Theoretische Grundlagen des Software Engineering

1: Grundlagen, Sprachen, Automaten

Stephan Schulz
schulz@eprover.org

Software
Systems
Engineering

Definition

Eine Definition ist eine genaue Beschreibung eines Objektes oder Konzepts.

- Definitionen können einfach oder komplex sein
- Definitionen müssen präzise sein - es muss klar sein, welche Objekte oder Konzepte beschrieben werden
- Oft steckt hinter einer Definition eine Intuition - die Definition versucht, ein "reales" Konzept formal zu beschreiben
  - Hilfreich für das Verständnis - aber gefährlich! Nur die Definition an sich zählt für formale Argumente

Mathematische Beweise

Ein Beweis ist ein Argument, das einen *verständigen* und *unvoreingenommenen* Empfänger von der *unbestreitbaren Wahrheit* einer Aussage überzeugt.

- Oft mindestens semi-formell
- Aussage ist fast immer ein Konditional
  - ...aber die Annahmen sind für semi-formelle Beweise oft implizit

Mengen

Definition: Eine Menge ist eine Sammlung von Objekten, betrachtet als Einheit

- Die Objekte heißen auch Elemente der Menge.
- Elemente können beliebige Objekte sein:
  - Zahlen
  - Worte
  - Andere Mengen (!)

Mengen (2)

Schreibweise:

- Aufzählung:
  - A = {2, 3, 5, 7, 11, 13}
  - **N** = {1, 2, 3, ...}
- Beschreibung: A= {x | x ist Primzahl und x≤13}
- Mengenzugehörigkeit
  - 2∈A (2 ist in A, 2 ist Element von A)
  - 4∉A (4 ist nicht in A, 4 ist keine Element von A

Mengen (3)

Schreibweise:

- Teilmengen
  - $\{2,5,7\} \subseteq A$ {...ist Teilmenge von...}
  - $\{2, 5, 7\} \subsetneq A$ {...ist echte Teilmenge von...}
  - $\{2,5,7\} \subset A$){...ist (echte) Teilmenge von...}
  - Analog: Obermengen mit $\supset$, $\supseteq$, $\supsetneq$
  - $\{1, 2, 3\} \not\subseteq A$ (...ist keine Teilmenge...)

Mengen (4)

- Vereinigung: $A \cup B = \{x \mid x \in A \text{ oder } x \in B\}$
- Schnitt: $A \cap B = \{x \mid x \in A \text{ und } x \in B\}$
- Subtraktion: $A \setminus B = \{x \mid x \in A \text{ und } x \notin B\}$
- {} ist die leere Menge. Schreibweise auch $\emptyset$.
  - $\emptyset \subseteq M$ für jede beliebige Menge M!
- $\mathbf{N} = \{1, 2, 3, ...\}$ ist die Menge der natürlichen Zahlen
- $\mathbf{N_0} = \{0, 1, 2, 3, ...\} = \mathbf{N} \cup \{0\}$
- $\mathbf{Z} = \{...-2, -1, 0, 1, 2, ...\}$ ist die Menge der ganzen Zahlen

Mengen (5)

Definition: |A| ist die Kardinalität einer Menge A

- Für endliche Mengen: |A| ist die Anzahl der Elemente in A

Beispiele:

- $|\emptyset|=0$
- $|\{a, b, c\}| = 3$
- $|\mathbf{N}|$ behandeln wir woanders ;-)

Potenzmengen

Definition: Die Potenzmenge einer Menge A ist die Menge aller ihrer Teilmengen

- Schreibweise: 2^A

Beispiele:

- $2^{\emptyset} = \{\emptyset\}$
- $2^{\{1,2,3\}}=\{\emptyset, \{1\},\{2\},\{3\}, \{1,2\}, \{1,3\}, \{2,3\},\{1,2,3\}\}$

Tupel

Definition: Ein Tupel eine Sequenz von Elementen

- T = (1, 2, 3)
- Ein Tupel mit n Elementen heißt n-Tupel
  - T ist ein 3-Tupel oder Tripel
  - (1,2) ist ein 2-Tupel oder Paar
  - $A \times B = \{(a,b) | a \in A, b \in B\}$ ist die Menge aller 2-Tupel über A,B
    - $A \times B$ heißt das kartesische Produkt von A und B
    - Analog: $A \times B \times C$ (...)

Tupel-Beispiele

Sei A={1,2,3}, B={a,b,c}

- A×A = {(1,1),(1,2),(1,3), (2,1), (2,2), (2,3), (3,1),(3,2),(3,3)}
- A×B = {(1,a),(1,b),(1,c),(2,a),(2,b),(2,c),(3,a),(3,b),(3,c)}
- B×A = {(a,1),(a,2),(a,3),(b,1),(b,2),(b,3),(c,1),(c,2),(c,3)}
- Insbesondere A×B ≠ B×A
- C={(x,y)∈A×A|x<y} ={(1,2),(1,3),(2,3)}
  - C ist die <-Relation, eingeschränkt auf A

Relationen

Definition:

- Eine n-stellige Relation R ist eine Menge von n-Tupeln
  - Also $R \subseteq A_1 \times ... \times A_n$
  - Ist $(a_1, ..., a_n) \in R$, so schreiben wir $R(a_1, ..., a_n)$
- Gilt $A_1 = A_2 = ... = A_n$, so heißt R homogen
- Der Begriff "Relation" wird oft mit der eingeschränkten Bedeutung "2-stellige Relation" ("binäre Relation") verwendet.

Binäre Relationen

Definition:

- Eine (binäre) Relation R ist eine Menge von Paaren
  - Falls (a,b)∈R: "a und b stehen in Relation R"
  - Statt R(a,b) schreiben wir auch aRb, falls (a,b)∈R.

Relationen - Beispiele

- $= \subseteq \mathbf{N}\times\mathbf{N}$ ist eine binäre Relation über den natürlichen Zahlen
  - $= = \{(1,1),(2,2),(3,3),(4,4)...\}$
- $< \subseteq \mathbf{N}\times\mathbf{N}$ ist eine binäre Relation über $\mathbf{N}$
  - Wir verwenden das selbe Symbol für verschiedene Relationen: $<\subseteq\mathbf{Z}\times\mathbf{Z}$, $<\subseteq\mathbf{R}\times\mathbf{R}$, ...
- $P=\{(x,2x)|x \in \mathbf{N}\}=\{(1,2), (2,4), (3,6)...\}$ setzt jede natürliche Zahl in Relation mit ihrer Verdopplung
  - Beachte: $P \subseteq <$

Eigenschaften von Relationen (1)

Definition: Sei $R \subseteq A \times B$ eine binäre Relation

- Gilt $\forall a \in A \exists b \in B: R(a,b)$, so heißt R linkstotal
  - Jedes Element aus A steht in Relation zu mindestens einem Element aus B
- Gilt $\forall a \in A, \forall b,c \in B: R(a,b) \wedge R(a,c) \Rightarrow b=c$, so heißt R

  rechtseindeutig
  - Jedes Element aus A steht in Relation zu höchstens einem Element aus B

Eigenschaften von Relationen (2)

Definition: Sei $R \subseteq A \times A$ eine homogene binäre Relation

- Gilt $\forall a \in A: R(a,a)$, so heißt R reflexiv
- Gilt $\forall a,b \in A: R(a,b) \Rightarrow R(b,a)$, so heißt R symmetrisch
- Gilt $\forall a,b,c \in A: R(a,b) \wedge R(b,c) \Rightarrow R(a,c)$, so heißt R transitiv
- Ist R reflexiv, symmetrisch und transitiv, so ist R eine Äquivalenzrelation

Beispiel: > auf **N**

Betrachte > auf **N**×**N**

- Linkstotal? ✗ ∄b∈**N** mit 1>b
- Rechtseindeutig? ✗ 4>2 und 4>1, aber 1≠2
- Reflexiv? ✗ (1,1)∉<
- Symmetrisch? ✗ 2>1, aber (1,2)∉<
- Transitiv? ✓ a>b und b>c impliziert a>c

Beispiel: ≤ auf **N**

Betrachte ≤ auf **N**×**N**

- Linkstotal? ✓ 1≤a für alle a aus **N**
- Rechtseindeutig? ✗ 1≤1 und 1≤2, aber 1≠2
- Reflexiv? ✓ a≤a für alle a aus **N**
- Symmetrisch? ✗ 1≤2, aber nicht 2≤1
- Transitiv? ✓ a≤b und b≤c impliziert a≤c

Beispiel: Gleichheit

Betrachte = auf **A**×**A**

- Linkstotal? ✓ a=a für alle a aus **A**
- Rechtseindeutig? ✓ a=b und a=c impliziert b=c
- Reflexiv? ✓ a=a für alle a aus **A**
- Symmetrisch? ✓ a=b impliziert b=a
- Transitiv? ✓ a=b und b=c impliziert a=c

Ergo:

- = ist eine Äquivalenzrelation
- = ist eine Funktion (die Identitätsfunktion)

Funktionen (1)

Definition: Eine (totale) Funktion (auch: Abbildung) f von A nach B ist eine linkstotale, rechtseindeutige Relation über A×B

- Schreibweise: $f:A \rightarrow B$
- Wir schreiben $f(a)=b$ oder $f:a \mapsto b$ falls $(a,b) \in f$
- A heißt Definitionsmenge (oder Domäne) von f
- B heißt Zielmenge (oder Wertemenge) von f
- $f(A) = \{f(a) | a \in A\}$ ist die Bildmenge (oder das Bild) von f

Funktionen (2)

Beachte: Die Definitionsmenge kann ein kartesisches Produkt sein:

- Schreibweise: $f:(A \times B) \rightarrow C$
- Wenn die Definitionsmenge n-Tupel enthält, so heißt f eine n-stellige Funktion

Beachte auch: Wir können umgekehrt jede Funktion als 1-stellig betrachten

- Sie bildet dann ein einzelnes Tupel auf einen Wert ab

Eigenschaften von Funktionen

Definition: Sei $f:A \rightarrow B$ eine Funktion

- f heißt surjektiv ("auf B") falls $f(A)=B$
  - Jedes Element der Zielmenge ist im Bild der Funktion
- f heißt injektiv (eindeutig), falls $\forall a,b:f(a)=f(b) \Rightarrow a=b$
  - Jedes Element der Zielmenge hat höchstens ein Urbild
- f heißt bijektiv, falls f injektiv und surjektiv ist
  - Jedem Element der Definitionsmenge ist genau ein Element der Zielmenge zugeordnet

Wertetabellen

Endliche 1- und 2-stellige Funktionen können als Wertetabellen dargestellt werden:

n	inc(n)
0	1
1	2
2	3
3	4
4	0

$inc:\{0,1,2,3,4\}\rightarrow\{0,1,2,3,4\}$

$inc=\{(0,1),(1,2),(2,3),(3,4),(4,0)\}$

inc ist injectiv

inc ist surjectiv

inc ist bijektiv

Wertetabellen (Zweistellig)

$+_4$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

- Addition in $\mathbf{Z}_4$
- f: $\mathbf{Z}_4 \times \mathbf{Z}_4 \rightarrow \mathbf{Z}_4$
- $(x,y) \mapsto (x+y) \bmod 4$
- Frage: Surjektiv?
- Frage: Injektiv?
- Frage: Bijektiv?

DingDingDing!

Sowohl Definitionsmenge als auch Zielmenge einer Funktion können

- Mengen von Tupeln
- Mengen von Mengen
- Mengen von Funktionen

sein!

Beispiele

Funktionsgenerator für Increment-Funktionen:

- incer:$\mathbf{N} \rightarrow (\mathbf{N} \rightarrow \mathbf{N})$
- $a \mapsto \{(x, x+a) | x \in \mathbf{N}\}$
  - incer(4)(4)=8, incer(4)(5)=9, incer(0)=Identitätsfunktion

Potenzmengen-Bilder für Mengen über $\mathbf{N}$:

- $P: 2^{\mathbf{N}} \rightarrow 2^{2^{\mathbf{N}}}$
- $A \mapsto 2^A$

Uff! Genug Mathe! (Hah!)

Alphabet

Definition: Ein Alphabet Σ ist eine nichtleere, endliche Menge von Symbolen.

- Die Symbole heißen auch Buchstaben oder Zeichen des Alphabets.

Beispiele:

- $\Sigma=\{0,1\}$ (Binäralphabet)
- $\Sigma=\{a,...,z\}$ (Kleinbuchstaben)
- $\Sigma=\{x|x$ ist ein ASCII-Zeichen$\}$

Worte

Definition: Ein Wort über einem Alphabet Σ ist eine endliche Sequenz von Buchstaben aus Σ.

- Wir schreiben ϵ für das leere Wort.
- $|w|$ ist die Länge des Wortes w
- Σ^k ist die Menge aller Worte mit Länge k über Σ.
  - Also: $\Sigma^0 = \{\epsilon\}$ für jedes Σ.
- Σ^* ist die Menge aller Worte über Σ.
  - Also: $\Sigma^* = \bigcup_{i \in \mathbf{N}} \Sigma^i$

Beispiele

- *0001*, *101*, *10010* sind Worte über $\Sigma=\{0, 1\}$
- *hallo* ist ein Wort über $\Sigma=\{a, ..., z\}$
- $\epsilon \in \Sigma^*$ für jedes beliebige Alphabet Σ
- $|\epsilon|=0$
- Sei Σ_{ascii} die Menge aller ASCII-Zeichen
  - Ein C-Programm ist (in der Regel) ein Wort über Σ_{ascii}
  - $(\Sigma_{ascii} \setminus \{\text{‘\0’}, \text{‘/’}\})^* \setminus \{\epsilon\}$ ist die Menge der gültigen Filenamen in UNIX (und LINUX)

Worte (2)

Definition: Sei $w \in \Sigma^k$.

Wir schreiben $w = w_1 w_2 ... w_k$, w_i.

- Seien $u = u_1 u_2 ... u_n$ und $v = v_1 v_2 ... v_m \in \Sigma^*$
  - $uv = u_1 u_2 ... u_n v_1 v_2 ... v_m$ ist die Konkatenation von u und v
- Falls $w = uvt$ $(u,v,t \in \Sigma^*)$, so heißt v ein Teilwort von w
  - Falls $u \neq \epsilon$ oder $v \neq \epsilon$, so heißt v echtes Teilwort von w
  - Falls $u = \epsilon$, so heißt v Anfangswort von w
  - Falls $t = \epsilon$, so heißt v Endwort von w

Formale Sprachen

Definition: Sei Σ ein Alphabet.

- Eine formale Sprache L über Σ ist eine Teilmenge von Σ^*

Beispiele: Sei Σ={a, ..., z}

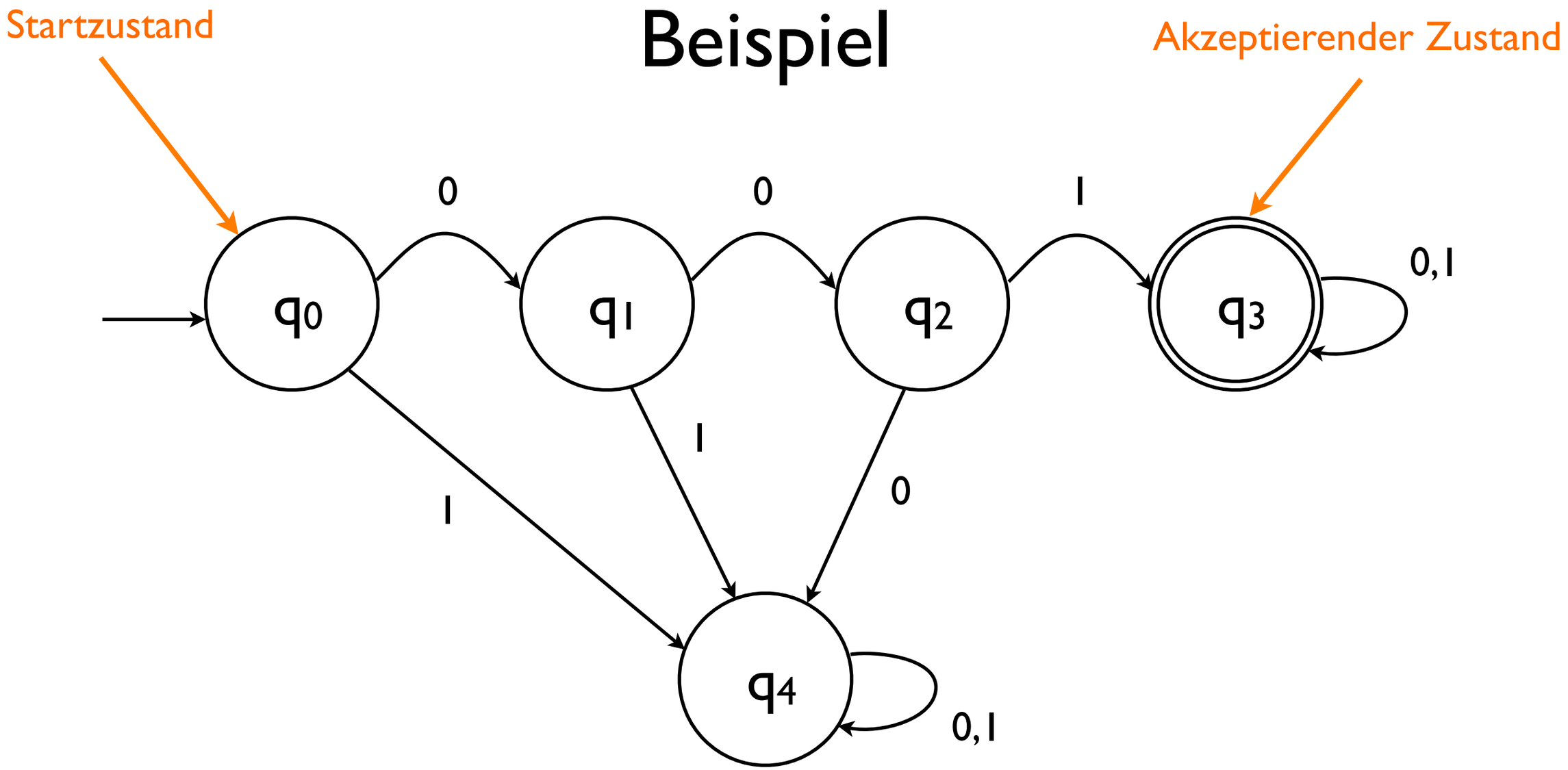
- {} ist eine formale Sprache über Σ
- {ϵ} ist eine formale Sprache über Σ
- {av|v $\in \Sigma^*$} (die Menge der Worte, die mit a beginnen) ist eine formale Sprache über Σ^*

Endliche Automaten

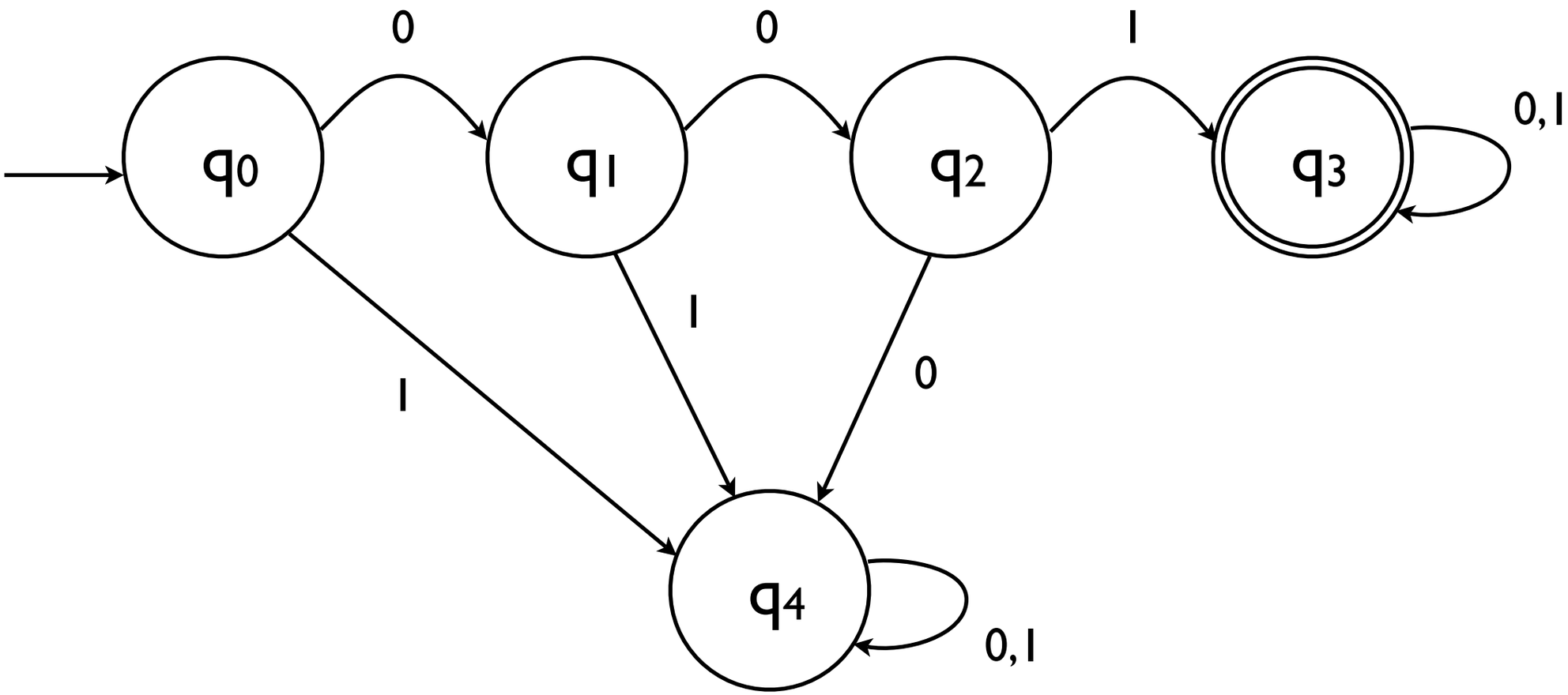
Endliche Automaten (EAs) verarbeiten Worte

- Ein EA beginnt in einem definierten Anfangszustand
- Er liest das Wort Zeichen für Zeichen
- Je nach aktuellem Zustand und aktuellem Zeichen geht er in einen neuen Zustand über
- Ist das Wort zu Ende, befinden er sich im Endzustand
- Er akzeptiert ein Wort, wenn der Endzustand ein akzeptierender Zustand ist
- Bequeme Darstellung im Diagramm

Beispiel


Startzustand
Akzeptierender Zustand
q0
q1
q2
q3
q4
0
0
1
0,1
1
1
0
0,1

Beispiel



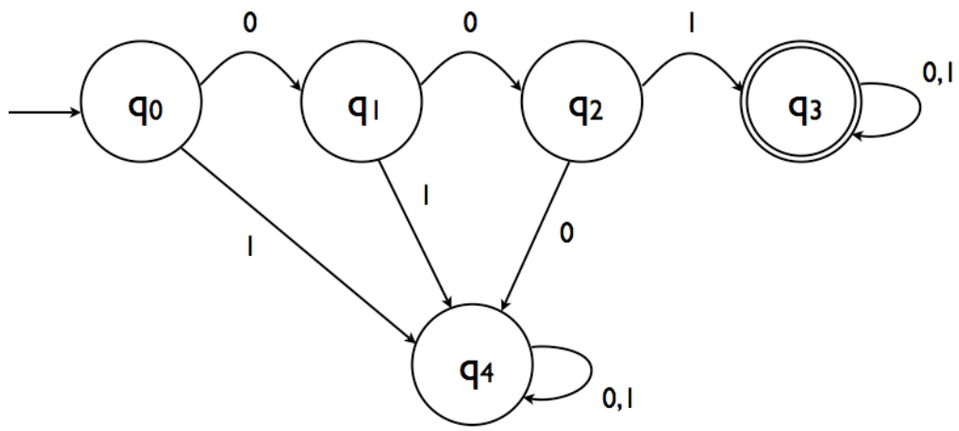
Was passiert mit Worten 01111, 0010101, 0100111?

Endliche Automaten: Formale Definition

Definition: Ein (deterministischer) endlicher Automat ist ein 5-Tupel $A=(Q, \Sigma, \delta, q_0, F)$, wobei

- Q ist eine endliche Menge (die Zustände)
- Σ ist ein Alphabet
- $\delta: Q \times \Sigma \rightarrow Q$ ist die Übergangsfunktion
- $q_0 \in Q$ ist der Startzustand
- $F \subseteq Q$ ist die Menge der akzeptierenden Zustände

Beispiel formal



$A = (Q, \Sigma, \delta, q_0, F)$

- $Q=\{q_0, q_1, q_2, q_3, q_4\}$
- Der Startzustand ist q_0
- $\Sigma=\{0,1\}$
- $F=\{q_3\}$

δ	0	1
q_0	q_1	q_4
q_1	q_2	q_4
q_2	q_4	q_3
q_3	q_3	q_3
q_4	q_4	q_4

L(A)

Definition: Sei $A= (Q, \Sigma, \delta, q_0, F)$ ein endlicher Automat

- A akzeptiert ein Wort $w \in \Sigma^*$, wenn er beim Start in Zustand q_0 nach Verarbeitung des Wortes in einem Zustand $q \in F$ ist
- $L(A)=\{w \in \Sigma^* | A \text{ akzeptiert } w\}$ ist die Sprache von A
  - Auch: "A erkennt L(A)"

Ende

Aufgaben

Zeigen oder widerlegen Sie:

- Jede Äquivalenzrelation ist linkstotal
- Jede Äquivalenzrelation ist rechtseindeutig
- Für jede endliche Menge A gilt: $|2^A|=2^{|A|}$

Geben Sie eine Äquivalenzrelation (außer =) über $\mathbf{N}$ an

Konstruieren sie Automaten, die folgende Sprachen erkennen (jeweils Übergangsgraph und formale Darstellung):

- $\{01v|v\in\Sigma^*\}$, $\{u01|u\in\Sigma^*\}$, $\{u01v|u,v\in\Sigma^*\}$