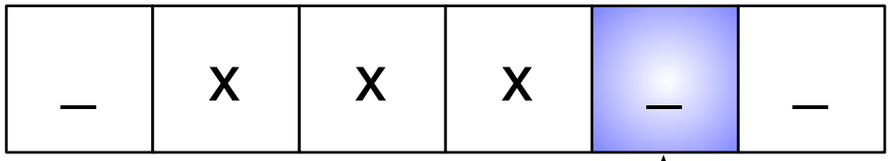




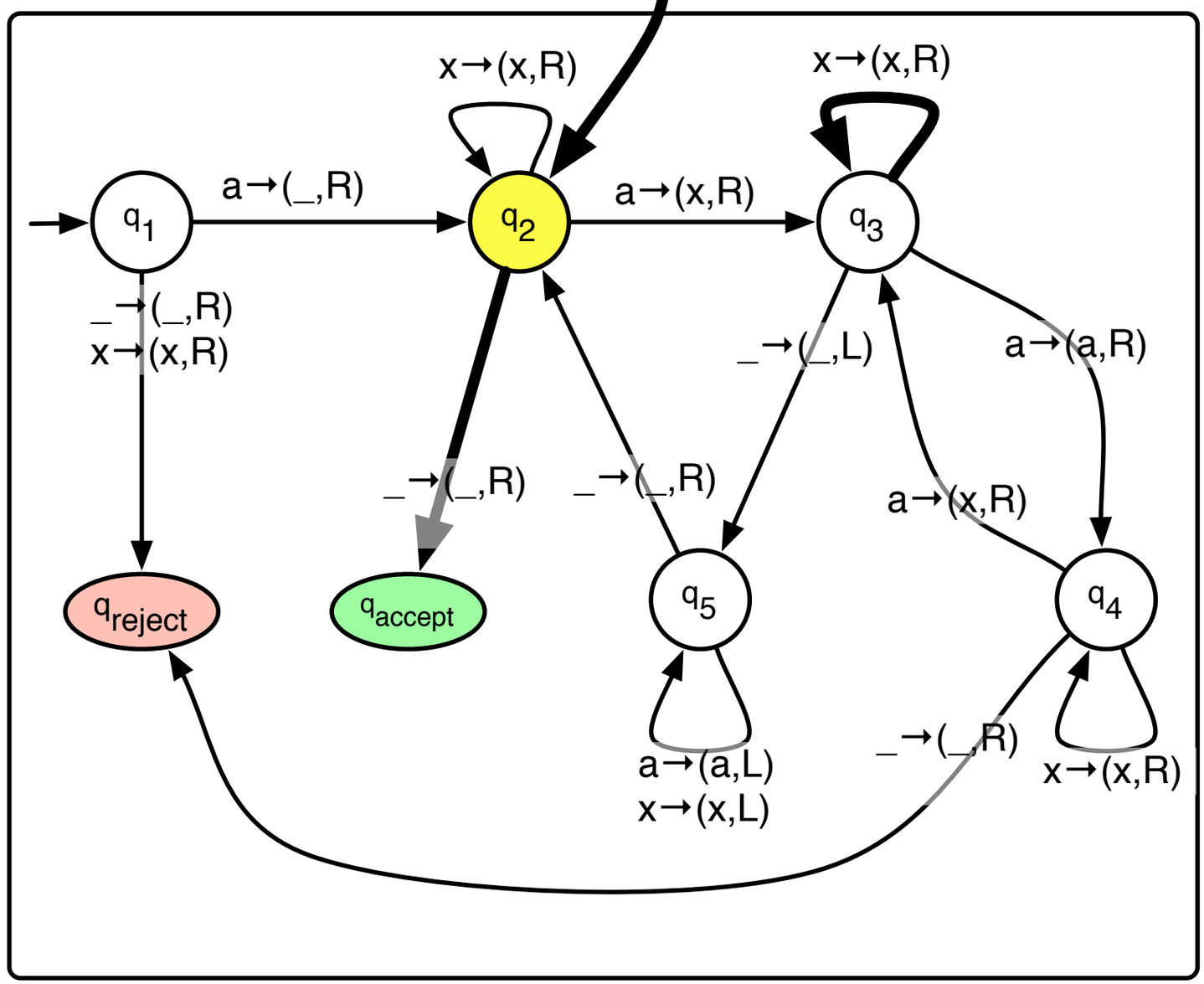
Konfiguration



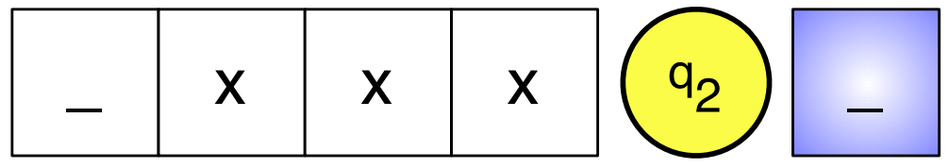
Eingabeband



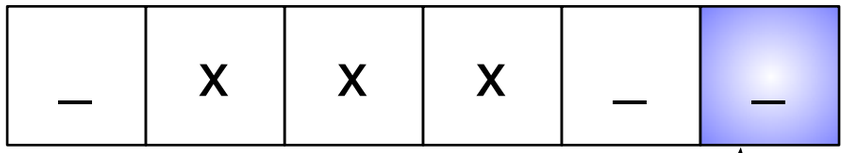
...



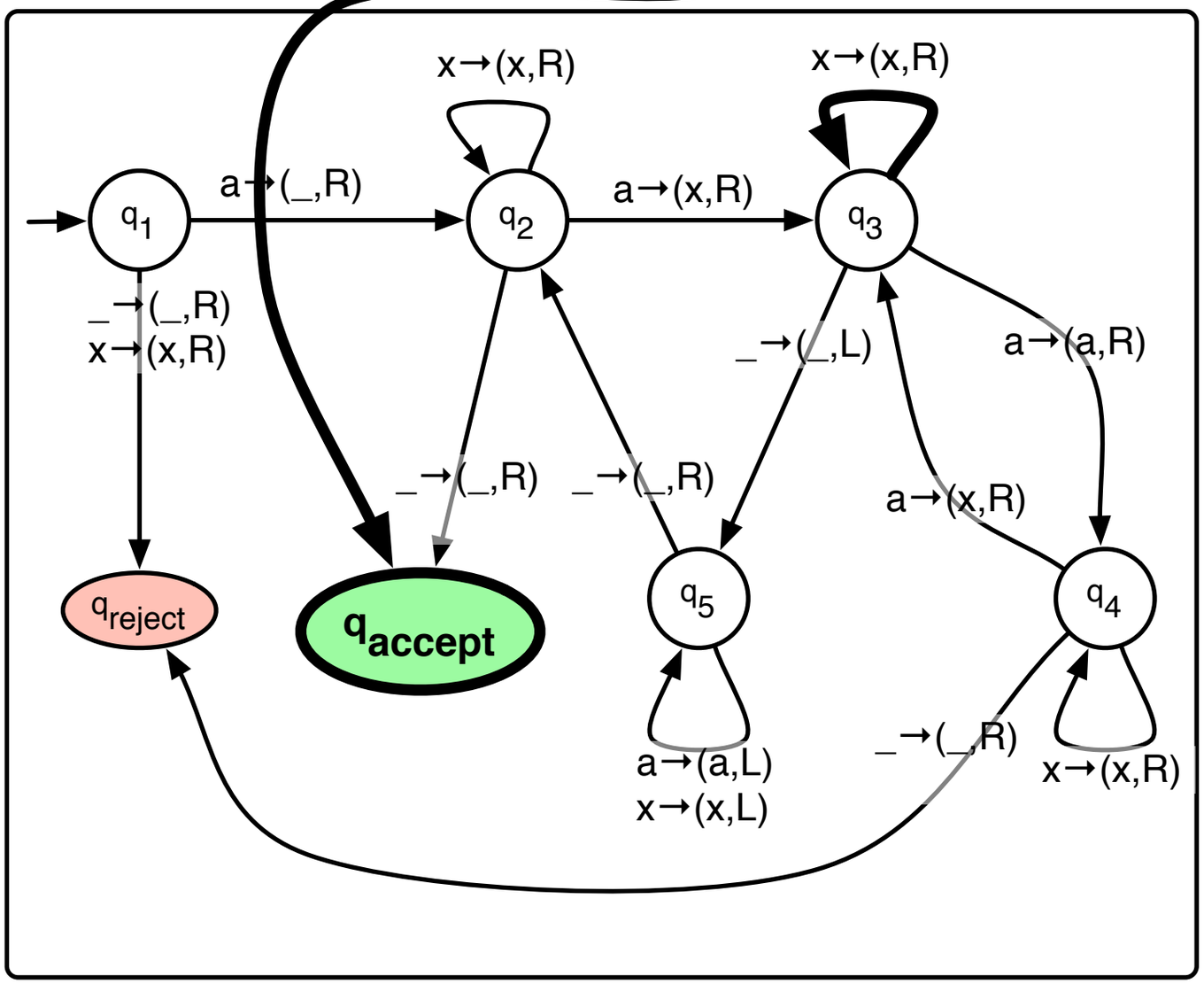
Konfiguration



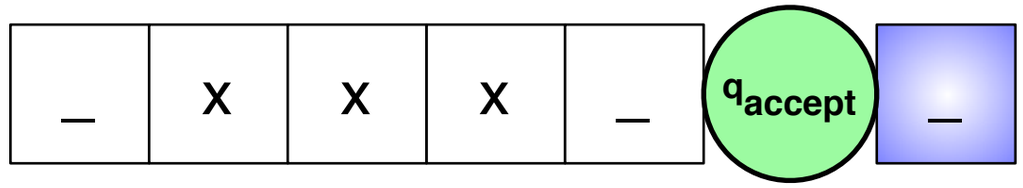
Eingabeband



...



akzeptierende Konfiguration

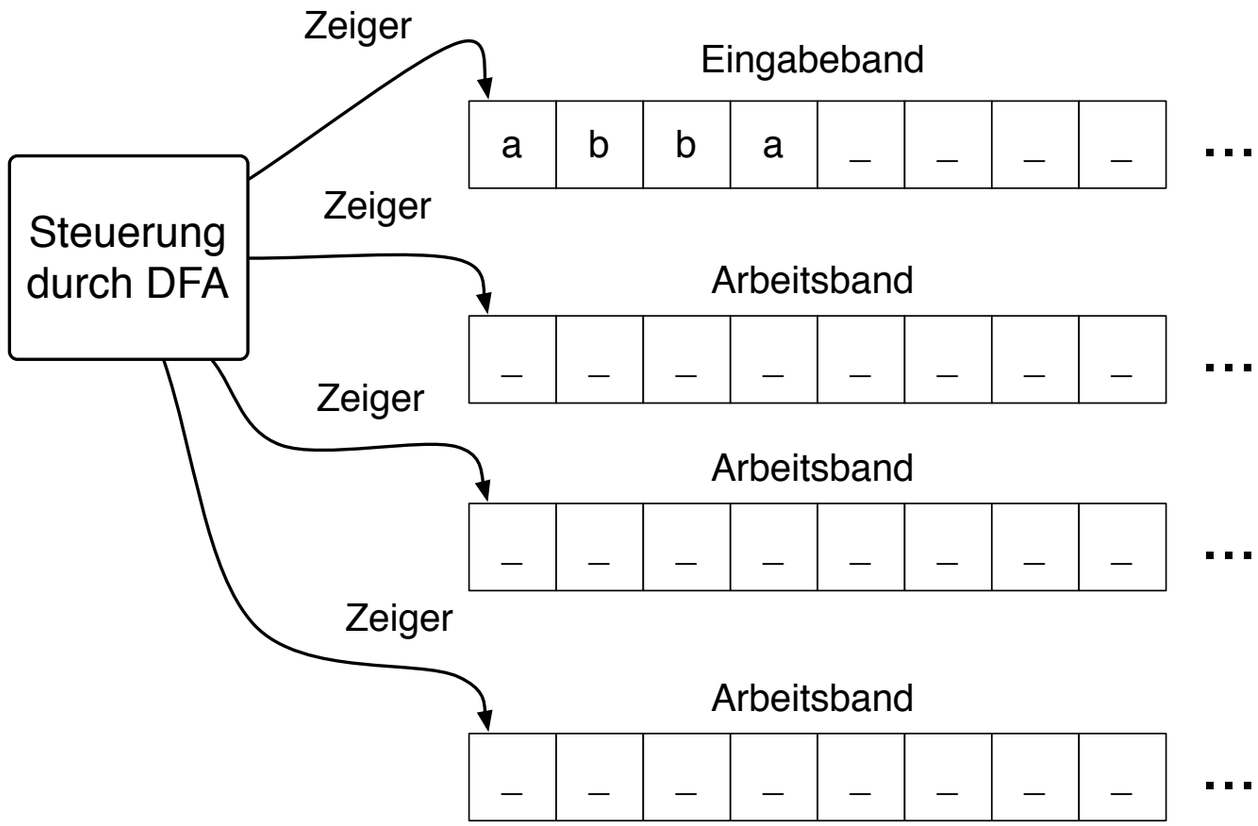


Berechenbarkeitstheorie

Mehrband-DTM

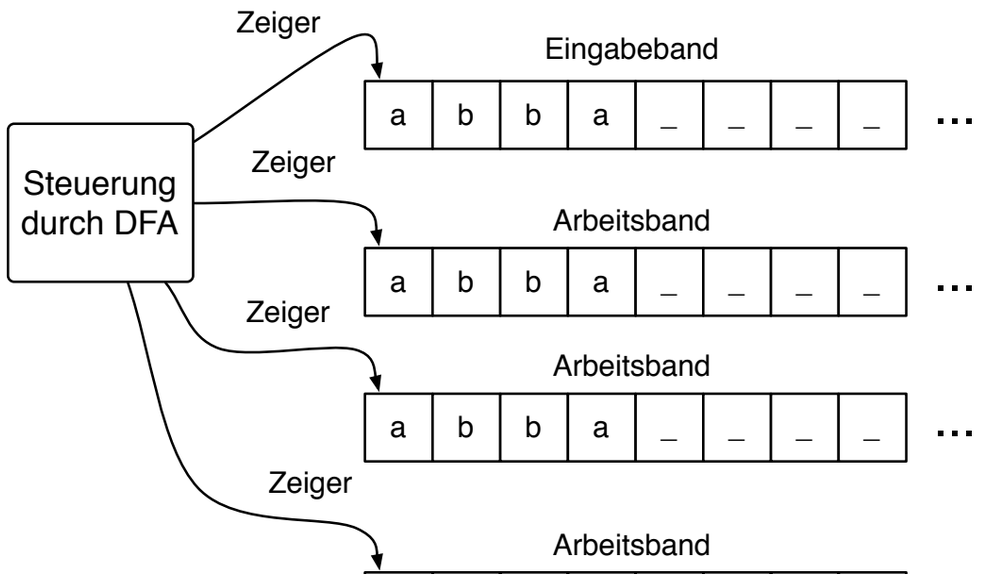
Mehrband-Turing-Maschinen

- **Eine Mehrband oder k-Band Turingmaschine (k-Band DTM)**
  - hat k Bänder mit je einem Kopf.
- **Die Übergangsfunktion ist dann von der Form**
  - $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$
- **Die Arbeitsweise ist analog zu 1-Band-DTMs definiert.**
  - Zu Beginn steht die Eingabe auf dem ersen Band,
  - sonst stehen überall Blanks.

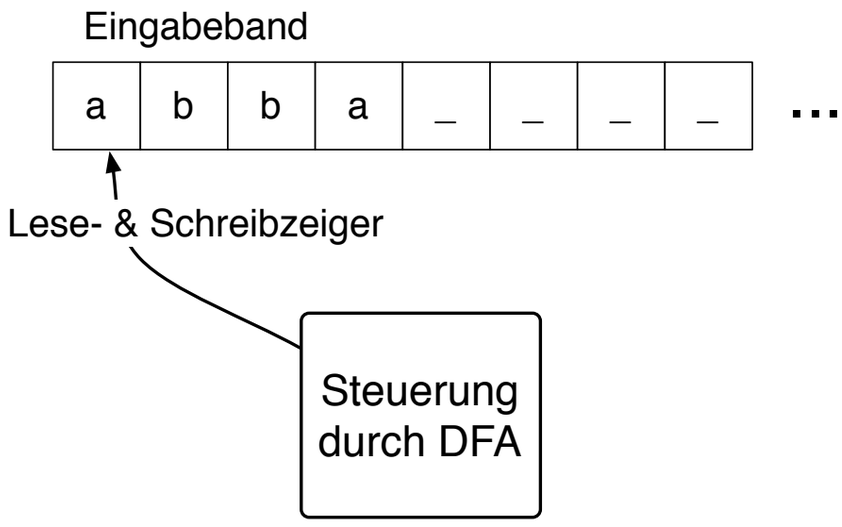


Äquivalenz von 1-Band und Mehrband-Turing-Maschinen

- **Satz:**
  - Zu jeder Mehrband Turingmaschine gibt es eine äquivalente 1-Band Turingmaschine

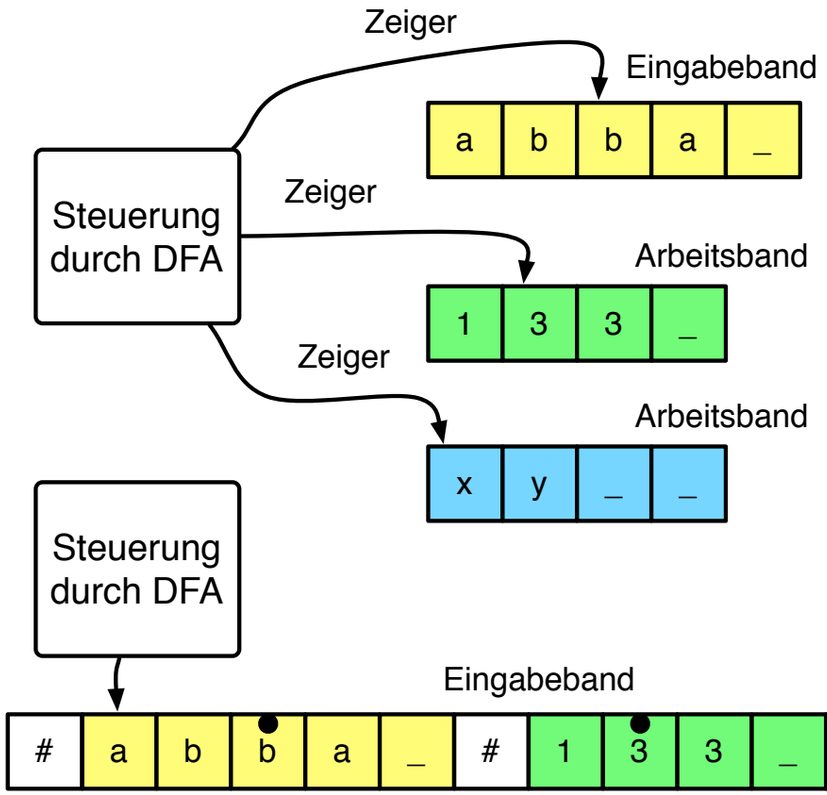


- **Beweis:**
  - Idee: Simuliere Mehrband DTM M auf 1-Band DTM S



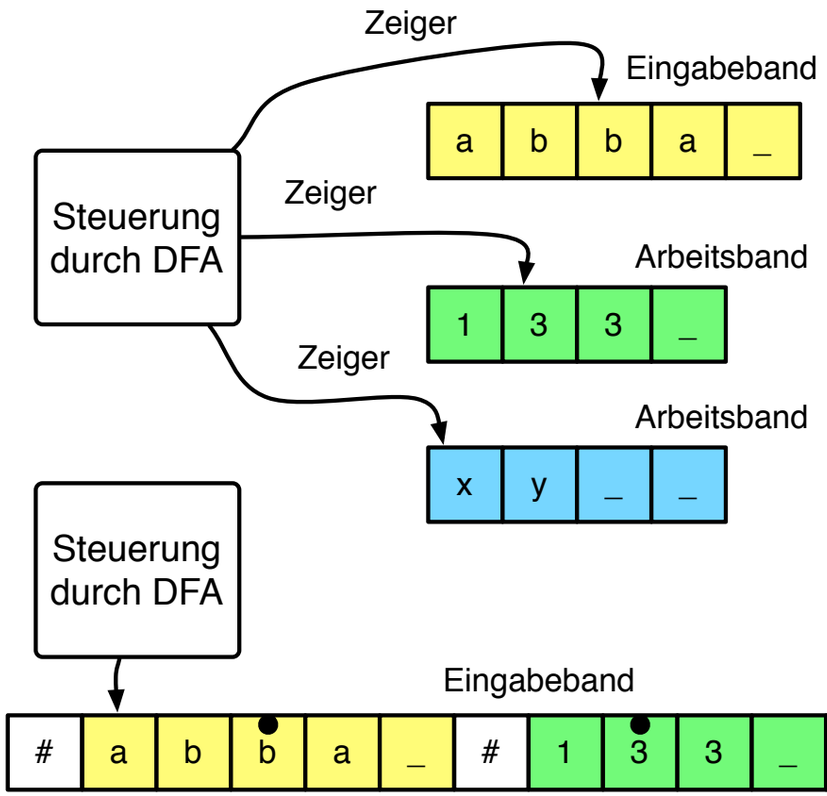
Fortsetzung des Beweises

- Schreibe k Bänder hintereinander auf ein Band
- Sei # ∉ Γ zusätzliches Symbol
  - Verwende # um Bänder zu trennen
- Für $x \in \Gamma$ füge $\dot{x} \in \Gamma$ zum Alphabet hinzu
- Der Punkt repräsentiert die aktuelle Position des Kopfes auf dem Band



Beweis Fortsetzung

- **Bei Eingabe w = w_1, … , w_n:**
  1. S bildet Startkonfiguration von M ab:
     $$\# \dot{w}_1 w_2 \ldots w_n \# \_ \# \_ \# \ldots \#$$
  2. Simulation eines Schrittes:
     - Gehe von links nach rechts über das Band und suche die Markierungen # und finde die Zeichen unter den virtuellen Köpfen
     - Gehe von links nach rechts über das Band und führe die Änderungen entsprechend der Übergangsfunktion von M durch
  3. Falls ein virtueller Kopf rechts auf ein # bewegt wird, schreibe ein _ und bewege alle folgenden Zeichen eine Position nach rechts



Äquivalenz von 1-Band und Mehrband Turingmaschinen

- **Korollar:**
  - Eine Sprache L ist genau dann **rekursiv aufzählbar**, wenn es eine **Mehrband-Turingmaschine** gibt, die L akzeptiert
  - Eine Sprache L ist genau dann **rekursiv**, wenn es eine **Mehrband-Turingmaschine** gibt, die L entscheidet.

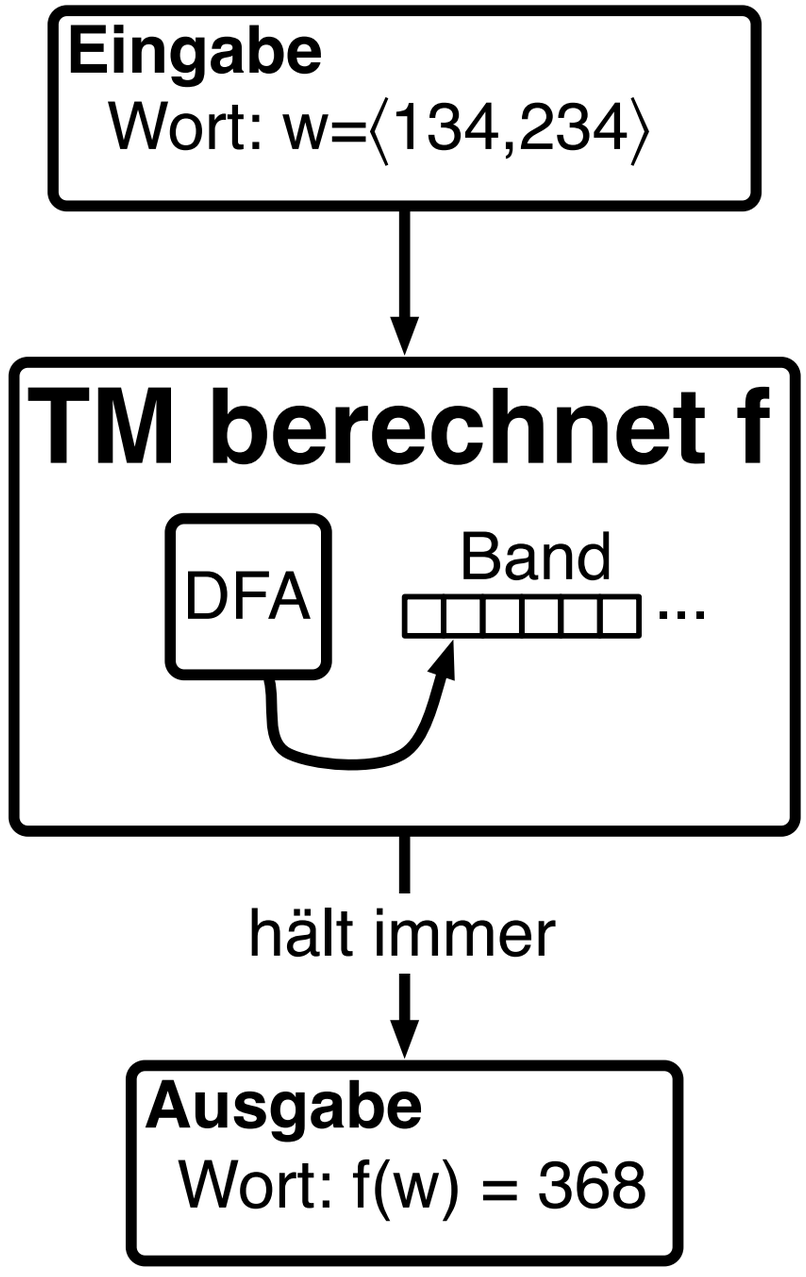
Berechenbarkeitstheorie

Berechenbar und Aufzählbar

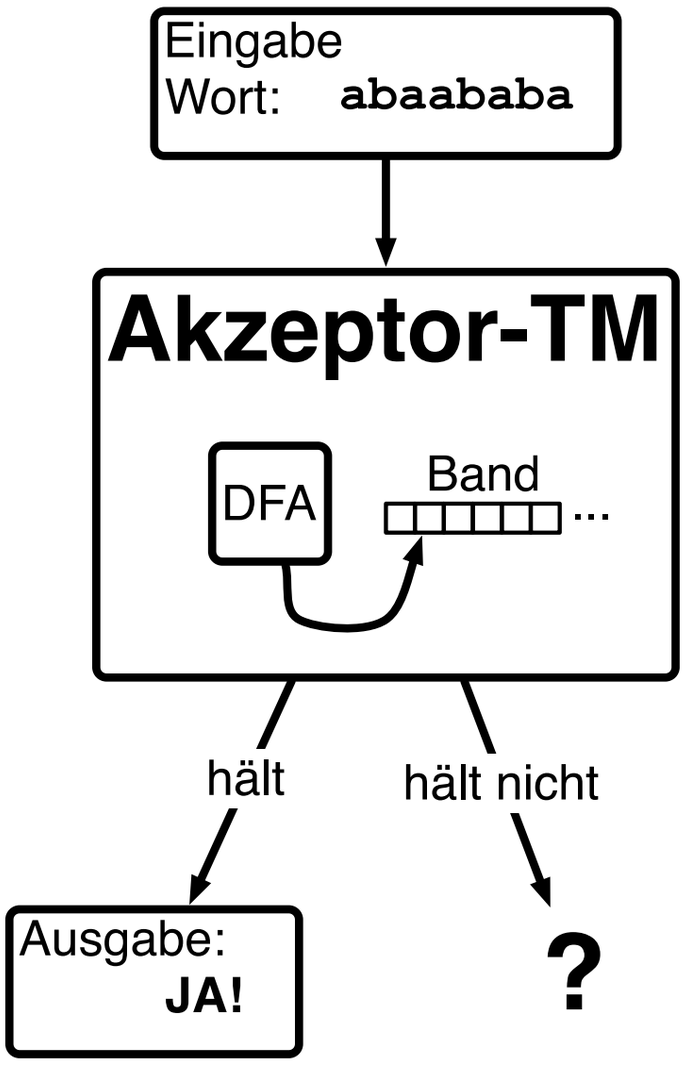
65

Berechenbare Funktionen

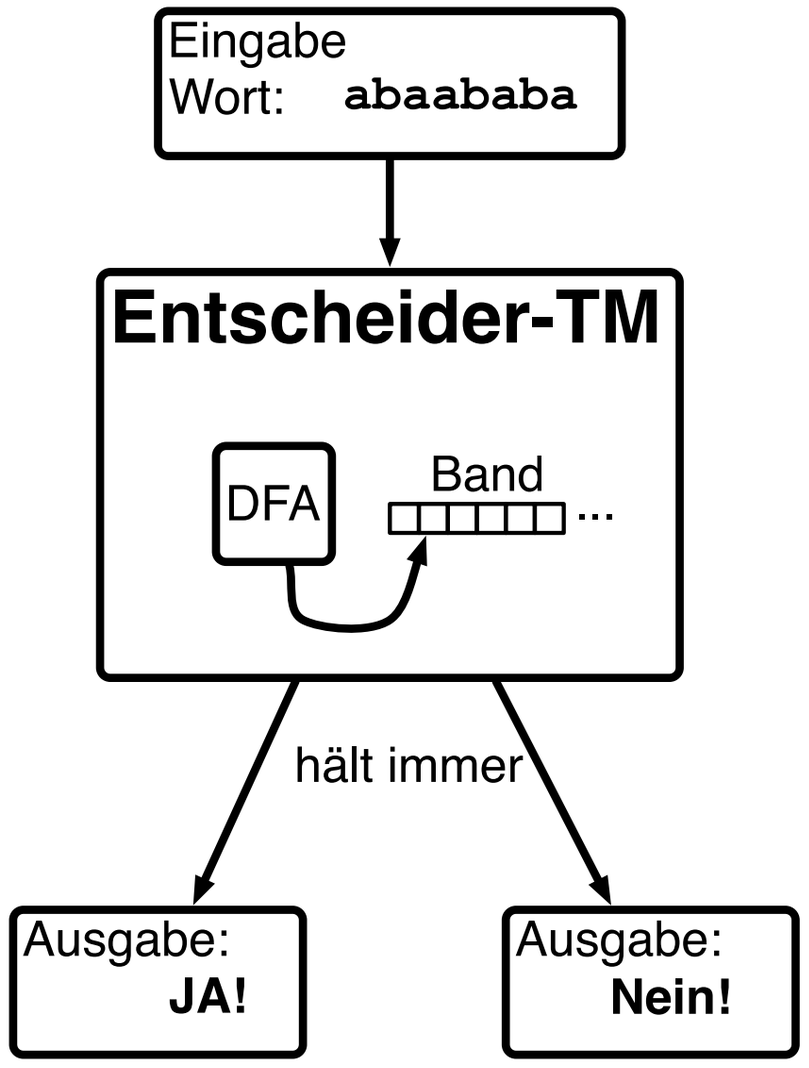
- **Definition**
  - Eine Funktion f: $\Sigma^* \rightarrow \Sigma^*$ ist berechenbar, falls eine Turing-Maschine für jede Eingabe w mit dem Ergebnis f(w) auf dem Band hält.
- **Beispiele für berechenbare Funktionen:**
  - Addition, Division, Multiplikation, Vergleich, Sortieren, Division, ...



Rekursive und rekursiv aufzählbare Sprachen



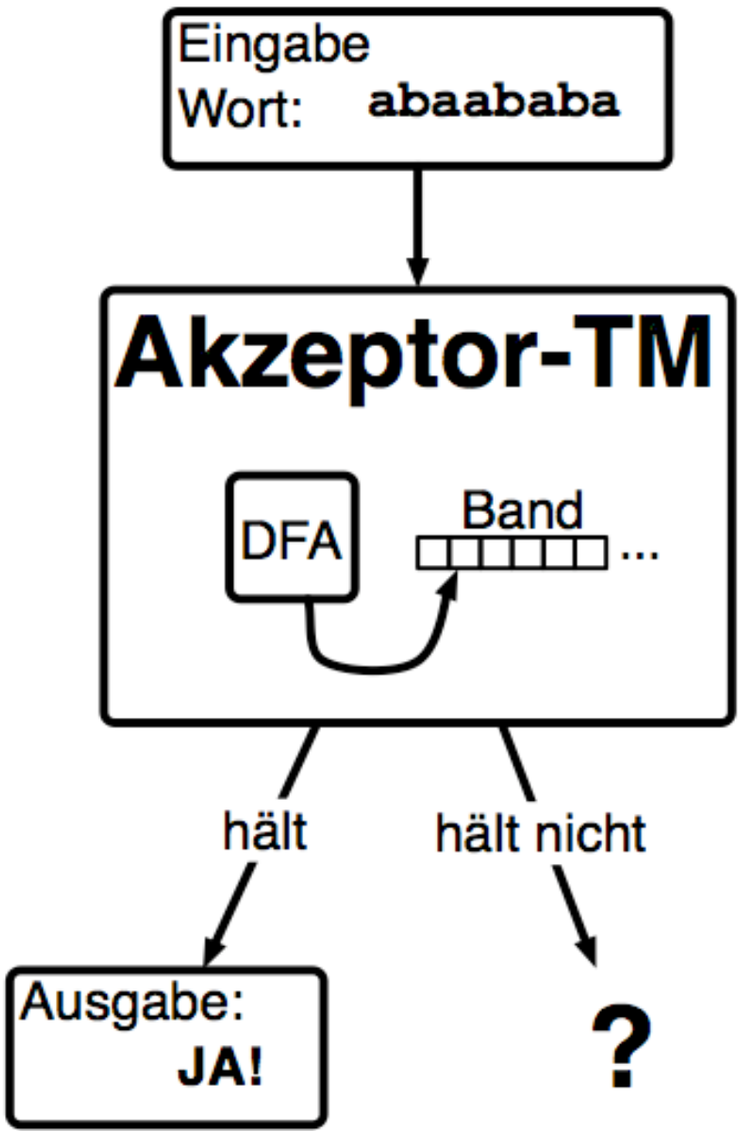
- **Eine Sprache L heißt *rekursiv aufzählbar*, falls es eine Turingmaschine M gibt, die L akzeptiert**
- **Eine Sprache L heißt *rekursiv* oder *entscheidbar*, falls es eine Turingmaschine M gibt, die L entscheidet**

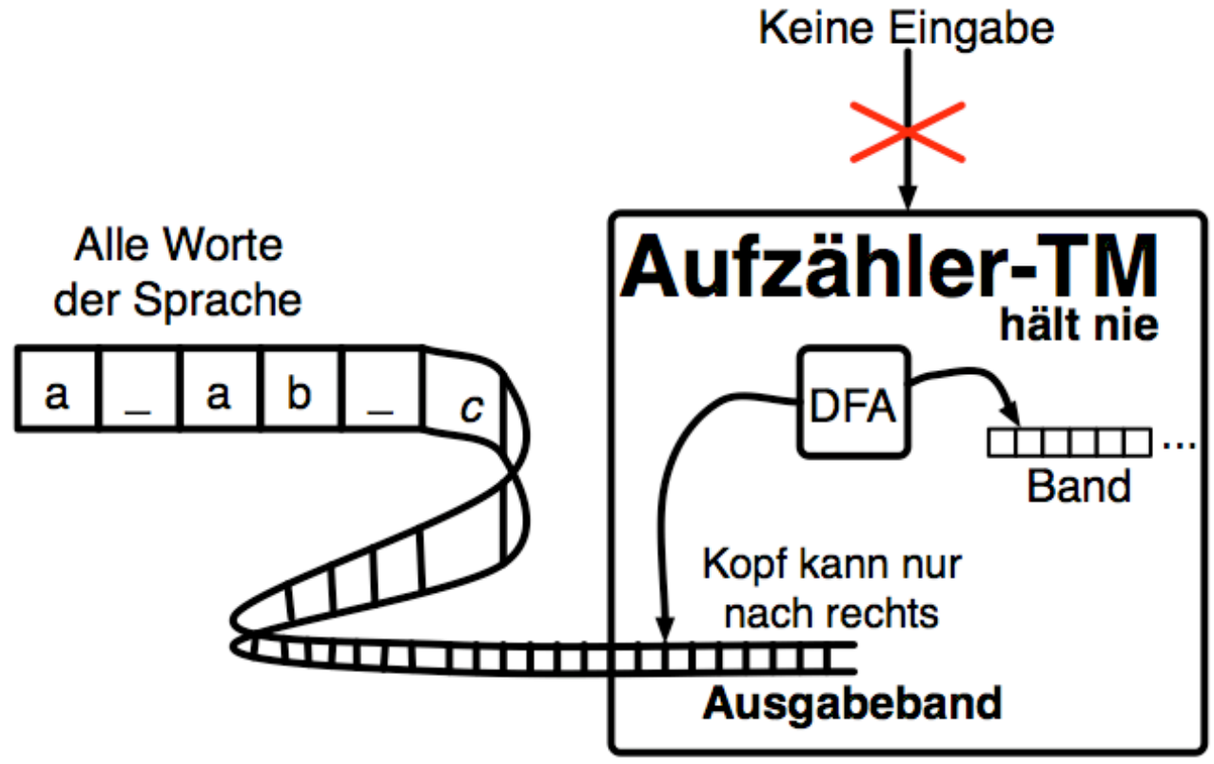


Warum rekursiv aufzählbar rekursiv aufzählbar heißt

- **Definition**
  - Eine **aufzählende Turing-Maschine** ist eine Turingmaschine, die mit einem zusätzlichen speziellen Ausgabe-Band ausgestattet ist.
    - Die Turing-Maschine muss nicht unbedingt halten.
    - Auf dem Ausgabeband kann die Turingmaschine nur nach rechts gehen.
    - Wörter sind durch das Sondersymbol “_” von einander getrennt und können damit weder gelöscht noch überschrieben werden.
    - Die Vereinigung aller jemals erzeugten Wörter, beschreibt die Sprache der aufzählenden Turing-Maschine.
- **Theorem**
  - Eine Sprache ist rekursiv aufzählbar genau dann wenn eine aufzählende Turing-Maschine sie beschreibt.

Akzeptor = Aufzähler





Beweis des Theorems

- **Theorem**
  - Eine Sprache ist rekursiv aufzählbar genau dann wenn eine aufzählende Turing-Maschine sie beschreibt.
- **Beweis (⇐)**
  - Sei U eine Aufzähler-TM für die Sprache A
  - Wir konstruieren eine Akzeptor-TM K wie folgt
  - K = “Auf Eingabe w:
    - Simuliere U.
    - Jedes Mal, wenn U eine Ausgabe macht, vergleiche sie mit w
    - Falls w erscheint, akzeptiere”
- **Beweis (⇒) Sei K eine Akzeptor-TM**
  - Sei $s_1,s_2,..$ eine einfach erzeugbare Folge aller Zeichenketten
  - z.B. längenlexikographisch: ϵ,0,1,00,01,10,11,000,001,...
  - Wir konstruieren eine Aufzähler-TM U wie folgt:
  - U = “
  - Für i =1,2, ..
    - Für jedes w aus $\{s_1,s_2,..,s_i\}$
      - Falls K auf Eingabe w in i Schritten hält und akzeptiert, gib w aus.”
- **Falls K eine Eingabe w in endlicher Zeit akzeptiert, wird sie ausgegeben**
  - sogar beliebig häufig
- **Andere Ausgaben werden von U nicht erzeugt.**

Rekursiv aufzählbar beinhaltet rekursiv entscheidbar

- **Korollar**
  - Jede rekursiv entscheidbare Menge ist rekursiv aufzählbar
- **Beweis:**
  - Betrachte die DTM, die eine rekursiv entscheidbare Menge M entscheidet
  - Diese DTM ist bereits ein Maschine, die M akzeptiert
    - Da die aufzählbare Mengen über die Akzeptor-TM definiert ist

Berechenbarkeitstheorie

NTM

Die Nichtdeterministische Turing-Maschine (NTM)

- **Definition**
  - Eine (**nicht-**deterministische 1-Band) Turingmaschine (NTM) wird beschrieben durch ein 7-Tupel M = $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$.
  - Dabei sind Q, Σ, Γ endliche, nichtleere Mengen
    - Q ist die Zustandsmenge,
    - Σ ist das Eingabealphabet,
    - Γ das Bandalphabet.
    - $q_0 \in Q$ ist der Startzustand.
    - $q_{accept} \in Q$ ist der akzeptierende Endzustand
    - $q_{reject} \in Q$ ist der ablehnende Endzustand
  - δ **: Q** $\times \Gamma \rightarrow$ **P(Q** $\times \Gamma \times$ **{L,R}) ist die Übergangsfunktion.**
  - Sie ist für kein Argument aus $\{q_{accept}, q_{reject}\} \times \Gamma$ definiert.

Konfiguration

- **Momentaufnahme einer NTM**
  - Bei Bandinschrift uv
  - dabei beginnt u am linken des Bandes und hinter v stehen nur Blanks
  - Zustand q,
  - Kopf auf erstem Zeichen von v

- **Konfiguration C = uqv**

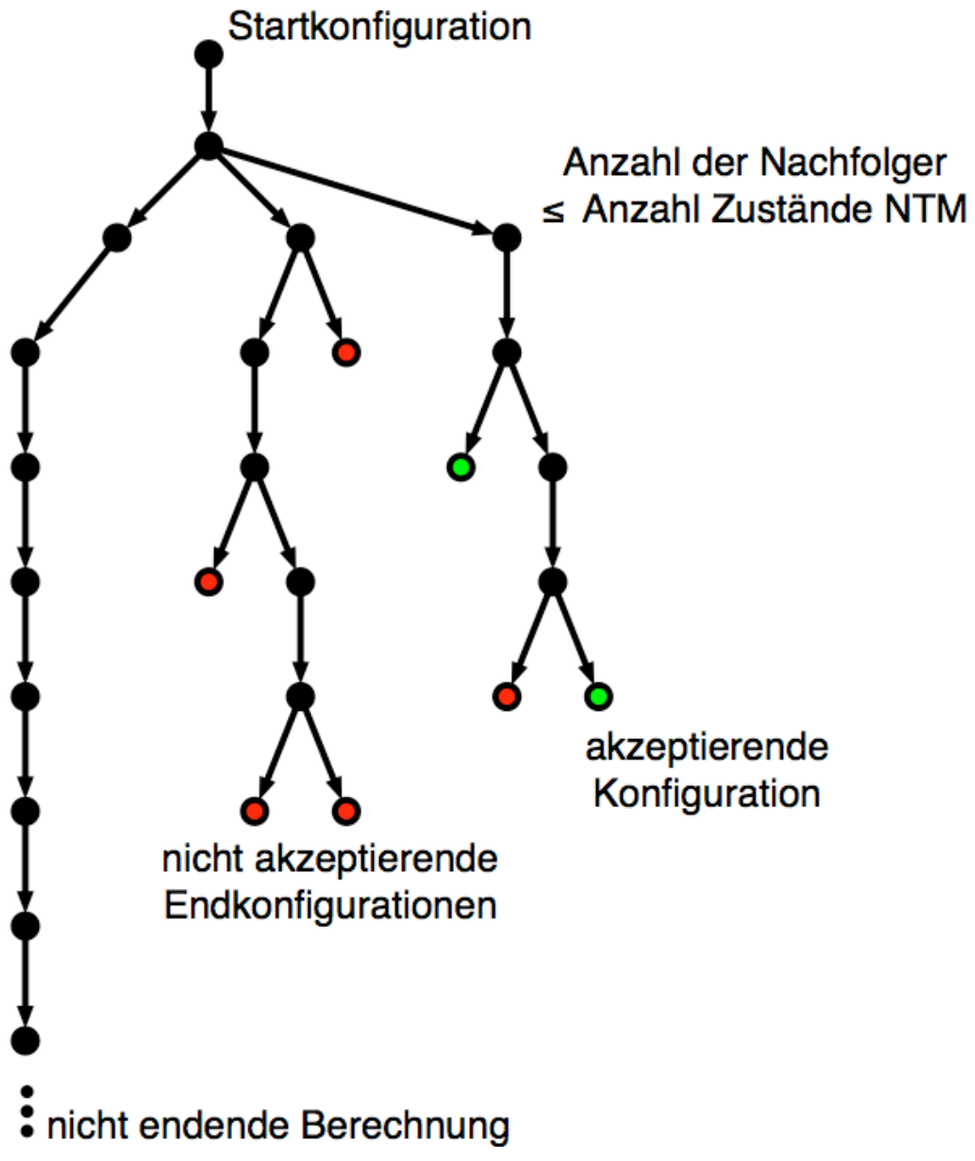
Konfigurationen einer NTM

- **Startkonfiguration**
  - q_0w, wobei w die Eingabe ist
- **Akzeptierende Konfiguration**
  - Konfigurationen mit Zustand q_{accept}
- **Ablehnende Konfiguration**
  - Konfigurationen mit Zustand q_{reject}
- **Haltende Konfiguration**
  - akzeptierende oder ablehnende Konfigurationen

Aufeinanderfolgende Konfigurationen

- **Gegeben: Konfigurationen C_1, C_2**
- **Wir sagen**
  - Konfiguration C_1 führt zu C_2, falls die NTM von C_1 in einem Schritt zu C_2 übergehen kann.
- **Gegeben**
  - Seien a, b, c $\in \Gamma$, u, v $\in \Gamma^*$ und Zustände q_i , q_j gegeben
- **Wir sagen**
  - uaq_ibv führt zu uq_jacv, falls $(q_j,c,L) \in \delta(q_i,b)$ und
  - uaq_ibv führt zu $uacq_jv$, falls $(q_j,c,R) \in \delta(q_i,b)$

Berechnungsbaum einer NTM

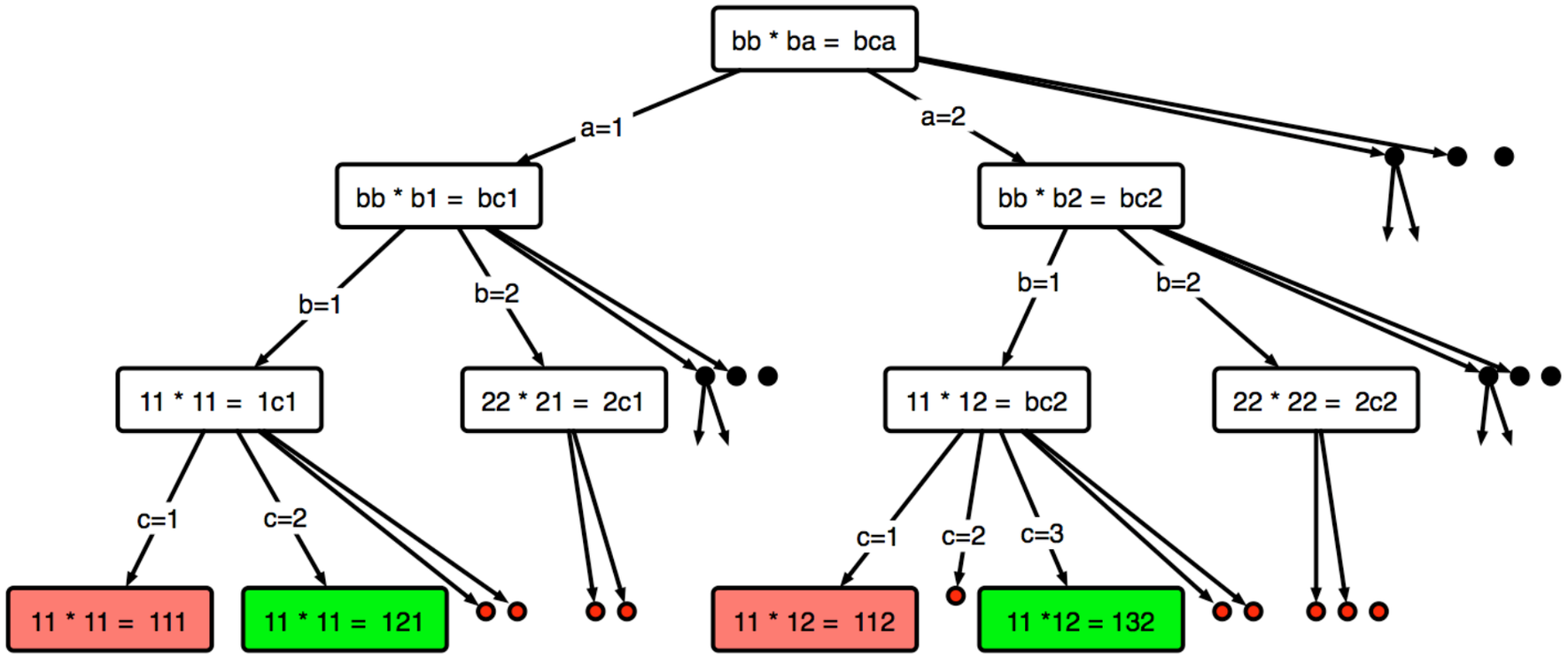


Akzeptanz einer NTM

- **Eine Nichtdeterministische Turingmaschine M akzeptiert eine Eingabe w, falls es eine Folge von Konfigurationen C_1, C_2, …, C_k gibt, so dass**
  - C_1 ist die Startkonfiguration von M bei Eingabe w
  - C_i kann zu C_{i+1} überführt werden
  - C_k ist eine akzeptierende Konfiguration
- **Die von M akzeptierten Worte bilden die von M akzeptierte Sprache L(M)**
- **Eine NTM entscheidet eine Sprache, wenn jede Eingabe zu einem endlichen Berechnungsbaum der Konfigurationen führt.**

Die Nützlichkeit einer NTM

- **NTMs können “raten”**
- **Beispielproblem:**
  - Gibt es eine Lösung für das Zahlenrätsel:
    - bb * ba = bca
- **NTM:**
  - Für alle Buchstaben
  - “rate” eine Möglichkeit
  - setze Buchstaben ein
  - Verifiziere Gleichung wobei jeder Buchstabe für eine Ziffer steht

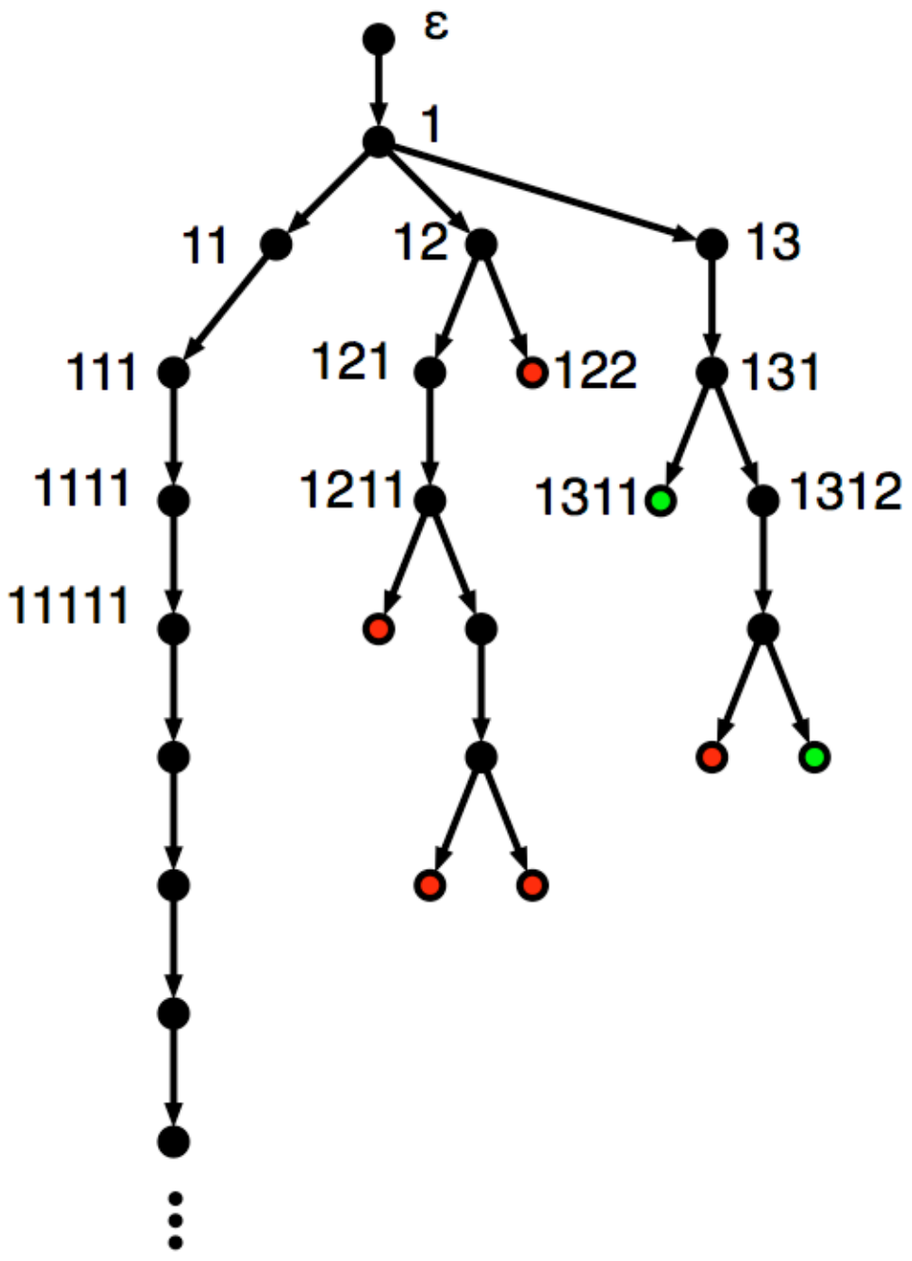


Äquivalenz von DTM und NTM

- **Theorem**
  - Zu jeder nichtdeterminischen Turingmaschine M gibt es eine äquivalente deterministische Turingmaschine D, d.h.
    - Falls M auf Eingabe x akzeptiert, dann akzeptiert D auf x und hält
    - Falls M auf x nicht akzeptiert, dann akzeptiert D nicht
    - Falls M auf allen Berechnungspfaden hält, dann hält auch D

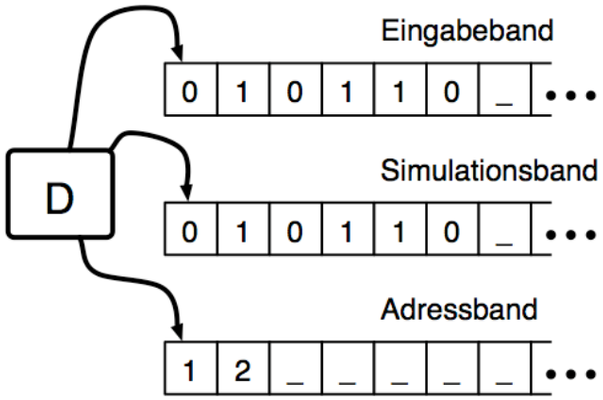
Äquivalenz von DTM und NTM

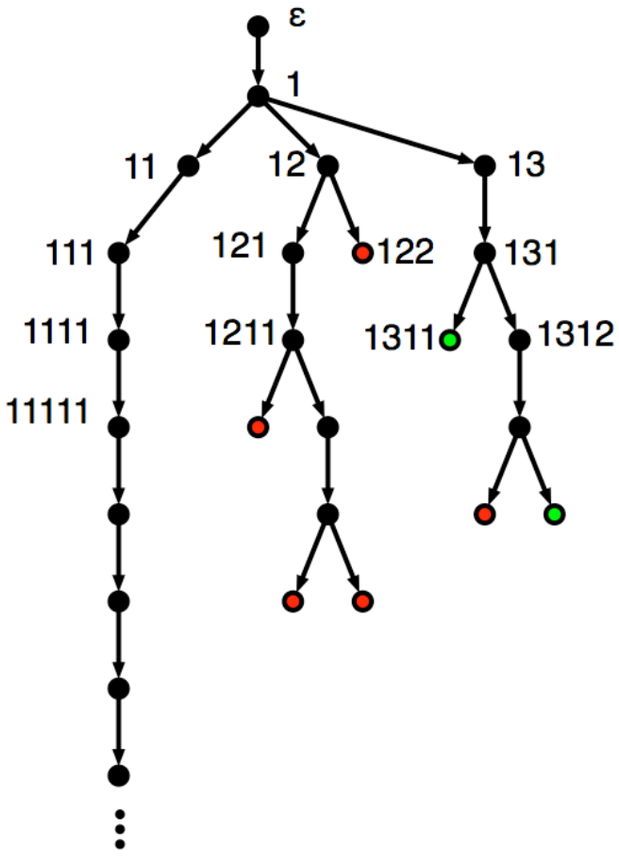
- **Theorem**
  - Zu jeder nichtdeterminischen Turingmaschine M gibt es eine äquivalente deterministische Turingmaschine D.
- **Beweisidee:**
- **Simuliere Berechnungsbaum**
  - ausgehend von Startkonfiguration
- **Verwende BFS (breadth first search)**
  - DFS nicht möglich
- **DTM D:**
  - Drei Bänder:
    - Eingabeband
    - Simulationsband
    - Adressband



Äquivalenz von DTM und NTM

- **Beweisidee:**
  - Simulator DTM D:
  - Drei Bänder:
    - Eingabeband
    - Simulationsband
    - Adressband
  - Zähle alle Knoten des Berechnungsbaums im Adressband auf. Sortierung:
    - zuerst Länge
    - dann lexikographisch
      - ϵ,
      - 1,2,3,
      - 11,12,13, 21,22,23, 31,32,33,
      - 111,112,...
  - Führe die (nun deterministische) Berechnung auf dem Berechnungspfad mit dem Simulationsband durch



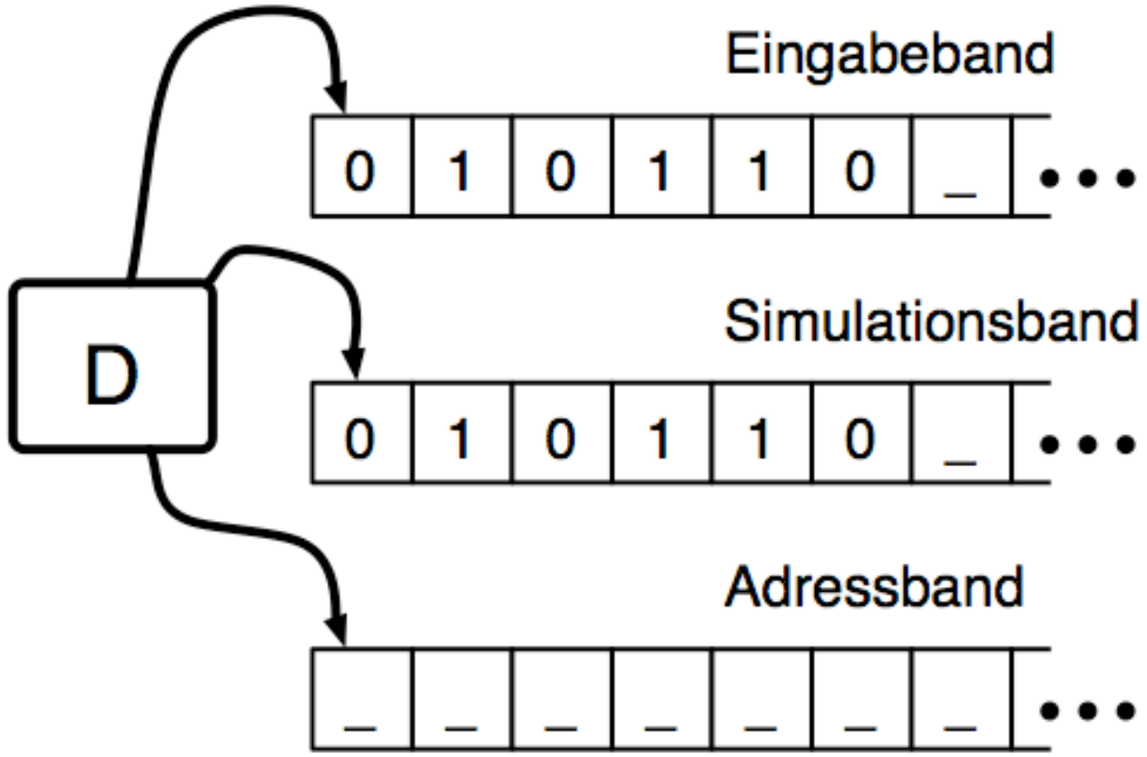


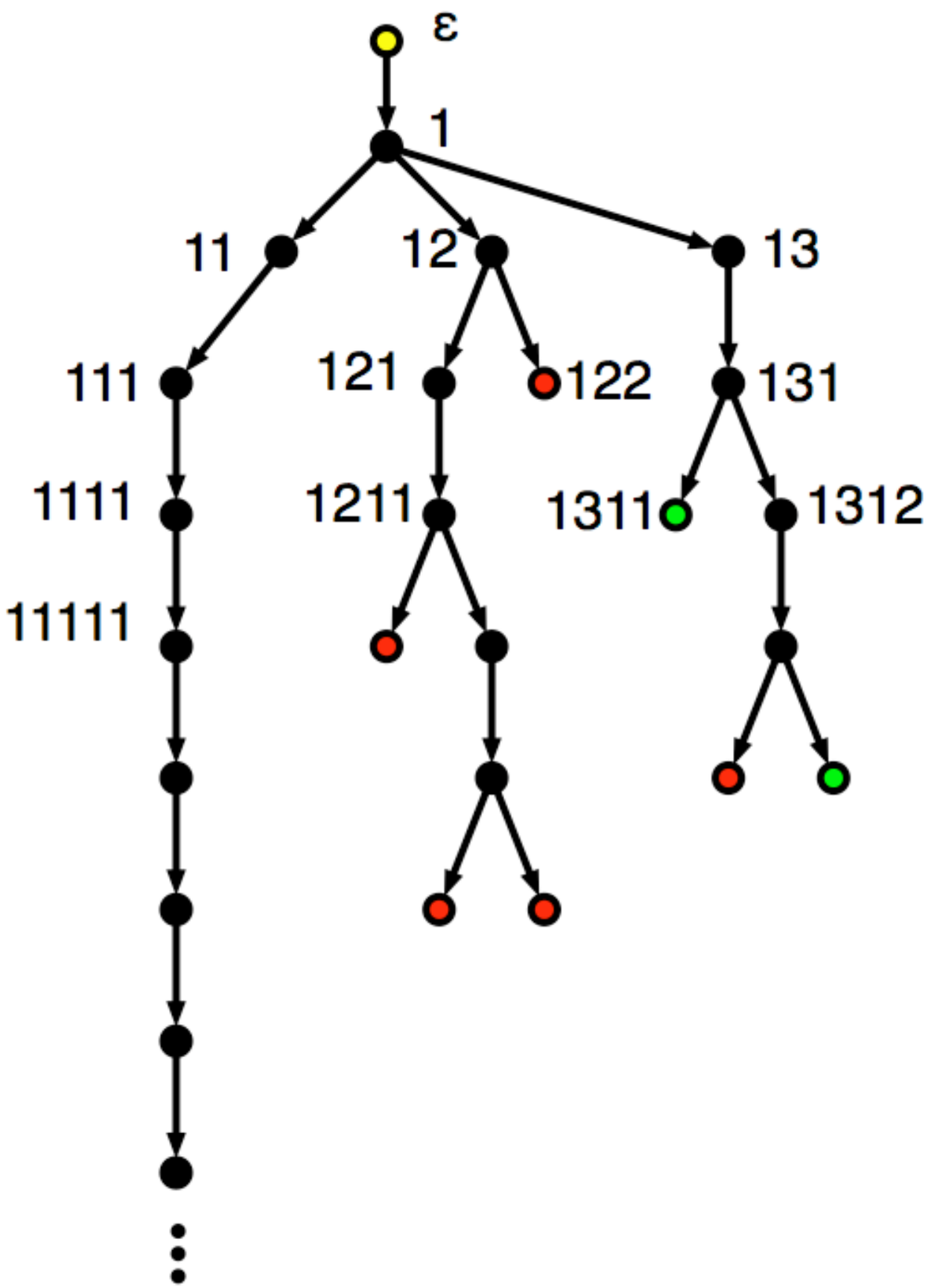
NTMs und DTMs sind äquivalent
Beweis (1.)

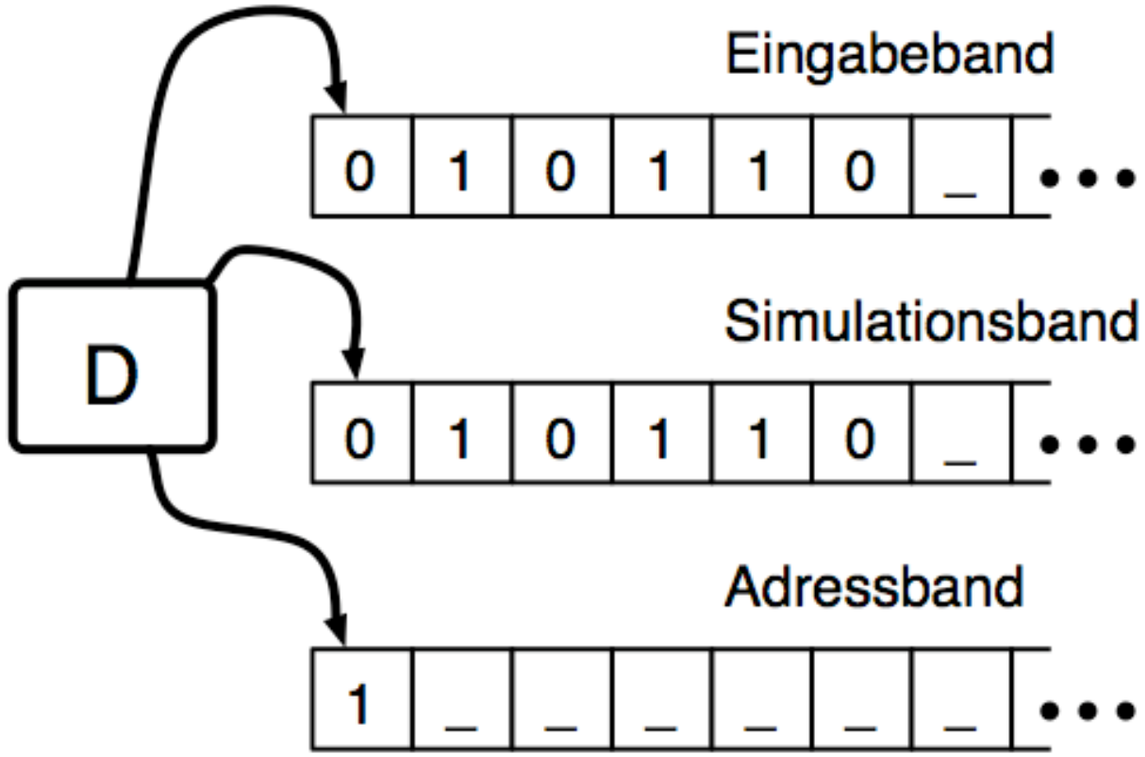
- **Beweis:**
  - Die simulierende DTM hat drei Bänder
    - Wir wissen, dass 3-Band DTMs und DTMs äquivalent sind
  - Auf Band 1 wird das Eingabewort geschrieben.
    - Dieses Band bleibt während der ganzen Berechnung unverändert
  - Auf Band 2 wird eine Kopie des Bandes der NTM M in einer Konfiguration im Berechnungsbaums gespeichert
  - Auf Band 3 wird die Position der NTM M im Berechnungsbaum nachverfolgt
  - Jeder Knoten im Berechnungsbaum kann höchstens b=2 |Q| |Γ| Nachfolger haben,
    - wobei Q die Zustandsmenge der NTM M ist und Γ das Bandalphabet
  - Wir betrachten daher das Alphabet {1,2,..,b}
    - Die Adresse 231 steht dann für die 2. Möglichkeit des Berechnungspfades ab der Startkonfiguration
      - dessen 3. Kindes
      - und dessen 1. Kindes
  - Die Kopfposition auf dem Band beschreibt die Tiefe im Berechnungsbaumes während der Simulation der NTM
  - Das leere Wort beschreibt die Startkonfiguration

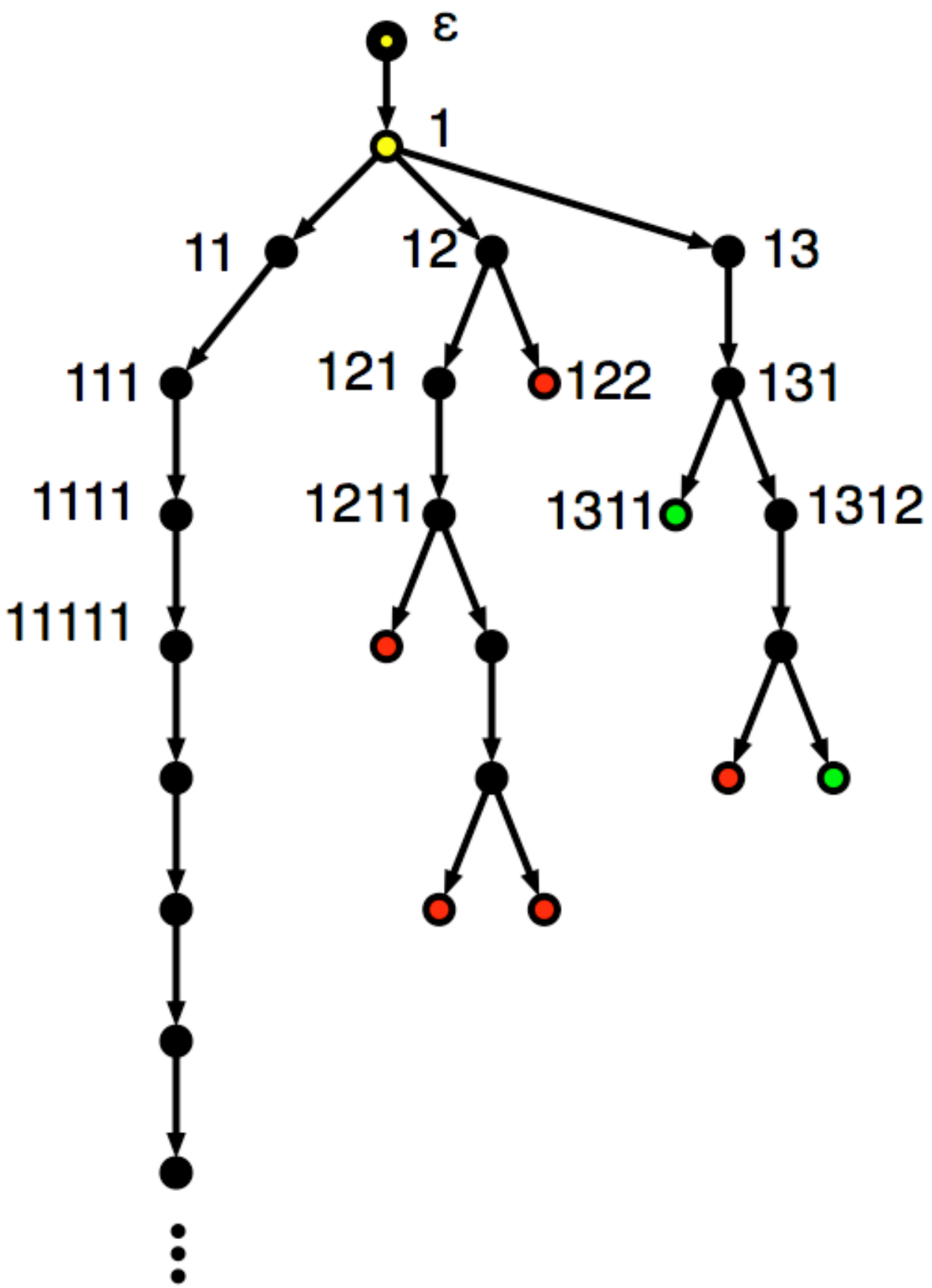
NTMs und DTMs sind äquivalent
Beweis (2.)

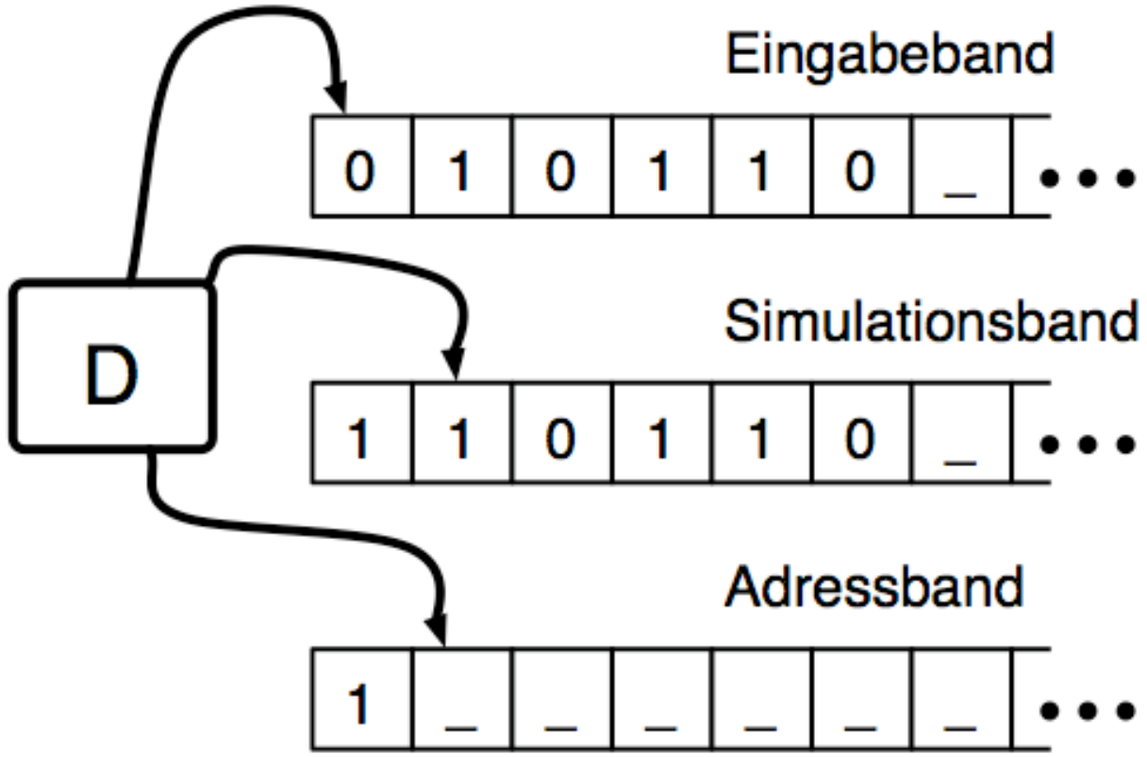
- **Beweis (Fortsetzung):**
- **Die DTM D arbeitet wie folgt:**
  - Band 1 wird mit der Eingabe w beschrieben
  - Band 2 und Band 3 sind leer
  - 1. Schritt: Kopiere Band 1 auf Band 2
    - Benutze Band 2 um einen Schritt der nichtdeterministischen Berechnung der NTM M zu berechnen.
      Hierzu wird die x. Möglichkeit durchgeführt, wobei x das aktuelle Zeichen auf Band 3 ist.
  - 2. Schritt: Ist x = "_", wird keine Berechnung durchgeführt und überprüft, ob M im akzeptierenden Zustand ist.
    - Falls ja, akzeptiert D und hält
    - Gibt es keine x. Möglichkeit oder ist der Zustand verwerfend gehe zu Schritt 4
  - 3. Schritt: Ansonsten bewege den Kopf auf Band 3 nach rechts und wiederhole Schritt 2
  - 4. Schritt: Ersetze die Zeichenkette auf Band 3 durch die lexikographisch nächste Zeichenkette gleicher Länge.
    - Gibt es keine dieser Länge mehr, ersetze sie durch die lexikographisch erste Zeichenkette der nächstgrößeren Länge, falls mindestens eine Berechnung für diese Adresslänge abgebrochen wurde
    - Ansonsten, verwirft D und hält
- **Die hier beschriebene DTM erfüllt die gewünschte Eigenschaft.**

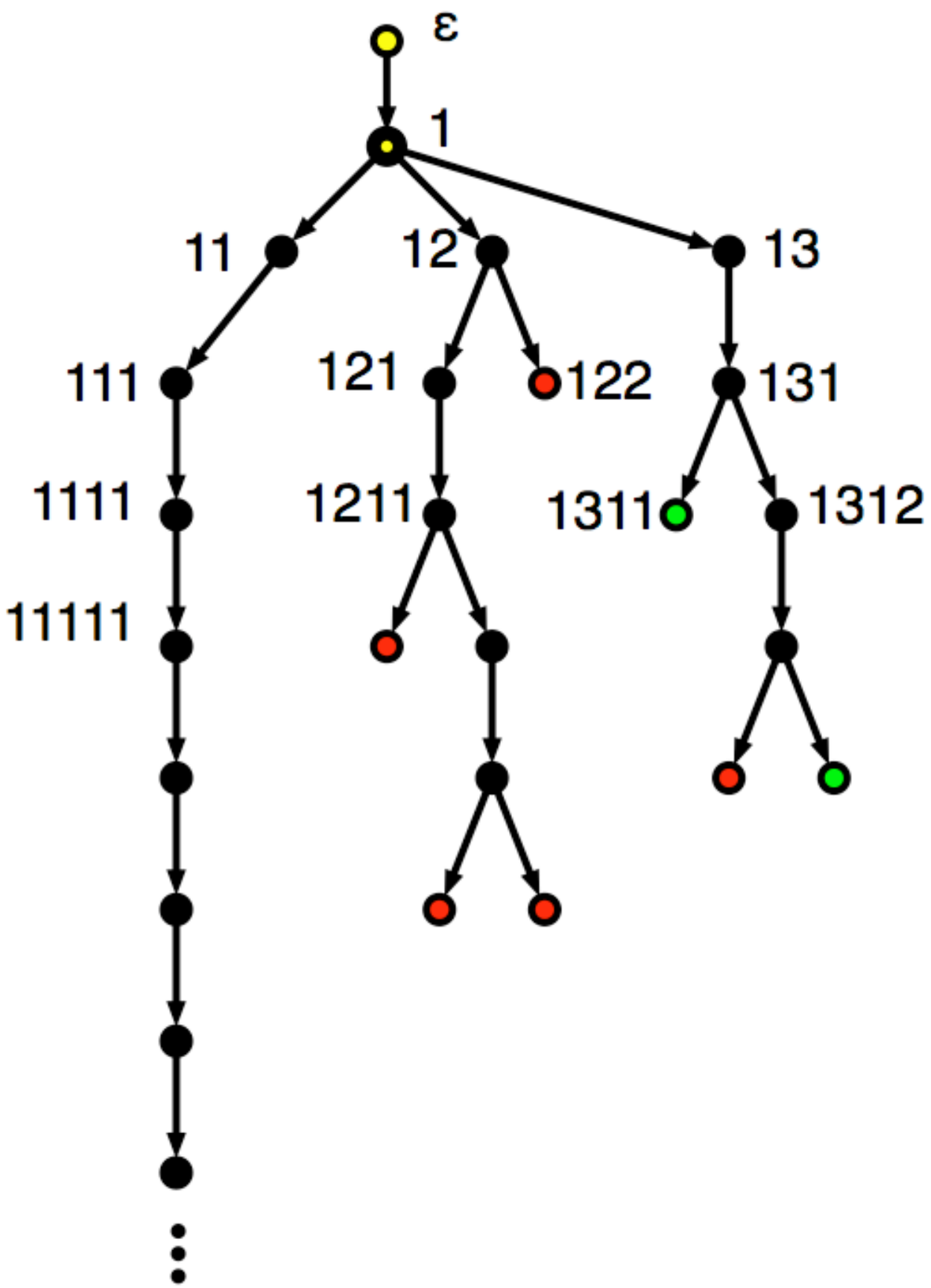

Eingabeband
0 1 0 1 1 0 _
Simulationsband
0 1 0 1 1 0 _
Adressband
_ _ _ _ _ _ _
D

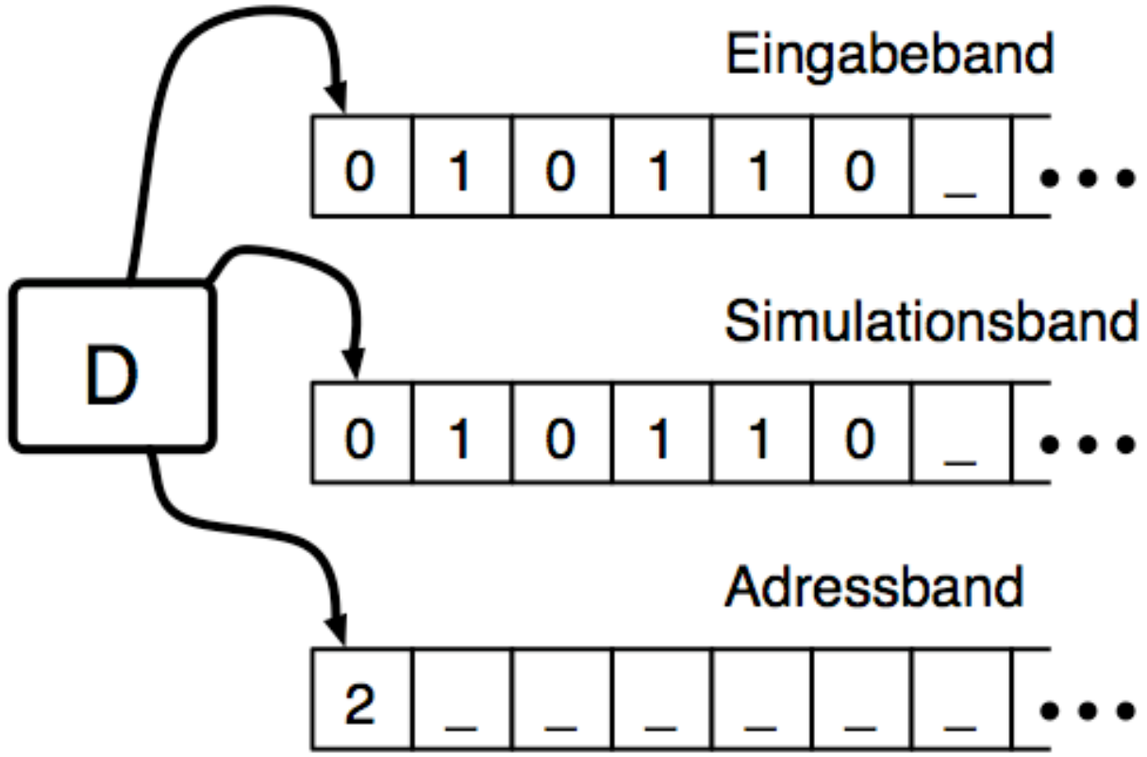

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

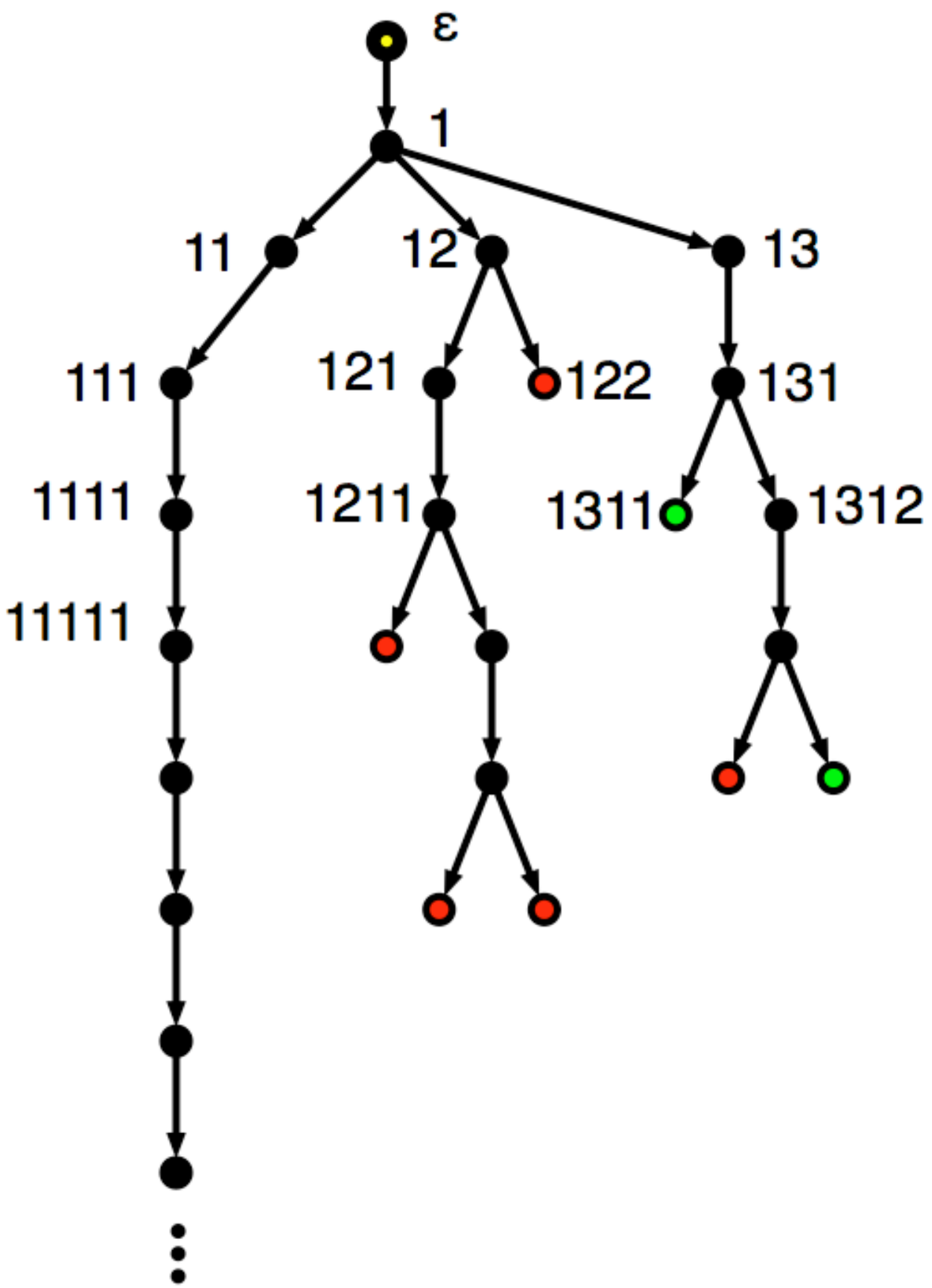

Eingabeband
0 1 0 1 1 0 _
Simulationsband
0 1 0 1 1 0 _
Adressband
1 _ _ _ _ _ _
D

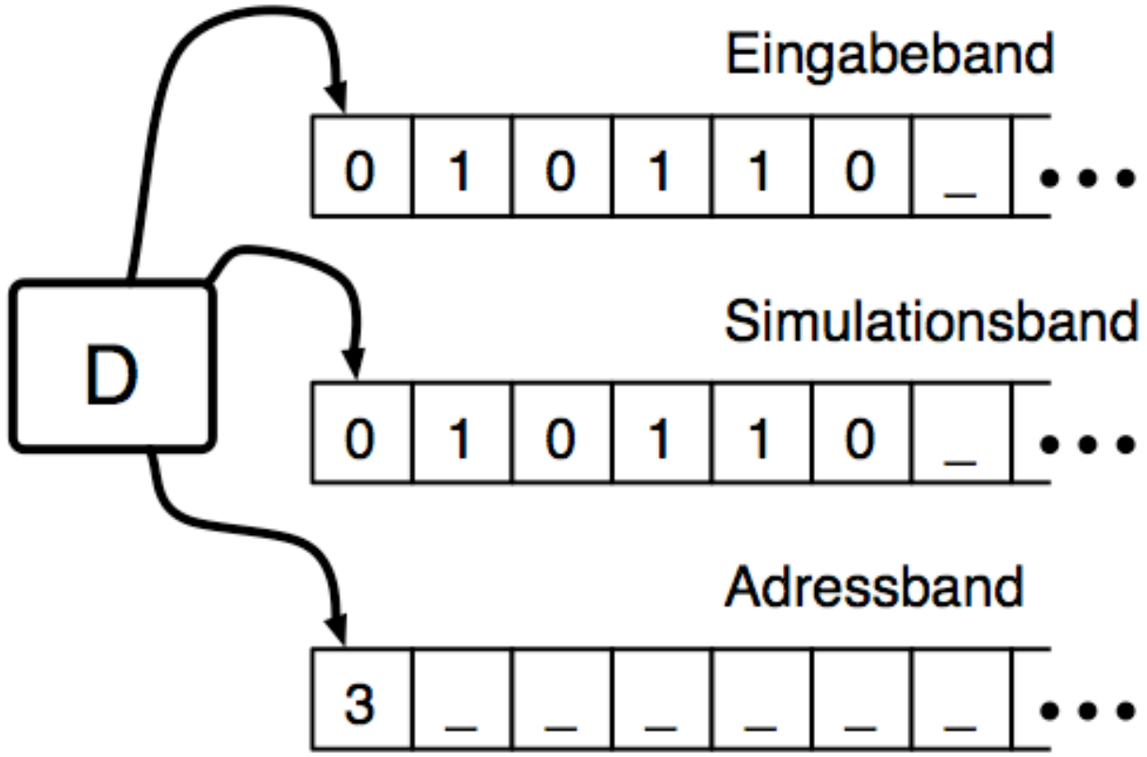

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

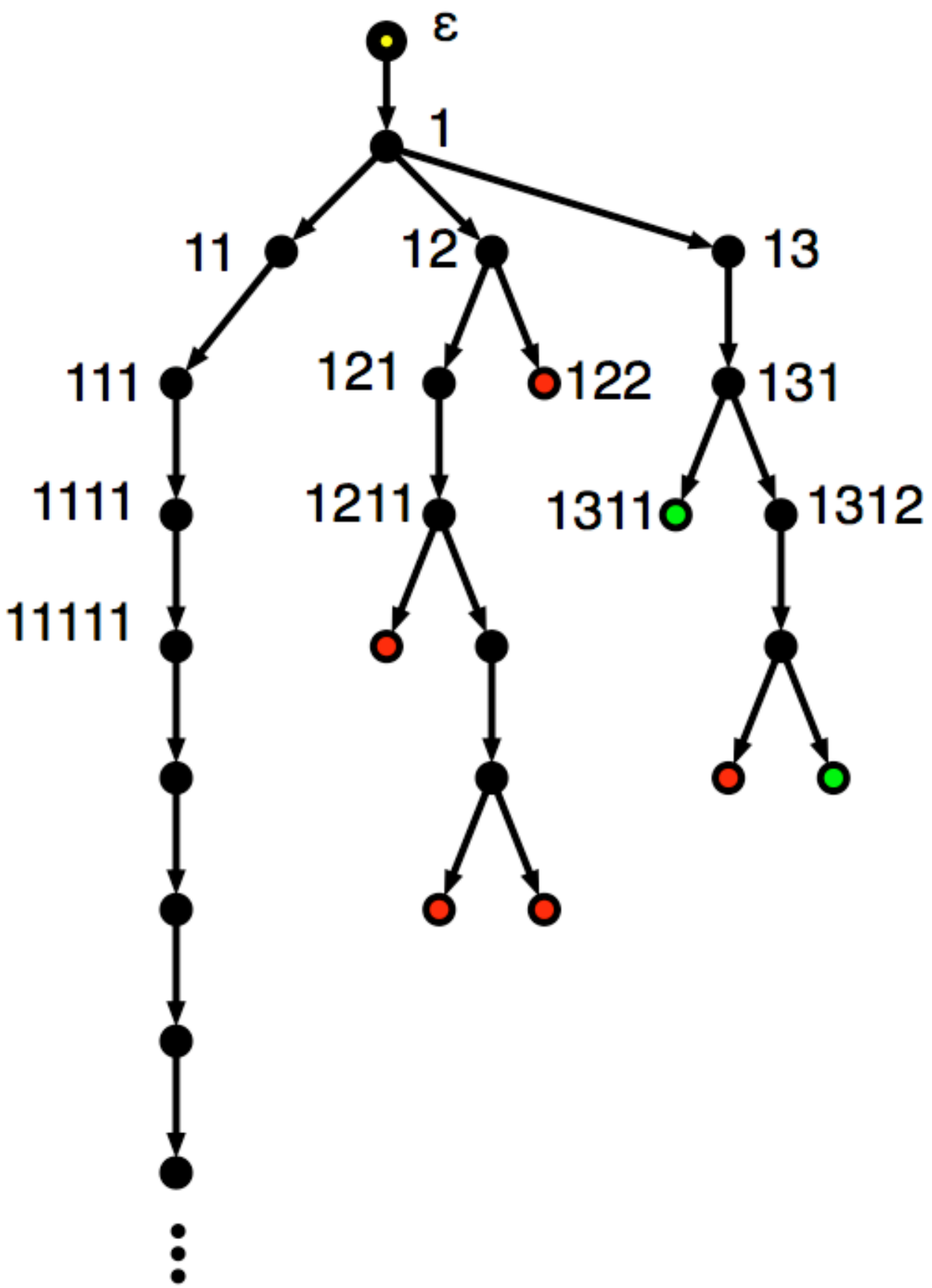

Eingabeband
0 1 0 1 1 0 _
Simulationsband
1 1 0 1 1 0 _
Adressband
1 _ _ _ _ _ _
D

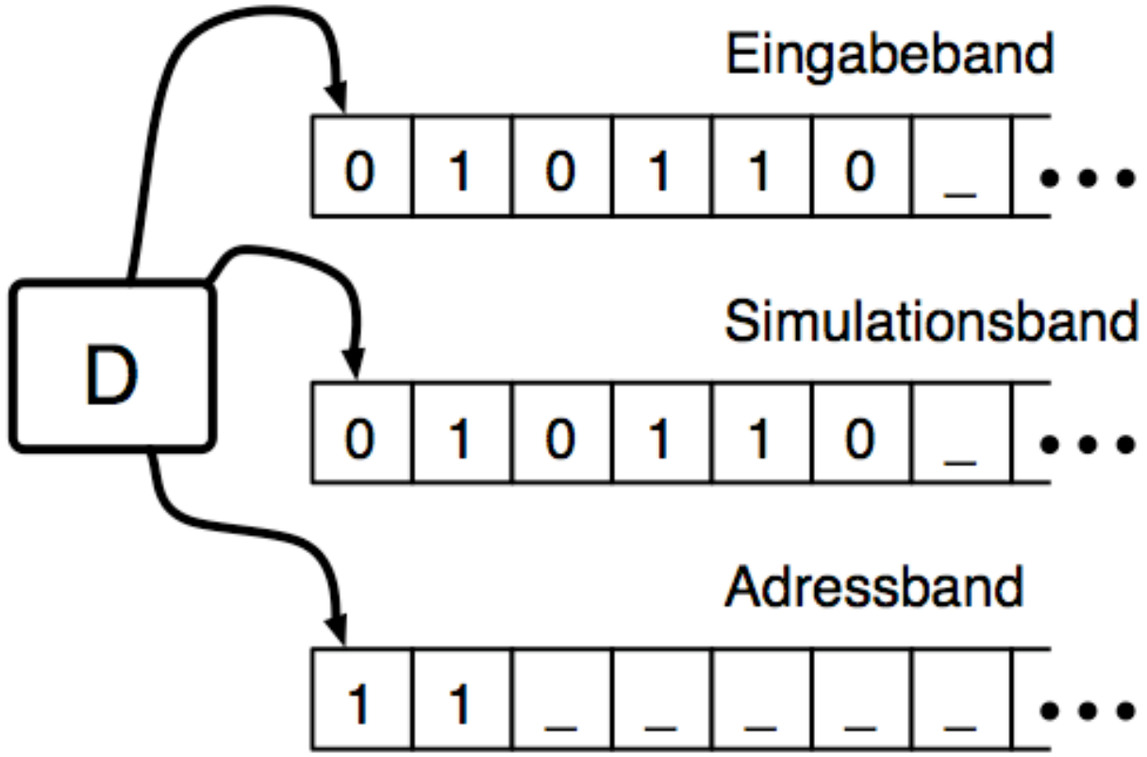

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

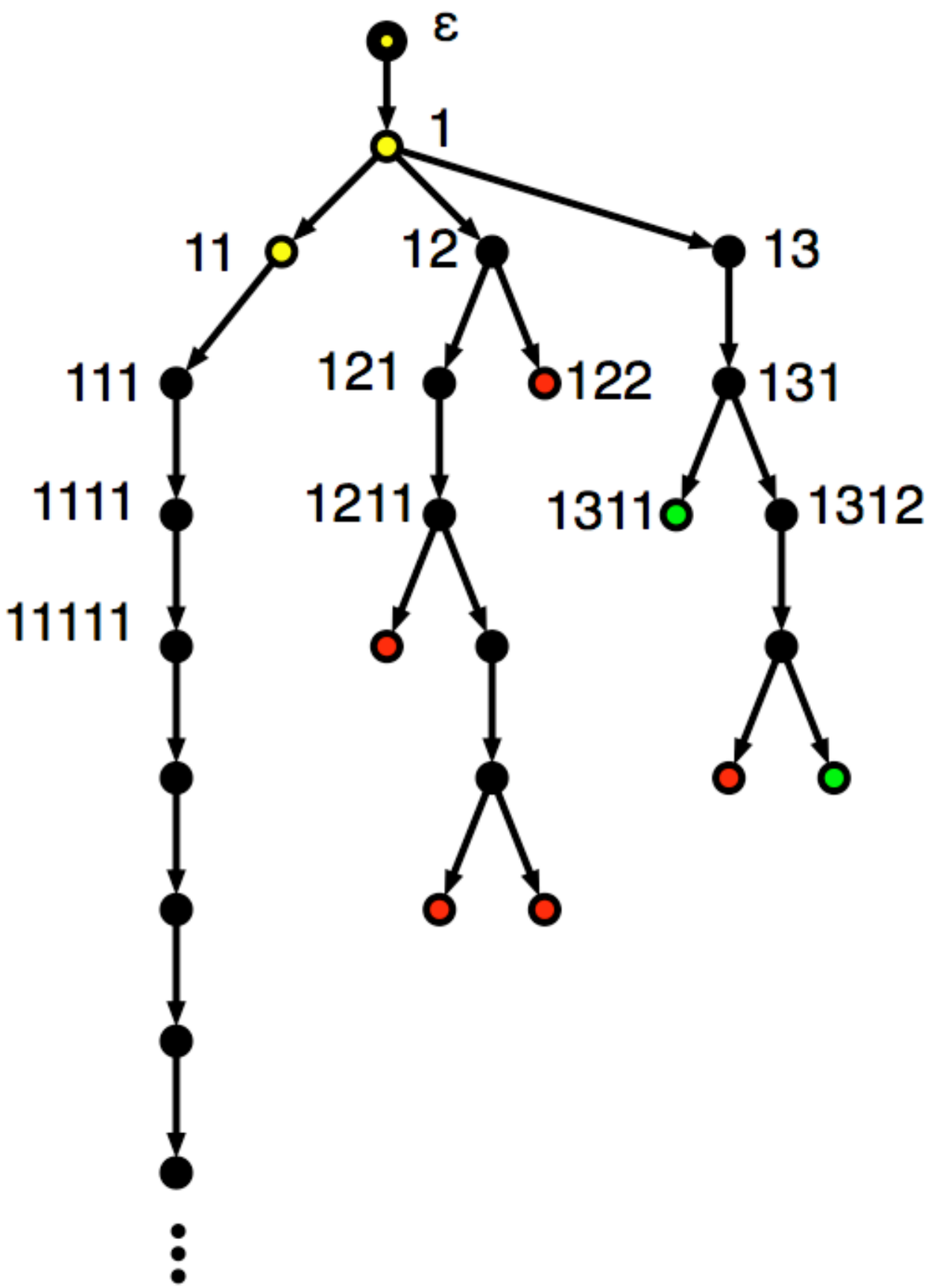

Eingabeband
0 1 0 1 1 0 _
Simulationsband
0 1 0 1 1 0 _
Adressband
2 _ _ _ _ _ _
D

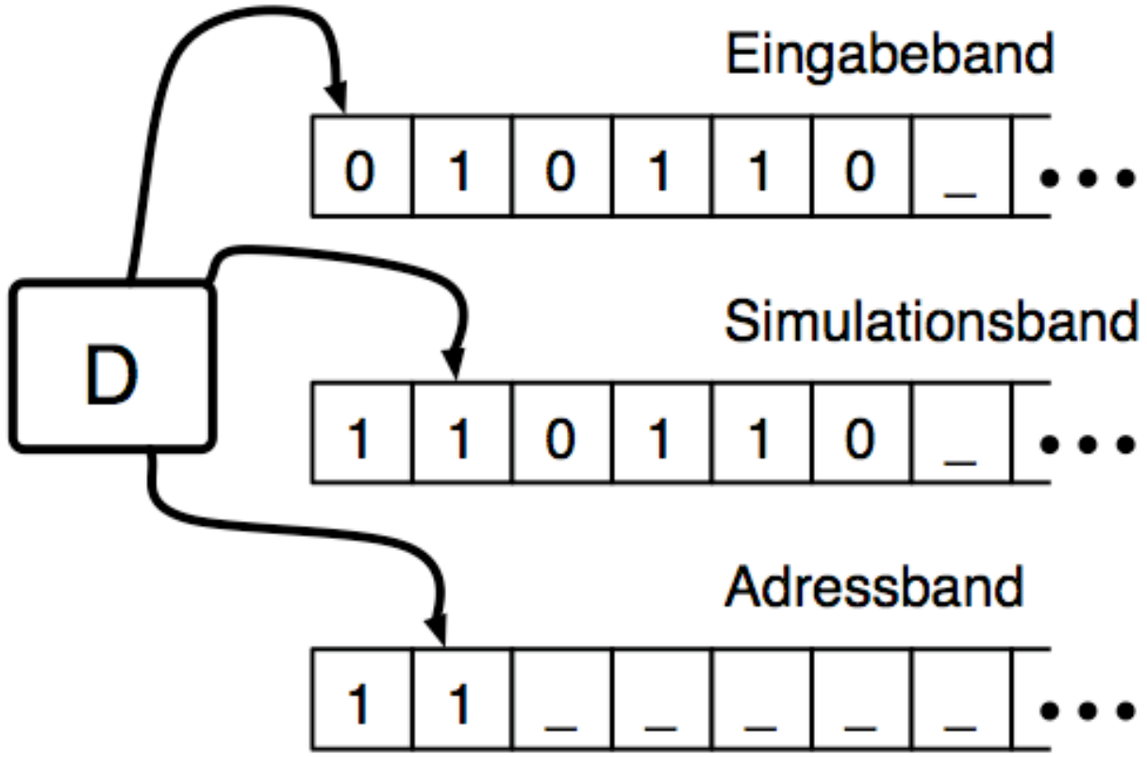

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

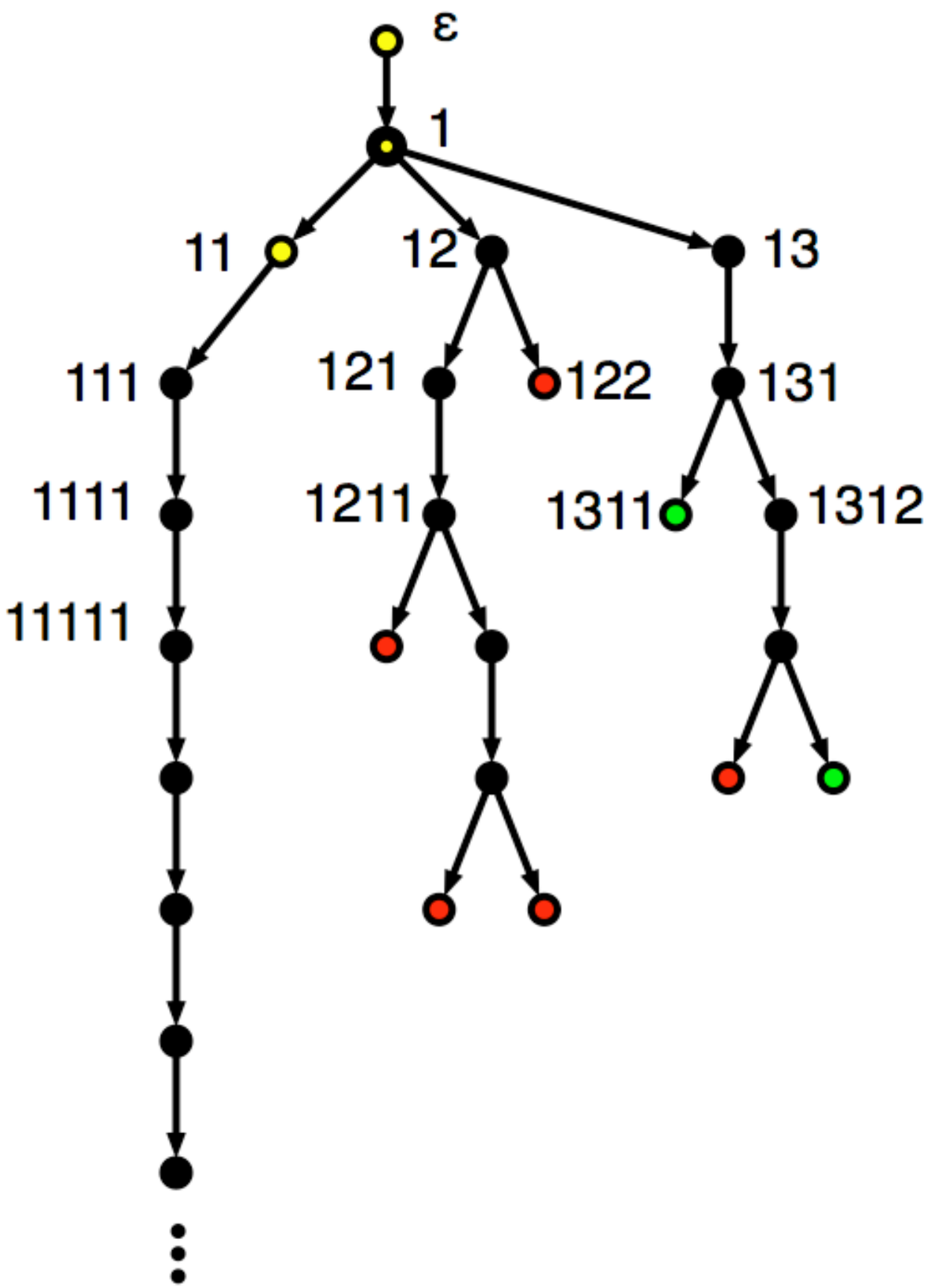

Eingabeband
0 1 0 1 1 0 _
Simulationsband
0 1 0 1 1 0 _
Adressband
3 _ _ _ _ _ _
D

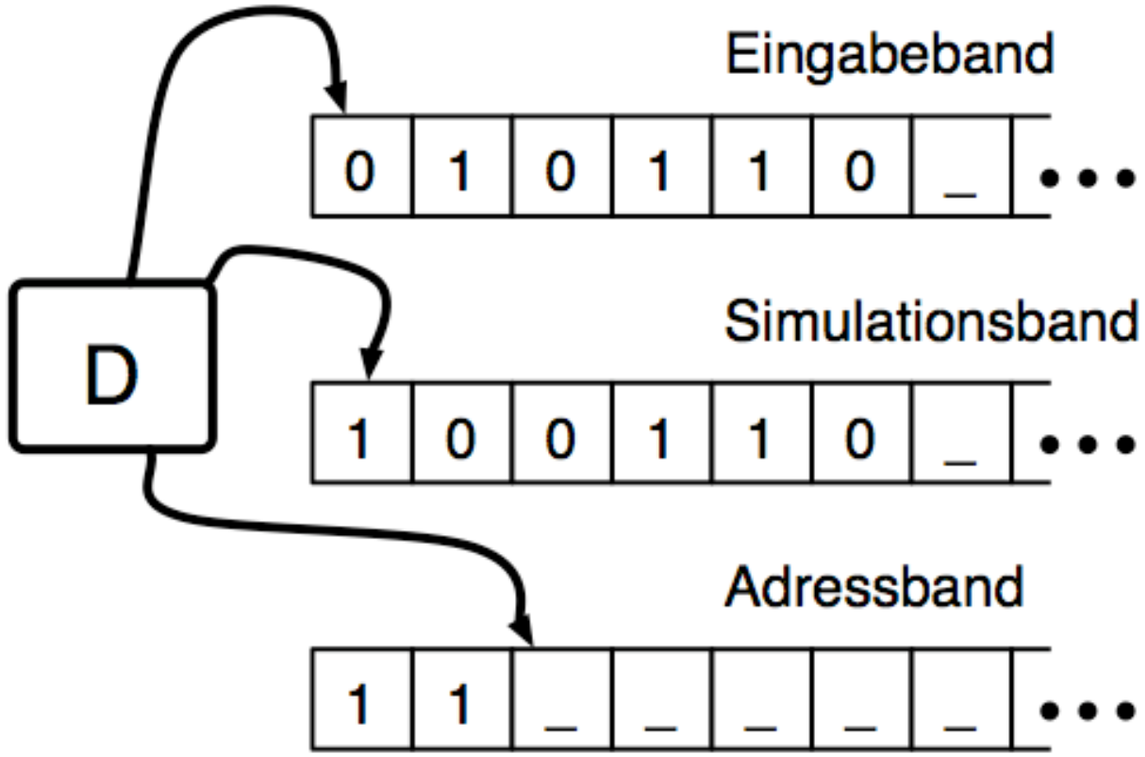

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

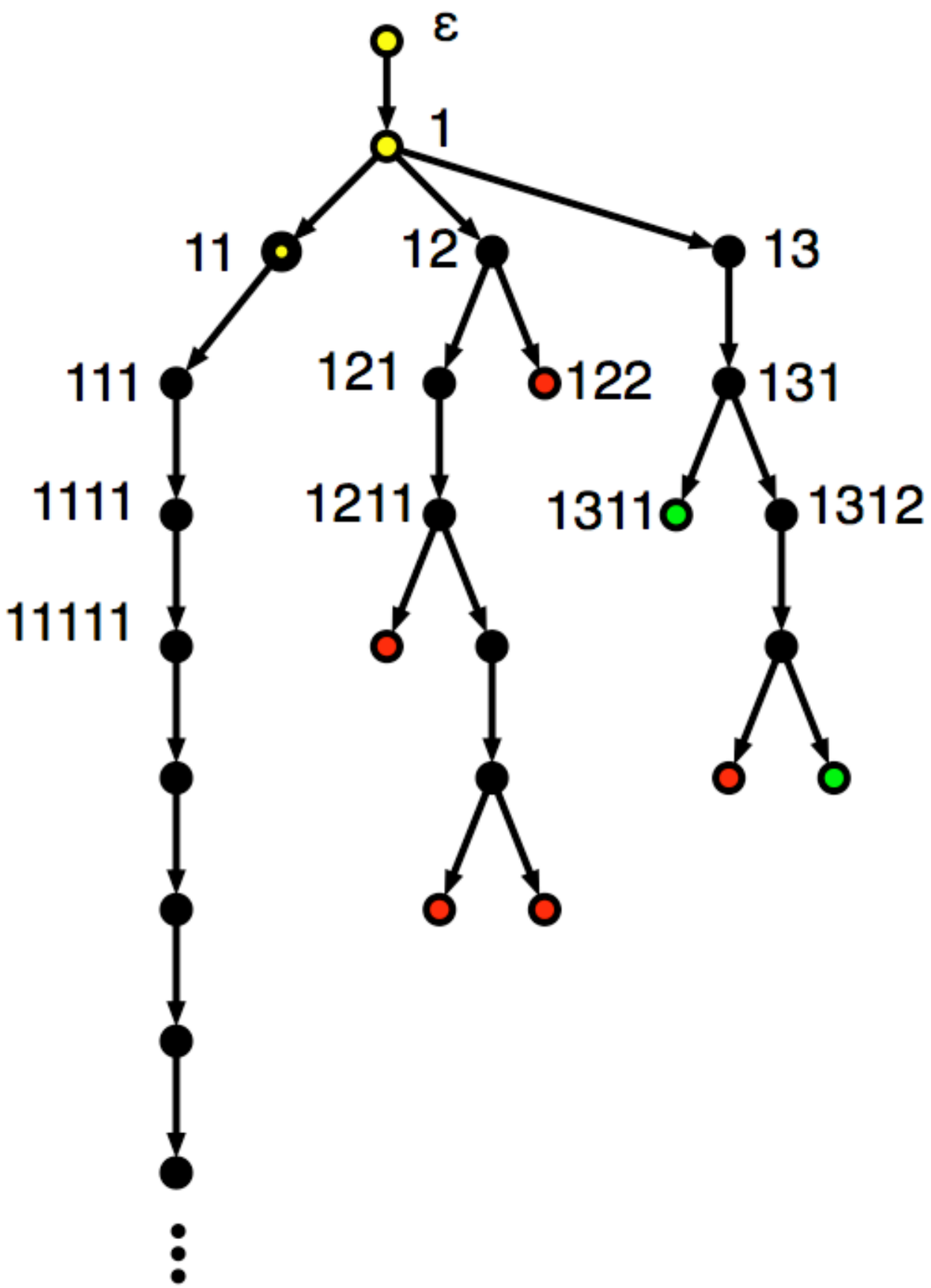

Eingabeband
0 1 0 1 1 0 _
D
Simulationsband
0 1 0 1 1 0 _
Adressband
1 1 _ _ _ _ _

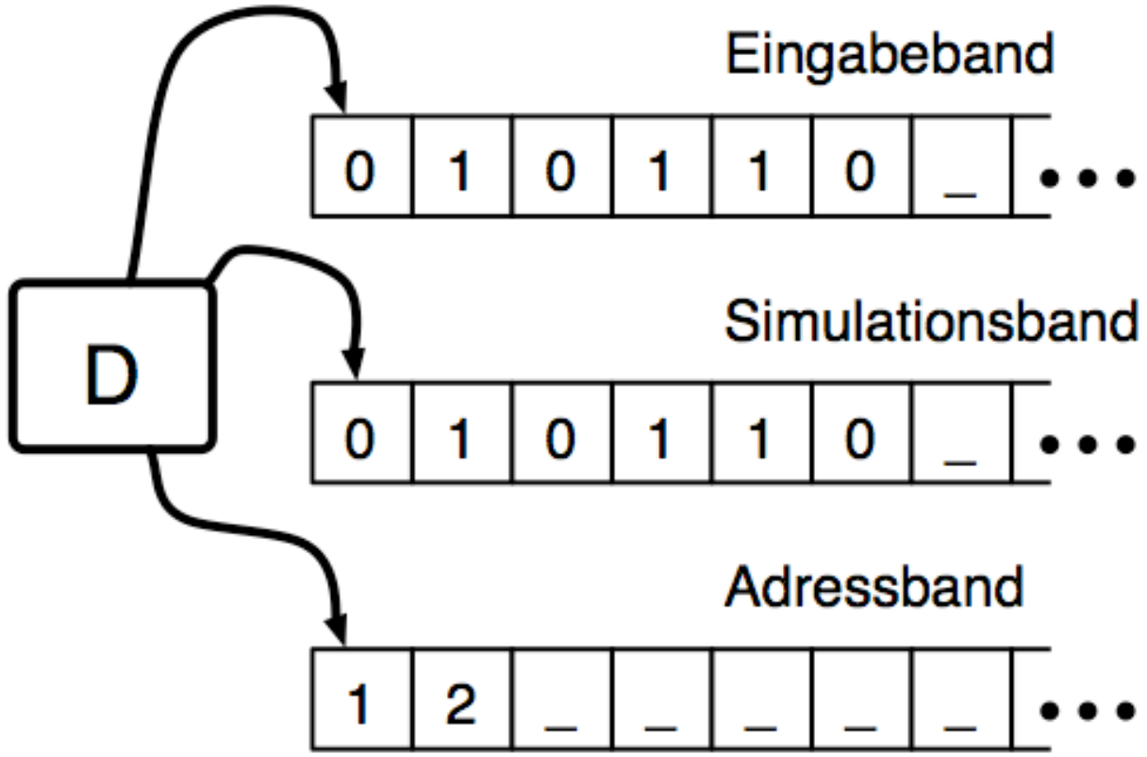

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

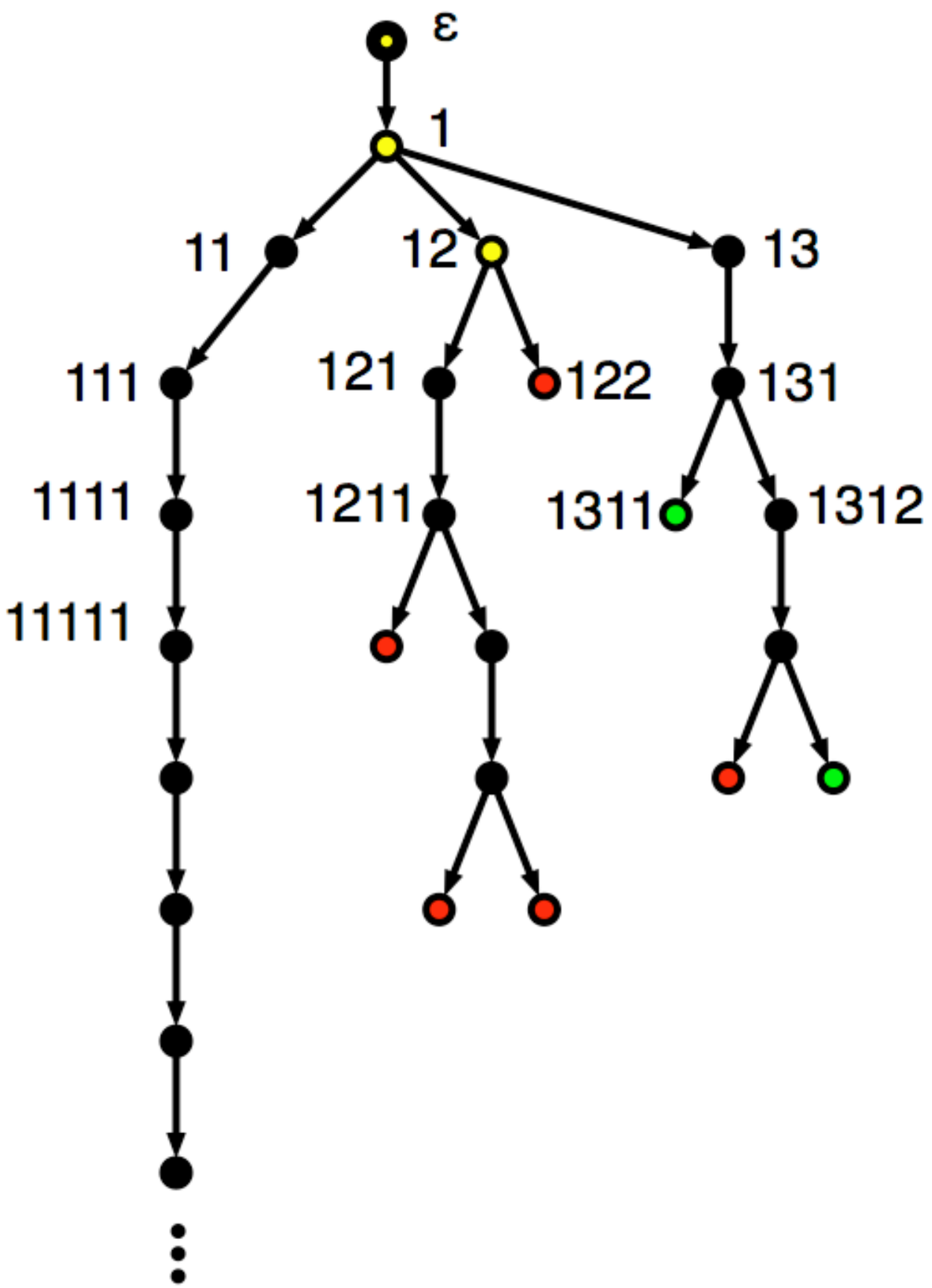

Eingabeband
0 1 0 1 1 0 _
Simulationsband
1 1 0 1 1 0 _
Adressband
1 1 _ _ _ _ _
D

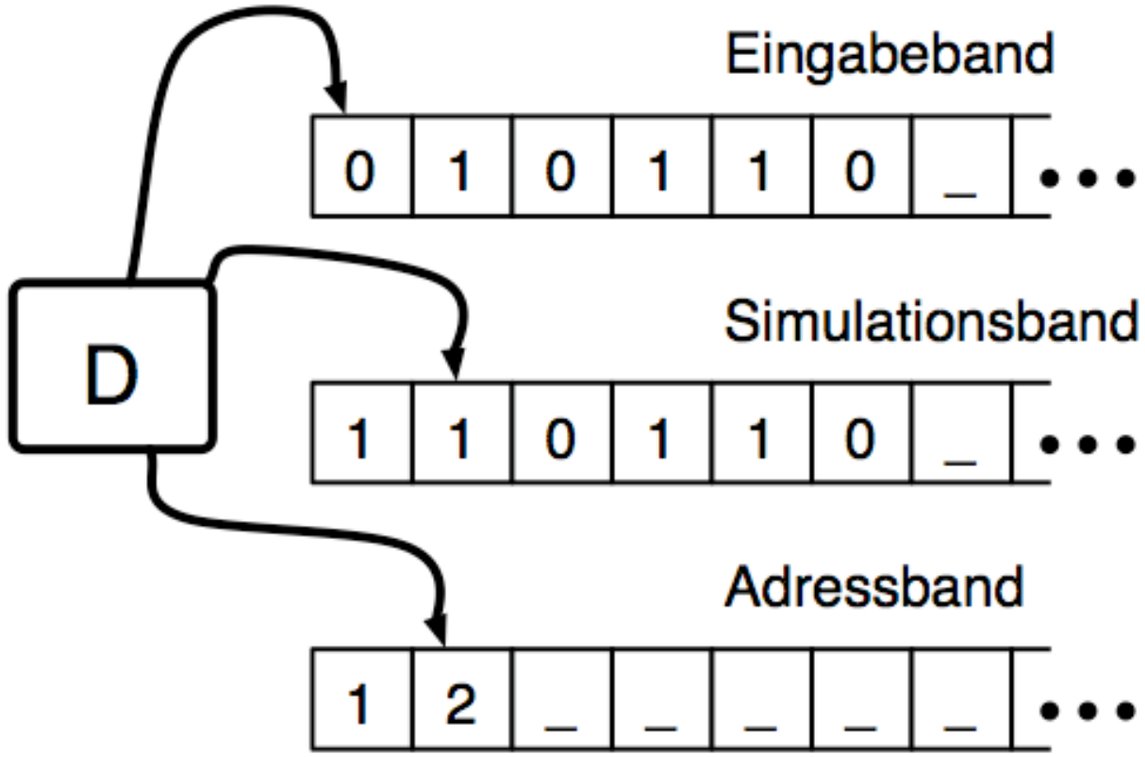

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

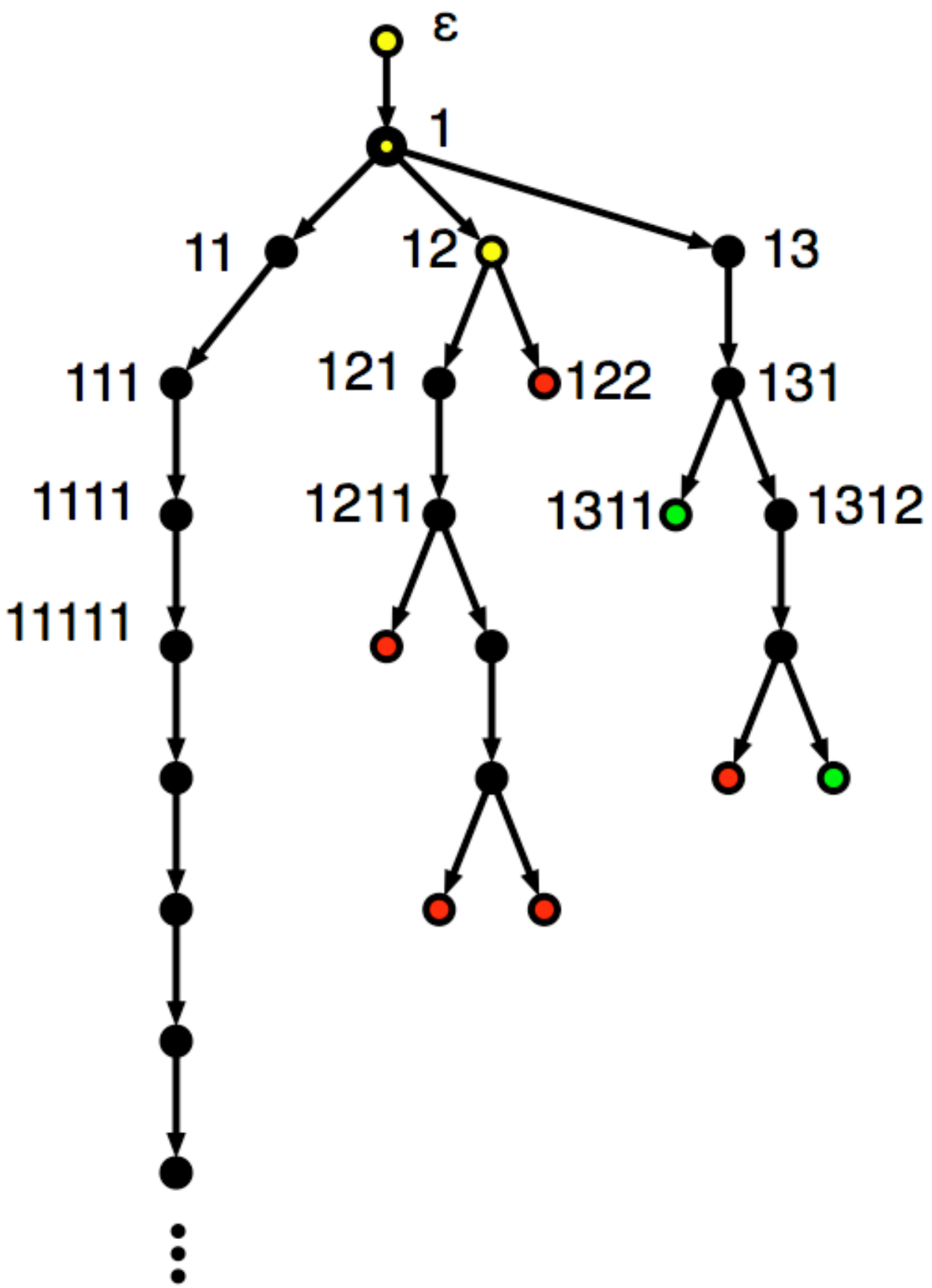

Eingabeband
0 1 0 1 1 0 _
Simulationsband
1 0 0 1 1 0 _
Adressband
1 1 _ _ _ _ _
D

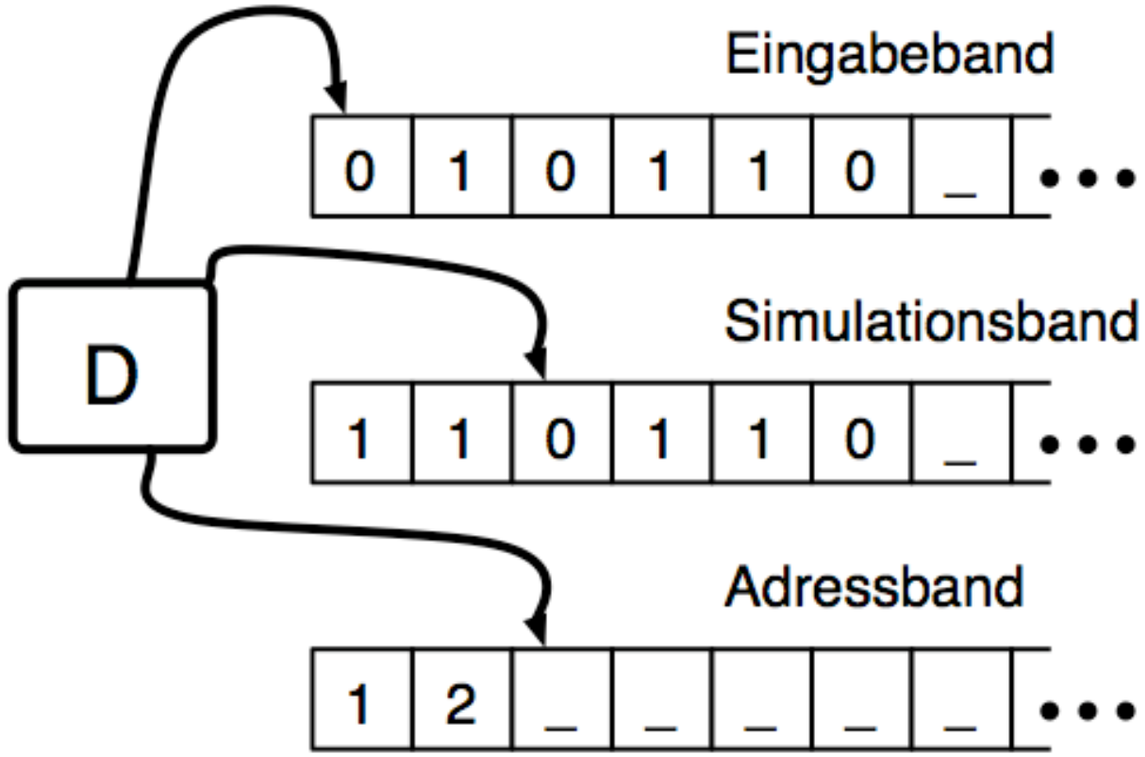

ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

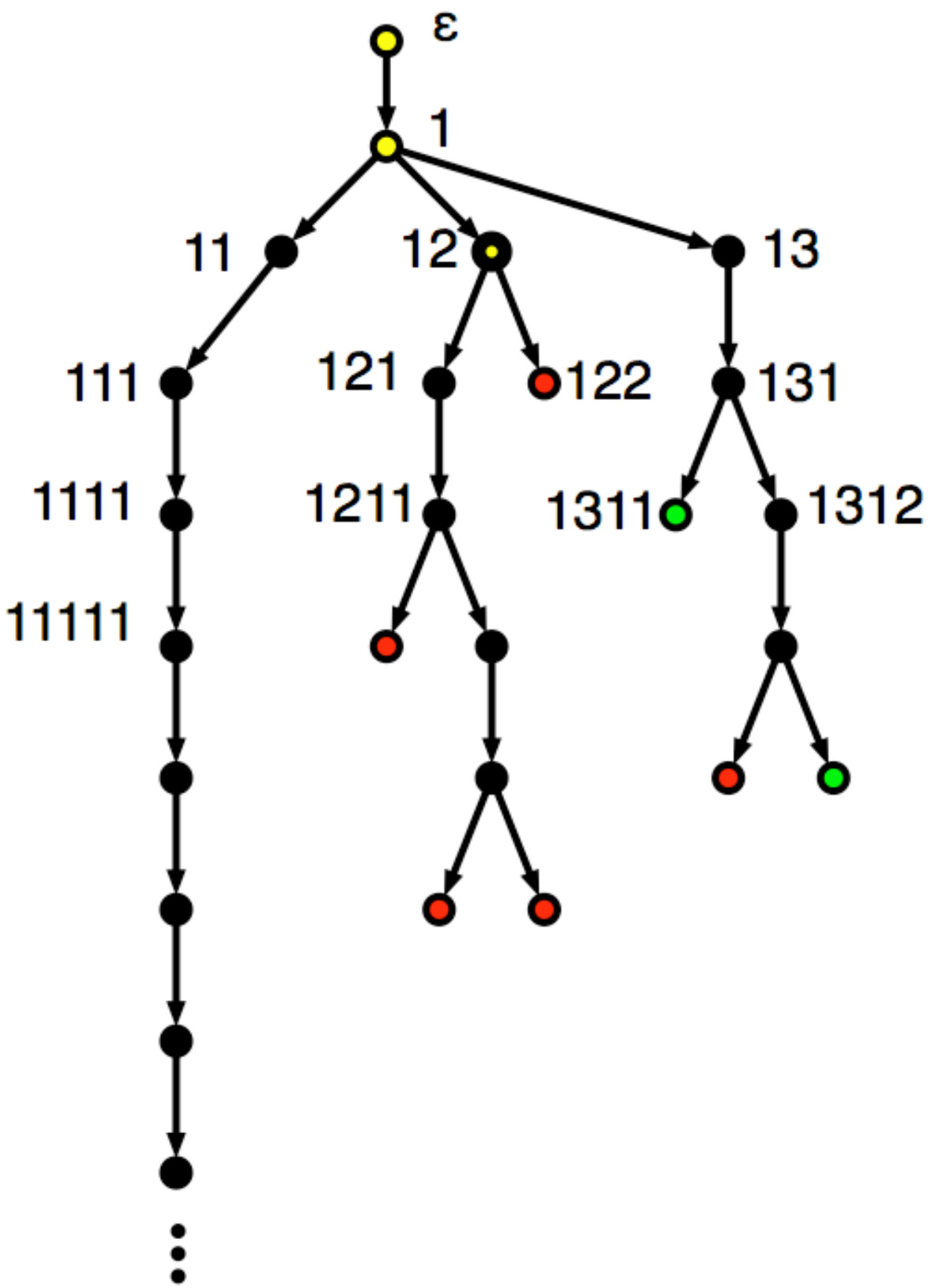

Eingabeband
0 1 0 1 1 0 _
Simulationsband
0 1 0 1 1 0 _
Adressband
1 2 _ _ _ _ _
D


ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111


Eingabeband
0 1 0 1 1 0 _
Simulationsband
1 1 0 1 1 0 _
Adressband
1 2 _ _ _ _ _
D


ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111


Eingabeband
0 1 0 1 1 0 _
Simulationsband
1 1 0 1 1 0 _
Adressband
1 2 _ _ _ _ _
D


ε
1
11
12
13
111
121
122
131
1111
1211
1311
1312
11111

Schlussfolgerungen

- **Korollar:**
  - Eine Sprache ist genau dann rekursiv aufzählbar, wenn sie es eine nichtdeterministische Turing-Maschine gibt, die jedes Wort der Sprache akzeptiert.
  - Eine Sprache ist genau dann rekursiv entscheidbar, wenn eine nichtdeterministische Turingmaschine sie entscheidet.

Berechenbarkeitstheorie

Die Church-Turing-These

Wiederholung: Turing-Maschine

- **Was ist eine Turing-Maschine:**
  - DFA + 1 Band
- **Zum ersten Mal:**
  - Turing-Maschinen können endlos rechnen (d.h. abstürzen)
- **Turing-Maschinen akzeptieren Sprachen,**
  - wenn sie jedes Wort der Sprache “erkennen”
  - wenn sie ansonsten irgendwas machen (außer akzeptieren)
    - d.h. z.B. verwerfen, unendlich lange rechnen, etc.
- **Turing-Maschinen entscheiden Sprachen,**
  - wenn sie auf jeder Eingabe halten und
  - akzeptieren, wenn das Wort in der Sprache ist und
  - verwerfen, wenn das Wort nicht in der Sprache ist.

Entscheidbarkeit und rekursive Aufzählbarkeit

- Eine Sprache L heißt **rekursiv aufzählbar**, falls es eine Turingmaschine M gibt, die L **akzeptiert**
  - Ausschließlich Ja-Antworten
  - Programmabsturz = wird als Nein interpretiert
- Eine Sprache L heißt **rekursiv entscheidbar**, falls es eine Turingmaschine M gibt, die L entscheidet
  - Ja oder Nein! (in einer endlichen Zeit)

Der Maschinenpark der Turingmaschinen

- **Keller-Automaten (PDA)**
  - NFA + Keller
- **1-Band-Turing-Maschinen (TM, DTM)**
  - DFA + Band
- **Mehr-Band-Turing-Maschinen (k-Tape-TM)**
  - DFA + Band + Band + Band
- **Nichtdeterministische Turing-Maschine (NTM)**
  - NFA + Band

Was heißt entscheidbar?

- **Genau dann wenn eine**
  - 1-Band-TM, 2-Band-TM, 3-Band-TM, ..., k-Band-TM
  - NTM
- **für jedes Wort einer Sprache in endlicher Zeit ausgibt, ob es in L ist oder nicht, dann ist die Sprache entscheidbar.**
- **Gilt auch für ein Programm in**
  - Java
  - C
  - C++
  - Pascal
  - Fortran

Hilberts 10. Problem

- **Rede auf dem internationalen Mathematiker-Kongreß, Paris 1900**
  - mit 23 (seiner Zeit aktuellen Problemen)
- **10. Problem**
  - *Eine Diophantische Gleichung mit irgend welchen Unbekannten und mit ganzen rationalen Zahlencoefficienten sei vorgelegt: man soll ein Verfahren angeben, nach welchem sich mittelst einer endlichen Anzahl von Operationen entscheiden löst, ob die Gleichung in ganzen rationalen Zahlen lösbar ist.*
- **Frage beantwortet durch**
  - Yuri Matiyasevich 1970
  - nach Vorarbeiten von Martin Davis und Julia Robinson

Diophantisches Polynom

- **Beispielprobleme**
  - Gibt es x,y aus den ganzen Zahlen **Z**, so dass
    - x + y - 3 = 0 gilt?
  - Gibt es x,y,z aus **Z** so dass gilt: $6x^3yz^2 + 3y^2 - x^3 - 10 = 0$
- **Lösung des 10. Hilbertschen Problem:**
  - Geht nicht!
  - Soll heißen: es gibt kein algorithmisches Verfahren, dass dieses Problem lösen kann.
- **Liegt das an unserem eingeschränkten Begriff der Berechenbarkeit?**
  - Gibt es mächtigere Programmiersprachen als die der Turing-Maschine, Java, C++, ...

Die Church-Turing-These

- **Nachdem man eine Reihe von verschiedenen diskreten Berechnungsmodellen als gleichwertig bewiesen hat, hat die Fachwelt die folgende These als allgemeingültig angesehen:**
  - Lambda-Kalkül von Alonzo Church (1936)
  - Turing-Maschine von Alan Turing (1936)
- **Church-Turing-These**
  - Der intuitive Begriff eines Algorithmus wird vollständig beschrieben durch die Algorithmen, welche Turing-Maschinen beschreiben können.
- **Tatsächlich sind alle Maschinen, die bisher von Menschenhand gebaut wurden, durch eine Turing-Maschine beschreibbar.**
- **Hoffnung für Gegenbeispiele:**
  - Analog-Computer
  - Quanten-Computer
  - Programmierer, die man in schwarze Löcher wirft

Hilberts 10. Problem

- **ist nicht entscheidbar, aber rekursiv aufzählbar**
- **Theorem**
  - Die Menge der diophantischen Gleichungen mit ganzzahligen Lösungen ist rekursiv aufzählbar.
- **Beweis: folgt gleich**
- **Theorem**
  - Hilberts 10. Problem ist nicht rekursiv entscheidbar.
- **Beweis**
  - sprengt den Rahmen dieser Vorlesung

Hilberts 10. Problem ist rekursiv aufzählbar

- **Theorem**
  - Die Menge der diophantische Gleichungen mit ganzzahligen Lösungen ist rekursiv aufzählbar.
- **Beweis**
  - Wir konstruieren eine Akzeptor-TM M
  - Gegeben eine diophantische Gleichung $G(x_1,x_2,..,x_m)$ mit den Variablen $x_1,x_2,..,x_m$
  - M = “Für b = 1,2,3,...

    Für alle $x_1,x_2,..,x_m \in \{-b,-b+1,...,-1,0,1,2,..., b\}$

    Falls $G(x_1,x_2,..,x_m) = 0$ akzeptiere”
  - Beweis der Korrektheit:
    - Falls für $y_1,y_2,..,y_m$: $G(y_1,y_2,..,y_m) = 0$
      - Dann wird für b= $\max\{|y_1|, |y_2|,.., |y_m|\}$ die Kombination $y_1,y_2,..,y_m$ in $x_1,x_2,..,x_m$ eingesetzt
    - Falls für alle $x_1,x_2,..,x_m$: $G(x_1,x_2,..,x_m) \neq 0$,
      - akzeptiert M niemals.

Berechenbarkeitstheoire

REG und Entscheidbare Sprachen

Entscheidbare Reguläre Sprachprobleme

- **Das Wortproblem regulärer Sprachen**
  - Gegeben:
    - Ein DFA B
    - ein Wort w
  - Gesucht:
    - Akzeptiert B das Wort w?
- **Beschrieben durch die Sprache:**

$$\mathrm{A_{DFA}} = \{\langle \mathrm{B}, \mathrm{w} \rangle \mid B \text{ ist ein DFA der eine Eingabe } x \text{ akzeptiert}\}$$

  - DFA ist geeignet kodiert
  - Klammern stehen für eine geeignete Tupelkodierung
- **Das Problem einer Sprache L wird beschrieben durch die charakteristische Funktion der Sprache L**

$$\chi_{\mathrm{L}}(\mathrm{w}) = \begin{cases} 1, & w \in L \\ 0, & w \notin L \end{cases}$$

Das Wortproblem regulärer Sprachen

- **Theorem**
  - Das Wortproblem der regulären Sprachen ist entscheidbar.
- **Beweis**
  - Konstruiere eine Turing-Masche M, die folgendes berechnet:
  - M = “Für Eingabe <B,w> mit DFA B und Wort w:
    1. Simuliere B auf Eingabe w
    2. Falls die Simulation akzeptiert, dann akzeptiere
       Sonst verwerfe”
  - Implementationsdetails:
    - Übergangsfunktion ist geeignet kodiert auf einem Band
    - Zustand ist auf einem separaten Band
    - Eingabe auf einem dritten Band
    - Suche nächsten Übergang in der Kodierung des DFAs

Das Leerheitsproblem regulärer Sprachen

- **Das Leerheitsproblem regulärer Sprachen:**
  - Gegeben:
    - Ein DFA A
  - Entscheide:
    - Ist L(A) = ∅
- **Die zugehörige Sprache wird beschreiben durch:**

$$\mathsf{E}_{\mathrm{DFA}} = \{\langle \mathsf{A} \rangle \mid A \text{ ist ein DFA und } L(A) = \emptyset \}$$

- **Theorem**
  - Das Leerheitsproblem regulärer Sprachen ist entscheidbar

Das Leerheitsproblem regulärer Sprachen

- **Theorem**
  - Das Leerheitsproblem regulärer Sprachen ist entscheidbar.

    $\mathrm{E}_{\mathrm{DFA}} = \{\langle \mathrm{A} \rangle \mid A \text{ ist ein DFA und } L(A) = \emptyset \}$
- **Beweis**
  - Ein DFA akzeptiert mindestens ein Wort, wenn der DFA mindestens einen vom Startzustand erreichbaren Zustand besitzt
  - T = “Auf Eingabe A, wobei A ein DFA ist
    - Markiere Startzustand A
    - Wiederhole
    - Markiere jeden Folgezustand eines markierten Zustands
    - bis kein neuer Folgezustand in A markiert wurde
    - Falls kein akzeptierender Zustand markiert wurde
      - Akzeptiere
      - sonst: verwerfe”

Das Äquivalenzproblem regulärer Sprachen

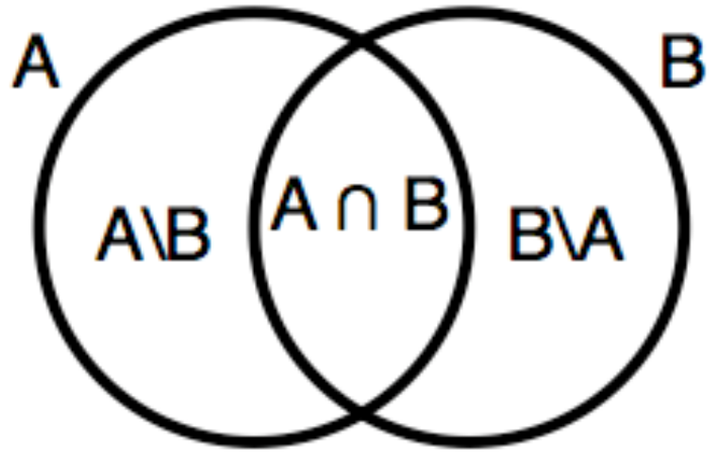
- **Das Äquivalenzproblem regulärer Sprachen:**
  - Gegeben:
    - Zwei DFAs A, B
  - Entscheide:
    - Ist L(A) = L(B)
- **Die zugehörige Sprache wird beschrieben durch:**

$$\mathrm{EQ}_{\mathrm{DFA}} = \{\langle \mathrm{A}, \mathrm{B} \rangle \mid A, B \text{ sind DFAs und } L(A) = L(B) \}$$

- **Theorem**
  - Das Äquivalenzproblem regulärer Sprachen ist entscheidbar

- **Beobachtung für zwei Mengen A,B:**

$$A = B \Leftrightarrow (A \setminus B) \cup (B \setminus A) = \emptyset$$



Das Äquivalenzproblem regulärer Sprachen

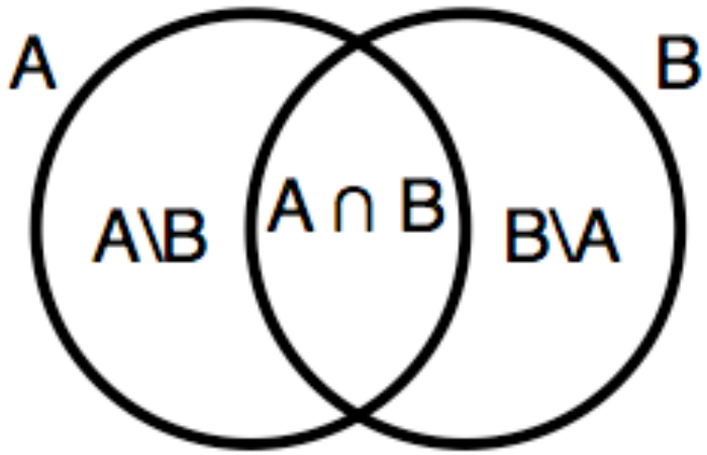
- **Theorem**
  - Das Äquivalenzproblem regulärer Sprachen ist entscheidbar

$$\mathrm{EQ}_{\mathrm{DFA}} = \{\langle A, B\rangle \mid A, B \text{ sind DFAs und } L(A) = L(B)\}$$

- **Beobachtung für zwei Mengen A,B:**

$$\mathbf{A = B} \Leftrightarrow \mathbf{(A \setminus B) \cup (B \setminus A)} = \emptyset$$

- **Beweis:**
  - Konstruiere DFA X für die Sprache $(A \setminus B) \cup (B \setminus A)$
  - Durch Mehrfachanwendung des kartesischen Produkts
  - Teste ob $L(X) = \emptyset$
    - mit Hilfe der Turing-Maschine für das Leerheitsproblem



Berechenbarkeitstheorie

CFG und Unentscheidbare Sprachen

Das Wortproblem der kontextfreien Sprachen

- **Wortproblem der kontextfreien Sprachen:**
  - Gegeben:
    - Kontextfreie Grammatik G in geeigneter Kodierung
    - Wort w
  - Gesucht:
    - Ist $w \in L(G)$?
- **Theorem**
  - Das Wortproblem der kontextfreien Sprachen ist entscheidbar.
- **Beweis**
  - Wandle Grammatik in Chomsky-Normalform um
  - Führe Cocke-Younger-Kasami-Algorithmus für eine gegebene kontextfreie Grammatik und ein gegebenes Wort durch
  - Akzeptiere falls das Wort vom Startsymbol abgeleitet werden kann, ansonsten verwerfe.

Das Leerheitsproblem der kontextfreien Sprachen

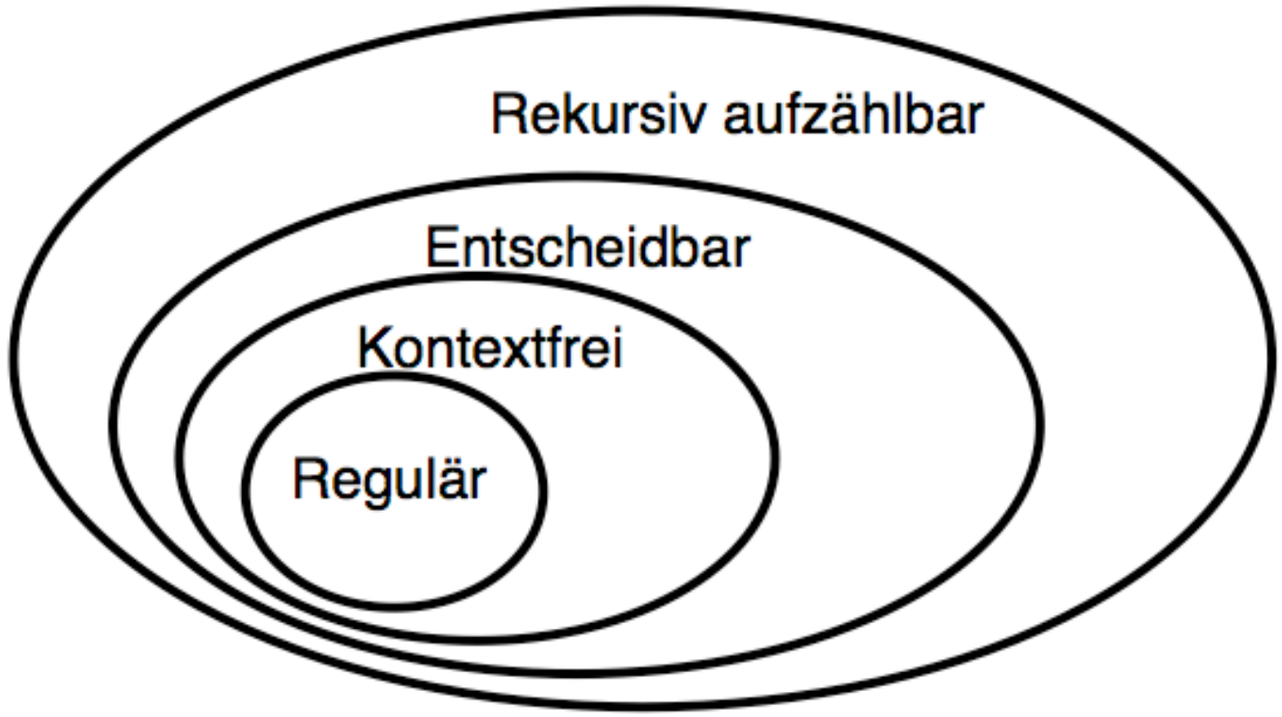
- **Das Leerheitsproblem kontextfreier Sprachen:**
  - Gegeben:
    - Eine kontextfreie Grammatik G
  - Entscheide:
    - Ist $L(G) = \emptyset$?
- **Theorem**
  - Das Leerheitsproblem kontextfreier Sprachen ist entscheidbar.
- **Beweis**
  - Wandle G in Chomsky-Normalform um
  - Markiere alle Terminalsymbole
  - Solange neue Markierungen erscheinen
    - Markiere alle Nichtterminale, die Regeln besitzen deren rechte Seite vollständig markiert ist
  - Falls das Startsymbol markiert ist, verwerfe
  - Ansonsten akzeptiere

Das Äquivalenzproblem der kontextfreien Sprachen

- **Das Äquivalenzproblem kontextfreier Sprachen:**
  - Gegeben:
    - Kontextfreie Grammatiken G, G’
  - Entscheide:
    - Ist L(G) = L(G’)
- **Problem:**
  - Kontextfreie Sprachen sind **nicht** abgeschlossen
    - unter Komplement
    - unter Schnittoperationen
  - Beweis des Äquivalenzproblem der regulären Sprachen ist nicht übertragbar
- **Theorem**
  - Das Äquivalenzproblem kontextfreier Sprachen ist **nicht** entscheidbar.
- **(ohne Beweis)**

Beziehungen zwischen den Sprachen

- **Jede reguläre Sprache ist eine kontextfreie Sprache.**
- **Jede kontextfreie Sprache ist eine entscheidbare Sprache.**
  - folgt aus der Entscheidbarkeit des Wortproblems der kontextfreien Sprachen.
- **Jede entscheidbare Sprache ist eine rekursiv aufzählbare Sprache.**



Ende