Zusammenfassung Automaten und formale Sprachen FS 17

# Einleitung

## Grundbegriffe

### Definition 1

Ein **Alphabet** ist eine Menge von Zeichen (Symbolen, Buchstaben). Eine endliche Folge x1x2…xn (wobei die xi aus stammen) wird ein **Wort** der Länge n genannt. Die Länge eines Wortes u wird mit |u| bezeichnet.

Für jedes Alphabet existiert ausserdem das leere Wort der Länge 0, welches mit bezeichnet wird.

Für jedes Alphabet schreiben wir \* für die Menge aller möglichen Wörter über . Ausserdem sei + die Menge aller möglichen Wörter ohne das leere Wort.

### Definition 2

Zwei Wörter sind genau dann **gleich** wenn sie gleich lang sind und das i-te Zeichen des einen Wortes immer gleich dem i-ten Zeichen des anderen Wortes ist.

Für Wörter u= x1x2…xn und v = y1y2…ym ist uv := x1x2…xny1y2…ym als die **Konkatenation** definiert. Die Konkatenation von u und dem leeren Wort gibt wieder u.

Die n-te Potenz un eines Wortes u ist definiert als u0 := und un+1 = unu

#### Bemerkung

Für jedes Alphabet gilt u(vw)=(uv)w (Kommutativität).

### Definition 3

Ein Wort u heisst **Teilwort** eines Wortes v, genau dann wenn es Wörter w1 und w2 gibt, so dass v=w1uw2 gilt. Ist w1=, nennen wir u ein Anfangsstück von v, ist w2= ist u ein Endstück von v.

Das **Spiegelbild** sp(u) eines Wortes u ist definiert durch sp() := und sp(x1x2…xn) := xn…x2x1 . Wörter mit der Eigenschaft sp(u)=u werden **Palindrome** genannt. ^

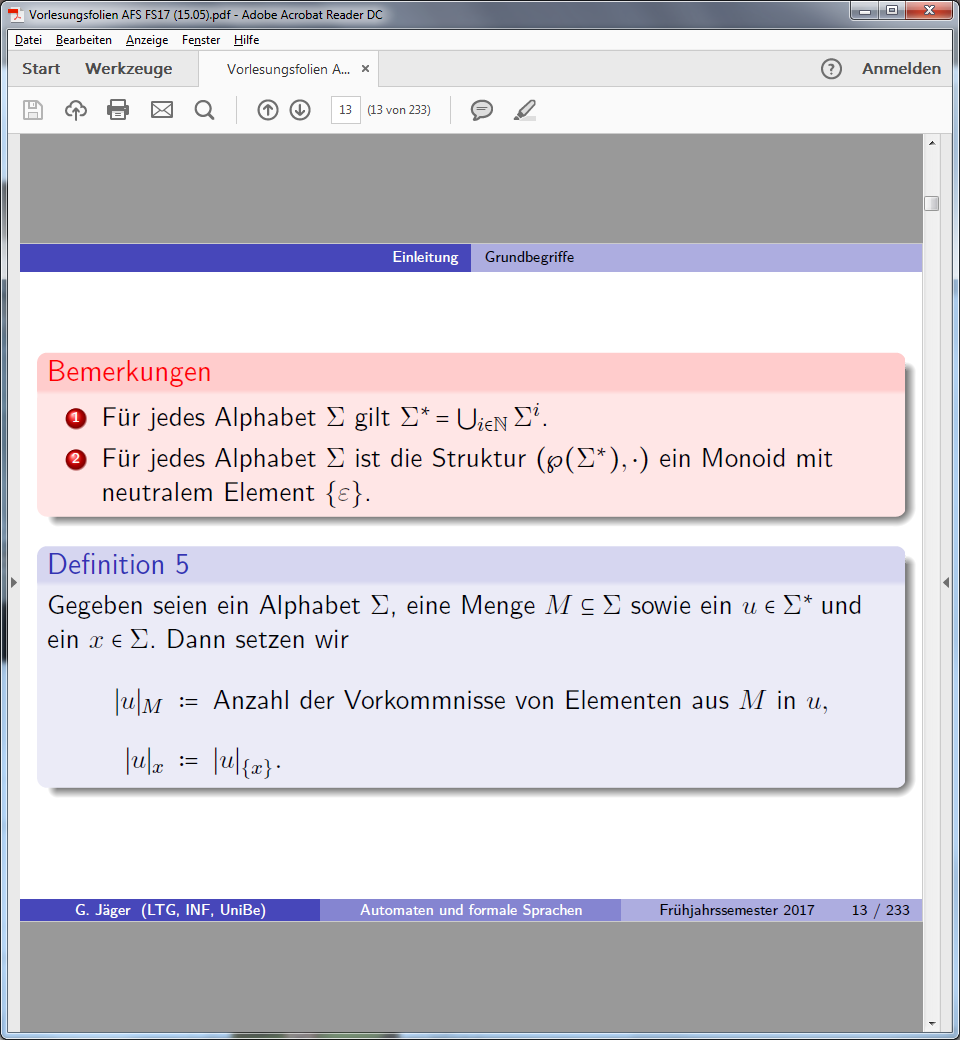
### Definition 4

Gegeben das Alphabet

Sind M und N Teilmengen von \*, so ist das **Produkt** MN definiert durch MN := {uv: u aus M und v aus N}. Also alle möglichen Kombinationen von Wörtern aus M und N. Häufig schreibt man für dieses Produkt nur MN. Für mehrere Teilmengen Mi von \* gilt ausserdem: M1M2…Mn := (…(M1M2)…Mn)

Für jede Teilmenge M von \* und jede natürliche Zahl n ist die n-te Potenz induktiv definiert als   
M0 :={} und Mn+1M. Ausserdem setzen wir M\* := (i eine natürliche Zahl) und M+:=M\* ohne das leere Wort.

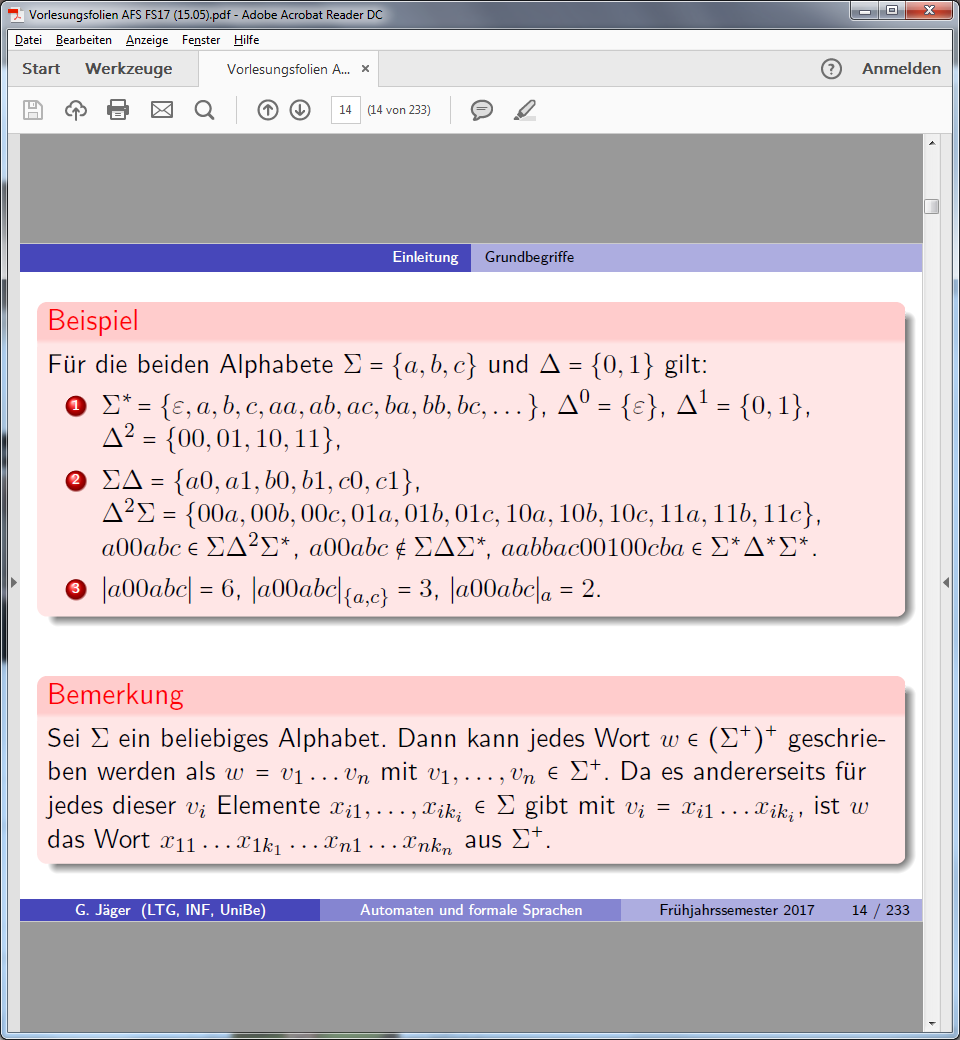
#### Bemerkung



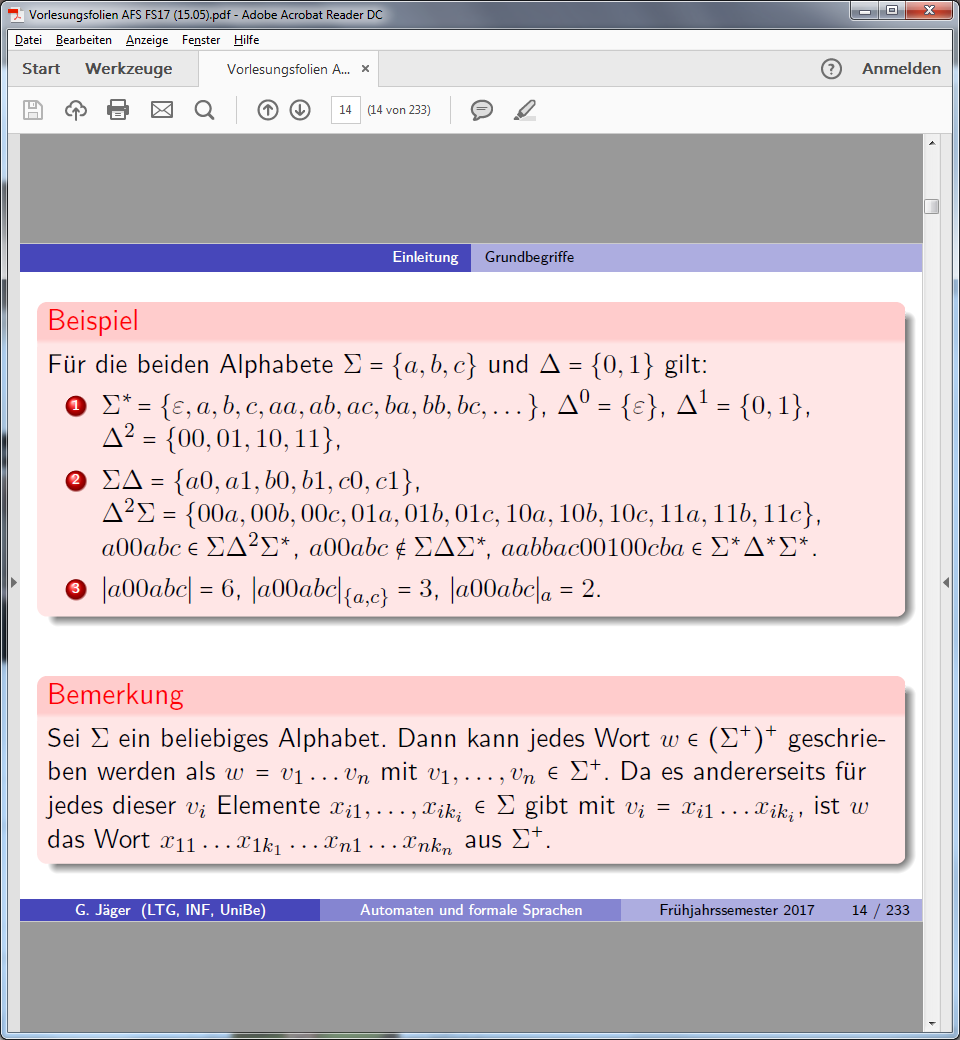
### Definition 5

Gegeben seien ein Alphabet , eine Teilmenge M davon, sowie ein Wort u aus \* und ein Symbol x aus . Dann setzen wir |u|M := Anzahl der Vorkommnisse von Elementen (Symbolen) aus M in u und |u|x := Anzahl der Vorkommnisse von x in u.

#### Beispiel



#### Bemerkung



### Lemma 6

Für jedes Alphabet und alle M,N,N1,N2 Teilmengen von \* gilt

und \*=(M\*N)\*M\*.

### Definition 7

Gegeben sei ein Alphabet . Dann bezeichnet man jede Teilmenge L von \* als **formale Sprache** über .

# Endliche Automaten

## Allgemeines

Ein endlicher Automat ist ein Automat, der sich nur in endlich vielen verschiedenen Zuständen befinden kann. Sehr schematisch kann man einen Automaten in drei Komponeneten untergliedern: Die **Eingabe**, womit dem Automaten von aussen Informationen geliefert werden (z.B. Geldeinwurf). Den **Zustandsraum**: Ein Automat befindet sich zu jedem Zeitpunkt in einem bestimmten Zustand. Als Reaktion auf eine Eingabe kann er dann etwa eine bestimmte Sequenz von Zuständen durchlaufen. Die **Ausgabe**, welche ein Text oder Waren sein kann. Eine Eingabe muss aber nicht zwingend zu einer Ausgabe führen.

Deterministisch heisst, dass der Automat von einem Zustand mit einem Input nur in einem weiteren Zustand landen kann.

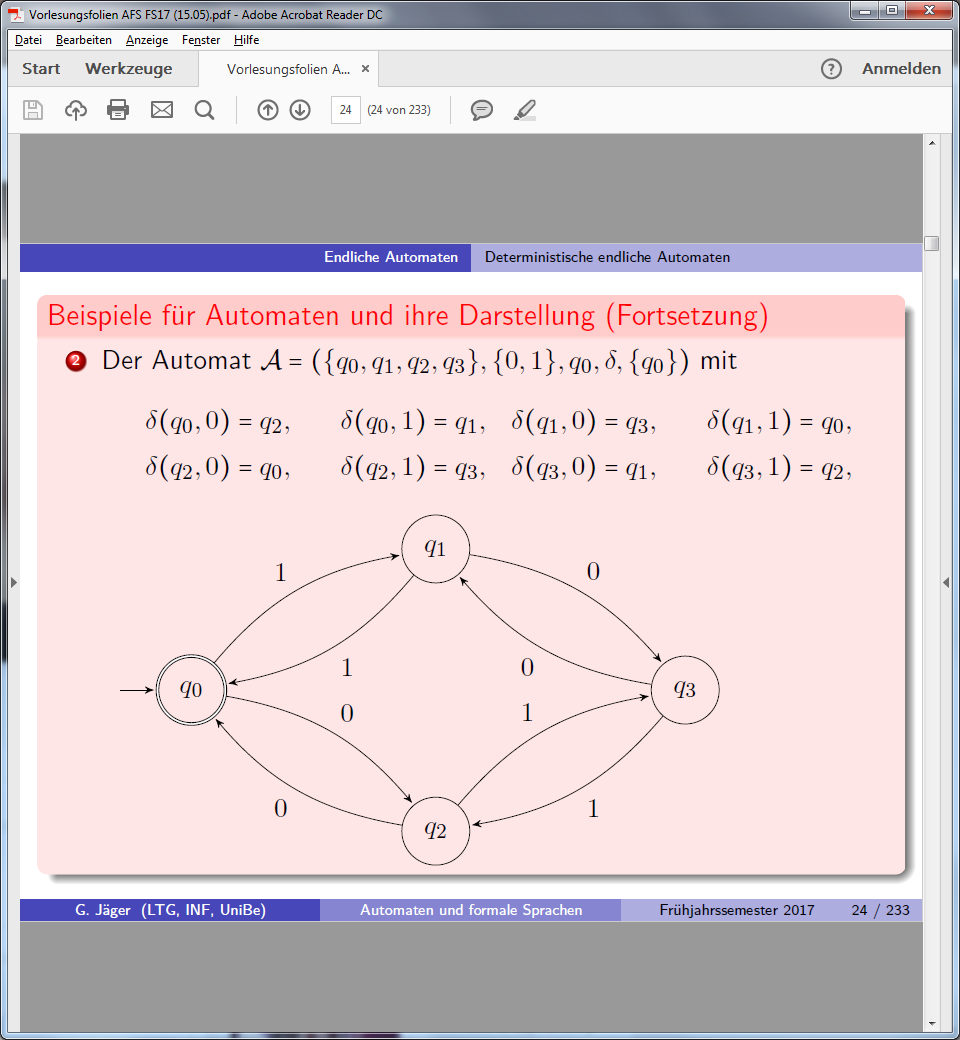
## Deterministische endliche Automaten

Ein **deterministischer endlicher Automat (DEA)** ist ein 5-Tupel A = (Q, , q0, ,F).   
Q ist eine nicht-leere endliche Menge, die **Zustandsmenge**  
 ist eine nicht-leere endliche Menge, das **Eingabealphabet**  
q0 aus Q ist der **Anfangszustand**  
 : Q x -> ist die **Transitionsfunktion**, die von einem Zustand mit einem Eingabeelement in einen Zustand überführt.   
F Q ist die Menge der **Endzustände**.

Der Zustand q‘ = (q,a) nennt man den **Nachfolgezustand von q unter a**. Befindet sich ein Automat im Zustand q und liest das Eingabezeichen a, geht er in den Zustand q‘. Vor dem ersten Einlesen befindet sich der Automat im Startzustand q0.

### Graphische Darstellung von DEAs durch Transitionsgraphen

Knoten repräsentieren Zustände, beschriftete Kanten Übergänge und der Anfangszustand ist der Knoten mit eingehender Knoten ohne Quelle, die Endzustände sind doppelt umkreist.



### Definition 9

Gegeben sei der DEA A= (Q, , q0, ,F). Durch Induktion nach dem Aufbau der Wörter über dem Alphabet definieren wir nun die Erweiterung \*: Q x\* -> Q

Das Wort wird also von links nach rechts, Schritt für Schritt, abgearbeitet und jeweils der allfällige Zustandswechsel vollzogen.

Für ein DEA gilt offensichtlich \*(q,a) = (q,a).

### Lemma 10

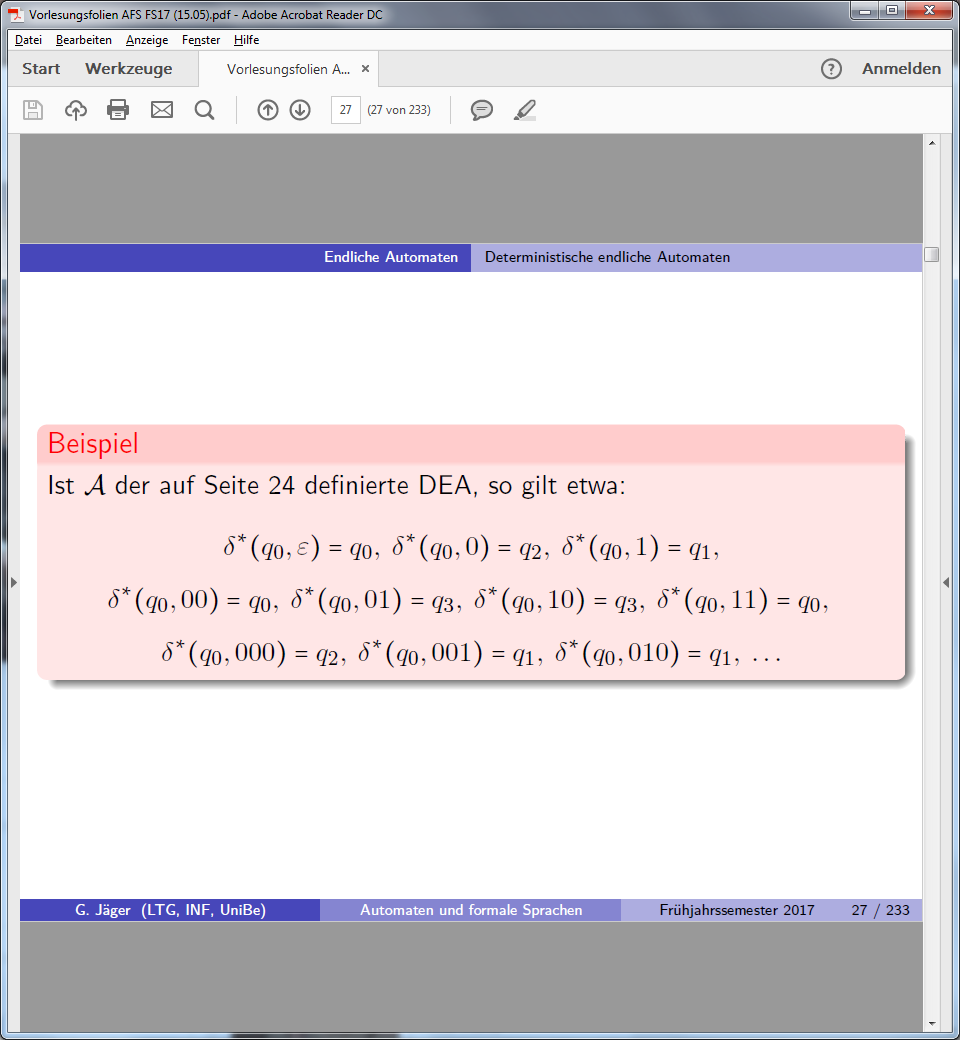
Sei A= (Q, , q0, ,F) ein DEA.

Ist a1…an ein Wort aus \* und ist q1,…qn die Folge von Zuständen, die definiert ist durch   
qi+1:=(qi, ai+1), dann gilt: \*(q0, a1…an) = qn

Für alle q aus Q und alle u,v aus \* gilt

Also wird zuerst das Wort u, dann das Wort v abgearbeitet.

#### Beispiel



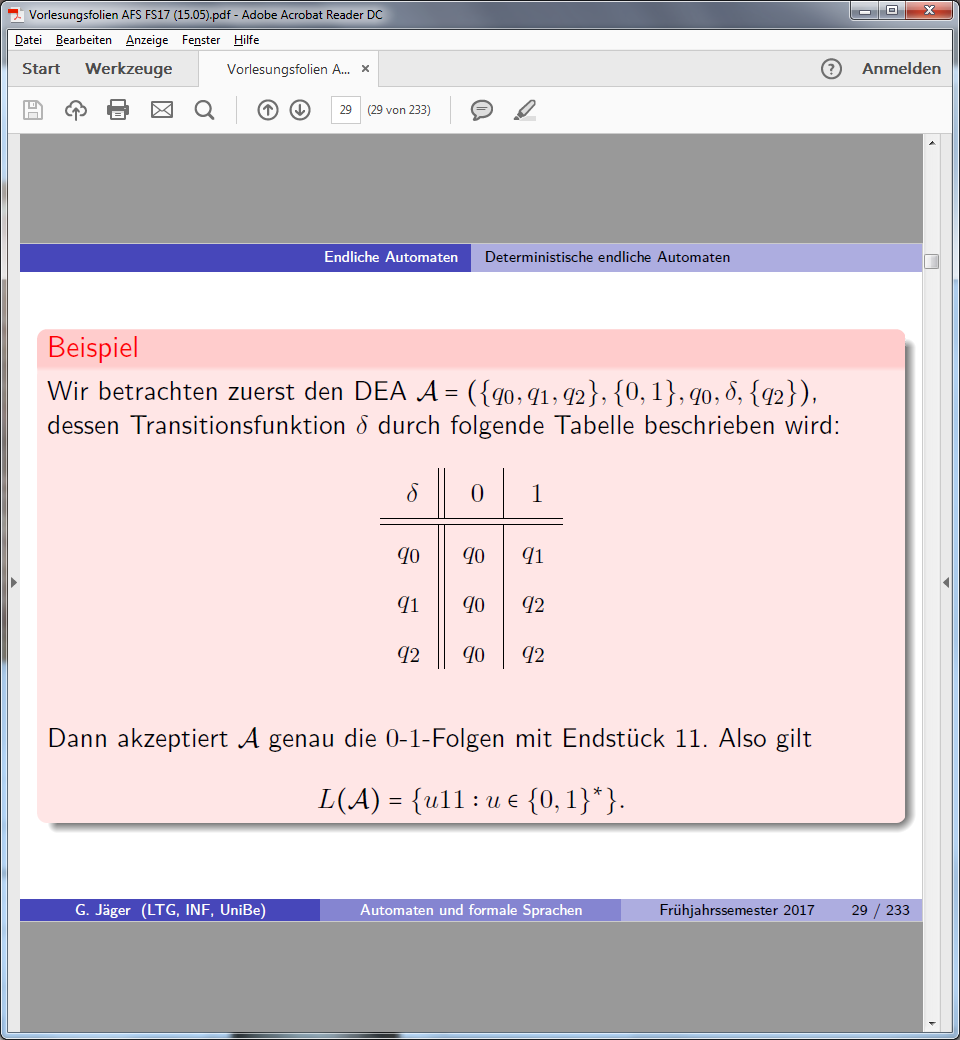
### Definition 11

Es sei A= (Q, , q0, ,F) ein DEA. Dann sagen wir, dass A das Wort u aus \* genau dann **akzeptiert**, wenn \*(q0, u) ein Endzustand ergibt. Also wenn der Automat nach dem Abarbeiten des Wortes sich in einem Endzustand befindet.

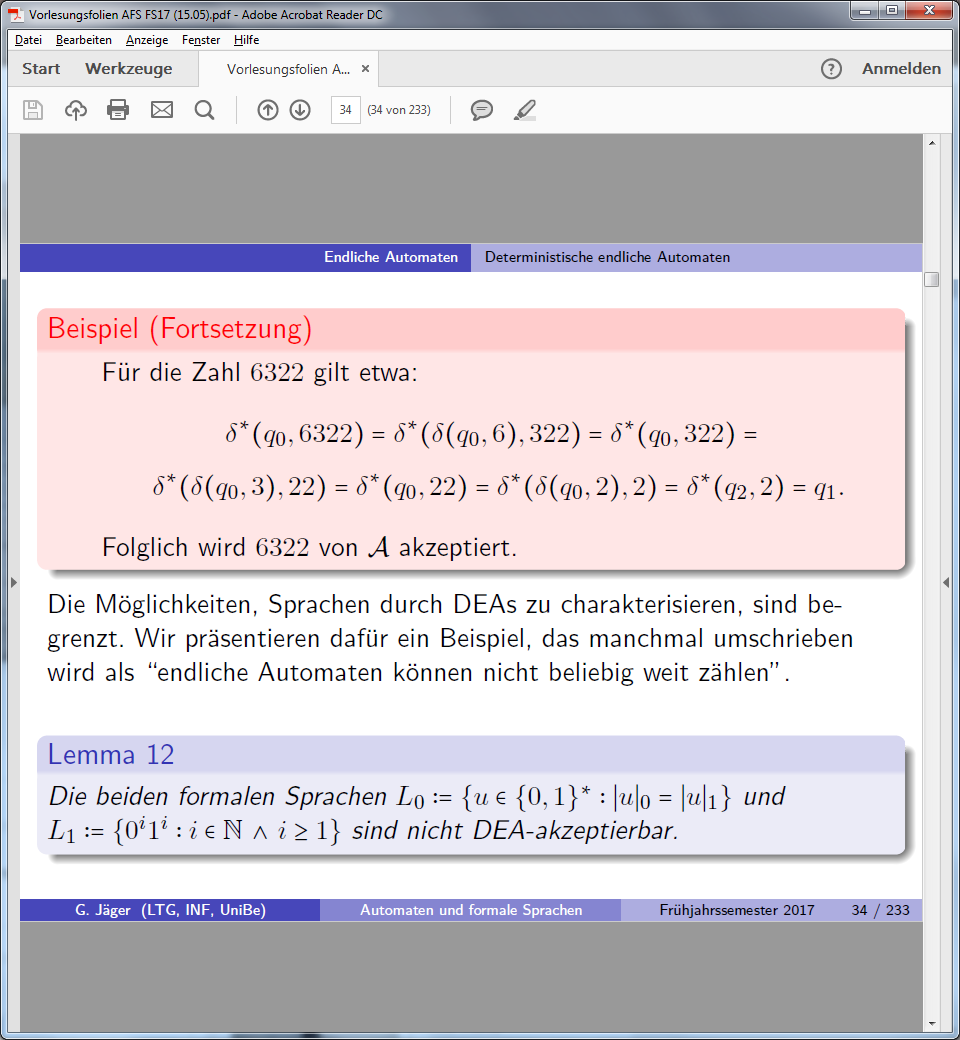
Die von einem DEA A **akzeptierte Sprache L(A)** ist definiert als alle Wörter, die der DEA akzeptiert.

Eine Sprache L heisst genau dann **DEA-akzeptierbar**, wenn es einen DEA A gibt mit L=L(A).

#### Beispiel



### Lemma 12



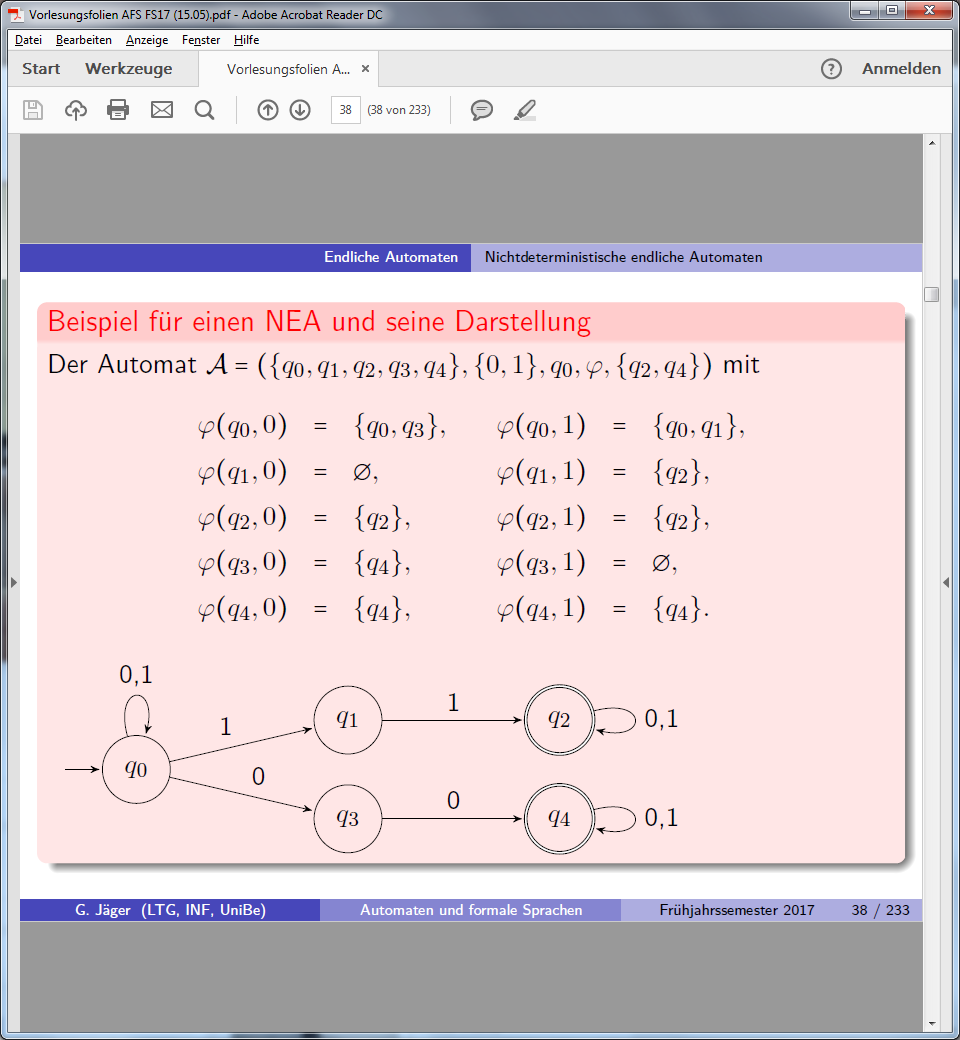
## Nichtdeterministische endliche Automaten

Im Gegensatz zum DEA kann ein Automat hier bei beliebigen Zuständen für einen Input in mehrere Zustände wechseln.

### Definition 13

Ein **nichtdeterministischer endlicher Automat (NEA)** ist ein 5-Tupel A = (Q, , q0, ,F), wobei der Unterschied zum DEA bei der Transitionsfunktion liegt:

: Q x -> p(Q), wobei p(Q) die Potenzmenge (alle möglichen Mengen formbar aus Q) von Q ist. (q, a) = ist dabei zugelassen.



### Definition 14

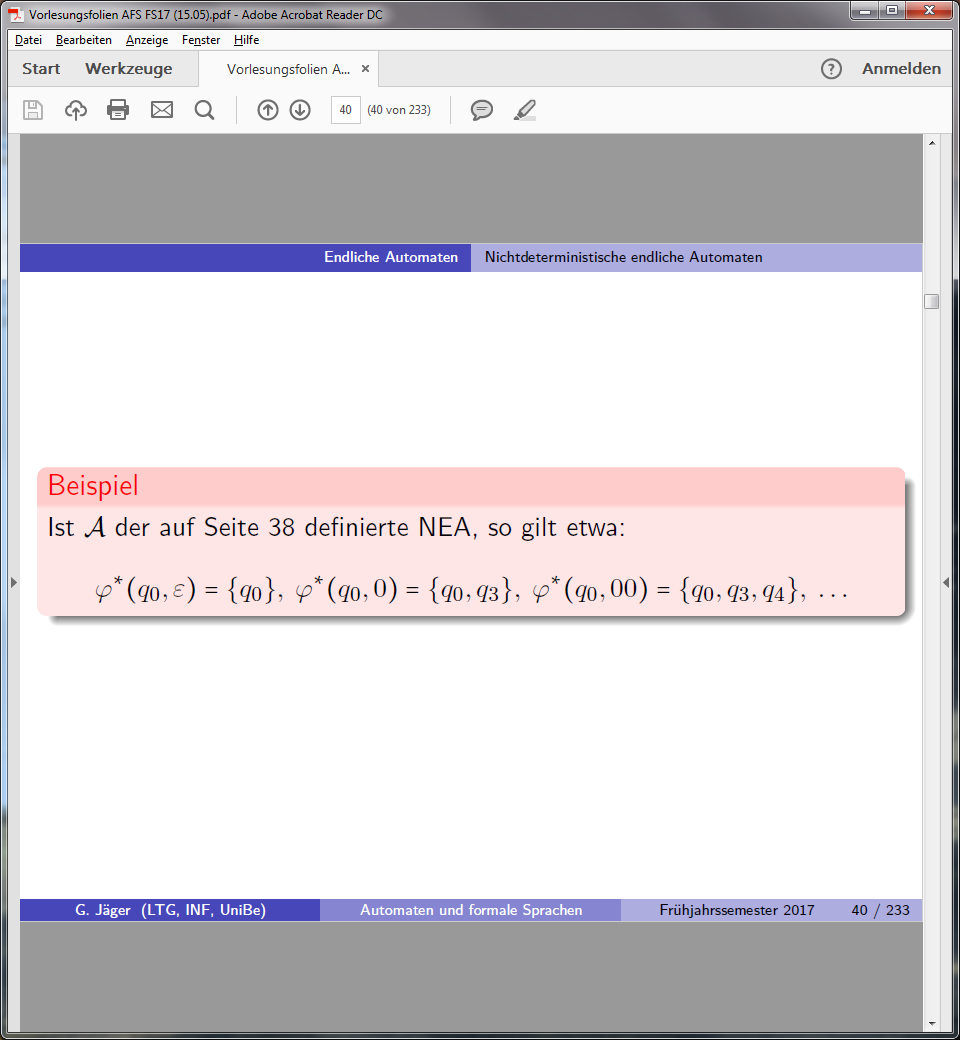
Wir erweitern nun die Transitionsfunktion induktiv auf Wörter, ähnlich wie beim DEA.

Dabei ist \*(q,) := {q} und \*(q,ua) := wobei q aus Q, u aus \* und a aus ist. Ausgeschrieben wird für jeden Zustand aus der Zustandsmenge, die nach u erreicht/erreichbar ist geschaut, in welchen Zuständen der Automat nach a landet und dann diese Menge vereinigt.

Intuitiv beschreibt diese Menge dann alle Zustände, in der der Automat nach dem Abarbeiten von links nach rechts landen kann.

#### Beispiel

Mit dem oben definierten NEA würde dann gelten



### Definition 15

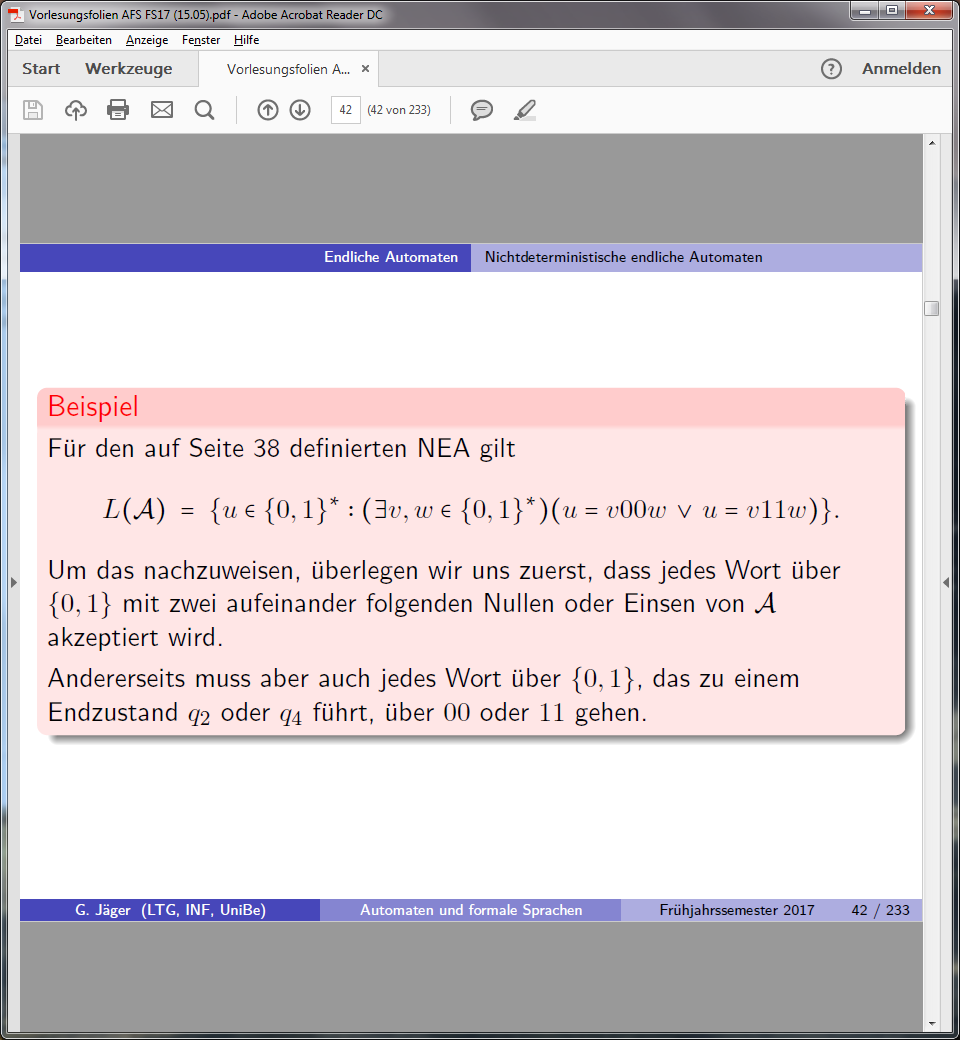
Sei A = (Q, , q0, ,F) ein NEA. Dann sagen wir, dass A das Wort u aus \* genau dann **akzeptiert**, wenn \*(q0, u) F . Also wenn einer der Zustände aus der Zustandsmenge, die nach u erreichbar ist ein Endzustand ist.

Die vom NEA A **akzeptierte Sprache L(A)** ist dann die Menge der akzeptierten Wörter.

Eine Sprache heisst genau dann **NEA-akzeptierbar**, wenn es einen NEA A mit L = L(A) gibt.

#### Beispiel

Mit dem oben definierten NEA würde dann gelten



### Lemma 16

Jeder DEA kann auch als NEA verstanden werden und für jeden NEA gibt es einen DEA, der dieselbe formale Sprache akzeptiert.

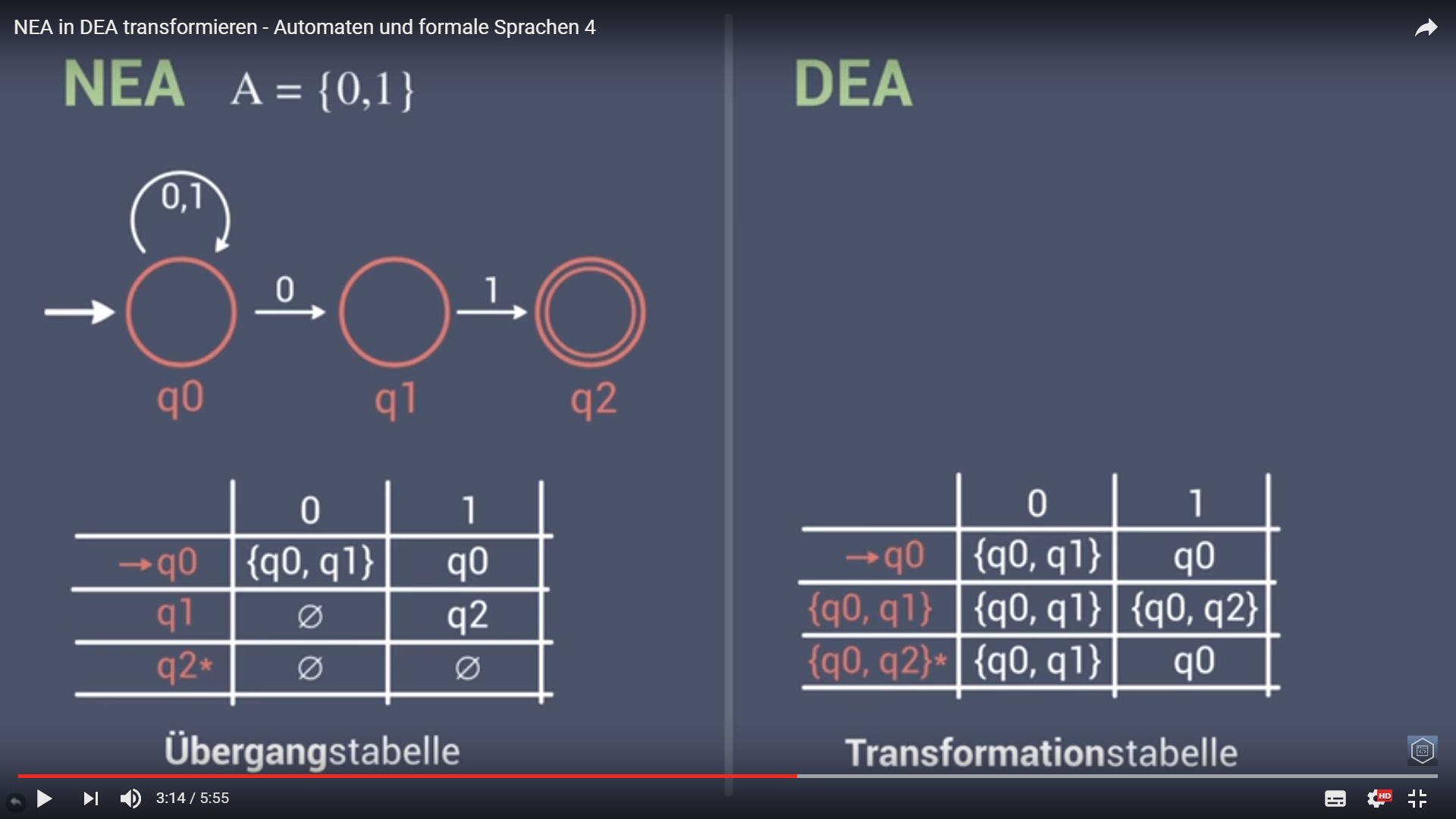
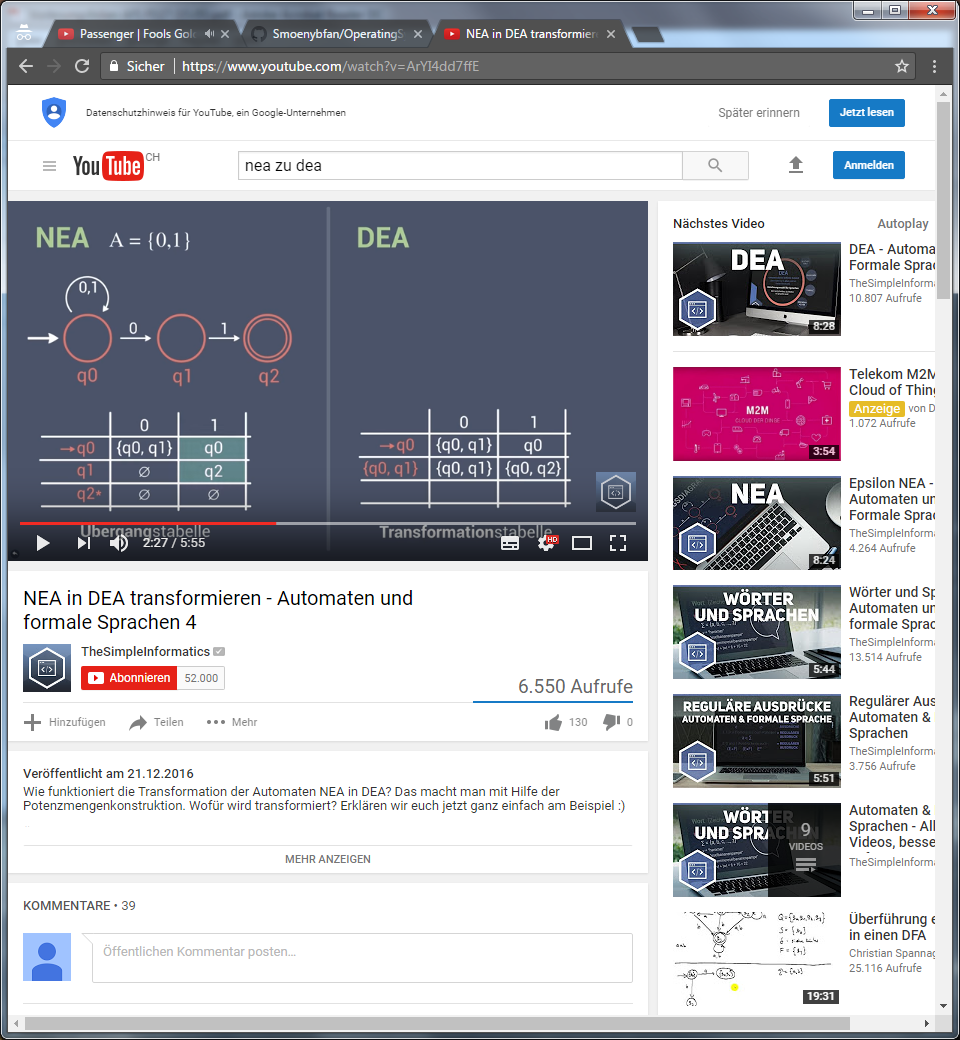
Sei A := (Q, , q0, ,F) ein beliebiger DEA. Dann definieren wir : Q x -> p(Q), (q, a) :={}, so ist B := (Q, , q0, ,F) ein NEA, für den gilt \*(q, u) = {\*(q, u)}. Weiter gilt L(A) = L(B).

### Theorem 17

Für jeden NEA A gibt es einen DEA B, mit L(A) = L(B), also einen DEA, der dieselbe Sprache akzeptiert.

Diesen DEA finden wir über die Potenzmengenkonstruktion. Wir übernehmen den Startzustand des NEA. Dann schauen wir, in welchen Zuständen, beziehungsweise Mengen von Zuständen der NEA vom Startzustand aus landet. Die Einträge, die so entstanden sind werden zu Zuständen und für diese wird dann wieder geprüft, in welchen Zuständen der NEA (für jeweils alle Zustände der Menge) landet, usw.

Im Grunde genommen wird für jede Teilmenge der Potenzmenge betrachtet, in welchen Mengen von Zuständen sie landet. Der oben beschriebene Weg lässt bereits einige unnötige Zustände weg (betrachte Bemerkung).

#### Bemerkung

Hat ein NEA n Zustände, hat der konstruierte äquivalente Automat 2n Zustände. Viele dieser Zustände sind aber nicht notwendig, da sie nicht erreichbar sind und können weggelassen werden. Das ist im obigen Beispiel schon geschehen, z.B. {q0,q1,q2} ist kein erreichbarer Zustand des NEA.

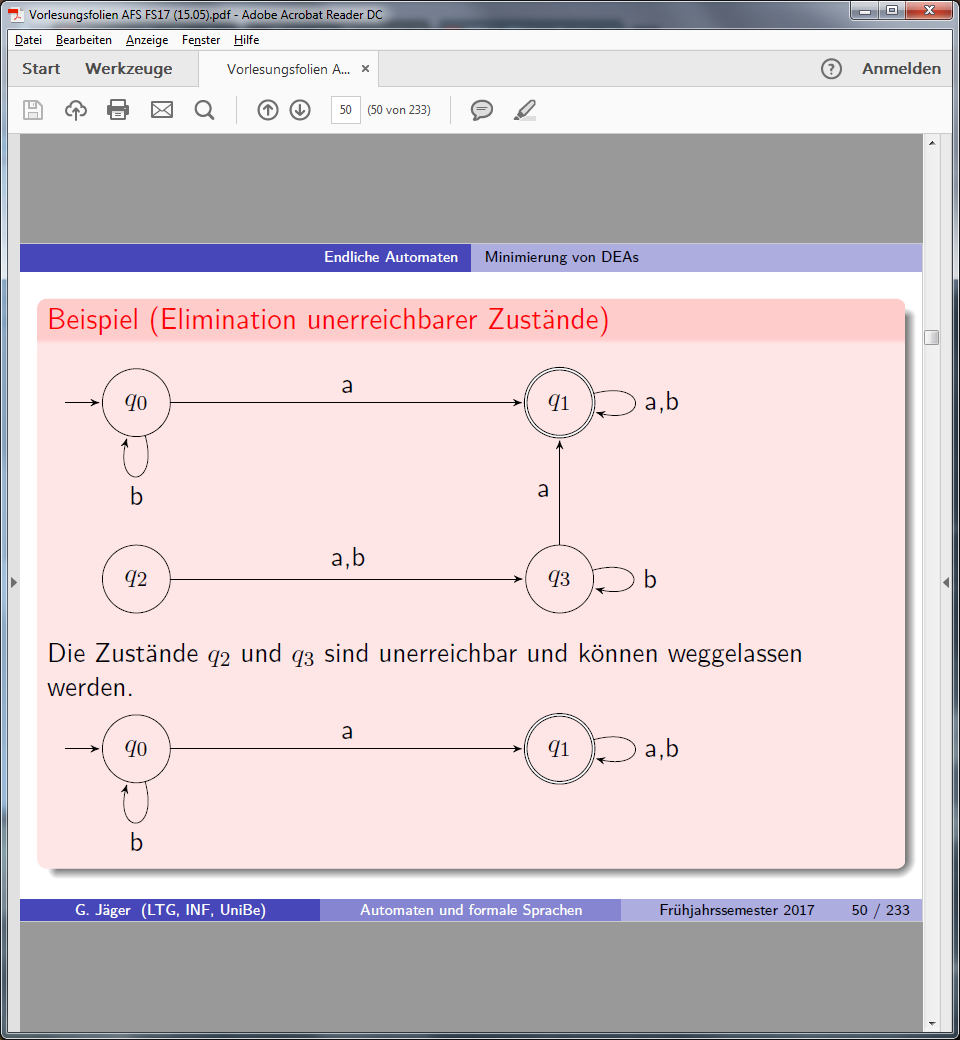
## Minimierung von DEAs

Wir wollen nun einen DEA in einen DEA transformieren, der mit möglichst wenigen Zuständen auskommt. Dafür werden unerreichbare Zustände eliminiert und eine Identifizierung von Zuständen durchgeführt.

### Schritt 1

Wir bestimmen im gegebenen DEA alle Zustände, die vom DEA erreichbar sind. Alle unerreichbaren Zustände können dann gestrichen und die Transitionsfunktion entsprechend angepasst werden.

#### Beispiel

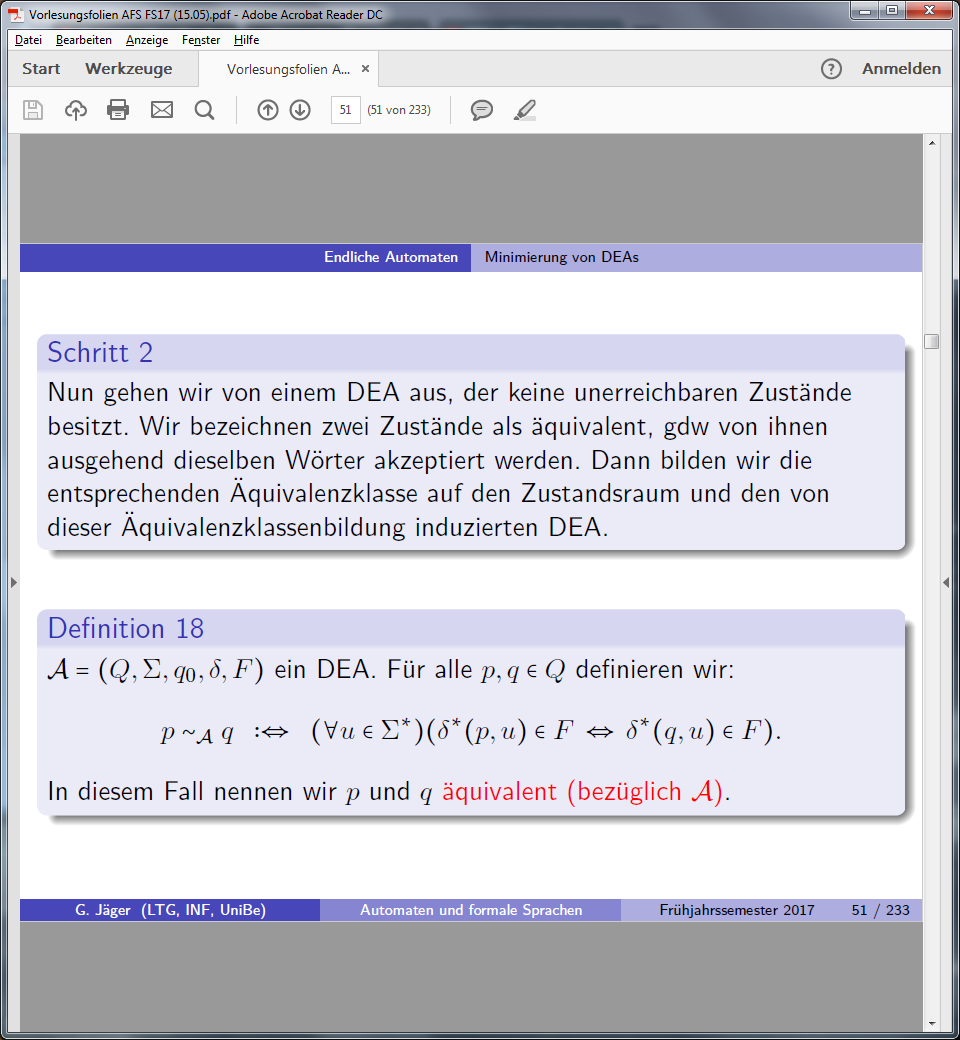


### Schritt 2

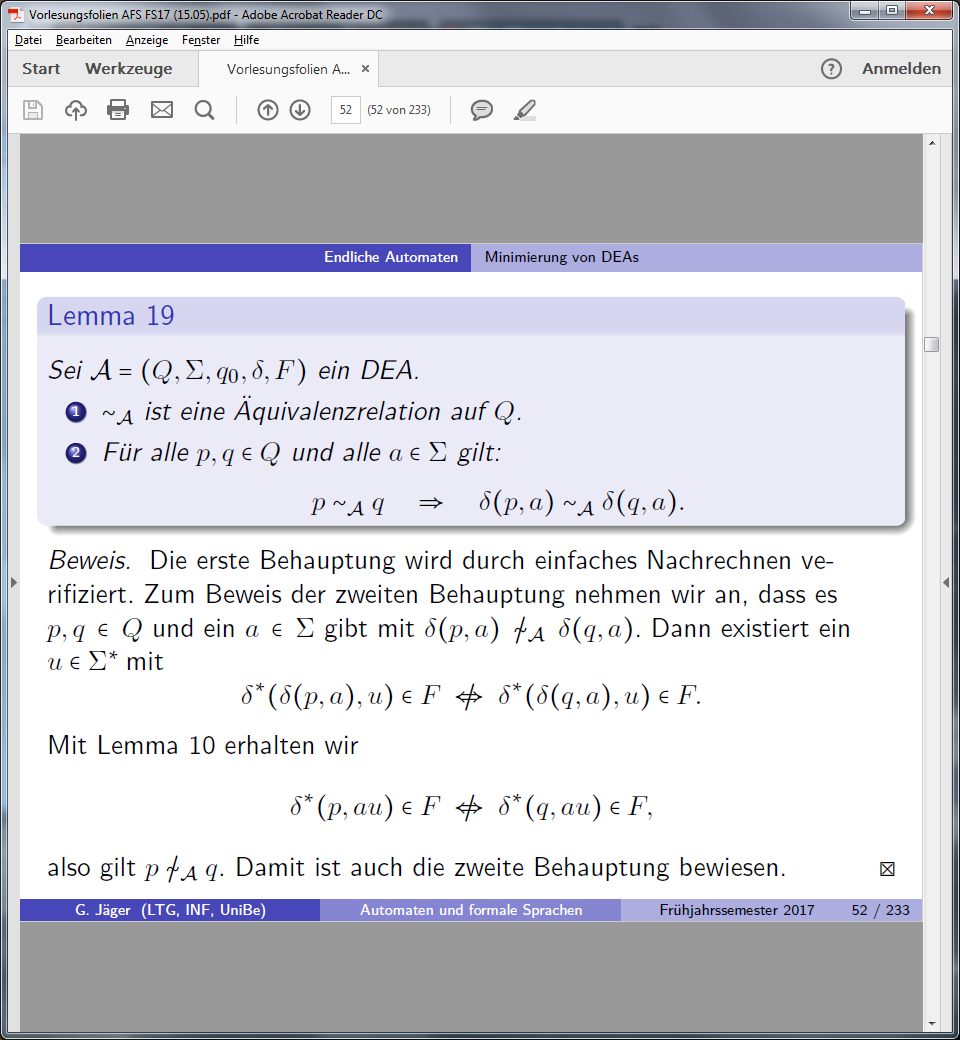
Wir gehen nun von einem DEA aus, der keine unerreichbaren Zustände mehr besitzt. Wir bezeichnen zwei Zustände genau dann als äquivalent, wenn von ihnen ausgehend dieselben Wörter akzeptiert werden. Wir können dann die Äquivalenzklassen zu jeweils einem Zustand zusammenfassen.

### Definition 18

Sei A := (Q, , q0, ,F) ein DEA. Zwei Zustände p,q aus Q sind genau dann **äquivalent (bez. A)**, wenn gilt:

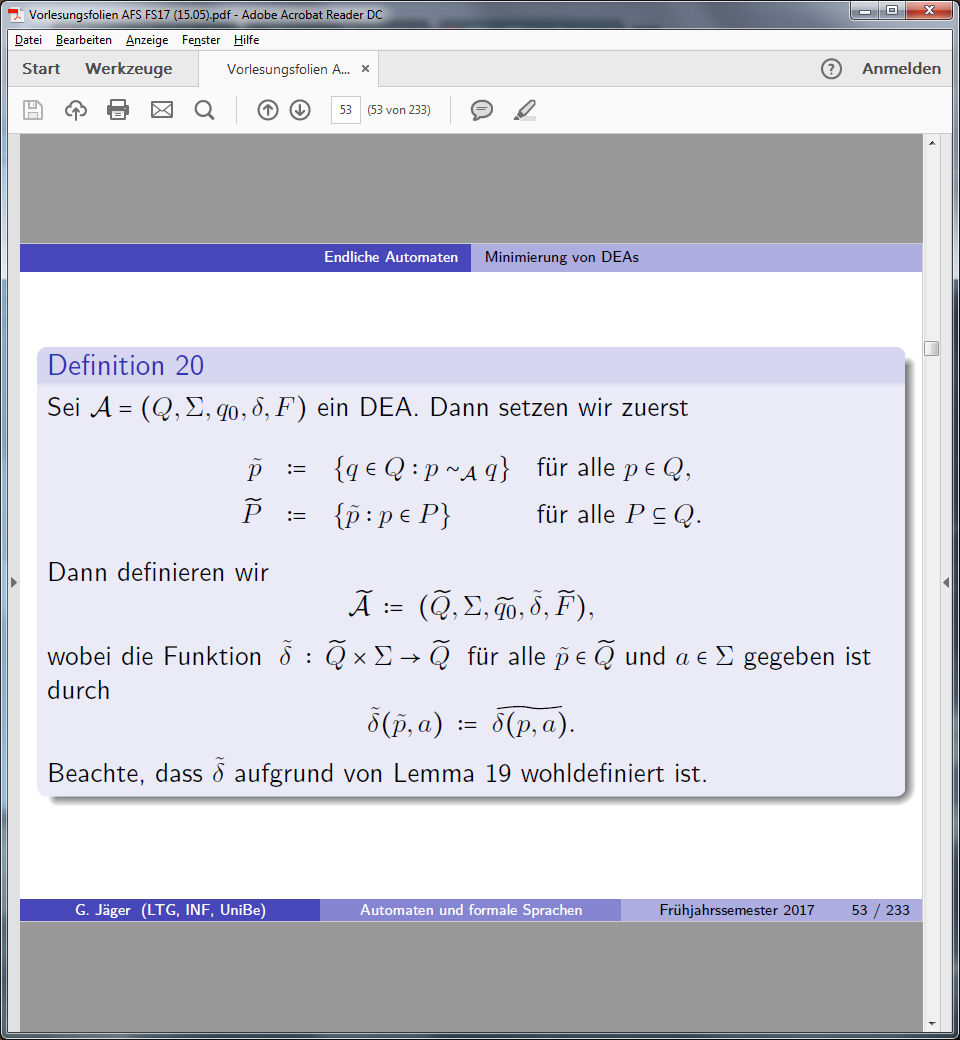


### Lemma 19



2 besagt, dass wenn zwei Zustände äquivalent sind, es auch der Zustand ist, der von beiden mit dem gleichen Eingabesymbol erreichbar ist.

### Definition 20



Wir sehen, dass die Menge der zu p äquivalenten Zustände sind. ist dann die Menge der Zustände, die zu den Elementen von P äquivalent sind.