Formale Sprachen und Komplexitätstheorie

WS 2019/20

Robert Elsässer

Organisatorisches

Vorlesung:

- Di 11:15 – 12:45 T.01

Proseminar:

- Di 13:00 – 13:45 T.01
- Mi 14:15 – 15:00 T.03
- Beginn: nächste Woche

Organisatorisches

Heimübungen:

- Jede Woche ein Übungsblatt (dienstags)
- 1. Blatt diese Woche
- Abgabe: Dienstag bis 11:00 Uhr
- Erfolgreicher Abschluss (Gruppen zu 2-4 Personen)
  - mind. 50% der erreichbaren Punkte
  - mind. einmal korrekt Vorrechnen
  - Online ankreuzen
- Erste bewertete Aufgabe: übernächste Woche
- Vorstellung *einer* Musterlösung im Proseminar

Organisatorisches

Proseminargruppen:

- PLUSonline
  - erreichbar über https://online.uni-salzburg.at/plus_online/webnav.ini
  - ITS-Login und Passwort
  - Bis zu 25 Teilnehmer
  - eine Gruppe
  - Übungsaufgaben und Vorlesungsfolien über die Webseite der Vorlesung: http://fl.cosy.sbg.ac.at
- Abmeldung bis zum 25.10.2019, 23:55 Uhr
  - Nach diesem Zeitpunkt ist keine Abmeldung mehr möglich und es folgt eine Bewertung der Leistungen am Ende des Semesters

Organisatorisches

Klausur:

- Eine Korrelation mit den PS-Aufgaben ist zu erwarten
- Es gab in der Vergangenheit einen direkten Zusammenhang zwischen PS-Teilnahme bzw. -abgabe und gutem Abschneiden bei Klausuren

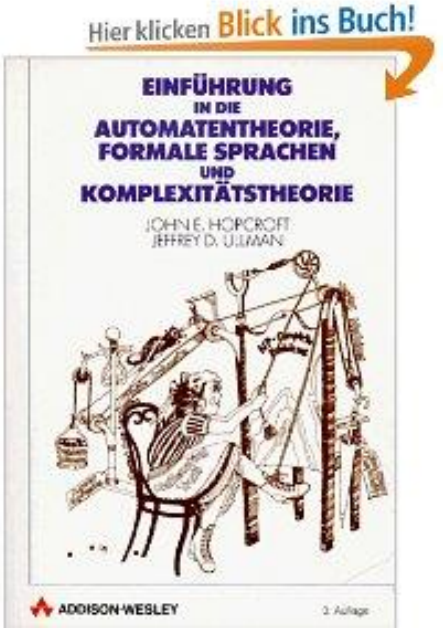
Sprechzeiten:

- Di 10:00 – 11:00
- (Raum 2.23, Jakob-Haringer-Straße 2)

Organisatorisches

Literatur:

- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, 3. Auflage, Pearson Studium, 2011
- Die Vorlesungsfolien wurden unter Verwendung der Folien und des Skriptes der Vorlesung *„Einführung in die Berechenbarkeit, Komplexität und Formale Sprachen“* von Prof. Dr. Johannes Blömer, Prof. Dr. Friedhelm Meyer auf der Heide bzw. Prof. Dr. Christian Scheideler erstellt. Die meisten Bilder, Definitionen und Beschreibungen wurden aus den oben genannten Unterlagen übernommen.
- Die Folien wurden von Eva Lugstein (ehem. Studienassistentin am FB Cowi) überarbeitet.



Quelle: amazon.de

Motivation

- Was lässt sich mit dem Computer lösen?
  Wie effizient lassen sich einzelne Probleme lösen?
- Grenzen und Möglichkeiten eines Rechners.
- Eine geeignete Formalisierung ist Voraussetzung für eine systematische Lösung.
- Als Ausdrucksmittel muss man passende Kalküle und Notationen anwenden können.

Ziele

- Einen Überblick über grundlegende Formalisierungsmethoden zu bekommen
- Die für die Methoden typische Techniken zu erlernen
- Techniken an typischen Beispielen anzuwenden

Insgesamt soll erlernt werden:

- Aufgaben präzise zu formalisieren und zu analysieren
- berechenbare von unberechenbaren Problemen zu unterscheiden
- festzustellen, ob ein Problem effiziente Lösungen haben kann

Durchführung

Zu jedem Bereich soll(en):

- mit einigen typischen Beispielen motivierend hineingeführt werden
- der konzeptionelle Kern der Methode vorgestellt werden
- Anwendungstechniken an Beispielen gezeigt und in den Proseminaren erfahren werden
- an einem durchgehenden Beispiel größere Zusammenhänge gelernt werden

Inhaltsangabe

- Einleitung, Motivation
- Turingmaschinen
- Arbeitstechniken

Einführung

- Unentscheidbare Probleme
- Das Halteproblem
- Reduktionen

Berechenbarkeit

- Zeitkomplexität
- Die Klassen P und NP
- NP-Vollständigkeit
- NP-vollständige Probleme

Komplexität

- Formale Sprachen und Automaten
- Kellerautomaten und kontextfreie Sprachen
- Kontextsensitive Sprachen

Formale Sprachen

1. Einführung

$\mathrm{ALG}(n)$

```
1 for t = 1 to ∞
2     for x = 1 to t
3         for y = 1 to t
4             for z = 1 to t
5                 if x^n + y^n = z^n
6                     return (x, y, z)
```

$\mathrm{ALG}(n)$ hält bei Eingabe n genau dann, wenn $x^n + y^n = z^n$ für irgendwelche natürlichen Zahlen x, y, z.

Könnte man das Halteproblem lösen, so würde man den großen Satz von Fermat beweisen oder widerlegen.

1. Einführung

Gibt es einen Algorithmus HALTE, der

- als Eingabe einen beliebigen Algorithmus ALG und eine Eingabe w für ALG erhält und
- entscheidet, ob ALG bei Eingabe w hält?

Satz von Turing:

Einen solchen Algorithmus kann es nicht geben.

1. Einführung

- Was sind die wesentlichen Möglichkeiten und Grenzen von Computern?

- **Berechenbarkeit und Komplexität:**
  - Was ist ein Problem?
    - Berechnung einer Funktion, Optimierungsproblem, Entscheiden einer Sprache
    - Multiplikation zweier Matrizen, Kürzeste Wege finden, usw…
  - Wie modelliert man einen Computer?
    - Registermaschinen (RAM), λ-Kalkulus, μ-Rekursion, Turingmaschinen, usw…

1. Einführung

- **Problem des Handlungsreisenden:**
  - Wien, Krems, St. Pölten, Wiener Neustadt, Mürzzuschlag, Graz, Wolfsberg, Klagenfurt, Villach, Spital, Lienz, Brenner, Innsbruck, Kitzbühel, Bischofshofen, Hallein, Salzburg, Braunau, Ried, Wels, Linz, Enns, Amstetten, Zwettl, Stockerau

1. Einführung

- Was sind die wesentlichen Möglichkeiten und Grenzen von Computern?

- **Berechenbarkeit und Komplexität:**
  - Was ist ein Problem?
    - Berechnung einer Funktion, Optimierungsproblem, Entscheiden einer Sprache
    - Multiplikation zweier Matrizen, Kürzeste Wege finden, usw…
  - Wie modelliert man einen Computer?
    - Registermaschinen (RAM), λ-Kalkulus, μ-Rekursion, Turingmaschinen, usw…

1. Einführung

- Alle sinnvollen Rechenmodelle liefern aus unserer Sicht die gleichen Ergebnisse
  - **Churchsche These**

- Es gibt Probleme, die algorithmisch nicht gelöst werden können (z.B. das Halteproblem)
  - **Berechenbarkeit**

- Manche Problem können zwar algorithmisch, aber nicht effizient gelöst werden (z.B. das Problem des Handlungsreisenden)
  - **Komplexität**

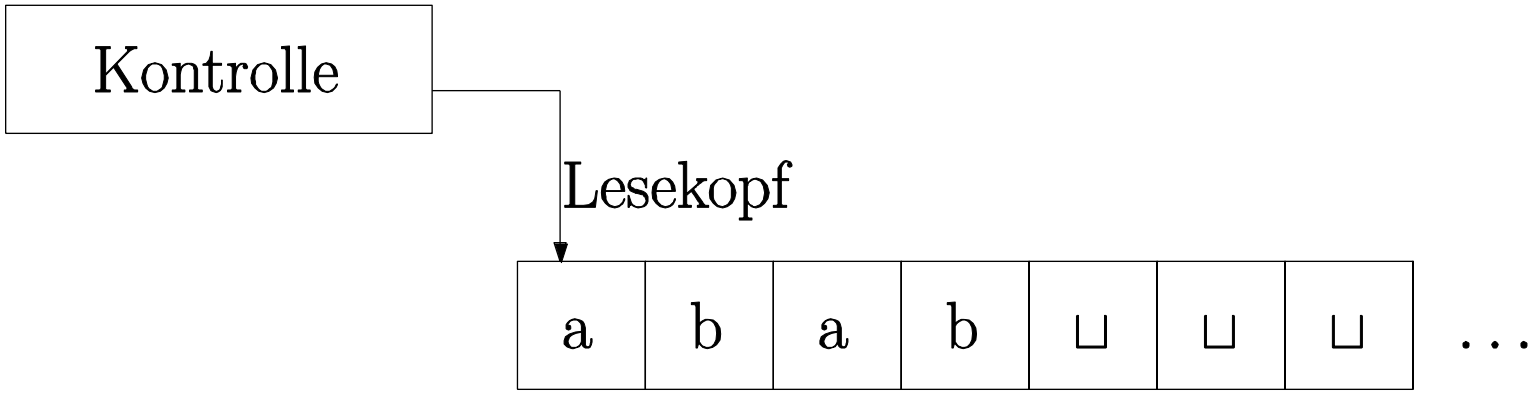
1. Einführung

- Wie werden Sprachen beschrieben?
  - Mit Hilfe von Grammatiken – syntaktisch korrekte Java-Programme
- Wie analysiert man Worte?
  - Mit Hilfe der Syntaxanalyse
- Einfachster Fall: Ist x in L? – Automaten
  - Ist ein Programm syntaktisch korrekt?
- Verschiedene mächtige Grammatiken und Automatentypen:
  - Reguläre Ausdrücke, kontextfreie Grammatiken, kontextsensitive Grammatiken, …

1. Einführung

- **Turingmaschine**
  - Arbeitet auf unbeschränktem Band
  - Eingabe steht zu Beginn am Anfang des Bands
  - Auf dem Rest des Bandes steht t (Blank)
  - Position auf dem Band wird durch den sog. *Lesekopf* beschrieben

Turingmaschine



- Der jeweils nächste Rechenschritt ist eindeutig festgelegt durch den aktuellen Zustand und das aktuell gelesene Zeichen.
- Der Rechenschritt überschreibt das aktuelle Zeichen, bewegt den Kopf nach rechts oder nach links und verändert den Zustand.

Alan Turing

Quelle: rutherfordjournal.org

- Studium in Cambridge
- Entschlüsselung von Enigma-Verschlüsselungen
- Professor in Manchester
- Nach Alan Turing wird der renommierteste Preis in der Informatik benannt
- „The Imitation Game“

https://en.wikipedia.org/wiki/Alan_Turing

1. Einführung

- Teste, ob ein Wort in der folgenden Sprache liegt:
  - $L = \{w\#w \mid w \text{ in } \{0,1\}^*\}$

- **Vorgehensweise:**
  - Von links nach rechts über das Wort laufen
  - Erstes Zeichen links merken und markieren
  - Erstes Zeichen rechts von # vergleichen und markieren
  - Für alle Zeichen wiederholen bis Blank erreicht
  - Rechts dürfen dann nur noch Blanks folgen
  - Falls Zeichen an einer Stelle nicht übereinstimmen ablehnen, sonst am Ende akzeptieren

1. Einführung

$0\ 1\ 1\ 0\ 0\ 0\ \#\ 0\ 1\ 1\ 0\ 0\ 0\ t$ ...

$x\ 1\ 1\ 0\ 0\ 0\ \#\ 0\ 1\ 1\ 0\ 0\ 0\ t$ ...

$x\ 1\ 1\ 0\ 0\ 0\ \#\ x\ 1\ 1\ 0\ 0\ 0\ t$ ...

$x\ 1\ 1\ 0\ 0\ 0\ \#\ x\ 1\ 1\ 0\ 0\ 0\ t$ ...

$x\ x\ 1\ 0\ 0\ 0\ \#\ x\ 1\ 1\ 0\ 0\ 0\ t$ ...

$x\ x\ x\ x\ x\ x\ \#\ x\ x\ x\ x\ x\ x\ t$ ... accept

1. Einführung

Definition

Eine *(deterministische 1-Band) Turingmaschine* (DTM) 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 und es gilt:

- Σ ist Teilmenge von Γ
- t in $\Gamma \setminus \Sigma$ ist das *Blanksymbol* (auch $\sqcup$)
- Q ist die *Zustandsmenge*
- Σ ist das *Eingabealphabet*
- Γ ist 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
- $\delta: Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \to Q \times \Gamma \times \{L, R\}$ ist die (partielle) *Übergangsfunktion*. Sie ist für kein Argument aus $\{q_{accept}, q_{reject}\} \times \Gamma$ definiert.

1. Einführung

- 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 q_{accept} oder q_{reject} erreicht wird, ist die Bearbeitung beendet

1. Einführung

Momentaufnahme einer Turingmaschine:

- Bei Bandinschrift uv (dabei beginnt u am linken Ende des Bandes und hinter v stehen nur Blanks)
- Zustand q
- Kopf auf erstem Zeichen von v

Konfiguration $C = uqv$

1. Einführung

- 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 $a, b, c \text{ in } \Gamma, u, v \text{ in } \Gamma^*$ und Zustände q_i, q_j gegeben
- Wir sagen:
  - uaq_ibv führt zu uq_jacv, falls $\delta(q_i, b) = (q_j, c, L)$ und
  - uaq_ibv führt zu $uacq_jv$, falls $\delta(q_i, b) = (q_j, c, R)$

1. Einführung

- Startkonfiguration:
  - $q_0 w$, 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

1. Einführung

Definition

Eine Turingmaschine M akzeptiert eine Eingabe w, falls es eine Folge von Konfigurationen $C_1, C_2, \ldots, C_k$ gibt, sodass

1. C_1 ist die Startkonfiguration von M bei Eingabe w
2. C_i führt zu C_{i+1}
3. 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 haltenden Konfiguration C_k resultiert.

1. Einführung

Definition

- 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.*

1. Einführung

Gesucht: Turingmaschine, die $L = \{0^{2^n} \mid n \geq 0\}$ entscheidet

Arbeitsweise:

1. Gehe von links nach rechts über die Eingabe und ersetze jede zweite 0 durch x
2. Wenn nur eine 0 auf dem Band ist, akzeptiere
3. Falls die Anzahl der 0en ungerade ist, lehne ab
4. Bewege den Kopf zurück an das linke Ende

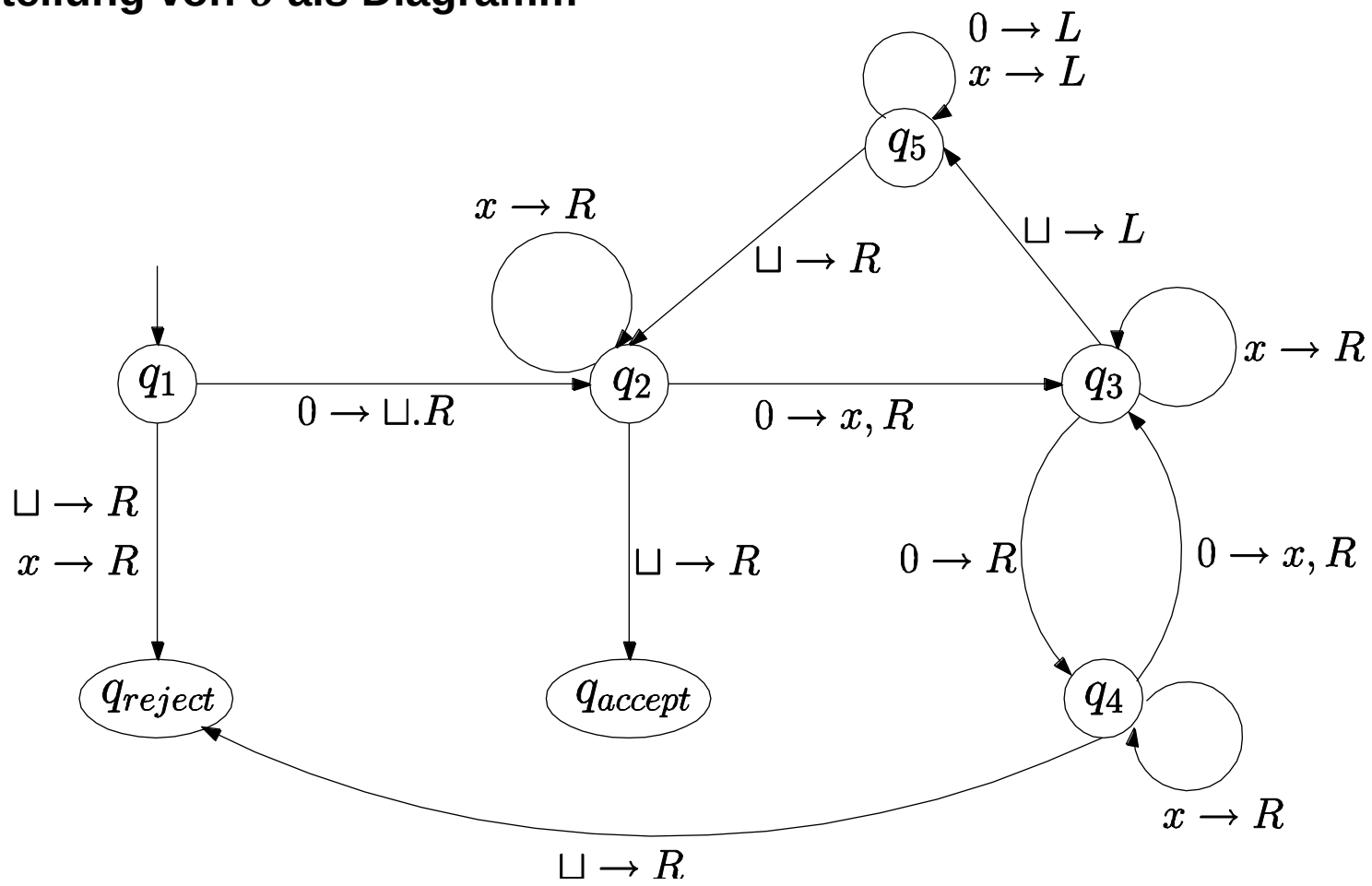
1. Einführung

Definition der Turingmaschine

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, t\}$
- q_1: Startzustand
- q_{accept}: Akzeptierender Endzustand
- q_{reject}: Ablehnender Endzustand
- δ: Übergangsfunktion

1. Einführung

Darstellung von δ als Diagramm

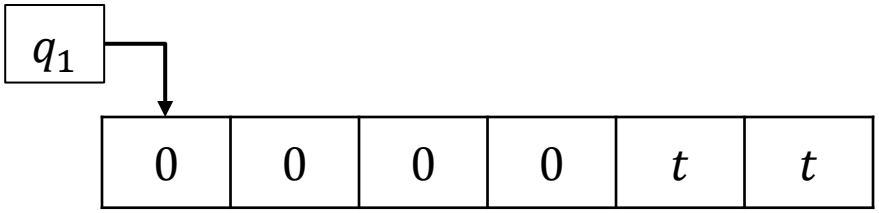


1. Einführung

Darstellung von δ als Tabelle

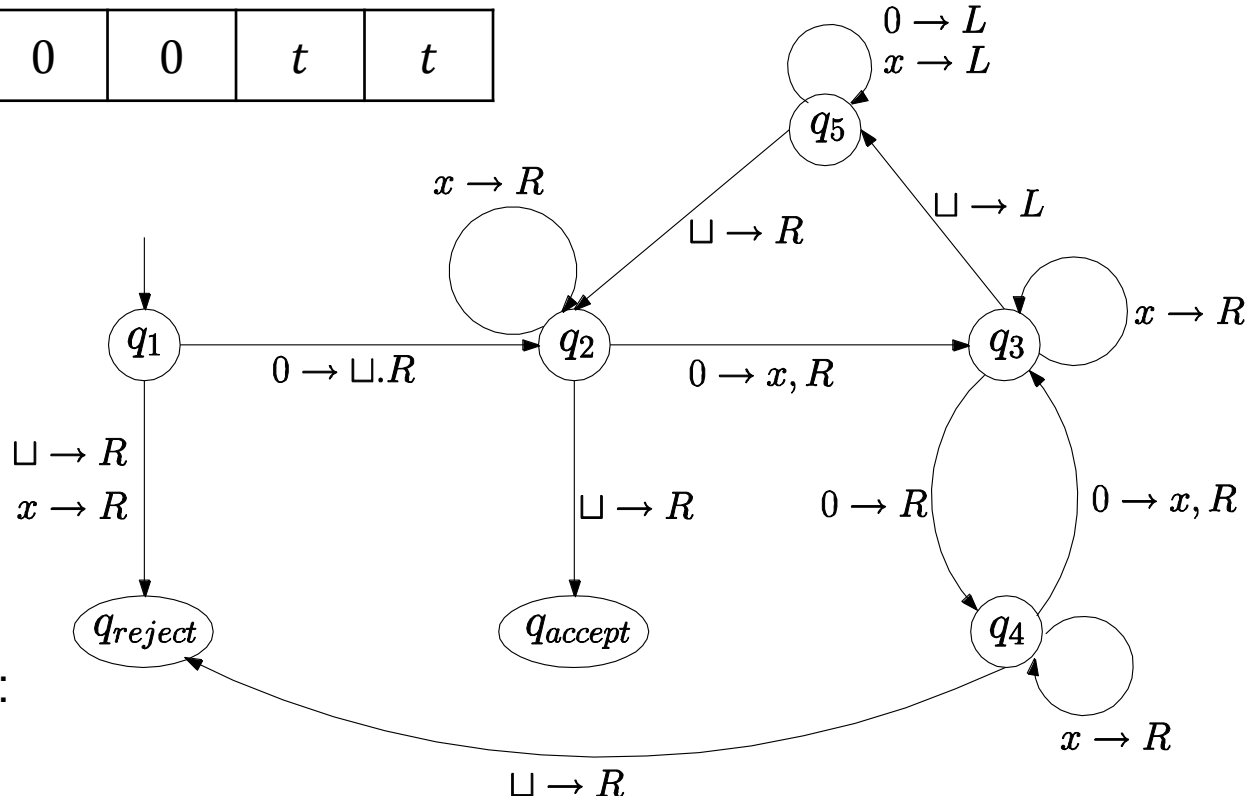
δ	0	x	t
q_1	(q_2, t, R)	(q_{reject}, x, R)	(q_{reject}, t, R)
q_2	(q_3, x, R)	(q_2, x, R)	(q_{accept}, t, R)
q_3	$(q_4, 0, R)$	(q_3, x, R)	(q_5, t, L)
q_4	(q_3, x, R)	(q_4, x, R)	(q_{reject}, t, R)
q_5	$(q_5, 0, L)$	(q_5, x, L)	(q_2, t, R)

1. Einführung

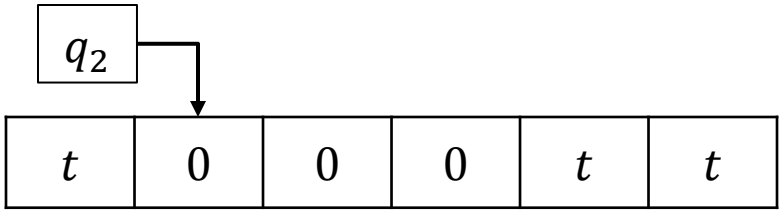


- Beispiel für das Wort: 0000

- Startkonfiguration: $q_1 0000$

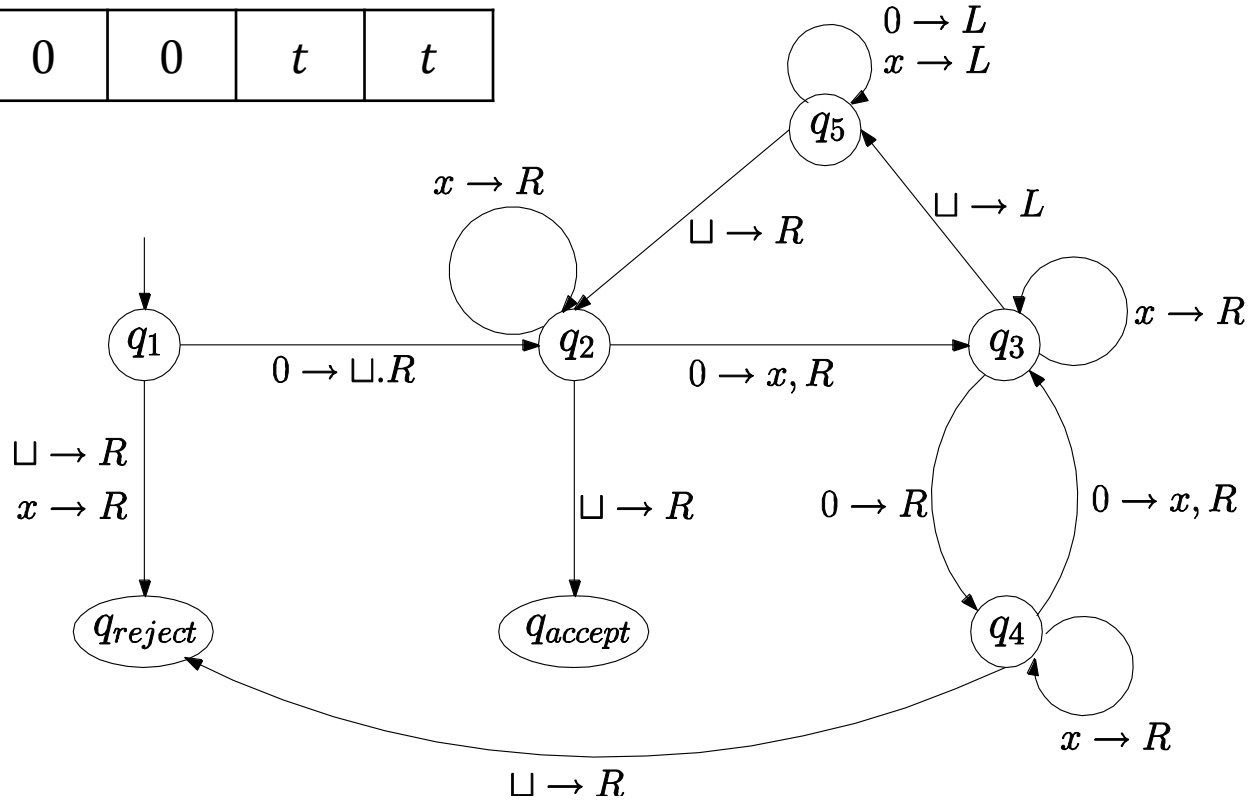


1. Einführung

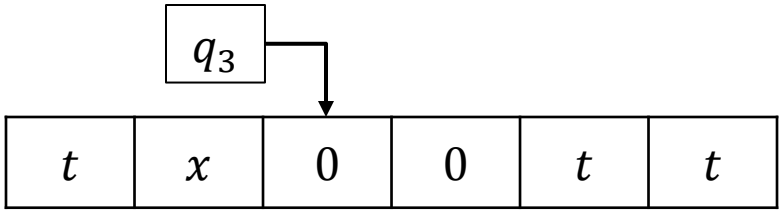


- Beispiel für das Wort: 0000

- Konfiguration: tq_2000

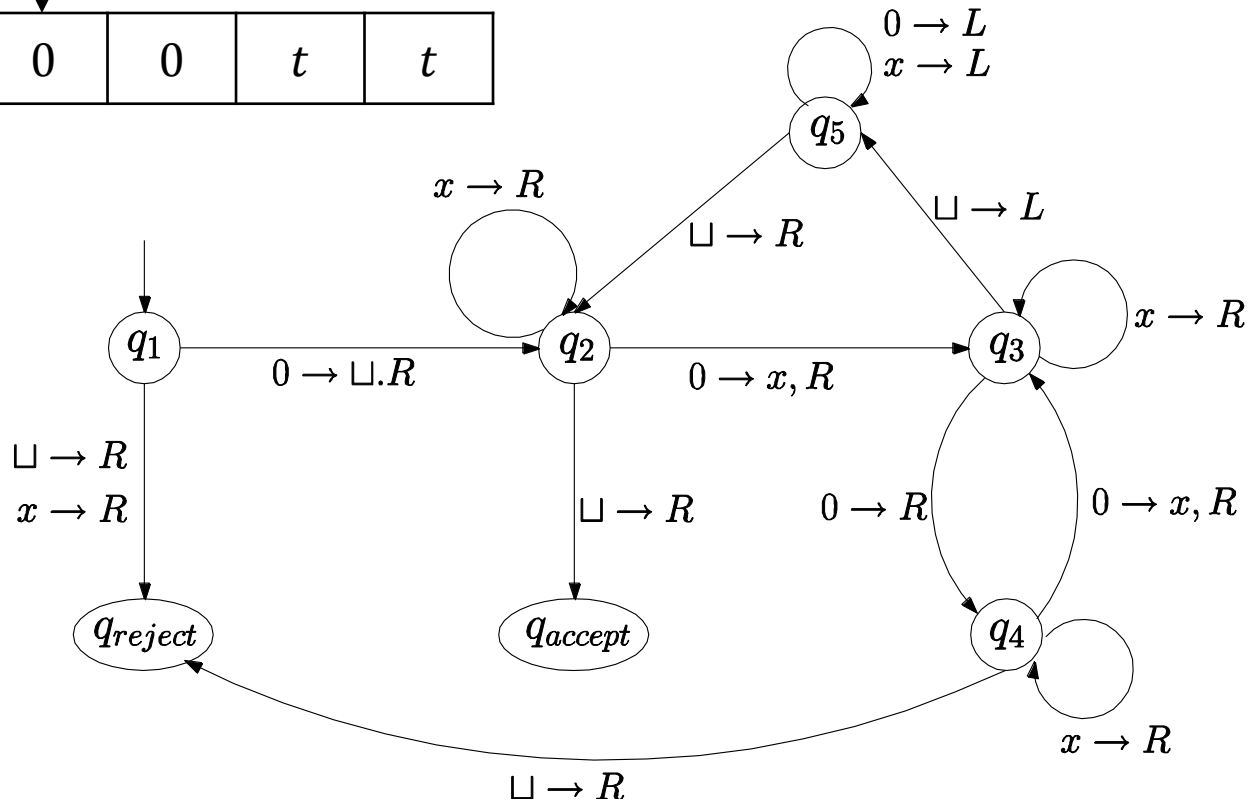


1. Einführung

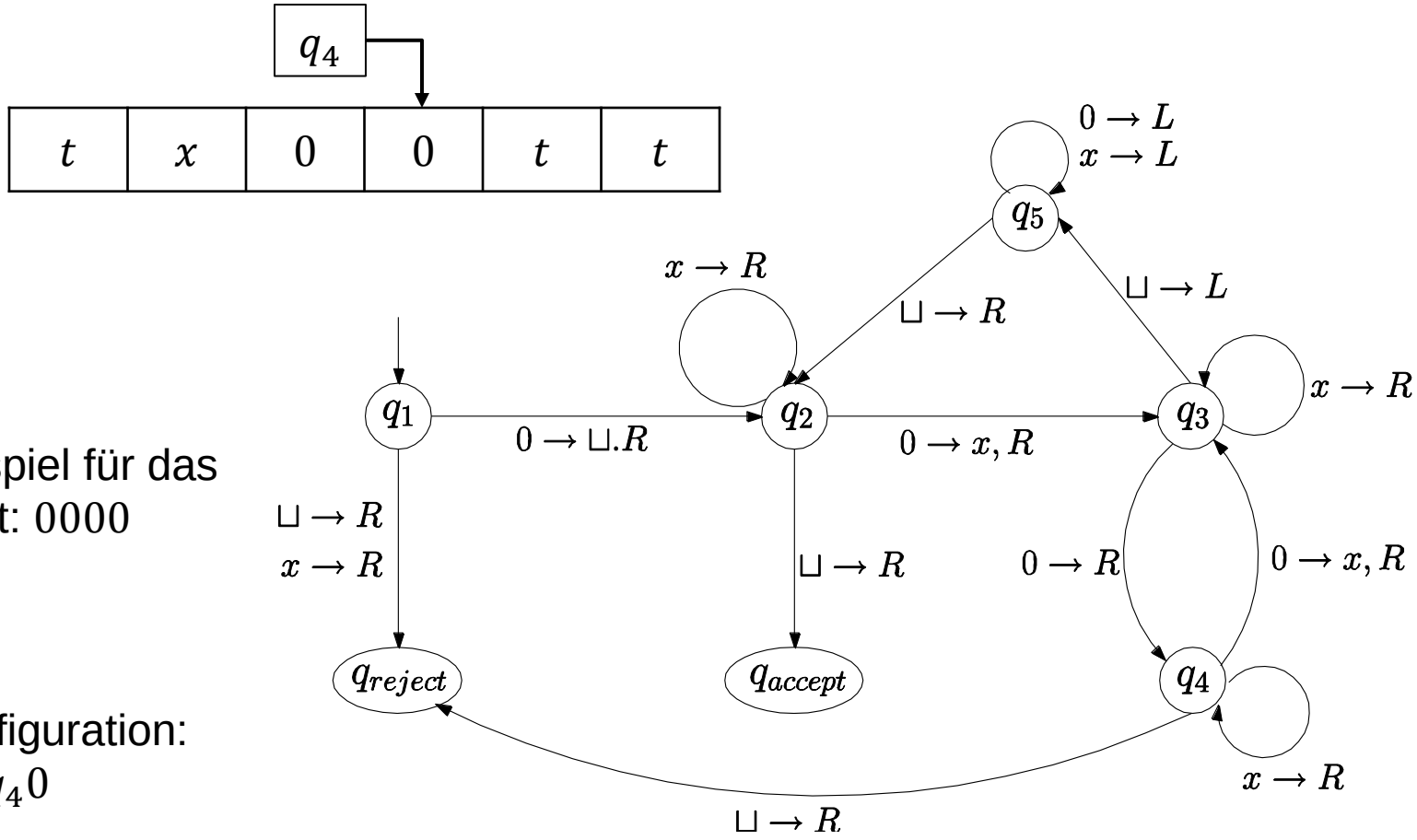


- Beispiel für das Wort: 0000

- Konfiguration: txq_300



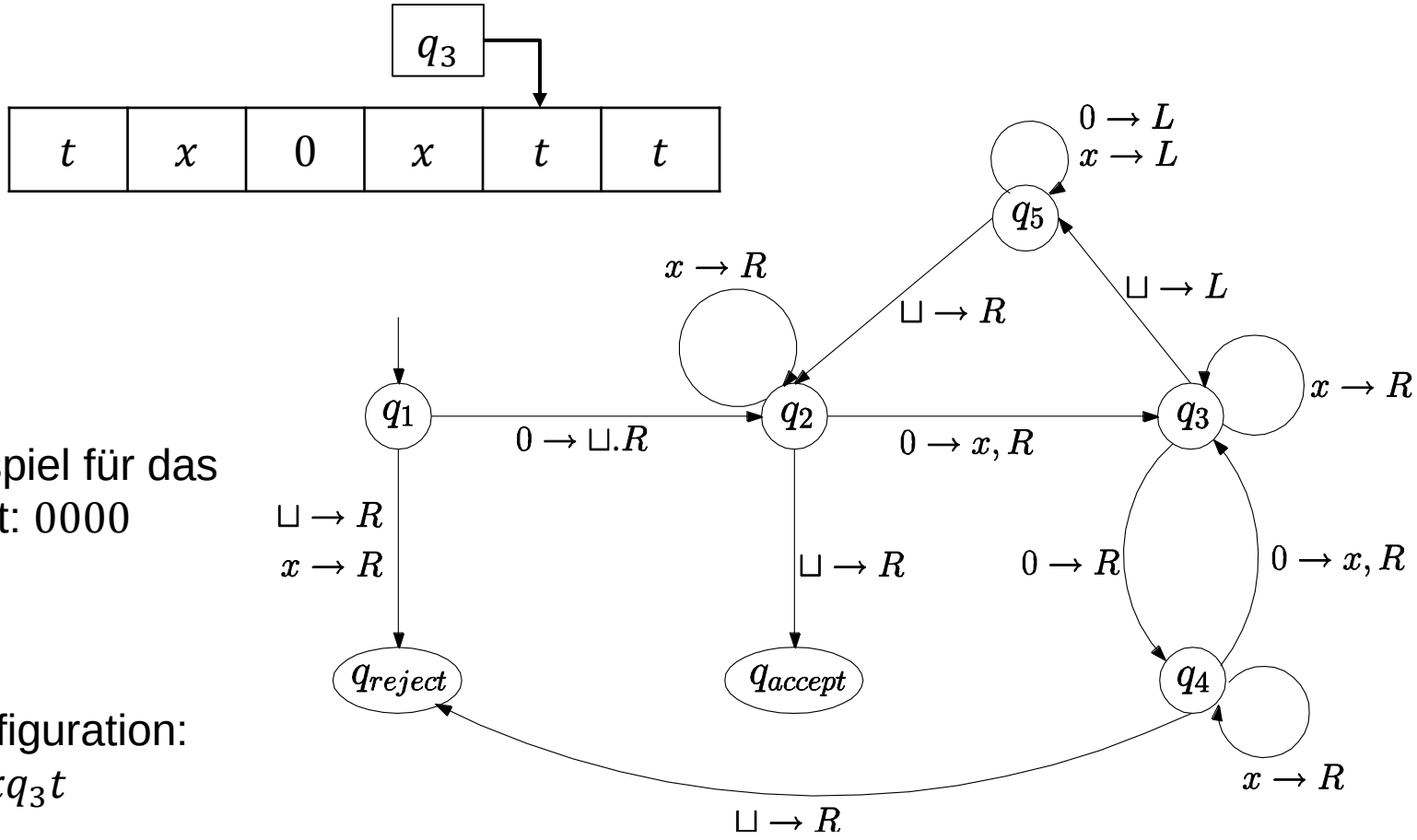
1. Einführung



- Beispiel für das Wort: 0000

- Konfiguration: $tx0q_40$

1. Einführung

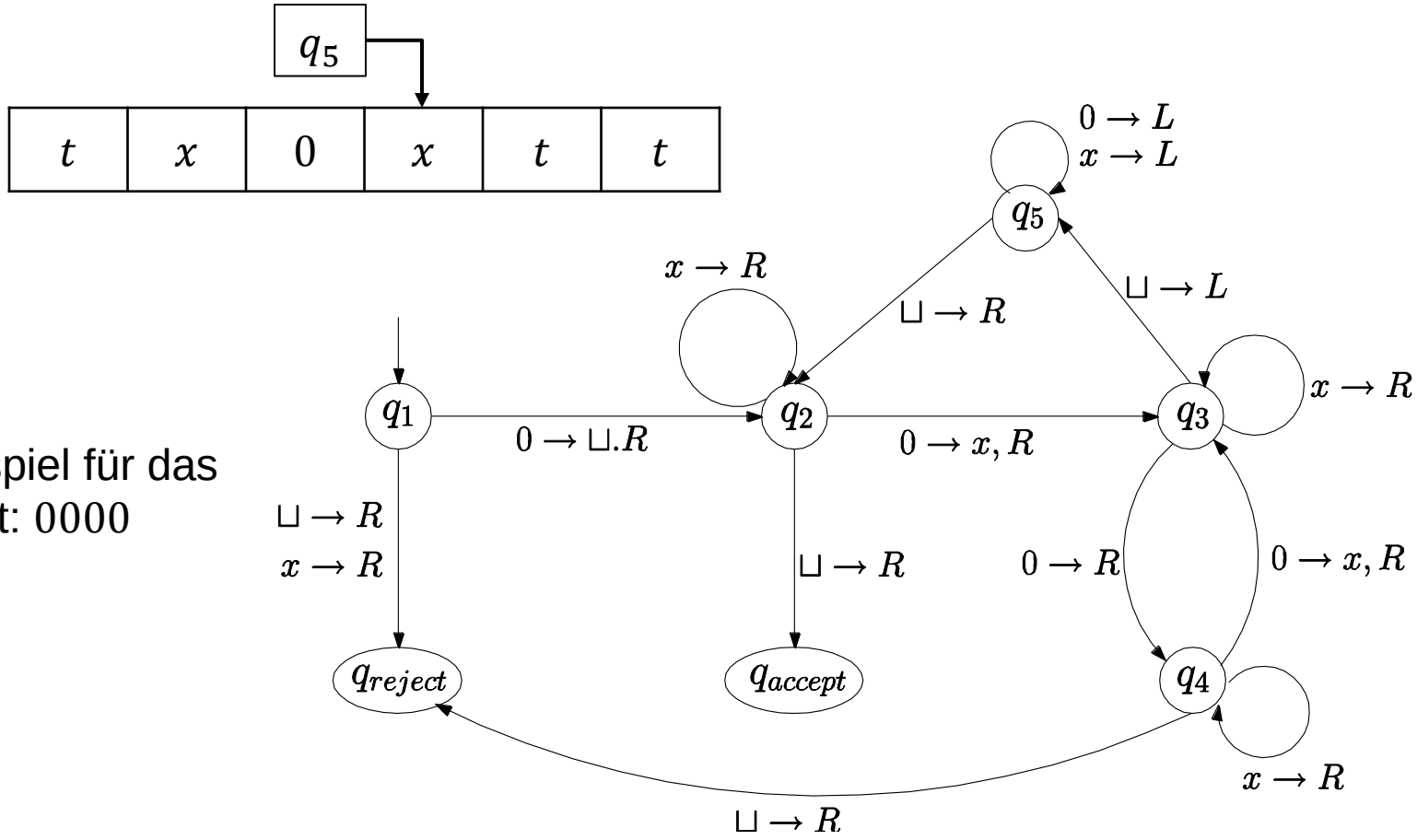


- Beispiel für das Wort: 0000

- Konfiguration: $tx0xq_3t$

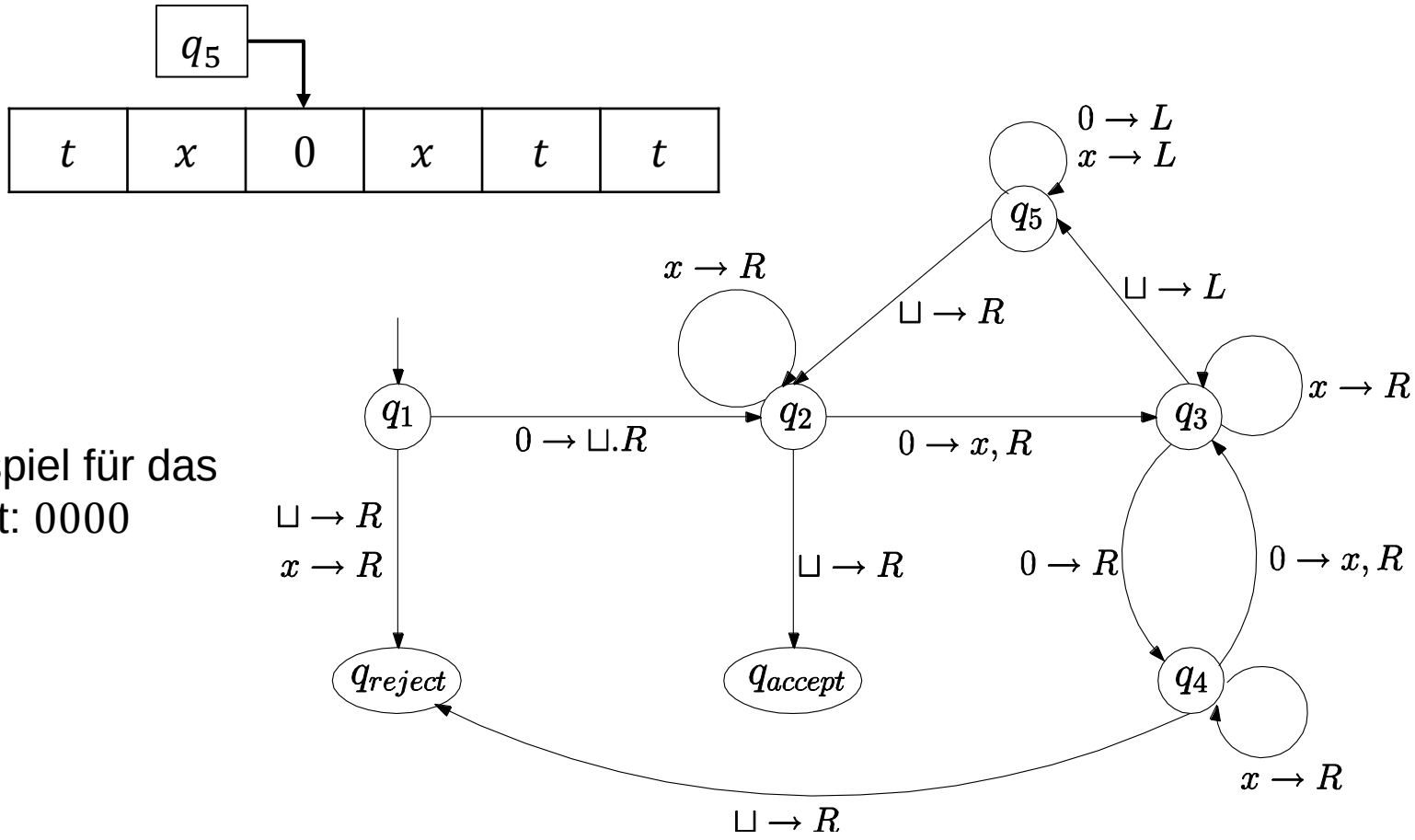
1. Einführung

- Beispiel für das Wort: 0000



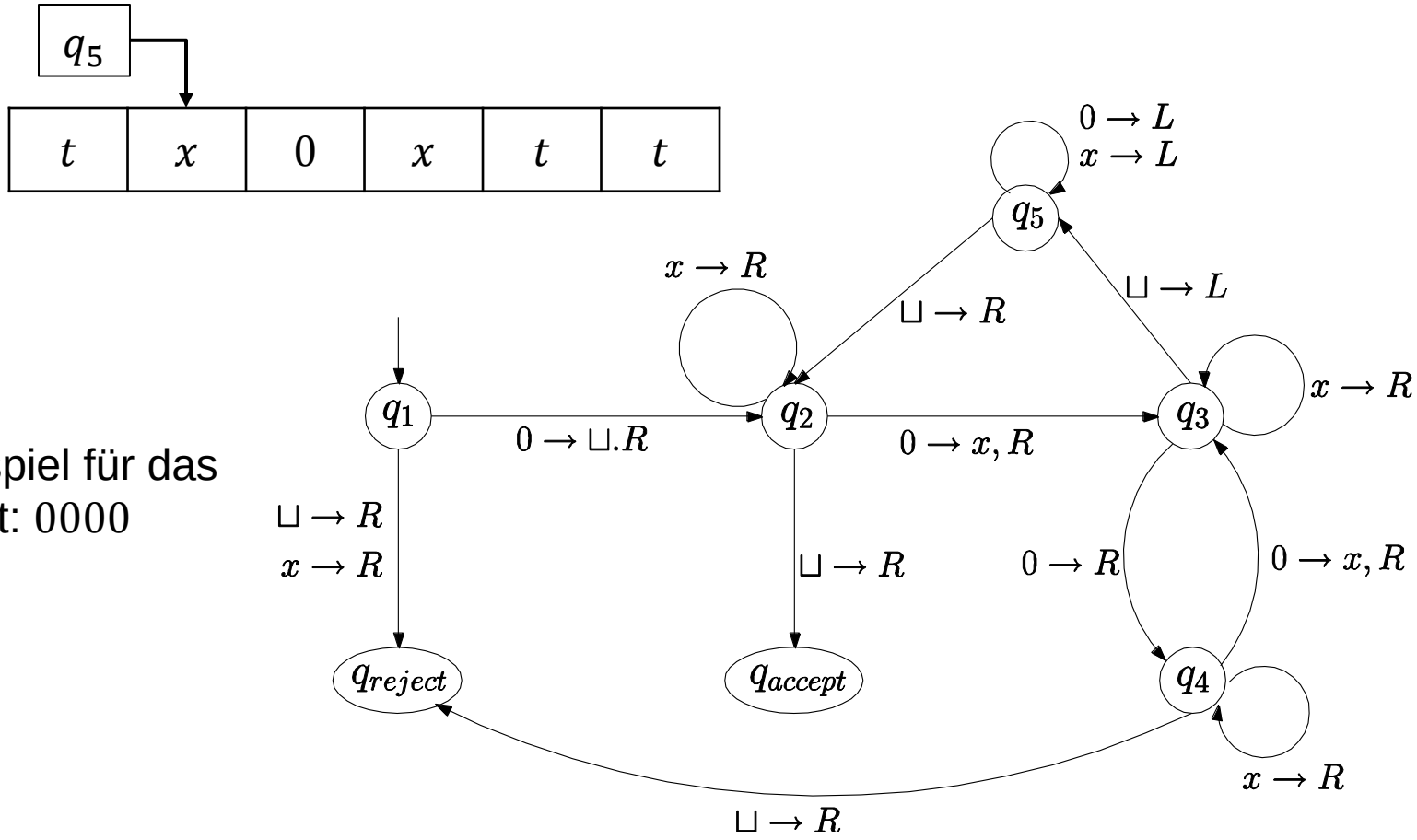
1. Einführung

- Beispiel für das Wort: 0000

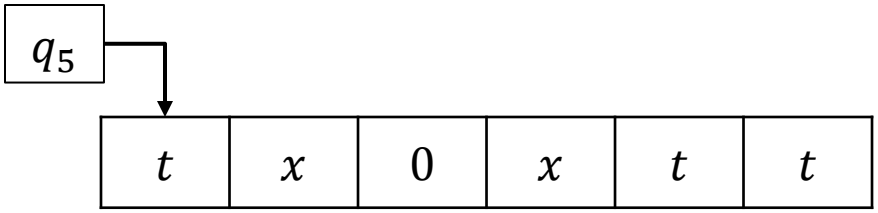


1. Einführung

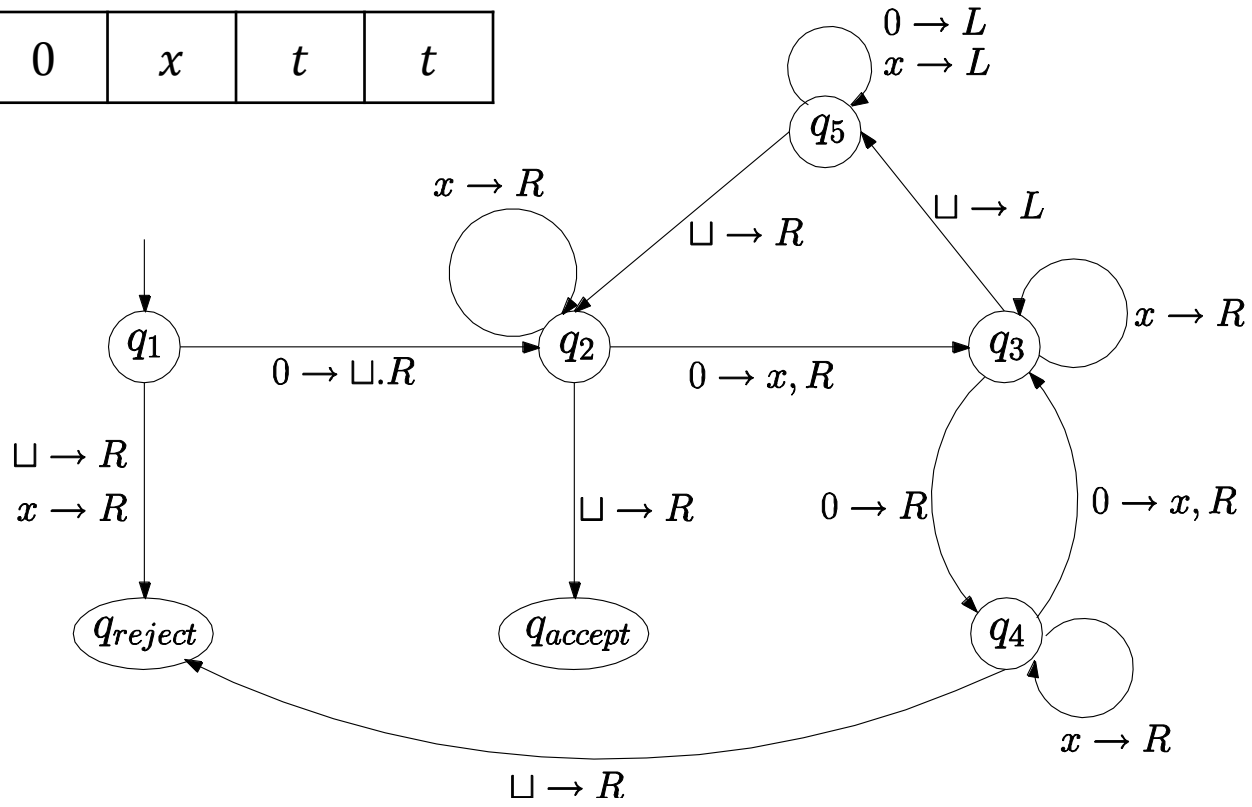
- Beispiel für das Wort: 0000



1. Einführung

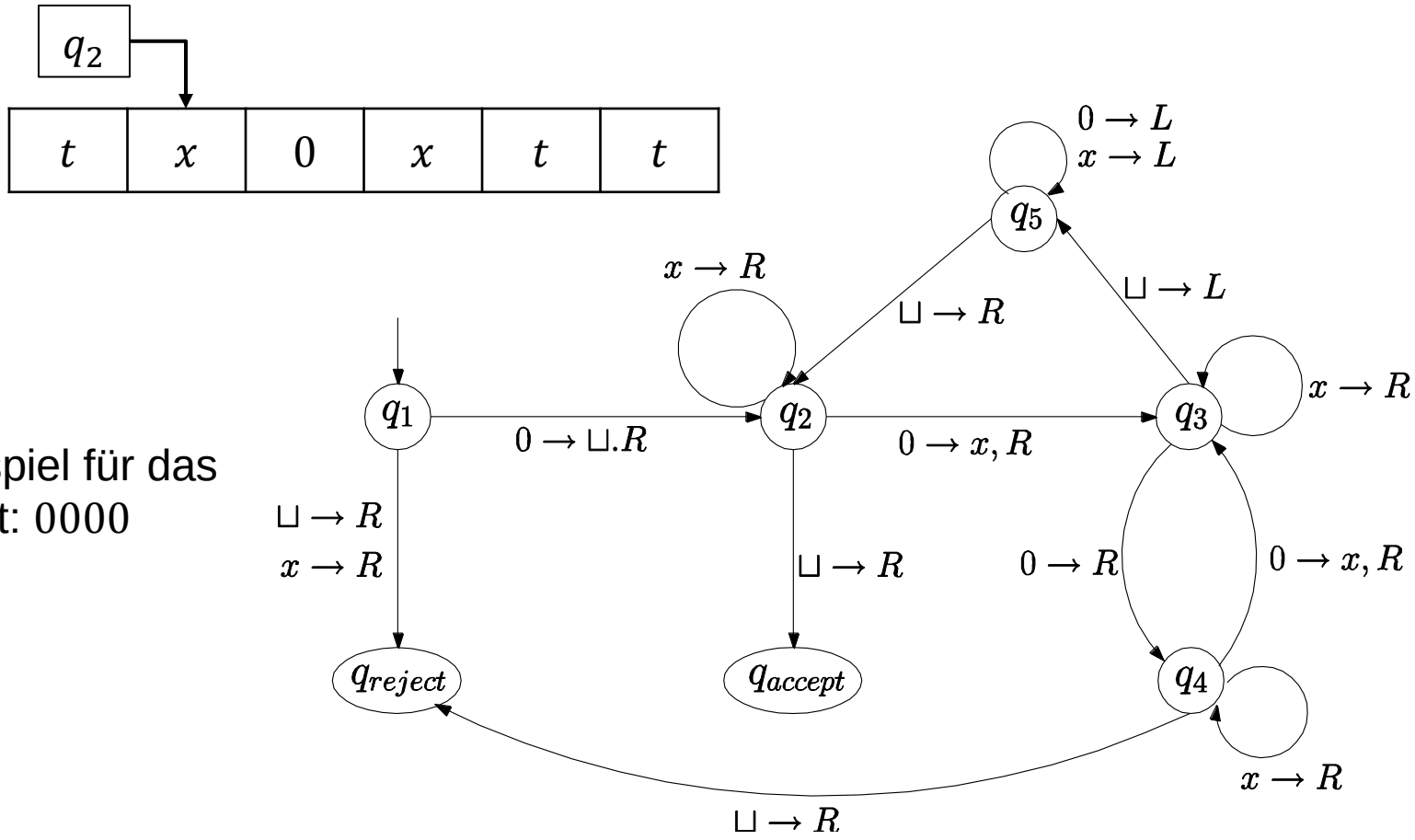


- Beispiel für das Wort: 0000



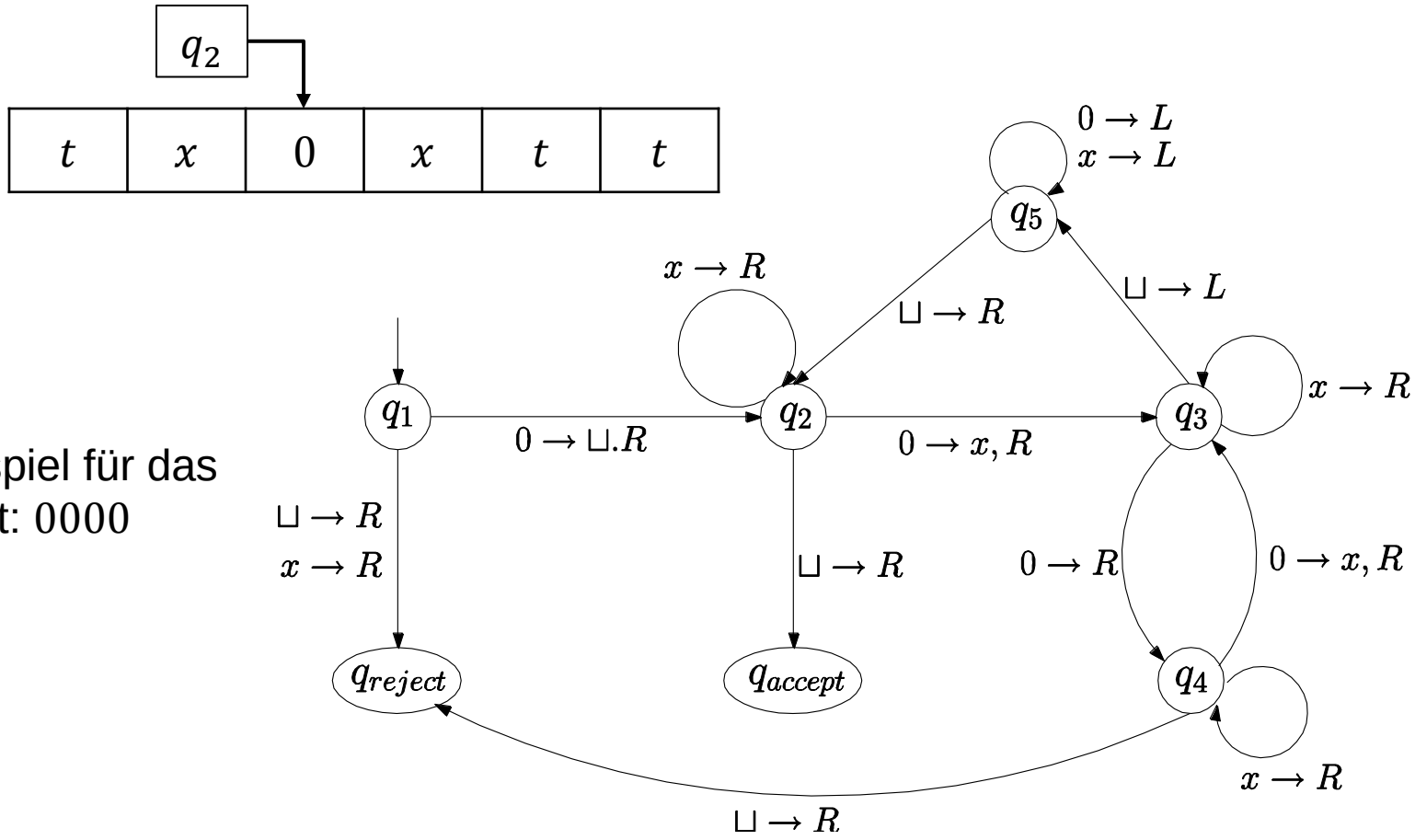
1. Einführung

- Beispiel für das Wort: 0000



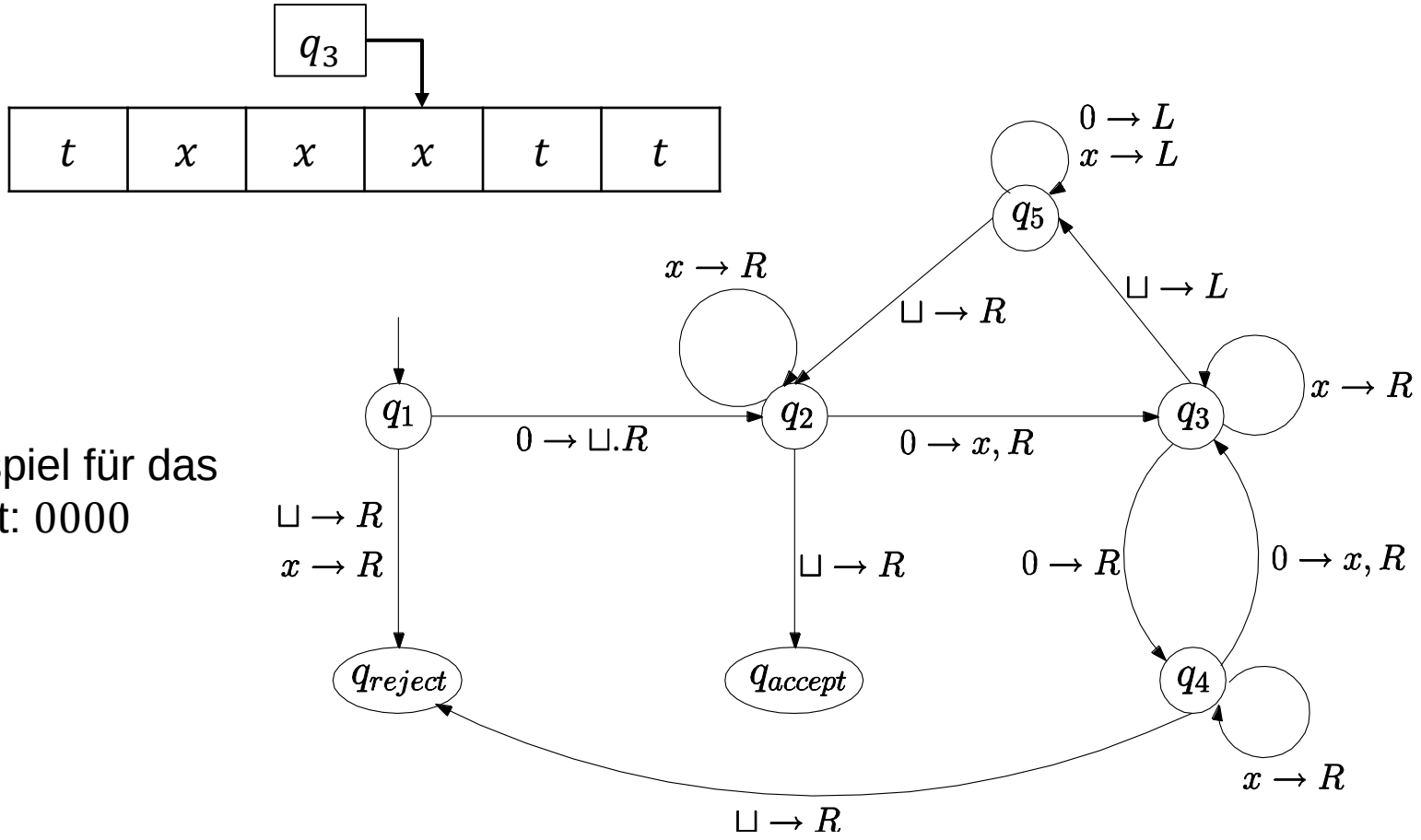
1. Einführung

- Beispiel für das Wort: 0000



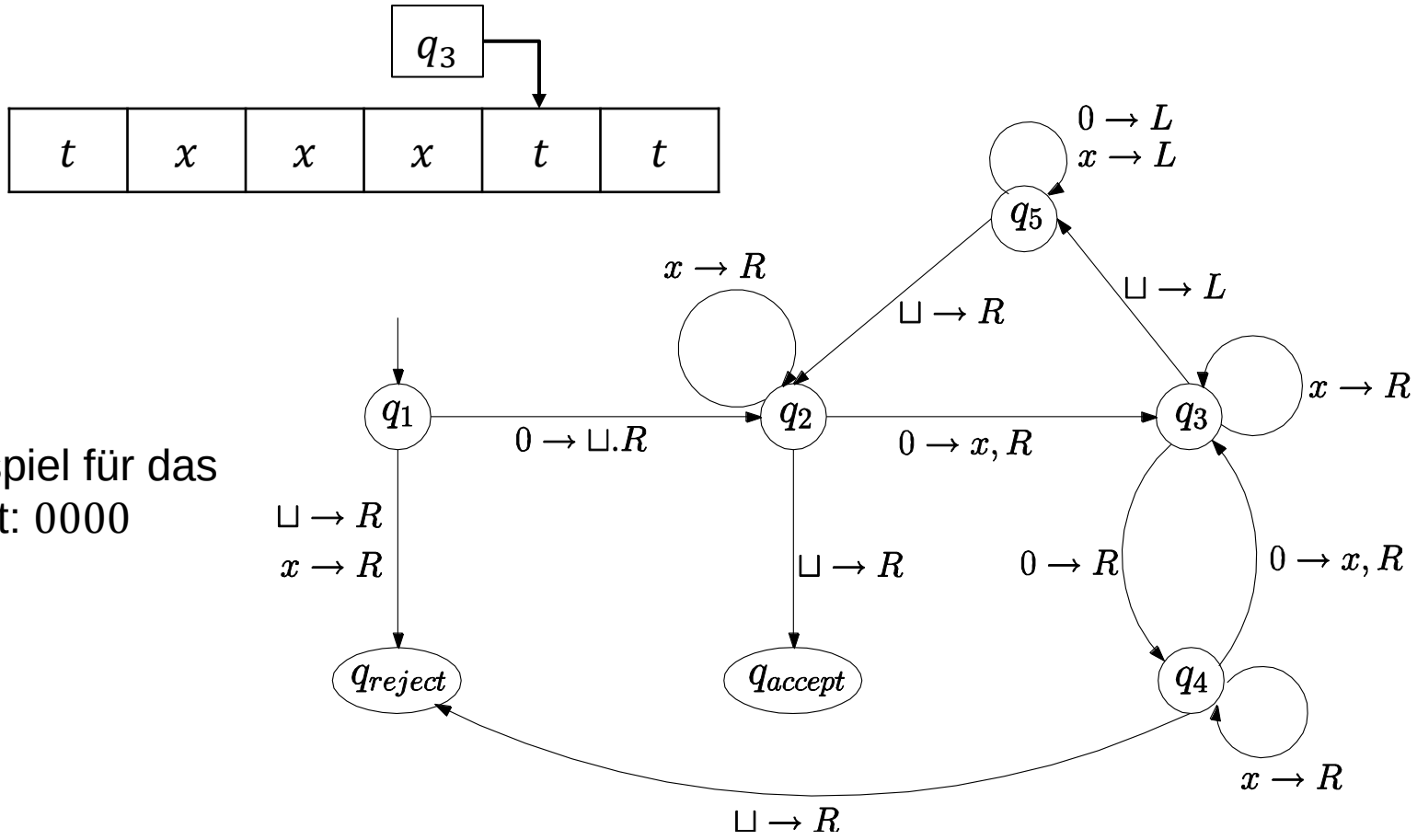
1. Einführung

- Beispiel für das Wort: 0000



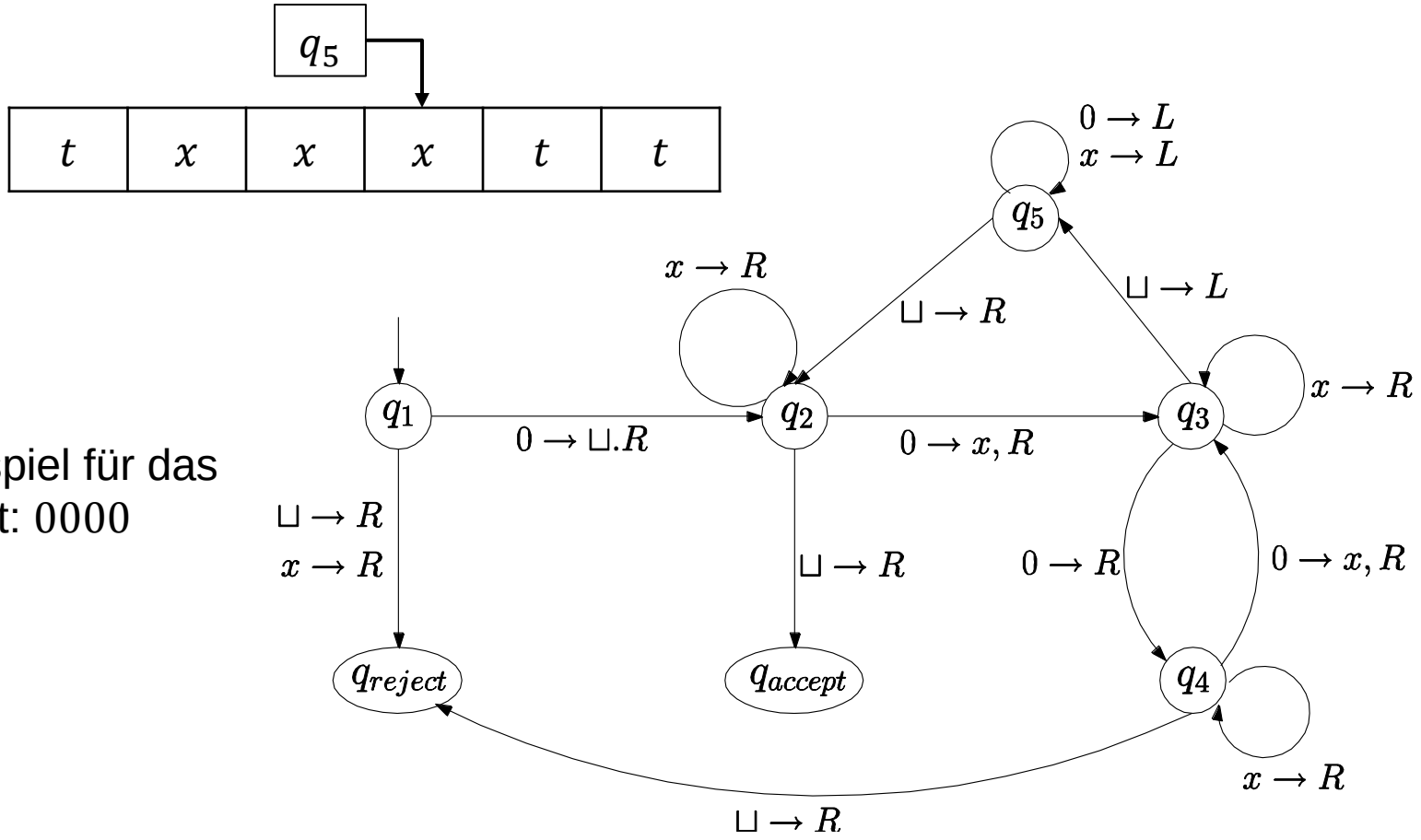
1. Einführung

- Beispiel für das Wort: 0000



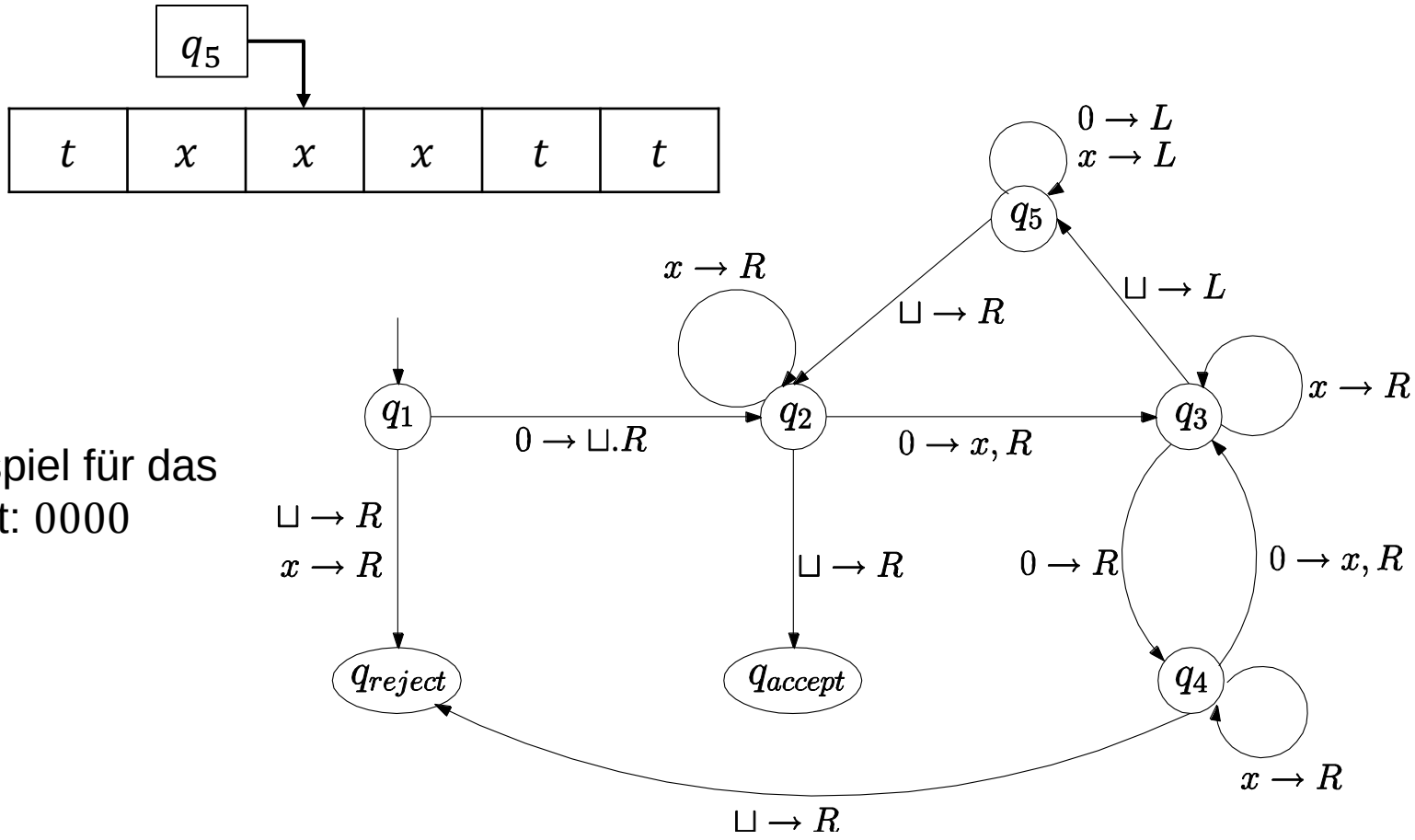
1. Einführung

- Beispiel für das Wort: 0000

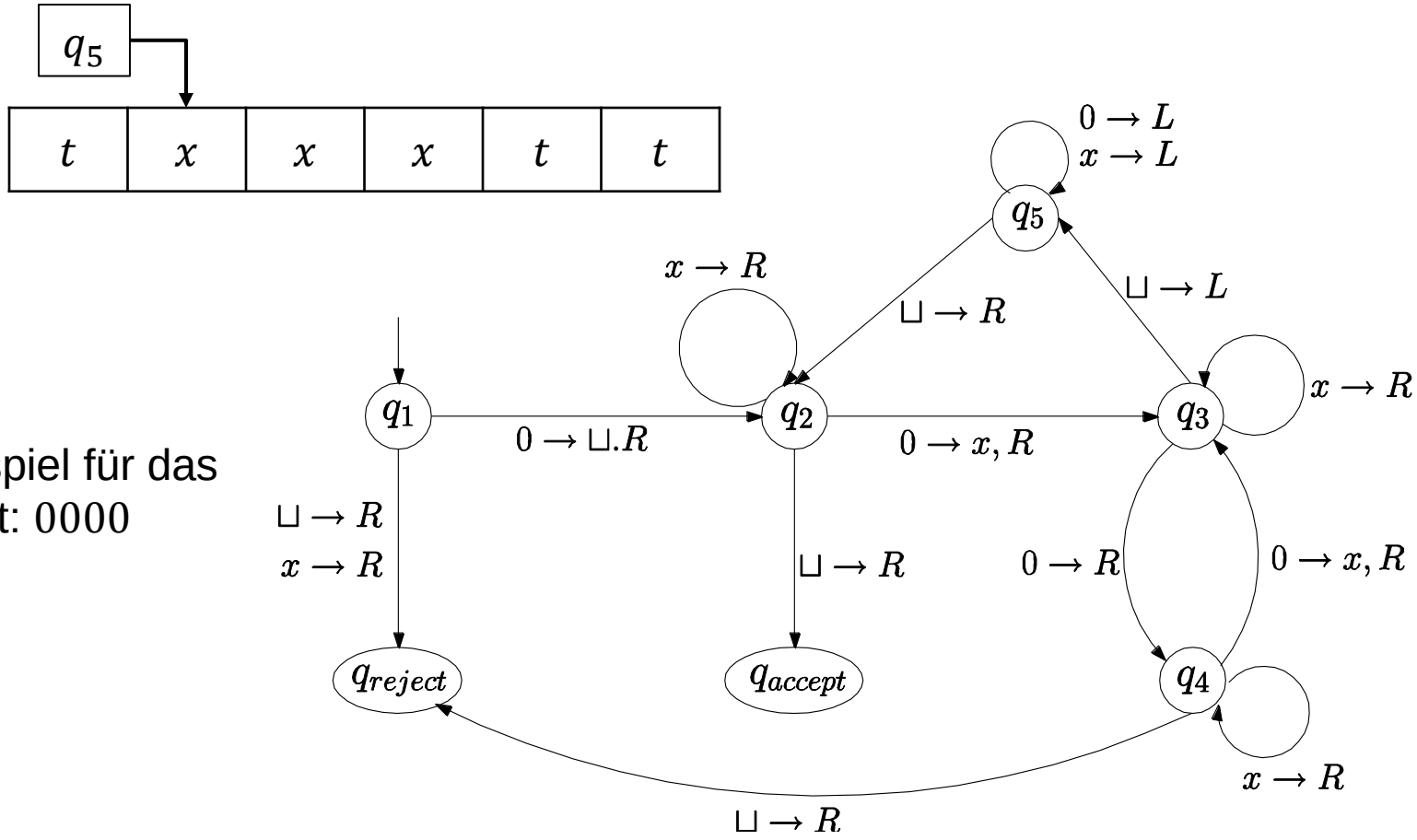


1. Einführung

- Beispiel für das Wort: 0000

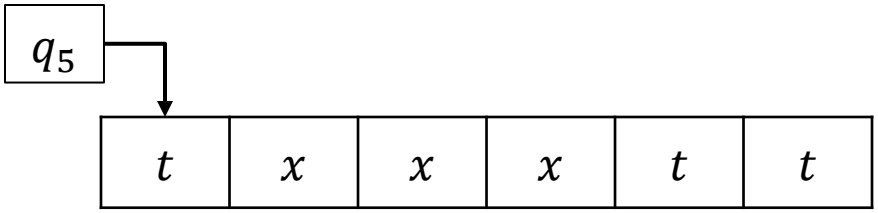


1. Einführung

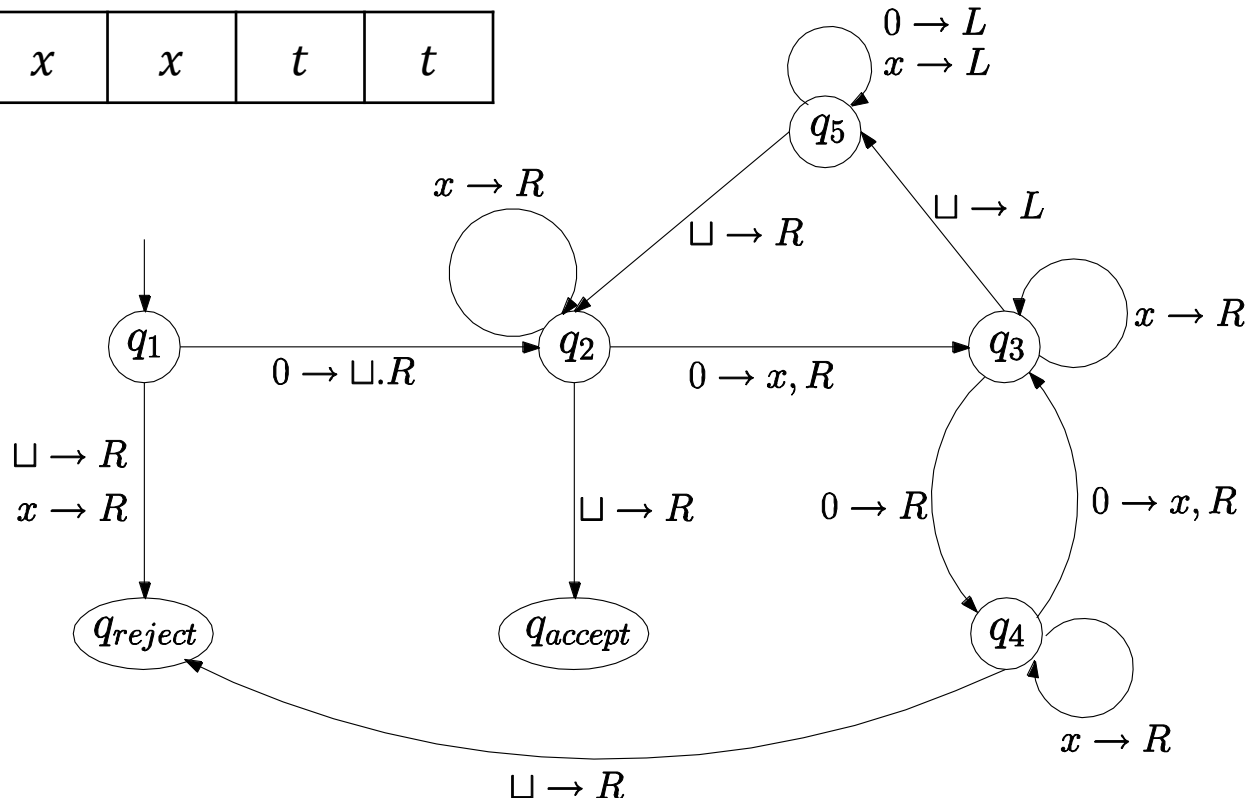


- Beispiel für das Wort: 0000

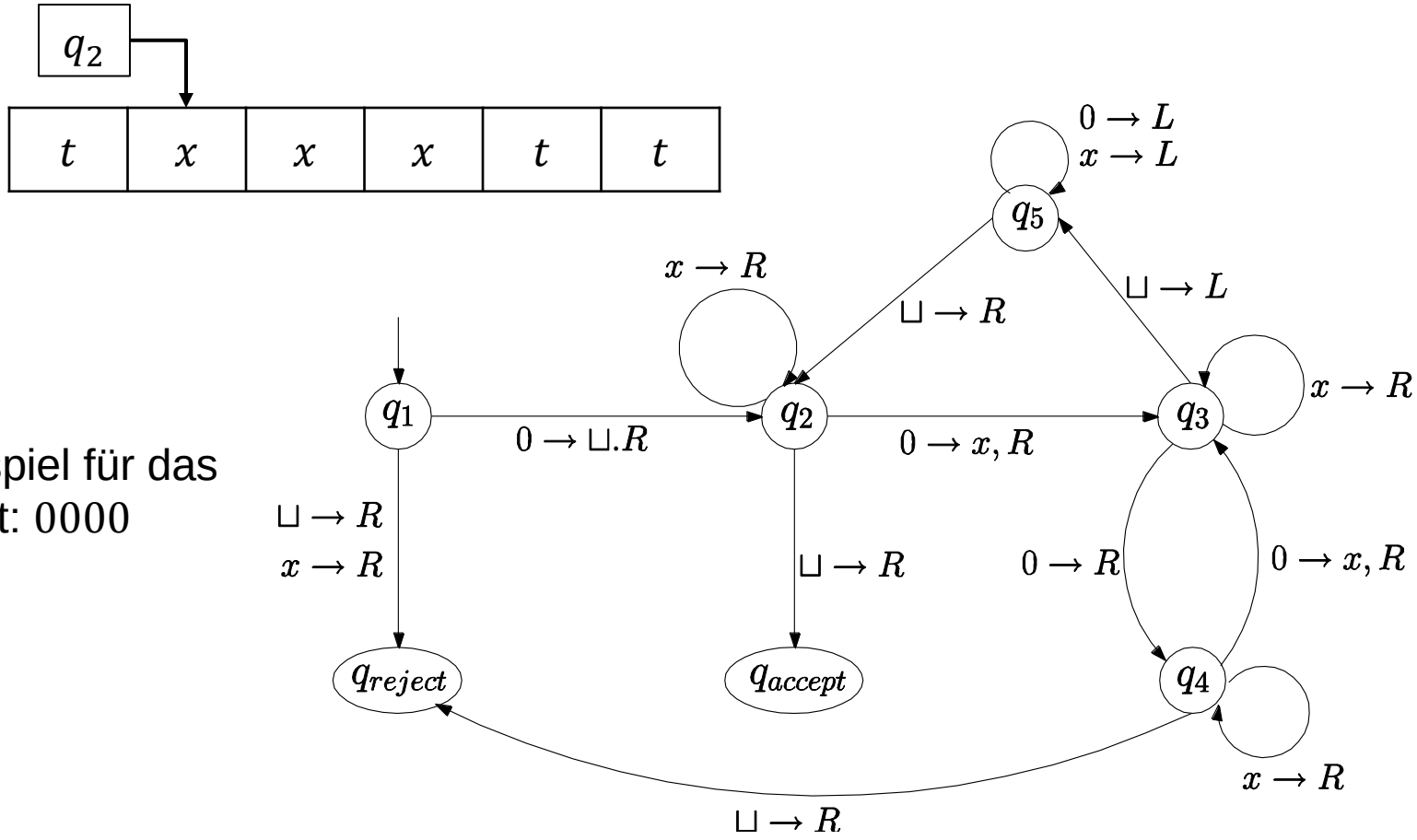
1. Einführung



- Beispiel für das Wort: 0000



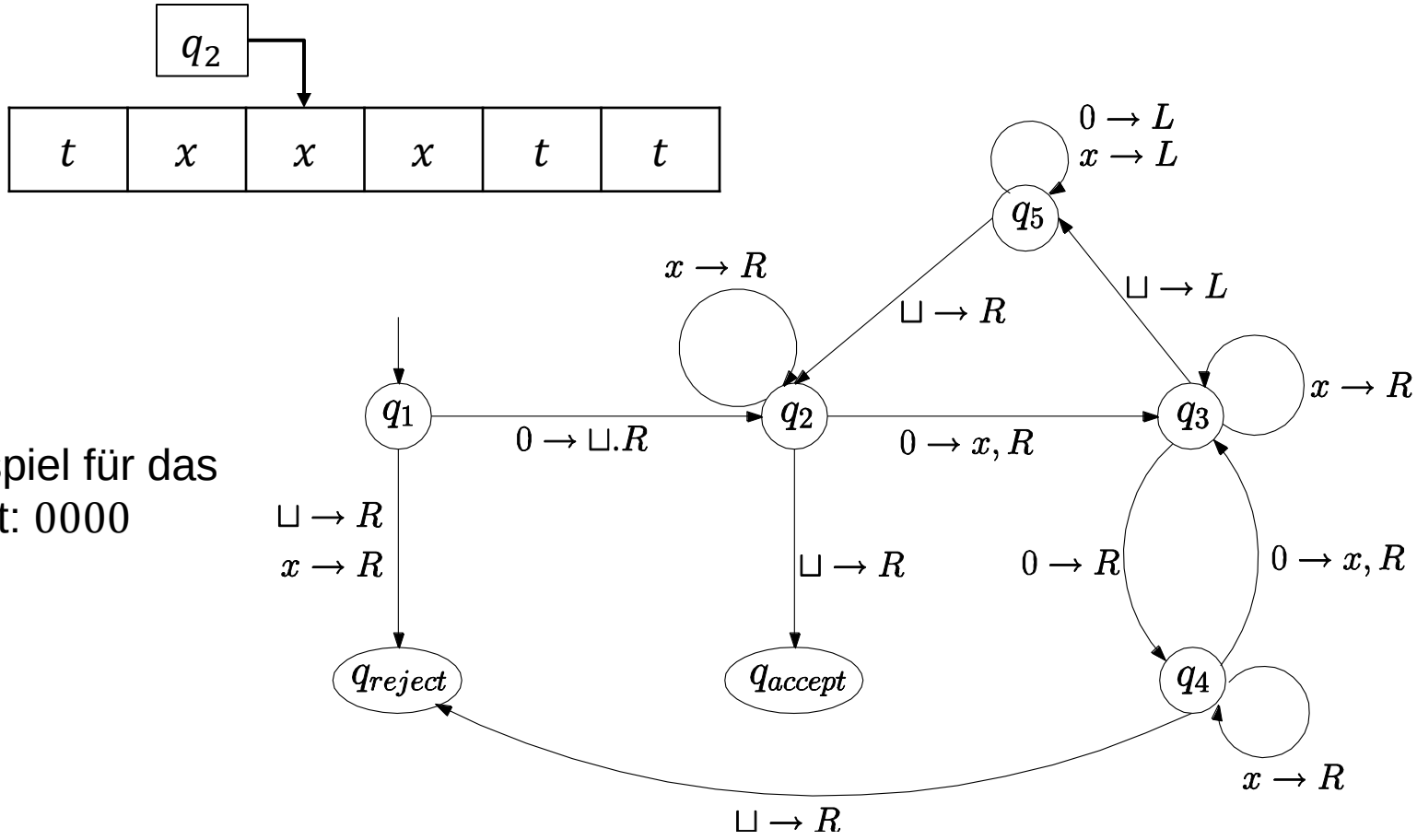
1. Einführung



- Beispiel für das Wort: 0000

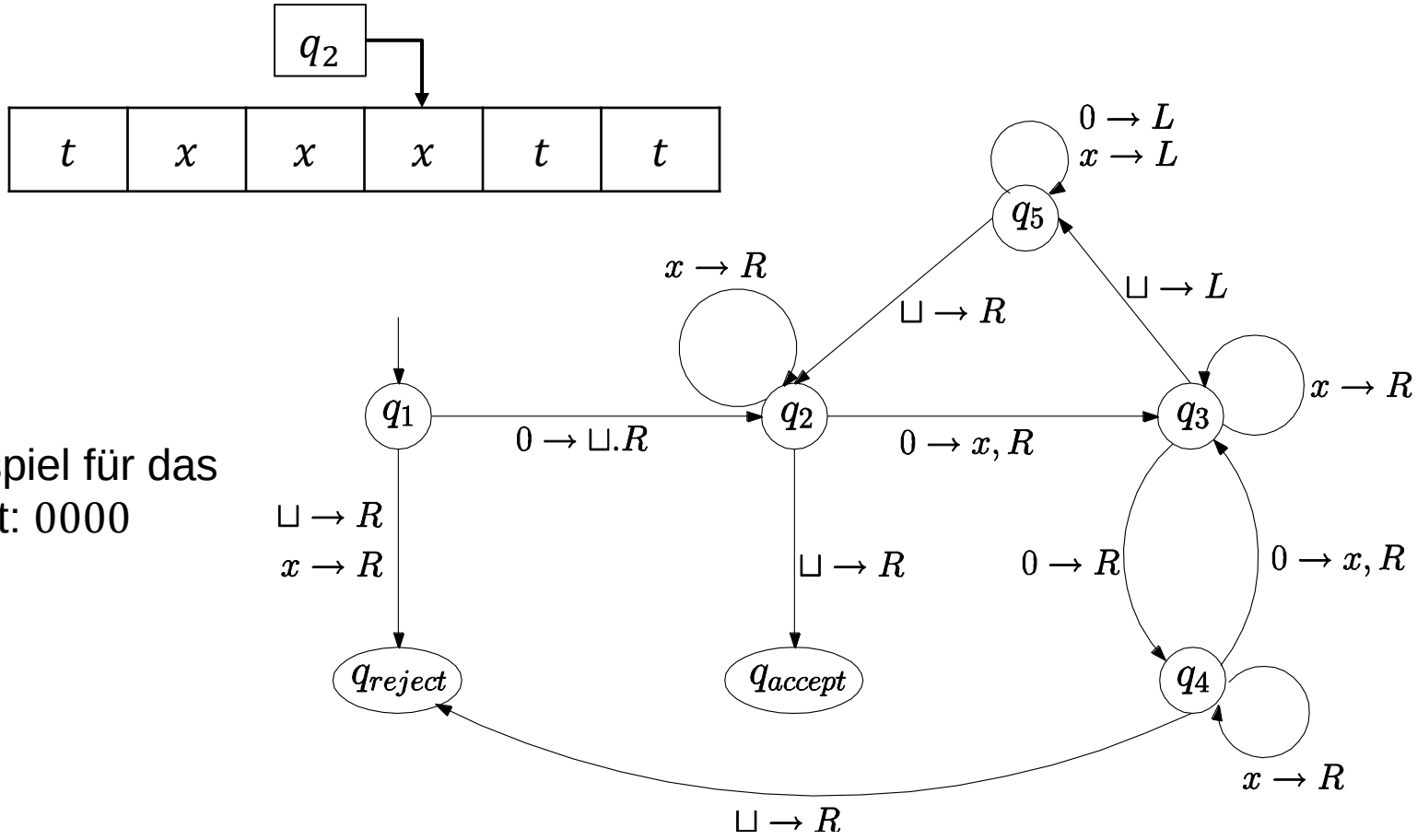
1. Einführung

- Beispiel für das Wort: 0000



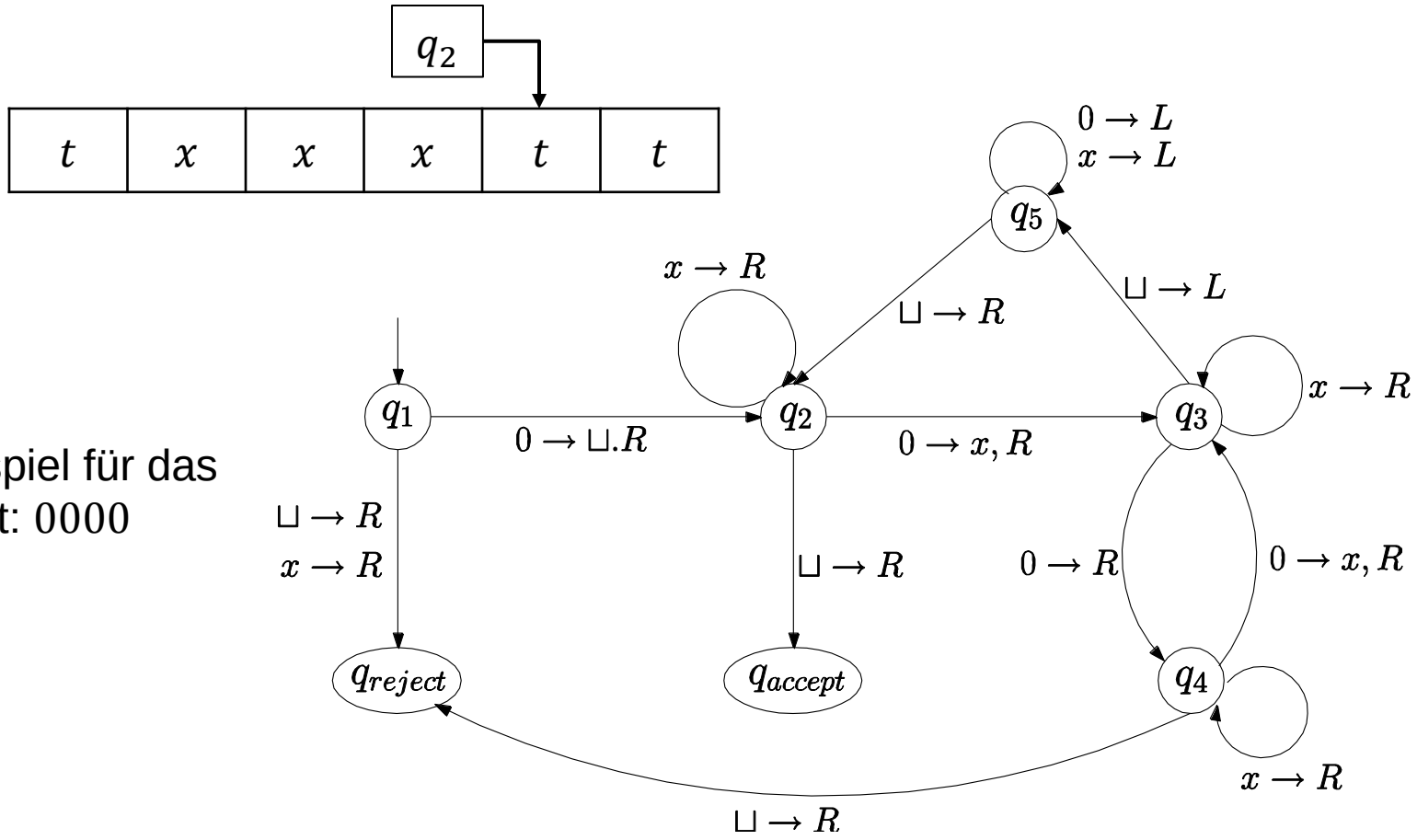
1. Einführung

- Beispiel für das Wort: 0000

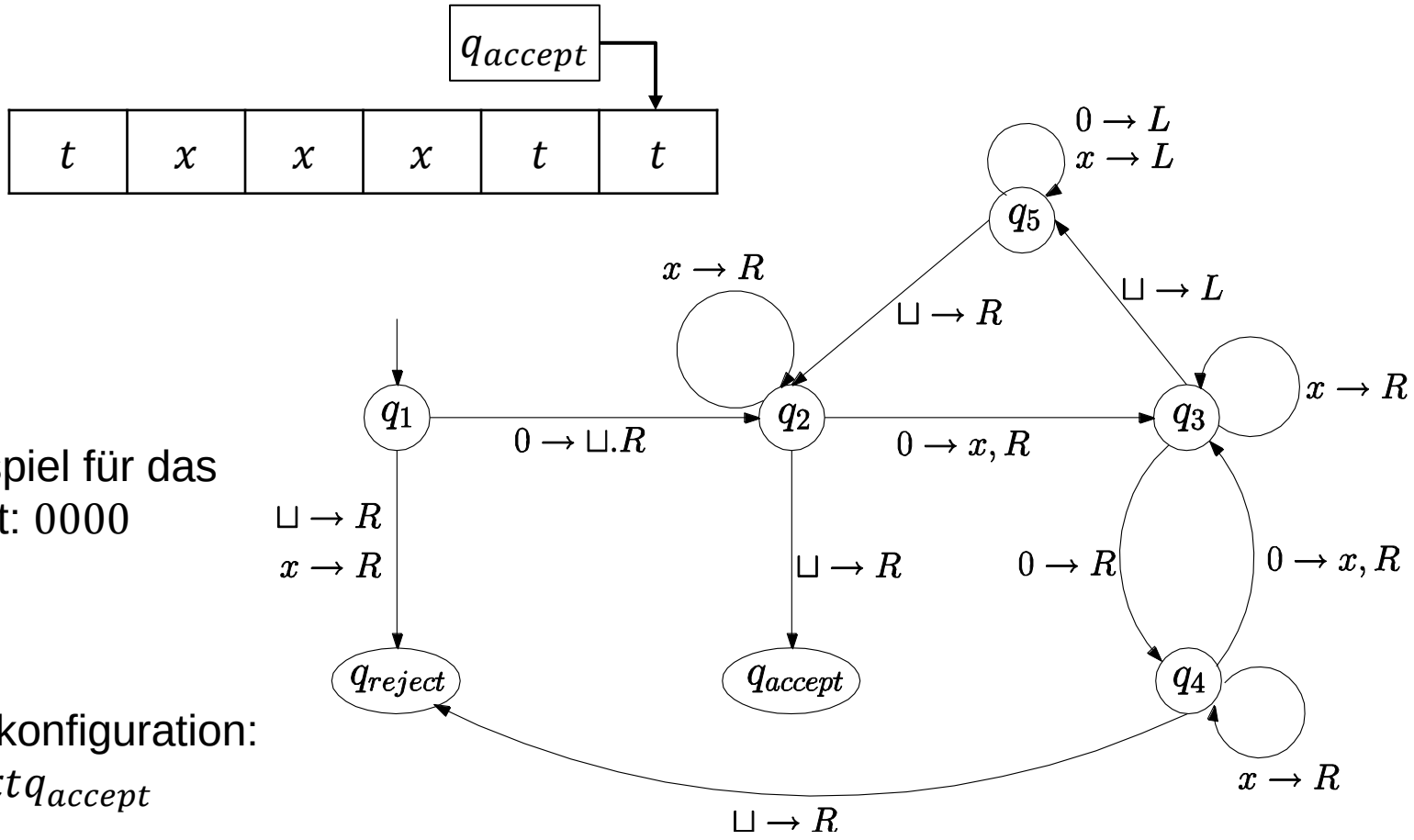


1. Einführung

- Beispiel für das Wort: 0000



1. Einführung



- Beispiel für das Wort: 0000

- Endkonfiguration: $txxxtq_{accept}$

1. Einführung

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

1. Einführung

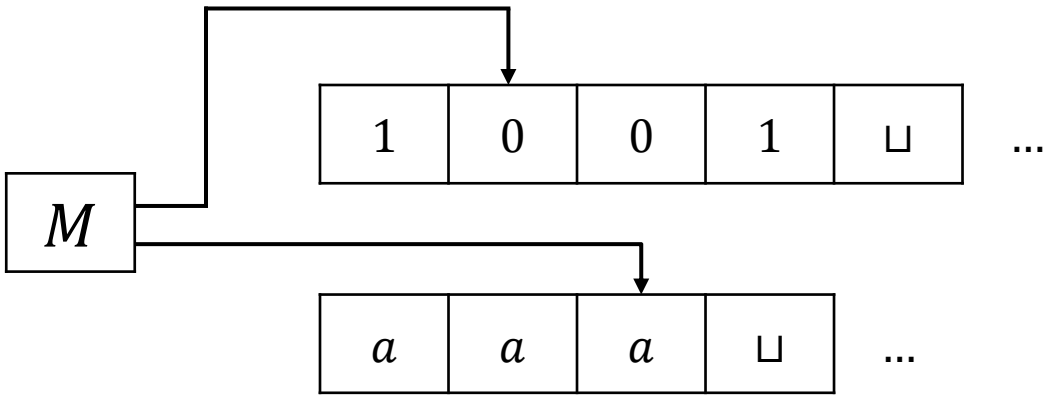
Satz

Zu jeder Mehrband-Turingmaschine gibt es eine äquivalente 1-Band-Turingmaschine.

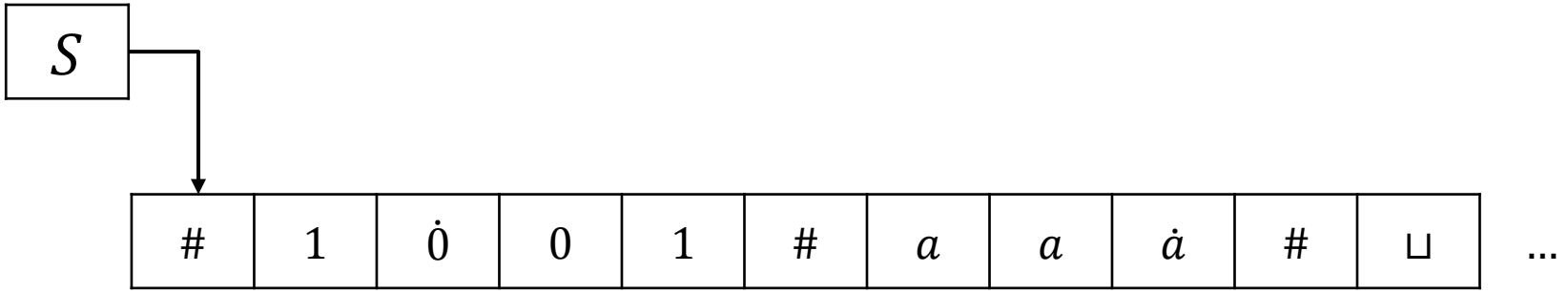
Beweis

Idee:
Simuliere Mehrband-DTM M auf 1-Band-DTM S.

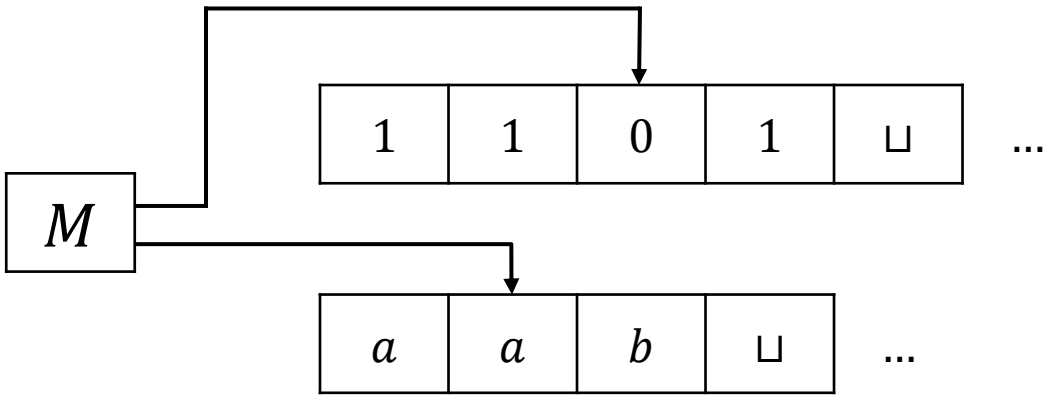
1. Einführung



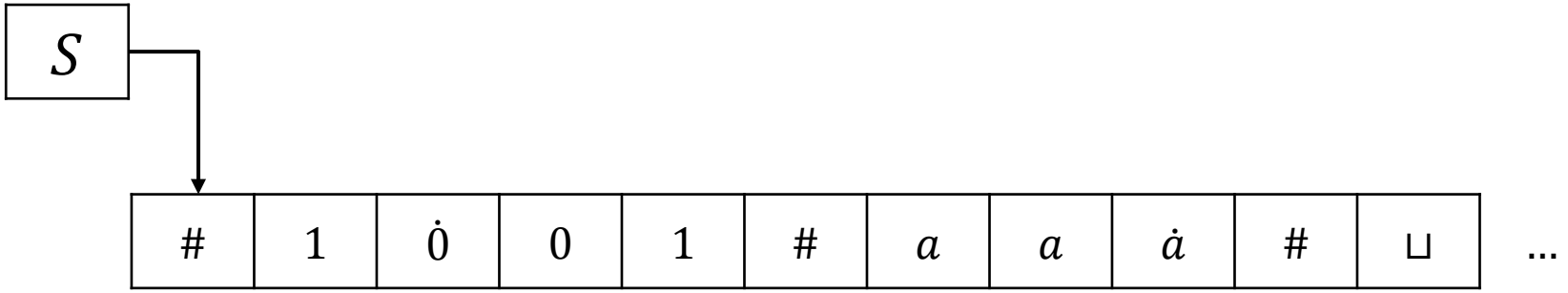
$$\delta(q, 0, a) = (q', 1, b, R, L)$$



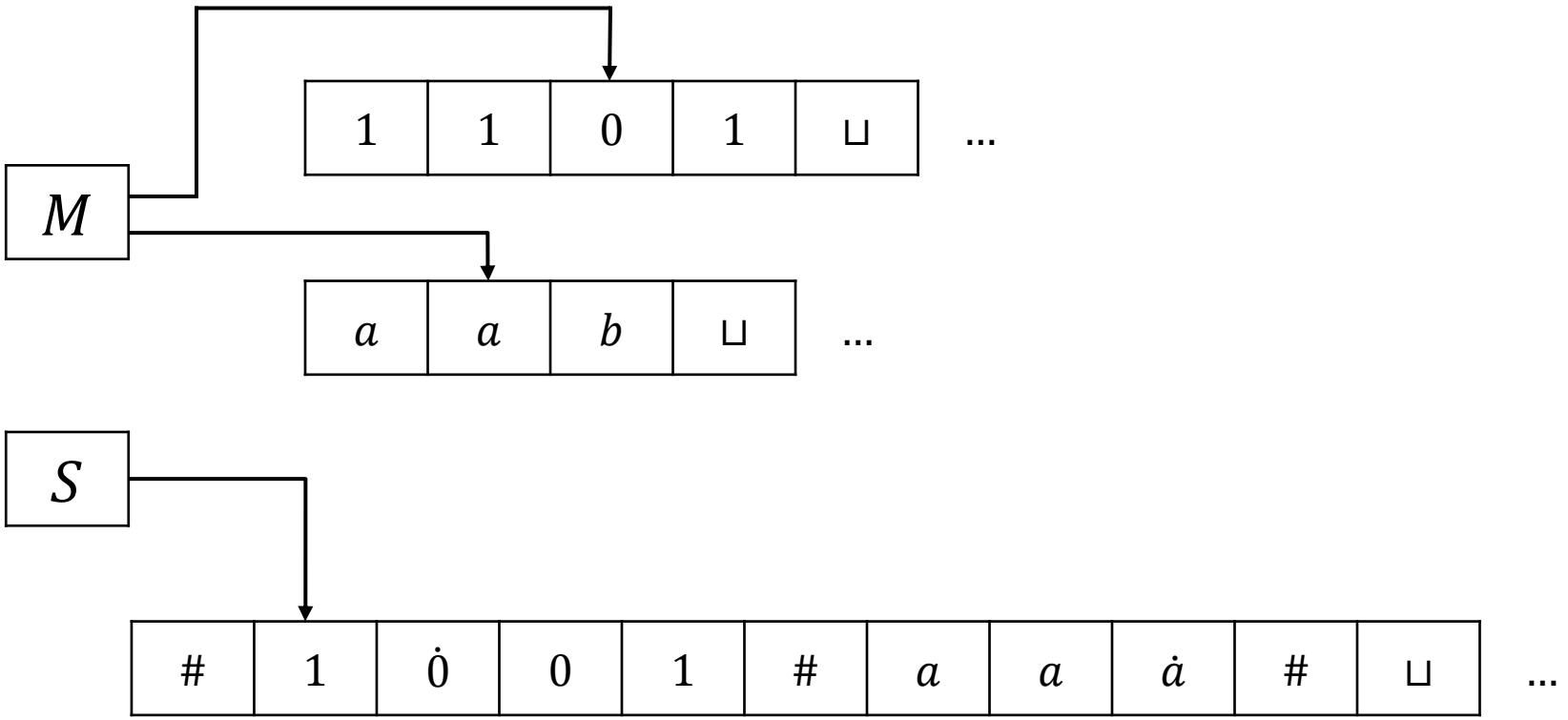
1. Einführung



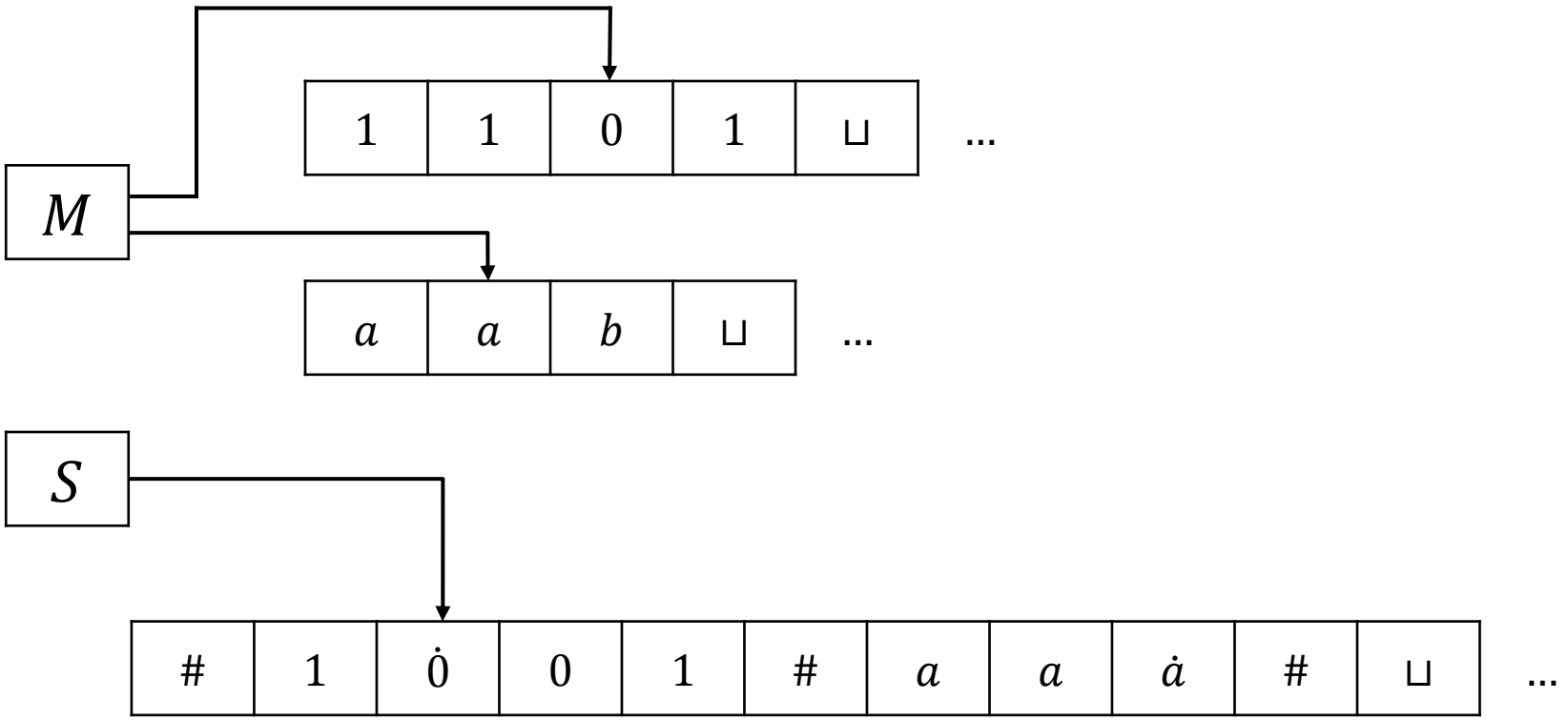
$$\delta(q, 0, a) = (q', 1, b, R, L)$$



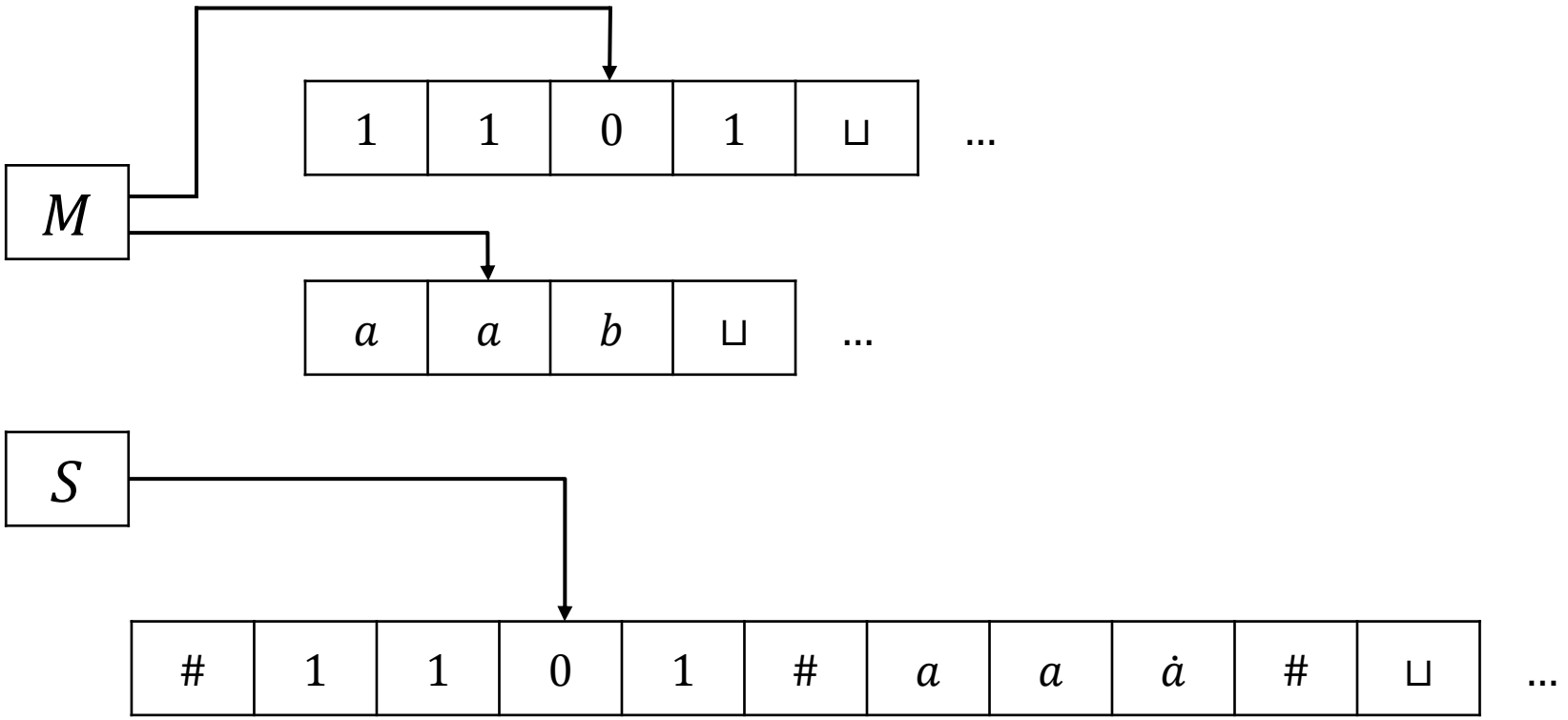
1. Einführung



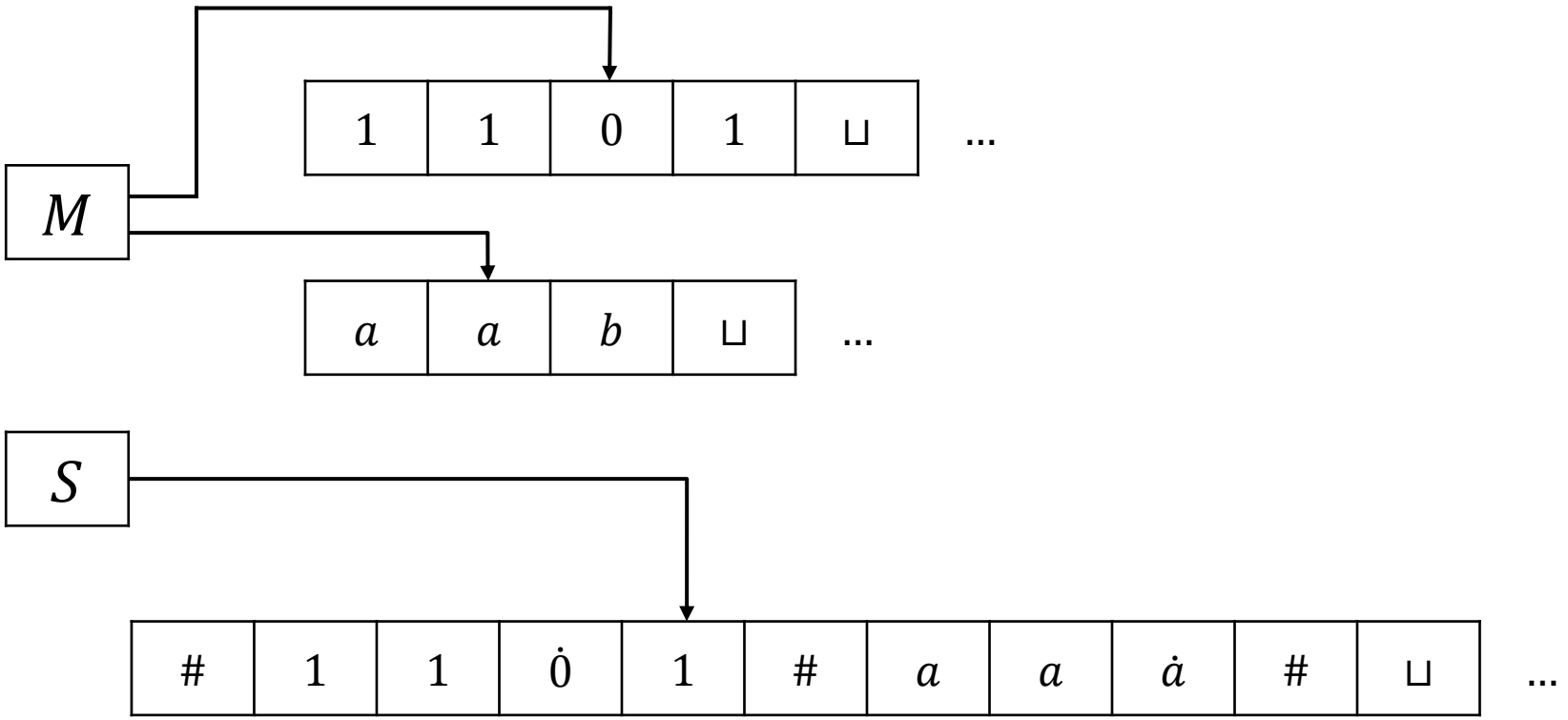
1. Einführung



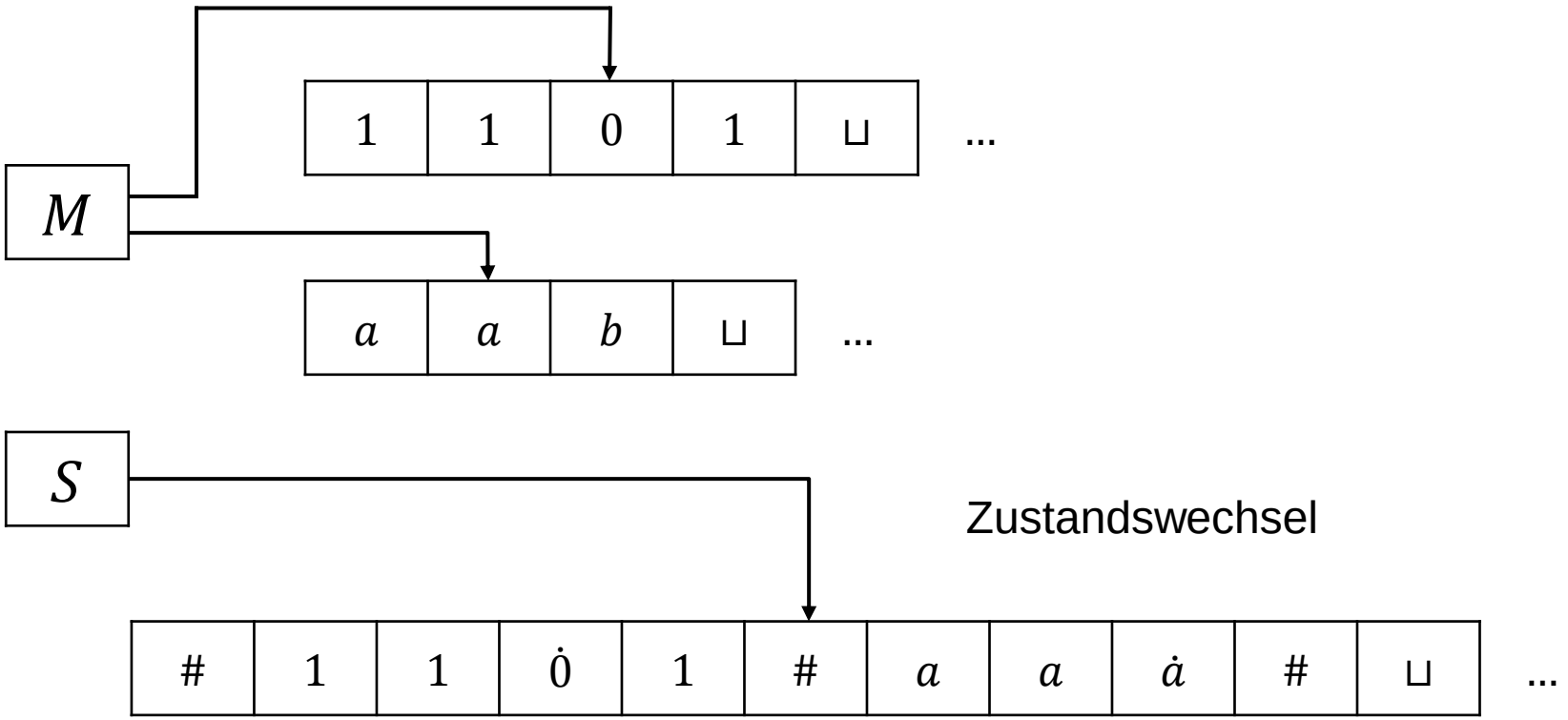
1. Einführung



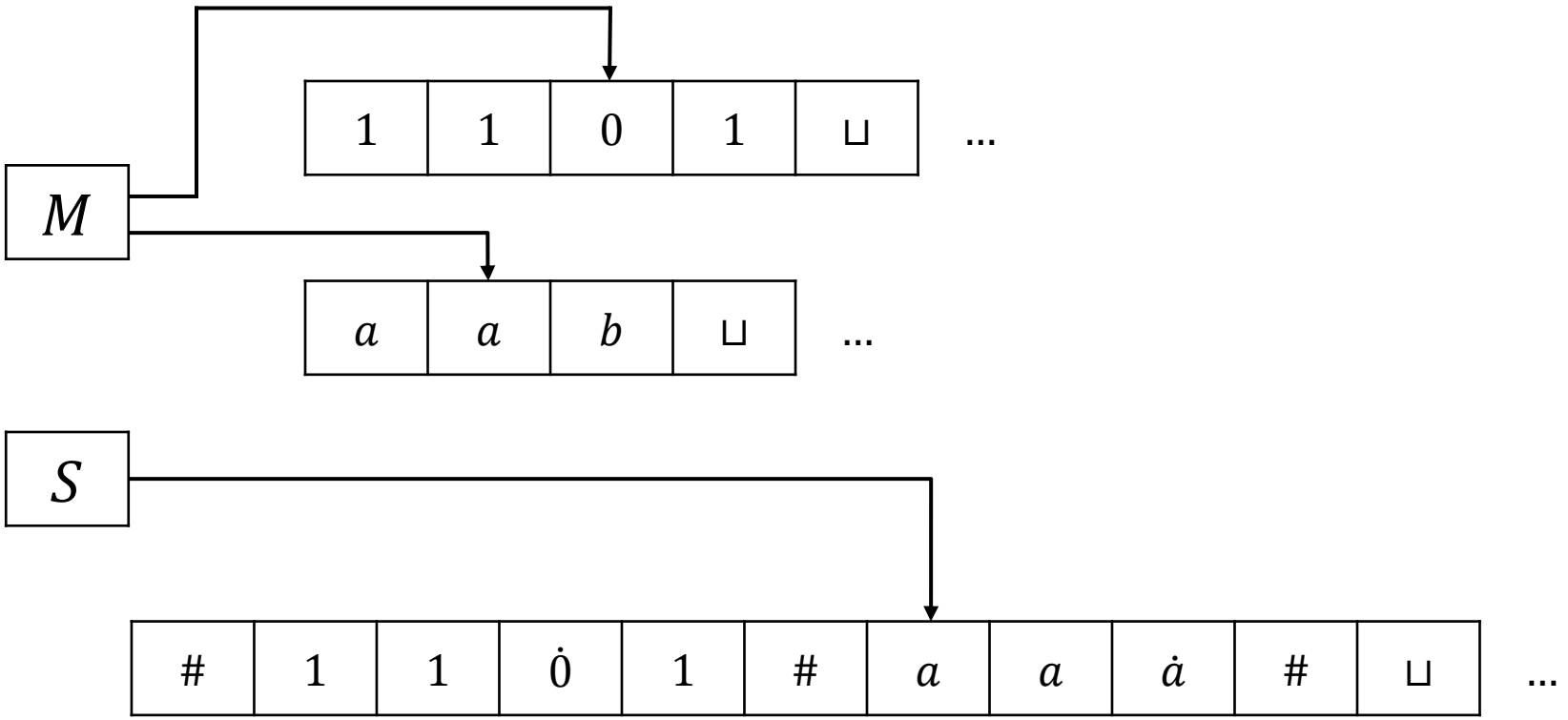
1. Einführung



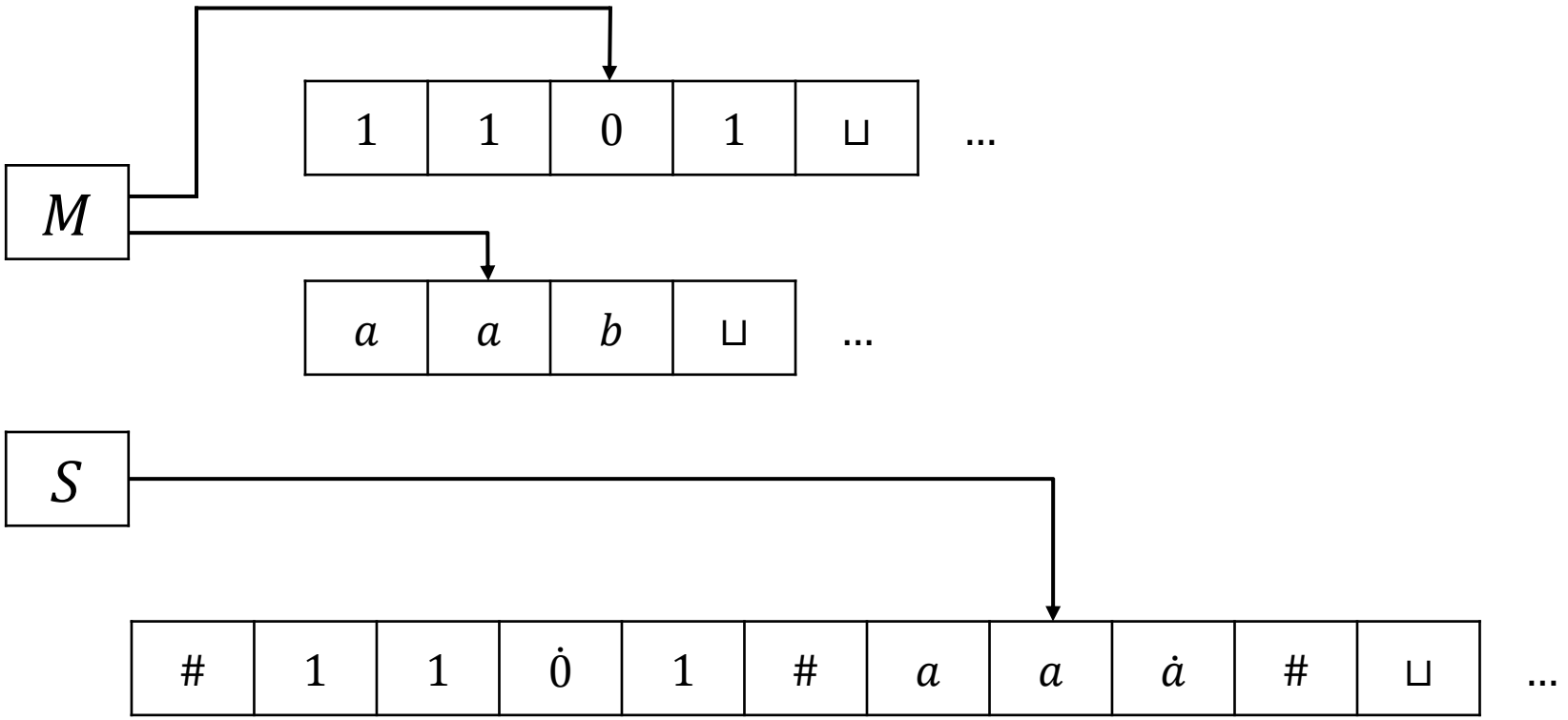
1. Einführung



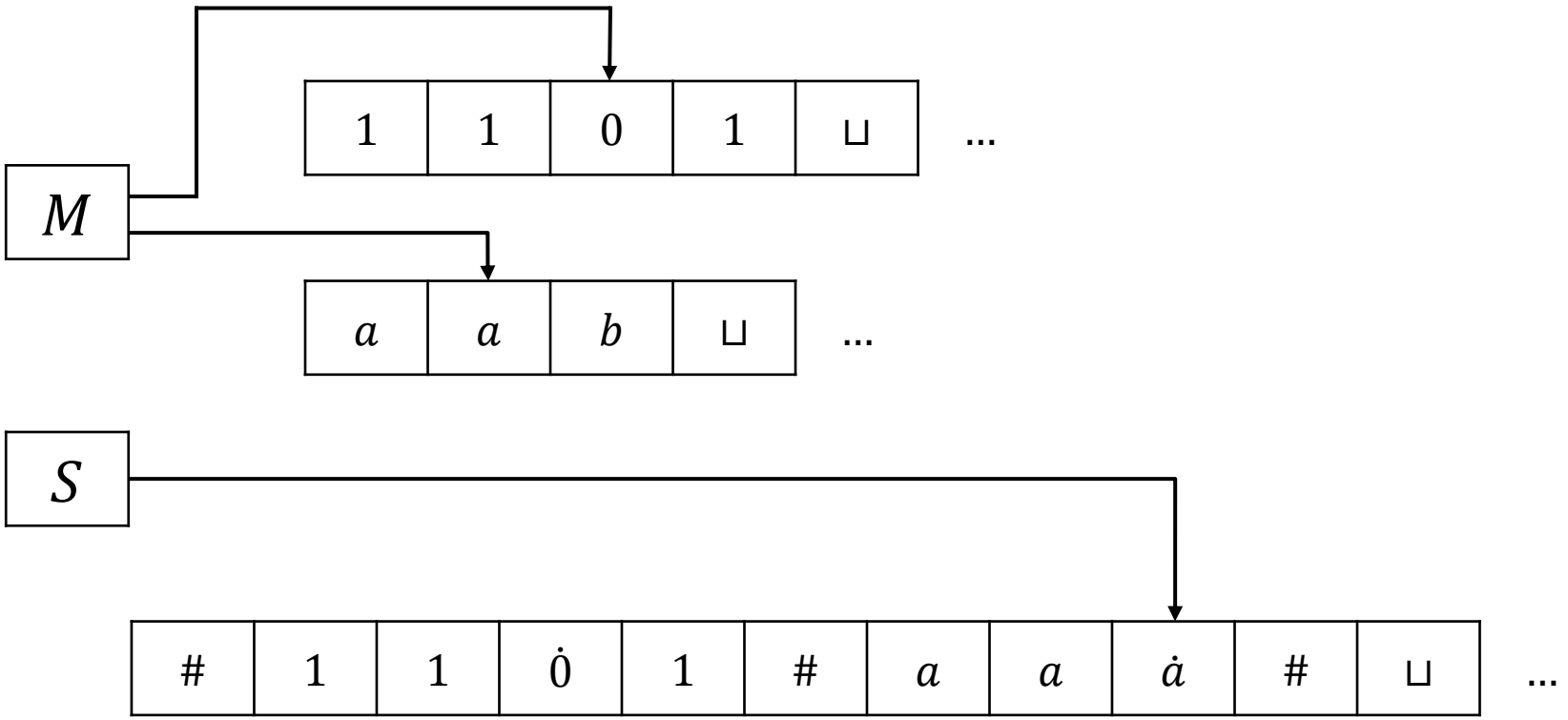
1. Einführung



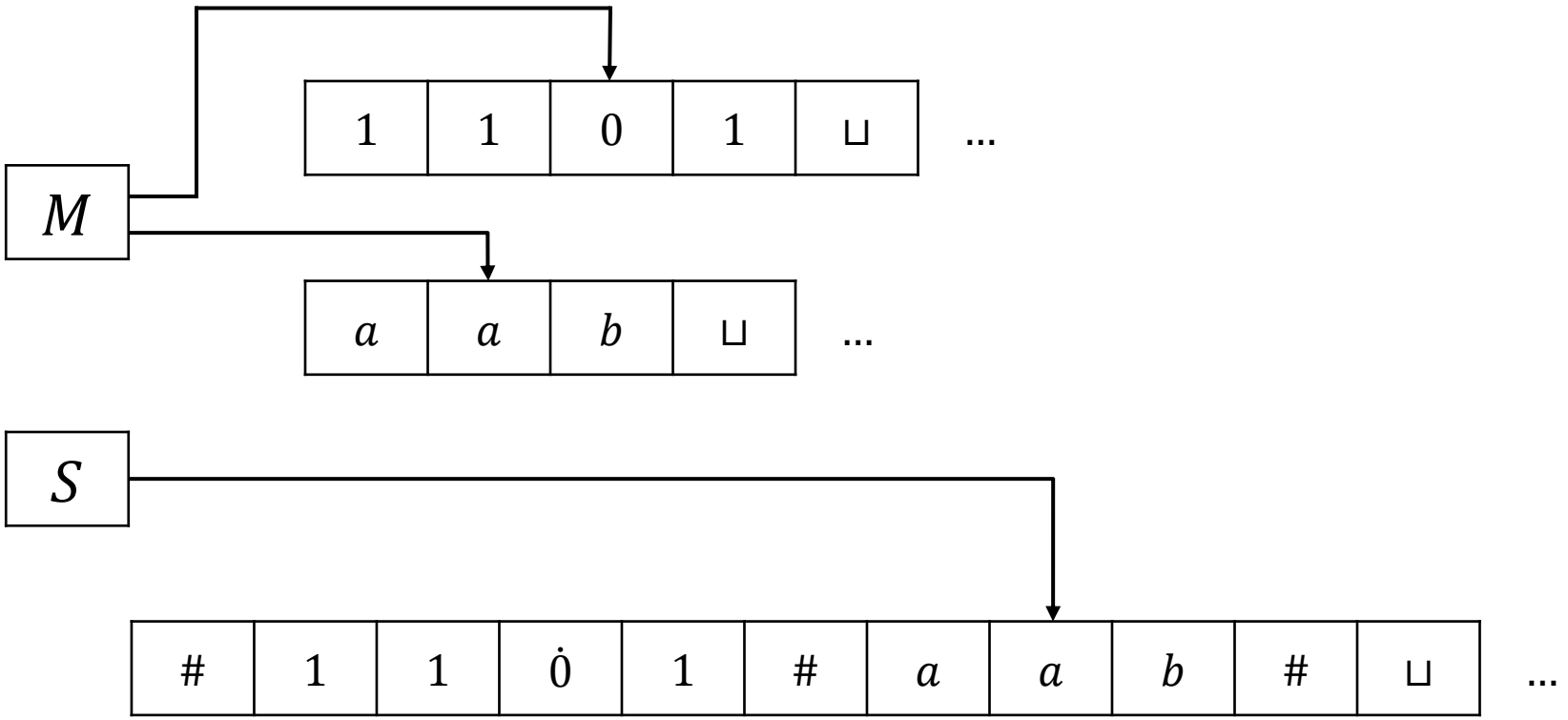
1. Einführung



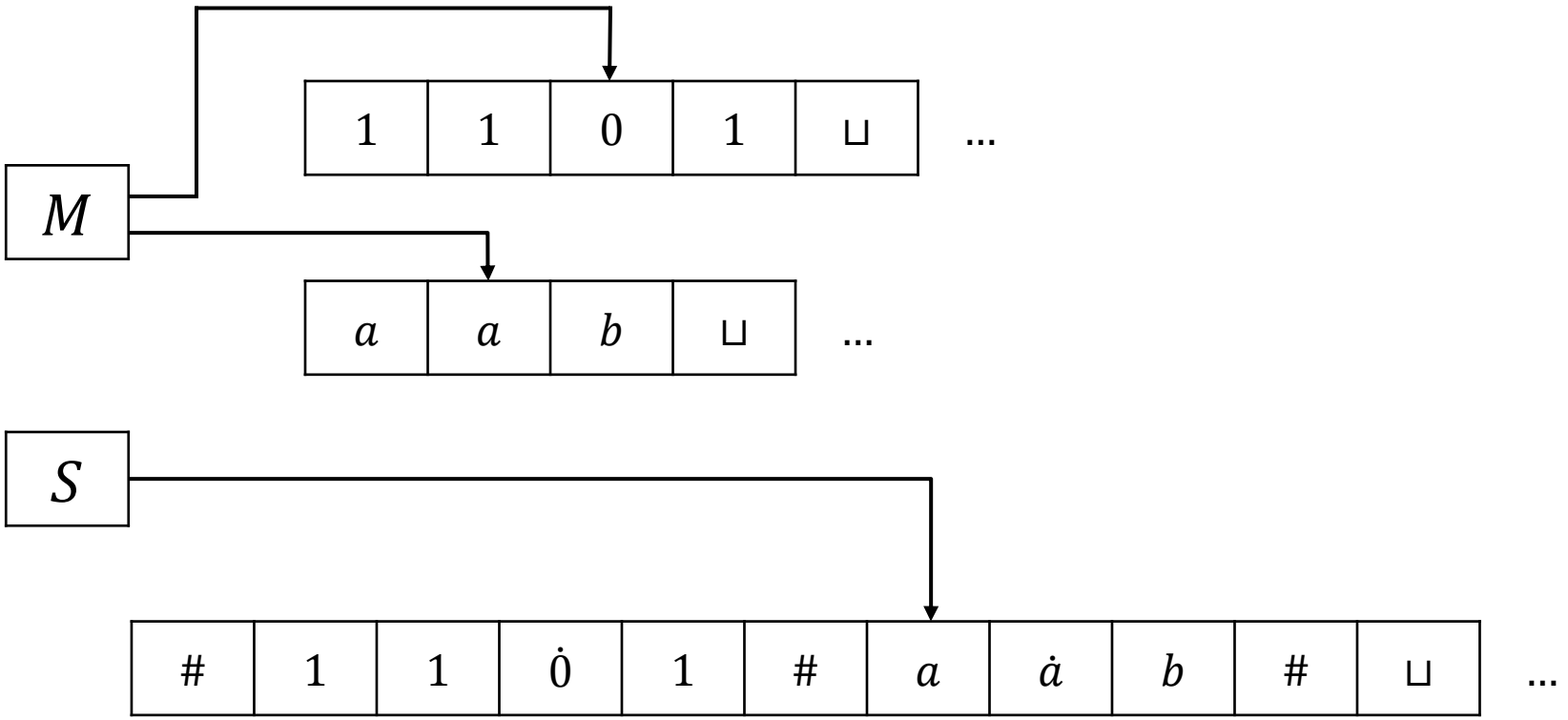
1. Einführung



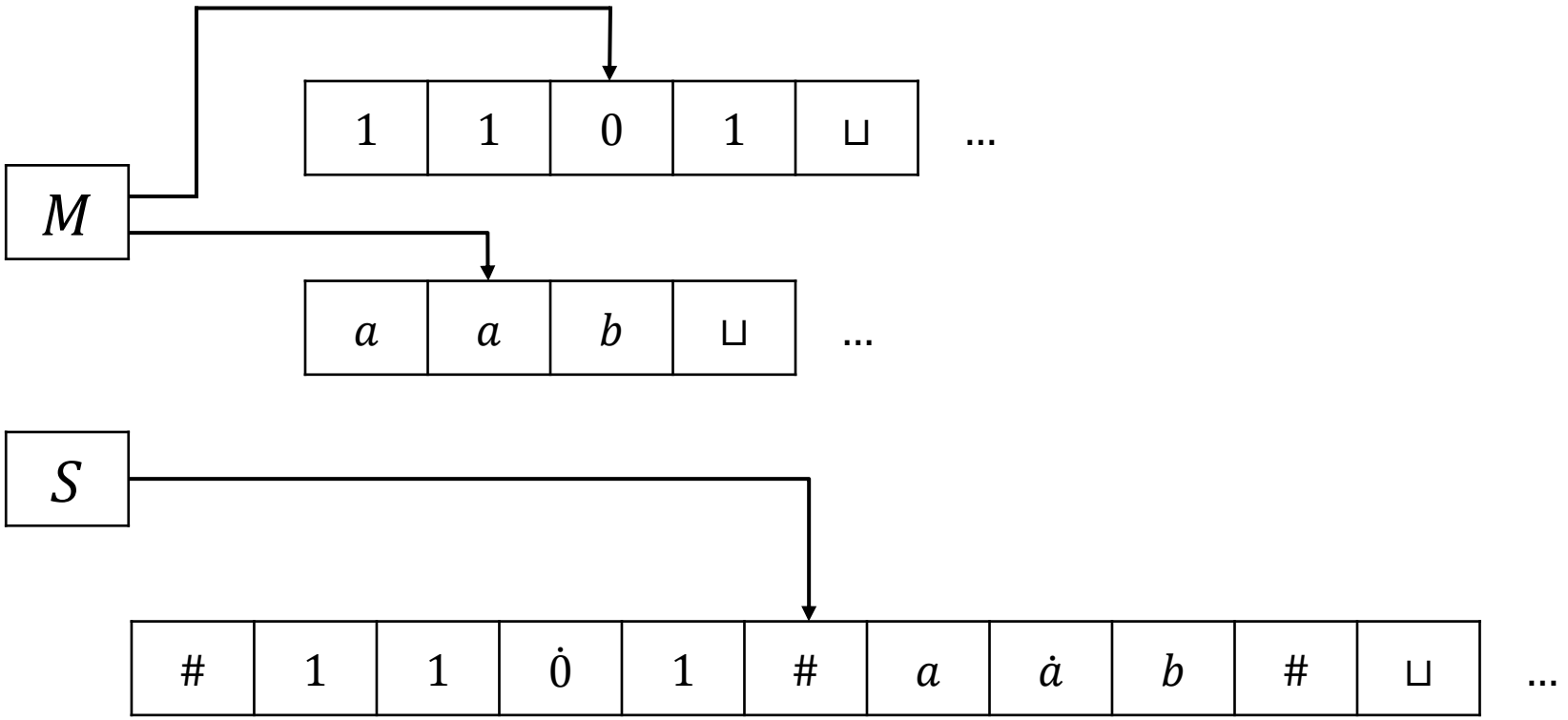
1. Einführung



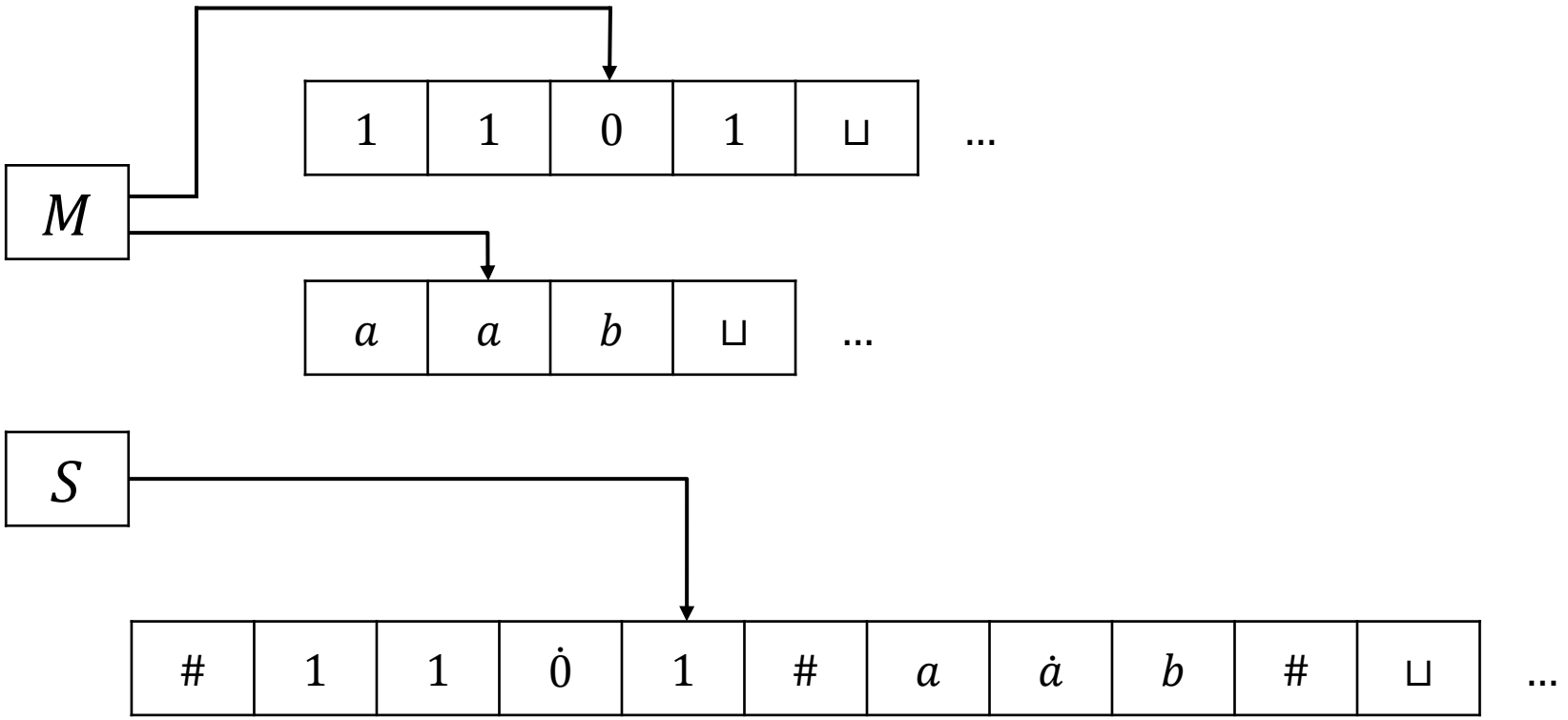
1. Einführung



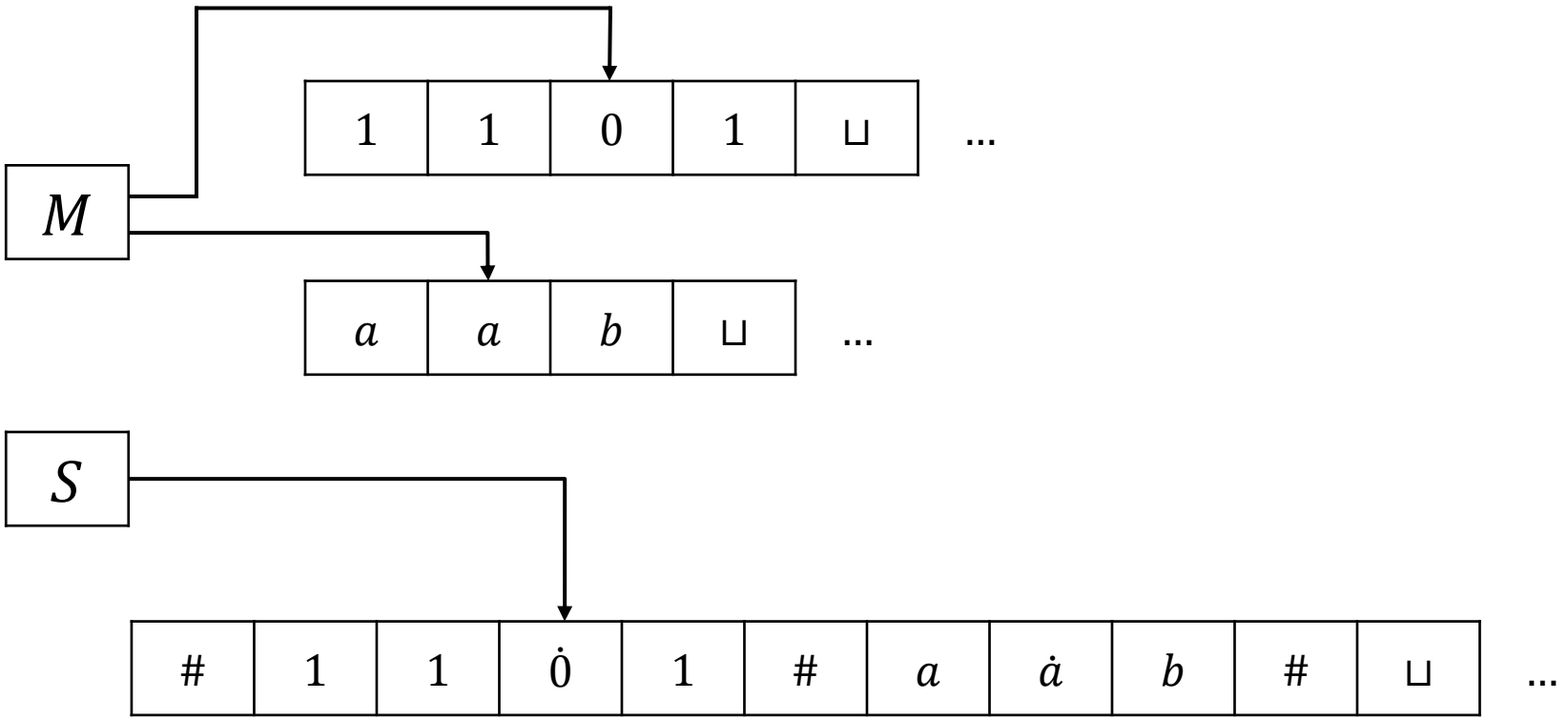
1. Einführung



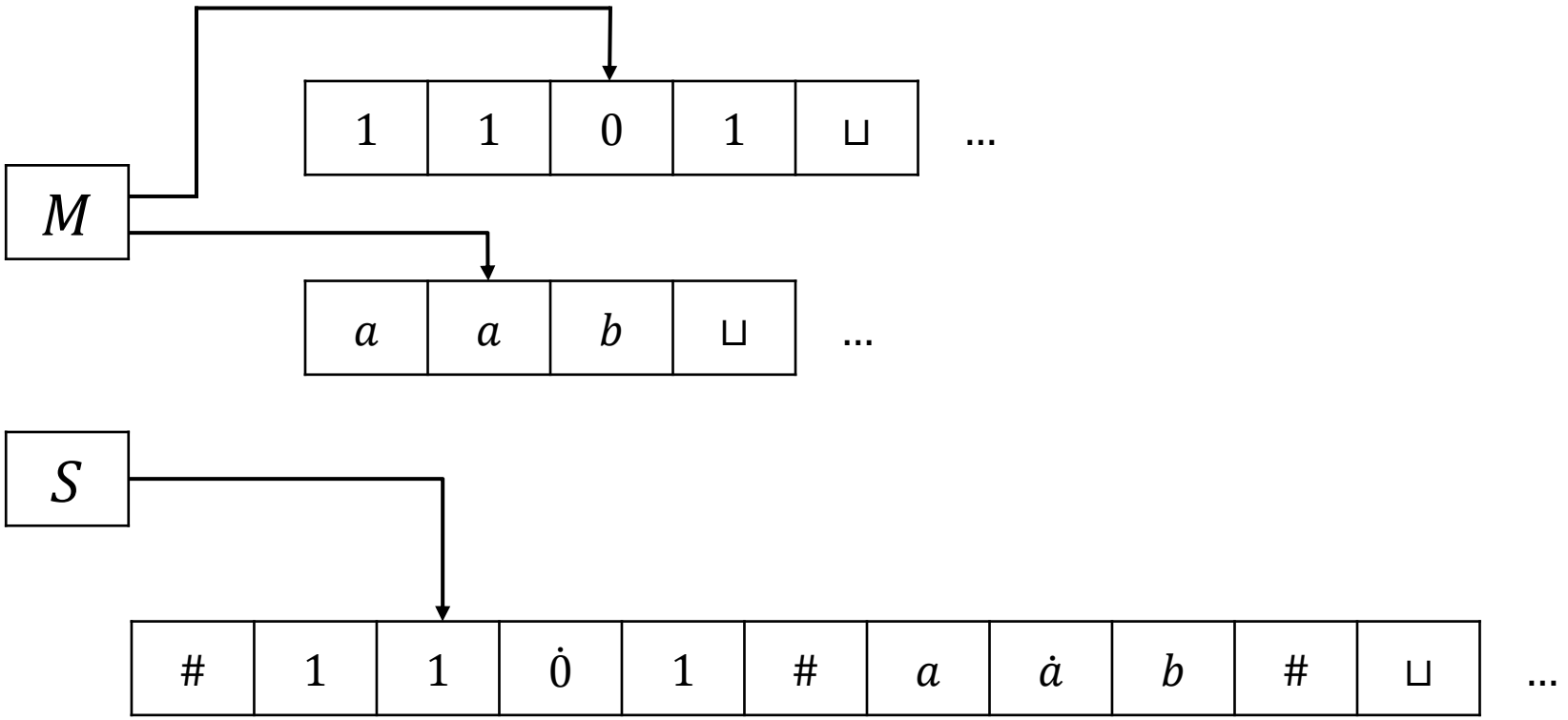
1. Einführung



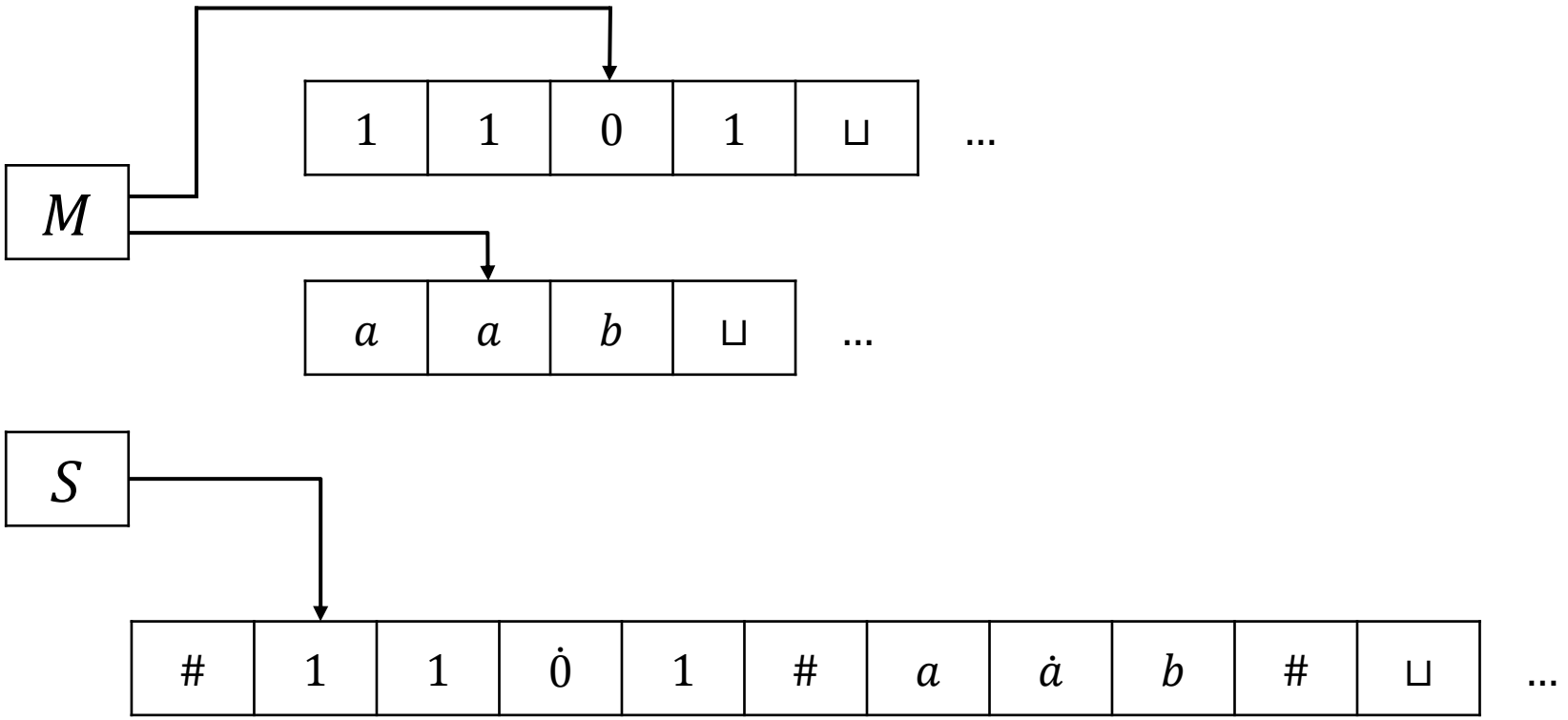
1. Einführung



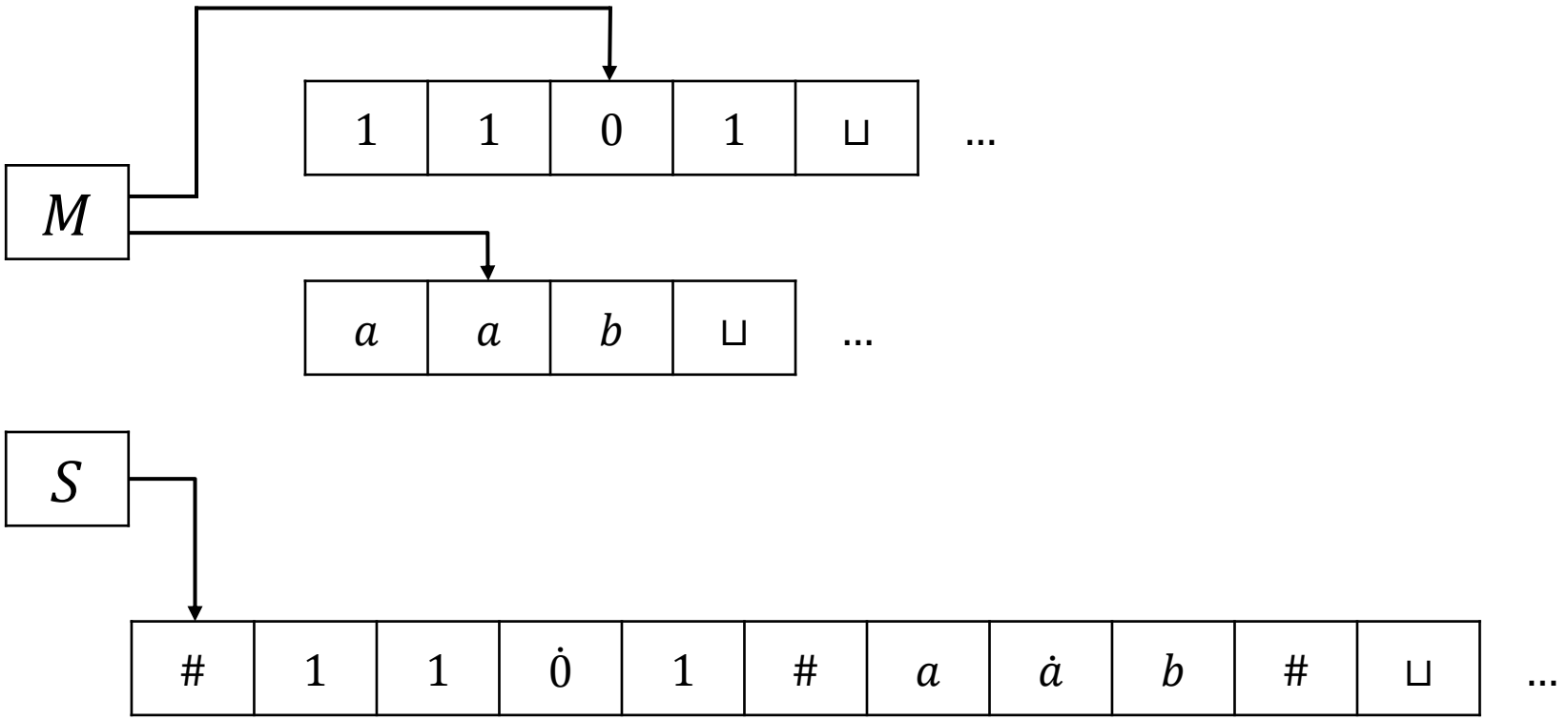
1. Einführung



1. Einführung



1. Einführung



Korollar

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