Kapitel 1

Deterministische Endliche Automaten

Abschnitt 1.1

Motivation und Informelle Einführung

Endliche Automaten

Endliche Automaten sind ein einfaches Berechnungsmodell, mit dem sich sequentielle Prozesse beschreiben lassen.

Endliche Automaten

Endliche Automaten sind ein einfaches Berechnungsmodell, mit dem sich sequentielle Prozesse beschreiben lassen.

Ein endlicher Automat hat eine endliche Menge von Zuständen, Übergänge zwischen diesen Zuständen werden durch Aktionen ausgelöst.

Endliche Automaten

Endliche Automaten sind ein einfaches Berechnungsmodell, mit dem sich sequentielle Prozesse beschreiben lassen.

Ein endlicher Automat hat eine endliche Menge von Zuständen, Übergänge zwischen diesen Zuständen werden durch Aktionen ausgelöst.

Außerdem hat ein endlicher Automat einen ausgezeichneten Startzustand, in dem die Berechnung beginnen soll, und möglicherweise ein oder mehrere Endzustände, die anzeigen, dass eine Berechnung (erfolgreich) abgeschlossen ist.

Endliche Automaten

Endliche Automaten sind ein einfaches Berechnungsmodell, mit dem sich sequentielle Prozesse beschreiben lassen.

Ein endlicher Automat hat eine endliche Menge von Zuständen, Übergänge zwischen diesen Zuständen werden durch Aktionen ausgelöst.

Außerdem hat ein endlicher Automat einen ausgezeichneten Startzustand, in dem die Berechnung beginnen soll, und möglicherweise ein oder mehrere Endzustände, die anzeigen, dass eine Berechnung (erfolgreich) abgeschlossen ist.

Eine Berechnung eines endlichen Automaten bezeichnet man üblicherweise als Lauf des Automaten.

Darstellung

Endliche Automaten lassen sich als gerichtete Graphen darstellen, deren Knoten den Zuständen entsprechen und deren Kanten mit Aktionen beschriftet sind.

Darstellung

Endliche Automaten lassen sich als gerichtete Graphen darstellen, deren Knoten den Zuständen entsprechen und deren Kanten mit Aktionen beschriftet sind.

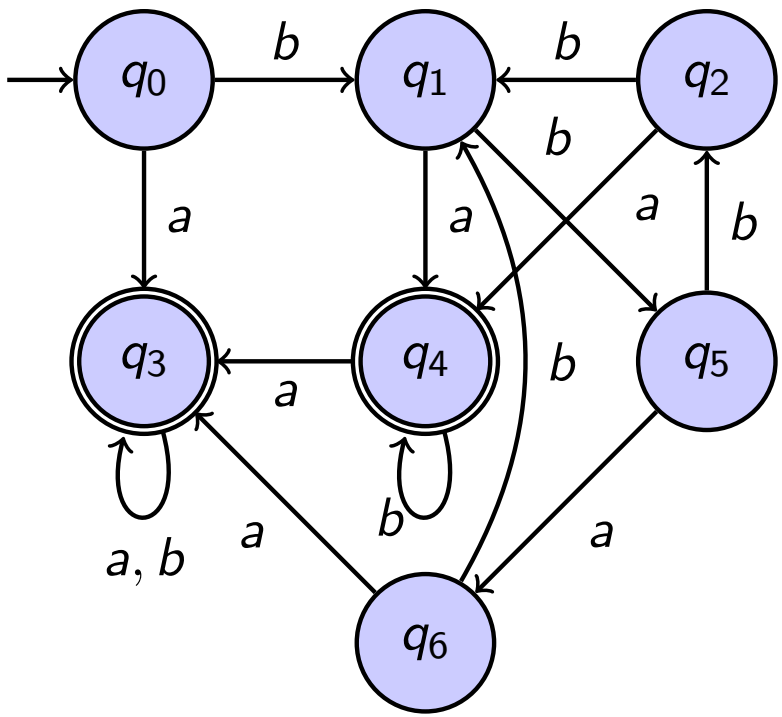
In der graphischen Darstellung wird der Anfangszustand durch einen Pfeil gekennzeichnet, die Endzustände werden doppelt umrandet.

Darstellung

Endliche Automaten lassen sich als gerichtete Graphen darstellen, deren Knoten den Zuständen entsprechen und deren Kanten mit Aktionen beschriftet sind.

In der graphischen Darstellung wird der Anfangszustand durch einen Pfeil gekennzeichnet, die Endzustände werden doppelt umrandet.

Beispiel 1.1



Zustände:

$q_0, \ldots, q_6$

Aktionen:

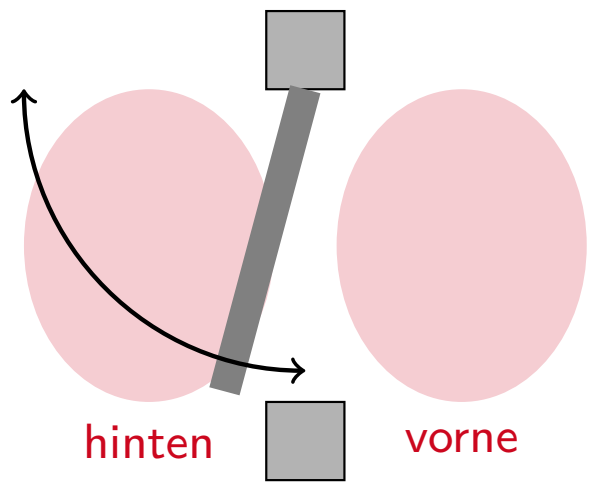
a, b

Beispiel 1.2: Controller für eine automatische Tür

Informelle Beschreibung

Wir modellieren einen Controller für eine automatische Tür. Die Tür ist mit je einem Sensor vor und hinter der Tür ausgestattet. Sie soll sich automatisch öffnen, wenn jemand von vorne kommt, darf sich aber weder öffnen noch schließen, wenn jemand hinter der Tür steht.

Schematische Darstellung

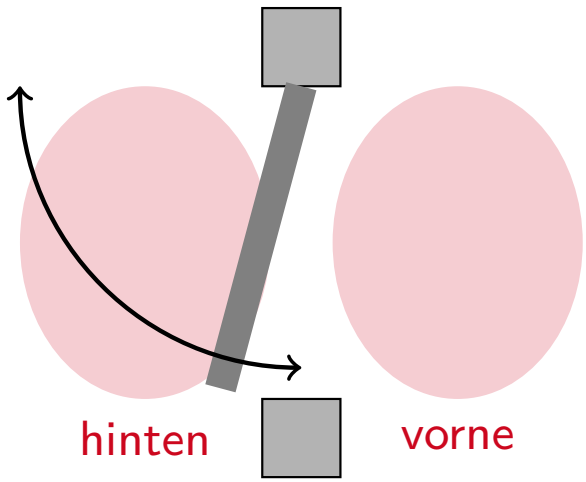


Beispiel 1.2: Controller für eine automatische Tür

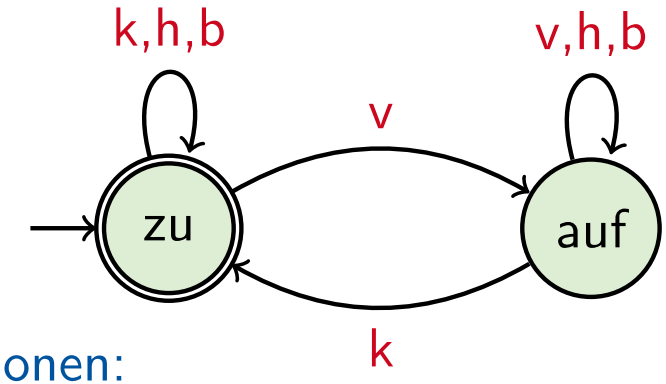
Informelle Beschreibung

Wir modellieren einen Controller für eine automatische Tür. Die Tür ist mit je einem Sensor vor und hinter der Tür ausgestattet. Sie soll sich automatisch öffnen, wenn jemand von vorne kommt, darf sich aber weder öffnen noch schließen, wenn jemand hinter der Tür steht.

Schematische Darstellung



Endlicher Automat



Aktionen:

kein Sensor gibt ein Signal
nur der vordere Sensor gibt ein Signal
nur der hintere Sensor gibt ein Signal
beide Sensoren geben ein Signal.

Beispiel 1.3: Getränkeautomat

Informelle Beschreibung

Getränkeautomat, der Tee, Espresso, und Cappuccino verkauft.

Beispiel 1.3: Getränkeautomat

Informelle Beschreibung

Getränkeautomat, der Tee, Espresso, und Cappuccino verkauft.

- Tee und Espresso kosten je 1€, Cappuccino kostet 2€.

Beispiel 1.3: Getränkeautomat

Informelle Beschreibung

Getränkeautomat, der Tee, Espresso, und Cappuccino verkauft.

- Tee und Espresso kosten je 1€, Cappuccino kostet 2€.
- Die Nutzerin kann entweder 1€ oder 2€ einwerfen.

Beispiel 1.3: Getränkeautomat

Informelle Beschreibung

Getränkeautomat, der Tee, Espresso, und Cappuccino verkauft.

- Tee und Espresso kosten je 1€, Cappuccino kostet 2€.
- Die Nutzerin kann entweder 1€ oder 2€ einwerfen.
- Wirft sie 2€ ein, so kann sie die Taste [C] drücken, und erhält einen Cappuccino.

Beispiel 1.3: Getränkeautomat

Informelle Beschreibung

Getränkeautomat, der Tee, Espresso, und Cappuccino verkauft.

- Tee und Espresso kosten je 1€, Cappuccino kostet 2€.
- Die Nutzerin kann entweder 1€ oder 2€ einwerfen.
- Wirft sie 2€ ein, so kann sie die Taste [C] drücken, und erhält einen Cappuccino.
- Wirft sie 1€ ein, so kann sie entweder die Taste [T] drücken und erhält einen Tee, oder sie kann die Taste [E] drücken und erhält einen Espresso, oder sie kann einen weiteren Euro einwerfen und dann die Taste [C] drücken, dann erhält sie einen Cappuccino.

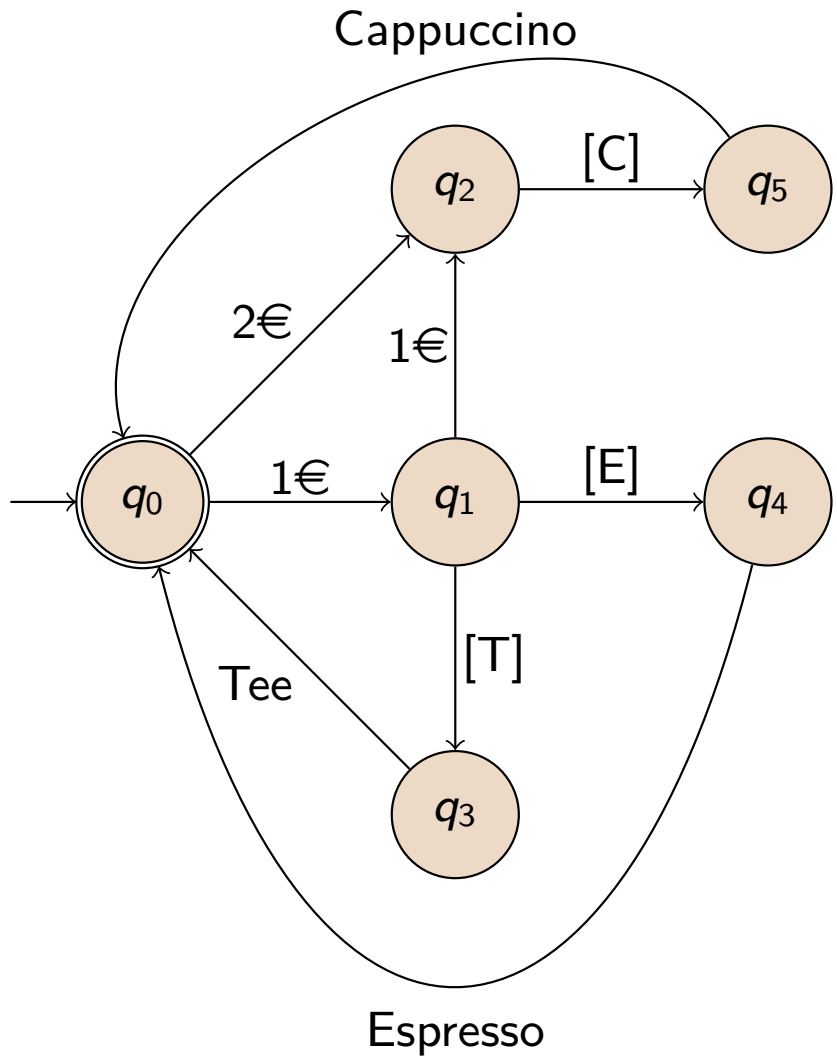
Beispiel 1.3: Getränkeautomat

Informelle Beschreibung

Getränkeautomat, der Tee, Espresso, und Cappuccino verkauft.

- Tee und Espresso kosten je 1€, Cappuccino kostet 2€.
- Die Nutzerin kann entweder 1€ oder 2€ einwerfen.
- Wirft sie 2€ ein, so kann sie die Taste [C] drücken, und erhält einen Cappuccino.
- Wirft sie 1€ ein, so kann sie entweder die Taste [T] drücken und erhält einen Tee, oder sie kann die Taste [E] drücken und erhält einen Espresso, oder sie kann einen weiteren Euro einwerfen und dann die Taste [C] drücken, dann erhält sie einen Cappuccino.

Endlicher Automat



Beispiel 1.4: Mustererkennung

Problemstellung

In einer Bitsequenz soll das Muster 1101 gefunden werden.

Beispiel 1.4: Mustererkennung

Problemstellung

In einer Bitsequenz soll das Muster 1101 gefunden werden.

Lösung mit endlichem Automaten

Der Automat liest die einzelnen Bits der Reihe nach.

Aktionen: 0, 1 (das gerade gelesene Bit)

Beispiel 1.4: Mustererkennung

Problemstellung

In einer Bitsequenz soll das Muster 1101 gefunden werden.

Lösung mit endlichem Automaten

Der Automat liest die einzelnen Bits der Reihe nach.

Aktionen: 0, 1 (das gerade gelesene Bit)

Zustände: 5 Zustände nichts, 1, 11, 110, 1101, in denen sich der Automat merkt, wieviel vom Muster 1101 schon gelesen wurde.

Beispiel 1.4: Mustererkennung

Problemstellung

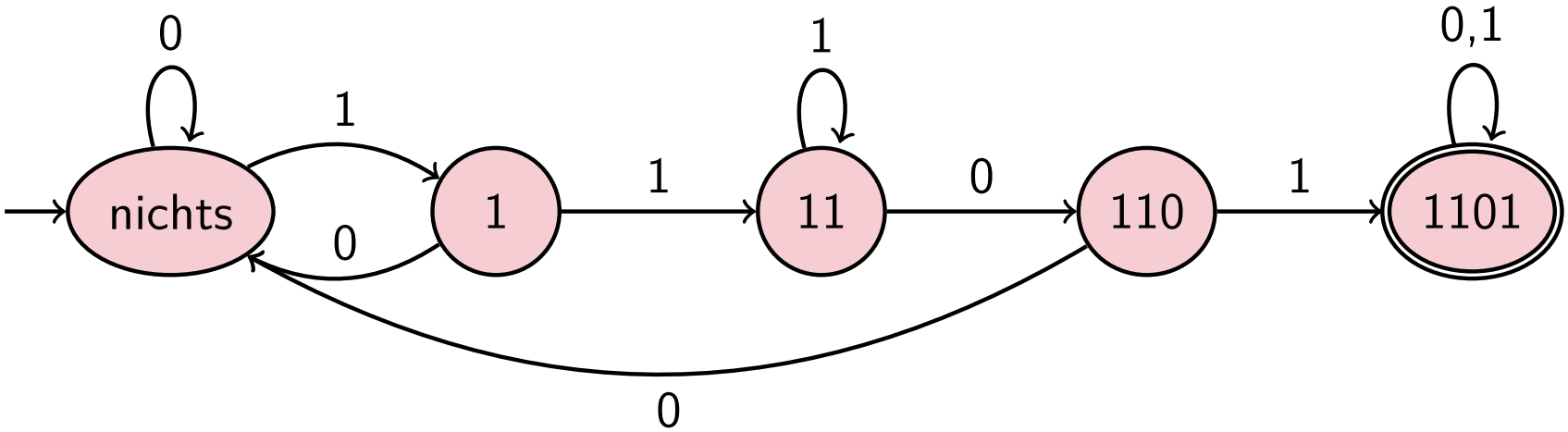
In einer Bitsequenz soll das Muster 1101 gefunden werden.

Lösung mit endlichem Automaten

Der Automat liest die einzelnen Bits der Reihe nach.

Aktionen: 0, 1 (das gerade gelesene Bit)

Zustände: 5 Zustände nichts, 1, 11, 110, 1101, in denen sich der Automat merkt, wieviel vom Muster 1101 schon gelesen wurde.



Beispiel 1.5: Teilbarkeit durch 3

Problemstellung

Es soll geprüft werden, ob eine Dezimalzahl durch drei teilbar ist.

Beispiel 1.5: Teilbarkeit durch 3

Problemstellung

Es soll geprüft werden, ob eine Dezimalzahl durch drei teilbar ist.

Idee

Prüfe, ob die Quersumme durch drei teilbar ist.

Beispiel 1.5: Teilbarkeit durch 3

Problemstellung

Es soll geprüft werden, ob eine Dezimalzahl durch drei teilbar ist.

Idee

Prüfe, ob die Quersumme durch drei teilbar ist.

Lösung mit endlichem Automaten

Der Automat liest die einzelnen Ziffern der Reihe nach.

Aktionen: 0, . . . , 9 (Lesen einer Ziffer)

Beispiel 1.5: Teilbarkeit durch 3

Problemstellung

Es soll geprüft werden, ob eine Dezimalzahl durch drei teilbar ist.

Idee

Prüfe, ob die Quersumme durch drei teilbar ist.

Lösung mit endlichem Automaten

Der Automat liest die einzelnen Ziffern der Reihe nach.

Aktionen: 0, . . . , 9 (Lesen einer Ziffer)

Zustände: 3 Zustände 0, 1, 2, in denen sich der Automat den Rest der bisher gebildeten Quersumme bei Division durch 3 merkt.

Beispiel 1.5: Teilbarkeit durch 3

Problemstellung

Es soll geprüft werden, ob eine Dezimalzahl durch drei teilbar ist.

Idee

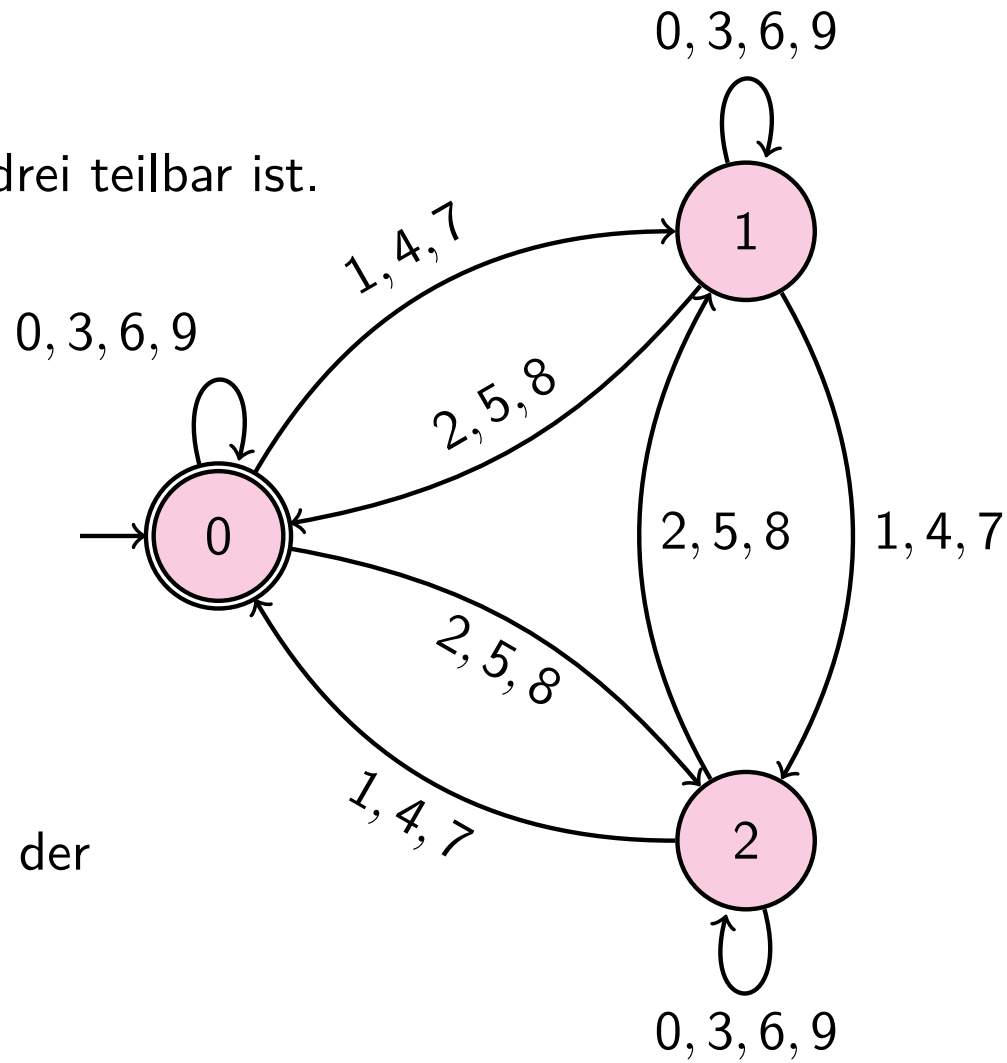
Prüfe, ob die Quersumme durch drei teilbar ist.

Lösung mit endlichem Automaten

Der Automat liest die einzelnen Ziffern der Reihe nach.

Aktionen: 0, . . . , 9 (Lesen einer Ziffer)

Zustände: 3 Zustände 0, 1, 2, in denen sich der Automat den Rest der bisher gebildeten Quersumme bei Division durch 3 merkt.



Bemerkungen

- Endliche Automaten haben keinen Speicher, sie können lediglich beschränkt viel Information in ihren Zuständen speichern. Das ist die entscheidende Einschränkung dieses Berechnungsmodells.

Bemerkungen

- Endliche Automaten haben keinen Speicher, sie können lediglich beschränkt viel Information in ihren Zuständen speichern. Das ist die entscheidende Einschränkung dieses Berechnungsmodells.
- Man könnte auf die Idee kommen, dass ja alle Computer prinzipiell nur endlichen Speicher haben und dass deswegen endliche Automaten ein universelles Berechnungsmodell sind.

Bemerkungen

- Endliche Automaten haben keinen Speicher, sie können lediglich beschränkt viel Information in ihren Zuständen speichern. Das ist die entscheidende Einschränkung dieses Berechnungsmodells.
- Man könnte auf die Idee kommen, dass ja alle Computer prinzipiell nur endlichen Speicher haben und dass deswegen endliche Automaten ein universelles Berechnungsmodell sind.
  Das ist aber keine geeignete Modellierung von Computern. Es ist vernünftig, anzunehmen, dass diese (potentiell) einen unbeschränkten Speicher haben.

Bemerkungen

- Endliche Automaten haben keinen Speicher, sie können lediglich beschränkt viel Information in ihren Zuständen speichern. Das ist die entscheidende Einschränkung dieses Berechnungsmodells.
- Man könnte auf die Idee kommen, dass ja alle Computer prinzipiell nur endlichen Speicher haben und dass deswegen endliche Automaten ein universelles Berechnungsmodell sind.
  Das ist aber keine geeignete Modellierung von Computern. Es ist vernünftig, anzunehmen, dass diese (potentiell) einen unbeschränkten Speicher haben.
- Die Rolle der Endzustände ist unterschiedlich:

Bemerkungen

- Endliche Automaten haben keinen Speicher, sie können lediglich beschränkt viel Information in ihren Zuständen speichern. Das ist die entscheidende Einschränkung dieses Berechnungsmodells.
- Man könnte auf die Idee kommen, dass ja alle Computer prinzipiell nur endlichen Speicher haben und dass deswegen endliche Automaten ein universelles Berechnungsmodell sind.
  Das ist aber keine geeignete Modellierung von Computern. Es ist vernünftig, anzunehmen, dass diese (potentiell) einen unbeschränkten Speicher haben.
- Die Rolle der Endzustände ist unterschiedlich:
  - Manchmal (Türcontroller, Kaffemaschine) signalisieren sie lediglich, dass sich das System in einem Default- oder Ruhezustand befindet.

Bemerkungen

- Endliche Automaten haben keinen Speicher, sie können lediglich beschränkt viel Information in ihren Zuständen speichern. Das ist die entscheidende Einschränkung dieses Berechnungsmodells.
- Man könnte auf die Idee kommen, dass ja alle Computer prinzipiell nur endlichen Speicher haben und dass deswegen endliche Automaten ein universelles Berechnungsmodell sind.
  Das ist aber keine geeignete Modellierung von Computern. Es ist vernünftig, anzunehmen, dass diese (potentiell) einen unbeschränkten Speicher haben.
- Die Rolle der Endzustände ist unterschiedlich:
  - Manchmal (Türcontroller, Kaffemaschine) signalisieren sie lediglich, dass sich das System in einem Default- oder Ruhezustand befindet.
  - Manchmal (Mustererkennung, Teilbarkeit) zeigen sie das Resultat der Berechnung an und sind damit von entscheidender Bedeutung.

Von Aktionenfolgen zu formalen Sprachen

In der abstrakten Automatentheorie ist es sinnvoll, die Aktionen eines Automaten einfach mit Buchstaben zu bezeichnen.

Von Aktionenfolgen zu formalen Sprachen

In der abstrakten Automatentheorie ist es sinnvoll, die Aktionen eines Automaten einfach mit Buchstaben zu bezeichnen.

Jeder Lauf eines endlichen Automaten entspricht einer Folge von Aktionen, also einem Wort.

Von Aktionenfolgen zu formalen Sprachen

In der abstrakten Automatentheorie ist es sinnvoll, die Aktionen eines Automaten einfach mit Buchstaben zu bezeichnen.

Jeder Lauf eines endlichen Automaten entspricht einer Folge von Aktionen, also einem Wort.

Die Menge aller erfolgreichen Läufe eines Automaten entspricht dann einer Menge von Wörtern, die wir als die Sprache des Automaten bezeichnen.

Abschnitt 1.2

Wörter und Sprachen

Definition 1.6

1. Ein Alphabet ist eine nichtleere endliche Menge, deren Elemente wir als Symbole, Buchstaben, oder Zeichen bezeichnen.

Definition 1.6

1. Ein Alphabet ist eine nichtleere endliche Menge, deren Elemente wir als Symbole, Buchstaben, oder Zeichen bezeichnen.
2. Ein Wort (oder String) über einem Alphabet ist eine endliche Folge (ein Tupel) von Zeichen aus dem Alphabet.

Definition 1.6

1. Ein Alphabet ist eine nichtleere endliche Menge, deren Elemente wir als Symbole, Buchstaben, oder Zeichen bezeichnen.
2. Ein Wort (oder String) über einem Alphabet ist eine endliche Folge (ein Tupel) von Zeichen aus dem Alphabet.
3. Eine formale Sprache (kurz: Sprache) über einem Alphabet ist eine Menge von Wörtern über diesem Alphabet.

Definition 1.6

1. Ein Alphabet ist eine nichtleere endliche Menge, deren Elemente wir als Symbole, Buchstaben, oder Zeichen bezeichnen.
2. Ein Wort (oder String) über einem Alphabet ist eine endliche Folge (ein Tupel) von Zeichen aus dem Alphabet.
3. Eine formale Sprache (kurz: Sprache) über einem Alphabet ist eine Menge von Wörtern über diesem Alphabet.

Motivation

Digitale Systeme verarbeiten Daten, die als Bitwörter kodiert sind. Deswegen sind Wörter und Sprachen grundlegende Objekte der Informatik.

Alphabete

Alphabete sind nichtleere endliche Mengen von Zeichen.

Alphabete

Alphabete sind nichtleere endliche Mengen von Zeichen.

Notation

- Σ, Γ und Varianten wie Σ_1, Γ' stehen für Alphabete.
- $a, b, c, \ldots$ und Varianten stehen für Zeichen.

Alphabete

Alphabete sind nichtleere endliche Mengen von Zeichen.

Notation

- Σ, Γ und Varianten wie Σ_1, Γ' stehen für Alphabete.
- $a, b, c, \ldots$ und Varianten stehen für Zeichen.

Beispiele 1.7

1. Das boolsche Alphabet $\{0, 1\}$.

Alphabete

Alphabete sind nichtleere endliche Mengen von Zeichen.

Notation

- Σ, Γ und Varianten wie Σ_1, Γ' stehen für Alphabete.
- $a, b, c, \ldots$ und Varianten stehen für Zeichen.

Beispiele 1.7

1. Das boolsche Alphabet $\{0, 1\}$.
2. Das Morsealphabet $\{\cdot, -, \quad\}$.

Alphabete

Alphabete sind nichtleere endliche Mengen von Zeichen.

Notation

- Σ, Γ und Varianten wie Σ_1, Γ' stehen für Alphabete.
- $a, b, c, \ldots$ und Varianten stehen für Zeichen.

Beispiele 1.7

1. Das boolsche Alphabet $\{0, 1\}$.
2. Das Morsealphabet $\{\cdot, -, \quad\}$.
3. Das Standardalphabet für deutsche Texte:

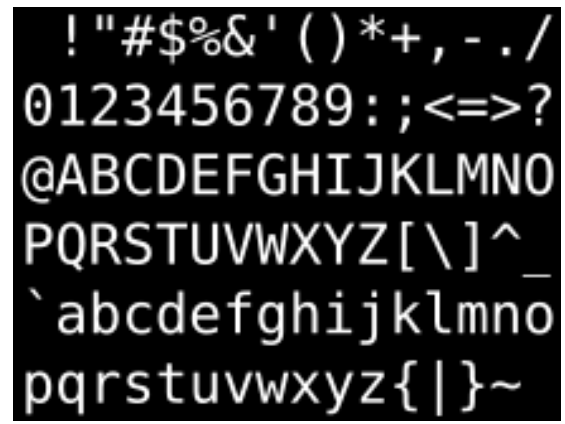
$$\Sigma_{\mathrm{DE}} := \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}, \ddot{\mathrm{A}}, \ddot{\mathrm{O}}, \ddot{\mathrm{U}}, \mathrm{a}, \ldots, \mathrm{z}, \ddot{\mathrm{a}}, \ddot{\mathrm{o}}, \ddot{\mathrm{u}}, \ss, ., ,, :, ;, !, ?, „, “, \quad\}.$$

Alphabete

Alphabete sind nichtleere endliche Mengen von Zeichen.

Notation

- Σ, Γ und Varianten wie Σ_1, Γ' stehen für Alphabete.
- $a, b, c, \ldots$ und Varianten stehen für Zeichen.



ASCII Symbole

Beispiele 1.7

1. Das boolsche Alphabet $\{0, 1\}$.
2. Das Morsealphabet $\{\cdot, -, \quad\}$.
3. Das Standardalphabet für deutsche Texte:

$$\Sigma_{DE} := \{A, B, \ldots, Z, Ä, Ö, Ü, a, \ldots, z, ä, ö, ü, ß, ., ,, :, ;, !, ?, „, “, \quad\}.$$

4. Das ASCII-Alphabet Σ_{ASCII} und das UTF-8-Alphabet Σ_{UTF8}, in denen typischerweise Textfiles kodiert sind.

   ASCII steht für “American Standard Code for Information Interchange”, UTF für “UCS Transformation Format”, UCS für “Universal Character Set”.

Wörter

Wörter sind endliche Zeichenfolgen.

Wörter

Wörter sind endliche Zeichenfolgen.

Notation

- Wir schreiben Wörter ohne Klammern und Kommata, also $aaba$ und nicht (a, a, b, a).

Wörter

Wörter sind endliche Zeichenfolgen.

Notation

- Wir schreiben Wörter ohne Klammern und Kommata, also $aaba$ und nicht (a, a, b, a).
- $u, v, w, \ldots$ und Varianten stehen für Wörter.

Wörter

Wörter sind endliche Zeichenfolgen.

Notation

- Wir schreiben Wörter ohne Klammern und Kommata, also $aaba$ und nicht (a, a, b, a).
- $u, v, w, \ldots$ und Varianten stehen für Wörter.
- Σ^* bezeichnet die Menge aller Wörter über dem Alphabet Σ.

Wörter

Wörter sind endliche Zeichenfolgen.

Notation

- Wir schreiben Wörter ohne Klammern und Kommata, also $aaba$ und nicht (a, a, b, a).
- $u, v, w, \ldots$ und Varianten stehen für Wörter.
- Σ^* bezeichnet die Menge aller Wörter über dem Alphabet Σ.
- ε bezeichnet das leere Wort.

Wörter sind endliche Zeichenfolgen.

Notation

- Wir schreiben Wörter ohne Klammern und Kommata, also $aaba$ und nicht (a, a, b, a).
- $u, v, w, \ldots$ und Varianten stehen für Wörter.
- Σ^* bezeichnet die Menge aller Wörter über dem Alphabet Σ.
- ε bezeichnet das leere Wort.
- $|w|$ bezeichnet die Länge des Wortes w.

  Beispiele: $|aaab| = 4, |\varepsilon| = 0$

Wörter sind endliche Zeichenfolgen.

Notation

- Wir schreiben Wörter ohne Klammern und Kommata, also $aaba$ und nicht (a, a, b, a).
- $u, v, w, \ldots$ und Varianten stehen für Wörter.
- Σ^* bezeichnet die Menge aller Wörter über dem Alphabet Σ.
- ε bezeichnet das leere Wort.
- $|w|$ bezeichnet die Länge des Wortes w.

  Beispiele: $|aaab| = 4, |\varepsilon| = 0$
- $|w|_a$ bezeichnet die Häufigkeit des Buchstaben a im Wort w.

  Beispiele: $|aaab|_a = 3, |aaab|_b = 1$

Wörter sind endliche Zeichenfolgen.

Notation

- Wir schreiben Wörter ohne Klammern und Kommata, also $aaba$ und nicht (a, a, b, a).
- $u, v, w, \ldots$ und Varianten stehen für Wörter.
- Σ^* bezeichnet die Menge aller Wörter über dem Alphabet Σ.
- ε bezeichnet das leere Wort.
- $|w|$ bezeichnet die Länge des Wortes w.

  Beispiele: $|aaab| = 4, |\varepsilon| = 0$
- $|w|_a$ bezeichnet die Häufigkeit des Buchstaben a im Wort w.

  Beispiele: $|aaab|_a = 3, |aaab|_b = 1$
- vw bezeichnet die Verkettung der Wörter v und w

  Beispiel: Für $v = aaba$ und $w = ba$ ist $vw = aababa$.

Definition 1.8

Seien u, v Wörter. Dann ist u

1. ein Präfix (Anfangsstück) von v, falls es ein Wort w gibt, so dass $v = uw$;
   Schreibweise: $u \sqsubseteq v$;

Definition 1.8

Seien u, v Wörter. Dann ist u

1. ein Präfix (Anfangsstück) von v, falls es ein Wort w gibt, so dass $v = uw$;
   Schreibweise: $u \sqsubseteq v$;
2. ein Infix von v, falls es Wörter w, w' gibt, so dass $v = wuw'$;

Definition 1.8

Seien u, v Wörter. Dann ist u

1. ein Präfix (Anfangsstück) von v, falls es ein Wort w gibt, so dass $v = uw$;
   Schreibweise: $u \sqsubseteq v$;
2. ein Infix von v, falls es Wörter w, w' gibt, so dass $v = wuw'$;
3. ein Suffix (Endstück) von v, falls es ein Wort w gibt, so dass $v = wu$.

Definition 1.8

Seien u, v Wörter. Dann ist u

1. ein Präfix (Anfangsstück) von v, falls es ein Wort w gibt, so dass $v = uw$;
   Schreibweise: $u \sqsubseteq v$;
2. ein Infix von v, falls es Wörter w, w' gibt, so dass $v = wuw'$;
3. ein Suffix (Endstück) von v, falls es ein Wort w gibt, so dass $v = wu$.

Beispiel 1.9

Sei $v = aaba$.

- Die Präfixe von v sind $\varepsilon, a, aa, aab, aaba$.

Definition 1.8

Seien u, v Wörter. Dann ist u

1. ein Präfix (Anfangsstück) von v, falls es ein Wort w gibt, so dass $v = uw$;
   Schreibweise: $u \sqsubseteq v$;
2. ein Infix von v, falls es Wörter w, w' gibt, so dass $v = wuw'$;
3. ein Suffix (Endstück) von v, falls es ein Wort w gibt, so dass $v = wu$.

Beispiel 1.9

Sei $v = aaba$.

- Die Präfixe von v sind $\varepsilon, a, aa, aab, aaba$.
- Die Suffixe von v sind $\varepsilon, a, ba, aba, aaba$.

Definition 1.8

Seien u, v Wörter. Dann ist u

1. ein Präfix (Anfangsstück) von v, falls es ein Wort w gibt, so dass $v = uw$;
   Schreibweise: $u \sqsubseteq v$;
2. ein Infix von v, falls es Wörter w, w' gibt, so dass $v = wuw'$;
3. ein Suffix (Endstück) von v, falls es ein Wort w gibt, so dass $v = wu$.

Beispiel 1.9

Sei $v = aaba$.

- Die Präfixe von v sind $\varepsilon, a, aa, aab, aaba$.
- Die Suffixe von v sind $\varepsilon, a, ba, aba, aaba$.
- Die Infixe von v sind alle Präfixe und Suffixe sowie ab, b.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Notation

K, L und Varianten bezeichnen Sprachen.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Notation

K, L und Varianten bezeichnen Sprachen.

Beispiele 1.10

Über dem Alphabet $\{0, 1\}$:

- Menge aller Wörter, die 1101 als Infix enthalten.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Notation

K, L und Varianten bezeichnen Sprachen.

Beispiele 1.10

Über dem Alphabet $\{0, 1\}$:

- Menge aller Wörter, die 1101 als Infix enthalten.
- Menge aller Binärdarstellungen von Primzahlen.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Notation

K, L und Varianten bezeichnen Sprachen.

Beispiele 1.10

Über dem Alphabet $\{0, 1\}$:

- Menge aller Wörter, die 1101 als Infix enthalten.
- Menge aller Binärdarstellungen von Primzahlen.
- Menge aller Wörter, die eine gerade Anzahl von 1en enthalten.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Notation

K, L und Varianten bezeichnen Sprachen.

Beispiele 1.10

Über dem Alphabet $\{0, 1\}$:

- Menge aller Wörter, die 1101 als Infix enthalten.
- Menge aller Binärdarstellungen von Primzahlen.
- Menge aller Wörter, die eine gerade Anzahl von 1en enthalten.

Beispiele 1.11

Über einem beliebigen Alphabet Σ:

- Die leere Sprache $\emptyset$.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Notation

K, L und Varianten bezeichnen Sprachen.

Beispiele 1.10

Über dem Alphabet $\{0, 1\}$:

- Menge aller Wörter, die 1101 als Infix enthalten.
- Menge aller Binärdarstellungen von Primzahlen.
- Menge aller Wörter, die eine gerade Anzahl von 1en enthalten.

Beispiele 1.11

Über einem beliebigen Alphabet Σ:

- Die leere Sprache $\emptyset$.
- Die Sprache $\{\varepsilon\}$, die nur das leere Wort enthält.

Sprachen

Eine Sprache über einem Alphabet Σ ist eine Teilmenge von Σ^*.

Notation

K, L und Varianten bezeichnen Sprachen.

Beispiele 1.10

Über dem Alphabet $\{0, 1\}$:

- Menge aller Wörter, die 1101 als Infix enthalten.
- Menge aller Binärdarstellungen von Primzahlen.
- Menge aller Wörter, die eine gerade Anzahl von 1en enthalten.

Beispiele 1.11

Über einem beliebigen Alphabet Σ:

- Die leere Sprache $\emptyset$.
- Die Sprache $\{\varepsilon\}$, die nur das leere Wort enthält.
- Σ^*.

Beispiele 1.12

Über dem Standardalphabet der deutschen Sprache Σ_{DE}:

- Menge alle deutschen Wörter,

Beispiele 1.12

Über dem Standardalphabet der deutschen Sprache Σ_{DE}:

- Menge alle deutschen Wörter,
- Menge aller grammatikalisch korrekten deutschen Sätze,

Beispiele 1.12

Über dem Standardalphabet der deutschen Sprache Σ_{DE}:

- Menge alle deutschen Wörter,
- Menge aller grammatikalisch korrekten deutschen Sätze,
- Menge aller Palindrome.

  Beispiele: ANNA, REITTIER, RENTNER, O|GENIE|DER|HERR|EHRE|DEIN|EGO

Beispiele 1.12

Über dem Standardalphabet der deutschen Sprache Σ_{DE}:

- Menge alle deutschen Wörter,
- Menge aller grammatikalisch korrekten deutschen Sätze,
- Menge aller Palindrome.

  Beispiele: ANNA, REITTIER, RENTNER, O|GENIE|DER|HERR|EHRE|DEIN|EGO

Beispiele 1.13

Über dem UTF8-Alphabet Σ_{UTF8} (eigentlich Unicode):

- Menge aller JAVA-Schlüsselwörter,

Beispiele 1.12

Über dem Standardalphabet der deutschen Sprache Σ_{DE}:

- Menge alle deutschen Wörter,
- Menge aller grammatikalisch korrekten deutschen Sätze,
- Menge aller Palindrome.

  Beispiele: ANNA, REITTIER, RENTNER, O|GENIE|DER|HERR|EHRE|DEIN|EGO

Beispiele 1.13

Über dem UTF8-Alphabet Σ_{UTF8} (eigentlich Unicode):

- Menge aller JAVA-Schlüsselwörter,
- Menge aller in JAVA erlaubten Variablennamen,

Beispiele 1.12

Über dem Standardalphabet der deutschen Sprache Σ_{DE}:

- Menge alle deutschen Wörter,
- Menge aller grammatikalisch korrekten deutschen Sätze,
- Menge aller Palindrome.

  Beispiele: ANNA, REITTIER, RENTNER, O|GENIE|DER|HERR|EHRE|DEIN|EGO

Beispiele 1.13

Über dem UTF8-Alphabet Σ_{UTF8} (eigentlich Unicode):

- Menge aller JAVA-Schlüsselwörter,
- Menge aller in JAVA erlaubten Variablennamen,
- Menge aller syntaktisch korrekten JAVA-Programme.

Abschnitt 1.3

Deterministische Endliche Automaten (Formal)

Definition 1.14

Ein deterministischer endlicher Automat (kurz: DFA) ist ein 5-Tupel

$$(Q, \Sigma, \delta, q_0, F);$$

dabei ist

- Q eine endliche Menge, deren Elemente wir als die Zustände bezeichnen,
- Σ ein Alphabet, das Eingabealphabet,
- $\delta : Q \times \Sigma \to Q$ eine Abbildung, die Transitionsfunktion,
- $q_0 \in Q$ der Anfangszustand,
- $F \subseteq Q$ die Menge der Endzustände oder akzeptierenden Zustände.

Definition 1.14

Ein deterministischer endlicher Automat (kurz: DFA) ist ein 5-Tupel

$$(Q, \Sigma, \delta, q_0, F);$$

dabei ist

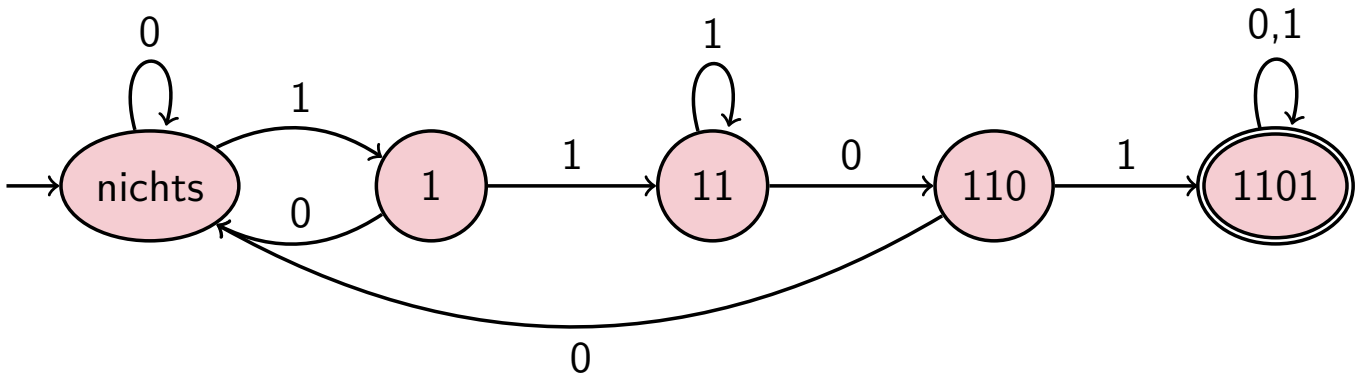
- Q eine endliche Menge, deren Elemente wir als die Zustände bezeichnen,
- Σ ein Alphabet, das Eingabealphabet,
- $\delta : Q \times \Sigma \to Q$ eine Abbildung, die Transitionsfunktion,
- $q_0 \in Q$ der Anfangszustand,
- $F \subseteq Q$ die Menge der Endzustände oder akzeptierenden Zustände.

Bemerkung 1.15

Das Kürzel DFA kommt vom englischen "Deterministic Finite Automaton."

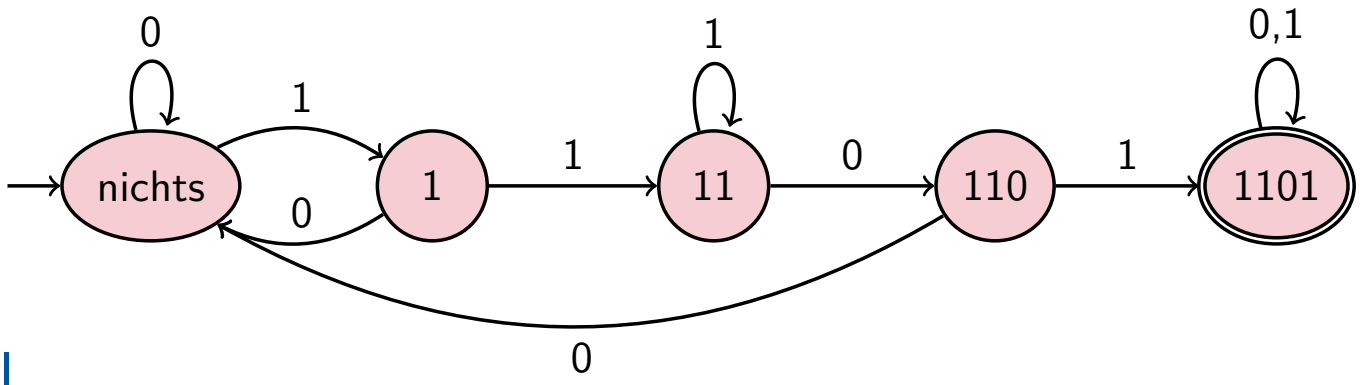
Beispiel 1.4 (Forts.)

Graphische Darstellung



Beispiel 1.4 (Forts.)

Graphische Darstellung

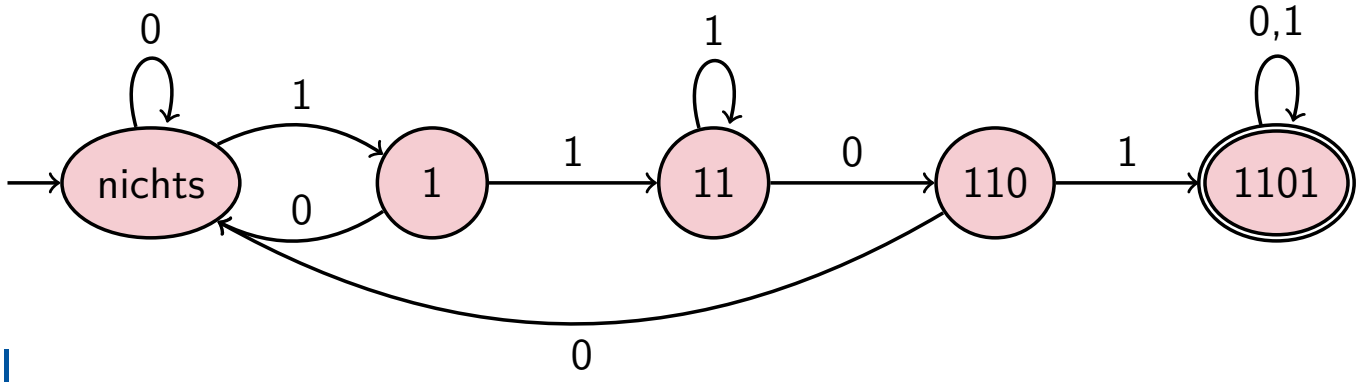


Formal

- Zustandsmenge $Q := \{\text{nichts}, 1, 11, 110, 1101\}$,

Beispiel 1.4 (Forts.)

Graphische Darstellung

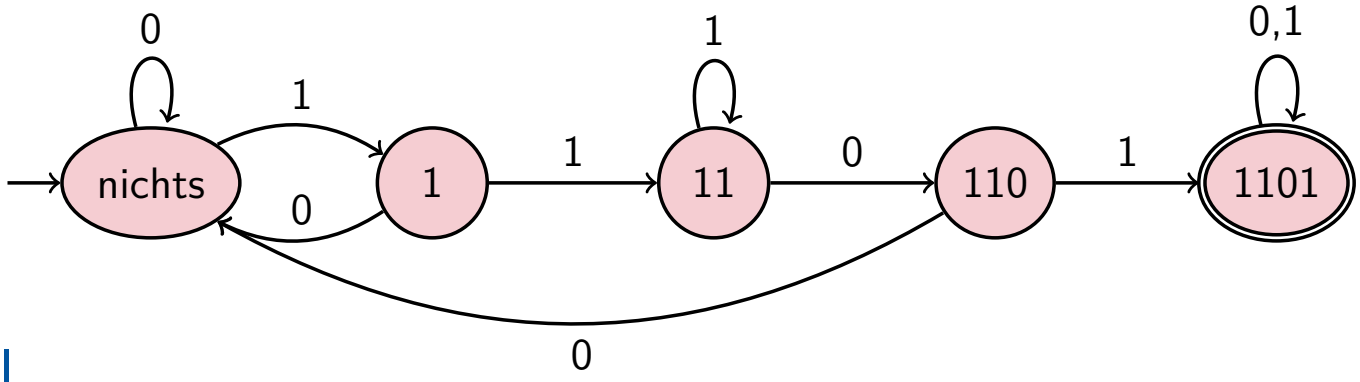


Formal

- Zustandsmenge $Q := \{\text{nichts}, 1, 11, 110, 1101\}$,
- Eingabealphabet $\Sigma := \{0, 1\}$,

Beispiel 1.4 (Forts.)

Graphische Darstellung



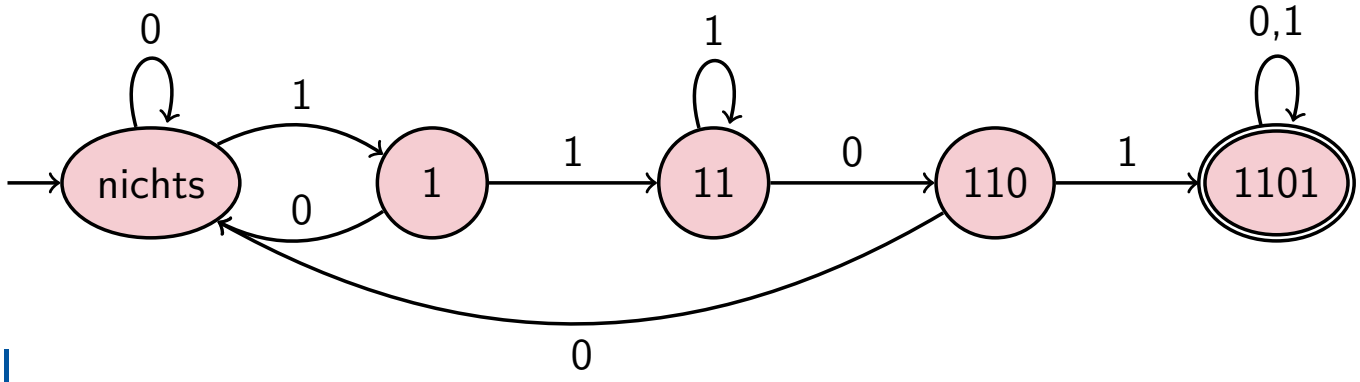
Formal

- Zustandsmenge $Q := \{\text{nichts}, 1, 11, 110, 1101\}$,
- Eingabealphabet $\Sigma := \{0, 1\}$,
- Transitionsfunktion δ gegeben durch folgende Tabelle:

q	nichts	1	11	110	1101
$\delta(q, 0)$	nichts	nichts	110	nichts	1101
$\delta(q, 1)$	1	11	11	1101	1101

Beispiel 1.4 (Forts.)

Graphische Darstellung



Formal

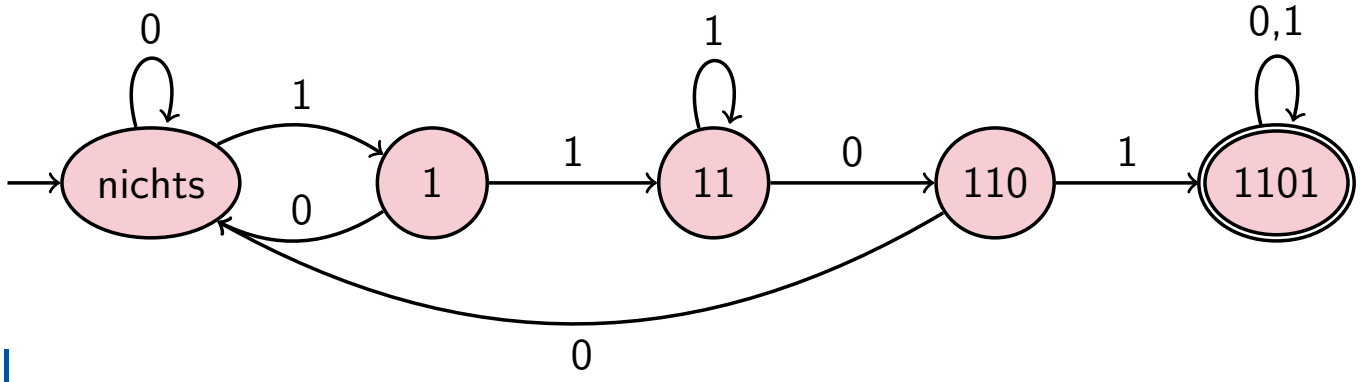
- Zustandsmenge $Q := \{\text{nichts}, 1, 11, 110, 1101\}$,
- Eingabealphabet $\Sigma := \{0, 1\}$,
- Transitionsfunktion δ gegeben durch folgende Tabelle:

q	nichts	1	11	110	1101
$\delta(q, 0)$	nichts	nichts	110	nichts	1101
$\delta(q, 1)$	1	11	11	1101	1101

- Anfangszustand $q_0 :=$ nichts,

Beispiel 1.4 (Forts.)

Graphische Darstellung



Formal

- Zustandsmenge $Q := \{\text{nichts}, 1, 11, 110, 1101\}$,
- Eingabealphabet $\Sigma := \{0, 1\}$,
- Transitionsfunktion δ gegeben durch folgende Tabelle:

q	nichts	1	11	110	1101
$\delta(q, 0)$	nichts	nichts	110	nichts	1101
$\delta(q, 1)$	1	11	11	1101	1101

- Anfangszustand $q_0 :=$ nichts,
- Menge der Endzustände $F := \{1101\}$.

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

- Knoten als Darstellungen von Zuständen,

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

- Knoten als Darstellungen von Zuständen,
- eine gerichtete mit a beschriftete Kante von Knoten q zu Knoten r im Falle $\delta(q, a) = r$,

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

- Knoten als Darstellungen von Zuständen,
- eine gerichtete mit a beschriftete Kante von Knoten q zu Knoten r im Falle $\delta(q, a) = r$,
- Kennzeichnung des Anfangszustandes durch einen Pfeil,

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

- Knoten als Darstellungen von Zuständen,
- eine gerichtete mit a beschriftete Kante von Knoten q zu Knoten r im Falle $\delta(q, a) = r$,
- Kennzeichnung des Anfangszustandes durch einen Pfeil,
- Kennzeichnung der Endzustände durch Umkreisung.

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

- Knoten als Darstellungen von Zuständen,
- eine gerichtete mit a beschriftete Kante von Knoten q zu Knoten r im Falle $\delta(q, a) = r$,
- Kennzeichnung des Anfangszustandes durch einen Pfeil,
- Kennzeichnung der Endzustände durch Umkreisung.

Wir bezeichnen den Graphen in dieser Darstellung auch als den Transitionsgraphen des Automaten.

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

- Knoten als Darstellungen von Zuständen,
- eine gerichtete mit a beschriftete Kante von Knoten q zu Knoten r im Falle $\delta(q, a) = r$,
- Kennzeichnung des Anfangszustandes durch einen Pfeil,
- Kennzeichnung der Endzustände durch Umkreisung.

Wir bezeichnen den Graphen in dieser Darstellung auch als den Transitionsgraphen des Automaten.

Notation

- DFAs (und später andere Automaten) bezeichnen wir mit $\mathcal{A}, \mathcal{B}$ und Varianten wie $\mathcal{A}', \mathcal{B}_1$.

Graphische Darstellung

In der graphischen Darstellung eines DFAs (und später anderer Automaten) verwenden wir

- Knoten als Darstellungen von Zuständen,
- eine gerichtete mit a beschriftete Kante von Knoten q zu Knoten r im Falle $\delta(q, a) = r$,
- Kennzeichnung des Anfangszustandes durch einen Pfeil,
- Kennzeichnung der Endzustände durch Umkreisung.

Wir bezeichnen den Graphen in dieser Darstellung auch als den Transitionsgraphen des Automaten.

Notation

- DFAs (und später andere Automaten) bezeichnen wir mit $\mathcal{A}, \mathcal{B}$ und Varianten wie $\mathcal{A}', \mathcal{B}_1$.
- Zustände bezeichnen wir mit p, q, r, s und Varianten.

Definition 1.16

1. Ein DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ akzeptiert ein Wort $w = a_1 \dots a_n \in \Sigma^*$, wenn es Zustände $r_0, \dots, r_n \in Q$ gibt, so dass:

   (i) $r_0 = q_0$,

   (ii) $\delta(r_{i-1}, a_i) = r_i$ für $1 \leq i \leq n$,

   (iii) $r_n \in F$.

Definition 1.16

1. Ein DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ akzeptiert ein Wort $w = a_1 \dots a_n \in \Sigma^*$, wenn es Zustände $r_0, \dots, r_n \in Q$ gibt, so dass:

   (i) $r_0 = q_0$,

   (ii) $\delta(r_{i-1}, a_i) = r_i$ für $1 \leq i \leq n$,

   (iii) $r_n \in F$.

   Wir bezeichnen $(r_0, a_1, r_1, a_2, \dots, r_{n-1}, a_n, r_n)$ als einen akzeptierenden Lauf von $\mathcal{A}$.

Definition 1.16

1. Ein DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ akzeptiert ein Wort $w = a_1 \dots a_n \in \Sigma^*$, wenn es Zustände $r_0, \dots, r_n \in Q$ gibt, so dass:

   (i) $r_0 = q_0$,

   (ii) $\delta(r_{i-1}, a_i) = r_i$ für $1 \leq i \leq n$,

   (iii) $r_n \in F$.

   Wir bezeichnen $(r_0, a_1, r_1, a_2, \dots, r_{n-1}, a_n, r_n)$ als einen akzeptierenden Lauf von $\mathcal{A}$.

2. Die von einem DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ erkannte Sprache ist

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \mathcal{A} \text{ akzeptiert } w\}.$$

Definition 1.16

1. Ein DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ akzeptiert ein Wort $w = a_1 \dots a_n \in \Sigma^*$, wenn es Zustände $r_0, \dots, r_n \in Q$ gibt, so dass:

   (i) $r_0 = q_0$,

   (ii) $\delta(r_{i-1}, a_i) = r_i$ für $1 \leq i \leq n$,

   (iii) $r_n \in F$.

   Wir bezeichnen $(r_0, a_1, r_1, a_2, \dots, r_{n-1}, a_n, r_n)$ als einen akzeptierenden Lauf von $\mathcal{A}$.

2. Die von einem DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ erkannte Sprache ist

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \mathcal{A} \text{ akzeptiert } w\}.$$

3. Eine Sprache L heißt DFA-erkennbar, wenn es einen DFA $\mathcal{A}$ gibt, so dass $L = L(\mathcal{A})$.

Definition 1.17

Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein DFA.
Ein Lauf von $\mathcal{A}$ ist eine Folge $(r_0, a_1, r_1, a_2, \ldots, r_{n-1}, a_n, r_n)$, für ein $n \geq 0$, wobei $r_0, \ldots, r_n \in Q$ und $a_1, \ldots, a_n \in \Sigma$, so dass

(i) $r_0 = q_0$,

(ii) $\delta(r_{i-1}, a_i) = r_i$ für $1 \leq i \leq n$,

Definition 1.17

Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein DFA.
Ein Lauf von $\mathcal{A}$ ist eine Folge $(r_0, a_1, r_1, a_2, \ldots, r_{n-1}, a_n, r_n)$, für ein $n \geq 0$, wobei $r_0, \ldots, r_n \in Q$ und $a_1, \ldots, a_n \in \Sigma$, so dass

(i) $r_0 = q_0$,

(ii) $\delta(r_{i-1}, a_i) = r_i$ für $1 \leq i \leq n$,

Wir bezeichnen $(r_0, a_1, r_1, a_2, \ldots, r_{n-1}, a_n, r_n)$ oder einfach die Zustandsfolge $(r_0, r_1, \ldots, r_n)$ auch als den Lauf von $\mathcal{A}$ auf dem Wort $w = a_1 \ldots a_n$.

Definition 1.17

Sei $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ ein DFA.
Ein Lauf von $\mathcal{A}$ ist eine Folge $(r_0, a_1, r_1, a_2, \ldots, r_{n-1}, a_n, r_n)$, für ein $n \geq 0$, wobei $r_0, \ldots, r_n \in Q$ und $a_1, \ldots, a_n \in \Sigma$, so dass

(i) $r_0 = q_0$,

(ii) $\delta(r_{i-1}, a_i) = r_i$ für $1 \leq i \leq n$,

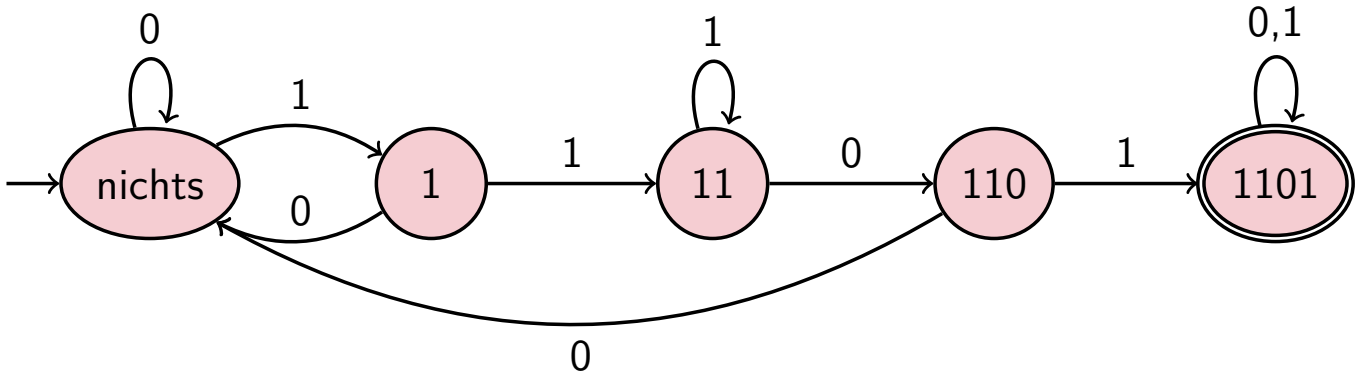
Wir bezeichnen $(r_0, a_1, r_1, a_2, \ldots, r_{n-1}, a_n, r_n)$ oder einfach die Zustandsfolge $(r_0, r_1, \ldots, r_n)$ auch als den Lauf von $\mathcal{A}$ auf dem Wort $w = a_1 \ldots a_n$.

Beobachtung 1.18

Zu jedem Wort $w \in \Sigma^*$ gibt es genau einen Lauf von $\mathcal{A}$ auf w.

Beispiel 1.4 (Forts.)

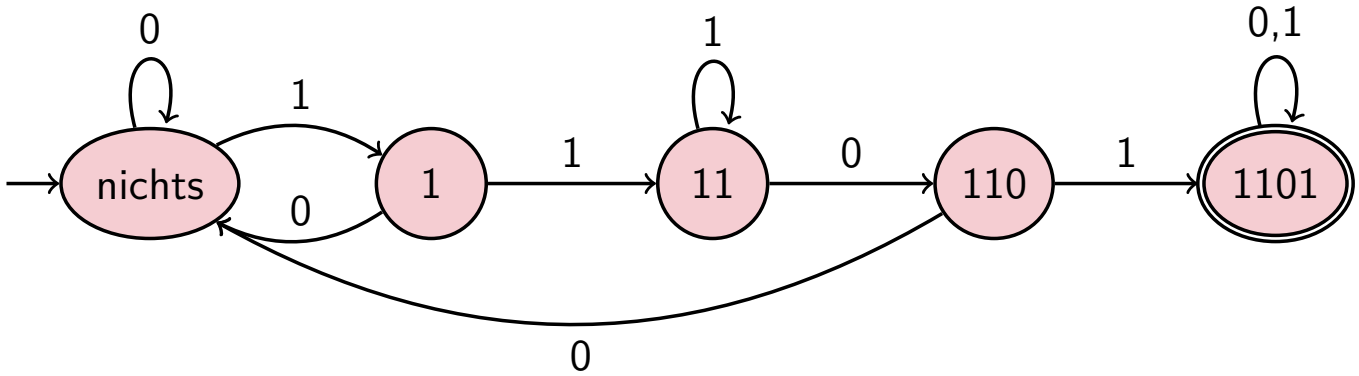
Sei $\mathcal{A}$ der Automat



Dann gilt $L(\mathcal{A}) = \{w \in \{0,1\}^* \mid 1101 \text{ ist ein Infix von } w\}$.

Beispiel 1.4 (Forts.)

Sei $\mathcal{A}$ der Automat



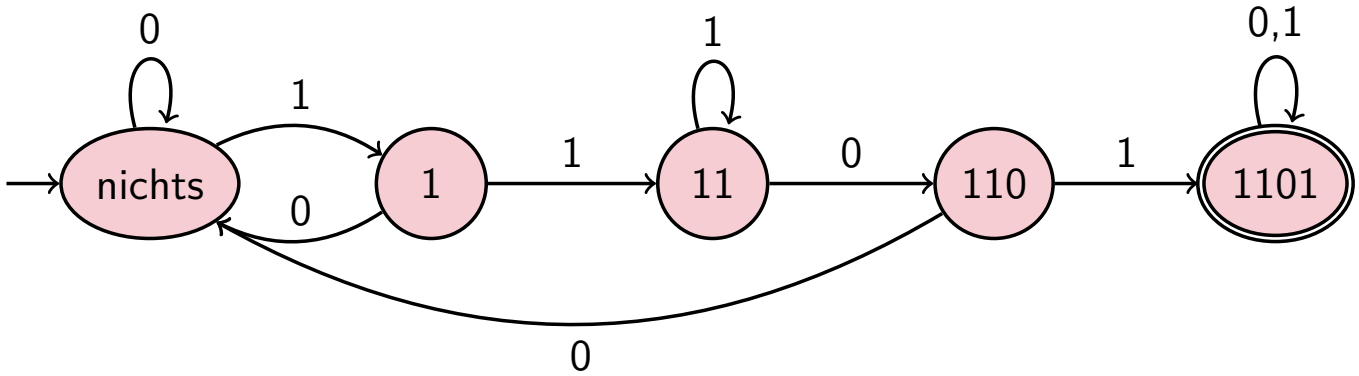
Dann gilt $L(\mathcal{A}) = \{w \in \{0,1\}^* \mid 1101 \text{ ist ein Infix von } w\}$.

Akzeptierender Lauf von $\mathcal{A}$ auf dem Wort $01111010 \in L(\mathcal{A})$:

(nichts, 0, nichts, 1, 1, 1, 11, 1, 11, 1, 11, 0, 110, 1, 1101, 0, 1101)

Beispiel 1.4 (Forts.)

Sei $\mathcal{A}$ der Automat



Dann gilt $L(\mathcal{A}) = \{w \in \{0,1\}^* \mid 1101 \text{ ist ein Infix von } w\}$.

Akzeptierender Lauf von $\mathcal{A}$ auf dem Wort $01111010 \in L(\mathcal{A})$:

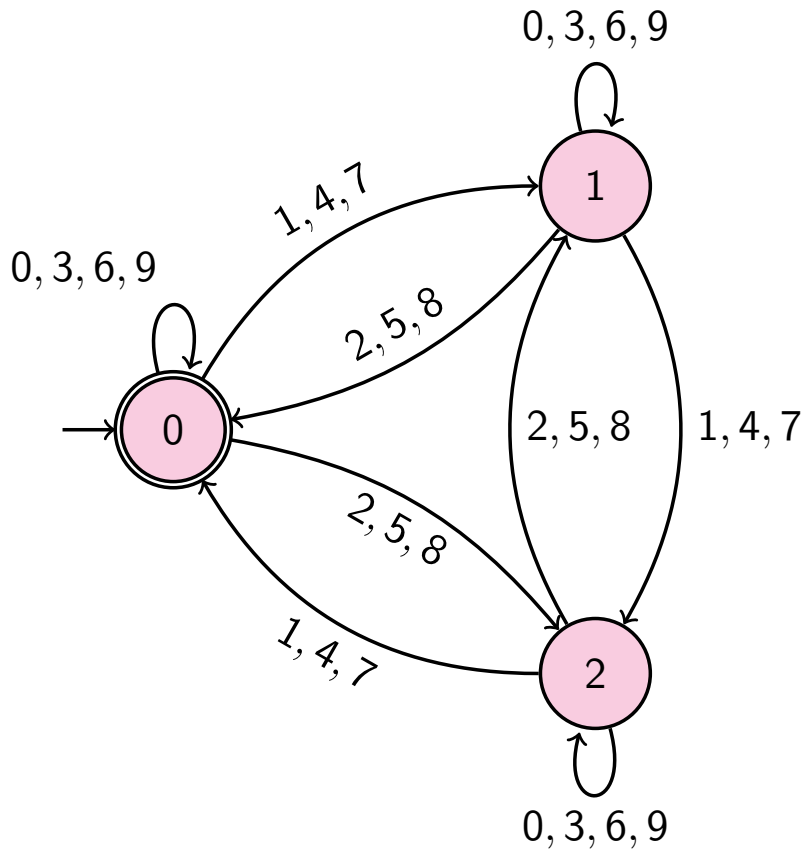
(nichts, 0, nichts, 1, 1, 1, 11, 1, 11, 1, 11, 0, 110, 1, 1101, 0, 1101)

Lauf von $\mathcal{A}$ auf dem Wort $01111001 \notin L(\mathcal{A})$:

(nichts, 0, nichts, 1, 1, 1, 11, 1, 11, 1, 11, 0, 110, 0, nichts, 1, 1)

Beispiel 1.5 (Forts.)

Sei $\mathcal{A}$ der Automat



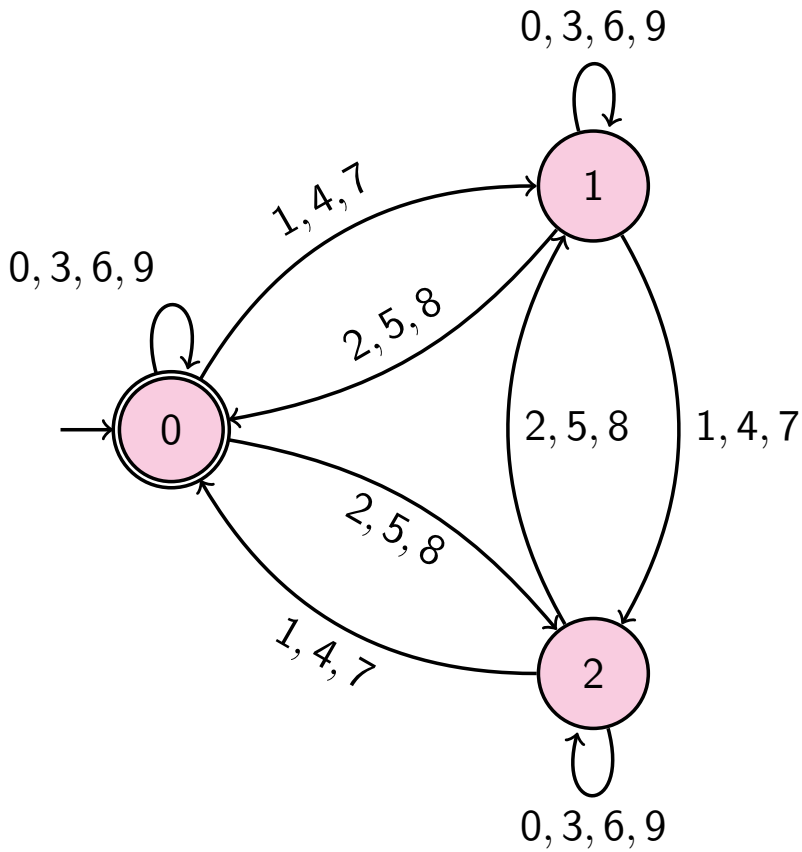
Beispiel 1.5 (Forts.)

Sei $\mathcal{A}$ der Automat

Dann gilt

$L(\mathcal{A}) = \{w \in \{0,\ldots,9\}^* \mid$
w (als Dezimalzahl) ist
durch 3 teilbar$\}$

(wobei wir das leere Wort ε als
die Zahl 0 lesen).



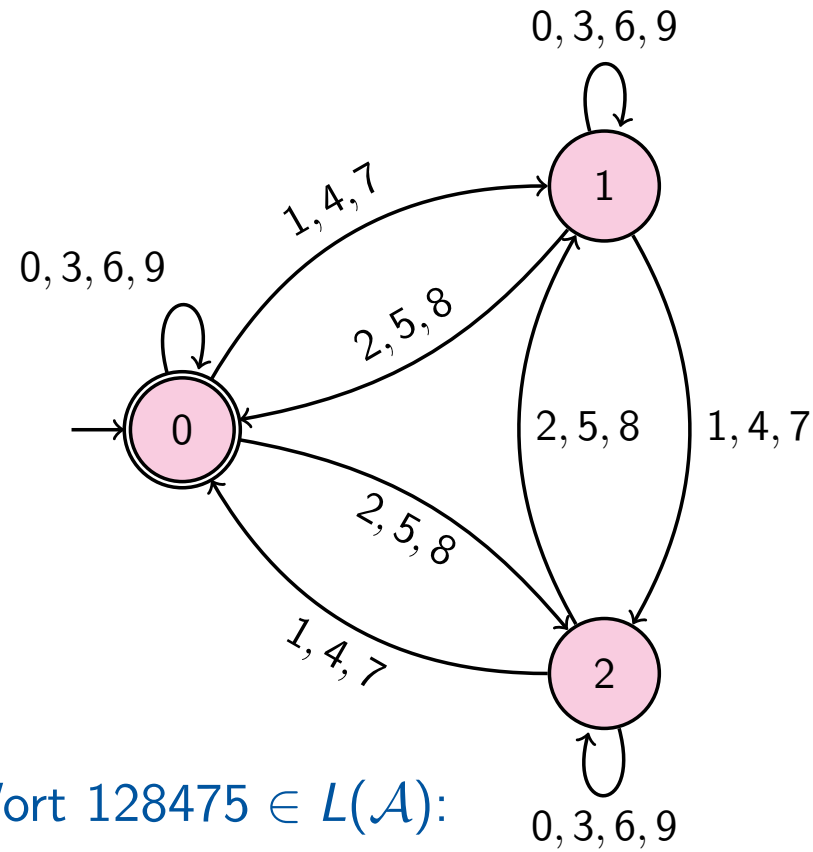
Beispiel 1.5 (Forts.)

Sei $\mathcal{A}$ der Automat

Dann gilt

$L(\mathcal{A}) = \{w \in \{0,\ldots,9\}^* \mid$
w (als Dezimalzahl) ist
durch 3 teilbar$\}$

(wobei wir das leere Wort ε als
die Zahl 0 lesen).

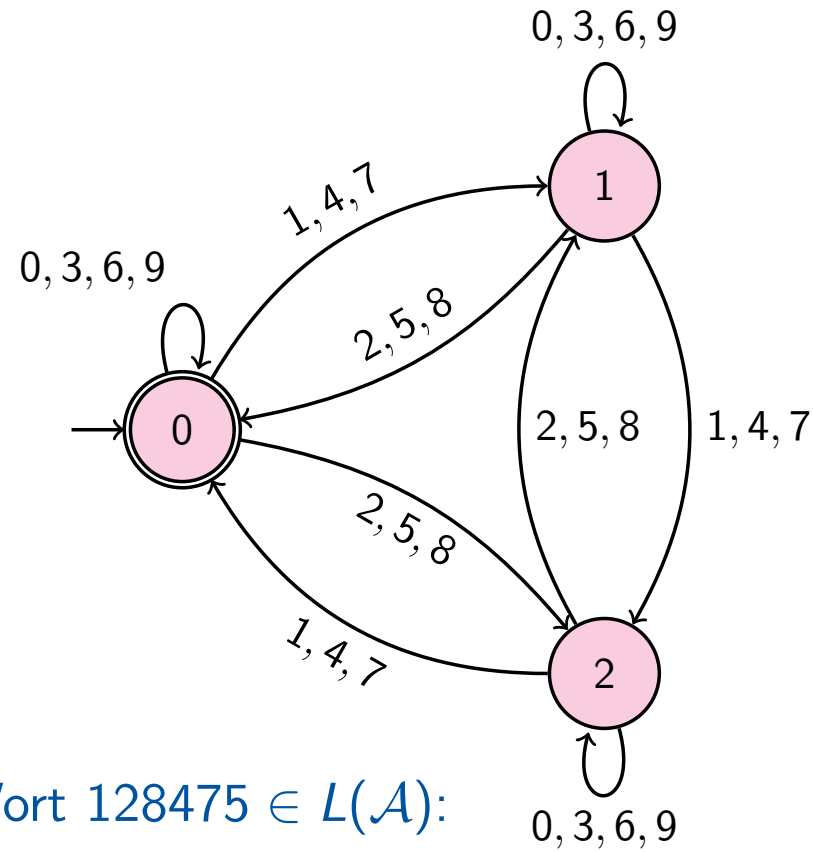


Akzeptierender Lauf von $\mathcal{A}$ auf dem Wort 128475 $\in L(\mathcal{A})$:

(0, 1, 1, 2, 0, 8, 2, 4, 0, 7, 1, 5, 0)

Beispiel 1.5 (Forts.)

Sei $\mathcal{A}$ der Automat



Dann gilt

$L(\mathcal{A}) = \{w \in \{0,\ldots,9\}^* \mid$
w (als Dezimalzahl) ist
durch 3 teilbar$\}$

(wobei wir das leere Wort ε als
die Zahl 0 lesen).

Akzeptierender Lauf von $\mathcal{A}$ auf dem Wort $128475 \in L(\mathcal{A})$:

$$(0,\ 1,\ 1,\ 2,\ 0,\ 8,\ 2,\ 4,\ 0,\ 7,\ 1,\ 5,\ 0)$$

Lauf von $\mathcal{A}$ auf dem Wort $228475 \notin L(\mathcal{A})$:

$$(0,\ 2,\ 2,\ 2,\ 1,\ 8,\ 0,\ 4,\ 1,\ 7,\ 2,\ 5,\ 1)$$

Beispiel 1.19: Teilbarkeit durch 3 binär

Gesucht

Ein DFA, der die Sprache

$$L := \{w \in \{0,1\}^* \mid w \text{ (als Binärzahl) ist durch 3 teilbar}\}$$

erkennt.

Beispiel 1.19: Teilbarkeit durch 3 binär

Gesucht

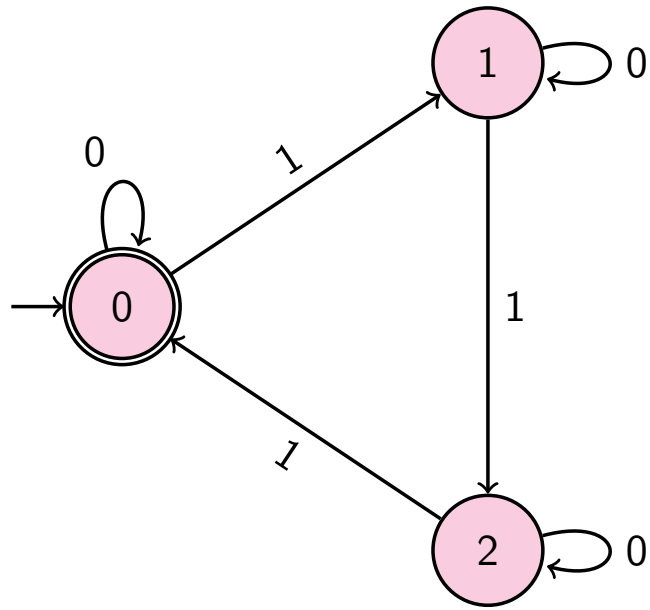
Ein DFA, der die Sprache

$$L := \{w \in \{0,1\}^* \mid w \text{ (als Binärzahl) ist durch 3 teilbar}\}$$

erkennt.

Erster Ansatz

$\mathcal{A}_1$ sei folgender DFA:



Beispiel 1.19: Teilbarkeit durch 3 binär

Gesucht

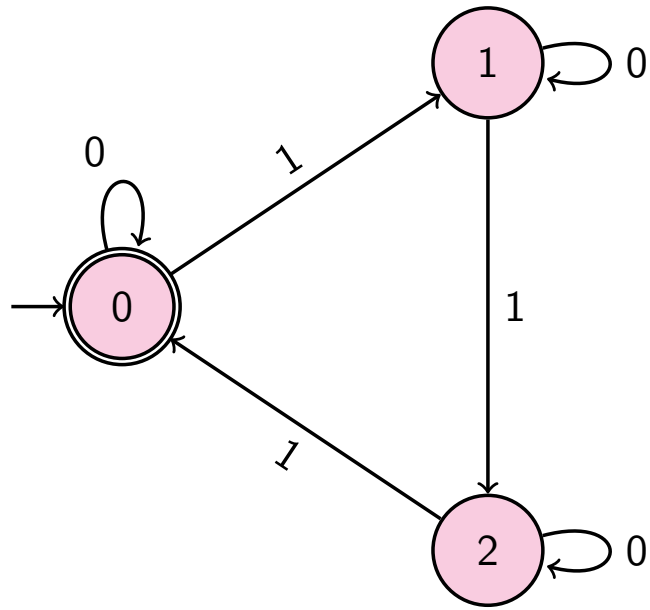
Ein DFA, der die Sprache

$$L := \{w \in \{0,1\}^* \mid w \text{ (als Binärzahl) ist durch 3 teilbar}\}$$

erkennt.

Erster Ansatz

$\mathcal{A}_1$ sei folgender DFA:



Leider gilt $L(\mathcal{A}_1) \neq L$, denn

$$11 \in L \setminus L(\mathcal{A}_1).$$

Beispiel 1.19: Teilbarkeit durch 3 binär

Gesucht

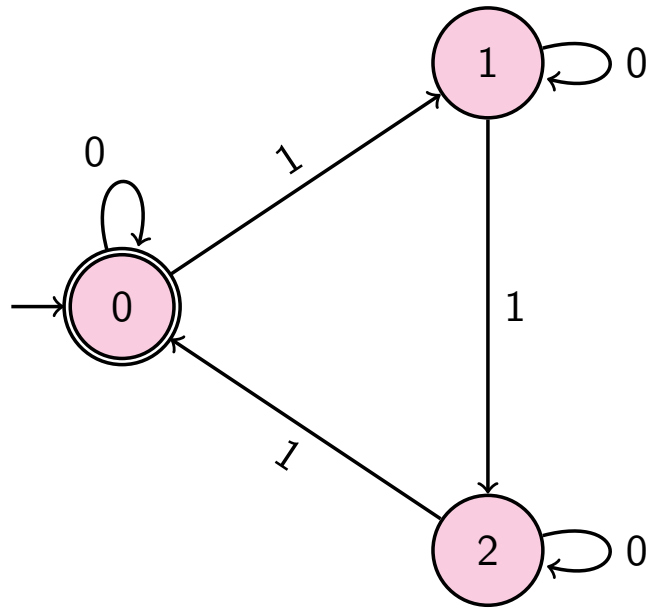
Ein DFA, der die Sprache

$$L := \{w \in \{0,1\}^* \mid w \text{ (als Binärzahl) ist durch 3 teilbar}\}$$

erkennt.

Erster Ansatz

$\mathcal{A}_1$ sei folgender DFA:



Leider gilt $L(\mathcal{A}_1) \neq L$, denn

$$11 \in L \setminus L(\mathcal{A}_1).$$

Lauf von $\mathcal{A}_1$ auf 11: (0, 1, 1, 1, 2).

Beispiel 1.19: Teilbarkeit durch 3 binär

Gesucht

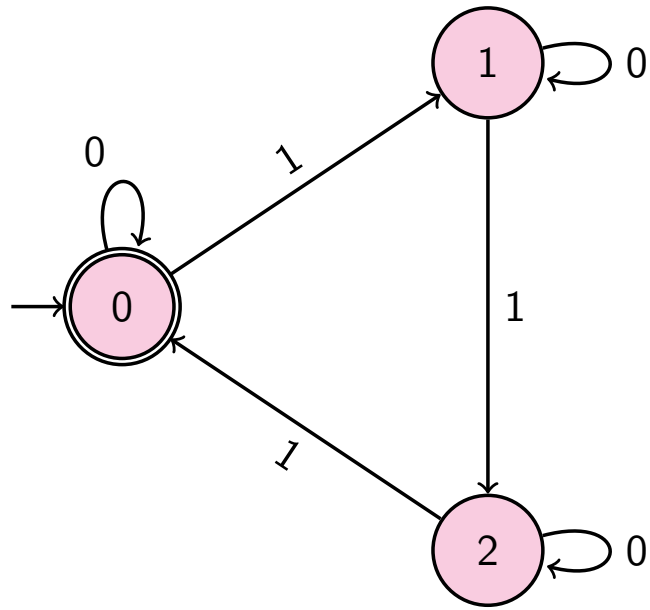
Ein DFA, der die Sprache

$$L := \{w \in \{0,1\}^* \mid w \text{ (als Binärzahl) ist durch 3 teilbar}\}$$

erkennt.

Erster Ansatz

$\mathcal{A}_1$ sei folgender DFA:



Leider gilt $L(\mathcal{A}_1) \neq L$, denn

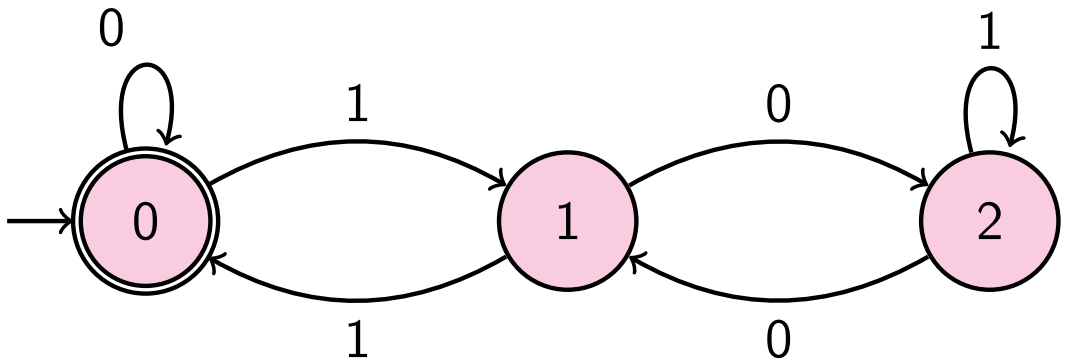
$$11 \in L \setminus L(\mathcal{A}_1).$$

Lauf von $\mathcal{A}_1$ auf 11: (0, 1, 1, 1, 2).

Ist L überhaupt DFA-erkennbar?

Zweiter Ansatz

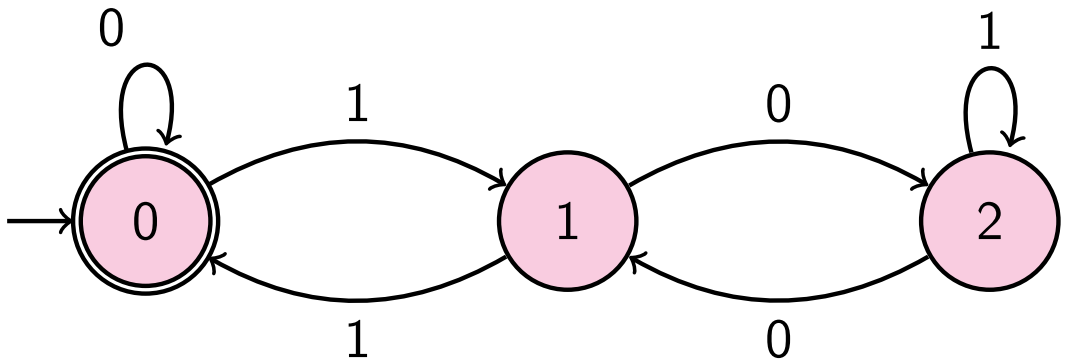
$\mathcal{A}_2$ sei folgender DFA:



Beispiel 1.19 (Forts.)

Zweiter Ansatz

$\mathcal{A}_2$ sei folgender DFA:

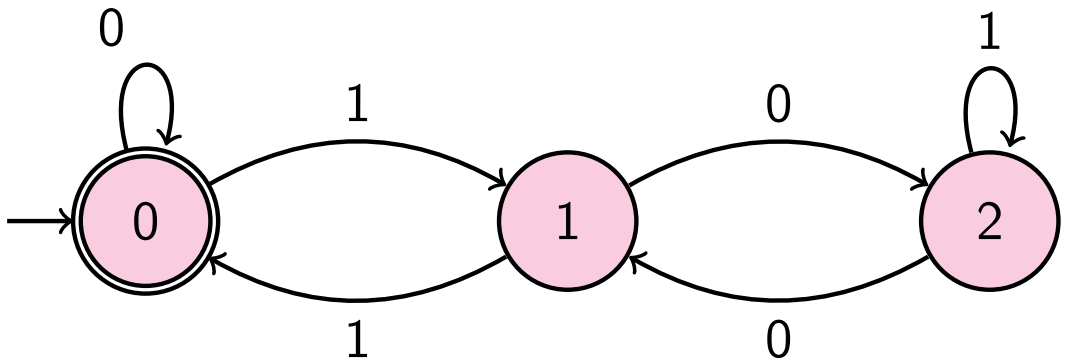


Akzeptierender Lauf von $\mathcal{A}_2$ auf $11 \in L$: $(0, 1, 1, 1, 0)$

Beispiel 1.19 (Forts.)

Zweiter Ansatz

$\mathcal{A}_2$ sei folgender DFA:



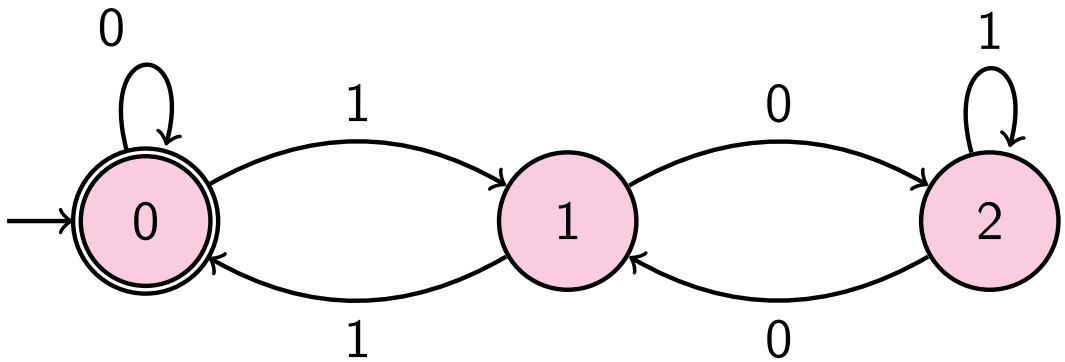
Akzeptierender Lauf von $\mathcal{A}_2$ auf $11 \in L$: $(0, 1, 1, 1, 0)$

Akzeptierender Lauf von $\mathcal{A}_2$ auf $\overbrace{10101}^{\text{21 binär}} \in L$: $(0, 1, 1, 0, 2, 1, 2, 0, 1, 1, 0)$

Beispiel 1.19 (Forts.)

Zweiter Ansatz

$\mathcal{A}_2$ sei folgender DFA:



Akzeptierender Lauf von $\mathcal{A}_2$ auf $11 \in L$: $(0, 1, 1, 1, 0)$

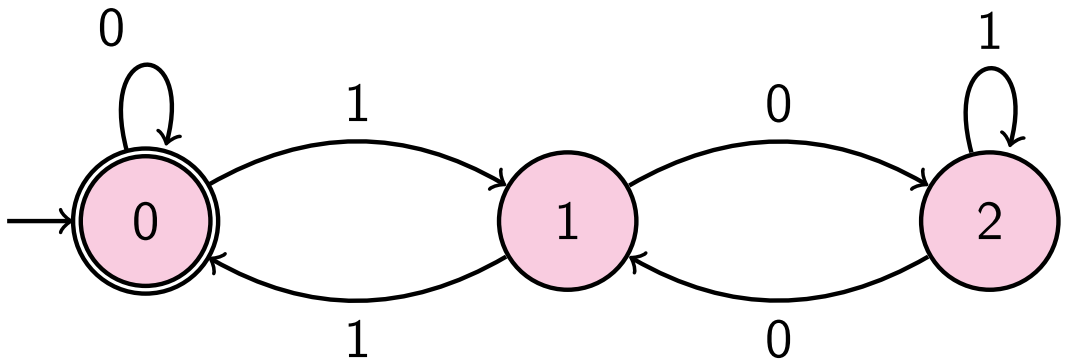
Akzeptierender Lauf von $\mathcal{A}_2$ auf $\overbrace{10101}^{21 \text{ binär}} \in L$: $(0, 1, 1, 0, 2, 1, 2, 0, 1, 1, 0)$

Lauf von $\mathcal{A}_2$ auf $\overbrace{1001010}^{74 \text{ binär}} \notin L$: $(0, 1, 1, 0, 2, 0, 1, 1, 0, 0, 0, 1, 1, 0, 2)$

Beispiel 1.19 (Forts.)

Zweiter Ansatz

$\mathcal{A}_2$ sei folgender DFA:



Akzeptierender Lauf von $\mathcal{A}_2$ auf $11 \in L$: $(0, 1, 1, 1, 0)$

Akzeptierender Lauf von $\mathcal{A}_2$ auf $\overbrace{10101}^{21 \text{ binär}} \in L$: $(0, 1, 1, 0, 2, 1, 2, 0, 1, 1, 0)$

Lauf von $\mathcal{A}_2$ auf $\overbrace{1001010}^{74 \text{ binär}} \notin L$: $(0, 1, 1, 0, 2, 0, 1, 1, 0, 0, 0, 1, 1, 0, 2)$

Behauptung

$$L(\mathcal{A}_2) = L.$$

Für $w \in \{0,1\}^*$ sei $b(w) \in \mathbb{N}$ der Wert von w als Binärzahl.

Beispiele: $b(10101) = 21, \quad b(1001010) = 74, \quad b(\varepsilon) = 0$

Formal ist für $w = a_1 \dots a_n$

$$b(w) = \sum_{i=1}^{n} a_i \cdot 2^{n-i}.$$

Lemma
Seien $w \in \{0,1\}^*$ und $a \in \{0,1\}$. Dann gilt

$$b(wa) = 2 \cdot b(w) + a.$$

Beweis.
Sei $w = a_1 \dots a_n$. Dann ist

$$b(wa) = \sum_{i=1}^{n} a_i \cdot 2^{n+1-i} + a \cdot 2^0 = 2 \cdot \sum_{i=1}^{n} a_i \cdot 2^{n-i} + a = 2 \cdot b(w) + a.$$

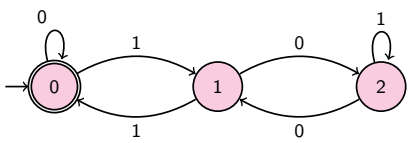
□

Erinnerung
Für $x, y, z \in \mathbb{N}$, $z \neq 0$ schreiben wir

$$x \equiv y \mod z,$$

wenn „x kongruent zu y modulo z'", d.h., $x = kz + y$ für ein $k \in \mathbb{N}$.
Wenn $0 \leq y < z$ bedeutet dies, dass y der Rest von x bei ganzzahliger Division durch z ist.

$\mathcal{A}_2$



Behauptung (umformuliert)
$\mathcal{A}_2$ akzeptiert $w \in \{0,1\}^* \iff b(w) \equiv 0 \mod 3$.

Beweis.
Per vollständiger Induktion über $n \geq 0$ zeigen wir für alle Wörter $w = a_1 \dots a_n \in \Sigma^*$:

Ist $(r_0, r_1, \dots, r_n)$ der Lauf von $\mathcal{A}_2$ auf w, so gilt

$$r_n \equiv b(w) \mod 3.$$

Induktionsanfang: $n = 0$.
Der Lauf von $\mathcal{A}_2$ auf $w = \varepsilon$ ist (0), und es gilt $b(\varepsilon) = 0$.

Induktionsschritt: $n \to n+1$.
Sei $(r_0, \dots, r_{n+1})$ der Lauf von $\mathcal{A}_2$ auf $w = a_1 \dots a_{n+1}$.

Induktionsannahme (IA): $r_n \equiv b(a_1 \dots a_n) \mod 3$.

Behauptung: $r_{n+1} \equiv b(a_1 \dots a_{n+1}) \mod 3$.

Beweis: Nach dem Lemma gilt

$$b(w) = b(a_1 \dots a_{n+1}) = 2b(a_1 \dots a_n) + a_{n+1}.$$

Nach (IA) also $b(w) \equiv 2r_n + a_{n+1} \mod 3$.

Fall 1: $r_n = 0, a_{n+1} = 0$.
Dann gilt $b(w) \equiv 2r_n + a_{n+1} \equiv 0 \mod 3$ und $r_{n+1} = \delta(0,0) = 0$.
(δ bezeichnet die Transitionsfunktion von $\mathcal{A}_2$.)

Fall 2: $r_n = 0, a_{n+1} = 1$.
Dann gilt $b(w) \equiv 1 \mod 3$ und $r_{n+1} = \delta(0,1) = 1$.

Fall 3: $r_n = 1, a_{n+1} = 0$.
Dann gilt $b(w) \equiv 2 \mod 3$ und $r_{n+1} = \delta(1,0) = 2$.

Fall 4: $r_n = 1, a_{n+1} = 1$.
Dann gilt $b(w) \equiv 3 \equiv 0 \mod 3$ und $r_{n+1} = \delta(1,1) = 0$.

Fall 5: $r_n = 2, a_{n+1} = 0$.
Dann gilt $b(w) \equiv 4 \equiv 1 \mod 3$ und $r_{n+1} = \delta(2,0) = 1$.

Fall 6: $r_n = 2, a_{n+1} = 1$.
Dann gilt $b(w) \equiv 5 \equiv 2 \mod 3$ und $r_{n+1} = \delta(2,1) = 2$.

Damit ist die Behauptung bewiesen. □

Abschnitt 1.4

Operationen auf Sprachen

Notation (für den ganzen Abschnitt 1.4)

Im ganzen Abschnitt halten wir ein Alphabet Σ fest.
K, L, M und Varianten wie K', L_1 stehen für Sprachen über Σ.

Notation (für den ganzen Abschnitt 1.4)

Im ganzen Abschnitt halten wir ein Alphabet Σ fest.
K, L, M und Varianten wie K', L_1 stehen für Sprachen über Σ.

Wie für alle Mengen können wir auch für Sprachen Durchschnitt, Vereinigung, und mengentheoretische Differenz bilden:

$$K \cup L, \quad K \cap L, \quad K \setminus L.$$

Notation (für den ganzen Abschnitt 1.4)

Im ganzen Abschnitt halten wir ein Alphabet Σ fest.
K, L, M und Varianten wie K', L_1 stehen für Sprachen über Σ.

Wie für alle Mengen können wir auch für Sprachen Durchschnitt, Vereinigung, und mengentheoretische Differenz bilden:

$$K \cup L, \quad K \cap L, \quad K \setminus L.$$

Bei gegebenem Σ können wir außerdem das Komplement in Σ^* bilden:

$$\overline{L} := \Sigma^* \setminus L.$$

Notation (für den ganzen Abschnitt 1.4)

Im ganzen Abschnitt halten wir ein Alphabet Σ fest.
K, L, M und Varianten wie K', L_1 stehen für Sprachen über Σ.

Wie für alle Mengen können wir auch für Sprachen Durchschnitt, Vereinigung, und mengentheoretische Differenz bilden:

$$K \cup L, \quad K \cap L, \quad K \setminus L.$$

Bei gegebenem Σ können wir außerdem das Komplement in Σ^* bilden:

$$\overline{L} := \Sigma^* \setminus L.$$

Bekanntlich gelten für die mengentheoretischen Operationen die Kommutativ-, Assoziativ- und Distributivgesetze sowie die de Morgan'schen Regeln.

$K \cup L = \overline{(\overline{K} \cap \overline{L})}$, $K \cap L = \overline{(\overline{K} \cup \overline{L})}$

Erinnerung

uv bezeichnet die Verkettung der Wörter $u, v \in \Sigma^*$.
Für $u = a_1 \dots a_m$ und $v = b_1 \dots b_n$ ist also $uv = a_1 \dots a_m b_1 \dots b_n$.

Erinnerung

uv bezeichnet die Verkettung der Wörter $u, v \in \Sigma^*$.
Für $u = a_1 \dots a_m$ und $v = b_1 \dots b_n$ ist also $uv = a_1 \dots a_m b_1 \dots b_n$.

Definition 1.20

Die Verkettung zweier Sprachen $K, L \subseteq \Sigma^*$ ist die Sprache

$$KL := \{uv \mid u \in K, v \in L\} \subseteq \Sigma^*.$$

Erinnerung

uv bezeichnet die Verkettung der Wörter $u, v \in \Sigma^*$.
Für $u = a_1 \dots a_m$ und $v = b_1 \dots b_n$ ist also $uv = a_1 \dots a_m b_1 \dots b_n$.

Definition 1.20

Die Verkettung zweier Sprachen $K, L \subseteq \Sigma^*$ ist die Sprache

$$KL := \{uv \mid u \in K, v \in L\} \subseteq \Sigma^*.$$

Beispiel 1.21

Für $K = \{101, 1\}$ und $L = \{011, 1\}$:

$$KL = \{101011, 1011, 11\}$$

Rechenregeln für die Verkettung

nicht kommutativ!
i.A. $KL \neq LK$

Satz 1.22

Für alle Sprachen K, L, M gilt

1. $(KL)M = K(LM)$, *(Assoziativgesetz)*
2. $L\{\varepsilon\} = \{\varepsilon\}L = L$,
3. $L\emptyset = \emptyset L = \emptyset$,
4. $K(L \cup M) = KL \cup KM$,
   $(K \cup L)M = KM \cup LM$. *(Distributivgesetze)*

Rechenregeln für die Verkettung

Satz 1.22

Für alle Sprachen K, L, M gilt

1. $(KL)M = K(LM)$, *(Assoziativgesetz)*
2. $L\{\varepsilon\} = \{\varepsilon\}L = L$,
3. $L\emptyset = \emptyset L = \emptyset$,
4. $K(L \cup M) = KL \cup KM$,
   $(K \cup L)M = KM \cup LM$. *(Distributivgesetze)*

Eigentlich müssten wir all diese Rechenregeln beweisen; exemplarisch beweisen wir nur das erste Distributivgesetz.

Seien $K, L, M \subseteq \Sigma^*$.

Behauptung 1

$$K(L \cup M) = KL \cup KM.$$

Um die Gleichheit der Mengen zu beweisen, beweisen wir, dass die erste in der zweiten und anschließend dass die zweite in der ersten enthalten ist.

Behauptung 1A

$$K(L \cup M) \subseteq KL \cup KM.$$

Beweis.
Wir zeigen, dass ein beliebiges (und damit jedes) Element von $K(L \cup M)$ in $KL \cup KM$ enthalten ist.
Sei $u \in K(L \cup M)$. Dann gibt es ein $v \in K$ und ein $w \in L \cup M$, so dass $u = vw$. w ist in $L \cup M$, also in L oder in M.

Fall 1: $w \in L$.
Dann ist $u = vw \in KL \subseteq KL \cup KM$.

Fall 2: $w \in M$.
Dann ist $u = vw \in KM \subseteq KL \cup KM$.

In beiden Fällen ist also $u \in KL \cup KM$. □

Behauptung 1B

$$K(L \cup M) \supseteq KL \cup KM.$$

Beweis.
Sei $u \in KL \cup KM$.

Fall 1: $u \in KL$.
Dann gibt es $v \in K, w \in L$, so dass $u = vw$. Weil $w \in L \cup M$ ist also $u \in K(L \cup M)$.

Fall 2: $u \in KM$.
Dann gibt es $v \in K, w \in M$, so dass $u = vw$. Weil $w \in L \cup M$ ist also $u \in K(L \cup M)$.

In beiden Fällen ist $u \in K(L \cup M)$. □

Die Behauptungen 1A und 1B implizieren sofort Behauptung 1. □

Definition 1.23

Sei $L \subseteq \Sigma^*$. Die Potenzen L^n, für $n \in \mathbb{N}$, sind induktiv wie folgt definiert:

$$L^0 := \{\varepsilon\},$$
$$L^{n+1} := LL^n \qquad \text{für alle } n \in \mathbb{N}.$$

Definition 1.23

Sei $L \subseteq \Sigma^*$. Die Potenzen L^n, für $n \in \mathbb{N}$, sind induktiv wie folgt definiert:

$$L^0 := \{\varepsilon\},$$
$$L^{n+1} := LL^n \qquad \text{für alle } n \in \mathbb{N}.$$

Beispiel 1.24

Für $L = \{ab, aab\}$ sind

// L (sin für ML)

$$L^0 = \{\varepsilon\}, \qquad L^1 = \{ab, aab\}, \qquad L^2 = \{abab, abaab, aabab, aabaab\},$$
$$L^3 = \{ababab, ababaab, abaabab, abaabaab, aababab, aababaab, \ldots\}.$$

Definition 1.23

Sei $L \subseteq \Sigma^*$. Die Potenzen L^n, für $n \in \mathbb{N}$, sind induktiv wie folgt definiert:

$$L^0 := \{\varepsilon\},$$
$$L^{n+1} := LL^n \qquad \text{für alle } n \in \mathbb{N}.$$

Beispiel 1.24

Für $L = \{ab, aab\}$ sind

$$L^0 = \{\varepsilon\}, \qquad L^1 = \{ab, aab\}, \qquad L^2 = \{abab, abaab, aabab, aabaab\},$$
$$L^3 = \{ababab, ababaab, abaabab, abaabaab, aababab, aababaab, \ldots\}.$$

Beobachtung 1.25

Die n-te Potenz L^n von L ist die n-fache Verkettung von L mit sich selbst. Das heißt, ein Wort u ist genau dann in L^n, wenn es Worte $v_1, \ldots, v_n \in L$ gibt, so dass $u = v_1 \ldots v_n$.

Definition 1.26

Die Iteration (auch Kleene-Stern) einer Sprache L ist die Sprache

$$L^* := \bigcup_{n \in \mathbb{N}} L^n$$

Definition 1.26

Die Iteration (auch Kleene-Stern) einer Sprache L ist die Sprache

$$L^* := \bigcup_{n \in \mathbb{N}} L^n$$

Beobachtung 1.27

1. $\varepsilon \in L^0 \subseteq L^*$.
2. *Ein Wort $u \neq \varepsilon$ liegt genau dann in L^*, wenn es ein $n \geq 1$ und Worte $v_1, \ldots, v_n \in L$ gibt, so dass $u = v_1 \ldots v_n$.*

Satz 1.28

Für alle $L \subseteq \Sigma^*$ gilt:

1. $L^*L^* = L^*$,
2. $(L^*)^* = L^*$,
3. $L^* = \{\varepsilon\} \cup LL^* = \{\varepsilon\} \cup L^*L$,
4. $\emptyset^* = \{\varepsilon\}$.

(Beweise als Übung.)

Abschnitt 1.5

Abschlusseigenschaften DFA-erkennbarer Sprachen

Komplement

Satz 1.29

Sei $L \subseteq \Sigma^*$ DFA-erkennbar. Dann ist auch $\overline{L}$ DFA-erkennbar.

Beispiel 1.30

Sei $L = \{w \in \{0, \ldots, 9\}^* \mid w \text{ als Dezimalzahl ist durch 3 teilbar}\}$.

Beispiel 1.30

Sei $L = \{w \in \{0, \ldots, 9\}^* \mid w \text{ als Dezimalzahl ist durch 3 teilbar}\}$.

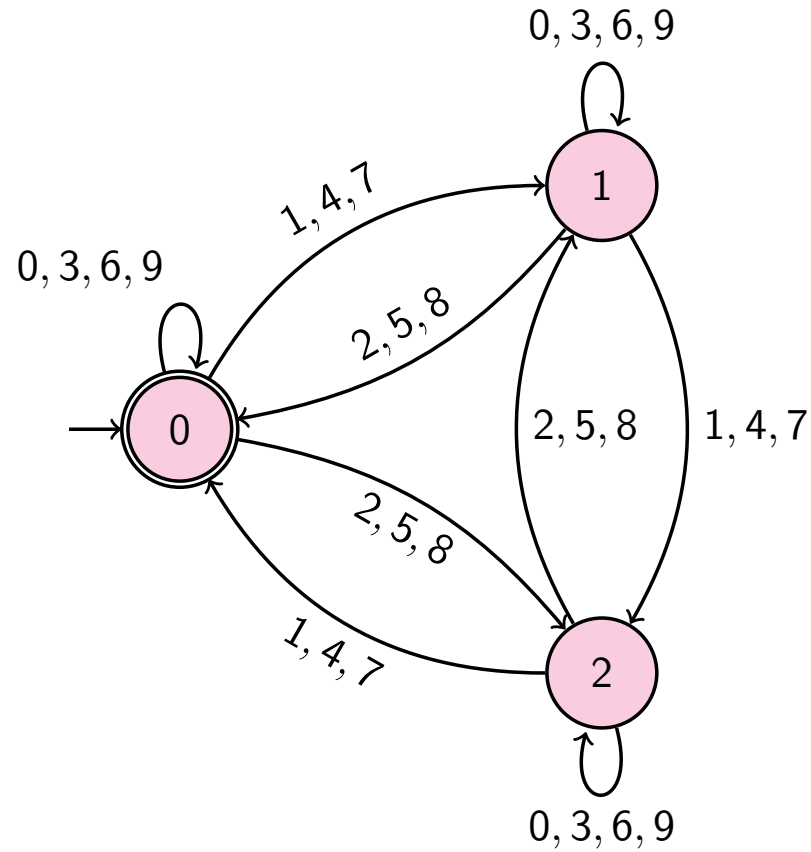
Dann ist $\overline{L} = \{w \in \{0, \ldots, 9\}^* \mid w \text{ als Dezimalzahl ist nicht durch 3 teilbar}\}$.

Beispiel 1.30

Sei $L = \{w \in \{0,\ldots,9\}^* \mid w \text{ als Dezimalzahl ist durch 3 teilbar}\}$.

Dann ist $\overline{L} = \{w \in \{0,\ldots,9\}^* \mid w \text{ als Dezimalzahl ist nicht durch 3 teilbar}\}$.

DFA für L:

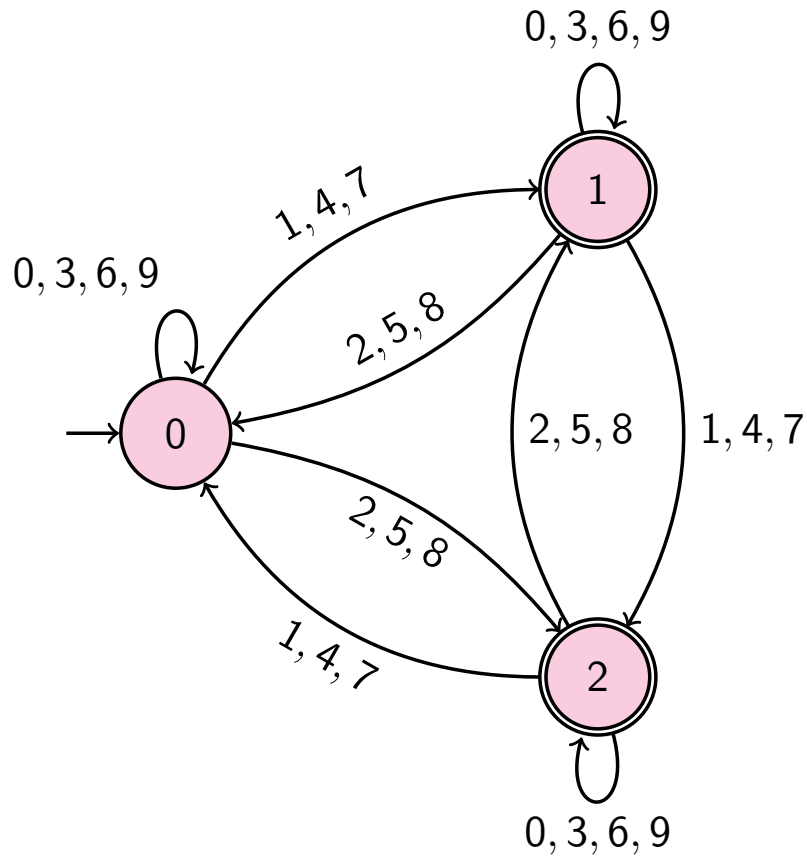


Beispiel 1.30

Sei $L = \{w \in \{0,\ldots,9\}^* \mid w \text{ als Dezimalzahl ist durch 3 teilbar}\}$.

Dann ist $\overline{L} = \{w \in \{0,\ldots,9\}^* \mid w \text{ als Dezimalzahl ist nicht durch 3 teilbar}\}$.

DFA für $\overline{L}$:



Satz 1.29
Sei $L \subseteq \Sigma^*$ DFA-erkennbar. Dann ist auch $\overline{L}$ DFA-erkennbar.

Beweis.
Sei $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ ein DFA mit $L(\mathcal{A}) = L$.
Sei $\overline{\mathcal{A}}$ der DFA, der aus $\mathcal{A}$ durch Vertauschen von Endzuständen und Nicht-Endzuständen entsteht, also

$$\overline{\mathcal{A}} := (Q, \Sigma, \delta, q_0, Q \setminus F).$$

Behauptung

$$L(\overline{\mathcal{A}}) = \overline{L}.$$

Offensichtlich folgt der Satz aus der Behauptung.

Beweis der Behauptung.
Sei $w = a_1 \dots a_n \in \Sigma^*$.

Wir zeigen:

$$\mathcal{A} \text{ akzeptiert } w \iff \overline{\mathcal{A}} \text{ akzeptiert } w \text{ nicht.} \qquad (\star)$$

Daraus folgt die Behauptung.

Weil $\mathcal{A}$ und $\overline{\mathcal{A}}$ den gleichen Anfangszustand und die gleiche Transitionsfunktion haben, haben sie den gleichen Lauf auf dem Wort w.
Sei $(r_0 \dots r_n)$ dieser Lauf. Dann gilt

$$\begin{aligned} \mathcal{A} \text{ akzeptiert } w &\iff r_n \in F \\ &\iff r_n \notin Q \setminus F \\ &\iff \overline{\mathcal{A}} \text{ akzeptiert } w \text{ nicht.} \end{aligned}$$

Damit ist $(\star)$ bewiesen. □

Durchschnitt

Satz 1.31

Seien $L_1, L_2 \subseteq \Sigma^*$ DFA-erkennbar. Dann ist auch $L_1 \cap L_2$ DFA-erkennbar.

Parallele Ausführung und Produktautomaten

Betrachte DFAs

$$\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1) \quad \text{und} \quad \mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2).$$

Beachte: Beide haben dasselbe Alphabet.

Parallele Ausführung und Produktautomaten

Betrachte DFAs

$$\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1) \quad \text{und} \quad \mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2).$$

Beachte: Beide haben dasselbe Alphabet.

Ziel

Konstruiere DFA, der $\mathcal{A}_1$ und $\mathcal{A}_2$ (synchron) parallel ausführt.

Parallele Ausführung und Produktautomaten

Betrachte DFAs

$$\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1) \quad \text{und} \quad \mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2).$$

Beachte: Beide haben dasselbe Alphabet.

Ziel

Konstruiere DFA, der $\mathcal{A}_1$ und $\mathcal{A}_2$ (synchron) parallel ausführt.

Produktkonstruktion

$$(Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F),$$

wobei $\delta : (Q_1 \times Q_2) \times \Sigma \to Q_1 \times Q_2$ definiert ist durch

$$\delta\big((r_1, r_2), a\big) := \big(\delta_1(r_1, a), \delta_2(r_2, a)\big)$$

nennen wir einen Produktautomaten von $\mathcal{A}_1$ und $\mathcal{A}_2$.

Zur Wahl der Endzustände

Ein Produktautomat

$$\mathcal{A} = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$$

zweier DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$ ist bis auf die Wahl der Menge $F \subseteq Q_1 \times Q_2$ der Endzutände eindeutig bestimmt.

Zur Wahl der Endzustände

Ein Produktautomat

$$\mathcal{A} = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$$

zweier DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$ ist bis auf die Wahl der Menge $F \subseteq Q_1 \times Q_2$ der Endzutände eindeutig bestimmt.

Abhängig vom Kontext kann es verschiedene sinnvolle Wahlen von F geben, beispielsweise

$$F = F_1 \times F_2, \qquad F = (Q_1 \times F_2) \cup (F_1 \times Q_2), \qquad F = F_1 \times (Q_2 \setminus F_2).$$

Zur Wahl der Endzustände

Ein Produktautomat

$$\mathcal{A} = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$$

zweier DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$ ist bis auf die Wahl der Menge $F \subseteq Q_1 \times Q_2$ der Endzutände eindeutig bestimmt.

Abhängig vom Kontext kann es verschiedene sinnvolle Wahlen von F geben, beispielsweise

$$F = F_1 \times F_2, \qquad F = (Q_1 \times F_2) \cup (F_1 \times Q_2), \qquad F = F_1 \times (Q_2 \setminus F_2).$$

Spricht man von „dem“ Produktautomaten, so meint man in der Regel den mit $F = F_1 \times F_2$.

Beispiel 1.32

Betrachte DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$ mit

$$L(\mathcal{A}_1) = \{w \in \{0,1\}^* \mid \overbrace{b(w)}^{\text{Wert von } w \text{ als Binärzahl}} \equiv 0 \bmod 2\}$$

$$L(\mathcal{A}_2) = \{w \in \{0,1\}^* \mid b(w) \equiv 0 \bmod 3\}$$

Wir konstruieren den Produktautomaten $\mathcal{A}$ mit $F = F_1 \times F_2$.

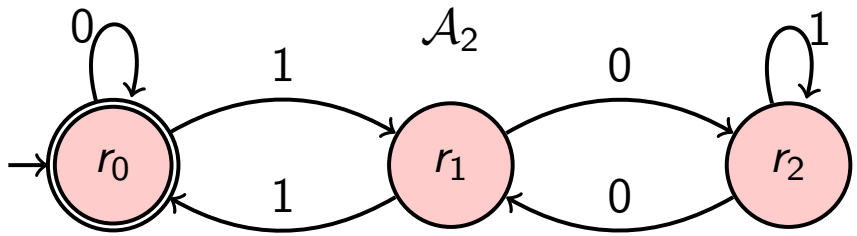
Beispiel 1.32

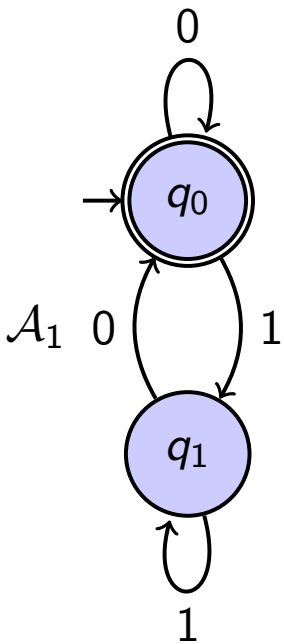
Betrachte DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$ mit

$$L(\mathcal{A}_1) = \{w \in \{0,1\}^* \mid \overbrace{b(w)}^{\text{Wert von } w \text{ als Binärzahl}} \equiv 0 \bmod 2\}$$

$$L(\mathcal{A}_2) = \{w \in \{0,1\}^* \mid b(w) \equiv 0 \bmod 3\}$$

Wir konstruieren den Produktautomaten $\mathcal{A}$ mit $F = F_1 \times F_2$.





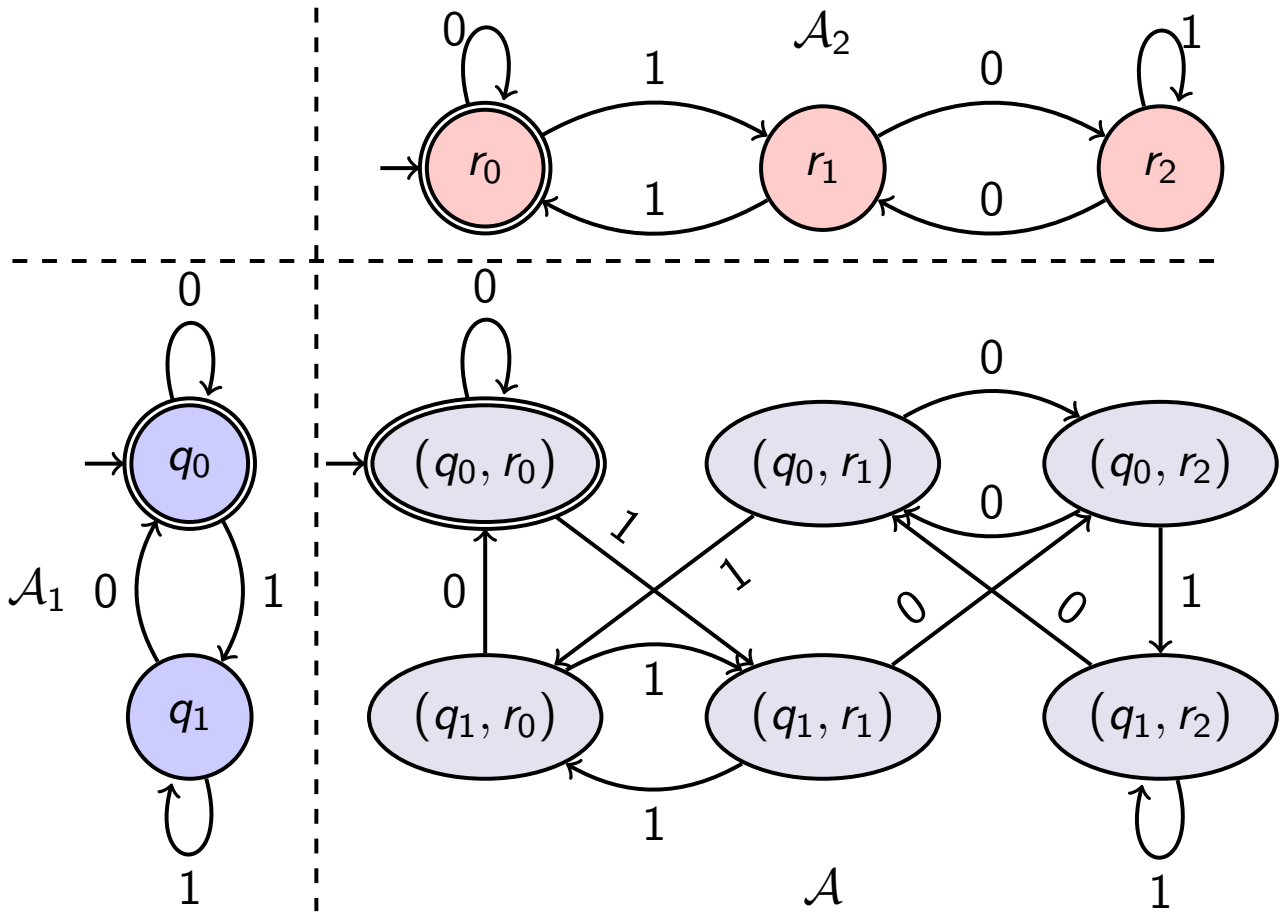
Beispiel 1.32

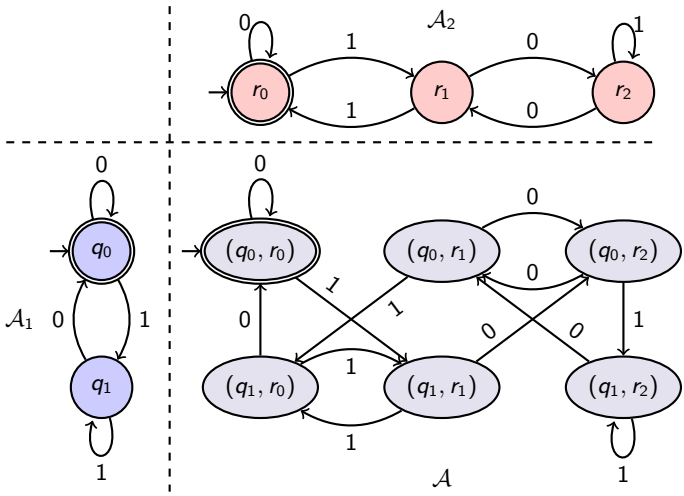
Betrachte DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$ mit

$$L(\mathcal{A}_1) = \{w \in \{0,1\}^* \mid \overbrace{b(w)}^{\text{Wert von } w \text{ als Binärzahl}} \equiv 0 \bmod 2\}$$

$$L(\mathcal{A}_2) = \{w \in \{0,1\}^* \mid b(w) \equiv 0 \bmod 3\}$$

Wir konstruieren den Produktautomaten $\mathcal{A}$ mit $F = F_1 \times F_2$.





Frage

Was ist $L(\mathcal{A})$?

Betrachten wir einige Läufe des Automaten.

1. $w = 110110$ mit $b(w) = 54$.

   Lauf von $\mathcal{A}_1$ auf w

   $$(q_0, 1, q_1, 1, q_1, 0, q_0, 1, q_1, 1, q_1, 0, q_0) \quad \text{akzeptiert.}$$

   Lauf von $\mathcal{A}_2$ auf w

   $$(r_0, 1, r_1, 1, r_0, 0, r_0, 1, r_1, 1, r_0, 0, r_0) \quad \text{akzeptiert.}$$

   Lauf von $\mathcal{A}$ auf w

   $$\big((q_0, r_0), 1, (q_1, r_1), 1, (q_1, r_0), 0, (q_0, r_0), 1, (q_1, r_1), 1, (q_1, r_0), 0, (q_0, r_0)\big)$$

   akzeptiert.

2. $w = 110100$ mit $b(w) = 52$.

   Lauf von $\mathcal{A}_1$ auf w

   $$(q_0, 1, q_1, 1, q_1, 0, q_0, 1, q_1, 0, q_0, 0, q_0) \quad \text{akzeptiert.}$$

   Lauf von $\mathcal{A}_2$ auf w

   $$(r_0, 1, r_1, 1, r_0, 0, r_0, 1, r_1, 0, r_2, 0, r_1) \quad \text{akzeptiert nicht.}$$

   Lauf von $\mathcal{A}$ auf w

   $$\big((q_0, r_0), 1, (q_1, r_1), 1, (q_1, r_0), 0, (q_0, r_0), 1, (q_1, r_1), 0, (q_0, r_2), 0, (q_0, r_1)\big)$$

   akzeptiert nicht.

3. $w = 11$ mit $b(w) = 3$.

   Lauf von $\mathcal{A}_1$ auf w

   $$(q_0, 1, q_1, 1, q_1) \quad \text{akzeptiert nicht.}$$

   Lauf von $\mathcal{A}_2$ auf w

   $$(r_0, 1, r_1, 1, r_0) \quad \text{akzeptiert.}$$

   Lauf von $\mathcal{A}$ auf w

   $$\big((q_0, r_0), 1, (q_1, r_1), 1, (q_1, r_0)\big) \quad \text{akzeptiert nicht.}$$

4. $w = 101$ mit $b(w) = 5$.

   Lauf von $\mathcal{A}_1$ auf w

   $$(q_0, 1, q_1, 0, q_0, 1, q_1) \quad \text{akzeptiert nicht.}$$

   Lauf von $\mathcal{A}_2$ auf w

   $$(r_0, 1, r_1, 0, r_2, 1, r_2) \quad \text{akzeptiert nicht.}$$

   Lauf von $\mathcal{A}$ auf w

   $$\big((q_0, r_0), 1, (q_1, r_1), 0, (q_0, r_2), 1, (q_1, r_2)\big) \quad \text{akzeptiert nicht.}$$

Vermutung

$$L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \{w \in \{0,1\}^* \mid b(w) \equiv 0 \bmod 6\}.$$

Parallele Ausführung

Das folgende Lemma zeigt, dass ein Produktautomat seine beiden Komponenten parallel ausführt.

Das folgende Lemma zeigt, dass ein Produktautomat seine beiden Komponenten parallel ausführt.

Lemma 1.33

Seien $\mathcal{A}_1$ und $\mathcal{A}_2$ DFAs mit Eingabealphabet Σ und $\mathcal{A}$ ein Produktautomat von $\mathcal{A}_1$ und $\mathcal{A}_2$.
Sei $w = a_1 \dots a_n \in \Sigma^*$, und seien $(r_0, \dots, r_n)$ und $(s_0, \dots, s_n)$ die Läufe von $\mathcal{A}_1$ bzw. $\mathcal{A}_2$ auf w. Dann ist

$$\big((r_0, s_0), \dots, (r_n, s_n)\big)$$

der Lauf von $\mathcal{A}$ auf w.

Beweis.
Seien

$$\begin{aligned}\mathcal{A}_1 &= (Q_1, \Sigma, \delta_1, q_{01}, F_1),\\ \mathcal{A}_2 &= (Q_2, \Sigma, \delta_2, q_{02}, F_2),\\ \mathcal{A} &= (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F).\end{aligned}$$

Dann gilt für alle $r \in Q_1, s \in Q_2, a \in \Sigma$:

$$\delta\big((r,s),a\big) = \big(\delta_1(r,a), \delta_2(s,a)\big).$$

Seien

$$\begin{aligned}&(r_0, \ldots, r_n),\\ &(s_0, \ldots, s_n),\\ &(t_0, \ldots, t_n)\end{aligned}$$

die Läufe von $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}$ auf $w = a_1 \ldots a_n$.

Wir zeigen: Für $0 \leq i \leq n$ ist

$$t_i = (r_i, s_i).$$

Wir beweisen das per Induktion über i.

Induktionsanfang $i = 0$.
Nach Definition eines Laufes gilt $r_0 = q_{01}$, $s_0 = q_{02}$ und $t_0 = (q_{01}, q_{02})$.
Also ist $t_0 = (r_0, s_0)$.

Induktionsschritt $i \to i+1$, für $i < n$
Nach Induktionsannahme gilt $t_i = (r_i, s_i)$.
Nach Definition eines Laufes sind

$$r_{i+1} = \delta_1(r_i, a_{i+1}), \quad s_{i+1} = \delta_2(s_i, a_{i+1}), \quad t_{i+1} = \delta(t_i, a_{i+1}).$$

Also

$$t_{i+1} = \delta(t_i, a_{i+1}) = \delta\big((r_i, s_i), a_{i+1}\big) = \big(\delta_1(r_i, a_{i+1}), \delta_2(s_i, a_{i+1})\big) = (r_{i+1}, s_{i+1}).$$

□

Satz 1.31
Seien $L_1, L_2 \subseteq \Sigma^*$ DFA-erkennbar. Dann ist auch $L_1 \cap L_2$ DFA-erkennbar.

Lemma 1.34
Seien $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ DFAs und

$$\mathcal{A} = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$$

ihr Produktautomat mit $F := F_1 \times F_2$.
Dann gilt

$$L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2).$$

Beweis des Satzes aus dem Lemma.
Wähle DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$, so dass $L_1 = L(\mathcal{A}_1)$ und $L_2 = L(\mathcal{A}_2)$. Sei $\mathcal{A}$ der Produktautomat von $\mathcal{A}_1$ und $\mathcal{A}_2$ mit $F = F_1 \times F_2$. Nach dem Lemma gilt dann

$$L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = L_1 \cap L_2.$$

Also ist $L_1 \cap L_2$ DFA-erkennbar. □

Aus dem Lemma folgt auch, dass die Vermutung aus Beispiel 1.32 richtig war.

Beweis des Lemmas.
Sei $w = a_1 \dots a_n \in \Sigma^*$.

Wir zeigen:

$$\mathcal{A} \text{ akzeptiert } w \iff \mathcal{A}_1 \text{ akzeptiert } w \text{ und } \mathcal{A}_2 \text{ akzeptiert } w. \qquad (\star)$$

Daraus folgt $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Seien

$$(r_0, \dots, r_n),$$
$$(s_0, \dots, s_n)$$

die Läufe von $\mathcal{A}_1$ und $\mathcal{A}_2$ auf w.
Nach Lemma 1.33 ist dann

$$\big((r_0, s_0), \dots, (r_n, s_n)\big).$$

der Lauf von $\mathcal{A}$ auf w.
Also gilt:

$$\begin{aligned} \mathcal{A}_1 \text{ akzeptiert } w &\iff r_n \in F_1, \\ \mathcal{A}_2 \text{ akzeptiert } w &\iff s_n \in F_2, \\ \mathcal{A} \text{ akzeptiert } w &\iff (r_n, s_n) \in F = F_1 \times F_2, \end{aligned}$$

Daraus folgt sofort $(\star)$:

$$\begin{aligned} \mathcal{A} \text{ akzeptiert } w &\iff (r_n, s_n) \in F_1 \times F_2 \\ &\iff r_n \in F_1 \text{ und } s_n \in F_2 \\ &\iff \mathcal{A}_1 \text{ akzeptiert } w \text{ und } \mathcal{A}_2 \text{ akzeptiert } w. \end{aligned}$$

□

Vereinigung

Satz 1.35

Seien $L_1, L_2 \subseteq \Sigma^*$ DFA-erkennbar. Dann ist auch $L_1 \cup L_2$ DFA-erkennbar.

Beweis von Satz 1.35

Erster Beweis.

$$
\begin{aligned}
L_1, L_2 \text{ DFA-erkennbar} &\iff \overline{L}_1, \overline{L}_2 \text{ DFA-erkennbar} && (\textit{Satz } 1.29)\\
&\iff \overline{L}_1 \cap \overline{L}_2 \text{ DFA-erkennbar} && (\textit{Satz } 1.31)\\
&\iff \overline{(\overline{L}_1 \cap \overline{L}_2)} \text{ DFA-erkennbar} && (\textit{Satz } 1.29).
\end{aligned}
$$

Nach der De Morgan'schen Regel gilt

$$L_1 \cup L_2 = \overline{(\overline{L}_1 \cap \overline{L}_2)}.$$

□

Zweiter Beweis.

Seien $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ und $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ DFAs mit $L(\mathcal{A}_1) = L_1$ und $L(\mathcal{A}_2) = L_2$.
Sei

$$\mathcal{A} = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$$

ihr Produktautomat mit $F := (Q_1 \times F_2) \cup (F_1 \times Q_2)$.

Für alle $(r, s) \in Q_1 \times Q_2$ gilt dann

$$(r, s) \in F \iff r \in F_1 \quad \text{oder} \quad s \in F_2.$$

Daraus folgt wie im Beweis von Lemma 1.34:

$$L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2).$$

Also ist $L_1 \cup L_2 = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ DFA-erkennbar. □

Hintereinanderausführung

Seien L_1, L_2 DFA-erkennbar, etwa von DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$.

Hintereinanderausführung

Seien L_1, L_2 DFA-erkennbar, etwa von DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$.

Frage

Ist auch $L_1 L_2$ DFA-erkennbar?

Hintereinanderausführung

Seien L_1, L_2 DFA-erkennbar, etwa von DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$.

Frage

Ist auch $L_1 L_2$ DFA-erkennbar?

Der Produktautomat von $\mathcal{A}_1$ und $\mathcal{A}_2$ führt die beiden Automaten parallel aus. Jetzt fragen wir nach einem Automaten, der $\mathcal{A}_1$ und $\mathcal{A}_2$ hintereinander ausführt.

Hintereinanderausführung

Seien L_1, L_2 DFA-erkennbar, etwa von DFAs $\mathcal{A}_1$ und $\mathcal{A}_2$.

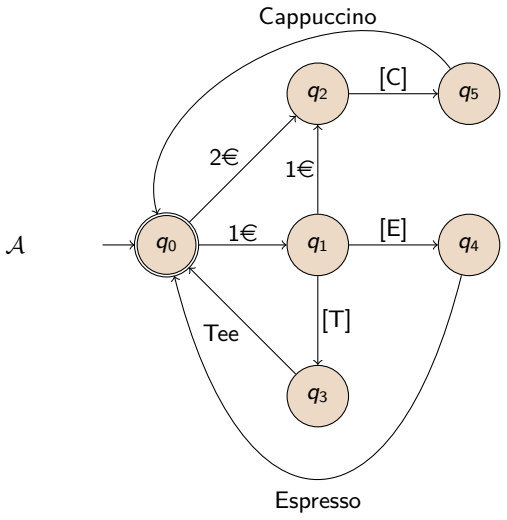
Frage

Ist auch $L_1 L_2$ DFA-erkennbar?

Der Produktautomat von $\mathcal{A}_1$ und $\mathcal{A}_2$ führt die beiden Automaten parallel aus. Jetzt fragen wir nach einem Automaten, der $\mathcal{A}_1$ und $\mathcal{A}_2$ hintereinander ausführt.

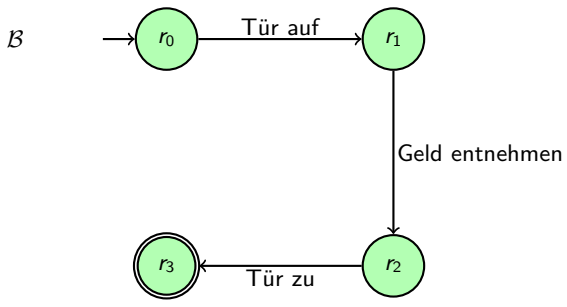
Wir sprechen auch von paralleler Komposition und sequentieller Komposition der beiden Automaten.

In Beispiel 1.3 haben wir folgendes Automatenmodell für einen Getränkeautomaten betrachtet:



Das ist kein DFA, weil nicht in jedem Zustand jede Aktion möglich ist. Diese Problem lässt sich aber leicht beheben, indem man einen "toten Zustand" einführt, zu dem alle nicht erlaubten Transitionen führen.

Mit dem selben Getränkeautomaten wollen wir jetzt einen zweiten Prozess assoziieren:



Jetzt wollen wir einen Automaten entwerfen, der die Hintereinanderausführung der beiden Prozesse modelliert, also einen Automaten für die Sprache

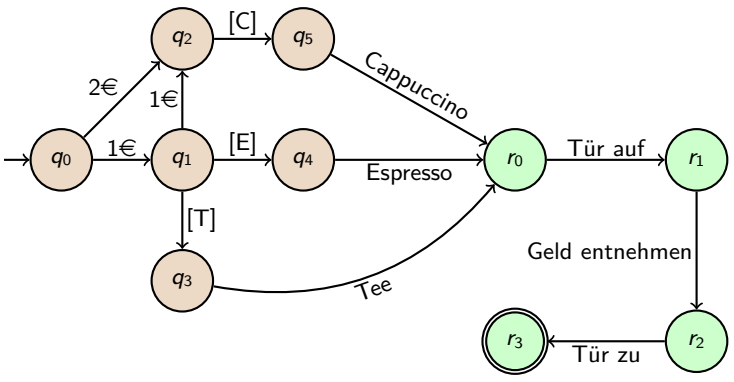
$$L(\mathcal{A})L(\mathcal{B}).$$

Noch lieber würden wir diese sequentielle Komposition iterieren, also einen Automaten für die Sprache

$$\big(L(\mathcal{A})L(\mathcal{B})\big)^*$$

entwerfen. Der Einfachheit halber betrachten wir aber nur $L(\mathcal{A})L(\mathcal{B})$.

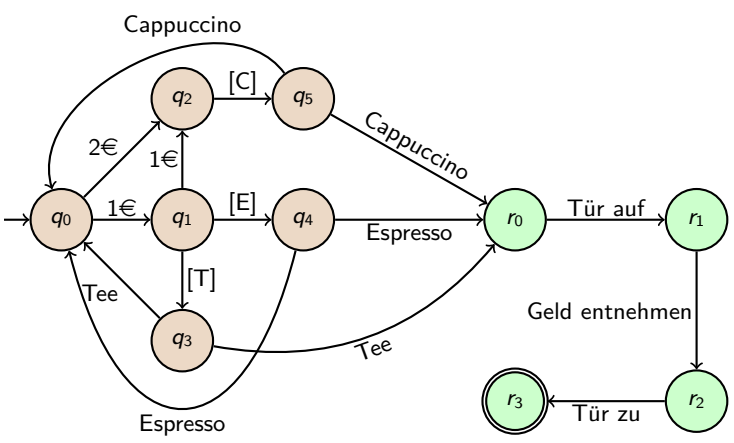
Erster Ansatz



Problem

Erkennt die falsche Sprache, denn man kann nur ein Getränk kaufen und muss anschließend sofort das Geld entnehmen.

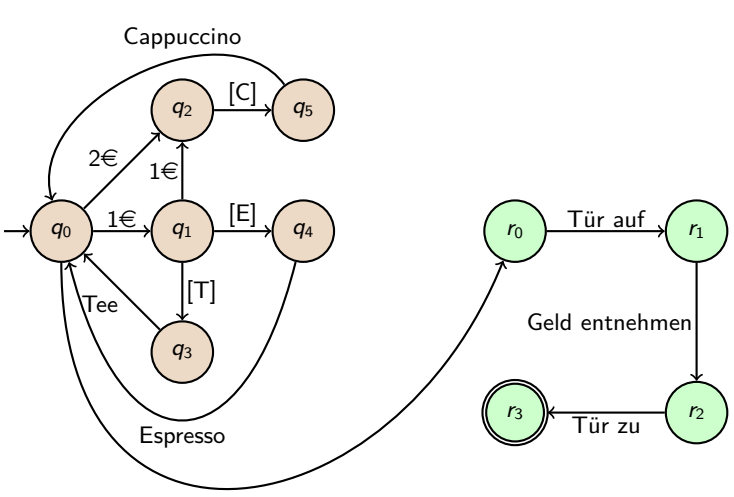
Zweiter Ansatz



Problem

Ist nicht deterministisch. Beispielsweise sind im Zustand q_3 zwei „Tee"-Transitionen möglich.
Außerdem erkennt der Automat immer noch die falsche Sprache, weil mindestens ein Getränk gekauft werden muss, bevor das Geld entnommen wird.

Dritter Ansatz



Problem

Die Transition von q_0 nach r_0 entspricht keiner Aktion.

Fazit

Mit den bestehenden Techniken lässt sich die sequentielle Komposition zweier Prozesse nicht adäquat modellieren.
Nützlich wäre eine Erweiterung des Modells, die Nichtdeterminismus und „leere Transitionen“ erlaubt.