Automatentheorie und ihre Anwendungen
Teil 1: endliche Automaten auf endlichen Wörtern

Thomas Schneider

15.–22. April 2015

Überblick

Und nun . . .

Wörter, Sprachen, . . .

- **Symbole** $a, b, \dots$
- **Alphabet** Σ: endliche nichtleere Menge von Symbolen
- **(endliches) Wort** w über Σ:
  endliche Folge $w = a_1 a_2 \dots a_n$ von Symbolen $a_i \in \Sigma$
- **leeres Wort** ε
- **Wortlänge** $|a_1 a_2 \dots a_n| = n, \quad |\varepsilon| = 0$
- **Menge aller Wörter** über Σ: Σ^*
- **Sprache** L über Σ: Teilmenge $L \subseteq \Sigma^*$ von Wörtern
- **Sprachklasse** $\mathcal{L}$: Menge von Sprachen

Endliche Automaten

Definition 1

Ein **nichtdeterministischer endlicher Automat (NEA)** über einem Alphabet Σ ist ein 5-Tupel $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, wobei

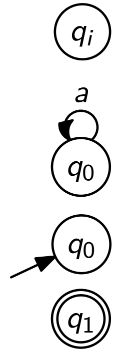
- Q eine endliche nichtleere **Zustandsmenge** ist,
- Σ ein Alphabet ist,
- $\Delta \subseteq Q \times \Sigma \times Q$ die **Überführungsrelation** ist, (*)
- $I \subseteq Q$ die Menge der **Anfangszustände** ist,
- $F \subseteq Q$ die Menge der **Endzustände** ist.

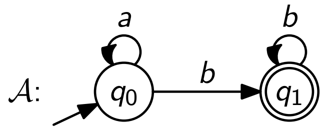
- (*) bedeutet:
  Δ besteht aus Tripeln (q, a, q') mit $q, q' \in Q$ und $a \in \Sigma$
- $(q, a, q') \in \Delta$ bedeutet intuitiv:
  ist $\mathcal{A}$ in Zustand q und liest ein a, geht er in Zustand q' über.

Beispiel und graphische Repräsentation von NEAs

Betrachte $\mathcal{A} =$
$(\{q_0, q_1\},\ \{a, b\},\ \{(q_0, a, q_0), (q_0, b, q_1), (q_1, b, q_1)\},\ \{q_0\}, \{q_1\})$

- Zustände: q_0, q_1
- Alphabet $\{a, b\}$
- Übergänge: von q_0 mittels a zu q_0, ...
- Anfangszustand q_0
- Endzustand q_1





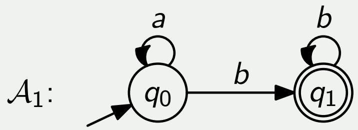
Berechnungen und Akzeptanz

Definition 2

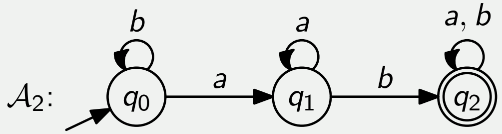
Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ ein NEA.

- Eine **Berechnung** (Run) von $\mathcal{A}$ auf $w = a_1 a_2 \dots a_n$ ist eine Folge
  $$q_0 q_1 q_2 \dots q_n,$$
  so dass für alle $i = 0, \dots, n-1$ gilt: $(q_i, a_{i+1}, q_{i+1}) \in \Delta$.
  Man sagt/schreibt: w **überführt** q_0 in q_n, $\quad q_0 \left|\frac{a_1 \dots a_n}{\quad}\right._{\mathcal{A}} q_n$
- $\mathcal{A}$ **akzeptiert** $w = a_1 a_2 \dots a_n$,
  wenn es eine Berechnung $q_0 q_1 q_2 \dots q_n$ von $\mathcal{A}$ auf w gibt mit $q_0 \in I$ und $q_n \in F$.
- Die von $\mathcal{A}$ **erkannte Sprache** ist
  $L(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ akzeptiert } w\}$.

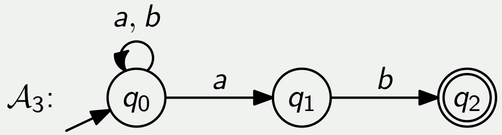
Beispiele



$L(\mathcal{A}_1) = \{a^n b^m \mid n \geqslant 0, m \geqslant 1\}$



$L(\mathcal{A}_2) = \{w \in \{a, b\}^* \mid w \text{ enthält Teilwort } ab\}$



$L(\mathcal{A}_3) = \{w \in \{a, b\}^* \mid w \text{ endet auf } ab\}$

Erkennbare Sprache

Definition 3

Eine Sprache $L \subseteq \Sigma^*$ ist **(NEA-)erkennbar**,
wenn es einen NEA $\mathcal{A}$ gibt mit $L = L(\mathcal{A})$.

Determinismus

Definition 4

Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ ein NEA.

Enthält Δ für jedes $q \in Q$ u. jedes $a \in \Sigma$ **genau 1** Tripel (q, a, q') und enthält I **genau 1** Zustand,

dann ist $\mathcal{A}$ ein **deterministischer endlicher Automat (DEA)**.

$\rightsquigarrow$ Nachfolgezustand für jedes Paar (q, a) eindeutig bestimmt

- Jeder DEA ist ein NEA,
  aber nicht umgekehrt (z. B. $\mathcal{A}_1, \mathcal{A}_3$ auf Folie 8).
- Auf Folie 8 ist nur $\mathcal{A}_2$ ein DEA;
  $\mathcal{A}_1$ kann mittels *Papierkorbzustand* zum DEA werden
  – und $\mathcal{A}_3$?

Potenzmengenkonstruktion

Frage: Sind DEAs und NEAs gleichmächtig?
D. h.: $\{L \mid L \text{ ist DEA-erkennbar}\} = \{L \mid L \text{ ist NEA-erkennbar}\}$?

Antwort: Ja!

Satz 5

Sei $\mathcal{A}$ ein NEA.

Dann gibt es einen DEA $\mathcal{A}^d$ mit $L(\mathcal{A}^d) = L(\mathcal{A})$.

Beweis: siehe Tafel.

Im schlimmsten Fall kann $\mathcal{A}^d$ im Vergleich zu $\mathcal{A}$ exponentiell viele Zustände haben (s. Hopcroft et al. 2001, S. 65).

Und nun . . .

Stichwortsuche

Typisches Problem aus dem Internetzeitalter

Gegeben sind **Stichwörter** $w_1, \dots, w_n \in \Sigma^*$
und **Dokumente** $D_1, \dots, D_M \in \Sigma^*$.

Finde alle j, so dass D_j mindestens ein (alle) w_i als Teilwort hat.

- relevant z.B. für **Suchmaschinen**
- übliche Technologie: **invertierter Index (II)**
  speichert für jedes im Internet auftretende w_i
  eine Liste aller Dokumente D_j, die w_i enthalten
- IIs sind zeitaufwändig zu erstellen
  und setzen voraus, dass die D_j sich nur langsam ändern

Grundbegriffe Textsuche Abschlusseig. Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme

Stichwortsuche ohne invertierte Indizes?

IIs versagen, wenn

- die (relevanten) Dokumente sich schnell ändern:
  - Suche in tagesaktuellen Nachrichtenartikeln
  - Einkaufshelfer sucht nach bestimmten Artikeln
    in aktuellen Seiten von Online-Shops
- die Dokumente nicht katalogisiert werden können:
  - Online-Shops wie Amazon generieren oft Seiten für ihre Artikel
    nur auf Anfragen hin.
  - $\rightsquigarrow$ Wie kann man dafür Stichwortsuche implementieren?

Grundbegriffe Textsuche Abschlusseig. Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme

Ein Fall für endliche Automaten!

Wdhlg.: Typisches Problem aus dem Internetzeitalter

Gegeben sind **Stichwörter** $w_1, \dots, w_n \in \Sigma^*$
und **Dokumente** $D_1, \dots, D_M \in \Sigma^*$.

Finde alle j, so dass D_j mindestens ein w_i als Teilwort hat.

Ziel: konstruiere NEA $\mathcal{A}$, der

- ein D_j zeichenweise liest und
- in einen Endzustand geht gdw. er eins der w_i findet

Beispiel 6

Seien $w_1 = \texttt{web}$ und $w_2 = \texttt{ebay}$.

Wie muss $\mathcal{A}$ aussehen?

Grundbegriffe Textsuche Abschlusseig. Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme

Implementation des NEAs $\mathcal{A}$

Eine Möglichkeit:

1. Determinisierung (Potenzmengenkonstruktion)
2. Simulation des resultierenden DEA $\mathcal{A}^d$

Wird $\mathcal{A}^d$ nicht zu groß?
($2^{27} > 134$ Mio. Zustände bei Stichw. „Binomialkoeffizient“, „Polynom“)

Nein,

- für unsere spezielle Form von $\mathcal{A}$,
- mit unserer Version der Potenzmengenkonstruktion

wird $\mathcal{A}^d$ genauso viele Zustände haben wie $\mathcal{A}$!

Beispiel: siehe Tafel

Zum Nachdenken

Übung:

- Konstruiere den DEA $\mathcal{A}^d$ für $\mathcal{A}$ von Folie 15.
- Beschreibe die Konstruktion von $\mathcal{A}^d$ allgemein, wenn $w_1, \ldots, w_n$ gegeben sind, mit $w_i = a_{i1} \ldots a_{i\ell_i}$ für jedes $i = 1, \ldots, n$.

Und nun . . .

Operationen auf Sprachen sind Operationen auf Mengen

Wie können (NEA-erkennbare) Sprachen kombiniert werden?

- Boolesche Operationen
  - **Vereinigung** $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ oder } w \in L_2\}$
  - **Schnitt** $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ und } w \in L_2\}$
  - **Komplement** $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$
- Wortoperationen
  - **Konkatenation** $L_1 \cdot L_2 = \{vw \mid v \in L_1 \text{ und } w \in L_2\}$
  - **Kleene-Hülle** $L^* = \bigcup_{i \geqslant 0} L^i$, wobei $L^0 = \{\varepsilon\}$ und $L^{i+1} = L^i \cdot L$ für alle $i \geqslant 0$

Frage

Unter welchen Op. sind die NEA-erkennbaren Sprachen abgeschlossen?

Abgeschlossenheit

Satz 7

Die Menge der NEA-erkennbaren Sprachen ist abgeschlossen unter den Operationen $\cup, \cap, ^-, \cdot, ^*$.

Das heißt:

Wenn L, L_1, L_2 NEA-erkennbar sind, dann sind auch

- $L_1 \cup L_2$
- $L_1 \cap L_2$
- $\overline{L}$
- $L_1 \cdot L_2$
- L^*

NEA-erkennbar.

Beweis: Direkte Konsequenz aus den folgenden Lemmata. □

Abgeschlossenheit unter Vereinigung

Lemma 8

Seien $\mathcal{A}_1, \mathcal{A}_2$ NEAs über Σ.
Dann gibt es einen NEA $\mathcal{A}_3$ mit $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Beweis: Seien $\mathcal{A}_i = (Q_i, \Sigma, \Delta_i, I_i, F_i)$ für $i = 1, 2$.
O. B. d. A. gelte $Q_1 \cap Q_2 = \emptyset$.
Konstruieren $\mathcal{A}_3 = (Q_3, \Sigma, \Delta_3, I_3, F_3)$ wie folgt.

- *Idee: vereinige $\mathcal{A}_1$ und $\mathcal{A}_2$.*
- $Q_3 = Q_1 \cup Q_2$
- $\Delta_3 = \Delta_1 \cup \Delta_2$
- $I_3 = I_1 \cup I_2$
- $F_3 = F_1 \cup F_2$ (Beispiel siehe Tafel)

Dann gilt $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$. (Tafel) □

Abgeschlossenheit unter Komplement

Lemma 9

Sei $\mathcal{A}$ ein NEA über Σ.
Dann gibt es einen NEA $\mathcal{A}^c$ mit $L(\mathcal{A}^c) = \overline{L(\mathcal{A})}$.

Beweis:

- *Idee:* Vertausche End- und Nicht-Endzustände (im DEA!)

O. B. d. A. sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ ein DEA (Satz 5).
Dann erkennt $\mathcal{A} = (Q, \Sigma, \Delta, I, Q \setminus F)$ die Sprache $\overline{L(\mathcal{A})}$. □

Folgerung 10

Seien $\mathcal{A}_1, \mathcal{A}_2$ NEAs über Σ.
Dann gibt es einen NEA $\mathcal{A}_3$ mit $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Gilt wegen $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Abgeschlossenheit unter Verkettung

Lemma 11

Seien $\mathcal{A}_1, \mathcal{A}_2$ NEAs über Σ.
Dann gibt es einen NEA $\mathcal{A}_3$ mit $L(\mathcal{A}_3) = L(\mathcal{A}_1) \cdot L(\mathcal{A}_2)$.

Beweis: Seien $\mathcal{A}_i = (Q_i, \Sigma, \Delta_i, I_i, F_i)$ für $i = 1, 2$.
O. B. d. A. gelte $Q_1 \cap Q_2 = \emptyset$ und $I_i = \{q_{0i}\}$.
Konstruieren $\mathcal{A}_3 = (Q_3, \Sigma, \Delta_3, I_3, F_3)$ wie folgt.

- *Idee:* „Hintereinanderhängen" von $\mathcal{A}_1$ und $\mathcal{A}_2$.

(Details siehe ThI1 – nicht wichtig für diese Vorlesung) □

Abgeschlossenheit unter Kleene-Hülle

Lemma 12

Sei $\mathcal{A}$ ein NEA über Σ.
Dann gibt es einen NEA $\mathcal{A}^k$ mit $L(\mathcal{A}^k) = L(\mathcal{A})^*$.

Beweis: Sei $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$. Konstruieren
$\mathcal{A}^k = (Q^k, \Sigma, \Delta^k, I^k, F^k)$ wie folgt.

- *Idee:* Lege „Schleife" um $\mathcal{A}$;
  Löse ε-Kanten auf wie im $\cdot$-Fall

(Details siehe ThI1 – nicht wichtig für diese Vorlesung) □

Und nun ...

Ziel dieses Abschnitts

Wie können NEA-erkennbare Sprachen bequem charakterisiert werden?

Reguläre Sprachen

Definition 13

Eine Sprache $L \subseteq \Sigma^*$ ist **regulär**, falls gilt:

- $L = \emptyset$ oder
- $L = \{\varepsilon\}$ oder
- $L = \{a\}$, $a \in \Sigma$, oder
- L lässt sich durch (endlichmaliges) Anwenden der Operatoren $\cup, \cdot, {}^*$ aus den vorangehenden Fällen konstruieren.

Beispiele:

$(\{a\} \cup \{b\})^* \cdot \{a\} \cdot \{b\}$ (siehe $\mathcal{A}_3$ auf Folie 8)

$\{b\}^* \cdot \{a\} \cdot \{a\}^* \cdot \{b\} \cdot (\{a\} \cup \{b\})^*$ (s. $\mathcal{A}_2$ auf Folie 8)

Reguläre Ausdrücke

Definition 14

Ein **regulärer Ausdruck (RA)** r über Σ und die *zugehörige Sprache* $L(r) \subseteq \Sigma^*$ werden induktiv wie folgt definiert.

- $r = \emptyset$ ist ein RA mit $L(r) = \emptyset$
- $r = \varepsilon$ ist ein RA mit $L(r) = \{\varepsilon\}$
- $r = a$, für $a \in \Sigma$, ist ein RA mit $L(r) = \{a\}$
- $r = (r_1 + r_2)$ ist ein RA mit $L(r) = L(r_1) \cup L(r_2)$
- $r = (r_1 r_2)$ ist ein RA mit $L(r) = L(r_1) \cdot L(r_2)$
- $r = (r_1)^*$ ist ein RA mit $L(r^*) = (L(r))^*$

Beispiele: (wir lassen Klammern weg soweit eindeutig)

$(a + b)^* ab$ (siehe $\mathcal{A}_3$ auf Folie 8)

$b^* aa^* b(a + b)^*$ (siehe $\mathcal{A}_2$ auf Folie 8)

Reguläre und NEA-erkennbare Sprachen

Satz 15 (Kleene)

Sei $L \subseteq \Sigma^*$ eine Sprache.

1. *L ist regulär gdw. es einen RA r gibt mit $L = L(r)$.*
2. *L ist regulär gdw. L NEA-erkennbar ist.*

Beweis.

1. Folgt offensichtlich aus Def. 13, 14.
2. Benutze Teil (1).

„$\Rightarrow$“: Induktion über Aufbau von r.
IA: gib Automaten an, die $\emptyset, \{\varepsilon\}, \{a\}$ erkennen.
IS: benutze Abschlusseigenschaften – Lemmas 8, 11, 12

„$\Leftarrow$“: siehe Theoretische Informatik 1.

□

Anwendungen regulärer Ausdrücke

- RAs werden verwendet, um „Muster“ von zu suchendem Text zu beschreiben

  z. B.: suche alle Vorkommen von „PLZ Ort“:
  $(0 + \cdots + 9)^5 \llcorner\!\lrcorner (A + \cdots + Z)(a + \cdots + z)^*$
- Programme zum Suchen von Mustern im Text übersetzen RAs in NEAs/DEAs und simulieren diese
- wichtige Klassen von Anwendungen:
  lexikalische Analyse, **Textsuche**

Komfortablere Syntax regulärer Ausdrücke

- UNIX und andere Anwendungen erweitern Syntax von RAs
- Hier: nur „syntaktischer Zucker“ – die Erweiterungen, die nicht aus den regulären Sprachen herausführen

- Alphabet Σ: alle ASCII-Zeichen
- RA `.` mit $L(.) = \Sigma$
- RA $[a_1 a_2 \ldots a_k]$, Abkürzung für $a_1 + a_2 + \cdots + a_k$
- RAs für Bereiche: z. B. `[a-z0-9]`, Abkü. für `[ab...z01...9]`
- Operator | anstelle +
- Operator ?: $r?$ steht für $\varepsilon + r$
- Operator +: $r+$ steht für rr^*
- Operator $\{n\}$: $r\{5\}$ steht für $rrrrr$
- Klammern und $*$ wie gehabt

PLZ-Ort-Beispiel:
`[0-9]{5}␣[A-Z][a-z]*`

Anwendung: lexikalische Analyse

- **Lexer** durchsucht Quelltext eines Programms nach **Token**:
  zusammengehörende Zeichenfolgen, z. B. Kennwörter, Bezeichner
- Ausgabe des Lexers: Token-Liste,
  wird an Parser weitergegeben
- Mit RAs: Lexer leicht programmier- und modifizierbar
- UNIX-Kommandos `lex` und `flex` generieren Lexer
  - Eingabe: Liste von Einträgen RA + Code
  - Code beschreibt Ausgabe des Lexers für das jeweilige Token
  - generierter Lexer wandelt RAs in DEAs um,
    um Vorkommen der Tokens zu finden (siehe Folie 15)
  - anhand des Zustands des DEAs lässt sich bestimmen,
    welches Token gefunden wurde

Beispieleingabe für lex

```
else
  {return(ELSE)}

[A-Za-z][A-za-z0-9]*
  {<Trage gefundenen Bezeichner in Symboltabelle ein>;
  return(ID);}

>=
  {return(GE);}

=
  {return(EQ);}
```

(Lexer-Generator muss Prioritäten beachten:
else wird auch vom 2. RA erkannt, ist aber reserviert)

Anwendung: Finden von Mustern im Text

Beispiel: Suchen von Adressen (Str. + Hausnr.) in Webseiten

Solche Angaben sollen gefunden werden:

```
Parkstraße 5
Enrique-Schmidt-Straße 12a
Breitenweg 24A
Knochenhauergasse 30-32
```

aber auch solche:

```
Straße des 17. Juni 17
...boulevard, ...allee, ...platz, ...
Postfach 330 440
Am Wall 8
```

$\rightsquigarrow$ Ausmaß der Variationen erst während der Suche deutlich

$\rightsquigarrow$ Gesucht: einfach modifizierbare Beschreibung der Muster

Mustersuche mit regulären Ausdrücken

Mögliches Vorgehen:

1. Beschreibung des Musters mit einem einfachen RA
2. Umwandlung des RA in einen NEA
3. Implementation des NEA wie auf Folie 15
4. Test
5. Wenn nötig, RA erweitern und Sprung zu Schritt 2

Adresssuche mit regulären Ausdrücken

So kann sich der RA entwickeln:

- Vorkommen von „straße“ etc.:[1]
  `straße|str\.|weg|gasse`
- Plus Name der Straße und Hausnummer:
  `[A-Z][a-z]*(straße|str\.|weg|gasse) [0-9]*`
- Hausnummern mit Buchstaben (12a), -bereiche (30–32):
  `[A-Z][a-z]*(straße|str\.|weg|gasse)`
  `([0-9]*[A-Za-z]?-)?[0-9]*[A-Za-z]?`
- und mehr:
  - Straßennamen mit Bindestrichen
  - „Straße“ etc. am Anfang
  - Plätze, Boulevards, Alleen etc.
  - Postfächer
  - ...

[1] Weil der UNIX-RA `.` für Σ reserviert ist, steht `\.` für $\{.\}$

Und nun ...

Pumping-Lemma

Wie zeigt man, dass L **nicht** NEA-erkennbar (regulär) ist?

Satz 16 (Pumping-Lemma)

Sei L eine NEA-erkennbare Sprache.
Dann gibt es eine Konstante $p \in \mathbb{N}$,[2]
so dass für alle Wörter $w \in L$ mit $|w| \geqslant p$ gilt:
Es gibt eine Zerlegung $w = xyz$ mit $|y| > 0$ und $|xy| \leqslant p$,
so dass $xy^iz \in L$ für alle $i \in \mathbb{N}$.[2]

Beweis: siehe Tafel. □

[2] $\mathbb{N} = \{0, 1, 2, \dots\}$

Anwendung des Pumping-Lemmas

Zur Erinnerung:

Satz 16 (Pumping-Lemma)

Wenn *L eine NEA-erkennbare Sprache ist,*
dann gibt es *eine Konstante $p \in \mathbb{N}$,*
so dass ***für alle*** *Wörter $w \in L$ mit $|w| \geqslant p$ gilt:*
Es gibt *eine Zerlegung $w = xyz$ mit $|y| > 0$ und $|xy| \leqslant p$,*
so dass $xy^iz \in L$ ***für alle*** *$i \in \mathbb{N}$.*

Benutzen Kontraposition:

Wenn es **für alle** Konstanten $p \in \mathbb{N}$
ein Wort $w \in L$ mit $|w| \geqslant p$ **gibt**, so dass es
für alle Zerlegungen $w = xyz$ mit $|y| > 0$ und $|xy| \leqslant p$
ein $i \in \mathbb{N}$ **gibt** mit $xy^iz \notin L$,
dann ist L **keine** NEA-erkennbare Sprache.◀ Bsp.: s. Tafel

Bemerkungen zum Pumping-Lemma

- Bedingung in Satz 16 ist **notwendig** dafür, dass L NEA-erkennbar ist.
- nicht **hinreichend** (Bsp.: $\{a^nb^kc^k \mid n, k \geqslant 1\} \cup \{b^nc^k \mid n, k \geqslant 0\}$)

$\rightsquigarrow$ Pumping-L. nur zum **Widerlegen** von Erkennbarkeit verwendbar,
nicht zum Beweisen, dass L regulär ist

- Notwendige und hinreichende Bedingung: *Jaffes Pumping-L.*

Der Satz von Myhill-Nerode

Ziel: notwendige **und** hinreichende Bedingung für Erkennbarkeit

Definition 17

Sei $L \subseteq \Sigma^*$ eine Sprache.

Zwei Wörter $u, v \in \Sigma^*$ sind **L-äquivalent** (Schreibweise: $u \sim_L v$), wenn für alle $w \in \Sigma^*$ gilt:

$$uw \in L \quad \text{genau dann, wenn} \quad vw \in L$$

$\sim_L$ heißt **Nerode-Rechtskongruenz** und ist Äquivalenzrelation:

- reflexiv: offensichtlich
- symmetrisch: offensichtlich
- transitiv: offensichtlich

Index von $\sim_L$: Anzahl der Äquivalenzklassen

Der Satz von Myhill-Nerode

Satz 18 (Myhill-Nerode)

$L \subseteq \Sigma^*$ is NEA-erkennbar gdw. $\sim_L$ endlichen Index hat.

Ohne Beweis. Beispiel siehe Tafel.

Interessantes "**Nebenprodukt**" des Beweises:

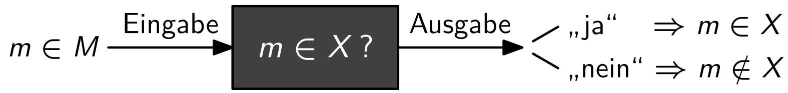
Endlicher Index n von $\sim_L$
= minimale Anzahl von Zuständen in einem DEA, der L erkennt.

Und nun ...

Entscheidbarkeit

(Entscheidungs-)Problem

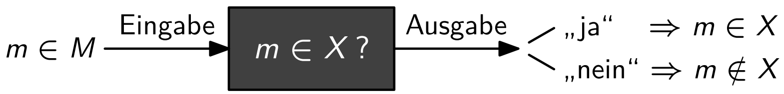
- ... ist eine Teilmenge $X \subseteq M$
- Beispiele:
  - X = Menge aller Primzahlen, $M = \mathbb{N}$
  - X = Menge aller NEAs $\mathcal{A}$ mit $L(\mathcal{A}) \neq \emptyset$,
    M = Menge aller NEAs
- man stelle sich eine Blackbox vor:



Entscheidbarkeit: X ist entscheidbar,
wenn es einen Algorithmus A gibt, der die Blackbox implementiert.

(Programmiersprache u. Rechnermodell sind relativ unerheblich:
erweiterte Churchsche These)

Komplexität



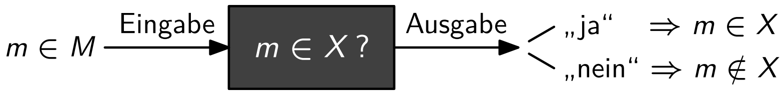
Komplexität:
zusätzliche Anforderungen an Zeit-/Speicherplatzbedarf von A

- **Polynomialzeit:** Anzahl Rechenschritte von A ist $\leqslant pol(|m|)$, $|m|$: Länge der Eingabe; pol ist ein festes **Polynom**
- **Polynomieller Platz:** von A benötigter Speicherplatz $\leqslant pol(|m|)$
- **Exponentialzeit:** Anzahl Rechenschritte von A ist $\leqslant 2^{pol(|m|)}$
- ...

Einige übliche Komplexitätsklassen

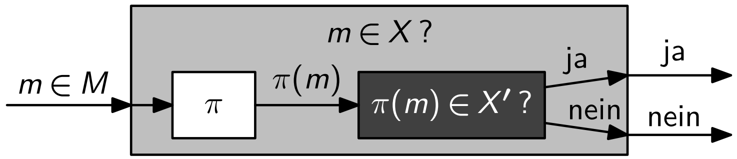
Name	Bedeutung	Beispiel-Problem
L	logarithm. Speicherplatz	Grapherreichbarkeit
NL	nichtdetermin. log. Platz	"
P	Polynomialzeit	Primzahlen
NP	nichtdeterminist. Polyzeit	Erfüllbarkeit Auss.-logik
PSpace	polynom. Speicherplatz	Erfüllbarkeit QBF
ExpTime	Exponentialzeit	
NExpTime	nichtdet. Exponentialzeit	
ExpSpace	exponentieller Platz	
⋮	⋮	
	unentscheidbar	Erfüllbarkeit Präd.-logik

Reduktion



(Polynomielle) Reduktion von $X \subseteq M$ nach $X' \subseteq M'$
ist eine (in Polyzeit berechenbare) Funktion $\pi : M \to M'$ mit

$$m \in X \qquad \text{gdw.} \qquad \pi(m) \in X'$$



Wenn X zu X' reduzierbar, dann ist X **höchstens so schwer** wie X'.

Wenn alle Probleme aus Komplexitätsklasse $\mathcal{C}$ auf X reduzierbar, dann ist X **schwer für** $\mathcal{C}$.

Bestimmung der Komplexität

Normalerweise zeigt man, dass ein Problem $X \subseteq M$ ...

- **in** einer Komplexitätsklasse $\mathcal{C}$ liegt, indem man
  - einen Algorithmus A findet, der X löst
  - zeigt, dass A korrekt ist (*ja*/*nein*-Antworten) und terminiert
  - zeigt, dass A für jedes $m \in M$ **höchstens** die $\mathcal{C}$-Ressourcen braucht

    ... A kann z. B. eine Reduktion zu einem Problem aus $\mathcal{C}$ sein
- **schwer (hard) für** $\mathcal{C}$ ist, indem man
  - ein Problem $X' \subseteq M'$ findet, dass **schwer für** $\mathcal{C}$ ist
  - und eine Reduktion von X' nach X angibt
- **vollständig für** $\mathcal{C}$ ist, indem man zeigt, dass es
  - **in** $\mathcal{C}$ liegt und
  - **schwer für** $\mathcal{C}$ ist

Entscheidungsprobleme für endliche Automaten

- Betrachten wesentliche Eigenschaften von Sprachen (Sprachen repräsentiert durch NEAs oder reguläre Ausdr.)
  - Ist eine gegebene Sprache leer?
  - Ist ein gegebenes Wort w in einer Sprache L?
  - Beschreiben zwei Repräsentationen einer Sprache tatsächlich dieselbe Sprache?
- Wichtig für die bisher gesehenen Anwendungen (Ü: finde Gründe!)
- Art der Repräsentation spielt manchmal eine Rolle:
  - Umwandlung DEA $\longrightarrow$ NEA: konstante Zeit ☻
  - NEA $\longrightarrow$ DEA: Exponentialzeit ☹
  - reg. Ausdr. $\longrightarrow$ NEA: Polynomialzeit ☻
  - NEA $\longrightarrow$ reg. Ausdr.: Exponentialzeit ☹

Wir beschränken uns im Folgenden auf NEAs.

Das Leerheitsproblem

Ist definiert als $\{\mathcal{A} \mid L(\mathcal{A}) = \emptyset\}$.

Satz 19

Das Leerheitsproblem ist entscheidbar.

Beweis: Siehe Tafel. ●

Komplexität: **NL**-vollständig (Wegsuche in gerichteten Graphen)

Das Wortproblem

Ist definiert als $\{(\mathcal{A}, w) \mid w \in L(\mathcal{A})\}$.

Satz 20

Das Wortproblem ist entscheidbar.

Beweis: Reduktion zum Leerheitsproblem – siehe Tafel. ●

Komplexität: **NL**-vollständig (modifizierte Wegsuche)
(Reduktion zum Leerheitsproblem liefert nur „in **P**“)

Das Äquivalenzproblem

Ist definiert als $\{(\mathcal{A}_1, \mathcal{A}_2) \mid L(\mathcal{A}_1) = L(\mathcal{A}_2)\}$.

Satz 21

Das Äquivalenzproblem ist entscheidbar.

Beweis: Siehe Tafel. ●

Komplexität: für DEAs in **P**, für NEAs **PSPACE**-vollständig
(aus Beweis folgt nur „in **EXPTIME**“)

Das Universalitätsproblem

Ist definiert als $\{\mathcal{A} \mid L(\mathcal{A}) = \Sigma^*\}$.

Satz 22

Das Universalitätsproblem ist entscheidbar.

Beweis: Übungsaufgabe.

Komplexität: für DEAs in P, für NEAs PSPACE-vollständig
(aus Beweis folgt nur „in EXPTIME“)

Grundbegriffe Textsuche Abschlusseig. Reguläre Ausdrücke Charakterisierungen Entscheidungsprobleme

Literatur für diesen Teil

John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman.
Introduction to Automata Theory, Languages and Computation.
2. Auflage, Addison-Wesley, 2001. Kapitel 1,2.
Verfügbar in SUB:
Zentrale `a inf 420 e/028(2)`
dt. Versionen 2002–11:
2× TB Technik `n 438/101(3,2)`
dt. Version 2006: 2× Zentrale `a inf 420 ef/238a,b`
Vorgängerversion von 1979: Zentrale `a inf 420 e/806`
oder BB Math.-MZH `19h kyb 315 e/996a`
oder auf dt., 1988: 2× Zentr. `a kyb 308 f/352, (a)`

Meghyn Bienvenu.
Automata on Infinite Words and Trees.
Vorlesungsskript, Uni Bremen, WS 2009/10.
http://www.informatik.uni-bremen.de/tdki/lehre/ws09/automata/automata-notes.pdf