Theoretische Informatik: Automaten und formale Sprachen

Prof. Dr. Sibylle Schwarz
HTWK Leipzig, Fakultät IM
Gustav-Freytag-Str. 42a, 04277 Leipzig
Zimmer Z 411 (Zuse-Bau)
https://informatik.htwk-leipzig.de/schwarz
sibylle.schwarz@htwk-leipzig.de

Sommersemester 2025

Einordnung der Theoretischen Informatik

Informatik Lehre von der Darstellung und Verarbeitung von Information durch Algorithmen

Teilgebiete der Informatik:

theoretisch
- Sprachen zur Formulierung von Information und Algorithmen,
- Möglichkeiten und Grenzen der maschinellen Berechenbarkeit,
- Grundlagen für technische und praktische (und angewandte) Informatik

technisch
- maschinelle Darstellung von Information
- Mittel zur Ausführung von Algorithmen

(Rechnerarchitektur, Hardware-Entwurf, Netzwerk, ...)

praktisch Entwurf und Implementierung von Algorithmen (Betriebssysteme, Compilerbau, SE, ...)

angewandt Anwendung von Algorithmen (Text- und Bildverarbeitung, Datenbanken, KI, Medizin-, Bio-, Wirtschafts-, Medieninformatik, ...)

Teilgebiete der theoretischen Informatik

Formale Sprachen

- Repräsentation von Informationen und Aufgaben in maschinenlesbarer Form (Mensch-Maschine-Kommunikation)
- Ausdrucksstärke und Flexibilität von Programmiersprachen
- Übersetzung von Programmiersprachen (z.B. in ausführbaren Code)

Maschinenmodelle (Automaten)

- Möglichkeiten und Grenzen verschiedener Modelle zur (maschinellen) Ausführung von Algorithmen

Berechenbarkeitstheorie (Master)

- Welche Aufgaben sind überhaupt algorithmisch (mit Hilfe verschiedener Maschinenmodelle) lösbar? Auch negative Antworten sind sehr hilfreich (sparen Aufwand für ungeeignete Lösungsansätze)

Komplexitätstheorie (Master)

- Welche Aufgaben sind mit beschränkten Ressourcen (z.B. Zeit, Speicherplatz) lösbar?
- Für welche Aufgaben können schnelle Algorithmen existieren?

Prinzipien der theoretischen Informatik

ältester Zweig der Informatik (lange vor dem ersten Computer)

Mathematische Prinzipien:

- Abstraktion
  - ermöglicht verallgemeinerte Aussagen und breit einsetzbare Verfahren,
  - Ergebnisse und Verfahren oft nicht sofort praktisch anwendbar, müssen auf spezielle Situationen angepasst werden.
- Beweisbarkeit
  - erfordert präzise Modellierung der Aufgabe
  - Nachweis der Korrektheit von Hard- und Software (Tests können dies nicht !)

Wissen aus der theoretischen Informatik

- veraltet über viele Jahre kaum
- Grundlage für Verständnis von (schnelllebigem) Spezialwissen, z.B. konkrete Programmiersprachen, Domain-spezifische Sprachen, Transformationen verschiedener Darstellungen

Aus der Modulbeschreibung

C993 Theoretische Informatik: Automaten und formale Sprachen

Arbeitsaufwand: Präsenzzeit 56 h (= 2 SWS V + 2 SWS S)
Vor- und Nachbereitungszeit 94 h (≈ 6 h je Woche)

Voraussetzungen: anwendungsbereite Kenntnisse auf den Gebieten Modellierung, Logik, Algorithmen und Datenstrukturen, Aufwandsabschätzungen

Lernziele: Die Studierenden sind in der Lage, wichtige Klassen formaler Sprachen als Grundlage von Programmier- und Beschreibungssprachen einzuordnen und kennen die wesentlichen Eigenschaften der Sprachklassen.
Sie kennen die entsprechenden abstrakten Maschinenmodelle und Algorithmen und können sie zur Darstellung und Lösung praktischer Aufgabenstellungen einsetzen.
Die Studierenden wissen, dass nicht jedes formal darstellbare Problem algorithmisch lösbar ist.

Inhalt der Lehrveranstaltung

- Formale Sprachen
  - Wiederholung: Alphabet, Wort, Sprache, Operationen darauf
  - reguläre Ausdrücke
  - Wortersetzung
  - Grammatiken
  - Chomsky-Hierarchie
- Maschinenmodelle
  - Endliche Automaten
  - Kellerautomaten
  - Turing-Maschinen
- Berechenbarkeit (Ausblick auf Master-Modul)
  - berechenbare Funktionen
  - Berechnungsmodelle
  - These von Church
  - algorithmische Entscheidbarkeit / Unentscheidbarkeit
- Komplexität (Ausblick auf Master-Modul)

jeweils mit vielen Beispielen

Lehrveranstaltungen

Folien, Übungsserien, aktuelle Informationen unter
`https://informatik.htwk-leipzig.de/schwarz/lehre/ss25/tib`

Vorlesung	donnerstags / freitags	(2 h / Woche)
Selbststudium	(Hausaufgaben):	(6 h / Woche)
schriftliche	Übungsserien (ca. zu jeder Vorlesung) Besprechung in der folgenden Übung	
Autotool	jeweils nach der Vorlesung bis Sonntag der nächsten Woche	
Seminar	Besprechung der Übungsserien (incl. Selbsstest) (Vorrechnen der Musterlösungen)	

Prüfung

Prüfungsvorleistungen:

- $\geq$ 3 Kurzvorträge zu schriftlichen Übungsaufgaben (Übungen) und
- $\geq$ 50% aller Punkte für Autotool-Pflichtaufgaben

Prüfung: Klausur 90 min
Aufgabentypen ähnlich Übungsaufgaben
(Hilfsmittel: beidseitig handbeschriebenes A4-Blatt)

Formale Sprachen

Syntax natürlicher Sprachen:

- Rechtschreibung: korrekte Wörter
- Grammatik: Aufbau korrekter Sätze

Definition von Programmiersprachen:

Syntax	Form der Sprachelemente
Semantik	Bedeutung der Sprachelemente und -strukturen
Pragmatik	Regeln zur zweckmäßigen Anwendung

Syntax von Programmiersprachen:

- Schlüsselwörter, Bezeichner, Darstellung von Zahlen, ...
- Programmstrukturen:
  Form der Ausdrücke, Anweisungen, Deklarationen, ...

Formale Sprachen: Beispiele

Programmiersprachen (Java):

```
while (b != 0) { if (a > b) a = a - b; else b = b - a; }
```

Regeln für korrekte Syntax (EBNF):

```
Statement ::=  ... | IfStmt | WhileStmt | ... ;
WhileStmt ::= "while" "(" Expr ")" Statement;
IfStmt    ::= "if" "(" Expr ")" Statement ( "else" Statement )?;
Expr      ::= ...
```

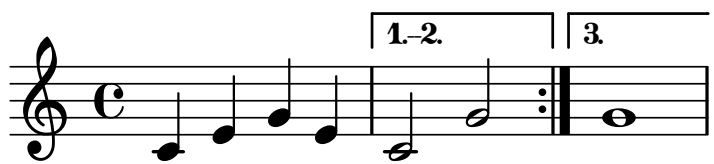
Domain-spezifische Sprachen , z.B. Autotool-Lösungen zu AL-Modell

```
listToFM[(x,True),(y,False),(z,False)]
```

Regeln für korrekte Syntax (EBNF):

```
belegung      ::= "listToFM" "[" var-wert-ps "]"
var-wert-ps   ::= "" | var-wert-paar | var-wert-paar "," var-wert-ps
var-wert-paar ::= "(" var-name, wert ")"
wert          ::= "True" | "False"
var-name      ::= ...
```

Graphische Sprachen , z.B.



Maschinenmodell: endlicher Automat

Beschreibung des dynamischen Verhaltens von Systemen

Modellierung von Abläufen (Zustandsübergangssysteme)

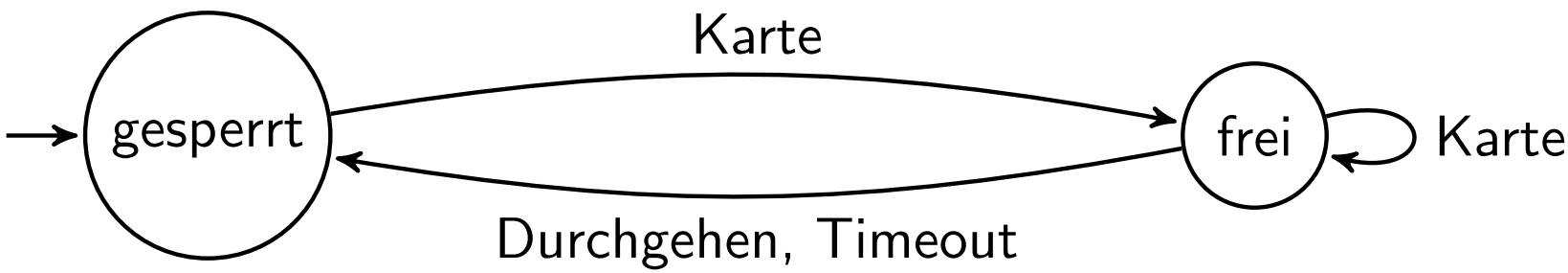
Beispiele:

- Bedienoperationen an Geräten oder Software
- Schaltfolgen von Ampelanlagen
- Steuerung von Produktionsanlagen
- Ablauf von (Geschäfts-)Prozessen

Beispiel: (Pool-)Einlass mit Karte

Automat definiert durch

- Zustände: gesperrt, frei
- Startzustand: gesperrt
- Aktionen (Eingabesymbole):
  Karte (anlegen), Durchgehen, Timeout
- Zustandsübergänge: (gesperrt, Karte) $\rightarrow$ frei
  (frei, Karte) $\rightarrow$ frei
  (frei, Durchgehen) $\rightarrow$ gesperrt
  (frei, Timeout) $\rightarrow$ gesperrt

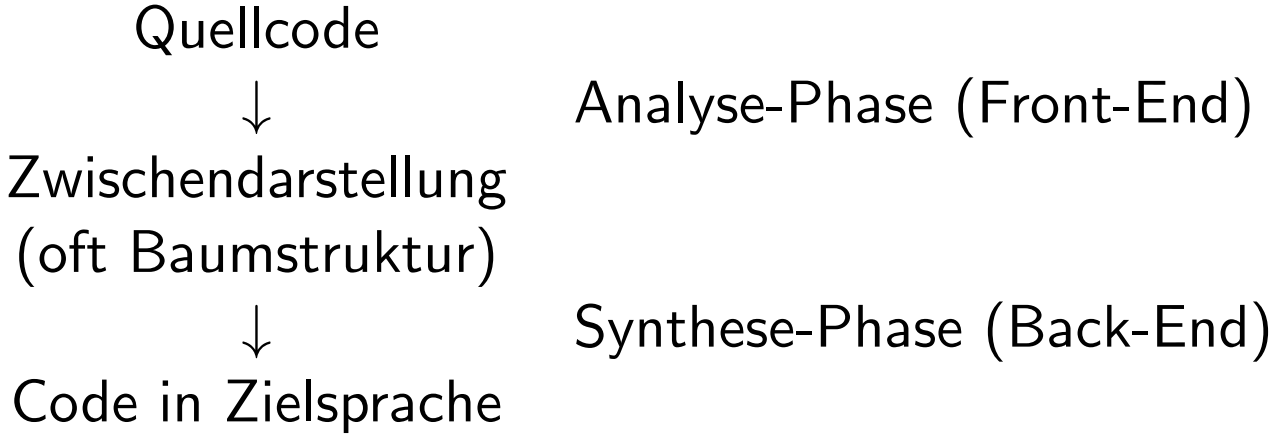


definiert mögliche (erlaubte) Folgen von Aktionen
Diese Folgen lassen sich durch reguläre Ausdrücke darstellen:
(Karte Karte*(Durchgehen + Timeout))*

Anwendung bei der Übersetzung von Programmen

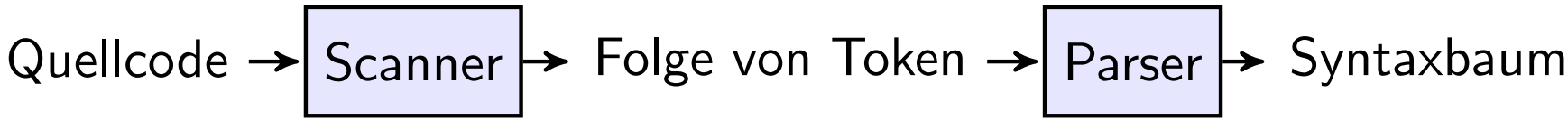
Übersetzung von Quell- in Zielsprache
(z.B. C, Java in Maschinen- oder Byte-Code)

meist in zwei Phasen über eine (gemeinsame) Abstraktion:



(im Master-Modul Compilerbau)

Analyse-Phase



lexikalische Analyse (Scanner)
: lineare Analyse des Quelltextes,
Aufteilung in Einheiten (Token)
z.B. Schlüsselwörter, Bezeichner, Zahlen
reguläre Sprachen, endliche Automaten

Syntaxanalyse (Parser)
: hierarchische Struktur des Quelltextes
z.B. Ausdrücke, Verzweigungen, Schleifen
kontextfreie Sprachen, Kellerautomaten

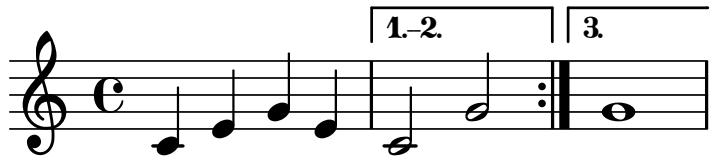
semantische Analyse Annotationen im Syntaxbaum,
: z.B. Typprüfungen

Einsatz ähnlicher Transformations- und Analyse-Methoden

- Compiler für Programmiersprachen (z. B. Java → Bytecode)
- Interpreter für Programmiersprachen (z. B. Java-Bytecode)
- Übersetzung von Daten zwischen verschiedenen Formaten
  z. B. LilyPond (`http://www.lilypond.org`) übersetzt

```
\repeat volta 3 { c' e' g' e' | }
\alternative { { c'2  g' | } { g'1 | } }
```

  u. A. in



- Verarbeitung von Domain-spezifischen Sprachen
- Textformatierung
- Dokumentbeschreibungssprachen
- kontextabhängige Hilfe in Entwicklungsumgebungen
- statische Analyse zur Fehlersuche in Programmen
- graphische Editoren (z.B. für UML-Diagramme) mit automatischer Programmgenerierung

Berechenbarkeit / Entscheidbarkeit

Halteproblem:
Kann ein (Test-)programm U existieren, welches für jedes beliebige (Dienst-)Programm P (Eingabe als Quelltext) entscheidet, ob P nach endlich vielen Schritten anhält?

Antwort (Herleitung später): Nein

Folgerungen:

- Alle Versuche, ein solches Programm zu erzeugen, müssen fehlschlagen.
- Sinnvoll ist jedoch die Suche nach Verfahren, die für eine möglichst große Teilmenge aller (Dienst-)Programme P entscheiden, ob P nach endlich vielen Schritten anhält.
- Entwickler von P (Dienstleister) muss nachweisen, dass sein Programm P nach endlich vielen Schritten anhält.

(im INM-Modul Theoretische Informatik)

Komplexität

Beispiel Primzahltest

Aufgabe: Ist eine gegebene Zahl n eine Primzahl?

Instanz der Aufgabe: Ist 12347 eine Primzahl?

lösbar durch den Algorithmus:

1. Für alle $i \in \{2, \ldots, n\}$:
   Test: Ist n durch i teilbar?
   - ja: Ende mit Ausgabe n ist nicht prim.
   - nein: weiter (mit Test für $i + 1$)
2. Ausgabe: n ist prim.

Dieser Test ist für große Zahlen aufwendig. Geht es besser?

- Was bedeutet aufwendig und besser?
- Wie aufwendig ist eine Berechnung?
- Wie aufwendig ist die Lösung einer Aufgabe?

(im INM-Modul Theoretische Informatik)

Literatur

- Uwe Schöning:
  Theoretische Informatik - kurzgefasst (Spektrum 2001)
- John E. Hopcroft, Jeffrey D. Ullman:
  Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie (Addison-Wesley 1990)

Online Verfügbar (über Bilbliothek):

- Dirk W. Hoffmann: Theoretische Informatik (Hanser 2018)
- Ulrich Hedtstück: Einführung in die Theoretische Informatik Formale Sprachen und Automatentheorie (Oldenbourg 2012)
- Lukas König, Friederike Pfeiffer-Bohnen, Hartmut Schmeck Theoretische Informatik – ganz praktisch (De Gruyter 2016)
- Heinz-Peter Gumm, Manfred Sommer
  Informatik – Band 3: Formale Sprachen, Compilerbau, Berechenbarkeit und Komplexität (De Gruyter 2019)
- Gottfried Vossen, Kurt-Ulrich Witt:
  Grundkurs Theoretische Informatik (Vieweg 2016)
- Renate Winter: Theoretische Informatik (Oldenbourg 2002)

WH: Alphabet, Wort, Sprache

Für jede Menge A heißt

$$A^n = \underbrace{A \times \cdots \times A}_{n} = \{w_1 \cdots w_n \mid \forall i : w_i \in A\}$$

Menge aller Wörter der Länge n über A
(n-Tupel, Vektoren, Folgen, Listen, Zeichenketten)

$A^* = \bigcup_{\{n \in \mathbb{N}\}} A^n$ Menge aller Wörter über A
(endliche Folgen, Listen, Zeichenketten)

$A^0 = \{\varepsilon\}$ mit leerem Wort ε

Alphabet (endliche) Menge A von Symbolen

Wort endliche Folge von Symbolen $w = w_1 \cdots w_n$ mit
$\forall i \in \{1, \ldots, n\} : w_i \in A$

Länge eines Wortes $|w|$ = Anzahl der Symbole in w

Anzahl der Vorkommen eines Symboles in einem Wort
$|w|_a$ = Anzahl der a in w (für $a \in A$)

Sprache Menge von Wörtern $L \subseteq A^*$

Beispiele

- Alphabet $A = \{0, 1\}$
  Wörter $\in A^* = \{0, 1\}^*$: Menge aller Binärwörter
  Sprachen $\subseteq A^*$, z.B.
  - $\{w \in \{0, 1\}^* \mid w_1 \neq 0\}$ Menge aller Binärzahlen ohne führende Nullen
  - $\{w \in \{0, 1\}^* \mid w_1 \neq 0 \wedge w_{|w|-1} = w_{|w|} = 0\}$ Menge aller Binärdarstellungen durch 4 teilbarer Zahlen ohne führende Nullen
- Alphabet $A = \{a, b\}$
  Wörter $\in A^* = \{a, b\}^*$: Menge aller Wörter, die höchstens die Buchstaben a und b enthalten
  Sprachen $\subseteq A^*$, z.B.
  - $\emptyset$
  - $\{a, b\}$
  - $\{a\}^* = \{\varepsilon, a, aa, aaa, \ldots\}$
  - $\{w \in \{a, b\}^* \mid w_1 = a \wedge w_{|w|} = a\} = \{a, aa, aaa, aba, aaaa, abaa, aaba, abba, \ldots\}$

WH: Darstellung von Wörtern

extensional durch Angabe der Symbole in ihrer Reihenfolge
Beispiele: $u = 321$,
$v = abababababa$,
$w = w_1 \cdots w_4$ mit $w_1 = w_2 = w_3 = a$, $w_4 = b$

intensional durch Angabe einer Eigenschaft, die für jeden Index i das i-te Symbol eindeutig bestimmt.

Beispiele:

$u \in \{0, \ldots, 4\}^3$ mit $\forall i \in \{1, \ldots, 3\} : u_i = 4 - i$,

$$v \in \{a, b\}^{11} \text{ mit } \forall i \in \{1, \ldots, 11\} : v_i = \begin{cases} a & \text{falls } i \in 2\mathbb{N} + 1 \\ b & \text{sonst} \end{cases}$$

$$w \in \{a, b\}^4 \text{ mit } w_4 = b \wedge \forall i \in \{1, \ldots, 3\} : w_i = a$$

WH: Darstellung von Sprachen

extensional durch Angabe der Elemente
(nur Beschreibung endlicher Sprachen möglich)
Beispiele: $\{\varepsilon, a, aa, aaa\}$, $\{b, ba, baa, baaa\}$,
$\{a, b, aa, bb, aaa, bbb\}$

intensional durch Angabe einer Eigenschaft, die genau alle Wörter der Sprache haben.
(auch Beschreibung unendlicher Sprachen möglich)
Beispiele: $\{w \in \{a\}^* \mid |w| \leq 3\}$,
$\{w \in \{a, b\}^* \mid w_1 = b \wedge \forall i \in \{2, \ldots, |w|\} : w_i = a\}$,
$\{w \in \{a, b\}^* \mid \forall i \in \{2, \ldots, |w|\} : w_i = w_1\}$

später in dieser LV noch mehr Formalismen zur endlichen Beschreibung von eingeschränkten Sprachklassen
(reguläre Ausdrücke, Grammatiken, Automaten, ...)

WH: Operationen auf Wörtern

Operationen auf Wörtern $u, v \in A^*$:

Verkettung $\circ : A^* \times A^* \to A^*$, wobei

$$\forall u \in A^* \; \forall v \in A^* \; \forall i \in \{1, \ldots, |u| + |v|\} :$$

$$(u \circ v)_i = \begin{cases} u_i & \text{falls } i \leq |u| \\ v_{i-|u|} & \text{sonst} \end{cases}$$

assoziativ, nicht kommutativ, ε ist neutral

Damit ist $(A^*, \circ, \varepsilon)$ ein Monoid.

Spiegelung $^R : A^* \to A^*$, wobei

$$\forall u \in A^* \; \forall i \in \{1, \ldots, |u|\} : u_i^R = u_{|u|+1-i}$$

$u \in A^*$ heißt Palindrom gdw. $u^R = u$

Fakt

- *Für jedes Wort $u \in A^*$ gilt $\left(u^R\right)^R = u$.*
- *Für je zwei beliebige Wörter $u, v \in A^*$ gilt $(u \circ v)^R = v^R \circ u^R$.*

ÜA: Beweis

WH: Relationen zwischen Wörtern

Präfix $\sqsubseteq \subseteq A^* \times A^*$:

$$\forall u \in A^* \; \forall v \in A^* : ((u \sqsubseteq v) \quad \leftrightarrow \quad (\exists w \in A^* \; (u \circ w = v)))$$

z.B. *tom* $\sqsubseteq$ *tomate* (mit $w = ate$)

Suffix (Postfix):

$$\forall u \in A^* \; \forall v \in A^* : (u \text{ Suffix von } v \quad \leftrightarrow \quad (\exists w \in A^* \; (w \circ u = v)))$$

z.B. *enten* ist Suffix von *studenten* (mit $w = stud$)

Infix (Faktor, zusammenhängendes Teilwort):

$$\forall u \in A^* \; \forall v \in A^* : (u \text{ Infix von } v \; \leftrightarrow \; (\exists w \in A^* \; \exists w' \in A^* : (w \circ u \circ w' = v)))$$

z.B. *oma* ist Infix von *tomate* (mit $w = t$ und $w' = te$)

Präfix-, Suffix- und Infixrelation sind Halbordnungen
(aber keine totalen Ordnungen).

Weitere Ordnungen auf Wörtern

Jede totale Ordnung $<$ auf dem Alphabet A definiert totale Ordnungen auf A^*:

lexikographische Ordnung auf A^* ($\leq_{\text{lex}} \subseteq A^* \times A^*$):

$$\forall u, v \in A^* : u \leq_{\text{lex}} v \text{ gdw.}$$

1. $u \sqsubseteq v$ oder
2. $\exists w \in A^* \; \exists x, y \in A : x < y \wedge wx \sqsubseteq u \wedge wy \sqsubseteq v$

quasi-lexikographische Ordnung auf A^* ($\leq_{\text{qlex}} \subseteq A^* \times A^*$):

$$\forall u, v \in A^* : u \leq_{\text{qlex}} v \text{ gdw.}$$

1. $|u| < |v|$ oder
2. $|u| = |v| \wedge u \leq_{\text{lex}} v$

Beispiele: für $A = \{a, b\}$ mit $a < b$

- $ab \sqsubseteq aba$, $ab \leq_{\text{lex}} aba$, $ab \leq_{\text{qlex}} aba$
- $abab \not\sqsubseteq abba$, aber $abab \leq_{\text{lex}} abba$ und $abab \leq_{\text{qlex}} abba$,
- $aaa \leq_{\text{lex}} ab$, aber $aaa \not\leq_{\text{qlex}} ab$
- $ab \not\leq_{\text{lex}} aaba$, aber $ab \leq_{\text{qlex}} aaba$

WH: Sprachen als Mengen

Sprachen $L \subseteq A^*$ sind Mengen von Wörtern
Eigenschaften: leer, endlich, abzählbar, überabzählbar
Mengenrelationen auf Sprachen:

$$L \subseteq L' \quad \text{gdw.} \quad \forall w \in A^* : w \in L \rightarrow w \in L' \text{ gilt}$$
$$L = L' \quad \text{gdw.} \quad \forall w \in A^* : w \in L \leftrightarrow w \in L' \text{ gilt}$$

Mengenoperationen auf Sprachen:

$$\begin{aligned} L \cup L' &= \{w \mid w \in L \vee w \in L'\} \\ L \cap L' &= \{w \mid w \in L \wedge w \in L'\} \\ L \setminus L' &= \{w \mid w \in L \wedge w \notin L'\} \\ L \Delta L' &= (L \setminus L') \cup (L' \setminus L) \end{aligned}$$

Komplement einer Sprache $L \subseteq A^*$: $\overline{L} = A^* \setminus L$
Beispiel:

$$L = \bigcup_{n \in 3\mathbb{N}} A^n \qquad \overline{L} = \bigcup_{n \in \{3i+1 \mid i \in \mathbb{N}\}} A^n \cup \bigcup_{n \in \{3i+2 \mid i \in \mathbb{N}\}} A^n$$

WH: Operationen auf Sprachen

Spiegelung $L^R = \{w^R \mid w \in L\}$

Verkettung $\circ$ von Sprachen:

$$L_1 \circ L_2 = \{u \circ v \mid u \in L_1 \wedge v \in L_2\}$$

iterierte Verkettung von Sprachen $L \subseteq A^*$

$$L^0 = \{\varepsilon\} \qquad \forall n \in \mathbb{N}: \quad L^{n+1} = L^n \circ L = \underbrace{L \circ \cdots \circ L}_{n+1-\text{mal}}$$

$$L^* = \bigcup_{n \in \mathbb{N}} L^n \qquad L^+ = \bigcup_{n \in \mathbb{N} \setminus \{0\}} L^n$$

Spezialfall $L = \{u\} \in A^*$:

$$u^n = \underbrace{u \cdots u}_{n-\text{mal}} \in A^*, \quad u^* = \{u\}^* = \{u^n \mid n \in \mathbb{N}\} \subseteq A^*$$

$$u^+ = u^* \setminus \{\varepsilon\} = \{u\}^+ = \{u^n \mid n \in \mathbb{N} \setminus \{0\}\} \subseteq A^*$$

WH: Reguläre Ausdrücke – Syntax

Die Menge $\mathrm{RegExp}(A)$ aller regulären Ausdrücke über einem Alphabet A ist (induktiv) definiert durch:

- IA $\emptyset \in \mathrm{RegExp}(A)$,
  $\varepsilon \in \mathrm{RegExp}(A)$ und
  für jedes Symbol $a \in A$ gilt $a \in \mathrm{RegExp}(A)$
- IS für alle $E \in \mathrm{RegExp}(A)$ und $F \in \mathrm{RegExp}(A)$ gilt
  $(E + F), EF, (E)^* \in \mathrm{RegExp}(A)$.

Beispiele: $\varepsilon + a$, $\varepsilon + \emptyset$, $(a + \emptyset)^*$, $\varepsilon + ((ab)^* a)^*$

dieselbe Definition kürzer: $\mathrm{RegExp}(A) = \mathrm{Term}(\Sigma_F, \emptyset)$
für die Signatur
$\Sigma_F = \{(\emptyset, 0), (\varepsilon, 0), (^*, 1), (+, 2), (\cdot, 2)\} \cup \{(a, 0) \mid a \in A\}$
(Baumdarstellung)

WH: Reguläre Ausdrücke – Semantik

Jeder reguläre Ausdruck $E \in \mathrm{RegExp}(A)$ repräsentiert eine Sprache $L(E) \subseteq A^*$.

$$
\begin{array}{lrcl}
 & L(\emptyset) & = & \emptyset \\
 & L(\varepsilon) & = & \{\varepsilon\} \\
\forall a \in A: & L(a) & = & \{a\} \\
\forall E, F \in \mathrm{RegExp}(A): & L(E + F) & = & L(E) \cup L(F) \\
\forall E, F \in \mathrm{RegExp}(A): & L(EF) & = & L(E) \circ L(F) \\
\forall E, F \in \mathrm{RegExp}(A): & L(E^*) & = & (L(E))^*
\end{array}
$$

Eine Sprache $L \subseteq A^*$ heißt genau dann regulär, wenn ein regulärer Ausdruck $E \in \mathrm{RegExp}(A)$ mit $L = L(E)$ existiert.

Beispiel: Die Menge L aller Dezimaldarstellungen natürlicher Zahlen ist regulär wegen $L = L\,(0 + (1 + 2 + \cdots + 9)(0 + 1 + \cdots + 9)^*)$

WH: Beispiele

Für $A = \{a, b\}$ gilt

$$
\begin{aligned}
L(ab^*) &= \{a, ab, abb, abbb, abbbb, \ldots\} = \{ab^i \mid i \in \mathbb{N}\} \\
L((ab)^*) &= \{\varepsilon, ab, abab, ababab, \ldots\} = \{(ab)^i \mid i \in \mathbb{N}\} \\
L((a+b)^*) &= \{a, b\}^* \\
L(a^*b^*) &= \{u \circ v \mid u \in a^* \wedge v \in b^*\} \\
L((a^*b^*)^*) &= \{a, b\}^* \\
L((a+b)^*aba) &= \{u \circ aba \mid u \in A^*\}^*
\end{aligned}
$$

Reguläre Ausdrücke ermöglichen eine endliche Darstellung unendlicher Sprachen.

Äquivalenz regulärer Ausdrücke

Zwei reguläre Ausdrücke $E, F \in \mathrm{RegExp}(A)$ heißen genau dann äquivalent, wenn $L(E) = L(F)$ gilt.

Beispiele (ÜA):

- $(a + b)^*$, $(a^* + b^*)^*$ und $a^*(ba^*)^*$ sind paarweise äquivalent
- ab^* und $(ab)^*$ sind nicht äquivalent
- $(11 + 0 + 110 + 011)^*$ und $(11 + 0)^*$ sind ...

Fakt

Die Äquivalenz regulärer Ausdrücke ist eine Äquivalenzrelation.

ÜA: Beweis

Interessante Fragen für formale Sprachen

- Ist ein gegebenes Wort w in der Sprache L enthalten?
  (häufig Wortproblem genannt)
- Enthält die Sprache L nur endlich viele Wörter?
- Gilt $L_1 \subseteq L_2$ für zwei gegebene Sprachen L_1 und L_2?
- Gilt $L_1 = L_2$ für zwei gegebene Sprachen L_1 und L_2?

Fragen zur Regularität:

- Lässt sich die Sprache L durch einen regulären Ausdruck definieren?
  (Gilt $\exists E \in \mathrm{RegExp}(A) : L = L(E)$?)
- Woran kann man erkennen, ob sich eine Sprache durch einen regulären Ausdruck definieren lässt?
- Gilt $L(E) = \emptyset$ für einen gegebenen regulären Ausdruck E?
- Gilt $L(E) = A^*$ für einen gegebenen regulären Ausdruck E?
- Ist ein gegebenes Wort w in der durch den regulären Ausdruck E definierten Sprache $L(E)$ enthalten?

Alle Antworten sind für endliche Sprachen einfach,

aber für unendliche Sprachen oft schwierig.

Wortproblem praktisch

Eingabe : Sprache $L \subseteq A^*$, Wort $w \in A^*$
Frage: Gilt $w \in L$?
Ausgabe: ja oder nein

Beispiele:

- Syntaktische Tests:
  - Ist die gegebene Zeichenkette die Dezimaldarstellung einer ganzen Zahl?
    (Sprache: Menge aller gültigen Dezimaldarstellungen)
  - Ist die gegebene Zeichenkette eine korrekt geformte Email-Adresse (der HTWK)?
  - Ist der gegebene Quelltext ein syntaktisch korrektes Java-Programm?
  - Ist die gegebene Zeichenkette die Binärdarstellung einer geraden Zahl? (durch drei teilbaren Zahl, usw.)
- Folgen von Aktionen:
  - An- und Ausziehen (in umgekehrter Reihenfolge)
  - Ist eine Folge von Aktionen möglich / zulässig ?
  - Führt eine Folge von Eingaben zu einem Fehler?

Wortersetzungssysteme

Alphabet A

Wortersetzungsregel $(l, r) \in A^* \times A^*$ (geschrieben $l \to r$)

Wortersetzungssystem endliche Menge von Wortersetzungsregeln

Beispiele:

- Regel $ba \to ab$,
- Wortersetzungssystem $S = \{a \to ab, ba \to c, abc \to \varepsilon\}$

Anwendung von Wortersetzungsregeln

Eine Regel $l \to r$ ist auf ein Wort $w \in A^*$ anwendbar,
falls l ein Infix von w ist ($w = v \circ l \circ v'$).

Beispiel: Regel $oma \to u$ ist

- auf *isomatte* anwendbar, $v = is, v' = tte$,
- auf *tomate* anwendbar, $v = t, v' = te$,
- auf *matte*, *sommer* und *normal* nicht anwendbar.

Eine Anwendung der Regel $l \to r$ auf ein Wort
$w = v \circ l \circ v'$ ergibt das Wort $v \circ r \circ v'$.
(Ersetzung des Infix l durch r)

Beispiel: $ab \to a$ angewendet auf $baababa = v \circ l \circ v'$

- mit $v = ba$ und $v' = aba$ ergibt $baaaba$
- mit $v = baab$ und $v' = a$ ergibt $baabaa$

Anwendung einer Regel auf ein Wort an einer Position im Wort

Ableitungsschritt

Ableitungsschritt $(w, (l \to r), p, w')$ im Wortersetzungssystem S mit

- Ausgangswort w,
- auf w anwendbare Regel $l \to r$ aus S,
- Position $p \in \{1, \ldots, |w|\}$ im Wort w, an der der Infix l beginnt
- w' ist das nach Anwendung der Regel $l \to r$ an Position p auf w entstandene Wort.

Beispiel:
$S = \{ab \to ba, a \to b\}$, $w = aba$
mögliche Ableitungsschritte in S:
$(aba, (ab \to ba), 1, baa)$
$(aba, (a \to b), 3, abb)$
$(aba, (a \to b), 1, bba)$

Ein-Schritt-Ableitungsrelation

Jedes Wortersetzungssystem $S \subseteq A^* \times A^*$ definiert eine Relation $\rightarrow_S \subseteq A^* \times A^*$, wobei genau dann $w \rightarrow_S w'$ gilt, wenn ein Ableitungsschritt $(w, (l \rightarrow r), p, w')$ mit $(l \rightarrow r) \in S$ existiert.

Beispiel: Für $S = \{ab \rightarrow ba, a \rightarrow b\}$ gilt

- $aba \rightarrow_S baa$ wegen $(aba, (ab \rightarrow ba), 1, baa)$
- $aba \rightarrow_S bba$ wegen $(aba, (a \rightarrow b), 1, bba)$
- $aba \rightarrow_S abb$ wegen $(aba, (a \rightarrow b), 3, abb)$
- $aba \not\rightarrow_S bbb$

Ableitungen

Eine Folge von Ableitungsschritten

$$(u, (l_1 \to r_1), p_1, u_2), (u_2, (l_2 \to r_2), p_2, u_3), \cdots, (u_{n-1}, (l_{n-1} \to r_{n-1}, p_{n-1}, v)$$

im Wortersetzungssystem S heißt Ableitung von u nach v in S.

Beispiel: $S = \{ab \to ba, a \to b\}$, $u = aba$
Folge von Ableitungsschritten
$(aba, (ab \to ba), 1, baa)$, $(baa, (a \to b), 3, bab)$, $(bab, (a \to b), 2, bbb)$

$$\underline{ab}a \overset{ab \to ba}{\longrightarrow} ba\underline{a} \overset{a \to b}{\longrightarrow} b\underline{a}b \overset{a \to b}{\longrightarrow} bbb$$

Länge der Ableitung = Anzahl der Ableitungsschritte

In jedem System S existiert für jedes $u \in A^*$ die leere Ableitung (der Länge 0) von u nach u.

Wiederholung: Hüllen binärer Relationen

$R \cup I_M$ heißt reflexive Hülle von $R \subseteq M^2$
(mit Identität $I_M = \{(x, x) \mid x \in M\}$)

$R \cup R^{-1}$ heißt symmetrische Hülle von $R \subseteq M^2$
(mit inverser Relation $R^{-1} = \{(y, x) \mid (x, y) \in R\}$)

Wiederholung: Verkettung $\circ$ der Relationen $R \subseteq M^2$ und $S \subseteq M^2$

$$R \circ S = \{(x, z) \in M^2 \mid \exists y \in M : (x, y) \in R \wedge (y, z) \in S\}$$

Iterierte Verkettung von $R \subseteq M^2$ mit sich selbst:

$$\begin{aligned} R^0 &= I_M \\ R^{n+1} &= R^n \circ R \\ R^+ &= \bigcup_{n \in \mathbb{N} \setminus \{0\}} R^n \subseteq M^2 \qquad \text{transitive Hülle} \\ R^* &= \bigcup_{n \in \mathbb{N}} R^n \subseteq M^2 \qquad \text{reflexiv-transitive Hülle} \end{aligned}$$

Ersetzungsrelation

Jedes Wortersetzungssystem $S \subseteq (A^* \times A^*)$ definiert die Ersetzungsrelation $\rightarrow_S^* \subseteq (A^* \times A^*)$, wobei genau dann $u \rightarrow_S^* v$ gilt, wenn eine Ableitung von u nach v existiert.

Beispiel: $S = \{a \rightarrow aa\}$,

- für jedes $n \geq 1$ gilt $ba \rightarrow_S^* b\underbrace{a \cdots a}_{n}$

  wegen $ba \rightarrow_S baa \rightarrow_S baaa \rightarrow_S \cdots \rightarrow_S b\underbrace{a \cdots a}_{n}$
- $b \rightarrow_S^* b$, aber für kein Wort $w \neq b$ gilt $b \rightarrow_S^* w$

($\rightarrow_S^*$ ist die reflexive transitive Hülle von $\rightarrow_S$)

(Beschränktes) Münzenspiel als Wortersetzungssystem

Symbole Höhe der Stapel $A = \{0, \ldots, 4\}$

Konfiguration Wort $w \in A^5$

Startkonfiguration Wort $W = 00400$

Spielzug Anwendung einer Regel aus dem Wortersetzungssystem

$$S = \left\{ i(j+2)k \to (i+1)j(k+1) \mid \begin{array}{l} i,j,k \in \{0,\ldots,4\} \\ \wedge\ i+(j+2)+k \leq 4 \end{array} \right\}$$

$$= \left\{ \begin{array}{l} 040 \to 121, 220 \to 301, 022 \to 103, 121 \to 202 \\ 030 \to 111, 130 \to 211, 031 \to 112, \\ 020 \to 101, 120 \to 201, 021 \to 102 \end{array} \right\}$$

Das Paar $(S, 00400)$ definiert die Menge (Sprache) aller aus 00400 durch $\to_S^*$ erreichbaren Konfigurationen:

$$L(S, 00400) = \{00400, 01210, 02020, 10120, 02101, 10201, 11011\}$$

Auf das Wort 11011 ist keine Regel aus S anwendbar.
Deshalb heißt 11011 Normalform bzgl. S

Was bisher geschah

- Alphabet, Wort, Sprache
- Operationen und Relationen auf Wörtern und Sprachen
- interessante Fragen für Sprachen und Wörter

Reguläre Ausdrücke

- Syntax, Semantik
- endliche Darstellung evtl. unendlicher Sprachen

Wortersetzungssysteme $P \subseteq A^* \times A^*$

- Wortersetzungssregel $l \to r$ mit $l, r \in A^*$
- Ableitung in P: endliche Folge von Ersetzungsschritten
- Ausdrucksstärke:
  - Repräsentation von (auch unendlichen) Sprachen
  - Modellierung von Zustandsübergängen
  - Ausführen von Berechnungen

Łukasiewicz-Sprache $L(\{b \to abb\}, b)$

Ableitbare Wörter über Teilalphabet

Beispiele: $A = \{a, b, c\}$,

$$S = \left\{ \begin{array}{lcl} c & \to & aca, \\ c & \to & bcb, \\ c & \to & \varepsilon, \\ c & \to & a, \\ c & \to & b \end{array} \right\}$$

$$L(S, c) = \{u \circ d \circ u^R \mid u \in \{a, b\}^* \wedge d \in \{a, b, c, \varepsilon\}\}$$

Menge aller Wörter in $L(S, c) \cap \{a, b\}^*$:
alle Palindrome in $\{a, b\}^*$

c ist Hilfssymbol zur Erzeugung der Palindrome

Natürliche Sprache

Wortersetzungssystem S enthält die Regeln:

Satz	$\rightarrow$	Subjekt Prädikat .
Subjekt	$\rightarrow$	mArtikel mSubstantiv
Subjekt	$\rightarrow$	wArtikel wSubstantiv
mArtikel	$\rightarrow$	Der
wArtikel	$\rightarrow$	Die
mSubstantiv	$\rightarrow$	Hund
wSubstantiv	$\rightarrow$	Sonne
Prädikat	$\rightarrow$	bellt
Prädikat	$\rightarrow$	scheint

Alphabet $A = \{\text{Der}, \text{Die}, \text{Hund}, \text{Sonne}, \text{scheint}, \text{bellt}, .\} \cup$
{ Satz,Subjekt, Prädikat, wArtikel, mArtikel, wSubstantiv, mSubstantiv}

Ableitbare Wörter in $L(S,$ Satz) ohne Hilfssymbole aus der Menge
{Satz, Subjekt, Prädikat, mArtikel, wArtikel, mSubstantiv, wSubstantiv}:
Menge grammatisch korrekter deutscher Sätze (dieser einfachen Form mit ausschließlich den Worten Der, Die, Hund, Sonne, scheint, bellt).

Aussagenlogische Formeln

Wortersetzungssystem S enthält die Regeln

Formel	$\to$	Variable
Formel	$\to$	Konstante
Formel	$\to$	($\neg$ Formel)
Formel	$\to$	(Formel $\vee$ Formel)
Formel	$\to$	(Formel $\wedge$ Formel)
Variable	$\to$	p
Variable	$\to$	q
Konstante	$\to$	$\mathbb{t}$
Konstante	$\to$	$\mathbb{f}$

Alphabet $A = \{\mathbb{t}, \mathbb{f}, p, q, \neg, \vee, \wedge, (,)\} \cup \{$ Formel,Variable, Konstante$\}$

$$\text{Formel} \to_S (\text{Formel} \wedge \text{Formel}) \to_S^2 (\text{Formel} \wedge \mathbb{f}) \to_S^* ((p \vee (\neg q)) \wedge \mathbb{f})$$

Wörter in $L(S, \text{ Formel }) \cap \{\mathbb{t}, \mathbb{f}, p, q, \neg, \vee, \wedge, (,)\}^*$:
Menge $\text{AL}_{\{\neg,\vee,\wedge,\mathbb{t},\mathbb{f}\}}(\{p, q\})$ aller aussagenlogischen Formeln mit Aussagenvariablen aus der Menge $\{p, q\}$
und Junktoren aus der Menge $\{\neg, \vee, \wedge, \mathbb{t}, \mathbb{f}\}$

Aussagenlogische DNF

Wortersetzungssystem S enthält die Regeln

DNF	$\to$	Minterm $\vee$ DNF
DNF	$\to$	Minterm
Minterm	$\to$	Literal $\wedge$ Minterm
Minterm	$\to$	Literal
Literal	$\to$	$\neg$ Variable
Literal	$\to$	Variable
Variable	$\to$	p
Variable	$\to$	q

Alphabet $A =$ $\{p, q, \neg, \vee, \wedge\}$
$\cup\{$ DNF, Minterm, Literal,Variable$\}$

$$\text{DNF} \to_S \text{Minterm} \vee \text{DNF} \to_S^3 p \vee \text{DNF} \to_S^* p \vee q \wedge \neg p \vee \neg q$$

Wörter in $L(S, \text{DNF}) \cap \{p, q, \neg, \vee, \wedge\}^*$:
Menge AL($\{p, q\}$) aller disjunktiven Normalformen mit
Aussagenvariablen aus der Menge $\{p, q\}$

Dezimaldarstellung natürlicher Zahlen

Wortersetzungssystem S enthält die Regeln

Zahl	$\to$	0
Zahl	$\to$	1Ziffernfolge
	$\vdots$	
Zahl	$\to$	9Ziffernfolge
Ziffernfolge	$\to$	0Ziffernfolge
	$\vdots$	
Ziffernfolge	$\to$	9Ziffernfolge
Ziffernfolge	$\to$	ε

Alphabet $\{0, \ldots, 9\} \cup \{$ Zahl, Ziffernfolge $\}$

Zahl $\to_S$ 3Ziffernfolge $\to_S$ 32Ziffernfolge $\to_S$ 327Ziffernfolge $\to_S$ 327

Wörter in $L(S,$ Zahl $) \cap \{0, \ldots, 9\}^*$:
Menge aller Dezimaldarstellungen natürlicher Zahlen

Programmiersprachen

Java-Syntax (Ausschnitt) in Backus-Naur Form (BNF)
(John Backus, Peter Naur)

```
<while statement>::=while(<expression>)<statement>

<expression>::=<assignment expression>

<assignment expression>::=<conditional expression>|<assignment>

<assignment>::=<left hand side><assignment operator><assignment expression>

<left hand side>::=<expression name>|<field access>|<array access>

<assignment operator>::=  = | *= | /= | %= | += | -= | <<= | >>= | >>>=
                        | &= | ^= | |=
```

$a \to r_1|r_2|\ldots|r_n$ statt mehrerer Regeln $a \to r_1, \ldots, a \to r_n$
$::=$ statt $\to$ (in ASCII darstellbar)
Hilfssymbole markiert durch $<$ und $>$

Definition Grammatik

Grammatik $G = (N, T, P, S)$ ist definiert durch

Nichtterminalsymbole: endliche Menge N
(Hilfssymbole)

Terminalsymbole: endliche Menge T
(Alphabet der erzeugten Sprache)

Wortersetzungssystem: $P \subseteq (N \cup T)^+ \times (N \cup T)^*$ (Produktionen)

Startsymbol $S \in N$

Beispiel: $G = (N, T, P, S)$ mit

$$
\begin{aligned}
N &= \{S\}, \\
T &= \{0, 1\}, \\
P &= \left\{ \begin{array}{lcl} S & \rightarrow & 0S1 \\ S & \rightarrow & \varepsilon \end{array} \right\}
\end{aligned}
$$

Grammatiken: Beispiele

- $G = (N, T, P, E)$ mit $N = \{E, F, G\}$, $T = \{(,), a, +, \cdot\}$ und

$$P = \left\{ \begin{array}{lcl} E & \to & G, \\ E & \to & E + G, \\ G & \to & F, \\ G & \to & G \cdot F, \\ F & \to & a, \\ F & \to & (E) \end{array} \right\} = \left\{ \begin{array}{lcl} E & \to & G \mid E + G, \\ G & \to & F \mid G \cdot F, \\ F & \to & a \mid (E) \end{array} \right\}$$

- $G = (N, T, P, S)$ mit $N = \{S, A, B, C\}$, $T = \{a, b, c\}$

$$P = \left\{ \begin{array}{lcl} S & \to & aSBC, \\ S & \to & aBC, \\ CB & \to & BC, \\ aB & \to & ab, \\ bB & \to & bb, \\ bC & \to & bc, \\ cC & \to & cc \end{array} \right\}$$

Ableitungen in Grammatiken

Ableitung in Grammatik $G = (N, T, P, S)$:
Ableitung im Ersetzungssystem P mit Startwort S

Beispiel: $G = (N, T, P, S)$ mit

$$
\begin{aligned}
N &= \{S, A, B\} \\
T &= \{0, 1\} \\
P &= \left\{ \begin{array}{lcl} S & \to & 0SA \\ S & \to & 0A \\ A & \to & 1 \\ B & \to & A \end{array} \right\}
\end{aligned}
$$

Durch Grammatiken definierte Sprachen

Grammatik $G = (N, T, P, S)$ definiert die Sprache

$$L(G) = \{w \in T^* \mid S \rightarrow_P^* w\} = L(P, S) \cap T^*$$

Beispiel: $G = (N, T, P, S)$ mit

$$\begin{aligned} N &= \{S, Z\} \\ T &= \{0, 1\} \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & 1Z, \\ S & \rightarrow & 0, \\ Z & \rightarrow & 0Z, \\ Z & \rightarrow & 1Z, \\ Z & \rightarrow & \varepsilon \end{array} \right\} \end{aligned}$$

definiert die Sprache $L(G) = \ldots$

Äquivalenz von Grammatiken

Zwei Grammatiken G_1 und G_2 heißen genau dann äquivalent, wenn $L(G_1) = L(G_2)$ gilt.

Beispiel: $G_1 = (N_1, \{0,1\}, P_1, S)$ mit $N_1 = \{S, A, B\}$ und

$$P_1 = \left\{ \begin{array}{lcl} S & \to & 0SA \\ S & \to & 0A \\ A & \to & 1 \\ B & \to & A \end{array} \right\}$$

und $G_2 = (N_2, \{0,1\}, P_2, S')$ mit $N_2 = \{S'\}$ und

$$P_2 = \{S' \to 0S'1, S' \to 01\}$$

sind äquivalent wegen $L(G_1) = L(G_2) = \ldots$

Äquivalenz von Grammatiken ist eine Äquivalenzrelation. (ÜA)

Grammatiken mit ε-Regeln

ε-Regel : Grammatik-Regel der Form $l \to \varepsilon$

Beispiele:

- $G_1 = (\{A, B\}, \{0, 1\}, P_1, A)$ mit
  $P_1 = \{A \to 1B0, B \to 1B0, B \to \varepsilon\}$
  ist äquivalent zu $G_2 = (\{A\}, \{0, 1\}, P_2, A)$ mit
  $P_2 = \{A \to 1A0, A \to 10\}$
  (ohne ε-Regel)
- $G = (\{A\}, \{0, 1\}, P, A)$ mit $P = \{A \to 1A0, A \to \varepsilon\}$
  ist nicht äquivalent zu einer Grammatik ohne ε-Regel

Chomsky-Hierarchie

(Noam Chomsky)

Eine Grammatik $G = (N, T, P, S)$ heißt vom Chomsky-Typ

0 immer,

1 , falls für jede Regel $(l \to r) \in P$ gilt: $|l| \leq |r|$
(monoton, kontextsensitiv)

2 , falls Typ 1 und für jede Regel $(l \to r) \in P$ gilt: $l \in N$
(kontextfrei)

3 , falls Typ 2 und für jede Regel $(l \to r) \in P$ gilt:
$l \in N$ und $r \in (T \cup (T \circ N))$
(regulär)

Eine Sprache $L \subseteq T^*$ heißt vom (Chomsky-)Typ i für $i \in \{0, \ldots, 3\}$, falls i die größte Zahl ist, für die eine Grammatik G vom Typ i mit $L \setminus \{\varepsilon\} = L(G)$ existiert.

$\mathcal{L}_i$ bezeichnet die Menge aller Sprachen vom Typ i.

Achtung: Es gibt Sprachen, die nicht durch eine Grammatik erzeugt werden können, also keinen Chomsky-Typ haben (Warum?)

Beispiele

- $G_1 = (\{S\}, \{a, b\}, P, S)$ mit
  $P = \{S \to aSb, S \to ab, S \to a, S \to b\}$ ist vom Typ 2
- $G_2 = (\{S\}, \{a, b\}, P, S)$ mit
  $P = \{S \to aSb, S \to a, S \to b, S \to \varepsilon\}$ ist vom Typ 0
  aber $L(G_2)$ ist vom Typ 2,
  weil $L(G_2) \setminus \{\varepsilon\} = L(G_2')$ für $G_2' = (\{S\}, \{a, b\}, P', S)$ mit
  $P' = \{S \to aSb, S \to a, S \to b, S \to ab\}$
- $G_3 = (\{S, A, B\}, \{a, b\}, P, S)$ mit
  $P = \{S \to aA, A \to aB, B \to bB, B \to b\}$ ist vom Typ 3
- $G_4 = (\{A, B, C\}, \{a, b, c\}, P, A)$ mit

$$P = \left\{ \begin{array}{lcl} A & \to & aABC, \\ A & \to & aBC, \\ CB & \to & BC, \\ aB & \to & ab, \\ bB & \to & bb, \\ bC & \to & bc, \\ cC & \to & cc \end{array} \right\}$$

  ist vom Typ 1

Wortproblem für Typ-1-Sprachen

gegeben : Grammatik $G = (N, T, P, S)$ vom Chomsky-Typ 1,
Wort $w \in T^*$
Frage : Gilt $w \in L(G)$?

Satz

Es existiert ein Algorithmus, welcher für jede beliebige Eingabe (G, w), *wobei*

- T *ein endliches Alphabet,*
- $w \in T^*$ *und*
- G *eine monotone Grammatik (Chomsky-Typ 1) über* T *sind*

die Wahrheit der Aussage $w \in L(G)$ *(in endlicher Zeit) korrekt beantwortet.*

(folgt aus entsprechendem Satz für nichtverkürzende Wortersetzungssysteme)
Im weiteren Verlauf dieses Semesters:
spezielle (effizientere) Verfahren für Grammatiken von Typ 2 und 3

WH: Mächtigkeit unendlicher Mengen

Eine Menge A heißt

abzählbar gdw. $\exists\, f : \mathbb{N} \to A$ surjektiv

überabzählbar sonst

Abschlusseigenschaften der Menge aller abzählbaren Mengen:
Sind alle Mengen A_i mit $i \in \mathbb{N}$ abzählbar, dann sind auch

- jede Teilmenge $B \subseteq A_1$ abzählbar
- $B = A_1 \cup A_2$ abzählbar
- $B = A_1 \times A_2$ abzählbar
  (nach erstem Diagonalverfahren von Cantor)
  z.B. $B_k = ((A_1)_m, (A_2)_n)$ für $k = \sum_{i=0}^{m+n} i + m$ (ÜA)
- $\forall k \in \mathbb{N} : B = A_1^k$ abzählbar (ÜA)

Ist jede der abzählbar vielen Mengen $A_0, A_1, A_2, \ldots$ abzählbar,
dann sind auch

- $\bigcup_{i \in \mathbb{N}} A_i$ abzählbar (Hilberts Hotel)
- $\bigtimes_{i \in \mathbb{N}} A_i$ abzählbar

(formale Beweise durch Induktion nach i)

WH: Abzählbare Mengen – Beispiele

A abzählbar gdw. $\exists f : \mathbb{N} \to A$ surjektiv

Beispiele:

- jede endliche Menge $A = \{a_0, \ldots, a_n\}$ mit

$$\forall i \in \mathbb{N} : f(i) = \begin{cases} a_1 & \text{, falls } i \leq n \\ a_0 & \text{, falls } i > n \end{cases}$$

- $\mathbb{N}$ mit $f =$ Identität
- jede Menge $M \subseteq \mathbb{N}$
- $\mathbb{N} \times \mathbb{N}$ (nach erstem Diagonalverfahren von Cantor)
- $\mathbb{Z} = \{m - n \mid (m, n) \in \mathbb{N}^2\} = \mathbb{N} \cup \{-n \mid n \in \mathbb{N} \setminus \{0\}\}$
- $\mathbb{Q} = \{m/n \mid (m, n) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})\}$
- $\forall k \in \mathbb{N} : \mathbb{N}^k$
- $\{1\}^*$
- $\{0, 1\}^*$
- A^* für jedes endliche Alphabet A (ÜA)

WH: Überabzählbare Mengen

Menge $2^{\mathbb{N}}$ aller Mengen natürlicher Zahlen ist überabzählbar, also mächtiger als $\mathbb{N}$.
(Es gibt überabzählbar viele Mengen natürlicher Zahlen.)
(indirekter) Beweis, dass $2^{\mathbb{N}}$ überabzählbar
zweites Diagonalverfahren von Cantor
Annahme: Es existiert eine surjektive Funktion $M : \mathbb{N} \to 2^{\mathbb{N}}$, d.h. für jede Menge $L \subseteq \mathbb{N}$ gilt also $L = M(k)$ für (wenigstens) ein $k \in \mathbb{N}$.

	0	1	2	$\cdots$	k	$\cdots$
$M(0)$	$0 \in M(0)$	$1 \in M(0)$	$2 \in M(0)$	$\cdots$	$k \in M(0)$	$\cdots$
$M(1)$	$0 \in M(1)$	$1 \in M(1)$	$2 \in M(1)$	$\cdots$	$k \in M(1)$	$\cdots$
$M(2)$	$0 \in M(2)$	$1 \in M(2)$	$2 \in M(2)$	$\cdots$	$k \in M(2)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	
$M(k)$	$0 \in M(k)$	$1 \in M(k)$	$2 \in M(k)$	$\cdots$	$k \in M(k)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	

Definiere Menge $L = \{i \in \mathbb{N} \mid i \notin M(i)\} \subseteq \mathbb{N}$
Wegen $L \subseteq \mathbb{N}$ existiert nach Annahme ein $k \in \mathbb{N}$ mit $L = M(k)$ und damit $k \in M(k)$ gdw. $k \notin M(k)$ (Widerspruch),
d.h. Annahme falsch, also $2^{\mathbb{N}}$ nicht abzählbar

Beispiele überabzählbarer Mengen

Mit dem zweiten Diagonalverfahren von Cantor lässt sich zeigen:

- $\mathbb{R}$ ist überabzählbar (mächtiger als $\mathbb{N}$).
  (Es gibt überabzählbar viele reelle Zahlen.)
- $[0,1]$ $\subset \mathbb{R}$ ist überabzählbar.
  (Intervall $[0,1]$ enthält überabzählbar viele reelle Zahlen.)
- $2^{(\{0,1\}^*)}$ Menge aller Sprachen $L \subseteq \{0,1\}^*$ ist überabzählbar, also mächtiger als $\{0,1\}^*$.
- $2^{(A^*)}$ ist für jedes nichtleere endliche Alphabet A überabzählbar.
  (Für jedes nichtleere endliche Alphabet A ist die Menge aller Sprachen über A überabzählbar.) (ÜA)

Ist jede Sprache regulär ?

Existiert für jedes endliche Alphabet A zu jeder Sprache $L \subseteq A^*$ ein regulärer Ausdruck E mit $L = L(E)$?

Nein, Beweis (2 Möglichkeiten):

1. konstruktiv durch Gegenbeispiel
   z.B. $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ (Nachweis erst später möglich)

2. nichtkonstruktiv (mit schon bekannten Methoden möglich):

   - Wieviele Sprachen $L \subseteq A^*$ gibt es? (Mächtigkeit von $2^{(A^*)}$) überabzählbar viele
   - Wieviele reguläre Ausdrücke über dem endlichen Alphabet A gibt es? abzählbar viele, weil
     - Alphabet $A' = A \cup \{\emptyset, \varepsilon, +, ^*, (,), \}$ endlich
     - Menge $(A')^*$ aller Wörter über A' abzählbar
     - RegExp$(A) \subset (A')^*$ (endliche Beschreibung)
     - also RegExp(A) selbst abzählbar

   Damit existieren sogar sehr viel mehr (überabzählbar viele) nicht reguläre Sprachen.

Warum sind reguläre Ausdrücke trotzdem interessant?

Wird jede Sprache von einer Grammatik erzeugt?

Existiert für jedes endliche Alphabet A zu jeder Sprache $L \subseteq A^*$ eine Grammatik $G = (N, T, P, S)$ mit $L = L(G)$?

Nein (Gegenbeispiel später)

Begründung (nichtkonstruktiv):

1. Wieviele Sprachen $L \subseteq A^*$ gibt es? (Mächtigkeit von $2^{(A^*)}$)

   überabzählbar viele

2. Wieviele Grammatiken $G = (N, T, P, S)$ gibt es?
   abzählbar viele, weil

   - Alphabet $A' = N \cup T \cup \{(,), , , \{, \}, \rightarrow, \varepsilon\}$ endlich
   - Menge $(A')^*$ aller Wörter über A' abzählbar
   - jede Grammatik ist ein Wort aus $(A')^*$
     (endliche Beschreibung der Sprache)
   - Menge aller Grammatiken ist Teilmenge der abzählbaren Menge $(A')^*$, also selbst abzählbar

Also existieren überabzählbar viele Sprachen, die nicht durch Grammatiken erzeugt werden können.

Warum sind Grammatiken trotzdem interessant?

Was bisher geschah

- Operationen auf Sprachen:
  - Mengenoperationen $\cup, \cap, \setminus, \overline{\ }, \Delta$
  - Sprachoperationen ${}^R, \circ, {}^*$
- reguläre Ausdrücke RegExp(X):

  IA: $\{\varepsilon, \emptyset\} \cup X \subseteq \text{RegExp}(X)$
  IS: $\forall E, F \in \text{RegExp}(X) : \{E + F, EF, E^*\} \subseteq \text{RegExp}(X)$

  Die Menge aller regulären Sprachen ist abgeschlossen unter $\cup, \circ, {}^*$
- Grammatiken, Chomsky-Hierarchie

endliche Automaten: NFA $A = (X, Q, \delta, I, F)$

- vollständige, deterministische NFA
- Isomorphie von NFA
- Akzeptanz von Wörtern durch NFA
- von NFA, DFA, akzeptierte Sprache $L(A)$
- Äquivalenz von NFA
- Menge REC(NFA) aller NFA-akzeptierbaren Sprachen

Vervollständigung von NFA

Motivation:
In vollständigen NFA existiert zu jedem Wort wenigstens ein Weg, Akzeptanz jedes Wortes durch Menge der mit dem letzten Symbol erreichten Zustände auf allen Wegen feststellbar.

Satz

Zu jedem NFA A existiert ein äquivalenter vollständiger NFA B.

Konstruktionsidee für $A = (X, Q_A, \delta_A, I, F)$ zu $B = (X, Q_B, \delta_B, I, F)$:

1. Hinzufügen eines zusätzlichen nichtakzeptierenden Zustandes e
   $Q_B = Q_A \cup \{e\}$
2. Hinzufügen zusätzlicher Kanten (p, e)
   $\forall a \in X : \delta_B(a) = \delta_A(a) \cup \{(p, e) \mid \neg\exists q \in Q_A : (p, q) \in \delta_A(a)\}$

Beispiel: $B = (\{a, b, c\}, \{0, 1, 2, 3, 4, 5\}, \delta, \{0\}, \{2, 4\})$ mit
$\delta(a) = \{(0, 0), (1, 2), (3, 4), (4, 4), (2, 5), (5, 5)\}$,
$\delta(b) = \{(0, 1), (0, 3), (1, 5), (2, 5), (3, 5), (4, 5), (5, 5)\}$
und $\delta(c) = \{(3, 0), (0, 5), (1, 5), (2, 5), (4, 5), (5, 5)\}$
(A enthält nur schwarze Elemente, Vervollständigung B auch alle roten)

Konstruktion von DFA aus NFA

Motivation: In DFA existiert zu jedem Wort w höchstens ein akzeptierender Weg, Feststellen der Akzeptanz in $|w|$ Schritten möglich.
Beispiel: $A = (\{a, b\}, \{0, 1, 2, 3, 4\}, \delta, \{0, 4\}, \{3\})$ mit
$\delta(a) = \{(0, 0), (4, 4)(0, 1), (1, 3), (3, 3)\}$ und
$\delta(b) = \{(0, 0), (4, 4), (4, 2), (2, 3), (3, 3)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q_A, \delta_A, I_A, F_A)$

gesucht: DFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit $L(A) = L(B)$

Potenzmengenkonstruktion: NFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit

$$
\begin{aligned}
Q_B &= 2^{Q_A} \quad \text{(Einschränkung auf von } I_B \text{ erreichbare genügt)} \\
I_B &= \{I_A\} \\
F_B &= \{M \subseteq Q_A \mid F_A \cap M \neq \emptyset\} \\
\forall a \in X: \ \delta_B(a) &= \{(M, N) \mid N = \{q \mid \exists p \in M : (p, q) \in \delta_A(a)\}\}
\end{aligned}
$$

Satz

Der nach der Potenzmengenkonstruktion aus dem NFA A konstruierte NFA B ist vollständig, deterministisch und äquivalent zu A.

Beispiel Potenzmengenkonstruktion

gegeben: NFA $A = (\{a, b\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1), (1, 1)\}$ und $\delta(b) = \{(1, 1), (1, 2)\}$

Potenzmengenkonstruktion: NFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit

$$
\begin{aligned}
Q_B &= 2^{Q_A} = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}\} \\
I_B &= \{\{0\}\} \\
F_B &= \{\{2\}, \{0,2\}, \{1,2\}, \{0,1,2\}\} \\
\delta_B(a) &= \left\{ \begin{array}{l} (\emptyset,\emptyset), (\{0\},\{1\}), (\{1\},\{1\}), (\{1,2\},\{1\}), \\ (\{2\},\emptyset), (\{0,1\},\{1\}), (\{0,2\},\emptyset), (\{0,1,2\},\{1\}) \end{array} \right\} \\
\delta_B(b) &= \left\{ \begin{array}{l} (\emptyset,\emptyset), (\{0\},\emptyset), (\{1\},\{1,2\}), (\{1,2\},\{1,2\}), \\ (\{2\},\emptyset), (\{0,1\},\{1,2\}), (\{0,2\},\emptyset), (\{0,1,2\},\{1,2\}) \end{array} \right\}
\end{aligned}
$$

(rot: Einschränkung auf aus dem Startzustand erreichbare Zustände)
Einschränkung isomorph zum DFA $B' = (\{a, b\}, \{0, 1, 2, 3\}, \delta, \{0\}, \{2\})$
mit $\delta(a) = \{(0, 1), (1, 1), (2, 1), (3, 3)\}$ und
$\delta(b) = \{(0, 3), (1, 2), (2, 2), (3, 3)\}$
mit Isomorphismus $h : Q_B \to \{0, 1, 2, 3\}$ mit
$h(\{0\}) = 0$, $h(\{1\}) = 1$, $h(\{1, 2\}) = 2$, $h(\emptyset) = 3$

NFA für Spiegelung

Beispiel: DFA $A = (\{a, b\}, \{0, 1, 2, 3\}, \delta_A, \{0\}, \{2, 3\})$ mit $\delta_A(a) = \{(0, 1), (1, 3), (3, 3)\}$ und $\delta_A(b) = \{(1, 2), (2, 2)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q, \delta_A, I_A, F_A)$

gesucht: NFA B mit $L(B) = (L(A))^R$

Konstruktion: NFA $B = (X, Q, \delta_B, I_B, F_B)$ mit

1. $I_B = F_A$
2. $F_B = I_A$
3. $\forall a \in X : \delta_B(a) = \delta_A(a)^{-1} = \{(q, p) \mid (p, q) \in \delta_A(a)\}$

Fakt

Der oben definierte NFA B akzeptiert die Sprache $(L(A))^R$.

Die Menge REC(NFA) ist abgeschlossen unter Spiegelung.

NFA für Komplement NFA-akzeptierbarer Sprachen

Beispiel:
vollständiger DFA $A = (\{a, b\}, \{0, 1, 2, 3\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1), (1, 3), (2, 2), (3, 3)\}$ und
$\delta(b) = \{(0, 3), (1, 2), (2, 2), (3, 3)\}$

allgemeines Verfahren:

gegeben: vollständiger DFA $A = (X, Q, \delta, I, F)$
(falls notwendig vorher Potenzmengenkonstruktion)

gesucht: NFA B mit $L(B) = \overline{L(A)}$

Konstruktion: NFA $B = (X, Q, \delta, I, Q \setminus F)$

Satz

Der oben definierte NFA B akzeptiert die Sprache $\overline{L(A)}$.

Die Menge REC(NFA) ist abgeschlossen unter Komplementbildung.

NFA für Vereinigung NFA-akzeptierbarer Sprachen

Beispiel: NFA $A = (\{a, b\}, \{0, 1\}, \delta_A, \{0\}, \{0\})$ mit
$\delta_A(a) = \{(0, 1), (1, 0)\}$ und $\delta_A(b) = \emptyset$
NFA $B = (\{a, b\}, \{2, 3, 4\}, \delta_B, \{2\}, \{4\})$ mit
$\delta_B(a) = \{(2, 3), (4, 3)\}$ und $\delta_B(b) = \{(3, 4)\}$

allgemeines Verfahren:

gegeben: NFA $A = (X, Q_A, \delta_A, I_A, F_A)$ und
NFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit $Q_A \cap Q_B = \emptyset$
(Zustände disjunkt umbenennen, falls notwendig)

gesucht: NFA C mit $L(C) = L(A) \cup L(B)$

Konstruktion: NFA $C = (X, Q_A \cup Q_B, \delta_C, I_A \cup I_B, F_A \cup F_B)$ mit
$\forall a \in X : \delta_C(a) = \delta_A(a) \cup \delta_B(a)$

Satz

Der oben definierte NFA C akzeptiert die Sprache $L(A) \cup L(B)$.

Die Menge REC(NFA) ist abgeschlossen unter Vereinigung.

NFA für Schnitt NFA-akzeptierbarer Sprachen

WH (deMorgan):
Für alle Mengen M, N gilt $M \cap N = \overline{\overline{M} \cup \overline{N}}$

bisher: Konstruktionen für

1. NFA $A \mapsto$ vollst. DFA A' mit $L(A') = L(A)$
2. vollst. DFA $A \mapsto$ vollst. DFA A' mit $L(A') = \overline{L(A)}$
3. NFA $A, B \mapsto$ NFA C mit $L(C) = L(A) \cup L(B)$

Zu gegebenen NFA A und B lässt sich damit auch ein NFA C konstruieren mit

$$L(C) = L(A) \cap L(B) = \overline{\overline{L(A)} \cup \overline{L(B)}}$$

Die Menge REC(NFA) ist also abgeschlossen unter Schnitt.

Diese Konstruktion ist aber meist aufwendig. Es geht besser.

Was bisher geschah

- Operationen auf Sprachen:
  - Mengenoperationen $\cup, \cap, \setminus, \overline{\ }, \Delta$
  - Sprachoperationen ${}^R, \circ, {}^*$
- reguläre Ausdrücke RegExp(X):

  IA: $\{\varepsilon, \emptyset\} \cup X \subseteq \text{RegExp}(X)$

  IS: $\forall E, F \in \text{RegExp}(X) : \{E + F, EF, E^*\} \subseteq \text{RegExp}(X)$

  Die Menge aller regulären Sprachen ist abgeschlossen unter $\cup, \circ, {}^*$
- Grammatiken, Chomsky-Hierarchie

endliche Automaten: NFA $A = (X, Q, \delta, I, F)$

- vollständige, deterministische NFA
- Akzeptanz von Wörtern durch NFA
- von NFA, DFA, akzeptierte Sprache $L(A)$
- Menge REC(NFA) aller NFA-akzeptierbare Sprachen
- Zu jedem NFA existiert ein äquivalenter vollständiger NFA. (Vervollständigung)
- Zu jedem NFA existiert ein äquivalenter (vollständiger) DFA. (Potenzmengenkonstruktion)
- REC(NFA) ist abgeschlossen unter Mengenoperationen $\cup, \overline{\ }, \cap, \setminus, \Delta$

DFA für Schnitt – Produktkonstruktion

Beispiel: DFA $A = (\{a, b\}, \{0, 1\}, \delta_A, \{0\}, \{1\})$ mit
$\delta_A(a) = \{(0, 0)\}$ und $\delta_A(b) = \{(0, 1), (1, 1)\}$
DFA $B = (\{a, b\}, \{2, 3\}, \delta_B, \{2\}, \{3\})$ mit
$\delta_B(a) = \{(2, 3), (3, 3)\}$ und $\delta_B(b) = \{(3, 3)\}$
allgemeines Verfahren:

gegeben: DFA $A = (X, Q_A, \delta_A, I_A, F_A)$, DFA $B = (X, Q_B, \delta_B, I_B, F_B)$

gesucht: DFA $C = (X, Q_C, \delta_C, I_C, F_C)$ mit $L(C) = L(A) \cap L(B)$

Konstruktion: (Produktautomat)
DFA $C = (X, Q_A \times Q_B, \delta_C, I_A \times I_B, F_A \times F_B)$ mit

$$\forall a \in X : \ \delta_C(a) = \{((p, q), (r, s)) \mid (p, r) \in \delta_A(a) \wedge (q, s) \in \delta_B(a)\}$$

Satz

Der oben definierte DFA C akzeptiert die Sprache $L(A) \cap L(B)$.

Modifikation:
dieselbe Konstruktion des DFA C aus vollständigen DFA A und B

- mit akzeptierenden Zuständen $F_C = F_A \times F_B$
  akzeptiert $L(A) \cap L(B)$
- mit akzeptierenden Zuständen $F_C = (F_A \times Q_B) \cup (Q_A \times F_B)$
  akzeptiert $L(A) \cup L(B)$

NFA mit ε-Übergängen

ε-NFA: NFA ohne Einschränkung $\delta(\varepsilon) = I_Q$, stattdessen $I_Q \subseteq \delta(\varepsilon)$
(zusätzliche Kanten im Automatengraphen mit Beschriftung ε)

ε-NFA vereinfachen oft die Modellierung von Sprachen und Abläufen.

Beispiel: $A = (\{a, b, c\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 0)\}$, $\delta(b) = \{(1, 1)\}$, $\delta(c) = \{(2, 2)\}$, $\delta(\varepsilon) = \{(0, 1), (1, 2)\}$

Akzeptanz von Wörtern durch ε-NFA analog zu NFA:
ε-NFA $A = (X, Q, \delta, I, F)$ akzeptiert ein Wort $w \in X^*$ genau dann, wenn
$\delta^*(w) \cap (I \times F) \neq \emptyset$.

$$\delta^*(w) = \delta^*(\varepsilon) \circ \delta(w_1) \circ \delta^*(\varepsilon) \circ \cdots \circ \delta^*(\varepsilon) \circ \delta(w_n) \circ \delta^*(\varepsilon)$$

akzeptierender Pfad für w in ε-NFA A:
$q_0 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_i \xrightarrow{w_1} q_{i+1} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_k \xrightarrow{w_{|w|}} q_{k+1} \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_n$

Der ε-NFA A im Beispiel oben akzeptiert die Sprache $L(A) = a^* b^* c^*$.

Fakt

Zu jedem NFA A existiert ein äquivalenter ε-NFA B mit genau einem Startzustand und genau einem akzeptierenden Zustand.

Eliminierung der ε-Übergänge aus ε-NFA

$\delta^*(\varepsilon)$ ist der transitive Abschluss der Relation $\delta(\varepsilon)$
ε-Hülle eines Zustandes $q \in Q$: $H_\varepsilon(q) = \{p \mid (q,p) \in \delta^*(\varepsilon)\}$

allgemeines Verfahren:

gegeben: ε-NFA $A = (X, Q, \delta, I, F)$ mit $\delta(\varepsilon) \supseteq I_Q$

gesucht: NFA B ohne ε-Übergänge mit $L(B) = L(A)$

Konstruktion (ε-Eliminierung): $B = (X, Q, \delta', I', F)$ mit

$$I' = \bigcup_{q \in I} H_\varepsilon(q)$$

$$\forall a \in X: \ \delta'(a) = \delta(a) \cup \left\{(p, q') \mid (p,q) \in \delta(a) \wedge q' \in H_\varepsilon(q)\right\}$$

Satz

Der durch ε-Eliminierung aus dem ε-NFA A konstruierte NFA B enthält keine ε-Übergänge und ist äquivalent zu A.

Was bisher geschah

Chomsky-Hierarchie für Sprachen:

- $\mathcal{L}_0$ Menge aller durch (beliebige) Grammatiken beschriebenen Sprachen
- $\mathcal{L}_1$ Menge aller monotonen (Kontextsensitive) Sprachen
- $\mathcal{L}_2$ Menge aller kontextfreien Sprachen
- $\mathcal{L}_3 = \mathrm{REG}$ Menge aller regulären Sprachen

- $\mathrm{REC(NFA)} = \mathrm{REC(DFA)} = \mathrm{REC}(\varepsilon - \mathrm{NFA})$
- REC(NFA) ist abgeschlossen unter $\cup, \cap, \overline{\ }, \circ, \ ^*, \ ^R$
- REC(NFA) = Menge aller regulären (durch reguläre Ausdrücke darstellbare) Sprachen
  (Konstruktion RegEx $\rightarrow$ NFA)

DFA-Minimierung

Ein vollständiger DFA A heißt genau dann minimal für die Sprache $L(A)$, wenn kein vollständiger DFA B mit $L(A) = L(B)$ und weniger Zuständen als A existiert.

Beispiel: Ist der folgende vollständige DFA minimal?
$A = (\{a, b\}, \{0, 1, 2, 3, 4\}, \delta, \{0\}, \{4\})$ mit
$\delta(a) = \{(0, 1), (1, 4), (2, 3), (3, 4), (4, 4)\}$ und
$\delta(b) = \{(0, 2), (1, 2), (2, 2), (3, 0), (4, 4)\}$

Idee (für vollständigen DFA $A = (X, Q, \delta, I, F)$):

- für jedes Paar (p, q) von Zuständen in Q:
  Suche nach einem (kürzesten) unterscheidenden Wort
- Zustände sind äquivalent, wenn kein unterscheidendes Wort existiert
- Zerlegung von Q in Äquivalenzklassen von Zuständen
- diese Klassen sind die Zustände des minimalen DFA.

Äquivalente Zustände in DFA

Ziel: Konstruktion eines minimalen DFA B mit $L(B) = L(A)$
zu gegebenem DFA A

Beispiel: $A' = (\{a, b\}, \{0, 1, 2, 3\}, \delta, \{0\}, \{3\})$ mit
$\delta(a) = \{(0, 1)\}$ und $\delta(b) = \{(0, 2), (1, 3), (2, 3)\}$
(vollständig ?)

Definition:
Jedes Paar aus DFA $A = (X, Q, \delta, I, F)$ und Zustand $q \in Q$
definiert den DFA $A_q = (X, Q, \delta, \{q\}, F)$
und damit die Sprache $L(A_p) \subseteq X^*$

Zustände $p, q \in Q$ heißen genau dann
äquivalent in A ($p \sim_A q$), wenn $L(A_p) = L(A_q)$ gilt.

Beispiel (oben):
$1 \sim_A 2$, weil $L(A_1) = L(A_2) = \{b\}$

Ein Wort $w \in X^*$ unterscheidet die Zustände $p \in Q$ und $q \in Q$
(beweist $p \not\sim_A q$) gdw. $w \in L(A_p) \Delta L(A_q)$

Beispiel (oben): $1 \not\sim_A 3$, weil $\varepsilon \in L(A_3) \setminus L(A_1)$

Algorithmus zur Bestimmung äquivalenter Zustände

Fakt

Falls das Wort $w \in X^*$ die Zustände $p', q' \in Q$ im DFA $A = (X, Q, \delta, I, F)$ unterscheidet und für ein $a \in X$ gilt $(p, p') \in \delta(a)$ und $(q, q') \in \delta(a)$, dann unterscheidet das Wort $aw \in X^*$ die Zustände $p, q \in Q$ in A.

Induktive Berechnung einer Folge T_i von Mengen von unterscheidbaren Zustandspaaren (als Zweiermengen):

$$
\begin{aligned}
T_0 &= \{\{p, q\} \mid p \in F, q \notin F\} \\
T_{i+1} &= T_i \cup \left\{ \{p, q\} \mid \exists a \in X \; \exists p', q' \in Q : \begin{pmatrix} \{p', q'\} \in T_i \\ \wedge (p, p') \in \delta(a) \\ \wedge (q, q') \in \delta(a) \end{pmatrix} \right\}
\end{aligned}
$$

(T_i: Menge aller durch ein w mit $|w| \leq i$ unterscheidbaren Paare)
Für jeden vollständigen DFA A existiert ein $n \in \mathbb{N}$ mit $T_n = T_{n+1}$.
Für jedes Paar $\{p, q\} \notin T_n$ sind p und q äquivalent in A.

Alternative Darstellung: Folge von Äquivalenzrelationen $\sim_i \subseteq Q^2$ mit
$p \sim_i q$ gdw. $\forall w \in X^* : |w| \leq i \rightarrow (w \in A_p \leftrightarrow w \in A_q)$
(auch als Folge der Zerlegungen in $\sim_i$-Äquivalenzklassen üblich)

Minimalautomat

gegeben: vollständiger DFA $A = (X, Q_A, \delta_A, I_A, F_A)$
Beispiel: vollst. DFA $A = (\{a, b\}, \{0, 1, 2, 3, 4\}, \delta, \{0\}, \{4\})$ mit
$\delta(a) = \{(0, 1), (1, 4), (2, 3), (3, 4), (4, 4)\}$ und
$\delta(b) = \{(0, 2), (1, 2), (2, 2), (3, 0), (4, 4)\}$

Minimalautomat für $L(A)$:
DFA $B = (X, Q_B, \delta_B, I_B, F_B)$ mit

$$\begin{aligned} Q_B &= \{[q]_A \mid q \in Q\} \quad \text{Äquivalenzklassen in } A \\ \forall a \in X : \delta_B(a) &= \{([p]_A, [q_A]) \mid (p, q) \in \delta_A(a)\} \\ I_B &= \{[q]_A \mid q \in I_A\} \\ F_B &= \{[q]_A \mid q \in F_A\} \end{aligned}$$

Satz

Zu jeder NFA-akzeptierbaren Sprache $L \subseteq X^*$ existiert ein (bis auf Isomorphie) eindeutiger vollständiger DFA A mit $L = L(A)$ und einer minimalen Anzahl von Zuständen (Minimalautomat für L).

Entscheidung der Äquivalenz von NFA

Wiederholung: NFA A und B äquivalent gdw. $L(A) = L(B)$

Eingabe:	NFA A, B	
Ausgabe:	1 (ja),	falls A und B äquivalent
	0 (nein),	sonst

Algorithmus:

1. Konstruktion vollständiger DFA A' und B' mit $L(A') = L(A)$ und $L(B') = L(B)$ (Potenzmengenkonstruktion),
2. Konstruktion der Minimalautomaten A'' zu A' und B'' zu B'
3. Test, ob A'' und B'' isomorph sind (Isomorphietest für Graphen).

Fakt

Die NFA A und B sind genau dann äquivalent, wenn die Automaten A'' und B'' isomorph sind.

Entscheidung der Äquivalenz regulärer Ausdrücke

Eingabe: Reguläre Ausdrücke E, F

Ausgabe: 1 (ja), falls E und F äquivalent ($L(E) = L(F)$)
0 (nein), sonst

Algorithmus:

1. Konstruktion der NFA A_E und A_F
   mit $L(A_E) = L(E)$ und $L(A_F) = L(F)$,
2. Konstruktion der DFA A'_E und A'_F mit
   $L(A'_E) = L(E)$ und $L(A'_F) = L(F)$
   (Potenzmengenkonstruktion),
3. Konstruktion der Minimalautomaten
   A''_E zu A'_E und A''_F zu A'_F
4. Test, ob A''_E und A''_F isomorph sind
   (Isomorphietest für Graphen)

Fakt

Die regulären Ausdrücke E und F sind genau dann äquivalent, wenn die Automaten A''_E und A''_F isomorph sind.

Beispiel: $(ab)^+a$ und $a(ba)^*ba$

Entscheidung der Äquivalenz von NFA

Wiederholung: NFA A und B äquivalent gdw. $L(A) = L(B)$

Eingabe:	NFA A, B	
Ausgabe:	1 (ja),	falls A und B äquivalent
	0 (nein),	sonst

Algorithmus:

1. Konstruktion vollständiger DFA A' und B' mit $L(A') = L(A)$ und $L(B') = L(B)$ (Potenzmengenkonstruktion),
2. Konstruktion der Minimalautomaten A'' zu A' und B'' zu B'
3. Test, ob A'' und B'' isomorph sind (Isomorphietest für Graphen).

Fakt

Die NFA A und B sind genau dann äquivalent, wenn die Automaten A'' und B'' isomorph sind.

NFA und reguläre Grammatiken

Beispiel: NFA $A = (\{a, b\}, \{0, 1, 2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0, 1)\}$ und $\delta(b) = \{(1, 0), (1, 2)\}$

reguläre Grammatik $G = (\{S, B\}, \{a, b\}, P, S)$ mit
$P = \{S \to aB, B \to bS, B \to b\}$

definieren beide die Sprache
$L(A) = L(G) = L((ab)^+) = \{(ab)^n \mid n \in \mathbb{N} \setminus \{0\}\}$

Satz

REC(NFA) $= \mathcal{L}_3$

dieselbe Aussage ausführlicher:
Eine Sprache $L \subseteq X^*$ ist genau dann NFA-akzeptierbar, wenn $L \setminus \{\varepsilon\}$ von einer regulären Grammatik (Chomsky-Typ 3) erzeugt wird.

Konstruktion: reguläre Grammatik $\rightarrow$ NFA

gegeben: reguläre Grammatik $G = (N, X, P, S)$ (Chomsky 3)
Konstruktion: NFA $A = (X, Q, \delta, I, F)$ mit

$$\begin{aligned} Q &= N \cup \{f\} \quad \text{mit } f \notin N \\ I &= \{S\} \\ F &= \{f\} \\ \text{für jedes } a \in X\text{: } \delta(a) &= \{(A, B) \mid (A \rightarrow aB) \in P\} \\ &\quad \cup \{(A, f) \mid (A \rightarrow a) \in P\} \end{aligned}$$

Beispiel: Konstruktion ergibt für die reguläre Grammatik
$G = (\{S, B\}, \{0, 1\}, P, S)$ mit
$P = \{S \rightarrow 0B, B \rightarrow 0B, B \rightarrow 1S, B \rightarrow 0\}$
den NFA $A = (\{0, 1\}, \{S, B, f\}, \delta, \{S\}, \{f\})$ mit
$\delta(0) = \{(S, B), (B, B), (B, f)\}$ und $\delta(1) = \{(B, S)\}$

Fakt (REC(NFA) $\subseteq \mathcal{L}_3$): Für den wie oben zur Grammatik G konstruierten NFA A gilt $L(G) = L(A)$.

Konstruktion: NFA $\rightarrow$ reguläre Grammatik

gegeben: NFA $A = (X, Q, \delta, I, F)$
Konstruktion: reguläre Grammatik $G = (N, X, P, S)$ mit

$$
\begin{aligned}
N &= Q \,\dot\cup\, \{S\} \quad (\text{mit } S \notin Q) \\
P &= \bigcup_{a \in X} \{p \to aq \mid (p,q) \in \delta(a)\} \\
&\quad \cup \bigcup_{a \in X} \{S \to aq \mid \exists s \in I : (s,q) \in \delta(a)\} \\
&\quad \cup \bigcup_{a \in X} \{p \to a \mid \exists f \in F : (p,f) \in \delta(a)\} \\
&\quad \cup \bigcup_{a \in X} \{S \to a \mid \exists s \in I \; \exists f \in F : (s,f) \in \delta(a)\}
\end{aligned}
$$

Beispiel: $A = (\{a,b\}, \{0,1,2\}, \delta, \{0\}, \{2\})$ mit
$\delta(a) = \{(0,1),(1,2)\}$ und $\delta(b) = \{(1,1),(2,1)\}$
konstruierte Grammatik $G = (\{S,0,1,2\}, \{a,b\}, P, S)$ mit
$P = \{0 \to a1, 1 \to b1, 1 \to a2, 2 \to b1, S \to a1, 1 \to a\}$

Fakt ($\mathcal{L}_3 \subseteq$ REC(NFA)): Für die wie oben aus dem NFA A konstruierte Grammatik G gilt $L(G) = L(A) \setminus \{\varepsilon\}$.

Ergebnisse über reguläre Sprachen

Für jede Sprache $L \subseteq X^*$ sind die folgenden Aussagen äquivalent:

- $L = L(E)$ für einen regulären Ausdruck E.
- L hat den Chomsky-Typ 3.
- Es existiert eine reguläre Grammatik G mit $L \setminus \{\varepsilon\} = L(G)$.
- Es existiert eine reguläre Grammatik G mit ε-Regel und $L = L(G)$.
- L ist NFA-akzeptierbar.
- Es existiert ein vollständiger DFA A mit $L = L(A)$.

Die Menge aller Sprachen vom Chomsky-Typ 3 ist abgeschlossen unter

- Mengenoperationen: $\cup, \cap, \overline{\ }, \setminus, \Delta$
- Sprachoperationen: $\circ$, *, ^{R}

Algorithmische Lösungen für reguläre Sprachen

alle Sprachen in Eingaben endlich beschrieben, z.B. als NFA

Wortproblem Eingabe: (L, w),
Ausgabe: ja, falls $w \in L$, sonst nein
(Suche nach mit w markiertem Pfad im Automatengraphen)

Leerheit Eingabe: L, Ausgabe: ja, falls $L = \emptyset$, sonst nein
(Erreichbarkeit akzeptierender Zustände in NFA für L)

Vollständigkeit Eingabe: L, Ausgabe: ja, falls $L = X^*$, sonst nein
(Erreichbarkeit im Automaten für $\overline{L}$)

(Un-)Endlichkeit Eingabe: L, Ausgabe: ja, falls L (un-)endlich, sonst nein
(Suche nach einem akzeptierenden Weg mit Schleife)

Inklusion Eingabe: L_1, L_2,
Ausgabe: ja, falls $L_1 \subseteq L_2$, sonst nein
(Test $L_1 \cap \overline{L_2} = \emptyset$)

Gleichheit Eingabe: L_1, L_2,
Ausgabe: ja, falls $L_1 = L_2$, sonst nein
(isomorphe Minimalautomaten oder Test $L_1 \Delta L_2 \stackrel{?}{=} \emptyset$)

Disjunktheit Eingabe: L_1, L_2,
Ausgabe: ja, falls $L_1 \cap L_2 = \emptyset$, sonst nein

Anwendung von NFA und regulären Sprachen

- Modellierung vom Verhalten von Zustandsübergangssystemen, z.B.
  - reale Geräte und Anlagen
  - Softwaresysteme

  Überprüfung, ob Zustandsübergangssystem bestimmte Anforderungen erfüllt
- Beschreibung der Syntax von Programmiersprachen z.B.
  - endlicher Mengen (z.B. Mengen aller Schlüsselwörter)
  - Folgen von Zeichen und Zeichenketten
  - Zahlendarstellungen
- lexikalische Analyse im Compiler
- Suche nach Zeichenketten in Texten (string matching)
- Textverarbeitung, z.B. Wortvervollständigung

Was bisher geschah

Sprache L ist vom Chomsky-Typ i gdw.
eine Grammatik $G = (N, T, P, S)$ mit $L(G) = L \setminus \{\varepsilon\}$ existiert.
Chomsky-Hierarchie: G ist vom Chomsky-Typ

0 immer,

1 , $\forall(l \to r) \in P : |l| \leq |r|$ (monoton, kontextsensitiv)

2 , $\forall(l \to r) \in P : |l| \leq |r| \wedge l \in N$ (kontextfrei)

3 , $\forall(l \to r) \in P : |l| \leq |r| \wedge l \in N \wedge r \in (T \cup (T \circ N))$ (regulär)

reguläre Sprachen ($\mathcal{L}_3 = \mathrm{REG} = \mathrm{REC(NFA)} = \mathrm{REC(DFA)}$):

- endliche Repräsentation durch
  - Darstellung als reguläre Ausdrücke
  - Erzeugung durch reguläre Grammatiken,
  - Akzeptanz durch NFA (DFA, ε-NFA, ...),
- abgeschlossen unter $\cup, \cap, \overline{\ }, \circ, {}^*, {}^R$

Fragen für kontextfreie Sprachen

Wiederholung:
Eine Sprache $L \subseteq X^*$ heißt genau dann kontextfrei, wenn für eine kontextfreie Grammatik (Chomsky-Typ 2) G gilt $L \setminus \{\varepsilon\} = L(G)$.

CF ($= \mathcal{L}_2$): Menge aller kontextfreien Sprachen

- Unter welchen der Operationen $\cup, \cap, \overline{\ }, \circ, {}^*, {}^R$ ist CF abgeschlossen?
- Gibt es ein passendes (deterministisches, eindeutiges) Maschinenmodell für kontextfreie Sprachen?
- Existieren algorithmische Lösungsverfahren für folgende Probleme für kontextfreie Sprachen:
  - Wortproblem
  - Leerheit, Vollständigkeit, Endlichkeit
  - Inklusion, Gleichheit, Disjunktheit

Kontextfreie Grammatiken – Wiederholung

Grammatik $G = (N, T, P, S)$ vom Chomsky-Typ 2

Alle Regeln in P haben die Form $l \to r$ mit
$l \in N$ und $r \in (N \cup T)^+$

w in G ableitbar gdw. $S \to_P^* w$

Grammatik $G = (N, T, P, S)$ erzeugt die Sprache
$L(G) = \{w \in T^* \mid S \to_P^* w\}$

Beispiel: Grammatik $G = (\{S\}, \{a, b\}, \{S \to aSb|ab\}, S)$

- ist vom Chomsky-Typ 2
- Ableitung für $aaabbb$ in G: $S \to aSb \to aaSbb \to aaabbb$
- G erzeugt die Sprache $L(G) = \{a^n b^n \mid n > 0\}$
  (vom Chomsky-Typ 2)

Für $L = \{a^n b^n \mid n > 0\}$ gilt $L \in \text{CF} \setminus \text{REG}$, d.h.

- L ist kontextfrei.
- L ist nicht regulär. (Wie kann man das zeigen?)

WH: Schubfachschluss-Prinzip

Schubfachschluss-Prinzip (Pigeonhole principle):
Für $|O| > |S|$ existiert keine injektive Funktion $f : O \to S$.

$$\forall O \forall S \forall f : ((|O| > |S| \wedge f : O \to S) \to \exists x \exists y : (x \neq y \wedge f(x) = f(y)))$$

Anschaulich:
Verteilt man mehr als n Objekte in n Schubfächer, dann gibt es (wenigstens) ein Schubfach, welches (wenigstens) zwei Objekte enthält.
Beispiele:

- Von 13 Personen haben mindestens zwei im selben Monat Geburtstag.
- Wieviele Karten aus einem Skatblatt muss man ziehen, damit zwei derselben Farbe dabei sind?
- vier verschiedenfarbige Paare Socken im Dunklen:
  Wieviele Socken muss man (blind) nehmen, damit ein vollständiges Paar dabei ist?
- Bei jeder Wanderung mit $> n$ Pausen in einem Gebiet mit n Rastplätzen waren wenigsten zwei Pausen am selben Platz.

Nichtreguläre Sprachen

Fakt

Die Sprache $L = \{a^n b^n \mid n > 0\}$ *ist nicht NFA-akzeptierbar.*

WH Definition REC(NFA): $L \in \text{REC(NFA)}$ gdw. $\exists$ NFA $A : (L(A) = L)$

Vorbereitung für Beweis (indirekt):

$$\forall L \subseteq X^* : L \notin \text{REC(NFA)} \quad \text{gdw.} \quad \neg\exists A : (L(A) = L) \quad \text{gdw.} \; \forall A : \neg(L(A) = L)$$

$$\text{gdw.} \quad \forall A : (\underbrace{(L(A) = L)}_{\text{Annahme } P(A)} \rightarrow \underbrace{\neg(L(A) = L)}_{\text{Folgerung } \neg P(A)})$$

Beweis: Annahme: für NFA $A = (X, Q, \delta, I, F)$ mit $|Q| = k$ gilt $L(A) = L$

Wegen $a^k b^k \in L$ existiert ein akzeptierender Pfad für $a^k b^k$ in A:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} \cdots \xrightarrow{a} q_k \xrightarrow{b} q_{k+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2k} \quad \text{mit } q_0 \in I, q_{2k} \in F$$

SFS: O: Positionen im a^k-Teil $|\{0, \ldots, k\}| > k$, S: Zustände $|Q| = k$

Nach SFS existieren $i, j \in \{0, \ldots, k\}$ mit $i < j$ und $q_i = q_j$.

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} \cdots \xrightarrow{a} q_i = q_j \xrightarrow{a} q_{j+1} \xrightarrow{a} \cdots \xrightarrow{a} q_k \xrightarrow{b} q_{k+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2k}$$

ist also akzeptierender Pfad für $a^{k-(j-i)} b^k$ in A, d.h $a^{k-(j-i)} b^k \in L(A)$

Aus $a^{k-(j-i)} b^k \notin L$ folgt $\neg(L(A) = L)$ im Widerspruch zur Annahme.

Also ist die Annahme falsch, d.h. es existiert kein NFA A mit $L = L(A)$.

Weitere nichtreguläre Sprachen

hilfreiches Prinzip:
Da REC(NFA) unter $\cap$ abgeschlossen ist, d.h. $\forall L, L', L'' \subseteq X^*$

$$(L \cap L' = L'' \wedge L \in \text{REC(NFA)} \wedge L' \in \text{REC(NFA)}) \rightarrow L'' \in \text{REC(NFA)}$$

gilt für alle Sprachen L, L', L'': (ÜA)

$$(L \cap L' = L'' \wedge L' \in \text{REC(NFA)} \wedge L'' \notin \text{REC(NFA)}) \rightarrow L \notin \text{REC(NFA)}$$

- $L = \{w \in \{a,b\}^* \mid |w|_a = |w|_b > 0\} \notin \text{REC(NFA)}$
  (wegen $L \cap \underbrace{L(a^*b^*)}_{\in \text{REC(NFA)}} = \{a^n b^n \mid n > 0\} \notin \text{REC(NFA)}$)
- $L = \{w \in \{a,b\}^* \mid 2|w|_a = |w|_b \in \mathbb{N}\} \notin \text{REC(NFA)}$
  (wegen $L \cap L(a^*b^*) = \{a^n b^{2n} \mid n > 0\} \notin \text{REC(NFA)}$
  nach Schubfachschluss).
- $L = \{w \in \{a,b\}^* \mid w = w^R\} \notin \text{REC(NFA)}$ (Palindrome)
  (wegen $L \cap L(a^*ba^*) = \{a^n ba^n \mid n \in \mathbb{N}\} \notin \text{REC(NFA)}$)
- $L = \{a^{(n^2)} \mid n \in \mathbb{N}\} \notin \text{REC(NFA)}$
  (nach Schubfachschluss, Wortlängen)

Ableitungsbäume für kontextfreie Grammatiken

Beispiel: Grammatik $G = (N, T, P, E)$ mit $N = \{E, F\}$,
$T = \{a, b, c, (,), +, *\}$,

$$P = \{E \to (E + E), E \to F * F, F \to E, E \to a, E \to b, E \to c\}$$

Ableitung für $w = (c * a + (b + a * a))$

allgemein: Grammatik $G = (N, T, P, S)$

Ableitung $S \to_P w_1 \to_P w_2 \to_P \cdots \to_P w_n = w$ für w in G

Ableitungsbaum zu einem Wort w in der kontextfreien Grammatik G (induktive Definition):

- Wurzel mit Markierung S
- für jeder Anwendung einer Regel $A \to_P r_1 \cdots r_n$: Verzweigung bei Symbol A in n Kinder mit Markierungen $r_1 \ldots, r_n$
- Markierung der Blätter (von links nach rechts): Wort w

Ein Ableitungsbaum repräsentiert i.A. mehrere Ableitungen.

Rechts- und Linksableitungen

Eine Ableitung für w in G heißt

Linksableitung , falls in jedem Schritt das am weitesten links stehende Nichtterminal

Rechtsableitung , falls in jedem Schritt das am weitesten rechts stehende Nichtterminal

ersetzt wird.

Zu jedem Ableitungsbaum für w in G existieren

- genau eine Linksableitung für w in G und
- genau eine Rechtsableitung für w in G.

Abschlusseigenschaften von CF

Satz

Sind L_1 und L_2 kontextfreie Sprachen, dann sind auch kontextfrei:

- $L_1 \cup L_2$
- $L_1 \circ L_2$
- L_1^*
- L_1^R

Konstruktionen für $L_i = L(G_i)$ mit $G_i = (N_i, T_i, P_i, S_i)$ für $i \in \{1, 2\}$,
wobei $N_1 \cap N_2 = \emptyset$ (falls nötig, umbenennen)
neues Nichtterminal $S' \notin N_1 \cup N_2$

$$
\begin{aligned}
L_1 \cup L_2 &= L(G_\cup) \text{ mit } G_\cup = (N_1 \cup N_2 \cup \{S'\}, T_1 \cup T_2, P_\cup, S'), \\
&\quad \text{wobei } P_\cup = P_1 \cup P_2 \cup \{S' \to S_1, S' \to S_2\} \\
L_1 \circ L_2 &= L(G_\circ) \text{ mit } G_\circ = (N_1 \cup N_2 \cup \{S'\}, T_1 \cup T_2, P_\circ, S'), \\
&\quad \text{wobei } P_\circ = P_1 \cup P_2 \cup \{S' \to S_1 S_2\} \\
L_1^* &= L(G_*) \text{ mit } G_* = (N_1 \cup \{S'\}, T_1, P_*, S') \text{ , wobei} \\
&\quad P_* = P_1 \cup \{S' \to \varepsilon, S' \to S_1 S'\} \\
L_1^R &\text{ (war ÜA 3.4)}
\end{aligned}
$$

noch offen: CF abgeschlossen unter $^-, \cap$?

Mehrdeutige kontextfreie Grammatiken

Kontextfreie Grammatik G heißt

eindeutig , falls für jedes Wort $w \in L(G)$
genau ein Ableitungsbaum in G existiert.

mehrdeutig , sonst.

Beispiel: Ableitungsbäume für $w = aaa$ in der Grammatik
$G = (N, T, P, E)$ mit $N = \{S\}$, $T = \{a\}$, $P = \{S \to SS, S \to a\}$

Eindeutigkeit von Grammatiken ist wichtig beim Entwurf von Programmiersprachen (Syntax)

„dangling else“-Problem:

```
<Anw> ::= if <Ausdr> then <Anw>
<Anw> ::= if <Ausdr> then <Anw> else <Anw>
```

Was bedeutet:

```
if C1 then if C2 then A else B
```

(Anwendungen im Modul Prinzipien von Programmiersprachen)

Inhärent mehrdeutige Sprachen

Sprachen L, für die keine eindeutige Grammatik G mit $L = L(G)$ existieren, heißen inhärent mehrdeutig.

Fakt

Die kontextfreie Sprache
$L = \{a^l b^m c^n \mid l, m, n > 0 \wedge (l = m \vee m = n)\}$
ist inhärent mehrdeutig.

Was bisher geschah

reguläre Sprachen (Chomsky-Typ 3):

- endliche Repräsentation durch
  - reguläre Grammatiken
  - NFA (DFA, ε-NFA, ...)
  - reguläre Ausdrücke
- abgeschlossen unter $\cup, \cap, \overline{\ }, \circ, {}^*, {}^R$
- nicht-reguläre Sprachen (mit Nachweis),
  z.B. $\{a^n b^n \mid n \in \mathbb{N}\}$, $\{a^n b a^n \mid n \in \mathbb{N}\}$, $\{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$
- algorithmische Entscheidbarkeit des Wortproblems ($\mathcal{O}(|w|)$),
  sowie Leerheit, Endlichkeit, Gleichheit usw. regulärer Sprachen

kontextfreie Sprachen (Chomsky-Typ 2):

- repräsentiert durch kontextfreie Grammatiken
- CF abgeschlossen unter $\cup, \circ, {}^*, {}^R$
  noch offen: CF auch unter $\cap, \overline{\ }$ abgeschlossen ?

Dyck-Sprachen

Klammerpaar, z.B. (und)
Dyck-Sprache: Menge aller korrekt geklammerten Ausdrücke, d.h.
$L(G)$ für die Grammatik $G = (\{S\}, \{(,)\}, \{S \to \varepsilon | (S) | SS\}, S)$
Beispiele:

- $()(()()) \in L(G)$, $\varepsilon \in L(G)$
- $()) \notin L(G)$, $(()(\notin L(G)$, $())(\notin L(G)$

Achtung:

- G hat Chomsky-Typ 0
- $G' = (\{S\}, \{(,)\}, \{S \to () | (S) | SS\}, S)$ hat Chomsky-Typ 2 und es gilt $L(G') = L(G) \setminus \{\varepsilon\}$
- Dyck-Sprache $L(G)$ hat also Chomsky-Typ 2

praktische Vereinbarung (Zulassung der ε-Sonderregel):
Grammatik $G = (N, T, P, S)$ hat auch dann Chomsky-Typ i, wenn sie

- nur Regeln der für Chomsky-Typ i erlaubten Form
- und evtl. die Regel $S \to \varepsilon$ enthält (nur für Startsymbol S)

Allgemeine Dyck-Sprachen

Menge aller korrekt geklammerten Ausdrücke
mit n Paaren von Klammern: $(_i,)_i$ für $i \in \{1, \ldots, n\}$
erzeugt durch Grammatik

$$
\begin{aligned}
G &= (\{S\}, \{(_i,)_i \mid i \in \{1, \ldots, n\}\}, P, S) \quad \text{mit} \\
P &= \left\{ \begin{array}{lcl} S & \to & \varepsilon \\ S & \to & SS \end{array} \right\} \cup \{S \to (_i S)_i \mid i \in \{1, \ldots, n\}\}
\end{aligned}
$$

Symbole müssen nicht notwendig Klammern sein, z.B.
$aacdacababdbbcabdb \in$ Dyck-Sprache mit
a statt $(_1$, b statt $)_1$, c statt $(_2$ und d statt $)_2$

Beispiel HTML

mehrere Paare öffnender und schließender Klammern (Tags)
<html> </html> , <head> </head> , <title> </title> , ...

```
<html>
  <head>
    <title>
       Automaten und formale Sprachen
    </title>
  </head>
  <body>
    <h1>
       Automaten und formale Sprachen
    </h1>
    ...
  </body>
</html>
```

Erreichbare und erzeugende Nichtterminale

kontextfreie Grammatik $G = (N, T, P, S)$

Nichtterminal $A \in N$ heißt

erreichbar aus B für $B \in N$, falls

$$\exists u, v \in (N \cup T)^* : B \to_P^* uAv$$

erreichbar in G falls A aus Startsymbol S erreichbar ist

erzeugend falls $\exists w \in T^* : A \to_P^* w$

Beispiel:
Grammatik $G = (N, T, P, S)$ mit $N = \{S, A, B, C, D, E\}$,
$T = \{a, b, c\}$,

$$P = \{S \to AB, A \to C, A \to a, D \to ScA, B \to b, C \to BbC\}$$

erreichbar aus S sind S, A, B, C

erzeugend sind S, A, B, D

Eliminierung nutzloser Nichtterminale

gegeben: kontextfreie Grammatik $G = (N, T, P, S)$

induktive Bestimmung aller erzeugenden Nichtterminale aus N:

$$\begin{aligned} M_0 &= \{A \mid (A \to w) \in P \wedge w \in T^*\} \\ M_{i+1} &= M_i \cup \{A \mid (A \to w) \in P \wedge w \in (T \cup M_i)^*\} \end{aligned}$$

Es gibt ein $n \in \mathbb{N} : M_n = M_{n+1}$, M_n enthält alle erzeugenden NT

Eliminierung aller nicht-erzeugenden NT:
Löschen aller Regeln, die nicht-erzeugende NT enthalten

induktive Bestimmung aller aus S erreichbaren Nichtterminale:

$$\begin{aligned} M'_0 &= \{S\} \\ M'_{i+1} &= M'_i \cup \{A \mid (B \to uAv) \in P \wedge B \in M'_i \wedge u, v \in (N \cup T)^*\} \end{aligned}$$

Es gibt ein $n \in \mathbb{N} : M'_n = M'_{n+1}$, M'_n enthält alle erreichbaren NT

Eliminierung aller nicht-erreichbaren NT:
Löschen aller Regeln, die nicht-erreichbare NT enthalten

Reduzierte kontextfreie Grammatiken

reduzierte Grammatik enthält nur erreichbare und erzeugende Nichtterminale

Fakt

Zu jeder kontextfreien Grammatik existiert eine äquivalente reduzierte Grammatik.

Erzeugung einer reduzierten Grammatik aus beliebiger kontextfreier Grammatik G:

nacheinander in dieser Reihenfolge:

1. G' entsteht durch Eliminierung aller nicht-erzeugenden Nichtterminale in G,
2. G'' entsteht durch Eliminierung aller nicht-erreichbaren Nichtterminale in G'

Satz: G'' ist reduziert und äquivalent zu G.

Kettenregeln

Kettenregel Regel $l \to r$ mit $l \in N$ und $r \in N$

Beispiel: $G = (\{A, B, S\}, \{0, 1\}, P, S)$ mit
$P = \{S \to 0S0, S \to 00, S \to A, A \to 1A, A \to B, B \to 1\}$

Eliminierung der Kettenregeln aus $G = (N, T, P, S)$:

- induktive Bestimmung aller Ketten-Paare

$$\begin{aligned} M_0 &= \{(A, A) \mid A \in N\} \\ M_{i+1} &= M_i \cup \{(A, C) \mid (A, B) \in M_i \wedge (B \to C) \in P\} \end{aligned}$$

  Es existiert ein $n \in \mathbb{N}$ mit $M_n = M_{n+1}$.
- $P' = \{A \to w \mid (A, B) \in M_n \wedge (B \to w) \in P \wedge w \notin N\}$
  - Hinzufügen aller Nicht-Kettenregeln $A \to w$, für welche P eine Regel $B \to w$ enthält und $(A, B) \in M_n$ (Ketten-Paar)
  - Löschen aller Kettenregeln

Satz: Die so erzeugte Grammatik G' ist äquivalent zu G.

Chomsky-Normalform

Kontextfreie Grammatiken $G = (N, T, P, S)$ mit
$\forall (l \to r) \in P : r \in (T \cup N^2)$ heißen in Chomsky-Normalform.
(Ableitungsbäume von Grammatiken in Chomsky-NF sind binäre Bäume.)

Satz

Zu jeder kontextfreien Grammatik G mit $\varepsilon \notin L(G)$ existiert eine äquivalente Grammatik in Chomsky-Normalform.

Beweis durch Konstruktion:

1. ε-Regeln und Kettenregeln eliminieren
   (evtl. zuvor Grammatik reduzieren, ε-Eliminierung war ÜA 3.3)
2. Hinzufügen neuer Nichtterminale $\{X_c \mid c \in T\}$ zu N,
   Hinzufügen aller Regeln $\{X_c \to c \mid c \in T\}$ zu P, Ersetzung jedes Terminals c durch X_c in allen rechten Regelseiten r mit $|r| > 1$
   (resultierende Grammatik hat nur Regeln $l \to r$ mit $r \in (T \cup N^+)$)
3. Ersetzung jeder Regel $A \to B_1 \dots B_n$ in P' mit $n > 2$ durch
   $n - 1$ Regeln: $A \to B_1 C_2, \dots, C_i \to B_i C_{i+1}, \dots, C_{n-1} \to B_{n-1} B_n$
   mit neuen Nichtterminalen C_i
   (resultierende Grammatik hat nur Regeln $l \to r$ mit $r \in (T \cup N^2)$)

Wortproblem für kontextfreie Sprachen

Eingabe: kontextfreie Grammatik $G = (N, T, P, S)$
(in Chomsky-Normalform)
Wort $w = w_1 \dots w_n \in T^*$
Ausgabe: ja, falls $w \in L(G)$, sonst nein

ist algorithmisch entscheidbar (Warum? Laufzeit?)

Beispiel: $G = (\{S, A, B\}, \{a, b, c\}, P, S)$ mit

$P = \{S \to SA, S \to a, A \to BS, B \to BB, B \to BS, B \to b, B \to c\}$

$aba \in L(G)$, $abc \notin L(G)$, $abacba \stackrel{?}{\in} L(G)$

Idee:

- für jedes Teilwort $w_i \dots w_j$ Suche nach Nichtterminalen, welche $w_i \dots w_j$ erzeugen,
- Beginn bei Teilwörtern der Länge 1
- alle Zerlegungen von $w_i \dots w_j$ in zwei Teilwörter testen

CYK-Algorithmus

(Cocke, Younger, Kasami)

induktive Berechnung der Mengen $N_{i,j} = \{A \in N \mid A \to_P^* w_i \dots w_j\}$
durch die Rekursionsgleichung:

$$N_{i,j} = \begin{cases} \{A \mid A \to w_i \in P\} & \text{falls } i = j \\ \bigcup_{i \le k \le j-1} \{A \mid A \to BC \in P \wedge B \in N_{i,k} \wedge C \in N_{k+1,j}\} & \text{falls } i < j \end{cases}$$

(Zerlegungen von $w_i \dots w_j = w_i \dots w_k \circ w_{k+1} \dots w_j$)

dynamische Programmierung: Tabelle (Tafel)

Laufzeit: $\mathcal{O}(|w|^3)$

Fakt

Für kontextfreie Grammatiken $G = (N, T, P, S)$ gilt
$w \in L(G)$ gdw. $S \in N_{1,|w|}$.

CYK-Algorithmus löst Wortproblem für CF in kubischer Zeit

Was bisher geschah

Menge REG aller regulären Sprachen (Chomsky-Typ 3):

- endliche Repräsentation durch
  - reguläre Grammatiken
  - Maschinenmodell NFA (DFA, ε-NFA, ...)
  - reguläre Ausdrücke
- abgeschlossen unter $\cup, \cap, \overline{\ }, \circ, {}^*, {}^R$
- algorithmische Entscheidbarkeit von
  - Wortproblem ($w \stackrel{?}{\in} L$) in $\mathcal{O}(|w|)$,
  - Leerheit, Endlichkeit, Gleichheit, ... regulärer Sprachen

Menge CF aller kontextfreien Sprachen (Chomsky-Typ 2):

- endliche Repräsentation durch
  - kontextfreie Grammatiken (reduziert, Chomsky-NF)
- abgeschlossen unter $\cup, \circ, {}^*, {}^R$,
- algorithmische Entscheidbarkeit von
  - Wortproblem in $\mathcal{O}(|w|^3)$ (CYK-Algorithmus)
  - Leerheit (ÜA)

Maschinenmodelle

Definition (endliche Beschreibung) durch

- interne Steuerung
  (z.B. endliche Menge von Zuständen)
- externen Speicher mit speziellen
  Zugriffsmöglichkeiten
  - Typ des Speicherinhaltes
    (z.B. endliches Wort über endlichem Alphabet)
  - Zugriffsmethode (z.B. Lesen / Schreiben,
    einmal / wiederholt, feste Reihenfolge)

Konfigurationen (Momentaufnahmen)
und endliche Menge zulässiger lokaler Übergänge zwischen Konfigurationen

schrittweise Berechnung : Folge von Konfigurationen von einer Startkonfiguration über zulässige Konfigurationsübergänge

Akzeptanz einer Eingabe durch akzeptierende Berechnung:
endliche Konfigurationenfolge zu einer akzeptierenden Konfiguration

Definition des Maschinenmodells NFA

Aufgabe: Lösung des Wortproblems $w \in L$ für reguläre Sprachen L (gegeben durch NFA A mit $L = L(A)$)

NFA $A = (X, Q, \delta, I, F)$ als abstrakte Maschine:

externer Speicher Eingabeband aus einzelnen linear angeordneten Speicherzellen
(enthält ein Wort $w \in X^*$, jede Zelle ein Symbol aus dem Alphabet, danach rechtes Randsymbol $\square$)

interner Speicher enthält nur einen Zustand aus Q

Steuerelement Zugriff auf

- Eingabeband (externer Speicher): Lesen
  Bewegung in jedem Schritt eine Zelle nach rechts
- Zustand (interner Speicher): Lesen und Schreiben

Arbeitsweise des Maschinenmodells NFA

Start Steuerelement (Lesekopf) in einem Startzustand über der ersten Zelle des Eingabewortes

Schritt besteht aus drei Teilen

1. NFA liest Zeichen in der Zelle unter Lesekopf
2. Zustandsübergang im Steuerelement (entsprechend Übergangsrelation und gelesenem Symbol)
3. Lesekopf bewegt sich zur rechten Nachbarzelle (bei ε-Übergängen keine Bewegung)

Ende Lesekopf auf der Zelle rechts neben dem letzten Symbol des Eingabewortes (auf dem ersten $\square$)

Akzeptanz NFA akzeptiert das Eingabewort gdw. akzeptierender Zustand im internen Speicher

Konfigurationen eines NFA

NFA $A = (X, Q, \delta, I, F)$

Konfiguration $(q, w) \in (Q \times X^*)$ mit
aktuellem Zustand $q \in Q$ und
noch nicht gelesenem Teil des Eingabewortes $w \in X^*$

Startkonfiguration (s, u) mit
Startzustand $s \in I$ und Eingabewort $u \in X^*$

akzeptierende Konfiguration (f, ε) mit akzeptierendem Zustand $f \in F$

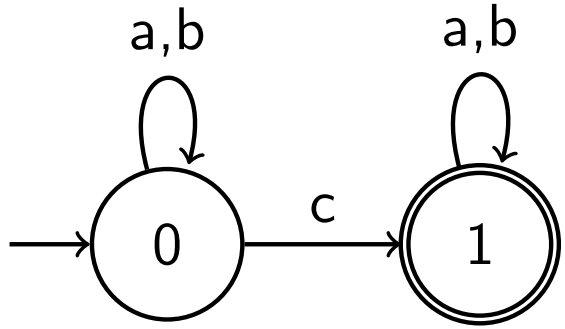
Folgekonfiguration zur Konfiguration (p, aw) mit $a \in X$ und $w \in X^*$:
Konfiguration (q, w) mit $(p, q) \in \delta(a)$
Notation: $(p, aw) \vdash (q, w)$

Berechnung für Eingabewort $w \in X^*$ in A:
Folge $((q_0, u_0), \ldots, (q_n, u_n))$ von Konfigurationen, wobei
für alle $i \in \{0, \ldots, n-1\}$ gilt $(q_i, u_i) \vdash (q_{i+1}, u_{i+1})$

akzeptierende Berechnung für Eingabewort $w \in X^*$ in A:
Berechnung für w mit Endkonfiguration $(q_n, v_n) = (f, \varepsilon)$
(analog zu akzeptierendem Pfad für w in A)

Beispiel

$A = (\{a, b, c\}, \{0, 1\}, \delta, \{0\}, \{1\})$ mit
$\delta(a) = \delta(b) = \{(0, 0), (1, 1)\}$ und $\delta(c) = \{(0, 1)\}$,



Eingabewort *acba*

Startkonfiguration $(0, acba)$

Endkonfiguration $(1, \varepsilon)$ nach der vollständigen Verarbeitung des Eingabewortes

akzeptierende Berechnung für Eingabewort *aaba* in A:

$$(0, acba) \vdash (0, cba) \vdash (1, ba) \vdash (1, a) \vdash (1, \varepsilon)$$

Man bemerke die Analogie zum akzeptierenden Pfad für *acba* in A

$$0 \xrightarrow{a} 0 \xrightarrow{c} 1 \xrightarrow{b} 1 \xrightarrow{a} 1$$

Maschinenmodelle und Algorithmen

- endliche Beschreibung
- schrittweise Bearbeitung
- (Ergebnis nach endlich vielen Schritten)

Jede abstrakte Maschine führt einen Algorithmus aus.
(z.B. zur Lösung des Wortproblems (L, w) für reguläre Sprachen L)
verschiedene Maschinenmodelle unterscheiden sich in

- Art und Kapazität des internen und externen Speichers
- Art des Zugriffs auf internen und externen Speicher
- Reihenfolge der Zugriffe (einmalig, wiederholt)
- Determinismus

Maschinen (z.B. ein gegebener NFA) definieren Sprachen:
Menge aller Eingaben mit akzeptierenden Berechnungen

Maschinenmodelle definieren Sprachklassen:
Menge aller durch eine solche Maschine definierten Sprachen (z.B. REC(NFA))

Kellerautomaten – Motivation

Die Sprache $L = \{wcw^R \mid w \in \{a, b\}^*\}$
ist kontextfrei, aber nicht regulär
(nicht NFA-akzeptierbar).

gesucht:
(einfaches) Maschinenmodell, welches die Sprache L akzeptiert

Maschine benötigt Speicher für Reihenfolge der Symbole in w, um auf diese in umgekehrter Reihenfolge zugreifen zu können (LIFO)

geeignete Datenstruktur für Speicher mit dieser Eigenschaft:
Stack (Keller)

Kellerautomat (pushdown automaton, PDA):
Erweiterung von NFA (mit ε-Übergängen)
um internen Speicher (vom Typ Keller, Stack)

Kellerautomaten (PDA) – Definition

nichtdeterministischer Kellerautomat (PDA)
$A = (X, Q, \Gamma, \delta, q_0, F, \bot)$ mit

- X Alphabet (endlich, nichtleer)
- Q Menge von Zuständen (endlich, nichtleer)
- Γ Kelleralphabet (endlich, nichtleer)
- $q_0 \in Q$ Startzustand
- $F \subseteq Q$ Menge der akzeptierenden Zustände
- $\delta : (X \cup \{\varepsilon\}) \to (Q \times Q \times \Gamma \times \Gamma^*)$
  für jedes $s \in X \cup \{\varepsilon\}$ eine Übergangsrelation
  (mit Änderung des Kellerinhaltes)
- $\bot \in \Gamma$ Kellerboden-Symbol

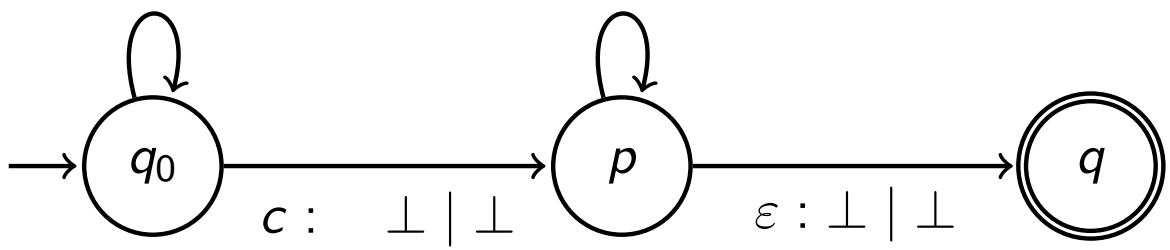
PDA – Beispiel

PDA $A = (\{a, b, c\}, \{q_0, p, q\}, \{\bot, A, B\}, \delta, q_0, \{q\}, \bot)$ mit

$$
\begin{aligned}
\delta(a) &= \{(q_0, q_0, \bot, A\bot), (q_0, q_0, A, AA), (q_0, q_0, B, AB), (p, p, A, \varepsilon)\} \\
\delta(b) &= \{(q_0, q_0, \bot, B\bot), (q_0, q_0, A, BA), (q_0, q_0, B, BB), (p, p, B, \varepsilon)\} \\
\delta(c) &= \{(q_0, p, \bot, \bot), (q_0, p, A, A), (q_0, p, B, B)\} \\
\delta(\varepsilon) &= \{(p, q, \bot, \bot)\}
\end{aligned}
$$

$a: \quad \bot \mid A\bot$
$a: \quad A \mid AA$
$a: \quad B \mid AB$
$b: \quad \bot \mid B\bot$
$b: \quad A \mid BA$
$b: \quad B \mid BB$

$a: \quad A \mid \varepsilon$
$b: \quad B \mid \varepsilon$



PDA – Konfigurationen

Konfiguration $(q, w, k) \in Q \times X^* \times \Gamma^*$ mit

- Zustand $q \in Q$
- (noch zu lesendes) Eingabewort $w \in X^*$
- Kellerinhalt $k \in \Gamma^*$
- Startkonfigurationen $(q_0, w, \bot)$ mit $w \in X^*$ beliebig
- Übergang zwischen Konfigurationen:
  $(p, aw, Gk) \vdash (q, w, k'k)$ mit $(p, q, G, k') \in \delta(a)$ für $k \in \Gamma^*$
  und $(p, w, Gk) \vdash (q, w, k'k)$ mit $(p, q, G, k') \in \delta(\varepsilon)$ für $k \in \Gamma^*$
- akzeptierende Konfigurationen (q, ε, k) mit $q \in F$, $k \in \Gamma^*$

akzeptierende Berechnung (Konfigurationenfolge) für w in A:
Folge $(q_0, w_0, k_0) \vdash \cdots \vdash (q_n, w_n, k_n)$ von Konfigurationen mit

- (q_0, w_0, k_0) ist Startkonfiguration für $w_0 = w$ (also auch $k_0 = \bot$)
- (q_n, w_n, k_n) ist akzeptierende Konfiguration mit $w_n = \varepsilon$
- für jedes $i \in \{1, \ldots, n\}$ gilt $(q_{i-1}, w_{i-1}, k_{i-1}) \vdash (q_i, w_i, k_i)$

Von PDA akzeptierte Sprachen

PDA $A = (X, Q, \Gamma, \delta, q_0, F, \bot)$ akzeptiert das Wort $w \in X^*$ gdw. eine akzeptierende Berechnung (Konfigurationenfolge) mit der Startkonfiguration $(q_0, w, \bot)$ existiert.

PDA $A = (X, Q, \Gamma, \delta, q_0, F, \bot)$ akzeptiert die Sprache

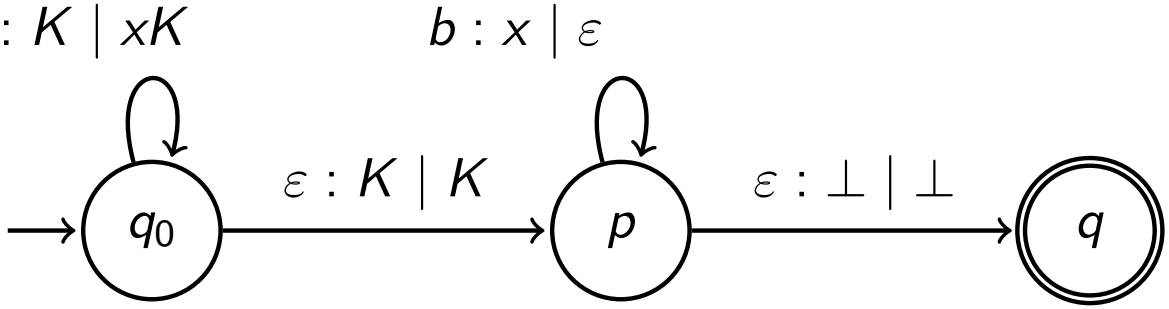
$$L(A) = \{w \in X^* \mid A \text{ akzeptiert } w\}$$

Sprache L heißt PDA-akzeptierbar gdw. ein PDA A mit $L = L(A)$ existiert.

PDA A und B heißen äquivalent gdw. $L(A) = L(B)$

PDA-Akzeptanz – Beispiel

PDA $A = (\{a, b\}, \{q_0, p, q\}, \{\bot, x\}, \delta, q_0, \{q\}, \bot)$ mit
$\delta(a) = \{(q_0, q_0, K, xK)\}$ für $K \in \{x, \bot\}$
$\delta(b) = \{(p, p, x, \varepsilon)\}$
$\delta(\varepsilon) = \{(q_0, p, K, K), (p, q, \bot, \bot)\}$ mit $K \in \{x, \bot\}$



für $K \in \{x, \bot\}$
A akzeptiert das Wort $aabb$ über die Konfigurationenfolge

$$(q_0, aabb, \bot) \vdash (q_0, abb, x\bot) \vdash (q_0, bb, xx\bot) \vdash (p, bb, xx\bot)$$
$$\vdash (p, b, x\bot) \vdash (p, \varepsilon, \bot) \vdash (q, \varepsilon, \bot)$$

und ε über die Konfigurationenfolge $(q_0, \varepsilon, \bot) \vdash (p, \varepsilon, \bot) \vdash (q, \varepsilon, \bot)$
aber nicht abb, aba
Es gilt $L(A) = \{a^n b^n \mid n \in \mathbb{N}\}$
Damit ist die Sprache $\{a^n b^n \mid n \in \mathbb{N}\}$ PDA-akzeptierbar.

Deterministische PDA

PDA $A = (X, Q, \Gamma, \delta, q_0, F, \bot)$ heißt deterministisch (DPDA) gdw.

$$\forall a \in X \; \forall q \in Q \; \forall K \in \Gamma : |\{(p, B) \mid (q, p, K, B) \in \delta(a) \cup \delta(\varepsilon)\}| \leq 1$$

Sprache L heißt deterministisch kontextfrei gdw. ein deterministischer PDA A mit $L = L(A)$ existiert.

Beispiele:

- $L = \{a^n b^n \mid n \in \mathbb{N}\}$ ist deterministisch kontextfrei,
- $L' = \{wcw^R \in \{a, b, c\}^* \mid w \in \{a, b\}^*\}$ ist deterministisch kontextfrei,
- $L'' = \{waw^R \in \{a, b\}^* \mid w \in \{a, b\}^*\}$ ist kontextfrei, aber nicht deterministisch kontextfrei.

Die Menge aller durch deterministische PDA akzeptierbaren Sprachen ist eine echte Teilmenge der Menge aller durch PDA akzeptierbaren Sprachen.

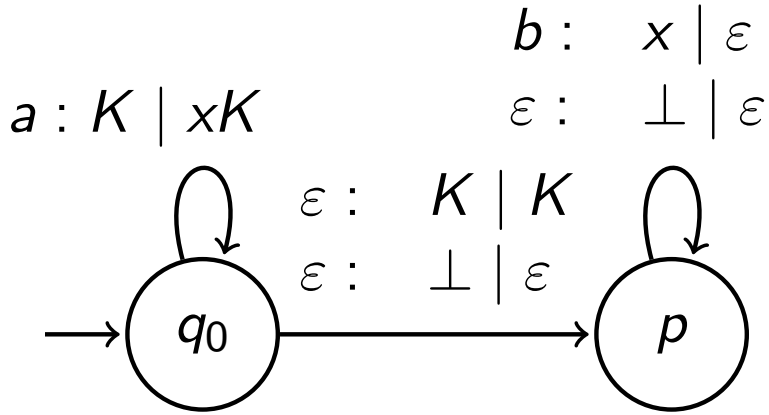
PDA – Akzeptanz durch leeren Keller

alternative Akzeptanzbedingung

akzeptierende Konfigurationen: $(q, \varepsilon, \varepsilon)$ mit $q \in Q$ beliebig

Angabe akzeptierender Zustände überflüssig

Beispiel: $L = \{a^n b^n \mid n \in \mathbb{N}\}$ ist PDA-akzeptierbar durch
PDA $A = (\{a, b\}, \{q_0, p\}, \{\bot, x\}, \delta, q_0, \bot)$ mit
$\delta(a) = \{(q_0, q_0, K, xK)\}$ für $K \in \Gamma$
$\delta(b) = \{(p, p, x, \varepsilon)\}$
$\delta(\varepsilon) = \{(q_0, p, K, K), (q_0, p, \bot, \varepsilon), (p, p, \bot, \varepsilon)\}$ für $K \in \{x, \bot\}$



für $K \in \{x, \bot\}$

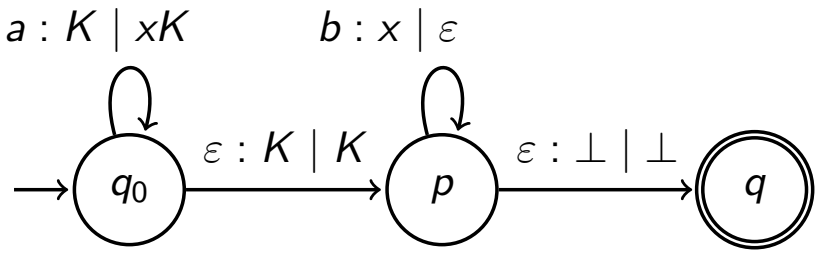
Akzeptanzbedingungen für PDA

zwei Möglichkeiten zur Definition akzeptierender Konfigurationen:

1. Akzeptanz durch akzeptierenden Zustand (analog NFA):
   (q, ε, k) mit $q \in F$ und $k \in \Gamma^*$ beliebig
2. Akzeptanz durch leeren Keller:
   $(q, \varepsilon, \varepsilon)$ mit $q \in Q$ beliebig

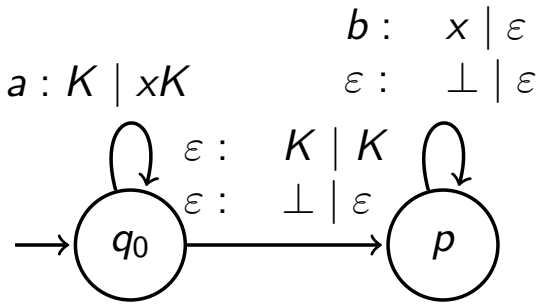
Beispiel: Die Sprache $L = \{a^n b^n \mid n \in \mathbb{N}\}$ wird

durch akzeptierende Zustände akzeptiert durch den PDA $A = (\{a, b\}, \{q_0, p, q\}, \{\bot, x\}, \delta, q_0, \{q\}, \bot)$ mit



für $K \in \{x, \bot\}$

mit leerem Keller akzeptiert durch den PDA $B = (\{a, b\}, \{q_0, p, q\}, \{\bot, x\}, \delta, q_0, \bot)$ mit



Äquivalenz beider Akzeptanzbedingungen

Satz

1. *Zu jedem PDA mit akzeptierenden Zuständen existiert ein äquivalenter PDA, der mit leerem Keller akzeptiert.*
2. *Zu jedem PDA, der mit leerem Keller akzeptiert, existiert ein äquivalenter PDA mit akzeptierenden Zuständen.*

Idee (PDA-Transformationen):

1. PDA A mit akzeptierenden Zuständen $\rightarrow$ PDA B mit leerem Keller:
   B enthält zusätzliches Kellerboden-Symbol und neuen Finalzustand,
   ursprünglicher Automat A arbeitet „darüber“,
   nach Übergang von A in einen akzeptierenden Zustand:
   ε-Übergänge in B in diesem akzeptierenden Zustand zum Entfernen aller im Keller verbliebenen Symbole (incl. beider Kellerboden-Symbole)
2. PDA A mit leerem Keller $\rightarrow$ PDA B mit akzeptierenden Zuständen:
   in B neuen (einzigen) akzeptierenden Zustand hinzufügen,
   für jeden Übergang in A in eine Konfiguration mit leerem Keller in B einen Übergang in diesen akzeptierenden Zustand hinzufügen.

Schnitt PDA-akzeptierbarer mit regulären Sprachen

Satz

- *Sind L eine PDA-akzeptierbare und L' eine reguläre Sprache, dann ist die Sprache $L \cap L'$ PDA-akzeptierbarer.*
- *Sind L eine durch einen DPDA-akzeptierbare und L' eine reguläre Sprache, dann ist die Sprache $L \cap L'$ durch einen DPDA akzeptierbarer.*

gegeben: PDA $A = (X, Q_A, \Gamma, \delta_A, i_A, F_A, \bot)$ mit $L = L(A)$
(Akzeptanz durch akzeptierende Zustände)
DFA $B = (X, Q_B, \delta_B, i_B, F_B)$ mit $L' = L(B)$

Produktkonstruktion:
PDA $C = (X, Q_A \times Q_B, \Gamma, \delta, (i_A, i_B), F_A \times F_B, \bot)$ mit

$$\forall a \in X : \delta(a) = \left\{ ((p_A, p_B), (q_A, q_B), G, k) \mid \begin{array}{l} (p_A, q_A, G, k) \in \delta_A(a) \\ \wedge (p_B, q_B) \in \delta_B(a) \end{array} \right\}$$

$$\delta(\varepsilon) = \{((p_A, p_B), (q_A, p_B), G, k) \mid (p_A, q_A, G, k) \in \delta_A(\varepsilon)\}$$

Für den oben konstruierten PDA C gilt $L(C) = L(A) \cap L(B)$.
Ist der PDA A deterministisch, dann ist auch C deterministisch.
Beispiel: $\{w \in \{a, b\}^* \mid w = w^R\} \cap a^* b a^*$

Anwendung zum Schnitt mit regulären Sprachen

$$\forall L, L' \subseteq X^* : ((L \in \text{CF} \wedge L' \in \text{REG}) \rightarrow (L \cap L') \in \text{CF})$$

(CF ist unter Schnitt mit regulären Sprachen abgeschlossen)

$$\forall L, L' \subseteq X^* : ((L' \in \text{REG} \wedge (L \cap L') \notin \text{CF}) \rightarrow L \notin \text{CF})$$

ist dazu äquivalent (analog ÜA) und nützlich

Beispiele: wegen $\{a^n b^n c^n \mid n \in \mathbb{N} \setminus \{0\}\} \notin \text{CF}$ (Nachweis später)

- Für $L = \{w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c\}$ und $L' = L(a^+ b^+ c^+) \in \text{REG}$
  ist $L \cap L' = \{a^n b^n c^n \mid n \in \mathbb{N} \setminus \{0\}\} \notin \text{CF}$ ()
  und damit auch $L \notin \text{CF}$
- Für $L = \{a^j b^k c^l d^m \mid j = 0 \vee k = l = m\}$ ist $L \cap L(ab^* c^* d^*) = \{ab^k c^k d^k \mid k \in \mathbb{N}\} \notin \text{CF}$
  und damit auch $L \notin \text{CF}$

Kontextfreie Grammatiken und PDA

Satz

Die Menge aller kontextfreien (d.h. durch eine kontextfreie Grammatik erzeugten) Sprachen ist genau die Menge aller PDA-akzeptierbaren Sprachen.

Beweis (konstruktiv):

1. Konstruktion eines PDA A aus gegebener kontextfreier Grammatik G mit $L(A) = L(G)$
2. Konstruktion einer kontextfreien Grammatik G aus gegebenem PDA A mit $L(A) = L(G)$

Kontextfreie Grammatik $\rightarrow$ PDA

Beispiel: Łukasiewicz-Sprache erzeugt durch die Grammatik
$G = (\{S\}, \{a, b\}, P, S)$ mit $P = \{S \rightarrow a, S \rightarrow bSS\}$

Lemma

Zu jeder kontextfreien Grammatik G existiert ein PDA A mit $L(A) = L(G)$.

gegeben: kontextfreie Grammatik $G = (N, X, P, S)$

Idee: Tiefensuche im Ableitungsbaum (Linksableitung)

Konstruktion: PDA $A = (X, \{q\}, N \cup X, \delta, q, S)$ mit LK-Akzeptanz und

$$\begin{aligned} \delta(\varepsilon) &= \{(q, q, B, w) \mid B \rightarrow w \in P\} \\ \forall a \in X : \delta(a) &= \{(q, q, a, \varepsilon)\} \end{aligned}$$

Für den so konstruierten PDA A gilt $L(A) = L(G)$.
Nachweis: Eindeutige Zuordnung zwischen Linksableitungen

$$S \rightarrow u_1 A_1 v_1 \rightarrow u_1 u_2 A_2 v_2 \rightarrow \cdots \rightarrow u_1 \ldots u_{m-1} A_{m-1} v_{m-1} \rightarrow w = u_1 \ldots u_m$$

(mit $u_i \in X^*$, $v_i \in (N \cup X)^*$) und akz. Berechnungen für $w = u_1 \ldots u_m$

$$\begin{aligned} (q, w, S) &\vdash (q, u_1 \ldots u_m, u_1 A_1 v_1) \vdash \cdots \vdash (q, u_2 \ldots u_m, A_1 v_1) \\ &\vdash (q, u_2 \ldots u_m, u_2 A_2 v_2) \vdash \cdots \vdash (q, u_m, u_m) \vdash \cdots \vdash (q, \varepsilon, \varepsilon) \end{aligned}$$

Nicht-kontextfreie Sprachen

Die Grammatik $G = (\{A, B, C\}, \{a, b, c\}, P, A)$ mit

$$P = \left\{ \begin{array}{ll} A & \to aABC, \\ A & \to aBC, \\ CB & \to BC, \\ aB & \to ab, \\ bB & \to bb, \\ bC & \to bc, \\ cC & \to cc \end{array} \right\}$$

erzeugt die Sprache $L(G) = \{a^n b^n c^n \mid n > 0\}$.
(Chomsky-Typ 1, kontextsensitiv)

Fakt

Die Sprache $L = \{a^n b^n c^n \mid n > 0\}$ ist nicht kontextfrei.

Nachweis für $L = \{a^n b^n c^n \mid n > 0\} \notin \text{CF}$

Idee: indirekter Beweis mit SFS (analog $\{a^n b^n \mid n > 0\} \notin \text{REG}$)

Beweis (Skizze): z.z.:

$\forall$ CFG $G = (N, T, P, S)$ in Chomsky-NF: $((L(G) = L) \rightarrow \neg(L(G) = L))$

Annahme: für CFG $G = (N, T, P, S)$ mit $|N| = k$ gilt $L(G) = L$

Wegen $w = a^{(2^k)} b^{(2^k)} c^{(2^k)} \in L$ existiert nach Annahme also ein Ableitungsbaum t für $a^{(2^k)} b^{(2^k)} c^{(2^k)}$ in G (Tafel)

t hat Höhe wenigstens $\lceil \log_2(3 \cdot 2^k) \rceil > \log_2(2 \cdot 2^k) = k + 1$, also existiert in t ein Pfad $p = [p_0, p_1, \ldots, p_{k'}] \in N^* \circ T$ mit $p_0 = S$ und $k' > k + 1$

SFS über $p' = [p_0, p_1, \ldots, p_{k'-1}] \in N^*$:

O: Positionen auf Pfad p': $|\{0, \ldots, k' - 1\}| = k' > k$,

S: Nichtterminale $|N| = k$

Nach SFS existieren $i, j \in \{0, \ldots, k' - 1\}$ mit $i < j$ und $p_i = p_j$.

Also ergibt die Ersetzung des Teilbaumes mit Wurzel p_i durch Teilbaum mit Wurzel p_j in t einen Ableitungsbaum für Wort $w' \neq a^{(2^k)} b^{(2^k)} c^{(2^k)}$.

Tafel: Analyse aller möglichen Wörter w' durch Zerlegungen (definiert durch Pfad p)

In allen Fällen gilt $w' \notin L$ und damit $\neg(L(G) = L)$ im Widerspruch zur Annahme.

Also ist die Annahme falsch und es existiert keine CFG G mit $L = L(G)$.

Beispiele nicht-kontextfreier Sprachen

- $\{a^n b^n c^n \mid n \in \mathbb{N} \setminus \{0\}\}$
- $\{a^n b^m a^n b^m \mid m, n \in \mathbb{N} \setminus \{0\}\}$ (ÜA)
- $\left\{a^{(2^k)} \mid k \in \mathbb{N}\right\}$

Man bemerke: $\{a^n b^n c^n \mid n \in \mathbb{N} \setminus \{0\}\} = L_1 \cap L_2$ für
$L_1 = \{a^n b^n c^m \mid m, n \in \mathbb{N} \setminus \{0\}\}$ und $L_2 = \{a^n b^m c^m \mid m, n \in \mathbb{N} \setminus \{0\}\}$
L_1 und L_2 sind beide kontextfrei.

Folgerung: CF ist nicht unter $\cap$ abgeschlossen.

Was folgt daraus für den Abschluss von CF unter $\overline{}$? (ÜA)

CF: Algorithmische Lösungen

algorithmisch lösbar sind für kontextfreie Sprachen (Grammatiken)

- Wortproblem: Eingabe (G, w)
  Ausgabe: ja, falls $w \in L(G)$, sonst nein
  (CYK-Algorithmus, $\mathcal{O}(|w|^3)$)
- Leerheitsproblem: Eingabe G,
  Ausgabe: ja, falls $L(G) = \emptyset$, sonst nein
  (ja, falls Startsymbol der Grammatik nicht erzeugend, Reduktion)
- Endlichkeitsproblem: Eingabe G,
  Ausgabe: ja, falls $L(G)$ endlich, sonst nein

nicht algorithmisch entscheidbar sind (Beweise später in TI Master)

- Inklusion $\subseteq$: Eingabe G_1, G_2,
  Ausgabe: ja, falls $L(G_1) \subseteq L(G_2)$, sonst nein
- Gleichheit $=$: Eingabe G_1, G_2,
  Ausgabe: ja, falls $L(G_1) = L(G_2)$, sonst nein
- Disjunktheit: Eingabe G_1, G_2,
  Ausgabe: ja, falls $L(G_1) \cap L(G_2) = \emptyset$, sonst nein
- Kontextfreiheit der Schnittsprache: Eingabe G_1, G_2,
  Ausgabe: ja, falls $L(G_1) \cap L(G_2)$ kontextfrei, sonst nein

CF – Zusammenfassung

CF kontextfreie Sprachen (Chomsky-Typ 2)

- CF=REC(PDA)
- CF abgeschlossen unter $\cup, \circ, {}^*, {}^R$, Schnitt mit regulären Sprachen, aber nicht unter $\overline{}, \cap$
- CYK-Algorithmus zur Lösung des Wortproblems
- Leerheit entscheidbar, Gleichheit, Schnitt in CF nicht

DCF deterministisch kontextfreie Sprachen

- DCF=REC(DPDA) ist echte Teilmenge von CF
  z.B. $\{wav \mid w, v \in \{a, b\}^* \wedge |w| = |v|\} \in \mathrm{CF} \setminus \mathrm{DCF}$
- DCF abgeschlossen unter ${}^R, \overline{}$, Schnitt mit regulären Sprachen, aber nicht unter $\cup, \cap, \circ, {}^*$
- Lösung des Wortproblems für Sprachen in DCF in $\mathcal{O}(|w|)$
- Leerheit, Gleichheit entscheidbar, Schnitt in DCF nicht

Was bisher geschah

Chomsky-Hierarchie für formale Sprachen
(definiert über Eigenschaften von Grammatiken): Chomsky-Typ

- 3 : reguläre Grammatiken / Sprachen
- 2 : kontextfreie Grammatiken / Sprachen
- 1 : monotone Grammatiken / kontextsensitive Sprachen
- 0 : von beliebigen Grammatiken erzeugte Sprachen

Maschinenmodelle zur Lösung des Wortproblems für Sprachen von Typ

- 3 : NFA, DFA, ε-NFA
- 2 : (nichtdeterministische) PDA
  $\mathrm{REC(NFA)} \subset \mathrm{REC(DPDA)} \subset \mathrm{REC(PDA)}$
  ($\mathrm{REC(DPDA)} \subset \mathrm{REC(PDA)}$ im SS25 nicht bewiesen, aber Beispiel)
  - $\{a^n b^n \mid n \in \mathbb{N}\} \in \mathrm{REC(DPDA)} \setminus \mathrm{REC(NFA)}$
  - $\{a^n b a^n \mid n \in \mathbb{N}\} \in \mathrm{REC(DPDA)} \setminus \mathrm{REC(NFA)}$
  - $\{wav \mid w, v \in \{a, b\}^* \wedge |w| = |v|\} \in \mathrm{REC(PDA)} \setminus \mathrm{REC(DPDA)}$
  - $\{a^n b^n c^n \mid n \in \mathbb{N}\}, \{a^n b^m a^n b^m \mid n \in \mathbb{N}\}, \{ww \mid w \in \{a, b\}^*\} \notin \mathrm{REC(PDA)}$ (Nachweis durch SFS)
- 1 : ?
- 0 : ?

Kontextsensitive Sprachen (Chomsky-Typ 1)

erzeugt durch Grammatiken $G = (N, T, P, S)$ mit Regeln der Form $l \to r$, wobei

- $l \in (N \cup T)^+$
- $r \in (N \cup T)^*$ und
- $|l| \leq |r|$ (nichtverkürzend, monoton)

Beispiel: $\{a^n b^n c^n \mid n > 0\}$ ist

- nicht kontextfrei (Nachweis durch SFS)
- erzeugt durch Grammatik $G = (\{A, B, C\}, \{a, b, c\}, P, A)$ mit

$$P = \left\{ \begin{array}{lcl} A & \to & aABC, \\ A & \to & aBC, \\ CB & \to & BC, \\ aB & \to & ab, \\ bB & \to & bb, \\ bC & \to & bc, \\ cC & \to & cc \end{array} \right\}$$

Linear beschränkter Automat (LBA) – Idee

Ziel: Maschinenmodell für Sprachen vom Chomsky-Typ 1

Definition (endliche Beschreibung) durch

- externer Speicher:
  1. Eingabeband aus Zellen
     - Typ des Speicherinhaltes:
       endliches Wort über endlichem Alphabet (Eingabealphabet)
     - Zugriffsmethode: Lesen
       Bewegung des Lese-Kopfes auf das rechte Nachbarfeld
- interner Speicher:
  2. endliche Menge von Zuständen (Lesen / Schreiben)
  3. Arbeitsband aus Zellen
     - Typ des Speicherinhaltes:
       endliches Wort (Länge = Länge des Eingabewortes)
       über endlichem Alphabet (Arbeitsalphabet)
     - Zugriffsmethode: Lesen und Schreiben
       Bewegung des Lese/Schreib-Kopfes auf ein Nachbarfeld (R/L)

übliche Vereinfachung der Struktur:

Zusammenfassen der Bänder im internen und externen Speicher zu einem Eingabe- und Arbeitsband (1. und 3.) mit Lese- und Schreib-Zugriff

WH: LBA – Definition

Linear beschränkter Automat (LBA) $M = (X, Q, \Gamma, \delta, q_0, F, \square)$ mit

- X endliches Eingabealphabet
- Q endliche Menge von Zuständen
- $\Gamma \supset X$ endliches Arbeitsalphabet
- $\delta \subseteq (\Gamma \times Q \times \Gamma \times Q \times \{L, R, N\})$
  Übergangsrelation
- q_0 Startzustand
- F akzeptierende Zustände
- $\square \in \Gamma \setminus X$ Leere-Zelle-Symbol
  (Markierung von Anfang und Ende der Eingabe)

LBA M heißt deterministisch gdw.
für jedes $a \in \Gamma$ und jedes $q \in Q$ gilt

$$|\{(a, q, b, p, x) \mid p \in Q, b \in \Gamma, x \in \{L, R, N\}\} \cap \delta| \leq 1$$

LBA – Beispiel

LBA $M =$
$(\{a, b, c\}, \{q_0, q_1, q_2, q_3, q_4, q_5, f\}, \{a, b, c, x, \square\}, \delta, q_0, \{f\}, \square)$ mit

$$\delta = \left\{ \begin{array}{l} (a, q_0, \square, q_1, R), (a, q_1, a, q_1, R), (x, q_1, x, q_1, R) \\ (b, q_1, x, q_2, R), (b, q_2, b, q_2, R), (x, q_2, x, q_2, R) \\ (c, q_2, x, q_3, R), (c, q_3, c, q_3, R), (\square, q_3, \square, q_4, L) \\ (a, q_4, a, q_4, L), (b, q_4, b, q_4, L) \\ (c, q_4, c, q_4, L), (x, q_4, x, q_4, L) \\ (\square, q_4, \square, q_0, R) \\ (x, q_0, \square, q_5, R), (x, q_5, \square, q_5, R), (\square, q_5, \square, f, N) \\ (\square, q_0, \square, f, R) \end{array} \right\}$$

Konfigurationen eines LBA

LBA $M = (X, Q, \Gamma, \delta, q_0, F, \square)$

Konfiguration $uqv \in (\Gamma^* \times q \times \Gamma^*)$ bedeutet:

- aktueller Bandinhalt: $w = uv$
- aktueller Zustand des LBA: q
- Schreib-/Lesekopf des LBA zeigt auf erstes Symbol von v (Leere-Zelle-Symbol, falls $v = \varepsilon$)

Startkonfigurationen $q_0 w$ mit $w \in X^*$
akzeptierende Konfigurationen uqv mit $q \in F$
Konfigurationsübergänge von upv mit $u = u'a$ und $v = bv'$:

$$\begin{aligned} &\text{für} \quad (b, p, c, q, R) \in \delta : \quad upv \vdash ucqv' \\ &\text{für} \quad (b, p, c, q, L) \in \delta : \quad upv \vdash u'qacv' \\ &\text{für} \quad (b, p, c, q, N) \in \delta : \quad upv \vdash uqcv' \end{aligned}$$

Akzeptanz durch LBA

LBA $M = (X, Q, \Gamma, \delta, q_0, F, \square)$

M akzeptiert das Wort $w \in X^*$ gdw.
eine Folge $k_0, \ldots, k_n$ von Konfigurationen $k_i \in \Gamma^* \times Q \times \Gamma^*$ exisiert, so dass gilt:

1. $k_0 = q_0 w$ ist die Startkonfiguration mit Eingabe w
2. $k_n \in \Gamma^* \times F \times \Gamma^*$ ist eine akzeptierende Konfiguration
3. für jedes $i \in \{1, \ldots, n\}$ ist $k_{i-1} \vdash k_i$ ein zulässiger Konfigurationsübergang

M akzeptiert die Sprache $L(M) = \{w \in X^* \mid M \text{ akzeptiert } w\}$

$L \subseteq X^*$ heißt LBA-akzeptierbar gdw. ein LBA M mit $L = L(M)$ existiert.

Beispiel: Sprache vom Chomsky-Typ 1

monotone Grammatik: Regeln $l \to r$ mit $l, r \in (T \cup N)^+$ mit $|l| \leq |r|$
$G = (\{S, M, A, B\}, \{a, b, x\}, P, S)$ mit

$$P = \left\{ \begin{array}{lcl} S & \to & Mx|x \\ M & \to & aA|bB|aMA|bMB \\ Aa & \to & aA, Ab \to bA, Ax \to ax \\ Ba & \to & aB, Bb \to bB, Bx \to bx \end{array} \right\}$$

Ableitung für das Wort *abbabbx*

$$\begin{array}{rcl} S & \to & \underline{M}x \to aM\underline{A}x \to a\underline{M}ax \\ & \to & abM\underline{Ba}x \to abMa\underline{B}x \to ab\underline{M}abx \\ & \to & abb\underline{Ba}bx \to abba\underline{Bb}x \to abbab\underline{B}x \to abbabbx \end{array}$$

Beobachtung: Für nichtverkürzende Grammatiken ist kein Wort in der Ableitung länger als das abgeleitete Terminalwort.

Satz

Die Menge aller Sprachen, die von einem LBA akzeptiert werden, ist genau die Menge aller Sprachen vom Chomsky-Typ 1.

Eigenschaften der Menge aller kontextsensitiven Sprachen

- Die Menge aller durch (nichtdeterministische) LBA akzeptierbaren Sprachen ist genau die Menge aller kontextsensitiven Sprachen. (hier ohne Beweis)
- Abschlusseigenschaften (im SS25 nicht bewiesen):
  Sind L_1 und L_2 kontextsensitive Sprachen, dann sind auch
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $\overline{L_1}$ (Satz von Immerman und Szelepcsényi)
  - $L_1 \circ L_2$
  - L_1^*
  - L_1^R

  kontextsensitive Sprachen.
- Für kontextsensitive Sprachen ist das Wortproblem (Gilt $w \in L$?) algorithmisch entscheidbar.
- Ob deterministische und nichtdeteministische LBA die selbe Sprachklasse akzeptieren, ist bisher unbekannt (LBA-Problem)

Beispiel: Sprache vom Chomsky-Typ 0

Regeln $l \to r$ mit $l \in (T \cup N)^+, r \in (T \cup N)^*$
$G = (\{S, T, \}, \{0, 1\}, P, S)$ mit

$$P = \left\{ \begin{array}{lcl} S & \to & 1S101 \\ S & \to & 01S00 \\ S & \to & 110S11 \\ 0S0 & \to & T \\ 0T0 & \to & T \\ 1S1 & \to & T \\ 1T1 & \to & T \\ T & \to & \varepsilon \end{array} \right\}$$

Gilt $\varepsilon \in L(G)$?

$$\begin{array}{rcl} S & \to & 110S11 \to 1100\underline{1S00}11 \to 11001\underline{110S11}0011 \\ & \to & 110011101\underline{S101}110011 \to 11001110\underline{T}01110011 \\ & \to^* & 110T011 \to 11T11 \to 1T1 \to T \to \varepsilon \end{array}$$

Beobachtung: Während der Ableitung kommen Wörter vor, die mehr Terminalsymbole enthalten als das abgeleitete Wort.

Turing-Maschinen

wie Maschinenmodell LBA mit dem einzigen Unterschied:

LBA längenbeschränktes Arbeitsband

TM beliebig langes Arbeitsband

Parameter des Maschinenmodelles TM:

- externer Speicher: Eingabeband aus Zellen
- interner Speicher:
  1. Zustand (Zugriff: Lesen und Schreiben) und
  2. beidseitig unendliches Arbeitsband
     (Zugriff: Lesen und Schreiben, Bewegung auf Nachbarzelle)

- häufige (äquivalente) Darstellung:
  beidseitig unendliches Eingabeband = Arbeitsband

Turing-Maschine – Definition

Turing-Maschine (TM) $M = (X, Q, \Gamma, \delta, q_0, F, \square)$ mit

- X endliches Eingabealphabet
- Q endliche Menge von Zuständen
- $\Gamma \supset X$ endliches Arbeitsalphabet
- $\delta \subseteq (\Gamma \times Q \times \Gamma \times Q \times \{L, R, N\})$ Übergangsrelation
- q_0 Startzustand
- F akzeptierende Zustände
- $\square \in \Gamma \setminus X$ Leere-Zelle-Symbol

TM M heißt deterministisch gdw.
für jedes $a \in \Gamma$ und jedes $q \in Q$ gilt

$$|\{(a, q, b, p, x) \mid p \in Q, b \in \Gamma, x \in \{L, R, N\}\} \cap \delta| \leq 1$$

Konfigurationen einer TM

TM $M = (X, Q, \Gamma, \delta, q_0, F, \square)$

Konfiguration $uqv \in (\Gamma^* \times q \times \Gamma^*)$ bedeutet:

- aktueller Bandinhalt: $w = uv$
- aktueller Zustand der TM: q
- Schreib-/Lesekopf der TM zeigt auf erstes Symbol von v (Leere-Zelle-Symbol, falls $v = \varepsilon$)

Startkonfigurationen $q_0 w$ mit $w \in X^*$
akzeptierende Konfigurationen uqv mit $q \in F$
Konfigurationsübergänge von upv mit $u = u'a$ und $v = bv'$:

$$\begin{aligned} &\text{für} \quad (b, p, c, q, R) \in \delta : \quad upv \vdash ucqv' \\ &\text{für} \quad (b, p, c, q, L) \in \delta : \quad upv \vdash u'qacv' \\ &\text{für} \quad (b, p, c, q, N) \in \delta : \quad upv \vdash uqcv' \end{aligned}$$

Akzeptanz durch TM

TM $M = (X, Q, \Gamma, \delta, q_0, F, \square)$

M akzeptiert das Wort $w \in X^*$ gdw.
eine Folge $k_0, \ldots, k_n$ von Konfigurationen $k_i \in \Gamma^* \times Q \times \Gamma^*$ exisiert, so dass gilt:

1. $k_0 = q_0 w$ ist die Startkonfiguration mit Eingabe w
2. $k_n \in \Gamma^* \times F \times \Gamma^*$ ist eine akzeptierende Konfiguration
3. für jedes $i \in \{1, \ldots, n\}$ ist $k_{i-1} \vdash k_i$ ein zulässiger Konfigurationsübergang

M akzeptiert die Sprache $L(M) = \{w \in X^* \mid M \text{ akzeptiert } w\}$

$L \subseteq X^*$ heißt Turing-akzeptierbar gdw. eine TM M mit $L = L(M)$ existiert.

Beispiele für Turing-Maschinen

- TM $M_1 = (\{a, b\}, \{q_0, f\}, \{a, b, \Box\}, \delta, q_0, \{f\}, \Box)$ mit

$$\delta = \{(a, q_0, a, q_0, L), (b, q_0, b, q_0, L), (\Box, q_0, \Box, f, R)\}$$

$L(M_1) = \{a, b\}^*$

- TM $M_2 = (\{a, b\}, \{q_0, q_a, q_b, f\}, \{a, b, \Box\}, \delta, q_0, \{f\}, \Box)$ mit

$$\delta = \left\{ \begin{array}{l} (a, q_0, \Box, q_a, R), (b, q_0, \Box, q_b, R), \\ (a, q_a, a, q_a, R), (b, q_a, b, q_a, R), \\ (a, q_b, a, q_b, R), (b, q_b, b, q_b, R), \\ (\Box, q_0, \Box, f, N), (\Box, q_a, a, f, N), (\Box, q_b, b, f, N) \end{array} \right\}$$

$L(M_2) = \{a, b\}^*$

- TM $M_3 = (\{1\}, \{q_0, q_1, q_2, q_3, q_4, f\}, \{1, x, \Box\}, \delta, q_0, \{f\}, \Box)$ mit

$$\delta = \left\{ \begin{array}{l} (1, q_0, 1, q_1, R), (1, q_1, x, q_2, R), (1, q_2, 1, q_3, R), \\ (1, q_3, x, q_2, R), (1, q_4, 1, q_4, L), (x, q_1, x, q_1, R), \\ (x, q_2, x, q_2, R), (x, q_3, x, q_3, R), (x, q_4, x, q_4, L), \\ (\Box, q_1, \Box, f, L), (\Box, q_2, \Box, q_4, L), (\Box, q_4, \Box, q_0, R) \end{array} \right\}$$

$L(M_3) = \{1^{(2^n)} \mid n \in \mathbb{N}\}$ (Unärdarstellung aller Zweierpotenzen)

Nichtdeterministische TM

TM $M = (\{0,1\}, \{q_0, q_1, f\}, \{0,1,\Box\}, \delta, q_0, \{f\}, \Box)$ mit

$$\begin{aligned}\delta \quad = \{ \quad & (0, q_0, 1, q_0, R) \\ & (1, q_0, 1, q_0, R) \\ & (1, q_0, 1, q_1, R) \\ & (1, q_1, 1, q_1, R) \\ & (\Box, q_1, \Box, f, L)\}\end{aligned}$$

akzeptiert z.B. das Wort 11011 mit (u.A.) folgender Berechnung

$$\begin{aligned} q_0 11011 \quad &\vdash \quad 1 q_0 1011 \vdash 11 q_0 011 \vdash 111 q_0 11 \vdash 1111 q_1 1 \\ &\vdash \quad 11111 q_1 \Box \vdash 1111 f 1 \end{aligned}$$

Simulation nichtdeterministischer TM

Satz

Zu jeder nichtdeterministischen TM M existiert eine deterministische TM M' mit $L(M) = L(M')$.

Beweisidee (konstruktiv):
Menge aller möglichen Berechnungen von M bei Eingabe von w bilden einen Baum (evtl. mit unendlichen Pfaden)

Knoten: Konfigurationen
Wurzel: Startkonfiguration (Startzustand, Eingabewort)
Blätter: akzeptierende Konfiguration
Kanten: zulässige Konfigurationsübergänge in M

M' führt Breitensuche in diesem Baum durch
(simuliert parallele Berechnung aller Möglichkeiten, dovetailing),
Bandinhalt von M' sind Konfigurationen aus M
M' akzeptiert, sobald (u.A.) eine akzeptierende Konfiguration von M auf dem Arbeitsband steht.

Operationen auf TM-akzeptierbaren Sprachen

Die Menge aller Turing-akzeptierbaren Sprachen ist

- abgeschlossen unter $\cap$, $\cup$, $\circ$, *, ^{R}
- nicht unter Komplement abgeschlossen. (Gegenbeispiel in TIM)

Konstruktionen

gegeben: TM A, TM B (Akzeptanz durch Halt)
gesucht: TM C mit

$\cap$: $L(C) = L(A) \cap L(B)$
(sequentieller Akzeptanztest durch beide TM)

$\cup$: $L(C) = L(A) \cup L(B)$
(verschränkter Akzeptanztest durch beide TM)

$\circ$: $L(C) = L(A) \circ L(B)$
(nichtdeterministische Zerlegung und verschränkter Test)

*: $L(C) = L(A)^*$
(nichtdeterministische Zerlegung und verschränkter Test)

^{R}: $L(C) = L(A)^R$
(Vorbereitung: auf dem Arbeitsband w durch w^R ersetzen)

mit Turing-Maschinen für „Unterprogramme“, z.B. Verschieben

TM und Sprachen vom Chomsky-Typ 0

Satz

Die Menge aller Sprachen, die von einer TM akzeptiert werden, ist genau die Menge aller Sprachen vom Chomsky-Typ 0.

Beweis (konstruktiv), Idee der Konstruktionen:

- Grammatik $\rightarrow$ TM:
  M berechnet für das Eingabewort w (letztes Wort einer Ableitung) eine Ableitung in der Grammatik „rückwärts“.
  (Bei Anwendung verlängernder und verkürzender Regeln wird der aktuelle Bandinhalt geeignet verschoben.)
  TM akzeptiert, wenn Bandinhalt = Startsymbol
- TM $\rightarrow$ Grammatik:
  Grammatik erzeugt „Kandidaten“ w' für jedes Eingabewort w.
  Grammatik-Regeln beschreiben die lokalen Änderungen bei Konfigurationsübergängen der TM.
  (Simulation der Berechnung der TM auf w durch Ableitung in der Grammatik)
  Wird in der Ableitung ein akzeptierender Zustand errreicht, Übersetzung des Kandidaten w' in Terminalwort w.